BỘ THÔNG TIN TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Đề cương môn học IOT và ứng dụng

Đề tài: Xây dựng hệ thống phát hiện xâm nhập sử dụng module ESP32-CAM

Nhóm: 05

Thành viên:	**Nguyễn Vinh Hiển**	**B21DCCN047**
	Trần Việt Hoàng	**B21DCCN057**
	Nguyễn Đức Lộc	**B21DCCN075**

Giảng viên hướng dẫn: **Trần Thị Thanh Thủy**

Hà Nội – 2024

Lời Cảm Ơn

Lời đầu tiên, nhóm em xin được gửi lời cảm ơn chân thành nhất đến cô Trần Thị Thanh Thủy. Trong quá trình học tập và tìm hiểu môn IOT và ứng dụng em đã nhận được rất nhiều sự quan tâm, giúp đỡ, hưỡng dẫn tâm huyết và tận tình của cô. cô đã giúp em tích lũy thêm nhiều kiến thức về kỹ năng quan trọng này để có thể hoàn thành được bài tiểu luận về đề tài: Xây dựng hệ thống phát hiện xâm nhập sử dụng module ESP32-CAM.

Trong quá trình làm bài chắc chắn khó tránh khỏi những thiếu sót. Do đó, em kính mong nhận được những lời góp ý của cô để bài tiểu luận của em ngày càng hoàn thiện hơn.

Em xin chân thành cảm ơn!

Mục lục

Danh sách các từ viết tắt

Viết tắt	Tiếng Anh	Giải nghĩa
IoT	Internet of Things	Mạng lưới các thiết bị thông minh kết nối với nhau qua internet, như ESP32, cảm biến PIR và các thiết bị khác.
HTTP	Hypertext Transfer Protocol	Giao thức truyền tải siêu văn bản, thường được sử dụng để truyền dữ liệu giữa ESP32 và server.
PIR	Passive Infrared Sensor	Cảm biến hồng ngoại thụ động, sử dụng để phát hiện chuyển động dựa trên sự thay đổi nhiệt độ của vật thể.
ESP	Espressif Systems Platform	Nền tảng của các chip vi điều khiển do Espressif Systems
SSID	Service Set Identifier	Tên mạng không dây mà ESP32 sẽ kết nối.
UART	Universal Asynchronous Receiver/Transmitter	Giao thức truyền dữ liệu nối tiếp, có thể sử dụng cho giao tiếp giữa ESP32 và các module khác.
GPIO	General Purpose Input/Output	Các chân I/O của ESP32 dùng để giao tiếp với cảm biến PIR và điều khiển chuông hoặc các thiết bị ngoại vi khác.

Danh sách các hình vẽ, bảng biểu

Mở đầu

1. Tên đề tài

Xây dựng hệ thống phát hiện xâm nhập sử dụng module ESP32-CAM

2. Lí do chọn đề tài

Là sinh viên ngành Công nghệ Thông tin, quan tâm đến các lĩnh vực IoT (Internet of Things) và bảo mật, chúng tôi nhận thấy việc phát triển các hệ thống giám sát thông minh có ý nghĩa rất lớn trong bối cảnh xã hội hiện đại. Đề tài "Xây dựng hệ thống phát hiện xâm nhập sử dụng module ESP32-CAM" là sự kết hợp giữa công nghệ IoT và các giải pháp bảo mật nhằm tạo ra một hệ thống có khả năng giám sát và bảo vệ tài sản một cách tự động và hiệu quả.

Lựa chọn đề tài này xuất phát từ những lý do sau:

- Tính thực tiễn cao: Việc ứng dụng IoT trong an ninh và giám sát đang trở thành xu hướng quan trọng, đặc biệt trong việc bảo vệ nhà cửa, văn phòng, và các khu vực công cộng. Hệ thống phát hiện xâm nhập có thể mang lại giải pháp tiết kiệm chi phí và dễ dàng triển khai cho nhiều đối tượng người dùng.
- Khả năng ứng dụng đa dạng: ESP32-CAM là một module có khả năng mạnh mẽ trong xử lý hình ảnh và tích hợp kết nối Wi-Fi. Hệ thống này có thể được mở rộng và tích hợp thêm nhiều tính năng khác, từ cảnh báo theo thời gian thực đến nhận diện đối tượng, giúp tăng cường hiệu quả bảo mật.
- Học hỏi và phát triển kỹ năng: Đề tài không chỉ giúp chúng tôi phát triển kiến thức về lập trình nhúng, xử lý hình ảnh mà còn củng cố hiểu biết về truyền thông không dây và giao tiếp giữa các thiết bị IoT. Đây là nền tảng vững chắc để áp dụng trong các dự án thực tế và

chuẩn bị cho những cơ hội nghề nghiệp trong tương lai, đặc biệt trong lĩnh vực trí tuệ nhân tạo và an ninh mạng.

- Tối ưu hóa chi phí: ESP32-CAM là một giải pháp rẻ tiền nhưng hiệu quả cao. Điều này giúp những sinh viên như chúng tôi dễ dàng tiếp cận và thực hiện dự án mà không cần đầu tư nhiều về tài chính, đồng thời cho thấy tiềm năng của các giải pháp công nghệ chi phí thấp trong việc giải quyết những vấn đề an ninh.

Với những lý do trên, chúng tôi tin rằng đề tài này không chỉ mang lại những giá trị thực tiễn cho cuộc sống mà còn là cơ hội quý báu để chúng tôi phát triển kiến thức, kỹ năng và kinh nghiệm trong lĩnh vực IoT và bảo mật.

3. Cấu trúc báo cáo

Báo cáo này gồm 5 phần:

- Mở đầu
- Chương 1: Cơ sở lý thuyết
- Chương 2: Thiết kế hệ thống
- Chương 3: Kết quả và kết luận
- Tài liệu tham khảo

Chương 1. Tổng quan

1.1. Tổng quan về IOT

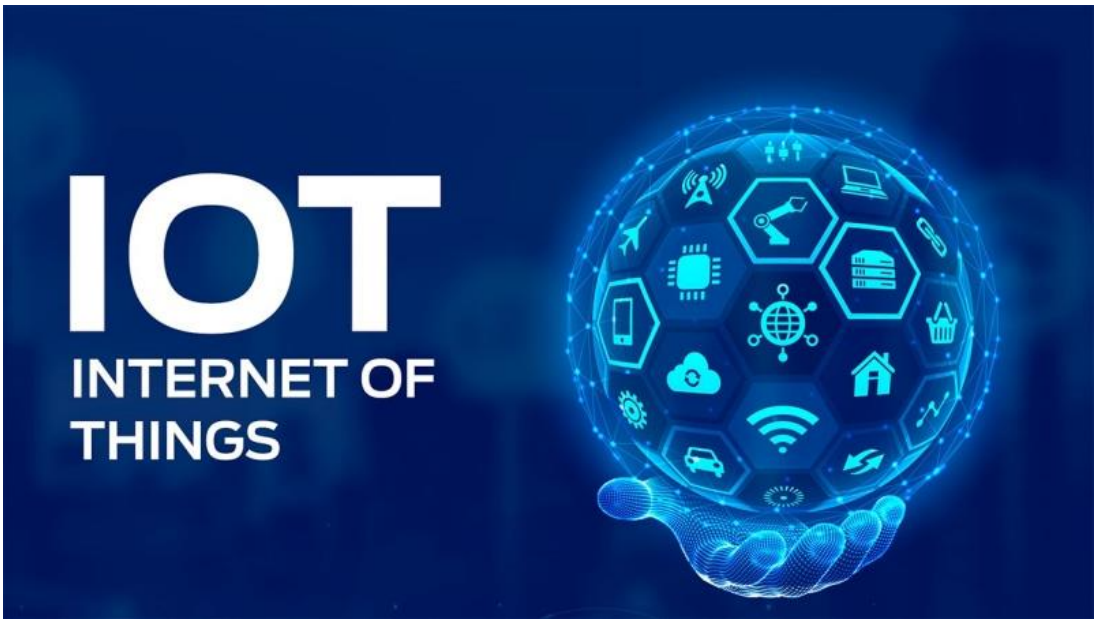
1.1.1 Khái niệm IOT

IOT viết tắt của thuật ngữ "Internet of Thing" đề cập đến hàng tỷ thiết bị vật lý được kết nối thông qua internet, thu thập và chia sẻ dữ liệu. Với IoT, các thiết bị thông minh có thể giao tiếp với nhau và với các hệ thống điều khiển từ xa, tạo ra một hệ sinh thái kết nối thông minh, đem lại nhiều tiện ích cho con người. Các thiết bị Iot bao gồm các cảm biến, máy tính nhúng, các thiết bị điện tử gia đình, ô tô tự lái, nhà thông minh, các thiết bị giám sát an ninh. Các thiết bị có thể giao tiếp với nhau thông qua internet để thực hiện các chức năng đa dạng như giám sát và điều khiển, tự động hóa, phân tích dữ liệu, và cung cấp dịch vụ thông minh cho người sử dụng.

Hệ thống IoT cho phép vật được cảm nhận hoặc được điều khiển từ xa thông qua hạ tầng mạng hiện hữu, tạo cơ hội cho thế giới thực được tích hợp trực tiếp hơn vào hệ thống điện toán, hệ quả là hiệu năng, độ tin cậy và lợi ích kinh tế được tăng cường bên cạnh việc giảm thiểu sự can dự của con người. Khi IoT được gia tố cảm biến và cơ cấu chấp hành, công nghệ này trở thành một dạng thức của hệ thống ảo-thực với tính tổng quát cao hơn, bao gồm luôn cả những công nghệ như điện lưới thông minh, nhà máy điện ảo, nhà thông minh, vận tải thông minh và thành phố thông minh. Mỗi vật được nhận dạng riêng biệt trong hệ thống điện toán nhúng và có khả năng phối hợp với nhau trong cùng hạ tầng Internet hiện hữu1.1.2 Thuật toán học máy

Về cơ bản, Internet Vạn Vật cung cấp kết nối chuyên sâu cho các thiết bị, hệ thống và dịch vụ, kết nối này mang hiệu quả vượt trội so với kiểu truyền tải máy-máy (M2M), đồng thời hỗ trợ đa

dạng giao thức, miền (domain), và ứng dụng. Kết nối các thiết bị nhúng này (luôn cả các vật dụng thông minh), được kỳ vọng sẽ mở ra kỷ nguyên tự động hóa trong hầu hết các ngành, từ những ứng dụng chuyên sâu như điện lưới thông minh, mở rộng tới những lĩnh vực khác như nhà thông minh, thành phố thông minh. [1]



Hình 1.1: Internet of Things (IoT)

1.1.2. Lịch sử hình thành

Khái niệm về một mạng lưới thiết bị được kết nối với nhau đã được thảo luận vào đầu năm 1982, với một máy bán hàng tự động Coke được thực hiện ở Đại học Carnegie Mellon trở thành thiết bị kết nối Internet đầu tiên trên thế giới.

Thuật ngữ "Internet of things" được sử dụng lần đầu tiên bởi Kevin Ashton vào năm 1999. Sau đó IoT trải qua nhiều giai đoạn và có bước phát triển nhảy vọt cho đến ngày nay.

