

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

MÔN HỌC: ĐỒ ÁN THIẾT KẾ LUẬN LÝ

**MY SMART HOME**

Giảng viên hướng dẫn: Nguyễn Thiên Ân

Học kỳ: HK241

Lớp: L03

Sinh viên thực hiện	Mã số sinh viên	Hoàn thành
Nguyễn Chí Cường	2210430	90%
Đỗ Huy Minh Dũng	2210568	100%
Nguyễn Trường Giang	2210830	100%
Võ Nguyên Giáp	2110142	100%

Thành phố Hồ Chí Minh, Tháng 12/2024

## Contents

<b>1</b>	<b>Giới thiệu</b>	<b>3</b>
1.1	Dẫn nhập . . . . .	3
1.2	Giới thiệu đề tài . . . . .	4
<b>2</b>	<b>Yêu cầu</b>	<b>5</b>
2.1	Yêu cầu người dùng . . . . .	5
2.2	Yêu cầu hệ thống . . . . .	5
<b>3</b>	<b>Thiết kế hệ thống</b>	<b>5</b>
3.1	Kiến trúc hệ thống . . . . .	5
3.2	Khối Things . . . . .	6
3.3	Khối IoT Gateway . . . . .	6
3.4	Khối Server . . . . .	6
3.5	Khối ứng dụng . . . . .	7
<b>4</b>	<b>Use-case Details</b>	<b>8</b>
4.1	Use Case Diagram . . . . .	8
4.2	Kích hoạt các thiết bị từ xa . . . . .	9
4.3	Quản lý tình trạng an ninh . . . . .	10
4.3.1	Quan sát khu vực sinh hoạt từ xa . . . . .	10
4.3.2	Thông báo và cảnh báo nguy hiểm . . . . .	11
4.4	Tạo các điều kiện tự động bật/tắt thiết bị . . . . .	13
4.4.1	Thêm một điều kiện bật/tắt thiết bị . . . . .	13
4.4.2	Xóa một điều kiện bật/tắt thiết bị . . . . .	14
4.5	Lên lịch bật/tắt các thiết bị . . . . .	16
4.5.1	Thêm một hẹn giờ bật/tắt thiết bị tự động . . . . .	16
4.5.2	Xóa một hẹn giờ bật/tắt thiết bị tự động . . . . .	17
<b>5</b>	<b>Class diagram</b>	<b>19</b>
5.1	Model View Controller . . . . .	19
5.2	Class diagram . . . . .	19
<b>6</b>	<b>Devices</b>	<b>21</b>
6.1	Các loại cảm biến môi trường . . . . .	21
6.1.1	Cảm biến nhiệt độ, độ ẩm DHT11 . . . . .	21
6.1.2	Cảm biến ánh sáng . . . . .	21
6.1.3	Cảm biến khí . . . . .	22



---

6.1.4	Cảm biến lửa . . . . .	22
6.2	Thiết bị điều khiển trong gia đình . . . . .	23
6.3	Mạch ổn áp sử dụng LM2596 . . . . .	24
6.4	Vi điều khiển ESP32 và sơ đồ kết nối . . . . .	24
6.4.1	Sơ đồ kết nối module Esp32: . . . . .	25
6.5	Kiến trúc hệ thống Things: . . . . .	26
<b>7</b>	<b>Hiện thực hệ thống</b>	<b>29</b>
7.1	Giao tiếp giữa khối Things với GateWay . . . . .	29
7.2	Kết nối gateway với Server IO . . . . .	30
7.2.1	Giao thức MQTT . . . . .	30
7.2.2	Hiện thực kết nối gateway bằng giao thức MQTT . . . . .	30
7.2.3	Python trên Gateway . . . . .	30
7.3	Giao tiếp giữa ứng dụng với Server IO . . . . .	31
7.4	Tích hợp mô hình AI . . . . .	32
7.4.1	Nhận diện khuôn mặt . . . . .	32
7.4.2	Nhận diện giọng nói . . . . .	33
<b>8</b>	<b>Giao diện ứng dụng</b>	<b>34</b>
8.1	Trang chủ (Home) . . . . .	34
8.2	Trang hẹn giờ (Scheduler) . . . . .	34
8.3	Trang quan sát (Camera) . . . . .	35
8.4	Trang đặt luật (Set Rule) . . . . .	36
<b>9</b>	<b>Mã nguồn và Github</b>	<b>37</b>
<b>Tài liệu tham khảo</b>		<b>37</b>

# 1 Giới thiệu

## 1.1 Dẫn nhập

Cuộc cách mạng công nghiệp 4.0 đang diễn ra với sự phát triển vượt bậc về công nghệ của nhiều nước trên thế giới. Cuộc cách mạng này không chỉ tập trung vào tự động hóa mà còn kết hợp với nhiều công nghệ tiên tiến như trí tuệ nhân tạo (AI), Internet of Things (IoT) và dữ liệu lớn (BigData). Sự kết nối giữa các thiết bị và cảm biến thông minh cho phép thu thập và phân tích dữ liệu thời gian thực, từ đó tối ưu hóa quy trình sản xuất và cung cấp dịch vụ. Điều này không chỉ giúp tăng năng suất mà còn cải thiện chất lượng sản phẩm và đáp ứng nhanh chóng nhu cầu của khách hàng.

Bài báo cáo này sẽ tìm hiểu về một lĩnh vực ứng dụng Internet of Things. IoT đang mở ra một kỷ nguyên mới cho nhà thông minh (smart home), một khía cạnh quan trọng trong Cách mạng công nghiệp 4.0. Với khả năng kết nối và điều khiển từ xa, các thiết bị trong nhà như đèn, khóa cửa, điều hòa không khí và hệ thống an ninh có thể hoạt động một cách thông minh hơn. Nhà thông minh không chỉ mang lại sự tiện nghi mà còn giúp tiết kiệm năng lượng và tăng cường an toàn cho người sử dụng. Nhờ vào việc thu thập và phân tích dữ liệu từ các cảm biến, người dùng có thể tối ưu hóa việc sử dụng tài nguyên, tạo ra một không gian sống thoải mái và an toàn hơn cho gia đình.



## 1.2 Giới thiệu đề tài

Chất lượng không khí và môi trường trong nhà đóng vai trò quan trọng đối với sức khỏe con người, bởi con người dành phần lớn thời gian ở trong nhà.

Nhiệt độ của môi trường sinh hoạt cũng có những tác động đến nhiều mặt về chức năng sinh lý của cơ thể con người. Khi nhiệt độ trong nhà quá lạnh, cơ thể sẽ phải tiêu hao nhiều năng lượng để duy trì nhiệt độ cơ thể, dẫn đến tăng nguy cơ bị cảm lạnh, cúm, hạ thân nhiệt. Ngược lại, ở môi trường quá nóng con người sẽ dễ bị say nắng, mệt mỏi, khó thở, và mất nước, dẫn đến nhiều mồ hôi gây ra sự khó chịu.

Dộ ẩm trong nhà cũng có nhiều ảnh hưởng lớn đến sức khỏe, đặc biệt là hệ hô hấp và làn da. Độ ẩm quá thấp sẽ dẫn đến không khí bị khô và khiến da dễ bị nứt nẻ, môi khô, niêm mạc hô hấp có thể bị kích ứng, gây ra các vấn đề như khô mũi, viêm họng, và làm tăng nguy cơ nhiễm trùng hô hấp. Độ ẩm quá cao làm cho môi trường ẩm ướt tạo điều kiện thuận lợi cho vi khuẩn, nấm mốc, và các loại mầm bệnh phát triển, gây ra dị ứng, hen suyễn và các vấn đề về đường hô hấp. Nấm mốc còn có thể phát triển trên tường, trần nhà, đồ nội thất, gây hại cho sức khỏe nếu hít phải trong thời gian dài.

Ánh sáng cũng có những tác động lớn đến sức khỏe thể chất và tinh thần của con người. Cả ánh sáng tự nhiên và nhân tạo đều đóng vai trò quan trọng trong việc duy trì nhịp sinh học, tâm trạng, và sức khỏe. Ánh sáng ảnh hưởng trực tiếp đến chu kỳ thức - ngủ tự nhiên của cơ thể. Môi trường có chất lượng ánh sáng thấp sẽ làm rối loạn nhịp sinh học, gây ra mệt mỏi, giảm hiệu suất công việc và suy giảm sức khỏe của chúng ta. Môi trường có chất lượng ánh sáng không tốt còn gây ra nhiều vấn đề nghiêm trọng về thị lực và sức khỏe mắt cho những người lao động trí óc trong thời gian dài.

Ngoài ra chúng ta còn có nhiều vấn đề về an ninh. An ninh trong nhà không chỉ bảo vệ tài sản mà còn đảm bảo an toàn cho con người. Mất an ninh có thể dẫn đến những hậu quả nguy hiểm như nguy cơ bị trộm cắp, đột nhập, an ninh kém sẽ gây mất cảm giác an toàn. Để đảm bảo an ninh, cần có các biện pháp bảo vệ, có thể đặt camera giám sát, có hệ thống báo động và kiểm tra định kỳ khu vực sinh hoạt của chúng ta.

Mục tiêu cụ thể của dự án sẽ tạo ra được một hệ thống đồng bộ, giám sát được chất lượng môi trường và an ninh theo thời gian thực, dữ liệu có thể được lưu trữ và có người dùng có thể xem các thống kê để đánh giá về chất lượng môi trường. Bên cạnh điều khiển từ xa các thiết bị theo thời gian thực, ứng dụng còn cho phép người dùng thiết lập các ngưỡng một cách trực quan và tiện dụng để có thể điều khiển từ xa, tự động dựa vào các giá trị cảm biến theo thời gian thực.

## 2 Yêu cầu

### 2.1 Yêu cầu người dùng

Đối với người dùng (user) ứng dụng được yêu cầu phải cung cấp được các dịch vụ sau đây:

- **Xem các thông số môi trường:** Người dùng có thể xem các thông số về nhiệt độ, độ ẩm, ánh sáng trong môi trường làm việc hoặc sinh hoạt của mình.
- **Quan sát các biểu đồ về biến động môi trường:** Người dùng có thể xem các biểu đồ line chart về nhiệt độ, độ ẩm, ánh sáng trong một khoảng thời gian vừa qua.
- **Quan sát tình trạng an ninh:** Người dùng có thể theo dõi tình trạng an ninh trên ứng dụng, kiểm tra liệu có xuất hiện người quen, người lạ mặt, hoặc không có ai trong khu vực.
- **Xem nhật ký của hệ thống:** Người dùng có thể xem lịch sử hoạt động của hệ thống, bao gồm cập nhật các thông số môi trường, phát hiện có người trong khu vực.
- **Điều khiển các thiết bị từ xa:** Người dùng có thể bật tắt các thiết bị điện từ xa bằng các nút nhấn trực tuyến trên ứng dụng.
- **Tạo các điều kiện bật/tắt thiết bị tự động:** Bên cạnh thao tác trên ứng dụng một cách thủ công, người dùng có thể cài đặt để bật hoặc tắt các thiết bị điện khi môi trường đạt đến một thông số nào đó do người dùng đặt ra.
- **Lên lịch bật/tắt các thiết bị tự động:** Người dùng có thể lên lịch tự động bật hoặc tắt các thiết bị theo ý muốn.
- **Nhận cảnh báo nguy hiểm** Hệ thống phải có cơ chế gửi cảnh báo đến thiết bị của người dùng khi phát hiện xảy ra sự kiện nguy hiểm.

### 2.2 Yêu cầu hệ thống

Để có thể hoàn thiện một ứng dụng IoT tốt. Hệ thống cần đáp ứng một số yêu cầu cũng như sau:

- Hệ thống có giao diện thân thiện, dễ tương tác.
- Thời gian thực thi và phản hồi trong các hành động của hệ thống cần phải nhanh chóng, không được trễ quá 2 giây.

## 3 Thiết kế hệ thống

### 3.1 Kiến trúc hệ thống

Ứng dụng Internet of Things sẽ bao gồm 4 khối chức năng chính:

- **Khối Things:** Các thiết bị vật lý trong hệ thống, có chức năng thu thập, truyền dữ liệu và thực hiện các tác vụ theo yêu cầu.

- Khối IoT Gateway: Làm cầu nối giữa các thiết bị (Things) với khối Server, quản lý kết nối và đảm bảo dữ liệu từ thiết bị được truyền tải một cách an toàn và ổn định đến Server.
- Khối Sever: Khối Server đóng vai trò trung tâm trong hệ thống IoT, là nơi tiếp nhận, xử lý, lưu trữ và phân tích dữ liệu từ các thiết bị (Things) thông qua IoT Gateway. Nó cũng cung cấp các dịch vụ và API cho khối Ứng dụng để người dùng tương tác với hệ thống.
- Khối Ứng dụng: Khối này cung cấp giao diện cho người dùng cuối để họ có thể giám sát, quản lý và tương tác với các thiết bị IoT trong hệ thống.

Khối Things sẽ gửi dữ liệu đến khối IoT Gateway bằng phương thức giao tiếp Serial. Trong khi đó, các khối IoT Gateway, Sever và Ứng dụng người dùng sẽ liên lạc với nhau thông qua giao thức MQTT.

