BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM
KHOA ĐIỆN – ĐIỆN TỬ

THỰC TẬP XỬ LÝ ẢNH

ĐỀ TÀI

ỨNG DỤNG XỬ LÝ ẢNH TRONG NHẬN DẠNG MÀU SẮC DÙNG RASPBERRY

GVHD: ThS. NGUYỄN DUY THẢO

Sinh viên: Cao Thị Lan Anh
MSSV: 21161390
Đỗ Hoàng Thắng
MSSV: 21161191

Tp. Hồ Chí Minh – 5/2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA ĐIỆN – ĐIỆN TỬ

THỰC TẬP XỬ LÝ ẢNH

ĐỀ TÀI

ỨNG DỤNG XỬ LÝ ẢNH TRONG NHẬN DẠNG MÀU SẮC DÙNG RASPBERRY

GVHD: ThS. NGUYỄN DUY THẢO

Sinh viên: **Cao Thị Lan Anh**

MSSV: 21161390

Đỗ Hoàng Thắng

MSSV: 21161191

Tp. Hồ Chí Minh – 5/2024

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN - ĐIỆN TỬ
www.utex.hcmute.edu.vn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

Tp. Hồ Chí Minh, tháng 05, Năm 2024

NHIỆM VỤ BÁO CÁO

Họ và tên sinh viên: Cao Thị Lan Anh **MSSV:** 21161390

Ngành: Công nghệ kỹ thuật Điện tử - Viễn thông **Lớp:** 21161DTCN2

Họ và tên sinh viên: Đỗ Hoàng Thắng **MSSV:** 21161121

Ngành: Công nghệ kỹ thuật Điện tử - Viễn thông **Lớp:** 21161DTCN2

Giảng viên hướng dẫn: ThS. Nguyễn Duy Thảo

Tên đề tài: *Ứng dụng xử lý ảnh trong nhận dạng màu sắc dùng raspberry.*

Nội dung thực hiện:

1. Tìm hiểu cấu trúc của các vi điều khiển Arduino Uno R3 và máy tính nhúng Raspberry Pi 5 sử dụng trong đề tài.
2. Tìm hiểu kiến thức xử lý ảnh, kỹ thuật nhận dạng và xây dựng bài toán nhận dạng màu sắc.
3. Thiết kế phần cứng thực hiện chức năng thu thập dữ liệu từ webcam để nhận dạng màu sắc. Cụ thể 7 màu: đỏ (Red), xanh lá (Green), xanh dương (Blue), đen (Black), tím (Purple), vàng (Yellow), cam (Orange). Máy tính nhúng sẽ xử lý để nhận dạng màu sắc và gửi dữ liệu sang Arduino Uno R3 để hiển thị màu tương ứng.

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN - ĐIỆN TỬ
www.utex.hcmute.edu.vn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

Tp. Hồ Chí Minh, tháng 05, Năm 2024

LỜI CẢM ƠN

Để hoàn thành tốt bài báo cáo này ngoài sự nổ lực của bản thân thì chúng em còn nhận được sự quan tâm giúp đỡ của Thầy và mọi người xung quanh.

Đặc biệt chúng em xin gửi đến ThS Nguyễn Duy Thảo - người đã tận tình hướng dẫn, giúp đỡ chúng em để hoàn thành báo cáo này một lời cảm ơn chân thành và sâu sắc nhất.

Chúng em thực sự biết ơn đến tất cả bạn bè của chúng em, những người đã có những góp ý quý báu trong thời gian chúng em làm dự án. Chúng em cũng phải thừa nhận các nguồn tài nguyên học tập mà chúng em đã nhận được từ nhiều nguồn khác nhau.

Vì thời gian nghiên cứu có hạn, trình độ hiểu biết của bản thân chúng em còn nhiều hạn chế nên bài báo cáo của chúng em không tránh khỏi những thiếu sót, em rất mong nhận được sự góp ý quý báu của tất cả các thầy cô để bài báo cáo của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, tháng 05, Năm 2024

Nhóm thực hiện

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN - ĐIỆN TỬ
www.utex.hcmute.edu.vn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc
Tp. Hồ Chí Minh, tháng 05, Năm 2024

TÓM TẮT

Ngày nay, cùng với sự phát triển và tiến bộ không ngừng của khoa học kỹ thuật thì xử lý ảnh là một trong những đề tài cần được quan tâm và phát triển. Từ những nghiên cứu ban đầu về ảnh trắng đen, ảnh xám, ảnh màu, xử lý ảnh đã được nghiên cứu chuyên sâu và ứng dụng rất nhiều trong cuộc sống. Bên cạnh đó, sự phổ biến của Raspberry Pi với kích thước nhỏ gọn, được xem như máy tính thu nhỏ nên có nhiều đề tài nghiên cứu và ứng dụng Raspberry Pi trong thực tiễn.

Ứng dụng Raspberry Pi vào xử lý ảnh nhằm đưa ra một số giải pháp xử lý ảnh để áp dụng vào đời sống. Trong báo cáo này, chúng em nghiên cứu, thiết kế và thi công hệ thống nhận dạng màu sắc bao gồm 7 màu sắc: đỏ (Red), xanh lá (Green), xanh dương (Blue), đen (Black), tím (Purple), vàng (Yellow), cam (Orange) và cố gắng tìm ra các thuật toán đáp ứng trong thời gian thực, những giải pháp tối ưu, đơn giản nhưng mang lại độ chính xác cao, đáp ứng được nhu cầu thực tế. Chúng em tiến hành mô phỏng những bài toán xử lý ảnh đơn giản, nhận dạng ảnh tĩnh và nhận dạng trực tiếp qua webcam trên chương trình Python và phát triển nhận dạng màu sắc trong thời gian thực trên Raspberry Pi 5. Trong phần ứng dụng nhận dạng màu sắc, chúng em sử dụng phương pháp nhận dạng màu dựa trên không gian màu HSV (Hue-Saturation-Value); ảnh được làm mờ bằng Gaussian Blur để giảm nhiễu, sau đó được chuyển đổi sang không gian màu HSV để dễ dàng nhận biết và phân loại màu sắc; các màu sắc được xác định thông qua phạm vi màu trong không gian màu HSV, mỗi màu sẽ được định nghĩa với giá trị tương ứng; tiếp theo một mask được tạo ra bằng cách so sánh mỗi pixel trong không gian màu HSV với phạm vi màu sắc được định nghĩa, sau đó contours được tìm thấy trong mask này; Bounding box được vẽ quanh các contours có diện tích lớn hơn một ngưỡng được xác định trước, đồng thời thông tin về màu sắc được hiển thị trên bounding box và gửi tín hiệu tương ứng tới Arduino thông qua kết nối serial để hiển thị

màu sắc tương ứng; cuối cùng ảnh với bounding box và thông tin màu sắc được lưu lại và hiển thị trên cửa sổ màn hình.

Tp. Hồ Chí Minh, tháng 05, Năm 2024

Nhóm thực hiện

MỤC LỤC

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Trong những năm gần đây, sự tiến bộ của khoa học kỹ thuật đã đẩy mạnh sự phát triển của lĩnh vực xử lý ảnh. Mặc dù là một ngành khoa học tương đối mới mẻ so với nhiều lĩnh vực khác, nhưng hiện nay, xử lý ảnh đang trở thành một trong những lĩnh vực phát triển nhanh chóng và thu hút sự quan tâm đặc biệt từ cộng đồng khoa học. Điều này thúc đẩy sự xuất hiện của nhiều trung tâm nghiên cứu và ứng dụng đa dạng trong lĩnh vực này.

Xử lý ảnh đóng vai trò quan trọng trong nhiều lĩnh vực thực tế của khoa học và kỹ thuật, cũng như trong cuộc sống hàng ngày. Từ sản xuất và kiểm tra chất lượng, đến việc điều khiển robot và các phương tiện tự lái, cũng như trong lĩnh vực an ninh và giám sát, nhận dạng đối tượng và màu sắc, hay các ứng dụng trong y học và sản xuất video, xử lý ảnh đã chứng minh vai trò quan trọng và không thể phủ nhận của nó.

Trong thời gian gần đây, việc sử dụng Raspberry Pi trong các ứng dụng khoa học và kỹ thuật đã trở nên phổ biến và hiệu quả hơn bao giờ hết. Với các đặc điểm như một máy tính thu nhỏ có kích thước xấp xỉ chiếc điện thoại cầm tay, chạy hệ điều hành mở, được trang bị bộ vi xử lý mạnh mẽ và tiêu thụ điện năng thấp, cho phép ta có thể cấu hình cho Raspberry Pi như một máy tính xử lý các bài toán phức tạp. Nhận thấy điều này, nhóm tác giả quyết định chọn đề tài ***"Ứng dụng xử lý ảnh trong nhận dạng màu sắc dùng Raspberry"*** nhằm đưa ra một số giải pháp thực tế và hữu ích trong đời sống hằng ngày.

1.2 Mục tiêu nghiên cứu

Đề tài này đánh dấu bước đầu tìm hiểu những ứng dụng của xử lý ảnh trong thực tế, đồng thời cũng là cơ hội để áp dụng những kiến thức đã được học. Qua việc nghiên cứu và làm việc chăm chỉ, chúng em đã cải thiện tác phong và hoàn thiện phương pháp, cũng như rèn luyện tư duy nghiên cứu và giải quyết vấn đề. Đề tài đặt ra những mục tiêu cụ thể như sau:

- Tìm hiểu những kiến thức về xử lý ảnh.

- Tìm hiểu về kỹ thuật nhận dạng.
- Tìm hiểu về Raspberry Pi 5 và Arduino Uno R3.
- Xây dựng chương trình nhận dạng màu sắc trên nền tảng Raspberry Pi 5.

1.3 Nội dung nghiên cứu

❖ **NỘI DUNG 1:** Tìm hiểu những kiến thức cơ bản về xử lý ảnh, Raspberry Pi 5, Arduino Uno R3, ngôn ngữ lập trình Python,…

❖ **NỘI DUNG 2:** Tìm hiểu về các kỹ thuật, thuật toán nhận dạng.

❖ **NỘI DUNG 3:** Cài đặt hệ điều hành, thư viện cần thiết cho Raspberry Pi 5.

❖ **NỘI DUNG 4:** Viết chương trình nhận dạng màu sắc.

❖ **NỘI DUNG 5:** Đánh giá kết quả thực hiện.

1.4 Giới hạn của đề tài

Các yếu tố trong điều kiện thực tế có thể ảnh hưởng đến các đặc tính của hệ thống xử lý ảnh thông thường. Dưới đây là một số hạn chế mà nhóm phải đối mặt khi thực hiện đề tài:

- *Điều kiện ánh sáng:* Sự khác biệt giữa ánh sáng ban ngày và ban đêm có thể ảnh hưởng đến hiệu suất của hệ thống. Do đó, nhóm chỉ nghiên cứu và thử nghiệm trong điều kiện ánh sáng ổn định để đạt được kết quả tốt nhất.
- *Khoảng cách nhận dạng:* Việc nhận dạng đối tượng có thể không chính xác nếu khoảng cách từ webcam đến đối tượng vượt quá 1 mét. Do đó, nhóm chỉ tập trung vào các phương pháp nhận dạng đối tượng trong khoảng cách này để đảm bảo độ chính xác.
- *Hạn chế thời gian:* Với hạn chế về thời gian, nhóm sẽ tập trung vào xử lý các bài toán đơn giản và một số ứng dụng phổ biến trên Raspberry Pi 5.

1.5 Bố cục của đề tài

Như vậy, với các yêu cầu về nhiệm vụ và mục tiêu đề ra, báo cáo được xây dựng bao gồm các chương sau:

❖ **Chương 1: Tổng quan**

Chương này trình bày đặt vấn đề dẫn nhập lý do chọn đề tài, mục tiêu, nội dung nghiên cứu, các giới hạn đề tài và bố cục báo cáo.

❖ **Chương 2: Cơ sở lý thuyết.**

Chương này trình bày giới thiệu tổng quan về xử lý ảnh, các thư viện sử dụng; tìm hiểu lý thuyết các thuật toán nhận dạng.

❖ **Chương 3: Thiết kế và thi công.**

Chương này trình bày về các chương trình và xây dụng bài toán nhận dạng màu sắc trên Raspberry Pi 5.

❖ **Chương 4: Kết quả đạt được và đánh giá**

Chương này trình bày kết quả đạt được sau khi thực hiện, đánh giá, nhận xét những gì đã đạt và chưa đạt những gì so với mục tiêu đề ra.

