TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TP. HỒ CHÍ MINH

BỘ MÔN ĐIỆN - ĐIỆN TỬ

BÁO CÁO MÔN HỌC

THIẾT KẾ HỆ THỐNG IOT

ĐỀ TÀI: KIỂM TRA ÁP SUẤT BÁNH VÀ CẢNH BÁO KHI XE VƯỢT QUÁ TỐC ĐỘ

Giảng viên hướng dẫn: TS. Lê Tiến Lộc

SVTH: Bùi Đức Hùng - 6251020055

Lớp : CQ.62.KS.ĐT&THCN

Hồ Chí Minh, tháng 12 năm 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TP. HỒ CHÍ MINH

BỘ MÔN ĐIỆN - ĐIỆN TỬ

BÁO CÁO MÔN HỌC

THIẾT KẾ HỆ THỐNG IOT

ĐỀ TÀI: KIỂM TRA ÁP SUẤT BÁNH VÀ CẢNH BÁO KHI XE VƯỢT QUÁ TỐC ĐỘ

Giảng viên hướng dẫn: TS. Lê Tiến Lộc

Nhóm SVTH: Nguyễn Thị Thu Thủy - 6251020018

Ngô Diệp Thái Châu - 6251020001

Bùi Đức Hùng - 6251020055

Lớp : CQ.62.KS.ĐT&THCN

Hồ Chí Minh, tháng 12 năm 2025

LỜI CẢM ƠN

Trước hết, nhóm xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô Khoa Điện – Điện tử đã tận tình giảng dạy, truyền đạt cho chúng em những kiến thức nền tảng và chuyên sâu về vi điều khiển, cảm biến, lập trình nhúng và IoT, giúp chúng em có đủ khả năng vận dụng vào quá trình nghiên cứu và triển khai đề tài này.

Đặc biệt, nhóm xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn – người đã tận tâm chỉ bảo, góp ý từng chi tiết kỹ thuật, định hướng phương pháp làm việc khoa học, giúp nhóm hoàn thiện mô hình và nội dung báo cáo đúng tiến độ.

Mặc dù nhóm đã cố gắng hết sức, song do hạn chế về thời gian, kinh phí và điều kiện thực tế, đề tài không tránh khỏi những thiếu sót. Nhóm rất mong nhận được những ý kiến đóng góp, nhận xét của quý thầy cô để hoàn thiện hơn trong các nghiên cứu sau này.

Cuối cùng, nhóm xin kính chúc quý thầy cô sức khỏe, thành công và luôn tận tâm với sự nghiệp trồng người.

Xin chân thành cảm ơn!

LỜI MỞ ĐẦU

Trong bối cảnh công nghiệp hóa và hiện đại hóa, phương tiện giao thông ngày càng trở nên phổ biến và đóng vai trò thiết yếu trong đời sống con người. Tuy nhiên, cùng với sự gia tăng về số lượng phương tiện, tình trạng tai nạn giao thông cũng trở nên đáng báo động. Một trong những nguyên nhân chủ yếu gây mất an toàn khi vận hành xe là áp suất lốp không phù hợp và việc chạy xe vượt quá tốc độ giới hạn cho phép.

Trước thực tế đó, việc nghiên cứu và xây dựng hệ thống giám sát áp suất bánh và cảnh báo tốc độ mang ý nghĩa thiết thực, nhằm nâng cao an toàn giao thông, giảm thiểu tai nạn, tiết kiệm nhiên liệu và kéo dài tuổi thọ phương tiện. Đề tài "Kiểm tra áp suất bánh và cảnh báo khi xe vượt quá tốc độ (tích hợp GPS – Google Maps API)" được thực hiện với mục tiêu thiết kế một hệ thống thông minh có khả năng đo, hiển thị và cảnh báo tự động, đồng thời kết nối dữ liệu lên nền tảng Google Maps để theo dõi vị trí phương tiện theo thời gian thực.

Đề tài được xây dựng dựa trên việc ứng dụng vi điều khiển ESP32, kết hợp với các cảm biến như BMP180 (đo áp suất), Encoder – GPS NEO-6M (đo tốc độ và định vị), cùng các khối hiển thị – cảnh báo như màn hình OLED và LED. Toàn bộ dữ liệu được thu thập, xử lý và truyền qua kết nối Wi-Fi, đáp ứng xu hướng phát triển của các hệ thống IoT trong giao thông thông minh (Smart Mobility).

Phương pháp thực hiện bao gồm: nghiên cứu lý thuyết, thiết kế phần cứng – phần mềm, mô phỏng và thử nghiệm, từ đó đánh giá độ chính xác và khả năng ứng dụng thực tế của hệ thống.

Nội dung báo cáo được chia thành 5 chương:

- Chương 1: Giới thiệu chung về đề tài, lý do chọn đề tài và mục tiêu nghiên cứu.
- Chương 2: Cơ sở lý thuyết và nguyên lý hoạt động của các cảm biến, module và vi điều khiển.
- Chương 3: Thiết kế phần cứng, phần mềm và hoạt động tổng thể của hệ thống.
- Chương 4: Kết quả thử nghiệm, đánh giá và phân tích hiệu năng hệ thống.
- Chương 5: Kết luận và hướng phát triển trong tương lai.

MỤC LỤC

LỜI CẢM ƠN 3
LỜI MỞ ĐẦU 4
DANH MỤC BẢNG BIỂU 7
DANH MỤC HÌNH ẢNH 8
CHƯƠNG 1: GIỚI THIỆU CHUNG 1
1.1. Lý do chọn đề tài 1
1.2. Mục tiêu đề tài. 1
1.3. So sánh giải pháp. 2
1.4. Giới hạn đề tài. 4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6
2.1. Tổng quan về hệ thống kiểm tra áp suất và cảnh báo tốc độ. 6
2.2. Cơ sở lý thuyết về đo áp suất. 7
2.2.1. Khái niệm áp suất. 7
2.2.2. Cảm biến áp suất BMP180/BMP280. 7
2.3. Cơ sở lý thuyết về đo tốc độ. 8
2.3.1. Nguyên lý đo tốc độ bằng Encoder. 8
2.3.2. Nguyên lý đo tốc độ bằng GPS. 9
2.4. Cơ sở lý thuyết về vi điều khiển ESP32. 9
2.4.1. Giao tiếp I2C 10
2.4.2. Kết nối Wi-Fi 11
2.5. Cơ sở lý thuyết về hiển thị và cảnh báo. 11
2.5.1. Màn hình hiển thị (LCD/OLED). 11
2.5.2. Đèn LED cảnh báo. 12
2.6. Cơ sở lý thuyết về Google Maps API 13

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG 14
3.1. Sơ đồ khối hệ thống. 14
3.2. Nguyên lý hoạt động. 16
3.3.1. Truyền dữ liệu từ ESP32 đến Server 18
CHƯƠNG 4: KẾT QUẢ VÀ ĐÁNH GIÁ HỆ THỐNG 23
4.1. Kết quả hiển thị và hoạt động hệ thống. 23
4.2. Hiển thị vị trí trên bản đồ Google Maps. 24
4.3. Hạn chế. 27
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 28
5.1. Kết luận. 28
5.2. Hướng phát triển của đề tài 28
5.3. Kết luận chung. 29
PHỤ LỤC 30
TÀI LIỆU THAM KHẢO 41

DANH MỤC BẢNG BIỂU

Bảng 1. So sánh các dự án đo áp suất và cảnh báo tốc độ 3
Bảng 2. Công nghệ sử dụng 18
Bảng 3. Logic so sánh tốc độ 21
Bảng 4. Dữ liệu giả lập theo các vị trí khác nhau. 24

DANH MỤC HÌNH ẢNH

Hình 1. Sơ đồ khối hệ thống. 7
Hình 2. Module cảm biến áp suất BMP180 8
Hình 3. Module encoder 9
Hình 4. Module GPS NEO-6M 9
Hình 5. ESP32 10
Hình 6. Màn hình LCD OLED 0.96 12
Hình 7. LED 12
Hình 8. Sơ đồ khối hệ thống 14
Hình 9. Schematic 14
Hình 10. PCB 15
Hình 11. Lưu đồ giải thuật 17
Hình 14. Server nhận và server gửi 19
Hình 15. Server truy vấn giới hạn tốc độ 20
Hình 16. Khai báo Fallback và điều kiện fallback 20
Hình 17. Logic so sánh tốc độ 21
Hình 18. Lưu dữ liệu vào MongoDB 21
Hình 19. Kết quả lưu dữ liệu tại MongoDB 22
Hình 20. Broadcast realtime cho Web Dashboard 22
Hình 21. Màn hình OLED hiển thị áp suất và tốc độ thực tế 23
Hình 22. Màn hình OLED hiển thị cảnh báo 25
Hình 23. Server gửi dữ liệu 25
Hình 24. Lịch sử hoạt động của xe 26
Hình 25.Web cập nhật vị trí, tốc độ, áp suất 26
Hình 26. Vẽ biểu đồ cảnh báo 27

CHƯƠNG 1: GIỚI THIỆU CHUNG

Chương này giới thiệu tổng quan về đề tài "Kiểm tra áp suất bánh và cảnh báo khi xe vượt quá tốc độ", bao gồm *lý do chọn đề tài, mục tiêu, phạm vi – đối tượng, phương pháp nghiên cứu, linh kiện được lựa chọn, ý nghĩa thực tiễn và bố cục báo cáo.*

1.1. Lý do chọn đề tài

Trong thời đại công nghiệp hóa và hiện đại hóa, nhu cầu di chuyển bằng phương tiện cá nhân ngày càng tăng. Tuy nhiên, tai nạn giao thông vẫn là một vấn đề nhức nhối, trong đó nguyên nhân từ áp suất lốp không đúng chuẩn và việc chạy xe quá tốc độ chiếm tỉ lệ đáng kể.

Xuất phát từ thực trạng tai nạn giao thông tại Việt Nam: Theo Ủy ban An toàn Giao thông Quốc gia, năm 2024 ghi nhận hơn 11.600 vụ tai nạn, trong đó khoảng 35% liên quan đến nguyên nhân kỹ thuật của phương tiện, như áp suất lốp không đúng chuẩn, mòn lốp**,** hoặc chạy quá tốc độ an toàn. Nghiên cứu của Bridgestone Việt Nam (2023) cũng cho thấy, 70% xe máy lưu thông tại đô thị có lốp bị non từ 10–20% so với khuyến nghị**,** làm tăng mức tiêu hao nhiên liệu khoảng 5–10% và rút ngắn tuổi thọ lốp hơn 30%**.** Ngoài ra, vượt tốc độ 10 km/h có thể làm tăng nguy cơ va chạm nghiêm trọng lên 40% (theo Tổ chức Y tế Thế giới – WHO).

Áp suất bánh xe thấp có thể làm tăng ma sát, gây nóng lốp, giảm tuổi thọ và tăng tiêu hao nhiên liệu; trong khi đó áp suất cao dễ dẫn đến nổ lốp, đặc biệt khi xe chạy ở tốc độ cao. Mặt khác, việc không kiểm soát tốc độ làm tăng nguy cơ tai nạn và vi phạm luật giao thông.

Từ thực tế đó, đề tài hướng tới xây dựng một hệ thống giám sát thông minh có khả năng đo và cảnh báo áp suất lốp – tốc độ xe theo thời gian thực, giúp người lái nhận biết sớm rủi ro**,** đảm bảo an toàn khi di chuyển, và nâng cao ý thức tuân thủ luật giao thông.

1.2. Mục tiêu đề tài.

Xây dựng hệ thống giám sát thông minh có khả năng đo, hiển thị và cảnh báo các thông số quan trọng của phương tiện gồm áp suất bánh xe và tốc độ di chuyển, đồng thời ghi nhận vị trí qua GPS và hiển thị trên bản đồ Google Maps thông qua API. Hệ thống giúp

người lái phát hiện sớm các nguy cơ mất an toàn**,** nâng cao ý thức tuân thủ tốc độ và tối ưu hiệu quả vận hành phương tiện**.**

1**.** Thiết kế và thi công mạch cảm biến áp suất bánh xe, sử dụng cảm biến áp suất analog

2. Xây dựng hệ thống đo tốc độ xe:

- Tính toán từ số vòng quay bánh xe (cảm biến Hall/encoder)
- Sử dụng tốc độ GPS tích hợp trong module định vị.