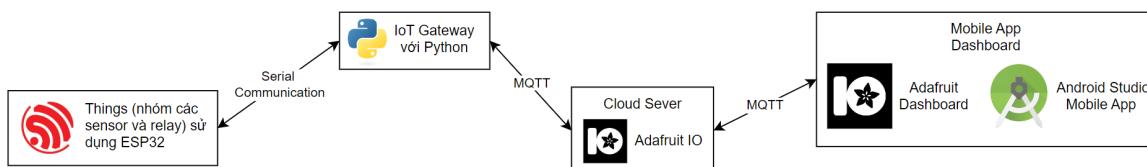


Figure 1: Kiến trúc chung của hệ thống IoT.

### 3.2 Khối Things

Thành phần chủ yếu của khối Things là các thiết bị vật lý và phần cứng như cảm biến, relay,... Các thiết bị này có chức năng thu thập dữ liệu, truyền dữ liệu và thực hiện các tác vụ cụ thể được yêu cầu. Các dữ liệu được thu thập sẽ được gửi đến server thông qua Gateway.

### 3.3 Khối IoT Gateway

IoT Gateway đóng vai trò làm cầu nối giữa các thiết bị (Things) và khối Server, quản lý kết nối và đảm bảo dữ liệu từ thiết bị được truyền tải một cách an toàn và ổn định đến Server. Khi nhận được dữ liệu từ khối Things, khối này sẽ gửi lên Server theo giao thức MQTT, đồng thời nếu nhận được lệnh điều khiển từ khối Sever, thì khối này sẽ truyền lệnh thực thi đến các thiết bị phần cứng tương ứng của khối Things. Khối này thường là vai trò của các vi điều khiển hoặc vi xử lý, các vi điều khiển hoặc vi xử lý nằm trong khối này sẽ được chạy một chương trình được người lập trình viết để hoàn thành các chức năng tương ứng.

### 3.4 Khối Server

Đây là nơi quản lý, lưu trữ và xử lý dữ liệu từ các thiết bị của khối Things, đồng thời đưa ra các phản hồi hoặc gửi các lệnh điều khiển đến khối Things thông qua khối IoT Gateway. Dữ liệu từ khối Things sẽ được lưu trữ, phân tích và xử lý tại Server, giúp phát hiện các sự kiện, xu hướng hoặc cảnh báo trong hệ thống.



### 3.5 Khối ứng dụng

Khối ứng dụng cung cấp giao diện cho người dùng cuối để họ có thể giám sát, quản lý và tương tác với các thiết bị IoT trong hệ thống một cách dễ dàng. Ứng dụng cho phép người dùng theo dõi dữ liệu từ xa. Người dùng có thể thực hiện các tác vụ điều khiển như bật/tắt thiết bị, điều chỉnh cài đặt thông qua ứng dụng. Ngoài ra ứng dụng còn cho phép người dùng quản lý và kiểm tra trạng thái hệ thống, lên lịch hoạt động cho thiết bị, và xem các báo cáo phân tích.

Đây là khối tương tác trực tiếp với người dùng, vì vậy khi hiện thực cần yêu cầu tính thẩm mỹ cao, nhưng cũng cần phải không được quá phức tạp để người dùng có thể tiếp cận một cách dễ dàng. Thông qua ứng dụng, người dùng có thể gửi các lệnh đến các thiết bị của khối Things thông qua sever và gateway để điều khiển hệ thống theo ý muốn của bản thân.

## 4 Use-case Details

### 4.1 Use Case Diagram

Use Case Diagram là một loại biểu đồ hành vi mô tả các tương tác giữa người dùng (hoặc các hệ thống bên ngoài) với hệ thống phần mềm. Nó tập trung vào việc thể hiện chức năng của hệ thống từ góc độ người dùng, trả lời câu hỏi "Hệ thống làm gì cho người dùng?". Use Case Diagram tập trung vào mục tiêu mà người dùng muốn đạt được.

Nhóm sẽ trình bày một số use case đặc trưng của hệ thống trong các mục phía sau.

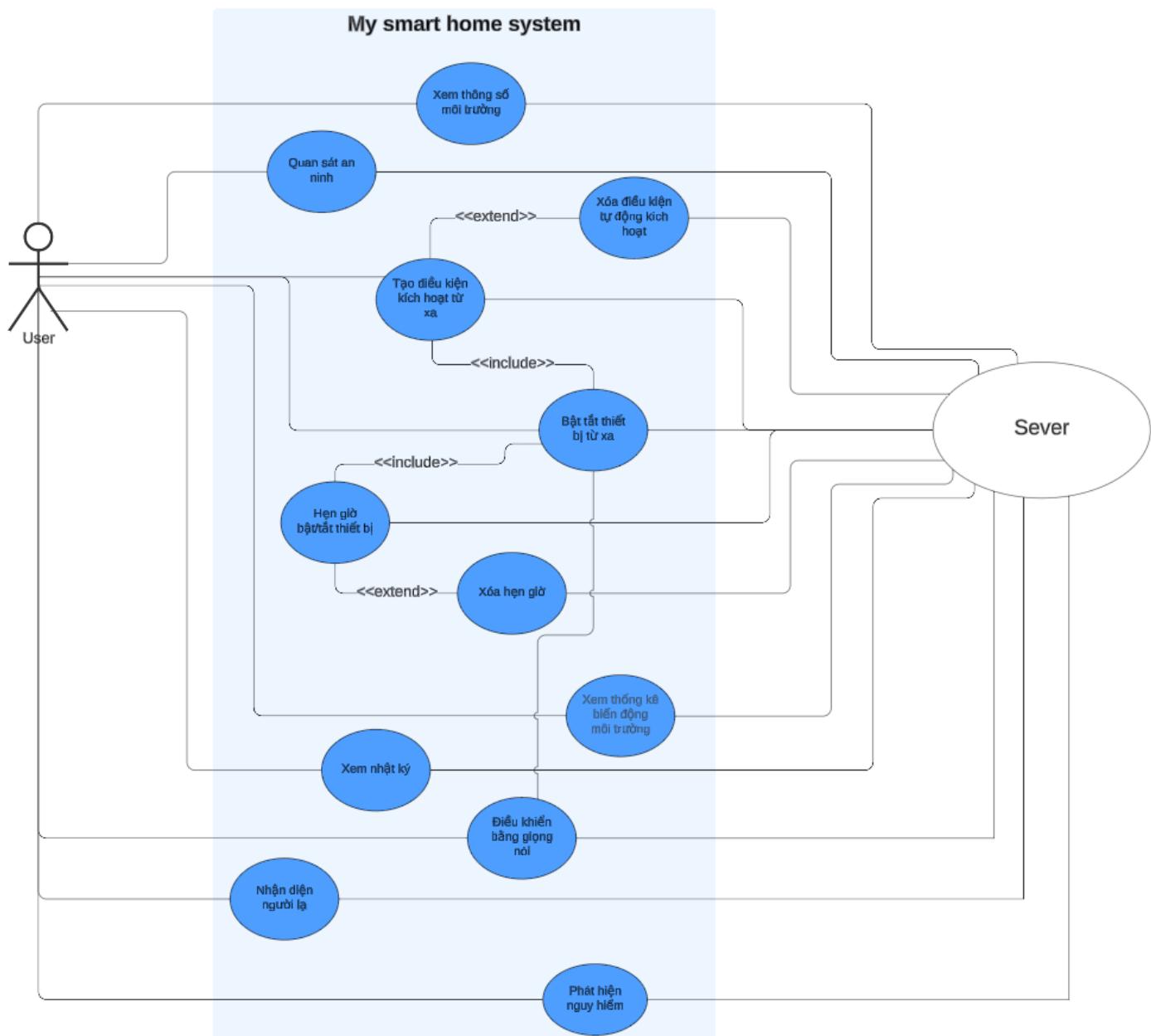


Figure 2: Use case Diagram

## 4.2 Kích hoạt các thiết bị từ xa

Use case ID	1
Tên use case	Kích hoạt các thiết bị từ xa
Actor	Người dùng, server
Mô tả	Hệ thống cung cấp các nút nhấn trực tuyến trên ứng dụng giúp cho người dùng có thể trực tiếp thực hiện thao tác bật hoặc tắt các thiết bị IoT trên điện thoại.
Tiền điều kiện	Hệ thống phải hoạt động bình thường, và các thiết bị nhận lệnh điều khiển phải tồn tại.
Normal flow	<ol style="list-style-type: none"><li>Người dùng nhấn vào biểu tượng "home" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>Ứng dụng hiển thị menu và xuất hiện các biểu tượng công tắc và các công tắc thể hiện trạng thái bật/tắt hiện tại của thiết bị.</li><li>Người dùng thao tác thay đổi trạng thái bật/tắt của thiết bị bằng cách nhấn vào công tắc trên màn hình.</li><li>Sever tiến hành gửi request bật/tắt thiết bị đến gateway theo yêu cầu của người dùng.</li><li>Thiết bị nhận request từ gateway gửi về và kích hoạt hoặc ngắt.</li><li>Công tắc thay đổi trạng thái hiện tại trên màn hình, tiếp tục lắng nghe yêu cầu từ người dùng.</li></ol>
Ngoại lệ	Kết nối đến sever thất bại, sever gửi request đến gateway không thành công hoặc thiết bị nhận yêu cầu không tồn tại. Khi một trong các vấn đề này xảy ra hệ thống sẽ báo lỗi.

Mở rộng chức năng: Ngoài việc sử dụng các công tắc trực tuyến trên ứng dụng di động ra hệ thống còn hỗ trợ người dùng có thể điều khiển các thiết bị bằng giọng nói.



## 4.3 Quản lý tình trạng an ninh

### 4.3.1 Quan sát khu vực sinh hoạt từ xa

Use case ID	2
Tên use case	Quan sát khu vực sinh hoạt từ xa
Actor	Người dùng
Mô tả	Ứng dụng cung cấp màn hình kết nối với camera được bố trí tại khu vực sinh hoạt của người dùng để người dùng có thể quan sát và theo dõi trực tiếp nhà của mình trên điện thoại.
Tiền điều kiện	Hệ thống quản lý camera và mạng lưới truyền dẫn dữ liệu phải hoạt động ổn định. Camera giám sát phải được kết nối với nguồn điện và mạng internet một cách ổn định để đảm bảo việc truyền tải hình ảnh trực tiếp, liên tục và chất lượng cao đến thiết bị đầu cuối của người dùng.
Normal flow	<ol style="list-style-type: none"><li>Người dùng nhấp vào biểu tượng "camera" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>Hệ thống thiết lập kết nối với camera.</li><li>Ứng dụng bắt đầu truyền tải hình ảnh trực tiếp từ camera đến thiết bị của người dùng.</li><li>Người dùng có thể xem hình ảnh trực tiếp từ camera trên màn hình thiết bị của mình</li><li>Người dùng có thể điều khiển camera để xoay, zoom, hoặc thay đổi góc quay.</li><li>Người dùng có thể chọn xem lại các bản ghi đã được camera ghi lại trong quá khứ bằng cách sử dụng thanh trượt thời gian hoặc các nút điều khiển khác.</li></ol>
Ngoại lệ	Kết nối mạng bị gián đoạn, camera không hoạt động do mất điện hoặc mất kết nối.

#### 4.3.2 Thông báo và cảnh báo nguy hiểm

Use case ID	3
Tên use case	Thông báo và cảnh báo nguy hiểm
Actor	Hệ thống
Mô tả	Hệ thống phát hiện sự kiện nguy hiểm và gửi thông báo đến thiết bị của người dùng.
Tiền điều kiện	Hệ thống hoạt động ổn định, kết nối mạng ổn định, các cảm biến giám sát đang hoạt động (không bị mất nguồn điện, mất kết nối).
Normal flow	<ol style="list-style-type: none"><li>1. Các cảm biến và thiết bị giám sát thu thập dữ liệu về môi trường xung quanh</li><li>2. Hệ thống so sánh dữ liệu thu thập được với các ngưỡng cảnh báo đã được thiết lập trước đó. Khi phát hiện dữ liệu vượt quá ngưỡng cảnh báo, hệ thống xác định rằng đã có sự cố xảy ra</li><li>3. Hệ thống kích hoạt cơ chế gửi thông báo nguy hiểm đến thiết bị của người dùng.</li><li>4. Người dùng nhận được thông báo trên điện thoại, máy tính hoặc các thiết bị khác.</li><li>5. Người dùng xác nhận đã nhận được thông báo.</li></ol>
Ngoại lệ	Mất kết nối mạng khiến thông báo gửi không thành công hoặc bị chậm trễ, thiết bị người dùng không hoạt động (bị tắt nguồn), các cảm biến đóng vai trò theo dõi và so sánh bị mất nguồn điện hoặc kết nối.

Mở rộng chức năng: Người dùng có thể nhận thông báo về việc có người lạ tiếp cận khu vực sinh hoạt được hỗ trợ bởi AI tích hợp trong camera.

