

VIETNAM NATIONAL UNIVERSITY  
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY  
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



## ĐỒ ÁN THIẾT KẾ LUẬN LÝ (CO3091)

---

Project report

## Xe chạy theo vật thể

---

Instructor: Trần Thanh Bình  
Performers: Dương Đức Nghĩa - 2011671  
Mai Lê Cường - 2022764  
Nguyễn Tuấn Anh - 2012609



## Mục lục

<b>1</b>	<b>Tổng quan</b>	<b>3</b>
1.1	Động lực . . . . .	3
1.2	Mục tiêu . . . . .	3
1.3	Phân công nhiệm vụ . . . . .	3
<b>2</b>	<b>Phase 1: Tìm hiểu đề, chuẩn bị</b>	<b>4</b>
2.1	Yêu cầu đặt ra . . . . .	4
2.2	Linh kiện đã sử dụng . . . . .	4
<b>3</b>	<b>Phase 2: Lắp ráp xe Arduino Uno, làm quen với module motor driver</b>	<b>5</b>
3.1	Lắp ráp xe Arduino Uno . . . . .	5
3.1.1	Chuẩn bị link kiện . . . . .	5
3.1.2	Lắp động cơ và bánh xe vào khung xe . . . . .	5
3.1.3	Kết nối động cơ với module L298N . . . . .	5
3.1.4	Kết nối arduino với module L298N . . . . .	6
3.1.5	Cấp nguồn cho module L298N . . . . .	6
3.1.6	Sơ đồ kết nối arduino . . . . .	6
3.1.7	Nạp code lên arduino, kết nối với arduino với laptop . . . . .	6
3.1.8	Source code để test động cơ . . . . .	6
3.2	Làm quen với Module Motor Driver L298N . . . . .	9
3.2.1	Thông số kỹ thuật . . . . .	9
3.2.2	Sơ đồ board mạch và các chân . . . . .	10
3.2.3	Datasheet . . . . .	11
<b>4</b>	<b>Phase 3: Test thử xe arduino</b>	<b>14</b>
4.1	Vấn đề gặp phải . . . . .	14
4.2	Cách khắc phục . . . . .	14
<b>5</b>	<b>Phase 4: Thực hiện giải thuật nhận diện vật thể với opencv</b>	<b>15</b>
5.1	Tìm hiểu về thư viện opencv . . . . .	15
5.2	Nhận diện vật thể bằng Hough Circle Transform . . . . .	16
5.2.1	Phương trình đường thẳng trong không gian ảnh (A) . . . . .	16
5.2.2	Mapping giữa không gian ảnh (A) và không gian Hough (B) . . . . .	17
5.3	Nhận diện bằng Contour Detection . . . . .	19
5.3.1	Ứng dụng của Contour trong Computer Vision . . . . .	19

5.3.2 Các bước phát hiện và vẽ Contour trong OpenCV . . . . .	20
<b>6 Phase 5: Triển khai trên raspberry</b>	<b>21</b>
6.1 Tìm hiểu về raspberry . . . . .	21
6.1.1 Raspberry là gì . . . . .	21
6.1.2 Cấu hình phần cứng . . . . .	22
6.2 Rasperberry Pi 4 . . . . .	22
6.2.1 Cấu hình chân Raspberry Pi . . . . .	22
6.2.2 Module giao tiếp dữ liệu nối tiếp Raspberry Pi . . . . .	23
6.2.3 Mô tả bảng mạch Raspberry pi 4 . . . . .	24
6.2.4 Các thiết bị ngoại vi chính khác . . . . .	24
6.2.5 Sơ đồ 2D Raspberry Pi . . . . .	25
6.2.6 Nối dây giữa L298N và Raspberry . . . . .	25
<b>7 Phase 6: Code opencv trên máy tính cá nhân</b>	<b>26</b>
7.1 Hướng tiếp cận nhận diện vật thể bằng hough circle transform . . . . .	26
7.2 Hướng tiếp cận nhận diện vật thể bằng contour detection . . . . .	26
7.3 Hướng tiếp cận kết hợp cả 2 phương pháp sử dụng voting . . . . .	27
7.4 Điều khiển hướng so với vật thể . . . . .	27
<b>8 Phase 7: Hoàn thiện sản phẩm</b>	<b>29</b>
8.1 Sơ đồ kết nối dây . . . . .	29
8.2 Thực hiện lại giải thuật trên raspberry . . . . .	29
8.3 Testing . . . . .	33
<b>9 Phase 8 (Bonus): Một số cải tiến so với lần demo trước</b>	<b>33</b>
9.1 Click chọn màu . . . . .	33
9.2 Sử dụng voting . . . . .	33
9.3 Những thay đổi khác . . . . .	34
<b>10 Tổng kết, tự đánh giá</b>	<b>35</b>
<b>11 File đính kèm</b>	<b>36</b>



# 1 Tổng quan

## 1.1 Động lực

- Một trong những lĩnh vực quan trọng của Trí tuệ nhân tạo (Artificial Intelligence) là thị giác máy (Computer Vision). Đây là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh, phát hiện các đối tượng, tạo ảnh, siêu phân giải hình ảnh và nhiều hơn vậy.
- Object Detection đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Nhóm mong muốn rằng có thể ứng dụng Object Detection trong việc thu nhận và xử lý hình ảnh và từ đó điều khiển được động cơ từ những dữ liệu trên.
- Nhằm cung cấp kiến thức về môn Thiết kế luận lý và Mạch điện tử và mong muốn phát triển thêm những kỹ năng liên quan đến phần cứng.

## 1.2 Mục tiêu

- Lắp ráp được xe Arduino có động cơ hoàn chỉnh.
- Kết nối động cơ với Module Driver để từ đó điều khiển được tốc độ và hướng chạy của xe.
- Tích hợp thêm Raspberry và Camera trên thân xe để ghi lại hình ảnh của vật thể và truyền thông tin vào máy tính.
- Viết và thực thi giải thuật OpenCV trên máy tính cá nhân, từ đó có thể tiếp cận để nhận diện thông tin trên màn hình, tín hiệu truyền vào làm cho xe chạy theo vật thể.

## 1.3 Phân công nhiệm vụ

MSSV	Họ và tên	Phân công	Phần trăm hoàn thành
2011671	Dương Đức Nghĩa	Thuật toán, báo cáo	100
2012764	Mai Lê Cường	Lắp ráp, báo cáo	100
2012609	Nguyễn Tuấn Anh	Lắp ráp, báo cáo	100

## 2 Phase 1: Tìm hiểu đề, chuẩn bị

### 2.1 Yêu cầu đặt ra

Lấy ảnh từ Camera, nhận diện vật thể bằng OpenCV hay học máy, điều khiển xe chạy phía sau vật thể.

### 2.2 Linh kiện đã sử dụng

Thiết bị	Công dụng
Khung Xe, Bánh xe	Mô hình để chạy theo vật thể
Arduino Uno R3	Làm quen với động cơ
Raspberry Pi 4 b+	Máy tính mini để thực thi giải thuật OpenCV
Module Driver L298N Red	Điều khiển động cơ
Module Bluetooth	Điều khiển xe bằng bluetooth
Camera Pi v1	Quan sát vật thể
HDMI	Kết nối Raspberry với màn hình
Pin AA Maxcell và 18650	Cung cấp nguồn
Tản nhiệt	Tản nhiệt cho Raspberry

### 3 Phase 2: Lắp ráp xe Arduino Uno, làm quen với module motor driver

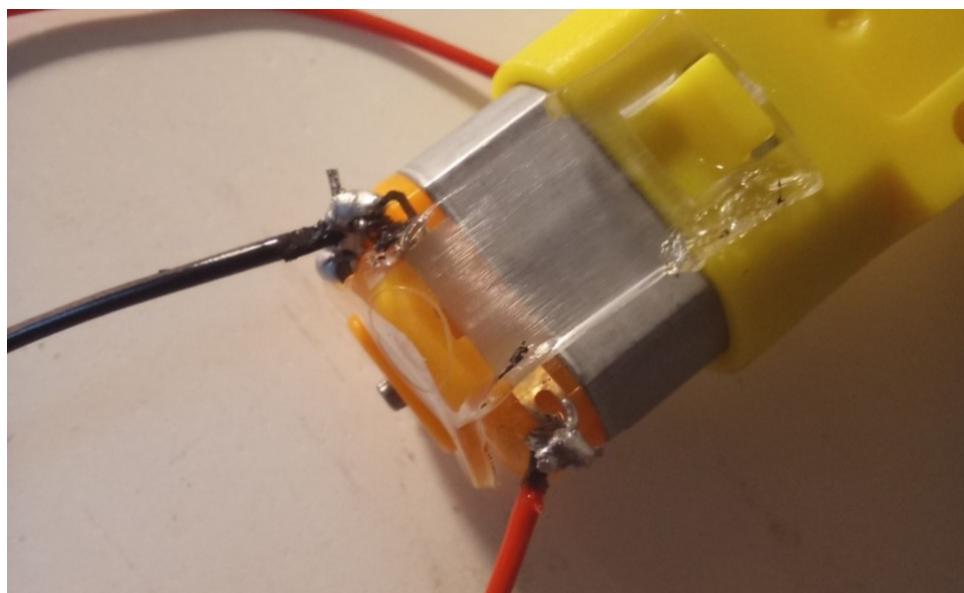
#### 3.1 Lắp ráp xe Arduino Uno

##### 3.1.1 Chuẩn bị link kiện

- 1 Arduino Uno R3
- 1 Arduino Motor Shield L293D
- 4 động cơ DC (màu vàng), 1 tấm Mica
- 2 hộp pin
- 6 viên pin 1,5V
- Các dây cắm

##### 3.1.2 Lắp động cơ và bánh xe vào khung xe

Bước 1: Hàn dây vào động cơ.



