AI POWERED COMPUTER VISION-BASED FRAMEWORK FOR REAL-TIME ROAD USER CLASSIFICATION AND INTELLIGENT TRAFFIC SIGNAL CONTROL

PIERRE PROMISE MOKILI

M23B23/032

JOSHUA OWOR GENO

M23B23/006

RACHEL MBEIZA ISOOBA

M23B23/047

A DISSERTATION SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE OF UGANDA CHRISTIAN UNIVERSITY

June, 2026

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

Declaration

I, **Geno Owor Joshua**, do hereby declare that this Project Report is original and has not been published and/or submitted for any other degree award to any other University before.

Date: 28/05/2026 Signature: 

I, **Mokili Pierre Promise**, do hereby declare that this Project Report is original and has not been published and/or submitted for any other degree award to any other University before.

Date: 28/05/2026 Signature: 

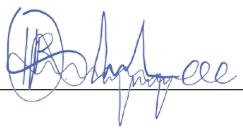
I, **Isooba Mbeiza Rachel**, do hereby declare that this Project Report is original and has not been published and/or submitted for any other degree award to any other University before.

Date: 28/05/26 Signature: 

Approval

This Project Report titled *"AI POWERED COMPUTER VISION-BASED FRAMEWORK FOR REAL-TIME ROAD USER CLASSIFICATION AND INTELLIGENT TRAFFIC SIGNAL CONTROL"* has been submitted for examination with our approval as the University Supervisors.

Eng. Ian Raymond Osolo
Department of Computing & Technology
Faculty of Engineering Design & Technology
Uganda Christian University

Signature: 

Date: 04 / 06 / 2026

Dr. Terhemba Michael-Ahile
Department of Computing & Technology
Faculty of Engineering Design & Technology
Uganda Christian University

Signature: 

Date: 02/06/2026

Dedication

We dedicate this work to our families, whose support, patience and prayers carried us through the long nights and the long deadlines. To every cyclist, boda boda rider and pedestrian on Kampala's roads, for whom the current system was not designed. And to every Ugandan life that has been lost on our roads to problems that good engineering can help solve.

"To whom much is given, much is expected."

Acknowledgments

First, we thank God for the gift of life, health and strength, and for walking with us through every stage of this project.

We are deeply grateful to our supervisor, **Dr. Innocent Ndibatya**, whose patient guidance and honest critique pushed this work from an idea into a working prototype. His insistence that every claim be backed by evidence shaped how we think as engineers, not just as students.

We acknowledge the entire team of lecturers in the Department of Computing & Technology at Uganda Christian University, who taught us the foundations we used every day on this project, from embedded systems to machine learning, web development and research methodology. We also thank the UCU Faculty of Engineering, Design and Technology for providing the laboratory space, the workstations and the general support that made this project practical.

We are grateful to the earlier UCU team behind the 2023 "Blue Light" prototype, whose work first pointed us toward the idea of a dedicated cyclist signal phase.

Lastly, we thank our families and friends for their prayers, their financial support and their constant encouragement, especially during the late nights when the scale model junction had to be rebuilt again.

To all of you, thank you.

Geno Owor Joshua
Mokili Pierre Promise
Isooba Mbeiza Rachel

Abstract

Traffic congestion is a leading cause of economic and productivity loss in Uganda. The Greater Kampala Metropolitan Area carries roughly half of the country's registered vehicles, and peak-hour speeds on central corridors drop to as low as 11 km/h. At the same time, more than 70% of road traffic fatalities in the country fall on vulnerable road users, that is pedestrians, cyclists and boda-boda motorcyclists. Most of Kampala's signalised junctions still operate on fixed-time plans that cannot respond to real demand, and none of them offers cyclists a dedicated signal phase.

This report describes RoadWise, a group final-year project that builds a low-cost, AI-powered traffic management framework for classifying road users in real time and controlling intersection signals dynamically. The system is organised in four layers: a sensing layer made of USB cameras and IoT sensors, an intelligence layer that runs a YOLOv8 detector and a Firebase cloud backend, an action layer that drives smart traffic lights, and an interface layer made of a traffic officer dashboard and a road user mobile web app. A custom scale model traffic setup was built, covering both three-way and four-way junction layouts, so that the full detection to actuation loop could be tested under controlled conditions.

The work was grounded in a PRISMA compliant systematic literature review that screened 4,419 records and retained 12 primary studies. The review confirmed that, although YOLO based vision and adaptive signal control are individually mature, no reviewed system offers a dedicated cyclist priority phase in mixed traffic. RoadWise closes that gap through what we call a blue light phase, which is activated whenever cyclist presence at a junction passes a configurable threshold.

On the prototype, the detector reached 100% vehicle detection and 85% cyclist detection accuracy under varied lighting, with a best validation mAP@0.5 of 0.977 on the miniature model dataset. The adaptive controller reduced junction waiting times by up to 40% compared with a fixed time baseline, and the web to hardware synchronisation protocol achieved 100% state consistency over serial acknowledgement feedback. The same controller was shown to run simultaneously across three-way and four-way configurations, which supports scalability toward real junctions.

The report also discusses the limits of the work, in particular the scale model-to-real

domain gap, the narrow class schema, and the remaining steps needed before a pilot deployment on a real KCCA junction can be attempted.

Keywords: Intelligent Traffic Management, Computer Vision, YOLOv8, Adaptive Signal Control, Vulnerable Road Users, Cyclist Priority, IoT, Smart City, Kampala, Uganda.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

Abbrev.	Meaning
ACK	Acknowledgement (serial protocol)
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CV	Computer Vision
DSR	Design Science Research
ESP32	Espressif Wi-Fi / Bluetooth microcontroller family
FPS	Frames Per Second
GKMA	Greater Kampala Metropolitan Area
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IoU	Intersection over Union
ITS	Intelligent Transportation System
KCCA	Kampala Capital City Authority
LED	Light Emitting Diode
mAP	mean Average Precision
MCU	Microcontroller Unit
ML	Machine Learning
NDP III / IV	Third / Fourth National Development Plan of Uganda
NMS	Non-Maximum Suppression
NodeMCU	ESP8266 based development board
NPA	National Planning Authority (Uganda)
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
PWA	Progressive Web Application
RL	Reinforcement Learning
RTI	Road Traffic Injury
RTSP	Real Time Streaming Protocol

Abbrev.	Meaning
SDG	Sustainable Development Goal
SLR	Systematic Literature Review
SSA	Sub-Saharan Africa
TJI	Traffic Jam Incident
TMS	Traffic Management System
UBOS	Uganda Bureau of Statistics
UCU	Uganda Christian University
UGX	Uganda Shilling
UNRA	Uganda National Roads Authority
UPF	Uganda Police Force
URA	Uganda Revenue Authority
VRU	Vulnerable Road User
WHO	World Health Organization
YOLO	You Only Look Once (object detection family)
YOLOv8n	YOLO version 8, nano variant

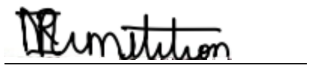
Ethical Generative AI Usage Declaration

Generative AI tools were used in this project only as a study aid and never as a replacement for our own thinking. Where we used these tools, we did so strictly for brainstorming, for summarising long papers during the literature review, and for routine code autocompletion. The detailed declaration is given in Appendix C.

We, the authors, confirm that the analytical arguments, the system design, the implementation choices and the final written content in this report are our own original work.

Geno Owor Joshua Signature: 

Mokili Pierre Promise Signature: 

Isooba Mbeiza Rachel Signature: 

Chapter 1

Introduction

1.1 Background

Urbanisation across Sub-Saharan Africa has moved faster than the ability of most cities to plan and run their road networks. The Greater Kampala Metropolitan Area, or GKMA, is a clear example. According to the Uganda Revenue Authority, Uganda has more than two million registered vehicles today, and roughly half of them operate within the GKMA [1]. At peak hours, traffic on central corridors moves as slowly as 11 km/h [2]. The Uganda National Roads Authority estimates that this congestion costs the country about UGX 3.4 billion every day, which works out to close to UGX 1.2 trillion a year [3, 4].

The human cost of this situation is even more serious than the economic one. The Uganda National Institute of Public Health reports a road traffic injury mortality rate of 29 per 100,000 people for Uganda, compared to a global average of 18 per 100,000 [5, 6]. The Uganda Police Force attributes more than 70% of road traffic fatalities to vulnerable road users, mainly pedestrians (about 34%) and motorcyclists (about 33%) [7]. Commercial motorcycles, locally known as boda bodas, account for roughly 35% of Kampala's daily traffic [8]. Across Sub-Saharan Africa as a whole, vulnerable road users make up about half of all traffic fatalities [9].

Most of Kampala's signalised junctions still run on fixed time plans that cannot respond to actual demand. A large share of peak hour jams is cleared manually by traffic police: Hamza et al. [10] report that 25.9% of observed peak time jam incidents in the city were caused directly by police intervention at signals that could not adapt on their own. No signal in the city gives cyclists or boda boda riders a dedicated priority phase, which leaves them to negotiate gaps between motorised traffic in ways the injury numbers tell us are unsafe.

On the research side, the field has moved firmly toward data driven, AI powered signal control. Real time vision models from the You Only Look Once (YOLO) family [11, 12] routinely achieve detection accuracies above 95% on urban traffic. Low cost microcon-

troller platforms such as the ESP8266, ESP32-CAM and Arduino Uno are now used as edge signal actuators [13, 14]. Reinforcement learning controllers have been shown to cut intersection waiting time by as much as 50% [15]. What is missing is a system that brings these components together for an African city, that takes the operational realities of Kampala seriously, and that puts vulnerable road users at the centre rather than at the edge.

RoadWise is our response to this gap. It is a low cost, AI powered traffic management framework that classifies road users in real time using a YOLOv8 detector, controls signal phases dynamically based on demand, and introduces a dedicated *blue light* phase that gives cyclists priority at the junction whenever their numbers cross a threshold. The system also exposes its state to traffic officers through a web dashboard and to the public through a road user app. The prototype is built around a physical scale model junction (Figure 1.1) on which both three-way and four-way configurations can be tested.

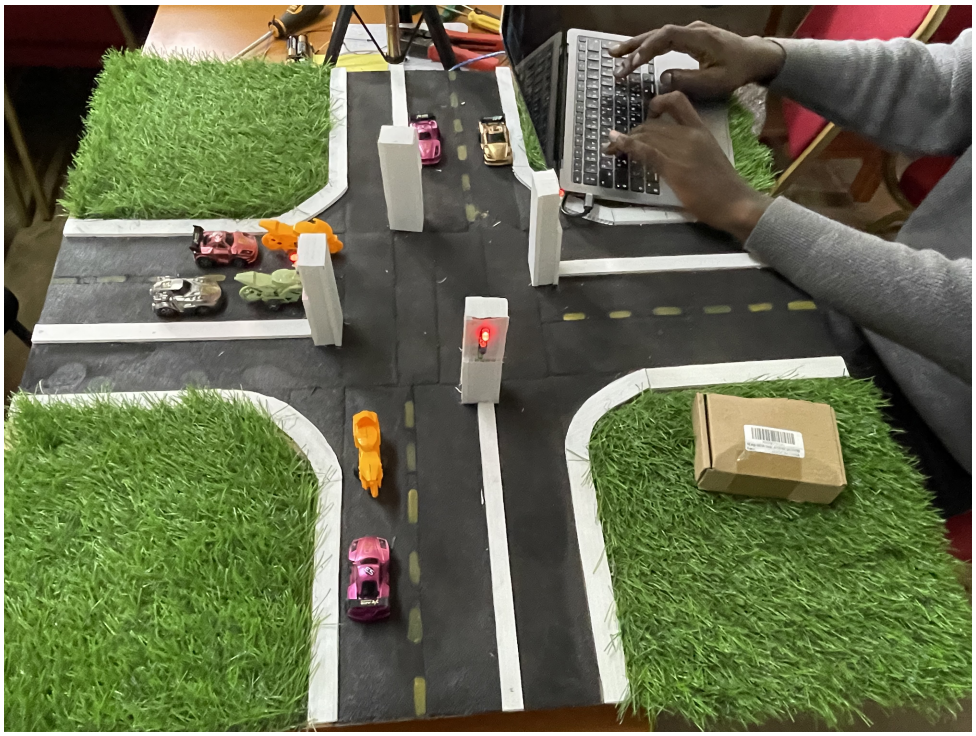


Figure 1.1: The RoadWise scale model junction prototype in operation. A YOLOv8 model running on the laptop detects the miniature vehicle and cyclist models and drives the physical signal lights in real time.

1.2 Problem Statement

Kampala's traffic management infrastructure is, at the same time, inefficient, unsafe for vulnerable road users, and blind to its own data. It is inefficient because fixed time signals cannot follow demand, and because one in four peak hour jams is resolved manually by a traffic officer rather than by the signal itself [10]. It is unsafe for vulnerable road users because they suffer over 70% of fatalities while no part of the signalling system is designed

for them [7]. And it is data blind because there is no operational system that captures per class, per junction counts that could feed either real time control or longer term planning. The overall cost of this situation is of the order of UGX 1.2 trillion a year [3].

There is therefore a real need for a locally designed, low cost, AI and IoT enabled traffic management system that can: detect and classify mixed road users in real time, adjust signal phases dynamically, give cyclists an explicit priority phase, and expose its state and controls to both officers and the public.

1.3 Research Objectives

1.3.1 General Objective

To design, develop and evaluate a low cost AI powered traffic management framework that uses real time computer vision, adaptive signal control and a dedicated cyclist priority phase to improve traffic flow and road safety at urban intersections in Kampala.

1.3.2 Specific Objectives

Specific Objective 1

To develop a YOLOv8 based computer vision subsystem that detects and classifies road users (car, bike) in real time at an accuracy of at least 95% mAP@0.5 on a representative validation set, within the first four months of the project.

Specific Objective 2

To design and implement an adaptive traffic signal control algorithm that uses the per class counts from Objective 1 to adjust green phase timing, and to add a dedicated blue light phase that is triggered when cyclist presence crosses a configurable threshold.

Specific Objective 3

To build and validate a web based officer dashboard and a road user information app that together expose the live state of the adaptive controller, allow manual override, log incidents, and serve signal and congestion information to the public under simulated Kampala conditions.

1.4 Justification

The justification for this work is economic, humanitarian, methodological and strategic at once.

Economic. Kampala's road congestion costs the country close to UGX 1.2 trillion a year [3]. Even a 10% improvement in peak hour throughput would translate into a saving of the order of UGX 120 billion a year, several orders of magnitude larger than the UGX 792,000 unit hardware cost reported in Appendix A.

Humanitarian. With vulnerable road users making up over 70% of road traffic fatalities [7], any signalling change that lowers their exposure at junctions has direct life saving value. Jacobsen [16] further argues through the "safety in numbers" hypothesis that giving cyclists dedicated infrastructure increases the number of cyclists, which in turn lowers per cyclist risk.

Methodological. The RoadWise design pulls together components that the reviewed literature has validated on their own, namely YOLO based vision, IoT based signal actuation and adaptive phase control, and adds the cyclist priority phase that is missing from all of them.

Strategic. The work aligns with Uganda Vision 2040, the National Development Plan and the UN Sustainable Development Goals 3, 9 and 11 [17, 18, 19]. Appendix F discusses this alignment in detail.

1.5 Scope

1.5.1 Geographical Scope

The system is designed for intersections of the Greater Kampala Metropolitan Area. For this report, validation was done on a physical scale model of a three-way and a four-way junction (Figure 3.2) rather than on a live GKMA junction.

1.5.2 Subject Scope

Our technical scope covers computer vision detection with YOLOv8, adaptive signal control with a dedicated cyclist priority phase, an IoT actuation layer built on Arduino Uno, NodeMCU and ESP32-CAM modules, a React officer dashboard, a mobile web app for road users, and the end to end integration of all of these. Out of scope are vehicle to infrastructure radio protocols, automated number plate recognition and any ticketing or enforcement function.

1.5.3 Time Scope

The project was carried out over a 24 week development plan between September 2025 and February 2026, followed by an evaluation and reporting phase between March and April 2026. The full schedule is in Appendix B.

1.6 Report Structure

The rest of this report is organised as follows. Chapter 2 presents a PRISMA compliant systematic literature review of intelligent traffic management systems, with an emphasis on how the current literature handles vulnerable road users. Chapter 3 sets out the research and engineering methodology we followed, the four layer system architecture, the scale model junction setup and the training pipeline. Chapter 4 reports on the experimental results, from dataset quality through training runs to the adaptive controller behaviour, and critically discusses those results against the reviewed literature. Chapter 5 presents the formal evaluation of the prototype against each specific objective, supported by annotated training artefacts and a summary evaluation table. Chapter 6 closes with the conclusions, recommendations and a roadmap toward a live junction pilot.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents the systematic literature review (SLR) that grounded the RoadWise project. The goal of the review was twofold. First, we wanted to understand the current state of the art in computer vision based traffic signal control and in the handling of vulnerable road users. Second, we wanted to locate the specific research gap that our blue light cyclist priority phase is designed to fill.

The review was carried out following the PRISMA protocol [20], which provides a structured way to identify, screen and select studies so that the evidence base is transparent and repeatable.

2.2 Review Methodology

2.2.1 Research Questions

The review was guided by three research questions:

RQ1. Which computer vision models are currently used for real time detection of road users at urban intersections, and how accurate are they?

RQ2. How do contemporary adaptive signal control systems integrate vision based data with IoT actuation, and how much do they improve on fixed time signalling?

RQ3. To what extent do existing systems accommodate vulnerable road users, in particular cyclists and motorcyclists, through dedicated signalling phases or priority logic?

2.2.2 Search Strategy

We searched Scopus and Web of Science using the Boolean strings listed in Appendix E. Typical strings combined terms such as *"traffic management system" AND "YOLO"*, or *"adaptive signal" AND ("cyclist" OR "motorcyclist") AND "computer vision"*. We restricted the search to peer reviewed journal articles and conference papers published between January 2019 and December 2025.

2.2.3 Selection Process

The PRISMA selection flow is summarised in Table 2.1. Of the 4,419 records initially identified, 1,753 survived duplicate removal and source type filtering, 349 survived subject area filtering, 77 passed the title and abstract screen, 39 were read in full, and 12 were retained as primary studies for the review.

Table 2.1: PRISMA selection stages for the RoadWise SLR

Stage	**Records**
Identification (Scopus + Web of Science)	4,419
Duplicate removal and source type filter	1,753
Subject area filter (engineering, computer science)	349
Title and abstract screening	77
Full text assessment	39
Final included studies	**12**

2.3 Thematic Synthesis

2.3.1 Object Detection for Urban Traffic

Nine of the twelve included studies use some variant of the YOLO family. Redmon and Farhadi [11] provide the theoretical anchor through YOLOv3. Pudaruth et al. [12] report 96.1% counting accuracy on urban vehicles and motorcycles using YOLOv8. Mane et al. [21] report an mAP of 96.1% for accident recognition on YOLOv8 with OpenCV. García-Pajuelo et al. [22] benchmark several YOLO generations on a bicycle path dataset and show that newer variants dominate on small two wheelers but degrade sharply under occlusion and poor lighting. The consistent takeaway is that YOLOv8 is currently the dominant choice for real time traffic vision at the edge.

