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:

NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)

Người hướng dẫn 1: **TS. NGUYỄN THỊ KIM TRÚC**
Người hướng dẫn 2: **NGUYỄN VĂN CƯỜNG**
Sinh viên thực hiện:
PHAN VĂN KHẢI – MSSV: 105200496 – LỚP: 20TDHCLC4

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)
Sinh viên thực hiện:
Phan Văn Khải Số thẻ SV: 105200496 Lớp: 20TDHCLC4

Trước khi áp dụng, việc kiểm tra lỗi trên bo mạch đã gắn linh kiện (PCBA) được thực hiện thủ công, tốn thời gian, ảnh hưởng mắt, đạt độ chính xác thấp và phụ thuộc vào kỹ năng công nhân. Đồ án này đề xuất một hệ thống tự động phát hiện lỗi trên PCBA (Printed Circuit Board Assembly) sau sản xuất, sử dụng mô hình học sâu YOLOv8. Hệ thống thích hợp phần cứng gồm kính hiển vi công nghiệp, cáp HDMI video capture, bộ chuyển động servo điều khiển bởi ESP32, và phần mềm chạy trên laptop/PC. Hệ thống hỗ trợ hai chế độ:

- **AUTO** (sử dụng bộ chuyển động để định vị PCBA vào vùng quan sát)
- **MANUAL** (bỏ bộ chuyển động, công nhân tự đặt PCBA và di chuyển bằng tay).

Quy trình hoạt động: công nhân đặt PCBA, nhấn nút (ở chế độ AUTO) hoặc điều chỉnh thủ công (MANUAL), YOLOv8 phân tích và hiển thị Real time các lỗi nếu có. Thực hiện đạt độ chính xác trên 90%, vượt trội so với phương pháp thủ công. Hệ thống cung cấp giải pháp tự động hóa kiểm tra PCBA hiệu quả, chi phí hợp lý, với tiềm năng ứng dụng công nghiệp. Hướng phát triển bao gồm mở rộng tập dữ liệu, nâng cấp phần cứng và tích hợp công nghệ để phát hiện lỗi phức tạp hơn.

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

Họ tên sinh viên: Phan Văn Khải Số thẻ sinh viên: 105200496
Lớp: 20TDHCLC4 Khoa: Điện Ngành: Kỹ thuật Điều khiển & Tự động hóa

1. *Tên đề tài đồ án:*

NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)

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:*

- Tập dữ liệu gồm 200 ảnh chụp thực tế từ bo mạch PCBA tại công ty Merry & Luxshare-ICT.
- Các ảnh được gán nhãn thủ công trên nền tảng roboflow.com với 2 loại lỗi:
  - Mất linh kiện
  - Nứt/vỡ linh kiện IC
- Cấu hình phần cứng ban đầu:
  - Kính hiển vi công nghiệp có độ phân giải 4K
  - Cáp HDMI video capture tốc độ cao (30 FPS, độ trễ <100ms)
  - Vi điều khiển ESP32-WROOM-32, điều khiển 5 động cơ servo
- Mô hình YOLOv8m đã được fine-tune bằng thư viện Ultralytics trên GPU NVIDIA

4. *Nội dung các sản phẩm thuyết minh và tính toán:*

- Phân tích và đánh giá quy trình kiểm tra lỗi PCBA hiện tại trong nhà máy
- Tổng quan về mô hình học sâu YOLOv8 và lý do lựa chọn mô hình
- Thiết kế hệ thống phần cứng: sơ đồ mạch, cấu trúc servo, kết nối ESP32
- Xây dựng giao diện phần mềm nhận dạng lỗi, tương tác với ESP32 qua UART
- Thu thập dữ liệu thực tế, gán nhãn lỗi và huấn luyện mô hình YOLOv8
- Đánh giá kết quả mô hình bằng các chỉ số: Precision, Recall, mAP@0.5
- Tính toán góc quay servo từ xung PWM để đảm bảo sai số nhỏ hơn 0.2mm
- Tính toán thời gian xử lý trung bình và khả năng đáp ứng yêu cầu sản xuất

5. *Các bản vẽ, đồ thị:*
   - Sơ đồ tổng thể hệ thống gồm các khối chức năng: camera, ESP32, máy tính
   - Sơ đồ mạch điều khiển ESP32 kết nối servo và nút nhấn
   - Thiết kế mô phỏng 3D cơ cấu chuyển động servo
   - Lưu đồ thuật toán hoạt động của hệ thống (Auto & Manual)
   - Đồ thị huấn luyện YOLOv8: train loss, val loss, precision, recall theo epoch
   - Ảnh kết quả bounding box phát hiện lỗi thực tế (ảnh lỗi và ảnh đạt)
6. *Họ tên người hướng dẫn 1: Nguyễn Thị Kim Trúc*
7. *Họ tên người hướng dẫn 2: Nguyễn Văn Cường*
8. *Ngày giao nhiệm vụ đồ án: 03/03/2025*
9. *Ngày hoàn thành đồ án: 02/06/2025*

Đà Nẵng, ngày 17 tháng 06 năm 2025

Người hướng dẫn 1

Trưởng Bộ môn

Người hướng dẫn 2

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: Phan Văn Khải Số thẻ SV: 105200496
Tên đề tài ĐATN: NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)
Họ tên người HD1: Nguyễn Thị Kim Trúc Đơn vị: Khoa Điện
Họ tên người HD2: Nguyễn Văn Cường Đơn vị: Công ty Merry & Luxshare

Tuần	Ngày	Khối lượng		GVHD ký tên
		đã thực hiện (%)	tiếp tục thực hiện (%)	
1	03/03	Nhận đề tài	Trao đổi với người hướng dẫn định hướng đồ án	
2	10/03	Tìm hiểu đề tài, khảo sát tại nhà máy	Thu thập tài liệu	
3	17/03	Phân tích lỗi PCBA, tìm hiểu công nghệ ESP32 và kính hiển vi	Thiết kế hệ thống sơ bộ, sơ đồ kết nối ESP32 và servo	
4	24/03	Duyệt lần 1: Đánh giá khối lượng hoàn thành ___%: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	31/03	Lên kiến trúc tổng thể hệ thống	Thiết kế mô hình cơ khí servo và giao tiếp UART với PC	
6	07/04	Viết phần mềm xử lý ảnh từ kính hiển vi	Viết phần mềm nhận lệnh từ GUI đến ESP32	
7	14/04	Thu thập dữ liệu PCBA	Gán nhãn dữ liệu, khởi tạo tập train/val/test trên roboflow	
8	21/04	Duyệt lần 2: Đánh giá khối lượng hoàn thành ___%: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	28/04	Mô hình AI đã huấn luyện cơ bản	Chạy thử YOLOv8m, kiểm tra tính hoàn thiện từng chế độ AUTO/MANUAL	
10	05/05	Đánh giá precision, recall, mAP	Hoàn thiện giao diện người dùng	
11	12/05	Kiểm tra phần cứng	Hiệu chỉnh sai số phần cứng	
12	19/05	Duyệt lần 3: Đánh giá khối lượng hoàn thành ___%: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	26/05	Tổng hợp báo cáo và chỉnh sửa	Chuẩn bị bảo vệ	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trong môi trường sản xuất công nghiệp, đặc biệt là trong lĩnh vực điện tử, những sản phẩm mới, công nghệ mới chưa được tung ra thị trường thường có tính bảo mật cao, mọi đề xuất cải tiến hay ứng dụng công nghệ mới đều cần được cân nhắc kỹ lưỡng về tính khả thi, chi phí đầu tư và hiệu quả mang lại. Để một giải pháp được doanh nghiệp chấp nhận, người đề xuất không chỉ cần đưa ra ý tưởng, mà còn phải minh chứng được lợi ích cụ thể mà giải pháp đó đem lại thông qua các bản demo thực tế, các chỉ số đánh giá rõ ràng và dự toán chi phí hợp lý.

Do đặc thù của quy trình sản xuất, doanh nghiệp thường chỉ đầu tư trong phạm vi rủi ro có thể chấp nhận được — nhằm đảm bảo nếu giải pháp không đạt hiệu quả như kỳ vọng để lộ thông tin bảo mật thì thiệt hại cũng không quá lớn. Chính vì vậy, trong đề tài này, em đã lựa chọn thực hiện một bản demo trên loại mạch PCBA đơn giản, có cấu trúc dễ tiếp cận nhưng vẫn mô phỏng đầy đủ các vấn đề thực tế cần khắc phục.

Cụ thể, sau công đoạn kiểm tra chức năng của PCBA (kiểm tra nút nhấn, điện áp, sạc điện v.v, đa số lỗi trong quá trình kiểm tra chức năng thường do thao tác đặt mạch vào khuông test bị sai sót dẫn đến kim đâm sai vị trí hoặc nguyên nhân nào đó làm **vỡ, nứt nẻ linh kiện**, hoặc **linh kiện bị bong mất khỏi vị trí hàn**. Đây là những lỗi rất khó phát hiện bằng mắt thường nếu không sử dụng kính hiển vi công nghiệp, và quá trình kiểm tra thủ công hiện nay còn phụ thuộc nhiều vào thao tác và kinh nghiệm của công nhân, dẫn đến sai sót và tốn thời gian.

Với mong muốn cải thiện công đoạn kiểm tra ngoại quan này, em đã xây dựng mô hình phát hiện lỗi tự động sử dụng học sâu (Deep Learning), có khả năng **khoanh vùng lỗi trực tiếp trên ảnh mạch PCBA và hiển thị thời gian thực** qua màn hình. Mô hình được đào tạo và huấn luyện dựa trên YOLOv8 dựa trên những loại lỗi và vị trí lỗi cụ thể, giúp tiết kiệm công sức phù hợp với điều kiện thực tế tại nhà máy.

Tôi xin bày tỏ lòng biết ơn chân thành đến quý thầy cô, công ty, anh đồng hướng dẫn, bạn bè, và gia đình đã hỗ trợ tôi trong quá trình thực hiện đồ án tốt nghiệp này.

Trước tiên, tôi xin gửi lời cảm ơn sâu sắc đến **Cô Nguyễn Thị Kim Trúc**, người đã tận tình hướng dẫn, cung cấp những ý kiến chuyên môn quý báu vào động viên tôi hoàn thiện đồ án "NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)".

Tôi xin cảm ơn **Công ty MERRY & LUXSHARE-ICT** đã tạo điều kiện thuận lợi, cung cấp môi trường thực tế và hỗ trợ trang thiết bị để tôi thực hiện đồ án. Đặc biệt, tôi xin

tri ân **Anh Nguyễn Văn Cường** đã tận tình hướng dẫn, chia sẻ kinh nghiệm thực tiễn và hỗ trợ tôi trong quá trình triển khai hệ thống.

Tôi cũng xin cảm ơn các thầy cô trong Khoa Điện và Trường Đại Học Bách Khoa Đà Nẵng, những người đã trang bị cho tôi nền tảng kiến thức vững chắc và tạo điều kiện để tôi hoàn thành đồ án.

Bên cạnh đó, tôi xin cảm ơn bạn bè, đồng nghiệp đã đồng hành, hỗ trợ thu thập dữ liệu, thử nghiệm hệ thống và động viên tinh thần trong suốt quá trình thực hiện.

Cuối cùng, tôi xin tri ân gia đình, những người luôn là chỗ dựa vững chắc, động viên và tạo điều kiện để tôi hoàn thành đồ án này.

Mặc dù đã nỗ lực hết sức, đồ án không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được những ý kiến đóng góp để hoàn thiện hơn.

Trân trọng,

Phan Văn Khải

Ngày tháng 06 năm 2025

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Tôi xin cam đoan rằng đồ án tốt nghiệp với đề tài "NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG KIỂM TRA LỖI MẠCH ĐIỆN TỬ ĐÃ GIA CÔNG (PCBA)" là công trình nghiên cứu do chính tôi thực hiện dưới sự hướng dẫn của **Cô Nguyễn Thị Kim Trúc** và sự hỗ trợ của **Anh Nguyễn Văn Cường** tại **Công ty MERRY & LUXSHARE-ICT**.

Nội dung báo cáo này được thực hiện dựa trên quá trình nghiên cứu, thiết kế, và thử nghiệm của tôi. Các số liệu, kết quả thực nghiệm, và nội dung trình bày trong báo cáo là trung thực, không sao chép từ bất kỳ nguồn nào mà không được trích dẫn rõ ràng. Tôi chịu hoàn toàn trách nhiệm về tính trung thực và chính xác của các nội dung trong đồ án.

Tôi cũng cam kết rằng báo cáo này chưa từng được nộp cho bất kỳ tổ chức, cá nhân, hoặc chương trình đào tạo nào khác.

Mọi ý kiến đóng góp để hoàn thiện đồ án đều được tôi trân trọng đón nhận.

Đà Nẵng, ngày tháng 06 năm 2025

Sinh viên thực hiện

Phan Văn Khải

MỤC LỤC

TÓM TẮT i
NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP ii
PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP iv
LỜI NÓI ĐẦU VÀ CẢM ƠN v
LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT vii
MỤC LỤC viii
DANH SÁCH CÁC BẢNG, HÌNH VẼ xi
Chương 1: MỞ ĐẦU 1
1.1. Lý do chọn đề tài 1
1.2. Vấn đề nghiên cứu 1
1.3. Mục tiêu của đề tài 2
1.4. Phạm vi và đối tượng nghiên cứu, phương pháp nghiên cứu 3
1.5. Cơ cấu đồ án được chia thành 5 chương 4
Chương 2: TỔNG QUAN VỀ CÁC NGHIÊN CỨU VÀ CÔNG NGHỆ TRONG BỐI CẢNH HIỆN TẠI 5
2.1. Kỹ thuật phát hiện lỗi PCBA 5
2.1.1. Kiểm tra thủ công 5
2.1.2. Hệ thống kiểm tra tự động truyền thống (AOI) 6
2.2. Giới thiệu và phân tích mô hình YOLOv8 8
2.2.1. Tầm nhìn máy tính và vai trò trong công nghiệp 8
2.2.2. Sự phát triển của học sâu trong tầm nhìn máy tính 8
2.3. Giới thiệu và phân tích mô hình YOLOv8 9
2.3.1. Tổng quan về YOLOv8 9
2.3.2. Hàm mất mát của YOLOv8 10
2.3.3. Ứng dụng YOLOv8 trong công nghiệp 11
2.3.4. Lý do lựa chọn YOLOv8 cho đề tài 12
2.4. So sánh YOLOv8 với các phương pháp khác 12
2.5. Kính hiển vi công nghiệp và video capture 13
2.5.1. Kính hiển vi công nghiệp 13
2.5.2. Cáp HDMI video capture 14

2.6. Bộ điều khiển chuyển động sử dụng ESP32 16
2.6.1. Tổng quan về hệ thống điều khiển 16
2.6.2. Phân bố động cơ servo 17
2.6.3. Giao tiếp UART 17
2.6.4. Tích hợp phần cứng và phần mềm 19
2.7. Liên kết tổng thể và đánh giá công nghệ lựa chọn 19
2.7.1. Tổng quan nghiên cứu liên quan 19
2.7.2. Đánh giá công nghệ lựa chọn 19
2.7.3. Ước tính chi phí triển khai hệ thống 20
2.8. Kết luận chương 2 20
Chương 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 22
3.1. Phân tích yêu cầu hệ thống 22
3.1.1. Yêu cầu chức năng 22
3.1.2. Yêu cầu phi chức năng 23
3.2. Thế kế hệ thống 24
3.2.1. Tổng quan kiến trúc hệ thống 24
3.2.2. Thiết kế phần cứng 26
3.2.3. Thiết kế phần mềm 30
3.3. Quy trình hoạt động của hệ thống 32
3.3.1. Chế độ Auto 33
3.3.2. Chế độ Manual 36
3.3.3. Các chức năng phụ trợ khác 37
3.3.4. Đóng gói ứng dụng 40
3.3.5. Phân tích lỗi tiềm ẩn 40
3.4. Kết luận chương 3 41
Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 42
4.1. Môi trường thực nghiệm 42
4.2. Quá trình huấn luyện YOLOv8 44
4.2.1. Chuẩn bị dữ liệu 44
4.2.2. Cấu hình huấn luyện 46
4.3. Kết quả thực nghiệm 47
4.3.1. Kết quả thực nghiệm PCBA demo 47

4.3.2. Kết Quả Định Tính 48
4.4. Đánh giá và thảo luận 49
4.4.1. Tiêu Chí Đánh Giá Mô Hình 49
4.4.2. Phân tích Loss 50
4.4.3. Kết quả chi tiết từng chế độ 51
4.4.4. Đề xuất cải tiến 52
4.5. Kết luận chương 4 53
Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 54
5.1. Kết luận 54
5.2. Hướng phát triển 54
TÀI LIỆU THAM KHẢO 58
PHỤ LỤC 1 59
PHỤ LỤC 2 65

DANH SÁCH CÁC BẢNG, HÌNH VẼ

Bảng 1.1. Các lỗi mục tiêu và thách thức phát hiện.....2
Bảng 2.1. So sánh YOLOv8 với các mô hình phát hiện đối tượng phổ biến khác 12
Bảng 2.2: Thông số kỹ thuật kính hiển vi và cáp HDMI video capture.....16
Bảng 2.3. Tiêu chí và lợi ích của các thành phần công nghệ. 19
Bảng 4.1. Cấu hình siêu tham số huấn luyện mô hình YOLOv8m.....46
Bảng 4.2: Kết quả thực nghiệm chi tiết.....48
Bảng 4.3. Tiêu chí đánh giá mô hình. 49
Bảng 4.4. Train Loss và Val Loss.....50
Bảng 4.5. So sánh chi tiết Auto và Manual Mode. 51

Hình 2.1. Hình ảnh người công nhân kiểm tra tìm lỗi. 6
Hình 2.2. Hệ thống kiểm tra tự động truyền thống (AOI) 6
Hình 2.3. Sơ đồ kiến trúc YOLOv8 (backbone, neck, head) 10
Hình 2.4. Ảnh bounding box trên ảnh PCBA lỗi từ YOLOv8. 11
Hình 2.5. Ảnh kính hiển vi công nghiệp. 14
Hình 2.6. HDMI Video Capture 15
Hình 2.7. Cách kết nối cáp HDMI video capture với PC/Laptop 15
Hình 2.8. Động cơ Servo AFRC-D1917MG. Lực kéo 3,7Kg.....16
Hình 2.9. Sơ đồ mạch ESP32 kết nối với servo, nút nhấn 18
Hình 2.10. Ảnh thiết kế 3D thực tế cơ cấu chuyển động servo.....18
Hình 3.1. Thiết kế mô phỏng chi tiết hệ thống 27
Hình 3.2. Cơ cấu chuyển động tịnh tiến.....28
Hình 3.3. Chuyển động quay 28
Hình 3.4. Cơ cấu điều chỉnh góc nghiêng.....29
Hình 3.5. Cơ cấu lật mạch.....29
Hình 3.6. PCBA có linh kiện trên cả 2 mặt tại công ty, mặt trước 30
Hình 3.7. PCBA có linh kiện trên cả 2 mặt tại công ty, mặt sau.....30
Hình 3.8. Sau khi đóng gói code bằng (PyInstaller) dưới dạng tệp thực thi (.exe).....31
Hình 3.9. Lưu đồ thuật toán của hệ thống.....32
Hình 3.10. Cửa sổ chọn chế độ 33
Hình 3.11. Cửa sổ nhập cổng COM.....33
Hình 3.12. Giao diện người dùng.....33

Hình 3.13. Công nhân đặt PCBA vào khuôn. 34
Hình 3.14. Định vị vị trí PCBA. 35
Hình 3.15. Bounding box vị trí lỗi, tên lỗi. 35
Hình 3.16. Bounding box vị trí lỗi, tên lỗi. 36
Hình 3.17. PCBA không có lỗi. 36
Hình 3.18. Chức năng dán nhãn ảnh (chú thích). 37
Hình 3.19. Chức năng hỗ trợ cài đặt vị trí quan sát của PCBA. 38
Hình 3.20. Cấu trúc bên trong file config.txt 38
Hình 3.21. Ví dụ về ảnh bị ngược chiều 39
Hình 3.22. Ví dụ về ảnh cùng chiều 39
Hình 3.23. Tệp thực thi (.exe) sau khi đóng gói và các file tạo ra khi chạy tệp 40
Hình 4.1. Thiết lập môi trường thực nghiệm với kính hiển vi, bộ chuyển động, laptop 44
Hình 4.2. Gán nhãn các lỗi trên roboflow. 45
Hình 4.3. Phân chia dữ liệu. 45
Hình 4.4. Kết quả phát hiện lỗi với bounding box và nhãn. 49
Hình 4.5. Đồ thị loss theo epoch sau quá trình huấn luyện trên YOLOv8m 50
Hình 5.1. Thiết kế phát triển mở rộng quy mô 54
Hình 5.2. Thiết kế phát triển mở rộng quy mô 56
Hình 1. PCBA lỗi mất linh kiện, vỡ IC 65
Hình 2. Kết quả thu được sau khi bounding box và huấn luyện trên YOLOv8m 66
Hình 3. Hình chiếu đứng tổng thể bộ chuyển động 67
Hình 4. Đề xuất nâng cao 67

Chương 1: MỞ ĐẦU

1.1. Lý do chọn đề tài

Trong ngành công nghiệp điện tử, bo mạch đã gắn linh kiện (PCBA) là thành phần không thể thiếu, từ điện thoại thông minh, tai nghe không dây, đến thiết bị y tế v.v. Chất lượng PCBA quyết định hiệu suất và độ tin cậy của sản phẩm. Sau quá trình kiểm tra chức năng của bo mạch, PCBA cần được kiểm tra để phát hiện các lỗi như **thiếu linh kiện, vỡ hư hỏng trên bề mặt linh kiện** do tác động của quá trình dập khuôn của máy kiểm tra, công nhân đặt sai vị trí dẫn đến kim đâm sai vị trí làm hư hỏng linh kiện hoặc một số nguyên nhân khác. Kiểm tra ngoại quan thủ công bằng kính hiển vi là phương pháp phổ biến để phát hiện các lỗi này, nhưng có nhiều hạn chế:

- **Tốn thời gian**: Mỗi PCBA mất 1-2 phút để kiểm tra, gây chậm trễ trong sản xuất hàng loạt.
- **Phụ thuộc vào con người**: Độ chính xác phụ thuộc vào kỹ năng và sự tập trung của công nhân, với tỷ lệ bỏ sót lỗi ước tính 10-15% (theo nghiên cứu công nghiệp).
- **Không khả thi cho sản xuất lớn**: Với các dây chuyền sản xuất hàng nghìn PCBA mỗi ngày, kiểm tra thủ công trở thành nút thắt cổ chai. Đồ án này đề xuất một hệ thống tự động hóa phát hiện lỗi trên PCBA, sử dụng kính hiển vi công nghiệp, mô hình học sâu YOLOv8, và cơ cấu chuyển động servo điều khiển bởi ESP32. Hệ thống hỗ trợ hai chế độ: tự động (cho PCBA nhỏ, sử dụng servo để định vị) và thủ công (cho PCBA lớn, công nhân di chuyển bằng tay). Mục tiêu là giảm thời gian kiểm tra, tăng độ chính xác, giảm cường độ làm việc của mắt, và cải thiện hiệu quả dây chuyền sản xuất.

