TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA: CÔNG NGHỆ THÔNG TIN

SỐ PHÁCH:

TÊN ĐỀ TÀI TIỂU LUẬN

HỆ THỐNG CẢNH BÁO CHỐNG TRỘM DỰA TRÊN ESP32 VÀ CẢM BIẾN CHUYỂN ĐỘNG

MÔN: PHÁT TRIỂN ỨNG DỤNG IOT - NHÓM 5

MÃ HỌC PHẦN: 2024-2025.2.TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN : VÕ VIỆT DŨNG

HUẾ,THÁNG 4 NĂM 2025

MỤC LỤC

1. Giới thiệu 1
1.1. Tổng quan về bài toán 1
1.2. Mục tiêu nghiên cứu 1
1.3. Đối tượng nghiên cứu và phạm vi nghiên cứu 1
1.4. Phương pháp nghiên cứu 2
1.5. Cấu trúc của tiểu luận 2
2. Tổng quan lý thuyết 3
2.1. Giới thiệu về ESP32 3
2.2. Cảm biến PIR và nguyên lý hoạt động 5
2.3. Giao thức gửi thông báo qua Telegram 8
2.4. Giao thức gửi thông báo qua Email 10
3. Thiết kế và xây dựng hệ thống 13
3.1. Sơ đồ tổng quan hệ thống 13
3.2. Thành phần phần cứng 13
3.3. Sơ đồ kết nối Wokwi 14
3.4. Quy trình mô phỏng trên Wokwi 17
4. Triển khai phần mềm 20
4.1. Công cụ lập trình 20
4.2. Kết nối ESP32 với Wi-Fi 20
4.3. Xử lý tín hiệu từ cảm biến PIR 21
4.4. Tích hợp thông báo Telegram 22
4.5. Tích hợp thông báo Email 24
5. Thử nghiệm và đánh giá 25
5.1. Phương pháp thử nghiệm trên Wokwi 25
5.2. Kết quả thực nghiệm 26
5.3. Đánh giá hiệu suất 33
5.4. Khó khăn và cách khắc phục 35
6. Kết luận và hướng phát triển 37
6.1. Kết luận 37
6.2. Hướng phát triển 37
TÀI LIỆU THAM KHẢO 39

1. Giới thiệu

1.1. Tổng quan về bài toán

Trong bối cảnh xã hội hiện đại, vấn đề an ninh tại các khu dân cư, văn phòng và nhà riêng ngày càng trở nên quan trọng. Các vụ trộm cắp tài sản vẫn thường xuyên xảy ra, đặc biệt ở những nơi không có hệ thống giám sát hiệu quả. Các giải pháp truyền thống như khóa cửa hay camera giám sát đôi khi không đáp ứng được yêu cầu cảnh báo tức thời khi có sự xâm nhập. Vì vậy, việc phát triển một hệ thống cảnh báo trộm đơn giản, chi phí thấp nhưng hiệu quả là một nhu cầu cấp thiết. Hệ thống này sử dụng ESP32 kết hợp cảm biến chuyển động PIR để phát hiện sự xâm nhập và gửi thông báo qua Telegram hoặc email, mang lại khả năng giám sát từ xa và phản hồi nhanh chóng.

1.2. Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu là thiết kế và xây dựng một hệ thống cảnh báo trộm dựa trên vi điều khiển ESP32 và cảm biến PIR. Hệ thống hướng đến các mục tiêu cụ thể sau:

- **Phát hiện chuyển động chính xác**: Sử dụng cảm biến PIR để nhận diện sự xuất hiện của chuyển động trong khu vực giám sát, đảm bảo độ nhạy cao và giảm thiểu báo động sai.
- **Thông báo tức thời**: Khi phát hiện sự xâm nhập, hệ thống sẽ gửi thông điệp cảnh báo đến người dùng qua Telegram hoặc email trong thời gian thực, tận dụng kết nối Wi-Fi của ESP32 để đảm bảo tốc độ và độ tin cậy.
- **Chi phí thấp và dễ triển khai**: Tạo ra một giải pháp kinh tế, sử dụng các linh kiện phổ biến, dễ tiếp cận trên thị trường, đồng thời đơn giản hóa quy trình cài đặt để phù hợp với người dùng không chuyên.
- **Mô phỏng trên Wokwi**: Xây dựng và kiểm tra toàn bộ hệ thống trong môi trường ảo trên Wokwi, từ đó đánh giá hiệu quả hoạt động mà không cần đầu tư phần cứng ngay từ đầu.

  Mục tiêu cuối cùng là cung cấp một giải pháp an ninh hiệu quả, dễ dàng mở rộng trong tương lai, chẳng hạn như tích hợp thêm camera hoặc lưu trữ dữ liệu sự kiện.

1.3. Đối tượng nghiên cứu và phạm vi nghiên cứu

- Đối tượng nghiên cứu: Nghiên cứu tập trung vào hệ thống cảnh báo trộm bao gồm các thành phần chính như vi điều khiển ESP32, cảm biến chuyển động PIR, và các giao thức truyền thông qua internet (Telegram và email). Cụ thể, ESP32 đảm nhận vai trò xử lý tín hiệu và kết nối mạng, trong khi cảm biến PIR đóng vai trò phát hiện chuyển động, còn Telegram và email là các phương thức thông báo chính.

- Phạm vi nghiên cứu: Dự án giới hạn trong việc thiết kế, lập trình và mô phỏng hệ thống trên nền tảng Wokwi – một môi trường trực tuyến cho phép tạo sơ đồ kết nối, viết mã nguồn và thử nghiệm hoạt động của ESP32 cùng cảm biến PIR. Phạm vi không bao gồm việc triển khai thực tế trên phần cứng vật lý mà chỉ tập trung vào giai đoạn mô phỏng để kiểm chứng tính khả thi. Ngoài ra, nghiên cứu chỉ sử dụng hai

kênh thông báo là Telegram và email, không mở rộng sang các phương thức khác như SMS hay ứng dụng di động trong giai đoạn này.

1.4. Phương pháp nghiên cứu

Nghiên cứu được thực hiện thông qua các phương pháp sau:

- **Nghiên cứu lý thuyết**: Thu thập và phân tích tài liệu liên quan đến đặc điểm kỹ thuật của ESP32 (ví dụ: GPIO, Wi-Fi), nguyên lý hoạt động của cảm biến PIR, và cách sử dụng API Telegram cũng như giao thức SMTP cho email. Các tài liệu tham khảo bao gồm datasheet của linh kiện, hướng dẫn từ nhà sản xuất và các bài viết kỹ thuật trên internet.
- **Thiết kế mô phỏng**: Sử dụng Wokwi để vẽ sơ đồ kết nối giữa ESP32 và cảm biến PIR, mô phỏng tín hiệu đầu vào từ cảm biến và kiểm tra phản hồi của hệ thống khi có sự kiện chuyển động.
- **Lập trình**: Viết mã nguồn cho ESP32 bằng **Arduino IDE**, một công cụ lập trình phổ biến hỗ trợ ESP32, tích hợp các thư viện như HTTPClient (cho Telegram) và ESP32 Mail Client (cho email). Mã nguồn sẽ được tối ưu để xử lý tín hiệu từ PIR và gửi thông báo một cách hiệu quả.
- **Thử nghiệm**: Chạy mô phỏng trên Wokwi, tạo các kịch bản giả lập như "có chuyển động" hoặc "không có chuyển động", sau đó ghi nhận kết quả thông báo gửi qua Telegram/email.
- **Phân tích và đánh giá**: Tổng hợp dữ liệu từ quá trình thử nghiệm, so sánh với mục tiêu ban đầu, xác định ưu điểm, hạn chế và đề xuất giải pháp khắc phục nếu có vấn đề phát sinh.

1.5. Cấu trúc của tiểu luận

Tiểu luận được chia thành 6 chương chính:

- **Chương 1 - Giới thiệu**: Trình bày bối cảnh bài toán, mục tiêu, đối tượng, phạm vi, phương pháp nghiên cứu và cấu trúc tiểu luận, đặt nền tảng cho các phần tiếp theo.
- **Chương 2 - Tổng quan lý thuyết**: Cung cấp kiến thức cơ bản về ESP32, cảm biến PIR, và cách hoạt động của Telegram/email, giúp người đọc hiểu rõ các thành phần kỹ thuật.
- **Chương 3 - Thiết kế và xây dựng hệ thống**: Mô tả chi tiết sơ đồ tổng quan, danh sách linh kiện, cách kết nối trên Wokwi và quy trình mô phỏng từng bước.
- **Chương 4 - Triển khai phần mềm**: Hướng dẫn cách lập trình ESP32, từ kết nối Wi-Fi, xử lý tín hiệu PIR đến tích hợp thông báo qua Telegram và email.
- **Chương 5 - Thử nghiệm và đánh giá**: Phân tích kết quả mô phỏng trên Wokwi, đánh giá hiệu suất hệ thống và thảo luận các vấn đề gặp phải kèm giải pháp.
- **Chương 6 - Kết luận và hướng phát triển**: Tóm tắt thành tựu của dự án, đồng thời đề xuất các hướng phát triển trong tương lai như tích hợp thêm cảm biến hoặc ứng dụng thực tế .

2. Tổng quan lý thuyết

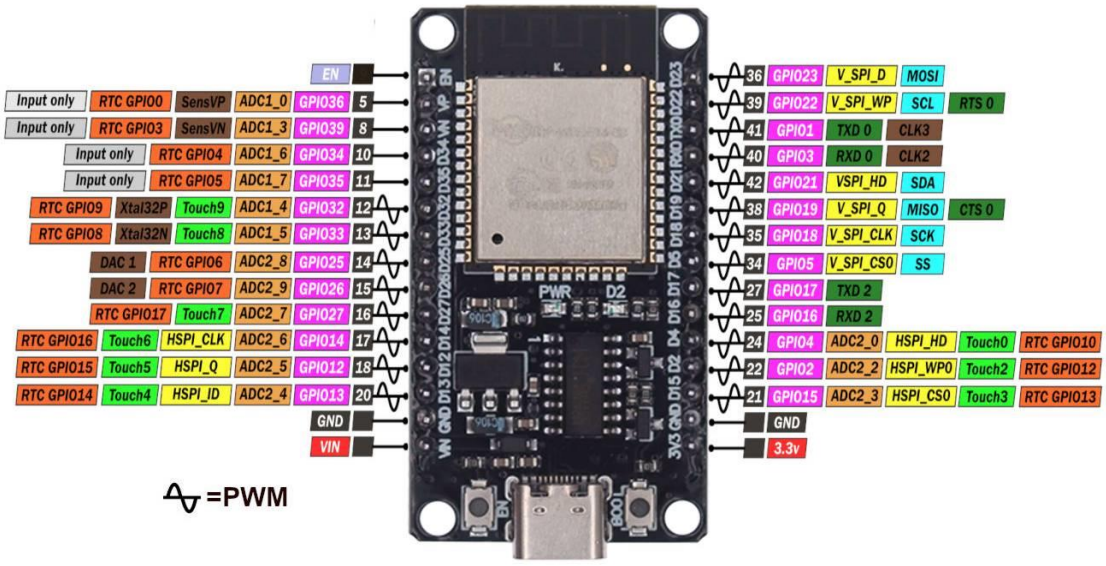
2.1. Giới thiệu về ESP32

ESP32 là một vi điều khiển tiên tiến do Espressif Systems phát triển, ra mắt vào năm 2016, được thiết kế để đáp ứng nhu cầu của các ứng dụng IoT nhờ hiệu suất cao và khả năng kết nối linh hoạt. Đây là bước tiến vượt bậc so với ESP8266, với các đặc điểm nổi bật sau:

- **Vi điều khiển:**
  - Bo mạch phát triển ESP32 được xây dựng dựa trên bộ vi điều khiển 32-bit lõi kép ESP32 mạnh mẽ.
  - Nó hoạt động ở tần số xung nhịp lên tới 240 MHz, đảm bảo hiệu suất mượt mà và hiệu quả.
- **Kết nối:**
  - Được trang bị Wi-Fi 802.11 b/g/n (2,4 GHz) để kết nối không dây, giúp truyền dữ liệu liền mạch.
  - Bluetooth 4.2/BLE (Bluetooth năng lượng thấp) được tích hợp, cho phép kết nối thiết bị IoT dễ dàng và tiết kiệm năng lượng.
- **Giao diện USB-C:**
  - Cổng USB-C có chức năng kép – cung cấp điện và truyền dữ liệu.
  - Được trang bị bộ chuyển đổi USB sang Serial CH340, sản phẩm đảm bảo kết nối ổn định với máy tính của bạn.
- **Chân GPIO:**
  - Với 38 chân GPIO, bạn có thể linh hoạt trong việc kết nối phần cứng.
  - Các chân này hỗ trợ nhiều chức năng khác nhau, bao gồm PWM, I2C, SPI, UART, v.v.
- **Bộ nhớ:**
  - Bo mạch chủ cung cấp 520KB SRAM để lưu trữ dữ liệu.
  - Sản phẩm có bộ nhớ Flash 16MB để lưu trữ chương trình, cung cấp đủ không gian cho các ứng dụng của bạn.
- **Nguồn điện:**
  - Bo mạch phát triển ESP32 hoạt động ở điện áp tối ưu là 3,3V.
  - Giao diện USB-C có thể được sử dụng để cung cấp điện và truyền dữ liệu, giúp đơn giản hóa việc thiết lập.
- **Thiết bị ngoại vi tích hợp:**
  - Cảm biến nhiệt độ tích hợp bổ sung khả năng cảm biến môi trường cho dự án của bạn.
  - Cảm biến hiệu ứng Hall được tích hợp để phát hiện từ trường.
  - Bộ đồng xử lý có công suất cực thấp giúp bạn xử lý các hoạt động tiết kiệm năng lượng.
- **Môi trường phát triển:**
  - Bo mạch tương thích với Arduino IDE, giúp cho việc phát triển và lập trình trở nên dễ dàng ngay cả với người mới bắt đầu.

