**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

TIỂU ĐOÀN 1

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

**Tên đề tài:**

**NGHIÊN CỨU CÔNG NGHỆ IOT, CÁC KỸ THUẬT HỌC MÁY TRONG PHÁT TRIỂN XE TỰ HÀNH.**

**Hà Nội, Tháng 4 năm 2020**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

TIỂU ĐOÀN 1

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

**Tên đề tài:**

**Nghiên cứu công nghệ iot, các kỹ thuật học máy trong phát triển xe tự hành**

**Mã số đề tài:**

**Lĩnh vực đăng ký thực hiện: CÔNG NGHỆ THÔNG TIN**

**Sinh viên tham gia thực hiện:**

**1. Học viên chủ trì:**

Lữ Thái Học - Lớp CNTT2 – c152 – d1

**2. Học viên tham gia:**

Nguyễn Trung Thành - Lớp CNTT2 – c152 – d1

Vũ Khánh Toàn - Lớp CNTT2 – c152 – d1

Giáo viên hướng dẫn: TS Hà Chí Trung – Khoa Công nghệ thông tin

**Hà Nội, Tháng 4 năm 2016**

**Mục lục**

[**LỜI MỞ ĐẦU** 4](#_Toc37643418)

[**1.** **Lý do chọn đề tài** 4](#_Toc37643419)

[**2.** **Mục tiêu và nhiệm vụ nghiên cứu** 4](#_Toc37643420)

[**3.** **Đối tượng và phạm vi nghiên cứu** 4](#_Toc37643421)

[**4.** **Phương pháp nghiên cứu** 5](#_Toc37643422)

[**5.** **Ý nghĩa khoa học của đề tài** 5](#_Toc37643423)

[**6.** **Bố cục đề tài** 5](#_Toc37643424)

[**PHẦN I. CÁC NỘI DUNG ĐÃ NGHIÊN CỨU THỰC HIỆN** 7](#_Toc37643425)

[**1.** **Nghiên cứu xây dựng thuật toán nhận diện đường bằng xử lý ảnh** 7](#_Toc37643426)

[**2.** **Xây dựng mô hình mạng Neural Network để nhận diện biển báo** 13](#_Toc37643427)

[**3.** **Nghiên cứu xây dựng chương trình điều khiển xe bằng phương pháp tìm quỹ đạo tối ưu và PID** 16](#_Toc37643428)

[**4. Xây dựng mô hình xe tự hành thử nghiệm sử dụng vi mạch Raspberry và Arduino** 19](#_Toc37643429)

[**PHẦN 2. CÀI ĐẶT CHƯƠNG TRÌNH VÀ CHẠY THỬ NGHIỆM** 23](#_Toc37643430)

[**1.** **Xây dựng chương trình** 23](#_Toc37643431)

[**2.** **Chạy thử nghiệm trên giả lập *(có video kèm theo)*** 23](#_Toc37643432)

[**3.** **Chạy thực tế với mô hình xe *(có video kèm theo)*** 24](#_Toc37643433)

[**PHẦN 3. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 25](#_Toc37643434)

[**1.** **Kết luận** 25](#_Toc37643435)

[**2.** **Hạn chế** 25](#_Toc37643436)

[**3.** **Hướng phát triển** 25](#_Toc37643437)

[**PHẦN 4. PHỤ LỤC** 27](#_Toc37643439)

**NGHIÊN CỨU CÔNG NGHỆ IOT, CÁC KỸ THUẬT HỌC MÁY TRONG PHÁT TRIỂN XE TỰ HÀNH**

# **LỜI MỞ ĐẦU**

1. **Lý do chọn đề tài**

Theo định nghĩa truyền thống, xe tự hành là loại ô tô có thể hoạt động mà không cần sự can thiệp từ con người dựa trên “bộ não” được lập trình thông minh với các khả năng nhận diện đường và các vật thể khác nhau, xác định chướng ngại vật cũng như xác định tuyến đường tốt nhất cho mỗi hành trình,…

Trong thời đại công nghệ thông tin hiện nay, với sự phát triển mạnh mẽ của xu hướng tự động hóa, xe tự hành đang là một lĩnh vực đang được nghiên cứu, đầu tư mạnh mẽ và phát triển một cách nhanh chóng.

Trên cơ sở yêu thích về công nghệ và nắm bắt xu hướng thời đại, nhóm đã đầu tư tìm hiểu và nghiên cứu về xe tự hành, với mong muốn được học hỏi và tích lũy các kiến thức về xe tự hành với các công nghệ được áp dụng, góp phần xây dựng các chương trình, chức năng, phần cứng và các ứng dụng hỗ trợ trong tự động hóa nói chung cũng như trong xe tự lái nói riêng.

1. **Mục tiêu và nhiệm vụ nghiên cứu**

Mục đích của đề tài nghiên cứu nhằm xây dựng các thuật toán nhận diện đường, biển báo, các thuật toán điều khiển, lập trình IoT và xây dựng mô hình xe chạy thử nghiệm các thuật toán với các chức năng sau:

* Nhận diện đường đi và chạy theo đường
* Né vật cản
* Nhận diện biển báo,
* Rẽ trái, phải, dừng khi có biển báo

Bên cạnh đó đề tài còn mong muốn giúp cho mọi người có một cái nhìn toàn diện hơn về xu hướng phát triển xe tự hành, khả năng và ứng dụng của xử lý ảnh, học máy và kỹ thuật điều khiển tự động hóa vào xe tự hành.

1. **Đối tượng và phạm vi nghiên cứu**

* ***Đối tượng nghiên cứu***

Đối tượng nghiên cứu của đề tài là xe tự hành và các công nghệ áp dụng vào xe tự hành, xử lý hình ảnh thu được từ camera, việc điều khiển xe dựa trên các thuật toán đã xây dựng.

* ***Phạm vi nghiên cứu***

Phạm vi nghiên cứu của đề tài liên quan đến các lĩnh vực:

* Xử lý ảnh thông qua việc sử dụng thư viện OpenCV (Open source Computer Vision)
* Kỹ thuật học máy và mạng CNN
* Dự đoán quỹ đạo và dùng hàm tối ưu để tìm quỹ đạo tối ưu
* Kỹ thuật điều khiển thông qua bộ điều khiển PID
* Lập trình IoT và sử dụng các vi mạch, động cơ, servo

1. **Phương pháp nghiên cứu**

* ***Phương pháp nghiên cứu lý thuyết***
* Tìm hiểu về cách lập trình với thư viện OpenCV, ứng dụng trong nhận diện đường theo màu sắc và xây dựng ảnh birdview cho đường đi.
* Tìm hiểu về kỹ thuật học máy, xây dựng và áp dụng mạng CNN trong nhận diện biển báo.
* Tìm hiểu nghiên cứu tính toán chọn hướng tối ưu cho xe tự hành
* Tìm hiểu về lý thuyết điều khiển, PID.
* Tìm hiểu về các phần cứng, vi mạch Raspberry và Arduino. Lập trình IoT kết nối các thiết bị..
* ***Phương pháp nghiên cứu thực nghiệm***
* Xây dựng thuật toán nhận diện đường bằng xử lý ảnh, so sánh phương pháp xây dựng được với phương pháp thông thường.
* Chạy thử nghiệm các thuật toán nhận diện, điều khiển trên giả lập có sẵn.
* Xây dựng mô hình xe và chạy thử nghiệm thực tế.

1. **Ý nghĩa khoa học của đề tài**

* ***Về mặt lý thuyết***
* Ứng dụng các công nghệ, thuật toán nghiên cứu được vào giả lập và xe tự hành mô hình đã xây dựng.
* Tạo tiền đề cho những nghiên cứu nâng cao và hiệu quả hơn trong tương lai
* ***Về mặt thực tiễn***
* Xây dựng nền tảng kiến thức về xử lý ảnh, điều khiển, lập trình IoT áp dụng trong tự động hóa, xe tự hành và các dự án IoT.

