

ĐẠI HỌC HUẾ  
TRƯỜNG ĐẠI HỌC KHOA HỌC  
KHOA CÔNG NGHỆ THÔNG TIN



**KHOÁ LUẬN  
TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**TÌM HIỂU ESP32 VÀ XÂY DỰNG ỨNG  
DỤNG DI ĐỘNG IOT BÁO CHÁY VỚI  
FLUTTER**

Sinh viên thực hiện: **TRẦN CÔNG DŨNG**

Khóa: **K45 – HỆ CHÍNH QUY**

Huế, tháng 6 – năm 2025

ĐẠI HỌC HUẾ  
TRƯỜNG ĐẠI HỌC KHOA HỌC  
KHOA CÔNG NGHỆ THÔNG TIN



KHOÁ LUẬN  
TỐT NGHIỆP ĐẠI HỌC  
NGÀNH CÔNG NGHỆ THÔNG TIN

Đề tài:

TÌM HIỂU ESP32 VÀ XÂY DỰNG ỨNG  
DỤNG DI ĐỘNG IOT BÁO CHÁY VỚI  
FLUTTER

Sinh viên thực hiện: Trần Công Dũng

Khóa: K45 - Hệ chính quy

Giáo viên hướng dẫn: ThS. Võ Việt Dũng

Huế, tháng 6 – năm 2025

## LỜI CẢM ƠN

Trước hết em xin gửi lời cảm ơn vô cùng to lớn đến ThS. Võ Việt Dũng, người đã không quản thời gian và tâm sức, tận tình hướng dẫn, định hướng nghiên cứu và đồng hành trong suốt quá trình hoàn thành đề tài khóa luận này.

Em xin trân trọng gửi lời cảm ơn đến Ban Giám hiệu Trường đại học Khoa Học, cùng toàn thể Quý Thầy Cô trong khoa Công nghệ thông tin đã luôn tạo điều kiện học tập và nghiên cứu tốt nhất, trang bị cho em những kiến thức nền tảng vững chắc trong suốt thời gian em học tập ở trường.

Em cũng xin chân thành cảm ơn bạn bè và các anh chị đồng nghiệp đã luôn chia sẻ, hỗ trợ và cùng em vượt qua những thử thách trong suốt quá trình thực hiện khóa luận.

Mặc dù đã cố gắng hoàn thành sản phẩm trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý thầy cô, điều đó giúp em có thêm kinh nghiệm cho những nghiên cứu tiếp theo.

Em xin chân thành cảm ơn!

Huế, ngày 01 tháng 06 năm 2025

Sinh viên thực hiện

Trần Công Dũng

## BẢNG CHỮ CÁI VIẾT TẮT

IOT	INTERNET OF THING
PCCC	PHÒNG CHÁY CHỮA CHÁY
API	APPLICATION PROGRAMMING INTERFACE
JWT	JSON WEB TOKEN
FCM	FIREBASE CLOUD MESSAGING
AOT	AHEAD OF TIME
JIT	JUST IN TIME
I/O	DIGITAL INPUT OUTPUT PIN

## DANH MỤC HÌNH ẢNH

<b>Hình 1 : Hình ảnh thực tế ESP32 .....</b>	<b>5</b>
<b>Hình 2: Hình ảnh thực tế DHT22 .....</b>	<b>7</b>
<b>Hình 3: Cảm biến khói, khí gas MQ2.....</b>	<b>7</b>
<b>Hình 4: Cảm biến lửa.....</b>	<b>8</b>
<b>Hình 5: Ngôn ngữ lập trình Dart .....</b>	<b>10</b>
<b>Hình 6 : Framework Flutter.....</b>	<b>11</b>
<b>Hình 7: Kiến trúc của Flutter .....</b>	<b>12</b>
<b>Hình 8: Firebase và dịch vụ.....</b>	<b>15</b>
<b>Hình 9: Logo Node js .....</b>	<b>18</b>
<b>Hình 10: Logo Express JS .....</b>	<b>19</b>
<b>Hình 11: Header trong JWT .....</b>	<b>20</b>
<b>Hình 12: Payload trong JWT .....</b>	<b>21</b>
<b>Hình 13: Signature trong JWT .....</b>	<b>21</b>
<b>Hình 14: Mô phỏng hoạt động của JWT.....</b>	<b>21</b>
<b>Hình 15: Minh họa kết nối Web Socket.....</b>	<b>22</b>
<b>Hình 16: Kết nối HTTP .....</b>	<b>23</b>
<b>Hình 17: Hình vẽ cách hoạt động của ứng dụng .....</b>	<b>25</b>
<b>Hình 18: Usecase tổng quát của ứng dụng .....</b>	<b>31</b>
<b>Hình 19: Usecase chức năng người dùng .....</b>	<b>32</b>
<b>Hình 20: Usecase Quản lý thiết bị.....</b>	<b>41</b>
<b>Hình 21: Usecase Xem và điều khiển thiết bị .....</b>	<b>43</b>
<b>Hình 22: Usecase Thông báo .....</b>	<b>45</b>
<b>Hình 23: Usecase Cảnh báo và báo động .....</b>	<b>47</b>
<b>Hình 24: Sơ đồ lớp .....</b>	<b>48</b>
<b>Hình 25: Sơ đồ tuần tự đăng nhập .....</b>	<b>49</b>
<b>Hình 26: Sơ đồ tuần tự đăng ký .....</b>	<b>50</b>
<b>Hình 27: Sơ đồ tuần tự quên mật khẩu.....</b>	<b>50</b>
<b>Hình 28: Sơ đồ tuần tự thay đổi mật khẩu .....</b>	<b>51</b>
<b>Hình 29: Sơ đồ tuần tự cập nhật thông tin cá nhân .....</b>	<b>51</b>
<b>Hình 30: Sơ đồ tuần tự kết nối thiết bị.....</b>	<b>52</b>
<b>Hình 31: Sơ đồ tuần tự cập nhật thông tin thiết bị .....</b>	<b>52</b>
<b>Hình 32: Sơ đồ tuần tự xóa thiết bị .....</b>	<b>53</b>
<b>Hình 33: Sơ đồ tuần tự hiển thị dữ liệu cảm biến .....</b>	<b>53</b>
<b>Hình 34: Sơ đồ tuần tự điều khiển thiết bị .....</b>	<b>54</b>
<b>Hình 35: Sơ đồ tuần tự hiển thị biểu đồ thống kê .....</b>	<b>54</b>
<b>Hình 36: Sơ đồ tuần tự thông báo.....</b>	<b>55</b>
<b>Hình 37: Sơ đồ tuần tự báo động .....</b>	<b>55</b>
<b>Hình 38: Sơ đồ hoạt động đăng nhập .....</b>	<b>56</b>
<b>Hình 39: Sơ đồ hoạt động tự động đăng nhập .....</b>	<b>57</b>
<b>Hình 40: Sơ đồ hoạt động đăng ký .....</b>	<b>58</b>
<b>Hình 41: Sơ đồ hoạt động quên mật khẩu .....</b>	<b>59</b>
<b>Hình 42: Sơ đồ hoạt động thay đổi mật khẩu .....</b>	<b>60</b>
<b>Hình 43: Sơ đồ hoạt động cập nhật thông tin cá nhân .....</b>	<b>61</b>
<b>Hình 44: Sơ đồ hoạt động kết nối thiết bị .....</b>	<b>62</b>
<b>Hình 45: Sơ đồ hoạt động cập nhật thông tin thiết bị .....</b>	<b>63</b>
<b>Hình 46: Sơ đồ hoạt động xóa thiết bị .....</b>	<b>64</b>
<b>Hình 47: Sơ đồ hoạt động hiển thị dữ liệu cảm biến .....</b>	<b>65</b>
<b>Hình 48: Sơ đồ hoạt động thống kê .....</b>	<b>66</b>

<b>Hình 49: Sơ đồ hoạt động thông báo .....</b>	66
<b>Hình 50: Giao diện đăng nhập .....</b>	70
<b>Hình 51: Giao diện quên mật khẩu .....</b>	71
<b>Hình 52: OTP Gửi về Gmail.....</b>	72
<b>Hình 53: OTP Khi được nhập đúng .....</b>	72
<b>Hình 54: Giao diện đăng ký tài khoản .....</b>	73
<b>Hình 55: Trang chủ .....</b>	74
<b>Hình 56: Biểu đồ thống kê .....</b>	75
<b>Hình 57: Drawer danh mục.....</b>	76
<b>Hình 58: Giao diện cấu hình ứng dụng .....</b>	77
<b>Hình 59: Giao diện lịch sử thông báo .....</b>	78
<b>Hình 60: Giao diện chỉnh sửa thông tin cá nhân .....</b>	79
<b>Hình 61: Giao diện thêm thiết bị .....</b>	80
<b>Hình 62: Pop up xác nhận xóa thiết bị .....</b>	81
<b>Hình 63: Giao diện chỉnh sửa thông tin thiết bị .....</b>	82
<b>Hình 64: Giao diện hiển thị các chỉ số lấy được từ cảm biến .....</b>	83
<b>Hình 65: Thông báo gửi về .....</b>	84
<b>Hình 66: Sản phẩm lắp đặt thực tế.....</b>	86

## DANH MỤC BẢNG BIÊU

Bảng 1: Bảng so sánh các framework cho lập trình mobile .....	13
Bảng 2 : Bảng so sánh một vài framework cho lập trình IOT .....	19
Bảng 3: Bảng so sánh HTTP và WebSocket .....	24
Bảng 4: Bảng đặc tả ca sử dụng đăng ký .....	33
Bảng 5: Bảng đặc tả ca sử dụng đăng nhập .....	35
Bảng 6: Bảng đặc tả ca sử dụng thay đổi mật khẩu.....	36
Bảng 7: Bảng đặc tả ca sử dụng cập nhật thông tin.....	38
Bảng 8: Bảng đặc tả ca sử dụng thay đổi giao diện người dùng .....	38
Bảng 9: Bảng đặc tả ca sử dụng thay đổi ngôn ngữ hiển thị .....	38
Bảng 10 : Bảng đặc tả ca sử dụng quên mật khẩu .....	40
Bảng 11: Bảng đặc tả ca sử dụng kết nối thiết bị .....	42
Bảng 12: Bảng đặc tả ca sử dụng sửa thông tin thiết bị.....	42
Bảng 13: Bảng đặc tả ca sử dụng xóa thiết bị .....	43
Bảng 14: Bảng đặc tả ca sử dụng hiển thị thông tin cảm biến .....	44
Bảng 15: Bảng đặc tả ca sử dụng điều khiển thiết bị .....	44
Bảng 16: Bảng đặc tả ca sử dụng hiển thị biểu đồ thống kê.....	44
Bảng 17: Bảng đặc tả ca sử dụng nhận thông báo .....	45
Bảng 18: Bảng đặc tả ca sử dụng tự động xóa thông báo .....	46
Bảng 19: Bảng đặc tả ca sử dụng lưu trữ thông báo.....	46
Bảng 20: Bảng đặc tả ca sử dụng báo động tại hiện trường.....	47
Bảng 21: Bảng đặc tả ca sử dụng gửi dữ liệu.....	48
Bảng 22: Collection User .....	67
Bảng 23: Collection Device .....	67
Bảng 24: Collection Sensordata_Raw .....	67
Bảng 25: Collection SensorData .....	68
Bảng 26: Bảng cấu hình lắp đặt cảm biến.....	69
Bảng 27: Ghi nhận Log trong kiểm thử .....	86
Bảng 28: Kịch bản và kết quả kiểm thử ở phần cứng .....	87
Bảng 29: Bảng tổng kết các chức năng đã phát triển thành công.....	88

# MỤC LỤC

<b>CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI .....</b>	1
1.1    Lý do chọn đề tài.....	1
1.2    Mục tiêu nghiên cứu .....	2
1.3    Đối tượng và phạm vi nghiên cứu .....	4
1.4    Phương pháp nghiên cứu .....	5
<b>CHƯƠNG II: TỔNG QUAN VỀ CÔNG NGHỆ SỬ DỤNG .....</b>	5
2.1    Giới thiệu về ESP32 và các cảm biến liên quan .....	5
2.2    Giới thiệu công nghệ sử dụng .....	9
2.3.1    Ngôn ngữ lập trình Dart .....	9
2.3.2    Framework Flutter.....	10
2.3.3    Firebase .....	14
2.3.4    Node js và framework Express js.....	17
2.3.5    Bảo mật Json Web Token.....	20
2.3.6    Giới thiệu về web socket server.....	22
2.3.7    Kiến trúc hệ thống IoT.....	25
<b>CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....</b>	26
3.1    Yêu cầu hệ thống.....	26
3.1.1    Yêu cầu phần cứng.....	26
3.1.2    Yêu cầu phần mềm .....	26
3.2    Mô tả bài toán .....	27
3.2.1    Chức năng Người dùng .....	27
3.2.2    Quản lý Thiết bị.....	29
3.2.3    Xem và Điều khiển Thiết bị .....	30
3.2.4    Cảnh báo và Báo động .....	30
3.3    Sơ đồ Usecase .....	31
3.3.1    Chức năng người dùng.....	32
3.3.2    Quản lý thiết bị .....	41
3.3.3    Xem và điều khiển thiết bị .....	43
3.3.4    Thông báo.....	45
3.3.5    Cảnh báo báo động .....	47
3.4    Sơ đồ lớp .....	48
3.5    Sơ đồ tuần tự .....	49
3.6    Sơ đồ hoạt động .....	56
3.7    Thiết kế cơ sở dữ liệu.....	67
<b>CHƯƠNG IV: TRIỂN KHAI HỆ THỐNG .....</b>	69
4.1    Cấu hình ESP32 và cảm biến.....	69

4.2	Giao diện ứng dụng .....	70
4.3	Kiểm thử hệ thống .....	85
CHƯƠNG V: KẾT QUẢ VÀ ĐÁNH GIÁ .....		88
5.1	Kết quả đạt được .....	88
5.2	Hạn chế của hệ thống .....	88
5.3	Đề xuất hướng phát triển .....	89
TÀI LIỆU THAM KHẢO .....		91

# CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI

## 1.1 Lý do chọn đề tài

### ❖ Bối cảnh và thực trạng vấn đề

Cháy nổ là một trong những thảm họa gây thiệt hại nghiêm trọng về cả tài sản lẫn tính mạng con người. Những năm gần đây, số lượng vụ cháy tại các khu chung cư, nhà máy, kho bãi, trung tâm thương mại và hộ gia đình ngày càng gia tăng, gây nên những hậu quả nghiêm trọng. Theo thống kê, nguyên nhân chủ yếu dẫn đến cháy nổ thường xuất phát từ chập điện, rò rỉ khí gas, bất cẩn trong sinh hoạt, đốt rác bừa bãi hoặc các sự cố kỹ thuật khác. Đặc biệt, các vụ cháy diễn ra vào ban đêm hoặc khi không có người trong khu vực dễ dẫn đến hậu quả nghiêm trọng hơn do không kịp thời phát hiện và xử lý.

Mặc dù đã có nhiều biện pháp phòng cháy chữa cháy (PCCC) được triển khai, nhưng phần lớn vẫn phụ thuộc vào sự quan sát của con người hoặc hệ thống cảnh báo cục bộ mà không có khả năng giám sát từ xa và thông báo kịp thời cho người dùng. Các hệ thống cảnh báo cháy truyền thống thường bị hạn chế về phạm vi hoạt động, tính linh hoạt và khả năng tích hợp với các thiết bị thông minh.

### ❖ Sự phát triển của công nghệ và ứng dụng IoT trong phòng cháy chữa cháy

Với sự phát triển mạnh mẽ của công nghệ Internet of Things (IoT), việc tích hợp cảm biến thông minh và hệ thống giám sát từ xa vào các giải pháp cảnh báo cháy đã mở ra nhiều hướng đi mới trong lĩnh vực PCCC. IoT giúp các thiết bị có thể kết nối với nhau qua mạng Internet, thu thập dữ liệu theo thời gian thực và tự động gửi cảnh báo khi phát hiện dấu hiệu bất thường.

Trong đó, ESP32 là một dòng vi điều khiển mạnh mẽ, có khả năng kết nối WiFi và tích hợp nhiều cảm biến, cho phép thu thập dữ liệu từ môi trường xung quanh một cách chính xác. Đồng thời, hệ thống cũng có thể điều khiển từ xa các thiết bị phòng cháy như hệ thống phun nước tự động hoặc relay ngắn nguồn điện, giúp giảm thiểu rủi ro và thiệt hại khi xảy ra sự cố cháy nổ.

❖ Ý nghĩa và tính thực tiễn của đề tài

Việc nghiên cứu và triển khai hệ thống IoT giám sát và cảnh báo cháy từ xa mang lại nhiều lợi ích quan trọng:

- Giám sát theo thời gian thực: Hệ thống thu thập dữ liệu môi trường liên tục, giúp người dùng theo dõi từ xa trên điện thoại di động mà không cần có mặt trực tiếp tại hiện trường.
- Cảnh báo kịp thời: Khi phát hiện nhiệt độ cao bất thường, độ ẩm thấp hoặc nồng độ khói vượt ngưỡng, hệ thống sẽ gửi thông báo ngay lập tức đến người dùng qua ứng dụng di động hoặc tin nhắn.
- Tích hợp điều khiển từ xa: Người dùng có thể chủ động tắt/mở relay để điều khiển thiết bị
- Tăng cường an toàn cho nhà ở và doanh nghiệp: Hệ thống phù hợp để ứng dụng trong nhà thông minh (Smart Home), nhà xưởng, kho bãi, văn phòng, siêu thị và nhiều môi trường khác.

## 1.2 Mục tiêu nghiên cứu

❖ Hệ thống giám sát và cảnh báo cháy thông minh dựa trên nền tảng IoT

Trong bối cảnh những rủi ro về hỏa hoạn luôn rình rập, việc chủ động giám sát và ứng phó kịp thời trở nên vô cùng cấp thiết. Hệ thống này không chỉ thu thập và phân tích dữ liệu môi trường quan trọng như nhiệt độ, độ ẩm và nồng độ khói một cách liên tục, mà còn sở hữu khả năng đưa ra những cảnh báo tức thì, giúp người dùng nắm bắt tình hình và hành động nhanh chóng. Hơn thế nữa, sự tiện lợi được đặt lên hàng

đầu khi người dùng có thể dễ dàng theo dõi và điều khiển hệ thống từ xa thông qua một ứng dụng di động trực quan.

- **Trải nghiệm người dùng:** Khai thác sức mạnh của ngôn ngữ Dart và framework Flutter để xây dựng một ứng dụng di động đa nền tảng mượt mà và trực quan. Ứng dụng này không chỉ hiển thị dữ liệu cảm biến một cách dễ hiểu mà còn cho phép người dùng điều khiển các thiết bị một cách linh hoạt.
- **Bộ xử lý trung tâm:** Với vi điều khiển ESP32 mạnh mẽ có khả năng thu thập dữ liệu chính xác từ các cảm biến nhiệt độ, độ ẩm và khói, sau đó truyền tải thông tin này một cách đáng tin cậy đến máy chủ.
- **API Server:** Một server API vững chắc được xây dựng trên nền tảng Node.js và framework Express.js sẽ đóng vai trò là cầu nối giao tiếp giữa các thiết bị và ứng dụng di động. Chúng tôi đặc biệt chú trọng đến việc bảo mật thông tin người dùng và dữ liệu hệ thống thông qua cơ chế xác thực và phân quyền bằng JWT.
- **Cơ sở dữ liệu MongoDB:** Linh hoạt và mạnh mẽ sẽ là nơi lưu trữ mọi thông tin quan trọng của hệ thống, từ hồ sơ người dùng, danh sách thiết bị đến lịch sử chi tiết của dữ liệu cảm biến.
- **Nắm vững xử lý thời gian thực:** đi sâu vào nghiên cứu các kỹ thuật lập trình IoT, đặc biệt là cơ chế truyền dữ liệu theo thời gian thực bằng WebSocket, đảm bảo mọi thay đổi của môi trường đều được cập nhật tức thì đến người dùng. Đồng thời, việc tích hợp Firebase Cloud Messaging (FCM) sẽ giúp hệ thống gửi cảnh báo khẩn cấp một cách hiệu quả.
- **Đảm bảo phản ứng nhanh chóng trong mọi tình huống:** Mục tiêu là triển khai chức năng cảnh báo cháy khẩn cấp đến thiết bị di động ngay cả khi ứng dụng không hoạt động. Điều này đảm bảo rằng hệ

thống luôn trong trạng thái sẵn sàng, đưa ra những cảnh báo kịp thời và chính xác trong những khoảnh khắc quan trọng nhất.

### 1.3 Đối tượng và phạm vi nghiên cứu

#### ❖ Đối tượng nghiên cứu

- Người dùng cuối: Là những người sử dụng hệ thống để nhận những cảnh báo, họ có thể là chủ nhà, chủ doanh nghiệp....
- Thiết bị IoT: Bao gồm vi điều khiển ESP32 và các cảm biến (nhiệt độ, độ ẩm, khói,...) được sử dụng để thu thập dữ liệu môi trường.
- Ứng dụng và hệ thống phần mềm: Gồm ứng dụng di động (Flutter), server API (Node.js/Express), cơ sở dữ liệu (MongoDB), và hệ thống cảnh báo (Firebase Cloud Messaging, WebSocket).

#### ❖ Phạm vi nghiên cứu

- Phạm vi kỹ thuật

- Thiết kế và lập trình hệ thống IoT đơn giản với một số cảm biến cơ bản để giám sát môi trường (nhiệt độ, độ ẩm, khói).
- Xây dựng ứng dụng di động Flutter phục vụ điều khiển và hiển thị dữ liệu theo thời gian thực.
- Phát triển API backend bằng Node.js/Express để xử lý dữ liệu, quản lý người dùng và thiết bị.
- Sử dụng MongoDB để lưu trữ thông tin người dùng, thiết bị và dữ liệu cảm biến.
- Tích hợp Firebase Cloud Messaging (FCM) để gửi thông báo cảnh báo cháy đến thiết bị di động kể cả khi ứng dụng đang đóng.

- Phạm vi ứng dụng

- Hệ thống được triển khai mô phỏng trong quy mô hộ gia đình hoặc doanh nghiệp nhỏ.
- Không bao gồm các yếu tố nâng cao như trí tuệ nhân tạo (AI), học máy (machine learning) hay hệ thống phòng cháy tự động (như phun nước, mở cửa...).

## 1.4 Phương pháp nghiên cứu

- Phương pháp khảo sát – phỏng vấn người dùng: Tiến hành trao đổi, khảo sát và phỏng vấn các đối tượng người dùng tiềm năng như chủ nhà, chủ doanh nghiệp nhỏ, hoặc nhân viên kỹ thuật để tìm hiểu nhu cầu thực tế trong việc giám sát và cảnh báo cháy. Qua đó, xác định các tính năng cần thiết của hệ thống.
- Phương pháp nghiên cứu tài liệu: Thu thập, phân tích các tài liệu học thuật, bài báo khoa học liên quan đến lĩnh vực Internet of Things (IoT), hệ thống báo cháy, và an toàn phòng cháy chữa cháy. Đồng thời tìm hiểu các quy định, tiêu chuẩn kỹ thuật của Nhà nước về hệ thống cảnh báo cháy nhằm đảm bảo giải pháp đề xuất có tính thực tiễn và khả thi.

## CHƯƠNG II: TỔNG QUAN VỀ CÔNG NGHỆ SỬ DỤNG

### 2.1 Giới thiệu về ESP32 và các cảm biến liên quan



**Hình 1 : Hình ảnh thực tế ESP32**

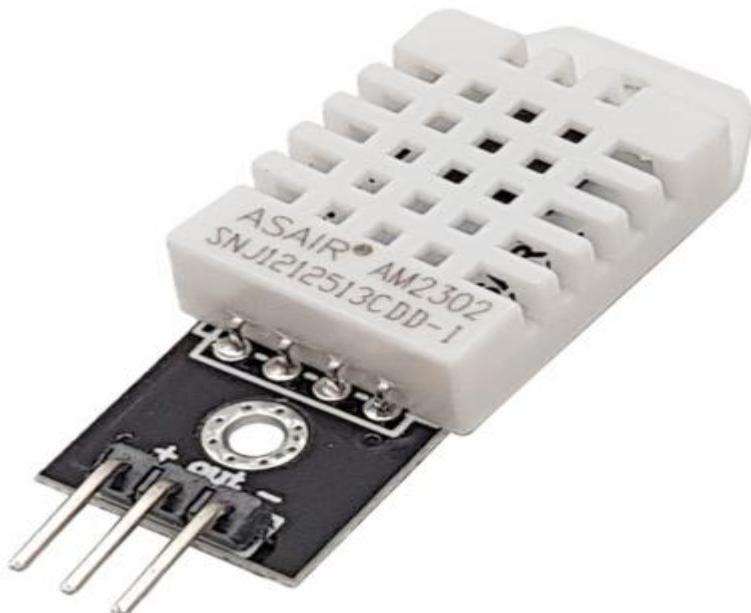
- Vi điều khiển ESP32 là một vi điều khiển tích hợp WiFi và Bluetooth. Đây là phiên bản nâng cấp của dòng ESP8266, với hiệu năng mạnh mẽ hơn và nhiều tính năng tích hợp hơn, phù hợp với các ứng dụng IoT (Internet of Things) hiện đại.[1]

### Các đặc điểm nổi bật của ESP32:

- Bộ xử lý mạnh mẽ: Dual-core Tensilica Xtensa LX6, tốc độ lên đến 240MHz. [1]
- RAM và bộ nhớ: 520KB SRAM, hỗ trợ Flash ngoài. [1]
- Tích hợp WiFi 802.11 b/g/n và Bluetooth 4.2 (Classic + BLE). [1]
- Số lượng chân I/O phong phú: Hỗ trợ ADC, DAC, SPI, I2C, UART,... [1]
- Tiêu thụ điện năng thấp: Hỗ trợ nhiều chế độ ngủ phù hợp với thiết bị chạy pin. [1]

Các cảm biến sử dụng trong hệ thống báo cháy: Để xây dựng hệ thống giám sát và cảnh báo cháy, ESP32 có thể kết nối với nhiều loại cảm biến nhằm thu thập dữ liệu môi trường. Các cảm biến tiêu biểu bao gồm:

- Cảm biến nhiệt độ và độ ẩm (DHT22)



**Hình 2: Hình ảnh thực tế DHT22**

**Chức năng:** Cảm biến DHT22 là một cảm biến kỹ thuật số dùng để đo nhiệt độ và độ ẩm, rất phổ biến trong các ứng dụng IoT, nhà thông minh, và hệ thống cảnh báo môi trường.[2]

- Cảm biến khói và khí gas MQ-2



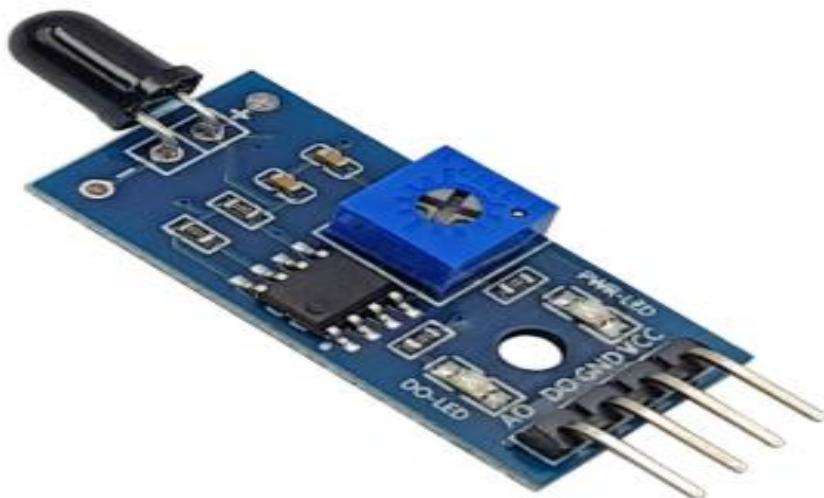
**Hình 3: Cảm biến khói, khí gas MQ2**

**Chức năng:** Phát hiện nồng độ khói, LPG, CO, và các khí cháy khác.

Tín hiệu ngõ ra: Tín hiệu tương tự (analog) hoặc số (digital).

**Ứng dụng:** Phát hiện nhiệt độ cao bất thường hoặc khói – dấu hiệu chính của đám cháy.

- Cảm biến lửa (Flame Sensor)
- 



**Hình 4: Cảm biến lửa**

**Chức năng:** Nhận biết ánh sáng phát ra từ lửa (bước sóng 760~1100 nm).

Ngõ ra: Digital, giúp kích hoạt cảnh báo khi phát hiện lửa trực tiếp.

**Ứng dụng:** Xác nhận cháy xảy ra khi có cả nhiệt, khói, và lửa.

- Vai trò kết hợp của ESP32 và cảm biến: ESP32 giữ vai trò là bộ điều khiển trung tâm, thu thập dữ liệu từ các cảm biến nói trên và xử lý đúng theo logic lập trình. Khi phát hiện dấu hiệu cháy (nhiệt độ cao, khói, khí gas, lửa...), ESP32 có thể:
  - Kích hoạt hệ thống cảnh báo cục bộ như loa, LED.