1.1.3. Kiến trúc của một hệ thống IoT

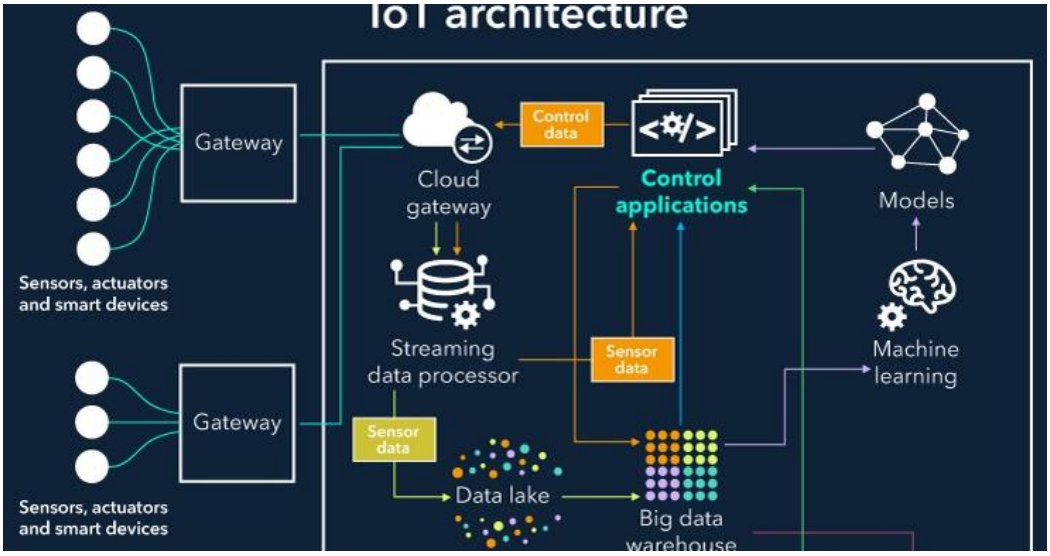
Kiến trúc IoT được đại diện cơ bản bởi 4 phần:

• Things (thiết bị) : gồm các thiết bị cuối tham gia vào mạng IoT (quạt, đèn,..), các thiết bị có thể được tích hợp với một cảm biến hoặc nhận lệnh tự người dùng.

• Gateway (Trạm kết nối): là cầu nối giữa các công nghệ truyền thông khác nhau

• Network and Cloud (Hạ tầng mạng): Gồm một hệ thống lớn các máy chủ, hệ thống lưu trữ và mạng ảo hóa được kết nối. Chịu trách nhiệm sử lý, chỉ huy và phân tích các dữ liệu thu thập được

• Services-creation and Solution Layers (Bộ phân tích và xử lý dữ liệu): Dữ liệu được thu thập phân tích chuyển đổi thành các thông tin hữu ích, hỗ trợ người dùng đưa ra quyết định quan trọng



Hình 1.2: Kiến trúc của một hệ thống IoT

1.1.4. Ứng dụng của IoT

Với sự phát triển vượt bậc của internet, Internet of Things có thể ứng dụng được trong bất kì lĩnh vực nào mà chúng ta muốn. Một số lĩnh vực nổi bật hiện nay được ứng dụng IoT nhiều nhất như là:

• Ứng dụng trong quản lý hạ tầng

• Ứng dụng trong y tế

- Ứng dụng trong tự động hóa nhà.
- Ứng dụng trong giao thông
- Phản hồi trong các tinh huống khẩn cấp
- Mua sắm thông minh
- Quản lý các thiết bị cá nhân
- Đồng hồ đo thông minh

Hình 1.3: Ứng dụng của IoT

1.1.5. Ưu và nhược điểm của IoT

*** Ưu điểm:**

- Khả năng truy cập thông tin từ mọi nơi, mọi lúc trên mọi thiết bị.
- Cải thiện kết nối giữa các thiết bị điện tử.
- Chuyển các gói dữ liệu qua mạng được kết nối tiết kiệm thời gian và tiền bạc.

- Tự động hóa các nhiệm vụ giúp cải thiện chất lượng dịch vụ của doanh nghiệp và giảm nhu cầu can thiệp con người.

*** Nhược điểm** :

- Vấn đề bảo mật khi có nhiều thiết bị kết nối
- Việc thu thập dữ liệu là thách thức lớn.
- Nếu hệ thống trung tâm bị hỏng sẽ dẫn tới có khả năng các thiết bị kết nối sẽ bị ảnh hỏng.
- Do không có tiêu chuẩn quốc tế nên dẫn tới các thiết bị từ các nhà sản xuất khác nhau có thể tương thích với nhau

1.2 Tổng quan về hệ thống phát hiện xâm nhập

Hệ thống phát hiện xâm nhập là các giải pháp công nghệ được thiết kế để giám sát, xác định, và phản ứng nhanh chóng trước các hành vi xâm nhập hoặc hoạt động bất thường có khả năng gây hại cho môi trường được bảo vệ, chẳng hạn như cơ sở hạ tầng vật lý hoặc hệ thống thông tin. Các hệ thống này thường áp dụng các kỹ thuật xử lý hình ảnh, máy học và trí tuệ nhân tạo để tăng cường khả năng bảo vệ trong thời gian thực, đặc biệt là trong các ứng dụng an ninh.

1.2.1. Công nghệ Sử dụng trong Phát hiện Xâm nhập

- **Cảm biến hình ảnh và nhận dạng khuôn mặt**: Cảm biến hình ảnh, camera, và công nghệ nhận dạng khuôn mặt là những thành phần cốt lõi của hệ thống phát hiện xâm nhập trong thời đại số. Những hệ thống này có thể ghi lại hình ảnh, phát hiện đối tượng, phân tích và nhận diện các khuôn mặt, từ đó xác định những người có mặt trong khu vực được giám sát.

- **Mô hình máy học và mạng nơ-ron**: Trong các hệ thống hiện đại, mô hình máy học và mạng nơ-ron nhân tạo được huấn luyện để nhận diện các khuôn mặt đã biết hoặc phát hiện ra các khuôn mặt lạ. Các mô hình này thường được huấn luyện trên tập dữ liệu lớn để cải thiện độ chính xác của nhận dạng.
- **Trí tuệ nhân tạo và phân tích thời gian thực**: AI hỗ trợ hệ thống xử lý dữ liệu một cách tự động và phản ứng theo thời gian thực khi có xâm nhập. Điều này cho phép phát hiện và cảnh báo khi nhận diện người lạ hoặc các hành vi bất thường.

1.2.2. Hoạt động của Hệ thống Phát hiện Xâm nhập

- **Thu thập dữ liệu**: Camera sẽ ghi lại dữ liệu hình ảnh từ khu vực được giám sát. Các hệ thống có thể tích hợp nhiều camera hoặc cảm biến để giám sát trên một phạm vi lớn hơn.
- **Nhận diện và phân tích khuôn mặt**: Hệ thống phân tích dữ liệu từ camera để phát hiện khuôn mặt. Các thuật toán nhận diện khuôn mặt sau đó so sánh những khuôn mặt đã ghi lại với cơ sở dữ liệu có sẵn để phân loại thành các nhóm đã biết hoặc chưa biết.
- **Cảnh báo và phản hồi**: Khi hệ thống phát hiện một khuôn mặt không xác định hoặc một hành vi đáng ngờ, hệ thống sẽ tự động gửi cảnh báo đến bộ phận an ninh. Một số hệ thống tích hợp cảnh báo trực tiếp đến điện thoại hoặc qua nền tảng như Telegram, email hoặc tin nhắn SMS, cho phép phản ứng nhanh chóng khi cần thiết.

1.2.3. Các Thách Thức và Khả Năng Nâng Cấp

- **Độ chính xác và khả năng chịu lỗi**: Việc phân biệt giữa người lạ và người trong cơ sở dữ liệu có thể gặp lỗi do chất lượng hình ảnh hoặc các điều kiện ánh sáng không ổn

định. Độ chính xác của hệ thống có thể được cải thiện qua các mô hình huấn luyện và tập dữ liệu lớn hơn.

- **Bảo mật và quyền riêng tư**: Hệ thống giám sát nhận dạng khuôn mặt có thể xâm phạm đến quyền riêng tư nếu dữ liệu không được bảo mật đúng cách. Để tránh rủi ro, các tổ chức cần tuân thủ nghiêm ngặt quy định về bảo vệ dữ liệu cá nhân và bảo mật thông tin.
- **Khả năng mở rộng và tích hợp**: Hệ thống phát hiện xâm nhập cần được thiết kế để có thể mở rộng khi nhu cầu giám sát tăng lên, đồng thời phải có khả năng tích hợp với các hệ thống an ninh và cơ sở dữ liệu hiện có.

1.2.4. Ứng dụng trong Thực Tế

Hệ thống phát hiện xâm nhập dựa trên nhận dạng khuôn mặt đang được triển khai trong nhiều lĩnh vực:

- **An ninh nhà máy và doanh nghiệp**: Bảo vệ khu vực sản xuất, văn phòng và khu vực lưu trữ để chống lại trộm cắp hoặc truy cập trái phép.
- **An ninh công cộng**: Các thành phố lớn đang ứng dụng công nghệ này để giám sát và kiểm soát tội phạm, giúp cải thiện sự an toàn cho người dân.
- **Nhà ở và khu dân cư**: Các hệ thống này có thể được tích hợp vào hệ thống an ninh gia đình, giúp phát hiện những người lạ vào khu vực quanh nhà và gửi cảnh báo cho chủ nhà.

1.3 Giới thiệu về phần mềm Arduino

1.3.1. Giới thiệu

Arduino là nền tảng mã nguồn mở, giúp con người xây dựng các ứng dụng điện tử có khả năng liên kết, tương tác với nhau một

cách tốt hơn. Arduino còn có thể xem như một chiếc máy tính thu nhỏ giúp người hin lập trình, thực hiện các dự án điện tử không cần tới công cụ chuyên biệt phục cho quá trình nạp code.

1.3.2. Ứng dụng

Arduino có nhiều ứng dụng trong cuộc sống hin ngày, nhất là trong việc chế tạo các thiết bị điện tử chất lượng cao. Có thể kể đến như:

- Làm Robot. Arduino có khả năng đọc các thiết bị cảm biến, điều khiển động cơ,... nên nó thường được hin để làm bộ xử lý trung tâm của rất nhiều loại robot.

- Game tương tác: Arduino có thể được sử dụng để tương tác với Joystick, màn hình,... khi chơi các game như Tetris, phá gạch, Mario…

- Máy bay không người lái.

- Điều khiển đèn tín hiệu giao thông, làm hiệu ứng đèn Led nhấp nháy trên các biển quảng cáo…

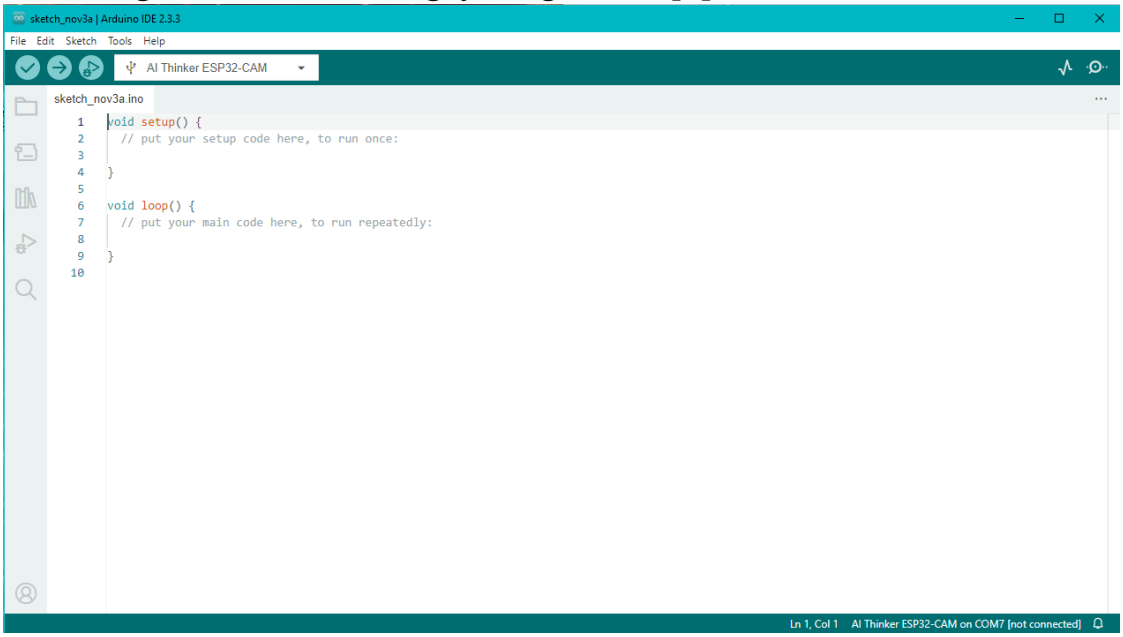
- Điều khiển các thiết bị cảm biến ánh hin, âm thanh.