- Nó cũng hỗ trợ MicroPython và các nền tảng phát triển phổ biến khác, đáp ứng nhiều sở thích khác nhau.

- **Yếu tố hình thức:**
  - Được thiết kế với kiểu dáng nhỏ gọn và các lỗ lắp tiêu chuẩn, sản phẩm này dễ dàng tích hợp vào nhiều dự án khác nhau.
  - Kích thước của bo mạch là [Ghi rõ kích thước] để lắp đặt chính xác.
- **Đèn báo LED:**
  - Đèn LED tích hợp sẵn để báo nguồn và trạng thái, giúp đơn giản hóa việc khắc phục sự cố và gỡ lỗi.
- **Nhiệt độ hoạt động:**
  - Bo mạch ESP32 Dev được thiết kế để chịu được nhiều điều kiện môi trường khác nhau, phù hợp cho cả ứng dụng trong nhà và ngoài trời.
- **Phát triển phần mềm:**
  - Tận dụng sự hỗ trợ rộng rãi từ cộng đồng mã nguồn mở với nhiều thư viện và tài nguyên, tạo điều kiện cho việc phát triển và tạo mẫu nhanh chóng.



Hình ảnh minh họa bảng mạch ESP32
Nguồn : electra.store

ESP32 đóng vai trò trung tâm trong hệ thống cảnh báo trộm, chịu trách nhiệm:

(1) đọc tín hiệu từ cảm biến PIR qua GPIO : ESP32 sử dụng chân GPIO để đọc tín hiệu từ cảm biến PIR, xác định trạng thái chuyển động (HIGH hoặc LOW)

(2) Kết nối Wi-Fi để gửi thông báo: ESP32 sử dụng module Wi-Fi tích hợp để kết nối với internet, gửi thông báo qua Telegram và email khi phát hiện chuyển động.

(3) Xử lý logic điều khiển : ESP32 thực hiện các logic điều khiển thông minh, ví dụ: chỉ gửi thông báo khi tín hiệu từ cảm biến PIR chuyển từ LOW sang HIGH, tránh lặp lại thông báo không cần thiết khi chuyển động kéo dài

2.2. Cảm biến PIR và nguyên lý hoạt động

PIR là viết tắt của *Passive Infrared Sensor* (Cảm biến hồng ngoại thụ động). Đây là một loại cảm biến điện tử đặc biệt với khả năng phát hiện sự thay đổi bức xạ hồng ngoại từ các vật thể phát nhiệt như con người, động vật hoặc các nguồn nhiệt khác trong môi trường xung quanh. Cảm biến này đóng vai trò quan trọng trong nhiều ứng dụng hiện đại, từ hệ thống an ninh, chiếu sáng thông minh đến các thiết bị điều khiển tự động trong nhà.

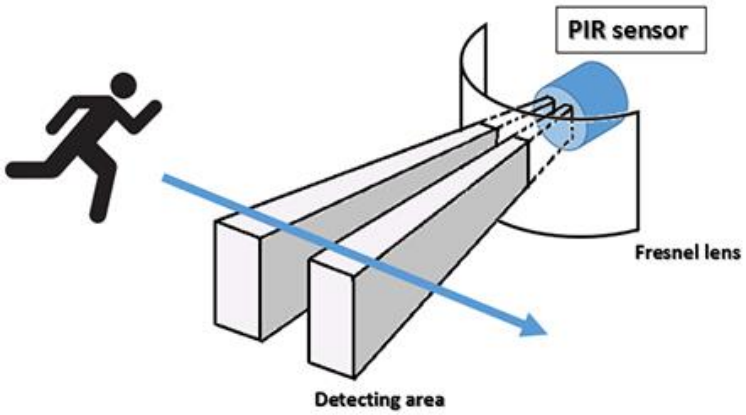
Cấu tạo chi tiết :

PIR dùng cho ứng dụng điều khiển chiếu sáng bao gồm ba thành phần chính:

- **Thấu kính Fresnel** : Thấu kính này nhận bức xạ hồng ngoại từ trường nhìn của nó và tập trung vào chính cảm biến. Vì thấu kính Fresnel được chia thành các phân đoạn nên cảm biến phía sau thấu kính nhận được nhiều chùm tia bức xạ hồng ngoại riêng biệt, mỗi chùm tia tương ứng với một phân đoạn riêng biệt của trường nhìn.

Hình ảnh thấu kính Frenel
Nguồn : Technical Support

- Thấu kính Fresnel thường được sử dụng trên cảm biến chiếm dụng hồng ngoại thụ động (chuyển động). Thiết kế của thấu kính Fresnel rất quan trọng đối với phạm vi và độ nhạy của cảm biến và sẽ thay đổi tùy theo ứng dụng dự định.



Nguyên lí hoạt động thấu kính Frenel
Nguồn : Technical Support

- Mục đích của thấu kính Fresnel trong ứng dụng cảm biến sự hiện diện là chia trường nhìn của cảm biến thành các phân đoạn riêng biệt để có thể phát hiện các mức bức xạ IR khác nhau di chuyển qua nó, cho biết sự hiện diện của vật thể đang chuyển động.

- **Cảm biến nhiệt điện** (thường được gọi trong điện tử là "pyro"). Đây là một thiết bị tương tự bao gồm hai phần tử phát hiện. Một phần tử được nối dây sao cho khi nhận được IR, nó phát ra điện áp cao trong khi nửa còn lại, khi nhận được IR, phát ra điện áp thấp. Mỗi phần tử nhận được càng nhiều IR thì điện áp càng lệch (lên hoặc xuống) so với 0. Nếu cả hai đều nhận được cùng một lượng thì các điện áp sẽ triệt tiêu lẫn nhau. Do đó, bất kỳ chuyển động nào của vật thể phát ra IR trong trường nhìn sẽ khiến điện áp đầu ra của pyro dao động - biểu thị chuyển động. Đầu vào IR càng mạnh (vì vật thể được phát hiện ở rất gần hoặc rất nóng) thì điện áp đầu ra sẽ dao động càng nhiều.

Hình ảnh cảm biến nhiệt điện
Nguồn : Technical Support

- Bên dưới vùng hình vuông ở phía trên của cảm biến là hai thành phần riêng biệt, được nối dây để tạo ra điện áp cao và thấp.

- **Mạch điện tử và phần mềm**:
  Mạch điện tử và phần mềm đi kèm cảm biến PIR có nhiệm vụ xử lý tín hiệu đầu ra từ cảm biến nhiệt điện để xác định xem có chuyển động hay không.

  - **Mạch điện tử**: Bao gồm bộ khuếch đại tín hiệu (amplifier) để khuếch đại tín hiệu nhỏ từ cảm biến nhiệt điện, bộ lọc tín hiệu (filter) để loại bỏ nhiễu, và mạch so sánh (comparator) để chuyển đổi tín hiệu analog thành tín hiệu số (HIGH hoặc LOW). Một số cảm biến PIR (như HC-SR501) còn có các biến trở để điều chỉnh độ nhạy và thời gian trễ.
  - **Phần mềm**: Phần mềm tích hợp trong vi điều khiển (như ESP32 trong dự án này) đọc tín hiệu số từ chân OUT của cảm biến PIR (chân D15) và xử lý để kích hoạt các hành động, như gửi thông báo, bật LED, hoặc kích hoạt buzzer. Trong dự án, mã nguồn được lập trình để chỉ gửi thông báo một lần trong vòng 5 phút, tránh lặp lại không cần thiết .

Nguyên lí hoạt động :

Cảm biến PIR hoạt động dựa trên việc phát hiện sự thay đổi bức xạ hồng ngoại trong trường nhìn của nó, từ đó suy ra sự hiện diện của chuyển động. Nguyên lý hoạt động thực hiện qua các bước sau:

1. **Thu nhận bức xạ hồng ngoại**:
   Thấu kính Fresnel đóng vai trò chia trường nhìn của cảm biến thành nhiều vùng phát hiện riêng biệt (như được minh họa trong hình ảnh). Mỗi vùng phát hiện tương ứng với một phân đoạn của thấu kính, cho phép cảm biến nhận biết sự thay đổi bức xạ hồng ngoại khi một vật thể di chuyển qua các vùng này.
   a. Khi không có chuyển động, bức xạ hồng ngoại từ môi trường (như nhiệt độ phòng) là ổn định, và cảm biến không tạo ra tín hiệu đáng kể.
   b. Khi một nguồn phát hồng ngoại (như cơ thể người với nhiệt độ khoảng 36-37°C) di chuyển ngang qua trường nhìn (hướng "Chuyển động ngang của nguồn hồng ngoại"), bức xạ hồng ngoại từ vật thể sẽ lần lượt đi qua các vùng phát hiện khác nhau. Thấu kính Fresnel tập trung bức xạ này và truyền đến cảm biến nhiệt điện bên trong.
2. **Phát hiện chuyển động qua cảm biến nhiệt điện**:
   Khi một vật thể phát ra bức xạ hồng ngoại (như cơ thể người với nhiệt độ khoảng 36-37°C) di chuyển qua trường nhìn, lượng bức xạ hồng ngoại trên mỗi phần tử

của cảm biến nhiệt điện thay đổi. Điều này làm cho điện áp đầu ra của cảm biến dao động (tăng hoặc giảm so với 0).

- Ví dụ: Nếu một người đi ngang qua cảm biến, phần tử thứ nhất có thể nhận được nhiều bức xạ hơn phần tử thứ hai, tạo ra điện áp dương. Khi người đó tiếp tục di chuyển, phần tử thứ hai nhận được nhiều bức xạ hơn, tạo ra điện áp âm. Sự dao động này được cảm biến ghi nhận như một tín hiệu chuyển động.

3. **Xử lý tín hiệu và kích hoạt hành động**:
   Mạch điện tử khuếch đại và lọc tín hiệu từ cảm biến nhiệt điện, sau đó chuyển đổi thành tín hiệu số (HIGH khi có chuyển động, LOW khi không có). Tín hiệu này được gửi đến ESP32 qua chân D15. Phần mềm trên ESP32 xử lý tín hiệu và thực hiện các hành động tương ứng, như gửi thông báo qua Telegram/email, bật LED, và kích hoạt buzzer .

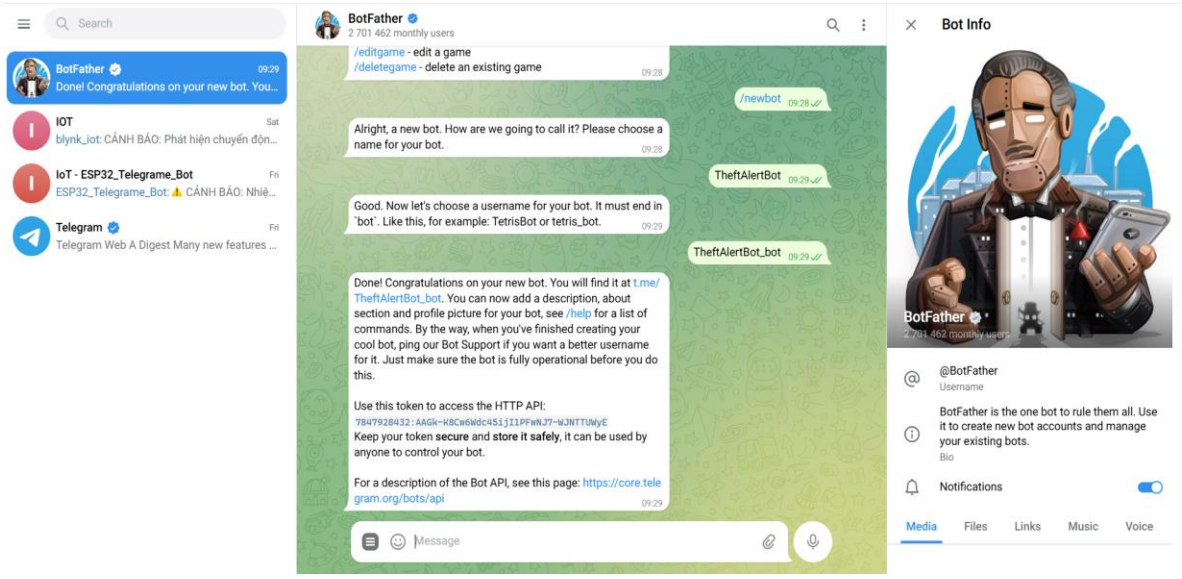
Ứng dụng trong dự án :

Cảm biến PIR được kết nối với chân GPIO 14 của ESP32. Khi tín hiệu chuyển từ LOW sang HIGH, ESP32 sẽ kích hoạt lệnh gửi thông báo qua Telegram hoặc email. Để tránh báo động sai (do gió, ánh sáng mặt trời), hệ thống có thể được tối ưu bằng cách đặt cảm biến ở vị trí tránh nhiễu và điều chỉnh độ nhạy.

2.3. Giao thức gửi thông báo qua Telegram

Telegram là một ứng dụng nhắn tin mã hóa ra đời năm 2013, nổi tiếng với tính bảo mật cao và khả năng hỗ trợ bot – các tài khoản tự động có thể tương tác với người dùng hoặc thiết bị qua API. Trong dự án này, Telegram được chọn làm kênh thông báo chính nhờ tốc độ nhanh và tính linh hoạt, phù hợp với yêu cầu cảnh báo tức thời khi phát hiện chuyển động.