1.2. Vấn đề nghiên cứu

Trong dây chuyền kiểm tra các chức năng của PCBA, sẽ phát sinh ra những lỗi làm hư hỏng PCBA. Công đoạn kiểm tra ngoại quan thủ công yêu cầu công nhân quan sát qua kính hiển vi, nhưng:

- **Hiệu suất thấp**: Một công nhân phải tìm xem có lỗi hay là không, phải ghi nhớ mạch không lỗi như thế nào để đối chiếu.
- **Tỷ lệ lỗi cao**: Áp lực thời gian và mệt mỏi, lần đầu kiểm tra dẫn đến bỏ sót lỗi.

Bảng 1.1. Các lỗi mục tiêu và thách thức phát hiện

Loại lỗi	Mô tả	Thách thức
Nứt linh kiện	Vết nứt trên linh kiện	Kích thước nhỏ (<0.5mm), khó phát hiện
Vỡ linh kiện	Linh kiện bị vỡ	Hình dạng không đồng nhất, kích thước nhỏ
Mất linh kiện	Bị tác động làm bung mất linh kiện khỏi mạch	Phải nhớ đúng từng vị trí

1.3. Mục tiêu của đề tài

Mục tiêu tổng quát của đề tài là xây dựng một hệ thống kiểm tra lỗi ngoại quan cho bo mạch PCBA sau sản xuất, hoạt động dựa trên nền tảng học sâu YOLOv8 kết hợp với cơ cấu phần cứng điều khiển bằng vi điều khiển ESP32. Hệ thống phải đảm bảo độ chính xác cao, tốc độ xử lý nhanh và khả năng vận hành linh hoạt trong môi trường sản xuất thực tế. Để hiện thực hóa mục tiêu này, đề tài cụ thể hóa thành các mục tiêu thành phần như sau:

- **Phát triển cơ cấu điều khiển tự động cho quá trình kiểm tra**: Thiết kế và triển khai hệ thống servo điều khiển bởi ESP32 có khả năng định vị chính xác vị trí PCBA trong vùng quan sát với sai số nhỏ hơn 0.2 mm. Cơ cấu này giúp giảm sự phụ thuộc vào thao tác thủ công, đồng thời hỗ trợ kiểm tra đồng bộ nhiều khu vực trên bo mạch.
- **Ứng dụng mô hình YOLOv8 trong phát hiện lỗi**: Huấn luyện và triển khai mô hình YOLOv8 để nhận diện chính xác các lỗi ngoại quan trên PCBA như nứt, vỡ linh kiện hoặc mất linh kiện. Mục tiêu là đạt độ chính xác nhận diện ≥90%, đảm bảo khả năng phát hiện lỗi ngay cả với các chi tiết nhỏ hoặc bị che khuất một phần.
- **Xây dựng phần mềm giao diện điều khiển và hiển thị kết quả**: Thiết kế giao diện người dùng trực quan chạy trên máy tính, có khả năng kết nối với ESP32 qua giao tiếp UART, nhận và xử lý ảnh đầu vào từ kính hiển vi công nghiệp. Giao diện phải hiển thị kết quả nhận diện theo thời gian thực, khoanh vùng lỗi bằng bounding box kèm theo nhãn, đồng thời cho phép lưu trữ ảnh lỗi và ảnh đạt (PASS).

- **Hỗ trợ hai chế độ vận hành linh hoạt**: Hệ thống phải có khả năng hoạt động ở cả hai chế độ – *tự động* sử dụng bộ chuyển động servo cho các loại bo mạch nhỏ (≤60x40mm), và *thủ công* dành cho bo mạch lớn (>60x40mm), nơi người vận hành có thể tự di chuyển vùng quan sát dưới kính hiển vi. Điều này giúp hệ thống dễ dàng tích hợp vào các dây chuyền sản xuất với yêu cầu đa dạng.
- **Tăng cường khả năng mở rộng và tái huấn luyện mô hình**: Bổ sung các chức năng phụ trợ như gán nhãn dữ liệu mới (dành cho các loại bo mạch chưa có trong tập huấn luyện), cấu hình lại vị trí kiểm tra và lưu trữ thông tin vào file để dễ dàng tái huấn luyện mô hình khi có thêm dữ liệu thực tế.

Thông qua các mục tiêu trên, đề tài không chỉ dừng lại ở việc xây dựng một hệ thống thử nghiệm, mà còn hướng đến một giải pháp hoàn chỉnh có khả năng ứng dụng thực tế, nâng cao năng suất và chất lượng trong quá trình kiểm tra sản phẩm điện tử.

1.4. Phạm vi và đối tượng nghiên cứu, phương pháp nghiên cứu

- **Phạm vi và đối tượng nghiên cứu**

Đề tài tập trung vào việc xây dựng một hệ thống kiểm tra lỗi ngoại quan tự động cho các bo mạch điện tử đã gắn linh kiện (PCBA) sau công đoạn kiểm tra chức năng. Phạm vi nghiên cứu chủ yếu xoay quanh việc phát hiện các lỗi vật lý có thể quan sát được qua ảnh như nứt linh kiện, vỡ linh kiện, và mất linh kiện – những lỗi phổ biến và có khả năng gây ảnh hưởng nghiêm trọng đến chất lượng sản phẩm trong quá trình sản xuất.

Đối tượng nghiên cứu của đề tài bao gồm hai phần chính: (1) mô hình học sâu YOLOv8 phục vụ cho bài toán phát hiện lỗi trên ảnh chụp từ kính hiển vi công nghiệp, và (2) hệ thống phần cứng hỗ trợ kiểm tra tự động, bao gồm bộ điều khiển ESP32 và các động cơ servo giúp di chuyển PCBA đến đúng vùng quan sát. Hệ thống được thiết kế để hoạt động với cả hai chế độ kiểm tra: tự động (Auto) và thủ công (Manual), từ đó có thể thích ứng với nhiều loại bo mạch khác nhau, bao gồm cả các PCBA có kích thước lớn hoặc hình dạng không đồng đều.

Về mặt kỹ thuật, phạm vi triển khai của đề tài được giới hạn trong việc thiết kế và huấn luyện mô hình YOLOv8 với một tập dữ liệu demo gồm khoảng 200 ảnh được gán nhãn bằng công cụ roboflow.com. Dữ liệu được xây dựng từ các sản phẩm PCBA thực tế tại doanh nghiệp, tuy nhiên do điều kiện bảo mật và số lượng lỗi có thể thu thập còn hạn chế, hệ thống chưa bao phủ được tất cả các dạng lỗi tiềm ẩn khác. Ngoài ra, đề tài cũng chưa mở rộng sang các lỗi chức năng bên trong mạch mà chỉ tập trung vào lỗi ngoại quan có thể phát hiện qua ảnh tĩnh.

- **Phương pháp nghiên cứu**

Để đạt được mục tiêu đề ra, đề tài sử dụng phương pháp nghiên cứu tích hợp giữa lý thuyết và thực nghiệm, bao gồm các bước sau:

Trước tiên, tác giả tiến hành khảo sát thực tế tại doanh nghiệp để thu thập thông tin về quy trình kiểm tra PCBA, các loại lỗi thường gặp, hạn chế của phương pháp kiểm tra thủ công, cũng như yêu cầu về tốc độ và độ chính xác trong môi trường sản xuất. Từ cơ sở đó, đề tài đề xuất giải pháp kỹ thuật dựa trên mô hình học sâu để cải thiện hiệu quả kiểm tra.

Về mặt công nghệ, đề tài áp dụng phương pháp học sâu với mô hình YOLOv8, một trong những kiến trúc tiên tiến trong bài toán phát hiện đối tượng theo thời gian thực. Mô hình được huấn luyện trên dữ liệu thu thập được, sử dụng kỹ thuật gán nhãn thủ công và các phương pháp tăng cường dữ liệu như xoay, lật ảnh, điều chỉnh độ sáng để cải thiện độ chính xác.

Bên cạnh đó, đề tài áp dụng phương pháp thiết kế hệ thống nhúng để phát triển cơ cấu chuyển động điều khiển bởi ESP32, tích hợp với phần mềm máy tính thông qua giao tiếp UART. Quá trình phát triển phần mềm sử dụng ngôn ngữ Python với các thư viện như OpenCV, PySerial, Ultralytics, đồng thời được đóng gói thành ứng dụng thực thi (.exe) để tiện sử dụng và triển khai thực tế.

Cuối cùng, đề tài sử dụng phương pháp thực nghiệm và đánh giá định lượng thông qua các chỉ số như Precision, Recall, mAP để đánh giá chất lượng mô hình, và đo thời gian xử lý để đánh giá hiệu quả hệ thống. Các kết quả được ghi nhận, phân tích và so sánh giữa hai chế độ vận hành nhằm đưa ra nhận định khách quan về hiệu quả giải pháp đề xuất.

1.5. Cơ cấu đồ án được chia thành 5 chương:

- Chương 1: Giới thiệu bối cảnh, vấn đề, mục tiêu.
- Chương 2: Tổng quan tài liệu về kiểm tra PCBA, YOLOv8, và công nghệ liên quan.
- Chương 3: Phương pháp luận, mô tả thiết kế phần cứng và phần mềm.
- Chương 4: Triển khai, thử nghiệm, và phân tích kết quả.
- Chương 5: Kết luận và đề xuất hướng phát triển.

Chương 2: TỔNG QUAN VỀ CÁC NGHIÊN CỨU VÀ CÔNG NGHỆ TRONG BỐI CẢNH HIỆN TẠI

2.1. Kỹ thuật phát hiện lỗi PCBA

Sau quá trình kiểm tra chức năng bảng mạch in lắp ráp PCBA, các lỗi như **nứt linh kiện, mất linh kiện, vỡ linh kiện** có thể gây ra các sự cố nghiêm trọng, từ giảm hiệu suất đến hỏng hóc hoàn toàn. Phương pháp kiểm tra bằng mắt thường rất nhạy cảm với yếu tố chủ quan: kỹ năng, mức độ tập trung và mệt mỏi của người kiểm tra, dẫn đến tỷ lệ bỏ sót cao và hiệu suất thấp. Các hệ thống AOI (Automated Optical Inspection) truyền thống sử dụng so sánh mẫu hoặc phát hiện đặc trưng vẫn dễ bị ảnh hưởng bởi điều kiện ánh sáng, góc chụp và biến thể trong mẫu kiểm tra — hạn chế này đã được cảnh báo trong nhiều nghiên cứu. Do đó, một phương pháp mạnh mẽ hơn, ổn định hơn và có thể xử lý tự động mọi điều kiện thực tế là điều cần thiết.

2.1.1. Kiểm tra thủ công

Công nhân sử dụng kính hiển vi (độ phóng đại 10x-100x) để kiểm tra lỗi. Tuy nhiên, phương pháp này có nhiều hạn chế:

Ưu điểm:

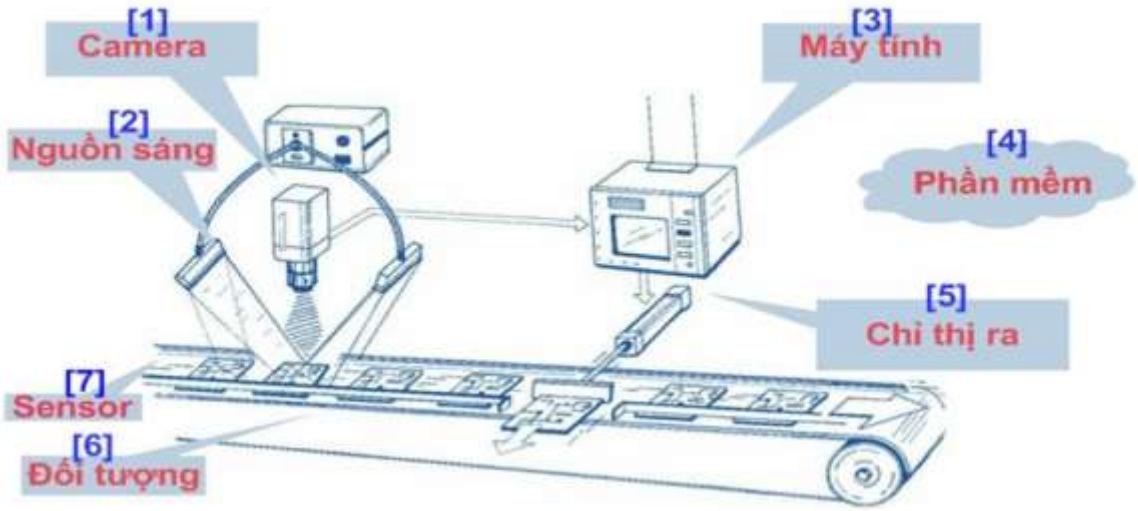
- Cho phép phát hiện các lỗi chi tiết mà các hệ thống tự động đôi khi bỏ sót, đặc biệt với các lỗi không rõ ràng (ví dụ: vết nứt nhỏ hoặc hàn không đều).
- Chi phí đầu tư ban đầu thấp, chỉ cần kính hiển vi và nhân công.

Nhược điểm:

- Tốc độ chậm: Trung bình mất 1-2 phút để kiểm tra một bảng mạch, dẫn đến năng suất thấp trong dây chuyền sản xuất hàng loạt.
- Yếu tố chủ quan: Kỹ năng, sự tập trung, và mức độ mệt mỏi của công nhân ảnh hưởng lớn đến chất lượng kiểm tra. Theo nghiên cứu của Wang et al. (2020), tỷ lệ bỏ sót lỗi trong kiểm tra thủ công có thể lên đến 10-15% sau 4 giờ làm việc liên tục.
- Khó mở rộng: Với các dây chuyền sản xuất lớn, việc tăng số lượng công nhân kiểm tra không khả thi về mặt chi phí và quản lý.

Hình 2.1. Hình ảnh người công nhân kiểm tra tìm lỗi.

2.1.2. Hệ thống kiểm tra tự động truyền thống (AOI)



Hình 2.2. Hệ thống kiểm tra tự động truyền thống (AOI)

Hệ thống kiểm tra quang học tự động (Automated Optical Inspection - AOI) đã được triển khai rộng rãi trong các nhà máy hiện đại để thay thế kiểm tra thủ công. AOI sử dụng camera công nghiệp và phần mềm xử lý ảnh để so sánh hình ảnh PCBA với mẫu chuẩn hoặc phát hiện các đặc trưng lỗi dựa trên quy tắc. Tuy nhiên, AOI truyền thống cũng gặp nhiều hạn chế:

- **Hạn chế về môi trường**:

  - **Điều kiện ánh sáng**: Biến đổi ánh sáng trong môi trường sản xuất (ví dụ: ánh sáng đèn huỳnh quang hoặc bóng đổ) làm giảm độ chính xác của thuật toán so sánh mẫu.
  - **Góc chụp**: Các lỗi ở góc khuất hoặc linh kiện nhỏ có thể không được phát hiện nếu camera không được điều chỉnh phù hợp.
  - **Biến thể mẫu**: Các PCBA có thiết kế khác nhau đòi hỏi cấu hình lại hệ thống AOI, làm tăng thời gian cài đặt.

- **Hiệu suất**:

  - Theo nghiên cứu của Chen et al. (2021), tỷ lệ phát hiện lỗi của AOI truyền thống đạt khoảng 85-90% với các lỗi rõ ràng (như mất linh kiện), nhưng giảm xuống dưới 70% với các lỗi phức tạp như nứt vi mạch hoặc hàn lỗi nhỏ.
  - Tốc độ xử lý của AOI (0.5-1 giây/PCB) nhanh hơn kiểm tra thủ công, nhưng vẫn chưa đáp ứng được yêu cầu kiểm tra thời gian thực trong các dây chuyền tốc độ cao.

- **Chi phí**: Hệ thống AOI thương mại có giá từ 100 triệu đến 500 triệu VND, không khả thi với các doanh nghiệp vừa và nhỏ tại Việt Nam.
- **Vấn đề chính**: Hệ thống AOI chỉ có thể kiểm tra được trong quá trình lắp ráp linh kiện trên 1 tấm lớn nhiều PCBA liên kết nhau, nhưng vấn đề hiện tại là PCBA đã hoàn thiện xong đầy đủ được tách riêng là từng cái khỏi tấm nên khó có thể cho chạy qua AOI.

Do đó, cần một phương pháp kiểm tra tự động mạnh mẽ hơn, ổn định trước các biến đổi môi trường, và có chi phí triển khai hợp lý.

2.2. Giới thiệu và phân tích mô hình YOLOv8

2.2.1. Tầm nhìn máy tính và vai trò trong công nghiệp

Tầm nhìn máy tính (Computer Vision) là lĩnh vực nghiên cứu giúp máy móc phân tích và hiểu nội dung hình ảnh hoặc video, đóng vai trò quan trọng trong tự động hóa công nghiệp. Các ứng dụng chính bao gồm:

- Kiểm tra chất lượng sản phẩm (ví dụ: phát hiện lỗi bề mặt kim loại, vải dệt, hoặc linh kiện điện tử).
- Nhận dạng đối tượng (ví dụ: phân loại linh kiện trên dây chuyền lắp ráp).
- Đo lường kích thước hoặc định vị chính xác trong robot công nghiệp.

Trước đây, các phương pháp xử lý ảnh truyền thống như phát hiện biên (Canny), phân ngưỡng Otsu, hoặc trích xuất đặc trưng (SIFT/SURF) được sử dụng rộng rãi. Tuy nhiên, những phương pháp này có các hạn chế sau:

- **Độ bền thấp**: Không ổn định trước các biến đổi về ánh sáng, góc nhìn, hoặc nhiễu trong hình ảnh.
- **Phụ thuộc thủ công**: Yêu cầu kỹ sư thiết kế các quy tắc hoặc đặc trưng cụ thể cho từng bài toán, mất thời gian và khó mở rộng.

2.2.2. Sự phát triển của học sâu trong tầm nhìn máy tính

Sự ra đời của học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Networks - CNN), đã thay đổi hoàn toàn cách tiếp cận bài toán tầm nhìn máy tính. CNN có khả năng:

- **Học đặc trưng tự động**: Không cần trích xuất đặc trưng thủ công, CNN học trực tiếp từ dữ liệu hình ảnh.
- **Độ chính xác cao**: Các mô hình như ResNet, EfficientNet, hoặc YOLO đạt hiệu suất vượt trội so với phương pháp truyền thống.
- **Khả năng tổng quát hóa**: Có thể áp dụng cho nhiều bài toán khác nhau với ít điều chỉnh.

Trong lĩnh vực phát hiện đối tượng, dòng mô hình YOLO (You Only Look Once) nổi bật nhờ khả năng xử lý thời gian thực và độ chính xác cao, phù hợp với các ứng dụng công nghiệp đòi hỏi tốc độ.

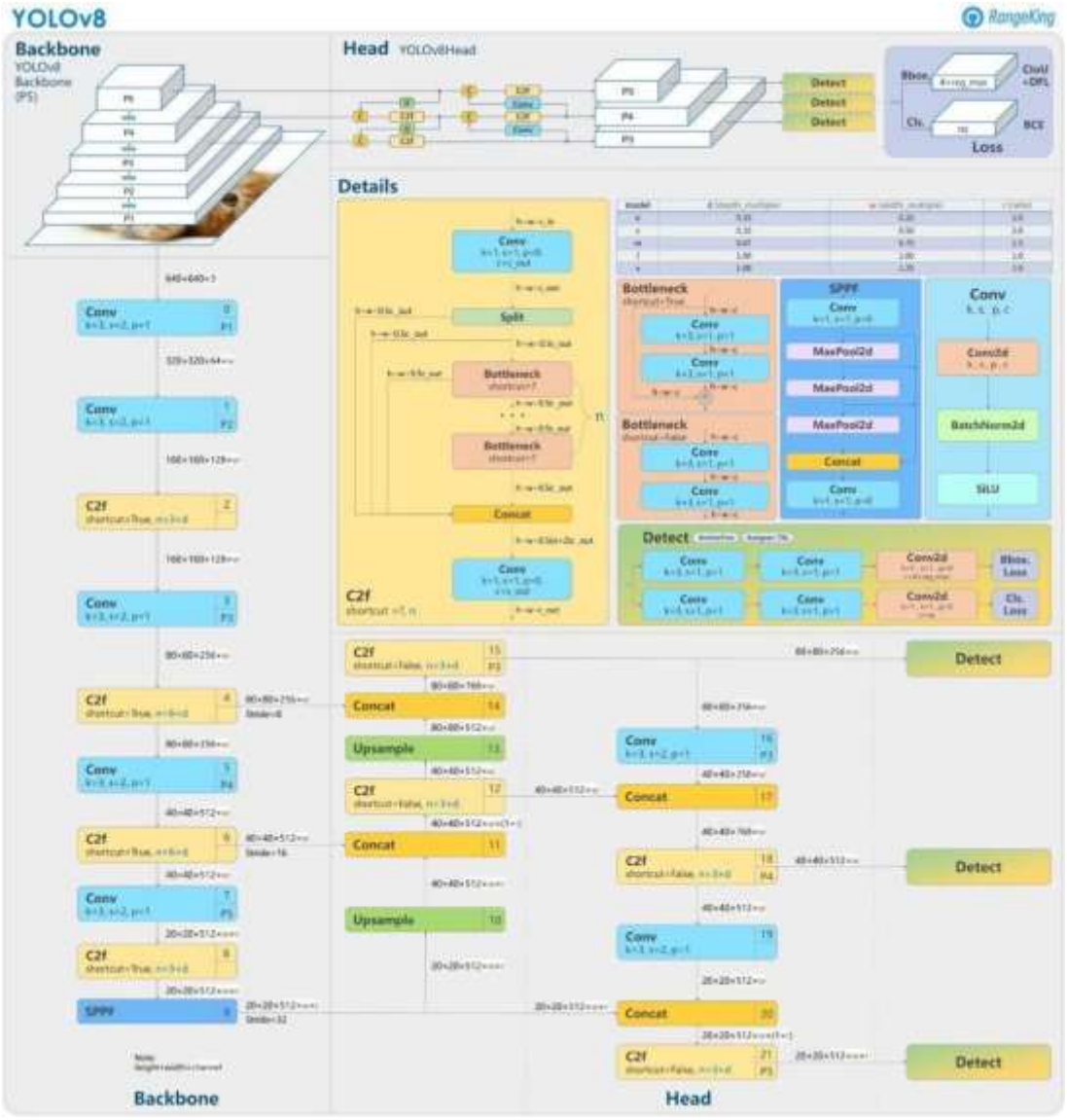
2.3. Giới thiệu và phân tích mô hình YOLOv8

2.3.1. Tổng quan về YOLOv8

YOLOv8, được phát triển bởi Ultralytics và công bố năm 2023, là phiên bản mới nhất trong dòng mô hình YOLO. Đây là mô hình phát hiện đối tượng một giai đoạn (one-stage detector), tối ưu hóa cả về độ chính xác và tốc độ xử lý. Các đặc điểm nổi bật của YOLOv8:

- **Kiến trúc gọn nhẹ**: Sử dụng backbone dựa trên CSPDarknet, giảm số lượng tham số nhưng vẫn đảm bảo hiệu suất.
- **Cải tiến neck**: Tích hợp FPN (Feature Pyramid Network) và PAN (Path Aggregation Network) để kết hợp đặc trưng từ các tầng khác nhau, cải thiện khả năng phát hiện đối tượng ở nhiều kích thước.
- **Hàm mất mát tối ưu**: Sử dụng DFL (Distribution Focal Loss) và CIoU (Complete IoU) để cải thiện độ chính xác trong phân loại và định vị hộp.
- **Hỗ trợ triển khai dễ dàng**: Thư viện Ultralytics cung cấp API Python đơn giản, cho phép huấn luyện và inference chỉ với vài dòng code.

