



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ CÔNG NGHỆ IOT HIỆN ĐẠI

Đề tài: Smart Home Security

Lớp: NT533.O21

GVHD: Nguyễn Khánh Thuật

Nhóm sinh viên thực hiện:

- | | |
|---------------------|----------|
| 1. Phan Quốc Huy | 21520935 |
| 2. Nguyễn Thanh Duy | 21520780 |
| 3. Lê Hoàng Tú | 21521608 |

MỤC LỤC

MỤC LỤC	2
LỜI NÓI ĐẦU	3
CHƯƠNG 1: MỞ ĐẦU	4
1.1. Lý do chọn đề tài	4
1.2. Mục tiêu cho đề tài	4
1.3. Đối tượng và phạm vi nghiên cứu	4
1.3.1. Đối tượng nghiên cứu	4
1.3.2. Phạm vi nghiên cứu	4
CHƯƠNG 2: TỔNG QUAN	5
2.1. Các vấn đề tồn tại	5
2.2. Vấn đề nhóm tập trung giải quyết	5
CHƯƠNG 3. NGHIÊN CỨU LÝ THUYẾT	6
3.1. Cơ sở lý thuyết	6
3.2. Thiết kế mô hình	8
CHƯƠNG 4. TRIỂN KHAI MÔ HÌNH	9
4.1. Thiết lập xác thực tài khoản mật khẩu trên MQTT Broker	9
4.2. Chương trình Client_pub.py	10
4.3. Chương trình Client_sub.py	14
CHƯƠNG 5. ĐÁNH GIÁ KẾT QUẢ	16
5.1. Kết quả thu được	16
5.2. Hình ảnh minh họa	16
5.3. Đánh giá	20
CHƯƠNG 6. KẾT LUẬN	21
6.1. Tổng kết	21
6.2. Những khó khăn trong quá trình làm đề tài	21
CHƯƠNG 7. ĐỊNH HƯỚNG PHÁT TRIỂN	21

LỜI NÓI ĐẦU

Lời đầu tiên, chúng em xin phép gửi lời cảm ơn tới thầy *Nguyễn Khánh Thuật* đã tận tình giảng dạy và chỉ bảo chúng em trong quá trình học tập cũng như thực hiện đồ án *Công nghệ Internet of Things hiện đại*. Với sự tâm huyết của mình, không chỉ là kiến thức trong sách vở, thầy còn truyền đạt cho chúng em những kiến thức thực tế, kỹ năng mềm khác.

Chúng em đã nỗ lực và cố gắng hoàn thành đề tài được giao một cách tốt nhất trong khả năng của bản thân, nhưng vì vẫn còn thiếu sót về mặt kiến thức nên không thể tránh được những sai sót cũng như những hạn chế trong quá trình hoàn thành đồ án. Dù vậy chúng em vẫn cố gắng và hoàn thiện từng ngày trong quá trình hoàn thành đồ án vừa qua, vậy nên em mong Thầy thông cảm với những thiếu sót của chúng em và góp ý cũng như chỉ bảo để giúp chúng em hoàn thiện hơn.

Một lần nữa, nhóm chúng em xin chân thành cảm ơn và xin được gửi lời chúc đến thầy *Nguyễn Khánh Thuật*. Kính chúc thầy luôn tiến tới, thành công trong sự nghiệp theo đuổi tri thức của thầy và mong rằng thầy sẽ luôn giữ được sự tâm huyết của mình đối với sinh viên.

CHƯƠNG 1: MỞ ĐẦU

1.1. Lý do chọn đề tài

Hiện nay, việc lắp đặt camera ở các hộ gia đình đã trở thành một xu hướng phổ biến trong thời gian gần đây. Sự phát triển của công nghệ đã tạo ra những thiết bị giám sát tiên tiến, nhỏ gọn và dễ sử dụng, đáp ứng nhu cầu bảo vệ an ninh và tăng cường sự an toàn cho cư dân.

Đối với các chung cư, ngoài vấn đề an ninh, việc lắp đặt camera cũng góp phần tăng cường quản lý và giám sát trong chung cư. Các vấn đề liên quan đến an ninh, trật tự và an toàn trong khu dân cư có thể được giải quyết một cách hiệu quả hơn thông qua việc sử dụng camera giám sát.

Có thể thấy, việc lắp đặt camera trong các hộ gia đình đã trở nên phổ biến hơn bao giờ hết. Do đó, các công nghệ giám sát ngày nay cũng cần được phát triển. Không còn chỉ là những chiếc “máy quay phim” như ngày trước nữa mà giờ đây với sự phát triển của công nghệ IOT, cũng như là sức mạnh về phần cứng bên trong những chiếc camera sẽ mở ra những giải pháp thông minh cải thiện cuộc sống của con người

1.2. Mục tiêu cho đề tài

Với sự phát triển không ngừng của công nghệ và phần cứng, cùng sự bùng nổ của trí tuệ nhân tạo đã giúp cho việc tích hợp các mô hình AI vào các thiết bị IOT trở nên thuận tiện và dễ dàng hơn.

Mục tiêu đề tài nhóm chúng em sẽ là tích hợp thuật toán deep learning giúp nhận diện con người trong việc xử lý hình ảnh nhận được từ các camera an ninh. Từ đó, có thể đưa ra những cảnh báo cho người dùng, đưa ra những con số thống kê.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

a) Camera an ninh

Trong đề tài này, nhóm chúng em sẽ tạm thời sử dụng camera laptop để đóng vai trò như các camera an ninh với độ phân giải 1080p và 60fps

b) Server biên

Server biên được sử dụng trong đề tài này vừa đóng vai trò nhận diện con người và xử lý ảnh từ video nhận được từ camera. Vừa đóng vai trò là mqtt client để gửi hình ảnh đến broker. Server này có thông số là CPU: Intel I3 1005G1 1.2GHz, Ram 8GB (tương đương với raspberry pi 4)

1.3.2. Phạm vi nghiên cứu

Đối với đề tài về camera an ninh, nhóm em sẽ chỉ tập trung vào nhóm các camera dùng trong các hộ gia đình vì đây thường là các camera thông thường chỉ có tính năng quay phim và lưu trữ. Hoặc là tại các chung cư, nơi mà các camera thường được đặt tại các dãy hành lang để tạo tiền đề cho các hư phát triển về sau

