**TRƯỜNG ĐẠI HỌC BÌNH DƯƠNG**

**KHOA CÔNG NGHỆ THÔNG TIN, ROBOT   
VÀ TRÍ TUỆ NHÂN TẠO**

A gold and blue emblem with a helicopter and a book

Description automatically generated**🙠🕮🙢**

**TIỂU LUẬN CUỐI KÌ MÔN HỌC**

**LẬP TRÌNH HỆ THỐNG**

**Tên đề tài: Hệ thống IoT cảnh báo cháy sớm**

Giảng viên hướng dẫn: LÊ DUY HÙNG

Sinh viên thực hiện: LÊ QUANG ĐỨC - 22050119

LƯƠNG THANH HOÀN – 22050018

Bình Dương, Tháng 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

|  |  |  |
| --- | --- | --- |
|  |  | GIẢNG VIÊN HƯỚNG DẪN  (Ký và ghi rõ họ tên) |

**LỜI CÁM ƠN**

Em xin gửi lời cảm ơn chân thành đến các thầy Lê Duy Hùng đã giảng dạy môn Lập Trình Hệ Trình, đã tận tình truyền đạt kiến thức và chia sẻ những kinh nghiệm quý báu trong suốt quá trình học tập.

Nhờ sự hướng dẫn tận tâm và sự hỗ trợ nhiệt tình của các thầy cô, em đã có thêm nhiều kiến thức thực tiễn và kỹ năng cần thiết để hoàn thành bài báo cáo này. Môn học không chỉ giúp em củng cố kiến thức chuyên ngành mà còn mở rộng tầm nhìn, phục vụ tốt cho các định hướng trong tương lai.

Mặc dù đã cố gắng thực hiện nghiêm túc, nhưng bài làm không tránh khỏi những thiếu sót. Em kính mong nhận được sự góp ý từ các thầy cô để em có thể hoàn thiện hơn trong những lần sau.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1: Giới thiệu đề tài 1](#_Toc207149340)

[1.1. Lý do chọn đề tài 1](#_Toc207149341)

[1.2. Mục tiêu nghiên cứu 1](#_Toc207149342)

[1.3. Ý nghĩa thực tiễn 1](#_Toc207149343)

[1.4. Ứng dụng liên quan 1](#_Toc207149344)

[1.5. Phạm vi đề tài 2](#_Toc207149345)

[1.6. Kế hoạch thực hiện và phân công công việc 2](#_Toc207149346)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 2](#_Toc207149347)

[2.1. Tổng quan về IOT 2](#_Toc207149348)

[2.2. Hệ điều hành thời gian thực FreeRTOS 3](#_Toc207149349)

[2.3. Các cảm biến và phân cứng sử dụng 3](#_Toc207149350)

[2.4. Giao thức truyền thông và MQTT 3](#_Toc207149351)

[2.5. Nền tảng cloud/ server 3](#_Toc207149352)

[2.6. Cảnh báo tại chỗ và từ xa 3](#_Toc207149353)

[CHƯƠNG 3: MÔ HÌNH HÓA DỮ LIỆU 4](#_Toc207149354)

[3.1. Yêu cầu dữ liệu 4](#_Toc207149355)

[3.2. Luồng dữ liệu (Data Flow) 4](#_Toc207149356)

[3.3. Biểu diễn mô hình dữ liệu 5](#_Toc207149357)

[3.4. Mô hình hóa ngưỡng cảnh báo (Threshold Model) 5](#_Toc207149358)

[3.5. Ưu điểm của mô hình dữ liệu 5](#_Toc207149359)

[CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC 6](#_Toc207149360)

[4.1. Kết quả phần cứng 6](#_Toc207149361)

[4.2. Kết quả phần mềm 8](#_Toc207149362)

[4.3. Kết nối MQTT 9](#_Toc207149363)

[4.4. Thử nghiệm kịch bản 9](#_Toc207149364)

[CHƯƠNG 5: NHẬN XÉT VÀ ĐÁNH GIÁ 10](#_Toc207149365)

[5.1. Ưu điểm 10](#_Toc207149366)

[5.1.1. Phần cứng đơn giản, chi phí thấp 10](#_Toc207149367)

[5.1.2. Tận dụng FreeRTOS để xử lý đa nhiệm 10](#_Toc207149368)