Hình 1: Hàn dây vào động cơ

Bước 2: Lắp động cơ vào bánh xe

Bước 3: Gắn bánh xe đã được lắp động cơ vào khung xe

Bước 4: Lắp encoder vào động cơ

##### 3.1.3 Kết nối động cơ với module L298N

Gắn các dây từ động cơ vào các chân OUT1, OUT2, OUT3 và OUT4 để điều khiển động cơ  
Để điều khiển được chiều quay của bánh xe cần chú ý đến giá trị luận lý ở các chân out1, out2, out3, out4

ENA	IN1	IN2	Description
0	x	x	Motor A is off.
1	0	0	Motor breaks and stops
1	0	1	Motor turns forward
1	1	0	Motor turns backward
1	1	1	Motor breaks and stops

Hình 2: Giá trị L298n

### 3.1.4 Kết nối arduino với module L298N

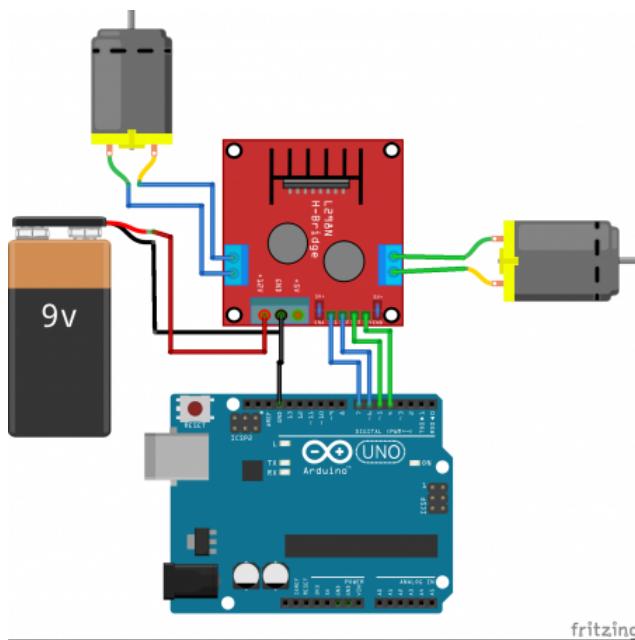
Nối các chân IN1-IN4, tương ứng với chân 7-4 trên arduino uno R3. ENA và ENB kết nối với nguồn để luôn ở mức logic 1.

Kết nối GND của motor driver L298N với GND arduino

### 3.1.5 Cấp nguồn cho module L298N

Kết nối hộp pin 9V: cực dương với đầu vào 9V trên L298N.

### 3.1.6 Sơ đồ kết nối arduino



Hình 3: Sơ đồ kết nối xe arduino

### 3.1.7 Nạp code lên arduino, kết nối với arduino với laptop

### 3.1.8 Source code để test động cơ

```
1 #define IN1 7
2 #define IN2 6
```



```
3 #define IN3 5
4 #define IN4 4
5 #define MAX_SPEED 255
6 #define MIN_SPEED 0
7 void setup()
8 {
9     pinMode(IN1, OUTPUT);
10    pinMode(IN2, OUTPUT);
11    pinMode(IN3, OUTPUT);
12    pinMode(IN4, OUTPUT);
13 }
14
15 void motor_1_Dung() {
16     digitalWrite(IN1, LOW);
17     digitalWrite(IN2, LOW);
18 }
19
20 void motor_2_Dung() {
21     digitalWrite(IN3, LOW);
22     digitalWrite(IN4, LOW);
23 }
24
25 void motor_1_Tien() {
26     digitalWrite(IN1, HIGH);
27     digitalWrite(IN2, LOW);
28 }
29
30 void motor_1_Lui() {
31     digitalWrite(IN1, LOW);
32     digitalWrite(IN2, HIGH);
33 }
34
35 void motor_2_Tien(int speed) {
36     speed = constrain(speed, MIN_SPEED, MAX_SPEED);
37     analogWrite(IN3, speed);
38     digitalWrite(IN4, LOW);
39 }
40
41 void motor_2_Lui(int speed) {
42     speed = constrain(speed, MIN_SPEED, MAX_SPEED);
43     analogWrite(IN4, 255 - speed);
44     digitalWrite(IN3, HIGH);
45 }
46
47 void loop()
48 {
49     motor_1_Tien();
50     digitalWrite(5, 40);
51     delay(1000);
```

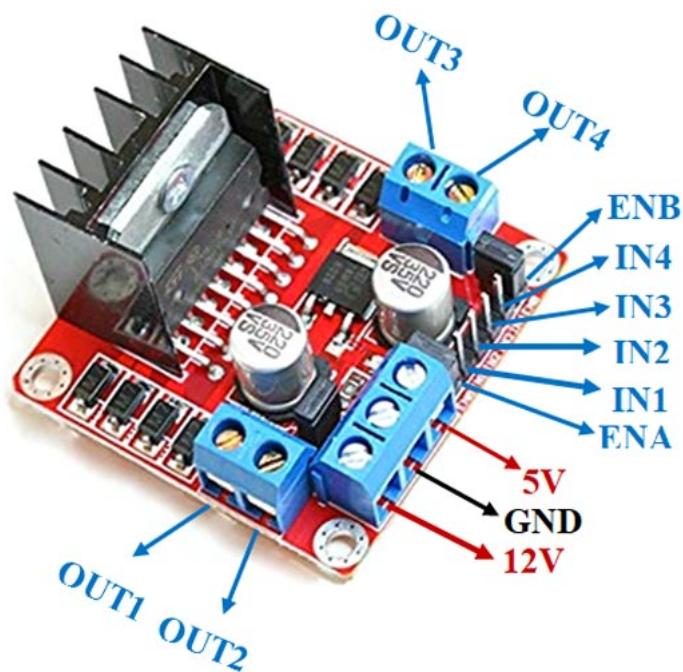


```
52     motor_1_Lui();  
53     digitalWrite(5, 40);  
54     delay(1000);  
55 }
```

### 3.2 Làm quen với Module Motor Driver L298N

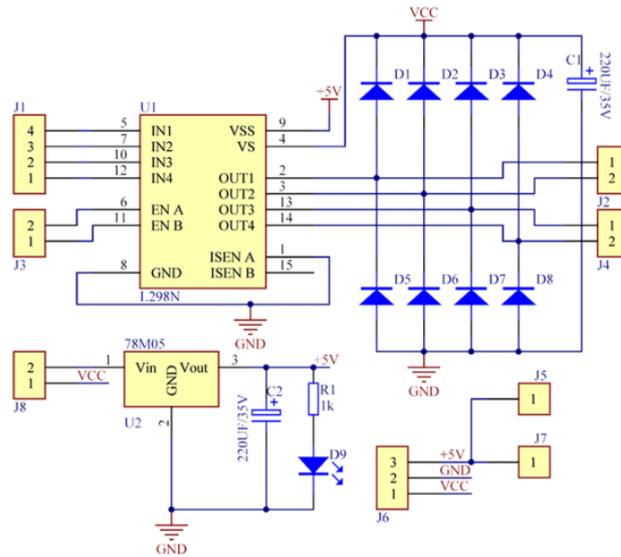
#### 3.2.1 Thông số kỹ thuật

- ENA: Pin được sử dụng để kích hoạt động cơ bên trái
- IN1: Động cơ bên trái 1. đầu vào
- IN2: Động cơ bên trái 2. đầu vào
- IN3: Động cơ bên phải 1. đầu vào
- IN4: Động cơ bên phải 2. đầu vào
- ENB: Chân được sử dụng để kích hoạt kênh động cơ
- MotorA: Đầu ra động cơ bên trái
- MotorB: Đầu ra động cơ bên phải
- VCC: Đầu vào điện áp cung cấp (4.8V-24V)
- GND: Nối đất
- Đầu ra 5V: 5V

