ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

XÂY DỰNG HỆ THỐNG TỰ ĐỘNG PHÁT HIỆN VÀ PHÂN LOẠI ĐIỂM ĐEN TRÊN BỀ MẶT SẢN PHẨM

Người hướng dẫn: **TS. NGUYỄN THỊ KIM TRÚC**
Sinh viên thực hiện:
1. LÊ ĐỨC ANH – MSSV: 105200290 – LỚP: 20TDH1
2. PHÙNG QUỐC KHÁNH – MSSV: 105200497 – LỚP: 20TDHCLC4

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài:

XÂY DỰNG HỆ THỐNG TỰ ĐỘNG PHÁT HIỆN VÀ PHÂN LOẠI ĐIỂM ĐEN TRÊN BỀ MẶT SẢN PHẨM

Sinh viên thực hiện 1: LÊ ĐỨC ANH

Số thẻ SV: 105200290 Lớp: 20TDH1

Sinh viên thực hiện 2: PHÙNG QUỐC KHÁNH

Số thẻ SV: 105200497 Lớp: 20TDHCLC4

Trong xu thế phát triển mạnh mẽ của công nghiệp 4.0, các dây chuyền sản xuất hiện đại đang ngày càng ưu tiên áp dụng công nghệ tự động hóa và thị giác máy tính nhằm nâng cao hiệu suất, chất lượng sản phẩm cũng như giảm thiểu chi phí vận hành. Xuất phát từ thực tiễn sản xuất tại Công ty Luxshare ICT – Nghệ An, nơi sản xuất hàng loạt các sản phẩm nhựa vỏ ngoài cho thiết bị điện tử, nhóm sinh viên thực hiện đề tài đã phát hiện ra một bài toán nổi bật: kiểm tra và phát hiện lỗi "điểm đen" – một dạng khuyết tật ngoại quan phổ biến nhưng khó phát hiện, ảnh hưởng trực tiếp đến chất lượng thẩm mỹ và giá trị thương mại của sản phẩm.

Đề tài tập trung vào việc nghiên cứu, thiết kế và chế tạo một hệ thống tích hợp giữa phần cứng và phần mềm, có khả năng tự động hóa quy trình kiểm tra bề mặt sản phẩm theo tiêu chuẩn công nghiệp. Hệ thống bao gồm ba khối chính: (1) Cơ cấu phần cứng điều khiển bởi vi điều khiển ESP32 kết hợp động cơ servo và stepper; (2) Hệ thống camera thu nhận ảnh sản phẩm và giao tiếp với máy tính; (3) Phần mềm xử lý ảnh trên máy tính (viết bằng Python sử dụng OpenCV, rembg, PIL…) đảm nhiệm nhiệm vụ nhận diện, phân tích và đánh giá điểm đen trên ảnh.

Hệ thống có thể vận hành ở hai chế độ: Tự động (Auto) – cho phép kiểm tra toàn bộ 5 mặt của sản phẩm (trước, sau, trái, phải, trên) theo trình tự định sẵn mà không cần sự can thiệp của con người; và Thủ công (Manual) – cho phép người vận hành lựa chọn và kiểm tra từng mặt riêng biệt. Phần mềm giao diện người dùng (GUI) hỗ trợ trực quan quá trình vận hành, cho phép hiển thị ảnh, vùng cắt, kết quả phân tích và điều khiển hệ thống một cách linh hoạt.

Quá trình xử lý ảnh bao gồm các bước: chụp ảnh sản phẩm, tách nền bằng vùng ROI hoặc thư viện rembg, chuyển ảnh sang ảnh xám, thiết lập ngưỡng độ xám ($\geq 60\%$) và quét ảnh bằng cửa sổ trượt 3x3 pixel để xác định điểm đen có diện tích ≥ 0.02 mm². Kết quả phân tích được đánh dấu trực tiếp lên ảnh với đường viền, lưu trữ cùng kết quả kiểm định dưới dạng tệp ảnh và bảng log, phục vụ truy xuất và đánh giá chất lượng sản phẩm.

Kết quả thực nghiệm cho thấy hệ thống đạt được độ chính xác cao trong việc phát hiện điểm đen, tốc độ kiểm tra đáp ứng tiêu chuẩn thực tế (≤ 60 giây/sản phẩm), đồng thời đảm bảo tính ổn định và khả năng mở rộng. Với ưu điểm là chi phí thấp, tính linh hoạt cao, dễ tích hợp vào dây chuyền sản xuất sẵn có, đề tài hứa hẹn là một giải pháp hiệu quả và thực tiễn trong các hệ thống kiểm định chất lượng sản phẩm nhựa

Trong tương lai, nhóm thực hiện định hướng phát triển thêm các mô hình AI nhằm nâng cao khả năng nhận diện nhiều loại lỗi khác (như trầy xước, mất liệu), đồng thời tối ưu thời gian xử lý và khả năng học tập theo dữ liệu thực tế.

Phụ lục 02

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Lê Đức Anh	105200290	20TDH1	Kỹ thuật điều khiển và tự động hóa
2	Phùng Quốc Khánh	105200497	20TDHCLC4	Kỹ thuật điều khiển và tự động hóa

1. Tên đề tài đồ án:

Xây dựng hệ thống tự động phát hiện và phân loại điểm đen trên bề mặt sản phẩm .....

.....

2. *Đề tài thuộc diện: ☑Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

3. Các số liệu và dữ liệu ban đầu:

.....

.....

4. Nội dung các phần thuyết minh và tính toán:

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Lê Đức Anh	1. Lập nguyên lý hoạt động, lưu đồ thuật toán của hệ thống. 2. Thiết kế các thiết bị cần được sử dụng trong hệ thống. 3. Thiết kế cơ chế truyền thông, gửi lệnh điều khiển và nhận tín hiệu phản hồi của phần mềm máy tính với các thiết bị khác. 4. Thiết kế cơ chế phát hiện và khoanh vùng điểm đen đáp ứng với tiêu chuẩn đề ra. 5. Thiết kế giao diện hoạt động của hệ thống. 6. Thiết kế hệ thống phần cứng (khung đỡ, mạch điều khiển, cơ cấu xoay lật và cơ cấu nâng hạ camera). 7. Thiết kế chương trình nhận lệnh từ chương trình máy tính và điều khiển hệ thống phần cứng để xoay, lật sản phẩm và nâng, hạ camera. 8. Thực hiện kiểm tra tính chính xác và độ ổn định thực tế của hệ thống.
2	Phùng Quốc Khánh	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung

5. Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):

a. Phần chung:

TT	Họ tên sinh viên	Nội dung

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung

6. Họ tên ngườihướng dẫn:	*Phần/ Nội dung hướng dẫn:*
Võ Văn Quỳnh	1. Phương pháp xử lý bài toán xác định độ xám của điểm đen. 2. Phương pháp tính toán, đánh giá độ ổn định và chính xác của hệ thống. 3. Cơ chế kiểm tra diện tích của điểm đen. 4. Các hướng phát triển có thể được áp dụng trong tương lai

7. Ngày giao nhiệm vụ đồ án: 6/3/2023

8. Ngày hoàn thành đồ án: 26/6/2023

Đà Nẵng, ngày tháng 7 năm 2023

Trưởng Bộ môn Tự động hóa **Người hướng dẫn**

TS. Giáp Quang Huy

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Lê Đức Anh Phùng Quốc Khánh	Số thẻ SV: 105200290 105200497
Tên đề tài ĐATN: *" Xây dựng hệ thống tự động phát hiện và phân loại điểm đen trên bề mặt sản phẩm"*	
Họ tên người HD: TS. Nguyễn Thị Kim Trúc	Đơn vị: Trường Đại học Bách Khoa - ĐHĐN
Họ tên người HD: Võ Văn Quỳnh	Đơn vị: Công ty TNHH Luxshare ICT

Tuần	Ngày	Khối lượng		GVHD ký tên
		đã thực hiện (%)	tiếp tục thực hiện (%)	
1	17/02/2025	Nhận đề tài	Gặp người hướng dẫn định hướng đồ án	
2	24/02/2025	Nghiên cứu phân tích tính ứng dụng của đề tài	Phân tích phương hướng giải quyết bài toán	
3	03/03/2025	Xác định yêu cầu và bài toán đặt ra		
4	10/03/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN ϒ Không tiếp tục thực hiện ĐATN ϒ		
5	17/03/2025	Nghiên cứu thuật toán giải quyết bài toán nhận dạng điểm đen	Lên ý tưởng thiết kê mô hình phần cứng	
6	24/03/2025	Thu thập mẫu sản phẩm để tiến hành thử nghiệm tính khả thi của thuật toán	Đánh giá tính khả thi của ý tưởng	
7	31/03/2025	Lên phương án thiết kế hệ thống, quy trình công nghệ	Lên danh sách thiết bị cần sử dụng, và đặt mua các thiết bị	
8	07/04/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN ϒ Không tiếp tục thực hiện ĐATN ϒ		
9	14/04/2025	Tiến hành lắp ráp hoàn chỉnh phần cứng	Thiết kế LĐTT, Lập trình chương trình cho vi điều khiển ESP32	
10	21/04/2025	Tinh chỉnh bản thiết kế mô hình phần cứng	Hoàn tất toàn bộ phần cứng	

11	28/04/2025	Lên phương án thiết kế phần mềm máy tính để điều khiển hệ thống	Xây dựng lưu đồ thuật toán và Lập trình chương trình trên Pycharm	
12	05/05/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN ϒ Không tiếp tục thực hiện ĐATN ϒ		
13	12/05/2025	Tinh chỉnh toàn bộ hệ thống	Hoàn thiện hệ thống	
14	19/05/2025	Chạy thử nghiệm sản phẩm trong môi trường thực tế	Tiến hành đánh giá tính hiệu quả độ ổn định của mô hình	
15	26/05/2025	Hoàn thiện mô hình, tổng hợp tài liệu, hoàn thiện các phần trong báo cáo		

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trong quá trình thực tập tốt nghiệp tại Công ty Luxshare ICT – Nghệ An, chúng em đã có cơ hội tiếp cận thực tế với quy trình sản xuất các sản phẩm nhựa của công ty, đồng thời được tạo điều kiện tham gia hỗ trợ và giải quyết một số vấn đề thường gặp trong quá trình sản xuất. Một trong những vấn đề nổi bật mà chúng em nhận thấy là việc kiểm tra và loại bỏ các sản phẩm có lỗi điểm đen trên bề mặt – một dạng lỗi ngoại quan ảnh hưởng lớn đến chất lượng thẩm mỹ và yêu cầu kiểm định nghiêm ngặt trong sản xuất hàng loạt.

Trong suốt quá trình thực hiện đồ án, chúng em xin được gửi lời cảm ơn chân thành đến cô Nguyễn Thị Kim Trúc – giảng viên hướng dẫn tại Trường Đại học Bách khoa – Đại học Đà Nẵng, cùng anh Võ Văn Quỳnh và anh Phạm Xuân Cường – cán bộ hướng dẫn phía Công ty Luxshare ICT – Nghệ An, đã tận tình hỗ trợ, định hướng và tạo mọi điều kiện thuận lợi để chúng em hoàn thành tốt đề tài này.

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Chúng tôi xin cam đoan rằng toàn bộ nội dung của Đồ án Tốt nghiệp với đề tài: " Xây dựng hệ thống tự động phát hiện và phân loại điểm đen trên bề mặt sản phẩm " là kết quả nghiên cứu và thực hiện của chính tôi, dưới sự hướng dẫn của giảng viên TS. Nguyễn Thị Kim Trúc và cán bộ hướng dẫn tại doanh nghiệp anh Võ Văn Quỳnh.

Chúng tôi khẳng định không sao chép, sử dụng trái phép bất kỳ nội dung nào từ các công trình nghiên cứu, tài liệu, luận văn, đồ án, bài báo,... của người khác mà không trích dẫn rõ ràng và đúng quy định. Nếu phát hiện có sự sao chép, gian lận, hoặc vi phạm đạo đức học thuật trong quá trình thực hiện đồ án, tôi xin hoàn toàn chịu trách nhiệm trước nhà trường và pháp luật.

Chúng tôi cam kết thực hiện đồ án với tinh thần trung thực, nghiêm túc và tuân thủ đầy đủ các quy định về liêm chính học thuật của Trường Đại học Bách Khoa – Đại học Đà Nẵng (DUT)

Sinh viên thực hiện

{Chữ ký, họ và tên sinh viên}

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU:

…….....

…….....

…….....

…….....

…….....

…….....

CHỮ VIẾT TẮT:

…….....

…….....

…….....

…….....

Ghi chú:

- Ký hiệu: mỗi mục ký hiệu gồm ký hiệu và phần tên gọi, diễn giải ký hiệu.
- Cụm từ viết viết tắt là các chữ cái và các ký hiệu thay chữ được viết liền nhau, để thay cho một cụm từ có nghĩa, thường được lặp nhiều lần trong đồ án.

MỞ ĐẦU

Trong bối cảnh công nghiệp hóa và tự động hóa ngày càng phát triển, việc kiểm soát chất lượng sản phẩm đóng vai trò then chốt trong việc đảm bảo uy tín và năng lực cạnh tranh của doanh nghiệp. Tại xưởng sản xuất BU8 – SBU5 của Công ty Luxshare ICT – Nghệ An, quy trình sản xuất các bộ phận vỏ ngoài bằng nhựa cho hộp đựng tai nghe Bluetooth đòi hỏi tiêu chuẩn nghiêm ngặt về chất lượng bề mặt. Tuy nhiên, lỗi ngoại quan, đặc biệt là lỗi điểm đen, thường xuyên xuất hiện với tỷ lệ cao (chiếm khoảng 70% tổng số lỗi ngoại quan) và gây khó khăn trong việc kiểm soát chất lượng. Lỗi này không chỉ làm giảm tính thẩm mỹ của sản phẩm mà còn ảnh hưởng trực tiếp đến uy tín thương hiệu và yêu cầu của khách hàng. Hiện tại, công đoạn kiểm tra ngoại quan tại xưởng chủ yếu được thực hiện thủ công, dẫn đến tốn kém thời gian, nhân lực và tiềm ẩn nguy cơ bỏ sót lỗi do yếu tố con người.

Xuất phát từ thực tiễn trên, đồ án tốt nghiệp này được thực hiện với mục tiêu thiết kế và phát triển một hệ thống kiểm tra ngoại quan tự động dựa trên công nghệ thị giác máy tính, tập trung vào việc phát hiện và phân loại lỗi điểm đen trên bề mặt sản phẩm nhựa. Hệ thống nhằm thay thế một phần công đoạn kiểm tra thủ công, nâng cao độ chính xác, tăng tốc độ kiểm tra, đồng thời giảm thiểu chi phí vận hành và phụ thuộc vào nhân công. Đề tài không chỉ đáp ứng nhu cầu thực tế tại Công ty Luxshare ICT mà còn hướng đến khả năng ứng dụng rộng rãi trong các dây chuyền sản xuất công nghiệp khác.

Mục tiêu nghiên cứu: Mục tiêu chính của đề tài là xây dựng một hệ thống xử lý ảnh ứng dụng trong môi trường công nghiệp, có khả năng:

- Phát hiện chính xác lỗi điểm đen trên bề mặt sản phẩm với các đặc trưng về độ xám ($\geq 60\%$) và diện tích (≥ 0.02 mm^2).
- Phân loại sản phẩm đạt (OK) hoặc không đạt (NG) dựa trên tiêu chuẩn chất lượng.
- Tự động hóa quy trình kiểm tra, giảm thiểu thao tác thủ công, từ đó nâng cao năng suất và đảm bảo tính ổn định, nhất quán trong kiểm soát chất lượng.
- Cung cấp giao diện thân thiện, dễ sử dụng, hỗ trợ người vận hành theo dõi và quản lý kết quả kiểm tra.

Phạm vi nghiên cứu: Đề tài tập trung vào việc phát hiện và phân loại lỗi điểm đen – một loại lỗi ngoại quan phổ biến trên bề mặt các bộ phận vỏ ngoài bằng nhựa của hộp đựng tai nghe Bluetooth. Phạm vi nghiên cứu được giới hạn như sau:

- Đối tượng nghiên cứu: Các sản phẩm nhựa có kích thước 3x7 cm, với 5 mặt cần kiểm tra (Trước, Sau, Trái, Phải, Trên).
- Môi trường ứng dụng: Hệ thống được thiết kế để tích hợp vào dây chuyền sản xuất tự động hóa tại xưởng BU8 – SBU5 hoặc sử dụng độc lập như một thiết bị kiểm tra hỗ trợ.

- Công nghệ sử dụng: Áp dụng các kỹ thuật xử lý ảnh số, bao gồm lọc nhiễu, phân tích độ sáng, và phát hiện bất thường, kết hợp với phần cứng như camera công nghiệp và vi điều khiển ESP32.

Phương pháp nghiên cứu: Đề tài sử dụng phương pháp xử lý ảnh số dựa trên thư viện OpenCV, kết hợp với các kỹ thuật như:

- Lọc nhiễu và tách nền (sử dụng vùng cắt ROI hoặc thư viện rembg).
- Chuyển đổi ảnh sang không gian màu xám để phân tích độ sáng.
- Phân ngưỡng và phát hiện contour để xác định vị trí, kích thước của điểm đen. Hệ thống được kiểm nghiệm thông qua tập dữ liệu ảnh thực tế thu thập từ dây chuyền sản xuất, với các thí nghiệm đánh giá độ chính xác, độ lặp lại (Repeatability), và độ tái lập (Reproducibility) theo phương pháp GRR (Gage Repeatability and Reproducibility). Kết quả thực nghiệm được phân tích để xác minh khả năng ứng dụng của hệ thống trong môi trường sản xuất thực tế.

Ý nghĩa của đề tài: Hệ thống kiểm tra ngoại quan tự động mang lại các giá trị sau:

- Kinh tế: Giảm chi phí nhân công và thời gian kiểm tra, từ đó tối ưu hóa hiệu suất sản xuất.
- Kỹ thuật: Ứng dụng công nghệ thị giác máy tính hiện đại, góp phần thúc đẩy tự động hóa trong sản xuất công nghiệp.
- Thực tiễn: Đáp ứng nhu cầu kiểm soát chất lượng nghiêm ngặt tại Công ty Luxshare ICT và có tiềm năng mở rộng ứng dụng sang các ngành công nghiệp khác.
- Xã hội: Giảm khối lượng công việc thủ công, cải thiện điều kiện làm việc cho nhân viên và nâng cao chất lượng sản phẩm cuối cùng.

Cấu trúc đồ án: Nội dung đồ án được trình bày trong 6 chương với cấu trúc logic và chi tiết như sau:

- Chương 1: Tổng quan đề tài – Giới thiệu bối cảnh thực tế tại Công ty Luxshare ICT, bài toán kiểm tra ngoại quan, và các yêu cầu kỹ thuật của hệ thống.
- Chương 2: Phân tích hệ thống – Trình bày cơ sở lý thuyết, các phương pháp kiểm tra ngoại quan truyền thống và hiện đại, cũng như kiến trúc tổng thể của hệ thống.
- Chương 3: Thiết kế phần cứng và phần mềm – Mô tả chi tiết quá trình chọn lựa thiết bị phần cứng (camera, vi điều khiển, động cơ) và phát triển phần mềm xử lý ảnh.
- Chương 4: Thiết kế chương trình điều khiển hệ thống – Giải thích quy trình vận hành ở cả chế độ tự động và thủ công, cùng thuật toán xử lý ảnh để phát hiện điểm đen.
- Chương 5: Kết quả thực nghiệm – Phân tích kết quả kiểm tra thực tế, bao gồm độ chính xác, độ lặp lại, độ tái lập theo phương pháp GRR, và đánh giá hiệu quả của hệ thống.

- Chương 6: Kết luận và hướng phát triển – Tóm tắt những thành tựu đạt được, đánh giá tiềm năng ứng dụng, và đề xuất các cải tiến như tích hợp AI, IoT, hoặc tối ưu thuật toán.

Chương 1: TỔNG QUAN ĐỀ TÀI

1.1 Giới thiệu bài toán kiểm tra ngoại quan sản phẩm

Tại xưởng sản xuất BU8 – SBU5 của Công ty Luxshare ICT – Nghệ An, quy trình sản xuất các bộ phận vỏ ngoài bằng nhựa cho sản phẩm tai nghe Bluetooth đòi hỏi tiêu chuẩn cao về chất lượng bề mặt. Tuy nhiên, trong thực tế sản xuất, vẫn thường xuyên phát sinh nhiều loại lỗi ngoại quan như xước, mẻ, dập, thiếu liệu… Trong đó, lỗi điểm đen là một dạng lỗi đặc thù, xuất hiện với tần suất cao (chiếm tỷ lệ lỗi xấp xỉ 70% trong số tổng lỗi) và gây nhiều khó khăn trong việc kiểm soát chất lượng. Nguyên nhân của lỗi này thường đến từ các yếu tố như tạp chất trong nguyên liệu đầu vào, bụi bẩn môi trường, vụn kim loại từ khuôn và trục vít, hoặc nhựa cháy bị kẹt trong cơ cấu đúc. Đây là những yếu tố rất khó kiểm soát triệt để trong thời gian ngắn và dễ tái diễn trong các lô sản xuất liên tục.

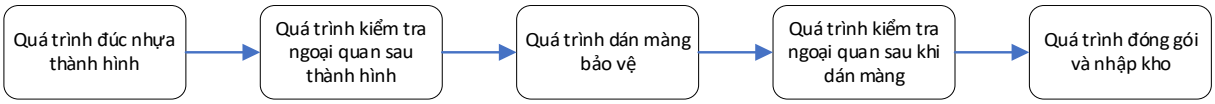
Do đặc thù khó phát hiện bằng máy móc truyền thống, các sản phẩm sau khi đúc luôn cần trải qua công đoạn kiểm tra ngoại quan thủ công nhằm phát hiện và loại bỏ những sản phẩm lỗi. Điều này đòi hỏi số lượng lớn nhân công, tiêu tốn nhiều thời gian và nguồn lực, đồng thời tiềm ẩn nguy cơ bỏ sót lỗi do yếu tố con người. Thực trạng này ảnh hưởng trực tiếp đến năng suất, chi phí sản xuất và tiến độ giao hàng.