❖ **Chương 5: Kết luận và hướng phát triển**

Chương này đưa ra kết luận về việc thực hiện báo cáo, đồng thời đưa ra hướng phát triển để có được một đề tài hoàn thiện và đáp ứng được nhu cầu cho cuộc sống hiện đại như ngày nay.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

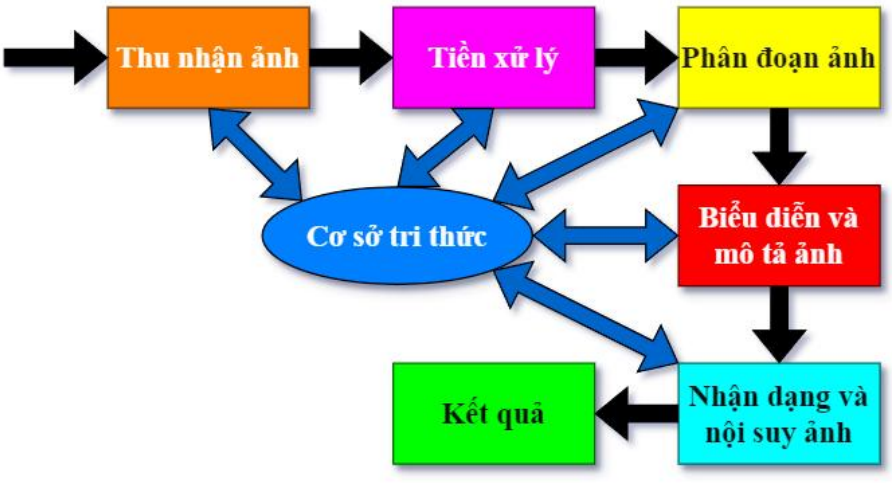
2.1 Tổng quan về xử lý ảnh

2.1.1 Tổng quan

Trong những năm gần đây, xử lý ảnh là một nghành khoa học mới mẻ nhưng tốc độ phát triển nhanh chóng và được nghiên cứu, phát triển bởi các trung tâm nghiên cứu, các trường đại học,… Và với rất nhiều ứng dụng khác nhau.

Các phương pháp xử lý ảnh bắt đầu từ những ứng dụng chính như: nâng cao chất lượng độ sáng và độ phân giải của ảnh, phân tích ảnh. Ứng dụng đầu tiên được biết đến đó chính là nâng cao chất lượng hình ảnh báo được truyền qua cáp từ Luân Đôn đến New York từ những năm 1920. Càng về sau, nhờ sự xuất hiện và phát hiện mạnh mẽ của máy tính đã tạo điều kiện cho các quá trình thực hiện các thuật toán xử lý ảnh được nâng cao và phát triển hơn. Các ứng dụng của xử lý ảnh càng được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như: khôi phục hình ảnh, chỉnh sửa, điều chỉnh độ phân giải; trong lĩnh vực y tế; trong do thám, thám hiểm; truyền và mã hóa; thị giác máy tính, robot; xử lý màu; lĩnh vực nhận dạng,…

Các bước cần thiết trong xử lý ảnh. Trước đây, hình ảnh được thu từ camera là các ảnh tương tự. Gần đây, với sự phát triển không ngừng của công nghệ, ảnh màu hoặc ảnh đen trắng được lấy từ camera, sau đó được chuyển trực tiếp qua ảnh số để dễ dàng cho các bước xử lý tiếp theo. Dưới đây sẽ mô tả về các bước trong xử lý ảnh.



Hình 2.1 *Các bước cơ bản trong xử lý ảnh.*

Thu nhận ảnh: Ảnh được nhận qua camera màu hoặc trắng đen. Thông thường ảnh được nhận qua camera, video, máy scan,…

Tiền xử lý: Sau bộ thu nhận ảnh, hình ảnh có thể có độ tương phản thấp nên cần đưa vào bộ tiền xử lý để nâng cao chất lượng hình ảnh. Bộ tiền xử lý có chức năng lọc nhiễu, nâng độ tương phản để làm cho ảnh rõ hơn và sắc nét hơn.

Phân đoạn ảnh: Còn gọi là phân vùng ảnh. Là tách một ảnh đầu vào thành các vùng thành phần nhỏ hơn để biểu diễn phân tích và nhận dạng ảnh.

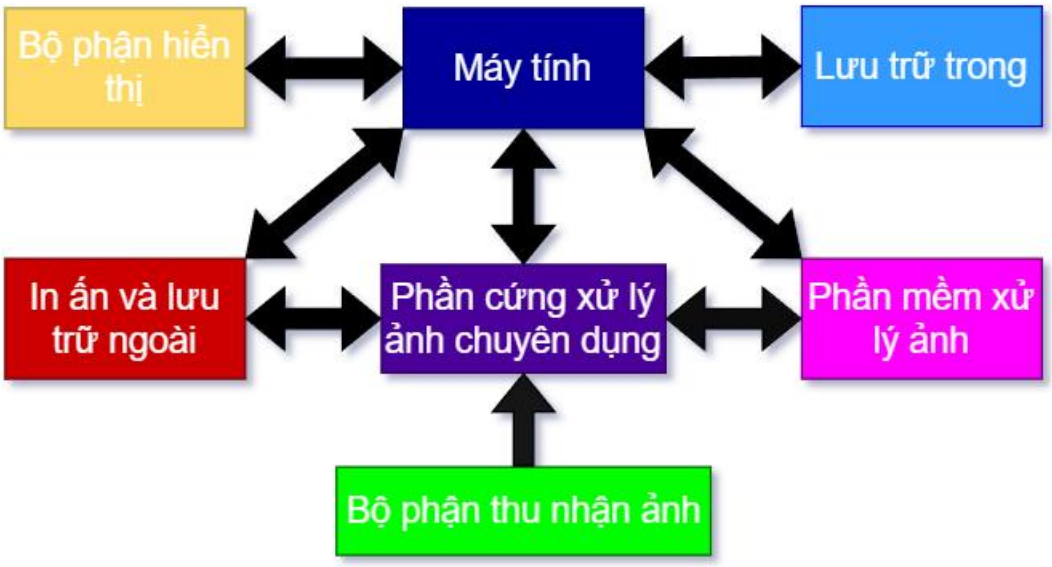
Biểu diễn và mô tả ảnh: Ảnh đã được phân loại chứa nhiều điểm ảnh của vùng ảnh. Việc biến đổi các số liệu này thành dạng thích hợp cho việc xử lý tiếp theo của máy tính. Chúng ta phải tìm các vùng đặc trưng của ảnh, tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc để làm cơ sở cho sự phân biệt giữa lớp đối tượng này với lớp đối tượng khác trong phạm vi của ảnh mà chúng ta nhận được.

Nhận dạng và nội suy ảnh: Nhận dạng ảnh là quá trình xác định ảnh. Bằng cách so sánh mẫu với mẫu chuẩn đã được lưu trữ từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Các mô hình toán học về ảnh được phân loại với hai dạng cơ bản:

- ❖ Nhận dạng theo tham số.
- ❖ Nhận dạng theo cấu trúc.

Cơ sở tri thức: Như đã biết, ảnh là một đối tượng phức tạp về đường nét, độ sáng tối, dung lượng điểm ảnh. Trong nhiều quá trình xử lý và phân tích ảnh. Ngoài việc đơn giản hóa các phương pháp toán học để đảm bảo tiện lợi cho xử lý, người ta mong muốn bắt chước quy trình tiếp nhận và xử lý ảnh theo phương pháp trí tuệ con người. Cho nên, cơ sở tri thức được phát huy và được xử lý theo pháp trí tuệ con người ở nhiều khâu khác nhau.

2.1.2 Các thành phần cơ bản của hệ thống ảnh



Hình 2.2 *Các thành phần cơ bản của hệ thống xử lý ảnh.*

Bộ phận thu nhận ảnh: Máy quay (Camera), máy quét (scaners) chuyên dụng, các bộ cảm biến ảnh.

Phần cứng xử lý ảnh chuyên dụng: Bộ số hóa (chuyển đổi ảnh truyền thống từ bên ngoài thành dạng dữ liệu số mà máy tính có thể hiểu được). Phần cứng thực hiện các thao tác cơ bản để nâng cao tốc độ xử lý ảnh.

Máy tính: Thiết bị thông thường hoặc chuyên dụng.

Bộ phận lưu trữ: Bắt buộc phải có. Lưu trữ tạm thời để phục vụ và sử dụng cho quá trình xử lý hiện tại. Lưu trữ vĩnh viễn là lưu trữ dữ diệu, truy cập không thường xuyên.

Bộ phận hiển thị: Màn hình máy tính…

In ấn: Ghi lại ảnh: máy in, máy chiếu…

2.1.3. Các vấn đề cơ bản trong xử lý ảnh

2.1.3.1. Ảnh và điểm ảnh

Ảnh số là một tập hợp nhiều điểm ảnh, mỗi điểm ảnh được gọi là một pixel. Điểm ảnh là một phần tử của ảnh số tại tọa độ (x, y) biểu diễn một màu sắc nhất

định (có thể là độ xám với ảnh đen trắng). Mỗi điểm ảnh được xem như là một chấm nhỏ li ti trong một tấm ảnh. Bằng phương pháp đo lường và thống kê một lượng lớn các điểm ảnh, chúng ta có thể tái cấu trúc các điểm ảnh này thành một ảnh mới gần giống với ảnh ban đầu.

Hình 2.3 *Ảnh và điểm ảnh.*

2.1.3.2. Độ phân giải của ảnh

Độ phân giải của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiện thị. Khoảng cách giữa các điểm ảnh sao cho mắt người vẫn thấy được sự liên tục của ảnh. Độ phân giải được phân bố theo trục x và y trong không gian hai chiều. Với cùng một ảnh, độ phân giải càng cao thì ảnh càng chứa nhiều thông tin và sắc nét hơn.

Hình 2.4 *Độ phân giải của ảnh.*

Ở hình bên trái có độ phân giải là 220x147 pixels, hình bên phải có độ phân giải là 960x640 pixels. Cho thấy rằng, với độ phân giải càng cao thì độ sắc nét của ảnh càng cao và hình ảnh càng rõ nét.

2.1.3.3. Mức xám của ảnh

Một điểm ảnh có hai đặc trưng cơ bản đó chính là vị trí (x, y) của điểm ảnh và độ xám của ảnh. Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại thời điểm đó.

Hình 2.5 *Sự khác nhau giữa ảnh màu và ảnh xám.*

2.1.3.4. Biến đổi ảnh

Trong xử lý ảnh do số điểm ảnh lớn hơn các tính toán nhiều (độ phức tạp tính toán cao) đòi hỏi dung lượng bộ nhớ lớn, thời gian tính toán lâu. Các phương pháp khoa học kinh điển áp dụng cho xử lý ảnh hầu như khó khả thi. Người ta sử dụng các phép toán tương đương hoặc biến đổi sang miền xử lý khác để dễ tính toán, sau khi đã xử lý dễ dàng, dùng biến đổi ngược để đưa về miền xác định ban đầu, các biến đổi thường gặp trong xử lý ảnh bao gồm:

- Biến đổi Fourier, Cosin, Sin.
- Biến đổi (mô tả) ảnh bằng tích chập, tích Kronecker.
- Các biến đổi khác như KL (Karhumen Loeve), Hadamard.

2.2 Màu sắc

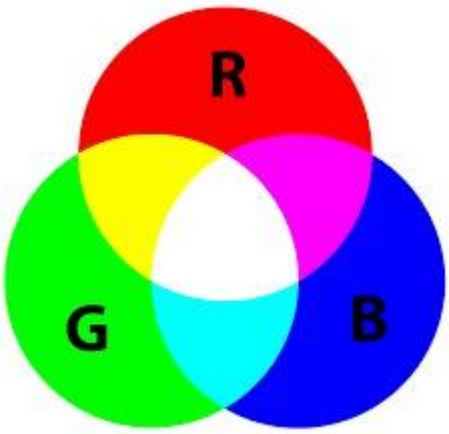
Màu sắc được tạo ra bởi các ánh sáng với các bước sóng khác nhau, mắt người bao gồm ba loại tế bào cảm nhận màu có thể nhìn được bảy triệu màu nhưng thực chất chúng ta chỉ có thể cảm nhận sự khác biệt vài ngàn màu. Một màu có thể được biểu diễn bởi ba thuộc tính: Sắc thái màu (Hue), độ bão hòa (Saturation), và độ chói (Itensity).

Trong xử lý ảnh và đồ họa, mô hình màu là một chỉ số kỹ thuật của một hệ tọa độ màu 3 chiều có thể dùng để biểu diễn tất cả các màu. Ví dụ như mô hình màu RGB (Red, Green, Blue): là một đơn vị tập các màu thành phần sắp xếp theo hình lập phương của hệ trục tọa độ Đề các.

Mục đích của mô hình màu là cho phép biểu diễn một phần các màu nhìn thấy được bằng các chỉ số kỹ thuật quy ước. Sau đây, ta xem xét một số mô hình hay được sử dụng nhất.

2.2.1 Không gian màu RGB

RGB là không gian màu cơ bản được áp dụng từ lâu cho màn hình CRT. Trong không gian màu này, mỗi điểm màu là sự kết hợp của ba thành phần đơn màu (Đỏ-Red, Xanh lục-Green, Xanh da trời-Blue). Đây là một trong những không gian màu được sử dụng phổ biến nhất cho việc xử lý và lưu trữ dữ liệu ảnh số. Tuy nhiên do tính chất tương quan cao giữa các kênh, giá trị cảm nhận không đồng nhất, sự pha trộn giữa dữ liệu thành phần màu và dữ liệu về độ sáng mà không gian RGB không được ưa thích sử dụng cho việc phân tích màu cũng như trong các thuật toán nhận dạng dựa trên màu sắc.



