

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



ĐỒ ÁN THIẾT KẾ LUẬN LÝ

Phân loại vật phẩm trên băng chuyền
bằng OpenCV thông qua máy tính nhúng Raspberrypi

GVHD: Trần Thanh Bình
SV thực hiện: Nguyễn Trường Thảo – 2114798
Nguyễn Trần Quang Vũ – 2115325
Lê Phan Quốc Vũ – 2115321

Tp. Hồ Chí Minh, Tháng 12/2023

Mục lục

1	Tổng quan	2
1.1	Giới thiệu đề tài	2
1.2	Giới hạn đề tài	2
2	Ý tưởng	2
3	Hiện thực	4
3.1	Phân cứng sử dụng	4
3.1.1	Raspberry pi 3b+	4
3.1.1.a	Định nghĩa	4
3.1.1.b	Thông số kỹ thuật	4
3.1.1.c	Một số ứng dụng của thiết bị	4
3.1.1.d	Hệ điều hành	5
3.1.2	Động cơ servo SG90	5
3.1.2.a	Định nghĩa	5
3.1.2.b	Thông số kỹ thuật	6
3.1.2.c	Ứng dụng	6
3.1.3	Hệ thống băng tải	6
3.1.3.a	Định nghĩa	6
3.1.3.b	Cấu tạo băng tải	7
3.1.3.c	Ứng dụng	7
3.1.4	Arduino nano	8
3.1.4.a	Định nghĩa	8
3.1.4.b	Thông số kỹ thuật	8
3.1.4.c	Ứng dụng	9
3.1.5	Màn hình led 2004	9
3.1.5.a	Định nghĩa	10
3.1.5.b	Thông số kỹ thuật	10
3.1.5.c	Ưu điểm của thiết bị	10
3.1.6	Raspberry pi Camera 5MP	11
3.1.6.a	Định nghĩa	11
3.1.6.b	Thông số kỹ thuật	11
3.1.6.c	Ưu điểm của thiết bị	11
3.2	Xử lý ảnh	13
3.2.1	Ý tưởng	13
3.2.2	Hiện thực	13
3.3	Xử lý động cơ	23
3.3.1	Ý tưởng	23
3.3.2	Hiện thực	24
3.3.2.a	Khởi tạo các động cơ servo	24
3.3.2.b	Sử dụng Software-timer để quản lí thời gian thực hiện cho các tác của Servo	24
3.3.2.c	Servo sẽ nhận tín hiệu từ Uart để tiến hành xử lí	25
3.3.2.d	Hệ thống sẽ thực hiện tự động	25
3.3.2.e	Mô phỏng cơ chế nhận và xử lí dữ liệu	26
4	Kết quả đạt được	26
5	Phân công công việc	27
5.1	Phân công	27
	Tài liệu	27

1 Tổng quan

1.1 Giới thiệu đề tài

Ngày nay, xã hội đang càng ngày càng phát triển. Nhu cầu về hàng hóa ngày càng cao, công nghiệp hóa, hiện đại hóa ngày càng được chú trọng ở mọi lĩnh vực. Trong đó, tự động hóa được xem là một trong các hướng đi giúp đẩy mạnh công nghiệp hóa, hiện đại hóa. Càng ngày càng có nhiều mô hình tự động hóa được ra đời, các máy móc dần dần được con người ứng dụng vào trong các nhà máy xí nghiệp, nhằm giúp gia tăng được tốc độ công việc cũng như đáp ứng nhu cầu của thị trường. Mô hình Băng chuyền không còn xa lạ đối với hầu hết các xí nghiệp cũng như doanh nghiệp sản xuất vật liệu xây dựng, vận chuyển hàng hóa, máy móc, ... Một trong những yếu tố giúp cho hệ thống đạt hiệu quả cao nhất nhằm đáp ứng nhu cầu thị trường là hệ thống băng chuyền phân loại sản phẩm, một phần quan trọng đảm bảo việc phân phối và đóng gói sản phẩm được thực hiện một cách chính xác và nhanh chóng. Đối mặt với sự đa dạng ngày càng tăng của sản phẩm và yêu cầu chính xác ngày càng cao, cần có những cải tiến vượt trội trong công nghệ để đảm bảo hiệu suất và đồng đều trong quy trình này.

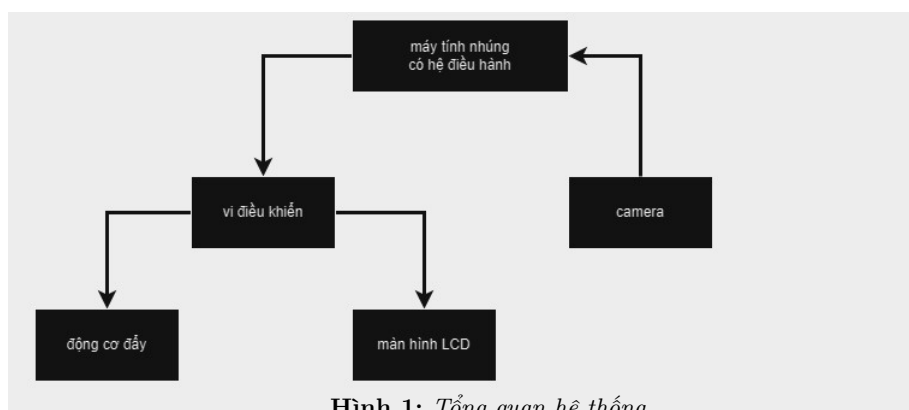
Vì vậy, trong đề tài này, nhóm chúng em sẽ hiện thực nghiên cứu và phát triển dự án băng chuyền phân loại sản phẩm tự động.

1.2 Giới hạn đề tài

Trong phạm vi một đề tài nghiên cứu, với những giới hạn về kiến thức, thời gian và kinh phí đề tài giới hạn bởi:

- Để ứng dụng kỹ thuật xử lý ảnh trên băng tải thì độ chính xác phân loại sản phẩm chưa quá cao. Còn có thể bị nhầm lẫn bởi các yếu tố khách quan như là độ nhiễu của hình ảnh, bóng của vật phẩm, ...
- Đề tài chỉ xây dựng với mô hình nhỏ và gọn không làm thành dây chuyền sản xuất, sử dụng máy tính nhúng, Arduino, băng tải, động cơ Servo và hiển thị số lượng sản phẩm lên màn hình LCD
- Các vật phẩm dùng để phân loại gồm có hình tròn, vuông, tam giác với kích thước vừa đủ đặt vào băng chuyền và có chiều cao từ 3 đến 5cm

2 Ý tưởng

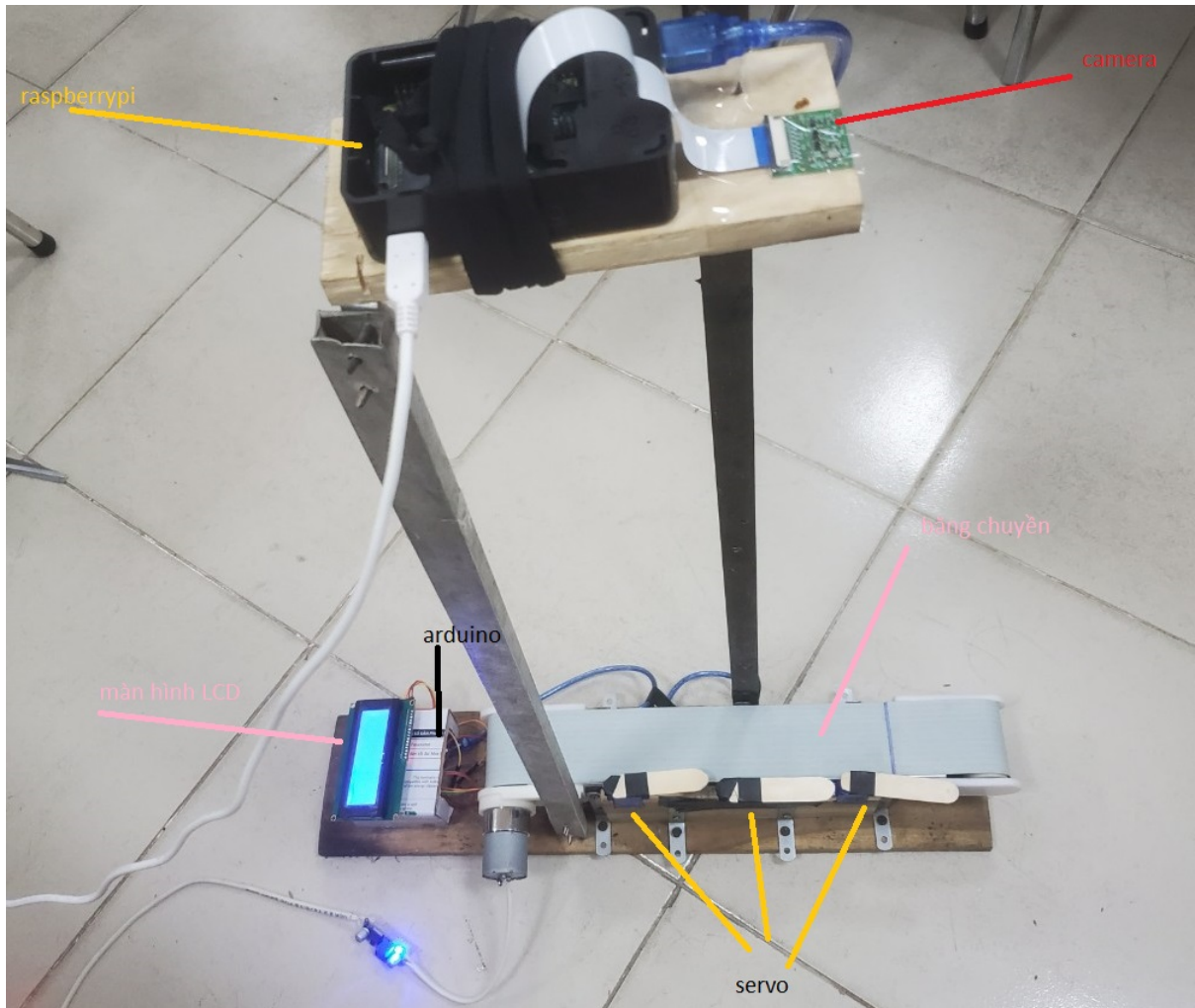


Hình 1: Tổng quan hệ thống

Hệ thống gồm 4 thành phần chính như sau:

