VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FACULTY OF ELECTRICAL - ELECTRONICS ENGINEERING

LE NHAT TAN

INTELSCHOOLBUS - THE SYSTEM FOR COUNTING AND CONTROLLING STUDENTS ON SCHOOL BUSES

UNDERGRADUATE THESIS OF ELECTRONICS - TELECOMMUNICATION ENGINEERING

HO CHI MINH CITY, YEAR 2024

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF ELECTRICAL - ELECTRONICS ENGINEERING

LE NHAT TAN - 42001078

INTELSCHOOLBUS - THE SYSTEM FOR COUNTING AND CONTROLLING STUDENTS ON SCHOOL BUSES

UNDERGRADUATE THESIS OF ELECTRONICS – TELECOMMUNICATION ENGINEERING

Advised by
Dr. Le Anh Vu

HO CHI MINH CITY, YEAR 2024

ACKNOWLEDGMENT

First and foremost, I would like to extend my deepest gratitude to my undergraduate thesis advisor, Dr. Le Anh Vu. Throughout the process of completing my thesis, I encountered many questions and uncertainties, all of which were met with his patient and enthusiastic guidance. He not only clarified my doubts but also imparted valuable knowledge related to my research topic. I am sincerely thankful for the time and effort he dedicated to helping me understand complex concepts. His passionate mentorship and clear direction played a crucial role in enabling me to successfully complete this significant and meaningful thesis. It is a deeply humane and purposeful project, one I have always aspired to undertake, and I feel incredibly fortunate that he helped me seize this opportunity.

I would also like to express my appreciation to all the faculty members of the Department of Electrical and Electronics Engineering. The foundational knowledge they imparted has equipped me with the understanding needed to pursue advanced research and undertake meaningful projects like this one. Additionally, I am grateful to Mr. Nghia and Mr. Dung, senior peers who generously supported me throughout the process. My special thanks also go to Mr. Tien, Vice Principal of Ruby Hong Ngoc Primary and Secondary School, for providing the necessary support and permissions that allowed me to deploy my system on the school bus. During the system testing phase, Mr. Tri, the school bus driver, offered invaluable assistance with great enthusiasm and I am sincerely thankful for his kindness and cooperation.

Every form of assistance I received from Dr. Le Anh Vu, the faculty members, and my peers is something I will always cherish and hold dear. Once again, I sincerely thank Dr. Le Anh Vu for accepting me as his student and guiding me through my undergraduate thesis with such dedication and care.

Chi Minh City, day month year 2024

Author

Le Nhat Tan

This thesis was carried out at Ton Duc Thang University.

Advisor: ..... Dr. Le Anh Vu .....

.....

(Title, full name and signature)

This thesis is defended at the Undergraduate Thesis Examination Committee was hold at Ton Duc Thang University on … /…/……

Confirmation of the Chairman of the Undergraduate Thesis Examination Committee and the Dean of the faculty after receiving the modified thesis (if any).

CHAIRMAN	**DEAN OF FACULTY**
TS. Nguyễn Hữu Khánh Nhân	
.....	.....

DECLARATION OF AUTHORSHIP

I hereby declare that this thesis was carried out by myself under the guidance and supervision of ....Dr. Le Anh Vu.....; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

I will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

Ho Chi Minh City, day 21 month 12, year 2024

Author

Tân

Le Nhat Tan

ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN –ĐIỆN TỬ

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

TP. HCM, ngày 16 tháng 9 năm 2024

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

(Bản nhiệm vụ này được đóng vào trang thứ nhất Đồ án tốt nghiệp)

Họ tên sinh viên: **Lê Nhật Tân**

Ngành: Kỹ thuật Điện tử - Viễn thông Lớp: **20040201** MSSV: **42001078**

1. Tên đề tài: **Intelschoolbus - hệ thống đếm và kiểm soát học sinh trên xe buýt đưa đón của trường học**

2. Tên đề tài (tiếng Anh): ***Intelschoolbus - the system for counting and controlling students on school buses***

3. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):

- Tìm hiểu các mô hình nhận dạng, bán theo và đếm người lên xuống xe đủ nhẹ để nhúng được xuống phần cứng giá thành rẻ
- Tìm hiểu cơ chế truy cập dữ liệu ngoài mạng LAN hỗ trợ kết nối IoT.
- Tìm hiểu về định vị GPS gắn với API của bản đồ
- Thiết kế tích hợp các module camera, GPS, nguồn và board mạch nhúng đủ nhẹ trên một phần cứng gắn được trên xe buýt đưa đón học sinh.
- Phát triển tích hợp và phần cứng nhúng các thuật toán nhận dạng, bám theo và đếm người, thuật toán giám sát vị trí xe
- Tích hợp phần cứng giải thuật Backend vào cơ sở dữ liệu và giao diện người dùng frontend
- Phân tích và đánh giá kết quả phần cứng và phần mềm
- Viết báo cáo, slides và poster.

4. Ngày giao nhiệm vụ đồ án tốt nghiệp: **16/9/2024.**

5. Ngày bảo vệ 50% đồ án tốt nghiệp: **04/11/2024 - 09/11/2024.**

6. Ngày hoàn thành và nộp về khoa: **05/01/2025.**

7. Giáo viên hướng dẫn: Phần hướng dẫn:

TS. Lê Anh Vũ **100%**

Nội dung và yêu cầu Đồ án tốt nghiệp đã được thông qua Bộ môn và Khoa.

Ngày 16 tháng 9 năm 2024

TRƯỞNG BỘ MÔN

TS. Trần Thanh Phương

GIẢNG VIÊN HƯỚNG DẪN

TS. Lê Anh Vũ

PHỤ TRÁCH KHOA

TS. Trần Thanh Phương

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ

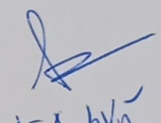
CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

LỊCH TRÌNH LÀM ĐỒ ÁN TỐT NGHIỆP

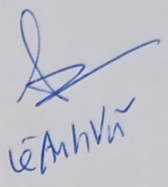
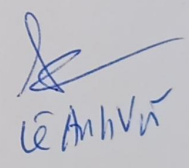
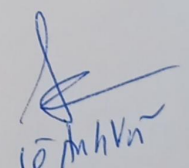
Họ tên sinh viên: Lê Nhật Tân
Lớp: 20040201 MSSV: 42001078
Tên đề tài: Intelschoolbus - hệ thống đếm và kiểm soát học sinh trên xe buýt đưa đón của trường học

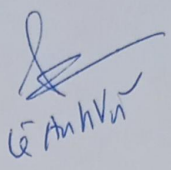
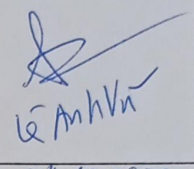
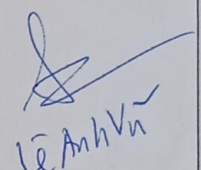
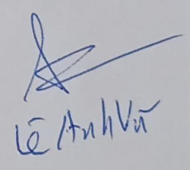
Tuần/Ngày	Khối lượng		GVHD ký
	Đã thực hiện	Tiếp tục thực hiện	
1 (09/09/2024 – 15/09/2024)	- Nhận tờ nhiệm vụ Đồ án - Hiểu rõ yêu cầu nội dung đề tài, quy định thực hiện đồ án	- Tìm hiểu các mô hình nhận dạng, bám theo và đếm người lên xuống xe đủ nhẹ để nhúng được xuống phần cứng giá thành rẻ và các sản phẩm đếm người trên thị trường sau đó đặt mục tiêu về hiệu suất và kích thước cho sản phẩm của đồ án	12/09/2024 Lê Anh Vũ
2 (16/09/2024 – 22/09/2024)	- Thiết kế sơ đồ khối chức năng cho hệ thống và lập danh sách lựa chọn phần cứng sử dụng như: thiết bị nhúng (Raspberry Pi 4 Model B 8GB RAM	- Viết thuật toán đếm vật thể động di chuyển 1 hướng - Chuẩn bị các mô hình để chạy thử nghiệm trên Raspberry Pi	20/09/2024 Lê Anh Vũ

	và Sipeed MaixCAM Development Board NPU RISCV AI), camera (camera module và webcam), module GPS, anten GPS, Bộ Phát Wi-Fi Di Động 4G, sạc dự phòng, hộp nhựa, tản nhiệt và quạt tản nhiệt DC - Thiết lập Raspberry Pi, thử nghiệm hoạt động của camera module hồng ngoại và webcam.		
3 (23/09/2024 – 29/09/2024)	- Thử nghiệm thành công mô hình pre-trained YOLOv8n, mô hình pre-trained YOLOv8s, mô hình pre-trained MobileNet V1 chuyển đổi sang TensorFlow Lite 16-bit and 32-bit floating-point types, mô hình pre-trained YOLOv8n chuyển	- Tiếp tục thử nghiệm các mô hình - Chuẩn bị bộ dữ liệu hình ảnh đầu người trên Roboflow	25/09/2024 Lê Anh Vũ

	đổi sang TensorFlow Lite 16-bit and 32-bit floating-point types trên Raspberry Pi với thuật toán đếm vật thể động di chuyển theo 1 hướng		
4 (30/09/2024 – 06/10/2024)	- Thực hiện gán nhãn bộ dữ liệu hình ảnh đầu người và fine tune với mô hình YOLOv9, sau đó chạy thử nghiệm mô hình này trên Raspberry Pi - Ghi nhận các kết quả thử nghiệm với mô hình pre-trained MobileNet V2 chuyển đổi sang TensorFlow Lite 16-bit and 32-bit floating-point types cùng thư viện PyTorch và thuật toán Faster R-CNN trên Raspberry Pi	- Tiếp tục thử nghiệm các mô hình và thuật toán trên Raspberry Pi - Tìm hiểu về phần cứng Sipeed MaixCAM Development Board NPU RISCV AI - Phát triển thuật toán đếm vật thể động di chuyển 2 hướng ngược nhau	06./10/2024 
5	- Thử nghiệm thuật toán MOG2, thuật	- Tìm hiểu kết nối UART giữa module	10../10/2024

(07/10/2024 – 13/10/2024)	toán Camshift, mô hình pre-trained SSD300_VGG16 cùng thư viện PyTorch trên Raspberry Pi - Làm quen với Sipeed MaixCAM Development Board NPU RISCV AI - Đếm thành công các vật thể động di chuyển 2 hướng với MaixCAM	GPS NEO M8N với MaixCAM - Tìm hiểu cơ chế truy cập dữ liệu ngoài mạng LAN hỗ trợ kết nối IoT	Lê Anh Vũ
6 (14/10/2024 – 20/10/2024)	- Thực hiện đếm vật thể động di chuyển 2 hướng và lấy dữ liệu GPS (kinh độ, vĩ độ, tốc độ) với MaixCAM - Đóng gói tất cả dữ theo định dạng JSON để gửi ra ngoài mạng LAN thông qua giao thức MQTT đến địa chỉ IP công cộng của AWS	- Thiết kế mạch in PCB kết nối các module với nhau - Bố trí vị trí lắp đặt trên xe buýt để chạy thử nghiệm hệ thống nhằm thu thập thêm hình ảnh về khách lên xuống xe, quan sát sự ổn định và nhiệt độ hoạt động của hệ thống	18/10/2024 Lê Anh Vũ
7	- Lắp ráp hệ thống gồm hộp nhựa, pin dự	- Tiếp tục thu thập dữ liệu hình ảnh và đánh	26/10/2024

(21/10/2024 – 27/10/2024)	phòng, mạch in PCB, module GPS và anten, đèn hồng ngoại, quạt tản nhiệt DC, Bộ Phát Wi-Fi Di Động 4G - Theo dõi hiệu suất và nhiệt độ hoạt động của hệ thống - Theo dõi thời lượng pin dự phòng, Bộ Phát Wi-Fi Di Động 4G	giá hiệu suất hệ thống khi - Tìm hiểu tích hợp phần cứng giải thuật Backend	
8 (28/10/2024 – 03/11/2024)	Đánh giá khối lượng hoàn thành..50..% X được tiếp tục/không tiếp tục thực hiện ĐATN		
9 (04/11/2024 – 10/11/2024)	- Lập trình Backend nhận dữ liệu từ hệ thống và lưu trữ dưới định dạng JSON trong máy tính	- Tìm hiểu liên kết dữ liệu GPS gắn với API của bản đồ	05./11/2024 
10 (11/11/2024 – 17/11/2024)	- Hoàn thành Backend nhận và lưu trữ dữ liệu: số lượng khách lên và xuống, thông tin dung lượng thẻ nhớ, nhiệt độ, thông tin từ GPS	- Tiếp tục lập trình Backend lưu trữ dữ liệu vào cơ sở dữ liệu SQLite - Tìm hiểu cách chuyển đổi mô hình hoạt động trên MaixCAM - Thử nghiệm phần Backend	16/11/2024 

11 (18/11/2024 – 24/11/2024)	- Chuyển đổi thành công mô hình với dữ liệu tùy chỉnh hoạt động trên MaixCAM - Xác định vị trí xe buýt với API bản đồ	- Thiết kế giao diện Frontend cho website hiển thị dữ liệu - Thử nghiệm hiệu suất của mô hình đã fine-tune	20./11/2024  Lê Anh Vũ
12 (25/11/2024 – 01/12/2024)	- Tích hợp thành công phần cứng giải thuật Backend vào cơ sở dữ liệu và Frontend	- Đánh giá lại hệ thống sau khoảng thời gian chạy thực tế	26./11./2024  Lê Anh Vũ
13 (02/12/2024 – 08/12/2024)	- Tối ưu hóa hệ thống cả phần cứng lẫn phần mềm sau khi phân tích và đánh giá toàn diện hệ thống - Viết báo cáo, slides và poster	- Tiếp tục viết báo cáo, slides và poster	04./12./2024  Lê Anh Vũ
14 (09/12/2024 – 15/12/2024)	- Hoàn thành báo cáo, slides và poster - Đóng gói hệ thống thành phẩm với phần cứng và phần mềm tích hợp, đảm bảo hoạt động hiệu quả	- Sẵn sàng bảo vệ đồ án	12./12./2024  Lê Anh Vũ
15 (16/12/2024 – 25/12/2024	Đã hoàn thành…100..% Đồ án tốt nghiệp X được bảo vệ/không được bảo vệ ĐATN		

INTELSCHOOLBUS - HỆ THỐNG ĐẾM VÀ KIỂM SOÁT HỌC SINH TRÊN XE BUÝT ĐƯA ĐÓN CỦA TRƯỜNG HỌC

TÓM TẮT

Đếm hành khách theo thời gian thực trên xe buýt là nhiệm vụ quan trọng để nâng cao hiệu quả giám sát trong giao thông, giáo dục và phát triển thành phố thông minh, phù hợp với xu hướng và mục tiêu chuyển đổi số. Dự án này giới thiệu một hệ thống nhỏ gọn có camera hỗ trợ AI tích hợp với mô-đun GPS để giám sát và đếm hành khách hoặc học sinh theo thời gian thực trên xe buýt, giải quyết các mối quan tâm về an toàn và vận hành trong giao thông công cộng.

So với các nghiên cứu hiện có sử dụng nhiều phương pháp khác nhau như DeepSORT, Kalman Filters và các biến thể YOLO với phần cứng như Raspberry Pi và Arduino, công trình này đã cải tiến bằng cách tận dụng phần cứng thế hệ tiếp theo (kiến trúc RISC-V) có NPU tích hợp mạnh mẽ với mô hình được tối ưu hóa cho hệ thống nhúng, kết hợp các linh kiện thiết yếu để vận hành hiệu quả và ổn định trong điều kiện thực tế.

Cụ thể, hệ thống này sử dụng mô hình YOLO được chuyển đổi sang định dạng nhẹ hơn (cvimodel), dùng mô hình phát hiện đầu người được đào tạo từ 10000 tấm hình kết hợp với theo dõi ByteTrack. Trong bài báo cáo cũng nêu bật những hạn chế của các sản phẩm hiện có và các tính năng được tối ưu hóa của hệ thống được đề xuất.

Hệ thống cũng đã được đặt hàng và triển khai thử nghiệm trên các xe buýt trong Hệ thống Giáo dục Hồng Ngọc - Ruby (thuộc Hệ thống Giáo dục Trí Đức). Kết quả của đề tài này đã được chấp nhận tại Hội nghị quốc tế lần thứ 13 (năm 2025) về Công nghệ thông tin xanh và nhân văn (ICGHIT) do Công nghệ thông tin xanh và nhân văn (GHIT), Viện Kỹ sư điện tử và thông tin (IEIE) đồng tổ chức. Ngoài ra, dự án này đã lọt vào Vòng Tranh Đấu của Cuộc thi tìm kiếm dự án đổi mới sáng tạo ứng dụng trí tuệ nhân tạo AI tại thành phố Hồ Chí Minh năm 2024 (AI. STAR 2024)", do Sở Khoa học và Công nghệ Thành phố Hồ Chí Minh tổ chức, với SIHUB là đơn vị thực hiện.

INTELSCHOOLBUS - THE SYSTEM FOR COUNTING AND CONTROLLING STUDENTS ON SCHOOL BUSES

ABSTRACT

Real-time passenger counting on buses is a crucial task for enhancing monitoring efficiency in transportation, education, and smart city development, aligning with the trends and goals of digital transformation. This project introduces a compact system featuring an AI-powered camera integrated with a GPS module to monitor and count passengers or students in real time on buses, addressing safety and operational concerns in public transportation.

Compared to existing research utilizing various methods such as DeepSORT, Kalman Filters, and YOLO variants with hardware like Raspberry Pi and Arduino, this work advances the field by leveraging next-generation hardware based on RISC-V architecture. It incorporates a powerful integrated NPU with an optimized model for embedded systems, combined with essential components to ensure effective and stable operation in real-world conditions.

Specifically, the system employs a YOLO model converted to a lighter format (cvimodel), utilizing a head detection model trained on 10,000 images in conjunction with ByteTrack tracking. The report also highlights limitations of existing solutions and the optimized features of the proposed system.

The system has been ordered and deployed on buses within The Hong Ngoc – Ruby School Education System (part of the Tri Duc Education System). The results of this project were accepted at The 13th International Conference on Green and Human Information Technology (ICGHIT) is co-hosted by Green and Human Information Technology (GHIT), The Institute of Electronics and Information Engineers (IEIE). Additionally, this project advanced to the Competition Round of the "AI. STAR 2024" contest for innovative AI application projects in Ho Chi Minh City, organized by the Department of Science and Technology of Ho Chi Minh City, with SIHUB serving as the implementing unit.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CLI	Command Line Interface
CNN	Convolutional Neural Network
COCO	Common Objects in Context
DB	Database
DGPS	Differential Global Positioning System
GMM	Gaussian Mixture Models
GPIO	General-Purpose Input/Output
GPS	Global Positioning System
GRU	Gated Recurrent Unit
HDOP	Horizontal Dilution of Precision
I2C	Inter-Integrated Circuit
IoT	Internet of Things
IPEX	IPEX MHF connector
JSON	JavaScript Object Notation
KDE	Kernel Density Estimation
LAN	Local Area Network
LED	Light Emitting Diode
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MIPI CSI	Mobile Industry Processor Interface Camera Serial Interface
MOG2	Mixture of Gaussian 2
MQTT	Message Queuing Telemetry Transport
MSE	Mean Squared Error
NLP	Natural Language Processing

NMEA	National Marine Electronics Association
NPU	Neural Processing Unit
ONNX	Open Neural Network Exchange
ORB	Oriented FAST and Rotated BRIEF
PCA	Principal Component Analysis
RISC-V	Reduced Instruction Set Computer - Five
RNN	Recurrent Neural Network
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SMA	SubMiniature version A
SPI	Serial Peripheral Interface
SURF	Speeded-Up Robust Features
TCP/IP	Transmitter Control Protocol/Internet Protocol
TOPS	Tera Operations Per Second
UART	Universal Asynchronous Receiver-Transmitter
UTC	Coordinated Universal Time
YOLO	You Only Look Once

CHAPTER 1. INTRODUCTION

1.1 Reason for choosing topic

The ensuring student safety during school bus transportation has become a critical issue. Since 2020, numerous incidents involving students being left on school buses have exposed weaknesses in traditional monitoring practices. With approximately 18.5 million students enrolled in primary, secondary, and high schools during the 2023–2024 academic year, the need for advanced technological solutions is pressing to meet the needs of society as well as comply with Decree No. 86/2014/ND-CP, the 2008 Road Traffic Law, Circular No. 12/2017/TT-BGTVT, Resolution No. 88/NQ-CP, and Directive No. 31/CT-TTg.

This project introduces a compact system featuring an AI-enabled camera integrated with a GPS module to monitor and count passengers or students in real time on buses, addressing safety and operational concerns in public transportation.

