

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT VĨNH LONG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CNTT1

**ỨNG DỤNG XỬ LÝ ẢNH TRONG VIỆC XÁC ĐỊNH
LÀN ĐƯỜNG DÀNH CHO XE TỰ HÀNH**

CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên thực hiện: Nguyễn Bá Anh Hòa

MSSV: 18004038

Lớp: ĐH.CÔNG NGHỆ THÔNG TIN 1CTT18A1

Khóa: 2018-2022

Người hướng dẫn: Th.S Nguyễn Văn Hiếu

Vĩnh Long, tháng 6/2020



TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT VĨNH LONG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CNTT1

**ỨNG DỤNG XỬ LÝ ẢNH TRONG VIỆC XÁC ĐỊNH
LÀN ĐƯỜNG DÀNH CHO XE TỰ HÀNH**

CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên thực hiện: Nguyễn Bá Anh Hòa

MSSV: 18004038

Lớp: ĐH.CÔNG NGHỆ THÔNG TIN 1CTT18A1

Khóa: 2018-2022

Người hướng dẫn: Th.S Nguyễn Văn Hiếu

Vĩnh Long, tháng 6/2020

Đơn đăng ký

TRƯỜNG ĐẠI HỌC SPKT VĨNH LONG
KHOA CÔNG NGHỆ THÔNG TIN

PHIẾU GIAO ĐỒ ÁN CÔNG NGHỆ THÔNG TIN 1

Tên đồ án: Ứng dụng xử lý ảnh trong việc xác định làn đường dành cho xe tự hành

Nhiệm vụ:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Phương pháp đánh giá: ☐ Báo cáo trước hội đồng ☐ Chấm thuyết minh

Ngày giao đồ án: ngày 18 tháng 3 năm 2020

Ngày hoàn thành đồ án: ngày 22 tháng 6 năm 2020

Số lượng sinh viên thực hiện đồ án: 1

Họ và tên sinh viên: Nguyễn Bá Anh Hào MSSV: 18004038

Vĩnh Long, ngày 18 tháng 3 năm 2020

Khoa CNTT

Người hướng dẫn

ThS. Nguyễn Văn Hiếu

NHẬN XÉT VÀ ĐÁNH GIÁ ĐIỂM CỦA NGƯỜI HƯỚNG DẪN

- Ý thức thực hiện:

.....

.....

.....

.....

.....

- Nội dung thực hiện:

.....

.....

.....

.....

.....

- Hình thức trình bày:

.....

.....

.....

.....

.....

- Tổng hợp kết quả:

.....

.....

☐ Tổ chức báo cáo trước hội đồng

☐ Tổ chức chấm thuyết minh

Vĩnh Long, ngày ... tháng ... năm ...

Người hướng dẫn

ThS. Nguyễn Văn Hiếu

Vĩnh Long, ngày tháng năm

BẢNG ĐÁNH GIÁ CỦA CÁN BỘ HƯỚNG DẪN TỐT NGHIỆP

I. THÔNG TIN CHUNG

Họ tên: Học hàm, học vị:

Đơn vị công tác:

Hướng dẫn sinh viên:

Mã số sinh viên: Ngành:

Tên đề tài hướng dẫn:

II. NHẬN XÉT

1. Ưu điểm:

.....
.....

2. Hạn chế:

.....
.....

III. ĐÁNH GIÁ

TT	Nội dung đánh giá	Điểm tối đa	Điểm đánh giá
1	Hình thức trình bày quyền thuyết minh và hình vẽ (<i>theo quy định của nhà trường, không có lỗi chính tả, ngắn gọn, mạch lạc, súc tích, ...</i>)	2.0	
2	Ý thức và thái độ trong thực hiện đề tài	1.0	
3	Khả năng và bản lĩnh xử lý, giải quyết vấn đề của sinh viên trong thực hiện đề tài	1.0	

4	Thực hiện các nội dung của đề tài (<i>về nội dung chuyên môn và khoa học cũng như về phương pháp nghiên cứu, xử lý vấn đề của đề án, KLTN có gì đúng, sai, có gì mới, mức độ sáng tạo</i>)	3.0	
5	Mối liên hệ với những vấn đề liên quan (<i>cơ sở lý thuyết và các hướng nghiên cứu khác có liên quan</i>)	1.0	
6	Tính ứng dụng thực tiễn (<i>phạm vi và mức độ ứng dụng, triển vọng của đề tài, tính mới, tính sáng tạo</i>)	2.0	
Tổng số		10	

IV. KẾT LUẬN

.....

.....

NGƯỜI HƯỚNG DẪN

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Em xin chân thành cảm ơn đến tất cả các thầy cô trong khoa Công Nghệ Thông Tin đã hết lòng truyền đạt kiến thức và hướng dẫn tận tình để em hoàn thành đồ án này.

Đặc biệt, em chân thành cảm ơn thầy Nguyễn Văn Hiếu, thầy Trần Phan Trường, thầy Tô Nguyễn Hoàng Phúc đã tận tình hướng dẫn em làm đồ án. Thầy đã truyền đạt lại những kiến thức quý báu để giúp em hoàn thành đồ án.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, bạn bè, người thân và thành viên của cộng đồng diễn đàn xử lý ảnh đã hỗ trợ, động viên, giúp đỡ em rất nhiều trong quá trình học tập và thực hiện đồ án. Với điều kiện thời gian cũng như kiến thức còn hạn chế, đồ án này không thể tránh được những thiếu sót. Rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để em có điều kiện bổ sung, nâng cao khả năng và ý thức của mình.

Em xin chân thành cảm ơn!

Vĩnh Long, ngày... tháng... năm 2019

Sinh viên

Nguyễn Bá Anh Hào

MỤC LỤC

LỜI NÓI ĐẦU	1
CHƯƠNG 1: TỔNG QUAN.....	2
1.1 SƠ LƯỢC VỀ ĐỀ TÀI.....	2
1.2 MỤC ĐÍCH NGHIÊN CỨU	2
1.3 ĐỐI TƯỢNG NGHIÊN CỨU	2
1.4 PHẠM VI NGHIÊN CỨU.....	2
1.5 Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	4
2.1 NGÔN NGỮ LẬP TRÌNH PYTHON	4
2.1.1 Giới thiệu ngôn ngữ lập trình Python.....	4
2.1.2 Lịch sử phát triển của ngôn ngữ lập trình Python	4
2.1.3 Đặc điểm ngôn ngữ lập trình Python.....	4
2.2 Phần mềm JetBrains Pycharm	5
2.2.1 Giới thiệu Pycharm.....	5
2.2.2 Công cụ Remote Development	6
2.3 GIAO THỨC SSH.....	7
2.4 Giao thức VNC	9
2.5 Máy tính nhúng Raspberry Pi	10
2.5.1 Giới thiệu về Raspberry Pi	10
2.5.2 Hệ điều hành Raspbian.....	11
2.6 Thuật toán PID.....	12
2.7 THƯ VIỆN XỬ LÝ ẢNH OPENCV	13
2.7.1 Giới thiệu.....	13
2.7.2 Ứng dụng.....	13
2.7.3 Sử dụng một số hàm cơ bản với OpenCV.....	14
2.8 LÝ THUYẾT XỬ LÝ ẢNH.....	15
2.8.1 Ảnh kỹ thuật số.....	15
2.8.2 Độ phân giải ảnh.....	15
2.8.3 Điểm ảnh	15
2.8.4 Ảnh màu	16
2.8.5 Kênh màu HSV	16

2.8.6 Region of Interest	17
2.8.7 Lọc màu ảnh	18
2.8.8 Phát hiện đường thẳng bằng giải thuật Hough Transform (Hough Line)	19
CHƯƠNG 3: XÂY DỰNG THUẬT TOÁN VÀ MÔ HÌNH	20
3.1 Xử lý ảnh	20
3.1.1 Lấy mẫu thực nghiệm và các thông số đầu vào	20
3.1.2 Lọc nhiễu từ môi trường	21
3.1.3 Tạo vùng quan tâm (Region of Interest)	22
3.1.4 Chuyển ảnh sang tham số đường thẳng với giải thuật Hough Transform	22
3.1.5 Tính góc quay xe	24
3.1.6 Hiển thị đường thẳng điều hướng	24
3.2 CHI TIẾT IN 3D	25
3.2.1 Khung xe Donkey Cage	25
3.2.3 Cơ cấu đánh lái	26
3.3 SƠ ĐỒ MẠCH ĐIỆN	26
3.4 KẾT QUẢ PHÂN CỨNG	27
3.4.1 Cơ cấu đánh lái trên mô hình thực tế	27
3.4.2 Khối nguồn, khối điều khiển trên mô hình thực tế	28
3.4.3 Thân xe	29
CHƯƠNG 4: THỰC NGHIỆM	30
4.1 MÔI TRƯỜNG THỰC NGHIỆM	30
4.1.1 Môi trường vật lý	30
4.1.2 Môi trường phần mềm	30
4.2 KẾT QUẢ THỰC NGHIỆM	32
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	35
5.1 KẾT QUẢ ĐẠT ĐƯỢC, ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA ĐỀ TÀI	35
5.1.1 Kết quả đạt được	35
5.1.2 Ưu điểm	35
5.1.3 Nhược điểm	35
5.2 HƯỚNG PHÁT TRIỂN	35

DANH MỤC ẢNH

Hình 2.1 Bảng xếp hạng các phổ biến nhất thế giới 2019	5
Hình 2.2 Phần mềm Pycharm.....	6
Hình 2.3 Remote Development.....	7
Hình 2.4 Giao thử SSH trên Commandline	8
Hình 2.5 Phần mềm PuTTY.....	9
Hình 2.6 Giao thức VNC	9
Hình 2.7 VNC Viewer.....	10
Hình 2.8 Mạch Raspberry Pi.....	11
Hình 2.9 Giao diện Raspian	12
Hình 2.10 Thuật toán PID và thuật toán Bang Bang	13
Hình 2.11 Ứng dụng phát hiện vật thể.....	14
Hình 2.12 Hiển thị hình ảnh lên cửa sổ với OpenCV	15
Hình 2.13 Sự khác biệt giữa các độ phân giải.....	15
Hình 2.14 3 kênh màu	16
Hình 2.15 RGB sang HSV	17
Hình 2.16 Hình thường và sau khi áp dụng RoI	18
Hình 2.17 Áp dụng phân ngưỡng.....	18
Hình 2.18 Tìm đường thẳng	19
Hình 3.1 Nhiễu xuất hiện	20
Hình 3.2 Nhiễu không còn nữa	21
Hình 3.3 Áp dụng RoI.....	22
Hình 3.4 Donkey Cage.....	25
Hình 3.5 Đế bắt linh kiện	25
Hình 3.6 Đế bắt servo.....	26
Hình 3.7 Thanh đánh lái.....	26
Hình 3.8 Sơ đồ mạch điện.....	27
Hình 3.9 Hệ thống đánh lái	27
Hình 3.10 Khối nguồn và khối điều khiển	28
Hình 3.11 Thân trước xe	29
Hình 3.12 Bên hông cùng khối xử lý	29
Hình 4.1 Môi trường thử nghiệm	30
Hình 4.2 Nhập IP của Raspberry Pi	31
Hình 4.3 Giao diện và môi trường phần mềm điều khiển.....	31
Hình 4.4 Giao diện thực nghiệm	32
Hình 4.5 Xe đang queo cua	32
Hình 4.6 Xe trên đường thẳng.....	33
Hình 4.7 Chiếc búa xuất hiện nhưng không bị hiểu nhầm thành đường đi	34
Hình 4.8 Bàn tay không bị nhận nhầm.....	34

DANH MỤC BẢNG BIỂU

Bảng 2.1 Các phiên bản Python	4
Bảng 4.1 Các khó khăn và khả năng giải quyết	33

LỜI NÓI ĐẦU

Hiện nay xã hội đang phát triển với tốc độ chóng mặt, sự bùng nổ của “Cách mạng công nghiệp 4.0” được nhắc đi nhắc lại trên các kênh truyền thông. Xử lý ảnh, thị giác máy tính, Big Data, kết nối vạn vật hay trí tuệ nhân tạo là những lĩnh vực đi đầu trong cuộc cách mạng công nghiệp của loài người lần này. Việc cho máy tính có khả năng nhìn và hiểu là một mắt xích quan trọng trong tương lai khi mà xe tự lái, robot đang dần thay thế con người trong các công việc nguy hiểm. Do đó em cảm thấy mình phải trở thành một mắt xích trong cuộc cách mạng này, đề tài em chọn là Ứng dụng xử lý ảnh trong việc xác định làn đường dành cho xe tự hành. Là một đề tài yêu cầu nhiều về khả năng xử lý ảnh, lập trình nhúng. Bài toán sẽ là vấn đề về việc xử lý làn đường bằng OpenCV sau đó chuyển dữ liệu thô thành các vector chỉ đường để điều hướng cho xe.

Kết quả đạt được:

- Có thêm kiến thức về Python, OpenCV, lập trình nhúng
- Xe tự hành có thể tự di chuyển tốt
- Khả năng mở rộng đề tài trong tương lai lớn

Kết quả chưa đạt được:

- Thuật toán chưa hoàn hảo và cần được nâng cấp nhiều
- Cơ cấu xe còn chưa tốt
- Vấn đề nhận diện chưa thật sự hoàn hảo

CHƯƠNG 1: TỔNG QUAN

1.1 SƠ LƯỢC VỀ ĐỀ TÀI

Xử lý ảnh là một lĩnh vực phức tạp bắt đầu được quan tâm kể từ sau Chiến Tranh Thế Giới thứ 2 do máy tính phát triển nhanh. Đến Chiến Tranh Lạnh hay đúng hơn là vào cuộc đua vào vũ trụ của Soviet và Mỹ đến nay, xử lý ảnh được phát triển liên tục, hàng loạt thuật toán tiên tiến được bổ sung.

Lập trình những lúc đầu sử dụng ngôn ngữ Assembly sau này là C/C++ và khá khó để cho người mới bắt đầu, 10 năm trở lại đây với sự xuất hiện của Arduino đã trở nên dễ tiếp cận hơn nhờ vào giao diện thân thiện, cộng đồng lớn mạnh. Nhưng mọi thứ chỉ thật sự bùng nổ khi Raspberry Pi xuất hiện, một bo mạch máy tính nhỏ gọn và tiết kiệm năng lượng, kết hợp với nó là ngôn ngữ Python thân thiện dễ làm quen và có rất nhiều thư viện hỗ trợ. Bo mạch Raspberry Pi cung cấp khả năng xử lý tuyệt vời khi chạy các tác vụ xử lý ảnh và thường được ứng dụng vào công nghiệp như các dự án giám sát chuyển động, đếm số lượng xe, quản lý chống kẹt xe và xe tự hành.

Ứng dụng của xử lý ảnh hiện diện xung quanh chúng ta như định danh vân tay, quét mã vạch, phân loại sản phẩm, ứng dụng dịch văn bản bằng hình ảnh của Google Dịch, xe tự lái của Tesla, chuỗi phần mềm nổi tiếng của hãng Adobe.

Xử lý ảnh là một lĩnh vực tiềm năng trong tương lai, khi máy móc thay thế con người. Nắm được nguyên do do đó em chọn đề tài: “Ứng dụng xử lý ảnh trong việc xác định làn đường dành cho xe tự hành”.

1.2 MỤC ĐÍCH NGHIÊN CỨU

Mục đích nghiên cứu đề tài nhằm:

- Giúp cho người học hiểu rõ các nguyên lý cơ bản trong lĩnh vực xử lý ảnh.
- Có các kiến thức về lập trình nhúng.
- Hiểu hơn về ngôn ngữ Python.
- Đặc tính và khả năng của board mạch Raspberry Pi.
- Thư viện xử lý ảnh OpenCV.

1.3 ĐỐI TƯỢNG NGHIÊN CỨU

- Tất cả mọi người muốn tìm hiểu về xử lý ảnh
- Các lập trình viên muốn tìm hiểu thêm về xử lý ảnh, lập trình nhúng và ngôn ngữ lập trình Python.

