

**Software Engineering Department  
Braude College**

**Capstone Project Phase B – 61999**

25-1-D-23

PedXing : Prioritizing Pedestrians at Every Crossing

**Advisor:**

**Mr. Uzi Rosen**

**Authors :**

**Amni Abo Samra**

**Hamza Abu Nimer**

**git repository :** [**https://github.com/AmniAbo/Final-Project.git**](https://github.com/AmniAbo/Final-Project.git)

**Table of Contents**

[**1.Introduction**](#_pqli872axl8b) **3**

[**2. Hardware Implementation**](#_qnq1fmn5jacd) **4**

[2.1 Sensor Configuration and Placement](#_fz5abg239uus) 4

[2.2 Pedestrian Detection via Camera](#_rblw7tmdizcg) 4

[2.3 Microcontroller Setup](#_pworpg7ftbjw) 5

[2.4 Traffic Light Wiring and Control Logic](#_gvv5jo6n004a) 6

[2.5 3D Modeling and System Architecture](#_au8fz1lk8e6) 7

[**3. Machine Learning Model**](#_v1jxogoypjrt) **9**

[3.1 Dataset Description](#_5pncd2f5klv5) 9

[3.2 Machine Learning Overview and Model Selection](#_4azqdboswzxp) 12

[**4. Software Development**](#_pscabq7wo35e) **18**

[4.1 System Architecture Overview](#_sxqppvga9urc) 18

[4.2 Activity Diagram](#_eix9nisvxgk9) 20

[4.3 Web-Based Monitoring and Simulation Platform](#_dqk4kfefgrbn) 20

[**5. Results**](#_tbxjqv361nkf) **21**

[5.1 Scenarios](#_idjbffry3907) 21

[5.2 Evaluation of Smart System Behavior](#_i2mxnsj89lax) 23

[5.3 Prediction Accuracy and Model Evaluation](#_c0oyil5kpbco) 25

[5.4 Evaluation Against Project Goals](#_4hwygaru4rnl) 30

[**6. Challenges and Solutions**](#_axolco33ieu3) **31**

[**7. Future Work**](#_v6e0f3my1pg) **32**

[7.1 Multi-Intersection and Citywide Expansion](#_kg96mw6e70dp) 32

[7.2 Pedestrian Prioritization by Environment and Density](#_t7zvyuopq4ix) 32

[7.3 Integration with V2X (Vehicle-to-Everything) Technology](#_7sd5o0p2s9ew) 32

[7.4 Advanced Management and Analytics Interface for Municipal Authorities](#_64unjl6au74) 33

[**8. Conclusions**](#_457vff4v8azc) **33**

[**9. References**](#_7c3z74n6w7m3) **33**

# **1.Introduction**

Urban traffic management is a key part of designing smart and safe transportation systems. Traditional traffic light systems tend to focus mainly on vehicle flow, relying on fixed schedules or basic sensor inputs. However, pedestrians, arguably the most vulnerable users on the road, don’t always receive the attention they deserve, especially in challenging weather conditions like rain, extreme heat, or cold.

The system we developed introduces a new approach to intersection management: it actively prioritizes pedestrians. In other words, it gives them the right of way when certain conditions apply, particularly during extreme weather or when pedestrian density is high. This approach aligns with the principles of smart cities, which aim to balance safety, accessibility, and fair mobility for all road users.

As part of the project, we built a scaled model of an urban intersection featuring four vehicle lanes, each with its own traffic light, along with a dedicated pedestrian crossing and signal. The system uses ultrasonic sensors to detect vehicles in each lane and a smart AI-powered camera to identify and count pedestrians in real time. All this data is sent to a central microcontroller, which processes the information using a supervised machine learning algorithm, Random Forest, pre-trained on real-world traffic scenarios.

To ensure reliable and continuous operation, the system is connected to Firebase, a real-time cloud-based database platform. We also developed a dedicated web interface that includes a live dashboard showing real-time data from the intersection, such as which traffic light is currently green, how many pedestrians have been detected, and the status of each sensor. In addition, the website features a 3D simulation screen that visually illustrates the state of the intersection in real time, including vehicle movement, pedestrian crossing, and the traffic light states as determined by the system.

By combining IoT technologies, intelligent sensors, artificial intelligence, and a user-friendly web interface, we created a system that understands the human context, gives proper priority to pedestrians, and promotes safer, more inclusive, and more innovative urban mobility.

# 

# 

# **2. Hardware Implementation**

This chapter outlines the hardware components of the project and explains how each part contributes to building a smart, real-time traffic light control system. The system integrates sensors, an AI camera, microcontrollers, and LEDs to simulate a working intersection. Components communicate through Firebase and MQTT, enabling a fully synchronized control logic between software and hardware. All physical elements are assembled within a 3D-modeled intersection.

## **2.1 Sensor Configuration and Placement**

The traffic intersection uses four ultrasonic sensors, each installed at the entrance of a lane to detect approaching vehicles. These sensors calculate distance by emitting ultrasonic waves and measuring their echo time, allowing the system to determine car presence and flow density

All sensor data is sent to the M5Stack Core2 controller and uploaded to Firebase using the UiFlow environment, forming the foundation for intelligent decision-making by the traffic scheduling algorithm.

*Figure 1: Ultrasonic Sensor*

## **2.2 Pedestrian Detection via Camera**

To enable intelligent pedestrian prioritization, a UnitV2 AI camera is installed facing one pedestrian crossing of the intersection. The camera uses a Yolo\_20class trained to detect whether a pedestrian is present in the field of view. Once a pedestrian is detected, the information is uploaded to Firebase, where it influences the traffic light scheduling logic to ensure safe and efficient crossings.

The camera is securely mounted in a fixed position to maintain a consistent angle and view of the crosswalk, ensuring accurate detection in varied lighting conditions.



*Figure 2: UnitV2 AI camera*

## **2.3 Microcontroller Setup**

The system uses three microcontrollers: one M5Stack Core2 as the central controller, and two M5 Atom Lite units for managing the traffic lights. The Core2 gathers data from all sensors and the camera, then uploads it to Firebase. Based on scheduled decisions received from a Google Colab-based ML algorithm, it sends updated traffic light instructions to the Atom Lite devices via MQTT protocol.

Each Atom Lite is responsible for controlling the LEDs of two traffic lights. This decentralized system helps reduce latency and simplifies the physical wiring of each light cluster.