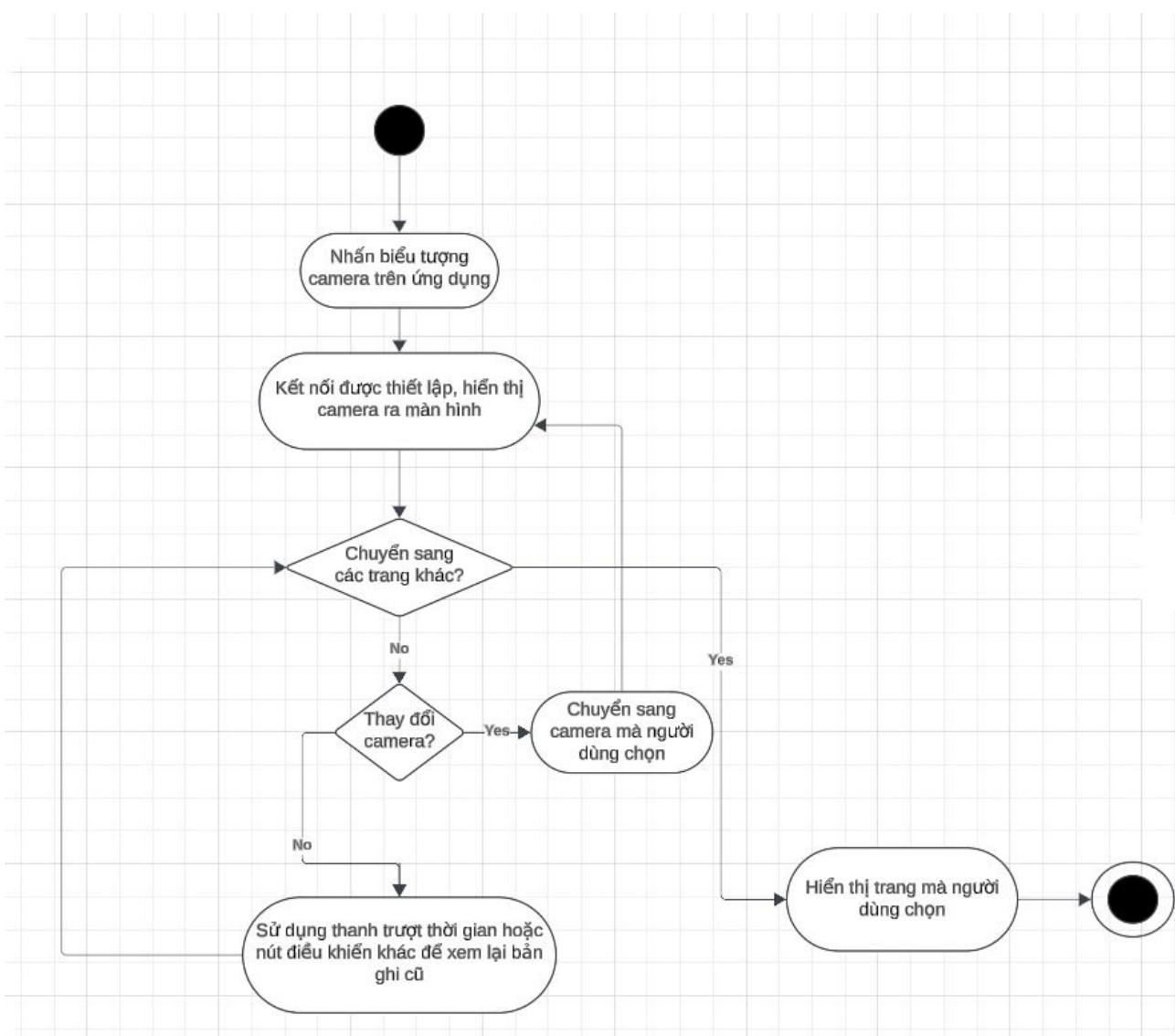


Figure 3: Activity Diagram cho chức năng quản lý tình trạng an ninh

## 4.4 Tạo các điều kiện tự động bật/tắt thiết bị

### 4.4.1 Thêm một điều kiện bật/tắt thiết bị

Use case ID	4
Tên use case	Tạo các điều kiện tự động bật/tắt thiết bị
Actor	Người dùng
Mô tả	Hệ thống cung cấp các cài đặt trên ứng dụng giúp cho người dùng có thể gián tiếp bật hoặc tắt các thiết bị IoT khi các thông số môi trường đạt đến ngưỡng so sánh mà người dùng cho rằng cần thay đổi trạng thái hiện tại của thiết bị hoạt động.
Tiền điều kiện	Hệ thống phải hoạt động bình thường, phải có các lệnh bật/tắt thiết bị tồn tại trong danh sách công việc và các thiết bị nhận lệnh điều khiển phải tồn tại.
Normal flow	<ol style="list-style-type: none"><li>Người dùng nhấn vào biểu tượng "Set Rule" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>Ứng dụng hiển thị danh sách các luật hiện tại và xuất hiện các lựa chọn để người dùng có thể thêm một luật mới. Các trường cần được điền để tạo ra một luật bao gồm loại thông số, ngưỡng so sánh, loại so sánh, thiết bị và hành vi của thiết bị.</li><li>Người dùng thao tác chọn loại thông số, ngưỡng so sánh, loại so sánh, hành vi và thiết bị được chọn để thay đổi trạng thái theo nhu cầu. Sau đó người dùng nhấn vào nút "thêm" để thêm một luật mới.</li><li>Luật mới được thêm và hiển thị ở cuối danh sách các luật dành cho các thiết bị hiện tại.</li><li>Hệ thống liên tục kiểm tra danh sách các luật được người dùng tạo ra khi nhận được cập nhật thông số môi trường từ các cảm biến và gửi request bật/tắt thiết bị ngay lập tức khi thực hiện các so sánh và phát hiện các thông số đạt đến ngưỡng so sánh yêu cầu.</li><li>Thiết bị nhận request từ sever và kích hoạt hoặc ngắt.</li><li>Công tắc thay đổi trạng thái hiện tại trên màn hình "home" theo đúng trạng thái hiện tại để báo cho người dùng biết một luật đã được thực thi.</li></ol>
Alternative Flow	Alternative 1: Tại bước 3 nếu người dùng không điền đầy đủ các trường được yêu cầu, hệ thống sẽ không tạo ra luật mới và gửi một thông báo yêu cầu người dùng điền các thông tin còn sót lại trên màn hình thiết bị. Alternative 2: Tại bước 4 nếu luật mới bị trùng hoặc có xung đột về loại so sánh với các luật đã tồn tại, hệ thống sẽ không tạo ra luật mới và gửi một thông báo về vấn đề này đến người dùng trên màn hình thiết bị.
Ngoại lệ	Mất kết nối mạng, các thiết bị nhận yêu cầu không tồn tại (mất nguồn điện hoặc mất kết nối với ứng dụng). Khi một trong các vấn đề này xảy ra hệ thống sẽ không thể bật/tắt thiết bị một cách tự động.

#### 4.4.2 Xóa một điều kiện bật/tắt thiết bị

Use case ID	5
Tên use case	Xóa một luật bật/tắt thiết bị ra khỏi danh sách
Actor	Người dùng
Mô tả	Người dùng có thể loại bỏ một luật kiểm tra dữ liệu để bật/tắt thiết bị nếu họ cảm thấy luật đó không còn cần thiết nữa.
Tiền điều kiện	Luật xuất hiện trong danh sách trên thiết bị người dùng.
Normal flow	<ol style="list-style-type: none"><li>Người dùng nhấn vào biểu tượng "Set Rule" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>Ứng dụng hiển thị danh sách các luật hiện tại trên màn hình thiết bị.</li><li>Người dùng bấm vào biểu tượng thùng rác trên thẻ ghi luật bật/tắt thiết bị hiện tại.</li><li>Luật bị xóa sẽ biến mất khỏi danh sách các luật trên màn hình.</li><li>Do đã bị xóa khỏi danh sách nên hệ thống sẽ không kiểm tra luật này để kích hoạt thiết bị tự động nữa.</li></ol>

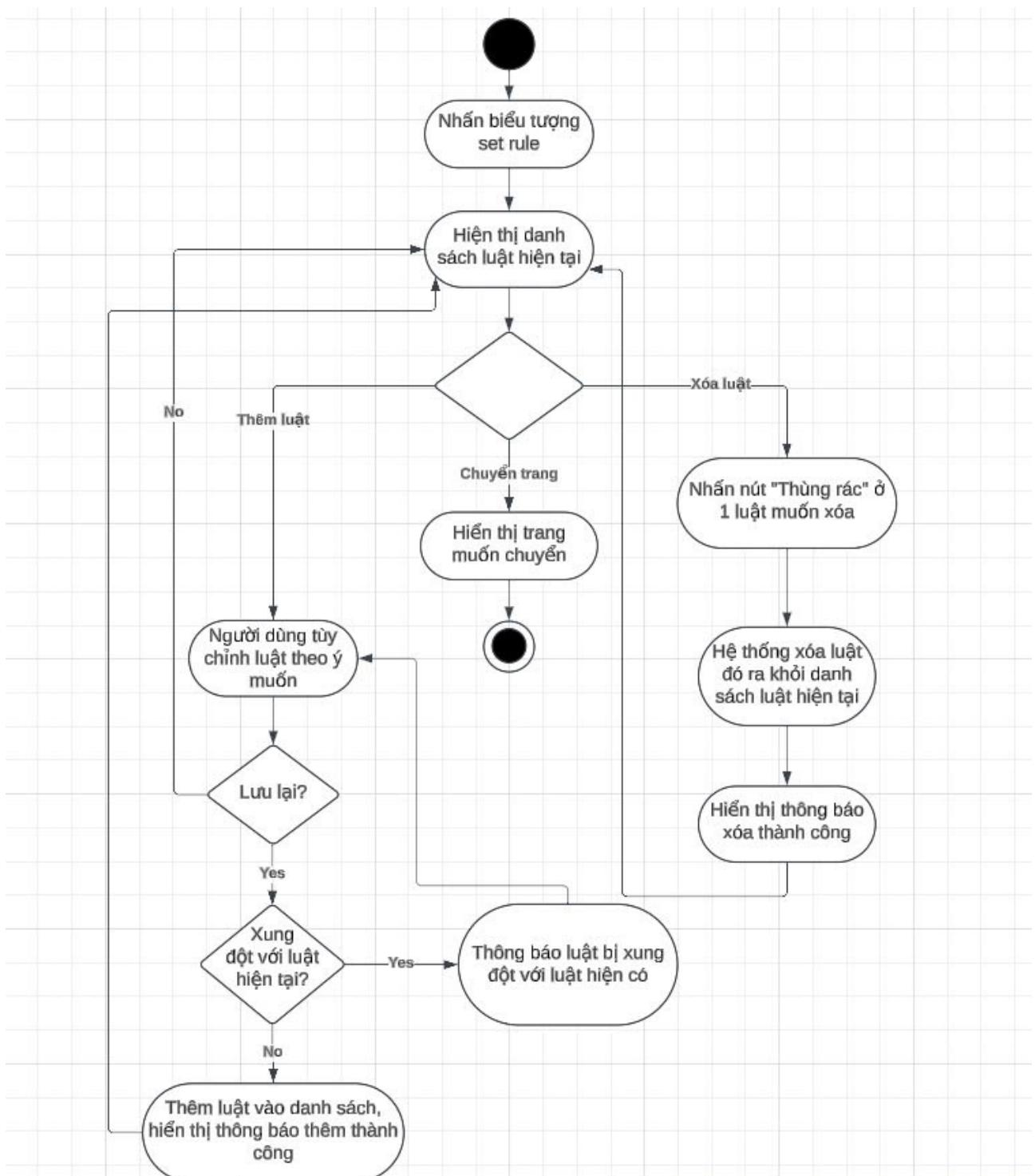


Figure 4: Activity Diagram của chức năng thêm luật



## 4.5 Lên lịch bật/tắt các thiết bị

### 4.5.1 Thêm một hẹn giờ bật/tắt thiết bị tự động

Use case ID	6
Tên use case	Lên lịch bật/tắt các thiết bị
Actor	Người dùng
Mô tả	Hệ thống cung cấp dịch vụ hẹn giờ trên ứng dụng giúp cho người dùng có thể gián tiếp bật hoặc tắt các thiết bị IoT tại thời điểm được cài đặt bởi người dùng.
Tiền điều kiện	Hệ thống phải hoạt động bình thường, phải có các lệnh bật/tắt thiết bị tồn tại trong danh sách hẹn giờ và các thiết bị nhận lệnh điều khiển phải tồn tại.
Normal flow	<ol style="list-style-type: none"><li>Người dùng nhấn vào biểu tượng "Scheduler" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>Ứng dụng hiển thị danh sách các lịch hẹn giờ hiện tại và có nút nhấn để người dùng đi đến cài đặt thêm một lịch hẹn giờ mới.</li><li>Người dùng nhấn vào biểu tượng dấu cộng trên màn hình để đi đến cài đặt lịch hẹn giờ kích hoạt thiết bị.</li><li>Người dùng thao tác chọn thời điểm thực hiện hành vi, loại hành vi (bật hoặc tắt) và thiết bị được chọn. Sau đó người dùng nhấn vào nút "thêm" để thêm một lịch hẹn giờ mới.</li><li>Lịch hẹn giờ mới sẽ được thêm vào và hiển thị ở cuối danh sách các lịch hẹn giờ của các thiết bị hiện tại.</li><li>Hệ thống liên tục kiểm tra các lịch hẹn giờ được người dùng tạo ra và sẽ gửi request thực hiện hành vi của thiết bị ngay lập tức khi đến thời gian kích hoạt bật/tắt thiết bị.</li><li>Thiết bị sẽ nhận request được gửi từ hệ thống ở bước 6 và thực hiện hành vi được yêu cầu.</li><li>Công tắc của thiết bị thay đổi trạng thái trên trang "home" theo đúng trạng thái hiện tại của thiết bị và gửi thông báo đã thực thi đến người dùng</li></ol>
Alternative Flow	<p>Alternative 1: Tại bước 4 nếu người dùng thêm một hẹn giờ xung đột với các hẹn giờ đã xuất hiện trong danh sách thì hệ thống sẽ không chấp nhận hẹn giờ đó và gửi thông báo xung đột đến người dùng.</p> <p>Alternative 2: Tại bước 4 nếu người dùng không muốn thiết lập hẹn giờ nữa thì họ có thể nhấn vào nút quay lại trên ứng dụng hoặc của thiết bị để hủy việc thêm hẹn giờ.</p>
Ngoại lệ	Hệ thống thời gian thực của hệ thống hoạt động không ổn định, kết nối đến sever thất bại, sever gửi request đến gateway không thành công hoặc thiết bị nhận yêu cầu không tồn tại.