1.4 PHẠM VI NGHIÊN CỨU

Phạm vi của đề án này trải từ phần cứng đến phần mềm

- Phần mềm:
- + Xử lý ảnh.
- + Thiết kế đồ họa 3D tạo hình cho khung xe.

- + Lập trình nhúng.
- + Kỹ thuật băm xung.
- + Điều tốc động cơ sử dụng thuật toán PID.
- + Ứng dụng kỹ thuật xây dựng từ xa (Remote Development).
- + Giao thức SSH.
- Phần cứng:
 - + Bo mạch xử lý Raspberry Pi.
 - + Xử lý tín hiệu.
 - + Cơ cấu điều khiển.
 - + Giao tiếp Camera.

1.5 Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN

Cùng với sự phát triển của phần cứng máy tính thì xử lý ảnh và đồ hoạ cũng đã phát triển mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Nó đóng vai trò quan trọng trong việc tương tác giữa người và máy. Do vậy đề án này mong muốn trình bày một phần nhỏ của kỹ thuật xử lý ảnh và lập trình nhúng cùng những tiềm năng không giới hạn của chúng trong tương lai không xa.

Trong thực tiễn đã có những hãng xe trang bị hệ thống giữ đúng làn đường (Lane Keepline System) hoặc hãng xe điện Tesla có khả năng tự lái (Auto Pilot) nhưng hầu như chưa có một hệ thống nào đủ tin tưởng hoặc hoàn hảo cho khả năng tự lái hoàn toàn (Fully Auto Pilot), hy vọng qua đề án này sẽ giúp cho người nghiên cứu và cả những lập trình viên có thể tìm thấy thuật toán tối ưu hơn để áp dụng trong tương lai không xa

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 NGÔN NGỮ LẬP TRÌNH PYTHON

2.1.1 Giới thiệu ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum. Nó dễ dàng để tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình. Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Python có cấu trúc dữ liệu cấp cao mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng. Cú pháp lệnh của Python là điểm cộng vô cùng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết script và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.

2.1.2 Lịch sử phát triển của ngôn ngữ lập trình Python

Kể từ khi xuất hiện đến nay ngôn ngữ lập trình Python luôn được cập nhật liên tục, được bổ sung các thư viện và các dự án mã nguồn mở do cộng đồng khổng lồ của mình đóng góp

Phiên bản	Ngày phát hành
Python 1.0 (bản phát hành chuẩn đầu tiên) Python 1.6 (Phiên bản 1.x cuối cùng)	01/1994 05/09/2000
Python 2.0 (Giới thiệu list comprehension) Python 2.7 (Phiên bản 2.x cuối cùng)	16/10/2000 03/07/2010
Python 3.0 (Loại bỏ cấu trúc và mô-đun trùng lặp) Python 3.6 (Tối ưu hóa tốc độ) Python 3.8 Phiên bản mới nhất hiện nay	03/12/2008 23/12/2016 14/10/2019

Bảng 2.1 Các phiên bản Python

2.1.3 Đặc điểm ngôn ngữ lập trình Python

Python là ngôn ngữ được sử dụng nhiều nhất hiện tại, ứng dụng lên hầu hết các lĩnh vực quan trọng.

Rank	Change	Language	Share	Trend
1		Python	29.49 %	+4.5 %
2		Java	19.57 %	-2.4 %
3		Javascript	8.4 %	+0.1 %
4		C#	7.35 %	-0.4 %
5		PHP	6.34 %	-1.2 %
6		C/C++	5.87 %	-0.4 %
7		R	3.82 %	-0.2 %
8		Objective-C	2.6 %	-0.7 %
9		Swift	2.57 %	-0.1 %
10		Matlab	1.87 %	-0.2 %

Hình 2.1 Bảng xếp hạng các phổ biến nhất thế giới 2019

Không phải ngẫu nhiên mà các lập trình viên đều thích chọn Python. Đây là một ngôn ngữ rất dễ sử dụng, cú pháp trong sáng ngắn gọn, dễ dàng chạy trên các hệ thống lớn nhỏ. Python trong lĩnh vực thiết kế web có framework Django, trong xử lý ảnh có thư viện đa ngôn ngữ OpenCV và hầu hết các ứng dụng sử dụng trí tuệ nhân tạo đều dùng Python để vận hành.

Thực hiện việc trao đổi giá trị trong C++ :

```
1. int temp;
2. temp = a;
3. a = b;
4. b = temp;
```

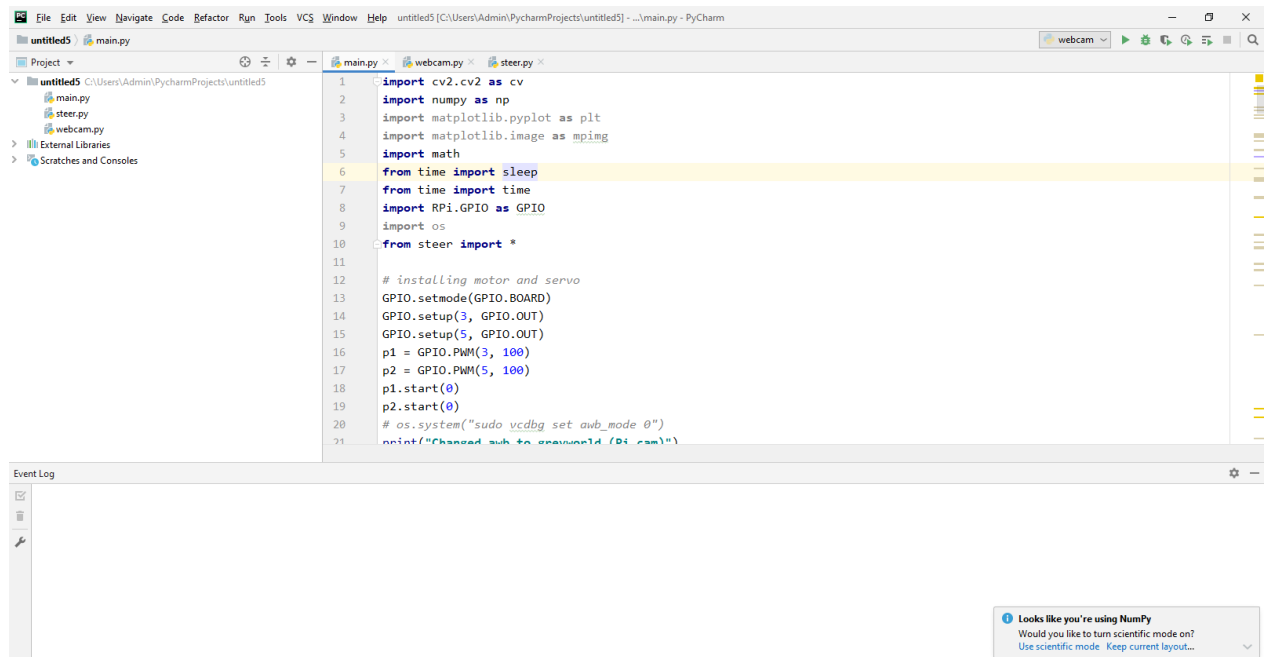
Thực hiện việc trao đổi giá trị trong Python:

```
a,b = b,a
```

2.2 Phần mềm JetBrains Pycharm

2.2.1 Giới thiệu Pycharm

Pycharm là môi trường phát triển tích hợp (IDE) mạnh mẽ nhất hiện tại dành cho ngôn ngữ Python. Nó được thiết kế để cung cấp tất cả các công cụ mà một lập trình viên có thể cần trong khi làm việc về phát triển Python.



Hình 2.2 Phần mềm Pycharm

- Ưu điểm:

+ Hỗ trợ tự động import.

+ Khả năng tự hoàn thành.

+ Nhắc lệnh tốt.

+ Nhiều extension tốt.

+ Cộng đồng lớn mạnh.

+ Tính năng Remote Development cho phép lập trình và chạy code từ xa, áp dụng tốt cho các phiên bản Window Server Core, Linux nonGUI.

- Nhược điểm:

+ Do nhiều tính năng dẫn đến việc nặng về phần cứng, các máy tính cấu hình thấp.

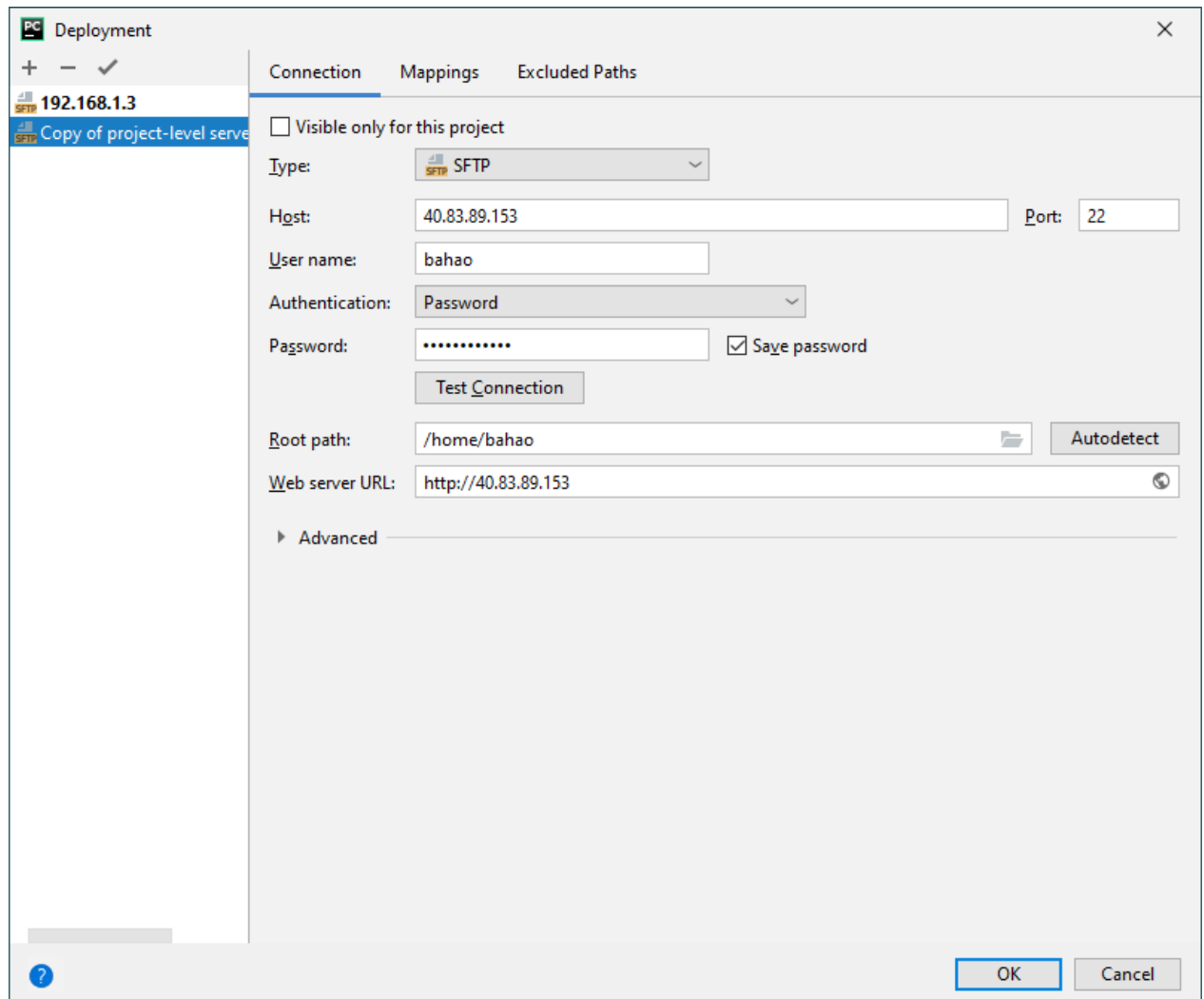
+ Các board mạch phát triển không có khả năng vận hành phần mềm này.

+ Giá thành đắt đỏ

2.2.2 Công cụ Remote Development

Công cụ Remote Development là giải pháp thay thế hoàn hảo cho việc phát triển phần mềm từ xa mà không phải nạp mã nguồn lại nhiều lần, cung cấp khả năng sửa lỗi và

kiểm thử từ xa, gần đây chức năng Remote Development đã được hãng JetBrains tích hợp vào phần mềm Pycharm.



Hình 2.3 Remote Development

Pycharm là một trong những IDE tiên trong hỗ trợ chức năng thông dịch từ xa, thư mục dự án sẽ được đồng bộ hóa (Sync) với thiết bị cần nạp mã nguồn. Tự động hóa việc lập trình từ xa hoàn toàn.

Đây là một ưu điểm khi sử dụng Pycharm so với các phần mềm khác trong việc lập trình nhúng, ta có thể lập trình Raspberry Pi mà không cần phải cắm dây hoặc sử dụng phần mềm soạn thảo Nano lỗi thời và rất khó sử dụng.

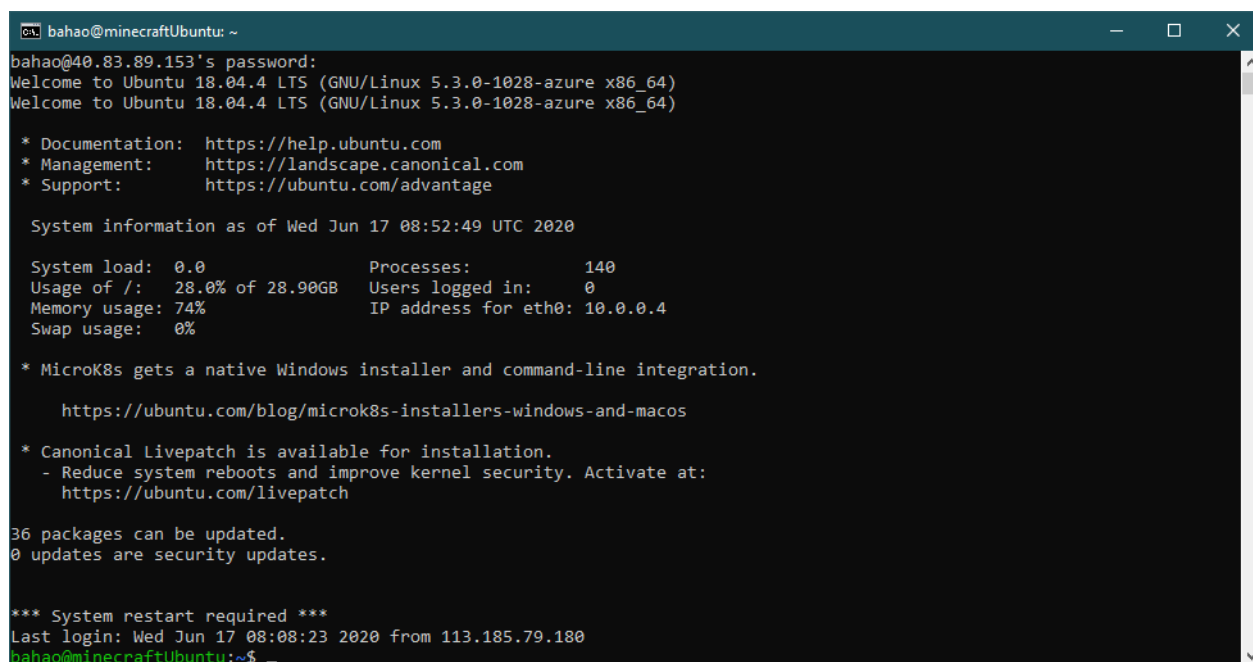
2.3 GIAO THỨC SSH

SSH, hoặc được gọi là Secure Shell, là một giao thức điều khiển từ xa cho phép người dùng kiểm soát và chỉnh sửa server từ xa qua Internet. Dịch vụ được tạo ra nhằm thay

thể cho trình Telnet vốn không có mã hóa và sử dụng kỹ thuật cryptographic để đảm bảo tất cả giao tiếp gửi tới và gửi từ server từ xa diễn ra trong tình trạng mã hóa.

