

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

XE TỰ HÀNH

GVHD: ThS. HỒ TRUNG MỸ

SVTH: NGUYỄN ĐỨC

NGUYỄN NGỌC LÂM

MSSV: 1410934

1411963

TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2018

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
TRƯỜNG ĐẠI HỌC BÁCH KHOA Độc lập – Tự do – Hạnh phúc.

----- ☆ -----

----- ☆ -----

Số: _____/BKĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Điện Tử**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

- HỌ VÀ TÊN: NGUYỄN ĐỨC
NGUYỄN NGỌC LÂM
MSSV: 1410934
1411963
- NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG**
LỚP : DD14
- Đề tài: Xe tự hành
- Nhiệm vụ:
Thiết kế một xe tự hành có các chức năng sau:
 - Đi theo làn đường giao thông.
 - Nhận diện bốn biển báo giao thông (Stop, Park, Limit_40 và Danger).
 - Tự động tránh vật cản.
- Ngày giao nhiệm vụ luận văn: 15/03/2018
- Ngày hoàn thành nhiệm vụ: 12/06/2018
- Họ và tên người hướng dẫn: **Phản hướng dẫn**
Thầy **HỒ TRUNG MỸ** **Phương án thiết kế một xe tự hành.**

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày..... tháng..... năm 20
CHỦ NHIỆM BỘ MÔN

NGƯỜI HƯỚNG DẪN CHÍNH

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):.....
Đơn vị:.....
Ngày bảo vệ :
Điểm tổng kết:
Nơi lưu trữ luận văn:

LỜI CẢM ƠN

Trong thời gian làm luận văn tốt nghiệp, chúng em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô, gia đình và bạn bè.

Chúng em xin gửi lời cảm ơn chân thành đến giáo viên hướng dẫn là thầy Hồ Trung Mỹ, giảng viên Bộ môn Điện tử, ngành Điện tử - Viễn thông trường Đại học Bách Khoa Thành phố Hồ Chí Minh, người đã có những đóng góp về những yêu cầu, hướng phát triển của đề tài trong suốt quá trình thực hiện luận văn tốt nghiệp.

Chúng em cũng xin chân thành cảm ơn các thầy cô giáo trong trường Đại học Bách Khoa Thành phố Hồ Chí Minh nói chung, các thầy cô trong Khoa Điện- Điện tử nói riêng đã dạy dỗ cho chúng em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp chúng em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ chúng em trong suốt quá trình học tập.

Cuối cùng, chúng em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và hoàn thành luận văn tốt nghiệp này.

Tp. Hồ Chí Minh, ngày 12 tháng 06 năm 2018 .

Sinh viên

Nguyễn Đức

Nguyễn Ngọc Lâm

TÓM TẮT LUẬN VĂN

Luận văn này trình bày về mô hình xe tự hành sử dụng kết hợp giữa Raspberry Pi 3 và Arduino Iot Wifi Uno gồm những yêu cầu cơ bản sau:

- Xe tự lái theo làn đường, xe tự căn làn, chủ động giảm tốc độ khi rẽ trái, phải theo vạch đường.
- Có thể nhận diện bốn loại biển báo giao thông (Stop, Park, Limit_40, Danger).
- Kết hợp với cảm biến siêu âm để có thể tự động phanh tránh vật cản.

Xe tự hành hoạt động theo cơ chế:

- Thu hình ảnh trực tiếp từ Pi Camera.
- Xử lý từng khung ảnh thu được trên Raspberry Pi 3 để xác định được làn đường, từ đó đưa ra thông số góc quẹo và nhận diện bốn loại biển báo giao thông.
- Truyền tín hiệu đã xử lý ảnh từ Raspberry Pi 3 sang Arduino thông qua kết nối Serial của cổng USB.
- Arduino sẽ đưa ra tín hiệu lái thực hiện điều khiển động cơ xe.

MỤC LỤC

| | |
|--|----|
| 1. GIỚI THIỆU | 1 |
| 1.1. Tổng quan..... | 1 |
| 1.2. Tình hình nghiên cứu trong và ngoài nước | 2 |
| 1.2.1. Tình hình nghiên cứu trên thế giới | 2 |
| 1.2.2. Tình hình nghiên cứu tại Việt Nam | 4 |
| 1.3. Nhiệm vụ luận văn | 5 |
| 2. LÝ THUYẾT | 7 |
| 2.1. Các kit phần cứng..... | 7 |
| 2.1.1. Máy tính nhúng Raspberry Pi 3 | 7 |
| 2.1.2. Kit Iot Wifi Uno..... | 10 |
| 2.2. Cài đặt các công cụ lập trình cần thiết | 11 |
| 2.2.1. Cài đặt hệ điều hành cho Raspberry Pi 3 | 11 |
| 2.2.2. Cài đặt Arduino Iot Wifi Uno | 16 |
| 2.2.3. Ứng dụng Blynk | 17 |
| 2.3. Thư viện OpenCV | 18 |
| 2.3.1. Giới thiệu về thư viện mã nguồn mở OpenCV | 18 |
| 2.3.2. Tổ chức thư viện OpenCV | 19 |
| 2.4. Đặc trưng Haar-Like | 20 |
| 2.4.1. Giới thiệu và cách tính giá trị đặc trưng Haar-like | 20 |
| 2.4.2. Các bước tạo đặc trưng cho một biến báo giao thông | 22 |
| 2.5. Mạng CNN (Convolutional Neural Network) | 25 |
| 2.6. Các phần cứng phụ..... | 28 |
| 2.6.1. Cảm biến siêu âm SRF05 | 28 |
| 2.6.2. Mạch điều khiển động cơ L298N | 29 |

| | |
|--|----|
| 2.6.3. Camera Raspberry Pi V1 5MP | 31 |
| 3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG | 32 |
| 3.1. Yêu cầu thiết kế..... | 32 |
| 3.2. Sơ đồ khối tổng quát | 33 |
| 3.3. Sơ đồ khối chi tiết | 33 |
| 4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM | 34 |
| 4.1. Yêu cầu đặt ra cho phần mềm..... | 34 |
| 4.2. Giải thuật chương trình chính | 35 |
| 4.3. Giải thuật các chương trình con | 36 |
| 4.3.1. Giải thuật nhận diện biển báo giao thông | 36 |
| 4.3.2. Giải thuật điều khiển xe bằng tay thu thập dữ liệu | 37 |
| 4.3.3. Giải thuật đo khoảng cách | 38 |
| 5. KẾT QUẢ THỰC HIỆN | 38 |
| 5.1. Cách thức đo đạc, thử nghiệm..... | 38 |
| 5.1.1. Các phần cứng và phần mềm được sử dụng trong thực thi chương trình..... | 38 |
| 5.1.2. Các bước tiến hành | 38 |
| 5.2. Dữ liệu huấn luyện | 40 |
| 5.2.1. Làn đường..... | 40 |
| 5.2.2. Biển báo giao thông | 41 |
| 5.3. Kết quả thu được | 42 |
| 5.3.1. Dữ liệu tạo ra sau khi huấn luyện | 42 |
| 5.3.2. Kết quả kiểm chứng..... | 42 |
| 5.3.3. Kết quả thi công..... | 46 |
| 5.3.4. Kết quả nhận diện biển báo giao thông | 47 |
| 5.3.5. Kết quả mô phỏng chạy tự động..... | 48 |

| | |
|---|----|
| 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... | 49 |
| 6.1. Kết luận | 49 |
| 6.2. Hướng phát triển | 50 |
| 7. TÀI LIỆU THAM KHẢO | 51 |
| 8. PHỤ LỤC | 52 |
| 8.1. Code huấn luyện dữ liệu (hình ảnh và thông số) làn đường | 52 |
| 8.2. Code thu thập dữ liệu (bên Raspberry) | 54 |
| 8.3. Code thu thập dữ liệu (bên Arduino) | 57 |
| 8.4. Code tự động chạy (bên Raspberry) | 59 |
| 8.5. Code tự động chạy (bên Arduino)..... | 63 |

DANH SÁCH HÌNH MINH HỌA

| | |
|--|----|
| Hình 1.1 Xe hơi công nghệ tự lái thông minh | 1 |
| Hình 1.2 Mô phỏng xe tự hành với các cảm biến nhận diện..... | 3 |
| Hình 1.3 Xe tự lái của Google | 3 |
| Hình 1.4 Vật cản hình người cao 1,7m được đặt trên đường để thử nghiệm..... | 4 |
| Hình 1.5 Mạch (phải) và bộ trợ lái trên chiếc xe..... | 4 |
| Hình 2.1 Các cổng kết nối trên Raspberry Pi 3 | 7 |
| Hình 2.2 Pinout của Iot Wifi Uno | 10 |
| Hình 2.3 File img dùng để cài Raspbian | 11 |
| Hình 2.4 Tìm thư mục chứa file cài đặt..... | 11 |
| Hình 2.5 Chọn file cài đặt Raspbian..... | 12 |
| Hình 2.6 Tiến hành ghi dữ liệu hệ điều hành vào thẻ nhớ | 12 |
| Hình 2.7 Quá trình ghi dữ liệu thành công..... | 13 |
| Hình 2.8 Giao diện của Raspbian trên màn hình kết nối với kit Raspberry Pi 3 | 13 |
| Hình 2.9 Cài đặt thư viện ESP8266 cho Board Arduino..... | 16 |
| Hình 2.10 Cách thức hoạt động của Blynk..... | 17 |
| Hình 2.11 Tổ chức thư viện OpenCV..... | 19 |
| Hình 2.12 Bốn đặt trưng Haar-like cơ bản | 20 |
| Hình 2.13 Các đặc trưng mở rộng của các đặc trưng Haar-like cơ sở | 21 |
| Hình 2.14 Cách tính Integral Image của ảnh..... | 22 |
| Hình 2.15 Ví dụ cách tính nhanh các giá trị mức xám của vùng D trên ảnh” [7] | 22 |
| Hình 2.16 Quá trình khoanh vùng biển báo giao thông để huấn luyện..... | 23 |
| Hình 2.17 Quá trình huấn luyện biển báo giao thông | 24 |
| Hình 2.18. Pixel map của hai ảnh khác nhau | 25 |

| | |
|---|----|
| Hình 2.19 Tính tích chập của một ma trận | 26 |
| Hình 2.20 Pooling một ma trận | 26 |
| Hình 2.21 Dạng tổng thể của một mạng CNN | 27 |
| Hình 2.22 Các layer từ Lower đến Higher | 27 |
| Hình 2.23 Sơ đồ chân của cảm biến siêu âm SRF05..... | 29 |
| Hình 2.24 Sơ đồ chân Module điều khiển động cơ L298N..... | 30 |
| Hình 2.25 Ảnh thực tế Pi Camera | 31 |
| Hình 2.26 Kích hoạt (enable) cho Camera kết nối với Raspberry Pi 3 | 32 |
| Hình 5.1 Project để điều khiển xe bằng tay (góc quẹo và tốc độ) trên Blynk..... | 39 |
| Hình 5.2 File .csv chứa thông số góc quẹo ứng với khung ảnh tương ứng..... | 40 |
| Hình 5.3 Hình ảnh đã được lấy ngưỡng dùng để huấn luyện..... | 40 |
| Hình 5.4 Ảnh tích cực (chứa đối tượng huấn luyện) định dạng bmp..... | 41 |
| Hình 5.5 Ảnh phủ định (không chứa đối tượng huấn luyện) đã được xám hóa..... | 41 |
| Hình 5.6 Kết quả và file thu được sau khi huấn luyện lần đường..... | 42 |
| Hình 5.7 File xml thu được sau khi huấn luyện biến báo giao thông..... | 42 |
| Hình 5.8 Mẫu thử cho biến báo Park..... | 43 |
| Hình 5.9 Mẫu thử cho biến báo Stop..... | 44 |
| Hình 5.10 Mẫu thử cho biến báo Limit_40 | 44 |
| Hình 5.11 Mẫu thử cho biến báo Danger | 44 |
| Hình 5.12 Mô hình xe tự hành..... | 46 |
| Hình 5.13 Kết quả nhận diện nhiều biển báo cùng lúc..... | 47 |
| Hình 5.14 Mô hình làn đường để xe chạy | 48 |

1. GIỚI THIỆU

1.1. Tổng quan

“Hiện đã có rất nhiều tập đoàn sản xuất xe hơi và công nghệ lớn trên thế giới đã tham gia vào cuộc chạy đua phát triển xe hơi công nghệ tự lái thông minh (sau đây gọi tắt là xe tự lái, xe tự hành) mà không cần đến bàn tay can thiệp của con người, trong đó có những tên tuổi nổi bật như Tesla, Daimler, Google,...

Xu thế xe tự lái đang khiến cho các tập đoàn sản xuất ô tô truyền thống lo ngại, xe tự lái có thể trở thành sự cạnh tranh rất nghiêm trọng, trở thành đối thủ lớn của ngành công nghiệp sản xuất ô tô truyền thống. Hiện General Motors, Toyota hay những hãng khác đang thử nghiệm công nghệ xe tự lái, để chuẩn bị cho xu thế cạnh tranh mới trong tương lai.

Dù tiêu cực hay tích cực, xe tự lái có thể là xu thế phát triển tất yếu của guồng quay kỹ thuật tiên tiến. Công chúng sẽ không mong đợi sự xuất hiện trong vài năm tới, thậm chí cả thập kỷ tiếp theo, nhưng sẽ từng bước tiếp cận với hình thức lái xe tự động để chuẩn bị cho sự bùng nổ công nghệ trong tương lai.” [2]



Hình 1.1 Xe hơi công nghệ tự lái thông minh

1.2. Tình hình nghiên cứu trong và ngoài nước

1.2.1. Tình hình nghiên cứu trên thế giới

“Xuất hiện lần đầu tiên năm 1939 dưới sự tài trợ của General Motors tại Hội chợ thế giới, xe không người lái chủ yếu phục vụ mục đích nghiên cứu khoa học. Thế nhưng những năm gần đây, các hãng xe lớn như GM, Volkswagen (VW), Audi, BMW, Volvo hay Cadillac đã bắt đầu thử nghiệm hệ thống tự vận hành trên nhiều mẫu xe ứng dụng.

Trong năm 2005, BMW đã cho những phiên bản tự hành chạy trên đường thử. Năm 2008, GM công bố khởi đầu những nghiên cứu xe không người lái cho tới năm 2015 và lần bán trên đường năm 2018. Hai năm sau đó, phiên bản tự hành Audi TTS đã đạt tới tốc độ gần với xe đua trên đường lái tới đỉnh núi Pikes Peak, Hoa Kỳ.

Năm 2011 thực sự là năm khởi sắc trong lĩnh vực nghiên cứu xe tự động. Volvo cho biết đã phát triển hệ thống tự động hóa trên đường cao tốc và sẽ tích hợp trên xe năm 2020. Bên cạnh đó, ngài Alan Taub, phó chủ tịch Viện nghiên cứu và phát triển toàn cầu của GM, khẳng định kế hoạch ra mắt xe bán tự động năm 2015 và xe tự động toàn phần năm 2020.

Không nằm ngoài xu hướng đó, VW hiện nay đang thử nghiệm hệ thống điều khiển tự động tạm thời (TAP) cho phép xe tự lái với tốc độ lên tới 128 km/giờ trên đường cao tốc. Nhà máy ô tô tự động đầu tiên của Trung Quốc kết hợp với Đại học Công nghệ Quốc phòng Quốc gia đã thử nghiệm mẫu xe không người lái Hongqi HQ3 trên quãng đường 280 km, với vận tốc 88 km/giờ trong điều kiện đường cao tốc tấp nập.

Những tháng đầu năm 2012, Google đã thử nghiệm phiên bản tự hành của mẫu xe Toyota Prius trên những con đường bang California. Mới đây, Cadillac ra mắt hệ thống bán tự động có tên gọi “Super Cruise” với hứa hẹn sẽ sớm đưa vào sản xuất.”

[2]



Hình 1.2 Mô phỏng xe tự hành với các cảm biến nhận diện



Hình 1.3 Xe tự lái của Google

1.2.2. Tình hình nghiên cứu tại Việt Nam

“Mới đây, các kỹ sư của Việt Nam (nhóm nghiên cứu FPT Software) đã lần đầu tiên cho lăn bánh thử nghiệm thành công một chiếc xe ô tô tự lái.



Hình 1.4 Vật cản hình người cao 1,7m được đặt trên đường để thử nghiệm



Hình 1.5 Mạch (phải) và bộ trợ lái trên chiếc xe

Tại buổi thử nghiệm, xe ô tô tự lái có thể chạy trong khuôn viên của tòa nhà với tốc độ 25km/h. Trong quá trình di chuyển, xe tự căn làn, chủ động rẽ trái, phải theo vạch đường, giảm tốc độ khi vào khúc cua, xác định đối tượng trên đường để tự động phanh và vòng tránh vật cản.

Thông tin trên đường được thu thập bởi hệ thống camera, công nghệ radar, cảm biến siêu âm, sóng laser gắn trên nóc và xung quanh xe. Dữ liệu sẽ được tự động xử lý và máy tính trung tâm đưa ra lệnh đánh lái phù hợp với tình hình thực tế.

Sau 1 năm nghiên cứu và phát triển, các nhà nghiên cứu trong nước đã tham gia phát triển cả phần cứng và phần mềm của ô tô tự lái. Hiện tình trạng giao thông ở những thành phố lớn của Việt Nam như Hà Nội, Hồ Chí Minh vẫn còn là một thách thức để ứng dụng xe tự lái.” [3] Dự kiến, phải mất 10 năm nữa để cập nhật phát triển công nghệ thì xe tự lái mới có thể chạy trên đường phố ở Việt Nam. Tuy nhiên trước mắt, xe tự lái được ứng dụng trong sân golf, trong kho hàng, di chuyển giữa các tòa nhà hay trong sân bay là hoàn toàn khả thi.

Việt Nam hiện đang ở những bước đi đầu tiên, nghiên cứu và ứng dụng xe tự lái. Các viện nghiên cứu, các đại học lớn ở Việt Nam đang rất hào hứng tham gia ở lĩnh vực còn nhiều tiềm năng này.

Với sự bùng nổ của trí tuệ nhân tạo, các hãng ô tô đang đầu tư mạnh tay cho xe không người lái, tuy nhiên, sân chơi này giờ không còn là sân chơi của các hãng ô tô truyền thống mà đã dịch chuyển sang các công ty phần mềm, khi 90% sáng tạo của ô tô nằm ở đây và các nhà khoa học của Việt Nam đang bắt kịp rất nhanh với xu thế này.

1.3. Nhiệm vụ luận văn

Từ nghiên cứu tổng quan và tình hình nghiên cứu trong và ngoài nước, nhận thấy được tiềm năng trong lĩnh vực xe tự hành, đồng thời còn nhiều điểm có thể sáng tạo và phát triển, em chọn đề tài “Xe tự hành”.

Kết quả cần đạt: xe có thể chạy theo làn đường, tự động căn làn, chủ động rẽ trái, phải theo vạch đường, có thể nhận diện một số loại biển báo giao thông, có thể tự động phanh tránh vật cản phía trước.

Để thực hiện được dự án này, ta có những nội dung cần phải hoàn thành:

Nội dung 1: Tìm hiểu lý thuyết Raspberry Pi 3, Arduino Iot Wifi Uno.

- Ý tưởng: Phần cứng phải chính xác, nhỏ gọn, dễ sử dụng và giá cả phù hợp.

- Mục tiêu: Có thể thực hiện tốt giao tiếp với các cảm ứng, động cơ xe, camera, Kit sử dụng phải đủ mạnh để có thể xử lý ảnh, thời gian đáp ứng nhanh.
- Cách tiếp cận: so sánh giữa các bo mạch thông dụng Intel Galileo, Arduino và Raspberry Pi 3. Từ đó chúng em quyết định sử dụng Raspberry Pi 3 kết hợp với Arduino, trong đó Raspberry Pi 3 sẽ có nhiệm vụ thu hình ảnh từ camera, xử lý ảnh thu được, sau đó truyền tín hiệu qua Arduino để điều khiển động cơ.

Nội dung 2: Thiết kế phần mềm nhận diện biển báo giao thông, xác định làn đường xe, đọc khoảng cách đến vật cản.

- Ý tưởng: áp dụng xử lý ảnh trong các giải thuật phát hiện, nhận diện và xử lý các hình ảnh, video từ camera của Raspberry Pi 3.
- Mục tiêu: thực hiện nhận diện biển báo và xác định làn đường xe trên kit Raspberry Pi 3, giải thuật phải tối ưu để thực hiện xe chạy chính xác và tiết kiệm năng lượng, kết hợp với cảm biến siêu âm.
- Cách tiếp cận: sử dụng ngôn ngữ lập trình Python và Arduino, các giải thuật xử lý Haar-Like, thư viện OpenCV, tflearn và các package cần thiết khác.

Nội dung 3: Thiết kế phần cứng là xe tự lái.

- Ý tưởng: Dùng một bo mạch để thực hiện điều khiển động cơ xe, có khả năng kết nối với cảm biến siêu âm, một bo mạch khác xử lý ảnh của giải thuật nhận diện biển báo và làn đường xe như một máy tính.
- Mục tiêu: sử dụng kit Raspberry Pi 3 kết hợp Arduino để có thể thực hiện được những yêu cầu trên.
- Cách tiếp cận: tìm hiểu về kit Raspberry Pi 3 và Arduino, xây dựng khung xe kết hợp với các động cơ, cảm biến siêu âm.

Nội dung 4: Xây dựng mô hình giao thông kiểu mẫu.

- Ý tưởng và mục tiêu: thiết kế một mô hình giao thông thu nhỏ như thực tế, gồm các biển báo giao thông, vạch đường cho xe chạy, các vật cản trên đường.
- Cách tiếp cận: tìm nguyên liệu cần thiết, tham để xây dựng mô hình giao thông.

