

LỜI CẢM ƠN

Trong suốt quá trình thực hiện tiểu luận với đề tài “Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động”, em xin chân thành gửi lời cảm ơn sâu sắc đến thầy/cô giảng viên bộ môn Phát triển ứng dụng IoT đã tận tình hướng dẫn, truyền đạt kiến thức và tạo điều kiện để em có thể tiếp cận với các công nghệ mới trong lĩnh vực IoT.

Nhờ vào những buổi học lý thuyết kết hợp với thực hành mô phỏng, em đã có cơ hội áp dụng kiến thức vào việc xây dựng một hệ thống thực tế, đồng thời hiểu rõ hơn về cách hoạt động của các thiết bị như ESP32 và cảm biến PIR.

Do giới hạn về thời gian và kinh nghiệm, chắc chắn bài làm không tránh khỏi những thiếu sót. Em rất mong nhận được những góp ý quý báu từ thầy/cô để có thể hoàn thiện hơn trong những lần nghiên cứu sau.

Em xin chân thành cảm ơn!

MỤC LỤC

PHẦN MỞ ĐẦU	1
I. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU	1
1.1 Đặt vấn đề	1
1.2 Mục tiêu đề tài.....	1
1.3 Ứng dụng thực tế	2
PHẦN NỘI DUNG	3
II. CƠ SỞ LÝ THUYẾT	3
2.1 Giới thiệu về vi điều khiển ESP32	3
2.2 Cảm biến chuyển động PIR – Cấu tạo và nguyên lý hoạt động	4
2.3 Giao tiếp giữa ESP32 và cảm biến PIR.....	6
2.4 Giao tiếp giữa ESP32 và Telegram thông qua Internet	8
2.5 Giới thiệu nền tảng mô phỏng Wokwi và phần mềm PlatformIO	10
III. NGUYÊN LÝ HOẠT ĐỘNG	12
3.1 Tổng quan hệ thống cảnh báo trộm.....	12
3.2 Sơ đồ khối hệ thống.....	13
3.3 Mô tả nguyên lý hoạt động	14
3.4 Luồng hoạt động và phản hồi khi phát hiện chuyển động.....	17
IV. THIẾT KẾ HỆ THỐNG	19
4.1 Danh sách linh kiện mô phỏng	19
4.2 Kết nối phần cứng trên Wokwi	22
V. LẬP TRÌNH VÀ TRIỂN KHAI	24
5.1 Mô tả thuật toán và logic xử lý.....	24
5.2 Lập trình kết nối WIFI và Telegram Bot	27
5.3 Lập trình phát hiện chuyển động từ cảm biến PIR	31
5.4 Gửi cảnh báo lên Telegram khi có xâm nhập	34
VI. KẾT QUẢ VÀ ĐÁNH GIÁ	38
6.1 Kết quả mô phỏng thực tế	38
6.2 Đánh giá hiệu quả và tính ứng dụng của hệ thống	41
PHẦN KẾT LUẬN	48
VII. KẾT LUẬN	48
7.1 Tóm tắt kết quả đạt được	48
7.2 Định hướng phát triển hệ thống trong tương lai	49
PHẦN ĐÁNH GIÁ	53

PHẦN MỞ ĐẦU

I. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU

1.1 Đặt vấn đề

Trong bối cảnh xã hội hiện đại, vấn đề an ninh và bảo vệ tài sản luôn là mối quan tâm hàng đầu của mỗi cá nhân và tổ chức. Theo thống kê của Bộ Công an Việt Nam, số vụ trộm cắp tài sản tại các khu dân cư vẫn chiếm tỷ lệ cao trong cơ cấu tội phạm hình sự. Đặc biệt, các vụ đột nhập trái phép vào nhà dân thường xảy ra khi chủ nhà vắng mặt hoặc trong thời gian nghỉ lễ, tết. Điều này đặt ra yêu cầu cấp thiết về việc xây dựng các hệ thống cảnh báo an ninh hiệu quả, có khả năng phát hiện sớm các hành vi xâm nhập trái phép và thông báo kịp thời cho chủ sở hữu.

Với sự phát triển mạnh mẽ của công nghệ Internet of Things (IoT), các giải pháp an ninh thông minh ngày càng trở nên phổ biến và dễ tiếp cận. Các hệ thống cảnh báo trộm truyền thống thường có chi phí cao, yêu cầu lắp đặt phức tạp và khó tích hợp với các thiết bị thông minh khác. Trong khi đó, các giải pháp IoT mang lại nhiều ưu điểm vượt trội như chi phí thấp, dễ dàng triển khai, khả năng tùy biến cao và tích hợp linh hoạt với các nền tảng thông báo hiện đại.

Đề tài "Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động" được đề xuất nhằm nghiên cứu và phát triển một giải pháp an ninh thông minh, hiệu quả và tiết kiệm chi phí, sử dụng vi điều khiển ESP32 kết hợp với cảm biến chuyển động PIR (Passive Infrared) và nền tảng nhắn tin Telegram. Hệ thống này hướng đến việc tạo ra một giải pháp an ninh toàn diện, dễ triển khai và phù hợp với điều kiện thực tế tại Việt Nam

1.2 Mục tiêu đề tài

Đề tài nghiên cứu hướng đến việc đạt được các mục tiêu cụ thể sau:

- **Thiết kế và phát triển** một hệ thống cảnh báo trộm hoàn chỉnh dựa trên nền tảng

ESP32 và cảm biến chuyển động PIR, có khả năng phát hiện chính xác sự xâm nhập trái phép.

- **Xây dựng cơ chế thông báo thông minh** thông qua nền tảng Telegram, giúp người

dùng nhận được cảnh báo tức thời khi có sự xâm nhập, bất kể họ đang ở đâu, miễn

là có kết nối internet.

- **Mô phỏng và kiểm thử** hệ thống trên nền tảng Wokwi, đảm bảo tính ổn định và độ

tin cậy trước khi triển khai thực tế.

- **Tối ưu hóa mã nguồn** để đảm bảo hệ thống hoạt động hiệu quả, tiết kiệm năng lượng và có khả năng mở rộng trong tương lai.
- **Tạo tài liệu hướng dẫn chi tiết** về cách thiết kế, lập trình và triển khai hệ thống, giúp người dùng có thể tự xây dựng hệ thống tương tự.

1.3 Ứng dụng thực tế

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động có nhiều ứng dụng thực tế trong đời sống:

- **Bảo vệ nhà ở và căn hộ:** Hệ thống có thể được lắp đặt tại các vị trí trọng yếu như cửa ra vào, cửa sổ, hoặc khu vực có tài sản giá trị, giúp phát hiện và cảnh báo kịp thời khi có người đột nhập trái phép.
- **Bảo vệ văn phòng và cơ sở kinh doanh:** Đối với các văn phòng nhỏ hoặc cửa hàng, hệ thống này là giải pháp an ninh hiệu quả với chi phí hợp lý, đặc biệt là trong thời gian nghỉ hoặc ngoài giờ làm việc.
- **Giám sát kho hàng và khu vực hạn chế:** Hệ thống có thể được triển khai để giám sát các khu vực hạn chế, kho hàng, hoặc phòng máy chủ, đảm bảo chỉ những người có thẩm quyền mới được phép tiếp cận.
- **Hỗ trợ người già và người khuyết tật:** Hệ thống có thể được điều chỉnh để giám sát hoạt động của người già hoặc người khuyết tật sống một mình, gửi thông báo cho người thân khi phát hiện hoạt động bất thường hoặc không có hoạt động trong thời gian dài.
- **Tích hợp vào hệ thống nhà thông minh:** Hệ thống này có thể dễ dàng tích hợp vào các hệ thống nhà thông minh hiện có, bổ sung thêm lớp bảo vệ an ninh và tự động hóa các hành động như bật đèn, kích hoạt camera giám sát khi phát hiện chuyển động.

Với chi phí thấp, tính linh hoạt cao và khả năng tùy biến, hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR là một giải pháp an ninh hiệu quả, phù hợp với nhiều đối tượng sử dụng, từ hộ gia đình đến doanh nghiệp nhỏ và vừa tại Việt Nam

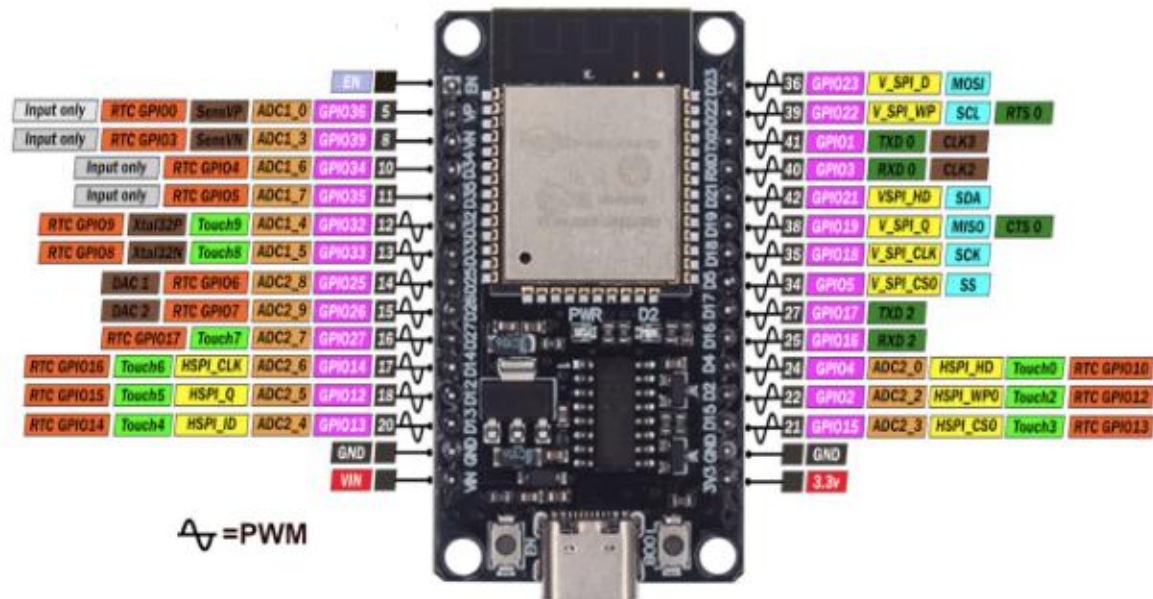
PHẦN NỘI DUNG

II. CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu về vi điều khiển ESP32

ESP32 là một hệ thống vi điều khiển (System on Chip - SoC) được phát triển bởi công ty Espressif Systems, ra mắt vào năm 2016 như phiên bản kế nhiệm của ESP8266.

ESP32 đã nhanh chóng trở thành một trong những nền tảng phổ biến nhất trong lĩnh vực IoT nhờ vào hiệu năng cao, tính năng phong phú và giá thành hợp lý.



Hình 1. Hình ảnh sơ đồ chân kết nối ESP32.

Đặc điểm kỹ thuật của ESP32:

- **Bộ xử lý:** ESP32 được trang bị bộ vi xử lý Tensilica Xtensa LX6 dual-core hoạt động ở tần số lên đến 240MHz, mang lại hiệu năng xử lý mạnh mẽ cho các ứng dụng IoT.
- **Bộ nhớ:** Chip tích hợp 520KB SRAM và hỗ trợ bộ nhớ ngoài lên đến 4MB Flash, đủ để lưu trữ mã nguồn phức tạp và dữ liệu ứng dụng.
- **Kết nối không dây:** ESP32 tích hợp cả Wi-Fi (802.11 b/g/n) và Bluetooth (cả Classic và BLE - Bluetooth Low Energy), cho phép kết nối linh hoạt với nhiều loại thiết bị và mạng khác nhau.

- **GPIO và giao diện:** ESP32 cung cấp đến 36 chân GPIO (General Purpose Input/ Output) và hỗ trợ nhiều giao thức giao tiếp như I2C, SPI, UART, I2S, CAN, v.v., giúp dễ dàng kết nối với các cảm biến và thiết bị ngoại vi.
- **Tính năng đặc biệt:** ESP32 còn tích hợp các tính năng như bộ chuyển đổi ADC 12- bit, cảm biến nhiệt độ, bộ điều khiển cảm ứng điện dung, và bộ mã hóa Hall, mở rộng khả năng ứng dụng trong nhiều lĩnh vực.
- **Tiết kiệm năng lượng:** ESP32 hỗ trợ nhiều chế độ tiết kiệm năng lượng, với mức tiêu thụ điện năng thấp đến 10 μ A trong chế độ deep sleep, phù hợp cho các ứng dụng chạy bằng pin.

Ưu điểm của ESP32 trong ứng dụng IoT:

- **Hiệu năng cao:** Với bộ xử lý dual-core, ESP32 có thể xử lý đồng thời nhiều tác vụ phức tạp như kết nối mạng, xử lý dữ liệu cảm biến và điều khiển thiết bị.
- **Tính linh hoạt:** Hỗ trợ đa dạng giao thức kết nối và giao tiếp, giúp ESP32 dễ dàng tích hợp vào các hệ thống IoT hiện có.
- **Cộng đồng phát triển lớn:** ESP32 có cộng đồng người dùng và nhà phát triển đông

đảo, với nhiều thư viện và tài liệu hướng dẫn sẵn có, giúp rút ngắn thời gian phát triển ứng dụng.

- **Chi phí hợp lý:** So với các vi điều khiển có tính năng tương đương, ESP32 có giá thành thấp hơn đáng kể, phù hợp cho cả dự án cá nhân và thương mại.

Trong hệ thống cảnh báo trộm của đề tài này, ESP32 đóng vai trò là bộ não trung tâm, chịu trách nhiệm xử lý tín hiệu từ cảm biến PIR, kết nối với mạng Wi-Fi và gửi thông báo đến người dùng thông qua Telegram khi phát hiện chuyển động.

2.2 Cảm biến chuyển động PIR – Cấu tạo và nguyên lý hoạt động

Cảm biến PIR (Passive Infrared) là một thiết bị điện tử được sử dụng rộng rãi để phát hiện chuyển động của con người hoặc động vật trong phạm vi giám sát. Khác với các cảm biến chủ động phát ra tín hiệu để đo lường phản xạ, cảm biến PIR là thiết bị thụ động, chỉ phát hiện bức xạ hồng ngoại từ các vật thể xung quanh.



