

**THE UNIVERSITY OF DANANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF ADVANCED SCIENCE AND TECHNOLOGY**



GRADUATION THESIS

**MAJOR: IN MECHANICAL ENGINEERING –
AUTOMATIC PRODUCTION**

**AI SOLDERING INSPECTION
FOR X-D0240-326 IN ANTENNA POTTING LINE**

Supervisor : Assoc. Prof. Dr. DANG PHUOC VINH
Co-supervisor : Eng. NGUYEN DANG KHANH
Students : NGO VAN TRUNG_123200087
DUONG MAI LY _123200108
Class : 20PFIEV1

DaNang, June 2025

[illegible]

REVIEWER'S COMMENT

This image shows a full page of white paper with horizontal dotted lines, typical of primary-ruled notebook paper. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

GRADUATION PROJECT TASK

No.	Student's name	Students ID	Class	Major
1	Ngo Van Trung	123200087	20PFIEV1	Automated production
2	Duong Mai Ly	123200108	20PFIEV1	Automated production

1. Topic: *AI Soldering Inspection for X-D0240-326 in Antenna Potting line*
2. Topics covered by: ☐ Subject to an IP agreement on project outcomes
3. Initial figures and data: *Informations and images are taken from PREMO.CO.LTD.*
4. Contents of explanations and calculations:

a) General task

No.	Student's name	Contents
1	Ngo Van Trung	Conduct research and study theoretical concepts and documentation related to topics such as mechanical design, sensors, AI models, and communication methods, with a focus on applications in AI-based inspection systems.
2	Duong Mai Ly	

b) Specific task

No.	Student's name	Contents
-----	----------------	----------

1	Ngo Van Trung	- Design of a camera fixture system - Collecting data and Training the AI model - Conduct hardware selection based on technical calculations. - Deploy the model in Python - Device communication and interface display using C#
2	Duong Mai Ly	- PLC programming - Collecting data and Training the AI model - Deploy the model in Python - Device communication and interface display using C# - Conduct hardware selection based on technical calculations.

5. *Drawings, graphs (specify types and sizes of drawings):*

a) *General task*

No.	Student's name	Contents
1	Ngo Van Trung	Drawing 3D on SolidWorks 360 and 2D AutoCAD, Draw.io
2	Duong Mai Ly	

b) *Specific*

No.	Student's name	Contents
1	Ngo Van Trung	- 1 Electrical Schematic Diagram A0 - 1 Fixture Drawings A0
2	Duong Mai Ly	- 1 Controller Diagram Drawings A0 - 1 Flowchart A0

6. *Supervisor : Assoc.Prof.Dr Dang Phuoc Vinh*

7. *Project assignment date: 17/02/2025*

8. *Project completion date: 30/05/2025*

Head of Department

DaNang, June 2025

Supervisor

ABSTRACT

During our internship at PREMO VIETNAM Co., Ltd., we had the opportunity to directly engage with the Antenna Potting production line, where we explored various technologies across different stages of the line, gaining practical knowledge about manufacturing processes and identifying issues within the factory, particularly in weld quality inspection. This experience inspired the development of the thesis titled “AI Soldering Inspection for X-D0240-326 in Antenna Potting line,” aiming to apply artificial intelligence (AI) to automate and optimize defect detection in an industrial production environment.

The project began with the research, selection, and integration of hardware components, including an industrial camera, suitable lenses, custom camera jigs, and a central processing unit, to create a complete inspection station compatible with the existing production line. We utilized the NVIDIA Jetson as the image processing server, connected to the camera to capture data, analyze it using an AI model, and transmit classification results via an internal network, ensuring fast processing, low latency, and high stability in the production environment.

Real-world production images were collected, labeled, and used to train a deep learning model, fine-tuned to detect weld imperfections with high accuracy and reliability under factory conditions. The system was synchronized with PLCs and actuators, enabling real-time inspection and classification of products as pass or fail, automatically sorting them into separate trays before proceeding to the next stage.

Despite time and resource constraints, the project achieved positive outcomes, including reducing one manual inspection worker, shortening inspection time, and improving defect detection consistency. The project demonstrates the feasibility of integrating AI into traditional manufacturing, laying the groundwork for future intelligent quality control systems.

ACKNOWLEDGEMENTS

The five-year journey at the University of Science and Technology – The University of Danang has not only equipped me with specialized knowledge but also shaped my mindset, skills, and professional attitude. Each semester marked a step toward maturity, and throughout this journey, I was fortunate to receive dedicated guidance from my lecturers—those who inspired me and laid a solid foundation for the path I am pursuing.

I am particularly grateful to the FAST Faculty, where I had the opportunity to study in an innovative and specialized environment, gaining early exposure to technology and practical applications. The knowledge acquired here has been a vital asset, enabling me to confidently step into the professional world.

In my final semester, I undertook the thesis project titled: “AI Soldering Inspection for X-D0240-326 in Antenna Potting line”. This was not merely a technical assignment but a valuable real-world experience, where I learned to collaborate in an industrial setting, develop appropriate technical solutions, and, most importantly, take responsibility for every stage of the project.

I extend my heartfelt gratitude to Assoc. Prof. Dr. Dang Phuoc Vinh, my academic supervisor, who provided unwavering guidance and support from the initial ideas to the completion of the thesis. His detailed feedback and timely assistance were crucial in helping me maintain focus, scientific rigor, and a clear direction throughout the process.

I also express my sincere thanks to PREMO VIETNAM Co., Ltd., and particularly to engineers Nguyen Trong Tai and Nguyen Dang Khanh, for creating a supportive environment and offering invaluable guidance during my internship. Their support enabled me to directly engage with the production line, test the system, and validate the effectiveness of my work.

Furthermore, I would like to acknowledge Mr. Le Hoai Nam, my faculty advisor, as well as the lecturers from the Faculty of Mechanical Engineering, Faculty of Mechatronics, and all lecturers at the University of Science and Technology, who have imparted knowledge and inspired me throughout my academic journey.

I am deeply grateful to my family and friends, who have consistently supported, listened, and motivated me to overcome pressures and challenges.

Finally, I sincerely thank the Thesis Defense Committee for taking the time to review and evaluate my work.

Sincerely, thank you!

COMMITMENT

I hereby certify that this graduation thesis is the result of my own research and implementation under the guidance of Assoc. Prof. Dr. Dang Phuoc Vinh and with support from engineers at PREMO Co., Ltd.

All contents, calculations, images, tables, and references are presented honestly, objectively, and are properly cited from reliable sources in accordance with academic regulations. I strictly do not copy any published material without proper citation.

The structure of the report, technical drawings, and overall presentation follow the prescribed format set by the university.

I take full responsibility before the law and the thesis evaluation committee for the accuracy and integrity of the entire content of this thesis. If any violation is found, I will take full responsibility.

Students

Ngo Van Trung

Duong Mai Ly

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	1
1.1. Introduction to PREMO VIETNAM	1
1.1.1 About PREMO	1
1.1.2. About PREMO VIETNAM	1
1.1.3. Organization structure.....	3
1.2. Product portfolio	3
1.3. Antenna Potting line	4
1.4. Stage 4 - Wire Soldering	6
1.5. Problem statement	8
1.6. Topic overview	8
CHAPTER 2: MECHANICAL DESIGN	12
2.1. Problem statement	12
2.2. Fixture design for the camera	13
2.3. Background design	18
2.4. Light shielding frame design	22
CHAPTER 3: CONTROL SYSTEM DESIGN	26
3.1. System overview.....	26
3.2. Equipment selection.....	27
3.2.1. Camera	27
3.2.2. Lens	30
3.2.3. Switch PoE	32
3.2.4. NVIDIA Jetson	34

3.2.5. PLC.....	37
3.3. Connecting devices	39
3.3.1. Fast API.....	40
3.3.2. S7.Net library	41
3.3.3. TCP/IP protocol.....	42
3.4. Training the AI model.....	44
3.4.1. Prepare the data file.....	44
3.4.2. Training model	52
3.4.3. Result.....	53
3.4.4. Continuous model improvement	61
3.5. Source code analysis	62
3.6. System Diagram.....	67
3.6.1. Activity Diagram.....	67
3.6.2. PLC Diagram: Station 5: Camera	70
3.6.3. PLC Diagram: Station 6: Output.....	71
CHAPTER 4: CONCLUSION	73
4.1. Results.....	73
4.2. Problems	76
4.3. Future works	77
REFERENCE	79
LIST OF SYMBOLS AND ABBREVIATION	80
APPENDIX	81

LIST OF FIGURE

Figure 1. 1 PREMO VIETNAM (F1)	1
Figure 1. 2 The grand opening ceremony of PREMO's second factory in VIETNAM (F2)	2
Figure 1. 3 Organizational Chart of PREMO VIETNAM factory	3
Figure 1. 4 Some of PREMO's main products	4
Figure 1. 5 Operational Diagram of the Antenna Potting line	5
Figure 1. 6 Output product of each step	6
Figure 1. 7 Diagram of soldering machine stations	7
Figure 1. 8 Overview of the Wire Soldering machine	9
Figure 1. 9 Product before soldering	10
Figure 1. 10 Pin legs before soldering	10
Figure 1. 11 OK product	11
Figure 1. 12 NG product	11
Figure 2. 1 Rotary table holding two products at the same time	12
Figure 2. 2 3D model of the camera mount with the camera attached	13
Figure 2. 3 3D model of the Camera jig	14
Figure 2. 4 Dimensions of the Camera jig	15
Figure 2. 5 Dimensions of the Camera jig 2	15
Figure 2. 6 Dimensions of the Camera jig 3	16
Figure 2. 7 Dimensions of the Camera jig 4	17
Figure 2. 8 Dimensions of the Camera jig 5	17
Figure 2. 9 3D model of the Background	19
Figure 2. 10 Dimension of Background jig 1	19
Figure 2. 11 Dimensions of the Background jig 2	20
Figure 2. 12 Dimensions of the Background jig 3	20
Figure 2. 13 Dimensions of the Background jig 4	21
Figure 2. 14 Dimensions of the Cylinder	21
Figure 2. 15 3D model of the Light shielding frame	22
Figure 2. 16 Dimensions of the Cover jig 1	23

Figure 2. 17 Dimensions of the Cover jig 1	23
Figure 2. 18 Dimensions of the Cover.....	24
Figure 2. 19 The visual appearance of the system after all components have been assembled onto the machine.	25
Figure 3. 1 System overview	26
Figure 3. 2 The HT-GE502C-T1P-C camera	30
Figure 3. 3 Specifications of M2514-MP2	32
Figure 3. 4 Dimensions of the lens M2514-MP2	32
Figure 3. 5 PoE switch supplies power and transmits data at the same time	32
Figure 3. 6 Switch PoE TL-SG1008MP.....	34
Figure 3. 7 Jetson Orin NX.....	34
Figure 3. 8 System & Mounting Dimensions.....	36
Figure 3. 9 I/O Mechanical Layout	36
Figure 3. 10 PLC 1214C DC/DC/DC.....	38
Figure 3. 11 Device connection diagram.....	39
Figure 3. 12 TCP/IP model.....	43
Figure 3. 13 PLC diagram: Collecting data.....	45
Figure 3. 14 CSharp diagram: Collecting data	47
Figure 3. 15 Example of the predefined_classes.txt file	49
Figure 3. 16 Interface of the Labellmg software.....	49
Figure 3. 17 Label an image	50
Figure 3. 18 Example of the data.yaml file	51
Figure 3. 19 Code to set up the working program.....	52
Figure 3. 20 AI model training code	53
Figure 3. 21 Result charts of Model 1	53
Figure 3. 22 Training progress plots of Model 1	54
Figure 3. 23 Confusion Matrix of Model 1	56
Figure 3. 24 Result charts of Model 1	58
Figure 3. 25 Training progress plots of Model 2.....	59
Figure 3. 26 Confusion Matrix of Model 2	60

Figure 3. 27 C# Diagram.....	67
Figure 3. 28 Display interface of the program	68
Figure 3. 29 Python Diagram	69
Figure 3. 30 PLC Diagram: Station 5: Camera	70
Figure 3. 31 PLC Diagram: Station 6: Output	71
 Figure 4. 1 Interface after displaying the program results	 75
Figure 4. 2 Actual processing time when applied to the system.....	75

LIST OF TABLE

Table 3. 1 Comparison between GigE Camera and USB 3.0 Camera	29
Table 3. 2 Technical specifications of the HT-GE502C-T1P-C camera.....	30
Table 3. 3 Parameters affecting lens selection	31
Table 3. 4 Comparison of PoE switches available at PREMO company.....	33
Table 3. 5 Technical specifications	37
Table 3. 6 Some characteristic parameters of PLC 1214C DC/DC/DC.....	38
Table 3. 7 The processing time when testing the model 1	57
Table 3. 8 The processing time when testing the model 2	61
Table 3. 9 Libraries used in the program.....	65
Table 4. 1 System performance evaluation	73

CHAPTER 1: INTRODUCTION

1.1. Introduction to PREMO VIETNAM

1.1.1 About PREMO

PREMO is a Spain-based company and was founded in 1962 at Málaga, Spain. It engaged in the development, manufacturing and sales of electronic components with special focus on the key enabling technologies of the 4th Industrial Revolution: Connected and Electric Vehicles, Solar and Industrial Automation.

Over 1300 employees with 8 design centers, 4 production locations and an extensive sales network let PREMO have a global presence in more than 36 countries to meet our customers specific needs.

1.1.2. About PREMO VIETNAM

PREMO VIETNAM was established on October 29, 2015 and officially commenced operations in 2016. The headquarters and manufacturing plant of PREMO VIETNAM are located at Lot 21, Dien Nam - Dien Ngoc Industrial Park, Dien Ban Town, Quang Nam Province. This is the third electronic component manufacturing plant of the PREMO GROUP, following plants in Morocco and China.



Figure 1. 1 PREMO VIETNAM (F1)

On June 10, 2022, the grand opening ceremony of Premo's second factory in Vietnam was solemnly held with the presence of the Company's Board of Directors, distinguished delegates, guests, and Premo Vietnam's staff. The second factory is the culmination of the dedication, commitment, and passion for work of Premo Vietnam's team.



Figure 1. 2 The grand opening ceremony of PREMO's second factory in VIETNAM (F2)

1.1.3. Organization structure

The organizational structure of PREMO VIETNAM factory is designed based on a functional management model, ensuring close coordination of production, technical administration, and business development under unified leadership.



Figure 1. 3 Organizational Chart of PREMO VIETNAM factory

1.2. Product portfolio

Product portfolio includes RFID antennas (worldwide leader), AR/VR Motion Tracking Sensors, power transformers, inductors & chokes, current sensors, EMC filters, and PLC components. In addition to our broad range of standard components, off-the-shelf products, PREMO also designs custom solutions to fit customer requirements, based on the latest technologies to help your systems be more efficient.



Figure 1. 4 Some of PREMO's main products

Factory Overview:

Currently, the product lines in production at factory 1 include X-D0240-326, 3DC11, X-D0240-085R2-S, X-D0287-078,...

Consists of 9 lines:

- 3DC11
- A01
- 3DC15
- HPM1 + HPM3
- Midrange
- HPM2 + Midrange
- Antenna Potting
- Trafo Auto
- Sample

1.3. Antenna Potting line

For our project, we will carry it out on the Antenna Potting line. Antenna Potting Line is a semi-automatic line combined with machine and manual labor. The main product of the line is a signal receiving antenna used in the PKE (Passive Keyless Entry

system), with the material code X-D0240-326. At PREMO company, this product is abbreviated as “326”.

In this line, most of the processes are automated, with manual labor only involved in the visual inspection after wire soldering and the visual inspection after potting.

Operational Diagram of the Antenna Potting Line

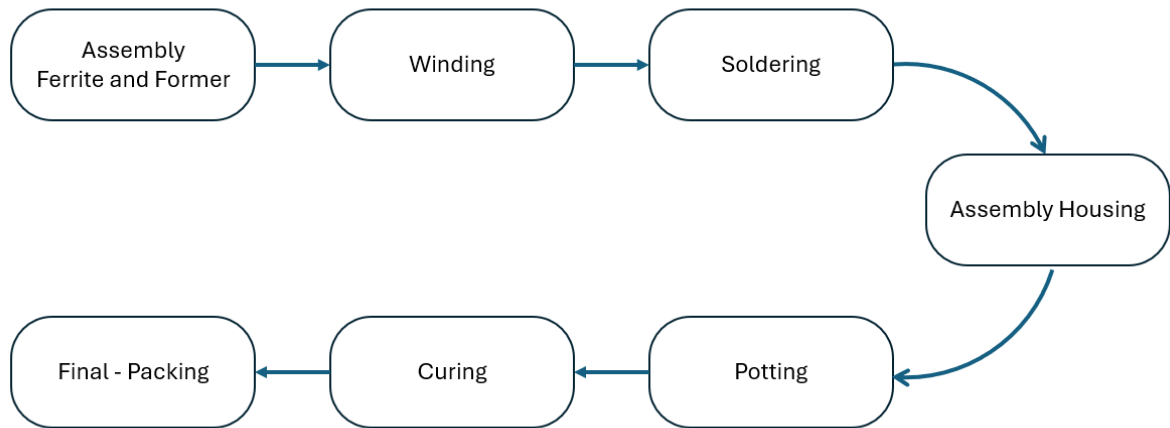


Figure 1. 5 Operational Diagram of the Antenna Potting line

The main product of the line is X-D0240-326, but the Antenna Potting Line doesn't carry out all the stages to produce the product.

The stages to produce the complete X-D0240-326 product:

The production process of the S-D0240-326 product is built as a closed and logical sequence of stages to ensure production efficiency and product quality. From the preparation of components, assembly, technical processing to quality inspection and final packaging, each step in the process strictly adheres to technical standards and quality control measures. This process helps ensure high reliability of the final product, meeting the demands of both customers and the market.

Production process of product X-D0240-326:

- Stage 1: Injection Coil Former and Housing
- Stage 2: Assembly Ferrite with Coil Former
- Stage 3: Winding
- Stage 4: Soldering
- Stage 5: Assembly Housing
- Stage 6: Potting
- Stage 7: Curing
- Stage 8: Final and Packaging

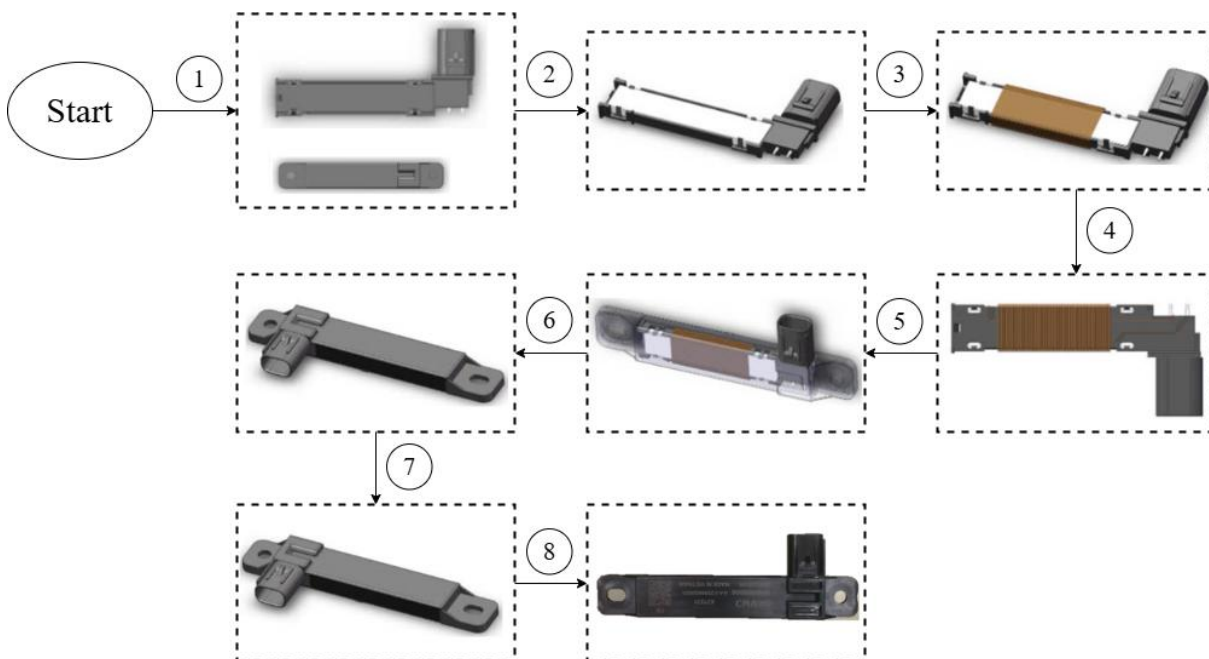


Figure 1. 6 Output product of each step

1.4. Stage 4 - Wire Soldering

Wire soldering is a crucial technique in electronic component manufacturing. It ensures reliable electrical connections, directly impacting the overall performance and functionality of electronic devices. Currently, this machine has two inspection stations: the continuity test and the visual solder point. However, only the continuity test is automated, while the visual solder point is still inspected manually. Therefore, this will be the phase where we focus on implementing the project to automate solder joint inspection instead of manual checking.

Soldering Process

1. Dipping the flux
2. Automatic cleaning tin bath
3. Automatic down of the tool
4. Preheat
5. Automatic soldering between 1s to 2s
6. Automatic set down of the tool
7. Continuity test
8. Visual solder point
9. Automatic put the failure into scrap box
10. Automatic put the OK parts into the output conveyor

The robo glider then picks up the product and places it in the cavity position on the tray of the cargo machine. The soldering process is performed automatically according to the process: Flux → Preheat → Soldering → Continuity → Visual → Sorting.

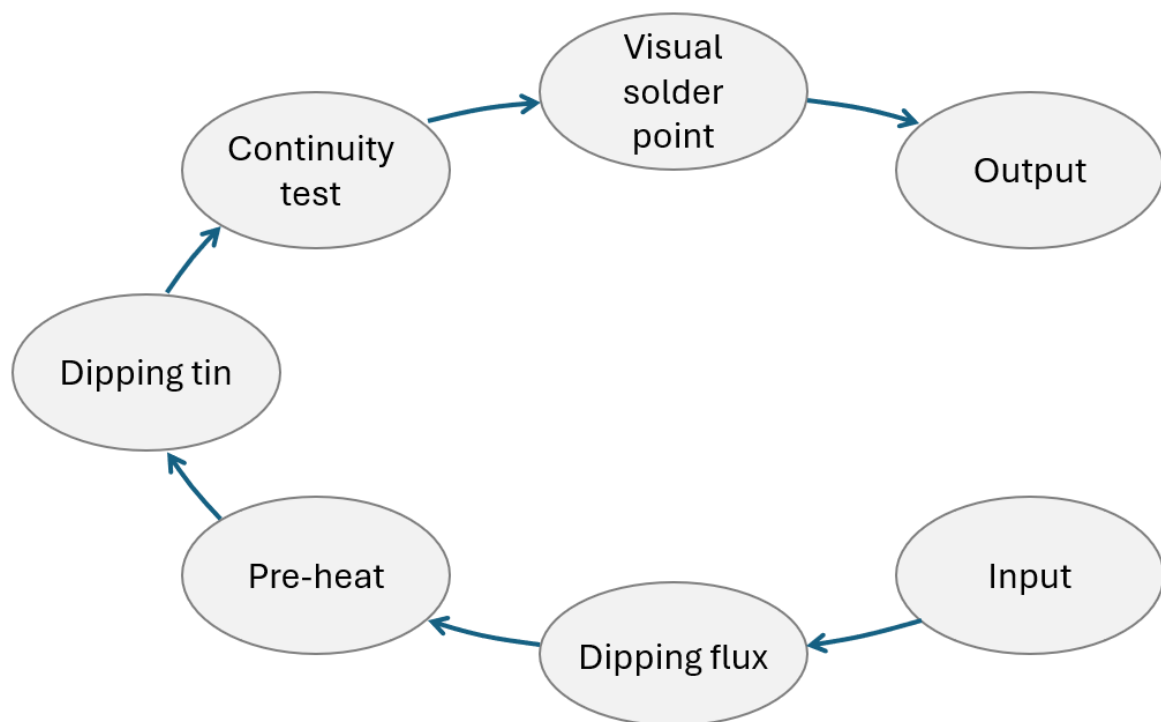


Figure 1. 7 Diagram of soldering machine stations

1.5. Problem statement

The quality inspection process after wire soldering plays a crucial role in ensuring that products meet standards before moving to the next stage. Currently, this process is still performed manually by inspection workers. This leads to certain limitations, such as human errors, inconsistent performance, and high labor costs.

To optimize the production process, reduce reliance on manual labor, and enhance inspection efficiency, our team is implementing the AI Soldering Inspection project - an automated visual inspection system for post-soldering products using artificial intelligence. This system replaces manual inspection workers with an AI model capable of detecting defects quickly, accurately, and continuously.

In this project, my team is responsible for designing, selecting, and implementing the hardware system to ensure stable operation and compatibility with the actual production line. After deploying the hardware system, we begin collecting samples and training the AI model for defect detection. Additionally, we are also in charge of programming the PLC (Programmable Logic Controller), a crucial controller that automates the system's operation process.

1.6. Topic overview

Objectives of the study

With this topic, we will focus on the following issues:

- Design, selection, and implementation of hardware
- Connecting devices within the system
- Training the AI model
- System deployment

General requirements

The improved system is a fully automated system used in the electronic component manufacturing process.

Technical requirements

For the current soldering machine, in each operating cycle, the machine performs the soldering process simultaneously on two products placed in parallel on a specialized

fixture. After implementing AI Soldering Inspection, the system must ensure that the cycle time for each product remains at 5 seconds.

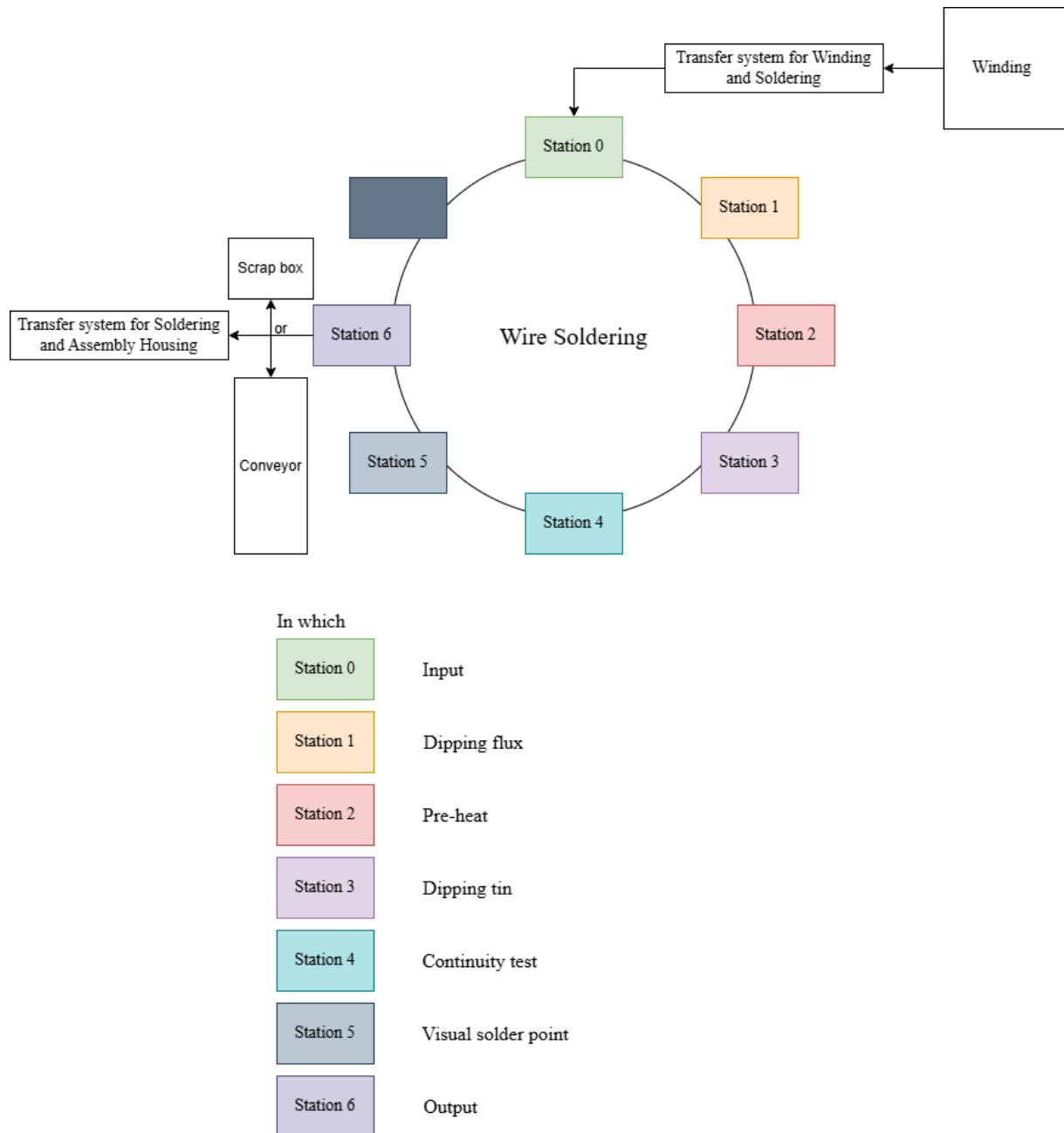


Figure 1. 8 Overview of the Wire Soldering machine

The image above provides a general overview of the Wire Soldering machine: The input of the Wire Soldering machine is the output from the Winding machine. Products that have been wound with wire by the Winding machine are picked up and placed onto the conveyor by a robotic arm. At Station 0 (the input station of the Wire

Soldering machine), another robotic arm picks up the products from the conveyor and transfers them into the machine for soldering processes at the subsequent stations.

The product before soldering has two pin legs, with each pin wrapped with 3 to 4 wire loops. A product meets the requirements if both pin legs are soldered with at least one wire loop.



Figure 1. 9 Product before soldering



Figure 1. 10 Pin legs before soldering

AI soldering inspection for X-D0240-326 in Antenna Potting line

After the soldering process is completed, both pin legs are coated with a layer of solder. This becomes the input for Station 5 – Visual Solder Point (visual inspection of the solder joints). However, in some cases, the solder may not adhere evenly to the pin legs, leading to potential soldering defects. Therefore, visual inspection is essential to ensure quality. A product is considered acceptable only if both pin legs are properly soldered and each leg has at least one wire loop covered with solder.



Figure 1.11 OK product

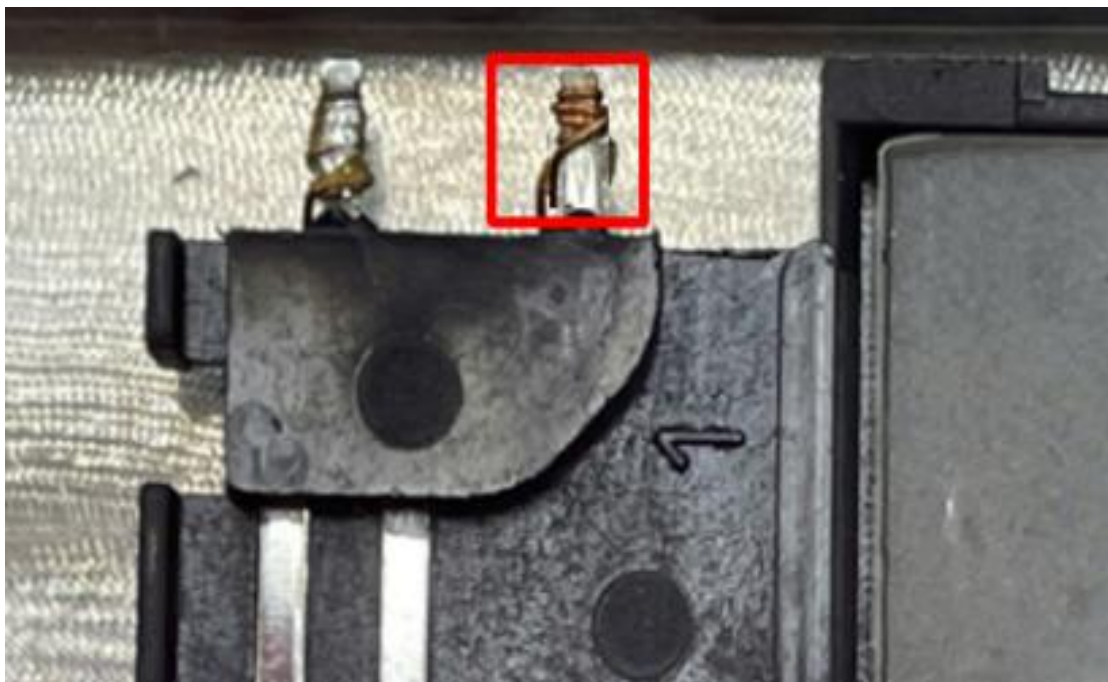


Figure 1.12 NG product

CHAPTER 2: MECHANICAL DESIGN

2.1. Problem statement

The current soldering system performs parallel soldering of two products simultaneously, posing a significant challenge in installing cameras for solder quality inspection. To ensure accuracy, the cameras need to capture images from both sides of each product. However, due to space constraints and the arrangement of equipment, positioning the cameras to fully observe both sides of both products is highly difficult.