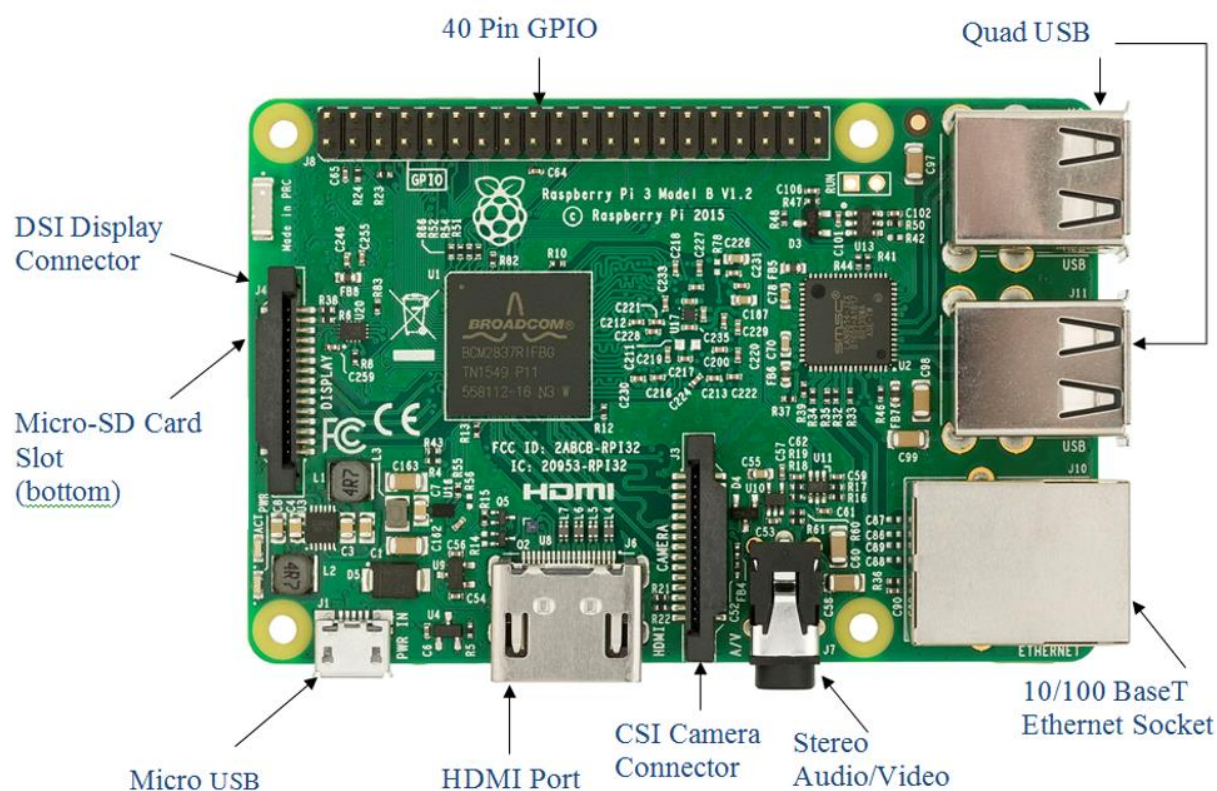
2. LÝ THUYẾT

2.1. Các kit phần cứng

2.1.1. Máy tính nhúng Raspberry Pi 3

Máy tính Raspberry Pi 3 Model B là bo mạch Mini Computer được sử dụng nhiều nhất hiện nay với nhiều ưu điểm:

- CPU phiên bản mới BCM2837 với tốc độ 1.2GHz 4 nhân với kiến trúc ARM Cortex-A53 64-bit. Tốc độ của Pi 3 vượt trội hơn 60 – 70% các phiên bản cũ.
- Tích hợp Wifi chuẩn 802.11n và Bluetooth 4.1.
- Tương thích với thiết kế phần cứng và phần mềm trên các phiên bản cũ.



Hình 2.1 Các cổng kết nối trên Raspberry Pi 3

- **Ethernet:** Giúp mạch Raspberry Pi 3 kết nối với các modem/router để kết nối với Internet (tốc độ tối đa lên đến 10/100 Mb/s).
- **USB 2.0 Host:** 4 cổng USB dùng cho việc nhận tín hiệu các thiết bị ngoại vi như webcam, usb micro, usb,... Raspberry Pi 3 hỗ trợ lên đến 128 thiết bị ngoại vi.
- **Video:** HDMI hỗ trợ phiên bản 1.3/1.4 và Composite RCA (PAL and NTSC)
- **Audio:** Cổng ra 3.5 và HDMI
- **GPIO:** 40 chân
- **Camera:** 15 chân kết nối MIPI Camera Serial Interface (CSI-2) chuyên biệt sử dụng cho Raspberry.
- **Display:** Kết nối Display Serial Interface (DSI)
- **Memory Card Slot:** MicroSD

Thông số kỹ thuật chi tiết Raspberry Pi 3

SOC

| | |
|---------------------|-------------------|
| SOC Type | Broadcom BCM2837 |
| Core Type | Cortex-A53 64-bit |
| No. Of Cores | 4 |
| GPU | VideoCore IV |
| CPU Clock | 1.2 GHz |
| RAM | 1 GB |

Wired Connectivity

| | |
|-------------------|------------|
| USB Ports | Yes |
| Ethernet | Yes |
| SATA Ports | No |

| | |
|-------------------------|-------------------------------------|
| HDMI | Yes |
| Analog Video Out | Yes (shared with audio jack) |
| Analog Audio Out | Yes |
| Analog Audio In | No |
| SPI | Yes |
| I2C | Yes |
| GPIO | Yes |
| LCD Panel | Yes |
| Camera | Yes |
| SD/MMC | Yes microSD |
| Serial | - |

Wireless Connectivity (On-Board)

| | |
|-------------------|--------------------|
| Wi-Fi | Yes 802.11n |
| Bluetooth® | Yes 4.1 |

Dimensions

| | |
|---------------|--------------------|
| Height | 2.22 in (56.5 mm) |
| Width | 3.37 in (85.6 mm) |
| Depth | 0.66929 in (17 mm) |
| Weight | 1.58 oz (45 g) |

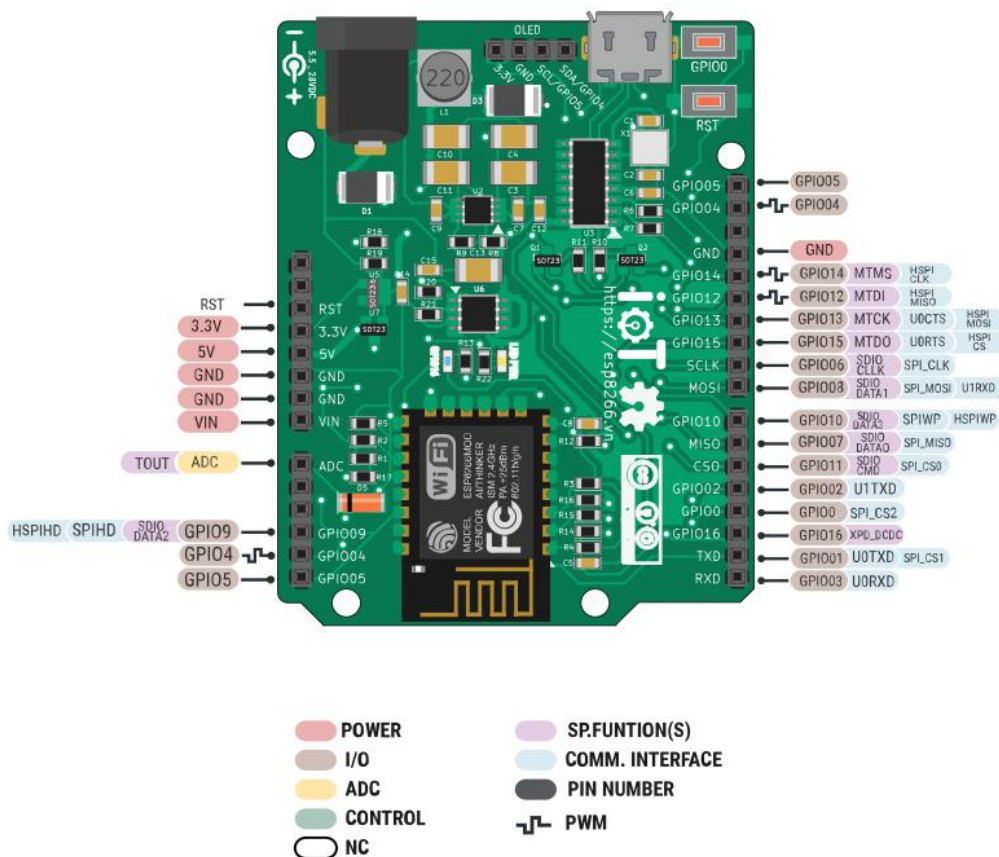
Power

| | |
|----------------------|------------------|
| Power ratings | 800 mA |
| Power sources | microUSB or GPIO |

2.1.2. Kit Iot Wifi Uno

Sử dụng ESP8266 làm bộ điều khiển, tương thích chân với Arduino Uno và mạnh mẽ hơn nhiều:

- Sử dụng 2 nguồn cấp DC từ 5.5V đến 28VDC (1.2A/5V) và nguồn 5V USB
- Tự động vào chế độ nạp qua cổng **Serial**
- Sử dụng với ESP8266 Arduino và các thư viện Arduino
- Hỗ trợ tài liệu từ <https://esp8266.vn>
- Có thêm 1 Nút nhấn và một đèn LED tương tự **NodeMCU**.
- Có thêm 1 header hỗ trợ **OLED**



Hình 2.2 Pinout của Iot Wifi Uno

2.2. Cài đặt các công cụ lập trình cần thiết

2.2.1. Cài đặt hệ điều hành cho Raspberry Pi 3


- **Tổng quan về Raspbian**

Raspberry Pi có rất nhiều hệ điều hành hỗ trợ, trong đó có Raspbian là hệ điều hành chính thức của Raspberry Pi Foundation. Nó cũng được hãng khuyến cáo sử dụng, nhất là cho người mới bắt đầu làm quen với RPI. Hệ điều hành Raspbian dựa trên Debian, hệ điều hành Linux vô cùng lớn và rất nhiều tài liệu phong phú, được tối ưu hoá cho kit Raspberry Pi.

- **Cài đặt Raspbian cho Raspberry Pi 3.**

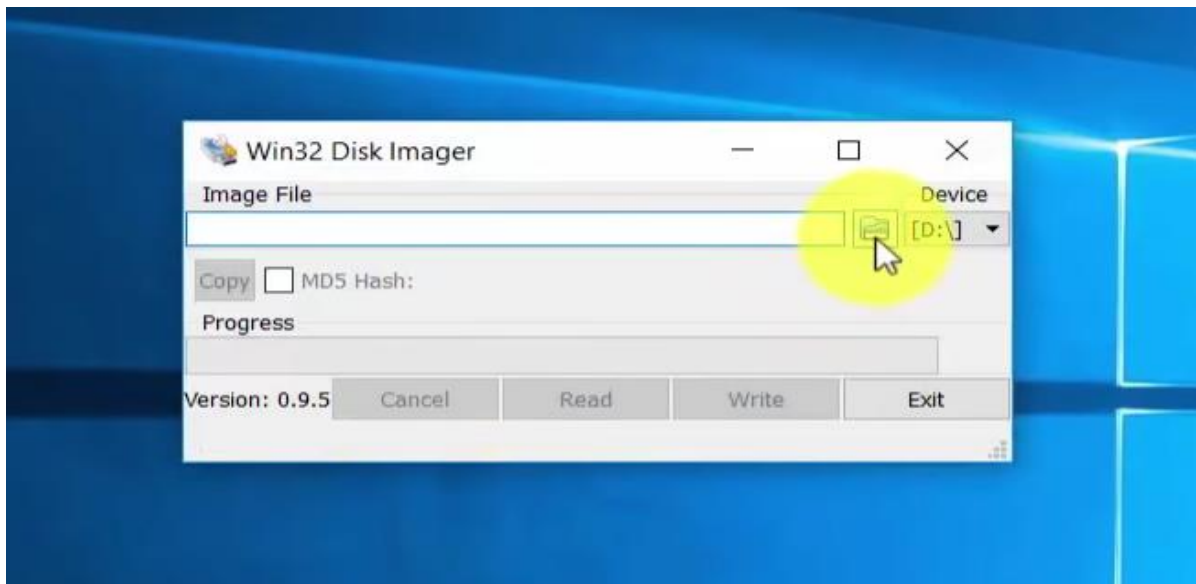
Một thẻ Micro SD Card: sử dụng Mirco SD class 10 dung lượng 16 GB.

Bước 1: Tải file cài đặt

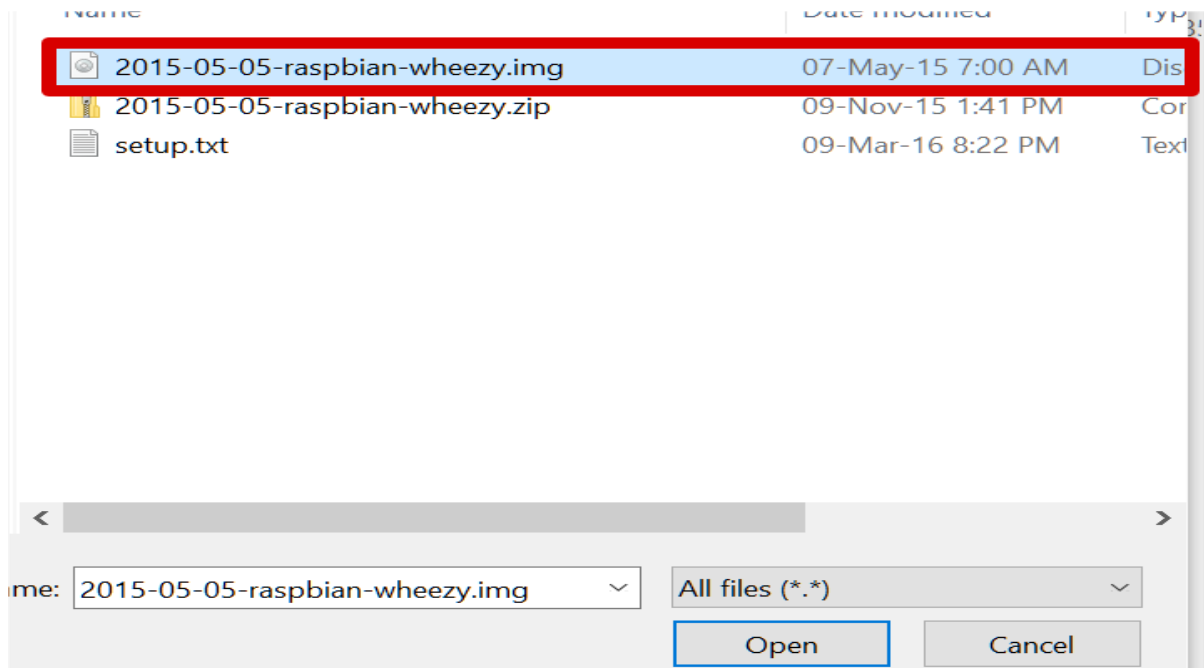
| | | | |
|---|-------------------|-----------------|-------|
|  2015-05-05-raspbian-wheezy.img | 07-May-15 7:00 AM | Disc Image File | 3,200 |
|---|-------------------|-----------------|-------|

Hình 2.3 File img dùng để cài Raspbian

Bước 2: Dùng phần mềm Win32 Disk Imager, chọn file cài đặt

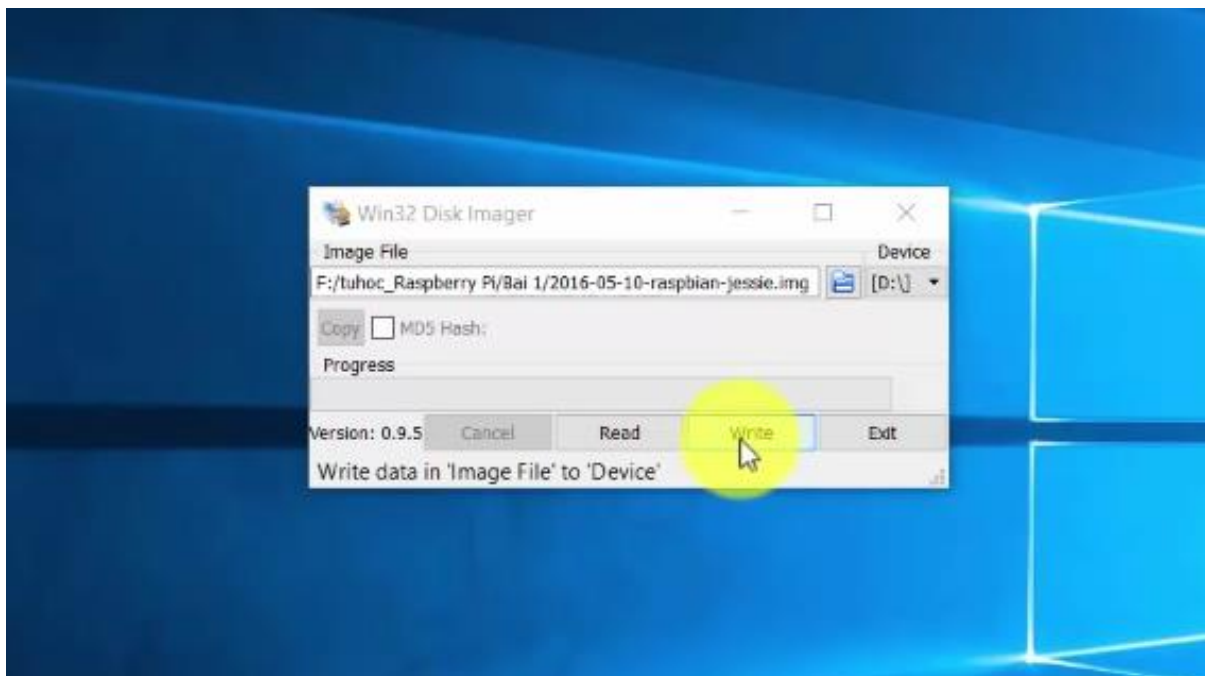


Hình 2.4 Tìm thư mục chứa file cài đặt

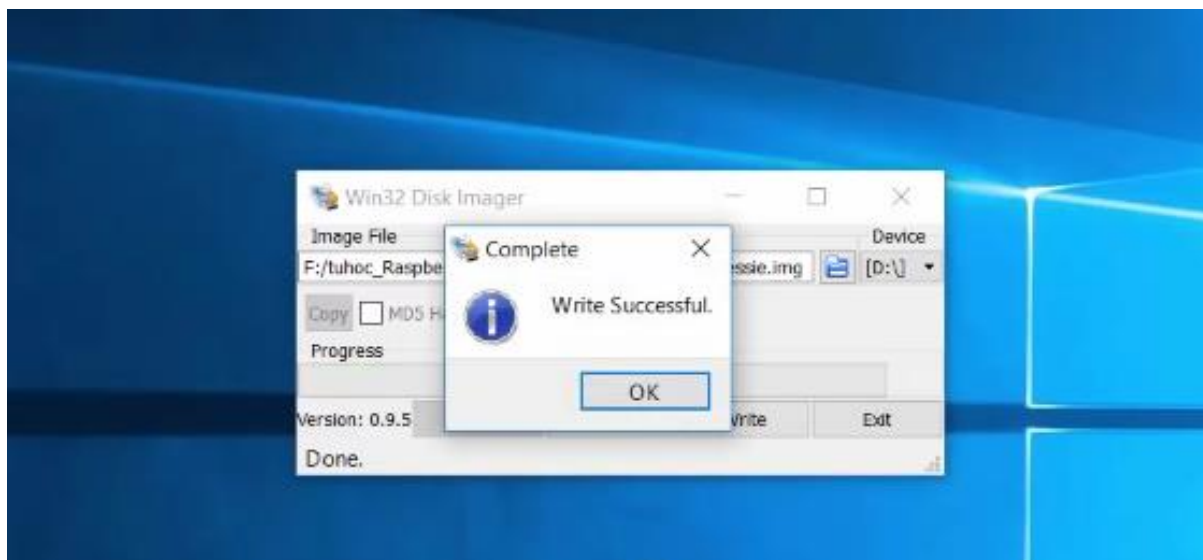


Hình 2.5 Chọn file cài đặt Raspbian

Bước 3: Nhấp vào Write to disk để cài đặt

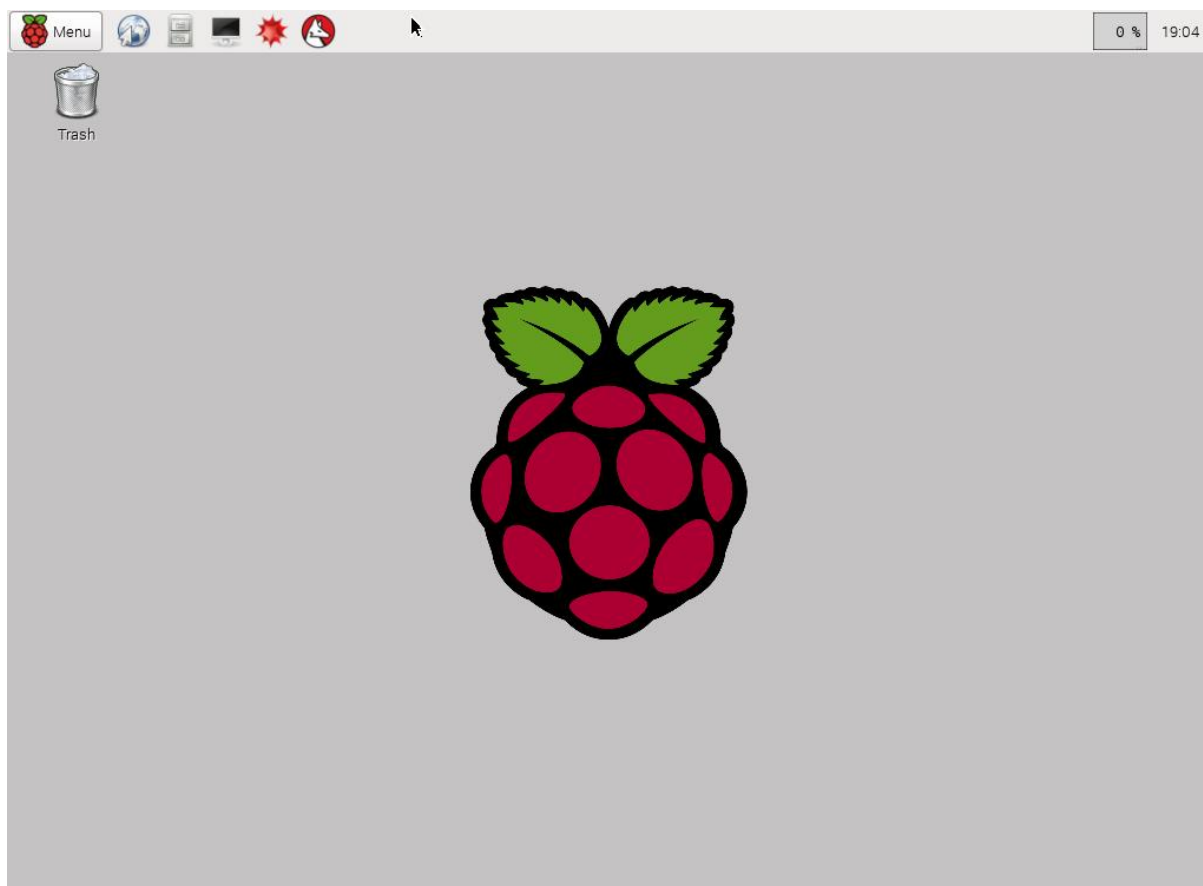


Hình 2.6 Tiến hành ghi dữ liệu hệ điều hành vào thẻ nhớ



Hình 2.7 Quá trình ghi dữ liệu thành công

Bước 4: Sau khi cài đặt hoàn thành, lắp thẻ SD vào khe Micro SD Card của Raspberry Pi 3, cấp nguồn và chạy kit.