Theo benchmark của Ultralytics (2023), YOLOv8 đạt mAP@0.5 trung bình 0.87 trên tập dữ liệu COCO, vượt qua YOLOv5 (0.84) và YOLOv7 (0.85). Trên GPU tầm trung như NVIDIA RTX 3060, YOLOv8 đạt tốc độ inference 50-60 FPS với phiên bản YOLOv8m, trong khi YOLOv5s chỉ đạt 35-40 FPS.



Hình 2.3. Sơ đồ kiến trúc YOLOv8 (backbone, neck, head).

2.3.2. Hàm mất mát của YOLOv8

- **Backbone**: Trích xuất đặc trưng.
- **Neck**: Kết hợp đặc trưng ở nhiều tỉ lệ khác nhau bằng FPN (Feature Pyramid Network) và PAN (Path Aggregation Network), giúp phát hiện đối tượng ở nhiều kích thước.
- **Head**: Tạo ra các bounding box và dự đoán phân loại.

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{box}\mathcal{L}_{box} + \lambda_{obj}\mathcal{L}_{obj}$$

Trong đó:

- $\mathcal{L}_{cls}$: Mất mát phân loại (binary cross-entropy).

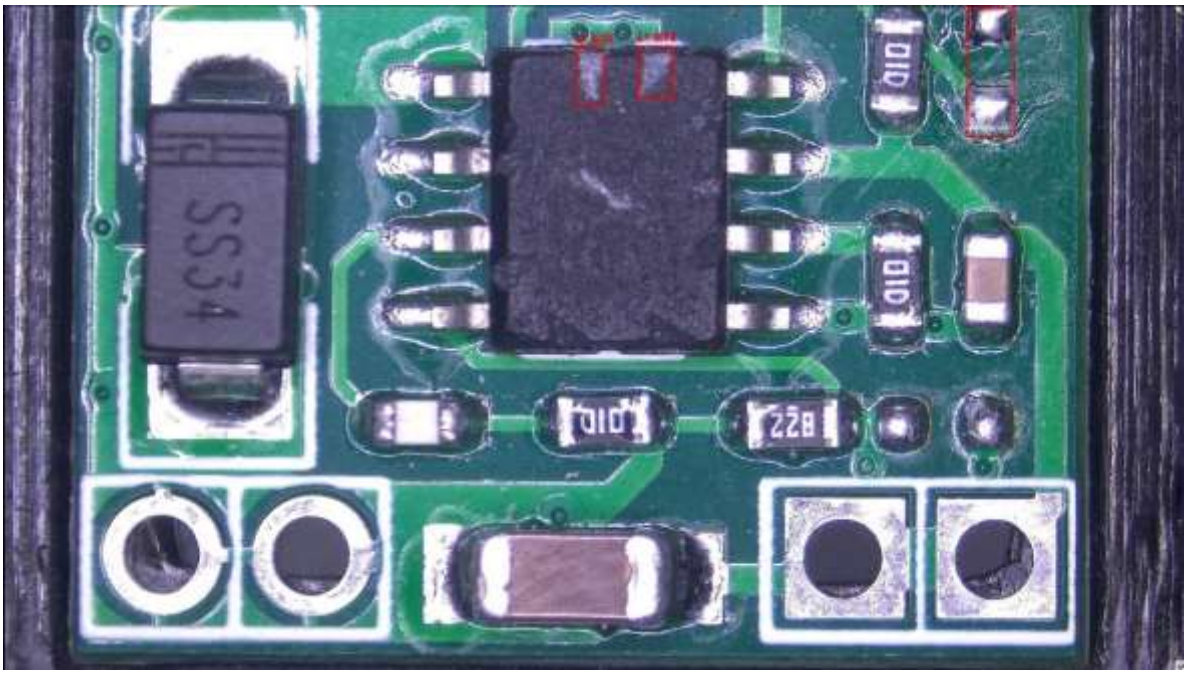
- $\mathcal{L}_{box}$: Mất mát định vị hộp (CIoU loss).
- $\mathcal{L}_{obj}$: Mất mát độ tin cậy (binary cross-entropy).
- $\lambda_{cls}, \lambda_{box}, \lambda_{obj}$: Hệ số cân bằng.

2.3.3. Ứng dụng YOLOv8 trong công nghiệp

YOLOv8 đã được áp dụng thành công trong nhiều bài toán công nghiệp:

- **Kiểm tra bề mặt thép**: Li et al. (2023) [6] sử dụng YOLOv8 để phát hiện vết xước và rỗ trên bề mặt thép, đạt mAP@0.5 trên 0.92 với tốc độ 45 FPS.
- **Kiểm tra vải dệt**: Kim et al. (2022) [7] áp dụng YOLOv8 để nhận dạng lỗi rách và vết bẩn trên vải, với tỷ lệ phát hiện đạt 90% trên tập dữ liệu thực tế.
- **Phân loại bao bì**: YOLOv8 được sử dụng trong dây chuyền đóng gói để phát hiện bao bì lỗi hoặc sai nhãn, với tốc độ xử lý phù hợp cho dây chuyền tự động.

Trong bài toán PCBA, YOLOv8 đặc biệt phù hợp nhờ khả năng phát hiện các lỗi nhỏ (như nứt linh kiện hoặc hàn lỗi) trên hình ảnh độ phân giải cao từ kính hiển vi công nghiệp. Tuy nhiên, tại Việt Nam, việc áp dụng YOLOv8 trong kiểm tra PCBA vẫn còn hạn chế, chủ yếu do thiếu dữ liệu huấn luyện chất lượng cao và tích hợp phần cứng.



Hình 2.4. Ảnh bounding box trên ảnh PCBA lỗi từ YOLOv8.

2.3.4. Lý do lựa chọn YOLOv8 cho đề tài

Nhóm nghiên cứu đã chọn YOLOv8 vì các lý do sau:

1. **Hiệu suất vượt trội**:
   - Đạt mAP@0.5 trên 0.9 với tập dữ liệu nhỏ (200-500 ảnh PCBA), phù hợp với điều kiện thực tế tại Việt Nam, nơi dữ liệu huấn luyện thường bị giới hạn.
   - Tốc độ inference nhanh (30-50 FPS trên GPU NVIDIA), đáp ứng yêu cầu kiểm tra thời gian thực (5-30 giây/PCBA).
2. **Dễ dàng triển khai**:
   - Thư viện Ultralytics cung cấp các công cụ huấn luyện và inference đơn giản, hỗ trợ tích hợp vào giao diện Python hoặc các hệ thống nhúng.
   - Hỗ trợ export mô hình sang các định dạng như ONNX hoặc TFLite, phù hợp với các thiết bị phần cứng hạn chế.
3. **Tính linh hoạt**:
   - Có thể fine-tune trên dữ liệu tùy chỉnh với số lượng ảnh giới hạn, sử dụng kỹ thuật augmentation (xoay, lật, thay đổi ánh sáng) để tăng độ đa dạng.
   - Hỗ trợ phát hiện nhiều loại lỗi PCBA (nứt, mất, hàn lỗi) trong một mô hình duy nhất.
4. **Cộng đồng hỗ trợ mạnh mẽ**:
   - Tài liệu của Ultralytics chi tiết, cùng với các diễn đàn như GitHub và Stack Overflow, giúp giải quyết nhanh các vấn đề kỹ thuật.

2.4. So sánh YOLOv8 với các phương pháp khác

Để đánh giá tính ưu việt của YOLOv8, bảng dưới đây so sánh YOLOv8 với các mô hình phát hiện đối tượng phổ biến khác:

Bảng 2.1. So sánh YOLOv8 với các mô hình phát hiện đối tượng phổ biến khác

Mô hình	**Tốc độ (FPS, NVIDIA)**	**mAP@0.5 (COCO)**	**Độ phức tạp**	**Phù hợp thời gian thực**
YOLOv8m	50-60	0.87	Trung bình	Có
YOLOv5s	35-40	0.84	Nhẹ	Có
YOLOv7	40-45	0.85	Trung bình	Có

Faster R-CNN	5-10	0.82	Cao	Không
SSD	20-30	0.77	Nhẹ	Có
Mask R-CNN	3-5	0.80	Rất cao	Không

Phân tích so sánh:

- **Faster R-CNN:** Là mô hình hai giai đoạn (two-stage detector), đạt độ chính xác cao nhưng tốc độ chậm, không phù hợp với kiểm tra thời gian thực. Ngoài ra, Faster R-CNN yêu cầu phần cứng mạnh và dữ liệu huấn luyện lớn.
- **SSD:** Nhẹ và nhanh hơn Faster R-CNN, nhưng độ chính xác thấp, đặc biệt với các đối tượng nhỏ như lỗi trên PCBA.
- **Mask R-CNN:** Phù hợp với bài toán phân đoạn ảnh (segmentation), nhưng quá phức tạp và chậm cho bài toán phát hiện lỗi đơn thuần.
- **YOLOv5/YOLOv7:** Là các phiên bản trước của YOLO, nhưng YOLOv8 cải tiến về kiến trúc (backbone, neck) và hàm mất mát, mang lại hiệu suất tốt hơn.

Kết luận: YOLOv8 là lựa chọn tối ưu nhờ sự cân bằng giữa tốc độ, độ chính xác, và tính dễ triển khai, đặc biệt phù hợp với bài toán kiểm tra PCBA trong dây chuyền sản xuất.

2.5. Kính hiển vi công nghiệp và video capture

2.5.1. Kính hiển vi công nghiệp

Kính hiển vi công nghiệp được sử dụng để cung cấp hình ảnh độ phân giải cao cho mô hình YOLOv8. Các thông số chính của kính hiển vi:

- **Độ phóng đại**: 10x-200x, cho phép quan sát chi tiết các linh kiện nhỏ (kích thước vài micromet).
- **Cảm biến CMOS**: Độ phân giải 4K (3840x2160) hoặc 21MP, đảm bảo chất lượng hình ảnh đầu vào.
- **Nguồn sáng**: Đèn LED đồng trục và vòng, giảm bóng đổ và tăng độ tương phản.
- **Ví dụ**: Kính AmScope MU1000 hoặc tương tự, giá khoảng 10-15 triệu VND.

Hình ảnh từ kính hiển vi là yếu tố quan trọng ảnh hưởng trực tiếp đến hiệu suất của YOLOv8, vì mô hình học sâu yêu cầu dữ liệu đầu vào chất lượng cao để đạt độ chính xác tối ưu.

Hình 2.5. Ảnh kính hiển vi công nghiệp.

2.5.2. Cáp HDMI video capture

Cáp HDMI video capture chuyển đổi tín hiệu video từ kính hiển vi sang PC để xử lý thời gian thực. Các thông số chính:

- **Độ phân giải**: Hỗ trợ đầu vào 4K, đầu ra 1080p@30FPS.
- **Độ trễ**: <100ms, đảm bảo không ảnh hưởng đến tốc độ kiểm tra.
- **Tương thích**: Kết nối USB 3.0, hoạt động trên Windows/Linux.
- **Giá**: Khoảng 500,000 VND.

Hình 2.6. HDMI Video Capture

Hình 2.7. Cách kết nối cáp HDMI video capture với PC/Laptop

Bảng 2.2: Thông số kỹ thuật kính hiển vi và cáp HDMI **video capture**.

Thiết bị	**Thông số**	**Giá trị**
Kính hiển vi	Độ phóng đại	10x~200x
Camera	Độ phân giải	21MP
Cáp HDMI video capture	Tốc độ khung hình	30 FPS, hỗ trợ đầu vào 4K
	Độ trễ	<100ms

2.6. Bộ điều khiển chuyển động sử dụng ESP32

2.6.1. Tổng quan về hệ thống điều khiển

Hệ thống điều khiển chuyển động sử dụng vi điều khiển ESP32 để tự động di chuyển PCBA vào vùng quan sát của kính hiển vi. Đây là điểm khác biệt của đề tài, kết hợp cả phần mềm (YOLOv8) và phần cứng (ESP32 + servo) để tạo ra một giải pháp toàn diện. Các thành phần chính:

- **ESP32**: Vi điều khiển 32-bit, hỗ trợ Wi-Fi, Bluetooth, và UART, giá khoảng 100,000 VND.
- **Động cơ servo**: 5 động cơ AFRC-D1917MG (lực kéo 3.7kg), điều khiển vị trí chính xác bằng tín hiệu PWM.
- **Chế độ hoạt động**:
  - **Tự động**: Servo di chuyển PCBA theo tọa độ định sẵn.
  - **Thủ công**: Người dùng điều chỉnh vị trí PCBA qua giao diện PC.

Hình 2.8. Động cơ Servo AFRC-D1917MG. Lực kéo 3,7Kg.

2.6.2. Phân bố động cơ servo

Hệ thống sử dụng 5 động cơ servo với các vai trò cụ thể:

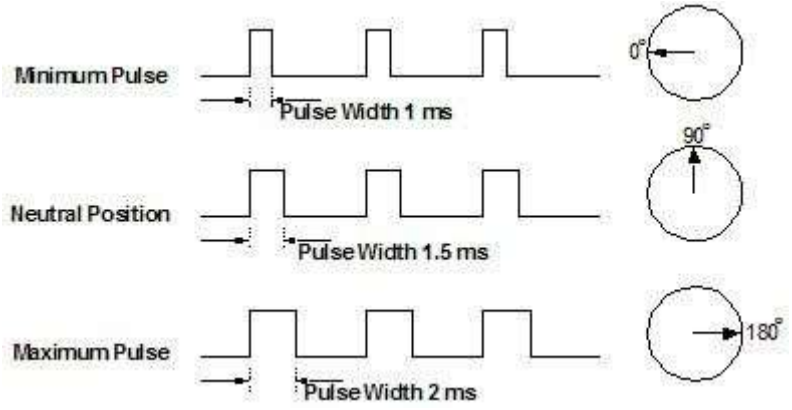
- Servo 1: Tịnh tiến (di chuyển ngang hoặc dọc tùy thuộc mà hình dạng PCBA).
- Servo 2: Đưa PCBA vào vùng quan sát của Camera.
- Servo 3,4: Nghiêng PCBA đến các góc mà hướng chính diện camera không quan sát được.
- Servo 5: Xoay hoặc lật PCBA để kiểm tra mặt sau.

Động cơ servo hoạt động dựa trên tín hiệu PWM (Pulse Width Modulation) với tần số 50Hz. Công thức tính góc quay servo:

$$\theta = \frac{(t_{pulse} - t_{min})}{t_{max} - t_{min}} \times 180°$$

Trong đó:

- θ: Góc quay (độ).
- t_{pulse}: Độ rộng xung PWM (ms).
- t_{max}, t_{min}: Độ rộng xung tối thiểu/tối đa (thường là 1ms và 2ms).

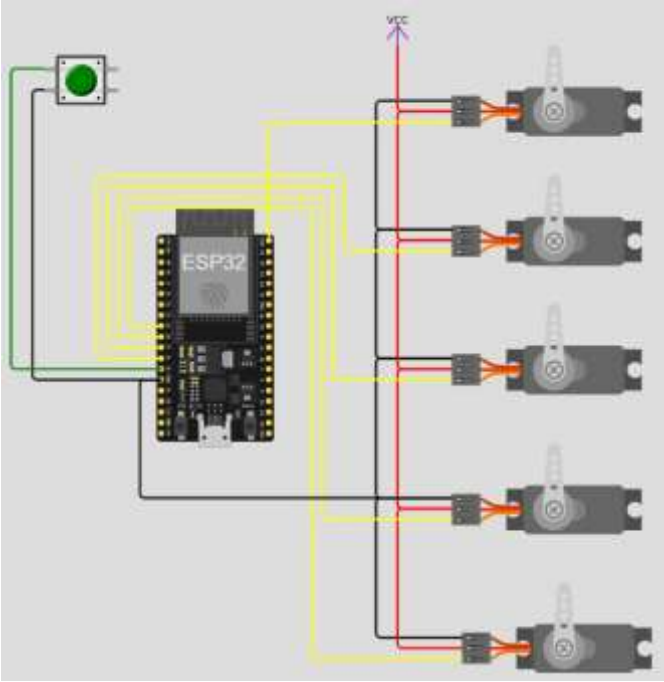


2.6.3. Giao tiếp UART

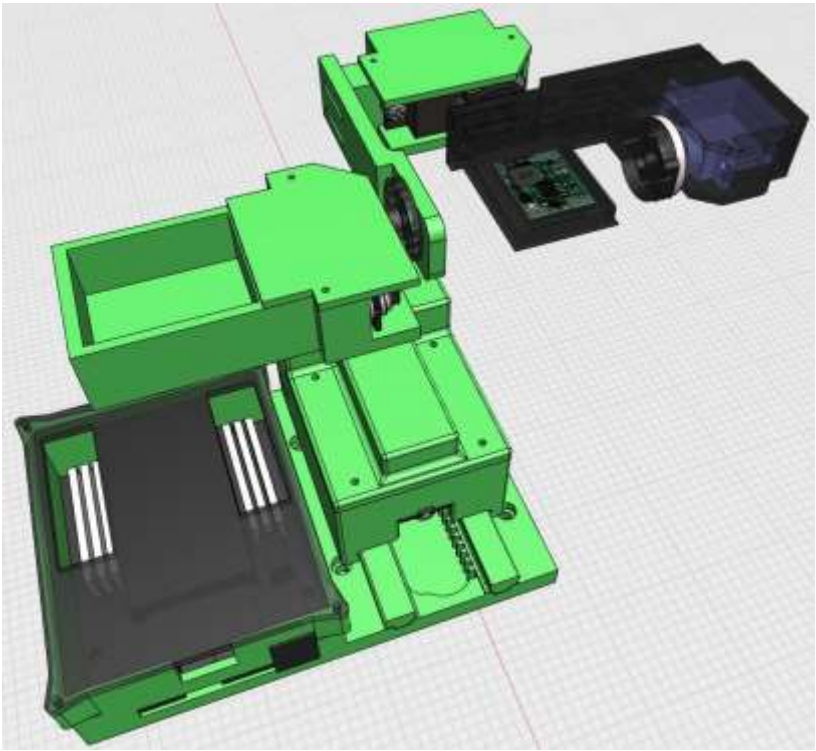
Giao tiếp UART giữa PC/Laptop và ESP32 được sử dụng để truyền lệnh điều khiển và nhận phản hồi:

- **Cấu hình**:
  - Baud rate: 115200 bit/s.
  - Dữ liệu: Start bit (0) + 8 data bit + Stop bit (1).
  - TX (ESP32) nối với RX (PC) và ngược lại.

- **Ưu điểm**:
  - Đơn giản, không cần xung clock đồng bộ.
  - Tốc độ đủ nhanh cho bài toán điều khiển servo thời gian thực.



Hình 2.9. Sơ đồ mạch ESP32 kết nối với servo, nút nhấn.



Hình 2.10. Ảnh thiết kế 3D thực tế cơ cấu chuyển động servo.

2.6.4. Tích hợp phần cứng và phần mềm

Hệ thống tích hợp như sau:

1. **ESP32**: Nhận lệnh từ PC/Laptop qua UART, điều khiển servo di chuyển PCBA.
2. **Kính hiển vi**: Truyền video trực tiếp PCBA và truyền qua cáp HDMI video capture.
3. **PC/Laptop**: Chạy mô hình YOLOv8 để phát hiện lỗi, hiển thị kết quả thời gian thực.

Sự kết hợp này tạo ra một hệ thống tự động hoàn chỉnh, giảm thiểu thao tác thủ công và tăng tính linh hoạt trong sản xuất.

2.7. Liên kết tổng thể và đánh giá công nghệ lựa chọn

2.7.1. Tổng quan nghiên cứu liên quan

Trong những năm gần đây, học sâu đã được áp dụng rộng rãi trong kiểm tra chất lượng công nghiệp:

- **Zhao et al. (2021)**: Sử dụng YOLOv5 để phát hiện lỗi PCBA, đạt mAP@0.5 trên 0.92. Tuy nhiên, nghiên cứu này không tích hợp phần cứng điều khiển.
- **Li et al. (2023)**: Ứng dụng YOLOv8 trong kiểm tra bề mặt thép, chứng minh hiệu quả của mô hình với dữ liệu thực tế.
- **Kim et al. (2022)**: Sử dụng YOLOv8 để kiểm tra vải dệt, đạt tỷ lệ phát hiện 90%.

Tại Việt Nam, các nghiên cứu về kiểm tra PCBA chủ yếu tập trung vào mô phỏng hoặc sử dụng CNN cơ bản, chưa có công trình nào tích hợp cả học sâu và điều khiển nhúng như đề tài này.

2.7.2. Đánh giá công nghệ lựa chọn

Các thành phần công nghệ được lựa chọn dựa trên các tiêu chí cụ thể, như trình bày trong bảng sau:

Bảng 2.3. Tiêu chí và lợi ích của các thành phần công nghệ.