Traditional methods, such as manual counts or basic sensors, are error-prone and lack scalability, especially in complex urban environments. And the existing studies utilizing various methods such as DeepSORT, Kalman Filters, and YOLO variants with hardware like Raspberry Pi and Arduino.

Hence, there is a need for an optimized approach leveraging AI on small-scale, scalable hardware to address the challenges mentioned above in the methods for counting and managing people on vehicles.

In general, this is both an urgent social issue with high humanitarian value and a practical application that can be immediately implemented with two major educational institutions in Ho Chi Minh City (Ruby, Tri Duc) and as well as providing a great idea for the AI application solution competition organized by the Department of Science and Technology of Ho Chi Minh City. In addition, the project also contributes to further research in machine learning, AI, and people-counting methods, supporting the upcoming The 13th International Conference on Green and Human

Information Technology (ICGHIT) in January 2025 at Ton Đuc Thang University, Nha Trang.

1.2 Target implementation

This project aims to develop an integrated system for real-time student counting on school buses. The project begins by researching lightweight recognition, tracking, and counting models that are affordable (under $150) but operate effectively on resource-constrained devices. The system must be able to access data outside the LAN, ensuring seamless IoT connectivity for remote monitoring and data access. Additionally, the project will explore GPS systems and their integration with mapping APIs, enabling tracking the bus's location with an error margin of less than 5 meters and monitoring the number of students in real-time via a web interface with a database for storing information. The system also ensures the integration of power source and embedded hardware with custom dataset trained AI model allowing continuous operation across three school shifts: morning, noon, and afternoon, equating to 12 hours; dimensions not exceeding 200x200x100 mm to easy installation on school buses. Throughout the product development process, the project will analyze and evaluate both hardware and software results in ensure optimal performance and reliability.

Finally, the project will produce report, presentation, and poster. The system's performance requirements are to achieve a frame rate above 10 FPS and an accuracy greater than 95%, with potential for further improvement.

1.3 Practical significance and object of topic

The practical significance of this project lies in its potential to address a critical and pressing issue: ensuring student safety during school bus transportation. The recent incidents of students being left on buses highlight the inadequacy of traditional monitoring methods. This project offers a technologically advanced solution to mitigate these risks by providing real-time, automated monitoring of passenger

counts and vehicle location. The system's development is not just a technological advancement but also a response to societal needs and compliance with existing regulations.

By providing a reliable and scalable system, this project aims to enhance the safety and well-being of the approximately 18.5 million students who rely on school bus transportation. Furthermore, the project's successful deployment within the Hong Ngoc – Ruby School Education System and its recognition in prestigious competitions demonstrate its real-world applicability and potential impact.

The object of this project is to develop a compact, cost-effective for real-time student counting and location monitoring on school buses. This system integrates a custom-trained AI camera, GPS, IoT connectivity, and a web interface, all deployed on affordable embedded hardware. The goal is to provide a practical solution for enhancing student safety and bus management, achieving high accuracy and real-time performance within a cost-effective and easily deployable design.

1.4 Highlights of topic

The project's highlight lies in applying advanced technology and artificial intelligence to the traditional problem of people counting while addressing existing challenges such as product cost, integrated system design, and compatibility with embedded devices. With these breakthroughs, the project demonstrates high practical value for society. Additionally, it contributes to the development of the IoT and AI community by researching cutting-edge technologies such as the new RISC-V CPU architecture, YOLO model, cvimodel model, and an optimized and lightweight web interface for monitoring.

CHAPTER 2. OVERVIEW

2.1 Analyze and evaluate existing products

Table 2.1: Comparison Table of People Counting Products on the Market

Feature	People Counter Using IR Sensor	People Counter Using Thermal Camera	People Counter Using 3D Camera
Technology	Uses infrared sensor	Uses thermal sensor	3D camera with depth sensor
Counting Performance	Performance decreases if many people move simultaneously or overlap	High accuracy in dark environments but struggles with multiple people simultaneously	Accurate but requires more complex processing
Investment Cost	Costs around $260	Average cost starts from $190, depending on sensor quality	High cost (over $630) due to complex technology and specialized installation

Table 2.1 presents the specifications and pricing to highlight the limitations of existing market solutions. The IR sensor struggles with overlapping movements, reducing accuracy in crowds. Thermal cameras work well in low light but face issues with multiple people. 3D cameras offer high accuracy but are costly, complex, and require specialized installation. So based on this table, this work will develop a system that meets the initial criteria and addresses the current issues.

2.2 Problems to be solved

This project addresses the limitations of current people counting technologies for school bus monitoring. Existing IR sensors struggle with overlapping movements, thermal cameras have difficulty with multiple people, and 3D cameras are too costly and complex as Figure 2.1. This project aims to develop a cost-effective, accurate, and scalable solution that overcomes these limitations, providing real-time, reliable counting and monitoring for enhanced student safety.

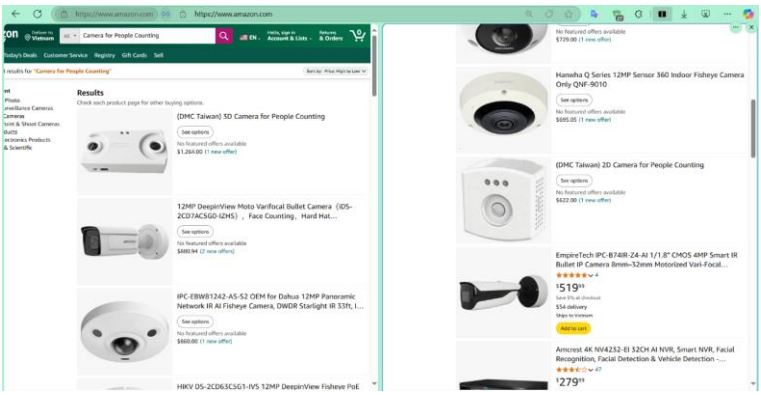


Figure 2.1: Prices of people-counting cameras on Amazon website

According to the outlined objectives, this project will design a product that meets the performance and hardware requirements listed in Table 2.2.

Table 2.2: Performance and hardware requirements

Property	Required Specification
Accuracy	Over 95%
FPS (Frames per Second)	Over 10
Cost	Below $150
Weight	Below 200 grams
Operating Temperature	Below 80°C
Operating Time	Minimum 12 hours continuously
System Size	Smaller than 200x200x100 mm
Other Standards	Industrial-grade wired connection, dustproof, and water-resistant IPX22

CHAPTER 3. THEORETICAL BASIS

3.1 Definition of Machine Learning and Computer Vision

3.1.1 Definition of Machine Learning

Machine learning is a branch of artificial intelligence focused on developing methods and techniques that enable computers to learn from data without being explicitly programmed. Instead of being instructed on specific tasks, computers learn from data to improve their performance over time.

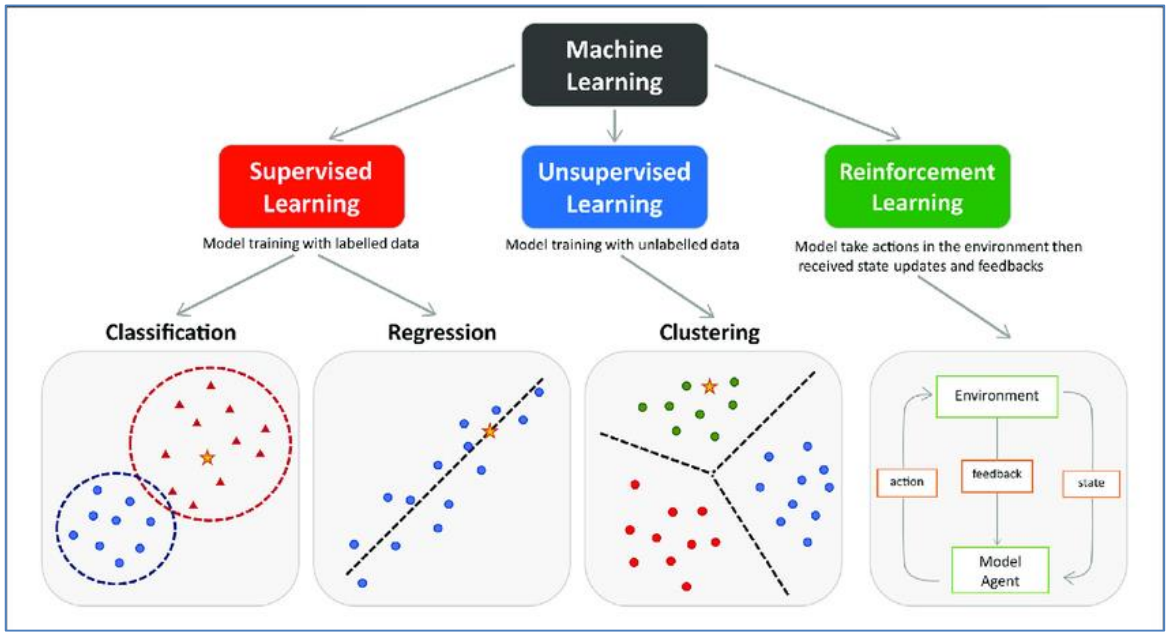


Figure 3.1: Machine Learning Classification

(Source: University College London)

As shown in Figure 3.1, machine learning methods include supervised learning, unsupervised learning, and reinforcement learning, each encompassing various specific techniques and algorithms. Machine learning can be classified into the following categories:

- Supervised Learning: This type of learning involves a labeled dataset, where each data point is paired with the desired output. Common types of supervised learning include:

  - Classification: Categorizing an image or a region within an image into predefined classes.
  - Object Detection: Using to locate and classify objects within an image.
  - Segmentation: Using to identify and segment objects at the pixel level.
- Unsupervised Learning: This approach involves using an unlabeled dataset, where the model learns to identify patterns and structures within the data. Key methods of unsupervised learning include:
  - Clustering: Dividing data into groups (clusters) where data points within the same group are similar, while those in different groups are distinct. Common algorithms include K-means clustering and hierarchical clustering.
  - Dimensionality Reduction: Reducing the number of input variables while retaining essential information. This simplifies data complexity and uncovers hidden structures. Methods include Principal Component Analysis (PCA) and t-SNE (t-distributed Stochastic Neighbor Embedding).
  - Density Estimation: Determining the probability density function of data to understand its distribution and key features. Techniques include Gaussian Mixture Models (GMM) and Kernel Density Estimation (KDE).
- Reinforcement Learning: In this approach, an agent learns to achieve a goal within an environment by interacting with it. The agent performs actions and receives feedback in the form of rewards or penalties. The objective is to develop an optimal policy that maximizes long-term rewards.

3.1.2 Definition of Computer Vision

Computer vision is a subfield of artificial intelligence that focuses on developing methods and technologies enabling computers to understand and analyze images and videos automatically. It involves tasks like object detection, image classification, and motion tracking.

Currently, OpenCV is recognized as a powerful and widely used open-source library specifically designed for computer vision tasks. Essentially, it is a toolkit containing numerous functions and algorithms that enable developers to process, analyze, and interpret images and videos. In this project, I also use this library to work with embedded devices for input image processing, applying supervised machine learning with a fine-tuned YOLO model based on a collected head detection dataset.

These are some techniques that applied to the input images before they are fed into the neural network. They aim to standardize the data, improve its quality, and enhance the model's learning process:

- Auto-Orient: Auto-orientation is a preprocessing step that automatically adjusts the orientation of an image to a standard orientation (typically upright). This is achieved by analyzing the image's metadata (e.g., EXIF data) or by detecting features that indicate the image's orientation. Purpos to ensure that all images are in a consistent orientation, regardless of how they were captured. This prevents the model from learning orientation-specific features and improves its generalization ability. Implementation: Algorithms for auto-orientation typically involve analyzing the image's metadata, detecting dominant lines or edges, or using machine learning models to predict the correct orientation.
- Resize: To ensure that all input images have a consistent size, which is often required by neural networks. Stretching involves non-uniform scaling, which can introduce distortions but is sometimes preferred to maintain the original aspect ratio. Implementation: Resizing algorithms typically involve interpolation techniques (e.g., bilinear, bicubic) to calculate the pixel values of the resized image. Stretching does not preserve the aspect ratio and can distort the image.
    - Grayscale: Conversion transforms a color image into a grayscale image, where each pixel represents a single intensity value rather than three color channels (red, green, blue). Purpose to reduce the dimensionality of the

input data, which can simplify the model and reduce computational cost. It can also make the model more robust to variations in color. Implementation: Grayscale conversion is typically done by averaging the values of the red, green, and blue channels or by using a weighted average based on the perceived brightness of each color. A common formula is as Equation (3.1):

$$Gray = 0.299*Red + 0.587*Green + 0.114*Blue \quad (3.1)$$

- Auto-Adjust Contrast: Using Contrast Stretching: A technique that enhances the contrast of an image by expanding the range of intensity values. It maps the original intensity values to a new range, typically from 0 to 255 (or the maximum value for the image's bit depth). Purpose to improve the visibility of details in the image, especially in low-contrast or poorly illuminated images. It can also help the model learn more robust features. Implementation: Contrast stretching typically involves finding the minimum and maximum intensity values in the image and then mapping the original values to the new range using a linear or non-linear transformation.

Additionally, augmentation methods are applied to training images to create variations of the original data. These techniques aim to enhance the diversity of the training dataset, improve the model's robustness, and prevent overfitting:

- Outputs per training example: This parameter specifies the number of augmented images generated from each original training image. Purpose: To increase the size of the training dataset, which can improve the model's generalization ability.
- Flip: Horizontal, Vertical. Image flipping involves mirroring an image along its horizontal or vertical axis. Purpose: To create variations of the original images that are still realistic. It can help the model learn features that are invariant to flipping. Implementation: Flipping is typically done by reversing the order of pixels along the specified axis.

- 90° Rotate: Clockwise, Counter-Clockwise, Upside Down. Image rotation involves rotating an image by a specified angle. In this case, the images are rotated by 90 degrees clockwise, counter-clockwise, or 180 degrees (upside down). Purpose: To create variations of the original images that are still realistic. It can help the model learn features that are invariant to rotation. Implementation: Rotation is typically done by transforming the coordinates of the pixels in the image.
- Shear: Image shearing involves distorting an image by shifting pixels along one axis based on their position along the other axis. Purpose: To create variations of the original images that are still realistic. It can help the model learn features that are invariant to shearing. Implementation: Shearing is typically done by transforming the coordinates of the pixels in the image using a shear matrix.
- Grayscale: This augmentation applies grayscale conversion to a specified percentage of the training images. Purpose: To introduce variations in color and improve the model's robustness to grayscale images. Implementation: The same grayscale conversion method as described in preprocessing is used.
- Hue: Adjustment involves changing the color of an image by shifting the hue values in the color space. Purpose: To create variations in color and improve the model's robustness to changes in lighting and color conditions. Implementation: Hue adjustment is typically done by converting the image to a color space like HSV (Hue, Saturation, Value) and then shifting the hue channel.
- Saturation: Adjustment involves changing the intensity of the colors in an image. Purpose: To create variations in color and improve the model's robustness to changes in lighting and color conditions. Implementation: Saturation adjustment is typically done by converting the image to a color space like HSV and then scaling the saturation channel.

- Brightness: Adjustment involves changing the overall lightness or darkness of an image. Purpose: To create variations in lighting and improve the model's robustness to changes in illumination. Implementation: Brightness adjustment is typically done by adding or subtracting a constant value from the pixel intensities.
- Exposure: Adjustment involves changing the overall brightness of an image, similar to brightness adjustment, but often with a more nuanced effect. Purpose: To create variations in lighting and improve the model's robustness to changes in illumination. Implementation: Exposure adjustment is typically done by applying a non-linear transformation to the pixel intensities.
- Blur: Image blurring involves reducing the sharpness of an image by averaging the pixel values in a local neighborhood. Purpose: To create variations in image sharpness and improve the model's robustness to blurry images. Implementation: Blurring is typically done using a Gaussian filter or a similar convolution kernel.
- Noise: Addition involves adding random values to the pixel intensities of an image. Purpose: To create variations in image quality and improve the model's robustness to noisy images. Implementation: Noise addition is typically done by adding random values from a Gaussian or uniform distribution to the pixel intensities.

So these preprocessing and augmentation techniques are essential for training robust and accurate deep learning models for image-based tasks. They help to standardize the input data, increase its diversity, and improve the model's generalization ability. The specific parameters and methods used may vary depending on the task and the dataset.

3.2 YOLO model

3.2.1 Definition and structure

YOLO is an algorithm in the field of machine learning, specifically deep learning, used for object detection tasks. It was first introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi.

YOLO employs a convolutional neural network, as illustrated in Figure 3.2, to process images. It divides the image into a grid, makes multiple predictions for each grid cell, filters out low-confidence predictions, and removes overlapping bounding boxes to produce the final output.

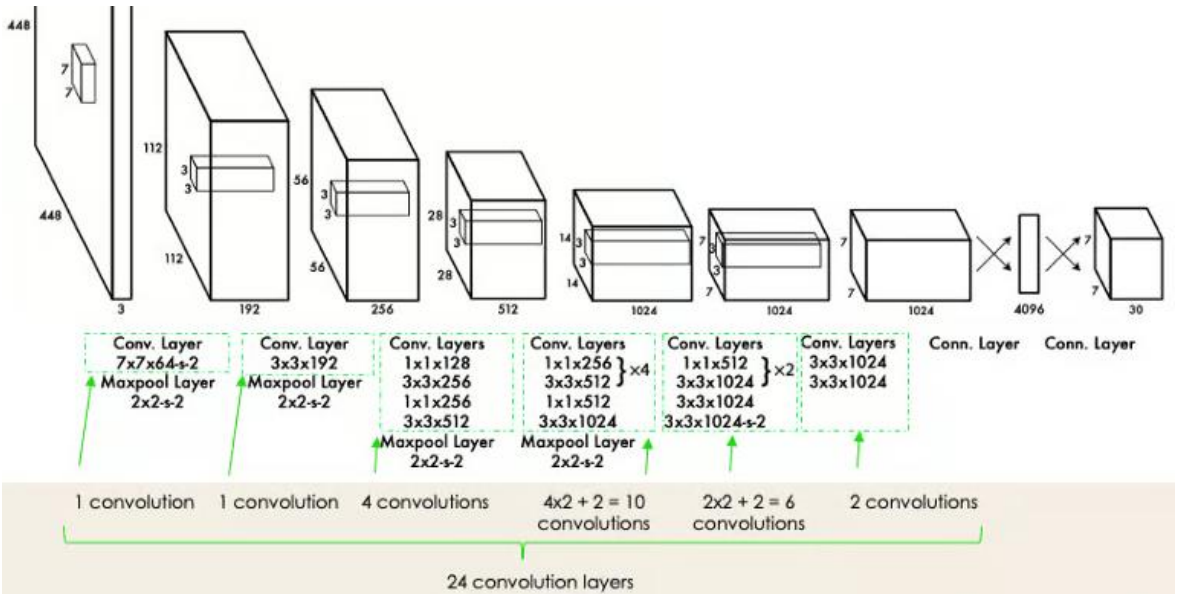


Figure 3.2: Architecture of YOLO

(Source: DataCamp)

YOLO is a single-stage detector that processes both object localization and classification in a single network run. While YOLO is not the only single-stage detection model (for example, MobileNetSSDv2 is another popular single-shot detector), it generally outperforms others in terms of speed and accuracy.

By treating the detection task as a one-step regression method to determine bounding boxes, YOLO models are typically very fast and compact. This makes them quicker to train and easier to deploy, especially on edge devices with limited

computational power. Table 3.1 below highlights the clear advantages of YOLO due to its CNN-based architecture, which is highly suitable for object detection tasks.