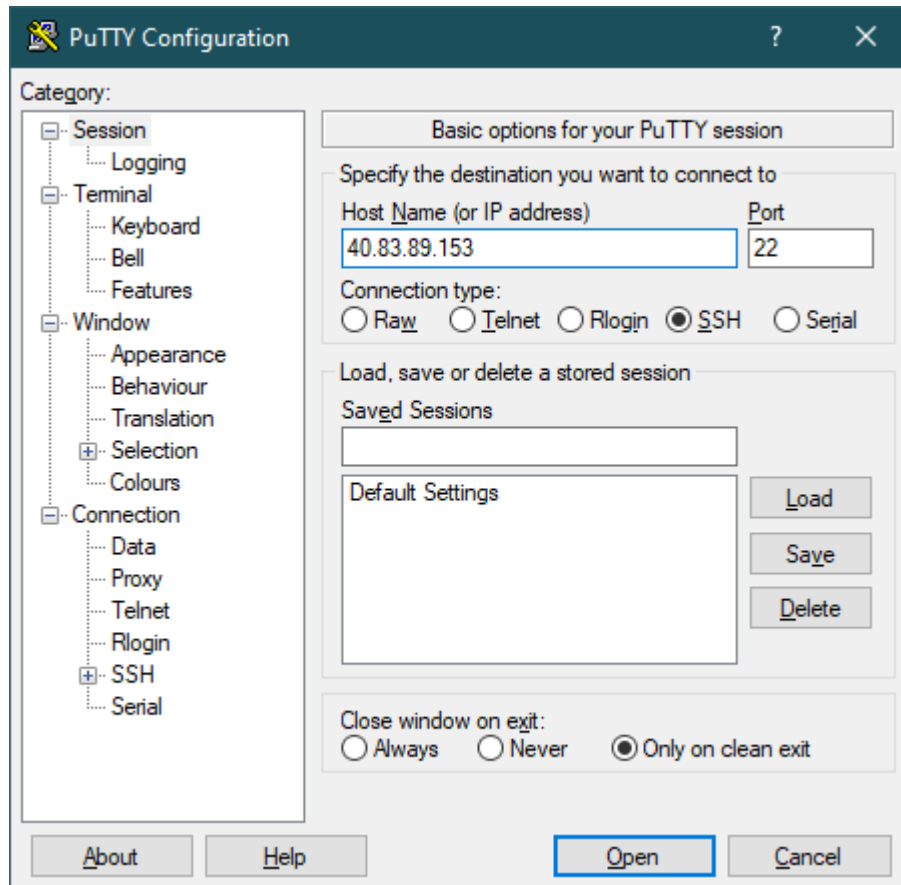
Giao thức SSH là cách thức giao tiếp phổ biến của các hệ thống không dây với nhau. Tron phạm vi đồ án này, SSH được sử dụng để làm cầu nối giao tiếp từ máy tính đến Raspberry Pi.

Có nhiều cách để sử dụng SSH là dùng Commandline hoặc phần mềm bên thứ 3.



```
bahao@minecraftUbuntu: ~  
bahao@40.83.89.153's password:  
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1028-azure x86_64)  
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1028-azure x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Wed Jun 17 08:52:49 UTC 2020  
  
System load:  0.0      Processes:      140  
Usage of /:   28.0% of 28.90GB   Users logged in:  0  
Memory usage: 74%      IP address for eth0: 10.0.0.4  
Swap usage:   0%  
  
* MicroK8s gets a native Windows installer and command-line integration.  
  https://ubuntu.com/blog/microk8s-installers-windows-and-macos  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
36 packages can be updated.  
0 updates are security updates.  
  
*** System restart required ***  
Last login: Wed Jun 17 08:08:23 2020 from 113.185.79.180  
bahao@minecraftUbuntu:~$
```

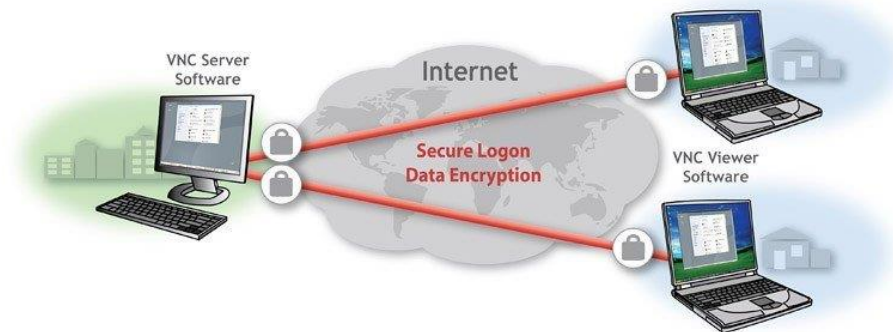
Hình 2.4 Giao thử SSH trên Commandline



Hình 2.5 Phần mềm PuTTY

2.4 Giao thức VNC

VNC (Virtual Network Computing) là một công nghệ kỹ thuật dùng để chia sẻ giao diện màn hình từ xa (remote desktop sharing). VNC sẽ giúp người dùng hiển thị được màn hình của máy tính hoặc hệ thống ở xa ngay trên mạng cục bộ của người dùng và có thể điều khiển thao tác qua kết nối mạng.

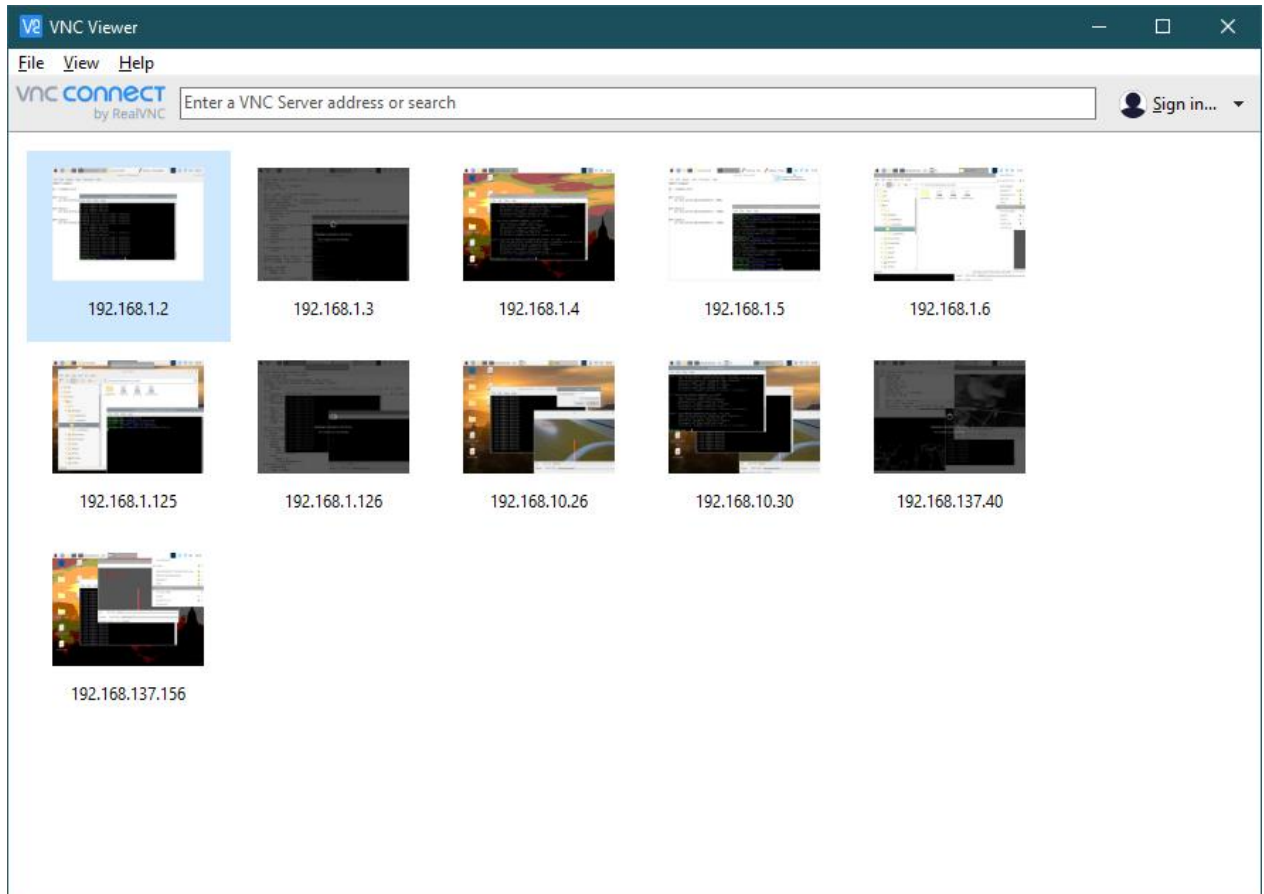


Hình 2.6 Giao thức VNC

Giao thức VNC vận hành dựa trên cơ chế Client – Server và được thiết kế trên ý tưởng của Remote Frame Buffer (RFB). VNC Client (viewer) sẽ chia sẻ các input như (bàn

phím, di chuyển chuột, click chuột,...) với VNC Server. VNC Server sẽ ghi lại các nội dung hiển thị framebuffer và chia sẻ chúng lại cho VNC Client.

Thông thường giao thức VNC sẽ sử dụng TCP và dùng port 5900 trở lên. Giao thức này còn có yêu cầu một số đặc điểm phía người dùng như tỉ lệ màn hình, độ phân giải màn hình, độ rộng màn hình... để đáp ứng được hoạt động của VNC. VNC Viewer là phần mềm được sử dụng rộng rãi hỗ trợ cho giao thức này.



Hình 2.7 VNC Viewer

2.5 Máy tính nhúng Raspberry Pi

2.5.1 Giới thiệu về Raspberry Pi

Sử dụng cấu trúc ARM, Raspberry Pi ra đời lần đầu vào năm 2012. Raspberry Pi hiện nay đã trải qua 4 thế hệ với phiên bản mới nhất là Raspberry Pi 4.

Nhiệm vụ ban đầu của dự án Raspberry Pi là tạo ra máy tính rẻ tiền có khả năng lập trình cho những sinh viên, nhưng Pi đã được sự quan tâm từ nhiều đối tượng khác nhau. Đặc tính của Raspberry Pi xây dựng xoay quanh bộ xử lý SoC Broadcom BCM2835 (là chip xử lý mobile mạnh mẽ có kích thước nhỏ hay được dùng trong điện thoại di động) bao gồm CPU, GPU, bộ xử lý âm thanh/hình ảnh và các tính năng khác...tất cả được tích hợp bên trong chip siêu nhỏ này.

Raspberry Pi không thay thế hoàn toàn hệ thống để bàn hoặc máy xách tay . Bạn không thể chạy Windows trên đó vì BCM2835 dựa trên cấu trúc ARM nên không hỗ trợ mã x86/x64 , nhưng vẫn có thể chạy bằng Linux với các tiện ích như lướt web , môi trường Desktop và các nhiệm vụ khác. Được trang bị nhiều cổng giao tiếp thông dụng như RJ45, USB, HDMI, các chân GPIO cùng hệ điều hành được tùy biến tiện lợi với các dịch vụ SSH, VNC tiện dụng cho việc kết nối không cần màn hình. Vi xử lý không quá mạnh nhưng vừa đủ với các ứng dụng IoT, xử lý ảnh.

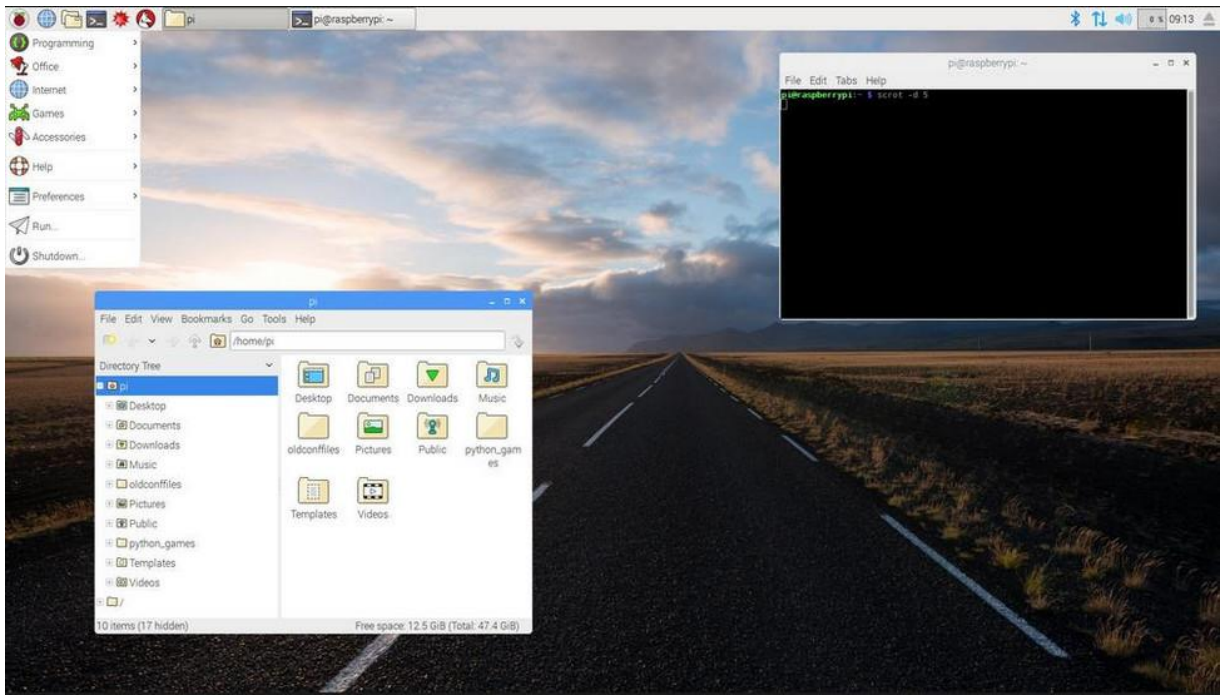


Hình 2.8 Mạch Raspberry Pi

Đồ án này sử dụng Raspberry Pi 4 (trang bị chip ARM Cortex-A72 64bit xung nhịp 1.5 GHZ và 1GB RAM) để làm bộ xử lý chính cho hệ thống xử lý ảnh và hệ thống lái xe.

2.5.2 Hệ điều hành Raspian

Raspian là một trong những hệ điều hành linux có thể sử dụng trên Raspberry Pi và hỗ trợ rất tốt cho Raspberry Pi. Dựa trên Debian và cài sẵn nhiều phần mềm tiện dụng biến Raspberry Pi thành một chiếc máy tính mini đúng nghĩa như nghe nhạc, lướt web, soạn văn bản, gửi email. Về mặt IoT, Raspian hỗ trợ khả năng tùy biến rất lớn cho các chân tín hiệu trên Raspberry Pi. Khả năng chạy các dịch vụ từ năm này sang năm khác rất tốt.