Hình 2.8 Giao diện của Raspbian trên màn hình kết nối với kit Raspberry Pi 3

- **Cài đặt Python 3 thư viện OpenCV, tflearn và các package cần thiết khác để xử lý ảnh bằng ngôn ngữ lập trình Python.**

Ta sẽ gõ lệnh trên cửa sổ Terminal để cài đặt các gói trên Raspbian.

Bước 1: Mở rộng hệ thống tệp.

+ *sudo raspi-config > Advanced Options > A1 Expand filesystem > Nhấn "Enter".*

+ Khởi động lại Raspberry: *sudo shutdown -r now.*

Bước 2: Giải phóng ổ đĩa (tùy chọn)

sudo apt-get purge wolfram-engine

*sudo apt-get purge libreoffice**

sudo apt-get clean

sudo apt-get autoremove

Bước 3: Cài đặt các gói phụ thuộc.

+ CMAKE: *sudo apt-get install build-essential cmake pkg-config -y*

+ Image I/O: *sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev -y*

+ Video I/O:

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y

sudo apt-get install libxvidcore-dev libx264-dev -y

+ GTK GUI: *sudo apt-get install libgtk2.0-dev libgtk-3-dev -y*

+ Optimization: *sudo apt-get install libatlas-base-dev gfortran -y*

Bước 4: Cài đặt Python 3 và numpy.

sudo apt-get install python3 python3-setuptools python3-dev -y

wget <https://bootstrap.pypa.io/get-pip.py>

sudo python3 get-pip.py

sudo pip3 install numpy

Bước 5: Tải OpenCV 3.4.0

cd~

wget -O opencv.zip <https://github.com/Itseez/opencv/archive/3.4.0.zip>

unzip opencv.zip

wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.4.0.zip

unzip opencv_contrib.zip

Bước 6: Biên dịch và cài đặt OpenCV 3.4.0 cho Python 3.

```
cd opencv-3.4.0
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D BUILD_opencv_java=OFF \
-D BUILD_opencv_python2=OFF \
-D BUILD_opencv_python3=ON \
-D PYTHON_DEFAULT_EXECUTABLE=$(which python3) \
-D INSTALL_C_EXAMPLES=OFF \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D BUILD_EXAMPLES=ON \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.0/modules \
-D WITH_CUDA=OFF \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS= OFF ..
```

Bước 7: Trao đổi kích thước Space trước khi biên dịch để thêm nhiều bộ nhớ ảo.
Mở `/etc/dphys-swapfile` và chỉnh sửa giá trị `CONF_SWAPSIZE`.

```
sudo nano /etc/dphys-swapfile
CONF_SWAPSIZE=1024.
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

Bước 8: Chuẩn bị biên dịch.

```
make -j4
```

Bước 9: Cài đặt kiến trúc.

```
sudo make install
sudo ldconfig
```

Bước 10: Kiểm tra OpenCV 3.4.0 đã được cài đặt trên Python 3.

```
import cv2
cv2.__version__
```

Bước 11: Các đặt tensorflow và các gói cần thiết khác.

+ Tensorflow:

```
wget https://github.com/samjabrahams/tensorflow-on-raspberry-pi/releases/download/v1.1.0/tensorflow-1.1.0-cp34-cp34m-linux\_armv7l.whl
sudo pip3 install tensorflow-1.1.0-cp34-cp34m-linux_armv7l.whl
sudo pip3 uninstall mock
sudo pip3 install mock
```

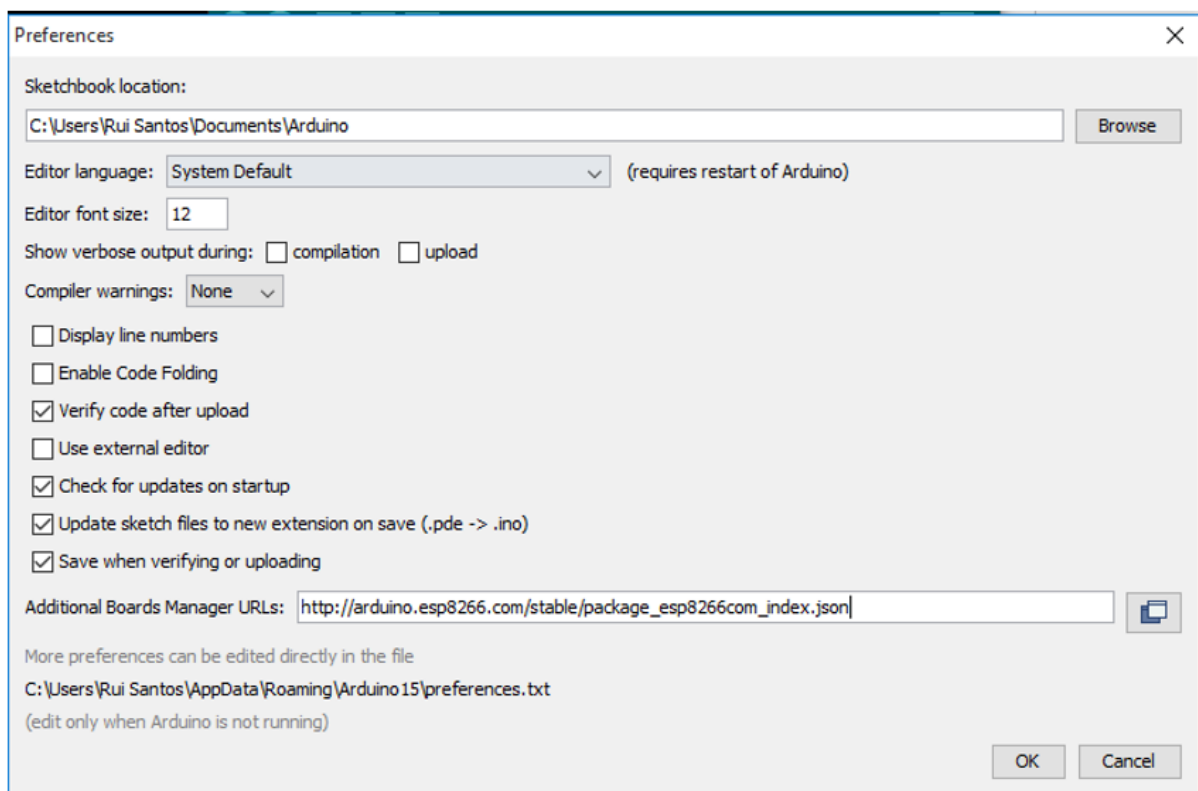

- + Tflern: `sudo pip3 install tflearn`.
- + Scipy: `sudo pip3 install scipy`
- + Scikit-learn: `sudo pip3 install scikit-learn`
- + Scikit-image: `sudo pip3 install scikit-image`
- + pandas: `sudo pip3 install pandas`

2.2.2. Cài đặt Arduino Iot Wifi Uno

Bước 1: Cài đặt Arduino IDE tải từ Arduino website.

Bước 2: Mở chương trình Arduino và cửa sổ Preferences.

- Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` vào **Additional Board Manager URLs**.



Hình 2.9 Cài đặt thư viện ESP8266 cho Board Arduino

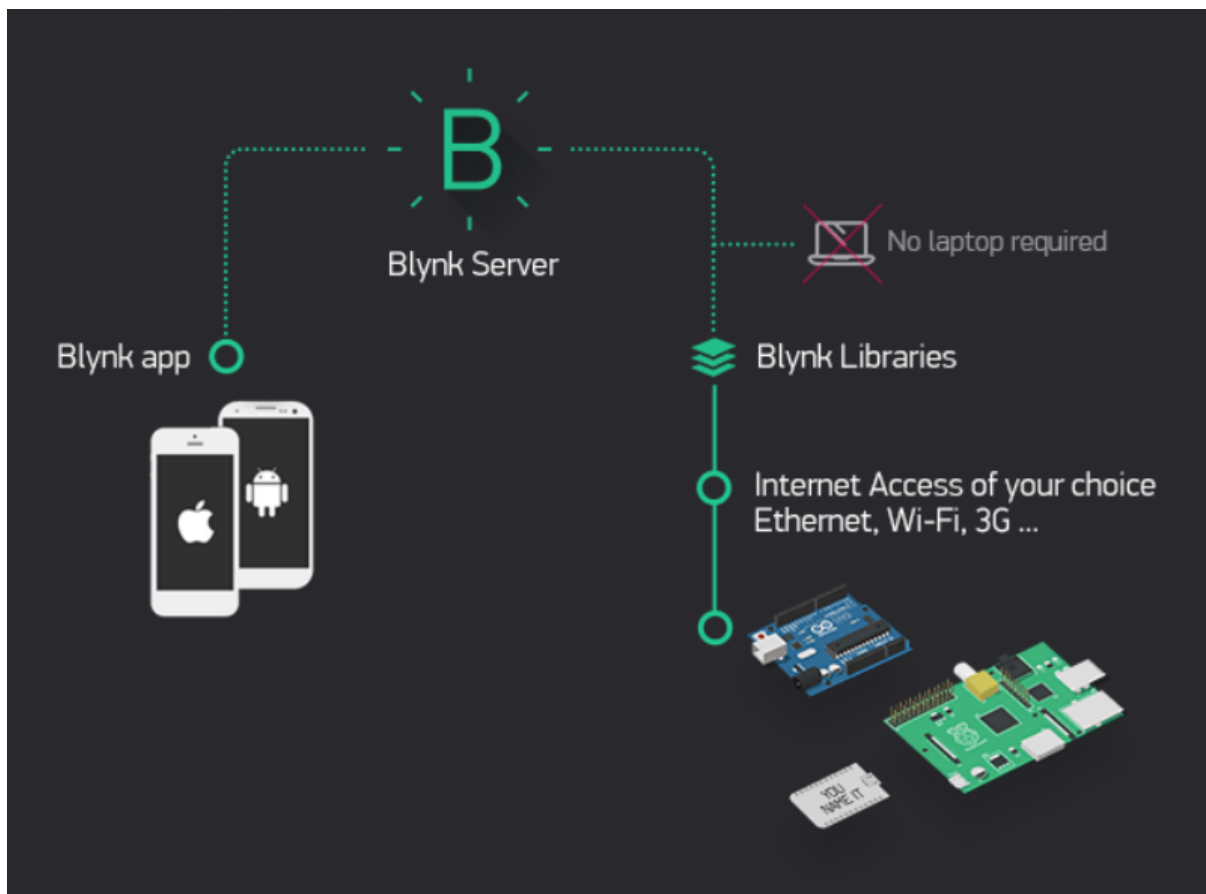
- Mở Boards Manager từ Tools > Board menu và tìm **esp8266** platform.
- Chọn phiên bản bạn cần từ cửa sổ Drop-down.
- Click nút **install**.

2.2.3. Ứng dụng Blynk

Blynk là một nền tảng có ứng dụng iOS, Android cho phép điều khiển Arduino, Raspberry Pi, ESP8266. Blynk được thiết kế cho các ứng dụng IoT, nó có thể điều khiển phần cứng từ xa, hiển thị dữ liệu cảm biến, lưu trữ dữ liệu,...

Blynk có 3 thành phần chính trong nền tảng:

- **Blynk App:** cho phép tạo các giao diện tuyệt vời cho các dự án của bạn bằng cách sử dụng các widget khác nhau.
- **Blynk Server:** truyền tải thông tin giữa Smarthome và phần cứng. Blynk Server có thể là 1 đám mây của Blynk hoặc có thể cài đặt trên máy cá nhân. Thậm chí có thể cài đặt trên Raspberry Pi.
- **Blynk Libraries:** cho tất cả các nền tảng phần cứng phổ biến, cho phép giao tiếp với máy chủ, xử lý các lệnh đến và đi.



Hình 2.10 Cách thức hoạt động của Blynk

2.3. Thư viện OpenCV

2.3.1. Giới thiệu về thư viện mã nguồn mở OpenCV

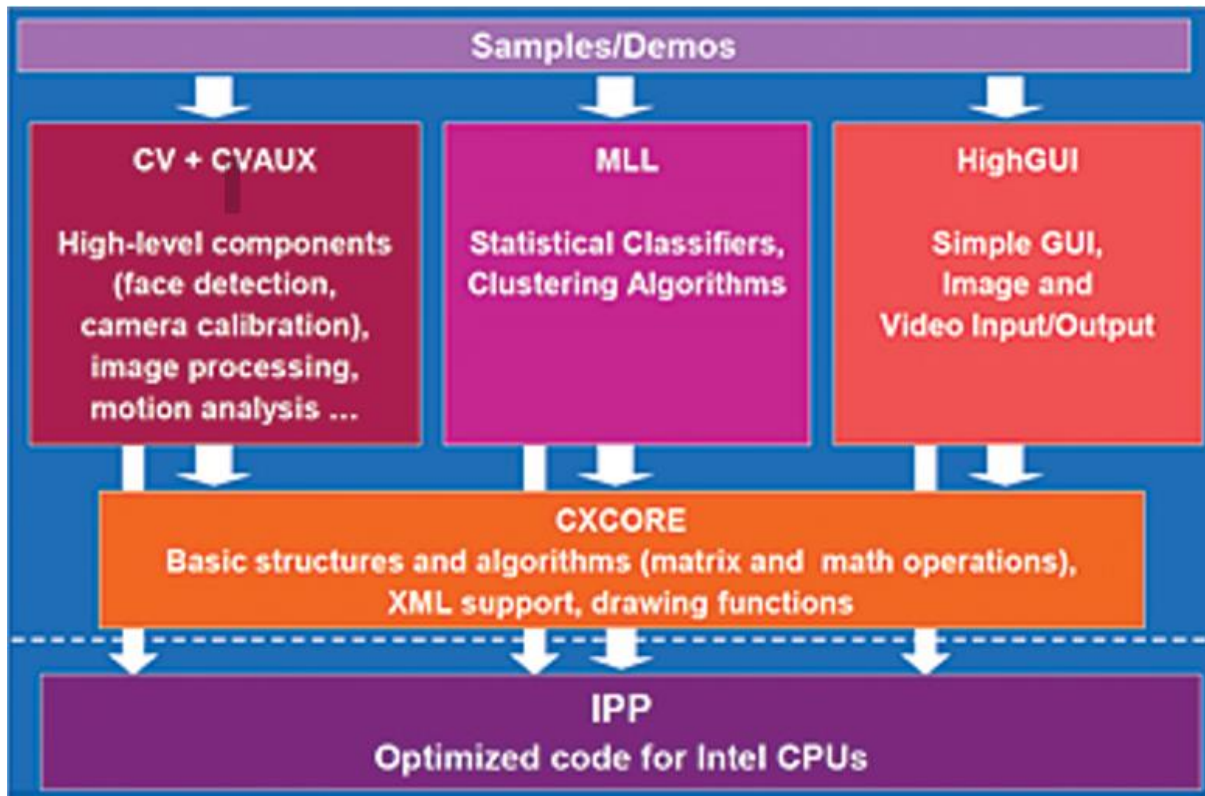
“OpenCV (Open Computer Vision library) do Intel phát triển, được giới thiệu năm 1999 và hoàn thiện thành phiên bản 1.0 năm 2006. Thư viện OpenCV - gồm khoảng 500 hàm – được viết bằng ngôn ngữ lập trình C và tương thích với các hệ điều hành Windows, Linux, Mac OS... đóng vai trò xác lập chuẩn giao tiếp, dữ liệu, thuật toán cho lĩnh vực CV và tạo điều kiện cho mọi người tham gia nghiên cứu và phát triển ứng dụng.

Trước OpenCV không có một công cụ chuẩn nào cho lĩnh vực xử lý ảnh. Các đoạn code đơn lẻ do các nhà nghiên cứu tự viết thường không thống nhất và không ổn định. Các bộ công cụ thương mại như Matlab, Simulink, Halcon, v.v... lại có giá cao chỉ thích hợp cho các công ty phát triển các ứng dụng lớn. Ngoài ra còn có các giải pháp kèm theo thiết bị phần cứng mà phần lớn là mã đóng và được thiết kế riêng cho từng thiết bị, rất khó khăn cho việc mở rộng ứng dụng.

OpenCV là công cụ hữu ích cho những người bước đầu làm quen với xử lý ảnh số vì các ưu điểm sau:

- OpenCV là công cụ chuyên dụng: Được Intel phát triển theo hướng tối ưu hóa cho các ứng dụng xử lý và phân tích ảnh, với cấu trúc dữ liệu hợp lý, thư viện tạo giao diện, truy xuất thiết bị phần cứng được tích hợp sẵn. OpenCV thích hợp để phát triển nhanh ứng dụng.
 - OpenCV là công cụ mã nguồn mở: Không chỉ là công cụ miễn phí (với BSD license), việc được xây dựng trên mã nguồn mở giúp OpenCV trở thành công cụ thích hợp cho nghiên cứu và phát triển, với khả năng thay đổi và mở rộng các mô hình, thuật toán.
 - OpenCV đã được sử dụng rộng rãi: Từ năm 1999 đến nay, OpenCV đã thu hút được một lượng lớn người dùng, trong đó có các công ty lớn như Microsoft, IBM, Sony, Siemens, Google và các nhóm nghiên cứu ở Stanford, MIT, CMU, Cambridge... Nhiều forum hỗ trợ và cộng đồng người dùng đã được thành lập, tạo nên kênh thông tin rộng lớn hữu ích cho việc tham khảo tra cứu.”
- [6]

2.3.2. Tổ chức thư viện OpenCV



Hình 2.11 Tổ chức thư viện OpenCV

“**CXCORE** chứa các định nghĩa kiểu dữ liệu cơ sở. Ví dụ, các cấu trúc dữ liệu cho ảnh, điểm và hình chữ nhật được định nghĩa trong *cxtypes.h*. CXCORE cũng chứa đại số tuyến tính và phương pháp thống kê, chức năng duy trì và điều khiển chuỗi. Một số ít, các chức năng đồ họa để vẽ trên ảnh cũng được đặt ở đây.

CV chứa các thuật toán về xử lý ảnh và định kích cỡ camera. Các chức năng hình họa máy tính cũng được đặt ở đây.

CVAUX được mô tả trong tài liệu của OpenCV như chứa các mã cũ và thử nghiệm. Tuy nhiên, các giao diện đơn cho sự nhận diện ảnh ở trong module này. Code sau này chúng được chuyên dụng cho nhận diện mặt và chúng được ứng dụng rộng rãi cho mục đích đó.

HIGHGUI và **CVCAM** được đặt trong cùng thư mục là “otherlibs”.

- **HIGHGUI** : chứa các giao diện vào ra cơ bản, nó cũng chứa các khả năng cửa sổ mở rộng và vào ra video.

- **CVCAM** : chứa các giao diện cho video truy cập qua DirectX trên nền Windows 32 bits.

Các chức năng của openCV tập trung vào thu thập ảnh, xử lý ảnh và các thuật toán phân tích dữ liệu ảnh, bao gồm:

- *Truy xuất ảnh và phim*: đọc ảnh số từ camera, từ file, ghi ảnh và phim
- *Cấu trúc dữ liệu ảnh số và các dữ liệu hỗ trợ cần thiết*: ma trận, vector, chuỗi, xâu và cây
- *Xử lý ảnh căn bản*: các bộ lọc có sẵn, tìm chi tiết cạnh, góc, chỉnh đổi màu, phóng to thu nhỏ, và hiệu chỉnh histograms
- *Xử lý cấu trúc*: tìm viền, nhận chuyển động, thay đổi trong không gian 3D, đổi chiều bản mẫu, xấp xỉ các đơn vị hình học cơ sở - mặt phẳng, đa giác, ellipse, đường thẳng...
- *Phân tích dữ liệu ảnh*: nhận dạng thực thể, theo dõi các chi tiết và phân tích chuyển động
- *Tạo giao diện đơn giản*: hiển thị ảnh, thao tác bàn phím, chuột, thanh trượt để chỉnh thông số (nếu cần thiết các bạn có thể tự tạo thêm các phím điều khiển thông qua thao tác chuột, hoặc tích hợp thêm các thư viện về giao diện như wxWidgets)
- *Chức năng vẽ, chú thích lên ảnh.*” [6]

2.4. Đặc trưng Haar-Like

2.4.1. Giới thiệu và cách tính giá trị đặc trưng Haar-like

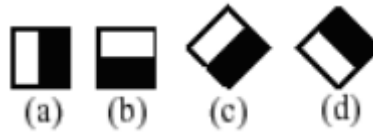
“Đặc trưng Haar-like là đặc trưng hình ảnh số được sử dụng trong nhận diện hình ảnh do Viola và Jones công bố, được sử dụng trong hệ thống nhận diện khuôn mặt thời gian thực đầu tiên. Đặc trưng Haar-like gồm 4 đặc trưng cơ bản để xác định khuôn mặt người. Mỗi đặc trưng Haar-like là sự kết hợp của hai hay ba hình chữ nhật “trắng” hay “đen” như trong hình sau:



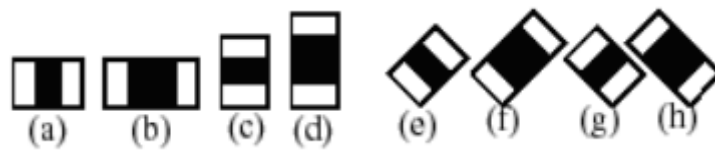
Hình 2.12 Bốn đặc trưng Haar-like cơ bản

Để sử dụng các đặc trưng này vào việc xác định khuôn mặt người, 4 đặc trưng Haar-like cơ bản được mở rộng ra, và được chia làm 3 tập đặc trưng như sau:

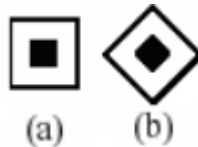
1. Đặc trưng cạnh (edge features):



2. Đặc trưng đường (line features):



3. Đặc trưng xung quanh tâm (center-surround features):



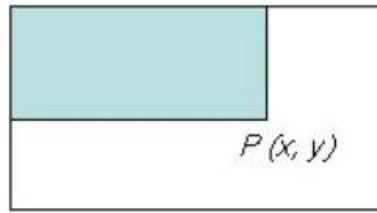
Hình 2.13 Các đặc trưng mở rộng của các đặc trưng Haar-like cơ sở

Dùng các đặc trưng trên, ta có thể tính được giá trị của đặc trưng Haar-like là sự chênh lệch giữa tổng của các pixel của các vùng đen và các vùng trắng như trong công thức sau:

$$f(x) = \text{Tổng}_{\text{vùng đen}}(\text{các mức xám của pixel}) - \text{Tổng}_{\text{vùng trắng}}(\text{các mức xám của pixel})$$

Sử dụng giá trị này, so sánh với các giá trị của các giá trị pixel thô, các đặc trưng Haar-like có thể tăng/giảm sự thay đổi in-class/out-of-class (bên trong hay bên ngoài lớp khuôn mặt người), do đó sẽ làm cho bộ phân loại dễ hơn.

Như vậy ta có thể thấy rằng, để tính các giá trị của đặc trưng Haar-like, ta phải tính tổng của các vùng pixel trên ảnh. Nhưng để tính toán các giá trị của các đặc trưng Haar-like cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, không đáp ứng được cho các ứng dụng đòi hỏi tính run-time. Do đó Viola và Jones đưa ra một khái niệm gọi là Integral Image, là một mảng 2 chiều với kích thước bằng với kích thước của ảnh cần tính các đặc trưng Haar-like, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ dựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.



$$P(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Hình 2.14 Cách tính Integral Image của ảnh

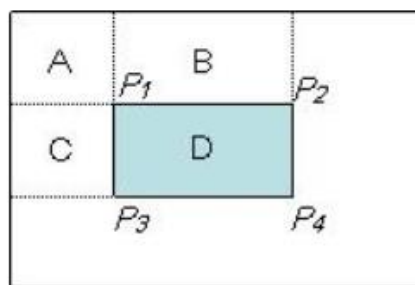
Sau khi đã tính được Integral Image, việc tính tổng các giá trị mức xám của một vùng bất kỳ nào đó trên ảnh thực hiện rất đơn giản theo cách sau:

Giả sử ta cần tính tổng các giá trị mức xám của vùng D như trong hình 2.15, ta có thể tính như sau:

$$D = A + B + C + D - (A+B) - (A+C) + A$$

Với $A + B + C + D$ chính là giá trị tại điểm P4 trên Integral Image, tương tự như vậy $A+B$ là giá trị tại điểm P2, $A+C$ là giá trị tại điểm P3, và A là giá trị tại điểm P1. Vậy ta có thể viết lại biểu thức tính D ở trên như sau:

$$D = \underbrace{(x_4, y_4)}_{A+B+C+D} - \underbrace{(x_2, y_2)}_{(A+B)} - \underbrace{(x_3, y_3)}_{(A+C)} + \underbrace{(x_1, y_1)}_A$$



Hình 2.15 Ví dụ cách tính nhanh các giá trị mức xám của vùng D trên ảnh” [7]

2.4.2. Các bước tạo đặc trưng cho một biên báo giao thông

Bước 1: Thu thập cơ sở dữ liệu hình ảnh.

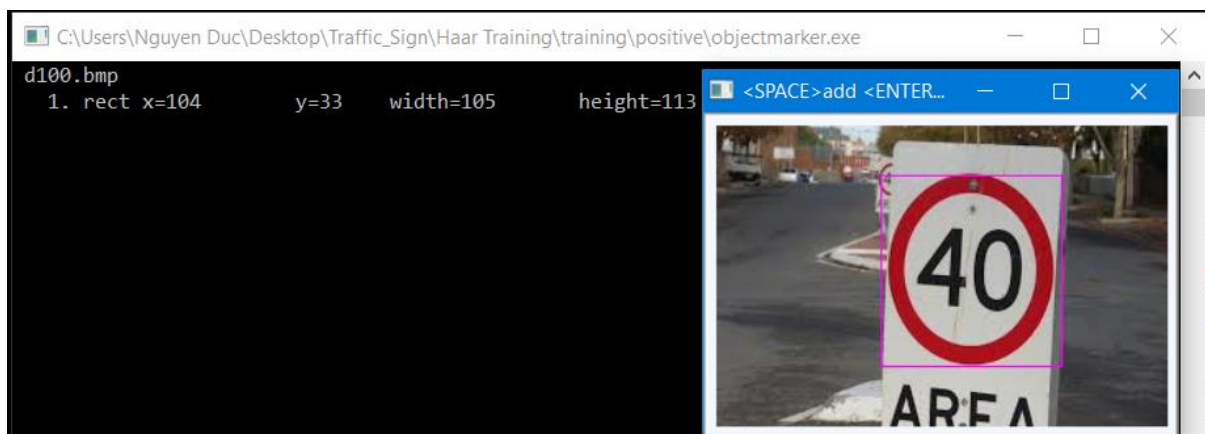
- Mỗi loại biển báo ta thu thập khoảng 200 ảnh tích cực với định dạng bmp (có biển báo cần huấn luyện trong ảnh) và khoảng 200 ảnh phủ định đã được xám hóa (không có biển báo cần huấn luyện trong ảnh).
- Số lượng ảnh thu thập để huấn luyện càng nhiều, độ chính xác càng cao.

Bước 2: Sắp xếp ảnh phủ định.

- Đưa tất cả ảnh phủ định vào trong thư mục ...\\training\\negative
- Chạy file *create_list.bat*
- Sau khi chạy xong, trong thư mục sẽ tạo ra file *bg.txt* chứa nội dung ảnh.

Bước 3: Khoanh đối tượng cần huấn luyện trong ảnh tích cực.

- Đưa tất cả ảnh tích cực vào thư mục ...\\training\\positive\\rawdata
- Chạy file *objectmaker.exe*
- Khoanh vào đối tượng cần huấn luyện.
- Sau khi khoanh sẽ xuất hiện vị trí và kích cỡ của khung hình chữ nhật, lặp lại cho tới khi khoanh hết tất cả ảnh, ta tạo được file *info.txt* được tạo ra chứa thông tin ảnh



Hình 2.16 Quá trình khoanh vùng biển báo giao thông để huấn luyện

Bước 4: Tạo vector cho ảnh tích cực.

Chạy file *samples_creation.bat* (chứa thông tin về đường dẫn ảnh tích cực và file vector, số lượng cũng như chiều cao, chiều rộng của ảnh tích cực) xuất ra vector của ảnh tích cực.

Bước 5: Huấn luyện Haar-Like.

- Chạy file *haartraining.bat* huấn luyện từ thông tin của ảnh phủ định và ảnh tích cực.