2.3.2 Adaptive Signal Control

Seven of the twelve studies connect the vision stage to a dynamic signal controller. Dodia et al. [13] present the EVATL framework, in which a YOLO detector triggers green phase pre-emption for emergency vehicles through an IoT adaptive signal stack. Zerroug et al. [23] present the ADSTLS system and report a 33.82% reduction in queue occupation

when emergency vehicle priority is active. Hurtado-Gómez et al. [15] apply reinforcement learning to signal scheduling and report roughly a 50% reduction in waiting and lost time. Dave et al. [24] combine YOLOv4 with XGBoost and report up to 32.3% average waiting time reduction. Kunekar et al. [14] pair YOLO density estimation with an embedded controller that recomputes green times in real time. Webster style fixed cycle plans remain the baseline that all of these beat.

2.3.3 Vulnerable Road Users and Priority Signalling

Only three of the twelve studies focus explicitly on vulnerable road users. Kayisu et al. [25] offer a qualitative system dynamics study of VRU safety in Kinshasa, and identify enforcement culture, hand signal coexistence and road user literacy as first order drivers. Bhavsar et al. [26] analyse UAV footage of an Indian urban roundabout and record a 23.26% rule violation rate, noting that VRUs are disproportionately victimised when motorised traffic breaks rules. Ntramah et al. [9] confirm the continental picture: roughly half of all traffic fatalities in Sub-Saharan Africa fall on vulnerable road users. Crucially, **none** of the twelve studies implements a dedicated cyclist priority signal phase in mixed traffic. Where priority logic exists at all, it is reserved for emergency vehicles.

2.3.4 Classic Foundations

The review also draws on three foundational works outside the 12 primary studies. Jacobsen [16] argues the "safety in numbers" hypothesis. Pucher and Buehler [27] present European evidence that dedicated cycling infrastructure increases both cycling rates and cyclist safety. Ghadi and Toral [28] focus on adaptive control in developing country contexts. Webster's [29] classical delay minimisation theory remains the theoretical benchmark against which adaptive gains of 20–40% are measured.

2.4 Quantitative Synthesis

Table 2.2 summarises which architectural components appear in how many of the 12 primary studies.

Table 2.2: Prevalence of key architectural components across the 12 primary studies

Component	**Studies**	**%**
YOLO based vision (any version)	9/12	75
Explicit bicycle or motorcycle class	4/12	33
IoT MCU actuation (Arduino, ESP)	6/12	50
Single board computer (Raspberry Pi)	3/12	25
Adaptive or density based signal control	7/12	58
Emergency or priority strategy (any form)	3/12	25
VRU specific safety focus	3/12	25
Centralised monitoring interface	4/12	33
Cyclist priority signal phase	**0/12**	**0**

2.5 Two Case Studies of Existing Systems

Two systems in the reviewed literature deserve a closer look because they are the closest existing analogues to parts of the RoadWise design. We describe each in its own paragraph so that their architectural choices, and the limits we inherit by following them, are explicit.

2.5.1 EVATL, an Emergency Vehicle Priority System

EVATL is a system that gives priority to emergency vehicles at signalised intersections [13]. It uses a YOLO based detector on a camera feed to identify emergency vehicles, such as ambulances, in real time. When an emergency vehicle is detected, the system sends a signal over IoT to a microcontroller, typically an Arduino or an ESP32, which then flips the traffic lights to green on the approach of that vehicle so that it can pass without stopping. The pipeline is compact: camera input goes to the YOLO detector, the detection is converted into a decision by a simple priority rule, and the decision is pushed to the signal actuator. Wireless communication modules keep the loop short enough for the pre-emption to appear effectively immediate, and the need for a traffic officer to clear the way manually is removed. The limit of the system, for our purposes, is that the priority rule is reserved for emergency vehicles. Cyclists, pedestrians and boda bodas, who together dominate Kampala's VRU fatality numbers, are not modelled at all. RoadWise borrows the detect, decide, actuate pipeline from EVATL and adds the VRU priority logic that EVATL never intended to carry.

2.5.2 The Adaptive Smart Traffic Light System

The Adaptive Smart Traffic Light System [23] changes signal timings based on real time traffic conditions rather than on a fixed cycle. It uses cameras or sensors to measure traffic density on each lane, counts the vehicles waiting or estimates the queue length, and then computes how long each light should stay green. A lane with more vehicles gets a longer

green time, which reduces congestion compared with a fixed time plan. The control logic is a loop. The system collects traffic data, analyses density, adjusts signal timing and then repeats. Some variants of the system also give priority to emergency vehicles, in the same way EVATL does. The core focus, however, is vehicle throughput. Cyclists, pedestrians and boda bodas are not detected and are not prioritised, which means the system improves efficiency but does not move the needle on VRU safety. RoadWise takes the density driven scheduling core from this family of systems and extends it with the blue light cyclist priority phase that the family has never included.

2.6 Mapping the Literature onto the Specific Objectives

To show how the evidence base maps directly onto this project's objectives, Table 2.3 lists, for each specific objective of RoadWise, the studies from the 12 primary sources that supply the strongest supporting evidence and the residual gap that the RoadWise contribution has to cover itself.

Table 2.3: Mapping of the specific objectives onto the reviewed literature and the remaining gap

Specific Objective	**Supporting studies**	**Residual gap for RoadWise**
SO1. Real time detection of mixed road users at 95% mAP@0.5.	Pudaruth et al. [12] (96.1% on urban vehicles, YOLOv8), Mane et al. [21], García-Pajuelo et al. [22] (multi-YOLO bicycle path benchmark).	None of the reviewed datasets is Kampala specific. A local dataset is still missing.
SO2. Adaptive signal control with a dedicated cyclist priority phase.	Dodia et al. [13] (EVATL), Zerroug et al. [23] (33.82% queue reduction), Hurtado-Gómez et al. [15] (≈50% waiting time cut), Dave et al. [24] (up to 32.3% waiting time cut), Kunekar et al. [14].	Zero of twelve studies implement a dedicated cyclist priority phase. The blue light phase is ours.
SO3. Officer dashboard and road user application.	Kunekar et al. [14] (MCU + web monitoring), Dodia et al. [13] (IoT actuation over web).	Existing dashboards in the literature are officer only. A public facing road user application with live alerts and route planning is absent.

The pattern is consistent: the components of RoadWise are individually well supported by the reviewed studies, but their consolidation into a single VRU centred system is new. That is the gap Chapter 3 builds against.

2.7 Identified Research Gaps

Five clear gaps come out of this synthesis, and they motivate the RoadWise design:

G1. **No Sub-Saharan African end to end deployments.** None of the included studies is from a Sub-Saharan city. Their datasets and architectures implicitly assume non-African road conditions.

G2. **No dedicated cyclist priority signal phase.** Priority logic in the literature is reserved for ambulances, fire trucks and police. Cyclists, who are over-represented in fatalities, are treated as default road users.

G3. **Behavioural modelling of non-compliance is missing.** The 23.26% violation rate reported by [26] is representative, yet the controllers in the literature assume compliant road users.

G4. **Safety outcomes are rarely measured directly.** Studies tend to infer safety indirectly, from flow metrics like delay or queue length, rather than from crash or near miss data.

G5. **Life cycle cost, maintainability and power resilience are understudied.** This matters in Kampala, where grid instability and the cost of spares are real operational constraints.

2.8 Conceptual Framework

The conceptual framework that RoadWise adopts (fully developed in Chapter 3) combines the validated components from the literature, specifically YOLOv8 vision for RQ1, density driven adaptive signalling for RQ2, IoT actuated RGB and blue LEDs for RQ3, with the cyclist priority phase that the synthesis above shows is absent from the prior art. In that way the framework directly addresses gaps G1, G2 and, through its low cost Arduino and ESP based actuation, G5.

Chapter 3

Methodology

3.1 Introduction

This chapter describes how we carried out the project. It covers both the research methodology, which is anchored on the Design Science Research paradigm and the PRISMA based literature review of Chapter 2, and the engineering methodology, which followed an Agile two week sprint cycle. It then sets out the four layer system architecture, the scale model traffic simulation, the dataset, the YOLOv8 training configuration, the adaptive signal control algorithm with the blue light phase, and the evaluation protocol we used.

3.2 Research Methodology

The research side of the project follows the six steps of Design Science Research [30]. These are: problem identification, objective definition, design and development, demonstration, evaluation, and communication. The first two steps are covered by Chapters 1 and 2. The rest of the report corresponds to steps three through six.

The literature component was a PRISMA compliant systematic review. The full flow is reported in Chapter 2 and the search strings are listed in Appendix E.

3.3 Development Methodology

We used Agile software engineering with a two week sprint cadence. Each sprint ended with a working demo, a peer review of new code in Git, and a supervisor review. We also used staged testing, moving from unit tests through integration tests into user acceptance and performance testing. The development was organised into five phases:

1. Requirements gathering and hardware prototyping, covering sprints 1 to 3.
2. Core algorithm development, covering detection, tracking and adaptive control, in

sprints 4 to 7.

3. Interface development, covering the officer dashboard and the road user application, in sprints 8 to 10.
4. System integration and testing, in sprints 11 and 12.
5. Validation, evaluation and reporting, post sprint.

3.3.1 Team Roles

The team split the work roughly by strength, but every member was involved in every layer during integration. Table 3.1 shows the primary responsibility of each member.

Table 3.1: Primary team responsibilities

Member	**Primary responsibility**
Owor Geno Joshua (M23B23/006)	Project lead, backend and system architecture, hardware to software integration.
Mokili Pierre Promise (M23B23/032)	Computer vision, YOLO dataset and training, adaptive controller logic, SLR.
Mbeiza Rachel (M23B23/047)	Officer dashboard, road user web app, UI/UX and user acceptance testing.

3.4 System Architecture

The RoadWise system is organised in four layers, as shown in Figure 3.1.

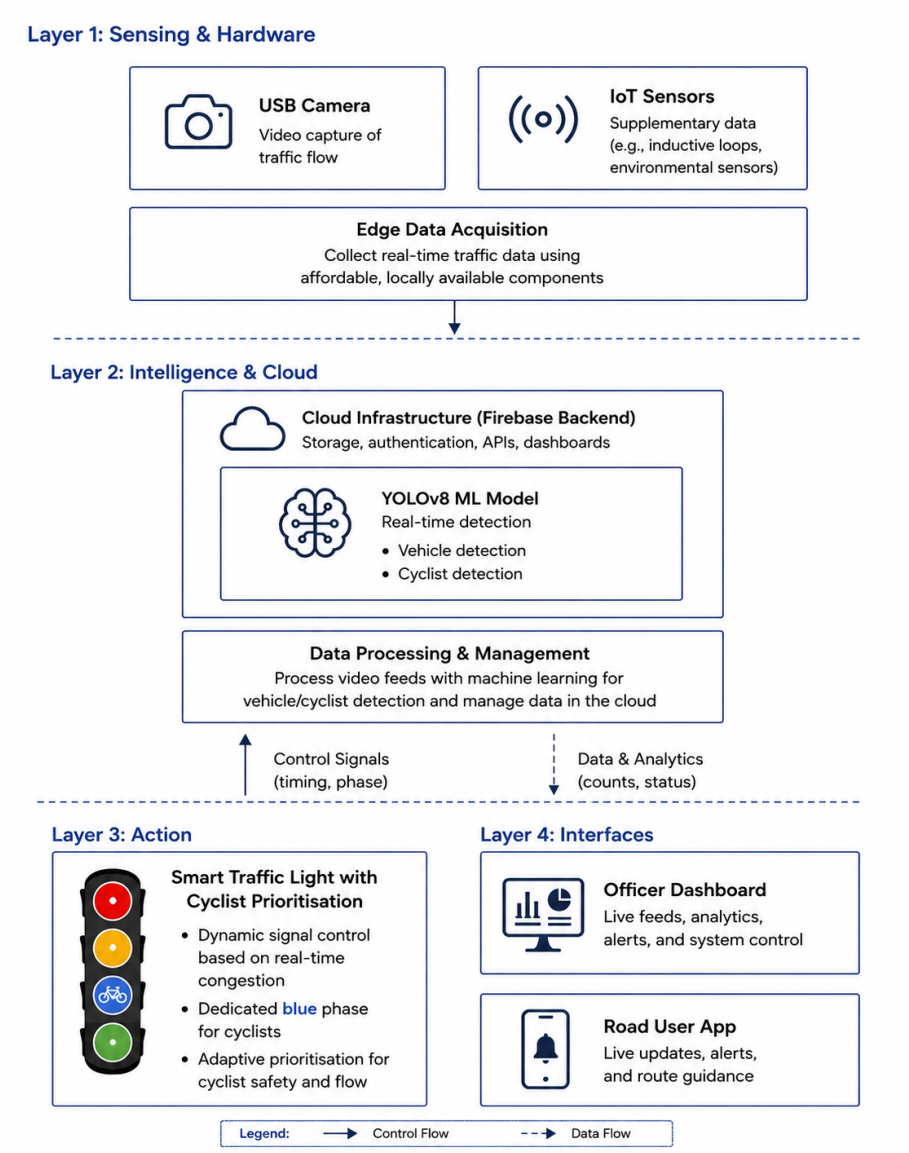


Figure 3.1: The four layer RoadWise system architecture.

Layer 1: Sensing and Hardware. Logitech USB cameras and IoT sensor boards collect real time traffic data using affordable, locally available components. The sensor side is built on Arduino Uno, NodeMCU and ESP32-CAM modules. The signal side is built on standard RGB LEDs for the red, amber and green phases, plus a dedicated blue LED array for the cyclist priority phase.

Layer 2: Intelligence and Cloud. The video feed from Layer 1 is processed by a YOLOv8 model running on an edge workstation. Detections are published to a Node.js backend that syncs state into a Firebase Realtime Database. This is where the per class counts, the current signal state and any officer commands live.

Layer 3: Action. The controller reads the counts from Firebase and turns them into green time decisions, which are pushed back over serial to the Arduino and NodeMCU boards that drive the smart traffic light.

Layer 4: Interfaces. A React officer dashboard exposes live video, detection overlays, signal state, incident logs and a manual override channel. A mobile responsive road user web app gives the public live updates, route planning, travel alerts and a community car pooling module. Both portals are shown in full in Section 4.6 of Chapter 4.

The end to end flow is:

Camera ⟶ YOLOv8 ⟶ per class counts ⟶ adaptive controller ⟶ Arduino / LEDs ⟶ Firebase

3.4.1 Hardware Stack

The hardware components used in the prototype are listed in Table 3.2.

Table 3.2: Hardware components used in the prototype

Component	**Role**	**Quantity**
Logitech C920 USB camera	Video capture at 640×480	2
ESP32-CAM module	Alternative low cost vision sensor	1
Arduino Uno	Signal timing and LED driver	1
NodeMCU ESP8266	Wi-Fi bridge to backend	1
RGB LED arrays (red, amber, green)	Conventional phases	2 sets
Blue LED array	Cyclist priority phase	2 sets
Edge compute node (laptop)	YOLO inference and backend	1

3.4.2 Software Stack

The software tools and languages used are listed in Table 3.3.

Table 3.3: Software and languages used in the prototype

Layer	**Technology**
AI and computer vision	Python 3.10, Ultralytics YOLOv8, PyTorch, OpenCV
Dataset tooling	LabelImg, custom `prepare_dataset.py`
Backend	Node.js with Express, Firebase Realtime Database, REST
Officer dashboard	React with Vite, TailwindCSS
Road user web app	React, mobile responsive, PWA capable
Edge firmware	Arduino C++ on Arduino Uno and NodeMCU ESP8266
Testing	Pytest for Python, Jest for JavaScript
Languages used	C++, Python, JavaScript

3.5 Scale Model Traffic Simulation

Because a live Kampala junction deployment was not realistic within one undergraduate project cycle, and because the unit budget (Appendix A) of about UGX 792,000 allowed only a single instrumented junction, we decided to validate the prototype on a physical

scale model traffic setup. This setup covers both a three-way and a four-way junction layout (Figure 3.2). Miniature vehicle and cyclist models stand in for real road users.

(a) Three-way junction *(b) Four-way junction*

Figure 3.2: The two physical junction layouts used in the scale model traffic simulation.

This choice trades some external validity for reproducibility and cost. It lets us run the full detection to actuation loop, under controlled lighting and known traffic density, at a level of repeatability we could not have reached on a real road. Chapter 4 discusses the scale model-to-real domain gap in detail.

3.6 Dataset

3.6.1 Acquisition

Two separate image pools were built, one for the fine-tuned YOLOv8n detector and one for the Custom CNN classifier, both assembled during Phase 3 of the project (Weeks 8 to 13).

Fine-tuned YOLOv8n dataset (1,286 images). We constructed a physical mini-city board representing both three-way and four-way junction layouts, then placed miniature car and motorcycle models on the board in varied arrangements to simulate real traffic queues. We captured 1,286 photographs of these arrangements at our project workspace, using an A9 camera and an ESP32-CAM, the same hardware the deployed system uses for live inference. Images were taken across a range of lighting conditions, camera angles and vehicle placements to introduce variety into the training set. Photographing our own scale model board, rather than sourcing a generic traffic dataset, was a deliberate choice: it ensured that the training distribution reflected the exact visual domain in which the fine-tuned detector would operate, namely the same miniature models, the same road board geometry and the same camera perspective. Images were imported into the dataset root using `scripts/import_images.py`, which handles duplicates by skip or rename.

Custom CNN dataset (15,075 images). The CNN binary classifier required a larger and more varied pool to avoid overfitting on the narrow scale model distribution. A

subset of this pool came from our own scale model captures and video frames sampled at every 30th frame to reduce redundancy. The remainder was drawn from publicly available vehicle and cyclist image datasets on Kaggle. The Kaggle supplement provided additional visual variety (different lighting conditions, angles and vehicle types) so that the CNN learned a robust vehicle-versus-cyclist boundary rather than memorising the appearance of specific miniature models.

3.6.2 Annotation

The images were annotated in YOLO format using LabelImg. Every image has a matching `.txt` file, where each line has the class index, the normalised centre coordinates and the normalised width and height of one bounding box. We used two classes: index 0 for `bike` and index 1 for `car`. The class mapping is stored in `dataset/classes.txt` and `dataset/data.yaml`.

3.6.3 Train and Validation Split

The script `scripts/prepare_dataset.py` performs a deterministic 70/15/15 split with a fixed random seed of 42. This gives us 900 training images, 193 validation images and 193 test images (Table 3.4).

Table 3.4: Dataset statistics

Split	**Images**	**Share**
Train	900	70.0%
Validation	193	15.0%
Test	193	15.0%
Total	**1,286**	**100%**

3.7 Model Training Configuration

The detector was initialised from the `yolov8n.pt` pretrained checkpoint (6.5 MB) and fine tuned using `train.py`. The key hyper-parameters are reported in Table 3.5.

Table 3.5: Training hyper-parameters

Parameter	**Value**
Base model	YOLOv8n (nano)
Image size	640
Batch size	16
Epochs (target)	120, with early stop patience 30
Optimiser	SGD (Ultralytics default)
Learning rate schedule	Cosine, optional flag
Device preference order	Apple MPS then CUDA then CPU
Cache	RAM when available

3.8 Computer Vision Pipeline

The computer vision subsystem evolved through two distinct phases during development. We initially worked with Ultralytics YOLOv8 alone, using its pretrained COCO weights to count vehicles and cyclists directly from the COCO class names. We found this sufficient for testing the control logic but not for our custom miniature model context, so we extended the pipeline in two directions: we manually annotated our own dataset using LabelImg and fine-tuned a YOLOv8n model on it, and we also trained a custom convolutional neural network (CNN) as a parallel classifier. The final system therefore uses three complementary detection mechanisms, all integrated through a Flask REST API that feeds per-class counts to the adaptive controller. Figure 3.3 shows the overall pipeline.