CHƯƠNG 2: TỔNG QUAN

2.1. Các vấn đề tồn tại

a) Quyền riêng tư

Việc sử dụng camera an ninh trong hộ gia đình có thể xâm phạm đến quyền riêng tư của những người sống trong nhà. Việc giám sát và ghi lại hoạt động của gia đình có thể gây ra sự không thoải mái và mâu thuẫn trong mối quan hệ gia đình. Do đó, trước khi cài đặt camera an ninh, bạn cần xem xét và thảo luận với gia đình về việc sử dụng camera và đảm bảo tuân thủ các quy định về quyền riêng tư và pháp luật.

b) Vùng phủ sóng và góc quan sát

Để đảm bảo hiệu quả của hệ thống camera an ninh, bạn cần xác định các vị trí lắp đặt camera phù hợp. Điều này đòi hỏi đánh giá vùng phủ sóng và góc quan sát để đảm bảo rằng các khu vực quan trọng trong hộ gia đình được bao phủ và giám sát đầy đủ.

c) Kết nối và lưu trữ dữ liệu

Camera an ninh thường yêu cầu kết nối internet và lưu trữ dữ liệu. Bạn cần đảm bảo rằng hệ thống mạng trong nhà ổn định để camera có thể hoạt động một cách liên tục. Ngoài ra, bạn cần xem xét các tùy chọn lưu trữ dữ liệu, bao gồm việc sử dụng đám mây (cloud) hoặc lưu trữ dữ liệu trên thiết bị cục bộ.

d) Phát hiện và cảnh báo

Một vấn đề quan trọng là khả năng phát hiện và cảnh báo khi có hoạt động bất thường trong khu vực được giám sát. Các hệ thống camera an ninh hiện đại có tích hợp các tính năng phát hiện chuyển động và nhận dạng đối tượng. Tuy nhiên, đôi khi có thể xảy ra các sai sót hoặc cảnh báo không chính xác. Điều này đòi hỏi sự cấu hình và điều chỉnh cẩn thận để giảm thiểu các cảnh báo sai và tăng tính chính xác.

2.2. Vấn đề nhóm tập trung giải quyết

Phần lớn các camera an ninh hiện nay chỉ có thể ghi hình và lưu trữ lại video vào một thẻ nhớ hoặc là sẽ có một ứng dụng để người dùng có thể theo dõi và truy cập vào nội dung bên trong thẻ nhớ để xem nội dung đã được ghi lại. Tuy nhiên, nếu có xảy ra những vấn đề không mong muốn như trộm cắp hay có người lạ vào nhà thì gần như những chiếc camera này không mang lại hiệu quả trong việc kiểm soát an ninh. Và khi sự cố đã xảy ra rồi thì ta mới cần đến những chiếc camera này trong khi thứ người dùng thật sự cần là một giải pháp để phòng vệ và ra tay kịp thời. Vì thế, có một hệ thống kiểm soát an ninh để nhận diện con người và lập tức gửi thông báo cho người dùng là một điều cần thiết

CHƯƠNG 3. NGHIÊN CỨU LÝ THUYẾT

3.1. Cơ sở lý thuyết

a) Giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhị phân, đơn giản và được thiết kế để truyền tải thông điệp giữa các thiết bị trong mạng máy tính. MQTT thường được sử dụng trong các ứng dụng IoT (Internet of Things) để giao tiếp giữa các thiết bị.

Cơ chế hoạt động của MQTT:

- **Publisher:** Là thiết bị hoặc ứng dụng gửi thông điệp tới một hoặc nhiều topic trên broker MQTT. Publisher chịu trách nhiệm tạo và gửi các thông điệp đến broker thông qua các chủ đề.
- **Subscriber:** Là thiết bị hoặc ứng dụng đăng ký theo dõi các topic trên broker MQTT để nhận thông điệp từ các topic đó. Subscriber nhận các thông điệp mà nó quan tâm từ broker, bằng cách đăng ký các chủ đề.
- **Broker:** Là máy chủ trung gian hoạt động như một cầu nối giữa các Publisher và Subscriber. Broker nhận thông điệp từ Publisher và chuyển tiếp đến các Subscriber đã đăng ký theo dõi các topic tương ứng. MQTT Broker có trách nhiệm quản lý các phiên kết nối, quản lý các topic và định tuyến các thông điệp.

b) Thuật toán YOLO

YOLO (You Only Look Once) là một thuật toán phát hiện đối tượng trong thị giác máy tính được phát triển bởi Joseph Redmon, Ali Farhadi và Object Detection Lab tại Đại học Washington. Nó được giới thiệu lần đầu tiên trong bài báo "You Only Look Once: Unified, Real-Time Object Detection" vào năm 2016.

Điểm đặc biệt của YOLO so với các thuật toán phát hiện đối tượng trước đây như R-CNN, Faster R-CNN, SSD là nó thực hiện phát hiện và phân loại đối tượng chỉ trong một lần truyền hình ảnh duy nhất. Điều này giúp YOLO đạt được tốc độ xử lý nhanh hơn nhiều so với các thuật toán khác, đồng thời cũng đơn giản hơn về mặt kiến trúc mạng nơ-ron.

c) Thư viện ultralytics

Ultralytics là một thư viện mã nguồn mở mạnh mẽ cho việc nhận dạng và phân loại đối tượng trong ảnh và video. Với các mô hình tiên tiến như YOLOv5 và EfficientDet, Ultralytics cung cấp các công cụ dễ sử dụng để xử lý thị giác máy tính. Nó hỗ trợ đào tạo mô hình, đánh giá hiệu suất và cung cấp các bộ dữ liệu phổ biến như COCO và Pascal VOC. Với Ultralytics, bạn có thể triển khai các ứng dụng thị giác máy tính dễ dàng và hiệu quả.

Hiện tại, thư viện ultralytics chỉ hỗ trợ cho 2 phiên bản YOLOv5 và YOLOv8 với những model đã được pretrain. Dưới đây là một số phiên bản model đã được pretrain và thông số của chúng.



Performance Metrics

Performance

Detection

See [Detection Docs](#) for usage examples with these models trained on [COCO](#), which include 80 pre-trained classes.