Hình 1.1: Thao tác kiểm tra ngoại quan cho sản phẩm

Yêu cầu và đặt điểm của hệ thống kiểm tra ngoại quan:

- Điểm đen có độ xám ≥ 60%, diện tích đối với điểm đen cần bắt là 0.02mm2.
- Thời gian tiêu chuẩn để kiểm tra một sản phẩm: 60s.
- Sản phẩm có kích thước 3x7cm, có 5 mặt cần kiểm tra ngoại quan với tên các mặt là: Trước, Sau, Trái, Phải, Trên.
- Các mặt của sản phẩm đều được áp dụng cùng một tiêu chuẩn của điểm đen bao gồm cả về độ xám, vị trí và diện tích.

Quy trình sản xuất sản phẩm bao gồm các công đoạn cơ bản như sau:



Hình 1.2: Quy trình sản xuất sản phẩm cơ bản

Theo quy trình sản xuất như trên, hầu hết các lỗi điểm đen đều phải được phát hiện tại công đoạn kiểm tra ngoại quan sau thành hình. Bởi vì sau khi dán màng bảo vệ (film), việc kiểm tra ngoại quan và phát hiện điểm đen là rất khó. Vậy nên, cần phải cải thiện vấn đề chất lượng sản phẩm ngay tại bước kiểm tra ngoại quan sau thành hình này.

Hình 1.3: Sản phẩm sau khi dán màng bảo vệ

Trong bối cảnh đó, việc ứng dụng các giải pháp tự động hóa và thị giác máy tính để xây dựng hệ thống phát hiện và phân loại lỗi điểm đen là hết sức cần thiết. Không chỉ giúp giảm phụ thuộc vào nhân lực, một hệ thống kiểm tra tự động còn góp phần nâng cao độ chính xác, tăng tốc độ kiểm tra, đồng thời đảm bảo tính ổn định và nhất quán trong quá trình kiểm soát chất lượng sản phẩm. Đây chính là định hướng và mục tiêu cốt lõi của đề tài

1.2 Các phương án truyền thống và hiện đại trong kiểm tra ngoại quan

1.2.1 Phương pháp kiểm tra ngoại quan truyền thống

Trong các dây chuyền sản xuất công nghiệp, kiểm tra ngoại quan bằng mắt thường (hay còn gọi là kiểm tra thủ công) là phương pháp phổ biến lâu đời nhất. Nhân công sẽ quan sát trực tiếp từng sản phẩm để phát hiện các lỗi ngoại quan như xước, mẻ, thiếu liệu, và đặc biệt là điểm đen – một loại lỗi nhỏ nhưng ảnh hưởng nghiêm trọng đến tính thẩm mỹ. Phương pháp này có ưu điểm là dễ triển khai, không yêu cầu đầu tư thiết bị ban đầu, và có thể áp dụng linh hoạt trong nhiều trường hợp.

Tuy nhiên, phương pháp kiểm tra thủ công tồn tại nhiều hạn chế. Thứ nhất, nó phụ thuộc rất lớn vào yếu tố con người, dễ xảy ra sai sót do mỏi mắt, chủ quan, hoặc điều kiện ánh sáng không ổn định. Thứ hai, tốc độ kiểm tra chậm, không đáp ứng được yêu cầu sản lượng cao. Cuối cùng, việc duy trì số lượng lớn nhân công kiểm tra cũng làm tăng chi phí vận hành và khó kiểm soát chất lượng một cách nhất quán.

1.2.2 Các phương pháp kiểm tra hiện đại sử dụng công nghệ thị giác máy tính

Để khắc phục các nhược điểm của phương pháp thủ công, các hệ thống kiểm tra ngoại quan tự động sử dụng thị giác máy tính (Computer Vision) ngày càng được nghiên cứu và ứng dụng trong công nghiệp. Các hệ thống này sử dụng camera công nghiệp kết hợp với phần mềm xử lý ảnh nhằm phát hiện các dị thường trên bề mặt sản phẩm, trong đó có điểm đen.

Một số kỹ thuật xử lý ảnh phổ biến được sử dụng bao gồm: phân ngưỡng (thresholding), biến đổi không gian màu, phân tích histogram, lọc hình thái học (morphology), phát hiện biên (Canny, Sobel), cửa sổ trượt (sliding window), hoặc phân tích

contour. Các kỹ thuật này được xây dựng dựa trên các nguyên lý cơ bản của thị giác máy tính [1] và có thể được kết hợp để phát hiện các vùng có độ tương phản thấp nhưng khác biệt rõ với nền – đặc trưng của điểm đen.

1.2.3 Các phương pháp ứng dụng trí tuệ nhân tạo

Gần đây, với sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và học sâu (Deep Learning), các mô hình mạng nơ-ron tích chập (CNN) như YOLO, SSD, Faster R-CNN đã được ứng dụng để nhận dạng và phân loại lỗi ngoại quan, bao gồm cả điểm đen. Các mô hình này cho phép phát hiện tự động các loại lỗi với độ chính xác cao, khả năng thích ứng tốt với các biến thể hình dạng, vị trí và kích thước điểm đen trên sản phẩm.

Đặc biệt, các mô hình segmentation như U-Net, DeepLab, U²-Net có thể phát hiện chính xác vùng lỗi ở mức pixel, được phát triển dựa trên các nguyên lý học sâu được trình bày chi tiết trong nghiên cứu.[2] Những mô hình này giúp cải thiện hiệu quả trong việc phân loại và khoanh vùng lỗi nhỏ như điểm đen. Tuy nhiên, những hệ thống sử dụng AI yêu cầu lượng dữ liệu huấn luyện lớn, khả năng tính toán cao, và cần đầu tư ban đầu về thời gian và kỹ thuật.

1.2.4 So sánh giữa các phương pháp

Bảng 1.1: So sánh giữa các phương pháp kiểm tra ngoại quan

Tiêu chí	**Thủ công (truyền thống)**	**Thị giác máy tính (truyền thống)**	**AI / Deep Learning**
Độ chính xác	Thấp - Trung bình	Trung bình - Cao	Cao - Rất cao
Tốc độ	Thấp	Trung bình - Cao	Cao
Tính ổn định	Thấp (phụ thuộc con người)	Trung bình	Cao
Khả năng mở rộng	Khó	Trung bình	Tốt
Yêu cầu kỹ thuật ban đầu	Thấp	Trung bình	Cao (dữ liệu, GPU, lập trình)
Chi phí vận hành lâu dài	Cao	Trung bình	Thấp (sau khi triển khai)

1.3 Mục tiêu và yêu cầu hệ thống

Hệ thống được xây dựng trong khuôn khổ đề tài áp dụng phương pháp thị giác máy tính nhằm cải thiện độ chính xác, tốc độ xử lý, tính ổn định trong quá trình kiểm tra, đồng thời giảm thiểu độ phức tạp trong thiết kế và triển khai ban đầu. Hệ thống hướng đến khả năng hoạt động tự động, thân thiện với người sử dụng và dễ dàng tích hợp vào môi trường sản xuất thực tế.

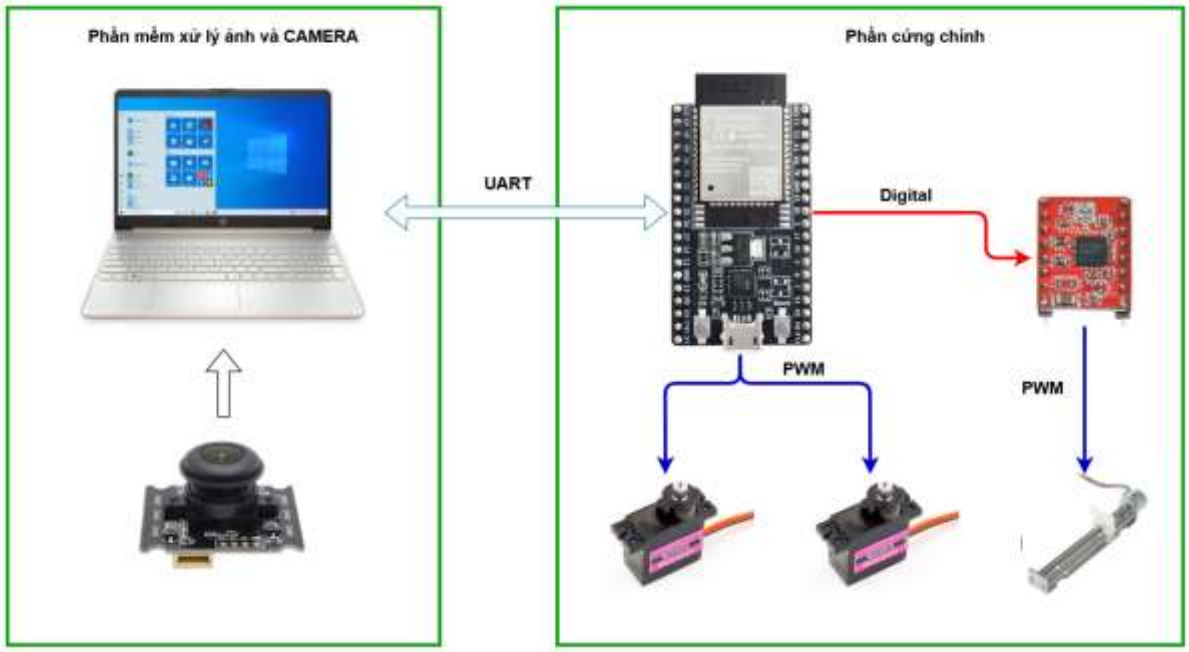
Hệ thống cần đáp ứng một số yêu cầu kỹ thuật quan trọng như sau:

- Có khả năng nhận diện chính xác điểm đen trên bề mặt sản phẩm với nhiều mức độ xám khác nhau.

- Có khả năng phân loại kết quả theo tiêu chí NG/OK dựa trên đặc điểm từng mặt kiểm tra, bao gồm số lượng điểm đen, diện tích vùng lỗi, cũng như mức độ xám tương đối.
- Cung cấp một giao diện trực quan, thân thiện, giúp người vận hành dễ dàng thao tác và theo dõi kết quả.
- Tự động hóa tối đa quy trình kiểm tra, giảm thiểu các bước thao tác thủ công không cần thiết nhằm tối ưu hóa thời gian xử lý và nâng cao hiệu suất toàn hệ thống.

Chương 2: PHÂN TÍCH HỆ THỐNG

2.1 Kiến trúc tổng thể của hệ thống



Hình 2.1: Kiến trúc tổng thể của hệ thống

Hệ thống trên là một giải pháp kiểm tra chất lượng sản phẩm dựa trên công nghệ xử lý ảnh, có khả năng nhận diện các khuyết tật (điểm đen) trên bề mặt sản phẩm. Hệ thống bao gồm hai thành phần chính: phần mềm xử lý ảnh và phần cứng điều khiển.

Phần mềm bao gồm một camera kết nối với máy tính, chịu trách nhiệm thu thập hình ảnh bề mặt sản phẩm. Hình ảnh được xử lý trên phần mềm chạy trên máy tính để phát hiện các điểm đen. Nếu phát hiện sản phẩm bị lỗi, phần mềm sẽ gửi tín hiệu điều khiển qua giao tiếp UART đến vi điều khiển ESP32.

Phần cứng sử dụng ESP32 làm thiết bị điều khiển trung tâm, nhận tín hiệu từ máy tính và điều khiển các thiết bị ngoại vi. Hai động cơ servo, kết nối qua giao tiếp PWM, được sử dụng để lật sản phẩm hoặc điều chỉnh cụm camera, giúp cải thiện độ chính xác khi lấy nét hoặc thay đổi góc nhìn khi cần thiết. Ngoài ra, ESP32 còn điều khiển một driver động cơ bước, hỗ trợ di chuyển sản phẩm đến vị trí kiểm tra chính xác hoặc hỗ trợ quá trình tự động lấy nét. Toàn bộ hệ thống hoạt động phối hợp nhịp nhàng để đảm bảo quy trình kiểm tra diễn ra tự động, chính xác và hiệu quả.

Bên cạnh đó, máy tính không chỉ đảm nhận vai trò xử lý ảnh mà còn cung cấp giao diện tương tác để điều khiển hệ thống, đồng thời thực hiện truyền thông. Máy tính có thể được xem như một server điều khiển trung tâm, với khả năng tích hợp cho các mô-đun tự động hóa khác cũng như tiềm năng phát triển thành một hệ thống IoT.

2.2 Mô tả chức năng từng thành phần

2.2.1 Camera

Camera đóng vai trò là thiết bị thu hình ảnh bề mặt sản phẩm. Hình ảnh này sẽ được gửi đến máy tính để xử lý. Camera có thể được cố định hoặc lắp trên cơ cấu tịnh tiến của step giúp camera có thể lấy nét bề mặt của sản phẩm. Chất lượng hình ảnh từ camera quyết định độ chính xác của quá trình phát hiện lỗi trên sản phẩm.

2.2.2 Máy tính

Máy tính chịu trách nhiệm xử lý ảnh thu được từ camera. Phần mềm xử lý ảnh được cài đặt trên máy tính (thường sử dụng các thư viện như OpenCV) sẽ phân tích hình ảnh để phát hiện các khuyết tật như điểm đen, vết xước, v.v. Sau khi phân tích, nếu phát hiện lỗi hoặc cần điều chỉnh, máy tính sẽ gửi lệnh điều khiển xuống vi điều khiển thông qua giao tiếp UART. Ngoài ra, máy tính còn có thể hiển thị kết quả kiểm tra và lưu trữ dữ liệu phục vụ phân tích sau này.

2.2.3 Vi điều khiển ESP32

ESP32 đóng vai trò trung tâm điều khiển phần cứng trong hệ thống. Nó nhận lệnh từ máy tính thông qua UART, sau đó phân tích và điều khiển các thiết bị ngoại vi như servo và driver động cơ bước. ESP32 giúp tự động hóa các thao tác như lật sản phẩm, điều chỉnh camera, hoặc di chuyển sản phẩm đến vị trí khác, đảm bảo hệ thống hoạt động trơn tru và đồng bộ.

2.2.4 Servo Motor

Các động cơ servo trong hệ thống được sử dụng để điều chỉnh góc quay của sản phẩm. Chúng có khả năng quay chính xác đến một góc nhất định, phù hợp cho các thao tác như lật sản phẩm sang mặt khác. Servo được điều khiển bằng tín hiệu PWM phát ra từ vi điều khiển ESP32.

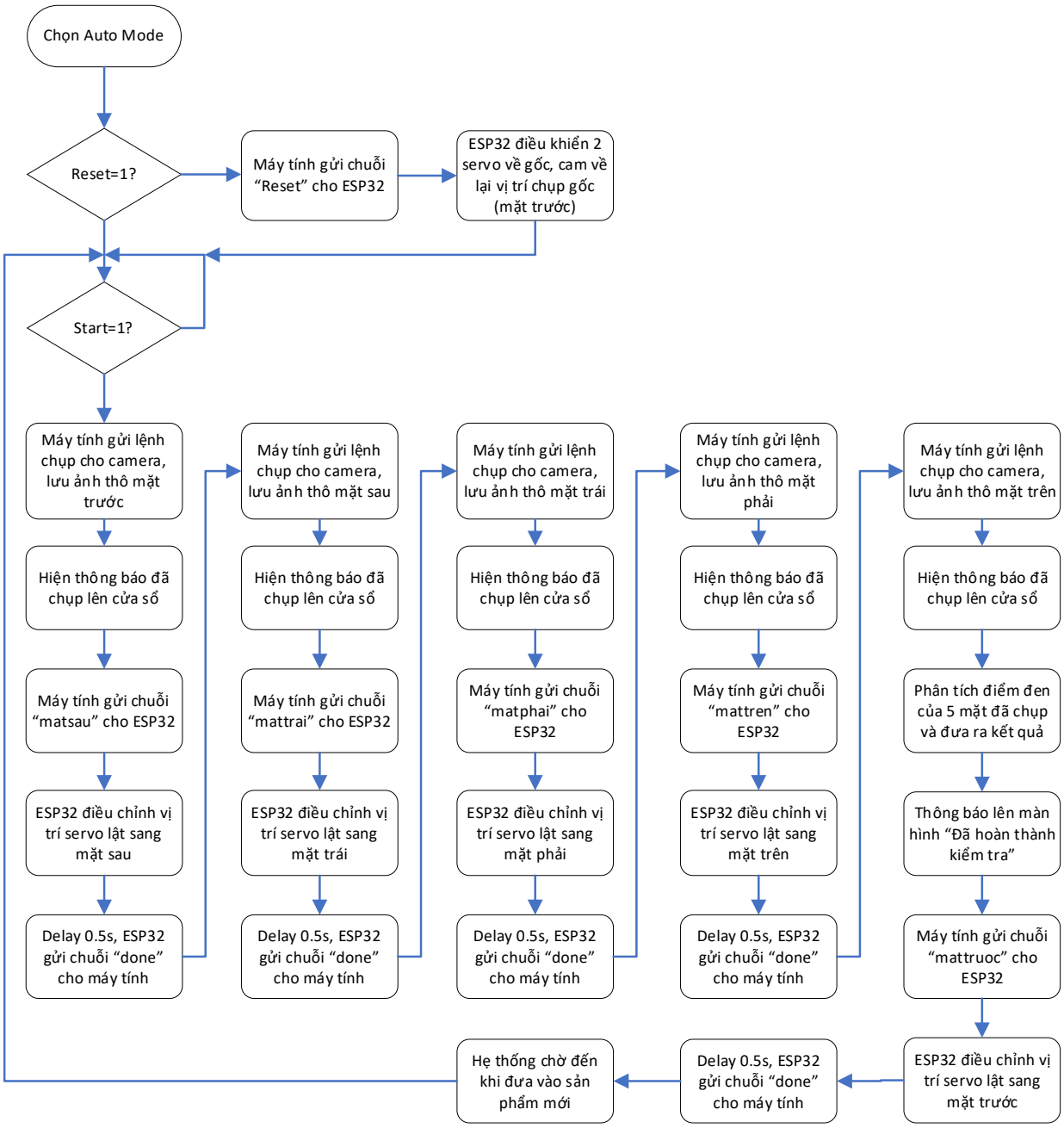
2.2.5 Driver điều khiển động cơ bước

Driver động cơ bước là cầu nối giữa vi điều khiển và động cơ bước. Nó nhận tín hiệu điều khiển từ ESP32 và tạo ra các xung PWM cần thiết để điều khiển chuyển động của động cơ bước. Nhờ có driver này, ESP32 không cần trực tiếp tạo xung chính xác, giúp giảm tải cho vi điều khiển.

2.2.6 Động cơ bước (Stepper Motor)

Động cơ bước được sử dụng để di chuyển sản phẩm hoặc cụm kiểm tra theo từng bước nhỏ, với độ chính xác cao. Trong hệ thống, động cơ bước có thể điều chỉnh vị trí của sản phẩm trên băng tải hoặc thay đổi khoảng cách giữa sản phẩm và camera để hỗ trợ quá trình lấy nét. Việc sử dụng động cơ bước giúp nâng cao tính tự động và độ chính xác của toàn bộ hệ thống kiểm tra.

2.3 Lưu đồ hoạt động chế độ Auto



Hình 2.2: Lưu đồ thuật toán chế độ Auto

Quy trình hoạt động tổng quát

Bước 1: Khởi động chế độ Auto Mode

Khi người dùng chọn chế độ Auto Mode, hệ thống sẽ bắt đầu kiểm tra trạng thái Reset. Nếu tín hiệu Reset = 1, máy tính gửi chuỗi "Reset" đến ESP32 để đưa toàn bộ hệ thống về trạng thái ban đầu. ESP32 thực hiện điều khiển hai servo quay về vị trí gốc – tức góc nhìn mặt trước của sản phẩm – để sẵn sàng bắt đầu kiểm tra.

Bước 2: Kích hoạt quá trình kiểm tra

Sau khi đã sẵn sàng, hệ thống kiểm tra tín hiệu Start. Nếu Start = 1, máy tính bắt đầu tiến trình kiểm tra lần lượt các mặt của sản phẩm.

2.3.1 Chi tiết từng bước kiểm tra

Mặt trước

- Máy tính gửi lệnh chụp ảnh mặt trước, ảnh được lưu lại.
- Hiển thị thông báo "đã chụp mặt trước" lên giao diện.
- Gửi chuỗi "matsau" đến ESP32.
- ESP32 điều khiển servo xoay sản phẩm sang mặt sau.
- Sau 0.5 giây, ESP32 gửi chuỗi "done" báo hoàn thành thao tác.

Mặt sau

- Máy tính gửi lệnh chụp ảnh mặt sau và lưu ảnh.
- Hiển thị thông báo đã chụp lên giao diện.
- Gửi chuỗi "matrai" đến ESP32.
- ESP32 điều khiển servo lật sang mặt trái.
- Sau 0.5 giây, gửi chuỗi "done" về máy tính.

Mặt trái

- Máy tính tiếp tục chụp ảnh mặt trái và lưu lại.
- Gửi chuỗi "matphai" để điều khiển ESP32 xoay sản phẩm sang mặt phải.
- ESP32 xử lý và phản hồi "done" sau khi hoàn tất chuyển động.

Mặt phải

- Máy tính chụp ảnh mặt phải, hiển thị thông báo.
- Gửi chuỗi "mattren" đến ESP32 để chuyển sang mặt trên.
- Sau khi servo xoay xong, ESP32 phản hồi "done".