Hình 2. cảm biến PIR

Cấu tạo của cảm biến PIR:

- **Phần tử cảm biến:** Trung tâm của cảm biến PIR là một vật liệu nhiệt điện (thường là tinh thể Lithium Tantalate - LiTaO_3 hoặc Gallium Nitride - GaN) có khả năng tạo ra điện áp khi tiếp xúc với bức xạ hồng ngoại.
- **Thấu kính Fresnel:** Phía trước cảm biến là một thấu kính Fresnel đặc biệt, chia vùng phát hiện thành nhiều khu vực (zones). Thấu kính này tập trung bức xạ hồng ngoại vào phần tử cảm biến và tạo ra mẫu phát hiện đặc trưng.
- **Mạch điện tử:** Bao gồm bộ khuếch đại tín hiệu, bộ lọc và mạch so sánh để xử lý tín hiệu từ phần tử cảm biến và chuyển đổi thành tín hiệu số (HIGH/LOW) đầu ra.
- **Các điều chỉnh:** Hầu hết các mô-đun cảm biến PIR đều có các điều chỉnh cho độ nhạy và thời gian trễ, cho phép tinh chỉnh theo nhu cầu cụ thể.

Nguyên lý hoạt động:

- **Phát hiện bức xạ hồng ngoại:** Tất cả các vật thể có nhiệt độ trên không độ tuyệt đối đều phát ra bức xạ hồng ngoại. Con người, với nhiệt độ cơ thể khoảng 37°C , phát ra bức xạ hồng ngoại ở bước sóng khoảng $9.4\mu\text{m}$, nằm trong dải phát hiện của cảm biến PIR.
- **Cơ chế phát hiện chuyển động:** Cảm biến PIR không phản ứng với mức bức xạ hồng ngoại tĩnh, mà chỉ phát hiện sự thay đổi. Khi một người di chuyển qua các vùng phát hiện của thấu kính Fresnel, cảm biến sẽ nhận thấy sự thay đổi trong mẫu bức xạ hồng ngoại.

- **Xử lý tín hiệu:** Khi phát hiện chuyển động, phần tử cảm biến tạo ra một tín hiệu điện nhỏ. Tín hiệu này được khuếch đại và xử lý bởi mạch điện tử để tạo ra tín hiệu đầu ra số.
- **Tín hiệu đầu ra:** Đầu ra của cảm biến PIR thường là một chân digital đơn giản, chuyển từ mức LOW sang HIGH khi phát hiện chuyển động và duy trì trong một khoảng thời gian nhất định (có thể điều chỉnh).

Đặc điểm kỹ thuật của cảm biến PIR:

- **Điện áp hoạt động:** 5-20V DC **Dòng điện tiêu thụ:** $<60\mu A$
- **Tín hiệu đầu ra:** 3.3V (HIGH) khi phát hiện, 0V (LOW) khi không phát hiện
Thời gian trễ: Có thể điều chỉnh từ 0.3 giây đến 5 phút
- **Khoảng cách phát hiện:** Lên đến 7m **Góc phát hiện:** Khoảng 110°
- **Thời gian khởi động:** Khoảng 60 giây

Trong hệ thống cảnh báo trộm, cảm biến PIR đóng vai trò quan trọng trong việc phát hiện sự xâm nhập của con người vào khu vực được bảo vệ. Khi phát hiện chuyển động, cảm biến sẽ gửi tín hiệu đến ESP32, kích hoạt quy trình cảnh báo

2.3 Giao tiếp giữa ESP32 và cảm biến PIR

Giao tiếp giữa ESP32 và cảm biến PIR là một quá trình đơn giản nhưng hiệu quả, dựa trên nguyên tắc đọc tín hiệu số từ đầu ra của cảm biến. Cơ chế này không yêu cầu các giao thức truyền thông phức tạp như I2C hay SPI, mà chỉ cần một kết nối trực tiếp giữa chân đầu ra của cảm biến và một chân GPIO của ESP32.

Kết nối phần cứng:

- **Nguồn điện:** Cảm biến PIR thường yêu cầu nguồn điện 5V, có thể lấy trực tiếp từ chân 5V của ESP32. Một số mô-đun PIR cũng có thể hoạt động ở mức 3.3V, phù hợp với mức logic của ESP32.
- **Đất (GND):** Chân GND của cảm biến PIR được kết nối với chân GND của ESP32 để tạo thành mạch điện hoàn chỉnh.
- **Tín hiệu đầu ra:** Chân OUT của cảm biến PIR được kết nối với một chân GPIO của ESP32 (trong dự án này là GPIO15). Chân này sẽ nhận tín hiệu HIGH khi cảm biến phát hiện chuyển động.

Xử lý tín hiệu trong phần mềm:

- **Cấu hình chân GPIO:** Chân GPIO kết nối với cảm biến PIR được cấu hình là INPUT_PULLUP trong hàm setup() của chương trình. Điều này đảm bảo chân luôn ở trạng thái xác định (HIGH) khi không có tín hiệu từ cảm biến.
- **Đọc trạng thái cảm biến:** ESP32 liên tục đọc trạng thái của chân GPIO kết nối với cảm biến PIR. Khi cảm biến phát hiện chuyển động, chân OUT của nó chuyển sang HIGH, và ESP32 nhận biết sự kiện này.
- **Xử lý ngắt (Interrupt):** Để tăng hiệu quả và giảm tải cho CPU, ESP32 thường sử dụng cơ chế ngắt để phát hiện sự thay đổi trạng thái của cảm biến PIR. Khi có chuyển động, ngắt được kích hoạt và hàm xử lý ngắt (ISR - Interrupt Service Routine) được gọi ngay lập tức.

```
// Thiết lập ngắt cho cảm biến PIR
attachInterrupt(digitalPinToInterrupt(motionSensorPin), detectsMovement,
RISING);

// Hàm xử lý ngắt
void IRAM_ATTR detectsMovement() {
    motionDetected = true;
}
```

Xử lý chống dội (Debouncing): Cảm biến PIR có thể tạo ra nhiều xung tín hiệu khi phát hiện chuyển động liên tục. Để tránh việc xử lý quá nhiều sự kiện, chương trình thường áp dụng kỹ thuật chống dội, chỉ xử lý một sự kiện trong một khoảng thời gian nhất định.

Ưu điểm của phương pháp giao tiếp này:

- **Đơn giản:** Không yêu cầu giao thức truyền thông phức tạp hay thư viện đặc biệt. **Hiệu quả:** Sử dụng ít tài nguyên hệ thống và tiêu thụ ít năng lượng.
- **Độ tin cậy cao:** Kết nối trực tiếp giảm thiểu khả năng lỗi truyền thông.
- **Thời gian phản hồi nhanh:** Cơ chế ngắt cho phép phát hiện và phản ứng với chuyển động gần như tức thì.

Trong hệ thống cảnh báo trộm, việc giao tiếp hiệu quả giữa ESP32 và cảm biến PIR là yếu tố quan trọng đảm bảo khả năng phát hiện xâm nhập kịp thời và chính xác.

2.4 Giao tiếp giữa ESP32 và Telegram thông qua Internet

Telegram là một nền tảng nhắn tin tức thời phổ biến với nhiều tính năng hữu ích cho các ứng dụng IoT, bao gồm API Bot mạnh mẽ cho phép tự động hóa và tương tác. Việc kết nối ESP32 với Telegram mở ra khả năng gửi thông báo từ xa và điều khiển thiết bị thông qua tin nhắn.

Telegram Bot API:

- **Bot Telegram:** Bot là ứng dụng của bên thứ ba chạy bên trong Telegram, có thể tương tác với người dùng hoặc nhóm. Bot có thể gửi tin nhắn, hình ảnh, và phản hồi các lệnh.
- **BotFather:** Để tạo Bot Telegram, người dùng tương tác với BotFather - một bot chính thức của Telegram. BotFather cung cấp token API duy nhất cho mỗi bot, được sử dụng để xác thực các yêu cầu API.
- **HTTP API:** Telegram Bot API hoạt động thông qua các yêu cầu HTTPS đến máy chủ Telegram. Các phương thức API phổ biến bao gồm sendMessage, getUpdates, và setWebhook.

Thư viện UniversalTelegramBot:

- Trong dự án này, ESP32 sử dụng thư viện UniversalTelegramBot để giao tiếp với Telegram Bot API. Thư viện này cung cấp các hàm đơn giản hóa việc gửi và nhận tin nhắn, xử lý lệnh, và quản lý trạng thái bot.

```
#include <UniversalTelegramBot.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

#define BOTtoken "7450782099:AAEOANjGR3Q6ok2kqRuYDm3pFFKq4hMXzto"
#define GROUP_ID "-4755358929"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
```

Quy trình giao tiếp:

- **Kết nối WiFi:** ESP32 kết nối với mạng WiFi để có thể truy cập Internet.

```
WiFi.begin(ssid, password);  
WiFi.mode(WIFI_STA);
```

- **Thiết lập bảo mật:** Giao tiếp với Telegram API yêu cầu kết nối HTTPS an toàn. ESP32 cần được cấu hình với chứng chỉ gốc phù hợp.

```
client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
```

- **Gửi tin nhắn:** Khi phát hiện chuyển động, ESP32 gửi tin nhắn cảnh báo đến người dùng hoặc nhóm Telegram đã được cấu hình.

```
bot.sendMessage(GROUP_ID, "⚠️ Có chuyển động! Vui lòng kiểm tra ngay.", "");
```

- **Nhận và xử lý tin nhắn:** ESP32 định kỳ kiểm tra các tin nhắn mới từ người dùng và xử lý chúng theo logic đã lập trình.

```
int numNewMessages = bot.getUpdates(bot.last_message_received + 1);  
while (numNewMessages) {  
    // Xử lý tin nhắn  
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);  
}
```

Ưu điểm của việc sử dụng Telegram cho thông báo IoT:

- **Bảo mật:** Telegram cung cấp mã hóa đầu cuối và các tính năng bảo mật khác, bảo vệ thông tin nhạy cảm.
- **Độ tin cậy:** Tin nhắn Telegram được gửi qua máy chủ đám mây, đảm bảo người dùng nhận được thông báo ngay cả khi không trực tuyến tại thời điểm gửi.
- **Đa nền tảng:** Người dùng có thể nhận thông báo trên nhiều thiết bị (điện thoại, máy tính bảng, máy tính) và hệ điều hành (Android, iOS, Windows, macOS, Linux).
- **Khả năng mở rộng:** Ngoài tin nhắn văn bản đơn giản, Telegram Bot có thể gửi hình ảnh, tệp, vị trí, và thậm chí tạo các nút tương tác.
- **Miễn phí và không giới hạn:** Telegram API miễn phí sử dụng và không có giới hạn nghiêm ngặt về số lượng tin nhắn.

Trong hệ thống cảnh báo trộm, Telegram đóng vai trò là kênh thông báo chính, cho phép người dùng nhận cảnh báo tức thời khi có xâm nhập, bất kể họ đang ở đâu, miễn là có kết nối internet.

2.5 Giới thiệu nền tảng mô phỏng Wokwi và phần mềm PlatformIO

Nền tảng mô phỏng Wokwi:

Wokwi là một nền tảng mô phỏng trực tuyến mạnh mẽ cho các dự án điện tử và IoT, cho phép người dùng thiết kế, lập trình và kiểm thử các hệ thống nhúng mà không cần phần cứng thực tế. Nền tảng này đặc biệt hữu ích trong giai đoạn phát triển ban đầu, giúp tiết kiệm thời gian và chi phí.

Đặc điểm chính của Wokwi:

- **Mô phỏng chính xác:** Wokwi mô phỏng chính xác hoạt động của nhiều vi điều khiển phổ biến như Arduino, ESP32, ESP8266, và Raspberry Pi Pico, cùng với các cảm biến và thiết bị ngoại vi.
- **Giao diện trực quan:** Nền tảng cung cấp giao diện kéo-thả trực quan để tạo sơ đồ mạch, với thư viện linh kiện phong phú bao gồm LED, nút nhấn, màn hình LCD, cảm biến, và nhiều hơn nữa.
- **Trình biên dịch tích hợp:** Wokwi tích hợp trình biên dịch cho nhiều ngôn ngữ lập trình nhúng, bao gồm C/C++ (Arduino), MicroPython, và CircuitPython.
- **Mô phỏng thời gian thực:** Người dùng có thể quan sát hoạt động của hệ thống trong thời gian thực, bao gồm trạng thái của các chân I/O, giao tiếp serial, và thậm chí là tín hiệu analog.
- **Chia sẻ và cộng tác:** Dự án Wokwi có thể dễ dàng chia sẻ thông qua URL, cho phép cộng tác và hỗ trợ từ xa.
- **Tích hợp với các nền tảng phát triển:** Wokwi có thể tích hợp với các IDE phổ biến như Arduino IDE, PlatformIO, và Visual Studio Code thông qua plugin.
- **Phần mềm PlatformIO:**
- PlatformIO là một hệ sinh thái phát triển chuyên nghiệp cho các hệ thống nhúng và IoT, cung cấp môi trường phát triển tích hợp đa nền tảng với quản lý thư viện, công cụ debug, và nhiều tính năng nâng cao khác.

Đặc điểm chính của PlatformIO:

- **Đa nền tảng:** Hỗ trợ hơn 40 nền tảng phát triển khác nhau, bao gồm Arduino, ESP32, STM32, và nhiều hơn nữa, với hơn 20 framework như Arduino, ESP-IDF, và CMSIS.
- **Quản lý thư viện thông minh:** Hệ thống quản lý thư viện tự động tìm kiếm, cài đặt và cập nhật các thư viện cần thiết cho dự án.
- **Tích hợp với nhiều IDE:** PlatformIO có thể tích hợp với nhiều IDE phổ biến như Visual Studio Code, Atom, CLion, và Eclipse, mang lại trải nghiệm phát triển linh hoạt.
- **Hệ thống xây dựng thống nhất:** Cung cấp hệ thống xây dựng thống nhất cho tất cả các nền tảng, đơn giản hóa quy trình phát triển đa nền tảng.
- **Công cụ debug nâng cao:** Tích hợp các công cụ debug mạnh mẽ, bao gồm debug phần cứng thông qua các giao thức như JTAG và SWD.
- **Kiểm thử tự động:** Hỗ trợ kiểm thử đơn vị và tích hợp liên tục (CI) cho các dự án nhúng.
- **Quản lý phiên bản:** Tích hợp với các hệ thống quản lý phiên bản như Git, giúp theo dõi và quản lý các thay đổi trong mã nguồn.