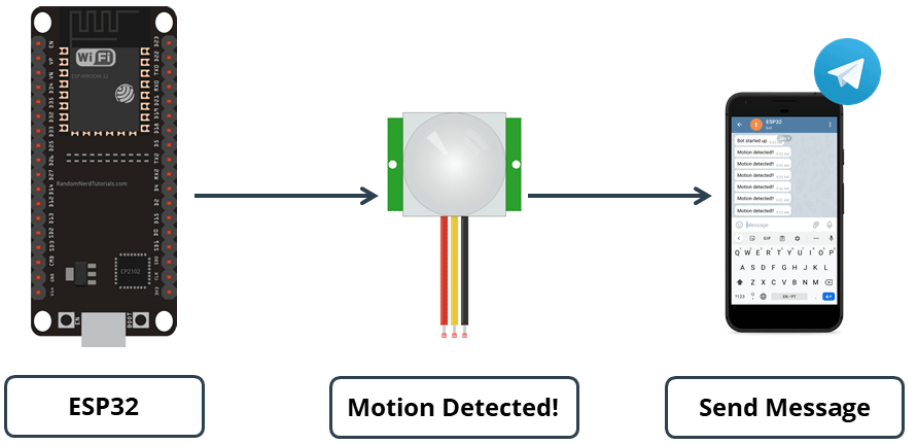
- **Thiết lập bot trên Telegram**:
  Để sử dụng Telegram trong hệ thống, trước tiên cần tạo một bot thông qua BotFather – một bot chính thức của Telegram dùng để quản lý các bot khác. Người dùng gửi lệnh "/newbot" trong ứng dụng Telegram, đặt tên cho bot (ví dụ: "TheftAlertBot"), và nhận được một chuỗi ký tự gọi là **API Token**. Chuỗi này có dạng một mã dài kết hợp số và chữ, ví dụ như một dãy khoảng 40 ký tự, đóng vai trò như "chìa khóa" để ESP32 giao tiếp với máy chủ Telegram. Sau đó, để xác định người nhận thông báo, người dùng cần gửi một tin nhắn thử đến bot và sử dụng công cụ kiểm tra để lấy **Chat ID** – một dãy số duy nhất đại diện cho tài khoản hoặc nhóm nhận tin nhắn. Cả API Token và Chat ID đều cần được lưu trữ cẩn thận để sử dụng trong quá trình gửi thông báo.



- **Nguyên lý hoạt động**:
  Giao thức gửi thông báo qua Telegram dựa trên **Telegram Bot API**, hoạt động theo mô hình client-server sử dụng giao thức HTTP/HTTPS – một phương thức truyền dữ liệu phổ biến trên internet. Khi cảm biến PIR phát hiện chuyển động, ESP32 sẽ tạo một yêu cầu HTTP gửi đến máy chủ Telegram. Yêu cầu này bao gồm ba phần chính:

  - Địa chỉ máy chủ Telegram (luôn cố định là "api.telegram.org").
  - API Token để xác thực bot.
  - Thông tin tin nhắn, bao gồm Chat ID (địa chỉ người nhận) và nội dung thông báo (ví dụ: "Cảnh báo: Phát hiện chuyển động lúc 14:30, ngày 26/03/2025").
    Máy chủ Telegram nhận yêu cầu, kiểm tra tính hợp lệ của API Token, sau đó chuyển tiếp tin nhắn đến Chat ID tương ứng. Toàn bộ quá trình này diễn ra qua kết nối Wi-Fi của ESP32, tận dụng khả năng kết nối mạng không dây của vi điều khiển.

- **Quy trình chi tiết**:
  1. ESP32 phát hiện tín hiệu từ cảm biến PIR chuyển từ trạng thái không có chuyển động sang có chuyển động.
  2. ESP32 kết nối đến mạng Wi-Fi đã được cấu hình trước (ví dụ: mạng gia đình hoặc điểm phát sóng).
  3. ESP32 xây dựng một chuỗi địa chỉ web hoàn chỉnh, kết hợp API Token, Chat ID và nội dung thông báo thành một đường dẫn duy nhất.
  4. Yêu cầu được gửi đi dưới dạng một lệnh gọi qua internet, tương tự như khi người dùng truy cập một trang web.
  5. Máy chủ Telegram xử lý yêu cầu và gửi tin nhắn đến ứng dụng Telegram của người dùng trong vòng chưa đến một giây nếu mạng ổn định.



Minh họa hoạt động Telegram khi ESP32 phát hiện chuyển động
Nguồn : Randoms Nerd Tutorial

- **Ưu điểm**:

  - **Tốc độ cao**: Tin nhắn được gửi gần như tức thời, thường chỉ mất dưới 1 giây, rất phù hợp với yêu cầu cảnh báo nhanh.
  - **Miễn phí**: Telegram không tính phí cho việc gửi tin nhắn qua bot, không giới hạn số lượng thông báo.
  - **Đa nền tảng**: Người dùng có thể nhận thông báo trên điện thoại, máy tính bảng, hoặc máy tính mà không cần phần cứng bổ sung.
  - **Dễ tích hợp**: API đơn giản, chỉ cần một yêu cầu HTTP là đủ để gửi tin nhắn, không đòi hỏi cấu hình phức tạp.

- **Hạn chế**:

  - **Phụ thuộc internet**: Nếu Wi-Fi bị gián đoạn hoặc không ổn định, thông báo sẽ không được gửi đi.
  - **Bảo mật thông tin**: API Token và Chat ID cần được giữ bí mật; nếu bị lộ, người khác có thể giả mạo hoặc chặn thông báo.
  - **Yêu cầu người dùng cài đặt Telegram**: Không phải ai cũng sử dụng ứng dụng này, có thể gây bất tiện cho một số đối tượng.

Telegram được ưu tiên trong hệ thống này vì khả năng phản hồi nhanh và tính phổ biến ngày càng tăng, đặc biệt trong cộng đồng công nghệ và IoT.

2.4. Giao thức gửi thông báo qua Email

Email là một phương thức liên lạc điện tử lâu đời, sử dụng giao thức **SMTP (Simple Mail Transfer Protocol)** để gửi thư từ thiết bị này đến thiết bị khác qua internet. Được phát triển từ năm 1982, SMTP vẫn là tiêu chuẩn phổ biến cho các hệ thống gửi email tự

động, và trong dự án này, nó đóng vai trò như một kênh thông báo dự phòng bên cạnh Telegram.

- **Thiết lập tài khoản email**:
  Để gửi email từ ESP32, cần một tài khoản email hoạt động, chẳng hạn như Gmail – một trong những dịch vụ phổ biến nhất hiện nay. Người dùng cần cung cấp thông tin sau:

  - **Máy chủ SMTP**: Với Gmail, địa chỉ là "smtp.gmail.com".
  - **Cổng kết nối**: Thường là 465 nếu dùng mã hóa SSL (bảo mật cao) hoặc 587 nếu dùng TLS (bảo mật nhẹ hơn).
  - **Thông tin đăng nhập**: Bao gồm địa chỉ email (ví dụ: "example@gmail.com") và mật khẩu. Tuy nhiên, với Gmail, nếu tài khoản bật xác thực hai yếu tố (2FA), người dùng phải tạo một "Mật khẩu ứng dụng" – một chuỗi 16 ký tự đặc biệt (ví dụ: "abcd efgh ijkl mnop") – thông qua cài đặt bảo mật của Google. Thông tin này sẽ được cấu hình trong hệ thống để ESP32 có thể đăng nhập và gửi email.

- **Nguyên lý hoạt động**:
  Giao thức SMTP hoạt động như một "bưu điện điện tử", trong đó ESP32 đóng vai trò là "người gửi thư", còn máy chủ SMTP (như smtp.gmail.com) là "bưu điện" trung gian. Khi cảm biến PIR phát hiện chuyển động, ESP32 thực hiện các bước sau:

  - Kết nối đến máy chủ SMTP qua Wi-Fi, sử dụng giao thức TCP/IP – nền tảng của truyền thông internet.
  - Gửi một loạt lệnh SMTP để xác thực (như thông báo "xin chào", cung cấp tài khoản/mật khẩu) và truyền dữ liệu email.
  - Dữ liệu email bao gồm:

    - Địa chỉ người gửi (thường là email đã cấu hình).
    - Địa chỉ người nhận (có thể là cùng email hoặc email khác).
    - Tiêu đề (ví dụ: "Cảnh báo an ninh").
    - Nội dung (ví dụ: "Hệ thống phát hiện chuyển động lúc 14:30, ngày 26/03/2025").

  - Máy chủ SMTP nhận dữ liệu, xác minh thông tin đăng nhập, sau đó chuyển tiếp email đến hộp thư của người nhận qua hệ thống máy chủ email toàn cầu.

- **Quy trình chi tiết**:
  1. ESP32 nhận tín hiệu từ cảm biến PIR, xác định có chuyển động.
  2. ESP32 kết nối đến mạng Wi-Fi đã được định sẵn.

3. ESP32 mở một phiên kết nối với máy chủ SMTP (smtp.gmail.com) qua cổng 465 hoặc 587.
4. ESP32 gửi thông tin đăng nhập (email và mật khẩu ứng dụng) để xác thực với máy chủ.
5. Sau khi xác thực thành công, ESP32 gửi thông tin email (người nhận, tiêu đề, nội dung).
6. Máy chủ SMTP xử lý và gửi email đến hộp thư của người dùng, thường mất từ 2 đến 10 giây tùy tốc độ mạng và dịch vụ email.

- **Ưu điểm**:

  - **Phổ biến**: Hầu hết mọi người đều có email, không cần cài đặt ứng dụng bổ sung như Telegram.
  - **Lưu trữ lâu dài**: Email được lưu trong hộp thư, cho phép người dùng tra cứu lại thông báo bất cứ lúc nào.
  - **Tính linh hoạt**: Có thể gửi đến nhiều người nhận cùng lúc hoặc thêm thông tin chi tiết (như thời gian, vị trí).
  - **Độc lập với ứng dụng**: Không yêu cầu người dùng phải mở ứng dụng cụ thể để nhận thông báo.

- **Hạn chế**:

  - **Tốc độ chậm hơn**: So với Telegram, email mất nhiều thời gian hơn để đến tay người nhận, đôi khi bị trì hoãn nếu máy chủ bận.
  - **Phụ thuộc bảo mật**: Nếu cấu hình sai (như không dùng mật khẩu ứng dụng với Gmail), máy chủ sẽ từ chối gửi email.
  - **Phức tạp hơn**: Quá trình xác thực và gửi email đòi hỏi nhiều bước hơn so với Telegram, dễ xảy ra lỗi nếu thông tin không chính xác.
  - **Phụ thuộc internet**: Tương tự Telegram, email không hoạt động nếu Wi-Fi bị gián đoạn.

Email được tích hợp như một phương thức bổ sung trong hệ thống, đảm bảo người dùng vẫn nhận được cảnh báo ngay cả khi không sử dụng Telegram, đồng thời cung cấp khả năng lưu trữ thông tin lâu dài.

3. Thiết kế và xây dựng hệ thống

3.1. Sơ đồ tổng quan hệ thống

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động được thiết kế để phát hiện sự xâm nhập và gửi thông báo đến người dùng thông qua Telegram hoặc email. Sơ đồ tổng quan của hệ thống bao gồm các khối chức năng chính:

- **Khối cảm biến**: Cảm biến PIR (Passive Infrared Sensor) phát hiện chuyển động bằng cách nhận biết sự thay đổi bức xạ hồng ngoại trong khu vực giám sát. Khi có chuyển động, cảm biến gửi tín hiệu đến khối xử lý trung tâm.
- **Khối xử lý trung tâm**: Vi điều khiển ESP32 nhận tín hiệu từ cảm biến PIR, xử lý dữ liệu, và thực hiện các tác vụ như kết nối Wi-Fi, gửi thông báo qua Telegram hoặc email.
- **Khối thông báo**: Hệ thống sử dụng hai kênh thông báo là Telegram và email để gửi cảnh báo đến người dùng khi phát hiện chuyển động.
- **Khối hiển thị**: Một đèn LED đỏ được tích hợp để biểu thị trạng thái hoạt động của hệ thống (bật sáng khi có chuyển động).
- **Khối âm thanh** : Một còi báo động (buzzer) được tích hợp để biểu thị trạng thái hoạt động của hệ thống (phát ra âm thanh khi có chuyển động).
- **Nguồn điện**: Cung cấp năng lượng cho ESP32, cảm biến PIR và LED, được mô phỏng trong môi trường Wokwi.

Quy trình hoạt động tổng quan: Cảm biến PIR phát hiện chuyển động → ESP32 nhận tín hiệu → ESP32 kích hoạt đèn LED , còi báo động và gửi thông báo qua Telegram/email → Người dùng nhận cảnh báo. Hệ thống được thiết kế đơn giản, hiệu quả và tập trung vào việc mô phỏng trên Wokwi để kiểm tra tính khả thi trước khi triển khai thực tế.

3.2. Thành phần phần cứng

Hệ thống cảnh báo trộm được thiết kế và mô phỏng trên Wokwi, một nền tảng trực tuyến cho phép tạo sơ đồ mạch và lập trình vi điều khiển mà không cần phần cứng thực tế. Việc xác định thành phần cứng là cần thiết để hiểu rõ cách chúng tương tác trong thiết kế . Các thành phần chính bao gồm :

- **ESP32**:
    - Sử dụng module ESP32 DevKit V1 (có sẵn trong Wokwi), một phiên bản phổ biến của ESP32 với các chân GPIO, nguồn 3.3V/5V và GND được mô phỏng đầy đủ.
    - ESP32 đảm nhiệm vai trò xử lý tín hiệu, kết nối Wi-Fi và gửi thông báo.
- **Cảm biến PIR**:

- Cảm biến chuyển động PIR (HC-SR501), có 3 chân: VCC (nguồn), GND (đất), và OUT (tín hiệu đầu ra).
- Trong Wokwi, cảm biến PIR được mô phỏng với khả năng xuất tín hiệu HIGH (3.3V) khi có chuyển động và LOW (0V) khi không có chuyển động.