Mặt trên

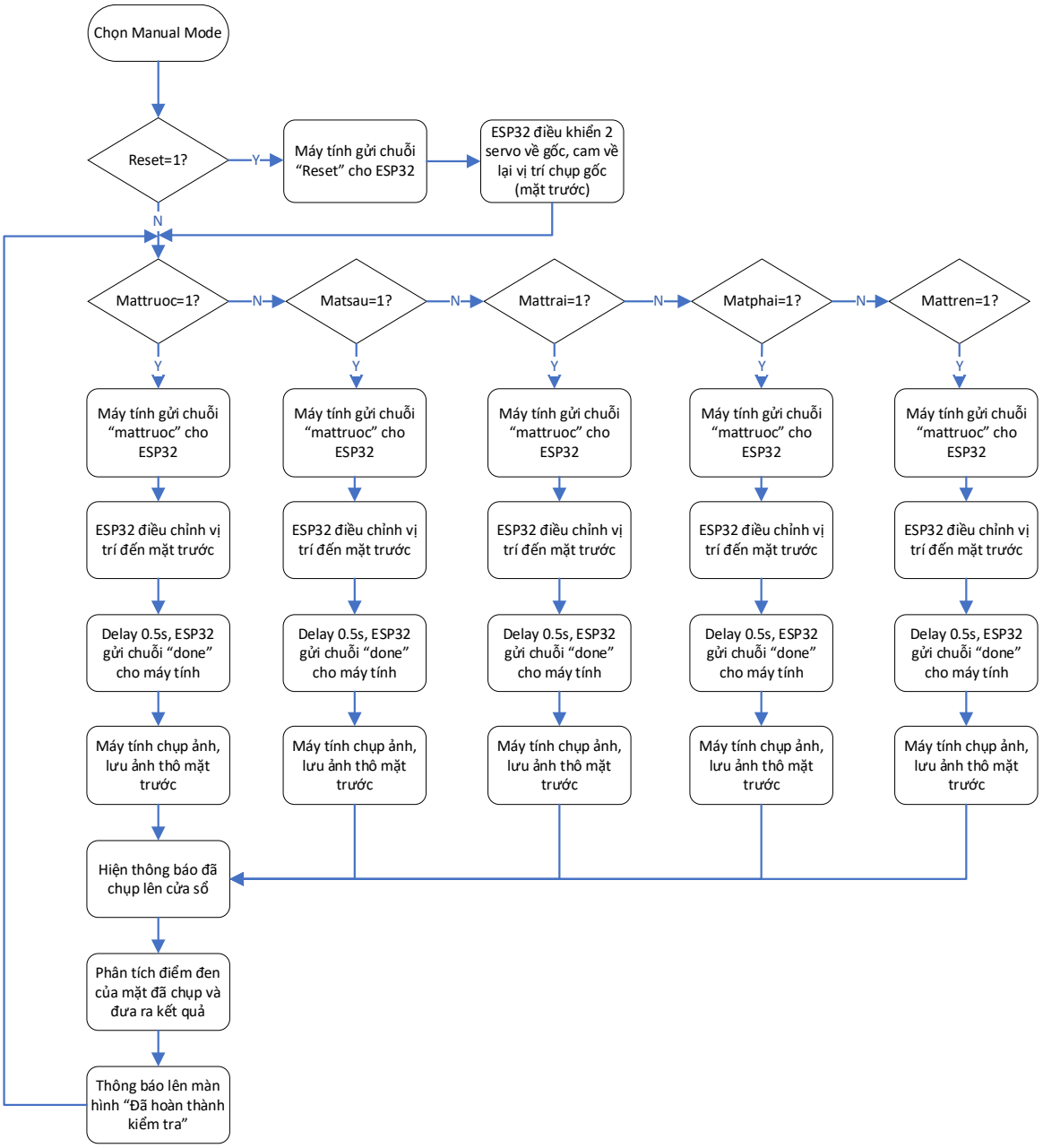
- Chụp ảnh mặt trên sản phẩm, lưu lại.
- Máy tính tiến hành phân tích hình ảnh 5 mặt vừa chụp để phát hiện điểm đen hoặc lỗi sản phẩm.

- Hiển thị kết quả phân tích và thông báo “Đã hoàn thành kiểm tra”.

2.3.2 Kết thúc chu trình và chuẩn bị sản phẩm mới

Sau khi hoàn tất kiểm tra, máy tính gửi chuỗi "mattruoc" để yêu cầu ESP32 đưa sản phẩm trở về vị trí ban đầu (mặt trước). Hệ thống lúc này chờ sản phẩm mới được đưa vào và bắt đầu chu trình kiểm tra tiếp theo.

2.4 Lưu đồ hoạt động chế độ Manual



Hình 2.3: Lưu đồ thuật toán chế độ MANUAL

2.4.1 Quy trình hoạt động tổng quát

Kích hoạt Manual Mode và kiểm tra Reset

Khi người dùng chọn chế độ Manual Mode, hệ thống đầu tiên sẽ kiểm tra biến Reset. Nếu Reset = 1, máy tính gửi chuỗi "Reset" đến ESP32 để đưa hai servo về vị trí ban đầu (góc mặt trước). Sau đó, hệ thống sẵn sàng nhận lệnh thủ công từ người dùng.

2.4.2 Quy trình kiểm tra thủ công từng mặt

Hệ thống lần lượt kiểm tra các tín hiệu lệnh điều khiển gửi từ người dùng:

Kiểm tra tín hiệu Mattruoc = 1 (Mặt trước)

- Máy tính gửi chuỗi "mattruoc" đến ESP32.
- ESP32 điều khiển servo quay đến mặt trước.
- Sau 0.5 giây, ESP32 gửi chuỗi "done" cho máy tính.
- Máy tính chụp ảnh, lưu ảnh thô mặt trước và hiển thị thông báo đã chụp.

Kiểm tra tín hiệu Matsau = 1 (Mặt sau)

- Máy tính gửi chuỗi "mattruoc" đến ESP32 để đưa về mặt trước trước khi chuyển mặt.
- ESP32 điều khiển servo về mặt trước, rồi quay sang mặt sau.
- Gửi chuỗi "done", sau đó chụp và lưu ảnh thô mặt sau.

Kiểm tra tín hiệu Mattrai = 1 (Mặt trái)

- Máy tính gửi chuỗi "mattruoc" đến ESP32 để đảm bảo quay về mặt chuẩn.
- ESP32 điều khiển servo về mặt trái và gửi "done".
- Máy tính chụp và lưu ảnh thô mặt trái.

Kiểm tra tín hiệu Matphai = 1 (Mặt phải)

- Máy tính gửi chuỗi "mattruoc" và ESP32 điều khiển về mặt phải.
- Sau "done", ảnh thô mặt phải được chụp và lưu lại.

Kiểm tra tín hiệu Mattren = 1 (Mặt trên)

- Máy tính gửi chuỗi "mattruoc" và ESP32 chuyển vị trí camera đến mặt trên.
- Sau "done", chụp và lưu ảnh mặt trên.

2.4.3 Phân tích và hiển thị kết quả

Sau khi chụp xong một mặt bất kỳ:

- Ảnh sẽ được hiển thị lên cửa sổ giao diện.
- Hệ thống tiến hành phân tích điểm đen trên mặt sản phẩm đã chụp.
- Kết quả phân tích được hiển thị trên màn hình với thông báo: "Đã hoàn thành kiểm tra".

Chương 3: THIẾT KẾ PHẦN CỨNG VÀ PHẦN MỀM

3.1 Thiết kế phần cứng

3.1.1 Chọn thiết bị

a. Camera OV3660 3MP

Hình 3.1: Camera OV3660

OV3660 là một cảm biến hình ảnh CMOS 3 megapixel (3MP) từ OmniVision, thường được tích hợp trong các module camera như M5Stack Timer Camera hoặc các module USB. Nó có độ phân giải tối đa 2048x1536 (QXGA), hỗ trợ đầu ra định dạng như RAW, RGB, YUV, và JPEG.

Thông số kỹ thuật:

- Độ phân giải: Tối đa 2048x1536 tại 15fps (có thể thay đổi tùy cấu hình).
- Góc nhìn (FOV): 110°.
- Kích thước pixel: 1.75µm x 1.75µm, sử dụng công nghệ OmniBSI cho độ nhạy cao.
- Giao tiếp: USB 2.0 (với module USB) hoặc giao tiếp song song DVP (với ESP32).
- Tính năng tự động: Điều chỉnh phơi sáng (AEC), cân bằng trắng (AWB), kiểm soát độ lợi (AGC), v.v.
- Ứng dụng: Phù hợp cho giám sát, nhận diện khuôn mặt, robot, hoặc hệ thống tự động hóa.

OV3660 có thể được sử dụng để thu thập hình ảnh bề mặt sản phẩm, phù hợp với vai trò camera trong phần mềm xử lý ảnh mà bạn mô tả. Khi kết hợp với ESP32, nó có thể truyền dữ liệu qua UART hoặc Wi-Fi (nếu được hỗ trợ).

b. ESP32 module dev kit

Hình 3.2: ESP32

Thông tin tổng quan: ESP32 là dòng vi điều khiển do Espressif Systems phát triển, tích hợp CPU lõi kép Xtensa LX6, bộ nhớ SRAM, và các giao thức không dây như Wi-Fi và Bluetooth. ESP32 thường được sử dụng trong IoT, tự động hóa, và các dự án robot.

Thông số kỹ thuật:

Bảng 3.1: Bảng thông số kỹ thuật ESP32

Hạng mục	Thông số kỹ thuật
Vi điều khiển chính	ESP32-WROOM-32 (tích hợp WiFi & Bluetooth)
CPU	Dual-core Xtensa® 32-bit LX6, xung nhịp tối đa 240 MHz
Bộ nhớ SRAM	520 KB
Bộ nhớ Flash	4 MB (tùy phiên bản)
Kết nối không dây	Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR và BLE
Số chân GPIO	Tối đa 34 chân I/O (đa chức năng)
ADC	18 kênh, độ phân giải 12-bit
DAC	2 kênh, độ phân giải 8-bit
PWM	Có hỗ trợ trên nhiều chân GPIO
UART	3 kênh UART
SPI	4 kênh SPI
I2C	2 kênh I2C
I2S	2 kênh I2S
CAN	Có hỗ trợ trên một số mô-đun

Hạng mục	Thông số kỹ thuật
Điện áp hoạt động	3.3V logic (nguồn cấp qua USB 5V)
Bộ ổn áp tích hợp	AMS1117 hoặc tương đương
Giao tiếp với máy tính	Micro USB (CP2102 hoặc CH340), dùng để nạp chươngtrình và UART
Kích thước bo mạch	Khoảng 51 mm x 25.5 mm
Nút tích hợp	Nút EN (Reset) và nút BOOT
Hệ điều hành hỗ trợ	Windows, macOS, Linux
Ngôn ngữ lập trình tương thích	Arduino IDE, ESP-IDF, MicroPython, PlatformIO

Ứng dụng trong hệ thống: ESP32 nhận tín hiệu từ máy tính qua UART, điều khiển hai động cơ servo (qua PWM) và driver động cơ bước (qua Digital hoặc PWM). ESP32 phù hợp để xử lý tín hiệu điều khiển và giao tiếp với các thiết bị ngoại vi, đồng thời có thể mở rộng để tích hợp thêm cảm biến hoặc truyền dữ liệu qua Wi-Fi trong trường hợp phát triển thêm để trở thành hệ thống IoT.

c. Servo MG90S

Hình 3.3: Servo MG90S

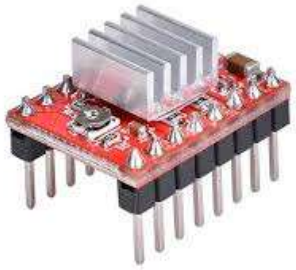
Thông tin tổng quan: MG90S là một servo micro với bánh răng kim loại, được nâng cấp từ SG90, cung cấp mô-men xoắn cao hơn và độ bền tốt hơn, phù hợp cho các ứng dụng như robot, máy bay RC, hoặc hệ thống tự động hóa.

Thông số kỹ thuật:

- Kích thước: ~23x12x29 mm
- Trọng lượng: ~13.4 g
- Điện áp: 4.8 – 6.0 V
- Momen xoắn: 1.8 – 2.2 kg·cm
- Tốc độ: ~0.1 giây/60°
- Góc quay: 180°
- Loại: servo mini, hộp số kim loại
- Dòng điện tối đa: ~700 mA

Ứng dụng trong hệ thống: Hai servo MG90S trong sơ đồ được kết nối qua PWM với ESP32, phù hợp để lật sản phẩm hoặc điều chỉnh góc camera. Mô-men xoắn 1.8-2.2 kg/cm đủ để xử lý các chuyển động nhẹ nhàng.

d. Module điều khiển động cơ Step A4988



Hình 3.4: Module điều khiển động cơ A4988

Thông số kỹ thuật:

- Điện áp logic (VDD): 3.3V – 5V
- Điện áp động cơ (VMOT): 8V – 35V
- Dòng tối đa: ~2A (với tản nhiệt tốt)
- Dòng khuyến nghị: ≤ 1A (không tản nhiệt)
- Bảo vệ quá nhiệt, quá dòng, ngắn mạch: Có
- Có chiết áp điều chỉnh dòng đầu ra
- Tương thích với động cơ bước loại: Bipolar (2 pha)

e. Bộ động cơ bước và hệ vitme

Hình 3.5: Bộ động cơ bước và hệ vitme

Thông số kỹ thuật:

- Loại động cơ: Động cơ bước 2 pha 4 dây
- Điện áp hoạt động: DC 4 – 9V
- Dòng tiêu thụ: 500 mA
- Góc bước: 18°

- Tốc độ tối đa: ~25 mm/s
- Bước ren trục vít: 0.5 mm (mỗi bước di chuyển 0.025 mm)
- Chiều dài trục vít: 90 mm
- Đường kính trục vít: 3 mm
- Chiều rộng thanh trượt: 15 mm
- Hành trình hiệu quả: 80 mm

f. Cảm biến khe quang FC-03

Hình 3.6: Cảm biến khe quang FC-03

Thông số kỹ thuật:

- Điện áp hoạt động
- 3.3V - 5V DC
- Dòng điện tiêu thụ
- Khoảng 20mA
- Cảm biến chính
- Khe quang hồng ngoại (infrared slot sensor)
- Khoảng cách phát hiện
- Vật cản đi qua khe (khoảng cách khe ~5mm)
- Giao tiếp đầu ra
- Digital (DO) – mức cao/thấp tùy theo có vật cản hay không
- IC chính trên mạch
- LM393 (Comparator để so sánh và khuếch đại tín hiệu từ cảm biến)

g. Đèn LED vòng

Hình 3.7: Đèn LED vòng

Thông số kỹ thuật:

- Số lượng LED: ~56–144 LED
- Nguồn cấp: AC 100–240V (kèm adaptor EU plug)
- Công suất: 4–10W
- Ánh sáng: Trắng (nhiệt độ màu khoảng 5500–6500K)
- Đường kính trong: ~60 mm
- Điều chỉnh độ sáng: Có (nút vặn dimmer)

3.1.2 Các cơ cấu cơ khí chính của hệ thống

a. Cơ cấu điều chỉnh góc chụp

Cơ cấu điều chỉnh góc chụp sản phẩm bao gồm 2 động cơ servo và một jig cố định sản phẩm. 2 động cơ này được gắn với nhau thông qua một khung nhựa hình chữ L nhằm tạo ra sự vuông góc về trục, cắt qua nhau của cả 2 động cơ. Kết quả của thiết kế này sẽ tạo ra một cơ cấu xoay, lật 360° với mỗi bước xung là 18°.

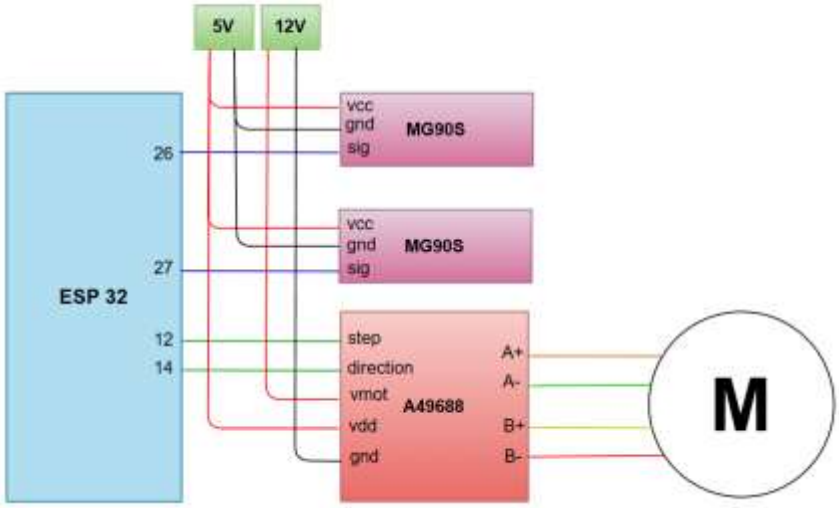
Hình 3.8: Cơ cấu xoay, lật sản phẩm

b. Cơ cấu điều chỉnh độ cao camera

Cơ cấu điều chỉnh độ cao camera bao gồm một tấm phẳng hình chữ L được cố định vào bộ động cơ bước và hệ vitme, camera được gắn vào mặt dưới của tấm chữ L. Ngay bên cạnh của tấm chữ L, một bộ cảm biến quang nhằm xác định gốc cho cơ cấu cũng được lắp đặt, đảm bảo sự chính xác của hệ thống khi cần hiệu chỉnh định kỳ hoặc khi có sự cố cần thiết lập lại.

Hình 3.9: Cơ cấu điều chỉnh độ cao camera

3.1.3 Sơ đồ kết nối phần cứng



Hình 3.10: Sơ đồ kết nối phần cứng

ESP32:

- Chân GPIO 26 → tín hiệu điều khiển cho servo MG90S thứ nhất
- Chân GPIO 27 → tín hiệu điều khiển cho servo MG90S thứ hai
- Chân GPIO 12 → chân step của A4988
- Chân GPIO 14 → chân direction của A4988

Nguồn:

- 5V: cung cấp cho 2 servo MG90S và chân VDD của A4988
- 12V: cấp cho chân vmot của A4988 để điều khiển động cơ bước

Servo MG90S (x2):

- Mỗi servo có 3 chân:
  - VCC nối 5V
  - GND nối GND chung
  - SIG nối GPIO 26 hoặc 27 từ ESP32

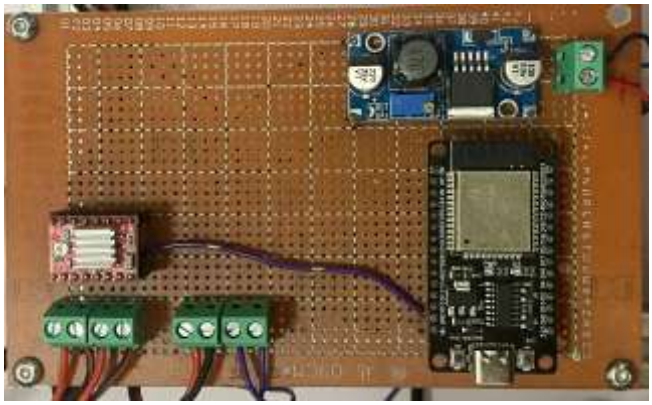
Driver A4988:

- Step và direction nhận tín hiệu từ ESP32
- VĐ cấp 5V logic
- VMOT cấp 12V để chạy động cơ
- GND nối GND chung
- Kết nối đến động cơ bước 2 pha:
  - A+, A−
  - B+, B−

3.1.4 Mạch điều khiển động cơ và hệ thống phần cứng hoàn thiện

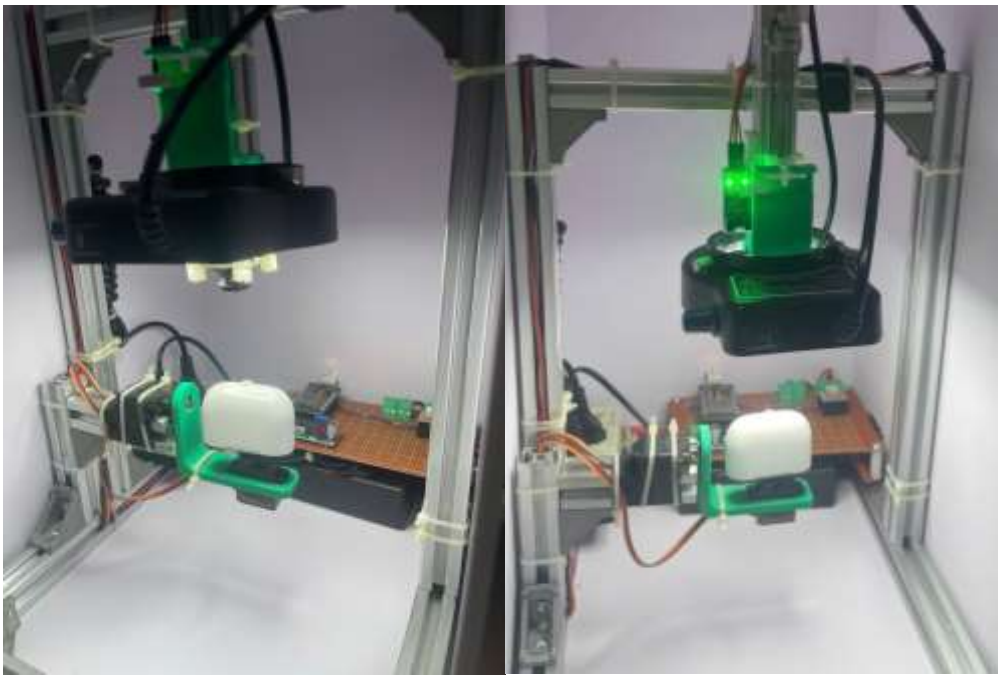
a. Mạch điều khiển động cơ

Từ sơ đồ kết nối phần cứng, thực hiện thiết kế mạch điều khiển động cơ bao gồm ESP32, nguồn cấp cho động cơ bước và driver cho động cơ bước.