*Figure 3: M5Stack Core2*

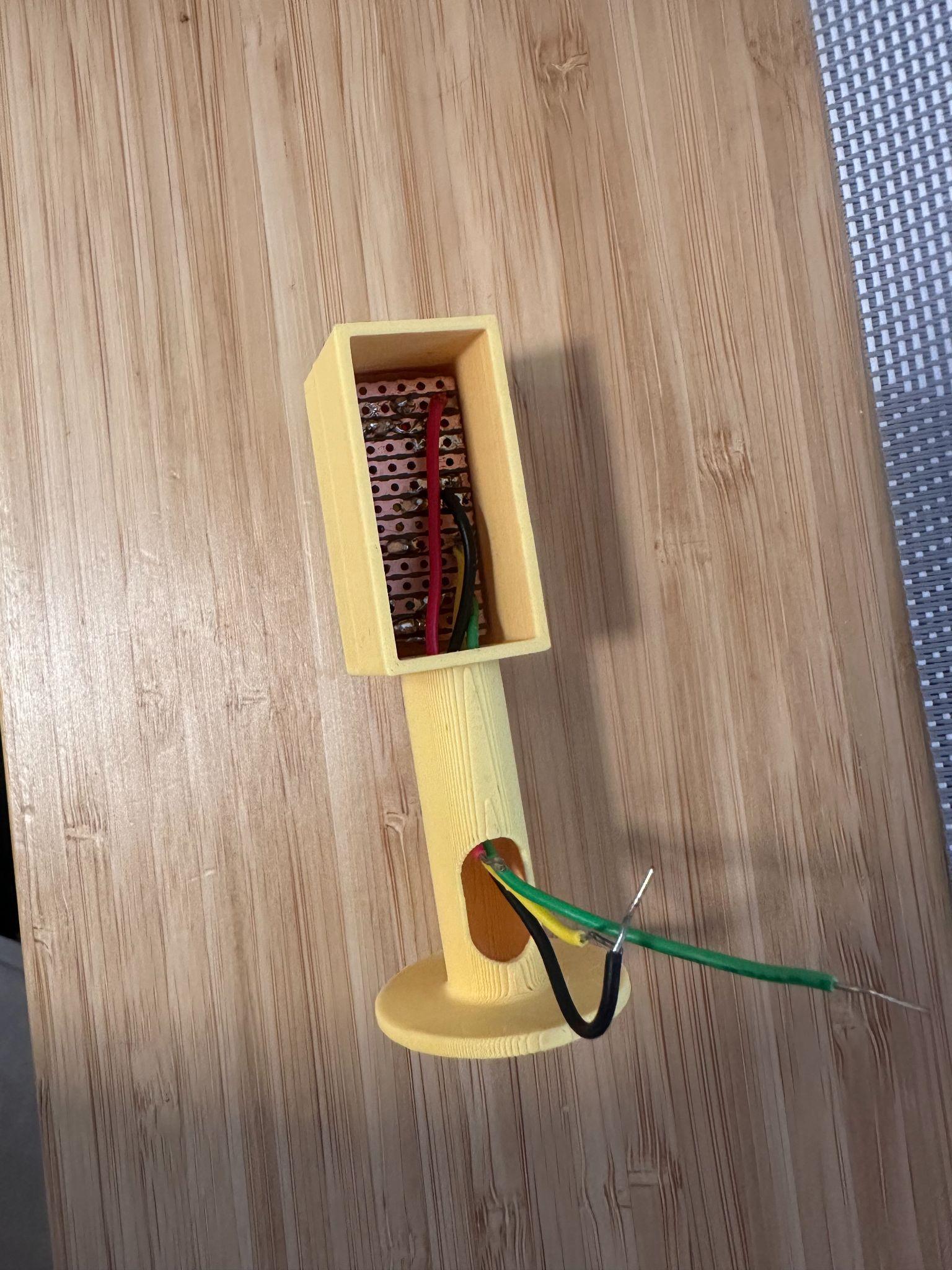


*Figure 4: atom lite*

## **2.4 Traffic Light Wiring and Control Logic**

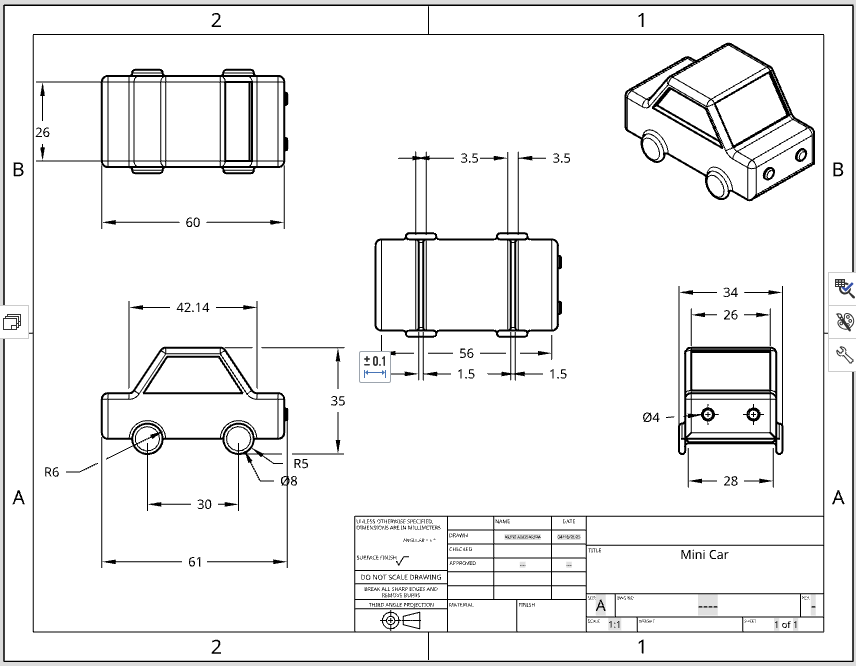
Traffic lights are simulated using LED modules, each consisting of red, yellow, and green LEDs mounted inside a 3D-printed traffic light housing. These lights are wired to the Atom Lite microcontrollers, which receive their color states from the central M5Stack Core2 based on decisions computed in Google Colab.

The control logic uses real-time data from sensors and camera inputs to determine the ideal light pattern for each intersection direction. These decisions are written into Firebase by the scheduling algorithm, extracted by the Core2, and pushed to the Atom Lites via MQTT. The lights update accordingly in real-time.