1. **Bố cục đề tài**

* **Mở đầu**
* **Các nội dung đã nghiên cứu thực hiện**

1. **Nghiên cứu xây dựng thuật toán nhận diện đường bằng xử lý ảnh**
2. **Xây dựng mô hình mạng Neuron nework để nhận diện biển báo**
3. **Nghiên cứu xây dựng chương trình điều khiển xe bằng phương pháp tìm quỹ đạo tối ưu và PID**
4. **Xây dựng mô hình xe tự hành thử nghiệm sử dụng vi mạch Raspberry và Arduino**

* **Cài đặt chương trình trên Raspberry và chạy thử nghiệm**
* **Kết luận và hướng phát triển**
* **Phụ lục**

# **PHẦN I. CÁC NỘI DUNG ĐÃ NGHIÊN CỨU THỰC HIỆN**

## **Nghiên cứu xây dựng thuật toán nhận diện đường bằng xử lý ảnh**

* 1. **Cơ bản về xử lý ảnh và thư viện OpenCV**

Xử lý ảnh và thị giác máy tính là lĩnh vực mà ngày nay được phát triển và ứng dụng rất rộng rãi trong rất nhiều lĩnh vực khác nhau nhờ vào sự phát triển mạnh mẽ của các hệ thống máy tính, các thuật toán và công trình nghiên cứu khác nhau của nhiều nhà khoa học trên thế giới.

Ở Việt Nam, các ứng dụng về xử ảnh đã được triển khai rộng rãi trên các lĩnh vực như nhận dạng biển biển số, hệ thống nhận dạng vân tay, nhận dạng chữ viết, con số, ký tự, nhận dạng khuôn mặt, cử chỉ, nhận dạng đối tượng và hành vi, mã QR, Robot và xe tự lái, phân tích hình ảnh y tế, …

Trong các công cụ xử lý ảnh, OpenCV(Open source computer vision) là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính, xử lý ảnh và máy học, cùng với các tính năng tăng tốc GPU trong hoạt động thời gian thực.

* 1. **Nhận diện đường dựa trên màu sắc với OpenCV**
     1. ***Không gian màu trong OpenCV***

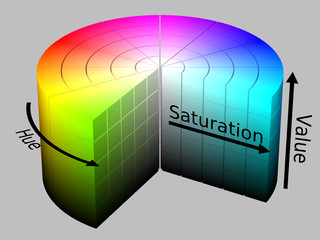
Không gian màu (Colorspaces) là một mô hình toán học biểu diễn các màu sắc trong thực tế dưới dạng số học. Có rất nhiều không gian màu trong OpenCV như: BGR,BGRA, HSV, CMYK,… Trong đó mỗi chữ cái đại diện cho một kênh màu(channel): B:Blue, G:Green, R:Red, H:Hue, S:Saturation, V:Value, A:Alpha, …

Một không gian màu có thể có 1 hoặc nhiều kênh, ví dụ RBG sẽ có 3 kênh (Red, Blue, Green) trong khi RBGA có thêm 1 kênh là Alpha.

Các pixel trong từng kênh của ảnh màu mang một giá trị có phạm vi tùy thuộc vào loại ảnh:

* 8 bit: [0,255]
* 16 bit: [0,65536]
* 32 bit (floating-point): [0.0,1.0]
  + 1. ***Nhận diện đường dựa trên không gian màu HSV***

Không gian màu HSV là không gian màu được dùng nhiều trong chỉnh sửa ảnh, phân tích ảnh và một phần của lĩnh vực thị giác máy tính. Hệ không gian màu này dựa vào 3 thông số sau để mô tả màu sắc H = hue: màu sắc, S = saturation: độ đậm đặc, sự bảo hòa, V = value: giá trị cường độ sáng.



*Không gian màu HSV*

Không gian màu HSV dùng để xử lý màu rất thuận tiện, vì mỗi màu sẽ có một giá trị Hue [0,180], còn Value và Saturation chỉ thị độ sáng và độ bão hòa của màu. Hệ màu này thường được sử dụng để lọc màu cho việc nhận diện đối tượng theo màu sắc.

* ***Nhận diện đường bằng cách sử dụng ngưỡng màu thông thường***

Ta tìm ra phần đường dựa trên sự khác biệt màu sắc của đường so với background (phương pháp lọc màu ảnh), chương trình dùng hệ màu HSV để lọc màu:

* Chuyển ảnh gốc từ BGR sang HSV.
* Tìm mã màu HSV trung bình của đường dựa trên mẫu cắt ra từ đường, giới hạn mã vùng màu của đường.
* Lọc ra những pixel có giá trị nằm trong giới hạn mã màu tìm được (sử dụng hàm cv::inRange()), các pixel có giá trị nằm ngoài vùng giới hạn được gán bằng giá trị đặc biệt khác (ví dụ màu đen).

***Ưu điểm:*** Thuật toán đơn giản chỉ cần tính ra mã màu sắc của đường rồi áp dụng ngưỡng phù hợp muốn để nhận diện.

***Khuyết điểm:*** Độ chính xác chưa cao, nhận diện không tốt với các ảnh có độ phân giải thấp.

* ***Nhận diện đường bằng cách tính độ hợp lý (likelihood) của từng pixel so với mẫu***

Các quá trình xử lý nhận diện:

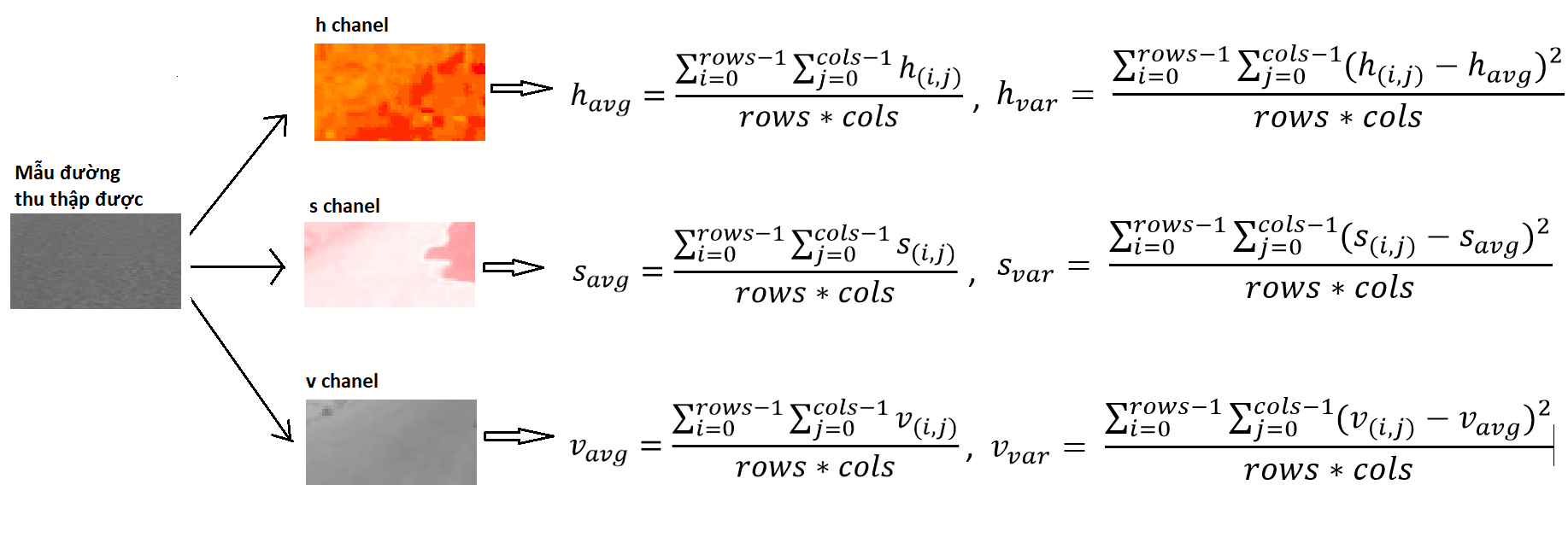


Figure 1.Tính toán giá trị trung bình (average) và phương sai (variance) của mẫu đường theo kênh màu