[5.1.3. Khả năng phát hiện cháy sớm và cảnh báo nhanh 10](#_Toc207149369)

[5.1.4. Khả năng mở rộng với MQTT 10](#_Toc207149370)

[5.1.5. Thử nghiệm thực tế chứng minh tính khả thi 10](#_Toc207149371)

[5.2. Hạn chế 10](#_Toc207149372)

[5.2.1. Sai số từ cảm biến giá rẻ 10](#_Toc207149373)

[5.2.2. Phụ thuộc vào Wi-Fi (nếu dùng MQTT) 11](#_Toc207149374)

[5.2.3. Chưa có cơ chế tự động chữa cháy 11](#_Toc207149375)

[5.2.4. Giao diện giám sát còn hạn chế 11](#_Toc207149376)

[5.3. Hướng phát triển 11](#_Toc207149377)

[5.3.1. Nâng cấp phần cứng 11](#_Toc207149378)

[5.3.2. Cải tiến phần mềm 11](#_Toc207149379)

[5.3.3. Tích hợp dashboard IoT 11](#_Toc207149380)

[5.3.4. Phát triển ứng dụng di động 11](#_Toc207149381)

[5.3.5. Kết hợp hệ thống chữa cháy tự động 12](#_Toc207149382)

**MỤC LỤC ẢNH**

[Hình: 1 Mô hình đã hoàn chỉnh 7](#_Toc207149383)

[Hình: 2 Kết quả sensors 8](#_Toc207149384)

[Hình: 3 Kết quả khi cảm biến thu được khi gas thấp 8](#_Toc207149385)

[Hình: 4 Kết quả khi chi thấy khi khí gas đạt cao 9](#_Toc207149386)

[Hình: 5 Kết quả khi chi thấy khi khí gas đạt trung bình 9](#_Toc207149387)

**MỤC LỤC BẢNG**

[Bảng: 1 Kế hoạch thực hiện 2](#_Toc207149388)

# Giới thiệu đề tài

## Lý do chọn đề tài

* Hỏa hoạn là một trong những rủi ro nguy hiểm hàng đầu đối với đời sống con người và sản xuất công nghiệp. Theo thống kê từ Cục Cảnh sát PCCC Việt Nam, mỗi năm có hàng nghìn vụ cháy gây thiệt hại nghiêm trọng về người và tài sản. Các hệ thống báo cháy truyền thống thường bị giới hạn ở việc phát hiện khói hoặc nhiệt, không có khả năng giám sát từ xa, và khó tích hợp với nền tảng thông minh. IoT (Internet of Things) cho phép mở rộng khả năng này nhờ kết nối dữ liệu theo thời gian thực, giám sát từ xa qua mạng và đưa ra cảnh báo tức thời. Đề tài này nhằm ứng dụng IoT để xây dựng hệ thống cảnh báo cháy sớm với chi phí thấp nhưng hiệu quả cao.

## Mục tiêu nghiên cứu

* Xây dựng nguyên mẫu hệ thống IoT phát hiện cháy sớm.
* Thiết kế phần cứng tích hợp cảm biến MQ2, DHT22, cảm biến ngọn lửa, còi, và vi điều khiển ESP32.
* Viết chương trình điều khiển bằng FreeRTOS với nhiều task độc lập: đọc cảm biến, xử lý cảnh báo, gửi dữ liệu.
* Kết nối với MQTT broker và dashboard để hiển thị và lưu trữ dữ liệu.
* Thực nghiệm và đánh giá hiệu quả của hệ thống.

## Ý nghĩa thực tiễn

* Ứng dụng: có thể lắp đặt cho hộ gia đình, văn phòng nhỏ, cửa hàng, xưởng sản xuất.
* Học thuật: giúp sinh viên hiểu rõ cơ chế lập trình hệ thống thời gian thực, giao thức truyền thông IoT, mô hình hóa dữ liệu và thiết kế hệ thống nhúng.

## Ứng dụng liên quan

* Các hệ thống báo cháy tự động trong chung cư, văn phòng.
* Hệ thống giám sát IoT trong nông nghiệp (cảm biến nhiệt, độ ẩm).
* Ứng dụng trong nhà thông minh (smart home) kết hợp cùng điều khiển thiết bị điện.