- Camera: dùng để quay video cho toàn bộ băng chuyền
- Máy tính nhúng: dùng để xử lý ảnh từ camera (yêu cầu có hệ điều hành vì sẽ dùng OpenCV để xử lý ảnh) và gửi tín hiệu về cho vi điều khiển thông qua giao thức UART mỗi khi vật phẩm đi đến nơi cần phân loại
- Vi điều khiển: dùng để nhận tín hiệu từ máy tính nhúng, xử lý tín hiệu đó và tạo tín hiệu để điều khiển động cơ và màn hình LCD

- Động cơ đẩy: dùng để đẩy vật phẩm từ băng chuyền ra ngoài để phân loại, mỗi loại sản phẩm cần phân loại ứng với một động cơ
- Màn hình LCD: dùng để hiển thị số lượng vật phẩm ứng với mỗi loại vật phẩm



Hình 2: Mô hình ngoài thực tế

3 Hiện thực

3.1 Phần cứng sử dụng

Thiết bị	Số lượng
Raspberry pi 3b+	1
Động cơ servo SG90	3
Băng chuyền	1
Arduino nano ATmega328P	1
Màn hình lcd 2004	1
Raspberry pi Camera 5MP	1

3.1.1 Raspberry pi 3b+



Hình 3: Hình ảnh Raspberry

3.1.1.a Định nghĩa

- Máy tính nhúng là một thiết bị, một hệ thống được thiết kế để phục vụ cho một yêu cầu, một bài toán, ứng dụng, một chức năng nhất định nào đó và được ứng dụng nhiều trong lĩnh vực công nghiệp, tự động hóa điều khiển, quan trắc và truyền tin...
- Raspberry Pi là một máy tính nhúng chạy hệ điều hành Linux và được tích hợp đầy đủ các chức năng của một máy tính thông thường. Raspberry Pi được phát triển bởi Raspberry Pi Foundation - một tổ chức phi lợi nhuận. Mục tiêu chính của Raspberry pi là cung cấp một giải pháp giáo dục và giúp đỡ người học nhất là trong lĩnh vực khoa học và kỹ thuật máy tính.

3.1.1.b Thông số kỹ thuật

3.1.1.c Một số ứng dụng của thiết bị

- **Học lập trình và giáo dục** : Raspberry pi tạo môi trường thuận lợi để thực hành và phát triển kỹ năng lập trình.
- **các dự án DIY** : Kích thước nhỏ cùng giá thành hợp lý và kết nối được với nhiều phần cứng khác nhau khiến raspberry pi là thiết bị lý tưởng cho các dự án DIY.
- **Giải trí đa phương tiện** : Với nhiều chức năng được tích hợp như máy tính thông thường, raspberry pi có thể cung cấp dịch vụ giải trí đa phương tiện như xem phim, lướt web, ngoài ra còn có thể chơi game trên thiết bị này...

CPU	Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC @1.4GH
RAM	1GB LPDDR2 SDRAM
Cổng USB	4 x 2.0
Cổng GPIO	40 pin
Kết nối	wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet
Lưu trữ	Micro SD
Multimedia	full-size HDMI x 1 MIPI DSI Display x 1 MIPI CSI Camera
Nguồn điện sử dụng	5V / 2.5A DC cổng MicroUSB

Bảng 1: Thông số kỹ thuật

- **Dự án IOT** : Raspberry pi thường được dùng trong các dự án IOT để kết nối và thu thập dữ liệu từ các thiết bị khác. Đồng thời có thể tạo nên một vài hệ thống như hệ thống an ninh giám sát, máy chủ web và dịch vụ mạng, ...

3.1.1.d Hệ điều hành

Trong đồ án của chúng em, Raspberry pi được sử dụng sẽ được cài hệ điều hành là Raspberry pi OS (Legacy), hệ điều hành này tương thích với hầu hết các phiên bản của thiết bị Raspberry pi và hỗ trợ đầy đủ chức năng lập trình python cũng như kết nối UART, I2C cho đồ án này.

Lưu ý, khi ta sử dụng thư viện [picamera2](#) để sử dụng camera trong Python thì ta nên tắt tính năng Legacy Camera trong mục Interface Option (gõ `sudo raspi-config` để thực hiện)

Trong dự án này, nhóm em sử dụng Python phiên bản 3.9.2 và OpenCV phiên bản 4.5.0

3.1.2 Động cơ servo SG90



Hình 4: Hình ảnh Servor

3.1.2.a Định nghĩa

- Động cơ servo là một loại máy móc chuyên dùng để cung cấp cơ năng cho một thiết bị, dây chuyền hay cơ cấu trong quá trình sản xuất và chế tạo. Chúng cung cấp lực kéo cho các dây truyền hay cơ cấu khác hoạt động theo. Thiết bị này sử dụng từ trường để biến điện năng thành cơ năng
- **Phân loại** : Có 2 loại động cơ servo là AC (xoay chiều) và DC (1 chiều):

- + AC servo motor: loại động cơ này dùng nguyên tắc của dòng điện xoay chiều để điều khiển vị trí. Servo AC thường được sử dụng trong các ứng dụng công nghiệp như máy CNC, máy phay, máy tiện cơ khí, thủy lực
- + DC servo motor: là loại động cơ servo phổ biến nhất, sử dụng nguyên tắc phản hồi về vị trí để điều khiển động cơ đến vị trí mong muốn. Thường được dùng cho công suất nhỏ như máy bơm nước, máy nén khí,...
- Động cơ servo SG90 là loại động cơ được sử dụng phổ biến trong các mô hình điều khiển nhỏ và đơn giản như cánh tay robot. Động cơ có tốc độ phản ứng nhanh được tích hợp sẵn Driver điều khiển động cơ, dễ dàng điều khiển góc quay bằng phương pháp điều khiển độ rộng xung PWM.

3.1.2.b Thông số kỹ thuật

- Điện áp hoạt động : 4.8 - 5 VDC
- Tốc độ : 60 độ trong 0.12s (4.8 V)
- Lực kéo : 1.6 Kg.cm
- Kích thước : 21x12x22 mm
- Trọng lượng : 9g