- Làm máy in 3D, …

1.3.3. Phần mềm Arduino IDE

Arduino IDE là một môi trường phát triển tích hợp đa nền tảng, làm việc cùng với một bộ điều khiển Arduino để viết, biên dịch và tải code lên bo mạch. Phần mềm này cung cấp sự hỗ trợ cho một loạt các bo mạch Arduino như Arduino Uno, Nano, Mega, Pro hay Pro Mini, ... Ngôn ngữ tổng quát cho Arduino C và C++, do đó phần mềm phù hợp cho những lập trình viên đã quen thuộc với cả 2 ngôn ngữ này. Các tính năng như làm nổi bật cú pháp, thụt đầu dòng tự động, ... làm cho nó trở thành một sự thay thế hiện đại cho các

IDE khác. Arduino IDE có thư viện code mẫu quá phong phú, viết chương trình trên Arduino IDE khá dễ dàng cộng thêm OpenSource viết riêng cho Arduino thì ngày càng nhiều. [2]



Hình 1.5: Giao diện phầm mềm Arduino

Đây là công cụ hổ trợ viết code và nạp code cho các bo mạch Arduino cũng như các mạch NodeMCU. Phần mềm được hổ trợ miễn phí cho người dùng , với bản cập nhật mới nhất là Arduino 2.2.3.

1.4 Giới thiệu về cảm biến hồng ngoại PIR

Trên thị trường hiện nay, có rất nhiều loại cảm biến hồng ngoại với các đặc tính khác nhau để phù hợp với nhiều mục đích sử dụng, như cảm biến PIR (Passive Infrared), cảm biến IR, cảm biến hồng ngoại thu phát,…

Cảm biến hồng ngoại PIR (Passive Infrared Sensor) là loại cảm biến được ứng dụng rộng rãi hiện nay nhờ vào chi phí thấp và khả năng dễ dàng lấy dữ liệu qua tín hiệu số. Cảm biến PIR có khả năng phát hiện chuyển động của các vật thể phát nhiệt (như con người, động vật) trong vùng giám sát bằng cách phát hiện sự thay

đổi cường độ hồng ngoại trong môi trường. Cảm biến này hoạt động dựa trên nguyên tắc phát hiện bức xạ hồng ngoại do cơ thể sống phát ra và chuyển đổi nó thành tín hiệu điện để xác định sự hiện diện hoặc di chuyển.

Trong đề tài này, em lựa chọn sử dụng cảm biến hồng ngoại thụ động PIR HC-SR505 vì tính năng phát hiện chuyển động hiệu quả, ổn định, và độ nhạy cao của nó. Module cảm biến chuyển động 12V 5V HC-SR505 (HC-SR505 Mini Infrared PIR Motion Sensor) sử dụng công nghệ hồng ngoại, có độ nhạy cao, nhỏ gọn, tiết kiệm điện năng. Cảm biến HC-SR505 có sự khác biệt với cảm biến chuyển động PIR ở chỗ là nếu có sự chuyển động phát ra hồng ngoại trong khoảng quét của cảm biến thì cảm biến luôn có tín hiệu (3.3V) cho đến khi vùng quét không còn thân nhiệt hồng ngoại (trở về mức Ouput là LOW).

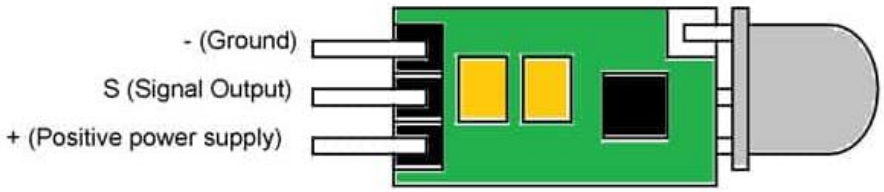
Hình 1.6: Cảm biến chuyển động PIR HC-SR505

- Thông số kỹ thuật:

- Điện áp hoạt động: 4.5~20VDC

- Dòng tiêu thụ: <60uA
- Mức tín hiệu: High 3.3V / Low 0V
- Trigger: repeatable trigger
- Thời gian trễ T sau khi kích hoạt: 8s + -30%
- Góc quét: Max 100 độ (hình nón có tâm là cảm biến)
- Khoảng cách bắt: 3 meters
- Đường kính thấu kính: 10mm
- Kích thước: 10 x 23mm
- Nhiệt độ hoạt động: -20°C - 80°C

- Sơ đồ chân HC-SR505:



Hình 1.7: Sơ đồ chân kết nối của HC-SR505

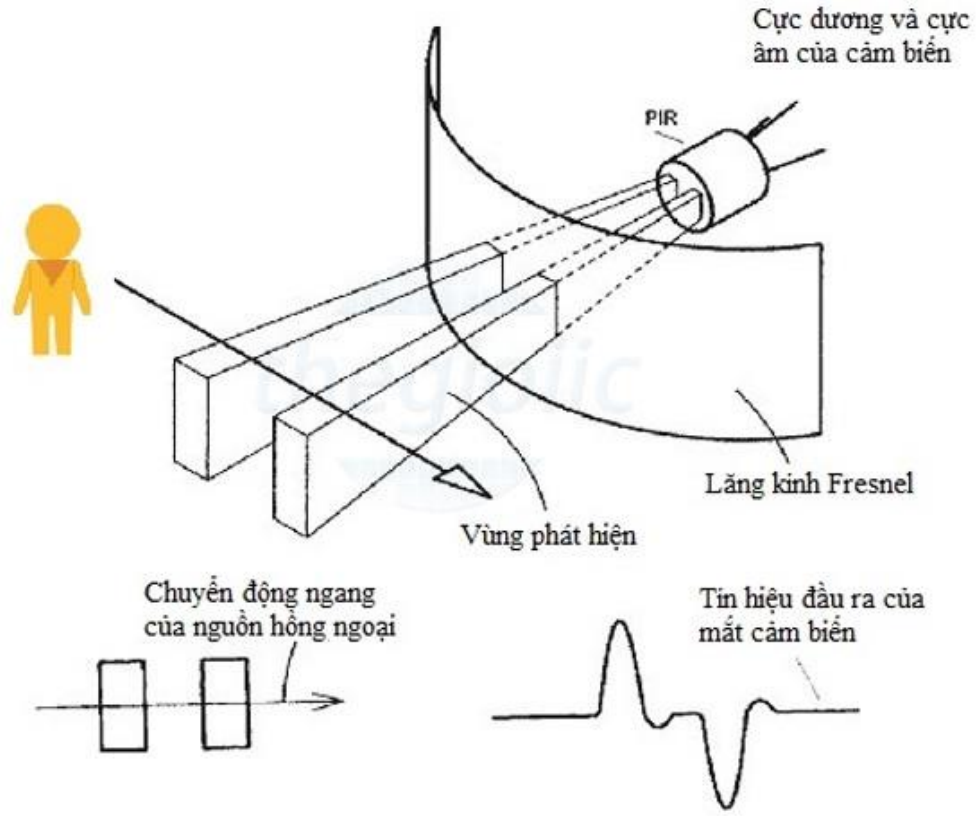
- Nó có ba chân:

- Chân 1 là chân nối đất.
- Chân 2 là chân đầu ra tín hiệu.
- Chân 3 (D +) là chân cấp nguồn. Cấp + 5V vào chân này.

- Nguyên lý hoạt động:

Cảm biến thân nhiệt chuyển động PIR HC-SR505 Mini sẽ xuất ra tín hiệu mức cao (High) khi phát hiện vật thể nhiệt chuyển động trong vùng quét, tín hiệu này sau đó sẽ được giữ ở mức cao trong khoảng thời gian trễ T sau khi kích hoạt, lúc này nếu cảm biến vẫn bắt được tín hiệu sẽ vẫn duy trì chân tín hiệu mức cao trong thời

gian trễ T, chỉ khi trong khoảng thời gian trễ T mà cảm biến không bắt được tín hiệu thì chân tín hiệu cảm biến mới trở về mức thấp (Low).



Hình 1.8: Nguyên lý hoạt động của cảm biến hồng ngoại PIR

1.5 Mạch chuyển USB UART TTL FT232RL

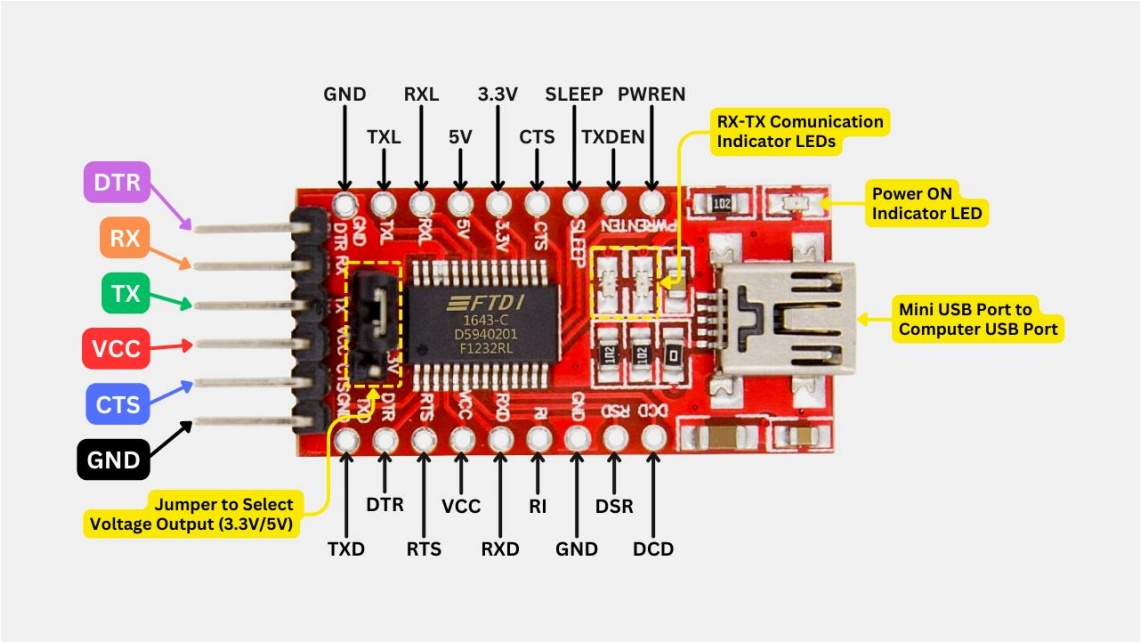
Mạch chuyển đổi USB sang UART TTL sử dụng IC FT232RL là một loại mạch giao tiếp phổ biến, giúp kết nối các thiết bị sử dụng giao tiếp UART (TTL) như vi điều khiển, module GPS, module Bluetooth, v.v., với máy tính qua cổng USB. IC FT232RL được phát triển bởi FTDI (Future Technology Devices International), và là một trong những IC phổ biến nhất cho việc chuyển đổi tín hiệu USB sang TTL UART.

