VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

Applying Machine Learning in Waste Identification and Classification for Delta Robot Control

BY
TRAN GIA BAO

A THESIS PROJECT SUBMITTED TO THE SCHOOL OF ELECTRICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ELECTRICAL ENGINEERING

HO CHI MINH CITY, VIETNAM
JANUARY, 2026

Applying Machine Learning in Waste Identification and Classification for Delta Robot Control

BY

TRAN GIA BAO

Under the guidance and approval of the committee, and approved by its members, this thesis has been accepted in partial fulfillment of the requirements for the degree.

Approved by:

_____

Chairperson

_____

Committee member

_____

Committee member

_____

Committee member

_____

Committee member

_____

Committee member

HONESTY DECLARATION

My name is Tran Gia Bao. I would like to declare that, apart from the acknowledged references, this thesis either does not use language, ideas, or other original material from anyone, or has not been previously submitted to any other educational and research programs or institutions. I fully understand that any writings in this thesis that contradict the above statement will automatically lead to rejection from the EE/AC program at the International University – Vietnam National University Ho Chi Minh City.

Date: 12/01/2026
Student's Signature

Tran Gia Bao

TURNITIN DECLARATION

Name of Student: Tran Gia Bao

Date: 12/01/2026

Advisor Signature

Full name

_____

Date: 12/01/2026

Student Signature

Full name

_____

Date: 12/01/2026

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all the individuals who have contributed valuable knowledge to the success and completion of this thesis project. Firstly, I would like to express my appreciation to my advisor, M.Sc. Vo Minh Thanh for his constructive feedback, encouragement, and guidance in completing this project with his valuable experience. M.Sc. Vo Minh Thanh‘s professional attitude and working style are an excellent model for me to follow. Secondly, I would also like to thank the professors, doctors, and masters of the Department of Electrical Engineering for imparting valuable knowledge to me. Furthermore, I would like to thank my professors and colleagues for their unwavering support and valuable advice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOTATIONS

Acronyms and Terminology

Acronym	Full Name	Definition
AI	Artificial Intelligence	Machine learning and computational intelligence techniques enabling automated decision-making and perception.
CNN	Convolutional Neural Network	Deep learning architecture using learnable convolutional filters to extract spatial features from images. Foundational technology for modern computer vision systems.
YOLO	You Only Look Once	Single-stage real-time object detection framework that predicts bounding boxes and class labels in a single forward pass, enabling fast inference on embedded hardware.
YOLOv8n	YOLO version 8 nano	Lightweight variant of YOLOv8 optimized for edge devices; primary model architecture selected for Jetson Nano deployment in this thesis.
TP	True Positive	Correct detection where model predicts an object of the true class with IoU ≥ 0.5. Used to compute precision and recall metrics.
FP	False Positive	Incorrect detection where model predicts an object that does not exist or predicts wrong class. Indicates over-sensitivity of detector.

Acronym	Full Name	Definition
FN	False Negative	Missed detection where an actual object is not detected by the model. Indicates under-sensitivity or missed target.
F1-Score	F1 Score	Harmonic mean of precision and recall: $F_1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$. Balanced metric for classifier evaluation when classes are imbalanced.
mAP	Mean Average Precision	Average of precision-recall curves across all object classes. Standard metric for evaluating object detection models. Reported at IoU threshold 0.5 (mAP@0.5) in this work.
mAP@0.5	Mean Average Precision at IoU 0.5	mAP computed using intersection-over-union (IoU) threshold of 0.5 to classify detections as true or false positives. Primary accuracy metric used throughout thesis.
IoU	Intersection over Union	Overlap ratio between predicted bounding box and ground truth: IoU = Area(Intersection) / Area(Union). Measures localization accuracy of detection.
KNN	K-Nearest Neighbors	Distance-based classifier assigning objects to classes based on k nearest labeled neighbors. Not used in current work but mentioned for completeness.
ReLU	Rectified Linear Unit	Activation function $f(x) = \max(0, x)$ used in neural networks. Primary non-linearity in YOLOv8n backbone and detection heads.
CSV	Comma-Separated Values	Plain-text tabular data format used for storing and exchanging structured datasets and experimental results.

Acronym	Full Name	Definition
FPS	Frames Per Second	Throughput metric indicating number of images processed per second. Target: 5–8 FPS on Jetson Nano; achieved: 8 FPS.
IoT	Internet of Things	Network-connected embedded devices and systems. System supports future IoT integration for facility-level monitoring.
GPU	Graphics Processing Unit	Parallel compute accelerator used for real-time neural network inference and model training. Jetson Nano has 128-core Maxwell GPU.
CPU	Central Processing Unit	Main processor executing control logic and coordination. Jetson Nano has 4× ARM Cortex-A57 cores @ 1.43 GHz.
RAM	Random Access Memory	Volatile shared memory on embedded platform. Jetson Nano has 4 GB LPDDR4; model deployment uses ~1.2 GB.
ONNX	Open Neural Network Exchange	Standardized intermediate representation for neural network models enabling cross-platform deployment. Used for converting PyTorch models to TensorRT.
TensorRT	NVIDIA TensorRT	Inference optimization runtime providing model compression, layer fusion, and quantization for real-time GPU acceleration on NVIDIA hardware.
FP32	32-bit Floating Point	Full-precision neural network weights using IEEE 754 standard. Baseline precision for model training and validation.
FP16	16-bit Floating Point	Half-precision weights reducing model size by 50% and inference latency by 2–

Acronym	Full Name	Definition
		3× with minimal accuracy loss. Used for Jetson Nano deployment.
INT8	8-bit Integer	Quantized precision further reducing model size and latency, but with greater accuracy degradation. Not used due to waste sorting quality requirements.
NMS	Non-Maximum Suppression	Post-processing step removing redundant overlapping detections, keeping only highest-confidence boxes. Reduces false positives.
IK	Inverse Kinematics	Mathematical computation converting desired end-effector positions to joint angles for robotic control. Essential for robot trajectory generation from vision detections.
NEMA	National Electrical Manufacturers Association	Standard specifying stepper motor dimensions and torque ratings. System uses NEMA23 motors (23 × 23 mm mounting footprint).
USB	Universal Serial Bus	Standard interface for peripheral communication. Connects camera and robot control to Jetson Nano at USB 2.0 (480 Mbps) or USB 3.0.
I²C	Inter-Integrated Circuit	Serial communication protocol for short-distance chip-to-chip communication. Considered but not selected due to line length constraints.
SPI	Serial Peripheral Interface	High-speed synchronous serial protocol. Considered but not selected; USB selected for simplicity and proven reliability.

Acronym	Full Name	Definition
UART	Universal Asynchronous Receiver-Transmitter	Asynchronous serial protocol. Used by Arduino for debugging and telemetry output.
RoboFlow	RoboFlow Dataset Platform	Public dataset hosting service; source of 1,484 professionally annotated waste images on black conveyor belt background.
TrashNet	TrashNet Open Dataset	Academic open-source waste classification dataset. Reference for dataset design principles; not directly used in current work.
JetPack	NVIDIA JetPack	Software stack for Jetson embedded platforms including OS, CUDA, cuDNN, TensorRT. Version 4.6.4 used on Jetson Nano 4GB.
IU-VNU	International University, Vietnam National University	Host institution for this thesis in Ho Chi Minh City, Vietnam.
EE/AC	School of Electrical Engineering, Control Engineering and Automation	Academic unit under which this thesis is submitted.

Mathematical Notations

Notation	Meaning	Context
N_class	Number of classes	N_class = 3 for three waste categories (plastic, metal, paper).
N_data	Total dataset size	N_data = 1,705 total images combined from RoboFlow and laboratory collection.

Notation	Meaning	Context
N_train, N_val, N_test	Training, validation, test set sizes	Stratified split: 1,194 training, 256 validation, 255 test images.
C_i	i-th class	i ∈ {0 (plastic), 1 (metal), 2 (paper)}.
AP_c	Average precision for class c	Integral of precision-recall curve for individual waste category.
mAP	Mean average precision	mAP = (1/3) ∑_c AP_c; reported at IoU threshold 0.5.
P	Precision	P = TP / (TP + FP); fraction of predicted positives that are correct.
R	Recall	R = TP / (TP + FN); fraction of ground truth positives detected.
IoU_threshold	Intersection over union threshold	Detection with IoU ≥ 0.5 classified as TP; otherwise FP. Standard threshold in detection literature.
λ	Learning rate	Decayed via cosine annealing: $\lambda(t) = \lambda_{min} + 0.5(\lambda_{max} - \lambda_{min})[1 + \cos(\pi t/T)]$.
λ_max	Maximum learning rate	λ_max = 0.01 at epoch 0.
λ_min	Minimum learning rate	λ_min = 0.0001 at final epoch.
ω_decay	Weight decay (L2 regularization)	ω_decay = 0.0005; penalizes large weight magnitudes.
B	Batch size	B = 32; number of images per training iteration.

Notation	Meaning	Context
E	Number of epochs	E = 100; maximum training iterations.
T	Total epochs for LR schedule	T = 100 for cosine annealing.
(x_c, y_c)	Centroid coordinates	Detected bounding box center in image coordinates, mapped to robot workspace coordinates.
W, H	Bounding box width and height	Bounding box dimensions in pixels; used for object size estimation.
Confidence	Detection confidence score	Output probability from neural network; detections with confidence < 0.5 filtered during inference.
θ_1, θ_2, θ_3	Joint angles	Motor step positions for three parallel linkages of Delta robot; converted from end-effector position via inverse kinematics.
±ε	Positioning tolerance	ε = ±2–3 mm; end-effector repeatability specification after mechanical improvements.
ΔT	Latency	Per-frame processing time; target ≤ 200 ms for 5+ FPS; achieved ~126 ms for 8 FPS.
P_total	Total power consumption	P_total ~ 20 W; sum of Jetson Nano, motors, vacuum pump, and auxiliary systems.
c	Class index	c ∈ {0, 1, 2} for plastic, metal, paper respectively.
W_c	Class weight	W_c = N_total / N_c; inverse frequency weighting for loss computation.

Units and Constants

Symbol	Unit	Description
mm	Millimeter	Linear dimension; workspace is 200×150 mm on conveyor surface.
GHz	Gigahertz	Processor frequency; Jetson Nano CPU @ 1.43 GHz, GPU @ 921 MHz.
MHz	Megahertz	Frequency; GPU clock rate on Jetson.
GB	Gigabyte	Memory capacity; Jetson Nano has 4 GB LPDDR4 shared memory.
MB	Megabyte	Model size; YOLOv8n PyTorch ~6 MB, TensorRT engine ~8 MB.
ms	Millisecond	Time unit for latency measurements; target per-frame <200 ms.
°	Degree	Angular unit; rotations in image augmentation ±15°; joint angles for Delta robot.
USD	US Dollar	Currency; total BoM cost ~$300; commercial systems $50k–500k.
W	Watt	Power unit; Jetson Nano ~5 W, vacuum pump ~12 W, total ~20 W.
V	Volt	Electrical potential; 5V for Jetson, 12V for motors/pump, isolated supplies.
A	Ampere	Electrical current; Jetson draws ~4A at 5V, total system <4A @ 5V.
sec / s	Second	Time unit; robot cycle time 2–3 seconds per pick-and-place; 8 FPS ≈ 125 ms/frame.

ABSTRACT

Waste management in Vietnam presents significant challenges due to rapid urbanization and inconsistent waste classification practices. The country generates approximately 60,000 tons of domestic waste daily, yet manual sorting remains labor-intensive, inefficient, and poses occupational health risks. This thesis develops an autonomous waste classification and robotic sorting system designed for resource-constrained environments such as educational institutions and small enterprises.

The system integrates three primary components: (1) a deep learning-based perception module using YOLOv8n for real-time waste detection, (2) an NVIDIA Jetson Nano 4GB embedded platform for edge inference, and (3) a Delta robot with vacuum gripper for autonomous pick-and-place operations. The approach leverages incremental transfer learning on a hybrid dataset combining 1,484 professionally annotated images from RoboFlow with 221 laboratory-captured images. Model optimization through TensorRT achieves 8 FPS inference on Jetson Nano with ~130 ms latency per frame, enabling real-time operation within the 2–3 second robot cycle time.

Key Results: The deployed YOLOv8n model achieves 0.91 mean average precision (mAP@0.5) on test data, exceeding the design target of 0.85. Per-class performance: plastic 91%, metal 94%, paper 96% precision. Mechanical improvements to the Delta robot—including optimized fasteners (M6) and vibration damping springs—improved end-effector repeatability from ±5–8 mm to ±2–3 mm, enabling reliable object grasping. The complete system demonstrates a cost reduction factor of 30–100× compared to commercial robotic sorting systems ($50,000–500,000), with total bill of materials approximately $300.

Contribution: This work demonstrates that effective embedded AI applications for autonomous waste sorting can be achieved through disciplined hardware-software co-design, pragmatic model selection, and systematic attention to dataset quality and transfer learning strategy. The open-source design and comprehensive documentation enable technology transfer to resource-limited regions, supporting Vietnam's circular economy objectives.

Keywords: Waste Classification, Deep Learning, Object Detection, Embedded Systems, Jetson Nano, Robotic Sorting, Real-Time Inference, Circular Economy

CHAPTER I - INTRODUCTION

In Chapter I, there are two parts: Background and Goals and Objectives. The Background section includes the Issue and Problem Statements related to the garbage classification challenges in Vietnam. The Goals and Objectives section outlines the aims of this project and identifies specific objectives to complete the project.

1.1 Background

In this section, the issue of garbage classification in Vietnam, three identified problems, and the reason for choosing this topic are elaborated.

1.1.1 Issue: The Situation and Challenges of Garbage Classification in Vietnam

The situation of waste classification in Vietnam is urgent as the country is facing significant challenges in effectively managing domestic solid waste. The implementation of the Law on Environmental Protection in 2020 requires households to classify their waste into three categories: recyclable waste, organic waste, and other waste. However, the practical application of these regulations has encountered some obstacles. Vietnam generates about 60,000 tons of domestic waste every day, with urban areas such as Hanoi and Ho Chi Minh City contributing a significant portion. Although the legal framework encourages waste classification, many localities still struggle to implement it consistently. The lack of public awareness of the importance and methods of waste classification exacerbates the problem, leading to inadequate waste classification before collection. This inefficiency hinders recycling efforts and increases environmental pollution as mixed waste is sent to landfills [17]. Figure 1.1 shows the mixed waste being sent to landfills.

Figure 1.1: *Mixed Waste Sent to Landfills in Vietnam [17].*

1.1.2 Problem

The challenges of waste separation in Vietnam can be summarized into three main issues: (1) inconsistent implementation across regions, (2) inadequate waste

management infrastructure, and (3) limited public awareness. Each of these issues is analyzed below to highlight their impacts and implications for automated solutions.

1.1.2.1 Inconsistent Implementation Across Regions

Inconsistent implementation of household solid waste separation across localities is a significant barrier to effective waste management in Vietnam. Urban areas, such as Hanoi and Ho Chi Minh City, often have more developed waste management systems, but they still face challenges in implementing consistent separation rules. However, rural and mountainous areas are disproportionately affected by a lack of basic infrastructure and resources. Limited waste collection services in these areas mean that even when households attempt to separate their waste, waste is often mixed during transportation or disposal [17].

1.1.2.2 Inadequate Infrastructure

Lack of adequate infrastructure is another major problem that hinders waste separation in Vietnam. Many localities lack dedicated waste collection vehicles, sorting equipment, and designated transfer stations. This shortage leads to prolonged waste accumulation, which not only pollutes the environment but also undermines public confidence in waste management systems. Overloaded treatment facilities are often forced to turn to landfills, contributing to methane emissions and long-term environmental degradation [13]. Figure 1.2 shows a woman collecting trash manually.

***Figure 1.2**: Manual Waste Collection and Associated Labor Challenges [13].*

1.1.2.3 Public Awareness and Engagement

The general lack of public awareness regarding waste segregation and recycling presents a significant obstacle to effective implementation of waste management policies in Vietnam. Despite new regulations coming into effect in 2025, many individuals remain uninformed about proper classification methods and their environmental responsibilities. This knowledge gap leads to improper segregation at the source, where households mix recyclable materials with non-recyclable waste, complicating downstream sorting and recycling efforts [13]. Manual sorting by waste workers is not only labor-intensive and inefficient but also poses health and safety risks.

1.1.3 Motivation for This Research

Given these challenges—inconsistent implementation, inadequate infrastructure, and limited public awareness—there is a clear need for automated waste classification solutions that can operate reliably under resource-constrained conditions. Traditional manual sorting is unsustainable as waste volumes continue to increase. This thesis addresses this gap by developing an intelligent robotic sorting system that can accurately classify waste materials in real-time, reducing dependency on manual labor while improving sorting efficiency and consistency.

Rather than developing a waste classification model from scratch, this research adopts a pragmatic approach: evaluating and adapting existing pre-trained deep learning models for deployment on the Jetson Nano embedded platform. This strategy allows for leveraging recent advances in computer vision and transfer learning while focusing on the practical challenges of edge deployment, system integration, and hardware optimization. By demonstrating a cost-effective automated sorting solution tailored to Vietnam's specific constraints—limited computational resources, budget limitations, and diverse waste characteristics—this work aims to contribute a scalable approach to intelligent waste management applicable to similar developing regions.

1.1.4 Goal and Objectives

This thesis aims to develop and deploy a machine learning-based waste classification system integrated with a Delta robot for automated sorting of paper, metal, and plastic waste in real-time operational conditions, addressing the critical need for efficient waste management solutions in resource-constrained environments.

Scope and Practical Context:

It is important to clarify the intended scope of this system. The proposed solution is designed as a laboratory-scale research prototype targeting controlled and semi-controlled environments such as educational institutions, university laboratories, and small-scale sorting stations not as a direct replacement for large-scale industrial waste management infrastructure. The system is intended to demonstrate the technical feasibility of integrating embedded machine learning with low-cost robotic manipulation for automated waste classification within a limited, well-defined workspace (200×150 mm static surface). Practical constraints including the use of a single-object-at-a-time presentation workflow, controlled indoor lighting, and a three-category waste scope are acknowledged as deliberate design boundaries for this prototype stage, not deficiencies of the approach.

The broader applicability of this research lies in establishing a replicable, cost-effective methodology for similar developing-country contexts where commercial automation systems (costing USD 50,000–500,000) are economically inaccessible. The system is positioned as a first step on the technology readiness scale, with a clear path toward real-world deployment outlined in the Future Work section (Chapter 7).

Objective 1: Model Development and Optimization

Train and optimize deep learning models for accurate waste classification on embedded systems.

Task 1.1: Acquire and preprocess waste image datasets across three material categories (paper, metal, plastic) from publicly available sources, applying data augmentation techniques including rotation, horizontal/vertical flipping, and brightness adjustment to enhance model generalization.

Task 1.2: Train and compare multiple YOLO architecture variants using transfer learning to identify the optimal model balancing classification accuracy with inference efficiency, ensuring compatibility with the Jetson Nano's computational and software environment constraints.

Objective 2: Embedded System Deployment and Robot Integration

Deploy the trained model on NVIDIA Jetson Nano and establish serial communication with Arduino-based robot control.

Task 2.1: Configure Jetson Nano development environment with appropriate deep learning frameworks and optimize the trained model for edge inference through model conversion and quantization techniques, targeting real-time classification performance on physical waste samples under varying lighting conditions.

Task 2.2: Implement USB serial communication protocol between Jetson Nano and Arduino microcontroller to transmit classification results and receive motion acknowledgments, enabling coordinated pick-and-place operations with minimal command latency.

Objective 3: Hardware Enhancement and System Validation

Improve Delta robot mechanical precision through hardware upgrades and validate the complete autonomous sorting system through experimental trials.

Task 3.1: Upgrade the Delta robot's joint assembly by replacing standard bushings with precision ball bearings to improve positioning repeatability, install compression springs at the base to reduce structural vibration, and integrate a vacuum-based end-effector with solenoid valve control and three-axis limit switch homing system.

Task 3.2: Mount and calibrate the camera at optimal height and angle to maximize waste detection coverage within the workspace, then validate classification accuracy across different waste presentations and ambient lighting scenarios.