Phương pháp điều khiển PWM

- Độ rộng xung 0.5ms - 2.5ms tương ứng 0 - 180 độ
- Tần số 50Hz, chu kỳ 20ms

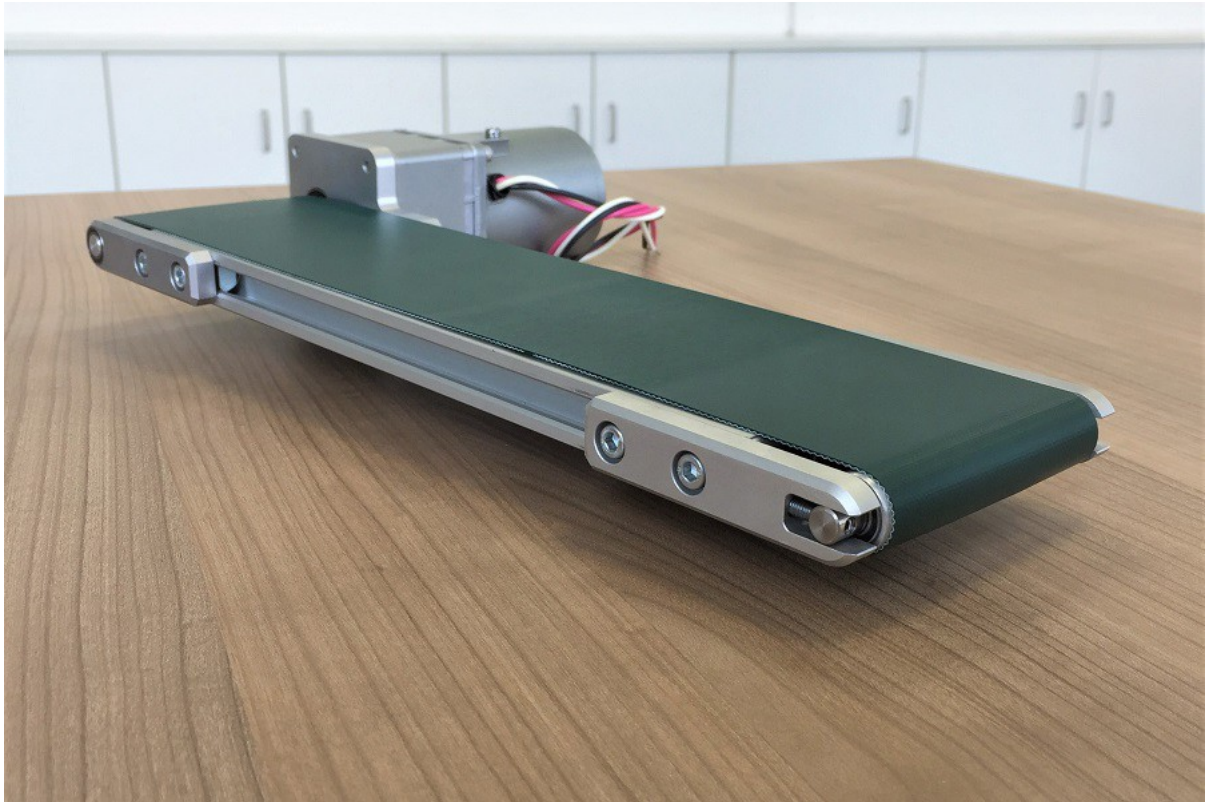
3.1.2.c Ứng dụng

- **Các thiết bị y tế** : Động cơ servo được sử dụng trong các thiết bị y tế như máy hút phổi hoặc máy thở để cung cấp độ chính xác và độ tin cậy cao.
- **Robotics** : Động cơ servo thường được sử dụng trong các robot để kiểm soát chính xác vị trí và góc quay của các khớp và cụm cảm biến.
- **Máy in 3D** : Động cơ servo có thể được sử dụng trong máy in 3D để di chuyển vị trí của đầu in chính xác, giúp tạo ra các đối tượng 3D với chi tiết và độ chính xác cao.
- **Mô hình và đồ chơi điều khiển từ xa** : Động cơ servo thường xuất hiện trong mô hình và đồ chơi điều khiển từ xa, giúp làm cho các phần chuyển động như cánh cửa, bánh xe, hoặc tay robot chính xác và linh hoạt.
- **Các thiết bị điện tử** : Động cơ servo được sử dụng trong các thiết bị như máy in, máy quét và máy chấm công để cung cấp độ chính xác và độ tin cậy cao.
- Ngoài ra động cơ servo còn được dùng rộng rãi trong các thiết bị tàu thủy như cần cẩu, mỏ neo,... hoặc các thiết bị vận chuyển như thang máy hoặc thang kéo,....

3.1.3 Hệ thống băng tải

3.1.3.a Định nghĩa

- Băng tải (băng chuyền) là một thiết bị xử lý vật liệu cơ khí di chuyển hàng hóa, vật tư từ nơi này sang nơi khác trong một đường dẫn xác định cho trước. Băng tải đặc biệt hữu ích trong các ứng dụng liên quan đến việc vận chuyển vật liệu nặng hoặc cồng kềnh. Hệ thống băng tải cho phép vận chuyển nhanh chóng và hiệu quả đối với nhiều loại vật liệu. Bên cạnh đó là sản xuất, băng chuyền giảm nguy cơ chấn thương lưng, đầu gối, vai và chấn thương chỉnh hình khác.
- Thay vì vận chuyển sản phẩm bằng công nhân vừa tốn thời gian, chi phí nhân công lại tạo ra môi trường làm việc lộn xộn thì băng chuyền tải có thể giải quyết điều đó. Nó giúp tiết kiệm sức lao động, số lượng nhân công, giảm thời gian và tăng năng suất.



Hình 5: Băng chuyền mini

3.1.3.b Cấu tạo băng tải

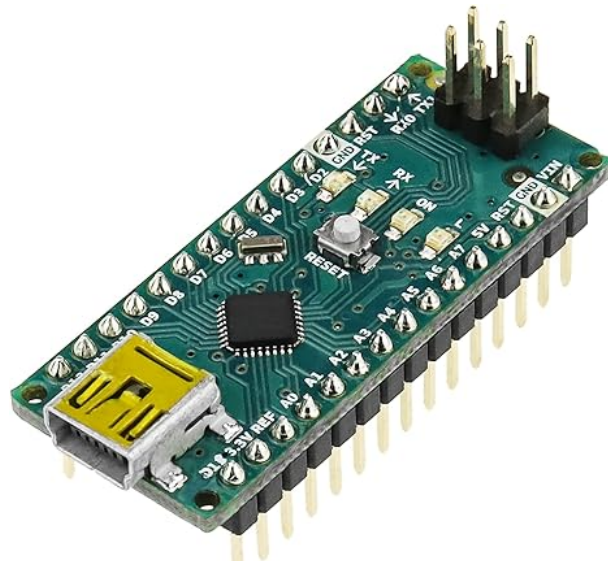
- Khung băng tải : thường được làm bằng nhôm định hình, thép sơn tĩnh điện hoặc inox
- Dây băng tải : thường là băng dây PVC dày 2mm hoặc 3mm hoặc dây băng PU dày 1.5mm
- Động cơ chuyển động : là động cơ giảm công suất 0.2KW, 0.4KW, 0.75KW, 1.5KW, 2.2KW
- Bộ điều khiển băng tải : PLC, biến tần, Speed controller, cảm biến, Rơ-le, Contactor,....
- Hệ thống bàn thao tác trên băng chuyền thường bằng gỗ, thép hoặc inox, trên bề mặt có dán thảm cao su chống tĩnh điện.
- Kích thước băng tải :
 - + Dài : 40 cm
 - + Rộng : 6 cm

3.1.3.c Ứng dụng

- Trong ngành sản xuất, lắp ráp linh kiện, thiết bị điện tử, lắp ráp ô tô, xe máy, xe đạp điện...
- Trong ngành sản xuất thực phẩm, y tế, dược phẩm, may mặc, giấy dép,...
- Dùng để vận chuyển hàng hóa, đóng gói sản phẩm,....

Lợi ích : băng chuyền được ứng dụng trong các ngành công nghiệp, chế biến thực phẩm, khai thác,... nhằm hỗ trợ tối đa cho doanh nghiệp trong quá trình vận chuyển hàng hóa nhanh chóng, an toàn và thuận tiện. Bên cạnh đó, hệ thống băng tải – băng chuyền có thể được lắp đặt bất cứ nơi nào, mọi địa hình, không những mang lại hiệu quả kinh tế cao nó còn giảm thiểu tai nạn trong lao động bảo đảm tính an toàn lao động cao.

3.1.4 Arduino nano



Hình 6: Hình ảnh Arduino nano

3.1.4.a Định nghĩa

Arduino là nền tảng mã nguồn mở giúp con người xây dựng các ứng dụng điện tử có khả năng liên kết, tương tác với nhau tốt hơn, hỗ trợ phát triển lập trình nhúng. Arduino có thể xem như một chiếc máy tính thu nhỏ giúp người dùng lập trình, thực hiện các dự án điện tử không cần tới công cụ chuyên biệt phục vụ cho quá trình nạp code. Arduino nano là một trong những phiên bản của thiết bị Arduino. Arduino được thiết kế nhằm đến sự nhỏ gọn của bo mạch, giảm kích thước và trọng lượng kéo theo giá thành rẻ, tuy nhiên Arduino nano vẫn giữ được các chức năng cốt lõi của một Arduino.

3.1.4.b Thông số kỹ thuật

Arduino nano	Thông số kỹ thuật
Số chân Analog IO	8
Cấu trúc	AVR
Tốc độ xung	16 MHz
Dòng tiêu thụ IO	40mA
Số chân Digital IO	22
Bộ nhớ EEPROM	1KB
Bộ nhớ Flash	32 KB với 2KB sử dụng bởi Bootloader
Điện áp ngõ vào	7 - 12 V
Vi điều khiển	ATmega328P
Điện áp hoạt động	5 V
Kích thước bo mạch	18 x 45 mm
Ngõ ra PWM	6
SRAM	2KB

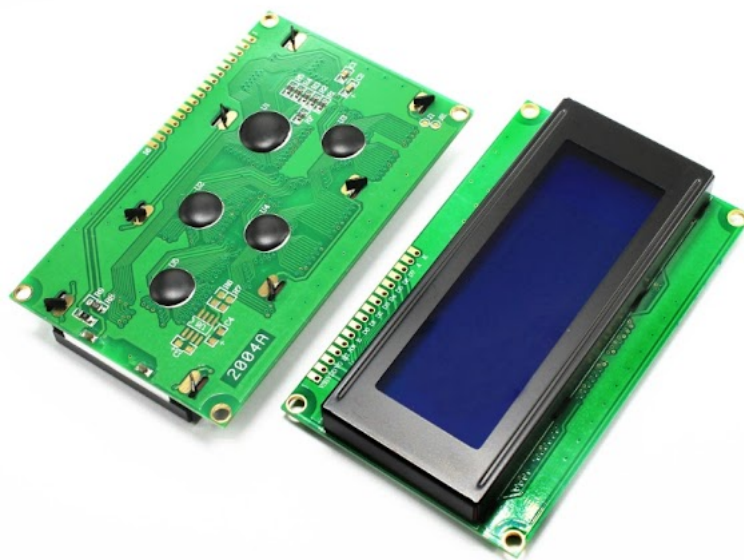
Bảng 2: Thông số kỹ thuật của Arduino nano