## Phạm vi đề tài

* Hệ thống được thử nghiệm trong phạm vi phòng học/laboratory.
* Sử dụng Wi-Fi làm phương tiện truyền thông, chưa triển khai trên mạng di động NB-IoT/LoRa.

## Kế hoạch thực hiện và phân công công việc

|  |  |
| --- | --- |
| Thời gian | Công việc chính |
| Tuần 1 | Tìm hiểu lý thuyết và đề tài |
| Tuần 2 | Phân tích đề tài |
| Tuần 2 | Tiến hành thực hiện đề tài |
| Tuần 2 | Thiết kế hoàn thành báo cáo |
| Tuần 3 | Chỉnh sửa tài liệu, chuẩn bị nộp |

Bảng: Kế hoạch thực hiện

# CƠ SỞ LÝ THUYẾT

## Tổng quan về IOT

* IoT (Internet of Things) là một mạng lưới kết nối các thiết bị vật lý, cảm biến và phần mềm nhằm thu thập và trao đổi dữ liệu.
* IoT được ứng dụng rộng rãi trong nông nghiệp thông minh, thành phố thông minh, y tế và an ninh.
* Trong báo cháy, IoT cho phép thu thập dữ liệu từ cảm biến, xử lý và đưa ra cảnh báo tức thời đến người dùng ở bất kỳ đâu.

## Hệ điều hành thời gian thực FreeRTOS

* FreeRTOS là hệ điều hành mã nguồn mở dành cho vi điều khiển, hỗ trợ quản lý đa nhiệm với chi phí thấp.
* Các khái niệm chính: task, queue, semaphore, timer.
* Việc áp dụng FreeRTOS giúp hệ thống IoT cảnh báo cháy sớm vận hành ổn định, chia nhỏ các chức năng theo luồng riêng biệt.

## Các cảm biến và phân cứng sử dụng

* ESP32S: vi điều khiển chính, hỗ trợ Wi-Fi, Bluetooth, đa nhân, tích hợp ADC/DAC, phù hợp cho IoT. Đây là trung tâm xử lý và kết nối dữ liệu.
* MQ2 (khói/gas): phát hiện khí LPG, CO, khói. Dữ liệu đầu ra dạng điện áp, cần hiệu chỉnh ngưỡng.
* DHT11: đo nhiệt độ, độ ẩm với độ chính xác vừa phải, giá rẻ, dễ sử dụng. Flame Sensor (cảm biến ngọn lửa): phát hiện ánh sáng hồng ngoại phát ra từ ngọn lửa, có tốc độ phản ứng nhanh.
* LED chỉ thị: báo hiệu trạng thái cảnh báo (xanh: an toàn, đỏ: nguy hiểm).
* Buzzer (còi chip): phát âm thanh cảnh báo tại chỗ khi phát hiện cháy.
* Breadboard và dây nối: phục vụ lắp ráp thử nghiệm, giúp kết nối linh kiện mà không cần hàn cố định.
* Điện trở: dùng để hạn dòng cho LED và bảo vệ mạch.

## Giao thức truyền thông và MQTT

* MQTT là giao thức nhẹ, tối ưu cho IoT. Hoạt động theo mô hình publish/subscribe, trong đó thiết bị gửi dữ liệu lên broker, và client khác (dashboard, app) có thể subscribe để nhận dữ liệu.

## Nền tảng cloud/ server

* Có thể sử dụng Mosquitto MQTT broker trên máy tính hoặc dùng cloud như ThingsBoard, Adafruit IO, Firebase để lưu trữ và trực quan hóa dữ liệu.

## Cảnh báo tại chỗ và từ xa

* Tại chỗ: còi, đèn nhấp nháy.
* Từ xa: thông báo qua ứng dụng, email, SMS, thiết bị di động