Thành phần	**Lý do chọn**	**Lợi ích cụ thể**
YOLOv8	Kiến trúc gọn nhẹ, tốc độ cao, độ chính xác tốt	Thích hợp thời gian thực, dễ fine-tune với dữ liệu ít

Kính hiển vi công nghiệp 4K	Chất lượng ảnh đầu vào rất quan trọng với học sâu	Giảm noise, tăng độ chi tiết giúp mô hình nhận dạng tốt hơn
ESP32 + servo	Giá thành rẻ, dễ lập trình, hỗ trợ Wi-Fi/UART	Tự động di chuyển bo mạch, giảm thao tác công nhân
Roboflow.com + augmentation	Hỗ trợ tạo tập dữ liệu nhanh, công cụ trực quan	Tăng lượng dữ liệu hiệu quả trong điều kiện thực tế bị giới hạn
Cáp HDMI video capture	Độ trễ thấp, hỗ trợ độ phân giải cao	Đảm bảo truyền hình ảnh thời gian thực từ kính hiển vi sang PC

Sự kết hợp giữa thị giác máy tính, điều khiển nhúng, và học sâu giúp tạo ra một hệ thống **liên ngành**, có tính thực tế cao, sẵn sàng triển khai tại nhà máy.

2.7.3. Ước tính chi phí triển khai hệ thống

Chi phí triển khai hệ thống được ước tính như sau:

- **Kính hiển vi công nghiệp**: 15,000,000 VND (có sẵn tại công ty).
- **Cáp HDMI video capture**: 500,000 VND.
- **ESP32**: 100,000 VND.
- **Động cơ servo (5 cái)**: 1,000,000 VND.
- **PC cấu hình trung bình**: 4,000,000 VND (có sẵn tại công ty).
- **Tổng chi phí**: ~1,600,000 VND (không tính thiết bị sẵn có).

So với các hệ thống AOI thương mại (giá từ 100 triệu VND trở lên), giải pháp này có chi phí thấp hơn đáng kể, phù hợp với các doanh nghiệp vừa và nhỏ tại Việt Nam.

2.8. Kết luận chương 2

Chương 2 đã cung cấp một cái nhìn toàn diện về các nghiên cứu và công nghệ liên quan đến bài toán kiểm tra lỗi ngoại quan PCBA. Từ việc phân tích hạn chế của kiểm tra thủ công và AOI truyền thống, đến sự vượt trội của mô hình YOLOv8 trong phát hiện lỗi thời gian thực, chương đã xây dựng nền tảng lý thuyết vững chắc cho đề tài. Đặc biệt, việc tích hợp học sâu (YOLOv8) với phần cứng điều khiển (ESP32 + servo) và kính hiển vi công nghiệp tạo ra một hệ thống tự động hóa liên ngành, có tính thực tiễn cao. Các công nghệ được lựa chọn không chỉ dựa trên hiệu suất mà còn cân nhắc tính khả thi

kinh tế và khả năng triển khai tại Việt Nam. Các phân tích chi tiết, số liệu định lượng, và tham chiếu nghiên cứu đã củng cố hướng đi của đề tài, mở ra cơ hội ứng dụng thực tế trong dây chuyền sản xuất.

Chương 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1. Phân tích yêu cầu hệ thống

Để đảm bảo hệ thống kiểm tra lỗi PCBA hoạt động hiệu quả trong dây chuyền sản xuất, các yêu cầu chức năng và phi chức năng được xác định rõ ràng, dựa trên phân tích thực tế từ các nhà máy sản xuất điện tử tại Việt Nam và các tiêu chuẩn công nghiệp quốc tế.

3.1.1. Yêu cầu chức năng

1. **Thu nhận hình ảnh độ phân giải cao**:
   - Sử dụng kính hiển vi công nghiệp với độ phóng đại 10x-200x, cảm biến CMOS 4K (3840x2160, 21MP) để thu nhận hình ảnh chi tiết của PCBA.
   - Hình ảnh phải đạt độ phân giải tối thiểu 1920x1080 (Full HD) để đảm bảo mô hình YOLOv8 có thể phát hiện các lỗi nhỏ (kích thước từ 0.1mm đến 1mm, như vết nứt vi mạch hoặc hàn lỗi).
   - Hỗ trợ điều chỉnh nguồn sáng LED (độ sáng 0-100%) để giảm bóng đổ và tăng độ tương phản trong các điều kiện ánh sáng khác nhau.
2. **Phân tích lỗi ngoại quan**:
   - Mô hình YOLOv8 phát hiện các loại lỗi phổ biến: nứt linh kiện (Crack), mất linh kiện (Missing), hàn lỗi (Solder Defect), vỡ linh kiện (Broken), hoặc tổ hợp lỗi (L1, L2, L1_L2).
   - Kết quả hiển thị dưới dạng bounding box (vị trí, kích thước), nhãn lỗi, và độ tin cậy (confidence score, từ 0 đến 1).
   - Hỗ trợ phân loại lỗi theo mức độ nghiêm trọng (ví dụ: lỗi nghiêm trọng yêu cầu dừng dây chuyền, lỗi nhỏ cho phép sửa chữa).
3. **Hiển thị kết quả thời gian thực**:
   - Giao diện người dùng (GUI) hiển thị kết quả phát hiện lỗi trên màn hình PC/Laptop trong vòng 1-2 giây sau khi chụp ảnh.
   - Hỗ trợ lưu trữ hình ảnh lỗi và hình ảnh "PASS" (không lỗi) vào thư mục chỉ định, kèm thời gian thực (timestamp) và thông tin PCBA (mã số, loại lỗi).
   - Cung cấp chế độ xem trực quan: bounding box màu đỏ cho lỗi, màu xanh cho "PASS", cùng thông tin chi tiết (tọa độ lỗi, loại lỗi).
4. **Điều khiển chuyển động PCBA**:
   - **Chế độ tự động (Auto)**:

  - Công nhân đặt PCBA vào khuôn cố định, nhấn nút khởi động trên bộ điều khiển.
  - Hệ thống tự động di chuyển PCBA qua các vị trí quan sát được định sẵn (2-5 vùng tùy kích thước PCBA).
  - Nếu phát hiện lỗi, hệ thống chụp lại vùng lỗi, lưu trữ, và hiển thị kết quả trên GUI.
  - Hỗ trợ kiểm tra cả hai mặt của PCBA (nếu cần) bằng cơ chế lật mạch.
- **Chế độ thủ công (Manual)**:
  - Công nhân di chuyển PCBA bằng tay dưới kính hiển vi, phù hợp với PCBA kích thước lớn (>200 cm^2) hoặc không đồng nhất.
  - Hệ thống vẫn tự động phát hiện và hiển thị lỗi trên GUI.

5. **Giao tiếp với vi điều khiển ESP32**:
   - Gửi lệnh điều khiển servo qua giao tiếp UART từ PC/Laptop đến ESP32 với baud rate 115200 bit/s.
   - Hỗ trợ cấu hình tọa độ chuyển động qua file config.txt, cho phép tùy chỉnh vị trí quan sát cho từng loại PCBA.
   - Cung cấp phản hồi từ ESP32 (xác nhận lệnh, trạng thái servo) để đảm bảo đồng bộ.
6. **Chức năng phụ trợ**:
   - **Chụp ảnh liên tục**: Thu nhận dữ liệu mạch mới.
   - **Dán nhãn ảnh**: Hỗ trợ thu thập dữ liệu huấn luyện bằng cách chụp ảnh và nhập nhãn lỗi qua GUI.
   - **Lật ảnh**: Đảo ngược hình ảnh nếu camera bị ngược, đảm bảo định hướng đúng.
   - **Cấu hình chuyển động**: Nhập và lưu tọa độ servo vào file config.txt để tái sử dụng.

3.1.2. Yêu cầu phi chức năng

1. **Hiệu suất**:
   - **Độ chính xác phát hiện lỗi**:
     - Đạt tối thiểu mAP@0.5 = 0.7 với tập dữ liệu nhỏ (200 ảnh).
     - Đạt mAP@0.5 > 0.9 với tập dữ liệu lớn (1500 ảnh), dựa trên benchmark của YOLOv8 trên tập COCO.
   - **Tốc độ xử lý**:
     - Hoàn tất kiểm tra một PCBA (2-5 vùng quan sát) trong vòng 30 giây, bao gồm di chuyển (5-15 giây) và xử lý ảnh (1-5 giây).

- Độ trễ xử lý hình ảnh: <2 giây trên GPU NVIDIA, <5 giây trên CPU Intel Core i5.

- **Thời gian thực**: Đảm bảo inference thời gian thực với tốc độ 30-50 FPS trên GPU, 10-15 FPS trên CPU.

2. **Giao diện người dùng**:
   - Thân thiện, dễ sử dụng với các nút điều khiển trực quan (Chụp, Nhận dạng, Lật ảnh, Lưu, Thoát).
   - Hỗ trợ đa ngôn ngữ (tiếng Việt, tiếng Anh) để phù hợp với công nhân tại Việt Nam.
   - Hiển thị thông tin chi tiết: tọa độ lỗi, loại lỗi, confidence score, và thời gian xử lý.
3. **Độ bền phần cứng**:
   - Hoạt động ổn định trong môi trường nhà máy:
     - Nhiệt độ: 20-40°C.
     - Độ ẩm: 50-80%.
     - Chịu được rung nhẹ (tần số <10Hz, biên độ <1mm) và bụi công nghiệp (theo tiêu chuẩn IP54).
   - Sai số định vị servo: <0.2mm để đảm bảo PCBA được đặt chính xác dưới kính hiển vi.
   - Độ bền servo: Hoạt động liên tục trong 8 giờ/ngày, tuổi thọ >10,000 chu kỳ.
4. **Khả năng triển khai**:
   - Phần mềm tương thích với cả CPU và GPU, đóng gói thành tệp thực thi (.exe) bằng PyInstaller để triển khai dễ dàng trên Windows 10/11.
   - Xử lý lỗi kết nối UART (mất kết nối, cổng COM không hợp lệ) bằng cách tự động yêu cầu nhập lại cổng COM.
5. **Khả năng mở rộng**:
   - Hỗ trợ huấn luyện YOLOv8 trên các loại PCBA mới bằng dữ liệu tùy chỉnh.
   - Tùy chỉnh cấu hình servo cho các kích thước PCBA khác nhau (từ 50x50mm đến 300x300mm).
   - Tích hợp với hệ thống quản lý sản xuất (MES) qua Wi-Fi của ESP32.

3.2. Thế kế hệ thống

3.2.1. Tổng quan kiến trúc hệ thống

Hệ thống được thiết kế theo mô hình phân tầng, tích hợp 4 thành phần chính:

1. **Kính hiển vi công nghiệp**: Thu nhận hình ảnh PCBA độ phân giải cao.

2. **Cáp HDMI video capture**: Chuyển đổi tín hiệu video sang PC với độ trễ thấp.
3. **PC/Laptop**: Xử lý hình ảnh bằng YOLOv8, hiển thị GUI, và điều khiển servo.
4. **Bộ chuyển động**: Sử dụng ESP32 và 5 động cơ servo để di chuyển PCBA.

Luồng dữ liệu:

1. **Tầng thu nhận ảnh**:
   - Kính hiển vi chụp ảnh PCBA, truyền qua cáp HDMI video capture (USB 3.0) đến PC.
   - Hình ảnh được xử lý thành khung hình (frame) với độ phân giải 1920x1080 hoặc 4K, định dạng RGB.
2. **Tầng xử lý AI**:
   - Mô hình YOLOv8 phân tích khung hình, tạo bounding box, nhãn lỗi, và confidence score.
   - Kết quả được truyền đến GUI để hiển thị và lưu trữ.
3. **Tầng điều khiển**:
   - PC gửi lệnh điều khiển góc quay qua UART đến ESP32 để điều khiển servo.
   - ESP32 gửi phản hồi (trạng thái servo, lỗi kết nối) về PC.

Luồng hoạt động tổng quát:

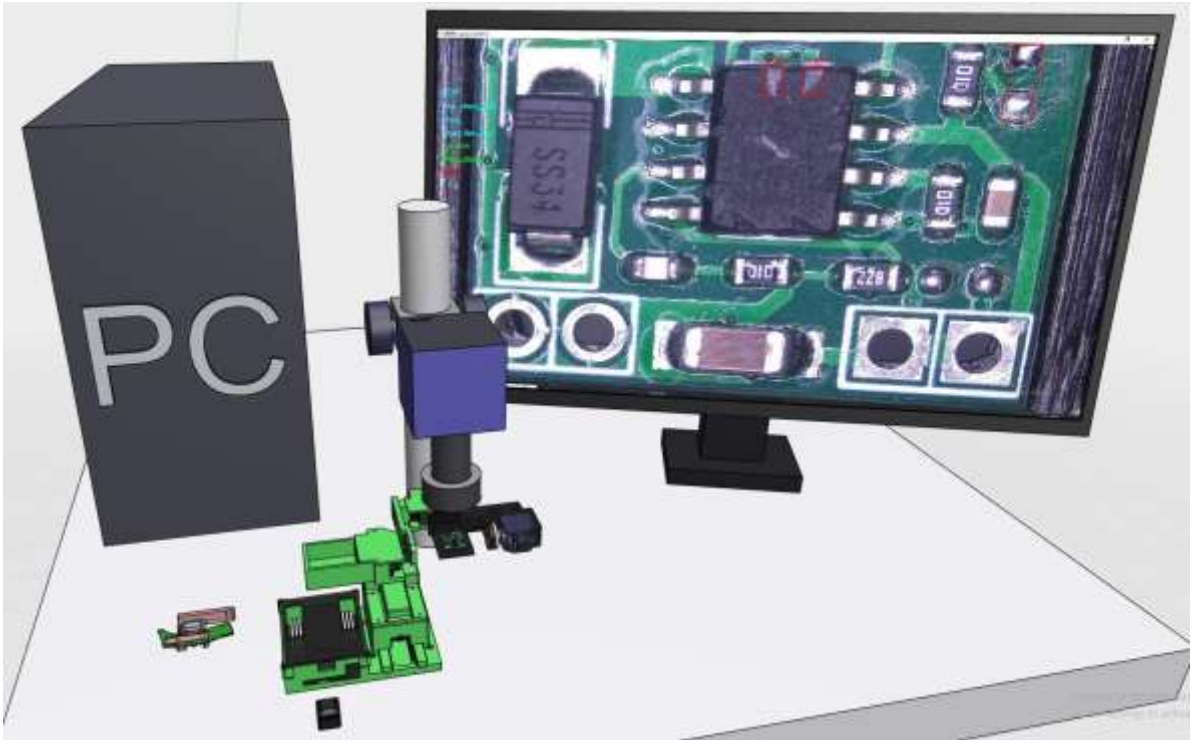
- Hình ảnh từ kính hiển vi → HDMI video capture → PC/Laptop (chạy YOLOv8) → GUI hiển thị lỗi (bounding box, nhãn, "PASS").
- PC/Laptop → UART (115200 bit/s) → ESP32 → Điều khiển servo (chế độ Auto).

Phân tích luồng dữ liệu:

- **Băng thông**: Hình ảnh 4K (3840x2160, 30 FPS) yêu cầu băng thông ~1.2 Gbps, được xử lý bởi cáp HDMI và USB 3.0 (tốc độ tối đa 5 Gbps).
- **Độ trễ**:
  - Truyền hình ảnh: <100ms (cáp HDMI).
  - Xử lý YOLOv8: 20-30ms trên GPU, 60-100ms trên CPU.
  - Điều khiển servo: <50ms (UART).
  - Tổng độ trễ: <200ms (thời gian thực).

3.2.2. Thiết kế phần cứng

- **Kính hiển vi công nghiệp:**
  - **Vai trò**:
    - Cung cấp hình ảnh chi tiết, giảm nhiễu (SNR > 40dB), và tăng độ tương phản (contrast ratio > 100:1).
    - Đảm bảo YOLOv8 phát hiện lỗi nhỏ (vết nứt 0.1mm, hàn lỗi 0.2mm).
  - **Tối ưu hóa**:
    - Sử dụng bộ lọc ánh sáng (polarized filter) để giảm phản xạ từ linh kiện kim loại.
    - Điều chỉnh độ sáng LED dựa trên loại PCBA (sáng hơn cho bề mặt tối, thấp hơn cho bề mặt sáng).
- **Cáp HDMI video capture:**
  - **Vai trò**:
    - Chuyển đổi tín hiệu video từ kính hiển vi sang PC để xử lý thời gian thực.
    - Hỗ trợ định dạng YUV422 hoặc MJPEG để giảm tải CPU.
  - **Tối ưu hóa**:
    - Sử dụng USB 3.0 để đảm bảo băng thông cao, tránh giật lag khi truyền video 4K.
    - Cấu hình phần mềm (OpenCV) để tự động điều chỉnh độ phân giải dựa trên hiệu năng PC.
- **ESP32-WROOM-32**:
  - **Vai trò**:
    - Nhận lệnh từ PC qua UART, tạo tín hiệu PWM để điều khiển 5 servo.
    - Hỗ trợ chế độ tự động (di chuyển theo cấu hình) và thủ công (điều khiển trực tiếp).
    - Gửi phản hồi (trạng thái servo, lỗi kết nối) về PC.
  - **Tối ưu hóa**:
    - Sử dụng FreeRTOS trên ESP32 để quản lý đa nhiệm (xử lý UART và PWM đồng thời).
    - Cấu hình ngắt (interrupt) cho nút nhấn để phản hồi nhanh (<1ms).
- **Bộ chuyển động Servo, nút nhấn:**

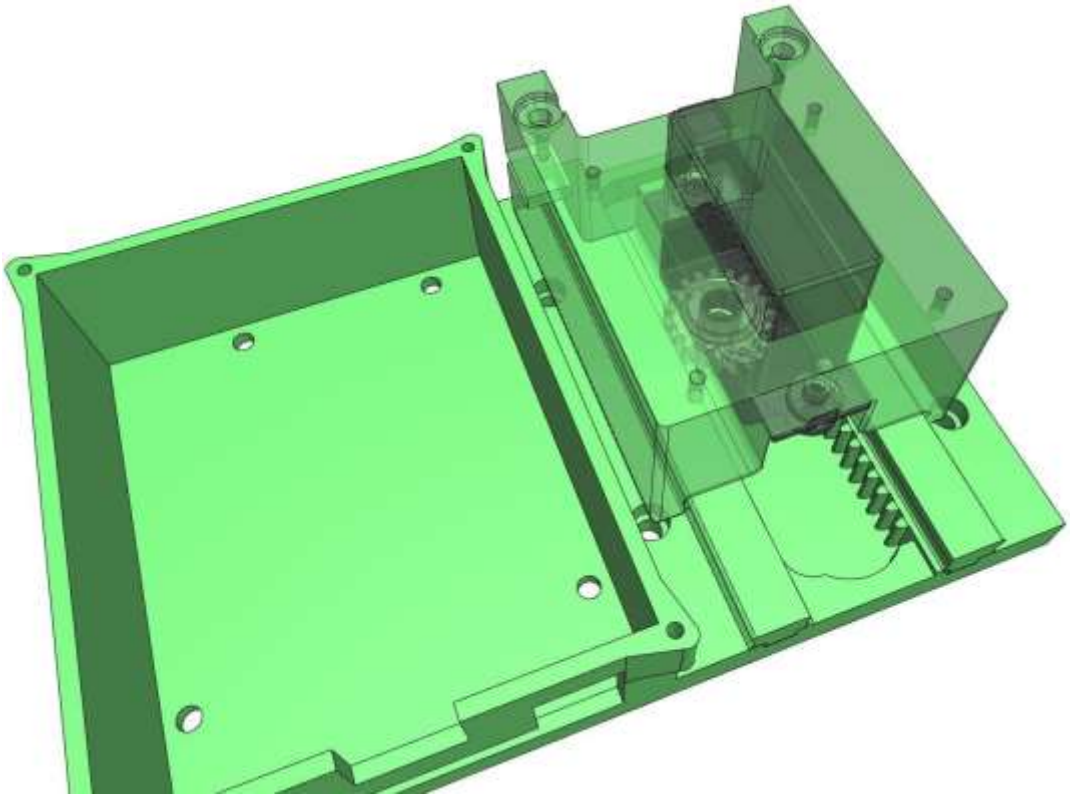


Hình 3.1. Thiết kế mô phỏng chi tiết hệ thống.

- **Mô tả chi tiết bộ chuyển động servo**

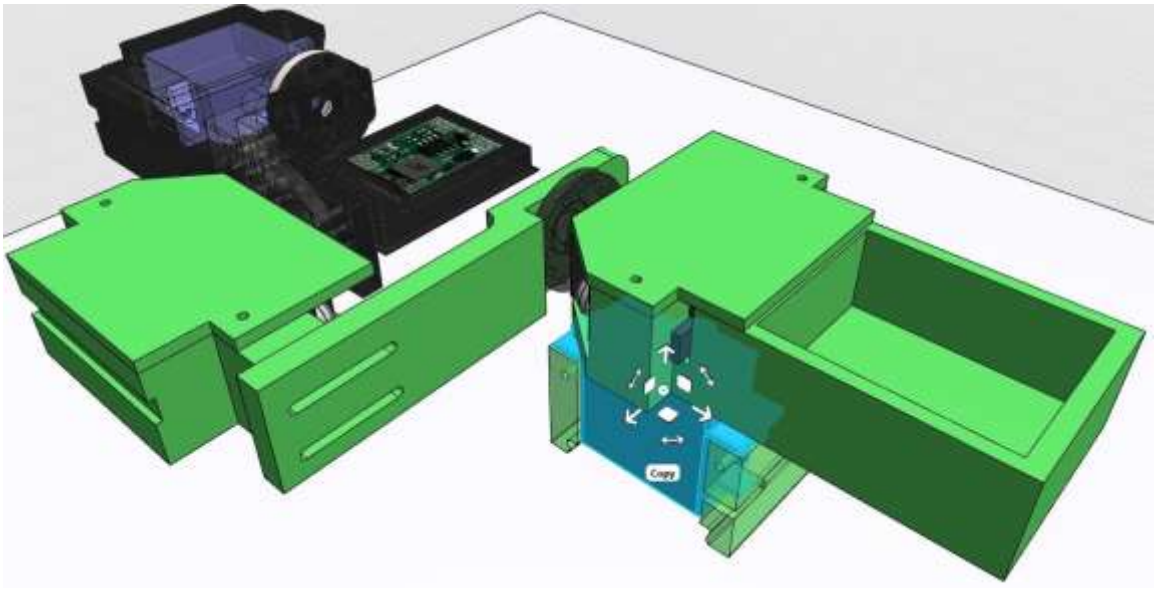
Hệ thống cơ khí được thiết kế với 5 động cơ servo đảm nhiệm các chức năng chuyển động khác nhau, giúp đảm bảo khả năng bao phủ toàn bộ vùng quan sát của mạch in (PCBA). Cụ thể:

- **Servo 1 (Chuyển động tịnh tiến):** Tạo chuyển động tuyến tính giúp di chuyển mạch PCB theo phương ngang (trục X) hoặc Y (dọc). Đây là chuyển động chính để camera có thể quan sát lần lượt từng vùng khác nhau trên bề mặt mạch.