Model	YAML	size (pixels)	mAP ^{val} ₅₀₋₉₅	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
yolov5nu.pt	yolov5n.yaml	640	34.3	73.6	1.06	2.6	7.7
yolov5su.pt	yolov5s.yaml	640	43.0	120.7	1.27	9.1	24.0
yolov5mu.pt	yolov5m.yaml	640	49.0	233.9	1.86	25.1	64.2
yolov5lu.pt	yolov5l.yaml	640	52.2	408.4	2.50	53.2	135.0
yolov5xu.pt	yolov5x.yaml	640	53.2	763.2	3.81	97.2	246.4
yolov5n6u.pt	yolov5n6.yaml	1280	42.1	211.0	1.83	4.3	7.8
yolov5s6u.pt	yolov5s6.yaml	1280	48.6	422.6	2.34	15.3	24.6
yolov5m6u.pt	yolov5m6.yaml	1280	53.6	810.9	4.36	41.2	65.7
yolov5l6u.pt	yolov5l6.yaml	1280	55.7	1470.9	5.47	86.1	137.4
yolov5x6u.pt	yolov5x6.yaml	1280	56.8	2436.5	8.98	155.4	250.7

Hình 1. Thông số của YOLOv5

Performance Metrics

Performance

Detection (COCO) Detection (Open Images V7) Segmentation (COCO) Classification (ImageNet) Pose (COCO) OBB (DOTAv1)

See [Detection Docs](#) for usage examples with these models trained on [COCO](#), which include 80 pre-trained classes.

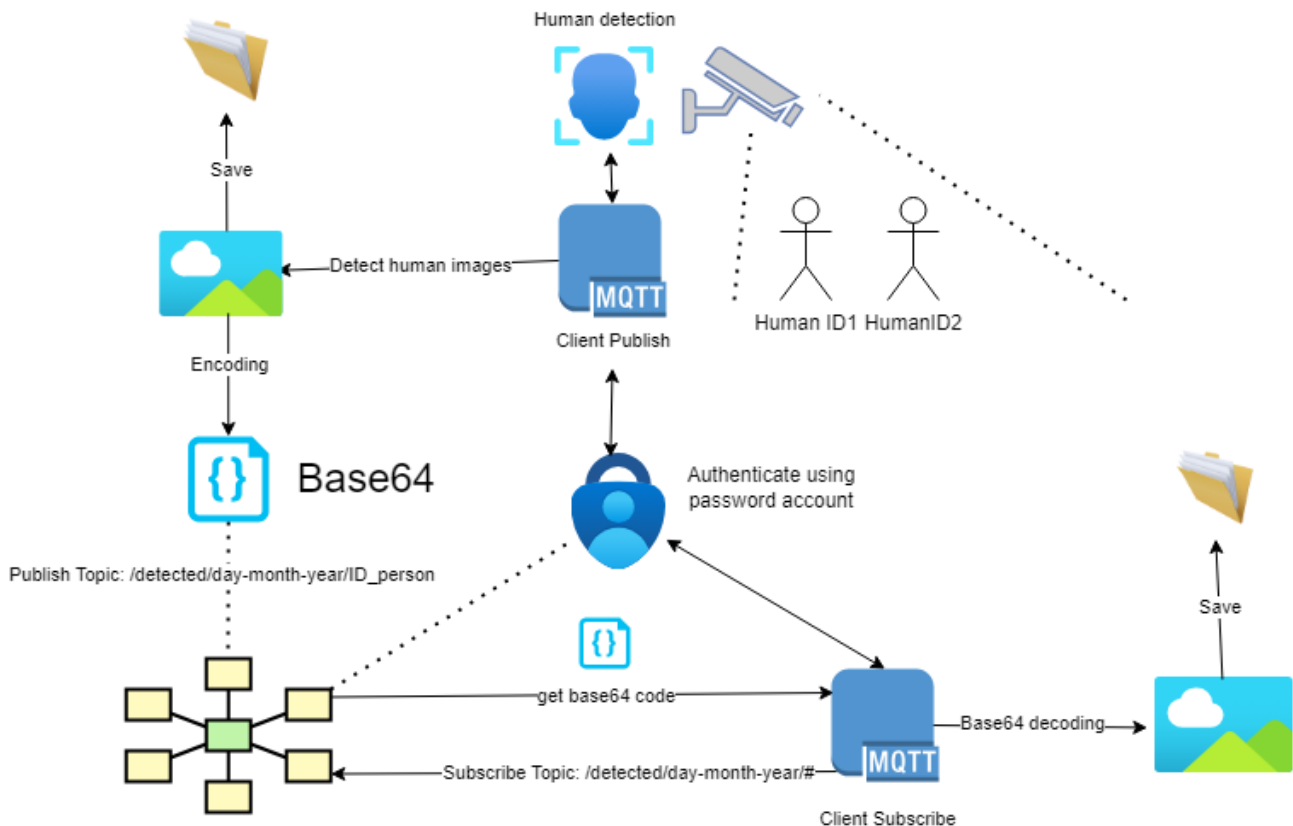
Model	size (pixels)	mAP ^{val} ₅₀₋₉₅	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Hình 2. Thông số của YOLOv8

Xét về độ chính xác thì YOLOv8 hoàn toàn hơn hẳn so với phiên bản tiền nhiệm nhưng đồng thời số lượng phép tính cũng nhiều hơn và tốc độ xử lý cũng chậm hơn các phiên bản của YOLOv5. Do đó, khi cân nhắc về model được sử dụng trong đồ án, nhóm chúng em quyết định sẽ chọn yolov5nu vì đây là phiên bản nhẹ nhất của YOLOv5 và số độ chính xác tương đối ngang với yolov8n (34.3 – 37.3) nhưng lại có số lượng phép tính ít hơn và tốc độ xử lý

nhấn hơn (73.6 – 80.4). Và với khả năng xử lý có giới hạn của những camera an ninh thì đây sẽ là lựa chọn phù hợp hơn.