#### 4.5.2 Xóa một hẹn giờ bật/tắt thiết bị tự động

Use case ID	6
Tên use case	Xóa hẹn giờ bật/tắt thiết bị
Actor	Người dùng
Mô tả	Người dùng có thể hủy bỏ hẹn giờ hiện tại hoặc xóa một hẹn giờ ra khỏi danh sách nếu lịch hẹn giờ đó không còn cần thiết nữa.
Tiền điều kiện	Phải có hẹn giờ bật/tắt thiết bị tồn tại trong danh sách các lịch hẹn giờ.
Normal flow	<ol style="list-style-type: none"><li>1. Người dùng nhấn vào biểu tượng "Scheduler" trên ứng dụng (nếu người dùng đang ở một trang khác).</li><li>2. Ứng dụng hiển thị danh sách các hẹn giờ hiện tại trên màn hình thiết bị.</li><li>3. Người dùng nhấn vào biểu tượng thùng rác trên thẻ ghi hẹn giờ hiện tại.</li><li>4. Hẹn giờ bị xóa sẽ biến mất khỏi danh sách các hẹn giờ trên màn hình thiết bị.</li><li>5. Hệ thống sẽ không kiểm tra hẹn giờ đã bị xóa để kích hoạt thiết bị tự động nữa.</li></ol>

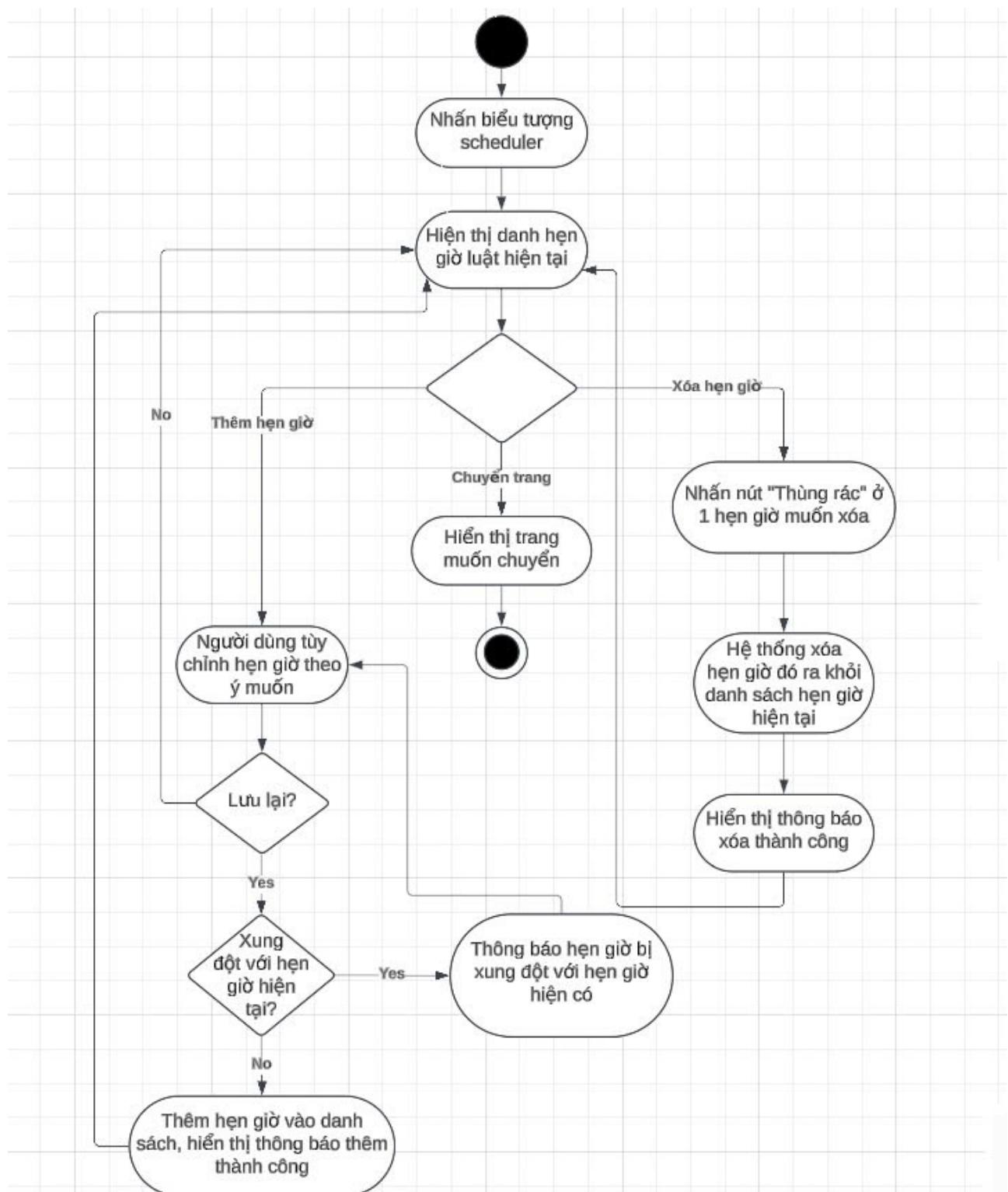


Figure 5: Activity Diagram cho chức năng hẹn giờ

## 5 Class diagram

### 5.1 Model View Controller

Dự án được cấu trúc thành 4 khối chính (3+1). Để hiểu rõ hơn về cách các thành phần này tương tác với nhau, nhóm sẽ giới thiệu mô hình MVC được áp dụng trong dự án.

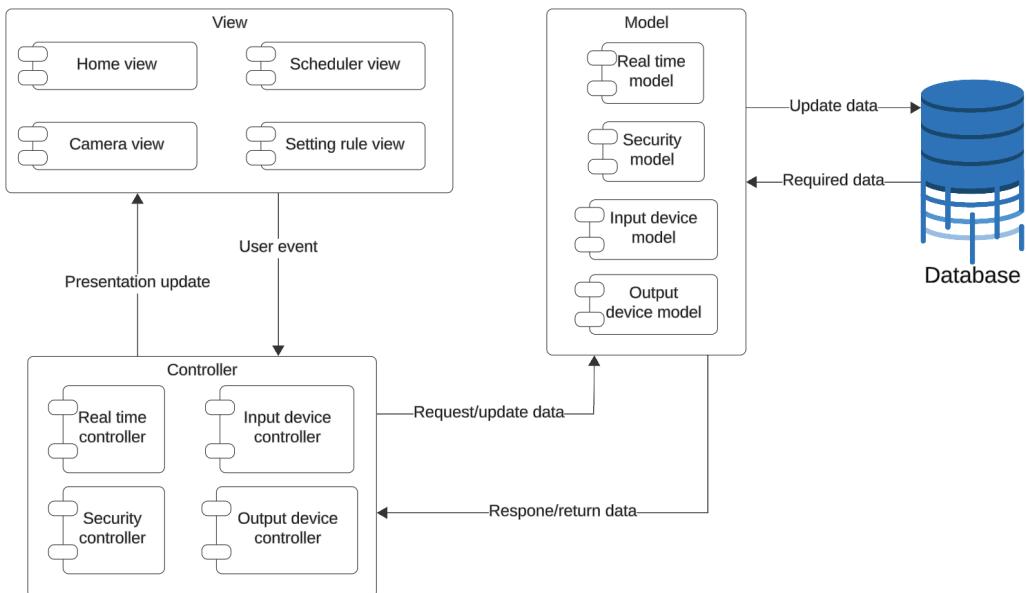
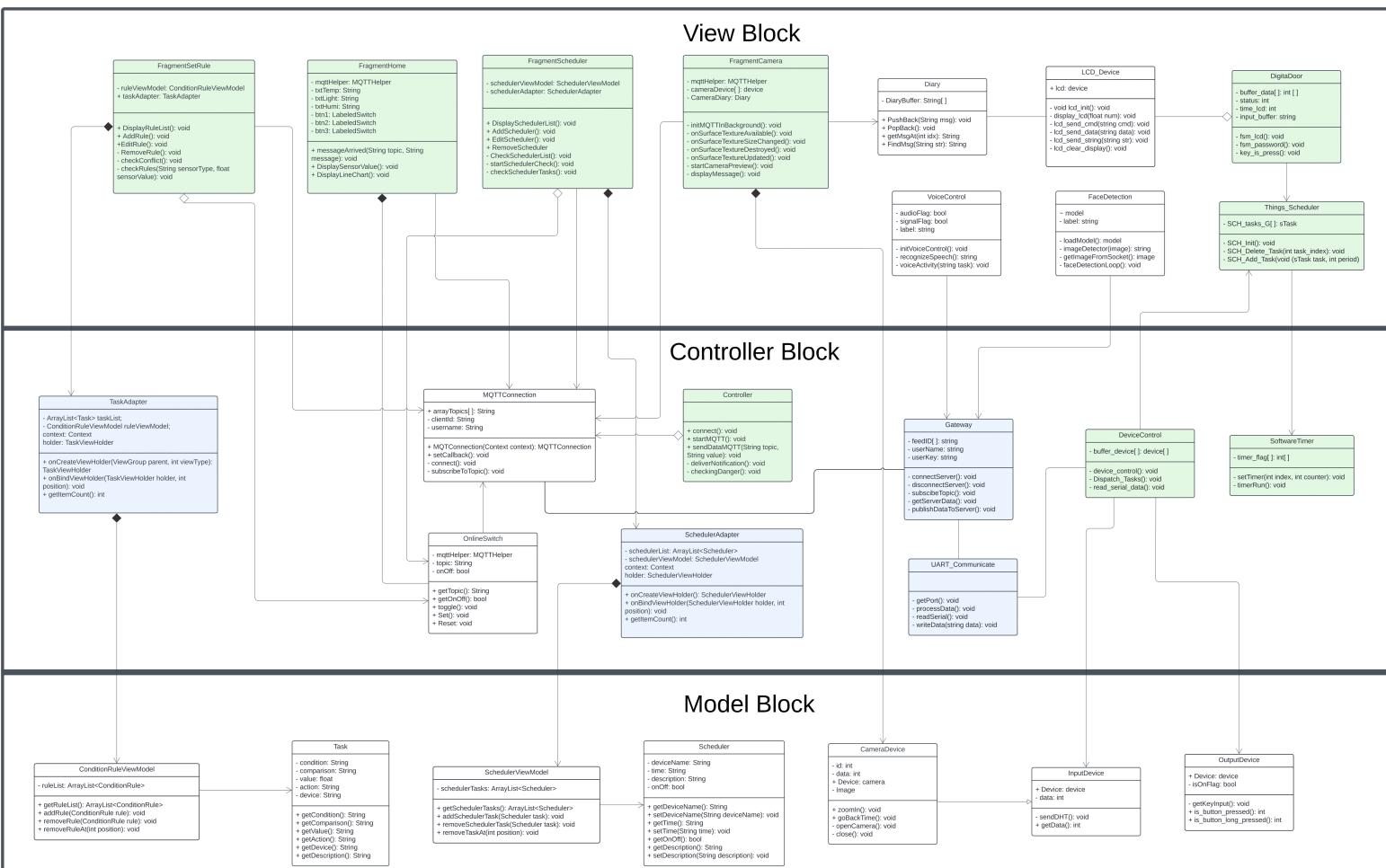


Figure 6: Kiến trúc MVC của hệ thống

- Khối View chịu trách nhiệm hiển thị thông tin cho người dùng và nhận tương tác từ người dùng. Nó là giao diện mà người dùng nhìn thấy và tương tác trực tiếp. Các thành phần thuộc khối View đại diện cho các màn hình giao diện giúp người dùng có thể tương tác.
- Khối Model đại diện cho dữ liệu của ứng dụng và các quy tắc nghiệp vụ, công việc có liên quan đến dữ liệu đó. Nó chứa các logic xử lý dữ liệu, truy cập dữ liệu từ cơ sở dữ liệu và cung cấp dữ liệu cho View. Các thành phần thuộc khối Model đại diện cho các phần dữ liệu và logic liên quan đến thời gian thực, bảo mật, thiết bị đầu vào và thiết bị đầu ra...
- Khối Controller đóng vai trò trung gian giữa View và Model. Nó nhận các sự kiện từ View, xử lý chúng, cập nhật Model và sau đó cập nhật View để phản ánh những thay đổi. Những thành phần thuộc khối này chịu trách nhiệm xử lý các sự kiện liên quan đến từng khía cạnh của hệ thống.

### 5.2 Class diagram

Class Diagram mô tả cấu trúc của một hệ thống bằng cách hiển thị các lớp, thuộc tính của chúng và các mối quan hệ giữa chúng. Nó cung cấp một cái nhìn tổng quan về các thành phần tạo nên hệ thống và cách chúng tương tác với nhau.



Class Diagram được trình bày dựa trên kiến trúc hệ thống MVC.