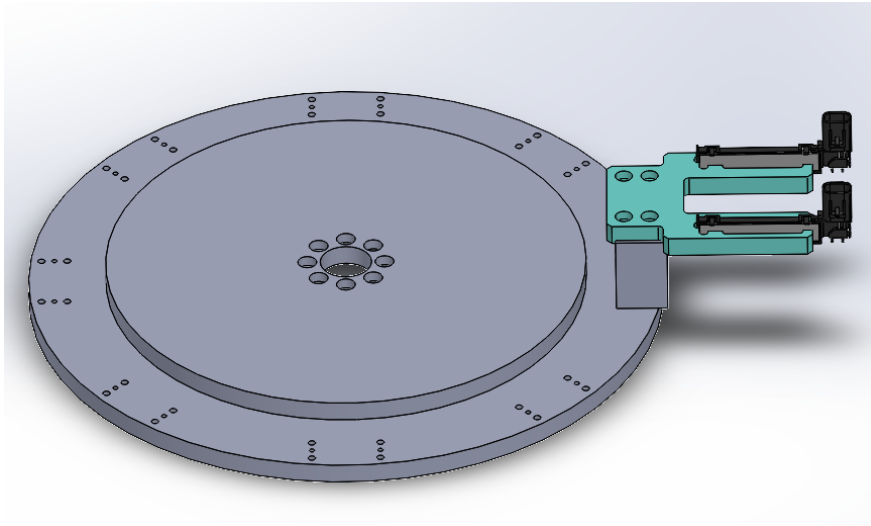


Figure 2. 1 Rotary table holding two products at the same time

We are considering installing only two cameras to capture one side of each product. However, this approach may lead to errors if one side of the product meets the requirements while the other does not. The quality standard requires that each pin must have at least one loop of wire soldered. Therefore, we have increased the quality requirement so that each pin must have at least two loops of wire soldered. This adjustment ensures that both sides meet the original requirement.

To ensure optimal performance of the camera installation system, the design and fabrication of suitable fixtures are crucial. The main components that need to be developed include:

- **Camera fixture:** This fixture is responsible for securely holding and positioning the camera with precision, ensuring the correct angle and distance to capture high-quality images.

- **Background:** The use of a dedicated background helps clearly separate the two products and eliminates image noise caused by background elements.
- **Light shielding frame:** To maintain stable lighting throughout the image capture process, a light shielding frame is necessary. This frame helps control the light source, preventing glare, shadows, or sudden changes in light intensity, thereby significantly improving image quality.

The synchronized integration of these fixtures will contribute to enhancing the accuracy and efficiency of the camera installation system.

2.2. Fixture design for the camera

Design objectives:

- Fix the camera
- Simple structure
- Quick disassembly and easy adjustment

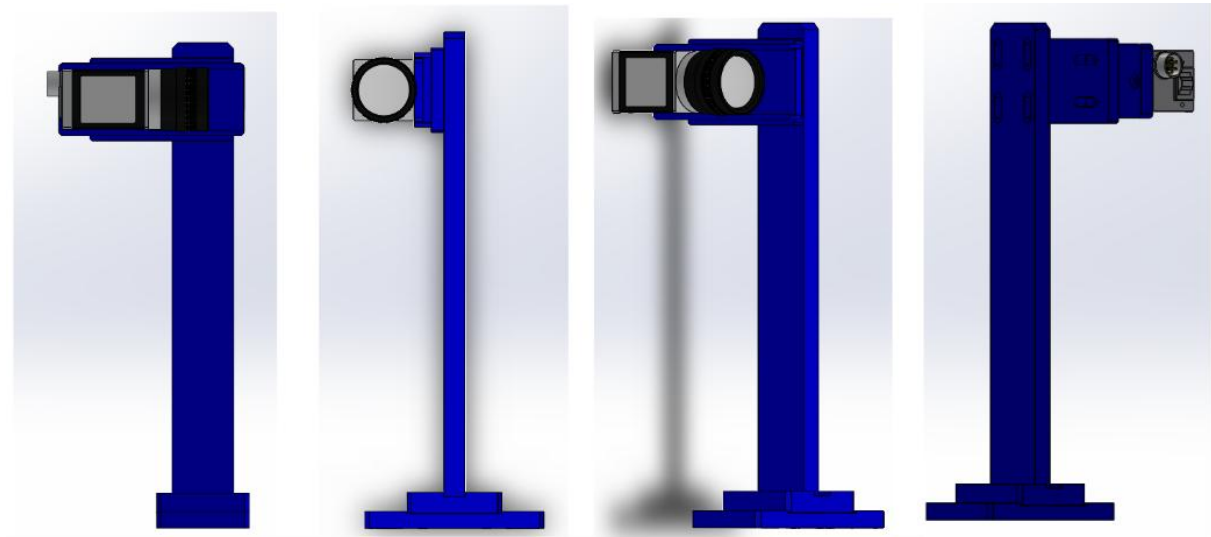


Figure 2. 2 3D model of the camera mount with the camera attached

To position the camera appropriately, we designed a mounting fixture assembled from five aluminum bars. This configuration allows for easy adjustment of both the height and the distance to the product, providing maximum flexibility in camera placement.

Part Drawing of PART FIXTURE

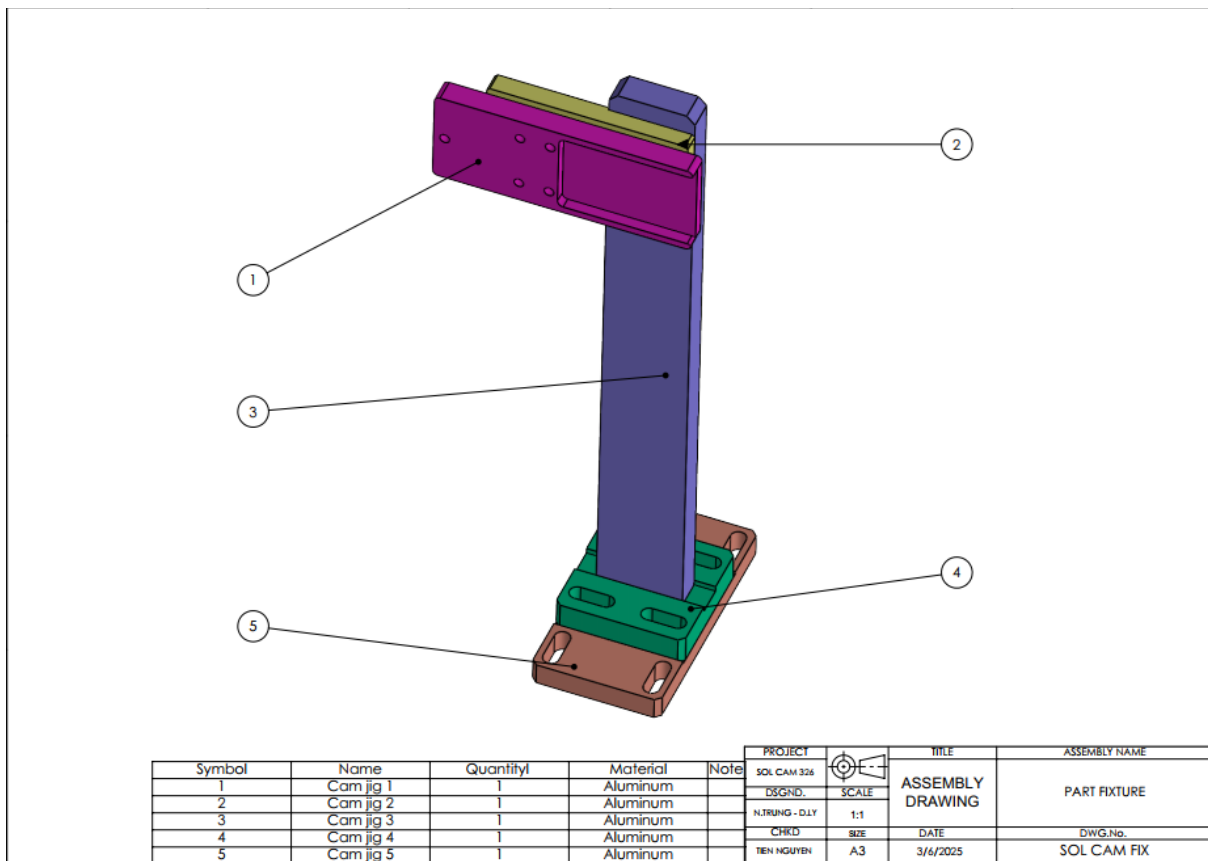


Figure 2. 3 3D model of the Camera jig

Dimensions of the Camera jig 1

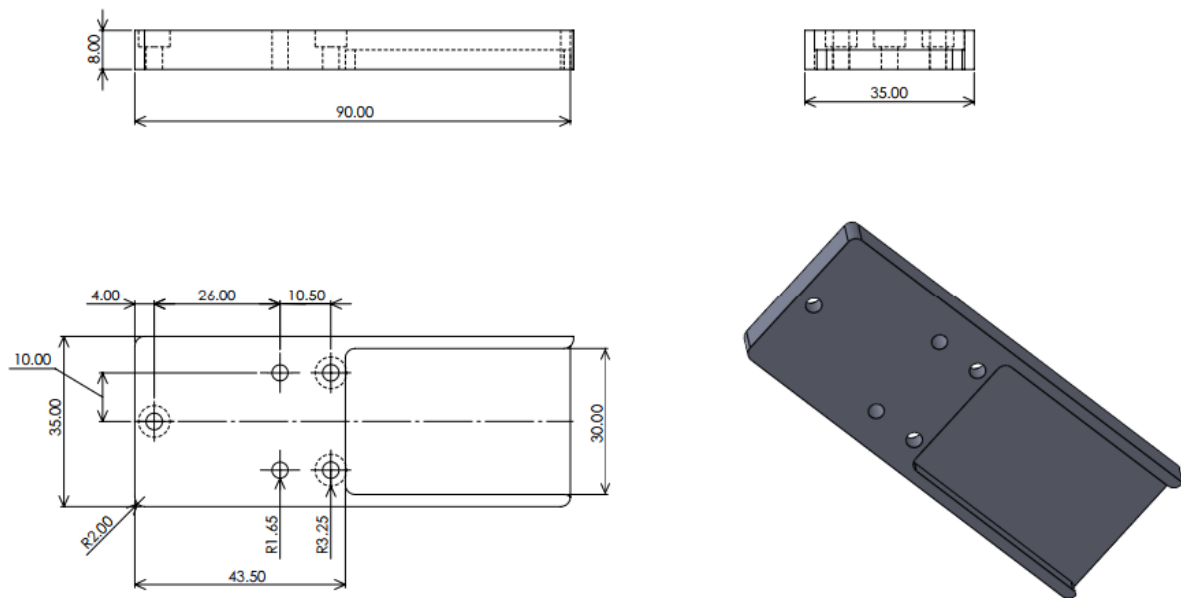


Figure 2. 4 Dimensions of the Camera jig

Dimensions of the Camera jig 2

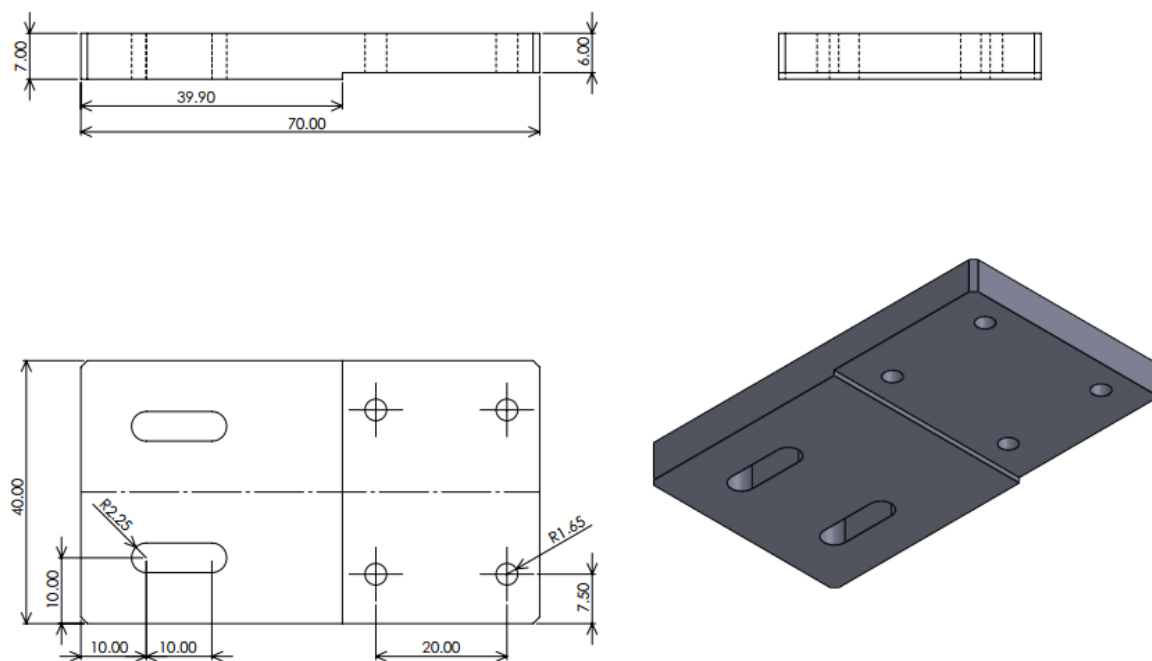


Figure 2. 5 Dimensions of the Camera jig 2

Dimensions of the Camera jig 3

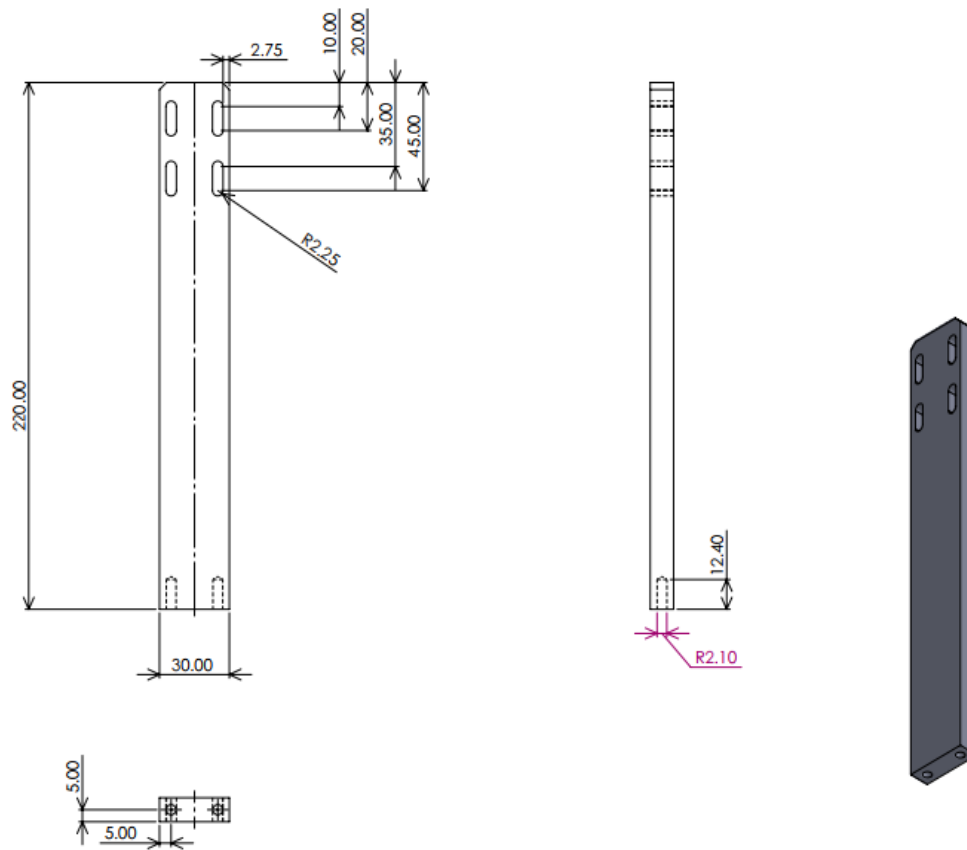


Figure 2. 6 Dimensions of the Camera jig 3

Dimensions of the Camera jig 4

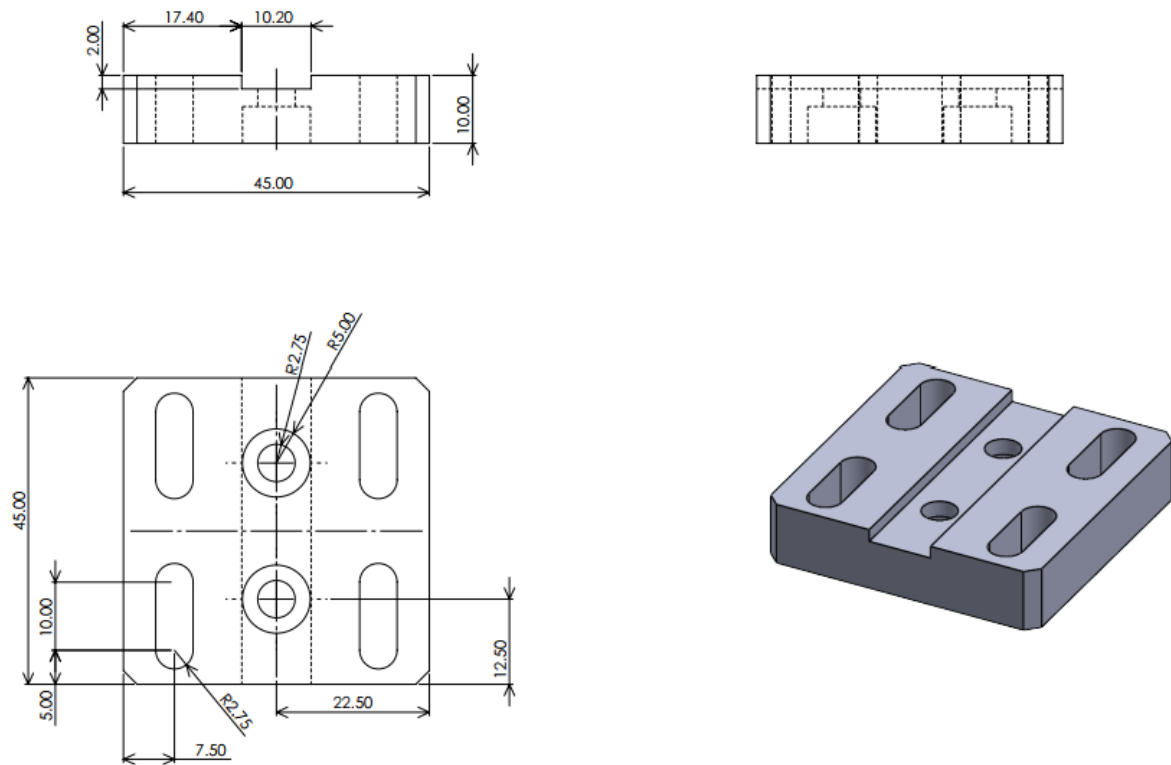


Figure 2. 7 Dimensions of the Camera jig 4

Dimensions of the Camera jig 5

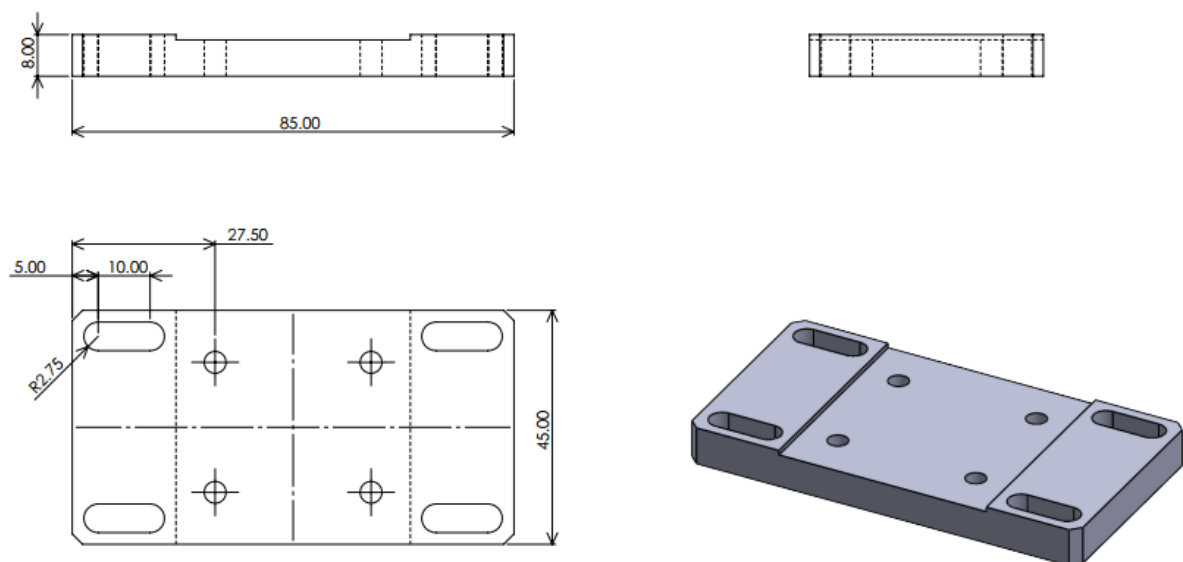


Figure 2. 8 Dimensions of the Camera jig 5

2.3. Background design

Design objectives:

- Background can be raised and lowered
- Simple structure and easy to install

After conducting trials by capturing images of the product from two sides and at two different angles, we observed that the images were affected by external environmental factors, which made it difficult to focus on the weld joints. Therefore, we devised a solution by introducing a movable black background positioned between the two products, actuated by a cylinder mechanism. This approach significantly improved the output image quality, making the weld joints much more distinguishable against the black background.

We selected the MGPM12-40Z cylinder as the main actuator for this section because it is a compact, dual-guide cylinder well-suited for installation in space-constrained systems. With a piston diameter of 12 mm and a stroke of 40 mm, it meets the requirements for short-stroke applications and moderate loads. Additionally, the MGPM12-40Z features integrated guide rods and slide bearings, providing stable, precise linear motion while minimizing misalignment.

Since the background structure consists of a small, black aluminum plate, only a lightweight and compact actuator is needed. Therefore, the MGPM12-40Z is an optimal choice in terms of both technical performance and spatial integration.