```

Parent node: 0

*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.243605
BACKGROUND PROCESSING TIME: 0.01
Precalculation time: 8.09
+-----+-----+-----+-----+-----+-----+
| N | %SMP | F | ST.THR | HR | FA | EXP. ERR |
+-----+-----+-----+-----+-----+-----+
| 1 | 100% | - | -0.915344 | 1.000000 | 1.000000 | 0.067500 |
+-----+-----+-----+-----+-----+-----+
| 2 | 100% | + | -1.761648 | 1.000000 | 1.000000 | 0.050000 |
+-----+-----+-----+-----+-----+-----+
| 3 | 100% | - | -1.040223 | 1.000000 | 0.325000 | 0.027500 |
+-----+-----+-----+-----+-----+-----+
Stage training time: 4.79
Number of used features: 3

Parent node: 0
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+-----+-----+
| 0 | 1 |
+-----+-----+

```

Hình 2.17 Quá trình huấn luyện biểu báo giao thông

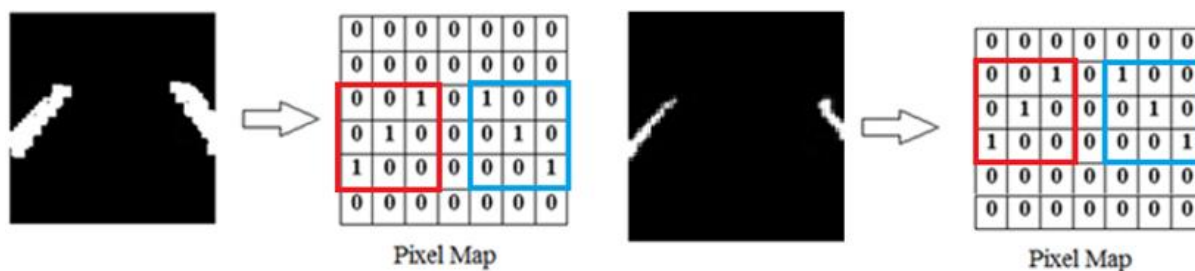
- Sau khi huấn luyện xong, ta sẽ được các thư mục chứa các file thống kê trong ...*\training\cascades*.

Bước 6: Tạo file XML

- Copy các thư mục các file thống kê vào thư mục ...*\cascade2xml\data*
- Chạy file *convert.bat* tại thư mục ...*\cascade2xml*
- Sau khi hoàn tất, ta có file .xml chứa đặc trưng của đối tượng đã huấn luyện.

2.5. Mạng CNN (Convolutional Neural Network)

CNN là sự kết hợp giữa mạng nơron và tích chập để nhận diện và phân loại ảnh kể cả khi chúng bị co, tịnh tiến hoặc biến dạng.

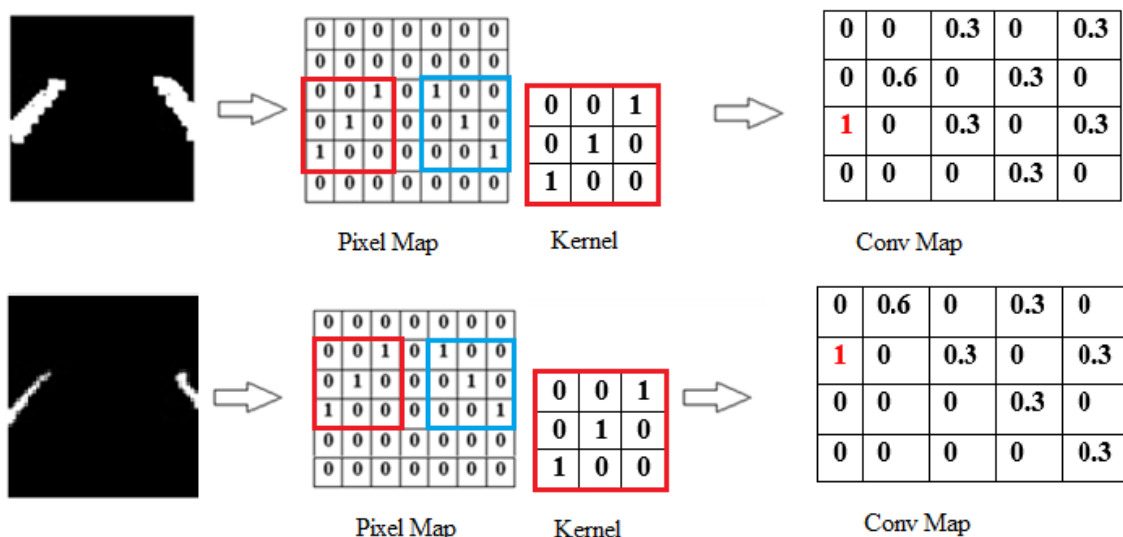


Hình 2.18. Pixel map của hai ảnh khác nhau

Như ta nhận thấy, ở 2 hình bên trên đều là đường thẳng, nhưng sau khi phân tích ra điểm ảnh thì ta thu được 2 Pixel Map như trên. Nếu cơ chế hoạt động của máy tính giống với bộ não con người thì nó sẽ dễ dàng nhận ra đây là 2 ảnh gần như giống nhau. Tuy nhiên, vị trí của các phần tử màu **đỏ** và **xanh** có thay đổi, và đối với máy tính, 2 Pixel Map bên trên hoàn toàn khác nhau \Rightarrow 2 ảnh khác nhau.

Để khắc phục điều này, ta cần sử dụng mạng nơron tích chập (CNN). Trong CNN, các đặc trưng có tính ảnh hưởng quan trọng được so sánh với nhau. Cụ thể hơn ở trong 2 hình trên đó là phần được khoanh màu **đỏ** và **xanh**. Nếu ở hình 1 mạng CNN tìm ra được 2 đặc trưng và tương tự cũng tìm được 2 đặc trưng như thế ở hình 2 thì nó sẽ cho rằng 2 hình đó giống nhau.

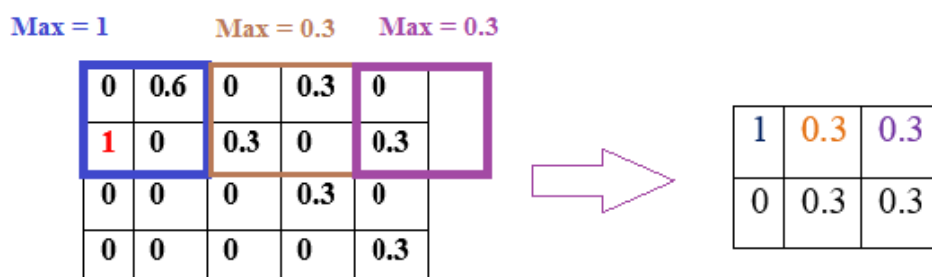
Tuy nhiên, đối với 1 ảnh hoàn toàn mới, thì CNN không biết được chính xác vị trí của các đặc trưng nên nó sẽ dịch chuyển các đặc trưng khắp ảnh để tìm ra chỗ ăn khớp. Thuật toán để tính toán sự trùng khớp của các đặc trưng được gọi là *Tích Chập*. Cụ thể hơn là đem nhân ảnh mới cho đặc trưng để thu được 1 ma trận mới.



Hình 2.19 Tính tích chập của một ma trận

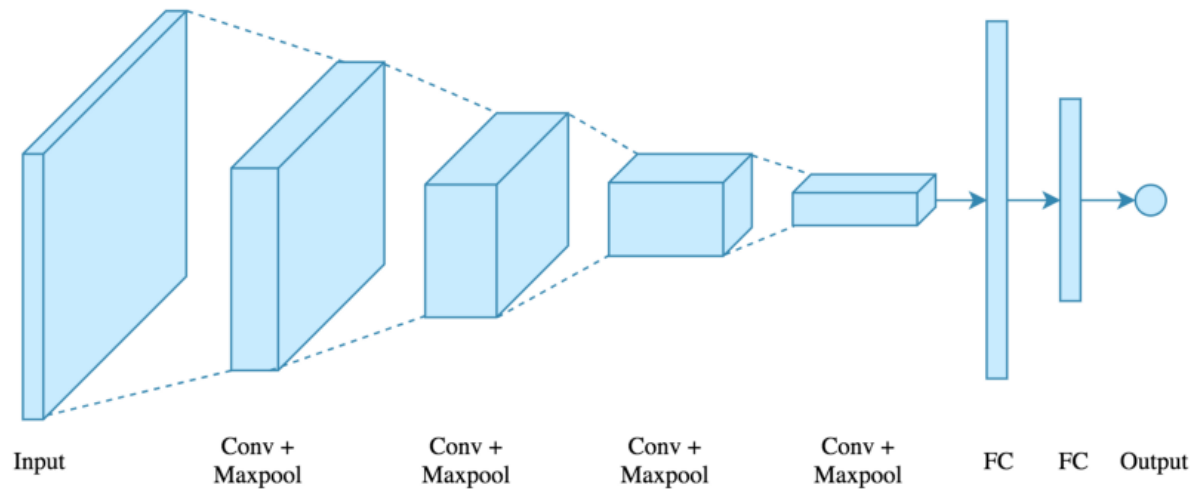
Như vậy là ta đã xác định được vị trí của đặc trưng trong ảnh mới, đó cũng chính là vị trí của điểm **1**. Trên thực tế khó có thể ra được chính xác điểm có giá trị là **1** đấy, nên ta có thể lấy các giá trị 0.8, 0.9 ... là các giá trị có độ tin cậy đủ cao. Tiếp đến là ta sẽ lấy các đặc trưng màu **xanh** để thực hiện tích chập, cách làm tương tự bên trên.

Đối với dữ liệu dạng ảnh có độ phân giải lớn, sau khi ta sử dụng tích chập mà ma trận mới thu được vẫn có kích thước tương đối lớn ta có thể làm chúng co lại trong khi vẫn giữ nguyên các thông tin quan trọng bằng phương pháp pooling. Tức là đem một cửa sổ hình vuông di chuyển dọc theo ma trận. Trong cửa sổ hình vuông đó, các giá trị của ma trận sẽ được đồng hóa thành 1 giá trị lớn nhất.



Hình 2.20 Pooling một ma trận

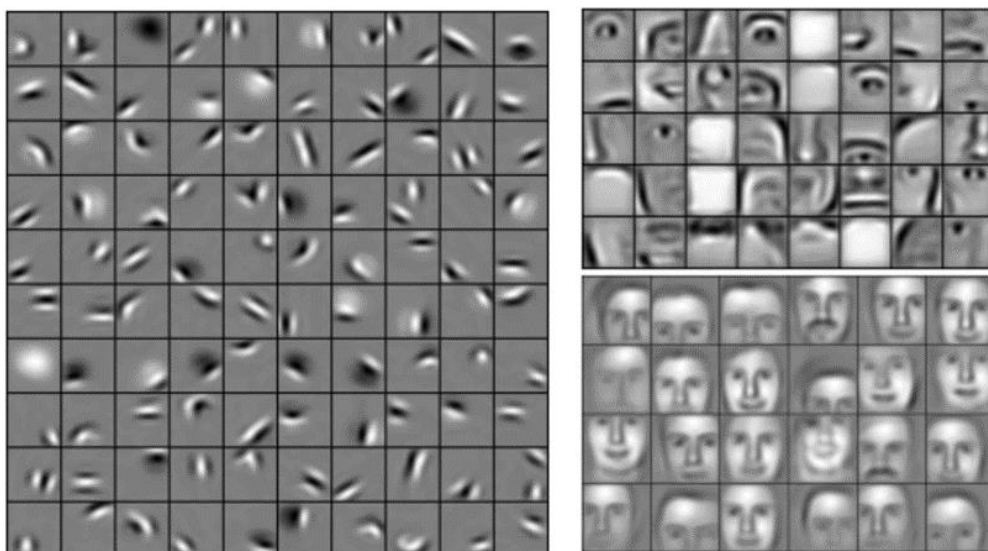
Nhìn chung, 1 mạng CNN tổng thể có dạng như sau :



Hình 2.21 Dạng tổng thể của một mạng CNN

Mỗi một tầng như hình trên ta gọi là một layer.

Đối với mỗi một đặc trưng, khi đi qua một layer ta sẽ có một ma trận ngõ ra có số chiều bằng với ảnh ngõ vào. Do đó ngõ ra của mỗi layer sẽ là tập hợp tất cả các ma trận thu được. Các layer đầu tiên, hay còn gọi là lower layer, các đặc trưng thường đơn giản như là đường cong, điểm sáng hay các hình dạng cơ bản Các layer càng về sau thì đặc trưng của chúng càng phức tạp và chi tiết hơn.



Hình 2.22 Các layer từ Lower đến Higher

Như vậy, khi ta muốn dùng mạng CNN để dự đoán, thì ta sẽ lấy ngõ ra của higher layer cho đi qua lớp fully connected. Tại đây, ma trận 2D hay 3D sẽ được chuyển thành 1D và tất cả các giá trị này sẽ được xử lý giống nhau. Mỗi một giá trị biểu diễn cho một ngõ ra tương ứng với một độ tin cậy nhất định. Cụ thể trong đề tài này, ngõ ra có dạng : [a ,b ,c, d, e].

Với :

- a độ tin cậy đối với dữ đoán ngõ ra là góc queo -60 độ
- b độ tin cậy đối với dữ đoán ngõ ra là góc queo -30 độ
- c độ tin cậy đối với dữ đoán ngõ ra là góc queo 0 độ
- d độ tin cậy đối với dữ đoán ngõ ra là góc queo 30 độ
- f độ tin cậy đối với dữ đoán ngõ ra là góc queo 60 độ

Ví dụ : ngõ ra = [0.3, 0.4, 0.35, 0.8, 0.5] thì mạng đã dự đoán được ngõ ra tương ứng đối với ảnh nhận vào sẽ là góc queo 30 độ vì độ tin cậy của nó là lớn nhất.

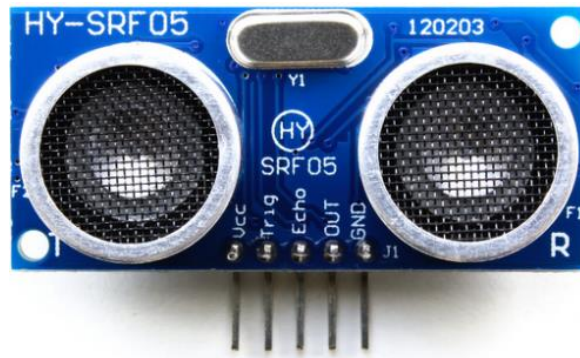
2.6. Các phần cứng phụ

2.6.1. Cảm biến siêu âm SRF05

Cảm biến siêu âm SRF05 là một thiết bị đo khoảng cách sử dụng nguyên tắc sóng siêu âm gồm 2 loa (thu và phát), 5 chân để kết nối với Arduino, có tầm hoạt động từ 0 đến 400cm.

Cơ chế dùng sóng âm để định vị gồm có 3 bước:

- Thiết bị phát ra sóng âm.
- Sóng âm này va chạm với môi trường xung quanh và phản xạ lại.
- Dựa vào thời gian phát và thu, khoảng cách giữa thiết bị và môi trường xung quanh được tính ra.



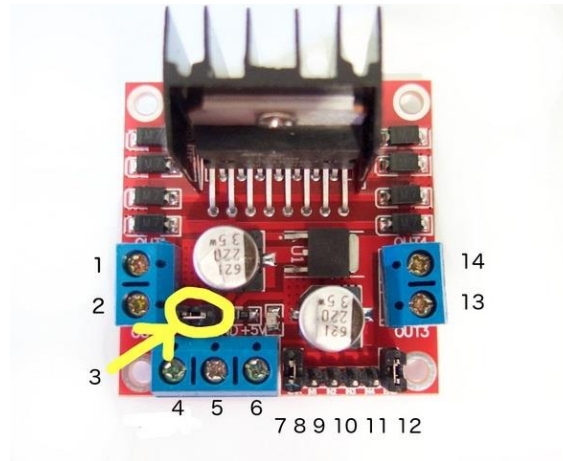
Hình 2.23 Sơ đồ chân của cảm biến siêu âm SRF05

Chức năng từng chân của cảm biến siêu âm:

- **Vcc:** cấp nguồn cho cảm biến.