3.1.4.c Ứng dụng

Raspberry pi và Arduino nano là các thiết bị hỗ trợ lập trình nhúng. Tuy nhiên, Raspberry pi là một máy tính nhúng đầy đủ, được thiết kế để chạy hệ điều hành và ứng dụng như một máy tính thông thường. Trong khi Arduino sử dụng vi điều khiển để giao tiếp gần hơn với phần cứng và không có đầy đủ chức năng của máy tính. Mặc dù vậy, Arduino nano vẫn có những ứng dụng tương đối giống so với Raspberry pi :

- **Giáo dục và học lập trình** : thực hành để giải quyết bài toán về điện tử cơ bản, dự án mô phỏng để giảng dạy về cảm biến và điều khiển, tìm hiểu giao tiếp giữa cảm biến và vi điều khiển.
- **Robotics** : Arduino nano có khả năng đọc các thiết bị cảm biến, điều khiển động cơ,... nên nó thường được dùng để làm bộ xử lý trung tâm của rất nhiều loại robot.
- **Giải trí** : Arduino nano có thể được sử dụng để tương tác với Joystick, màn hình,... khi chơi các game như Tetris, phá gạch, Mario...
- **Đo lường và giám sát** : chính vì đặc tính giao tiếp được với các cảm biến, arduino nano thường được dùng trong các dự án kiểm tra và giám sát các điều kiện tự nhiên bên ngoài như thời tiết, nhiệt độ, độ ẩm, ...
- **Thử nghiệm và phát triển sản phẩm** : Arduino nano thường được sử dụng để thử nghiệm các ý tưởng nhúng và điều khiển trong quá trình phát triển sản phẩm. Được sử dụng nhiều trong tạo mẫu và testing sản phẩm trong công nghiệp.
- **IOT và tự động hóa** : các ví dụ có thể kể đến như điều khiển tín hiệu giao thông, làm biển quảng cáo LED, Drone - máy bay không người lái, máy in 3D trong công nghiệp, cánh tay Robot, xây dựng nhà thông minh,

3.1.5 Màn hình led 2004



Hình 7: LCD 20x4

3.1.5.a Định nghĩa

Màn hình tinh thể lỏng LCD (Liquid Crystal Display) được cấu tạo nên bởi các tế bào (các điểm ảnh) chứa tinh thể lỏng với khả năng thay đổi tính phân cực của ánh sáng và thay đổi cường độ ánh sáng truyền qua khi kết hợp với các loại kính lọc phân cực.

Nói một cách dễ hiểu hơn thì LCD chính là công nghệ dùng đèn nền để tạo ánh sáng chứ không tự phát sáng được.

Cấu tạo : Cấu tạo của màn hình LCD gồm 6 lớp xếp chồng lên nhau

- Kính lọc phân cực thẳng đứng có tác dụng lọc ánh sáng tự nhiên khi vào
- Lớp kính có điện cực ITO
- Lớp tinh thể lỏng
- Lớp kính có điện cực ITO chung
- Kính lọc phân cực nằm ngang
- Gương phản xạ, tác dụng phản xạ lại ánh sáng với người quan sát

Vốn dĩ màn hình LCD hiển thị màu sắc nhờ vào những điểm ảnh chứa tinh thể lỏng có thể thay đổi màu sắc và cường độ ánh sáng.

Màn hình LCD 2004 là một màn hình LCD với khả năng hiển thị 20 ký tự trên mỗi dòng và có tổng cộng 4 dòng, màn hình LCD không chỉ cung cấp một giao diện thân thiện với người dùng mà còn là công cụ hữu ích trong nhiều dự án nhúng. Với khả năng giao tiếp được với nhiều vi điều khiển khác nhau, LCD 2004 đã trở thành một phần quan trọng trong việc hiển thị thông tin từ các dự án DIY đơn giản cho đến các hệ thống điều khiển và giám sát phức tạp.

3.1.5.b Thông số kỹ thuật

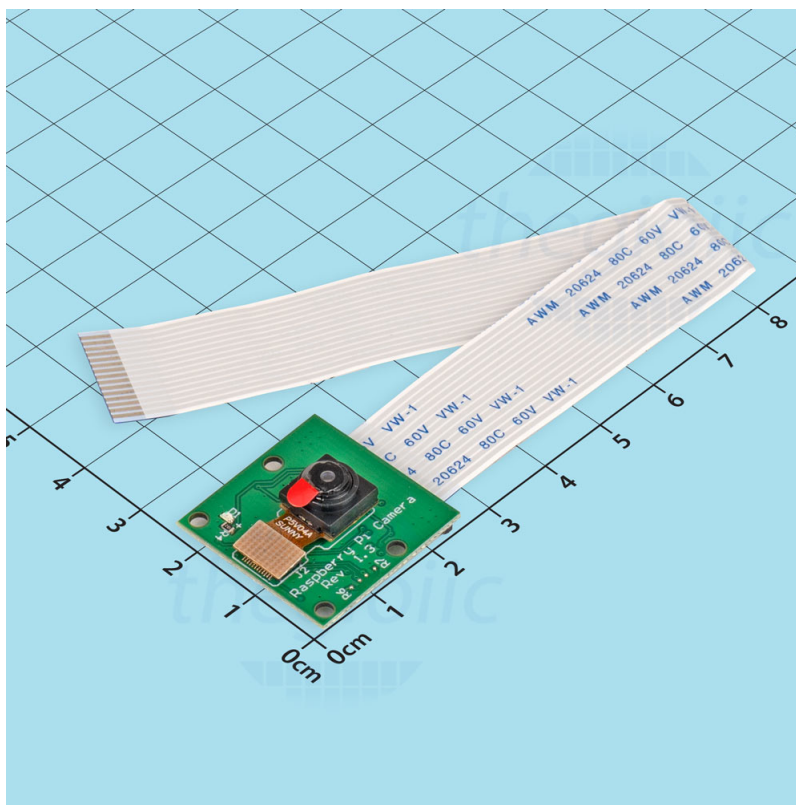
Màn hình LCD 2004 được trang bị driver HD44780, có khả năng hiển thị 4 dòng với mỗi dòng 20 ký tự, màn hình có độ bền cao.

- Điện áp hoạt động : 5V
- Kích thước : 98 x 60 x 13.5 mm
- Được trang bị thêm module I2C PCF8574 thuận tiện cho giao tiếp giữa màn hình và các thiết bị khác
- Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.

3.1.5.c Ưu điểm của thiết bị

- **Hiển thị nhiều dữ liệu** : lcd 2004 có khả năng hiển thị 4 dòng với mỗi dòng có tối đa 20 ký tự cung cấp một không gian hiển thị rộng lớn thông tin và dữ liệu. LCD 2004 cho nhiều không gian hơn nếu so với LCD 1602 chỉ có 2 dòng và mỗi dòng chỉ chứa được 16 ký tự.
- **Dễ đọc và hiển thị rõ ràng** : Màn hình LCD sử dụng công nghệ hiển thị cặp đối nghịch (STN) hoặc đối nghịch dương (TN), giúp hiển thị rõ ràng và dễ đọc dữ liệu dưới nhiều điều kiện ánh sáng.
- **Tương Thích Với Nhiều Ngôn Ngữ Lập Trình** : Màn hình LCD 2004 thường tương thích với nhiều loại ngôn ngữ lập trình từ Python cho tới C, C++, giúp tích hợp nhanh chóng vào các dự án sử dụng vi điều khiển như Arduino và Raspberry Pi hay STM32.
- **Tích hợp sẵn các chức năng mở rộng** : Màn hình LCD 2004 tích hợp một số chức năng mở rộng như điều khiển đèn nền, LED, điều khiển độ sáng tối, nút bấm,...
- **Giá thành hợp lý** : Giá thành của LCD 2004 tương đối rẻ, rất thích hợp cho những người mới bắt đầu hoặc học sinh sinh viên có thể tiếp cận để nghiên cứu và tìm hiểu, bên cạnh đó còn là giải pháp kinh tế cho nhiều dự án.

3.1.6 Raspberry pi Camera 5MP



Hình 8: Raspberry pi Camera 5MP

3.1.6.a Định nghĩa

Để tiếp cận gần hơn với các dự án liên quan đến thị giác máy tính, các camera là các thiết bị tiên quyết không thể thiếu. Trước đây, khi tạo tác máy tính nhúng giống như Raspberry pi, điều duy nhất chúng ta có thể làm là sử dụng một webcam cắm vào cổng USB. Mặc dù tốc độ xử lý nhanh nhưng các webcam lại có giá thành khá cao, nhất là các webcam có độ phân giải lớn. Tuy nhiên kể từ tháng 5/2013, Raspberry pi camera 5MP đã trở thành giải pháp hiệu quả do chính công ty mẹ của Raspberry pi công bố và phát hành.