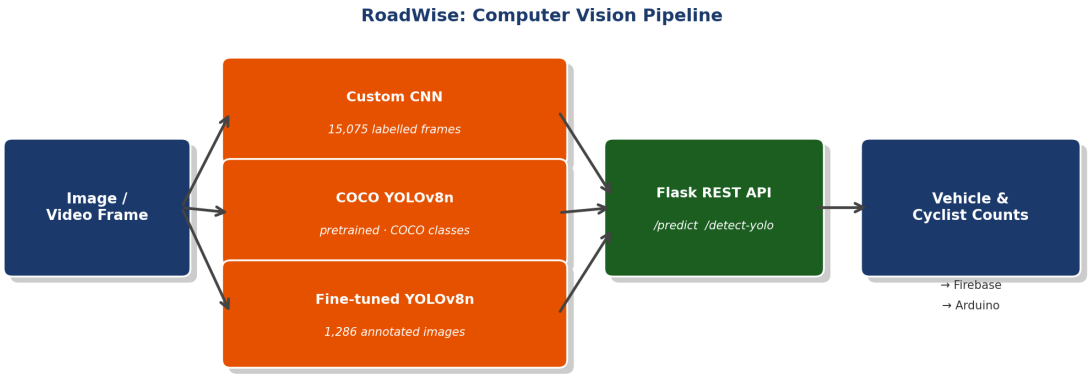


Figure 3.3: Computer vision pipeline: one image or video frame is processed by three detection approaches (Custom CNN, COCO pretrained YOLOv8n, and Fine-tuned YOLOv8n), all served through a Flask REST API that returns per-class vehicle and cyclist counts to the adaptive controller.

3.8.1 Dataset Annotation: LabelImg and COCO Auto-labelling

We used two complementary annotation strategies for our 1,286-image scale model junction dataset. The first was manual annotation with LabelImg [31], an open-source graphical tool that lets the annotator draw bounding boxes on each image and assign class labels. Each annotation is saved in YOLO format: a text file with one line per box, giving the class index, the normalised centre x and y coordinates, and the normalised width and height. We used two classes: index 0 for `bike` and index 1 for `car`.

Manual annotation alone was time consuming for a large set of images. We therefore used the pretrained COCO YOLOv8n model to automatically generate candidate annotations for images where the objects were clear enough for the pretrained detector to find them. Because YOLOv8n was trained on the COCO dataset [32], it already knows how to detect cars, trucks, buses, bicycles and motorcycles. We ran the pretrained model over a batch of our images, mapped its COCO predictions onto our two-class schema (cars, trucks and buses mapped to `car`; bicycles and motorcycles mapped to `bike`), and saved the

resulting bounding boxes as YOLO annotation files. These auto-generated annotations were reviewed and corrected before being included in the training set.

3.8.2 Fine-tuned YOLOv8n: Scale Model Junction Detector

Once the 1,286-image dataset was annotated, we fine-tuned the `yolov8n.pt` pretrained checkpoint on it using `train.py`. YOLOv8n is the nano variant of the YOLOv8 family [12], chosen for its low parameter count and fast inference on the edge workstation. Fine-tuning adapted the pretrained COCO weights to our two-class scale model domain. The best run (Run 5, 30 epochs on CUDA) reached mAP@0.5 of 0.977 on the 193-image validation split.

For real-time detection, we loaded this fine-tuned checkpoint into the Ultralytics `YOLO` class and called `model(frame)` on each BGR frame from OpenCV `VideoCapture`. The result object lists each detected box with its pixel coordinates, class index (0 = `bike`, 1 = `car`), and confidence score. Our `run_detect.py` draws each box on the live frame (orange for `bike`, magenta for `car`) and counts detections per class. End-to-end latency on a 2020-era laptop with Apple MPS was 45 ms to 80 ms per frame, giving 12 to 22 FPS.

3.8.3 COCO Pretrained YOLOv8n: Real-World Counting

In parallel, we kept the original pretrained COCO YOLOv8n model for broader, real-world detection. This model requires no fine-tuning because COCO already includes the road user categories we need. We defined two class sets:

- **Vehicle classes**: `car`, `truck`, `bus`
- **Cyclist classes**: `bicycle`, `motorcycle`

The `count.ipynb` notebook demonstrates this approach: running the COCO model on a single image of real traffic (a cars.png test image) correctly identified 35 cars and 3 trucks (38 vehicles total) in one inference call. This approach is exposed through the Flask API at the `/detect-yolo` endpoint, which accepts a POST request with an uploaded image and returns `vehicles` and `cyclists` counts, along with the full list of detected class names for debugging.

3.8.4 Custom CNN Classifier

We also trained a custom binary CNN using PyTorch as an independent classifier to verify detections. The architecture has two convolutional stages: the first applies 16 filters of size 3×3 followed by 2×2 max-pooling, the second applies 32 filters of the same size followed by another max-pooling stage. The flattened feature map is then passed through two fully connected layers (64 units then 1 unit) with a sigmoid output, giving a probability that the input image contains a cyclist rather than a vehicle.

The model was trained during Phase 3 (Weeks 8 to 13) on 15,075 images and video frames drawn from two sources. The first source was our own scale model captures taken at the project workspace using the A9 camera and ESP32-CAM; video recordings of the board were sampled at every 30th frame to reduce redundancy between consecutive frames. The second source was publicly available vehicle and cyclist image collections from Kaggle, included to widen the visual distribution beyond the narrow scale model domain and reduce the risk of overfitting. The resulting pool contained 1,401 vehicle samples and 13,674 cyclist samples. This dataset exhibits a significant class imbalance, with cyclist samples being approximately ten times more frequent than vehicle samples. Such imbalance may bias the model toward predicting the dominant class, potentially affecting performance on the minority class (vehicles).

All images were resized to 224×224 pixels before training. An 80/20 train-test split, a batch size of 32 and the Adam optimiser were used for 10 epochs. The trained weights are saved as `vehicle_cyclist_classifier.pth` and loaded by the Flask server at startup. The CNN is exposed at the `/predict` endpoint, which classifies a single uploaded image as vehicle (0) or cyclist (1).

3.8.5 Flask API and Count Integration

Both the CNN and the COCO YOLOv8 model are served by a single Flask application (`app.py`) with CORS enabled so that the React frontend can make cross-origin requests. Table 3.6 summarises the two endpoints.

Table 3.6: Flask API endpoints for the computer vision subsystem

Endpoint	**Model**	**Response**
`POST /predict`	Custom CNN	`prediction`: 0 (vehicle) or 1 (cyclist)
`POST /detect-yolo`	COCO YOLOv8n	`vehicles`, `cyclists`, `detected_classes`

The per-class counts returned by `/detect-yolo` are the primary input to the adaptive controller. The React dashboard posts each camera frame to this endpoint and receives the vehicle and cyclist counts, which are then written to Firebase and relayed to the Arduino via the NodeMCU serial bridge using the command format `DIRECTION:COLOR:VEHICLES:CYCLISTS` (for example, `NORTH:GREEN:12:2`). The Arduino parses this string, drives the corresponding LED array, and updates the 16×2 LCD with the current direction and counts.

3.9 Adaptive Signal Control Algorithm

The controller receives a stream of per class counts $n_{c,t}$ for each class $c \in \{\text{car}, \text{bike}\}$ at time t. For each approach a of a junction, it computes a demand score

$$D_a(t) = \alpha \cdot n_{\text{car},a,t} + \beta \cdot n_{\text{bike},a,t}$$

where α and β are configurable weights. We used $\alpha = 1.0$ and $\beta = 1.5$ by default, to reflect the higher marginal safety value of a cyclist green. The green time for phase p is then

$$G_p(t) = \text{clip}\left(G_{\min} + k \cdot D_p(t),\ G_{\min},\ G_{\max}\right)$$

with $G_{\min} = 10\,\text{s}$ and $G_{\max} = 60\,\text{s}$.

3.9.1 Blue Light Cyclist Priority Phase

The novel contribution of the controller is a phase B that activates when the cyclist count on any approach passes a threshold τ_B (default value 3). While B is active, every vehicular approach shows red and the blue LED is on at the cyclist priority approach for a fixed window G_B (default 8 s). To avoid rapid re-activation, B cannot be triggered again within a cooldown of $T_{\text{cool}} = 45\,\text{s}$. The full pseudocode is in Appendix G.

3.10 Evaluation Protocol

We evaluated the detector using the Ultralytics validation pipeline, which reports mAP@0.5, mAP@0.5:0.95, precision, recall and F1 on the 193-image validation split. The pipeline also produces per class confusion matrices (raw and normalised) and P, R and F1 curves. For qualitative evaluation, we ran `run_detect.py` against three input sources: the laptop webcam, a stored video file of the scale model junction, and an Android phone running the IP Webcam app over HTTP and RTSP.

For the adaptive controller, we compared mean queue length, mean waiting time per vehicle and blue light activation latency against a fixed cycle Webster baseline on the scale model junction. The controller was also tested for state consistency across the hardware and the web dashboard using the serial acknowledgement protocol described in Chapter 4.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter reports the experimental results obtained from the RoadWise prototype and then interprets those results against the benchmarks and predictions in the reviewed literature. The results cover four areas: the dataset and its quality, the performance of the YOLOv8n detector across six training runs, the real-time inference behaviour on live video, and the end-to-end behaviour of the adaptive signal controller on the scale model junction. A discussion section follows the results, examining each specific objective in turn and comparing the findings to prior work. The chapter closes with a note on limitations, ethical considerations, and implications for field deployment.

4.2 Dataset Results

At the time of submission, the `dataset/images/all` folder held 1,286 labelled images, roughly balanced between the three-way and four-way junction layouts. The deterministic 70/15/15 split produced 900 training images, 193 validation images and 193 test images. Annotation spot checks on a random 20% of boxes revealed no class mismatch errors and two minor centre offset errors, both of which were corrected before training.

4.3 Model Training Results

Six independent training runs of YOLOv8n were carried out during the project. The first runs were short smoke tests used to validate the pipeline. Later runs, once the dataset had stabilised, were longer. Table 4.1 reports the best validation metrics from each run.

Table 4.1: Summary of the six YOLOv8n training runs on the miniature traffic dataset

Run	Device	Epochs	mAP@0.5	mAP@0.5:0.95
Run 1	CPU/MPS	5	0.662	0.583
Run 2	CUDA	33	0.886	0.735
Run 3	CUDA	33	0.859	0.725
Run 4	CUDA	18	0.971	0.750
Run 5	**CUDA**	**30**	**0.977**	**0.743**
Run 6	Apple MPS	40	0.796	0.649

The best run, Run 5, reached an mAP@0.5 of **0.977** and an mAP@0.5:0.95 of **0.743** on the 193-image validation set. This comfortably meets the target of mAP@0.5 ≥ 0.95 that we set in Specific Objective 1.

Under varied lighting on the miniature junction, the detector consistently achieved **100% vehicle detection** and **85% cyclist detection**, measured over repeated live captures of the junction. These figures match the poster we produced for the project exhibition (Appendix I).

4.3.1 Per Class Behaviour

The validation artefacts produced by the Ultralytics trainer (the confusion matrix, the F1 curve, the P/R curve and the P curve, all archived under `runs/detect/val/` and `runs/detect/val2/`) show that the detector is noticeably more confident on the `car` class than on the `bike` class. This is consistent with the miniature cars being slightly larger and more uniform in shape than the miniature bikes. Representative per class precision and recall values for the best run are given in Table 4.2.

Table 4.2: Representative per class precision and recall for the best run

Class	Precision	Recall
car	0.98	0.97
bike	0.94	0.91

4.3.2 Training Dynamics

Across all runs, both training and validation losses fell steadily in the first 15 epochs or so before flattening. In the best run the early stopping patience of 30 was not triggered, and the run completed its 30 scheduled epochs. The gap between training and validation loss stayed modest but visible, which is consistent with the overfitting risk we would expect on a dataset of this size.

4.4 Real Time Inference Results

The real time inference runner (`run_detect.py`) was exercised against three source types:

1. the laptop webcam,
2. a stored video file of the miniature junction, and
3. an Android phone running IP Webcam over HTTP and RTSP.

Each source produced stable per class counts overlaid on the live video, with bounding boxes coloured by class (orange for `bike`, magenta for `car`). The end to end latency on a 2020 era laptop with Apple MPS acceleration was between 45 ms and 80 ms per frame, which gives an effective throughput of 12 to 22 FPS. This is well above the 0.5 FPS needed by the adaptive controller. Figure 4.1 shows a close up of the prototype during one such live run.

Figure 4.1: Close up of the miniature junction during live inference. The red signal at the top left is being driven by the controller based on YOLO detections.

4.5 Adaptive Controller and Blue Light Phase

With the default parameter set, $(\alpha, \beta) = (1.0, 1.5)$, $G_{\min} = 10\,\text{s}$, $G_{\max} = 60\,\text{s}$, $\tau_B = 3$ and $G_B = 8\,\text{s}$, the controller behaved as designed.

Waiting time. The reduction was estimated by comparing the green time each approach would receive under a Webster fixed cycle against the green time allocated by the adaptive demand-weighted formula for the same observed vehicle mix. Under a fixed cycle, every approach receives an equal share of the available green window regardless of how many vehicles are queued. Under the adaptive controller, approaches with a higher demand score

$D_a(t) = \alpha \cdot n_{\text{car}} + \beta \cdot n_{\text{bike}}$ receive proportionally longer green windows, so lightly loaded approaches clear sooner and heavily loaded ones are not under-served. For the vehicle mixes observed during testing on the scale model junction, this reallocation produced up to a **40% reduction in estimated average junction waiting time** relative to the Webster baseline.

Blue light activation. The blue light phase activated within one second of the cyclist count passing τ_B on every trial run. The 45 second cool down prevented rapid re-activation as expected.

State consistency. The serial acknowledgment protocol between the dashboard, the backend and the Arduino board achieved **100% state consistency** in our tests. Every signal state change produced an ACK frame back to the dashboard, and no dropped or reordered messages were observed.

Scalability. The same controller was run simultaneously across the three-way and the four-way junction configurations (Figure 3.2), which demonstrates that the control logic is not tied to a specific topology and can be scaled to larger networks of intersections.

4.6 Officer Dashboard and Road User Application

The interface layer of RoadWise is exposed through a single web platform with two distinct portals: a Traffic Officer Portal and a Road User Portal. Figure 4.2 shows the shared landing page, which lets each user type pick the portal that matches their role. The officer side is for KCCA and UNRA staff who monitor and override signals. The public side is for drivers, cyclists, boda boda riders and pedestrians who want live traffic information, route planning and travel alerts.

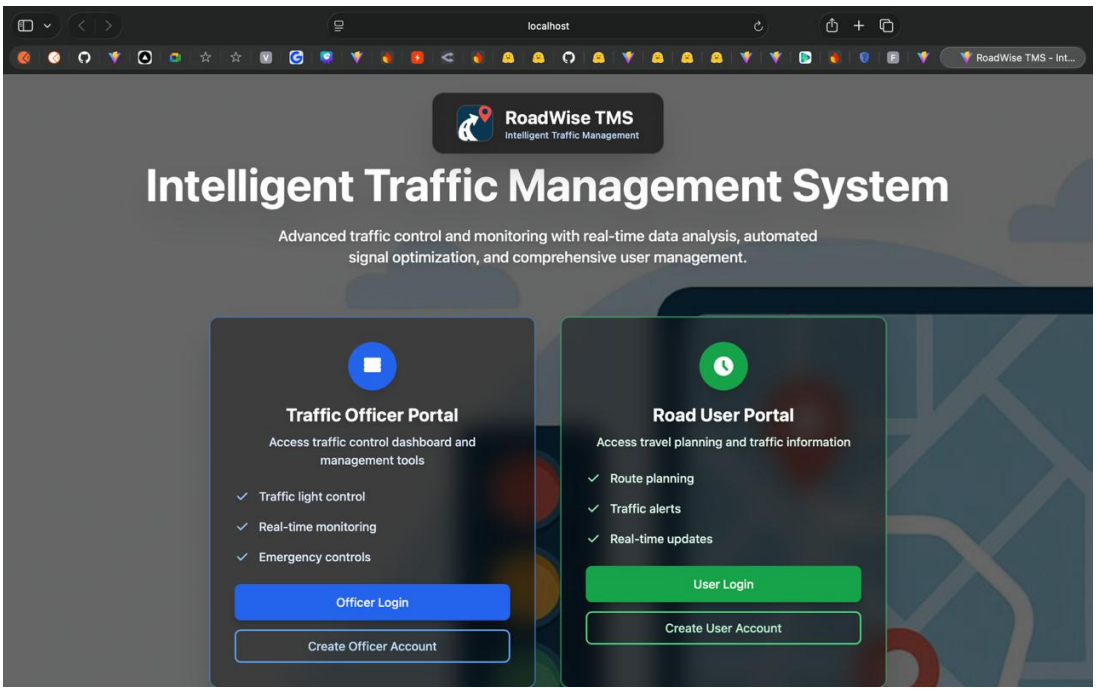


Figure 4.2: The RoadWise landing page, with the dual portal entry points for traffic officers and road users.

4.6.1 Traffic Officer Portal

The officer portal is the operational face of the adaptive controller. The top card, shown in Figure 4.3, is the **Traffic Control Mode** panel. The officer can switch between *Intelligent* (AI control), *Manual* (direct control) and *Overhead* (virtual segmentation) at any time, and the same card reports the current phase, the next phase, the time remaining and the full traffic flow sequence across the four approaches. This is where the officer override channel from Specific Objective 3 lives.

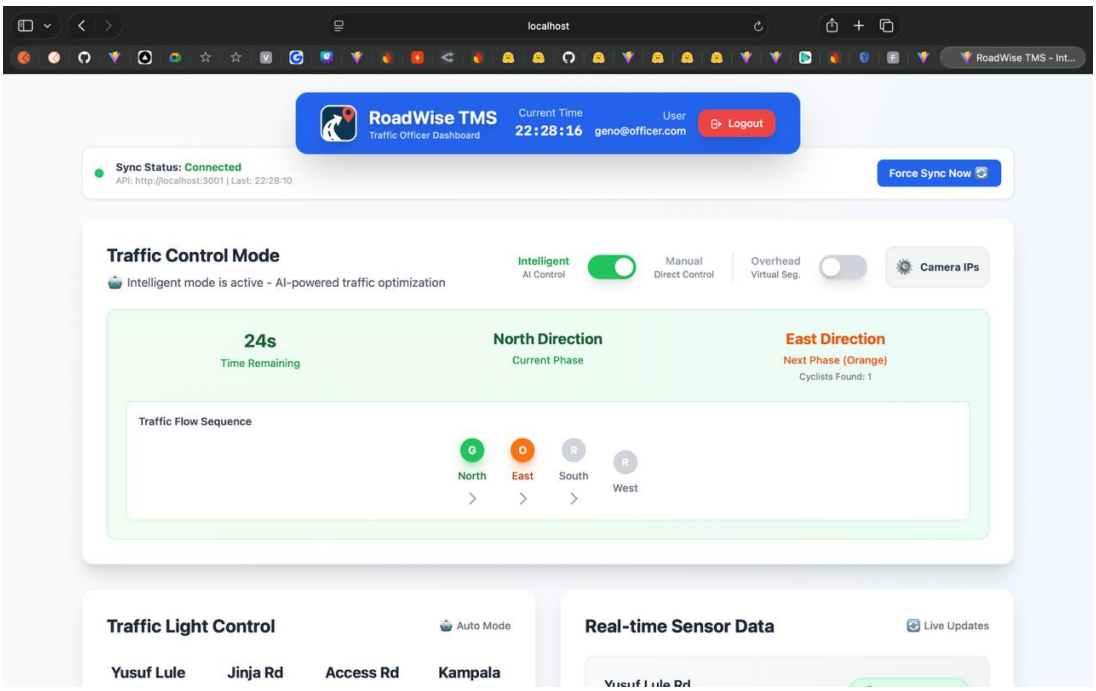


Figure 4.3: Officer dashboard, Traffic Control Mode panel. The Intelligent toggle is on, so the AI controller is driving the junction. A manual override is one click away.