- **Trigger:** kích hoạt quá trình phát sóng âm. Quá trình kích hoạt khi một chu kì điện cao / thấp diễn ra. (LOW-HIGH-LOW)
- **Echo:** bình thường sẽ ở trạng thái 0V, được kích hoạt lên 5V ngay khi có tín hiệu trả về, sau đó trở về 0V.
- **Gnd:** nối với cực âm của mạch
- **OUT:** không sử dụng

2.6.2. Mạch điều khiển động cơ L298N

Mạch điều khiển động cơ L298N được tích hợp 2 mạch cầu H, có khả năng điều khiển 2 động cơ DC và 1 động cơ bước. Dòng ra tối đa 2A và nguồn cấp có thể từ 5 đến 35V. Động cơ A được điều khiển bằng cách cấp các mức điện áp 1 và 0 vào 2 chân IN1 IN2 . Tương tự động B được điều khiển bằng cách cấp mức 1, 0 vào IN3 IN4. Đảo các mức 1 , 0 này sẽ làm động cơ đảo chiều và có thể điều khiển tốc độ động cơ bằng cách cấp xung trực tiếp vào các chân IN. Chân Enable còn có tác dụng “phanh” gấp động cơ. Khi đưa chân Enable tương ứng với động cơ xuống mức 0 .



Hình 2.24 Sơ đồ chân Module điều khiển động cơ L298N

Chức năng từng chân của mạch điều khiển động cơ L298N:

1. DC motor 1 "+" hoặc stepper motor A+.
2. DC motor 1 "-" hoặc stepper motor A-.
3. 12V jumper - tháo jumper qua nếu sử dụng nguồn trên 12V. Jumper này dùng để cấp nguồn cho IC ổn áp tạo ra nguồn 5V nếu nguồn trên 12V sẽ làm cháy IC nguồn.
4. Cắm dây nguồn cung cấp điện áp cho motor vào đây từ 6V đến 35V.
5. Cắm chân GND của nguồn vào đây.
6. Ngõ ra nguồn 5V, nếu jumper đầu vào không rút ra.
7. Chân Enable của Motor 1, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước.
8. IN1.
9. IN2.
10. IN3.
11. IN4.
12. Chân Enable của Motor 2, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước
13. DC motor 2 "+" hoặc stepper motor B+.
14. DC motor 2 "-" hoặc stepper motor B-.

2.6.3. Camera Raspberry Pi V1 5MP

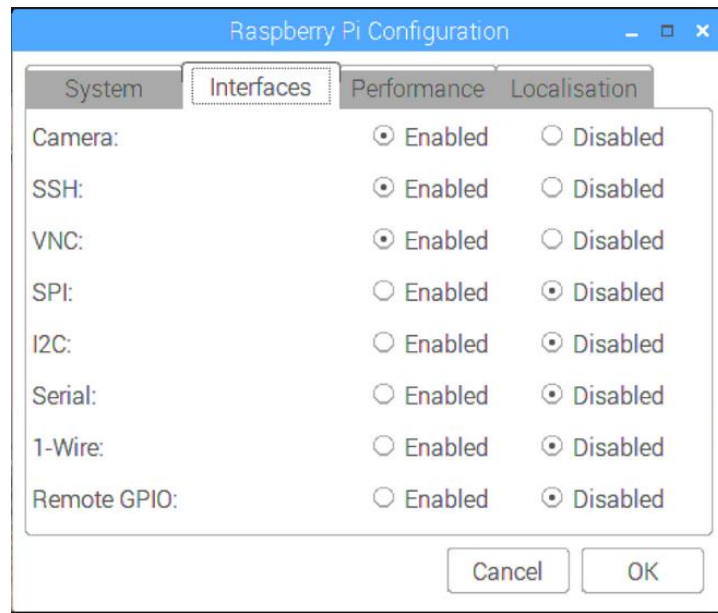
Camera Raspberry Pi V1 5MP là Version đầu tiên của module camera cho Raspberry Pi với cảm biến OV5647 độ phân giải 5MP, sử dụng tương thích với tất cả dòng Raspberry Pi từ trước đến nay, chất lượng hình ảnh tốt, độ phân giải cao và có khả năng quay phim ở chất lượng HD.

Thông số kỹ thuật:

- Module Camera V1 cho Raspberry Pi.
- Cảm biến: OV5647
- Độ phân giải: 5MP
- Độ phân giải hình: 2592x1944 pixel.
- Quay phim HD 1080p30, 720p60, VGA 640x480p60.
- Lens: Fixed Focus.
- Conector: Ribon conector.
- Kích thước: 25x24x9 mm



Hình 2.25 Ảnh thực tế Pi Camera



Hình 2.26 Kích hoạt (enable) cho Camera kết nối với Raspberry Pi 3

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

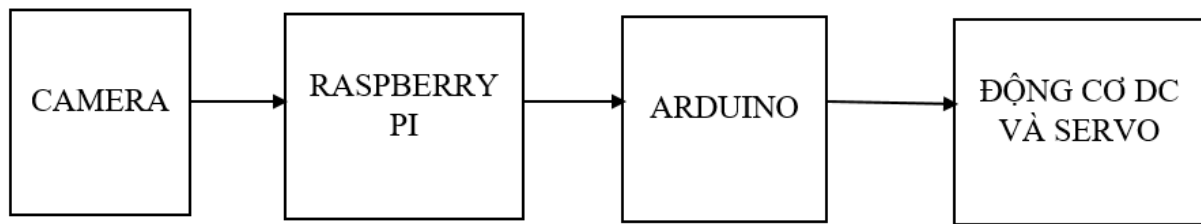
3.1. Yêu cầu thiết kế

Yêu cầu: Phần cứng đề cao tính chính xác, nhỏ gọn, thời gian đáp ứng nhanh, tiết kiệm năng lượng, dễ sử dụng và giá cả phù hợp. Kit sử dụng phải đủ mạnh để xử lý hình ảnh và video mà không bị quá tải.

Dưới cấp độ sinh viên, rất nhiều bo mạch có thể sử dụng trong ứng dụng lập trình. Nhóm quyết định chọn Raspberry Pi 3 kết hợp với Arduino với những ưu điểm sau đây:

- Raspberry Pi 3 có thể làm tốt các công việc liên quan đến hình ảnh, video,...
- Raspberry Pi 3 dùng bộ nguồn 5V 700mA trong khi đó mạch Intel Galileo lại cần đến bộ nguồn 5V 2000mA (2A), dẫn đến tiêu hao nhiều năng lượng.
- Raspberry Pi 3 có bộ nhớ RAM 1GB khắc phục tình trạng quá tải khi chạy chương trình.
- Arduino thuận tiện trong việc điều khiển động cơ, đồng thời giảm sự quá tải cho Raspberry Pi 3 khi phải làm nhiều tác vụ cùng một lúc.
- Arduino có thể kết hợp với module Ethernet ESP8266 để truyền tín hiệu và điều khiển từ xa.

3.2. Sơ đồ khối tổng quát



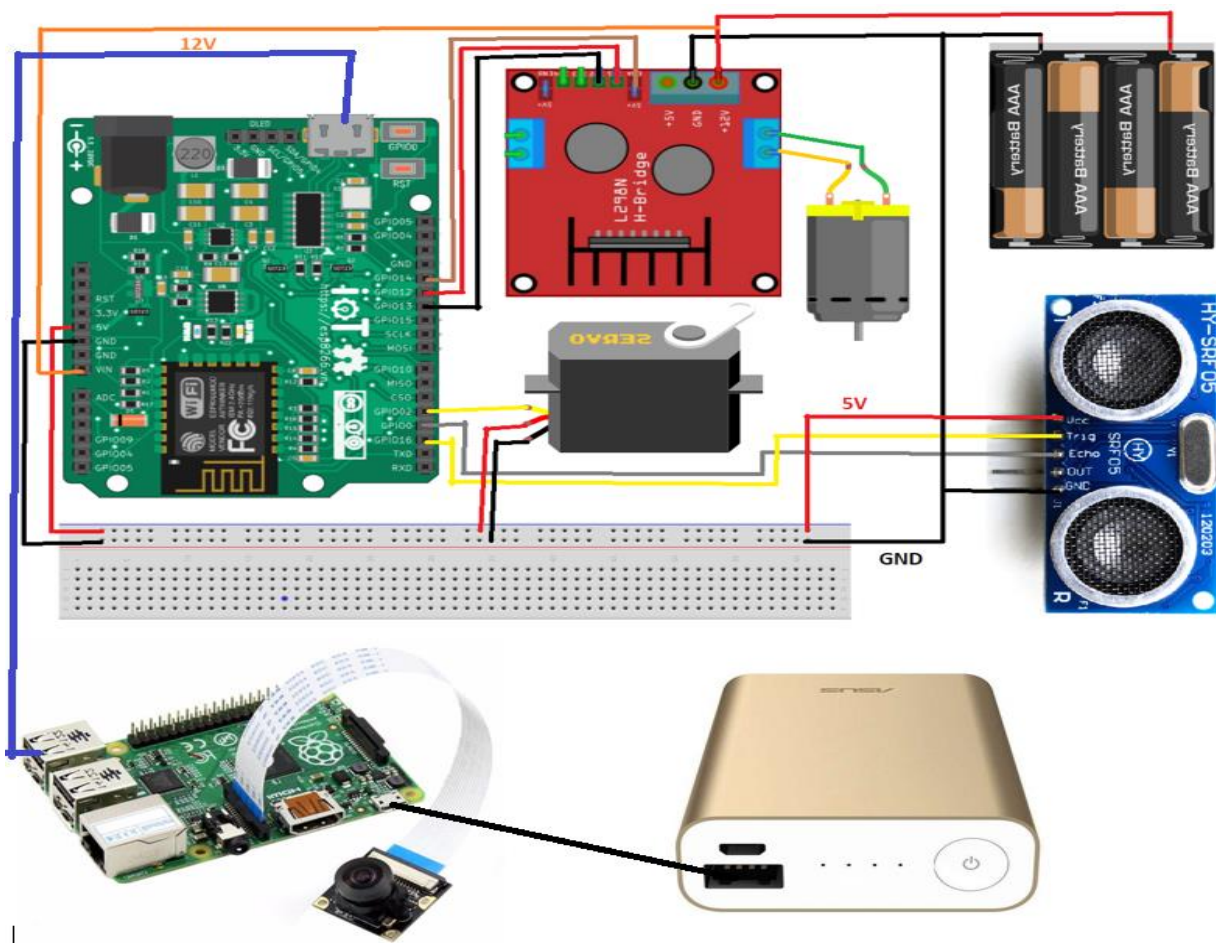
Khối CAMERA: thu hình ảnh truyền về kit Raspberry Pi 3 dưới dạng từng frame ảnh.

Khởi RASPBERRY PI: nhận hình ảnh từ camera, tiến hành xử lý ảnh xác định làn đường và nhận diện biển báo, truyền tín hiệu qua Arduino.

Khởi ARDUINO: thu nhận tín hiệu từ Raspberry Pi 3 để thực hiện điều khiển động cơ xe.

Khối ĐỘNG CƠ DC VÀ SERVO: thực hiện tín hiệu lái theo yêu cầu từ Arduino.

3.3. Sơ đồ khối chi tiết



Nguồn: pin Lipo 3 cell cấp nguồn 12V cho L298N và Arduino, pin dự phòng cấp nguồn 5V 2A cho Raspberry Pi 3.

Raspberry Pi 3 kết hợp Pi Camera: thu hình ảnh trực tiếp, xử lý ảnh.

Arduino: điều khiển động cơ DC, Servo, cảm biến siêu âm, lấy tín hiệu thông qua Serial nối với cổng USB của Raspberry Pi 3.

Mạch điều khiển động cơ L298N: điều khiển tốc độ động cơ DC GA37Y370 (2 bánh sau).

Động cơ Servo MG996R: điều khiển góc quay của xe từ -90 đến 90 độ (2 bánh trước).

Cảm biến siêu âm SRF05: xác định khoảng cách tới vật cản phía trước (từ 0 đến 400cm).

4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

4.1. Yêu cầu đặt ra cho phần mềm

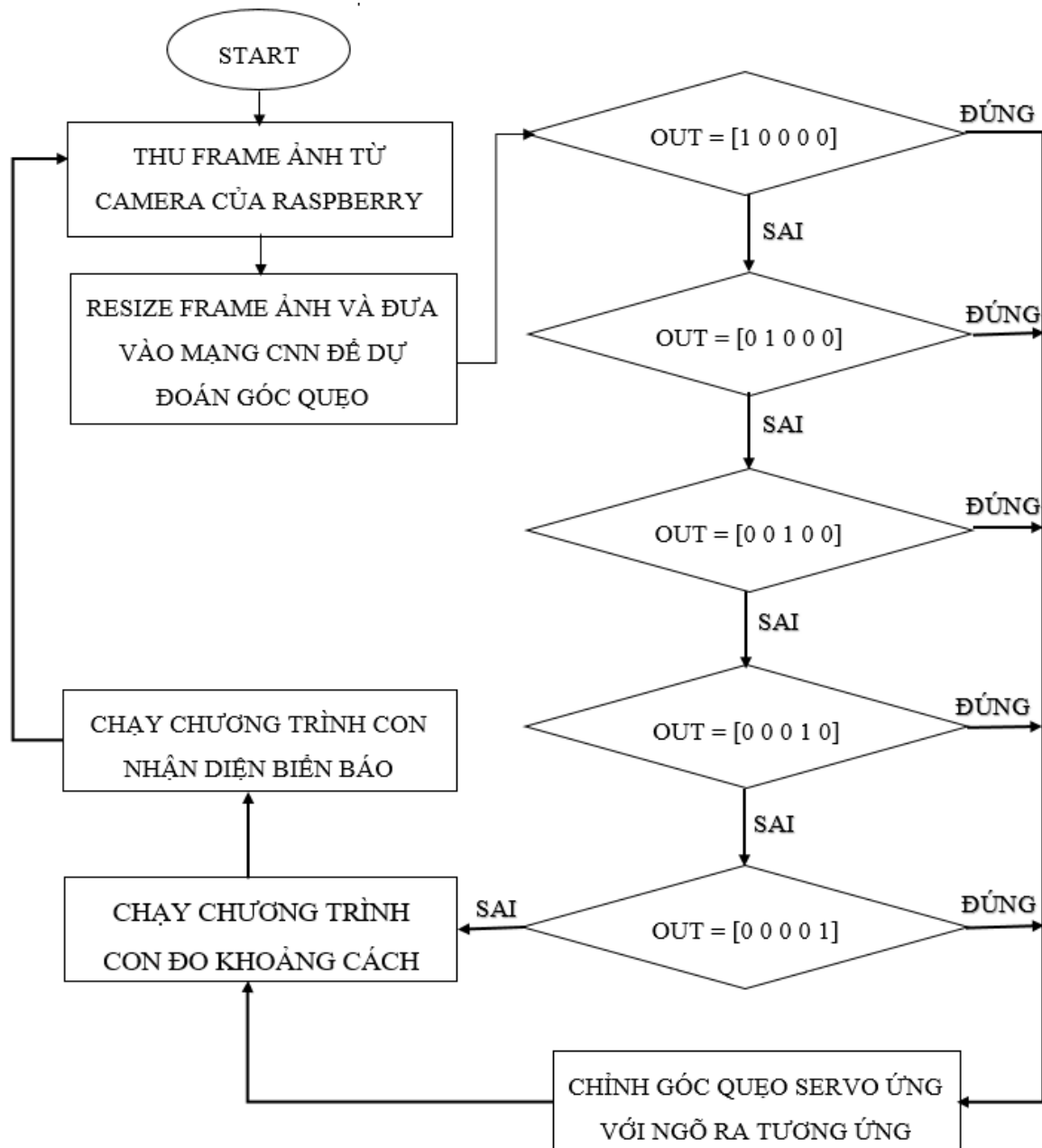
Phần mềm được viết theo ngôn ngữ lập trình Python và Arduino (được thiết kế dựa trên C) để thuận tiện cho việc sử dụng và tương thích với kit Raspberry Pi 3 và Arduino Iot Wifi Uno.

Đề cao tính chính xác và đáp ứng nhanh thời gian thực.

Giải thuật tối ưu để có thể dễ phát hiện và sửa lỗi, đồng thời cũng tiết kiệm năng lượng để xe có thể vận hành trong thời gian dài.

Sử dụng 2 module chính là điều khiển bằng tay để thu thập dữ liệu huấn luyện và chạy tự động với cơ sở dữ liệu đã được huấn luyện.

4.2. Giải thuật chương trình chính



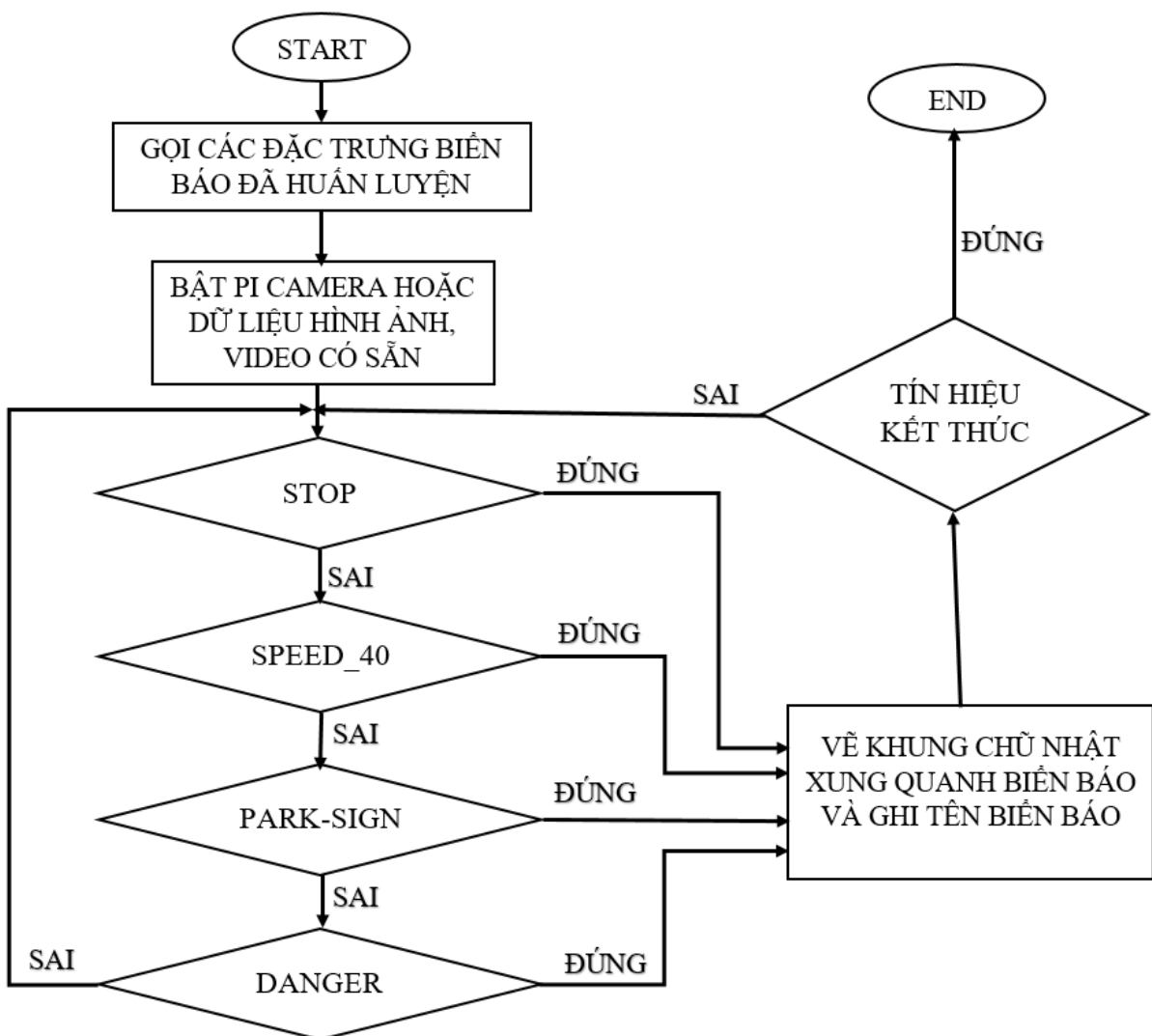
Chú thích:

- [1 0 0 0 0]: ma trận góc quẹo -60 độ (quẹo trái nhiều)
- [0 1 0 0 0]: ma trận góc quẹo -30 độ (quẹo trái tương đối)
- [0 0 1 0 0]: ma trận góc quẹo 0 độ (đi thẳng)
- [0 0 0 1 0]: ma trận góc quẹo 30 độ (quẹo phải tương đối)
- [0 0 0 0 1]: ma trận góc quẹo 60 độ (quẹo phải nhiều)