3.1.6.b Thông số kỹ thuật

- 5 megapixel
- 2592×1944 ảnh tĩnh
- 1080p ở 30 khung hình / giây
- 720p ở 60 khung hình / giây
- 640x480p ở 60/90 khung hình / giây
- Giao tiếp CSI với cáp ribbon 150mm
- Mô-đun máy ảnh PCB + nặng 2.4g, cáp ribbon nặng 1g

3.1.6.c Ưu điểm của thiết bị

Camera Module 5MP của Raspberry Pi mang lại nhiều ưu điểm quan trọng, giúp nó trở thành một công cụ linh hoạt và mạnh mẽ cho nhiều ứng dụng khác nhau. Dưới đây là một số ưu điểm chính của Camera Module 5MP:

- Camera 5MP cung cấp chất lượng hình ảnh sắc nét và rõ ràng, làm cho nó phù hợp cho nhiều ứng dụng từ giám sát đến chụp ảnh và quay video chất lượng cao.
- Camera Module được thiết kế để dễ dàng kết nối với Raspberry Pi thông qua cổng CSI (Camera Serial Interface). Quá trình cài đặt và sử dụng nhanh chóng và thuận tiện.
- Với khả năng quay video Full HD (1080p), Camera Module là một lựa chọn tuyệt vời cho việc ghi lại video chất lượng cao, đặc biệt là trong các dự án giám sát và quay phim.
- Được sử dụng rộng rãi trong cộng đồng Raspberry Pi, nên có nhiều tài liệu, hướng dẫn và dự án mẫu giúp người sử dụng tìm hiểu và tận dụng đầy đủ khả năng của Camera Module.
- Kích thước nhỏ và thiết kế linh hoạt của Camera Module làm cho nó phù hợp với nhiều dự án có yêu cầu về không gian và trọng lượng.
- Thiết bị phù hợp cho cả người mới bắt đầu và những người muốn thực hiện các dự án DIY hoặc giáo dục trong lĩnh vực hình ảnh và video.

3.2 Xử lý ảnh

3.2.1 Ý tưởng

- Ta tạo một vùng ảnh bao bọc bằng chuyển để dễ xử lý
- Tạo các đường ranh để xác định vị trí cần đẩy vật. Gồm có 4 đường ranh:
 - Đường ranh dùng để xác định vật, nếu đi qua đường ranh này rồi thì vật thể được xem như đã được xác định hình dạng
 - Đường ranh dùng để đẩy vật hình tròn, nếu vật hình tròn nằm trong khoảng đường ranh này thì gửi tín hiệu xuống cho vi điều khiển để điều khiển động cơ dùng cho đẩy hình tròn
 - Đường ranh dùng để đẩy vật hình vuông, nếu vật hình vuông nằm trong khoảng đường ranh này thì gửi tín hiệu xuống cho vi điều khiển để điều khiển động cơ dùng cho đẩy hình vuông
 - Đường ranh dùng để đẩy vật hình tam giác, nếu vật hình tam giác nằm trong khoảng đường ranh này thì gửi tín hiệu xuống cho vi điều khiển để điều khiển động cơ dùng cho đẩy hình tam giác
- Xử lý chống nhiễu cho vật
- Sử dụng hàm `cv2.findContours` để xác định hình dáng vật thể
- Sử dụng thuật toán để keep tracking các object trước đó
- Kiểm tra vật có nằm trong các khoảng các đường ranh phía trên không

3.2.2 Hiện thực

1. Khởi tạo camera và COM Port để truyền tín hiệu UART Sau khi chúng ta kết gắn cổng USB vào raspberrypi để kết nối với arduino, gõ lệnh `ls/dev/tty*` để biết thông tin về cổng kết nối (có dạng `/dev/ttyUSB <number>`)

Lưu ý: Mỗi khi ta tháo kết nối giữa raspberrypi với arduino thì thông tin về cổng kết nối có thể sẽ thay đổi, nên ta cần phải kiểm tra trước khi chạy chương trình

```
1 #import library for pi camera
2 from picamera2 import Picamera2
3 #import library for UART Port
4 import serial
5 if __name__ == "__main__":
6     #init pi camera
7     cap = Picamera2()
8     cap.preview_configuration.main.size = (640,480)
9     cap.preview_configuration.main.format = "RGB888"
10    cap.preview_configuration.controls.FrameRate = 30
11    cap.preview_configuration.align()
12    cap.configure("preview")
13    cap.start()
14
15    #init COM Port for UART transfer
16    addr_com_port = "/dev/ttyUSB0"
17    #Note: variable "addr_com_port" may be change according raspberrypi
18    data_serial = serial.Serial(addr_com_port, 9600)
```

2. Tạo vùng ảnh bao bọc bằng chuyển

Lưu ý: Các thông số để xác định khu vực bằng chuyển được chỉnh tay và có thể camera sẽ bị lệch so với lần trước, nên ta cũng cần kiểm tra và tùy chỉnh các thông số mỗi khi ta di dời mô hình

```
1 import cv2 as cv
2 import numpy as np
3 if __name__ == "__main__":
```



```

4  #init pi camer
5  #init COM Port
6
7  while 1:
8      #init region image for conveyt
9      #Note: range of region may be change because camera is not fix in physical system
10     img = cap.capture_array()[50:450:, 250:320]

```

3. Tạo các đường ranh

Ta tạo các biến toàn cục để lưu tọa độ của các đường ranh đó và vẽ các đường ranh đó lên hình để dễ dàng quan sát

```

1  #init variable for save coordinate of line
2  point_detec = 280
3  point_tron = 230
4  point_vuong = 155
5  point_tam_giac = 60
6
7  if __name__ == "__main__":
8      #init pi camera
9      #init COM Port
10
11     while 1:
12         #init region image for conveyt
13         #draw line detect on image for easy to demonstrate
14         cv.line(img_contour, (0, point_detec), (width - 1, point_detec), (255,0, 0), 3)
15         cv.line(img_contour, (0, point_tron), (width - 1, point_tron), (255,0, 0), 3)
16         cv.line(img_contour, (0, point_vuong), (width - 1, point_vuong), (255,0, 0), 3)
17         cv.line(img_contour, (0, point_tam_giac), (width - 1, point_tam_giac), (255,0, 0), 3)

```



Hình 9: Các đường ranh và vùng nhận diện

4. Xử lý chống nhiễu cho vật

- Vì chúng ta đặt camera cố định trong quá trình sử dụng, nên ta dùng phương thức Background Subtraction để xác định khung nền của ảnh.
- Dùng threshold để loại bỏ đi bóng của vật

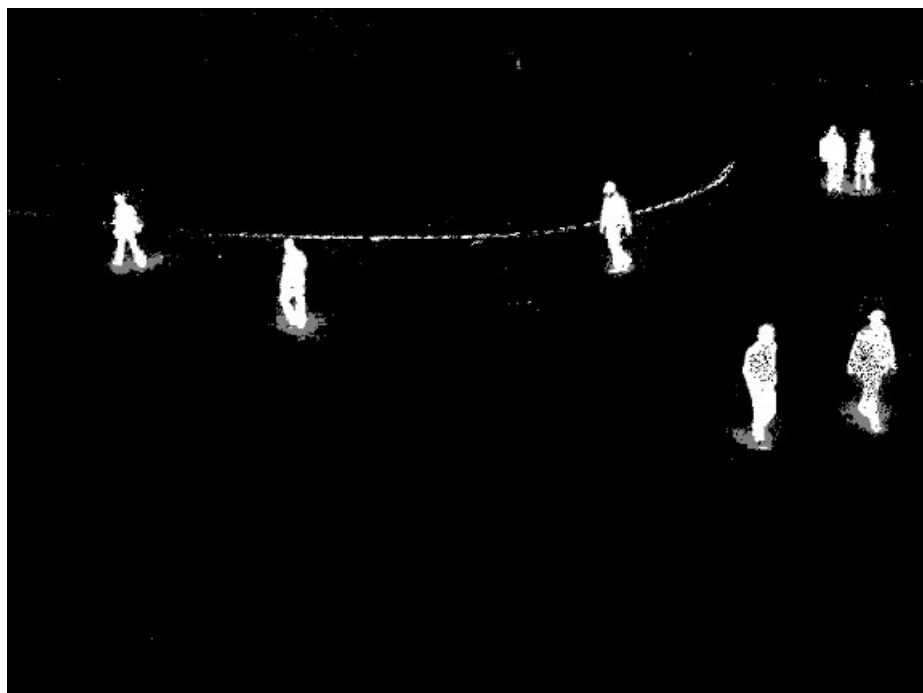
```

1 if __name__ == "__main__":
2     #create foreground mask
3     object_detector = cv.createBackgroundSubtractorKNN()
4     while 1:
5         #init region image for conveyt
6         img = cap.capture_array()[50:450:, 250:320]
7         #copy a new img with name "img_contour" for using after
8         img_contour = img.copy()
9         #filter noise for image
10        #apply background
11        detec = object_detector.apply(img)
12
13        #using threshold for filter shadow
14        _, detec = cv.threshold(detec,190,255,cv.THRESH_BINARY)
15        blur = cv.GaussianBlur(detec,(5,5),0)
16        _, detec = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
17
18        #draw line detect on display

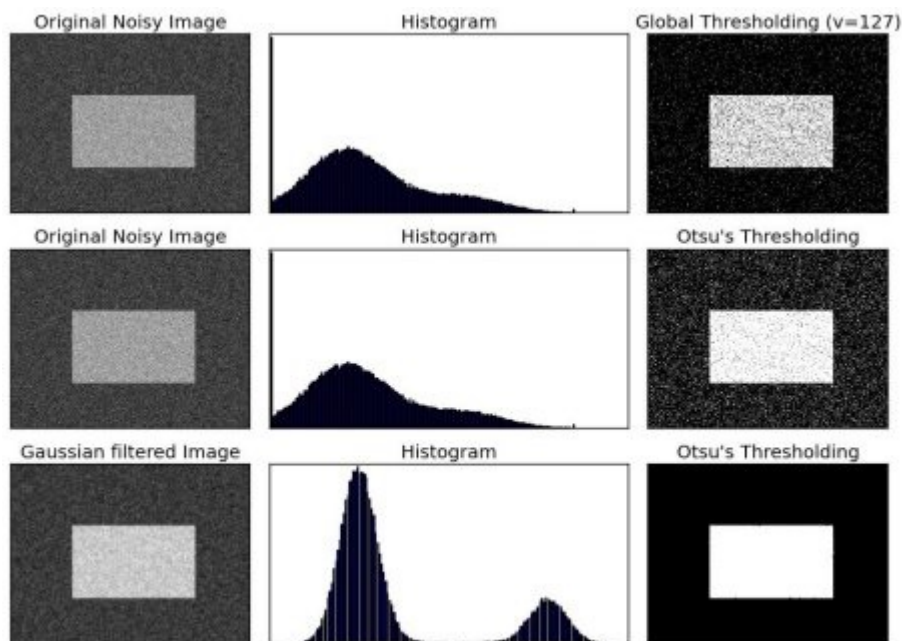
```