Table 3.1: Comparison CNN vs RNN

Feature	Convolutional Neural Networks (CNNs)	Recurrent Neural Networks (RNNs)
Primary Use Case	Spatial data processing (images, videos), feature extraction	Sequential data processing (text, speech, time series)
Data Structure	Grid-like data (pixels in images, voxels in 3D data)	Sequential data (ordered sequences of data points)
Key Operation	Convolution: Applying filters to extract local features	Recurrence: Processing data points in a sequence, maintaining a hidden state
Processing Direction	Typically processes the entire input at once (parallel processing)	Processes data sequentially, one element at a time
Memory	No inherent memory of past inputs	Maintains a hidden state that acts as a memory of past inputs
Parameter Sharing	Parameters (filter weights) are shared across spatial locations	Parameters are shared across time steps
Input Size	Fixed input size (typically requires resizing or padding)	Variable input size (can handle sequences of different lengths)
Output Size	Can produce fixed-size outputs (classification) or	Can produce fixed-size outputs (sentiment analysis) or variable-size

	variable-size outputs (segmentation)	outputs (machine translation)
Handling of Spatial Relationships	Excellent at capturing local spatial relationships (e.g., edges, textures)	Limited in capturing spatial relationships in grid-like data
Handling of Temporal Relationships	Limited in capturing temporal relationships in sequential data	Excellent at capturing temporal relationships in sequential data
Computational Complexity	Generally efficient for parallel processing, can be computationally intensive for very deep networks	Can be computationally intensive, especially for long sequences due to sequential processing
Common Applications	Image classification, object detection, image segmentation, image generation, video analysis	Natural language processing (NLP), speech recognition, time series analysis, machine translation, text generation
Strengths	Effective for spatial feature extraction, robust to small variations in input, efficient parallel processing	Effective for sequential data processing, can model long-range dependencies, handles variable-length sequences
Weaknesses	Limited in handling sequential data, not inherently designed for temporal relationships	Can be computationally expensive, can struggle with very long sequences, limited in capturing spatial relationships

Typical Layers	Convolutional layers, pooling layers, fully connected layers	Recurrent layers (RNN, LSTM, GRU), fully connected layers

3.2.2 YOLO versions

Currently, there are 11 versions of YOLO, including both official and experimental releases. Figure 3.3 illustrates the speed differences among models. However, within the scope of this project, this project only use three versions: YOLOv5, YOLOv8, and YOLO11—for testing and deployment. Some notable new features of these three YOLO versions are as follows:

- YOLOv5 only requires the installation of torch and some lightweight python libraries. The YOLOv5 models train extremely quickly which helps cut down on experimentation costs as build the model. YOLOv8 was developed by Ultralytics, also created the influential and industry-defining YOLOv5 model.
- YOLOv8 includes numerous architectural and developer experience changes and improvements over YOLOv5. YOLOv8 comes with a lot of developer-convenience features, from an easy-to-use CLI to a well-structured Python package. YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box.
- YOLO11 is the latest iteration in the Ultralytics YOLO series of real-time object detectors, redefining what's possible with cutting-edge accuracy, speed, and efficiency. Building upon the impressive advancements of previous YOLO versions, YOLO11 introduces significant improvements in architecture and training methods, making it a versatile choice for a wide range of computer vision tasks.

Each YOLO version includes suffix letters such as "n," "s," "m," "l," and "x" to denote different sizes of the same base model architecture. They represent a scaling of the model's parameters, such as the number of channels, layers, and filters.

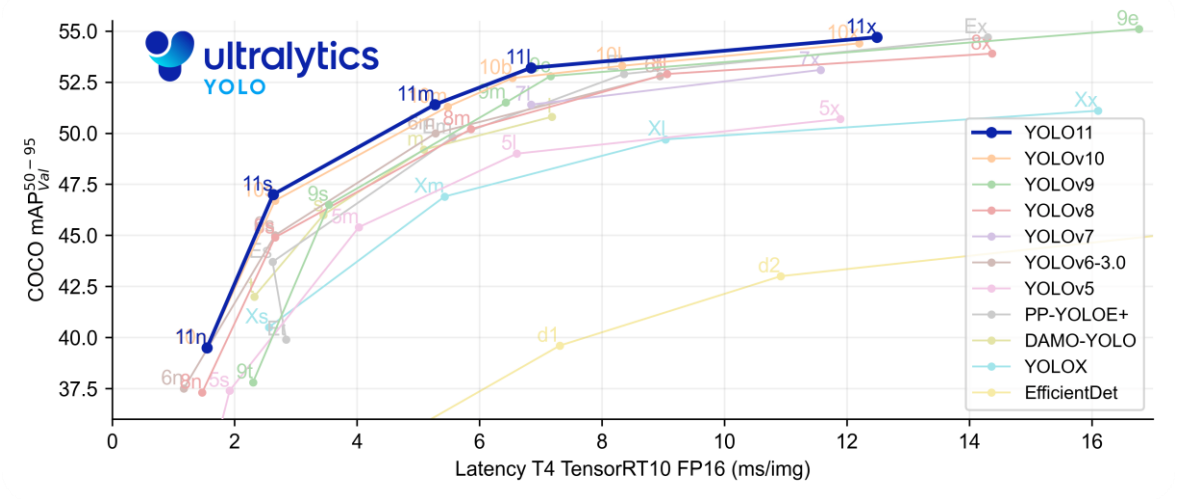


Figure 3.3: Latency of YOLO versions (ms/image)

(Source: Ultralytics)

3.2.3 Fine-Tune a YOLO model

Fine-tuning, in the context of deep learning, refers to the process of taking a pre-trained model (a model that has already been trained on a large dataset) and further training it on a new, often smaller, dataset that is specific to a particular task or domain. Instead of training a model from scratch, fine-tuning leverages the knowledge already learned by the pre-trained model, allowing it to adapt to the new task more quickly and effectively. Some important terms in AI model training:

- Epochs:

Definition: An epoch represents one complete pass of the entire training dataset through the neural network during the training process. Analogy: Imagine you're studying for an exam. An epoch is like going through all the chapters in your textbook once. Purpose: Iterative Learning: Neural networks learn iteratively. Each epoch allows the model to adjust its internal parameters (weights and biases) based on the errors it makes during that pass. Convergence: By repeating the process over multiple epochs, the model gradually improves its ability to make accurate predictions, eventually converging towards a solution. Fine-Tuning: Epochs are used to fine-tune the model's parameters, allowing it to learn complex patterns in the data. Number of Epochs: The number of epochs required for training depends on the complexity of

the task, the size of the dataset, and the architecture of the neural network. Too few epochs may result in underfitting (the model doesn't learn enough), while too many epochs may lead to overfitting (the model learns the training data too well and performs poorly on unseen data). Monitoring: During training, it's crucial to monitor the model's performance (e.g., loss and accuracy) across epochs to determine the optimal number of epochs.

- Imgsz (Image Size):

Definition: Imgsz refers to the size (dimensions) of the input images that are fed into the neural network during training. It's typically expressed as a single number representing the width and height of the square image (e.g., imgsz=640 means the image is 640 pixels wide and 640 pixels high). Purpose: Input Compatibility: Neural networks, especially convolutional neural networks (CNNs), often require input images to have a specific size. Computational Efficiency: Resizing images to a consistent size helps with efficient processing and batching during training. Feature Extraction: The input size can affect the features that the model learns. Smaller images may result in the loss of fine-grained details, while larger images may require more computational resources. Image Resizing: Before being fed into the network, input images are typically resized to the specified imgsz. This resizing may involve interpolation or other techniques to maintain image quality. Trade-Offs: Choosing the right imgsz involves a trade-off between computational cost and the level of detail captured in the images.

- Learning Rate:

It adjusts the update level of parameters at each training step and a learning rate that is too high may skip the optimal solution, while one that is too low slows down the training process.

- Loss Function:

This function measures the difference between the model's predictions and actual labels. Common functions: MSE (Mean Squared Error), Cross-Entropy Loss.

- Optimizer:

Using for updates the model's parameters based on the loss function and gradients.

- Overfitting:

Occurs when the model learns too much detail from the training data, reducing performance on new data.

- Underfitting:

Occurs when the model doesn't learn enough from the data, resulting in poor performance on both training and new data.

- Data Augmentation:

A method of enriching data by generating variations such as flipping, rotating, or cropping images.

3.2.4 Evaluation metrics

Figure 3.4 shows the formula for calculating IoU between two bounding boxes. The area of Overlap is the area of the common portion between the two bounding boxes. Area of Union is the combined area of the two bounding boxes, computed by adding the areas of each bounding box and then subtracting the intersection area. IoU ranges from 0 to 1. At IoU=0, there's no overlap between bounding boxes, and at IoU=1, they fully overlap.

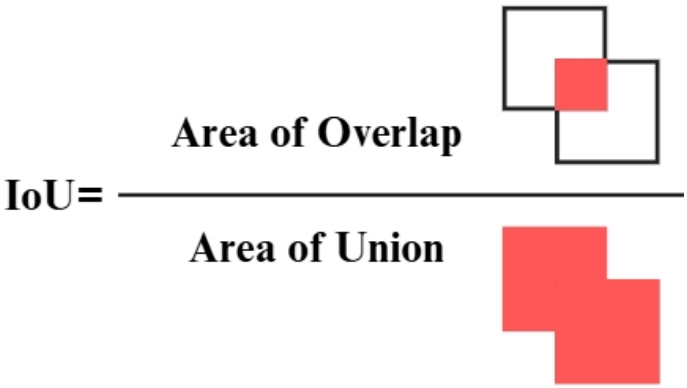


Figure 3.4: IoU formula

Accuracy (A) in AI detection is a metric that measures the percentage of correct predictions out of the total predictions made by the model. It is one of the most commonly used metrics for evaluating the performance of machine learning models,

particularly in classification tasks. The formula for calculating accuracy is expressed by Equation (3.2):

$$Accuracy\ (A) = \frac{(tp + tn)}{(tp + fp + tn + fn)} \tag{3.2}$$

The F1-score, Equation (3.2), is a crucial metric for evaluating the performance of machine learning models, particularly in classification tasks. It is defined as the harmonic mean of precision and recall like Equation (3.3), providing a balanced measure that reflects the model's predictive capability more accurately by considering both precision and recall equally.

$$F_{measure}\ (F1) = \frac{(2 \times precision \times recall)}{(precision + recall)} \tag{3.3}$$

Precision: measures how well a model can identify the correct labels or classes for the data. precision was calculated by dividing the number of true positives (tp) by the total number of true positives and false positives (fp). The formula is as Equation (3.4). Where tp is the number of predictions that are correct and actually correct, and fp is the number of predictions that are correct but are actually wrong. In terms of object detection, determine the true positive and false positive based on the IoU among two bounding boxes. The predicted bounding box which has a high IoU value with its correspondence ground truth bounding box will be considered as tp and the other with a low IoU value is fp.

$$Precision\ (P) = \frac{tp}{(tp + fp)} \tag{3.4}$$

Recall: measures how well a model can correctly identify the relevant data points or the positive class out of all the data points that belong to that class. It is calculated by dividing the number of tp by the sum of true positives and false negatives (fn). The formula is as Equation (3.5). where fn is the number of false predictions that are actually correct. In terms of object detection, fn is the number of objects that the model failed to detect correctly. They are the objects that belong to the positive class, but the model cannot recognize them. To calculate fn in object detection, need to

compare the ground truth bounding boxes with the modelpredictions and count how many ground truths are not matched with any model predictions.

$$Recall\ (R) = \frac{tp}{tp + fn} \tag{3.5}$$

F1-Confidence Curve: This curve shows how the F1 score changes as the confidence threshold is varied. Confusion Matrix is a standard confusion matrix, showing the counts of true positives, false positives, and false negatives for each class. Confusion Matrix Normalized is a normalized confusion matrix, showing the proportions of true positives, false positives, and false negatives for each class. Training and Validation Loss and Metrics show the training and validation loss and metrics over the training epochs.

FPS: indicates the number of images the model can process in one second. This index is very important when evaluating the real-time processing ability of the model. High FPS shows that the model has fast processing capabilities, suitable for applications that need immediate response.

3.3 Tracking algorithms

Some of the methods for tracking moving objects that were researched and tested in this study include MOG2, CamShift, ByteTrack, DeepSORT, and DeepSORT Real-Time.

3.3.1 MOG2

MOG2 is a method that relies on the combination of frames rather than a single frame. It uses a mixture of Gaussian distributions to store the longevity of pixels in the frame, which helps it adapt better to changes in lighting.

3.3.2 CamShift (Continuously Adaptive Mean Shift)

CamShift is a variant of the Mean Shift method, which adjusts the size and angle of the tracking window based on the object's color distribution. It uses a regression

image of the frame to find the new position of the object and adjust the tracking window's size.

3.3.3 ByteTrack

ByteTrack is a multi-object tracking method that connects detections from each frame, rather than only high-confidence detections. This helps to detect and track low-confidence objects, minimizing the loss of true objects.

3.3.4 DeepSORT

DeepSORT is a multi-object tracking method that uses neural networks to improve object discrimination. It combines shape and color features to track objects across frames.

3.3.5 DeepSORT Real-Time

DeepSORT Real-Time is a faster version of DeepSORT, designed to operate efficiently in real-time. It provides high-speed, accurate multi-object tracking.

3.4 ONNX model

The Open Neural Network Exchange (ONNX) is a collaborative open-source initiative, which was established under the auspices of the Linux Foundation, aimed at promoting the interoperability of artificial intelligence (AI) models. ONNX serves as a standardized format for representing deep learning and machine learning models, thereby facilitating seamless integration and deployment across different AI frameworks and tools. Some key attributes of ONNX can be told:

- Interoperability: ONNX defines a common set of operators and a file format that allows developers to utilize models across various AI frameworks, such as PyTorch, TensorFlow, and others, without encountering compatibility issues. This enables a flexible workflow from model development to deployment.

- Optimization: ONNX-compatible runtimes and libraries are designed to maximize performance across diverse hardware platforms, including CPUs, GPUs, and specialized accelerators, thus ensuring efficient utilization of computational resources.
- Community-Driven Development: As a community-led project, ONNX encourages contributions from developers worldwide. Its open governance structure ensures transparency and fosters collaboration, innovation, and continuous improvement.
- Model Conversion: ONNX supports converting models from different AI frameworks into the ONNX format. This capability simplifies the process of model sharing and deployment, making it easier to integrate AI models into production environments.
- Extensibility: The ONNX format is designed to be extensible, allowing for the incorporation of new operators and functionalities as the field of AI evolves. This ensures that ONNX remains relevant and adaptable to emerging AI techniques and technologies.

One of the most significant benefits of using ONNX is framework agnostic, that means ONNX allows developers to choose their preferred framework for model development while maintaining the flexibility to deploy across various platforms. By enabling optimized execution on a variety of hardware platforms, ONNX helps achieve better performance and efficiency in AI workloads. Its compatibility with hardware accelerators such as GPUs, TPUs, and NPUs ensures faster inference and lower latency. Furthermore, tools like ONNX Runtime optimize models through graph-level transformations, quantization, and other performance enhancements, making ONNX a powerful choice for deploying AI models in production.

3.5 UART protocol

UART likes a dedicated messenger service for sending data between devices, using a simple and flexible method. At its core, UART is a hardware communication

protocol that enables asynchronous serial communication. "Asynchronous" means that the sender and receiver don't need to be perfectly synchronized by a shared clock signal. Instead, it rely on start and stop bits to frame each piece of data it send. "Serial" means that data is transmitted one bit at a time, sequentially, over a single wire (or two wires, for full-duplex communication).

In UART, each piece of data (usually a byte) is wrapped with a start bit at the beginning and one or more stop bits at the end. These bits act like the envelope, telling the receiver when a new piece of data is arriving and when it's finished.

The sender and receiver agree on a communication speed, called the baud rate, which determines how fast the bits are transmitted. This is like agreeing on how fast the post office will deliver the mail. The baud rate must be the same on both sides for successful communication. Data transmitted over UART (Figure 3.5) is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART configuration), an optional parity bit, and 1 or 2 stop bits.

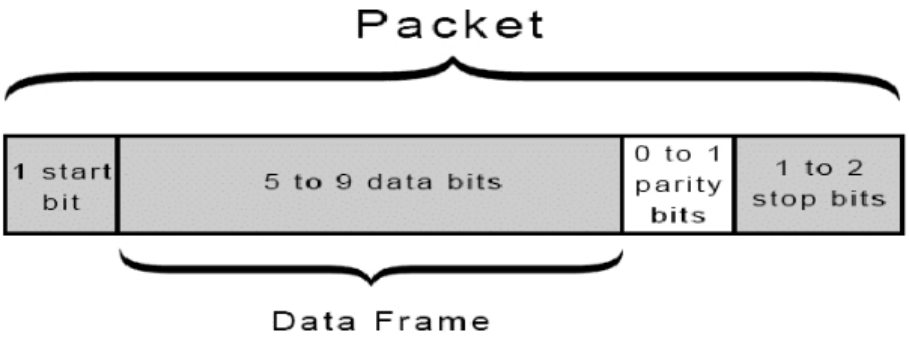


Figure 3.5: UART's data transmitted architecture

(Source: VINAFE CO.,LTD)

UART typically uses two wires for communication: one for transmitting data (TX) and one for receiving data (RX). In some cases, only one wire is used for half-duplex communication, where devices take turns sending and receiving.

3.6 MQTT protocol

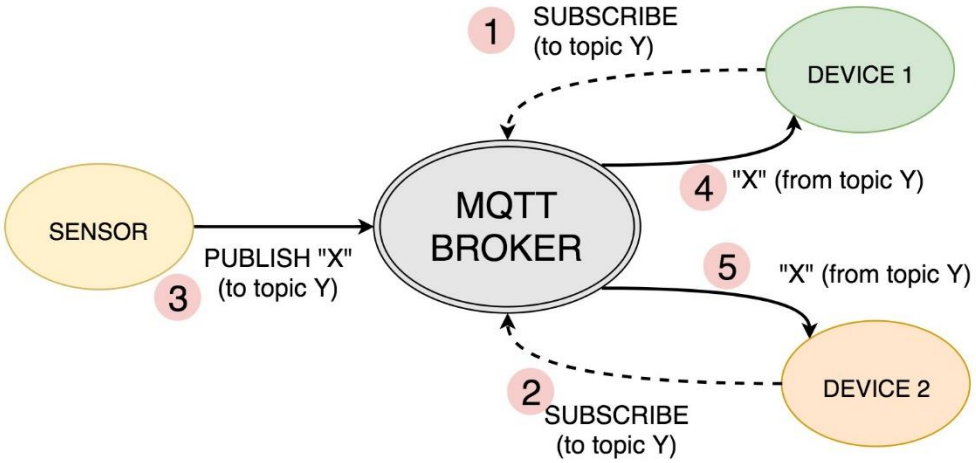


Figure 3.6: MQTT's operation

(Source: Norwegian Creations AS)

MQTT is a lightweight, publish-subscribe network protocol designed for machine-to-machine (M2M) communication and the Internet of Things. It operates on top of the TCP/IP protocol and is particularly well-suited for resource-constrained devices and unreliable networks. Unlike traditional client-server models, MQTT employs a broker-centric architecture, enabling efficient and scalable message distribution. Its core principles are:

- Publish-Subscribe Model: MQTT utilizes a publish-subscribe messaging pattern likes Figure 3.6. Devices that wish to send data "publish" messages to a specific topic, while devices that want to receive data "subscribe" to those topics. The MQTT broker acts as an intermediary, routing messages from publishers to subscribers.
- Broker as Central Hub: The MQTT broker is a central server that manages all message traffic. It receives messages from publishers, filters them based on topics, and forwards them to the appropriate subscribers.
- Topic-Based Routing: Messages are categorized and routed based on topics, which are hierarchical strings that define the message's content or purpose. This allows for flexible and granular message filtering.

- Lightweight and Efficient: MQTT is designed to be lightweight and efficient, minimizing bandwidth usage and processing overhead. This makes it suitable for devices with limited resources and networks with high latency or packet loss.

MQTT does not dictate the specific data format within a message payload. It is a transport protocol, and the payload can contain any type of data (text, JSON, binary). However, the protocol itself does define the structure of the MQTT packets that encapsulate the payload. An MQTT packet consists of three main parts:

- Fixed Header: This header is present in every MQTT packet and contains essential information about the packet type and flags. It is typically 2-5 bytes in length:
  - Control Packet Type (4 bits): Specifies the type of MQTT message (CONNECT, PUBLISH, SUBSCRIBE).
  - Flags (4 bits): Provides additional information about the message, such as QoS level, retain flag, and duplicate flag.
  - Remaining Length (Variable Byte): Indicates the length of the remaining part of the packet.
- Variable Header: This header is optional and its content depends on the packet type. It may contain information such as:
  - Protocol Name and Version: Used in the CONNECT packet.
  - Topic Name: Used in the PUBLISH and SUBSCRIBE packets.
  - Message Identifier: Used for QoS levels 1 and 2.
  - Client Identifier: Used in the CONNECT packet.
- Payload: This is the actual data being transmitted. It can be of variable length and contains the application-specific data.

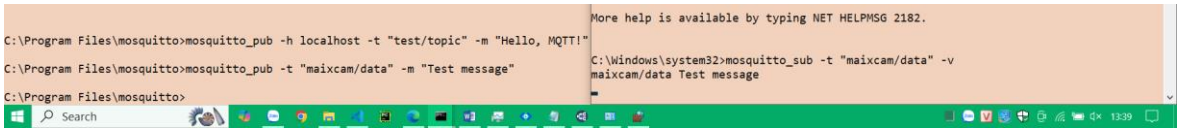


Figure 3.7: MQTT simulation

And MQTT operates on top of TCP/IP, which is a byte-oriented protocol. The MQTT packets are transmitted as a stream of bytes. Each byte consists of 8 bits. The number of bits transmitted depends on the size of the MQTT packet, which varies based on the packet type, variable header, and payload size. Figure 3.7 shows a simulation of MQTT protocol in LAN network, get the laptop's IP as a MQTT broker.