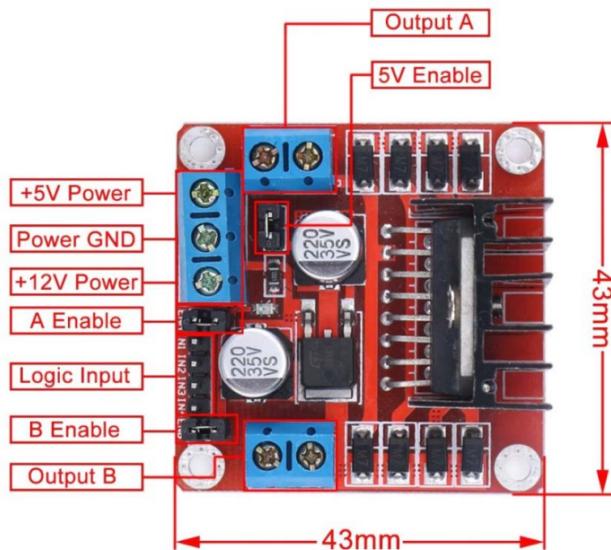


Hình 4: L298N Red Motor Driver

### 3.2.2 Sơ đồ board mạch và các chân

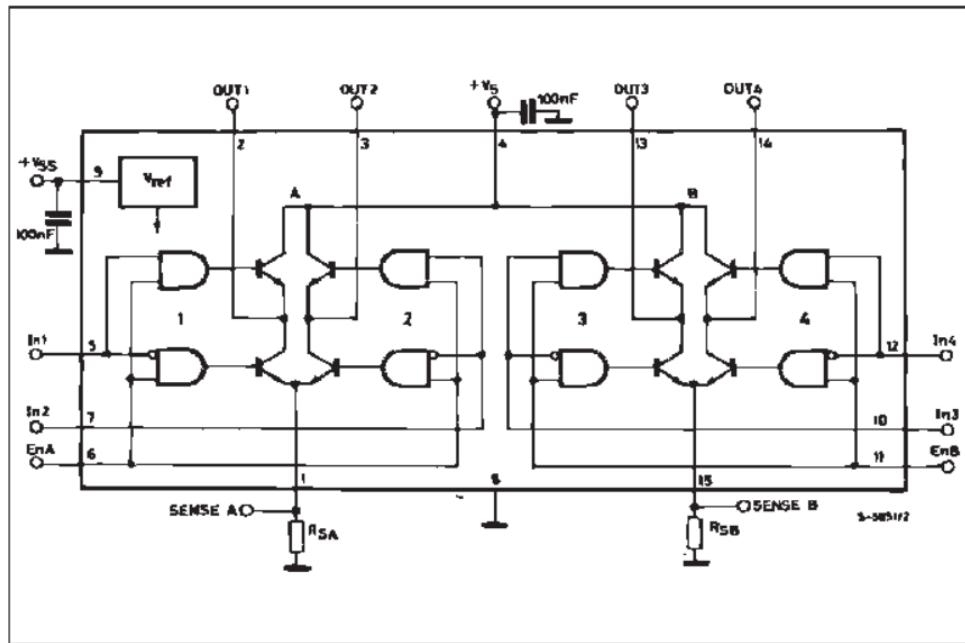


Hình 5: Sơ đồ



Hình 6: Board mạch và các chân

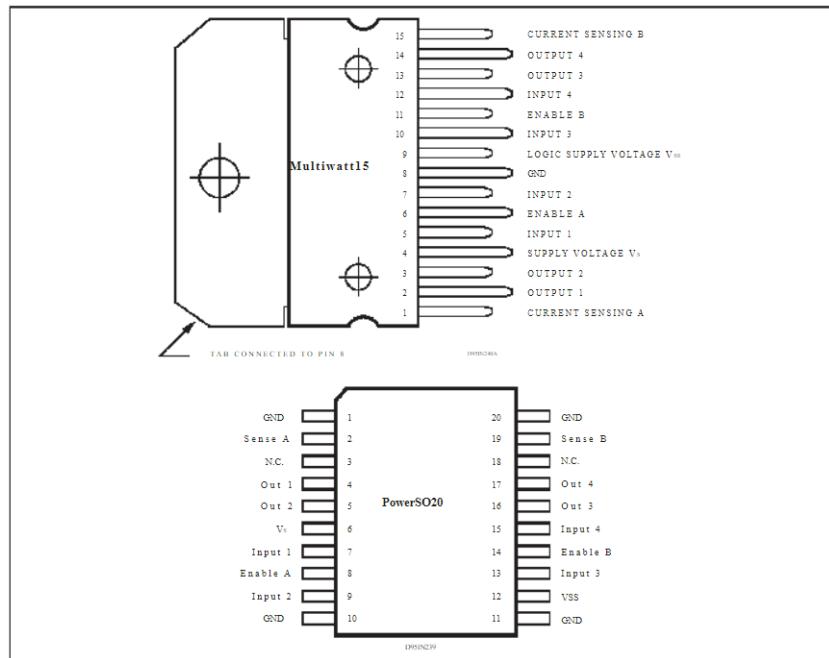
### 3.2.3 Datasheet



Hình 7: Block Diagram

Symbol	Parameter	Value	Unit
V <sub>s</sub>	Power Supply	50	V
V <sub>ss</sub>	Logic Supply Voltage	7	V
V <sub>I,Ven</sub>	Input and Enable Voltage	-0.3 to 7	V
I <sub>O</sub>	Peak Output Current (each Channel)		
	- Non Repetitive ( $t = 100 \mu s$ )	3	A
	-Repetitive (80% on -20% off; $t_{on}=10ms$ )	2.5	A
	-DC Operation	2	A
V <sub>sens</sub>	Sensing Voltage	-1 to 2.3	V
P <sub>tot</sub>	Total Power Dissipation ( $T_{case}=75^{\circ}C$ )	25	W
T <sub>op</sub>	Junction Operating Temperature	-25 to 130	°C
T <sub>sg,Tj</sub>	Storage and Junction Temperature	-40 to 150	°C

Hình 8: Absolute Maximum Ratings



Hình 9: Pin Connections(Top View)

PIN FUNCTIONS (refer to the block diagram)

MW. 15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4;6		V <sub>S</sub>	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A 100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
-	3;18	N.C.	Not Connected

Hình 10: Pin Functions)



ELECTRICAL CHARACTERISTICS ( $V_s = 42V$ ;  $V_{SS} = 5V$ ;  $T_j = 25^{\circ}C$ ; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min .	Typ .	Max.	Unit
$V_s$	Supply Voltage (pin 4)	Operative Condition	$V_{IH}=2.5$	46	7	V
$V_{SS}$	Logic Supply Voltage (pin 9)		4.5	5	7	V
$I_s$	Quiescent Supply Current (pin 4)	$V_{en}=H; I_L=0$ $V_{en}=L$	$V_i=L$ $V_i=H$	13 50	22 70	mA mA
$I_{SS}$	Quiescent Current from $V_{SS}$ (pin 9) $V_{en}=H; I_L=0$	$V_{en}=L$	$V_i=X$		4	mA
		$V_{en}=L$	$V_i=L$ $V_i=H$	24 7	36 12	mA mA
		$V_{en}=L$	$V_i=X$		6	mA
$V_{IL}$	Input Low Voltage (pins 5, 7, 10, 12)		-0.3		1.5	V
$V_{IH}$	Input High Voltage (pins 5, 7, 10, 12)		2.3		$V_{SS}$	V
$I_{IL}$	Low Voltage Input Current (pins 5, 7, 10, 12)	$V_i=L$			-10	$\mu A$
$I_{IH}$	High Voltage Input Current (pins 5, 7, 10, 12)	$V_i = H \leq V_{SS}-0.6V$		30	100	$\mu A$
$V_{en}=L$	Enable Low Voltage (pins 6, 11)		-0.3		1.5	V
$V_{en}=H$	Enable High Voltage (pins 6, 11)		2.3		$V_{SS}$	V
$I_{en}=L$	Low Voltage Enable Current (pins 6, 11)	$V_{en}=L$			-10	$\mu A$
$I_{en}=H$	High Voltage Enable Current (pins 6, 11)	$V_{en}=H \leq V_{SS}-0.6V$		30	100	$\mu A$
$V_{CE(H)}$	Source Saturation Voltage	$I_L=1A$ $I_L=2A$	0.95	1.35 2	1.7 2.7	V
$V_{CE(L)}$	Saturation Voltage	$I_L=1A$ (5) $I_L=2A$ (5)	0.85	1.2 1.7	1.6 2.3	V
$V_{CEst}$	Total Drop	$I_L=1A$ (5) $I_L=2A$ (5)	1.80		3.2 4.9	V
$V_{ism}$	Sensing Voltage (pins 1, 15)		-1 (1)		2	V

Hình 11: Electrical Characteristics (1)

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min .	Typ .	Max.	Unit
$t_{1(Vi)}$	Source Current Turn-off Delay	0.5 V to 0.9 $I_L$ (2); (4)		1.5		$\mu s$
$t_{12(Vi)}$	Source Current Fall Time	0.9 $I_L$ to 0.1 $I_L$ (2); (4)		0.2		$\mu s$
$t_{3(Vi)}$	Source Current Turn-on Delay	0.5 V to 0.1 $I_L$ (2); (4)		2		$\mu s$
$t_{4(Vi)}$	Source Current Rise Time	0.1 $I_L$ to 0.9 $I_L$ (2); (4)		0.7		$\mu s$
$t_{5(Vi)}$	Sink Current Turn-off Delay	0.5 V to 0.9 $I_L$ (3); (4)		0.7		$\mu s$
$t_{6(Vi)}$	Sink Current Fall Time	0.9 $I_L$ to 0.1 $I_L$ (3); (4)		0.25		$\mu s$
$t_{7(Vi)}$	Sink Current Turn-on Delay	0.5 V to 0.9 $I_L$ (3); (4)		1.6		$\mu s$
$t_{8(Vi)}$	Sink Current Rise Time	0.1 $I_L$ to 0.9 $I_L$ (3); (4)		0.2		$\mu s$
$f_c(Vi)$	Commutation Frequency	$I_L=2A$		25	40	KHz
$t_{11(Ven)}$	Source Current Turn-off Delay	0.5 V to 0.9 $I_L$ (2); (4)		3		$\mu s$
$t_{12(Ven)}$	Source Current Fall Time	0.9 $I_L$ to 0.1 $I_L$ (2); (4)		1		$\mu s$
$t_{13(Ven)}$	Source Current Turn-on Delay	0.5 V to 0.1 $I_L$ (2); (4)		0.3		$\mu s$
$t_{14(Ven)}$	Source Current Rise Time	0.1 $I_L$ to 0.9 $I_L$ (2); (4)		0.4		$\mu s$
$t_{15(Ven)}$	Sink Current Turn-off Delay	0.5 V to 0.9 $I_L$ (3); (4)		2.2		$\mu s$
$t_{16(Ven)}$	Sink Current Fall Time	0.9 $I_L$ to 0.1 $I_L$ (3); (4)		0.35		$\mu s$
$t_{17(Ven)}$	Sink Current Turn-on Delay	0.5 V to 0.9 $I_L$ (3); (4)		0.25		$\mu s$
$t_{18(Ven)}$	Sink Current Rise Time	0.1 $I_L$ to 0.9 $I_L$ (3); (4)		0.1		$\mu s$

1) 1)Sensing voltage can be -1 V for  $t \leq 50 \mu sec$ , in steady state  $V_{min} \geq -0.5V$ .

2) See fig. 2.

3) See fig. 4.

4) The load must be a pure resistor.

Hình 12: Electrical Characteristics (continue)



## 4 Phase 3: Test thử xe arduino

### 4.1 Vấn đề gặp phải

- 2 bánh xe chạy không cùng chiều
- 2 bánh xe chạy yếu, không đủ để demo

### 4.2 Cách khắc phục

- Đổi lại gắn dây cho 2 động cơ nối tiếp với nhau.
- Khắc phục 2 bánh xe chạy yếu bằng cách nối tiếp thêm nguồn pin để gia tăng lượng điện năng cung cấp cho động cơ.