Với α là khả giới hạn xác xuất chấp nhận được

β là khoảng chênh lệch độ sáng chấp nhận được

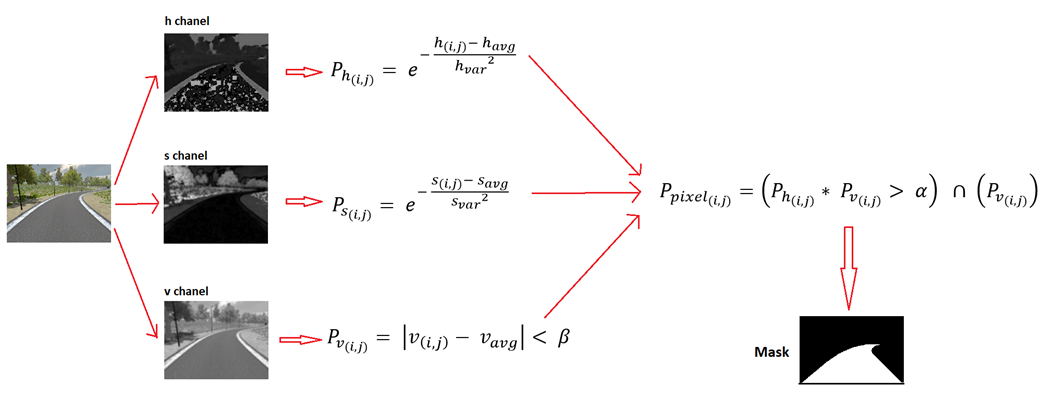


Figure 2. Tính toán độ hợp lý của từng pixel so với mẫu

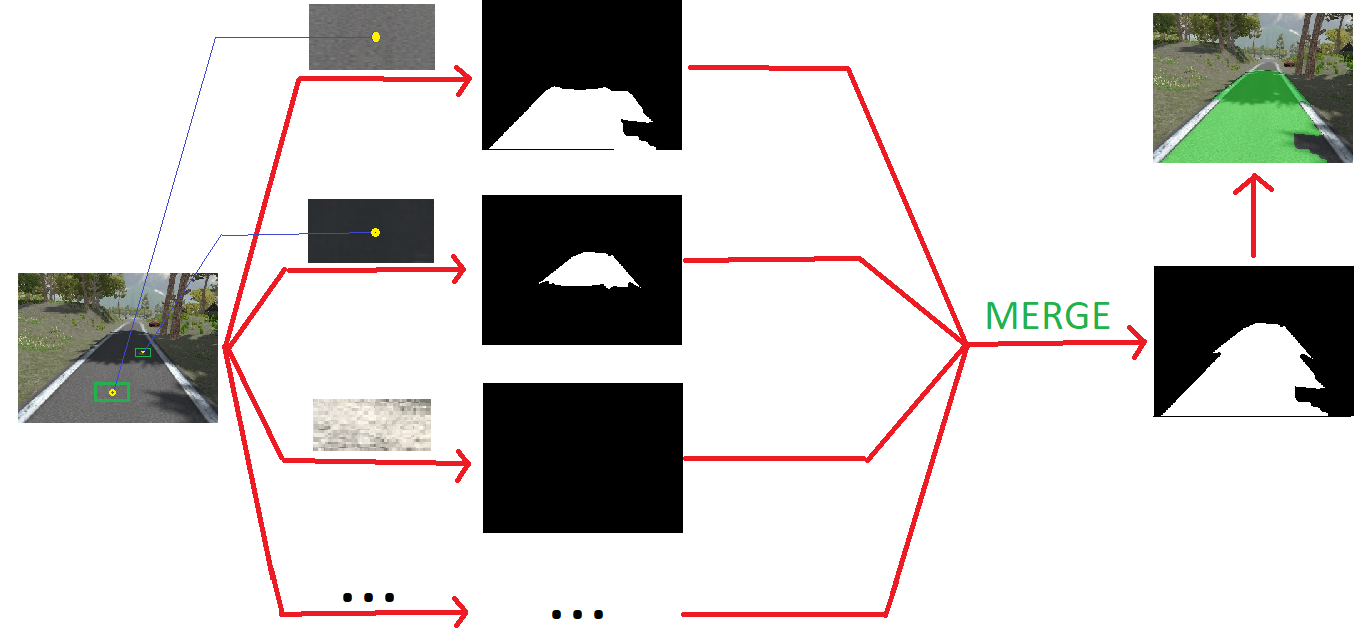


Figure 3. Tổng quát với nhiều mẫu đường thu thập đường dùng làm mẫu nhận diện

***Ưu điểm:*** Nhận diện được đường với độ chính xác cao hơn phương pháp dùng ngưỡng cố định, ít bị nhận sai background và nhận diện được các ảnh có độ phân giải thấp

***Khuyết điểm:*** Tốn công sức lập trình, ngưỡng để xét độ hợp lý (likelihood) của pixel so với mẫu cần tính thử nghiệm để tìm ra ngưỡng phù hợp, phụ thuộc vào phân bố các giá trị màu của các kênh màu HSV.

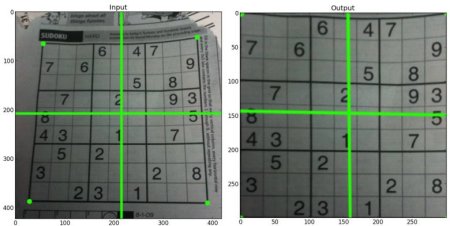
* ***So sánh kết quả nhận diện giữa hai phương pháp (Ảnh dùng để thử nghiệm thu thập được từ giả lập, ảnh tự chụp và mạng internet)***

|  |  |  |
| --- | --- | --- |
| **Ảnh đầu vào** | **Nhận diện bằng ngưỡng màu thông thường** | **Nhận diện bằng cách tính độ hợp lý (likelihood) của từng pixel so với mẫu** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | C:\Users\Lu Thai Hoc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\thresh.jpg | C:\Users\Lu Thai Hoc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\avg_var.jpg |
|  |  |  |
|  |  | C:\Users\Lu Thai Hoc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\avg_var.jpg |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. **Perspective Transformation**

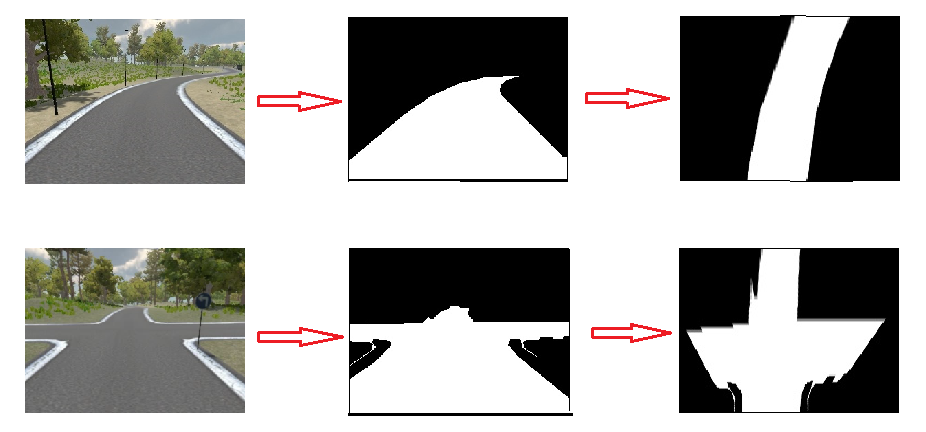
Trong Computer Vision, **Perspective Transformation** là thuật ngữ dùng để chỉ sự ánh xạ giữa các điểm trên hai mặt phẳng ảnh mà tương ứng đến cùng vị trí trên một đối tượng phẳng trong thế giới thực.

OpenCV cung cấp function **cv2.getPerspectiveTransform()** để tạo ma trận chuyển đổi và function **cv2.warpPerspective()** để thực hiện phép biến đổi ma trận.