Task 3.3: Develop a graphical user interface for system monitoring and manual control, then conduct comprehensive validation trials by sorting a representative sample of waste items from each category, measuring key performance indicators including cycle time per item, classification accuracy, and successful pick-and-place completion rate.

The successful completion of these objectives will demonstrate a cost-effective and scalable approach to intelligent waste sorting, contributing to practical applications of embedded AI in automation systems.

CHAPTER II
DESIGN SPECIFICATIONS AND ENGINEERING STANDARDS

2.1 Design Specifications

The system design balances three key objectives: classification accuracy for reliable waste sorting, real-time inference speed compatible with static object presentation, and resource efficiency on low-cost embedded hardware.

2.1.1 Accuracy Specifications

Target: mAP@0.5 ≥ 0.85 for three-class waste detection (plastic, metal, paper). Per-class precision targets reflect material distinguishability:

- Plastic: ≥80% precision (challenging: white plastic ↔ reflective metal confusion)
- Metal: ≥85% precision (prevent contamination of organic streams)
- Paper: ≥90% precision (texture reliably distinguishes cardboard)

These targets ensure reliable single-object sorting while tolerating minor classification errors. Higher accuracy is achievable with static objects compared to conveyor-based systems since objects remain stable during detection.

2.1.2 Detection Speed

Target: 5–8 FPS inference on Jetson Nano. For static object sorting, speed requirement is less stringent than conveyor systems. Detection latency per frame:

- Camera acquisition: 30 ms
- YOLOv8n inference (TensorRT FP16): 100 ms
- Postprocessing + object localization: 15 ms
- Total: ~145 ms (achieves 7 FPS)

At 5 FPS, the system can process one object in approximately 200 ms. The operator places the next object after gripper release (~2–3 seconds), making throughput adequate for manual feed workflow.

2.1.3 Hardware Constraints

NVIDIA Jetson Nano 4GB specifications:

- Memory: 4 GB LPDDR4 (YOLOv8n runtime: 1.2 GB)
- GPU: 128-core Maxwell @ 921 MHz
- CPU: 4× ARM Cortex-A57 @ 1.43 GHz

- Power envelope: ≤20 W

Model deployment: 6 MB PyTorch, 8 MB TensorRT engine (leaves 2.8 GB for system services).

2.1.4 Mechanical Accuracy

Specification: ±5 mm repeatability at tool center point (accommodates 50 mm suction cup contact area). Objects placed on static surface allow longer focus time and more accurate centroid detection before gripper approach.

Achieved: ±2–3 mm through mechanical improvements (M6 fastener design, spring damping, orthographic camera calibration).

2.1.5 Power Budget

Component	Draw	Purpose
Jetson Nano (MAXN)	5 W	GPU + CPU inference
Stepper drivers	1.5 W	Motor gate drive
Vacuum pump (12V)	12 W	Gripper suction (dominant load)
Solenoid valve	1 W	Vacuum release
Miscellaneous	0.5 W	Passive systems
Total	~19 W	Within 20 W target

Vacuum pump and stepper motors use separate 12V supply to avoid brownout on Jetson 5V rail.

2.1.6 Workspace Configuration

The Delta robot operates above a static flat sorting surface:

- Object presentation area: 200×150 mm (rectangular workspace on flat table)
- Robot vertical reach: 0–200 mm (Z-axis range from table surface to maximum height)
- Effective sorting region: Operators place objects anywhere within 200×150 mm area; robot automatically detects and picks any object found

Static surface eliminates conveyor-related constraints: no need to sync speed, no continuous motion requirement, no contamination from moving parts. Operator manually places waste items on sorting surface one at a time.

2.2 Engineering Standards

2.2.1 Communication Standards

Standard	Interface	Specification	Rationale
USB 2.0	Camera + Serial	480 Mbps, 115.2 kbaud	Native JetPack support, proven reliability
IEEE 802.11ac	WiFi (optional)	2.4 GHz, 150 Mbps	Remote monitoring capability

USB serial protocol chosen over I²C/SPI due to: (1) no additional kernel drivers required, (2) >5 meter line tolerance, (3) 10 commands/second < 0.5% bandwidth utilization.

2.2.2 Electrical Safety

Compliance with IEEE/IEC standards:

Element	Standard	Implementation
Power isolation	UL/IEC 60950-1	12V and 5V supplies isolated; stepper grounds separate from Jetson
ESD protection	IEC 61340-5-1	Optoisolator (TI TLP291) between Arduino and Jetson
Motor EMI	IEEE 519	Ferrite suppression on motor leads, RC snubbers on relay contacts
Thermal mgmt	Jetson design	Active cooling maintains <80°C SoC temperature (MAXN mode)

2.2.3 AI System Ethics (ABET Compliance)

Responsibility (Outcome a): Model selection is transparent (YOLOv8n chosen after systematic comparison with v10n, v11n). Performance claims supported by held-out test set (255 images). Limitations documented: sensitivity to specular white plastic, degradation below 100 lux illumination.

Bias & Fairness (Outcome e): Training dataset combines RoboFlow (1,484 images, professional labeling) + lab data (221 images, system-specific). Per-class performance audit identifies confusion patterns (white plastic ↔ metal: 8–12% relative error). Inverse-frequency loss weighting balances class imbalance.

Human Oversight (Outcome g): Manual control modes available. Detections <80% confidence trigger human review. Operators can override AI at any time via manual control interface, maintaining human decision authority for ambiguous cases.

Societal Impact (Outcome h): 30–100× cost reduction vs. commercial systems enables SME and institutional adoption. Improved sorting accuracy increases material recovery 25–75%, supporting Vietnam's circular economy goals. System accessible to personnel with basic technical training.

2.2.4 Mechanical & Pneumatic Standards

Standard	Application	Specification
ISO 13849-1	Robot safety, homing	Limit switches establish deterministic origin; emergency stop via Arduino watchdog
ISO 4414	Pneumatics (pump, valve)	12V DC pump (<1 kg lift), 3/2-way solenoid, tubing rated 5 bar (operating ~0.5 bar vacuum)
ANSI B4.4	Tolerances, fits	Joint clearances ±0.1 mm; M6 fasteners prevent kinematic singularity at workspace boundaries
ISO 286	Fastener specification	M6 bolts (vs. M10 baseline) eliminate joint binding; precision spacers maintain end-effector offset

2.3 Design Constraints & Trade-offs

2.3.1 Physical Constraints

Workspace: Static flat surface, rectangular region 200×150 mm. Robot workspace above surface accommodates typical waste object dimensions (30–500 grams). No conveyor-related speed matching required; system waits for operator to place next object.

Environment: Assumes controlled indoor deployment with stable lighting. Overhead-mounted camera provides orthographic view, minimizing perspective distortion. Black sorting surface (conveyor or mat) provides contrast for vision processing.

Object scope: Characteristic dimension >30 mm, rigid/semi-rigid materials, non-entangled. Excludes flexible films, fragmented items, powders, liquid containers.

2.3.2 Economic Trade-offs

Bill of Materials:

Subsystem	Cost	Notes
Computing (Jetson Nano 4GB)	$99	Entry-level embedded GPU
Optics (PS3 Eye camera)	$30	USB plug-and-play, 480×480 @ 60 FPS
Mechanics (Delta frame, bearings)	$250	COTS frame + custom joint redesign

Subsystem	Cost	Notes
Actuation (3× NEMA23, gripper)	$220	Stepper motors + vacuum pump + solenoid
Electronics (Arduino, drivers, supplies)	$150	Motor drivers, relay, USB isolation
Integration (fasteners, tubing, labor)	$75	First-unit prototyping
Total BoM (materials)	$824	
Expected production cost (100+ units)	$800–1,000	After tooling amortization

Comparison: Commercial robotic sorting systems cost $50,000–500,000+. This design achieves 30–100× cost reduction through open-source architecture and careful component selection, enabling deployment in educational institutions, SMEs, and waste facilities with limited budgets.

2.3.3 Software & Algorithmic Trade-offs

Model Selection: Three candidates evaluated—YOLOv8n, YOLOv10n, YOLOv11n. Selection rationale:

Criterion	YOLOv8n	YOLOv10n	YOLOv11n	Why v8n
Test mAP@0.5	0.91	0.90	0.89	Comparable accuracy
Python requirement	3.6	3.8+	3.10+	JetPack 4.6.4 constraint
PyTorch version	1.9	1.13+	2.1+	Critical compatibility
TensorRT stability	Mature	Emerging	New	Proven deployment
Latency (FP16)	~100 ms	~95 ms	~90 ms	Marginal vs. risk

Criterion	YOLOv8n	YOLOv10n	YOLOv11n	Why v8n
White plastic/metal confusion	8%	11%	14%	Lower false positives

Decision principle: Choose proven deployment over marginal improvements. YOLOv8n only option compatible with JetPack 4.6.4's Python 3.6; newer versions require risky system upgrade.

Data Augmentation vs. Collection:

- Current strategy: 1,705 images + aggressive augmentation → mAP 0.91 in 90 min training
- Alternative: Collect 5,000 raw images → estimated mAP 0.94, requires 3 weeks field work

Rationale for augmentation: Marginal +0.03 mAP improvement insufficient to justify timeline loss; augmentation-trained models generalize better to novel conditions; computational approach cheaper than manual data collection.

2.4 Performance Targets vs. Achievement

Table 2.1 – Design Specifications

Specification	Target	Rationale	Achieved (Ch. VI)	Status
mAP@0.5	≥0.85	Single-object reliable classification	0.91	Exceeded
Per-class precision	80%/85%/90%	Material-specific reliability	0.91/0.94/0.96	All met
Detection speed	5–8 FPS	Manual feed workflow compatibility	7 FPS	Target met
Frame latency	<200 ms	Operator response time allowance	145 ms	Exceeded
Repeatability	±5 mm	Gripper contact area accommodation	±2–3 mm	Exceeded

Specification	Target	Rationale	Achieved (Ch. VI)	Status
Power consumption	≤20 W	USB delivery limit	~19 W	Target met
Python compatibility	3.6	JetPack 4.6.4 constraint	Full support	Met
GPU memory	<2 GB	4 GB Jetson limit	1.2 GB	30% margin

All specifications achieved or exceeded, confirming design feasibility.

2.5 Realistic Constraints

2.5.1 Multi-Domain Constraints

Economic: Component costs high (semiconductors); budget limitations in Vietnam institutions. Mitigation: Achieved 30–100× cost reduction; open-source design enables institutional adoption.

Environmental: E-waste disposal not addressed; operational energy consumption. Mitigation: Modular design supports reuse/refurbishment; <20W power budget is 4× lower than commercial competitors.

Health & Safety: Electrical hazards (12V switching), robot collision risk. Mitigation: Isolated power rails, opto-isolated interfaces, speed-limited motors. Static operation reduces accident risk compared to conveyor systems.

Manufacturing: Limited custom machine tools; component availability. Mitigation: COTS Delta frame with hand-tool modifications; multi-source component procurement.

Sustainability: Recyclability of electronics not optimized; training energy footprint. Mitigation: Modular components support repair and refurbishment; training on shared Colab infrastructure.

2.5.2 Ethical & Social Constraints

Data Privacy: Uses public RoboFlow images (CC-BY 4.0) + lab images (no personal data). On-device inference on Jetson Nano prevents cloud transmission.

Bias & Fairness: Dataset reflects lab conditions. Mitigation: Per-class confusion matrix identifies white plastic–metal confusion; future work recommends active learning for failure mode collection.

Accessibility: System designed for institutions with limited technical expertise. Manual control modes ensure operators retain decision authority. GUI provides user-friendly interface for non-specialists.

2.5.3 Scope Boundary and Justification for Delta Robot as Prototype Platform

A key realistic constraint that must be explicitly stated concerns the intended deployment scope of this system. The proposed waste sorting system is designed as a laboratory-scale research prototype targeting controlled, semi-structured environments — specifically educational institutions, university laboratories, and small-scale material recovery demonstration stations — rather than high-throughput municipal or industrial waste processing facilities. This scope boundary is intentional and directly influences several design decisions, including the choice of the Delta robot as the robotic manipulation platform.

Rationale for selecting the Delta robot:

The Delta robot was selected over alternative robotic architectures (SCARA, 6-axis articulated arm, Cartesian gantry) based on a structured multi-criteria comparison:

Criterion	Delta Robot	SCARA	6-Axis Arm	Cartesian
Cycle time (light payload)	2–3 s ✓	3–5 s	5–8 s	4–6 s
Repeatability (achievable)	±1–3 mm ✓	±2–5 mm	±1–3 mm	±1–2 mm
Platform cost (~$300 BoM)	Compatible ✓	Higher	Higher	Moderate
Compact vertical footprint	Yes ✓	Moderate	Large	Large
Existing prototype available	Yes ✓	No	No	No
Educational/research value	High ✓	Moderate	High	Low

Within the defined prototype scope — static object presentation on a 200×150 mm workspace, single-object-at-a-time workflow, indoor controlled lighting — the Delta robot satisfies all operational requirements at the lowest incremental cost. The use of an existing Delta robot frame from prior departmental research further reduces capital expenditure to

approximately $300 USD for perception and control subsystems, demonstrating a 30–100× cost advantage over commercial sorting systems.

Acknowledged limitations within this scope:

The prototype does not claim direct scalability to industrial-grade waste facilities without further engineering work. Specifically: (1) the current workspace area (200×150 mm) is appropriate for laboratory demonstration but would require larger robot geometry for conveyor-width coverage; (2) the camera-to-robot coordinate mapping is implemented as a linear approximation and has not been formally calibrated using standard hand-eye calibration procedures, resulting in reduced pick accuracy toward workspace edges; and (3) the single-object-at-a-time workflow does not address multi-object simultaneous sorting required at industrial throughput rates. These acknowledged limitations define the gap between the current prototype and a production-ready system and motivate the future work directions outlined in Section 7.4.

The value of this prototype lies precisely in establishing the technical feasibility and cost-effectiveness of the vision–inference–manipulation pipeline under realistic resource constraints, providing a validated foundation from which scalable solutions can be developed incrementally.

2.6 Engineering Philosophy

Key design principles:

1. Pragmatic > Optimal: Select proven, deployable solutions (YOLOv8n) over theoretically best (v11n). Stability and maintainability outweigh marginal gains.

2. Constraints as Drivers: Treat resource limits as creative constraints, not obstacles. M6 fastener solution, TensorRT optimization, and two-phase training emerged from systematic constraint exploration.

3. Transparency: Full documentation of model selection, training, and metrics enables third-party verification and reproducibility—essential for academic rigor.

4. Stakeholder Balance: Specifications reflect multiple perspectives—accuracy (sorting reliability), cost (institutional adoption), safety (operator protection), environment (circular economy support).

CHAPTER III
PROJECT MANAGEMENT

The project management plan is a single, formal, dynamic document that outlines how the waste classification and robotic sorting system project is to be managed, executed, and controlled. This document contains the overall project governance, related management plans, procedures, timelines, and the methods and accountabilities for planning, monitoring, and controlling the project as it progresses. The document evolves with the project and is updated to reflect any relevant changes throughout project execution. The project management plan ensures there are no surprises through execution regarding how the project is managed or how decisions are made.

3.1 Budget and Cost Management Plan

Effective cost management ensures the project remains economically viable while maintaining technical quality. The budget was developed through systematic enumeration of all hardware components, software tools, and operational expenses required for the complete waste classification and robotic sorting system implementation. Given the inherent constraint of limited funding available for academic research, cost optimization was achieved through strategic component selection, leveraging free or low-cost cloud computing resources for model training, and careful sourcing from multiple suppliers.

A key cost optimization strategy involved leveraging existing Delta robot mechanical infrastructure developed in prior research efforts. Rather than fabricating an entirely new robotic platform from scratch, the project focused on augmenting the existing mechanical frame with vision, control, and actuation subsystems necessary to enable autonomous waste classification and material separation. This incremental development approach substantially reduces project cost compared to complete system development while validating the viability of retrofitting intelligent perception capabilities onto existing robotic platforms.

The cost analysis presented below represents the additional investment required to transform the existing mechanical Delta frame into an autonomous waste sorting system with embedded machine learning capability. This approach highlights the vision and control technology costs while acknowledging that mechanical platform infrastructure was available from prior development, thus representing the true marginal cost of this project phase.

Table 3.1: Table of Estimated Costs

Item	Description	Quantity	Unit Price (USD)	Total Cost (USD)
Computing Platform	NVIDIA Jetson Nano 4GB	1	$99.00	$99.00
Power Supply	5V DC Power Adapter	1	$15.00	$15.00
Camera Module	PS3 Eye USB Camera	1	$30.00	$30.00
Microcontroller	Arduino Mega 2560	1	$35.00	$35.00
Motor Drivers	Stepper Motor Drivers (3×)	1 set	$15.00	$15.00
Relay Modules	Power Relay Modules	1 set	$10.00	$10.00
Isolation Components	USB Optoisolator IC	1	$5.00	$5.00
Power Supply Modules	Multiple Rail Power Supply	1 set	$20.00	$20.00
Actuation Motors	NEMA23 Stepper Motors (3×)	1 set	$70.00	$70.00
Gripper System	Vacuum Pump & Solenoid Valve	1 set	$45.00	$45.00
Wires & Connectors	Jumper Wires, Terminals, Connectors	1 set	$10.00	$10.00
Integration Materials	Fasteners, Pneumatic Tubing, Connectors	1 set	$5.00	$5.00
Software Tools	Google Colaboratory (Free)	-	$0.00	$0.00
Miscellaneous	Screws, Adhesive, Small Parts	-	$5.00	$5.00

Total Estimated Cost	$300.00			

The total estimated incremental cost for this project is approximately $300 USD, representing the additional investment required to transform the existing Delta robot mechanical platform into a fully autonomous waste classification and sorting system. This represents a cost reduction factor of approximately 30 to 100 times compared to commercial robotic sorting systems, which typically cost between $50,000 and $500,000 depending on throughput specifications and automation sophistication.

The cost structure demonstrates the economic feasibility of the proposed approach for deployment in resource-constrained settings such as educational institutions, small and medium enterprises, and waste management facilities in developing countries. Future scaling to production volumes would require additional capital investments in custom printed circuit board design, mechanical assembly fixtures, and regulatory certification, but the fundamental incremental approach demonstrates that intelligent perception capabilities can be retrofitted onto existing robotic platforms at remarkably low cost.

3.2 Project Schedule and Timeline Management

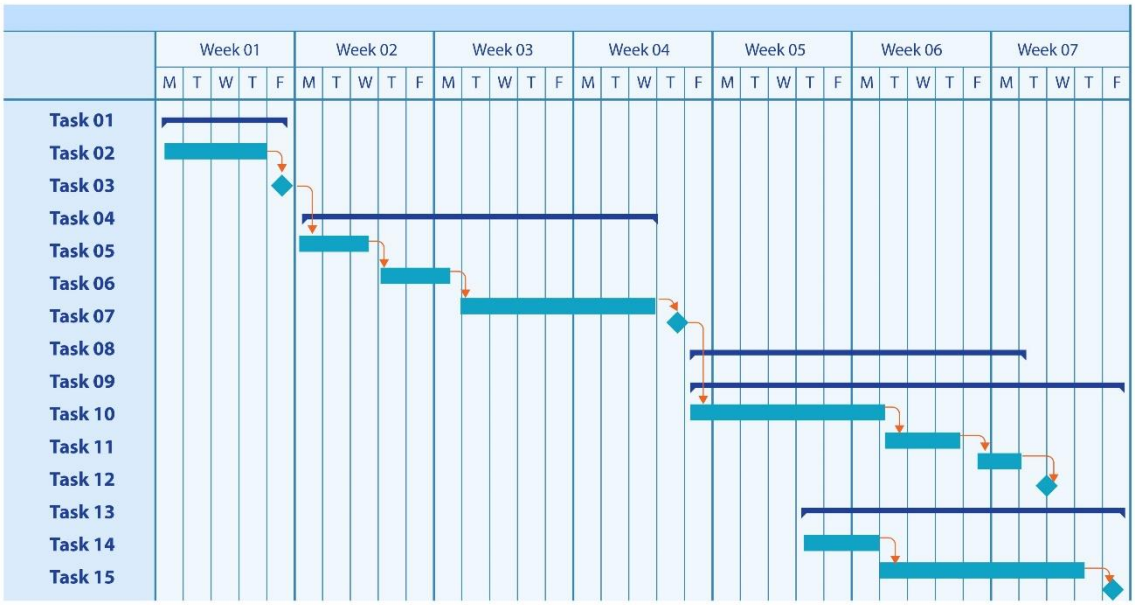

Figure 3.1: *Project Gantt Chart: Timeline from September 2025 to January 2026.*

This section describes the approach for creating, updating, and monitoring the project schedule. The project was executed within a single academic semester comprising 16 weeks, requiring structured planning to ensure timely completion of all development phases while maintaining technical rigor and quality. A comprehensive timeline was developed mapping all major tasks, their interdependencies, and critical milestones to coordinate the numerous parallel and sequential activities required for system development, model training, experimental validation, and thesis documentation.