A → B: Mối quan hệ class A sử dụng class B

A ——— B: Mối quan hệ song phương, các phần tử của 2 class liên hệ mật thiết với nhau

A ♦—— B: Mối quan hệ phụ thuộc, khi class A mất thì class B cũng sẽ mất

A ◇—— B: Mối quan hệ độc lập, khi class A mất thì class B vẫn tồn tại

Figure 7: Các loại mối quan hệ giữa hai class

## 6 Devices

### 6.1 Các loại cảm biến môi trường

#### 6.1.1 Cảm biến nhiệt độ, độ ẩm DHT11

##### 1. Cấu tạo

- Cảm biến DHT11 bao gồm một phần tử cảm biến độ ẩm điện dung và một điện trở nhiệt để cảm nhận nhiệt độ. Tụ điện cảm biến độ ẩm có hai điện cực với chất nền giữ ẩm làm chất điện môi giữa chúng. Thay đổi giá trị điện dung xảy ra với sự thay đổi của các mức độ ẩm. IC đo, xử lý các giá trị điện trở đã thay đổi này và chuyển chúng thành dạng kỹ thuật số.
- Để đo nhiệt độ, cảm biến này sử dụng một nhiệt điện trở có hệ số nhiệt độ âm, làm giảm giá trị điện trở của nó khi nhiệt độ tăng. Để có được giá trị điện trở lớn hơn ngay cả đối với sự thay đổi nhỏ nhất của nhiệt độ, cảm biến này thường được làm bằng gốm bán dẫn hoặc polymer.

##### 2. Tính năng:

- Cảm biến có thể đo nhiệt độ từ 0 ° C đến 50 ° C và độ ẩm từ 20% đến 90% với độ chính xác ± 1 ° C và ± 1%.
- DHT11 là cảm biến rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1 wire (giao tiếp digital 1 dây truyền dữ liệu duy nhất). Bộ tiền xử lý tín hiệu tích hợp trong cảm biến giúp bạn có được dữ liệu chính xác mà không phải qua bất kỳ tính toán nào.

#### 6.1.2 Cảm biến ánh sáng

##### 1. Cấu tạo:

- Cấu tạo của cảm biến ánh sáng khá đơn giản sẽ bao gồm: mắt cảm biến/ đầu dò ánh sáng, dây dẫn, chíp xử lý biến đổi thành tín hiệu điện. Các linh kiện bên trong được bảo vệ bằng lớp vỏ ngoài bảo vệ sensor ánh sáng khỏi va đập hoặc tác động lực từ dị vật ảnh hưởng.

##### 2. Tính năng:

- Cảm biến Photoresistors (LDR) là loại cảm biến được sử dụng nhiều nhất trong các thiết bị cảm biến. Nó chính là chất cảm quang, hay còn được gọi là điện trở phụ thuộc ánh sáng (LDR). Chất cảm quang này có tác dụng kiểm tra xem đèn bật hay tắt. Và nó so sánh mức độ ánh sáng của môi trường theo tính chất tương đối trong suốt một ngày. Chất phát quang này được làm từ một vật liệu bán dẫn có điện trở cao. Chất bán dẫn này rất nhạy với ánh sáng, có thể nhìn thấy ánh sáng gần với hồng ngoại.

### 6.1.3 Cảm biến khí

#### 1. Cấu tạo

- Lớp bảo vệ: Được bao phủ bởi hai lớp lưới thép không gỉ, gọi là *anti-explosion network*, nhằm ngăn chặn nguy cơ nổ từ bộ phận làm nóng bên trong. Lớp lưới này cũng bảo vệ cảm biến, lọc bụi, và chỉ cho phép các phần tử khí đi qua.
- Cấu trúc bên trong:
  - Gồm sáu chân, có hình dạng ngôi sao:
  - Hai chân H: Làm nóng cảm biến, kết nối bằng dây Niken-Crom.
  - Bốn chân A và B: Truyền tín hiệu, kết nối bằng dây bạch kim, phản ánh sự thay đổi dòng điện trong cảm biến.
  - Hình ống trung tâm được làm bằng gốm ( $\text{Al}_2\text{O}_3$ ), phủ Thiếc Dioxide ( $\text{SnO}_2$ ), một vật liệu nhạy cảm với khí dễ cháy.
- Lớp gốm cải thiện hiệu quả làm nóng, đảm bảo cảm biến đạt nhiệt độ hoạt động ổn định để phát hiện khí dễ cháy.

#### 2. Tính năng

- Cảm biến khí Gas MQ2 là một trong những cảm biến được sử dụng rộng rãi nhất trong các dòng cảm biến MQ. Nó là một cảm biến MOS (Metal Oxide Semiconductor). Cảm biến oxit kim loại hay còn được gọi là (Điện trở hóa trị) vì cảm biến dựa trên sự thay đổi điện trở của cảm biến khi tiếp xúc với khí.
- Cảm biến khí gas arduino hoạt động trên 5V DC và tiêu thụ khoảng 800mW. Nó có thể phát hiện nồng độ LPG, Khói, Rượu, Propane, Hydrogen, Methane và Carbon Monoxide từ 200 đến 10000 ppm.

### 6.1.4 Cảm biến lửa

#### 1. Cấu tạo:

- Cảm biến ngọn lửa là thiết bị cảm biến được sử dụng để phát hiện sự hiện diện của ngọn lửa và các yếu tố liên quan đến sự cháy nổ như nhiệt độ cao, tia hồng ngoại, tia cực tím, v.v...
- Cảm biến ngọn lửa quang điện: sử dụng ống quang điện để phát hiện ánh sáng do ngọn lửa phát ra. Cảm biến ngọn lửa quang điện có thể được chia thành hai loại nhỏ:
  - Cảm biến ngọn lửa hồng ngoại: Phát hiện bức xạ hồng ngoại do ngọn lửa phát ra.
  - Cảm biến ngọn lửa cực tím: Phát hiện bức xạ cực tím do ngọn lửa phát ra.

#### 2. Tính năng:

Cảm biến ngọn lửa sử dụng các bộ phận cảm biến để thu thập thông tin về môi trường xung quanh như ánh sáng, nhiệt độ, thành phần khí hoặc mức độ ion hóa. Dữ liệu thu thập được phân tích để xác định xem có dấu hiệu của ngọn lửa hay không. Nếu phát hiện thấy dấu hiệu của ngọn lửa, cảm biến ngọn lửa sẽ kích hoạt tín hiệu báo động. Tín hiệu này có thể được truyền đến hệ thống báo cháy hoặc trực tiếp kích hoạt còi báo động, đèn nhấp nháy hoặc các thiết bị chữa cháy khác.

## 6.2 Thiết bị điều khiển trong gia đình

### 1. Quạt:

- Quạt thông gió sử dụng trong gia đình giúp điều hòa không khí, cân bằng không khí giữa bên trong và bên ngoài phòng.
- Ngoài ra khi phát hiện khói lạ hoặc có cháy, quạt sẽ tự động bật để đưa các khí độc ra bên ngoài.

**2. Còi báo động:** Khi phát hiện ra khói lạ hoặc có cháy, cùng với quạt, còi báo động sẽ được bật để mọi người sơ tán khẩn cấp.

### 3. Đèn:

- Đèn là thiết bị không thể thiếu của mọi gia đình, nhất là vào ban đêm.
- Đèn có thể được kích hoạt bằng nút nhấn vật lý hoặc nút nhấn trên ứng dụng điện thoại.

### 4. Servo:

- Ứng dụng của servo sẽ là thiết bị để đóng mở cửa tự động.
- Khi nhập mật khẩu đúng hoặc nhận diện được khuôn mặt chủ nhà, cửa sẽ tự động mở. Ở đây servo quay 180 độ thì có nghĩa là đang đóng và mở cửa.

### 5. Màn hình LCD:

- Để người dùng trong gia đình có thể tiếp nhận thông tin nhanh chóng mà không cần xem ứng dụng điện thoại, màn hình LCD được sử dụng.
- Màn hình mỗi 10 giây luân phiên hiển thị nhiệt độ, độ ẩm, ánh sáng ra màn hình.
- Ngoài ra nó còn dùng để hiển thị mật khẩu khi mở cửa bằng mật khẩu và thông báo cho người dùng biết mật khẩu có chính xác không.
- Cú pháp nhập mật khẩu sẽ như sau:
  - Nhấn \* để vào chế độ nhập mật khẩu.
  - Nhập chuỗi mật khẩu 1234.
  - Nhấn # để xác nhận nhập hoàn tất.
  - Khi nhập sai, người dùng được yêu cầu nhập lại. Khi nhập đúng cửa sẽ mở và chuyển sang chế độ hiển thị môi trường.

### 6. Relay:

- Sử dụng nguồn 5V từ mạch ổn áp để bật tắt các thiết bị còi và quạt.
- Được điều khiển trực tiếp từ tín hiệu esp32.

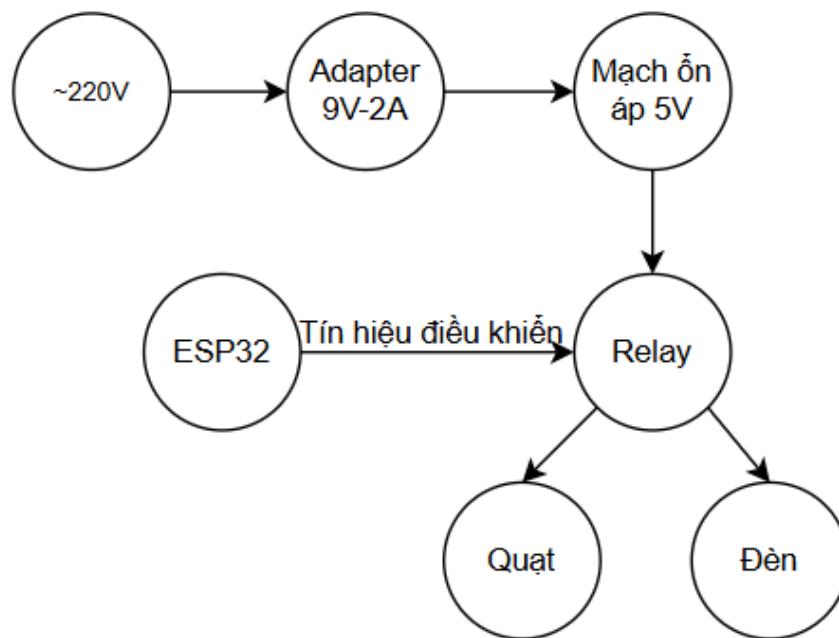


Figure 8: Sơ đồ mạch điều khiển các thiết bị.

### 6.3 Mạch ổn áp sử dụng LM2596

LM2596 là một trong những IC phổ biến trong thiết kế các bộ chuyển đổi điện áp DC-DC, đặc biệt là bộ hạ áp (buck converter). Với khả năng điều chỉnh điện áp đầu ra linh hoạt và hiệu suất cao, LM2596 là lựa chọn lý tưởng cho các ứng dụng yêu cầu nguồn cung ổn định.

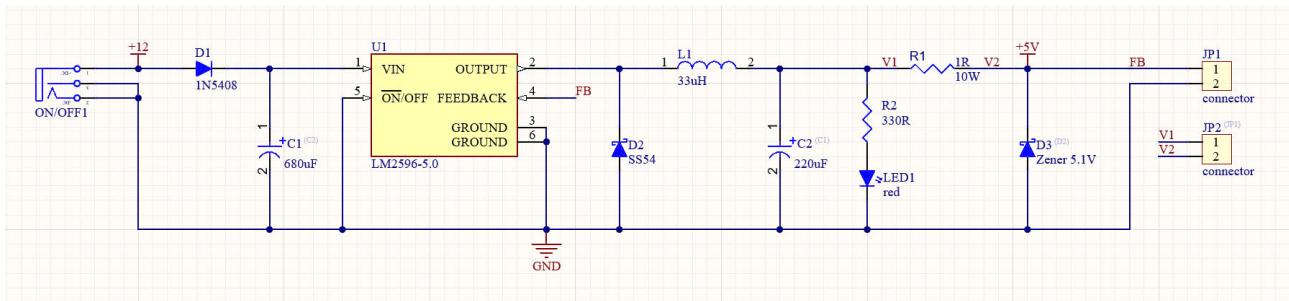


Figure 9: Sơ đồ mạch nguồn sử dụng LM2596.

### 6.4 Vi điều khiển ESP32 và sơ đồ kết nối

ESP32 là vi điều khiển tiết kiệm chi phí và năng lượng thấp, tích hợp Bluetooth và Wi-Fi chế độ kép, mang lại tính linh hoạt, mạnh mẽ, và đáng tin cậy cho nhiều ứng dụng. Là phiên bản kế thừa của ESP8266, ESP32 cung cấp hiệu suất và tính năng vượt trội, được phát triển bởi Espressif Systems, và được ứng dụng rộng rãi trong IoT, robot, và tự động hóa. Thiết kế tiết kiệm năng lượng với khả năng quản lý chế độ ngủ giúp tăng tuổi thọ pin, phù hợp với các thiết bị chạy bằng pin.