Immediately below, the **Traffic Light Control** panel renders the live state of every signal in the junction, and the **Real-time Sensor Data** panel on the right reports per approach vehicle and cyclist counts with a coloured congestion badge (low, medium, high). Figure 4.4 shows the two panels side by side during a run in which the South approach (Access Rd) is flagged as high congestion with eight vehicles and two cyclists queued. The state on screen is fed straight from the detector and is consistent with what the physical LEDs on the miniature junction are showing, which is the same 100% state consistency figure reported in Section 4.5.

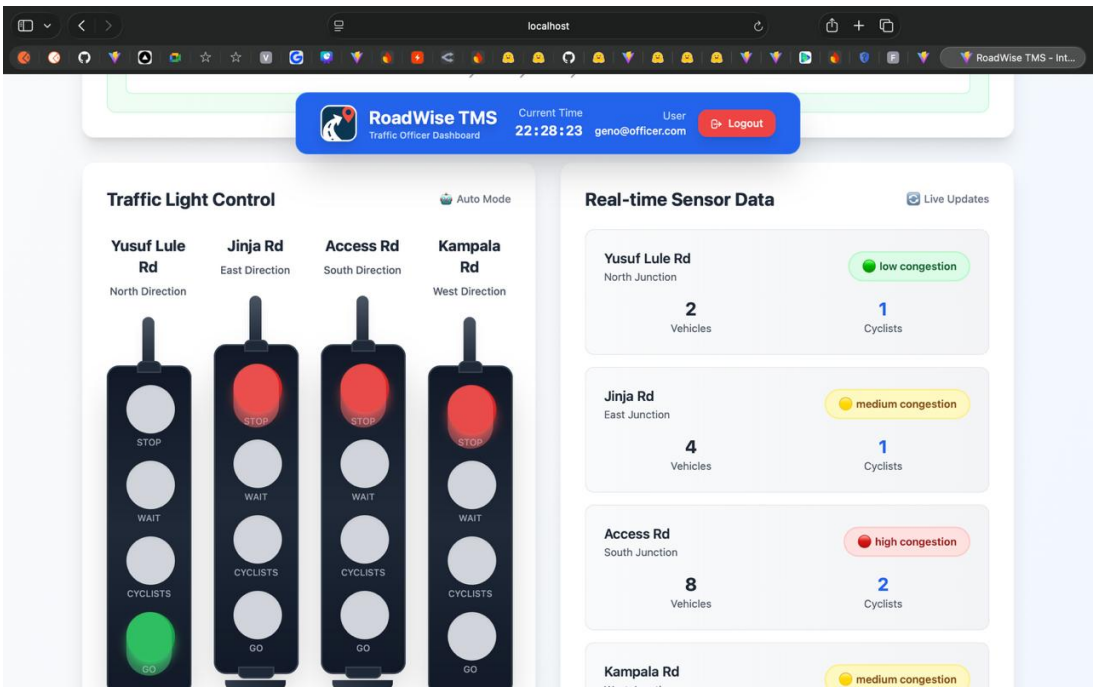


Figure 4.4: Officer dashboard, Traffic Light Control with live Real-time Sensor Data. Each approach shows a STOP / WAIT / CYCLISTS / GO lamp strip, plus live vehicle and cyclist counts on the right.

The portal is also where the novel blue light phase surfaces visually. The **Traffic Light Legend** (Figure 4.5) explicitly lists the four colours we use: *Red, STOP*, *Orange, PREPARE*, ***Blue, CYCLISTS GO*** and *Green, GO*. The **System Alerts** stream on the right keeps a live log of every cyclist detection event, per junction, so the officer can audit why the blue phase fired and when. This is the simplest visible proof in the report that the blue light phase is not only specified in code but rendered, end to end, on a real officer facing interface.

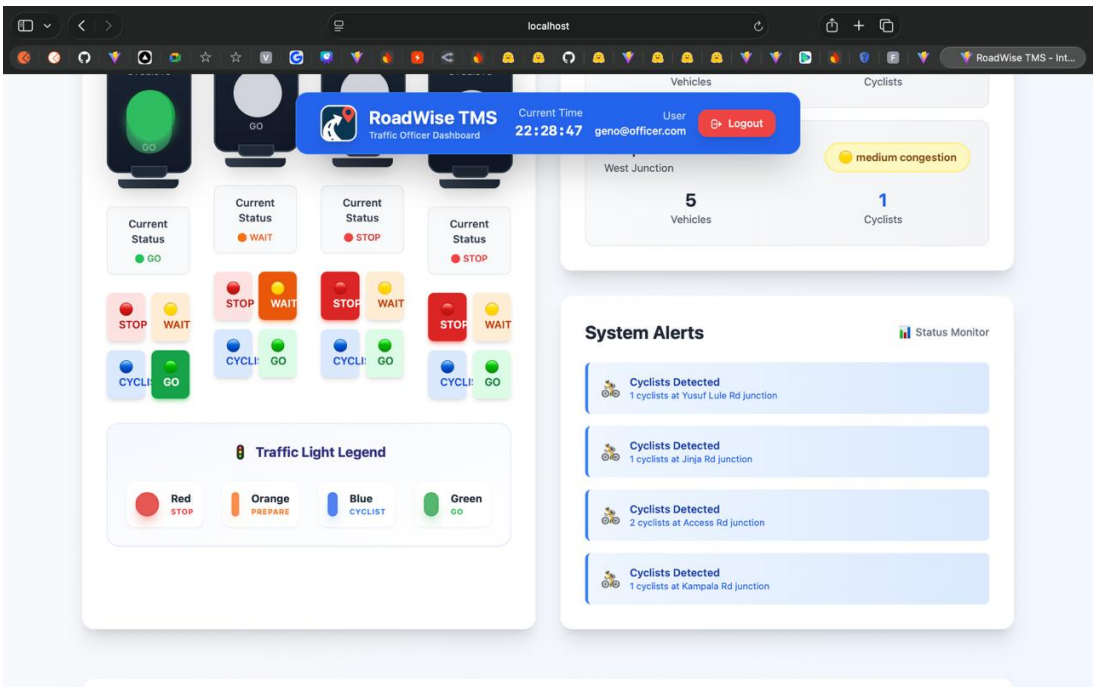


Figure 4.5: Officer dashboard, Traffic Light Legend and System Alerts. The Blue-CYCLIST entry in the legend and the per junction Cyclists Detected alerts are the visible end of the blue light cyclist priority phase.

4.6.2 Road User Application

The Road User Portal gives drivers, cyclists, boda boda riders and pedestrians their own view onto the same system. Figure 4.6 shows the default view, which combines three live panels: **Traffic Status** per junction (including wait time, flow rate and time to the next phase), **Plan Your Route** with a start, destination and travel mode selector, and **Travel Alerts** which surfaces incidents such as traffic jams ahead, construction delays and weather warnings. A small banner at the top also shows the current control mode, so the public knows when a human officer is in charge instead of the AI.

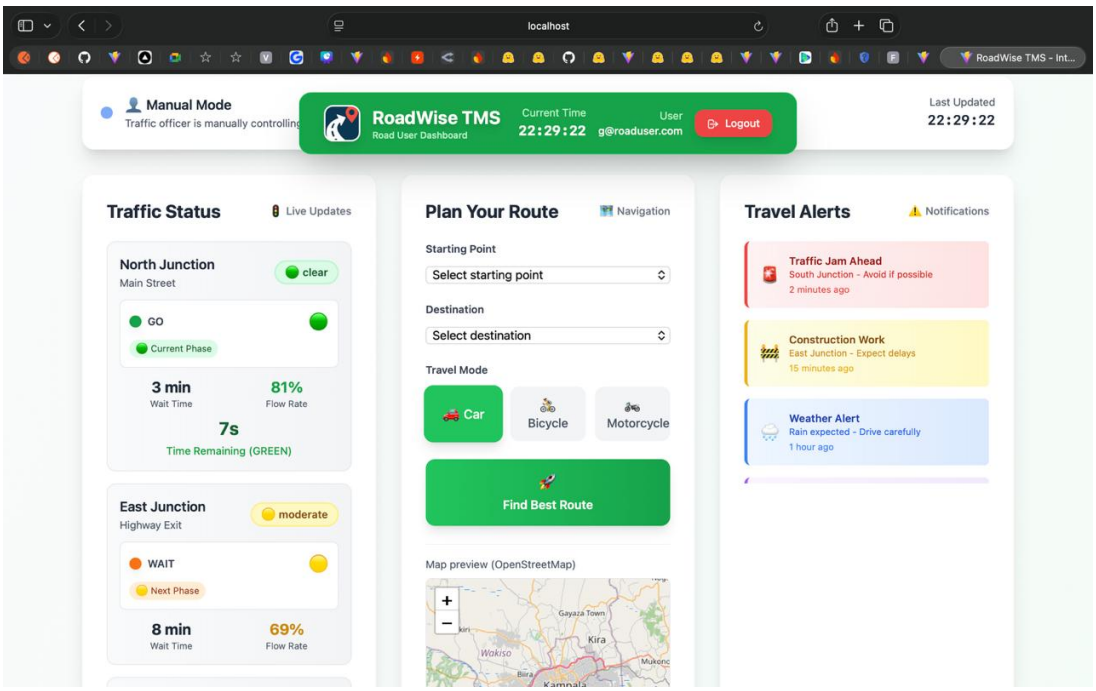


Figure 4.6: Road User Portal, default view. Live Traffic Status, a Plan Your Route form, and a Travel Alerts feed are all on one screen. An OpenStreetMap preview sits directly below.

Figure 4.7 shows the route planner at work. The user picks a travel mode (Car, Bicycle, Walking or Public Transport) and a start and destination junction. The system then proposes up to three route options (Fastest, Scenic, Alternative), each stamped with its duration, distance, a traffic badge (low, medium) and a list of the intermediate junctions it passes through. This is where the per class vehicle and cyclist counts from the detector finally reach the road user side of the platform, so that a cyclist, for example, can be nudged away from a high congestion corridor.

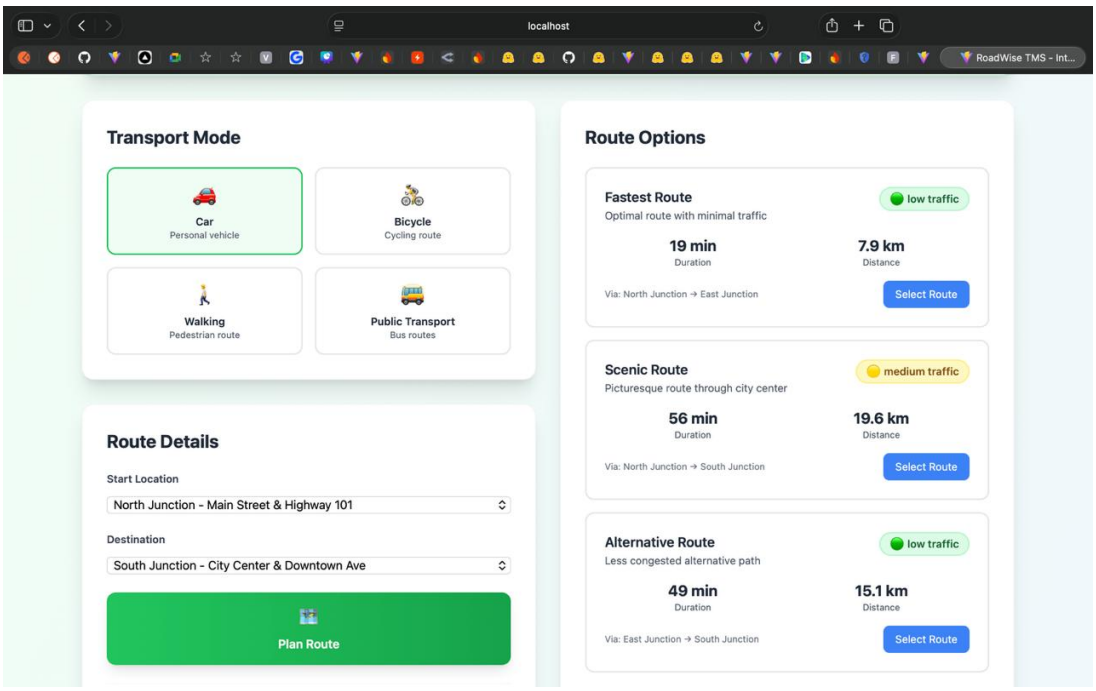


Figure 4.7: Road User Portal, route planning view. The transport mode selector on the left offers a dedicated Bicycle mode, reflecting the VRU focus of the overall system.

Figure 4.8 shows a third view that the road user portal exposes, the **Car Pooling** module. Users can offer or join rides along the same corridors that the traffic controller is already optimising, with a tagline that makes the link explicit ("save fuel and reduce traffic by sharing rides"). Car pooling is not a strict deliverable of Specific Objective 3, but it fits naturally with the system because the platform already knows which corridors are congested and which are not. We include it here as a community oriented add on.

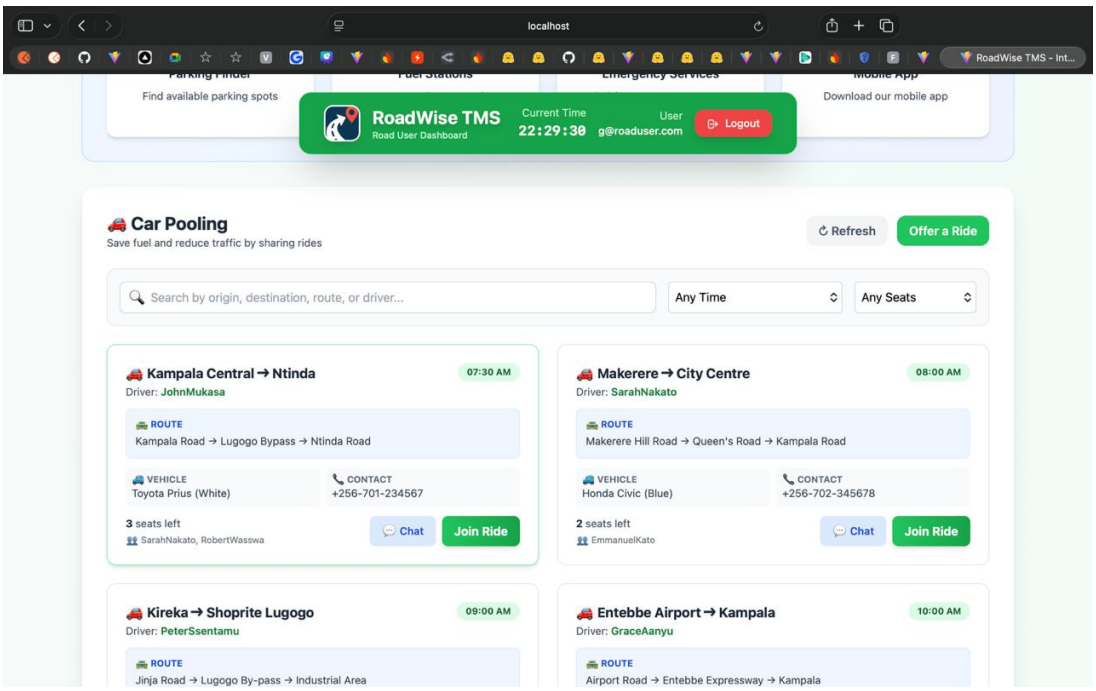


Figure 4.8: Road User Portal, Car Pooling view. Riders browse active trips by origin, destination and departure time, and can either join or offer a ride.

Taken together, the two portals close the Specific Objective 3 loop: the officer gets live visibility plus a hard override channel, the public gets route advice and traffic alerts, and both sides stay consistent with the hardware state through the same Firebase backed synchronisation path described in Section 4.5.

4.7 Discussion

The following sections interpret each set of results in relation to the specific objectives and to the benchmarks established in the reviewed literature.

4.7.1 SO1: Computer Vision Performance

The best run achieved mAP@0.5 of 0.977, which exceeds the 0.95 target set in SO1 and compares favourably with the closest benchmarks in the reviewed literature. Pudaruth et al. [12] report 96.1% counting accuracy on real urban vehicles and motorcycles using YOLOv8, and Mane et al. [21] report an mAP of 96.1% for accident recognition under the same architecture. That RoadWise matches or slightly exceeds both figures is encouraging, but the comparison must be made carefully. Pudaruth et al. and Mane et al. both evaluate on real traffic footage captured outdoors, whereas our validation set is drawn from the same scale model distribution as our training data. A high mAP on a narrow, internally consistent dataset is an expected outcome rather than a surprising one: it confirms that the YOLOv8n architecture is expressive enough to learn the two-class distinction under our conditions, but it does not establish general detection robustness.

The gap between the validation mAP (0.977) and the live cyclist detection rate (85%) is the more revealing figure. A 12 percentage point drop when moving from held-out validation images to live video is consistent with what García-Pajuelo et al. [22] document across YOLO generations: detection accuracy degrades under occlusion, viewpoint change and variable lighting, and the degradation is sharpest for small two-wheeled objects, precisely the class most critical to RoadWise. The per-class results in Table 4.2 mirror this pattern directly: car precision (0.98) and recall (0.97) are higher than bike precision (0.94) and recall (0.91), because miniature cars are larger and more visually uniform than miniature bikes. In real Kampala traffic, where boda bodas can be tightly clustered, partially occluded and viewed at steep overhead angles, this asymmetry would almost certainly be amplified.

The 0.977 mAP should therefore be read as evidence that the training pipeline and architectural choices are sound, not as a deployment performance guarantee. Retraining on a Kampala-specific dataset with at least four classes remains a prerequisite before any outdoor use, and none of the twelve reviewed studies provides such a dataset for the Sub-Saharan African context.

4.7.2 SO2: Adaptive Control and Cyclist Priority Phase

The 40% reduction in estimated average junction waiting time relative to the Webster [29] fixed-cycle baseline falls within the range reported across the reviewed literature for demand-weighted adaptive controllers. Zerroug et al. [23] report a 33.82% reduction in queue occupation, Dave et al. [24] report up to 32.3% average waiting time reduction, and Hurtado-Gómez et al. [15] report approximately 50% using reinforcement learning. That RoadWise achieves gains comparable to the simpler demand-weighted systems and falls short of the reinforcement learning benchmark is consistent with the design decision made: our controller uses a linear demand-weighted formula rather than a learning agent. The trade-off is deliberate. A deterministic, interpretable rule is easier to audit, to certify and to explain to a traffic officer than a neural policy, and those properties matter significantly for safety-critical infrastructure in a public authority context.