*Figure 5: 3D-Printed Traffic Light Module*

## **2.5 3D Modeling and System Architecture**

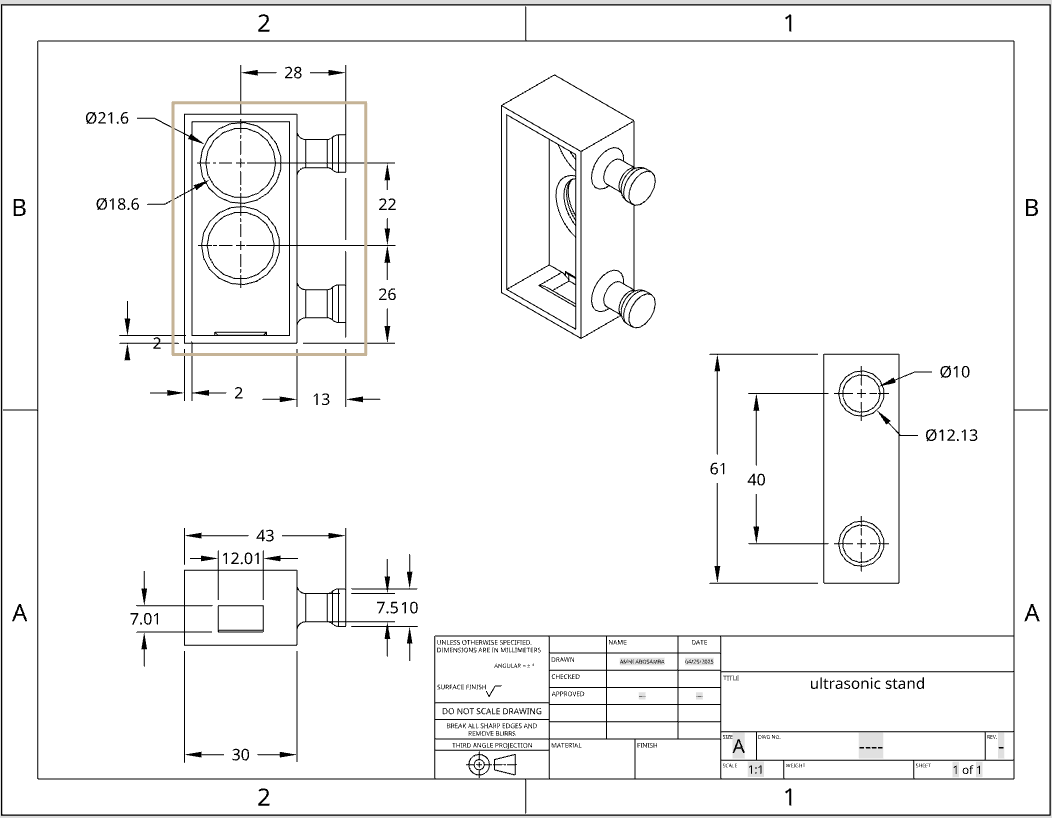
This section presents the architectural design and 3D modeling of the miniature intersection system. The intersection, including traffic light housings and physical lanes, was designed using Onshape and later fabricated using 3D printing. This allowed for a realistic, compact, and organized setup for mounting sensors, microcontrollers, and lights.

#### 

**• Mini Car Design**

A scaled-down vehicle was designed in Onshape and 3D printed to simulate car presence at the intersection. This car is used in conjunction with ultrasonic sensors to test vehicle detection and traffic flow logic.

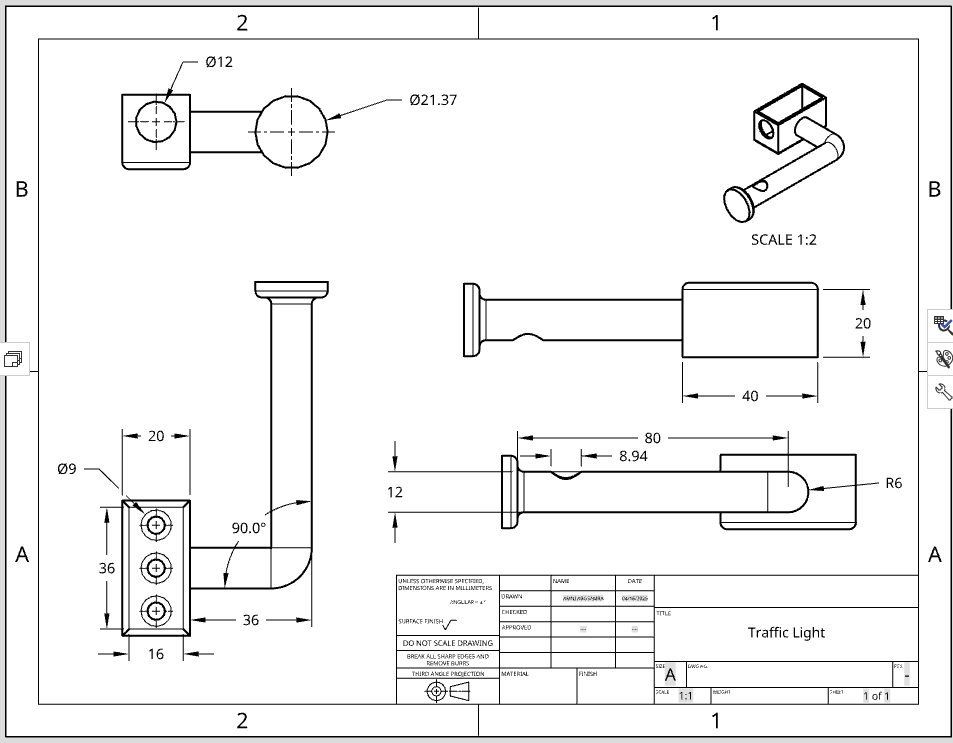
*Figure 6: Mini Car Design*



**• Ultrasonic Sensor Holder**

Custom holders were modeled to mount the ultrasonic sensors securely at accurate angles and distances. These holders ensure that the sensors face the road lanes directly and maintain stable readings without manual adjustments.

*Figure 7: Ultrasonic Sensor Holder Design*



**• Traffic Light Model**

A miniature traffic light casing was designed to house red, yellow, and green LEDs. The casing was printed and mounted upright using the Onshape dimensions. This component gives the setup a realistic traffic system appearance and protects the internal wiring.

*Figure 8: Traffic Light Casing Design*

# **3. Machine Learning Model**

## **3.1 Dataset Description**

To support the development of our smart intersection system, we created a custom dataset that simulates various traffic and pedestrian scenarios. Since we didn’t have access to real-world sensor data, we generated the dataset synthetically, modeling it after a scaled-down four-way urban intersection. Each data entry includes input from four virtual ultrasonic sensors (one per lane) to indicate whether a vehicle is present, as well as simulated pedestrian counts based on typical behaviors we modeled from the UnitV2 AI camera system.

We aimed to reflect both normal and edge-case situations, including smooth traffic flow and potentially dangerous or inefficient conditions. The dataset was structured in JSON format and processed using Google Colab for accessibility and ease of experimentation. In total, we generated approximately one million labeled samples. To simulate a balanced range of real-world conditions, we adopted a 50/50 distribution between “good” and “bad” scenarios, half of the data represents safe and efficient traffic patterns, while the other half captures problematic or unsafe situations such as congestion, poor timing, or conflicting signal states. This even split was designed to help the model learn both typical behavior and rare but critical edge cases. These labels were crucial for training and testing our machine learning model, enabling it to generalize effectively while remaining responsive to potential failures or anomalies.