3.2. Thiết kế mô hình



MQTT Broker

Hình 3. Mô hình chính của đồ án

Trong mô hình này, ta sẽ sử dụng một camera để ghi hình. Video từ camera sẽ được chia thành các frame để xử lý theo từng khung hình tại mqtt client

Bên trong mqtt client, ta sẽ dùng thuật toán YOLOv5n để nhận diện khuôn con người. Khi model nhận được các frame ảnh, model sẽ xử lý và nhận diện các vật thể có trong frame ảnh đó bao gồm cả con người và các đối tượng này sẽ được phân biệt bằng các box.

```
0: 480x640 1 person, 91.6ms
ultralytics.engine.results.Boxes object with attributes:

cls: tensor([0.])
conf: tensor([0.8746])
data: tensor([[188.2035, 189.6910, 477.8592, 478.9917, 1.0000, 0.8746, 0.0000]])
id: tensor([1.])
is_track: True
orig_shape: (480, 640)
shape: torch.Size([1, 7])
xywh: tensor([[333.0314, 334.3413, 289.6557, 289.3008]])
xywhn: tensor([[0.5204, 0.6965, 0.4526, 0.6027]])
xyxy: tensor([[188.2035, 189.6910, 477.8592, 478.9917]])
xyxyn: tensor([[0.2941, 0.3952, 0.7467, 0.9979]])
Speed: 1.0ms preprocess, 91.6ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
```

Bên trong mỗi box, ta sẽ có khá nhiều thông tin nhưng ta sẽ chỉ tập trung vào một số trường thông tin như sau:

- Cls: chứa thông tin về loại đối tượng đang được nhận diện. Đối với con người thì cls sẽ là 0
- Conf: chứa thông tin về mức độ “tự tin” của model về tính chính xác về đối tượng này
- Id: chứa thông tin về id của từng box để phân biệt giữa các box
- Is_track: khi trường này được set là true thì tức là khi một người đi qua một vật thể bị che khuất tầm nhìn trong thời gian ngắn thì model vẫn có thể tiếp tục track theo người đó để gán lại ID
- Xyxy: chứa thông tin về điểm góc trên bên trái và góc dưới bên phải để tạo thành box

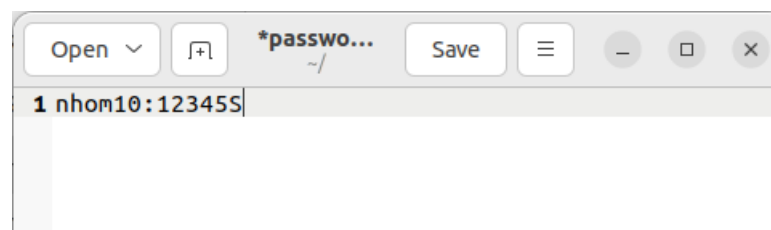
Sau khi đã chia các đối tượng thành các box thì ta có thể dựa vào thông tin cls để phân loại chỉ xử lý các box detect con người. Và từ thông tin về xyxy có trong mỗi box ta có thể cắt các phần ảnh có chứa con người và lưu vào trong local thay vì phải lưu đầy đủ một video hay một tấm hình. Những tấm ảnh sau khi được cắt ra và lưu lại sẽ được mã hoá theo định dạng base64. Ảnh sau khi được mã hoá sẽ được client publish theo topic là detected/day-month-year/ID-person với QOS là 2.

MQTT broker sẽ đóng vai trò phân phối và xác thực bằng username và password cho cả 2 client. Client subscribe sẽ subscribe vào topic detected/day-month-year/# để lấy toàn bộ payload là các đoạn mã ở dạng base64 mà bên phía camera đã nhận được theo từng ngày. Sau đó mã hoá các payload này là đã có thể nhận được các hình ảnh và lưu chúng vào local

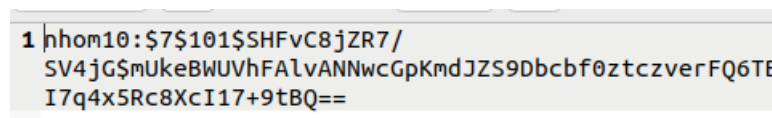
CHƯƠNG 4. TRIỂN KHAI MÔ HÌNH

4.1. Thiết lập xác thực tài khoản mật khẩu trên MQTT Broker

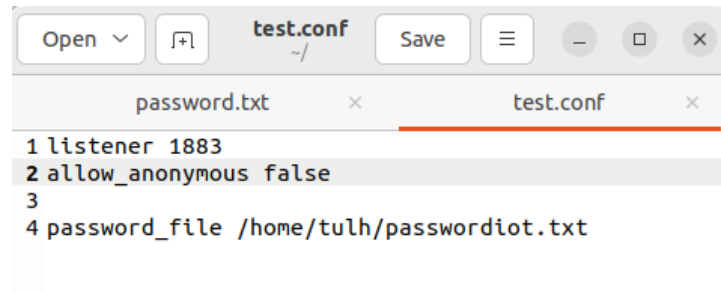
Tạo 1 file chứa username và password



Sau đó sử dụng lệnh **mosquitto_passwd -U password.txt** để tạo ra hàm băm từ mật khẩu đó và gán vào thay cho mật khẩu hiện tại



Tạo 1 file .conf để cấu hình các thông tin liên quan đến mosquitto



listener 1883: lắng nghe port 1883

allow_anonymous: cho phép người ngoài truy cập vào broker của Mosquitto

password_file /home/tulh/passwordiot.txt: cung cấp đường dẫn đến file mật khẩu

```
tulh@tulh:~$ mosquitto -c test.conf -v
1715270970: mosquitto version 2.0.11 starting
1715270970: Config loaded from test.conf.
1715270970: Opening ipv4 listen socket on port 1883.
1715270970: Opening ipv6 listen socket on port 1883.
1715270970: mosquitto version 2.0.11 running
```

Sử dụng lệnh **mosquitto -c test.conf -v** để khởi chạy mosquitto với cấu hình theo file test.conf

```
tulh@tulh:~$ mosquitto_pub -h 192.168.216.54 -t test -m hello -r -u nhom10 -P 12345 -d
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r1, m1, 'test', ... (5 bytes))
Client (null) sending DISCONNECT
tulh@tulh:~$ mosquitto_pub -h 192.168.216.54 -t test -m hello -r -u nhom10 -P 123456 -d
Client (null) sending CONNECT
Client (null) received CONNACK (5)
Connection error: Connection Refused: not authorised.
Error: The connection was refused.
```

Khi published message vào cần nhập đúng username và password nếu không thì kết nối sẽ bị từ chối.