Hình 2.6 *Không gian màu RGB.*

2.2.2 Không gian RGB chuẩn hóa

Không gian RGB chuẩn hóa là không gian màu nhận được từ không gian RGB cơ bản theo công thức chuẩn hóa đơn giản sau đây:

$$r = \frac{R}{R+G+B};\ g = \frac{G}{R+G+B};\ b = \frac{B}{R+G+B} \qquad (3.1)$$

Có thể dễ dàng nhận thấy rằng, trong không gian này, $r + g + b = 1$. Do đó chỉ cần hai trong ba thành phần trên là đủ để biểu diễn không gian màu này, thành phần thứ ba sẽ không còn giá trị và có thể được bỏ qua, để rút ngắn được số chiều của không gian này. Hai thành phần còn lại thường được gọi là các thành phần "màu tinh khiết" (pure color). Thông thường, hai thành phần r và b thường được giữ lại, còn g bị rút bỏ đi. Tính chất cần chú ý của không gian màu này đó là tính bất biến đối với sự thay đổi về hướng của bề mặt. Nghĩa là nếu như không quan tâm đến ánh sáng xung quanh, thì không gian chuẩn hóa RGB là bất biến đối với sự thay đổi về hướng về mặt liên quan đến nguồn chiếu (tất nhiên là dưới một vài giả thiết nhất định). Kết hợp với phép chuyển đổi đơn giản từ không gian màu RGB cơ bản mà không gian RGB chuẩn hóa này ngày càng được sử dụng rộng rãi trong nhiều lĩnh vực, trong đó có lĩnh vực nhận dạng.

2.2.3 Không gian màu HSV

HSV là viết tắt của Hue, Saturation, Value. Trong đó, hai thành phần H, S cũng tương tự như của HIS, riêng V là thành phần biểu thị giá trị độ sáng. Trong không gian màu màu này, các màu đều cũng được biểu diễn dựa trên 3 thành phần H, S, V này. Công thức biến đổi từ không gian RGB sang không gian HSV như sau:

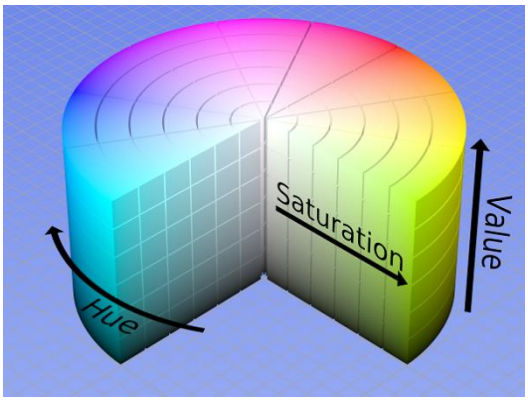
$$H = H' * 60$$

$$S = \frac{Max(R,G,B) - Min(R,G,B)}{Max(R,G,B)}$$

$$V = Max(R,G,B)$$

$$Trong\ đó: H' = \begin{cases} \dfrac{G-B}{Max(R,G,B) - Min(R,G,B)}, Max(R,G,B) = R \\ \dfrac{B-R}{Max(R,G,B) - Min(R,G,B)}, Max(R,G,B) = G \\ \dfrac{R-G}{Max(R,G,B) - Min(R,G,B)}, Max(R,G,B) = B \end{cases}$$

Hệ thống tọa độ có dạng hình trụ và tập màu thành phần của không gian bên trong mô hình màu được xác định là hình nón như trong hình 2.9



Hình 2.7 *Mô hình màu HSV.*

Sắc màu (Hue) hoặc H được đo bởi góc quanh trục đứng với màu đỏ là 0^0, màu lục là 120^0, màu lam là 240^0. Các màu bù nằm ở vị trí đối diện với những màu gốc. S lấy giá trị từ 0 trên đường trục tâm (trục V) đến 1 trên các mặt bên của hình chóp sáu cạnh. Sự bão hòa được hiểu là mức độ tươi của màu, các màu xám từ đen tới trắng có S là 0. Giá trị V được hiểu là độ sáng của màu, V=0 thì là màu đen, V=1 thì là màu có độ sáng tối đa.

2.3 Giới thiệu về phần cứng

2.3.1 Vi điều khiển Arduino Uno R3

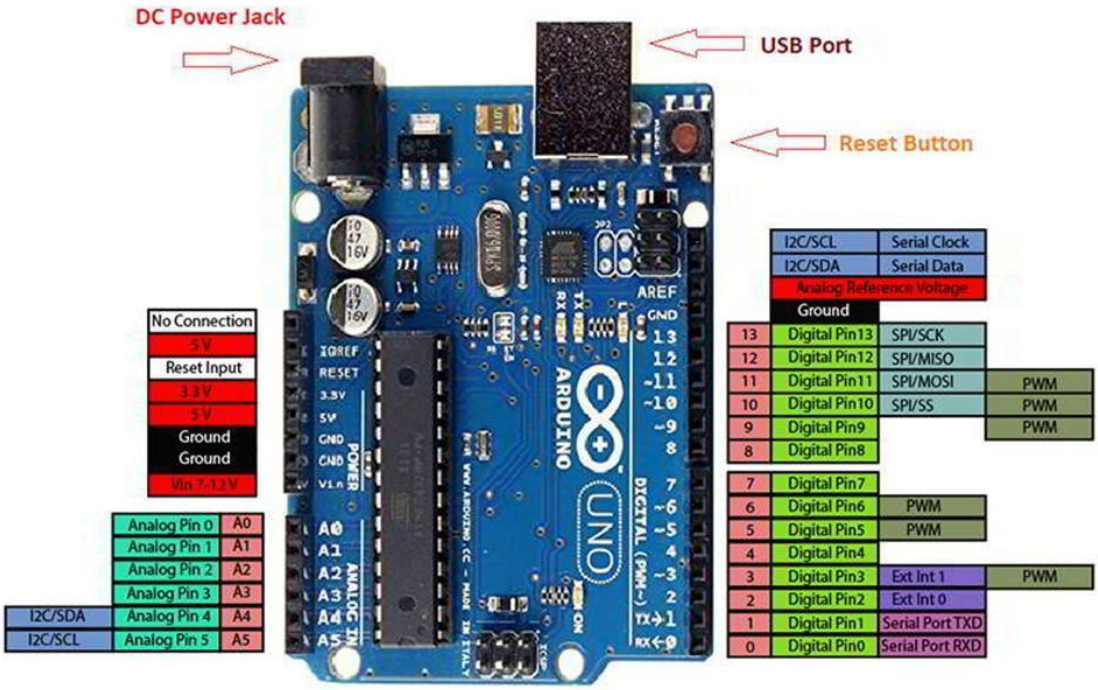
Hiện nay có rất nhiều dòng vi điều khiển khác nhau, có thể làm khối xử lý cho đề tài như Arduino, PIC, AVR, ARM, Raspberry… Đối với đề tài này nhóm quyết định sử dụng Arduino với mã nguồn mở cũng như dễ lập trình bởi có nhiều thư viện hỗ trợ. Tuy nhiên Arduino có khá nhiều loại nên cần lựa chọn sao cho phù hợp với đề tài. Dưới đây là ***bảng 2.1*** liệt kê các dòng Arduino phổ biến hiện nay.

Bảng 2.1 Các dòng Adruino được sử dụng phổ biến hiện nay

Thông số kỹ thuật	Arduino Nano	Arduino Uno	Arduino Mega
Điện áp hoạt động	7 – 12V DC	7 – 12V DC	7 – 12V DC
Dòng điện trên mỗi chân I/O	30mA	40mA	20mA
Số chân Analog	8 chân	6 chân	16 chân
Số chân Digital	14 chân	14 chân	54 chân
Kích thước	18.5x43.18mm	66.3x60mm	101x53.2mm
Giá thành	95.000 VNĐ	94.000 VNĐ	345.000 VNĐ

Dựa vào thông tin trong ***bảng 2.1***, nhóm quyết định lựa chọn Arduino Uno R3 cho đề tài này. Arduino Uno R3 có giá thành phải chăng, các thông số kỹ thuật phù hợp và số chân I/O đủ để sử dụng. Trong khi đó, Arduino Mega có số lượng chân I/O rất lớn, dẫn đến việc có nhiều chân không được sử dụng theo yêu cầu của đề tài. Còn Arduino Nano có giá thành cao hơn Arduino Uno nên cũng không phù hợp với đề tài của nhóm.

Arduino Uno R3, được thể hiện trong ***hình 2.8***, là một bo mạch phát triển được xây dựng dựa trên chip Atmega328P, thường được sử dụng trong các ứng dụng điện tử. Đây là lựa chọn lý tưởng cho những người mới bắt đầu tiếp cận lập trình vi điều khiển. Arduino Uno R3 tích hợp đầy đủ các tính năng, bao gồm các chân I/O số, chân PWM, chân ADC, chân UART và chân I2C. Nó được lập trình thông qua phần mềm Arduino IDE với sự hỗ trợ của các thư viện có sẵn. Arduino Uno R3 có nhiều ứng dụng hữu ích và đa dạng, từ các dự án điện tử cơ bản như bật/tắt đèn, điều khiển servo, đo nhiệt độ và độ ẩm, cho đến các dự án phức tạp hơn như điều khiển robot và các ứng dụng yêu cầu độ chính xác cao. Thông số kỹ thuật của Arduino Uno R3 được mô tả chi tiết trong ***bảng 2.2*** dưới đây.



Hình 2.8 *Sơ đồ chân của board Arduino Uno R3*

Bảng 2.2 Thông số kỹ thuật của board Arduino Uno R3

STT	Thông số kỹ thuật	Giá trị
1	Điện áp hoạt động	7 – 12V DC
2	Dòng điện tiêu thụ	40mA
3	Tần số hoạt động	16 MHz
4	Số chân Digital	14 (6 chân PWM)
5	Số chân Analog	6 (độ phân giải 10 bit)
6	Dòng tối đa trên mỗi chân I/O	30 mA
7	Dòng tối đa 5V	500 mA
8	Dòng tối đa 3.3V	50 mA
9	Bộ nhớ Flash	32KB
10	SRAM	2KB
11	EEPROM	1KB

2.3.2 Máy tính nhúng Raspberry Pi 5

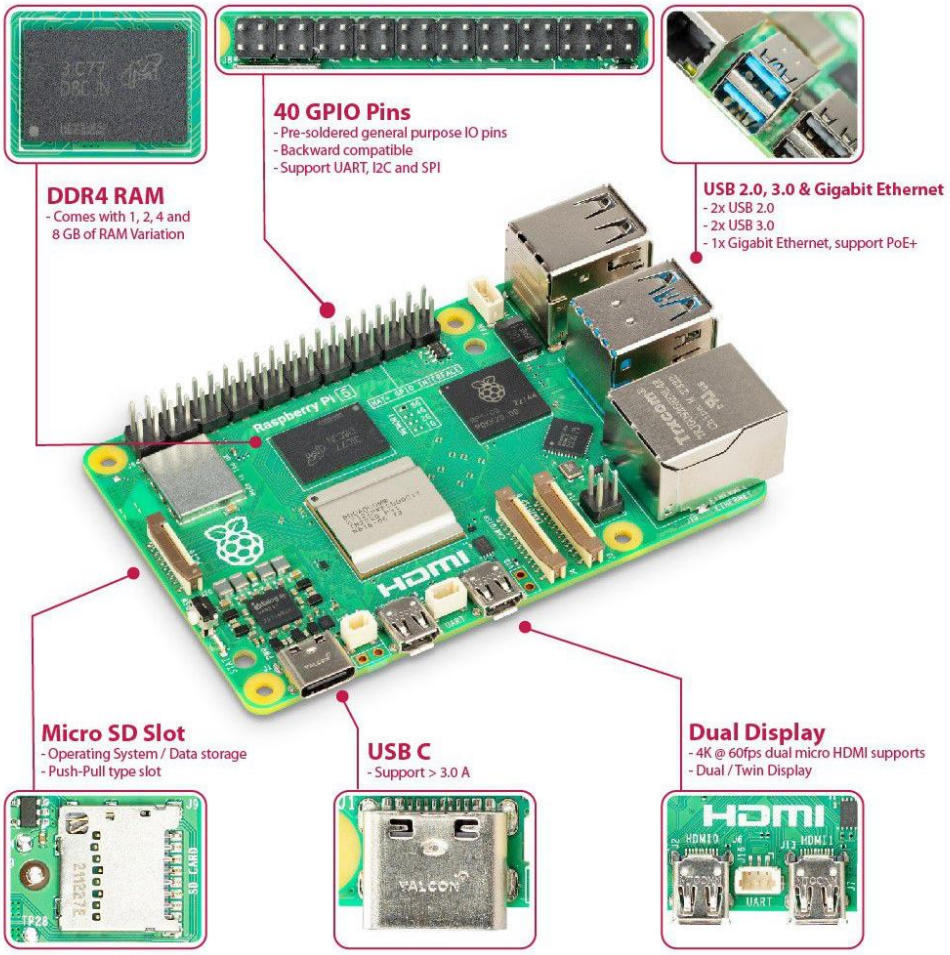
Theo yêu cầu của đề tài, máy tính nhúng được sử dụng phải có khả năng xử lý được các bài toán liên quan đến xử lý ảnh. Vì vậy nhóm quyết định sử dụng máy tính nhúng Raspberry Pi 5 có đủ các chức năng để thực hiện các yêu cầu của đề tài đề ra. Máy tính nhúng Raspberry Pi 5, như được minh họa trong hình 2.3, đã được nghiên cứu và phát triển bởi hãng Raspberry Pi Foundation. Dưới đây là ***bảng 2.3*** liệt kê các dòng Raspberry phổ biến hiện nay.