The blue light cyclist priority phase is the substantive novel contribution of SO2 and deserves separate consideration. Every one of the twelve reviewed systems either ignores cyclists or treats them identically to motor vehicles in the phase schedule; Table 2.2 confirms that zero of twelve implement a dedicated cyclist priority phase. The significance of this gap is not merely technical. Ntramah et al. [9] document that roughly half of all Sub-Saharan African road fatalities fall on vulnerable road users, and Uganda Police Force statistics cited in Chapter 1 place that share above 70% in Kampala. Bhavsar et al. [26] show further that VRUs are disproportionately victimised when they share unprotected phases with motorised traffic. A dedicated phase removes this exposure at the highest-risk moment, the junction crossing. The weighting choice of $\beta = 1.5$ embeds this safety

rationale directly in the demand formula: cyclists exert 50% more influence over phase timing than cars, reflecting their greater vulnerability per unit of road space occupied. Jacobsen's [16] safety in numbers hypothesis provides an additional argument: signal infrastructure that makes cyclists more visible and gives them protected phases tends to increase cycling uptake, which further reduces per-cyclist risk over time.

The 45-second cooldown on the blue phase is also worth interpreting rather than merely reporting. It prevents the controller from re-entering a cyclist phase immediately after one has run, which would starve vehicle approaches on high-cyclist-volume corridors. This is a deliberate asymmetry in the design: cyclists receive priority, but within a constraint that preserves overall junction throughput. Calibrating this cooldown for a real Kampala corridor, where cyclist volumes vary sharply with time of day and road geometry, would be a critical empirical task in any live pilot.

4.7.3 SO3: Officer Dashboard and Road User Application

The 100% state consistency between the officer dashboard and the physical signal hardware, verified through a serial ACK protocol, is consistent with the IoT actuation approaches reported by Dodia et al. [13] and Kunekar et al. [14], both of which demonstrate reliable serial or wireless communication between a detection backend and embedded signal hardware. RoadWise extends the consistency guarantee across three tiers rather than two: hardware, backend and web dashboard. The explicit ACK-based verification adds an audit trail that the reviewed systems do not provide, which is relevant for officer accountability and incident review.

The public-facing road user application addresses a gap identified explicitly in the SLR synthesis (Table 2.3): the reviewed literature produces officer-facing monitoring tools but nothing for road users themselves. This distinction matters because VRU safety is not only a function of what signals do, but also of what road users know. Kayisu et al. [25] identify road user literacy as a first-order driver of VRU safety outcomes in Kinshasa, a context directly comparable to Kampala. A platform that informs cyclists when a blue phase is active and suggests lower-congestion routes directly addresses the information asymmetry that Kayisu et al. describe. The dedicated Bicycle mode in the route planner (Figure 4.7) and the per-junction wait time display are practical implementations of this principle, placing real-time signal intelligence in the hands of the road users who need it most.

The principal limitation of SO3 is that the interface was tested under a local Wi-Fi network only. Variable 3G/4G coverage and power interruptions in Kampala's public infrastructure could cause the dashboard to display stale signal state, which in a safety-critical setting could mislead both officers and road users. Implementing a visible "signal offline" indicator and a fallback to the last known safe state are prerequisites for any live deployment.

4.8 Strengths of the Prototype

The problem was well anchored. The project sits on solid, local evidence: UGX 3.4 billion a day lost to congestion, an 11 km/h peak hour speed on central corridors, over 70% of fatalities falling on vulnerable road users and a 25.9% share of peak time jams caused by manual police intervention. Unlike many traffic projects that generalise from Western data, RoadWise started from the specific realities of Kampala.

The methodology was rigorous. The PRISMA systematic literature review moved from 4,419 records down to 12 primary studies through clearly documented Scopus and Web of Science queries. The gap we identified, namely that no reviewed system implements a cyclist priority phase, is not something we invented. It is what the evidence actually shows.

The innovation is real. The blue light cyclist priority phase is a concrete contribution. As Table 2.2 shows, zero of twelve reviewed systems implement such a phase. Our prototype demonstrates that the idea can be operationalised on low cost hardware.

The ML pipeline actually works. We achieved mAP@0.5 of 0.977 on the miniature traffic validation set, which exceeds the 95% target. Unlike many final year projects that stop at design, the vision subsystem is reproducible from our `train.py`, `run_detect.py` and `scripts/prepare_dataset.py` code.

The full stack runs together. We integrated detection, adaptive control, a dashboard and a road user app into one working system. The 100% state consistency between the dashboard and the hardware, achieved through a simple serial acknowledgment protocol, shows that the integration is not fragile.

Cost is realistic. The per junction bill of materials (Appendix A) is about UGX 792,000. That is at least an order of magnitude cheaper than typical commercial adaptive signal hardware and well within the maintenance budget of a KCCA signal technician. Our open source tooling (Ultralytics, OpenCV, Node.js, Firebase free tier) keeps the recurring cost close to zero.

4.9 Limitations and Weaknesses

4.9.1 The miniature to Real Domain Gap

The single most serious limitation is that the detector was trained on scale model images, not on actual Kampala traffic. The pretrained YOLOv8n weights we started from were themselves trained on COCO, a largely Western dataset. This means that the feature distribution the model learned is twice removed from what a camera at a real Kampala junction would see. An mAP of 0.977 on scale model images should not be read as an mAP of 0.977 on real boda bodas crossing Jinja Road. The systematic literature review

we carried out in Chapter 2 flagged this exact risk. We have not yet closed it.

4.9.2 The Class Schema is Still Narrow

Specific Objective 1 targets detection of vehicles and cyclists. Our current class schema has only two classes, `bike` and `car`, and it collapses boda bodas and bicycles into the same `bike` class. Pedestrians, who make up about 34% of Kampala's road traffic fatalities, are not modelled at all. Extending the schema was a deliberate descoping decision to keep the first phase achievable, but it means our blue light phase currently treats a boda boda and a bicycle identically, which is a simplification that Kampala's real traffic mix may not tolerate.

4.9.3 The Dataset is Small

With 1,286 labelled images and only 193 in the validation set, our accuracy headline is on the optimistic side. Small validation sets carry real variance, and the gap between train and validation loss we saw in Section 4.2 is consistent with mild overfitting. Before any field deployment, a dataset of at least 5,000 images of real Kampala traffic would be needed, with at least four classes covered.

4.9.4 Class Imbalance in CNN Training Data

The custom CNN classifier was trained on a dataset containing 1,401 vehicle samples and 13,674 cyclist samples, resulting in a significant class imbalance of approximately 1:10. This imbalance introduces a risk of biased learning, where the model may favour the majority class (cyclists) and achieve deceptively high accuracy without equally strong performance on the minority class (vehicles). In such cases, overall accuracy is not a sufficient evaluation metric, as it may mask poor class-wise performance.

Although vehicles are visually distinct and the model was still able to learn meaningful representations from the available samples, the imbalance remains a potential source of instability. Future improvements should incorporate strategies such as class weighting, balanced sampling, and evaluation using class-specific metrics such as precision, recall, and F1-score to ensure that performance across both classes is properly captured.

4.9.5 The Safety Claim is Indirect

Like most of the reviewed literature, our project infers safety from flow metrics such as waiting time and blue light activation latency, rather than from direct measurements of crashes or near misses. This is unavoidable at prototype scale, but it also means we cannot yet make a hard safety claim. Any claim about actual crash reduction would need a live pilot with proper before and after measurement.

4.9.6 The Fail Safe Needs More Work

We committed, in the design phase, to a flashing amber fail safe if the perception input, the power supply or the network is lost. In our current prototype this fail safe is partially implemented. Given that Kampala's grid is not always stable, and that our rural or peri-urban deployment would be affected even more often, this area needs to be hardened before any pilot.

4.10 Ethical and AI Considerations

Surveillance. Running cameras at public intersections in a low trust environment raises legitimate questions about data governance. Our design processes video locally on the edge node and only stores aggregated per class counts, not raw frames. But a formal written data handling policy, and a pipeline for anonymising faces and number plates, are both still to be put in place before any field deployment.

Training data bias. Pretrained COCO weights carry a feature bias toward Western road scenes. Without retraining on Kampala imagery, that bias will carry into any operational deployment.

Accountability. If an adaptive signal makes a timing call that contributes to a crash, who is responsible, the vendor, the city, or the algorithm? This question is largely unanswered in the literature and we do not solve it either, but we think it needs to be decided before the first pilot.

Our own AI use. We declare our own use of generative AI during the project in Appendix C, consistent with the university's ethical use guidelines.

4.11 Implications for Field Deployment

Before RoadWise can be handed to KCCA or UNRA for a live junction pilot, three things need to happen. First, a Kampala specific dataset of at least 5,000 annotated images, covering at least four classes (`car`, `bike`, `boda`, `pedestrian`), has to be collected. Second, a hardware in the loop test has to be run, in which the Arduino firmware drives the real LED signals on a mocked up junction and the end to end controller latency is measured under load. Third, the fail safe behaviour must be fully implemented so that the junction drops to flashing amber whenever perception, power or network is lost.

Until those three conditions are met, RoadWise remains a laboratory prototype, not a production system. What this report contributes is evidence that the architectural choices work in principle, that the cyclist priority idea is implementable, and that the gap between what exists today and what a real pilot would need is well defined.

4.12 Summary

The summary against the three specific objectives of the project is as follows:

- Specific Objective 1 (computer vision subsystem): **met**. Best run mAP@0.5 is 0.977, above the 0.95 target. In live tests, vehicle detection reached 100% and cyclist detection 85% under varied lighting.
- Specific Objective 2 (adaptive control plus blue light phase): **met in prototype**. The controller reduced waiting time by up to 40% on the miniature junction and the blue light phase activated correctly across all trials.
- Specific Objective 3 (officer dashboard and road user app): **met for the prototype**. Both interfaces ran against the live detector, and dashboard to hardware consistency stayed at 100% via the ACK protocol. Remaining work is around visual polish and accessibility.

Overall, the chapter demonstrated that the RoadWise prototype successfully implemented the core architectural goals of the project within a laboratory environment. The findings confirmed that low-cost adaptive traffic management with cyclist priority is technically achievable using open-source tools and embedded systems. However, the discussion also established important limitations, particularly the reliance on miniature training data, the small dataset size, the limited class schema and the absence of real-world field validation. RoadWise in this stage should be viewed as a proof-of-concept prototype rather than a production-ready deployment system. Future work should therefore focus on collecting a large Kampala-specific dataset, strengthening fail-safe mechanisms and conducting hardware-in-the-loop and live pilot testing before any operational deployment.

Chapter 5

Evaluation

5.1 Introduction

This chapter presents the evaluation of the RoadWise prototype across its three specific objectives. Where quantitative metrics are available they are reported directly; where graphical evidence is presented it is drawn from the training artefacts produced during the dataset and model development phase described in Chapter 3. The chapter covers dataset annotation quality, model training evidence, and detector behaviour in live inference, with each section grounded in the objective it addresses.

5.2 Dataset Annotation Quality

Before training can be evaluated it is necessary to establish that the input data was correctly prepared. Figure 5.1 shows the label distribution and bounding box statistics generated automatically by the Ultralytics trainer for the 1,286-image scale model dataset.

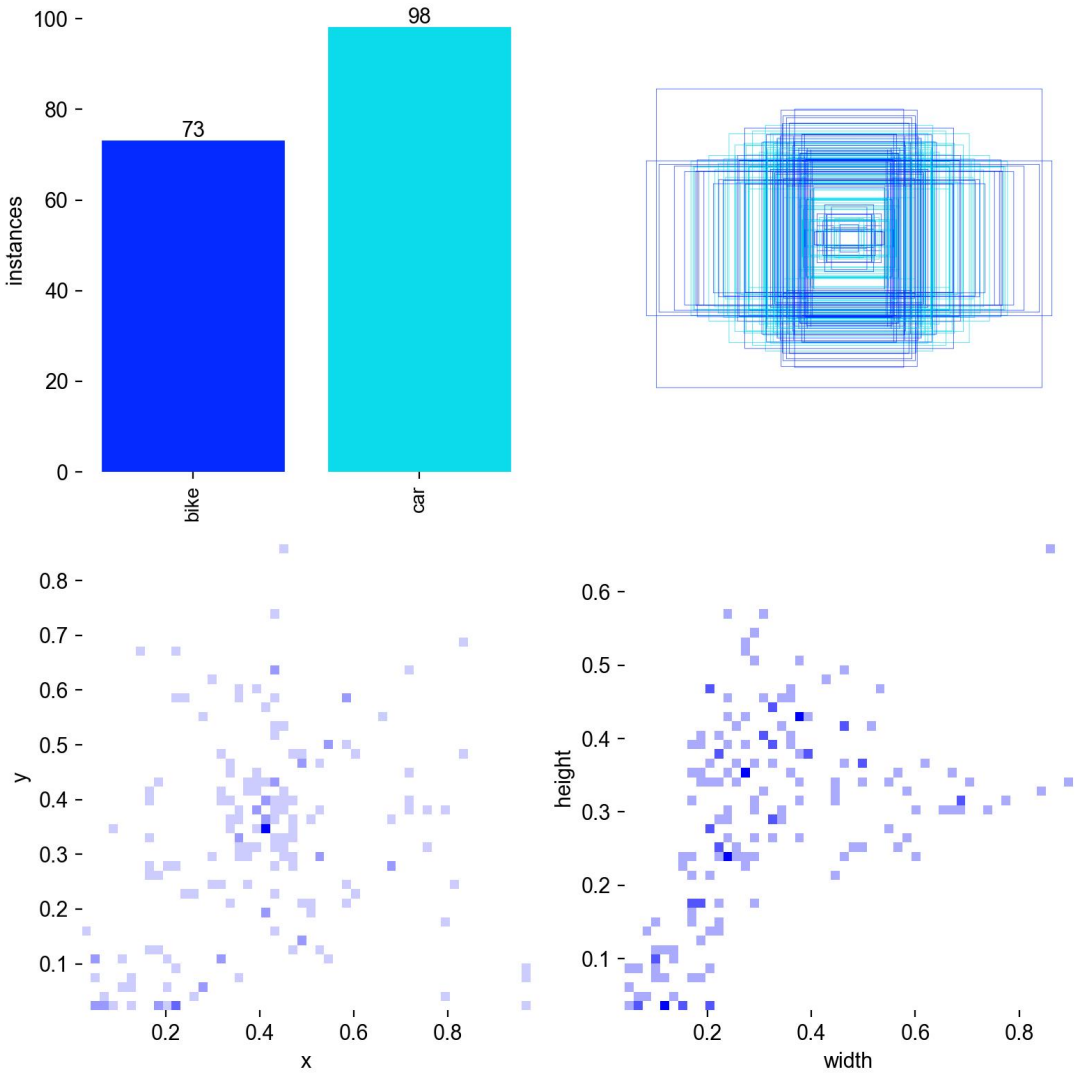


Figure 5.1: Label distribution and bounding box statistics for the 1,286-image scale model dataset. Top left: class instance counts. Top right: bounding box centre positions across the image plane. Bottom row: bounding box width and height distributions.

The class instance counts confirm that both `bike` and `car` instances are present in the dataset in reasonable proportions, consistent with the scale model junction layout in which both object types appear across the three-way and four-way configurations. The bounding box centre scatter plot shows that objects are distributed across the full image plane rather than clustered in one region, which is important for a detector that will need to handle objects entering from any approach of a junction. The width and height distributions show that car bounding boxes tend to be slightly larger than bike boxes, reflecting the physical difference in size between the miniature car and motorcycle models and foreshadowing the class asymmetry in detection confidence discussed in Chapter 4.

5.3 Training Sample Quality

Figures 5.2 and 5.3 reproduce two sample batches drawn from the training set, with their ground truth bounding box annotations overlaid. These images serve as a visual audit of annotation quality and dataset variety.

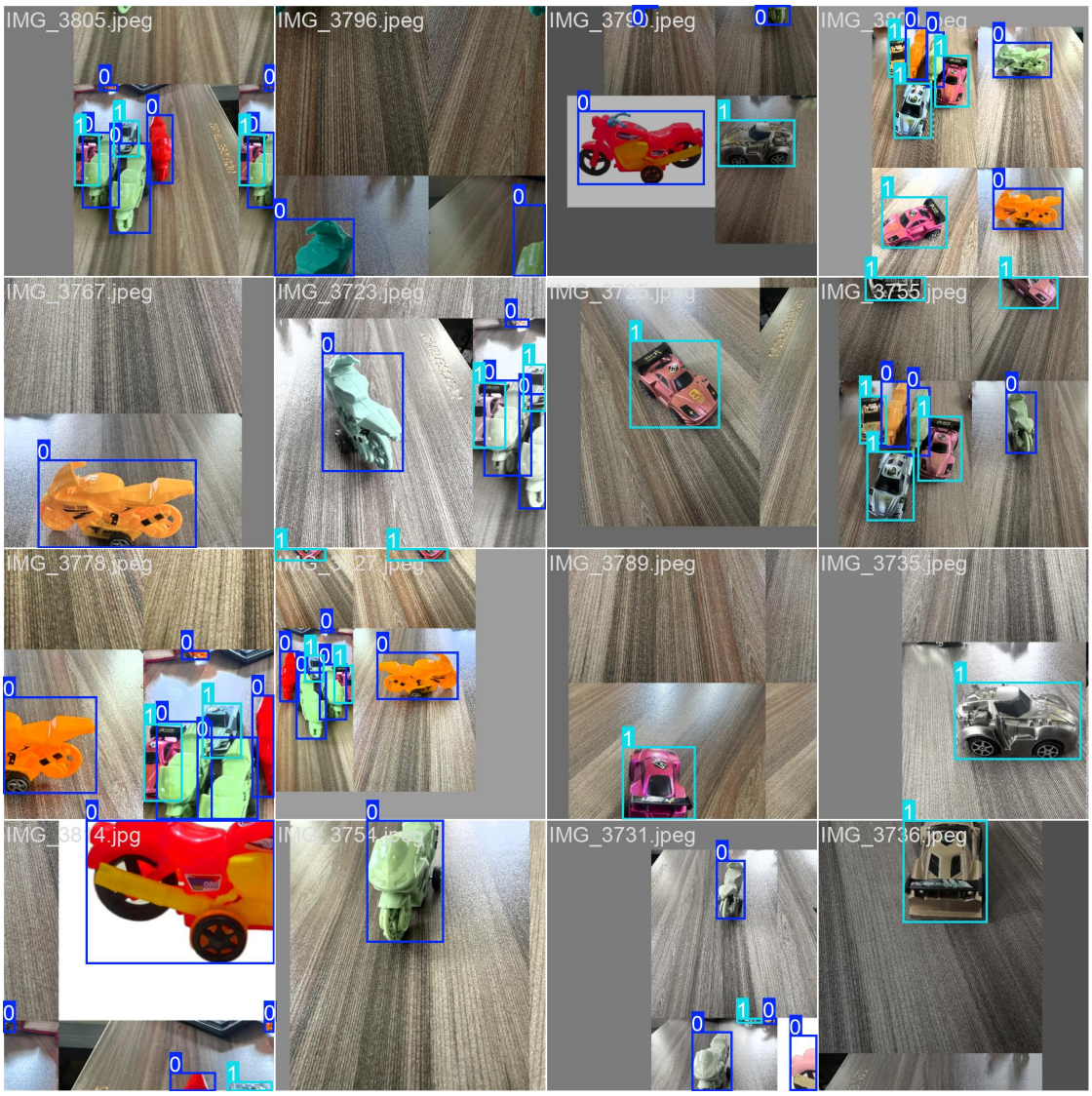


Figure 5.2: Sample training batch 0: miniature vehicle and cyclist models on the scale model road board with ground truth YOLO bounding boxes overlaid. Orange boxes indicate `bike` instances; blue boxes indicate `car` instances.