4.2. Chương trình Client_pub.py

Khai báo các thông số về module, mqtt và ngưỡng threshold để bắt đầu gửi hình ảnh

```
model = YOLO("yolov5nu.pt")
broker = '192.168.216.54' # Địa chỉ IP của broker
port = 1883
username = "nhom10" # MQTT broker username
password = "12345" # MQTT broker password
threshold = 0.8 # Ngưỡng chấp nhận để gửi ảnh (%)
```

Dùng hàm VideoCapture trong thư viện cv2 để lấy video từ camera, khởi tạo một dictionary tên detected_people để lưu map dữ liệu về ID và conf của mỗi ID. Mục đích là để so sánh nếu conf lúc sau cao hơn ban đầu thì gửi lại ảnh của ID đó

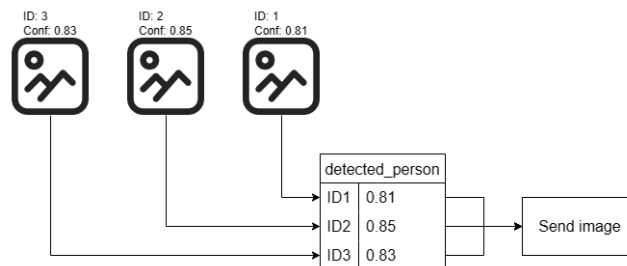
Khởi tạo biến ret để ghi nhận kết quả khi đọc hình ảnh từ camera. Khi không còn đọc được hình ảnh từ camera thì ret sẽ được gán là False và kết thúc vòng lặp. Lưu kết quả nhận diện được từ frame ảnh vào biến result bằng hàm track() trong model. Lí do cho việc sử dụng

hàm track là để khi một người đi qua một vật thể chắn ngang tạm thời thì model vẫn có thể tiếp tục detect người đó sao khi vật thể đó đi qua.

```
while ret:
    ret, frame = cam.read()
    results = model.track(frame, stream=True, persist=True)
```

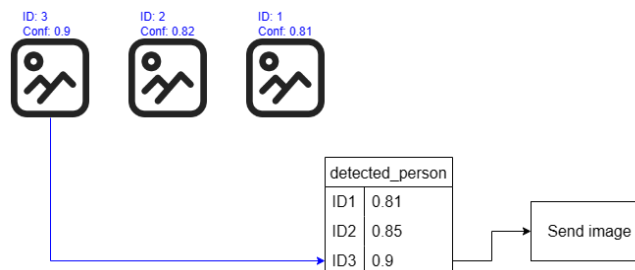
Biến results lúc này sẽ được gán là một đối tượng của class `ultralytics.engine.results.Results`. Trong class này có rất nhiều thông số nhưng ta chỉ cần quan tâm thông số về các boxes.

Trong mỗi box, kiểm tra xem thông tin conf của box có lớn hơn threshold hay không. Nếu có thì bắt đầu lấy các thông tin về ID và conf. Nếu ID chưa tồn tại trong `detected_person` thì lưu ID với conf vào `detected_person`. Sau đó tiến hành gửi cái image vừa nhận được



Hình 4. Khi vừa nhận diện một người mới

Còn nếu ID đã tồn tại tức là hình ảnh người này đã được gửi đi trước đó rồi thì kiểm tra xem ảnh mới nhất chỉ số conf có cao hơn không. Nếu có thì gửi và cập nhật lại conf theo ID trong `detected_person`.



Hình 5. Khi một ID nhận được conf tốt hơn

Việc này là để tránh tình trạng hình ảnh được gửi đi liên tục mà chỉ gửi những bức ảnh dễ nhận diện nhất của một người. Cuối cùng là tạo ra 1 thread để thực thi việc cắt ảnh từ các box và lưu vào bộ nhớ local rồi gửi ảnh đi thông qua mqtt

Về hàm `send_image()`, nhận vào 3 tham số. Thứ nhất là box, để lấy thông tin về kích thước phần ảnh cần cắt ra. Thứ hai là frame, để lấy ảnh cần cắt. Cuối cùng là now để lấy thông tin về ngày tháng năm. Ảnh được cắt sẽ lưu vào `crop_person` dưới dạng ndarray

```
def send_image(box, frame, now):
    x1, y1, x2, y2 = box.xyxy[0] # lưu thông số kích thước của box
    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
    crop_person = frame[y1:y2, x1:x2] # cắt phần ảnh có người
    id_person = str(int(box.id[0]))
    year, month, day, hour, minute, second = (
        str(now.year),
        str(now.month),
        str(now.day),
        str(now.hour),
        str(now.minute),
        str(now.second),
    )
```

Tạo thư mục tại local để lưu trữ hình ảnh với tên file là ngày-tháng-năm-conf

```
file_path = ".\\detected" # Tạo thư mục detect
if not os.path.exists(file_path):
    os.mkdir(file_path)
file_path += "\\ " + day + "-" + month + "-" + year # mỗi ngày là một
thư mục
if not os.path.exists(file_path):
    os.mkdir(file_path)
file_path += "\\ " + "ID" + id_person # thư mục mỗi ngày chứa các thư
mục ID người
if not os.path.exists(file_path):
    os.mkdir(file_path)
image_name = (
    file_path + "\\ "
    + hour + "-"
    + minute + "-"
    + second + conf ".jpg"
)
is_saved = cv2.imwrite(image_name, crop_person)
```

Gọi hàm `connect_mqtt()` để thiết lập và kết nối với một broker MQTT. Khởi động một luồng MQTT để bắt đầu lắng nghe và xử lý các tin nhắn MQTT đến. Tạo chuỗi topic để định danh và phân loại tin nhắn. Trong ví dụ này, chuỗi topic được tạo dựa trên ngày, tháng, năm và ID của người. Mở tệp hình ảnh có tên `image_name` dưới dạng chế độ đọc nhị phân ("rb"). Đọc dữ liệu của tệp hình ảnh và lưu vào biến `img_bytes`. Mã hóa dữ liệu hình ảnh thành chuỗi base64. Điều này cho phép dữ liệu hình ảnh được truyền qua mạng dưới dạng chuỗi văn bản. Tuy nhiên kích thước của dữ liệu sau đi được mã hoá sẽ tăng lên do cứ mỗi 3 byte nhị phân thì sẽ được mã hoá thành 4 byte kí tự và như thế sẽ làm cho dữ liệu sau khi được mã hoá sẽ tăng lên 25%.