## 5 Phase 4: Thực hiện giải thuật nhận diện vật thể với opencv

### 5.1 Tìm hiểu về thư viện opencv



- OpenCV là dạng viết tắt của cụm từ “Open Source Computer Vision Library” – một thư viện nguồn mở dành cho Machine Learning và Computer Vision. OpenCV được tạo ra để phân tích và học máy theo thời gian thực cũng như xử lý hình ảnh, video. Hiện bộ công cụ OpenCV còn được bổ sung tính năng tăng tốc GPU theo real-time. Đối tượng chính của OpenCV đó là các ứng dụng theo thời gian thực (real-time applications).
- Ưu điểm OpenCV đó là nó cung cấp hơn 2.500 thuật toán cổ điển và hiện đại, tất cả đều được tối ưu hóa cho học máy và thị giác máy tính. Thư viện nguồn mở này có giấy phép BSD, vì vậy nó được phát hành miễn phí dành cho cả mục đích học tập và thương mại.
- OpenCV tương thích với nhiều nền tảng như Windows, macOS, Linux, Android, iOS và các ngôn ngữ lập trình khác nhau như Java, Python, C hay C++. Thư viện này có khả năng tận dụng tối đa khả năng sẵn có của phần cứng, từ đó đảm bảo các ứng dụng máy tính đạt được hiệu suất cao nhất.
- OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau bao gồm:
  - Hình ảnh street view
  - Kiểm tra và giám sát tự động
  - Robot và xe hơi tự lái
  - Phân tích hình ảnh y học
  - Tìm kiếm và phục hồi hình ảnh/video
  - Phim – cấu trúc 3D từ chuyển động
  - Nghệ thuật sắp đặt tương tác
- Những cấu trúc module của opencv (gói có chứa các static libraries (thư viện liên kết tĩnh) hoặc shared libraries (thư viện liên kết động))
  - Core functionality (core): Đây là module nhỏ gọn cho phép xác định các cấu trúc dữ liệu cơ bản. Trong đó bao gồm những tính năng cơ bản mà tất cả các module khác đều cần đến.
  - Image Processing (imgproc): Imgproc là module xử lý hình ảnh của OpenCV. Trong đó có chuyển đổi không gian màu, phép biến đổi hình học, lọc hình ảnh tuyến tính và phi tuyến, biểu đồ...
  - Video Analysis (video): Trong module phân tích video, bạn có thể tìm thấy các thuật toán như tách nền, tính toán chuyển động, theo dõi vật thể...
  - Camera Calibration and 3D Reconstruction (calib3d): Module này bao gồm các thuật toán liên như tái tạo 3D, dự đoán kiểu dáng, hình học đa chiều cơ bản, thư tín âm thanh nổi...
  - Object Detection (objdetect): Module này cho phép nhận dạng đối tượng và mô phỏng các predefined classes (hàm được định nghĩa sẵn)...
  - 2D Features Framework (features2d): Đây là module chứa các thuật toán phát hiện các đặc trưng của bộ nhận diện, thông số đối chơi, bộ truy xuất thông số...

- Object Detection (objdetect): phát hiện các đối tượng và mô phỏng của các hàm được định nghĩa sẵn – predefined classes (vd: khuôn mặt, mắt, cốc, con người, xe hơi,...).
  - High-level GUI (highgui): giao diện dễ dùng để thực hiện việc giao tiếp UI đơn giản.
  - Video I/O (videoio): giao diện dễ dùng để thu và mã hóa video.
  - GPU: Các thuật toán tăng tốc GPU từ các modun OpenCV khác.
  - Các module hỗ trợ khác, ví dụ như FLANN và Google test wrapper, Python binding,...
- Các ngôn ngữ lập trình hỗ trợ OpenCV:
    - C++: Ngôn ngữ này cung cấp nhiều option và được sử dụng phổ biến. Nếu bạn sử dụng Visual Studio làm IDE thì nên sử dụng C++. Mặc dù khi tiếp xúc ban đầu khá phức tạp tuy nhiên các thiết lập của C++ đem đến nhiều lợi ích khi phát triển sản phẩm trong tương lai.
    - Python: OpenCV-Python có ưu điểm là không cần thiết lập nhiều, ngắn gọn. Vì vậy thường được sử dụng để test OpenCV. Python cũng cho phép lập trình trên nhiều hệ điều hành.
    - Android: Có thể trở thành xu hướng nhờ sự tiện lợi và được tích hợp sẵn camera.
    - Java: Khá giống với C++ với thế mạnh là tốc độ nhanh chóng và tính đa nền tảng.
    - C#: Ưu điểm của C# là được hỗ trợ bởi thư viện đa nền tảng EmguCV, cho phép code nhanh chóng và tiện lợi.

## 5.2 Nhận diện vật thể bằng Hough Circle Transform

Việc nhận diện bằng hough circle transform có phương pháp khá tương đồng với hough line transform nên nhóm sẽ trình bày giải thuật đằng sau hough line transform

Ý tưởng chung của việc phát hiện đường thẳng trong thuật toán này là tạo mapping từ không gian ảnh (A) sang một không gian mới (B) mà mỗi đường thẳng trong không gian (A) sẽ ứng với một điểm trong không gian (B).

### 5.2.1 Phương trình đường thẳng trong không gian ảnh (A)

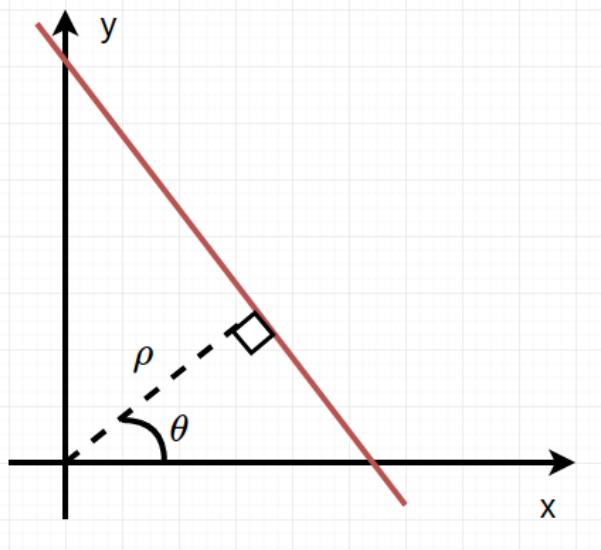
Như đã học ở cấp 2, phương trình đường thẳng cơ bản sẽ được biểu diễn theo 2 tham số a và b như sau:

$$y = ax + b \quad (1)$$

Tuy nhiên, với cách biểu diễn này, giá trị của a góc nghiêng trải dài từ  $-\infty$  đến  $+\infty$ . Thuật toán Hough Transform yêu cầu các giá trị a, b nằm trong một khoảng xác định (hay bị chặn trên dưới), ta phải sử dụng hệ tọa độ cực để biểu diễn phương trình đường thẳng. Cách biểu diễn này cũng nằm trong chương trình toán trung học:

$$p = x\cos(\theta) + y\sin(\theta) \quad (2)$$

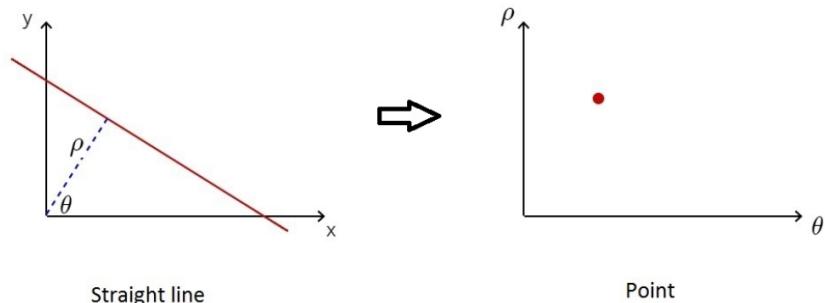
Xét thấy trong phương trình tọa độ cực, giá trị của góc  $\theta$  có thể bị chặn lại trong khoảng  $[0, \pi]$ . Trên thực tế, không gian ảnh là không gian hữu hạn (bị chặn lại bởi các cạnh của ảnh), do vậy giá trị p cũng bị chặn.



Hình 13: Đường thẳng trong hệ tọa độ cực

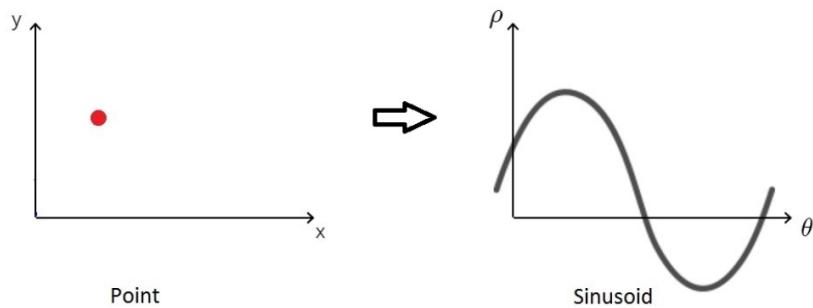
### 5.2.2 Mapping giữa không gian ảnh (A) và không gian Hough (B)

Từ một đường thẳng trong không gian ảnh (A) với 2 tham số  $\rho$  và  $\theta$ , chúng ta sẽ map sang không gian Hough (B) thành một điểm.



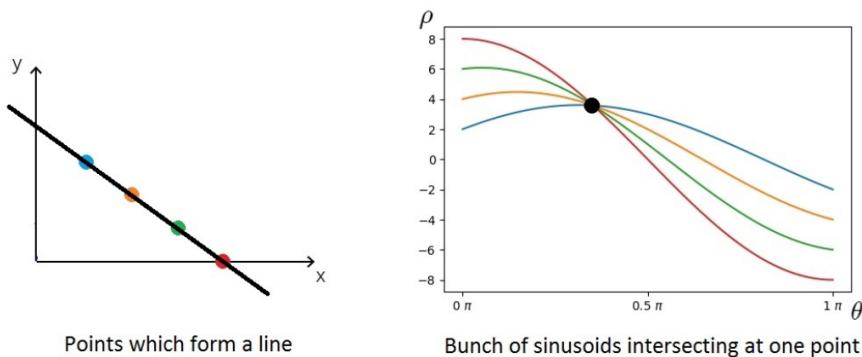
Hình 14: Mapping một đường thẳng từ không gian ảnh sang không gian Hough

Từ một điểm trong không gian ảnh, chúng ta lại có được một hình sin trong không gian Hough:



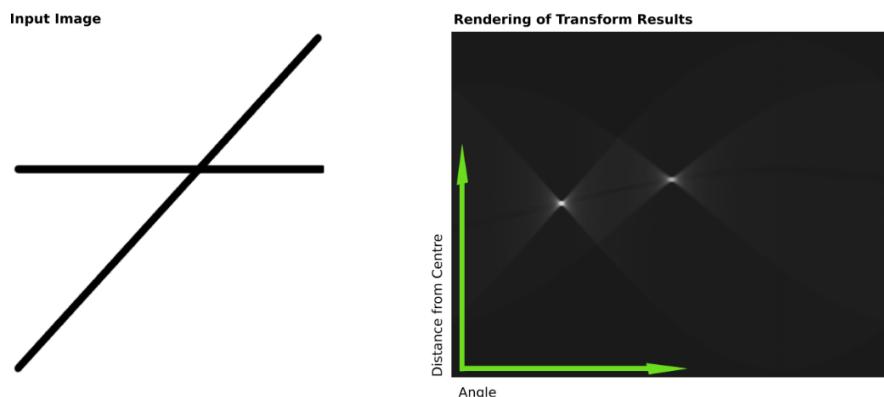
Hình 15: Mapping một điểm từ không gian ảnh sang không gian Hough

Các điểm nằm trên cùng một đường thẳng lại có biểu diễn là các hình sin giao nhau tại một điểm trong không gian Hough. Đây là nơi xuất phát ý tưởng của thuật toán Hough Transform. Chúng ta sẽ dựa vào các điểm giao nhau này để suy ngược lại phương trình đường thẳng trong không gian ảnh.