Thông số kỹ thuật của Mạch chuyển USB UART TTL FT232RL

- Loại đầu nối: Cổng USB-B mini
- IC mô-đun: chip FTDI FT232RL
- Điện áp hoạt động: DC 5V/3.3V
- Dòng điện rút tối đa: 500mA @5V; 50mA @3.3V
- Các chỉ số truyền thông: RXD và TXD
- Tốc độ truyền dữ liệu: 183 Baud đến 3 Mbaud (RS232, RS422, RS485) ở mức TTL
- Bộ đệm nhận: 128 Byte
- Bộ đệm truyền: 256 Byte
- Kích thước bảng (L x W): 36mm x 18mm
- Trọng lượng: 4g

Sơ đồ chân chân ra của Mạch chuyển USB UART TTL FT232RL

- DTR: Data terminal ready control output / Handshake signal. (có thể reset arduino khi nạp chương trình)
- RXD: Receive asynchronous data Input – nhận tín hiệu
- TXD: Transmit asynchronous data output – truyền tín hiệu
- VCC: Chân nguồn cấp, có thể chọn 5V hoặc 3.3VDC thông qua Jumper
- CTS: Clear to send control input / handshake signal (không sử dụng)



Hình 1.9: Sơ đồ các chân của mạch chuyển USB UART TTL FT232RL

Mạch chuyển USB UART TTL FT232RL sử dụng IC FT232RL từ chính hãng FTDI, mạch được thiết kế nhỏ gọn nhưng vẫn ra chân đầy đủ, rất dễ sử dụng với mọi hệ điều hành Windows, Mac, Linux

- Chip có sẵn ổn áp và dao động tích hợp bên trong, hoạt động rất ổn định so với các dòng chip USB to serial khác
- Mạch có thể hoạt động ở 2 chế độ 5v hoặc 3v3, bằng cách thiết lập trên jumper trên mạch
- Chân cắm ra gồm 2 loại theo chuẩn FTDI (phù hợp với Arduino) và chuẩn UART thường, được ký hiệu rõ ràng trên mạch. Đầu vào sử dụng loại USB B mini.
- Ngoài ra, trên mạch có sẵn 2 led cho tín hiệu TX và RX, giúp theo dõi trực tiếp trạng thái tín hiệu.

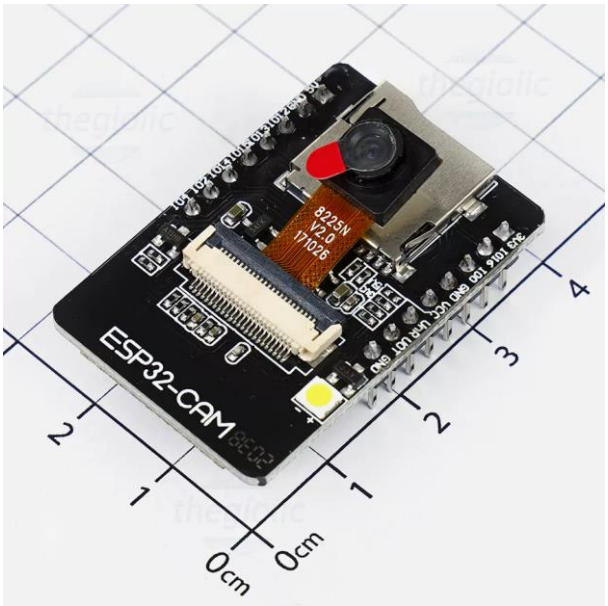
USB to Serial – FTDI có kích thước nhỏ gọn dễ dàng tích hợp vào ứng dụng:

- Làm mạch nạp cho các bản Arduino không tích hợp mạch nạp onboard như: ChipiPRO-LITE, Arduino Pro. Lilypad…
- Làm trung gian giao tiếp bo mạch với máy tính. rất hữu ích khi cần truyền dữ liệu từ bo mạch lên máy tính để kiểm tra, phân tích.
- Làm mạch nạp cho một số dòng vi điều khiển ARM, AVR, 89, PIC,… có hỗ trợ nạp bằng UART.

1.6 Giới thiệu về module ESP32-CAM

1.6.1 ESP32-CAM là gì?

ESP32-CAM là một module camera phát triển dựa trên vi điều khiển ESP32 của Espressif, tích hợp khả năng xử lý hình ảnh và truyền dữ liệu qua Wi-Fi, rất phù hợp cho các ứng dụng IoT (Internet of Things) như giám sát an ninh, nhận diện khuôn mặt, và các dự án camera thông minh. Với kích thước nhỏ gọn và giá thành



thấp, ESP32-CAM đã trở thành một lựa chọn phổ biến trong cộng đồng lập trình viên và các nhà phát triển hệ thống nhúng.

Hình 1.10: Module ESP32-CAM

1.6.3 Thông số kỹ thuật

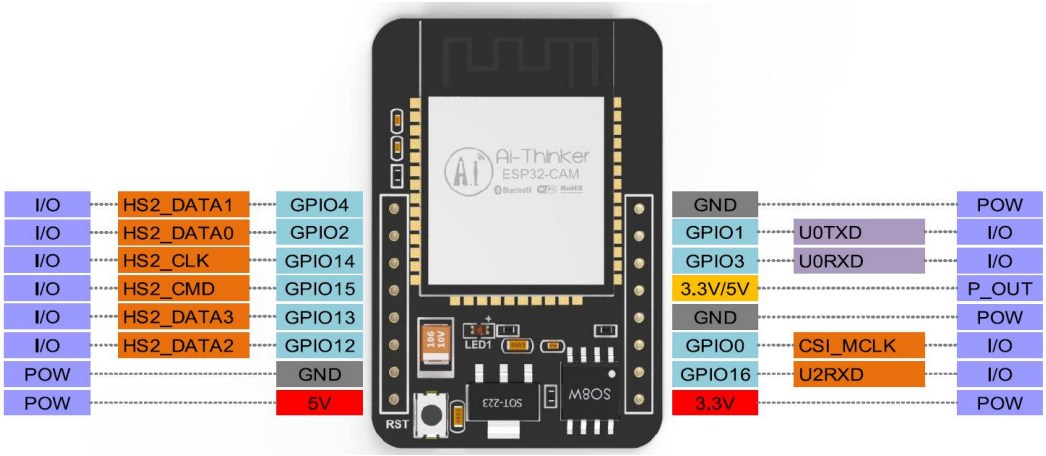
- Module: ESP32-CAM
- Kiểu: Dip-16
- Kích thước: 27 x 40.5 x 4.5 mm
- SPI Flash: Mặc định 32Mbit
- RAM: Internal520KB + PSRAM 4M bên ngoài
- Bluetooth: tiêu chuẩn Bluetooth4.2BR / EDR và BLE
- Wifi: 802.11 b/g/n/e/i
- Giao diện hỗ trợ: UART, SPI, I2C, PWM
- Hỗ trợ thẻ TF: Hỗ trợ tối đa 4G
- Cổng IO: 9
- Tốc độ cổng giao tiếp: 115200 bps mặc định
- Định dạng đầu ra hình ảnh: JPEG (chỉ được hỗ trợ bởi OV2640), BMP
- Phạm vi phổ: 2412 ~ 2484 MHz
- Dạng ăng ten: ăng-ten PCB trên bo mạch, đạt được 2dBi
- Truyền tín hiệu:
    - 802.11b: 17 ± 2 dBm (@ 11Mbps)
    - 802.11g: 14 ± 2 dBm (@ 54Mbps)
    - 802.11n: 13 ± 2 dBm (@ MCS7)
- Độ nhạy
    - CCK, 1 Mb/giây: -90dBm
    - CCK, 11 Mb/giây: -85dBm

    - 6 Mb/giây (1/2 BPSK): -88dBm
    - 54 Mbps (3/4 64-QAM): -70dBm
    - MCS7 (65 Mb/giây, 72,2 Mb/giây): -67dBm
- Sự tiêu thụ năng lượng
    - Tắt đèn flash: 180mA @ 5V
    - Bật đèn flash và điều chỉnh độ sáng tối đa: 310mA @ 5V
    - Sleep Deep: Mức tiêu thụ điện năng thấp nhất có thể đạt 6mA @ 5V
    - Modem-bed: lên tới 20mA @ 5V
    - Sleep Light: lên tới 6,7mA@5V
- Bảo vệ: WPA / WPA2 / WPA2-Enterprise / WPS
- Phạm vi cung cấp điện: 5V
- Nhiệt độ hoạt động: -20 °C ~ 85 °C
- Môi trường lưu trữ: -40 °C ~ 90 °C, <90% rh

1.6.3 Sơ đồ chân ESP32-CAM

- Có ba chân GND và ba chân cấp nguồn: 3,3V, 5V và 3,3V hoặc 5V.
- GPIO 1 (TX) và GPIO 3 (RX) là các chân giao tiếp serial UART mặc định. Chúng được sử dụng để truyền và nhận dữ liệu qua giao tiếp serial. Khi nạp mã cho ESP32 qua USB hoặc bộ chuyển USB-to-Serial, dữ liệu được truyền và nhận qua hai chân này.
- GPIO 0 đóng một vai trò quan trọng vì nó quyết định xem ESP32 có ở chế độ nạp mã(Flashing mode) hay không. Khi GPIO 0 được kết nối với GND, ESP32 ở chế độ Flash, cho phép nạp mã từ máy tính. Sau khi nạp xong, ngắt kết nối GPIO 0 khỏi GND để ESP32 khởi động lại và chạy mã đã nạp.

- Các chân sau được kết nối nội bộ với đầu đọc thẻ nhớ microSD qua giao thức SPI (Serial Peripheral Interface) hoặc SDIO (Secure Digital Input Output):
    - GPIO 14: chân **Clock (CLK)** trong giao thức giao tiếp SPI/SDIO với thẻ SD. Tín hiệu xung nhịp (clock) được gửi qua chân này để đồng bộ hóa việc truyền và nhận dữ liệu giữa ESP32 và thẻ SD.
    - GPIO 15: Chân **Command (CMD)** được dùng để truyền lệnh và điều khiển đến thẻ SD.
    - GPIO 2: Data 0, đường dữ liệu chính để truyền tải các bit dữ liệu từ ESP32 đến thẻ SD hoặc ngược lại.
    - GPIO 4: Data 1 (cũng được kết nối với đèn LED trên bo mạch)
    - GPIO 12: Data 2
    - GPIO 13: Data 13



Hình 1.11: Sơ đồ chân kết nối của ESP32-CAM

1.7. Thuật toán Haar Cascade Classifier

1.7.1 Giới thiệu cơ bản

Haar Cascade Classifier được sử dụng để phát hiện khuôn mặt từ hình ảnh hoặc video trong thời gian thực. Thuật toán này được phát triển bởi **Paul Viola** và **Michael Jones** vào năm 2001 trong công trình của họ có tên là "Rapid Object Detection using a Boosted Cascade of Simple Features".

Haar Cascade Classifier là một kỹ thuật phát hiện đối tượng mạnh mẽ, đặc biệt phổ biến trong các ứng dụng nhận diện khuôn mặt. Nó sử dụng một chuỗi các bộ phân loại đơn giản, mỗi bộ dựa trên một tập hợp các tính năng đơn giản (Haar-like features), nhằm phát hiện các đặc điểm khác nhau của khuôn mặt (như mắt, mũi, miệng, hoặc toàn bộ khuôn mặt).

Tập tin `haarcascade_frontalface_default.xml`: Đây là một file XML chứa dữ liệu huấn luyện từ hàng ngàn mẫu khuôn mặt, được tạo ra bằng cách sử dụng thuật toán Haar Cascade. Mô hình này đã được huấn luyện đặc biệt để phát hiện các khuôn mặt nhìn trực diện (frontal face).

Được sử dụng rộng rãi trong các ứng dụng phát hiện khuôn mặt như camera giám sát, ứng dụng di động, hệ thống nhận diện khuôn mặt, và các ứng dụng trong thị giác máy tính.

1.7.2 Các bước của thuật toán

***B1: Haar-like Features**:*

Haar-like features là các khối đơn giản mô tả sự khác biệt về độ sáng giữa các vùng khác nhau trong một hình ảnh. Ví dụ: một feature đơn giản có thể phát hiện sự khác biệt giữa vùng sáng của trán và vùng tối của mắt.

Các tính năng này được lấy mẫu trên toàn bộ ảnh đầu vào ở nhiều tỷ lệ và vị trí khác nhau, nhằm phát hiện đối tượng trong nhiều điều kiện khác nhau.