3.7 GPS signal decoding

GPS modules typically output data in the form of NMEA (National Marine Electronics Association) sentences. These sentences are ASCII text strings that contain various pieces of information about the GPS position, time, and other parameters. Two commonly used sentences are $GNGGA and $GNVTG, which provide essential data for location and speed. Decoding $GNGGA and $GNVTG NMEA sentences involves parsing the comma-separated fields, extracting the relevant data (latitude, longitude, and speed), converting the data to appropriate units, and verifying the checksum.

3.7.1 $GNGGA Sentence: Global Positioning System Fix Data

The $GNGGA sentence provides the core information about the GPS fix, including latitude, longitude, and fix quality. The general format of the $GNGGA sentence is: $GNGGA,hhmmss.sss,llll.llll,a,yyyyy.yyyy,b,x,xx,x.x,x.x,M,x.x,M,x.x, xxxx*hh<CR><LF>.

Where: $GNGGA is sentence identifier. hhmmss.sss: UTC time of the fix (hours, minutes, seconds, milliseconds). llll.llll: Latitude in degrees and decimal minutes. a: Latitude direction (N for North, S for South). yyyyy.yyyy: Longitude in degrees and decimal minutes. b: Longitude direction (E for East, W for West). x: GPS fix quality indicator (0 = invalid, 1 = GPS fix, 2 = DGPS fix, etc.). xx: Number of satellites being tracked. x.x: Horizontal dilution of precision (HDOP). x.x: Altitude above mean sea level. M: Units of altitude (meters). x.x: Height of geoid above WGS84 ellipsoid. M: Units of geoid height (meters). x.x: Time since last DGPS update. xxxx: DGPS

reference station ID. *hh: Checksum (hexadecimal). <CR><LF>: Carriage return and line feed characters.

3.7.2 $GNVTG Sentence: Course Over Ground and Ground Speed

The $GNVTG sentence provides information about the course over ground and ground speed. The general format of the $GNVTG sentence is: $GNVTG,x.x,T,x.x,M,x.x,N,x.x,K,a*hh<CR><LF>.

Where: $GNVTG is sentence identifier. x.x: Course over ground in degrees (true). T: Indicates that the course is relative to true north. x.x: Course over ground in degrees (magnetic). M: Indicates that the course is relative to magnetic north. x.x: Ground speed in knots. N: Indicates that the speed is in knots. x.x: Ground speed in kilometers per hour. K: Indicates that the speed is in kilometers per hour. a: Mode indicator (A = Autonomous, D = Differential, E = Estimated, N = Not valid). *hh: Checksum (hexadecimal). <CR><LF>: Carriage return and line feed characters.

3.8 Flask web framework

Flask is a microframework that emphasizes simplicity and flexibility. It was created by Armin Ronacher in 2010 and has since gained popularity for its ease of use and scalability. Flask is particularly suited for small to medium-sized web applications, though it can also handle more complex projects with the right extensions and configurations.

The structure of a Flask application is quite simple. The main directory typically includes the following components:

- app.py or main.py: The primary Python file that contains the main application code. This file defines the routes, views, and any other logic for handling requests.
- templates/: This folder contains HTML files that are used to render dynamic content in the browser. Flask uses Jinja2 templating to generate dynamic HTML content.

- static/: A directory for static files such as CSS, JavaScript, images, and other assets that do not change, which are served directly to clients.
- __init__.py: If the Flask app is structured into multiple modules, this file initializes the application and imports other parts of the app. It can also be used to configure the application, set up the database, and register routes.
- requirements.txt: A file that lists all the Python packages needed to run the application. This is used by pip to install dependencies.
- config.py: A configuration file where application settings (like database connections, secret keys, and other settings) are stored.
- models.py (optional): This file contains the database model definitions for the app, especially when using an ORM (like SQLAlchemy) to interact with a database.

3.9 GPS localization integrated with the map API

OpenStreetMap is a collaborative, open-source project that aims to create a free, editable map of the world. Founded in 2004, OSM has grown into a comprehensive geographic database with millions of contributors worldwide. It provides free geographic data and map imagery for various applications, including web mapping, navigation, and location-based services. OSM data is available under the Open Database License, allowing users to freely use, share, and adapt the information.

And Leaflet is an open-source JavaScript library for creating interactive, mobile-friendly maps. Developed by Vladimir Agafonkin and first released in 2011, Leaflet has become one of the most popular mapping libraries due to its simplicity, performance, and extensibility. It supports various map providers, including OpenStreetMap, and offers a wide range of plugins for additional functionality. Leaflet is designed to work efficiently across all major desktop and mobile platforms, making it an excellent choice for web-based mapping applications.

So OpenStreetMap with Leaflet provides a powerful solution for displaying geographic locations IoT systems. This approach offers several advantages, including

customizable map styles, the ability to zoom to specific areas, and the option to overlay additional information on the map.

3.10 SQLite database

Table 3.2: Comparison between SQLite and SQL

Feature	SQLite	Traditional SQL Databases
Type	Serverless, self-contained	Client-server based
Storage	Single file	Multiple files
Concurrent Access	Limited (file-based locking)	Robust (row-level locking)
Data Types	Dynamic typing	Strict typing
Scalability	Suitable for smaller datasets	Suitable for large-scale applications
Transactions	ACID compliant	ACID compliant
Network Access	Not required	Required
Setup	Minimal	Complex
Security	Basic	Advanced
Concurrent Users	Single writer, multiple readers	Multiple writers and readers
Size	Lightweight (< 1MB)	Larger footprint
Performance	Fast for read operations	Optimized for complex queries
Portability	Highly portable	Less portable

SQLite is a self-contained, serverless, zero-configuration, transactional SQL database engine. It's a popular choice for applications that need a simple, lightweight, and efficient database solution. SQLite has some key points that suitable for embedded systems:

- Self-contained: SQLite is a complete database system that resides in a single file. It doesn't require a separate server process, which simplifies setup and maintenance.
- Serverless: Unlike traditional database management systems, SQLite operates directly from the application, without the need for a server process. This reduces complexity and overhead.

- Zero-configuration: SQLite does not require any configuration or setup. It's ready to use out of the box, making it extremely convenient for developers.
- Lightweight: The entire SQLite library is small in size, usually under 1 MB, which makes it ideal for applications where resources are limited.
- Transactional: SQLite supports ACID (Atomicity, Consistency, Isolation, Durability) transactions, ensuring reliable data handling and integrity.
- Cross-platform: SQLite is cross-platform and can be used on various operating systems, including Windows, Linux, and macOS.

Table 3.2, comparing the key features of the SQLite database system with SQL. SQLite excels in scenarios requiring a lightweight, embedded, and file-based database solution, particularly where server-based database systems are impractical or unnecessary.

3.11 AWS Public IP Address

In the context of Amazon Web Services (AWS), a public IP address is a globally routable IP address that allows resources within an AWS environment to communicate with the internet. This address is essential for enabling external clients and devices to connect to services hosted within AWS, such as an MQTT broker.

AWS Public IP Addresses have these features:

- Global Reachability: A public IP address provides a globally unique identifier for a resource within AWS, enabling communication from any internet-connected device.
- Dynamic or Static Allocation: AWS offers both dynamic and static public IP addresses. Dynamic addresses may change when an instance is stopped and restarted, while static addresses (Elastic IPs) remain constant.
- Association with EC2 Instances: Public IP addresses are typically associated with Elastic Compute Cloud (EC2) instances, which are virtual servers within AWS.

- Network Address Translation (NAT): When an EC2 instance with a public IP address communicates with the internet, AWS performs NAT to translate the instance's private IP address to its public IP address.
- Security Considerations: Public IP addresses are exposed to the internet and therefore require appropriate security measures, such as firewalls and security groups, to protect against unauthorized access.

An MQTT broker is a server that facilitates communication between MQTT clients using a publish-subscribe messaging pattern. When deploying an MQTT broker within AWS, a public IP address can be used as the endpoint for clients to connect to. Using an AWS Public IP Address for an MQTT Broker provide some benefits:

- Accessibility: Enables clients from anywhere on the internet to connect to the MQTT broker.
- Scalability: Allows for easy scaling of the MQTT broker infrastructure by adding more EC2 instances.
- Flexibility: Provides flexibility in choosing the MQTT broker software and configuration.
- Cost-Effectiveness: Can be a cost-effective solution for small to medium-sized MQTT deployments.

So an AWS public IP address serves as a crucial component for enabling external connectivity to resources within AWS, including MQTT brokers. By properly configuring an EC2 instance with a public IP address and an MQTT broker, clients can connect and exchange messages over the internet. It is essential to implement appropriate security measures and consider factors such as scalability and network performance when using a public IP address for an MQTT broker endpoint.

CHAPTER 4. INTELSCHOOLBUS – THE SYSTEM DESIGN FOR COUNTING AND CONTROLLING STUDENTS ON SCHOOL BUSES

4.1. System design

4.1.1 The transmitter

The transmitter side of the system includes a power supply for MCU, components (DC fan, Infrared light, LED) and a Wi-Fi router powered by a 4G SIM as Figure 4.1.

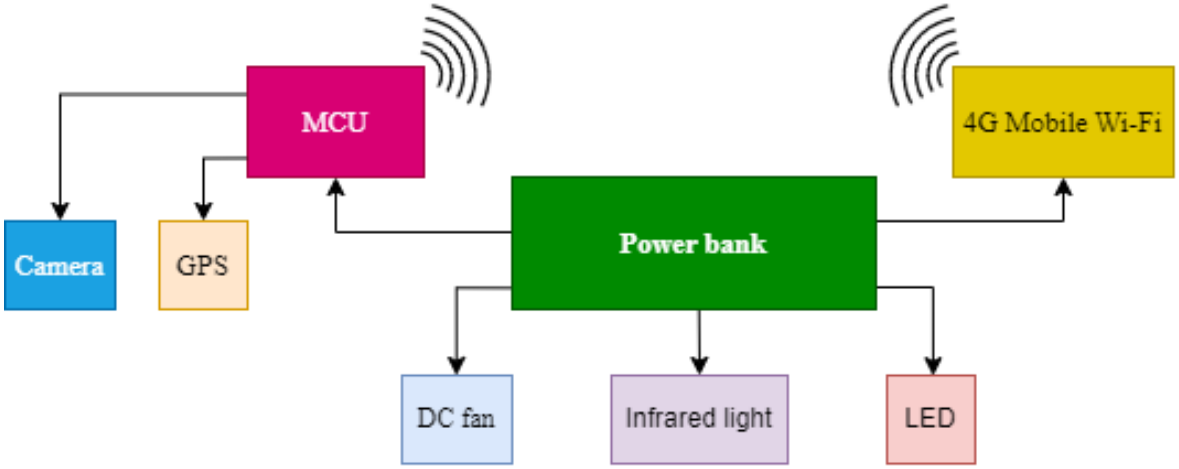


Figure 4.1: Transmitter

4.1.2 The receiver

In the receiver side, as shown in Figure 4.2, includes a web application powered by Flask and a database using SQLite. The web application will receive information such as GPS data, the number of students boarding and alighting, memory card capacity, and system temperature from the transmitter and store it in the database.

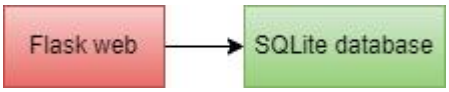


Figure 4.2: Receiver

4.2. Detailed transmitter side design

4.2.1. Power bank

Figure 4.3: Power bank

(Source: Energizer)

The system uses the Energizer 20,000mAh / 3.7V Li-Polymer Power Bank - UE20010 (Figure 4.3) for the system because it includes Energizer's comprehensive safety features, such as protection against overcharging, over-discharging, overcurrent, and short circuits. This focus on safety is essential, especially given the potential hazards associated with lithium-ion batteries and the deployment environment on buses, where flammable materials are present.

4.2.2. 4G Mobile Wi-Fi

The system will connect to the internet using the TP-Link M7200 4G LTE 150Mbps Mobile Wi-Fi Router, as shown in Figure 4.4. This device ensures a stable network connection with a download speed of 150 Mbps and an upload speed of 50 Mbps.

Figure 4.4: 4G Mobile Wi-Fi

(Source: TP-Link)

4.2.3. MCU

The heart of system is MaixCAM, it can called to be a foray into the frontier of embedded AI, powered by RISC-V and a dedicated Neural Processing Unit (NPU). Developed by Sipeed, MaixCAM leverages the open-source RISC-V standard, known for its flexibility, modularity, and transparency. Unlike proprietary architectures, RISC-V enables extensive customization and optimization, allowing designers to tailor the processor's core to specific application demands. In MaixCAM, this results in a highly efficient, power-conscious processing unit ideal for embedded devices with limited resources.

Figure 4.5 illustrates the appearance of the MaixCAM and its connection pins. The image highlights the board's layout, showing essential pins such as power input, UART, I2C, GPIO, and SPI interfaces.

At its core, MaixCAM features the 1GHz RISC-V C906, a processor capable of running at 700 MHz. This specialized hardware accelerator is designed for computationally intensive matrix operations central to modern deep learning algorithms. By offloading such tasks from the main CPU cores, the NPU accelerates

AI inference, enabling real-time applications like object detection, image classification, and facial recognition.

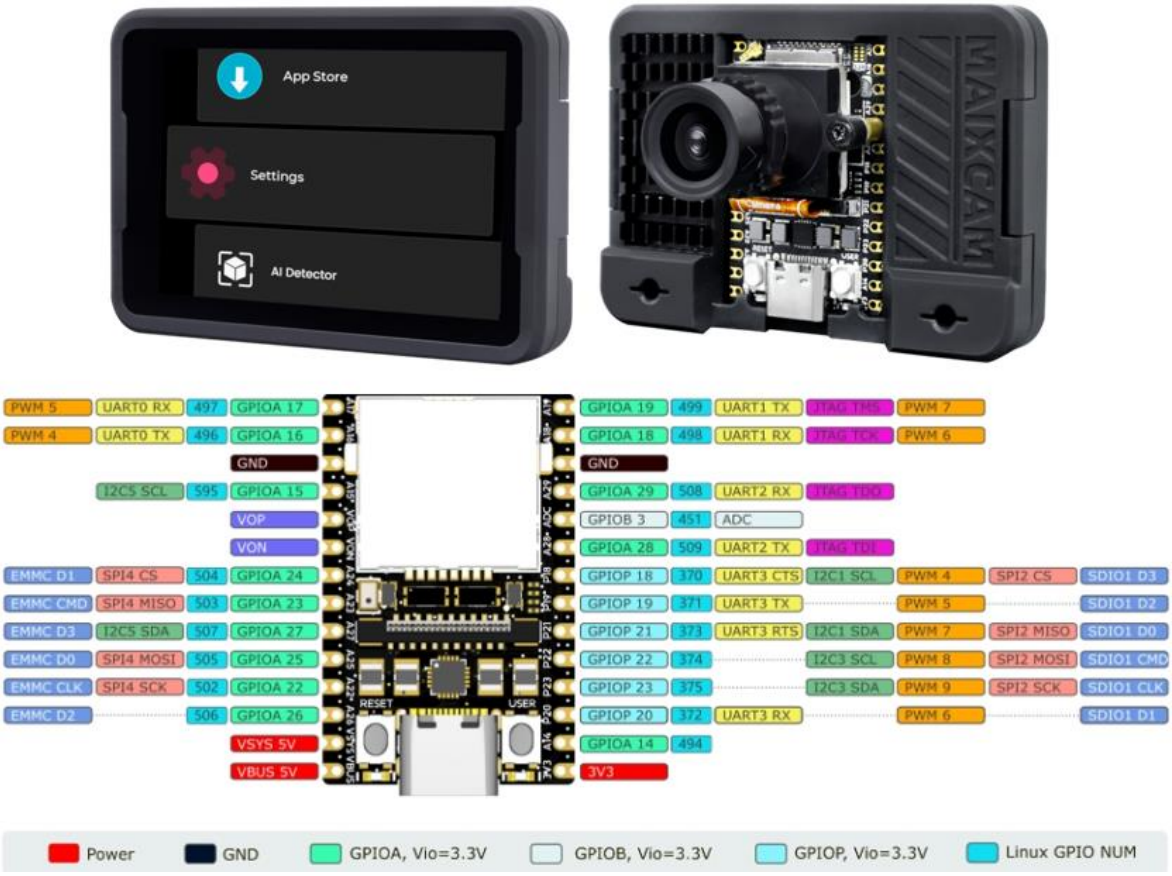


Figure 4.5: MaixCAM and its pins layout

(Source: Sipeed)

The NPU delivers 1TOPS@INT8, it means the system can perform one trillion 8-bit integer operations per second. The synergy between the RISC-V CPU and the dedicated NPU defines MaixCAM's unique capabilities. The CPU handles system control, data preprocessing, and AI result post-processing, while the NPU focuses on core AI computations. This balanced division of tasks creates an optimized system for high-performance, real-time applications like real-time people counting.

The comparison in Table 4.1 highlights the specialized embedded capabilities of the Sipeed MaixCAM against the general-purpose performance and expandability of the Raspberry Pi 4 Model B.

Table 4.1: Comparison Between Sipeed MaixCAM and Raspberry Pi 4 Model B (8GB RAM)

Specification	Sipeed MaixCAM	Raspberry Pi 4 Model B (8GB RAM)
CPU	RISC-V C906, 1GHz	Broadcom BCM2711, Quad-core Cortex-A72, 1.5GHz
RAM	256MB DDR3	8GB LPDDR4-2400 SDRAM
Camera Support	Supports up to 5MP cameras (officially 4MP)	Camera support via MIPI CSI port
Display	2.3-inch IPS touchscreen, 552x368 resolution	No built-in display
Wireless Connectivity	WiFi 6 and BLE 5.4	WiFi 802.11ac (2.4GHz and 5GHz), Bluetooth 5.0
USB Ports	USB Type-C (USB 2.0)	2 USB 3.0 and 2 USB 2.0 ports
GPIO	Two 14-pin GPIO headers	40-pin GPIO header
Storage	TF card / SD NAND	Micro-SD card slot
Power Consumption	Powered via USB Type-C with 5V/100mA	Powered via USB-C (minimum 5V/3A required)

4.2.4. GPS

GPS Module NEO M8N is a high-performance GPS device developed by u-blox, offering precise positioning and high reliability. Equipped with 72 channels, it supports multiple satellite constellations, including GPS, GLONASS, Galileo, and BeiDou. The module provides positioning accuracy of up to 2.0 meters and a startup sensitivity of -167 dBm.

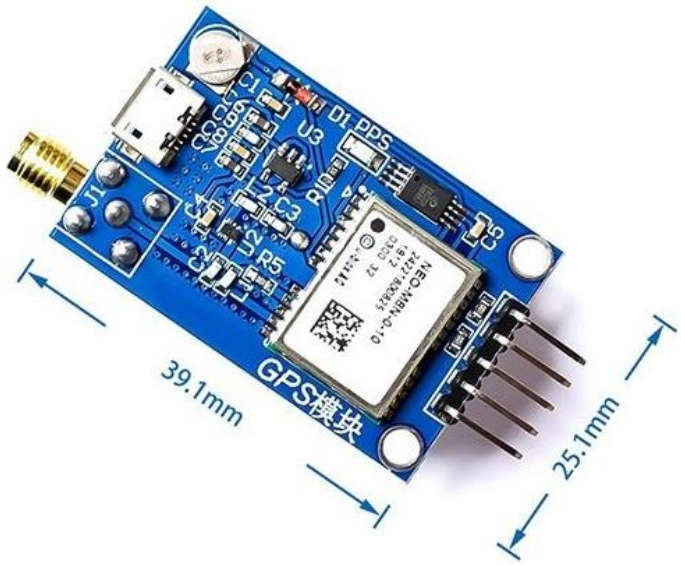


Figure 4.6: Module GPS NEO M8N

(Source: Thegioiic)

Additionally, it supports advanced GPS services such as AssistNow GNSS, enhancing performance in challenging environments. It also includes an advanced GPS assistance feature, AssistNow GNSS, which enhances performance in challenging environments.

Figure 4.7: GPS' antenna with SMA port

(Source: MLTech)

The system will use a module with an SMA port (as Figure 4.6) connected to 1575.42 MHz antenna (Figure 4.7) to achieve optimal performance compared to the IPEX antenna port.