3. Thiết kế giao diện hiển thị thông số áp suất, tốc độ và trạng thái cảnh báo trên màn hình LCD OLED , bảo đảm dễ quan sát, trực quan.

4**.** Lập trình chức năng cảnh báo tự động bằng đèn LED hoặc coi khi:

- Áp suất bánh xe thấp hơn hoặc cao hơn ngưỡng an toàn.
- Xe vượt quá tốc độ giới hạn được cài đặt sẵn.

5. Tích hợp module GPS để thu thập tọa độ (vĩ độ, kinh độ) theo thời gian thực.

6. Kết nối mạng Wi-Fi qua ESP32 và sử dụng Google Maps API để:

- Gửi vị trí và dữ liệu cảnh báo lên nền tảng web hoặc điện thoại**.**
- Hiển thị vị trí và hành trình xe trên bản đồ.
- Đảm bảo hệ thống hoạt động ổn định**,** độ chính xác cao**,** dễ lắp đặt trên các phương tiện nhỏ (xe máy, xe mô hình).

1.3. So sánh giải pháp.

Tiêu chí / Giải pháp	**Hệ thống TPMS thương mại (ô tô)**	**Đề tài sinh viên trước (đo áp suất + tốc độ cơ bản)**	**Đề tài hiện tại (tích hợp GPS – Google Maps)**
Cảm biến áp suất	Dùng cảm biến không dây riêng cho từng bánh, tín hiệu RF 433 MHz, giá cao (~800 k–2 triệu/bộ)	Dùng cảm biến analog MPX5700AP hoặc MPS20N0040D, cần mạch khuếch đại	Cảm biến BMP180/BMP280 – đo áp suất chính xác, giá rẻ, dễ giao tiếp I²C với ESP32
Đo tốc độ	Sử dụng cảm biến ABS hoặc dữ liệu từ ECU (CAN bus)	Đếm xung từ Encoder gắn trục bánh → tính tốc độ tuyến tính	Kết hợp Encoder + GPS NEO-6M để vừa đo tốc độ cơ học, vừa có vận tốc và tọa độ thực tế

Tiêu chí / Giải pháp	Hệ thống TPMS thương mại (ô tô)	Đề tài sinh viên trước (đo áp suất + tốc độ cơ bản)	Đề tài hiện tại (tích hợp GPS – Google Maps)
Định vị / Theo dõi	Có GPS tích hợp sẵn trong hệ thống giám sát xe (telematics)	Không có chức năng định vị	Module GPS NEO-6M – độ chính xác 2–5 m, dễ dùng, hỗ trợ dữ liệu NMEA (vĩ độ, kinh độ, tốc độ)
Cảnh báo	Hiển thị trên màn hình xe hoặc điện thoại qua sóng RF/Bluetooth	Đèn LED, còi buzzer khi vượt ngưỡng áp suất	LED + còi + hiển thị LCD, có thể mở rộng gửi dữ liệu cảnh báo qua Internet (Wi-Fi)
Hiển thị dữ liệu	Màn hình tích hợp trong xe hoặc điện thoại (app)	Màn hình LCD 16x2 hoặc OLED 0.96" đơn giản	LCD 16x2 / OLED I²C hiển thị đồng thời áp suất, tốc độ, trạng thái cảnh báo
Bộ xử lý trung tâm	Chip chuyên dụng TPMS, MCU ARM, có kết nối CAN bus	Arduino Uno / Nano – đơn giản nhưng hạn chế bộ nhớ, không Wi-Fi	ESP32 – mạnh, tích hợp Wi-Fi, Bluetooth, đủ chân I/O và bộ nhớ để xử lý GPS + cảm biến
Giao tiếp và kết nối	RF/Bluetooth, kết nối ứng dụng điện thoại	Không có kết nối ngoài	Wi-Fi (ESP32) – dùng Google Maps API để gửi và hiển thị vị trí trên bản đồ
Chi phí hệ thống	1 – 3 triệu đồng / bộ	400 – 600 ngàn đồng	~700 – 900 ngàn đồng (bao gồm GPS, LCD, cảm biến BMP, ESP32)
Tính mở rộng	Khó can thiệp, hệ thống đóng	Cơ bản, không kết nối mạng	Dễ mở rộng: thêm IoT dashboard, cảnh báo qua Telegram / app di động

Bảng 1. So sánh các dự án đo áp suất và cảnh báo tốc độ

Trong quá trình nghiên cứu và so sánh các giải pháp hiện có, nhóm nhận thấy cấu hình hệ thống được lựa chọn đáp ứng tốt các yêu cầu về độ chính xác, tính ổn định, khả năng mở rộng và chi phí hợp lý. Cụ thể:

- Cảm biến BMP180/BMP280: được chọn vì có độ chính xác cao trong dải đo áp suất từ 300–1100 hPa, sai số thấp (±1 hPa), kích thước nhỏ gọn và giao tiếp chuẩn I²C, giúp dễ dàng kết nối và xử lý dữ liệu với vi điều khiển. So với các cảm biến tương tự như MPX5700AP, BMP180/BMP280 có ưu thế về kích thước, độ nhạy và mức tiêu thụ điện năng.
- Module Encoder: được sử dụng để xác định tốc độ quay của bánh xe, từ đó tính toán vận tốc tuyến tính của phương tiện. Giải pháp này đảm bảo khả năng đo tốc độ độc

lập với tín hiệu GPS, đặc biệt hữu ích trong các điều kiện môi trường che khuất hoặc mất tín hiệu vệ tinh (ví dụ: hầm, khu vực đô thị dày đặc).

- Module GPS NEO-6M: có khả năng thu nhận tọa độ (vĩ độ, kinh độ) và vận tốc tức thời với sai số chỉ khoảng 2–5 m, tốc độ cập nhật dữ liệu 1 Hz đến 10 Hz. Module này được tích hợp nhằm xác định vị trí phương tiện trong thời gian thực và truyền dữ liệu định vị lên nền tảng Google Maps thông qua API, phục vụ mục tiêu mở rộng hệ thống theo hướng giám sát hành trình thông minh.
- Vi điều khiển ESP32: được lựa chọn nhờ hiệu năng xử lý mạnh mẽ, tích hợp Wi-Fi và Bluetooth sẵn có, cùng khả năng tương thích với nhiều cảm biến và giao thức truyền thông. So với Arduino Uno, ESP32 có bộ nhớ lớn hơn, tốc độ xử lý nhanh hơn và hỗ trợ kết nối IoT linh hoạt, phù hợp cho các ứng dụng giám sát thời gian thực và mở rộng về sau.
- Màn hình LCD/OLED và đèn LED cảnh báo: đảm nhiệm vai trò hiển thị trực quan các thông số đo lường (áp suất, tốc độ, trạng thái cảnh báo) và thông báo tức thời cho người lái khi phát hiện điều kiện bất thường. Đây là giải pháp đơn giản nhưng hiệu quả trong việc tăng tính tương tác và an toàn của hệ thống.

=> Tổng thể, cấu hình phần cứng này cho phép hệ thống đạt được sự cân bằng tối ưu giữa tính năng – độ chính xác – chi phí, đồng thời đảm bảo tính khả thi khi triển khai thực tế và khả năng mở rộng thành mô hình giám sát phương tiện thông minh (TPMS tích hợp GPS tracking) trong các nghiên cứu hoặc ứng dụng tương lai.

1.4. Giới hạn đề tài.

Đề tài "Kiểm tra áp suất bánh và cảnh báo khi xe vượt quá tốc độ (tích hợp GPS – Google Maps API)" được thực hiện trong phạm vi mô hình thử nghiệm, vì vậy có một số giới hạn như sau:

- Giới hạn về phần cứng:
  - Hệ thống chỉ được mô phỏng hoặc triển khai trên xe mô hình chứ chưa áp dụng trực tiếp trên ô tô thực tế.

- Cảm biến áp suất sử dụng loại thông dụng đo áp suất tương đối chứ chưa đạt chuẩn TPMS (Tire Pressure Monitoring System) thương mại.
- Module GPS dùng loại phổ thông có độ chính xác khoảng 2–5 mét, nên dữ liệu vị trí có thể sai lệch trong môi trường đô thị.

- Giới hạn về phần mềm:
  - Dữ liệu vị trí và cảnh báo chỉ được gửi và hiển thị trên nền tảng Google Maps thông qua API, chưa có cơ sở dữ liệu lưu trữ lịch sử hành trình lâu dài.
- Giới hạn về tính năng hoạt động:
  - Hệ thống mới chỉ cảnh báo cục bộ bằng đèn LED, chưa có thông báo qua điện thoại di động (SMS, Telegram, app).
  - Ngưỡng áp suất và tốc độ giới hạn được thiết lập thủ công trong chương trình, chưa có giao diện tùy chỉnh linh hoạt.
  - Chưa xét đến ảnh hưởng của nhiệt độ môi trường đến sai số cảm biến áp suất.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Chương này trình bày các cơ sở lý thuyết liên quan đến hoạt động của cảm biến, module GPS, vi điều khiển ESP32 và hệ thống hiển thị – cảnh báo.Những kiến thức này là nền tảng để nhóm thiết kế phần cứng, lập trình phần mềm và xây dựng hệ thống hoàn chỉnh được.

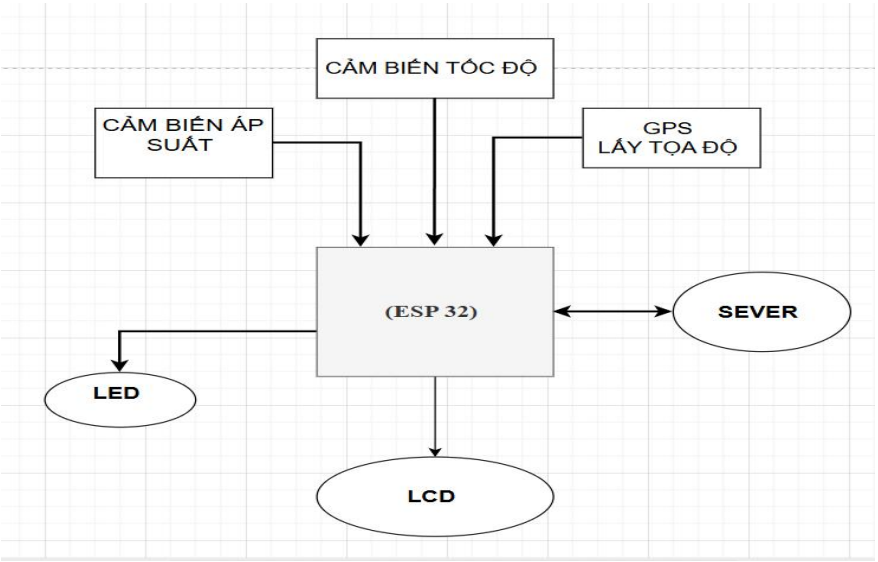
2.1. Tổng quan về hệ thống kiểm tra áp suất và cảnh báo tốc độ.

Hệ thống kiểm tra áp suất bánh và cảnh báo tốc độ là một mô hình giám sát an toàn phương tiện có khả năng thu thập, xử lý và hiển thị các thông số quan trọng trong quá trình vận hành xe.

Các thông số chính bao gồm:

- Áp suất bánh xe, phản ánh tình trạng lốp và ảnh hưởng trực tiếp đến độ bám đường, quãng đường phanh, và tiêu hao nhiên liệu.
- Tốc độ di chuyển, cho phép phát hiện và cảnh báo khi vượt ngưỡng giới hạn cho phép.
- Vị trí GPS, hỗ trợ theo dõi hành trình và xác định vị trí phương tiện khi xảy ra cảnh báo.

Hệ thống hoạt động dựa trên nguyên tắc đo – xử lý – hiển thị – cảnh báo, trong đó cảm biến thu thập tín hiệu đầu vào, vi điều khiển xử lý dữ liệu và xuất kết quả ra màn hình, đồng thời kích hoạt các tín hiệu cảnh báo khi cần thiết.



Hình 1. Sơ đồ khối hệ thống.