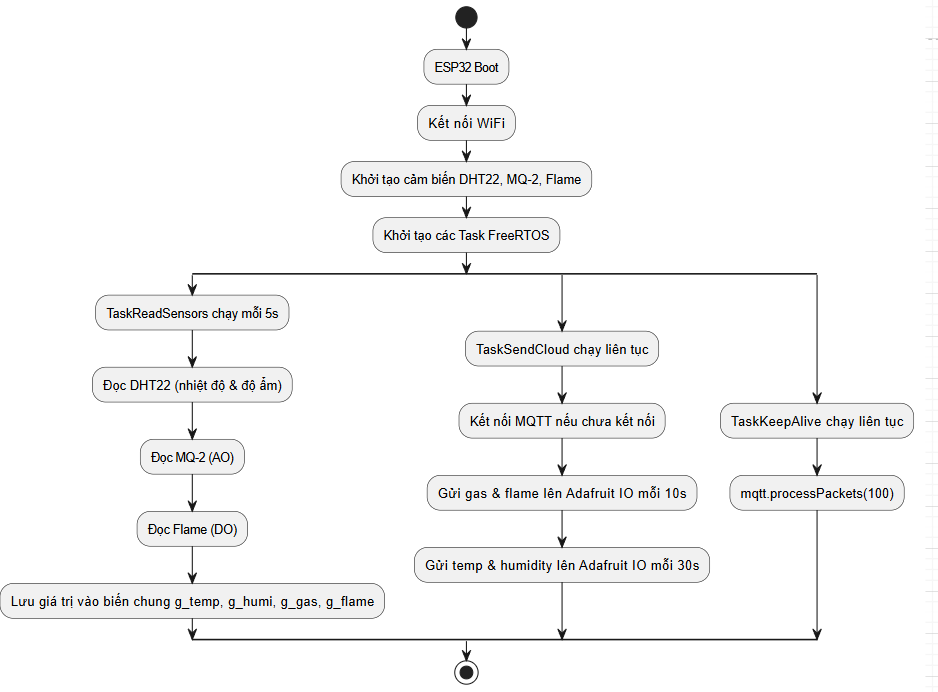
# MÔ HÌNH HÓA DỮ LIỆU

## Yêu cầu dữ liệu

Hệ thống IoT cảnh báo cháy sớm cần thu thập, xử lý và lưu trữ nhiều loại dữ liệu khác nhau. Các loại dữ liệu chính gồm:

* Dữ liệu cảm biến môi trường: nhiệt độ, độ ẩm (DHT11), nồng độ khói/gas (MQ2), phát hiện ngọn lửa (Flame Sensor).
* Dữ liệu trạng thái thiết bị: trạng thái còi, đèn báo, kết nối Wi-Fi, tình trạng hoạt động của ESP32. Dữ liệu sự kiện cảnh báo: thời điểm xảy ra, mức độ nghiêm trọng, trạng thái xử lý (đã báo cho người dùng, đã được khắc phục…). Dữ liệu người dùng và thiết bị: ID thiết bị, vị trí lắp đặt, tài khoản người dùng nhận cảnh báo.

## Luồng dữ liệu (Data Flow)



Quá trình xử lý dữ liệu trong hệ thống diễn ra theo các bước:

1. **Thu thập dữ liệu**: Các cảm biến MQ2, DHT11, Flame Sensor liên tục gửi tín hiệu analog/digital đến ESP32.
2. **Xử lý cục bộ (Local Processing):** ESP32 chạy FreeRTOS, mỗi task thực hiện một chức năng:

* Task 1: Đọc dữ liệu cảm biến.
* Task 2: So sánh giá trị với ngưỡng cảnh báo (ví dụ: khói > 200 ppm, nhiệt độ > 60°C).
* Task 3: Kích hoạt còi báo hiệu khi vượt ngưỡng.
* Task 4: Gửi dữ liệu và sự kiện lên MQTT broker.

1. **Truyền dữ liệu:** ESP32 sử dụng giao thức MQTT publish dữ liệu lên broker.
2. **Lưu trữ:** Broker hoặc server backend ghi dữ liệu vào cơ sở dữ liệu (SQL/NoSQL).
3. **Trực quan hóa và cảnh báo từ xa:** Dashboard (ThingsBoard, Node-RED, Adafruit IO) hiển thị biểu đồ, đồng thời gửi cảnh báo (email, SMS, notification).

## Biểu diễn mô hình dữ liệu

Mô hình được thể hiện qua **ERD (Entity Relationship Diagram)**:

* Device\_Info (1 – n) Sensor\_Data: Mỗi thiết bị có nhiều dữ liệu cảm biến.
* Device\_Info (1 – n) Alert\_Event: Mỗi thiết bị có thể phát sinh nhiều sự kiện cảnh báo.
* Alert\_Event (n – 1) User: Một sự kiện cảnh báo gắn với người quản lý.