Hình 2.9 Giao diện Raspbian

2.6 Thuật toán PID

Sau khi giải quyết xong bài toán xử lý ảnh và có thông số đầu ra là góc độ để điều hướng cho xe thì việc nan giải tiếp theo là lựa chọn một thuật toán điều khiển thật chuẩn xác. Thuật toán đơn giản nhất thường dùng là thuật toán Bang Bang. Đây

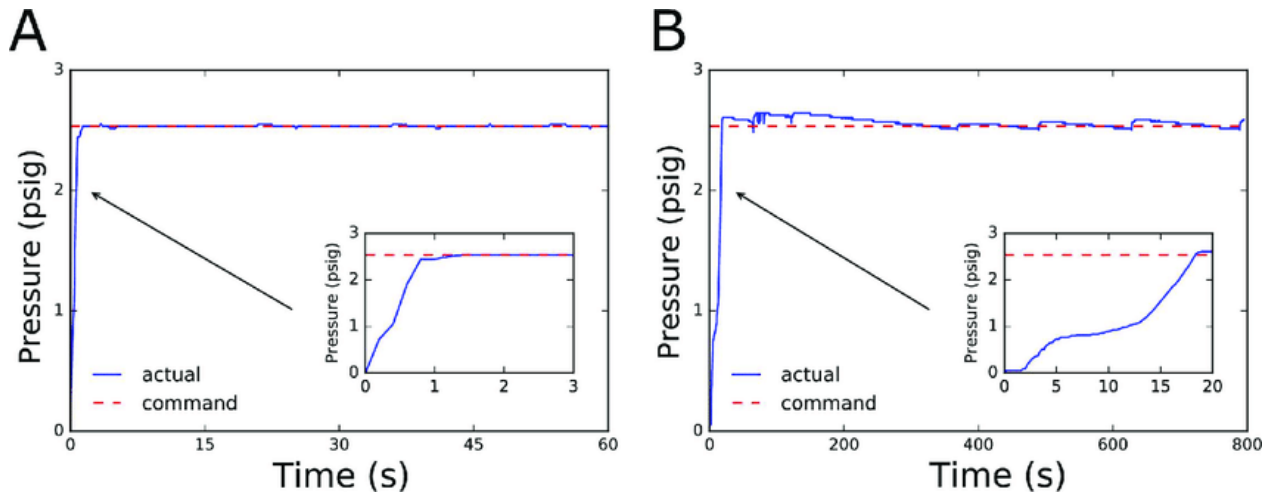
Các hệ thống đơn giản thường áp dụng kiểu xử lý Bang Bang:

```

1. def BangBang(angle):
2.     deviation = 90 - angle
3.     if(deviation > 0):
4.         turn_right()
5.     elif(deviation < 0):
6.         turn_left()
7.     else:
8.         go_on()
9.     forward(100)

```

Đây là một thuật toán đơn giản, thiếu chính xác và đặc biệt tốn tài nguyên vì thế không thể áp dụng trong công nghiệp hay các hệ thống phức tạp. PID viết tắt cho Proportional, Integral và Derivative tức là Tỷ lệ - Tích phân – Đạo hàm. PID là một thuật toán điều khiển được áp dụng rộng rãi trong lĩnh vực tự động hóa và robotic do tính chính xác cao và tốn ít tài nguyên hơn. Thuật toán PID có thể được tìm thấy trong tủ lạnh, máy nước nóng, máy giặt, máy lạnh và rất nhiều máy móc khác. Giống như ngôn ngữ bậc thấp và bậc cao, thuật toán PID đã được đơn giản hóa hơn rất nhiều.



Hình 2.10 Thuật toán PID và thuật toán Bang Bang

2.7 THƯ VIỆN XỬ LÝ ẢNH OPENCV

2.7.1 Giới thiệu

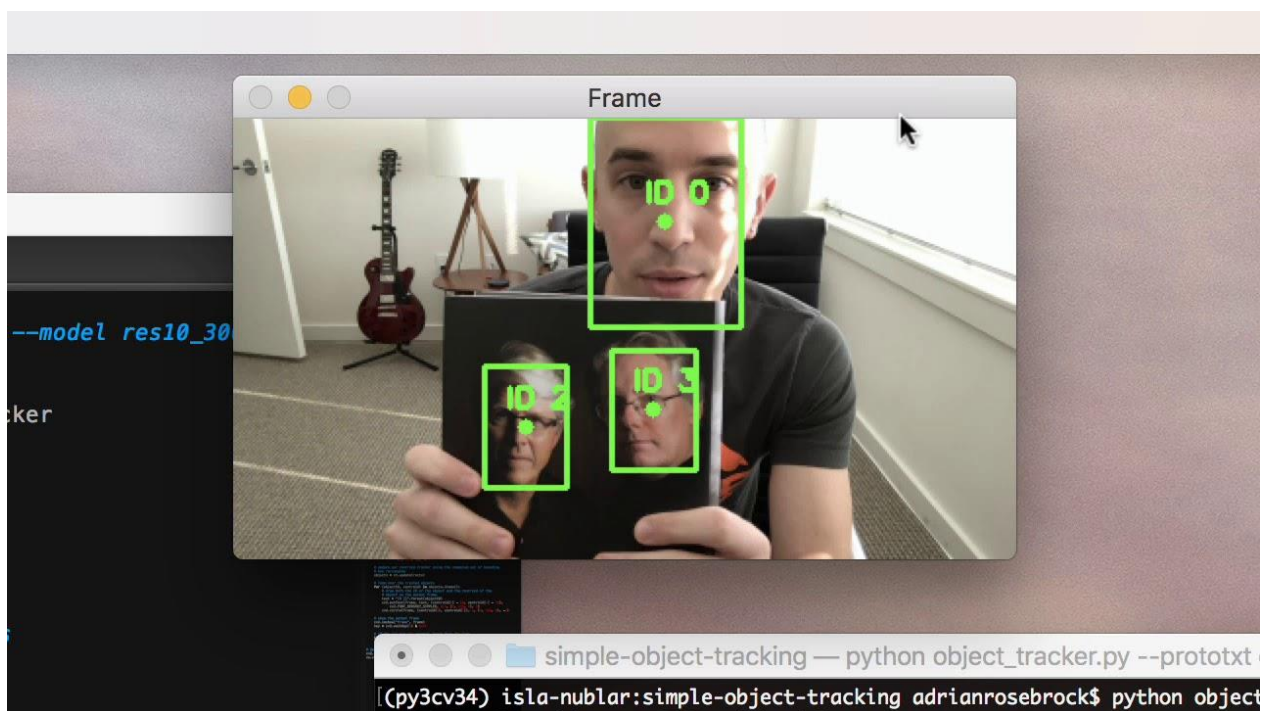
OpenCV là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực.

OpenCV được phát hành theo giấy phép BSD, do đó nó hoàn toàn miễn phí cho cả học thuật và thương mại. Nó có các interface C++, C, Python, Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để tính toán hiệu quả và với sự tập trung nhiều vào các ứng dụng thời gian thực. Được viết bằng tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế của xử lý đa lõi. Được sử dụng trên khắp thế giới, OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần. Phạm vi sử dụng từ nghệ thuật tương tác, cho đến lĩnh vực khai thác mỏ, bản đồ trên web hoặc công nghệ robot.

2.7.2 Ứng dụng

OpenCV đang được sử dụng rộng rãi trong các ứng dụng bao gồm:

- Hình ảnh street view
- Kiểm tra và giám sát tự động
- Robot và xe hơi tự lái
- Phân tích hình ảnh y tế
- Tìm kiếm và phục hồi hình ảnh/video
- Phim - cấu trúc 3D từ chuyển động
- Nghệ thuật sắp đặt tương tác
- Phát hiện vật thể



Hình 2.11 Ứng dụng phát hiện vật thể

2.7.3 Sử dụng một số hàm cơ bản với OpenCV

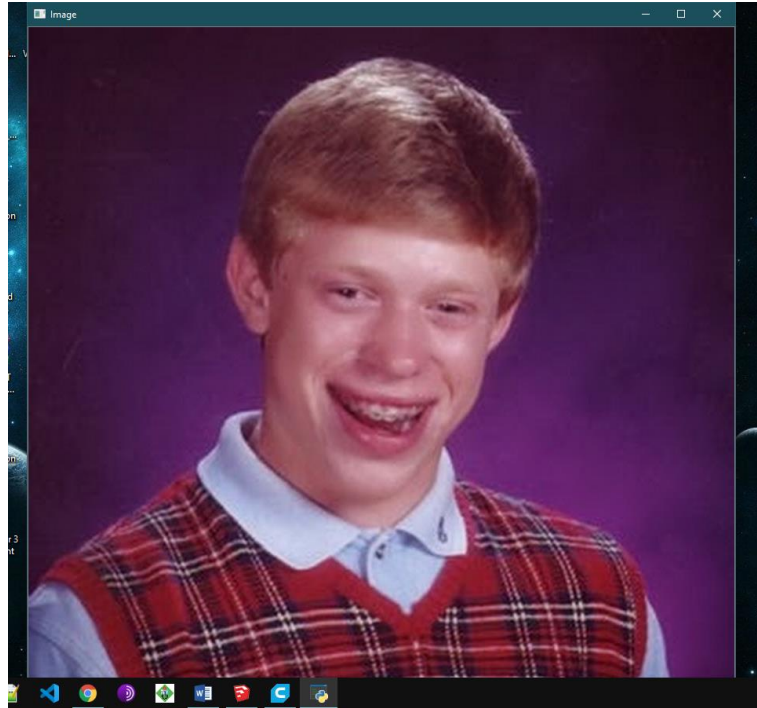
a) Mở ảnh và hiển thị hình ảnh

Khai báo thư viện OpenCV bằng lệnh `import cv2` và đưa địa chỉ của ảnh nằm trong biến `location` cho hàm `cv.imread()` để lưu dữ liệu của ảnh vào hàm `image`. Cuối cùng sử dụng hàm `cv.imshow()` để hiển thị ảnh lên màn hình.

```

1. #Khai báo thư viện openCV
2. import cv2 as cv
3. #Khai báo vị trí ảnh
4. location = r'C:\Users\Admin\Desktop\xe tu hanh\demo opencv\brian.jpg'
5. #Load ảnh
6. image = cv.imread(location)
7. #Show ảnh lên màn hình
8. cv.imshow("Image", image)
9. cv.waitKey(0)

```



Hình 2.12 Hiển thị hình ảnh lên cửa sổ với OpenCV

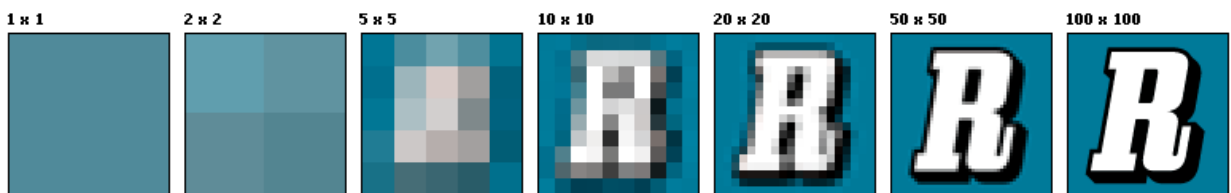
2.8 LÝ THUYẾT XỬ LÝ ẢNH

2.8.1 Ảnh kỹ thuật số

Ảnh kỹ thuật số là một dạng biểu diễn của ảnh ở dạng ma trận số 2 chiều. Tùy vào độ phân giải của ảnh có cố định hay không, ảnh kỹ thuật số được chia ra làm 2 loại là ảnh vector (độ phân giải không cố định) và ảnh raster (có độ phân giải cố định).

2.8.2 Độ phân giải ảnh

Độ phân giải ảnh là mức độ chi tiết mà ảnh có thể thể hiện. Thuật ngữ này được dùng cho ảnh raster hay ảnh có độ phân giải cố định.



Hình 2.13 Sự khác biệt giữa các độ phân giải

Hình 2.6.3 cho ta thấy sự khác biệt giữa các độ phân giải khác nhau, theo đó độ phân giải càng cao thì ảnh càng nét hơn.

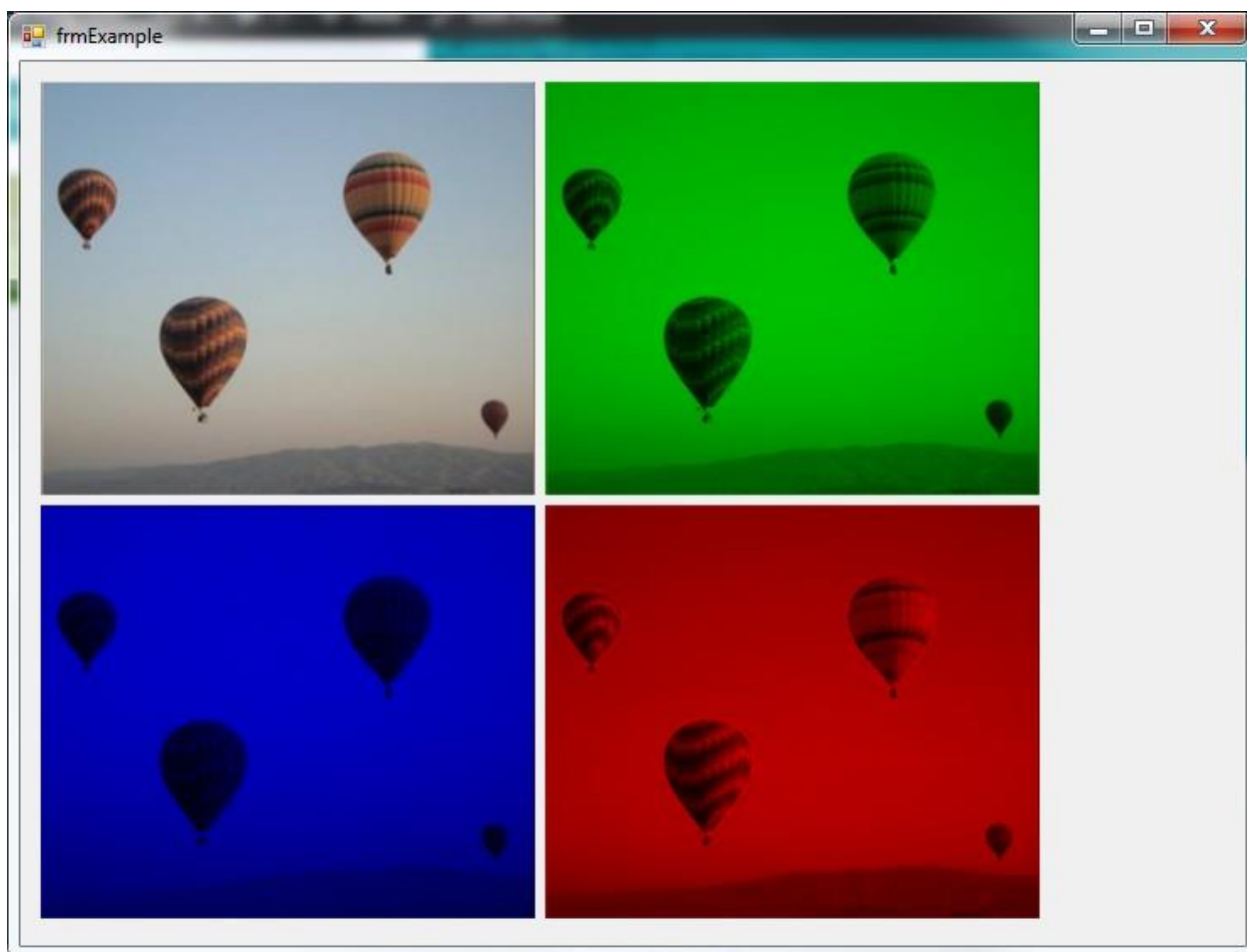
2.8.3 Điểm ảnh

Trong ảnh kỹ thuật số, một điểm ảnh (pixel) là phần tử nhỏ nhất của ảnh raster (raster image). Mỗi một điểm ảnh là một mẫu (sample) của ảnh. Càng nhiều điểm ảnh, ảnh kỹ

thuật số càng biểu diễn chính xác hơn về nội dung của ảnh gốc. Đặc trưng của một điểm ảnh gồm 2 thành phần: tọa độ (x,y) và cường độ sáng (intensity).

2.8.4 Ảnh màu

Để biểu diễn ảnh màu, theo cách biểu diễn RGB, ba ma trận mức xám 256, ứng với ba màu đỏ (R), lục (G), lam(B) được sử dụng. Màu sắc của một điểm ảnh được quyết định bởi giá trị cường độ (intensity) tại ba ma trận màu cùng tọa độ.



Hình 2.14 3 kênh màu

2.8.5 Kênh màu HSV

Không gian màu HSV, còn gọi là kênh màu HSV. Khác với RGB dựa vào 3 màu chính là đỏ (Red), xanh lá(Green), xanh lam(Blue). HSV là một không gian màu dựa trên ba số liệu:

- H: (Hue) Vùng màu
- S: (Saturation) Độ bão hòa màu
- B (hay V): (Bright hay Value) Độ sáng

Kênh màu HSV thường được sử dụng trong các tác vụ theo dõi vật thể (object tracking) dựa vào màu sắc.