2.2. Cơ sở lý thuyết về đo áp suất.

2.2.1. Khái niệm áp suất.

Áp suất là đại lượng vật lý đặc trưng cho lực tác dụng vuông góc lên một đơn vị diện tích, được xác định theo công thức:

$$P = \frac{F}{A}$$

Trong đó:

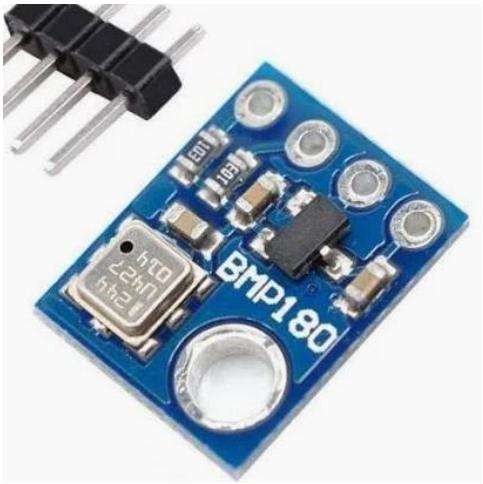
- P: Áp suất (Pa hoặc bar)
- F: Lực tác dụng (N)
- A: Diện tích chịu lực (m^2)

Trong hệ thống đo áp suất bánh xe, cảm biến sẽ chuyển đổi áp suất khí trong lốp thành tín hiệu điện áp tương ứng để vi điều khiển đọc và xử lý.

2.2.2. Cảm biến áp suất BMP180/BMP280.

BMP180/BMP280 là cảm biến áp suất kỹ thuật số của hãng Bosch, hoạt động theo nguyên lý áp điện trở (piezoresistive): khi áp suất không khí tác động lên màng cảm biến, điện trở bên trong thay đổi, tạo ra sự biến thiên điện áp.

Một bộ ADC tích hợp bên trong cảm biến sẽ chuyển đổi tín hiệu tương tự này sang tín hiệu số, truyền qua giao tiếp I^2C hoặc SPI đến vi điều khiển.



Hình 2. Module cảm biến áp suất BMP180

Đặc điểm chính:

- Dải đo: 300 – 1100 hPa (tương đương độ cao -500 m đến 9000 m so với mực nước biển).
- Sai số trung bình: ±1 hPa (≈ ±0.1 m).
- Điện áp hoạt động: 1.8 – 3.6 V.
- Tích hợp cảm biến nhiệt độ để bù sai số nhiệt.

Cảm biến BMP180/BMP280 được sử dụng để đo áp suất tương đối trong bánh xe mô hình, cho độ chính xác cao và dễ kết nối với ESP32.

2.3. Cơ sở lý thuyết về đo tốc độ.

2.3.1. Nguyên lý đo tốc độ bằng Encoder.

Encoder quay (Rotary Encoder) là cảm biến biến đổi chuyển động quay của trục bánh xe thành xung điện. Mỗi vòng quay tạo ra một số xung nhất định, từ đó có thể tính tốc độ tuyến tính:

$$v = \frac{N \times C}{t} \quad (1)$$

Trong đó:

- v: tốc độ (m/s)
- N: số xung đếm được trong thời gian ttt
- C: chu vi bánh xe (m)

Hình 3. Module encoder

Giá trị tốc độ sau đó được chuyển sang đơn vị km/h để hiển thị và so sánh với giới hạn tốc độ cài đặt.

2.3.2. Nguyên lý đo tốc độ bằng GPS.

Module GPS NEO-6M ngoài chức năng định vị còn có thể trích xuất tốc độ tức thời dựa trên dữ liệu quỹ đạo và dịch chuyển tọa độ theo thời gian.

Thông tin được truyền qua cổng UART ở định dạng chuỗi NMEA 0183 – đặc biệt câu lệnh câu lệnh $GPRMC (Recommended Minimum Specific GPS/Transit Data) chứa vận tốc (tính bằng knot, 1 knot ≈ 1.852 km/h) và tọa độ GPS.

Hình 4. Module GPS NEO-6M

Trong đề tài, GPS được dùng song song với Encoder, nhằm:

- Kiểm chứng độ chính xác của phép đo tốc độ.
- Lấy vận tốc thực và vị trí xe để gửi lên Google Maps API khi xảy ra cảnh báo.

2.4. Cơ sở lý thuyết về vi điều khiển ESP32.

ESP32 là vi điều khiển hiệu năng cao được phát triển bởi Espressif Systems, tích hợp sẵn Wi-Fi và Bluetooth, phù hợp cho các ứng dụng IoT (Internet of Things). Thiết bị được thiết kế dựa trên kiến trúc dual-core Tensilica Xtensa LX6 với tốc độ xử lý lên đến 240

MHz, bộ nhớ RAM 520 KB, và flash từ 4 MB đến 16 MB, đáp ứng tốt các yêu cầu xử lý dữ liệu cảm biến, hiển thị và truyền thông không dây trong thời gian thực.

ESP32 hỗ trợ nhiều giao thức truyền thông ngoại vi như I²C, SPI, UART, ADC, PWM, GPIO, đặc biệt là hai thành phần nổi bật trong đề tài này: giao tiếp I²C và kết nối Wi-Fi.

Hình 5. ESP32

2.4.1. Giao tiếp I2C

I²C (Inter-Integrated Circuit) là chuẩn truyền thông nối tiếp hai dây, gồm:

- SDA (Serial Data Line): đường truyền dữ liệu.
- SCL (Serial Clock Line): đường xung nhịp điều khiển.

Cho phép nhiều thiết bị (multi-master, multi-slave) cùng kết nối trên hai đường dây chung, giúp giảm số lượng chân kết nối trên vi điều khiển. ESP32 hỗ trợ hai cổng I²C độc lập (I²C0 và I²C1), có thể cấu hình trên bất kỳ chân GPIO nào, với tốc độ truyền tối đa lên tới 400 kHz (Fast Mode).

Trong đề tài, I²C được sử dụng làm giao thức chính để giao tiếp giữa ESP32 với các cảm biến và màn hình hiển thị, cụ thể:

- Cảm biến BMP180/BMP280 truyền dữ liệu áp suất và nhiệt độ.
- Màn hình OLED/LCD nhận dữ liệu để hiển thị giá trị áp suất, tốc độ, trạng thái cảnh báo.

- Kích hoạt LED khi vượt ngưỡng.
- Truyền dữ liệu vị trí và cảnh báo lên nền tảng Google Maps thông qua Wi-Fi.

Nhờ đặc tính ổn định, chống nhiễu tốt và lập trình đơn giản, I²C giúp hệ thống hoạt động đồng bộ và tiết kiệm tài nguyên phần cứng so với các chuẩn truyền thông khác như UART hoặc SPI.

2.4.2. Kết nối Wi-Fi

Một trong những ưu điểm nổi bật của ESP32 là khả năng tích hợp mô-đun Wi-Fi chuẩn IEEE 802.11 b/g/n, cho phép thiết bị kết nối trực tiếp với mạng Internet hoặc mạng nội bộ (LAN) mà không cần phần cứng bổ sung.

Trong đề tài, Wi-Fi đóng vai trò truyền dữ liệu đo được (áp suất, tốc độ, tọa độ GPS) từ ESP32 lên nền tảng trực tuyến để hiển thị trên bản đồ Google Maps thông qua API hoặc giao thức HTTP/MQTT.

Cơ chế này giúp hệ thống mở rộng từ mô hình giám sát cục bộ thành hệ thống giám sát thông minh thời gian thực, có khả năng theo dõi phương tiện từ xa.

Các ưu điểm của việc sử dụng Wi-Fi trên ESP32:

- Tốc độ truyền cao (lên đến 150 Mbps) – phù hợp cho dữ liệu thời gian thực.
- Phạm vi phủ sóng rộng, dễ kết nối với router hoặc điểm phát di động.
- Hỗ trợ bảo mật WPA/WPA2, đảm bảo an toàn dữ liệu.
- Dễ dàng tích hợp các dịch vụ web như Google Maps API, Firebase, Thingspeak, hoặc Blynk.

2.5. Cơ sở lý thuyết về hiển thị và cảnh báo.

2.5.1. Màn hình hiển thị (LCD/OLED).

Màn hình LCD OLED 0.96 dùng để hiển thị:

- Áp suất bánh xe đo được.
- Tốc độ di chuyển hiện tại.
- Trạng thái cảnh báo (bình thường / vượt ngưỡng).

- OLED hoạt động trên giao tiếp I²C, tiêu thụ ít điện năng và cho độ tương phản cao, phù hợp với không gian hiển thị hạn chế.

Hình 6. Màn hình LCD OLED 0.96

2.5.2. Đèn LED cảnh báo.

LED được kích hoạt khi:

- Áp suất thấp hơn hoặc cao hơn giá trị cài đặt.
- Tốc độ vượt quá giới hạn an toàn.

Các phần tử này giúp người lái nhận biết nhanh tình trạng xe, ngay cả khi không nhìn trực tiếp vào màn hình hiển thị.

Hình 7. LED

2.6. Cơ sở lý thuyết về Google Maps API

Google Maps API là giao diện lập trình ứng dụng cho phép hiển thị bản đồ và đánh dấu vị trí theo tọa độ GPS.

Trong đề tài, dữ liệu GPS (vĩ độ, kinh độ) được ESP32 gửi qua HTTP request hoặc MQTT đến web server, sau đó hiển thị vị trí xe trên bản đồ Google Maps.

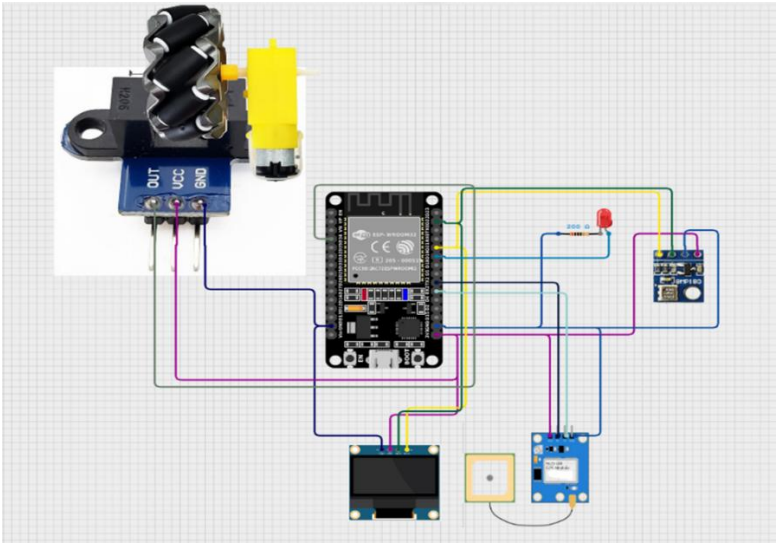
Trong đề tài được dùng để:

- Theo dõi vị trí xe theo thời gian thực.
- Xác định vị trí tại thời điểm cảnh báo.
- Mở rộng hệ thống theo hướng IoT Tracking – TPMS thông minh.