## 2.2 Giới thiệu công nghệ sử dụng

### 2.3.1 Ngôn ngữ lập trình Dart

Dart là một ngôn ngữ lập trình được phát triển bởi Google, với mục tiêu xây dựng các ứng dụng có hiệu suất cao trên nhiều nền tảng như web, thiết bị di động,... Dart đóng vai trò cốt lõi trong nền tảng Flutter, framework cho việc lập trình cross-platform app để xây dựng ứng dụng có thể hoạt động được trên đa nền tảng. Ngôn ngữ Dart được thiết kế với cú pháp hiện đại, dễ đọc và dễ học, đặc biệt phù hợp với những người quen thuộc với Java, JavaScript hoặc C#[2]. Dart hỗ trợ :

- Hệ thống kiểu mạnh (Sound typing): Biến luôn giữ đúng kiểu dữ liệu khai báo. [2]
- Null Safety: Biến không thể null nếu không được phép rõ ràng. [2]
- Async/Await: Hỗ trợ lập trình bất đồng bộ một cách tự nhiên. [2]
- Generators, Streams, Arrow Syntax, Getters/Setters: Giúp viết code ngắn gọn và hiệu quả. [2]

Đặc điểm nổi bật của Dart:

- Hiệu suất cao: Dart hỗ trợ biên dịch mã nguồn sang mã máy (native) bằng AOT (Ahead-of-Time) và hỗ trợ JIT (Just-in-Time) trong quá trình phát triển, giúp đẩy nhanh vòng lặp phát triển (hot reload). [2]
- Đa nền tảng: Dart có thể biên dịch sang JavaScript để chạy trên trình duyệt, hoặc sang mã máy để chạy trên Android, iOS, Windows, macOS, Linux. [2]
- Ngôn ngữ an toàn kiểu (type-safe): Dart sử dụng kiểm tra kiểu tĩnh và hỗ trợ null safety, giúp tránh lỗi null trong quá trình runtime. [2]
- Thân thiện với nhà phát triển: Dart tích hợp các công cụ như format, phân tích cú pháp (analyzer), kiểm thử (test) và hỗ trợ quản lý gói qua pub.dev. [2]

Nền tảng biên dịch: Dart cung cấp khả năng biên dịch linh hoạt phù hợp với từng nền tảng:

- Dart Native: Sử dụng JIT trong quá trình phát triển, AOT để biên dịch ra mã máy native cho sản phẩm. [2]
- Dart Web: Biên dịch sang JavaScript hoặc WebAssembly (Wasm) để chạy trên trình duyệt. [2]
- Dart Runtime: Quản lý bộ nhớ, thực thi mã, kiểm tra kiểu tại runtime và điều khiển các isolates (đơn vị xử lý độc lập). [2]



*Hình 5: Ngôn ngữ lập trình Dart*

### 2.3.2 Framework Flutter

Flutter là một SDK phát triển ứng dụng di động nguồn mở được tạo ra bởi Google. Nó được sử dụng để phát triển ứng dụng cho Android và iOS, cũng là phương thức chính để tạo ứng dụng cho Google Fuchsia

#### Các tính năng nổi bật của flutter:

- Ngôn ngữ lập trình Dart: Flutter sử dụng Dart, một ngôn ngữ lập trình hiện đại, nhanh chóng và dễ học, giúp phát triển ứng dụng hiệu quả và linh hoạt. [3]
- Hot Reload: Tính năng này cho phép các nhà phát triển nhìn thấy ngay lập tức các thay đổi trong mã nguồn khi ứng dụng đang chạy

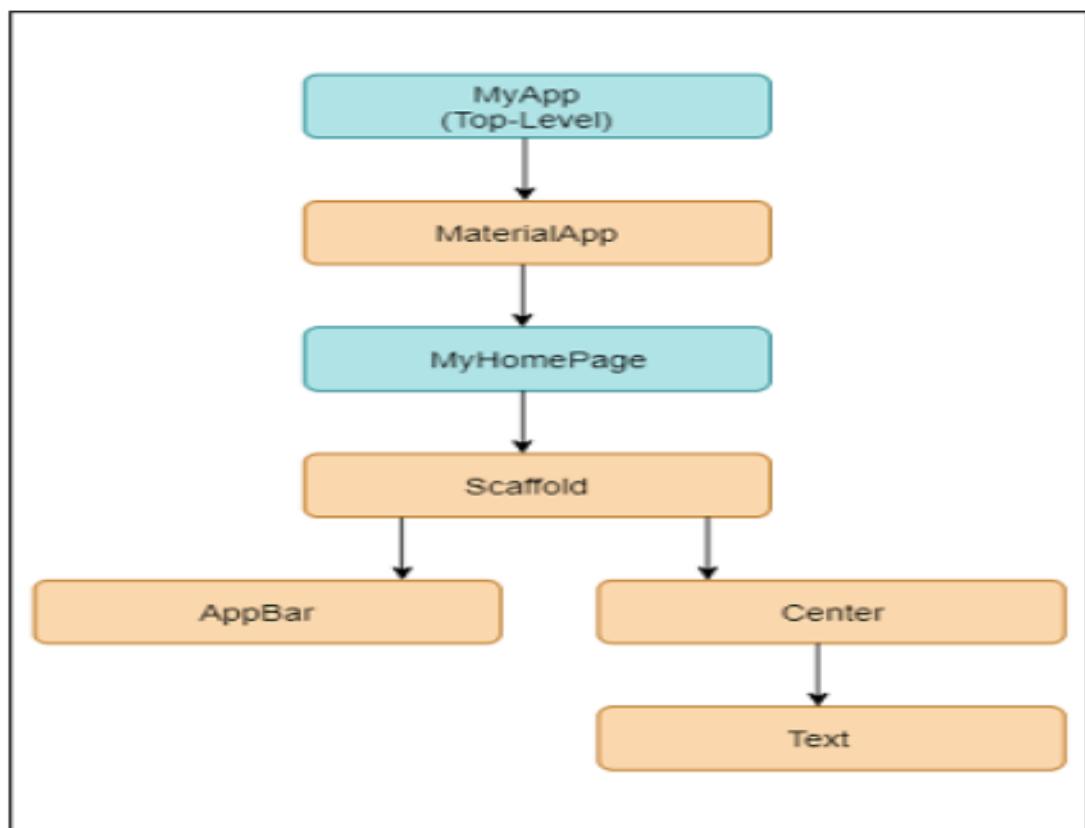
mà không cần phải khởi động lại ứng dụng. Điều này giúp tăng tốc độ phát triển và thử nghiệm. [3]

- Widgets phong phú: Flutter cung cấp một bộ sưu tập widgets phong phú và tùy biến cao, giúp tạo ra giao diện người dùng đẹp mắt và nhất quán trên các nền tảng. [3]
- Hiệu năng cao: Flutter biên dịch trực tiếp sang mã máy, giúp ứng dụng chạy nhanh và mượt mà trên các thiết bị. [3]
- Thiết kế giao diện đẹp mắt: Flutter hỗ trợ thiết kế giao diện người dùng phức tạp và hấp dẫn với các hiệu ứng hoạt hình và chuyển tiếp mượt mà. [3]
- Hỗ trợ đa nền tảng: Với Flutter, bạn có thể phát triển ứng dụng cho cả iOS, Android, web, và máy tính để bàn từ một cơ sở mã nguồn duy nhất, giúp tiết kiệm thời gian và công sức. [3]



**Hình 6 : Framework Flutter**

## Kiến trúc của flutter:



*Hình 7: Kiến trúc của Flutter[4]*

- Trong hình trên, chúng ta có thể thấy rằng tất cả các thành phần đều là các widget có chứa các widget con. Do đó, ứng dụng Flutter tự nó là một widget.
- Framework Flutter có hai bộ widget phù hợp với các ngôn ngữ thiết kế cụ thể. Đây là Material Design cho ứng dụng Android và Cupertino Style cho ứng dụng IOS.

Tiêu chí	Flutter	React native	Kotlin	Swift
Ngôn ngữ	Dart	Java script	Kotlin	Swift
Phát triển bởi	Google	Meta	JetBrains (hợp tác với Google)	Apple
Nền tảng	iOS Android Web Desktop Embedded	iOS Android Web	Android	iOS iPadOS, macOS watchOS
Trải nghiệm ui/ux	Mượt, tùy biến mạnh	Gần native, hạn chế tùy biến	Native	Native
Hot reload	Có	Có	Không	Không
Open source	Có	Có	Có	Không
Cộng đồng	Lớn, phát triển mạnh	Lớn	Nhỏ, đang phát triển	Chỉ phù hợp với hệ sinh thái của apple

**Bảng 1: Bảng so sánh các framework cho lập trình mobile**

Kết luận:

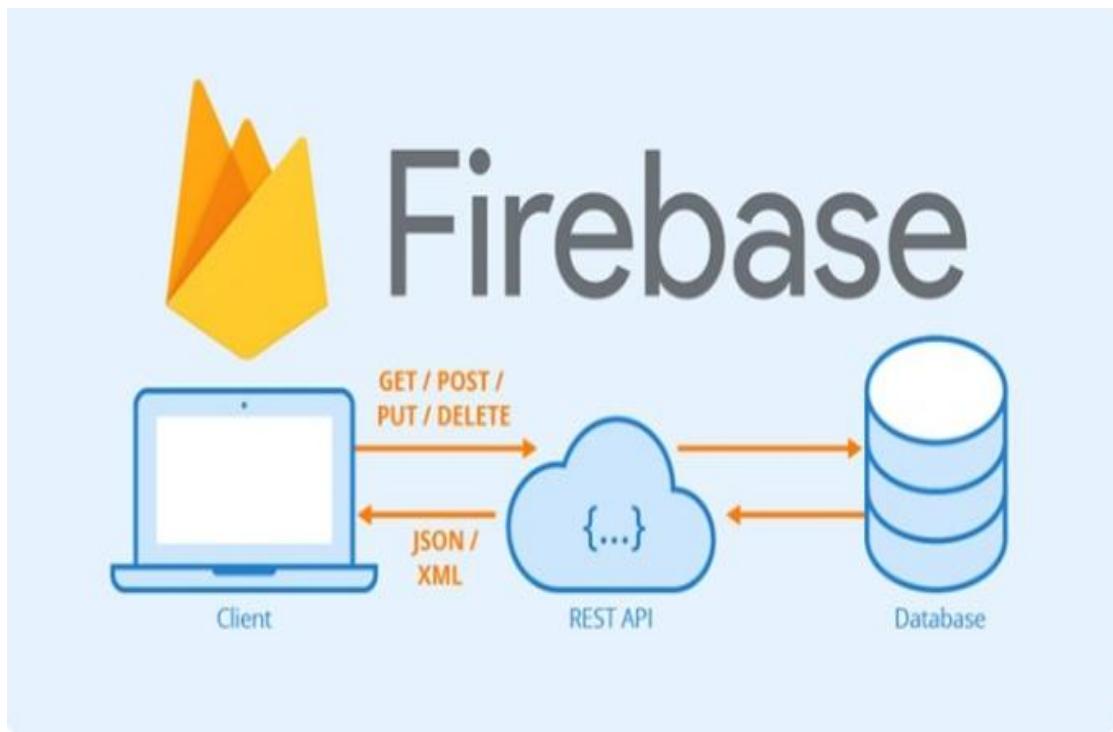
- Một codebase – Nhiều nền tảng: ios, android, web: Tiết kiệm thời gian và chi phí phát triển.
- UI đẹp – Tuỳ biến cao: có thể tạo giao diện theo ý muốn, nhất quán trên mọi nền tảng.

- Hiệu suất cao: App chạy mượt.
- Phát triển nhanh với Hot Reload: Có thể thay đổi code và xem kết quả ngay lập tức mà không cần build lại toàn app. => Tăng tốc độ làm UI, debug, thử nghiệm.
- Tích hợp tốt với hệ sinh thái Google: Flutter hoạt động rất tốt với: Firebase (auth, cloud messaging, realtime db...), Google Maps, Google Pay, Ads,...
- Cộng đồng lớn – Tài liệu nhiều : Flutter có cộng đồng phát triển mạnh, nhiều ví dụ, plugin.

### 2.3.3 Firebase

Firebase là một trong những BaaS (Backend as a Service), tức là một dịch vụ cung cấp các giải pháp backend cho các ứng dụng web và di động. [5]

Firebase được ra đời vào năm 2011 bởi James Tamplin và Andrew Lee với tên gọi ban đầu là Evolve, một nền tảng cung cấp các API để tích hợp tính năng chat vào các trang web. Sau đó họ đã phát triển Evolve thành Firebase và công bố nó vào tháng 4 năm 2012. Đến tháng 10 năm 2014, Firebase đã được Google mua lại và trở thành một phần của [5]



**Hình 8: Firebase và dịch vụ**

Firebase chia làm 2 nhóm sản phẩm chính:

### **Build – Xây dựng ứng dụng**

- Authentication: Đăng nhập bằng email, số điện thoại, Google, Facebook,... [6]
- Firestore / Realtime Database: Lưu trữ và đồng bộ dữ liệu real-time. [6]
- Cloud Functions: Chạy backend không cần server (serverless). [6]
- Firebase Hosting: Hosting cho web tĩnh (nhanh và bảo mật). [6]
- Cloud Storage: Lưu trữ ảnh, video, file lớn. [6]
- Firebase ML: Tích hợp Machine Learning vào app. [6]

### **Vận hành và cải tiến ứng dụng**

- Crashlytics: Theo dõi lỗi crash theo thời gian thực. [6]
- Performance Monitoring: Theo dõi hiệu suất ứng dụng. [6]
- Remote Config: Thay đổi giao diện hoặc tính năng mà không cần update app. [6]
- A/B Testing: Thủ nghiệm tính năng để chọn phương án hiệu quả. [6]

- Analytics: Theo dõi hành vi người dùng, tương thích với Google Analytics. [6]

Lý do lựa chọn sử dụng Firebase Cloud Messaging cho gửi thông báo cho sản phẩm:

- Dịch vụ miễn phí, không cần tự vận hành server gửi thông báo: Firebase Cloud Messaging (FCM) là miễn phí và được Google vận hành, giúp tiết kiệm thời gian và chi phí. Không cần cấu hình hay bảo trì hệ thống gửi thông báo riêng.
- Gửi thông báo đến thiết bị ngay cả khi app đang đóng : Hỗ trợ gửi thông báo ngay cả khi ứng dụng đang chạy nền hoặc bị tắt. Đây là tính năng rất quan trọng quan trọng cho các thiết bị di động.
- Tích hợp dễ dàng với Flutter và Node.js: Firebase có SDK chính thức cho Flutter, Android, iOS, dễ tích hợp. Có thể gửi thông báo thông qua server Node.js thông qua HTTP.
- Hỗ trợ nhiều hình thức thông báo: Gửi thông báo đơn giản. Gửi thông báo có kèm data để xử lý tùy biến. Hỗ trợ gửi đến từng thiết bị, nhóm người dùng, hoặc theo topic.
- Bảo mật và phân quyền dễ quản lý: Hệ thống token xác thực giúp gửi đúng đến thiết bị của người dùng tương ứng. Kết hợp với hệ thống xác thực JWT trong app để phân quyền và gửi thông báo

### 2.3.4 Node.js và framework Express.js

#### ❖ Node.js là gì?

Node.js là một môi trường runtime JavaScript mã nguồn mở, đa nền tảng, được xây dựng trên V8 engine của Chrome. Được thiết kế để tạo ra các ứng dụng mạng có hiệu suất cao, Node.js cung cấp khả năng xử lý bất đồng bộ và không chặn (non-blocking I/O) trên một luồng đơn — lý tưởng cho các ứng dụng real-time và hệ thống quy mô lớn.[7]

#### Đặc điểm nổi bật:

- Node.js kết hợp V8 JavaScript engine với kiến trúc I/O không đồng bộ, giúp xử lý hàng nghìn kết nối đồng thời với độ trễ thấp. [7]
- Hoạt động trên một luồng duy nhất, nhưng xử lý các tác vụ I/O bất đồng bộ, giúp tăng khả năng phản hồi và mở rộng. [7]
- Hơn một triệu package có sẵn thông qua Node Package Manager (NPM), giúp bạn xây dựng mọi thứ từ API đơn giản đến hệ thống phân tán phức tạp. [7]
- Viết cả frontend và backend bằng cùng một ngôn ngữ JavaScript giúp thống nhất công nghệ và tăng tốc phát triển. [7]
- Node.js hỗ trợ scale theo chiều ngang, cho phép ứng dụng mở rộng trên nhiều máy chủ để phục vụ hàng triệu người dùng. [7]



[www.metricsviews.com](http://www.metricsviews.com)

*Hình 9: Logo Node js*

❖ **Giới thiệu về Express js**

Express.js là một "bộ khung" được xây dựng dựa trên Node.js. Nó cung cấp các công cụ, cấu trúc và tính năng giúp bạn xây dựng các ứng dụng web và API một cách dễ dàng và nhanh chóng hơn so với việc sử dụng Node.js thuần.

- Định tuyến: Xác định cách ứng dụng phản hồi các yêu cầu từ client. [8]
- Middleware: Cho phép bạn thực hiện các tác vụ trung gian trong quá trình xử lý request và response. [8]
- Xử lý request và response: Cung cấp các đối tượng và phương thức để làm việc với dữ liệu gửi đến từ client và gửi dữ liệu phản hồi lại. [8]
- Template engine: Tích hợp với nhiều công cụ tạo giao diện động để hiển thị dữ liệu lên trang web. [8]



*Hình 10: Logo Express JS*

Tiêu chí	Node.js + Express	ASP.NET Core	Django (Python)
<b>Ngôn ngữ</b>	Java script	C#	Python
<b>Hỗ trợ websocket</b>	Tốt	Tốt nhưng cấu hình khó .	Tốt
<b>Cộng đồng và thư viện</b>	Lớn với npm	Lớn với nuget	Lớn với pip
<b>Thông báo</b>	Dễ dàng tích hợp firebase	Khó khăn trong việc cấu hình tích hợp firebase	Khó khăn trong việc cấu hình tích hợp firebase
<b>Độ khó</b>	Dễ	Khó	DỄ

*Bảng 2: Bảng so sánh một vài framework cho lập trình IOT*

Vậy với việc sử dụng FCM và Websocket để thông báo và đọc dữ liệu thì node js + express js là sự lựa chọn tối ưu với việc sản phẩm cần hiệu năng cao, realtime và tốc độ phát triển nhanh.

### 2.3.5 Bảo mật Json Web Token

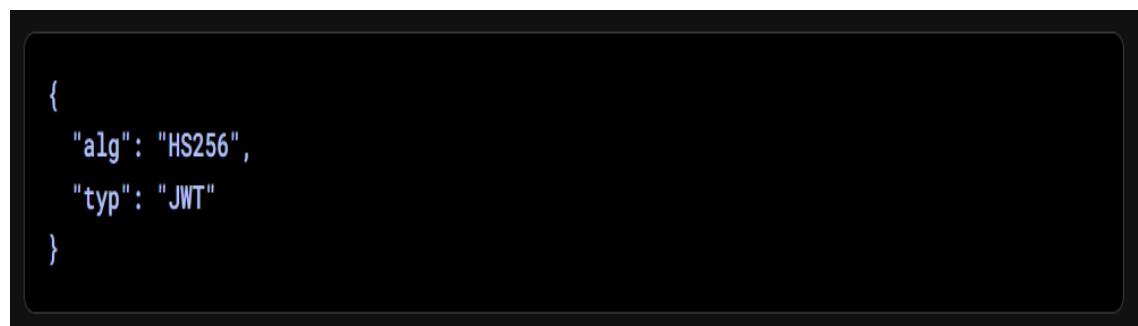
JSON Web Token (JWT) là một chuẩn mở (RFC 7519) cho phép truyền tải thông tin giữa các bên một cách an toàn dưới dạng đối tượng JSON. Thông tin trong JWT được ký số để đảm bảo tính toàn vẹn và xác thực. JWT có thể được ký bằng secret key (HMAC) hoặc public/private key (RSA hoặc ECDSA).

Khi nào nên sử dụng JWT?

- Xác thực: Sau khi người dùng đăng nhập thành công, JWT sẽ được gửi kèm trong mỗi yêu cầu tiếp theo. Nhờ đó, hệ thống xác định quyền truy cập tài nguyên, dịch vụ... JWT rất phổ biến trong Single Sign-On.
- Trao đổi thông tin: JWT giúp truyền thông tin giữa các hệ thống một cách bảo mật. Do có chữ ký, bên nhận có thể xác minh được dữ liệu không bị thay đổi.

JWT bao gồm ba phần chính: Header, Payload, và Signature. Mỗi phần được mã hóa bằng Base64 và được nối với nhau bằng dấu chấm (.)

- Header: Chứa thông tin về thuật toán mã hóa và loại token.



**Hình 11: Header trong JWT[9]**

- Payload: Chứa thông tin (claims) của token. Các claims có thể là các thông tin định danh như tên người dùng, quyền truy cập, hoặc các thông tin khác.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

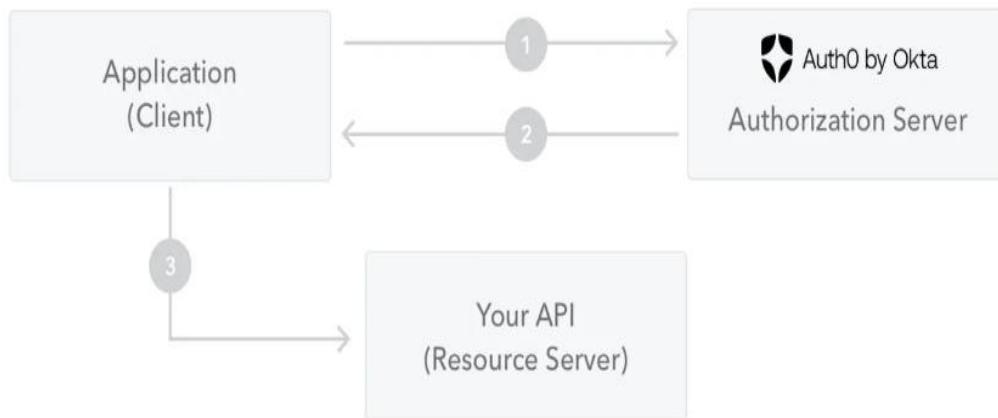
**Hình 12: Payload trong JWT[9]**

- Signature: Được tạo ra bằng cách mã hóa header và payload với một khóa bí mật. Mục đích là để đảm bảo tính toàn vẹn của token.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

**Hình 13: Signature trong JWT[9]**

### Mô phỏng hoạt động của json web token



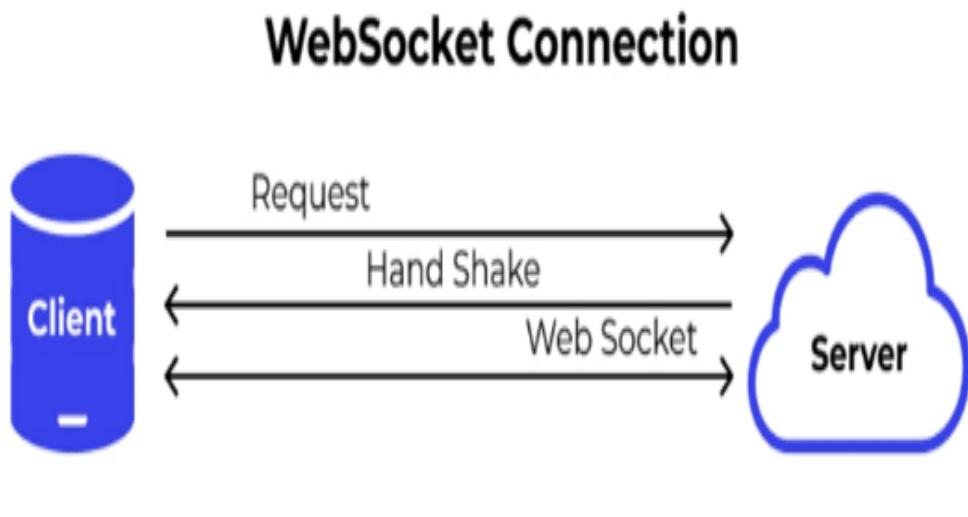
**Hình 14: Mô phỏng hoạt động của JWT[9]**

- Người dùng đăng nhập và hệ thống trả về JWT.
- Ở mỗi lần truy cập tài nguyên, JWT được gửi kèm trong header
- Server xác minh token và cho phép truy cập nếu hợp lệ.

### 2.3.6 Giới thiệu về web socket server

WebSocket là một giao thức truyền thông cung cấp các kênh liên lạc song công hoàn toàn qua một kết nối TCP duy nhất giữa máy khách và máy chủ. Không giống như HTTP truyền thông tuân theo mô hình phản hồi yêu cầu, giao thức này cho phép giao tiếp hai chiều. Điều này có nghĩa là máy khách và máy chủ có thể gửi dữ liệu cho nhau bất cứ lúc nào, giúp dữ liệu được truyền đi nhanh chóng mà không cần phải tải lại trang web. [10]

#### Mô hình hoạt động của Websocket Server



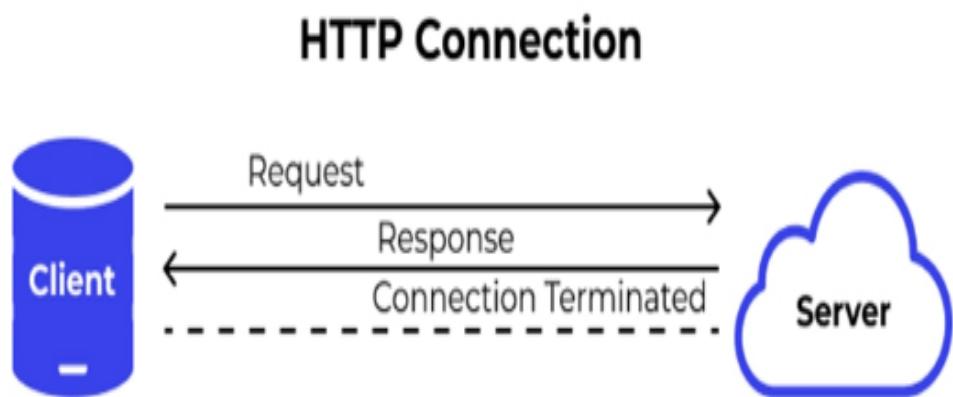
**Hình 15: Minh họa kết nối Web Socket[10]**

- Khi một client như trình duyệt, ứng dụng di động hoặc thiết bị IoT muốn thiết lập kết nối WebSocket với server, nó sẽ khởi tạo bằng cách gửi một HTTP request đặc biệt, gọi là WebSocket handshake. Đây là một cơ chế để server nâng cấp kết nối từ HTTP thông thường sang WebSocket. Nếu server hỗ trợ giao thức WebSocket, nó sẽ phản

hồi bằng một HTTP response chứa tiêu đề xác nhận rằng việc chuyển đổi được chấp thuận. bắt đầu mở websocket

- Sau khi handshake thành công Client và Server có thể gửi dữ liệu cho nhau bất kỳ lúc nào, không cần chờ request hay phản hồi như trong mô hình HTTP.
- Kết nối WebSocket sẽ duy trì mở liên tục trong suốt phiên làm việc, cho đến khi:
  - Một trong hai bên (client hoặc server) chủ động đóng kết nối.
  - Sự cố mạng, lỗi hệ thống hoặc timeout xảy ra khiến kết nối bị ngắt.
- Khi kết nối WebSocket kết thúc, tài nguyên được giải phóng hoàn toàn, đảm bảo không gây tốn bộ nhớ hoặc rò rỉ kết nối.