Code chuyển từ kênh màu RGB sang HSV trên OpenCV:

```
1. import cv2 as cv
2. location = r'brian.jpg'
3. image = cv.imread(location)
4. image_hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)
5. cv.namedWindow('image', cv.WINDOW_NORMAL)
6. cv.resizeWindow('image', 600, 600)
7. cv.imshow("image", image_hsv)
8. cv.waitKey(0)
```

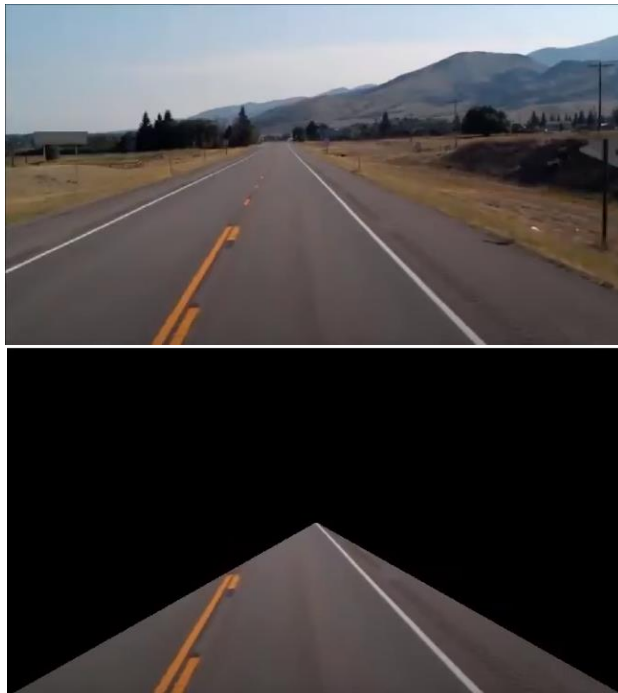


Hình 2.15 RGB sang HSV

2.8.6 Region of Interest

Region of Interest hay còn gọi là vùng quan tâm, viết tắt là ROI là kỹ thuật cắt ảnh để tập trung vào một vùng cần xử lý mà qua đó giảm thiểu thời gian cũng như tài nguyên cho hệ thống. Bên cạnh việc tiết kiệm tài nguyên, ROI còn giúp phần mềm giảm thiểu việc bị nhiễu do các tác nhân ngoại cảnh khác.

Một hình có độ phân giải 1024*1024 có 1,048,576 điểm ảnh phải xử lý. Chúng ta sử dụng kỹ thuật tạo ROI có thể giảm số lượng điểm ảnh cần xử lý xuống còn 30%



Hình 2.16 Hình thường và sau khi áp dụng RoI

2.8.7 Lọc màu ảnh

Kỹ thuật này lấy đầu vào là 2 dãy màu có giá trị thấp nhất và cao nhất của vật thể rồi dùng hàm `inRange()` để phân ngưỡng ảnh tạo thành một ảnh mới (Hình 2.17).

```
1. import cv2
2. import numpy as np
3. img = cv2.imread('golum.jpg')
4. hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
5. lower_blue = np.array([90, 120, 0], dtype="uint8")
6. upper_blue = np.array([150, 255, 255], dtype="uint8")
7. mask = cv2.inRange(hsv, lower_blue, upper_blue)
8. cv2.imshow("mask", mask)
```

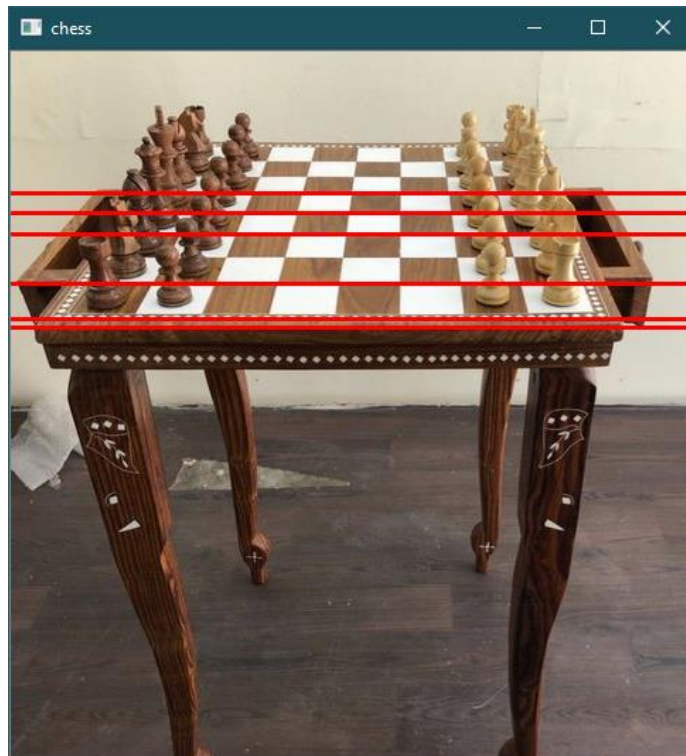


Hình 2.17 Áp dụng phân ngưỡng

2.8.8 Phát hiện đường thẳng bằng giải thuật Hough Transform (Hough Line)

Một đường thẳng có thể được mô tả bằng Rho và Theta. Góc Theta có giá trị từ 0 đến 180 độ, do đó nếu muốn thử các đường thẳng mà mỗi đường thẳng lệch nhau chỉ 1 độ, tức đường thẳng $n+1$, $\text{Theta} = n$ độ thì ta sẽ có 180 đường thẳng có thể thử. Góc Theta đặc tả một đường thẳng là chưa đủ, nên sẽ cần thêm khoảng cách Rho để xác định khoảng cách từ đường thẳng đến gốc tọa độ O chiếu vuông góc với đường thẳng. Giá trị tối đa của Rho bằng đường chéo của ảnh. Từ đó ta suy ra được ma trận thống kê với khởi tạo bằng 0 và kích thước bằng $\text{Rho} * \text{Theta}$.

```
1. import cv2 as cv
2. import numpy as np
3. img = cv.imread(cv.samples.findFile('chess.jpg'))
4. gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
5. edges = cv.Canny(gray, 50, 150, apertureSize = 3)
6. lines = cv.HoughLines(edges, 1, np.pi/180, 200)
7. for line in lines:
8.     rho, theta = line[0]
9.     a = np.cos(theta)
10.    b = np.sin(theta)
11.    x0 = a*rho
12.    y0 = b*rho
13.    x1 = int(x0 + 1000*(-b))
14.    y1 = int(y0 + 1000*(a))
15.    x2 = int(x0 - 1000*(-b))
16.    y2 = int(y0 - 1000*(a))
17.    cv.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
18. cv.imshow("chess", img)
19. cv.waitKey(0)
```



Hình 2.18 Tìm đường thẳng

CHƯƠNG 3: XÂY DỰNG THUẬT TOÁN VÀ MÔ HÌNH

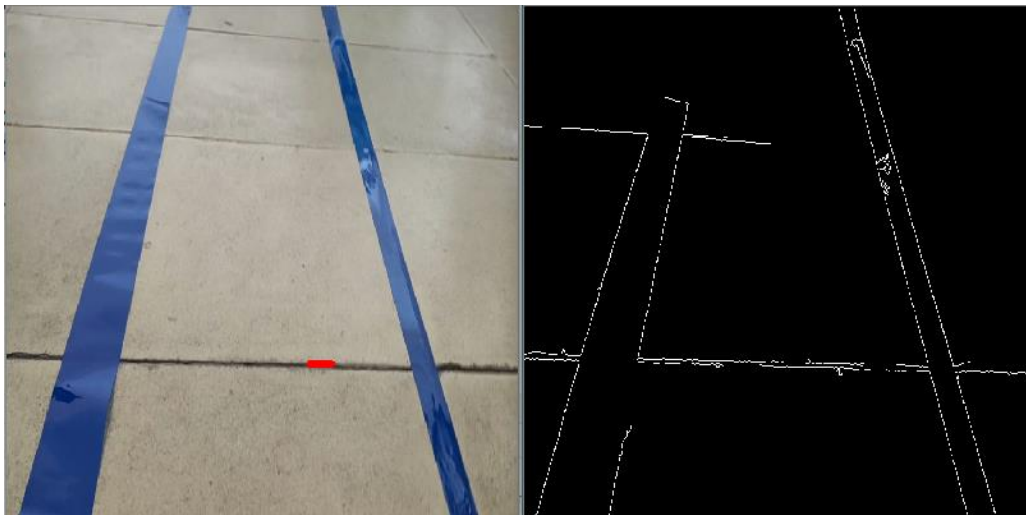
3.1 Xử lý ảnh

3.1.1 Lấy mẫu thực nghiệm và các thông số đầu vào

Khi thực hiện một bài toán xử lý ảnh theo dõi vật thể trong thực tế, các màu sắc ít bị trùng lặp với môi trường như xanh dương và xanh lá sẽ được sử dụng. Trong đồ án này băng keo xanh dương được sử dụng do đặc tính màu sắc không bị trùng lặp, ít bị nhiễu, khả năng phản xạ lại môi trường thấp và sự nổi bật khi chuyển sang kênh màu HSV.

Thực hiện lấy mẫu thử nghiệm bằng cách chụp hình 2 đường thẳng băng keo xanh bằng camera của Raspberry Pi và thử nghiệm sử dụng kỹ thuật tìm đường thẳng trong hình bằng hàm Canny và HoughLine với đoạn code sau:

```
1. import cv2
2. import numpy as np
3.
4. img = cv2.imread('lane.png')
5. gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6. edges = cv2.Canny(gray, 100, 400)
7. rho = 1
8. theta = np.pi / 180
9. min_threshold = 10
10. lines = cv2.HoughLinesP(edges, rho, theta, min_threshold,
11.                          np.array([]), minLineLength=20, maxLineGap=0)
12. for x1,y1,x2,y2 in lines[0]:
13.     cv2.line(img, (x1,y1), (x2,y2), (0,0,255), 5)
14. print(lines)
15. cv2.namedWindow('ed', cv2.WINDOW_NORMAL)
16. cv2.namedWindow('hough', cv2.WINDOW_NORMAL)
17. cv2.resizeWindow('ed', 400, 400)
18. cv2.resizeWindow('hough', 400, 400)
19. cv2.imshow("ed", edges)
20. cv2.imshow('hough', img)
21. cv2.waitKey(0)
```

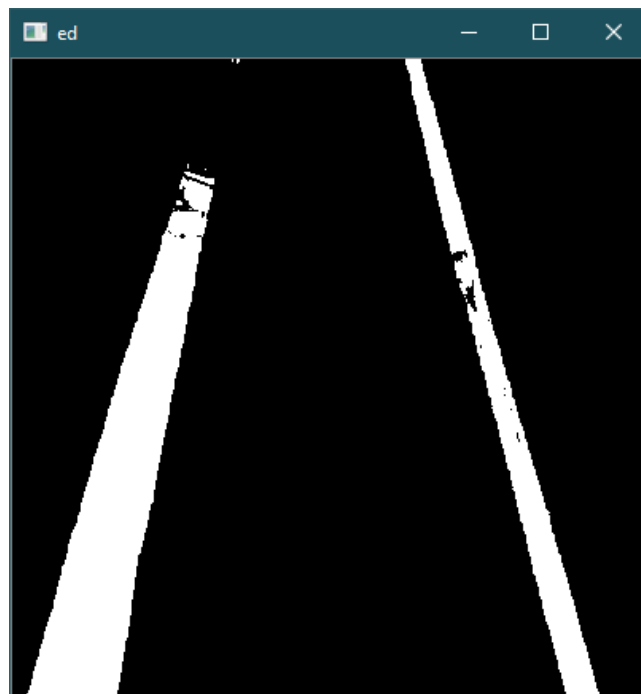


Hình 3.1 Nhiễu xuất hiện

3.1.2 Lọc nhiễu từ môi trường

Do có sự tương đồng trong tính chất nên khi sử dụng HoughLine thì làn gạch cũng vô tình bị nhận nhầm là làn đường. Do đó ta cần sử dụng kỹ thuật lọc đối tượng bằng màu, sử dụng hệ màu HSV và phân ngưỡng ảnh bằng hàm `inRange()`.

```
1. import cv2
2. import numpy as np
3. img = cv2.imread('lane.png')
4. hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
5. lower_blue = np.array([90, 120, 0], dtype="uint8") # Phần giới hạn thấp nhất của d
y màu xanh dương
6. upper_blue = np.array([150, 255, 255], dtype="uint8") # Phần giới hạn cao nhất của
dây màu xanh dương
7. mask = cv2.inRange(hsv, lower_blue, upper_blue) # Lọc ra mọi thứ ngoài dây màu đã đ
ược cho
8. edges = cv2.Canny(mask, 20, 100)
9. min_threshold = 10
10. lines = cv2.HoughLinesP(edges, 1, np.pi / 180, min_threshold,
11.                             np.array([]), minLineLength=50, maxLineGap=10)
12. if lines is not None:
13.     for line in lines:
14.         for x1, y1, x2, y2 in line:
15.             cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), thickness=3)
16.
17. cv2.namedWindow('ed', cv2.WINDOW_NORMAL)
18. cv2.namedWindow('threshold', cv2.WINDOW_NORMAL)
19. cv2.namedWindow('hough', cv2.WINDOW_NORMAL)
20. cv2.resizeWindow('ed', 400, 400)
21. cv2.resizeWindow('hough', 400, 400)
22. cv2.resizeWindow('threshold', 400, 400)
23. cv2.imshow("ed", mask)
24. cv2.imshow('threshold', edges)
25. cv2.imshow('hough', img)
26. cv2.waitKey(0)
```

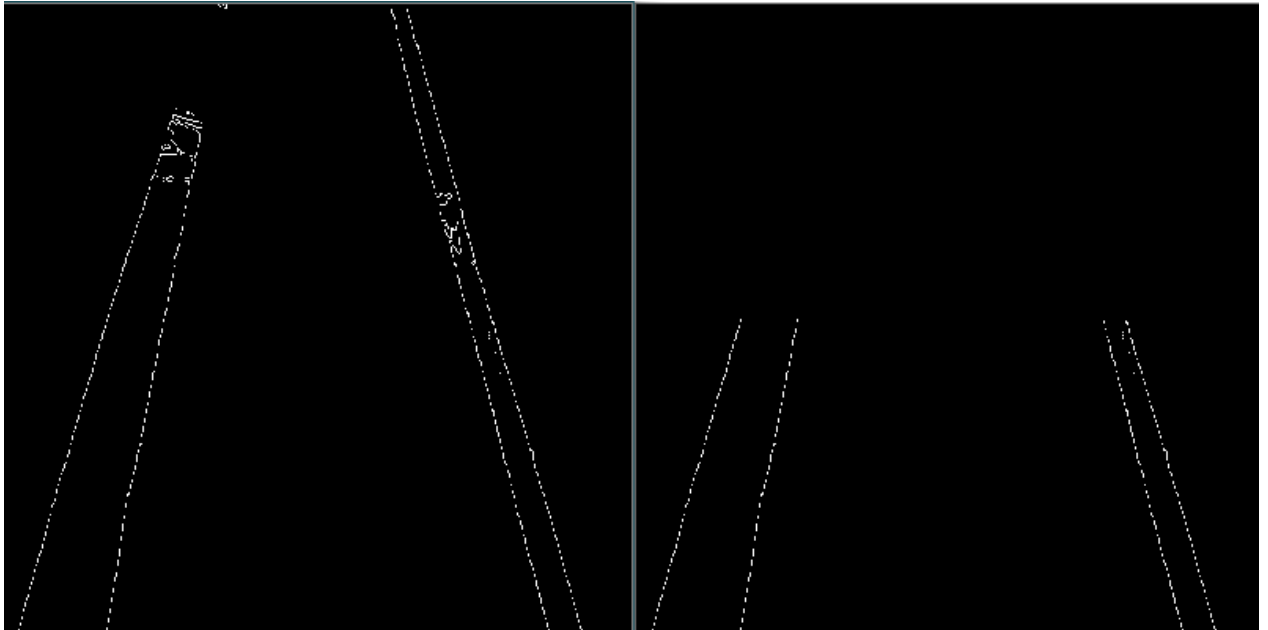


Hình 3.2 Nhiễu không còn nữa

3.1.3 Tạo vùng quan tâm (Region of Interest)

Tiến hành giới hạn việc tập trung vào một vùng duy nhất, tránh các vấn đề nhiễu ngoài môi trường và những làn đường phía xa.