Part Drawing of PART CYLINDER

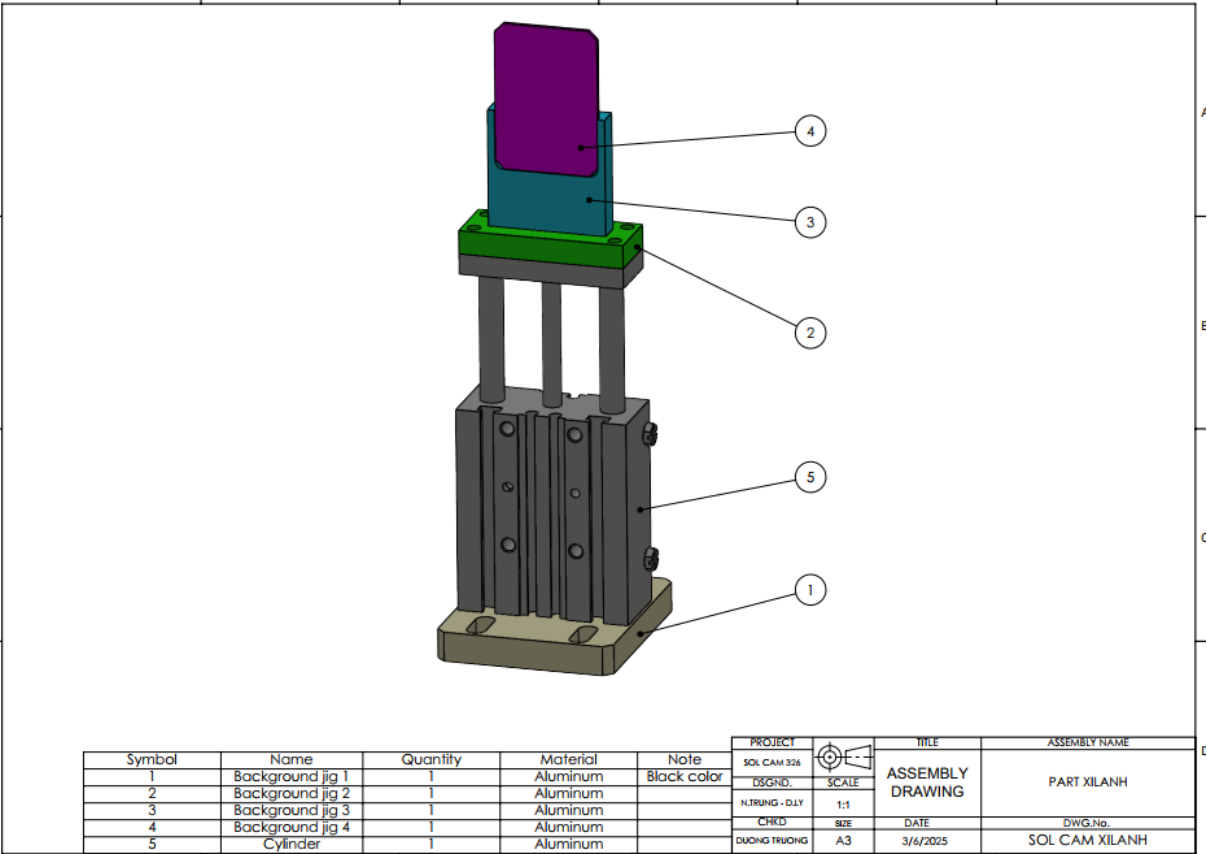


Figure 2. 9 3D model of the Background

Dimensions of the Background jig 1

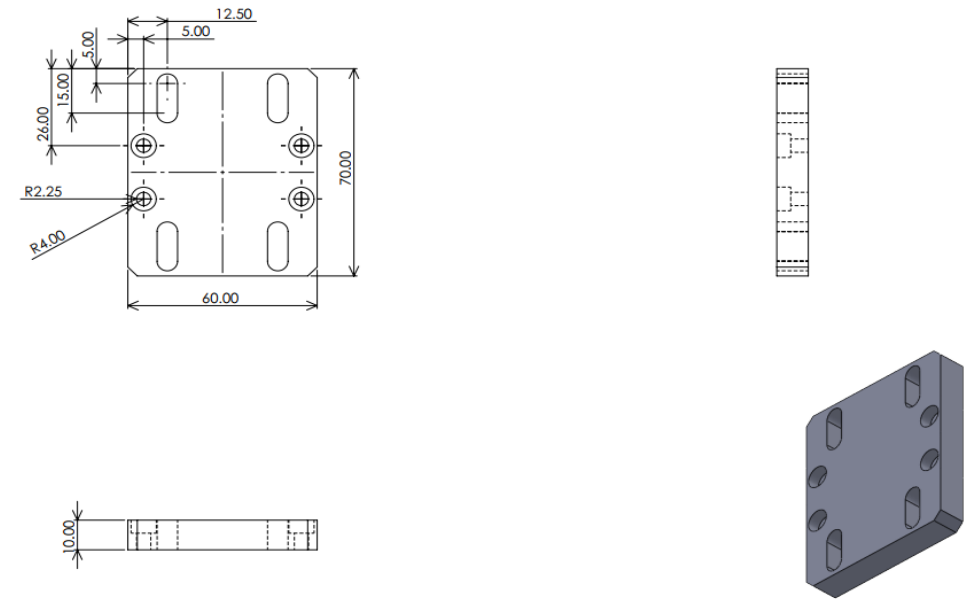


Figure 2. 10 Dimension of Background jig 1

Dimensions of the Background jig 2

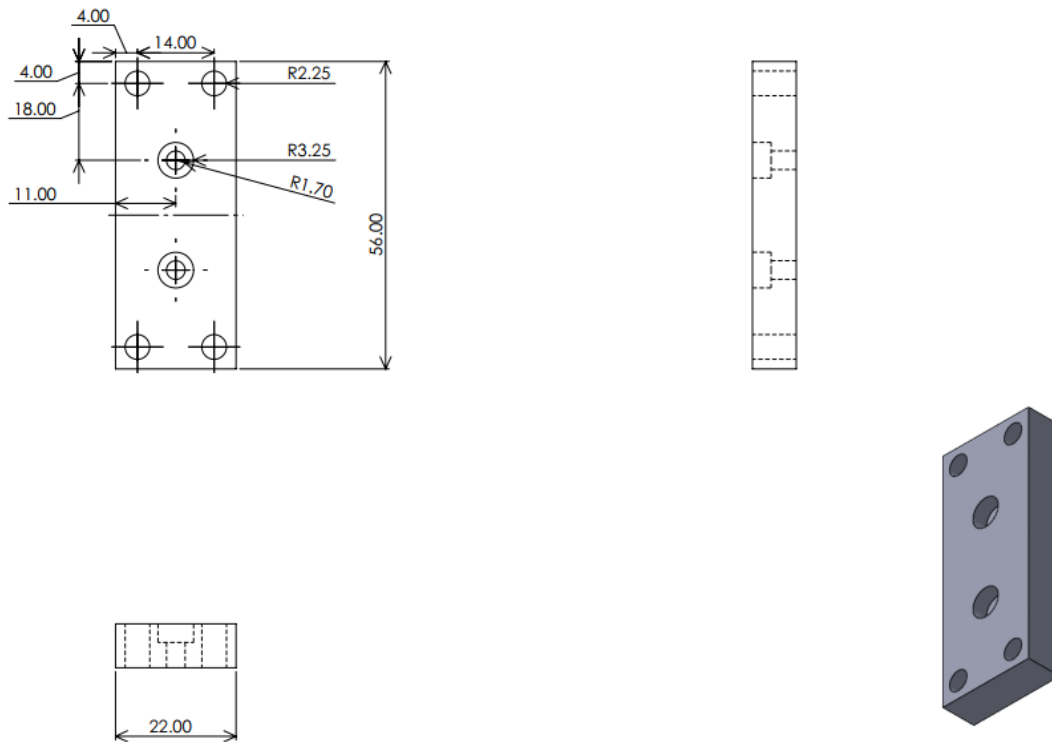


Figure 2. 11 Dimensions of the Background jig 2

Dimensions of the Background jig 3

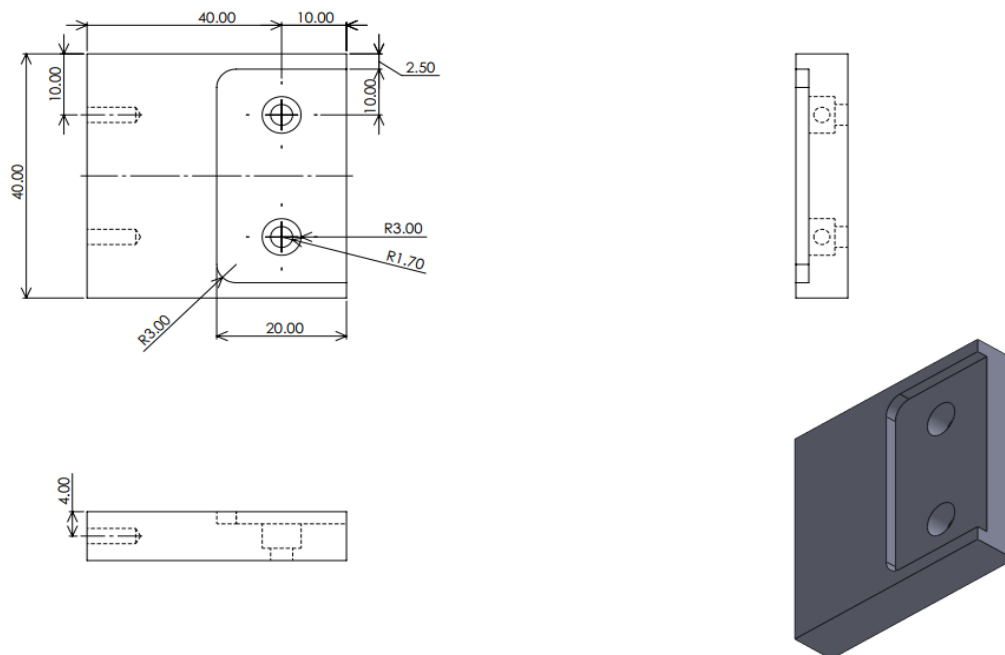


Figure 2. 12 Dimensions of the Background jig 3

Dimensions of the Background jig 4

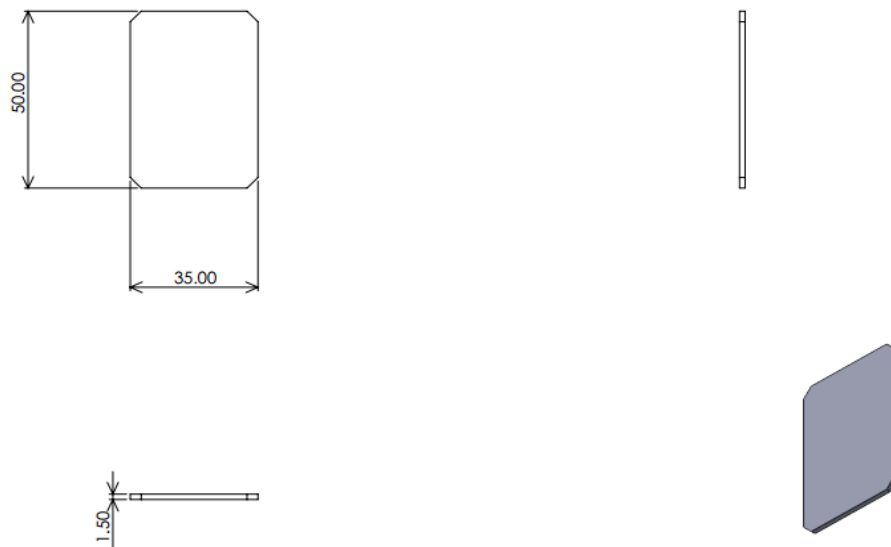


Figure 2. 13 Dimensions of the Background jig 4

Dimensions of the Cylinder

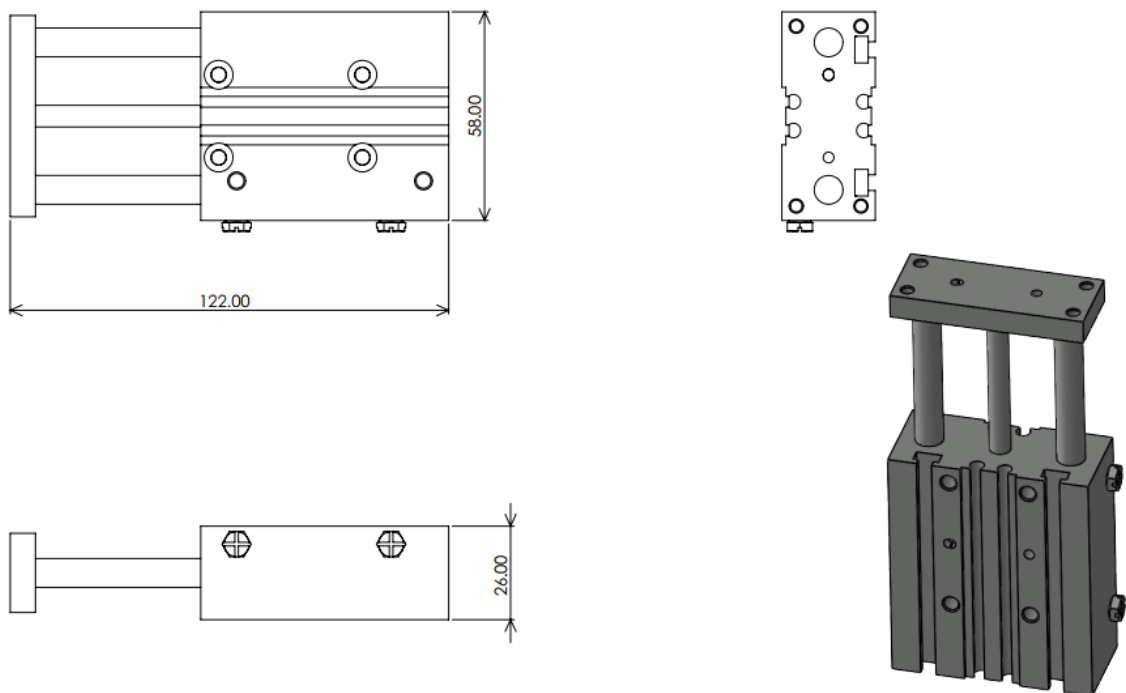


Figure 2. 14 Dimensions of the Cylinder

2.4. Light shielding frame design

Design objectives:

- Simple structure and easy to install
- Block external light, ensuring stable lighting inside the frame.

To ensure the consistency of the output images, we developed a structure similar to a canopy by assembling aluminum blocks. This structure helps reduce the reflection of factory lighting on the weld joints, thereby minimizing negative impacts on image quality during the capturing process.

Part Drawing of PART COVER

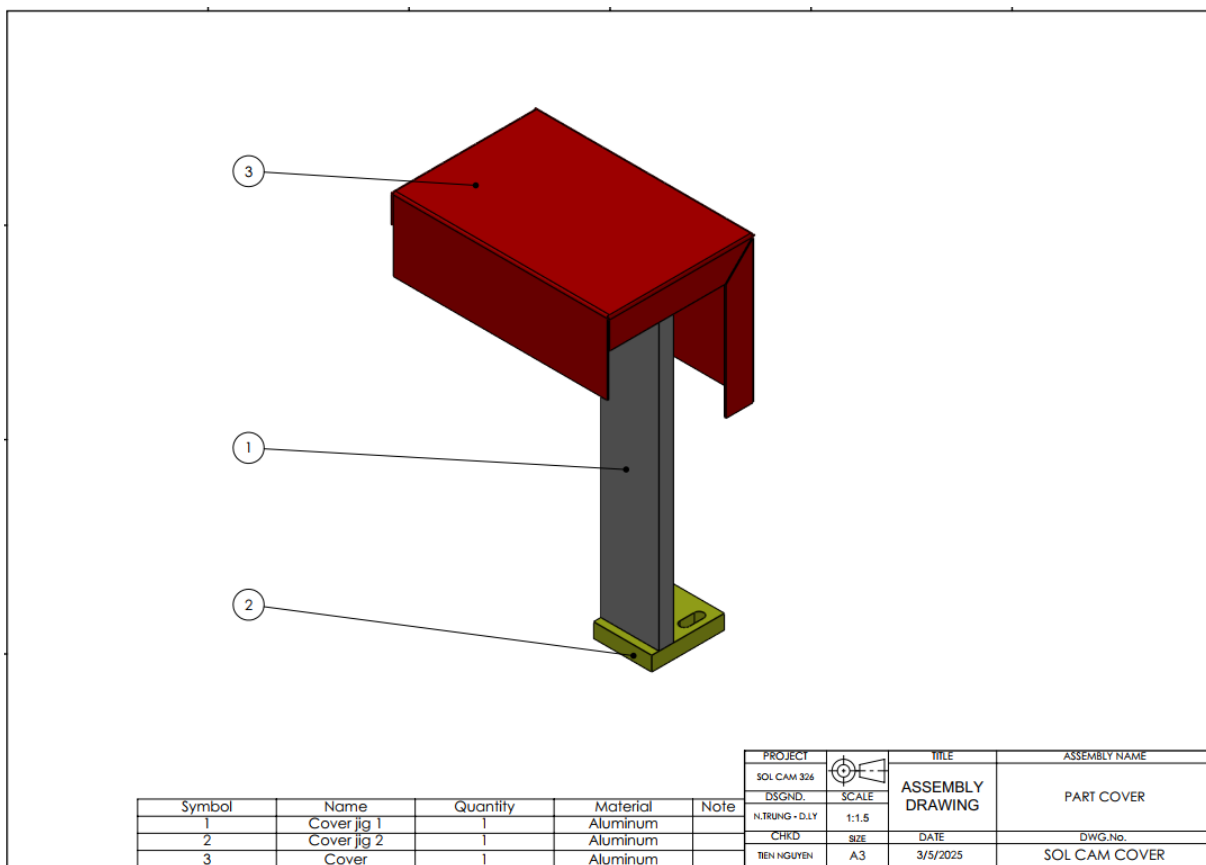


Figure 2. 15 3D model of the Light shielding frame

Dimensions of the Cover jig 1

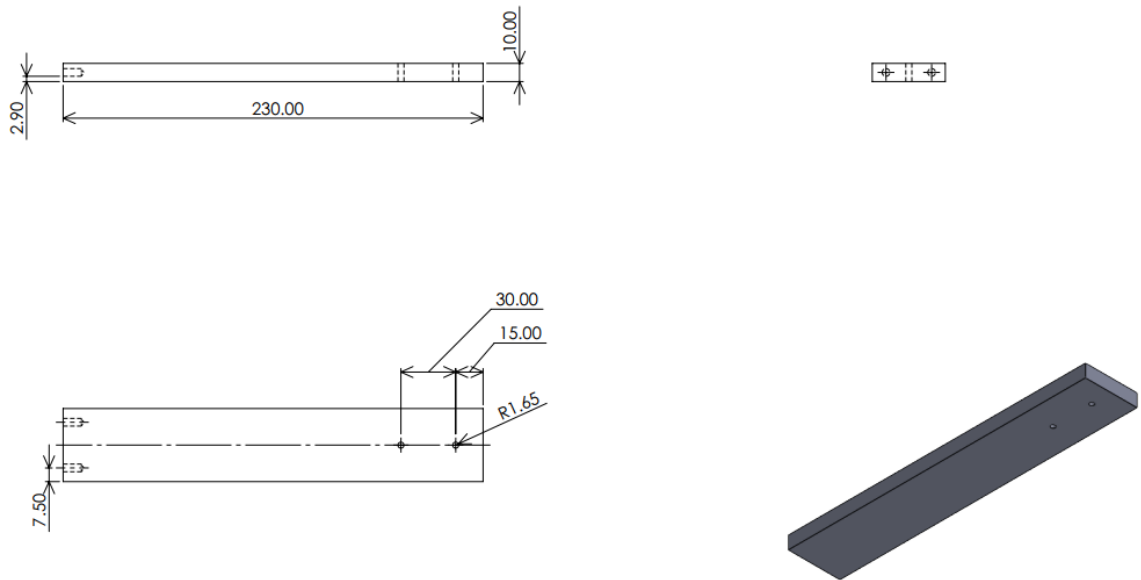


Figure 2. 16 Dimensions of the Cover jig 1

Dimensions of the Cover jig 2

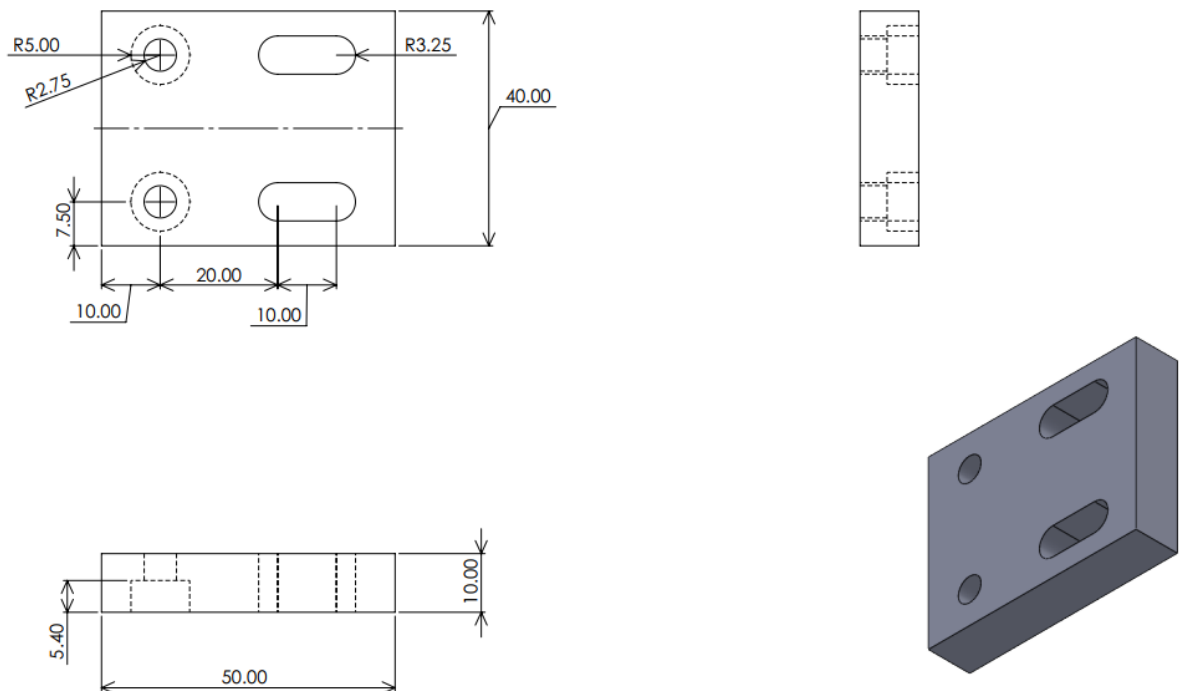


Figure 2. 17 Dimensions of the Cover jig 1

Dimensions of the Cover

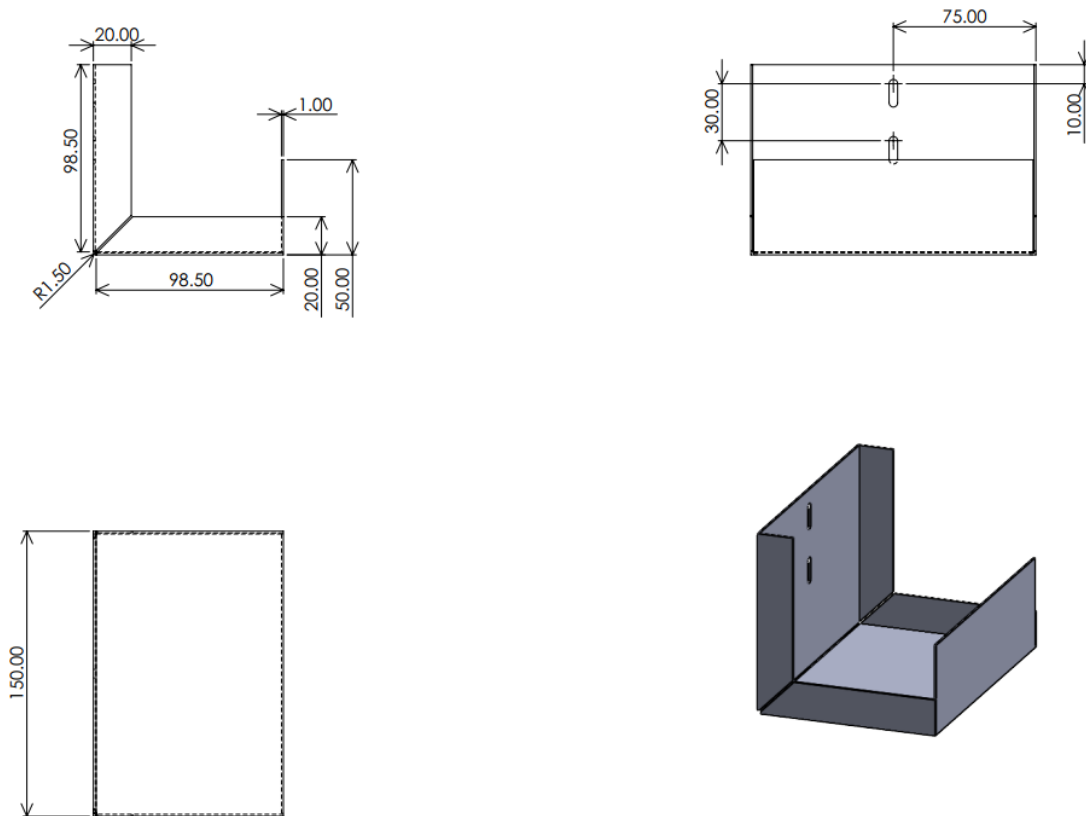


Figure 2.18 Dimensions of the Cover

These components are located at Station 5: the Camera station of the Wire Soldering machine. They are used to mount the camera, separate the two products to prevent image interference, and provide shielding to stabilize the internal lighting, ensuring that the captured images have minimal lighting variation.

The visual appearance of the system after all components have been assembled onto the machine.

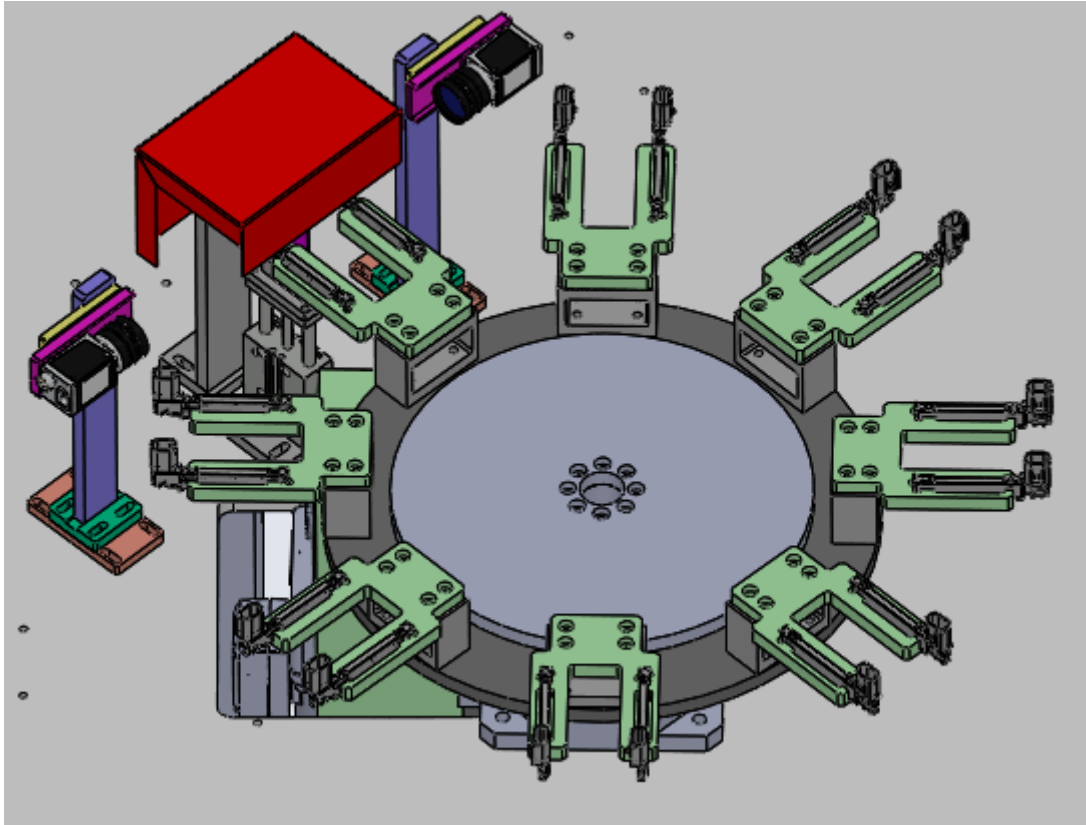


Figure 2. 19 The visual appearance of the system after all components have been assembled onto the machine.

CHAPTER 3: CONTROL SYSTEM DESIGN

3.1. System overview

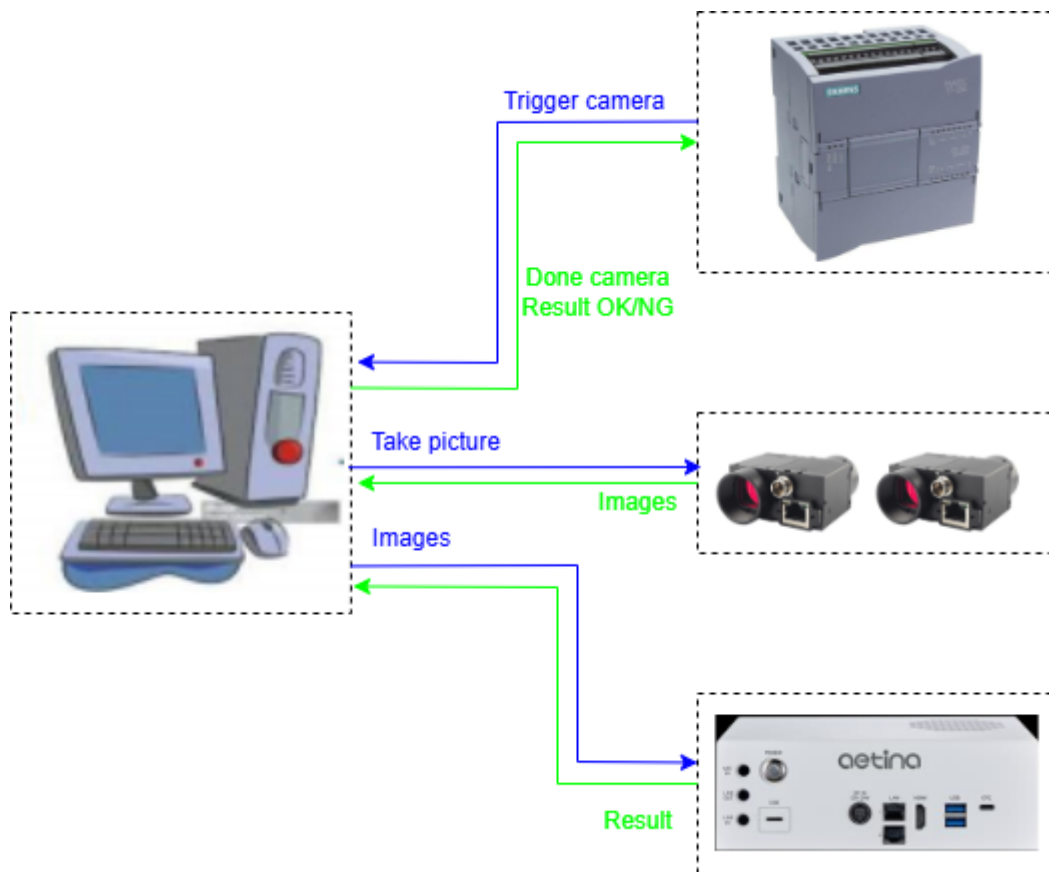


Figure 3. 1 System overview

The system uses a computer (running C# software) as the central coordinator, acting as a bridge between the PLC, cameras, and Jetson. The operation process follows these steps:

1. The PLC sends a trigger signal to the C# application to request image capture.
2. The C# application receives the signal from the PLC and sends control commands to the cameras to capture images.
3. Image data from the cameras is sent back to the C# application.
4. The C# application transmits the images to the Jetson for AI model analysis and result generation. At the same time, it sends a "camera done" signal back to the PLC to indicate that image capture is complete.

5. After processing, the Jetson returns the results to the C# application, including the image, bounding boxes, and OK/NG labels.
6. Once the C# application receives the results, it displays them on the user interface and sends the OK/NG result back to the PLC for the next steps.

For the system to operate efficiently, stably, and accurately, it is necessary to ensure both of the following core factors simultaneously:

- **Equipment selection:** Selecting the appropriate equipment plays a critical role in the performance, reliability, and overall efficiency of the system.
 - **Connection and communication between devices:** Ensuring seamless and accurate connectivity between devices is a prerequisite for the system to operate smoothly without communication errors during operation.
 - **AI model training:** Besides the hardware factors, the quality of the system largely depends on the accuracy and stability of the AI model. Therefore, the model training process needs to be conducted methodically, with a clear data strategy and continuous evaluation throughout development.
- The three factors above form the foundation that determines the success of the automatic solder joint inspection system. Focusing on only one factor while neglecting the other may cause operational errors, inaccurate results, or failure to meet the required production line speed.

3.2. Equipment selection

3.2.1. Camera

In the AI-based solder joint defect inspection system, selecting the appropriate camera plays a crucial role in ensuring the quality of input images, directly affecting the accuracy of the recognition model. After considering factors such as quality, performance, and stability, we decided to use industrial cameras instead of conventional ones.

First, high image quality is the most important factor. Industrial cameras offer high resolution, good light sensitivity, and more accurate color reproduction compared to conventional cameras. This allows the system to capture sharp images of solder joints.

Second, the high stability and durability of industrial cameras are crucial for our system. The welding environment presents many challenges, such as high temperatures, vibrations, and solder fumes. Industrial cameras are designed to withstand these conditions without compromising performance. They can operate continuously 24/7 without overheating or degrading image quality over time, ensuring the system runs stably for long periods without the need for frequent replacements or maintenance.

Third, high processing speed and flexible connectivity are major advantages of industrial cameras. In production lines, especially real-time solder joint inspection systems, the camera's data capture and transmission speed must be fast enough to avoid slowing down the manufacturing process. Industrial cameras support high-speed data transfer protocols such as GigE, USB 3.0, or Camera Link, enabling the system to process images quickly and accurately.

Fourth, the ability to customize with specialized lenses and filters makes the system more flexible in image acquisition. Industrial cameras support various lenses with different focal lengths and apertures, allowing image optimization for specific conditions.

Finally, the ease of customization and integration with the AI system is a key reason for choosing industrial cameras.

With these advantages, using industrial cameras is the optimal choice to ensure accuracy, stability, and high performance for the AI-based solder joint defect inspection system. This not only enables precise defect detection but also enhances product quality, minimizes defect rates, and optimizes the manufacturing process.

Table 3. 1 Comparison between GigE Camera and USB 3.0 Camera

	GigE Camera	USB 3.0 Camera
Data output	1 Gbps	5 Gbps
Max framerate (fps)	25	80
Cable length (m)	100	4.6
Devices connection	LAN port	USB port
Connecting cable	Ethernet	USB
Power supply	Use a separate power supply	PC power supply
Price	No difference in price	

Choose to use GigE camera because:

- GigE uses a LAN port – common in industrial environments and easy to integrate with existing infrastructure. Multiple cameras can be connected via network switch.
- Using a separate power supply provides more flexibility in delivering stable power, especially when relying on power from the PC is not possible.
- GigE can transmit data continuously and more stably over long distances, suitable for systems requiring 24/7 operation such as industrial solder inspection systems.

Based on the GigE cameras available at PREMO company, we use the HT-GE502C-T1P-C camera in the solder inspection system.

Table 3. 2 Technical specifications of the HT-GE502C-T1P-C camera

Specifications	Value
Sensor Type	CMOS
Sensor Format	1/2.5"
Resolution (H x V)	2592 px × 1944 px
Pixel Size (H x V)	2.2 μm × 2.2 μm
Dynamic range	74.3 dB
Frame Rate	24 fps
Signal-to-Noise Ratio (SNR)	40 dB



Figure 3. 2 The HT-GE502C-T1P-C camera

3.2.2. Lens

The lens determines the field of view, magnification, and image quality obtained. Choosing the appropriate lens helps ensure clear images with sufficient information for the AI/image analysis system to operate accurately.

Table 3. 3 Parameters affecting lens selection

Parameter	Value
Pixel size	2.2 μ m $\times$ 2.2 μ m
Resolution	2592px $\times$ 1944px
Working distance (WD)	80mm
Field of View (FoV)	25mm $\times$ 25 mm
Mount lens	C-Mount
Camera application	Soldering inspection system

$$\text{Sensor size} = \text{Pixel size} \times \text{Resolution} = (2.2\mu\text{m} \times 2592) \times (2.2\mu\text{m} \times 1944)$$

$$\text{Sensor size} = 5.7\text{mm} \times 4.3\text{mm}$$

Calculate the lens focal length:

$$\text{Focal Length} = \frac{\text{Sensor size} \times \text{Working Distance}}{\text{FoV}}$$

$$\text{Focal Length} = \frac{7.14\text{mm} \times 80\text{mm}}{25\text{mm}} = 22.85\text{mm}$$

Choose a lens with a 25mm focal length to ensure the field of view covers the object sufficiently and maintains appropriate spatial resolution for solder defect detection.

Based on calculations and system requirements, the 25mm C-mount lens supporting 5MP resolution is a suitable choice for the solder inspection system with a 1/2.5" sensor camera. The lens meets the desired field of view at a working distance of 80mm.

Choose lens M2514-MP2.

Model No.		M2514-MP2		Effective	Front	φ 17.8mm	
Focal Length		25mm		Lens Aperture	Rear	φ 12.0mm	
Max. Aperture Ratio		1:1.4		Distortion	2/3type	-0.3%(y=5.5)	
Max. Image Format		8.8mm x 6.6mm(φ 11mm)			1/2type	-0.1%(y=4.0)	
Operation Range	Iris	F1.4 - F16C		Back Focal Length		13.1mm	
	Focus	0.3m - Inf.		Flange Back Length		17.526mm	
Control	Iris	Manual		Mount		C-Mount	
	Focus	Manual		Filter Size		M30.5 P=0.5mm	
Object Dimension at M.O.D	2/3 type	10.6cm x 7.9cm		Dimensions		φ 33.5mm x 36mm	
	1/2 type	7.7cm x 5.8cm		Weight		71.2g	
Angle of View	D	2/3 type	24.9°	1/2 type	18.2°		
	H		20.0°		14.6°		
	V		15.1°		11.0°		
Operating Temperature		-10℃ - +50℃					

M.O.D : Minimum Object Distance

Figure 3. 3 Specifications of M2514-MP2

Dimensions

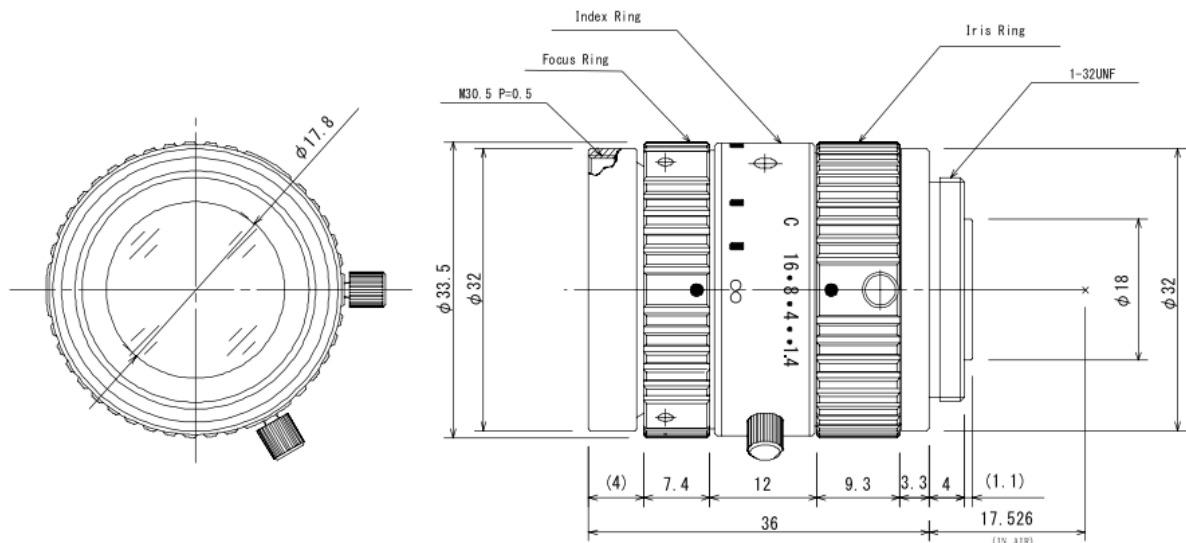


Figure 3. 4 Dimensions of the lens M2514-MP2

3.2.3. Switch PoE

In the AI Soldering Inspection project, we chose a PoE (Power over Ethernet) switch to power GigE cameras due to several key advantages.