Hình 16: Mapping nhiều điểm thẳng hàng từ không gian ảnh sang không gian Hough

Mỗi đường thẳng khác nhau sẽ tạo thành một điểm sáng (nơi giao nhau của nhiều hình sin) trên không gian Hough. Dưới đây là sự biểu diễn 2 đường thẳng trong không gian Hough.



Hình 17: Biểu diễn 2 đường thẳng trong không gian Hough

### 5.3 Nhận diện bằng Contour Detection

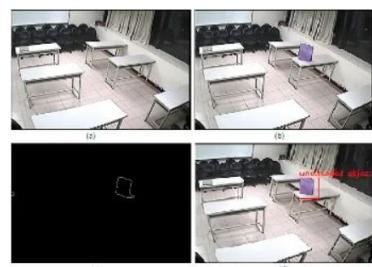
#### 5.3.1 Ứng dụng của Contour trong Computer Vision

Phát hiện chuyển động : Trong video giám sát, công nghệ phát hiện chuyển động có nhiều ứng dụng, từ môi trường an ninh trong nhà và ngoài trời, kiểm soát giao thông, phát hiện hành vi trong các hoạt động thể thao, phát hiện các đối tượng không được giám sát và thậm chí nén video.



Hình 18: Một ví dụ về phát hiện đối tượng chuyển động

Phát hiện đối tượng không giám sát: Bất kỳ đối tượng không giám sát nào ở những nơi công cộng thường được coi là đối tượng đáng ngờ.



Hình 19: Nhận dạng và đánh dấu đối tượng không được giám sát

Phân đoạn nền/ tiền cảnh : Để thay thế nền của một hình ảnh bằng một hình ảnh khác, bạn cần thực hiện trích xuất hình ảnh-tiền cảnh (tương tự như phân đoạn hình ảnh). Sử dụng các đường viền là một cách tiếp cận có thể được sử dụng để thực hiện phân đoạn.



Hình 20: Trích xuất nền và thêm nền mới vào hình ảnh



### 5.3.2 Các bước phát hiện và vẽ Contour trong OpenCV

#### 1. Đọc hình ảnh và chuyển đổi nó sang định dạng thang độ xám

Đọc hình ảnh và chuyển đổi hình ảnh sang định dạng thang độ xám. Chuyển đổi hình ảnh sang thang độ xám là rất quan trọng vì nó chuẩn bị hình ảnh cho bước tiếp theo. Việc chuyển đổi hình ảnh thành một hình ảnh thang độ xám của một kênh rất quan trọng đối với việc tạo ngưỡng, điều này lại cần thiết để thuật toán phát hiện đường viền hoạt động bình thường.

#### 2. Áp dụng ngưỡng nhị phân

Trong khi tìm các đường viền, trước tiên hãy luôn áp dụng ngưỡng nhị phân hoặc phát hiện cạnh Canny cho hình ảnh thang độ xám. Ở đây, chúng tôi sẽ áp dụng ngưỡng nhị phân.

Điều này chuyển đổi hình ảnh thành đen trắng, làm nổi bật các đối tượng quan tâm để giúp mọi thứ dễ dàng cho thuật toán phát hiện đường viền. Ngưỡng biến đường viền của đối tượng trong ảnh thành màu trắng hoàn toàn, với tất cả các pixel có cùng cường độ. Giờ đây, thuật toán có thể phát hiện đường viền của các đối tượng từ các pixel trắng này.

#### 3. Tìm đường viền

Sử dụng findContours() chức năng để phát hiện các đường viền trong hình ảnh.

#### 4. Vẽ các đường viền trên hình ảnh RGB gốc

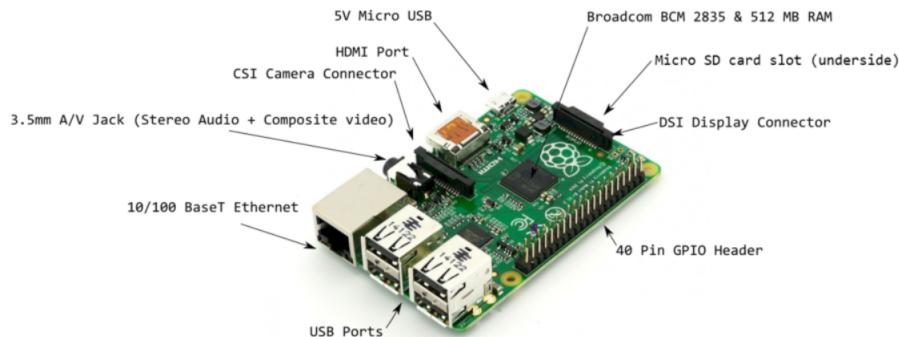
Khi các đường viền đã được xác định, hãy sử dụng drawContours() chức năng để phủ các đường viền lên hình ảnh RGB gốc.

## 6 Phase 5: Triển khai trên raspberry

### 6.1 Tìm hiểu về raspberry

#### 6.1.1 Raspberry là gì

- Raspberry Pi là chiếc máy tính kích thước nhỏ được tích hợp nhiều phần cứng mạnh mẽ đủ khả năng chạy hệ điều hành và cài đặt được nhiều ứng dụng trên nó. Với giá chỉ vài chục USD, Raspberry hiện đang là mini computer nổi bật nhất hiện nay.
- Raspberry Pi sản xuất bởi 3 OEM: Sony, Qsida, Egoman. Và được phân phối chính bởi Element14, RS Components và Egoman.
- 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 /video , và các tính năng khác ... tất cả được tích hợp bên trong chip có điện năng thấp 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 . Tuy nhiên Raspberry Pi là một thiết bị đa năng đáng ngạc nhiên với nhiều phần cứng có giá thành rẻ nhưng rất hoàn hảo cho những hệ thống điện tử , những dự án DIY , thiết lập hệ thống tính toán rẻ tiền cho những bài học trải nghiệm lập trình ...



Hình 21: Hình ảnh về Raspberry

### 6.1.2 Cấu hình phần cứng

Phiên bản Raspberry Pi đầu tiên được phát hành tháng 2 năm 2012, và tới nay đã có nhiều phiên bản khác nhau, với sự nâng cấp của phần cứng, cũng như hướng tới những mục tiêu khác nhau.

Raspberry Pi	Model A	Model B+	Pi 2, Model B	Pi 3, Model B	Pi Zero
Vi xử lý	Broadcom BCM2835, ARMv6 (32bit) single core	Broadcom BCM2836, ARMv7 (32bit) quad core	Broadcom BCM2837, ARMv8 (64bit) quad-core	Broadcom BCM2835, ARM11 (32bit) single core	
GPU		Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p30, 250 MHz		Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p60, 400 MHz	Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p30, 250 MHz
Tốc độ xử lý	700 Mhz	900 Mhz	1.2 Ghz	1.0 Ghz	
Power Ratings	200mA @ 5V	600mA @ 5V	800mA @ 5V	800mA @ 5V	160mA @ 5V
RAM (chia sẻ với GPU)	256 MB SDRAM (400Mhz)	512 MB SDRAM (400Mhz)	1 GB SDRAM (400MHz)	1GB LPDDR2 (900Mhz)	512 MB SDRAM (400MHz)
Bộ nhớ		Micro SD			
GPIO		40			
Kết nối	1xUSB 2.0 CSI - cổng camera DSI - cổng kết nối màn hình cảm ứng	4xUSB 2.0 10/100mb Ethernet CSI, DSI	4xUSB 2.0 10/100mb Ethernet wifi 802.11 n Bluetooth 4.1 CSI, DSI		1 micro USB
Video & audio		1080p HDMI, stereo audio 3.5mm jack		1080p mini HDMI, stereo audio through PWM on GPIO	
Kích thước	65x56mm		85x56mm		65x30mm

Hình 22: Cấu hình phần cứng của Raspberry

## 6.2 Rasperberry Pi 4

### 6.2.1 Cấu hình chân Raspberry Pi

- Raspberry Pi 4 có thẻ sử dụng trong hệ thống nhúng bên ngoài để giao tiếp tín hiệu. Có tổng cộng 40 chân , trong đó 28 chân là chân GPIO và các chân còn lại là chân nguồn
- Các chân GPIO không chỉ thực hiện các chức năng I / O đơn giản mà còn hỗ trợ giao thức UART, SPI và I 2 C. Các giao thức này dành riêng cho mọi chân và tất cả chức năng của chúng được thảo luận dưới đây:
- Power In: Trong Raspberry pi, có hai cách cấp nguồn, một là từ cổng nguồn USB-C và thứ hai là từ các chân 5V. Chân 5V được kết nối trực tiếp với cổng adapter USB-C
- Đầu vào ở chân 5V phải ổn định và theo đúng thông số kỹ thuật. Trong trường hợp có điện áp cao hơn, thiết bị có thể bị cháy
- Các chân đầu vào 5V sẽ không có cầu chì và bộ điều chỉnh điện áp khi được sử dụng làm đầu vào cấp nguồn, do đó nguồn điện 5V phải theo đúng thông số kỹ thuật để tránh hư hại. Chân đầu vào cấp nguồn cho Raspberry Pi 4 được cung cấp bên dưới: Từ chân 2 đến 6 -> + 5V, chân 6 —> GND.
- Ground: Raspberry Pi 4 có nhiều chân ground được kết nối bên trong và các chân này này có thể làm điểm nối đất chung cho nguồn điện hoặc thiết bị bên ngoài. Danh sách các chân ground được đưa ra dưới đây: 6,9,14,20,25,30,34,39.