- **Đèn LED đỏ**:

  - Một đèn LED màu đỏ được thêm vào để hiển thị trực quan trạng thái của hệ thống. LED sáng khi có chuyển động, giúp dễ dàng quan sát trong quá trình mô phỏng.
  - LED có hai chân: Anode (A, cực dương) và Cathode (C, cực âm).

- **Còi báo động (buzzer) :**

  - Một buzzer chủ động được thêm để phát âm thanh cảnh báo tại chỗ khi có chuyển động với hai chân : GND (đất) và SIG (tín hiệu) .
  - Trong Wokwi, buzzer được mô phỏng với khả năng phát ra âm thanh ảo (hiển thị bằng hiệu ứng đồ họa hoặc thông báo trên giao diện) khi được kích hoạt .

- **Nguồn điện**:

  - Trong thực tế, ESP32 và PIR cần nguồn 5V (qua chân VIN của ESP32) hoặc 3.3V, có thể cấp qua USB hoặc pin.
  - Trên Wokwi, nguồn điện được giả lập tự động, không cần linh kiện vật lý riêng, nhưng vẫn đảm bảo cung cấp điện áp phù hợp cho các thành phần.

- **Kết nối Wi-Fi**:

  - ESP32 tích hợp sẵn module Wi-Fi, không cần phần cứng bổ sung. Trong Wokwi, kết nối Wi-Fi được mô phỏng bằng cách cấu hình thông tin mạng (SSID và mật khẩu) trong mã nguồn.

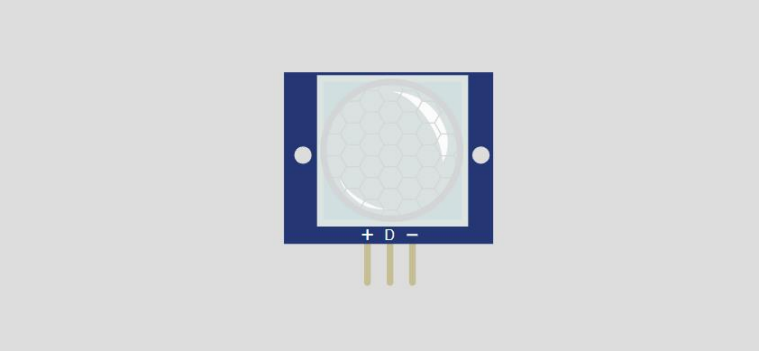
3.3. Sơ đồ kết nối Wokwi

Sơ đồ kết nối trên Wokwi bao gồm ESP32, cảm biến PIR , một đèn LED đỏ và một còi báo động Buzzer. Dưới đây là mô tả chi tiết:

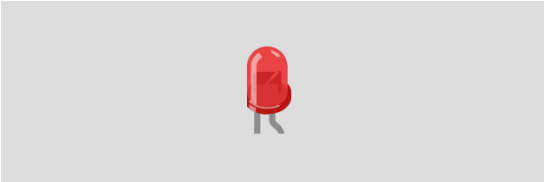
- **Các thành phần trong sơ đồ**:

  - **ESP32 DevKit V1**: Một module ESP32 với các chân được đánh số rõ ràng, bao gồm GPIO, VIN (5V), 3V3 (3.3V), và GND.

- **Cảm biến PIR**: Có 3 chân: VCC, GND, và OUT, được mô phỏng với khả năng phát hiện chuyển động ảo.



- **LED đỏ**: Một đèn LED với hai chân Anode (A) và Cathode (C), dùng để hiển thị trạng thái.



- **Còi báo động (buzzer) :** Một buzzer chủ động với hai chân : GND (đất) và SIG (tín hiệu), được mô phỏng với khả năng phát ra âm thanh ảo khi được kích hoạt .

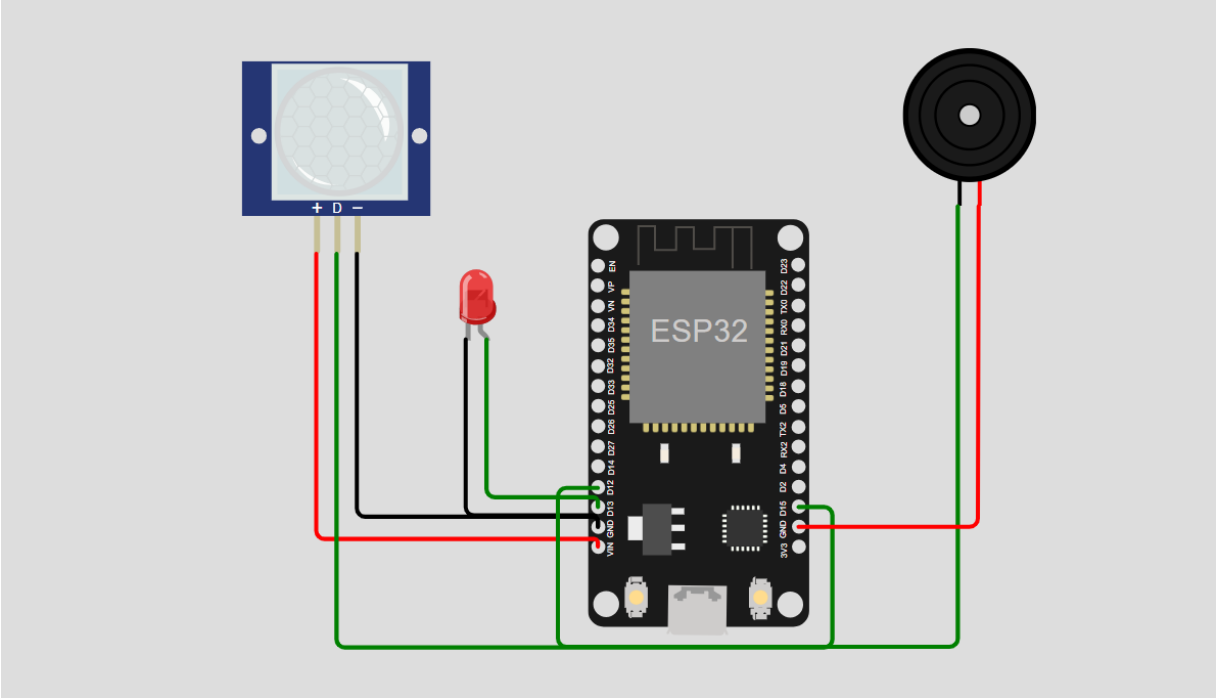
- **Kết nối chi tiết**:
  - **Cảm biến PIR**:
    - **VCC của PIR**: Kết nối với chân VIN của ESP32 (cung cấp 5V). Trong sơ đồ Wokwi, dây màu đỏ được vẽ từ chân VIN của ESP32 đến chân VCC của PIR. Chân VIN được chọn vì cảm biến PIR HC-SR501 thường hoạt động tốt ở mức 5V, và ESP32 DevKit V1 có thể cung cấp mức điện áp này.
    - **GND của PIR**: Kết nối với chân GND.2 (một trong các chân GND) của ESP32 để hoàn thành mạch điện. Dây màu đen được vẽ từ chân GND của PIR đến chân GND.2 của ESP32.
    - **OUT của PIR**: Kết nối với chân D15 (GPIO 15) của ESP32 để truyền tín hiệu chuyển động. Dây màu xanh lá được vẽ từ chân OUT của PIR đến chân D15 của ESP32. Khi có chuyển động, chân OUT sẽ xuất tín hiệu HIGH, và ESP32 sẽ đọc tín hiệu này qua GPIO 15.
  - **LED đỏ**:
    - **Anode (A) của LED**: Kết nối với chân D13 (GPIO 13) của ESP32 để điều khiển trạng thái LED. Dây màu xanh lá được vẽ từ chân D13 của ESP32 đến chân Anode của LED. Chân D13 được chọn vì nó thường được dùng để điều khiển LED trên các module ESP32 DevKit V1 (có thể tận dụng LED tích hợp trên bo mạch, nhưng ở đây là LED ngoài).
    - **Cathode (C) của LED**: Kết nối với chân GND.2 của ESP32 để hoàn thành mạch. Dây màu đen được vẽ từ chân Cathode của LED đến chân GND.2 của ESP32. (Lưu ý: Trong thực tế, cần một điện trở nối tiếp khoảng 220-330 ohm để bảo vệ LED, nhưng trong Wokwi, điều này được bỏ qua vì LED ảo không cần điện trở).
  - **ESP32 với Serial Monitor**:
    - Chân TX0 và RX0 của ESP32 được kết nối với Serial Monitor của Wokwi (mô phỏng giao tiếp UART). Điều này cho phép hiển thị

thông tin gỡ lỗi (như trạng thái tín hiệu hoặc thông báo gửi đi) trong quá trình mô phỏng.

- **Buzzer** :

  - **GND của buzzer**: Kết nối với chân GND.2 của ESP32 để hoàn thành mạch. Dây màu đen được vẽ từ chân GND của buzzer đến chân GND.2 của ESP32.
  - **SIG của buzzer**: Kết nối với chân D12 (GPIO 12) của ESP32 để điều khiển trạng thái. Dây màu xanh lá được vẽ từ chân SIG của buzzer đến chân D12 của ESP32. Khi chân D12 được đặt ở trạng thái HIGH, buzzer sẽ kêu; khi ở trạng thái LOW, buzzer sẽ tắt.

➢ **Sơ đồ hoàn chỉnh** :



3.4. Quy trình mô phỏng trên Wokwi

Quy trình mô phỏng trên Wokwi được thực hiện qua các bước cụ thể, từ việc tạo dự án, thiết lập sơ đồ, đến kiểm tra hoạt động của hệ thống:

- **Bước 1: Tạo dự án mới trên Wokwi**

  - Truy cập trang web Wokwi (wokwi.com), đăng nhập hoặc tạo tài khoản nếu cần.
  - Chọn "Start from scratch" hoặc chọn mẫu dự án ESP32 để bắt đầu.

- Giao diện Wokwi hiển thị ba khu vực chính: khu vực vẽ sơ đồ (bên trái), khu vực viết mã (bên phải), và cửa sổ mô phỏng (xuất hiện khi chạy).

- **Bước 2: Thêm linh kiện**

  - Từ thanh công cụ bên trái, tìm và chọn "ESP32 DevKit V1", kéo vào khu vực vẽ sơ đồ. Linh kiện này được đặt ở vị trí trung tâm để dễ kết nối.
  - Tìm “PIR Motion Sensor” trong danh sách linh kiện, thêm vào và đặt ở phía trên bên trái của ESP32.
  - Tìm “LED” trong danh sách, chọn màu đỏ, thêm vào và đặt ở vị trí giữa ESP32 và PIR để dễ quan sát.
  - Tìm “Buzzer” trong danh sach, thêm vào và đặt ở vị trí phù hợp .

- **Bước 3: Kết nối linh kiện**

  - Sử dụng công cụ vẽ dây trong Wokwi để thực hiện các kết nối:

    - Nối chân VIN của ESP32 với VCC của PIR (dây đỏ).
    - Nối chân GND.2 của ESP32 với GND của PIR (dây đen).
    - Nối chân D15 của ESP32 với OUT của PIR (dây xanh lá).
    - Nối chân D13 của ESP32 với Anode của LED (dây xanh lá).
    - Nối chân GND.2 của ESP32 với Cathode của LED (dây đen).
    - Nối chân GND của buzzer với GND của ESP32 (dây đen)
    - Nối chân SIG của buzzer với D12 của ESP32 (dây xanh lá)

  - Kiểm tra lại các kết nối để đảm bảo không có dây nào bị bỏ sót hoặc nối sai. Wokwi sẽ hiển thị lỗi nếu có vấn đề, ví dụ như kết nối GND không đúng.

- **Bước 4: Cấu hình mô phỏng**

  - **Cảm biến PIR**: Trong Wokwi, cảm biến PIR được mô phỏng với khả năng kích hoạt thủ công. Người dùng có thể nhấp vào biểu tượng cảm biến để giả lập chuyển động, làm chân OUT chuyển từ LOW sang HIGH.
  - **LED**: LED đỏ sẽ sáng khi chân D13 của ESP32 được đặt ở trạng thái HIGH, biểu thị rằng hệ thống đã phát hiện chuyển động.
  - **Wi-Fi**: Kết nối Wi-Fi được mô phỏng bằng cách cấu hình thông tin mạng (SSID và mật khẩu) trong mã nguồn. Trong Wokwi, việc gửi thông báo qua Telegram/email không thực sự diễn ra (do không có internet thật), nhưng có thể giả lập bằng cách in thông điệp ra Serial Monitor.
  - **Serial Monitor**: Chân TX0 và RX0 của ESP32 được kết nối với Serial Monitor để hiển thị thông tin gỡ lỗi, chẳng hạn như trạng thái tín hiệu từ PIR hoặc nội dung thông báo gửi đi.

- **Buzzer :** Trong Wokwi , buzzer được mô phỏng khả năng phát âm thanh. Khi chân D12 của ESP32 được đặt ở trạng thái HIGH, buzzer sẽ kêu (hiển thị bằng hiệu ứng đồ họa như sóng âm trên giao diện Wokwi).
- **Điều khiển qua Telegram:** Các lệnh Telegram (“/status”, “/on”, “/off”, “/help”) được mô phỏng bằng cách giả lập tin nhắn đến từ người dùng. Trong Wokwi, có thể sử dụng Serial Monitor để nhập thủ công và quan sát phản hồi .