21-22-11	JPG File	46 KB
21-22-12	JPG File	39 KB
21-22-13	JPG File	39 KB
21-22-14	JPG File	39 KB
21-22-15	JPG File	42 KB
21-22-16	JPG File	41 KB
21-22-11.jpg_encoded_base64	Text Document	61 KB
21-22-12.jpg_encoded_base64	Text Document	52 KB
21-22-13.jpg_encoded_base64	Text Document	52 KB
21-22-14.jpg_encoded_base64	Text Document	52 KB
21-22-15.jpg_encoded_base64	Text Document	56 KB
21-22-16.jpg_encoded_base64	Text Document	55 KB

Việc mã hoá base64 chỉ đơn giản là để chuyển đổi dữ liệu ảnh sang text để làm payload cho mqtt chứ không mang tính bảo mật. Tin nhắn được gửi đến topic đã được xác định trước đó với qos là 2.

```
client = connect_mqtt()
client.loop_start()
topic = ("detected" + "/"
        + day + "-" + month + "-" + year + "/"
        + "ID" + id_person)
file = open(image_name, "rb")
img_bytes = file.read()
image_b64 = base64.b64encode(img_bytes).decode("utf-8")
client.publish(topic, payload=image_b64, topic=topic)
file.close()
client.loop_stop()
```

Hàm connect_mqtt() được sử dụng để thiết lập và kết nối với một broker MQTT. Đầu tiên, tạo một đối tượng MQTT client bằng cách gọi mqtt_client.Client(), sau đó đặt tên người dùng và mật khẩu cho kết nối sử dụng client.username_pw_set(username, password). Tiếp theo, kết nối đến broker MQTT thông qua client.connect(host=broker, port=port). Cuối cùng, hàm trả về đối tượng client đã được thiết lập.

```
def connect_mqtt():
    client = mqtt_client.Client()
    client.username_pw_set(username, password)
    client.connect(host=broker, port=port)
    return client
```

Hàm publish(client, data, topic) được sử dụng để gửi tin nhắn qua MQTT. Nó nhận đối tượng MQTT client, dữ liệu cần gửi và chủ đề (topic) để gửi tin nhắn tới. Hàm, client.publish(topic=topic, payload=str(data), qos=1) được sử dụng để gửi tin nhắn. Tin nhắn được gửi đến topic và qos là 2. Hàm trả về một kết quả, và result[0] lấy phần tử đầu tiên trong kết quả. Nếu giá trị này là 0, nghĩa là tin nhắn đã được gửi thành công, và thông báo tương ứng được in ra màn hình. Nếu giá trị khác 0, nghĩa là tin nhắn gửi không thành công, và thông báo tương ứng cũng được in ra màn hình.

```
def publish(client, data, topic):
    result = client.publish(topic=topic, payload=str(data), qos=2)
    msg_status = result[0]
    if msg_status == 0:
        print(f"message sent to topic {topic}")
    else:
        print(f"Failed to send message to topic {topic}")
```

4.3. Chương trình Client_sub.py

Khai báo thư viện:

```
import os
import cv2
from datetime import datetime
from paho.mqtt.client import Client
import numpy as np
import base64
```

Khai báo IP và Port của Broker và các biến môi trường về thời gian

```
broker = '192.168.216.54' # Địa chỉ IP của broker
port = 1883
now = datetime.now()
year, month, day = str(now.year), str(now.month), str(now.day)
topic_today = day + "-" + month + "-" + year
```

Khởi tạo hàm on_message(client,userdata,msg) để xử lý payload nhận được, cụ thể sẽ mã hóa ảnh thành mã Base64, sau đó thực hiện lưu ảnh lại ở thư mục với đường dẫn như sau detected/day-month-year/ID_Person, tiếp theo là lưu ảnh về với tên là <giờ> + “h” + <phút> + “p” + <giây> + “s” + “.jpg”

```
def on_message(client, userdata, msg):
    topic = msg.topic
    payload = msg.payload
    print(topic)
    image_data = base64.b64decode(payload)
    nparr = np.frombuffer(image_data, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    global year, month, day, hour, minute, second, now
    now = datetime.now()
    year, month, day, hour, minute, second = (
        str(now.year),
        str(now.month),
        str(now.day),
        str(now.hour),
        str(now.minute),
        str(now.second),
    )
    file_path = ".\\receive"
    if not os.path.exists(file_path):
        os.mkdir(file_path)
    file_path += "\\ " + day + "-" + month + "-" + year
    if not os.path.exists(file_path):
        os.mkdir(file_path)
```

```
image_name = (
    file_path
    + "\\ "
    + hour
    + "h"
    + minute
    + "m"
    + second
    + "s "
    + ".jpg"
)
is_saved = cv2.imwrite(image_name, img)
if is_saved:
    print("ảnh đã được lưu")
else:
    print("ERROR")
```

Khởi tạo chương trình subscribe(client:Client) để subscribe vào topic tất cả topic “detected/” + day +“-” +month + “-” + year /#, sử dụng “#” trong topic để có thể nhận được toàn bộ thông điệp từ topic “detected/” + day +“-” +month + “-” + year / và thông điệp của các topic con của nó.


```
def subscribe(client: Client):
    global topic_today
    topic_today = day + "-" + month + "-" + year
    client.subscribe("detected/"+topic_today+"/#")
    client.on_message = on_message
```

Khởi tạo chương trình connect_mqtt() để thực hiện kết nối tới MQTT broker thông qua tài khoản mật khẩu, nếu kết nối thành công sẽ có thông báo, tương tự như thế nếu không thể kết nối được.

```
def connect_mqtt() -> Client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)
    username = "nhom10"
    password = "12345"
    client = Client()
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Khởi tạo hàm run() đăng ký theo dõi các topic và duy trì kết nối để nhận và xử lý các thông điệp từ broker.