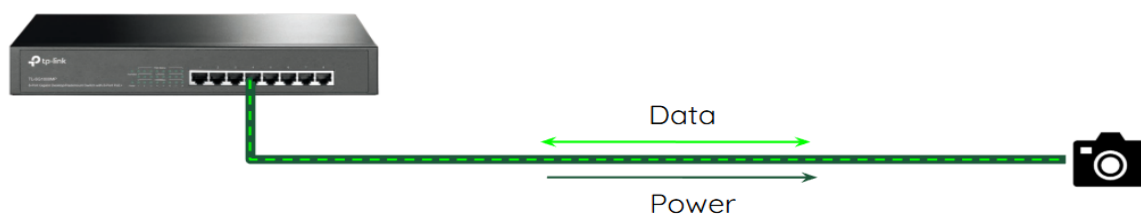


Figure 3. 5 PoE switch supplies power and transmits data at the same time

First, PoE helps reduce the number of cables, requiring only a single Ethernet cable to transmit both data and power, making the system tidier and easier to manage. This also enables flexible deployment, allowing cameras to be installed in various locations without needing nearby power outlets, while also ensuring easy scalability when adding more devices without worrying about separate power sources.

Additionally, a PoE switch ensures a stable power supply, preventing disruptions caused by inconsistent power sources, and allows for remote monitoring and control with smart PoE switches. Compared to running separate power lines for each camera, a PoE switch reduces installation and operational costs while minimizing labor efforts, especially in an industrial production environment.

With these advantages, a PoE switch is the optimal choice for the AI soldering inspection system using GigE cameras.

Table 3. 4 Comparison of PoE switches available at PREMO company

	TL-SG1008MP	TL-SG2210P
Number of Gigabit Ports	8 ports (PoE)	10 ports (8 ports PoE + 2 ports SFP)
Total PoE Power Budget	126W	63W
SFP Support	No	2 SFP ports
Usage	Just plug & play, no configuration required	Web-based management capabilities
Price	≈ 1.700.000 VND	≈ 3.000.000 VND

Choose to use TL-SG1008MP because the system only needs to supply power to 2 cameras and exchange data between the cameras and the PC.



Figure 3. 6 Switch PoE TL-SG1008MP

3.2.4. NVIDIA Jetson

In the context of the 4.0 industrial revolution driving the rapid development of artificial intelligence (AI) and automation technologies, the application of modern solutions in production processes has become a key factor in enhancing productivity and product quality. Particularly in product inspection systems after soldering—a crucial step determining the stability and safety of electronic products—accuracy and timeliness in defect detection are essential. NVIDIA Jetson Orin NX 16GB, with its superior processing power, energy efficiency, and high flexibility, is a groundbreaking solution that brings a new advancement to quality inspection in the electronics industry.



Figure 3. 7 Jetson Orin NX

The Necessity of Using NVIDIA Jetson Orin NX in Product Inspection:

In the product inspection system after soldering, quickly and accurately detecting defects during the soldering process is vital to ensure quality and reduce errors. Jetson Orin NX meets these requirements with its outstanding features:

- Outstanding processing performance: Equipped with an Ampere architecture GPU with 1024 cores and 16GB LPDDR5 RAM, Jetson Orin NX 16GB can handle multitasking and complex data processing in real-time, enabling quick and accurate detection of soldering defects.

- Real-time data analysis and feedback: The ability to process and analyze data instantly allows the inspection system to continuously monitor quality, detect, and alert defects immediately, thus optimizing the production process.
- Easy integration and development: The JetPack SDK provides a complete development environment with all necessary AI libraries and optimization tools, minimizing development time and facilitating the deployment of solutions in existing systems.
- Energy savings and operational cost reduction: Jetson Orin NX is configurable with power consumption ranging from 10W to 25W, balancing performance and energy savings, helping businesses reduce operational costs while protecting the environment.

The Innovations Brought by Jetson Orin NX to the Production Process:

The NVIDIA Jetson Orin NX technology transforms how businesses inspect the quality of products after soldering by applying AI and processing data at the edge (edge computing) with the following advantages:

- Accelerating the inspection process: Jetson Orin NX enhances speed and accuracy compared to manual inspection or traditional systems, allowing better responsiveness to fast production demands and large volumes.
- Advanced AI applications: Integrating deep learning models like YOLO helps automatically detect soldering defects, reducing reliance on human labor and minimizing human error.
- Data utilization and production optimization: Processing data directly at the device increases efficiency, reduces latency, and provides a smarter and more flexible production process, aligning with the trends of Industry 4.0.

With the full consensus of the company, we decided to adopt the Jetson Orin NX 16GB module for our system.

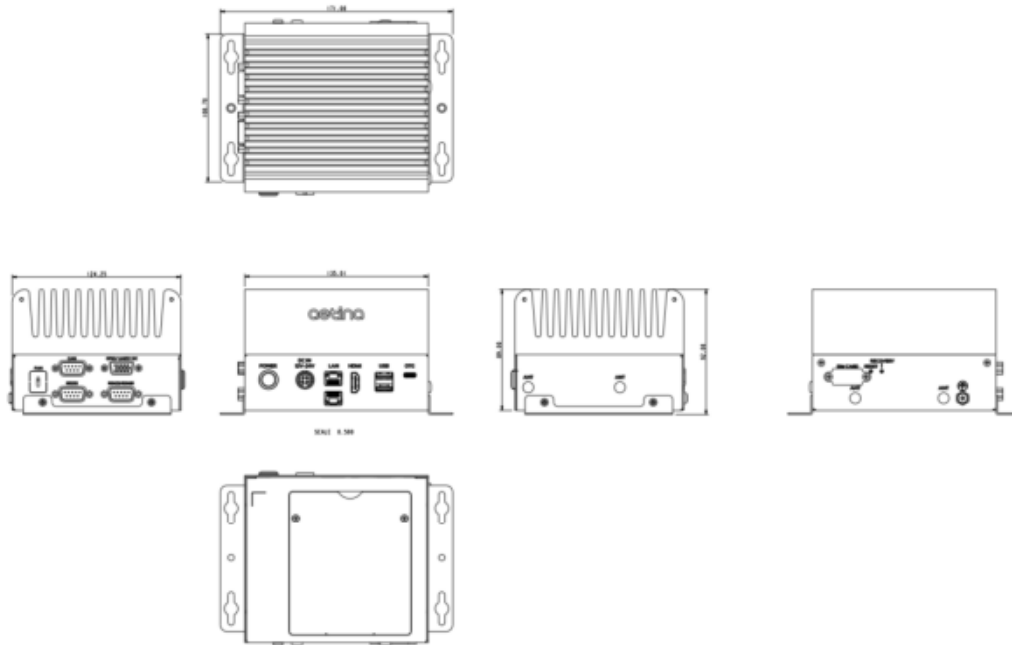


Figure 3. 8 System & Mounting Dimensions

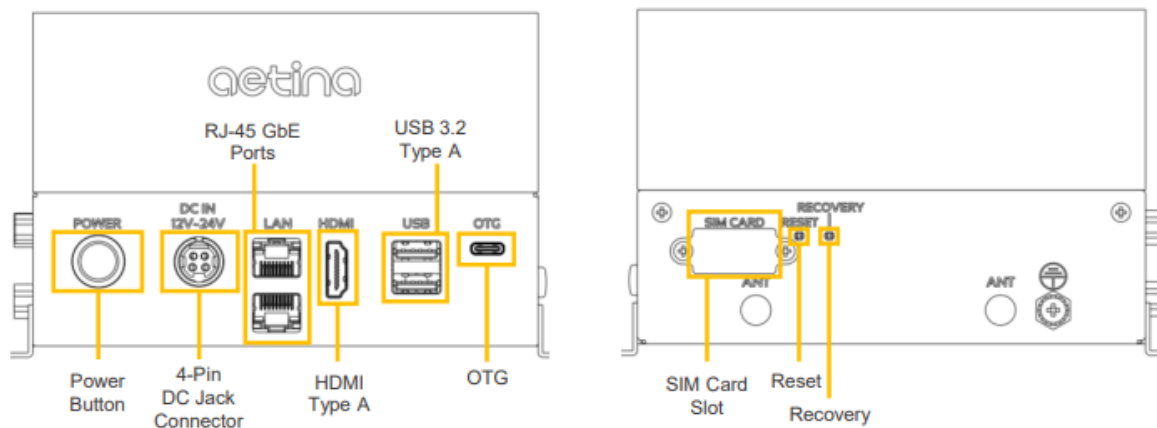


Figure 3. 9 I/O Mechanical Layout

Table 3. 5 Technical specifications

Parameters	Description
AI Performance	100 TOPS
GPU	1024-core NVIDIA Ampere, with 32 Tensor cores
CPU	8-core Arm Cortex-A78AE v8.2 64-bit
Memory	16GB LPDDR5 128-bit. (Bandwidth: 102.4 GB/s)
LAN	2 × RJ45 GbE port

3.2.5. PLC

In the field of industrial automation, selecting a central control unit is a critical factor that greatly influences the efficiency and reliability of the entire system. A PLC (Programmable Logic Controller) is a specialized control device designed to operate reliably in harsh industrial environments, where conventional microcontrollers may not meet the requirements in terms of stability, scalability, and flexibility in programming and maintenance.

Compared to other control solutions, PLCs stand out thanks to their user-friendly programming via standard languages such as LAD (Ladder Logic), FBD (Function Block Diagram), STL (Statement List), etc., along with the ability to communicate with a wide range of peripheral devices using popular industrial protocols like Profinet, Modbus TCP, and more. This makes system integration simpler and more time-efficient. After careful consideration and technical consultation from PREMO Vietnam Ltd, our team decided to use the Siemens S7-1214C DC/DC/DC PLC for the project. This model belongs to Siemens' well-known S7-1200 series and offers strong processing capabilities, a high number of digital I/O ports, built-in industrial communication support, and is particularly well-suited for small to medium-sized automation applications. The DC/DC/DC configuration (DC power supply, DC digital inputs, DC digital outputs) matches our system's electrical setup, allowing for cost-effective implementation and safe operation.

With these advantages, the Siemens S7-1214C DC/DC/DC PLC fully meets the technical requirements and practical demands of our system, while also providing a solid foundation for future expansion and upgrades.



Figure 3. 10 PLC 1214C DC/DC/DC

Table 3. 6 Some characteristic parameters of PLC 1214C DC/DC/DC

Memory	150 kilobyte
Digital Inputs	14
Digital Outputs	10
Analog Inputs	2 (0-10V)
Supply Voltage	20.4-28.8V (24VDC)
Resolution	10 bit
Communication	PROFINET
Interface	Ethernet RJ45

3.3. Connecting devices

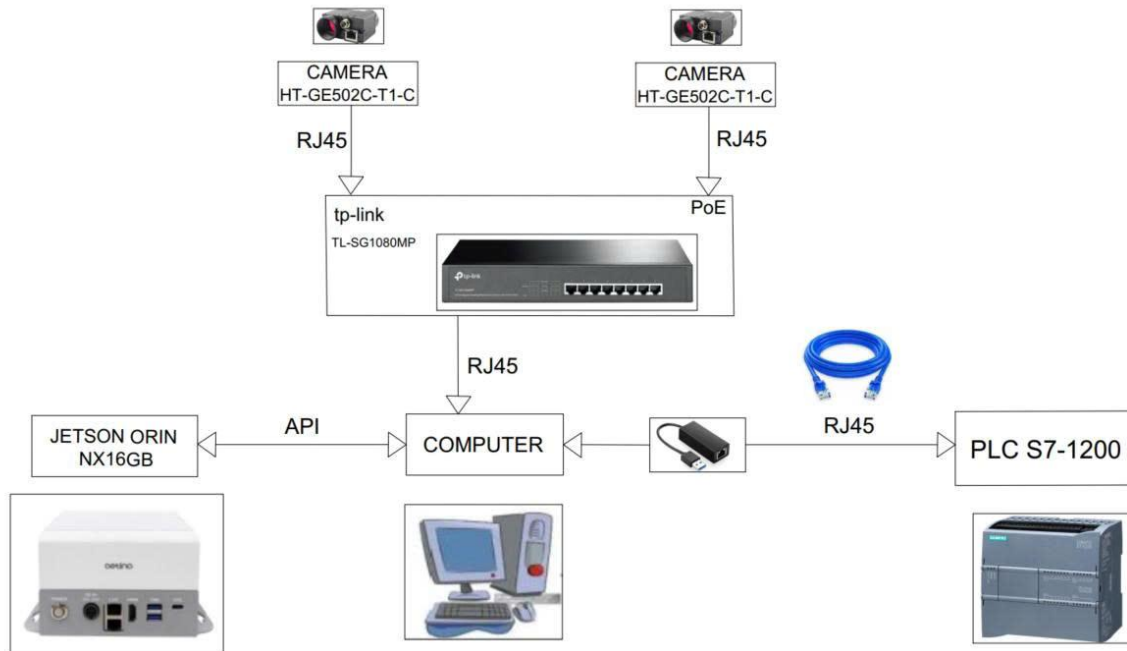


Figure 3. 11 Device connection diagram

The connection and communication between devices are carried out via Ethernet network as follows:

- The PoE switch plays the role of supplying power and transmitting data:
 - Supply power and transmit data for two GigE cameras.
 - Power the NVIDIA Jetson device via the PoE port.
 - Establish a local network connection between devices including PC, cameras, and Jetson.
- The PC is connected to the PoE switch to:
 - Communicate and control two cameras through the manufacturer's SDK.
 - Exchange data with NVIDIA Jetson via the local network.
- To ensure flexibility in system configuration and easy changes when necessary, create a "config.txt" configuration file in the C# software to store:
 - IP address of the S7-1200 PLC.
 - Serial Number of each camera.

- IP address of the server.
- On the Jetson (running the AI program), it is necessary to set the PC's IP address in the source code so that the Jetson can send result data (images, OK/NG labels) to the C# software via TCP/IP network communication.
- The PLC communicates with the PC using the S7.Net library, which enables reading/writing data to the PLC memory area via IP address.

3.3.1. Fast API

FastAPI is a modern, high-performance Python web framework for building APIs, designed for rapid development and efficient handling of asynchronous tasks. Built on Starlette and Pydantic, FastAPI is ideal for processing large data payloads, such as images, in real-time. In this thesis, FastAPI is deployed on the Jetson server to receive images from the C# application, process them using computer vision algorithms, and return results over the LAN.

Roles in the System:

- **Image Reception:** FastAPI provides API endpoints on the Jetson server to accept image files sent from the C# application via HTTP POST requests over TCP/IP.
- **Efficient Image Processing:** FastAPI's asynchronous capabilities enable the Jetson server to process images concurrently using libraries like OpenCV or TensorFlow, supporting tasks like object detection or quality inspection.
- **Result Delivery:** FastAPI returns processing results (e.g., analysis data) to the C# application via JSON responses, enabling the system to act on outcomes, such as updating the user interface or triggering PLC actions.
- **Data Validation:** FastAPI uses Pydantic to validate incoming image data, ensuring only correct data is processed, enhancing system reliability.
- **Scalability:** FastAPI's modular design allows easy addition of new endpoints for advanced image processing tasks, supporting system expansion.

3.3.2. S7.Net library

The S7.Net library is an open-source library developed to facilitate communication between C# applications and Siemens programmable logic controllers (PLCs), particularly the S7-1200 series. This library enables developers to read and write data to/from PLCs via the TCP/IP communication protocol, leveraging Ethernet/Profinet connectivity. S7.Net is designed to be simple, user-friendly, and suitable for industrial automation applications, ranging from monitoring and data acquisition to device control.

Key Features of S7.Net:

- **Support for Siemens PLCs:** S7.Net is compatible with S7-1200, S7-1500, S7-300, and S7-400 PLCs, making it easy to integrate into existing automation systems.
- **TCP/IP Communication:** The library uses the TCP/IP protocol to connect with PLCs, ensuring fast and stable data transmission, suitable for real-time applications such as camera control.
- **Versatile Data Read/Write:** S7.Net supports access to various data types, including bits, bytes, words, double words, and Data Blocks (DB) in PLCs.
- **Seamless Integration with C#:** Designed specifically for the .NET environment, S7.Net allows C# developers to easily implement applications interfacing with PLCs without requiring complex tools like OPC Server.
- **Open Source and Community Support:** As an open-source project, S7.Net allows customization and extension based on project requirements. Its large developer community provides documentation and technical support.
- **In this System,** the S7.Net library was selected to connect the C# application with the S7-1200 PLC due to the following reasons:

- **Simplicity and Efficiency:** S7.Net provides easy-to-use classes and methods, reducing development time and simplifying the programming of PLC communication.
- **High Performance:** The TCP/IP protocol of S7.Net ensures fast data transmission, meeting the real-time requirements of the camera control system.
- **Compatibility with S7-1200 PLC:** The library fully supports the features of the S7-1200 PLC, including reading/writing data from Data Blocks (DB) and controlling outputs to trigger cameras.
- **Cost-Free and Flexible:** As an open-source library, S7.Net requires no licensing fees, making it suitable for academic projects like this thesis. Its flexibility allows customization to meet specific system requirements.
- **Integration with Visual Studio 2022:** S7.Net can be easily installed via the NuGet Package Manager in Visual Studio, streamlining its integration into C# projects.

Application of S7.Net in the System:

- In this system, the S7.Net library was utilized to:
- **Connect to the S7-1200 PLC:** Establish communication between the C# application and the S7-1200 PLC via Ethernet.
- **Control Cameras:** Send control signals from the C# application to the PLC to activate the two cameras for image capture based on pre-programmed scenarios.
- **Data Acquisition and Processing:** Read status data from the PLC (e.g., camera status, feedback signals) to monitor and adjust the system's operation.
- **User Interface Integration:** Combine with the C# user interface to display system status and provide control buttons for operators.

3.3.3. TCP/IP protocol

TCP/IP (Transmission Control Protocol/Internet Protocol) is a foundational suite of communication protocols that enables reliable, ordered, and error-checked data

transmission between devices over a network. In this thesis, TCP/IP facilitates communication between the C# application (running on a personal computer), the Siemens S7-1200 PLC, and the Jetson server, all connected via a LAN cable. It serves as the backbone for data exchange, ensuring robust and real-time communication for camera control and image processing.

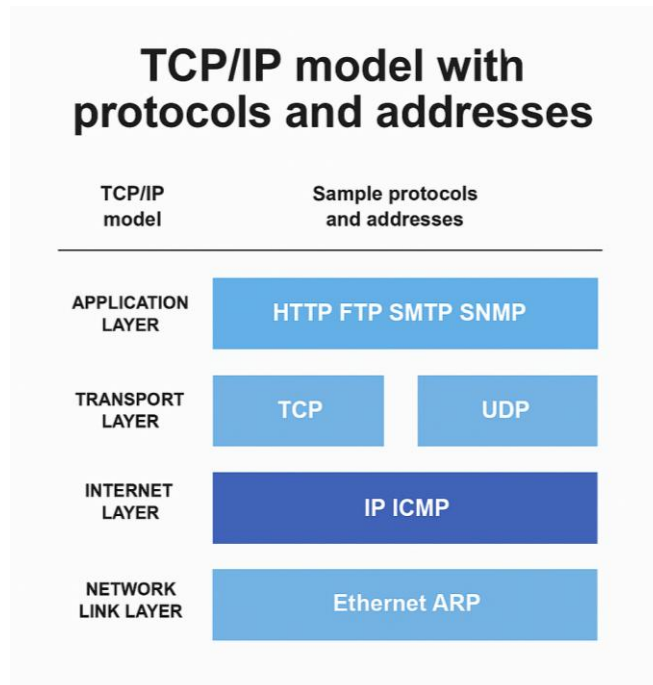


Figure 3. 12 TCP/IP model

Roles in the System:

- **Reliable Communication:** TCP/IP ensures stable data exchange between the C# application and the S7-1200 PLC using the S7.Net library, enabling accurate transmission of camera control commands (e.g., triggering image capture) and status feedback over the LAN.
- **Image Data Transfer:** TCP/IP supports the transfer of image files from the C# application to the Jetson server via FastAPI, ensuring large image payloads are delivered without loss or corruption.
- **Real-Time Performance:** The protocol's low-latency and error-checking mechanisms enable real-time coordination between camera control (via PLC) and image processing (via Jetson), critical for time-sensitive operations.

- **Network Stability:** Using a LAN cable, TCP/IP leverages Ethernet's high-speed and reliable connectivity, minimizing delays and ensuring consistent communication across the system.
- **Scalability:** TCP/IP allows the system to integrate additional devices (more cameras or servers) over the LAN, supporting future expansion without major infrastructure changes.

3.4. Training the AI model

3.4.1. Prepare the data file

To train an AI model, preparing the dataset is an essential step. A sufficiently large, diverse, and high-quality image dataset must be collected and constructed to accurately reflect real-world scenarios (OK/NG). These images need to be precisely labeled so that the model can learn the characteristics of each type of object, thereby improving accuracy and reliability during classification after deployment.

3.4.1.1. Collecting data

❖ PLC diagram: Collecting data

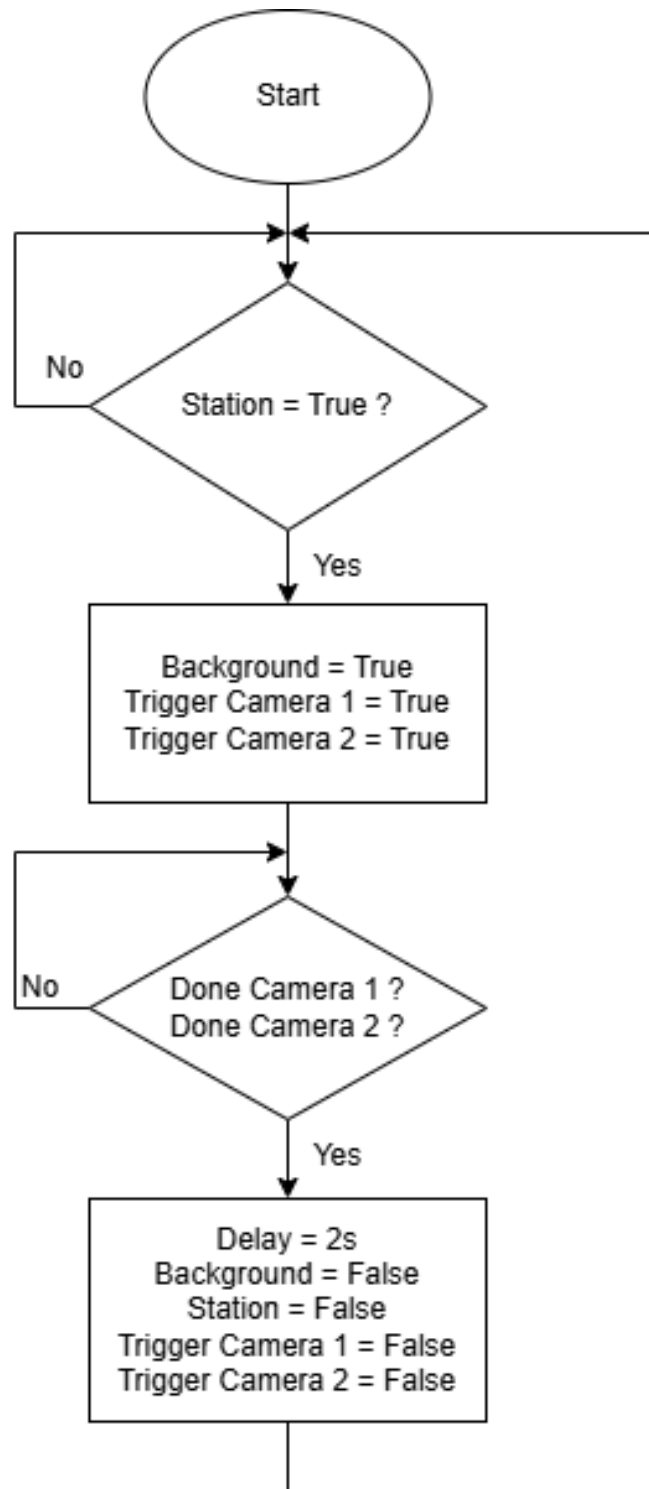


Figure 3. 13 PLC diagram: Collecting data

Algorithm Flow Description:

The system flowchart illustrated in Figure 3.13 outlines the image acquisition control process using two industrial cameras. The logic is governed by the PLC and C# application via signal exchange.

Initially, the system remains in a waiting state until the condition Station = True is satisfied, indicating that a product is positioned correctly at the inspection station. Once this condition is met, the background is enabled (Background = True) to notify the C# application of readiness.

Subsequently, trigger signals are sent to both cameras (Trigger Camera 1 = True and Trigger Camera 2 = True) to initiate simultaneous image capture. The system then monitors the completion flags from both cameras (Done Camera 1 and Done Camera 2). Once both flags are confirmed, the system proceeds to the post-capture stage.

A timer (T = 2s) is used to ensure sufficient processing time before resetting the system state. After the timer elapses, the background signal is deactivated (Background = False), the station flag is reset (Station = False), and trigger signals are set to false (Trigger Camera 1 = False, Trigger Camera 2 = False), preparing the system for the next cycle.

❖ Csharp diagram: Collecting data

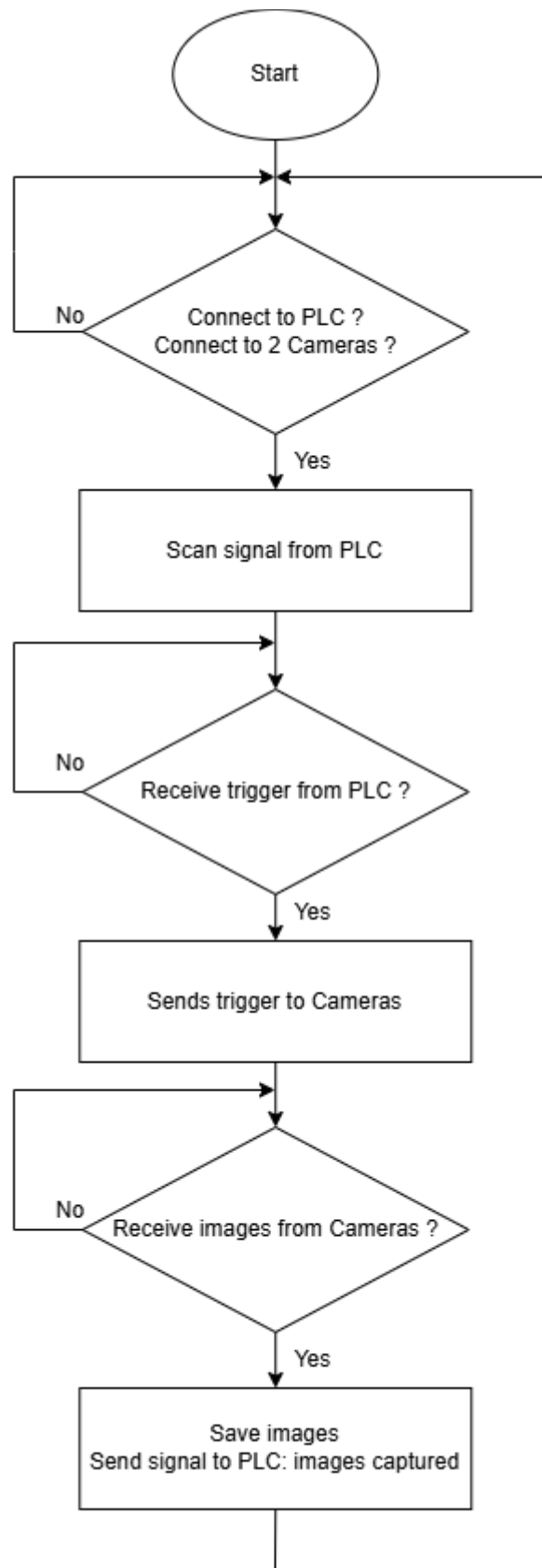


Figure 3. 14 CSharp diagram: Collecting data

Algorithm Flow Description:

The flowchart in Figure 3.14 illustrates the image acquisition and control process from two industrial cameras via control signals from the PLC. The algorithm is implemented in the C# communication software, acting as an intermediary between the PLC and the image acquisition devices.

The process begins with the initialization of communication with the PLC and the two connected cameras. Once a successful connection is established, the system continuously scans for trigger signals from the PLC.

Upon receiving a valid trigger signal, the software sends capture commands to both cameras simultaneously. The system then waits until image data from both cameras are received. Once the images are successfully acquired, the software stores the images in a predefined location.

After completing the image-saving process, the system sends a feedback signal to the PLC to confirm that the images have been captured and saved successfully. The cycle then returns to the initial state, ready to process the next product.

3.4.1.2. Data classification

The data used for training and evaluating the AI model consists of a total of 1,100 images collected by each camera, including 550 OK images (products meeting requirements) and 550 NG images (products not meeting requirements). This balanced dataset helps the model fully learn the characteristics of both product types.

From the total 1,000 images, we split them into two sets: training and validation, in the ratio of 9:1. Specifically, 900 images are used to train the model, 100 images are used for validation to monitor and evaluate accuracy during training, thereby reducing overfitting. The remaining 100 images from the original dataset are kept untouched and do not participate in training. They are used to test the model's real-world performance after training is completed.

The number of OK and NG product images in each set is equal. This data division method ensures that the model can learn well from the data while maintaining objectivity during evaluation.

3.4.1.3. Labelling

In this step, we use the LabellImg software to perform labeling for the training and validation image sets of the AI model. This is an important step to provide quality input data, helping the model learn the image features related to the objects to be detected. Therefore, this step must be carried out carefully and accurately.

First, open and edit the file “predefined_classes.txt” – this file contains the list of classes that the user wants to use during the labeling process. In this project, we define two main classes, including:

- PIN OK: represents pins that are soldered correctly and meet requirements.
- PIN NG: represents pins that are soldered inadequately and do not meet requirements.

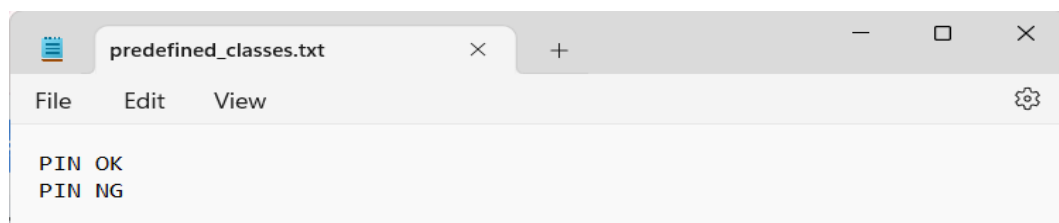


Figure 3. 15 Example of the predefined_classes.txt file

After defining the classes, launch the LabellImg software and set the label saving format to YOLO to ensure compatibility with the YOLO model that will be used later.