Đầu tiên ta chuyển hình ảnh trên sang hàm phát hiện cạnh Canny để trích xuất 4 cạnh. Hệ tọa độ x,y bắt đầu từ phía trên cùng bên trái của cửa sổ, do đó ta tiến hành trích xuất hai thông tin là chiều cao và chiều dài của frame hình để tiến hành cắt và tạo khung cho frame hình. Sau đó tạo một hình chữ nhật có tọa độ nằm ở nửa dưới frame hình. Tiến hành tô toàn bộ hình chữ nhật là màu xanh với phương thức fillPoly của lớp cv2. Cuối cùng sử dụng toán tử bitwise để cắt nửa dưới của hình.



Hình 3.3 Áp dụng Roi

3.1.4 Chuyển ảnh sang tham số đường thẳng với giải thuật Hough Transform

Hàm HoughLineP trả về các mảng với mỗi mảng có 4 giá trị là x1, y1, x2, y2 tương ứng với đường thẳng mà nó phát hiện được.

```
1. def get_lines(frame):  
2.     rho = 1  
3.     theta = np.pi / 180  
4.     min_threshold = 10  
5.     lines = cv2.HoughLinesP(frame, rho, theta, min_threshold,  
6.                             np.array([]), minLineLength=5, maxLineGap=0)  
7.     return lines
```

Tiến hành in ra các thông số ta được như sau:

```
1. [[ 37 417 40 410]]  
2. [[568 259 571 266]]  
3. [[119 242 124 231]]  
4. [[613 292 616 299]]...
```

Áp dụng công thức khá quen thuộc $y = mx + b$. Trong đó m là độ dốc (slope) của đường thẳng và b là điểm chặn của trục y . Từ đó chúng ta tiến hành tạo một hàm lấy các tham số trả về của giải thuật HoughLine để tính ra độ dốc của đường thẳng. Dựa theo trục tọa độ chúng ta có thể thấy đường thẳng bên trái sẽ có độ dốc âm vì $x_1 < x_2$ và $y_2 < y_1$ và $\text{slope} = (y_2 - y_1) / (x_2 - x_1)$ còn đường thẳng bên phải sẽ ngược lại hoàn toàn tức là độ dốc dương $x_2 > x_1, y_2 > y_1$. Trường hợp $x_2 = x_1$ độ dốc sẽ là vô cùng và chắc chắn phải bị bỏ qua để tránh lỗi chia cho không (divine by zero). Để tăng thêm độ chính xác, mỗi khung hình sẽ được chia thành 2 phần trái và phải bởi một đường biên. Số liệu sau xử lý sẽ trả về độ dốc trung bình và điểm chặn của 2 làn trái – phải.

```

1. def average_slope_intercept(frame, line_segments):
2.     lane_lines = []
3.     if line_segments is None:
4.         print("no line segment detected")
5.         return lane_lines
6.
7.     height = frame.shape[0]
8.     width = frame.shape[1]
9.     left_fit = []
10.    right_fit = []
11.    boundary = 1 / 3
12.
13.    left_region_boundary = width * (1 - boundary)
14.    right_region_boundary = width * boundary
15.
16.    for line_segment in line_segments:
17.        for x1, y1, x2, y2 in line_segment:
18.            if x1 == x2:
19.                print("skipping vertical lines (slope = infinity)")
20.                continue
21.
22.            fit = np.polyfit((x1, x2), (y1, y2), 1)
23.            slope = (y2 - y1) / (x2 - x1)
24.            intercept = y1 - (slope * x1)
25.
26.            if slope < 0:
27.                if x1 < left_region_boundary and x2 < left_region_boundary:
28.                    left_fit.append((slope, intercept))
29.            else:
30.                if x1 > right_region_boundary and x2 > right_region_boundary:
31.                    right_fit.append((slope, intercept))
32.
33.    left_fit_average = np.average(left_fit, axis=0)
34.    if len(left_fit) > 0:
35.        lane_lines.append(make_points(frame, left_fit_average))
36.
37.    right_fit_average = np.average(right_fit, axis=0)
38.    if len(right_fit) > 0:
39.        lane_lines.append(make_points(frame, right_fit_average))
40.
41.    # lane_lines là một mảng 2 chiều trả về tọa độ trái - phải với mỗi bên 4 điểm là
    x1,y1,x2,y2
42.    # Ví dụ: lane_lines = [[x1,y1,x2,y2],[x1,y1,x2,y2]]
43.    # Các điểm tọa độ là pixel
44.    return lane_lines

```

3.1.5 Tính góc quay xe

Đây là bước xử lý cuối cùng sau khi có được giá trị thô của 2 line trái và phải. Ta tiến hành tính toán và đưa ra lệnh để servo xoay bánh điều hướng và quyết định tốc độ của bánh sau. Các tính toán này dựa theo công thức lượng giác đã cho.

Ở đây chúng ta chia ra làm 3 trường hợp là khi cả 2 làn đường được nhìn thấy, 1 làn đường được nhìn thấy và cả 2 làn đường đều không được nhìn thấy.

- Ở trường hợp thứ nhất: Ta chỉ cần quan tâm đến x2 của làn trái và làn phải rồi lấy trung bình cộng của cả 2 làm điểm queo gọi là x_offset, y_offset thì luôn là chiều cao chia 2.
- Trường hợp thứ hai: Ta lấy x2 trừ cho x1 và đặt kết quả trong biến x_offset, biến y_offset vẫn là chiều cao chia 2.
- Trường hợp cuối cùng: đặt x_offset bằng không và đặt y_offset là chiều cao chia 2.

```
1. def get_steering_angle(frame, lane_lines):
2.     height, width, _ = frame.shape
3.
4.     if len(lane_lines) == 2: #Trường hợp 1
5.         _, _, left_x2, _ = lane_lines[0][0]
6.         _, _, right_x2, _ = lane_lines[1][0]
7.         mid = int(width / 2)
8.         x_offset = (left_x2 + right_x2) / 2 - mid
9.         y_offset = int(height / 2)
10.
11.    elif len(lane_lines) == 1: #Trường hợp 2
12.        x1, _, x2, _ = lane_lines[0][0]
13.        x_offset = x2 - x1
14.        y_offset = int(height / 2)
15.
16.    elif len(lane_lines) == 0: #Trường hợp 3
17.        x_offset = 0
18.        y_offset = int(height / 2)
19.
20.    angle_to_mid_radian = math.atan(x_offset / y_offset)
21.    angle_to_mid_deg = int(angle_to_mid_radian * 180.0 / math.pi)
22.    steering_angle = angle_to_mid_deg + 90
23.
24.    return steering_angle
```

3.1.6 Hiển thị đường thẳng điều hướng

Để tiện cho việc kiểm tra và sửa lỗi, chúng ta hiển thị đường thẳng lên hình.

Tạo một hình mới rộng có kích thước tương đương frame ảnh hiện tại. Tạo các biến x1, y1, x2, y2 để lưu giá trị từ hàm get_steering_angle.

```
1. def display_heading_line(frame, steering_angle, line_color=(0, 0, 255), line_width=5
2. )
3.     heading_image = np.zeros_like(frame)
4.     height, width, _ = frame.shape
5.     steering_angle_radian = steering_angle / 180.0 * math.pi
6.     x1 = int(width / 2)
```

```

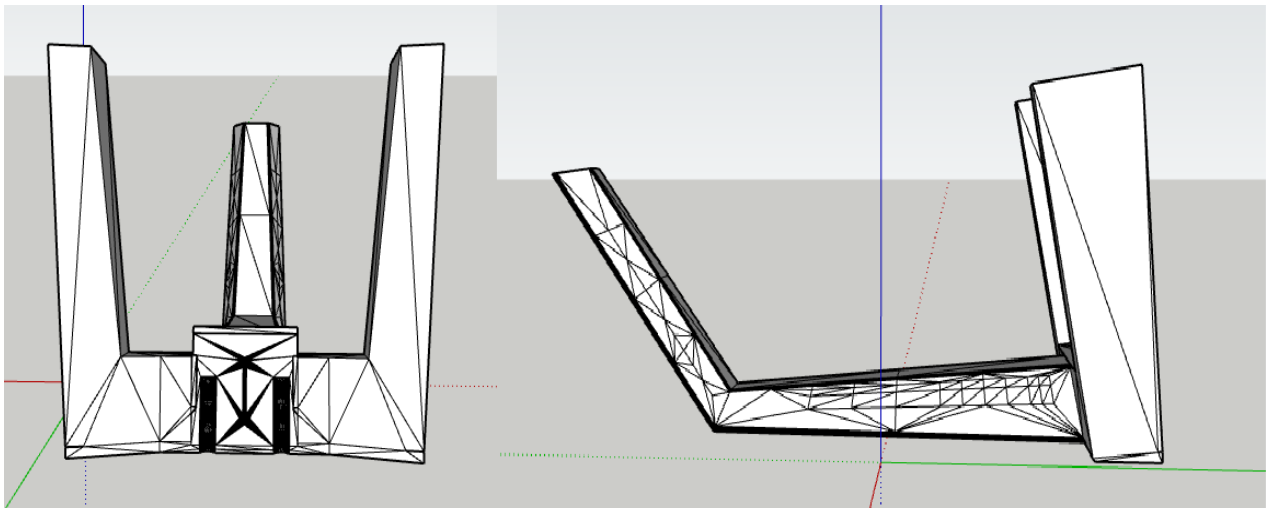
7.     y1 = height
8.     x2 = int(x1 - height / 2 / math.tan(steering_angle_radian))
9.     y2 = int(height / 2)
10.
11.     cv2.line(heading_image, (x1, y1), (x2, y2), line_color, line_width)
12.
13.     heading_image = cv2.addWeighted(frame, 0.8, heading_image, 1, 1)
14.
15.     return heading_image

```

3.2 CHI TIẾT IN 3D

3.2.1 Khung xe Donkey Cage

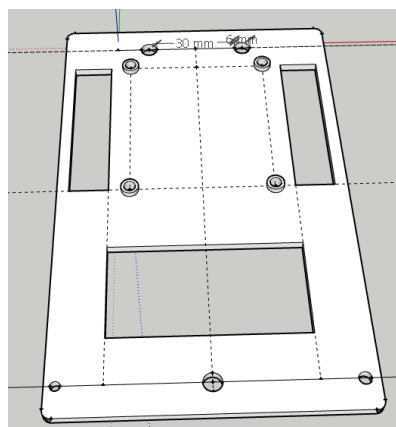
Chúng ta sẽ cần một chiếc khung để đỡ board mạch Raspberry Pi cùng camera và pin được gọi là Donkey Cage giống như chiếc lồng đặt trên lưng con lừa, được vẽ trên Sketchup 2020 và trích xuất ra định dạng in 3D bằng phần mềm Ultimate Cura.



Hình 3.4 Donkey Cage

3.2.2 Đế bắt mạch

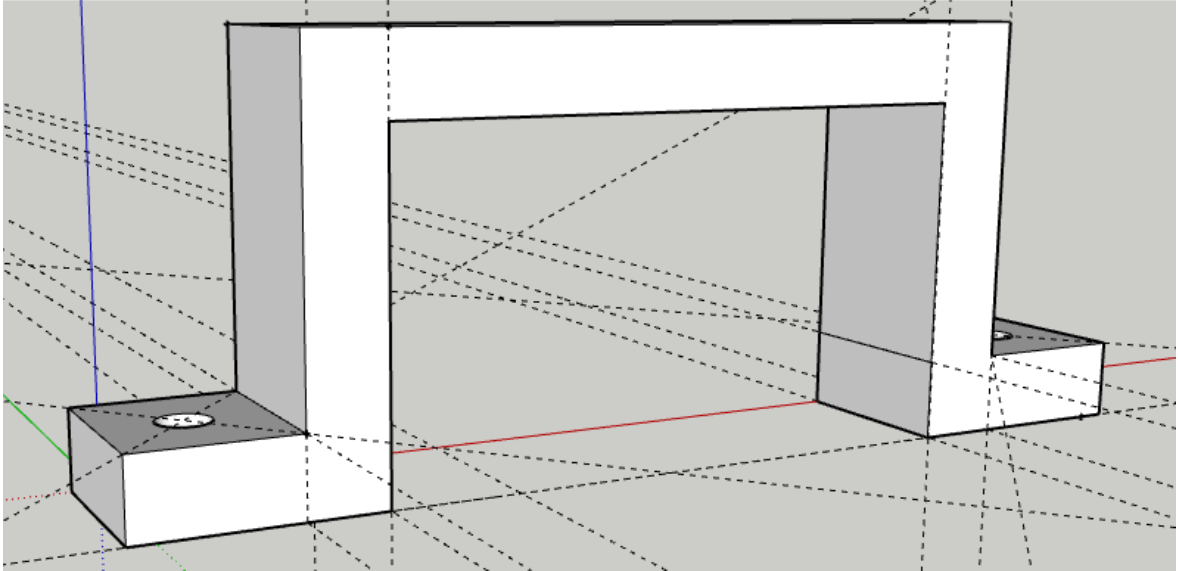
Một đế bắt mạch Raspberry Pi và cố định các chi tiết điện tử rất quan trọng, cần đo đạc kĩ càng khoảng cách để in ra chi tiết chính xác, vì các chi tiết lớn như thế này mất rất nhiều thời gian để in ra hoàn chỉnh.



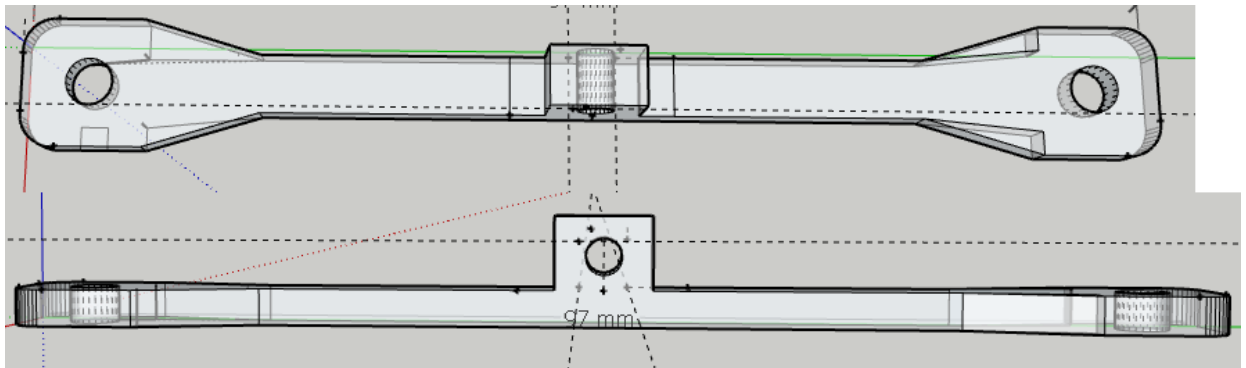
Hình 3.5 Đế bắt linh kiện

3.2.3 Cơ cấu đánh lái

Để bắt servo dùng để cố định servo vào thanh đánh lái gọi là steering bar. Đây là một cơ cấu khá phức tạp và tốn thời gian đo đạc thiết kế. Để bắt servo được cố định bằng 2 ốc có kích cỡ M1.2.