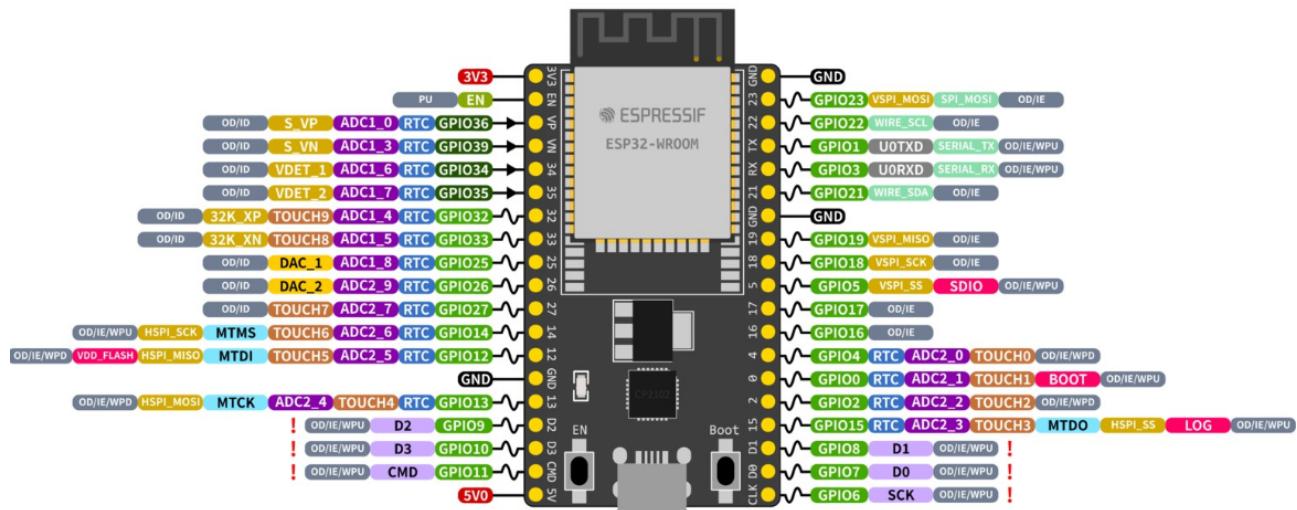


Figure 10: Module Esp32.

#### 6.4.1 Sơ đồ kết nối module Esp32:

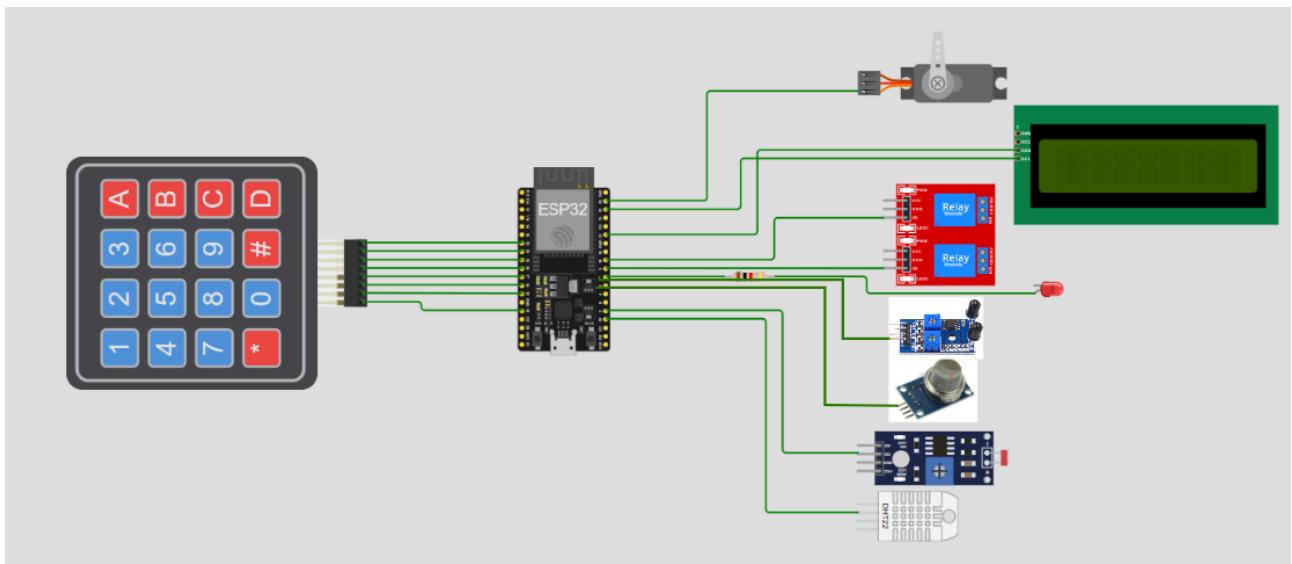


Figure 11: Sơ đồ kết nối module esp32.

Tên chân (Pin)	Chức năng	Số chân trên ESP32
DHTPIN	Cảm biến nhiệt độ và độ ẩm	15
LIGHTPIN	Cảm biến ánh sáng	2
GAS_PIN	Cảm biến khí gas	4
FIRE_PIN	Cảm biến lửa	16
LED_PIN	Đèn LED	17
FAN_PIN	Quạt	5
MOTOR_PIN	Động cơ	23
BUZZER_PIN	Chuông báo	18
LCD_PIN	Màn hình LCD	21(SDA),22(SCL)
Phím ma trận	Nhập mật khẩu	12,13,14,25,26,27,32,33

Table 1: Bảng mô tả các chân của ESP32 trong dự án.

## 6.5 Kiến trúc hệ thống Things:

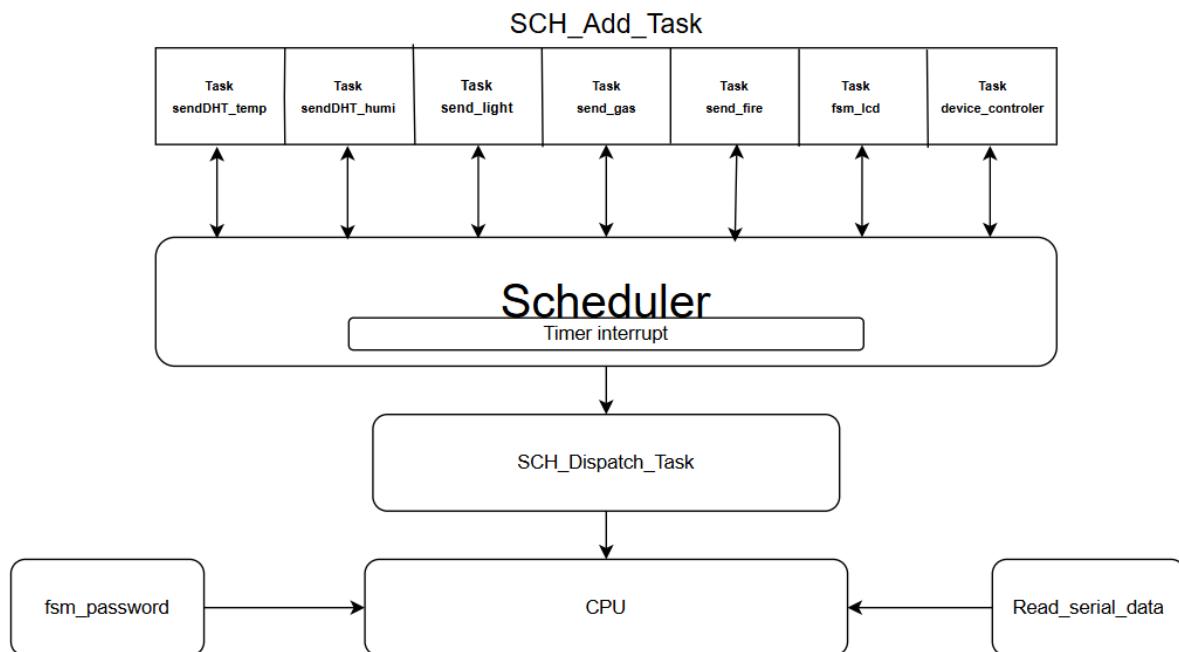


Figure 12: Kiến trúc hệ thống Things.

## Kiến trúc hệ thống Things:

- 1. Thêm Tác Vụ
  - sendDHT\_temp
  - sendDHT\_humi
  - send\_light
  - send\_gas
  - send\_fire

- fsm\_lcd

- device\_controller

- 2. Scheduler

- Bộ lập lịch hoạt động dựa trên nhân là **Timer interrupt**, đảm bảo các tác vụ được thực hiện theo khoảng thời gian định trước.

- 3. Dispatch Task

- Bộ lập lịch gửi các tác vụ đến hàm SCH\_Dispatch\_Task, chịu trách nhiệm phân tích và chuẩn bị các tác vụ để chuyển đến CPU.

- 4. CPU(Loop)

- CPU là thành phần xử lý chính trong hệ thống, chịu trách nhiệm thực hiện các lệnh từ các tác vụ đã được phân tích.

- 5. Các Thành Phần Bổ Sung CPU tương tác với các quy trình bổ sung như:

- fsm\_password: Quản lý mật khẩu.

- Read\_serial\_data: Đọc dữ liệu từ cổng nối tiếp.

### Hàm sendDHT\_temp:

- Đọc nhiệt độ từ cảm biến DHT.
- Nếu dữ liệu không hợp lệ, gửi thông báo lỗi.
- Nếu hợp lệ, lưu nhiệt độ vào buffer\_data và gửi qua Serial: !sensor1:T:%.1f#.

### Hàm sendDHT\_humi:

- Đọc độ ẩm từ cảm biến DHT.
- Nếu dữ liệu không hợp lệ, gửi thông báo lỗi.
- Nếu hợp lệ, lưu độ ẩm vào buffer\_data và gửi qua Serial: !sensor2:H:%.1f#.

### Hàm send\_light:

- Đọc giá trị ánh sáng từ cảm biến quang.
- Chuyển đổi giá trị thành phần trăm và lưu vào buffer\_data.
- Gửi dữ liệu qua Serial: !sensor3:L:%.1f#.

### Hàm send\_fire:

- Đọc trạng thái cảm biến lửa.
- Gửi trạng thái cảm biến qua Serial: !sensor4:F:%d#.

### Hàm send\_gas:



- Đọc trạng thái cảm biến khí gas.
- Gửi trạng thái cảm biến qua Serial: !sensor5:G:%d#.

### Hàm read\_serial\_data

- Đọc dữ liệu từ Serial.
- Kiểm tra định dạng dữ liệu (!B:<index>:<data>#).
- Nếu hợp lệ, gán data vào buffer\_device[index - 1].
- Nếu không hợp lệ, gửi thông báo lỗi.

**Hàm device\_control:** Nhận dữ liệu từ buffer\_device thực hiện bật tắt các thiết bị.

### Hàm fsm\_lcd:

- **Chức năng:** Hiển thị thông tin từ các cảm biến (nhiệt độ, độ ẩm, ánh sáng) trên LCD.
- **Trạng thái:**
  - **INIT:**
    - \* Khởi tạo LCD.
    - \* Chuyển trạng thái sang **NHIET\_DO**.
  - **NHIET\_DO, DO\_AM, ANH\_SANG:**
    - \* Hiển thị lần lượt thông tin nhiệt độ, độ ẩm, ánh sáng.
    - \* Sau khoảng thời gian **timelcd**, chuyển sang trạng thái tiếp theo.
    - \* Nếu phát hiện phím nhấn (**flag\_key**), chuyển sang **INIT\_PASSWORD**.

### Hàm fsm\_password:

- **Chức năng:** Quản lý nhập mật khẩu và kiểm tra đúng/sai.
- **Trạng thái:**
  - **INIT\_PASSWORD:**
    - \* Hiển thị thông báo: "Nhập mật khẩu".
  - **READ\_PASSWORD:**
    - \* Nhận phím số.
    - \* Nếu nhấn #:
      - So sánh với mật khẩu đúng.
      - Nếu đúng: Hiển thị "Mật khẩu đúng", bật MOTOR.
      - Nếu sai: Hiển thị "Sai! Thử lại", yêu cầu nhập lại.
    - \* Nếu nhấn \*: Xóa nhập liệu và quay lại **INIT\_PASSWORD**.
    - \* Hết thời gian (**timer\_flag[0]**): Tắt MOTOR, chuyển trạng thái về **INIT**.

## 7 Hiện thực hệ thống

### 7.1 Giao tiếp giữa khối Things với GateWay

Trong một hệ thống IOT, việc đảm bảo giao tiếp giữa các khối với nhau là vô cùng quan trọng. Ở hệ thống này giao thức UART được sử dụng để giao tiếp giữa khối thiết bị Things và GateWat.

UART là giao thức truyền thông nối tiếp không đồng bộ được sử dụng rộng rãi trong các hệ thống nhúng để trao đổi dữ liệu giữa hai thiết bị. Trong giao tiếp UART, dữ liệu được truyền từng bit qua một cặp dây dẫn.

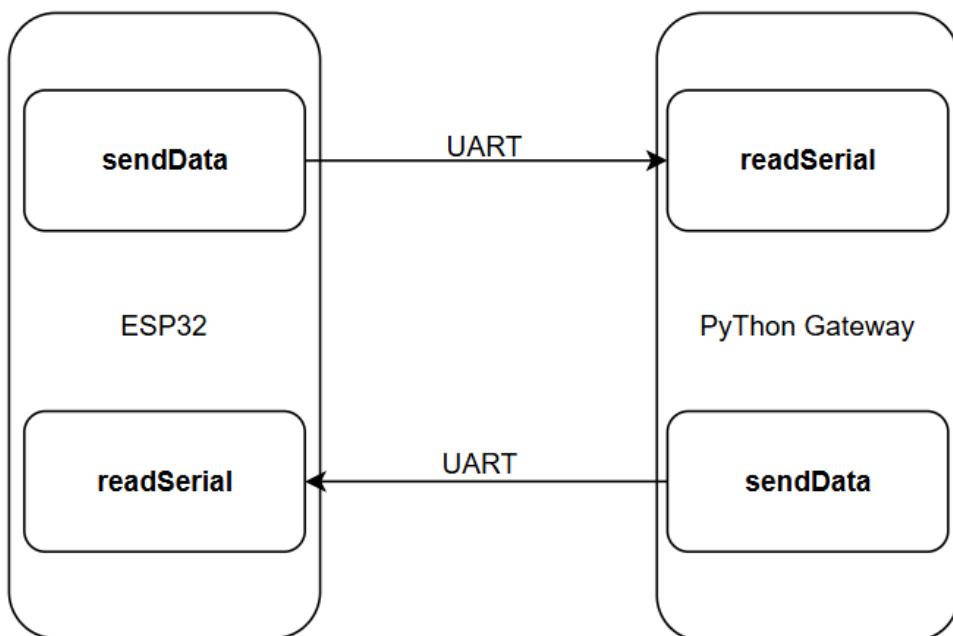


Figure 13: Giao tiếp giữa ESP32 và GateWay.