Hình 3.11: Mạch điều khiển động cơ

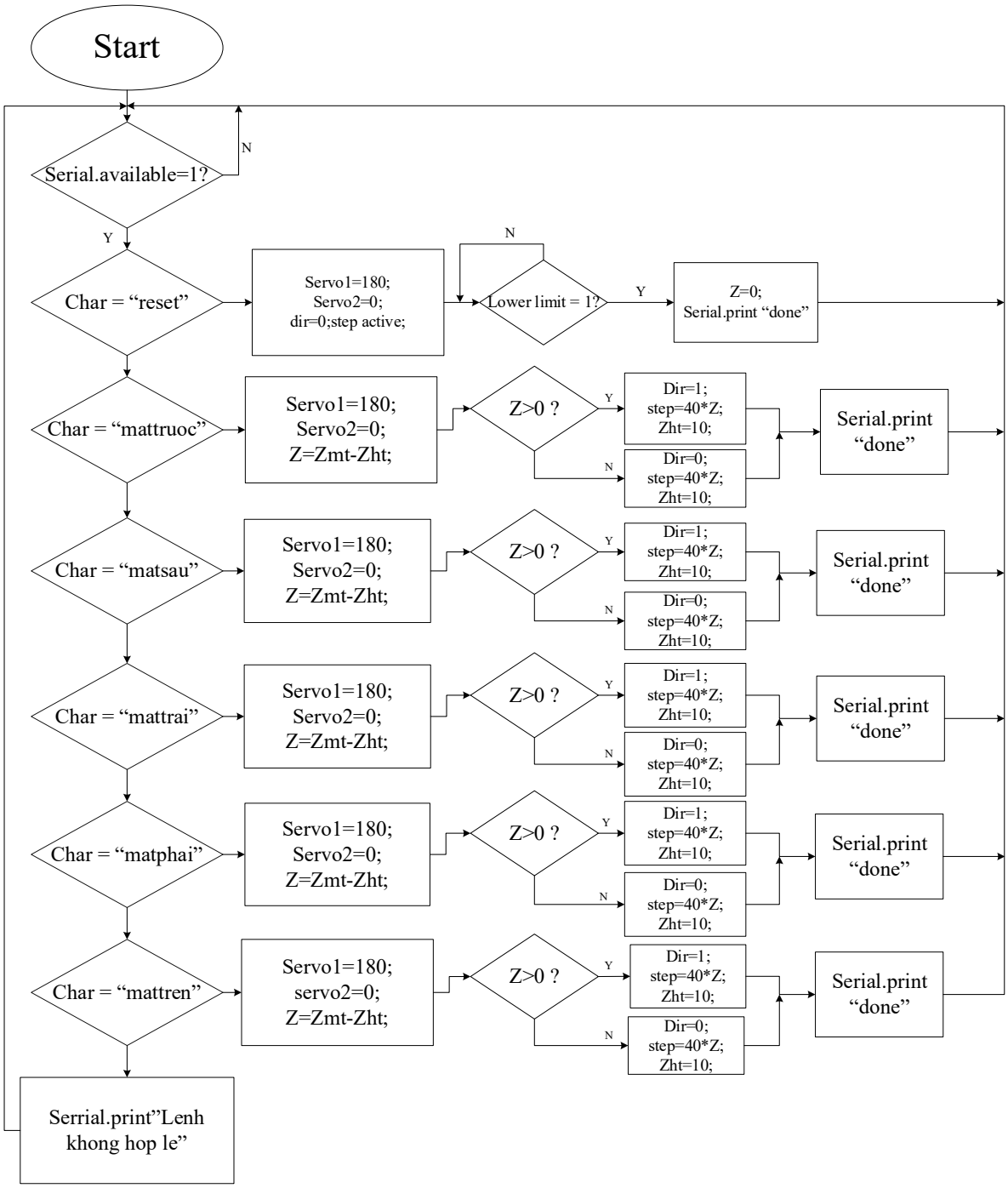
b. Hệ thống phần cứng hoàn thiện



Hình 3.12: Hệ thống phần cứng hoàn thiện

3.2 Thiết kế phần mềm trên ESP32

3.2.1 Lưu đồ thuật toán:



Hình 3.13: Lưu đồ thuật toán ESP32

3.2.2 Giải thích tổng quan

- **Mục đích**: Thuật toán điều khiển servo để xoay sản phẩm qua các vị trí tương ứng với các mặt (trước, sau, trái, phải, trên) dựa trên các lệnh nhận được từ giao diện phần

mềm (ví dụ: reset, mattruoc, matsau, v.v.). Sau khi hoàn thành mỗi bước xoay, ESP32 gửi phản hồi "done" để thông báo cho phần mềm.

- **Các thành phần chính**:
  - **Servo1 và Servo2**: Hai servo hoặc động cơ bước được sử dụng để điều khiển chuyển động (có thể đại diện cho hai trục quay).
  - **Z=Zmt-Zht**: Biến Z đại diện cho vị trí hiện tại, được so sánh với ngưỡng Zmt (vị trí mục tiêu) và Zht (bước di chuyển).
  - **dir**: Hướng di chuyển (1 cho đi lên, 0 cho đi xuống).
  - **Char**: Lệnh nhận được từ phần mềm (ví dụ: reset, mattruoc, v.v.).
- **Luồng hoạt động**: Bắt đầu từ trạng thái "Start", hệ thống kiểm tra tính sẵn sàng của cổng nối tiếp (Serial.available), sau đó xử lý các lệnh và điều khiển servo dựa trên giá trị của Char.

3.2.3 Phân tích chi tiết luồng hoạt động

a. Khởi đầu (Start)
   - **Điều kiện**: Kiểm tra Serial.available == 1?
     - Nếu Không (N): Hệ thống chờ dữ liệu từ cổng nối tiếp (không có lệnh mới).
     - Nếu Có (Y): Chuyển sang đọc lệnh Char.

b. Đọc lệnh (Char)
   - Hệ thống đọc ký tự lệnh từ cổng nối tiếp (Serial.read()). Các lệnh bao gồm:
     - reset
     - mattruoc (mặt trước)
     - matsau (mặt sau)
     - mattrai (mặt trái)
     - matphai (mặt phải)
     - mattren (mặt trên)
   - Mỗi lệnh tương ứng với một vị trí mục tiêu (Zmt) và chuỗi điều khiển servo.

c. Xử lý lệnh reset
   - Thao tác:
     - Đặt Servo1 = 180, Servo2 = 0.
     - Đặt dir = 0 (hướng đi xuống), step = active (bắt đầu di chuyển).
   - **Mục đích**: Đưa hệ thống về trạng thái ban đầu (vị trí gốc) bằng cách đặt servo về góc 180° và 0°.
   - **Kiểm tra**: So sánh Z = Zmt - Zht với ngưỡng Z > 0?

- Nếu Có (Y): Giảm Zht (di chuyển xuống), lặp lại kiểm tra.
- Nếu Không (N): Gửi "done" qua cổng nối tiếp (Serial.print("done")) để báo hiệu hoàn tất.

d. Xử lý lệnh mattruoc (mặt trước)

- Thao tác:
  - Đặt Servo1 = 180, Servo2 = 0.
  - Đặt dir = 0 (hướng đi xuống), step = 40, Zht = 10.
- **Mục đích**: Điều chỉnh servo để xoay sản phẩm đến mặt trước, với step = 40 là bước di chuyển và Zht = 10 là khoảng cách di chuyển nhỏ.
- **Kiểm tra**: So sánh Z = Zmt - Zht với ngưỡng Z > 0?
  - Nếu Có (Y): Giảm Zht, lặp lại kiểm tra.
  - Nếu Không (N): Gửi "done".

e. Xử lý lệnh matsau (mặt sau)

- Thao tác:
  - Đặt Servo1 = 180, Servo2 = 0.
  - Đặt dir = 0, step = 40, Zht = 10.
- **Mục đích**: Tương tự như mattruoc, nhưng điều chỉnh để xoay đến mặt sau. Các giá trị step và Zht có thể được điều chỉnh để phù hợp với vị trí cụ thể.
- **Kiểm tra**: So sánh Z > 0? và lặp lại nếu cần, sau đó gửi "done".

f. Xử lý lệnh mattrai (mặt trái)

- Thao tác:
  - Đặt Servo1 = 180, Servo2 = 0.
  - Đặt dir = 0, step = 40, Zht = 10.
- **Mục đích**: Xoay sản phẩm đến mặt trái với các thông số điều khiển tương tự.
- **Kiểm tra**: So sánh Z > 0?, gửi "done" khi hoàn tất.

g. Xử lý lệnh matphai (mặt phải)

- Thao tác:
  - Đặt Servo1 = 180, Servo2 = 0.
  - Đặt dir = 0, step = 40, Zht = 10.
- **Mục đích**: Điều chỉnh servo để đưa sản phẩm đến mặt phải.
- **Kiểm tra**: So sánh Z > 0?, gửi "done" khi xong.

h. Xử lý lệnh mattren (mặt trên)

- Thao tác:

- Đặt Servo1 = 180, Servo2 = 0.
- Đặt dir = 0, step = 40, Zht = 10.

- **Mục đích**: Xoay sản phẩm đến mặt trên với các thông số điều khiển tương tự.
- **Kiểm tra**: So sánh Z > 0?, gửi "done" khi hoàn tất.

i. Kết thúc (Serial.print "Lenh khong hop le")

- **Điều kiện**: Nếu lệnh Char không khớp với bất kỳ lệnh nào trên, hệ thống gửi thông báo "Lenh khong hop le" (lệnh không hợp lệ).
- **Mục đích**: Báo hiệu lỗi hoặc lệnh không được nhận diện, giúp người dùng hoặc phần mềm điều chỉnh.

Giải thích về dir (hướng di chuyển)

- **dir = 1 (đi lên)**: Hướng di chuyển tăng dần theo giá trị Z (chưa thấy trong luồng đồ, có thể được sử dụng trong các trường hợp khác).
- **dir = 0 (đi xuống)**: Hướng di chuyển giảm dần theo giá trị Z, được sử dụng trong tất cả các lệnh trong luồng đồ này. Điều này cho thấy hệ thống được thiết kế để xoay sản phẩm từ một vị trí cao về vị trí thấp hơn (hoặc ngược lại tùy cấu hình servo).

3.2.4 Cơ chế hoạt động tổng quát

a. **Khởi động**: Hệ thống kiểm tra kết nối cổng nối tiếp.
b. **Nhận lệnh**: Đọc lệnh từ phần mềm (qua Serial.read()).
c. **Điều khiển servo**: Dựa trên lệnh, đặt góc cho Servo1 và Servo2, xác định hướng dir và bước di chuyển step.
d. **Điều chỉnh vị trí**: So sánh Z = Zmt - Zht với ngưỡng Z > 0 để điều chỉnh dần vị trí, với Zht giảm dần cho đến khi đạt mục tiêu.
e. **Hoàn tất**: Khi Z <= 0, gửi "done" để thông báo hoàn thành.

3.3 Thiết kế phần mềm trên máy tính

3.3.1 Tổng quan các chức năng chính

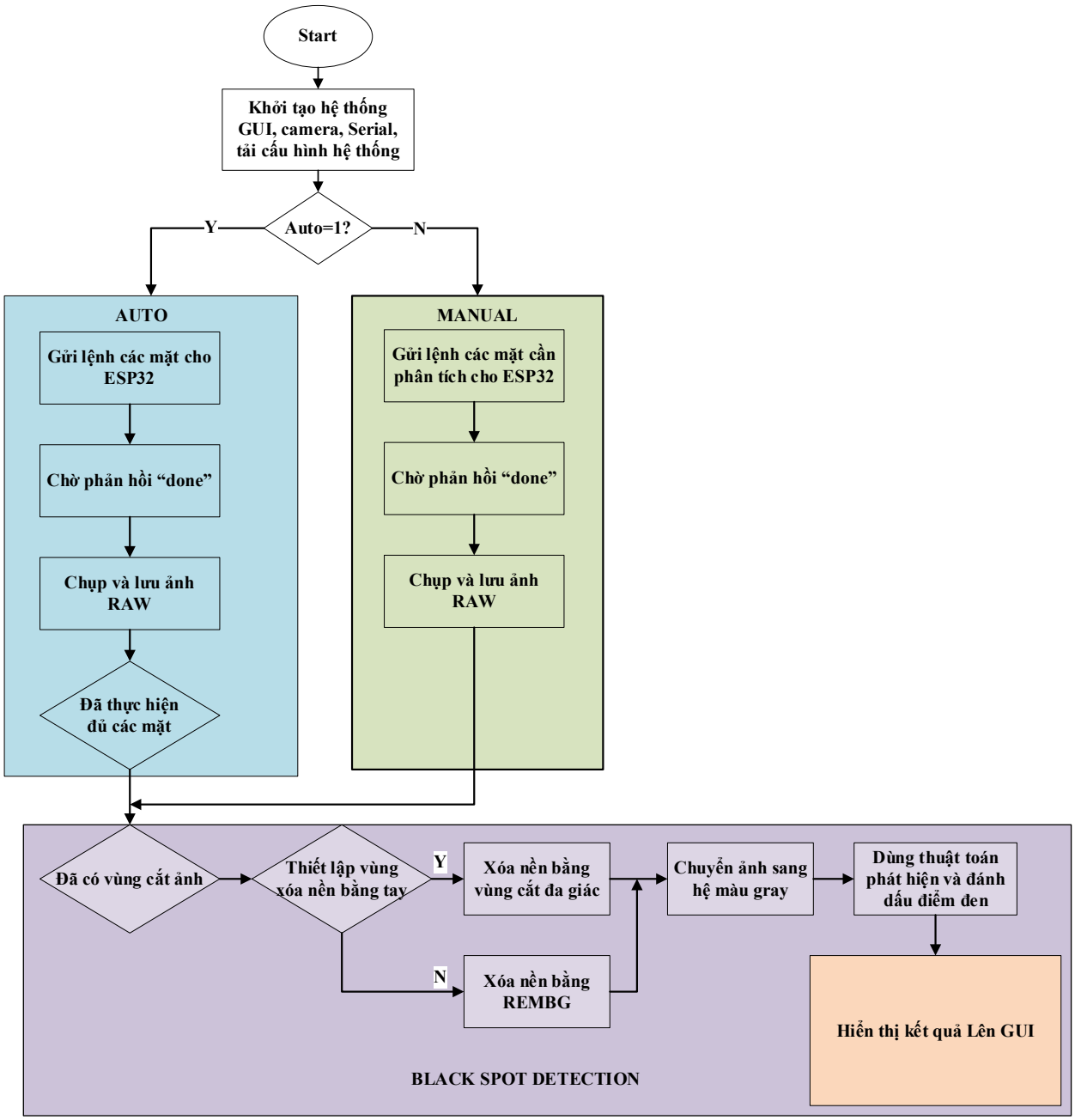
Dưới đây là **mô tả tổng quan hệ thống dưới dạng các module chính**, giúp làm rõ cấu trúc phần mềm và vai trò của từng thành phần:

Hình 3.14: Sơ đồ mô tả các chức năng chính của phần mềm

Bảng 3.2: Các chức năng chính của phần mềm

STT	Tên Module	Chức năng chính	Thư viện sử dụng	Tính năng/Chi tiết nổi bật
1	Giao tiếp phần cứng (Serial Communication)	Gửi lệnh điều khiển mô tơ quay đến ESP32, nhận phản hồi xác nhận (done).	serial	Retry tối đa 2 lần khi không nhận phản hồi
2	Chụp ảnh và Lưu ảnh (Image Capture)	Chụp ảnh từng mặt sản phẩm, lưu ảnh gốc (RawData/) và ảnh xử lý (PicData/).	OpenCV, datetime, os	Độ phân giải ảnh 2048 x 1536 px
3	Phân tích ảnh & phát hiện điểm đen	Cắt ảnh theo vùng đa giác hoặc tách nền tự động, chuyển ảnh sang grayscale, phát hiện điểm đen.	OpenCV, numpy, rembg, PIL	Vẽ contour điểm đen lên ảnh kết quả
4	Giao diện người dùng (GUI)	Hiển thị giao diện điều khiển, chọn mặt sản phẩm, chế độ Auto/Manual, xem ảnh kết quả, thiết lập vùng cắt	tkinter, PIL.ImageTk	Khung trái (điều khiển), giữa (ảnh), phải (kết quả)
5	Quản lý vùng cắt (Crop Configuration)	Vẽ vùng cắt đa giác trên ảnh, lưu tọa độ vùng cắt vào file JSON, reset vùng cắt mặc định	cv2, json, filedialog	Hỗ trợ nhập số cạnh, kéo thả điểm vùng cắt bằng chuột
6	Quản lý chế độ hoạt động (Operation Mode)	Chuyển đổi giữa chế độ Auto và Manual, đảm bảo chỉ 1 chế độ hoạt động, kiểm soát logic thao tác		Kiểm soát điều kiện logic trước thao tác

3.3.2 Lưu đồ thuật toán



Hình 3.15:Lưu đồ thuật toán phần mềm

3.3.3 Phân tích luồng đồ thuật toán

Luồng đồ thuật toán này mô tả quy trình hoạt động của hệ thống kiểm tra bề mặt sản phẩm bằng hình ảnh, bao gồm cả chế độ Tự động (Auto) và Thủ công (Manual). Dưới đây là phân tích chi tiết từng bước, được trình bày rõ ràng và kết luận cuối cùng.

a. Khởi đầu (Start)

- **Mô tả**: Hệ thống bắt đầu bằng việc khởi tạo các thành phần chính, bao gồm GUI (giao diện người dùng), camera, cổng nối tiếp (Serial), và thiết lập môi trường hoạt động (tải cấu hình, kết nối phần cứng).
- **Mục đích**: Đảm bảo tất cả các thiết bị (camera, ESP32) sẵn sàng và kết nối ổn định trước khi thực hiện quy trình kiểm tra.
- **Điều kiện**: Chuyển sang bước tiếp theo để kiểm tra chế độ vận hành.

b. Kiểm tra chế độ (Auto = 1?)

- **Mô tả**: Hệ thống kiểm tra xem chế độ được chọn là Tự động (Auto = 1) hay Thủ công (Auto = 0).
  - Nếu **Có (Y)**: Chuyển sang nhánh **AUTO**.
  - Nếu **Không (N)**: Chuyển sang nhánh **MANUAL**.
- **Mục đích**: Phân nhánh quy trình dựa trên lựa chọn của người dùng, đảm bảo hệ thống hoạt động theo chế độ phù hợp (tự động hóa hoặc điều khiển thủ công).
- **Ý nghĩa**: Cung cấp tính linh hoạt, cho phép hệ thống thích nghi với các kịch bản sử dụng khác nhau.

c. Nhánh AUTO (Chế độ Tự động)

- Mô tả:
  - **Gửi lệnh tự động đến ESP32**: Phần mềm tự động gửi các lệnh điều khiển (ví dụ: mattruoc, reset) đến ESP32 để xoay sản phẩm qua các mặt.
  - **Chờ phản hồi "done"**: ESP32 thực hiện lệnh và gửi phản hồi "done" khi hoàn tất.
- **Mục đích**: Tự động hóa toàn bộ quy trình xoay và chụp ảnh, giảm sự can thiệp của con người, phù hợp với sản xuất hàng loạt.
- Tiếp theo: Chuyển sang bước chụp ảnh.

d. Nhánh MANUAL (Chế độ Thủ công)

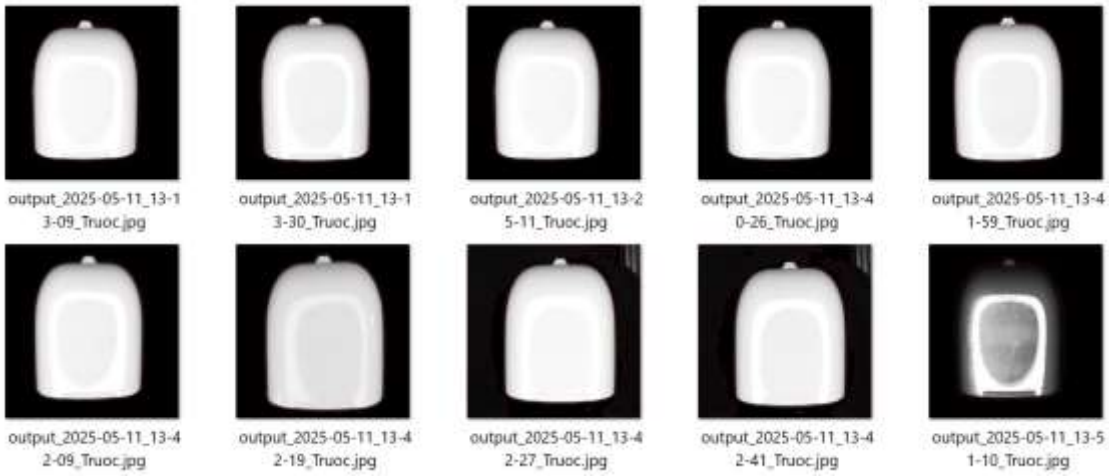
- Mô tả:
  - **Gửi lệnh thủ công đến ESP32**: Người dùng chọn mặt cần kiểm tra qua GUI và gửi lệnh tương ứng đến ESP32.
  - **Chờ phản hồi "done"**: ESP32 xoay sản phẩm đến vị trí yêu cầu và gửi phản hồi "done".
- **Mục đích**: Cho phép người dùng kiểm soát thủ công từng bước, phù hợp cho kiểm tra chi tiết hoặc kiểm tra lại.

- **Tiếp theo**: Chuyển sang bước chụp ảnh, tương tự nhánh AUTO.

e. Chụp ảnh và lưu ảnh (Chụp ảnh từ camera RAW)

- **Mô tả**: Sau khi nhận "done", hệ thống sử dụng camera để chụp ảnh mặt sản phẩm và lưu vào thư mục RAW (RawData/).

Hình 3.16: Lưu trữ hình ảnh thô

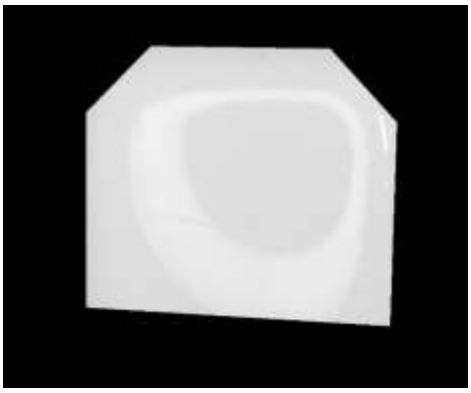


Hình 3.17: Lưu trữ hình ảnh đã qua xử lý

- **Mục đích**: Ghi nhận dữ liệu hình ảnh thô làm cơ sở cho quá trình xử lý, đảm bảo dữ liệu đầu vào chính xác và đầy đủ.
- **Điều kiện**: Kiểm tra Đã xử lý chưa?
  - Nếu **Có (Y)**: Bỏ qua bước xử lý nền, chuyển sang xử lý tiếp theo.
  - Nếu **Không (N)**: Tiến hành xử lý nền.

f. Xử lý nền (Xóa nền bằng vùng cắt hoặc REMBG)

- Mô tả:
  - **Nếu có vùng cắt (ROI)**: Sử dụng vùng cắt đã định nghĩa để tạo mặt nạ (mask) loại bỏ nền.



Hình 3.18: Hình ảnh đã xử lý khi áp dụng vùng cắt

- **Nếu không có vùng cắt**: Sử dụng thư viện REMBG để tự động loại bỏ nền.

Hình 3.19: Hình ảnh đã xử lý khi không áp dụng vùng cắt

- **Mục đích**: Loại bỏ các phần không liên quan (nền) để tập trung vào bề mặt sản phẩm, giảm nhiễu trong quá trình phát hiện khuyết điểm. Đồng thời, khi kiểm tra ngoại quan, các vị trí xung quanh mép dưới của sản phẩm sẽ không cần phải kiểm tra bởi vì các vị trí này không che khuất bởi màng bảo vệ của công đoạn sau. Vậy nên, các vị trí này sẽ được để lại cho công đoạn kiểm tra dán màng bảo vệ.
- **Điều kiện**: Kiểm tra Xóa nền bằng vùng cắt chưa?
  - Nếu **Có (Y)**: Chuyển sang chuyển đổi ảnh xám.
  - Nếu **Không (N)**: Sử dụng REMBG và tiếp tục.

g. Chuyển đổi và xử lý ảnh (Black Spot Detection)

- Mô tả:
  - **Chuyển sang ảnh xám**: Chuyển ảnh từ RGB sang grayscale để đơn giản hóa xử lý.
  - **Phân ngưỡng và phát hiện điểm đen**: Áp dụng ngưỡng để xác định vùng tối nghi là điểm đen.
  - **Đánh dấu contour**: Sử dụng thuật toán contour để vẽ đường viền quanh khu vực khuyết điểm.