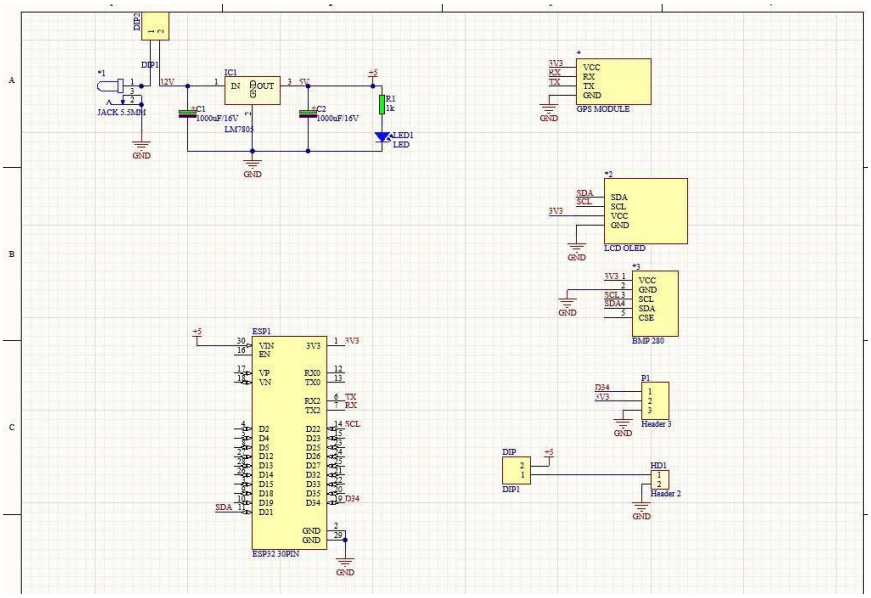
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

Chương này trình bày quá trình thiết kế phần cứng, phần mềm và nguyên lý hoạt động của hệ thống ***"Kiểm tra áp suất bánh và cảnh báo khi xe vượt quá tốc độ".*** *Mục tiêu của chương là mô tả rõ ràng mối liên hệ giữa các module (cảm biến – vi điều khiển – hiển thị – cảnh báo) và cách chúng phối hợp để tạo thành một hệ thống giám sát hoàn chỉnh.*

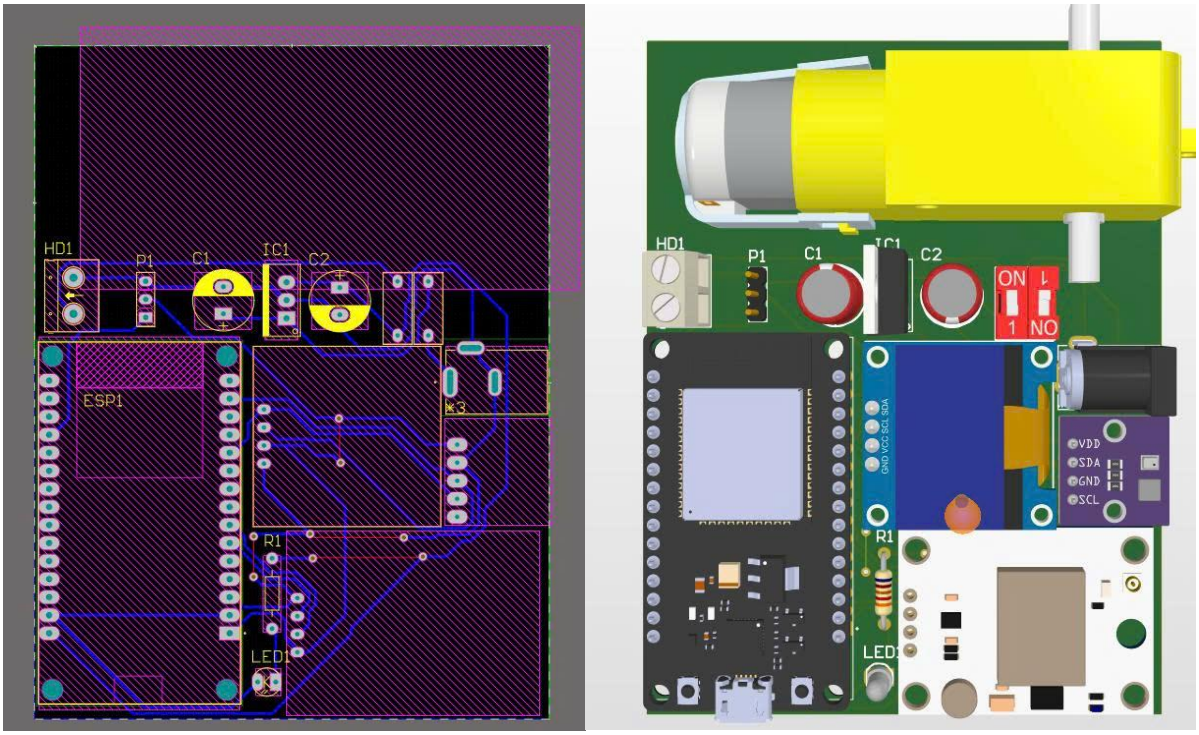
3.1. Sơ đồ khối hệ thống.



Hình 8. Sơ đồ khối hệ thống



Hình 9. Schematic



Hình 10. PCB

Hệ thống được chia thành 3 khối chính:

1. Khối cảm biến (Input)
   - Cảm biến áp suất BMP180: thu nhận giá trị áp suất trong bánh xe và gửi dữ liệu qua giao tiếp I²C đến ESP32.
   - Module Encoder: đếm xung quay từ bánh xe để tính tốc độ.
   - Module GPS NEO-6M: thu nhận tọa độ và vận tốc thực tế của xe.
2. Khối xử lý trung tâm

ESP32 là vi điều khiển chính, có nhiệm vụ:

- Thu thập dữ liệu từ các cảm biến qua I²C.
- Xử lý, tính toán tốc độ, áp suất, phát hiện điều kiện vượt ngưỡng.
- Kích hoạt tín hiệu cảnh báo (LED).
- Truyền dữ liệu vị trí và cảnh báo lên nền tảng Google Maps thông qua Wi-Fi.

3. Khối hiển thị và cảnh báo (Output)
   - Màn hình OLED 0.96 inch: hiển thị giá trị áp suất, tốc độ và trạng thái cảnh báo.

- LED cảnh báo: sáng khi vượt tốc độ hoặc áp suất ngoài giới hạn.

3.2. Nguyên lý hoạt động.

Khởi động: Khi cấp nguồn, ESP32 khởi tạo kết nối I²C, GPS và Wi-Fi. Màn hình OLED hiển thị thông báo khởi động và các giá trị ban đầu.

Đo áp suất:

Cảm biến BMP180 gửi dữ liệu áp suất đến ESP32 qua I²C. Vi điều khiển so sánh giá trị đo được với ngưỡng chuẩn (ví dụ: 32 psi ± 5%).

- Nếu áp suất nằm trong ngưỡng an toàn → hiển thị "an toàn".
- Nếu áp suất thấp/cao hơn giới hạn → kích hoạt LED cảnh báo.

Đo tốc độ:

- ESP32 đọc xung từ Encoder để tính tốc độ bánh xe. Đồng thời, module GPS NEO-6M cung cấp vận tốc thực tế qua chuỗi $GPRMC.
- Hai giá trị được đối chiếu để đảm bảo độ chính xác.

Cảnh báo vượt tốc độ:

Nếu tốc độ thực tế $v > v_{max}$ (ví dụ 40 km/h), ESP32 bật LED đỏ, đồng thời đánh dấu tọa độ GPS tại thời điểm vượt ngưỡng.

Gửi dữ liệu lên Google Maps:

Thông qua Wi-Fi, ESP32 gửi dữ liệu gồm:

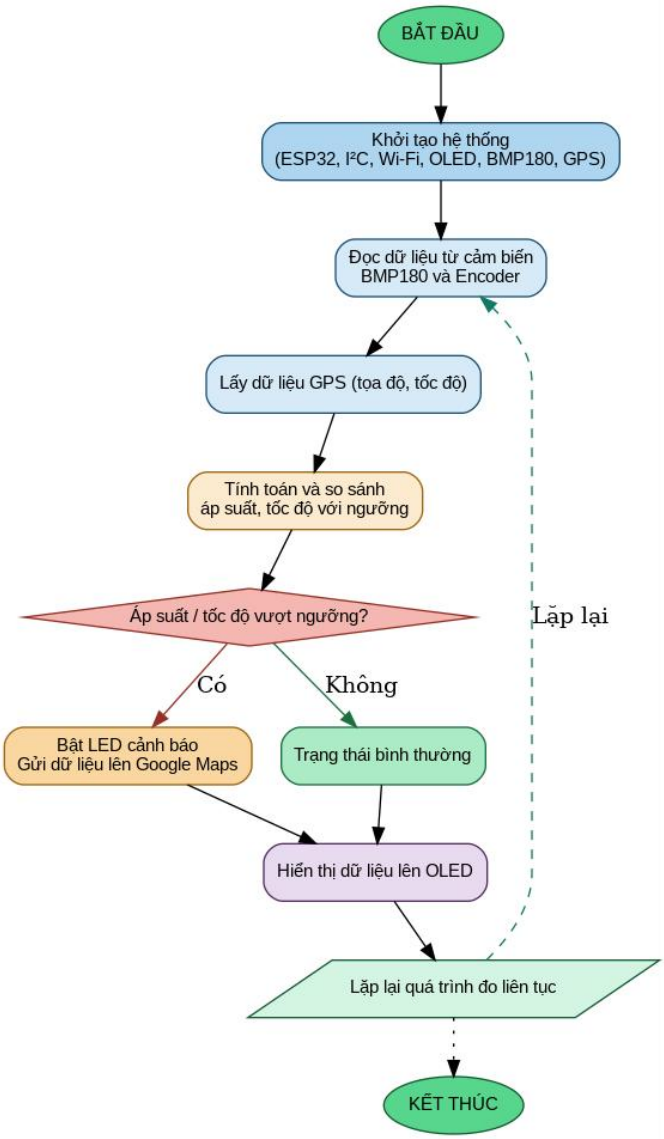
- Tọa độ GPS (latitude, longitude)

- Áp suất, tốc độ, trạng thái cảnh báo
- Dữ liệu được hiển thị trên bản đồ Google Maps thông qua API.

Hiển thị kết quả:

- OLED cập nhật liên tục các thông số:

- Dòng 1: Áp suất bánh xe
- Dòng 2: Tốc độ di chuyển (km/h)
- Dòng 3: Trạng thái cảnh báo (Safe / Warning)



Hình 11. Lưu đồ giải thuật

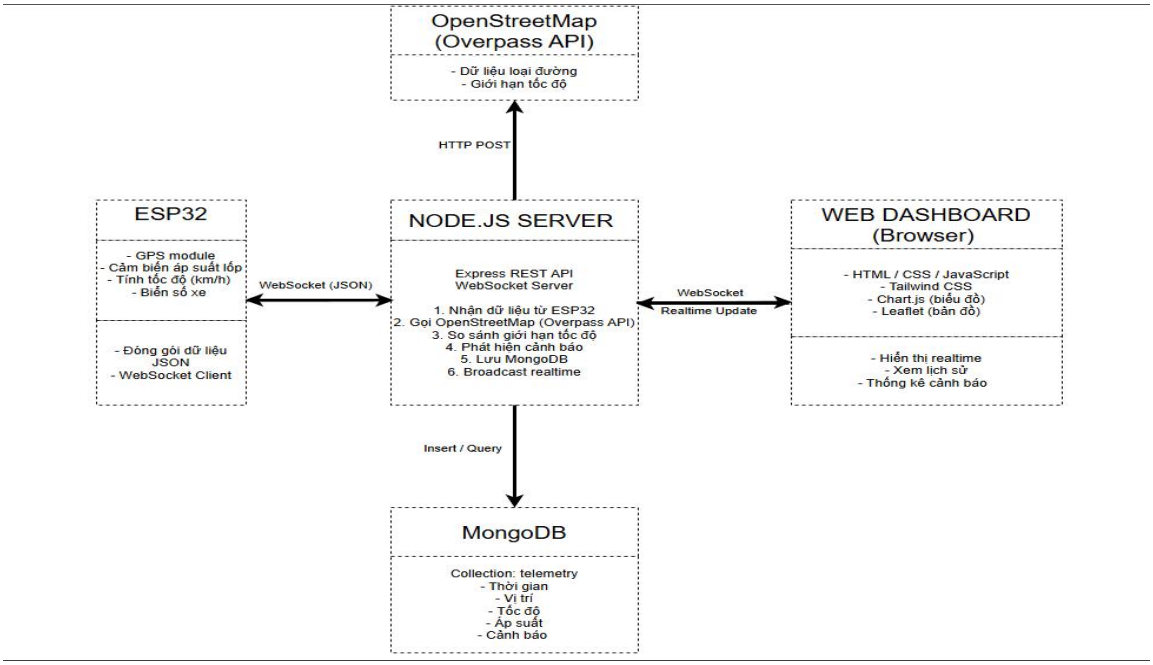
3.3. Luồng xử lý dữ liệu của hệ thống.

Hệ thống được thiết kế theo mô hình IoT – Client/Server, trong đó ESP32 đóng vai trò là thiết bị thu thập dữ liệu, server đảm nhiệm xử lý trung tâm, và Web Dashboard dùng để hiển thị thông tin theo thời gian thực. Luồng xử lý dữ liệu của hệ thống được mô tả như sau.