```
def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()
if __name__ == '__main__':
    run()
```

CHƯƠNG 5. ĐÁNH GIÁ KẾT QUẢ

5.1. Kết quả thu được

Những dữ liệu và hình ảnh nhận được từ bộ nhớ và qua giao thức mqtt đều đã được gửi thành công. Tuy nhiên vẫn còn những độ trễ nhất định khi truyền hình ảnh thông qua giao thức MQTT.

5.2. Hình ảnh minh họa

Thử đăng sai mật khẩu để kiểm tra Broker đã được thiết lập xác thực hay chưa

```
broker = '192.168.216.54' # Địa chỉ IP của broker
port = 1883
username = "nhom10" # MQTT broker username
password = "12345234234324" # MQTT broker password wrong
now = datetime.now()
year, month, day = str(now.year), str(now.month), str(now.day)
topic_today = day + "-" + month + "-" + year

> python -u d:\Benjamin_stuff\Python\Project\client_subscribe.py:153: DeprecationWarning: Callb
o latest version
client = Client()
Failed to connect, return code %d
5
Failed to connect, return code %d
5
Failed to connect, return code %d
5
```

Broker

```
tulh@tulh:~$ mosquitto -c test.conf -v
1715244062: mosquitto version 2.0.11 starting
1715244062: Config loaded from test.conf.
1715244062: Opening ipv4 listen socket on port 1883.
1715244062: Opening ipv6 listen socket on port 1883.
1715244062: mosquitto version 2.0.11 running
1715244089: New connection from 192.168.216.32:61296 on port 1883.
1715244089: Sending CONNACK to 192.168.216.32 (0, 5)
1715244089: Client <unknown> disconnected, not authorised.
1715244090: New connection from 192.168.216.32:61297 on port 1883.
1715244090: Sending CONNACK to 192.168.216.32 (0, 5)
1715244090: Client <unknown> disconnected, not authorised.
1715244092: New connection from 192.168.216.32:61299 on port 1883.
1715244092: Sending CONNACK to 192.168.216.32 (0, 5)
1715244092: Client <unknown> disconnected, not authorised.
```

Tiếp theo thì đăng nhập với username và password đúng

```
broker = '192.168.216.54' # Địa chỉ IP của broker
port = 1883
username = "nhom10" # MQTT broker username
password = "12345" # MQTT broker password wrong
now = datetime.now()
year, month, day = str(now.year), str(now.month), str(now.day)
topic_today = day + "-" + month + "-" + year
```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS D:\Benjamin_stuff\Python\Project> python -u "d:\Benjamin_stuff\Python\Project\
d:\Benjamin_stuff\Python\Project\client_subscribe.py:153: DeprecationWarning: Call
o latest version
    client = Client()
Connected to MQTT Broker!

```

```



















ties Terminal

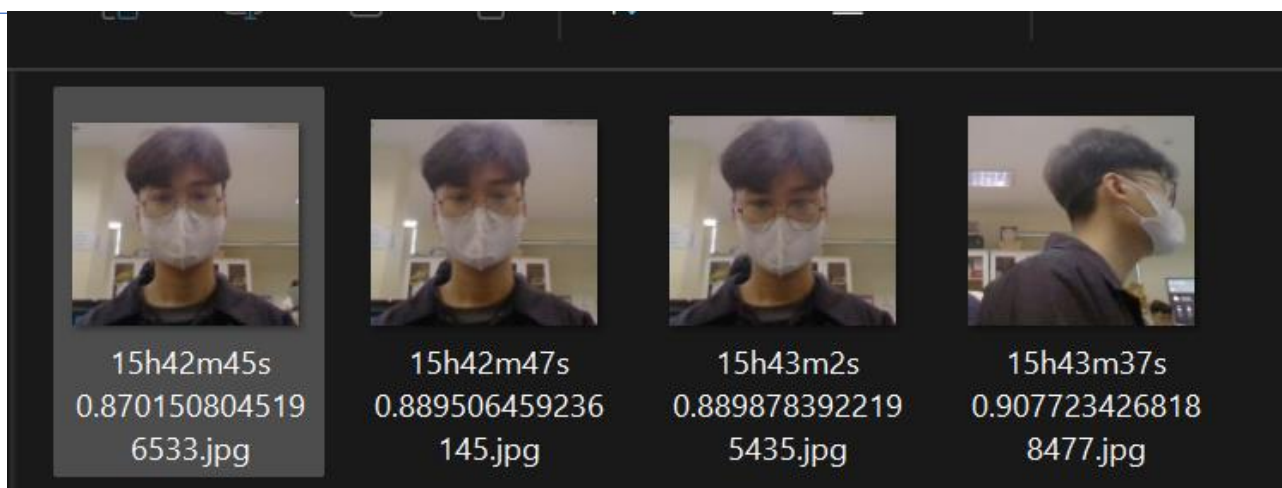
Broker

tulh@tulh:~$ mosquitto -c test.conf -v
1715244062: mosquitto version 2.0.11 starting
1715244062: Config loaded from test.conf.
1715244062: Opening ipv4 listen socket on port 1883.
1715244062: Opening ipv6 listen socket on port 1883.
1715244062: mosquitto version 2.0.11 running
1715244089: New connection from 192.168.216.32:61296 on port 1883.
1715244089: Sending CONNACK to 192.168.216.32 (0, 5)
1715244089: Client <unknown> disconnected, not authorised.
1715244090: New connection from 192.168.216.32:61297 on port 1883.
1715244090: Sending CONNACK to 192.168.216.32 (0, 5)
1715244090: Client <unknown> disconnected, not authorised.
1715244092: New connection from 192.168.216.32:61299 on port 1883.
1715244092: Sending CONNACK to 192.168.216.32 (0, 5)
1715244092: Client <unknown> disconnected, not authorised.
1715244109: New connection from 192.168.216.32:61310 on port 1883.
1715244109: New client connected from 192.168.216.32:61310 as auto-F40B
1715244109: No will message specified.
1715244109: Sending CONNACK to auto-F40B2897-4F74-EF27-356E-8E827C296F2
1715244109: Received SUBSCRIBE from auto-F40B2897-4F74-EF27-356E-8E827C
1715244109:      detected/9-5-2024/# (QoS 0)
1715244109: auto-F40B2897-4F74-EF27-356E-8E827C296F2D 0 detected/9-5-20
1715244109: Sending SUBACK to auto-F40B2897-4F74-EF27-356E-8E827C296F2D

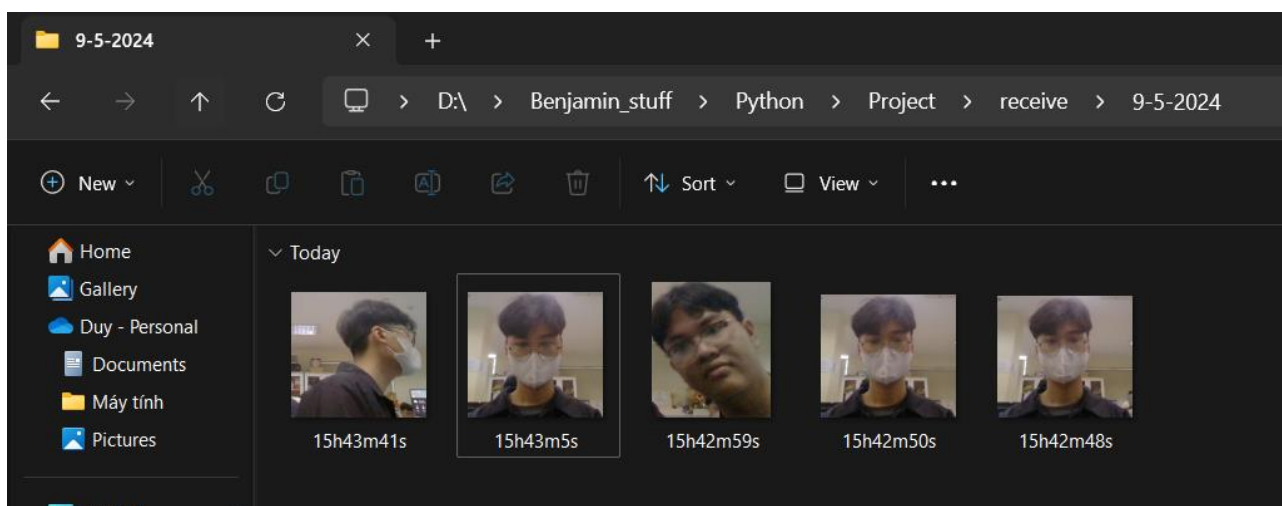
```