***B2: Integral Image (Hình ảnh tích phân)**:*

Để tính toán nhanh các Haar-like features, một kỹ thuật gọi là "hình ảnh tích phân" được sử dụng. Integral image cho phép tính tổng các giá trị pixel của bất kỳ vùng hình chữ nhật nào trong ảnh chỉ với vài phép tính đơn giản, giúp tối ưu hóa tốc độ xử lý.

***B3: Training with AdaBoost (Huấn luyện với AdaBoost)**:*

Bộ phân loại Haar Cascade được huấn luyện với thuật toán AdaBoost, giúp chọn ra những Haar-like features quan trọng nhất từ hàng nghìn features tiềm năng, từ đó tối ưu hóa việc phân loại và phát hiện khuôn mặt.

***B4: Cascade of Classifiers**:*

Để tăng tốc quá trình phát hiện, thuật toán sử dụng một chuỗi (cascade) các bộ phân loại. Mỗi ảnh con sẽ phải đi qua một chuỗi các bộ phân loại, nếu vượt qua một bộ, nó sẽ được kiểm tra tiếp ở bộ kế tiếp. Điều này giúp loại bỏ những vùng không phải khuôn mặt một cách nhanh chóng, và chỉ kiểm tra kỹ hơn ở những vùng có khả năng chứa khuôn mặt.

1.7.3 Ưu và nhược điểm của thuật toán

Ưu điểm của thuật toán Haar Cascade:

- **Tốc độ nhanh**: Được thiết kế để phát hiện khuôn mặt trong thời gian thực, rất nhanh và hiệu quả khi xử lý các khung hình video liên tục.

- **Nhẹ**: Các bộ phân loại đơn giản nên việc sử dụng ít tài nguyên hệ thống, phù hợp với các ứng dụng nhúng hoặc thiết bị có cấu hình thấp.

Nhược điểm:

- **Độ chính xác thấp hơn với những góc nhìn khác nhau**: Mô hình huấn luyện `haarcascade_frontalface_default.xml` chỉ hoạt động tốt nhất khi khuôn mặt ở góc nhìn trực diện. Nếu khuôn mặt nghiêng, quay ngang, hoặc bị che khuất một phần, khả năng phát hiện sẽ giảm đi.
- **Nhạy cảm với ánh sáng**: Những sự thay đổi về điều kiện ánh sáng có thể gây khó khăn cho thuật toán, dẫn đến kết quả phát hiện không chính xác.
- **Không phù hợp cho các tác vụ nhận diện phức tạp**: Haar Cascade chỉ phù hợp cho phát hiện đối tượng, không phải nhận diện đối tượng cụ thể (như nhận diện danh tính).

1.8 Thuật toán LBPH

1.8.1 Giới thiệu về thuật toán

LBPH (Local Binary Patterns Histogram) là một thuật toán nhận diện khuôn mặt dựa trên đặc trưng cục bộ của hình ảnh. Nó dựa vào cách biểu diễn các mẫu cục bộ trên khuôn mặt bằng cách so sánh mối quan hệ của các pixel với các pixel xung quanh, từ đó tạo ra các **mẫu nhị phân**.

Thuật toán **LBPH** (Local Binary Patterns Histogram) được sử dụng trong các hệ thống nhận diện khuôn mặt. Nó cung cấp một phương pháp đơn giản nhưng hiệu quả để nhận diện khuôn mặt, và là một trong những thuật toán phổ biến nhất trong các hệ thống nhận

diện khuôn mặt với OpenCV. Thuật toán này được hỗ trợ trong OpenCV qua phương thức `LBPHFaceRecognizer_create()`.

Các thông số quan trọng của LBPH:

- **radius**: Bán kính của các pixel xung quanh để so sánh (mặc định là 1).
- **neighbors**: Số pixel xung quanh được xem xét trong LBP (mặc định là 8).
- **grid_x, grid_y**: Số lượng ô chia ảnh theo chiều ngang và chiều dọc (thường là 8x8).

1.8.2 Các bước của thuật toán:

B1: Chia hình ảnh thành các ô nhỏ (Cell):

Ảnh được chia thành nhiều ô con có kích thước nhỏ (thường là 3x3 pixel hoặc 5x5 pixel).

B2: Tính toán Local Binary Patterns (LBP):

Với mỗi pixel trong một ô con, ta so sánh giá trị của nó với các pixel xung quanh theo mẫu 3x3.

Nếu giá trị của pixel trung tâm lớn hơn hoặc bằng giá trị pixel xung quanh, ta ghi là 1, ngược lại là 0.

Kết quả là một dãy nhị phân 8 bit, đại diện cho mẫu cục bộ trong ô con. Mẫu nhị phân này được chuyển thành một số thập phân, gọi là **Local Binary Pattern (LBP)**.

Ví dụ, giả sử ô con 3x3 có các giá trị độ xám như sau:

```
50  55  60
45  48  40
55  60  62
```

So sánh từng giá trị pixel xung quanh với pixel trung tâm (48):

```
1  1  1
0     0
1  1  1
```

Kết quả nhị phân là `11100111`, sau đó được chuyển thành số thập phân là `231`. Số này đại diện cho ô con đó.

***B3: Xây dựng biểu đồ (Histogram) cho từng ô**:*

Sau khi tính toán giá trị LBP cho mỗi ô con, ta tạo ra một biểu đồ tần suất xuất hiện các giá trị LBP trong ô con đó.

***B4: Ghép biểu đồ của tất cả các ô thành một biểu đồ tổng thể**:*

Kết quả là một biểu đồ đại diện cho toàn bộ khuôn mặt. Đây là biểu diễn đặc trưng của khuôn mặt dựa trên phân phối các mẫu cục bộ LBP.

***B5: So sánh biểu đồ (Histogram Matching)**:*

Để nhận diện khuôn mặt, biểu đồ của khuôn mặt cần nhận diện sẽ được so sánh với biểu đồ của các khuôn mặt đã lưu trong cơ sở dữ liệu. Phương pháp thường dùng là tính khoảng cách giữa các biểu đồ, chẳng hạn như **khoảng cách Euclidean** hoặc **khoảng cách chi bình phương**. Khuôn mặt với khoảng cách ngắn nhất được xem là khuôn mặt phù hợp.

1.8.3 Ưu và nhược điểm

Ưu điểm của LBPH:

- **Hiệu quả trong các điều kiện ánh sáng khác nhau**: Thuật toán LBPH có khả năng nhận diện khuôn mặt tốt ngay cả khi điều kiện ánh sáng thay đổi, vì nó dựa trên các đặc trưng cục bộ của ảnh.

- **Dễ hiểu và dễ cài đặt**: LBPH là một thuật toán khá đơn giản, dễ cài đặt và trực quan.
- **Chạy nhanh**: Thuật toán này có thể chạy khá nhanh, phù hợp cho các ứng dụng thời gian thực như hệ thống an ninh hoặc ứng dụng di động.

Nhược điểm của LBPH:

- **Không hiệu quả cho việc nhận diện khuôn mặt với sự thay đổi về góc độ**: LBPH hoạt động tốt nhất với các khuôn mặt được chụp trực diện. Khi khuôn mặt quay nghiêng nhiều hoặc có sự thay đổi lớn về góc độ, thuật toán này có thể không nhận diện tốt.
- **Không mạnh mẽ bằng các mô hình hiện đại hơn**: So với các mô hình nhận diện khuôn mặt hiện đại như các mô hình dựa trên mạng nơ-ron sâu (Deep Learning), LBPH kém chính xác và hiệu quả hơn trong các tình huống phức tạp.

Chương 2. Thiết kế hệ thống

2.1 Mô tả hệ thống

Hệ thống phát hiện xâm nhập này được thiết kế nhằm giám sát và bảo vệ một khu vực cụ thể bằng cách sử dụng công nghệ IoT với module ESP32-CAM và cảm biến chuyển động PIR. Khi có chuyển động, hệ thống sẽ kích hoạt camera, gửi hình ảnh về server để lưu trữ và thông báo cho người dùng. Người dùng có thể điều khiển chế độ phát hiện chuyển động từ xa thông qua giao diện điều khiển trên trình duyệt.

Các thành phần chính của hệ thống:

- **Cảm biến chuyển động PIR**: Cảm biến PIR (Passive Infrared Sensor) phát hiện sự thay đổi trong bức xạ hồng ngoại khi có người hoặc vật thể di chuyển trong khu vực giám sát. Khi phát hiện chuyển động, cảm biến sẽ gửi tín hiệu để kích hoạt camera ESP32-CAM.
- **ESP32-CAM**: Là module chính của hệ thống, ESP32-CAM tích hợp sẵn camera và khả năng kết nối Wi-Fi. Khi nhận tín hiệu từ cảm biến PIR, camera sẽ được kích hoạt để chụp ảnh khu vực giám sát. ESP32-CAM sau đó gửi hình ảnh này về server thông qua Wi-Fi để lưu trữ và xử lý.
- **Server lưu trữ và phân tích hình ảnh**: Server Flask nhận hình ảnh từ ESP32-CAM, lưu trữ và phân loại chúng. Hình ảnh sẽ được lưu vào các thư mục riêng, ví dụ `registered_faces/` cho ảnh chụp từ lệnh của người dùng và `detected/` cho ảnh phát hiện chuyển động. Nếu cần, server có thể gửi cảnh báo đến người dùng.
- **Giao diện điều khiển từ xa**: Người dùng có thể truy cập vào giao diện web do server cung cấp để bật/tắt chế độ phát hiện chuyển động và xem lại hình ảnh từ hệ thống. Giao diện này cũng cho phép đăng ký khuôn mặt bằng cách yêu cầu ESP32-CAM chụp ảnh và gửi về server.

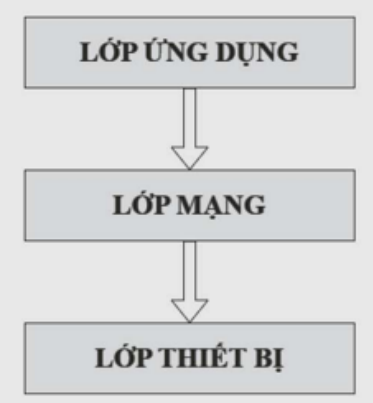
Quy trình hoạt động:

- **Phát hiện chuyển động**: Cảm biến PIR phát hiện chuyển động trong khu vực giám sát và gửi tín hiệu đến ESP32-CAM.
- **Kích hoạt camera**: ESP32-CAM nhận tín hiệu từ cảm biến PIR và chụp ảnh khu vực giám sát.

- **Gửi dữ liệu hình ảnh**: ESP32-CAM gửi ảnh đã chụp về server qua kết nối Wi-Fi.
- **Lưu và phân loại hình ảnh**: Server lưu ảnh vào thư mục phù hợp (như `registered_faces/` hoặc `detected/`).
- **Nhận thông báo**: Nếu chế độ cảnh báo được bật, server có thể gửi thông báo cho người dùng khi phát hiện chuyển động.
- **Kiểm tra và điều khiển từ xa**: Người dùng có thể kiểm tra hình ảnh từ giao diện web và bật/tắt chế độ phát hiện chuyển động nếu cần thiết.

2.2 Sơ đồ khối của hệ thống

2.2.1 Kiến trúc IoT của hệ thống



Hình 2.1: Kiến trúc IoT 3 lớp

Hệ thống được xây dựng dựa trên kiến trúc IoT 3 lớp

a, Lớp thiết bị: Lớp này bao gồm tất cả các thiết bị vật lý có khả năng thu thập dữ liệu và thực hiện các hành động dựa trên dữ liệu đó. Trong hệ thống này, lớp thiết bị bao gồm:

- **ESP32-CAM**: Là module chính của hệ thống, tích hợp camera và khả năng kết nối Wi-Fi. ESP32-CAM thực hiện nhiệm vụ chụp ảnh khi có tín hiệu từ cảm biến PIR và gửi hình ảnh về server.