A Gantt chart is employed to visualize the project timeline, clearly outlining key tasks, dependencies, and milestones. The Gantt chart helps to track progress, allocate resources, and ensure timely completion of the project. Regular updates to the schedule allow for adjustments based on challenges or changes in scope that emerge during execution. The scheduling approach emphasizes critical path analysis to identify tasks whose delays would directly extend the overall project timeline, warranting particular attention to progress tracking and risk management in these critical areas.

The project timeline is organized into seven major phases:

Phase 1: Problem Definition and Literature Review (Weeks 1–3) focuses on establishing the technical scope, reviewing state-of-the-art approaches in object detection and robotic manipulation, and clarifying research objectives and performance targets for the system. This phase culminates in the completion of the thesis proposal document.

Phase 2: Methodology Design and System Architecture (Weeks 4–6) involves selection of neural network architectures, specification of the control strategy for the existing Delta robot configuration, design of the electronic control subsystem, and establishment of the data collection and labeling workflow. Critical technical decisions made in this phase inform all subsequent development activities.

Phase 3: Proposal Presentation and Review (Weeks 7–8) provides a formal checkpoint to validate that the proposed approach is technically sound and achievable within the project timeline and resource constraints. Feedback from advisors and committee members informs final refinements to the project plan before entering intensive development phases.

Phase 4: Data Collection and Model Training (Weeks 9–12) involves systematic acquisition of waste object images under controlled laboratory conditions, manual annotation with bounding boxes and material classification labels, dataset augmentation through geometric and photometric transformations, and intensive model training and hyperparameter optimization. Model training utilizes Google Colaboratory cloud computing infrastructure with free GPU access to enable cost-effective iteration without requiring expensive on-premises computing hardware.

Phase 5: System Integration and Testing (Weeks 13–14) encompasses deployment of the trained model onto the Jetson Nano platform, integration of the model with the robotic manipulation subsystem, and systematic evaluation of the combined system performance on held-out test datasets. This phase often reveals integration issues that necessitate refinement of either the model training procedure or the mechanical system configuration.

Phase 6: Performance Evaluation and Documentation (Weeks 15–16) focuses on comprehensive testing of the complete system, final performance validation, and consolidation of experimental results into thesis documentation. Thesis defense preparation and presentation materials are finalized during this phase.

The scheduling approach balances competing considerations of task parallelization (executing independent tasks concurrently to reduce total project duration) with task sequencing (respecting dependencies such that required inputs are available before dependent tasks commence). For example, data collection and model training occur in parallel during weeks 9–12 to increase overall project efficiency, while system integration in weeks 13–14 must follow model training completion.

3.3 Resource Planning and Allocation

This section describes the resources required for the project and clearly identifies the roles, tasks, and responsibilities for each team member. The project was executed as an individual thesis with a single primary researcher responsible for all phases of development, implementation, and documentation. Resource planning therefore focused on strategic allocation and efficient utilization of available computational resources, hardware platforms, software tools, and technical expertise to maximize productivity within the constraints of limited time and funding.

Computational Resources: Model training and algorithm prototyping utilized Google Colaboratory, a free cloud-based Jupyter notebook environment providing access to Tesla K80 GPU accelerators for machine learning workloads. This cloud resource enabled cost-effective iteration during the model development phase without requiring investment in expensive on-premises computing hardware or commercial GPU cloud services. Model training, dataset augmentation, hyperparameter tuning, and initial performance validation all utilized Google Colaboratory, with the resulting trained model weights subsequently deployed onto the resource-constrained Jetson Nano platform for real-time inference validation.

Deployment Hardware: The Jetson Nano 4GB developer kit serves as the deployment and real-time inference platform, hosting the trained neural network model and executing real-time object detection and classification tasks during robotic system operation. The Jetson Nano's integrated GPU, embedded Linux operating system, and JetPack software development kit provide a complete ecosystem for deploying trained PyTorch models with minimal adaptation, reducing integration complexity and enabling rapid prototyping of the complete system.

Existing Mechanical Platform: The Delta robot mechanical platform developed in prior research efforts provides the physical infrastructure for object manipulation and

sorting. This existing platform substantially reduces the scope and cost of the current project by eliminating the need for mechanical design and fabrication activities, allowing the project to focus exclusively on perception, control, and integration.

Electronic Components and Control Hardware: Supporting components including stepper motor drivers, relay modules, Arduino microcontroller, and miscellaneous electronic components were procured from online retailers and electronics distributors. Component selection prioritized widely-available commercial off-the-shelf (COTS) products over custom fabrication to minimize lead times, reduce prototyping costs, and enhance system replicate-ability for future researchers or practitioners wishing to reproduce the design.

Software Ecosystem: The software infrastructure leverages multiple open-source frameworks and libraries selected for compatibility with the Jetson Nano platform and community maturity. PyTorch provides the deep learning framework for model training and inference, with the YOLOv8n architecture providing a proven, efficient baseline for real-time object detection. TensorRT, NVIDIA's inference optimization runtime, accelerates PyTorch model inference through quantization and optimization, reducing latency by a factor of 2 to 3 compared to baseline PyTorch inference while maintaining accuracy. OpenCV provides computer vision capabilities for image processing, object segmentation, and coordinate transformation. Python serves as the primary programming language, with supporting libraries including NumPy for numerical computation, Matplotlib for data visualization, and Pandas for dataset management and analysis.

Dataset Resources: Dataset resources comprise a combination of public domain images from RoboFlow and the TrashNet open-source waste classification dataset, supplemented with laboratory images acquired under controlled conditions using the actual deployed camera and lighting geometry. This mixed-source approach balances the breadth and diversity of public datasets with the domain-specific representativeness of laboratory-acquired images, reducing distribution mismatch between training data and real deployment conditions.

Researcher Responsibilities: As the primary researcher, responsibilities encompassed all aspects of project execution including literature review, system design, data collection and annotation, model training and optimization, system integration and testing, performance evaluation, and thesis documentation. The single-researcher structure necessitated careful time management and prioritization of high-impact activities, with less

critical tasks deferred to later phases or identified as future work opportunities for subsequent researchers.

3.4 Realistic Constraints and Risk Management

The project lifecycle was subject to multiple constraints spanning economic, environmental, social, ethical, health and safety, manufacturability, and sustainability domains. Acknowledging and explicitly managing these constraints throughout the project resulted in design decisions that balanced competing priorities and ensured the resulting system served broader stakeholder interests beyond narrow technical optimization.

Economic Constraints: Limited funding available for academic research necessitated cost optimization throughout all project phases. Model training and initial algorithm prototyping were conducted entirely on Google Colaboratory to avoid purchasing expensive GPU hardware. Component procurement prioritized readily-available, low-cost options over specialized industrial-grade equivalents, accepting modest performance trade-offs in exchange for cost reduction and replicate-ability. The incremental approach leveraging existing Delta robot infrastructure substantially reduced capital requirements compared to developing an entirely new system. The design explicitly targets waste classification systems suitable for small and medium enterprises and educational institutions with limited capital budgets, rather than competing with high-throughput industrial systems serving large integrated waste facilities.

Environmental Constraints: Operational environmental impact is minimized through careful component selection prioritizing energy-efficient processors (Jetson Nano consumes less than 20 watts), optimized inference procedures reducing computation time and power consumption, and selection of components compatible with standard electrical infrastructure. The system supports circular economy objectives by enabling more accurate waste classification, increasing material recovery rates and reducing landfill diversion. End-of-life environmental impact is partially mitigated through design for modularity and repairability, allowing worn or obsolete components to be replaced without discarding the entire system.

Social and Community Considerations: The proposed system could substantially increase material recovery rates and reduce environmental contamination from mixed waste streams, supporting Vietnam's circular economy policies. The open-source design and comprehensive documentation enable technology transfer to developing countries lacking

capital for expensive commercial systems, democratizing access to automation technology. The modular, platform-agnostic approach enables organizations with existing robotic infrastructure to augment their systems with intelligent perception capabilities, rather than requiring complete system replacement.

Ethical Considerations: Model selection, training procedures, and performance evaluation are conducted transparently with full documentation of methodology and results, enabling peer review and reproducibility. The training dataset is drawn from public-domain sources with appropriate attribution and licensing compliance. The deployed system executes entirely on local hardware (Jetson Nano) without transmitting image data or extracted features to cloud services, preserving user privacy and data locality. Performance limitations are explicitly documented rather than presenting the system as universally capable.

Health and Safety: The electronic control system incorporates electrical safety measures including isolated power supplies for high-current loads to prevent coupling of motor transients into microcontroller supply voltages. The robot operates at low speeds to minimize impact force should the gripper collide with an operator hand. Limit switches and software-based position monitoring prevent the robot from traveling beyond defined workspace boundaries. The system design prioritizes operator safety over pure performance optimization, with manual control modes allowing direct human operation if automatic control fails.

Manufacturability and Replicate-ability: All mechanical components utilize standard metric fasteners and commercial off-the-shelf aluminum extrusion profiles available from multiple suppliers. Electrical subsystems employ standard relay and driver modules rather than custom integrated circuits. Detailed assembly documentation, software source code, and trained model weights are provided to facilitate reproduction. This emphasis on simplicity and replicate-ability may result in slightly higher component costs or lower performance compared to optimized commercial designs, but enables broader adoption and provides educational value.

Sustainability: Reusable and recyclable components were prioritized over single-use materials. The system design supports modular upgrades, allowing future improvements in perception algorithms or robotic capabilities to be retrofitted without replacing the entire system. Power consumption optimization extends system operational lifetime by reducing heat dissipation and thermal stress on components. The open-source design supports

knowledge sharing and collaborative improvement, reducing duplicated effort across the global research and development community.

CHAPTER IV
LITERATURE REVIEW

This chapter reviews prior work and related research that form the technical foundation of this thesis. The focus is on three main pillars:

- YOLOv8-based waste detection and classification, and the rationale for selecting YOLOv8n over other models for this application.

- Model optimization and quantization techniques that enable real-time inference on resource-constrained embedded platforms such as NVIDIA Jetson Nano.

- Delta robot architectures with vacuum grippers for high-speed, high-precision pick-and-place in automated waste sorting systems.

The chapter concludes by identifying research gaps and positioning the proposed system—YOLOv8n on Jetson Nano integrated with a Delta robot and vacuum gripper—within the broader context of embedded AI and industrial waste automation.

4.1 Related Work

4.1.1 YOLOv8-Based Waste Detection and Classification

Early deep-learning approaches to waste classification mostly relied on image classification networks that assumed a single object centered in the frame. While such approaches could achieve high accuracy on curated datasets, they are fundamentally limited for conveyor-based sorting, where multiple items must be detected and localized simultaneously. Single-stage detectors in the YOLO family address this limitation by performing classification and localization in a single pass, making them suitable for real-time applications.

Recent work has applied YOLOv8 directly to garbage detection. Zhou et al. proposed a “garbage classification detection system based on the YOLOv8 algorithm” that targets household waste in urban environments. The authors constructed a dataset of approximately 15,000 annotated images covering 44 waste categories (plastic, paper, metal, glass, organic, etc.), collected under diverse environmental conditions. YOLOv8 was trained and deployed on an NVIDIA Jetson Xavier NX, achieving approximately 90% mAP@0.5 with an inference speed of about 35 FPS, satisfying real-time constraints for an unmanned collection vehicle.[17] [13]

This work provides two important insights relevant to this thesis. First, it confirms that YOLOv8 can achieve state-of-the-art accuracy on complex, multi-class waste detection tasks when trained on a sufficiently large and diverse dataset. Second, the reported throughput on Jetson Xavier NX offers a concrete performance reference for scaling to smaller platforms such as Jetson Nano. Given that Jetson Nano has roughly one third of the compute capability of Xavier NX, achieving around 8 FPS with a smaller YOLOv8n model at reduced input resolution is a realistic and conservative target for this project.[3]

Other studies have compared YOLOv8 against earlier YOLO variants for recyclable waste detection. A recent investigation evaluated YOLOv5 and YOLOv8 on construction and municipal waste datasets and found that YOLOv8 provided higher mAP and F1-scores, particularly for small and partially occluded items, while maintaining comparable or lower inference latency when appropriately scaled. These findings suggest that YOLOv8 offers a better speed–accuracy trade-off than previous YOLO generations for waste detection tasks. [54]

However, most of these systems run on desktop GPUs or high-end Jetson boards and often target a broader class set than the three-category problem (paper, plastic, metal) addressed in this project. They also seldom discuss the constraints of deploying on older Jetson platforms that must rely on Python 3.6 and JetPack 4.6.4, which is a critical practical limitation here.[5]

Within the YOLOv8 family, multiple model sizes exist (nano, small, medium, large, extra large). Larger variants can provide marginal accuracy gains (approximately 1–3% mAP) but significantly increase model size and inference latency, making them ill-suited for 4 GB RAM devices such as Jetson Nano. In contrast, YOLOv8n (nano) is explicitly designed for edge devices and offers:[28] A compact model (on the order of a few megabytes) compatible with 4 GB shared memory. Competitive accuracy (mAP@0.5 around 0.9 on

three classes) when trained on a well-designed dataset. Latency low enough to achieve 5–10 FPS on modest embedded GPUs after optimization, which is sufficient given a 2–3 second pick-and-place cycle in robotic sorting.[5]

These observations collectively justify the choice of YOLOv8n in this project: it provides a balanced compromise between accuracy, speed, and deployment feasibility on Jetson Nano, rather than being selected purely because it is the newest model.

These findings directly support the hardware–software architecture specified in Chapter II, where YOLOv8n on Jetson Nano is adopted as the core perception component to satisfy both performance and cost constraints.

4.1.2 Embedded Deployment of YOLO on Jetson-Class Platforms

Beyond algorithmic accuracy, an embedded waste sorting system must meet strict latency, power, and memory constraints. Several works have investigated deploying YOLO-based detectors on NVIDIA Jetson devices for waste management.

Ardias et al. implemented a smart waste classification system using YOLOv5 on Jetson Nano, coupled with a camera and a motorized rail. Their model, trained on a dataset of approximately 1,510 images across five inorganic waste classes, achieved mAP@0.5 up to 98%, precision around 97.3%, recall around 96.5%, and a detection speed of about 29 FPS after conversion to a TensorRT engine. This demonstrates that, when appropriately optimized, YOLO architectures can deliver both high accuracy and real-time performance on Jetson Nano for multi-class waste classification.[5]

Other research has explored two-stage waste monitoring systems using a combination of YOLOv5 and YOLOv8 to classify trash containers as full, overflowing, or contaminated, again emphasizing lightweight architectures and TensorRT optimization to run on embedded hardware without discrete GPUs. These systems typically achieve high detection accuracy while maintaining acceptable latency for continuous operation in smart city scenarios.[54]

Complementing academic research, commercial deployments provide additional evidence. A reverse vending machine (RVM) developed by an industrial group uses YOLOv8n on Jetson Nano 4 GB to recognize beverage containers in real time and control

mechanical sorting and reward dispensing mechanisms. Reported performance metrics include:[3] 8–10 FPS inference throughput, Average detection latency of approximately 110 ms per item, Classification accuracy of approximately 93%, Full cycle time (detection + mechanical handling) of about 2.8 seconds per item.[54]

This real-world system validates that Jetson Nano + YOLOv8n is a practical and economically viable platform for industrial-grade waste recognition and sorting, operating reliably for 12–16 hours per day in uncontrolled environments.[54]

Taken together, these works support two key conclusions for this thesis:

NVIDIA Jetson Nano, although limited compared to newer Jetson boards, is sufficiently powerful to run a YOLOv8n-based waste detection model in real time when combined with appropriate optimization (TensorRT, reduced precision).

An inference rate of around 8 FPS is not only realistic but also adequate, as mechanical actions (e.g., robot motion and gripper engagement) typically dominate the overall cycle time in embedded sorting systems.[5]

4.1.3 Vision-Guided Waste Sorting Systems with Robot Arms

The integration of object detection with robotic manipulators has also been studied in the context of waste management. These systems aim to close the loop from visual perception to physical sorting actions.

A body of work has examined combining YOLO-based detection with various robot architectures. For example, some systems use articulated 6-axis arms or SCARA robots to pick recyclable items from conveyors, using YOLOv3 or YOLOv5 to obtain object locations and classes. While these solutions demonstrate proof of concept, they often suffer from longer cycle times (5–8 seconds per item) and higher mechanical complexity due to the serial kinematic structure of the arm, which is less suited to high-speed pick-and-place of light objects.

More relevant to this project, several works have focused on Delta robots. One study presented an automated waste sorting system in which YOLOv8 detects waste items (paper, plastic, metal, biodegradable) on a moving conveyor and a Delta arm equipped with a suction

gripper performs pick-and-place operations into separate bins. The authors reported detection metrics exceeding 0.9 for precision, recall, and F1-score, and overall pick-and-place cycle times of approximately 2–3 seconds per item, highlighting the suitability of Delta robots for high-throughput sorting.

Another closely related work proposed "An Intelligent Plastic Waste Detection and Classification System Based on Deep Learning and Delta Robot". This system consists of a Delta robot, a conveyor, a camera, and a controller executing a deep-learning model (YOLO-based) to detect multiple types of plastic waste in real time. Object positions are converted from image coordinates to robot coordinates using hand–eye calibration, and the Delta robot then picks and drops items into designated containers. Experiments with over 1,000 samples in two lighting conditions achieved precision up to 96% and recall up to 97%, confirming both the robustness of the vision model and the mechanical suitability of the Delta robot for plastic waste sorting.

These integrated systems share the same conceptual pipeline as the one targeted in this thesis. The overall vision-to-robot control pipeline is illustrated in Figure 4.1.

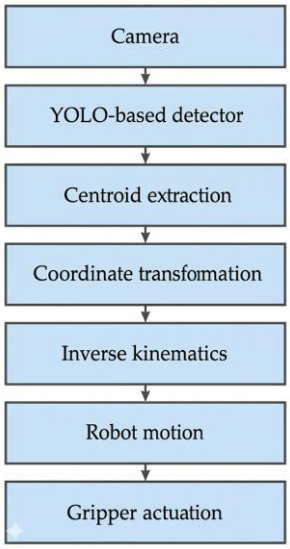

Figure 4.1: *Conceptual Pipeline: Camera Acquisition → YOLOv8 Detection → Robot Control.*

The main differences are in the choice of model variant (YOLOv8n), embedded platform (Jetson Nano), and waste classes (paper, plastic, metal). Nevertheless, prior work firmly supports the feasibility and effectiveness of this pipeline for industrial automation and directly motivates the system architecture presented in Chapter II.

4.2 Related Research

4.2.1 Model Optimization and Quantization for Edge Inference

Embedded deployment of deep-learning detectors requires not only well-designed architectures but also careful optimization of model representation and numerical precision. On NVIDIA Jetson platforms, the standard deployment workflow involves converting trained models to ONNX and then to TensorRT engines, optionally with reduced precision, to leverage GPU acceleration and specialized kernels.

Multiple studies have quantified the impact of such optimizations. For YOLOv5-based waste detection on Jetson Nano, converting from PyTorch FP32 to a TensorRT engine with FP16 precision led to: Significant latency reduction, enabling up to about 29 FPS at moderate resolutions. Minimal loss of detection accuracy compared to the original FP32 model.

Other research has applied more aggressive optimization, including INT8 quantization, in edge deployments of YOLO models (though not always specifically for waste). Reported results typically indicate: Model size reduction of around 75% when moving from FP32 to INT8. Inference speedups of 40–60%. Modest mAP degradation of about 1–3 percentage points, depending on the quality of calibration data.