Giải thích giải thuật: Khi bắt đầu chương trình, ta khởi tạo các thông số cho Pi Camera (resolution, fps,...). Camera thu ảnh trực tiếp, truyền từng frame ảnh về cho Raspberry. Frame ảnh thu được sẽ được xử lý ảnh để dự đoán góc quẹo (dùng mạng CNN) và nhận diện biển báo (dùng đặc trưng Haar Like), đồng thời lúc đó ta cũng đo khoảng cách (dùng cảm biến siêu âm SRF05) để xe có thể tự động tránh vật cản.

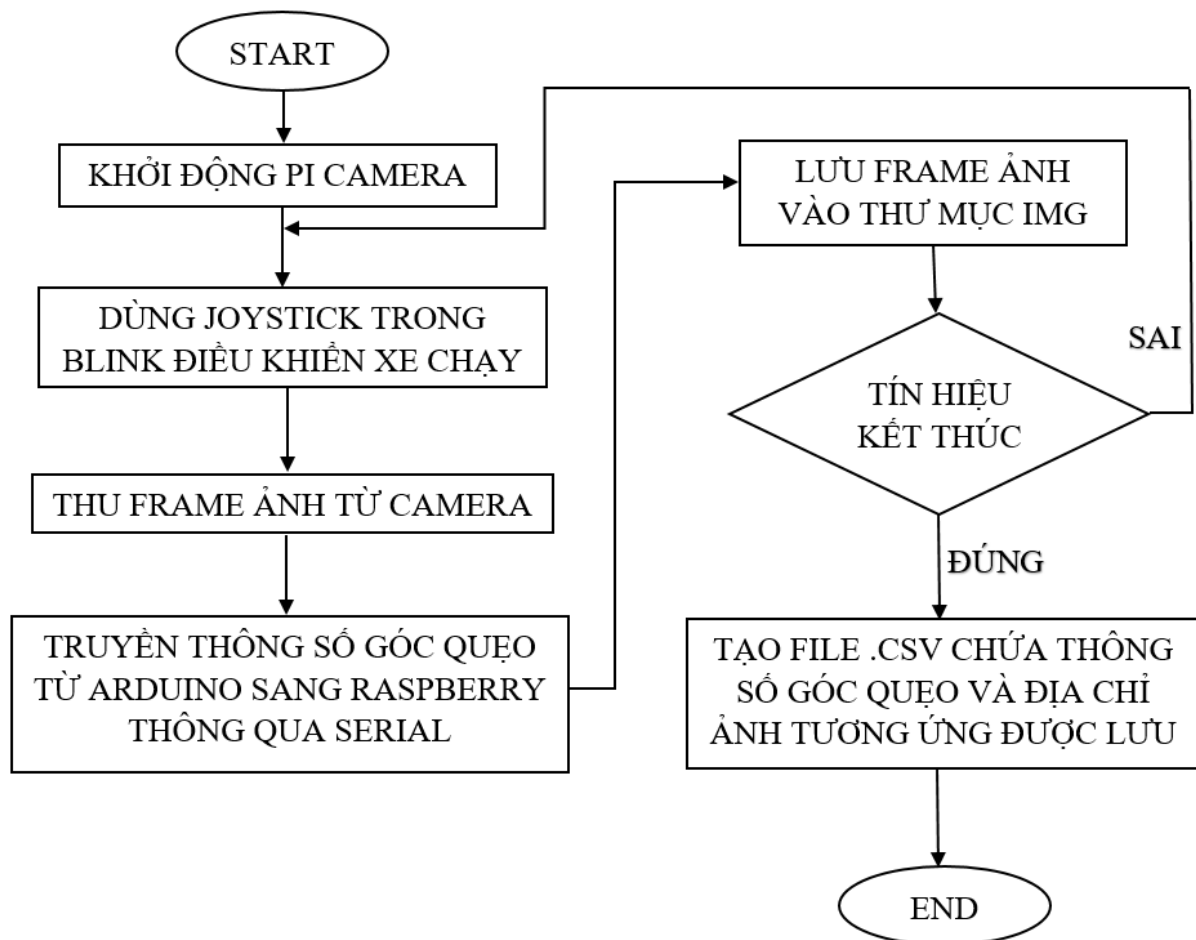
4.3. Giải thuật các chương trình con

4.3.1. Giải thuật nhận diện biển báo giao thông



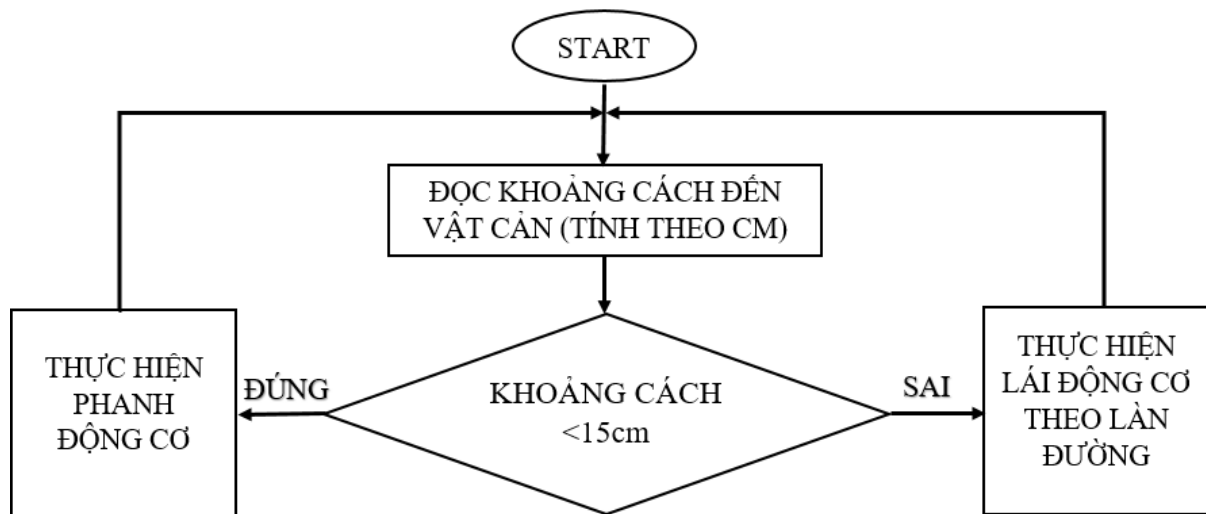
Giải thích giải thuật: Sau khi huấn luyện dữ liệu biển báo giao thông, ta có các file .xml chứa đặc trưng của các biển báo, ta gọi các đặc trưng này ra trong việc nhận diện biển báo. Khi phát hiện một loại biển báo, chương trình sẽ khoanh hình chữ nhật và xác định tên biển báo đó.

4.3.2. Giải thuật điều khiển xe bằng tay thu thập dữ liệu



Giải thích giải thuật: Ta sử dụng ứng dụng Blynk để điều khiển bằng tay cho xe chạy thông qua ESP8266. Mỗi lần sử dụng Joystick trên Blynk để điều khiển hướng cho xe chạy, frame ảnh được lưu lại trong một thư mục, đồng thời góc quẹo tương ứng cũng được lưu vào bộ nhớ tạm. Khi kết thúc quá trình điều khiển xe bằng tay, ta tạo ra một file .csv chứa nội dung góc quẹo với địa chỉ của frame ảnh tương ứng với góc đó. Từ đó, ta sử dụng dữ liệu này (gồm hình ảnh và file thông số) để tiến hành huấn luyện.

4.3.3. Giải thuật đo khoảng cách



Giải thích giải thuật: Khi gặp một vật thể trước mặt cách 15cm xe sẽ tự động phanh ngừng động cơ. Ngược lại, xe vẫn tiếp tục chạy theo làn đường.

5. KẾT QUẢ THỰC HIỆN

5.1. Cách thức đo đạc, thử nghiệm

5.1.1. Các phần cứng và phần mềm được sử dụng trong thực thi chương trình

- Phần cứng: Raspberry Pi 3, Pi Camera, Arduino IoT Wifi, Khung xe (gồm 1 động cơ DC, 1 động cơ Servo, 1 cầu vi sai,...), mạch điều khiển động cơ L298N, cảm biến siêu âm SRF05.
- Phần mềm: Chương trình lập trình: Python IDLE, Arduino IDE, Blynk.

Ngôn ngữ lập trình: Python, Arduino (được thiết kế dựa trên C)

5.1.2. Các bước tiến hành

- Điều khiển bằng tay:
 - Thu hình ảnh trực tiếp từ Pi Camera.

- Dùng phần mềm Blynk, điều khiển bằng tay thông qua Smartphone để lái xe chạy theo làn đường.
- Truyền dữ liệu góc quẹo từ Arduino sang Raspberry thông qua kết nối Serial của cổng USB.
- Ghi dữ liệu thu được vào một file định dạng .csv chứa thông số góc quẹo tương ứng với từng khung ảnh.
- Dùng hình ảnh và file dữ liệu vừa tạo được để huấn luyện.

➤ Điều khiển tự động:

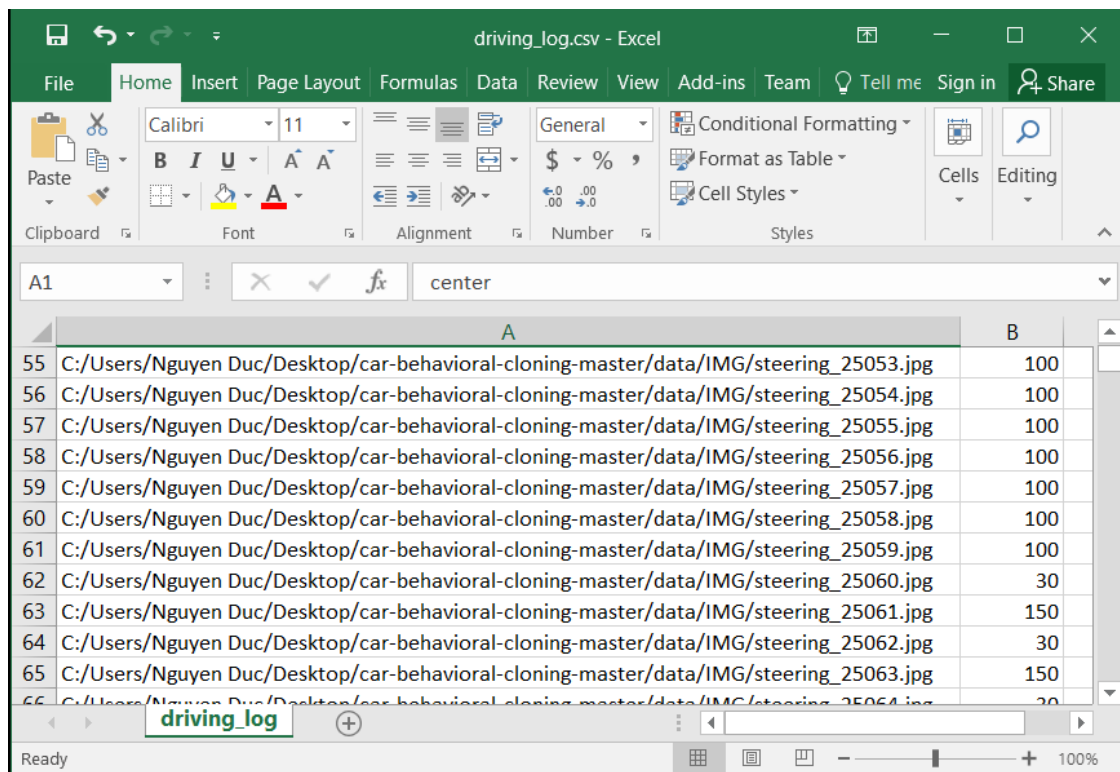
- Thu hình ảnh trực tiếp từ Pi Camera.
- Xử lý từng khung ảnh thu được trên Raspberry Pi 3 để xác định được làn đường, từ đó đưa ra thông số góc quẹo (dựa trên dữ liệu đã được huấn luyện) và nhận diện một số loại biển báo giao thông.
- Truyền tín hiệu đã xử lý ảnh từ Raspberry Pi 3 sang Arduino thông qua kết nối Serial của cổng USB.
- Arduino sẽ đưa ra tín hiệu lái thực hiện điều khiển động cơ xe.



Hình 5.1 Project để điều khiển xe bằng tay (góc quẹo và tốc độ) trên Blynk

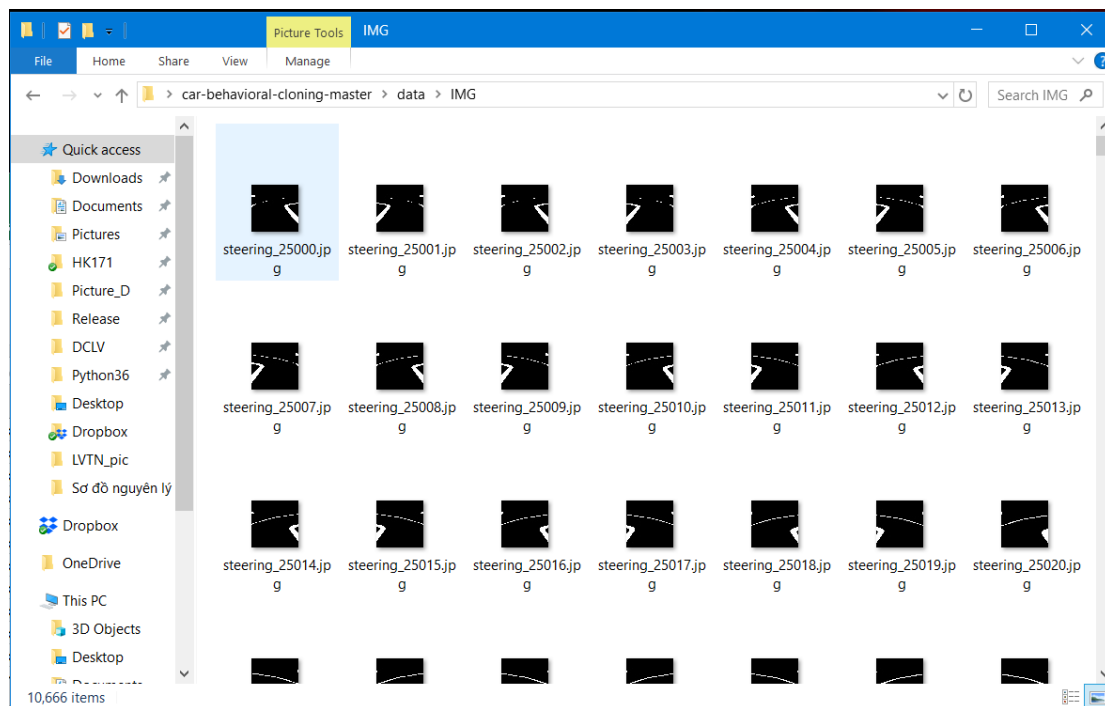
5.2. Dữ liệu huấn luyện

5.2.1. Làn đường



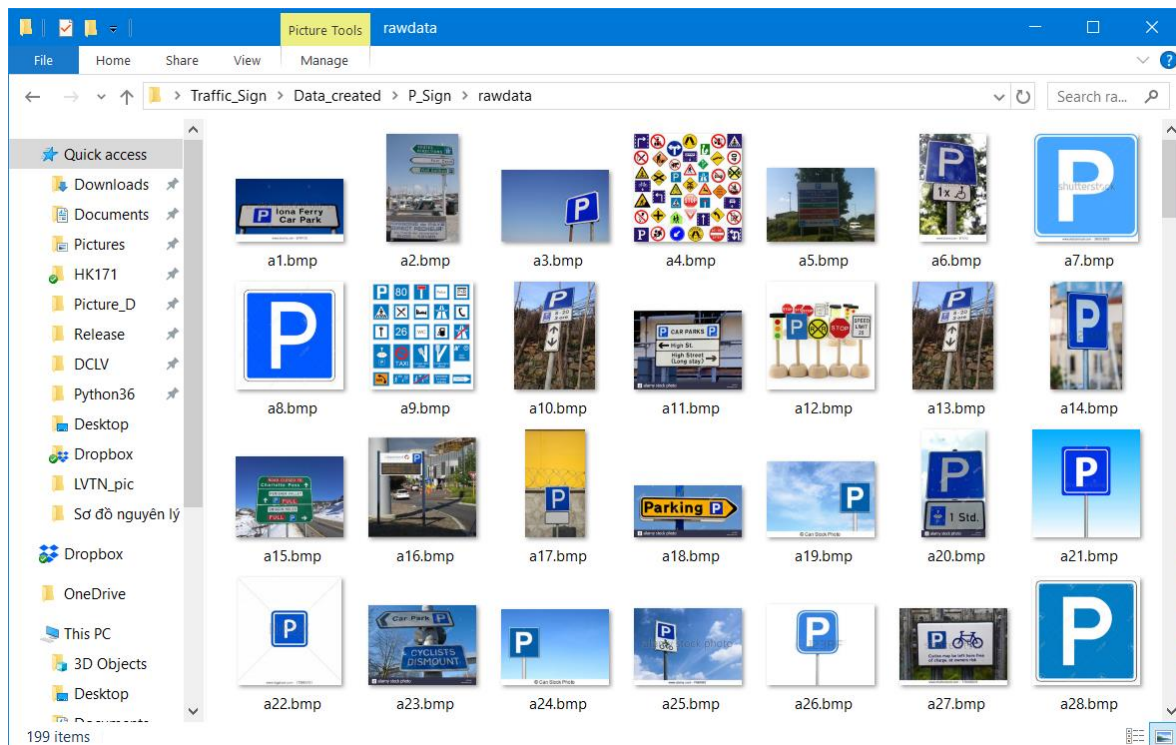
| | A | B |
|----|---|-----|
| 55 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25053.jpg | 100 |
| 56 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25054.jpg | 100 |
| 57 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25055.jpg | 100 |
| 58 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25056.jpg | 100 |
| 59 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25057.jpg | 100 |
| 60 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25058.jpg | 100 |
| 61 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25059.jpg | 100 |
| 62 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25060.jpg | 30 |
| 63 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25061.jpg | 150 |
| 64 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25062.jpg | 30 |
| 65 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25063.jpg | 150 |
| 66 | C:/Users/Nguyen Duc/Desktop/car-behavioral-cloning-master/data/IMG/steering_25064.jpg | 30 |

Hình 5.2 File .csv chứa thông số góc quay ứng với khung ảnh tương ứng

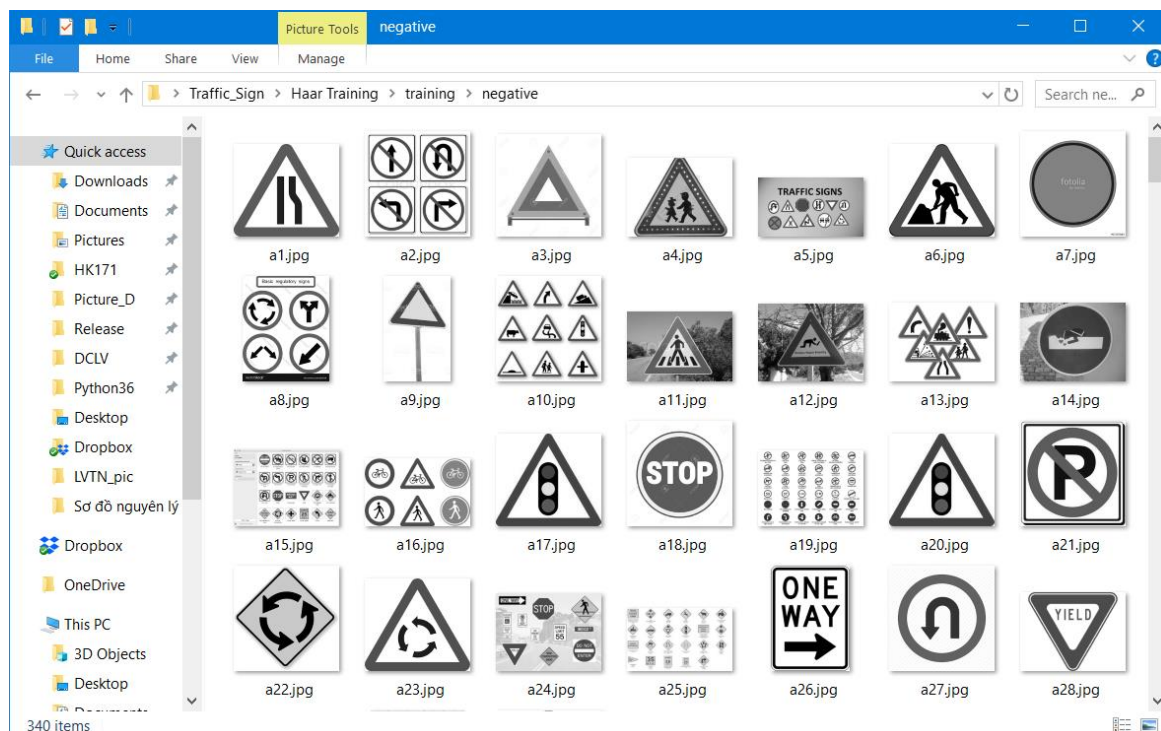


Hình 5.3 Hình ảnh đã được lấy ngưỡng dùng để huấn luyện

5.2.2. Biển báo giao thông



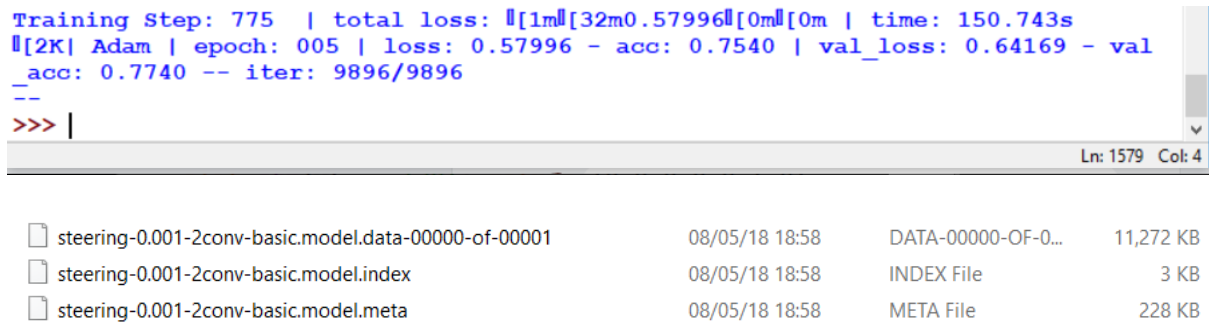
Hình 5.4 Ảnh tích cực (chứa đối tượng huấn luyện) định dạng bmp



Hình 5.5 Ảnh phủ định (không chứa đối tượng huấn luyện) đã được xám hóa

5.3. Kết quả thu được





5.3.1. Dữ liệu tạo ra sau khi huấn luyện



Training Step: 775 | total loss: 0.57996 | time: 150.743s
Adam | epoch: 005 | loss: 0.57996 - acc: 0.7540 | val_loss: 0.64169 - val_acc: 0.7740 -- iter: 9896/9896

| | | | |
|--|----------------|--------------------|-----------|
| steering-0.001-2conv-basic.model.data-00000-of-00001 | 08/05/18 18:58 | DATA-00000-OF-0... | 11,272 KB |
| steering-0.001-2conv-basic.model.index | 08/05/18 18:58 | INDEX File | 3 KB |
| steering-0.001-2conv-basic.model.meta | 08/05/18 18:58 | META File | 228 KB |

Hình 5.6 Kết quả và file thu được sau khi huấn luyện làn đường

| | | | |
|--|----------------|----------|-------|
|  danger_class.xml | 14/05/18 17:28 | XML File | 12 KB |
|  Limit.xml | 14/05/18 16:38 | XML File | 32 KB |
|  Psign_classifier.xml | 12/05/18 15:14 | XML File | 48 KB |
|  stopsign_classifier.xml | 01/10/17 12:42 | XML File | 91 KB |

Hình 5.7 File xml thu được sau khi huấn luyện biển báo giao thông

5.3.2. Kết quả kiểm chứng

a. Dự đoán góc queo của làn đường