Hình 3.6 Đế bắt servo



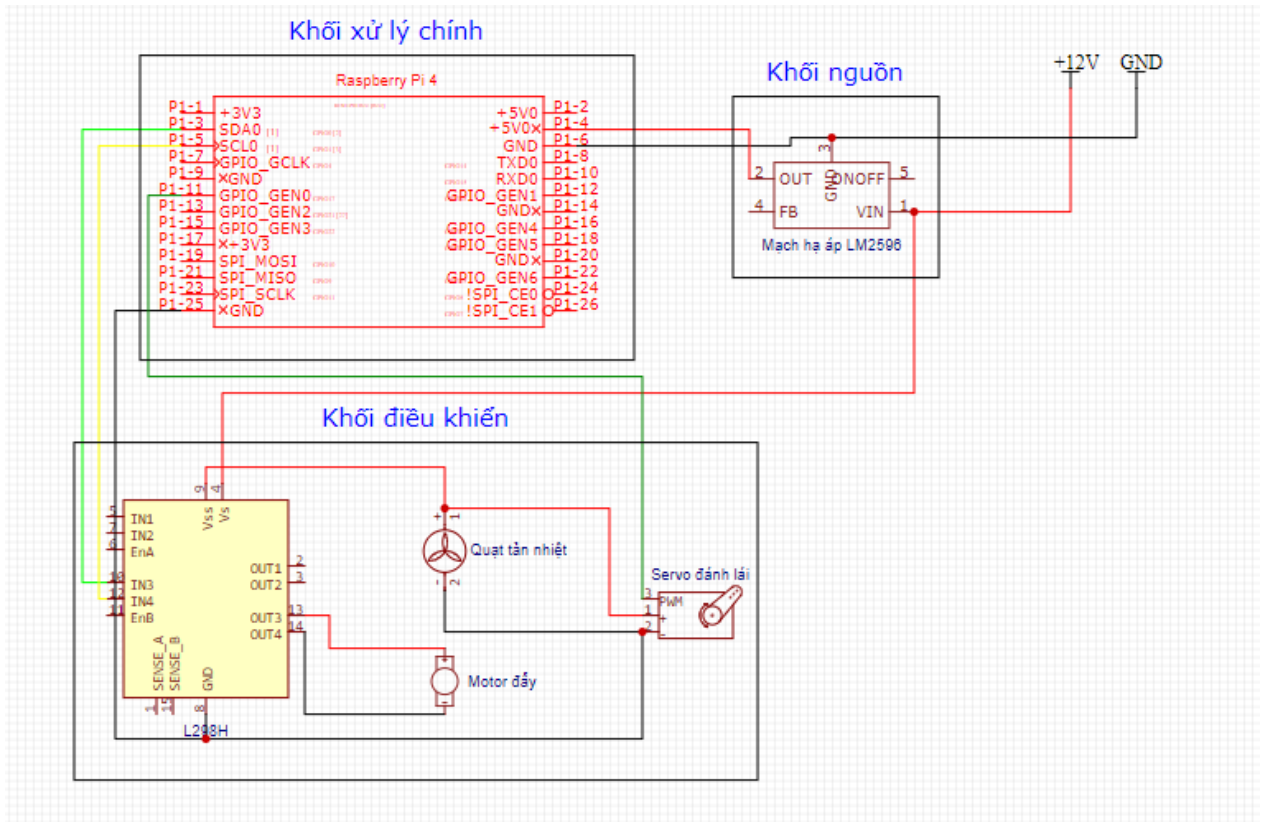
Hình 3.7 Thanh đánh lái

3.3 SƠ ĐỒ MẠCH ĐIỆN

Sơ đồ mạch điện chia làm 3 khối chính: Khối nguồn, khối xử lý, khối điều khiển.

- Khối nguồn lấy nguồn có hiệu điện thế 12V và giảm áp sử dụng mạch LM2596 để giảm hiệu điện thế còn 5V làm nguồn cung cấp chính cho khối xử lý
- Khối xử lý sử dụng mạch Raspberry Pi làm vi xử lý chính, các mã lệnh và hệ điều hành chạy trên khối này, khối xử lý điều khiển trực tiếp servo để tăng độ chính xác và giảm độ trễ. Servo nhận lệnh từ Raspberry Pi dưới dạng xung PWM và nhận nguồn từ khối điều khiển với hiệu điện thế 5V.
- Khối điều khiển với nhân chính là mạch cầu H L298 nhận nguồn với hiệu điện thế 12V và giảm xuống hiệu điện thế 9V để cấp cho động cơ đẩy chính của xe

qua ngõ OUT3, OUT4. Quạt tản nhiệt cho mạch cầu H L298 được cấp 5V qua ngõ VSS.

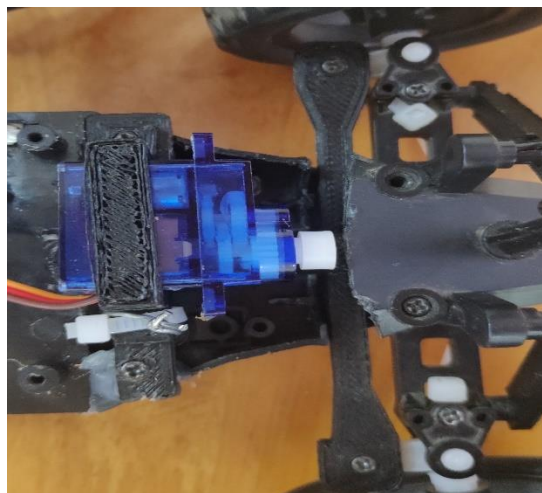


Hình 3.8 Sơ đồ mạch điện

3.4 KẾT QUẢ PHẢN CỨNG

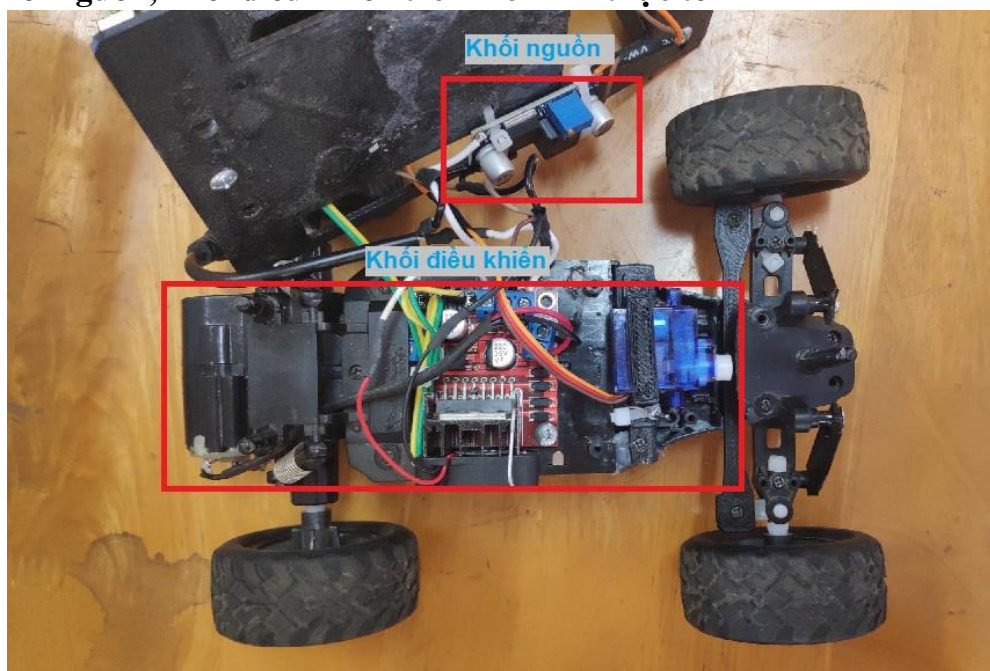
3.4.1 Cơ cấu đánh lái trên mô hình thực tế

Cơ cấu đánh lái gồm servo, đế bắt servo và thanh đánh lái đã được in ra và hoạt động đúng



Hình 3.9 Hệ thống đánh lái

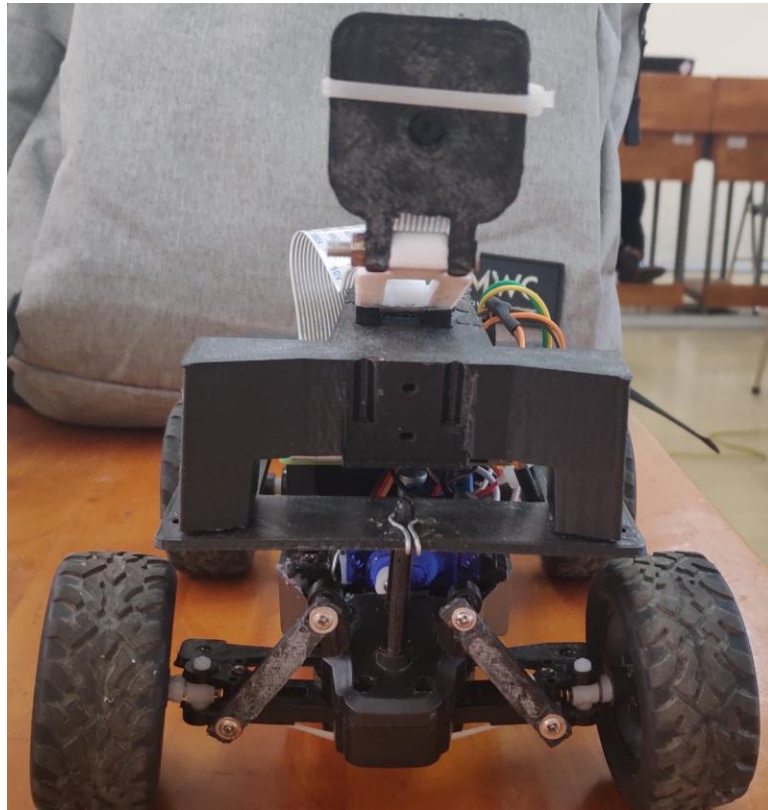
3.4.2 Khối nguồn, khối điều khiển trên mô hình thực tế



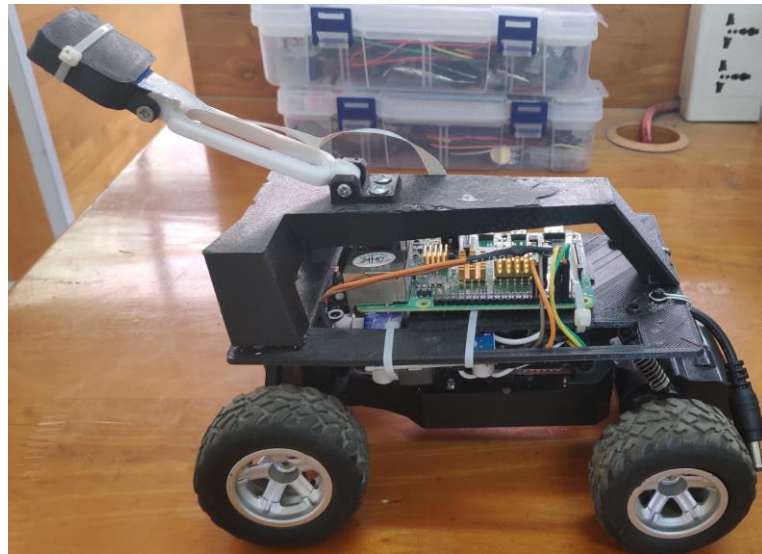
Hình 3.10 Khối nguồn và khối điều khiển

Khối điều khiển được bắt vào thân xe cùng quạt tản nhiệt để giảm nhiệt độ trong quá trình giảm áp. Khối nguồn được gắn bên dưới để gắn Raspberry Pi tận dụng luồng gió từ quạt tản nhiệt để giảm nhiệt độ cho cả 2 khối.

3.4.3 Thân xe



Hình 3.11 Thân trước xe



Hình 3.12 Bên hông cùng khối xử lý

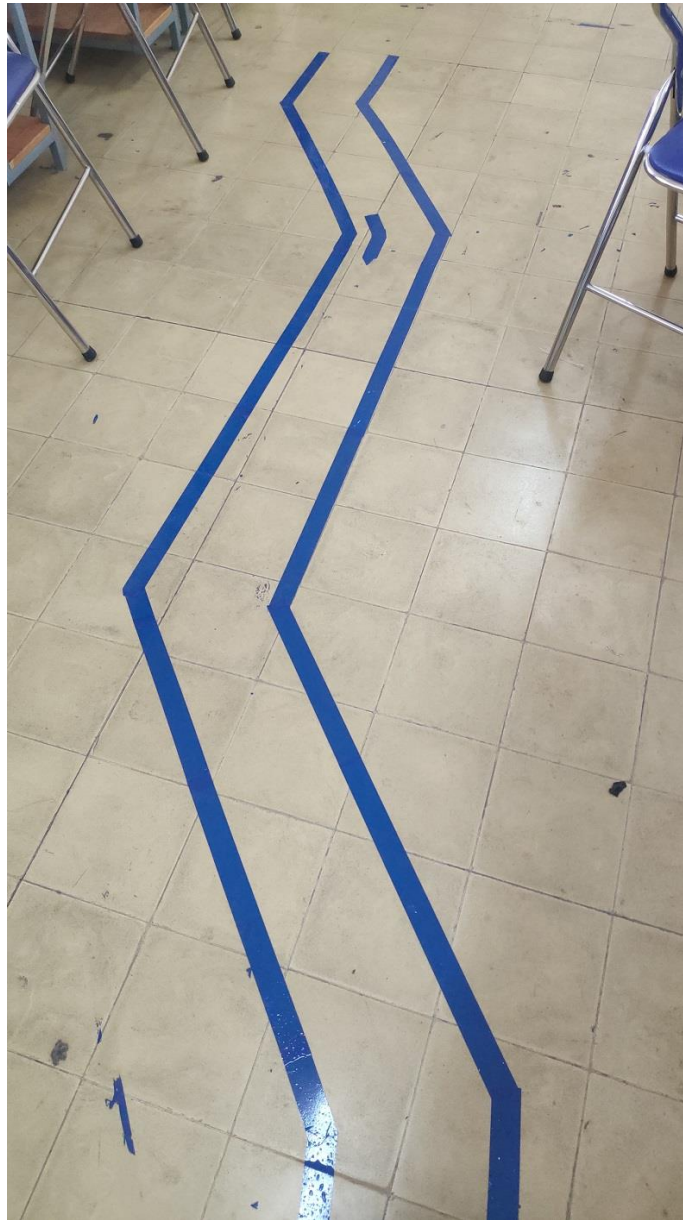
Thân xe có 4 cơ cấu giảm sóc để giảm độ rung cho Camera. Donkey cage sau khi in 3D. Khối xử lý được bắt trên đế bắt mạch và kết nối đến camera cùng khối điều khiển và khối nguồn.

CHƯƠNG 4: THỰC NGHIỆM

4.1 MÔI TRƯỜNG THỰC NGHIỆM

4.1.1 Môi trường vật lý

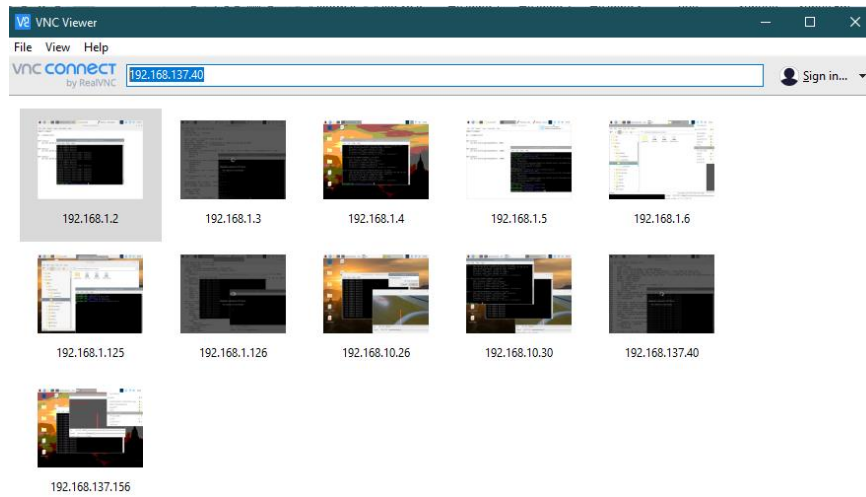
Môi trường thực nghiệm là sàn gạch được dán băng keo xanh làm làn hướng dẫn. Môi trường có ánh sáng đèn bình thường không quá tối và tránh ánh sáng gắt gây ảnh hưởng xấu đến camera giảm chất lượng ảnh dẫn đến sai lệch trong hệ thống xử lý ảnh.