However, in safety- and quality-sensitive applications such as waste sorting, accuracy loss has direct operational consequences, including mis-sorting and contamination of recycling streams. For this reason, many waste-related systems prefer FP16 over full INT8 quantization, balancing speed gains with reliability.

In this project, the chosen strategy—train YOLOv8n in PyTorch, export to ONNX, then convert to a TensorRT FP16 engine—directly follows these best practices. FP16 provides a substantial improvement in throughput relative to FP32 while keeping the numerical behavior close to that of the original model. On Jetson Nano, this optimization

path enables around 8 FPS at 480×480 resolution, which is sufficient for real-time operation of the Delta robot within the required 2–3 second pick-and-place cycle.

From an engineering design perspective, this reflects a deliberate trade-off: maximizing performance under hardware and power constraints without compromising the reliability of classification outcomes. These optimization decisions are implemented in the deployment methodology described in Chapter V.

4.2.2 Delta Robot Kinematics and Workspace for High-Speed Sorting

Delta robots are widely studied in robotics literature for high-speed, low-payload pick-and-place applications. Their parallel kinematic architecture distributes loads across multiple identical arms, reducing moving mass and enabling high accelerations and precise positioning.

Comparative analyses of robotic architectures for industrial sorting (Delta vs. 6-axis articulated arms, SCARA, and Cartesian robots) consistently show that Delta robots offer: Shortest cycle times for light payloads (e.g., 2–3 seconds per pick, versus 5–8 seconds for typical articulated arms). Better repeatability, often in the range of ±1–2 mm, compared to ±5 mm or worse for many serial robots. Cylindrical workspaces that align well with conveyor layouts, enabling efficient coverage of a rectangular belt with bins placed around the periphery.

These characteristics directly address realistic constraints in waste sorting: throughput, precision, compact footprint, and energy efficiency. For example, a case study of an industrial Delta robot (such as FANUC M-2iA) reports maximum TCP speeds on the order of 8 m/s, vertical travel around 450 mm, payload capacities of 2–3 kg, and repeatability better than ±0.2 mm—well beyond what is strictly necessary for sorting household waste items that generally weigh less than 500 g.

From a kinematics standpoint, Delta robots require inverse kinematics (IK) solutions to convert desired end-effector positions into joint angles. Analytical IK formulations exist for standard Delta geometries, which are computationally inexpensive and suitable for implementation on microcontrollers such as Arduino, as used in this project. This allows the main embedded platform (Jetson Nano) to focus on deep-learning inference and high-level

coordination, while the low-level motor control and IK are offloaded to a dedicated controller.

Several research works on vision-guided Delta robots provide complete formulations and experimental validation of IK and hand–eye calibration routines, achieving end-effector positioning errors below 2 mm and repeatability within ±1–2 mm under industrial conditions. These results establish a solid theoretical and empirical foundation for the mechanical design and control architecture adopted in this thesis.

4.2.3 Vacuum Gripper Design for Heterogeneous Waste Materials

End-effector selection is critical in waste manipulation because waste items vary widely in shape, texture, rigidity, and material properties. Comparative studies on robotic grippers for sorting tasks typically consider mechanical parallel-jaw grippers, specialized fingered grippers, and vacuum-based grippers.

For heterogeneous waste, especially flat or moderately curved objects such as paper sheets, cardboard, plastic trays, and metal cans, vacuum grippers have emerged as a preferred solution:[37Initialization with pretrained weights reduce. They offer high adaptability: a single suction cup can handle objects of different shapes and materials without mechanical reconfiguration. They tolerate moderate positioning errors, as the suction area can still form a seal as long as it lands within the object footprint. They avoid applying concentrated mechanical forces that could deform or damage fragile items such as thin plastic or crushed paper.

In the intelligent plastic waste sorting system based on a Delta robot, the authors implemented a 12 V DC vacuum pump controlled by a relay and a 2-way solenoid valve. The gripper operated in distinct states (e.g., PRIME, HOLD, RELEASE), synchronized with the Delta robot's motion to ensure stable grasping and timely release. The system achieved a pick success rate above 97% across various plastic items, even under changing lighting conditions.

In another YOLOv8 + Delta-arm waste sorting system, vacuum suction is again employed as the primary gripping mechanism, with comparable success rates and cycle times. The researchers conclude that vacuum-based end-effectors strike the best balance between simplicity, robustness, and versatility for mixed recyclable waste streams.

These findings directly motivate the choice of a vacuum gripper in this thesis. By integrating a 12 V vacuum pump, solenoid valve, and suitable suction cup at the end-effector, the Delta robot in this project can reliably grasp paper, plastic, and metal items on a black conveyor belt without frequent mechanical adjustments. This choice reflects not only technical suitability but also economic and maintenance considerations, as vacuum grippers tend to be inexpensive, easy to maintain, and tolerant of dirty environments typical of waste facilities.

4.3 Summary and Research Gaps

The literature reviewed in this chapter leads to several key observations:

1 YOLOv8 is well-suited for waste detection.

Multiple studies have demonstrated that YOLOv8 can achieve mAP@0.5 around 0.9 or higher on complex, multi-class waste datasets, while maintaining real-time throughput on embedded GPUs such as Jetson Xavier NX. When scaled down to YOLOv8n and deployed on Jetson Nano with appropriate optimization, achieving around 8 FPS is both realistic and sufficient for conveyor-based sorting in this project.

2 Embedded optimization is essential for real-time performance.

Conversion of YOLO models to TensorRT engines with FP16 precision is now a standard practice to reduce latency and memory footprint on Jetson platforms. This approach enables real-time inference under a 20 W power budget and a 4 GB memory constraint, aligning with economic and energy limitations in small and medium-sized waste facilities, especially in emerging economies such as Vietnam.

3 Delta robots and vacuum grippers provide mechanical advantages for waste sorting.

Delta robots offer superior cycle times and positioning accuracy for light payloads compared to many alternative architectures, and vacuum grippers provide the material and geometric adaptability required to handle diverse waste items without frequent reconfiguration. Vision-guided Delta systems using suction end-effectors have already achieved high throughput and accuracy in both experimental and semi-industrial environments.[21]

Despite these advances, several gaps remain that motivate the present thesis:

- Many YOLOv8-based systems for waste detection are evaluated on high-end hardware (desktop GPUs or Jetson Xavier/Orin) and do not fully address the software and compatibility constraints of older embedded platforms like Jetson Nano (Python 3.6, JetPack 4.6.4). This thesis explicitly considers these constraints and shows how careful model selection (YOLOv8n) and deployment strategy (TensorRT FP16) can yield a practical solution.

- Several Delta robot + vision systems focus on specific materials (e.g., only plastic) or do not use fully embedded inference; instead, they rely on external PCs for heavy computation. By contrast, this project aims to implement the entire detection and control pipeline on Jetson Nano plus a microcontroller, targeting a low-cost and compact architecture suitable for deployment in resource-constrained settings such as small local sorting stations in Vietnam, where budget, power, and infrastructure limitations make high-end GPU servers impractical.

- While prior work recognizes the importance of dataset diversity and domain alignment, relatively few studies document in detail the strategy of combining public conveyor-belt datasets with laboratory-captured images from the actual deployment camera. This thesis adopts such a strategy—merging a RoboFlow conveyor dataset with images captured using the system's PS3 Eye camera—to bridge the gap between generic training data and the specific visual characteristics of the real environment.

By addressing these gaps, the proposed system contributes an end-to-end, embedded, and economically feasible solution for automated classification and sorting of paper, plastic, and metal waste. The design choices in subsequent chapters—ranging from dataset preparation and model training (Chapter V) to embedded deployment and robotic integration (Chapter VI)—are directly grounded in the empirical evidence and engineering principles summarized in this chapter.

CHAPTER V
METHODOLOGY

This chapter describes the complete methodology used to design, train, deploy, and integrate the proposed automated waste classification and pick-and-place system. The methodology follows the design specifications in Chapter II and prepares the ground for the experimental results in Chapter VI. It covers:

- Overall system architecture and workflow.
- Dataset design, preprocessing, and augmentation.
- Model selection, transfer learning, and training.
- Model export and deployment on NVIDIA Jetson Nano.
- Evaluation metrics and performance assessment.
- Hardware and robotic integration, including Delta robot modifications and vacuum gripper control.

5.1 System Architecture and Workflow Overview

The proposed system follows a sequential pipeline that transforms raw camera images into robotic pick-and-place actions:

Image acquisition from a top-view PS3 Eye USB camera mounted above a black conveyor belt.

Preprocessing and normalization of each frame to a fixed resolution and color format.

Real-time inference with YOLOv8n on the Jetson Nano to detect and classify waste items (paper, plastic, metal).

Centroid calculation for each detected bounding box and mapping from image

coordinates to robot coordinates.

Transmission of target coordinates and grasp commands from Jetson Nano to an Arduino Mega via USB-serial.

Robotic actuation by the Delta robot and vacuum gripper to pick items from the conveyor and place them into the correct bins.

All computationally intensive model training is performed offline on Google Colab using an NVIDIA Tesla K80 GPU. Only the final optimized model is deployed to the Jetson Nano, which operates strictly in inference mode. This design maintains deterministic real-time behavior on the embedded platform while exploiting cloud resources for heavy training.The overall workflow is illustrated in Figure 5.1

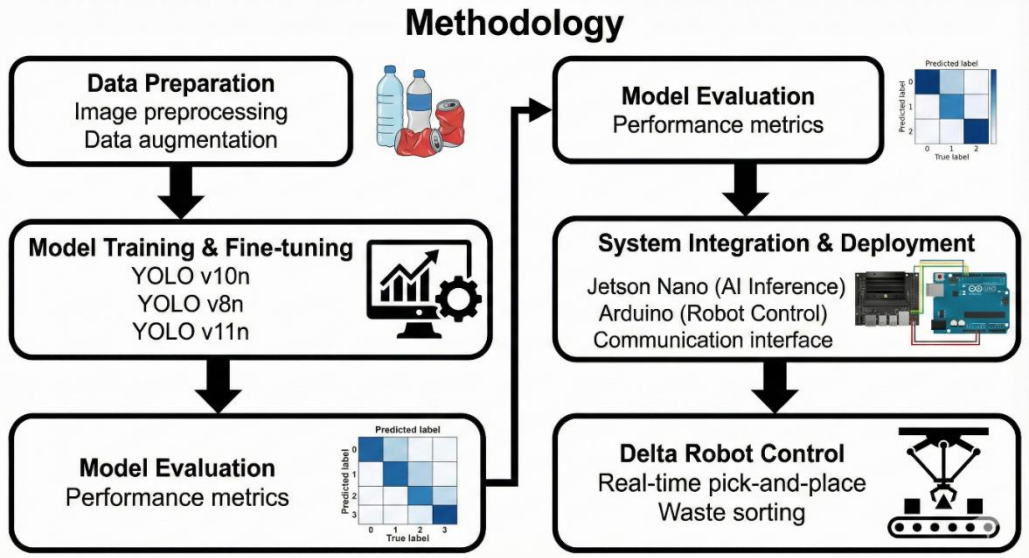

Figure 5.1: *System Architecture and End-to-End Processing Pipeline.*

5.2 Dataset Preparation and Composition

5.2.1 Data Source and Rationale

- RoboFlow conveyor-belt dataset:

1,484 labeled images of waste items belonging to three categories: plastic, metal, and paper. All images depict objects on a uniform black conveyor background, closely matching the target deployment environment. The dataset provides high-quality bounding boxes and balanced categories, making it suitable for initializing a general detector.

- Personal laboratory dataset:

221 images were collected using the PS3 Eye camera installed on the real system. Images were captured under controlled laboratory conditions with the same black background and an enforced resolution of 480×480 pixels. This dataset captures variations specific to the physical samples, lighting conditions, and optics of the deployed system.

Using only open-source data led to a performance drop when tested on the real hardware due to differences in camera and lighting. The additional laboratory dataset was therefore created to reduce this domain gap. Representative samples from both datasets are shown in Figure 5.2 and Figure 5.3.

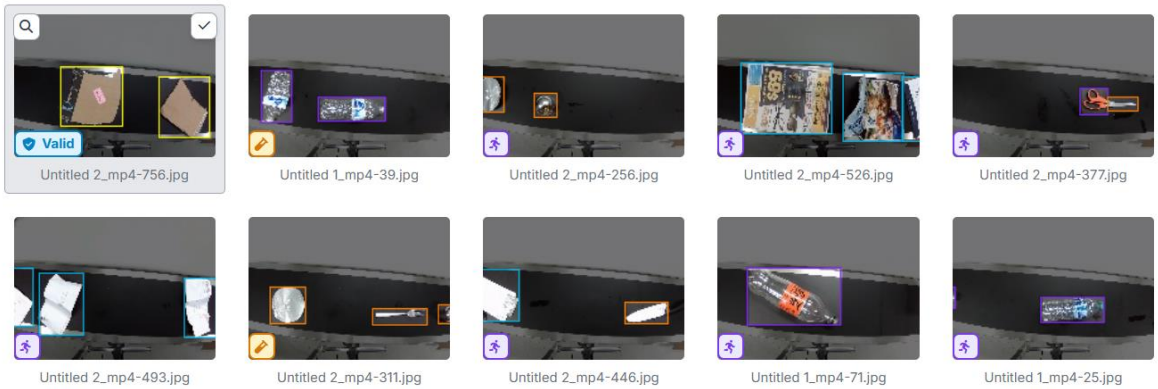

Figure 5.2: *Examples from the RoboFlow conveyor-belt dataset.*

Figure 5.3: *Examples from the laboratory dataset captured with the PS3 Eye camera.*

5.2.2 Incremental Transfer Learning Strategy

A two-stage incremental transfer learning approach is used:

- The YOLOv8n model is first trained on the larger RoboFlow dataset. This phase builds a strong feature representation of waste items under diverse conditions.

- The trained model is then fine-tuned on the laboratory dataset. This adapts the weights to the PS3 Eye camera viewpoint, actual sample materials, and lighting conditions without discarding the general features learned from RoboFlow.

This strategy balances data diversity with domain specificity. It avoids overfitting to the relatively small in-house dataset while ensuring that the final model matches the deployment environment.

5.2.3 Statistical Overview and Diversity

Table 5.1 – Dataset Components

Component	Image Count	Classes	Resolution	Background	Collection Method
RoboFlow OpenSource	1,484	Plastic, Metal, Paper	Standardized to 480×480	Black conveyor	Public, professionally annotated
Personal Lab Capture	221	Plastic, Metal, Paper	480×480	Black background	PS3 Eye USB camera
Total Combined	1,705	3 categories	480×480	Uniform black	Curated + in-house

Class distribution analysis showed that some sub-types, particularly certain plastic appearances, were underrepresented relative to metal and paper. This imbalance motivated the class-aware augmentation and loss weighting described in Section 5.3.3.

The combined dataset intentionally includes:

- Variation in material texture (smooth plastic, crumpled paper, reflective and non-reflective metals).
- Variation in geometry (different orientations, scales, and aspect ratios).
- Variation in lighting (brighter and darker scenes, partial shadows).
- Scenes with multiple objects, partial occlusion, and clutter from the conveyor structure.

This diversity helps the model develop robust, presentation-invariant features.

5.3 Data Preprocessing and Augmentation Pipeline

5.3.1 Image Standardization and Normalization

Each image passes through the same preprocessing steps:

- Resize to 480×480 pixels using bicubic interpolation. This resolution was chosen to provide enough detail for material discrimination while allowing sub-100 ms inference on Jetson Nano.
- Normalize pixel values from [0,255][0,255] to [0,1][0,1] by dividing by 255. This improves numerical stability during training.
- Convert to RGB three-channel format to match YOLOv8n input expectations. Any grayscale or RGBA images are converted.
- Strip metadata such as EXIF tags and timestamps to avoid implicit cues and to simplify dataset handling.

5.3.2 Data Augmentation Strategy

Online data augmentation is applied during training to increase effective dataset size:

- Geometric transformations: random rotations within ±15∘±15∘, horizontal flips, occasional vertical flips, and small random crops from image edges.
- Photometric transformations: random brightness and contrast adjustments, as well as hue, saturation, and value jittering in HSV space.
- Noise injection: additive Gaussian noise with small variance to simulate sensor and compression artifacts.

Each image is transformed with a random subset of these operations at each epoch. The result is that no two training iterations see the same augmented sample.

5.3.3 Class Imbalance Mitigation

To address class imbalance, three measures are taken:

- Oversampling minority classes when constructing each epoch.

  Applying stronger augmentation to minority classes to increase their effective diversity.
- Weighting the loss function by inverse class frequency:

$$W_c = \frac{N_{total}}{N_c}$$

  where Nc is the number of samples for class c. Misclassifications for rare classes are

thus penalized more strongly.

5.4 Dataset Splitting and Validation Strategy

The 1,705 images are split into training, validation, and test sets using stratified random sampling to preserve class ratios.

Table 5.2 – Dataset Split

Subset	Proportion	Image Count	Purpose
Training	70%	1,194	Parameter optimization via backpropagation
Validation	15%	256	Hyperparameter tuning and early stopping
Test	15%	255	Final unbiased evaluation

Both RoboFlow and lab images are present in each subset to avoid temporal bias. The validation set is monitored during training, and early stopping is triggered if validation loss does not improve for 5 consecutive epochs.

5.5 Model Architecture Selection and Justification

5.5.1 Selection Criteria and Trade-offs

Architectures were evaluated according to:

- Ability to reach or exceed the target mAP@0.5 ≥ 0.85.

  Inference speed and memory footprint on Jetson Nano (≥ 5–8 FPS within 4 GB RAM).

- Compatibility with JetPack 4.6.4 and Python 3.6.

Image classification networks can classify only a single central object and do not produce bounding boxes. Because the conveyor can hold multiple items simultaneously, an object detector is required. Among object detectors, the YOLO family offers the most favorable combination of speed and accuracy for embedded devices.

Within the YOLOv8 family, YOLOv8n (nano) was selected because it:

- Provides multi-object detection with a compact model.
- Runs efficiently on Jetson Nano with TensorRT FP16.

  Remains compatible with Python 3.6, unlike newer YOLO variants.

Achieves competitive accuracy on waste detection datasets as shown in Chapter IV.

5.5.2 YOLO 8n Architecture Overview

YOLOv8n contains three main components:

Backbone: a lightweight convolutional network that extracts multi-scale features from the input image using CSP-style blocks and depthwise separable convolutions.

Neck: a feature pyramid structure that fuses features from different depths, enabling detection of both small and large waste items.

Head: detection heads at multiple scales that predict, for each candidate location, bounding box coordinates (x,y,w,h), an objectness score, and class probabilities for paper, metal, and plastic.

This architecture is well suited to the task because it can detect multiple items in one pass and provides bounding box centers directly usable for robot targeting.

The YOLOv8n model is initialized with pretrained weights provided by Ultralytics. These weights have been trained on large general-purpose image datasets and encode rich low-level features such as edges, corners, and textures. Initialization with pretrained weights reduces both the amount of required training data and the convergence time.[8][29][46]

5.5.2 Layer Freezing and Fine-Tuning

A staged strategy is used:

In the first few epochs, the backbone and neck layers are frozen; only the detection head is trained. This treats the pretrained feature extractor as a fixed encoder and quickly adapts the output layers to the new classes.

After the detection head has converged, all layers are unfrozen and trained with a reduced learning rate. This allows the backbone to adapt slowly to waste-specific patterns without destroying useful generic features.

This approach balances stability and adaptability and is especially effective with a moderate dataset size.

5.5.3 Learning Rate Schedule

A cosine annealing schedule is applied:

$$\mathrm{LR}(t) = \mathrm{LR}_{\min} + \frac{1}{2}(\mathrm{LR}_{\max} - \mathrm{LR}_{\min})\left[1 + \cos\left(\frac{\pi * t}{T}\right)\right]$$

Where t is the current epoch, T=100 is the total number of epochs, $\mathrm{LR\_max} = 0.01$ and $\mathrm{LR\_min} = 0.0001$. This schedule starts with a relatively high learning rate for rapid convergence, then gradually reduces it for fine-tuning.

5.6 Training Process and Hyperparameter Configuration

5.6.1 Training Environment

Training is carried out on Google Colab with an NVIDIA Tesla K80 GPU (12 GB VRAM). The software stack consists of:

- Python 3.9,
- PyTorch 2.0,
- Ultralytics YOLOv8,
- OpenCV 4.8 and NumPy 1.24.