- **Mục đích**: Phát hiện và định vị chính xác các điểm đen trên bề mặt sản phẩm, đáp ứng tiêu chuẩn kiểm tra chất lượng.
- **Tiếp theo**: Chuyển sang hiển thị kết quả.

h. Hiển thị kết quả trên GUI

- **Mô tả**: Ảnh đã xử lý (với các điểm đen được đánh dấu) được hiển thị trên vùng canvas của GUI, cùng với kết quả kiểm tra (số lượng khuyết điểm, trạng thái).
- **Mục đích**: Cung cấp phản hồi trực quan cho người dùng, hỗ trợ đánh giá và lưu trữ dữ liệu để thống kê.
- **Kết thúc**: Quy trình hoàn tất khi kết quả được hiển thị.

3.3.4 Kết luận

Lưu đồ thuật toán này mô tả một quy trình kiểm tra bề mặt sản phẩm bằng hình ảnh rõ ràng, logic và linh hoạt, với hai chế độ vận hành (Tự động và Thủ công) để đáp ứng các nhu cầu khác nhau. Hệ thống bắt đầu bằng việc khởi tạo thiết bị, chọn chế độ, gửi lệnh đến ESP32 để điều khiển servo, chụp ảnh, xử lý ảnh bằng các bước tách nền (ROI/REMBG), chuyển đổi xám, phân ngưỡng và phát hiện contour, cuối cùng hiển thị kết quả trên GUI. Sự kết hợp giữa tự động hóa (dành cho hiệu quả cao) và điều khiển thủ công (dành cho tính linh hoạt) làm tăng giá trị thực tiễn của hệ thống. Tuy nhiên, hiệu suất có thể được tối ưu hơn bằng cách cải thiện tốc độ xử lý ảnh hoặc tích hợp thêm các thuật toán phát hiện khuyết điểm khác (như vết xước) trong tương lai.

Chương 4: THIẾT KẾ CHƯƠNG TRÌNH ĐIỀU KHIỂN HỆ THỐNG

4.1 Mô tả chung

Hệ thống kiểm tra bề mặt sản phẩm bằng hình ảnh được thiết kế để tự động hoặc thủ công chụp ảnh từng mặt của sản phẩm thông qua camera, sau đó xử lý ảnh để phát hiện các khuyết điểm, cụ thể là các điểm đen trên bề mặt. Hệ thống tích hợp phần mềm giao diện người dùng (GUI) được viết bằng Python với thư viện Tkinter, cho phép tương tác trực tiếp với phần cứng như ESP32, camera USB, servo hoặc động cơ bước. Phần mềm hỗ trợ hai chế độ vận hành: Tự động (Auto Mode) và Thủ công (Manual Mode).

Mục đích :

- Tự động hóa kiểm tra chất lượng: Hệ thống nhằm thay thế hoặc hỗ trợ con người trong việc kiểm tra bề mặt sản phẩm, đảm bảo phát hiện các khuyết điểm một cách nhanh chóng, chính xác và nhất quán.
- Nâng cao hiệu quả sản xuất: Giảm thời gian và công sức kiểm tra thủ công, đặc biệt trong các dây chuyền sản xuất quy mô lớn.
- Đảm bảo tính linh hoạt: Cung cấp cả chế độ tự động (phù hợp với sản xuất hàng loạt) và thủ công (phù hợp với kiểm tra chi tiết hoặc kiểm tra lại).
- Hỗ trợ phân tích dữ liệu: Lưu trữ và hiển thị kết quả kiểm tra để phục vụ thống kê, phân tích và cải thiện chất lượng sản phẩm.

4.2 Quy trình vận hành tổng quát

Quy trình hoạt động được thực hiện qua các bước chính:

4.2.1 Khởi động hệ thống

a. Mô tả:

- Người dùng bật phần mềm giao diện.
- Phần mềm khởi tạo các tham số, kết nối tới thiết bị ESP32 qua UART, mở camera và tải cấu hình vùng cắt (ROI).

b. Mục đích:

- **Chuẩn bị hệ thống sẵn sàng hoạt động**: Khởi động phần mềm và thiết lập các thông số ban đầu để đảm bảo tất cả các thành phần (phần mềm và phần cứng) hoạt động đồng bộ.
- **Thiết lập kết nối với phần cứng**: Kết nối với ESP32 qua UART để đảm bảo giao tiếp đáng tin cậy giữa phần mềm và phần cứng, cho phép gửi và nhận lệnh chính xác.
- **Kích hoạt camera**: Mở camera để sẵn sàng ghi nhận hình ảnh, đảm bảo dữ liệu đầu vào cho quá trình xử lý ảnh.
- **Tải cấu hình vùng cắt (ROI)**: Tải các thông số vùng quan tâm (nếu có) để tập trung xử lý ảnh trong khu vực cần thiết, giúp tăng hiệu quả và giảm thời gian xử lý.

- **Đảm bảo trạng thái ban đầu ổn định**: Khởi tạo các tham số giúp hệ thống hoạt động trong trạng thái được kiểm soát, tránh lỗi do cấu hình không đúng.

4.2.2 Lựa chọn chế độ vận hành

Hình 4.1: Giao diện lựa chọn chế độ vận hành

a. Mô tả:

- Người dùng chọn chế độ Tự động hoặc Thủ công thông qua nút chuyển chế độ trên giao diện.

b. Mục đích:

- **Cung cấp sự linh hoạt cho người dùng**: Cho phép người dùng chọn chế độ vận hành phù hợp với nhu cầu cụ thể, chẳng hạn như kiểm tra nhanh toàn bộ sản phẩm (Tự động) hoặc kiểm tra chi tiết một số mặt (Thủ công).
- **Tăng tính tương tác**: Giao diện GUI với nút chuyển chế độ giúp người dùng dễ dàng điều khiển hệ thống mà không cần kiến thức kỹ thuật sâu.
- **Định hướng quy trình vận hành**: Lựa chọn chế độ xác định cách hệ thống thực hiện các bước tiếp theo, đảm bảo quy trình phù hợp với mục tiêu kiểm tra (tốc độ hoặc độ chi tiết).
- **Đáp ứng các kịch bản sử dụng khác nhau**: Chế độ Tự động tối ưu cho sản xuất hàng loạt, trong khi chế độ Thủ công phù hợp cho kiểm tra lại hoặc phân tích chuyên sâu.

4.2.3 Điều khiển phần cứng

a. Mô tả:

- Dựa vào chế độ đã chọn, phần mềm gửi các lệnh điều khiển (ví dụ: mattruoc, matphai, reset) đến ESP32.
- ESP32 điều khiển servo hoặc động cơ bước để xoay mặt sản phẩm tương ứng.
- Sau khi hoàn tất, ESP32 gửi phản hồi "done".

b. Mục đích:

- **Điều khiển chính xác vị trí sản phẩm**: Đảm bảo sản phẩm được xoay đúng mặt cần kiểm tra để camera có thể chụp ảnh chính xác.

- **Đồng bộ hóa phần cứng và phần mềm**: Các lệnh từ phần mềm đến ESP32 và phản hồi "done" đảm bảo rằng mỗi bước xoay hoàn tất trước khi chuyển sang bước chụp ảnh, tránh lỗi đồng bộ.
- **Tự động hóa chuyển động**: Servo hoặc động cơ bước giúp tự động xoay sản phẩm, giảm thiểu sự can thiệp của con người và tăng hiệu quả kiểm tra.
- **Hỗ trợ kiểm tra toàn diện**: Cho phép hệ thống kiểm tra từng mặt sản phẩm một cách có hệ thống, đặc biệt trong chế độ Tự động.
- **Đảm bảo độ tin cậy**: Phản hồi "done" từ ESP32 xác nhận rằng thao tác phần cứng đã hoàn tất, giảm nguy cơ lỗi trong quá trình vận hành.

4.2.4 Chụp ảnh và lưu ảnh

a. Mô tả:

- Khi nhận phản hồi "done", phần mềm tiến hành chụp ảnh thông qua webcam.
- Ảnh được lưu lại vào thư mục theo phân loại từng mặt (RawData/, PicData/).

b. Mục đích:

- **Ghi nhận dữ liệu hình ảnh**: Chụp ảnh để cung cấp dữ liệu đầu vào cho quá trình xử lý và phân tích khuyết điểm trên bề mặt sản phẩm.
- **Đảm bảo chất lượng ảnh đầu vào**: Chụp ảnh ngay sau khi nhận "done" đảm bảo rằng sản phẩm đã được định vị đúng, giúp ảnh chụp có góc nhìn chính xác.
- **Lưu trữ dữ liệu có tổ chức**: Lưu ảnh vào các thư mục riêng (RawData/, PicData/) theo từng mặt sản phẩm giúp dễ dàng quản lý, truy xuất và phân tích sau này.
- **Hỗ trợ kiểm tra và thống kê**: Ảnh được lưu lại phục vụ cho việc phân tích khuyết điểm, kiểm tra lại hoặc báo cáo chất lượng sản phẩm.
- **Tạo cơ sở cho xử lý ảnh**: Ảnh thô (raw) là đầu vào quan trọng cho các bước xử lý tiếp theo, đảm bảo hệ thống có dữ liệu đầy đủ để phát hiện khuyết điểm.

4.2.5 Xử lý ảnh và phân tích lỗi

a. Mô tả:

- Nếu có vùng cắt (ROI) định nghĩa sẵn → dùng để tạo mask xử lý.
- Nếu không, sử dụng thư viện rembg để loại bỏ nền sản phẩm.
- Chuyển ảnh sang ảnh xám (grayscale).
- Phân ngưỡng ảnh để phát hiện vùng tối nghi là điểm đen.
- Áp dụng thuật toán phát hiện contour để đánh dấu khu vực khuyết điểm.

b. Mục đích:

- **Tập trung vào khu vực cần kiểm tra**: Sử dụng ROI hoặc rembg để loại bỏ các phần nền không liên quan, giúp giảm nhiễu và tập trung vào bề mặt sản phẩm.
- **Tối ưu hóa xử lý ảnh**: Chuyển ảnh sang grayscale giảm độ phức tạp tính toán, tăng tốc độ xử lý và chuẩn bị cho bước phân ngưỡng.

- **Phát hiện khuyết điểm chính xác**: Phân ngưỡng ảnh dựa trên độ sáng giúp xác định các vùng tối (điểm đen) theo tiêu chuẩn được định nghĩa (ngưỡng = độ sáng trung bình * 0.6).
- **Đánh dấu và định vị khuyết điểm**: Thuật toán contour giúp xác định và đánh dấu chính xác vị trí, kích thước của các điểm đen, cung cấp thông tin trực quan và dễ hiểu.
- **Đảm bảo đáp ứng tiêu chuẩn chất lượng**: Quy trình xử lý ảnh được thiết kế để phát hiện các khuyết điểm nhỏ (≥ 0.02 mm²), đáp ứng yêu cầu kiểm tra chất lượng cao trong sản xuất.

4.2.6 Hiển thị kết quả và ghi log

a. Mô tả:

- Ảnh đã xử lý được hiển thị trong vùng canvas giao diện GUI.
- Kết quả kiểm tra được hiển thị, lưu lại để thống kê.

b. Mục đích:

- **Cung cấp phản hồi trực quan**: Hiển thị ảnh đã xử lý với các vùng khuyết điểm được đánh dấu (ví dụ: đường viền đỏ quanh điểm đen) giúp người dùng dễ dàng đánh giá kết quả kiểm tra.
- **Hỗ trợ ra quyết định**: Kết quả kiểm tra được hiển thị ngay lập tức cho phép người vận hành quyết định sản phẩm đạt hay không đạt tiêu chuẩn chất lượng.
- **Lưu trữ dữ liệu để phân tích**: Ghi log kết quả kiểm tra vào hệ thống giúp theo dõi lịch sử, thống kê tỷ lệ khuyết tật và cải thiện quy trình sản xuất.
- **Tăng tính tương tác với người dùng**: Giao diện GUI cung cấp trải nghiệm trực quan, giúp người dùng không chuyên cũng có thể hiểu và sử dụng hệ thống.
- **Hỗ trợ báo cáo và truy xuất**: Dữ liệu lưu trữ (ảnh và kết quả) có thể được sử dụng để tạo báo cáo chất lượng hoặc kiểm tra lại khi cần.

c. Đặc điểm của quy trình:

- Đảm bảo quy trình kiểm tra có tổ chức: Quy trình được chia thành các bước rõ ràng để đảm bảo hệ thống hoạt động ổn định và dễ dàng bảo trì.
- Tăng tính tương tác với người dùng: Giao diện GUI giúp người dùng dễ dàng khởi động, chọn chế độ và theo dõi kết quả.
- Tích hợp phần cứng và phần mềm: Kết nối chặt chẽ giữa phần mềm và phần cứng (ESP32, camera) đảm bảo điều khiển chính xác và xử lý dữ liệu hiệu quả.
- Lưu trữ dữ liệu: Lưu ảnh và kết quả kiểm tra để phục vụ phân tích chất lượng sản phẩm hoặc truy xuất khi cần.

4.2.7 Ưu điểm của quy trình:

- Tính tuần tự và logic: Các bước được thiết kế để tối ưu hóa hiệu suất, từ khởi động, chụp ảnh đến xử lý và hiển thị kết quả.
- Khả năng mở rộng: Quy trình này cho phép thêm các bước hoặc tích hợp các thiết bị phần cứng mới nếu cần.
- Độ tin cậy: Phản hồi "done" từ ESP32 đảm bảo rằng mỗi bước điều khiển phần cứng đã hoàn tất trước khi chuyển sang bước tiếp theo, giảm nguy cơ lỗi.

4.3 Nguyên lý hoạt động chế độ Tự động

Trong chế độ Auto, toàn bộ quá trình kiểm tra diễn ra mà không cần tương tác tay người dùng.

Quy trình chi tiết:

Bước 1: Gửi lệnh "reset" đến ESP32 để đưa hệ thống về trạng thái ban đầu.

Bước 2: Lần lượt thực hiện cho từng mặt sản phẩm (theo thứ tự: Trước, Sau, Trái, Phải, Trên):

- Gửi lệnh điều khiển ESP32 xoay mặt tương ứng.
- Chờ ESP32 phản hồi "done".
- Chụp ảnh mặt đang hướng về phía camera.
- Lưu ảnh và thực hiện xử lý ảnh theo quy trình phát hiện điểm đen.

Bước 3: Sau khi kiểm tra đủ các mặt, hệ thống hiển thị kết quả tổng hợp.

*Ưu điểm**:** Nhanh chóng, nhất quán, tiết kiệm công sức người vận hành.*

4.4 Nguyên lý hoạt động chế độ Thủ công

Chế độ Manual cho phép người dùng tự điều khiển từng bước.

Quy trình chi tiết:

- Người dùng nhấn nút chọn mặt cần kiểm tra từ GUI.
- Gửi lệnh tương ứng đến ESP32.
- Sau khi servo xoay mặt xong và gửi "done", hệ thống chụp ảnh.
- Ảnh được xử lý và hiển thị ngay lập tức.

Ưu điểm:

- Linh hoạt khi cần kiểm tra lại hoặc kiểm tra cục bộ một vài mặt.

4.5 Phân tích thuật toán xử lý ảnh và phát hiện điểm đen

Quy trình xử lý ảnh sử dụng thư viện OpenCV và diễn ra theo các bước sau:

4.5.1 Tách nền sản phẩm:

- Nếu người dùng đã vẽ đa giác vùng cắt → dùng để tạo mặt nạ (mask).
- Nếu chưa, dùng rembg để tự động loại bỏ nền.

4.5.2 Chuyển đổi không gian màu:

- Chuyển ảnh RGB sang ảnh xám.

4.5.3 Phân tích độ sáng:

Với điều kiện bắt lỗi điểm đen là 60%, thực hiện tính toán độ sáng trung bình của sản phẩm và nhân với hệ số 0.6 để đảm bảo có thể phát hiện các điểm có độ xám bất thường và theo tiêu chuẩn.

- Tính độ sáng trung bình của vùng sản phẩm.
- Thiết lập ngưỡng điểm tối:

$$\text{Threshold} = \text{Brightness\_avg} * 0.6. \quad (4.1)$$

4.5.4 Quét từng vùng nhỏ:

Phương pháp thiết lập vùng nhỏ:

Bước 1: Tính kích thước thực tế của 1 pixel

- Chiều ngang (2048 pixel ứng với 75mm):

$$\frac{75mm}{2048} \approx 0.0366mm/pixel \quad (4.2)$$

- Chiều dọc (1536 pixel ứng với 60mm):

$$\frac{60mm}{1536} \approx 0.0391mm/pixel \quad (4.3)$$

Bước 2: Tính kích thước của khung nhỏ: dựa trên tiêu chuẩn bắt điểm đen của hệ thống, các điểm đen cần bắt có kích thước ≥0.02mm^2. Vì vậy, ta cần chọn khung nhỏ có kích thước nhỏ hơn 0.02mm^2.

- Ngang: $\sqrt{0.02} = 0.14 \geq 0.0366 * 3 \approx 0.1098mm$ (4.4)

- Dọc: $\sqrt{0.02} = 0.14 \geq 0.0391 * 3 \approx 0.1173mm$ (4.5)

Bước 3: Kết luận:

1. Khung nhỏ là một **khung ảnh 3x3 pixel** có kích thước thực tế xấp xỉ:$0.11mm * 0.12mm$
    - Duyệt ảnh theo từng vùng 3x3 pixel.
    - Nếu trung bình vùng đó < ngưỡng sáng → đánh dấu là điểm đen.
2. **Làm mịn và trích xuất contour:**
    - Làm mượt kết quả bằng bộ lọc trung vị.
    - Tìm contour xung quanh các vùng được đánh dấu.
3. **Vẽ và lưu kết quả:**
    - Các vùng được vẽ bằng đường viền đỏ.
    - Ảnh kết quả được lưu phục vụ phân tích và kiểm tra

4.6 Giao tiếp phần mềm – phần cứng

Bảng 4.1: Mô tả vai trò và chức năng từng thành phần

Thành phần	Vai trò	Mô tả
ESP32	Điều khiển thiết bị	Nhận lệnh từ GUI, xoay mặt, trả về "done"
Camera USB	Ghi nhận hình ảnh	Cung cấp dữ liệu ảnh để xử lý
Phần mềm (Python)	Điều khiển và xử lý ảnh	Gửi lệnh, phân tích ảnh, điều khiển giao diện
Người dùng	Tương tác GUI	Lựa chọn chế độ, kiểm tra và quan sát kết quả

Chương 5: KẾT QUẢ THỰC NGHIỆM

5.1 Ảnh chụp mẫu từng mặt sản phẩm

Mặt trước

Mặt sau

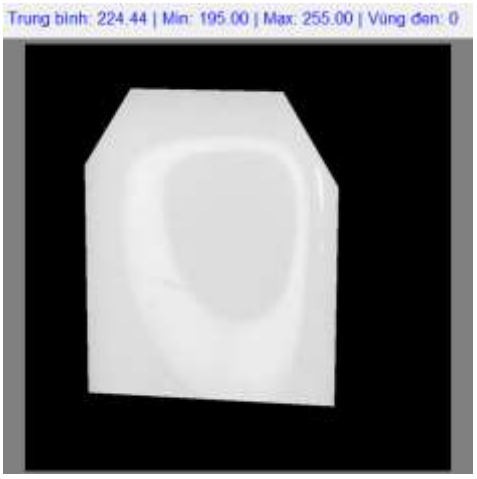
Mặt phải

Mặt trái

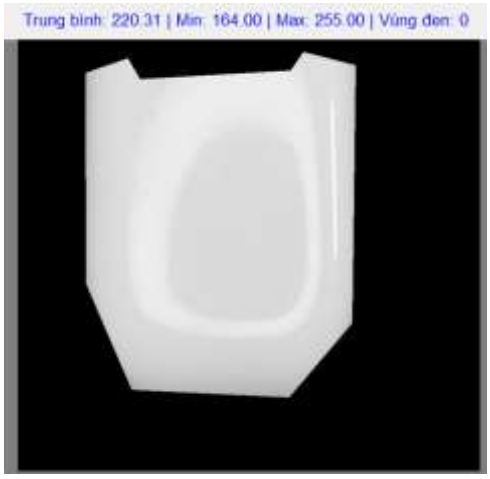
Mặt trên

Hình 5.1: Hình ảnh chụp chưa qua xử lý của các mặt sản phẩm

5.2 Kết quả phân tích điểm đen



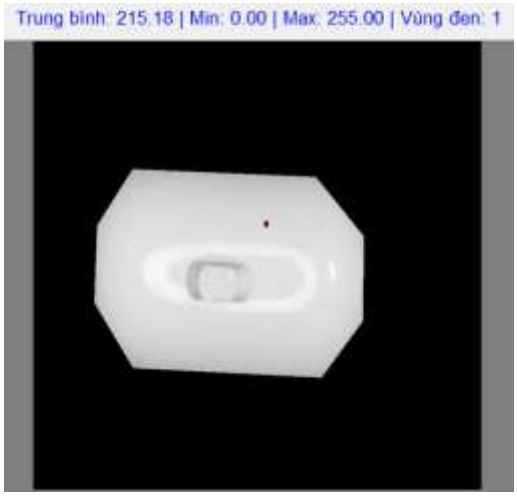
Mặt trước



Mặt sau

Mặt phải

Mặt trái



Mặt trên

Hình 5.2: Hình ảnh đã qua xử lý của các bề mặt sản phẩm

5.3 Độ chính xác nhận diện

Nhằm hạn chế nhiễu do hiện tượng đổ bóng và các phần rìa không mong muốn xung quanh sản phẩm, chương trình sử dụng phương pháp cắt ảnh theo khung có tọa độ cố định.

Cách tiếp cận này giúp tập trung xử lý vào vùng quan trọng, từ đó nâng cao độ chính xác trong việc nhận diện điểm đen bất thường.