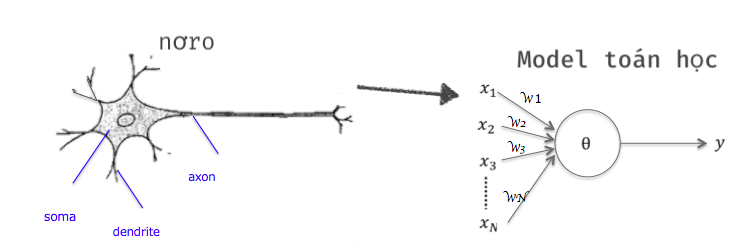
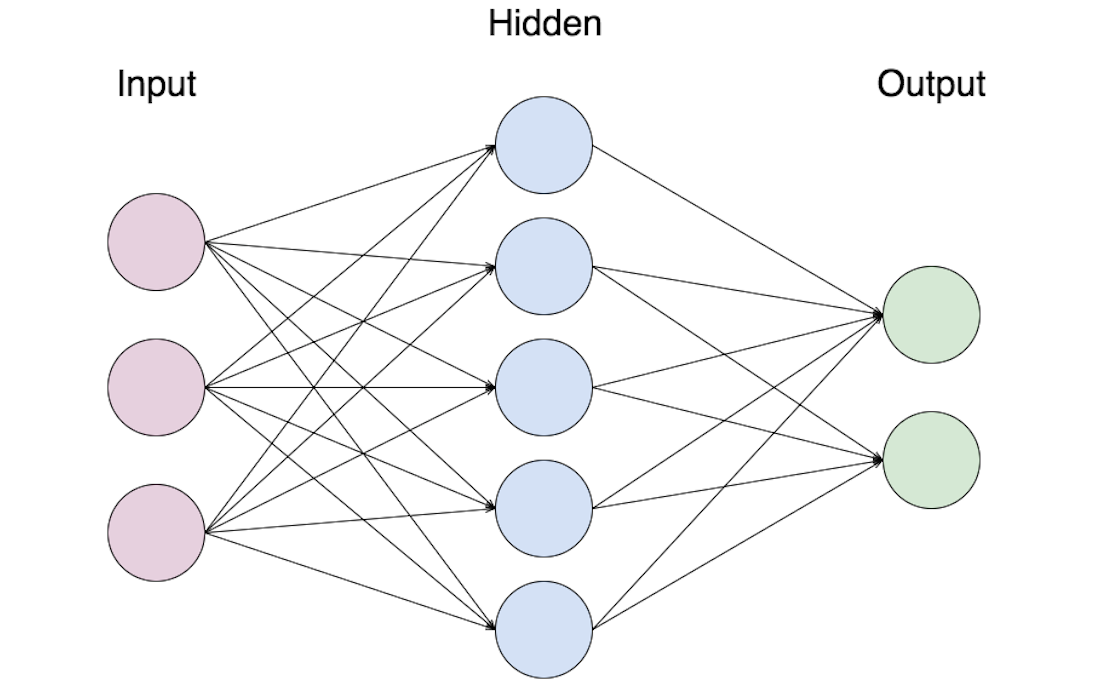
Kết quả phép Perpective Projection

Áp dụng trong lập trình xe tự hành, khi đã nhận diện được đường, ta sử dụng **Perspective Transformation** để chuyển đổi mask phần đường nhận diện được thành mask dạng Birdview (Birdview Transformation), đầu vào cho các thuật toán điều khiển sau này.



Birdview Transformation

## **Xây dựng mô hình mạng Neural Network để nhận diện biển báo**

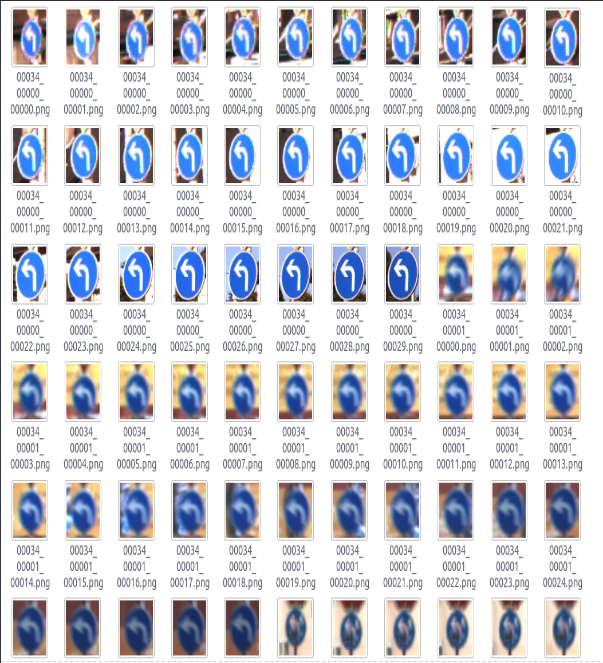
* 1. **Tổng quan về mô hình CNN:**
* ****Neural là mô hình toán học mô phỏng nơron trong hệ thống thần kinh con người. Model đó biểu hiện cho một số chức năng của neuron thần kinh con người
* ****Neural network là sự kết hợp của nhiều neural:
* Trong một Neural Network thông thường thì có 3 lớp chính:
  + Lớp input: Đây là lớp sẽ được đưa vào model, số lượng neural ở lớp này tượng trưng cho tổng số đặc trưng của dữ liệu
  + Lớp hidden: Lớp input sẽ được đưa vào lớp hidden, có thể có nhiều lớp hidden dựa trên kích thước model và dữ liệu. Mỗi lớp hidden sẽ có thể có số lượng neural khác nhau. Đầu ra của mỗi lớp hidden được thực hiện bằng phép nhân ma trận của lớp trước nó với các tham số học của lớp hiện tại và cộng thêm sai số dựa trên hàm kích hoạt.
  + Lớp output: Đầu ra của lớp hidden sẽ được đưa vào một hàm logistic ví dụ như sigmoid, softmax,.. Từ đó sẽ đưa ra xác suất của mỗi lớp đầu ra.
* Các kiểu khác nhau của Neural Networks được sử dụng với những mục đích khác nhau. Trong đó, CNN được dùng chủ yếu cho việc phân loại ảnh
  1. **Xây dựng mô hình CNN trong phân loại biển báo:**
     1. **Mô hình xây dựng:**
* Trong bài toán này, ta sẽ xây dựng mô hình để phân loại 3 loại biển báo là: Rẽ trái, rẽ phải và dừng lại.

****

* + 1. **Chi tiết mô hình:**
* Đầu vào của model là hình ảnh biển báo có kích thước: 64x64x3
* Qua 3 lớp Conv + Maxpool với kích thước cụ thể:
  + Conv1: f = 5, s = 2, p = 0, 8 filters
  + Maxpool1: f = 2
  + Conv2: f = 3, s = 1, p = 0, 16 filters
  + Maxpool2: f = 2
  + Conv3: f = 3, s = 1, p = 0, 32 filters
  + Maxpool3: f = 2

Mỗi Conv đều sử dụng hàm kích hoạt LeakyReLU với alpha = 0.1

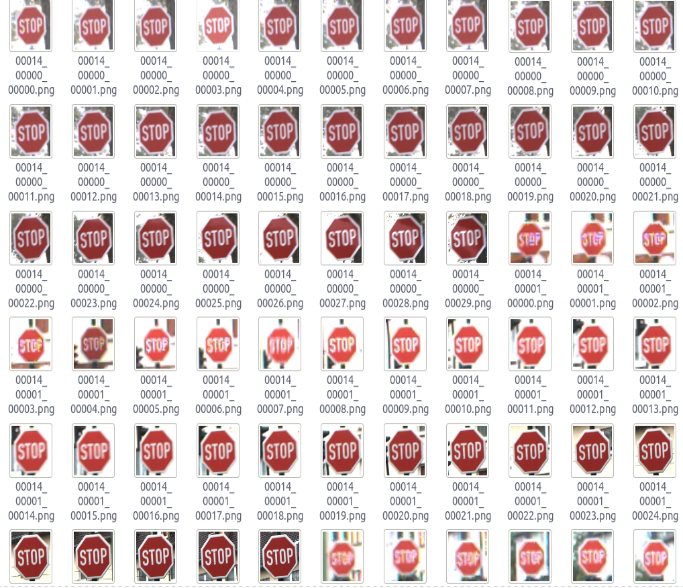
* Sử dụng lớp Flatten để trải phẳng ma trận đầu ra sau các lớp Conv + maxpool
* Sử dụng Drop out với tỉ lệ 0,3
* Xây dựng lớp Dense với 2 neural
* Sử dụng hàm kích hoạt sigmoid
* Sử dụng thuật toán tối ưu Adam với hệ số học = 0.001
* Hàm mất mát được sử dụng là binary\_crossentropy
* Kết quả đầu ra là 3 lớp: Left, Right và Stop.
  + 1. **Dữ liệu**
* Chia 3 loại biển báo cần phân loại ra thành 3 thư mục khác nhau, mỗi thư mục gồm hơn 600 ảnh của mỗi loại:

****

Left sign



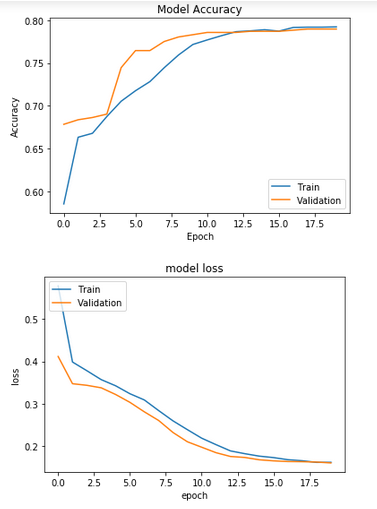
Right sign



*Stop sign*

* 1. **Đánh giá dựa trên mô hình và kết quả:**

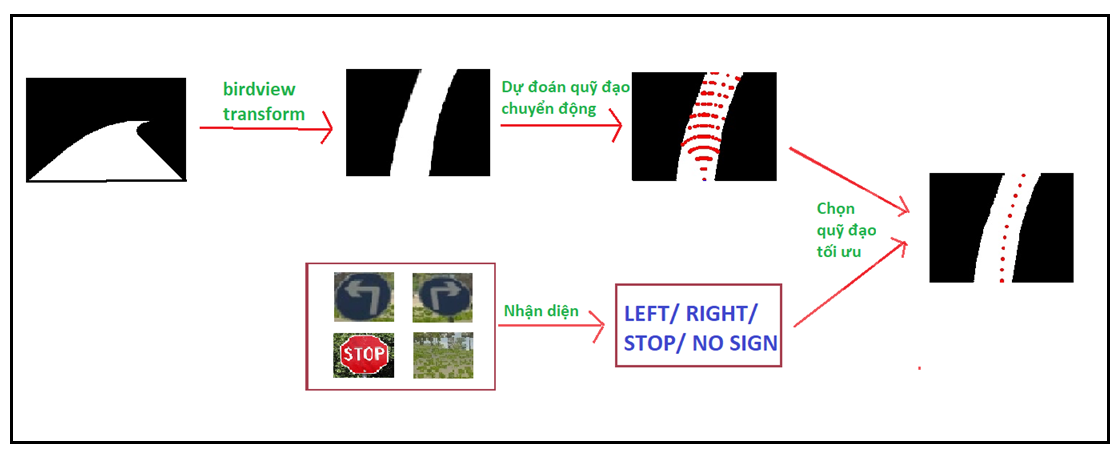
**Accuracy và loss của model:**

****

* Sau 30 epoch:
  + Tỉ lệ chính xác của model sau khi train là 79%, trên tập validation là 78%
  + Loss của model sau khi train là 0.16, trên tập validation là 0.15

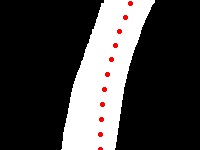
## **Nghiên cứu xây dựng chương trình điều khiển xe bằng phương pháp tìm quỹ đạo tối ưu và PID**

Sau khi xử lý ảnh đầu vào qua camera thu được ảnh birdview, ta tiến hành tìm hướng đi tối ưu cho xe bằng cách đặt quỹ đạo xe vào ảnh birdview thu được và chọn quỹ đạo tối ưu nhất.



* 1. **Phương trình quỹ đạo chuyển động của xe**

Vị trí của xe sau khi di chuyển trong thời gian ***t***, với vận tốc ***v*** và bán kính quỹ đạo ***r***:



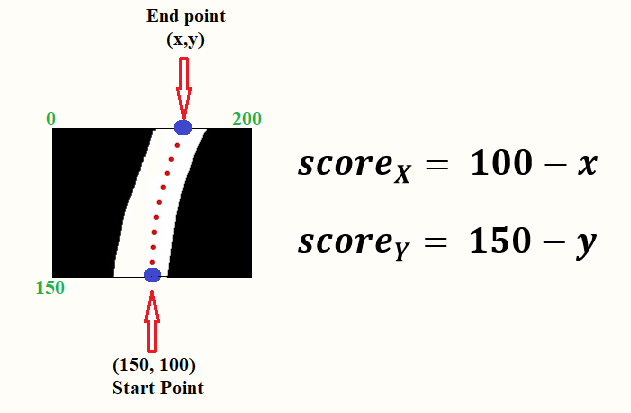
Quỹ đạo của xe trên ảnh birdview thu được

Bản chất của phương trình quỹ đạo xe chính là phương trình quỹ đạo di chuyển của một chất điểm trên đường tròn có bán kính ***r*** với vận tốc ***v*** và sau các khoảng thời gian ***t***. Ta tìm được và áp dụng phương trình này làm quỹ đạo của xe vì do khi ở vị trí hiện tại, nếu ta truyền cố định cho xe góc điều khiển ***a***, thì sau một thời gian ***t*** thì quỹ đạo xe vẽ ra sẽ là cung tròn. Góc điều khiển ***a*** càng nhỏ thì bán kính ***r*** càng lớn, khi ***a*** 🡪 0 thì ***r*** 🡪 ***∞,*** quỹ đạo của xe sẽ là đường thẳng.

Quỹ đạo của xe bắt đầu từ tâm chính giữa phía dưới của ảnh (do camera đặt chính giữa xe và phần đường phía dưới cùng nhìn thấy được chính là vị trí của xe) và kết thúc xe đi vào vùng tối (vùng không phải đường, nghĩa là quỹ đạo xe đi hết được vùng đường nhìn thấy được hoặc quỹ đạo đi vào hai bên lề đường).

* 1. **Hàm tối ưu chọn hướng điều khiển xe**

Hàm tối ưu sẽ tính điểm của quỹ đạo dựa trên quãng đường đi được (theo trục X và trục Y) của quỹ đạo và kết quả nhận diện biển báo hiện tại.



Điểm quỹ đạo đạt được theo trục X và Y

Điểm cuối cùng quỹ đạo đạt được sẽ được tính trong các trường hợp:

* Trường hợp không có biển báo xe đi thẳng theo đường:
* Trường hợp có biển báo rẽ trái:
* Trường hợp có biển báo rẽ phải:

*Với , là hệ số tỷ lệ*

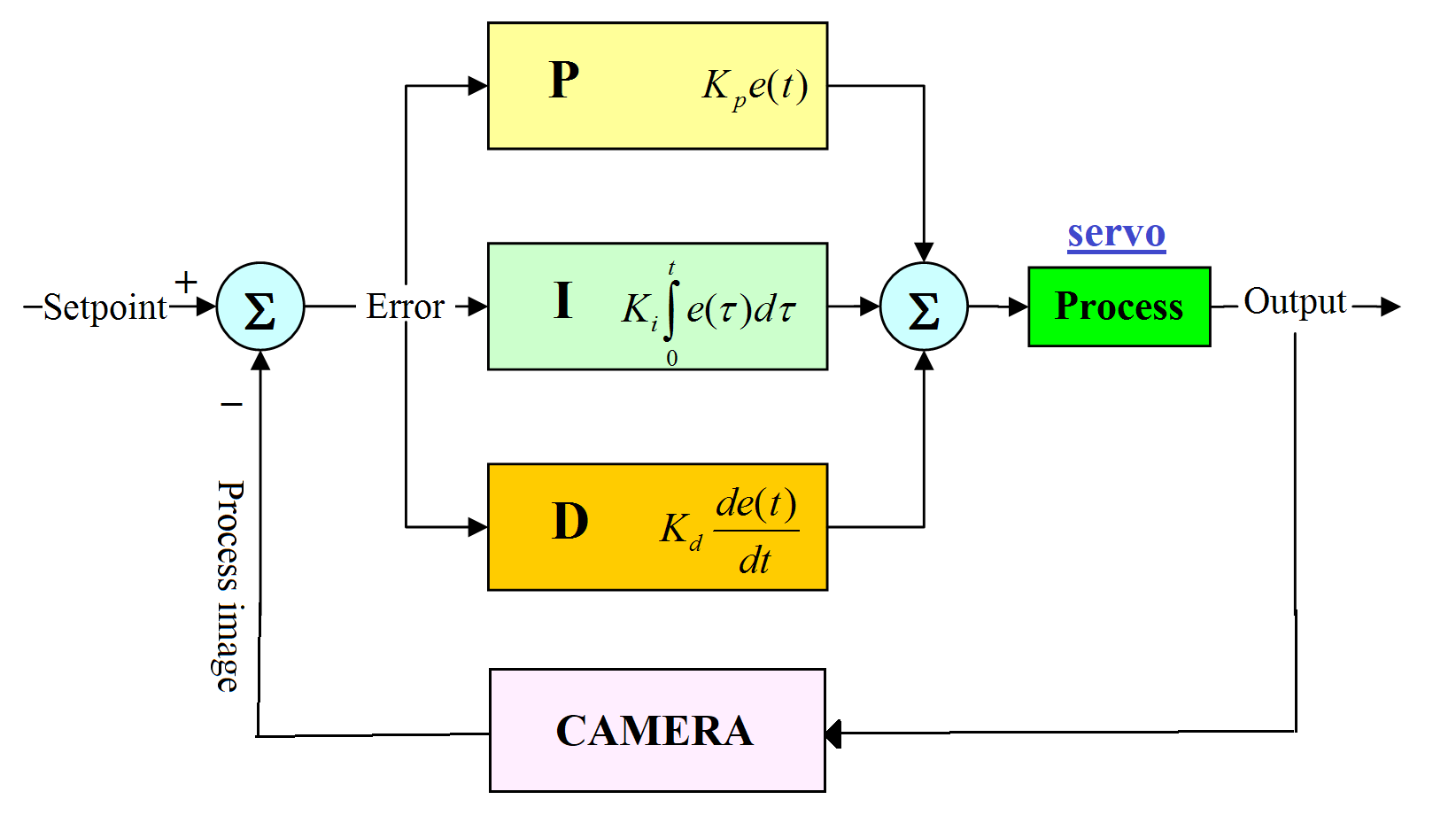
Chọn quỹ đạo có điểm (score) lớn nhất làm quỹ đạo tối ưu

* 1. **Xử lý đưa ra tín hiệu điều khiển bằng bộ điều khiển PID**

PID là sự kết hợp của 3 bộ điều khiển: **tỉ lệ**, **tích phân** và **vi phân**, có khả năng điều chỉnh sai số thấp nhất có thể, tăng tốc độ đáp ứng, giảm độ vọt lố, hạn chế sự dao động.