- **Cảm biến PIR**: Phát hiện chuyển động bằng cách cảm nhận sự thay đổi trong bức xạ hồng ngoại. Khi phát hiện chuyển động, cảm biến này sẽ kích hoạt ESP32-CAM để chụp ảnh.

b, Lớp mạng: Lớp này chịu trách nhiệm cho việc truyền tải dữ liệu giữa các thiết bị và server. Trong hệ thống này:

- **Kết nối Wi-Fi**: ESP32-CAM sử dụng kết nối Wi-Fi để gửi hình ảnh đến server. Điều này cho phép truyền tải dữ liệu hình ảnh nhanh chóng và hiệu quả.
- **Server**: Máy chủ Flask nhận hình ảnh từ ESP32-CAM và lưu trữ chúng. Server cũng có thể xử lý các yêu cầu từ người dùng qua giao diện web.

c, Lớp ứng dụng: Lớp này bao gồm các ứng dụng và giao diện mà người dùng tương tác trực tiếp. Trong hệ thống này:

- **Giao diện điều khiển web**: Người dùng có thể truy cập vào giao diện web để điều khiển chế độ phát hiện chuyển động, xem lại hình ảnh đã chụp và quản lý các thông báo. Giao diện này giúp người dùng dễ dàng tương tác với hệ thống và nhận thông tin cần thiết.
- **Chức năng thông báo**: Server có thể gửi thông báo cho người dùng nếu phát hiện có chuyển động, cung cấp khả năng theo dõi và phản ứng kịp thời với các sự kiện xâm nhập.

2.2.2 Sơ đồ khối của hệ thống

- Khối nguồn:

- Cấp nguồn cho toàn mạch, sử dụng nguồn 5VDC cấp cho khối xử lý trung tâm, cảm biến và nguồn 220VAC cho các thiết bị điện.

- Khối cảm biến (PIR):

  - Kết nối với khối vi xử lý trung tâm (ESP32-CAM) qua giao tiếp Digital I/O.
  - Khi phát hiện chuyển động, PIR sẽ gửi tín hiệu số (HIGH/LOW) tới ESP32-CAM để xử lý.

- Khối vi xử lý trung tâm (ESP32-CAM):

  - Nhận tín hiệu số từ PIR thông qua giao tiếp Digital I/O để xử lý các sự kiện liên quan đến phát hiện chuyển động.
  - Kết nối với thiết bị thực hiện (buzzer) cũng qua giao tiếp Digital I/O để điều khiển bật/tắt buzzer khi có sự kiện cần thông báo.
  - Giao tiếp với server qua giao thức HTTP, gửi dữ liệu từ ESP32-CAM lên server (có thể bao gồm hình ảnh chụp từ camera hoặc thông tin khác).

- Khối thiết bị thực hiện (buzzer):

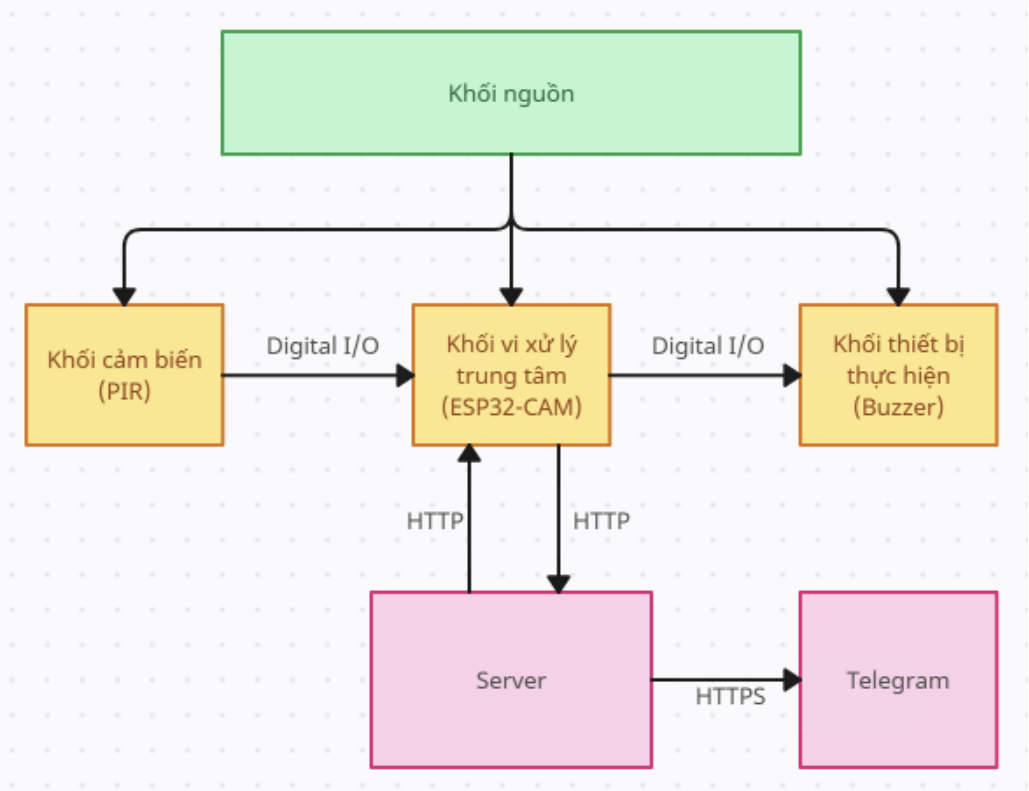
  - Nhận tín hiệu điều khiển từ ESP32-CAM qua Digital I/O. Buzzer sẽ được kích hoạt khi có tín hiệu từ ESP32-CAM (ví dụ khi phát hiện chuyển động hoặc có sự kiện cần cảnh báo).

- Server:

  - Nhận thông tin từ ESP32-CAM qua giao thức HTTP và có thể xử lý, lưu trữ hoặc phản hồi lại dữ liệu.
  - Gửi thông báo đến Telegram thông qua giao thức HTTPS để cảnh báo cho người dùng khi cần.

- Telegram:

- Nhận thông báo qua giao thức HTTPS từ server. Điều này có thể bao gồm việc gửi tin nhắn cảnh báo hoặc hình ảnh chụp từ camera cho người dùng trên Telegram.



Hình 2.2: Sơ đồ khối của hệ thống

2.3 Thiết kế phần cứng

Thiết bị	**Chân thiết bị**	**Kết nối với chân của ESP32-CAM**	**Chức năng**
PIR	VCC	5V	Cấp nguồn 5V từ ESP32-CAM
	GND	GND	Kết nối đất
	OUT	GPIO 14	Tín hiệu đầu ra của PIR kết nối đến GPIO
Buzzer	GND	GND	Kết nối đất

	Signal	GPIO 4	Điều khiển bật/tắt từ GPIO 4
FTDI	VCC (3.3V/5V)	3.3V	Cấp nguồn từ FTDI Adapter tới ESP32-CAM
	GND	GND	Kết nối đất
	TX	U0R	Chân dữ liệu ra của FTDI kết nối chân nhận dữ liệu của ESP32-CAM
	RX	U0T	Chân nhận dữ liệu của FTDI kết nối chân dữ liệu ra của ESP32-CAM
ESP32-CAM	3.3V	FTDI VCC	Nhận cấp nguồn từ FTDI
	5V	PIR VCC	Cấp nguồn cho PIR
	GND	FTDI GND / PIR GND / Buzzer GND	Kết nối đất
	GPIO 4	Buzzer signal	Điều khiển tín hiệu đến Buzzer
	GPIO 14	PIR OUT	Nhận tín hiệu từ cảm biến PIR

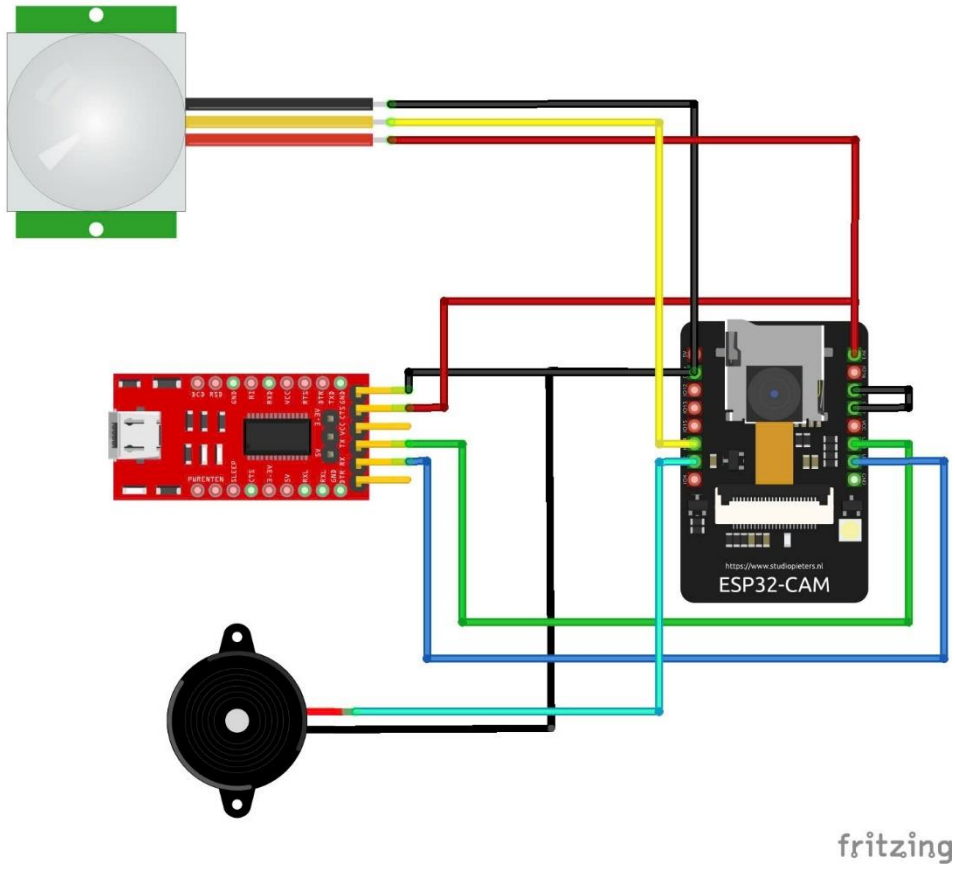
Bảng 2.3: Bảng chân kết nối các thiết bị trong hệ thống

Các bước thực hiện lắp ráp và ghép nối các mạch và module

- Bước 1: Kết nối module ESP32 với các module thiết bị
- Bước 2: Kết nối module cảm biến với module ESP32
- Bước 3: Cấp nguồn cho mạch
- Bước 4: Đo kiểm tra từng chân của các thiết bị đã kết nối hết chưa

• Bước 5: Gán các mạch vào mô hình nhà

• Bước 6: Cuối cùng nạp chương trình và test lại chương trình có đạt như yêu cầu ban đầu không.