4.2.5. DC fan, infrared light and LED

The system is cooled by a 5V DC fan manufactured by Nidec, which can operate continuously. Additionally, the system is equipped with a 3W infrared light, compact in size but enabling the camera to maintain good visibility in low light conditions and even in complete darkness. To indicate the system's operational status, a green LED light is installed in series with the backup power cable, integrated with a switch (as shown in Figure 4.8), allowing for easy activation of the working mode.

Figure 4.8: Power cable

(Source: TECHCO)

4.3. Detailed receiver side design

When the transmitter side operates, it will send data in the form of packaged JSON as shown in Figure 4.9. On the receiving side, the web application running on Flask will receive this data and display it with the interface shown in Figure 4.10. The protocol used for transmitter is MQTT, sending to a public IP address of AWS, so all users outside the LAN network can view the data.

```
data = {
    "latitude": current_lat,
    "longitude": current_lon,
    "speed": current_speed,
    "up_down_count": up_down_count,
    "down_up_count": down_up_count,
    "total_gb": total_gb,
    "used_gb": used_gb,
    "free_gb": free_gb,
    "usage_percent": round(memory_usage_percent, 2),
    "storage_full": storage_full,
    "temperature": cpu_temp,
    "gps_status": "OK" if gps_serial else "N/A"
}
```

Figure 4.9: Data in JSON format

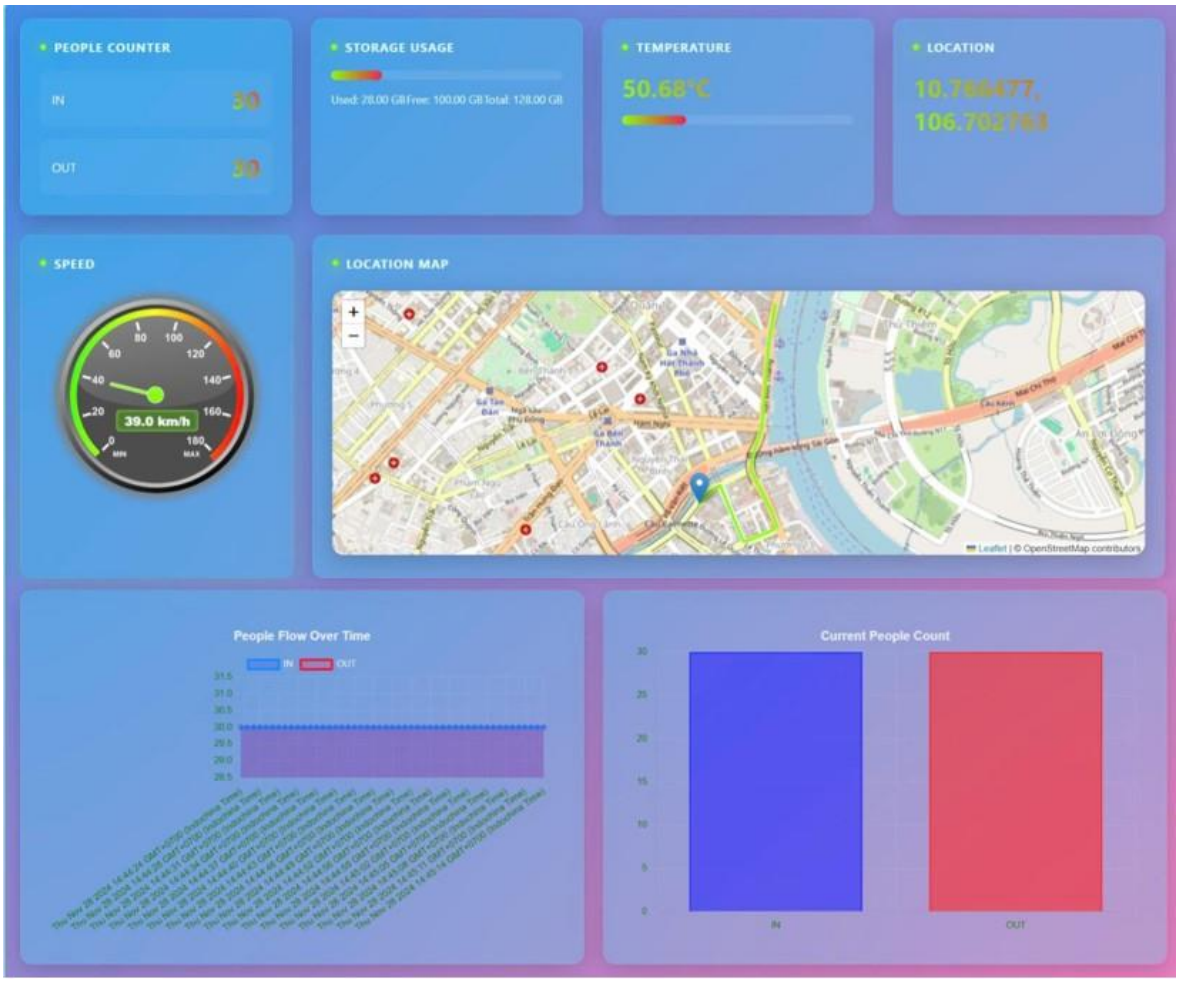


Figure 4.10: Web application

The web application displays information with optimal visual effectiveness for users, including:

- People Counter:

  - Layout: This section is clearly divided into "IN" and "OUT" counters, each displaying a numerical value.
  - Visual Emphasis: The numbers are large and prominently displayed, making them easily readable.
  - Color Coding: The use of distinct colors (likely blue for "IN" and red for "OUT") provides an immediate visual distinction between the two counts.
- Storage Usage:
  - Textual Representation: This section displays storage usage information using text labels and numerical values.
  - Clear Labels: The labels "Used," "Free," and "Total" are clearly defined, making the information easy to understand.
  - Progress Bar: A horizontal progress bar visually represents the proportion of used storage, providing a quick overview of storage capacity.
- Temperature:
  - Numerical Display: The temperature is displayed as a numerical value with a degree Celsius symbol.
  - Large Font: The numerical value is displayed in a large font, making it easily readable.
  - Progress Bar: A horizontal progress bar visually represents the temperature level, providing a quick overview of the temperature range.
- Location:
  - Coordinate Display: The location is displayed as latitude and longitude coordinates.
  - Clear Formatting: The coordinates are clearly formatted, making them easy to read.
- Speed:
  - Gauge Meter: The speed is displayed using a gauge meter, which provides a visual representation of the current speed.

  - Numerical Display: The current speed is also displayed numerically below the gauge.
  - Color Coding: The gauge meter uses color coding (likely green for safe speeds) to provide a quick visual indication of the speed range.
- Location Map:
  - Interactive Map: This section displays an interactive map, likely using Leaflet or a similar library.
  - Markers: The map displays markers, which likely represent locations of interest.
  - Zoom and Pan: The map provides zoom and pan controls for navigation.
- People Flow Over Time:
  - Line Chart: This section displays a line chart representing the flow of people over time.
  - Clear Labels: The chart includes clear labels for the axes and data series.
  - Color Coding: The chart uses color coding (likely blue for "IN" and red for "OUT") to distinguish between the two data series.
- Current People Count:
  - Bar Chart: This section displays a bar chart representing the current number of people "IN" and "OUT."
  - Clear Labels: The chart includes clear labels for the bars.
  - Color Coding: The chart uses color coding (likely blue for "IN" and red for "OUT") to distinguish between the two counts.

These components are exceptionally well-suited for displaying numerical data, particularly for counting purposes. The use of large, clear fonts, distinct color coding, and visual aids like progress bars, gauge meters, and charts ensures that users can easily and quickly grasp the numerical information presented. The design prioritizes readability and clarity, making it effortless for users to monitor the system's status and track key metrics.

4.4. Circuit diagram

The circuit diagram in Figure 4.11 is designed using Proteus software. The T-L, T-R, B-L, and B-R rails are the four fixed mounting points for the MaixCAM. The 5V-T rail supplies power to the infrared light, while the 5V-B rail powers the DC fan. The GPS rail is designated for connecting to the GPS module, which features an SMA port for antenna connection. The spacing between the T-L, T-R, and B-L, B-R rails serves as the passage through which the MaixCAM's lens extends.

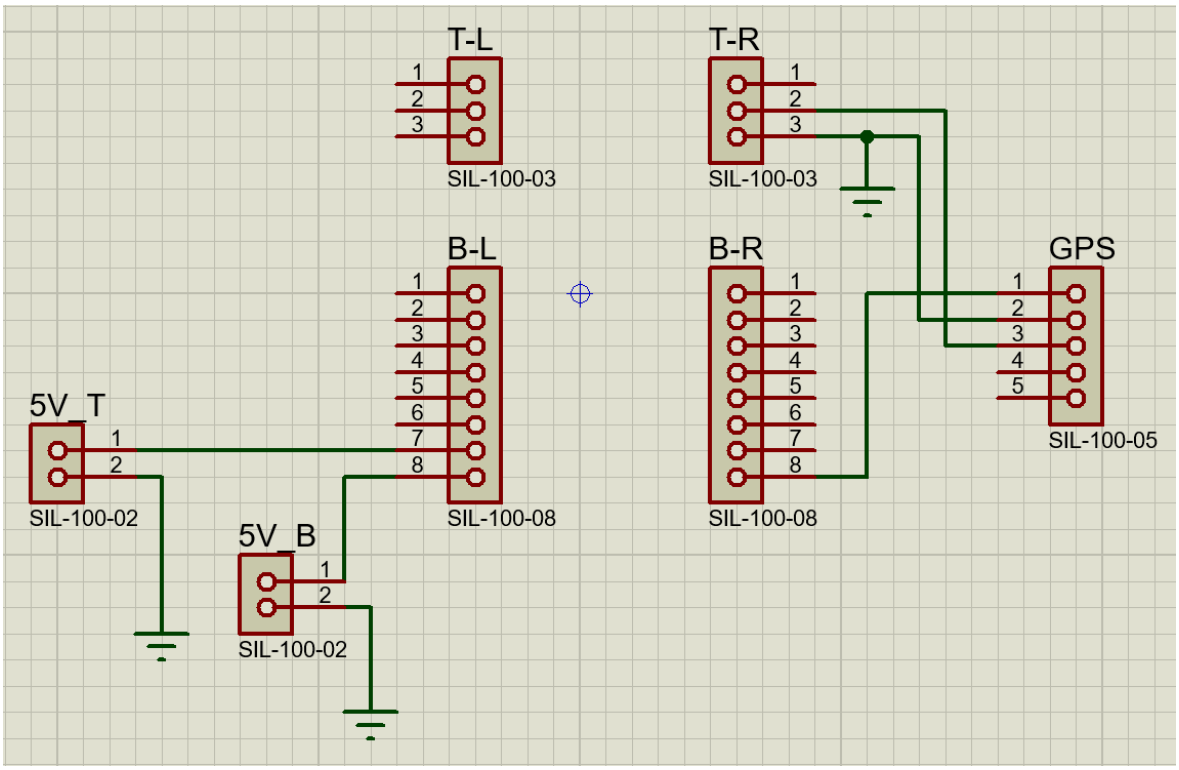


Figure 4.11: Circuit diagram

Regarding the components, the T-L, T-R, B-L, and B-R rails will use female headers, while the GPS and the two 5V power supply sections will use XH2.54mm 5-pin male headers and 2-pin headers for connection via 24AWG 2.54mm Bus cables. This type of connection ensures industrial standard quality, preventing loosening due to vibrations or electrostatic discharge during operation.

4.5. PCB layout

Figure 4.12 illustrates the system's PCB layout, which designed with a ground plane to minimize interference and ensure appropriate spacing between wires. The PCB design opted not to use Proteus's auto-routing feature, choosing instead to manually route the connections. The component placement was carefully measured and arranged in advance to ensure a precise and compact fit. The PCB design features a well-optimized size of 60x50 mm.

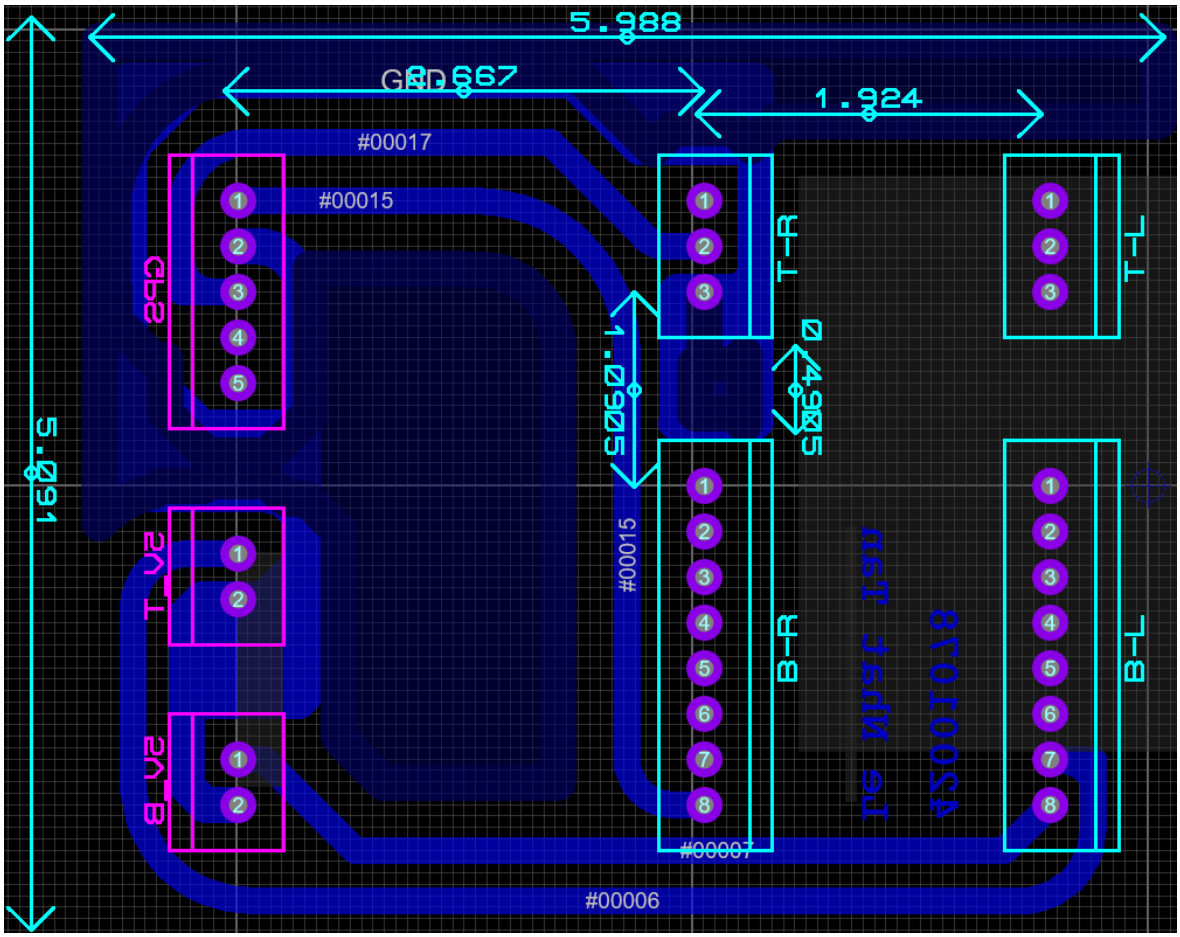


Figure 4.12: PCB layout

4.6. Calculating system parameters

4.6.1. DC fan

The Nidec-brand DC fan, used for cooling, operates at 5V, matching the output voltage of the MaixCAM. Similarly, the infrared light runs at the same voltage. The

power indicator LED is connected with a 300-ohm resistor to function properly with the 5V power supply, ensuring it stays within its operating voltage range of 2.8 to 3.2V.

4.6.2. 3W Infrared light

A 3W infrared (IR) light source is well-suited for cameras in nighttime or low-light environments because it provides illumination that is invisible to the human eye but highly sensitive to camera sensors, enabling clear image capture even in darkness. The 3W power output provides sufficient IR light for effective illumination. Furthermore, operating at 5V, the light source is considered safe for use with common low-voltage power supplies, minimizing the risk of electrical hazards.

4.6.3. LED

The current through the LED and resitor is calculated using the formula in Equation (4.1), and the result falls within the range under 20 mA, which is considered appropriate.

$$I = \frac{V_R}{R} = \frac{V_{source} - V_{LED}}{R} = \frac{5 - 3}{300} = 0.00667\ A = 6.67\ mA \qquad (4.1)$$

4.6.4. Antenna for module GPS

A resonant antenna operating at a frequency of 1575.42 MHz is optimally suited for use with the u-blox NEO-M8N GPS module and is compliant with the L1 band standard. This frequency corresponds precisely to the center frequency of the L1 band, which is the primary frequency utilized by the Global Positioning System (GPS) for civilian positioning and navigation. The NEO-M8N module is specifically designed to receive and process signals within this L1 band, making a 1575.42 MHz antenna a highly compatible and efficient choice for signal reception. Employing an antenna with this resonant frequency ensures optimal signal capture, thereby maximizing the accuracy and reliability of the GPS positioning data provided by the NEO-M8N module. Furthermore, the use of a 1575.42 MHz antenna aligns with the

established L1 standard, ensuring interoperability and adherence to the defined specifications for GPS signal reception.

4.7. Algorithm flowchart

4.7.1. The transmitter

Figure 4.13 illustrates the detailed operation of the transmitter side. It’s a program implements a real-time object detection and tracking system designed for a MaixCAM device. The system utilizes a YOLOv8 model for object detection, a ByteTrack algorithm for object tracking, and a UART interface for GPS data acquisition. The system continuously counts objects entering and exiting a defined region of interest (ROI) within the camera's field of view. Detected objects trigger image capture, which are then stored locally.

Furthermore, the system transmits the object counts, GPS coordinates, speed, storage status, and temperature data to an MQTT broker for remote monitoring. The system prioritizes performance, ensuring continuous counting and data transmitter, while capturing images only when objects are detected within the frame.

The code also includes a memory card capacity check; if the storage is full, image capture will stop. Collected images can be compiled into a video at 25 FPS, meaning each frame is captured at 40 ms intervals.