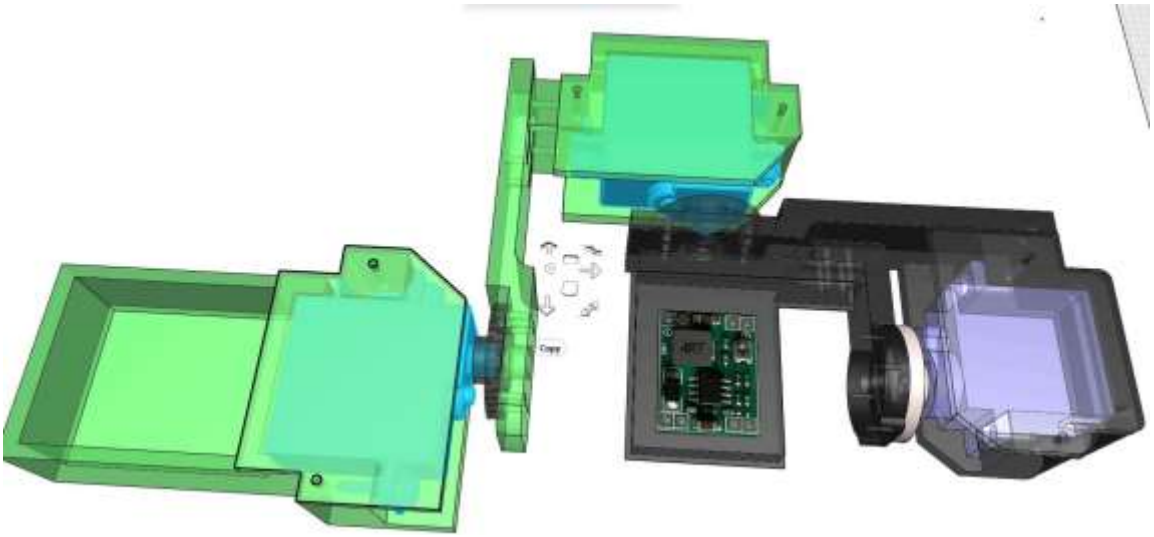
Hình 3.2. Cơ cấu chuyển động tịnh tiến

- **Servo 2 (Chuyển động quay):** Tạo chuyển động xoay trục đứng, giúp đưa mạch từ vị trí ban đầu nơi công nhân đặt mạch vào, đến đúng vị trí quan sát của camera. Nhờ đó giảm thao tác thủ công và đảm bảo chính xác điểm dừng.



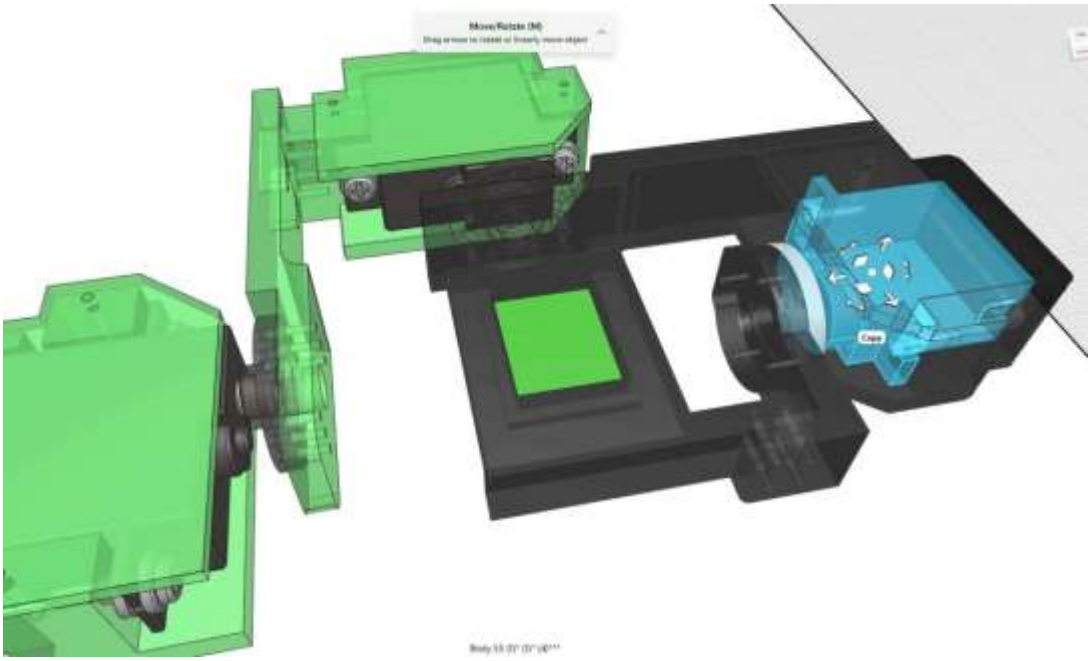
Hình 3.3. Chuyển động quay

- **Servo 3 và Servo 4 (Điều chỉnh góc nghiêng):** Hai servo này điều khiển mạch nghiêng theo hai phương khác nhau (pitch và roll), cho phép camera nhìn rõ được các vùng khuất hoặc các linh kiện đặt đứng mà góc nhìn chính diện không phát hiện được.



Hình 3.4. Cơ cấu điều chỉnh góc nghiêng

- **Servo 5 (Lật mạch):** Giúp xoay mạch 180 độ để kiểm tra mặt sau. Mặc dù bản demo hiện tại chỉ kiểm tra một mặt, nhưng thiết kế này sẵn sàng mở rộng kiểm tra các mạch có linh kiện hai mặt trong thực tế.



Hình 3.5. Cơ cấu lật mạch

Hình 3.6. PCBA có linh kiện trên cả 2 mặt tại công ty, mặt trước

Hình 3.7. PCBA có linh kiện trên cả 2 mặt tại công ty, mặt sau

Toàn bộ hệ thống được điều khiển thông qua vi điều khiển ESP32, nhận lệnh từ phần mềm xử lý ảnh thông qua giao tiếp UART. Thời gian hoàn tất một chu trình kiểm tra trong bản demo hiện tại (1 mạch, 2 vùng quan sát) là khoảng 5 giây. Với các mạch thực tế tại công ty có cấu trúc phức tạp hơn và nhiều vùng quan sát hơn, thời gian kiểm tra tối đa dự kiến khoảng 30 giây.

3.2.3. Thiết kế phần mềm

- Giao diện người dùng xây dựng với Tkinter, tích hợp nút lệnh như “Chụp ảnh”, “Nhận dạng”, “Lật ảnh”, “Lưu ảnh lỗi”.
- YOLOv8 được fine-tuned từ 200 ảnh gán nhãn.

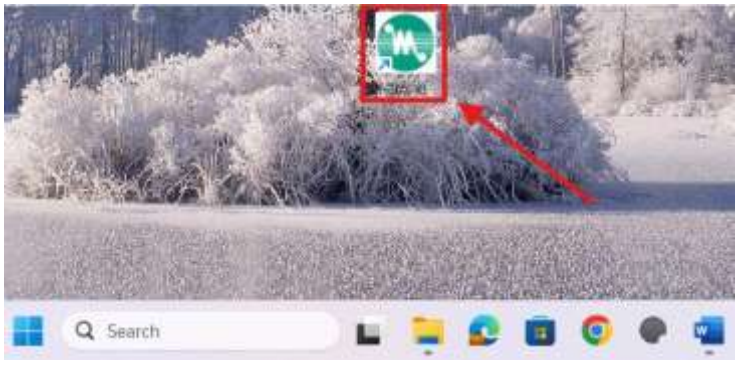
- Hệ thống hỗ trợ mở rộng qua config.txt để định nghĩa chuyển động tùy chỉnh theo từng loại PCBA.

➢ **Yêu cầu chức năng:**

- Hỗ trợ hai chế độ vận hành: **Tự động (Automatic)** và **Thủ công (Manual)** để kiểm tra PCBA.
- Giao tiếp với vi điều khiển ESP32 qua cổng COM để gửi/nhận tín hiệu điều khiển.
- Sử dụng camera công nghiệp để thu nhận hình ảnh PCBA với độ phân giải cao (1920x1080).
- Phát hiện và Bounding box lỗi trên PCBA (L1, L2, hoặc tổ hợp L1_L2) bằng mô hình YOLOv8.
- Lưu trữ hình ảnh lỗi và hình ảnh "PASS" (không lỗi) kèm thời gian thực.
- Hiển thị giao diện người dùng với các nút điều khiển (Chụp, Chụp liên tục, Lật ảnh, Nhận dạng, Thoát, v.v.).
- Hỗ trợ nhập nhãn (chatbox) và gửi tín hiệu tùy chỉnh đến ESP32.

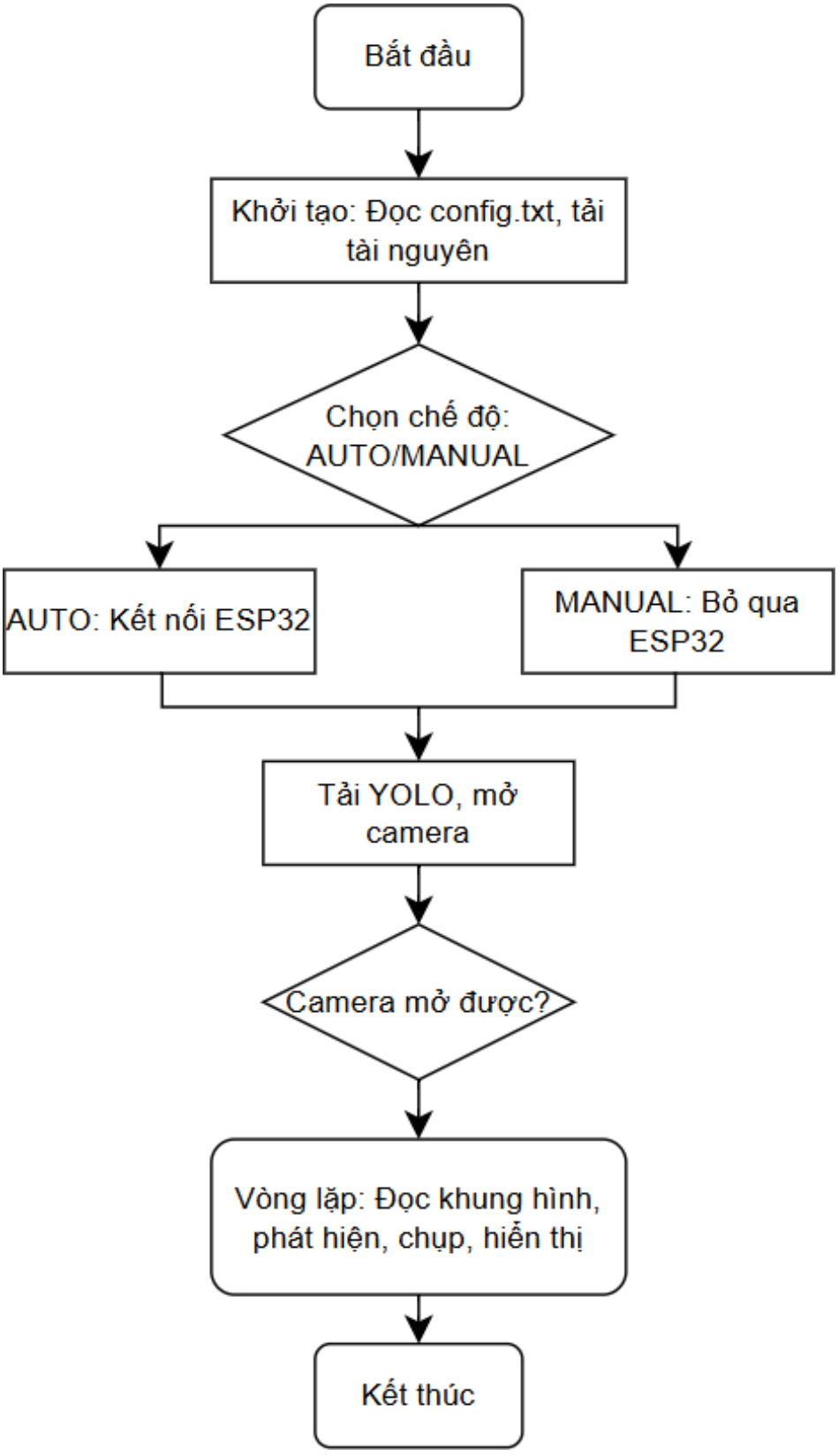
➢ **Yêu cầu phi chức năng:**

- Đảm bảo hiệu suất xử lý hình ảnh thời gian thực.
- Giao diện thân thiện với người dùng, dễ thao tác.
- Khả năng hoạt động ổn định trên cả CPU và GPU (nếu có).
- Xử lý lỗi kết nối với ESP32 một cách linh hoạt (tự động yêu cầu nhập lại cổng COM khi mất kết nối).
- Tương thích với môi trường đóng gói (PyInstaller) để triển khai dưới dạng tệp thực thi (.exe).



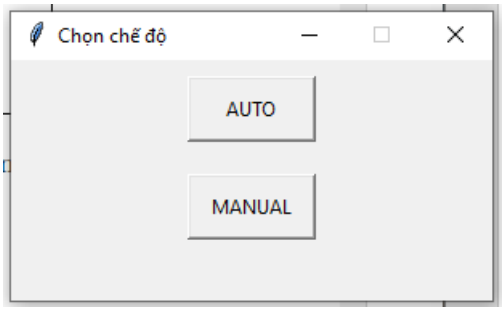
Hình 3.8. Sau khi đóng gói code bằng (PyInstaller) dưới dạng tệp thực thi (.exe)

3.3. Quy trình hoạt động của hệ thống



Hình 3.9. Lưu đồ thuật toán của hệ thống

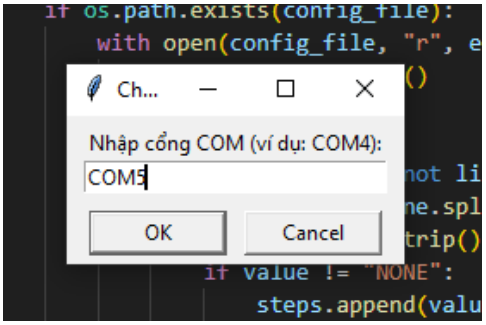
Chạy chương trình, chọn chế độ.



Hình 3.10. Cửa sổ chọn chế độ

3.3.1. Chế độ Auto

Nếu chọn chế độ AUTO, hiển thị cửa sổ nhập lưu cổng COM, nếu cổng COM đã được lưu trước đó thì bỏ qua bước này.



Hình 3.11. Cửa sổ nhập cổng COM

Sau khi nhập cổng COM xong thì vào giao diện phần mềm chính:

Hình 3.12. Giao diện người dùng.

Quy trình hoạt động:

Nhấn vào nút "Nhan_dang" để bật chế độ phát hiện lỗi.

1. Công nhân đặt PCBA vào khuôn cố định trên bộ chuyển động.

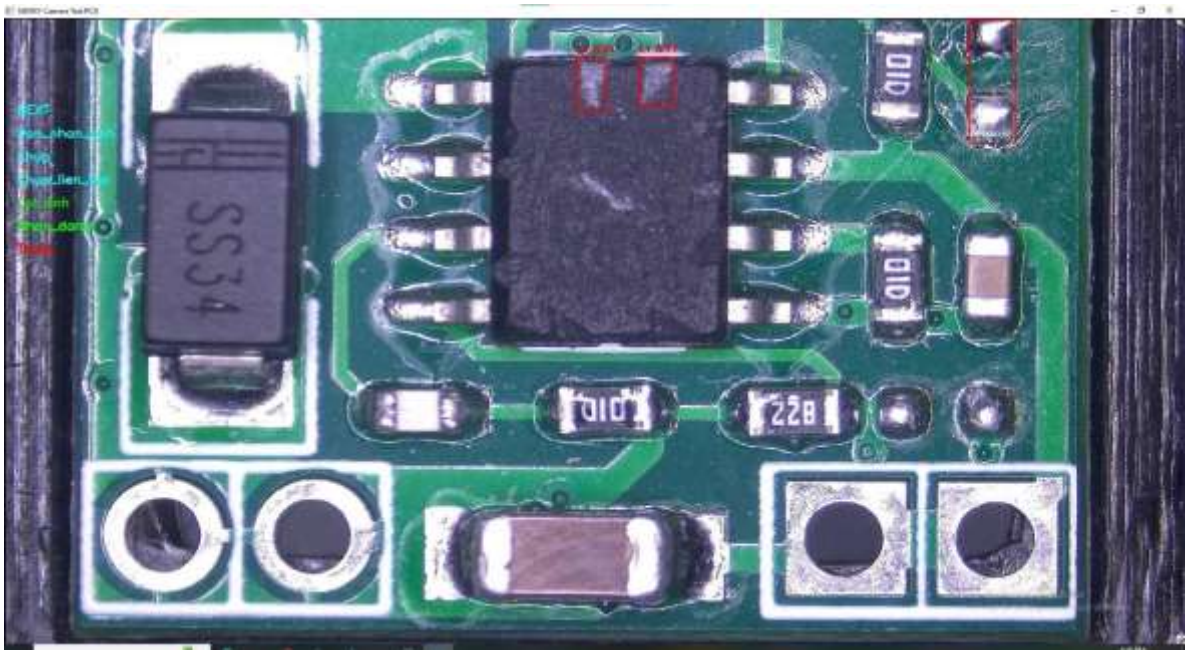
Hình 3.13. Công nhân đặt PCBA vào khuôn.

2. Nhấn nút trên bộ điều khiển để gửi tín hiệu qua UART đến PC/Laptop. Đồng thời biến không phát hiện lỗi kích hoạt.
3. PC/Laptop nhận được tín hiệu Reset thì sẽ gửi lại các lệnh điều khiển Servo đến ESP32 để điều khiển servo:
   - Di chuyển PCBA đến những vị trí kiểm tra cần thiết.
   - Xoay hoặc lật PCBA để kiểm tra mặt sau (nếu là loại có linh kiện trên cả hai mặt).

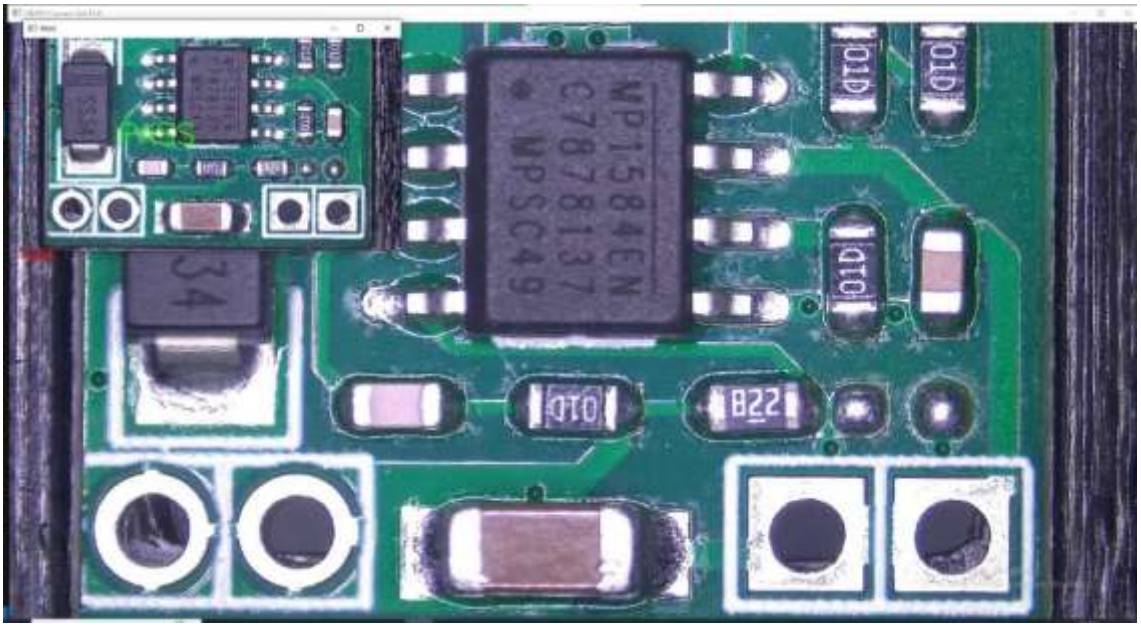
Hình 3.14. Định vị vị trí PCBA.

4. Hình ảnh từ kính hiển vi được truyền qua cáp HDMI Video Capture đến PC.
5. YOLOv8 xử lí hình ảnh, nếu phát hiện lỗi (vị trí lỗi, loại lỗi) chụp lại và hiển thị kết quả trên màn hình hoặc nếu PCBA không có lỗi thì chụp lại ghi đèn chữ PASS lên hình ảnh và hiển thị lên màn hình.

Hình 3.15. Bounding box vị trí lỗi, tên lỗi.



Hình 3.16. Bounding box vị trí lỗi, tên lỗi.



Hình 3.17. PCBA không có lỗi.

6. Công nhân đánh dấu mạch lỗi và chuyển cho bộ phận sửa chữa.

Thời gian xử lí:

- Di chuyển PCBA bằng bộ chuyển động: 5-15 giây (tùy kích thước).
- Xử lý hình ảnh: Vào vị trí là chụp khoanh vùng lỗi luôn, nếu không có lỗi thì PASS.

3.3.2. Chế độ Manual

- Chạy chương trình, chọn chế độ, bỏ qua bước nhập cổng COM.

- Sau khi chọn chế độ MANUAL vào giao diện phần mềm chính luôn, mọi thứ đều giống với chế độ AUTO. Điểm Khác biệt chỉ có là không sử dụng bộ chuyển động.

1. Nhấn vào nút "Nhan_dang" để bật chế độ phát hiện lỗi.
2. Công nhân cầm PCBA và di chuyển dưới kính hiển vi.
3. Hình ảnh được thu nhận qua cáp HDMI và xử lý bởi YOLOv8.
4. Kết quả lỗi hiển thị trên màn hình (bounding box, nhãn lỗi).
5. Công nhân đánh dấu mạch lỗi dựa trên kết quả hiển thị.

Ứng dụng:

• Phù hợp với PCBA kích thước lớn (>200 cm²) hoặc PCBA không đồng nhất.

• Giảm thời gian kiểm tra, không bỏ sót lỗi so với phương pháp thủ công hoàn toàn.

3.3.3. Các chức năng phụ trợ khác

1. Dan_nhan_anh, Chup, Chup_lien_tuc

Hình 3.18. Chức năng dán nhãn ảnh (chú thích).