**3.1.1 Data Features**

Each entry in the dataset contains a structured set of attributes:

* **Ultrasonic Sensor Readings (ultra1–ultra4)**: Binary indicators (1 = presence of a vehicle; 0 = absence) for each of the four intersection lanes.
* **Pedestrian Count (face)**: An integer representing the number of pedestrians detected in the crosswalk by the UnitV2 AI camera.
* **Temporal Information**: Includes time (formatted as HH:MM) and day (weekday label) to contextualize the conditions of each observation.
* **Environmental Conditions**: A boolean flag severe weather indicates the presence of rain, fog, or other adverse weather conditions.
* **Traffic Density**: A derived metric classified as "low", "medium", or "high", depending on the number of active ultrasonic sensors. A total of four active sensors denotes high traffic density, three indicates moderate, and two or fewer suggest low density.
* **Priority**: Based on contextual factors, this field identifies whether the current scheduling favors vehicles or pedestrians. If no pedestrians are detected (face = 0), the system prioritizes vehicle traffic. If five or more pedestrians are present or if severe weather conditions are reported, priority shifts to pedestrian traffic.
* **Traffic Light States**: Indicators for each of the four vehicle lanes (traffic\_light\_lane\_1 to traffic\_light\_lane\_4) and the pedestrian crossing (traffic\_light\_pedestrian) denote whether the light is red or green.
* **Label**: Each entry is categorized as either "good" or "bad" depending on the logical coherence and safety of the light configuration. This labeling supports both supervised training and evaluation of the system's decision-making accuracy.

**3.1.2 Scenario Generation Logic**

To build a dataset that would actually help train our traffic control model, we needed both “good” and “bad” examples. Since we didn’t have access to real-world data, we wrote two Python scripts to simulate different intersection situations, one script focused on safe, efficient behavior, and the other on mistakes or edge cases.

For the good scenarios, we followed a set of rules that match how a real intersection should behave. If there were at least two pedestrians or if the weather was bad, we gave priority to pedestrians, meaning the pedestrian light would turn green, and the vehicle lights stayed red (except for some side lanes that were allowed to turn green when safe). If there were cars in both directions, we used a simple toggling system that switched between horizontal and vertical flow every cycle. And if there were no cars or people detected at all, the lights stayed red.

For the bad scenarios, we intentionally added problems to reflect unsafe or inefficient behavior. Sometimes we gave green lights to both directions at once (which shouldn't happen), or we ignored pedestrians even when there were a lot of them waiting. In some cases, the traffic lights were just set randomly. These examples helped the model learn what *not* to do.

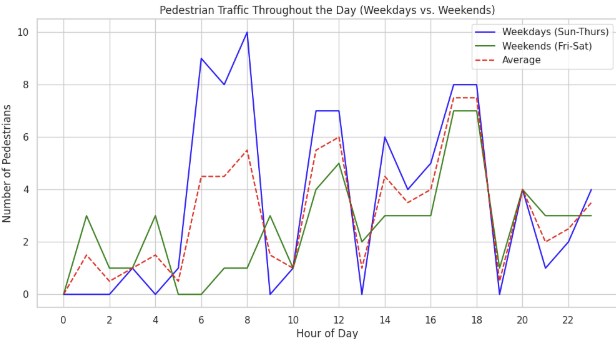
This setup gave us a balanced dataset of good and bad cases, and made sure the machine learning model saw a variety of real-world-like situations. The final output was saved as a JSON file and used for training and testing the system.

**3.1.3 Behavioral Modeling and Pedestrian Dynamics**

To enhance the realism of the dataset, particularly with regard to pedestrian traffic, the project integrates a behavioral model that simulates foot traffic density based on the hour of the day and the day of the week. The logic is encapsulated in a function that maps temporal patterns to expected pedestrian counts. Weekday (Sunday–Thursday) and weekend (Friday–Saturday) patterns are treated distinctly, reflecting typical urban pedestrian activity trends.

For example, during weekday morning rush hours (06:00–09:00) and evening rush hours (17:00–19:00), pedestrian counts are likely to range between 8 and 10 individuals. In contrast, weekend mornings may exhibit substantially lower pedestrian volumes, typically ranging from 0 to 3. Nighttime hours, regardless of the day, generally yield minimal pedestrian presence, unless overridden by specific conditions such as special events or weather anomalies.

This behavior-aware simulation mechanism allows the dataset to capture fluctuations in traffic demand across various periods, thereby enabling the scheduling algorithm to make temporally contextualized decisions.



*Figure 9: A diagram showing pedestrians traffic*

**3.1.4 Traffic Scheduling Logic**

We designed the traffic light scheduling system using a rule-based decision tree that reacts to both sensor data and basic behavioral rules. The goal was to keep things safe and efficient without making the logic too complex.

Here’s how it works:

**When Pedestrians Have Priority:**

* The pedestrian light turns green so they can cross safely.
* Lanes 2 and 4 (the side lanes) might also turn green, but only if there are cars waiting and it doesn’t interfere with pedestrian movement.
* All other lights stay red to avoid any potential conflicts.

**When Vehicles Have Priority:**

* The pedestrian light stays red.
* If there are cars in lanes 1 and 3 (the vertical lanes), and the horizontal lanes are empty, we allow vertical flow.
* If instead there’s traffic in lanes 2 and 4 (horizontal lanes), and lanes 1 and 3 are clear, we switch to horizontal flow.
* If all four lanes are busy, we alternate between vertical and horizontal in each cycle, so no direction is stuck for too long.
* And if no vehicles are detected at all, every light stays red, this saves power and helps keep the intersection clear.

This approach helped us build a predictable and explainable system that can react to different traffic and pedestrian situations without requiring machine learning to make every decision.

**3.1.5 Data Validation and Labeling**

To ensure logical consistency and support machine learning tasks, a correction mechanism is employed. This procedure scans each data entry, validates its coherence with traffic rules, and applies necessary corrections to traffic light states and derived attributes. The corrected entries are labeled as “good”, while entries with known inconsistencies either synthetically generated or sourced from randomized inputs are labeled as “bad”.

This dual-labeling system allows for the creation of a robust dataset that is well-suited for training classification models, evaluating traffic control strategies, and fine-tuning the decision logic of the scheduling algorithm.