- **Bước 5: Kiểm tra hoạt động**

  - **Kịch bản 1 - Hệ thống tắt**: Gửi lệnh "/off" qua Serial Monitor (mô phỏng lệnh Telegram). Hệ thống chuyển sang trạng thái tắt, LED và buzzer không hoạt động, ngay cả khi kích hoạt PIR. Serial Monitor hiển thị thông điệp: "Hệ thống đã được TẮT qua Telegram".
  - **Kịch bản 2 - Bật hệ thống và không có chuyển động**: Gửi lệnh "/on" để bật hệ thống. Quan sát trạng thái ban đầu: tín hiệu từ PIR là LOW, LED và buzzer đều tắt. Gửi lệnh "/status", Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Không có chuyển động - LED: Tắt - Còi báo động: Tắt".
  - **Kịch bản 3 - Có chuyển động**: Nhấp vào cảm biến PIR trên Wokwi để giả lập chuyển động. Tín hiệu từ chân OUT của PIR chuyển sang HIGH, ESP32 nhận tín hiệu qua chân D15, kích hoạt:
    - LED đỏ sáng (chân D13 HIGH).
    - Buzzer kêu (chân D12 HIGH, Wokwi hiển thị hiệu ứng âm thanh).
    - Gửi thông báo (mô phỏng bằng cách in thông điệp ra Serial Monitor, ví dụ: "Cảnh báo: Phát hiện chuyển động!").
      Gửi lệnh "/status", Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Có chuyển động - LED: Sáng - Còi báo động: Đang kêu".
  - **Kịch bản 4 - Tắt hệ thống khi đang có chuyển động**: Gửi lệnh "/off" trong lúc buzzer đang kêu. Hệ thống tắt, LED và buzzer ngừng hoạt động, Serial Monitor hiển thị: "Hệ thống đã được TẮT qua Telegram".
  - **Kịch bản 5 - Yêu cầu trợ giúp**: Gửi lệnh "/help", Serial Monitor hiển thị danh sách lệnh: "Các lệnh có sẵn: /status - Kiểm tra trạng thái hệ thống, /on - Bật hệ thống, /off - Tắt hệ thống, /help - Hiển thị trợ giúp".

4. Triển khai phần mềm

4.1. Công cụ lập trình

Để triển khai phần mềm cho hệ thống, công cụ lập trình được sử dụng là **Arduino IDE**, một môi trường phát triển phổ biến và dễ sử dụng cho các vi điều khiển như ESP32. Arduino IDE được chọn vì các lý do sau:

- **Tính phổ biến**: Arduino IDE có cộng đồng người dùng lớn, cung cấp nhiều tài liệu hướng dẫn và thư viện hỗ trợ, giúp đơn giản hóa quá trình lập trình.
- **Hỗ trợ ESP32**: Arduino IDE có thể được cấu hình để lập trình ESP32 thông qua việc cài đặt thêm **ESP32 Board Manager**. Điều này cho phép sử dụng các thư viện và hàm chuyên dụng cho ESP32, như kết nối Wi-Fi hoặc giao tiếp với cảm biến.
- **Tích hợp với Wokwi**: Wokwi hỗ trợ trực tiếp mã nguồn từ Arduino IDE, cho phép viết và chạy mã trong môi trường mô phỏng mà không cần phần cứng vật lý.
- **Giao diện đơn giản**: Arduino IDE có giao diện thân thiện, bao gồm trình soạn thảo mã, công cụ biên dịch, và Serial Monitor để gỡ lỗi.

Quy trình chuẩn bị Arduino IDE:

- Tải và cài đặt Arduino IDE từ trang web chính thức (arduino.cc).
- Cài đặt ESP32 Board Manager bằng cách thêm URL của Espressif vào phần "Preferences" của Arduino IDE, sau đó chọn và cài đặt bo mạch "ESP32 Dev Module" trong "Boards Manager".
- Cài đặt các thư viện cần thiết (sẽ được đề cập ở các mục sau) thông qua "Library Manager" trong Arduino IDE.

Arduino IDE sẽ được sử dụng để viết mã điều khiển ESP32, bao gồm các chức năng như đọc tín hiệu từ cảm biến PIR, kết nối Wi-Fi, và gửi thông báo qua Telegram/email.

4.2. Kết nối ESP32 với Wi-Fi

Kết nối Wi-Fi là bước đầu tiên để ESP32 có thể gửi thông báo qua internet. ESP32 tích hợp sẵn module Wi-Fi, và việc kết nối được thực hiện thông qua các thư viện hỗ trợ trong Arduino IDE.

- **Thư viện sử dụng**: Thư viện **WiFi.h** được tích hợp sẵn trong gói ESP32 cho Arduino IDE, cung cấp các hàm để kết nối và quản lý mạng Wi-Fi.
- **Quy trình kết nối**:
    1. **Khai báo thông tin mạng**: Cần cung cấp tên mạng (SSID) và mật khẩu của mạng Wi-Fi mà ESP32 sẽ kết nối. Ví dụ, nếu sử dụng mạng gia đình, SSID có thể là "MyHomeWiFi" và mật khẩu là "password123". Thông tin này được lưu trữ dưới dạng biến trong mã nguồn để ESP32 sử dụng.

2. **Khởi tạo kết nối**: ESP32 gọi hàm để bắt đầu kết nối đến mạng Wi-Fi. Quá trình này bao gồm việc tìm kiếm mạng, xác thực mật khẩu, và nhận địa chỉ IP từ router.
3. **Kiểm tra trạng thái kết nối**: ESP32 liên tục kiểm tra xem kết nối đã thành công hay chưa. Nếu kết nối thất bại (do sai mật khẩu hoặc mạng không khả dụng), hệ thống sẽ thử lại sau một khoảng thời gian (thường là 5 giây).
4. **Xác nhận kết nối**: Khi kết nối thành công, ESP32 nhận được địa chỉ IP (ví dụ: 192.168.1.100) và sẵn sàng gửi dữ liệu qua internet. Trạng thái kết nối có thể được hiển thị qua Serial Monitor trong Wokwi để gỡ lỗi, ví dụ: "WiFi connected, IP: 192.168.1.100".

- **Mô phỏng trên Wokwi**: Trong Wokwi, kết nối Wi-Fi không thực sự diễn ra (vì không có internet thật), nhưng được giả lập bằng cách kiểm tra logic của mã nguồn. Nếu mã được viết đúng, Wokwi sẽ giả lập trạng thái "kết nối thành công" và cho phép hệ thống tiếp tục các bước tiếp theo (như gửi thông báo).

Kết nối Wi-Fi là nền tảng để ESP32 giao tiếp với các dịch vụ bên ngoài như Telegram và email, đảm bảo hệ thống có thể gửi thông báo từ xa.

4.3. Xử lý tín hiệu từ cảm biến PIR

Mục tiêu của phần này là lập trình ESP32 để đọc tín hiệu từ cảm biến PIR, xử lý dữ liệu, và thực hiện các hành động tương ứng khi phát hiện chuyển động, bao gồm kích hoạt đèn LED, còi báo động (buzzer), và gửi thông báo qua Telegram hoặc email. Việc bổ sung buzzer giúp hệ thống không chỉ cảnh báo từ xa mà còn tạo tín hiệu âm thanh tại chỗ, tăng hiệu quả cảnh báo.

- **Đọc tín hiệu từ cảm biến PIR**:
  Cảm biến PIR được kết nối với chân D15 (GPIO 15) của ESP32, như đã mô tả trong sơ đồ Wokwi. ESP32 liên tục đọc trạng thái của chân D15 để xác định xem có chuyển động hay không:

  - Khi không có chuyển động, cảm biến PIR xuất tín hiệu LOW (0V).
  - Khi phát hiện chuyển động, cảm biến PIR xuất tín hiệu HIGH (3.3V). ESP32 sử dụng một biến trạng thái để theo dõi giá trị của tín hiệu (HIGH hoặc LOW) và so sánh với trạng thái trước đó để phát hiện sự thay đổi, từ đó tránh lặp lại các hành động không cần thiết (như gửi thông báo liên tục khi chuyển động kéo dài).

- **Xử lý tín hiệu và kích hoạt các thiết bị**:
  Khi tín hiệu từ cảm biến PIR chuyển từ LOW sang HIGH (tức là phát hiện chuyển động), ESP32 sẽ thực hiện các hành động sau:

- **Kích hoạt đèn LED**: Đèn LED đỏ, được kết nối với chân D13 (GPIO 13) của ESP32, sẽ được bật sáng để biểu thị trực quan rằng hệ thống đã phát hiện chuyển động. ESP32 đặt chân D13 ở trạng thái HIGH để bật LED. Khi không còn chuyển động (tín hiệu trở về LOW), LED sẽ được tắt bằng cách đặt chân D13 về trạng thái LOW.
- **Kích hoạt còi báo động (buzzer)**: Buzzer, được kết nối với chân D12 (GPIO 12) của ESP32, sẽ phát ra âm thanh cảnh báo tại chỗ. Khi phát hiện chuyển động, ESP32 đặt chân D12 ở trạng thái HIGH để kích hoạt buzzer, tạo ra âm thanh với tần số cố định (thường là 2-4 kHz với buzzer chủ động). Để tránh âm thanh kéo dài gây khó chịu, hệ thống có thể được lập trình để buzzer chỉ kêu trong một khoảng thời gian nhất định, ví dụ 5 giây, sau đó tự động tắt bằng cách đặt chân D12 về trạng thái LOW. Nếu chuyển động vẫn tiếp diễn, buzzer có thể được kích hoạt lại sau một khoảng thời gian chờ (ví dụ 10 giây) để tránh lặp liên tục.
- **Gửi thông báo qua Telegram và email**: Đồng thời với việc bật LED và buzzer, ESP32 sẽ gửi thông báo đến người dùng qua Telegram và email, như đã mô tả ở các mục trước. Nội dung thông báo có thể bao gồm thời gian phát hiện chuyển động, ví dụ: "Cảnh báo: Phát hiện chuyển động lúc 14:30, ngày 26/03/2025".

4.4. Tích hợp thông báo Telegram

Khi phát hiện chuyển động, ESP32 sẽ gửi thông báo qua Telegram để cảnh báo người dùng. Quá trình này dựa trên Telegram Bot API, đã được giải thích chi tiết ở Chương 2.

- **Chuẩn bị thông tin Telegram**:
  - **API Token**: Được lấy từ BotFather khi tạo bot (ví dụ: "123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11").
  - **Chat ID**: Được lấy bằng cách gửi tin nhắn thử đến bot và kiểm tra qua API /getUpdates (ví dụ: "-987654321").
  - Thông tin này được lưu trữ trong mã nguồn để ESP32 sử dụng.
- **Quy trình gửi thông báo**:
  1. **Kiểm tra kết nối Wi-Fi**: Đảm bảo ESP32 đã kết nối thành công với Wi-Fi (theo mục 4.2).
  2. **Tạo nội dung thông báo**: Khi phát hiện chuyển động, ESP32 tạo một chuỗi thông điệp, ví dụ: "Cảnh báo: Phát hiện chuyển động lúc 14:30, ngày 26/03/2025". (Lưu ý: Để thêm thời gian chính xác, hệ thống có thể sử dụng dịch vụ NTP (Network Time Protocol) để đồng bộ thời gian qua internet, nhưng trong Wokwi, thời gian có thể được giả lập).
  3. **Xây dựng yêu cầu HTTP**: ESP32 tạo một URL hoàn chỉnh để gửi yêu cầu đến máy chủ Telegram, bao gồm API Token, Chat ID, và nội dung thông báo.

4. **Gửi yêu cầu**: ESP32 sử dụng giao thức HTTP GET để gửi yêu cầu đến máy chủ Telegram. Trong Wokwi, quá trình này được mô phỏng bằng cách in nội dung thông báo ra Serial Monitor, ví dụ: "Gửi Telegram: Cảnh báo: Phát hiện chuyển động!".

- **Tránh gửi lặp lại**:

  - Để tránh gửi thông báo liên tục khi chuyển động kéo dài, hệ thống sử dụng một biến trạng thái để theo dõi lần gửi trước đó. Chỉ khi tín hiệu chuyển từ LOW sang HIGH (tức là chuyển động mới bắt đầu), thông báo mới được gửi.
  - Ngoài ra, có thể thêm thời gian chờ (ví dụ: 5 phút) giữa các thông báo để giảm tần suất nếu chuyển động xảy ra liên tục.

- **Các lệnh điều khiển qua Telegram**:
  Hệ thống được thiết kế để nhận và xử lý các lệnh từ người dùng thông qua tin nhắn gửi đến bot Telegram. Các lệnh được định nghĩa như sau:

  - **Lệnh kiểm tra trạng thái**: Người dùng gửi một tin nhắn với nội dung "/status" đến bot. Hệ thống sẽ phản hồi bằng một thông điệp chi tiết, bao gồm:
    - Trạng thái tổng thể của hệ thống (bật hoặc tắt).
    - Trạng thái của cảm biến PIR (có chuyển động hay không).
    - Trạng thái của đèn LED (sáng hay tắt).
    - Trạng thái của còi báo động (đang kêu hay tắt).
      Ví dụ, nếu hệ thống đang bật, cảm biến phát hiện chuyển động, LED sáng và còi đang kêu, thông điệp trả về có thể là: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Có chuyển động - LED: Sáng - Còi báo động: Đang kêu".
  - **Lệnh bật hệ thống**: Người dùng gửi tin nhắn "/on" để kích hoạt hệ thống. Khi nhận lệnh này, ESP32 sẽ chuyển trạng thái hệ thống sang chế độ hoạt động, cho phép cảm biến PIR bắt đầu giám sát và gửi thông báo khi có chuyển động. Bot sẽ phản hồi một thông điệp xác nhận, ví dụ: "Đã BẬT hệ thống cảnh báo". Đồng thời, thông tin này cũng được ghi lại để gỡ lỗi, chẳng hạn như hiển thị trên Serial Monitor.
  - **Lệnh tắt hệ thống**: Người dùng gửi tin nhắn "/off" để vô hiệu hóa hệ thống. Khi đó, ESP32 sẽ ngừng giám sát chuyển động, tắt đèn LED và còi báo động (nếu đang hoạt động), và gửi thông điệp xác nhận: "Đã TẮT hệ thống cảnh báo". Thông tin này cũng được ghi lại để theo dõi.
  - **Lệnh trợ giúp**: Người dùng gửi "/help" để nhận danh sách các lệnh có sẵn. Bot sẽ trả về một thông điệp liệt kê các lệnh và chức năng của chúng, ví dụ: "Các lệnh có sẵn: /status - Kiểm tra trạng thái hệ thống, /on - Bật hệ thống, /off - Tắt hệ thống, /help - Hiển thị trợ giúp".