Bảng 2.3 Các dòng Raspberry được sử dụng phổ biến hiện nay

Thông số kỹ thuật	Pi 3	Pi 4	Pi 5
Điện áp hoạt động	5V 2.5A	5V 3A	5V 5A
RAM	1GB	2GB, 4GB và 8GB	4GB và 8GB
Số chân GPIO	40 chân	40 chân	40 chân
Kích thước	85x56mm	85.6x56.5mm	88x56mm
Giá thành	1.265.000VNĐ	2.700.000VNĐ	3.190.000VNĐ

Dựa vào thông tin trong ***bảng 2.4***, nhóm quyết định lựa chọn Raspberry Pi 5 cho đề tài này. Raspberry Pi 5 có giá thành không quá cao, các thông số kỹ thuật

phù hợp và số cổng kết nối đủ để sử dụng. Trong khi đó, Raspberry Pi 3 có RAM khá thấp chỉ có 1GB, dẫn đến việc sử dụng có thể chậm về các yêu cầu của đề tài. Còn Raspberry Pi 4 có giá thành với phiên bản 8GB chỉ thấp hơn 490.000VNĐ nên nhóm đã lựa chọn bản mới nhất của Raspberry Pi 5 với hiệu suất của mạch đủ đáp ứng được đề tài hoặc thể chạy những bài toán ở cấp bậc cao hơn.



Hình 2.9 *Board Raspberry Pi 5*

Sử dụng hệ đều hành Raspberry Pi OS và sử dụng python để lập trình, Raspberry Pi 5 rất phù hợp cho người mới tiếp cận với dòng vi điều khiển này. Raspberry Pi 5 hỗ trợ kết nối các chuẩn truyền thông không dây như WiFi, Bluetooth, cũng như mở rộng số lượng chân GPIO đến 40 chân. Ngoài ra, Raspberry Pi 5 cũng hỗ trợ giao tiếp với các ngoại vi thông qua các chuẩn giao tiếp như UART, I2C, SPI. Thông số kỹ thuật của vi điều khiển Raspberry Pi 5 được mô tả chi tiết trong ***bảng 2.4***.

Bảng 2.4 Thông số kỹ thuật của board Raspberry Pi 5

STT	Thông số	Giá trị
1	Điện áp hoạt động	5V 5A
2	Bộ xử lý	ARM Cortex-A64
3	Display	2 x 4Kp60 HDMI
4	RAM	8GB
5	Lưu trữ	Micro SD
6	GPIO	40 chân
7	USB	2 x USB 2; 2 x USB 3
8	WiFi	Dual-band 802.11ac
9	Bluetooth	Bluetooth 5 / BLE

2.4 Chuẩn giao tiếp UART

Giới thiệu chung

UART (Universal Asynchronous Receiver - Transmitter) là một giao thức truyền thông phần cứng phổ biến được sử dụng để thiết lập kết nối nối tiếp không đồng bộ và điều chỉnh tốc độ truyền dữ liệu.

Giao thức UART đơn giản và được áp dụng rộng rãi, bao gồm hai đường truyền dữ liệu độc lập: TX (truyền) và RX (nhận). Dữ liệu được truyền và nhận dưới dạng các khung dữ liệu có cấu trúc chuẩn, bao gồm bit bắt đầu (START BIT), số bit dữ liệu, bit kiểm tra (PARITY BIT) và một hoặc nhiều bit dừng (STOP BIT).

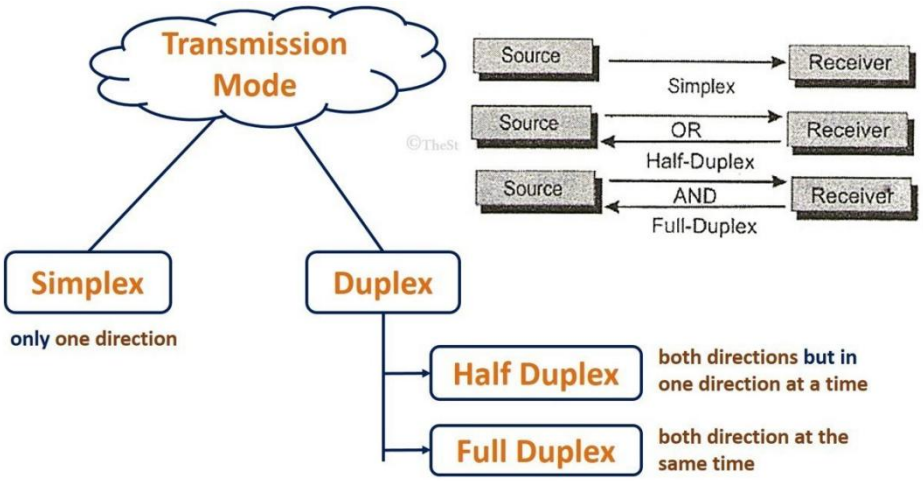
Hoạt động

UART (Universal Asynchronous Receiver-Transmitter) là giao thức truyền dữ liệu nối tiếp, có ba chế độ hoạt động như mô tả ở ***hình 2.10***.

❖ ***Chế độ Simplex:*** Chỉ tiến hành giao tiếp một chiều.

❖ ***Chế độ Half duplex:*** Dữ liệu sẽ di chuyển theo một hướng tại một thời điểm.

❖ ***Chế độ Full duplex:*** Thực hiện giao tiếp đồng thời đến và đi từ mỗi Master và Slave.

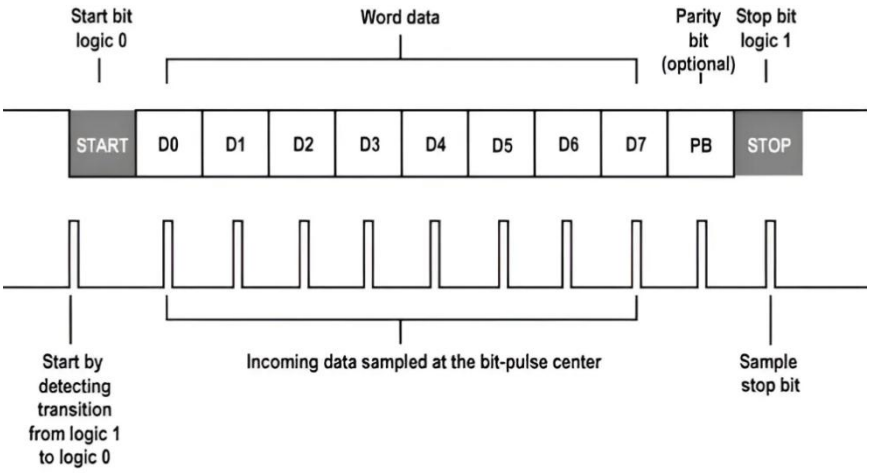


Hình 2.10 *Các chế độ hoạt động của UART*

Giao thức UART (Universal Asynchronous Receiver-Transmitter) cung cấp kết nối truyền thông giữa các chip thông qua việc kết nối chân TX (truyền) của một chip với chân RX (nhận) của một chip khác và ngược lại. Trong quá trình truyền dữ liệu, thông thường sử dụng mức điện áp 3.3V hoặc 5V.

UART là một giao thức giao tiếp giữa một thiết bị master và một thiết bị slave, nghĩa là một thiết bị chỉ giao tiếp với duy nhất một thiết bị khác. Dữ liệu được truyền qua UART song song với việc điều khiển bởi thiết bị. Khi tín hiệu truyền trên chân TX, giao diện UART chuyển đổi dữ liệu song song thành dạng nối tiếp và truyền tới thiết bị nhận thông qua chân RX. Ngược lại, chân RX của thiết bị nhận sẽ chuyển đổi dữ liệu từ dạng nối tiếp sang song song để giao tiếp với các thiết bị điều khiển.

Dữ liệu truyền qua UART được đóng gói thành các gói dữ liệu, gọi là packet như ở ***hình 2.11***. Mỗi gói dữ liệu bao gồm một bit bắt đầu, 5-9 bit dữ liệu (tuỳ thuộc vào bộ UART), một bit kiểm tra chẵn lẻ tùy chọn và một hoặc hai bit kết thúc. Quá trình truyền dữ liệu qua UART diễn ra theo thứ tự này, bắt đầu với bit bắt đầu, theo sau là dữ liệu truyền trong gói dữ liệu, sau đó là bit kiểm tra chẵn lẻ để xác minh tính hợp lệ của dữ liệu. Cuối cùng, một hoặc nhiều bit kết thúc được truyền với mức cao. Quá trình truyền gói dữ liệu qua UART được hoàn thành sau đó.



Hình 2.11 *Cấu trúc một gói tin UART*

2.5 Các phần mềm lập trình

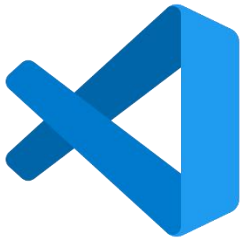
2.5.1 Arduino IDE

Phần mềm IDE Arduino (Arduino Integrated Development Environment) là một môi trường phát triển tích hợp được sử dụng để viết, nạp và sửa lỗi mã chương trình cho các board phát triển Arduino với logo như ***hình 2.12***. Được phát triển bởi Arduino, Arduino IDE cung cấp một giao diện đồ họa đơn giản và dễ sử dụng cho việc lập trình các dự án Arduino mà không cần biết quá nhiều về lập trình.

Hình 2.12 *Logo phần mềm Arduino IDE*

2.5.2 Visual Studio Code

Visual Studio Code chính là ứng dụng cho phép biên tập, soạn thảo hỗ trợ viết code cho nhiều nền tảng lập trình khác nhau. Với việc tích hợp Git trên Visual Studio Code giúp cho việc trao đổi, chỉnh sửa, viết code cho các ứng dụng lớn trở nên dễ dàng hơn, ngoài ra ứng dụng còn hỗ trợ gợi ý từ, các hàm function, sửa lỗi cú pháp và có cộng đồng người sử dụng nhiều. Logo phần mềm Visual Studio Code như ***hình 2.13.***



Hình 2.13 *Logo phần mềm Visual Studio Code*

2.5.3 Ngôn ngữ Python

2.5.3.1 Khái niệm

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.

2.5.3.2 Tính năng

Ngôn ngữ lập trình đơn giản, dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép người lập trình tập trung vào những giải pháp chứ không phải cú pháp.

Miễn phí, mã nguồn mở: Người lập trình có thể tự do sử dụng và phân phối Python, thậm chí là dùng nó cho mục đích thương mại. Vì là mã nguồn mở, không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật.

Khả năng mở rộng và có thể nhúng: Những ứng dụng đòi hỏi sự phức tạp rất

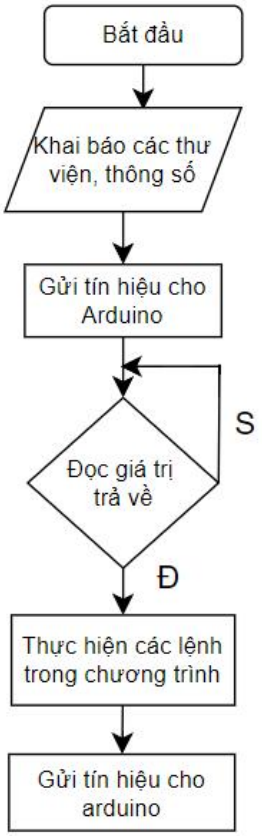
lớn, người lập trình có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào một module của Python. Điều này sẽ cung cấp cho ứng dụng những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.

Hướng đối tượng: Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

2.6 Thư viện pyserial

PySerial là một mô-đun API Python được sử dụng để đọc và ghi dữ liệu nối tiếp vào Arduino hoặc bất kỳ Vi điều khiển nào khác.