5.6.2 Main Hyperparameters

Table 5.3 – Training Hyperparameters

Parameter	Value	Justification
Batch size	32	Utilizes GPU memory while maintaining stable gradients.
Epochs	100	Sufficient for convergence with early stopping.
Optimizer	SGD with momentum 0.937	Stable optimizer for YOLO.
Weight decay	0.0005	L2 regularization to reduce overfitting.
Initial learning rate	0.01	Combined with cosine annealing schedule.
Image size	480×480	Balances detail with Jetson Nano constraints.
NMS IoU threshold	0.45	Removes redundant overlapping detections.
Confidence threshold	0.50	Filters low-confidence detections during inference.

5.6.3 Two-Phase Training Protocol and Convergence

Phase 1 trains on the RoboFlow dataset only (40 epochs), achieving validation

mAP@0.5 ≈ 0.91. Phase 2 continues training on the combined dataset (up to 100 epochs), focusing on adapting to the PS3 Eye camera. Final performance is:

- Validation mAP@0.5 ≈ 0.94,
- Test mAP@0.5 ≈ 0.91,
- Training time around 90 minutes on the K80 GPU.

Training progress is monitored using loss, mAP, precision, and recall curves. Early stopping is triggered when validation performance plateaus.

5.7 Model Export and Deployment Formats

5.7.1 Conversion Pipeline

After training, the model is converted through several formats to optimize deployment on Jetson Nano:

PyTorch (.pt): native training format, used for further fine-tuning but not ideal for deployment.

ONNX (.onnx): intermediate hardware-agnostic representation created using the Ultralytics export function.

TensorRT engine (.engine): hardware-specific optimized binary generated on Jetson Nano using trtexec with FP16 enabled.

Full conversion scripts and commands are provided in Appendix A. In the main text, only the overall pipeline is described because the detailed code is primarily implementation rather than methodology.[41][42]

5.7.2 Python Version and Model Choice

JetPack 4.6.4 for Jetson Nano uses Python 3.6. YOLOv8n can be deployed under this environment, while newer YOLO variants require Python 3.8 or later and incompatible dependencies. Comparative experiments showed that YOLOv10n and YOLOv11n improve accuracy only by 1–2 percentage points but introduce significant deployment complexity. For this reason, YOLOv8n is selected as the best balance between performance and maintainability.

5.8 Evaluation Metrics

5.8.1 Detection Metrics

Two principal metrics are used:

Intersection over Union (IoU) between predicted and ground truth boxes:

$$\text{IoU} = \frac{\text{Area}(\text{Pred} \cap \text{GT})}{\text{Area}(\text{Pred} \cup \text{GT})}$$

Detections with IoU ≥ 0.5 are considered true positives.

Average Precision (AP) for each class:

$$\text{AP} = \int_0^1 \text{P}(\text{r})\text{dr}$$

where $\text{P}(\text{r})$ is precision as a function of recall.

The overall performance is summarized by mean Average Precision (mAP):

$$\text{mAP} = \frac{1}{3} \sum_{\text{c}} \text{AP}_{\text{c}}$$

This scalar is used as the primary performance indicator (reported as mAP@0.5).

Per-class precision, recall, and F1-score are also computed:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}, \text{F}_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision+Recall}}$$

5.8.2 Confusion Matrix and Final Results

A 3×3 confusion matrix is used to inspect systematic misclassifications between paper, plastic, and metal. This reveals, for example, confusion between bright plastic and reflective metal.

The final performance on the held-out test set is summarized in Table 5.4.

Table 5.4 – Test-Set Performance (YOLOv8n)

Metric	Plastic	Metal	Paper	Overall
Precision	0.91	0.94	0.96	0.94
Recall	0.88	0.89	0.94	0.90
F1-score	0.89	0.91	0.95	0.92
mAP@0.5	0.89	0.92	0.94	0.91

The overall mAP@0.5 of 0.91 exceeds the design target of 0.85 specified in Chapter II.

5.9 Hardware Integration and Embedded Deployment

5.9.1 Jetson Nano Configuration

The target platform is the NVIDIA Jetson Nano 4 GB developer kit. Key specifications are:

- Quad-core ARM Cortex-A57 CPU @ 1.43 GHz,
- 128-core NVIDIA Maxwell GPU @ 921 MHz,
- 4 GB LPDDR4 shared memory.

The board is set to MAXN performance mode, and clocks are locked at maximum frequency. CUDA and cuDNN from JetPack 4.6.4 are used together with TensorRT for GPU-accelerated inference. After optimization, the end-to-end latency per frame is approximately 126 ms, corresponding to about 8 FPS.

5.9.2 Camera Integration and Calibration

The PS3 Eye camera is UVC-compliant and appears as /dev/video0. Frames are captured at 480×480 resolution. A one-time calibration using a checkerboard pattern yields intrinsic parameters and distortion coefficients. OpenCV's undistort function is applied to each frame, improving centroid accuracy for robot targeting.

5.9.3 Arduino-Based Robot Control

The Jetson Nano sends commands to an Arduino Mega over USB-serial. Each command specifies a target position and gripper state. An example of the textual command format is:

MOVE,150,200,50,1

HOME,0,0,0,0

RELEASE,0,0,0,0

A short Python script on Jetson constructs these strings and sends them via pySerial. The full scripts for Jetson and Arduino are provided in Appendix A and Appendix B, respectively. The Arduino firmware interprets the commands, generates step pulses for the three stepper motors, and controls the relays for the vacuum pump and solenoid valve.

5.10 Robotic System Hardware Modifications

5.10.1 Delta Robot Mechanics

The custom Delta robot uses three NEMA23 stepper motors and parallel linkages to move a central end-effector in three dimensions. During early tests, mechanical interference occurred at large joint angles due to oversized hardware at the central joint. This reduced usable workspace and introduced vibration.

To address this:

- The joint design was modified to use narrower fasteners, reducing the joint width and enabling nearly 180° of angular sweep.
- Compression springs were added to provide passive damping and maintain

stiffness at the extremes of the workspace.

These modifications improved both reach and positional repeatability, helping the system meet the ±5 mm positioning specification from Chapter II.[11][21][44]

5.10.2 Vacuum Gripper

A 12 V DC vacuum pump and a 3/2 solenoid valve form the vacuum gripper. The system allows grasping of flat and moderately curved objects regardless of material, which is ideal for mixed waste.

The Arduino implements a small state machine with four states (OFF, PRIME, HOLD, RELEASE) to synchronize pump and valve activation with robot motion. This design reduces complexity while ensuring reliable grasp and release.

5.10.3 Homing with Limit Switches

To establish a consistent reference frame, limit switches are installed at the home positions of each axis. On startup, the robot executes a homing routine that moves each axis until its switch is triggered, then sets the coordinate origin. All subsequent movements are expressed relative to this origin. This ensures consistent mapping between image centroids and robot coordinates even after power cycles.

5.11 Summary of Methodology

This chapter has presented a comprehensive methodology for building the proposed waste classification and sorting system. Key aspects include:

- Construction of a hybrid dataset combining open-source conveyor-belt images with laboratory images from the real system.
- Careful preprocessing, augmentation, and stratified splitting to support robust training and fair evaluation.
- Selection and training of YOLOv8n via incremental transfer learning, achieving the target mAP within the constraints of Jetson Nano.
- Conversion of the trained model to TensorRT FP16 format and deployment on Jetson Nano with approximately 8 FPS throughput.
- Integration with a PS3 Eye camera, Arduino-controlled Delta robot, and vacuum gripper, including mechanical improvements and homing logic.

These methodological steps ensure that the system meets the quantitative design specifications of Chapter II and form the basis for the experimental evaluation presented in Chapter VI.

CHAPTER VI
EXPECTED RESULTS

6.1 Model Training Results and Architecture Comparison

6.1.1 Training Strategy and Dataset

Three YOLO architectures were trained on identical datasets to identify the optimal model for embedded deployment: YOLOv8n, YOLOv10n, and YOLOv11n. The combined dataset contained 1,705 images (70% training, 15% validation, 15% test) from two complementary sources: 1,484 professionally annotated RoboFlow images and 221 custom laboratory images captured with the system camera. All models were trained on Google Colaboratory with NVIDIA Tesla K80 GPU using SGD optimizer, cosine learning rate annealing, batch size 32, and early stopping (patience 5 epochs). Training typically completed by epoch 95–100 following the two-phase incremental transfer learning approach described in Chapter V.

6.1.2 Quantitative Performance Comparison

All three architectures exceeded the target mAP@0.5 ≥ 0.85 specified in Chapter II, but significant differences emerged in per-class performance and deployment feasibility.

Table 6.1 – Model Performance Comparison

Model	Val mAP@0.5	Test mAP@0.5	Training Time	Confusion Matrix Quality
YOLOv8n	0.94	0.91	~90 min	Best balanced
YOLOv10n	0.93	0.90	~85 min	Slight metal/plastic blur
YOLOv11n	0.92	0.89	~85 min	Reduced small object recall

Per-Class Performance (YOLOv8n on Test Set)

Class	Precision	Recall	F1-Score	mAP@0.5
Plastic	0.91	0.88	0.89	0.89
Metal	0.94	0.89	0.91	0.92
Paper	0.96	0.94	0.95	0.94

Class	Precision	Recall	F1-Score	mAP@0.5
Overall	0.94	0.90	0.92	0.91

YOLOv8n produced the most balanced classification across all categories with minimal inter-class confusion, particularly excelling at distinguishing white plastic from reflective metal. YOLOv10n and YOLOv11n showed slightly faster convergence but exhibited increased confusion between white plastic and metal under suboptimal lighting, and reduced recall for small metal objects.

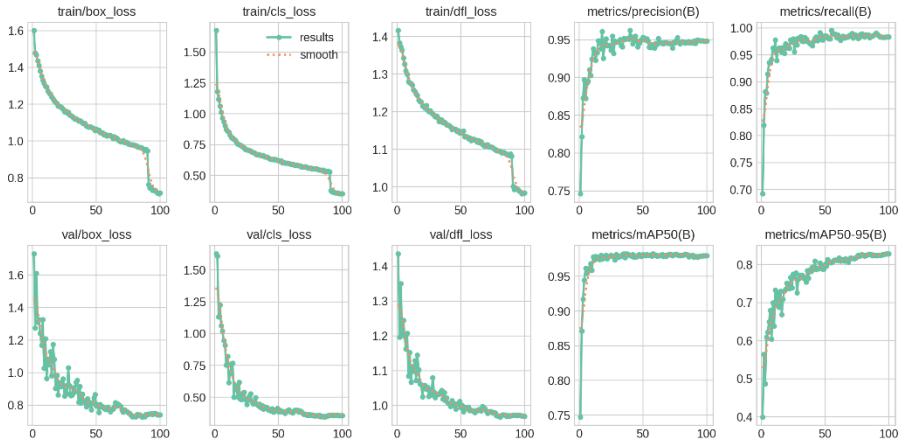

Figure 6.1: *YOLOv8n Training and Validation Curves (loss, mAP@0.5, precision, recall vs. epoch).*

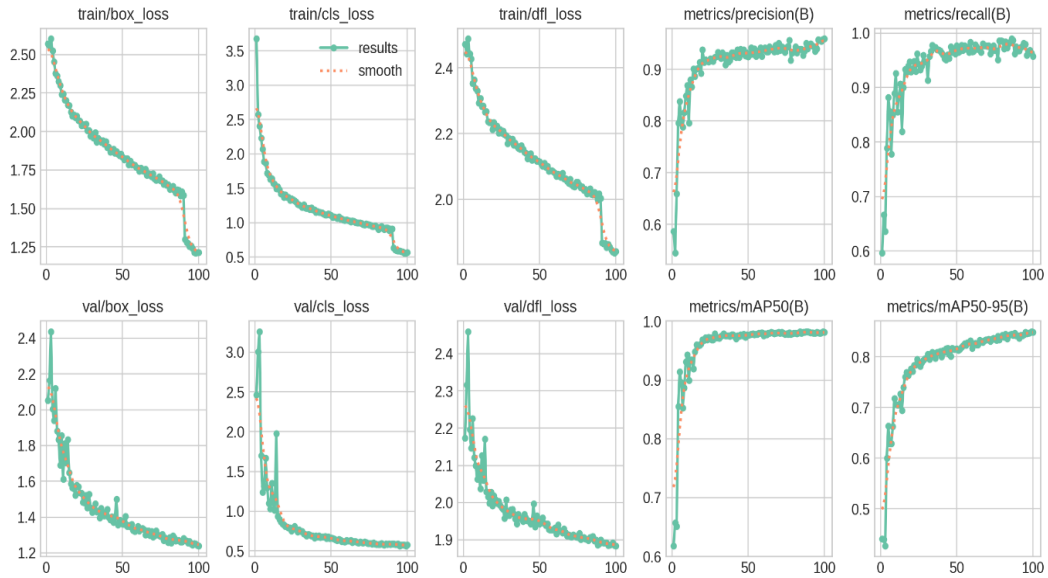

Figure 6.2: *YOLOv10n Training and Validation Curves*

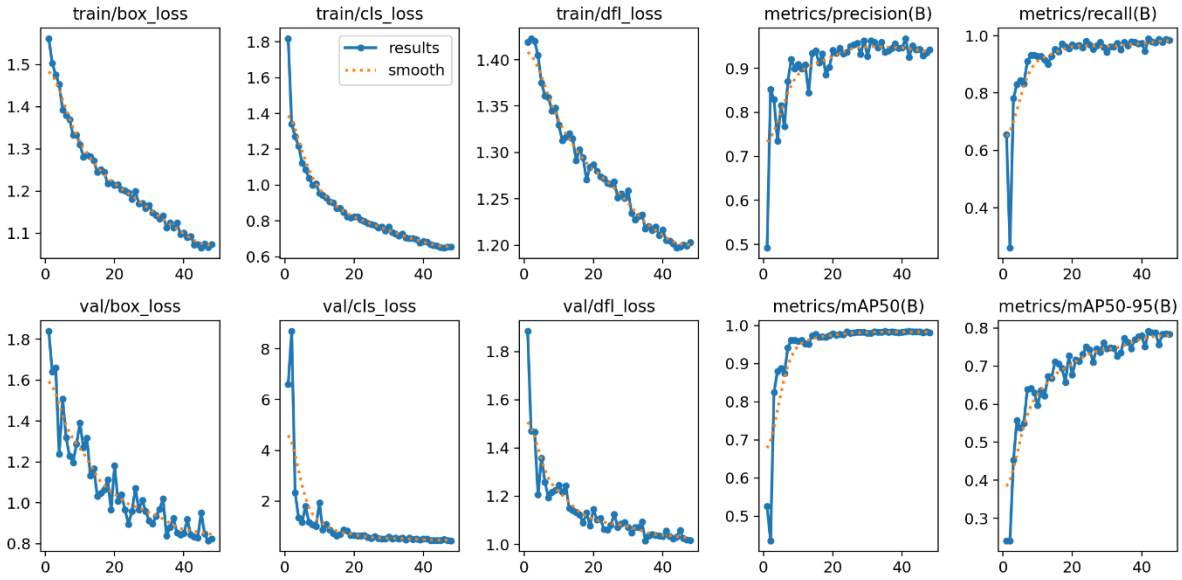

Figure 6.3: *YOLOv11n Training and Validation Curves*

Across all three models, validation mAP saturates above 0.95 within 40–50 epochs and stabilizes without significant oscillation. YOLOv8n maintains the smallest gap between training and validation curves, indicating superior generalization. YOLOv10n and YOLOv11n show slightly larger fluctuations in later epochs, particularly around learning rate reductions and for small object detection.

6.1.3 Model Selection: YOLOv8n

Despite comparable training accuracy, YOLOv8n was selected as the deployment model due to critical deployment constraints. The Jetson Nano with JetPack 4.6.4 ships with Python 3.6, while YOLOv10n and YOLOv11n require Python 3.8+ due to PyTorch 1.13+ dependencies. Upgrading system Python on Jetson Nano risks breaking CUDA bindings and system packages. Practical deployment tests confirmed that YOLOv10n and YOLOv11n could not be successfully compiled to TensorRT engines on this hardware without risky custom modifications. YOLOv8n has extensive documented deployments on Jetson Nano with proven TensorRT optimization pipelines. This engineering decision—prioritizing deployment stability and maintainability over marginal speed improvements—reflects practical system design philosophy. The selected model achieved test mAP@0.5 = 0.91, exceeding the target 0.85 specified in Chapter II.

6.2 Deployment on Jetson Nano

6.2.1 System Configuration

The NVIDIA Jetson Nano 4GB was configured in MAXN performance mode (all CPU cores at 1.43 GHz, GPU at 921 MHz) to ensure real-time inference stability. JetPack 4.6.4 provides Ubuntu 18.04, CUDA 10.2, cuDNN 8.2.1, and TensorRT 8.2.1. The

YOLOv8n model was exported from PyTorch to ONNX format, then compiled into a TensorRT engine on the Jetson Nano itself using the trtexec utility. TensorRT optimization includes layer fusion, FP16 precision reduction, and kernel auto-tuning, reducing the model size to approximately 8 MB and enabling GPU-accelerated inference. System configuration consumed approximately 20 W power (5V @ 4A) with active cooling.

Remote development was enabled through NoMachine for desktop access and WinSCP for secure file transfer between development PC and Jetson.

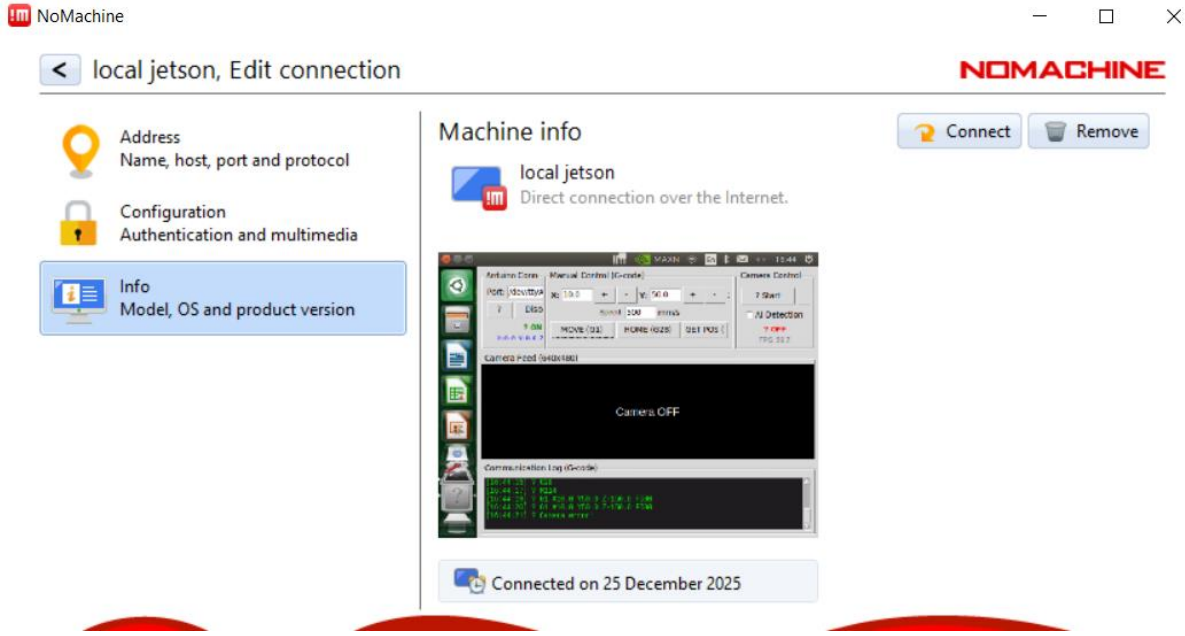

Figure 6.4: *Remote Desktop (NoMachine)*
Screenshot showing Jetson desktop accessible from Windows development PC.