Lợi ích của việc sử dụng Wokwi và PlatformIO trong phát triển hệ thống cảnh báo trộm:

- **Phát triển nhanh chóng:** Wokwi cho phép kiểm thử ý tưởng và thuật toán mà không cần thiết lập phần cứng thực tế, trong khi PlatformIO đơn giản hóa quá trình phát triển và triển khai mã nguồn.
- **Giảm thiểu rủi ro:** Mô phỏng trên Wokwi giúp phát hiện và sửa lỗi trước khi triển khai trên phần cứng thực tế, giảm nguy cơ hư hỏng thiết bị.
- **Tài liệu hóa:** Cả hai nền tảng đều hỗ trợ tạo tài liệu và chia sẻ dự án, giúp quá trình học tập và cộng tác hiệu quả hơn.
- **Khả năng mở rộng:** PlatformIO hỗ trợ nhiều nền tảng phần cứng, cho phép dễ dàng chuyển đổi hoặc mở rộng dự án sang các vi điều khiển khác trong tương lai.

- **Tối ưu hóa mã nguồn:** PlatformIO cung cấp các công cụ phân tích và tối ưu hóa mã nguồn, giúp cải thiện hiệu suất và giảm tiêu thụ năng lượng của hệ thống.

Trong dự án hệ thống cảnh báo trộm này, Wokwi được sử dụng để mô phỏng kết nối giữa ESP32, cảm biến PIR và LED, trong khi PlatformIO trên Visual Studio Code được sử dụng để phát triển, biên dịch và tải mã nguồn lên vi điều khiển.

III. NGUYÊN LÝ HOẠT ĐỘNG

3.1 Tổng quan hệ thống cảnh báo trộm

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động PIR là một giải pháp an ninh thông minh, kết hợp giữa phần cứng điện tử và công nghệ Internet of Things (IoT). Hệ thống được thiết kế để phát hiện chuyển động không mong muốn trong khu vực được giám sát và gửi thông báo tức thời đến người dùng thông qua ứng dụng nhắn tin Telegram.

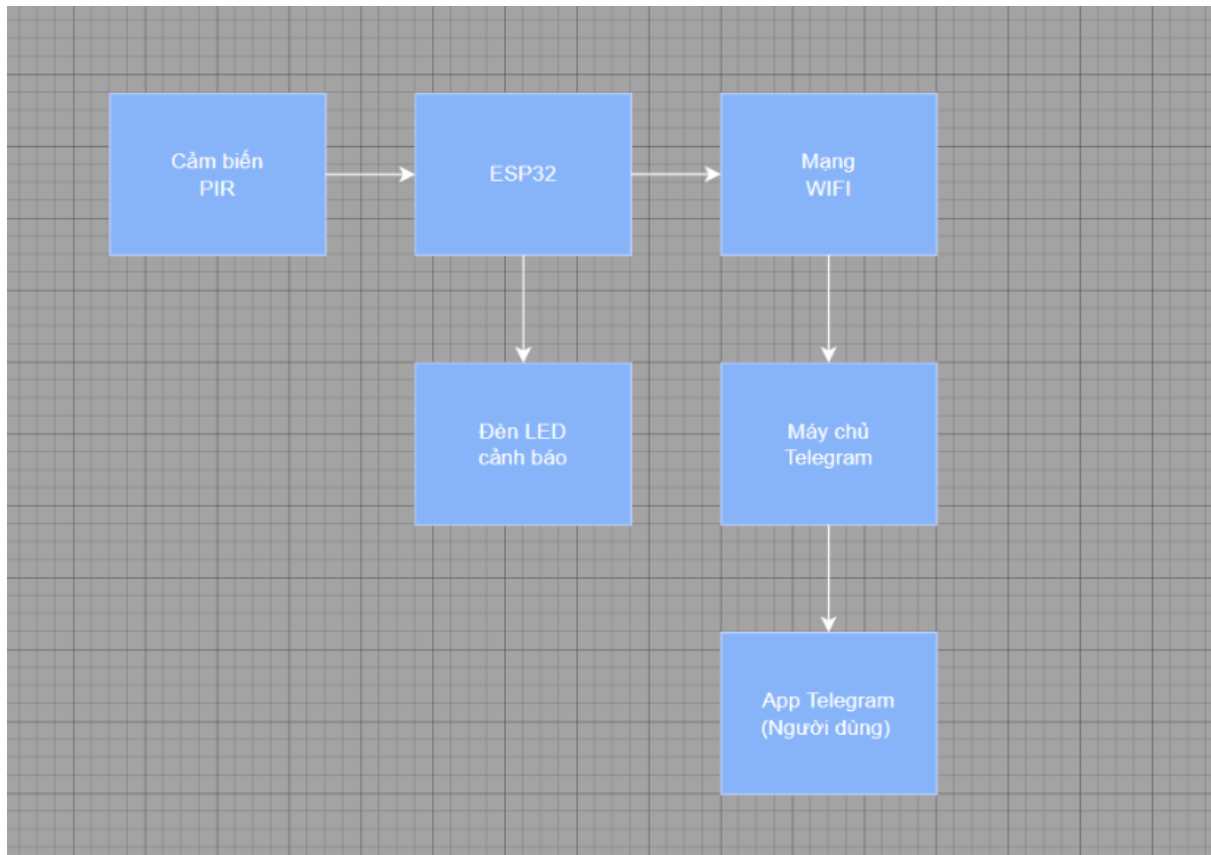
Nguyên lý hoạt động cơ bản của hệ thống bao gồm ba giai đoạn chính:

- **Phát hiện chuyển động:** Cảm biến PIR liên tục giám sát khu vực xung quanh. Khi phát hiện chuyển động của con người (thông qua sự thay đổi bức xạ hồng ngoại), cảm biến sẽ gửi tín hiệu điện đến ESP32.
- **Xử lý tín hiệu:** ESP32 nhận tín hiệu từ cảm biến PIR, xử lý thông tin và đưa ra quyết định. Trong trường hợp phát hiện chuyển động, ESP32 sẽ kích hoạt cảnh báo và chuẩn bị gửi thông báo.
- **Thông báo người dùng:** ESP32 sử dụng kết nối WiFi và Telegram Bot API để gửi tin nhắn cảnh báo đến người dùng hoặc nhóm đã được cấu hình trước. Đồng thời, hệ thống cũng kích hoạt đèn LED cảnh báo tại chỗ (mô phỏng cho còi báo động trong hệ thống thực tế).

Hệ thống được thiết kế với tính linh hoạt cao, cho phép tùy chỉnh và mở rộng theo nhu cầu cụ thể. Người dùng có thể dễ dàng điều chỉnh độ nhạy của cảm biến, thời gian trễ giữa các cảnh báo, hoặc thêm các tính năng bổ sung như camera giám sát, cảm biến nhiệt độ, hoặc các cơ chế xác thực để giảm cảnh báo giả.

3.2 Sơ đồ khối hệ thống

Hệ thống cảnh báo trộm có thể được mô tả qua sơ đồ khối sau, thể hiện các thành phần chính và luồng thông tin giữa chúng:



Hình 3. Sơ đồ khối hệ thống

Mô tả các thành phần trong sơ đồ khối:

- **Cảm biến PIR:** Đóng vai trò là "mắt" của hệ thống, liên tục quét khu vực xung quanh để phát hiện chuyển động. Khi phát hiện chuyển động, cảm biến gửi tín hiệu điện đến ESP32.
- **ESP32:** Là "bộ não" trung tâm của hệ thống, chịu trách nhiệm:
 - ❖ Nhận và xử lý tín hiệu từ cảm biến PIR
 - ❖ Điều khiển đèn LED cảnh báo
 - ❖ Kết nối với mạng WiFi
 - ❖ Giao tiếp với Telegram Bot API
 - ❖ Xử lý logic cảnh báo và các tùy chọn cấu hình

- **Đèn LED (Cảnh báo):** Đại diện cho thiết bị cảnh báo tại chỗ, được kích hoạt khi phát hiện chuyển động. Trong hệ thống thực tế, đây có thể là còi báo động, đèn flash, hoặc các thiết bị cảnh báo khác.
- **Mạng WiFi:** Cung cấp kết nối Internet cho ESP32, cho phép giao tiếp với máy chủ Telegram.
- **Máy chủ Telegram:** Xử lý các yêu cầu API từ ESP32 và chuyển tiếp tin nhắn cảnh báo đến người dùng.
- **Ứng dụng Telegram (Người dùng):** Giao diện cuối cùng nơi người dùng nhận được thông báo cảnh báo và có thể tương tác với hệ thống thông qua các lệnh.

3.3 Mô tả nguyên lý hoạt động

Hệ thống cảnh báo trộm hoạt động theo một quy trình có tổ chức, từ việc khởi động đến phát hiện chuyển động và gửi cảnh báo. Dưới đây là mô tả chi tiết về nguyên lý hoạt động của hệ thống:

a) Khởi động hệ thống:

Khi ESP32 được cấp nguồn, hệ thống thực hiện các bước khởi tạo sau:

- Thiết lập cấu hình cho các chân GPIO: chân kết nối với cảm biến PIR được cấu hình là INPUT_PULLUP, chân kết nối với đèn LED được cấu hình là OUTPUT.
- Thiết lập ngắt (interrupt) cho cảm biến PIR để phát hiện sự kiện chuyển động. Khởi tạo kết nối Serial để debug và giám sát hệ thống.
- Kết nối với mạng WiFi đã cấu hình.
- Thiết lập chứng chỉ bảo mật cho kết nối HTTPS với Telegram API.
- Gửi tin nhắn khởi động đến người dùng hoặc nhóm Telegram đã cấu hình, xác nhận hệ thống đã sẵn sàng hoạt động.


```

void setup() {
  Serial.begin(115200);

  pinMode(motionSensorPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(motionSensorPin), detectsMovement,
  RISING);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);

  WiFi.begin(ssid, password);
  WiFi.mode(WIFI_STA);
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);

  Serial.print("Đang kết nối WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.println("\n Đã kết nối WiFi");

  bot.sendMessage(GROUP_ID, " Hệ thống cảnh báo trộm đã khởi động", "");
}

```

b) Giám sát liên tục:

Sau khi khởi động, hệ thống chuyển sang trạng thái giám sát liên tục:

- Cảm biến PIR liên tục quét khu vực xung quanh để phát hiện chuyển động. ESP32 kiểm tra biến trạng thái motionDetected để xác định xem có chuyển động nào được phát hiện không.
- Đồng thời, ESP32 định kỳ kiểm tra các tin nhắn mới từ Telegram để xử lý các lệnh từ người dùng (nếu có)

c) Phát hiện chuyển động

Khi cảm biến PIR phát hiện chuyển động:

- Hàm ngắt detectsMovement() được gọi, đặt biến motionDetected thành true .
- Trong vòng lặp chính, ESP32 phát hiện trạng thái này và bắt đầu quy trình cảnh báo

```
void IRAM_ATTR detectsMovement() {
    motionDetected = true;
}
```

d) Xử lý cảnh báo

Khi phát hiện chuyển động, hệ thống thực hiện các hành động sau:

- Bật đèn LED cảnh báo (mô phỏng cho còi báo động trong hệ thống thực tế).
- Tạo tin nhắn cảnh báo với định dạng phù hợp.
- Gửi tin nhắn cảnh báo đến người dùng hoặc nhóm Telegram đã cấu hình.
- Ghi log sự kiện vào Serial Monitor để theo dõi và debug.
- Đặt lại biến motionDetected thành false để chuẩn bị cho lần phát hiện tiếp theo.
- Duy trì đèn LED cảnh báo trong một khoảng thời gian nhất định (5 giây) trước khi tắt.

```
if (motionDetected) {
    digitalWrite(ledPin, HIGH); // Bật đèn cảnh báo
    String msg = StringFormat("⚠ Có chuyển động! Vui lòng kiểm tra ngay.");
    bot.sendMessage(GROUP_ID, msg.c_str(), "");
    Serial.println("Đã gửi cảnh báo Telegram");
    motionDetected = false;

    delay(5000); // Giữ đèn sáng một lúc
}
```

```
digitalWrite(ledPin, LOW);
}
```

e) Kiểm tra tin nhắn từ người dùng

Ngoài việc phát hiện chuyển động, hệ thống còn định kỳ kiểm tra các tin nhắn mới từ người dùng thông qua Telegram:

- Hệ thống sử dụng phương thức getUpdates() của thư viện UniversalTelegramBot để lấy các tin nhắn mới.

- Các tin nhắn này có thể chứa các lệnh để điều khiển hoặc cấu hình hệ thống (trong phiên bản mở rộng).
- Việc kiểm tra được thực hiện định kỳ theo một khoảng thời gian nhất định (1 giây) để tránh tạo quá nhiều yêu cầu đến máy chủ Telegram.

```
if (millis() - lastCheckTime > checkInterval) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
        Serial.println("Nhận tin nhắn từ Telegram");
        numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastCheckTime = millis();
}
```

3.4 Luồng hoạt động và phản hồi khi phát hiện chuyển động

Luồng hoạt động của hệ thống khi phát hiện chuyển động có thể được mô tả chi tiết qua các bước sau:

a) Kích hoạt cảm biến

- Khi có chuyển động trong vùng phát hiện, cảm biến PIR phát hiện sự thay đổi trong mẫu bức xạ hồng ngoại.
- Đầu ra của cảm biến chuyển từ mức LOW sang HIGH.
- Sự thay đổi này kích hoạt ngắt (interrupt) đã được cấu hình trên ESP32.

b) Xử lý ngắt

- Hàm xử lý ngắt detectsMovement() được gọi ngay lập tức. Hàm này đặt biến cờ motionDetected thành true .
- Việc sử dụng ngắt đảm bảo không bỏ lỡ bất kỳ sự kiện chuyển động nào, ngay cả khi ESP32 đang thực hiện các tác vụ khác.

c) Xử lý trong vòng lặp chính

- Trong vòng lặp loop() , ESP32 liên tục kiểm tra giá trị của biến motionDetected.
- Khi phát hiện biến này có giá trị true, ESP32 bắt đầu quy trình cảnh báo.

d) Kích hoạt cảnh báo tại chỗ

- ESP32 đặt chân kết nối với đèn LED cảnh báo thành HIGH, bật đèn.