Sau khi xác thực thành công, tiến hành phát hiện được người từ camera, sau đó chương trình Client_pub.py sẽ lưu ảnh vào thư mục detected/day-month-year/ID_Person, tên ảnh sẽ được đặt theo mẫu như sau <giờ> + “h” + <phút> + “p” + <giây> + “s”, cuối cùng là mã hóa ảnh đó thành mã Base64 và publish lên Topic là detected/day-month-year/ID_Person.

Name	Date modified	Type
 client_sub.py	5/9/2024 8:29 PM	Python Source File
 client_pub.py	5/9/2024 3:42 PM	Python Source File
 Client.html	4/30/2024 10:46 AM	Chrome HTML Docu
 Biểu đồ không có tiêu đề.drawio.svg	5/9/2024 9:06 PM	Chrome HTML Docu
 Biểu đồ không có tiêu đề.drawio.png	5/9/2024 9:09 PM	PNG File
 Biểu đồ không có tiêu đề.drawio (2).png	5/9/2024 9:39 PM	PNG File
 Biểu đồ không có tiêu đề.drawio (1).png	5/9/2024 9:38 PM	PNG File
 Biểu đồ không có tiêu đề.drawio	5/9/2024 9:07 PM	DRAWIO File
 video demo	5/9/2024 3:52 PM	File folder
 Human_Detection_Yolov5	5/1/2024 12:55 PM	File folder
 http_human_detect	5/3/2024 3:36 PM	File folder
 detected	5/9/2024 3:42 PM	File folder
 .git	5/1/2024 12:52 PM	File folder
Name	Date modified	Type
 9-5-2024	5/9/2024 3:43 PM	File folder
Name	Date modified	Type
 ID1	5/9/2024 3:43 PM	File folder
 ID2	5/9/2024 3:42 PM	File folder
 ID3	5/9/2024 3:43 PM	File folder
 ID4	5/9/2024 3:43 PM	File folder



Sau khi nhận được mã Base64 từ Broker, chương trình Client_sub.py sẽ giải mã Base64 thành ảnh, lưu vào thư mục với tên như sau: receive/day-month-year, vì subscribe vào topic là detected/day-month-year/#, nên là ảnh của toàn bộ mẫu ảnh người phát hiện được đều sẽ được lưu vào đồng loạt một thư mục duy nhất, với tên mỗi ảnh đặt như sau <giờ> + “h” + <phút> + “p” + <giây> + “s”



5.3. Đánh giá

5.3.1. Ưu điểm

- Đơn giản dễ triển khai
- Dễ mở rộng
- Đưa ra thông báo kịp thời
- Độ trễ thấp

5.3.2. Nhược điểm

- Khi số người tăng lên một lượng nhất định, chương trình có thể nhận diện thiếu hoặc nhầm lẫn với đối tượng khác
- Nếu có nhiều hơn một Client publish thì chưa thể kiểm tra có cùng 1 người trên nhiều ảnh hay không
- Chưa có phân quyền giữa các Client, mọi Client đều có quyền publish và subscribe khi đăng nhập thành công

5.3.3. Một số giải pháp hạn chế hoặc khắc phục

- Nâng cấp mô hình nhận diện.
- Cấu hình phân quyền cho từng tài khoản và mật khẩu.

CHƯƠNG 6. KẾT LUẬN

6.1. Tổng kết

Mô hình đơn giản sử dụng mô hình phát hiện người và giao thức MQTT để gửi ảnh giữa các client trong mạng. Mô hình dễ triển khai, dễ tiếp cận, có nhiều cơ hội để nâng cấp và mở rộng. Tuy nhiên vẫn còn nhiều lỗ hổng trong bảo mật, cũng như chưa đáp ứng được hiệu suất phát hiện người nhất định, để đáp ứng với nhu cầu sử dụng camera an ninh ở bối cảnh hiện tại.

6.2. Những khó khăn trong quá trình làm đề tài

Ban đầu nhóm sử dụng ESP32 – Cam để đóng vai trò như một IP camera. Nhưng vào gần cuối lại xảy ra vấn đề thiết bị không hoạt động nên phải chuyển sang dùng camera của laptop

Lần đầu tiên tiếp xúc với các thuật toán nhận diện vật thể nên thời gian ban đầu tìm hiểu tụi em mất khá nhiều thời gian cho việc kiểm thử cho các thuật toán khác nhau rồi cuối cùng mới dùng Yolo.

Camera không đủ tốt, hình ảnh nhận được có thể bị mờ hoặc không rõ nét, dẫn đến khó nhận diện.

CHƯƠNG 7. ĐỊNH HƯỚNG PHÁT TRIỂN

- Triển khai mô hình Sort để tracking trên nhiều camera
- Triển khai lên hệ thống Cloud để dễ dàng thay đổi quy mô hệ thống
- Phân quyền publish/subscribe cho các MQTT Client