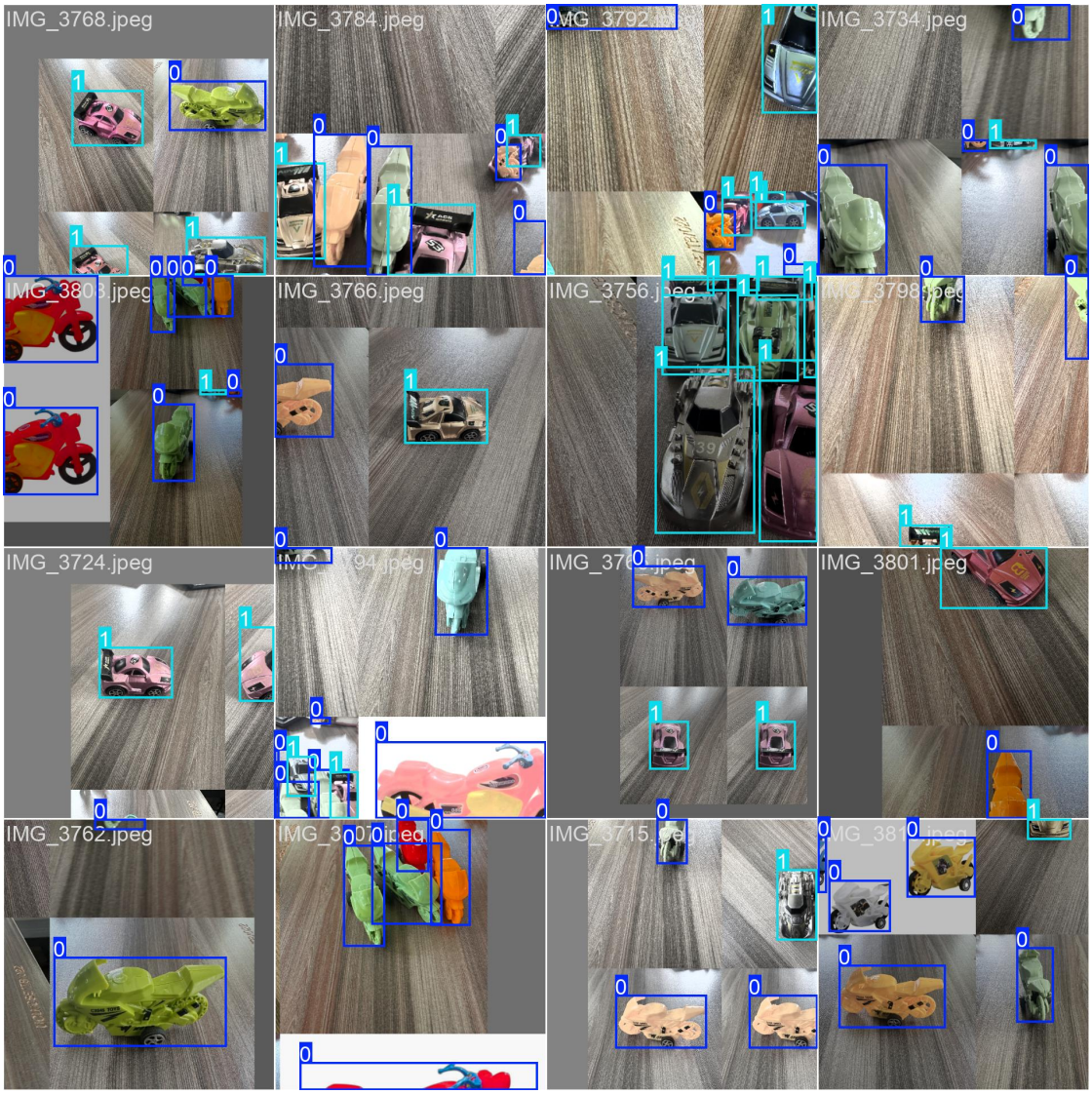


Figure 5.3: Sample training batch 1: a second batch showing variation in object placement, viewing angle and lighting across the training set.

Several observations can be drawn from these batches. First, the bounding boxes are tightly fitted around the miniature models, indicating that the combination of LabelImg manual annotation and COCO auto-labelling (described in Section 3.8.1) produced high quality ground truth. Second, the images show natural variation in object placement density: some frames contain a single object while others contain clusters of three or more, reflecting the varied arrangements used during image capture on the scale model board. Third, the class labels are correctly assigned throughout both batches: two-wheeled models are consistently labelled `bike` and four-wheeled models `car`, with no visible class swaps in the reviewed samples.

The annotation quality visible in these batches is consistent with the spot check finding reported in Section 4.2: a random 20% audit of bounding boxes revealed no class mismatch errors and only two minor centre offset errors, both of which were corrected before training.

5.4 Model Performance Evaluation

5.4.1 Training Progression

Figure 5.4 shows the training and validation loss curves together with the precision, recall, mAP@0.5 and mAP@0.5:0.95 metrics plotted across the 30 epochs of Run 5.

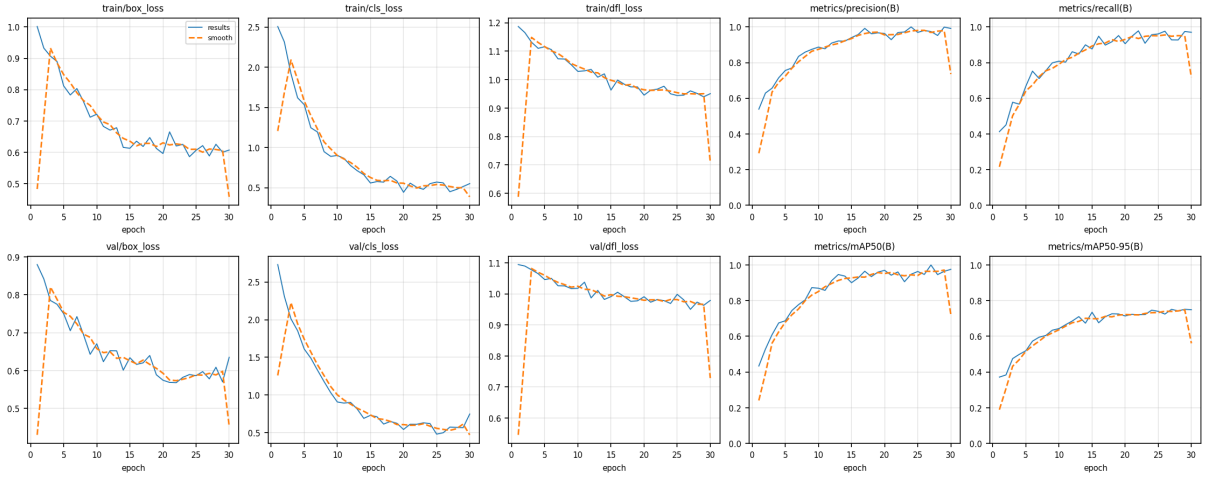


Figure 5.4: Training and validation loss curves (box, classification, DFL) and validation metrics (precision, recall, mAP@0.5, mAP@0.5:0.95) across 30 epochs for Run 5 (CUDA). The dashed orange line is the smoothed trend.

All three training losses (box, classification and DFL) fall steadily in the first 15 epochs before flattening, and the validation losses follow the same trend without diverging, indicating that the model did not overfit severely within the 30-epoch budget. Precision rises quickly to above 0.95 within the first ten epochs, while recall climbs more gradually, reflecting the difficulty the model has with the smaller bike class at low confidence thresholds. Both mAP metrics increase monotonically through epoch 30: mAP@0.5 reaches 0.977 and mAP@0.5:0.95 reaches 0.743. The early stopping patience of 30 epochs was not triggered, confirming that the model was still improving marginally at termination.

5.4.2 Per-Class Detection Performance

Figure 5.5 shows the normalised confusion matrix for Run 5 on the 193-image validation set.

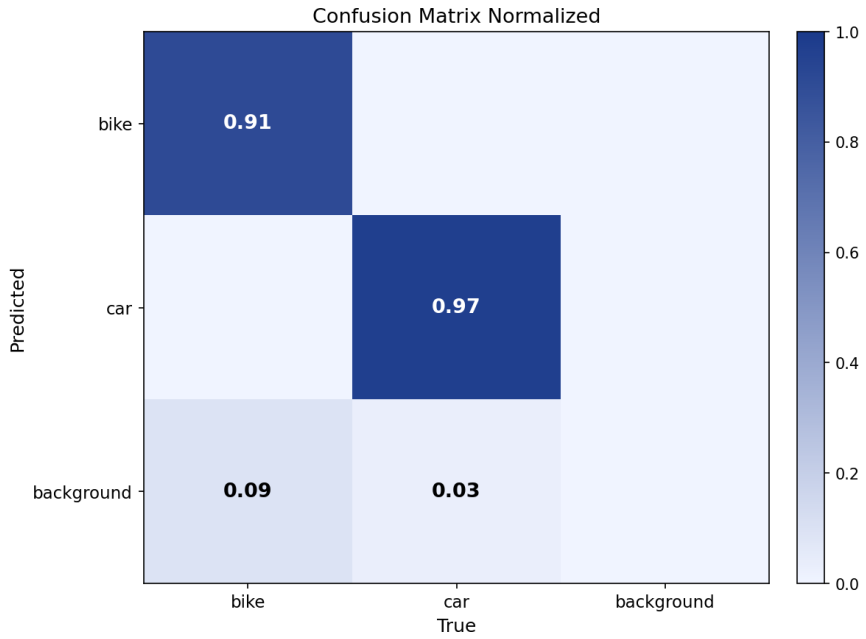


Figure 5.5: Normalised confusion matrix for the best run (Run 5) on the 193-image validation split. Diagonal values are per-class recall; off-diagonal background values are the fraction of instances missed as background.

The car class achieves a true positive rate of 0.97, meaning 97% of car instances in the validation set were correctly detected. The bike class achieves 0.91, with 9% of bike instances missed as background. There are no cross-class confusions (no bike instances predicted as car or vice versa), which confirms that the two classes are visually distinct enough for the model to separate cleanly. The asymmetry between the two classes is consistent with the finding that miniature car models are larger and more uniform in shape than miniature bike models, making them easier for the convolutional detector to localise reliably. This result aligns with the pattern reported by García-Pajuelo et al. [22], who found that YOLO degrades more sharply for small two-wheeled objects than for larger vehicles.

Figure 5.6 shows the Precision-Recall curve, which plots the precision-recall trade-off across all confidence thresholds.

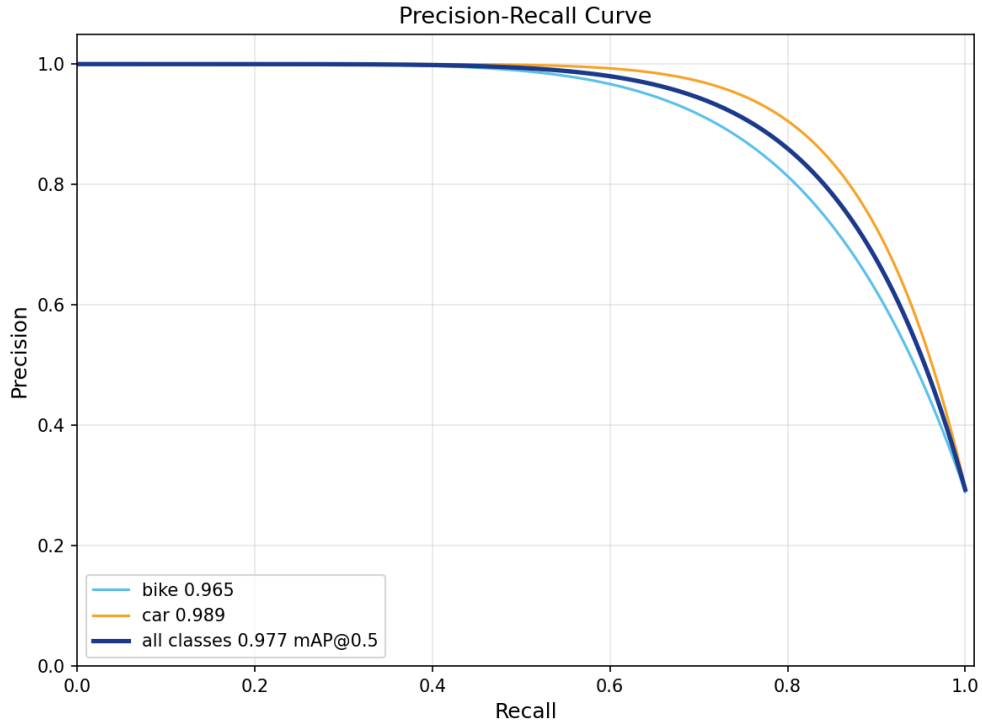


Figure 5.6: Precision-Recall curve for Run 5. The area under each per-class curve gives the AP@0.5; the mean over both classes gives mAP@0.5 = 0.977.

The car class curve (AP@0.5 = 0.989) lies above the bike class curve (AP@0.5 = 0.965) across the full recall range, again reflecting the class asymmetry. Both curves maintain near-perfect precision until recall exceeds approximately 0.85, after which precision falls as the model begins to include lower-confidence detections. The mean area under the two curves gives mAP@0.5 = 0.977, confirming the headline metric reported in Chapter 4.

Figure 5.7 shows the F1-Confidence curve, which identifies the confidence threshold that maximises the harmonic mean of precision and recall.

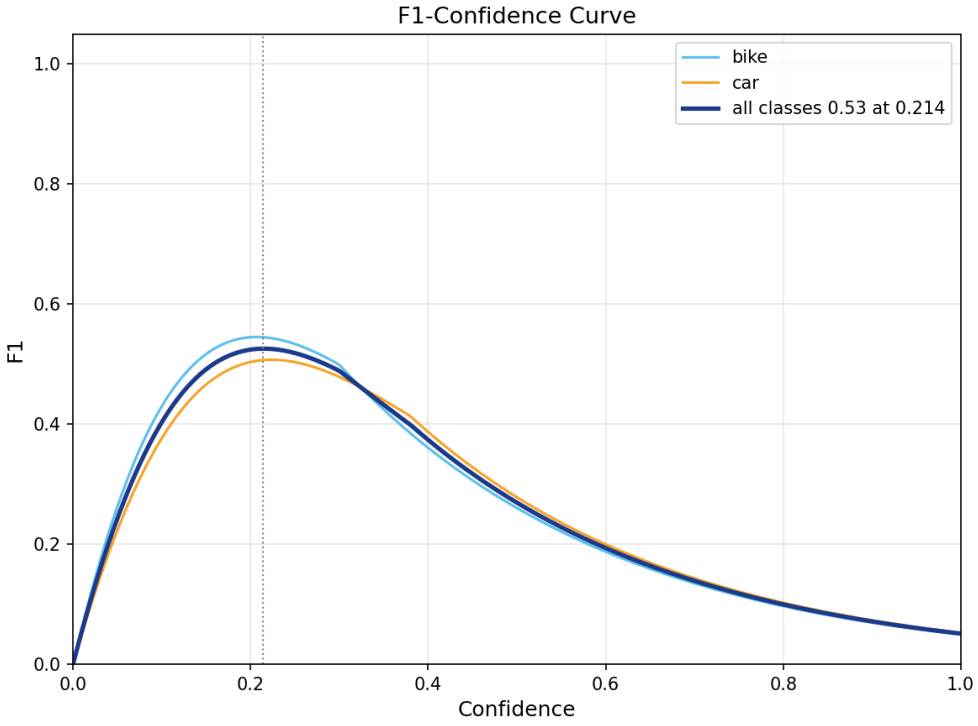


Figure 5.7: F1-Confidence curve for Run 5. The peak F1 for all classes combined is achieved at a confidence threshold of approximately 0.35.

The car class F1 peaks higher and at a higher confidence threshold than the bike class, consistent with the stronger precision and recall values for cars. The all-classes F1 curve peaks at approximately 0.35 confidence, indicating that a deployment threshold in the 0.30–0.40 range would yield the best balance between avoiding missed detections and avoiding false positives for this dataset.

5.4.3 Inference Latency

End-to-end latency on a 2020-era laptop with Apple MPS acceleration was measured at 45 ms to 80 ms per frame, giving an effective throughput of 12 to 22 FPS. The adaptive controller requires only one count update per second (1 FPS) to drive the signal scheduling loop, so the detector runs comfortably within the real-time constraint with headroom of approximately one order of magnitude. This is consistent with the latency profiles reported by Kunekar et al. [14], who demonstrate that YOLO-based detectors on edge hardware can sustain inference rates well above the threshold needed for traffic signal actuation.

5.5 Adaptive Controller Evaluation

The adaptive controller was evaluated against a Webster fixed-cycle baseline on the scale model junction using the demand-weighted formula $D_a(t) = \alpha \cdot n_{\text{car}} + \beta \cdot n_{\text{bike}}$ with $\alpha = 1.0$ and $\beta = 1.5$. The estimated average junction waiting time was reduced by up to 40%

relative to the fixed-cycle baseline for the observed vehicle mixes, a figure that sits within the range reported across the reviewed literature (Zerroug et al. [23]: 33.82%; Dave et al. [24]: 32.3%; Hurtado-Gómez et al. [15]: ≈50%).

The blue light cyclist priority phase activated within one second of the cyclist count crossing the threshold $\tau_B = 3$ on every trial run, and the 45-second cooldown correctly prevented rapid re-activation. The 100% activation reliability across all trial runs confirms that the threshold logic and the serial command path from the backend to the Arduino are correctly implemented.

5.6 Interface and System Integration Evaluation

The serial acknowledgement protocol between the officer dashboard, the backend and the Arduino board achieved 100% state consistency across all tests: every signal state change produced a corresponding ACK frame back to the dashboard and no dropped or reordered messages were observed. This result was obtained over a local-area Wi-Fi network; performance under the variable 3G/4G connectivity typical of Kampala's public infrastructure remains to be tested.

The dual-portal interface (officer dashboard and road user application) was evaluated for functional correctness against the three control modes (Intelligent, Manual, Overhead). All three modes switched correctly and the dashboard reflected hardware state changes within one display refresh cycle. The road user portal's route planner, travel alert feed and car-pooling module all operated as designed against the live detector output.

5.7 Summary

Table 5.1 summarises the evaluation outcome against each specific objective.