Hình 10: Hình ảnh ban đầu



Hình 11: Hình ảnh sau khi áp dụng `createBackgroundSubtractorKNN` (các vùng có màu xám được xác định là bóng của vật)



Hình 12: So sánh các cách xử lý ảnh bằng threshold

5. Xác định vật thể

Ý tưởng:

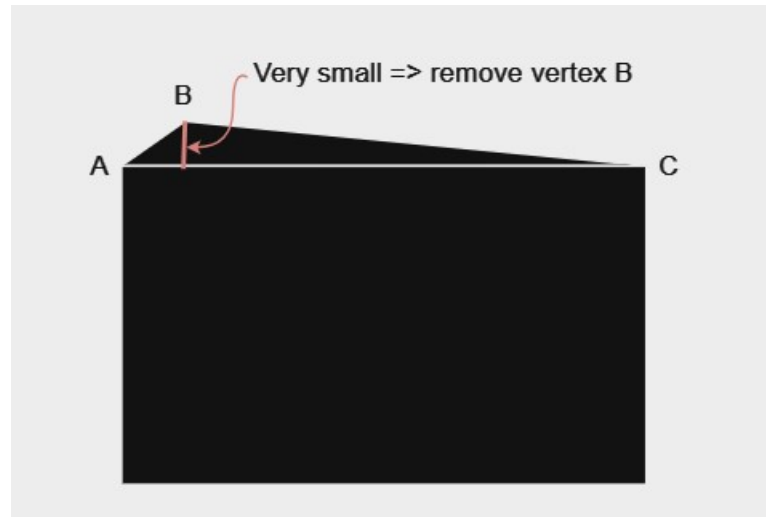
- Tạo một class item để lưu trữ vật khi phát hiện vật trên băng chuyền và tạo một mảng gồm các items đã được xác nhận
- Tạo 1 mảng gồm các item nhận được từ hàm cv2.findcontour
- Ứng với mỗi object trong mảng gồm các items đã được xác nhận, ta duyệt các phần tử trong mảng gồm các item nhận được từ hàm cv2.findcontour để tìm ra item gần với object nhất (với một ngưỡng khoảng cách xác định, nếu lớn hơn thì coi như object đó mất khỏi băng chuyền) rồi cập nhật object bằng các thông số của item gần nhất đó và đồng thời xóa item đó ra khỏi mảng
- Với những item nhận được từ hàm cv2.findcontour còn lại và nằm trong vùng nhận diện thì ta tạo mới object mới cho chúng và thêm vào mảng gồm các items đã được xác nhận
- Đối với những object không được cập nhật, ta có một biến dùng để đếm số lần biến đó không được cập nhật, nếu số lần đó vượt mức thì ta xóa biến đó mảng gồm các items đã được xác nhận

Hiện thực:

- Khai báo class item và mảng chứa các items đã được xác nhận:

```
1 class item:
2     id: int
3     shape: int
4     center: list #coordinate of item (x, y)
5
6     count_tam_giac = 0 #use to detect shape of item
7     count_vuong = 0 #use to detect shape of item
8     count_tron = 0 #use to detect shape of item
9     count_unknow = 0 #use to detect shape of item
10
11     count_ignore = 10 #decrease by one every item is not updated
12
13     is_update = 0 #use to know object is updatep from item which are determined in
14                     cv2.findcontour
15     is_detected = 0 #use to know object not in detect region
16
17     is_push = 0
18     def __init__(seft, id, center, shape):
19         seft.is_push = 0
20         seft.shape = shape
21         seft.id = id
22
23         seft.center = center
24 items: list[item] = []
```

- Xác định các vật thể từ hàm cv2.findcontour
 - Ta dùng hàm cv2.findcontour để xác định các vật thể xuất hiện trong khung hình
 - Tính diện tích của các vật thể đó, nếu vật thể đó quá nhỏ hoặc quá to thì ta bỏ qua để giảm nhiễu hình ảnh
 - Dùng hàm cv2.drawContours để bao bọc vật thể đó lại
 - Dùng hàm cv2.approxPolyDP để tìm các đỉnh vật thể đó và vẽ một hình chữ nhật bao bọc vật thể đó.
 - Giảm bớt các đỉnh của vật thể đó nếu khoảng cách từ một đỉnh đến 2 đỉnh liền kề nó quá nhỏ (< 10)



Hình 13: Xóa một đỉnh nếu khoảng cách đến đường thẳng nối 2 đỉnh kề quá nhỏ

- Tạo mới một item gồm vật thể đồ và gán nó vào danh sách các item được xác xác định trong hàm cv2.findcontour

```

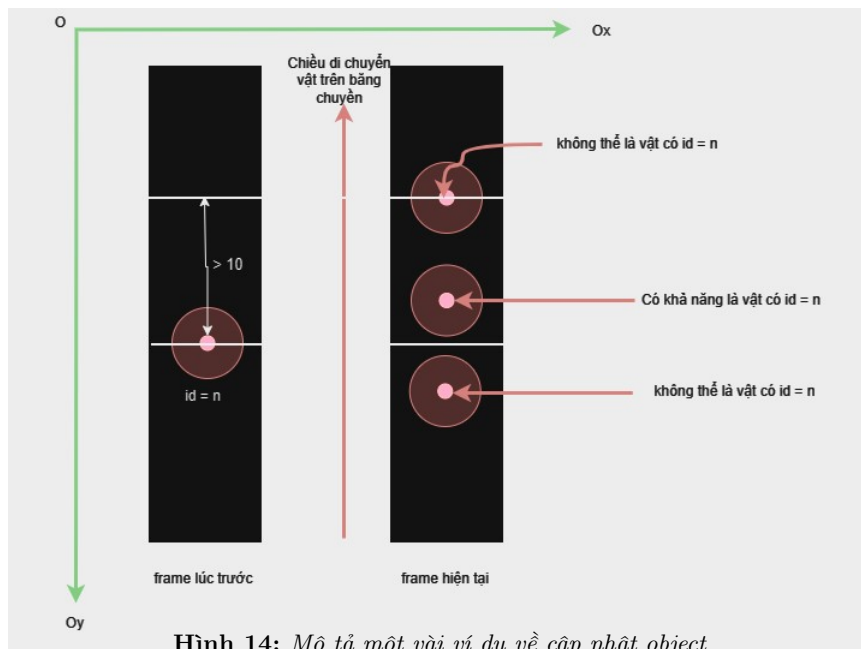
1 def getContours(img, img_contour):
2     contours, _ = cv.findContours(img, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
3
4     #list for object is determined from findcontour
5     list_contours_object = []
6
7     #detect item from cv2.findcontour
8     for cnt in contours:
9         area = cv.contourArea(cnt)
10        #check area of object for prevent noise
11        if ((area > 200) & (area < 5000)):
12            #draw shape for around object
13            cv.drawContours(img_contour, cnt, -1, (255, 0, 0), 2)
14
15            #find number of vertex in this object
16            peri = cv.arcLength(cnt, True)
17            approx = cv.approxPolyDP(cnt, 0.02*peri, True)
18
19            #draw rectangle for around object
20            x, y, w, h = cv.boundingRect(approx)
21            cv.rectangle(img_contour, (x, y), (x+w, y+h), (0,0,255), 2)
22            #draw point in center of this rectangle
23            center = [x+int(w/2), y+int(h/2)]
24            cv.circle(img_contour, tuple(center), 1, (0,0,255), 2)
25
26
27            shape: int = 0
28            if len(approx) >= 8: shape = 5 #This is circle
29            else:
30                arr = convert_approx_to_list(approx)
31                #filter for reduce vertex noise
32                #remove vertex which has distance from it to two adjacent vertex very small
33                shape = filter_for_remove_vectecx(arr)
34
35            new_object = item(id, center, shape)
36            list_contours_object.append(new_object)