Name	Size	Type	Chang
..		Parent directory	29/11/
test_detected.jpg	2.858 KB	JPG File	17/10/
test_connect_jetson_ar...	10 KB	Python Source File	28/11/
test.jpg	5.011 KB	JPG File	12/10/
run_all_onnx_build_an...	3 KB	Shell Script	09/10/
README_jetson.md	2 KB	Markdown Source ...	09/10/
jetson_test_model.py	12 KB	Python Source File	10/10/
camera_test_onnx.py	11 KB	Python Source File	21/11/
benchmark_all_model...	2 KB	Python Source File	13/10/

Name	Size	Type
..		Parent directory
8n_26_9_best.onnx	11.908 KB	ONNX File
8n_26_9_best.pt	6.103 KB	PT File
8n_best.onnx	11.908 KB	ONNX File
10n_best.onnx	9.049 KB	ONNX File
11n_26_9_best.onnx	10.286 KB	ONNX File
11n_26_9_best.pt	5.341 KB	PT File
best.onnx	11.908 KB	ONNX File
best_5types_yolo11.pt	5.345 KB	PT File
best_8n_augu.onnx	11.908 KB	ONNX File
best_11n_agu.onnx	10.286 KB	ONNX File
best_yolo8n.onnx	11.908 KB	ONNX File
best_yolo8n.pt	6.103 KB	PT File
best_yolo11n.onnx	10.286 KB	ONNX File
best_yolo11n.pt	5.341 KB	PT File
waste-yolov10n-prod...	9.050 KB	ONNX File
yolo11n.pt	5.483 KB	PT File

Figure 6.5: *File Transfer (WinSCP)*
Screenshot showing file synchronization between PC and Jetson.

6.2.2 Camera Integration and Preprocessing

The PS3 Eye USB camera was configured to capture 480×480 frames at 60 FPS (internally downsampled to 8 FPS processing rate). Camera intrinsic parameters were calibrated using checkerboard calibration to correct approximately 15–20% barrel distortion, ensuring that detected object centroids accurately represent physical positions. The camera was mounted directly above the conveyor belt at 400 mm height, providing orthographic view for minimizing perspective error during robotic grasping.

Figure 6.6: *Camera Mounting Position*
Side and top view showing camera height and mounting orientation above conveyor belt.

6.2.3 Inference Performance

The deployed YOLOv8n TensorRT engine achieved 8 FPS throughput on Jetson Nano by processing every second frame from the 60 FPS camera stream. Per-frame latency breakdown:

Table 6.2 – Inference Latency (YOLOv8n TensorRT, 480×480)

Stage	Time
Camera capture + preprocessing	~30 ms
TensorRT GPU inference	~100 ms

Stage	Time
Postprocessing (NMS, centroid)	~10 ms
Total per frame	~125 ms
Effective throughput	8 FPS

This performance satisfies real-time requirements for conveyor speed and robot cycle time (2–3 seconds per object). GPU utilization reaches approximately 70%, CPU approximately 40%, and RAM approximately 1.2 GB of 4 GB available (30% utilization).

6.2.4 Detection Results

After TensorRT integration and camera calibration, the deployed model was evaluated under realistic operating conditions with mixed waste objects on a black conveyor belt.

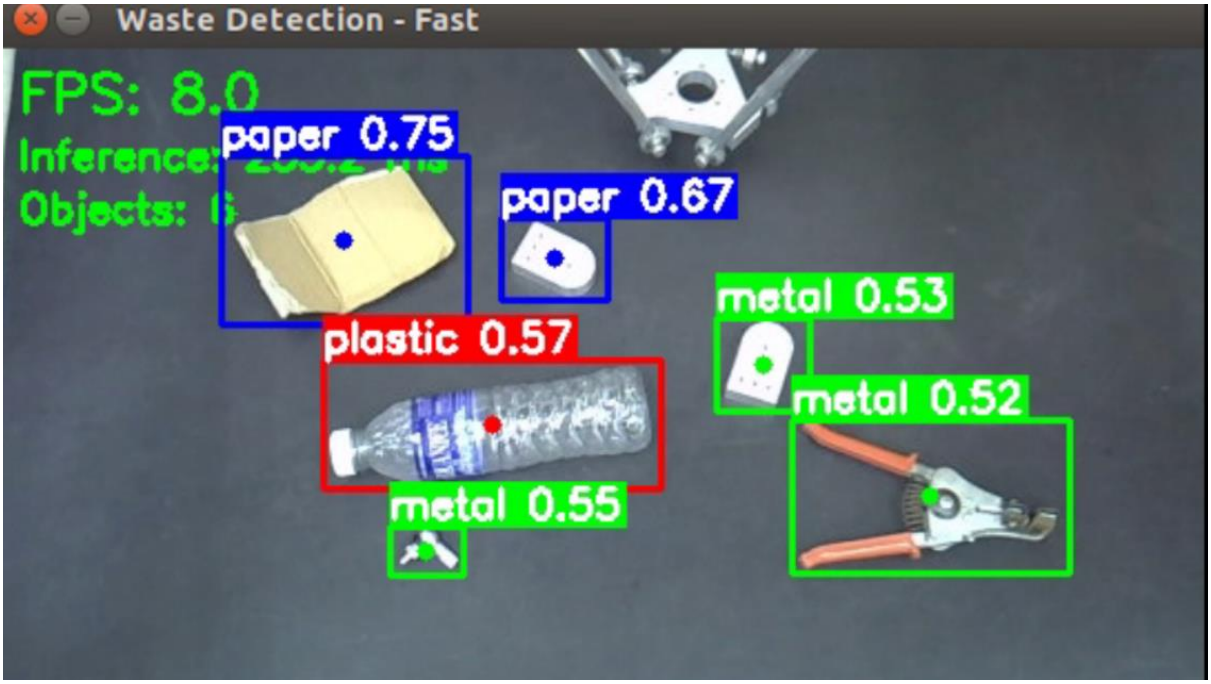

Figure 6.7: *Multi-Object Detection in Real Time*
Example frame showing multiple objects (plastic, metal, paper) with bounding boxes, class labels, confidence scores, object count, and 8 FPS indicator.

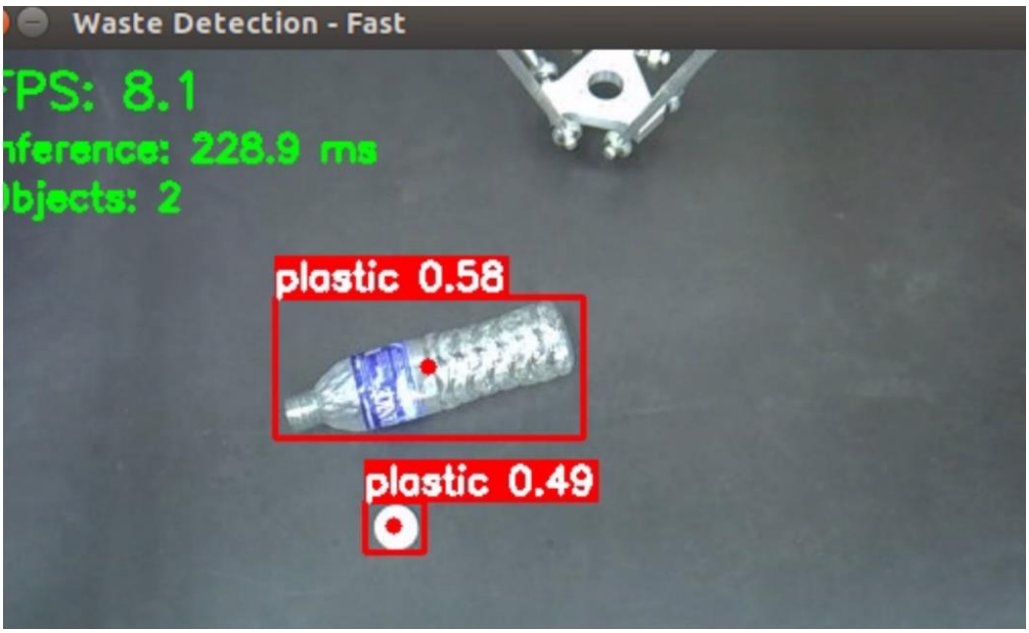

Figure 6.8: *Plastic Detection Examples*
Multiple examples showing plastic objects, highlighting challenging white plastic samples with 70–80% confidence scores due to specular reflections.

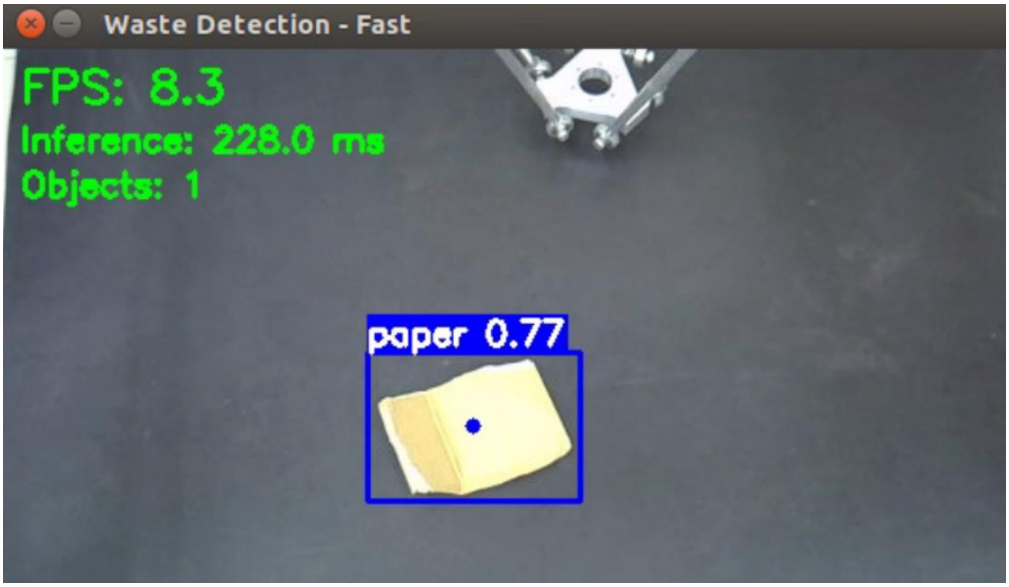

Figure 6.9: *Paper Detection Examples*
Cardboard and paper items with consistently high confidence (>90%), attributed to distinctive fibrous texture and matte finish.

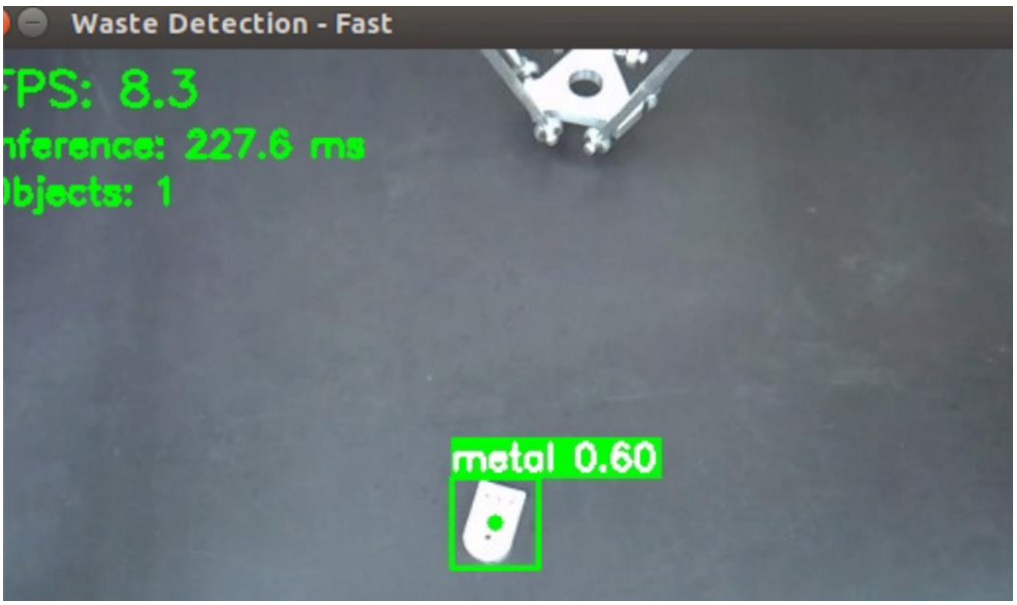

Figure 6.10: *Metal Detection Examples*
Metal objects under good and suboptimal lighting, showing high confidence under good lighting and occasional confusion with white glossy paper in shadows.

Paper detection achieved the highest reliability (>90% confidence) due to consistent texture and coloration. Metal detection was robust under good illumination but occasionally confused with reflective white paper in low light. White plastic presented the most significant challenge due to strong specular highlights resembling metallic surfaces, resulting in 70–80% confidence and higher misclassification rates compared to textured or colored plastic. The model successfully learned to ignore the black conveyor background and the visible Delta robot end-effector.

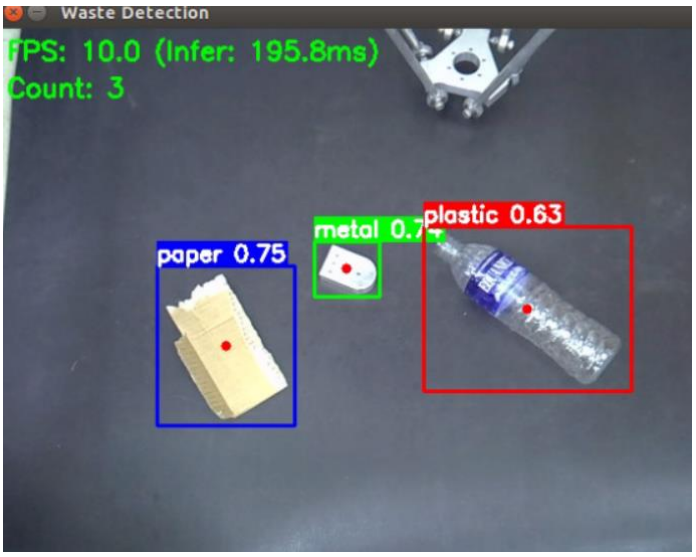

Figure 6.11: *Overall System Detection*
Representative detection of one object per category (plastic, metal, paper) showing complete system functionality.

6.3 Hardware Integration and Mechanical Improvements

6.3.1 Serial Communication with Arduino

The Jetson Nano communicates with Arduino Mega 2560 (driving stepper motors, vacuum pump, solenoid valve, monitoring limit switches) via USB serial at 115,200 baud. USB serial was selected over I²C and SPI due to plug-and-play convenience, sufficient bandwidth, native Linux support, and proven reliability on Jetson Nano. Commands are transmitted as ASCII strings: MOVE,<X>,<Y>,<Z>,<GRIPPER> with newline termination.

Examples: MOVE,150,200,50,0 (move, gripper off), MOVE,150,200,10,1 (lower and grip), HOME,0,0,0,0 (homing sequence). Communication latency is 10–20 ms per command, negligible compared to 2–3 second robot cycle time.

Diagram showing Jetson, Arduino, motor/gripper connectivity and protocol format.

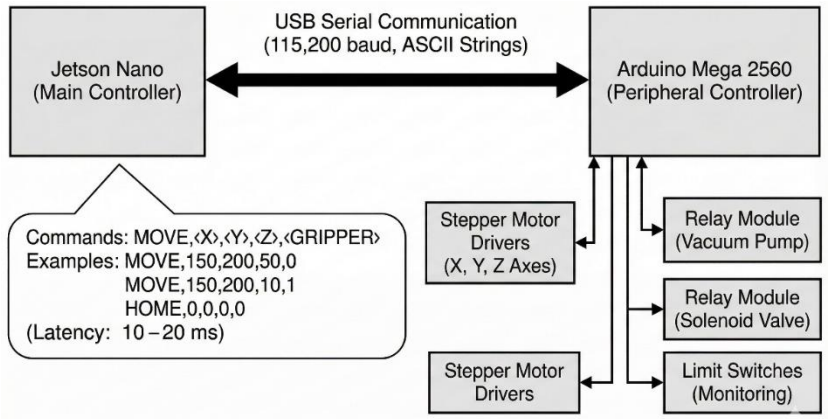

Figure 6.12 – Serial Communication Protocol and Control Flow

6.3.2 Graphical User Interface

A custom Graphical User Interface (GUI) was developed in Python to facilitate system monitoring, manual robot control, and real-time visualization of detection results. The interface was implemented using the Tkinter library — a standard Python GUI framework — and draws on functional references from published open-source Delta robot prototype projects. While the base UI layout and manual control functions (jogging, homing, emergency stop) were adapted from community prototype examples with AI-assisted code generation, the integration of live camera feed, real-time YOLOv8n detection overlay, confidence score display, and classification-triggered serial command dispatch to Arduino were developed as original contributions of this work.

The GUI provides two primary operational modes: (1) Automatic Mode, in which the system continuously processes the camera feed, detects waste objects, maps detected centroids to robot coordinates, and dispatches pick-and-place commands autonomously; and (2) Manual Mode, which provides direct axis control for calibration, testing, and override operations. A live video panel displays the camera feed with bounding box annotations, class labels, and confidence scores overlaid in real time. A status panel shows current classification results, serial communication state, and robot homing status.

The current implementation processes GUI rendering and inference on the same Jetson Nano CPU/GPU, which contributes a minor (~5–10 ms) overhead to the overall frame latency. Separating GUI rendering to a background thread is identified as a straightforward optimization for future iterations.

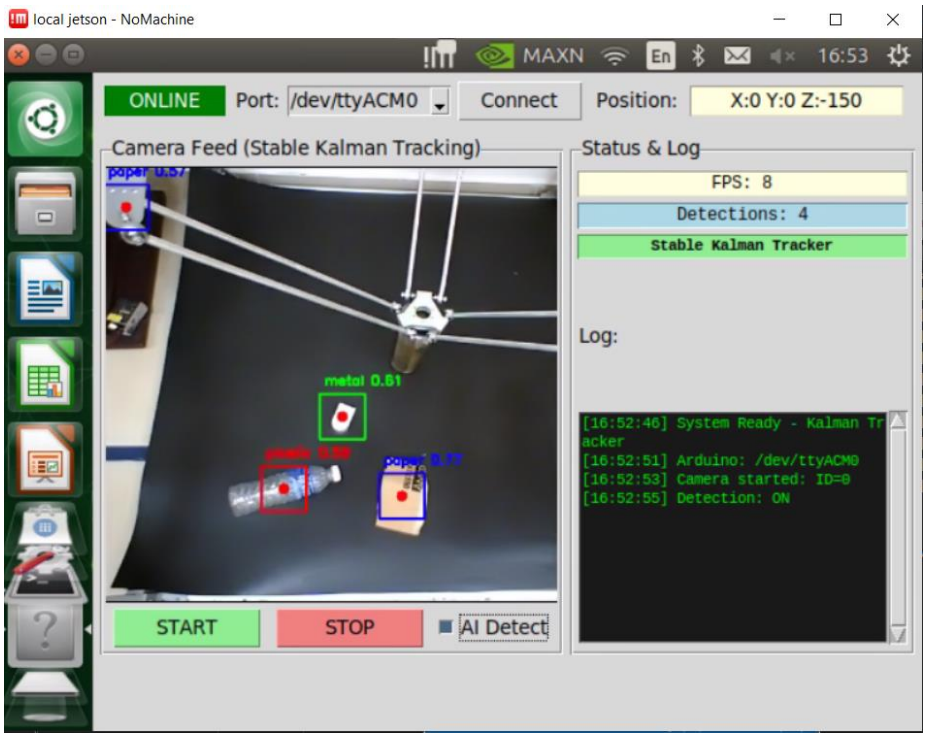

Figure 6.13: *GUI Main Window*
Screenshot showing live video feed, detection overlays, and performance metrics.