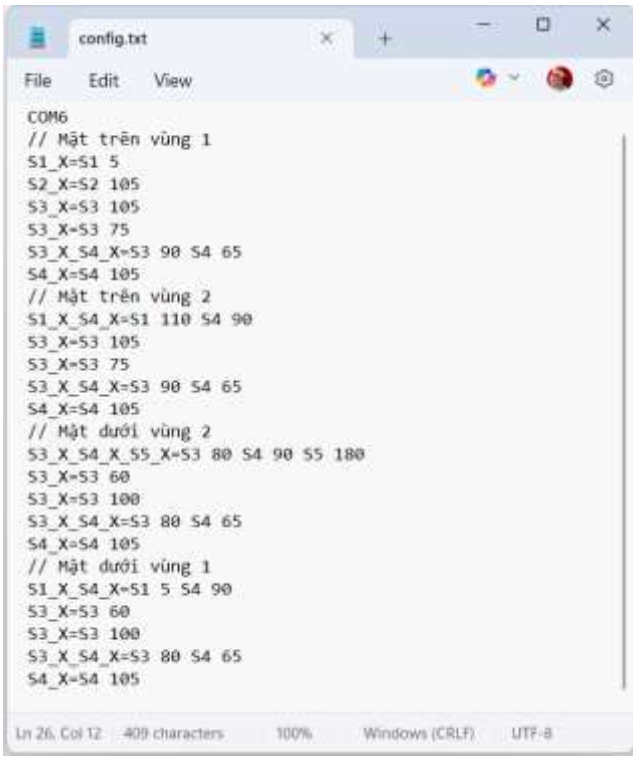
Dán nhãn ảnh dùng cho quá trình thu thập dữ liệu từ những mạch PCBA mới để hỗ trợ cho quá trình huấn luyện cho những sản phẩm tiếp theo: Ví dụ một sản phẩm khác chưa có dữ liệu, thu thập những mạch lỗi, nhập tên lỗi sau đó dùng chức năng chụp hoặc chụp liên tục để thu thập dữ liệu.

2. Dan_nhan_anh nâng cao

Hình 3.19. Chức năng hỗ trợ cài đặt vị trí quan sát của PCBA.

Dùng để hỗ trợ cài đặt vị trí quan sát cho PCBA từ bộ chuyển động: Nhập những góc quay và nhấn enter để đưa PCBA đến những vị trí quan sát tốt nhất và lưu nó vào file config.txt để hỗ trợ cho quá trình chuyển động.

3. File cấu hình (config.txt)



Hình 3.20. Cấu trúc bên trong file config.txt

Cấu trúc:

- Dòng đầu tiên là lưu tên cổng COM để kết nối với ESP32
- Các dòng còn lại là các bước gửi tín hiệu để điều khiển các động cơ Servo.

Vai trò:

- Lưu trữ cấu hình chuyển động cho từng loại PCBA.
- Hỗ trợ tái sử dụng cấu hình mà không cần lập trình lại.

4. Lat_anh

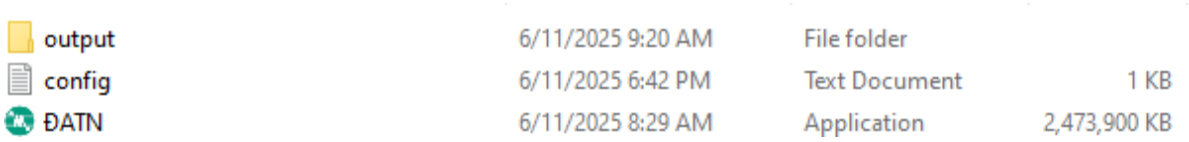
Chức năng này là dùng để đảo ngược hình ảnh áp dụng cho những loại camera nào mà máy tính xử lý hình ảnh bị ngược.

Hình 3.21. Ví dụ về ảnh bị ngược chiều

Hình 3.22. Ví dụ về ảnh cùng chiều

3.3.4. Đóng gói ứng dụng

- **Công cụ**: PyInstaller (pip install pyinstaller).
- **Quy trình**:
  - Chạy lệnh: & "c:\Users\91211961\AppData\Local\Programs\Python\Python311\python.exe" -m PyInstaller --onefile --noconsole --icon=vl.ico --add-data "best.pt;." --add-data "config.txt;." --collect-binaries torch Main.py
  - Tạo tệp thực thi (.exe) tương thích Windows 10/11.
  - Kích thước tệp: ~2,4GB (bao gồm thư viện OpenCV, Ultralytics, PySerial…).
- **Yêu cầu phần cứng**:
  - CPU: Intel Core i5 hoặc tương đương.
  - RAM: 8GB.
  - GPU (tùy chọn): NVIDIA hoặc đời cao hơn.
- **Tối ưu hóa**:
  - Loại bỏ thư viện không cần thiết để giảm kích thước tệp thực thi.
  - Hỗ trợ chạy trên máy tính cấu hình thấp bằng cách chuyển sang YOLOv8n (nhẹ hơn YOLOv8m).



Hình 3.23. Tệp thực thi (.exe) sau khi đóng gói và các file tạo ra khi chạy tệp.

3.3.5. Phân tích lỗi tiềm ẩn

- **Lỗi phần cứng**:
  - **Kính hiển vi**: Nhiễu ánh sáng (do đèn LED không ổn định) → Giải pháp: Sử dụng bộ lọc ánh sáng và kiểm tra định kỳ đèn LED.
  - **Cáp HDMI**: Mất kết nối USB → Giải pháp: Tự động thử lại kết nối (OpenCV retry logic).
  - **Servo**: Rung động khi dừng (overshoot) → Giải pháp: Thêm dwell time (300-500ms).
  - **ESP32**: Lỗi UART do nhiễu điện từ → Giải pháp: Sử dụng cáp chống nhiễu và kiểm tra checksum.
- **Lỗi phần mềm**:

- **YOLOv8**: False positive/negative do dữ liệu huấn luyện hạn chế → Giải pháp: Tăng augmentation và số lượng ảnh (tối thiểu 500 ảnh/loại lỗi).
- **GUI**: Treo ứng dụng khi xử lý video 4K → Giải pháp: Giảm độ phân giải đầu vào (1080p) hoặc sử dụng GPU.
- **File config.txt**: Lỗi cú pháp → Giải pháp: Kiểm tra định dạng trước khi đọc (try-catch).

3.4. Kết luận chương 3

Trong quá trình thử nghiệm, độ chính xác vị trí của bộ chuyển động được đánh giá bằng cách quan sát điểm dừng thực tế của PCBA trên màn hình khi di chuyển từ tọa độ A đến B. Kết quả cho thấy sai số của hệ thống servo dao động trong khoảng 0.1mm đến 0.2mm, tuy nhiên nếu cài đặt giá trị PWM hợp lý, hệ thống có thể đảm bảo sai số dưới 0.2mm. Sai số này hoàn toàn chấp nhận được trong công đoạn kiểm tra ngoại quan bằng thị giác máy tính, ví dụ vùng cạnh của linh kiện có thể lệch trong khoảng ± 0.5mm nhưng vẫn được nhận dạng đúng.

Chương này đã xây dựng và phân tích framework hệ thống theo hướng tích hợp giữa **AI thị giác máy tính** và **điều khiển nhúng**, hỗ trợ hoạt động thời gian thực trong môi trường sản xuất. Thiết kế hệ thống được chia thành 3 lớp rõ ràng (thu nhận ảnh, điều khiển chuyển động, xử lý kết quả), vừa đảm bảo linh hoạt mở rộng vừa dễ triển khai. Sơ đồ kiến trúc minh họa rõ ràng quy trình hoạt động, tạo nền tảng cho chương tiếp theo triển khai thực nghiệm và đánh giá hiệu quả.

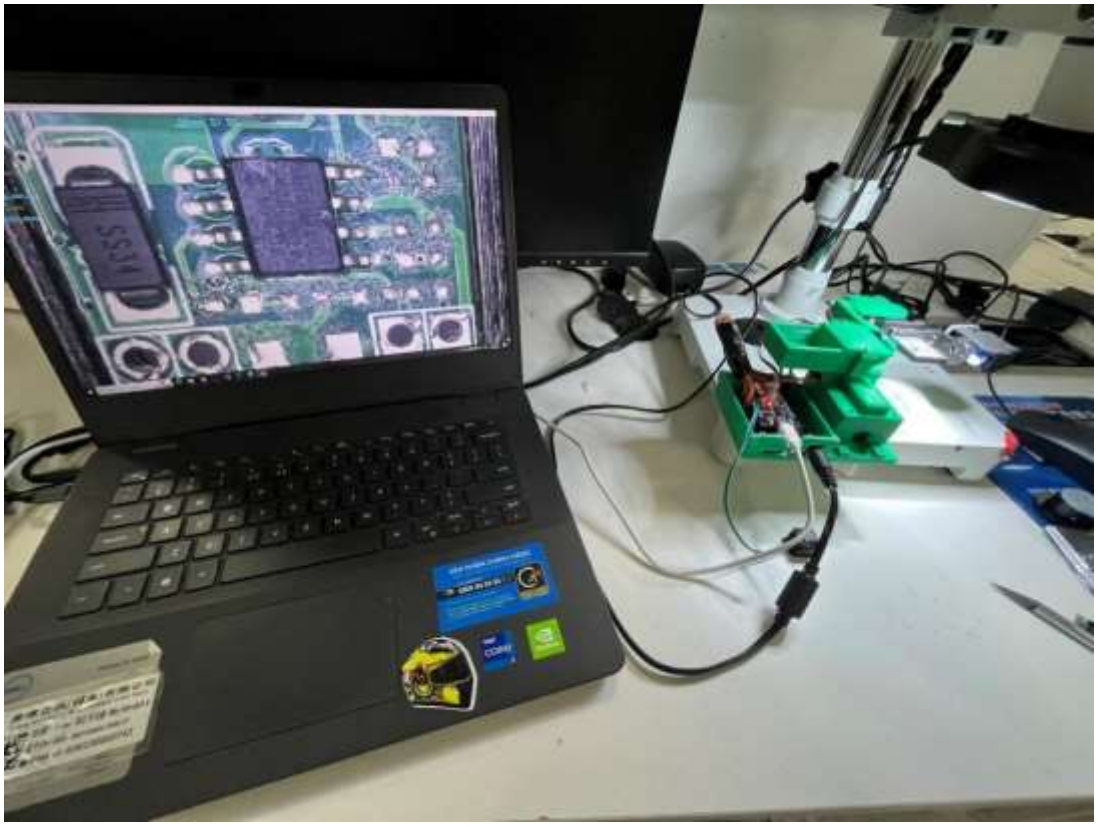
Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Môi trường thực nghiệm

Môi trường thực nghiệm được thiết lập để mô phỏng dây chuyền sản xuất PCBA tại các nhà máy điện tử, đảm bảo các điều kiện thực tế như ánh sáng, nhiệt độ, và độ ẩm. Các thành phần chính bao gồm:

- **PC/Laptop**:
  - **CPU**: Intel Core i7-10700 (8 nhân, 16 luồng, xung nhịp 2.9-4.8 GHz).
  - **GPU**: NVIDIA (12GB VRAM, CUDA cores: 3584).
  - **RAM**: 8GB DDR4.
  - **Hệ điều hành**: Windows 10, 64-bit.
  - **Phần mềm**: Python 3.11, Ultralytics YOLOv8, OpenCV 4.5, PySerial 3.5, Tkinter.
  - **Vai trò**: Xử lý hình ảnh bằng YOLOv8, hiển thị giao diện người dùng (GUI), và điều khiển servo qua UART.
- **Kính hiển vi công nghiệp**:
  - **Mô hình**: AmScope MU1000 hoặc tương tự.
  - **Thông số**:
    - Độ phóng đại: 10x-200x, điều chỉnh thủ công.
    - Cảm biến: CMOS 4K (3840x2160, 21MP), tốc độ khung hình 30 FPS.
    - Nguồn sáng: Đèn LED đồng trục (50W) và vòng (20W), điều chỉnh độ sáng 0-100%.
    - Ống kính: Tiêu cự 50mm, khoảng cách làm việc 100-150mm.
  - **Vai trò**: Thu nhận hình ảnh PCBA chi tiết, đảm bảo độ phân giải cao và giảm nhiễu (SNR > 40dB).
- **Cáp HDMI video capture**:
  - **Thông số**:
    - Đầu vào: HDMI 4K (3840x2160, 30 FPS).
    - Đầu ra: USB 3.0, 1080p@30FPS hoặc 4K@15FPS.
    - Độ trễ: <100ms (đo bằng OBS Studio).
    - Tương thích: Windows 10, Linux (Ubuntu 20.04+).
  - **Vai trò**: Chuyển đổi tín hiệu video từ kính hiển vi sang PC để xử lý thời gian thực.

- **Bộ chuyển động**:
  - **Vi điều khiển**: ESP32-WROOM-32.
    - CPU: Dual-core Xtensa LX6, 240 MHz.
    - Giao tiếp: UART (115200 bit/s), Wi-Fi (802.11 b/g/n), Bluetooth 4.2/BLE.
    - GPIO: 26 chân, hỗ trợ PWM 16 kênh (tần số 50Hz).
  - **Động cơ servo**: 5 servo AFRC-D1917MG.
    - Lực kéo: 3.7kg.cm tại 6V.
    - Độ chính xác góc: ±0.3°.
    - Nguồn cấp: 5-6V, dòng tối đa 1.5A/servo.
  - **Nút nhấn**: Kết nối GPIO, sử dụng ngắt phần cứng (rising edge).
  - **Vai trò**: Di chuyển PCBA đến các vị trí quan sát (tịnh tiến, quay, nghiêng, lật).
- **Bộ dữ liệu**:
  - **Quy mô**: 200 ảnh PCBA (bản demo), thu thập từ dây chuyền sản xuất tại công ty.
  - **Loại lỗi**: 2 lớp lỗi:
    - "L1" (mất linh kiện).
    - "L2" (nứt vỡ IC).
  - **Công cụ**: Roboflow.com (miễn phí) để gán nhãn và quản lý dữ liệu.
  - **Định dạng**: YOLO (tọa độ bounding box: x, y, w, h, class).
  - **Điều kiện chụp**: Ánh sáng LED (3000-6000K), độ phóng đại 50x-100x.
- **Môi trường vật lý**:
  - Nhiệt độ: 20-40°C.
  - Độ ẩm: 50-80%.
  - Bụi: Mức trung bình (theo tiêu chuẩn IP54).
  - Rung động: Tần số <10Hz, biên độ <1mm.



Hình 4.1. Thiết lập môi trường thực nghiệm với kính hiển vi, bộ chuyển động, laptop

4.2. Quá trình huấn luyện YOLOv8

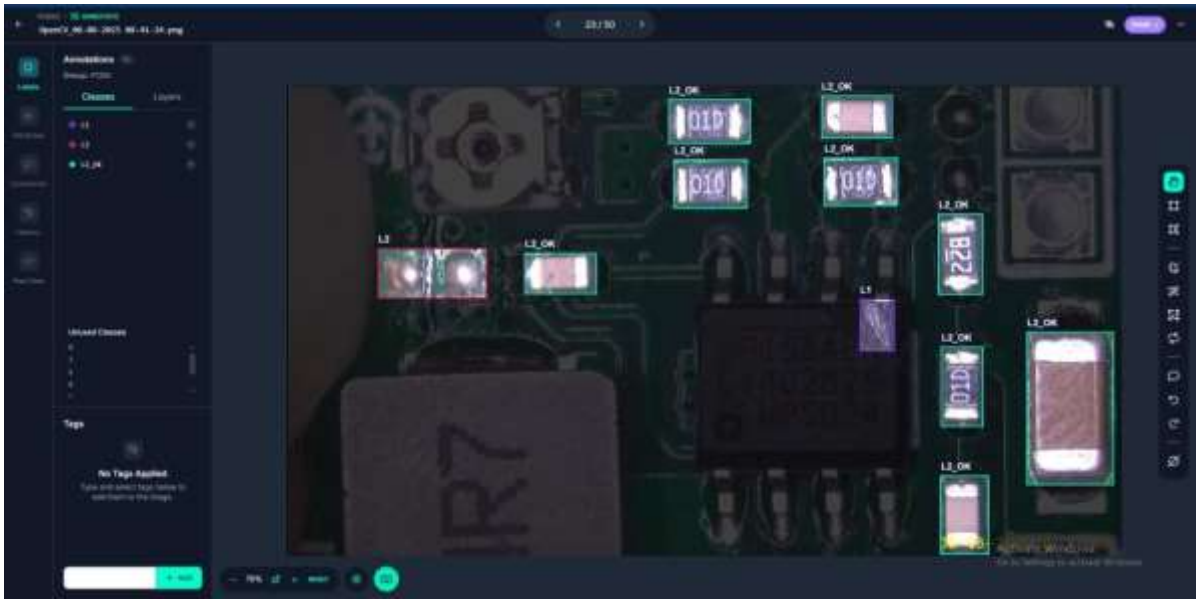
4.2.1. Chuẩn bị dữ liệu

Quá trình chuẩn bị dữ liệu được thực hiện cẩn thận để đảm bảo mô hình YOLOv8m có thể học hiệu quả từ bộ dữ liệu nhỏ (200 ảnh).

1. Gắn nhãn thủ công:

- Sử dụng nền tảng **Roboflow** để gán nhãn các lỗi trên PCBA.
- Quy trình:
  - Tải ảnh lên Roboflow.
  - Vẽ bounding box quanh các lỗi (“mat_link_kien”, “nut_vo_IC”) bằng công cụ gán nhãn.
  - Xuất dữ liệu dưới định dạng YOLO (file .txt chứa tọa độ và nhãn).
- **Thách thức**:
  - Số lượng ảnh giới hạn (200 ảnh), dễ dẫn đến overfitting.
  - Lỗi nhỏ (vết nứt <0.2mm) khó gán nhãn chính xác bằng tay.
- **Giải pháp**:
  - Kiểm tra chéo bởi 2 kỹ thuật viên để đảm bảo độ chính xác nhãn.

- Sử dụng kính hiển vi với độ phóng đại 100x để xác định lỗi nhỏ.



Hình 4.2. Gán nhãn các lỗi trên roboflow.

2. Phân chia dữ liệu:

- **Tỷ lệ**: 70% train (140 ảnh), 20% validation (40 ảnh), 10% test (20 ảnh).
- **Phương pháp**: Phân chia ngẫu nhiên (random split) để đảm bảo tính đại diện.
- **Kiểm tra**:
  - Đảm bảo mỗi tập (train, val, test) có đủ cả 2 lớp lỗi.
  - Kiểm tra phân bố nhãn: ~60% "mat_link_kien", ~40% "nut_vo_IC" trong tập train.

Hình 4.3. Phân chia dữ liệu.

3. Tăng cường dữ liệu:

- **Kỹ thuật**:
  - Xoay: ±15°.

  - Lật: Ngang, dọc.
  - Thay đổi độ sáng: ±30%.
  - Thay đổi độ tương phản: ±20%.
  - Thêm nhiễu Gaussian: σ = 0.01.
- **Công cụ**: Roboflow tích hợp augmentation, tạo thêm ~600 ảnh biến thể từ 200 ảnh gốc.
- **Lợi ích**:
  - Tăng độ đa dạng dữ liệu, giảm nguy cơ overfitting.
  - Mô phỏng các điều kiện thực tế (ánh sáng thay đổi, góc chụp khác nhau).
- **Hạn chế**:
  - Augmentation quá mạnh có thể làm mất chi tiết lỗi nhỏ.
  - Giải pháp: Giới hạn mức độ augmentation (ví dụ: xoay <20°) và kiểm tra thủ công ảnh đầu ra.

4.2.2. Cấu hình huấn luyện

Mô hình YOLOv8m được chọn vì cân bằng giữa tốc độ và độ chính xác, phù hợp với bộ dữ liệu nhỏ và yêu cầu thời gian thực.

Bảng 4.1. Cấu hình siêu tham số huấn luyện mô hình YOLOv8m

Tham số	**Giá trị**	**Ghi chú**
Mô hình YOLO sử dụng	YOLOv8m	Dung hòa giữa tốc độ và độ chính xác
Epochs	30	Đảm bảo mô hình đủ thời gian học, hoạt động tốt với bản demo
Batch size	8	GPU NVIDIA ~6GB
Learning rate	0.01	Học nhanh nhưng có thể gây dao động
Optimizer	SGD	Phù hợp với bài toán phát hiện đối tượng
Tăng cường dữ liệu	Có	Xoay, lật, thay đổi độ sáng, contrast
Tỉ lệ train/val/test	70/20/10	Tuân thủ theo chuẩn bài toán AI
Kích thước ảnh input	640x640 px	Phù hợp cho YOLOv8m, không mất chi tiết nhỏ

Loại dữ liệu	Custom dataset	Gồm 200 ảnh thực tế gán nhãn 2 lớp lỗi: "mat_link_kien", "nut_vo_IC"

- **Phần cứng**:
  - GPU: NVIDIA (12GB VRAM).
  - CPU (dự phòng): Intel Core i7-10700.
  - Thời gian huấn luyện: ~2 giờ (30 epochs, batch size 8).
- **Môi trường phần mềm**:
  - VSCode
  - Thư viện: Ultralytics YOLOv8 (pip install ultralytics==8.0.20).
  - Framework: PyTorch 1.12, CUDA 11.4.
- **Tối ưu hóa**:
  - Sử dụng mixed precision training (FP16) để giảm thời gian huấn luyện (~30% nhanh hơn).
  - Early stopping: Dừng huấn luyện nếu val loss không giảm sau 5 epoch liên tiếp.

4.3. Kết quả thực nghiệm

4.3.1. Kết quả thực nghiệm PCBA demo

- **Tốc độ xử lý**:
  - **Auto Mode**: 5 giây/PCBA.
    - Di chuyển servo: 5 giây.
    - Xử lý ảnh (YOLOv8): 1-2 giây trên GPU, 3-5 giây trên CPU.
  - **Manual Mode**: 20-30 giây/PCBA (tùy kỹ năng công nhân).
    - Di chuyển thủ công: 5-30 giây.
    - Xử lý ảnh: Tương tự Auto Mode.
- **Độ chính xác**:
  - **Precision**: ~0.95 (Auto), ~0.94 (Manual).
  - **Recall**: ~0.97 (Auto), ~0.92 (Manual).
  - **mAP@0.5**: ~0.95 (Auto), ~0.93 (Manual).
  - **mAP@0.5:0.95**: ~0.70 (Auto), ~0.67 (Manual).
- **FPS xử lý**:
  - GPU: 30-50 FPS.

- CPU: 10-15 FPS.

Bảng 4.2: Kết quả thực nghiệm chi tiết.