Thuật toán phát hiện điểm đen dựa trên độ xám trung bình của vùng ảnh con (WindowSlide), với các ngưỡng được thiết lập trước để điều chỉnh mức độ nhạy. Người dùng có thể siết chặt hoặc nới lỏng tiêu chuẩn phát hiện bằng cách thay đổi các hệ số liên quan đến độ xám. Đồng thời, diện tích vùng phát hiện cũng phụ thuộc trực tiếp vào kích thước của WindowSlide. Việc tăng hoặc giảm kích thước này cho phép điều chỉnh mức độ phát hiện lỗi phù hợp với yêu cầu thực tế. Bên cạnh đó, thời gian chạy trung bình trong mỗi lần kiểm tra đạt 40 giây, đã đảm bảo được yêu cầu về cải thiện tốc độ kiểm tra khi hoạt động trên chuyền.

Để kiểm tra tính ổn định và chính xác của hệ thống, ta có thể áp dụng phương pháp kiểm tra GRR

5.3.1 Giới thiệu về phương pháp kiểm tra GRR

GRR là viết tắt của Gage Repeatability and Reproducibility – một phương pháp trong MSA (Measurement System Analysis) dùng để đánh giá độ tin cậy của hệ thống đo lường.

Mục đích của kiểm tra GRR: GRR được dùng để xác định xem hệ thống đo có đủ ổn định và đáng tin cậy hay không, bằng cách đánh giá:

- **Repeatability (độ lặp lại)** – sai số do thiết bị đo gây ra khi cùng một người đo lặp lại nhiều lần trên cùng một mẫu.
- **Reproducibility (độ tái lập)** – sai số do người đo khác nhau đo cùng một mẫu bằng cùng thiết bị.

Phương pháp kiểm tra GRR gồm các bước chính:

- Chọn mẫu: Thường lấy 10 mẫu đại diện cho dải sản phẩm (tốt, xấu, trung bình…).
- Chọn người đo (operator): 2–3 người đo.
- Lặp lại phép đo: Mỗi người đo từng mẫu 2 hoặc 3 lần, theo thứ tự ngẫu nhiên để tránh nhớ giá trị.
- Thu thập dữ liệu và phân tích bằng phần mềm.
- Kết quả GRR: Trả về các tỷ lệ phần trăm thể hiện mức ảnh hưởng của Repeatability, Reproducibility và tổng GRR.

Tiêu chuẩn đánh giá GRR:

- GRR $\leq$ 10% $\rightarrow$ Hệ thống đo tốt, chấp nhận.
- 10% < GRR $\leq$ 30% $\rightarrow$ Có thể chấp nhận, nhưng nên xem xét cải thiện.
- GRR > 30% $\rightarrow$ Hệ thống đo không đạt, cần cải thiện hoặc thay đổi.

5.3.2 Lựa chọn mẫu:

Để kiểm tra GRR cho hệ thống, cần phải thực hiện thu thập và chọn ra 10 mẫu thử điểm đen có đặc điểm nằm trong dải sản phẩm tốt - trung bình - xấu tương ứng với các khoảng tiêu chuẩn về độ xám và kích thước như sau:

Bảng 5.1: Tiêu chuẩn đánh giá mức độ lỗi

Độ xám / Diện tích	**> 60%**
<0.02mm²	Tốt
0.02 mm² - 0.03mm²	Trung bình
>0.03mm²	Xấu

Bảng 5.2: Danh sách và đặc điểm ngoại quan mẫu kiểm tra

Số mẫu thử nghiệm	Phán định	Đặc điểm ngoại quan lỗi	Đánh giá mức độ
1	NG	Mặt trước: 0.08mm² và 0.05mm²	Xấu
2	OK	Mặt trước: độ xám <60%	Tốt
3	NG	Mặt trước: 0.03mm²; mặt trái: 0.04mm²	Trung bình
4	NG	Mặt trên: 0.05mm²	Xấu
5	OK	/	Tốt
6	OK	0.01mm²	Tốt
7	NG	Mặt phải: 0.02mm² và 0.03mm²	Trung bình
8	NG	Mặt phải: 0.06mm²	Xấu
9	NG	Mặt sau: 0.05mm²	Xấu
10	NG	Mặt sau: 0.07mm²; mặt trái: 0.02mm²	Xấu

5.3.3 Kiểm tra độ lặp lại:

Để kiểm tra độ lặp lại của hệ thống, sử dụng 10 mẫu đã chuẩn bị, tiến hành lấy ngẫu nhiên các mẫu, cố định người thao tác, cố định khuôn, thực hiện cho cả 10 mẫu, mỗi mẫu đo 3 lần. Kết quả sẽ cho thấy sự ổn định của giữa các lần đo tương ứng với các khoảng tiêu chuẩn khác nhau. Nếu kết quả đo có sai lệch, không đồng nhất, nghĩa là độ ổn định của máy giảm xuống và ngược lại.

Bảng 5.3: Kết quả kiểm tra độ lặp lại

Số mẫu thử nghiệm	Kết quả kiểm tra		
	Lần 1	**Lần 2**	**Lần 3**
1	NG (mặt trước:2 lỗi)	NG (mặt trước:2 lỗi)	NG (mặt trước:2 lỗi)
2	OK	OK	OK

3	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)
4	NG (mặt trên:2 lỗi)	NG (mặt trên:2 lỗi)	NG (mặt trên:2 lỗi)
5	OK	OK	OK
6	OK	OK	OK
7	NG (mặt phải:2 lỗi)	NG (mặt phải:2 lỗi)	NG (mặt phải:2 lỗi)
8	NG (mặt phải:1 lỗi)	NG (mặt phải:1 lỗi)	NG (mặt phải:1 lỗi)
9	NG (mặt sau:1 lỗi)	NG (mặt sau:1 lỗi)	NG (mặt sau:1 lỗi)
10	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)

Nhận xét:

a. Tính lặp lại cao**:**

- Cả 10 mẫu đều được đo 3 lần và tất cả các kết quả đều giống nhau ở cả giá trị phân loại (OK/NG) và vị trí lỗi chi tiết.
- Điều này cho thấy tính lặp lại (Repeatability) của người đo hoặc hệ thống đo là rất tốt.

b. Không có sự thay đổi ngẫu nhiên:

- Không có bất kỳ mẫu nào thay đổi từ OK sang NG hay ngược lại giữa các lần đo.
- Không có sự khác biệt về số lượng hoặc vị trí lỗi giữa các lần đo trên cùng một mẫu.

c. Phân loại rõ ràng và nhất quán:

- Kết quả phân loại OK/NG và ghi chú lỗi theo từng mặt của sản phẩm được ghi nhận chính xác, chi tiết và nhất quán.

Kết luận: Kết quả kiểm tra độ lặp lại (Repeatability) cho thấy hệ thống đo hoặc người thực hiện kiểm tra có độ ổn định rất cao, không có biến động giữa các lần đo. Như vậy, độ lặp lại đạt yêu cầu, đảm bảo độ tin cậy của hệ thống/phương pháp kiểm tra trong việc phân loại sản phẩm OK/NG và xác định lỗi cụ thể.

5.3.4 Kiểm tra độ tái lập:

Để kiểm tra độ tái lập của hệ thống, sử dụng 10 mẫu đã chuẩn bị, tiến hành lấy ngẫu nhiên các mẫu, cố định khuôn, thực hiện cho cả 10 mẫu, yêu càu có 3 người thực hiện kiểm tra. Khi thay đổi người kiểm tra, các thao tác kiểm tra có thể gây ra ảnh hưởng đến kết quả đo (Ví dụ: đặt sản phẩm vào khuôn bị lệch, bụi bẩn dính lên sản phẩm, điều chỉnh ảnh sáng không phù hợp,…). Vì vậy, việc kiểm tra độ tái lập cũng giúp xác định được sai số do thay đổi người kiểm tra có thể thoả mãn điều kiện bài toán đặt ra hay không.

Trước khi thực hiện kiểm tra, toàn bộ hệ thống cần phải được tắt và bật lại, khuôn phải được tháo ra trước khi kiểm tra, đèn hỗ trợ chiếu sáng phải được điều chỉnh về mức thấp nhất. Người thao tác bắt buộc phải thực hiện chạy lại hệ thống, lắp lại khuôn cũng như điều chỉnh độ sáng cho hợp lý.

Bảng 5.4: Kết quả kiểm tra độ tái lập

Số mẫu thử nghiệm	Kết quả kiểm tra		
	Người kiểm tra số 1	Người kiểm tra số 2	Người kiểm tra số 3
1	NG (mặt trước:2 lỗi)	NG (mặt trước:2 lỗi)	NG (mặt trước:2 lỗi)
2	OK	OK	OK
3	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)
4	NG (mặt trên:2 lỗi)	NG (mặt trên:2 lỗi)	NG (mặt trên:2 lỗi)
5	OK	OK	OK
6	OK	OK	OK
7	NG (mặt phải:2 lỗi)	NG (mặt phải:2 lỗi)	NG (mặt phải:2 lỗi)
8	NG (mặt phải:1 lỗi)	NG (mặt phải:1 lỗi)	NG (mặt phải:1 lỗi)
9	NG (mặt sau:1 lỗi)	NG (mặt sau:1 lỗi)	NG (mặt sau:1 lỗi)
10	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)

Nhận xét:

a. Kết quả kiểm tra giống nhau tuyệt đối giữa các người vận hành AOI:
   - Cả 10 mẫu đều cho kết quả giống nhau về phân loại **OK/NG** và mô tả lỗi cụ thể (mặt nào, bao nhiêu lỗi) trên 3 lần kiểm tra bởi 3 người vận hành khác nhau.

b. Không có sự biến động do thao tác vận hành hệ thống:
   - Không có sai số hoặc khác biệt giữa các người sử dụng hệ thống AOI.
   - Điều này chứng minh thao tác cài đặt, vận hành AOI giữa các người kiểm tra là rất đ**ồng nhất**, không gây ảnh hưởng tới kết quả đầu ra.

c. Hệ thống AOI hoạt động ổn định, không phụ thuộc người vận hành:
   - Với cùng một mẫu sản phẩm, hệ thống AOI cho ra kết quả lặp lại chính xác **bất kể người vận hành là ai**.

Kết luận: Kết quả kiểm tra độ tái lập (Reproducibility) chứng minh rằng sự khác biệt giữa các người vận hành hệ thống AOI là không đáng kể hoặc bằng 0. Hệ thống AOI hoạt động ổn định, nhất quán và không phụ thuộc vào người sử dụng, đảm bảo độ tin cậy khi đưa vào vận hành thực tế trong dây chuyền sản xuất. Vì vậy, độ tái lập đạt yêu cầu, hệ thống AOI có thể được tin tưởng trong việc phát hiện lỗi sản phẩm một cách chính xác và khách quan.

5.3.5 Kiểm tra độ chính xác:

Ngoài việc kiểm tra các hạng mục độ lặp lại và độ tái lập của GRR, hệ thống cũng cần phải thực hiện kiểm tra tương quan về độ chính xác với nhân viên ngoại quan. Quá trình thực hiện yêu cầu sử dụng 10 mẫu thử đã được chuẩn bị. Một nhân viên ngoại quan có kinh nghiệm thực hiện kiểm tra theo thứ tự ngẫu nhiên các mẫu đó, ghi chép kết quả kiểm tra và thực hiện kiểm tra các mẫu này bằng hệ thống. Kết quả kiểm tra của cả hai đối tượng sẽ được ghi lại và so sánh để kiểm tra độ chính xác so với con người.

Bảng 5.5: Kết quả kiểm tra độ chính xác

Số mẫu thử nghiệm	Kết quả kiểm tra	
	AOI	**Người ngoại quan**
1	NG (mặt trước:2 lỗi)	NG (mặt trước:2 lỗi)
2	OK	OK
3	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)	NG (mặt trước: 1 lỗi; mặt trái: 1 lỗi)
4	NG (mặt trên:1 lỗi)	NG (mặt trên:1 lỗi)
5	OK	OK
6	OK	OK
7	NG (mặt phải:2 lỗi)	NG (mặt phải:2 lỗi)
8	NG (mặt phải:1 lỗi)	NG (mặt phải:1 lỗi)
9	NG (mặt sau:1 lỗi)	NG (mặt sau:1 lỗi)
10	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)	NG (mặt sau: 1 lỗi, mặt trái: 1 lỗi)

Nhận xét:

- 10/10 mẫu có kết quả giống nhau hoàn toàn giữa AOI và người ngoại quan.
- Trùng khớp cả về phân loại OK/NG và vị trí, số lượng lỗi.
- AOI nhận diện chính xác cả lỗi nhỏ và lỗi ở các mặt khó quan sát.

Kết luận: Kết quả kiểm tra cho thấy hệ thống AOI có độ chính xác rất cao, hoàn toàn tương đương với phương pháp kiểm tra bằng mắt thường. Việc phát hiện lỗi đúng về cả số lượng và vị trí trên từng sản phẩm cho thấy AOI có thể thay thế đáng tin cậy cho người kiểm tra ngoại quan. Điều này không chỉ giúp đảm bảo tính nhất quán và khách quan trong việc đánh giá chất lượng sản phẩm, mà còn góp phần nâng cao hiệu quả kiểm tra trong môi trường sản xuất thực tế.

5.3.6 Kết luận:

Thông qua các kiểm tra GRR, bao gồm độ lặp lại (Repeatability), độ tái lập (Reproducibility) và độ chính xác (Accuracy), hệ thống kiểm tra bằng AOI đã chứng minh được tính ổn định và độ tin cậy cao. Cụ thể, kết quả kiểm tra cho thấy toàn bộ 10 mẫu thử đều cho kết quả giống nhau hoàn toàn giữa các lần đo lặp lại cũng như giữa các người vận hành khác nhau, không xuất hiện sai số hay biến động do thao tác. Ngoài ra, kết quả so sánh với kiểm tra bằng mắt thường từ người ngoại quan có kinh nghiệm cũng cho thấy sự trùng khớp tuyệt đối, chứng minh khả năng nhận diện lỗi chính xác cả về số lượng, vị trí và mức độ.

Bên cạnh đó, khi thực hiện đánh giá độ tương quan giữa con người và thiết bị AOI, hệ thống đạt độ chính xác tuyệt đối (100%) đối với các mặt: trước, sau, trái và phải. Tuy nhiên, đối với mặt trên, do đặc thù cấu trúc có đầu liệu thừa và hiện tượng đổ bóng gây ảnh hưởng đến quá trình xử lý ảnh, độ chính xác giảm xuống còn 90%. Dù vậy, mức sai lệch này vẫn nằm trong giới hạn chấp nhận được và không làm ảnh hưởng đến tính nhất quán chung của hệ thống. Phương pháp GRR được áp dụng theo hướng dẫn tiêu chuẩn công nghiệp.[3]

Như vậy, hệ thống AOI đáp ứng đầy đủ các yêu cầu về tính lặp lại, tái lập và độ chính xác trong kiểm tra chất lượng sản phẩm. Kết quả này tương đồng với các hệ thống kiểm tra ngoại quan tự động được nghiên cứu tại Việt Nam.[4] Hệ thống hoàn toàn có thể thay thế đáng tin cậy cho phương pháp kiểm tra thủ công trong môi trường sản xuất thực tế, góp phần nâng cao hiệu suất và đảm bảo tính khách quan trong đánh giá chất lượng sản phẩm.

5.4 Giao diện phần mềm thực tế

Hình 5.3 Giao diện chọn phương pháp chụp

5.4.1 Chế độ thủ công

Bước 1: Nhấn nút "Chạy thủ công" để thực hiện chạy chương trình chụp ảnh và phân tích thủ công. Cửa sổ "Chế độ chạy thủ công" mở ra.

Hình 5.4: Giao diện chế độ chạy thủ công

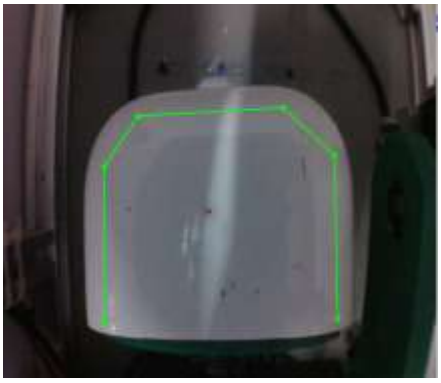
Bước 2: Điều chỉnh vị trí sản phẩm theo mong muốn và nhấn nút "Chụp ảnh" để lưu ảnh thô.

Hình 5.5: Hình ảnh sản phẩm đã được chụp

Bước 3: Nhấn nút “Chọn ảnh” để chọn ảnh cần phân tích.

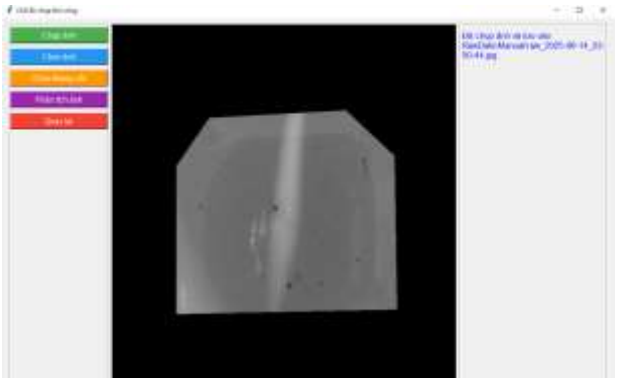
Bước 4: Nhấn nút “Chọn khung cắt” để thực hiện xác định tọa độ khung cắt ảnh.

Hình 5.6: Nhập số cạnh của khung cắt



Hình 5.7: Chọn tọa độ khung cắt và số cạnh của khung

Bước 5: Nhấn nút “Phân tích ảnh” để thực hiện phân tích, khoanh vùng lỗi và lưu hình ảnh kết quả.



Hình 5.8: Giao diện sau khi phân tích

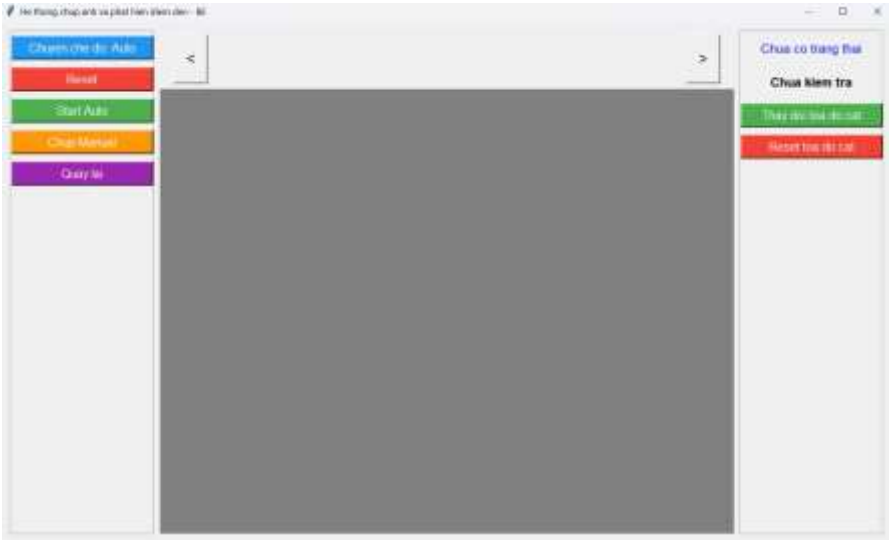
Bước 6: Nhấn nút "Quay lại" để trở về giao diện chọn chế độ chụp ban đầ

5.4.2 Chế độ tự động

Bước 1: Nhấn nút "Chạy tự động" để thực hiện chạy chương trình chụp ảnh và phân tích tự động. Cửa sổ "Chọn loại sản phẩm" mở ra.

Hình 5.9: Cửa sổ chọn loại sản phẩm

Bước 2: Nhấn chọn loại sản phẩm cần phân tích, cửa sổ "Hệ thống chụp ảnh và phát hiện điểm đen" tương ứng với sản phẩm mở ra.



Hình 5.10: Giao diện hệ thống chụp ảnh và phát hiện điểm đen

Bước 3: Nhấn nút "Thay đổi tọa độ cắt" để điều chỉnh khung cắt sản phẩm.

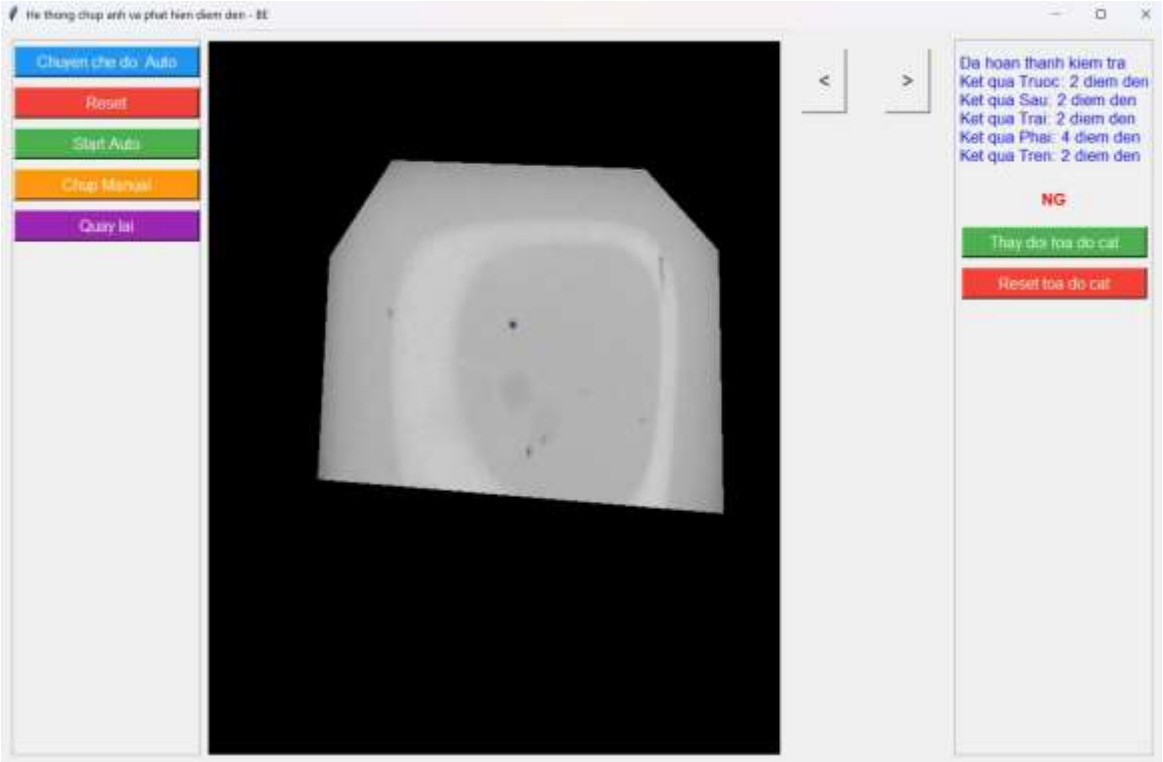
Hình 5.11: Chọn mặt cần thay đổi khung cắt

Bước 4: Nhập số cạnh và tọa độ khung cắt

Hình 5.12: Giao diện nhập số cạnh của khung cắt

Hình 5.13: Chọn tọa độ cắt của khung

Bước 4: Chọn “Start Auto” để bắt đầu thực hiện kiểm tra tự động. Kết quả được hiển thị tại khu vực bên phải của cửa sổ hiển thị, bao gồm số điểm đen phát hiện được ở mỗi mặt cũng như kết quả kiểm tra OK/NG.