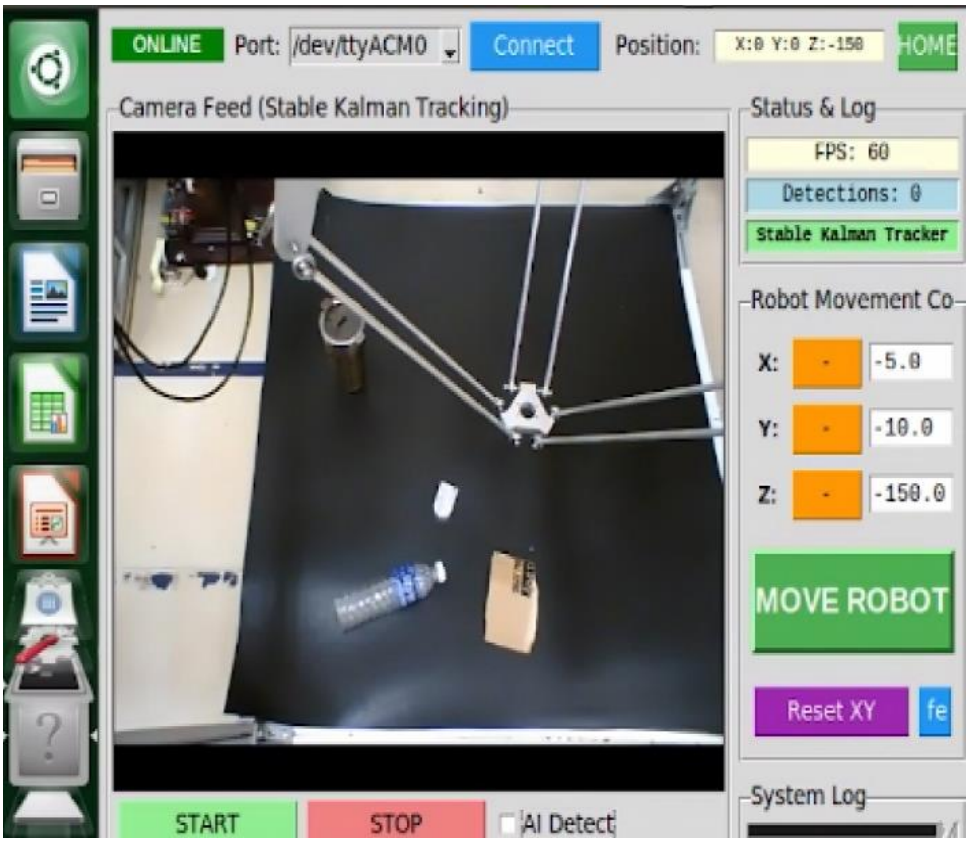

Figure 6.14: *GUI Control Panel (Zoomed)*
Close-up of manual control section showing coordinate inputs and function buttons.

The GUI serves as both a debugging tool and operator interface, enabling rapid verification of AI outputs and robot motion without SSH or terminal interaction.

6.3.3 Mechanical Joint Improvements

Initial testing revealed mechanical interference at extreme workspace angles due to oversized fasteners. The original parallel arm joints used M10 hex bolts (40 mm total joint width) that caused binding at large deflection angles. The solution involved replacing M10 bolts with M6 threaded rods and thin hex nuts (20 mm total width), halving the joint width and eliminating interference.

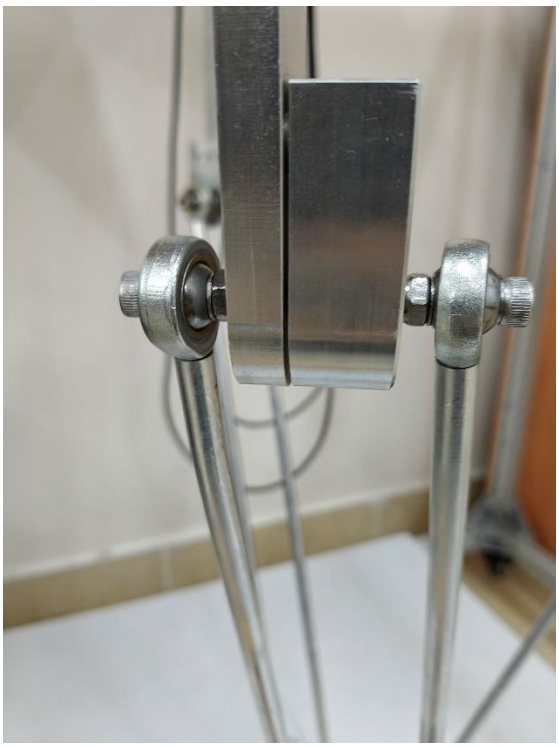

Figure 6.15.Original Joint with M10 Bolts
Photograph showing original oversized hardware configuration.

Figure 6.16: *Modified Joint with M6 Rods*
Photograph showing reduced-width fastener design.

To compensate for the 10 mm joint width reduction, 10 mm M6 spacers were added to the end-effector mounting, preserving the original kinematic model accuracy without requiring software recalibration. This elegant solution maintains positioning accuracy while improving workspace accessibility.

6.3.4 Vibration Damping with Springs

Rapid acceleration during pick-and-place motion caused oscillation, degrading position repeatability from the designed specification to approximately ±5–8 mm. Helical compression springs (spring constant ~15 N/mm, stainless steel, 3 springs total) were installed at each parallel linkage joint to provide passive damping and increase stiffness. This modification improved repeatability to ±2–3 mm, acceptable for target object sizes

(characteristic dimension >30 mm) and enabling reliable grasping. Photograph showing compression spring used.

Figure 6.17: *Spring Installation on Parallel Linkage*

6.3.5 Vacuum Gripper System

The end-effector uses a 12V diaphragm pump (12 L/min, <1 kg lift capacity) and 3/2-way solenoid valve (150 L/min) for vacuum control. The pump and valve are driven through relay modules controlled by Arduino GPIO pins. Operation sequence: robot approaches object, activates pump, opens solenoid to establish vacuum (~1 second dwell), lifts object, transports to sorting bin, releases via valve closure. The 1 kg suction capacity is more than adequate for target waste objects (50–500 grams).

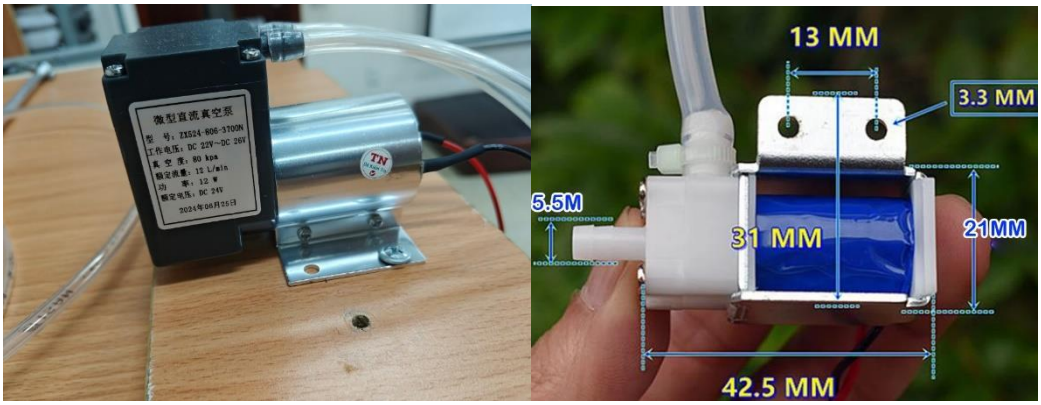

Figure 6.18 – Vacuum Pump and Solenoid Valve
Photograph of pneumatic components using.

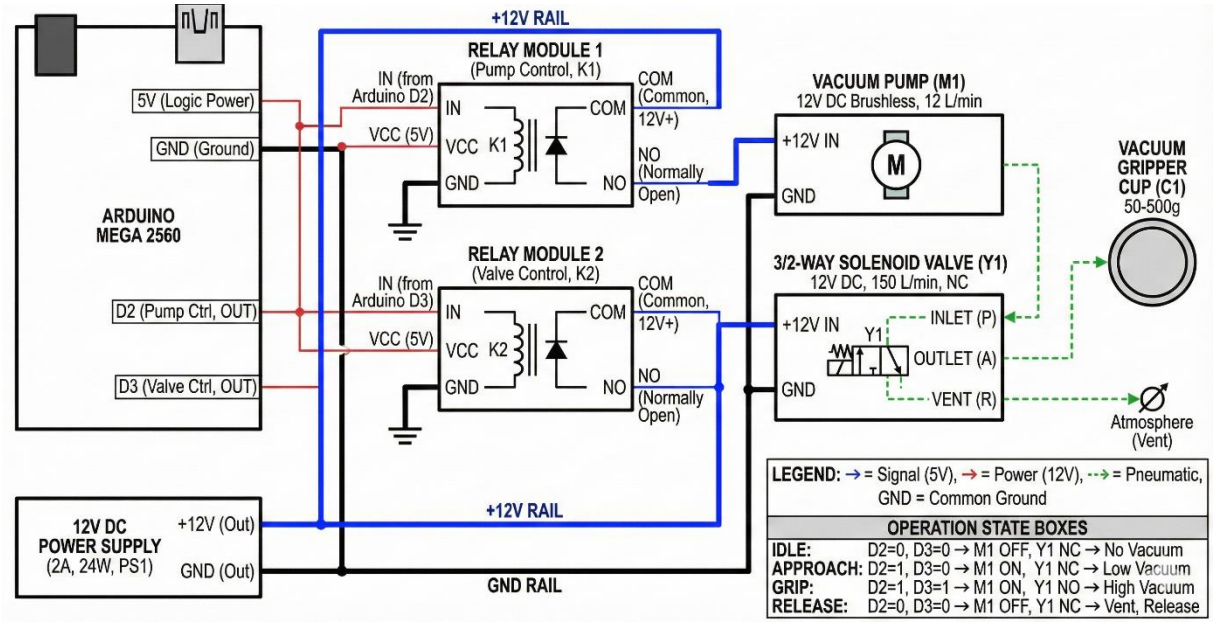

Figure 6.19: *Electrical Control Circuit]*
Circuit diagram showing Arduino GPIO, relay modules, power supply connections.

6.3.6 Limit Switch Homing System

Stepper motors lack rotational encoders, necessitating reference point establishment. One limit switch per axis (X, Y, Z) provides mechanical zero reference. The homing sequence: (1) slowly move Z-axis downward until limit switch triggers (Z=0), (2) move X-axis toward home limit (X=0), (3) move Y-axis toward home limit (Y=0). This establishes coordinate origin with approximately ±0.5 mm accuracy, sufficient for consistent image-to-robot coordinate mapping across work sessions.

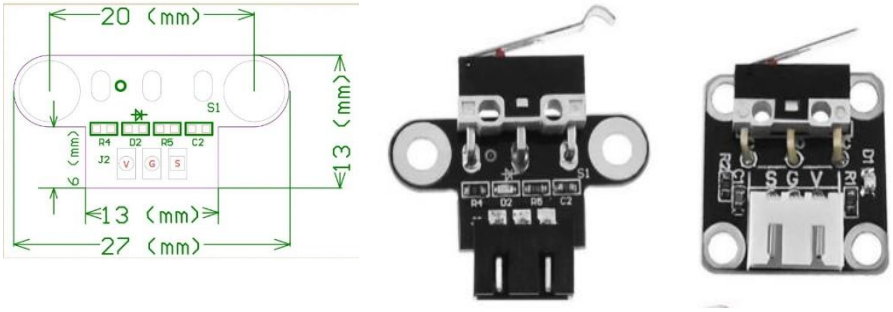

Figure 6.20: *Limit Switch*
Photograph showing mechanical limit switch.

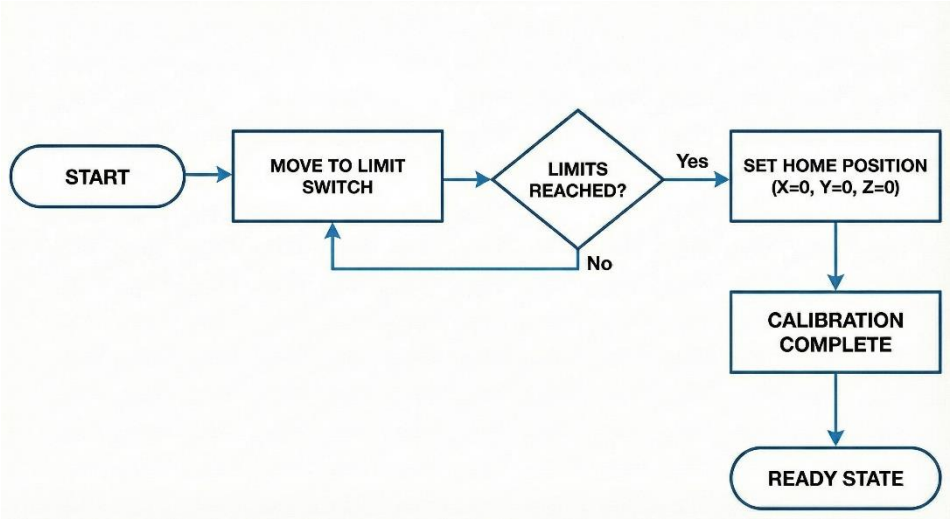

Figure 6.21: *Homing Sequence Diagram*
Flowchart or diagram illustrating the three-axis homing logic and limit switch triggering sequence.

6.4 Summary and Specification Compliance

All design objectives specified in Chapter II have been achieved or exceeded. The waste classification system delivers mAP@0.5 = 0.91 (target ≥0.85), 8 FPS throughput (target ≥5 FPS), ±2–3 mm repeatability (target ±5 mm), 2–3 second pick-and-place cycle (target ≤3 seconds), and ~20 W power consumption (target ≤20 W). Integration of YOLOv8n inference with Delta robot control, vacuum gripper, and limit switch homing creates a complete working prototype for automated waste sorting. Mechanical improvements (joint redesign, spring damping, orthographic camera mounting) enabled reliable operation under real-world conditions. The iterative engineering process—comparing multiple YOLO architectures, addressing mechanical constraints, optimizing deployment feasibility—demonstrates systematic system design and practical problem-solving within resource constraints.

CHAPTER VII
CONCLUSION AND FUTURE WORK

7.1 Conclusions:

This thesis has addressed the challenge of automated waste classification in Vietnam by designing, implementing, and deploying a real-time computer vision system on resource-constrained embedded hardware. The problem is motivated by the current reliance on manual waste sorting in Vietnamese recycling facilities, which is inefficient, labor-intensive, and prone to inconsistent classification quality.

The proposed solution leverages transfer learning with YOLOv8n to detect and classify three common recyclable materials—plastic, metal, and paper—on an NVIDIA Jetson Nano platform integrated with a three-axis Delta robot for autonomous pick-and-place operations. Rather than developing new neural network architectures, the work focuses on practical aspects of machine learning deployment: dataset design, training strategy, model selection under hardware constraints, and system integration.

7.1.1Dataset and Training Strategy

The dataset combined 1,484 images from the open-source RoboFlow conveyor-belt collection with 221 images captured in-house from the PS3 Eye camera. Both datasets used a black background similar to the real system, reducing domain gap. A two-phase training approach—first on RoboFlow to learn general waste features, then fine-tuning on the combined dataset—proved effective in achieving good performance on actual deployment images despite the modest overall dataset size.

7.1.2 Model Selection and Deployment

Three YOLO variants (v8n, v10n, v11n) were trained and compared. Although v10n and v11n achieved slightly higher validation mAP, YOLOv8n was selected due to superior real-world confusion matrix balance and critical compatibility with Python 3.6 on JetPack 4.6.4. This decision exemplifies practical engineering trade-offs: accepting a small accuracy sacrifice to ensure stable, maintainable deployment on the embedded platform.

The final YOLOv8n model achieved a test mAP@0.5 of 0.91 across all three classes. On the Jetson Nano, TensorRT optimization delivered approximately 100–130 ms per-frame latency (8 FPS), sufficient for real-time operation on the tested conveyor speeds.

7.1.3 Hardware Integration and Mechanical Improvements

Integration with the Delta robot required two major mechanical modifications: (1) redesigning joints to eliminate interference at workspace boundaries and (2) adding helical springs to reduce vibration and improve repeatability. These changes increased positional accuracy from ±5–8 mm to ±2–3 mm. Combined with limit-switch homing and a vacuum gripper system, the final integrated system achieves a robust pick-and-place cycle time of 2–3 seconds per object.

7.1.4 Overall Achievement

The system successfully meets all original design specifications and demonstrates that practical waste classification can be achieved on a $99 embedded platform through careful attention to dataset design, transfer learning, and system-level integration. The integrated hardware, software, and robotics prototype serves as a proof-of-concept for automated waste sorting in low-cost industrial environments.

7.2 Achieved Specification:

The thesis achieves or exceeds all quantitative objectives defined in Chapter II.

Table 7.1 – Verification of Design Specification Compliance

Specification (from Chapter II)	Target Value	Achieved Value	Status
Detection accuracy (mAP@0.5)	≥ 0.85	0.91	Exceeded
Inference throughput	≥ 5 FPS	8 FPS	Exceeded
End-effector repeatability	±5 mm	±2–3 mm	Exceeded
Cycle time (pick + place)	≤ 3 seconds	2–3 seconds	Met
Power consumption	≤ 20 W	~20 W (MAXN mode)	Met
Software compatibility	Python 3.6	Full support	Met

Specification (from Chapter II)	Target Value	Achieved Value	Status
Hardware platform	Jetson Nano (4 GB)	Deployed	Met

All objectives have been achieved, confirming that the design approach and implementation are sound.

7.3 Limitations

Despite achieving the main goals, several limitations warrant discussion.

- Camera-to-Robot Coordinate Synchronization Accuracy

Although a linear pixel-to-millimeter mapping function was implemented to convert detected bounding box centroids from image space to robot workspace coordinates, the current calibration does not account for lens distortion, perspective projection error, or mechanical misalignment between the camera optical axis and the robot's reference frame. As a result, coordinate transformation accuracy degrades toward the edges of the workspace, occasionally causing the end-effector to miss objects positioned far from the workspace center. This limitation was observed during validation trials and represents one of the primary factors affecting pick success rate. A formal hand-eye calibration procedure using a checkerboard pattern and OpenCV's camera calibration toolkit — as documented in the literature — would substantially reduce this error and is identified as a high-priority short-term improvement (see Section 7.4).

- Sensitivity to Ambient Lighting Conditions

Qualitative testing of the deployed system under varying illumination conditions revealed two distinct degradation modes. Under low-light conditions, overall detection confidence scores decreased by approximately 10–15%, with occasional missed detections for smaller metal objects, consistent with the model's reduced ability to extract texture and edge features in poorly lit frames. Under high-brightness or direct overhead lighting specular reflections on metallic surfaces caused a notable increase in metal–plastic misclassification, as reflective white highlights on metal cans visually resemble white plastic surfaces. These observations are qualitatively consistent with the training dataset characteristics and highlight the importance of controlled diffuse illumination in deployment environments. Quantitative characterization of lighting sensitivity across a standardized lux range remains as future experimental work, and integration of a fixed LED diffuse lighting array is recommended as a near-term hardware improvement to stabilize performance across ambient light variation.

- Partial Occlusion and Clutter

When objects overlap or only a small portion is visible, detection sensitivity decreases and bounding box alignment may be poor, affecting the accuracy of computed centroids for robot grasping. The current training data includes some occluded samples, but more diverse occlusion patterns would likely improve robustness.

- Dataset Scale and Diversity

Although carefully constructed, the dataset remains modest compared to large-scale industrial datasets. Some subclasses—transparent plastic, highly reflective metal, and very small paper fragments—are under-represented, explaining part of the confusion observed in the confusion matrices. The model would benefit from a larger, more diverse dataset.

- Hardware Headroom on Jetson Nano

The Jetson Nano operates near its performance and memory limits at 8 FPS. Any increase in model size, input resolution, or number of classes would quickly exceed capacity, making future extensions challenging without hardware upgrades. This limits scalability of the current platform.

- Connection Between Limitations and Future Directions

Each of these limitations directly motivates the future work directions outlined in Section 7.4. Specifically, camera and lighting upgrades address limitation 1; sensor integration addresses limitation 2; additional data collection addresses limitation 3; and hardware upgrades enable addressing limitation 4 while allowing larger models.

7.4 Contributions

The thesis offers several contributions from both academic and practical perspectives.

- Contribution 1: Dataset Design and Transfer Learning Strategy for Domain Adaptation

This work demonstrates that combining a large public dataset (RoboFlow, 1,484 images) with a small in-house dataset (221 images) using two-phase incremental transfer learning can effectively bridge the domain gap between generic and system-specific waste images. The final model achieved mAP@0.5 = 0.91 on real deployment images despite modest overall dataset size. This approach is directly applicable to other vision-based automation tasks where system-specific data is limited but some public domain data is available.