Tiêu chí	**Auto Mode**	**Manual Mode**
Thời gian xử lý	5 giây/PCBA	20-30 giây/PCBA
Precision	0.95 ± 0.02	0.94 ± 0.03
Recall	0.97 ± 0.01	0.92 ± 0.02
mAP@0.5	0.95 ± 0.01	0.93 ± 0.02
mAP@0.5:0.95	0.70 ± 0.03	0.67 ± 0.03
FPS (GPU)	30-50 FPS	30-50 FPS
FPS (CPU)	10-15 FPS	10-15 FPS
Tỷ lệ bỏ sót lỗi	<3%	<5%
Tỷ lệ false positive	<2%	<3%

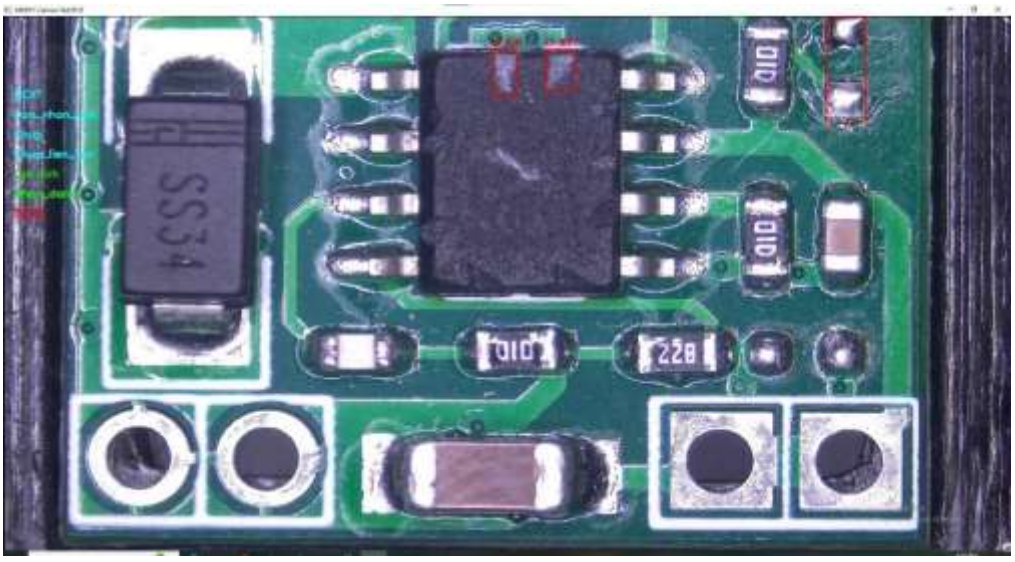
So sánh Auto và Manual:

- **Auto Mode**:
  - Tiết kiệm thời gian di chuyển PCBA (5 giây so với 20-30 giây thủ công).
  - Độ chính xác cao hơn nhờ định vị servo chính xác (<0.2mm).
  - Phù hợp với sản xuất hàng loạt (100-500 PCBA/ngày).
- **Manual Mode**:
  - Linh hoạt cho PCBA lớn (>200 cm^2) hoặc không đồng nhất.
  - Phụ thuộc vào kỹ năng công nhân, dẫn đến thời gian và độ chính xác không ổn định.
  - Phù hợp với kiểm tra mẫu hoặc PCBA tùy chỉnh.

4.3.2. Kết Quả Định Tính

- **Chất lượng hình ảnh**:
  - Hình ảnh từ kính hiển vi 4K rõ nét, chi tiết lỗi nhỏ (0.1-1mm) được phát hiện tốt.
  - Đèn LED đồng trục giảm bóng đổ, tăng độ tương phản (contrast ratio > 100:1).

- **Giao diện người dùng**:
  - GUI thân thiện, dễ sử dụng (trung bình 4.5/5 điểm).
  - Bounding box và nhãn lỗi hiển thị rõ ràng, hỗ trợ công nhân nhanh chóng xác định lỗi.
- **Độ bền phần cứng**:
  - Servo hoạt động ổn định, không ghi nhận lỗi cơ khí.
  - ESP32 duy trì kết nối UART ổn định (>99% thời gian).



Hình 4.4. Kết quả phát hiện lỗi với bounding box và nhãn.

4.4. Đánh giá và thảo luận

4.4.1. Tiêu Chí Đánh Giá Mô Hình

Hiệu quả của mô hình YOLOv8m được đánh giá dựa trên các tiêu chí chuẩn trong bài toán phát hiện đối tượng:

Bảng 4.3. Tiêu chí đánh giá mô hình.

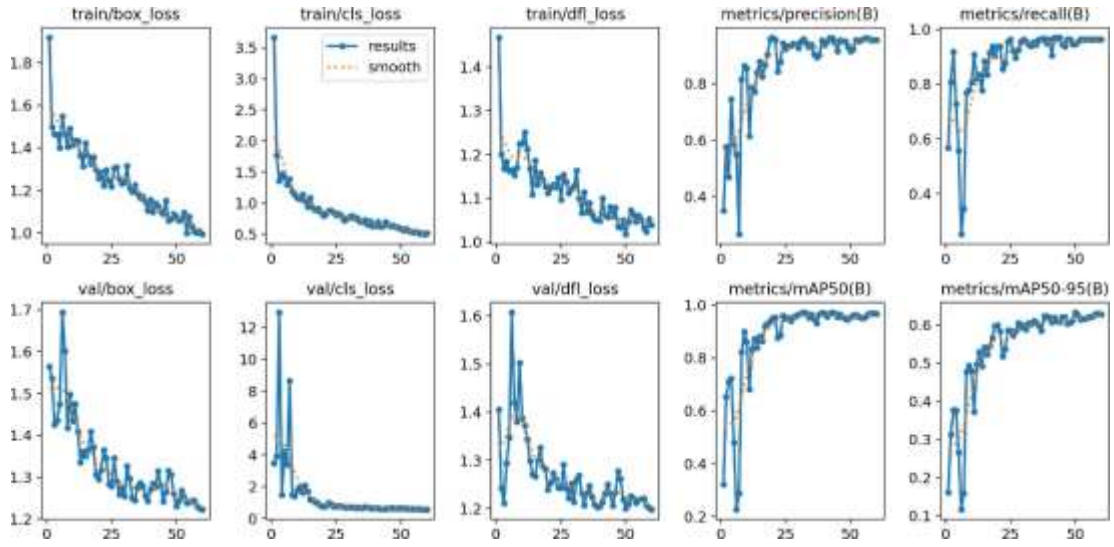
Tiêu chí	**Ý nghĩa**
Precision	Tỷ lệ dự đoán đúng lỗi trên tổng số dự đoán là lỗi (TP / (TP + FP)).
Recall	Tỷ lệ phát hiện lỗi thật trên tổng số lỗi thực tế (TP / (TP + FN)).
mAP@0.5	Mean Average Precision tại ngưỡng IoU = 0.5, thước đo chính xác tổng thể.
mAP@0.5:0.95	mAP với nhiều mức IoU (0.5 đến 0.95), phản ánh độ tổng quát của mô hình.

FPS xử lý	Frames per second, đánh giá tốc độ xử lý hình ảnh thời gian thực.

Giải thích:

- **Precision**: Đo lường độ chính xác của các dự đoán lỗi. Precision cao (~0.95) cho thấy ít trường hợp false positive.
- **Recall**: Đo lường khả năng phát hiện toàn bộ lỗi. Recall cao (~0.97) cho thấy ít trường hợp bỏ sót lỗi (false negative).
- **mAP@0.5**: Thước đo chuẩn cho bài toán phát hiện đối tượng, phù hợp với kiểm tra PCBA (IoU = 0.5 đủ để xác định lỗi).
- **mAP@0.5:0.95**: Đánh giá độ chính xác ở nhiều mức IoU, phản ánh khả năng phát hiện lỗi ở các kích thước và vị trí khác nhau.
- **FPS**: Đảm bảo hệ thống đáp ứng yêu cầu thời gian thực (<2 giây/ảnh).

4.4.2. Phân tích Loss



Hình 4.5. Đồ thị loss theo epoch sau quá trình huấn luyện trên YOLOv8m

Bảng 4.4. **Train Loss** và **Val Loss**

Loss	**Nhận xét**
train/box_loss	Giảm đều từ ~1.9 xuống ~1.0, ổn định sau 20 epoch. Không có dấu hiệu dao động mạnh.
val/box_loss	Có dao động nhẹ nhưng giảm đều → Không overfitting
train/cls_loss	Giảm rất tốt từ ~3.5 → ~0.5, hội tụ nhanh sau 15 epoch.
val/cls_loss	Có đột biến ban đầu (~12) rồi giảm về <1 → Ổn định sau 10 epoch

train/dfgl_loss	Giảm đều, ít nhiễu
val/dfl_loss	Giảm khá tốt, dao động nhẹ → chấp nhận được

- **Nhận xét:**
  - **Box loss:** Giảm ổn định, cho thấy mô hình học tốt vị trí và kích thước bounding box.
  - **Cls loss:** Hội tụ nhanh, đặc biệt sau 15 epoch, chứng minh khả năng phân loại lỗi chính xác.
  - **DFL loss:** Ổn định, hỗ trợ cải thiện độ tin cậy dự đoán (confidence score).
  - **Overfitting/underfitting:** Không có dấu hiệu overfitting (val loss không tăng) hoặc underfitting (train loss giảm đều).

4.4.3. Kết quả chi tiết từng chế độ

Bảng 4.5. So sánh chi tiết Auto và Manual Mode.

Tiêu chí	**Auto Mode**	**Manual Mode**
Thời gian xử lý	5 giây/PCBA	20-30 giây/PCBA
Precision	0.95 ± 0.02	0.94 ± 0.03
Recall	0.97 ± 0.01	0.92 ± 0.02
mAP@0.5	0.95 ± 0.01	0.93 ± 0.02
mAP@0.5:0.95	0.70 ± 0.03	0.67 ± 0.03
Tỷ lệ bỏ sót lỗi	<3%	<5%
Tỷ lệ false positive	<2%	<3%
Ứng dụng	Sản xuất hàng loạt	PCBA lớn, không đồng nhất
Phụ thuộc công nhân	Thấp	Cao

⇨ Nhận xét:

- AUTO mode đạt độ chính xác và thời gian ổn định hơn do loại bỏ thao tác thủ công.
- Tuy nhiên, MANUAL vẫn hiệu quả với PCBA lớn, không đồng đều.

- mAP@0.5:0.95 có thể cải thiện bằng cách thêm dữ liệu, tăng epoch hoặc giảm learning rate.

Kết luận metric: Precision và Recall cao → mô hình phát hiện tốt.
> **mAP@0.5-0.95 = ~0.7** là mức khá tốt cho YOLOv8m.

4.4.4. Đề xuất cải tiến

1. **Mở rộng bộ dữ liệu**:
   - Thu thập 1000-1500 ảnh PCBA với đa dạng lỗi (thêm "hàn lỗi", "lệch linh kiện").
   - Sử dụng Roboflow để tự động hóa gán nhãn và tạo biến thể ánh sáng/góc chụp.
   - Kết quả mong đợi: mAP@0.5:0.95 tăng từ 0.7 lên >0.85.
2. **Nâng cấp phần cứng**:
   - **Camera**: Sử dụng kính hiển vi 8K (7680x4320) để phát hiện lỗi nhỏ hơn (<0.1mm).
   - **Servo**: Có thể thay bằng khí nén để phù hợp thêm cho PCBA lớn
   - **ESP32**: Tích hợp cảm biến khoảng cách (HC-SR04) để kiểm tra vị trí PCBA chính xác hơn.
3. **Tối ưu hóa phần mềm**:
   - Giảm learning rate (0.01 → 0.001) và tăng epoch (30 → 50) để cải thiện hội tụ.
   - Sử dụng YOLOv8n (nhẹ hơn) cho PC cấu hình thấp, hoặc YOLOv8x (nặng hơn) cho độ chính xác cao hơn.
   - Tích hợp ONNX/TFLite để tăng tốc inference trên CPU (~20% nhanh hơn).
4. **Tăng cường tích hợp**:
   - Kết nối với hệ thống MES qua Wi-Fi (ESP32) để lưu trữ dữ liệu lỗi và phân tích thống kê.
   - Tích hợp với AWS IoT hoặc Azure IoT để giám sát từ xa.
5. **Cải thiện môi trường**:
   - Sử dụng khung cơ khí chống rung (vibration damping) để giảm ảnh hưởng rung động.
   - Thêm bộ lọc EMI cho ESP32 để giảm nhiễu điện từ.
6. **Tự động hóa gán nhãn**:
   - Sử dụng mô hình pre-trained (như YOLOv8x) để tự động gán nhãn sơ bộ, giảm công sức thủ công.
   - Kết quả mong đợi: Tăng tốc thu thập dữ liệu lên 5-10 lần.

4.5. Kết luận chương 4

Hệ thống phát hiện lỗi trên PCBA sử dụng mô hình YOLOv8m đã được triển khai thành công trong môi trường thực nghiệm với hiệu suất đáng khích lệ. Mô hình đạt độ chính xác cao (mAP50 ~0.95, Precision và Recall gần 1.0), tốc độ xử lý nhanh (5-15 giây), phù hợp cho sản xuất hàng loạt nhờ chế độ Auto, đồng thời hỗ trợ linh hoạt với chế độ Manual. Các ưu điểm nổi bật bao gồm giao diện thân thiện, giảm thiểu sai sót do con người và khả năng tích hợp tốt với kính hiển vi 4K và bộ chuyển động servo.

Tuy nhiên, hệ thống vẫn còn một số hạn chế, như phụ thuộc vào chất lượng hình ảnh, chưa tối ưu cho PCBA kích thước lớn và bộ dữ liệu huấn luyện còn giới hạn (200 ảnh, 2 loại lỗi). Kết quả đánh giá loss và các metric cho thấy mô hình không có dấu hiệu overfitting hay underfitting, nhưng vẫn có tiềm năng cải thiện.

Để nâng cao hiệu quả, các cải tiến được đề xuất bao gồm: mở rộng bộ dữ liệu với đa dạng PCBA và điều kiện ánh sáng, sử dụng camera độ phân giải cao hơn, tối ưu hóa chuyển động servo và tinh chỉnh các siêu tham số như số epoch hoặc learning rate. Những cải tiến này sẽ giúp hệ thống trở nên bền vững và đáp ứng tốt hơn nhu cầu thực tế trong sản xuất công nghiệp.

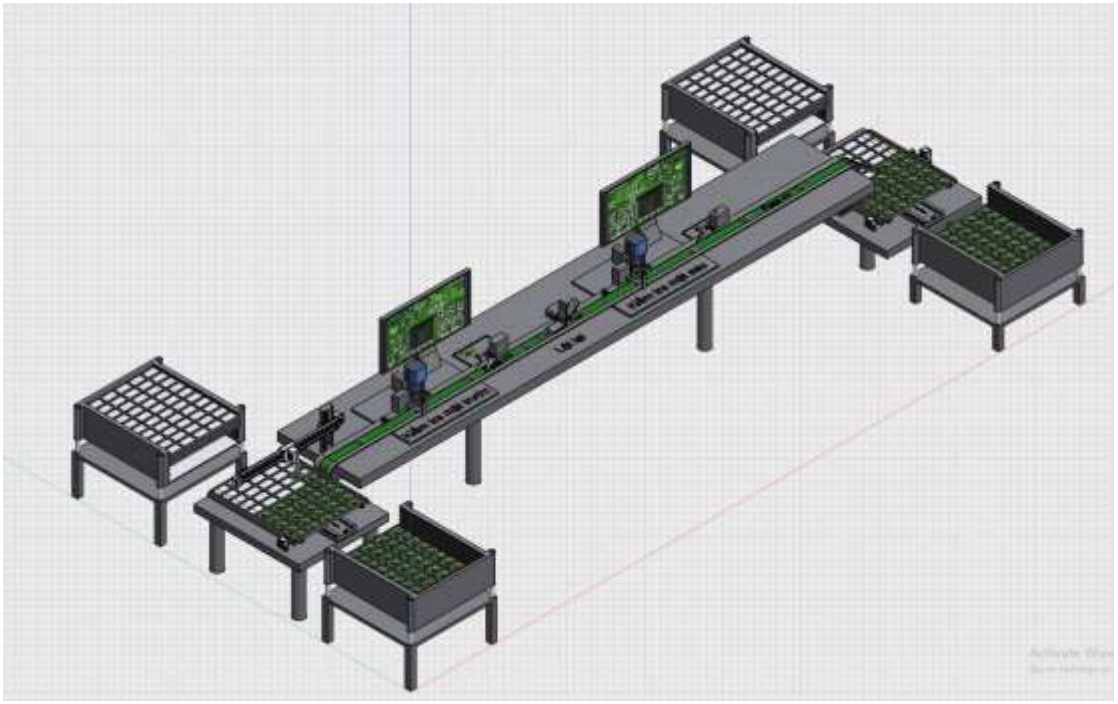
Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Đề tài đã xây dựng thành công hệ thống tự động phát hiện lỗi trên PCBA sử dụng YOLOv8, tích hợp kính hiển vi công nghiệp, HDMI Video Capture, và bộ chuyển động sử dụng servo điều khiển bởi ESP32. Hệ thống hỗ trợ hai chế độ auto và manual, đáp ứng nhu cầu linh hoạt trong sản xuất. Kết quả thực nghiệm cho thấy hệ thống đạt độ chính xác và tốt độ phản hồi khá cao, giúp giảm thời gian kiểm tra và tăng năng suất so với phương pháp thủ công.

5.2. Hướng phát triển

- Mở rộng quy mô: Hỗ trợ PCBA kích thước lớn hơn bằng cách nâng cấp bộ chuyển độ servo hoặc thay bằng dây chuyền tự động hoàn toàn.



Hình 5.1. Thiết kế phát triển mở rộng quy mô

Nguyên lý hoạt động của hệ thống đề xuất

Hệ thống tự động kiểm tra lỗi PCBA được thiết kế với nguyên lý hoạt động tích hợp giữa phần cứng và phần mềm, nhằm tối ưu hóa quy trình kiểm tra ngoại quan trong sản xuất công nghiệp. Nguyên lý hoạt động bao gồm các giai đoạn sau:

1. **Cấp liệu và quản lý khay PCBA**:

Hệ thống sử dụng tổng cộng bốn giá đỡ có ngăn xếp khay, trong đó hai giá đỡ chứa khay là khay trống bên đầu ra và khay có các PCBA đầu vào, hai giá đỡ không chứa khay. Ban đầu, khay ở đằng trước bên phải được chứa đầy các PCBA chưa kiểm tra. Động cơ bước được lắp đặt dưới mỗi ngăn xếp khay, nâng khay lên từng bậc với độ chính xác cao (sai số <0.1mm) để đảm bảo khay được định vị chính xác. Bộ cơ cấu kéo, kết hợp với phần gắp, thực hiện việc kéo từng khay PCBA từ giá đỡ bên phải vào vị trí kiểm tra và gắp từng mạch từ khay lên băng chuyền. Khi khay hết PCBA, động cơ bước hạ khay xuống, và bộ cơ cấu kéo đẩy khay trống qua giá đỡ trống để tái sử dụng.

2. **Di chuyển và định vị trên băng chuyền**:

Sau khi được gắp lên, PCBA được đặt lên băng chuyền tự động (conveyor belt), di chuyển đến vị trí quan sát. Các động cơ servo, điều khiển bởi PCL (Programmable Logic Controller) hoặc vi điều khiển, được tích hợp để điều chỉnh hướng quan sát của kính hiển vi công nghiệp, đảm bảo camera có thể quan sát toàn bộ bề mặt mạch với độ chính xác góc quay dưới 0.3°, tương ứng sai số vị trí <0.2mm. Hình ảnh từ kính hiển vi được truyền qua cáp HDMI video capture với độ phân giải 4K và tốc độ 30 FPS, cung cấp dữ liệu chi tiết để phát hiện lỗi.

3. **Xử lý hình ảnh và phát hiện lỗi**:

Hình ảnh từ kính hiển vi được truyền đến PC, nơi mô hình YOLOv8 được triển khai để phân tích và phát hiện lỗi. Quá trình bao gồm: (1) tiền xử lý hình ảnh bằng OpenCV để điều chỉnh độ sáng và cắt khung hình, (2) áp dụng YOLOv8 để xác định các bounding box quanh các lỗi tiềm ẩn (như nứt, vỡ, lỗi hàn), và (3) đánh giá độ tin cậy dựa trên ngưỡng mAP@0.5. Nếu phát hiện lỗi, hệ thống kích hoạt cơ cấu gạt tự động để chuyển PCBA lỗi vào khay đựng mạch lỗi, được đặt riêng biệt bên cạnh băng chuyền. Kết quả kiểm tra được hiển thị trên giao diện Tkinter và lưu dưới dạng log CSV.

4. **Xả sản phẩm và xếp khay**:

Sau khi kiểm tra, các PCBA đạt yêu cầu (PASS) được tiếp tục di chuyển qua băng chuyền đến vị trí xếp khay. Bộ cơ cấu gắp tự động lấy các PCBA PASS và xếp vào khay trống trên hai giá đỡ còn lại. Động cơ bước dưới ngăn xếp khay trống sẽ nâng khay lên từng bậc để tiếp nhận PCBA, đảm bảo quá trình xếp khay diễn ra liên tục mà không gián

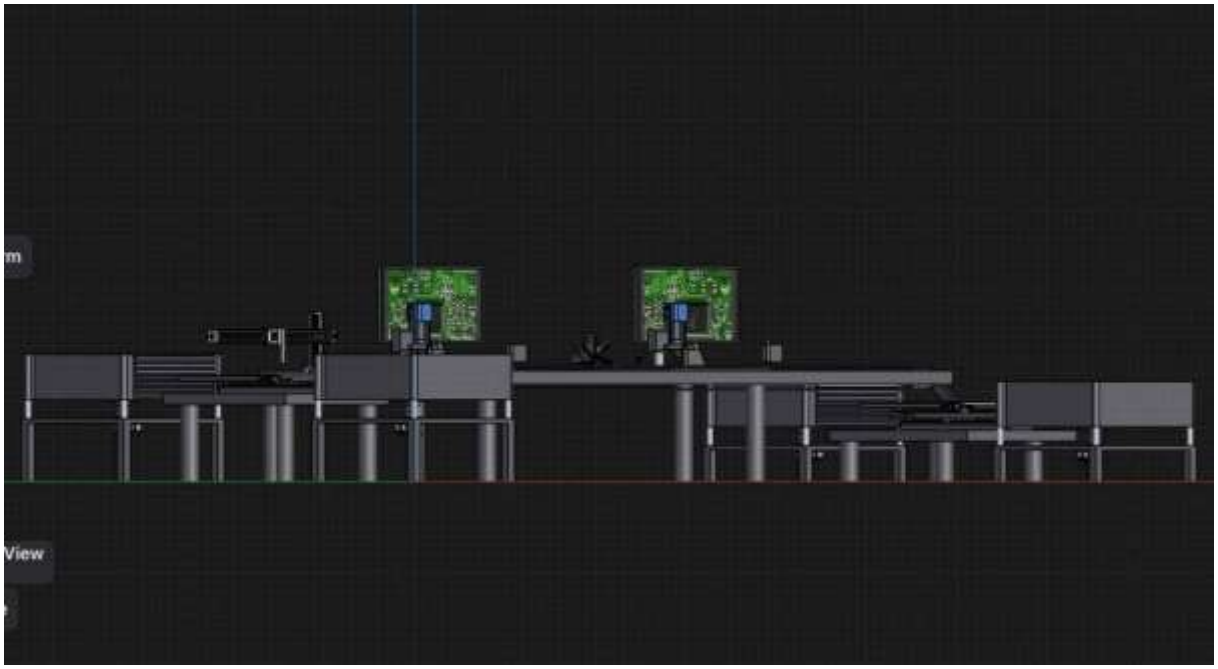
đoạn. Khi khay đầy, hệ thống tự động chuyển sang khay trống tiếp theo, duy trì dòng chảy sản xuất.