Hình 5.14: Giao diện hiển thị kết quả kiểm tra

5.5 Đánh giá hệ thống

Hệ thống hoạt động tương đối ổn định và chính xác. Tuy nhiên, khi có sai lệch tại vị trí khuôn cố định sản phẩm, tọa độ của các khung cắt ảnh có thể không còn chính xác nữa. Khi đó cần phải hiệu chỉnh lại tọa độ cho phù hợp.

Chương 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận về hoạt động hệ thống

Hệ thống có thể được áp dụng cho quy trình sản xuất công nghiệp, tích hợp trong các trạm máy tự động hóa hoặc hoạt động riêng lẻ tùy thuộc vào nhu cầu của doanh nghiệp. Mức độ ổn định và khả năng vận hành liên tục trong thời gian dài cho thấy hệ thống có tiềm năng ứng dụng thực tế cao trong các dây chuyền sản xuất có cường độ lớn.

6.2 Những gì đã đạt được

- Hệ thống đã thành công trong việc phát hiện điểm đen trên bề mặt sản phẩm, bao quát được các vùng cần kiểm tra.
- Các tiêu chuẩn điểm đen được bắt chính xác về cả độ xám và diện tích.
- Phát triển được các chức năng chụp thủ công và chụp tự động
- Có thể áp dụng cho nhiều loại sản phẩm.
- Tốc độ phát hiện điểm đen khá nhanh và tỷ lệ chính xác tương đối cao.
- Hệ thống có cơ cấu điều chỉnh ánh sáng và điều chỉnh khu vực cắt ảnh linh hoạt, có thể triển khai được ở nhiều môi trường và nhiều điều kiện làm việc.
- Thiết kế giao diện phần mềm đơn giản, dễ sử dụng, hỗ trợ người vận hành không cần kỹ năng lập trình chuyên sâu.
- Hệ thống có khả năng lưu trữ ảnh lỗi và xuất báo cáo kiểm tra phục vụ truy vết chất lượng sản phẩm.
- Cho phép chuyển đổi nhanh giữa các chế độ Auto/Manual phù hợp với các tình huống kiểm tra đa dạng.

6.3 Hướng phát triển

- Phát triển hệ thống kiểm tra tích hợp cho nhiều loại sản phẩm
- Rút ngắn tốc độ kiểm tra của hệ thống, loại bỏ các bước không cần thiết
- Phát triển hệ thống thay đổi độ sáng cho phù hợp với độ sáng môi trường, tránh gây ra nhiễu do đổ bóng
- Phát triển thêm cơ cấu tự động lắp sản phẩm vào jig.
- Phát triển thêm chức năng điều chỉnh góc độ chụp trong chế độ thủ công.
- Tích hợp AI để tự động phân loại lỗi theo mức độ nghiêm trọng, hỗ trợ công đoạn ra quyết định trong kiểm tra chất lượng.
- Xây dựng cơ sở dữ liệu lỗi để học máy, tối ưu dần ngưỡng nhận diện theo từng loại sản phẩm.
- Phát triển phiên bản giao diện web hoặc kết nối IoT để giám sát từ xa và thống kê kết quả theo thời gian thực.

- Tối ưu thuật toán xử lý ảnh để giảm thiểu tài nguyên tính toán, có thể triển khai trên các thiết bị nhúng nhỏ gọn như Raspberry Pi hoặc Jetson Nano.
- Tích hợp tính năng tự động hiệu chỉnh camera (auto calibration) để đảm bảo chất lượng hình ảnh ổn định khi thay đổi góc chụp hoặc môi trường làm việc.

KẾT LUẬN

Đề tài đã thành công trong việc xây dựng một hệ thống kiểm tra ngoại quan tự động sử dụng công nghệ thị giác máy tính để phát hiện lỗi điểm đen trên bề mặt sản phẩm tại xưởng sản xuất BU8 – SBU5 của Công ty Luxshare ICT – Nghệ An. Hệ thống đáp ứng đầy đủ các yêu cầu đề ra, bao gồm khả năng nhận diện chính xác điểm đen với độ xám ≥ 60% và diện tích ≥ 0.02 mm², đảm bảo thời gian kiểm tra tiêu chuẩn (60 giây/sản phẩm), và kiểm tra toàn diện cả 5 mặt sản phẩm (Trước, Sau, Trái, Phải, Trên). Kết quả kiểm tra GRR cho thấy hệ thống đạt độ lặp lại và tái lập tuyệt đối, với độ chính xác tương đương 100% so với kiểm tra thủ công bởi nhân viên ngoại quan trên các mặt Trước, Sau, Trái, Phải, và 90% trên mặt Trên do ảnh hưởng của đổ bóng và cấu trúc sản phẩm.

Hệ thống không chỉ giảm thiểu sự phụ thuộc vào nhân công, mà còn nâng cao hiệu suất, tính nhất quán và độ tin cậy trong kiểm tra chất lượng. Giao diện phần mềm thân thiện, tích hợp cả chế độ tự động và thủ công, cùng khả năng lưu trữ và xuất báo cáo giúp hỗ trợ hiệu quả cho việc quản lý chất lượng. Những kết quả này khẳng định tiềm năng ứng dụng thực tế của hệ thống trong dây chuyền sản xuất công nghiệp, đặc biệt trong các môi trường đòi hỏi kiểm tra chất lượng nghiêm ngặt.

Hướng Phát Triển

Để nâng cao hiệu quả và mở rộng ứng dụng của hệ thống, một số hướng phát triển trong tương lai bao gồm:

- Tích hợp AI và học sâu: Ứng dụng các mô hình như YOLO hoặc U-Net để tự động phân loại lỗi theo mức độ nghiêm trọng, đồng thời xây dựng cơ sở dữ liệu lỗi để tối ưu ngưỡng nhận diện theo từng loại sản phẩm. Việc xây dựng cơ sở dữ liệu lỗi có thể tham khảo các bộ dữ liệu lớn như ImageNet để nâng cao hiệu quả huấn luyện mô hình. [5]
- Tối ưu tốc độ và tài nguyên: Rút ngắn thời gian kiểm tra bằng cách loại bỏ các bước không cần thiết và tối ưu thuật toán để triển khai trên các thiết bị nhúng như Raspberry Pi hoặc Jetson Nano.
- Tự động hóa nâng cao: Phát triển cơ cấu tự động lắp sản phẩm vào jig, tích hợp tính năng tự động hiệu chỉnh camera và điều chỉnh ánh sáng thích ứng với môi trường để giảm nhiễu do đổ bóng.
- Kết nối IoT và giám sát từ xa: Xây dựng giao diện web hoặc tích hợp IoT để giám sát kết quả theo thời gian thực, hỗ trợ quản lý và phân tích dữ liệu từ xa. Xu hướng tích hợp IoT trong kiểm tra chất lượng đã được nghiên cứu và chứng minh hiệu quả trong sản xuất công nghiệp.[6]
- Mở rộng ứng dụng: Phát triển hệ thống để kiểm tra nhiều loại sản phẩm khác nhau, bổ sung khả năng nhận diện các lỗi ngoại quan khác như xước, mẻ, hoặc thiếu liệu.

Những cải tiến này sẽ giúp hệ thống không chỉ đáp ứng tốt hơn nhu cầu sản xuất hiện tại mà còn mở ra tiềm năng ứng dụng rộng rãi trong các ngành công nghiệp khác, góp phần nâng cao chất lượng và hiệu quả sản xuất.

TÀI LIỆU THAM KHẢO

[1] Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[3] AIAG. (2010). *Measurement Systems Analysis (MSA)* (4th ed.). Automotive Industry Action Group.

[4] Nguyen, T. D., et al. (2022). Nghiên cứu, thiết kế, chế tạo hệ thống kiểm tra đánh giá ngoại quan sản phẩm (QC) sử dụng công nghệ thị giác máy kết hợp trí thông minh nhân tạo AI. *Tạp chí Công Thương*.

[5] Russakovsky, O., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, 115(3)*, 211-252.

[6] Zhang, Y., & Zhang, J. (2023). IoT-based Quality Inspection System for Industrial Manufacturing. *Journal of Manufacturing Systems*, *68*, 123-135.

PHỤ LỤC

Chương trình giao diện và nhận dạng, phân loại điểm đen trên máy tính:

```
1. import tkinter as tk
2. from tkinter import filedialog, StringVar, OptionMenu, Canvas, Frame, Toplevel,
Entry, messagebox, Button, Label
3. import cv2
4. import numpy as np
5. from PIL import Image, ImageTk
6. import serial
7. import time
8. import os
9. from datetime import datetime
10. from rembg import remove
11. import json
12.
13. # Cấu hình
14. CROP_CONFIG_FILE = "crop_coords.json"
15. faces = ["Truoc", "Sau", "Trai", "Phai", "Tren"]
16. products = ["BE", "TE", "TI", "HSG", "CAP"]
17. BAUD_RATE = 9600
18. SERIAL_PORT = 'COM5'
19.
20. # Khởi tạo serial
21. try:
22.     ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
23.     time.sleep(2)
24. except serial.SerialException as e:
25.     print(f"Khong the ket noi voi {SERIAL_PORT}: {e}")
26.     messagebox.showerror("Loi", f"Khong the ket noi voi {SERIAL_PORT}. Kiem tra
cong COM.")
27.     exit()
28.
29. # Biến toàn cục
30. auto_mode = True
31. photo_count = 0
32. capture_status = ""
33. crop_coords = {}
34. processed_images = []
35. current_image_index = 0
36. arrow_frame = None
37. prev_button = None
38. next_button = None
39. manual_window = None
40. current_product = ""
41. root = None
42. manual_root = None
43. selected_image_path = None
44. manual_crop_name = "ManualCrop"
45. ok_ng_label = None
46. # Di chuyển selected_face khỏi đây, sẽ khởi tạo trong show_initial_interface()
47.
48. # Tạo thư mục lưu ảnh
49. for product in products:
50.     for folder in faces:
51.         os.makedirs(os.path.join("RawData", product, folder), exist_ok=True)
52.         os.makedirs(os.path.join("PicData", product, folder), exist_ok=True)
53. os.makedirs(os.path.join("RawData", "Manual"), exist_ok=True)
54.
55. # Load config cắt ảnh
56. if os.path.exists(CROP_CONFIG_FILE):
57.     with open(CROP_CONFIG_FILE, "r") as f:
58.         crop_coords = json.load(f)
59.
60. # Hàm chụp ảnh từ camera
61. def capture_image(folder=None, product=None):
62.     global photo_count, capture_status
63.     try:
```

```
64.         cap = cv2.VideoCapture(0)
65.         if not cap.isOpened():
66.             messagebox.showerror("Loi", "Khong the mo camera")
67.             return False
68.
69.         cap.set(cv2.CAP_PROP_FRAME_WIDTH, 2048)
70.         cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1536)
71.
72.         actual_width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
73.         actual_height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
74.         print(f"Do phan giai thuc te: {actual_width}x{actual_height}")
75.
76.         ret, frame = cap.read()
77.         if ret:
78.             photo_count += 1
79.             timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
80.             if folder and product:
81.                 raw_filename = f"raw_{timestamp}_{folder}.jpg"
82.                 raw_output_path = os.path.join("RawData", product, folder,
raw_filename)
83.                 cv2.imwrite(raw_output_path, frame)
84.                 filename = f"output_{timestamp}_{folder}.jpg"
85.                 output_path = os.path.join("PicData", product, folder, filename)
86.                 cv2.imwrite(output_path, frame)
87.                 capture_status = f"Da chup anh va luu vao {raw_output_path}"
88.                 update_status()
89.                 return output_path
90.             else:
91.                 raw_filename = f"raw_{timestamp}.jpg"
92.                 raw_output_path = os.path.join("RawData", "Manual",
raw_filename)
93.                 cv2.imwrite(raw_output_path, frame)
94.                 capture_status = f"Da chup anh va luu vao {raw_output_path}"
95.                 update_status()
96.                 return raw_output_path
97.         else:
98.             capture_status = "Loi: Khong the chup anh tu camera"
99.             update_status()
100.                return False
101.        except Exception as e:
102.            print(f"Loi chup anh: {e}")
103.            capture_status = f"Loi chup anh: {e}"
104.            update_status()
105.            return False
106.        finally:
107.            if cap and cap.isOpened():
108.                cap.release()
109.
110.    # Hàm phân tích điểm đen
111.    def analyze_black_spots(file_path, face_name):
112.        try:
113.            img = cv2.imread(file_path)
114.            if img is None:
115.                print(f"Loi: Khong the doc file anh: {file_path}")
116.                return "Khong the doc anh", 0
117.            img = cv2.resize(img, (800, 800))
118.            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
119.
120.            if face_name in crop_coords and len(crop_coords[face_name]) >= 3:
121.                points = crop_coords[face_name]
122.                mask = np.zeros(img.shape[:2], np.uint8)
123.                points_array = np.array(points, np.int32)
124.                cv2.fillPoly(mask, [points_array], 1)
125.            else:
126.                mask = remove_background_with_rembg(img)
127.
128.            segmented = gray * mask
129.            region = segmented[mask == 1]
130.            if region.size == 0:
131.                print(f"Loi: Khong tim thay vung san pham cho {face_name}")
132.                return "Khong tim thay vung san pham", 0
133.
134.            brightness_avg = np.mean(region)
135.            threshold_dark = brightness_avg * 0.6
136.            dark_mask = np.zeros_like(segmented, dtype=np.uint8)
137.            window_size = 3
138.
```

```
            for y in range(0, segmented.shape[0] - window_size, window_size):
                for x in range(0, segmented.shape[1] - window_size, window_size):
                    roi_mask = mask[y:y + window_size, x:x + window_size]
                    roi_gray = segmented[y:y + window_size, x:x + window_size]
                    if np.sum(roi_mask) == 0:
                        continue
                    roi_valid = roi_gray[roi_mask == 1]
                    if roi_valid.size == 0:
                        continue
                    if np.mean(roi_valid) < threshold_dark:
                        dark_mask[y:y + window_size, x:x + window_size] = 255

            dark_mask = cv2.medianBlur(dark_mask, 3)
            contours, _ = cv2.findContours(dark_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            result_img = cv2.cvtColor(segmented, cv2.COLOR_GRAY2BGR)
            cv2.drawContours(result_img, contours, -1, (0, 0, 255), 1)
            cv2.imwrite(file_path, result_img)

            return f"Ket qua {face_name}: {len(contours)} diem den",
len(contours)
        except Exception as e:
            print(f"Loi phan tich diem den: {e}")
            return f"Loi phan tich: {e}", 0

    # Hàm gửi lệnh và chờ phản hồi
    def send_command(command):
        max_attempts = 2
        timeout = 4
        for attempt in range(max_attempts):
            try:
                ser.write(command.encode())
                start_time = time.time()
                while time.time() - start_time < timeout:
                    if ser.in_waiting > 0:
                        response = ser.readline().decode().strip()
                        if response == "done":
                            return True
                    time.sleep(0.01)
                print(f"Thu {attempt + 1}: Khong nhan duoc 'done' tu ESP32 cho
lenh {command}")
            except serial.SerialException as e:
                print(f"Loi khi gui lenh {command}: {e}")
                messagebox.showerror("Loi", f"Khong the gui lenh {command}: {e}")
                return False
        messagebox.showerror("Loi", f"ESP32 khong co phan hoi sau {max_attempts}
lan thu cho lenh {command}")
        print(f"ESP32 khong co phan hoi sau {max_attempts} lan thu cho lenh
{command}")
        return False

    # Hàm cập nhật trạng thái
    def update_status():
        try:
            if result_label and result_label.winfo_exists():
                result_label.config(text=capture_status)
        except Exception as e:
            print(f"Loi cap nhat trang thai: {e}")

    # Hàm cập nhật nhãn OK/NG
    def update_ok_ng_status(num_black_spots):
        try:
            if ok_ng_label and ok_ng_label.winfo_exists():
                if num_black_spots == 0:
                    ok_ng_label.config(text="OK", fg="green")
                else:
                    ok_ng_label.config(text="NG", fg="red")
        except Exception as e:
            print(f"Loi cap nhat trang thai OK/NG: {e}")

    # Hàm reset hệ thống
    def reset_system():
        global capture_status
        try:
            if send_command("reset"):
                capture_status = "Da reset servo ve vi tri goc"
```

```
                update_status()
            else:
                messagebox.showerror("Loi", "Khong nhan duoc phan hoi tu ESP32")
        except Exception as e:
            print(f"Loi reset he thong: {e}")
            messagebox.showerror("Loi", f"Loi reset he thong: {e}")

    # Hàm hiển thị hình ảnh trước đó
    def show_previous_image():
        global current_image_index
        try:
            if processed_images and current_image_index > 0:
                current_image_index -= 1
                img_pil =
Image.fromarray(cv2.imread(processed_images[current_image_index]))
                zi.display_image(img_pil)
                update_status()
        except Exception as e:
            print(f"Loi hien thi anh truoc: {e}")
            capture_status = f"Loi hien thi anh truoc: {e}"
            update_status()

    # Hàm hiển thị hình ảnh tiếp theo
    def show_next_image():
        global current_image_index
        try:
            if processed_images and current_image_index < len(processed_images) -
1:
                current_image_index += 1
                img_pil =
Image.fromarray(cv2.imread(processed_images[current_image_index]))
                zi.display_image(img_pil)
                update_status()
        except Exception as e:
            print(f"Loi hien thi anh tiep theo: {e}")
            capture_status = f"Loi hien thi anh tiep theo: {e}"
            update_status()

    # Hàm xử lý chế độ Auto
    def start_auto():
        global capture_status, processed_images, current_image_index,
arrow_frame, prev_button, next_button
        try:
            if not auto_mode:
                messagebox.showinfo("Thong bao", "Vui long chon che do Auto")
                return

            processed_images = []
            current_image_index = 0

            # Xóa arrow_frame nếu đã tồn tại
            if arrow_frame is not None:
                arrow_frame.destroy()

            # Gửi lệnh reset trước khi bắt đầu
            if not send_command("reset"):
                messagebox.showerror("Loi", "Khong nhan duoc phan hoi tu lenh
reset")
                capture_status = "Loi: Khong the reset san pham ve mat truoc"
                update_status()
                return
            capture_status = "Da reset san pham ve mat truoc"
            update_status()
            time.sleep(1)

            commands = ["matsau", "mattrai", "matphai", "mattren", "mattruoc"]
            results = []
            overall_ok = True
            for i, face in enumerate(faces):
                file_path = capture_image(face, current_product)
                if file_path:
                    result, num_black_spots = analyze_black_spots(file_path,
face)
                    results.append(result)
                    processed_images.append(file_path)
                    img_pil = Image.fromarray(cv2.imread(file_path))
```

```
                    zi.display_image(img_pil)
                    update_status()
                    update_ok_ng_status(num_black_spots)
                    if num_black_spots > 0:
                        overall_ok = False

                if i < len(commands):
                    if not send_command(commands[i]):
                        messagebox.showerror("Loi", f"Khong nhan duoc phan hoi
khi xoay {commands[i]}")
                        continue

            capture_status = "Da hoan thanh kiem tra\n" + "\n".join(results)
            update_status()
            update_ok_ng_status(0 if overall_ok else 1)

            if processed_images:
                arrow_frame = Frame(canvas_frame)
                arrow_frame.pack(side="top", fill="x", pady=10)

                prev_button = Button(arrow_frame, text="<",
command=show_previous_image, font=("Arial", 16), width=3,
                                     height=2)
                prev_button.pack(side="left", padx=20)

                next_button = Button(arrow_frame, text=">",
command=show_next_image, font=("Arial", 16), width=3, height=2)
                next_button.pack(side="right", padx=20)

                img_pil = Image.fromarray(cv2.imread(processed_images[0]))
                zi.display_image(img_pil)
        except Exception as e:
            print(f"Loi trong che do Auto: {e}")
            capture_status = f"Loi trong che do Auto: {e}"
            update_status()

    # Hàm xử lý chế độ Manual
    def capture_manual():
        global capture_status, manual_window
        try:
            if auto_mode:
                messagebox.showinfo("Thong bao", "Vui long chon che do Manual")
                return

            if manual_window is not None and manual_window.winfo_exists():
                manual_window.lift()
                return

            manual_window = Toplevel(root)
            manual_window.title("Chon mat de chup")
            manual_window.geometry("300x300")
            manual_window.protocol("WM_DELETE_WINDOW", on_manual_window_close)

            Label(manual_window, text="Chon mat can chup:", font=("Arial", 14,
"bold")).pack(pady=10)

            def capture_selected_face(face):
                try:
                    command = f"mat{face.lower()}"
                    if send_command(command):
                        file_path = capture_image(face, current_product)
                        if file_path:
                            result, num_black_spots =
analyze_black_spots(file_path, face)
                            img_pil = Image.fromarray(cv2.imread(file_path))
                            zi.display_image(img_pil)
                            capture_status = result
                            update_status()
                            update_ok_ng_status(num_black_spots)
                        manual_window.destroy()
                    else:
                        messagebox.showerror("Loi", f"Khong nhan duoc phan hoi
khi xoay {face}")
                except Exception as e:
                    print(f"Loi chup mat {face}: {e}")
                    messagebox.showerror("Loi", f"Loi chup mat {face}: {e}")