	FUNCTION	PIN	PIN	FUNCTION
<b>3V3</b>		1	2	<b>5V</b>
<b>GPIO2</b>	SPI3 MOSI/SDA3	3	4	<b>5V</b>
<b>GPIO3</b>	SPI3 SCLK/SCL3	5	6	<b>GND</b>
<b>GPIO4</b>	SPI4 CE0 N/SDA3	7	8	TxD1/SPI5 MOSI
<b>GND</b>	GND	9	10	RxD1/SPI5 SCLK
<b>GPIO17</b>	SPI6 CE1 N	11	12	SPI6 CEO N
<b>GPIO27</b>	SDA6	13	14	<b>GND</b>
<b>GPIO22</b>	<b>3V3</b>	15	16	SCL6
<b>GPIO10</b>	SDA5	17	18	SPI3 CE1 N
<b>GPIO9</b>	RxD4/SCL4	19	20	<b>GND</b>
<b>GPIO11</b>	SCL5	21	22	SPI4 CE1 N
<b>GND</b>	GND	23	24	SDA4/TxD4
<b>GPIO0</b>		25	26	SCL4/SPI4 SCLK
<b>GPIO5</b>	SPI3 CE0 N/TxD2/SDA6	27	28	SPI3 MISO/SCL6/RxD2
<b>GPIO6</b>	SPI4 MISO/RxD3/SCL3	29	30	<b>GND</b>
<b>GPIO13</b>	SPI4 MOSI/SDA4	31	32	SDA5/SPI5 CEO N/TxD5
<b>GPIO19</b>	SPI5 MISO/RxD5/SCL5	33	34	<b>GND</b>
<b>GPIO26</b>	SPI6 MISO	35	36	SPI1 CE2 N
<b>GND</b>	SPI5 CE1 N	37	38	SPI6 MOSI
		39	40	SPI6 SCLK
<b>I2C</b>				<b>Ground</b>
<b>UART</b>				<b>5V Power</b>
<b>SPI</b>				<b>3V3 Power</b>

Hình 23: Sơ đồ chân GPIO của Raspberry Pi 4

- Hầu hết mọi thiết bị đều cần các chân đầu vào và đầu ra để giao tiếp. Trong thiết bị này có 28 chân GPIO được sử dụng làm đầu vào và đầu ra digital. Các chân GPIO trong bộ điều khiển có một số giá trị mặc định.
- Các chân GPIO từ 0-9 sẽ ở trạng thái logic cao và từ 10 trở lên các chân sẽ ở trạng thái logic thấp.
- Trong Raspberry Pi 4 có một khe cắm thẻ SD nhưng các cài đặt GPIO cũng hỗ trợ khả năng tương thích với thẻ SD. Chân SDIO trên thiết bị có thể được sử dụng cho thẻ SD khi có ứng dụng yêu cầu:
- SPI (Serial Peripheral Interface) là một giao thức được sử dụng cho giao tiếp chủ-tớ. Raspberry Pi sử dụng giao thức này để giao tiếp nhanh chóng giữa một hoặc nhiều thiết bị ngoại vi. Dữ liệu được đồng bộ hóa bằng đồng hồ (SCLK tại chân GPIO 11) từ thiết bị chính (Raspberry Pi) và dữ liệu được gửi từ Raspberry Pi tới thiết bị SPI bằng chân MOSI (Master Out Slave In). Nếu thiết bị SPI cần giao tiếp lại với Raspberry Pi, thì nó sẽ gửi dữ liệu trả lại bằng chân MISO (Master In Slave Out).
- Các chân GPIO chịu mức điện áp tối thiểu là 3V, tối đa là 5V. Nếu mức điện áp nằm ngoài giới hạn này sẽ làm hỏng mạch. Thông thường, chúng ta nên sử dụng một mạch mở rộng để kết nối với các thiết bị ngoại vi, không nên kết nối trực tiếp các thiết bị, linh kiện điện tử vào GPIO trên Raspberry Pi.
- PWM (Pulse Width Modulation – Điều chế độ rộng xung) là một kỹ thuật phổ biến được sử dụng để thay đổi độ rộng của các xung trong một chuỗi xung. PWM có nhiều ứng dụng như điều khiển độ sáng của đèn LED, điều khiển tốc độ của động cơ DC, điều khiển động cơ servo hoặc nơi bạn phải lấy ngõ ra analog bằng các thiết bị kỹ thuật số.

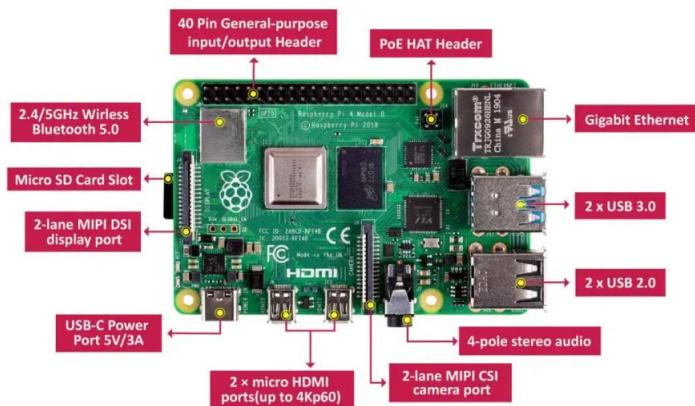
### 6.2.2 Module giao tiếp dữ liệu nối tiếp Raspberry Pi

- Có nhiều giao thức nối tiếp và UART là một trong số đó. Nó khá phổ biến vì có hệ thống giao tiếp đơn giản và phụ thuộc vào hầu hết các phần mềm.
- Để tạo ra tín hiệu đầu ra xung mong muốn Raspberry Pi 4 có một số chân PWM. Các chân này cấp dữ liệu trực tiếp cho các thiết bị ngoại vi điện áp thấp. Để tạo ra tín hiệu đầu tiên các chân phải được lập trình trước.

- Trong Raspberry Pi 4 có một khe cắm thẻ SD nhưng các cổng GPIO cũng hỗ trợ khả năng tương thích với thẻ SD. Chân SDIO trên thiết bị có thể được sử dụng cho thẻ SD khi có ứng dụng yêu cầu.

### 6.2.3 Mô tả bảng mạch Raspberry pi 4

- Raspberry Pi 4 có nhiều hệ thống liên lạc hiện đại. Nó có WiFi bên trong và Bluetooth để giao tiếp dữ liệu không dây. Có thể được sử dụng bên trong hệ thống ở bất cứ đâu mà không có bất kỳ sự xung đột nào.
- Pi chuyển đổi dữ liệu dễ dàng trong cùng một mạng do có WiFi. Thiết bị cũng có hỗ trợ mạng LAN trong trường hợp không có WiFi và là mạng truyền thông có dây.

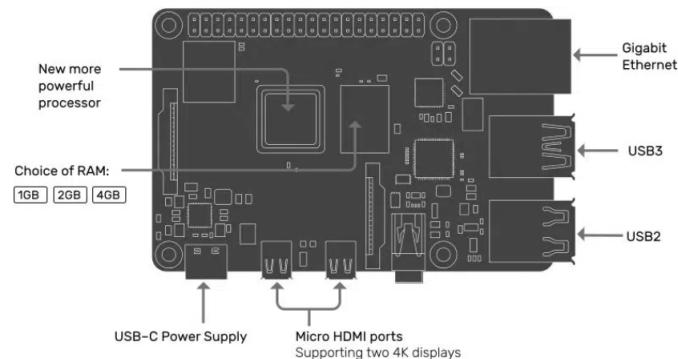


Hình 24: Bảng mạch của Raspberry Pi 4

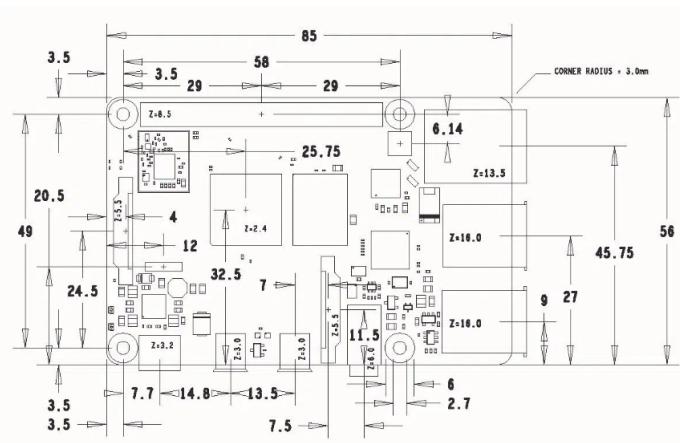
### 6.2.4 Các thiết bị ngoại vi chính khác

- USB: Có bốn cổng USB trong Raspberry Pi 4. Hai cổng hỗ trợ 2.0 nhưng hai cổng còn lại là 3.0. USB 3.0 này cho phép người dùng truyền dữ liệu nhanh.
- PoE Header: Do việc phổ biến sử dụng Raspberry Pi trong IoT và các dự án thông minh khác, PoE Header cũng đã xuất hiện trong Pi. PoE là một bo mạch cho phép người dùng dùng cáp điện cho thiết bị thông qua Ethernet Wire. Trong trường hợp dùng PoE, một PoE HAT bên ngoài sẽ được yêu cầu.
- Máy ảnh: Thiết bị có hỗ trợ máy ảnh. Có một cổng máy ảnh MIPI CSI 2-lane có thể được sử dụng để kết nối pi trực tiếp với máy ảnh và sử dụng mà không cần các giao tiếp thứ ba nào.
- Màn hình: Raspberry Pi 4 có thể được kết nối với màn hình LCD bên ngoài. Không sử dụng header mở rộng để giao tiếp với màn hình LCD như các thiết bị khác. Nó có một cổng MIPI DSI 2-lane riêng biệt có thể được sử dụng để giao tiếp với màn hình LCD tương thích bên ngoài.
- Âm thanh: Dữ liệu âm thanh có thể được truyền từ pi đến thiết bị display thông qua HDMI nhưng nó có một cổng âm thanh 4 cực riêng biệt có thể được sử dụng để gửi và nhận tín hiệu âm thanh. Tín hiệu từ thiết bị có thể được sử dụng bởi chương trình bên trong hoặc các thiết bị khác kết nối ở header mở rộng.
- THẺ SD: Đây là phần cần thiết nhất của Raspberry Pi. Hệ điều hành của Pi sẽ được lưu trong thẻ SD và sau đó được sử dụng thông qua khe cắm Thẻ SD.