```

goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.95853615
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.99569136
goc queo = 120 Predict : [0,1,0,0,0] Truth : [0 1 0 0 0] acc = 0.9901877
goc queo = 150 Predict : [1,0,0,0,0] Truth : [1 0 0 0 0] acc = 0.99598557
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.9996954
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.9995826
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.99483114
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.99507093
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.67452115
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.98444957
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.80551404
goc queo = 60 Predict : [0,0,0,1,0] Truth : [0 0 1 0 0] acc = 0.7039638
Failed prediction
goc queo = 120 Predict : [0,1,0,0,0] Truth : [0 1 0 0 0] acc = 0.9966174
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.7528082
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.93947625
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.89569956
goc queo = 60 Predict : [0,0,0,1,0] Truth : [0 0 0 1 0] acc = 0.5796194
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.8803303
goc queo = 150 Predict : [1,0,0,0,0] Truth : [1 0 0 0 0] acc = 0.98543143
goc queo = 100 Predict : [0,0,1,0,0] Truth : [0 0 1 0 0] acc = 0.805868

```

```

goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 1 0 0] acc =  0.9945574
goc queo = 30     Predict : [0,0,0,0,1] Truth :  [0 0 0 0 1] acc =  0.9963308
goc queo = 120    Predict : [0,1,0,0,0] Truth :  [0 1 0 0 0] acc =  0.9900774
goc queo = 30     Predict : [0,0,0,0,1] Truth :  [0 0 0 0 1] acc =  0.99936754
goc queo = 120    Predict : [0,1,0,0,0] Truth :  [0 1 0 0 0] acc =  0.7032498
goc queo = 60     Predict : [0,0,0,1,0] Truth :  [0 0 1 0 0] acc =  0.68839514
Failed prediction
goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 1 0 0] acc =  0.7292603
goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 1 0 0] acc =  0.99986887
goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 0 1 0] acc =  0.694566

goc queo = 120    Predict : [0,1,0,0,0] Truth :  [0 1 0 0 0] acc =  0.8796328
goc queo = 60     Predict : [0,0,0,1,0] Truth :  [0 0 0 1 0] acc =  0.98888844
goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 1 0 0] acc =  0.96340597
goc queo = 100    Predict : [0,0,1,0,0] Truth :  [0 0 1 0 0] acc =  0.9719105
goc queo = 60     Predict : [0,0,0,1,0] Truth :  [0 0 1 0 0] acc =  0.9445152
Failed prediction
goc queo = 120    Predict : [0,1,0,0,0] Truth :  [0 0 1 0 0] acc =  0.511858
Failed prediction
correct prediction : 767 out of 1000 images

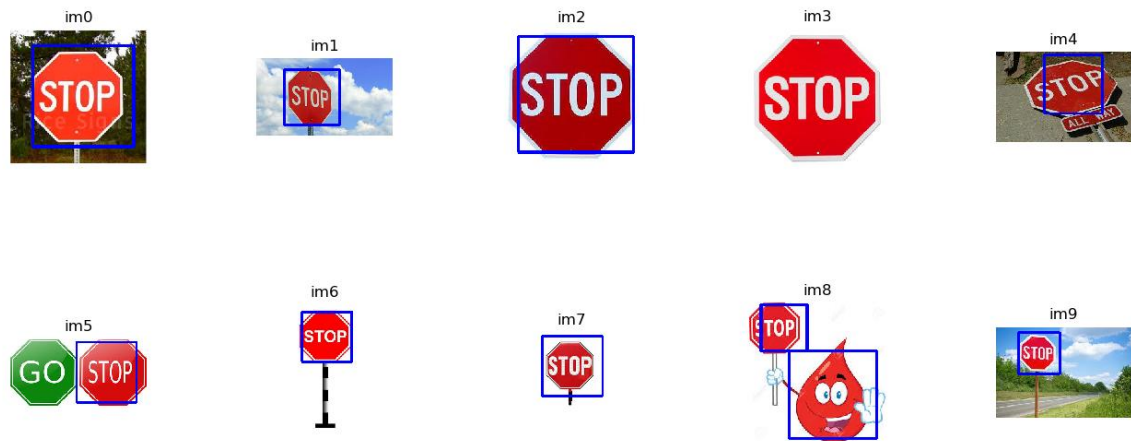
```

Dữ liệu dùng để kiểm tra độ chính xác là 1000 mẫu đã được label nhưng không nằm trong dữ liệu dùng để huấn luyện model. Kết quả số ảnh dự đoán chính xác là 767 ảnh trên 1000 ảnh, tương ứng 76.7%.

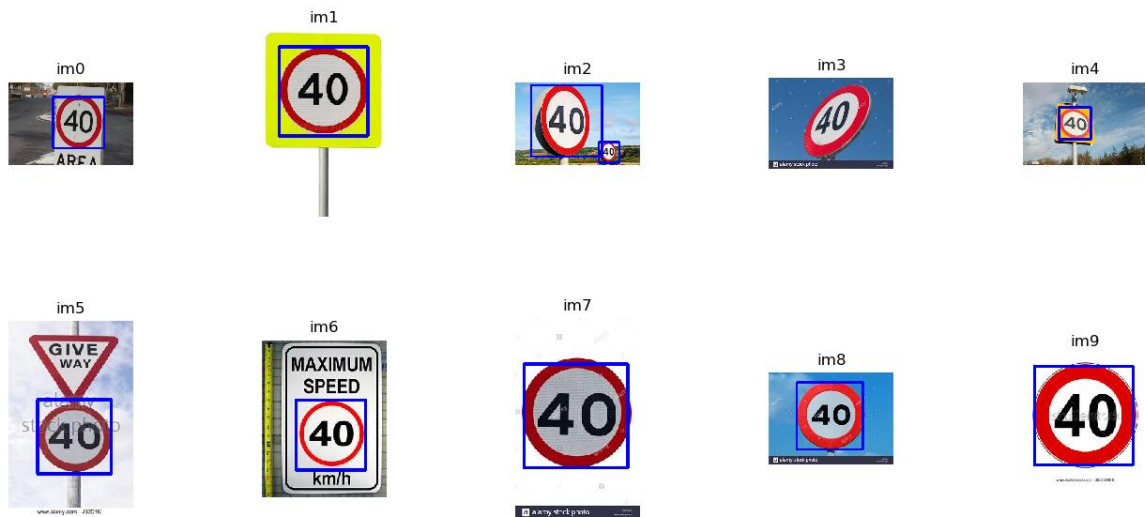
b. Dự đoán biển báo Park, Stop, Limit_40 và Danger



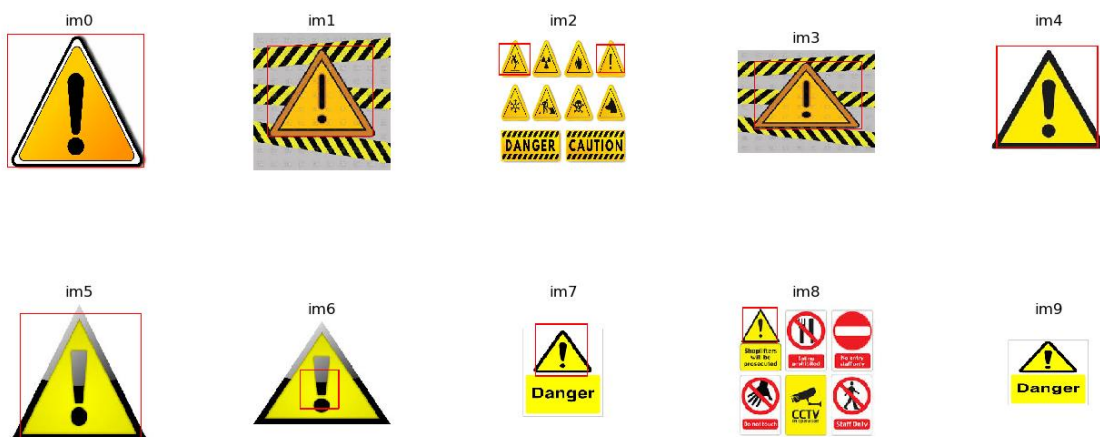
Hình 5.8 Mẫu thử cho biển báo Park



Hình 5.9 Mẫu thử cho biển báo Stop



Hình 5.10 Mẫu thử cho biển báo Limit_40



Hình 5.11 Mẫu thử cho biển báo Danger

Tỉ lệ chính xác của bốn loại biển báo được nhận diện trong hệ thống xe tự hành:

PARK: 154 detected out of 199 image (Dữ liệu dùng để kiểm tra độ chính xác là 199 mẫu đã được label nhưng không nằm trong dữ liệu dùng để huấn luyện model. Kết quả số ảnh dự đoán chính xác là 154 ảnh trên 199 ảnh, tương ứng 77.3%)

STOP: 94 detected out of 101 image (Dữ liệu dùng để kiểm tra độ chính xác là 101 mẫu đã được label nhưng không nằm trong dữ liệu dùng để huấn luyện model. Kết quả số ảnh dự đoán chính xác là 94 ảnh trên 101 ảnh, tương ứng 93.1%)

LIMIT_40: 156 detected out of 197 image (Dữ liệu dùng để kiểm tra độ chính xác là 197 mẫu đã được label nhưng không nằm trong dữ liệu dùng để huấn luyện model. Kết quả số ảnh dự đoán chính xác là 156 ảnh trên 197 ảnh, tương ứng 79.2%)

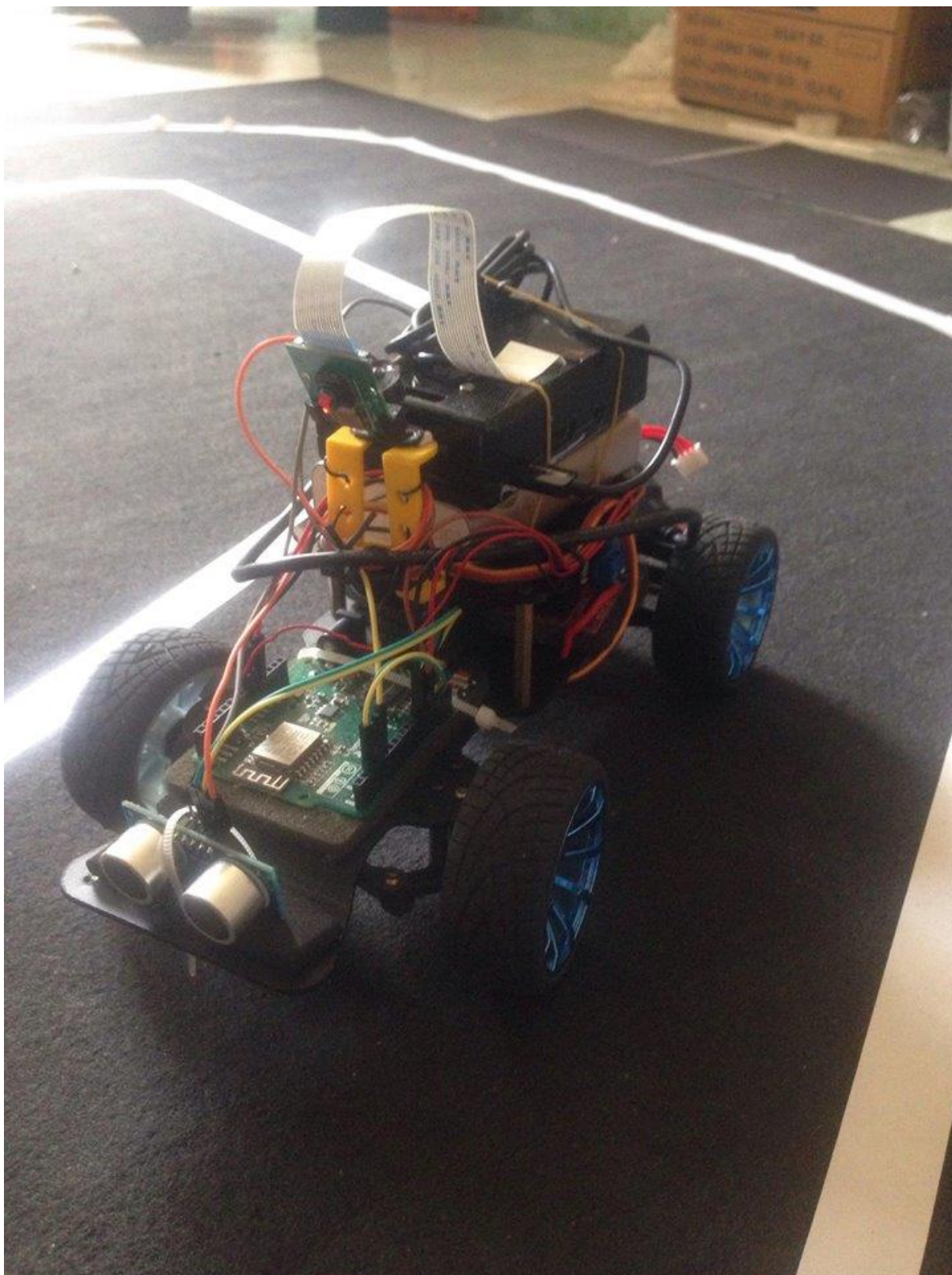
DANGER: 147 detected out of 207 image (Dữ liệu dùng để kiểm tra độ chính xác là 207 mẫu đã được label nhưng không nằm trong dữ liệu dùng để huấn luyện model. Kết quả số ảnh dự đoán chính xác là 147 ảnh trên 207 ảnh, tương ứng 71.0%)

c. Đánh giá toàn bộ hệ thống

Hệ thống xe tự hành là sự kết hợp giữa sự nhận diện góc quẹo của làn đường (sử dụng mạng CNN) và nhận diện 4 loại biển báo (sử dụng đặc trưng Haar-like). Hai phương pháp này hoạt động độc lập và do tốc độ xử lý của các tác vụ rất nhanh nên nhìn chung chúng xảy ra gần như đồng thời trong quá trình hoạt động của xe tự hành. Trong đó sai số của từng phần là:

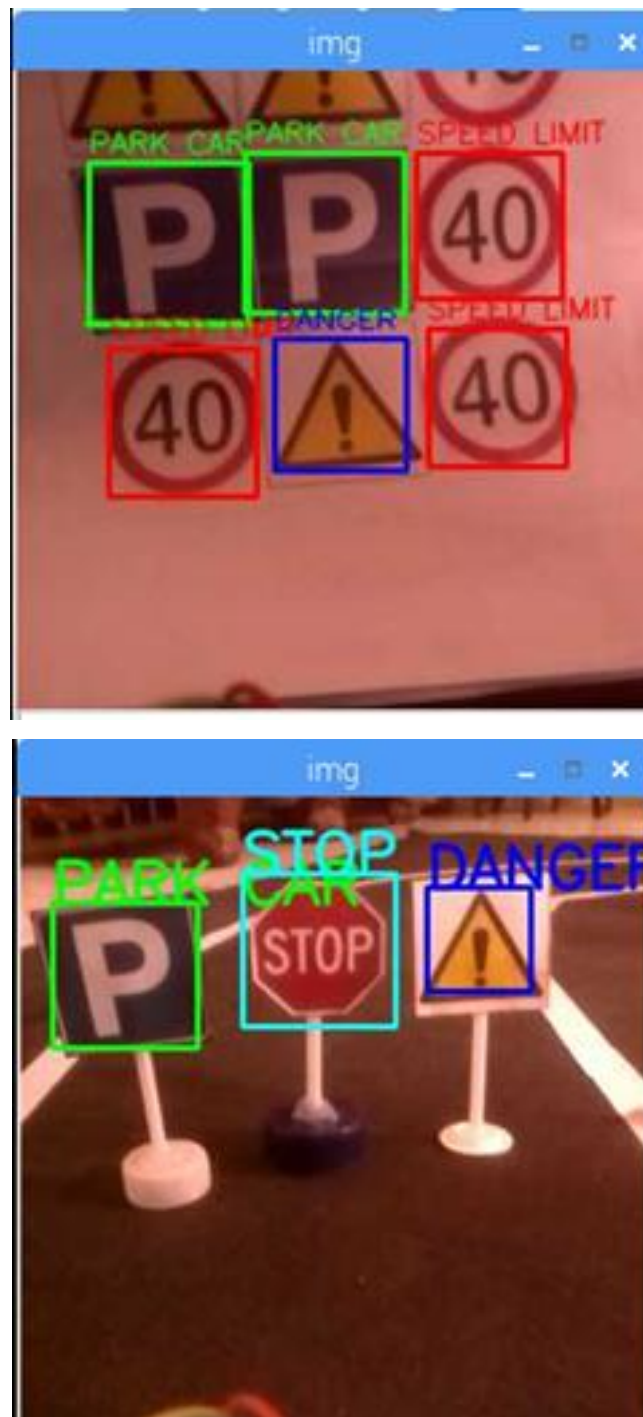
- Sai số khi nhận diện góc quẹo làn đường là 0.233.
- Sai số khi nhận diện biển báo Park, Stop, Limit_40 và Danger lần lượt là 0.227, 0.069, 0.208 và 0.29.

5.3.3. Kết quả thi công



Hình 5.12 Mô hình xe tự hành

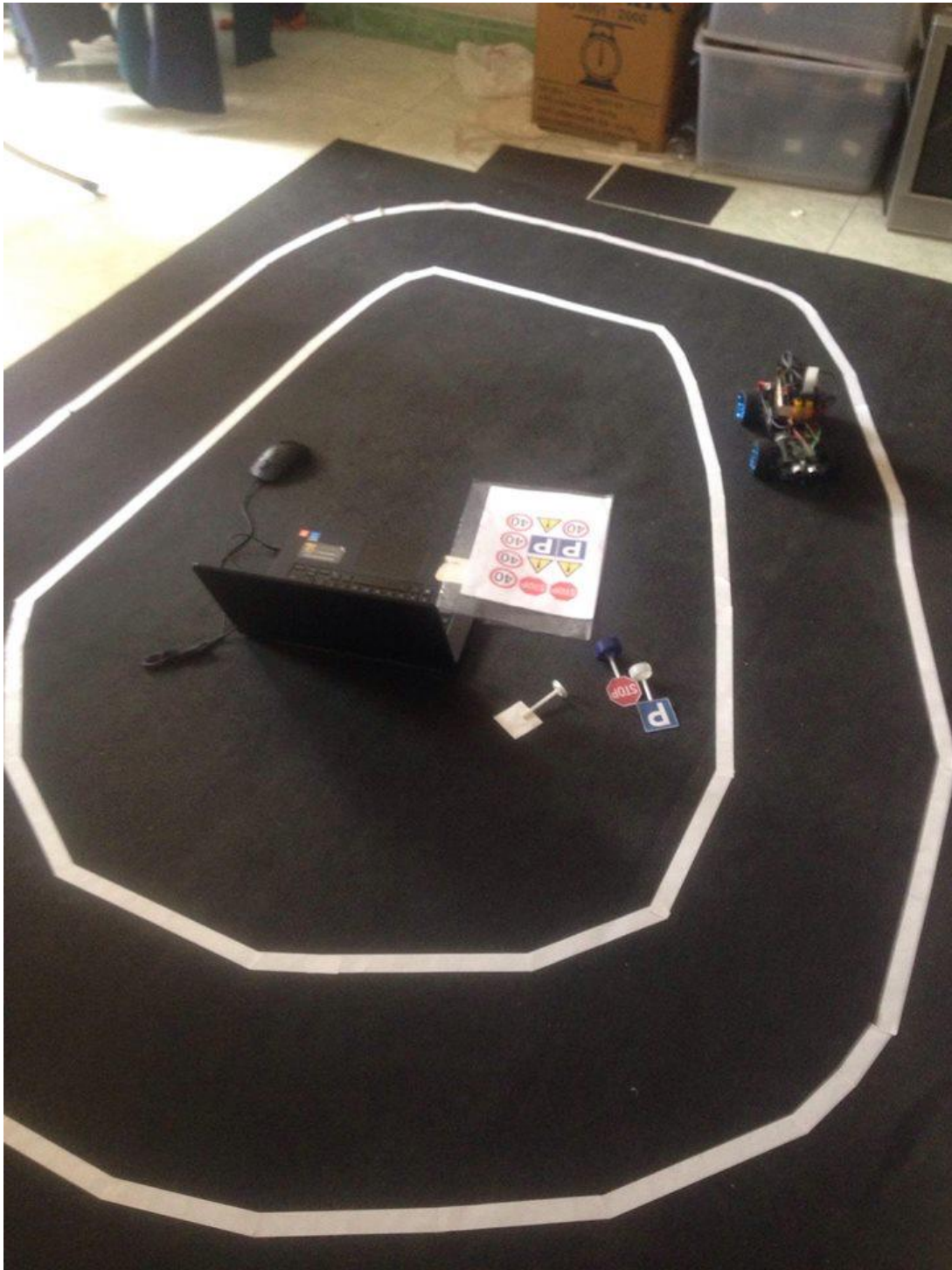
5.3.4. Kết quả nhận diện biển báo giao thông



Hình 5.13 Kết quả nhận diện nhiều biển báo cùng lúc

Nhận xét: Kết quả thực sự nhận diện được 4 biển báo giao thông đã huấn luyện. Tuy nhiên, không thể nhận diện nếu để biển báo quá xa.

5.3.5. Kết quả mô phỏng chạy tự động



Hình 5.14 Mô hình làn đường để xe chạy

Nhận xét: Xe có thể tự căn làn để tránh bị trật khỏi làn đường, rẽ trái, phải theo vạch đường, giảm tốc độ khi vào khúc cua quẹo, dừng lại khi gặp vật cản.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Sau một thời gian tìm tòi học hỏi cũng như được sự hướng dẫn của thầy **Hồ Trung Mỹ**, nhóm chúng em đã hoàn thành được đề tài: “**Xe tự hành**” trên Raspberry Pi 3 kết hợp với Arduino Iot Wifi Uno sử dụng ngôn ngữ lập trình Python và Arduino (được thiết kế dựa trên C). Trong quá trình thực hiện đề tài, nhóm chúng em đã đúc kết được rất nhiều kinh nghiệm quý báu cho bản thân để phục vụ quá trình học tập và làm việc sau này.

Đề tài của chúng em có những ưu nhược điểm như sau:

➤ Ưu điểm:

- ✓ Có thể tự động căn làn để không bị trật ra khỏi làn đường.
- ✓ Chủ động rẽ trái, phải theo vạch đường, giảm tốc độ khi gặp góc cua.
- ✓ Nhận diện được 4 loại biển báo giao thông.
- ✓ Tự động phanh khi gặp vật cản bất ngờ phía trước.

➤ Nhược điểm:

- ✓ Có thể bị sai góc quẹo trong một số trường hợp nhất định.
- ✓ Do tải và ma sát nhiều nên tốc độ xe vẫn còn chậm.
- ✓ Số lượng biển báo nhận diện chưa được phong phú.

Với những kết quả đạt được như trên, hệ thống rất hữu ích cho các bạn sinh viên trong việc thực hành trên mô hình thực tế đã triển khai bằng ngôn ngữ lập trình Python được ứng dụng nhiều trong các dự án IoT. Bên cạnh đó việc sử dụng Raspberry như một chiếc máy tính thu nhỏ giúp ta làm quen với hệ điều hành Linux và các ứng dụng tuyệt vời của nó.

6.2. Hướng phát triển

Từ ứng dụng “***Xe tự hành***”, ta có thể phát triển sản phẩm ở một mô hình lớn hơn, như xe tự động giám sát và thực hiện các yêu cầu bên trong một nhà máy hay xa hơn là chạy trên đường thực tế.

Hơn thế nữa, xe không người lái đang là xu thế cho tương lai và không còn là sân chơi của các hãng ô tô truyền thống mà đã dịch chuyển sang các công ty phần mềm khi 90% sáng tạo của ô tô nằm ở đây. Do đó việc phát triển các ứng dụng cho xe tự lái rất có tiềm năng trong tương lai.

7. TÀI LIỆU THAM KHẢO

- [1] A. Mordvintsev and K. Abid, “OpenCV-Python Tutorials Documentation Release 1” [Online]. [https://readthedocs.org/projects/opencv-python tutroals/downloads/pdf/latest/](https://readthedocs.org/projects/opencv-python/tutroals/downloads/pdf/latest/), Nov. 05, 2017.
- [2] Nguyễn Nhật, “[Xu hướng] Xe tự lái thông minh: Tương lai của ngành công nghệ ô tô là đây!” [Online]. <https://www.thegioididong.com/tin-tuc/xu-huong-xe-tu-lai-thong-minh-tuong-lai-cua-nganh-cong-nghe-oto-la-day--667481>, July. 17, 2015.
- [3] Ban Thời Sự, “Xe tự lái đầu tiên lăn bánh tại Việt Nam” [Online]. <http://vtv.vn/cong-nghe/xe-tu-lai-lan-dau-tien-lan-banh-tai-viet-nam-20171112095532888.htm>, Nov. 12, 2017.
- [4] A. Sohag, “How to Install OpenCV 3.4.0 with Python 3 on Raspberry Pi 3” [Online]. <http://www.life2coding.com/install-opencv-3-4-0-python-3-raspberry-pi-3>, Mar. 13, 2018.
- [5] Blynk, “Docs”, [Online]. <http://docs.blynk.cc/>
- [6] Bùi Trần Duy Vũ, “Công nghệ cảm quan máy tính” [Online]. <http://www.pcworld.com.vn/articles/cong-nghe/cong-nghe/2009/01/1193512/cong-nghe-cam-quan-may-tinh/>, Jan. 22, 2009.
- [7] Hoàng Đăng Quang, “Adaboost - Haar Features - Face detection” [Online]. http://www.ieev.org/2010/03/adaboost-haar-features-face-detection_22.html, Mar, 2010.
- [8] M. Rezaei, “Creating a Cascade of Haar-Like Classifiers: Step by Step” [Online]. <https://www.cs.auckland.ac.nz>
- [9] Nguyễn Phúc Lương, “Ứng dụng Convolutional Neural Network trong bài toán phân loại ảnh” [Online]. <https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8yIYM>, Nov. 28, 2017.
- [10] Tan Nguyen, “Mạng nơ-ron tích chập (P1)” [Online]. <https://viblo.asia/p/mang-no-ron-tich-chap-p1-DZrGNNjPGVB>, Sep. 27, 2016.

8. PHỤ LỤC

8.1. Code huấn luyện dữ liệu (hình ảnh và thông số) làn đường

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from utils import INPUT_SHAPE, batch_generator
import argparse
import os
import cv2
import string

PATH = 'C:/Users/Admin/Desktop/\
car-behavioral-cloning-master/data/driving_log.csv'
TEST_SIZE = 1000
IMG_SIZE = 50
LR = 1e-3
threshold = 170
MODEL_NAME = 'steering-gray-{}-{}.model'.format(LR,
                                                  '2conv-basic')

def label_img(y): #180-150-120-100-60-30-0
    if str(y) == '180': return [1,0,0,0,0]
    elif str(y) == '150': return [1,0,0,0,0]
    elif str(y) == '120': return [0,1,0,0,0]
    elif str(y) == '100': return [0,0,1,0,0]
    elif str(y) == '60': return [0,0,0,1,0]
    elif str(y) == '30': return [0,0,0,0,1]
    elif str(y) == '0': return [0,0,0,0,1]

def create_train_and_test_data():
    test_data = []
    train_data = []
    train_x, test_x, train_y, test_y = load_csv_data()
    for i in range(len(train_x)):
        path_img = (train_x[i].replace('\\', '/'))
        img = cv2.imread(path_img.replace('Nguyen Duc', 'Admin'),
                          cv2.IMREAD_GRAYSCALE)
        img[img>= threshold] = 255
        img[img < threshold] = 0
        img = cv2.resize(img, (128, 64))
        train_data.append([np.array(img),
                           np.array(label_img(train_y[i]))])
    np.save('train_data.npy', train_data)
    for i in range(len(test_x)):
        path_img = (test_x[i].replace('\\', '/'))
        img = cv2.imread(path_img.replace('Nguyen Duc', 'Admin'),
                          cv2.IMREAD_GRAYSCALE)
        img[img>= threshold] = 255
        img[img < threshold] = 0
        img = cv2.resize(img, (128, 64))
        test_data.append([np.array(img),
                           np.array(label_img(test_y[i]))])
    np.save('test_data.npy', test_data)
    return train_data, test_data
```



```

def load_csv_data():
    data_df = pd.read_csv(os.path.join(PATH))
    x = data_df['center'].values
    y = data_df['steering'].values
    tr_x, tst_x, tr_y, tst_y = train_test_split(x,
                                                y,
                                                test_size = TEST_SIZE,
                                                random_state=0)

    return tr_x, tst_x, tr_y, tst_y

train_data, test_data = create_train_and_test_data()
# If you have already created the dataset:
##train_data = np.load('train_data.npy')

import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, \
dropout, fully_connected, flatten
from tflearn.layers.estimator import regression
import tensorflow as tf

tf.reset_default_graph()

convnet = input_data(shape=[None, 128, 64, 1], name='input')

convnet = conv_2d(convnet, 24, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 36, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 48, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 3, activation='relu')
#convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 3, activation='relu')
#convnet = max_pool_2d(convnet, 5)

convnet = dropout(convnet, 0.8)
convnet = flatten(convnet, name='Flatten')

convnet = fully_connected(convnet, 100, activation='relu')

convnet = fully_connected(convnet, 50, activation='relu')

convnet = fully_connected(convnet, 5, activation='softmax')
convnet = regression(convnet, optimizer='adam',
                      learning_rate=LR,
                      loss='categorical_crossentropy',
                      name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')