### Mô hình hoạt động của http



**Hình 16: Kết nối HTTP[10]**

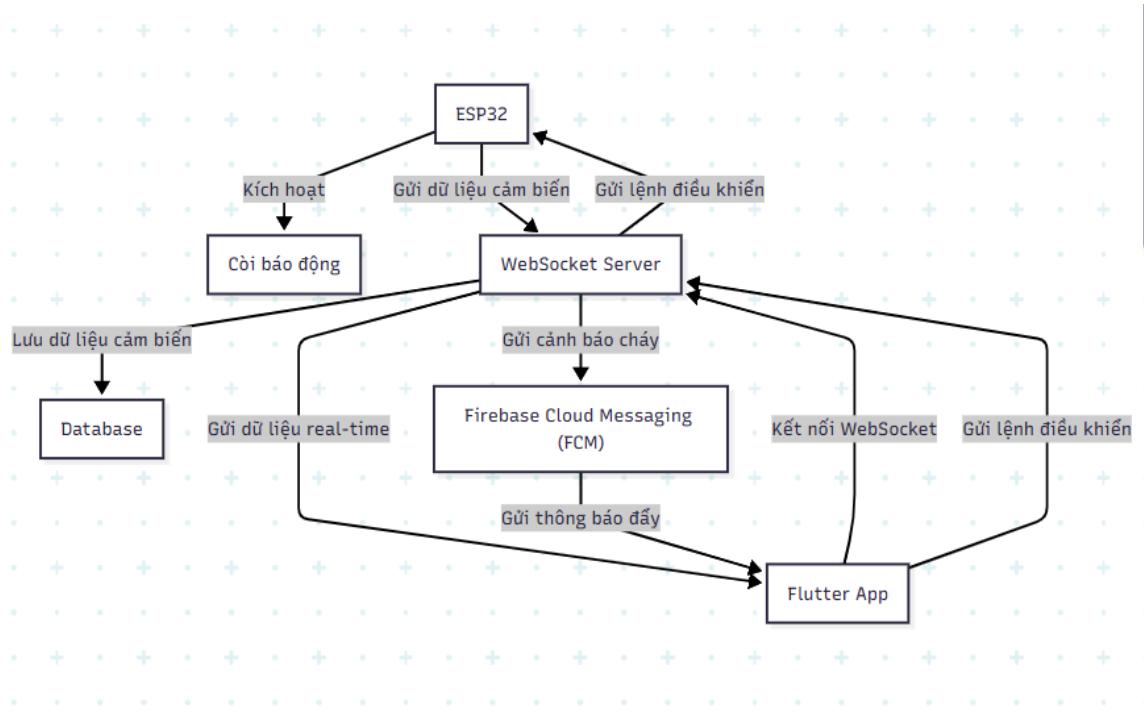
- Client (trình duyệt, app, thiết bị) sẽ gửi một request (yêu cầu) lên server, yêu cầu lấy dữ liệu hoặc thực hiện một hành động.
- Server sẽ xử lý request đó rồi trả về response (phản hồi)

### ❖ So sánh WebSocket HTTP và Websocket Server

Tiêu chí	HTTP	WebSocket
Loại kết nối	Yêu cầu - Phản hồi (Request-Response)	Hai chiều, liên tục (Full-Duplex)
Duy trì trạng thái	Không	Có
Ứng dụng	Trang web tĩnh, ứng dụng không yêu cầu truyền thông liên tục	Ứng dụng thời gian thực, ứng dụng trò chuyện, trò chơi
Tối ưu hóa tài nguyên	Phù hợp cho các yêu cầu đơn lẻ, không liên tục	Giảm tải lên mạng với kết nối liên tục
Tương thích	Rộng rãi trên tất cả trình duyệt và máy chủ	Yêu cầu hỗ trợ từ trình duyệt và máy chủ
Triển khai	Dễ dàng, không yêu cầu cấu hình phức tạp	Có thể yêu cầu kiến thức chuyên sâu về mạng và bảo mật
Tính năng	Phù hợp cho truyền tải dữ liệu không liên tục	Phù hợp cho truyền thông liên tục và thời gian thực

Bảng 3: Bảng so sánh HTTP và WebSocket

### 2.3.7 Kiến trúc hệ thống IoT



**Hình 17: Hình vẽ cách hoạt động của ứng dụng**

#### Mô tả luồng dữ liệu

- ESP32
  - Nhận dữ liệu từ cảm biến nhiệt độ, độ ẩm, mức khói.
  - Gửi dữ liệu đọc được từ cảm biến lên Server
  - Nhận lệnh điều khiển từ server
- WebSocket Server
  - Nhận kết nối từ Flutter client.
  - Gửi dữ liệu về Flutter client
  - Nhận lệnh điều khiển từ Flutter Client
  - Nhận dữ liệu từ ESP32
  - Gửi lệnh điều khiển của Flutter client
  - Gửi thông báo cho Flutter
  - Gửi dữ liệu sensor real-time cho Flutter.

- Flutter App
  - Kết nối WebSocket với ServerJS để:
  - Gửi lệnh điều khiển relay.
  - Nhận dữ liệu từ WebSocket Server
  - Nhận cảnh báo cháy thông qua Firebase Cloud Messaging (FCM) ngay cả khi app không chạy.

## **CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

### **3.1 Yêu cầu hệ thống**

#### **3.1.1 Yêu cầu phần cứng**

- ESP32: Bo mạch vi điều khiển hỗ trợ WiFi/Bluetooth nhận dữ liệu từ cảm biến và giao tiếp với server.
  - Cảm biến nhiệt độ/độ ẩm DHT22
  - Cảm biến khói MQ2.
  - Cảm biến thời gian thực DS3231
  - Led
  - Màn hình
  - Module cài chip
- Máy chủ (Server):
  - Hệ điều hành: Debian 12
  - Chạy môi trường Node.js để vận hành backend.
- Thiết bị người dùng:
  - Điện thoại Android cài đặt ứng dụng Flutter.
  - Kết nối internet ổn định.

#### **3.1.2 Yêu cầu phần mềm**

##### **• Phía thiết bị ESP32:**

- Visual studio code cài đặt PlatformIO để lập trình ESP32.

##### **• Server Backend:**

- Node.js và Thư viện hỗ trợ:

- ExpressJS: Framework API.
  - Socket.IO: Giao tiếp real-time với Flutter app.
  - Mongoose: ORM cho MongoDB.
- **Cơ sở dữ liệu:**
    - MongoDB để lưu trữ dữ liệu
  - **Firebase Cloud Messaging (FCM):**
    - Tài khoản Firebase để gửi thông báo đến Flutter app.
  - **Ứng dụng Flutter:**
    - Flutter SDK
  - **Các yêu cầu khác:**
    - Cấu hình bảo mật JWT để xác thực người dùng khi sử dụng WebSocket

## 3.2 Mô tả bài toán

### 3.2.1 Chức năng Người dùng

#### ❖ Đăng nhập

- Mô tả: Người dùng nhập tài khoản (email hoặc username) và mật khẩu để xác thực với hệ thống.
- Yêu cầu:
  - Kiểm tra thông tin nhập hợp lệ (không bỏ trống, đúng định dạng email).
  - Gửi API /login đến Server Node.js.
  - Server xác thực, nếu đúng trả về JWT Token, nếu sai trả lỗi.
  - App lưu JWT Token cục bộ bằng SharedPreferences để dùng các API khác. Thời hạn sử dụng của token là 7 ngày

#### ❖ Đăng ký

- Mô tả: Người dùng đăng ký tài khoản mới bằng email, số điện thoại, tên đăng nhập, mật khẩu.
- Yêu cầu:

- Kiểm tra tính hợp lệ: tên người dùng chưa tồn tại, email chưa tồn tại, mật khẩu đủ mạnh.
- Gửi API /register đến Server.
- Server lưu thông tin vào MongoDB.

### ❖ Quên mật khẩu

- Mô tả: Người dùng yêu cầu lấy lại mật khẩu bằng email.
- Yêu cầu:
  - Gửi yêu cầu /forgot-password -> Server gửi mã OTP qua email.
  - Người dùng nhập OTP sau khi xác thực đúng OTP thì trả về khung nhập mật khẩu mới trong màn hình forgotpassword\_page mật khẩu mới.
  - Gửi API /reset-password để đặt lại mật khẩu.

### ❖ Thay đổi mật khẩu

- Mô tả: Là 1 phần của việc cập nhật thông tin cá nhân, nếu người dùng không hài lòng với mật khẩu hiện tại thì có thể đổi mật khẩu cũ
- Yêu cầu:
  - Xác thực JWT Token.
  - Nhập mật khẩu cũ -> Nhập mật khẩu mới -> Xác nhận mật khẩu mới. Nếu mật khẩu cũ trùng khớp với mật khẩu hiện tại và mật khẩu mới được nhập trùng với mật khẩu nhập ở ô xác nhận mật khẩu thì cho phép thay đổi mật khẩu.
  - Server kiểm tra và cập nhật mật khẩu mới trong database.

### ❖ Cập nhật thông tin cá nhân

- Mô tả: Người dùng thay đổi tên, số điện thoại.
- Yêu cầu:
  - Gửi API /update-profile với dữ liệu mới.
  - Kiểm tra không cho phép trùng username và email
  - Server Node.js cập nhật lại MongoDB.

#### ❖ **Dark Mode**

- Mô tả: Cho phép người dùng chuyển đổi giao diện sáng/tối.
- Yêu cầu:
  - Lưu trạng thái Dark/Light vào SharedPreferences.
  - Không cần yêu cầu server.

#### ❖ **Thay đổi ngôn ngữ**

- Mô tả: Người dùng chọn ngôn ngữ giao diện.
- Yêu cầu:
  - Sử dụng thư viện easy\_localization.
  - Lưu ngôn ngữ sử dụng vào SharedPreferences.
  - Lưu trạng thái lựa chọn ngôn ngữ cục bộ.

### 3.2.2 Quản lý Thiết bị

#### ❖ **Thêm thiết bị**

- Mô tả: Người dùng thêm ESP32 mới vào tài khoản của mình.
- Yêu cầu:
  - Quét wifi, khi khởi động thiết bị lần đầu thiết bị sẽ phát wifi.
  - Gửi API /add-device (DeviceID, UserID).
  - Server xác nhận DeviceID hợp lệ và lưu liên kết.

#### ❖ **Sửa thông tin thiết bị**

- Mô tả: Người dùng cập nhật tên gọi hoặc vị trí thiết bị.
- Yêu cầu:
  - Gửi API /update-device với dữ liệu mới.
  - Server cập nhật trong MongoDB.

#### ❖ **Xóa thiết bị**

- Mô tả: Người dùng xóa thiết bị không còn sử dụng.
- Yêu cầu:
  - Gửi API /delete-device với DeviceID.
  - Server xóa liên kết thiết bị khỏi tài khoản.

### 3.2.3 Xem và Điều khiển Thiết bị

#### ❖ Hiển thị thông tin cảm biến

- Mô tả: Ứng dụng nhận real-time dữ liệu nhiệt độ, độ ẩm, khí ga đọc được từ thiết bị.
- Yêu cầu:
  - Esp32 gửi dữ liệu đọc được lên server Node js
  - Server Node Js nhận được dữ liệu.
  - App Flutter kết nối WebSocket tới Server Node.js.

#### ❖ Điều khiển thiết bị

- Mô tả: Người dùng điều khiển (bật/tắt relay, báo động) thiết bị từ app.
- Yêu cầu:
  - Gửi lệnh điều khiển.
  - Server gửi yêu cầu tới thiết bị.

#### ❖ Biểu đồ

- Mô tả: Hiển thị dữ liệu lịch sử dạng biểu đồ nhiệt độ, độ ẩm, khói.
- Yêu cầu:
  - App gửi API truy vấn lịch sử
  - Server trả về dữ liệu từ MongoDB.
  - App hiển thị biểu đồ thể hiện các chỉ số trung bình của từng ngày trong 30 ngày.

### 3.2.4 Cảnh báo và Báo động

#### ❖ Gửi thông báo về điện thoại

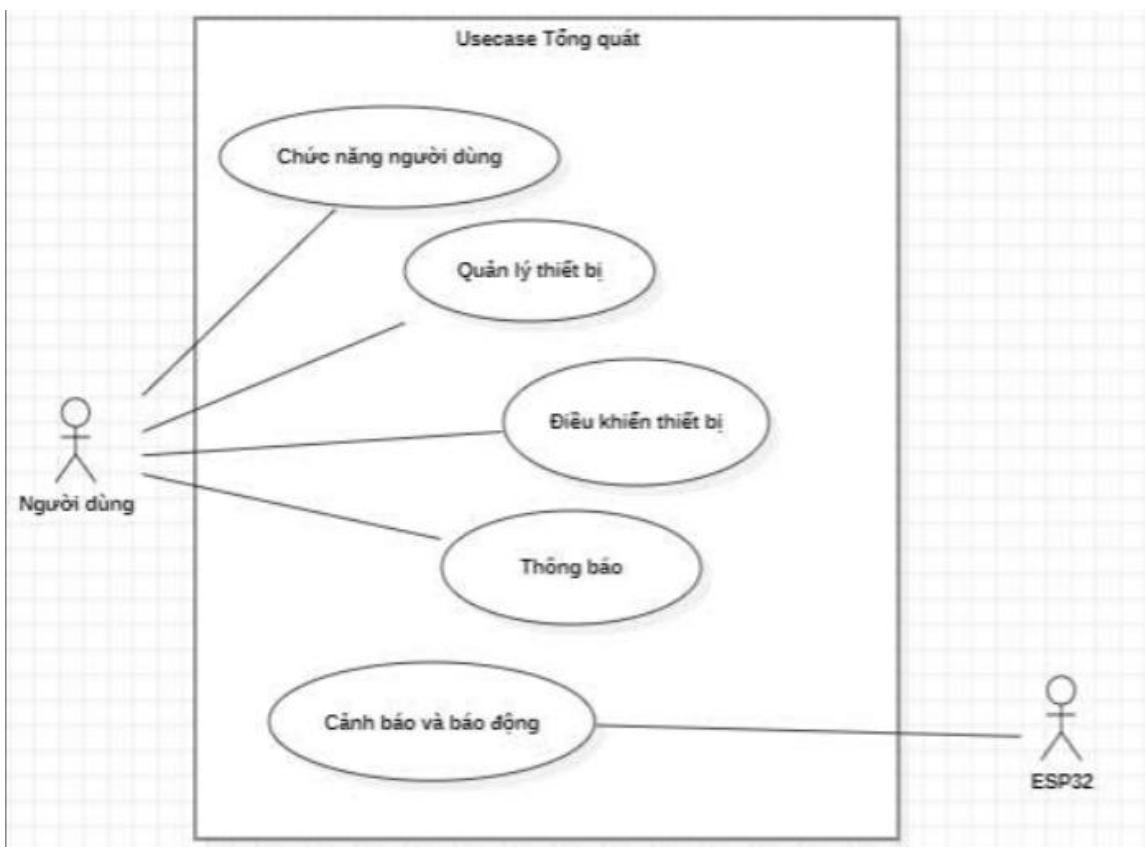
- Mô tả: Gửi thông báo cho app khi các chỉ số vượt ngưỡng.
- Yêu cầu:
  - Server Node.js nhận dữ liệu từ ESP32
  - Server sử dụng Firebase Cloud Messaging (FCM) để gửi push notification.

- Ứng dụng phải nhận được thông báo ở 3 trường hợp: Khi đang sử dụng app, Khi “app back ground” tức ở trạng thái đa nhiệm và khi app đã bị tắt hoàn toàn tức terminated app. Chỉ cần có Internet thông báo sẽ được gửi về điện thoại.
- Ứng dụng sẽ lưu thông báo vào bộ nhớ tạm để hiển thị danh sách các thông báo, click vào để xem chi tiết thông báo.
- Ứng dụng sẽ tự động xóa thông báo sau 1 tuần.

#### ❖ Báo động ở thiết bị

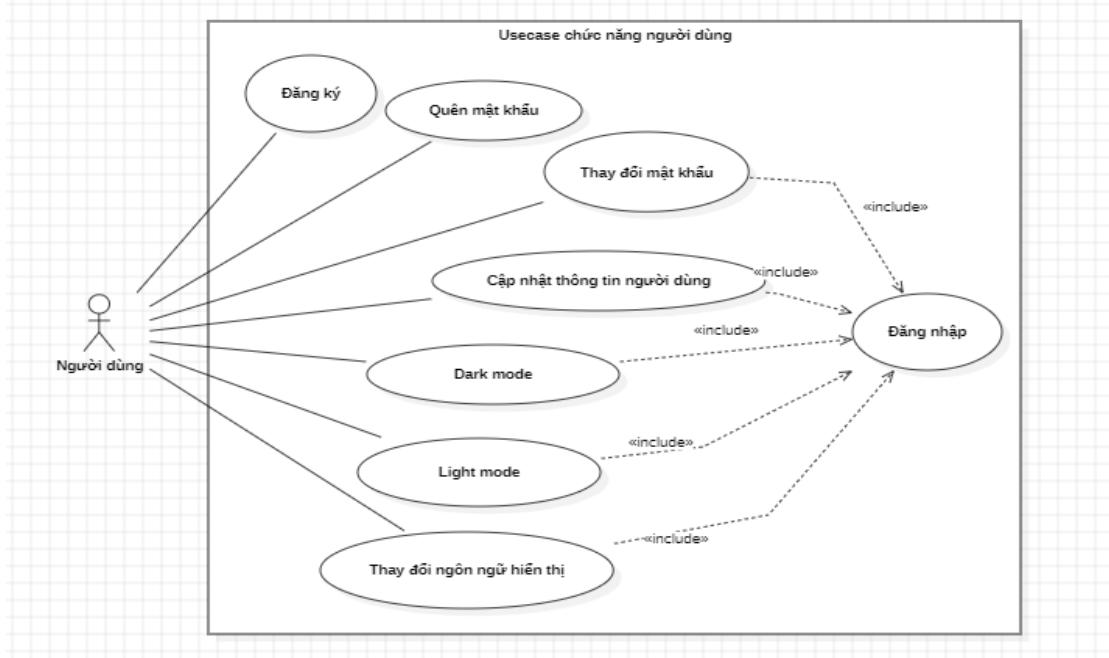
- Mô tả: Khi vượt ngưỡng an toàn, ESP32 tự bật còi/báo động ngay tại chỗ.
- Yêu cầu: ESP32 lập trình sẵn điều kiện kích hoạt báo động vật lý.

### 3.3 Sơ đồ Usecase



**Hình 18: Usecase tổng quát của ứng dụng**

### 3.3.1 Chức năng người dùng



**Hình 19: Usecase chức năng người dùng**

Tên ca sử dụng	Đăng ký
Tác nhân	Người dùng
Điều kiện đầu vào	
Các luồng sự kiện	<ul style="list-style-type: none"> <li>• Luồng sự kiện chính:             <ul style="list-style-type: none"> <li>- Người dùng mở ứng dụng và truy cập vào màn hình đăng ký.</li> <li>- Người dùng nhập các thông tin bắt buộc</li> <li>- Hệ thống kiểm tra</li> <li>- Nếu hợp lệ, hệ thống lưu thông tin người dùng mới vào cơ sở dữ liệu</li> <li>- Hệ thống thông báo đăng ký thành công và điều hướng người dùng đến màn hình đăng nhập</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>• Luồng báo lỗi</li> </ul> <ul style="list-style-type: none"> <li>- Hệ thống hiển thị thông báo lỗi: “Mật khẩu xác nhận không khớp.” =&gt; người dùng phải nhập lại.</li> <li>- Người dùng để trống một hoặc nhiều trường bắt buộc =&gt; Hệ thống không cho phép gửi dữ liệu và hiển thị thông báo lỗi tương ứng.</li> <li>- Người dùng nhập sai định dạng email =&gt; Hệ thống cảnh báo định dạng sai và yêu cầu nhập lại.</li> <li>- Nếu có kiểm tra độ mạnh mật khẩu, hệ thống hiển thị thông báo: “Mật khẩu quá yếu. Vui lòng chọn mật khẩu an toàn hơn.”</li> <li>- Trong quá trình gửi dữ liệu đến server, nếu xảy ra lỗi kết nối, hệ thống hiển thị thông báo: “Không thể kết nối máy chủ. Vui lòng kiểm tra mạng và thử lại sau.”</li> </ul>
Kết quả trả về	<ul style="list-style-type: none"> <li>- Thành công: Tài khoản được tạo, người dùng được chuyển đến màn hình đăng nhập.</li> <li>- Thất bại: Hiển thị lỗi tương ứng tùy từng luồng phụ, yêu cầu người dùng chỉnh sửa và thử lại.</li> </ul>

**Bảng 4: Bảng đặc tả ca sử dụng đăng ký**

Tên ca sử dụng	<b>Đăng nhập</b>
Tác nhân	Người dùng
Điều kiện đầu vào	Đã đăng ký thành công.
Các luồng sự kiện	<ul style="list-style-type: none"> <li>• Luồng sự kiện chính <ul style="list-style-type: none"> <li>- Người dùng mở ứng dụng và truy cập vào màn hình đăng nhập.</li> <li>- Người dùng nhập thông tin đăng nhập.</li> <li>- Người dùng nhấn nút “Đăng nhập”.</li> <li>- Hệ thống gửi thông tin đến server để kiểm tra.</li> <li>- Nếu thông tin chính xác và tồn tại trong cơ sở dữ liệu: Hệ thống xác thực thành công, Lưu thông tin đăng nhập để tự động đăng nhập ở lần tiếp theo, Người dùng được điều hướng đến màn hình chính của ứng dụng.</li> </ul> </li> <li>• Luồng báo lỗi <ul style="list-style-type: none"> <li>- Người dùng để trống email/username hoặc mật khẩu. Hệ thống hiển thị thông báo: “Vui lòng điền đầy đủ thông tin.”</li> <li>- Hệ thống kiểm tra nhưng không tìm thấy tài khoản khớp. Hiển thị thông báo lỗi: “Email hoặc mật khẩu không đúng.”</li> <li>- Trong quá trình xác thực, nếu mất kết nối Internet hoặc server phản hồi lỗi: Hiển thị thông báo: “Không thể kết nối đến máy chủ. Vui lòng thử lại sau.”</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>- Hệ thống phát hiện định dạng email không hợp lệ. Hiển thị thông báo: “Email không hợp lệ.”</li> </ul>
Kết quả trả về	<ul style="list-style-type: none"> <li>- <b>Thành công:</b> Người dùng được chuyển đến trang chính của ứng dụng. Hệ thống lưu thông tin để tự động đăng nhập lần sau.</li> <li>- <b>Thất bại:</b> Hiển thị thông báo lỗi cụ thể. Người dùng phải nhập lại thông tin đúng để tiếp tục.</li> </ul>
	<p><b>Thành công:</b> Tài khoản được tạo, người dùng được chuyển đến màn hình đăng nhập.</p> <p><b>Thất bại:</b> Hiển thị lỗi tương ứng tùy từng luồng phụ, yêu cầu người dùng chỉnh sửa và thử lại.</p>

*Bảng 5: Bảng đặc tả ca sử dụng đăng nhập*

Tên ca sử dụng	Thay đổi mật khẩu
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>• Luồng sự kiện chính</li> <li>- Người dùng truy cập chức năng thay đổi mật khẩu trong ứng dụng. Người dùng nhập: Mật khẩu hiện tại (mật khẩu cũ). Mật khẩu mới. Xác nhận lại mật khẩu mới.</li> <li>- Hệ thống kiểm tra.</li> </ul>

	<ul style="list-style-type: none"> <li>- Nếu hợp lệ, hệ thống cập nhật mật khẩu mới cho tài khoản người dùng.           <ul style="list-style-type: none"> <li>• Luồng báo lỗi</li> </ul> </li> <li>- Hệ thống phát hiện mật khẩu cũ không trùng khớp. Hiển thị thông báo: "Mật khẩu hiện tại không đúng. Vui lòng thử lại."</li> <li>- Hệ thống phát hiện mật khẩu mới không khớp với mật khẩu xác nhận. Hiển thị thông báo: "Mật khẩu mới và xác nhận không giống nhau."</li> <li>- Nếu có kiểm tra độ mạnh của mật khẩu, hệ thống cảnh báo: "Mật khẩu mới quá yếu. Vui lòng chọn mật khẩu an toàn hơn."</li> <li>- Nếu người dùng để trống bất kỳ trường nào, hệ thống sẽ hiển thị thông báo: "Vui lòng nhập đầy đủ thông tin."</li> <li>- Nếu quá trình thay đổi gặp lỗi máy chủ hoặc mất kết nối: "Có lỗi xảy ra. Vui lòng thử lại sau."</li> </ul>
Kết quả trả về	<ul style="list-style-type: none"> <li>- Thành công: Mật khẩu được cập nhật trong hệ thống. Hiển thị thông báo: "Thay đổi mật khẩu thành công."</li> <li>- Thất bại: Hiển thị thông báo lỗi cụ thể tùy trường hợp. Người dùng được yêu cầu sửa lại thông tin và thực hiện lại.</li> </ul>

**Bảng 6: Bảng đặc tả ca sử dụng thay đổi mật khẩu**

<b>Tên ca sử dụng</b>	<b>Cập nhật thông tin người dùng</b>
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>• Luồng sự kiện chính. <ul style="list-style-type: none"> <li>- Người dùng truy cập trang “Thông tin cá nhân”.</li> <li>- Nhập thông tin mới.</li> <li>- Nhấn nút “Cập nhật”.</li> <li>- Hệ thống kiểm tra.</li> </ul> </li> <li>• Luồng báo lỗi <ul style="list-style-type: none"> <li>- Hệ thống phát hiện username mới đã tồn tại. Hiển thị thông báo: "Tên đăng nhập đã được sử dụng. Vui lòng chọn tên khác."</li> <li>- Hệ thống phát hiện email mới đã được dùng cho tài khoản khác. Hiển thị thông báo: "Email đã được sử dụng. Vui lòng chọn email khác."</li> <li>- Hệ thống phát hiện định dạng email không đúng chuẩn. Hiển thị thông báo: "Email không hợp lệ. Vui lòng nhập lại."</li> <li>- Nếu quá trình cập nhật gặp lỗi máy chủ hoặc mất kết nối: Hiển thị thông báo: "Có lỗi xảy ra. Vui lòng thử lại sau."</li> </ul> </li> </ul>
Kết quả trả về	<ul style="list-style-type: none"> <li>- Thành công: Cập nhật username và email thành công. Hiển thị thông báo: "Cập nhật thông tin thành công."</li> </ul>

	<ul style="list-style-type: none"> <li>- Thất bại: Hiển thị thông báo lỗi cụ thể tùy vào tình huống</li> </ul>
--	--

**Bảng 7: Bảng đặc tả ca sử dụng cập nhật thông tin**

Tên ca sử dụng	Thay đổi màu giao diện (dark mode, light mode)
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Người dùng vào cài đặt</li> <li>- Người dùng chọn dark mode ấn open để sử dụng dark mode mặc định sẽ là light mode</li> </ul>
Kết quả trả về	Giao diện sẽ thay đổi theo chế độ sáng hoặc tối theo yêu cầu của người dùng

**Bảng 8: Bảng đặc tả ca sử dụng thay đổi giao diện người dùng**

Tên ca sử dụng	Thay đổi ngôn ngữ hiển thị
Tác nhân	Người dùng
Điều kiện đầu vào	
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Người dùng vào cài đặt</li> <li>- Người dùng chọn ngôn ngữ hiển thị</li> </ul> <p>Có thể lựa chọn ngôn ngữ cho ứng dụng ngay cả khi chưa đăng nhập</p>
Kết quả trả về	Ngôn ngữ hiển thị của giao diện sẽ thay đổi theo yêu cầu của người dùng

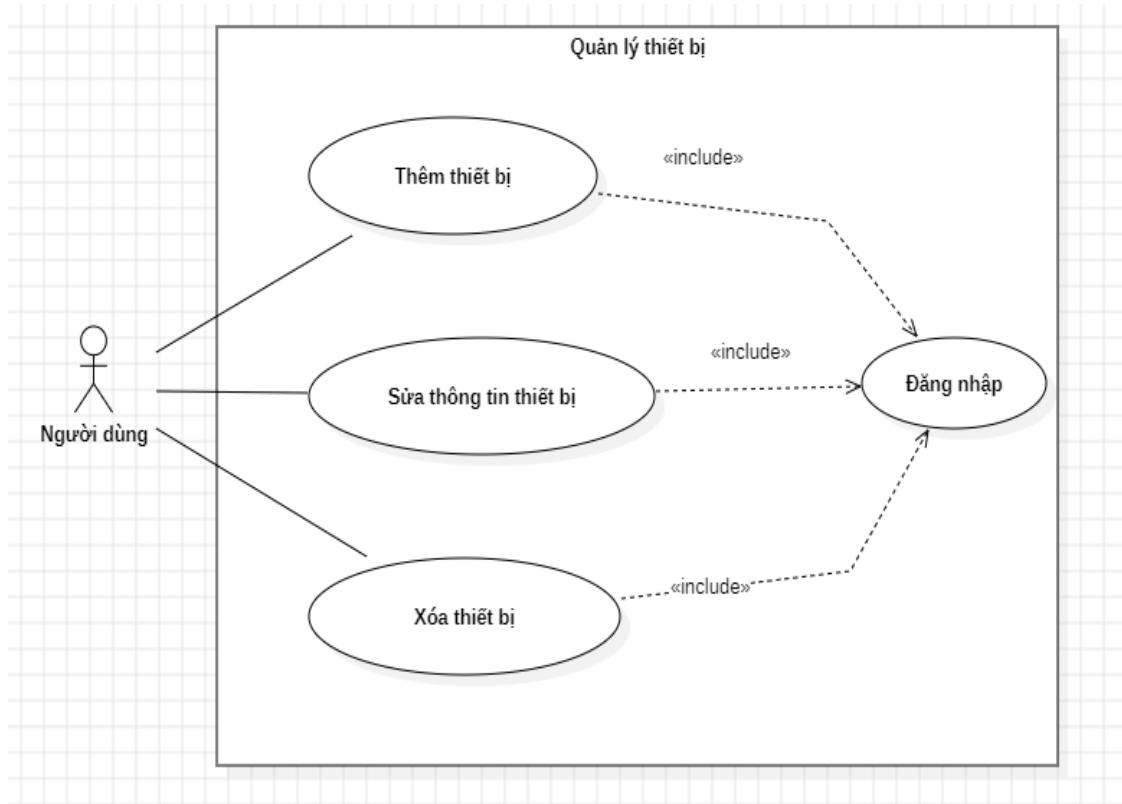
**Bảng 9: Bảng đặc tả ca sử dụng thay đổi ngôn ngữ hiển thị**