### 6.2.5 Sơ đồ 2D Raspberry Pi



Hình 25: Mô hình 2D vật lý của RPi (1)



Hình 26: Mô hình 2D vật lý của RPi (2)

### 6.2.6 Nối dây giữa L298N và Raspberry

IN1	18
IN2	16
IN3	15
IN4	13
PWMA	32
PWMB	33

## 7 Phase 6: Code opencv trên máy tính cá nhân

### 7.1 Hướng tiếp cận nhận diện vật thể bằng hough circle transform

```
1 def hough_circle(img):
2     def dist(x1, y1, x2, y2):
3         return (x1 - x2) ** 2 * +(y1 - y2) ** 2
4
5     prev_circle = None
6     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
7     blur = cv2.GaussianBlur(gray, (17, 17), 0)
8     blur2 = cv2.medianBlur(gray, 5)
9     rows = gray.shape[0]
10    circles = cv2.HoughCircles(blur2, cv2.HOUGH_GRADIENT, 1,
11                                minDist=rows / 8, param1=210, param2=37,
12                                minRadius=20, maxRadius=150)
13    if circles is not None:
14        circles = np.uint16(np.around(circles))
15        chosen = None
16        for i in circles[0, :]:
17            if chosen is None:
18                chosen = i
19            if prev_circle is not None:
20                if dist(chosen[0], chosen[1], prev_circle[0], prev_circle
21                        [1]) <= dist(i[0], i[1], prev_circle[0],
22                                     prev_circle[1]):
23                    chosen = i
24        cv2.circle(img, (chosen[0], chosen[1]), 1, (0, 100, 100), 3)
25        cv2.circle(img, (chosen[0], chosen[1]), chosen[2], (0, 0, 255),
26                   3)
27        prev_circle = chosen
```

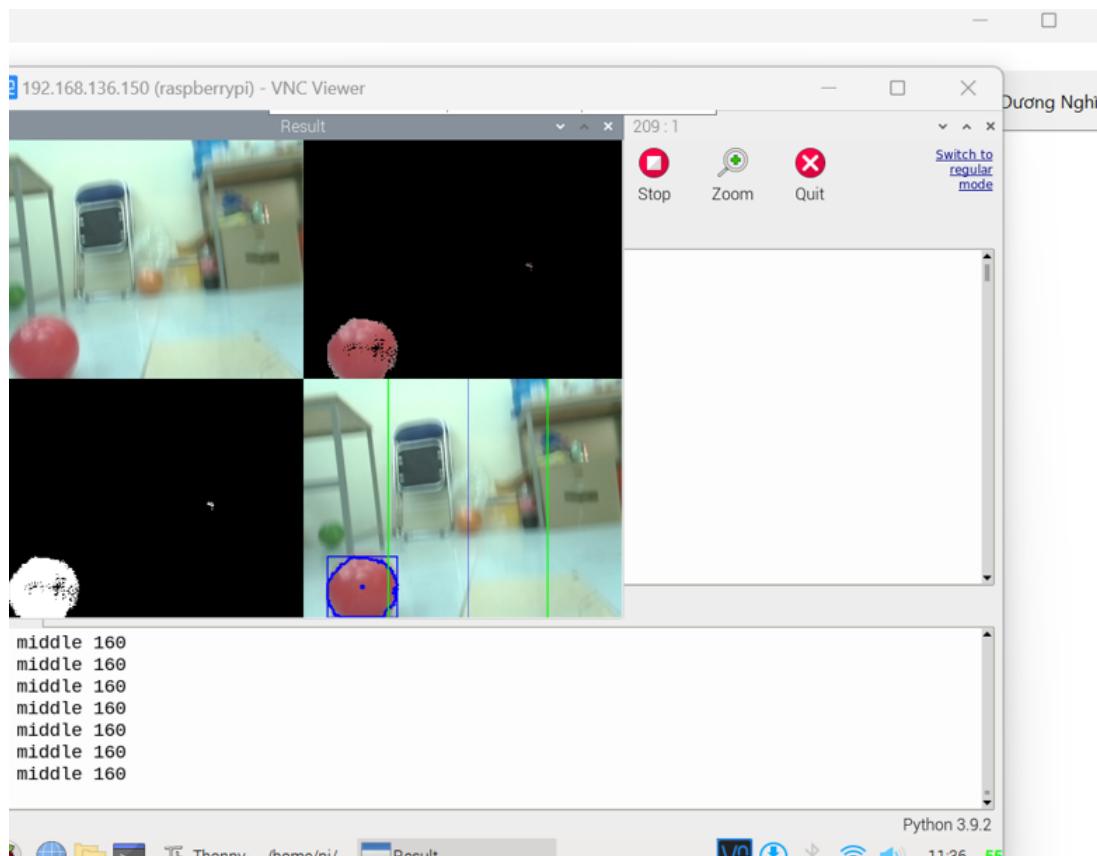
Nhận xét: Hướng tiếp cận này cho kết quả với độ chính xác khá thấp do độ sắc nét của camera không đủ để bắt được hình

### 7.2 Hướng tiếp cận nhận diện vật thể bằng contour detection

```
1 while True:
2     success, img = cap.read()
3     imgColor, mask = myColorFinder.update(img, hsvVals)
4     imgContour, contours = cvzone.findContours(img, mask, minArea= 1000)
5     if contours:
6         data = contours[0]['center'][0], \
7               contours[0]['center'][1], \
8               int(contours[0]['area'])
9         print(data)
```

```
11     cv2.line(imgContour, (middle_x,540), (middle_x,0) , (255,0,0), thickness = 1)
12     cv2.line(imgContour, (left_bound, 540), (left_bound, 0), (0, 255, 0), thickness=2)
13     cv2.line(imgContour, (right_bound, 540), (right_bound, 0), (0, 255, 0), thickness=2)
14     imgStack = cvzone.stackImages([img, imgColor, mask, imgContour], 2, 0.5)
15     cv2.imshow("Result", imgStack)
```

Nhận xét: Hướng tiếp cận này cho kết quả khá chính xác tuy nhiên không thể nhận diện vật thể khác màu, nếu muốn nhận diện vật thể màu khác cần đổi tham số đầu vào. Một điều nữa là phương pháp này rất nhạy cảm với nhiễu



Hình 27: Ball detection

### 7.3 Hướng tiếp cận kết hợp cả 2 phương pháp sử dụng voting

Nhóm đang có định hướng đi theo phương pháp này trong tương lai, kết hợp cả 2 phương trên bù trừ lẫn nhau bằng cách vote theo tỉ lệ nhất định

### 7.4 Điều khiển hướng so với vật thể

Bước 1: Để biết được vị trí của vật thể nhóm đã đóng khung contour được phát hiện thành khung hình chữ nhật rồi xác định tâm của hình chữ nhật bao ngoài

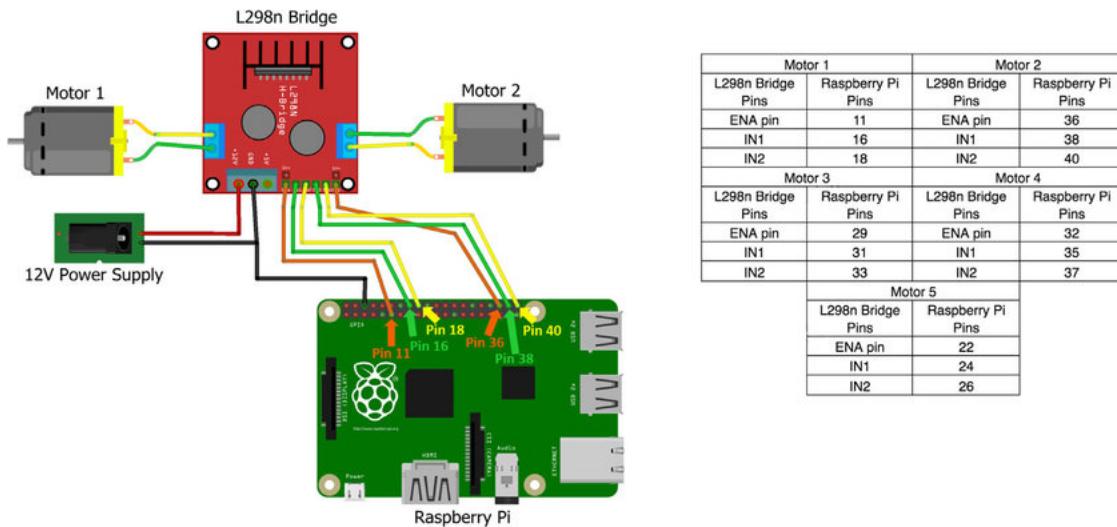
Bước 2: Xác định 2 boundary ở 2 bên (có thể thấy trong hình cuối của hình 26), nhóm giả định rằng nếu tâm hình chữ nhật rơi ra ngoài khung xanh lá thì sẽ điều khiển xe quay sao cho tâm trở lại vị trí đúng.