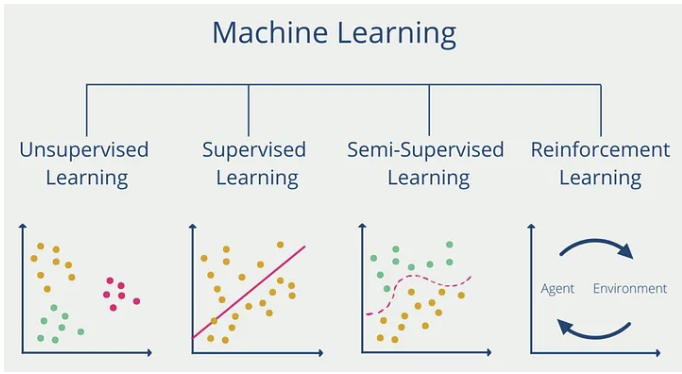
## **3.2 Machine Learning Overview and Model Selection**

Machine learning (ML) is a branch of artificial intelligence (AI) that enables systems to learn from data and make predictions or decisions without being explicitly programmed. In the context of this project, ML was used to predict traffic light states, determine lane priorities, and evaluate the quality of traffic scenarios. Selecting the right type of learning approach is critical for achieving accurate and reliable results.

There are three main types of machine learning:

**Types of Machine Learning:**

| **Type** | **Description** | **Examples** |
| --- | --- | --- |
| **Supervised Learning** | Trained on labeled data with known outcomes. | Random Forest, SVM, Decision Trees |
| **Unsupervised Learning** | Finds patterns in unlabeled data. | K-Means, PCA, DBSCAN |
| **Reinforcement Learning** | Learns through feedback and rewards. | Q-Learning, DQN |



*Figure 10: Main Types of Machine Learning Paradigms[2]*

**Why Supervised Learning?**

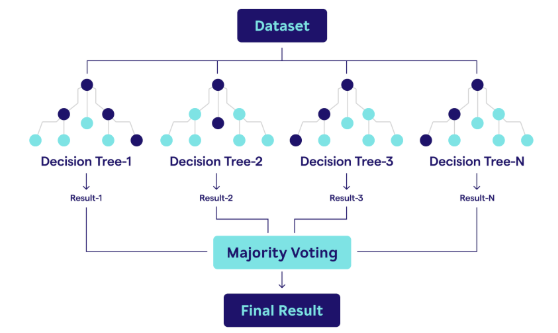
Supervised learning[2] was chosen for this project because it aligns closely with the structure of our dataset and the system’s objectives. The data is fully labeled, including traffic decisions and scenario classifications, which makes it well-suited for training models that learn from known outcomes.

This approach enables the model to map sensor inputs such as vehicle detection, pedestrian presence, and weather conditions to appropriate traffic control decisions. Supervised learning also provides clear evaluation metrics like accuracy and precision, which are essential for validating performance.

Additionally, the system requires multi-output predictions, including lane priorities and traffic light states. Supervised methods handle this effectively. Among the algorithms considered, Random Forest was selected for its reliability, flexibility, and strong performance with structured data. Further details about its implementation are provided in the following section.

**random forest**

The Random Forest algorithm[1] is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree is trained on a random subset of the data, and predictions are made by majority voting, making the model robust to noise and overfitting. This makes Random Forest particularly powerful for classification tasks involving noisy or synthetic datasets



*Figure 11: Architecture of the Random Forest Algorithm*

To enable intelligent decision-making within the traffic scheduling system, a machine learning model based on the Random Forest algorithm was developed and trained using the labeled dataset. This model is capable of handling multiple outputs simultaneously, making it suitable for predicting various traffic control attributes such as traffic light states, lane priority, and the overall quality of a traffic scenario.

The implementation begins by reading and processing data from a JSON file, which includes synthetic input features such as ultrasonic sensor readings (ultra1 to ultra4), pedestrian detection (face), and environmental conditions (severe weather). Categorical variables like traffic light states and priorities are numerically encoded to make the dataset suitable for model training.

The model employs a MultiOutputClassifier wrapping a RandomForestClassifier, enabling it to predict multiple target values in parallel. These include:

* Priority (whether the system prioritizes vehicles or pedestrians)
* Traffic light states for each lane and for pedestrians
* The scenario label (either "good" or "bad")

The dataset is split into training and testing subsets to evaluate performance. After training, the model's predictions are compared with actual labels, and individual accuracy scores are computed for each target variable. A detailed classification report is also generated for one of the output variables (e.g., traffic\_light\_lane\_1) to assess the model's performance in terms of precision, recall, and F1-score.

Finally, the trained model is saved as traffic\_model.pkl for future inference. The test predictions are organized in a DataFrame to allow side-by-side comparison of input features, actual labels, and predicted outcomes. For validation purposes, entries predicted as “good” are highlighted and displayed, showcasing scenarios the model has deemed logically correct and efficient.

This Random Forest model forms the core of the traffic decision logic, providing the ability to learn from past patterns and make real-time predictions to optimize traffic flow and enhance safety at intersections.

**3.3.6 Rationale for Choosing Random Forest**

The Random Forest algorithm was selected as the core of our traffic scheduling model due to its strong performance in handling complex, noisy, and high-dimensional datasets characteristics that define our sensor-driven traffic environment.

**Random Forest** is an ensemble learning method that builds a diverse set of decision trees, each trained on a random subset of the data and feature space. This technique, known as bootstrap aggregation (bagging), reduces variance and helps prevent overfitting. By aggregating the predictions from multiple uncorrelated trees, the model achieves higher accuracy and generalization.

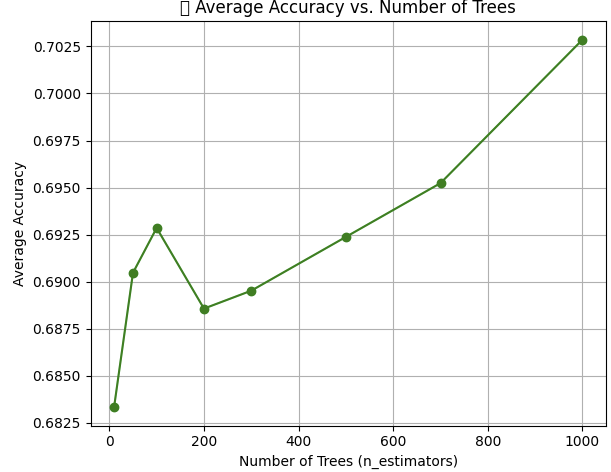
Key advantages relevant to our system include:

* **Multi-Output Integration**: Random Forest can be effectively combined with a MultiOutputClassifier, allowing simultaneous prediction of multiple dependent variables such as traffic light states for each lane, pedestrian signals, lane priority, and overall scenario assessment. This aligns perfectly with the real-time, multi-faceted nature of intersection control.
* **Robustness to Noise and Irregularity**: Traffic data, especially from urban sensors, can be noisy, incomplete, or inconsistent. Random Forest is inherently resilient to such imperfections, as individual noisy data points have limited influence on the final ensemble output.
* **Modeling Non-Linear Relationships**: Vehicle movement patterns and pedestrian flow often exhibit non-linear behaviors that cannot be captured well by linear models. Random Forest supports complex decision boundaries, enabling more realistic modeling of traffic dynamics.
* **Support for Mixed Data Types**: Our dataset includes both numerical features (e.g., ultrasonic sensor readings) and categorical values (e.g., light states). Random Forest handles these without requiring extensive preprocessing, such as normalization or one-hot encoding.
* **Feature Importance and Interpretability**: In addition to predictive power, Random Forest provides insight into feature importance, helping us identify which sensor inputs most influence the system’s decisions critical for debugging, optimization, and transparency.