Tải thư viện pyserial bằng CMD với cú pháp: **pip install pyserial**



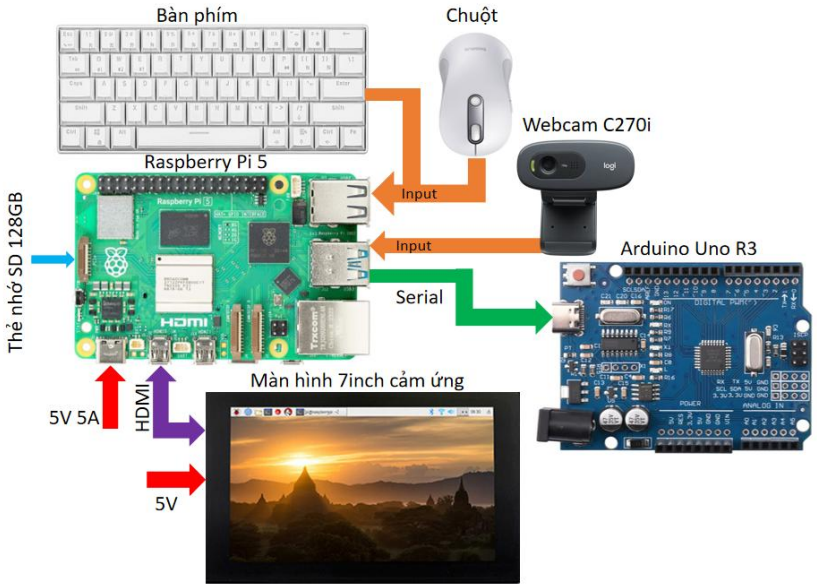
Hình 2.14 *Lưu đồ python giao tiếp Arduino*

Code: phụ lục

CHƯƠNG 3: THIẾT KẾ VÀ THI CÔNG

3.1 Yêu cầu của hệ thống

Với những yêu cầu của đề tài, nhóm đã hình thành sơ đồ khối của hệ thống như sau. Hệ thống gồm các thiết bị như: Raspberry Pi 5, bàn phím, chuột, màn hình cảm ứng, webcam C270i, Arduino Uno R3.



Hình 3.1 *Sơ đồ kết nối hệ thống*

Raspberry: là khối xử lý trung tâm, có chức năng xử lý tín hiệu nhận được từ bàn phím, chuột và webcam. Sau đó xuất tín hiệu, kết quả ra màn hình và gửi tín hiệu đến Arduino Uno R3 để điều khiển đèn sáng tương ứng với màu sắc.

Bàn phím, chuột: Có chức năng để lập trình, điều khiển Rasberry.

Webcam: Có chức năng lấy dữ liệu cần xử lý.

Arduino: Có chức năng hiển thị màu sắc.

Màn hình: Có chức năng hiển thị kết quả.

3.2 Sơ đồ khối và chức năng

Dựa vào yêu cầu thiết kế hệ thống đã đưa ra, nhóm đã thiết kế sơ đồ khối của hệ thống như ***hình 3.2.***

Hình 3.2 *Sơ đồ khối hệ thống*

Trước tiên, ta sẽ thiết lập một webcam để nhận diện màu sắc. Để phát hiện màu sắc, ta chỉ cần vùng màu. Một khi đã có vùng màu, ta có thể xác định được màu đó là màu gì.

3.2.1. Đọc ảnh

Để truy cập vào webcam, ta cần dùng thư viện imutils – một bộ các chức năng xử lý hình ảnh giúp làm việc trên OpenCV dễ dàng hơn.

Ban đầu chương trình sẽ thiết lập kết nối webcam hoặc webcam máy tính và lấy từng frame ảnh để xử lý.

3.2.2. Chuyển đổi sang HSV

Chúng tôi bắt đầu vòng lặp vô hạn các khung hình trong video. Sau đó tiến hành bước tiền xử lý bằng cách thay đổi kích thước nó để có chiều rộng là 450 pixel và chuyển sang hsv với câu lệnh trong OpenCV: hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV) từ thư viện OpenCV để chuyển đổi ảnh từ không gian màu BGR (Blue-Green-Red) sang HSV (Hue-Saturation-Value). Điều này là cần thiết vì không gian màu HSV thích hợp hơn cho việc phân tích màu sắc trong ảnh.

3.2.3. Xử lý ảnh

Sau khi đã chuyển đổi ảnh sang không gian màu HSV, cần áp dụng các phép xử lý để cải thiện chất lượng ảnh hoặc loại bỏ nhiễu. Quá trình "xử lý ảnh" được thực hiện thông qua việc áp dụng một bộ lọc làm mờ (blurring) để làm giảm nhiễu và làm mịn ảnh trước khi thực hiện nhận diện màu sắc với câu lệnh trong OpenCV: blurred = cv2.GaussianBlur(frame, (11, 11), 0).

3.2.4. Phát hiện màu sắc

Bước này tập trung vào việc phát hiện và nhận diện các vùng màu sắc cụ thể trong ảnh. Cụ thể như sau:

- Đầu tiên, chúng ta sẽ lặp qua danh sách các màu sắc cụ thể mà chúng ta quan tâm nhận diện. Mỗi màu sẽ được định nghĩa bởi ba thành phần: tên màu, giới hạn dưới và giới hạn trên trong không gian màu HSV, cùng với mã màu RGB tương ứng.

- *Tạo mặt nạ màu:* Dùng hàm cv2.inRange() để tạo một mặt nạ cho mỗi màu sắc dựa trên giới hạn màu sắc trong không gian màu HSV. Mặt nạ này sẽ là một ảnh nhị phân, trong đó các điểm ảnh nằm trong phạm vi màu sắc của mỗi màu sẽ có giá trị 255 (trắng), còn lại sẽ có giá trị 0 (đen).

- *Loại bỏ các vùng không cần thiết:* Đôi khi, có thể có các vùng màu không mong muốn trong ảnh. Chúng ta có thể loại bỏ những vùng này bằng cách sử dụng toán tử bitwise để áp dụng mặt nạ màu lên ảnh gốc và loại bỏ những vùng không cần thiết.

● *Tìm các contour:* Sử dụng hàm cv2.findContours() để tìm các contour trong mặt nạ màu đã được tạo ra. Contour là một tập hợp các điểm liên tục trên biên của các vùng màu sắc trong ảnh.

● *Phân tích các contour:* Chúng ta sẽ lặp qua danh sách các contour đã tìm được. Đối với mỗi contour, chúng ta sẽ tính toán diện tích của contour (sử dụng cv2.contourArea()) để đánh giá kích thước của vùng màu sắc. Các contour với diện tích lớn hơn một ngưỡng được xác định trước sẽ được coi là các vùng màu sắc cần quan tâm.

● *Vẽ đường viền và hiển thị tên màu:* Đối với các contour được xác định là vùng màu sắc cần quan tâm, chúng ta sẽ vẽ một hình chữ nhật bao quanh vùng đó và hiển thị tên màu tương ứng lên ảnh. Đồng thời, chúng ta gửi dữ liệu tương ứng với màu sắc này tới Arduino để thực hiện hiển thị màu sắc.

3.2.5. Gửi dữ liệu đến Arduino

Trong bước này sẽ gửi dữ liệu tương ứng với màu sắc được phát hiện từ quá trình "Phát hiện màu sắc" tới Arduino thông qua kết nối Serial.

3.2.6. Hiển thị màu

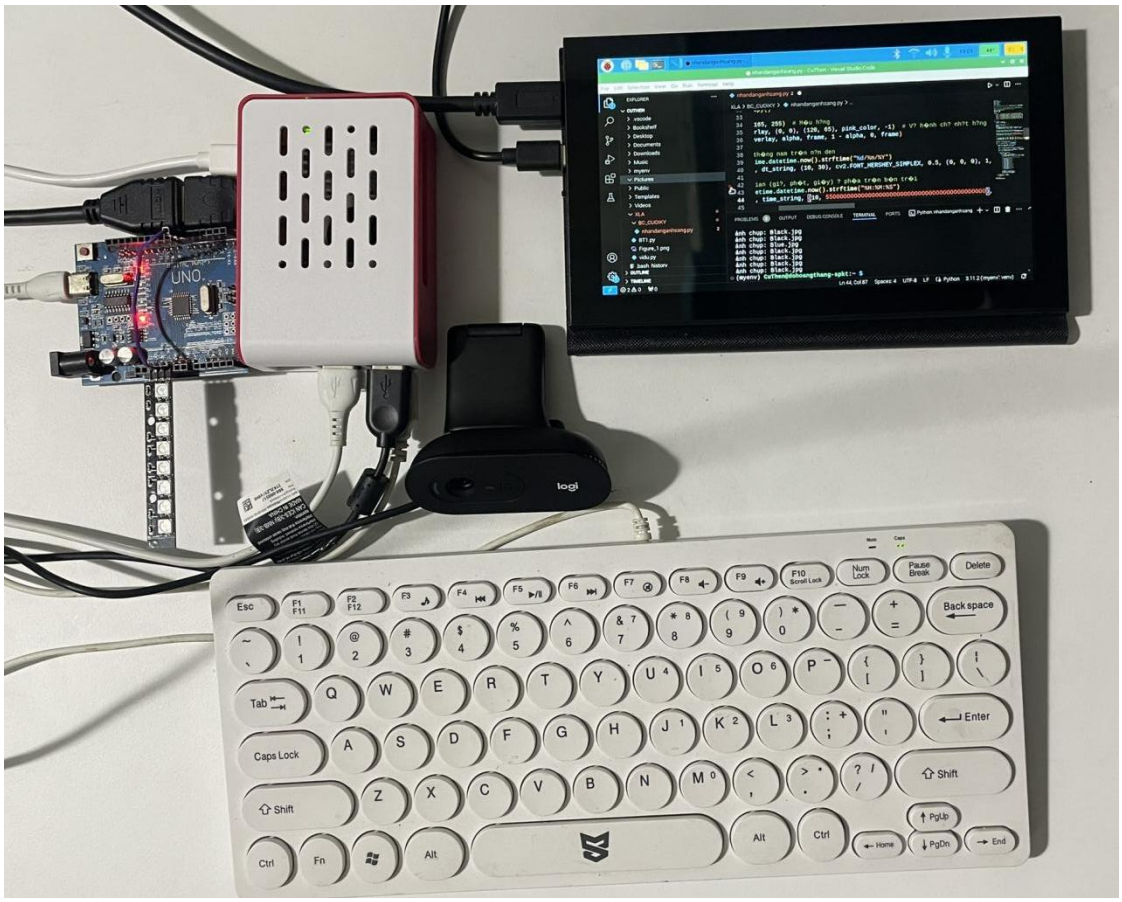
Cuối cùng Arduino hiển thị màu sắc tương ứng đã nhận được từ máy tính thông qua kết nối Serial.

CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC VÀ ĐÁNH GIÁ

4.1 Kết quả về phần cứng

4.1.1 Mô hình hoàn chỉnh

Sau khi trải qua các bước thi công phần cứng như lắp ráp các phần cứng vào Raspberry, viết chương trình và kiểm tra thì phần cứng hoàn chỉnh cho mạch được trình bày như ***hình 4.1*** bên dưới.



Hình 4.1 *Kết quả thi công phần cứng*

4.1.2 Kết quả giao diện hiển thị

Đối với giao diện hiển thị, các bố cục giao diện được trình bày trực quan và thân thiện với người dùng, với hàng loạt các tính năng hữu ích. Thông qua giao diện, người dùng có thể tự do quan sát và theo dõi trạng thái của các màu sắc ở mọi lúc mọi nơi, chỉ cần có màn hình. Giao diện được trình bày như ***hình 4.2***.