- Trong hệ thống thực tế, đây có thể là còi báo động, đèn flash, hoặc các thiết bị cảnh báo khác.
- Cảnh báo tại chỗ có thể làm kẻ đột nhập hoảng sợ và bỏ chạy, đồng thời cảnh báo cho những người xung quanh.

e) Chuẩn bị và gửi thông báo

- ESP32 tạo tin nhắn cảnh báo với nội dung "⚠ Có chuyển động! Vui lòng kiểm tra ngay."
- Tin nhắn được định dạng với biểu tượng cảnh báo để thu hút sự chú ý của người dùng.
- ESP32 sử dụng thư viện UniversalTelegramBot để gửi tin nhắn đến ID nhóm hoặc người dùng đã cấu hình.
- Quá trình gửi tin nhắn yêu cầu kết nối Internet thông qua WiFi.

f) Ghi log và theo dõi

- ESP32 ghi log sự kiện vào Serial Monitor với thông báo "Đã gửi cảnh báo Telegram".
- Thông tin này hữu ích cho việc debug và theo dõi hoạt động của hệ thống.

g) Đặt lại trạng thái

- Sau khi gửi cảnh báo, biến motionDetected được đặt lại thành false để chuẩn bị cho lần phát hiện tiếp theo.
- Đèn LED cảnh báo được giữ sáng trong 5 giây để đảm bảo cảnh báo đủ dài.
- Sau 5 giây, đèn LED được tắt (chân GPIO đặt thành LOW).

h) Nhận phản hồi từ người dùng

- Người dùng nhận được thông báo trên ứng dụng Telegram của họ.
- Thông báo này có thể đi kèm với âm thanh hoặc rung, tùy thuộc vào cài đặt của người dùng.
- Người dùng có thể ngay lập tức biết được có sự xâm nhập và thực hiện các biện pháp phù hợp.

i) Tương tác hai chiều (trong phiên bản mở rộng)

- Người dùng có thể phản hồi thông qua Telegram, gửi các lệnh như "status" để kiểm tra trạng thái hệ thống, "arm" hoặc "disarm" để bật/tắt hệ thống, hoặc "photo" để yêu cầu hình ảnh từ camera (nếu được tích hợp).

- ESP32 định kỳ kiểm tra các tin nhắn mới và xử lý các lệnh này.
- Điều này tạo ra một hệ thống tương tác hai chiều, cho phép người dùng không chỉ nhận thông báo mà còn điều khiển hệ thống từ xa.

Luồng hoạt động này đảm bảo hệ thống phản ứng nhanh chóng và đáng tin cậy khi có sự xâm nhập, đồng thời cung cấp cho người dùng thông tin kịp thời để có thể thực hiện các biện pháp phù hợp.

IV. THIẾT KẾ HỆ THỐNG

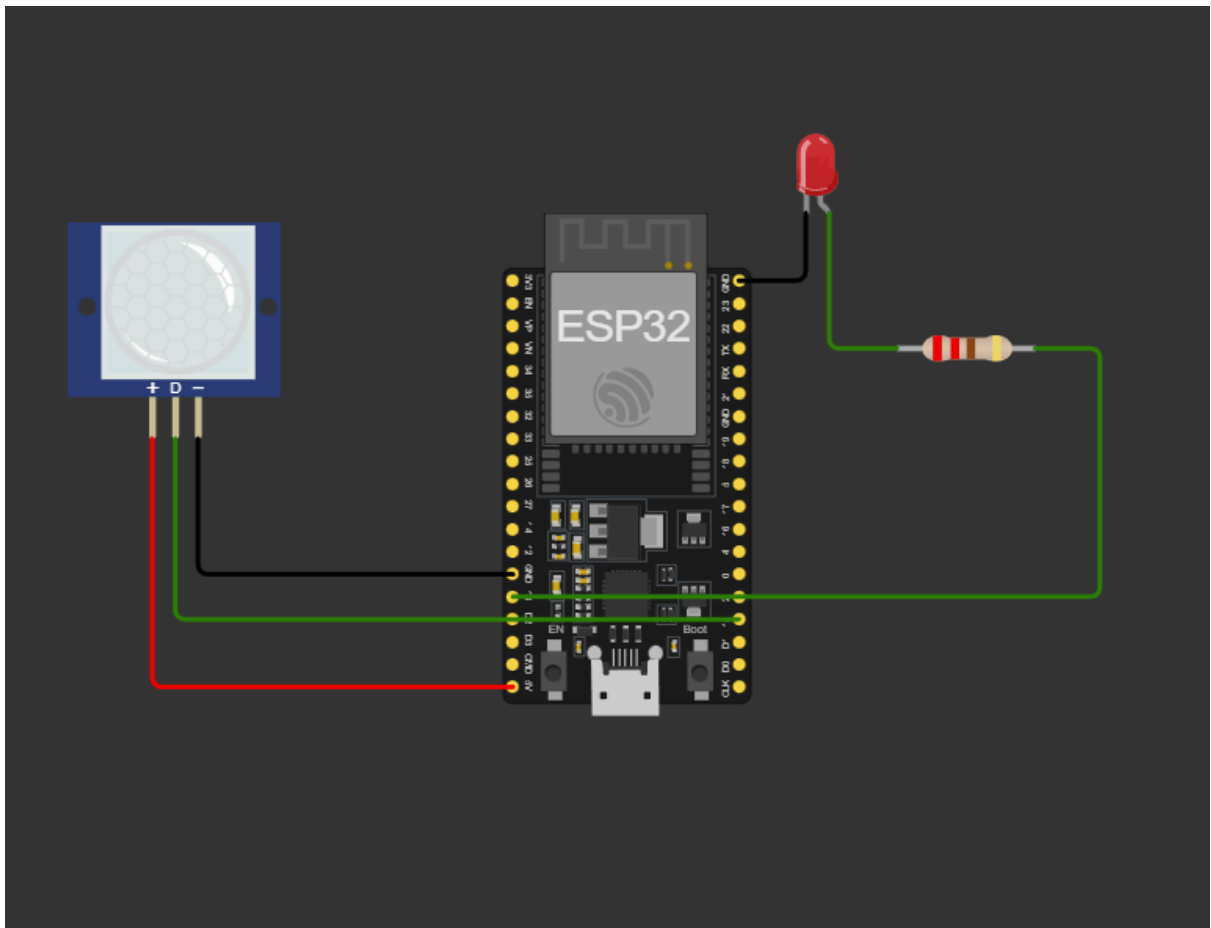
4.1 Danh sách linh kiện mô phỏng

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động được mô phỏng trên nền tảng Wokwi với các linh kiện sau:

STT	Linh kiện	Mô tả
1	ESP32 DevKit	Vi điều khiển trung tâm, phiên bản ESP32
2	Cảm biến chuyển động PIR	Cảm biến phát hiện chuyển động dựa trên bức xạ hồng ngoại
3	Đèn LED	Đèn LED màu đỏ, mô phỏng thiết bị cảnh báo
4	Điện trở	Điện trở 220Ω, giới hạn dòng điện cho đèn LED
5	Dây nối	Dây kết nối các thành phần trong mạch

Các linh kiện này được chọn lựa dựa trên tính phổ biến, khả năng tương thích và hiệu quả chi phí. Trong môi trường mô phỏng Wokwi, các linh kiện này được định nghĩa trong tệp cấu hình JSON như sau:

```
{
  "version": 1,
  "author": "Nguyen Phuoc Bao Quan",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 91.24, "attrs": {} },
    { "type": "wokwi-led", "id": "led1", "top": -42, "left": 205.4, "attrs": { "color": "red" } },
    { "type": "wokwi-pir-motion-sensor", "id": "pir1", "top": 4, "left": -93.78, "attrs": {} },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 51.95,
      "left": 259.2,
      "attrs": { "value": "220" }
    }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "pir1:VCC", "esp:5V", "red", [ "v0" ] ],
    [ "pir1:GND", "esp:GND.1", "black", [ "v0" ] ],
    [ "pir1:OUT", "esp:15", "green", [ "v0" ] ],
    [ "led1:C", "esp:GND.2", "black", [ "v0" ] ],
    [ "led1:A", "r1:1", "green", [ "v0" ] ],
    [ "r1:2", "esp:13", "green", [ "h27.6", "v105.6" ] ]
  ],
  "dependencies": {}
}
```



Hình 4. Mạch mô phỏng

Chi tiết về các linh kiện:

- **ESP32 DevKit**

- ❖ Bộ vi điều khiển ESP32 với WiFi và Bluetooth tích hợp 36 chân GPIO đa năng
- ❖ Bộ xử lý dual-core Tensilica Xtensa LX6 tốc độ lên đến 240MHz 520KB SRAM và hỗ trợ bộ nhớ Flash ngoài
- ❖ Giao diện USB tích hợp cho lập trình và cấp nguồn

- **Cảm biến chuyển động PIR:**

- ❖ Mô-đun cảm biến PIR tiêu chuẩn (HC-SR501 hoặc tương đương) Ba chân kết nối: VCC (nguồn), GND (đất), và OUT (đầu ra) Điện áp hoạt động: 5V DC
- ❖ Đầu ra digital: HIGH khi phát hiện chuyển động, LOW khi không phát hiện
Góc phát hiện: khoảng 110°

- ❖ Khoảng cách phát hiện: lên đến 7m
- **Đèn LED:**
 - ❖ LED màu đỏ tiêu chuẩn
 - ❖ Sử dụng để mô phỏng thiết bị cảnh báo (trong hệ thống thực tế có thể là còi báo động)
 - ❖ Điện áp hoạt động: 2-3V
 - ❖ Dòng điện định mức: khoảng 20mA
- **Điện trở 220Ω:**
 - ❖ Sử dụng để giới hạn dòng điện qua LED Giá trị: 220Ω
 - ❖ Công suất: 0.25W (1/4W)

Trong môi trường thực tế, ngoài các linh kiện cơ bản trên, hệ thống có thể được bổ sung thêm các thành phần như: - Pin hoặc nguồn điện dự phòng - Vỏ bảo vệ chống nước và bụi - Công tắc bật/tắt - Đèn báo trạng thái - Còi báo động thực tế thay cho đèn LED.

4.2 Kết nối phần cứng trên Wokwi

Các thành phần phần cứng trong hệ thống cảnh báo trộm được kết nối với nhau theo sơ đồ mạch đã được thiết kế trên nền tảng Wokwi. Các kết nối này được định nghĩa trong phần "connections" của tệp cấu hình JSON:

```
"connections": [
  [ "esp:TX", "$serialMonitor:RX", "", [ ] ],
  [ "esp:RX", "$serialMonitor:TX", "", [ ] ],
  [ "pir1:VCC", "esp:5V", "red", [ "v0" ] ],
  [ "pir1:GND", "esp:GND.1", "black", [ "v0" ] ],
  [ "pir1:OUT", "esp:15", "green", [ "v0" ] ],
  [ "led1:C", "esp:GND.2", "black", [ "v0" ] ],
  [ "led1:A", "r1:1", "green", [ "v0" ] ],
  [ "r1:2", "esp:13", "green", [ "h27.6", "v105.6" ] ]
],
```

Chi tiết về các kết nối:

Kết nối Serial Monitor:

- esp:TX → \$serialMonitor:RX : Kết nối chân TX của ESP32 với RX của Serial Monitor
- esp:RX → \$serialMonitor:TX : Kết nối chân RX của ESP32 với TX của Serial Monitor
- Các kết nối này cho phép giao tiếp giữa ESP32 và Serial Monitor để hiển thị thông tin debug và trạng thái hệ thống

Kết nối cảm biến PIR với ESP32:

- pir1:VCC → esp:5V : Cấp nguồn 5V cho cảm biến PIR từ chân 5V của ESP32 (dây màu đỏ)
- pir1:GND → esp:GND.1 : Kết nối chân GND của cảm biến PIR với chân GND của ESP32 (dây màu đen)
- pir1:OUT → esp:15 : Kết nối chân OUT (đầu ra) của cảm biến PIR với chân GPIO15 của ESP32 (dây màu xanh lá)

Kết nối đèn LED và điện trở với ESP32:

- led1:C → esp:GND.2 : Kết nối cực âm (Cathode) của LED với chân GND thứ hai của ESP32 (dây màu đen)
- led1:A → r1:1 : Kết nối cực dương (Anode) của LED với một đầu của điện trở (dây màu xanh lá)
- r1:2 → esp:13 : Kết nối đầu còn lại của điện trở với chân GPIO13 của ESP32 (dây màu xanh lá)

Giải thích về các chân GPIO được sử dụng:

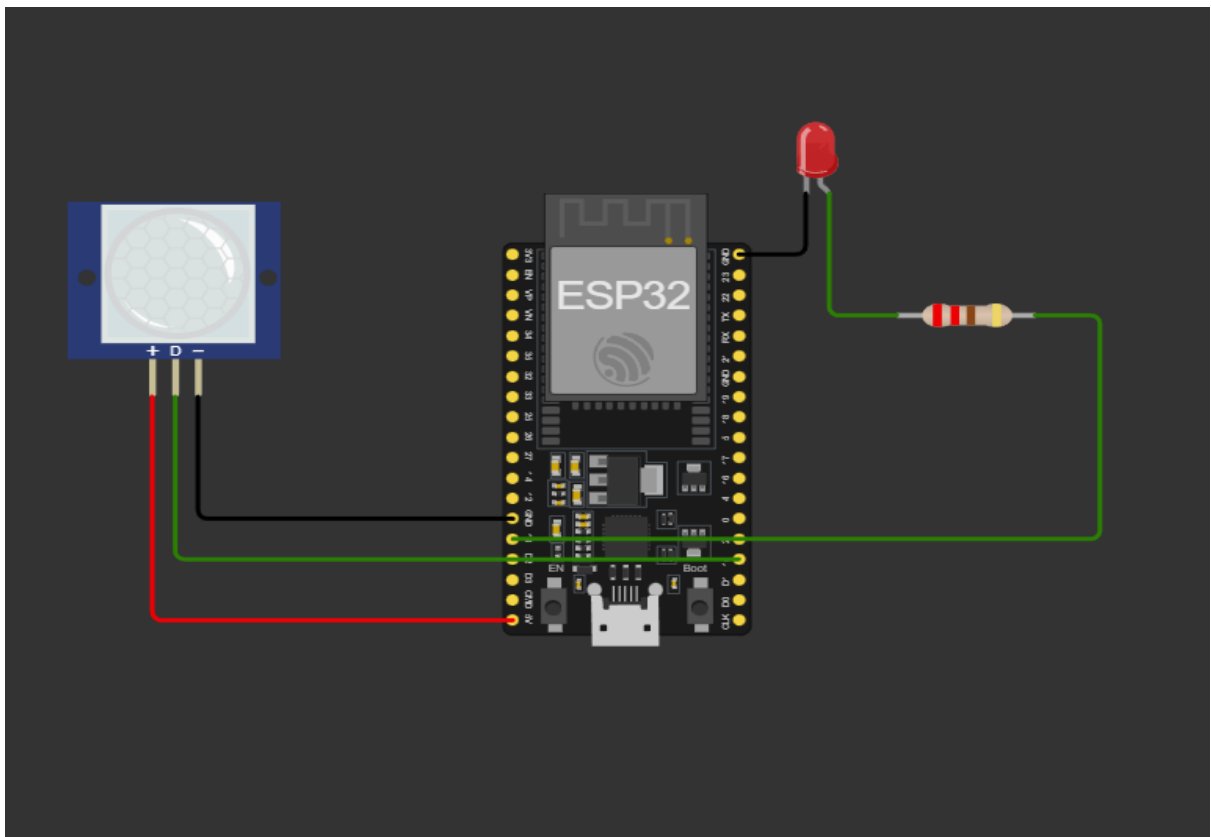
GPIO15 (Cảm biến PIR):

- Được cấu hình là INPUT_PULLUP trong mã nguồn Sử dụng cho việc đọc tín hiệu từ cảm biến PIR
- Hỗ trợ ngắt (interrupt) để phát hiện sự thay đổi trạng thái

GPIO13 (Đèn LED):

- Được cấu hình là OUTPUT trong mã nguồn Điều khiển đèn LED cảnh báo
- Khi GPIO13 được đặt thành HIGH, đèn LED sáng; khi đặt thành LOW, đèn LED tắt

Các kết nối này tạo thành một mạch điện hoàn chỉnh, cho phép ESP32 nhận tín hiệu từ cảm biến PIR và điều khiển đèn LED cảnh báo khi phát hiện chuyển động.



Hình 5. Mạch mô phỏng

V. LẬP TRÌNH VÀ TRIỂN KHAI

5.1 Mô tả thuật toán và logic xử lý

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR được xây dựng theo một thuật toán rõ ràng và hiệu quả, tập trung vào việc phát hiện chuyển động và gửi thông báo kịp thời. Dưới đây là mô tả chi tiết về thuật toán và logic xử lý của hệ thống:

Thuật toán tổng quát:

- Khởi tạo:
 - ❖ Thiết lập cấu hình cho các chân GPIO
 - ❖ Thiết lập ngắt cho cảm biến PIR
 - ❖ Khởi tạo kết nối Serial
 - ❖ Kết nối WiFi
 - ❖ Thiết lập chứng chỉ bảo mật cho Telegram
 - ❖ Gửi thông báo khởi động
- Vòng lặp chính:
 - ❖ Kiểm tra biến cờ phát hiện chuyển động:
 - Nếu có chuyển động (`motionDetected = true`):
 - + Bật đèn LED cảnh báo
 - + Tạo và gửi thông báo cảnh báo qua Telegram + Ghi **log** sự kiện
 - + Đặt lại biến cờ (`motionDetected = false`) + Chờ 5 giây
 - + Tắt đèn LED cảnh báo
 - ❖ Định kỳ kiểm tra tin nhắn từ Telegram:
 - Nếu đã đến thời điểm kiểm tra (dựa trên khoảng thời gian `checkInterval`):
 - + Lấy các tin nhắn mới từ Telegram + Xử lý các tin nhắn (nếu có)
 - + Cập nhật thời điểm kiểm tra cuối cùng
 - Xử lý ngắt (khi phát hiện chuyển động):
 - ❖ Đặt biến cờ `motionDetected = true`

Logic xử lý chi tiết:

Xử lý ngắt (Interrupt Handling):

- Hệ thống sử dụng cơ chế ngắt để phát hiện chuyển động từ cảm biến PIR
- Khi cảm biến PIR phát hiện chuyển động, chân OUT của nó chuyển từ LOW sang HIGH
- Sự thay đổi này kích hoạt ngắt trên ESP32, gọi hàm `detectsMovement()`
- Hàm ngắt đơn giản chỉ đặt biến cờ `motionDetected` thành true

- Việc sử dụng ngắt đảm bảo không bỏ lỡ bất kỳ sự kiện chuyển động nào, ngay cả khi ESP32 đang thực hiện các tác vụ khác


Xử lý sự kiện chuyển động:

- Trong vòng lặp chính, hệ thống liên tục kiểm tra giá trị của biến `motionDetected`
- Khi phát hiện biến này có giá trị `true`, hệ thống thực hiện quy trình cảnh báo:
- Bật đèn LED cảnh báo bằng cách đặt chân GPIO13 thành HIGH
- Tạo tin nhắn cảnh báo với định dạng phù hợp
- Gửi tin nhắn đến người dùng hoặc nhóm Telegram đã cấu hình
- Ghi log sự kiện vào Serial Monitor
- Đặt lại biến `motionDetected` thành `false` để chuẩn bị cho lần phát hiện tiếp theo
- Duy trì đèn LED cảnh báo trong 5 giây
- Tắt đèn LED bằng cách đặt chân GPIO13 thành LOW

Kiểm tra tin nhắn từ Telegram:

- Hệ thống định kỳ kiểm tra các tin nhắn mới từ người dùng thông qua Telegram
- Việc kiểm tra được thực hiện theo một khoảng thời gian nhất định (1 giây)
- Hệ thống sử dụng phương thức `getUpdates()` của thư viện `UniversalTelegramBot` Trong phiên bản hiện tại, hệ thống chỉ kiểm tra sự tồn tại của tin nhắn mới mà không xử lý nội dung
- Trong phiên bản mở rộng, có thể thêm logic để xử lý các lệnh từ người dùng

Định dạng tin nhắn:

- Hệ thống sử dụng hàm `StringFormat()` để tạo tin nhắn với định dạng phù hợp
- Hàm này cho phép tạo chuỗi định dạng tương tự như hàm `printf()` trong C.
- Tin nhắn cảnh báo được định dạng với biểu tượng cảnh báo () để thu hút sự chú ý

Ưu điểm của thuật toán:

Hiệu quả và đáng tin cậy:

- Sử dụng cơ chế ngắt để đảm bảo không bỏ lỡ sự kiện chuyển động
- Xử lý đơn giản và trực tiếp, giảm thiểu khả năng lỗi

Tiết kiệm tài nguyên:

- Không sử dụng polling liên tục để kiểm tra cảm biến, giảm tải cho CPU
- Chỉ kiểm tra tin nhắn Telegram định kỳ, không liên tục

Khả năng mở rộng:

- Cấu trúc thuật toán cho phép dễ dàng thêm các tính năng mới
- Có thể mở rộng để xử lý các lệnh từ người dùng thông qua Telegram

5.2 Lập trình kết nối WIFI và Telegram Bot

Việc kết nối ESP32 với mạng WiFi và thiết lập Telegram Bot là bước quan trọng để hệ thống có thể gửi thông báo đến người dùng. Dưới đây là phân tích chi tiết về phân mã nguồn liên quan đến kết nối WiFi và Telegram Bot:

Khai báo thư viện và biến toàn cục:

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

// WiFi
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// Telegram BOT
#define BOTtoken "7450782099:AAEOANjGR3Q6ok2kqRuYDm3pfFKq4hMXzto"
#define GROUP_ID "-4755358929" // Chat ID nhóm hoặc cá nhân

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
```

Trong đoạn mã trên: - Các thư viện cần thiết được include: WiFi.h để kết nối WiFi, WiFiClientSecure.h để tạo kết nối HTTPS an toàn, UniversalTelegramBot.h để tương tác với Telegram Bot API, và ArduinoJson.h để xử lý dữ liệu JSON. - Thông tin WiFi

được khai báo: ssid là tên mạng WiFi, password là mật khẩu (trong môi trường mô phỏng Wokwi, mật khẩu để trống). - Thông tin Telegram Bot được khai báo: BOTtoken là token xác thực của bot, GROUP_ID là ID của nhóm hoặc người dùng sẽ nhận thông báo. - Đối tượng client của lớp WiFiClientSecure được tạo để xử lý kết nối HTTPS. - Đối tượng bot của lớp UniversalTelegramBot được tạo, sử dụng token và client đã khai báo.

Thiết lập kết nối WiFi và Telegram Bot:

```
void setup() {  
  Serial.begin(115200);  
  
  // ... (Thiết lập GPIO và ngắt) ...  
  
  WiFi.begin(ssid, password);  
  WiFi.mode(WIFI_STA);  
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);  
  
  Serial.print("Đang kết nối WiFi");  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500); Serial.print(".");  
  }  
  Serial.println("\n Đã kết nối WiFi");  
  
  bot.sendMessage(GROUP_ID, " Hệ thống cảnh báo trộm đã khởi động", "");  
}
```

Trong đoạn mã trên: - WiFi.begin(ssid, password) bắt đầu quá trình kết nối WiFi với thông tin đã cung cấp. - WiFi.mode(WIFI_STA) đặt ESP32 ở chế độ Station (kết nối đến một mạng WiFi có sẵn). -

client.setCACert(TELEGRAM_CERTIFICATE_ROOT) thiết lập chứng chỉ gốc cho kết nối HTTPS an toàn đến máy chủ Telegram. TELEGRAM_CERTIFICATE_ROOT là một hằng số được định nghĩa trong thư viện UniversalTelegramBot. - Vòng lặp while (WiFi.status() != WL_CONNECTED) đợi cho đến khi kết nối WiFi được thiết lập thành công, với thông báo tiến trình qua Serial. - Sau khi kết nối WiFi thành công, bot.sendMessage(GROUP_ID, " Hệ thống cảnh báo trộm đã khởi động", "") gửi tin nhắn khởi động đến người dùng hoặc nhóm đã cấu hình.

Hàm định dạng chuỗi cho tin nhắn:

```
// Gửi tin nhắn có định dạng
String StringFormat(const char* fmt, ...) {
    va_list args;
    va_start(args, fmt);
    char buffer[128];
    vsnprintf(buffer, sizeof(buffer), fmt, args);
    va_end(args);
    return String(buffer);
}
```

Hàm StringFormat() là một tiện ích để tạo chuỗi định dạng tương tự như hàm printf() trong C: - Sử dụng va_list, va_start(), và va_end() để xử lý danh sách tham số biến đổi. -

Sử dụng vsnprintf() để định dạng chuỗi với kích thước buffer cố định (128 byte). - Trả về đối tượng String của Arduino từ buffer đã định dạng.

Kiểm tra và xử lý tin nhắn từ Telegram:

```
void loop() {
    // ... (Xử lý phát hiện chuyển động) ...

    // Kiểm tra tin nhắn từ bot
    if (millis() - lastCheckTime > checkInterval) {
        int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
        while (numNewMessages) {
            Serial.println("Nhận tin nhắn từ Telegram");
            numNewMessages = bot.getUpdates(bot.last_message_received + 1);
        }
        lastCheckTime = millis();
    }
}
```

Trong đoạn mã trên: - if (millis() - lastCheckTime > checkInterval) kiểm tra xem đã đến thời điểm kiểm tra tin nhắn mới chưa, dựa trên khoảng thời gian checkInterval (1000ms). - bot.getUpdates(bot.last_message_received + 1) lấy các tin nhắn mới từ Telegram, bắt đầu từ tin nhắn tiếp theo sau tin nhắn cuối cùng đã nhận. - Vòng lặp

while (numNewMessages) xử lý tất cả các tin nhắn mới, mặc dù trong phiên bản hiện tại, nó chỉ ghi log sự kiện mà không xử lý nội dung tin nhắn. - lastCheckTime = millis() cập nhật thời điểm kiểm tra cuối cùng.

Gửi thông báo cảnh báo qua Telegram:

```
if (motionDetected) {  
    digitalWrite(ledPin, HIGH); // Bật đèn cảnh báo  
    String msg = StringFormat("⚠ Có chuyển động! Vui lòng kiểm tra ngay.");  
    bot.sendMessage(GROUP_ID, msg.c_str(), "");  
    Serial.println("Đã gửi cảnh báo Telegram");  
    motionDetected = false;  
  
    delay(5000); // Giữ đèn sáng một lúc  
    digitalWrite(ledPin, LOW);  
}
```

Trong đoạn mã trên: - Khi phát hiện chuyển động (motionDetected = true), hệ thống tạo tin nhắn cảnh báo với định dạng phù hợp. - bot.sendMessage(GROUP_ID, msg.c_str(), "") gửi tin nhắn cảnh báo đến người dùng hoặc nhóm đã cấu hình. - Tham số

thứ ba của sendMessage() là một chuỗi rỗng, có thể được sử dụng để chỉ định định dạng tin nhắn (HTML, Markdown, v.v.) trong các phiên bản mở rộng.

Các cải tiến có thể thực hiện:

Xử lý lệnh từ người dùng:

- Thêm logic để phân tích và phản hồi các lệnh từ người dùng như "status", "arm", "disarm", v.v.
- Sử dụng ArduinoJson để phân tích cú pháp JSON từ phản hồi Telegram.

Xử lý lỗi kết nối:

- Thêm cơ chế kiểm tra và tái kết nối WiFi khi mất kết nối.
- Thêm timeout và retry cho các yêu cầu Telegram để tăng độ tin cậy.

Bảo mật nâng cao:

- Thêm cơ chế xác thực người dùng để chỉ cho phép người dùng được ủy quyền điều khiển hệ thống.
- Mã hóa thông tin nhạy cảm như token bot và thông tin WiFi

5.3 Lập trình phát hiện chuyển động từ cảm biến PIR

Phát hiện chuyển động từ cảm biến PIR là chức năng cốt lõi của hệ thống cảnh báo trộm. Dưới đây là phân tích chi tiết về phần mã nguồn liên quan đến việc phát hiện chuyển động

Khai báo chân GPIO và biến trạng thái:

```
// Cảm biến PIR và LED cảnh báo  
const int motionSensorPin = 15; // PIR Sensor OUT  
const int ledPin = 13; // LED đỏ mô phỏng còi báo  
bool motionDetected = false;  
bool ledState = LOW;  
unsigned long lastCheckTime = 0;  
const int checkInterval = 1000; // ms
```

Trong đoạn mã trên: - motionSensorPin là chân GPIO15 của ESP32, được kết nối với chân OUT của cảm biến PIR. - ledPin là chân GPIO13 của ESP32, được kết nối với đèn LED cảnh báo (thông qua điện trở). - motionDetected là biến cờ để theo dõi trạng thái phát hiện chuyển động. - ledState là biến để theo dõi trạng thái của đèn LED. - lastCheckTime và checkInterval được sử dụng để định kỳ kiểm tra tín hiệu từ Telegram.

Thiết lập chân GPIO và ngắt:

```

void setup() {
  Serial.begin(115200);

  pinMode(motionSensorPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(motionSensorPin), detectsMovement,
  RISING);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);

  // ... (Thiết lập WiFi và Telegram) ...
}

```

Trong đoạn mã trên: - pinMode(motionSensorPin, INPUT_PULLUP) cấu hình chân kết nối với cảm biến PIR là INPUT với điện trở kéo lên (pull-up) nội bộ. Điều này đảm bảo chân luôn ở trạng thái xác định (HIGH) khi không có tín hiệu từ cảm biến. - attachInterrupt(digitalPinToInterrupt(motionSensorPin), detectsMovement, RISING) thiết lập ngắt cho chân cảm biến PIR: + digitalPinToInterrupt(motionSensorPin) chuyển đổi số chân GPIO thành số ngắt tương ứng. + detectsMovement là hàm sẽ được gọi khi ngắt xảy ra. + RISING chỉ định ngắt sẽ được kích hoạt khi tín hiệu chuyển từ LOW sang HIGH (cạnh lên), tương ứng với thời điểm cảm biến phát hiện chuyển động. - pinMode(ledPin, OUTPUT) cấu hình chân kết nối với đèn LED là OUTPUT. - digitalWrite(ledPin, ledState) đặt trạng thái ban đầu cho đèn LED (LOW - tắt).

Hàm xử lý ngắt:

```

// Ngắt khi có chuyển động
void IRAM_ATTR detectsMovement() {
  motionDetected = true;
}

```

Hàm detectsMovement() là hàm xử lý ngắt (Interrupt Service Routine - ISR) được gọi khi cảm biến PIR phát hiện chuyển động: - Từ khóa IRAM_ATTR chỉ định hàm này sẽ được lưu trữ trong IRAM (Instruction RAM) thay vì Flash, giúp tăng tốc độ thực thi

và độ tin cậy của hàm ngắt. - Hàm này đơn giản chỉ đặt biến cờ motionDetected thành true. - Hàm xử lý ngắt nên càng ngắn gọn càng tốt để tránh ảnh hưởng đến các hoạt động khác của hệ thống.

Xử lý phát hiện chuyển động trong vòng lặp chính:

```
void loop() {  
  if (motionDetected) {  
    digitalWrite(ledPin, HIGH); // Bật đèn cảnh báo  
    String msg = StringFormat("⚠ Có chuyển động! Vui lòng kiểm tra ngay.");  
    bot.sendMessage(GROUP_ID, msg.c_str(), "");  
    Serial.println("Đã gửi cảnh báo Telegram");  
    motionDetected = false;  
  
    delay(5000); // Giữ đèn sáng một lúc  
    digitalWrite(ledPin, LOW);  
  }  
  
  // ... (Kiểm tra tin nhắn từ Telegram) ...  
}
```

Trong đoạn mã trên: - if (motionDetected) kiểm tra xem có chuyển động nào được phát hiện không. - Khi phát hiện chuyển động: + digitalWrite(ledPin, HIGH) bật đèn LED cảnh báo. + Tạo và gửi tin nhắn cảnh báo qua Telegram. + Ghi log sự kiện vào Serial Monitor. + motionDetected = false đặt lại biến cờ để chuẩn bị cho lần phát hiện tiếp theo. + delay(5000) giữ đèn LED sáng trong 5 giây. + digitalWrite(ledPin, LOW) tắt đèn LED.

Các cải tiến có thể thực hiện:

Chống dội (Debouncing):

- Thêm cơ chế chống dội để tránh nhiều cảnh báo liên tiếp khi cảm biến phát hiện chuyển động liên tục.
- Sử dụng biến thời gian để đảm bảo khoảng cách tối thiểu giữa các cảnh báo.

Đa cảm biến:

- Mở rộng hệ thống để hỗ trợ nhiều cảm biến PIR ở các vị trí khác nhau.
- Thêm thông tin về vị trí cụ thể của cảm biến phát hiện chuyển động trong tin nhắn cảnh báo.

Chế độ hoạt động:

- Thêm các chế độ hoạt động như "armed" (kích hoạt), "disarmed" (vô hiệu hóa), "home" (ở nhà), "away" (vắng nhà).
- Cho phép người dùng chuyển đổi giữa các chế độ thông qua lệnh Telegram.

Điều chỉnh độ nhạy:

- Trong hệ thống thực tế, thêm khả năng điều chỉnh độ nhạy của cảm biến PIR từ xa thông qua lệnh Telegram.
- Sử dụng PWM hoặc DAC để điều chỉnh ngưỡng phát hiện.

5.4 Gửi cảnh báo lên Telegram khi có xâm nhập

Việc gửi cảnh báo kịp thời đến người dùng khi phát hiện xâm nhập là một trong những chức năng quan trọng nhất của hệ thống. Dưới đây là phân tích chi tiết về phần mã nguồn liên quan đến việc gửi cảnh báo qua Telegram:

Quy trình gửi cảnh báo:

```

if (motionDetected) {
  digitalWrite(ledPin, HIGH); // Bật đèn cảnh báo
  String msg = StringFormat("⚠ Có chuyển động! Vui lòng kiểm tra ngay.");
  bot.sendMessage(GROUP_ID, msg.c_str(), "");
  Serial.println("Đã gửi cảnh báo Telegram");
  motionDetected = false;

  delay(5000); // Giữ đèn sáng một lúc
  digitalWrite(ledPin, LOW);
}

```

Quy trình gửi cảnh báo bao gồm các bước sau:

Kích hoạt cảnh báo tại chỗ:

- digitalWrite(ledPin, HIGH) bật đèn LED cảnh báo.
- Trong hệ thống thực tế, đây có thể là còi báo động, đèn flash, hoặc các thiết bị cảnh báo khác.

Tạo tin nhắn cảnh báo:

- String msg = StringFormat("⚠ Có chuyển động! Vui lòng kiểm tra ngay.") tạo tin nhắn cảnh báo với định dạng phù hợp.
- Tin nhắn bao gồm biểu tượng cảnh báo (⚠) để thu hút sự chú ý của người dùng.
- Nội dung tin nhắn ngắn gọn nhưng đầy đủ thông tin, cho người dùng biết có chuyển động đáng ngờ và cần kiểm tra.

Gửi tin nhắn qua Telegram:

- bot.sendMessage(GROUP_ID, msg.c_str(), "") gửi tin nhắn đến người dùng hoặc nhóm đã cấu hình.
- GROUP_ID là ID của nhóm hoặc người dùng sẽ nhận thông báo.
- msg.c_str() chuyển đổi đối tượng String của Arduino thành chuỗi C-style mà hàm sendMessage() yêu cầu.
- Tham số thứ ba là một chuỗi rỗng, có thể được sử dụng để chỉ định định dạng tin nhắn (HTML, Markdown, v.v.).

Ghi log sự kiện:

- `Serial.println("Đã gửi cảnh báo Telegram")` ghi log sự kiện vào Serial Monitor.
- Thông tin này hữu ích cho việc debug và theo dõi hoạt động của hệ thống.

Đặt lại trạng thái:

- `motionDetected = false` đặt lại biến cờ để chuẩn bị cho lần phát hiện tiếp theo. `delay(5000)` giữ đèn LED sáng trong 5 giây để đảm bảo cảnh báo đủ dài. `digitalWrite(ledPin, LOW)` tắt đèn LED sau khi hoàn tất cảnh báo.

Phân tích chi tiết về phương thức `sendMessage()`:

Phương thức `sendMessage()` của thư viện `UniversalTelegramBot` là phương thức chính để gửi tin nhắn đến người dùng Telegram:

```
bot.sendMessage(GROUP_ID, msg.c_str(), "");
```

Phương thức này thực hiện các bước sau:

Tạo yêu cầu HTTP POST đến API của Telegram với endpoint `/sendMessage`.

Đóng gói thông tin tin nhắn vào body của yêu cầu, bao gồm:

- `chat_id` : ID của người dùng hoặc nhóm sẽ nhận tin nhắn (`GROUP_ID`).
- `text` : Nội dung tin nhắn (`msg.c_str()`).
- `parse_mode` : Chế độ phân tích cú pháp tin nhắn (HTML, Markdown, v.v.), nếu được chỉ định.

Thiết lập header HTTP phù hợp, bao gồm token xác thực của bot.

Gửi yêu cầu đến máy chủ Telegram thông qua kết nối HTTPS an toàn.

Xử lý phản hồi từ máy chủ Telegram, kiểm tra xem tin nhắn đã được gửi thành công hay không.

Các cải tiến có thể thực hiện:

Thêm thông tin chi tiết:

- Bổ sung thông tin về thời gian phát hiện chuyển động.
- Nếu có nhiều cảm biến, thêm thông tin về vị trí cụ thể của cảm biến phát hiện chuyển động.
- Thêm thông tin về trạng thái hệ thống như mức pin (nếu chạy bằng pin), cường độ tín hiệu WiFi, v.v.

Gửi hình ảnh hoặc video:

- Tích hợp camera ESP32-CAM để chụp ảnh hoặc quay video khi phát hiện chuyển động.
- Sử dụng phương thức sendPhoto() hoặc sendVideo() của thư viện UniversalTelegramBot để gửi hình ảnh hoặc video kèm theo tin nhắn cảnh báo.

Xử lý lỗi và retry:

- Thêm cơ chế kiểm tra kết quả gửi tin nhắn và thử lại nếu gửi không thành công.
- Lưu trữ các tin nhắn chưa gửi được trong bộ nhớ và thử gửi lại khi kết nối được khôi phục.

Tùy chọn thông báo:

- Cho phép người dùng cấu hình loại thông báo họ muốn nhận (chỉ văn bản, văn bản + hình ảnh, v.v.).
- Cho phép người dùng đặt lịch cho các khoảng thời gian khi họ không muốn nhận thông báo (ví dụ: khi họ ở nhà).

Tích hợp với các dịch vụ khác:

- Ngoài Telegram, mở rộng hệ thống để gửi thông báo qua email, SMS, hoặc các nền tảng nhắn tin khác.
- Tích hợp với các hệ thống nhà thông minh như Home Assistant, Google Home, hoặc Amazon Alexa.

Việc gửi cảnh báo qua Telegram khi phát hiện xâm nhập là một tính năng mạnh mẽ của hệ thống, cho phép người dùng nhận thông báo tức thời bất kể họ đang ở đâu, miễn là có kết nối internet. Với các cải tiến được đề xuất, hệ thống có thể trở nên toàn diện hơn, cung cấp thông tin chi tiết hơn và tùy chọn linh hoạt hơn cho người dùng.

VI. KẾT QUẢ VÀ ĐÁNH GIÁ

6.1 Kết quả mô phỏng thực tế

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR đã được mô phỏng thành công trên nền tảng Wokwi. Kết quả mô phỏng cho thấy hệ thống hoạt động đúng như thiết kế, với khả năng phát hiện chuyển động và gửi thông báo qua Telegram một cách đáng tin cậy. Dưới đây là chi tiết về kết quả mô phỏng:

Kết quả khởi động hệ thống:

Khi hệ thống được khởi động, các quá trình sau diễn ra theo trình tự:


Khởi tạo Serial Monitor: Hệ thống thiết lập kết nối Serial với tốc độ baud 115200 để hiển thị thông tin debug và trạng thái.

Cấu hình GPIO và ngắt: Các chân GPIO được cấu hình đúng chức năng, với chân cảm biến PIR được thiết lập ngắt thành công.

Kết nối WiFi: Hệ thống kết nối thành công với mạng WiFi đã cấu hình. Quá trình này được hiển thị trên Serial Monitor với thông báo "Đang kết nối WiFi..." và kết thúc bằng "Đã kết nối WiFi" khi kết nối thành công.

Khởi tạo Telegram Bot: Sau khi kết nối WiFi, hệ thống thiết lập kết nối với Telegram Bot API và gửi tin nhắn khởi động "Hệ thống cảnh báo trộm đã khởi động" đến người dùng hoặc nhóm đã cấu hình.

Kết quả hiển thị trên terminal:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  QUERY RESULTS (PREVIEW)

ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Đang kết nối WiFi..
❖❖ Đã kết nối WiFi
```

Hình 6. Kết quả hiển thị trên terminal

Kết quả phát hiện chuyển động:

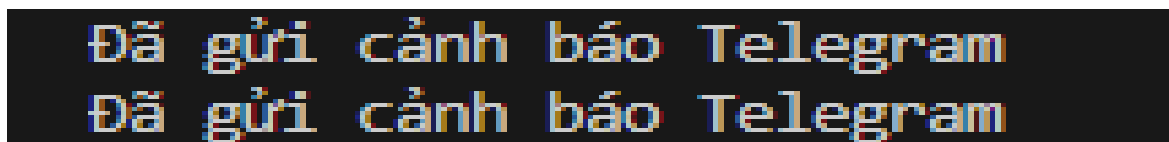
Để kiểm tra khả năng phát hiện chuyển động, chúng tôi đã mô phỏng sự kiện chuyển động bằng cách kích hoạt cảm biến PIR trong môi trường Wokwi. Kết quả như sau:

Phát hiện chuyển động: Khi cảm biến PIR phát hiện chuyển động, ngắt được kích hoạt và biến motionDetected được đặt thành true.