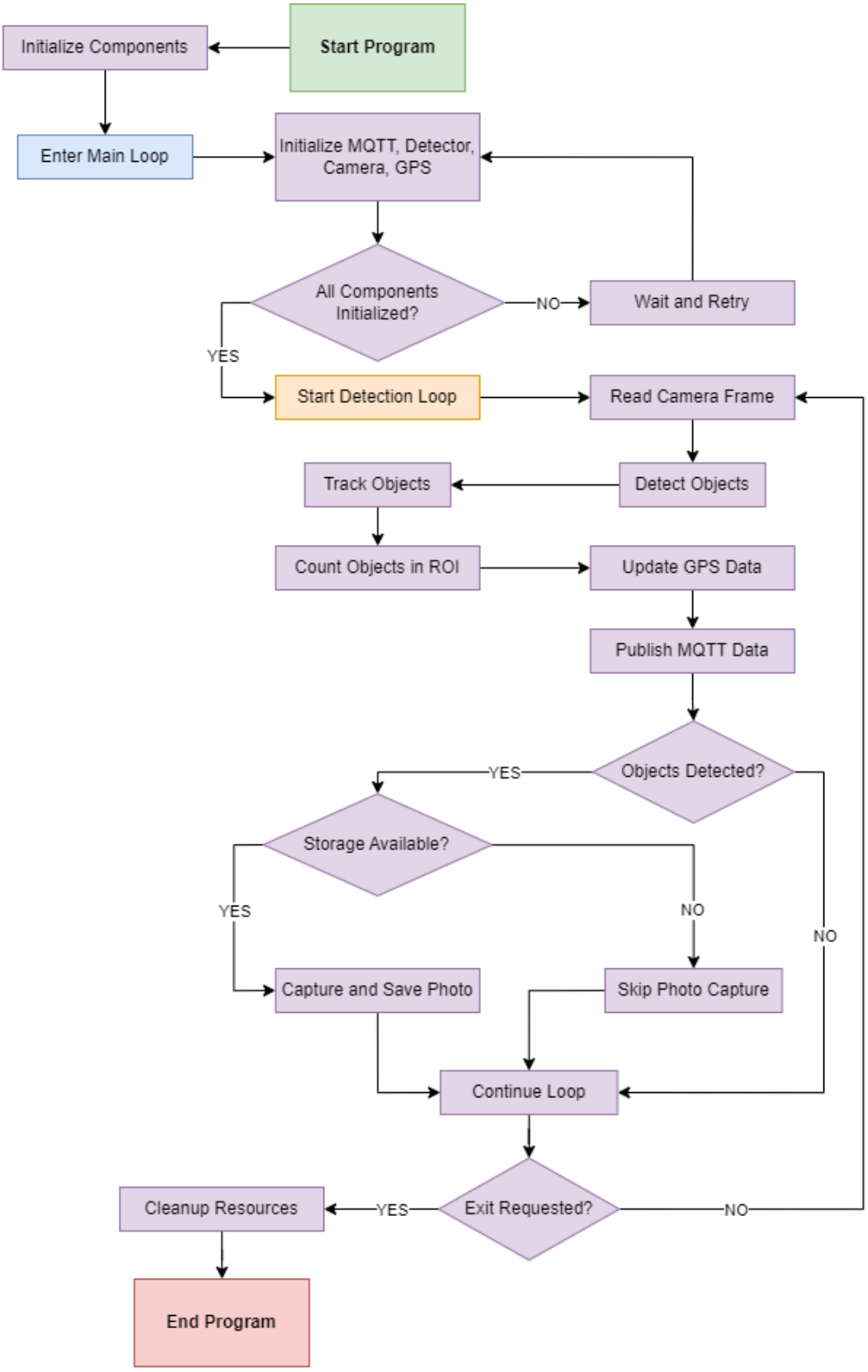


Figure 4.13: Algorithm flowchart of transmitter side

4.7.2. The receiver

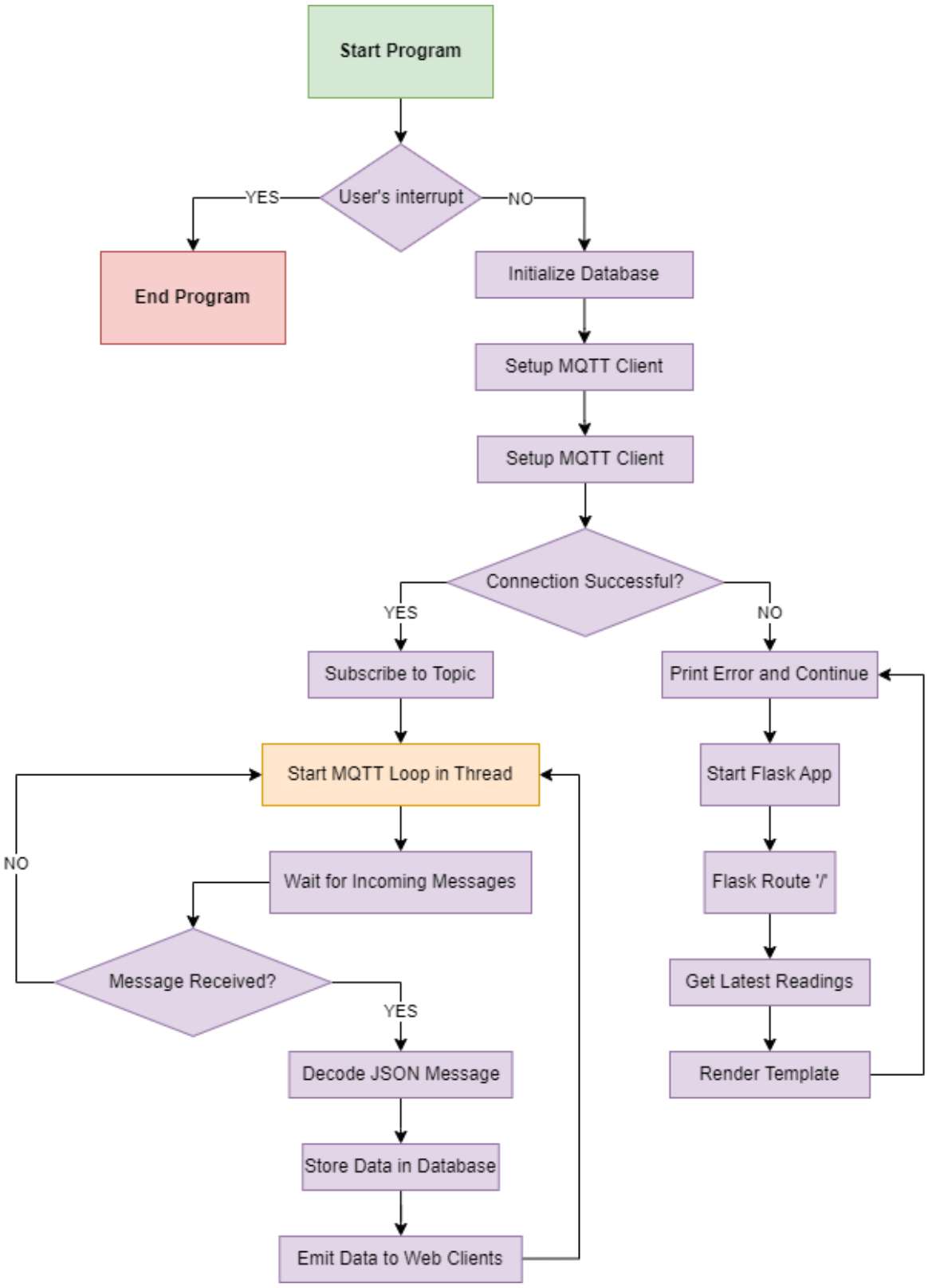


Figure 4.14: Algorithm flowchart of reception side

Figure 4.14 show the algorithm flowchart of program that implements a data acquisition and visualization system. It establishes a connection to an MQTT broker, subscribes to a specific topic, and processes incoming JSON messages. Upon receiving a message, the system stores the data in a local SQLite database and updates a real-time web interface using Flask and SocketIO. The system initializes a database upon startup, creates a table for received messages, and provides a web interface to display the latest data and historical readings. The MQTT communication is handled in a separate thread to ensure asynchronous operation.

CHAPTER 5. IMPLEMENTATION RESULTS

5.1. Experimental process

First, the project experimented with traditional models and tracking methods on Raspberry Pi 4 Model B with 8GB RAM. The device was tested with pre-trained models and traditional tracking algorithms, but the results in Table 5.1 revealed the limited suitability of the device priced at approximately $80 (according to DigiKey). It can be concluded that this hardware does not meet the requirements for real-world scenarios, where fast-moving people are difficult to track accurately.

Table 5.1: Performance comparison of different detection model and tracking methods on Raspberry Pi 4 8GB RAM

Input Resolution	Model and Method	People counting integration	FPS	Performance and Issues
Webcam 300x300	Pre-trained MobileNet_v1 converted to TFLite	Yes	6-8	Stable for counting individuals
Webcam 640x480	Pre-trained MobileNet_v1 converted to TFLite	Yes	5-7	Stable for counting individuals
Video 1920x1080 & 640x480	Pre-trained MobileNet_v1 converted to TFLite	Yes	~5	Some counting errors, misidentification
Video 1920x1080 & 640x480	Fine-tuned YOLOv9 (350 images, 79	Yes	<1	Poor performance

	epochs, mAP = 0.96) converted to TFLite16/32			
Webcam 640x480	Pre-trained YOLOv8n converted to TFLite16/32	No	<1	Poor performance
Webcam 640x480	Pre-trained YOLOv8n converted to TFLite with Deep Sort (real-time)	No	<1	Poor tracking, errors
Webcam 640x480	Pre-trained YOLOv8n converted to TFLite with Deep Sort	No	<1	Poor tracking, errors
Webcam 640x480	Pre-trained MobileNet_v2 using PyTorch + FasterRCNN	No	<1	Poor performance
Webcam 640x480	Pre-trained SSD300_VGG16 using PyTorch	No	<1	Poor performance
Webcam 1920x1080	OpenCV-based color detection	Yes	~30	Used for testing webcam
Video 640x480	MOG2 for separating	Yes	~12	Limited application,

	moving objects from the background			suitable for ideal inputs
Video 1080x1920	Pre-trained YOLOv8s with 1/3 frame processing rate (skipping 66.67% of frames). Tracks centroid, matches based on Euclidean distance, assigns and manages IDs	Yes	0-6	~50% errors
Video 1080x1920	Pre-trained YOLOv8n, tracks centroid, matches based on Euclidean distance, assigns and manages IDs	Yes	0-6	Accurate
Webcam 1080x1920	Pre-trained YOLOv8n, tracks centroid, matches based on Euclidean distance, assigns and manages IDs	Yes	<1	Poor performance

Webcam 640x640	Object tracking with CamShift using OpenCV	No	12-16	Inaccurate tracking

To address the performance issue, the system transitioned to using MaixCAM, a hardware platform with a RISC-V CPU architecture and an integrated NPU offering 1 TOPS@INT8 performance, as previously introduced. The experiment also employed pre-trained models, and the results presented in Table 5.2 demonstrated promising performance, making the device suitable for real-world deployment.

Table 5.2: Performance comparison of pre-trained models on MaixCAM

Model	Tracker	Number of Objects in Frame	FPS	Accuracy Rating
Pre-trained yolov5s	ByteTrack	0	34	v8>v5>11
		1	30	
		1-5	19	
		5-10	14	
		>10	12.5	
Pre-trained yolov8n	ByteTrack	0	35	v8>v5>11
		1	29	
		1-5	19	
		5-10	16	
		>10	14	
Pre-trained yolo11n_32o24_int8	ByteTrack	0	34	v8>v5>11
		1	29	
		1-5	20	
		5-10	18	
		>10	14	

After discovering the capabilities of MaixCAM, the project conducted tests using custom models, which were trained on a single class: human heads. Table 5.3

provides detailed information about the processing time and the number of images captured during deployments on school buses transporting students from Ruby and Tri Duc schools. Earlier in October, a survey of the school buses was conducted to determine the optimal installation positions for the system.

Table 5.3: Statistics of collected images

Time	Number of images
04/11 – 10/11	792
11/11 – 17/11	629
16/11 – 22/11	1379
30/11 – 05/12	668
Total	3468

Figure 5.1 shows the number of images and labeled objects annotated on the Roboflow platform. With this number of images, it will be proceeded to fine-tune the YOLOv8n model using the configuration specified in Table 5.4, along with preprocessing and augmentation methods as shown in Figure 5.2. Applying these methods will help increase the number of images to 10000.

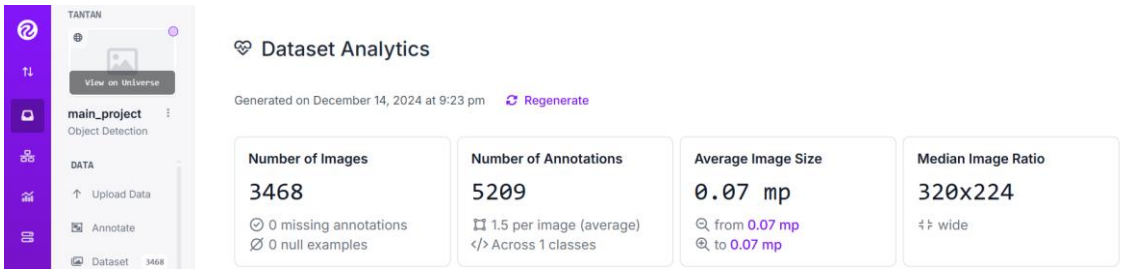


Figure 5.1: The project on Roboflow

Table 5.4: Fine-tuning configuration parameters

Parameters	Value
Epochs	120
Imgsz	640
Batch	64

Figure 5.3 illustrates the statistics for the epochs, showing that the loss value has stopped decreasing, indicating that the model has converged.

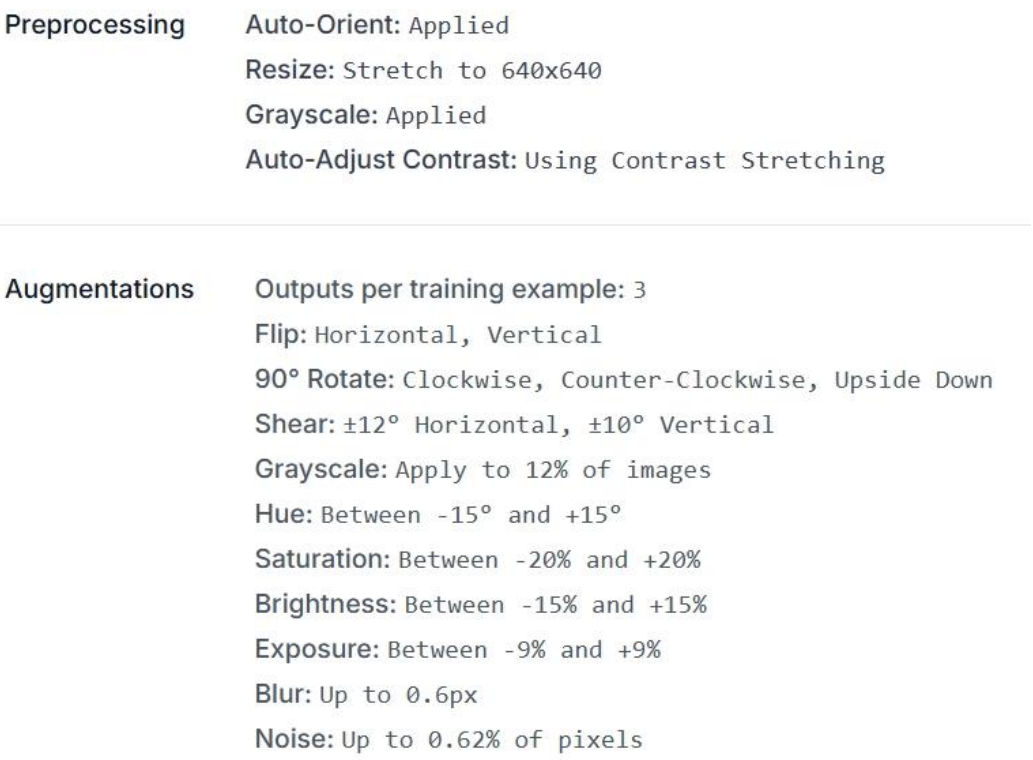


Figure 5.2: Image preprocessing and augmentation methods

epoch	time	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
92	7574.32	0.67804	0.3607	0.99281	0.99844	0.98985	0.9948	0.89012	0.50113	0.19778	0.88495	0.0024925	0.0024925	0.0024925
93	7656.99	0.68397	0.35721	0.99597	0.99883	0.98985	0.9948	0.89361	0.50266	0.19897	0.88553	0.00241	0.00241	0.00241
94	7738.71	0.68975	0.3598	0.99788	0.99888	0.98985	0.9948	0.89454	0.50384	0.19891	0.88514	0.0023275	0.0023275	0.0023275
95	7820.79	0.67534	0.36192	0.98956	0.99883	0.98985	0.9948	0.89575	0.5042	0.19944	0.8856	0.002245	0.002245	0.002245
96	7902.54	0.67654	0.35208	0.98039	0.99881	0.98985	0.9948	0.89372	0.50392	0.1989	0.88485	0.0021625	0.0021625	0.0021625
97	7985.17	0.67711	0.35366	0.9897	0.99874	0.98985	0.9948	0.89331	0.50472	0.19857	0.88498	0.00208	0.00208	0.00208
98	8066.94	0.66524	0.34524	0.98357	0.99991	0.98985	0.99485	0.89494	0.50549	0.19974	0.88607	0.0019975	0.0019975	0.0019975
99	8149.41	0.66874	0.34516	0.97637	1	0.98973	0.99485	0.89393	0.50431	0.19702	0.8854	0.001915	0.001915	0.001915
100	8232.07	0.67102	0.34984	0.98228	1	0.98967	0.99485	0.89345	0.50242	0.19727	0.88476	0.0018325	0.0018325	0.0018325
101	8314.58	0.65373	0.34613	0.97391	1	0.98918	0.99485	0.89314	0.50104	0.19535	0.88439	0.00175	0.00175	0.00175
102	8397.24	0.65911	0.34277	0.98344	1	0.98923	0.99485	0.8937	0.50059	0.19443	0.88438	0.0016675	0.0016675	0.0016675
103	8479.16	0.65356	0.34075	0.9805	1	0.98945	0.99485	0.89491	0.50072	0.1935	0.88477	0.001585	0.001585	0.001585
104	8561.32	0.65816	0.34118	0.97877	1	0.98963	0.99485	0.89467	0.50036	0.19418	0.88442	0.0015025	0.0015025	0.0015025
105	8643.36	0.65629	0.33908	0.97999	1	0.98928	0.99485	0.89678	0.4996	0.1924	0.88421	0.00142	0.00142	0.00142
106	8725.67	0.64627	0.33357	0.97043	1	0.98893	0.99485	0.8984	0.49865	0.1947	0.88414	0.0013375	0.0013375	0.0013375
107	8807.11	0.6471	0.33895	0.97609	1	0.98877	0.99485	0.89843	0.49799	0.19368	0.88392	0.001255	0.001255	0.001255
108	8888.89	0.64102	0.33402	0.97079	1	0.98871	0.99485	0.90032	0.49758	0.19307	0.8838	0.0011725	0.0011725	0.0011725
109	8970.41	0.64418	0.3333	0.97771	1	0.98837	0.99485	0.8986	0.49765	0.19212	0.88365	0.00109	0.00109	0.00109
110	9052.68	0.63819	0.33173	0.97113	1	0.98847	0.99485	0.89766	0.49711	0.19123	0.88337	0.0010075	0.0010075	0.0010075
111	9132.34	0.58877	0.27039	0.93175	1	0.9884	0.99485	0.89717	0.49645	0.19283	0.88296	0.000925	0.000925	0.000925
112	9209.63	0.57868	0.26499	0.92628	1	0.98853	0.99485	0.89654	0.49636	0.19214	0.88254	0.0008425	0.0008425	0.0008425
113	9286.56	0.57118	0.25766	0.92817	1	0.98893	0.99485	0.89447	0.49605	0.19267	0.88261	0.00076	0.00076	0.00076
114	9364.51	0.57201	0.25787	0.92646	1	0.98932	0.99485	0.89765	0.49598	0.19252	0.88236	0.0006775	0.0006775	0.0006775
115	9441.63	0.56981	0.26	0.92605	1	0.98981	0.99485	0.89602	0.49499	0.1931	0.88232	0.000595	0.000595	0.000595
116	9518.51	0.56643	0.25907	0.92378	0.99957	0.98985	0.99485	0.89701	0.49496	0.19308	0.88221	0.0005125	0.0005125	0.0005125
117	9595.8	0.56113	0.25292	0.91957	0.99939	0.98985	0.99485	0.89493	0.49475	0.19254	0.8824	0.00043	0.00043	0.00043
118	9672.58	0.55435	0.25084	0.91895	0.99945	0.98985	0.99485	0.89801	0.49467	0.19333	0.88222	0.0003475	0.0003475	0.0003475
119	9749.87	0.5504	0.24705	0.91982	0.99974	0.98985	0.99485	0.89938	0.49525	0.19242	0.88219	0.000265	0.000265	0.000265
120	9827.27	0.5567	0.2471	0.9198	1	0.98946	0.99485	0.8988	0.49543	0.19541	0.88227	0.0001825	0.0001825	0.0001825

Figure 5.3: Statistics for the epochs

The Figure 5.4 displays some cases where students are detected using the pretrained YOLOv8n model run on laptop, while Figure 5.5 presents cases where students are detected using custom YOLOv8n model. A comparison shows that the custom model specifically for head detection, performs effectively, successfully identifying most cases. Note that the YOLO model, after fine-tuning, will be converted to the ONNX format and then changed to the cvimodel format using

Sipeed's tool. Figure 5.6 shows content of the .mud file containing information about the cvimodel, MaixCAM will reference this file to access the cvimodel.