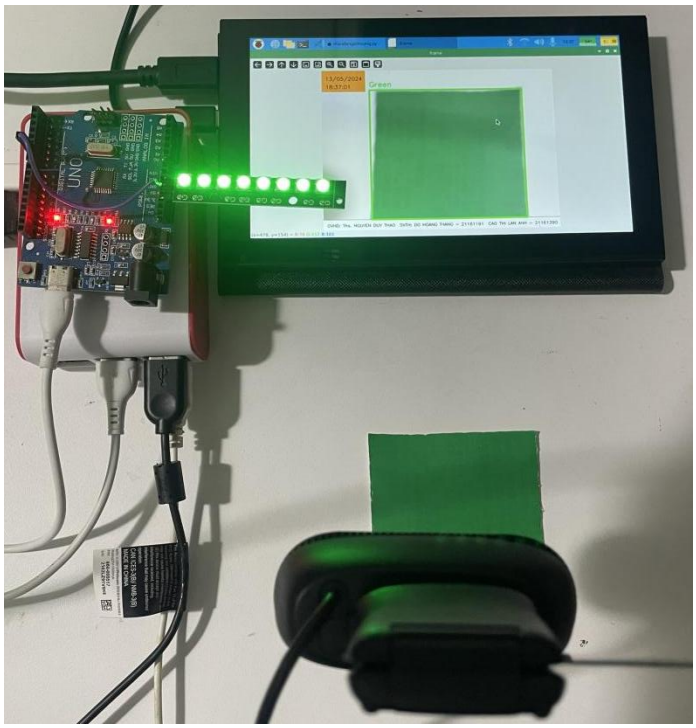
Hình 4.2 *Giao diện hiển thị python*

4.2 Kiểm tra hoạt động của hệ thống

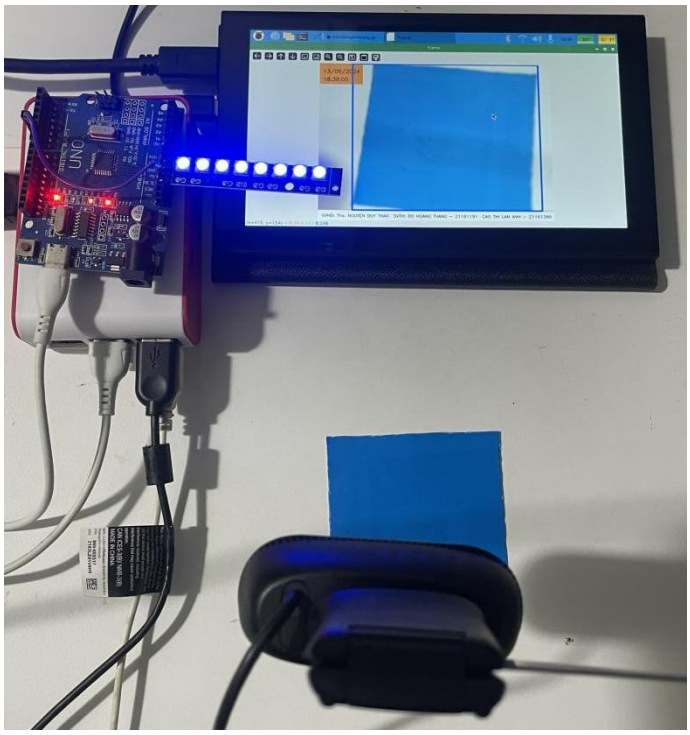
Để kiểm tra và đánh giá hoạt động của hệ thống, nhóm đưa ra các bài thử nghiệm về hoạt động của hệ thống đối với các điều kiện khác nhau để đánh giá được hệ thống một cách trực quan, chi tiết hơn.

4.2.1 Thử nghiệm với các khoảng cách khác nhau

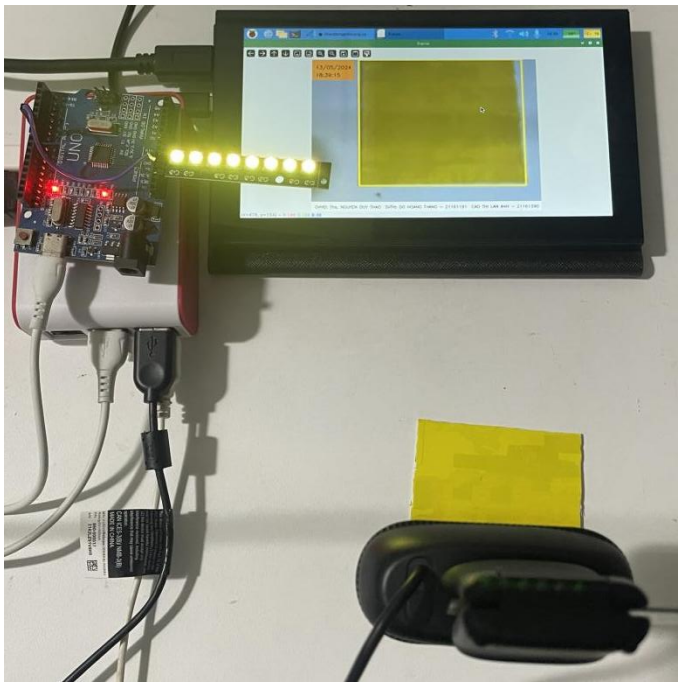
Thử nghiệm lần 1: Với khoảng cách 10cm từ webcam đến màu sắc.



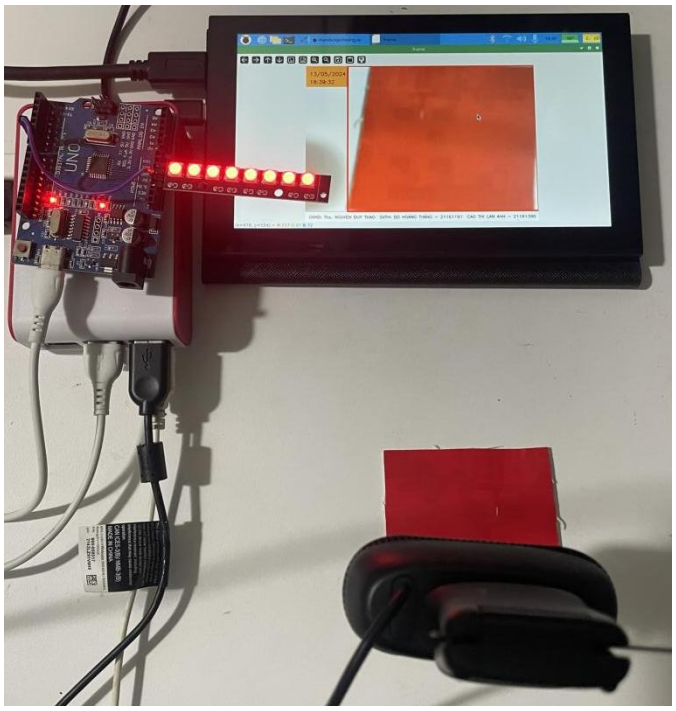
Hình 4.3 *Thử nghiệm màu xanh lá với khoảng cách 10cm*



Hình 4.4 *Thử nghiệm màu xanh dương với khoảng cách 10cm*

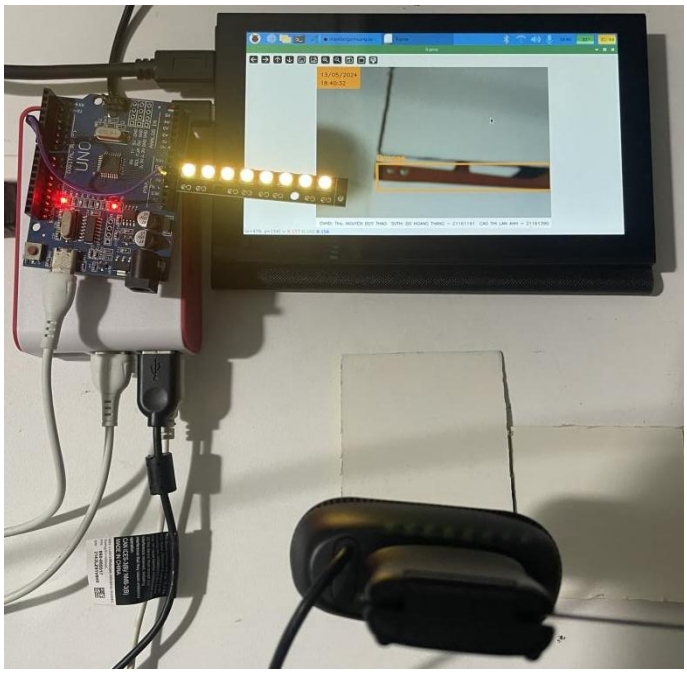


Hình 4.5 *Thử nghiệm màu vàng với khoảng cách 10cm*

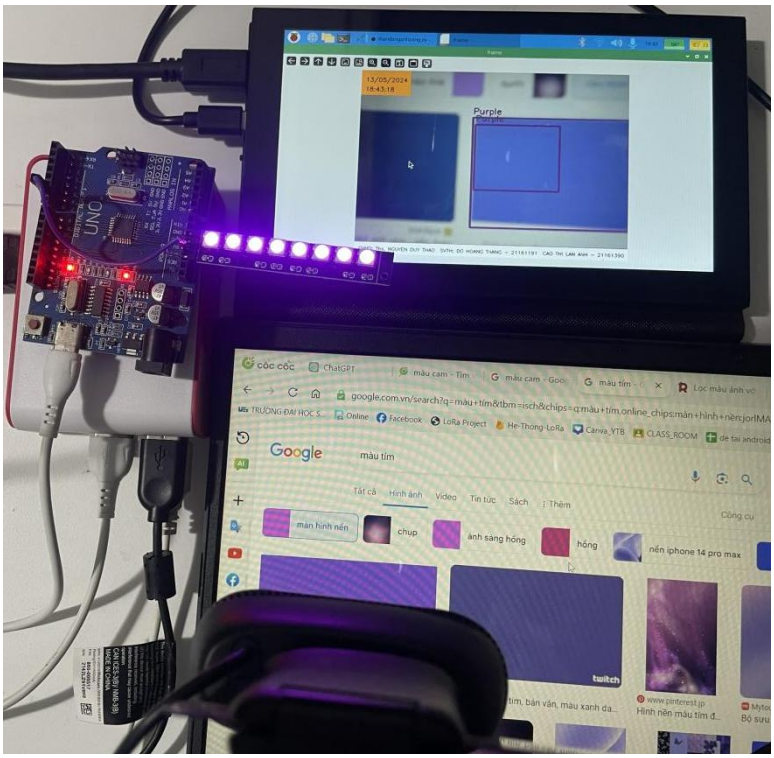


Hình 4.6 *Thử nghiệm màu đỏ với khoảng cách 10cm*

Hình 4.7 *Thử nghiệm màu đen với khoảng cách 10cm*

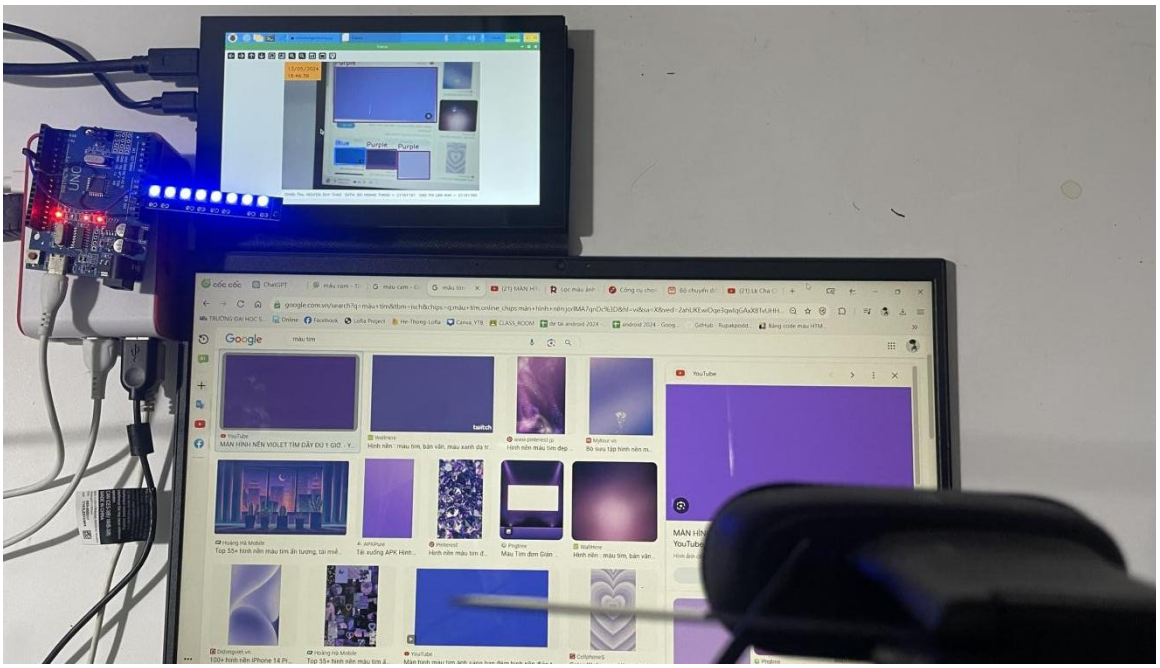


Hình 4.8 *Thử nghiệm màu cam với khoảng cách 10cm*



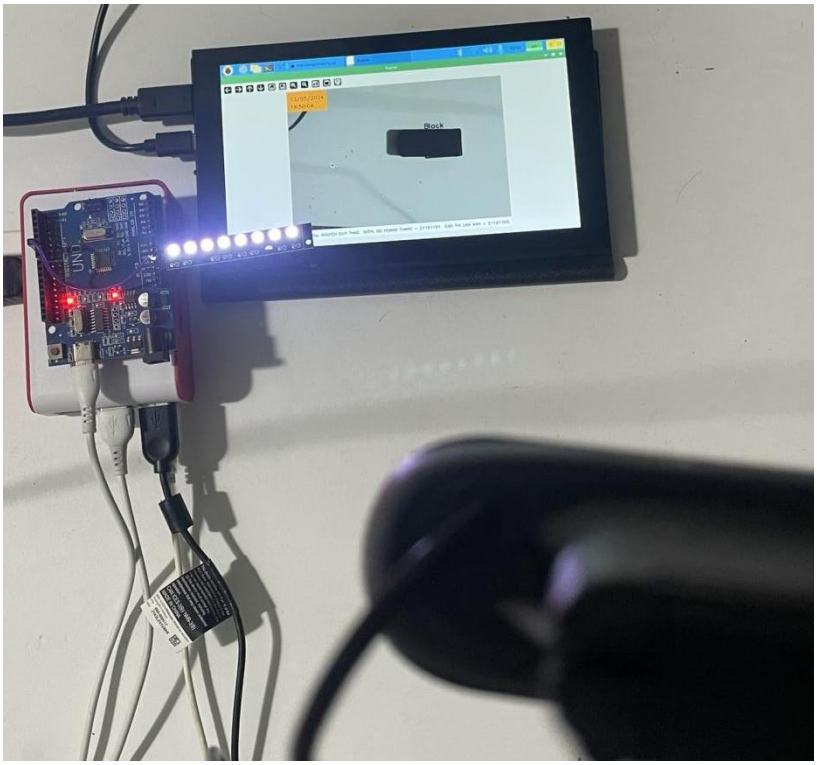
Hình 4.9 *Thử nghiệm màu tím với khoảng cách 10cm*

Thử nghiệm lần 2: Với khoảng cách 40cm từ webcam đến màu sắc.