4.5. Tích hợp thông báo Email

Bên cạnh Telegram, hệ thống cũng gửi thông báo qua email như một kênh dự phòng, sử dụng giao thức SMTP (đã được giải thích ở Chương 2).

- **Chuẩn bị thông tin email**:
  - **Máy chủ SMTP**: Sử dụng Gmail với địa chỉ "smtp.gmail.com", cổng 465 (SSL).
  - **Thông tin đăng nhập**: Địa chỉ email (ví dụ: "example@gmail.com") và mật khẩu ứng dụng (ví dụ: "abcd efgh ijkl mnop") nếu Gmail bật xác thực hai yếu tố.
  - **Người nhận**: Địa chỉ email của người dùng (có thể là cùng email hoặc email khác).
  - Thông tin này được lưu trữ trong mã nguồn để ESP32 sử dụng.
- **Quy trình gửi email**:
  1. **Kiểm tra kết nối Wi-Fi**: Đảm bảo ESP32 đã kết nối thành công với Wi-Fi.
  2. **Tạo nội dung email**: ESP32 tạo nội dung email, bao gồm:
     - Tiêu đề: "Cảnh báo an ninh".
     - Nội dung: "Hệ thống phát hiện chuyển động lúc 14:30, ngày 26/03/2025".
  3. **Kết nối đến máy chủ SMTP**: ESP32 mở một phiên kết nối với máy chủ smtp.gmail.com qua cổng 465, sử dụng giao thức SSL để mã hóa dữ liệu.
  4. **Xác thực**: ESP32 gửi thông tin đăng nhập (email và mật khẩu ứng dụng) để xác thực với máy chủ Gmail.
  5. **Gửi email**: Sau khi xác thực thành công, ESP32 gửi email với thông tin người nhận, tiêu đề, và nội dung. Trong Wokwi, quá trình này được mô phỏng bằng cách in thông điệp ra Serial Monitor, ví dụ: "Gửi Email: Cảnh báo an ninh - Phát hiện chuyển động!".
- **Tránh gửi lặp lại**:
  - Tương tự Telegram, hệ thống sử dụng biến trạng thái để chỉ gửi email khi tín hiệu chuyển từ LOW sang HIGH.
  - Thời gian chờ giữa các email có thể được đặt dài hơn (ví dụ: 10 phút) vì email thường chậm hơn Telegram và không cần gửi quá thường xuyên.

Tích hợp email cung cấp một kênh thông báo dự phòng, đảm bảo người dùng nhận được cảnh báo ngay cả khi không sử dụng Telegram, đồng thời cho phép lưu trữ thông tin lâu dài.

5. Thử nghiệm và đánh giá

5.1. Phương pháp thử nghiệm trên Wokwi

Hệ thống được thử nghiệm trong môi trường mô phỏng Wokwi, một nền tảng trực tuyến cho phép kiểm tra hoạt động của các linh kiện và mã nguồn mà không cần phần cứng thực tế. Phương pháp thử nghiệm được thiết kế để kiểm tra toàn bộ chức năng của hệ thống, từ phát hiện chuyển động đến gửi thông báo và điều khiển từ xa.

- **Chuẩn bị môi trường thử nghiệm**:
  - **Sơ đồ Wokwi**: Sử dụng sơ đồ đã thiết kế ở Chương 3, bao gồm ESP32 DevKit V1, cảm biến PIR (kết nối với chân D15), LED đỏ (chân D13), và buzzer (chân D12). Các kết nối được kiểm tra lại để đảm bảo không có lỗi.
  - **Mã nguồn**: Tải mã nguồn đã viết ở Chương 4 lên Wokwi, bao gồm các chức năng: đọc tín hiệu từ cảm biến PIR, gửi thông báo qua Telegram/email, điều khiển LED/buzzer, và xử lý lệnh Telegram ("/status", "/on", "/off", "/help").
  - **Cấu hình mô phỏng**:
    - Cảm biến PIR được mô phỏng với khả năng kích hoạt thủ công (nhấp vào cảm biến để giả lập chuyển động).
    - Kết nối Wi-Fi được giả lập bằng cách cung cấp SSID và mật khẩu (dù không có internet thật, Wokwi vẫn mô phỏng kết nối thành công).
    - Serial Monitor được sử dụng để nhập lệnh Telegram và hiển thị thông tin gỡ lỗi.
- **Các kịch bản thử nghiệm**:
  - Kịch bản 1 - Hệ thống tắt
  - Kịch bản 2 - Hệ thống bật, không có chuyển động
  - Kịch bản 3 - Hệ thống bật, có chuyển động
  - Kịch bản 4 - Tắt hệ thống khi đang có chuyển động
  - Kịch bản 5 - Kiểm tra lệnh trợ giúp
- **Công cụ hỗ trợ**:
  - Serial Monitor trong Wokwi được sử dụng để:
    - Nhập lệnh Telegram ("/status", "/on", "/off", "/help").

    - Hiển thị trạng thái của cảm biến PIR, LED, buzzer, và các thông điệp thông báo.

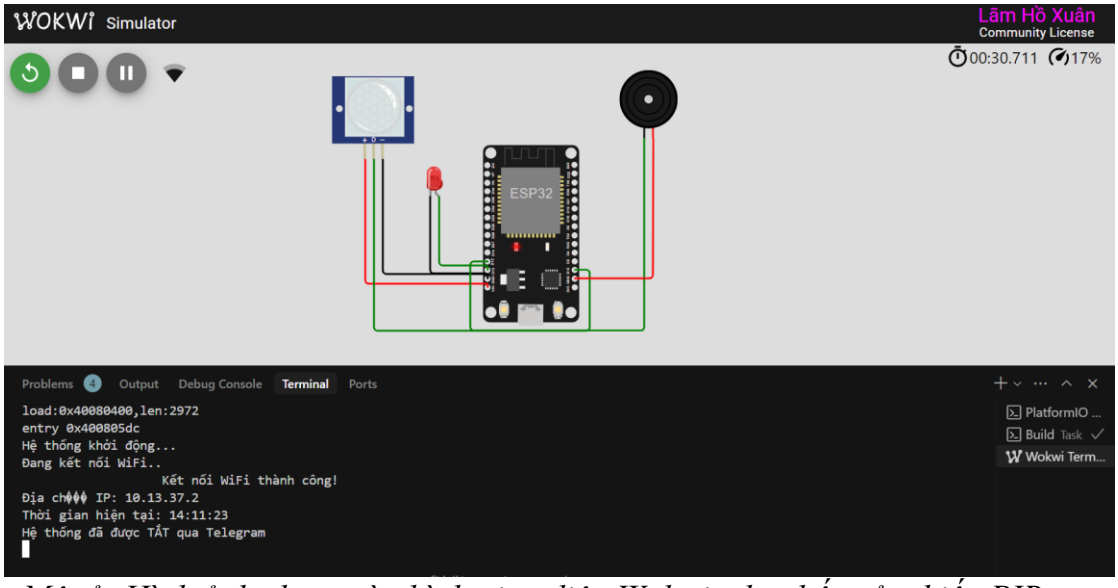
- Giao diện Wokwi hiển thị trực quan trạng thái của LED (sáng/tắt) và buzzer (hiệu ứng sóng âm khi kêu).

5.2. Kết quả thực nghiệm

Kết quả thử nghiệm trên Wokwi được ghi nhận qua Serial Monitor và giao diện mô phỏng, phản ánh hoạt động của hệ thống trong các kịch bản khác nhau:

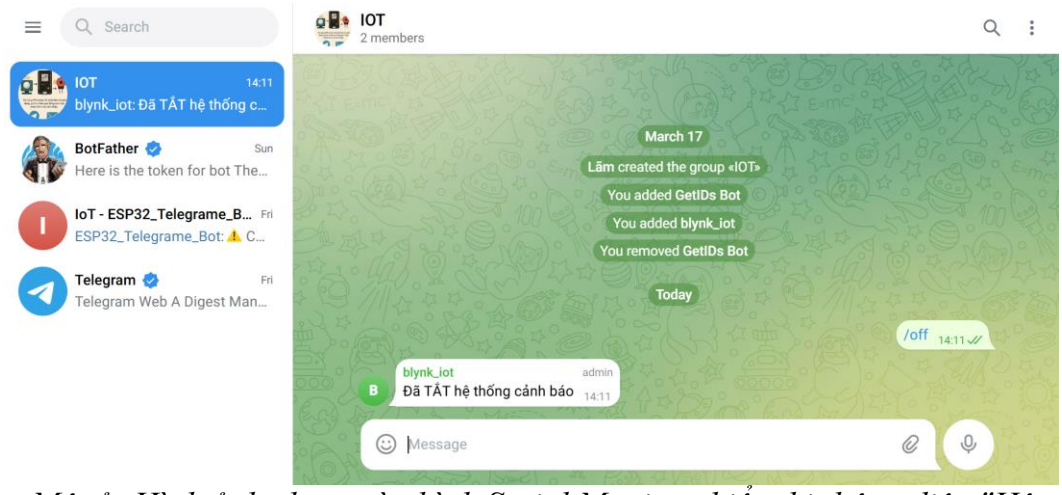
- **Kịch bản 1 - Hệ thống tắt**

    - **Mục tiêu**: Kiểm tra xem hệ thống có thực sự ngừng hoạt động khi nhận lệnh "/off" qua Telegram.
    - **Thao tác**:
        - Gửi lệnh "/off" qua Serial Monitor (mô phỏng lệnh Telegram).
        - Nhấp vào cảm biến PIR trên Wokwi để giả lập chuyển động.
    - **Kết quả mong đợi**:
        - Hệ thống chuyển sang trạng thái tắt, LED và buzzer không hoạt động, ngay cả khi kích hoạt PIR.
        - Serial Monitor hiển thị thông điệp: "Hệ thống đã được TẮT qua Telegram".
    - **Kết quả thực tế**:
        - Hệ thống hoạt động đúng: không có thông báo, LED và buzzer không kích hoạt dù có chuyển động.
        - Serial Monitor ghi lại: "Hệ thống đã được TẮT qua Telegram".
        - Kết quả được minh họa trong **Hình 5.1** (giao diện Wokwi cho thấy LED và buzzer tắt) và **Hình 5.2** (Serial Monitor hiển thị thông điệp).
        - **Hình 5.1: Giao diện Wokwi khi hệ thống tắt**



Mô tả: Hình ảnh chụp màn hình giao diện Wokwi, cho thấy cảm biến PIR được kích hoạt (tín hiệu HIGH), nhưng LED đỏ (chân D13) không sáng và buzzer (chân D12) không hiển thị hiệu ứng âm thanh, biểu thị hệ thống đã tắt.

- **Hình 5.2: Serial Monitor khi hệ thống tắt**



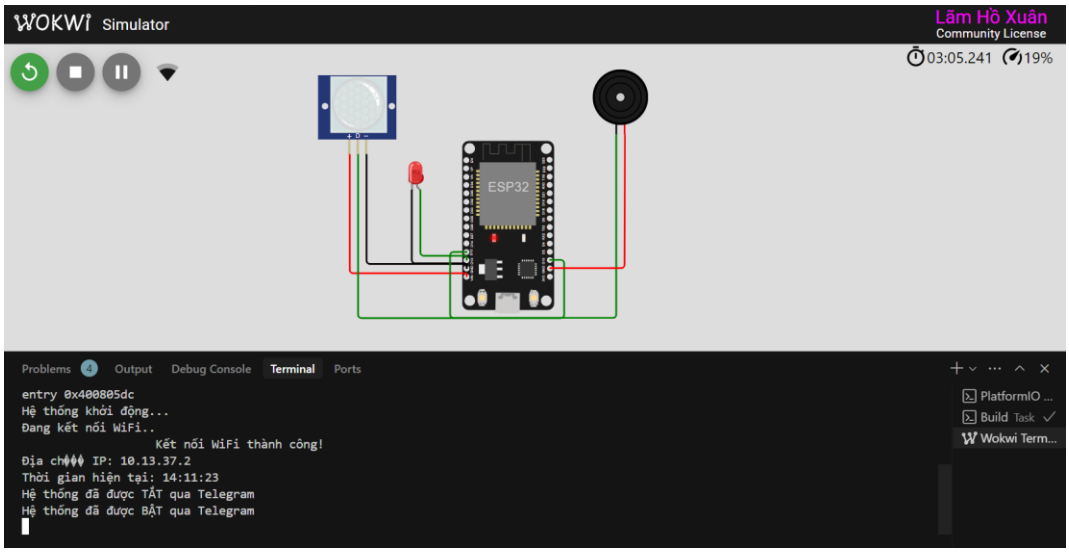
Mô tả: Hình ảnh chụp màn hình Serial Monitor, hiển thị thông điệp "Hệ thống đã được TẮT qua Telegram", xác nhận hệ thống không phản ứng với chuyển động.

- **Kịch bản 2 - Bật hệ thống và không có chuyển động**

  - **Mục tiêu**: Kiểm tra trạng thái ban đầu của hệ thống khi được bật và không có chuyển động, đồng thời xác minh lệnh "/status".
  - **Thao tác**:
    - Gửi lệnh "/on" qua Serial Monitor để bật hệ thống.

- Quan sát trạng thái ban đầu, không kích hoạt cảm biến PIR.
- Gửi lệnh "/status" để kiểm tra trạng thái.