Tên ca sử dụng	Quên mật khẩu
Tác nhân	Người dùng
Điều kiện đầu vào	Email đã được đăng ký và tồn tại trong cơ sở dữ liệu
Các luồng sự kiện	<ul style="list-style-type: none"> <li>• Luồng sự kiện chính</li> <li>- Tại màn hình đăng nhập, người dùng chọn liên kết “Quên mật khẩu” bên dưới nút đăng nhập. Hệ thống điều hướng đến màn hình quên mật khẩu.</li> <li>- Người dùng nhập email đã đăng ký trước đó.</li> <li>- Nhấn nút Gửi mã (Send OTP).</li> <li>- Hệ thống kiểm tra: <ul style="list-style-type: none"> <li>▪ Email có tồn tại trong hệ thống hay không.</li> <li>▪ Nếu hợp lệ, tạo và gửi mã OTP (One-Time Password) qua email.</li> <li>▪ OTP có hiệu lực trong 1 phút. Người dùng nhập OTP vào ô xác minh và nhấn Xác nhận.</li> <li>▪ Nếu OTP đúng và còn hiệu lực:</li> <li>▪ Hệ thống cho phép người dùng nhập mật khẩu mới.</li> <li>▪ Hệ thống cập nhật mật khẩu mới cho người dùng.</li> <li>▪ Hiển thị thông báo "Đổi mật khẩu thành công."</li> <li>▪ Điều hướng trở lại trang đăng nhập.</li> </ul> </li> <li>• Luồng báo lỗi</li> </ul>

	<ul style="list-style-type: none"> <li>- Hệ thống kiểm tra và không tìm thấy email. Hiển thị thông báo: "Email không tồn tại trong hệ thống."</li> <li>- Người dùng nhập sai mã OTP. Hiển thị thông báo: "Mã OTP không chính xác. Vui lòng kiểm tra lại."</li> <li>- Người dùng nhập OTP sau thời gian 1 phút. Hệ thống từ chối xác thực và thông báo: "Mã OTP đã hết hạn. Vui lòng yêu cầu gửi lại mã mới."</li> <li>- Gặp lỗi máy chủ hoặc kết nối mạng khi gửi OTP hoặc thay đổi mật khẩu. Hiển thị thông báo: "Có lỗi xảy ra. Vui lòng thử lại sau."</li> </ul>
Kết quả trả về	<ul style="list-style-type: none"> <li>- Thành công: Người dùng đặt lại mật khẩu thành công. Nhận thông báo: "Đổi mật khẩu thành công." Tự động chuyển hướng đến màn hình đăng nhập.</li> <li>- Thất bại: Hiển thị thông báo lỗi cụ thể theo từng bước thất bại để người dùng thao tác lại.</li> </ul>

*Bảng 10 : Bảng đặc tả ca sử dụng quên mật khẩu*

### 3.3.2 Quản lý thiết bị



**Hình 20: Usecase Quản lý thiết bị**

Tên ca sử dụng	Kết nối thiết bị
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Ở màn hình chính của ứng dụng sẽ hiển thị nút "Thêm thiết bị" ở phần dưới của ứng dụng. Khi người dùng ấn vào nút "Thêm thiết bị" thì sẽ được điều hướng đến trang quét QR, ứng dụng sẽ yêu cầu người dùng cấp quyền cho phép sử dụng camera.</li> <li>- Sau khi quét QR thành công sẽ lấy được deviceid và các input khác là các thông tin mà người dùng muốn nhập vào.</li> </ul>

Kết quả trả về	Thêm thiết bị thành công sẽ có thông báo thành công và điều hướng về màn hình chính
----------------	---

**Bảng 11: Bảng đặc tả ca sử dụng kết nối thiết bị**

Tên ca sử dụng	Sửa thông tin thiết bị
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Ở màn hình chính nằm ở góc bên phải của mỗi thiết bị , “device” thì sẽ có icon đại diện ý nghĩa của việc chỉnh sửa</li> <li>- Người dùng sẽ chọn vào biểu tượng đó để thay đổi các thông tin như vị trí, tình trạng,... của thiết bị</li> </ul>
Kết quả trả về	Sửa thông tin thiết bị thành công sẽ có thông báo thành công và điều hướng về màn hình chính

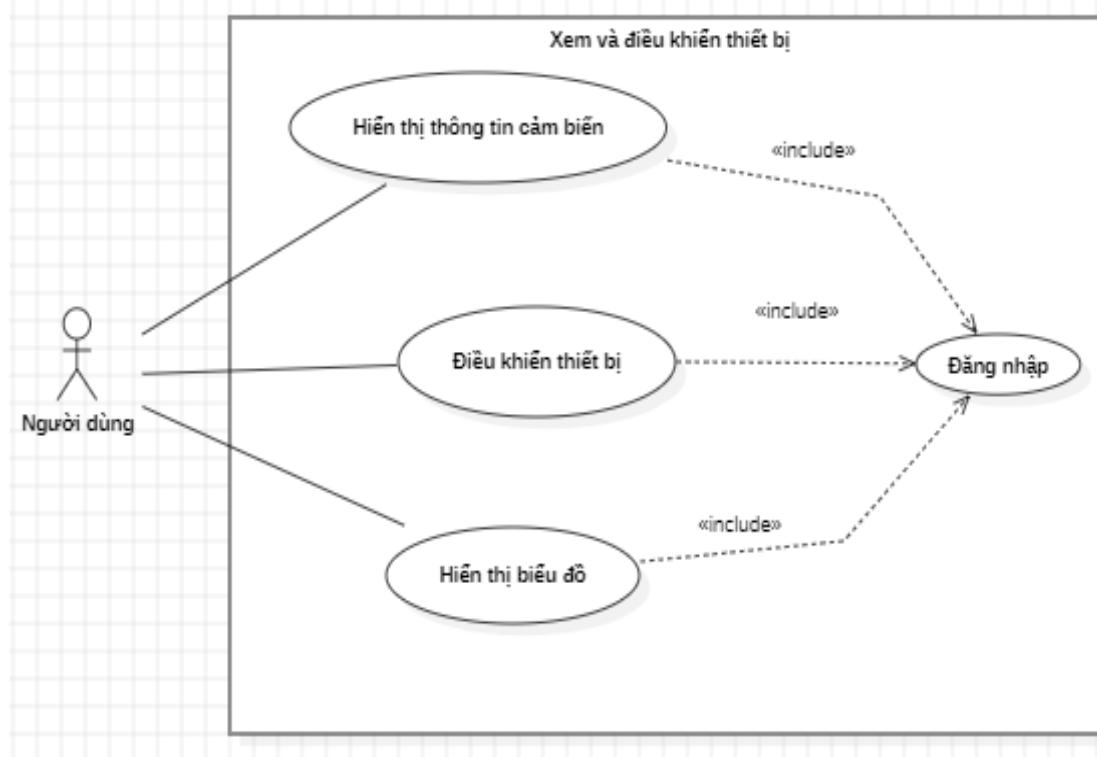
**Bảng 12: Bảng đặc tả ca sử dụng sửa thông tin thiết bị**

Tên ca sử dụng	Xóa thiết bị
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Ở màn hình chính nằm , ở mỗi thiết bị, khi người dùng kéo thiết bị sang trái sẽ có icon xóa, người dùng chọn xóa pop up hiện ra để xác nhận lại yêu cầu</li> <li>- Người dùng xác nhận thì cả thiết bị lẫn biểu đồ sẽ được xóa</li> </ul>

Kết quả trả về	Sửa thông tin thiết bị thành công sẽ có thông báo thành công và điều hướng về màn hình chính
----------------	--

**Bảng 13: Bảng đặc tả ca sử dụng xóa thiết bị**

### 3.3.3 Xem và điều khiển thiết bị



**Hình 21: Usecase Xem và điều khiển thiết bị**

Tên ca sử dụng	Hiển thị thông tin cảm biến
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Người dùng chọn thiết bị ở HomePage.</li> <li>- Server sẽ xác thực người dùng thêm 1 lần nữa và điều hướng tới trang chứa các số liệu mà cảm biến ghi nhận theo thời gian thực.</li> </ul>

Kết quả trả về	Màn hình hiển thị các thông số dữ liệu thay đổi liên tục theo thời gian thực
----------------	--

**Bảng 14: Bảng đặc tả ca sử dụng hiển thị thông tin cảm biến**

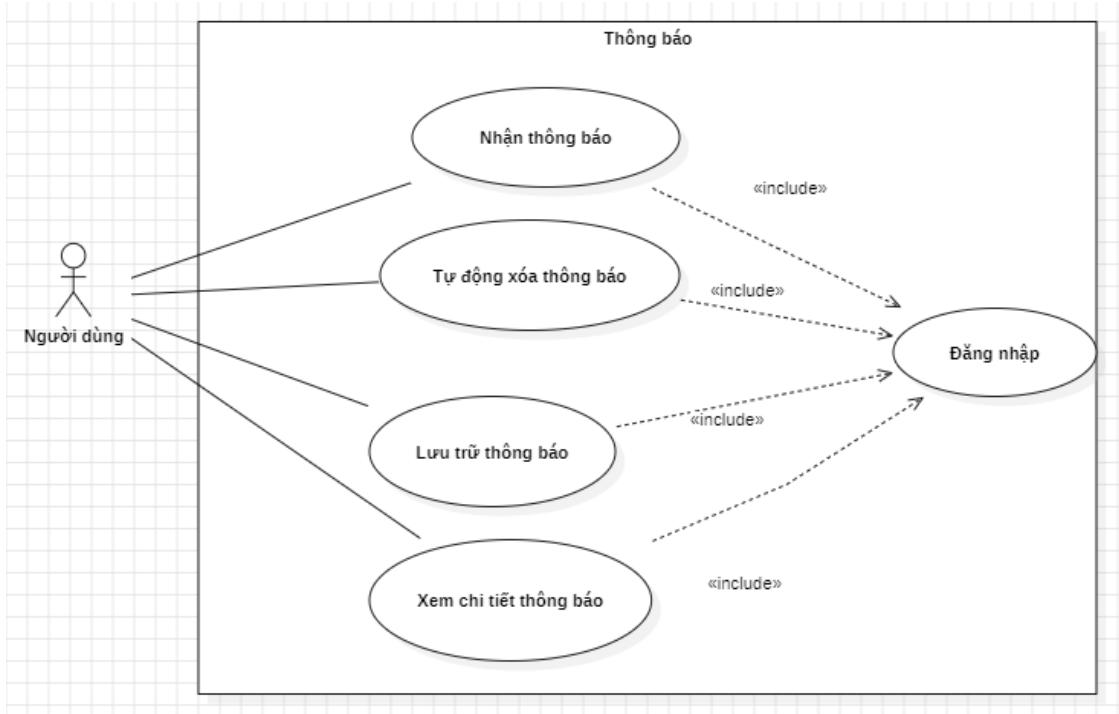
Tên ca sử dụng	Điều khiển thiết bị
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Người dùng chọn thiết bị ở HomePage.</li> <li>- Server sẽ xác thực người dùng thêm 1 lần nữa và điều hướng tới trang chứa các số liệu mà cảm biến ghi nhận theo thời gian thực ở trang đó có 1 relay để bật tắt để điều khiển thiết bị.</li> </ul>
Kết quả trả về	Cảm biến sẽ về trạng thái off và không gửi dữ liệu.

**Bảng 15: Bảng đặc tả ca sử dụng điều khiển thiết bị**

Tên ca sử dụng	Hiển thị biểu đồ
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Ở màn hình chính có drawer, người dùng chọn drawer.</li> <li>- Danh mục hiện ra, chọn thống kê</li> <li>- Tại đây mỗi thiết bị sẽ có 1 bảng thống kê về các chỉ số.</li> </ul>
Kết quả trả về	Biểu đồ hiển thị trực quan về các chỉ số và cảnh báo.

**Bảng 16: Bảng đặc tả ca sử dụng hiển thị biểu đồ thống kê**

### 3.3.4 Thông báo



**Hình 22: Usecase Thông báo**

Tên ca sử dụng	Nhận thông báo
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Sau khi đăng ký và đăng nhập thành công vào thiết bị, ứng dụng sẽ tạo ra fcm_token để có thể sử dụng thông báo được gửi về từ firebase.</li> <li>- Thông báo gửi về sẽ hiển thị trong danh mục thông báo.</li> <li>- Có phân loại thông báo đã đọc và thông báo chưa đọc</li> </ul>
Kết quả trả về	Thông báo sẽ được trả về khi các chỉ số vượt ngưỡng hoạt cảm biến nhận được tín hiệu có thể gây cháy.

**Bảng 17: Bảng đặc tả ca sử dụng nhận thông báo**

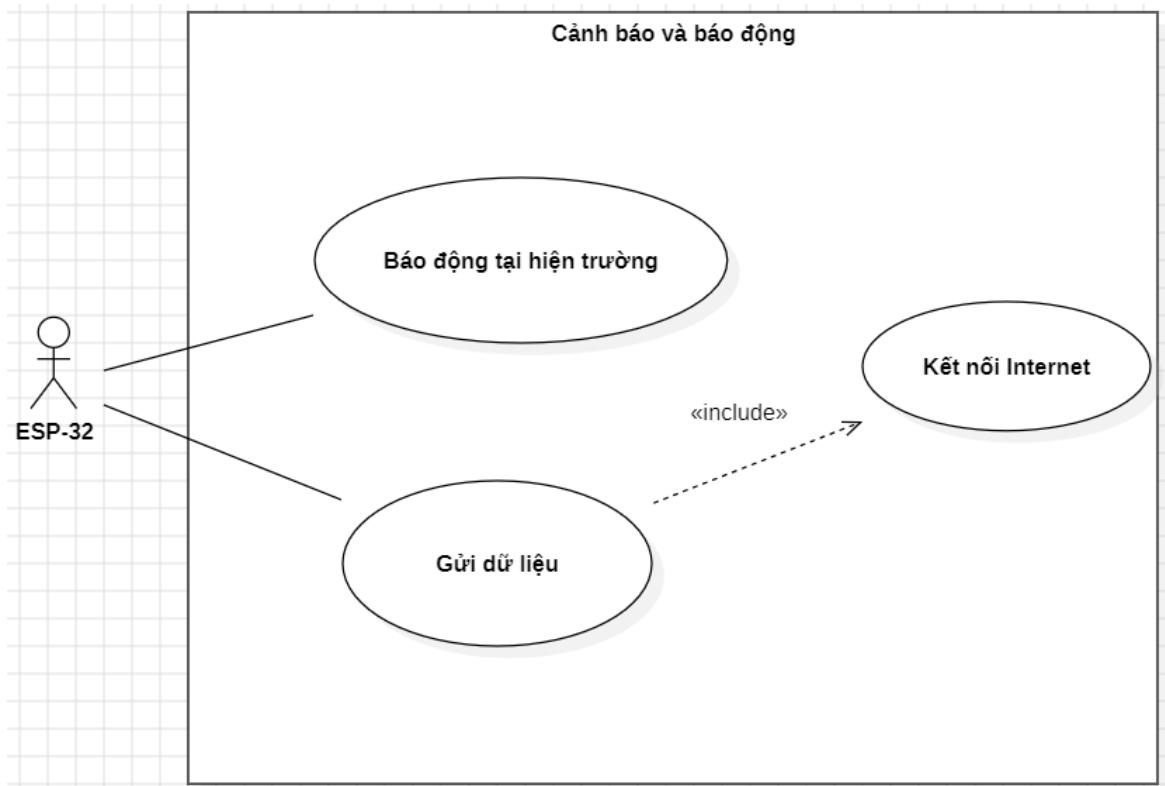
Tên ca sử dụng	Tự động xóa thông báo
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Thông báo sẽ được lưu vào SharedPreferences của flutter và sẽ tự động xóa sau 7 ngày dù người dùng đã đọc hay chưa.</li> </ul>
Kết quả trả về	Biểu đồ biểu thị trực quan về các chỉ số và cảnh báo.

**Bảng 18: Bảng đặc tả ca sử dụng tự động xóa thông báo**

Tên ca sử dụng	Lưu trữ thông báo
Tác nhân	Người dùng
Điều kiện đầu vào	Đăng nhập thành công
Các luồng sự kiện	<ul style="list-style-type: none"> <li>- Thông báo sẽ được lưu vào SharedPreferences của flutter và sẽ tự động xóa sau 30 ngày dù người dùng đã đọc hay chưa.</li> </ul>
Kết quả trả về	

**Bảng 19: Bảng đặc tả ca sử dụng lưu trữ thông báo**

### 3.3.5 Cảnh báo báo động



**Hình 23: Usecase Cảnh báo và báo động**

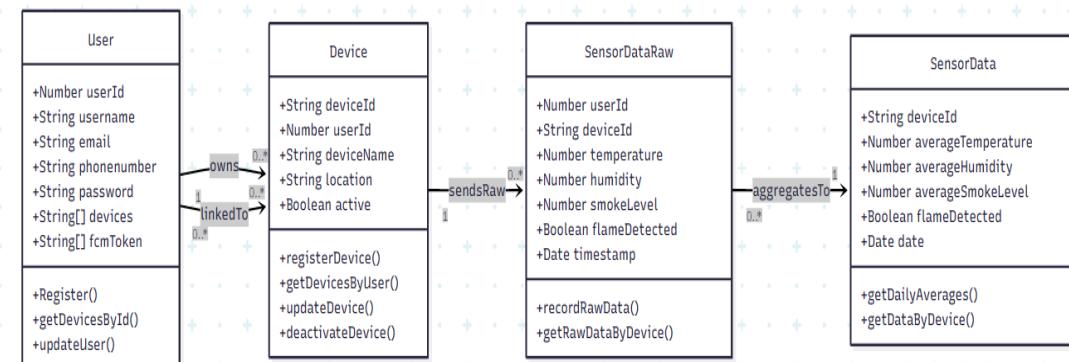
Tên ca sử dụng	Báo động tại hiện trường
Tác nhân	ESP-32
Điều kiện đầu vào	Thiết bị cần được cấp nguồn đầy đủ, lắp đặt đúng kỹ thuật và kết nối thành công.
Các luồng sự kiện	- ESP32 liên tục thu thập và ghi nhận dữ liệu cảm biến, thực hiện các phép so sánh và đánh giá điều kiện. Khi phát hiện các giá trị vượt ngưỡng định sẵn, thiết bị sẽ lập tức kích hoạt hệ thống cảnh báo tại chỗ.
Kết quả trả về	- Còi sẽ là phương tiện để báo động tại chỗ, thiết lập kích hoạt còi sẽ phát ra âm thanh để cảnh báo.

**Bảng 20: Bảng đặc tả ca sử dụng báo động tại hiện trường**

Tên ca sử dụng	Gửi dữ liệu
Tác nhân	ESP-32
Điều kiện đầu vào	Thiết bị cần được cấp nguồn đầy đủ, lắp đặt đúng kỹ thuật và kết nối thành công.
Các luồng sự kiện	- ESP32 sẽ gửi dữ liệu các chỉ số cảm biến lên server
Kết quả trả về	- Server nhận dữ liệu từ ESP-32

**Bảng 21: Bảng đặc tả ca sử dụng gửi dữ liệu**

### 3.4 Sơ đồ lớp



**Hình 24: Sơ đồ lớp**

#### ❖ Mô tả sơ đồ lớp hệ thống IoT báo cháy:

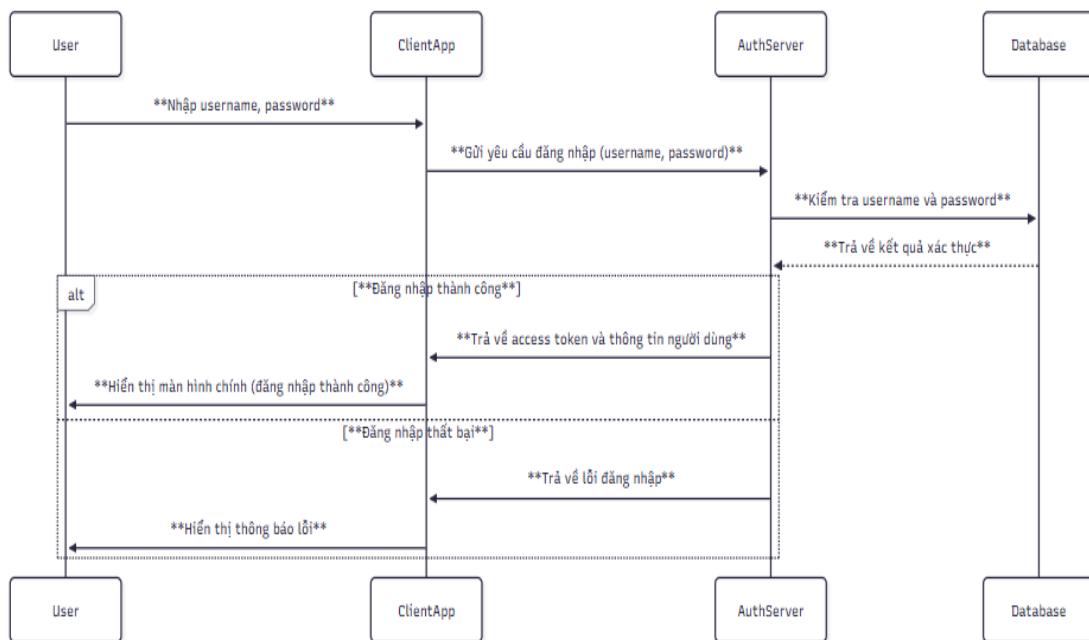
- **User:** Đại diện người dùng hệ thống, quản lý nhiều thiết bị. Lưu thông tin cá nhân và các thiết bị mà họ sở hữu.
- **Device:** Là thiết bị IoT (ví dụ ESP32), mỗi thiết bị thuộc về một người dùng và gửi dữ liệu cảm biến về hệ thống.

- **SensorDataRaw:** Lưu trữ dữ liệu cảm biến thô gửi từ thiết bị, gồm nhiệt độ, độ ẩm, mức khói, trạng thái lửa, cùng thời gian ghi nhận.
- **SensorData:** Là dữ liệu đã được tổng hợp (ví dụ trung bình theo ngày) từ các bản ghi thô, dùng để báo cáo và phân tích dài hạn.

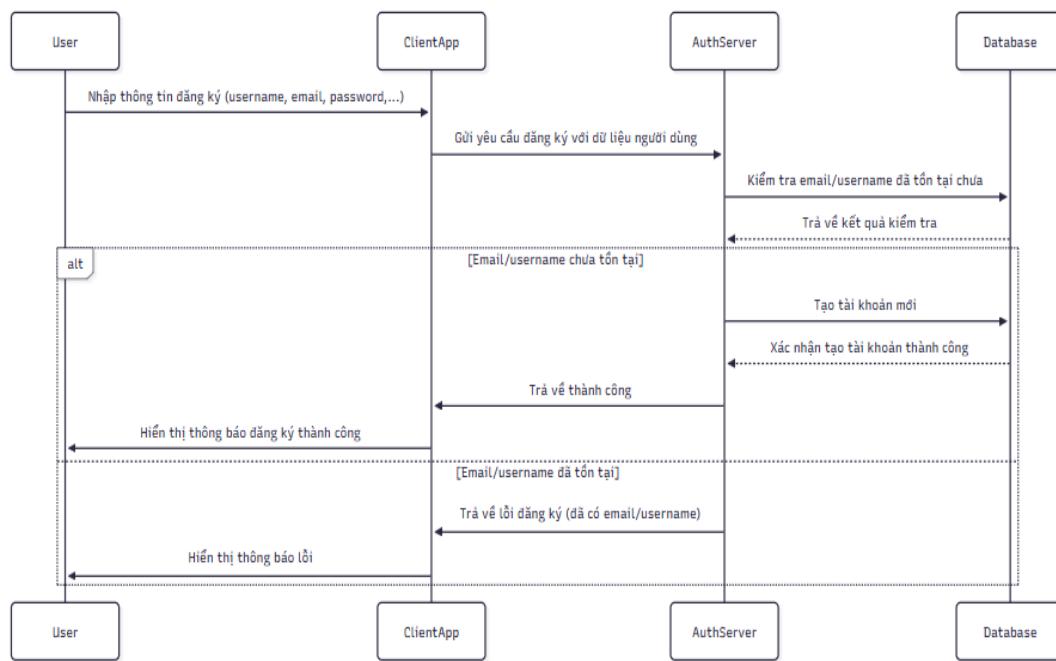
❖ **Mối quan hệ:**

- Mỗi người dùng có nhiều thiết bị.
- Mỗi thiết bị gửi nhiều dữ liệu thô (SensorDataRaw).
- Dữ liệu thô được tổng hợp để tạo dữ liệu trung bình (SensorData)