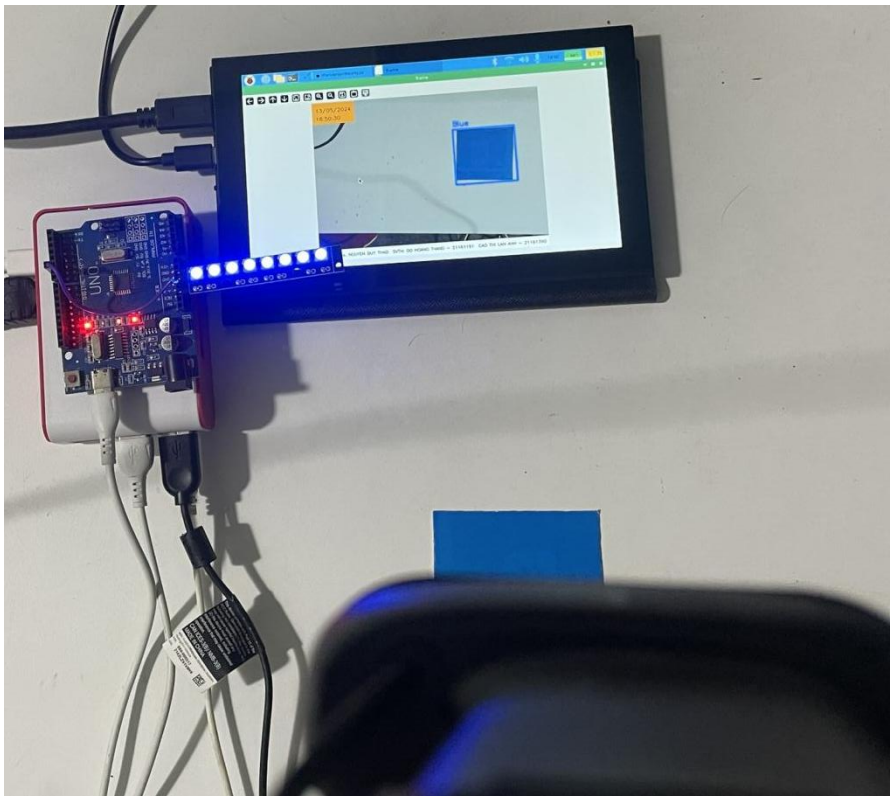


Hình 4.10 *Thử nghiệm màu tím với khoảng cách 40cm*

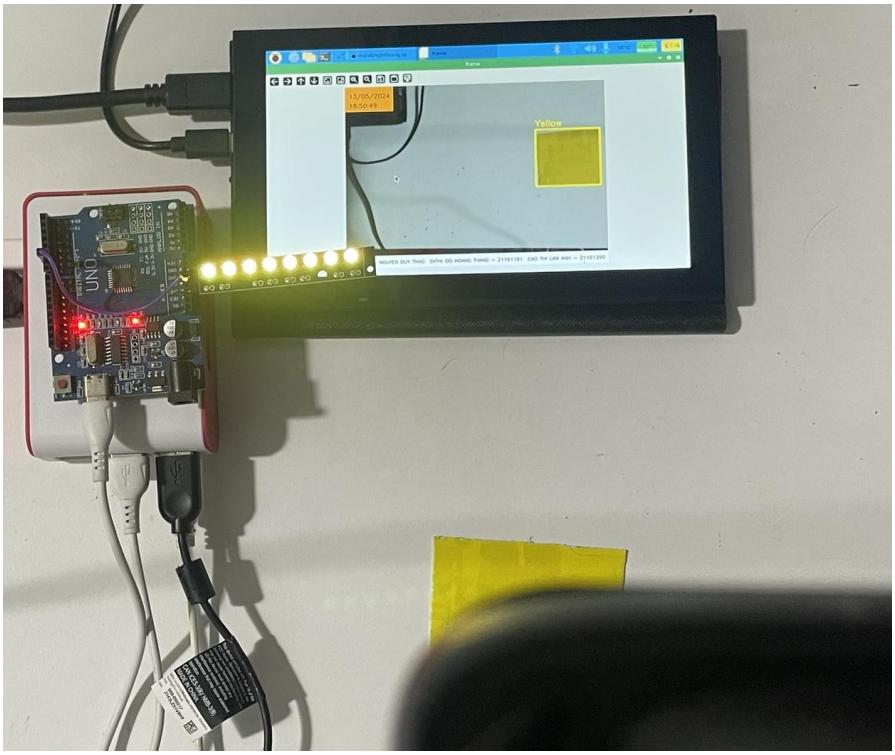
Hình 4.11 *Thử nghiệm màu xanh lá với khoảng cách 40cm*



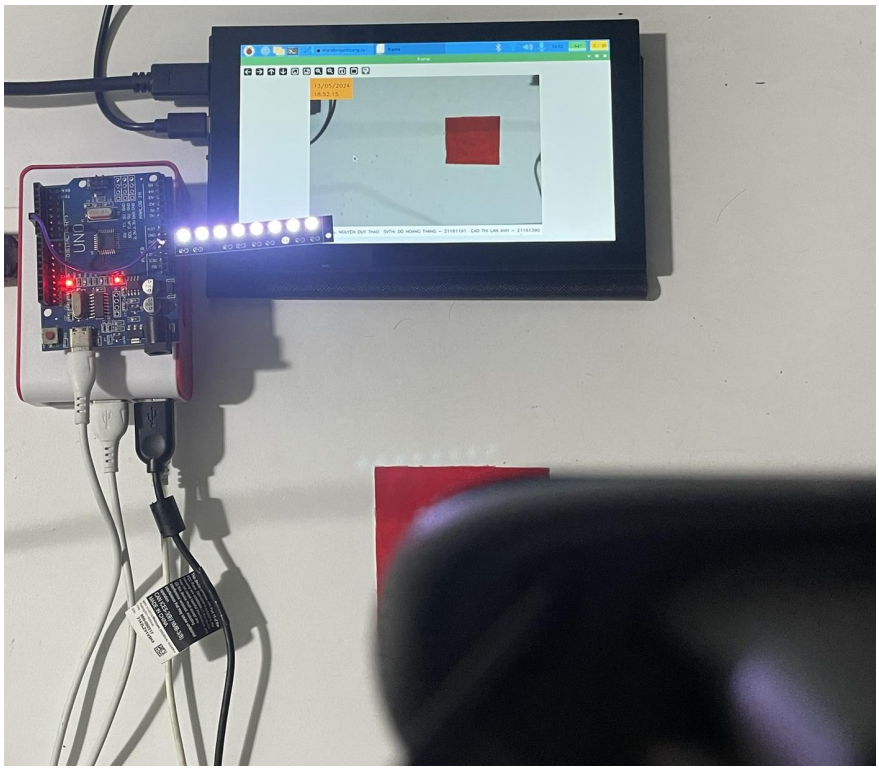
Hình 4.12 *Thử nghiệm màu đen với khoảng cách 40cm*



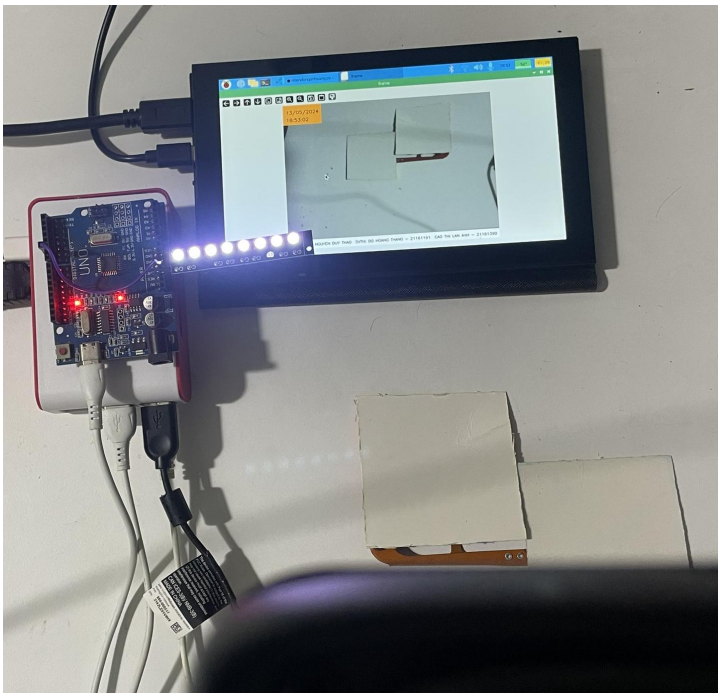
Hình 4.13 *Thử nghiệm màu xanh dương với khoảng cách 40cm*



Hình 4.14 *Thử nghiệm màu vàng với khoảng cách 40cm*



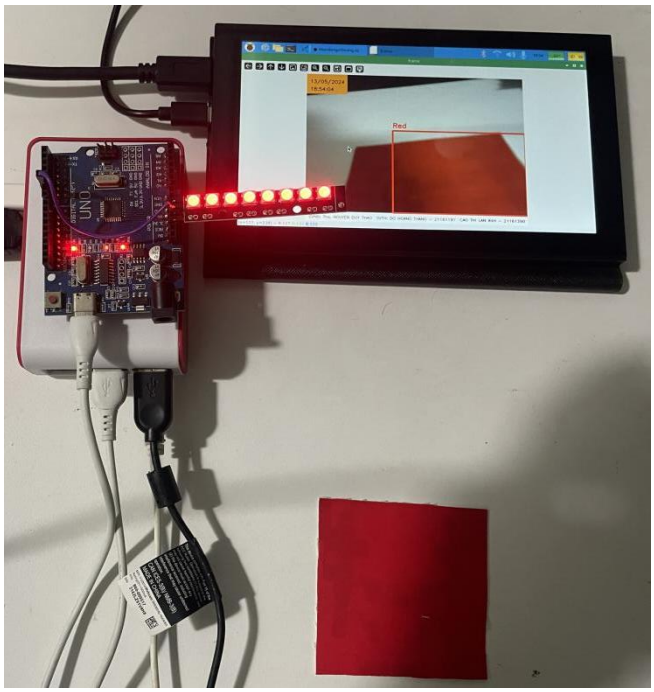
Hình 4.15 *Thử nghiệm màu đỏ với khoảng cách 40cm*



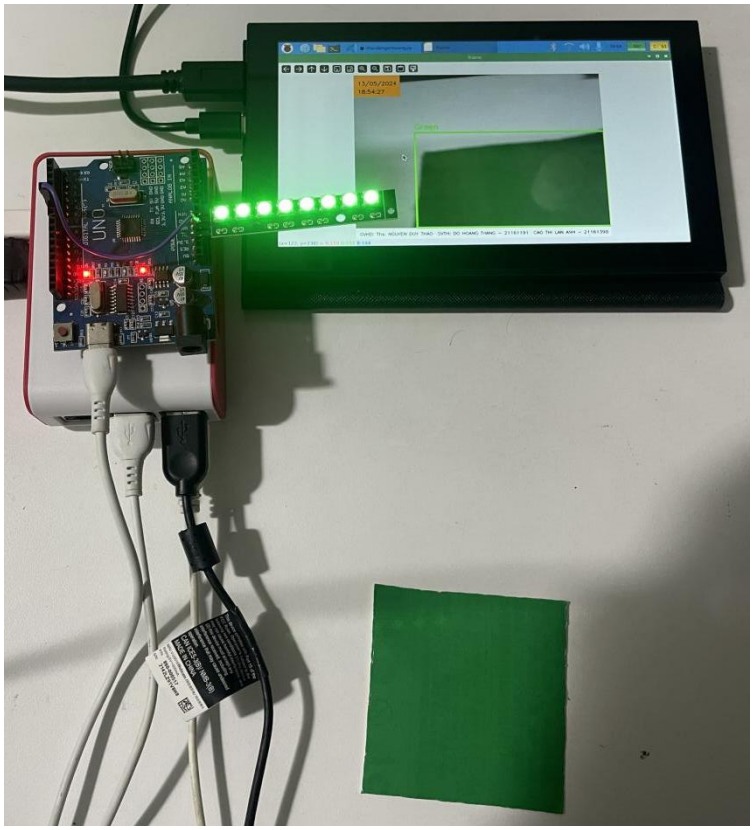
Hình 4.16 *Thử nghiệm màu cam với khoảng cách 40cm*