```

- Cập nhật các object có trong danh sách gồm các object đã được xác nhận trước dựa trên danh sách

các item được xác định trong hàm `cv2.findcontour`

- Ứng với mỗi object có trong danh sách gồm các object đã được xác nhận trước ta duyệt qua danh sách các item được xác định trong hàm `cv2.findcontour`
- Nếu item mới có tọa độ điểm y lớn hơn tọa độ điểm y của object thì ta bỏ qua. Vì băng chuyển của chúng ta có chiều quay dây vật theo hướng ngược với trục Oy, nên nếu item mới có tọa độ điểm y lớn hơn thì chứng tỏ item đó không phải của object mà ta muốn update
- Xác định item được xác định trong hàm `cv2.findcontour` có khoảng cách đến object bé nhất. Nếu khoảng cách đó không quá lớn thì ta cập nhật object bằng các giá trị của item đó và xác nhận vật thể của object đó thông qua tham số shape của item vừa chọn đó
- Xóa item đó ra khỏi danh sách các item được xác định trong hàm `cv2.findcontour`.



Hình 14: Mô tả một vài ví dụ về cập nhật object

```

1 def getContours(img, img_contour):
2     contours, _ = cv.findContours(img, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
3
4     #list for object is determined from findcontour
5     list_contours_object = []
6
7     #detect item from cv2.findcontour
8     for cnt in contours:
9         #code in above
10        None
11
12    #update object in list "items" (list of object was detected before)
13    for object in items:
14        min_distance:int = 9999
15        index_min_distance: int = 9999
16
17        for i in range(len(list_contours_object)):
18
19            distance = object.center[1] - list_contours_object[i].center[1]
20            if distance < 0:
21                continue
22            if min_distance >= distance:
23                index_min_distance = i
24                min_distance = distance

```



```

25
26     if min_distance < 20:
27         min_object = list_contours_object[index_min_distance]
28
29         if object.is_detected == 0:
30             object.detect_object_many_times(min_object.shape)
31
32         object.center = min_object.center
33         object.is_update = 1
34         cv.putText(img_contour, f"{object.shape}, {object.id}",
35                    tuple(object.center), cv.FONT_HERSHEY_SIMPLEX, 0.3, (0,0,0), 1)
36         list_contours_object.remove(min_object)

```

Hàm `detect_object_many_times(seft, shape)` dùng để xác định hình dạng của item đó là tròn, vuông hay tam giác thông qua biến *shape* và các trạng thái trước đó của object

- Nếu item đó nằm ngoài vùng detect thì ta kết thúc hàm
- Nếu item có 3 lần liên tục tham số *shape* bằng nhau
 - * Nếu item chưa được nhận diện trước đó (*seft.shape == 0*) thì ta mới kết luận hình dạng của item đó
 - * Nếu item đã được nhận diện trước đó và có hình dạng khác với tham số *shape* thì ta cộng dồn biến *seft.count_unknow* cho 1
 - Nếu giá trị của biến *seft.count_unknow* ≥ 2 thì ta gán biến *seft.shape = 0*

```

1 class item:
2     def detect_object_many_times(seft, shape):
3         if seft.is_detected == 1:
4             return
5         if shape == 3:#TAM_GIAC
6             seft.count_tam_giac += 1
7             seft.count_vuong = 0
8             seft.count_tron = 0
9             if seft.count_tam_giac >= 3:
10                 if seft.shape == 0:#UNKNOWN
11                     seft.shape = 3 #TAM GIAC
12                     count_unknow = 0
13                 elif ((seft.shape == 4) | (seft.shape == 5)):
14                     seft.count_unknow += 1
15                     if seft.count_unknow >= 2:
16                         seft.shape = 0 #UNKNOWN
17                         seft.count_vuong = 0
18                         seft.count_tam_giac = 0
19                         seft.count_tron = 0
20             elif shape == 4:#VUONG
21                 #Similar to shape == 3
22                 None
23             elif shape == 5:#TRON
24                 #Similar to shape == 3
25                 None

```

- Thêm các item còn lại trong danh sách các item nhận được từ hàm `cv2.findcontour` vào danh sách các items đã được xác nhận nếu item đó nằm trong vùng detect (các item mới được đặt trên băng chuyền)

```

1 def getContours(img, img_contour):
2     contours, _ = cv.findContours(img, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
3
4     #list for object is determined from findcontour

```

```
5 list_contours_object = []
6
7 #detect item from cv2.findcontour
8 for cnt in contours:
9     #code in above
10    None
11
12 #update object in list "items" (list of object was detected before)
13 for object in items:
14     #code in above
15    None
16
17 #add new items from "list_contours_object" (items was detected in cv2.findcontour)
18 for new_object in list_contours_object:
19     #out of detect region
20     if new_object.center[1] < point_detec:
21         continue
22     new_object.is_update = 1
23     id += 1
24     new_object.shape = 0
25     items.append(new_object)
26
27 cv.putText(img_contour, f"{new_object.shape}, {new_object.id}",
28            tuple(new_object.center), cv.FONT_HERSHEY_SIMPLEX, 0.3, (0,0,0), 1)
```

- Xóa các object trong danh sách các items đã được xác nhận nếu nó không được update nhiều lần

```
1 def getContours(img, img_contour):
2     contours, _ = cv.findContours(img, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
3
4     #list for object is determined from findcontour
5     list_contours_object = []
6
7     #detect item from cv2.findcontour
8     for cnt in contours:
9         #code in above
10        None
11
12 #update object in list "items" (list of object was detected before)
13 for object in items:
14     #code in above
15    None
16
17 #add new items from "list_contours_object" (items was detected in cv2.findcontour)
18 for new_object in list_contours_object:
19     #code in above
20    None
21
22 #remove object which was not updated
23 items_remove = []
24 for i in range(len(items)):
25     object = items[i]
26     if object.is_update == 0:
27         items[i].count_ignore -= 1
28         if items[i].count_ignore <= 0:
29             items_remove.append(object)
30     else:
31         items[i].count_ignore = 10
32         items[i].is_update = 0
33 for i in items_remove:
```

```
34 items.remove(i)
```

- Tổng quát

```
1 def getContours(img, img_contour):
2     contours, _ = cv.findContours(img, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
3
4     #list for object is determined from findcontour
5     list_contours_object = []
6
7     #detect item from cv2.findcontour
8     for cnt in contours:
9         #code in above
10        None
11
12    #update object in list "items" (list of object was detected before)
13    for object in items:
14        #code in above
15        None
16
17    #add new items from "list_contours_object" (items was detected in cv2.findcontour)
18    for new_object in list_contours_object:
19        #code in above
20        None
21
22    #remove object which was not updated
23    items_remove = []
24    for i in range(len(items)):
25        #code in above
26        None
27    for i in items_remove:
28        items.remove(i)
29 if __name__ == "__main__":
30     #create foreground mask
31
32     while 1:
33         #init region image for conveyt
34
35         #filter noise for image
36
37         #draw line detect on display
38
39         #detect object using cv2.findcontour
40         getContours(detec, img_contour)
```

6. Kiểm tra vật thể đi qua các đường ranh

Duyệt qua từng phần tử của danh sách *items* (mảng gồm các items đã được xác nhận)

- Nếu vật phẩm đó đã được xác nhận (đi qua đường ranh detect)
 - Nếu vật phẩm đó đã được đẩy từ trước thì ta bỏ qua
 - Nếu vật phẩm là hình tam giác và nằm phạm vi đường ranh đẩy hình tam giác
 - * Bật cờ vật phẩm đã được đẩy lên
 - * Truyền tín hiệu xuống vi điều khiển để đẩy hình tam giác bằng giao tiếp UART
 - Tương tự cho hình vuông và tròn
- Ngược lại, nếu vật phẩm đó chưa được xác nhận: Kiểm tra vật phẩm có nằm trong phạm vi đường ranh xác nhận không, nếu có thì bật cờ vật phẩm đã được xác nhận lên

```
1 def checkThrough(data_serial):
2     for item in items:
3         if item.is_detected == 1:
4             if item.is_push == 1:
5                 continue
6
7             if item.shape == 3:
8                 if ((item.center[1] <= point_tam_giac + 10) & (item.center[1] >= point_tam_giac
9                     - 10)):
10                     item.is_push = 1
11                     #send signal through UART
12                     data_serial.write("3\r".encode())
13             elif item.shape == 4:
14                 #Similar above
15                 None
16             elif item.shape == 5:
17                 #Similar above
18                 None
19             else:#item hasn't detected
20                 if ((item.center[1] <= point_detec + 10) & (item.center[1] >= point_detec - 10)):
21                     item.is_detected = 1
```

Tổng quát:

```
1 if __name__ == "__main__":
2     #create foreground mask
3
4     while 1:
5         #init region image for conveyt
6
7         #filter noise for image
8
9         #draw line detect on display
10
11         #detect object using cv2.findcontour
12         getContours(detec, img_contour)
13
14         #check object through line detect
15         checkThrough(data_serial)
```

3.3 Xử lý động cơ

3.3.1 Ý tưởng

- Trước tiên chúng ta cần khởi tạo động cơ cho Servo bằng cách Setup trong ứng dụng Arduino IDE
- Tạo một software timer để theo dõi và kiểm soát thời gian của hệ thống nhằm đảm bảo một số tác vụ được diễn ra trong một khoảng thời gian nhất định.
- Dùng cơ chế UART kết nối giữa các con vi điều khiển với nhau. Được biết UART là một giao thức truyền thông tiêu chuẩn dựa trên chuẩn RS-232 (Recommended Standard 232). Nó thường được sử dụng để kết nối các vi điều khiển, cảm biến, mô-đun không dây và nhiều thiết bị khác trong các ứng dụng nhúng.
- Dữ liệu sẽ được truyền thông qua cơ chế UART để Servo xử lý các thao tác trong việc phân loại sản phẩm.