Điều khiển PID là một kiểu điều khiển có hồi tiếp vòng kín được sử dụng rộng rãi trong hệ thống điện, tự động hóa, điện tử,…

Áp dụng bộ điều khiển PID trong tự động điều khiển xe tự hành giúp quá trình di chuyển xe ổn định hơn, các giá trị điều khiển qua bộ điều khiển PID giúp xe nhanh chóng về vị trí mong muốn cũng như giúp hạn chế tình trạng dao động liên tục quanh setpoint.

****

Sơ đồ PID đối với mô hình xe thử nghiệm

**Giải thích sơ đồ:**

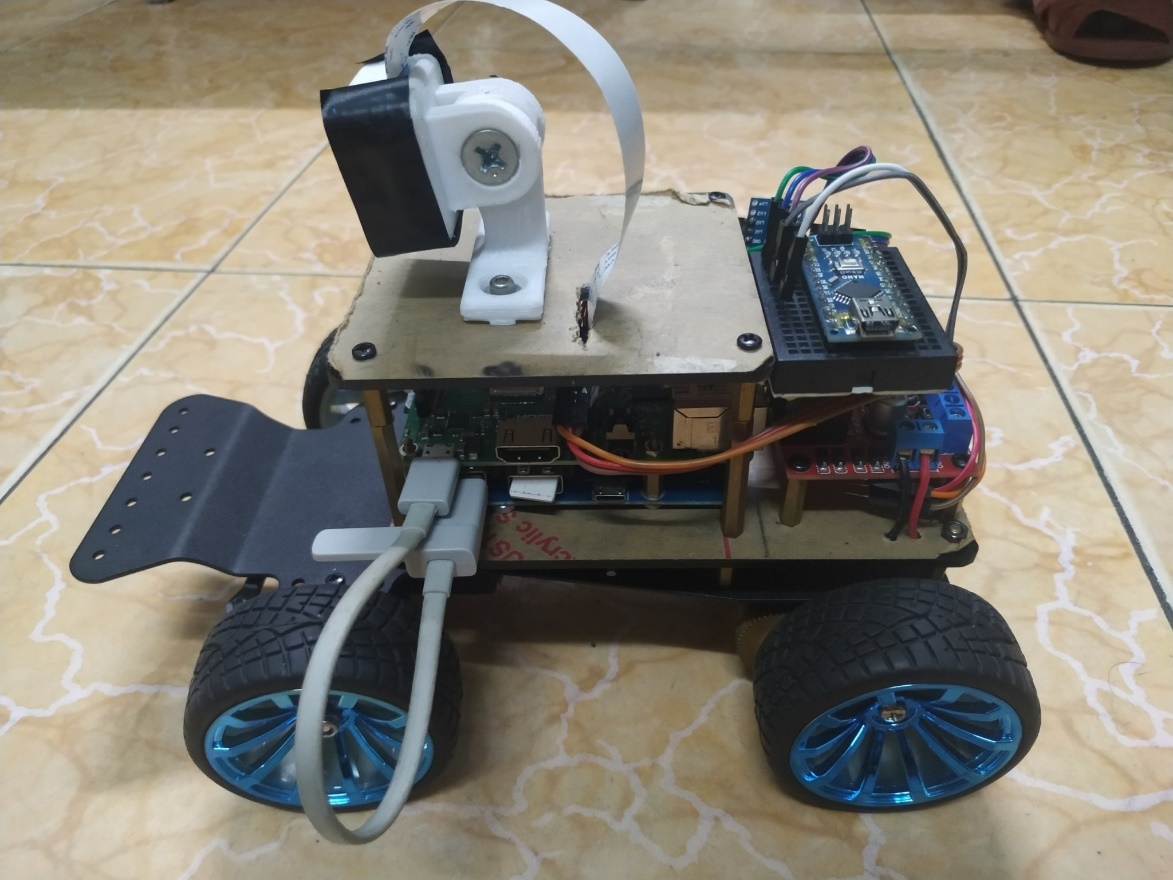
- Setpoint: là giá trị góc truyền vào servo để xe đi thẳng với góc 0 độ. (Vì quá trình điều khiển luôn mong muốn đưa xe về thẳng hướng với đường và di chuyển góc 0 độ). Đối với mô hình xe nhóm đã xây dựng được có setpoint = 90 độ (truyền góc 90 độ vào servo xe sẽ đi thẳng).

- Process image: Quá trình xử lý ảnh và đưa ra giá trị góc của quỹ đạo tối ưu ***a*** (góc so với hướng xe hiện tại)

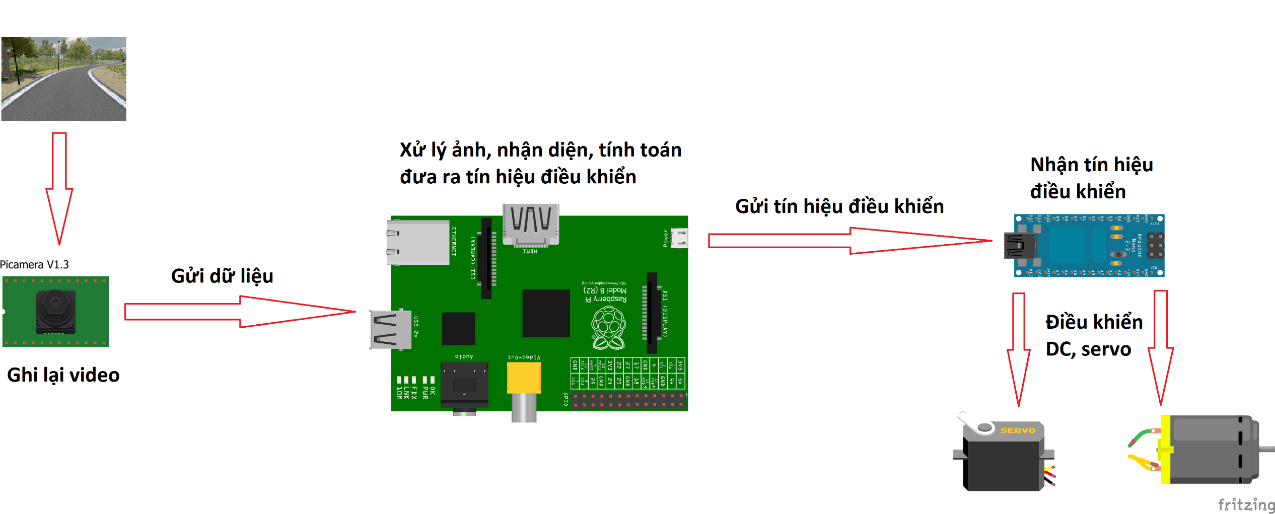
- Giá trị góc điều khiển Error = setpoint – ***a***

## **4. Xây dựng mô hình xe tự hành thử nghiệm sử dụng vi mạch Raspberry và Arduino**

Nhóm đề tài đã nghiên cứu xây dựng mô hình xe sử dụng vi mạch Raspberry và Arduino, điều khiển hướng xe bằng servo MG996 và di chuyển bằng động cơ giảm tốc 12V.