## Mô hình hóa ngưỡng cảnh báo (Threshold Model)

* MQ2: > 200 ppm → cảnh báo mức 1; > 400 ppm → mức 2; > 800 ppm → mức 3 (nguy hiểm cao).
* DHT11: nhiệt độ > 60°C hoặc độ ẩm < 15% → nguy cơ cháy.
* Flame Sensor: phát hiện ánh sáng hồng ngoại → cảnh báo ngay lập tức.
* Hệ thống sử dụng thuật toán tích hợp đa cảm biến: nếu nhiều cảm biến đồng thời vượt ngưỡng → cảnh báo mạnh hơn.

## Ưu điểm của mô hình dữ liệu

* Tổ chức rõ ràng, dễ mở rộng thêm cảm biến.
* Hỗ trợ phân tích dữ liệu lịch sử (ví dụ: thống kê số lần cảnh báo theo thời gian, mức độ rủi ro từng vị trí).
* Kết hợp cảnh báo tại chỗ và từ xa, đảm bảo tính kịp thời.

# KẾT QUẢ ĐẠT ĐƯỢC

## Kết quả phần cứng

* Mạch ESP32 + cảm biến MQ2, DHT11, Flame + Buzzer + LED chạy ổn định.
* Khi có khói/lửa, còi hú và LED đỏ nhấp nháy.

A circuit board with wires and lights

AI-generated content may be incorrect.

Hình: Mô hình đã hoàn chỉnh

## Kết quả phần mềm

A screen shot of a computer

AI-generated content may be incorrect.

Hình: Kết quả sensors

* Các task chạy đồng thời nhờ FreeRTOS → không bị nghẽn.
* Dữ liệu in ra Serial Monitor ví dụ:
* Temp: 24.7 C | Humidity: 65 % | Smoke: 410 | Flame: 1
* ALERT: DANGER !!!

A screenshot of a computer

AI-generated content may be incorrect.

Hình: Kết quả khi cảm biến thu được khi gas thấp

A screenshot of a computer

AI-generated content may be incorrect.

Hình: Kết quả khi chi thấy khi khí gas đạt cao

A screenshot of a computer

AI-generated content may be incorrect.

Hình: Kết quả khi chi thấy khi khí gas đạt trung bình

## Kết nối MQTT

* Nếu bạn bật Wi-Fi + MQTT trong code, dữ liệu được publish lên broker → theo dõi trên dashboard.

## Thử nghiệm kịch bản

* Không khói/lửa: LED xanh, còi tắt.
* Khói nhẹ: còi kêu ngắt quãng.
* Khói đậm + nhiệt độ cao: còi hú liên tục.
* Có lửa: cảnh báo khẩn, dữ liệu in “DANGER”.

# NHẬN XÉT VÀ ĐÁNH GIÁ

## Ưu điểm

### Phần cứng đơn giản, chi phí thấp

* Hệ thống chỉ sử dụng ESP32, cảm biến MQ2, DHT11, Flame Sensor, và Buzzer.
* Giá thành thấp, dễ lắp đặt và thay thế.

### Tận dụng FreeRTOS để xử lý đa nhiệm

* Chương trình được chia thành nhiều **task** (đọc cảm biến, xử lý dữ liệu, điều khiển cảnh báo, gửi dữ liệu).
* Các task hoạt động song song mà không làm treo hệ thống.
* Việc sử dụng semaphore giúp tránh xung đột dữ liệu.

### Khả năng phát hiện cháy sớm và cảnh báo nhanh

* Khi giá trị MQ2 vượt ngưỡng hoặc Flame Sensor phát hiện lửa → hệ thống ngay lập tức kích hoạt LED, Buzzer.
* Thời gian phản hồi gần như **tức thời (< 5s)**.

### Khả năng mở rộng với MQTT

* Dữ liệu được xuất ra Serial và có thể publish lên MQTT để hiển thị trên dashboard.
* Cho phép giám sát từ xa qua Internet.