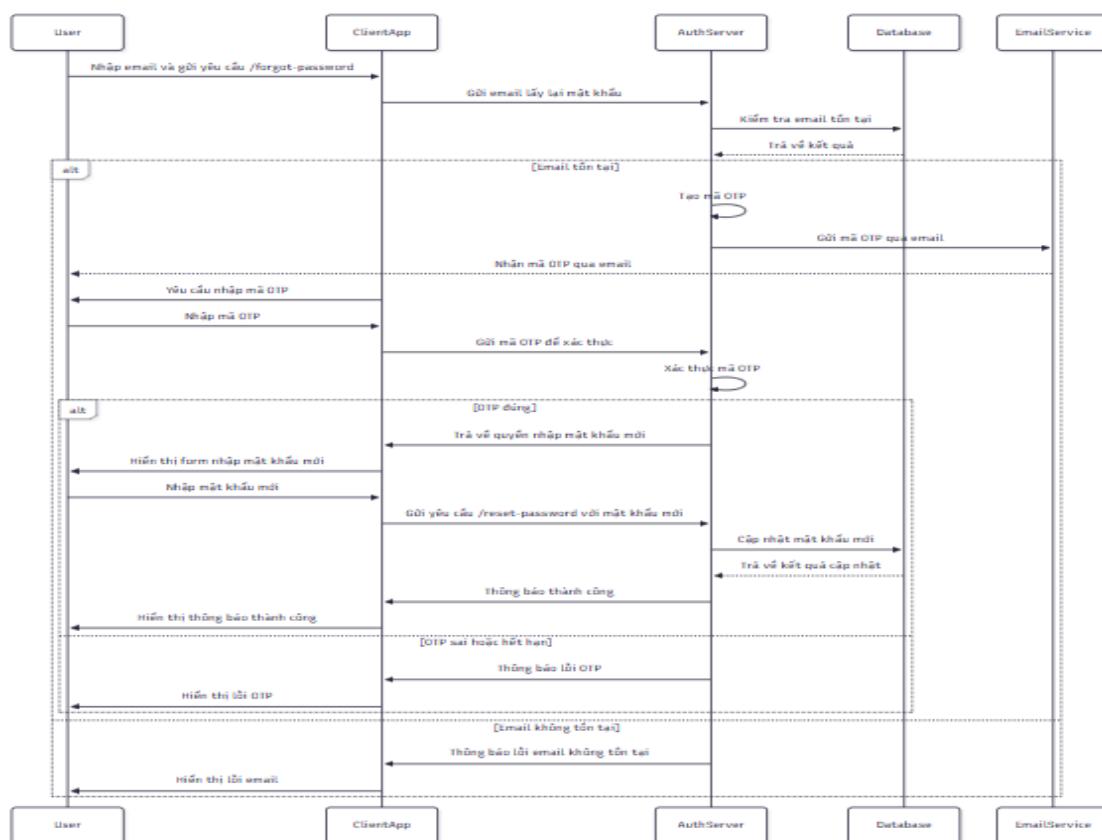
### 3.5 Sơ đồ tuần tự



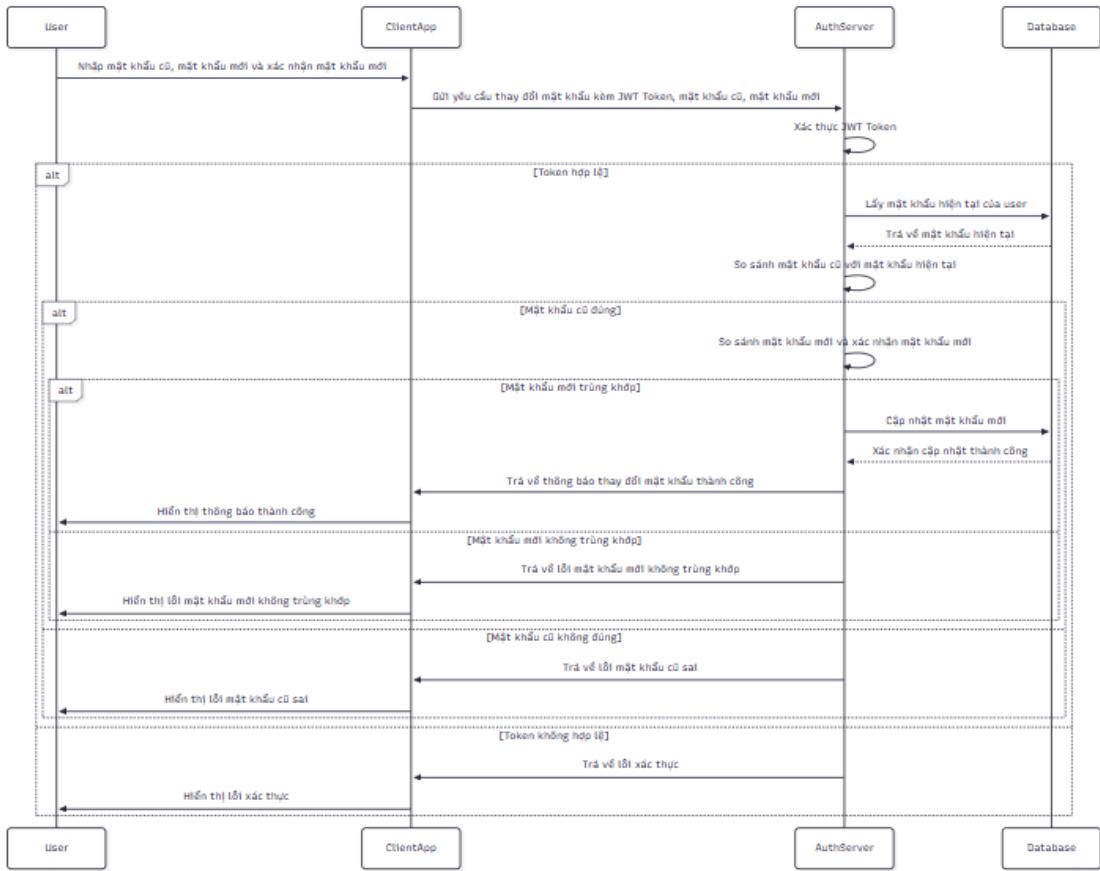
**Hình 25: Sơ đồ tuần tự đăng nhập**



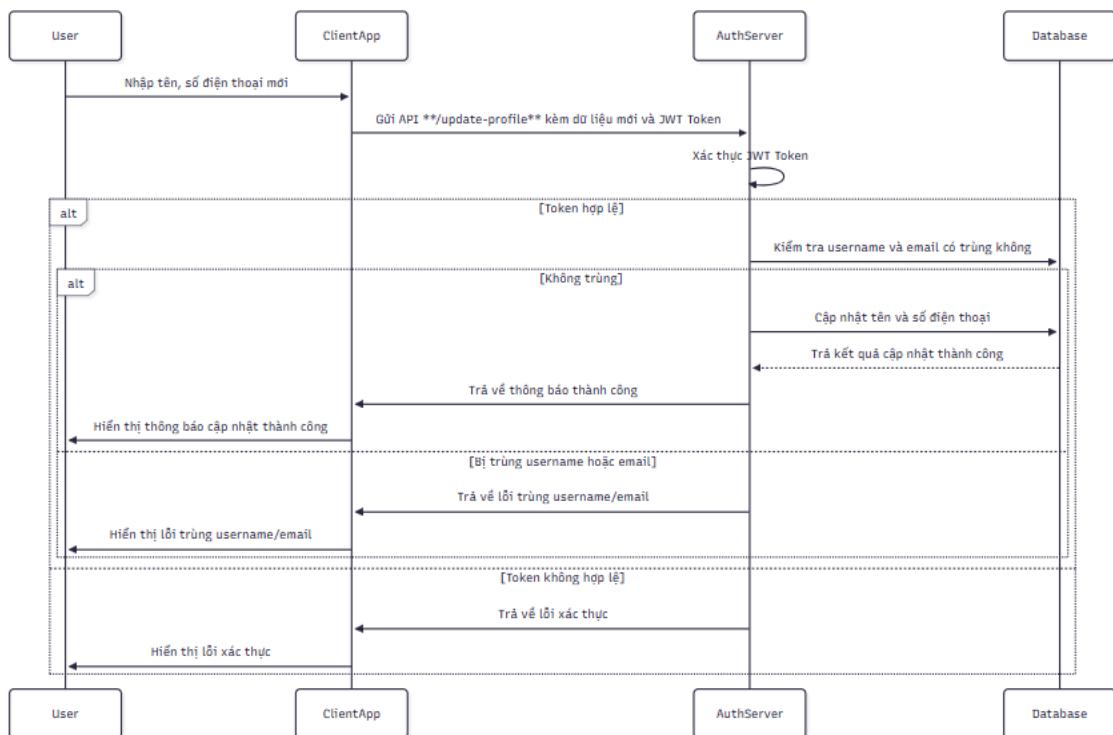
**Hình 26: Sơ đồ tuần tự đăng ký**



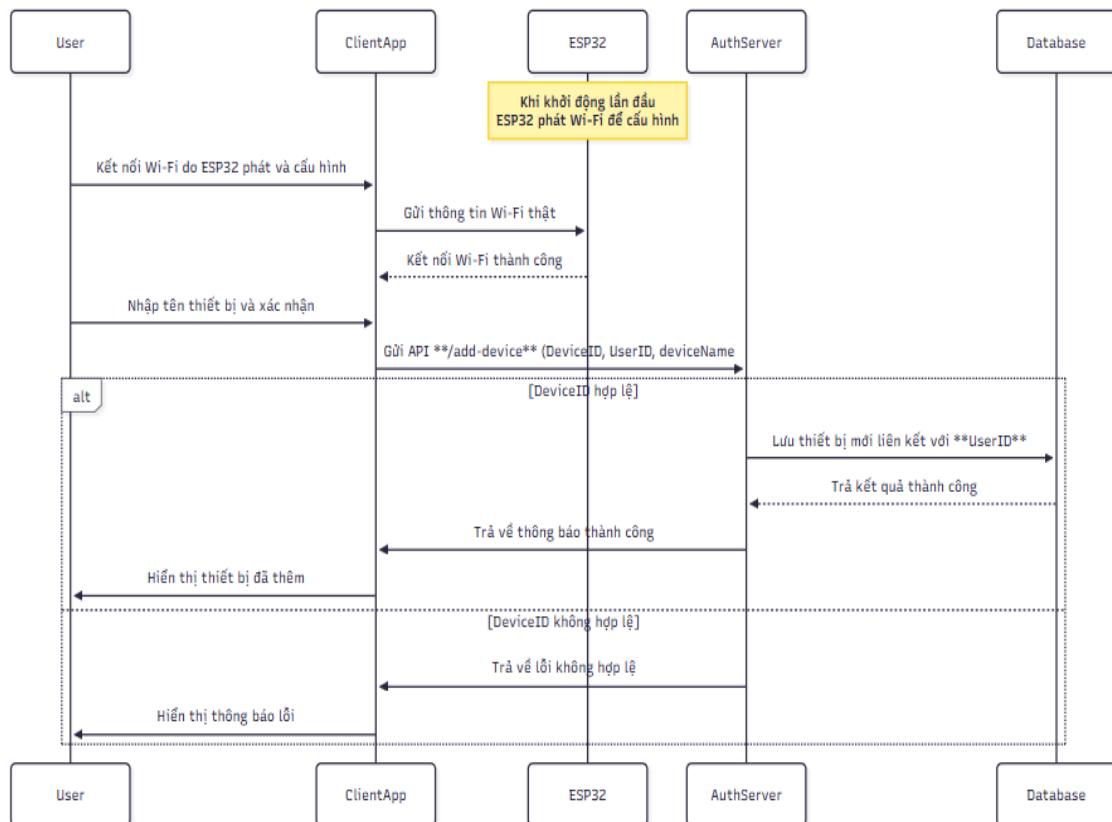
**Hình 27: Sơ đồ tuần tự quên mật khẩu**



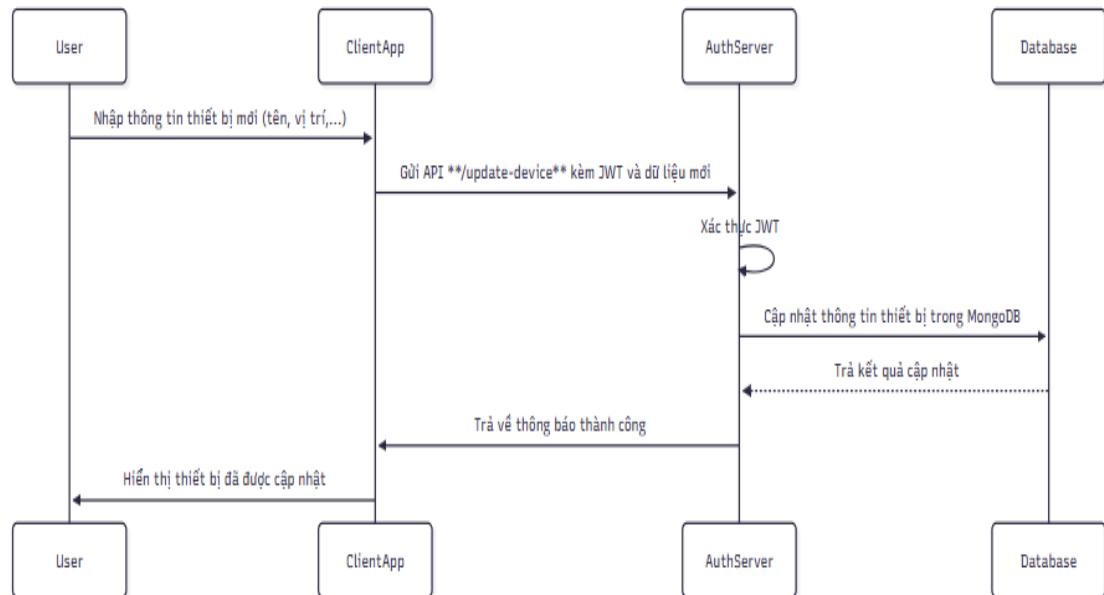
**Hình 28: Sơ đồ tuần tự thay đổi mật khẩu**



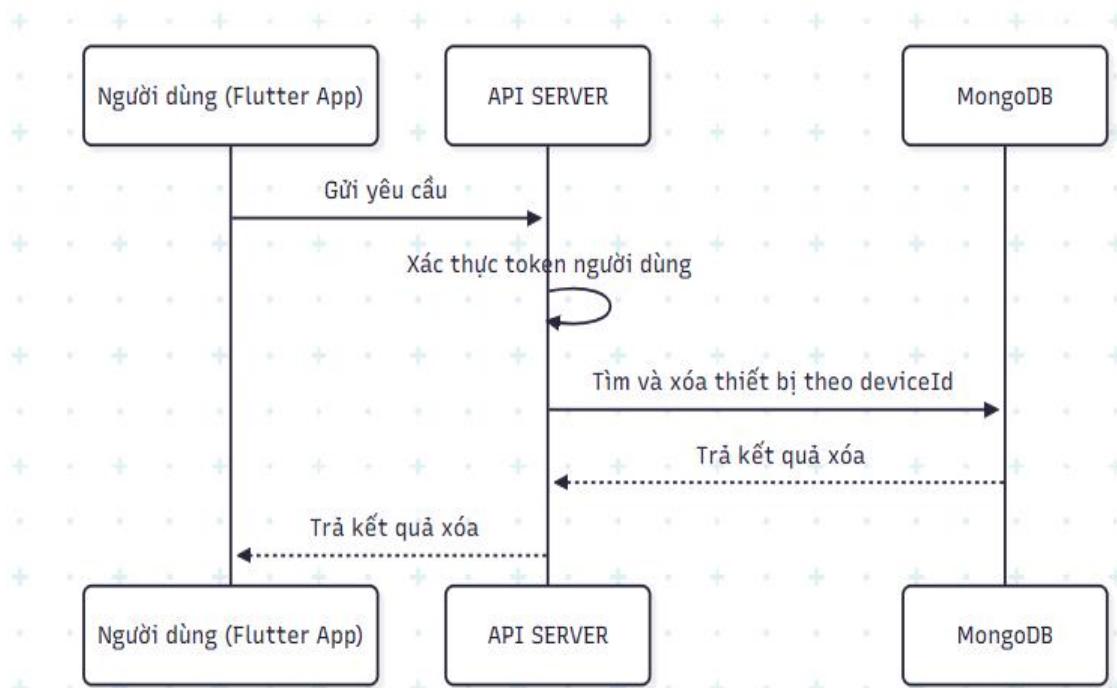
**Hình 29: Sơ đồ tuần tự cập nhật thông tin cá nhân**



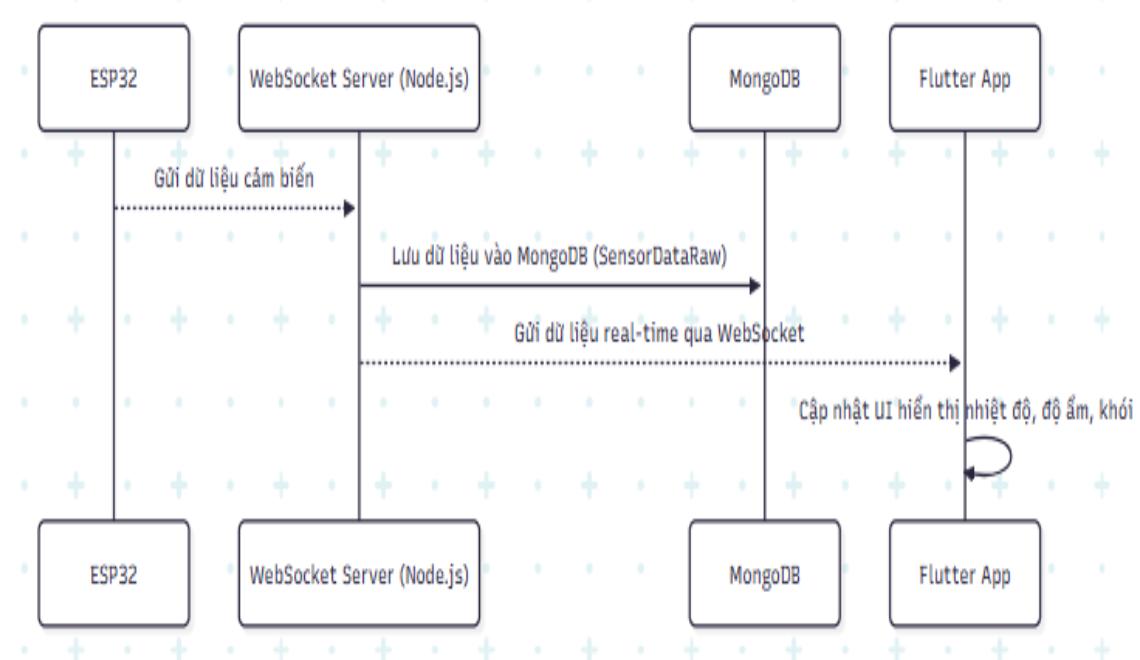
**Hình 30: Sơ đồ tuần tự kết nối thiết bị**



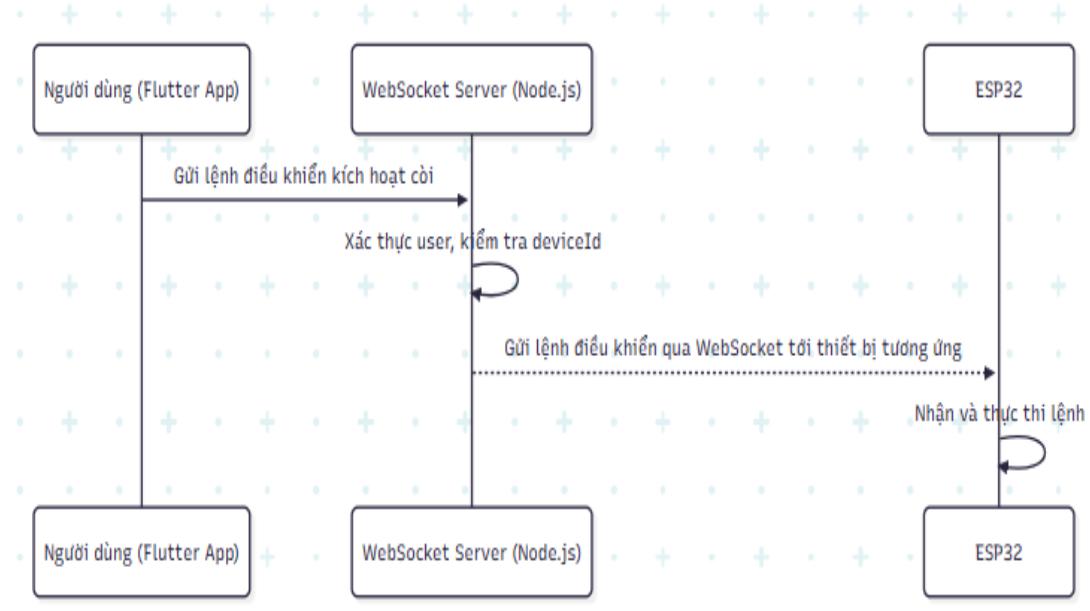
**Hình 31: Sơ đồ tuần tự cập nhật thông tin thiết bị**



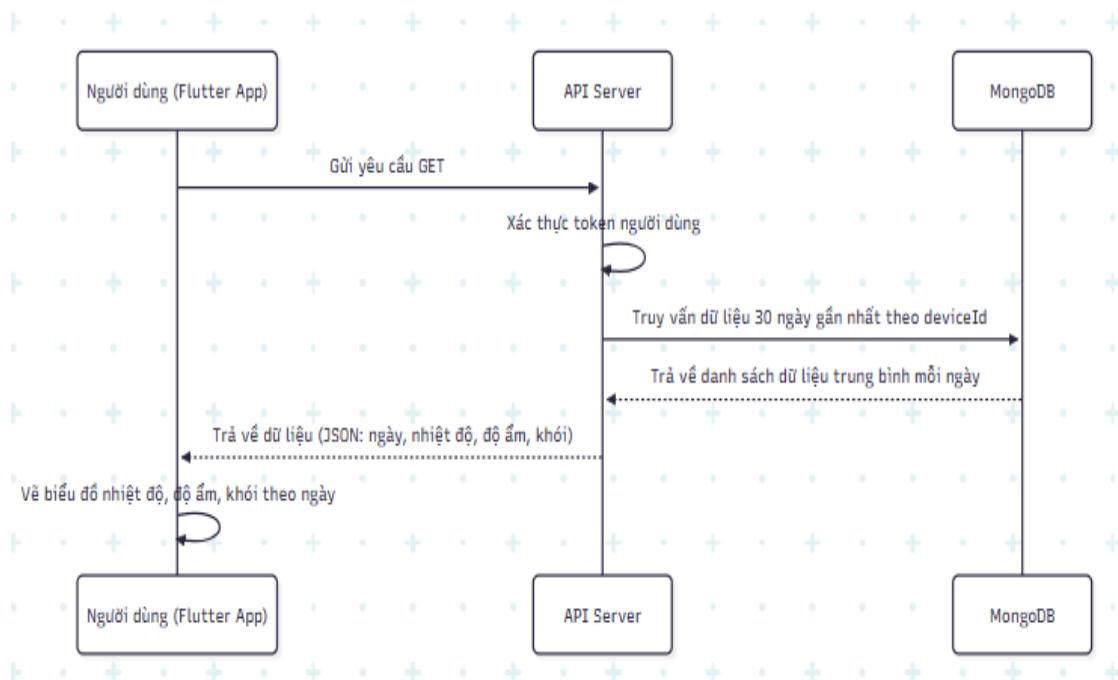
**Hình 32: Sơ đồ tuần tự xóa thiết bị**



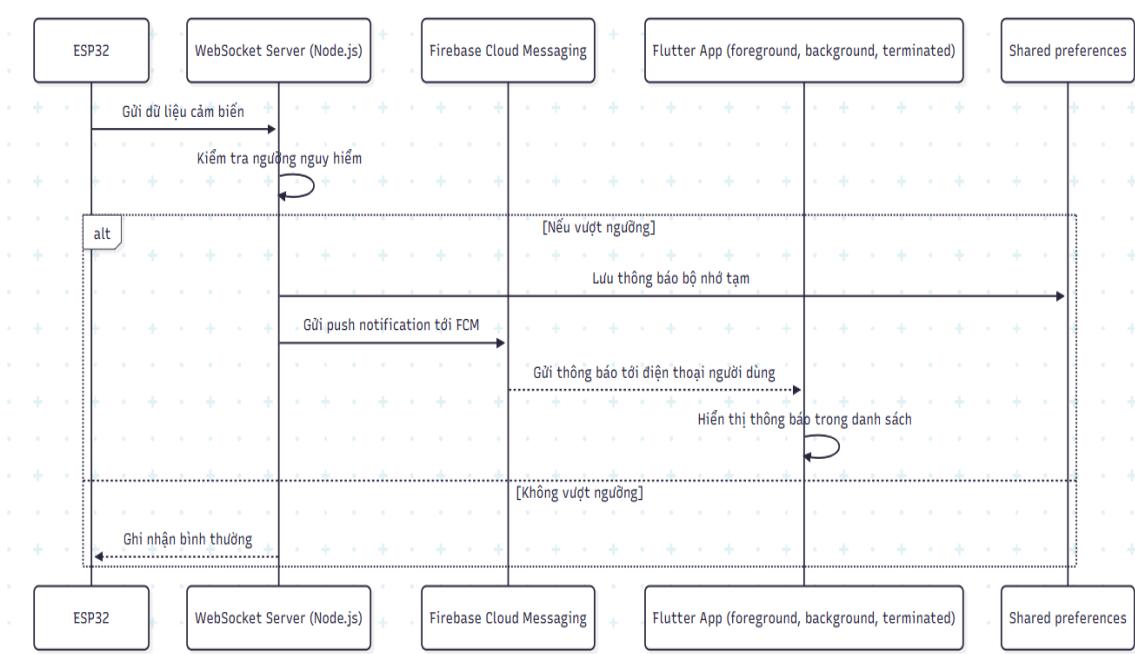
**Hình 33: Sơ đồ tuần tự hiển thị dữ liệu cảm biến**



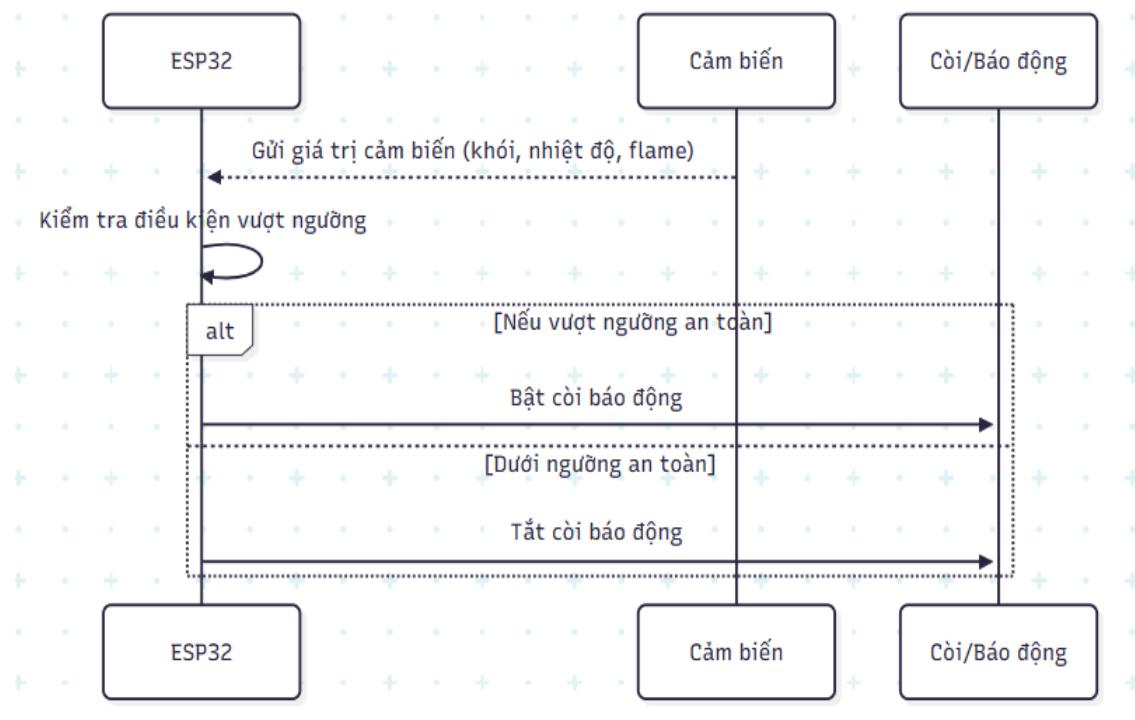
**Hình 34: Sơ đồ tuần tự điều khiển thiết bị**