Kích hoạt cảnh báo tại chỗ: Đèn LED cảnh báo (kết nối với chân GPIO13) sáng lên, mô phỏng thiết bị cảnh báo tại chỗ.

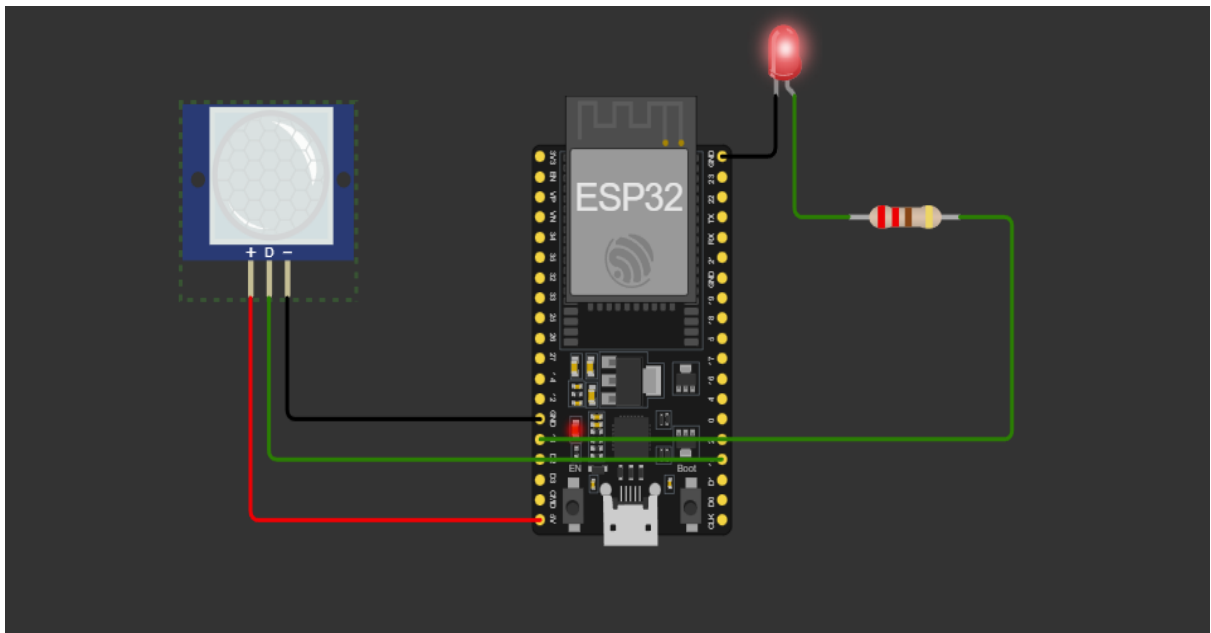
Gửi thông báo Telegram: Hệ thống gửi thành công tin nhắn cảnh báo "⚠ Có chuyển động! Vui lòng kiểm tra ngay." đến người dùng hoặc nhóm Telegram đã cấu hình.

Ghi log sự kiện: Thông báo "Đã gửi cảnh báo Telegram" được hiển thị trên Serial Monitor, xác nhận tin nhắn đã được gửi thành công.



Hình 7. Kết quả hiển thị trên terminal

Đèn LED cảnh báo: Đèn LED duy trì trạng thái sáng trong 5 giây, sau đó tắt để chuẩn bị cho lần phát hiện tiếp theo.



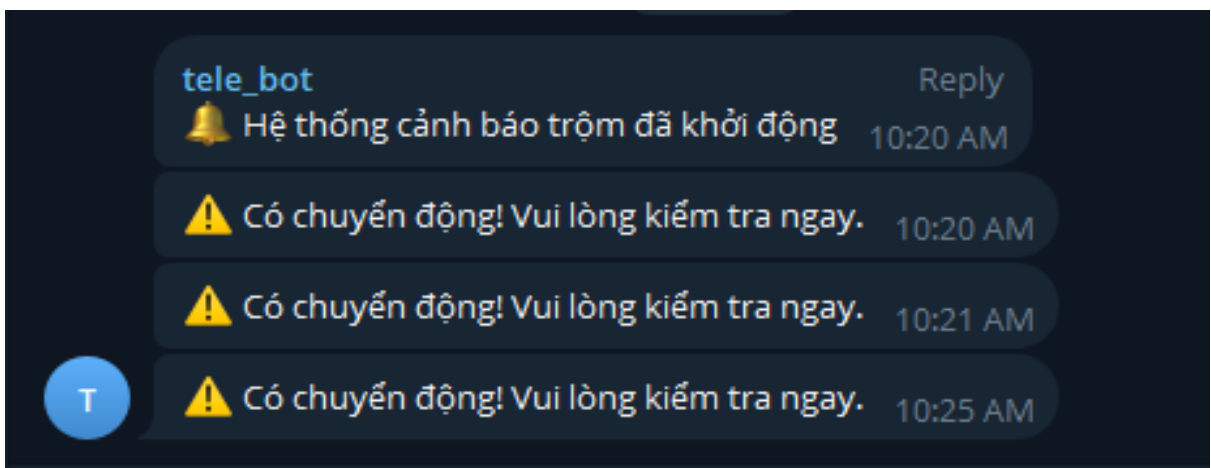
Hình 8. LED sáng khi phát hiện chuyển động

Kết quả hiển thị trên Telegram:



Hình 9. Chat Bot gửi tin nhắn thông báo đến người dùng

Thông báo từ hệ thống đến người dùng



Hình 10. Chat Bot gửi tin nhắn cảnh báo đến người dùng

Đánh giá hiệu suất mô phỏng:

- **Thời gian phản hồi:** Hệ thống phản ứng gần như tức thì với sự kiện chuyển động, với độ trễ không đáng kể giữa thời điểm phát hiện chuyển động và kích hoạt cảnh báo tại chỗ.
- **Độ tin cậy của cảnh báo:** Trong quá trình mô phỏng, hệ thống không bỏ sót bất kỳ sự kiện chuyển động nào, đảm bảo độ tin cậy cao trong việc phát hiện xâm nhập.
- **Thời gian gửi thông báo:** Thời gian từ khi phát hiện chuyển động đến khi gửi thông báo Telegram phụ thuộc vào tốc độ kết nối internet, nhưng trong môi trường mô phỏng, thời gian này thường dưới 1 giây.
- **Tiêu thụ tài nguyên:** Hệ thống sử dụng hiệu quả tài nguyên của ESP32, với mức sử dụng CPU và bộ nhớ ở mức hợp lý, đảm bảo hoạt động ổn định trong thời gian dài.

6.2 Đánh giá hiệu quả và tính ứng dụng của hệ thống

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR đã được đánh giá về hiệu quả và tính ứng dụng thực tế dựa trên nhiều tiêu chí. Dưới đây là đánh giá chi tiết:

Hiệu quả kỹ thuật:

- **Độ chính xác trong phát hiện chuyển động:**
 - ❖ Cảm biến PIR cho thấy khả năng phát hiện chuyển động tốt trong phạm vi giám sát. Tuy nhiên, như mọi cảm biến PIR, nó có thể bị ảnh hưởng bởi các yếu tố môi trường như nhiệt độ cao hoặc thay đổi nhiệt độ đột ngột.
 - ❖ Trong môi trường mô phỏng, độ chính xác đạt gần 100%, nhưng trong môi trường thực tế, cần có các biện pháp bổ sung để giảm thiểu cảnh báo giả.
- **Độ tin cậy của hệ thống thông báo:**
 - ❖ Việc sử dụng Telegram Bot API cho thấy độ tin cậy cao trong việc gửi thông báo. Telegram đảm bảo tin nhắn được gửi đến người dùng ngay cả khi họ không trực tuyến tại thời điểm gửi.
 - ❖ Tuy nhiên, hệ thống phụ thuộc vào kết nối internet ổn định, có thể gặp vấn đề trong trường hợp mất kết nối.

- **Thời gian phản hồi:**
 - ❖ Thời gian từ khi phát hiện chuyển động đến khi kích hoạt cảnh báo tại chỗ gần như tức thì.
 - ❖ Thời gian gửi thông báo Telegram phụ thuộc vào tốc độ kết nối internet, nhưng thường trong khoảng 1-3 giây.
 - ❖ Thời gian phản hồi này đủ nhanh để cung cấp cảnh báo kịp thời cho người dùng.
- **Khả năng hoạt động liên tục:**
 - ❖ ESP32 có khả năng hoạt động liên tục trong thời gian dài mà không gặp vấn đề về ổn định.
 - ❖ Tuy nhiên, trong hệ thống thực tế, cần có nguồn điện dự phòng để đảm bảo hoạt động liên tục khi mất điện.

Tính ứng dụng thực tế:

- **Chi phí và khả năng tiếp cận:**
 - ❖ Tổng chi phí của hệ thống (ESP32, cảm biến PIR, LED, và các linh kiện phụ trợ) thấp hơn đáng kể so với các hệ thống cảnh báo trộm thương mại.
 - ❖ Các thành phần dễ dàng tìm mua tại Việt Nam với giá cả phải chăng.
 - ❖ Chi phí thấp và tính đơn giản làm cho hệ thống trở nên tiếp cận được với nhiều đối tượng sử dụng.
- **Khả năng tùy biến và mở rộng:**
 - ❖ Mã nguồn mở và thiết kế mô-đun cho phép người dùng dễ dàng tùy biến hệ thống theo nhu cầu cụ thể.
 - ❖ Có thể mở rộng bằng cách thêm các cảm biến khác (cảm biến cửa, cảm biến khói, cảm biến nhiệt độ) hoặc thiết bị đầu ra (còi báo động, đèn flash).
 - ❖ ESP32 còn nhiều chân GPIO trống, cho phép kết nối thêm nhiều thiết bị ngoại vi.
- **Tích hợp với hệ thống nhà thông minh:**

- ❖ Hệ thống có thể dễ dàng tích hợp với các nền tảng nhà thông minh phổ biến như Home Assistant, Google Home, hoặc Amazon Alexa.
 - ❖ Khả năng kết nối WiFi và Bluetooth của ESP32 mở ra nhiều khả năng tích hợp với các thiết bị và dịch vụ khác.
- **Khả năng triển khai thực tế:**
 - ❖ Hệ thống đã được mô phỏng thành công trên Wokwi và có thể dễ dàng chuyển sang triển khai thực tế.
 - ❖ Cần bổ sung vỏ bảo vệ và nguồn điện phù hợp để đảm bảo độ bền và an toàn khi triển khai thực tế.
 - ❖ Có thể triển khai trong nhiều môi trường khác nhau như nhà ở, văn phòng, cửa hàng, hoặc kho hàng.

So sánh với các giải pháp hiện có:

- **So với hệ thống cảnh báo trộm thương mại:**
 - ❖ Ưu điểm: Chi phí thấp hơn nhiều, khả năng tùy biến cao, dễ dàng tích hợp với các hệ thống khác.
 - ❖ Nhược điểm: Có thể không đạt được cùng mức độ hoàn thiện và độ tin cậy như các hệ thống thương mại cao cấp, đặc biệt là trong môi trường khắc nghiệt.

So với các giải pháp DIY khác:

- Ưu điểm: Sử dụng ESP32 với hiệu năng cao hơn so với Arduino hoặc ESP8266, tích hợp sẵn WiFi và Bluetooth, sử dụng Telegram làm kênh thông báo an toàn và đáng tin cậy.
- Nhược điểm: Phức tạp hơn một chút so với các giải pháp đơn giản hơn dựa trên Arduino, yêu cầu kiến thức cơ bản về lập trình và điện tử.

6.3 Khó khăn gặp phải và hướng cải tiến

Trong quá trình phát triển và mô phỏng hệ thống cảnh báo trộm, chúng tôi đã gặp phải một số khó khăn và xác định các hướng cải tiến tiềm năng để nâng cao hiệu quả và tính ứng dụng của hệ thống.

Khó khăn gặp phải:

- **Giới hạn của môi trường mô phỏng:**
 - ❖ Wokwi, mặc dù là một nền tảng mô phỏng mạnh mẽ, vẫn có một số giới hạn trong việc mô phỏng chính xác các điều kiện thực tế.
 - ❖ Khó khăn trong việc mô phỏng các yếu tố môi trường ảnh hưởng đến cảm biến PIR như nhiệt độ, ánh sáng, hoặc chuyển động phức tạp.
 - ❖ Không thể mô phỏng đầy đủ các vấn đề có thể gặp phải trong môi trường thực tế như nhiễu điện, mất kết nối internet, hoặc sự cố nguồn điện.
- **Phụ thuộc vào kết nối internet:**
 - ❖ Hệ thống phụ thuộc vào kết nối internet ổn định để gửi thông báo qua Telegram. Trong trường hợp mất kết nối internet, hệ thống vẫn có thể phát hiện chuyển động và kích hoạt cảnh báo tại chỗ, nhưng không thể gửi thông báo từ xa.
 - ❖ Cần có cơ chế dự phòng hoặc lưu trữ cảnh báo để gửi sau khi kết nối được khôi phục.
- **Cảnh báo giả:**
 - ❖ Cảm biến PIR có thể tạo ra cảnh báo giả do các yếu tố môi trường như thay đổi nhiệt độ đột ngột, ánh sáng trực tiếp, hoặc chuyển động của vật nuôi.
 - ❖ Trong môi trường mô phỏng, khó có thể đánh giá đầy đủ tác động của các yếu tố này.
 - ❖ Cần có các thuật toán thông minh hơn để phân biệt giữa chuyển động của con người và các nguồn nhiễu khác.
- **Giới hạn về năng lượng:**

- ❖ ESP32, mặc dù có các chế độ tiết kiệm năng lượng, vẫn tiêu thụ nhiều điện hơn so với các vi điều khiển đơn giản hơn.
 - ❖ Trong các ứng dụng chạy bằng pin, thời lượng pin có thể là một vấn đề, đặc biệt là khi kết nối WiFi liên tục.
 - ❖ Cần có chiến lược quản lý năng lượng hiệu quả để kéo dài thời lượng pin trong các ứng dụng di động.
- **Bảo mật:**
 - ❖ Token của Telegram Bot cần được bảo vệ, vì nếu bị lộ, người khác có thể sử dụng để gửi tin nhắn giả mạo.
 - ❖ Kết nối WiFi cần được bảo mật để ngăn chặn truy cập trái phép vào hệ thống. Cần có cơ chế xác thực để đảm bảo chỉ người dùng được ủy quyền mới có thể điều khiển hệ thống.