Overall, Random Forest offers a powerful combination of accuracy, robustness, and operational simplicity. These qualities make it an ideal choice for driving intelligent, real-time decisions in a dynamic traffic control environment.

**3.3.7 Model Tuning: Number of Trees Comparison**

An important hyperparameter in a Random Forest model is the number of decision trees (n\_estimators). To determine the optimal number of trees for the Random Forest model, several training runs were conducted with increasing values: 10, 50, 100, 200, 500, and 1000. Each run evaluated two main metrics: prediction accuracy and training time.



*Figure 12: Training Time vs. Tree Count Figure 13: Accuracy vs. Tree Count*

| **Number of Trees (n\_estimators)** | **Average Accuracy** | **Training Time (seconds)** |
| --- | --- | --- |
| 10 | 0.683 | 0.6 |
| 50 | 0.690 | 1.5 |
| 100 | 0.693 | 2.2 |
| 200 | 0.688 | 4.1 |
| 500 | 0.692 | 7.6 |
| 750 | 0.695 | 10.4 |
| 1000 | 0.703 | 14.3 |

**Average Accuracy vs. Number of Trees**

the average accuracy across all predicted outputs initially increases with the number of trees. Accuracy improves notably between 10 and 100 trees, peaking slightly at 100, followed by minor fluctuations. While accuracy continues to rise slowly beyond 500 trees, the gains are marginal.

**Training Time vs. Number of Trees**

a linear increase in training time is clearly visible as the number of trees grows. The time required increases significantly past 500 trees, reaching over 14 seconds at 1000 trees.

Although accuracy improves slightly beyond 500 trees, the increased training time is not justified by the minimal performance gain. Therefore, 100 trees was chosen as the final configuration, offering a balanced trade-off between model performance and computational efficiency.

# **4. Software Development**

This chapter describes the software components used to build the smart traffic control system, including the system architecture design, real-time database integration, sensor data processing, and the development of a user interface for information display. In addition, an interactive visual interface was developed, allowing users to view a live simulation of the intersection, including the movement of vehicles and pedestrians based on real-time data received from the sensors and the camera.

## **4.1 System Architecture Overview**

The system architecture consists of several integrated layers that coordinate hardware components, communication protocols, cloud infrastructure, and intelligent control logic. Each layer has a distinct role in collecting data, processing it, and controlling the traffic lights accordingly. The layered structure ensures modularity, scalability, and real-time responsiveness across the entire intersection system.

1. Device Layer

* M5Stack Core2 – Serves as the central controller. It receives input from the sensors and AI camera, sends the data to Firebase, and transmits control commands to the Atom Lite units via MQTT.
* Ultrasonic Sensors – Detect the presence of vehicles at the entrance to each lane.
* AI Camera (UnitV2) – Identifies pedestrians and uploads detection results to Firebase.
* Atom Lite – Controls the physical traffic lights (LEDs) based on commands received from the M5Stack Core2.

2. Network Layer

* REST API – Used for sending sensor and camera data from the M5Stack Core2 and Atom lite to Firebase.
* MQTT Protocol – Used by the Core2 to send real-time control commands to the Atom Lite units.