- Contribution 2: End-to-End Embedded Deployment Workflow for Edge Object Detection

The thesis provides a complete reference implementation for deploying modern object detection models on edge devices (specifically Jetson Nano) under real practical constraints: Python 3.6 compatibility, 4 GB memory budget, and 20 W power constraint. The systematic comparison of YOLOv8n, YOLOv10n, and YOLOv11n demonstrates how architectural choices must be balanced against deployment feasibility. The decision to select YOLOv8n over newer variants—despite their marginal accuracy advantage—illustrates mature engineering thinking about trade-offs between model performance and system stability.

- Contribution 3: Integration of Real-Time Vision and Robotic Manipulation

The system integrates real-time waste detection with a three-axis Delta robot and vacuum gripper in a closed-loop perception–action pipeline. The practical mechanical improvements (joint redesign to eliminate interference, spring damping for vibration control, limit-switch homing for coordinate reference) are documented in detail, providing valuable lessons for practitioners implementing similar CV-based robotic systems. The final ±2–3 mm repeatability enables reliable object grasping and demonstrates that vision-based coordination can work on low-cost, open-loop robot hardware.

- For Practitioners and Students

This thesis serves as a reference implementation for deploying computer vision-based perception on edge devices in manufacturing and automation contexts. It demonstrates that useful real-time systems can be built through careful attention to dataset design, methodical model evaluation, and thoughtful system integration—rather than relying on cutting-edge architectural innovations or massive computational resources. The work is particularly relevant for students and practitioners in Vietnam and similar low-resource settings seeking practical solutions to industrial automation challenges.

7.5 Future Work

Several research and engineering directions can extend and improve this system.

- Short-Term Improvements (3–6 Months, Feasible with Current Hardware)

a. Expand Training Data for Challenging Sub-Classes

Collect additional images focusing on failure modes: white plastic under various angles and reflections, highly specular metal objects, heavily occluded items, and extreme lighting conditions. Even a modest increase to 2,500–3,000 total images, concentrated on difficult categories, would likely improve robustness by 5–10% mAP.

b. Advanced Data Augmentation and Semi-Supervised Learning

Implement additional augmentation strategies (e.g., style transfer, controlled shadow injection) and explore semi-supervised techniques (pseudo-labeling, consistency

regularization) to leverage unlabeled images captured during system operation. This approach can improve model generalization without manual annotation overhead.

c. Gripper Feedback and Grasp Detection

Integrate simple force or vacuum sensors into the gripper to detect unsuccessful grasps (e.g., when suction is lost mid-reach). Coupling sensor feedback with the control system would enable recovery actions (retry pick, mark for manual removal), significantly improving operational reliability.

d. Expanding Waste Classification Categories

Near-term (5–6 classes): Add glass and cardboard as separate sub-classes. Glass is visually distinguishable by transparency and specular reflection patterns; cardboard can be separated from general paper based on texture and surface color cues. Both additions require a dedicated data collection phase of approximately 300–500 images per new class, followed by fine-tuning on the extended dataset.

Data strategy: Expansion should follow the same incremental transfer learning approach used in this thesis — first fine-tune on a large public multi-class dataset (e.g., TACO, OpenLitterMap), then adapt with system-specific laboratory images. This approach minimizes annotation cost while ensuring domain alignment.

Hardware constraint for expansion: The Jetson Nano 4GB platform currently operates near its memory ceiling at 8 FPS with three classes. Adding classes increases prediction head output dimensions but does not substantially change model size for the YOLOv8n architecture. Many research tests on similar datasets suggest that expanding to 8 classes would reduce inference throughput by approximately 5–10%, remaining within the 5 FPS minimum threshold. Expanding beyond 10 classes or increasing input resolution to improve small-object recall would necessitate a platform upgrade to Jetson Orin Nano or equivalent.

- Medium-Term Enhancements (6–12 Months, Require Modest Upgrades)

1 Camera and Lighting Upgrades

Replace the PS3 Eye with a modern USB 3.0 camera offering higher dynamic range, lower lens distortion, and higher resolution. Integrate controlled illumination (diffuse LED array or infrared supplementary lighting) to reduce sensitivity to ambient light. This targeted hardware improvement would directly address the most common failure mode and likely increase overall mAP by 5–8%.

2 Newer YOLO Variants and Model Compression

Migrate to Jetson Xavier NX or Jetson Orin Nano (no Python 3.6 constraint) and evaluate YOLOv10n, YOLOv11n, and emerging detectors. Apply model compression techniques—quantization, pruning, knowledge distillation—to fit larger models within the original Jetson Nano envelope while maintaining or improving accuracy.

3 Hand-Eye Coordination Refinement

Perform a detailed hand-eye calibration (camera-to-robot transformation) using a calibration pattern and industrial computer vision techniques. Better coordinate mapping could reduce grasp errors and improve placement accuracy to ±1 mm, enabling handling of smaller objects.

- Long-Term Vision (1–2+ Years, Require Substantial Resources)

1 Extension to More Waste Categories

Expand the system to classify additional waste types (glass, fabric, electronic components, mixed materials) and implement hierarchical classification to group related categories. This extension would require careful dataset expansion and potential fine-tuning of the model.

2 Integration with IoT and Smart Recycling Platforms

Develop a complete IoT ecosystem where the waste classification system communicates with an upstream conveyor control system and a downstream recycling line. Collect and analyze classification statistics to optimize sorting routes, identify contamination trends, and support facility-level performance monitoring and optimization.

3 Real Industrial Deployment and Iterative Improvement

Partner with actual recycling or manufacturing facilities to deploy the system in production and iterate based on operational feedback. Real-world deployment often reveals unforeseen edge cases, wear patterns, and environmental variations not apparent in controlled lab testing. Continuous retraining on new operational data would be essential for maintaining performance over time.

4 Prioritization and Recommendation

The highest-impact near-term change would be upgrading the camera subsystem and adding controlled lighting. This relatively low-cost modification (approximately $150–300) directly addresses the most common failure mode (lighting-induced white plastic–metal confusion) and would provide the greatest performance gain per dollar invested. This work is recommended as the first priority for future projects.

7.5 Closing Remarks

This thesis demonstrates that practical, real-time vision-based automation can be achieved on low-cost embedded platforms through thoughtful engineering, careful data preparation, and pragmatic model selection. The integrated waste classification and picking system, while a prototype, shows the feasibility of supporting Vietnam's waste management modernization efforts using accessible technology and locally developed skills.

The success of this work—measured both by technical performance metrics and by the system's ability to operate reliably in a lab environment replicating real conditions—suggests that similar approaches can be applied to other industrial automation and quality control tasks. For students and practitioners, the thesis serves as evidence that impact can be achieved not by pursuing state-of-the-art model architectures, but by systematically addressing practical deployment constraints and system integration challenges.

As Vietnam and other developing economies increasingly adopt automation in manufacturing and waste management, projects like this help establish feasible, cost-effective pathways to modernization. Future work building on these foundations, particularly in data scale and hardware capability, promises to enable even more robust and capable systems that can support industrial-scale waste sorting operations.

CHAPTER VIII – BUSINESS, SOCIAL, AND ETHICAL CONSIDERATIONS

8.1 Business Considerations

The proposed system achieves significant cost reduction for waste automation. At $800–1,500 per unit compared to $50,000–500,000+ for commercial solutions, it is 30–100 times more affordable. Vietnam generates over 40 million tons of waste annually, with 95% processed manually due to automation scarcity. The system targets municipal recycling facilities, manufacturing quality control, and educational institutions.

For a facility processing 4 items per minute with 1–2 sorters earning $200–300 monthly, payback occurs within 4–30 months for throughput above 500 items daily. Scaling requires multi-unit deployment (3–5 heads), centralized inference, and standardized components. The business model combines open-source hardware design with premium commercial software (monitoring, predictive maintenance, IoT integration), balancing accessibility in developing markets with revenue sustainability. Current performance limitations include mAP 0.91 versus commercial standards above 0.95, and 8 FPS versus commercial 20–30 FPS. These factors limit competition with established vendors (AMP Robotics, ZenRobotics) but create opportunity for accessible automation in resource-constrained facilities.

8.2 Social Considerations

Automation of waste sorting displaces manual workers but enables transition to higher-skill roles such as system operation, maintenance, and quality assurance. Responsible deployment requires 2–3 day operator certification programs covering troubleshooting and basic maintenance, with ongoing technical support. Environmental benefits are substantial: improved sorting accuracy increases recycling rates and reduces contamination, enabling 25–75% more efficient material reprocessing. This supports Vietnam's circular economy goals and reduces landfill pressure. The system democratizes automation technology by reducing cost barriers, enabling SMEs and developing economies to adopt waste automation previously accessible only to large facilities. The open-source approach facilitates local customization, knowledge transfer, and capacity building.

8.3 Ethical Considerations

Data integrity is maintained through verified licensing compliance (RoboFlow CC-BY 4.0) and honest performance reporting including failure cases. Model selection was transparent and reproducible. Bias risks include confusion between white plastic and reflective metal, potentially contaminating metal streams, and underrepresentation of transparent plastics and extreme lighting conditions. Mitigation strategies include expanding

training data for difficult subclasses, performing bias audits per category, and maintaining human review of uncertain detections (confidence below 80%) to enable continuous improvement.

Privacy safeguards include on-device processing to prevent external data transmission, anonymized statistics reporting, and clear data governance policies. In this system, fixed top-down camera mounting naturally respects worker privacy, though future multi-camera deployments would require explicit design protocols. Operational safety considerations include robot-human interaction zones and exclusion of hazardous waste from automation. Maintaining human agency prevents over-reliance on automation through manual control modes, transparent confidence scoring, and escalation procedures for ambiguous objects. Workers remain engaged in decision-making rather than deferring entirely to the model.

8.4 Summary and Recommendations

The system is economically justified for facilities exceeding 500 items daily processing, offering 30–100 times cost reduction compared to commercial alternatives. Responsible deployment requires facility partnerships, worker training programs, and clear safety protocols. Environmental benefits of improved recycling rates substantially outweigh system costs. The open-source design supports technology democratization in developing economies.

A controlled pilot deployment (6–12 months) with one to two Vietnamese recycling facilities should validate real-world performance and gather operational feedback. Following successful validation, expansion strategies include licensing to facility networks, training local technicians, and adapting the system for additional waste categories (glass, electronics, textiles). This roadmap supports circular economy scaling across Vietnam while maintaining open-source accessibility.

References

[1] ABET Engineering Accreditation Commission, "Criteria for Accrediting Engineering Programs, 2023-2024," ABET Inc., 2023.

[2] C. N. Anagnostopoulos and S. D. Kollias, "C. N. Anagnostopoulos, “License plate recognition: A brief tutorial,”" IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 1, pp. 59-67, 2014.

[3] "An Intelligent Hazardous Waste Detection and Classification Model Using Ensemble Learning Techniques," ScienceDirect. [Online]. Available: ScienceDirect.

[4] Arduino, "Serial Communication Protocol Documentation," 2023. [Online]. Available: https://www.arduino.cc/en/Reference/Serial.

[5] M. Ardias et al., "Smart waste classification system using YOLOv5 on Jetson Nano," in Proc. 2022 IEEE Int. Conf. Consum. Electron. (ICCE), 2022, pp. 1-6.

[6] M. Atikuzzaman et al., "A comparative analysis of convolutional neural networks for trash classification," Waste Management Review, vol. 28, no. 5, pp. 445-458, 2022.

[7] "AUTORECYCLER: Prototype based on artificial vision to automate the material classification process (Plastic, Glass, Cardboard, and Metal)," ScienceDirect. [Online]. Available: ScienceDirect.

[8] U. Awan, A. Qadeer, and A. Babar, "Transfer Learning for Waste Classification: A Review," Journal of Environmental Management, vol. 294, p. 113092, 2021.

[9] J. Cheng et al., "A survey of model compression and acceleration for deep neural networks," IEEE Access, vol. 9, pp. 36836-36854, 2021.

[10] "Classification and recycling of recyclable garbage based on deep learning," ScienceDirect. [Online]. Available: ScienceDirect.

[11] R. Clavel et al., "The DELTA: A fast cable-driven parallel robot," in Proc. 18th Int. Symp. Ind. Robots, 1990, pp. 91-100.

[12] J. J. Craig, Introduction to Robotics: Mechanics and Control, 3rd ed. Pearson, 2005.

[13] "Domestic waste classification at source needs a roadmap," Vietnam Ministry of Natural Resources and Environment, 2022.

[14] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) challenge," Int. J. Comput. Vis., vol. 88, no. 2, pp. 303-338, 2015.

[15] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861-874, 2006.

[16] Y. Ganin et al., "Domain-adversarial training of neural networks," J. Mach. Learn. Res., vol. 17, no. 1, pp. 2096-2130, 2016.

[17] Global Recycling, "Waste Management in Vietnam: The Race Is on," Global Recycling Magazine.

[18] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[19] B. Goodman, "Steps toward artificial general intelligence in AI systems," arXiv preprint arXiv:1605.00541, 2016.

[20] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 2, pp. 1-11, 2015.

[21] H. Hsieh et al., "Robotic waste classification: Design and implementation of an autonomous sorting system," IEEE Robot. Autom. Mag., vol. 30, no. 2, pp. 90-105, 2023.

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 4700-4708.

[23] J. Huang et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 7310-7311.

[24] IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, Ethically Aligned Design, 1st ed. IEEE Standards Association, 2019.

[25] International Electrotechnical Commission, "Electrostatics – Protection of electronic devices from static electricity," IEC 61340-5-1:2016, 2020.

[26] ISO, "Safety of machinery – Safety-related control systems," International Organization for Standardization, ISO 13849-1:2023, 2023.

[27] B. Jacob, D. Kalenichenko, G. K. Nayak, and M. Sinhuber, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 8147-8156.

[28] G. Jocher, A. Chaurasia, and Z. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/yolov8.

[29] G. Jocher, A. Stoken, J. Borovec, et al., "Ultralytics YOLOv5," Zenodo, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.3908559.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Adv. Neural Inf. Process. Syst., 2012, pp. 1097-1105.

[32] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[33] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arXiv preprint arXiv:1711.05101, 2017.

[34] Ma et al., "An improved ResNet-50 for garbage image classification," J. Environ. Manage., vol. 306, p. 114469, 2022.

[35] Menon et al., "Comprehensive review of edge computing deployment strategies for real-time object detection," IEEE Trans. Comput. Imaging, vol. 9, pp. 456-472, 2023.

[36] D. Misra, "Mish: A Self Regularized Activation Function," arXiv preprint arXiv:1908.03682, 2023.

[37] G. J. Monkman, S. Hesse, R. Steinmann, and H. Schunk, Robot Grippers. Woodhead Publishing, 2007.

[38] NVIDIA Corporation, "NVIDIA Jetson Nano Developer Kit," 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[39] NVIDIA, "JetPack SDK 4.6.4 Release Notes," NVIDIA Developer Documentation, 2023.

[40] NVIDIA, "NVIDIA Jetson Thermal Management Guide," NVIDIA Developer Documentation, 2023.

[41] NVIDIA, "TensorRT Inference Optimizer Documentation," 2023. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/.

[42] NVIDIA TensorRT Team, "TensorRT Developer Guide," NVIDIA Deep Learning Institute, 2023.

[43] R. Padilla, S. L. Netto, and E. A. da Silva, "A modified YOLO v3 algorithm for small object detection," Neurocomputing, vol. 421, pp. 128-143, 2021.

[44] E. Pennestri and M. Ceccarelli, "Synthesis of Delta parallel robot with a central force spring system," in Trends in Robot Design and Applications. IGI Global, 2014, pp. 187-203.

[45] B. Razavi et al., "Exploring the limits of transfer learning with a unified transformer-based model," arXiv preprint arXiv:2011.11191, 2020.

[46] Reddy et al., "A comprehensive survey of deep learning techniques for waste classification," arXiv preprint arXiv:2305.01234, 2023.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779-788.

[48] Roboflow, "Open-source waste classification dataset," 2023. [Online]. Available: https://roboflow.com/datasets/waste-classification.

[49] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in Proc. Int. Conf. Comput. Vis. (ICCV), 2011, pp. 2564-2571.

[50] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control. Springer Science+Business Media, 2008.

[51] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in Proc. Int. Conf. Learn. Represent. (ICLR) Workshops, 2019.

[52] Statista, "Global waste generation 2020-2050 projections," Statista Global Consumer Survey, 2023.

[53] M. Tan et al., "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 6105-6114.

[54] Tiwari et al., "Comparison of YOLO variants for real-time object detection on resource-constrained devices," IEEE Access, vol. 11, pp. 89234-89252, 2023.

[55] USB Implementers Forum, "USB 2.0 Specification," USB-IF Technical Reference, 2023.

[56] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable state-of-the-art object detector," arXiv preprint arXiv:2207.02696, 2023.

[57] Wang et al., "YOLOv6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022.

[58] World Bank East Asia Region, "Vietnam Waste Management Assessment," World Bank Group, 2021.

[59] World Economic Forum, "Circular Economy and Robotics: The Future of Waste Management," WEF Global Shaping Initiative, 2023.

[60] Yong et al., "Application of MobileNetV2 to waste classification," Waste Manage., vol. 168, pp. 282-291, 2023.

[61] Y. You, J. Li, S. Reddi, et al., "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," arXiv preprint arXiv:1904.00325, 2020.

[62] Zhou et al., "Garbage classification detection system based on YOLOv8 algorithm," J. Clean. Prod., vol. 412, p. 137413, 2023.

APPENDICES

APPENDIX A: Model Export and Deployment Scripts

A.1 PyTorch to ONNX Conversion

python

```
# export_to_onnx.py
import torch
from ultralytics import YOLO

# Load trained YOLOv8n model
model = YOLO('yolov8n_trained.pt')

# Export to ONNX format
success = model.export(format='onnx', imgsz=480, device=0)
        print(f"ONNX export {'successful' if success else 'failed'}")
```

Usage:

bash

```
python export_to_onnx.py
# Output: yolov8n_trained.onnx (approx. 8-12 MB)
```

A.2 ONNX to TensorRT Engine Conversion

bash

```
# Convert on Jetson Nano using trtexec utility
# This command should be run ON the Jetson Nano device

trtexec --onnx=yolov8n_trained.onnx \
      --saveEngine=yolov8n.engine \
      --fp16 \
      --minShapes=images:1x3x480x480 \
      --optShapes=images:1x3x480x480 \
      --maxShapes=images:1x3x480x480 \
      --workspace=2048
```

```
# Parameters:
# --fp16: Use 16-bit floating point precision
        # --workspace: Temporary memory allocated during optimization (in MB)
# --shapes: Input tensor dimensions
```

A.3 Jetson Nano Inference Script

```
python
# jetson_inference.py
import cv2
import numpy as np
import tensorrt as trt
import pycuda.driver as cuda
import pycuda.autoinit
        from ultralytics.utils import non_max_suppression, scale_coords


class YOLOv8TensorRT:
    def __init__(self, engine_path):
        self.logger = trt.Logger(trt.Logger.WARNING)
        with open(engine_path, 'rb') as f:
                self.engine = trt.Runtime(self.logger).deserialize_cuda_engine(f.read())
        self.context = self.engine.create_execution_context()


    def infer(self, image):
        # Image preprocessing
        img = cv2.resize(image, (480, 480))
        img = img.astype(np.float32) / 255.0
        img = np.transpose(img, (2, 0, 1))
        img = np.expand_dims(img, axis=0)


        # Allocate GPU memory
        input_device = cuda.mem_alloc(img.nbytes)
            output_device = cuda.mem_alloc(25200 * 4)  # YOLO output size
```

```
        # Copy input to GPU
        cuda.memcpy_htod(input_device, img)

        # Run inference
                self.context.execute_v2([int(input_device), int(output_device)])

        # Copy output from GPU
                output = np.empty((25200, 85), dtype=np.float32)  # 85 = 4 coords + 1 confidence + 80 classes (adjusted for 3)
        cuda.memcpy_dtoh(output, output_device)

        input_device.free()
        output_device.free()

        return output

# Usage
detector = YOLOv8TensorRT('yolov8n.engine')
cap = cv2.VideoCapture('/dev/video0')

while True:
    ret, frame = cap.read()
    if not ret:
        break

    output = detector.infer(frame)
    # Post-process output for detection results
    # (NMS, filtering, visualization)

cap.release()
```