3.3.2 Hiện thực

Bảng thể hiện cách nối các chân của arduino với servo và màn hình LCD

Pin	Chức năng	
D2	Servo 1 (động cơ đẩy hình tròn)	OUTPUT
D3	Servo 2 (động cơ đẩy hình vuông)	OUTPUT
D4	Servo 3 (động cơ đẩy hình tam giác)	OUTPUT
D13	LED blink mỗi 1 giây	OUTPUT
A4	SDA của LCD	OUTPUT
A5	SCL của LCD	OUTPUT

3.3.2.a Khởi tạo các động cơ servo

```
1 #include <Servo.h>
2 #define SERVO_1 2
3 #define SERVO_2 3
4 #define SERVO_3 4
5
6 Servo dong_co_1;
7 Servo dong_co_2;
8 Servo dong_co_3;
9
10 void init_servo(){
11     dong_co_1.attach(SERVO_1);
12     dong_co_2.attach(SERVO_2);
13     dong_co_3.attach(SERVO_3);
14
15     dong_co_1.write(180);
16     dong_co_2.write(180);
17     dong_co_3.write(180);
18 }
```

3.3.2.b Sử dụng Software-timer để quản lý thời gian thực hiện cho các tác của Servo

- Setup thời gian hiện thực cho các Servor

```
1 int timer_servo_1_count;
2 uint8_t timer_servo_1_flag;
3 void set_timer_servo_1(int duration){
4     timer_servo_1_flag = 0;
5     timer_servo_1_count = duration;
6 }
7 uint8_t get_timer_servo_1(){
8     return timer_servo_1_flag;
9 }
```

- Setup thời gian thực hiện cho đèn led

```
1 int timer_led_blink_count;
2 uint8_t timer_led_blink_flag;
3 void set_timer_led_blink(int duration){
4     timer_led_blink_count = duration;
5     timer_led_blink_flag = 0;
6 }
7 uint8_t get_timer_led_blink(){
8     return timer_led_blink_flag;
9 }
```

```
9 }
```

- Hiện thực 1 timer-run để báo hiệu các tác vụ hay một sự kiện được diễn ra.

```
1 void timer_run(){
2     timer_servo_1_count --;
3     if(timer_servo_1_count <= 0){
4         timer_servo_1_flag = 1;
5     }
6     timer_led_blink_count --;
7     if(timer_led_blink_count <= 0){
8         timer_led_blink_flag = 1;
9     }
10 }
```

3.3.2.c Servo sẽ nhận tín hiệu từ Uart để tiến hành xử lý

- Nhận tín hiệu

```
1 void recv_uart(){
2     if(Serial.available()){
3         String du_lieu = "";
4         du_lieu = Serial.readStringUntil('\r');
5         if(du_lieu == "1"){
6             servo_state[0] = 1;
7         }else if(du_lieu == "2"){
8             servo_state[1] = 1;
9         }else if(du_lieu == "3"){
10            servo_state[2] = 1;
11        }
12    }
```

- Xử lý dữ liệu

```
1 void uart_handle_servo_1(){
2     switch(servo_state[0]){
3         case 1:// Day vat
4             dong_co_1.write(60);
5             //set timer for waiting servo run
6             set_timer_servo_1(30);
7             servo_state[0] = 0;
8             break;
9         case 0:
10            //return servo
11            if(get_timer_servo_1() == 1){
12                dong_co_1.write(0);
13            }
14            break;
15        default: break;
16    }
17 }
```

3.3.2.d Hệ thống sẽ thực hiện tự động

```
1 void loop() {
```

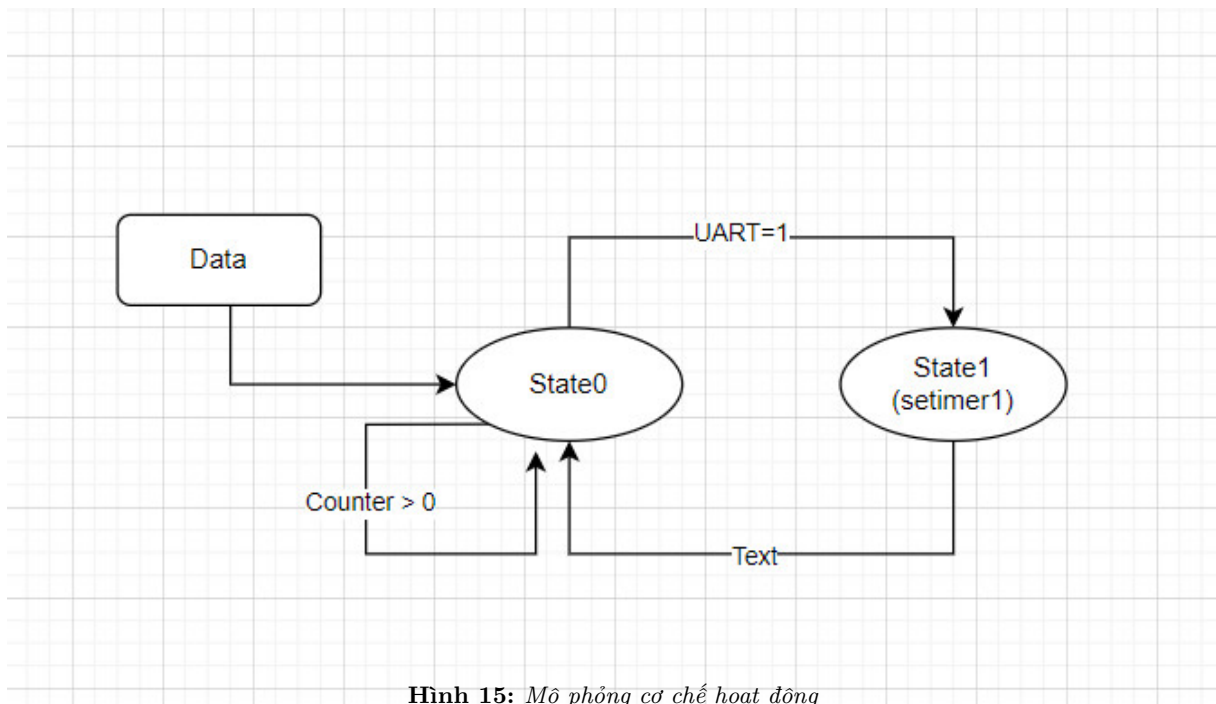


```

2
3  timer_run();
4  blink_led();
5
6  recv_uart();
7  uart_handle_servo_1();
8  uart_handle_servo_2();
9  uart_handle_servo_3();
10 delay(10);
11 }

```

3.3.2.e Mô phỏng cơ chế nhận và xử lý dữ liệu



Hình 15: Mô phỏng cơ chế hoạt động

4 Kết quả đạt được

Qua dự án này, chúng em học được những điều sau:

- Biết cách sử dụng raspberrypi: cài đặt hệ điều hành, sử dụng camera dành riêng cho raspberrypi, lập trình Python với thư viện OpenCV trên raspberrypi, ...
- Biết các kỹ thuật về xử lý ảnh cơ bản bằng cách sử dụng thư viện OpenCV
- Biết cách điều khiển động cơ servo bằng arduino
- Lập trình arduino

Video demo của dự án: <https://drive.google.com/file/d/1AOXUNfjJ-tzNRTrE20RW003B43xQyaEz/view?usp=sharing>

5 Phân công công việc

5.1 Phân công

Họ và tên	MSSV	Phân công công việc
Nguyễn Trần Quang Vũ	2115325	Điều khiển màn hình LCD, cài đặt raspberrypi
Nguyễn Trường Thản	2114798	Xử lý ảnh, làm mô hình
Lê Phan Quốc Vũ	2115321	Điều khiển động cơ servo

Tài liệu

- Doxygen. *Image Thresholding*. Truy cập từ: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
- Doxygen. *Contours : Getting Started*. Truy cập từ: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html
- Doxygen. *How to Use Background Subtraction Methods*. Truy cập từ: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html
- Q - engineering *Install OpenCV on Raspberry Pi*. August 10, 2023. Truy cập từ: <https://qengineering.eu/install-opencv-on-raspberry-pi.html>