Thành phần	Công nghệ
Thiết bị	ESP32
Ngôn ngữ IoT	C/C++ (Arduino)
Backend	Node.js + Express
Frontend	HTML, CSS, JavaScript
Realtime	WebSocket
Database	MongoDB
Bản đồ	OpenStreetMap
Biểu đồ	Chart.js
Map UI	Leaflet
Data format	JSON

Bảng 2. Công nghệ sử dụng

3.3.1. Truyền dữ liệu từ ESP32 đến Server



ESP32 có nhiệm vụ thu thập các thông tin vận hành của xe như:

- Vị trí GPS
- Tốc độ di chuyển
- Áp suất

Các dữ liệu này được ESP32 gửi liên tục về server thông qua kết nối mạng theo thời gian thực. Việc truyền dữ liệu realtime giúp hệ thống có thể giám sát trạng thái của xe ngay tại thời điểm đang vận hành.

Hình 14. Server nhận và server gửi

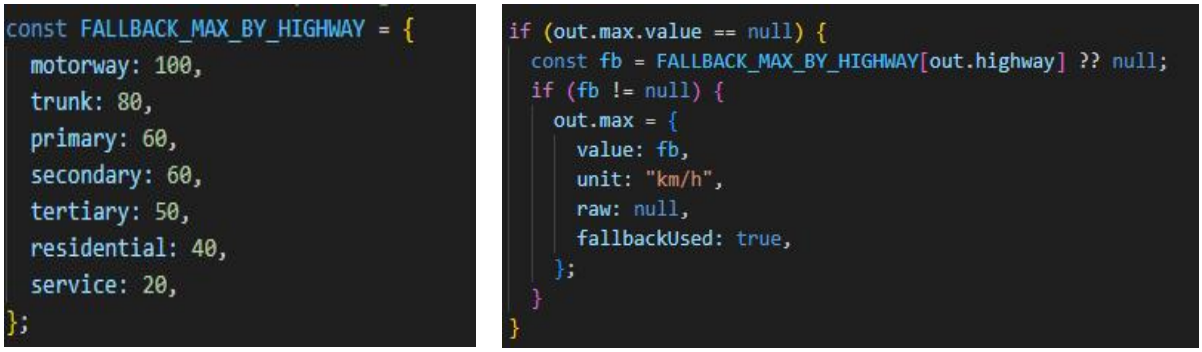
Sau khi nhận dữ liệu từ ESP32, server tiến hành xử lý và thực hiện truy vấn giới hạn tốc độ tương ứng với vị trí GPS nhận được. Server sử dụng dữ liệu bản đồ từ OpenStreetMap (OSM) để xác định giới hạn tốc độ của đoạn đường hiện tại. Quá trình này bao gồm:

- Nhận dữ liệu vị trí và tốc độ từ ESP32
- Gửi yêu cầu truy vấn giới hạn tốc độ đến OSM
- Nhận kết quả giới hạn tốc độ từ OSM để phục vụ so sánh

Hình 15. Server truy vấn giới hạn tốc độ

Trong một số trường hợp, OpenStreetMap không cung cấp đầy đủ thông tin về giới hạn tốc độ của đoạn đường. Để đảm bảo hệ thống luôn hoạt động ổn định, cơ chế **Fallback** được triển khai. Cơ chế Fallback bao gồm:

- Khai báo giá trị giới hạn tốc độ mặc định
- Xác định điều kiện kích hoạt Fallback khi OSM không trả về dữ liệu hợp lệ



Hình 16. Khai báo Fallback và điều kiện fallback

Nhờ đó, hệ thống luôn có giá trị giới hạn tốc độ để so sánh, tránh lỗi hoặc gián đoạn khi dữ liệu từ OSM bị thiếu.

Sau khi xác định được giới hạn tốc độ (từ OSM hoặc Fallback), server tiến hành so sánh với tốc độ thực tế của xe. Để giảm thiểu cảnh báo giả do sai số GPS hoặc nhiễu tín hiệu, hệ thống sử dụng một biên an toàn (margin)**.** Cảnh báo chỉ được kích hoạt khi:

- Tốc độ thực tế vượt quá giới hạn tốc độ
- Vượt quá ngưỡng margin cho phép

```
const overMax = limitMax != null ? speedKmh > limitMax + marginKmh : false;
const underMin = limitMin != null ? speedKmh < Math.max(0, limitMin - marginKmh) : false;
```

Hình 17. Logic so sánh tốc độ

Tốc độ	Giới hạn	Margin	Kết quả
65 km/h	60	5	Không cảnh báo
70 km/h	60	5	overMax = true

Với margin: Giảm cảnh báo giả do sai số GPS

Bảng 3. Logic so sánh tốc độ

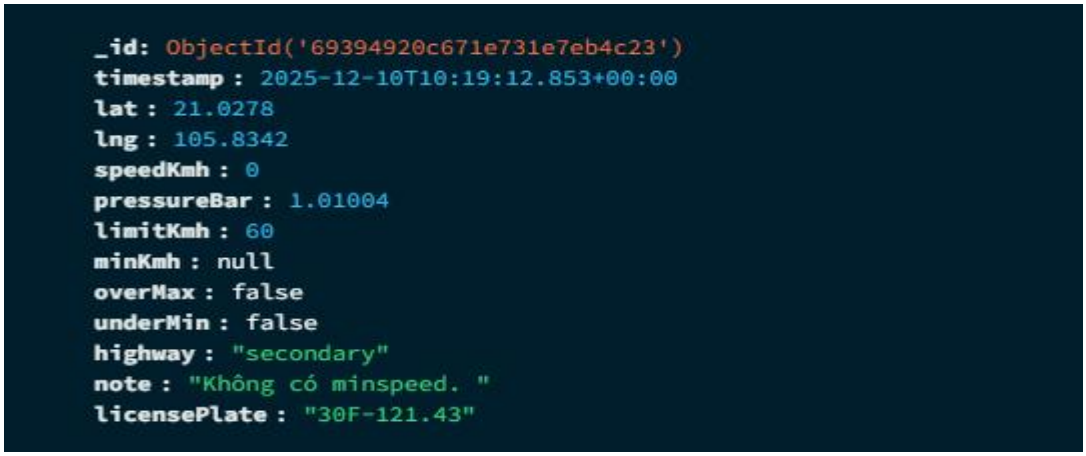
Cách tiếp cận này giúp tăng độ chính xác của hệ thống cảnh báo và hạn chế các cảnh báo không cần thiết.

Toàn bộ dữ liệu sau khi xử lý được lưu trữ vào cơ sở dữ liệu MongoDB, bao gồm:

- Dữ liệu vị trí
- Tốc độ
- Trạng thái cảnh báo

```
if (telemetryCol) {
  const doc = {
    timestamp: new Date(),
    lat,
    lng,
    speedKmh,
    pressureBar: typeof pressureBar === "number" ? pressureBar : null,
    limitKmh: limitMax,
    minKmh: limitMin,
    overMax,
    underMin,
    highway: r.highway || null,
    note,
    licensePlate,
  };
  try {
    await telemetryCol.insertOne(doc);
  } catch (err) {
    console.error("[Mongo] insert error:", err);
  }
```

Hình 18. Lưu dữ liệu vào MongoDB



Hình 19. Kết quả lưu dữ liệu tại MongoDB

Song song với quá trình lưu trữ, server thực hiện broadcast dữ liệu realtime đến Web Dashboard. Nhờ đó, giao diện web có thể:

- Cập nhật vị trí, tốc độ, áp suất của xe ngay lập tức
- Hiển thị cảnh báo mà không cần tải lại trang (refresh)

```
  const broadcast = {
    type: "telemetry_update",
    ...doc,
    timestamp: doc.timestamp.toISOString(),
  };

  wss.clients.forEach((client) => {
    if (
      client !== ws &&
      client.readyState === 1 &&
      client.clientType === "dashboard"
    ) {
      client.send(JSON.stringify(broadcast));
    }
  });
}
```

Hình 20. Broadcast realtime cho Web Dashboard

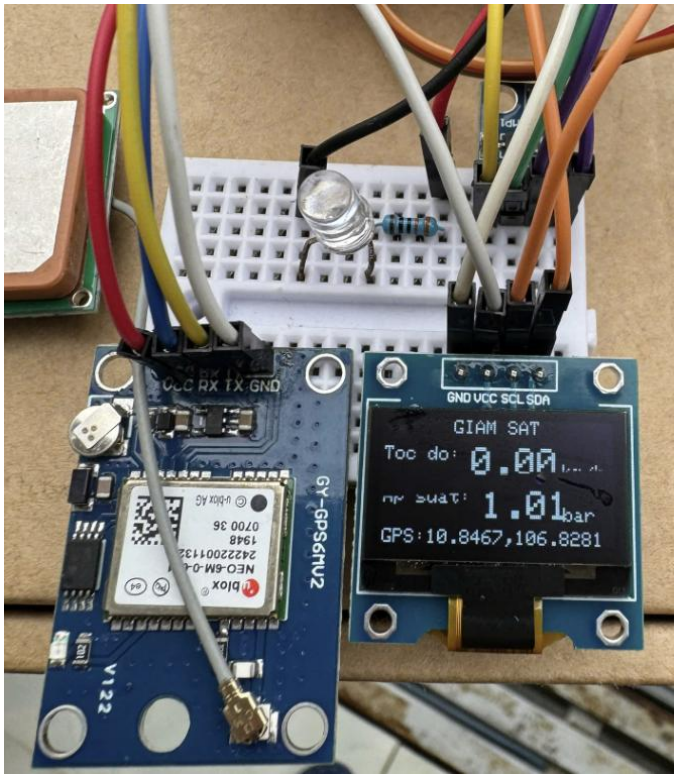
Ngoài ra, hệ thống cung cấp các API để:

- Truy vấn dữ liệu lịch sử
- Thống kê cảnh báo theo ngày
- Vẽ biểu đồ trực quan trên Dashboard

CHƯƠNG 4: KẾT QUẢ VÀ ĐÁNH GIÁ HỆ THỐNG

Chương này trình bày kết quả thực nghiệm của hệ thống sau khi thi công và lập trình hoàn chỉnh, đồng thời đánh giá mức độ ổn định, độ chính xác và khả năng đáp ứng của hệ thống trong các điều kiện mô phỏng. Các phép thử được tiến hành trong môi trường phòng thí nghiệm với các cảm biến BMP180, GPS NEO-6M, Encoder, kết nối ESP32 và màn hình OLED hiển thị.

4.1. Kết quả hiển thị và hoạt động hệ thống.



Hình 21. Màn hình OLED hiển thị áp suất và tốc độ thực tế

Khi hệ thống hoạt động, màn hình OLED hiển thị đầy đủ các thông số chính gồm:

- Áp suất bánh xe (bar).
- Tốc độ xe (km/h).
- GPS.

Dữ liệu áp suất được cập nhật trung bình mỗi 2 giây, trong khi tốc độ được cập nhật liên tục theo chu kỳ 100–200 ms, đảm bảo độ mượt và thời gian phản hồi nhanh.

4.2. Hiển thị vị trí trên bản đồ Google Maps.

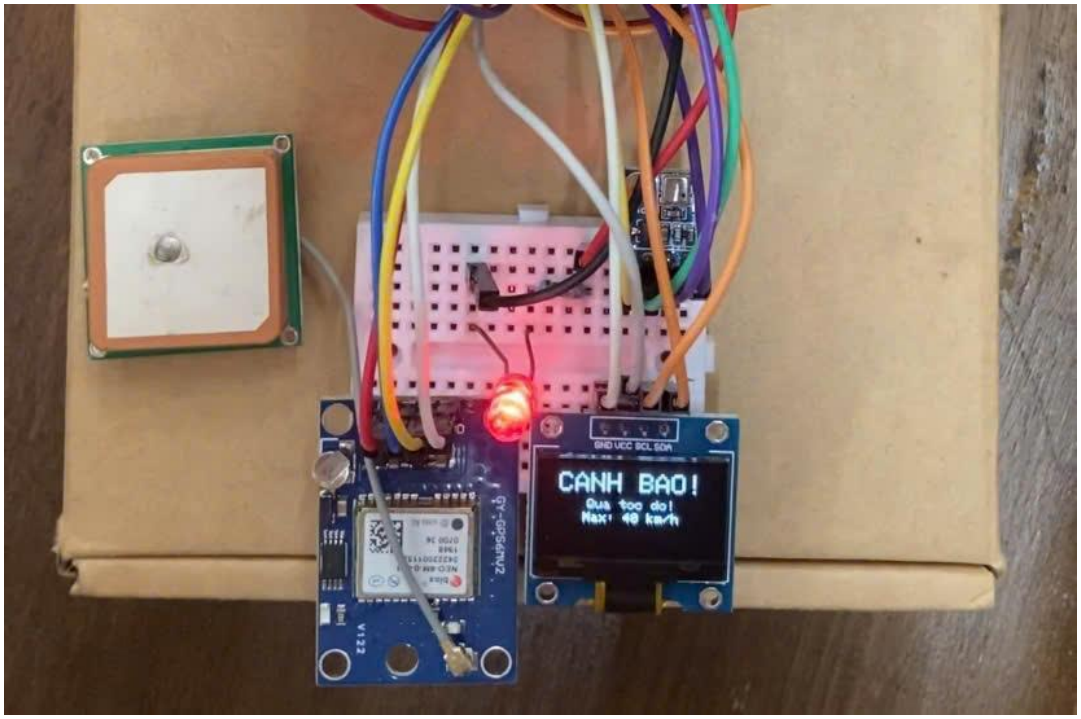
ESP32 sử dụng Wi-Fi để gửi dữ liệu tọa độ GPS (latitude, longitude) và trạng thái cảnh báo lên nền tảng web có tích hợp Google Maps API**.**

Chức năng này chứng minh khả năng mở rộng hệ thống theo hướng IoT giám sát phương tiện thời gian thực, không chỉ đo cục bộ mà còn truyền dữ liệu qua Internet.