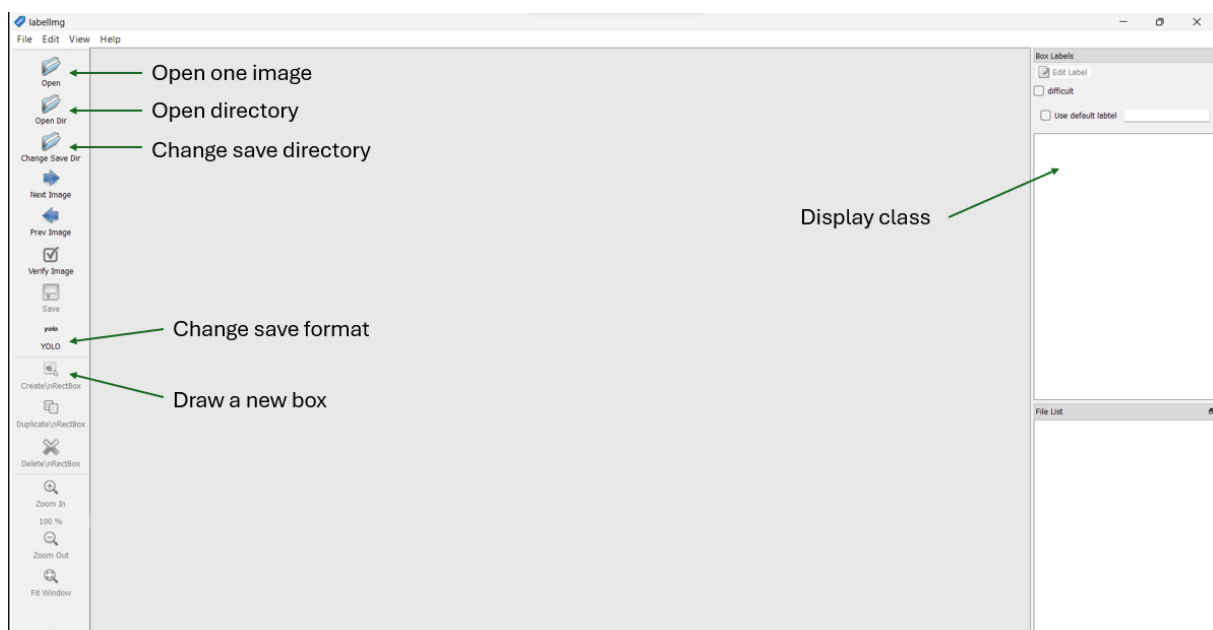


Figure 3. 16 Interface of the LabellImg software

Next, open an image in the train/validation dataset and manually draw bounding boxes around the two pins appearing in the image. Each bounding box will be assigned a corresponding class: if the pin meets the requirements, select “PIN OK”; otherwise, select “PIN NG.”



Figure 3. 17 Label an image

After completing the labeling of the images, the software will automatically generate a “.txt” text file containing information about the bounding boxes and corresponding classes (coordinates normalized according to the YOLO format). This file will be saved in the same folder as the corresponding image.

Finally, we use this “.txt” file as the basis for performing auto labeling on the train/validation dataset with a Python program. Then, we open the LabelImg software to review and correct any bounding boxes assigned to the wrong class. This helps us save time and improve consistency in the dataset.

3.4.1.4. Data folder organization

After completing the labeling process for the images, we organize and arrange the data into directories systematically to prepare for AI model training. Proper data structuring will help the YOLO model accurately recognize the images.

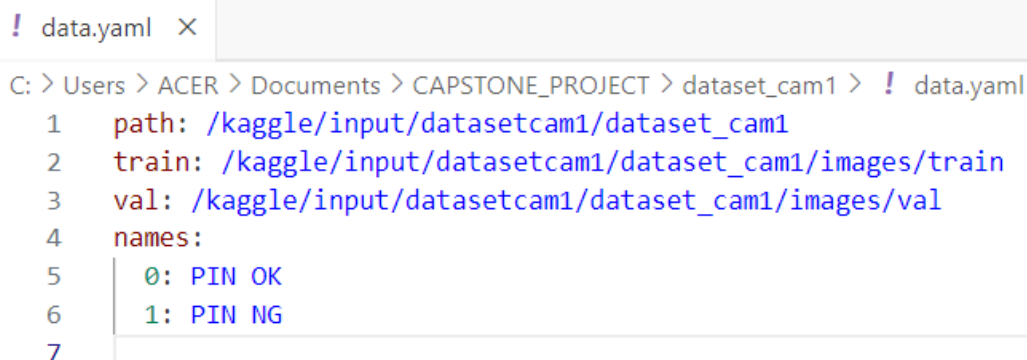
Organize the data into separate directories for each camera as follows:

- dataset_cam1 / dataset_cam2
 - images
 - train
 - val
 - label
 - train
 - val
 - data.yaml

In which:

- The “images/train” and “images/val” folders contain images for training and validation.
- The “label/train” and “label/val” folders contain the corresponding “.txt” label files.
- The “data.yaml” file serves as the main configuration file, helping the model know where to find images, labels, and information about the classes to be recognized.

Create the “data.yaml” file as follows:



```
! data.yaml X
C: > Users > ACER > Documents > CAPSTONE_PROJECT > dataset_cam1 > ! data.yaml
1 path: /kaggle/input/datasetcam1/dataset_cam1
2 train: /kaggle/input/datasetcam1/dataset_cam1/images/train
3 val: /kaggle/input/datasetcam1/dataset_cam1/images/val
4 names:
5   0: PIN OK
6   1: PIN NG
7
```

Figure 3. 18 Example of the data.yaml file

In which:

- path: root path to the main data folder “dataset_cam1” / “dataset_cam2”.
- train: path to the “images/train” folder.
- val: path to the “images/val” folder.
- names: list of classes the model needs to learn.

3.4.2. Training model

3.4.2.1. Model selection

During the development of the AI solder inspection system, selecting the appropriate model version plays a crucial role, directly affecting accuracy, performance, and practical deployment. YOLO stands out with its ability to process images quickly while maintaining high accuracy. Therefore, we chose the YOLO model as the main foundation for the system.

We selected the YOLOv8 model for our project without using higher YOLO versions for the following reasons:

- YOLOv8 is a version that strikes a good balance between accuracy and processing speed, featuring a more modern architecture compared to previous versions, while not being too heavy compared to later versions.
- YOLOv8 has been widely adopted by the community, so it has more supporting documentation. This helps us easily find solutions to various issues.
- For the solder inspection system, high accuracy is required but not excessively strict.

3.4.2.2. Training model

After completing the data preparation process, we proceeded to train the AI model. The training was conducted on the Kaggle platform, which provides a convenient programming environment and suitable hardware resources for AI model training.

To carry out the model training process on the Kaggle platform, we followed these steps:

- Upload the necessary data files, including the training dataset and test images.
- Set up the working environment, specifically by installing the Ultralytics library – a tool that supports training and deploying YOLO models.

```
pip install ultralytics opencv-python matplotlib
```

Figure 3. 19 Code to set up the working program

- Configure and run the training script, including setting up data paths, training parameters, and initiating the model training process.

```
from ultralytics import YOLO

# Load mô hình YOLOv8l detection
model = YOLO("yolov8l.pt")

# Huấn luyện với data.yaml
model.train(
    data='/kaggle/input/datasetcam1/dataset_cam1/dataset.yaml',
    epochs=100,
    imgsz=640,
    batch=4
)
```

Figure 3. 20 AI model training code

3.4.3. Result

3.4.3.1. Model 1

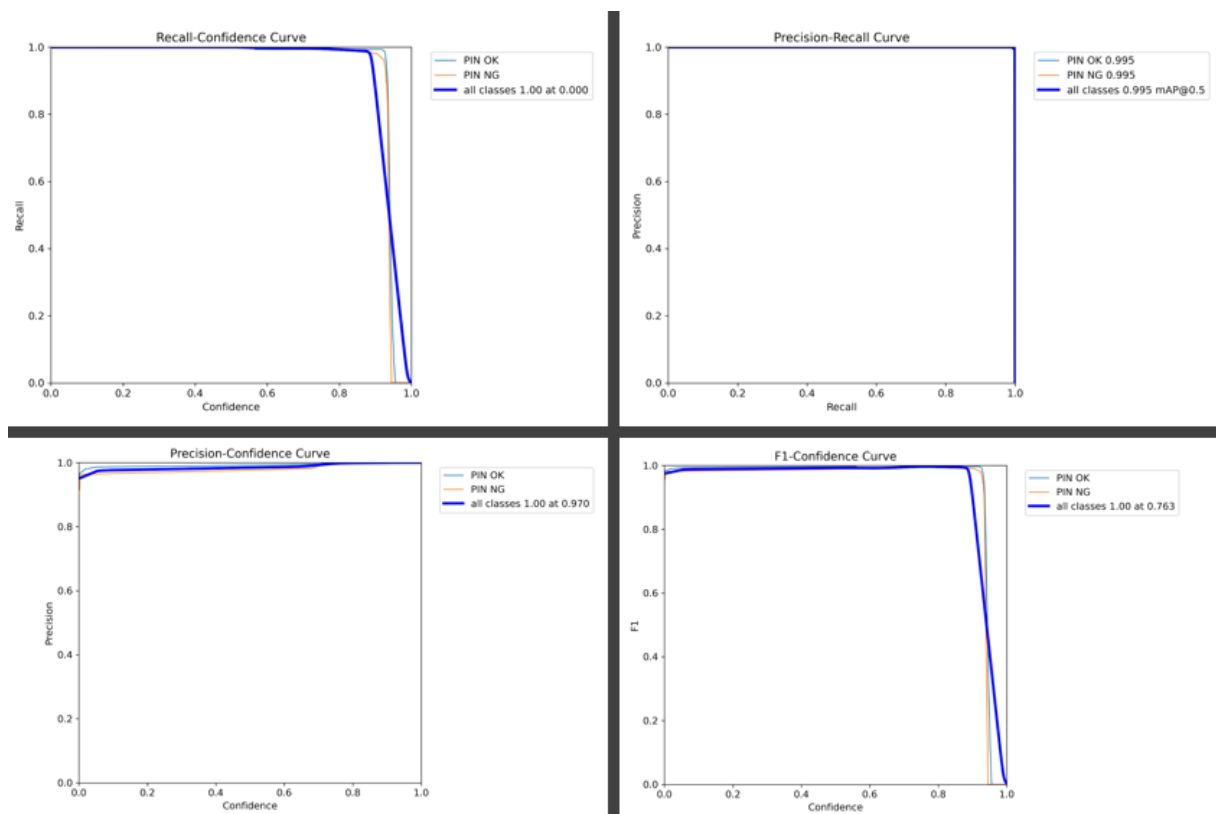


Figure 3. 21 Result charts of Model 1

Comments:

- The charts indicate that the model appears to be well-balanced between precision and recall.

The precision-recall curve shows a large area under the curve ($mAP@0.5 = 0.995$), indicating that the model performs effectively across various confidence thresholds.

- The F1-confidence curve reaches its peak ($F1 = 1.00$) at a confidence threshold of 0.763, suggesting that the model maintains a strong balance between precision and recall when operating around that threshold.
- Meanwhile, the precision-confidence curve shows that the model achieves perfect precision (1.00) at a confidence of 0.970, which is ideal in applications that prioritize accuracy and minimal false positives.
- The recall-confidence curve confirms that the model sustains high recall (~ 1.00) at very low thresholds (starting from 0.000), which means it is highly effective at identifying all relevant instances, even in uncertain cases.

→ The classification model demonstrates high and stable performance, with a good balance between precision and recall. It exhibits excellent discriminative ability between the "PIN OK" and "PIN NG" classes.

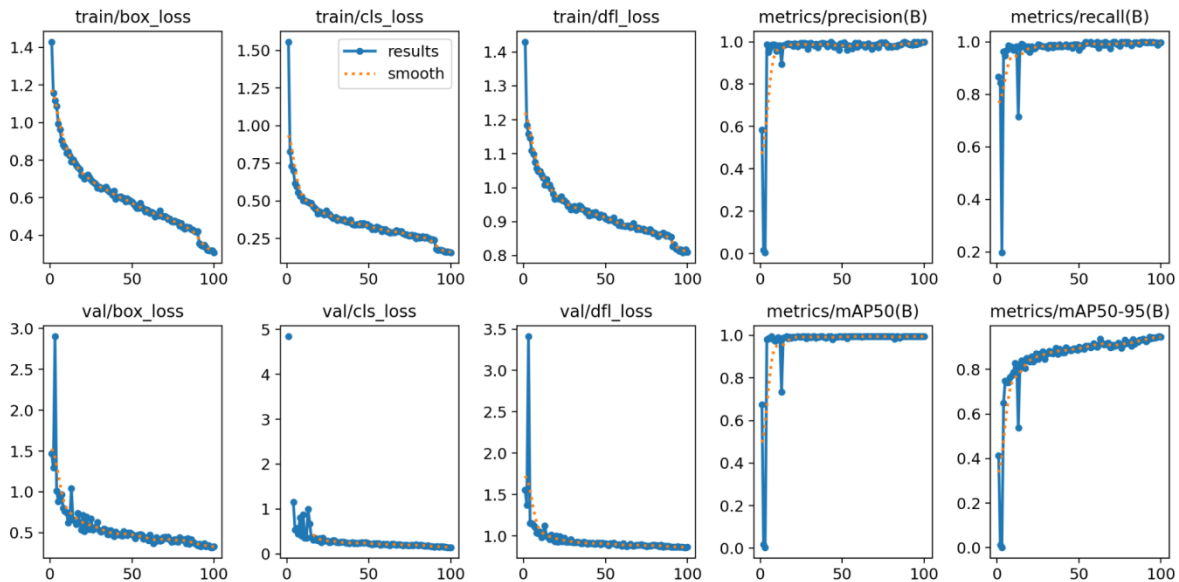


Figure 3.22 Training progress plots of Model 1

Comments:

- Loss Functions:
 - The values of train/box_loss, train/cls_loss, and train/df_l_loss steadily decrease over the epochs, indicating that the model is effectively learning from the training data and progressively minimizing prediction errors.
 - The validation loss values (val/box_loss, val/cls_loss, val/df_l_loss) also show a significant downward trend without any notable increase, suggesting that the model does not suffer from overfitting and maintains good generalization performance.
 - Although some minor fluctuations appear at the beginning of training (especially in val/cls_loss), this is expected due to random weight initialization and early model adjustments.
- Evaluation Metrics:
 - Precision (B) and Recall (B) quickly reach values close to 1.0 (99%) after around 10 epochs and remain stable, demonstrating the model's high accuracy and low false negative rate in detecting defective objects.
 - The mAP@0.5 (B) score reaches near-perfect values (~1.0) and stays consistent throughout the training process, indicating excellent performance in detecting defects at an IoU threshold of 0.5.
 - The mAP@0.5:0.95 (B) metric gradually increases and reaches approximately 0.88–0.90 by the end of training, showing that the model is capable of robust and precise localization under stricter IoU conditions, making it suitable for real-world defect detection tasks.

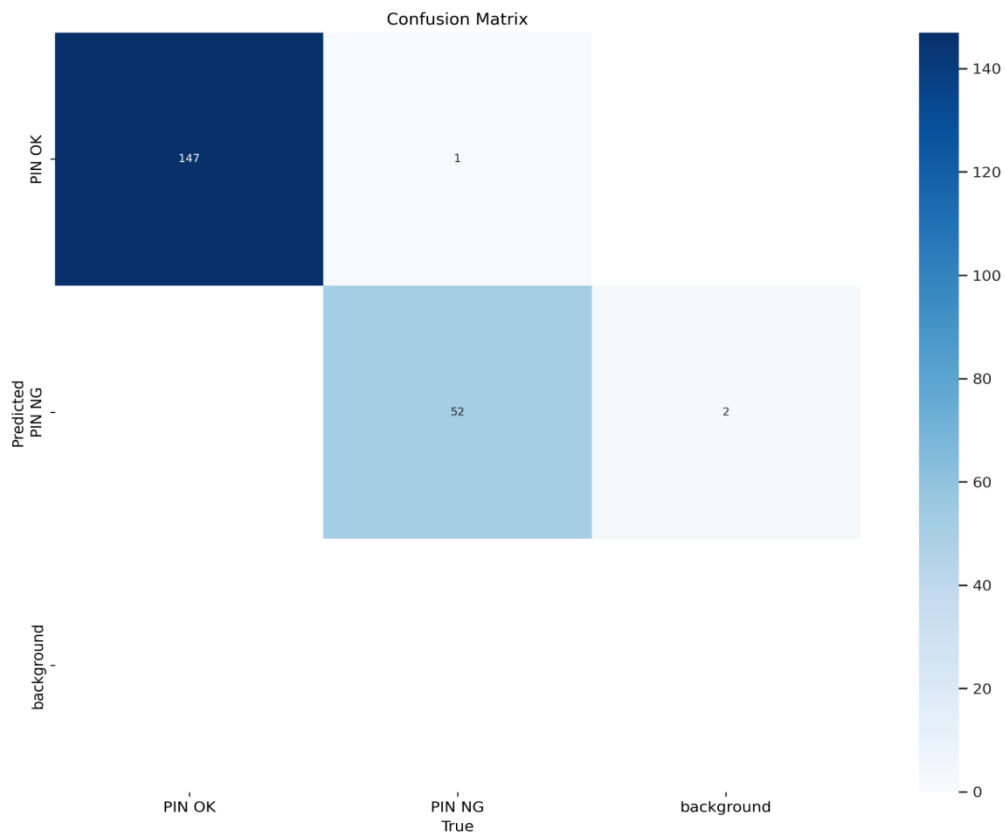


Figure 3. 23 Confusion Matrix of Model 1

Comments:

- High accuracy with OK labels: None of the 147 OK pin legs were mistakenly classified as NG, resulting in a 100% accuracy rate for the test image file.
- Among the 53 NG pin legs, 1 was misclassified as OK, accounting for only 1.89%.
- There were 2 objects that were not pin legs but were still detected. Although these could introduce some noise, the number is small and does not significantly affect the system, so it is acceptable.

In terms of performance, the processing time when testing the model is as follows:

Table 3. 7 The processing time when testing the model 1

	Max	Min
Preprocess	49.7 ms	2.3 ms
Inference	116.8 ms	50.5 ms
Postprocess	746.3 ms	1.6 ms

In which:

- Preprocess: This includes steps such as image resizing and data formatting before feeding it into the model.
- Inference: The time the model takes to perform inference to detect objects in the image.
- Postprocess: Handling the output results, including filtering bounding boxes, removing duplicates, and labeling the detected objects.

Comments:

- The processing time for the first image is much higher than for the subsequent images. This occurs due to system initialization, model loading, and memory setup.
- The following images reflect the actual performance of the system after stabilization.

3.4.3.2. Model 2

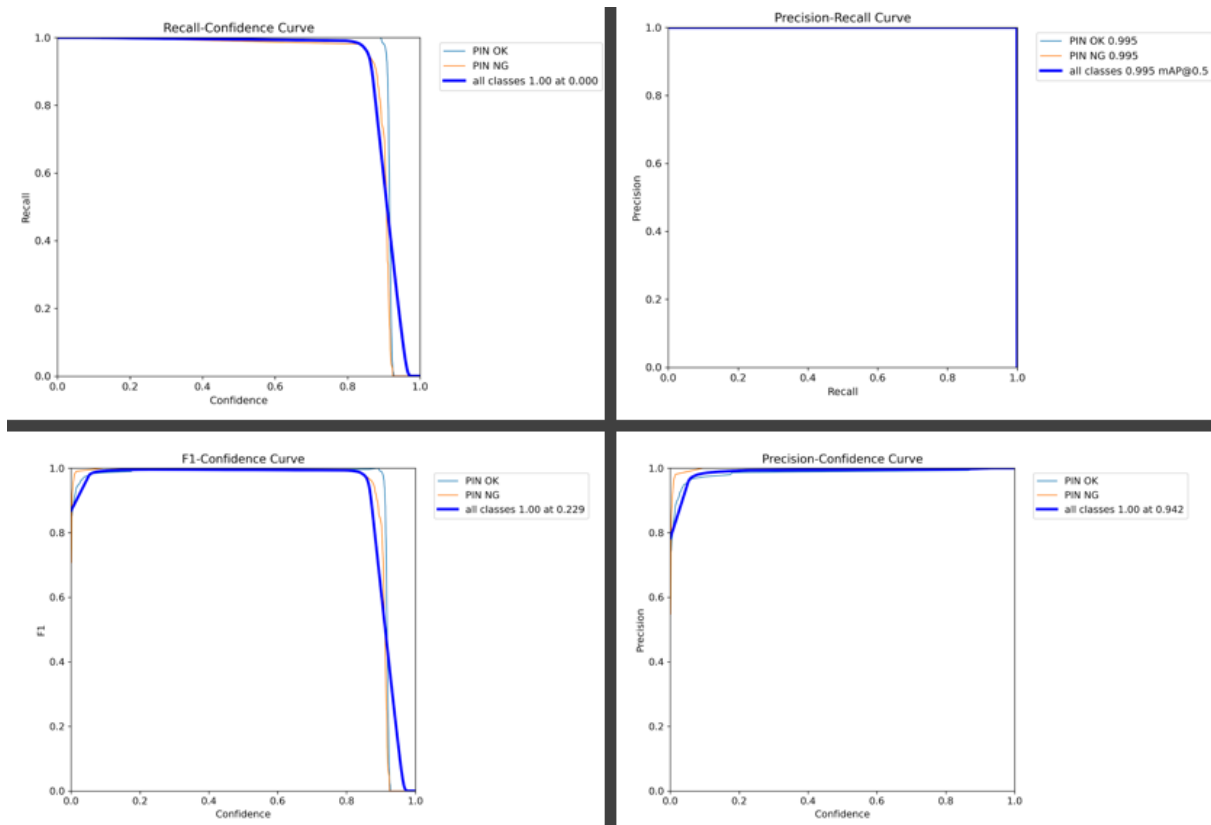


Figure 3. 24 Result charts of Model 1

Comments:

The charts indicate that the model appears to be well-balanced between precision and recall.

- The precision-recall curve shows a large area under the curve (mAP@0.5 = 0.995), indicating that the model performs effectively across various confidence thresholds.
- The F1-confidence curve reaches its peak (F1 = 1.00) at a relatively low confidence threshold of 0.229, suggesting that the model maintains a strong balance between precision and recall even when operating at lower thresholds.
- Meanwhile, the precision-confidence curve shows that the model achieves perfect precision (1.00) at a confidence of 0.942, which is ideal in applications that prioritize accuracy and minimal false positives.

- The recall-confidence curve confirms that the model sustains high recall (~1.00) across a broad range of confidence values, which means it is highly effective at identifying all relevant instances, including edge cases.
→ The classification model demonstrates high and stable performance, with a good balance between precision and recall. It exhibits excellent discriminative ability between the "PIN OK" and "PIN NG" classes.

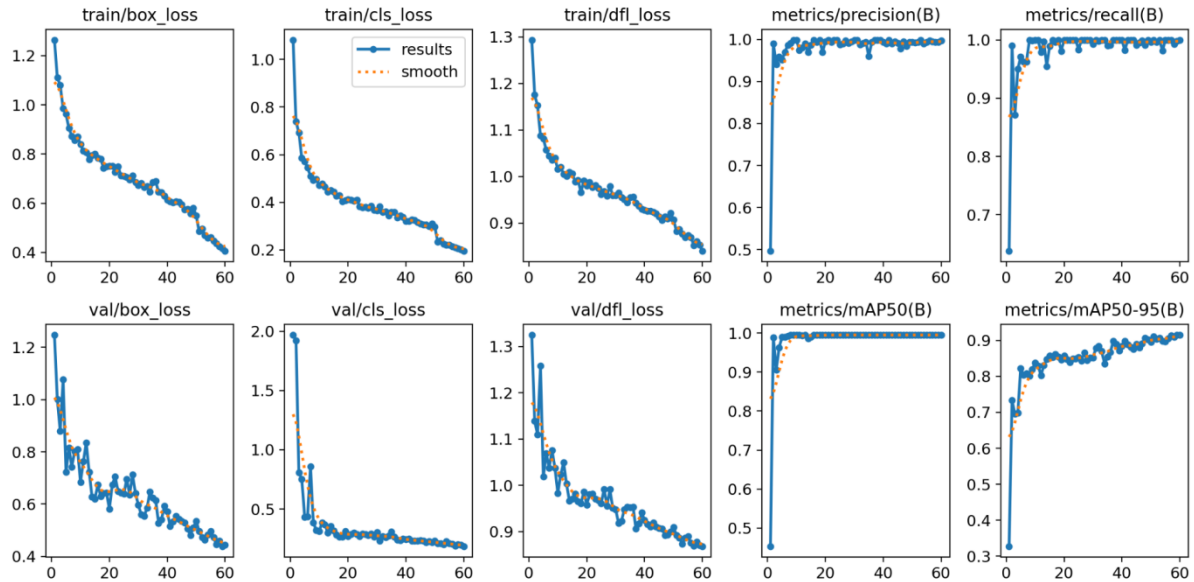


Figure 3. 25 Training progress plots of Model 2

Comments:

- Loss Functions:
 - The values of *train/box_loss*, *train/cls_loss*, and *train/dfl_loss* steadily decrease over the training epochs, indicating that the model is effectively learning meaningful information from the data.
 - The validation loss values (*val/box_loss*, *val/cls_loss*, *val/dfl_loss*) also show a consistent downward trend without significant increases, suggesting that the model is not experiencing overfitting.
 - Minor fluctuations in the validation losses at the beginning are normal and expected due to random initialization and early adjustments of the model's parameters.

- Evaluation Metrics:
 - Precision (B) and Recall (B) reach values close to 1.0 (99%), demonstrating the model's high accuracy in detecting defective objects and its low rate of missed detections.
 - mAP@0.5 (B) remains near 1.0 after approximately 20 epochs, indicating excellent performance in detecting objects at an IoU threshold of 0.5.
 - mAP@0.5:0.95 (B) steadily increases and reaches around 0.88–0.90 by the end of the training process, showing that the model performs reliably across stricter IoU thresholds and is capable of precise defect localization.

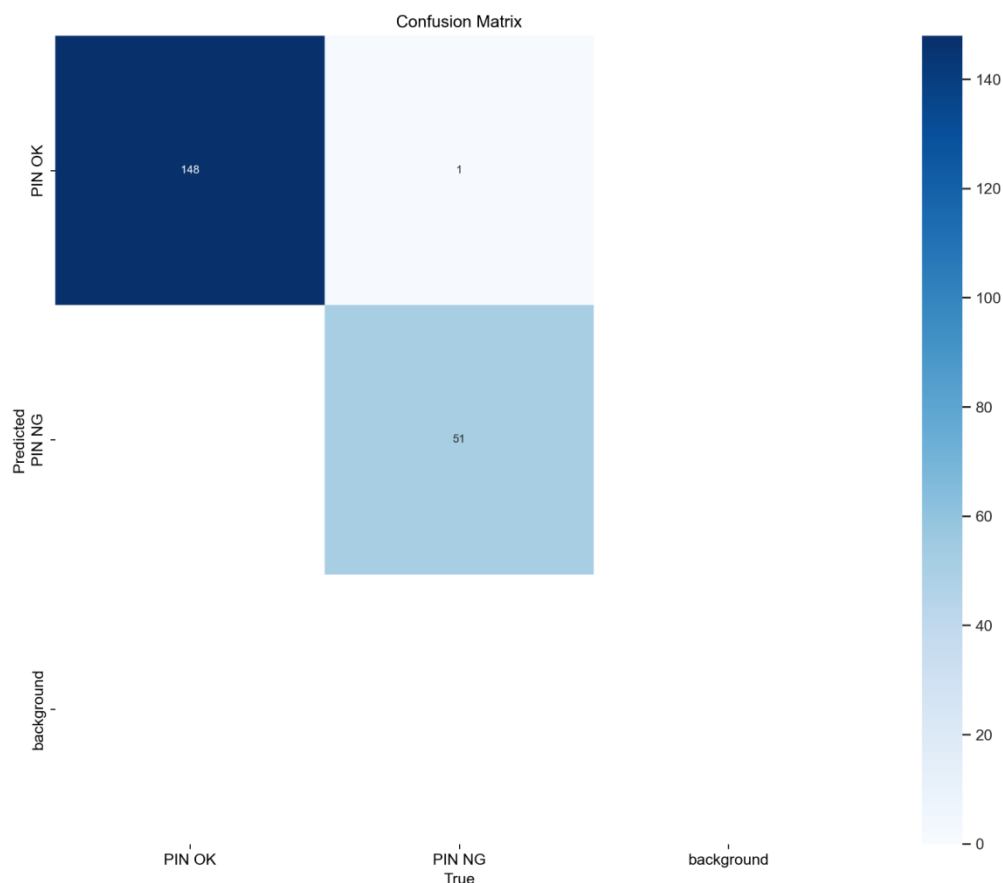


Figure 3. 26 Confusion Matrix of Model 2

Comments:

- High accuracy with OK labels: None of the 148 OK pin legs were mistakenly classified as NG, resulting in a 100% accuracy rate for the test image file.
- Among the 52 NG pin legs, 1 was misclassified as OK, accounting for only 1.9%.
- No non-pin objects were mistakenly detected.

In terms of performance, the processing time when testing the model is as follows:

Table 3. 8 The processing time when testing the model 2

	Max	Min
Preprocess	32.4 ms	2.4 ms
Inference	95.9 ms	50.8 ms
Postprocess	868.9 ms	1.8 ms

3.4.4. Continuous model improvement

The AI model training process for detecting solder joint defects was conducted through multiple iterative cycles of testing and continuous improvement to achieve high accuracy and practical applicability.

In the initial phase, the team trained the YOLOv8 model on the initial dataset collected from the production line. However, due to the limited and insufficiently diverse data, the results did not meet the accuracy requirements, and the defect detection rate remained low. The model was prone to overfitting on some samples or missing small defects because it had not been fully trained on a wide variety of real-world cases.

Recognizing these limitations, the team focused on expanding and improving the quality of the training dataset by:

- Collecting additional sample images from multiple production batches, ensuring the data covers a diverse range of solder joint defects and different lighting conditions and shooting angles.
- Carefully editing and supplementing data labels to ensure the accuracy of bounding boxes and clear classification between OK and NG.

After each data augmentation and tuning of training parameters (learning rate, batch size, number of epochs, etc.), the model was re-evaluated on a separate test set to measure performance. The results showed a clear improvement in accuracy, increased defect detection rates, and reduced prediction errors.

This process was repeated iteratively to optimize the model to best fit the specific characteristics of the real production environment. Continuous improvement of data and the model not only enhanced performance but also ensured the system could operate reliably and stably when deployed in practical applications.

3.5. Source code analysis

The system software is developed using the C# programming language on the Windows Forms (WinForms) platform with the .NET Framework 4.8. This is a powerful and widely used development environment in industrial applications due to its high compatibility with the Windows operating system, support for designing intuitive graphical user interfaces (GUI), and ease of integration with peripheral device control libraries such as cameras, PLCs, or network devices. The programming and compilation process is carried out in Visual Studio 2022, a professional integrated development environment (IDE) that enhances software development efficiency through debugging tools, source code management, and support for a wide range of libraries.

Meanwhile, the AI processing components are deployed independently on the Jetson Orin NX 16GB hardware platform, a specialized embedded device from NVIDIA capable of accelerating deep learning model processing with an integrated GPU. On the Jetson, the team uses the Python programming language to build the AI server system, where YOLOv8 (You Only Look Once version 8) — a modern and real-time optimized object detection model — is trained to identify various defects in industrial solder joints. The AI server uses FastAPI, a high-performance web framework that sets up RESTful APIs to receive images from the C# software, process them with the AI model, and return results in JSON format or as images annotated with bounding boxes.

❖ Configuration management and logging

The system uses the file “config.txt” to store important configuration parameters such as the IP address of the PLC, the image save path, and the address of the Python

server. This approach allows flexibility in modifying settings without the need to recompile the source code.

The logging function is implemented through a “console” method that displays real-time operation information on the user interface while simultaneously saving it to the “log.txt” file. This is highly useful for monitoring system status and debugging.

❖ **Industrial camera integration**

The control and image acquisition from the camera are handled through the MVSDK library provided by the manufacturer. Key aspects of this integration include:

- Initialization and Connection: The application automatically detects connected cameras and creates corresponding control objects (*CameraGrabber*, *CameraHandle*).
- Trigger Configuration: Cameras are configured in trigger mode, allowing image capture to be initiated via control signals from the PLC or manual commands.
- Image Processing and Saving: The *CameraGrabber_SaveImageComplete* function is registered as a callback to handle the image immediately after capture. Images are saved in JPEG format with a specified quality setting.
- Asynchronous Image Processing: An independent class, *AsyncSaveImage*, is implemented to manage image saving on separate threads. This prevents the user interface from freezing during the image writing process.

❖ **Communication with PLC**

The system uses the S7.Net library to establish connection and exchange data with the Siemens S7-1200 PLC.