**Hình 35: Sơ đồ tuần tự hiển thị biểu đồ thống kê**

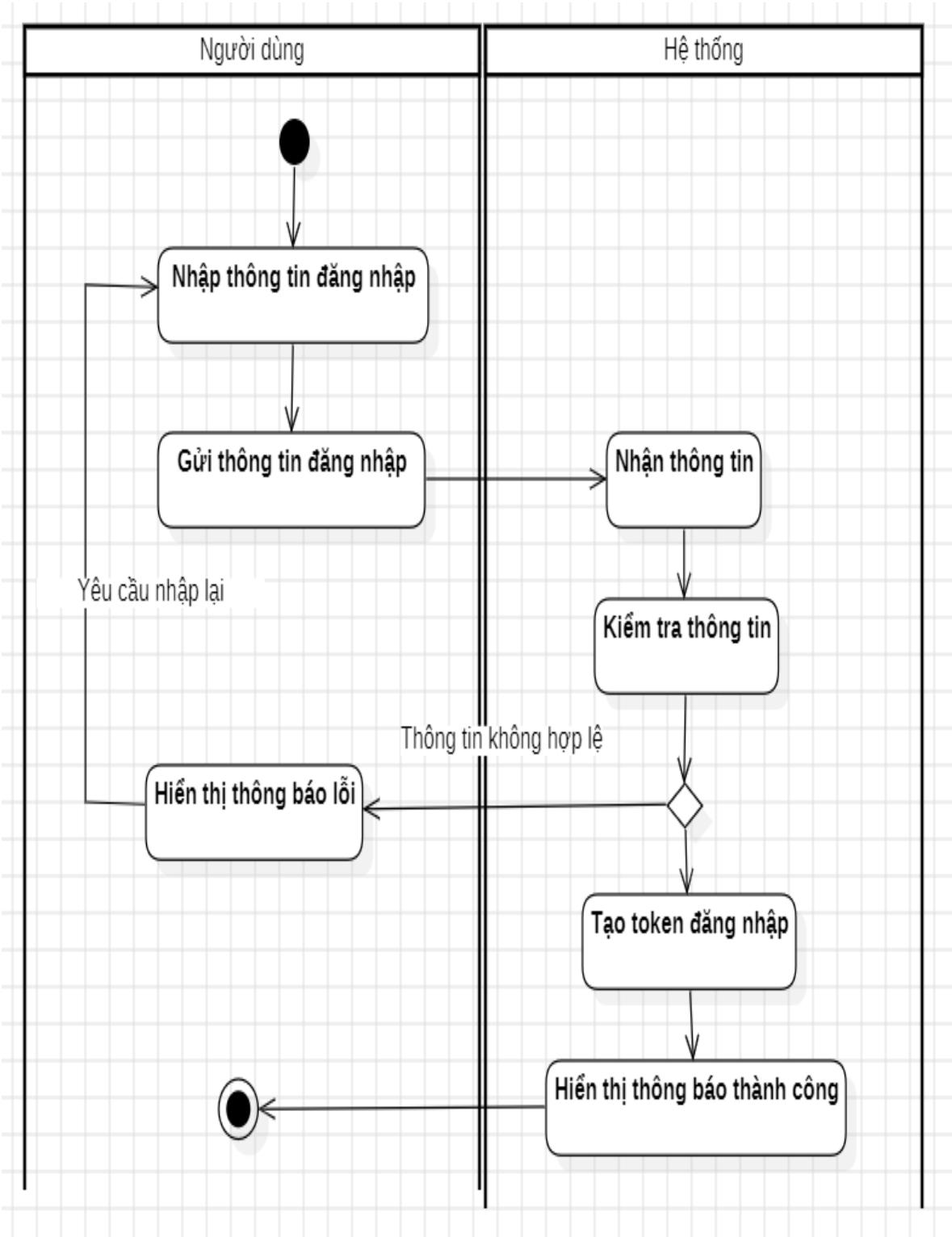


**Hình 36: Sơ đồ tuần tự thông báo**

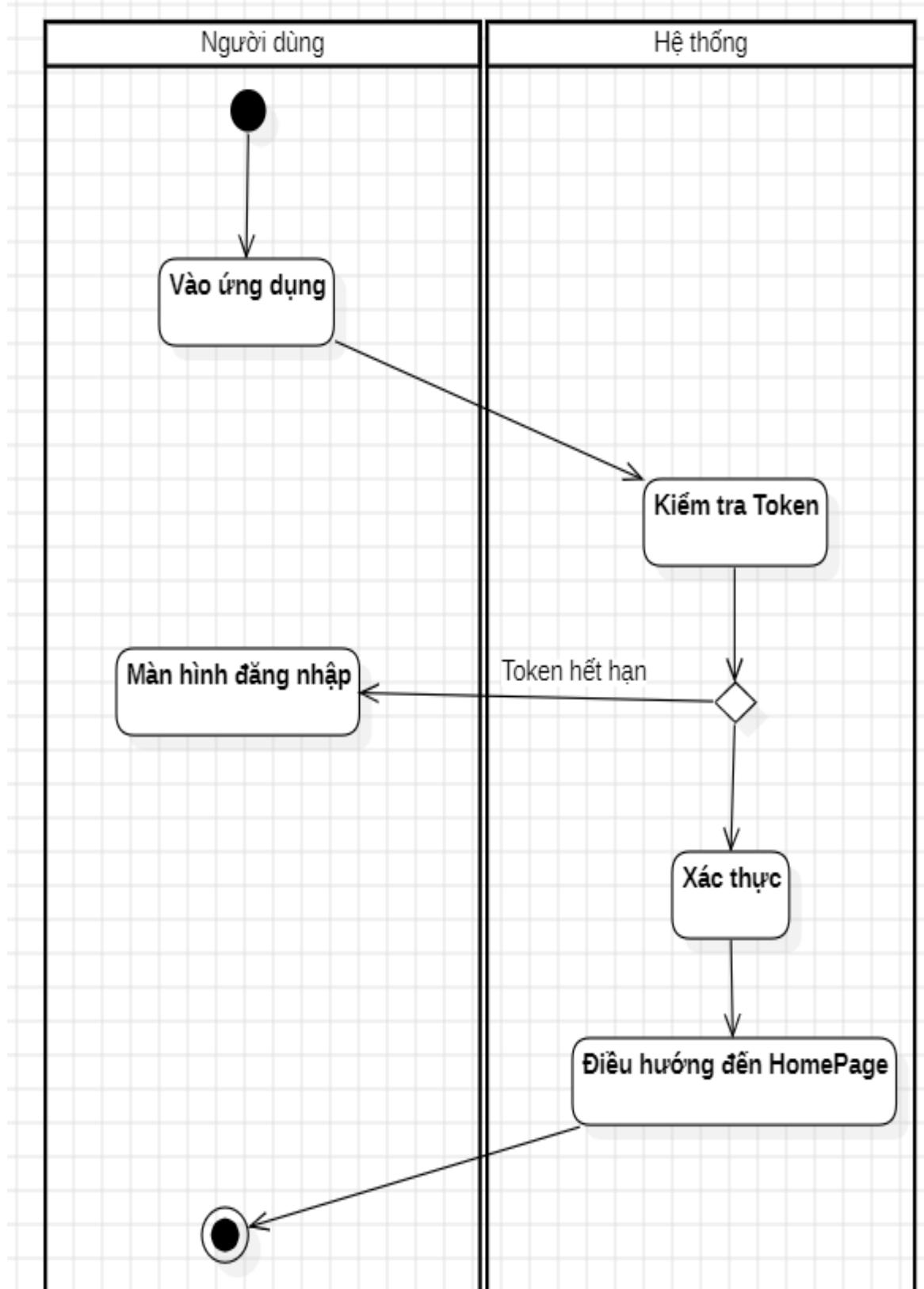


**Hình 37: Sơ đồ tuần tự báo động**

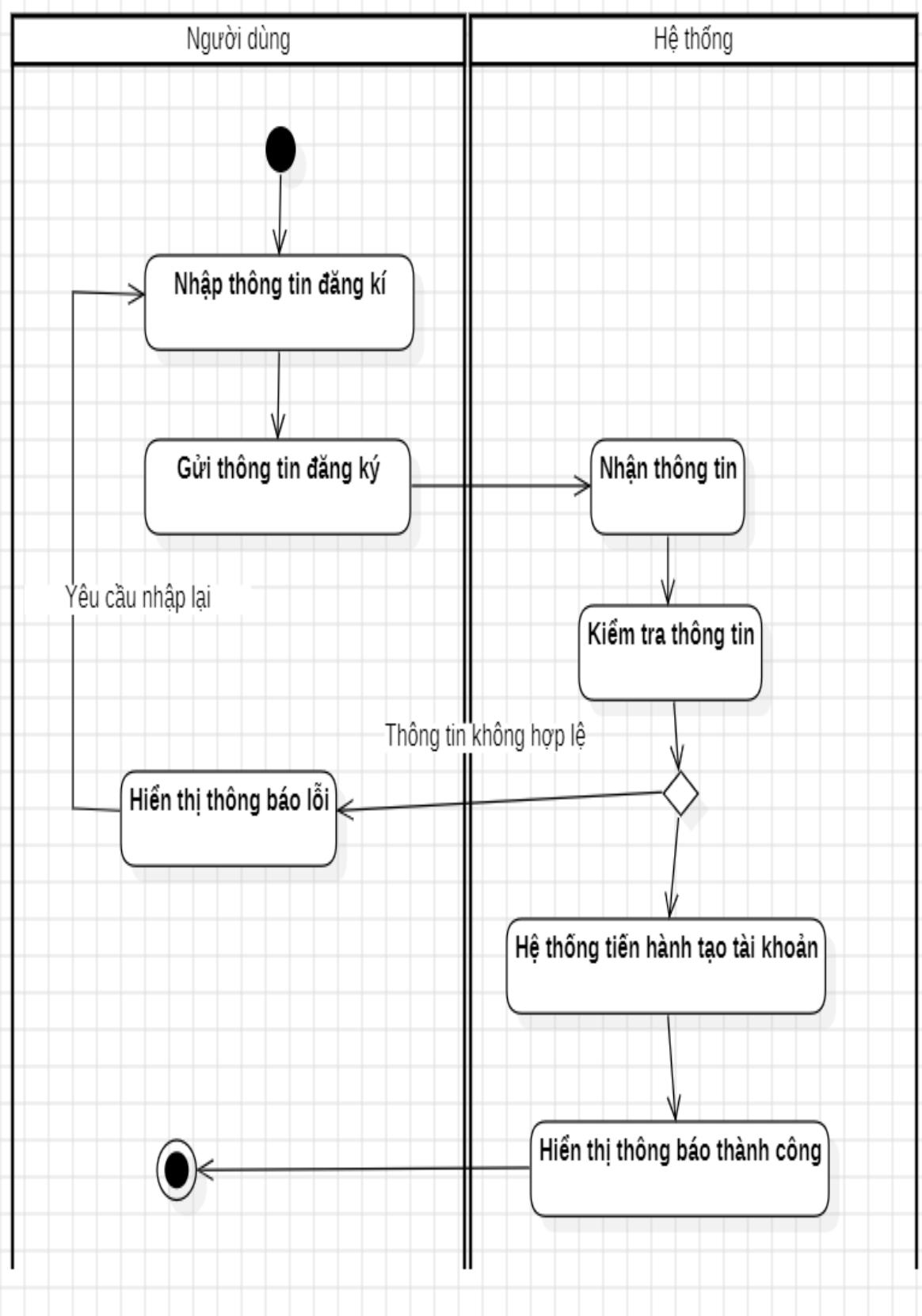
### 3.6 Sơ đồ hoạt động



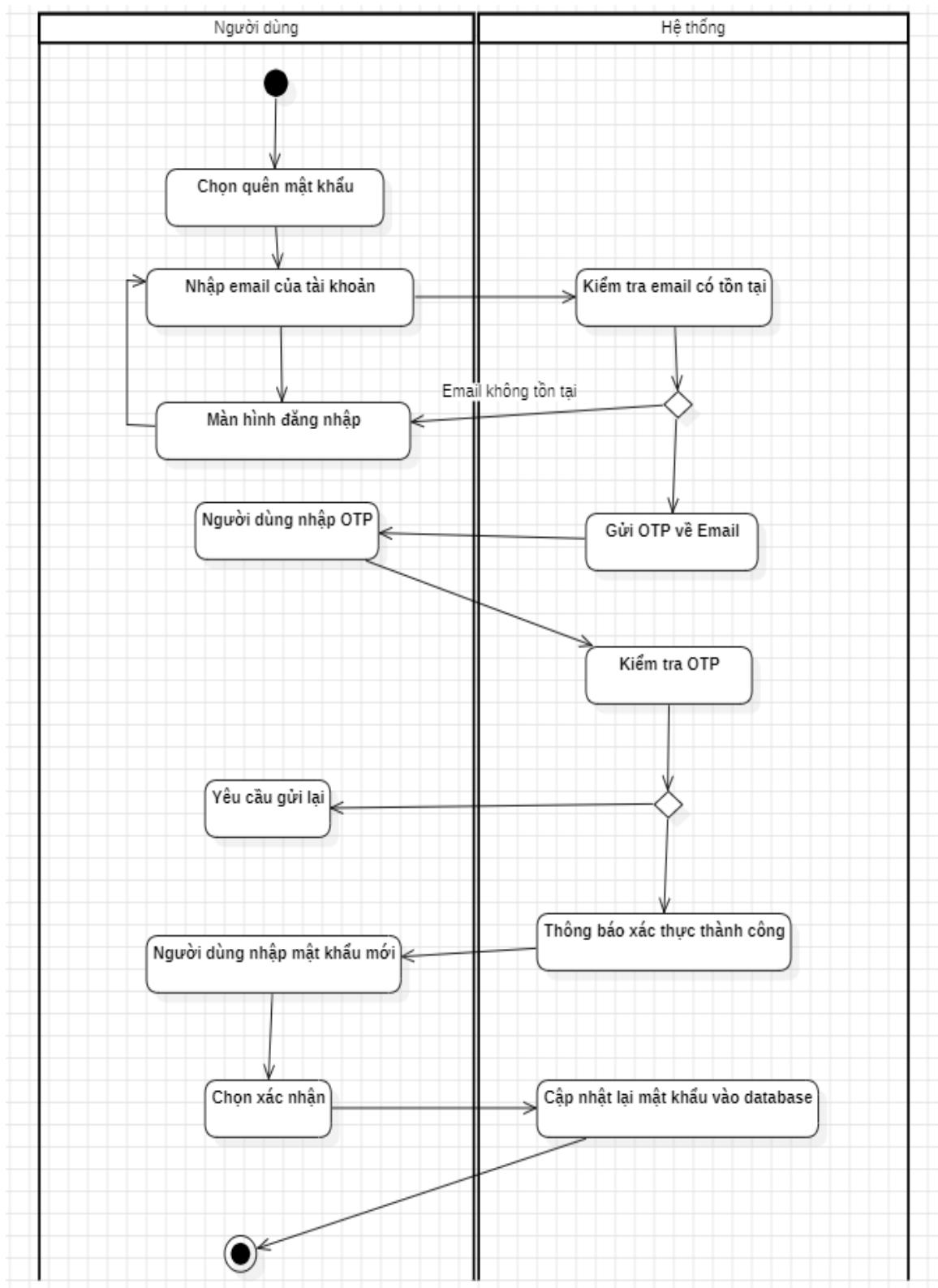
**Hình 38: Sơ đồ hoạt động đăng nhập**



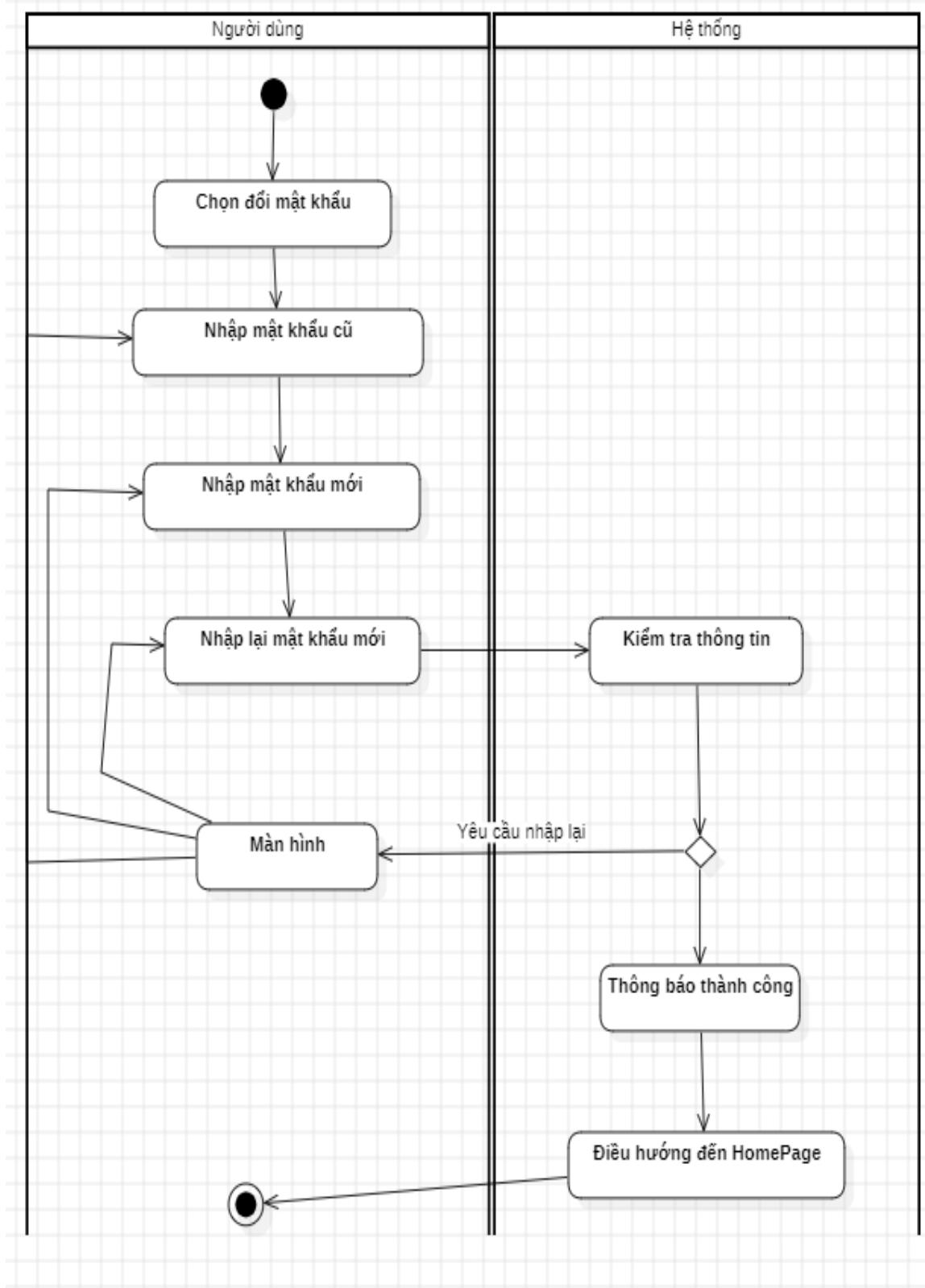
**Hình 39: Sơ đồ hoạt động tự động đăng nhập**



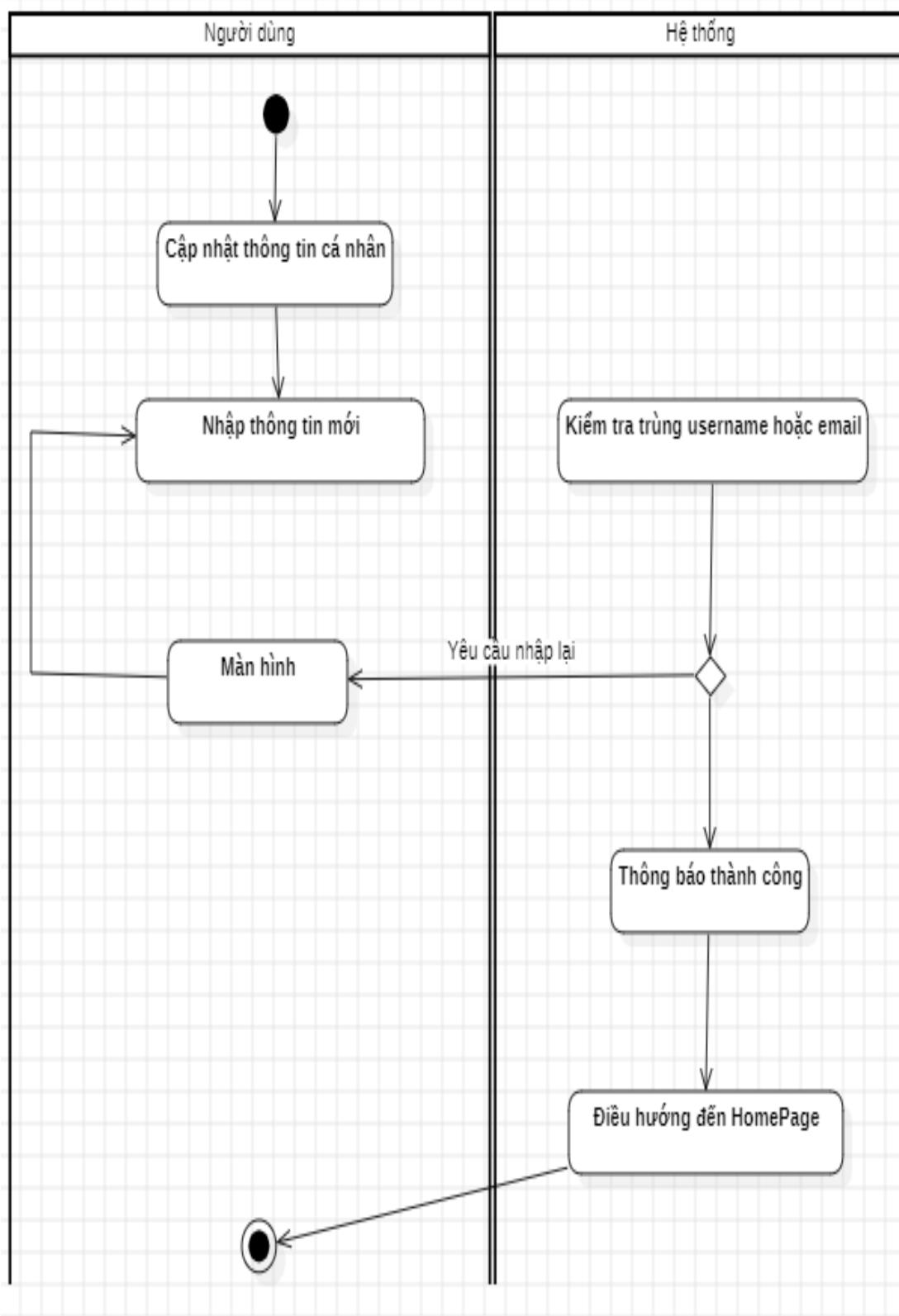
**Hình 40: Sơ đồ hoạt động đăng ký**



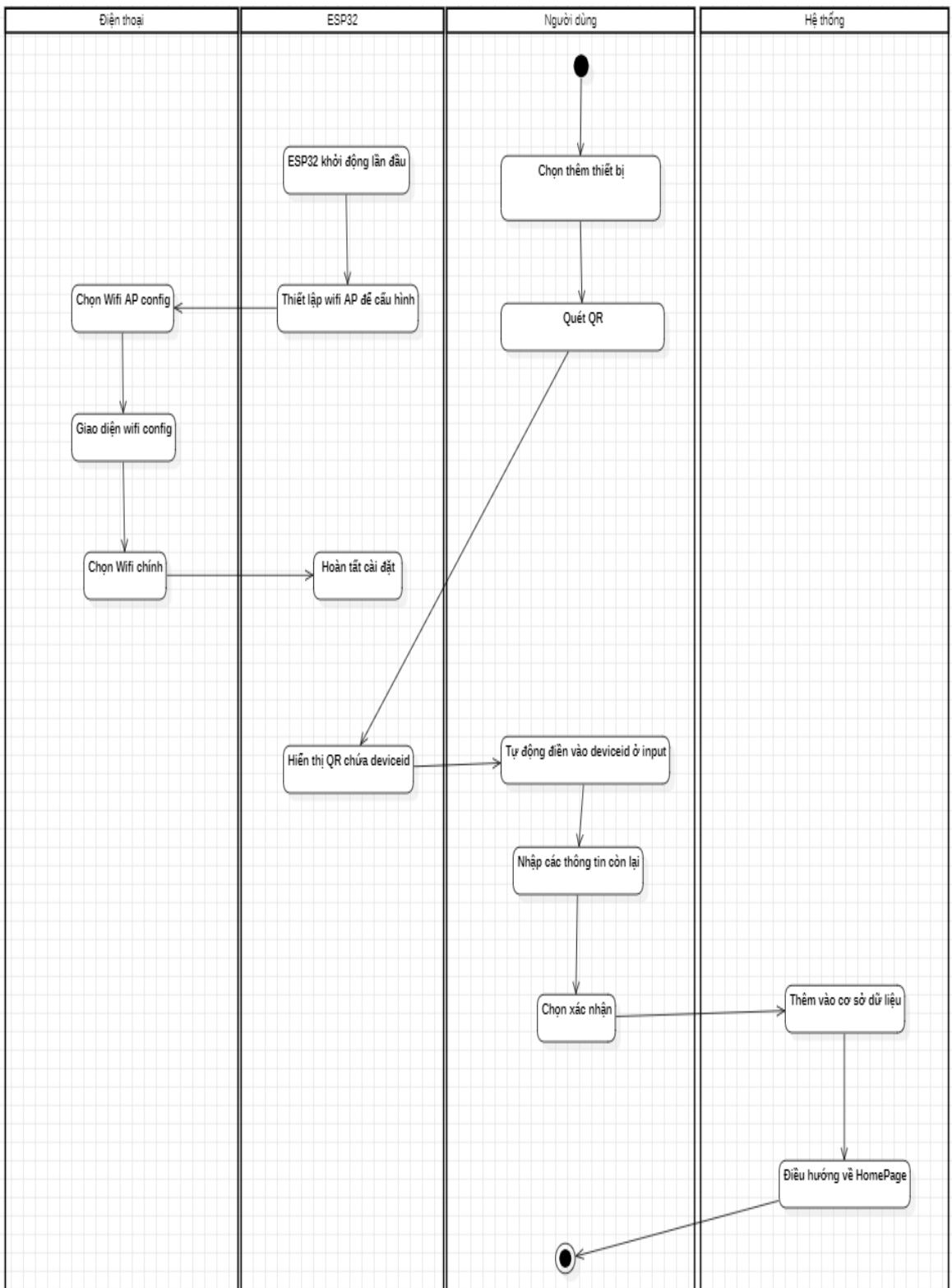
**Hình 41: Sơ đồ hoạt động quên mật khẩu**



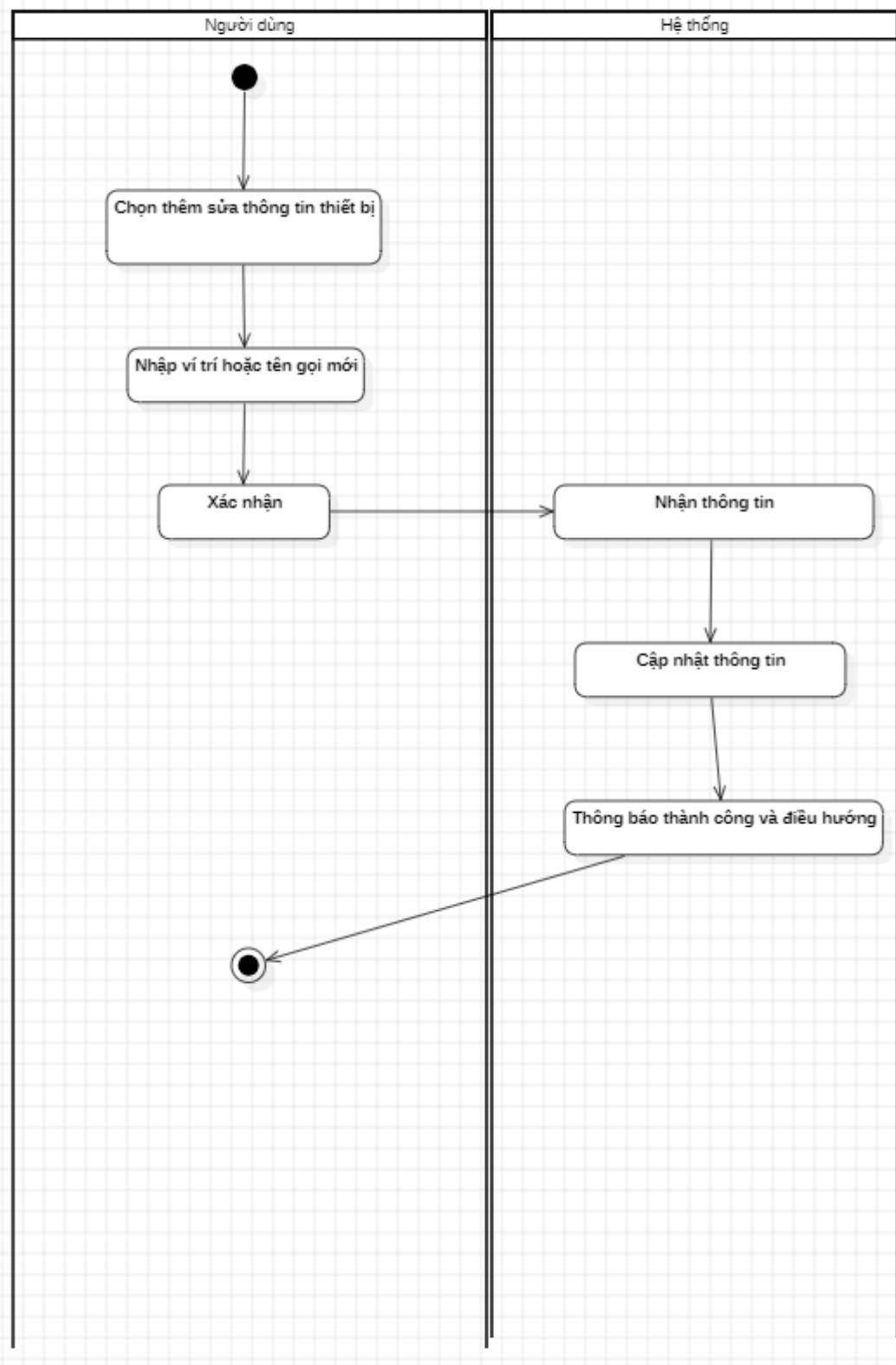
**Hình 42: Sơ đồ hoạt động thay đổi mật khẩu**



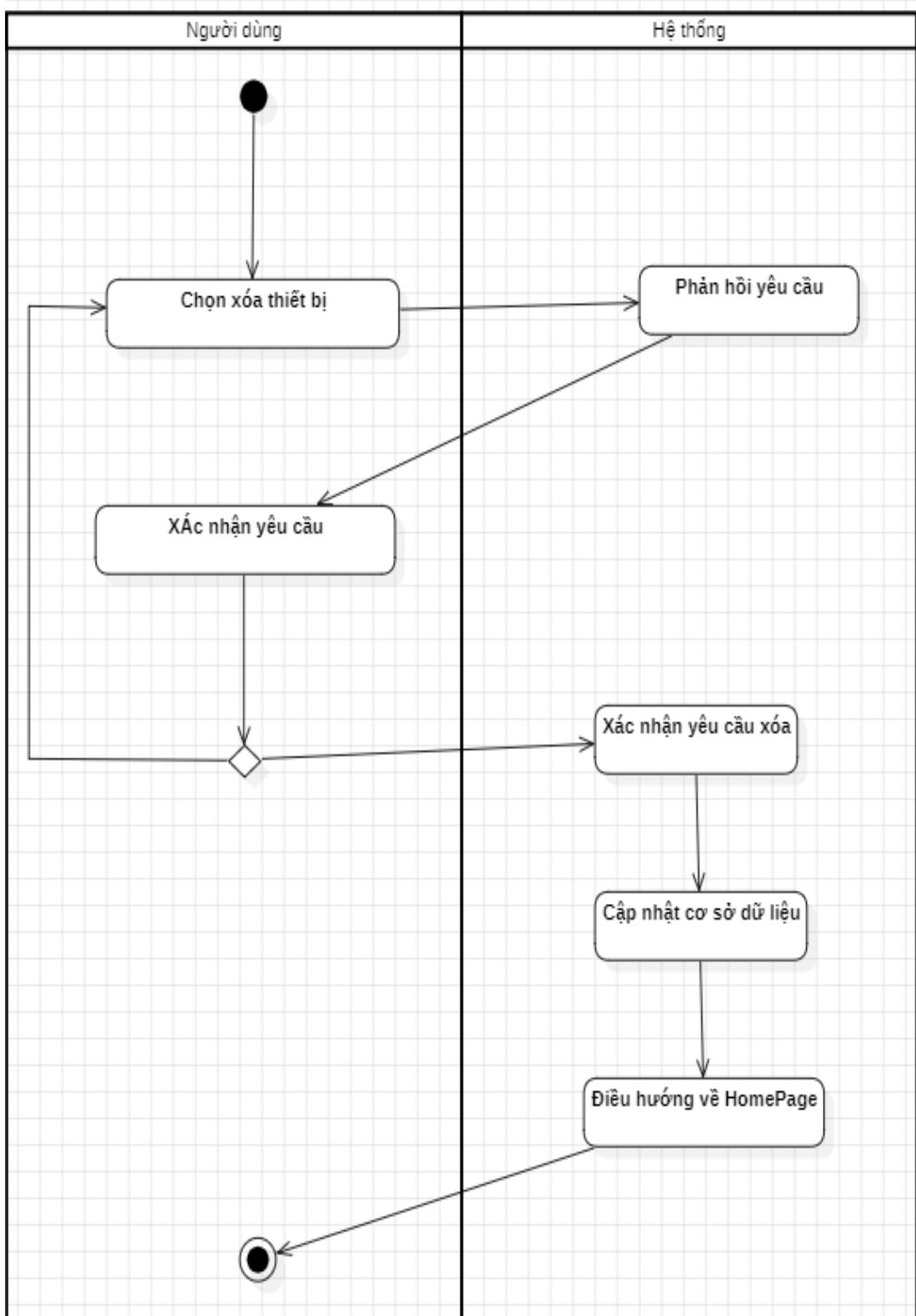
**Hình 43: Sơ đồ hoạt động cập nhật thông tin cá nhân**