Giả sử bảng số liệu giả lập như kiểu bạn đi qua nhiều vị trí khác nhau, mỗi lần hệ thống đọc được:

STT	Vị trí (lat, lon)	Áp suất đo (bar)	Vận tốc giới hạn cài (km/h)	Vận tốc GPS đo được (km/h)	Trạng thái LED cảnh báo
1	10.801200, 106.680500	2.30	40	32.5	Tắt (an toàn)
2	10.801450, 106.681000	1.95	40	34.2	Bật (áp suất thấp)
3	10.801800, 106.681500	2.45	40	41.8	Bật (vượt tốc độ)
4	10.802150, 106.682000	2.10	40	39.6	Tắt
5	10.802500, 106.682450	2.55	40	45.3	Bật (vượt tốc độ)
6	10.802900, 106.682900	1.80	40	28.7	Bật (áp suất thấp)
7	10.803200, 106.683300	2.35	40	40.0	Tắt
8	10.803550, 106.683700	2.70	40	42.9	Bật (vượt tốc độ)

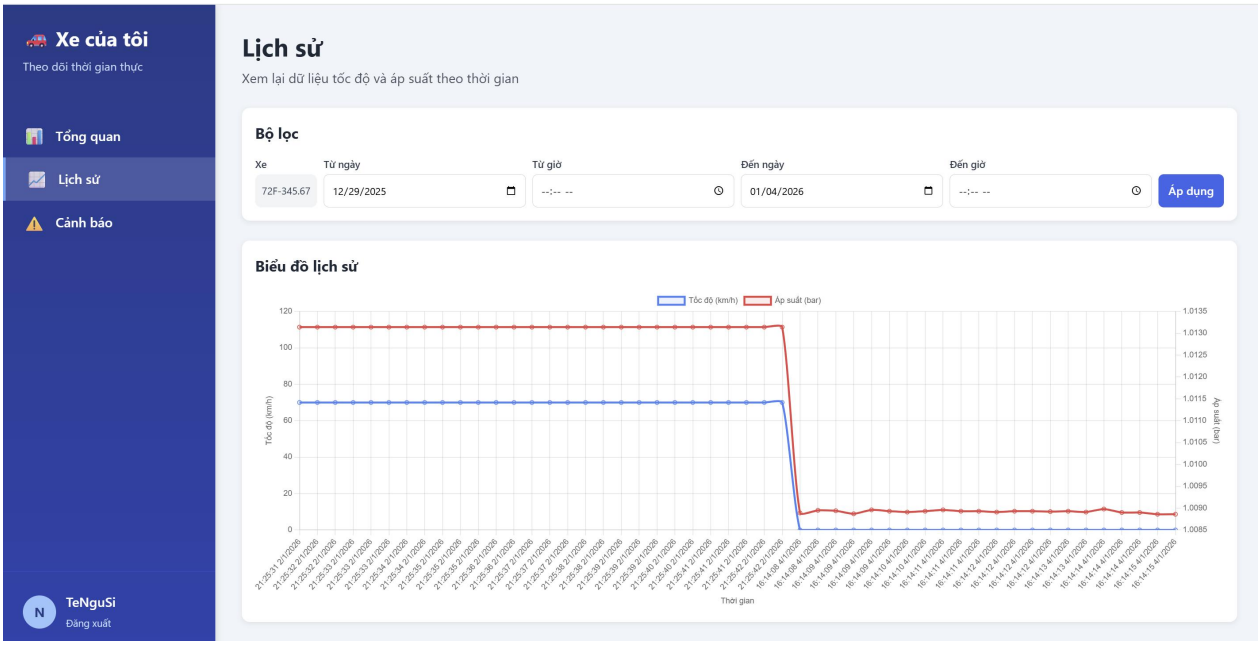
Bảng 4. Dữ liệu giả lập theo các vị trí khác nhau.



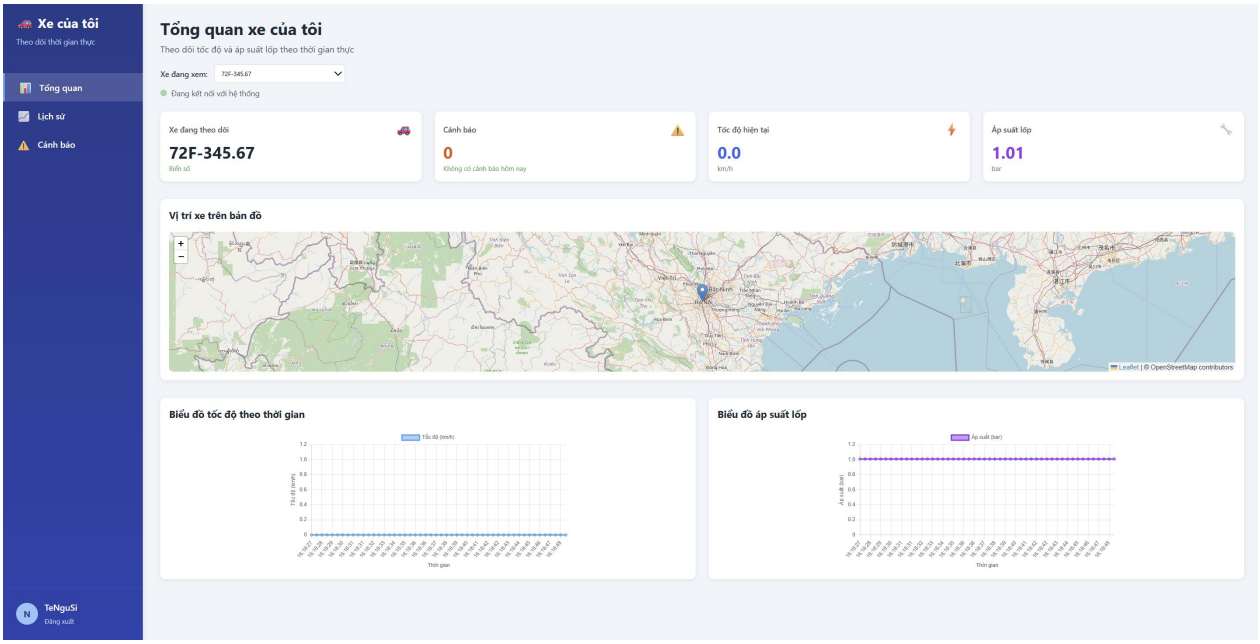
Hình 22. Màn hình OLED hiển thị cảnh báo

```
[2025-11-08T02:40:03.306Z] WS  IN   #4 ::ffff:172.20.10.4 { pos: '10.84766,106.79424', speedKmh: 1.814, margin: 5 }
[2025-11-08T02:40:03.307Z] WS  OUT  #4 ::ffff:172.20.10.4 {
  limitKmh: 40,
  minKmh: null,
  overMax: false,
  underMin: false,
  highway: 'residential',
  fbMax: true,
  fbMin: false
}
[2025-11-08T02:40:03.872Z] WS  IN   #4 ::ffff:172.20.10.4 { pos: '10.84766,106.79424', speedKmh: 1.739, margin: 5 }
[2025-11-08T02:40:03.873Z] WS  OUT  #4 ::ffff:172.20.10.4 {
  limitKmh: 40,
  minKmh: null,
  overMax: false,
  underMin: false,
  highway: 'residential',
  fbMax: true,
  fbMin: false
}
[2025-11-08T02:40:04.202Z] WS  IN   #4 ::ffff:172.20.10.4 { pos: '10.84766,106.79424', speedKmh: 1.739, margin: 5 }
[2025-11-08T02:40:04.203Z] WS  OUT  #4 ::ffff:172.20.10.4 {
  limitKmh: 40,
  minKmh: null,
  overMax: false,
  underMin: false,
  highway: 'residential',
  fbMax: true,
  fbMin: false
}
```

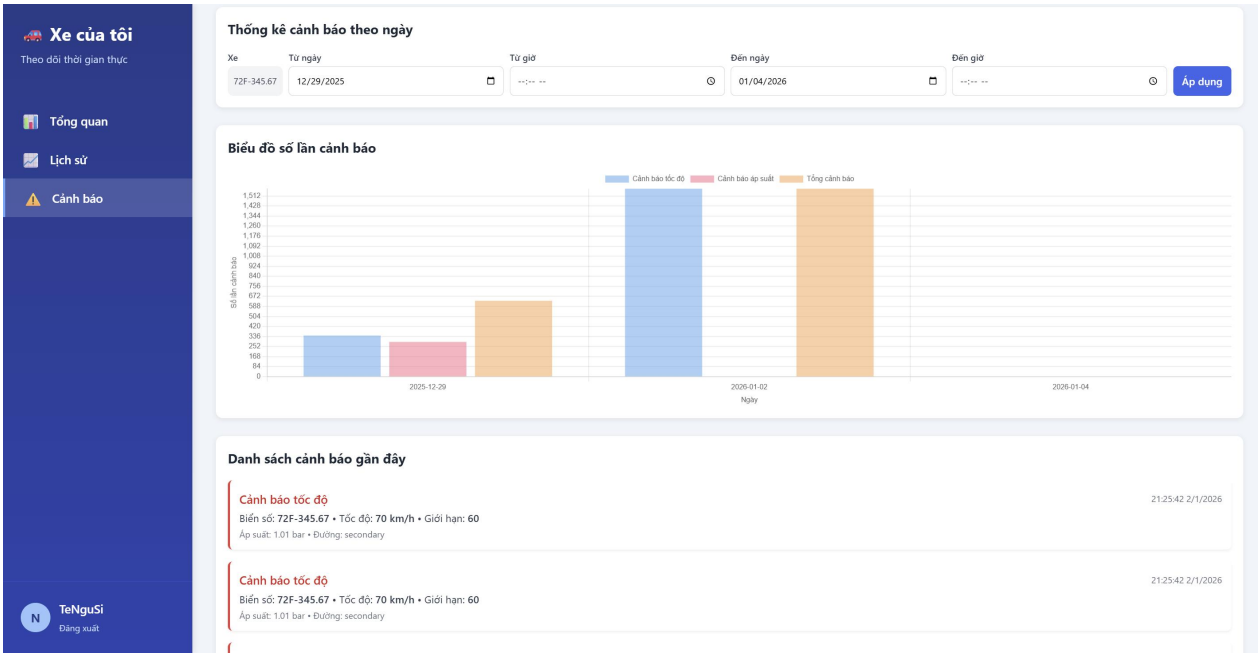
Hình 23. Server gửi dữ liệu



Hình 24. Lịch sử hoạt động của xe



Hình 25.Web cập nhật vị trí, tốc độ, áp suất



Hình 26. Vẽ biểu đồ cảnh báo

4.3. Hạn chế.

Do điều kiện thực tế về thời gian, kinh phí và môi trường thử nghiệm, đề tài hiện mới chỉ dừng ở mức độ mô hình giả lập, chưa triển khai trên phương tiện thật.