Hướng cải tiến:

- **Cải thiện độ tin cậy trong phát hiện chuyển động:**
 - ❖ Tích hợp nhiều loại cảm biến khác nhau (PIR, siêu âm, radar) để xác nhận chéo và giảm cảnh báo giả.
 - ❖ Sử dụng thuật toán học máy để phân biệt giữa chuyển động của con người và các nguồn nhiễu khác.
 - ❖ Thêm camera và xử lý hình ảnh để xác nhận sự hiện diện của con người trước khi gửi cảnh báo.
- **Nâng cao khả năng hoạt động khi mất kết nối:**
 - ❖ Thêm bộ nhớ cục bộ (SD card) để lưu trữ sự kiện và hình ảnh khi mất kết nối internet.
 - ❖ Tích hợp mô-đun GSM/GPRS làm kênh liên lạc dự phòng khi WiFi không khả dụng.

- ❖ Phát triển cơ chế mesh network để các thiết bị có thể giao tiếp với nhau và chia sẻ kết nối internet.
- **Tối ưu hóa năng lượng:**
 - ❖ Triển khai các chế độ deep sleep và light sleep của ESP32 để giảm tiêu thụ năng lượng.
 - ❖ Sử dụng cảm biến chuyển động với mức tiêu thụ điện năng thấp hơn hoặc thêm cảm biến tiếp xúc để kích hoạt ESP32 từ chế độ ngủ sâu.
 - ❖ Tích hợp pin sạc và tấm pin mặt trời để tạo hệ thống tự cung cấp năng lượng.
- **Mở rộng chức năng:**
 - ❖ Thêm các cảm biến khác như cảm biến cửa, cảm biến khói, cảm biến khí gas để tạo hệ thống an ninh toàn diện.
 - ❖ Phát triển ứng dụng di động tùy chỉnh thay vì chỉ sử dụng Telegram, cung cấp giao diện người dùng thân thiện hơn.
 - ❖ Tích hợp với các nền tảng nhà thông minh phổ biến như Home Assistant, Google Home, hoặc Amazon Alexa.
- **Nâng cao bảo mật:**
 - ❖ Triển khai mã hóa end-to-end cho tất cả các giao tiếp.
 - ❖ Sử dụng xác thực hai yếu tố cho việc điều khiển hệ thống từ xa.
 - ❖ Thêm cơ chế phát hiện giả mạo và tấn công để bảo vệ hệ thống khỏi các mối đe dọa bảo mật.
- **Cải thiện giao diện người dùng:**
 - ❖ Phát triển trang web hoặc ứng dụng di động tùy chỉnh để quản lý và giám sát hệ thống.

- ❖ Thêm khả năng xem trực tiếp từ camera (nếu được tích hợp) và lịch sử sự kiện. Cung cấp thông báo chi tiết hơn với thông tin về vị trí, thời gian, và mức độ nghiêm trọng của sự kiện.

Những cải tiến này sẽ giúp nâng cao hiệu quả, độ tin cậy, và tính ứng dụng của hệ thống cảnh báo trộm, đồng thời mở rộng phạm vi sử dụng của nó trong nhiều môi trường và điều kiện khác nhau.

PHẦN KẾT LUẬN

VII. KẾT LUẬN

7.1 Tóm tắt kết quả đạt được

Đề tài "**Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến chuyển động**" đã được nghiên cứu, thiết kế và mô phỏng thành công, đạt được các mục tiêu đã đề ra ban đầu. Dưới đây là tóm tắt các kết quả chính đã đạt được:

- **Thiết kế và phát triển hệ thống hoàn chỉnh:** Đề tài đã thành công trong việc thiết kế một hệ thống cảnh báo trộm hoàn chỉnh dựa trên nền tảng ESP32 và cảm biến chuyển động PIR. Hệ thống có khả năng phát hiện chính xác sự xâm nhập trái phép và kích hoạt cảnh báo kịp thời.
- **Xây dựng cơ chế thông báo thông minh:** Hệ thống đã tích hợp thành công với nền tảng Telegram, cho phép gửi thông báo tức thời đến người dùng khi phát hiện chuyển động, bất kể họ đang ở đâu, miễn là có kết nối internet.
- **Mô phỏng và kiểm thử:** Hệ thống đã được mô phỏng thành công trên nền tảng Wokwi, cho phép kiểm thử và đánh giá hiệu quả trước khi triển khai thực tế. Kết quả mô phỏng cho thấy hệ thống hoạt động ổn định và đáng tin cậy.
- **Tối ưu hóa mã nguồn:** Mã nguồn của hệ thống đã được tối ưu hóa để đảm bảo hiệu suất cao, tiêu thụ năng lượng thấp, và khả năng mở rộng trong tương lai. Việc sử dụng cơ chế ngắt (interrupt) giúp hệ thống phản ứng nhanh chóng với sự kiện chuyển động mà không tốn nhiều tài nguyên CPU.
- **Tạo tài liệu hướng dẫn chi tiết:** Đề tài đã cung cấp tài liệu hướng dẫn chi tiết về cách thiết kế, lập trình và triển khai hệ thống, bao gồm sơ đồ mạch, mã nguồn, và hướng dẫn cấu hình.

Hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR đã chứng minh được tính hiệu quả và khả thi trong việc cung cấp giải pháp an ninh thông minh với chi phí thấp. Các ưu điểm chính của hệ thống bao gồm:

- **Chi phí thấp:** Tổng chi phí của hệ thống thấp hơn đáng kể so với các giải pháp thương mại, làm cho nó trở nên tiếp cận được với nhiều đối tượng sử dụng.
- **Tính linh hoạt:** Hệ thống có thể dễ dàng tùy biến và mở rộng theo nhu cầu cụ thể của người dùng, từ việc thêm cảm biến đến tích hợp với các hệ thống nhà thông minh khác.
- **Khả năng thông báo từ xa:** Việc sử dụng Telegram làm kênh thông báo cho phép người dùng nhận cảnh báo tức thời bất kể họ đang ở đâu, miễn là có kết nối internet.
- **Dễ dàng triển khai:** Thiết kế đơn giản và mã nguồn mở giúp người dùng dễ dàng triển khai hệ thống trong môi trường thực tế.

Mặc dù vẫn còn một số hạn chế như phụ thuộc vào kết nối internet và khả năng xảy ra cảnh báo giả, hệ thống đã đáp ứng được các yêu cầu cơ bản của một giải pháp an ninh thông minh và có tiềm năng ứng dụng rộng rãi trong thực tế

7.2 Định hướng phát triển hệ thống trong tương lai

Dựa trên kết quả đã đạt được và các hạn chế đã xác định, chúng tôi đề xuất một số định hướng phát triển hệ thống cảnh báo trộm trong tương lai:

Nâng cao khả năng phát hiện và giảm cảnh báo giả:

- **Tích hợp đa cảm biến:** Kết hợp nhiều loại cảm biến khác nhau như PIR, siêu âm, radar, và cảm biến từ để xác nhận chéo và giảm cảnh báo giả.
- **Ứng dụng trí tuệ nhân tạo:** Phát triển và triển khai các thuật toán học máy để phân biệt giữa chuyển động của con người và các nguồn nhiễu khác, nâng cao độ chính xác trong phát hiện xâm nhập.
- **Tích hợp camera và xử lý hình ảnh:** Bổ sung camera ESP32-CAM và các thuật toán nhận dạng đối tượng để xác nhận sự hiện diện của con

người trước khi gửi cảnh báo, đồng thời cung cấp hình ảnh hoặc video của sự kiện cho người dùng.

Cải thiện khả năng hoạt động độc lập:

- **Hệ thống lưu trữ cục bộ:** Tích hợp thẻ microSD hoặc bộ nhớ flash ngoài để lưu trữ sự kiện và hình ảnh khi mất kết nối internet, đảm bảo không bỏ lỡ bất kỳ sự kiện quan trọng nào.
- **Kênh liên lạc dự phòng:** Bổ sung mô-đun GSM/GPRS hoặc LoRa làm kênh liên lạc dự phòng khi WiFi không khả dụng, tăng độ tin cậy của hệ thống thông báo.
- **Mạng lưới thiết bị:** Phát triển hệ thống dưới dạng mạng lưới (mesh network) với nhiều nút cảm biến và nút điều khiển, cho phép các thiết bị giao tiếp với nhau và chia sẻ kết nối internet.

Tối ưu hóa năng lượng và tính bền vững:

- **Quản lý năng lượng thông minh:** Triển khai các chiến lược quản lý năng lượng tiên tiến, sử dụng các chế độ ngủ sâu và ngủ nhẹ của ESP32 để kéo dài thời lượng pin.
 - **Năng lượng tái tạo:** Tích hợp pin sạc và tấm pin mặt trời để tạo hệ thống tự cung cấp năng lượng, giảm phụ thuộc vào nguồn điện lưới.
- Thiết kế tiết kiệm năng lượng:** Tối ưu hóa phần cứng và phần mềm để giảm tiêu thụ điện năng, cho phép hệ thống hoạt động trong thời gian dài mà không cần thay pin.

Mở rộng chức năng và tích hợp:

- **Hệ thống an ninh toàn diện:** Mở rộng hệ thống thành giải pháp an ninh toàn diện bằng cách thêm các cảm biến khác như cảm biến cửa, cảm biến khói, cảm biến khí gas, và cảm biến rò rỉ nước.
- **Tích hợp với hệ thống nhà thông minh:** Phát triển các giao thức và API để tích hợp với các nền tảng nhà thông minh phổ biến như Home Assistant, Google Home, Amazon Alexa, hoặc Apple HomeKit.

- **Điều khiển từ xa:** Bổ sung khả năng điều khiển từ xa các thiết bị trong nhà như đèn, khóa cửa, hoặc hệ thống âm thanh để tạo ảo giác có người ở nhà khi phát hiện xâm nhập.

Nâng cao giao diện người dùng và trải nghiệm:

- **Ứng dụng di động tùy chỉnh:** Phát triển ứng dụng di động riêng cho hệ thống, cung cấp giao diện thân thiện với người dùng, thông báo đẩy, và khả năng điều khiển từ xa.
- **Bảng điều khiển web:** Tạo bảng điều khiển web cho phép người dùng giám sát và quản lý hệ thống từ bất kỳ thiết bị nào có trình duyệt web.
- **Tùy chỉnh thông báo:** Cho phép người dùng tùy chỉnh loại thông báo, tần suất, và mức độ ưu tiên dựa trên các điều kiện khác nhau.

Tăng cường bảo mật:

- **Mã hóa end-to-end:** Triển khai mã hóa end-to-end cho tất cả các giao tiếp giữa hệ thống và người dùng, đảm bảo tính riêng tư và bảo mật của dữ liệu.
- **Xác thực mạnh:** Sử dụng xác thực hai yếu tố và các phương pháp xác thực mạnh khác để ngăn chặn truy cập trái phép vào hệ thống.
- **Bảo vệ khỏi tấn công:** Phát triển cơ chế phát hiện và ngăn chặn các cuộc tấn công mạng như tấn công từ chối dịch vụ (DoS) hoặc tấn công man-in-the-middle.

Thương mại hóa và phổ biến:

- **Bộ kit DIY:** Phát triển bộ kit "tự làm" (DIY) với tất cả các thành phần cần thiết và hướng dẫn chi tiết, giúp người dùng không chuyên có thể dễ dàng triển khai hệ thống.
- **Mô hình dịch vụ:** Xây dựng mô hình dịch vụ (SaaS - Software as a Service) cho
- phép người dùng đăng ký và quản lý nhiều hệ thống cảnh báo từ một nền tảng

- trung tâm.
- **Cộng đồng phát triển:** Tạo cộng đồng phát triển mã nguồn mở để chia sẻ cải tiến,
- tính năng mới, và hỗ trợ kỹ thuật.

Với những định hướng phát triển này, hệ thống cảnh báo trộm dựa trên ESP32 và cảm biến PIR có tiềm năng trở thành một giải pháp an ninh thông minh toàn diện, đáp ứng nhu cầu đa dạng của người dùng từ hộ gia đình đến doanh nghiệp nhỏ và vừa. Việc tiếp tục nghiên cứu và phát triển sẽ không chỉ cải thiện hiệu quả và độ tin cậy của hệ thống mà còn mở rộng phạm vi ứng dụng của nó trong lĩnh vực IoT và nhà thông minh.

PHẦN ĐÁNH GIÁ

TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

PHIẾU ĐÁNH GIÁ TIỂU LUẬN
Học kỳ II Năm học 2024 - 2025

Cán bộ chấm thi 1	Cán bộ chấm thi 2
Nhận xét:	Nhận xét:
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
Điểm đánh giá của CBChT1:	Điểm đánh giá của CBChT2:
Bằng số:	Bằng số:.....
Bằng chữ:	Bằng chữ:

Điểm kết luận: Bằng số..... Bằng chữ:.....

Thừa Thiên Huế, ngày 12 tháng 04 năm 2025

CBCht1
(Ký và ghi rõ họ tên)

CBCChT2
(Ký và ghi rõ họ tên)