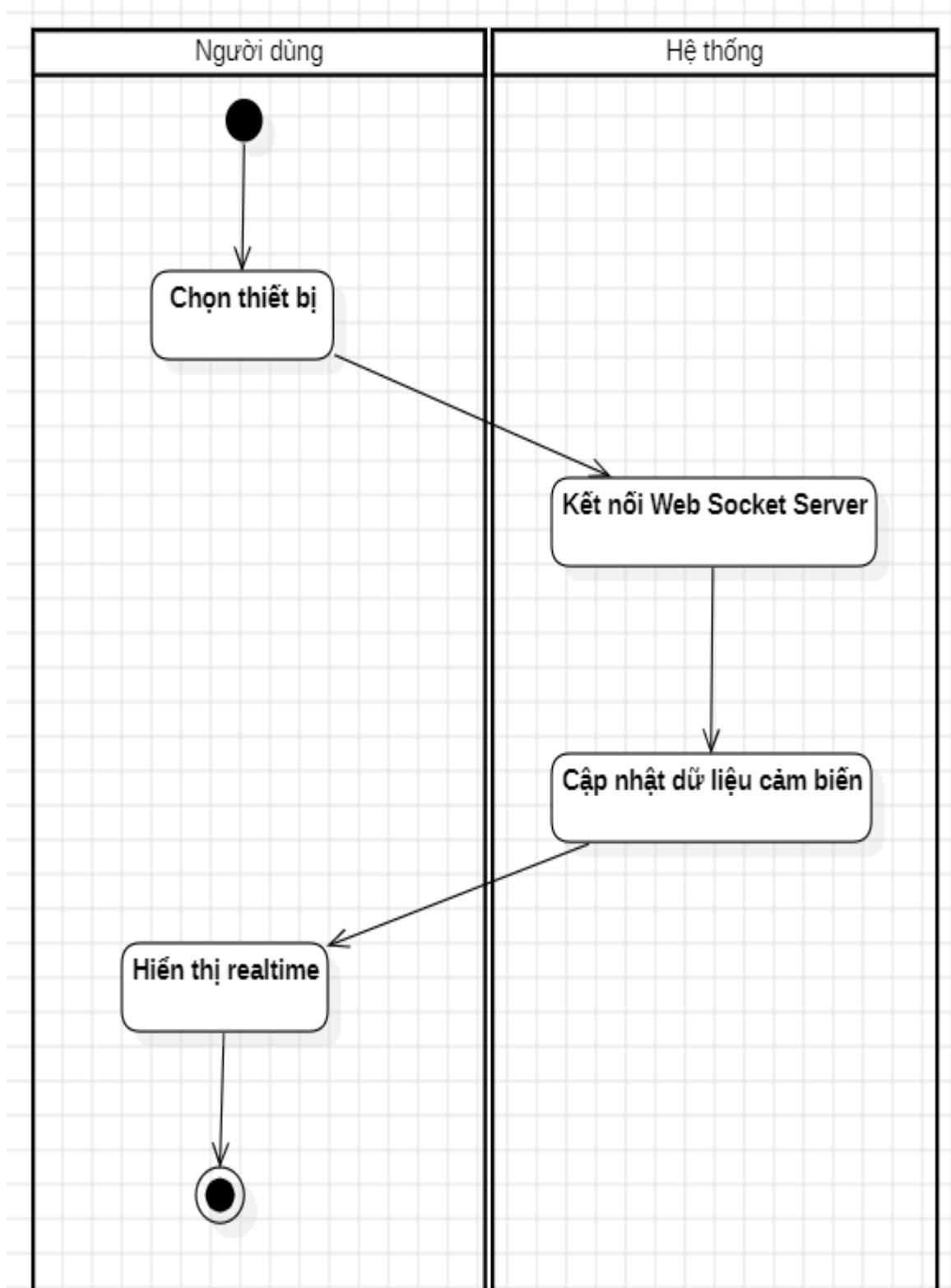
**Hình 44: Sơ đồ hoạt động kết nối thiết bị**



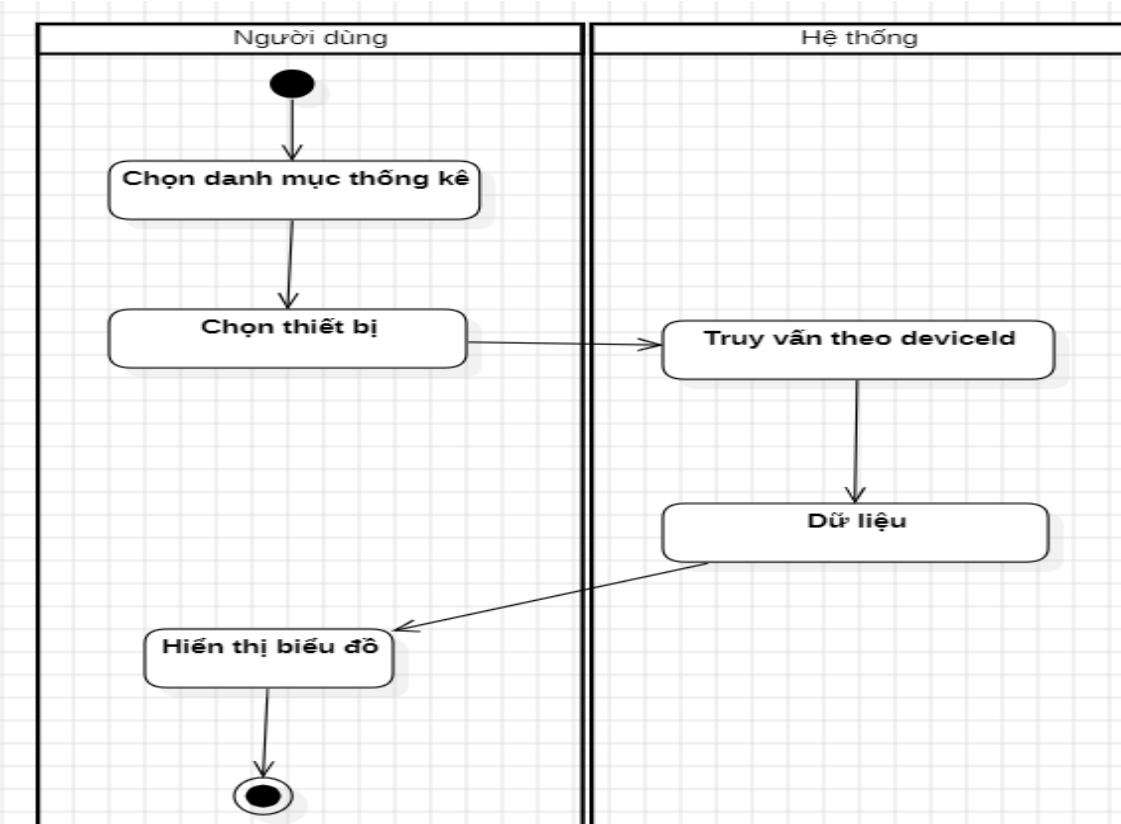
**Hình 45: Sơ đồ hoạt động cập nhật thông tin thiết bị**



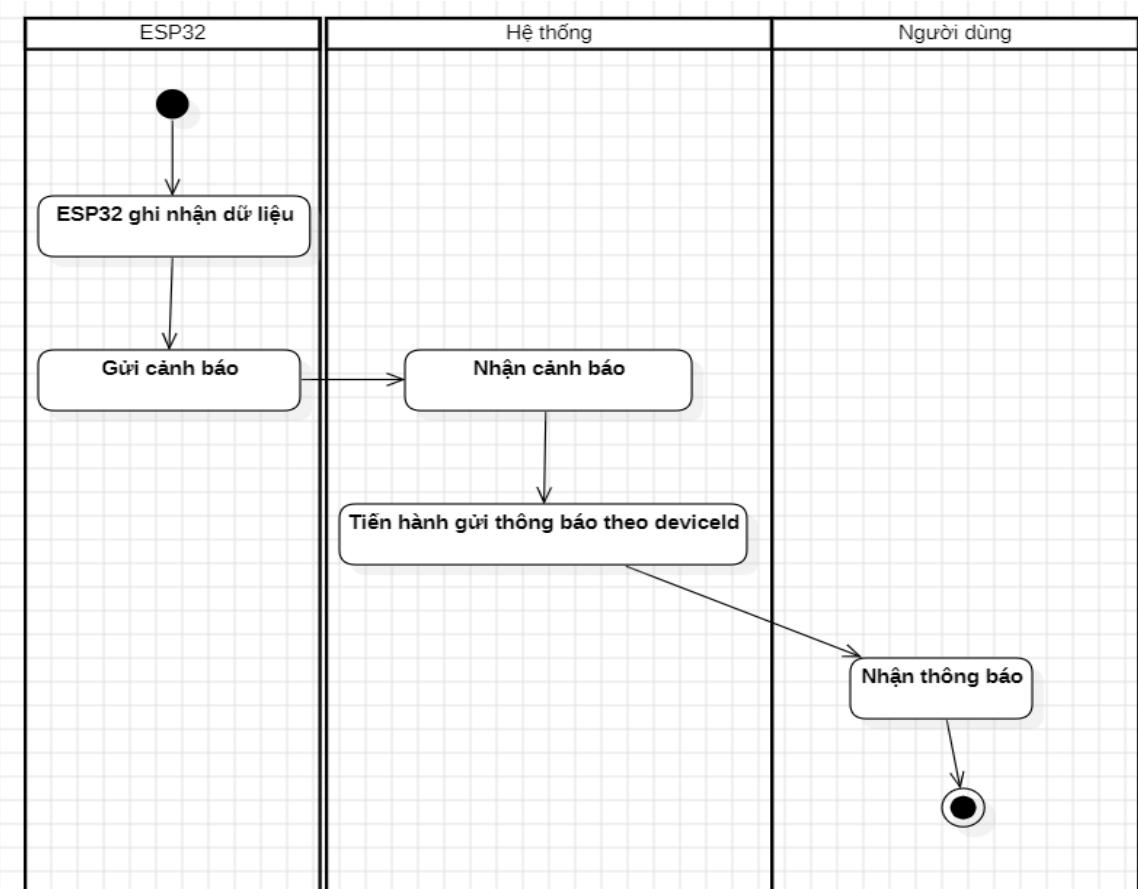
**Hình 46: Sơ đồ hoạt động xóa thiết bị**



**Hình 47: Sơ đồ hoạt động hiển thị dữ liệu cảm biến**



*Hình 48: Sơ đồ hoạt động thống kê*



*Hình 49: Sơ đồ hoạt động thông báo*

### 3.7 Thiết kế cơ sở dữ liệu

Trường	Kiểu dữ liệu	Bắt buộc	Đặc điểm
UserId	Number	Có	Duy nhất
Username	String	Có	
Email	String	Có	Duy nhất
PhoneNumber	String	Có	Mảng
Password	String	Có	Mã hóa
Devices	String	Không	Mảng
FcmToken	String	Không	Mảng

Bảng 22: Collection User

Trường	Kiểu dữ liệu	Bắt buộc	Đặc điểm
DeviceId	String	Có	Duy nhất
UserId	Number	Có	Tham chiếu qua User
deviceName	String	Có	
Location	String	Có	
Active	Bool	Có	

Bảng 23: Collection Device

Trường	Kiểu dữ liệu	Bắt buộc	Đặc điểm
UserId	Number	Có	
DeviceId	String	Có	
Temperature	Number	Có	
Humidity	Number	Có	
SmokeLevel	Number	Có	
FlameDetected	Bool	Có	
TimeStamp	Date	Có	

Bảng 24: Collection Sensordata\_Raw

Trường	Kiểu dữ liệu	Bắt buộc	Đặc điểm
DeviceId	String	Có	
AverageTemperature	Number	Có	
AverageHumidity	Number	Có	
AverageSmokeLevel	Number	Có	
FlameDetected	Bool	Có	
Date	Date	Có	

*Bảng 25: Collection SensorData*

❖ **Nhiệm vụ của các Collection**

- Users: Chứa các thông tin của người dùng mục đích giúp xác thực và xác định người dùng với thiết bị.
- Devices: Chứa các thông tin thiết bị, kết nối đến User qua UserId
- SensorDataRaw: Lưu trữ dữ liệu thô được ghi liên tục trong ngày.
- SensorDatas: Lưu dữ liệu tổng hợp từ cảm biến: trung bình theo ngày, tối ưu truy vấn hiển thị biểu đồ.

❖ **Quan hệ giữa các Collection**

- 1 người dùng → N thiết bị thông qua userId.
- 1 thiết bị → N bản ghi sensor thô (sensorDataRaw) thông qua deviceId.
- 1 thiết bị (devices) → N bản ghi tổng hợp (sensordatas) thông qua deviceId.

## CHƯƠNG IV: TRIỂN KHAI HỆ THỐNG

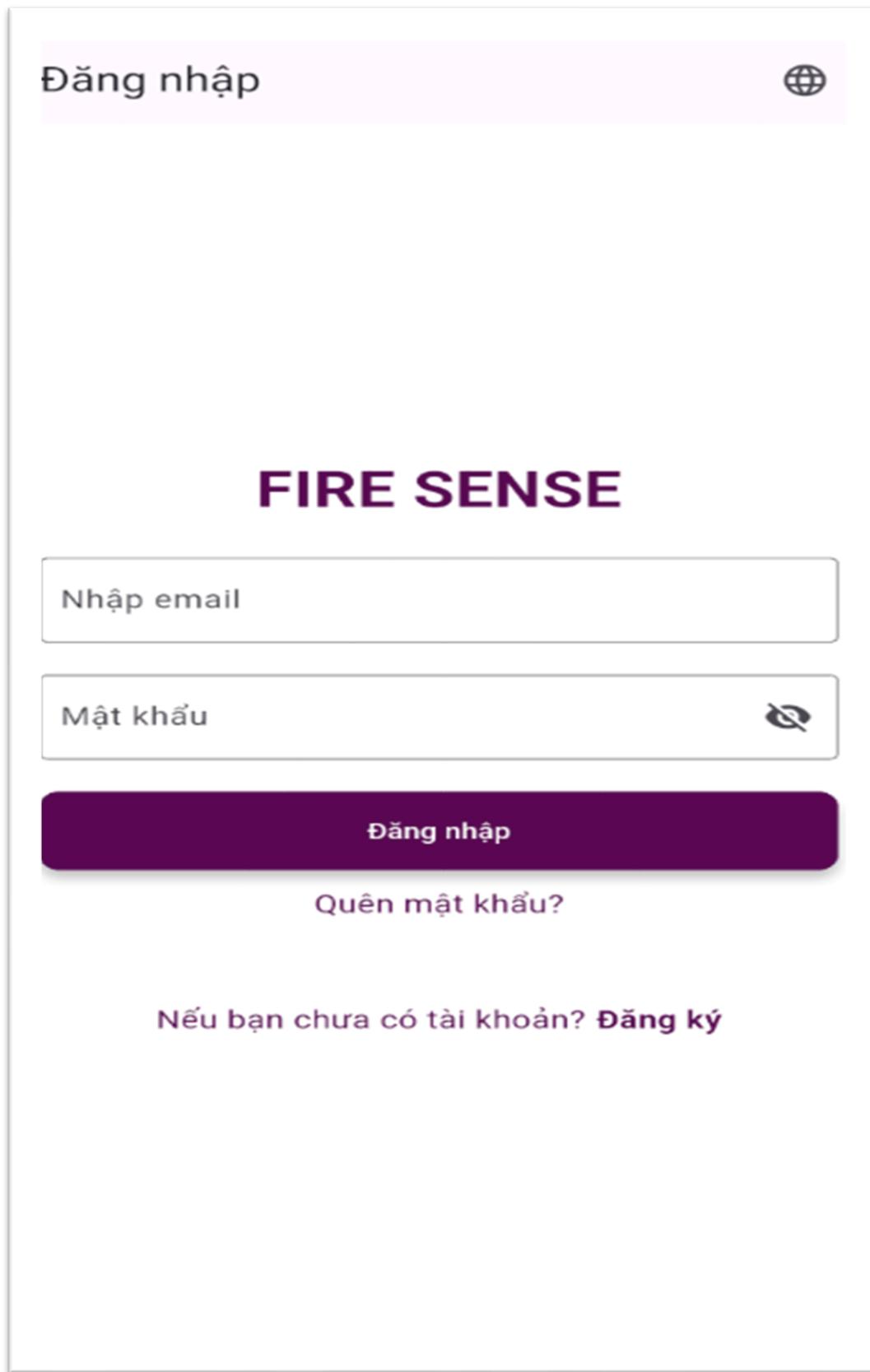
### 4.1 Cấu hình ESP32 và cảm biến

Trong hệ thống, vi điều khiển **ESP32** đóng vai trò trung tâm, kết nối với các cảm biến để thu thập dữ liệu môi trường như nhiệt độ, độ ẩm, khí gas, lửa, đồng thời điều khiển các thiết bị cảnh báo.

Thiết bị	Chức năng	Chân ESP32
Cảm biến khí MQ-2	Đo nồng độ khói/gas (analog & digital)	A0 = GPIO 33 D0 = GPIO 13
Cảm biến lửa	Phát hiện ngọn lửa qua ánh sáng hồng ngoại	A0 = GPIO 32 D0 = GPIO 34
Đồng hồ thời gian thực DS3231	Ghi lại thời gian thực khi phát hiện sự kiện	SDA = GPIO 16, SCL = GPIO 17
Cảm biến nhiệt độ độ ẩm DHT22	Đo nhiệt độ độ ẩm	Data = GPIO4
Buzzer (còi cảnh báo)	Cảnh báo bằng âm thanh	GPIO 15
LED (đỏ, vàng, xanh)	Hiển thị trạng thái thiết bị	Đỏ: GPIO 27 Vàng: GPIO 26 Xanh: GPIO 25

*Bảng 26: Bảng cấu hình lắp đặt cảm biến*

#### 4.2 Giao diện ứng dụng



*Hình 50: Giao diện đăng nhập*

← Quên mật khẩu?



email

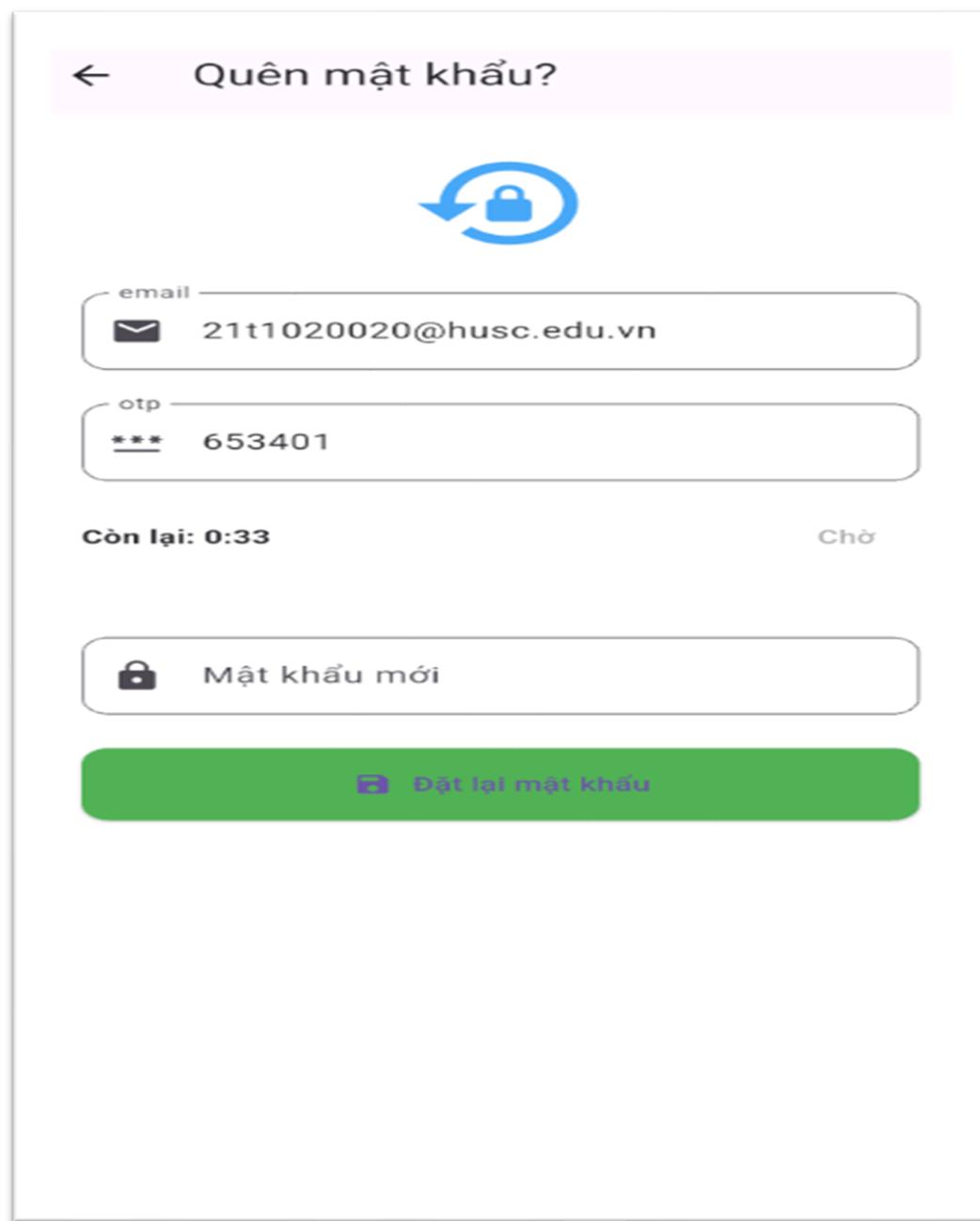
 21t1020020@husc.edu.vn

> Gửi OTP

*Hình 51: Giao diện quên mật khẩu*



**Hình 52: OTP Gửi về Gmail**



**Hình 53: OTP Khi được nhập đúng**

## FIRE SENSE

Nhập email

Nhập số điện thoại

Tên người dùng

Mật khẩu



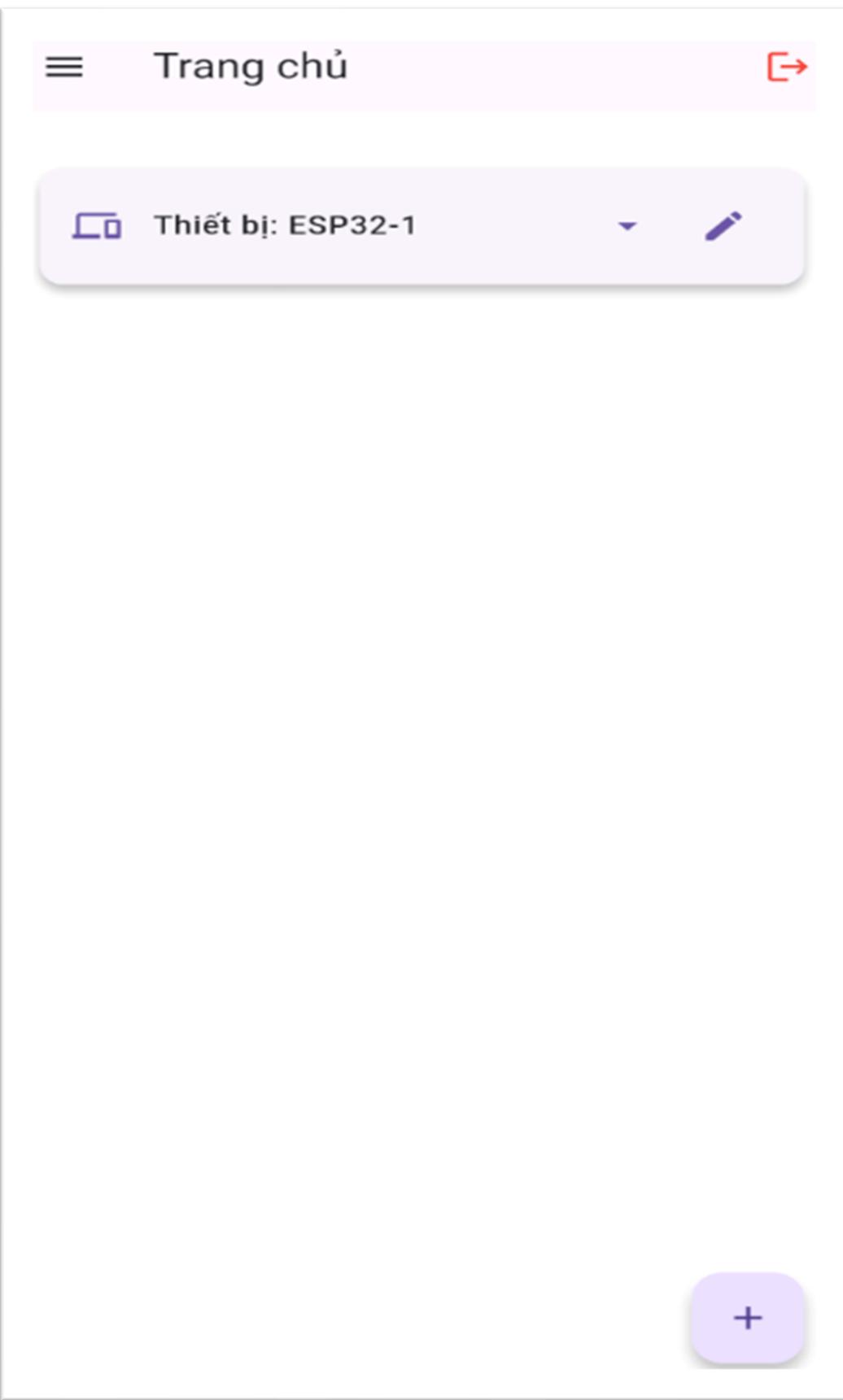
Xác nhận mật khẩu



Đăng ký

Đã có tài khoản? [Đăng nhập](#)