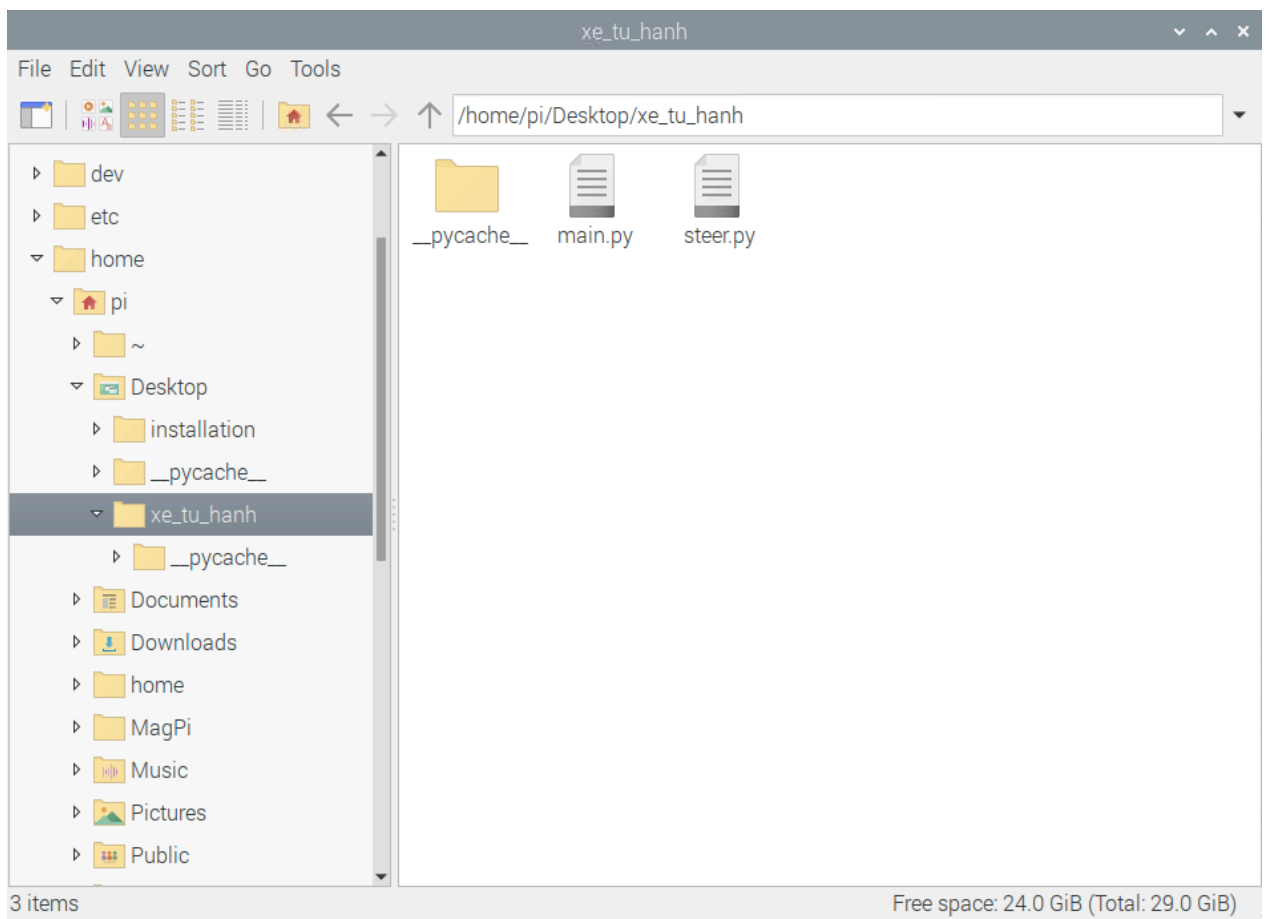
Hình 4.1 Môi trường thử nghiệm

4.1.2 Môi trường phần mềm

Raspberry Pi sử dụng hệ điều hành Raspbian có trình thông dịch Python 3.6 và thư viện chuẩn OpenCV đã được cài đặt sẵn. Sử dụng laptop để phát wifi và cho Raspberry Pi kết nối vào sau đó dùng VNC Viewer để điều khiển Raspberry Pi.



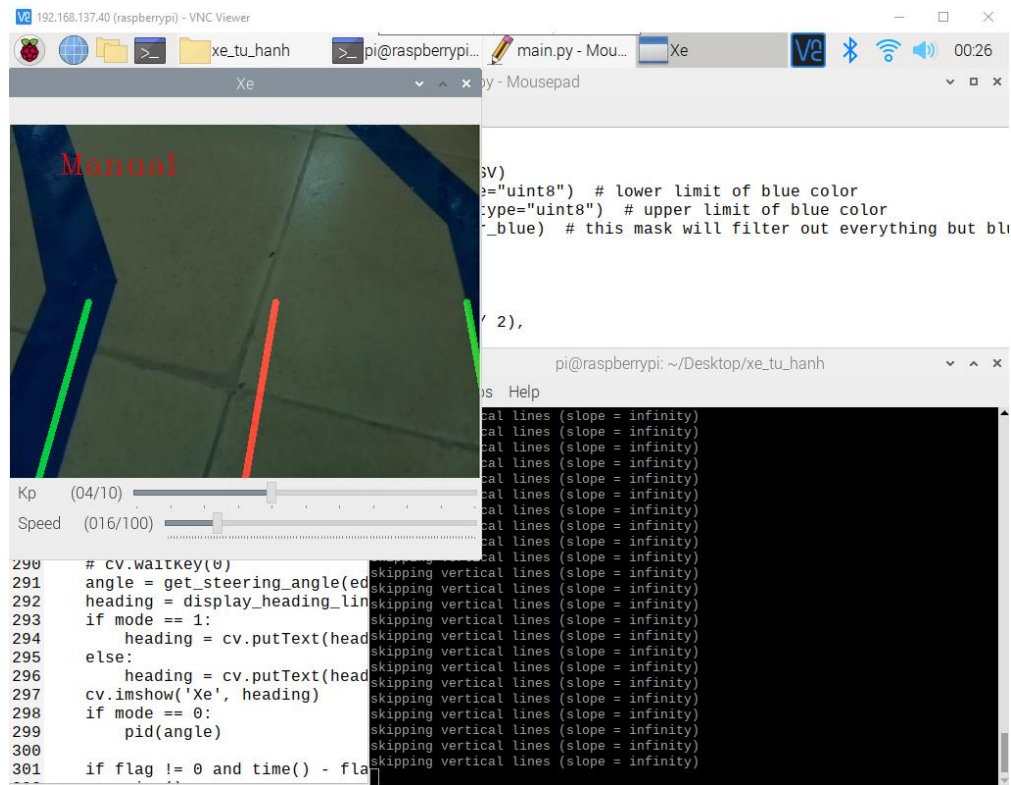
Hình 4.2 Nhập IP của Raspberry Pi



Hình 4.3 Giao diện và môi trường phần mềm điều khiển

4.2 KẾT QUẢ THỰC NGHIỆM

Khi xe bắt đầu chạy, camera xác định làn đường cho ra kết quả điều hướng khá chính xác và không có dấu hiệu nhiễu.



Hình 4.4 Giao diện thực nghiệm

Khi xe tiến hành chạy (Hình 4.5), cơ chế đánh lái hoạt động tốt và ổn định, xe sẽ lạng qua lại để điều chỉnh đường đi sao cho luôn ở giữa 2 làn đường.



Hình 4.5 Xe đang quẹo cua



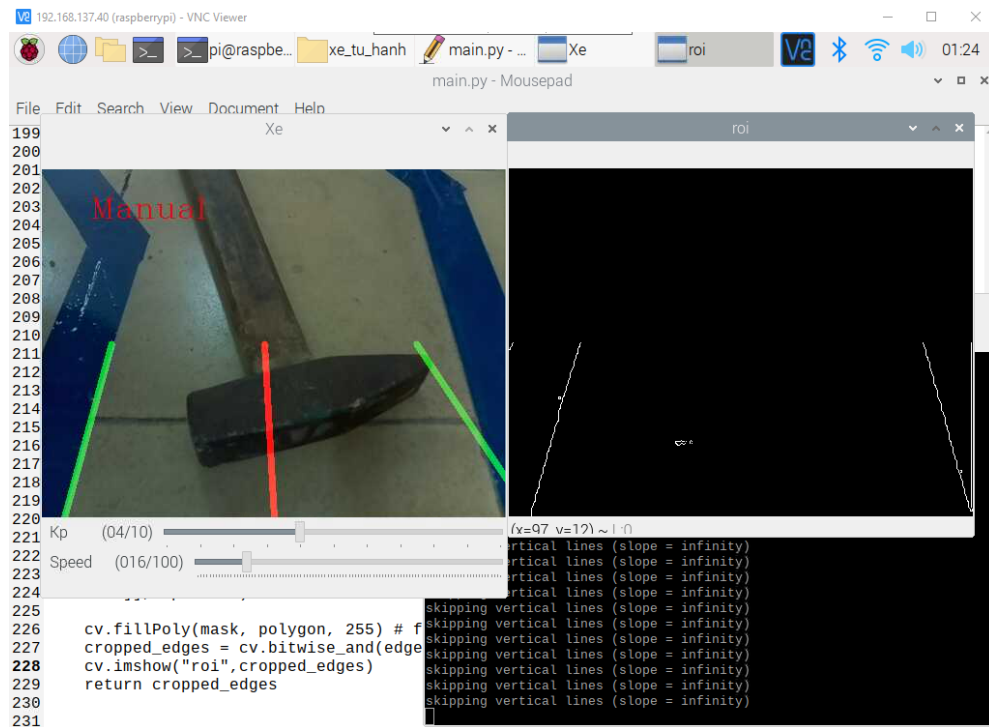
Hình 4.6 Xe trên đường thẳng

Hình 4.6 cho thấy kết quả đi trên đường thẳng khá trơn tru và không bị lệch. Vấn đề xảy ra khi xe vào khúc cua gắt lớn hơn 45 độ thì xe bắt đầu bị lệch ra làn giữa và cố bần vào làn bên phải. Nguyên nhân là do khúc cua quá gắt và hệ thống lái không xử lý được.

Sau khi thử nghiệm trên đường ta tiến thành thử nghiệm khả năng chống nhiễu. Do tính chất của việc cắt ảnh là sử dụng màu sắc nên có thể thấy chiếc búa và bàn tay không thể làm nhiễu được hệ thống xử lý.

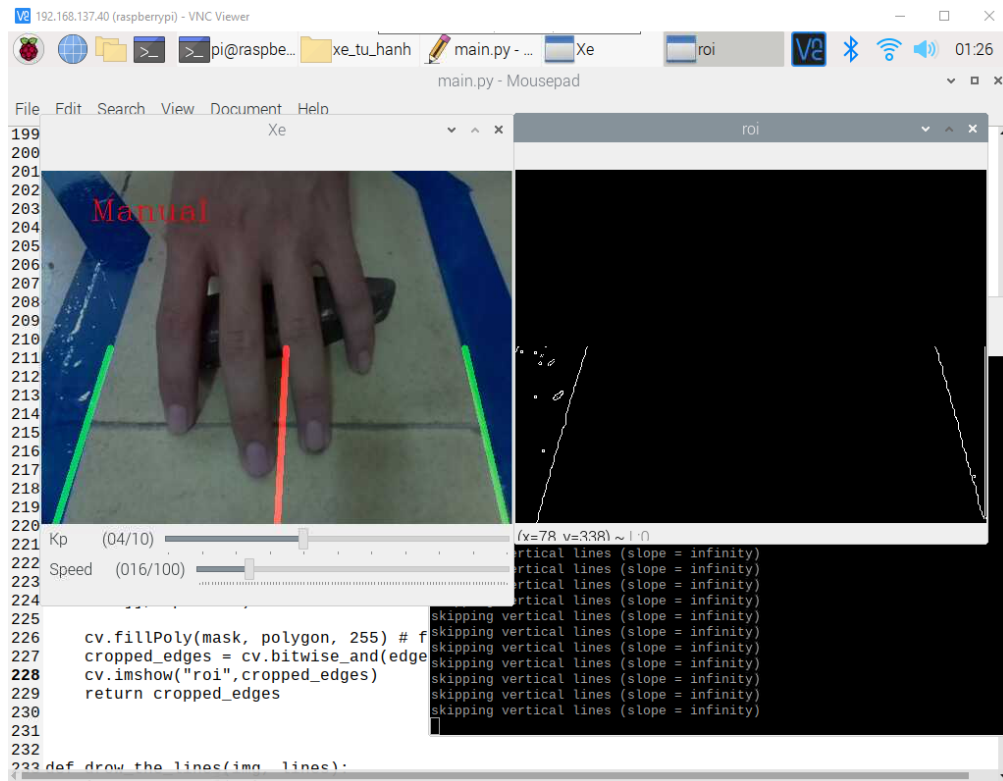
Chương ngại vật, vật gây nhiễu	Khả năng xử lý
Đường thẳng	Tốt
Góc cua dưới 45 độ	Tốt
Góc cua trên 45 độ	Bị trật khỏi làn
Cây búa	Không bị nhiễu loạn
Bàn tay	Không bị nhiễu loạn
Vật có màu xanh tương đương với làn đường	Bị nhiễu loạn

Bảng 4.1 Các khó khăn và khả năng giải quyết



Hình 4.7 Chiếc búa xuất hiện nhưng không bị hiểu nhầm thành đường đi

Trên hình 4.7 là kết quả thử nghiệm việc xử lý hình ảnh làn đường sau giai đoạn RoI.



Hình 4.8 Bàn tay không bị nhận nhầm

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT QUẢ ĐẠT ĐƯỢC, ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA ĐỀ TÀI

5.1.1 Kết quả đạt được

- Hệ thống hoạt động trơn tru ở điều kiện nhiều.
- Hoàn toàn có thể áp dụng trong thực tế với hệ thống khép kín như robot thu hoạch nông nghiệp, robot giao thức ăn trong nhà hàng, hệ thống tự động giữ làn đường.
- Phương pháp xử lý hình ảnh mang tính ứng dụng cao
- Giúp người nghiên cứu đề tài tiếp cận được với nhiều công nghệ tiên phong hiện nay như xử lý ảnh, lập trình nhúng, in 3D và ngôn ngữ lập trình Python.

5.1.2 Ưu điểm

- Hệ thống có thể nhận dạng làn đường và đưa ra chỉ dẫn cho xe với độ chính xác cao.
- Mô hình có khả năng mở rộng tính năng trong tương lai.
- Mang tính ứng dụng cao

5.1.3 Nhược điểm

- Khả năng chính xác còn hạn chế và có khả năng bị nhiễu bởi thuật toán chưa thật sự tối ưu trong nhiều trường hợp khác nhau.
- Phản cứng xe còn chưa tốt
- Khả năng nhận biết còn hạn chế
- Thuật toán chưa hoàn toàn tối ưu

5.2 HƯỚNG PHÁT TRIỂN

- Tối ưu hóa giải thuật
- Có thể tích hợp thêm trí tuệ nhân tạo để nhận diện biển báo và các cảm biến để xác định vật cản.
- Thử nghiệm ứng dụng vào thực tế.

TÀI LIỆU THAM KHẢO

- [1] <https://www.youtube.com/watch?v=yvfI4p6Wyvk>
- [2] <http://opencv.org/>
- [3] <https://www.instructables.com/>
- [4] <https://vi.wikipedia.org/>
- [5] <https://viblo.asia/p/tuan-1-gioi-thieu-xu-ly-anh-yMnKMdEQ57P>
- [6] <https://docs.python.org/>
- [7] <https://www.raspberrypi.org/documentation/>
- [8] <https://www.3cx.com/docs/installing-pbx-raspberry-pi/>
- [9] <https://www.dexterindustries.com/howto/installing-the-raspberry-pi-camera/>