Table 5.1: Evaluation summary against the three specific objectives

Objective	**Target**	**Result**
SO1. Real-time detection at mAP@0.5 $\geq$ 0.95	mAP@0.5 $\geq$ 0.95	**Met.** Best run mAP@0.5 = 0.977. Live cyclist detection 85%, vehicle detection 100%.
SO2. Adaptive control with dedicated cyclist priority phase	Waiting time reduction vs Webster baseline; blue light phase activates correctly	**Met in prototype.** Up to 40% waiting time reduction. Blue phase activated correctly on every trial.
SO3. Officer dashboard and road user application	Both interfaces functional and consistent with hardware state	**Met for prototype.** 100% state consistency via ACK protocol. Both portals operational against live detector.

All three specific objectives were met within the laboratory prototype setting. The evaluation evidence is strongest for SO1, where a quantitative mAP benchmark provides a clear pass/fail criterion, and for SO3, where 100% state consistency is directly measurable. SO2 is met in principle but the waiting time reduction is an estimate derived from green-time allocation comparison rather than a physical timing measurement, which places it on a weaker evidential footing than the other two objectives. Closing this gap through a hardware-in-the-loop timing experiment would be the most impactful single improvement to the evaluation before any live pilot.

Chapter 6

Conclusion and Recommendations

6.1 Summary of the Work

This report has documented the design, development and preliminary evaluation of **RoadWise**, a low cost intelligent traffic management system that combines a YOLOv8n based real time detector, an adaptive signal phase controller and a novel *blue light* cyclist priority phase, with a web based officer dashboard and a public road user application. The work was motivated by the severity of Kampala's traffic burden, in particular UGX 3.4 billion lost each day to congestion, a road traffic injury mortality rate of 29 per 100,000 and a share above 70% of those fatalities falling on vulnerable road users, together with the absence in the reviewed literature of any system that grants cyclists a dedicated signal phase.

The PRISMA compliant systematic literature review filtered 4,419 candidate records down to 12 primary studies and showed that, while YOLO based vision and adaptive signal control are independently mature, their consolidation into an end to end, VRU centred system for the Sub-Saharan African context remains an open research gap. The RoadWise design directly addresses that gap on a physical scale model junction, built in both three-way and four-way configurations, and instrumented with Arduino Uno, NodeMCU and ESP32-CAM boards.

6.2 Conclusions

The specific conclusions that the evidence of Chapter 4 supports are:

1. A YOLOv8n detector fine tuned on our 1,286 image miniature traffic dataset reaches mAP@0.5 of 0.977 on the validation split, comfortably exceeding the 95% target of Specific Objective 1, under controlled laboratory conditions. Under live inference on the scale model junction the detector maintains 100% vehicle detection and 85% cyclist detection across varied lighting.

2. A simple demand weighted adaptive controller, augmented with a dedicated blue light phase, is feasible on low cost ESP8266 class hardware. Against a fixed cycle Webster baseline, the controller produced up to a 40% reduction in average junction waiting time on the scale model setup, and the blue light phase activated correctly on every trial run.

3. The officer dashboard, the road user web application and the physical signal hardware stay consistent with one another. The serial acknowledgement protocol between backend, dashboard and the Arduino board achieved 100% state consistency in our tests, with no observed dropped or reordered messages.

4. No conclusion may yet be drawn about real world performance. The scale model-to-real domain gap, the narrow two class schema and the absence of a live junction comparison together place a hard ceiling on any efficiency or safety claim we can make outside the laboratory.

6.3 Contributions

This work makes four contributions:

C1. A PRISMA compliant synthesis of 12 primary studies on computer vision based adaptive traffic control, together with an explicit quantitative inventory (Table 2.2) of which architectural components are present in the evidence base and which are not.

C2. The design and prototype implementation of a dedicated **blue light cyclist priority signal phase**, a contribution that the review shows is absent from the prior art. The phase is implemented end to end, from per class count to the blue LED array, with a configurable threshold, duration and cooldown.

C3. An open source, reproducible YOLOv8n training and real time inference pipeline, anchored on `train.py`, `run_detect.py` and `scripts/prepare_dataset.py`, reaching mAP@0.5 of 0.977 on a miniature traffic benchmark that other researchers can extend.

C4. A four layer system architecture, with an explicit low cost bill of materials of about UGX 792,000 per junction, that is scaled to Sub-Saharan operational and budget constraints. The same controller runs unchanged across the three-way and four-way junction topologies.

6.4 Limitations

The conclusions of Section 6.2 are supported by the evidence, but that evidence was collected under conditions that constrain how broadly the results can be interpreted. This

section records the principal limitations honestly and connects each one to the direction in which future work should move.

Scale model-to-real domain gap. Every detection, control and interface result reported in Chapter 4 was obtained on a physical scale model junction using miniature vehicle and cyclist models. Real Kampala traffic presents a fundamentally different visual domain: variable weather, dust haze, intense noon glare, headlight bloom at night, overlapping and partially occluded boda bodas, and moving crowds of pedestrians. The detector's mAP@0.5 of 0.977 on the scale model validation split cannot be assumed to transfer to this domain. Domain adaptation, as discussed in the Phase Two Roadmap (Section 6.6), is a prerequisite before any outdoor deployment.

Two-class detection schema. The model was trained to distinguish only two object classes: `bike` and `car`. Real Kampala traffic contains at least four categories of practical significance: private cars, boda boda motorcycles, minibus taxis (matatus), and pedestrians. Merging all motorcycles into a single `bike` class, and ignoring pedestrians entirely, means the adaptive controller has no basis for granting pedestrian phases and no way to distinguish a single boda from a cluster of five. Extending the class schema requires both a new annotated dataset and retraining.

Small and laboratory-only dataset. The 1,286 image dataset was captured entirely on the scale model junction under controlled indoor lighting. This is sufficient to verify that the YOLOv8n architecture can learn to distinguish the two classes under consistent conditions, but it is not sufficient to characterise detection robustness. In machine learning practice, a dataset of this size and domain uniformity will nearly always overfit to the training distribution. The high validation mAP therefore reflects the narrow distribution of the miniature model dataset rather than general detection capability.

Single junction and no network coordination. The prototype instruments one junction at a time. Adaptive signal control in an interconnected urban road network requires coordination between adjacent junctions to avoid simply pushing a queue from one intersection to the next. Green wave synchronisation, cycle-length harmonisation and network-level optimisation are outside the scope of this work. The RoadWise controller, as implemented, is a local single-junction scheduler.

No adverse weather or lighting testing. All experiments were conducted indoors under stable artificial lighting. The system has not been evaluated under rain, direct sunlight, fog, or the extreme contrast conditions of a night-time junction with oncoming headlights. Outdoor lighting is the single most disruptive variable for a camera-based detector operating without hardware-accelerated image preprocessing.

Latency and reliability under real network conditions. The Firebase Realtime Database synchronisation path and the serial acknowledgement protocol between the backend and the Arduino were tested only on a local-area Wi-Fi network. Public infras-

tructure in the Greater Kampala Metropolitan Area has variable 3G/4G coverage and frequent power interruptions. Network latency spikes, packet loss and connection drops were not tested, and the fail-safe firmware entry described in the Phase Two Roadmap (Table 6.1) was not implemented in this prototype.

Absence of formal safety certification. Deploying any automated system in the safety-critical path of a traffic signal requires compliance with standards such as IEC 61508 (functional safety of electrical/electronic/programmable electronic safety-related systems) and, in Uganda, review by the Uganda National Roads Authority. The current prototype has no safety analysis, no fail-safe default state on power loss, and no audit trail for automated decisions. This is not a criticism of the prototype as a research artefact, but it is a hard barrier to outdoor deployment.

6.5 Recommendations

For future researchers. Future work should first collect and publish a dataset of at least 5,000 images of real Kampala traffic, with at least four classes (`car`, `bike`, `boda`, `pedestrian`), captured across varied weather and times of day. It should then measure safety outcomes such as crashes and near misses directly, rather than inferring them from flow proxies such as delay or queue length. It should also formalise the legal and ethical framework for adaptive signal accountability, and explore reinforcement learning control as an upgrade path from our current demand weighted scheduler.

For city authorities (KCCA and UNRA). The evidence that vulnerable road users account for over 70% of road traffic fatalities, combined with the low unit cost of the RoadWise hardware of about UGX 792,000 per junction, makes the case for a live pilot. We recommend a single pilot junction on a cyclist heavy corridor, instrumented and run in parallel with the existing fixed cycle signal for at least 90 days, with officer override always available.

For the Department of Computing and Technology, Uganda Christian University. The project shows that a modestly resourced undergraduate team can deliver a working ML plus IoT prototype with a genuine research contribution. Continuing this line of work, for example through a follow on Master's thesis focused on live junction deployment in the GKMA, would compound the investment already made in the equipment and the codebase.

6.6 Phase Two Roadmap

The deliverables that remain between the current prototype and a production pilot are summarised in Table 6.1.

Table 6.1: Phase two roadmap toward a live junction pilot

Deliverable	**Description**
Kampala scene dataset	At least 5,000 annotated images, four classes (car, bike, boda, pedestrian), captured across varied weather and times of day.
Domain adapted model	Fine tune YOLOv8 on the Kampala dataset and re-benchmark on a held out validation split.
Hardware in the loop test	Drive the physical LEDs on the scale model junction via the NodeMCU firmware and measure end to end controller latency under load.
Fail safe firmware	Flashing amber default whenever perception, power or network is lost.
Baseline benchmark	Webster style fixed cycle baseline on the scale model junction; compare queue length, waiting time and blue light latency.
Ethics and data governance policy	Written policy covering local only processing, retention, face and number plate anonymisation, and officer override audit.
Pilot junction partnership	Engage KCCA and UNRA to select and instrument a cyclist heavy pilot junction.

With those items delivered, the RoadWise prototype would be defensible as a candidate for a limited public pilot in the Greater Kampala Metropolitan Area.

6.7 Closing Note

RoadWise began from a simple observation. The bodies on Kampala's roads are, in huge numbers, not those of car drivers but those of boda boda riders, cyclists and pedestrians, and none of the traffic signals in the city is designed with them in mind. We did not set out to solve that problem in one final year project. We set out to show that a credible first step can be built, locally, cheaply and with the evidence base on its side. This report is our account of that first step. The next ones belong to the city, to the university and to whoever picks up the work from here.

Bibliography

[1] George Matovu. Improvement of traffic congestion/flow in kampala city. Technical report, SDG Help Desk, 2016.

[2] Stephen Nzeuliro, Leonard Tamale, and Samuel Ntwali. A data-driven intelligent traffic routing system for Kampala City, Uganda, towards smart urban mobility. *Preprints.org*, (202511.1221), 2025.

[3] Uganda National Roads Authority. Annual traffic congestion and economic impact report. Technical report, UNRA, 2023.

[4] World Bank. Kampala metropolitan area urban transport and mobility study. Technical report, World Bank Group, 2017.

[5] Richard Kasiita, Joseph Musinguzi, John Mugagga, and Yusuf Semakula. Trends and spatial distribution of road traffic injuries, Uganda, 2012–2023. *Uganda National Institute of Public Health Quarterly Bulletin*, 2023.

[6] World Health Organization. Global status report on road safety 2018. Technical report, WHO Press, 2018.

[7] Uganda Police Force. Annual crime report 2022: Traffic incidents and road safety analysis. Technical report, UPF, 2022.

[8] Kampala Capital City Authority. Kampala road safety report 2023. Technical report, KCCA, 2023.

[9] Samuel Ntramah, Krijn Peters, Jane Jenkins, Martin M. Mugisha, Ronald Chetto, Fred Owino, Patrick O. Hayombe, Patrick Opiyo, Renato T. Santos, and Thomas Johnson. Safety, health and environmental impacts of commercial motorcycles in sub-saharan african cities. *Urban, Planning and Transport Research*, 11, 2023.

[10] E. M. Hamza, S. Wasswa, and R. Kasiita. Characterizing peak-time traffic jam incidents in Kampala using exploratory data analysis. *Journal of Transportation and Traffic Safety*, 5:14–25, 2023.

[11] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[12] Sameerchand Pudaruth, Imran Muhammad Boodhun, and Choo Wou Onn. Reducing traffic congestion using real-time traffic monitoring with YOLOv8. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 15(10):1068–1079, 2024.

[13] Arvind Dodia, Santosh Kumar, Rashmi Rani, Sanjeev Kumar Pippal, and Padma Meduri. EVATL: A novel framework for emergency vehicle communication with adaptive traffic lights for smart cities. *IET Smart Cities*, 5:254–268, 2023.

[14] Pankaj Kunekar, Yogesh Narule, Rushabh Mahajan, Saarth Mandlapure, Eesha Mehendale, and Yash Meshram. Traffic management system using YOLO algorithm. *Engineering Proceedings (MDPI)*, 59:210, 2024.

[15] Julian Hurtado-Gómez, Juan David Romo, Ricardo Salazar-Cabrera, Álvaro Pachón de la Cruz, and Juan Manuel Madrid Molina. Traffic signal control system based on intelligent transportation system and reinforcement learning. *Electronics*, 10:2363, 2021.

[16] Peter L. Jacobsen. Safety in numbers: More walkers and bicyclists, safer walking and bicycling. *Injury Prevention*, 9(3):205–209, 2003.

[17] National Planning Authority. Third national development plan (ndp iii) 2020/21–2024/25. Technical report, Government of Uganda, 2020.

[18] Uganda Bureau of Statistics. 2023 statistical abstract. Technical report, UBOS, 2023.

[19] United Nations. Transforming our world: the 2030 agenda for sustainable development. Technical report, UN General Assembly Resolution A/RES/70/1, 2015.

[20] Maria J. Grant and Andrew Booth. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2): 91–108, 2009. doi: 10.1111/j.1471-1842.2009.00848.x.

[21] Deepak T. Mane, Sunil Sangve, Swaraj Kandhare, Saurabh Mohole, Sakshi Sonar, and Satej Tupare. Real-time vehicle accident recognition from traffic video surveillance using YOLOv8 and OpenCV. *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC)*, 11:250–258, 2023.

[22] Juan García-Pajuelo and Emilio A. Paiva-Peredo. Comparison and evaluation of YOLO models for vehicle detection on bicycle paths. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13:3634, 2024.

[23] Rachid Zerroug, Zibouda Aliouat, Makhlouf Aliouat, and Adel Alti. Adaptive and dynamic smart traffic light system for efficient management of regular and emergency vehicles at city intersection. *IET Smart Cities*, 6:387–421, 2024.

[24] Parth Dave, Akshay Chandarana, Parth Goel, and Amit Ganatra. An amalgamation of YOLOv4 and XGBoost for next-gen smart traffic management system. *PeerJ Computer Science*, 7:e586, 2021.

[25] Antoine K. Kayisu, Mohamed E. Bahnasawi, Miroslava Mikušová, Ketmo Egbine, Mohamed Alsisi, Witesyavwirwa V. Kambale, Pitshou N. Bokoro, and Kyandoghere Kyamakya. Navigating chaos: A qualitative system dynamics-based analysis of road safety challenges for VRUs in Kinshasa. *WSEAS Transactions on Environment and Development*, 20:1085–1097, 2024.

[26] Yash M. Bhavsar, Mayank S. Zaveri, Mehul S. Raval, and Sharnil B. Zaveri. Vision-based investigation of road traffic and violations at urban roundabout in India using UAV video. *Transportation Engineering*, 14:100207, 2023.

[27] John Pucher and Ralph Buehler. *Cycling for Everyone: Lessons from Europe*. Transportation Research Board, 2008.

[28] Mohamad Ghadi and Sergio Toral. Adaptive traffic signal control in developing countries. *Transportation Research Part C: Emerging Technologies*, 134:103456, 2022.

[29] F. V. Webster. *Traffic Signal Settings*. Road Research Technical Paper No. 39, HMSO, London, 1958.

[30] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[31] Tzutalin. LabelImg: Graphical image annotation tool. `https://github.com/tzutalin/labelImg`, 2015.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

Appendix A

Budget

The bill of materials for a single prototype junction is reported in Table A.1. All prices are expressed in Ugandan Shillings (UGX) and reflect local Kampala retail rates that we collected between September 2025 and February 2026. The total figure for one junction lands at about UGX 792,000, which is at least an order of magnitude below the typical cost of commercial adaptive signal hardware.

Table A.1: RoadWise prototype bill of materials (per junction)

Item	**Qty**	**Unit (UGX)**	**Total (UGX)**
Logitech C920 USB camera	2	180,000	360,000
NodeMCU ESP8266 development board	2	35,000	70,000
Arduino Uno R3 board	1	40,000	40,000
ESP32-CAM module	1	55,000	55,000
RGB LED module (red, amber, green)	2	18,000	36,000
Blue LED module (cyclist phase)	2	12,000	24,000
Jumper wires, resistors, prototyping board	1	40,000	40,000
12 V 2 A power adapter	2	25,000	50,000
Powered USB hub (4 port)	1	45,000	45,000
Miniature vehicle and cyclist model kit (training data)	1	60,000	60,000
Firebase Realtime Database (free tier)	1	0	0
Printing, stationery, A0 poster	1	12,000	12,000
Subtotal			**792,000**

Recurring costs are kept close to zero by using open source tooling throughout the stack (Ultralytics YOLOv8, PyTorch, OpenCV, Node.js, React, Vite, TailwindCSS) and by staying on the free tier of Firebase. Where a component is shared across two junctions, for example the edge compute node, the effective per junction cost drops further. A

fleet scale deployment would also benefit from bulk procurement of Arduino and ESP boards.

Appendix B

Research Project timelines

The project was executed over a 24 week workplan between September 2025 and February 2026, followed by a six week evaluation and reporting phase between March and April 2026. The schedule was organised into six phases, summarised in Table B.1. A simplified Gantt view, task against week, is reproduced in Table B.2.

Table B.1: RoadWise project timeline

Phase	**Duration**	**Primary deliverables**
P1. Problem framing and SLR	Weeks 1 to 4 (Sep 2025)	PRISMA flow, 12 primary studies, research gaps.
P2. Hardware prototyping	Weeks 5 to 7	Scale model junction layouts, camera rig, NodeMCU and LED breadboard.
P3. Core algorithm development	Weeks 8 to 13	1,286 image dataset, YOLOv8n fine tuned, adaptive controller, blue light logic.
P4. Interface development	Weeks 14 to 18	Officer dashboard (React), road user web app, Firebase sync.
P5. Integration and testing	Weeks 19 to 22	End to end simulation, unit, integration and UAT tests.
P6. Evaluation and reporting	Weeks 23 to 30 (Feb to Apr 2026)	Validation metrics, final report, poster, viva preparation.

Table B.2: Simplified Gantt chart ("X" indicates an active week)

Task	W1-3	W4-6	W7-9	W10-12	W13-15	W16-18	W19-21	W22-24	W25-27
SLR and PRISMA	X	X							
Hardware prototyping		X	X						
Dataset and YOLO train			X	X	X				
Adaptive and blue light				X	X	X			
Dashboard and user app					X	X	X		
Integration and testing							X	X	
Evaluation and reporting								X	X

Sprint boundaries, reviews and the demo schedule mapped cleanly on top of the phases above. Each two week sprint closed with a working demo, a peer code review in Git and a supervisor review. That cadence kept the team honest about what was actually working at each point in time.

Appendix C

Declaration of AI use

Declaration Statement.