Các tín hiệu cảm biến và dữ liệu đo được chủ yếu được mô phỏng (giả lập) và quan sát qua cổng Serial Monitor của Arduino IDE nhằm kiểm tra hoạt động của phần mềm, thuật toán và phản hồi cảnh báo. Một số giới hạn cụ thể gồm:

- Dữ liệu vận tốc GPS và áp suất bánh xe được giả lập hoặc lấy mẫu ở điều kiện cố định, chưa thử nghiệm trong môi trường thực tế ngoài trời.
- Hệ thống chưa có cơ chế đo trực tiếp trên bánh xe thực, do hạn chế về phần cơ khí và độ kín của cảm biến áp suất.
- Việc truyền dữ liệu lên nền tảng Google Maps chỉ được thử nghiệm trong chế độ mô phỏng tọa độ GPS, chưa gắn cảm biến GPS trên phương tiện di chuyển thực.
- Độ chính xác và độ trễ truyền dữ liệu chưa được kiểm chứng trong điều kiện nhiễu hoặc mất kết nối Wi-Fi.

→ Tuy nhiên, mô hình đã thể hiện đầy đủ quy trình xử lý, hiển thị, và cảnh báo, chứng minh tính khả thi của hệ thống khi triển khai thực tế sau này.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận.

Sau quá trình nghiên cứu, thiết kế và thực nghiệm, đề tài "Kiểm tra áp suất bánh và cảnh báo khi xe vượt quá tốc độ (tích hợp GPS – Google Maps API)" đã hoàn thành các mục tiêu đề ra, bao gồm:

Thiết kế và thi công mô hình hệ thống sử dụng ESP32 làm bộ điều khiển trung tâm, tích hợp các cảm biến BMP180 (đo áp suất), Encoder và GPS NEO-6M (đo tốc độ và vị trí), cùng màn hình OLED và LED cảnh báo.

Xây dựng chương trình điều khiển có khả năng:

- Thu thập và xử lý dữ liệu cảm biến theo thời gian thực qua giao tiếp I^2C.
- Hiển thị các thông số áp suất, tốc độ và trạng thái trên màn hình OLED.
- Cảnh báo bằng đèn LED khi áp suất hoặc tốc độ vượt giới hạn cài đặt.
- Truyền dữ liệu lên nền tảng Google Maps thông qua kết nối Wi-Fi, phục vụ việc theo dõi vị trí phương tiện.

Mô hình hoạt động ổn định, phản hồi nhanh, giao diện hiển thị rõ ràng và có khả năng mở rộng cho các ứng dụng IoT giám sát phương tiện.

Kết quả thử nghiệm cho thấy hệ thống đạt độ chính xác cao trong việc đo lường và cảnh báo, đồng thời đảm bảo khả năng truyền dữ liệu không dây ổn định, đáp ứng yêu cầu của một mô hình TPMS (Tire Pressure Monitoring System) kết hợp theo dõi tốc độ thông minh.

→ Nhìn chung, đề tài đã đạt được đầy đủ các mục tiêu thiết kế, chứng minh tính khả thi, hiệu quả và ứng dụng thực tế trong lĩnh vực an toàn giao thông thông minh.

5.2. Hướng phát triển của đề tài

Mặc dù mô hình hiện tại mới ở mức giả lập và thử nghiệm trong phòng, nhưng hệ thống có tiềm năng lớn để mở rộng và phát triển trong tương lai theo các hướng sau:

Triển khai thực tế trên phương tiện thật:

- Lắp đặt cảm biến áp suất không dây gắn trực tiếp trên từng bánh xe.
- Tích hợp nguồn cấp từ ắc quy hoặc pin sạc, đảm bảo hoạt động lâu dài.
- Phát triển giao diện người dùng (UI/UX):
- Thiết kế ứng dụng web hoặc mobile (Android/iOS) để hiển thị dữ liệu trực quan hơn, đồng bộ với Google Maps.
- Cho phép tùy chỉnh ngưỡng áp suất và tốc độ trực tiếp trên giao diện.
- Lưu trữ và phân tích dữ liệu trên Cloud:
- Sử dụng Firebase, Thingspeak hoặc MQTT Broker để lưu trữ dữ liệu hành trình, tạo báo cáo thống kê, đồ thị thời gian thực.
- Áp dụng AI hoặc Machine Learning để dự đoán tình trạng lốp và thói quen lái xe.

Tối ưu phần cứng – năng lượng:

- Thiết kế mạch PCB chuyên dụng, gọn nhẹ, tiết kiệm năng lượng.
- Sử dụng chế độ Deep Sleep của ESP32 để kéo dài thời gian hoạt động.

Mở rộng phạm vi ứng dụng:

- Ứng dụng cho xe xáy, ô tô, xe đạp điện, hoặc mô hình học tập IoT.
- Kết hợp với các cảm biến khác (nhiệt độ, rung, độ nghiêng) để xây dựng hệ thống giám sát phương tiện toàn diện.

5.3. Kết luận chung.

Đề tài là một hướng nghiên cứu mang tính ứng dụng cao, góp phần vào việc nâng cao an toàn giao thông, đồng thời tạo môi trường thực hành thiết thực cho sinh viên kỹ thuật trong lĩnh vực điện – điện tử – IoT.

Mặc dù còn giới hạn về điều kiện thực nghiệm, mô hình đã chứng minh được tính khả thi, dễ triển khai và mở rộng, là cơ sở vững chắc cho các nghiên cứu và ứng dụng thực tế sau này.