- Connection: The function *Connect_PLC* is responsible for establishing the initial connection and updating the connection status on the user interface.
- Trigger signal reading: A timer is used to continuously monitor the trigger signal from the PLC. When the signal is received, the application initiates the image capture and subsequent processing.

- Result sending: The system is capable of sending inspection results (OK/NG) back to the PLC through specific bit addresses, enabling the PLC to make decisions based on the AI analysis results.

❖ **Communication with the AI server (Jetson)**

The system integrates an independent AI backend developed in Python using the TCP/IP Socket protocol.

- Server launch: The C# application can automatically start the AI Server Python script via a command-line process (cmd.exe)
- Data transmission: The *Predict_Python* function manages sending images from C# to the Python Server and receiving prediction results (including processed images and OK/NG status). Data is transmitted using a custom binary protocol that includes the size and content of the data.
- Asynchronous processing: The entire communication process with the AI Server is performed asynchronously (*async/await* and *Task.Run*), ensuring the user interface remains responsive throughout AI processing. Prediction results (images and OK/NG status) are displayed directly on the user interface.

❖ **Multithreading and asynchronous processing**

The use of C# features like *async/await* and *Task.Run* is a strong point of the source code. It allows the application to perform time-consuming tasks concurrently, such as capturing images from multiple cameras (*Trigger2Camera*) and network communication with the AI Server, without freezing the user interface. In particular, *Task.WhenAll* in *Trigger2Camera* enables parallel triggering of cameras, optimizing waiting time.

❖ **Libraries used in the program**

During the development of the control and image processing software, the team utilized various libraries and SDKs to ensure stable system operation and effective communication with devices such as cameras, PLCs, and the Jetson.

Table 3. 9 Libraries used in the program

Library	Function
System	Provides basic classes and data types.
System.Collections.Generic	Supports data structures like List< >, Dictionary< >, etc. → Used to store lists of cameras, images, AI results, etc.
System.Drawing	Handles images, bitmaps, colors, drawing, etc. → Displays images from the camera, draws bounding boxes from AI results.
System.Text	Advanced string processing (StringBuilder, encoding, etc). → Optimizes string concatenation and logging.
System.Windows.Form	Provides components for Windows Forms GUI. → Creates the user interface, displays images, buttons, and controls.
System.Runtime.InteropServices	Enables interaction with unmanaged code (DLLs written in C/C++). → Calls functions from the camera SDK (MVSDK) written in C/C++.
MVSDK	Main namespace of the MindVision camera SDK. → Accesses classes and functions for controlling and configuring the camera.
CameraHandle=System.Int32	Defines an alias CameraHandle for int. → Increases clarity when working with multiple cameras.
MvApi = MVSDK.MvApi	Shortens MVSDK.MvApi to MvApi.
System.IO	Supports file I/O operations. → Used to read/write images, configurations, log files, etc.

S7.Net	Library for communicating with Siemens S7-1200/1500 PLCs over Ethernet. → Receives trigger signals from the PLC and sends OK/NG results back.
System.Threading	Supports multithreading. → Enables concurrent tasks to keep the UI responsive.
System.Linq	Provides query capabilities for collections (filter, select, where, etc.). → Simplifies processing of image lists
System.Threading.Tasks	Supports asynchronous programming (async/await). → Sends images to Jetson and receives results without blocking the UI thread.

3.6. System Diagram

3.6.1. C# Diagram

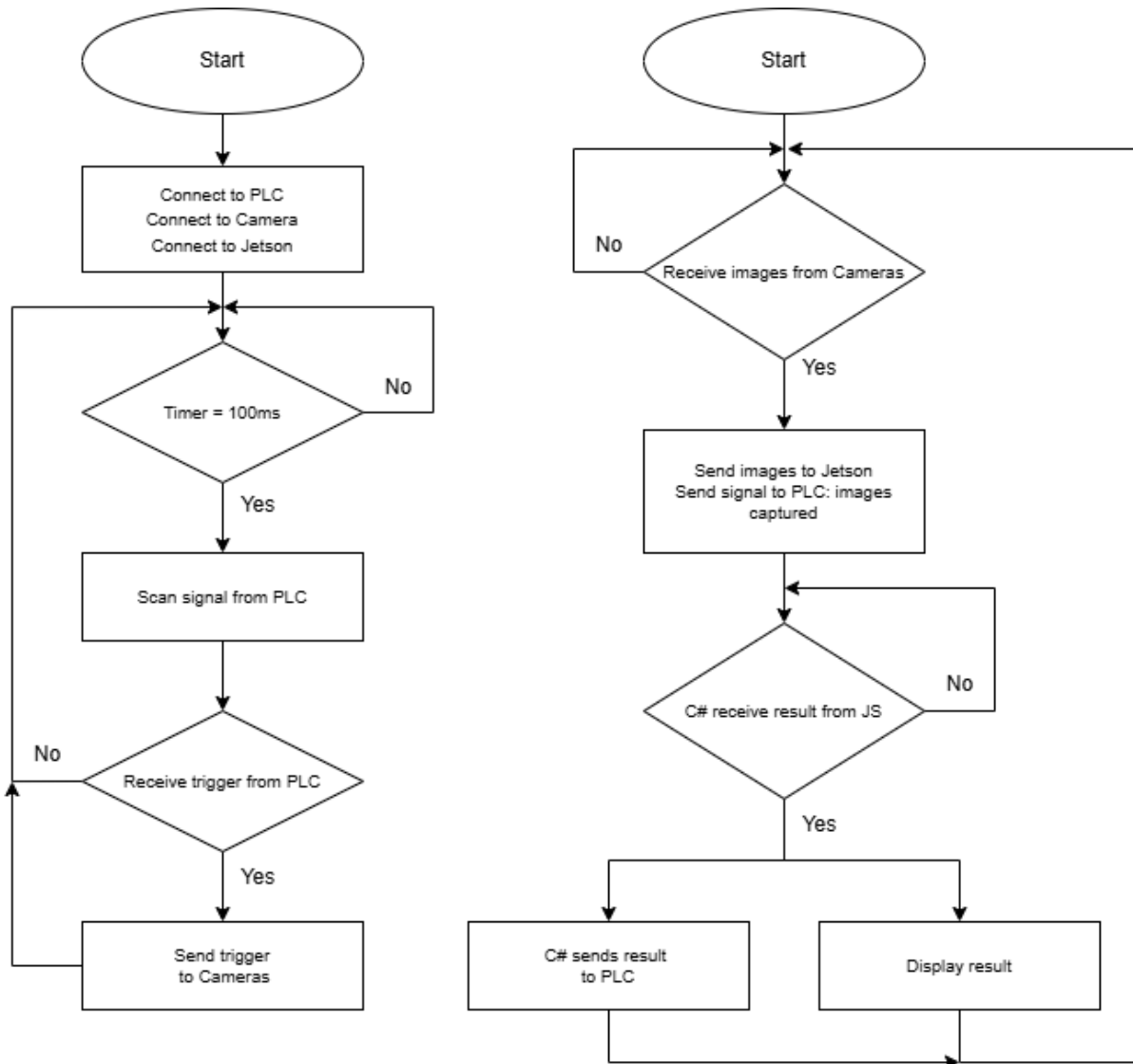


Figure 3. 27 C# Diagram

Algorithm Flow Description:

The algorithm flowchart above illustrates the overall operation of the system, including the communication between C#, the PLC, the two cameras, and the Jetson device. The PLC's operation is only described in general terms here; the detailed explanation will be presented in a later section.

At the start of the program, C# connects to the devices using their IP addresses and Serial Numbers. Once connected, C# activates a timer with an interval of 100 ms. Every 100 ms, C# scans the signal from the PLC. If C# receives a trigger signal from the PLC, it sends capture commands to both cameras. Once the two cameras have taken the images, they send them back to C#. After receiving the images, C# forwards them to the Jetson to run the AI model, while also sending a signal back to the PLC indicating that image capture is complete. When Jetson receives the images from C#, it runs the AI model and then sends the results back to C#. These results include the images, bounding boxes, and OK/NG labels of the products. C# receives the data and sends it to the PLC so that it can carry out the next tasks. At the same time, C# displays the results on the interface.

The program interface is designed according to the image below. It comprises two picture boxes and a control panel, which displays the connection status between devices.

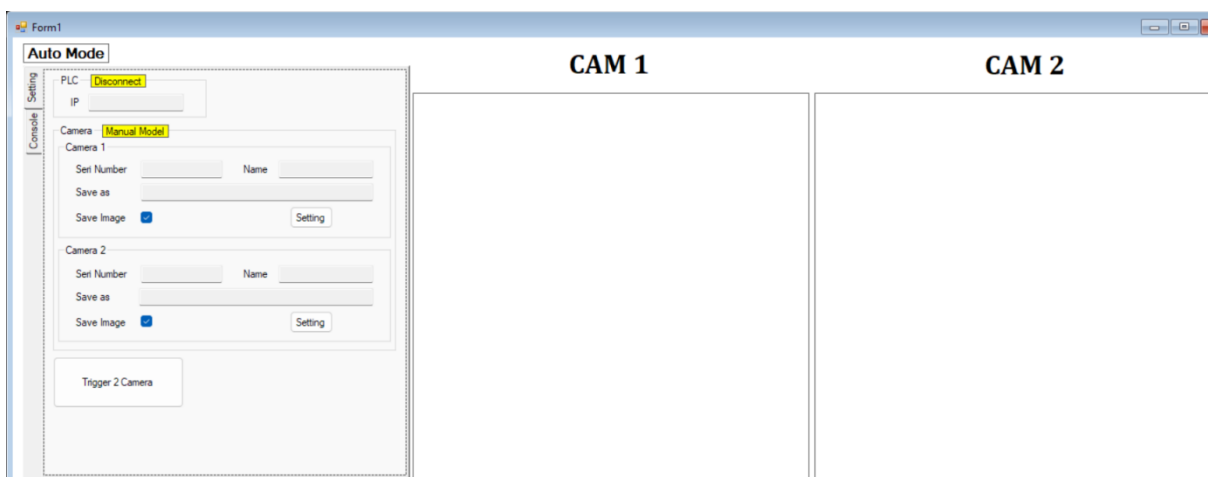


Figure 3. 28 Display interface of the program

3.6.2. Python Diagram

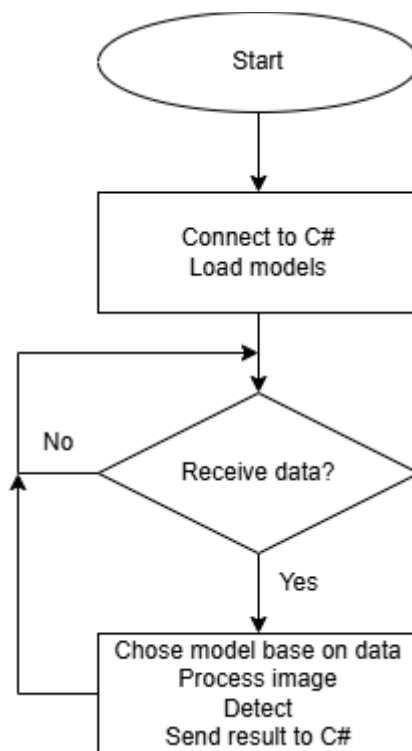


Figure 3. 29 Python Diagram

Algorithm Flow Description:

The algorithm begins by establishing a connection with the C# application and loading all required YOLO models. Once the connection is established, the server enters a continuous loop, waiting for incoming image data.

When data is received, the server selects the appropriate model based on the provided identifier, processes the image, runs detection, and sends the result (including the annotated image and OK/NG status) back to C#.

If no data is received, the server keeps listening until new data arrives.

3.6.3. PLC Diagram: Station 5: Camera

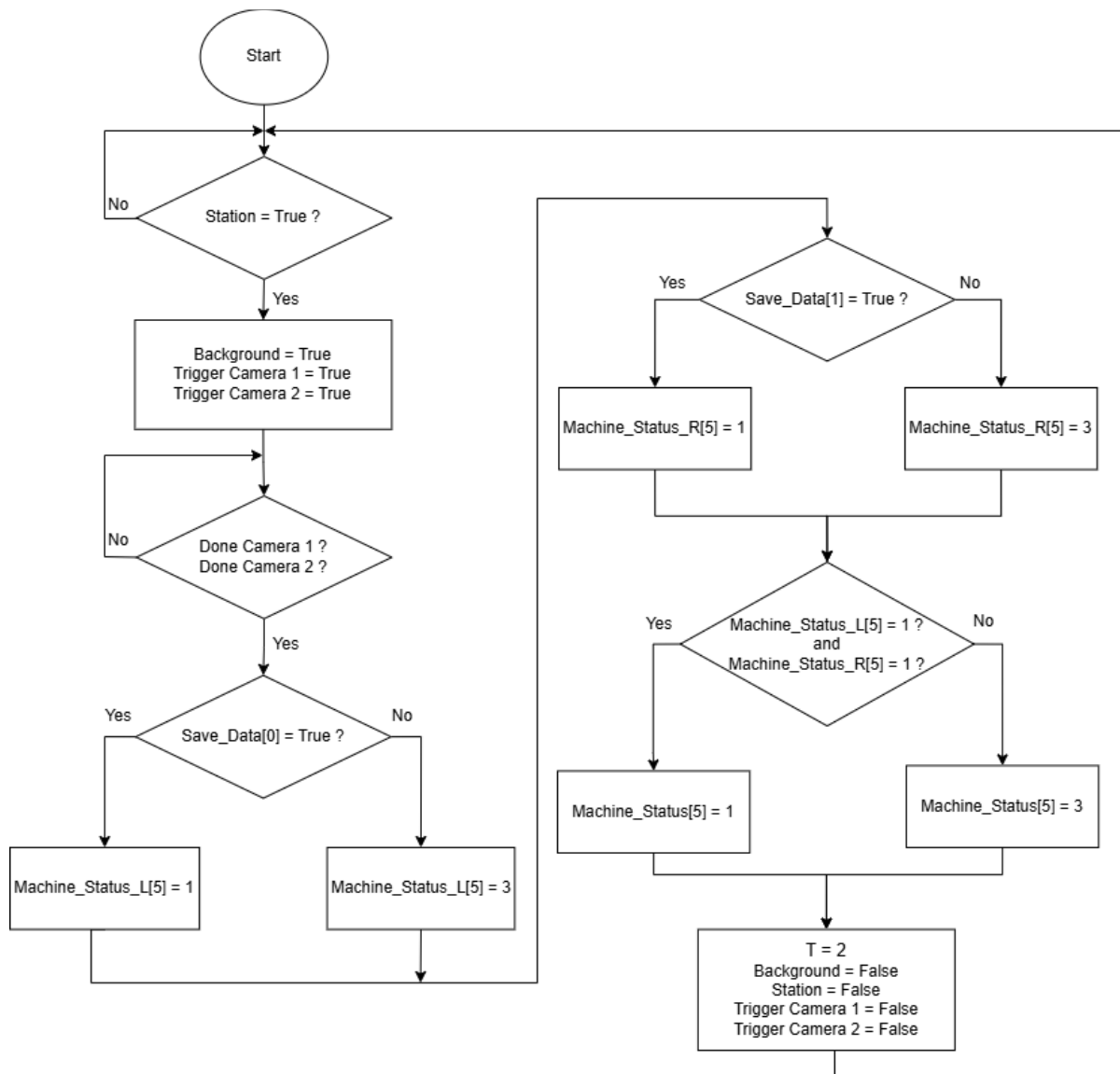


Figure 3. 30 PLC Diagram: Station 5: Camera

Algorithm Flow Description:

Figure 3.30 describes the tasks performed at Station 5: Camera.

First, both products must be in the correct position (Station = True) before any further actions are taken. Once the products are properly positioned, the program raises the background plate (Background = True) to separate the two products. Then, the PLC sends a trigger signal (to C#) to capture images of the products. After confirming that the images have been captured and results have been received from C# (Done Camera 1, Done Camera 2), the PLC stores the results in the Save_Data variable (Save_Data[0] is the result for the left pin, Save_Data[1] for the right pin). If a pin's result is OK, it sets Machine_Status_L[5] = 1 (for the left pin) or Machine_Status_R[5] = 1 (for the right pin); otherwise, it sets Machine_Status[5] = 3. Then a condition check will be performed: if both pin legs are equal to 1, then Machine_Status[5] = 1. otherwise, Machine_Status[5] = 3. After that, it waits for 2 seconds, then lowers the background plate. The two products are now out of the station, so Station = False and the trigger signal is turned off.

3.6.4. PLC Diagram: Station 6: Output

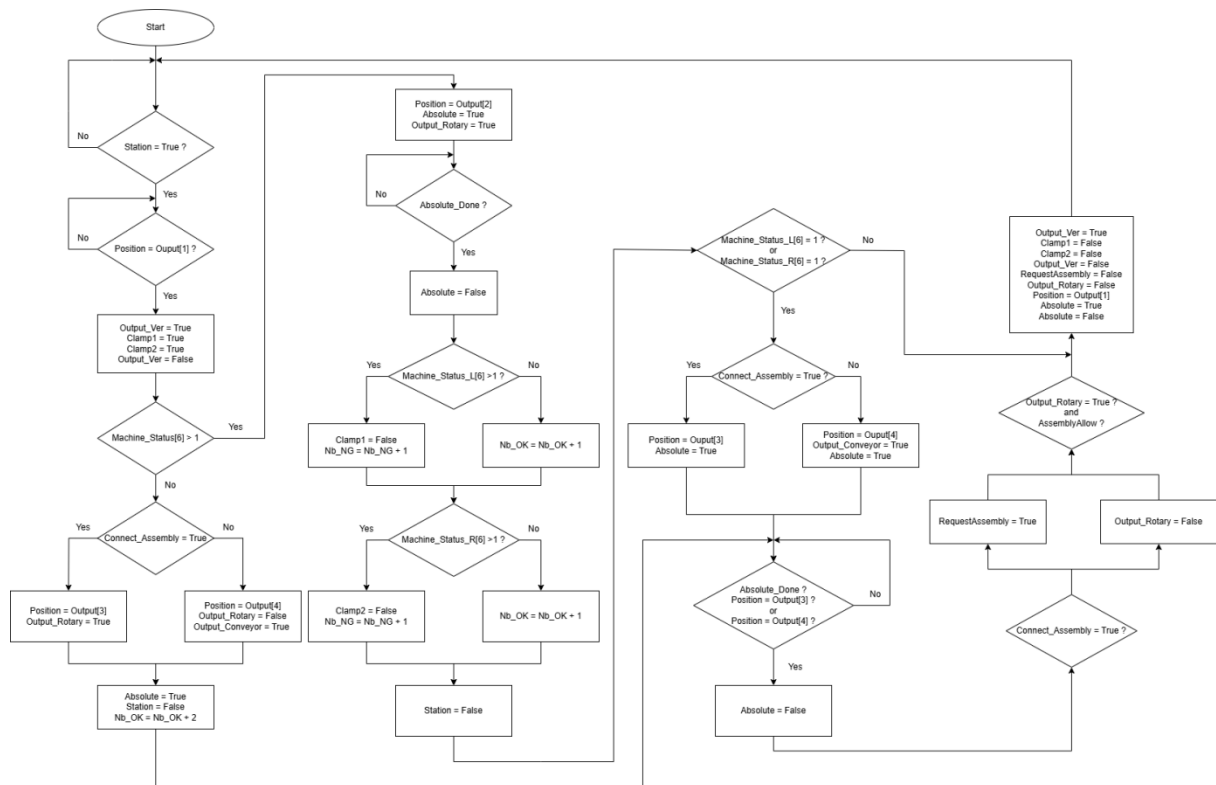


Figure 3. 31 PLC Diagram: Station 6: Output

Algorithm Flow Description:

Figure 3.31 describes the actions at Station 6: Output.

When the two products arrive at the correct position of the Output station (Station = True), the system checks whether the robotic arm is at position 1 (Position = Output[1], which is the initial position for picking up the products). If it is already at position 1, the PLC controls the robotic arm to move down (Output_Ver = True), and then the two grippers clamp the products (Clamp1 = True, Clamp2 = True). After clamping the products, the robotic arm moves back up to the initial position (Output_Ver = False).

The system then checks the status of both products. If either one or both products are NG (Machine_Status[6] > 1), the robotic arm will move to position 2 (Position = Output[2], where NG products are dropped into the scrap box). At this position, the status of each product is checked again. If the product in the left gripper is NG (Machine_Status_L[6] > 1), the left gripper releases the product into the scrap box (Clamp1 = False), and the NG product counter is increased by 1 (Number_NG = Number_NG + 1). Otherwise, the gripper remains closed and the OK product counter is increased by 1 (Number_OK = Number_OK + 1). The same steps apply to the right gripper.

If one of the products is OK (Machine_Status_L[6] = 1 or Machine_Status_R[6] = 1), the system checks the connection to the Assembly Housing machine. If the connection is available (Connect_Assembly = True), the robotic arm moves to position 3 (Position = Output[3], which is the drop-off point for the conveyor leading to the Assembly Housing machine). Otherwise, the robotic arm moves to position 4 (Position = Output[4], which is the drop-off point for the secondary conveyor). The robotic arm moves to a similar position if both products are OK (Machine_Status[6] = 1), and in this case, the OK product counter is increased by 2 (Number_OK = Number_OK + 2).

If a connection and request to the Assembly Housing machine is successful, the gripper rotates 90 degrees so the product is properly aligned before being placed onto the conveyor. If there is no connection, the gripper does not rotate. After that, the robotic arm moves down, releases both grippers, and then moves back up. Finally, the robotic arm returns to the initial position and waits for the next products to arrive at the station.

CHAPTER 4: CONCLUSION

4.1. Results

After completing the implementation and integration of the control software, cameras, and AI model, we conducted testing on various pairs of products to evaluate the performance of the system. Additionally, we performed repeated inspections on the same product pairs to assess consistency. However, the results remained stable with no variation observed. Therefore, a comparative performance table for repeated trials is not included. The table below presents the data for 10 pairs of products among those tested.

Table 4. 1 System performance evaluation

Pair of products	Total processing time (ms)	Display time (s)	Precision (%)
1	167.4	14.21	100
2	172.7	17.74	75
3	169.2	11.32	100
4	165.8	14.17	100
5	171.7	14.52	100
6	167.6	13.49	100
7	169.1	15.02	100
8	167.5	14.93	100
9	170.2	13.8	100
10	168.9	15.78	100
Average	170.01	14.85	97.5

Comments:

- Performance overview:
 - Total number of product pairs tested: 10 pairs.
 - Average precision: 97.5%
 - The AI model operates consistently and achieves relatively high accuracy in most cases.
- Processing time:
 - Average processing time: Approximately 169.01 milliseconds, with low variation.
 - Average display time: Approximately 14.498 seconds. This value may be high due to transmission latency but is acceptable as it does not affect system performance.

The user interface is designed to be intuitive and user-friendly, enabling operators to easily monitor and evaluate solder joint conditions in real time.

The main interface includes the following key components:

- Image display frames: Two main image frames (PictureBoxes) are arranged to show images captured from the two industrial cameras. These images are continuously updated based on trigger signals from the PLC.
- AI prediction results: On the displayed images, suspected defect locations are marked with bounding boxes in different colors, along with classification labels (OK/NG or specific defect names). This helps operators quickly identify and analyze the results.
- System status information: Status bars or message boxes display the PLC connection status, camera status, and data reception status from the AI server.
- Summary results: The software also shows a real-time summary of the number of products inspected, the counts of OK and NG products, supporting production management purposes.

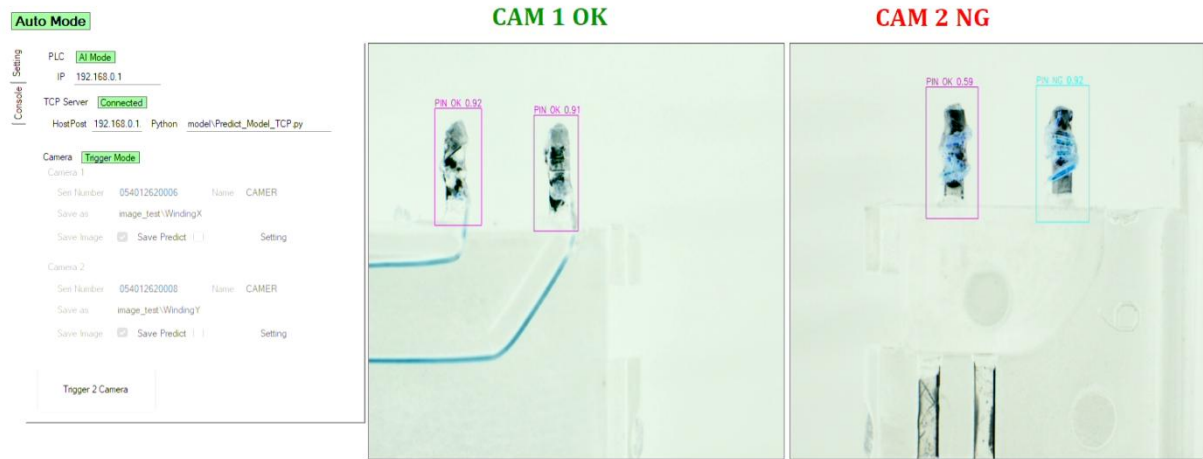


Figure 4. 1 Interface after displaying the program results

The system has been deployed and directly applied on the production line, enabling automatic and continuous solder joint quality inspection. During actual operation, the AI model's processing speed is somewhat slower compared to the independent testing results on the model. The main cause of this difference stems from the real operating environment, which is affected by various factors such as more limited hardware resources compared to the training environment, network load during data transmission, as well as the complex interactions between hardware and software components within the system. However, this delay is minimal and does not significantly impact the overall system performance under real production conditions.

```
Speed: 39.8ms preprocess, 123.9ms inference, 3.7ms postprocess per image at shape (1, 3, 480, 640)
2025-05-21 14:10:04 Status: True,model0
2025-05-21 14:10:04 Status: False,model1
```

Figure 4. 2 Actual processing time when applied to the system

Additionally, data exchange between devices—such as industrial cameras, the control software on the PC, and the AI server running on the Jetson Orin NX—also adds a slight latency before prediction results are displayed on the user interface. This time varies depending on local network conditions, the number of images to process, and the system's current workload. Typically, the delay is a few seconds, which remains within an acceptable range for automatic inspection operations on the production line.

Importantly, the user interface only serves to display the final inspection results and does not participate in direct control of devices or the production process. Therefore, delays in updating images and results on the interface do not interrupt or affect the actual

operation of the production line. The system continues to maintain continuity, stability, and reliability during operation, fully meeting the requirements for solder joint quality inspection without causing unexpected failures or interruptions.

Overall, despite some inherent latency during real deployment, the system has proven to be feasible, effective, and highly applicable in industrial production environments, contributing to improved inspection quality and reduced errors compared to traditional manual methods.

4.2. Problems

During the implementation of the project, the team encountered several significant challenges related to time constraints, technical experience, and real-world deployment conditions at the factory.

- Specifically, the total duration allocated for the project was approximately three months, covering all phases: theoretical research, familiarization with hardware devices, development of control software, data collection and labeling for training, AI model training, testing, and real-world deployment. For a complex system like automated soldering inspection using AI combined with PLCs and industrial cameras, this timeline was quite limited. It required the team to plan clearly, allocate time effectively for each phase, and work continuously to stay on schedule.
- Regarding the AI model, the initial training attempts did not produce the desired results. This was primarily due to a limited training dataset, which caused the model to overfit or miss defects in real production scenarios. The team had to repeatedly adjust by collecting more data, applying data augmentation, and improving the labeling process to enhance the quality of the input dataset. Re-training the model also demanded significant time and computing power, creating further challenges for continuous testing.
- One of the most significant challenges was the deployment and testing of the system on the actual production line. Since the production line operates continuously, stopping the machines for integration, adjustments, and testing was rarely allowed. The team only had a few short time slots during

shift breaks or scheduled maintenance to conduct tests. This added pressure to the testing process, as even a small configuration or software error could render an entire test session unsuccessful. Additionally, the inability to test continuously under various operational conditions limited the ability to comprehensively evaluate the system's performance in real-world environments.

4.3. Future works

Within the scope of the current project, the AI-based automated solder inspection system has so far focused solely on detecting one main type of defect: insufficient solder – a common issue that directly impacts product quality on the production line. However, in industrial classification, soldering defects are generally divided into two main groups: solder ball defects and insufficient solder defects.

The decision to address only one defect type in the initial stage was made to maintain focus, simplify the model training process, and shorten the time required for system prototyping and deployment. Nevertheless, in order to enhance inspection effectiveness and meet future quality control requirements, it is essential to expand the model's capabilities to detect both defect types simultaneously.

The next development steps for the system will include:

- Collecting additional image data related to solder ball defects, ensuring diversity in terms of shooting conditions, lighting, and defect positioning.
- Accurately labeling data for both defect types, which will serve as the foundation for multi-class defect detection model training.
- Improving the AI model architecture by transitioning from a simple binary classification model to a multi-class object detection model with better differentiation capabilities.
- Retraining the model with increased complexity, requiring optimization in terms of algorithms, processing resources, and training time.
- Testing and evaluating the model on the actual production line to ensure it can detect multiple types of defects simultaneously with appropriate speed and accuracy.

In addition to expanding the AI model, the system may also be upgraded in other areas such as improving the user interface, optimizing data transmission speed between devices, or implementing an automated error feedback mechanism to achieve tighter integration with the plant's control system. These are important directions toward completing an intelligent solder inspection system that can be effectively applied in real-world industrial production environments.