**ESP32 có hai chức năng chính:**

- **sendData:** ESP32 thu thập dữ liệu quan trắc từ môi trường, gửi dữ liệu đến Python Gateway qua giao tiếp UART.
- **readSerial:** Chức năng đọc dữ liệu được gửi từ Python Gateway qua UART, sau đó xử lý dữ liệu và lưu vào buffer.

**Python Gateway cũng có hai chức năng chính:**

- **sendData:** GateWay lấy dữ liệu (thao tác điều khiển từ người dùng từ ứng dụng) sau đó gửi dữ liệu tới ESP32 qua giao tiếp UART.
- **readSerial:** Đọc dữ liệu từ ESP32 gửi tới, xử lý dữ liệu sau đó gửi lên Server.

**Giao tiếp:**



- Giao thức **UART** (Universal Asynchronous Receiver-Transmitter) được sử dụng để trao đổi dữ liệu hai chiều giữa ESP32 và Python Gateway.
- Các mũi tên biểu thị luồng dữ liệu được gửi và nhận giữa hai thiết bị.
- Dữ liệu được gửi là chuỗi String có định dạng như sau: "!(Thiết bị gửi):" + (ID thiết bị gửi) + ":" + data + "#".

## 7.2 Kết nối gateway với Server IO

### 7.2.1 Giao thức MQTT

Giao thức MQTT (Message Queuing Telemetry Transport) là một giao thức nhắn tin nhẹ theo mô hình publish/subscribe, được thiết kế để kết nối các thiết bị với các máy chủ trung tâm trong các hệ thống Internet of Things, phù hợp với tài nguyên hạn chế và băng thông mạng thấp.

MQTT là một trong những giao thức được sử dụng rộng rãi nhất trong các ứng dụng IoT, cho phép các thiết bị gửi và nhận thông tin thông qua một broker trung gian. Ưu điểm của MQTT là khả năng mở rộng tốt, độ tin cậy cao và tiêu thụ năng lượng thấp, làm cho nó trở thành lựa chọn lý tưởng cho việc kết nối và giao tiếp giữa các thiết bị IoT.

### 7.2.2 Hiện thực kết nối gateway bằng giao thức MQTT

IoT gateway đóng một vai trò quan trọng, là cầu nối giữa các thiết bị IoT và server. Trong dự án này, gateway được hiện thực bằng ngôn ngữ lập trình python. Nhóm sử dụng gateway để thực hiện các tác vụ sau:

1. Nhận dữ liệu từ ESP32 qua cổng COM
2. Gửi data nhận được lên server trong quá trình giao tiếp với các thiết bị phần cứng.
3. Nhận data từ server để kích hoạt các thiết bị phần cứng khi cần thiết
4. Nhận diện khuôn mặt và giọng nói, phân tích dữ liệu và đưa lên server để thực hiện các tác vụ kế tiếp

### 7.2.3 Python trên Gateway

Python sẽ mở cổng COM, đọc dữ liệu được gửi từ ESP32, và xử lý dữ liệu này trước khi gửi lên server. Để hiện thực Gateway bằng python, nhóm sử dụng hai thư viện chính:

- **pyserial:** Thư viện pyserial trong Python được sử dụng để giao tiếp với cổng COM, tạo liên kết giao tiếp giữa gateway với các thiết bị trong khối things.
- **Adafruit\_IO:** Nền tảng IoT dùng để thu thập, xử lý và hiển thị dữ liệu cảm biến, sử dụng giao thức MQTT để giao tiếp giữa các thiết bị. MQTTClient là một lớp trong thư viện Adafruit IO dùng để kết nối với server MQTT của Adafruit IO, gửi và nhận dữ liệu đến/từ các Feeds.

- **asyncio** sử dụng code bất đồng bộ, chia các nhiệm vụ xử lý của mô hình AI thành các luồng riêng biệt thực thi song song, cho phép xử lý đồng thời nhiều tác vụ cùng một lúc.
- **threading** thư viện hỗ trợ thực thi đa luồng.

### 7.3 Giao tiếp giữa ứng dụng với Server IO

Nhóm định nghĩa lớp MQTTHelper để hỗ trợ và đơn giản hóa việc ứng dụng giao tiếp với Server IO của Adafruit sử dụng giao thức MQTT. Các chức năng của ứng dụng giúp người dùng gián tiếp tương tác với các cảm biến và thiết bị IoT sẽ truy cập vào lớp này để tiến hành giao tiếp, thao tác, lấy dữ liệu về hoặc cập nhật dữ liệu mới lên server.

Lớp MQTTHelper sẽ chịu trách nhiệm thiết lập và cấu hình các thông số kết nối với Server IO của Adafruit, đăng ký các topic dữ liệu, nhận và xử lý dữ liệu từ server cũng như thiết lập các hàm call back để nhận các thông báo về kết nối, mất kết nối và tin nhắn đến.

Dể có thể thể hiện thực lớp này bằng ngôn ngữ java, cần import thư viện **Paho MQTT Android** và **org.eclipse.paho.client.mqttv3** khi hiện thực trên hệ điều hành Android. Dưới đây là các thuộc tính và phương thức của lớp MQTTHelper được nhóm hiện thực:

```
1 public class MQTTHelper{
2     public MqttAndroidClient mqttAndroidClient;
3     public final String[] arrayTopics;
4     final String clientId;
5     final String username;
6     final String password;
7     final String serverUri;
8     public MQTTHelper(Context context){
9         mqttAndroidClient = new MqttAndroidClient(context, serverUri,
10                                         clientId);
11         mqttAndroidClient.setCallback(new MqttCallbackExtended() {
12             @Override
13             public void connectComplete(boolean b, String s);
14             @Override
15             public void connectionLost(Throwable throwable);
16             @Override
17             public void messageArrived(String topic,
18                                         MqttMessage mqttMessage);
19             @Override
20             public void deliveryComplete(IMqttDeliveryToken
21                                         iMqttDeliveryToken);
22         });
23         //TODO: connect to server
24     }
25     public void setCallback(MqttCallbackExtended callback);
26     private void connect();
27     private void subscribeToTopic();
28 }
```

- mqttAndroidClient: Đây là đối tượng client MQTT, chịu trách nhiệm quản lý kết nối đến MQTT broker, gửi và nhận tin nhắn. Có thể truy cập thuộc tính này từ bên ngoài lớp MQTTHelper.

- arrayTopics: Một mảng lưu trữ danh sách các topic (chủ đề) mà client MQTT sẽ đăng ký (subscribe) để nhận tin nhắn. Có thể được truy cập từ bên ngoài lớp và giá trị của các phần tử trong mảng chỉ được gán một lần trong quá trình khởi tạo đối tượng.
- clientId: ID của client MQTT. Mỗi client kết nối đến broker cần có một và chỉ một ID duy nhất.
- username: Tên người dùng để xác thực với MQTT broker.
- password: Mật khẩu để xác thực với MQTT broker. Trong hiện thực giao thức MQTT với Adafruit, nhóm định nghĩa đây là **Active Key** và bảo mật thuộc tính này để đảm bảo an ninh không bị lộ thông tin server ra ngoài.
- serverUri: URI của MQTT broker.
- MQTTHelper(Context context): Đây là hàm khởi tạo (constructor) của lớp MQTTHelper.
- void setCallback(MqttCallbackExtended callback): Thiết lập hoặc thay đổi callback cho mqttAndroidClient. Callback này xử lý các sự kiện như kết nối thành công, mất kết nối, nhận tin nhắn và gửi tin nhắn thành công.
- void connect(): Phương thức này chịu trách nhiệm thiết lập kết nối đến MQTT broker. Chỉ có thể được gọi từ bên trong lớp MQTTHelper.
- void subscribeToTopic(): Phương thức này đăng ký client MQTT vào các topic được lưu trữ trong arrayTopics. Chỉ có thể được gọi từ bên trong lớp MQTTHelper.

Mã nguồn về hiện thực các phương thức của lớp: [Click here](#)

## 7.4 Tích hợp mô hình AI

Dự án sử dụng **Google Teachable Machine** làm công cụ hỗ trợ cho việc sử dụng mô hình AI - công cụ hỗ trợ xây dựng mô hình học máy mà không cần có kiến thức sâu về lập trình hay AI. Nhóm sử dụng nhận diện hình ảnh và âm thanh cho dự án này.

### 7.4.1 Nhận diện khuôn mặt

Sau khi huấn luyện mô hình nhận diện hình ảnh trên Teachable Machine, ta tải về model Tensorflow để sử dụng cho dự án. Camera sẽ lấy ảnh và so sánh với model đã tạo, dữ liệu sẽ được đẩy lên adafruit để tiếp tục xử lý cho phần tiếp theo.

#### Các thư viện sử dụng

- **Keras** với hàm load\_model được sử dụng để tải một mô hình học máy đã được training là file .h5 trong Keras.
- **opencv-python** là một thư viện xử lý ảnh và video dùng để nhận diện khuôn mặt, xử lý video... Tác dụng chính là thu ảnh từ webcam rồi chuyển đổi, xử lý và phân tích ảnh, sau đó hiển thị hoặc lưu trữ ảnh/video.
- **Numpy** là thư viện xử lý các mảng số học (arrays), chuyển đổi ảnh thành mảng để đưa vào mô hình, xử lý dữ liệu đầu vào và đầu ra cho các mô hình học sâu. Numpy làm trung gian để xử lý dữ liệu ảnh (chuyển đổi ảnh từ OpenCV thành mảng số



học để đưa vào mô hình). cv2 lấy ảnh từ webcam, xử lý ảnh trước khi đưa vào mô hình. load\_model (keras) tải mô hình học sâu đã được huấn luyện, dự đoán đối tượng hoặc phân loại dựa trên dữ liệu từ cv2 và numpy.

- **socket** tạo môi ống dẫn pipe liên kết hỗ trợ truyền tải dữ liệu hình ảnh từ camera giúp AI nhận được ảnh và nhận diện khuôn mặt ngừng người có hành vi tiếp cận.

#### 7.4.2 Nhận diện giọng nói

Dự án sử dụng Google Teachable Machine để tạo mô hình nhận diện audio, sau đó dữ liệu sẽ được upload lên Google Cloud, nhóm sử dụng link dẫn tới mô hình này, kết nối sử dụng giao tiếp giữa Websocket và MQTT để nhận diện âm thanh, so sánh với mô hình đã được upload rồi trả về kết quả đến Python.

##### Các thư viện sử dụng

- **websockets** xây dựng websocket client/server, giao tiếp với websocket để nhận dữ liệu âm thanh.
- **webbrowser** trình duyệt mặc định, truy cập vào trang HTML được phục vụ bởi HTTP server, giúp AI ghi lại quá trình phân tích giọng nói trên trang web địa phương.
- **json** cung cấp công cụ hỗ trợ xử lý chuyển đổi dữ liệu từ websocket client.
- **http.server** cung cấp công cụ xây dựng HTTP server cơ bản.
- **socketserver** hỗ trợ các kết nối mạng dễ dàng hơn.

## 8 Giao diện ứng dụng

### 8.1 Trang chủ (Home)

Trang chủ cho phép người dùng xem các thông tin cơ bản về thời tiết để họ có thể biết tình hình về môi trường sinh hoạt của mình, trang chủ còn hiển thị các biểu đồ, đồ thị giúp người dùng quan sát các biến động thông số môi trường trong khoảng thời gian vừa qua. Ngoài ra, trang chủ còn có các nút nhấn công tắc hiển thị trạng thái hoạt động của các thiết bị. Người dùng có thể thay đổi trạng thái của thiết bị bằng cách nhấn vào nút công tắc.