PHỤ LỤC

Code arduino

```
// demo thực tế

#include <Arduino.h>
#include <Wire.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <WebSocketsClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#include <Adafruit_BMP085.h>   // BMP180 compatible
#include <Adafruit_Sensor.h>
#include <TinyGPSPlus.h>

// ================== WiFi / WebSocket ==================
const char* WIFI_SSID = "Thuyne";          // <-- WiFi
const char* WIFI_PASS = "01072003";   // <-- WiFi

// ws://<PC-IP>:3000/ws
const char* WS_HOST = "172.20.10.3";   // <-- IP PC chạy server
const uint16_t WS_PORT = 3000;
const char* WS_PATH = "/ws";

const char* LICENSE_PLATE = "72F-345.67";

// ================== OLED / LED ==================
#define LED_PIN 19
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// ================== Encoder (GPIO34) ==================
// Ghi chú: GPIO34 input-only, không có pull-up nội -> cần kéo lên/kéo xuống
ngoài nếu đầu ra encoder là dạng hở
#define ENC_PIN               34
#define PULSES_PER_REV        20        // <-- số xung/vòng theo encoder của
bạn
#define SAMPLE_MS             500       // khoảng lấy mẫu tốc độ
#define MIN_PULSE_US          200       // lọc nhiễu biên
#define WHEEL_CIRCUMFERENCE_M 0.21f     // <-- chu vi bánh/đĩa (m)

volatile uint32_t encPulseCount = 0;
```

```
volatile uint32_t encLastPulseUs = 0;
void IRAM_ATTR onEncPulse() {
  uint32_t nowUs = micros();
  if (nowUs - encLastPulseUs >= MIN_PULSE_US) {
    encPulseCount++;
    encLastPulseUs = nowUs;
  }
}

// ================== BMP180 (I2C) ==================
Adafruit_BMP085 bmp;

// ================== GPS (UART2) ==================
#define GPS_RX 16   // ESP32 RX2  ← TX GPS
#define GPS_TX 17   // ESP32 TX2  → RX GPS
#define GPS_BAUD 9600
HardwareSerial SerialGPS(2);
TinyGPSPlus gps;

// ================== Trạng thái & ngưỡng ==================
WebSocketsClient ws;
bool wsConnected = false;

float speed_kmh    = 0.0f;   // từ encoder
float pressure_bar = 0.0f;   // từ BMP180

// Ngưỡng áp suất (cục bộ cho OLED/LED)
float PRESSURE_LIMIT_MIN = 1.00f;  // bar
float PRESSURE_LIMIT_MAX = 2.00f;  // bar

// GPS: current + last-fix
bool   gpsFix = false;          // có fix hiện tại?
double curLat = 0.0, curLng = 0.0;
double lastLat = 0.0, lastLng = 0.0;
unsigned long lastFixMs = 0;    // thời điểm có fix lần gần nhất (millis)

// ================== Toạ độ nhập tay qua Serial ==================
bool   manualCoord = false;
double manualLat   = 0.0;
double manualLng   = 0.0;

// Phản hồi server cho tốc độ
bool  srv_overMax = false;
bool  srv_underMin = false;
int   srv_limit_kmh = -1;
int   srv_min_kmh   = -1;
```

```
String srv_note;

const uint32_t SEND_INTERVAL_MS = 400;
uint32_t lastSendMs = 0;

const uint32_t BLINK_INTERVAL_MS = 350;
uint32_t lastBlinkMs = 0;
bool blinkOn = false;

// ================== Helpers hiển thị ==================
void drawCenteredText(const String &text, int16_t y, uint8_t textSize = 1) {
  int16_t x1, y1; uint16_t w, h;
  display.setTextSize(textSize);
  display.getTextBounds(text, 0, y, &x1, &y1, &w, &h);
  int16_t x = (SCREEN_WIDTH - (int16_t)w) / 2;
  display.setCursor(x, y);
  display.print(text);
}
void drawCenteredLines(const String lines[], uint8_t count, int16_t startY,
uint8_t textSize = 1, uint8_t spacing = 10) {
  for (uint8_t i = 0; i < count; i++) drawCenteredText(lines[i], startY + i *
spacing, textSize);
}
void updateBlink() {
  uint32_t now = millis();
  if (now - lastBlinkMs >= BLINK_INTERVAL_MS) { lastBlinkMs = now; blinkOn
= !blinkOn; }
}

// ================== Màn hình ==================
void showNormal(bool showGpsLine = true) {
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);
  drawCenteredText(F("GIAM SAT"), 0, 1);

  // Tốc độ
  display.setTextSize(1);
  display.setCursor(2, 14); display.print(F("Toc do: "));
  display.setTextSize(2);   display.print(speed_kmh, 2);
  display.setTextSize(1);   display.setCursor(100, 22);
display.print(F("km/h"));

  // Áp suất
  display.setTextSize(1);
  display.setCursor(2, 36); display.print(F("Ap suat: "));
  display.setTextSize(2);   display.print(pressure_bar, 2);
```

```
  display.setTextSize(1);   display.setCursor(100, 44);
display.print(F("bar"));

  // GPS dòng trạng thái
  if (showGpsLine) {
    display.setTextSize(1);
    display.setCursor(2, 56);
    if (gpsFix) {
      display.print(F("GPS:"));
      display.print(curLat, 4);
      display.print(",");
      display.print(curLng, 4);
    } else {
      unsigned long age = (millis() - lastFixMs) / 1000;
      display.print(F("GPS:"));
      display.print(age);
      display.print(F("s)"));
    }
  }

  display.display();
}

void showAlert() {
  bool pressHigh = (pressure_bar > PRESSURE_LIMIT_MAX);
  bool pressLow  = (pressure_bar < PRESSURE_LIMIT_MIN);
  bool spdAlert  = (srv_overMax || srv_underMin);
  bool prsAlert  = (pressHigh || pressLow);

  display.clearDisplay();

  if (blinkOn) {
    display.setTextColor(SSD1306_WHITE);
    drawCenteredText(F("CANH BAO!"), 2, 2);
  }

  String lines[6];
  uint8_t n = 0;

  if (prsAlert && !spdAlert) {
    if (pressHigh) lines[n++] = "Ap suat cao!";
    if (pressLow)  lines[n++] = "Ap suat thap!";
    lines[n++] = String(PRESSURE_LIMIT_MIN,2) + "-" +
String(PRESSURE_LIMIT_MAX,2) + " bar";
  }
  else if (spdAlert && !prsAlert) {
```

```
    if (srv_overMax)  lines[n++] = "Qua toc do!";
    if (srv_underMin) lines[n++] = "Toc do thap!";
    if (srv_limit_kmh >= 0) lines[n++] = String("Max: ") + srv_limit_kmh + "
km/h";
  }
  else if (spdAlert && prsAlert) {
    if (srv_overMax)  lines[n++] = "Qua toc do!";
    if (srv_underMin) lines[n++] = "Toc do thap!";
    if (srv_limit_kmh >= 0) lines[n++] = String("Max: ") + srv_limit_kmh + "
km/h";
    if (pressHigh) lines[n++] = "Ap suat cao!";
    if (pressLow)  lines[n++] = "Ap suat thap!";
    lines[n++] = String(PRESSURE_LIMIT_MIN,2) + "-" +
String(PRESSURE_LIMIT_MAX,2) + " bar";
  }
  else {
    lines[n++] = "Dang xu ly...";
  }

  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  drawCenteredLines(lines, n, 26, 1, 10);
  display.display();
}

// ================== LED ==================
void updateLED() {
  bool pressHigh = (pressure_bar > PRESSURE_LIMIT_MAX);
  bool pressLow  = (pressure_bar < PRESSURE_LIMIT_MIN);
  bool alert = srv_overMax || srv_underMin || pressHigh || pressLow;
  digitalWrite(LED_PIN, (alert && blinkOn) ? HIGH : LOW);
}

// ================== WebSocket ==================
void handleWSMessage(const String& msg) {
  StaticJsonDocument<512> d;
  DeserializationError err = deserializeJson(d, msg);
  if (err) { Serial.print(F("[WS] JSON err: ")); Serial.println(err.c_str());
return; }
  if (d.containsKey("error")) { Serial.print(F("[WS] Error: "));
Serial.println((const char*)d["error"]); return; }
  if (d["type"] && String((const char*)d["type"]) != "compare_result") return;

  srv_limit_kmh = d["limitKmh"].isNull() ? -1 : (int)d["limitKmh"].as<int>();
  srv_min_kmh   = d["minKmh"].isNull() ? -1 : (int)d["minKmh"].as<int>();
  srv_overMax   = d["overMax"]  | false;
```

```
  srv_underMin  = d["underMin"] | false;
  srv_note      = d["note"]     | "";

  Serial.printf("[WS<-] limit=%d min=%d over=%d under=%d note=%s\n",
    srv_limit_kmh, srv_min_kmh, srv_overMax, srv_underMin, srv_note.c_str());
}

void onWSEvent(WStype_t type, uint8_t * payload, size_t length) {
  switch (type) {
    case WStype_CONNECTED:   wsConnected = true;  Serial.println(F("[WS] 
Connected")); break;
    case WStype_DISCONNECTED:wsConnected = false; Serial.println(F("[WS] 
Disconnected")); break;
    case WStype_TEXT: {
      String msg((char*)payload, length);
      handleWSMessage(msg);
      break;
    }
    case WStype_PING:  Serial.println(F("[WS] PING"));  break;
    case WStype_PONG:  Serial.println(F("[WS] PONG"));  break;
    default: break;
  }
}

void sendWS() {
  uint32_t now = millis();
  if (!wsConnected || (now - lastSendMs < SEND_INTERVAL_MS)) return;
  lastSendMs = now;

  double sendLat, sendLng;

  if (manualCoord) {
    // Ưu tiên toạ độ do bạn nhập từ Serial
    sendLat = manualLat;
    sendLng = manualLng;
  } else if (!gpsFix && lastFixMs == 0) {
    // demo: nếu chưa có gì cả thì dùng toạ độ cố định
    sendLat = 21.0278;
    sendLng = 105.8342;
  } else {
    sendLat = gpsFix ? curLat : lastLat;
    sendLng = gpsFix ? curLng : lastLng;
  }

  unsigned long fixAgeMs = (lastFixMs == 0) ? (unsigned long)UINT32_MAX : 
(millis() - lastFixMs);
```

```
  StaticJsonDocument<256> doc;
  doc["lat"] = sendLat;
  doc["lng"] = sendLng;
  doc["gpsFix"] = gpsFix;
  doc["fixAgeMs"] = fixAgeMs;
  doc["speedKmh"] = speed_kmh;
  doc["pressureBar"] = pressure_bar;
  doc["margin"] = 5;
  doc["licensePlate"] = LICENSE_PLATE;  // nếu bạn đã thêm biển số

  String out; serializeJson(doc, out);
  ws.sendTXT(out);
}


// ================== Đọc cảm biến ==================
void pollGPS() {
  while (SerialGPS.available()) gps.encode(SerialGPS.read());

  bool validLoc = gps.location.isValid();
  if (gps.location.isUpdated() && validLoc) {
    curLat = gps.location.lat();
    curLng = gps.location.lng();
    gpsFix = true;
    lastLat = curLat;
    lastLng = curLng;
    lastFixMs = millis();
  } else if (!validLoc) {
    gpsFix = false;
  }
}

bool readPressureBMP180(float &barOut) {
  // readPressure() trả Pa; trả 0 nếu lỗi
  int32_t Pa = bmp.readPressure();
  if (Pa <= 0) return false;
  barOut = Pa / 100000.0f; // Pa -> bar
  return true;
}

// ================== Nhập toạ độ & data từ Serial ==================
void handleSerialCoordinateInput() {
  if (!Serial.available()) return;

  static String line;
  while (Serial.available()) {
```

```
    char c = Serial.read();
    if (c == '\r') continue;

    if (c == '\n') {
      line.trim();
      if (line.length() > 0) {
        // Đổi dấu phẩy thành khoảng trắng để dễ tách
        line.replace(',', ' ');

        float vals[4];
        int count = 0;

        int idx = 0;
        while (idx < line.length() && count < 4) {
          int next = line.indexOf(' ', idx);
          if (next < 0) next = line.length();
          String token = line.substring(idx, next);
          token.trim();
          if (token.length() > 0) {
            vals[count++] = token.toFloat();
          }
          idx = next + 1;
        }

        // vals[0] = lat, vals[1] = lng, vals[2] = speed, vals[3] = pressure

        if (count >= 2) {
          double lat = vals[0];
          double lng = vals[1];

          // Không check quá chặt, chỉ tránh đều =0
          if (lat != 0.0 || lng != 0.0) {
            manualLat = lat;
            manualLng = lng;
            manualCoord = true;      // bật cờ dùng toạ độ nhập tay
            gpsFix = false;          // để dễ phân biệt trên OLED nếu muốn
            lastLat = lat;
            lastLng = lng;
            lastFixMs = millis();

            Serial.print(F("[SER] Manual coord set: "));
            Serial.print(lat, 6);
            Serial.print(F(", "));
            Serial.println(lng, 6);
          } else {
            Serial.println(F("[SER] Invalid coord (both 0)"));
```

```
          }
        }

        if (count >= 3) {
          speed_kmh = vals[2];
          Serial.print(F("[SER] Manual speed_kmh = "));
          Serial.println(speed_kmh, 2);
        }

        if (count >= 4) {
          pressure_bar = vals[3];
          Serial.print(F("[SER] Manual pressure_bar = "));
          Serial.println(pressure_bar, 2);
        }
      }
      line = "";
    } else {
      // giới hạn độ dài dòng tránh tràn
      if (line.length() < 80) line += c;
    }
  }
}


// ================== Setup / loop ==================
void setup() {
  Serial.begin(115200); delay(200);
  pinMode(LED_PIN, OUTPUT); digitalWrite(LED_PIN, LOW);

  // I2C (OLED + BMP180)
  Wire.begin(21, 22);  // SDA=21, SCL=22

  // OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("[ERR] OLED not found (0x3C)!"));
    for(;;);
  }
  display.clearDisplay(); display.setTextSize(2);
display.setTextColor(SSD1306_WHITE);
  drawCenteredText(F("Starting"), 22, 2); display.display(); delay(600);

  // BMP180
  if (!bmp.begin()) {
    Serial.println(F("[ERR] BMP180 not found (0x77)!"));
  } else {
    Serial.println(F("[BMP180] OK"));
  }
```

```
  // GPS UART2
  SerialGPS.begin(GPS_BAUD, SERIAL_8N1, GPS_RX, GPS_TX);
  Serial.println(F("[GPS] UART2 started"));

  // WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("[WiFi] Connecting");
  uint32_t t0 = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - t0 < 12000) { delay(400);
Serial.print("."); }
  Serial.println();
  if (WiFi.status() == WL_CONNECTED) { Serial.print("[WiFi] ");
Serial.println(WiFi.localIP()); }
  else { Serial.println("[WiFi] Offline."); }

  // WebSocket
  ws.onEvent(onWSEvent);
  ws.begin(WS_HOST, WS_PORT, WS_PATH);
  ws.setReconnectInterval(2000);
  ws.enableHeartbeat(15000, 3000, 2);

  // Encoder
  pinMode(ENC_PIN, INPUT); // cần phần cứng kéo lên/kéo xuống phù hợp
  attachInterrupt(digitalPinToInterrupt(ENC_PIN), onEncPulse, RISING);
}

void loop() {
  // 1) Nhấp nháy + WS
  updateBlink();
  ws.loop();

  // 2) Cập nhật GPS + BMP180
  pollGPS();

  // CHỈ đọc cảm biến áp suất thật nếu KHÔNG nhập tay
  if (!manualCoord) {
    float pbar;
    if (readPressureBMP180(pbar)) {
      pressure_bar = pbar;
    }
  }

  // 3) Tính tốc độ từ encoder mỗi SAMPLE_MS
  static uint32_t lastCalcMs = 0;
```

```
  uint32_t nowMs = millis();
  if (nowMs - lastCalcMs >= SAMPLE_MS) {
    lastCalcMs = nowMs;

    // CHỈ tính tốc độ encoder nếu KHÔNG nhập tay
    if (!manualCoord) {
      noInterrupts();
      uint32_t pulses = encPulseCount;
      encPulseCount = 0;
      interrupts();

      float revolutions = (float)pulses / (float)PULSES_PER_REV;
      float rps = revolutions * (1000.0f / (float)SAMPLE_MS);
      float v_mps = rps * WHEEL_CIRCUMFERENCE_M;
      speed_kmh = v_mps * 3.6f;

      Serial.printf("[ENC] Real: %.2f km/h | Press: %.2f bar\n", speed_kmh,
pressure_bar);
    } else {
      // (Tuỳ chọn) In ra log nhắc nhở đang ở chế độ Manual
      // Serial.printf("[MANUAL] Sim: %.2f km/h | Press: %.2f bar\n",
speed_kmh, pressure_bar);
    }
  }

  handleSerialCoordinateInput();

  // 4) Gửi WS
  sendWS();

  // 5) Hiển thị + LED
  bool pressHigh = (pressure_bar > PRESSURE_LIMIT_MAX); // > 2.0 bar
  bool pressLow  = (pressure_bar < PRESSURE_LIMIT_MIN); // < 1.0 bar
  bool anyAlert  = srv_overMax || srv_underMin || pressHigh || pressLow;

  updateLED();
  if (anyAlert) showAlert(); else showNormal(true);

  delay(10);
}
```

TÀI LIỆU THAM KHẢO

1. Bosch Sensortec. BMP180 / BMP280 Digital Barometric Pressure Sensor Data Sheet, Rev. 2.5, 2022. https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/
2. u-blox AG. NEO-6M GPS Module Hardware Integration Manual, Rev. 1.0, 2021. https://content.u-blox.com
3. Arduino Official Documentation. TinyGPS++ Library for NMEA Data Parsing, 2023. https://github.com/mikalhart/TinyGPSPlus
4. Google Developers. Google Maps Platform Documentation – Maps JavaScript API & Geocoding API, 2024. https://developers.google.com/maps/documentation
5. Nguyễn Văn Hiếu, Trần Quốc Khánh. Giáo trình Vi điều khiển và Ứng dụng IoT, Nhà xuất bản Khoa học & Kỹ thuật, Hà Nội, 2022.
6. Lê Minh Tiến. Thiết kế hệ thống nhúng với ESP32 và IoT Cloud, Đại học Bách Khoa TP.HCM, 2023.
7. Tổ chức Y tế Thế giới (WHO). Global Status Report on Road Safety, 2023. https://www.who.int/publications
8. Ủy ban An toàn Giao thông Quốc gia Việt Nam. Báo cáo thống kê tai nạn giao thông năm 2024, Hà Nội, 2025.
9. Bridgestone Việt Nam. Nghiên cứu về tình trạng áp suất lốp và tiêu hao nhiên liệu trên xe máy Việt Nam, Báo cáo kỹ thuật nội bộ, 2023.
10. Datasheet Encoder – Keyes KY-040. Keyestudio Technical Manual, 2022. https://wiki.keyestudio.com/
11. ESP-IDF Documentation. Wi-Fi and I²C Communication Guide, Espressif Developer Portal, 2024.
12. Nguyễn Hữu Thành. Cảm biến và đo lường trong kỹ thuật điện tử, Nhà xuất bản Khoa học Tự nhiên & Công nghệ, 2021.