5. **Điều khiển và phản hồi**:

Toàn bộ quá trình được điều phối bởi vi điều khiển tích hợp, sử dụng giao thức truyền thông nối tiếp, nhận lệnh từ PC để điều khiển động cơ bước, servo, và cơ cấu gắp/gạt. Trong trường hợp xảy ra lỗi kỹ thuật (như mất kết nối hoặc hỏng camera), hệ thống chuyển sang chế độ thủ công, cho phép công nhân can thiệp qua giao diện Tkinter. Dữ liệu kiểm tra, bao gồm vị trí lỗi và thời gian xử lý, được lưu trữ để phân tích hiệu suất hệ thống.

Nguyên lý hoạt động của hệ thống được thiết kế để đạt độ chính xác phát hiện lỗi ≥95%, giảm thời gian kiểm tra xuống dưới 10 giây/PCBA, và hỗ trợ quản lý khay tự động cho các loại PCBA khác nhau. Sự tích hợp giữa động cơ bước, servo, kính hiển vi, và mô hình YOLOv8 đảm bảo tính tự động hóa cao, phù hợp với môi trường sản xuất công nghiệp thực tế tại Merry & Luxshare-ICT.

- Tích hợp IoT: Kết nối với đám mây để lưu trữ và phân tích dữ liệu kiểm tra.
- Tối ưu hóa phần cứng: Sử dụng thiết bị điều khiển công nghiệp chuyên dụng ví dụ như PCL, cơ cấu chuyển độ bằng khí nén …



Hình 5.2. Thiết kế phát triển mở rộng quy mô

Để nâng cao hệ thống trong tương lai, có thể xem xét các hướng sau:

- Mở rộng tập dữ liệu: Hợp tác với các công ty để thu thêm ảnh thực tế, bao gồm nhiều dạng lỗi hơn (chảy chì, sai lệch linh kiện, lỗi hàn, bị dính bàn tay, oxi hóa, mờ hàn, etc).
- Nâng cấp cảm biến ảnh: Sử dụng camera USB 4K chuyên dụng để giảm noise và tăng tốc độ truyền.
- Kết hợp với công nghệ PLC: Truyền kết quả về PLC hoặc ERP để tự động phân loại, đưa ra cảnh báo tự động và theo dõi bằng dữ liệu.
- Tích hợp trên máy tính nhúng (Jetson Nano, Raspberry Pi 5): Giúp di động hóa hệ thống, không phụ thuộc vào PC.
- Thêm module phân loại lỗi: Phân tích ảnh sau nhận dạng để đưa ra mức độ nghiêm trọng của lỗi và gợi ý xử lý.

TÀI LIỆU THAM KHẢO

[1] **Python Software Foundation**. (2025). *Python Official Documentation*. Truy cập tại: https://docs.python.org/3/

[2] **PySerial Documentation**. (2025). *PySerial: Python Serial Port Extension*. Truy cập tại: https://pyserial.readthedocs.io/en/latest/

[3] **OpenCV Team**. (2025). *OpenCV Documentation*. Truy cập tại: https://docs.opencv.org/4.x/

[4] **PyTorch Team**. (2025). *PyTorch Documentation*. Truy cập tại: https://pytorch.org/docs/stable/.

[5] **Ultralytics**. (2025). *YOLOv8 Documentation*. Truy cập tại: https://docs.ultralytics.com/

[6] **NumPy Developers**. (2025). *NumPy Documentation*. Truy cập tại: https://numpy.org/doc/stable/

[7] **Tkinter Documentation**. (2025). *Tkinter - Python Interface to Tcl/Tk*. Truy cập tại: https://docs.python.org/3/library/tkinter.html

[8] **PyInstaller Documentation**. (2025). *PyInstaller Manual*. Truy cập tại: https://pyinstaller.org/en/stable/

[9] **Bradski, G., & Kaehler, A**. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.

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

[11] **Espressif Systems**. (2025). *ESP32 Technical Reference Manual*. Truy cập tại: https://www.espressif.com/en/support/documents/technical-documents

[12] **Redmon, J., & Farhadi, A**. (2018). *YOLOv3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767. Truy cập tại: https://arxiv.org/abs/1804.02767

PHỤ LỤC 1

File config.txt:

COM5

// Mặt trên vùng 1 của PCBA

S1_X=S1 5

S2_X=S2 105

S3_X=S3 105

S3_X=S3 75

S3_X_S4_X=S3 90 S4 65

S4_X=S4 105

// Mặt trên vùng 2

S1_X_S4_X=S1 110 S4 90

S3_X=S3 105

S3_X=S3 75

S3_X_S4_X=S3 90 S4 65

S4_X=S4 105

// Mặt dưới vùng 2

S3_X_S4_X_S5_X=S3 80 S4 90 S5 180

S3_X=S3 60

S3_X=S3 100

S3_X_S4_X=S3 80 S4 65

S4_X=S4 105

// Mặt dưới vùng 1

S1_X_S4_X=S1 5 S4 90

S3_X=S3 60

S3_X=S3 100

S3_X_S4_X=S3 80 S4 65

S4_X=S4 105

Mã chương trình trên ESP32:

```
#include <ESP32Servo.h>

#define SERVO1_PIN 14
#define SERVO2_PIN 27
#define SERVO3_PIN 26
#define SERVO4_PIN 18
#define SERVO5_PIN 23
#define BUTTON_PIN 12

Servo servo1, servo2, servo3, servo4, servo5;
int angle1 = 0, angle2 = 0, angle3 = 85, angle4 = 85, angle5 = 0;

void setup() {
  Serial.begin(115200);

  servo1.attach(SERVO1_PIN);
  servo2.attach(SERVO2_PIN);
  servo3.attach(SERVO3_PIN);
  servo4.attach(SERVO4_PIN);
  servo5.attach(SERVO5_PIN);

  pinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
    resetServos(); // Đặt tất cả servo về góc ban đầu
    Serial.println("Nhap lenh: S1 90 S2 45 hoac Reset");
}

void loop() {
    if (digitalRead(BUTTON_PIN) == LOW) {
        resetServos();
        delay(300); // Chống dội phím
    }

    if (Serial.available() > 0) {
        String inputString = Serial.readStringUntil('\n');
        inputString.trim();
        if (inputString.equalsIgnoreCase("Reset")) {
            resetServos();
        } else {
            processServoCommands(inputString);
        }
    }
}

// Hàm xử lý nhiều lệnh servo cùng lúc
void processServoCommands(String command) {
    Serial.print("Lenh nhan: ");
    Serial.println(command);
```

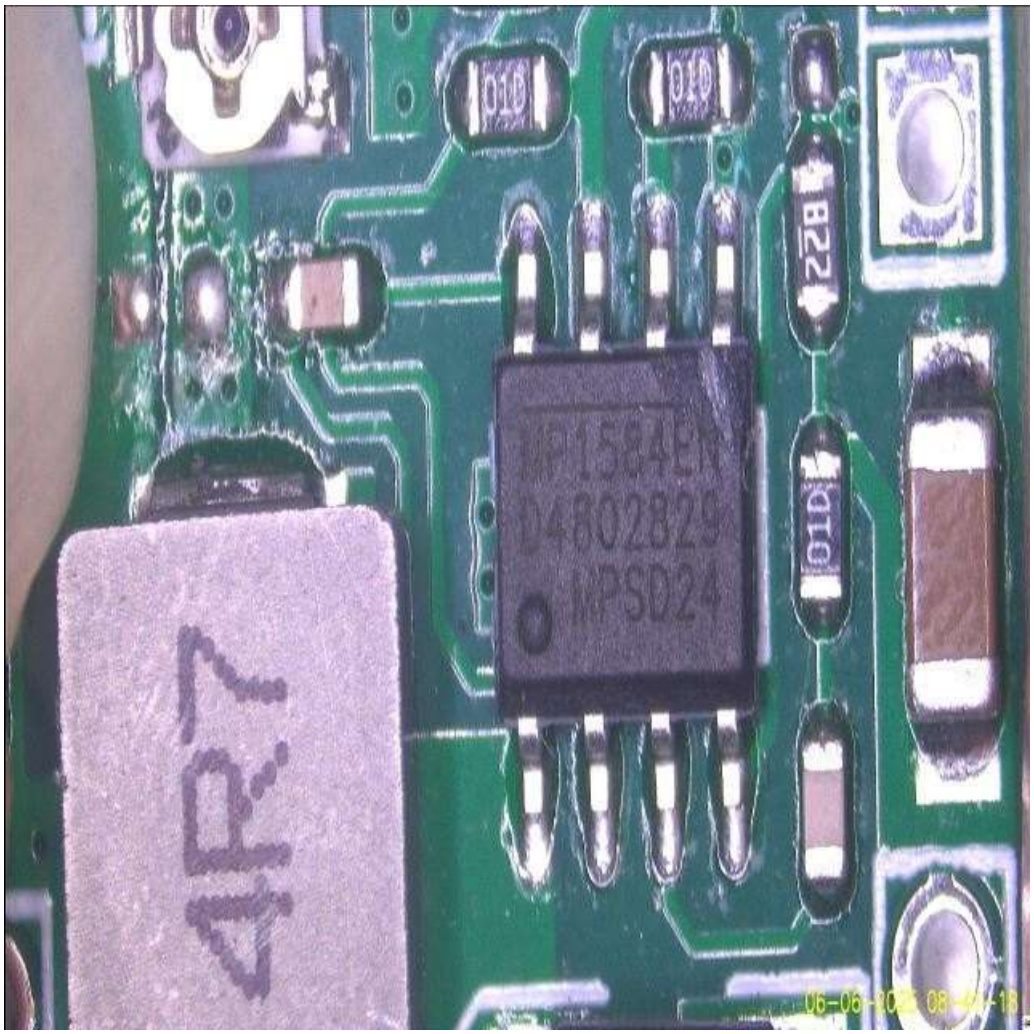
```
  int startIndex = 0;
  while (startIndex < command.length()) {
    int spaceIndex = command.indexOf(' ', startIndex);
    if (spaceIndex == -1) break;  // Không còn khoảng trắng, thoát vòng lặp

    String servoCmd = command.substring(startIndex, spaceIndex);  // Lấy "S1", "S2", ...
    startIndex = spaceIndex + 1;
    spaceIndex = command.indexOf(' ', startIndex);
    if (spaceIndex == -1) spaceIndex = command.length();

    String angleStr = command.substring(startIndex, spaceIndex);  // Lấy góc
    startIndex = spaceIndex + 1;

    int angle = angleStr.toInt();
    if (angle < 0 || angle > 180) continue; // Bỏ qua nếu góc không hợp lệ

    int servoNum = servoCmd.charAt(1) - '0';
    switch (servoNum) {
      case 1:
        angle1 = angle;
        servo1.write(angle1);
        break;
      case 2:
        smoothMove(servo2, angle2, angle); // quay chậm
        angle2 = angle;
```

```
            break;
        case 3:
            smoothMove(servo3, angle3, angle); // quay chậm
            angle3 = angle;
            break;
        case 4:
            smoothMove(servo4, angle4, angle); // quay chậm
            angle4 = angle;
            break;
        case 5:
            angle5 = angle;
            servo5.write(angle5);
            break;
        default:
            continue;
    }

    Serial.print("Servo ");
    Serial.print(servoNum);
    Serial.print(" -> ");
    Serial.print(angle);
    Serial.println(" do");
  }
}
```

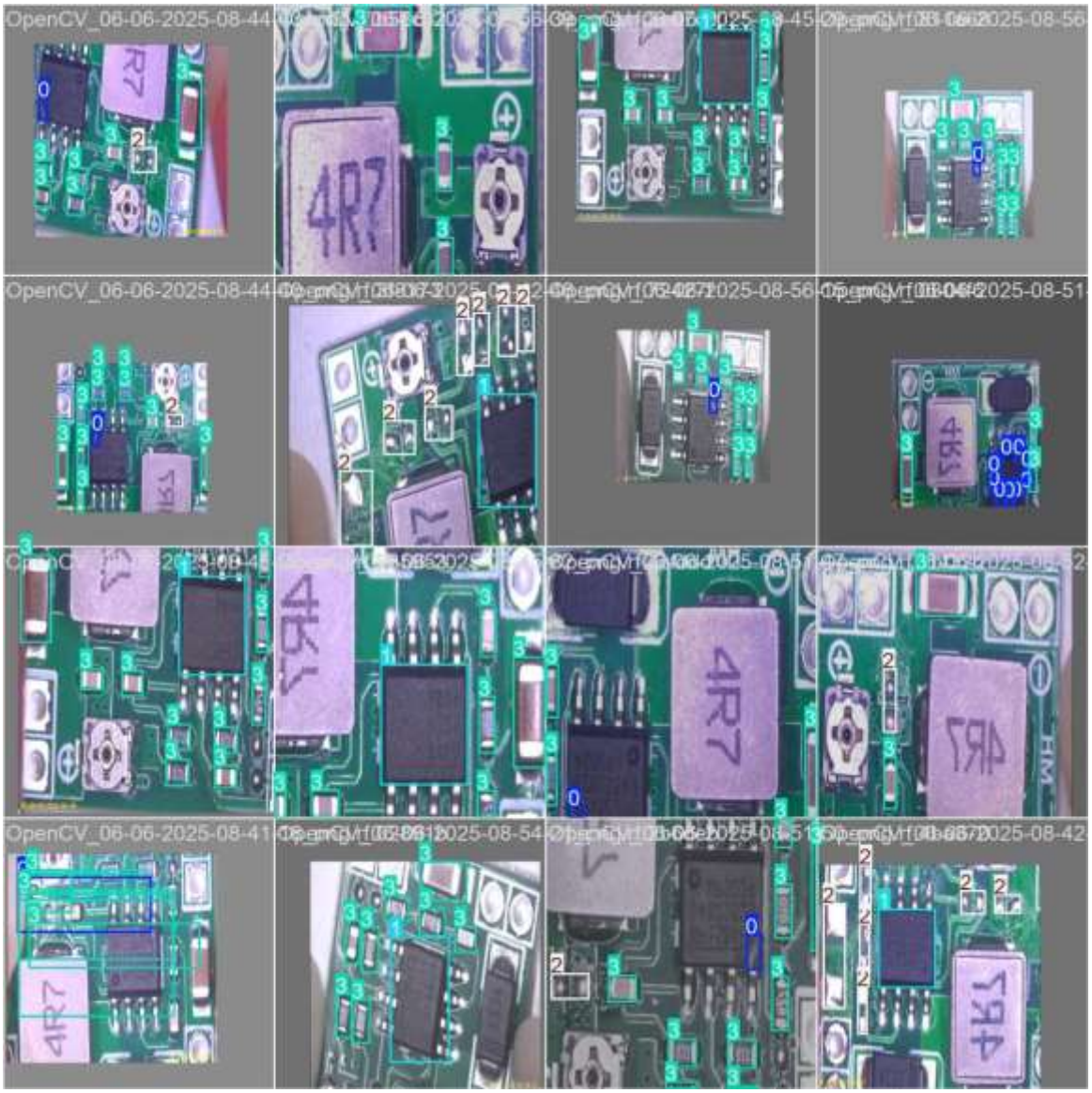
```
// Hàm reset tất cả servo về vị trí ban đầu.
void resetServos() {
    Serial.println("Reset");
    angle1 = angle2 = 0;
    angle3 = 85;
    angle4 = 85;
    angle5 = 0;
    servo1.write(angle1);
    servo2.write(angle2);
    servo3.write(angle3);
    servo4.write(angle4);
    servo5.write(angle5);
}

// Hàm giúp servo quay từ từ từ góc cũ đến góc mới
void smoothMove(Servo &servo, int fromAngle, int toAngle) {
    int step = (fromAngle < toAngle) ? 1 : -1;
    for (int pos = fromAngle; pos != toAngle; pos += step) {
        servo.write(pos);
        delay(4); // Điều chỉnh độ chậm tại đây (10ms mỗi bước)
    }
    servo.write(toAngle); // Đảm bảo dừng chính xác ở góc cuối
}
```

PHỤ LỤC 2

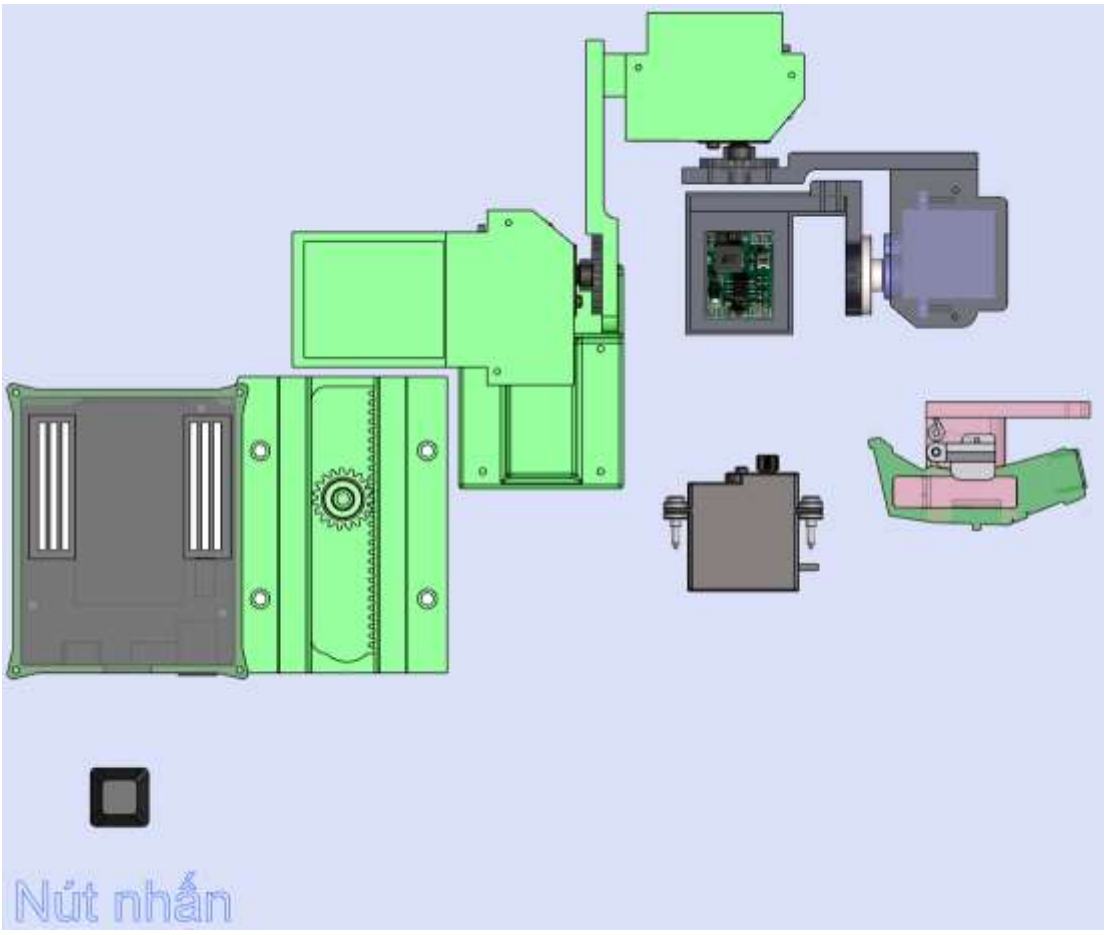
Một số ảnh đầu vào, bounding box, thiết kế phần cứng:



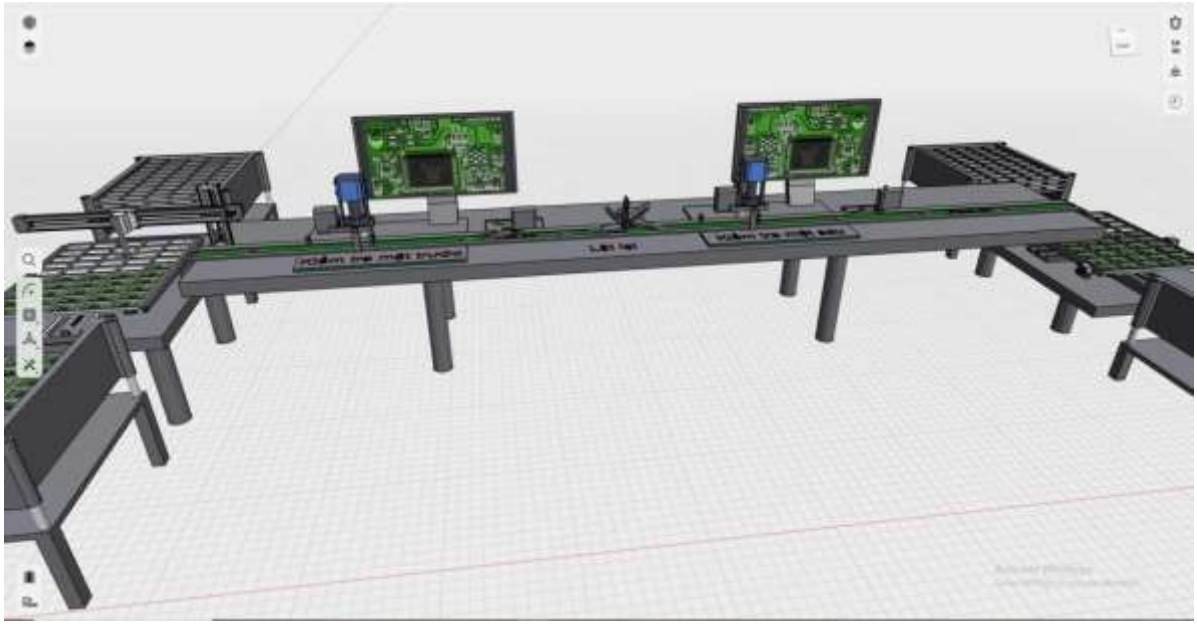
Hình 1. PCBA lỗi mất linh kiện, vỡ IC



Hình 2. Kết quả thu được sau khi bounding box và huấn luyện trên YOLOv8m



Hình 3. Hình chiếu đứng tổng thể bộ chuyển động



Hình 4. Đề xuất nâng cao