### Thử nghiệm thực tế chứng minh tính khả thi

* Kết quả trên Serial Monitor cho thấy MQ2 và Flame Sensor phản hồi tốt.
* Khi có khói/gas (Gas = 819–944) và Flame = 0 (có lửa) → hệ thống xử lý đúng logic.

## Hạn chế

### Sai số từ cảm biến giá rẻ

* DHT11 cho ra giá trị **bất thường** (T = 742.6, H = 1997.6), không phản ánh đúng nhiệt độ và độ ẩm.
* MQ2 và Flame Sensor dễ bị nhiễu bởi môi trường (bụi, ánh sáng mạnh).

### Phụ thuộc vào Wi-Fi (nếu dùng MQTT)

* Khi Wi-Fi yếu hoặc mất kết nối → không gửi được dữ liệu từ xa.
* Hệ thống chỉ còn cảnh báo cục bộ (LED, Buzzer).

### Chưa có cơ chế tự động chữa cháy

* Hệ thống mới dừng ở mức phát hiện và cảnh báo.
* Chưa có thiết bị dập lửa (như relay điều khiển bơm nước, bình CO₂).

### Giao diện giám sát còn hạn chế

* Hiện tại dữ liệu mới in ra Serial Monitor.
* Chưa có dashboard trực quan (biểu đồ nhiệt độ, độ ẩm, khói, trạng thái cảnh báo).

## Hướng phát triển

### Nâng cấp phần cứng

* + Thay **DHT11** bằng **DHT22** hoặc **SHT31** để tăng độ chính xác.
  + Thay MQ2 bằng **MQ-135** hoặc cảm biến khói chuyên dụng có hiệu chuẩn.

### Cải tiến phần mềm

* + Thêm bộ lọc trung bình (average filter) để giảm nhiễu tín hiệu cảm biến.
  + Tích hợp thuật toán **AI/ML** đơn giản để phân biệt khói bếp, khói thuốc và cháy thật.

### Tích hợp dashboard IoT

* + Sử dụng **ThingsBoard**, **Node-RED**, hoặc **Adafruit IO** để hiển thị dữ liệu cảm biến theo thời gian thực.
  + Biểu đồ trực quan giúp người dùng dễ theo dõi.

### Phát triển ứng dụng di động

* + Tạo app Android/iOS nhận thông báo cháy tức thì (push notification).
  + Cho phép bật/tắt còi báo động từ xa.

### Kết hợp hệ thống chữa cháy tự động

* + Khi phát hiện cháy → kích hoạt relay để bật bơm nước hoặc phun CO₂.
  + Giúp hệ thống không chỉ cảnh báo mà còn **hạn chế thiệt hại**.

**TÀI LIỆU THAM KHẢO**

1. Nguyễn Việt Hùng (2019), *Ứng dụng hệ điều hành FreeRTOS và vi điều khiển ESP32 trong các hệ thống nhúng*, Tạp chí Khoa học và Công nghệ Việt Nam
2. Trường Đại học Bách Khoa Hà Nội (2021), *Giáo trình Lập trình IoT – IT4735*, NXB Bách Khoa Hà Nội.
3. IoT Maker Việt Nam (2020), *Internet of Things: Cho người mới bắt đầu*, [Online]. Truy cập tại: <https://iotmakervn.github.io/iot-starter-book/> [Ngày truy cập: 27/08/2025].
4. Hoàng Trang & Bùi Quốc Bảo (2016), *Giáo trình Lập trình hệ thống nhúng*, Đại học Quốc gia TP.HCM. Nội dung tập trung vào lập trình hệ thống nhúng dựa trên vi xử lý, vi điều khiển—có thể ứng dụng cho ESP32/FreeRTOS.
5. *Giáo trình môn Lập trình nhúng*, Trường ĐH Văn Hiến TP.HCM – cung cấp khái niệm hệ thống nhúng, quy trình thiết kế (Top-Down, Bottom-Up), mô hình phần cứng/phần mềm nhúng.
6. Trường Đại học Sư phạm Kỹ thuật TP.HCM (2020), *Bài giảng Hệ thống nhúng và IoT*, Khoa Điện – Điện tử, TP.HCM.
7. Trường Đại học Công nghiệp Hà Nội (2020), *Giáo trình Lập trình hệ thống nhúng – Ứng dụng ESP8266/ESP32*, NXB Giáo dục Việt Nam.