REFERENCE

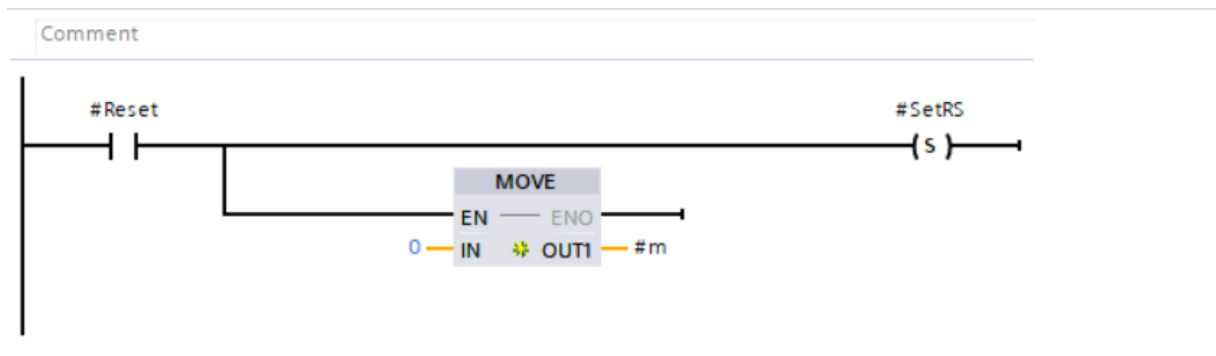
- [1] Confidential documents of PREMO VIETNAM. Allowed and approved to use for graduation thesis purpose by PREMO VIETNAM.
- [2] MANG TT CN_Lect_LQHuy_C2-Co so ky thuat_1-2015
- [3] MANG TT CN_Lect_LQHuy_C1-Khai niem ve TT va mang TTCN_1-2015_Handouts
- [4] Giáo Trình Hệ Thống Truyền Động Thủy Lực Và Khí Nén (Trần Ngọc Hải Cb Trần Xuân Tuyền)
- [5] Duong Thi Hien Vi, Dr. Nguyen Le Hoa, Eng, Nguyen Trong Tai, “ Develop an automated soldering inspection system using image processing technique with NI Vision Builder AI”, 2023.
- [6] Électronique_ NXB Giáo Dục_ Biên dịch: Văn Thế Vinh
- [7] (PDF) Aircraft Surface Defect Inspection System Using AI with UAVs
- [8] Aetina_AIE-KO22_32_KN32_42_datasheet
- [9] TL-SG1008MP_1.0
- [10] M2514-MP2+spec
- [11] HT-GE502C-T1-C Datasheet-EN_V24.11.18
- [12] JETSON-ORIN-NX-16G-DEV-KIT - Waveshare Wiki
- [13] AIE-KO22/32 | AIE-KN32/42
- [14] Huateng Vision
- [15] Cylinder MGPM12-40Z
- [16] Ultralytics YOLOv8 - NVIDIA Jetson AI Lab
- [17] Fast API Tutorial
- [18] S7.Net documentation
- [19] TCP/IP
- [20] grupopremo.com
- [21] Models Supported by Ultralytics
- [22] Tính toán, lựa chọn Ống kính (Lens) cho Camera công nghiệp | MC&TT Co., Ltd

LIST OF SYMBOLS AND ABBREVIATION

1. **AI** : Artificial Intelligence
2. **PLC** : Programmable Logic Controller
3. **NG** : Not Good
4. **C#** : Csharp
5. **GigE** : Gigabit Ethernet
6. **USB** : Universal Serial Bus
7. **LAN** : Local Area Network
8. **PoE** : Power Over Ethernet
9. **NVIDIA** : Next Generation Visual Interface Device
10. **GPU** : Graphics Processing Unit
11. **RAM** : Random Access Memory
12. **SDK** : Software Development Kit
13. **YOLO** : You Only Live Once
14. **I/O** : Input/Output
15. **CPU** : Central Processing Unit
16. **TCP/IP** : Transmission Control Protocol / Internet Protocol
17. **API** : Application Programming Interface
18. **HTTP** : HyperText Transfer Protocol
19. **JSON** : JavaScript Object Notation
20. **cls_loss** : Classification Loss
21. **dfl_loss** : Distribution Focal Loss
22. **GUI** : Graphical User Interfaces

APPENDIX

❖ PLC



● Collecting data

Reset

```
1 IF #SetRS THEN
2
3 CASE #m OF
4
5     0:
6         #O_Camera_UpDown := FALSE;
7         "F_Go_Camera_Station" := FALSE;
8         #n := 0;
9         #m := 0;
10        #SetRS := FALSE;
11
12 END_CASE;
13
14 END_IF;
```

Auto

```
1
2 IF #Auto AND #Enable THEN
3   CASE #n OF
4
5     0:
6       IF "F_Go_Camera_Station" AND #I_Camera_UpDown_Rep THEN
7         #n := 1;
8       END_IF;
9
10    1:
11      #O_Camera_UpDown := TRUE;
12      IF #I_Camera_UpDown_Act THEN
13        #n := 2;
14      END_IF;
15
16    2:
17      "Vision_AI".Trigger[0] := '1';
18      "Vision_AI".Trigger[1] := '2';
19      #n := 3;
20
21    3:
22      IF "IEC_Timer_Camera".Q THEN
23        #O_Camera_UpDown := FALSE;
24        #n := 4;
25      END_IF;
26
27    4:
28      IF #I_Camera_UpDown_Rep THEN
29        #n := 5;
30      END_IF;
```

- Station 5: Camera

Reset

```
1 IF #SetRS THEN
2
3   CASE #m OF
4
5     0:
6       #O_Camera_UpDown := FALSE;
7       "F_Go_Camera_Station" := FALSE;
8       #n := 0;
9       #m := 0;
10      #SetRS := FALSE;
11
12   END_CASE;
13
14 END_IF;
```

Auto

```

1
2 IF #Auto AND #Enable THEN
3     CASE #n OF
4
5         0:
6             IF "F_Go_Camera_Station" AND #I_Camera_UpDown_Rep THEN
7                 #n := 1;
8             END_IF;
9
10        1:
11            #O_Camera_UpDown := TRUE;
12            IF #I_Camera_UpDown_Act THEN
13                #n := 2;
14            END_IF;
15
16        2:
17            "Vision_AI".Trigger[0] := '1';
18            "Vision_AI".Trigger[1] := '2';
19            #n := 3;
20
21        3:
22            IF ("Done_cam1" AND "Done_cam2") THEN
23                IF "Vision_AI".Save_Data[0] THEN // RESULT WAIT CAMERA1
24                    "Machine_Status_DB".Machine_Status_L[5] := 1; // OK CameraL
25                ELSE
26                    "Machine_Status_DB".Machine_Status_L[5] := 3; // NG CameraL
27                END_IF;
28
29                IF "Vision_AI".Save_Data[1] THEN // RESULT WAIT
30                    "Machine_Status_DB".Machine_Status_R[5] := 1; // OK CameraR
31                ELSE
32                    "Machine_Status_DB".Machine_Status_R[5] := 3; // NG CameraR
33                END_IF;
34                #n := 4;
35            END_IF;
36
37        4:
38            "Vision_AI".Save_Data[0] := FALSE;
39            "Vision_AI".Save_Data[1] := FALSE;
40            IF "Machine_Status_DB".Machine_Status_R[5] = 1
41                AND "Machine_Status_DB".Machine_Status_L[5] = 1 THEN
42                "Machine_Status_DB".Machine_Status[5] := 1; // OK Camera
43            ELSE
44                "Machine_Status_DB".Machine_Status[5] := 3; // NG Camera
45            END_IF;
46            #n := 5;
47
48        5:
49            IF "IEC_Timer_Camera".Q THEN
50                #O_Camera_UpDown := FALSE;
51                #n := 6;
52            END_IF;

```

```
52      6:
53      IF #I_Camera_UpDown_Rep THEN
54          #n := 7;
55      END_IF;
56      7:
57          "F_Go_Camera_Station" := FALSE;
58          #n := 0;
59
60      END_CASE;
61  END_IF;
```

Station 6: Output

Reset

```
1  IF #SetRS THEN
2  CASE #m OF
3
4      0:
5          #ResquestAssembly := FALSE;
6          "Motion_Control_DB".Exe_Home := FALSE;
7          "Motion_Control_DB".Exe_Absolute := FALSE;
8          #Busy := FALSE;
9
10         "F_Go_Output_Station" := FALSE;
11         #O_Output_Ver := FALSE;
12
13         #n := 0;
14         #m := 1;
15     1:
16
17     IF #I_Output_Ver_Rep THEN
18         #O_Output_Rotary := FALSE;
19         #O_Output_Clamp1 := FALSE;
20         #O_Output_Clamp2 := FALSE;
21         "Motion_Control_DB".Exe_Home := TRUE;
22         #m := 2;
23     END_IF;
24     2:
25     IF "Motion_Control_DB".Home_Done THEN
26         "Motion_Control_DB".Exe_Home := FALSE;
```

```

27      #m := 3;
28      END_IF;
29      3:
30
31      "Motion_Control_DB".Position := "Position_Axis".Output[1];
32      "Motion_Control_DB".Velocity := "Position_Axis".Speed;
33      "Motion_Control_DB".Exe_Absolute := TRUE;
34      #m := 4;
35      4:
36      IF "Motion_Control_DB".MoveAbsolute_Done THEN
37          "Motion_Control_DB".Exe_Absolute := FALSE;
38          #m := 5;
39      END_IF;
40
41      5:
42      IF "Motion_Control_DB".Position = "Position_Axis".Output[1]
43          AND #I_Output_Clamp1_Rep AND #I_Output_Clamp2_Rep
44          AND #I_Output_Rotary_Rep AND #I_Output_Ver_Rep THEN
45          "Manual".O_Output_Rotary := FALSE;
46          "Manual".O_Output_Ver := FALSE;
47          "Manual".O_Output_Clamp1 := FALSE;
48          "Manual".O_Output_Conveyor := FALSE;
49          #SetRS := FALSE;
50          #m := 0;
51      END_IF;
52
53      END_CASE;
54
55      END_IF;

```

Auto

```

1
2  IF #Auto AND #Enable THEN
3      CASE #n OF
4
5          0:
6          IF "F_Go_Output_Station"
7              AND #I_Output_Clamp1_Rep AND #I_Output_Clamp2_Rep
8              AND #I_Output_Rotary_Rep AND #I_Output_Ver_Rep
9              AND "Motion_Control_DB".Position = "Position_Axis".Output[1] THEN
10              #O_Output_Ver := TRUE;
11              #n := 1;
12          END_IF;
13
14          1:
15          IF #I_Output_Ver_Act THEN
16              #O_Output_Clamp1 := TRUE;
17              #O_Output_Clamp2 := TRUE;
18              #n := 2;
19          END_IF;
20
21          2:
22          IF #I_Output_Clamp1_Act AND #I_Output_Clamp2_Act THEN
23              #O_Output_Ver := FALSE;
24              #n := 3;
25          END_IF;
26
27          3:
28          IF #I_Output_Ver_Rep THEN
29              IF "Machine_Status_DB".Machine_Status[6] > 1

```

```
30 OR "Machine_Status_DB".Machine_Status[6] = 0 THEN
31 "Motion_Control_DB".Position := "Position_Axis".Output[2]; // NG Part
32 "Motion_Control_DB".Exe_Absolute := TRUE;
33 #O_Output_Rotary := TRUE; // Done Station
34 #n := 4;
35 END_IF;
36 END_IF;
37
38 IF #I_Output_Ver_Rep THEN
39 IF "Machine_Status_DB".Machine_Status[6] = 1 THEN
40 IF "Enable_Station_DB".F_Enable_Connect_Assembly THEN
41 "Motion_Control_DB".Position := "Position_Axis".Output[3]; // OK Part
42 #O_Output_Rotary := TRUE;
43 ELSE
44 "Motion_Control_DB".Position := "Position_Axis".Output[4]; // OK Part
45 #O_Output_Rotary := FALSE;
46 #O_Output_Conveyor := TRUE;
47 END_IF;
48 "Motion_Control_DB".Exe_Absolute := TRUE;
49 "F_Go_Output_Station" := FALSE;
50 "Delay".Number_OK := "Delay".Number_OK + 2;
51 #n := 8; // Done Station
52 END_IF;
53 END_IF;
54
55 4:
56 IF #I_Output_Rotary_Act
57 AND "Motion_Control_DB".MoveAbsolute_Done
58
59 AND "Motion_Control_DB".Position = "Position_Axis".Output[2] THEN
60 "Motion_Control_DB".Exe_Absolute := FALSE;
61 #n := 5;
62 END_IF;
63
64 5:
65 IF "Machine_Status_DB".Machine_Status_L[6] > 1 OR "Machine_Status_DB".Machine_Status_L[6] = 0 THEN
66 #O_Output_Clamp1 := FALSE;
67 "Delay".Number_NG := "Delay".Number_NG + 1;
68 ELSE
69 "Delay".Number_OK := "Delay".Number_OK + 1;
70 END_IF;
71
72 IF "Machine_Status_DB".Machine_Status_R[6] > 1 OR "Machine_Status_DB".Machine_Status_R[6] = 0 THEN
73 #O_Output_Clamp2 := FALSE;
74 "Delay".Number_NG := "Delay".Number_NG + 1;
75 ELSE
76 "Delay".Number_OK := "Delay".Number_OK + 1;
77 END_IF;
78 "F_Go_Output_Station" := FALSE; // Done Station
79 #n := 6;
80
81 6:
82 IF NOT #O_Output_Clamp1 OR NOT #O_Output_Clamp2 THEN
83 #n := 7;
84 END IF;
```

```

84 | 7:
85 | IF "Machine_Status_DB".Machine_Status_L[6] = 1 OR "Machine_Status_DB".Machine_Status_R[6] = 1 THEN
86 | IF "Enable_Station_DB".F_Enable_Connect_Assembly THEN
87 |     "Motion_Control_DB".Position := "Position_Axis".Output[3]; // OK Part
88 | ELSE
89 |     "Motion_Control_DB".Position := "Position_Axis".Output[4]; // OK Conveyor
90 |     #O_Output_Conveyor := TRUE;
91 | END_IF;
92 | "Motion_Control_DB".Exe_Absolute := TRUE;
93 | #n := 8;
94 | ELSIF "Machine_Status_DB".Machine_Status_L[6] > 1 AND "Machine_Status_DB".Machine_Status_R[6] > 1 THEN
95 |     #n := 12;
96 | END_IF;
97 | 8:
98 | IF "Motion_Control_DB".MoveAbsolute_Done
99 |     AND ("Motion_Control_DB".Position = "Position_Axis".Output[3]
100 |     OR "Motion_Control_DB".Position = "Position_Axis".Output[4]) THEN
101 |     "Motion_Control_DB".Exe_Absolute := FALSE;
102 |     #n := 9;
103 | END_IF;
104 | 9:
105 | IF "Enable_Station_DB".F_Enable_Connect_Assembly THEN
106 |     #ResquestAssembly := TRUE;
107 |     IF #I_Output_Rotary_Act THEN
108 |         #n := 10;
109 |     END_IF;
110 | ELSE
111 |     #O_Output_Rotary := FALSE;
112 |     IF #I_Output_Rotary_Rep THEN
113 |         #n := 11;
114 |     END_IF;
115 | END_IF;
116 | 10 :
117 | IF #AssemblyAllow THEN
118 |     #n := 11;
119 | END_IF;
120 | 11:
121 | #O_Output_Ver := TRUE;
122 | IF #I_Output_Ver_Act THEN
123 |     #n := 12;
124 | END_IF;
125 | 12:
126 | #O_Output_Clamp1 := FALSE;
127 | #O_Output_Clamp2 := FALSE;
128 | IF #I_Output_Clamp1_Rep AND #I_Output_Clamp2_Rep THEN
129 |     #O_Output_Ver := FALSE;
130 |     #n := 13;
131 | END_IF;
132 | 13:
133 | IF #I_Output_Ver_Rep THEN
134 |     #ResquestAssembly := FALSE;
135 |     #n := 14;
136 | END_IF;
137 | ..
138 | 14:
139 | IF NOT #ResquestAssembly THEN
140 |     #O_Output_Rotary := FALSE;
141 |     "Motion_Control_DB".Position := "Position_Axis".Output[1];
142 |     "Motion_Control_DB".Exe_Absolute := TRUE;
143 |     #n := 15;
144 | END_IF;
145 | 15:
146 | IF "Motion_Control_DB".MoveAbsolute_Done
147 |     AND #I_Output_Rotary_Rep AND #I_Output_Ver_Rep
148 |     AND "Motion_Control_DB".Position = "Position_Axis".Output[1] THEN
149 |     "Motion_Control_DB".Exe_Absolute := FALSE;
150 |     #n := 0;
151 | END_IF;
152 | END_CASE;
153 | END_IF;

```


❖ Csharp

● Collecting data

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using MVSDK;
using CameraHandle = System.Int32;
using MvApi = MVSDK.MvApi;
using System.IO;
using S7.Net;
using System.Threading;
using System.Linq;
using System.Threading.Tasks;

namespace FirstStepMulti
{
    public partial class Form1 : Form
    {
        string fileConfig = @"text\config.txt";
        string fileLog = @"text\log.txt";
        private Plc PLC;
        bool Flat_PLC = true, Flat_Camera = false;
        string filename1, filename2;
        string SN_CAM1, SN_CAM2;
        string Path_Save_Image1, Path_Save_Image2;
        int CamNo = 1000;
        string SN_CAM;
        string Name_CAM;
        string[] folders;
        string[] array_SN_CAM = new string[2];
        bool[] flat_Connected_Cam = new bool[2];
        #region variable
        private AsyncSaveImage m_AsyncSave = new AsyncSaveImage();
        protected IntPtr[] m_Grabber = new IntPtr[4];
        protected CameraHandle[] m_hCamera = new CameraHandle[4];
        protected tSdkCameraDevInfo[] m_DevInfo;
        protected IntPtr m_ImageBuffer;
        protected tSdkFrameHead m_tFrameHead;
        protected pfnCameraGrabberFrameCallback m_FrameCallback;
        protected pfnCameraGrabberSaveImageComplete m_SaveImageComplete;
```

```
protected pfnCameraGrabberSaveImageComplete m_SaveImageComplete1;
#endregion

public Form1()
{
    InitializeComponent();
    Load_Config();
    TextBox[] Txt_NameCam = { txt_CamName1, txt_CamName2 };
    TextBox[] Txt_SNCam = { txt_SNCAM1, txt_SNCAM2 };
    folders = new string[2];
    m_FrameCallback = new pfnCameraGrabberFrameCallback(CameraGrabberFrameCallback);
    m_SaveImageComplete = new pfnCameraGrabberSaveImageComplete(CameraGrabberSaveImageComplete);
    m_SaveImageComplete1 = new pfnCameraGrabberSaveImageComplete(CameraGrabberSaveImageComplete1);
    MvApi.CameraEnumerateDevice(out m_DevInfo);
    int NumDev = (m_DevInfo != null ? Math.Min(m_DevInfo.Length, 4) : 0);
    for (int i = 0; i < NumDev; ++i)
    {
        SN_CAM = Encoding.ASCII.GetString(m_DevInfo[i].acSn).TrimEnd('\0').Substring(0, 12);
        Name_CAM = Encoding.ASCII.GetString(m_DevInfo[i].acFriendlyName).TrimEnd('\0').Substring(0, 5);
        array_SN_CAM[i] = SN_CAM;
        if (SN_CAM == SN_CAM1) { CamNo = 0; }
        if (SN_CAM == SN_CAM2) { CamNo = 1; }
        Txt_NameCam[CamNo].Text = Name_CAM;
        Txt_SNCam[CamNo].Text = SN_CAM;
        if (MvApi.CameraGrabber_Create(out m_Grabber[CamNo], ref m_DevInfo[i]) ==
CameraSdkStatus.CAMERA_STATUS_SUCCESS)
        {
            flat_Connected_Cam[CamNo] = true;
            MvApi.CameraGrabber_GetCameraHandle(m_Grabber[CamNo], out m_hCamera[i]);
            MvApi.CameraCreateSettingPage(m_hCamera[i], this.Handle, m_DevInfo[i].acFriendlyName, null, (IntPtr)0, 0);
            MvApi.CameraGrabber_SetRGBCallback(m_Grabber[CamNo], m_FrameCallback, IntPtr.Zero);
            if (CamNo == 1)
            {
                MvApi.CameraGrabber_SetSaveImageCompleteCallback(m_Grabber[CamNo], m_SaveImageComplete1,
IntPtr.Zero);
            }
            if (CamNo == 0)
            {
                MvApi.CameraGrabber_SetSaveImageCompleteCallback(m_Grabber[CamNo], m_SaveImageComplete,
IntPtr.Zero);
            }
            tSdkCameraCapbility cap;
            MvApi.CameraGetCapability(m_hCamera[i], out cap);
            if (cap.sIspCapacity.bMonoSensor != 0)
```

```
MvApi.CameraSetIspOutFormat(m_hCamera[i],
(uint)MVSDK.emImageFormat.CAMERA_MEDIA_TYPE_MONO8);
    //Cài đặt chế độ trigger
    MvApi.CameraSetTriggerMode(m_hCamera[i], 1);
    MvApi.CameraSetTriggerCount(m_hCamera[i], 1);
    if (m_Grabber[CamNo] != IntPtr.Zero)
        MvApi.CameraGrabber_StartLive(m_Grabber[CamNo]);
    }
}
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    for (int i = 0; i < 4; ++i)
    {
        if (m_Grabber[i] != IntPtr.Zero)
            MvApi.CameraGrabber_Destroy(m_Grabber[i]);
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    Connect_PLC();
    Load_tooltip();
    Load_Config();// load nội dung từ txt plc, camera, địa chỉ lưu ảnh.....
    LoadImages();
    Connected_2Cam();
}

private void CameraGrabberFrameCallback(IntPtr Grabber, IntPtr pFrameBuffer, ref tSdkFrameHead pFrameHead, IntPtr
Context)
{
}

// SAVE IMAGES
private void CameraGrabberSaveImageComplete(IntPtr Grabber,
    IntPtr Image,
    CameraSdkStatus Status,
    IntPtr Context)
{
    if (Image != IntPtr.Zero && checkBox1.Checked)
    {
        filename1 = System.IO.Path.Combine(Path_Save_Image1,
            string.Format("{0:ddMMyyyy_HHmmssffff}_Right.jpg", DateTime.Now));
        MvApi.CameraImage_SaveAsJpeg(Image, filename1, 70);
        console("Đang lưu ảnh");
    }
}
```

```
MvApi.CameraImage_Destroy(Image);
}
private void CameraGrabberSaveImageComplete1(IntPtr Grabber, IntPtr Image, CameraSdkStatus Status, IntPtr Context)
{
    if (Image != IntPtr.Zero && checkBox2.Checked)
    {
        filename2 = System.IO.Path.Combine(Path_Save_Image2,
            string.Format("{0:ddMMyyyy_HHmssfff}_Left.jpg", DateTime.Now));
        MvApi.CameraImage_SaveAsJpeg(Image, filename2, 70);
    }
    MvApi.CameraImage_Destroy(Image);
}
//SETTING CAMERA
private void buttonSettings1_Click(object sender, EventArgs e)
{
    if (m_Grabber[0] != IntPtr.Zero)
        MvApi.CameraShowSettingPage(m_hCamera[0], 1);
}
private void buttonSettings2_Click(object sender, EventArgs e)
{
    Console.WriteLine("a");
    if (m_Grabber[1] != IntPtr.Zero)
        MvApi.CameraShowSettingPage(m_hCamera[1], 1);
}
// TRIGGER CAMERA
private void buttonSnap1_Click(object sender, EventArgs e)
{
    SOFTTRIGGER(0);
}
private void buttonSnap2_Click(object sender, EventArgs e)
{
    SOFTTRIGGER(1);
}
public class AsyncSaveImage
{
    private class Item
    {
        public CameraHandle hCamera;
        public IntPtr Image;
        public string FileName;
        public emSdkFileType FileType;
        public byte Quality;
    }
    private Queue<Item> mImageQ = new Queue<Item>();
}
```

```
private int mMaxQSize = 1024;
private EventWaitHandle mQEvent = new EventWaitHandle(false, EventResetMode.ManualReset);
private EventWaitHandle mQuitEvent = new EventWaitHandle(false, EventResetMode.ManualReset);
private List<Thread> mSaveThreads;
private int mThreadsNum = System.Environment.ProcessorCount;
public AsyncSaveImage(int MaxQSize = 0, int nThreadNum = 0)
{
    if (MaxQSize > 0)
    {
        mMaxQSize = MaxQSize;
    }
    if (nThreadNum > 0)
    {
        mThreadsNum = nThreadNum;
    }
}
public void Start()
{
    if (mSaveThreads != null)
        return;
    mSaveThreads = new List<Thread>();
    mQuitEvent.Reset();
    for (int i = 0; i < mThreadsNum; ++i)
    {
        var thread = new Thread(SaveProc);
        thread.Start();
        mSaveThreads.Add(thread);
    }
}
public void Stop()
{
    if (mSaveThreads != null)
    {
        mQuitEvent.Set();
        foreach (var thread in mSaveThreads)
        {
            thread.Join();
        }
        mSaveThreads = null;
    }
}
public void Clear()
{
    lock (mImageQ)
```

```
{
    foreach (var item in mImageQ)
    {
        MvApi.CameraImage_Destroy(item.Image);
    }
    mImageQ.Clear();
}
}

public bool SaveImage(CameraHandle hCamera,
    string lpszFileName,
    IntPtr pbyImageBuffer,
    ref tSdkFrameHead pFrInfo,
    emSdkFileType byFileType,
    Byte byQuality)
{
    IntPtr Image;
    if (MvApi.CameraImage_Create(out Image, pbyImageBuffer, ref pFrInfo, 1) !=
CameraSdkStatus.CAMERA_STATUS_SUCCESS)
        return false;

    Item new_item = new Item();
    new_item.hCamera = hCamera;
    new_item.Image = Image;
    new_item.FileName = lpszFileName;
    new_item.FileType = byFileType;
    new_item.Quality = byQuality;
    lock (mImageQ)
    {
        if (mImageQ.Count < mMaxQSize)
        {
            mImageQ.Enqueue(new_item);
            mQEvent.Set();
            return true;
        }
        else
        {
            MvApi.CameraImage_Destroy(Image);
            return false;
        }
    }
}

private void SaveProc()
{
    while (true)
```

```
{
    if (WaitHandle.WaitAny(new WaitHandle[] { mQuitEvent, mQEvent }) == 0)
        break;
    Item item;
    lock (mImageQ)
    {
        if (mImageQ.Count < 1)
        {
            mQEvent.Reset();
            continue;
        }
        item = mImageQ.Dequeue();
    }

    SaveItem(item);
    MvApi.CameraImage_Destroy(item.Image);
}

private void SaveItem(Item item)
{
    IntPtr Image = item.Image;
    IntPtr pRaw, pHeadPtr;
    MvApi.CameraImage_GetData(Image, out pRaw, out pHeadPtr);
    tSdkFrameHead FrameHead = (tSdkFrameHead)Marshal.PtrToStructure(pHeadPtr, typeof(tSdkFrameHead));
    uint uIspOutFmt = 0;
    MvApi.CameraGetIspOutFormat(item.hCamera, ref uIspOutFmt);

    int w = FrameHead.iWidth, h = FrameHead.iHeight;
    if (FrameHead.iWidthZoomSw > 0 && FrameHead.iHeightZoomSw > 0)
    {
        w = FrameHead.iWidthZoomSw;
        h = FrameHead.iHeightZoomSw;
    }
    bool IspOutGray = (uIspOutFmt == (uint)MVSDK.cmImageFormat.CAMERA_MEDIA_TYPE_MONO8);
    int RequireBufferSize = w * h * (IspOutGray ? 1 : 3);
    IntPtr pFrameBuffer = MvApi.CameraAlignMalloc(RequireBufferSize, 16);
    if (pFrameBuffer != IntPtr.Zero)
    {
        MvApi.CameraImageProcess(item.hCamera, pRaw, pFrameBuffer, ref FrameHead);
        MvApi.CameraDisplayRGB24(item.hCamera, pFrameBuffer, ref FrameHead);
        MvApi.CameraSaveImage(item.hCamera, item.FileName, pFrameBuffer, ref FrameHead, item.FileType,
item.Quality);
        MvApi.CameraAlignFree(pFrameBuffer);
    }
}
```

```
    }
}
//SAVE FILE CONFIG
private void txt_PathSaveImg1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Modify_Config(5, txt_PathSaveImg1.Text);
    }
}
private void txt_PathSaveImg2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Modify_Config(6, txt_PathSaveImg2.Text);
    }
}
private void txt_PLC_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Modify_Config(4, txt_PLC.Text);
    }
}
//PLC- Switch AI-NoAI
private void Label_PLC_SW_Click(object sender, EventArgs e)
{
    if (Flat_PLC == false)
    {
        PLC.Write("M255.1", false);
        ConnectPLC.Start();
        Label_PLC_SW.Text = "AI Mode";
        Label_PLC_SW.BackColor = Color.LightGreen;
        Flat_PLC = true;
    }
}
//Camera - Manual Mode
private void Label_Camera_SW_Click(object sender, EventArgs e)
{
    if (flat_Connected_Cam[0] && flat_Connected_Cam[1])
    {
        if (Flat_Camera == true)
        {
            Flat_Camera = false;
        }
    }
}
```

```
Label_Camera_SW.BackColor = Color.Yellow;
Label_Camera_SW.Text = "Manual Mode";
buttonSettings1.Enabled = true;
buttonSettings2.Enabled = true;
}
else
{
    Flat_Camera = true;
    Label_Camera_SW.Text = "Trigger Mode";
    Label_Camera_SW.BackColor = Color.LightGreen;
    buttonSettings1.Enabled = false;
    buttonSettings2.Enabled = false;
}
}
}

//TIMER
private async void ConnectPLC_Tick(object sender, EventArgs e)
{
    try
    {
        if ((bool)PLC.Read("DB30.DBX0.0") == true)
        {
            console("PLC trigger Cam 1");
            SOFTTRIGGER(0); // Chụp ảnh Cam 1
        }
        if ((bool)PLC.Read("DB30.DBX256.0") == true) // Đảm bảo chỉ trigger một lần cho mỗi lệnh PLC
        {
            console("PLC trigger Cam 2");
            SOFTTRIGGER(1); // Chụp ảnh Cam 2
        }
        PLC.Write("M1001.3", true); // Báo PLC đã chụp xong Cam 1
        {
            PLC.Write("M1001.4", true); // Báo PLC đã chụp xong Cam 2
        }
    }
    catch (Exception)
    {
    }
}

private void CheckAuto_Manual_Tick(object sender, EventArgs e)
{
    if (Flat_PLC && Flat_Camera )
    {
        Label_AutoorManual.Text = "Auto Mode";
    }
}
```

```
        Label_AutoorManual.BackColor = Color.LightGreen;
    }
    else
    {
        Label_AutoorManual.Text = "Manual Mode";
        Label_AutoorManual.BackColor = Color.Yellow;
    }
}
//private void ConnectServer_Tick(object sender, EventArgs e)
//{
//}
// CHƯƠNG TRÌNH CON
private void Load_Config()
{
    if (File.Exists(fileConfig))
    {
        string[] lines = File.ReadAllLines(fileConfig);
        console("Đã load đường dẫn model.");
        console("Đã load file lưu ảnh.");
        txt_PLC.Text = lines[4]; // Load địa chỉ plc
        console("Đã load địa chỉ PLC.");
        txt_PathSaveImg1.Text = lines[5];
        txt_PathSaveImg2.Text = lines[6];
        Path_Save_Image1 = txt_PathSaveImg1.Text;
        Path_Save_Image2 = txt_PathSaveImg2.Text;
        console("Đã load ảnh test model.");
        folders = new string[] { Path_Save_Image1, Path_Save_Image2 };
        SN_CAM1 = lines[9];
        SN_CAM2 = lines[10];
        console(SN_CAM2);
    }
    else
    {
        console($"Vui lòng kiểm tra lại file config: {fileConfig}");
    }
}
private void Modify_Config(int row, string content)
{
    string[] lines = File.Exists(fileConfig) ? File.ReadAllLines(fileConfig) : new string[0];
    lines[row] = content;
    File.WriteAllLines(fileConfig, lines);
    console($"Đã thay dòng thứ {row} bằng: '{content}'");
}
private void console(string content)
```

```
{
    DateTime dt = DateTime.Now;
    listBox_console.Items.Add(dt + " - " + content);
    listBox_console.TopIndex = listBox_console.Items.Count - 1;
    try
    {
        using (StreamWriter writer = new StreamWriter(fileLog, true))
        {
            writer.WriteLine($"{DateTime.Now:yyyy-MM-dd HH:mm:ss} - {content}");
        }
    }
    catch (Exception e)
    {
    }
}

private void LoadImages()
{

    PictureBox[] pic = { DispWnd1, DispWnd2 };
    for (int i = 0; i < folders.Length; i++)
    {
        if (Directory.Exists(folders[i]))
        {

            var images = Directory.GetFiles(folders[i], "*.*")
                .Where(file => file.EndsWith(".jpg") || file.EndsWith(".png") || file.EndsWith(".jpeg") || file.EndsWith(".bmp"))
                .ToList();
            foreach (var image in images) ;
        }
        else
        {
            console($"Thư mục {folders[i]} không tồn tại!");
        }
    }
}

private void DispWnd1_Click(object sender, EventArgs e)
{

}

// C sharp - Camera.
private void Connected_2Cam()
```

```
{
    if (flat_Connected_Cam[0] && flat_Connected_Cam[1] == true)//kết nối đủ 2 cam thì mới cho trigger
    {
        Flat_Camera = true;
        Label_Camera_SW.Text = "Trigger Mode";
        Label_Camera_SW.BackColor = Color.LightGreen;
        Trigger2Camera();
    }
    else
    {
        Label_Camera_SW.BackColor = Color.Yellow;
        Label_Camera_SW.Text = "Manual Mode";
        Flat_Camera = false;
    }
}

private async Task SOFTTRIGGER(int i)
{
    int a = Select_NoCamera(i);
    if (m_Grabber[i] != IntPtr.Zero)
    {
        MvApi.CameraSoftTrigger(m_hCamera[a]);
        MvApi.CameraGrabber_SaveImageAsync(m_Grabber[i]); // YÊU CẦU LƯU ẢNH SAU KHI CHỤP
        console($"Trigger Camera {i}");
    }
    else
    {
        console($"Trigger Camera {i} Fail.");
    }
}

// C sharp - PLC
void Connect_PLC()
{
    try
    {
        PLC = new Plc(CpuType.S71200, txt_PLC.Text, 0, 1);
        PLC.Open();
        if (PLC.IsConnected)
        {
            console($"Kết nối PLC: {txt_PLC.Text} thành công.");
            Label_PLC_SW.Text = "AI Mode";
            Label_PLC_SW.BackColor = Color.LightGreen;
            Flat_PLC = true;
            Label_PLC_SW.Enabled = true;
            ConnectPLC.Start();// gọi timer để kiểm tra trigger chụp ảnh từ PLC
        }
    }
}
```