Bước 3: Nếu trường hợp area của hình chữ nhật bao ngoài nhỏ hơn 1 threshold được định nghĩa sẵn thì xe sẽ tiến lên phía trước và ngược lại

```
1  diff = abs(middle_x - data[0])
2      turn_speed = diff*scale
3      if right_bound < data[0]:
4          print('turn left')
5      elif left_bound > data[0]:
6          print('turn right')
7      else:
8          if data[2] < area_threshold:
9              print('forward')
10         if data[2] > area_threshold + 5000:
11             print('backward')
12         elif area_threshold <= data[2] <= area_threshold + 5000:
13             print("stay")
```

## 8 Phase 7: Hoàn thiện sản phẩm

### 8.1 Sơ đồ kết nối dây



Hình 28: Sơ đồ kết nối

### 8.2 Thực hiện lại giải thuật trên raspberry

```
1 import cvzone
2 import cv2
3 from cvzone.ColorModule import ColorFinder
4 import RPi.GPIO as GPIO
5 import time
6 from picamera.array import PiRGBArray
7 from picamera import PiCamera
8 from threading import Thread
9
10
11 # ===== motor config =====
12 in1 = 22
13 in2 = 24
14 in3 = 17
15 in4 = 27
16 enA = 6
17 enB = 26
18 servoPin = 13
19
20 init_default_speed = 25
21
22 GPIO.setmode(GPIO.BCM)
23 GPIO.setwarnings(False)
24
25 GPIO.setup(servoPin, GPIO.OUT)
```

```
26 GPIO.setup(in1,GPIO.OUT, initial = GPIO.LOW)
27 GPIO.setup(in2,GPIO.OUT, initial = GPIO.LOW)
28 GPIO.setup(in3,GPIO.OUT, initial = GPIO.LOW)
29 GPIO.setup(in4,GPIO.OUT, initial = GPIO.LOW)
30
31 GPIO.setup(enA,GPIO.OUT)
32 GPIO.setup(enB,GPIO.OUT)
33
34 pA = GPIO.PWM(enA,10)
35 pA.start(0)
36 pB = GPIO.PWM(enB,10)
37 pB.start(0)
38 servo = GPIO.PWM(servoPin, 50)
39 servo.start(0)
40
41
42
43 # ===== end motor config ======
44
45 # ===== direction config ======
46 def stop():
47     pA.ChangeDutyCycle(0)
48     pB.ChangeDutyCycle(0)
49     GPIO.output(in1,GPIO.LOW)
50     GPIO.output(in2,GPIO.LOW)
51
52 def backward(speed):
53     pA.ChangeDutyCycle(speed)
54     pB.ChangeDutyCycle(speed)
55     GPIO.output(in1,GPIO.HIGH)
56     GPIO.output(in2,GPIO.LOW)
57     GPIO.output(in3,GPIO.LOW)
58     GPIO.output(in4,GPIO.HIGH)
59
60 def turn_right(speed):
61     pA.ChangeDutyCycle(speed)
62     pB.ChangeDutyCycle(speed)
63     GPIO.output(in1,GPIO.LOW)
64     GPIO.output(in2,GPIO.HIGH)
65     GPIO.output(in3,GPIO.LOW)
66     GPIO.output(in4,GPIO.HIGH)
67
68
69 def forward(speed):
70     pA.ChangeDutyCycle(speed)
71     pB.ChangeDutyCycle(speed)
72     GPIO.output(in1,GPIO.LOW)
73     GPIO.output(in2,GPIO.HIGH)
74     GPIO.output(in3,GPIO.HIGH)
```

```
75     GPIO.output(in4,GPIO.LOW)
76
77 def turn_left(speed):
78     pA.ChangeDutyCycle(speed)
79     pB.ChangeDutyCycle(speed)
80     GPIO.output(in1,GPIO.HIGH)
81     GPIO.output(in2,GPIO.LOW)
82     GPIO.output(in3,GPIO.HIGH)
83     GPIO.output(in4,GPIO.LOW)
84
85
86 def moveServo():
87     duty = 2
88     while duty <= 12:
89         servo.ChangeDutyCycle(duty)
90         time.sleep(0.3)
91         servo.ChangeDutyCycle(0)
92         time.sleep(0.7)
93         duty = duty + 1
94
95
96
97
98 # ===== end direction config =====
99
100 # camera config =====
101 camera = PiCamera()
102 camera.resolution = (640, 480)
103 camera framerate = 32
104 rawCapture = PiRGBArray(camera, size=(640, 480))
105 camera brightness = 60
106 # end camera config =====
107
108 # ===== my vars =====
109 middle_x = 330
110 scale = 0.3
111 middle_diff = 40
112 left_bound = middle_x - middle_diff
113 right_bound = middle_x + middle_diff
114 area_threshold = 20000
115
116
117 # =====
118 myColorFinder = ColorFinder(False) # change this to True to get the color
119     you want
120 hsvVals = {'hmin': 163, 'smin': 0, 'vmin': 0, 'hmax': 179, 'smax': 255,
121     'vmax': 255}
122 for frame in camera.capture_continuous(rawCapture, format="bgr",
123     use_video_port=True):
```

```
121     try:
122         img = frame.array
123         imgColor, mask = myColorFinder.update(img, hsvVals)
124         imgContour, contours = cvzone.findContours(img, mask, minArea= 1000)
125         if contours:
126             data = contours[0]['center'][0], \
127                   contours[0]['center'][1], \
128                   int(contours[0]['area'])
129 #             print(data)
130 #             print(right_bound, left_bound)
131
132             diff = abs(middle_x - data[0])
133             turn_speed = diff*scale
134             if right_bound < data[0]:
135                 turn_right(30)
136             elif left_bound > data[0]:
137                 turn_left(30)
138             else:
139                 if data[2] < area_threshold:
140                     forward(init_default_speed)
141                 if data[2] > area_threshold + 5000:
142                     backward(init_default_speed)
143                 elif area_threshold <= data[2] <= area_threshold + 5000:
144                     stop()
145             else:
146                 stop()
147
148             cv2.line(imgContour, (middle_x,540), (middle_x,0) ,(255,0,0),
149             thickness = 1)
150             cv2.line(imgContour, (left_bound, 540), (left_bound, 0), (0, 255, 0)
151             , thickness=2)
152             cv2.line(imgContour, (right_bound, 540), (right_bound, 0), (0, 255,
153             0), thickness=2)
154             #imgStack = cvzone.stackImages([img, imgColor, mask, imgContour], 2,
155             0.5)
156             cv2.imshow("Result", imgContour)
157             #cv2.imshow("Frame", img)
158             key = cv2.waitKey(1) & 0xFF
159             # clear the stream in preparation for the next frame
160             rawCapture.truncate(0)
161             # if the `q` key was pressed, break from the loop
162             if key == ord("q"):
163                 break
164             if key == ord("m"):
165                 Thread(target=moveServo).start()
```

```
166     if key == ord("d"):
167         turn_right(init_default_speed)
168     if key == ord("z"):
169         backward(init_default_speed)
170 except:
171     stop()
```

### 8.3 Testing

## 9 Phase 8 (Bonus): Một số cải tiến so với lần demo trước

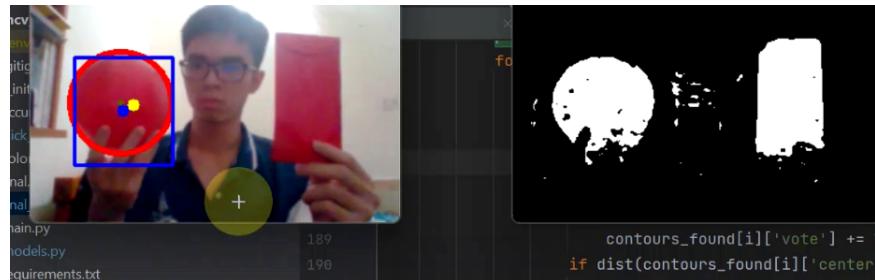
### 9.1 Click chọn màu

Cải thiện độ chính xác bằng việc click để chọn màu thay vì điều chỉnh thủ công bằng tay

```
1     if event == cv2.EVENT_LBUTTONUP:
2         color_bgr = frame[y, x]
3         color_rgb = tuple(reversed(color_bgr))
4         color_hsv = rgb2hsv(color_rgb[0], color_rgb[1], color_rgb[2])
5         colors.append(color_hsv)
```

### 9.2 Sử dụng voting

```
1             for i in range(3): # find most voted biggest cnt
2                 if i < len(contours_found):
3                     if circle_updated and dist(contours_found[i]['center'
4                         ][0], contours_found[i]['center'][1],
5                                         last_circle[0], last_circle
6                                         [1]) < 200:
7                         contours_found[i]['vote'] += 200
8                         if tracking_object and dist(contours_found[i]['center'
9                             ][0], contours_found[i]['center'][1],
10                                tracking_object[0],
11                                tracking_object[1]) < 5000:
12                                    contours_found[i]['vote'] += 75
13                                    if dist(contours_found[i]['center'][0], contours_found[i
14                                     ][0][1],
15                                     screen_center_point[0], screen_center_point[1])
16                                     < 2000:
17                                         contours_found[i]['vote'] += 25
18                                         contours_found[i]['vote'] += (2 - i)
```



Hình 29: Ball detection

### 9.3 Những thay đổi khác

- Cố định camera giảm việc khung hình bị xước trong quá trình chạy
- Thêm chế độ điều khiển bằng tay
- Thêm servo giúp camera quay 180 để tiện lợi trong khi dùng xe để trinh thám



## 10 Tổng kết, tự đánh giá

Qua Đồ án Thiết kế luận lý lần này, chúng em đã học được thêm rất nhiều kiến thức và kỹ năng mới. Đó là thông tin, cách sử dụng và vận hành Arduino, Motor Driver, Module Bluetooth cũng như lắp ráp động cơ cho xe. Rasberry Pi cũng là một công cụ rất bổ ích và thiết thực trong việc thực thi các tác vụ như một máy tính thu nhỏ.

Trong quá trình thực hiện đồ án sẽ không thể tránh khỏi có sai sót. Nhóm chúng em xin tiếp thu những ý kiến đóng góp từ thầy, cô để rút kinh nghiệm và cải thiện hơn trong những lần tiếp theo. Chúng em cũng xin chân thành cảm ơn thầy Trần Thanh Bình đã hỗ trợ cũng như giúp đỡ nhóm trong quá trình hoàn thiện bài báo cáo lần này.



## 11 File đính kèm

Source Code: Github: [https://github.com/duongnghia222/DATKLL\\_Object\\_Tracking\\_Robot/](https://github.com/duongnghia222/DATKLL_Object_Tracking_Robot/)  
Video Demo Updated: Youtube: <https://youtu.be/pZcHnqSAVPQ>



## Tài liệu

- [1]
- [2] <https://www.alldatasheet.com/datasheet-pdf/pdf/22440/STMICROELECTRONICS/L298N.html>
- [3] <https://blog.mecsuvn.vn/raspberry-pi-4-la-gi/>
- [4] <https://aicurious.io/posts/2019-10-24-hough-transform-phat-hien-duong-thang/>
- [5] <https://learnopencv.com/contour-detection-using-opencv-python-c>