Hình 2.4: Sơ đồ kết nối của hệ thống

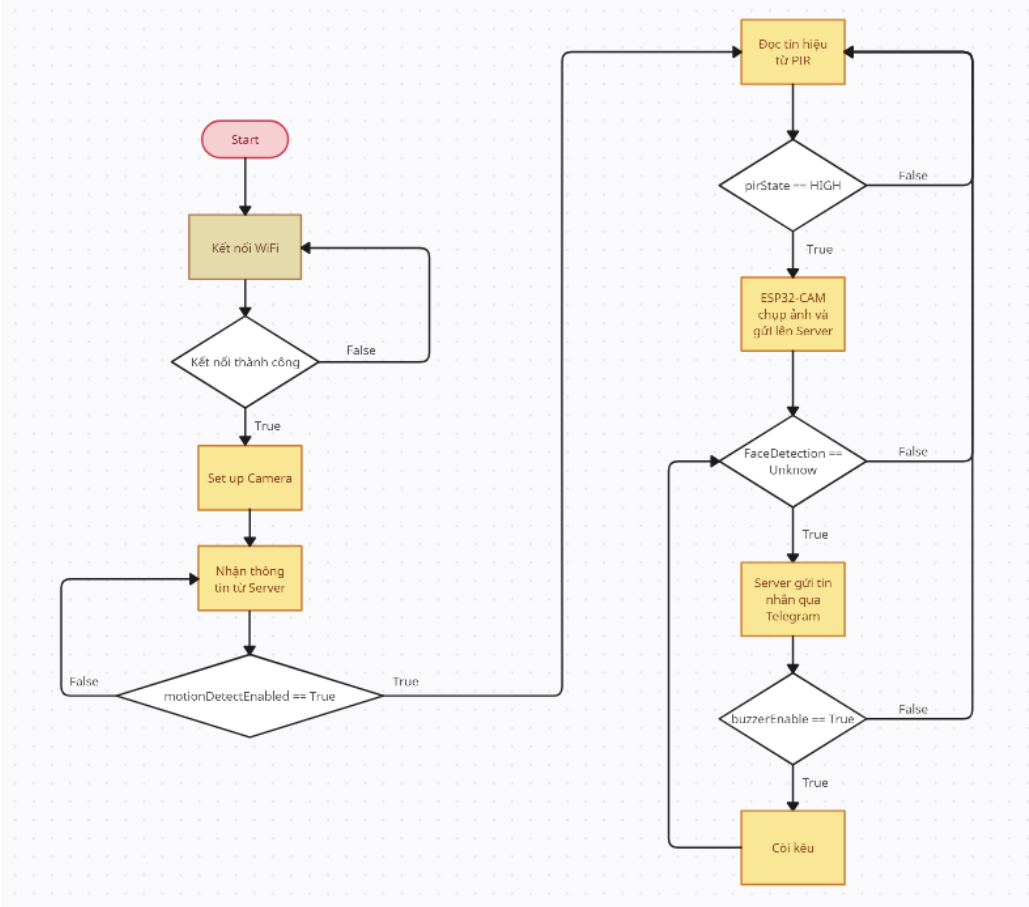
2.4 Thiết kế phần mềm

2.4.1. *Lưu đồ giải thuật*

Khi bắt đầu quá trình hoạt động thì sẽ thực hiện việc khởi tạo hệ thống. Kiểm tra hệ thống có được thiết lập hay chưa.

Hệ thống sẽ thực hiện việc kiểm tra xem có nhận được tín hiệu điều khiển hay chưa. Nếu có nhận được tín hiệu thì bắt đầu quá trình xử lý và đưa ra để điều khiển thiết bị được kết nối.

ESP32 sẽ tiến hành kết nối Internet (Wifi), và thiết lập kết nối với Server. Đợi khi kết nối thành công. Nếu có trao đổi dữ liệu với Server (người dùng tác động vào giao diện web hoặc có tín hiệu từ Server hoặc nút nhấn gửi xuống), thì thiết bị sẽ được điều khiển theo yêu cầu người dùng.



Hình 2.5: Lưu đồ giải thuật của hệ thống

2.4.2 Code chương trình cho ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <HTTPClient.h>

// Thông tin mạng WiFi
const char* ssid = "Fui";
const char* password = "mermermer";
```

```
const char* serverUrl = "http://192.168.7.245:5000/upload"; // Địa
chỉ server

#define PIR_PIN 14
#define BUZZER_PIN 4
bool buzzerEnabled = false;
bool motionDetected = false;
bool motionDetectEnabled = false;

// Camera setup
void setup_camera() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = 5;
  config.pin_d1 = 18;
  config.pin_d2 = 19;
  config.pin_d3 = 21;
  config.pin_d4 = 36;
  config.pin_d5 = 39;
  config.pin_d6 = 34;
  config.pin_d7 = 35;
  config.pin_xclk = 0;
  config.pin_pclk = 22;
  config.pin_vsync = 25;
  config.pin_href = 23;
  config.pin_sscb_sda = 26;
  config.pin_sscb_scl = 27;
  config.pin_pwdn = 32;
  config.pin_reset = -1;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t *s = esp_camera_sensor_get();
    if (s != NULL) {
```

```
        s->set_vflip(s, 1);
        s->set_hmirror(s, 1);
    }
}

// WiFi setup
void connect_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Serial.println(WiFi.localIP());
}

// Hàm chụp và gửi ảnh lên server
void capture_and_send_photo(const String& mode) {
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    return;
  }

  HTTPClient http;
  String url = String(serverUrl) + "?mode=" + mode;
  http.begin(url);
  http.addHeader("Content-Type", "image/jpeg");

  int httpResponseCode = http.POST(fb->buf, fb->len);
  if (httpResponseCode == 200) {
    Serial.println("Photo uploaded successfully");
  } else {
    Serial.printf("Failed to upload photo, code: %d\n",
httpResponseCode);
  }
  http.end();
  esp_camera_fb_return(fb);
}

// ESP32 web server nhận requests từ server
WiFiServer server(80);

void setup() {
```

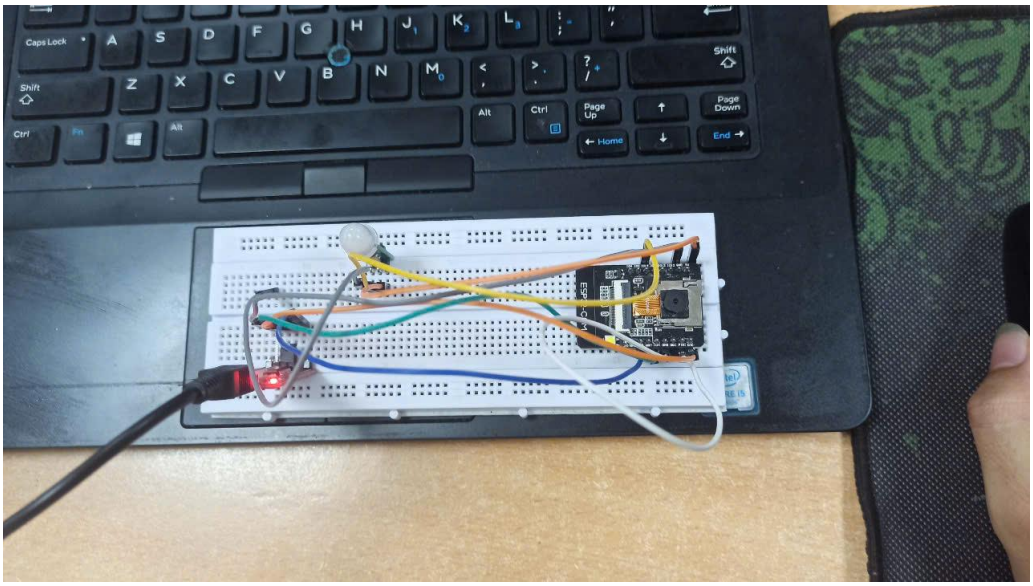
```
  Serial.begin(115200);
  connect_wifi();
  setup_camera();
  server.begin();

  pinMode(PIR_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  WiFiClient clientA = server.available();
  if (clientA) {
    Serial.println("New Client connected");
    String request = clientA.readStringUntil('\r');
    clientA.flush();

    // Nếu request từ server là chụp ảnh
    if (request.indexOf("/capture_photo") != -1) {
      capture_and_send_photo("register");
      clientA.println("HTTP/1.1 200 OK");
      clientA.println("Content-Type: text/html");
      clientA.println();
      clientA.println("Photo captured successfully");
    }

    // Nếu request từ server là bật cảm biến chuyển động
    else if (request.indexOf("/enable_motion") != -1) {
      motionDetectEnabled = true;
      Serial.println("Motion detection enabled");
      clientA.println("HTTP/1.1 200 OK");
      clientA.println("Content-Type: text/html");
      clientA.println();
      clientA.println("Motion detection enabled");
    }

    // Nếu request từ server là tắt cảm biến chuyển động
    else if (request.indexOf("/disable_motion") != -1) {
      motionDetectEnabled = false;
      Serial.println("Motion detection disabled by user");
      clientA.println("HTTP/1.1 200 OK");
      clientA.println("Content-Type: text/html");
      clientA.println();
      clientA.println("Motion detection disabled");
    }
```

```
  // Nếu request từ server là tắt chuông
  else if (request.indexOf("/enable_buzzer") != -1) {
    buzzerEnabled = true;
    Serial.println("Buzzer enabled");
    clientA.println("HTTP/1.1 200 OK");
    clientA.println("Content-Type: text/html");
    clientA.println();
    clientA.println("Buzzer enabled");
  }

  // Nếu request từ server là tắt chuông
  else if (request.indexOf("/disable_buzzer") != -1) {
    buzzerEnabled = false;
    Serial.println("Buzzer disabled");
    clientA.println("HTTP/1.1 200 OK");
    clientA.println("Content-Type: text/html");
    clientA.println();
    clientA.println("Buzzer disabled");
  }

  clientA.stop();
}

// Đọc dữ liệu từ PIR
if (motionDetectEnabled) {
  int pirState = digitalRead(PIR_PIN);

  if (motionDetected && pirState == LOW) {
    motionDetected = false;
    Serial.println("Motion detection ended.");
  }

  if (!motionDetected && pirState == HIGH) {
    motionDetected = true;
    Serial.println("Motion detected! Sending photo...");
    for(int i=0; i<10; ++i){
      Serial.printf("Pic %d\n", i);
      capture_and_send_photo("motion");
      delay(500);
    }