Mô hình xe xây dựng được



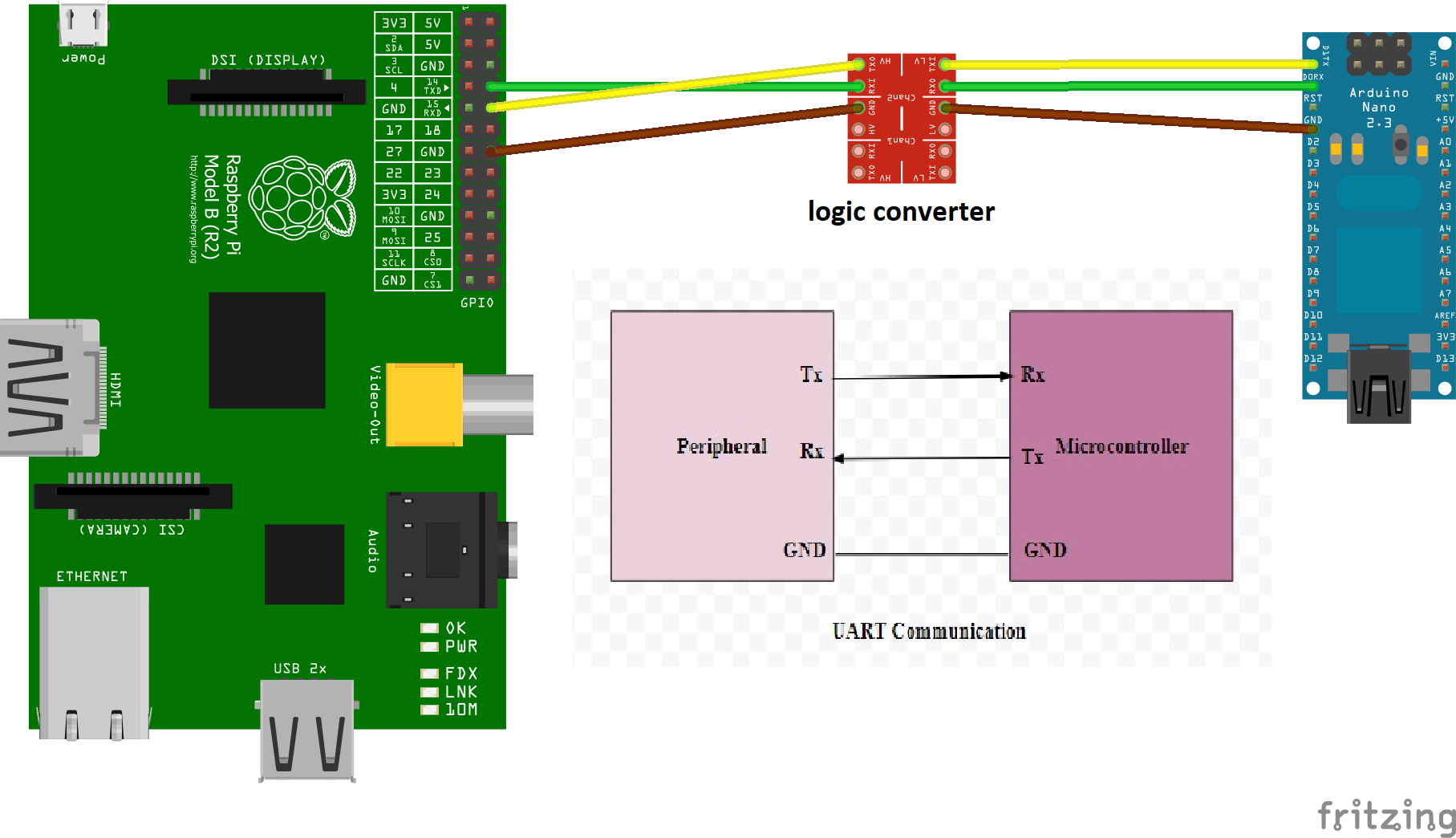
Chức năng của các bộ phận chính

* **Giao tiếp UART giữa Raspberry và Arduino**

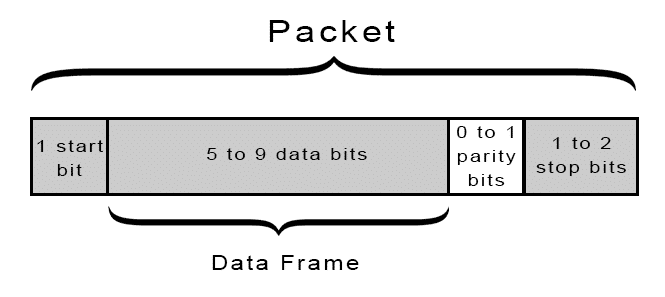
Bản thân mạch Raspberry cũng có các chân điều khiển có thể dùng để tạo xung đầu ra để điều khiển động cơ và servo. Nhưng vấn đề xảy ra là khả năng điều khiển động cơ, servo của Raspberry không thực sự tốt và Raspberry còn phải đọc và xử lý ảnh từ camera, tính toán đưa ra giá trị điều khiển cho động cơ và servo. Việc chỉ sử dụng mạch Raspberry để vừa thực hiện quá trình xử lý ảnh vừa điều khiển làm cho hệ thống không đáp ứng được thời gian thực, do đó chúng ta thêm vào một mạch Arduino để điều khiển động cơ, servo, giảm tải khối lượng công việc cho Raspberry, hơn nữa Arduino là một vi điều khiển được thiết kế với mục đích chính cho việc điều khiển, giao tiếp với các thiết bị và cảm biến, nên khả năng điều khiển động cơ và servo tốt hơn rất nhiều so với Raspberry.

Do đó, Raspberry sẽ đảm nhiệm chức năng chính là xử lý ảnh, nhận diện và tính toán đưa ra giá trị điều khiển. Sau đó sẽ truyền giá trị điều khiển này (Góc quay, tốc độ động cơ) cho Arduino, Arduino nhận tín hiệu và tạo xung điều khiển, điều khiển động cơ và servo. Việc truyền tín hiệu giữa Arduino và Raspberry được thực hiện bằng chuẩn giao tiếp UART.

UART (Universal Asynchronous Receive/Transmit) là chuẩn giao tiếp truyền nhận dữ liệu không đồng bộ. Đây là chuẩn giao tiếp phổ biến và dễ sử dụng , thường dùng trong giao tiếp giữa vi điều khiển với nhau hoặc với các thiết bị khác.



Sơ đồ giao tiếp giữa Raspberry và Arduino



Dữ liệu truyền qua UART

* Baudrate: Số bit truyền được trong 1s.
* Packet: quy định số bit truyền trong mỗi lần truyền, bit bắt đầu “Start bit”, các bit kết thúc “Stop bit”.
* Start bit: báo hiệu quá trình truyền dữ liệu.
* Data bits: dữ liệu cần giao tiếp, thường là 8 bit.
* Parity bit: bit kiểm tra chẵn lẻ, dùng để phát hiện lỗi.
* Stop bit: báo hiệu kết thúc một frame dữ liệu.

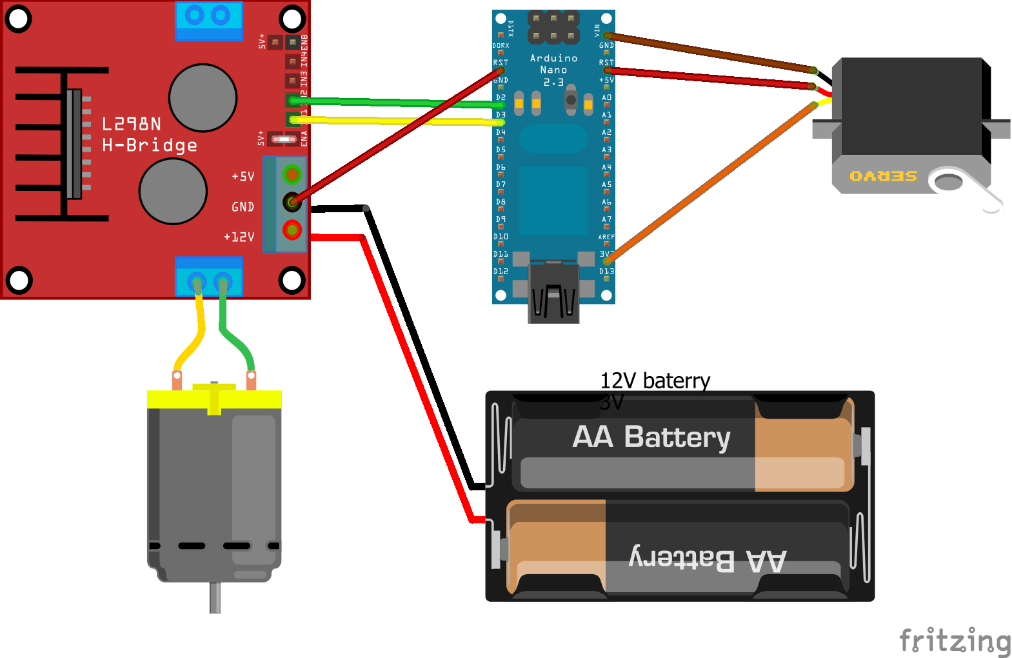
Quy ước chuỗi ký tự chứa giá trị điều khiển được truyền từ Raspberry sang Arduino:

* Điều khiển động cơ: “**M-F-Speed**” hoặc “**M-B-Speed**”. Trong đó: Ký tự ‘**M**’ là tín hiệu yêu cầu điều khiển động cơ, ký tự ‘**F**’, ‘**B**’ là tín hiệu điều khiển chiều của động cơ (forward hoặc backward), “**Speed**” là giá trị tốc độ động cơ.
* Điều khiển servo: “**S-Angle**”. Trong đó: Ký tự ‘**S**’ là tín hiệu yêu cầu điều khiển servo, “**Angle**” là góc quay của servo.

Mỗi chuỗi ký tự (string) gửi đi để điều khiển sẽ kết thúc bằng ký tự ‘**\r**’

* **Giao tiếp giữa Arduino với servo, motor**

Sau khi nhận được chuỗi ký tự (string) từ Raspberry, Arduino sẽ tiến hành đọc giá trị đầu tiên của chuỗi để xác định là chuỗi yêu cầu điểu khiển bộ phận nào sau đó tiếp tục tách chuỗi ký tự để lấy ra các giá trị điều khiển nằm ở phần sau của chuỗi, từ đó điều khiển các phần cứng theo mong muốn



Sơ đồ mạch kết nối giữa Arduino và DC, servo

# **PHẦN 2. CÀI ĐẶT CHƯƠNG TRÌNH VÀ CHẠY THỬ NGHIỆM**

1. **Xây dựng chương trình**

Chương trình chính được lập trình trên Raspberry với hệ điều hành Raspian.

Sử dụng ngôn ngữ lập trình python, thư viện chính là numpy và OpenCV 3.2

Cấu trúc chương trình chia làm 3 thread chính:

* Thread “get\_frame” : Đọc video từ camera
* Thread “main” : Chạy các module nhận diện đường, tính toán hướng tối ưu cho xe, cập nhật tín hiệu điều khiển
* Thread “show” : Hiển thị các hình ảnh kết quả

Các thread giao tiếp với nhau thông qua biến chung hệ thống, sử dụng thư viện config của python

1. **Chạy thử nghiệm trên giả lập *(có video kèm theo)***

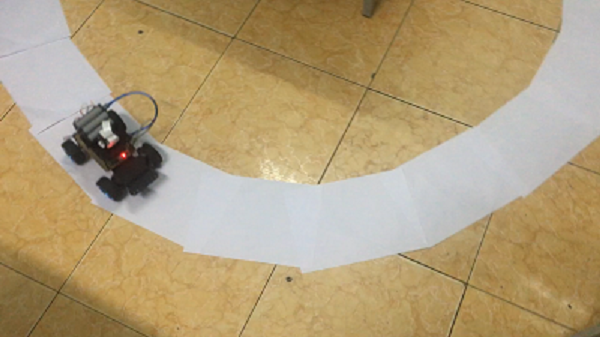
Xe nhận diện được và chạy theo đường, né được vật cản, nhận diện và rẽ trái/phải khi gặp biển báo.



Chạy thử nghiệm trên giả lập

1. **Chạy thực tế với mô hình xe *(có video kèm theo)***

Do hạn chế về mặt phần cứng (mạch raspberry không đáp ứng được thời gian thực khi sử dụng thư viện keras để nhận diện biển báo bằng mạng CNN đã xây dựng, nên mô hình chỉ mới chạy được với chức năng xử lý ảnh nhận diện đường và chương trình tính toán điều khiển xe theo đường có sẵn)



Chạy thử nghiệm xe mô hình

# **PHẦN 3. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

1. **Kết luận**

* ***Về mặt lý thuyết***
* Tìm hiểu và sử dụng lập trình OpenCV trong nhận diện đường và tạo ảnh Birdview áp dụng trong xe tự hành
* Nắm được lý thuyết về học máy, mạng CNN, xây dựng được mạng CNN để nhận diện biển báo
* Tìm hiểu lý thuyết điều khiển, áp dụng bộ điều khiển PID vào điều khiển xe tự hành
* Tìm hiểu về lập trình IoT, xây dựng mô hình, kết nối, điều khiển giữa phần cứng với phần mềm, tương tác giữa các phần cứng với nhau
* Tìm hiểu và lập trình trên Raspberry, xây dựng chương trình ứng dụng đa luồng để tăng hiệu suất chương trình đáp ứng thời gian thực
* ***Về mặt thực tiễn***

Về cơ bản, đề tài đã hoàn thành được các chức năng đã đề ra ban đầu như:

* Nhận diện được đường, biển báo
* Dự đoán quỹ đạo xe, xây dựng được hàm tối ưu để tìm quỹ đạo tối ưu
* Xây dựng được mô hình xe thử nghiệm sử dụng vi mạch Raspberry và Arduino
* Áp dụng được các công nghệ, thuật toán đã nghiên cứu vào xây dựng chương trình điều khiển xe tự hành
* Chạy thử nghiệm trên giả lập và chạy thực tế với xe mô hình, xe đi theo đường, né vật cản, nhận diện biển báo, rẽ trái, phải khi gặp biển báo

1. **Hạn chế**

Bên cạnh những kết quả đạt được, đề tài vẫn còn nhiều hạn chế như:

* Thuật toán nhận diện đường còn đơn giản, còn phụ thuộc nhiều vào màu sắc của đường, hiệu suất chưa cao
* Mô hình mạng CNN nhận diện biển báo với độ chính xác chưa cao (80%), chưa xây dựng mô hình mạng CNN để nhận diện nhiều loại biển báo.
* Tốc độ xử lý của chương trình còn chậm khi sử dụng nhiều mẫu đường làm mẫu để nhận diện
* Hạn chế về mô hình xe: mạch điều khiển chính Raspberry không đáp ứng được thời gian thực khi nhận diện biển báo, nên vẫn chưa áp dụng được các kĩ thuật học máy vào mô hình khi chạy thử nghiệm, xe chỉ mới áp dụng được các thuật toán xử lý ảnh và điều khiển để chạy bám theo đường

1. **Hướng phát triển**

Để cải thiện và nâng cao tính thông minh trong bài toán xây dựng và áp dụng công nghệ vào xe tự hành, nhóm cần tìm hiểu thêm các công cụ mới mạnh mẽ hơn và xây dựng các thuật toán tối ưu hơn:

* Nghiên cứu nhận diện đường bằng deep learning, tìm hiểu sử dụng semantic segmentation để nhận diện đường đạt hiệu quả cao hơn
* Cải thiện mô hình nhận diện biển báo để nhận diện với hiệu suất cao hơn và nhận được nhiều loại biển báo
* Cải thiện hàm tối ưu để tính toán đưa ra quỹ đạo tốt hơn
* Tìm hiểu sử dụng cảm biến khoảng cách Lidar và các cảm biến khác áp dụng trong xe tự hành
* Cải tiến phần cứng, sử dụng vi mạch xử lý mạnh mẽ hơn để đáp ứng các thuật toán học máy, cải thiện camera, động cơ,…

# **PHẦN 4. PHỤ LỤC**

**Các tài liệu tham khảo:**

Alexander Mordvintsev & Abid K. Revision. (2013). Geometric Transformation of Image. Truy xuất từ https://opencv-python-tutroals.readthedocs.io/

Basic of UART commuication. Truy xuất từ https://www.circuitbasics.com/

Joe Minichino & Joseph Howes (2015). Learning OpenCV 3 Computer Vision with Python. Second edition. Packt Publishing.

Normal distribution. Truy xuất từ https://en.wikipedia.org/wiki/

PID Controller. Truy xuất từ https://en.wikipedia.org/wiki/

Thresholding Operations using inRange từ https://docs.opencv.org/

Vikas Gupta. (2017). Color spaces in OpenCV (C++, Python). Truy xuất từ https://www.learnopencv.com/