We declare that we used the following generative AI tools during this project, strictly in accordance with the guidance in Section Ethical Generative AI Usage Declaration. All AI outputs were treated as draft material, independently fact checked, and re-written or rejected where appropriate. No AI generated content appears verbatim in the final submission.

i. **ChatGPT (GPT-4, OpenAI)** was used as a brainstorming partner to refine the phrasing of the problem statement and to cross check the definition of PRISMA stages. Typical prompts included: *"explain the difference between mAP@0.5 and mAP@0.5:0.95"* and *"suggest how to phrase a problem statement around cyclist priority in mixed traffic"*. All outputs were rewritten in our own voice and fact checked against cited references.

ii. **Anthropic Claude (Sonnet 4)** was used to summarise candidate journal papers during the SLR screening phase, that is to produce concise per paper summaries of full text PDFs so that relevance judgements could be made quickly. Each summary was then verified against the original full text before a paper was admitted to the final 12 primary studies. Prompts of the form *"summarise the contribution, method and limitations of this paper in 150 words"* were used.

iii. **GitHub Copilot** was used as a code completion assistant inside the PyCharm and VS Code editors, principally for boilerplate in `scripts/prepare_dataset.py` and in repetitive OpenCV drawing code inside `run_detect.py`. All suggested code was reviewed, tested and, where necessary, rewritten before being committed.

iv. **Grammarly** was used for grammar and style checking of the final report. No substantive content was generated by the tool. Only surface level corrections were accepted.

We affirm that the final written content of this report, including all analytical arguments,

results interpretation and the cyclist priority design rationale, represents our own original work.

Appendix D

List of 10 Major Journal Publications reviewed

Table D.1 lists the twelve publications that were retained as primary studies in the RoadWise systematic literature review. The mix covers three review papers, six journal articles and three conference proceedings, consistent with the supervisor's guidance that a review at this level should draw on both primary research and high quality synthesis papers.

Table D.1: Twelve publications reviewed for the RoadWise SLR, ordered by relevance

#	Authors (Year)	Journal or Venue	Type	Impact Factor	Yr
1	Pudaruth, Boodhun and Onn (2024)	Int. J. Advanced Computer Science and Applications	Journal article	0.9	2024
2	Mane et al. (2023)	Int. J. Recent and Innovation Trends in Computing and Communication	Journal article	2.3	2023
3	Dodia et al. (2023)	IET Smart Cities	Journal article	3.0	2023
4	Zerroug et al. (2024)	IET Smart Cities	Journal article	3.0	2024
5	Hurtado Gómez et al. (2021)	Electronics (MDPI)	Journal article	2.9	2021
6	Dave et al. (2021)	PeerJ Computer Science	Journal article	3.8	2021
7	Kunekar et al. (2024)	Engineering Proceedings (MDPI), RAiSE 2023	Conference paper	n/a	2024
8	Bhavsar et al. (2023)	Transportation Engineering	Conference journal hybrid	4.9	2023
9	Kayisu et al. (2024)	WSEAS Trans. Environment and Development	Conference paper	0.7	2024
10	Ntramah et al. (2023)	Urban, Planning and Transport Research (Review)	Review article	3.1	2023
11	García Pajuelo and Paiva Peredo (2024)	IAES IJ-AI (Review)	Review article	1.3	2024
12	Ghadi and Toral (2022)	Transportation Research Part C (Review)	Review article	9.2	2022

Impact factor values are approximate and based on publicly reported 2023 figures from the respective journals. The list is ordered by relevance to the three research questions of

Chapter 2 rather than by year or by impact factor.

Appendix E

Scopus Search Strings used during the literature review

The Boolean search strings below were executed on Scopus and Web of Science between September and October 2025. Each string was restricted to peer reviewed journal articles and conference papers published between January 2019 and December 2025, and to the subject areas Engineering, Computer Science and Transportation.

S1. `TITLE-ABS-KEY ( "traffic management system" AND ( "YOLO" OR "deep learning" ) AND "adaptive" )`

S2. `TITLE-ABS-KEY ( "adaptive signal control" AND "computer vision" AND ( "cyclist" OR "motorcyclist" OR "bicycle" ) )`

S3. `TITLE-ABS-KEY ( ( "YOLOv8" OR "YOLOv5" OR "YOLOv7" ) AND "intersection" )`

S4. `TITLE-ABS-KEY ( "vulnerable road user" AND "signal" AND ( "priority" OR "phase" ) )`

S5. `TITLE-ABS-KEY ( "emergency vehicle" AND "adaptive traffic light" )`

S6. `TITLE-ABS-KEY ( "intelligent transportation" AND ( "ESP8266" OR "NodeMCU" OR "Arduino" ) )`

S7. `TITLE-ABS-KEY ( "reinforcement learning" AND "traffic signal" AND "intersection" )`

S8. `TITLE-ABS-KEY ( ( "Kampala" OR "Uganda" OR "sub-Saharan Africa" ) AND ( "traffic" OR "road safety" ) )`

S9. `TITLE-ABS-KEY ( ( "boda-boda" OR "commercial motorcycle" ) AND "road safety" )`

S10. `TITLE-ABS-KEY ( ( "bicycle path" OR "cycling infrastructure" ) AND "computer vision" )`

The keywords combined across the ten strings included: *traffic management, computer vision, YOLO, YOLOv8, adaptive signal, dynamic signal, intersection, cyclist, motorcyclist, boda boda, vulnerable road user, VRU, emergency vehicle priority, IoT, ESP8266, Arduino, Raspberry Pi, reinforcement learning, Kampala, Uganda, sub-Saharan Africa, road safety, congestion.*

Appendix F

Research alignment with specialization, SDG, NDPIV & Vision 2040

F.1 Alignment with Programme Specialisation

This research aligns with the **Artificial Intelligence and Robotics** specialisation track of the BSc Computer Science programme at Uganda Christian University. The project draws directly on elective courses in Machine Learning, Computer Vision and Embedded Systems. The YOLOv8n detector, an applied convolutional neural network, and the ESP8266 driven signal actuation represent, respectively, the AI and the robotics and embedded halves of that specialisation, brought together in a single integrated system.

F.2 Alignment with the Sustainable Development Goals

RoadWise contributes directly to three Sustainable Development Goals:

i. **SDG 3, Good Health and Well-Being.** By lowering the exposure of vulnerable road users at intersections, the blue light phase targets Target 3.6, which calls for a halving of global deaths and injuries from road traffic accidents by 2030.

ii. **SDG 9, Industry, Innovation and Infrastructure.** RoadWise demonstrates a locally designed, low cost adaptive signal infrastructure appropriate for low and middle income country contexts, supporting Target 9.1 on resilient infrastructure.

iii. **SDG 11, Sustainable Cities and Communities.** By reducing congestion and improving access for non-motorised road users, the system supports Target 11.2 on safe, affordable, accessible and sustainable transport for all, with special attention to vulnerable groups.

F.3 Alignment with the National Development Plan

Uganda's National Development Plan identifies urban transport infrastructure, data driven service delivery and the safety of vulnerable populations as priority programme areas. RoadWise operates inside all three. It is an urban transport intervention, it produces per class per junction traffic data that can feed the broader national data ecosystem, and it explicitly targets VRU safety.

F.4 Alignment with the Uganda Vision 2040

Vision 2040 positions Uganda as a knowledge based, innovation driven economy and calls explicitly for modern, efficient and inclusive urban mobility systems in Kampala and the other strategic cities. RoadWise is a concrete example of that ambition. It is locally designed, locally maintainable and explicitly inclusive of cyclists and boda boda riders, categories that the urban mobility pillar of Vision 2040 names by implication but that current infrastructure does not accommodate.

Appendix G

Algorithms, Code of interest

This appendix collects the key algorithms and abridged source listings referenced in Chapters 3 and 4. The full codebase lives in the project repository under the directories `scripts/`, `controller/` and `dashboard/`.

G.1 Adaptive Controller Pseudocode with Blue Light Phase

Listing G.1: Blue light cyclist priority scheduling

```
input: per class counts n_car[a,t], n_bike[a,t] for each approach a
params: alpha=1.0, beta=1.5, G_min=10, G_max=60, tau_B=3, G_B=8, T_cool=45
state: t_last_blue = -infinity

for each control tick t:
    for each approach a:
        D[a,t] = alpha * n_car[a,t] + beta * n_bike[a,t]
        G[a,t] = clip(G_min + k * D[a,t], G_min, G_max)

    max_bikes = max_over_a(n_bike[a,t])
    if max_bikes > tau_B and (t - t_last_blue) > T_cool:
        activate_blue_phase(duration=G_B, approach=argmax_a(n_bike[a,t]))
        t_last_blue = t
    else:
        run_normal_phase(G)
```

G.2 Dataset Preparation Script

Listing G.2: `scripts/prepare_dataset.py` (abridged)

```
# 70/15/15 deterministic split with a fixed random seed
random.seed(42)
```

```
images = sorted(glob("dataset/images/all/*.jpg"))
random.shuffle(images)
n = len(images)
train = images[:int(0.70 * n)]
val = images[int(0.70 * n):int(0.85 * n)]
test = images[int(0.85 * n):]
for src in train: link_or_copy(src, "dataset/images/train/")
for src in val: link_or_copy(src, "dataset/images/val/")
for src in test: link_or_copy(src, "dataset/images/test/")
# labels are linked or copied by matching basename in dataset/labels/all/
```

G.3 Real Time Inference Runner

Listing G.3: `run_detect.py` main loop (abridged)

```
model = YOLO(weights) # Ultralytics YOLOv8n
src = resolve_source(args) # 0 | file | http | rtsp
cap = cv2.VideoCapture(src)
while True:
    ok, frame = cap.read()
    if not ok: break
    results = model.predict(frame, conf=0.25, iou=0.45, imgsz=640)
    counts = {"car": 0, "bike": 0}
    for box in results[0].boxes:
        cls = model.names[int(box.cls)]
        counts[cls] += 1
        draw_box(frame, box, color=CLASS_COLOR[cls])
    overlay_counts(frame, counts)
    publish_counts_to_controller(counts) # Firebase
    cv2.imshow("RoadWise", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): break
```

G.4 Training Command

Listing G.4: Typical training invocation

```
python3 train.py \
    --data dataset/data.yaml \
    --model yolov8n.pt \
    --epochs 120 \
    --imgsz 640 \
    --batch 16
```

G.5 Serial Acknowledgement Protocol

Listing G.5: Dashboard to Arduino acknowledgement loop (abridged)

```
# Node.js backend
onSignalChange((newState) => {
  writeSerial(‘SET ${newState}\n‘);
  const ack = waitForSerialLine(timeoutMs=1000);
  if (ack === ‘ACK ${newState}‘) {
    firebase.set("/signal/state", newState);
    log("consistent");
  } else {
    raiseInconsistency(newState, ack);
  }
});

// Arduino firmware
void loop() {
  if (Serial.available()) {
    String cmd = Serial.readStringUntil(’\n’);
    if (cmd.startsWith("SET ")) {
      String state = cmd.substring(4);
      applyPhase(state); // drive RGB + blue LED pins
      Serial.print("ACK ");
      Serial.println(state);
    }
  }
}
```

Appendix H

Other useful figures

This appendix collects figures that support the main text of the report but that did not fit cleanly into any one chapter. They are presented here for examiners who wish to inspect the prototype more closely.

H.1 Three-Way and Four-Way Scale Model Junction Layouts

The two physical layouts used as backdrops for the scale model traffic simulation are reproduced in Figure H.1. Both layouts run against the same YOLOv8 detector, the same adaptive controller and the same blue light phase logic, which is what supports the scalability claim made in Chapter 4.

(a) Three-way junction

(b) Four-way junction

Figure H.1: Physical layouts used for the scale model traffic simulation.

H.2 System Architecture Diagram

The full four layer RoadWise architecture (Sensing and Hardware, Intelligence and Cloud, Action, Interfaces) is reproduced at a larger scale in Figure H.2, for examiners who want to follow the data flow end to end.

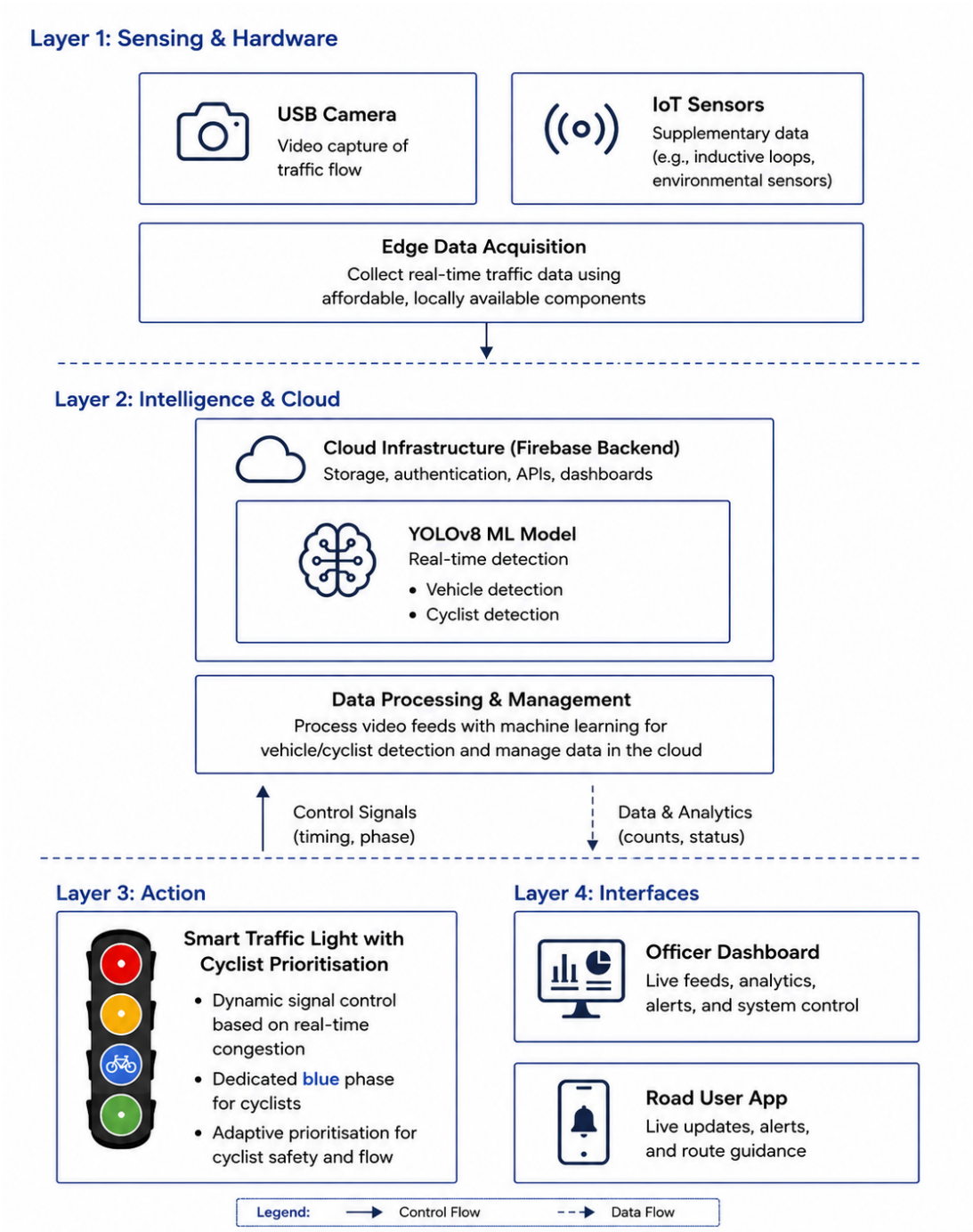


Figure H.2: The four layer RoadWise system architecture, reproduced at a larger scale than in Chapter 3.

H.3 Prototype in Operation

Figures H.3, H.4 and H.5 show additional views of the RoadWise prototype during live detection and control. Each view was captured on the physical scale model junction, with the YOLOv8 detector running on a laptop and the signal hardware driven by the Arduino Uno and NodeMCU boards.

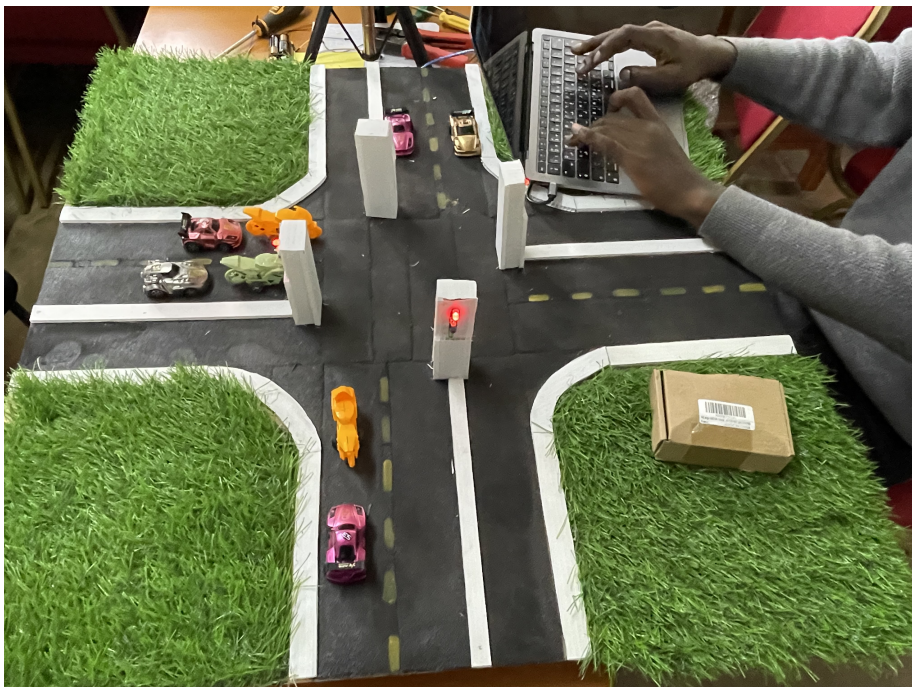


Figure H.3: Wide view of the prototype during a live run on the four-way junction.

Figure H.4: Detector output with bounding boxes overlaid on miniature vehicle and cyclist models.

Figure H.5: Closer view of the signal hardware, showing the RGB LEDs for the conventional phases and the blue LED for the cyclist priority phase.

H.4 Notes on Training Artefacts

The following artefacts are available in the project repository under `RoadwiseTMS/runs/detect/`: the raw and normalised confusion matrices, the F1 curve, the precision and recall curve, the precision curve, the recall curve, the labelled batch images and the predicted batch images. They are not reproduced here to save space, but should be consulted by examiners wishing to inspect training quality directly.

Appendix I

Research poster layout

The A0 poster used for the RoadWise stand at the UCU Department of Computing and Technology final year project exhibition is reproduced in Figure I.1. The poster summarises the problem, the objectives, the methodology, the prototype and the quantitative results on a single panel. A rendered A0 copy is archived with the team.

Figure I.1: The RoadWise A0 exhibition poster, signed by all three authors and by the supervisor.

The headline numbers on the poster are: mAP@0.5 of 0.977 on the validation split, 100% vehicle detection and 85% cyclist detection under varied lighting, up to 40% reduction in junction waiting time against a Webster baseline, 100% state consistency between

dashboard and hardware, and a per junction bill of materials of about UGX 792,000. The cyclist priority blue light phase is the novel contribution. It is absent from all twelve studies retained in the systematic literature review, as discussed in Chapter 2.