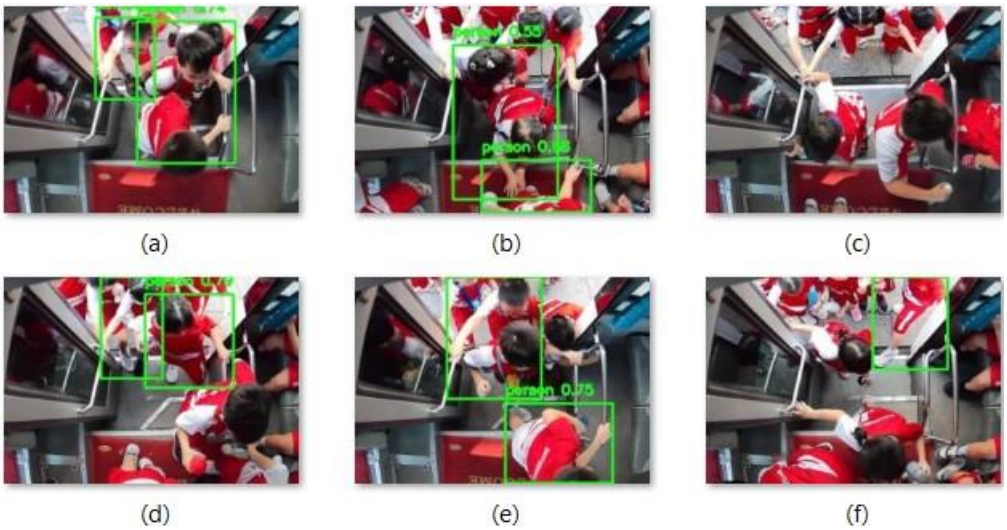


Figure 5.4: The students are detected using a pretrained model. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5; (f) Case 6

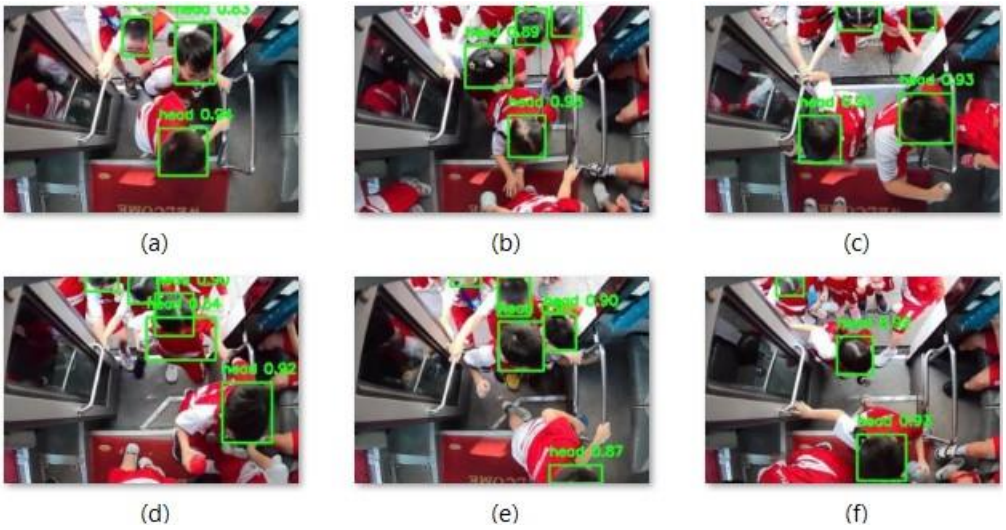


Figure 5.5: The students are detected using custom trained model. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5; (f) Case 6

The system was also measured and tested to evaluate its crash rate over 14 days, operating approximately 6 hours per day, yielding highly positive results with a crash rate of 0%. Regarding startup time, the system requires about 30 seconds to transition from an off state to connecting to Wi-Fi, starting the counting process, sending data via MQTT, and connecting to satellites for GPS signals. Depending on the

infrastructure, the average time for GPS connection, measured over 10 trials, is around 5 minutes.

```
[basic]
type = cvimodel
model = yolov8n_cv181x_int8_sym.cvimodel

[extra]
model_type = yolov8
input_type = rgb
mean = 0, 0, 0
scale = 0.00392156862745098, 0.00392156862745098, 0.00392156862745098
labels = head
```

Figure 5.6: File .mud

In terms of cooling performance, if the cooling system is removed and the system runs continuously for 12 hours in a non-air-conditioned environment, the temperature stabilizes at 77°C. However, with the cooling system active and in an air-conditioned environment, the temperature drops by more than 32°C, remaining below 45°C. Figure 5.7 shows the average temperature of the system when running for 15 minutes, measured using the MQTT Explorer application on laptop.

The system is powered by a 20,000 mAh portable battery, which has been verified to be able to operate for 12 hours when providing power to the MaixCAM and 4G Mobile Wi-Fi.

Figure 5.7: Temperature statics

Regarding the software, the website is powered by Flask and connected to a database system. It underwent continuous testing for 24 hours without any recorded connection or data loss errors. Similar results were observed over a month-long testing period, with an average of 6 hours of operation per day.

5.2. Experimental results

Figure 5.8: The completed system

Figure 5.8 provides a detailed description of the components in the system used in conjunction, it is a block with size 115x90x55 mm contains MaixCAM, module GPS and antenas, with LED indicator, IR LED and DC fan for drawing hot air from the inside out for cooling. The system, after careful consideration, has been selected for installation at the entrance of the bus, assuming the bus has only one door for passengers. Fig. 5.9 shows the system and its position. In that position, the camera captures a comprehensive view of passengers' heads and a wide area of movement, enabling effective tracking and inference of the passengers' movement direction.

Figure 5.9: The system deployed on the bus

Figure 5.10 provides a comprehensive view of the product's exterior, highlighting two ventilation holes, while also offering a detailed depiction of the internal layout of the system. The internal components are securely arranged and fixed using PU Tape made from acrylic material and the latest Nano gel technology. This adhesive is safe, non-toxic, and provides strong bonding.

Additionally, Figure 5.11 and 5.12 showcases the precise design of the PCB layout, ensuring a perfect fit with the MaixCAM lens and maintaining an optimal distance between the headers.

Figure 5.10: System appearance details

Figure 5.14 demonstrates the functionality of the database integrated into the website, where it stores information received via MQTT. The program successfully creates an SQLite database, which can be easily viewed through the DB Browser for SQLite application. This setup is also convenient for administrators, as the SQLite table names are organized by date (Figure 5.13), making data management more quick and efficient access.

The web application interface on the receiving side, in this case, the parent's computer, is achieved as shown in Figure 5.15.

Figure 5.11: System interior details

Figure 5.12: PCB Circuit connected to MaixCAM

New Volume (E:) > T460 > DATN > data_MQTT

Name	Type
1-12-24	Data Base File
2-12-24	Data Base File
3-12-24	Data Base File
4-12-24	Data Base File
5-12-24	Data Base File
6-12-24	Data Base File
7-12-24	Data Base File
8-12-24	Data Base File
9-12-24	Data Base File
10-12-24	Data Base File
11-12-24	Data Base File
12-12-24	Data Base File

Figure 5.13: Stored database files

DB Browser for SQLite - E:\T460\DATN\data\19-12-24.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: sensor_readings | Filter in any column

	id	timestamp	latitude	longitude	speed	up_down_count	down_up_count	total_gb	used_gb	free_gb	usage_percent	storage_full	temperature	gps_status
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1286	1286	2024-12-19 16:38:21	10.777278	106.706969	39.62	30	30	119.02	26.18	92.84	22.0	0	42.08	OK
1287	1287	2024-12-19 16:38:22	10.775321	106.706663	39.62	30	30	119.02	26.18	92.84	22.0	0	43.21	OK
1288	1288	2024-12-19 16:38:23	10.772925	106.706447	39.62	30	30	119.02	26.18	92.84	22.0	0	42.10	OK
1289	1289	2024-12-19 16:38:24	10.770824	106.706251	39.62	30	30	119.02	26.18	92.84	22.0	0	42.08	OK
1290	1290	2024-12-19 16:38:25	10.77022	106.70612	40.34	30	30	119.02	26.18	92.84	22.0	0	41.98	OK
1291	1291	2024-12-19 16:38:26	10.769574	106.705865	40.34	30	30	119.02	26.18	92.84	22.0	0	42.38	OK
1292	1292	2024-12-19 16:38:27	10.769574	106.705865	41.21	30	30	119.02	26.18	92.84	22.0	0	42.33	OK
1293	1293	2024-12-19 16:38:28	10.768332	106.705622	41.89	30	30	119.02	26.18	92.84	22.0	0	42.12	OK
1294	1294	2024-12-19 16:38:29	10.766888	106.706357	42.26	30	30	119.02	26.18	92.84	22.0	0	42.91	OK
1295	1295	2024-12-19 16:38:30	10.766095	106.706763	41.56	30	30	119.02	26.18	92.84	22.0	0	43.14	OK
1296	1296	2024-12-19 16:38:31	10.765302	106.707085	43.32	30	30	119.02	26.18	92.84	22.0	0	42.28	OK
1297	1297	2024-12-19 16:38:32	10.765088	106.706883	42.12	30	30	119.02	26.18	92.84	22.0	0	42.45	OK
1298	1298	2024-12-19 16:38:33	10.764532	106.705553	41.84	30	30	119.02	26.18	92.84	22.0	0	42.98	OK
1299	1299	2024-12-19 16:38:34	10.764474	106.705207	43.22	30	30	119.02	26.18	92.84	22.0	0	43.82	OK
1300	1300	2024-12-19 16:38:35	10.765415	106.704799	44.10	30	30	119.02	26.18	92.84	22.0	0	42.33	OK
1301	1301	2024-12-19 16:38:36	10.765995	106.704542	43.78	30	30	119.02	26.18	92.84	22.0	0	42.12	OK
1302	1302	2024-12-19 16:38:37	10.766764	106.704233	43.28	30	30	119.02	26.18	92.84	22.0	0	43.68	OK
1303	1303	2024-12-19 16:38:38	10.767497	106.703928	42.19	30	30	119.02	26.18	92.84	22.0	0	43.23	OK
1304	1304	2024-12-19 16:38:39	10.767064	106.703254	40.34	30	30	119.02	26.18	92.84	22.0	0	42.99	OK
1305	1305	2024-12-19 16:38:40	10.766477	106.702763	41.21	30	30	119.02	26.18	92.84	22.0	0	42.35	OK
1306	1306	2024-12-19 16:38:41	10.766206	106.701922	41.89	30	30	119.02	26.18	92.84	22.0	0	41.23	OK

Figure 5.14: Data received via MQTT stored in SQLite database

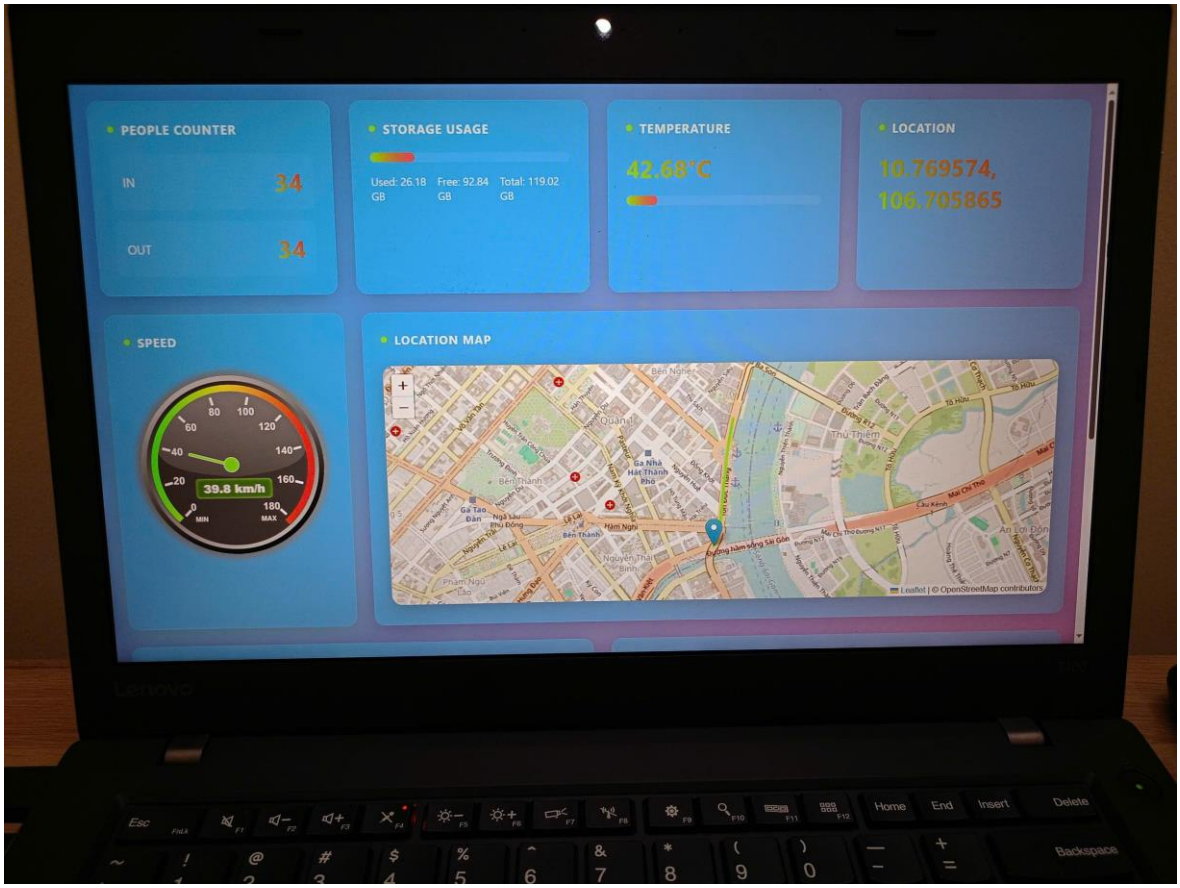


Figure 5.15: The web application displayed on the client's computer

CHAPTER 6. DATA ANALYSIS

6.1. Comments on the fine-tuned model

Figure 6.1 shows the F1 curve of the model after fine-tuning. It's essential to understand the F1 score itself. The F1 score is a metric that combines precision and recall into a single value. For Figures 6.2, 6.3, these are the P curve and R curve, and PR curve, respectively. Following that, Figure 6.4 shows the Confusion Matrix and Confusion Matrix Normalized, and Figure 6.5 displays the results of the parameters after fine-tuning across the epochs.

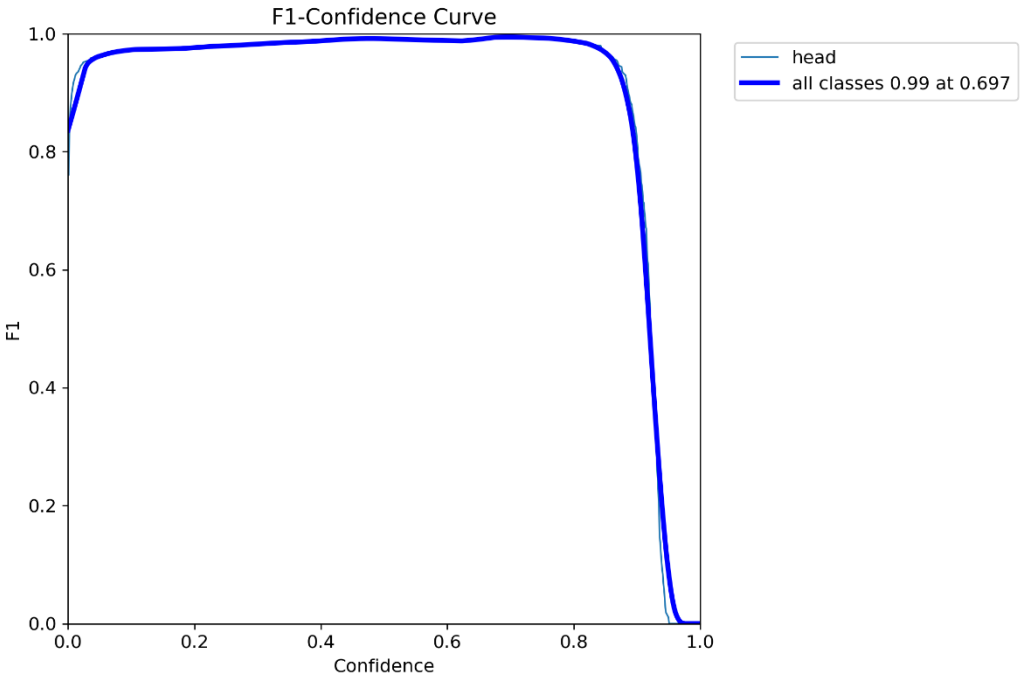


Figure 6.1: F1 curve

Comment on F1 curve: The model is performing well across a wide range of confidence thresholds. The optimal threshold of 0.697 suggests that the model is confident in its predictions. The sharp drop-off at higher thresholds indicates that the model may miss some true positives if the threshold is set too high.

Comment on P curve: The model maintains high precision, meaning that most of its predictions are correct. The optimal threshold of 0.713 suggests that the model is

very confident in its predictions. The sharp drop-off at higher thresholds indicates that the model may miss some true positives if the threshold is set too high.

Comment on R curve: The model maintains high recall, meaning that it is detecting most of the true positives. The sharp drop-off at higher thresholds indicates that the model may miss some true positives if the threshold is set too high.

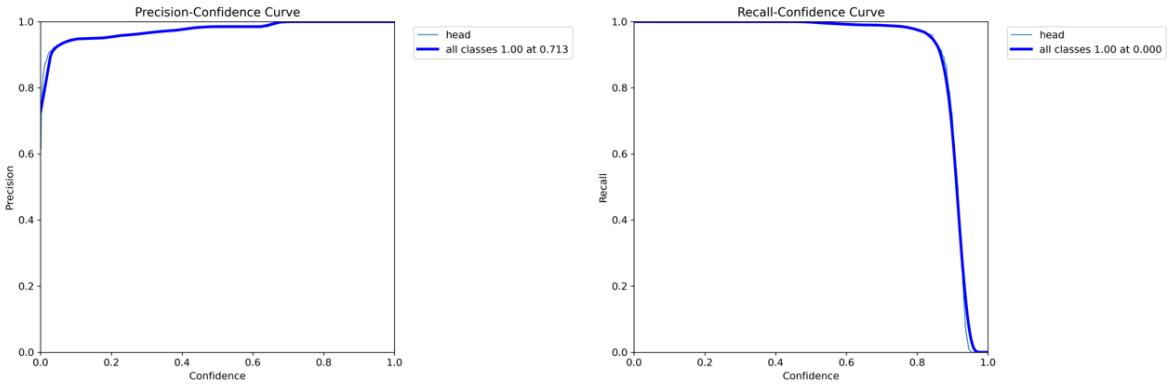


Figure 6.2: P curve and R curve

Comment on PR curve: The model is performing exceptionally well, achieving a high balance between precision and recall. The sharp drop-off at high precision indicates that the model may miss some true positives if the precision is prioritized too much.

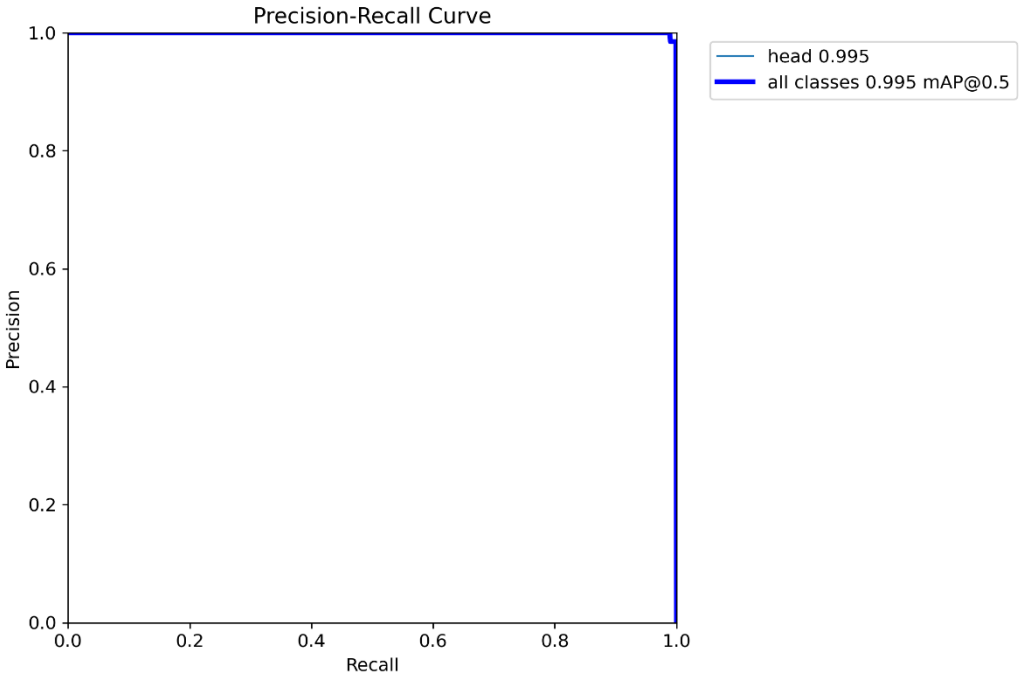


Figure 6.3: PR curve

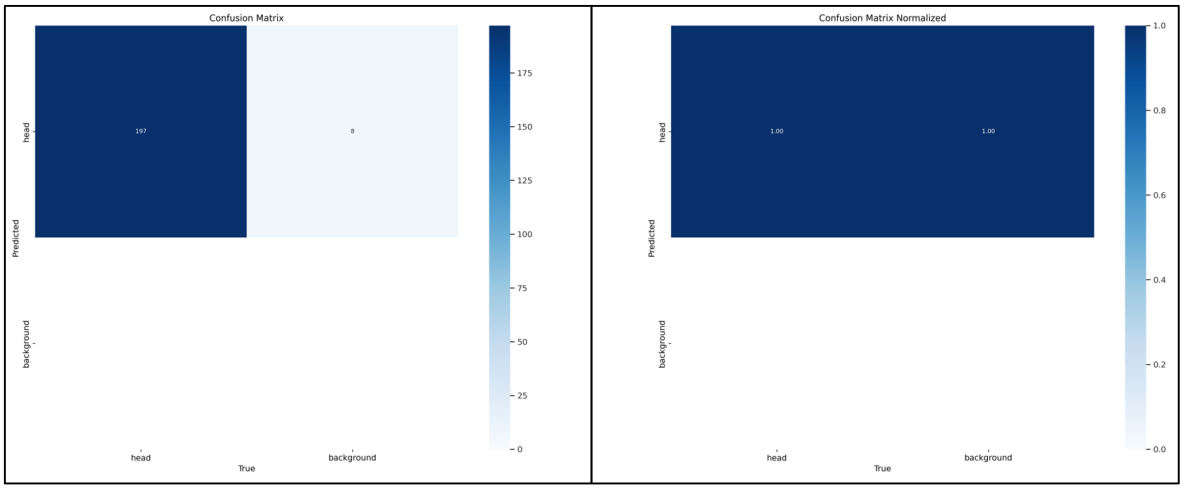


Figure 6.4: Confusion Matrix and Confusion Matrix Normalized

Comment on Confusion Matrix: The model is performing exceptionally well at detecting "head" instances with very few false positives. The model is not detecting any "background" instances, which is expected because it only trained to detect "head" and the rest is considered background. This matrix suggests that the model is highly specific to the "head" class.