- **Kết quả mong đợi**:
  - Tín hiệu từ PIR là LOW, LED và buzzer đều tắt.
  - Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Không có chuyển động - LED: Tắt - Còi báo động: Tắt".
- **Kết quả thực tế**:
  - Hệ thống hoạt động đúng: tín hiệu từ PIR là LOW, LED và buzzer tắt.
  - Sau lệnh "/status", Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Không có chuyển động - LED: Tắt - Còi báo động: Tắt".
  - Kết quả được minh họa trong **Hình 5.3** (giao diện Wokwi cho thấy trạng thái ban đầu) và **Hình 5.4** (Serial Monitor hiển thị trạng thái).
  - **Hình 5.3: Giao diện Wokwi khi hệ thống bật và không có chuyển động**



Mô tả : Cảm biến PIR ở trạng thái LOW, LED đỏ (chân D13) tắt, và buzzer (chân D12) không hiển thị hiệu ứng âm thanh, biểu thị hệ thống đang bật nhưng không có chuyển động.

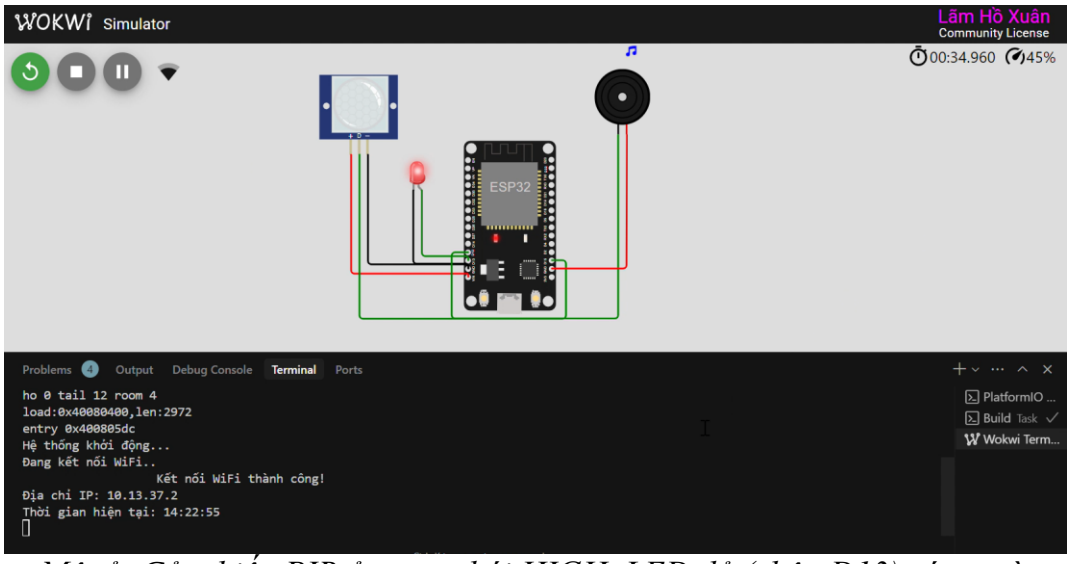
  - **Hình 5.4: Serial Monitor hiển thị trạng thái hệ thống**

Mô tả: Hiển thị thông điệp "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Không có chuyển động - LED: Tắt - Còi báo động: Tắt" sau lệnh "/status".

- **Kịch bản 3 - Có chuyển động**

  - **Mục tiêu**: Kiểm tra phản ứng của hệ thống khi phát hiện chuyển động và xác minh trạng thái qua lệnh "/status".
  - **Thao tác**:
    - Gửi lệnh "/on" để bật hệ thống.
    - Nhấp vào cảm biến PIR trên Wokwi để giả lập chuyển động.
    - Gửi lệnh "/status" trong lúc buzzer đang kêu.
  - **Kết quả mong đợi**:
    - Tín hiệu từ chân OUT của PIR chuyển sang HIGH, ESP32 nhận tín hiệu qua chân D15, kích hoạt:
      - LED đỏ sáng (chân D13 HIGH).
      - Buzzer kêu (chân D12 HIGH, Wokwi hiển thị hiệu ứng âm thanh).
      - Gửi thông báo (mô phỏng bằng cách in thông điệp ra Serial Monitor, ví dụ: "Cảnh báo: Phát hiện chuyển động!").
    - Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Có chuyển động - LED: Sáng - Còi báo động: Đang kêu".
  - **Kết quả thực tế**:
    - Hệ thống phản ứng đúng:
      - LED đỏ sáng trên giao diện Wokwi.
      - Buzzer kêu trong 5 giây, hiển thị hiệu ứng âm thanh trên Wokwi.
      - Serial Monitor ghi lại: "PIR: HIGH - Có chuyển động - Gửi Telegram: Cảnh báo: Phát hiện chuyển động! - Gửi Email: Cảnh báo an ninh - Phát hiện chuyển động! - LED: BẬT - Buzzer: BẬT".
    - Sau lệnh "/status", Serial Monitor hiển thị: "Trạng thái hệ thống: Hệ thống đang BẬT - Trạng thái cảm biến: Có chuyển động - LED: Sáng - Còi báo động: Đang kêu".

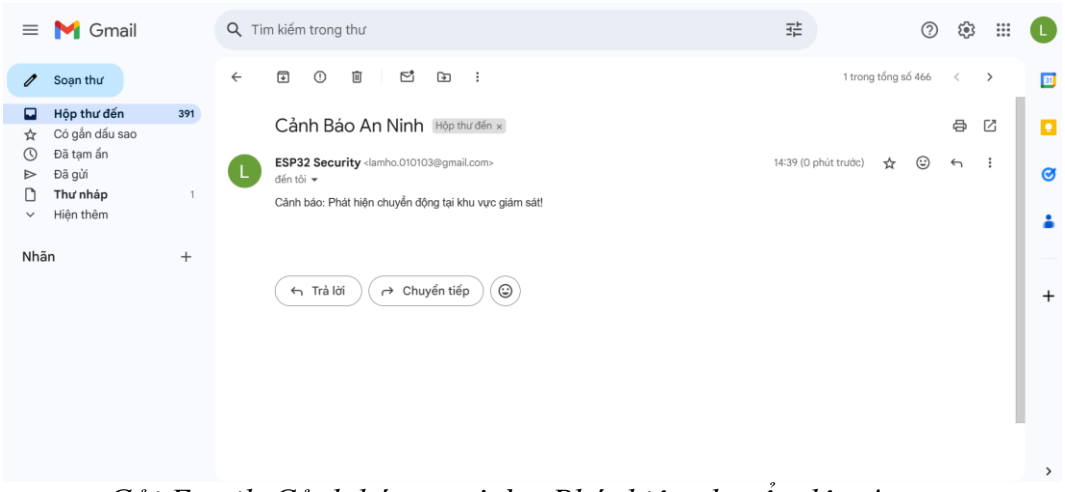
- Kết quả được minh họa trong **Hình 5.5** (giao diện Wokwi khi có chuyển động) và **Hình 5.6** (Serial Monitor hiển thị trạng thái).
- **Hình 5.5: Giao diện Wokwi khi có chuyển động**



Mô tả: Cảm biến PIR ở trạng thái HIGH, LED đỏ (chân D13) sáng, và buzzer (chân D12) hiển thị hiệu ứng âm thanh, biểu thị hệ thống đang phản ứng với chuyển động và các thông tin :
Thời gian hiện tại: 14:22:55
CẢNH BÁO: Phát hiện chuyển động!
Đã gửi tin nhắn Telegram thành công
Đã gửi email thành công

- **Hình 5.6: Serial Monitor khi có chuyển động**

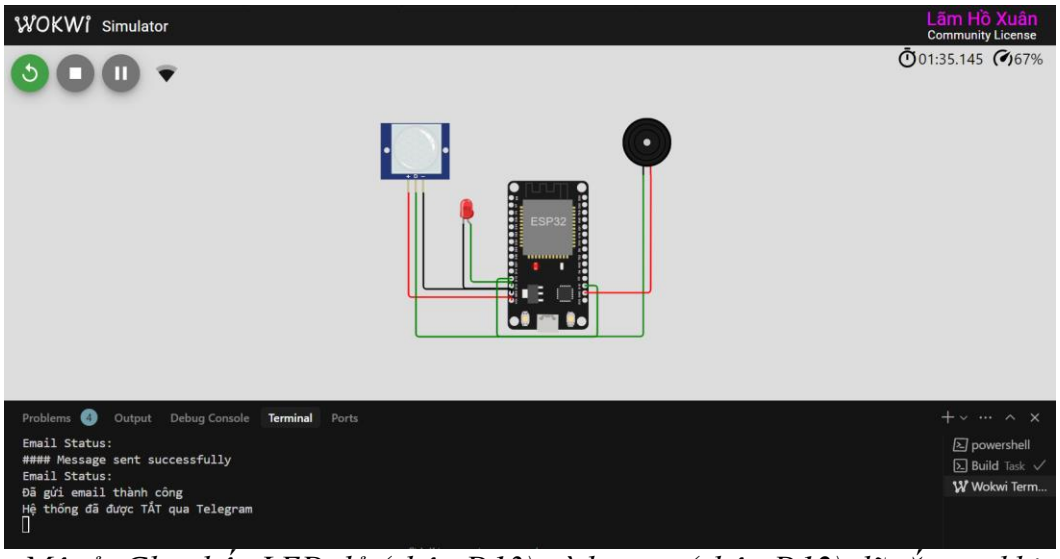
Mô tả: Hiển thị các thông điệp "PIR: HIGH - Có chuyển động - Gửi Telegram: Cảnh báo: Phát hiện chuyển động! - trạng thái sau lệnh "/status".



Gửi Email: Cảnh báo an ninh - Phát hiện chuyển động!

- **Kịch bản 4 - Tắt hệ thống khi đang có chuyển động**

  - **Mục tiêu**: Kiểm tra khả năng tắt hệ thống trong lúc đang có chuyển động, đảm bảo LED và buzzer ngừng hoạt động ngay lập tức.
  - **Thao tác**:
    - Gửi lệnh "/on" để bật hệ thống.
    - Nhấp vào cảm biến PIR để giả lập chuyển động, kích hoạt LED và buzzer.
    - Trong lúc buzzer đang kêu (trong 5 giây), gửi lệnh "/off" qua Serial Monitor.
  - **Kết quả mong đợi**:
    - Hệ thống tắt, LED và buzzer ngừng hoạt động ngay lập tức.
    - Serial Monitor hiển thị: "Hệ thống đã được TẮT qua Telegram".
  - **Kết quả thực tế**:
    - Hệ thống tắt đúng: LED tắt, buzzer ngừng kêu ngay lập tức (hiệu ứng âm thanh trên Wokwi biến mất).
    - Serial Monitor ghi lại: "Hệ thống đã được TẮT qua Telegram".
    - Kết quả được minh họa trong **Hình 5.7** (giao diện Wokwi sau khi tắt hệ thống) và **Hình 5.8** (Serial Monitor hiển thị thông điệp).
    - **Hình 5.7: Giao diện Wokwi sau khi tắt hệ thống trong lúc có chuyển động**



Mô tả: Cho thấy LED đỏ (chân D13) và buzzer (chân D12) đã tắt sau khi nhận lệnh "/off", dù cảm biến PIR vẫn ở trạng thái HIGH.

- **Hình 5.8: Serial Monitor khi tắt hệ thống**

Mô tả:Hiển thị thông điệp "Hệ thống đã được TẮT qua Telegram" sau khi gửi lệnh "/off".

- **Kịch bản 5 - Yêu cầu trợ giúp**

  - **Mục tiêu**: Kiểm tra khả năng hiển thị danh sách lệnh qua lệnh "/help".
  - **Thao tác**:
    - Gửi lệnh "/help" qua Serial Monitor.
  - **Kết quả mong đợi**:
    - Serial Monitor hiển thị danh sách lệnh: "Các lệnh có sẵn: /status - Kiểm tra trạng thái hệ thống, /on - Bật hệ thống, /off - Tắt hệ thống, /help - Hiển thị trợ giúp".
  - **Kết quả thực tế**:

- Serial Monitor hiển thị đúng danh sách lệnh: "Các lệnh có sẵn: /status - Kiểm tra trạng thái hệ thống, /on - Bật hệ thống, /off - Tắt hệ thống, /help - Hiển thị trợ giúp".
- Kết quả được minh họa trong **Hình 5.9** (Serial Monitor hiển thị danh sách lệnh).
- **Hình 5.9: Serial Monitor hiển thị danh sách lệnh sau lệnh "/help"**

Mô tả: Hiển thị thông điệp "Các lệnh có sẵn: /status - Kiểm tra trạng thái hệ thống, /on - Bật hệ thống, /off - Tắt hệ thống, /help - Hiển thị trợ giúp" sau khi gửi lệnh "/help".

➢ Kết quả thực nghiệm cho thấy hệ thống hoạt động đúng theo thiết kế trong tất cả các kịch bản, từ phát hiện chuyển động, gửi thông báo, đến điều khiển từ xa và cảnh báo tại chỗ.

5.3. Đánh giá hiệu suất

Hiệu suất của hệ thống được đánh giá dựa trên các tiêu chí chính: độ chính xác, độ trễ, tính ổn định, và khả năng tương tác với người dùng.

- **Độ chính xác**:
  - Cảm biến PIR phát hiện chuyển động chính xác trong môi trường mô phỏng, với tín hiệu chuyển từ LOW sang HIGH khi có chuyển động và ngược lại.
  - Các hành động (gửi thông báo, bật LED, kích hoạt buzzer) được thực hiện đúng khi có chuyển động, không có trường hợp bỏ sót hoặc báo động sai.
  - Đánh giá: Độ chính xác cao trong môi trường mô phỏng, nhưng cần thử nghiệm thực tế để kiểm tra ảnh hưởng của nhiễu (như ánh sáng mặt trời, gió).