4.2.2 Thử nghiệm với góc nghiêng

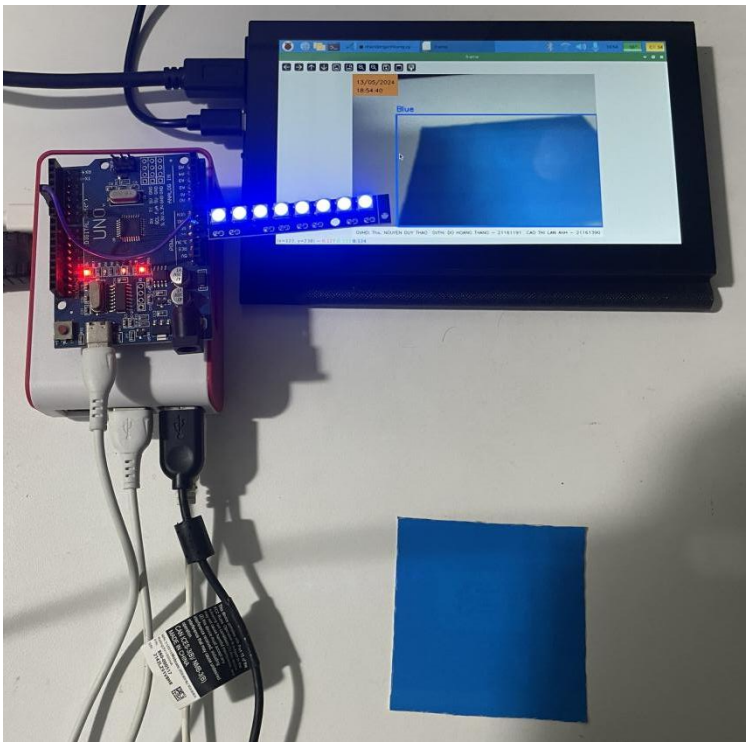
Thử nghiệm với góc nghiêng 45^0:



Hình 4.17 *Thử nghiệm màu đỏ với khoảng cách 10cm và góc nghiêng 45^0*



Hình 4.18 *Thử nghiệm màu xanh lá với khoảng cách 10cm và góc nghiêng 45^0*



Hình 4.19 *Thử nghiệm màu xanh dương với khoảng cách 10cm và góc nghiêng 45^0*

Kết quả cho thấy rằng độ chính xác khá cao ở cả khoảng cách và góc nghiêng. Nhưng về khoảng cách, có một số màu không nhận diện được cụ thể với khoảng cách 40cm thì màu đỏ (màu vật thể khá bóng nên để webcam xa không diện được màu đỏ), màu cam (vì màu vật thể quá nhỏ khi đưa webcam ra xa vật thể) không nhận diện được hai màu này.

Đối với góc nghiêng gần như chính xác khá cao.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong suốt quá trình thực hiện đề tài nhóm đã xây dựng được một hệ thống nhận diện màu sắc. Đề tài được xây dựng trên hệ điều hành Linux để mô phỏng và Raspberry Pi 5 để sử dụng thực tế với sự hỗ trợ của phương tiện phần mềm xử lý ảnh là thư viện mã nguồn mở OpenCV và dlib với nhiệm vụ là phát hiện màu sắc.

Trên thực tế chương trình này đã đáp ứng được các yêu cầu sau: Chương trình ứng dụng thực tế chạy trên Raspberry Pi 5. Chương trình ứng dụng trên nền Linux được viết từ ngôn ngữ Python và được truy xuất hàm từ thư viện mã nguồn mở OpenCV và dlib để xử lý ảnh trên Raspberry Pi 5. Với kết quả chạy trên thiết bị Raspberry Pi 5 tốc độ xử lý 2.4GHz, RAM 8GB. Kết quả đáp ứng tốt với yêu cầu của hệ thống.

5.1.1 Ưu điểm

Là chương trình được lập trình do đó tiết kiệm được chi phí thiết kế và thi công phần cứng cho một hệ thống nhận dạng với chức năng tương tự.

Linh hoạt và tiện lợi do kích thước của Raspberry Pi 5 rất nhỏ nên có thể tích hợp trực tiếp vào các hệ thống. Từ đó có thể phát triển được ứng dụng một cách hiệu quả hơn.

Kết quả mô phỏng và ứng dụng thực tế trên Raspberry Pi 5 đều cho độ chính xác cao.

Chương trình đáp ứng được yêu cầu thực tế, thời gian xử lý và nhận dạng rất nhanh để đáp ứng cho một ứng dụng yêu cầu thời gian thực.

5.1.2 Nhược điểm

Giải thuật chính sử dụng trên Raspberry Pi 5 để nhận dạng màu sắc là không gian màu HSV đây là giải thuật đơn giản và phụ thuộc nhiều vào điều kiện ngoại cảnh như ánh sáng, góc quay, tọa độ di chuyển nên khi thực hiện hương trình còn một số nhược điểm như sau:

➢ Khi màu nhạt thì phát màu sắc không chính xác, vị trí đánh dấu màu sắc bị lệch do phản chiếu ánh sáng khi ánh sáng chiếu trực tiếp vào vật thể màu.

➢ Giải thuật phát hiện nhiều màu sắc một lúc nên chưa nhận dạng được đâu là màu gì, đâu là màu ta cần nhận dạng.

➢ Hướng màu sắc phải tương đối trực diện. Nghiêng, xoay trái, xoay phải góc lớn hơn 45° thì giải thuật có thể không phát hiện được màu sắc.

5.2 Hướng phát triển

5.2.1. Hướng khắc phục

Tiền xử lý ảnh qua nhiều bước để có được kết quả chuẩn xác nhất.

Ứng dụng chương trình trong điều kiện ngoại cảnh phù hợp về khoảng cách, ánh sáng (lắp đặt đèn chiếu sáng), hạn chế số lượng và thời gian mà màu sắc làm nhiễu trong khung ảnh, hướng và góc nghiêng của vật thể… để đảm bảo độ tin cậy và giảm sai số tới mức thấp nhất cho hệ thống.

Cải thiện chất lượng webcam không phụ thuộc vào điều kiện ánh sáng để thu được kết quả tốt nhất, webcam có độ phân giải cao….

5.2.2 Đề xuất hướng phát triển

Nghiên cứu cải tiến những giải thuật phát hiện, nhận diện hiệu quả, chính xác và tối ưu hơn.

Xây dựng hệ thống với những mục đích đa dạng như hệ thống phân loại sản phẩm bằng màu sắc, hoặc là thiết bị học màu sắc cho học sinh.

Để tiện lợi và chủ động hơn trong sử dụng thì việc nghiên cứu và phát triển hệ thống trên các thiết bị di động là cần thiết ví dụ lập trình phần mềm, ứng dụng có thể cài đặt trên điện thoại thông minh.

Phát hiện và phân tích cấu trúc màu sắc trong không gian 3 chiều.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Thanh Hải, "*Giáo trình: Xử Lý Ảnh*", Xuất bản Đại Học Quốc Gia TP.HCM, 2013.

[2] Simon Monk, "Programming the Raspberry Pi Getting Started with Python", 2015.

[3] Joseph Howse, Prateek Joshi, Micheal Beyeler, "OpenCV: Computer Vision Projects with Python", 2016.

[4] Lương Mạnh Bá. Nguyễn Thanh Thủy, Chương 7, Nhập Môn Xử Lý ảnh số, Nxb Khoa học và Kỹ Thuật, 2002, trang 179-208.

PHỤ LỤC

1. Chương trình chính nhận dạng màu sắc

```
import cv2
import numpy as np
import imutils
import serial
import time
import datetime
ser = serial.Serial('/dev/ttyUSB0', 9600,
parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE,
bytesize=serial.EIGHTBITS, timeout=1)
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
colors = [
    ("Yellow", np.array([25, 70, 125]), np.array([35, 255,
255]), (0, 255, 255)),
    ("Green", np.array([65, 60, 60]), np.array([80, 255,
255]), (0, 255, 0)),
    ("Red", np.array([0, 70, 150]), np.array([15, 255, 255]),
(0, 0, 255)),
    ("Blue", np.array([90, 60, 70]), np.array([114, 255,
255]), (255, 0, 0)),
    ("Purple", np.array([125, 30, 50]), np.array([140, 255,
255]), (128, 0, 128)),
    ("Black", np.array([0, 0, 0]), np.array([180, 255, 30]),
(0, 0, 0)),
    ("Orange", np.array([5, 50, 50]), np.array([15, 255,
255]), (0, 165, 255))
]
while True:
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
```

```
    overlay = frame.copy()
    alpha = 1
    pink_color = (0, 165, 255)
    cv2.rectangle(overlay, (0, 0), (120, 65), pink_color, -1)
    cv2.addWeighted(overlay, alpha, frame, 1 - alpha, 0,
frame)
    dt_string = datetime.datetime.now().strftime("%d/%m/%Y")
    cv2.putText(frame, dt_string, (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
    time_string = datetime.datetime.now().strftime("%H:%M:%S")
    cv2.putText(frame, time_string, (10, 55),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
    cv2.rectangle(frame, (0, 450), (640, 480), (0, 0, 0),
2)
    cv2.rectangle(frame, (0, 450), (640, 480), (255, 255,
255), -1)
    cv2.putText(frame, "GVHD: Ths. NGUYEN DUY THAO  SVTH: DO
HOANG THANG - 21161191  CAO THI LAN ANH - 21161390", (10,
470), cv2.FONT_HERSHEY_SIMPLEX, 0.38, (0, 0, 0), 1,
cv2.LINE_AA)
    #làm mờ khung hình và chuyển đổi sang không gian màu HSV
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)


    #lặp qua từng màu trong danh sách màu
    for color_name, lower, upper, color in colors:
        #tạo mask dựa trên ngưỡng màu
        mask = cv2.inRange(hsv, lower, upper)
        #loại bỏ các nhiễu và cải thiện mask bằng bitwise_and
        mask = cv2.bitwise_and(mask, mask,
mask=cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY), 10,
255))
        #tìm contours trong mask
```

```
        cnts = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        #duyệt qua các contour tìm được
        for c in cnts:
            area = cv2.contourArea(c)
            if area > 5000:
                #vẽ khung giới hạn và ghi tên màu
                x, y, w, h = cv2.boundingRect(c)
                cv2.rectangle(frame, (x, y), (x + w, y + h),
color, 3)
                cv2.putText(frame, color_name, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)
                if color_name == "Yellow":
                    ser.write(b'0\n')
                elif color_name == "Green":
                    ser.write(b'1\n')
                elif color_name == "Red":
                    ser.write(b'2\n')
                elif color_name == "Blue":
                    ser.write(b'3\n')
                elif color_name == "Purple":
                    ser.write(b'4\n')
                elif color_name == "Black":
                    ser.write(b'5\n')
                elif color_name == "Orange":
                    ser.write(b'6\n')
                filename = f"{color_name}.jpg"
                cv2.imwrite(filename, frame)
                print(f"Captured image: {filename}")
    cv2.imshow("frame", frame)
```

```
        if cv2.waitKey(1) == ord("q"):
        break
cap.release()
cv2.destroyAllWindows()
```

2. Chương trình code Arduino

```
#include <Adafruit_NeoPixel.h>
#define LED_COUNT 8
#define LED_PIN   6
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB +
NEO_KHZ800);
void setup() {
  Serial.begin(9600);
  strip.begin();
  strip.show();
  strip.setBrightness(30);
  uint32_t colors[] = {strip.Color(255, 0, 0), strip.Color(0,
255, 0), strip.Color(0, 0, 255)};
    for (int j = 0; j < 3; j++) {
    uint32_t color = colors[j];
      for (int i = 0; i < LED_COUNT; i++) {
      strip.setPixelColor(i, color);
    }
    strip.show();
```

```
    delay(1000);
    for (int i = 0; i < LED_COUNT; i++) {
      strip.setPixelColor(i, strip.Color(0, 0, 0));
    }
    strip.show();
    delay(500);
  }
}
void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    Serial.println(data);
    delay(100);
    if (data == "0") {
      setColor(strip.Color(255, 255, 0));
    }
    else if (data == "1") {
      setColor(strip.Color(0, 255, 0));
    } else if (data == "2") {
      setColor(strip.Color(255, 0, 0));
    } else if (data == "3") {
      setColor(strip.Color(0, 0, 255));
```

```
    } else if (data == "4") {
      setColor(strip.Color(128, 0, 128));
    } else if (data == "5") {
      setColor(strip.Color(255, 255, 255));
    } else if (data == "6") {
      setColor(strip.Color(255, 165, 0));
    }
  }
  delay(20);
}
void setColor(uint32_t color) {
  for (int i = 0; i < LED_COUNT; i++) {
    strip.setPixelColor(i, color);
  }
  strip.show();
}
```