```

```
            for face in faces:
                Button(manual_window, text=face, font=("Arial", 12), width=15,
                       command=lambda f=face:
capture_selected_face(f)).pack(pady=5)
        except Exception as e:
            print(f"Loi trong che do Manual: {e}")
            messagebox.showerror("Loi", f"Loi trong che do Manual: {e}")

    def on_manual_window_close():
        global manual_window
        try:
            manual_window.destroy()
            manual_window = None
        except Exception:
            pass

    # Hàm vẽ đa giác
    def draw_polygon(img, num_sides):
        try:
            img_copy = img.copy()
            points = []
            window_name = "Ve da giac - Nhan chuot de them diem, nhan Enter de
hoan thanh"

            def mouse_callback(event, x, y, flags, param):
                nonlocal points, img_copy
                if event == cv2.EVENT_LBUTTONDOWN and len(points) < num_sides:
                    points.append([x, y])
                    cv2.circle(img_copy, (x, y), 5, (0, 255, 0), -1)
                    if len(points) > 1:
                        cv2.line(img_copy, tuple(points[-2]), tuple(points[-1]),
(0, 255, 0), 2)
                    cv2.imshow(window_name, img_copy)

            cv2.namedWindow(window_name)
            cv2.setMouseCallback(window_name, mouse_callback)
            cv2.imshow(window_name, img_copy)

            while True:
                key = cv2.waitKey(1) & 0xFF
                if key == 13 and len(points) == num_sides:
                    cv2.line(img_copy, tuple(points[-1]), tuple(points[0]), (0,
255, 0), 2)
                    cv2.imshow(window_name, img_copy)
                    cv2.waitKey(1)
                    break
                elif key == 27:
                    points = []
                    break
            cv2.destroyWindow(window_name)
            return points
        except Exception as e:
            print(f"Loi ve da giac: {e}")
            return []

    # Hàm cài đặt vùng cắt
    face_window = None

    def set_crop_for_face():
        global face_window
        try:
            if face_window is not None and face_window.winfo_exists():
                face_window.lift()
                return

            def choose_face(chosen_face):
                selected_face.set(chosen_face)
                num_sides_window = Toplevel(face_window)
                num_sides_window.title("Nhap so canh cua da giac")
                num_sides_window.geometry("300x150")
                Label(num_sides_window, text="Nhap so canh (toi thieu 3):",
font=("Arial", 12)).pack(pady=5)
                num_sides_entry = Entry(num_sides_window, font=("Arial", 12))
                num_sides_entry.pack(pady=5)

                def confirm_num_sides():
                    try:
```

```
                        num_sides = int(num_sides_entry.get())
                        if num_sides < 3:
                            messagebox.showerror("Loi", "So canh phai lon hon
hoac bang 3!")
                            return
                        num_sides_window.destroy()
                        file_path = filedialog.askopenfilename(filetypes=[("Image
files", "*.jpg *.png *.jpeg *.bmp")])
                        if not file_path:
                            return
                        img = cv2.imread(file_path)
                        if img is None:
                            messagebox.showerror("Loi", "Khong the doc file anh")
                            return
                        img = cv2.resize(img, (800, 800))
                        points = draw_polygon(img, num_sides)
                        if points:
                            crop_coords[selected_face.get()] = points
                            with open(CROP_CONFIG_FILE, "w") as f:
                                json.dump(crop_coords, f)
                            face_window.destroy()
                    except ValueError:
                        messagebox.showerror("Loi", "Vui long nhap mot so nguyen
hop le!")
                    except Exception as e:
                        print(f"Loi chon so canh: {e}")
                        messagebox.showerror("Loi", f"Loi chon so canh: {e}")

                Button(num_sides_window, text="Xac nhan",
command=confirm_num_sides, font=("Arial", 12), bg="#4CAF50",
                       fg="white", width=12).pack(pady=5)

            face_window = Toplevel()
            face_window.title("Chon mat can cai dat vung cat")
            face_window.geometry("300x200")
            face_window.protocol("WM_DELETE_WINDOW", lambda:
on_face_window_close())
            for face in faces:
                Button(face_window, text=face, font=("Arial", 12), width=20,
command=lambda f=face: choose_face(f)).pack(pady=5)
        except Exception as e:
            print(f"Loi cai dat vung cat: {e}")
            messagebox.showerror("Loi", f"Loi cai dat vung cat: {e}")

    def on_face_window_close():
        global face_window
        try:
            face_window.destroy()
            face_window = None
        except Exception:
            pass

    # Hàm reset tọa độ cắt
    def reset_crop_coords():
        global crop_coords
        try:
            crop_coords.clear()
            if os.path.exists(CROP_CONFIG_FILE):
                os.remove(CROP_CONFIG_FILE)
            messagebox.showinfo("Thong bao", "Da reset toa do cat")
        except Exception as e:
            print(f"Loi reset toa do cat: {e}")
            messagebox.showerror("Loi", f"Loi reset toa do cat: {e}")

    # Hàm xóa nền bằng rembg
    def remove_background_with_rembg(img):
        try:
            img_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
            result_pil = remove(img_pil)
            result_np = np.array(result_pil)
            if result_np.shape[2] == 4:
                alpha = result_np[:, :, 3]
                mask = np.where(alpha > 0, 1, 0).astype(np.uint8)
            else:
                mask = np.ones(img.shape[:2], dtype=np.uint8)
            return mask
        except Exception as e:
```

```
            print(f"Loi xoa nen: {e}")
            return np.ones(img.shape[:2], dtype=np.uint8)

    # Lớp ZoomableImage
    class ZoomableImage:
        def __init__(self, canvas):
            self.canvas = canvas
            self.canvas.bind("<ButtonPress-1>", self.start_move)
            self.canvas.bind("<B1-Motion>", self.do_move)
            self.canvas.bind("<MouseWheel>", self.do_zoom)
            self.image_id = None
            self.scale = 1.0
            self.img_tk = None
            self.start_x = self.start_y = 0

        def display_image(self, img_pil):
            try:
                self.scale = 1.0
                self.img_pil = img_pil
                self.update_image()
            except Exception as e:
                print(f"Loi hien thi anh: {e}")

        def update_image(self):
            try:
                width, height = self.img_pil.size
                new_size = int(width * self.scale), int(height * self.scale)
                img_resized = self.img_pil.resize(new_size)
                self.img_tk = ImageTk.PhotoImage(img_resized)
                if self.image_id:
                    self.canvas.delete(self.image_id)
                self.image_id = self.canvas.create_image(0, 0, anchor="nw",
image=self.img_tk)
                self.canvas.config(scrollregion=self.canvas.bbox(self.image_id))
            except Exception as e:
                print(f"Loi cap nhat anh: {e}")

        def do_zoom(self, event):
            try:
                factor = 1.1 if event.delta > 0 else 0.9
                self.scale *= factor
                self.update_image()
            except Exception as e:
                print(f"Loi zoom: {e}")

        def start_move(self, event):
            try:
                self.start_x = event.x
                self.start_y = event.y
            except Exception as e:
                print(f"Loi start_move: {e}")

        def do_move(self, event):
            try:
                dx = event.x - self.start_x
                dy = event.y - self.start_y
                self.canvas.move(self.image_id, dx, dy)
                self.start_x = event.x
                self.start_y = event.y
            except Exception as e:
                print(f"Loi do_move: {e}")

    # Hàm chuyển chế độ
    def toggle_mode():
        global auto_mode
        try:
            auto_mode = not auto_mode
            mode_button.config(text=f"Chuyen che do: {'Auto' if auto_mode else
'Manual'}")
        except Exception as e:
            print(f"Loi chuyen che do: {e}")

    # Hàm giao diện ban đầu
    def show_initial_interface():
        global root, selected_face
        try:
            if root is not None and root.winfo_exists():
```

```
                root.destroy()
            if manual_root is not None and manual_root.winfo_exists():
                manual_root.destroy()
        except Exception:
            pass
        root = tk.Tk()
        selected_face = tk.StringVar()  # Khởi tạo selected_face khi có root
        root.title("Chon phuong phap chup")
        root.geometry("400x300")
        Label(root, text="Chon phuong phap chup:", font=("Arial", 14,
"bold")).pack(pady=20)
        Button(root, text="Chay thu cong", command=show_manual_mode,
font=("Arial", 12), bg="#4CAF50", fg="white",
               width=20).pack(pady=10)
        Button(root, text="Chay tu dong", command=show_product_selection,
font=("Arial", 12), bg="#2196F3", fg="white",
               width=20).pack(pady=10)

        root.mainloop()

    # Hàm giao diện chế độ thủ công
    def show_manual_mode():
        global manual_root, root, result_label, zi, selected_image_path
        try:
            if root is not None and root.winfo_exists():
                root.destroy()
        except Exception:
            pass
        manual_root = tk.Tk()
        manual_root.title("Che do chup thu cong")
        manual_root.geometry("1200x800")

        main_frame = Frame(manual_root)
        main_frame.pack(fill="both", expand=True, padx=10, pady=10)

        left_frame = Frame(main_frame, bd=2, relief="groove", width=300)
        left_frame.pack(side="left", fill="y", padx=5)

        def capture_manual_image():
            global selected_image_path
            file_path = capture_image()
            if file_path:
                img_pil = Image.fromarray(cv2.imread(file_path))
                zi.display_image(img_pil)
                selected_image_path = file_path

        def select_image():
            global selected_image_path
            file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg *.png *.jpeg")])
            if file_path:
                img_pil = Image.fromarray(cv2.imread(file_path))
                zi.display_image(img_pil)
                selected_image_path = file_path
                capture_status = f"Da chon anh: {file_path}"
                update_status()

        def set_manual_crop():
            global selected_image_path
            if not selected_image_path:
                messagebox.showerror("Loi", "Vui long chon mot anh truoc!")
                return
            img = cv2.imread(selected_image_path)
            if img is None:
                messagebox.showerror("Loi", "Khong the doc file anh")
                return
            img = cv2.resize(img, (800, 800))
            num_sides_window = Toplevel(manual_root)
            num_sides_window.title("Nhap so canh cua da giac")
            num_sides_window.geometry("300x150")
            Label(num_sides_window, text="Nhap so canh (toi thieu 3):",
font=("Arial", 12)).pack(pady=5)
            num_sides_entry = Entry(num_sides_window, font=("Arial", 12))
            num_sides_entry.pack(pady=5)

            def confirm_num_sides():
                try:
```

```
641.                     num_sides = int(num_sides_entry.get())
642.                     if num_sides < 3:
643.                         messagebox.showerror("Loi", "So canh phai lon hon hoac
bang 3!")
644.                         return
645.                     num_sides_window.destroy()
646.                     points = draw_polygon(img, num_sides)
647.                     if points:
648.                         crop_coords[manual_crop_name] = points
649.                         with open(CROP_CONFIG_FILE, "w") as f:
650.                             json.dump(crop_coords, f)
651.                         capture_status = "Da luu khung cat cho che do thu cong"
652.                         update_status()
653.                 except ValueError:
654.                     messagebox.showerror("Loi", "Vui long nhap mot so nguyen hop
le!")
655.                 except Exception as e:
656.                     print(f"Loi chon so canh: {e}")
657.                     messagebox.showerror("Loi", f"Loi chon so canh: {e}")
658.
659.             Button(num_sides_window, text="Xac nhan", command=confirm_num_sides,
font=("Arial", 12)).pack(pady=5)
660.
661.         def analyze_manual_image():
662.             global selected_image_path
663.             if not selected_image_path:
664.                 messagebox.showerror("Loi", "Vui long chon mot anh truoc!")
665.                 return
666.             if manual_crop_name not in crop_coords:
667.                 messagebox.showerror("Loi", "Vui long thiet lap khung cat
truoc!")
668.                 return
669.             result, num_black_spots = analyze_black_spots(selected_image_path,
manual_crop_name)
670.             img_pil = Image.fromarray(cv2.imread(selected_image_path))
671.             zi.display_image(img_pil)
672.             capture_status = result
673.             update_status()
674.             update_ok_ng_status(num_black_spots)
675.
676.         Button(left_frame, text="Chup anh", command=capture_manual_image,
font=("Arial", 12), bg="#4CAF50", fg="white",
677.                width=20).pack(pady=5)
678.         Button(left_frame, text="Chon anh", command=select_image, font=("Arial",
12), bg="#2196F3", fg="white",
679.                width=20).pack(pady=5)
680.         Button(left_frame, text="Chon khung cat", command=set_manual_crop,
font=("Arial", 12), bg="#FF9800", fg="white",
681.                width=20).pack(pady=5)
682.         Button(left_frame, text="Phan tich anh", command=analyze_manual_image,
font=("Arial", 12), bg="#9C27B0", fg="white",
683.                width=20).pack(pady=5)
684.         Button(left_frame, text="Quay lai", command=lambda:
[manual_root.destroy(), show_initial_interface()],
685.                font=("Arial", 12), bg="#F44336", fg="white",
width=20).pack(pady=5)
686.
687.         right_frame = Frame(main_frame, bd=2, relief="groove", width=300)
688.         right_frame.pack(side="right", fill="y", padx=5)
689.         global result_label
690.         result_label = Label(right_frame, text="Chua co trang thai",
font=("Arial", 13), fg="blue", wraplength=280,
691.                              justify="left")
692.         result_label.pack(pady=10)
693.
694.         canvas_frame = Frame(main_frame)
695.         canvas_frame.pack(side="top", fill="both", expand=True)
696.         canvas = Canvas(canvas_frame, bg="gray", height=600)
697.         canvas.pack(side="left", fill="both", expand=True)
698.         global zi
699.         zi = ZoomableImage(canvas)
700.
701.         manual_root.mainloop()
702.
703.     # Hàm giao diện chọn sản phẩm
704.     def show_product_selection():
705.         global root
```

```
        try:
            if root is not None and root.winfo_exists():
                root.destroy()
            if manual_root is not None and manual_root.winfo_exists():
                manual_root.destroy()
        except Exception:
            pass
        product_window = tk.Tk()
        product_window.title("Chon loai san pham")
        product_window.geometry("400x300")
        Label(product_window, text="Chon loai san pham:", font=("Arial", 14,
"bold")).pack(pady=10)

        def select_product(product):
            global current_product
            current_product = product
            product_window.destroy()
            show_auto_mode()

        for product in products:
            Button(product_window, text=product, font=("Arial", 12), width=15,
                   command=lambda p=product: select_product(p)).pack(pady=10)

        product_window.mainloop()

    # Hàm giao diện chế độ tự động
    def show_auto_mode():
        global root, mode_button, result_label, canvas_frame, zi, ok_ng_label
        try:
            if manual_root is not None and manual_root.winfo_exists():
                manual_root.destroy()
        except Exception:
            pass
        root = tk.Tk()
        root.title(f"He thong chup anh va phat hien diem den -
{current_product}")
        root.geometry("1200x1000")

        main_frame = Frame(root)
        main_frame.pack(fill="both", expand=True, padx=10, pady=10)

        left_frame = Frame(main_frame, bd=2, relief="groove", width=300)
        left_frame.pack(side="left", fill="y", padx=5)
        global mode_button
        mode_button = Button(left_frame, text="Chuyen che do: Auto",
font=("Arial", 12), bg="#2196F3", fg="white", width=20,
                             command=toggle_mode)
        mode_button.pack(pady=5)
        Button(left_frame, text="Reset", font=("Arial", 12), bg="#F44336",
fg="white", width=20, command=reset_system).pack(
            pady=5)
        Button(left_frame, text="Start Auto", font=("Arial", 12), bg="#4CAF50",
fg="white", width=20,
               command=start_auto).pack(pady=5)
        Button(left_frame, text="Chup Manual", font=("Arial", 12), bg="#FF9800",
fg="white", width=20,
               command=capture_manual).pack(pady=5)
        Button(left_frame, text="Quay lai", font=("Arial", 12), bg="#9C27B0",
fg="white", width=20,
               command=lambda: [root.destroy(),
show_initial_interface()]).pack(pady=5)

        right_frame = Frame(main_frame, bd=2, relief="groove", width=300)
        right_frame.pack(side="right", fill="y", padx=5)
        global result_label
        result_label = Label(right_frame, text="Chua co trang thai",
font=("Arial", 13), fg="blue", wraplength=280,
                             justify="left")
        result_label.pack(pady=10)
        global ok_ng_label
        ok_ng_label = Label(right_frame, text="Chua kiem tra", font=("Arial", 13,
"bold"), fg="black", wraplength=280)
        ok_ng_label.pack(pady=10)
        Button(right_frame, text="Thay doi toa do cat", font=("Arial", 12),
bg="#4CAF50", fg="white", width=20,
               command=set_crop_for_face).pack(pady=5)
```

```
        Button(right_frame, text="Reset toa do cat", font=("Arial", 12),
bg="#F44336", fg="white", width=20,
                command=reset_crop_coords).pack(pady=5)

        global canvas_frame
        canvas_frame = Frame(main_frame)
        canvas_frame.pack(side="top", fill="both", expand=True)

        global arrow_frame
        arrow_frame = Frame(canvas_frame)
        arrow_frame.pack(side="top", fill="x", pady=5)

        global prev_button
        prev_button = Button(arrow_frame, text="<", command=show_previous_image,
font=("Arial", 16), width=3, height=2)
        prev_button.pack(side="left", padx=20)

        canvas = Canvas(canvas_frame, bg="gray", height=700)
        canvas.pack(side="left", fill="both", expand=True)

        global next_button
        next_button = Button(arrow_frame, text=">", command=show_next_image,
font=("Arial", 16), width=3, height=2)
        next_button.pack(side="right", padx=20)

        global zi
        zi = ZoomableImage(canvas)

        root.mainloop()

    # Chạy chương trình
    try:
        show_initial_interface()
    finally:
        try:
            if ser and ser.is_open():
                ser.close()
        except Exception as e:
            print(f"Loi dong serial: {e}")
```

Chương điều khiển động cơ của ESP32

```
#include <ESP32Servo.h>

#define STEP_PIN 12
#define DIR_PIN 14
#define ENDSTOP_PIN 16  // Chân D16 để đọc tín hiệu công tắc giới hạn
const int stepsPerRevolution = 20;  // Số bước cho 1 vòng quay (điều chỉnh
theo động cơ)
int stepDelayMicros = 700;         // Thời gian delay giữa các bước (µs)
int z,zmt,zht,dir;

Servo servo1;
Servo servo2;


void setup() {
  Serial.begin(9600);

  // Cấu hình chân điều khiển động cơ bước
  pinMode(STEP_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);
  pinMode(ENDSTOP_PIN, INPUT);  // Cấu hình D16 làm input

  // Phân bổ timer cho ESP32 (nên dùng khi điều khiển nhiều servo)
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);

  // Cài đặt tần số PWM và gắn servo vào chân GPIO 26 và 27
  servo1.setPeriodHertz(50);  // 50Hz cho servo
  servo1.attach(26, 500, 2400);  // Chân GPIO 26, xung min/max
  servo2.setPeriodHertz(50);
  servo2.attach(27, 500, 2400);  // Chân GPIO 27
```

```
31.
32.       // Đặt vị trí mặc định
33.       servo1.write(180);
34.       servo2.write(90);
35.     }
36.
37.     void loop() {
38.       if (Serial.available() > 0) {
39.         String command = Serial.readStringUntil('\n');
40.         command.trim();
41.
42.         // Kiểm tra lệnh dạng "x y" (hướng và số vòng)
43.         int spaceIndex = command.indexOf(' ');
44.         if (spaceIndex != -1) {
45.           String dirStr = command.substring(0, spaceIndex);
46.           String roundsStr = command.substring(spaceIndex + 1);
47.
48.           // Chuyển đổi hướng và số vòng
49.           int dir = dirStr.toInt();
50.           float rounds = roundsStr.toFloat();
51.
52.           // Kiểm tra hướng hợp lệ (0 hoặc 1) và số vòng hợp lệ
53.           if ((dir == 0 || dir == 1) && rounds > 0) {
54.             stepMotor(dir, rounds);
55.           } else {
56.             Serial.println("Lệnh không hợp lệ. Vui lòng nhập 'x y' với x là 0
      hoặc 1, y là số vòng.");
57.           }
58.         }
59.         // Xử lý các lệnh hiện có
60.         else if (command == "reset") {
61.           servo1.write(180);
62.           servo2.write(90);
63.           resetMotor();
64.           zht=0;
65.           stepMotor(1, 15);
66.           zht=15;
67.         } else if (command == "matsau") {
68.           zmt=15;
69.           z=zmt-zht;
70.           if(z>=0){dir=1;}else{dir=0;z=-z;}
71.           servo1.write(0);
72.           servo2.write(90);
73.           stepMotor(dir,z);
74.           zht=15;
75.
76.         } else if (command == "mattrai") {
77.           zmt=45;
78.           z=zmt-zht;
79.           if(z>=0){dir=1;}else{dir=0;z=-z;}
80.           servo1.write(180);
81.           servo2.write(180);
82.           stepMotor(dir,z);
83.           zht=45;
84.
85.         } else if (command == "matphai") {
86.           zmt=45;
87.           z=zmt-zht;
88.           if(z>=0){dir=1;}else{dir=0;z=-z;}
89.           servo1.write(180);
90.           servo2.write(0);
91.           stepMotor(dir,z);
92.           zht=45;
93.
94.         } else if (command == "mattren") {
95.           zmt=22;
96.           z=zmt-zht;
97.           if(z>=0){dir=1;}else{dir=0;z=-z;}
98.           servo1.write(90);
99.           servo2.write(90);
100.          stepMotor(dir, z);
101.          zht=22;
102.
103.
104.        } else if (command == "mattruoc") {
105.          zmt=15;
```

```
        z=zmt-zht;
        if(z>=0){dir=1;}else{dir=0;z=-z;}
        servo1.write(180);
        servo2.write(90);
        stepMotor(dir, z);
        zht=15;

      } else {
        Serial.println("Lệnh không hợp lệ. Các lệnh hợp lệ: reset, matsau,
mattrai, matphai, mattren, mattruoc, hoặc 'x y'.");
      }

      delay(100);  // Chờ servo di chuyển xong
      Serial.println("done");
    }
  }

  void stepMotor(bool dir, float numRounds) {
    int steps = (int)(numRounds * stepsPerRevolution);
    digitalWrite(DIR_PIN, dir ? HIGH : LOW);

    for (int i = 0; i < steps; i++) {
      digitalWrite(STEP_PIN, HIGH);
      delayMicroseconds(stepDelayMicros);
      digitalWrite(STEP_PIN, LOW);
      delayMicroseconds(stepDelayMicros);
    }

    Serial.print("Đã quay ");
    Serial.print(numRounds);
    Serial.print(" vòng, chiều: ");
    Serial.println(dir ? "thuận" : "ngược");
  }

  void resetMotor() {
    digitalWrite(DIR_PIN, LOW);  // Đặt chiều quay ngược (dir=0)

    while (digitalRead(ENDSTOP_PIN) == LOW) {  // Quay cho đến khi D16 = HIGH
      digitalWrite(STEP_PIN, HIGH);
      delayMicroseconds(stepDelayMicros);
      digitalWrite(STEP_PIN, LOW);
      delayMicroseconds(stepDelayMicros);
    }

    Serial.println("Động cơ bước đã reset, dừng tại công tắc giới hạn");
  }
```