Hình 54: Giao diện đăng ký tài khoản



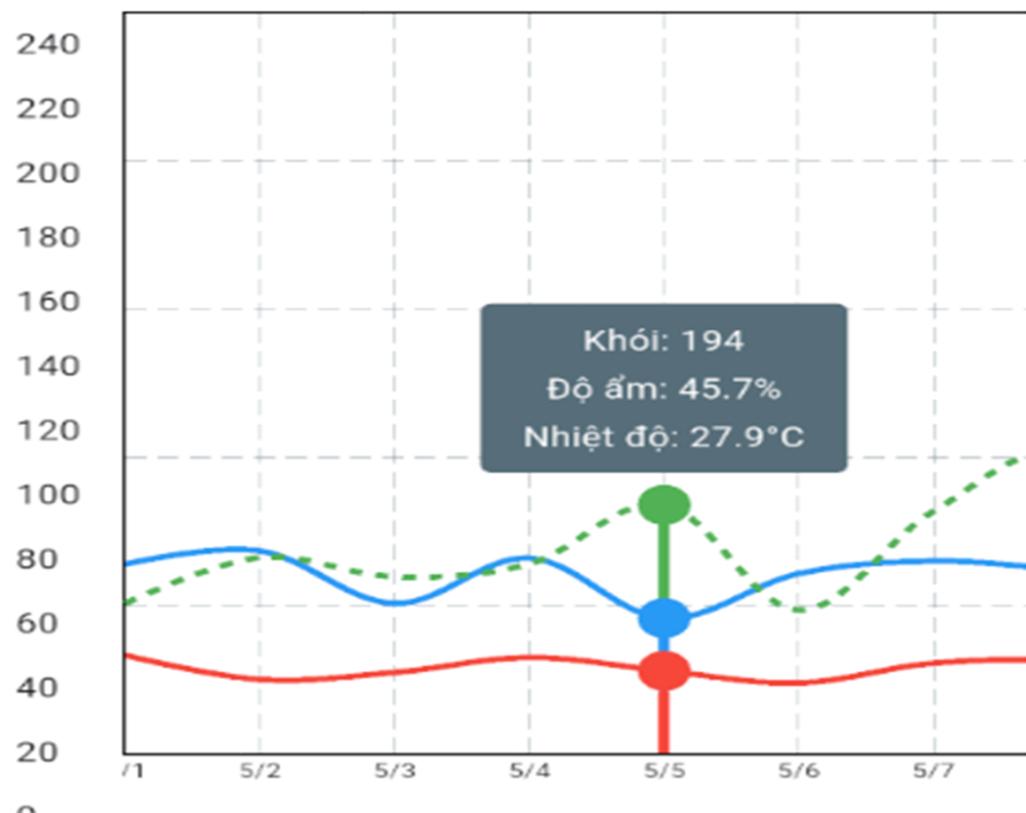
**Hình 55: Trang chủ**



Trang chủ



Thiết bị: ESP32-01



**Biểu đồ thống kê**

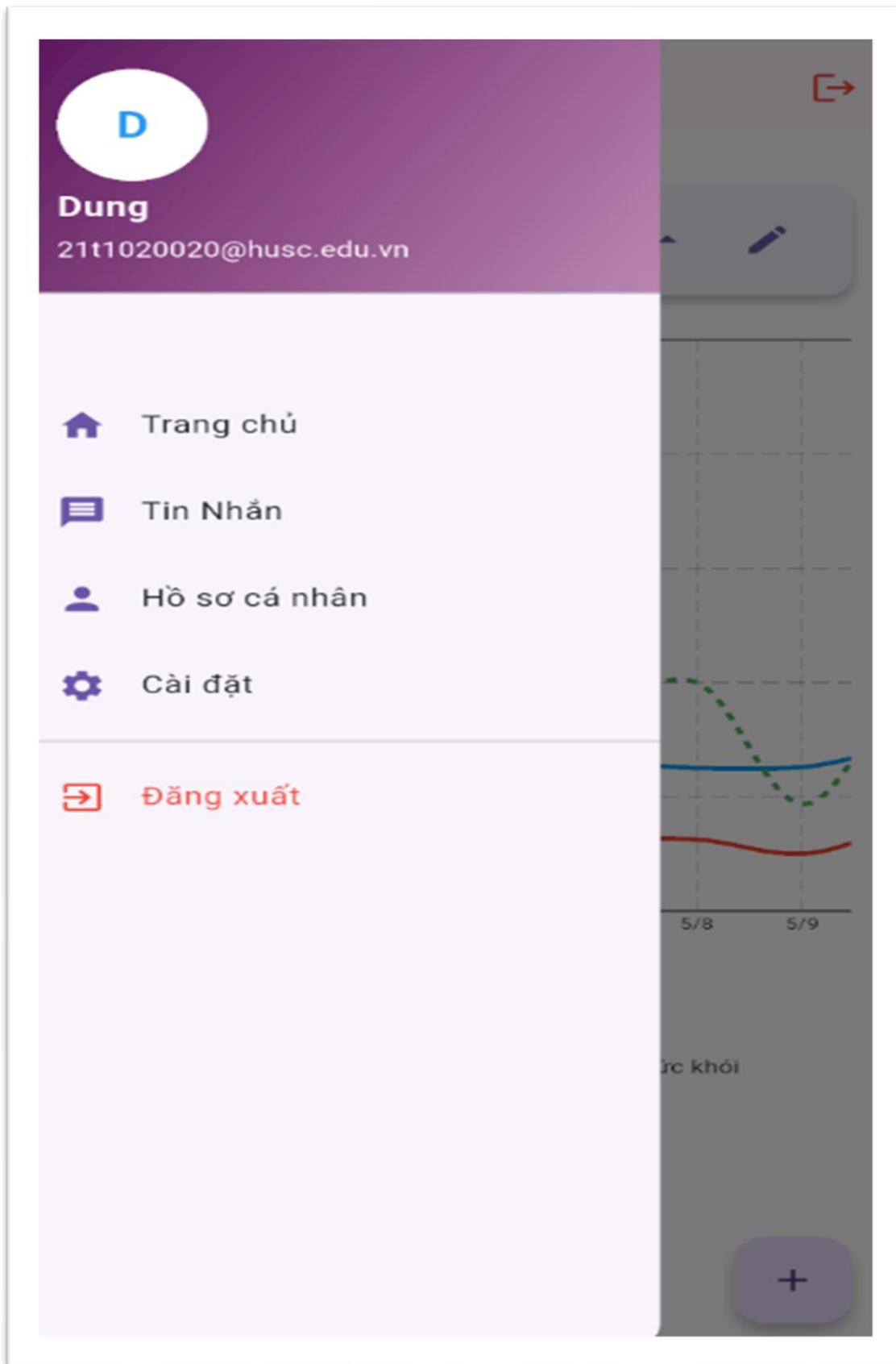
■ Nhiệt độ (°C)

■ Độ ẩm (%)

■ Mức khói



**Hình 56: Biểu đồ thống kê**



Hình 57: Drawer danh mục



## Cài đặt

### Cài đặt chung



Ngôn ngữ

Tiếng Việt ▾



Chế độ hiển thị



### Tài khoản



Thay đổi mật khẩu >



Đăng xuất >

### Thông tin



app\_version

Hình 58: Giao diện cấu hình ứng dụng

## Thông báo hệ thống

 Mức khói: 487

27/05/2025 22:17

 Mức khói: 1350

27/05/2025 22:18

 Mức khói: 1350

27/05/2025 22:18

 Mức khói: 1456

27/05/2025 22:18

 Mức khói: 1456

27/05/2025 22:18

 Mức khói: 1213

27/05/2025 22:18

 Mức khói: 1213

27/05/2025 22:18

 Mức khói: 647

27/05/2025 22:18

 Mức khói: 647

27/05/2025 22:18

 Mức khói: 1534

27/05/2025 22:18

 Mức khói: 1534

27/05/2025 22:18

*Hình 59: Giao diện lịch sử thông báo*

← Thông tin tài khoản

Tên người dùng



Dung

email



21t1020020@husc.edu.vn

Số điện thoại



Nhập số điện thoại

+ Thêm

Số điện thoại



0919092213



0918789232



Cập nhật thông tin

Hình 60: Giao diện chỉnh sửa thông tin cá nhân

← Thêm Thiết Bị

 Mã Thiết Bị

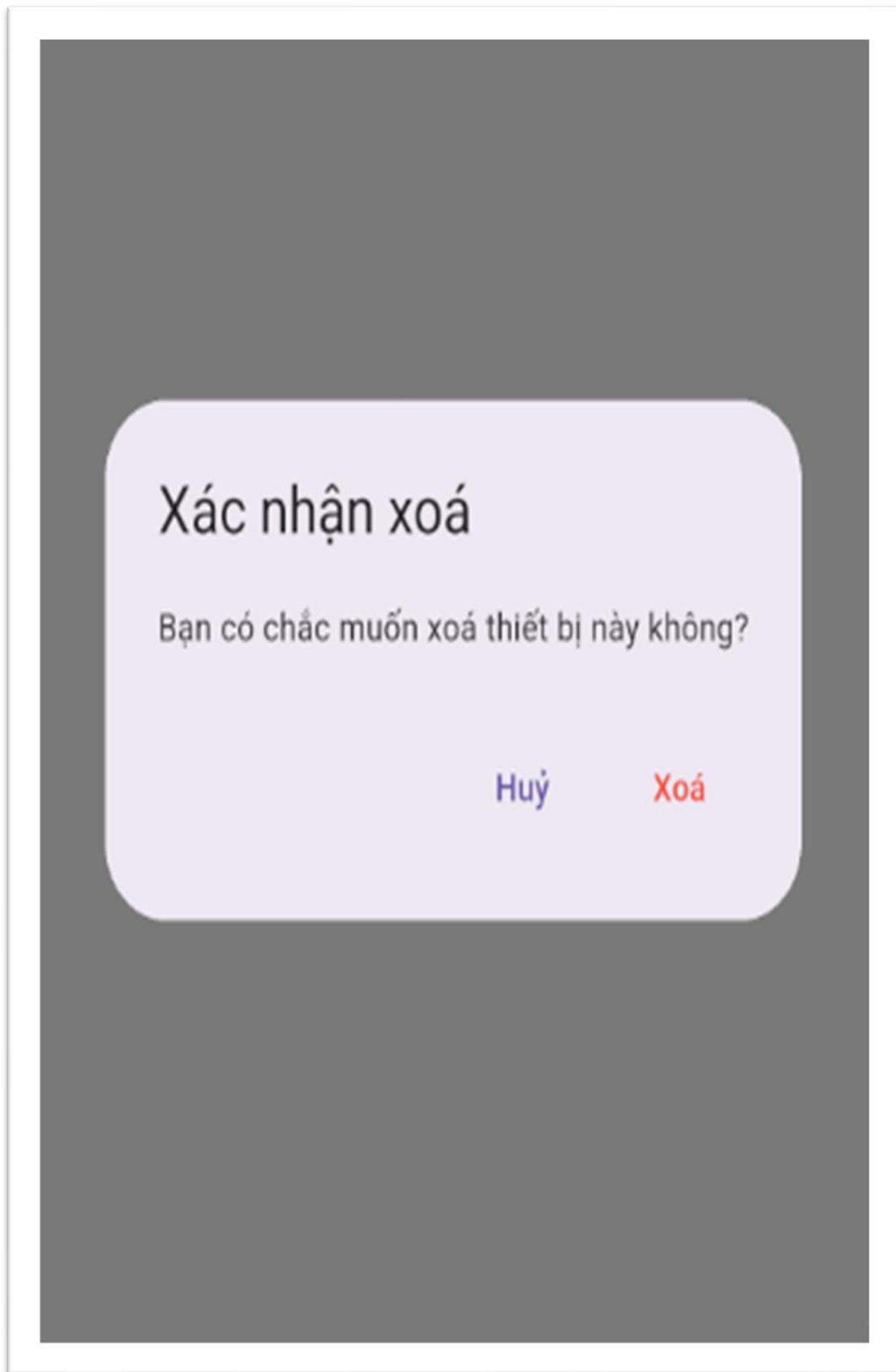
 Tên Thiết Bị

 Vị Trí

 Kích hoạt thiết bị 

+ Thêm Thiết Bị

**Hình 61: Giao diện thêm thiết bị**



*Hình 62: Pop up xác nhận xoá thiết bị*



## Chỉnh sửa thiết bị



Tên Thiết Bị —————



ESP32-1

Vị Trí —————



Phòng khách 1

Kích hoạt thiết bị



Cập nhật thông tin

**Hình 63: Giao diện chỉnh sửa thông tin thiết bị**

← Thông tin thiết bị



Nhiệt độ

24°C



Độ ẩm

40%



Mức khói

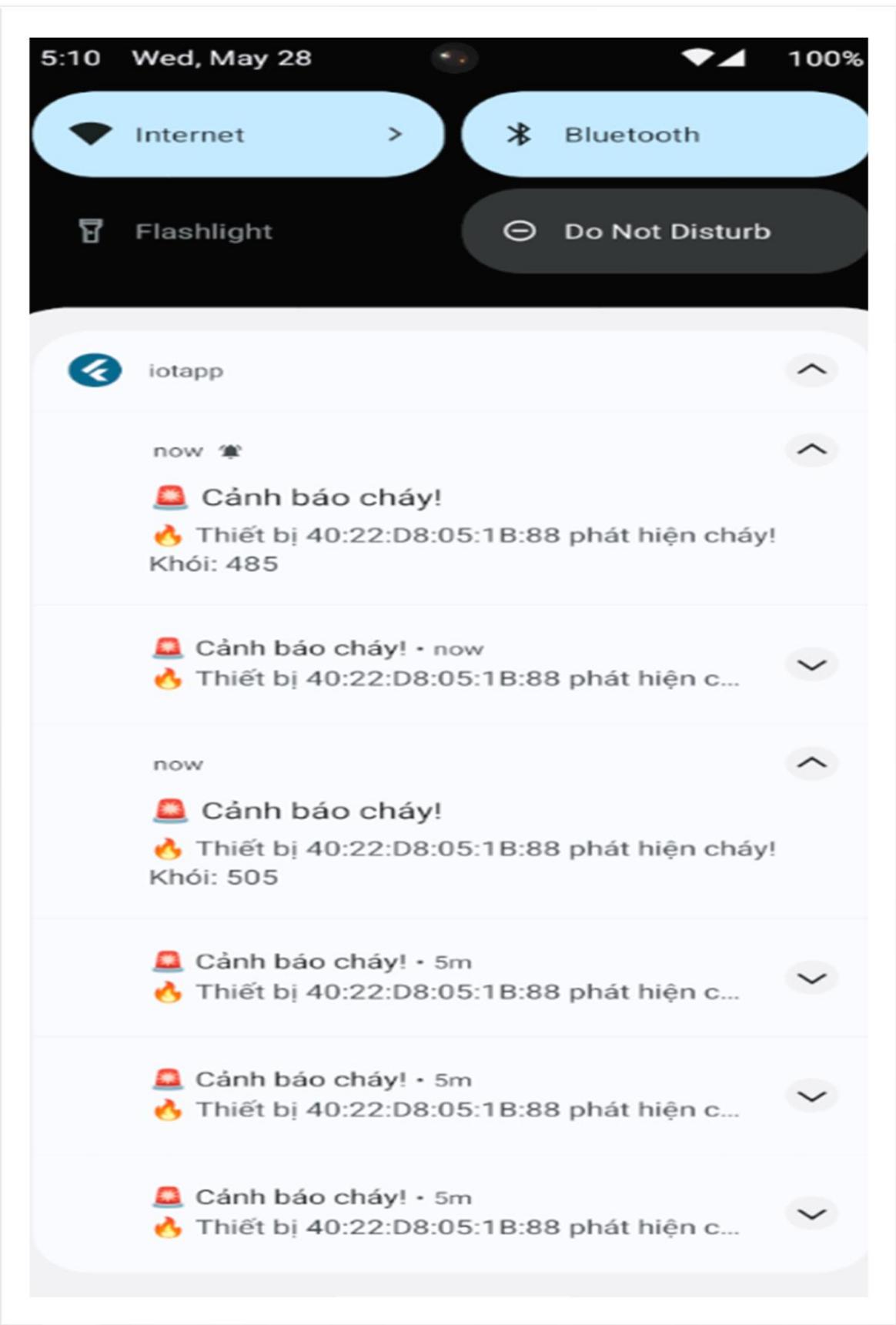
169



Điều khiển relay



Hình 64: Giao diện hiển thị các chỉ số lấy được từ cảm biến



*Hình 65: Thông báo gửi về*

### 4.3 Kiểm thử hệ thống

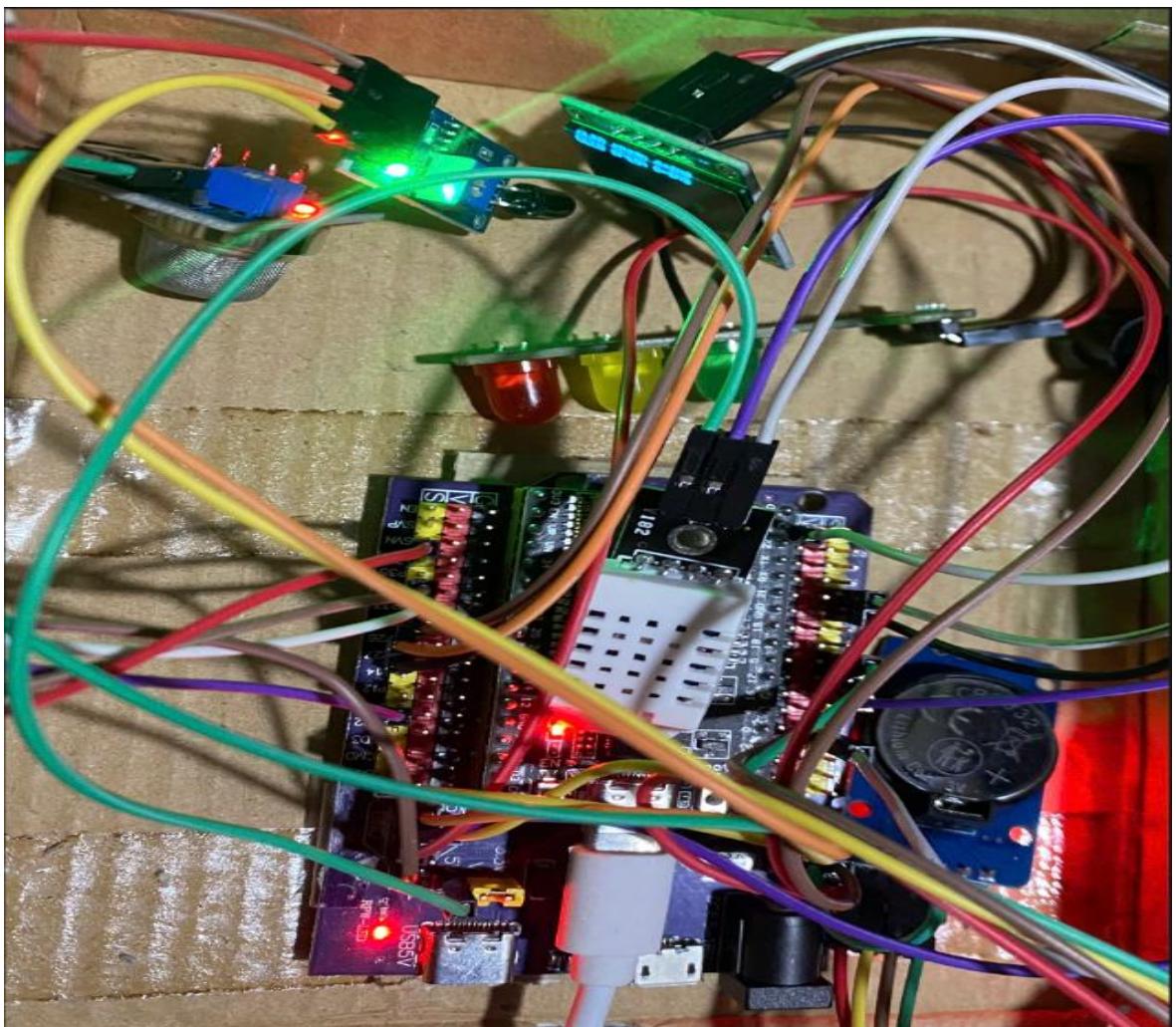
- Kiểm thử bằng phần mềm

Hạng mục kiểm thử	Kết quả
Kết nối ESP32 và cảm biến gửi dữ liệu lên server	<p>-ESP-32</p> <pre>*wm:Connecting to SAVED AP: TCD *wm:connectTimeout not set, ESP waitForResult... *wm:AutoConnect: SUCCESS *wm:STA IP Address: 172.20.10.2 WiFi đã kết nối! Địa chỉ IP hiện tại: 172.20.10.2 ✓ WebSocket đã kết nối, gửi xác thực... [WS] Đã gửi xác thực device_authenticate: {"type": "device_authenticate", "deviceId": "40:22:D8:05:1B:88"}  Setup hoàn thành. Nhận tin nhắn từ server: {"type": "auth_success", "message": "Thiết bị xác thực thành công"}</pre> <p>- Server</p> <pre>04 22 3a 22 34 30 3a 32 32 3a 44 38 ... 11 more bytes&gt; ⌚ Server nhận message từ client: {"type": "device_authenticate", "deviceId": "40:22:D8:05:1B:88"} ⚡ Thiết bị 40:22:D8:05:1B:88 đã kết nối WebSocket không cần JWT ⚠ Một client vừa kết nối chờ xác thực</pre>
Nhận cảnh báo FCM	 <p>iotapp 2 ✓</p> <p>⚠ Cảnh báo cháy! 🔥 Thiết bị 40:22:D8:05:1B:88...</p> <p>⚠ Cảnh báo cháy! 🔥 Thiết bị 40:22:D8:05:1B:88...</p>
Tài khoản – Đăng nhập	<pre>{   "message": "Đăng nhập thành công",   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJEsImIhdCI6MTc0ODQ4NjI3NCwIZXwiIjoxNzQ5MDIxMzc0fQ.   Be4QnVRj2rcyBLiYJXPCE5jIY9b3OMpa-qeS9JD0Os",   "user": {     "_id": "681a1ff654d9f306283444fd",     "userId": 1,     "username": "Dung",     "email": "2111926020@husc.edu.vn",     "phonenumber": [       "0919692213",       "0918789232",       "0914824906"     ],     "password": "\$2b\$10\$R4sNLb1kfPBn0zV.p/1buucbV1PRsJNm7ecF0T6G1tA8np.fhB76",     "devices": [       "40:22:D8:05:1B:88"     ],     "fcmToken": [       "fdn9dm_eTnyYD60UOLche:_APAt91bFbJGbGlz07Z8RausqqDUBUHg--enhzK4chHiyQJUs8M_gfH--KKd0dmZ4zv6QODN9ExyL0p_QG8npPBHTwTYcqfPmcCoak       OJHTo1F8P9yST1s"     ],     "__v": 37   } }</pre>

<p>Điều khiển thiết bị từ app</p>	<p>- Client Flutter</p> <pre>I/flutter (11581): ⚡ Gửi alarm command: {type: alarm_command, command: alarm_on, deviceId: 40:22:D8:05:1B:88} I/flutter (11581): ⚡ Gửi alarm command: {type: alarm_command, command: alarm_off, deviceId: 40:22:D8:05:1B:88}</pre> <p>- Server</p> <pre>⚡ Server nhận message từ client: {"type": "alarm_command", "command": "alarm_on", "deviceId": "40:22:D8:05:1B:88"} ⚡ Gửi lệnh đến thiết bị: 40:22:D8:05:1B:88, command: alarm_on</pre> <p>- Client ESP-32</p> <pre>Nhận tin nhắn từ server: {"type": "alarm_command", "command": "alarm_on", "deviceId": "40:22:D8:05:1B:88"} ⚠ Còi báo bật lại Nhận tin nhắn từ server: {"type": "alarm_command", "command": "alarm_off", "deviceId": "40:22:D8:05:1B:88"} ⚠ Còi báo bị tắt từ xa</pre>
-----------------------------------	---

*Bảng 27: Ghi nhận Log trong kiểm thử*

- Kiểm thử ở điều kiện thực tế



*Hình 66: Sản phẩm lắp đặt thực tế*

Kịch bản sử dụng	Kết quả quan sát
ESP32 phát mạng ban đầu	Khi chưa cấu hình Wi-Fi, ESP32 phát sóng Wi-Fi ở chế độ Access Point (AP), có tên đuôi _config. Người dùng có thể kết nối để thiết lập Wi-Fi ban đầu.
ESP32 hiển thị deviceId	Trên màn hình OLED Device ID dạng mã QR code để người dùng quét và liên kết thiết bị với tài khoản trên ứng dụng.
Hiển thị dữ liệu cảm biến môi trường	Màn hình OLED và ứng dụng Flutter hiển thị chính xác các thông số: nhiệt độ, độ ẩm đo từ cảm biến DHT22.
Phát hiện lửa bằng bật lửa gần cảm biến	Khi đưa lửa gần cảm biến, còi hú cảnh báo, màn hình hiện chữ "Cháy !!!", và tin nhắn cảnh báo được gửi qua FCM đến điện thoại .
Phát hiện khí gas	Cảm biến MQ-2 phát hiện khí gas ở nồng độ nguy hiểm, hệ thống kích hoạt còi cảnh báo, hiển thị "Cháy !!!", đồng thời gửi cảnh báo FCM tức thì.

Bảng 28: Kịch bản và kết quả kiểm thử ở phần cứng

## CHƯƠNG V: KẾT QUẢ VÀ ĐÁNH GIÁ

### 5.1 Kết quả đạt được

#### - Chức năng đã phát triển thành công

Chức năng	Mô tả chi tiết
Đọc dữ liệu cảm biến môi trường	- Đọc dữ liệu từ DHT22 (nhiệt độ, độ ẩm), MQ2 (khói, khí gas), Flame sensor (lửa)
Gửi dữ liệu cảm biến lên server	- Dữ liệu được gửi liên tục. - Lưu trữ vào MongoDB qua server Node.js
Nhận cảnh báo cháy qua FCM	- Khi vượt ngưỡng, hệ thống gửi cảnh báo FCM đến người dùng
Ứng dụng Flutter – hiển thị dữ liệu	- Hiển thị thông số cảm biến theo thời gian thực (WebSocket)
Xác thực người dùng – JWT	- Đăng nhập bằng JWT token, lưu trữ bằng SharedPreferences
Điều khiển Relay từ xa	Giao diện Flutter bật/tắt relay điều khiển còi báo động.
Quản lý nhiều thiết bị	- Hệ thống phân biệt dữ liệu theo deviceId, mỗi người dùng có thể quản lý nhiều thiết bị
Bảo mật kết nối	Server được cấu hình sử dụng HTTPS để mã hóa dữ liệu truyền tải.

*Bảng 29: Bảng tổng kết các chức năng đã phát triển thành công*

### 5.2 Hạn chế của hệ thống

Hệ thống hiện tại đã hoạt động ổn định trong các chức năng cơ bản. Tuy nhiên, vẫn còn tồn tại nhiều hạn chế về mặt kỹ thuật, khả năng mở rộng và tính độc lập, cụ thể như sau:

- **Chưa tích hợp kết nối mạng di động (SIM)**
  - ESP32 hiện chỉ kết nối được với Wi-Fi nội bộ. Trong điều kiện mất Wi-Fi hoặc cần triển khai tại khu vực không có mạng cố định, hệ thống sẽ không thể hoạt động.
  - Chưa hỗ trợ các module SIM (LTE) để gửi dữ liệu qua mạng di động, dẫn đến giới hạn triển khai thực tế (như nông thôn, vùng núi, công trình ngoài trời...).
- **Vấn đề hiệu suất và khả năng mở rộng**
  - Giới hạn đồng thời: Hệ thống hiện tại chỉ thử nghiệm với số lượng thiết bị và người dùng nhỏ. Nếu triển khai quy mô lớn sẽ cần điều chỉnh kiến trúc backend để hỗ trợ tải cao.
  - Tốc độ phản hồi: Trong một số điều kiện mạng yếu, dữ liệu gửi từ ESP32 có độ trễ lên đến vài giây, ảnh hưởng đến tính tức thời của cảnh báo cháy.
- **Hạn chế về chức năng và giao diện**
  - Giao diện Flutter mới chỉ cung cấp các tính năng cơ bản như hiển thị dữ liệu, đăng nhập và điều khiển thiết bị.
  - Giao diện chưa thân thiện: Giao diện người dùng (UI) thiết kế đơn giản, còn thô sơ và thiếu tính thẩm mỹ. Bố cục chưa tối ưu, màu sắc chưa hài hòa, thiếu biểu tượng trực quan và chưa có trải nghiệm người dùng. .

### **5.3 Đề xuất hướng phát triển**

Dựa trên các hạn chế đã nêu và tiềm năng ứng dụng thực tiễn, hệ thống có thể được phát triển theo các hướng sau:

- **Cải thiện hiệu suất và hạ tầng**
  - Tăng khả năng chịu tải: Sử dụng các kỹ thuật như caching, load balancing, phân cụm dữ liệu để tăng độ ổn định khi số lượng thiết bị và người dùng tăng cao.

- **Nâng cao trải nghiệm người dùng**
  - Cải thiện giao diện Flutter: Thiết kế lại UI hiện đại, màu sắc thân thiện, dễ thao tác, có hướng dẫn sử dụng.
- **Nghiên cứu và ứng dụng mới**
  - Ứng dụng trí tuệ nhân tạo (AI): Phân tích dữ liệu cảm biến để phát hiện sớm nguy cơ cháy nổ qua xu hướng tăng dần nhiệt độ, độ ẩm.
  - Tích hợp module sim để gia tăng tính linh hoạt của thiết bị, giúp thiết bị hoạt động độc lập với Wi-Fi, đặc biệt phù hợp trong môi trường ngoài trời, vùng không có kết nối mạng cố định (như rừng, trang trại, nhà kho,...) và có thể gửi thông báo sms hoặc trực tiếp gọi điện cho người dùng để đưa ra cảnh báo.
- **Bảo trì và nâng cấp**
  - Bảo trì định kỳ: Thiết lập lịch kiểm tra hệ thống định kỳ, cập nhật phần mềm/firmware thường xuyên để đảm bảo an toàn và ổn định.
  - Lập kế hoạch nâng cấp: Theo dõi xu hướng công nghệ mới về IoT, bảo mật, giao tiếp không dây để sẵn sàng nâng cấp hệ thống phù hợp trong tương lai.
  - Nâng Cấp Phần Cứng – Tự Động Chữa Cháy: Có thể kết hợp với các hệ thống chữa cháy tự động như bơm nước áp lực.... Hạn chế tối đa thiệt hại về tài sản và con người.

## **TÀI LIỆU THAM KHẢO**

- [1] ESPRESSIF, “ESP32,” [Online]. Available:  
[https://www.espressif.com/sites/default/files/documentation/esp32\\_data\\_sheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_data_sheet_en.pdf). [Accessed: 07-May-2025].
- [2] Dart, “Dart documentation,” [Online]. Available:  
<https://dart.dev/docs>. [Accessed: 08-May-2025].
- [3] Flutter, “Flutter documentation,” [Online]. Available:  
<https://docs.flutter.dev/>. [Accessed: 15-May-2025].
- [4] SlideShare, “Luận văn thạc sĩ: Tìm hiểu Flutter và ứng dụng,” [Online]. Available: <https://fr.slideshare.net/slideshow/luan-van-thac-si-tim-hieu-flutter-va-ung-dung/251409804>. [Accessed: 07-June-2024].
- [5] Firebase, “Firebase documentation,” [Online]. Available:  
<https://fptshop.com.vn/tin-tuc/thu-thuat/firebase-la-gi-159218>.  
[Accessed: 20-May-2025].
- [6] Firebase, “Firebase documentation,” [Online]. Available:  
<https://firebase.google.com/docs>. [Accessed: 07-Jun-2025].
- [7] Node.js Foundation, “About Node.js,” [Online]. Available:  
<https://nodejs.org/en/about>. [Accessed: 2--May-2025].
- [8] ExpressJS, “Introduction to Express,” [Online]. Available:  
<https://expressjs.com/>. [Accessed: 22-May-2025].
- [9] JWT.io, “Introduction to JWT,” [Online]. Available:  
<https://jwt.io/introduction>. [Accessed: 22-May-2025].
- [10] FPTShop, “WebSocket là gì?,” [Online]. Available:  
<https://fptshop.com.vn/tin-tuc/danh-gia/websocket-la-gi-168400>.  
[Accessed: 13-May-2025].