Comment on Confusion Matrix Normalized: The normalized matrix confirms the findings from the previous matrix. The model is perfectly predicting "head" but is not detecting "background".

Comment on Training and Validation Loss and Metrics: The model is learning effectively, as evidenced by the decreasing loss and increasing metrics. The validation curves are close to the training curves, indicating that the model is generalizing well to unseen data.

Overall comment:

- Excellent Performance. The model is performing exceptionally well on the "head" detection task, achieving high precision, recall, and F1 scores.
- Good Generalization: The training and validation curves suggest that the model is generalizing well to unseen data.
- Optimal Threshold: The F1 curve helps identify the optimal confidence threshold for your model.

- Potential for Improvement: While the model is performing well, there may be room for further improvement, especially in detecting other classes. Specific Task: The model is highly specific to the "head" class.

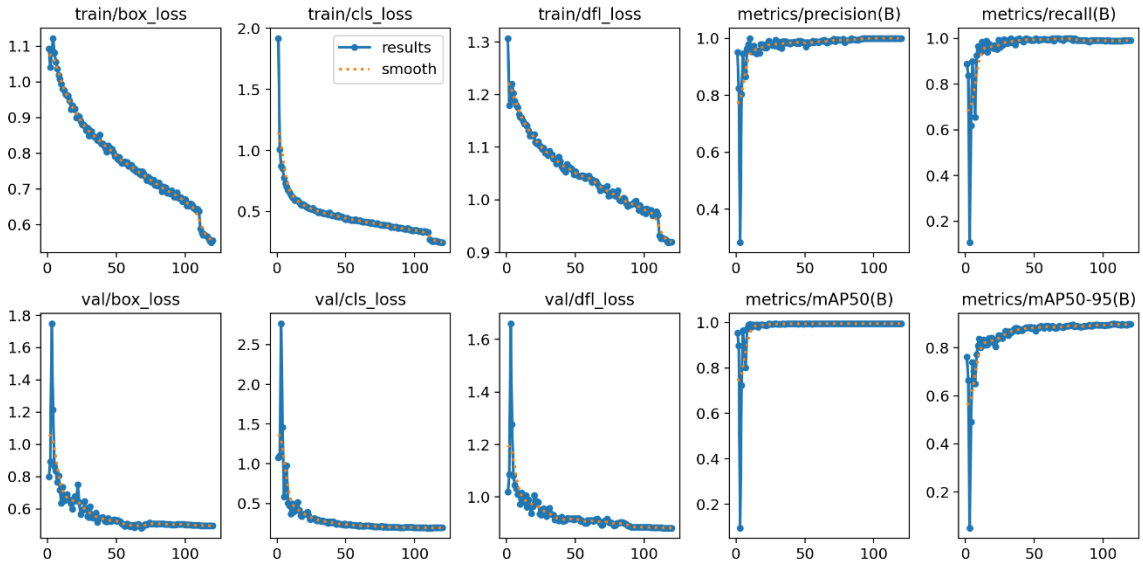


Figure 6.5: Training and Validation Loss and Metrics

During training, an IoU threshold of 0.7 will be used during the Non-Maximum Suppression (NMS) stage. It means when the object detection model predicts multiple bounding boxes that overlap for the same object, Non-Maximum Suppression (NMS) is used to select only the best box. If the IoU between two detected bounding boxes for the same object is greater than the 0.7 threshold, one of the bounding boxes will be discarded during the process. In this way, the IoU threshold helps reduce duplicate detections. Estimate IoU and accuracy after training:

- Estimating IoU:
  - mAP50: The mAP50 (mean Average Precision at IoU 0.5) gives a good indicator of the model's IoU performance. From the training and validation metrics (Figure 6.5), the mAP50 is close to 1.0. This suggests that the model is achieving a high IoU, with most bounding boxes having an IoU greater than 0.5.
  - While mAP50 gives an indication of the model's performance at a specific IoU threshold, it doesn't directly give the average IoU. However, given the

high mAP50, so it can infer that the average IoU is likely to reach almost 1.0 and mAP50-95 also showing strong performance around over 0.9.

- Estimated Average IoU: Based on the mAP50 and the shape of the precision-recall curve, it can estimate the average IoU to be over 0.9.

- Estimating Accuracy:
  - From Confusion Matrix: TP = 197, FP = 8, FN = 0, TN = 0 (since the model is not detecting "background" as a separate class).
  - Estimated Accuracy is calculated as Equation (6.1):

$$Accuracy\ (A) = \frac{tp + tn}{tp + fp + tn + fn} = \frac{197 + 0}{197 + 8 + 0 + 0} = 0.961 \quad (6.1)$$

  - Estimated Accuracy = 96.1%

6.2. Comments on the hardware

Figure 6.6: Hardware with IPX22

Figure 6.6 once again shows the system, but from a closer perspective. Hardware remark: The solid, compact box measures just 115x90x55 mm. It can be easily mounted on the vehicle's ceiling due to its lightweight of only 150 grams.

Table 6.1: The total hardware cost

Item No.	Component	Price (VND)	Price (USD)
1	MaixCAM	700000	27.552
2	GPS NEO M8N	288000	11.33568
3	GPS Antenna 1575.42 MHz SMA 3m Length	59000	2.32224
4	4G Mobile Wi-Fi	900000	35.424
5	Nidec DC fan	27000	1.06272
6	Plastic box 115x90x55 mm	11000	0.43296
7	3W Infrared light	25000	0.984
8	USB-Type C Cable with Switch	27000	1.06272
9	Energizer 20000mAh / 3.7V Li-Polymer Power Bank (UE20010)	420000	16.5312
10	Mobifone 4G SIM, 3GB/month, 1-year validity	176000	6.92736
11	PCB Board and other components	100000	3.936
TOTAL		**2733000**	**107.57088**

Additionally, it is well-sealed, preventing the insertion of any objects larger than 12.5 mm and is not affected by small water droplets falling vertically when the device is tilted at a 15-degree angle. The hardware operates for 12 hours under all conditions in the vehicle, including vibrations and temperature variations. The 4G SIM-based Wi-Fi transmitter also works well in most situations.

The total hardware cost is listed in Table 6.1, with prices in USD calculated as of December 15, 2024, based on an exchange rate of 1000 VND = $0.00003936 USD.

6.3. Comments on the MQTT protocol

The MQTT protocol works well and stably with both hardware and software, as it consistently sends complete information to the database.

6.4. Comments on the software

The web application running on Flask, integrated with an SQLite database, effectively handles the tasks of receiving, displaying, and storing information. It ensures that no data is missed or displayed incorrectly. The numbers obtained are all real-time, supporting the key element of the project.

6.5. Comments on the system's advantages

The system, after being designed and operated, has demonstrated notable advantages over traditional counting methods and products available on the market, such as:

- Real-time Monitoring: The system provides real-time monitoring of student counts and bus locations, addressing a critical societal need.
- AI-Powered Accuracy: AI-driven object detection and tracking ensure accurate and reliable counting.
- Safety Assurance: Automated counting and real-time location data reduce the risk of leaving students behind, supporting caregivers and reassuring parents.
- Low-Light Operation: The infrared light enhances functionality in low-light conditions, enabling the system to operate flexibly in various environments.
- Cost-Effective: With a total cost under $150, the system is a feasible solution for wide deployment on school buses. Integrating onboard power and Wi-Fi can save up to $52.

- Compact Design: Its compact size (115x90x55 mm), lightweight build (150 grams), and IPX22 dust and water resistance ensure easy installation without adding significant load.
- Autonomous Operation: Designed for automatic operation with minimal human intervention, supported by a 20,000mAh power bank ensuring at least 12 hours of continuous operation.
- Comprehensive Data Management: The web application provides a clear, detailed display and supports transparent data storage and analysis.
- IoT Connectivity: Remote monitoring and data access from multiple locations ensure real-time insights for school administrators and parents. The use of the MQTT protocol ensures reliable and efficient data transmitter over the internet.

6.6. Comments on the system's disadvantages

With the advantages that meet all the requirements, the system still has some drawbacks, such as:

- Environmental Constraints: The system's IPX22 rating only offers limited water resistance, making it less suitable for harsh outdoor environments.
- Power Dependency: Continuous operation relies on a charged power bank or an external power source, potentially limiting long-term autonomous use.
- Data Privacy and Security: Collecting and transmitting personal and location data may raise privacy concerns, requiring robust data security protocols.
- Connectivity Requirements: The system depends on internet connectivity for remote monitoring, making it less effective in areas with poor network coverage.

However, these are limitations that can be improved with current technology and more importantly, this project has solved the most challenging task: embedding lightweight models and algorithms into low-cost hardware, enabling widespread deployment.

CHAPTER 7. CONCLUSION

7.1. Conclusion

After conducting research, designing, testing, and optimizing, the system has successfully met the predefined requirements. Table 7.1 shows the performance and hardware specifications of the Intelschoolbus – the system for counting and controlling students on school buses.

Table 7.1: The completed product

Property	The completed product	Required Specification
Accuracy	96%	Over 95%
FPS (Frames per Second)	Over 10	Over 10
Cost	$108	Below $150
Weight	150 grams	Below 200 grams
Average operating temperature measured	Below 50°C	Below 80°C
Operating Time	12 hours continuously	Minimum 12 hours continuously
System Size	115x90x55 mm	Smaller than 200x200x100 mm
Other Standards	Industrial-grade wired connection, dustproof, and water-resistant IPX22	Industrial-grade wired connection, dustproof, and water-resistant IPX22

Thanks to the hardware, the system's software can accurately retrieve the bus's location and essential information. The software operates without interruption, ensuring continuous data storage. All data can be accessed in real time from devices outside the LAN network.

7.2. Thesis development

Intelschoolbus - the system for counting and controlling students on school buses can integrate the following potential areas:

- Advanced AI Camera:
  - High-resolution camera for clearer images.
  - Additional data collection for improved model accuracy.
- Integrated IoT Devices:
  - Microphone to detect unusual sounds or emergency alerts.
  - Using the 4G GPS module (likes Figure 7.1) directly to connect to 4G or LTE networks and utilize A-GPS technology to replace the Wi-Fi transmitter with the SIM and GPS module.
- AI Software Improvements:
  - Face recognition: Integrate facial recognition to verify students' identities.
  - Detect unauthorized individuals boarding the bus.
  - Pose estimation: Detect students' abnormal postures to identify falls or dangerous situations.
  - Behavior analysis: Identify unsafe behaviors like pushing or shoving when boarding or exiting.
- Data Management Software Enhancements:
  - Management application: Integrate a mobile app for parents to receive notifications when students board or leave the bus.
- Additional Features:
  - Emergency alerts: Send alerts to parents and the school if a student fails to get off at the correct stop.
  - Automated announcement system: Announce the number of remaining students on the bus to inform the driver when the trip is complete.
  - Statistical analysis: Generate daily reports on the number of students, boarding times, and routes, helping to improve the bus schedule.

- Security and privacy:
  - Data encryption: Apply security protocols like HTTPS and SSL to protect students' information.
  - Access control management: Grant data access permissions to teachers, parents, and the school based on appropriate levels.
- Future expansion capabilities:
  - Comprehensive IoT Integration: Connect the system to smart home or smart city ecosystems.
  - Blockchain integration: Store data on a blockchain to ensure transparency and immutability.
  - Big data analysis: Use data for optimizing routes, pickup or drop-off schedules, and operational costs.

Figure 7.1: Module 4G GPS Quectel EG800K

(Source: TDLOGY)

LIST OF PUBLISHED PAPERS BY AUTHOR

[ICGHIT 2025] Paper #1571099680 ('Intelschoolbus - the System for Counting Controlling Students on School Buses'): Accepted

REFERENCES

Chang, Y.-W., Pan, J.-W., & Hsu, Y.-W. (2020). Estimation of the number of passengers in a bus using deep learning. In *MDPI Sensors* (pp. 20-33). Basel, Switzerland: Multidisciplinary Digital Publishing Institute.

Chen, Y.-W., Li, Y.-F., & Huang, C.-S. (2020). Bus passenger counting using YOLO and DeepSORT with Kalman filter. In *IEEE Access* (pp. 17-26). Piscataway, NJ: IEEE Press.

Grönman, J., Sillberg, P., Rantanen, P., & Saari, M. (2019). People counting in a public event—Use case: Free-to-ride bus. In *Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 10-19). Rijeka, Croatia: IEEE Press.

Myrvoll, T. A., Håkegård, J. E., Matsui, T., & Septier, F. (2017). Counting public transport passengers using WiFi signatures of mobile devices. In *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1-6). Yokohama, Japan: IEEE Press.

Nguyen, D. T., Phan, M. K., Tran, P.-N., & Dang, D. N. M. (2024). Vietnamese traffic sign recognition using deep learning. In *Proceedings of the 2024 9th International Conference on Intelligent Information Technology* (pp. 30-35). New York, NY: Association for Computing Machinery.

Ramalingam, B., Le, A. V., Lin, Z., Weng, Z., Elara, R. M., & Pookkuttath, S. (2022). Optimal selective floor cleaning using deep learning algorithms and reconfigurable robot hTetro. In *Scientific Reports* (pp. 7-9). London, UK: Nature Publishing Group.

APPENDIX A. TRANSMITTER CODE

```
from maix import nn, camera, app, time, tracker, uart, image, pinmap
import pynmea2
import paho.mqtt.client as mqtt
import json
import os
import time as time2
from datetime import datetime, timedelta
import shutil
from pathlib import Path

ports = uart.list_devices()

pinmap.set_pin_function("A18", "UART1_RX")

device = "/dev/ttyS1"

# MQTT settings
broker_address = "12.179.111.186"
broker_port = 1883
topic = "data"
connected = False

# GPS settings
GPS_RETRY_INTERVAL = 8  # Retry interval for GPS in seconds
gps_serial = None  # Serial connection for GPS data
last_gps_update = 0  # Last GPS update timestamp
```

```
gps_update_interval = 2000  # Interval between GPS updates in ms
current_lat = None  # Current latitude
current_lon = None  # Current longitude
current_speed = None  # Current speed

def yolo_objs_to_tracker_objs(objs, valid_class_id):
    new = []
    for obj in objs:
        if len(valid_class_id) > 0 and obj.class_id not in valid_class_id:
            continue
        new.append(tracker.Object(obj.x, obj.y, obj.w, obj.h, obj.class_id, obj.score))
    return new

def obj_in_roi(obj, roi):
    x = obj.x + obj.w // 2
    y = obj.y + obj.h // 2
    if x >= roi[0] and x <= roi[0] + roi[2]:
        if y < roi[1]:
            return -1
        if y > roi[1] + roi[3]:
            return 1
        return 0
    return None

def count_tracks(img, count_roi, tracks, count, down_track_ids):
    img.draw_rect(count_roi[0], count_roi[1], count_roi[2], count_roi[3],
image.COLOR_YELLOW, thickness=2)
    for track in tracks:
        if track.lost:
```

```
            continue
        obj = track.history[-1]
        ret = obj_in_roi(obj, count_roi)
        if ret is None:
            continue
        if ret > 0 and track.id not in down_track_ids:
            # find if last positions in roi
            for o in track.history[::-1][1:]:
                if obj_in_roi(o, count_roi) == 0:
                    count += 1
                    down_track_ids.append(track.id)
                    break
    msg_h = image.string_size("1!aA,</?", scale=1.3).height()
    img.draw_string(0, img.height() - msg_h, f"up down: {count}", scale=1.3)
    if len(down_track_ids) > 500: # remove some history we not use
        down_track_ids = down_track_ids[300:]
    return count, down_track_ids


def main(disp):
    # configs
    conf_threshold = 0.3       # detect threshold
    iou_threshold = 0.5        # detect iou threshold
    max_lost_buff_time = 12    # the frames for keep lost tracks.
    track_thresh = 0.8         # tracking confidence threshold.
    high_thresh = 0.6          # threshold to add to new track.
    match_thresh = 0.9
    max_history_num = 25       # max tack's position history length.
    show_detect = False        # show detect
    valid_class_id = [0]
```

```
detector = nn.YOLOv8(model="/root/models/yolov8n.mud", dual_buff = True)
cam = camera.Camera(detector.input_width(), detector.input_height(), detector.input_format())
tracker0 = tracker.ByteTracker(max_lost_buff_time, track_thresh, high_thresh, match_thresh, max_history_num)

# counter
count_roi = [0, cam.height() - cam.height() // 2, cam.width(), cam.height() // 9]
up_down_count = 0
down_track_ids = []

while not app.need_exit():
    img = cam.read()
    objs = detector.detect(img, conf_th = conf_threshold, iou_th = iou_threshold)
    if show_detect:
        draw_yolo_results(img, objs, valid_class_id)
    objs = yolo_objs_to_tracker_objs(objs, valid_class_id)
    tracks = tracker0.update(objs)
    show_tracks(img, tracks)
    up_down_count, down_track_ids = count_tracks(img, count_roi, tracks, up_down_count, down_track_ids)
    img.draw_image(0, 0, img_back)
    disp.show(img)

    x, y, preesed = ts.read()
    if is_in_button(x, y, back_rect_disp):
        app.set_exit_flag(True)

disp = display.Display()
```

```
try:
    main(disp)
except Exception:
    import traceback
    msg = traceback.format_exc()
    img = image.Image(disp.width(), disp.height())
    img.draw_string(0, 0, msg, image.COLOR_WHITE)
    disp.show(img)
    while not app.need_exit():
        time.sleep_ms(100)

if __name__ == "__main__":
    try:
        main_loop()
    except KeyboardInterrupt:
        print("Program terminated by user")
    except Exception as e:
        print(f"Fatal error: {e}")
        import traceback
        print(traceback.format_exc())
```

APPENDIX B. RECEIVER CODE

```
import paho.mqtt.client as mqtt
from flask import Flask, render_template
from flask_socketio import SocketIO
import json
import threading

# MQTT settings
broker_address = "12.179.111.186"
broker_port = 1883
topic = "data"

app = Flask(__name__)
socketio = SocketIO(app)

# Store the latest data
latest_data = {}

# Define the callback function for when a connection is established
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker successfully!")
        # Subscribe to the specified topic
        client.subscribe(topic)
    else:
        print(f"Failed to connect, return code {rc}")
```

```
def on_message(client, userdata, msg):
    global latest_data
    # Decode and parse the JSON message payload
    message = json.loads(msg.payload.decode("utf-8"))
    print(f"Received message on topic {msg.topic}: {message}")
    latest_data = message
    socketio.emit('update_data', message)

mqtt_client = mqtt.Client()

# Set the callback functions
mqtt_client.on_connect = on_connect
mqtt_client.on_message = on_message

# Connect to the MQTT broker
print(f"Connecting to {broker_address}:{broker_port}")
mqtt_client.connect(broker_address, broker_port, 60)

# Start the MQTT loop in a separate thread
mqtt_thread = threading.Thread(target=mqtt_client.loop_forever)
mqtt_thread.start()

@app.route('/')
def index():
    return render_template('indexlocal.html', data=latest_data)

if __name__ == '__main__':
    socketio.run(app, debug=True, use_reloader=False)
```