  }
}
```

```
  if (buzzerEnabled) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(1000);
    digitalWrite(BUZZER_PIN, LOW);
  }
  delay(500);
}
```

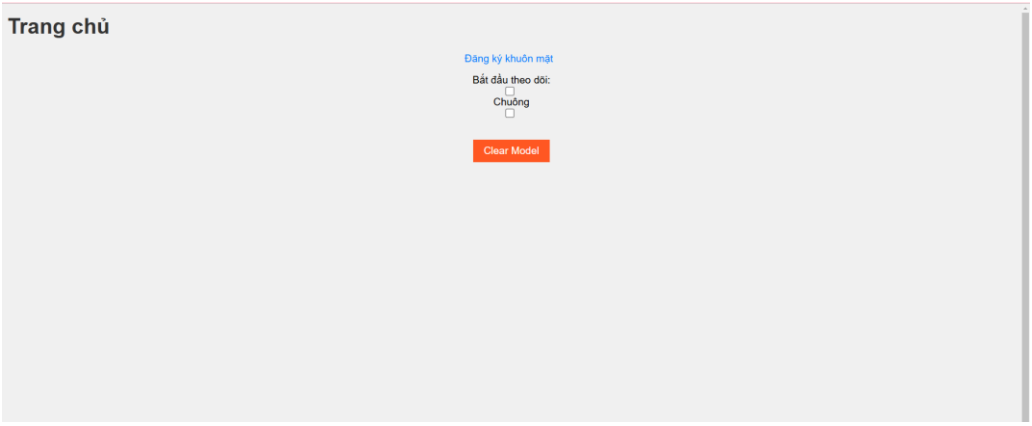
Chương 3: Kết quả và kết luận

3.1. Kết quả mô phỏng

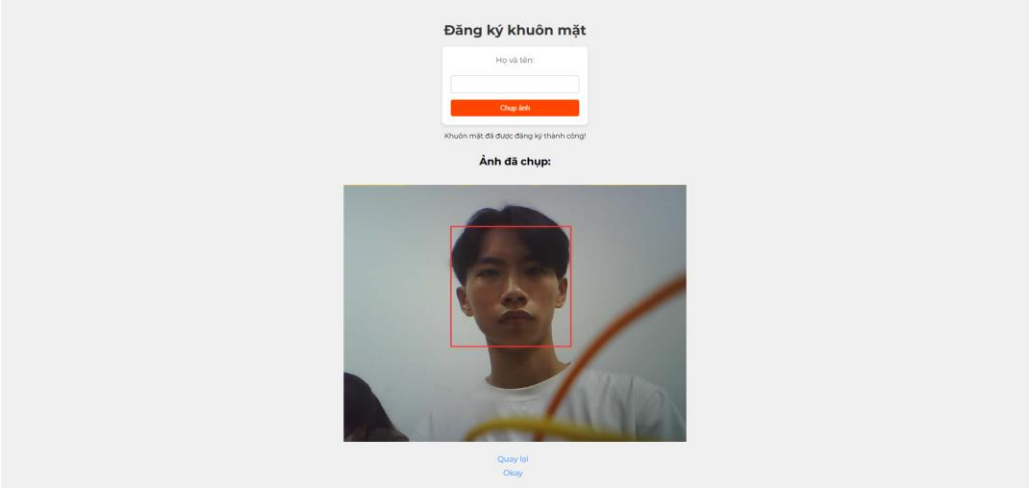


Hình 3.1: Mô phỏng hệ thống thực tế

Giao diện web người dùng:



Giao diện đăng ký khuôn mặt:



Tin nhắn telegram khi phát hiện xâm nhập:

3.2 Nhận xét kết quả

Hành động	Số lần	Thành công
Phát hiện chuyển động từ cảm biến PIR	20	20
Chụp ảnh và gửi ảnh lên server	20	18
Gửi thông báo qua Telegram bot	20	12
Phát hiện khuôn mặt	20	10
Bật còi báo động khi phát hiện xâm nhập	20	10

Bảng 3.2: Bảng đánh giá thử nghiệm hệ thống

Hệ thống giám sát và nhận diện khuôn mặt được xây dựng với ESP32-CAM, cảm biến PIR, và server Flask đã hoạt động đúng như mong đợi. Kết quả của quá trình triển khai và thử nghiệm được mô tả chi tiết dưới đây:

1. Kết nối thành công giữa ESP32-CAM và server

ESP32-CAM kết nối thành công với mạng WiFi, sau đó thiết lập liên lạc với server thông qua giao thức HTTP. Khi một yêu cầu gửi ảnh được kích hoạt, ESP32 chụp ảnh và tải lên server qua giao thức HTTP POST. Server nhận dữ liệu ảnh, lưu trữ và xử lý cho các mục đích khác nhau như nhận diện khuôn mặt hoặc lưu trữ sự kiện.

2. Chụp và gửi ảnh theo yêu cầu

Khi hệ thống nhận được yêu cầu từ server hoặc từ cảm biến PIR phát hiện chuyển động, ESP32-CAM sẽ kích hoạt camera để chụp ảnh. Ảnh được mã hóa dưới định dạng JPEG và gửi lên server. Qua thử nghiệm:

Hệ thống đã gửi thành công hình ảnh từ ESP32 lên server sau khi cảm biến PIR phát hiện chuyển động hoặc khi người dùng yêu cầu chụp ảnh từ giao diện web.

Thời gian trung bình từ khi phát hiện chuyển động đến khi hình ảnh được server xử lý là khoảng 1-2 giây, đảm bảo tính thời gian thực trong các tình huống giám sát.

3. Phát hiện chuyển động và kích hoạt hệ thống cảnh báo

Cảm biến PIR hoạt động hiệu quả trong việc phát hiện chuyển động. Khi có chuyển động trong vùng giám sát, cảm biến sẽ gửi tín hiệu đến ESP32, từ đó kích hoạt quá trình chụp ảnh và gửi thông tin đến server. Server xử lý thông tin và có thể gửi cảnh báo qua các kênh như Telegram bot. Qua thử nghiệm, hệ thống:

- Nhận diện tương đối chính xác các sự kiện chuyển động trong khoảng cách từ 2-3m.
- Phản hồi nhanh chóng với độ trễ tối thiểu khi phát hiện chuyển động.

4. Giao tiếp với Telegram Bot

Server được tích hợp với Telegram bot để gửi thông báo cho người dùng khi có sự kiện chuyển động hoặc khi có người truy cập vào hệ thống. Mỗi khi hệ thống phát hiện chuyển động, hình ảnh chụp được gửi lên server, và thông báo kèm theo hình ảnh sẽ được gửi tới Telegram bot của người dùng. Qua thử nghiệm:

Hệ thống gửi thông báo Telegram nhanh chóng sau khi nhận diện chuyển động, với thời gian trễ trung bình khoảng 1-3 giây từ lúc phát hiện chuyển động đến khi nhận được thông báo trên Telegram.

5. Hiển thị hình ảnh trên giao diện web

Server Flask được cấu hình để xử lý ảnh nhận từ ESP32-CAM và hiển thị trên giao diện web. Kết quả thử nghiệm cho thấy:

Hình ảnh được hiển thị rõ ràng, độ phân giải phù hợp với mục đích giám sát (SVGA - 800x600).

Giao diện web hoạt động ổn định, có thể cập nhật hình ảnh mới mỗi khi có sự kiện chụp ảnh từ ESP32-CAM.

3.4. Phương hướng tương lai

Để hệ thống giám sát và nhận diện khuôn mặt sử dụng ESP32-CAM hoạt động hiệu quả hơn và đáp ứng tốt hơn các nhu cầu thực tế, các phương hướng nâng cấp cả về phần cứng lẫn phần mềm có thể được xem xét như sau:

*1. **Nâng cấp thuật toán xử lý ảnh và nhận diện khuôn mặt***

Hệ thống hiện tại có thể chụp và gửi ảnh từ ESP32-CAM lên server, nhưng để tăng cường khả năng nhận diện và giám sát, cần nâng cấp các thuật toán xử lý ảnh:

Tích hợp nhận diện khuôn mặt: Thay vì chỉ chụp và gửi ảnh, hệ thống có thể tích hợp thêm các thuật toán nhận diện khuôn mặt (face recognition) để tự động nhận diện danh tính của người trong ảnh. Điều này có thể thực hiện bằng cách:

Tích hợp các thư viện như OpenCV, DeepFace, hoặc các mô hình học sâu như MTCNN và FaceNet.

Server có thể xử lý và lưu trữ dữ liệu khuôn mặt của các đối tượng đã được đăng ký để đối chiếu trong các lần phát hiện sau.

Cải thiện thuật toán phát hiện chuyển động: Cảm biến PIR hiện tại có thể phát hiện chuyển động nhưng đôi khi gặp phải các hạn chế như phát hiện chuyển động từ vật thể không mong muốn. Việc nâng cấp các thuật toán phát hiện chuyển động sử dụng xử lý

ảnh hoặc phân tích video có thể giúp hệ thống phát hiện chính xác hơn, tránh báo động giả.

Nén và tối ưu hóa ảnh: Cải thiện thuật toán nén ảnh để gửi dữ liệu nhanh hơn và giảm tải băng thông mạng. Sử dụng các phương pháp nén ảnh hiện đại (như WebP) hoặc giảm chất lượng ảnh mà không làm mất thông tin quan trọng.

*2. **Nâng cấp phần cứng***

Mặc dù ESP32-CAM là một module hiệu quả và chi phí thấp, việc nâng cấp phần cứng có thể mang lại hiệu suất cao hơn và tăng tính năng cho hệ thống:

Camera có độ phân giải cao hơn: ESP32-CAM hiện tại chỉ hỗ trợ các độ phân giải trung bình như SVGA. Việc nâng cấp camera có độ phân giải cao hơn sẽ giúp hệ thống chụp ảnh rõ nét hơn, cải thiện chất lượng nhận diện khuôn mặt và giám sát chi tiết hơn.

Sử dụng các board vi điều khiển mạnh hơn: Các board như ESP32-S3, Raspberry Pi Zero hoặc Raspberry Pi 4 có thể được xem xét để thay thế ESP32-CAM. Các bộ xử lý mạnh hơn này sẽ giúp hệ thống xử lý dữ liệu nhanh hơn, đồng thời có khả năng tích hợp các mô hình nhận diện khuôn mặt trực tiếp mà không cần phụ thuộc vào server quá nhiều.

Tích hợp thêm các cảm biến: Bên cạnh cảm biến PIR, có thể tích hợp thêm các cảm biến khác như cảm biến âm thanh, cảm biến nhiệt để hệ thống có thể phát hiện ra các tình huống cụ thể hơn (như phát hiện tiếng ồn hoặc phát hiện nhiệt độ cao có thể là dấu hiệu của cháy nổ).

*3. **Nâng cấp bảo mật và truyền thông***

Chuyển sang giao thức HTTPS: Hiện tại hệ thống sử dụng HTTP để truyền tải dữ liệu, điều này không đảm bảo an toàn cho các dữ liệu nhạy cảm như hình ảnh khuôn mặt. Nâng cấp lên HTTPS với SSL/TLS sẽ giúp bảo mật dữ liệu trong quá trình truyền tải giữa ESP32 và server, ngăn chặn việc bị nghe lén hoặc tấn công mạng.

Mã hóa dữ liệu: Tăng cường bảo mật bằng cách mã hóa hình ảnh và dữ liệu trước khi gửi lên server, đảm bảo dữ liệu không bị truy cập trái phép trong quá trình truyền tải.

Sử dụng MQTT hoặc WebSocket: MQTT hoặc WebSocket có thể thay thế HTTP cho các ứng dụng thời gian thực, giúp giảm độ trễ và tối ưu hóa việc truyền tải dữ liệu trong các ứng dụng IoT, đặc biệt khi số lượng thiết bị tăng lên.

*4. **Mở rộng tính năng của hệ thống***

Tích hợp hệ thống quản lý thông minh: Hệ thống có thể được nâng cấp để trở thành một phần của hệ thống giám sát thông minh hơn, bao gồm cả nhận diện đối tượng và hành vi (ví dụ: phát hiện hành vi bất thường như xâm nhập, phá hoại, hoặc có hành động nguy hiểm).

Tích hợp lưu trữ đám mây: Hiện tại hệ thống chỉ xử lý và lưu trữ ảnh tại server cục bộ. Việc tích hợp các giải pháp lưu trữ đám mây như Google Cloud, AWS S3 hoặc Firebase có thể giúp mở rộng khả năng lưu trữ không giới hạn và dễ dàng truy cập từ xa, đồng thời nâng cao khả năng quản lý dữ liệu.

Hệ thống cảnh báo đa phương tiện: Ngoài cảnh báo qua Telegram, hệ thống có thể mở rộng để hỗ trợ các phương tiện cảnh báo khác như email, tin nhắn SMS, hoặc thậm chí tích hợp hệ thống

cảnh báo vật lý như còi báo động hoặc đèn nháy khi phát hiện đối tượng xâm nhập.

*5. **Cải thiện trải nghiệm người dùng***

Nâng cấp giao diện web: Cải thiện giao diện người dùng trên web để quản lý và giám sát hệ thống dễ dàng hơn. Ví dụ, cung cấp giao diện theo thời gian thực (real-time) để theo dõi các sự kiện, lịch sử giám sát, và tùy chỉnh các thiết lập của hệ thống (chẳng hạn như vùng giám sát, tần suất chụp ảnh).

Ứng dụng di động: Phát triển ứng dụng di động (Android, iOS) để người dùng có thể dễ dàng quản lý và theo dõi hệ thống giám sát từ xa. Điều này cũng giúp gửi cảnh báo đến điện thoại di động trực tiếp thay vì qua các ứng dụng bên thứ ba như Telegram.

Tài liệu tham khảo

[1.] ESP32-CAM CameraWebServer Example https://github.com/espressif/arduino-esp32/tree/master/libraries/ESP32/examples/Camera/CameraWebServer

[2.] ESP32-CAM datasheet, https://www.espressif.com/sites/default/files/documentation/esp32-cam_datasheet_en.pdf

[3.] PIR HC-SR505 datasheet, https://components101.com/sites/default/files/component_datasheet/HC-SR501%20Datasheet.pdf

[4.] Lê Hải Thanh , Đoàn Vân Chi , Nguyễn Hữu Phát , Nguyễn Trọng Các. (2024). Nhận diện khuôn mặt với OPENCV và thuật toán LBPH. Tạp chí Nghiên cứu khoa học, Trường Đại học Sao Đỏ, Số 1 (84) 2024