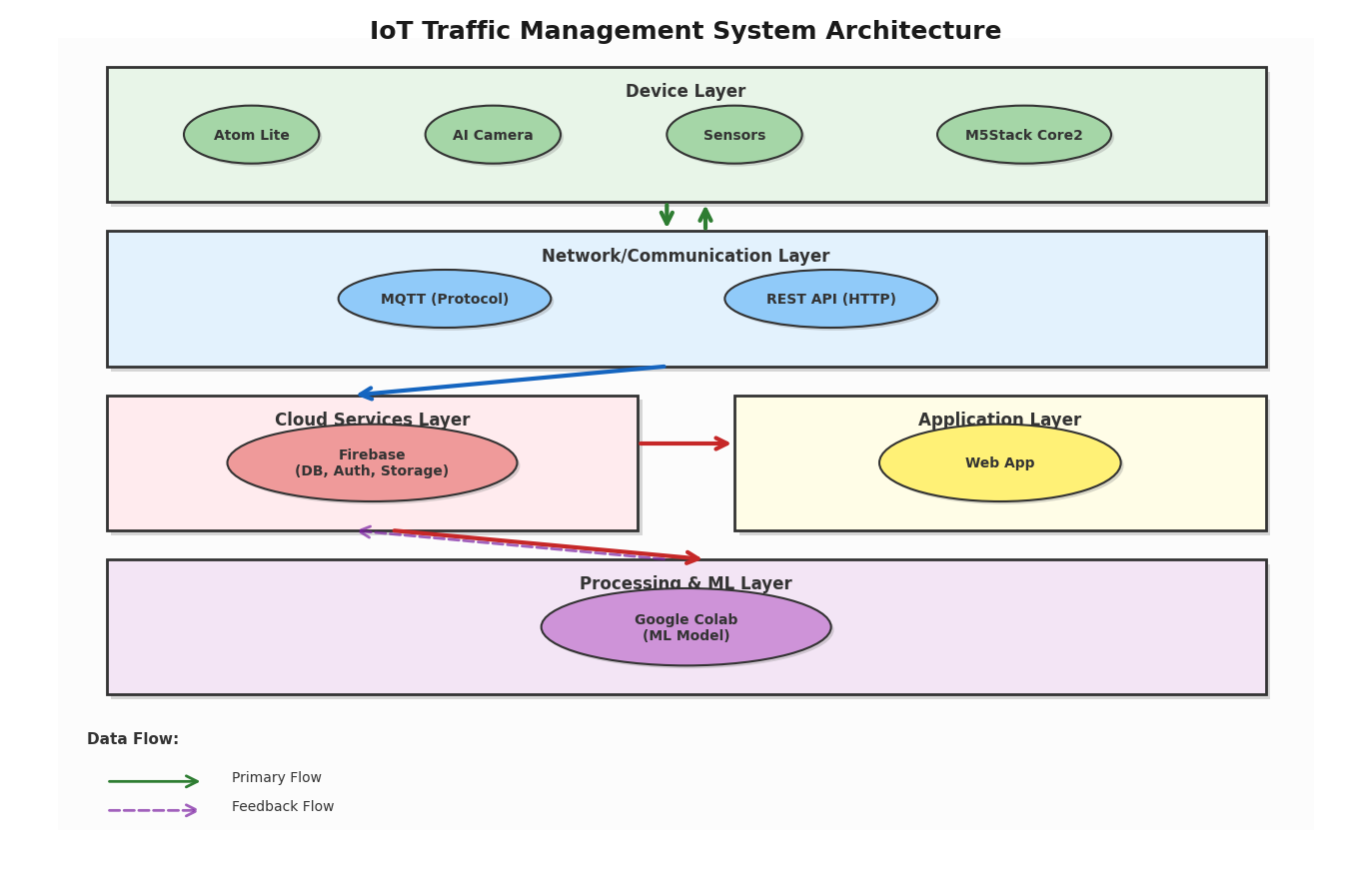
3. Cloud Layer

* Firebase – Functions as a real-time database and middleware, bridging between hardware components, the machine learning model, and the web application.

4. Processing and Machine Learning Layer

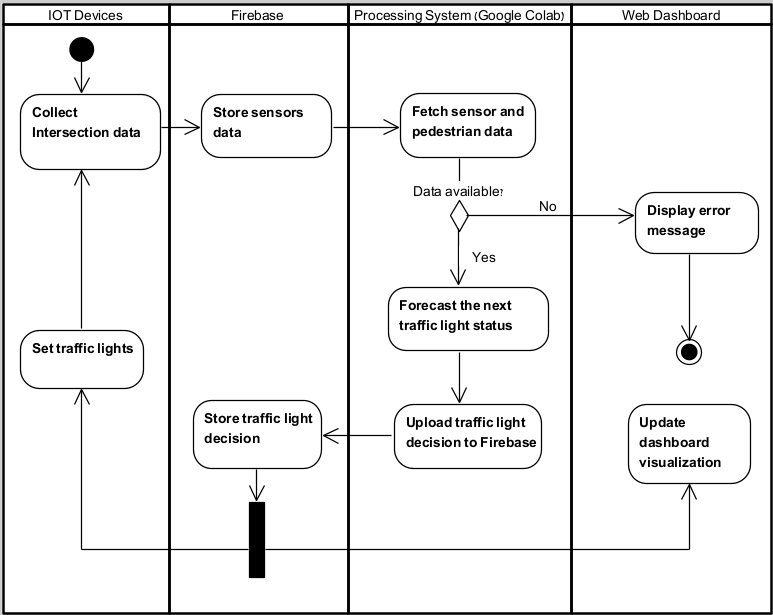
* Google Colab – Executes the ML model that calculates the optimal traffic light configuration and writes the decision back to Firebase.

5. Application Layer

* Web Application – Displays the real-time status of traffic lights, pedestrian activity, and sensor readings. The interface also includes a 3D simulation of the intersection and a summary of system decisions.