```

```
if os.path.exists('{} .meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')

train = train_data[:-200]
test = train_data[-200:]

X = np.array([i[0] for i in train]).reshape(-1,128,64,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,128,64,1)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=5,
        validation_set=({'input': test_x}, {'targets': test_y}),
        snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

model.save(MODEL_NAME)
```

8.2. Code thu thập dữ liệu (bên Raspberry)

```
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import serial
import pandas as pd
basename = "steering_%s.jpg"
label = []
steering_csv = []
t = 1 # dem so luong hinh anh
global steering # gia tri goc quay
i = 1 # dem hinh csv
csv_path = "./driving_log.csv"

camera = PiCamera()
camera.resolution = (50, 50)
camera.framerate = 20
rawCapture = PiRGBArray(camera, size=(50, 50))
camera.awb_mode = 'off'
camera.awb_gains = (1.8,1.5)
time.sleep(0.1)
ser = serial.Serial()
ser.baudrate = 115200
```

```

try:
    ser.port = '/dev/ttyUSB1'
    ser.open()
except serial.serialutil.SerialException :
    try:
        ser.port = '/dev/ttyUSB0'
        ser.open()
    except serial.serialutil.SerialException :
        ser.port = 'COM4'
        ser.open()
ser.flushInput()

def insert_csv(basename,steering):
    global i,t
    global label, steering_csv
    text='C:/Users/Nguyen Duc/Desktop/\
car-behavioral-cloning-master/data/IMG/%s'%(basename%(t))
    label.insert(i,text)
    steering_csv.insert(i,steering)
    i+=1
    t+=1

def flip_image(image,steering):
    flip_img=cv2.flip(image,1)
    if (steering != 100 ):
        steering = 180 - steering
    return flip_img, steering

def nothing(x):
    pass

cv2.namedWindow('abc')
cv2.createTrackbar('Threshold','abc',160,255,nothing)

for frame in camera.capture_continuous(rawCapture,
                                     format="bgr",
                                     use_video_port=True):

    img = frame.array
    image = img.astype(np.uint8)
    image = cv2.flip( image, -1 )
    img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    imshape = image.shape
    cv2.imshow('image',image)
    threshold = cv2.getTrackbarPos('Threshold','abc')
    img[:10,:]=0
    img[img < threshold ] = 0
    img[img >= threshold ] = 255
    cv2.imshow('anh',img)

```

```

try:
    if (ser.inWaiting() > 0 ):
        print('bye wait = ',ser.inWaiting())
        mess=ser.readline()
        try:
            steering = int(mess)
            if ( steering == 97):
                steering = 100
            print('steering angle = ',steering)|
            # save frame as JPEG file
            cv2.imwrite("./IMG/steering_%d.jpg" % (t), img)
            cv2.imwrite("./IMG1/steering_%d.jpg" % (t), image)
            insert_csv(basename,steering)
            flip_img,flip_steering = flip_image(img,steering)
            flip_img1,_ = flip_image(image,steering)
            cv2.imwrite("./IMG/steering_%d.jpg" % (t), flip_img)
            cv2.imwrite("./IMG1/steering_%d.jpg" % (t), flip_img1)
            insert_csv(basename,flip_steering)
            print('anh thu i,t = ',i,' ',t)
        except ValueError :
            print('mess = ',mess)
        ser.flushInput()
except OSError :
    print('rekt')
    print('port hien tai = ',ser.port)
    ser.close()
    time.sleep(3)
    ser.open()
    print(' co port connect chua : ',ser.isOpen())
    print(' so byte dang doi : ',ser.inWaiting())
    if ( ser.inWaiting() > 20):
        mess1=ser.readline()
        print('mess1 = ',mess1)
        ser.flushInput()
    pass

rawCapture.truncate(0)
if cv2.waitKey(10) & 0xFF == ord('q'):
    break
export_csv()
cv2.destroyAllWindows()

```

8.3. Code thu thập dữ liệu (bên Arduino)

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SPI.h>
#include <Servo.h>
Servo servo;
BlynkTimer timer;

int in1 = 12; //đỏ
int in2 = 13; //đen
int ConA = 14; //nâu

int activetimer;
#define SPEED 300 // from 0-1023

char auth[] = "5cfbadfc357c4cc8ba1ef0aba0ca882b";
char ssid[] = "Wifi Name";
char pass[] = "Wifi Password";

int pinValue ;
int PWM ;

BLYNK_WRITE(V1)
{
    pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
    pinValue = pinValue*30;
    if (pinValue > 180)
        pinValue = 180;
    if (( pinValue >= 90) && (pinValue < 150 ))
        pinValue = 97;
    if (pinValue >= 150)
        pinValue = 120;

    servo.write(pinValue);
}
```

```
BLYNK_WRITE(V2)
{
    PWM = param.asInt(); // assigning incoming value from pin V1 to a variable
    if(PWM > 511)
    {
        timer.enable(activetimer);
        digitalWrite(in2, HIGH);
        digitalWrite(in1, LOW);
    }
    else if ( (PWM == 511) )
    {
        timer.disable(activetimer);
        digitalWrite(in2, LOW);
        digitalWrite(in1, LOW);
    }
    else
    {
        timer.disable(activetimer);
        digitalWrite(in2, LOW);
        digitalWrite(in1, HIGH);
    }
}

void mytimerevent()
{
    //Truyen gia tri serial moi khi timer hoạt động
    Serial.println(pinValue);
}

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

    Blynk.begin(auth, ssid, pass);
    servo.attach(2);
    activetimer = timer.setInterval(100L,mytimerevent);
    timer.disable(activetimer);
    Blynk.virtualWrite(V2, 511);
    pinMode(in1, OUTPUT); //Declaring the pin modes, obviously they're outputs
    pinMode(in2, OUTPUT);
    pinMode(ConA, OUTPUT);
}

BLYNK_CONNECTED()
{
    Blynk.syncAll();
}

void loop()
{
    Blynk.run();
    timer.run();
}
```

8.4. Code tự động chạy (bên Raspberry)

```
import pandas as pd
import cv2
import numpy as np
import time
from sklearn.model_selection import train_test_split
TEST_SIZE = 20
IMG_SIZE = 50
LR = 1e-3
threshold = 120
high_threshold = 190
MODEL_NAME = 'steering-{}-{}.model'.format(LR, '2conv-basic') |
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected, flatten
from tflearn.layers.estimator import regression
from picamera.array import PiRGBArray
from picamera import PiCamera
camera = PiCamera()
camera.resolution = (320, 160)
camera.framerate = 15
rawCapture = PiRGBArray(camera, size=(320, 160))
camera.awb_mode = 'off'
camera.awb_gains = (1.8, 1.5)
time.sleep(0.1)
import serial
import struct
ser = serial.Serial()
ser.baudrate = 9600
try:
    ser.port = '/dev/ttyUSB1'
    ser.open()
except serial.serialutil.SerialException :
    try:
        ser.port = '/dev/ttyUSB0'
        ser.open()
    except serial.serialutil.SerialException :
        ser.port = '/dev/ttyUSB2'
        ser.open()
ser.flushInput()
ser.flushOutput()
time.sleep(2)
```



```

P_cascade = cv2.CascadeClassifier('./Cascade_Classifier/Psign2.xml')
Stop_cascade = cv2.CascadeClassifier('./Cascade_Classifier/stopsign_classifier.xml')
Danger_cascade = cv2.CascadeClassifier('./Cascade_Classifier/danger_class.xml')
Limit_cascade = cv2.CascadeClassifier('./Cascade_Classifier/Limit.xml')

import tensorflow as tf
import tensorflow as tf
import argparse
import os
import cv2
import string

def nothing(x):
    pass
def control(model_out):
    if(np.argmax(model_out) == 0):
        print('goc queo = 150 ', 'acc = ', model_out[0][0])
        ser.write(struct.pack('>B', 120))
        if (ser.inWaiting() > 0 ):
            reachedPos = str(ser.readline())
            print(reachedPos)
            ser.flushOutput()
    elif (np.argmax(model_out) == 1):
        print('goc queo = 120 ', 'acc = ', model_out[0][1])
        ser.write(struct.pack('>B', 120))
        if (ser.inWaiting() > 0 ):
            reachedPos = str(ser.readline())
            print(reachedPos)
            ser.flushOutput()
    elif (np.argmax(model_out) == 2):
        print('goc queo = 90 ', 'acc = ', model_out[0][2])
        ser.write(struct.pack('>B', 97))
        if (ser.inWaiting() > 0 ):
            reachedPos = str(ser.readline())
            print(reachedPos)
            ser.flushOutput()
    elif (np.argmax(model_out) == 3):
        print('goc queo = 60 ', 'acc = ', model_out[0][3])
        ser.write(struct.pack('>B', 60))
        if (ser.inWaiting() > 0 ):
            reachedPos = str(ser.readline())

            ser.flushOutput()
    elif (np.argmax(model_out) == 4):
        print('goc queo = 30 ', 'acc = ', model_out[0][4])
        ser.write(struct.pack('>B', 35))
        if (ser.inWaiting() > 0 ):
            reachedPos = str(ser.readline())
            print(reachedPos)
            ser.flushOutput()

```

```

convnet = input_data(shape=[None, 128, 64, 1], name='input')

convnet = conv_2d(convnet, 24, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 36, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 48, 5, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 3, activation='relu')

convnet = conv_2d(convnet, 64, 3, activation='relu')

convnet = dropout(convnet, 0.8)
convnet = flatten(convnet, name='Flatten')

convnet = fully_connected(convnet, 100, activation='relu')

convnet = fully_connected(convnet, 50, activation='relu')

convnet = fully_connected(convnet, 5, activation='softmax')
convnet = regression(convnet, optimizer='adam',
                      learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')

if os.path.exists('{} .meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')

cv2.namedWindow('abc')
cv2.createTrackbar('Threshold', 'abc', 160, 255, nothing)
cv2.createTrackbar('high_Threshold', 'abc', 170, 255, nothing)

font = cv2.FONT_HERSHEY_SIMPLEX

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    frame1 = image.astype(np.uint8)
    img_flip = cv2.flip( frame1, -1 )
    gray = cv2.cvtColor(img_flip, cv2.COLOR_BGR2GRAY)
    stop_flag=0
    stop = Stop_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in stop:
        cv2.rectangle(img_flip, (x,y), (x+w,y+h), (255,0,0), 2)
        cv2.putText(img_flip, 'STOP', (x,y-5), font, 0.5, (255,255,0), 1, cv2.LINE_AA)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img_flip[y:y+h, x:x+w]
        stop_flag = 1

```

```

threshold = cv2.getTrackbarPos('Threshold','abc')
high_threshold = cv2.getTrackbarPos('high_Threshold','abc')
gray[gray < threshold] = 0
gray[gray >= threshold] = 255
img1 = cv2.resize(gray, (128,64))
cv2.imshow('org',img1)
model_out = model.predict(img1.reshape(-1,128,64,1))
if(stop_flag == 0):
    control(model_out)
else:
    ser.write(struct.pack('>B', 103))
    if (ser.inWaiting() > 0):
        reachedPos = str(ser.readline())
        print(reachedPos)
        ser.flushOutput()

park = P_cascade.detectMultiScale(gray, 1.3, 5)
danger = Danger_cascade.detectMultiScale(gray, 1.1, 3)
Limit = Limit_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in park:
    cv2.rectangle(img_flip, (x,y), (x+w,y+h), (0,255,0),2)
    cv2.putText(img_flip, 'PARK CAR', (x,y-5), font, 0.5, (0,255,0),1,cv2.LINE_AA)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img_flip[y:y+h, x:x+w]

for (x,y,w,h) in danger:
    cv2.rectangle(img_flip, (x,y), (x+w,y+h), (255,0,0),2)
    cv2.putText(img_flip, 'DANGER', (x,y-5), font, 0.5, (255,0,0),1,cv2.LINE_AA)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img_flip[y:y+h, x:x+w]

for (x,y,w,h) in Limit:
    cv2.rectangle(img_flip, (x,y), (x+w,y+h), (0,0,255),2)
    cv2.putText(img_flip, 'SPEED LIMIT', (x,y-5), font, 0.5, (0,0,255),1,cv2.LINE_AA)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img_flip[y:y+h, x:x+w]

cv2.imshow('anh final',img_flip)
rawCapture.truncate(0)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break

cv2.destroyAllWindows()

```

8.5. Code tự động chạy (bên Arduino)

```
#include <Servo.h>
Servo servo;
#include <SPI.h>
#define TRIG_PIN 16
#define ECHO_PIN 0
#define TIME_OUT 5000
#define queoratnhieu 435
#define queonhieu 445
#define dithang 455

float GetDistance()
{
    long duration, distanceCm;
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    duration = pulseIn(ECHO_PIN, HIGH, TIME_OUT);
    distanceCm = duration / 29.1 / 2;
    return distanceCm;
}

int in1 = 12; //đỏ
int in2 = 13; //đen
int ConA = 14; //nâu

int SPEED = 350;
void MotorA_forward(){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    analogWrite(ConA, SPEED);
}

void MotorA_backward(){
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    analogWrite(ConA, SPEED);
}

void MotorA_OFF(){
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    analogWrite(ConA, 0);
}
```

```
void setup() {
    Serial.begin(9600);
    servo.attach(2);
    pinMode(in1, OUTPUT); //Declaring the pin modes, obviously they're outputs
    pinMode(in2, OUTPUT);
    pinMode(ConA, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
}

void loop() {
    int message;
    long distance = GetDistance();
    if(Serial.available()>0)
    {
        message = Serial.read();
        if(message == 103)
        {
            MotorA_OFF();
            Serial.print("STOP SIGN IN FRONT ");
            Serial.println(message);
        }
        else
        {
            if ((distance < 15) && (distance > 0))
            {
                MotorA_OFF();
                if(Serial.available()>0)
                message = Serial.read();
                //Serial.println(message);
                Serial.print("OBSTACLE IN FRONT ");
                Serial.println(distance);
            }

            else
            {
                if ( message == 35 )
                SPEED = queoratnhieu;
                else if ( message == 60)
                SPEED = queonhieu;
                else if ( message == 97)
                SPEED = dithang;
                else SPEED = queonhieu;
                servo.write(message);
                MotorA_forward();
                Serial.print("Servo in position: ");
                Serial.println(message);
            }
        }
    }
}
```