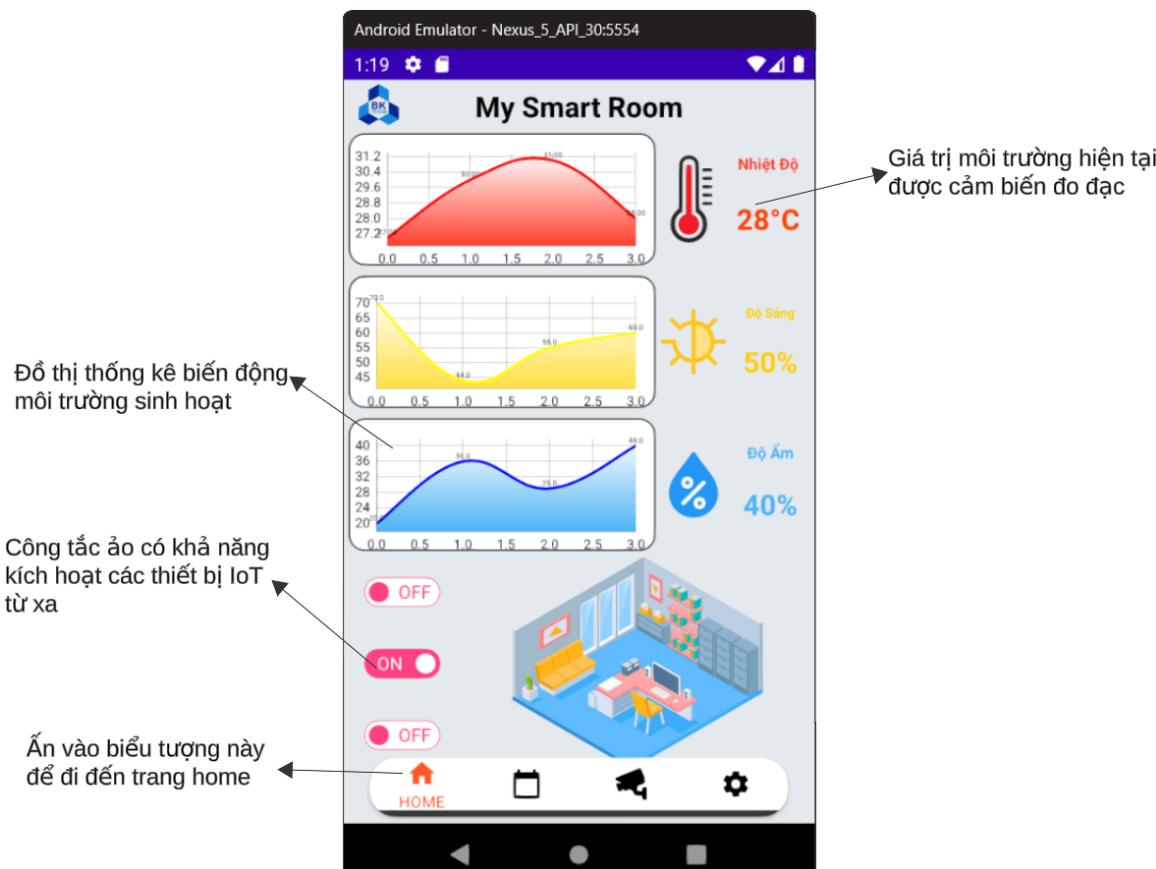


Figure 14: Giao diện của trang chủ

### 8.2 Trang hẹn giờ (Scheduler)

Trang hẹn giờ cho phép người dùng xem danh sách các hẹn giờ bật/tắt thiết bị hiện tại, đồng thời người dùng còn có thể thêm một hẹn giờ mới bằng cách nhấn vào biểu tượng dấu cộng bên dưới. Người dùng có thể xóa đi một hẹn giờ bằng cách nhấn vào biểu tượng thùng rác ở bên cạnh tác vụ hiện tại trong danh sách hẹn giờ.

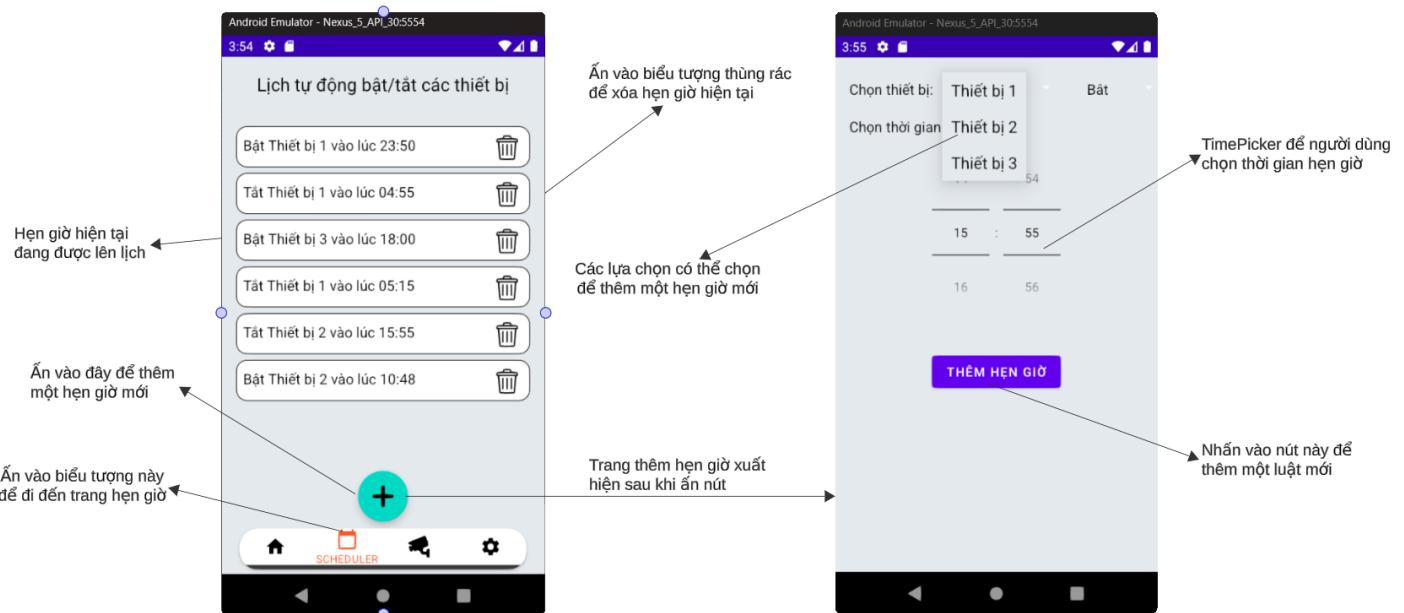


Figure 15: Giao diện của trang hẹn giờ

### 8.3 Trang quan sát (Camera)

Trang quan sát cho phép người dùng theo dõi trực tuyến khu vực sinh hoạt của mình thông qua thiết bị di động. Thiết bị di động được kết nối với camera bố trí tại khu vực sinh hoạt của người dùng.

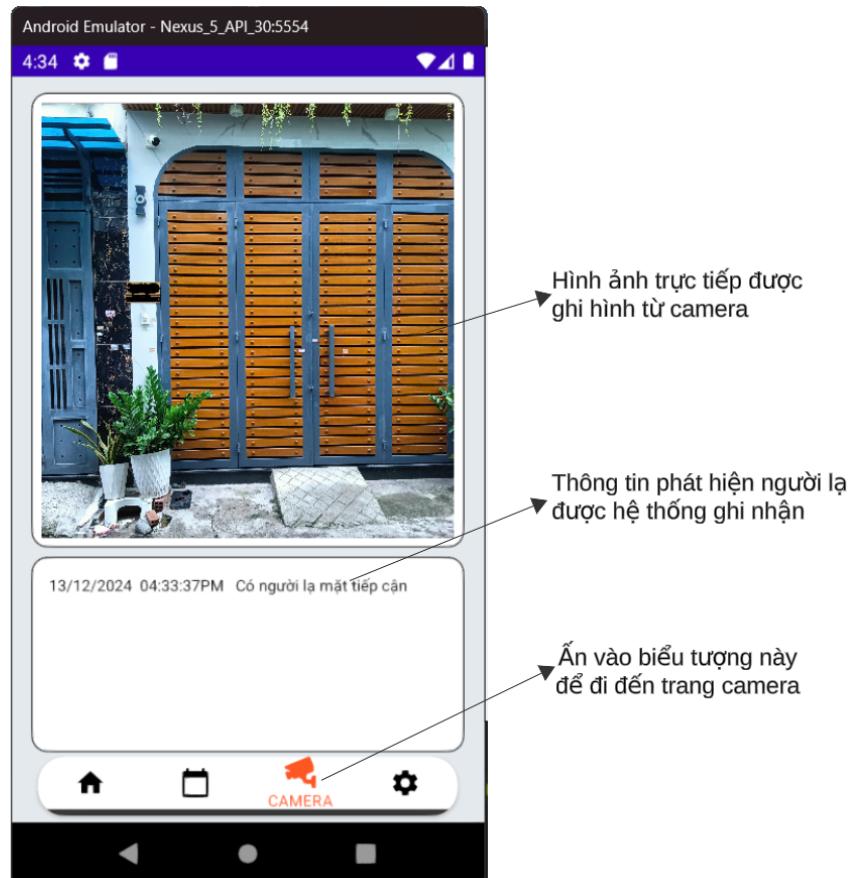


Figure 16: Giao diện của trang quan sát

## 8.4 Trang đặt luật (Set Rule)

Trang đặt luật cho phép điều khiển thiết bị theo ngữ cảnh biến của người dùng. Trang sẽ cho phép người dùng thêm hoặc xóa các luật một cách trực quan nhất để điều khiển thiết bị. Để thêm một luật mới, người dùng chọn các giá trị trong menu của những trường tương ứng và chỉ cần nhập duy nhất một trường giá trị số sau đó nhấn vào nút "Thêm". Điều kiện điều khiển thiết bị theo ngữ cảnh được tổ chức theo ngôn ngữ nói tự nhiên của con người dạng "Khi <điều kiện> tiến hành <công việc>", do đó rất dễ để người dùng sử dụng. Một danh sách các luật đã được thiết lập sẽ được hiển thị ngay phía dưới, nếu người dùng muốn xóa đi một luật nào đó họ chỉ cần nhấn trực tiếp vào biểu tượng thùng rác bên cạnh luật đó.

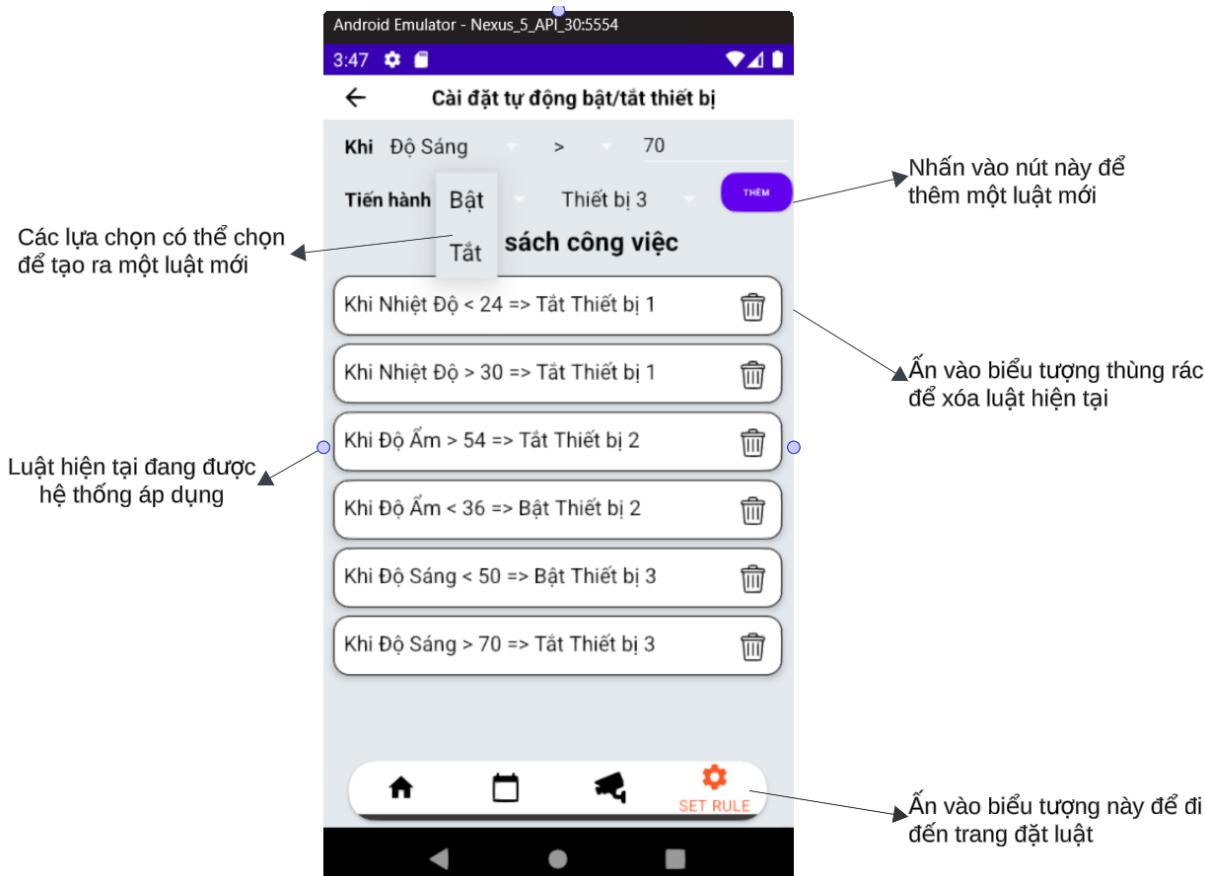


Figure 17: Giao diện của trang đặt luật



## 9 Mã nguồn và Github

- Mã nguồn: [https://github.com/dohuyminhdung/IoT\\_Project\\_Group\\_3](https://github.com/dohuyminhdung/IoT_Project_Group_3)
- Link báo cáo: <https://www.overleaf.com/read/kygqsytnqkmw#ed9fa2>

## References

- [1] Thầy Lê Trọng Nhân - Phát triển ứng dụng IoT
- [2] Python Socket Programming: Tutorial
- [3] Beginner Android Programming: Java