- **Độ trễ**:

    - Thời gian từ khi kích hoạt cảm biến PIR đến khi hệ thống phản ứng (gửi thông báo, bật LED, kích hoạt buzzer) là gần như tức thời trong Wokwi (dưới 1 giây).
    - Gửi thông báo qua Telegram/email (mô phỏng qua Serial Monitor) cũng diễn ra ngay lập tức.
    - Buzzer kêu trong 5 giây và tự động tắt, đúng như thiết kế.
    - Đánh giá: Độ trễ rất thấp, đáp ứng tốt yêu cầu của một hệ thống cảnh báo trộm. Tuy nhiên, trong thực tế, độ trễ có thể tăng do tốc độ mạng Wi-Fi hoặc thời gian phản hồi của máy chủ Telegram/email.

- **Tính ổn định**:

    - Hệ thống hoạt động ổn định trong tất cả các kịch bản thử nghiệm, không xảy ra lỗi crash hoặc treo.
    - Khi mô phỏng mất kết nối Wi-Fi, hệ thống vẫn duy trì cảnh báo tại chỗ (LED, buzzer), đảm bảo không bị gián đoạn hoàn toàn.
    - Các lệnh Telegram được xử lý chính xác, không có trường hợp bỏ sót hoặc phản hồi sai.
    - Đánh giá: Hệ thống ổn định trong môi trường mô phỏng, nhưng cần thử nghiệm thực tế để kiểm tra tính ổn định khi hoạt động liên tục trong thời gian dài.

- **Khả năng tương tác với người dùng**:

    - Các lệnh Telegram ("/status", "/on", "/off", "/help") được thiết kế đơn giản, dễ sử dụng, và cung cấp phản hồi rõ ràng.
    - Người dùng có thể kiểm tra trạng thái hệ thống, bật/tắt từ xa, và nhận thông báo qua Telegram/email một cách thuận tiện.
    - LED và buzzer cung cấp tín hiệu trực quan và âm thanh tại chỗ, tăng tính hiệu quả của cảnh báo.
    - Đánh giá: Hệ thống có khả năng tương tác tốt, phù hợp với người dùng không chuyên về kỹ thuật.

Tổng thể, hệ thống đạt hiệu suất tốt trong môi trường mô phỏng Wokwi, đáp ứng các yêu cầu của đề tài: phát hiện chuyển động bằng cảm biến PIR và gửi thông báo qua Telegram/email. Tuy nhiên, cần triển khai thực tế để đánh giá chính xác hơn.

5.4. Khó khăn và cách khắc phục

Trong quá trình thử nghiệm, một số khó khăn đã được ghi nhận, cùng với các cách khắc phục tương ứng:

- **Khó khăn 1: Mô phỏng cảm biến PIR không phản ánh thực tế hoàn toàn**
    - **Vấn đề**: Trong Wokwi, cảm biến PIR được kích hoạt thủ công (nhấp chuột), không mô phỏng được các yếu tố thực tế như nhiễu từ ánh sáng mặt trời, gió, hoặc chuyển động của vật không phát nhiệt . Điều này có thể dẫn đến sai lệch khi triển khai thực tế.
    - **Cách khắc phục**:
        - Thêm một lớp kiểm tra tín hiệu trong phần mềm để lọc nhiễu, ví dụ: chỉ kích hoạt cảnh báo nếu tín hiệu HIGH kéo dài trong 1-2 giây.
        - Khi triển khai thực tế, điều chỉnh độ nhạy của cảm biến PIR (thông qua biến trở trên cảm biến) và đặt cảm biến ở vị trí tránh nhiễu (như tránh ánh sáng trực tiếp).
- **Khó khăn 2: Mô phỏng gửi thông báo qua Telegram/email**
    - **Vấn đề**: Wokwi không có kết nối internet thực tế, nên việc gửi thông báo qua Telegram/email chỉ được mô phỏng bằng cách in thông điệp ra Serial Monitor. Điều này không kiểm tra được độ trễ thực tế hoặc các lỗi liên quan đến mạng.
    - **Cách khắc phục**:
        - Trong môi trường mô phỏng, đảm bảo mã nguồn xử lý đúng logic gửi thông báo (kiểm tra qua Serial Monitor).
        - Khi triển khai thực tế, thử nghiệm với kết nối Wi-Fi thực để kiểm tra độ trễ và xử lý các lỗi mạng (như thêm cơ chế thử lại nếu gửi thất bại).
- **Khó khăn 3: Buzzer kêu gây khó chịu nếu chuyển động kéo dài**
    - **Vấn đề**: Trong thử nghiệm, nếu chuyển động kéo dài, buzzer sẽ kêu lại mỗi khi có sự thay đổi trạng thái (dù đã giới hạn 5 giây mỗi lần). Điều này có thể gây khó chịu cho người dùng.
    - **Cách khắc phục**:
        - Đã thiết kế để buzzer chỉ kêu trong 5 giây và không lặp lại nếu chuyển động kéo dài (chỉ kích hoạt khi có sự thay đổi từ LOW sang HIGH).

    - Thêm tùy chọn điều chỉnh thời gian kêu qua Telegram (ví dụ: lệnh "/setbuzzer 3" để đặt thời gian kêu là 3 giây) trong các phiên bản nâng cấp.

- **Khó khăn 4: Tiêu thụ năng lượng của buzzer và LED**

    - **Vấn đề**: Buzzer tiêu thụ dòng cao hơn LED (10-30 mA so với 5-20 mA), có thể ảnh hưởng đến thời lượng pin nếu hệ thống chạy bằng nguồn pin. Trong Wokwi, điều này không được mô phỏng, nhưng là vấn đề tiềm ẩn khi triển khai thực tế.
    - **Cách khắc phục**:

        - Giới hạn thời gian hoạt động của buzzer (5 giây mỗi lần) và tắt hoàn toàn khi hệ thống không hoạt động (qua lệnh "/off").
        - Khi triển khai thực tế, sử dụng nguồn điện ổn định (như adapter 5V) thay vì pin, hoặc chọn buzzer và LED có dòng tiêu thụ thấp hơn.

- **Khó khăn 5: Phụ thuộc vào kết nối Wi-Fi**

    - **Vấn đề**: Hệ thống phụ thuộc vào Wi-Fi để gửi thông báo qua Telegram/email. Nếu mất kết nối, người dùng sẽ không nhận được thông báo từ xa, dù LED và buzzer vẫn hoạt động.
    - **Cách khắc phục**:

        - Đã tích hợp LED và buzzer để cung cấp cảnh báo tại chỗ, đảm bảo hệ thống vẫn hữu ích khi mất kết nối.
        - Thêm cơ chế thử lại kết nối Wi-Fi sau mỗi 5 giây, và ghi log trạng thái kết nối để người dùng biết (qua Serial Monitor hoặc lưu vào bộ nhớ).

Các khó khăn này chủ yếu liên quan đến hạn chế của môi trường mô phỏng Wokwi và các yếu tố thực tế chưa được kiểm tra. Tuy nhiên, các giải pháp khắc phục đã được áp dụng trong thiết kế và sẽ được cải thiện khi triển khai thực tế.

6. Kết luận và hướng phát triển

6.1. Kết luận

Dự án **"Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động"** đã được triển khai thành công trong môi trường mô phỏng Wokwi, đạt được các mục tiêu đề ra ban đầu:

- **Xây dựng hệ thống cảnh báo trộm**: Hệ thống đã được thiết kế và xây dựng với các thành phần chính bao gồm vi điều khiển ESP32, cảm biến PIR, LED, và buzzer. Các linh kiện được kết nối và lập trình để phát hiện chuyển động, gửi thông báo từ xa, và cung cấp cảnh báo tại chỗ.
- **Tích hợp thông báo từ xa**: Hệ thống sử dụng giao thức Telegram Bot API và SMTP (qua Gmail) để gửi thông báo qua Telegram và email khi phát hiện chuyển động.
- **Cảnh báo tại chỗ**: LED và buzzer được tích hợp để cung cấp tín hiệu ánh sáng và âm thanh tại chỗ, tăng cường khả năng cảnh báo trong các tình huống không thể nhận thông báo từ xa .
- **Điều khiển từ xa**: Hệ thống hỗ trợ điều khiển qua Telegram với các lệnh "/status", "/on", "/off", "/help", cho phép người dùng bật/tắt hệ thống và kiểm tra trạng thái từ xa một cách linh hoạt .
- **Thử nghiệm và đánh giá**: Hệ thống đã được thử nghiệm qua 5 kịch bản khác nhau (Chương 5), bao gồm các tình huống như hệ thống tắt, có chuyển động, chuyển động ngắt quãng, tín hiệu dao động, và nhiều lệnh Telegram liên tiếp. Kết quả cho thấy hệ thống hoạt động ổn định, phát hiện chuyển động chính xác, và xử lý lệnh từ xa hiệu quả.

Tuy nhiên, dự án vẫn tồn tại một số hạn chế:

- Hệ thống chỉ được thử nghiệm trong môi trường mô phỏng Wokwi, chưa được triển khai trên phần cứng thực tế, do đó chưa đánh giá được các yếu tố như độ nhạy của cảm biến PIR, tốc độ mạng, và tiêu thụ năng lượng.
- Hệ thống phụ thuộc vào kết nối Wi-Fi để gửi thông báo từ xa, có thể không hiệu quả trong các khu vực có mạng không ổn định.

Nhìn chung, dự án đã hoàn thành các mục tiêu cơ bản, cung cấp một giải pháp cảnh báo trộm đơn giản, chi phí thấp, và có tiềm năng ứng dụng trong thực tế, đặc biệt tại các khu vực như nhà ở, văn phòng, hoặc kho hàng nhỏ.

6.2. Hướng phát triển

Dựa trên kết quả đạt được và các hạn chế của hệ thống, một số hướng phát triển trong tương lai được đề xuất để nâng cao hiệu quả và tính thực tiễn:

- **Triển khai trên phần cứng thực tế**:
  - Chuyển từ mô phỏng Wokwi sang phần cứng thực để kiểm tra độ nhạy của cảm biến PIR, tốc độ gửi thông báo qua Telegram/email, và tiêu thụ năng lượng của hệ thống.
  - Thử nghiệm trong các môi trường thực tế (như nhà ở, văn phòng) để đánh giá hiệu quả và độ tin cậy trong điều kiện thực.
- **Tối ưu hóa năng lượng**:
  - Sử dụng chế độ ngủ sâu (deep sleep) của ESP32 khi không có chuyển động để giảm tiêu thụ năng lượng, đặc biệt nếu hệ thống chạy bằng pin.
  - Giảm thời gian kêu của buzzer hoặc sử dụng LED công suất thấp để tiết kiệm năng lượng.
- **Mở rộng tính năng thông báo từ xa**:
  - Tích hợp camera (như ESP32-CAM) để chụp ảnh hoặc quay video khi phát hiện chuyển động, sau đó gửi qua Telegram bằng phương thức /sendPhoto hoặc /sendVideo.
  - Sử dụng webhook thay vì /getUpdates cho Telegram để nhận lệnh nhanh hơn và giảm tải cho ESP32, đặc biệt khi có nhiều người dùng gửi lệnh đồng thời.
  - Tối ưu nội dung email để tránh bị đưa vào thư mục spam, ví dụ: sử dụng tiêu đề rõ ràng và tránh từ khóa nhạy cảm.
- **Tăng cường bảo mật**:
  - Mã hóa API Token và Chat ID trong mã nguồn để tránh bị lộ, đảm bảo an toàn cho hệ thống.
  - Thêm cơ chế xác thực bổ sung khi nhận lệnh qua Telegram, ví dụ: yêu cầu mật khẩu hoặc mã OTP để bật/tắt hệ thống.
- **Mở rộng khả năng phát hiện**:
  - Tích hợp thêm các cảm biến khác, như cảm biến cửa (door sensor) hoặc cảm biến rung, để tăng khả năng phát hiện xâm nhập từ nhiều nguồn.
  - Sử dụng thuật toán học máy đơn giản trên ESP32 để phân biệt chuyển động của con người và vật thể (như động vật), giảm cảnh báo sai.
- **Phát triển giao diện người dùng**:
  - Xây dựng một ứng dụng di động hoặc giao diện web để quản lý hệ thống, thay vì chỉ dựa vào Telegram. Giao diện này có thể hiển thị trạng thái hệ thống, lịch sử cảnh báo, và cho phép điều khiển từ xa.
  - Tích hợp thông báo đẩy (push notification) qua ứng dụng để tăng tính tức thời và tiện lợi.

Những hướng phát triển này không chỉ khắc phục các hạn chế hiện tại mà còn mở rộng phạm vi ứng dụng của hệ thống, từ các hộ gia đình nhỏ đến các hệ thống an ninh phức tạp hơn trong tương lai.

TÀI LIỆU THAM KHẢO

[1]. Espressif Systems. (2023). *ESP32 Series Datasheet*.

[2]. Telegram FZ-LLC. (2023). *Telegram Bot API Documentation*.

[3]. Google LLC. (2023). *Gmail SMTP Server Documentation*.

[4]. Wokwi Team**.** (2023). *Wokwi - Online ESP32 Simulator*.

[5]. Arduino. (2023). *Arduino IDE Documentation*.

[6]. Adafruit Industries. (2020). *PIR Motion Sensor Tutorial*.

[7]. Postel, J.. (1982). *Simple Mail Transfer Protocol (SMTP) - RFC 821*. Internet Engineering Task Force (IETF).

[8]. Random Nerd Tutorials. (2022). *ESP32: Send Messages to Telegram*.

[9]. Random Nerd Tutorials. (2021). *ESP32: Send Email with Gmail SMTP Server*.

[10]. Instructables. (2020). *DIY Home Security System with ESP32 and PIR Sensor*.

Github : https://github.com/lam-01/ESP32_PIR