```
    }
    else
    {
        Label_PLC_SW.Text = "Disconnect";
        Label_PLC_SW.BackColor = Color.Yellow;
        Flat_PLC = false;
        console($"Kết nối PLC: {txt_PLC.Text} thất bại.");
        return;
    }
}
catch (Exception ex)
{
    Label_PLC_SW.Text = "Error!";
    Label_PLC_SW.BackColor = Color.Red;
    Flat_PLC = false;
}
}
// Multil Threads
private async Task Trigger2Camera()
{
    await Task.WhenAll(
        Task.Run(() => SOFTTRIGGER(0)),
        Task.Run(() => SOFTTRIGGER(1))
    );
}
private void Load_tooltip()
{
    tooltip3.SetToolTip(checkBox1, "Lưu ảnh chưa được dự đoán.");
    tooltip4.SetToolTip(Label_Camera_SW, "Nhấp để chuyển sang chế độ Manual.");
}
private int Select_NoCamera(int i)
{
    if (i == 0)
    {
        return Array.IndexOf(array_SN_CAM, SN_CAM1);
    }
    if (i == 1)
    {
        return Array.IndexOf(array_SN_CAM, SN_CAM2);
    }
    return -1;
}
}
}
```

● System

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using MVSDK;//使用SDK接口
using CameraHandle = System.Int32;
using MvApi = MVSDK.MvApi;
using System.IO;
using S7.Net;
using System.Threading;
using System.Security.Cryptography;
using System.Linq;
using System.Diagnostics;
using System.Net.Sockets;
using Newtonsoft.Json;
using System.Threading.Tasks;
using System.Reflection;
using System.Xml.Schema;
using System.Drawing.Imaging;
using System.Diagnostics.Eventing.Reader;
```

```
namespace FirstStepMulti
{
    public partial class Form1 : Form
    {
        string fileConfig = @"text\config.txt";
        string fileLog = @"text\log.txt";
        private Plc PLC;
        bool Flat_TCP = false, Flat_PLC = true, Flat_Camera = false, Flat_AI = true;
        string filename1, filename2;
        bool[] LIVE_CAM = new bool[] { false, false };
        string SN_CAM1, SN_CAM2;
        string Path_Save_Image1, Path_Save_Image2;
        string[] labelname = { "CAM 1", "CAM 2" };
        int CamNo = 1000;
        string SN_CAM;
        string Name_CAM;
```

```
string[] folders;
string[] array_SN_CAM = new string[2];
ListBox[] listBoxes;
Label[] Label_NameModel;
bool[] flat_Connected_Cam = new bool[2];
bool[] flat_done_predict = new bool[2];
CheckBox[] CheckBox_SaveImagePredict;
IntPtr[] hDispWnds;
#region variable
private AsyncSaveImage m_AsyncSave = new AsyncSaveImage();
protected IntPtr[] m_Grabber = new IntPtr[4];
protected CameraHandle[] m_hCamera = new CameraHandle[4];
protected tSdkCameraDevInfo[] m_DevInfo;
protected IntPtr m_ImageBuffer;
protected tSdkFrameHead m_tFrameHead;
protected pfnCameraGrabberFrameCallback m_FrameCallback;
protected pfnCameraGrabberSaveImageComplete m_SaveImageComplete;
protected pfnCameraGrabberSaveImageComplete m_SaveImageComplete1;
protected pfnCameraGrabberSaveImageComplete m_SaveImageComplete2;
#endregion
public Form1()
{
    InitializeComponent();
    Load_Config();
    hDispWnds = new IntPtr[] { this.DispWnd1.Handle, this.DispWnd2.Handle };
    Label_NameModel = new Label[] { label_content_model1, label_content_model2, };
    CheckBox_SaveImagePredict = new CheckBox[] { checkBox_SaveImagePredict1, checkBox_SaveImagePredict2 };
    TextBox[] Txt_NameCam = { txt_CamName1, txt_CamName2 };
    TextBox[] Txt_SNCam = { txt_SNCAM1, txt_SNCAM2 };
    folders = new string[2];
    m_FrameCallback = new pfnCameraGrabberFrameCallback(CameraGrabberFrameCallback);
    MvApi.CameraEnumerateDevice(out m_DevInfo);
    int NumDev = (m_DevInfo != null ? Math.Min(m_DevInfo.Length, 4) : 0);
    for (int i = 0; i < NumDev; ++i)
    {
        SN_CAM = Encoding.ASCII.GetString(m_DevInfo[i].acSn).TrimEnd('\0').Substring(0, 12);
        Name_CAM = Encoding.ASCII.GetString(m_DevInfo[i].acFriendlyName).TrimEnd('\0').Substring(0, 5);
        array_SN_CAM[i] = SN_CAM;
        if (SN_CAM == SN_CAM1) { CamNo = 0; }
        if (SN_CAM == SN_CAM2) { CamNo = 1; }
        Txt_NameCam[CamNo].Text = Name_CAM;
        Txt_SNCam[CamNo].Text = SN_CAM;
        if (MvApi.CameraGrabber_Create(out m_Grabber[CamNo], ref m_DevInfo[i]) ==
            CameraSdkStatus.CAMERA_STATUS_SUCCESS)
```

```
{
    flat_Connected_Cam[CamNo] = true;

    MvApi.CameraGrabber_GetCameraHandle(m_Grabber[CamNo], out m_hCamera[i]);
    MvApi.CameraCreateSettingPage(m_hCamera[i], this.Handle, m_DevInfo[i].acFriendlyName, null, (IntPtr)0, 0);
    MvApi.CameraGrabber_SetRGBCallback(m_Grabber[CamNo], m_FrameCallback, IntPtr.Zero);

    if (CamNo == 1)
    {
        MvApi.CameraGrabber_SetSaveImageCompleteCallback(m_Grabber[CamNo], m_SaveImageComplete1,
IntPtr.Zero);
    }
    if (CamNo == 0)
    {
        MvApi.CameraGrabber_SetSaveImageCompleteCallback(m_Grabber[CamNo], m_SaveImageComplete,
IntPtr.Zero);
    }

    tSdkCameraCapbility cap;
    MvApi.CameraGetCapability(m_hCamera[i], out cap);
    if (cap.sIspCapacity.bMonoSensor != 0)
        MvApi.CameraSetIspOutFormat(m_hCamera[i],
(uint)MVSDK.emImageFormat.CAMERA_MEDIA_TYPE_MONO8);

    //Cài đặt chế độ trigger
    MvApi.CameraSetTriggerMode(m_hCamera[i], 1);
    MvApi.CameraSetTriggerCount(m_hCamera[i], 1);
    if(Flat_Camera== true)
    {
        MvApi.CameraGrabber_SetHWnd(m_Grabber[CamNo], hDispWnds[CamNo]);
    }
    //console($"Conneted to {raw.Substring(0, 5)}.");
    if (m_Grabber[CamNo] != IntPtr.Zero)
        MvApi.CameraGrabber_StartLive(m_Grabber[CamNo]);
    }
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    for (int i = 0; i < 4; ++i)
```

```
{
    if (m_Grabber[i] != IntPtr.Zero)
        MvApi.CameraGrabber_Destroy(m_Grabber[i]);
}

}

private void Form1_Load(object sender, EventArgs e)
{
    Connect_PLC();
    Load_tooltip();
    Load_Config();// load nội dung từ txt plc, camera , py
    LoadImages();
    Connected_2Cam();
    if (Flat_Camera == true)
    {
        Run_py();
        ConnectServer.Start();//Timer kết nối server(python)
        Connect_PLC();
    }
}

private void CameraGrabberFrameCallback(IntPtr Grabber, IntPtr pFrameBuffer, ref tSdkFrameHead pFrameHead, IntPtr
Context)
{
}

//SETTING CAMERA
private void buttonSettings1_Click(object sender, EventArgs e)
{
    if (m_Grabber[0] != IntPtr.Zero)
        MvApi.CameraShowSettingPage(m_hCamera[0], 1);
}

private void buttonSettings2_Click(object sender, EventArgs e)
{
    Console.WriteLine("a");
    if (m_Grabber[1] != IntPtr.Zero)
        MvApi.CameraShowSettingPage(m_hCamera[1], 1);
}

// TRIGGER CAMERA
private void buttonSnap1_Click(object sender, EventArgs e)
{
    SOFTTRIGGER(0);
}

private void buttonSnap2_Click(object sender, EventArgs e)
```

```
{
    SOFTTRIGGER(1);
}
private void buttonSnap3_Click(object sender, EventArgs e)
{
    SOFTTRIGGER(2);
}
public class AsyncSaveImage
{
    private class Item
    {
        public CameraHandle hCamera;
        public IntPtr Image;
        public string FileName;
        public emSdkFileType FileType;
        public byte Quality;
    }
    private Queue<Item> mImageQ = new Queue<Item>();
    private int mMaxQSize = 1024;
    private EventWaitHandle mQEvent = new EventWaitHandle(false, EventResetMode.ManualReset);
    private EventWaitHandle mQuitEvent = new EventWaitHandle(false, EventResetMode.ManualReset);
    private List<Thread> mSaveThreads;
    private int mThreadsNum = System.Environment.ProcessorCount;
    public AsyncSaveImage(int MaxQSize = 0, int nThreadNum = 0)
    {
        if (MaxQSize > 0)
        {
            mMaxQSize = MaxQSize;
        }
        if (nThreadNum > 0)
        {
            mThreadsNum = nThreadNum;
        }
    }
    public void Start()
    {
        if (mSaveThreads != null)
            return;

        mSaveThreads = new List<Thread>();
        mQuitEvent.Reset();
        for (int i = 0; i < mThreadsNum; ++i)
        {
            var thread = new Thread(SaveProc);
```

```
        thread.Start();
        mSaveThreads.Add(thread);
    }
}
public void Stop()
{
    if (mSaveThreads != null)
    {
        mQuitEvent.Set();
        foreach (var thread in mSaveThreads)
        {
            thread.Join();
        }
        mSaveThreads = null;
    }
}
public void Clear()
{
    lock (mImageQ)
    {
        foreach (var item in mImageQ)
        {
            MvApi.CameraImage_Destroy(item.Image);
        }
        mImageQ.Clear();
    }
}
public bool SaveImage(CameraHandle hCamera,
    string lpszFileName,
    IntPtr pbyImageBuffer,
    ref tSdkFrameHead pFrInfo,
    emSdkFileType byFileType,
    Byte byQuality)
{
    IntPtr Image;
    if (MvApi.CameraImage_Create(out Image, pbyImageBuffer, ref pFrInfo, 1) !=
CameraSdkStatus.CAMERA_STATUS_SUCCESS)
        return false;

    Item new_item = new Item();
    new_item.hCamera = hCamera;
    new_item.Image = Image;
    new_item.FileName = lpszFileName;
    new_item.FileType = byFileType;
```

```
new_item.Quality = byQuality;

lock (mImageQ)
{
    if (mImageQ.Count < mMaxQSize)
    {
        mImageQ.Enqueue(new_item);
        mQEvent.Set();
        return true;
    }
    else
    {
        MvApi.CameraImage_Destroy(Image);
        return false;
    }
}
}

private void SaveProc()
{
    while (true)
    {
        if (WaitHandle.WaitAny(new WaitHandle[] { mQuitEvent, mQEvent }) == 0)
            break;

        Item item;
        lock (mImageQ)
        {
            if (mImageQ.Count < 1)
            {
                mQEvent.Reset();
                continue;
            }
            item = mImageQ.Dequeue();
        }

        SaveItem(item);
        MvApi.CameraImage_Destroy(item.Image);
    }
}

private void SaveItem(Item item)
{
    IntPtr Image = item.Image;
```

```
IntPtr pRaw, pHeadPtr;
MvApi.CameraImage_GetData(Image, out pRaw, out pHeadPtr);
tSdkFrameHead FrameHead = (tSdkFrameHead)Marshal.PtrToStructure(pHeadPtr, typeof(tSdkFrameHead));

uint uIspOutFmt = 0;
MvApi.CameraGetIspOutFormat(item.hCamera, ref uIspOutFmt);

int w = FrameHead.iWidth, h = FrameHead.iHeight;
if (FrameHead.iWidthZoomSw > 0 && FrameHead.iHeightZoomSw > 0)
{
    w = FrameHead.iWidthZoomSw;
    h = FrameHead.iHeightZoomSw;
}
bool IspOutGray = (uIspOutFmt == (uint)MVSDK.emImageFormat.CAMERA_MEDIA_TYPE_MONO8);
int RequireBufferSize = w * h * (IspOutGray ? 1 : 3);
IntPtr pFrameBuffer = MvApi.CameraAlignMalloc(RequireBufferSize, 16);
if (pFrameBuffer != IntPtr.Zero)
{
    MvApi.CameraImageProcess(item.hCamera, pRaw, pFrameBuffer, ref FrameHead);
    MvApi.CameraDisplayRGB24(item.hCamera, pFrameBuffer, ref FrameHead);

    MvApi.CameraSaveImage(item.hCamera, item.FileName, pFrameBuffer, ref FrameHead, item.FileType,
item.Quality);
    MvApi.CameraAlignFree(pFrameBuffer);
}
}
}
//SAVE FILE CONFIG
private void txt_PathSaveImg1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Modify_Config(7, txt_PathSaveImg1.Text);
    }
}
private void txt_PathSaveImg2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Modify_Config(8,txt_PathSaveImg2.Text);
    }
}
private void txt_PLC_KeyDown(object sender, KeyEventArgs e)
{

```

```
        if (e.KeyCode == Keys.Enter)
        {
            Modify_Config(6, txt_PLC.Text);
        }
    }
    private void txt_hostpost_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Enter)
        {
            Modify_Config(10, txt_hostpost.Text);
        }
    }
    private void txt_PythonServer_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Enter)
        {
            Modify_Config(11, txt_PythonServer.Text);
        }
    }
    //PLC- Switch AI-NoAI
    private void Label_PLC_SW_Click(object sender, EventArgs e)
    {
        if (Flat_PLC == false)
        {
            PLC.Write("M255.1", false);
            ConnectPLC.Start();
            Label_PLC_SW.Text = "AI Mode";
            Label_PLC_SW.BackColor = Color.LightGreen;
            Flat_PLC = true;
        }
    }
    //Camera - Manual Mode
    private void Label_Camera_SW_Click(object sender, EventArgs e)
    {
        if (flat_Connected_Cam[0] && flat_Connected_Cam[1])
        {
            if (Flat_Camera == true)
            {
                Flat_Camera = false;
                Label_Camera_SW.BackColor = Color.Yellow;
                Label_Camera_SW.Text = "Manual Mode";
                buttonSettings1.Enabled = true;
                buttonSettings2.Enabled = true;
            }
        }
    }
}
```

```
else
{
    Flat_Camera = true;
    Label_Camera_SW.Text = "Trigger Mode";
    Label_Camera_SW.BackColor = Color.LightGreen;

    buttonSettings1.Enabled = false;
    buttonSettings2.Enabled = false;
}
}

}

//TIMER
private async void ConnectPLC_Tick(object sender, EventArgs e)
{
    try
    {
        if ((bool)PLC.Read("DB30.DBX0.0") == true && !flat_done_predict[0])
        {
            console("PLC trigger Cam 1");
            SOFTTRIGGER(0); // Chụp ảnh Cam 1
        }
        if ((bool)PLC.Read("DB30.DBX256.0") == true && !flat_done_predict[1]) // Đảm bảo chỉ trigger một lần cho mỗi
lệnh PLC
        {
            console("PLC trigger Cam 2");
            SOFTTRIGGER(1); // Chụp ảnh Cam 2
        }
        if (flat_done_predict[0] == true)
        {
            console("Ảnh Cam 1 đã được xử lý.");
            PLC.Write("M1001.3", true); // Báo PLC đã chụp xong Cam 1
            if (Label_NameModel[0].ForeColor == Color.Green) // Giả sử màu xanh là OK
            {
                PLC.Write("DB30.DBX514.0", true); // OK
            }
            else if (Label_NameModel[0].ForeColor == Color.Red) // Giả sử màu đỏ là NG
            {
                PLC.Write("DB30.DBX514.0", false); // NG
            }
            flat_done_predict[0] = false; // Reset cờ để sẵn sàng cho trigger tiếp theo
        }
    }
}
```

```
console("Ảnh Cam 2 đã được xử lý.");
PLC.Write("M1001.4", true); // Báo PLC đã chụp xong Cam 2

// Gửi trạng thái OK (1) hoặc NG (3) về PLC cho Cam 2
if (Label_NameModel[1].ForeColor == Color.Green) // Giả sử màu xanh là OK
{
    PLC.Write("DB30.DBX514.1", true); // OK
}
else if (Label_NameModel[1].ForeColor == Color.Red) // Giả sử màu đỏ là NG
{
    PLC.Write("DB30.DBX514.1", false); // NG
}
flat_done_predict[1] = false; // Reset cờ để sẵn sàng cho trigger tiếp theo
}
}
catch (Exception)
{
}
}
private void CheckAuto_Manual_Tick(object sender, EventArgs e)
{
    if (Flat_PLC && Flat_TCP && Flat_Camera && Flat_AI == true)
    {
        Label_AutoorManual.Text = "Auto Mode";
        Label_AutoorManual.BackColor = Color.LightGreen;
    }
    else
    {
        Label_AutoorManual.Text = "Manual Mode";
        Label_AutoorManual.BackColor = Color.Yellow;
    }
}
private void ConnectServer_Tick(object sender, EventArgs e)
{
    Connect_Server();
}
// CHƯƠNG TRÌNH CON
private void Load_Config()
{
    if (File.Exists(fileConfig))
    {
        string[] lines = File.ReadAllLines(fileConfig);
```

```
console("Đã load đường dẫn model.");
txt_PathSaveImg1.Text = lines[3]; // Load đường dẫn nơi lưu ảnh
txt_PathSaveImg2.Text = lines[4];
console("Đã load file lưu ảnh.");
txt_PLC.Text = lines[6]; // Load địa chỉ plc
console("Đã load địa chỉ PLC.");
txt_PathSaveImg1.Text = lines[7];
txt_PathSaveImg2.Text = lines[8];
Path_Save_Image1 = txt_PathSaveImg1.Text;
Path_Save_Image2 = txt_PathSaveImg2.Text;
console("Đã load ảnh test model.");
txt_hostpost.Text = lines[10];
txt_PythonServer.Text = lines[11];
folders = new string[] { Path_Save_Image1, Path_Save_Image2 };
SN_CAM1 = lines[12];
SN_CAM2 = lines[13];
console( SN_CAM2);
}
else
{
    console($"Vui lòng kiểm tra lại file config: {fileConfig}");
}
}
private void Modify_Config(int row, string content)
{
    string[] lines = File.Exists(fileConfig) ? File.ReadAllLines(fileConfig) : new string[0];
    lines[row] = content;
    File.WriteAllLines(fileConfig, lines);
    console($"Đã thay dòng thứ {row} bằng: '{content}'");
}
private void console(string content)
{
    DateTime dt = DateTime.Now;
    listBox_console.Items.Add(dt + " - " + content);
    listBox_console.TopIndex = listBox_console.Items.Count - 1;
    try
    {
        using (StreamWriter writer = new StreamWriter(fileLog, true))
        {
            writer.WriteLine($"{DateTime.Now:yyyy-MM-dd HH:mm:ss} - {content}");
        }
    }
    catch (Exception e)
    {
    }
```

```
    }
}
private void LoadImages()
{
    PictureBox[] pic = { DispWnd1, DispWnd2 };
    for (int i = 0; i < folders.Length; i++)
    {
        if (Directory.Exists(folders[i]))
        {
            var images = Directory.GetFiles(folders[i], "*..*")
                .Where(file => file.EndsWith(".jpg") || file.EndsWith(".png") || file.EndsWith(".jpeg") || file.EndsWith(".bmp"))
                .ToList();
            foreach (var image in images) ;
        }
        else
        {
            console($"Thư mục {folders[i]} không tồn tại!");
        }
    }
}

//C Sharp - Python
private void Run_py()
{
    string pythonFile = txt_PythonServer.Text;
    ProcessStartInfo start = new ProcessStartInfo();
    start.FileName = "cmd.exe";
    start.Arguments = $"/k python \"{pythonFile}\""; // dùng /k để giữ cửa sổ CMD mở sau khi chạy
    start.UseShellExecute = true; // phải dùng true để hiện cửa sổ CMD
    start.CreateNoWindow = false;
    try
    {
        Process.Start(start); // Lưu lại process
        console($"Đã tạo server - {txt_hostpost.Text}");
    }
    catch (Exception ex)
    {
        console("Không mở được file Python: " + ex.Message);
    }
}
private void Connect_Server()
{
    try
```

```
{
    using (TcpClient client = new TcpClient(txt_hostpost.Text, 5000))
    using (NetworkStream stream = client.GetStream())
    {
        byte[] flatStatusBytes = new byte[1];
        stream.Read(flatStatusBytes, 0, 1);
        bool flat_success_Server = BitConverter.ToBoolean(flatStatusBytes, 0);
        Label_ServerSW.BackColor = flat_success_Server ? Color.LightGreen : Color.Red;
        Label_ServerSW.Text = "Connected";
        Flat_TCP = true;
        //groupBox_AI.Enabled = flat_success_Server;
        ConnectServer.Stop();
    }
}
catch (Exception ex)
{
    console($"Lỗi kết nối tới server - {txt_hostpost.Text} - 5000.");
    Flat_TCP = false;
}
}
private async Task Predict_Python(int i, string imagePath)
{
    try
    {
        using (TcpClient client = new TcpClient(txt_hostpost.Text, 5000))
        using (NetworkStream stream = client.GetStream())
        {
            string model = $"model{i}";
            // Gửi tên model
            byte[] modelNameBytes = System.Text.Encoding.UTF8.GetBytes(model);
            byte[] modelNameLength = BitConverter.GetBytes(modelNameBytes.Length);
            await stream.WriteAsync(modelNameLength, 0, modelNameLength.Length);
            await stream.WriteAsync(modelNameBytes, 0, modelNameBytes.Length);

            // Gửi ảnh
            byte[] imageBytes = File.ReadAllBytes(imagePath);
            byte[] imageSize = BitConverter.GetBytes(imageBytes.Length);
            await stream.WriteAsync(imageSize, 0, imageSize.Length);
            await stream.WriteAsync(imageBytes, 0, imageBytes.Length);

            // Nhận kích thước ảnh predict
            byte[] predictedImageSizeBytes = new byte[4];
            await stream.ReadAsync(predictedImageSizeBytes, 0, 4);
            int predictedImageSize = BitConverter.ToInt32(predictedImageSizeBytes, 0);
```

```
// Nhận ảnh predict
byte[] predictedImageBytes = new byte[predictedImageSize];
int totalBytesRead = 0;
while (totalBytesRead < predictedImageSize)
{
    int bytesRead = await stream.ReadAsync(predictedImageBytes, totalBytesRead, predictedImageSize -
totalBytesRead);
    if (bytesRead == 0) break;
    totalBytesRead += bytesRead;
}
// Nhận trạng thái OK/NG
byte[] flatStatusBytes = new byte[1];
await stream.ReadAsync(flatStatusBytes, 0, 1);
bool flat_status_of_image = BitConverter.ToBoolean(flatStatusBytes, 0);
// Cập nhật giao diện trên UI thread
this.Invoke((MethodInvoker)() =>
{
    using (MemoryStream ms = new MemoryStream(predictedImageBytes))
    {
        PictureBox[] pic = { DispWnd1, DispWnd2 };
        if (pic[i].Image != null)
        {
            pic[i].Image.Dispose(); // giải phóng ảnh cũ tránh leak
        }
        pic[i].Image = Image.FromStream(ms);
    }
    Label_NameModel[i].Text = flat_status_of_image ? labelname[i] + " OK" : labelname[i] + " NG";
    Label_NameModel[i].ForeColor = flat_status_of_image ? Color.Green : Color.Red;
});
flat_done_predict[i] = true;
Console.WriteLine(flat_done_predict[i]);
}
}
catch (Exception ex)
{
    Console.WriteLine($"Lỗi Predict_Python model {i}: {ex.Message}");
}
}
private void DispWnd1_Click(object sender, EventArgs e)
{
}
// C sharp - Camera.
private void Connected_2Cam()
```

```
{
    if (flat_Connected_Cam[0] && flat_Connected_Cam[1] == true)//kết nối đủ 2 cam thì mới cho trigger
    {
        Flat_Camera = true;
        Label_Camera_SW.Text = "Trigger Mode";
        Label_Camera_SW.BackColor = Color.LightGreen;
        Trigger2Camera();
    }
    else
    {
        Label_Camera_SW.BackColor = Color.Yellow;
        Label_Camera_SW.Text = "Manual Mode";
        Flat_Camera = false;
    }
}

private async Task SOFTTRIGGER(int i)
{
    int a = Select_NoCamera(i);
    if (m_Grabber[i] != IntPtr.Zero)
    {
        MvApi.CameraSoftTrigger(m_hCamera[a]);
        MvApi.CameraGrabber_SaveImageAsync(m_Grabber[i]); // YÊU CẦU LƯU ẢNH SAU KHI CHỤP
        console($"Trigger Camera {i}");
    }
    else
    {
        console($"Trigger Camera {i} Fail.");
    }
}

// C sharp - PLC
void Connect_PLC()
{
    try
    {
        PLC = new Plc(CpuType.S71200, txt_PLC.Text, 0, 1);
        PLC.Open();
        if (PLC.IsConnected)
        {
            console($"Kết nối PLC: {txt_PLC.Text} thành công.");
            Label_PLC_SW.Text = "AI Mode";
            Label_PLC_SW.BackColor = Color.LightGreen;
            Flat_PLC = true;
            Label_PLC_SW.Enabled = true;
            ConnectPLC.Start();// gọi timer để kiểm tra trigger chụp ảnh từ PLC
        }
    }
}
```

```
}
else
{
    Label_PLC_SW.Text = "Disconnect";
    Label_PLC_SW.BackColor = Color.Yellow;
    Flat_PLC = false;
    console($"Kết nối PLC: {txt_PLC.Text} thất bại.");
    return;
}
}
catch (Exception ex)
{
    Label_PLC_SW.Text = "Error!";
    Label_PLC_SW.BackColor = Color.Red;
    Flat_PLC = false;
}
}
// Multil Threads
private async Task Trigger2Camera()
{
    await Task.WhenAll(
        Task.Run(() => SOFTTRIGGER(0)),
        Task.Run(() => SOFTTRIGGER(1))
    );
    while (flat_done_predict[1] != true)
    {
        await Task.Delay(10);
    }
    Console.WriteLine("Donmeeeee");
    flat_done_predict[1] = false;
}
private async Task Test()
{
    await Task.WhenAll(
        Task.Run(() => Predict_Python(0, filename1)),
        Task.Run(() => Predict_Python(1, filename2))
    );
}
private void Load_tooltip()
{
    toolTip3.SetToolTip(checkBox1, "Lưu ảnh chưa được dự đoán.");
    toolTip4.SetToolTip(Label_Camera_SW, "Nhấp để chuyển sang chế độ Manual.");
    toolTip5.SetToolTip(groupBox_PLC, "Nhấp để chuyển chế độ AI hoặc NoAI.");
}
```

```
private int Select_NoCamera(int i)
{
    if (i == 0)
    {
        return Array.IndexOf(array_SN_CAM, SN_CAM1);
    }
    if (i == 1)
    {
        return Array.IndexOf(array_SN_CAM, SN_CAM2);
    }
    return -1;
}
}
```

❖ Python

● Auto labelling

```
import os
import shutil

# Đường dẫn đến file nhãn mẫu (TXT theo định dạng YOLO)
template_label = "C:/Users/ACER/Documents/CAPSTONE_PROJECT/Train_model_2/Train/2025-03-09_10-36-39.txt"
# Thư mục chứa các ảnh cần gán nhãn
image_folder = "C:/Users/ACER/Documents/CAPSTONE_PROJECT/Train_model_2/Val/NG"
# Thư mục để lưu các file nhãn mới
output_label_folder = "C:/Users/ACER/Documents/CAPSTONE_PROJECT/Train_model_2/Val/NG"

# Tạo thư mục đầu ra nếu chưa có
if not os.path.exists(output_label_folder):
    os.makedirs(output_label_folder)

# Lặp qua tất cả các ảnh trong thư mục
for image_file in os.listdir(image_folder):
    if image_file.endswith((".jpg", ".png", ".JPG", ".PNG")): # Chỉ xử lý file ảnh
        # Tạo tên file nhãn mới dựa trên tên ảnh
        label_file = os.path.splitext(image_file)[0] + ".txt"
        output_label_path = os.path.join(output_label_folder, label_file)
        # Sao chép file nhãn mẫu sang vị trí mới
        shutil.copy(template_label, output_label_path)

print("Đã sao chép nhãn YOLO cho tất cả ảnh!")
```

● Server

```
import socket
import threading
import cv2
import torch
```

```
import struct
import numpy as np
from datetime import datetime
from ultralytics import YOLO

HOST = "192.168.0.1"
PORT = 5000
conf_threshold = 0.5

models = {
    "model0": YOLO("C:/Users/ACER/Downloads/best (1).pt"),
    "model1": YOLO("C:/Users/ACER/Documents/CAPSTONE_PROJECT/best.pt"),
}

num_crop = {"model0": 2, "model1": 2, "model2": 6}

def handle_client(conn, addr):
    try:
        # Nhận độ dài tên model
        model_name_length = struct.unpack("I", conn.recv(4))[0]
        model_name = conn.recv(model_name_length).decode()

        # Nhận kích thước ảnh
        image_size = struct.unpack("I", conn.recv(4))[0]

        # Nhận dữ liệu ảnh
        image_data = b""
        while len(image_data) < image_size:
            packet = conn.recv(4096)
            if not packet:
                break
            image_data += packet

        # Chuyển đổi ảnh
        nparr = np.frombuffer(image_data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        # Dự đoán
        model = models[model_name]
        results = model(img, conf=conf_threshold)
        flat_status_of_image = True
        box_count = 0
        if results and len(results[0].boxes) > 0:
            for box in results[0].boxes:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                conf = float(box.conf[0])
                cls = int(box.cls[0])
                label = model.names[cls]
                if "NG" in label:
                    flat_status_of_image = False
                    color = (0, 0, 255)
```

```
else:
    color = (0, 255, 0)
    cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
    cv2.putText(img, f'{label} {conf:.2f}', (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.5, color, 2)
    box_count += 1
    if box_count == num_crop[model_name]:
        break
if box_count < num_crop[model_name]:
    flat_status_of_image = False

# Mã hóa ảnh kết quả
_, encoded_img = cv2.imencode('.jpg', img)
encoded_img_bytes = encoded_img.tobytes()

# Gửi ảnh về C#
conn.send(struct.pack("I", len(encoded_img_bytes)))
conn.sendall(encoded_img_bytes)

# Gửi trạng thái
conn.send(struct.pack("?", flat_status_of_image))
print(f'{datetime.now().strftime("%Y-%m-%d %H:%M:%S')} Status: {flat_status_of_image},{model_name}')
except Exception as e:
    print(f'Lỗi xử lý client {addr}: {e}')
finally:
    conn.close()
def start_server():
    try:
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.bind((HOST, PORT))
        server.listen(5)
        conn1, addr1 = server.accept()
        conn1.send(struct.pack("?", True)) # khi file python được mở thì giờ true về c sharp để check kết nối
        print(f'{datetime.now().strftime("%Y-%m-%d %H:%M:%S')} Server started on {HOST}:{PORT}...')
        print(f'{datetime.now().strftime("%Y-%m-%d %H:%M:%S')} Waiting trigger.....')
    except Exception as e:
        print(f'Error starting server: {e}')
        return
while True:
    try:
        conn, addr = server.accept()
        #print(f'{datetime.now().strftime("%Y-%m-%d %H:%M:%S')} Connection from {addr}')
        threading.Thread(target=handle_client, args=(conn, addr)).start()
    except Exception as e:
        print(f'Lỗi accept client: {e}')
if __name__ == "__main__":
    print(f'connected')
    start_server()
```