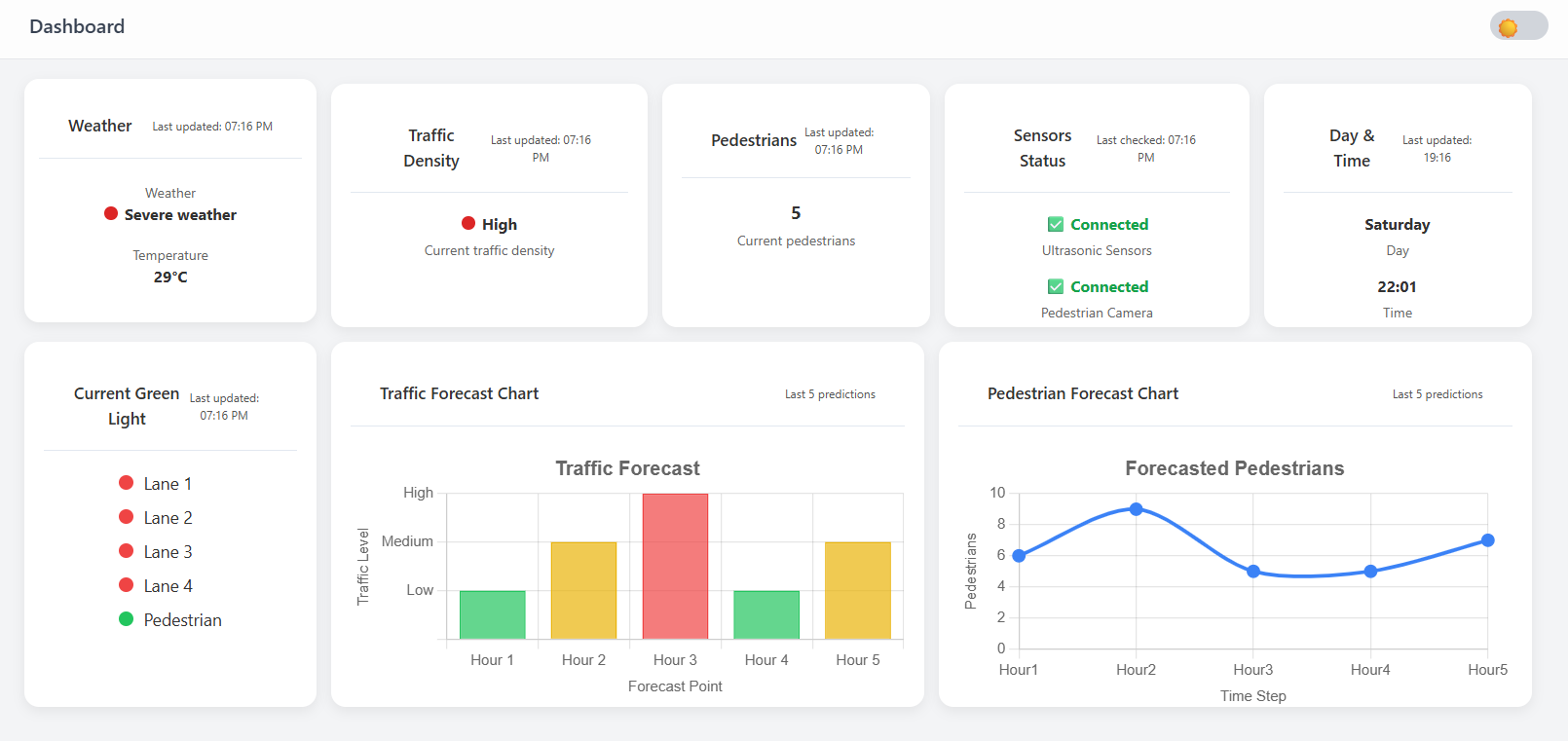
*Figure 14: System Architecture*

## **4.2 Activity Diagram**

*Figure 15: Activity Diagram*

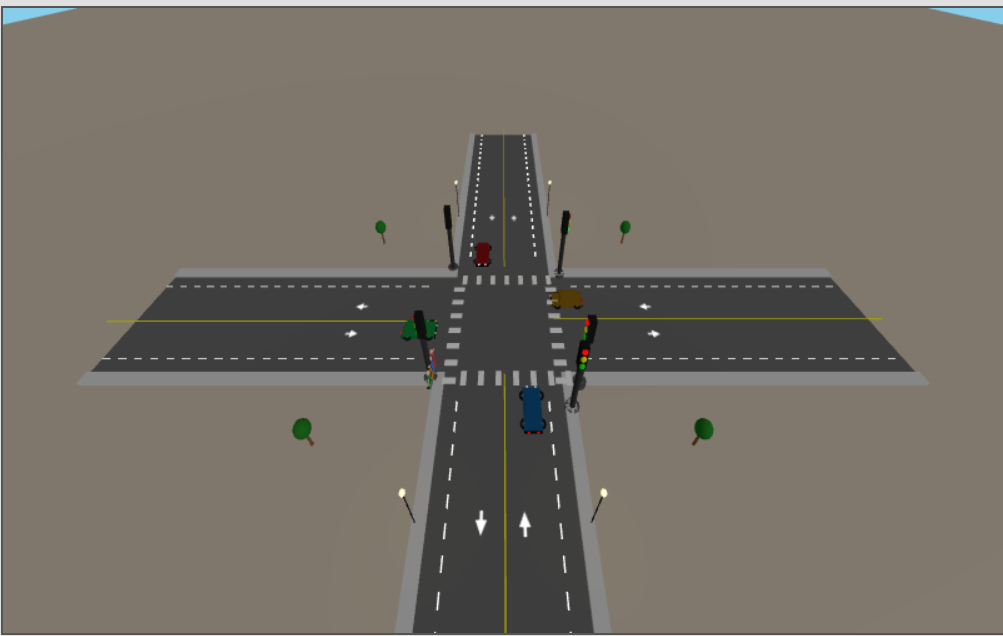
## **4.3 Web-Based Monitoring and Simulation Platform**

As part of the project, an interactive web platform was developed for real-time intersection monitoring and system decision analysis. Its main component is a live dashboard connected to Firebase, which updates automatically. The dashboard displays key data such as weather, traffic congestion, pedestrian count, traffic light status, and sensor readings, all shown with intuitive visuals using colors and symbols for quick recognition of critical events.



*Figure 16: Live dashboard of the traffic light system*

Alongside the dashboard, a 3D simulation of the intersection was developed to visualize real-time traffic, pedestrian movement, and light changes based on sensor data and system decisions. It serves as a tool for testing performance and demonstrating various scenarios..



*Figure 17: 3D Simulation of the Intersection in the Web Platform*

# **5. Results**

In this section, we evaluate the performance of our smart traffic light system. To do so, we construct a set of simulated scenarios that reflect various intersection states, aiming to assess the system's ability to adapt dynamically to real-time conditions. The evaluation includes data analysis, contextual match rate measurement, and tracking the number of traffic light activations during the simulation run for each defined scenario. Additionally, a separate confusion matrix is analyzed for each traffic light to evaluate the model’s accuracy in selecting the appropriate action.

## **5.1 Scenarios**

The scenarios constructed for system evaluation include various combinations of traffic and environmental conditions at the intersection. Each scenario is defined by key parameters: presence or absence of vehicles across the four road lanes, the number of pedestrians near the crosswalk, and the weather condition (normal or severe). These scenarios were carefully selected to represent a wide range of situations from typical traffic to extreme cases in order to demonstrate the system’s ability to respond and adapt its decisions in real time based on input conditions.

| Scenario | ultra1 | ultra2 | ultra3 | ultra4 | face | severe\_weather |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 2 | FALSE |
| 2 | 0 | 0 | 1 | 0 | 4 | FALSE |
| 3 | 1 | 0 | 0 | 0 | 6 | FALSE |
| 4 | 1 | 0 | 1 | 0 | 1 | TRUE |
| 5 | 0 | 1 | 0 | 0 | 3 | FALSE |
| 6 | 1 | 0 | 1 | 0 | 5 | TRUE |
| 7 | 0 | 0 | 0 | 1 | 0 | FALSE |
| 8 | 1 | 0 | 1 | 0 | 2 | TRUE |
| 9 | 1 | 0 | 0 | 0 | 4 | FALSE |
| 10 | 0 | 1 | 1 | 0 | 6 | FALSE |
| 11 | 0 | 0 | 0 | 0 | 1 | FALSE |
| 12 | 1 | 0 | 1 | 0 | 3 | TRUE |
| 13 | 0 | 0 | 0 | 0 | 5 | FALSE |
| 14 | 0 | 0 | 1 | 1 | 0 | FALSE |
| 15 | 1 | 1 | 0 | 0 | 2 | FALSE |
| 16 | 1 | 0 | 1 | 0 | 4 | TRUE |
| 17 | 0 | 0 | 0 | 0 | 6 | FALSE |
| 18 | 1 | 0 | 1 | 0 | 1 | TRUE |
| 19 | 0 | 0 | 0 | 0 | 3 | FALSE |
| 20 | 1 | 1 | 1 | 0 | 5 | TRUE |
| 21 | 0 | 0 | 0 | 0 | 2 | FALSE |
| 22 | 0 | 0 | 1 | 0 | 4 | FALSE |
| 23 | 1 | 0 | 0 | 0 | 6 | FALSE |
| 24 | 1 | 0 | 1 | 0 | 1 | TRUE |
| 25 | 0 | 1 | 0 | 0 | 3 | FALSE |
| 26 | 1 | 0 | 1 | 0 | 5 | TRUE |
| 27 | 0 | 0 | 0 | 1 | 0 | FALSE |
| 28 | 1 | 0 | 1 | 0 | 2 | TRUE |
| 29 | 1 | 1 | 0 | 0 | 2 | FALSE |
| 30 | 1 | 0 | 1 | 0 | 4 | TRUE |
| 31 | 0 | 0 | 0 | 0 | 6 | FALSE |
| 32 | 1 | 0 | 1 | 0 | 1 | TRUE |
| 33 | 0 | 0 | 0 | 0 | 3 | FALSE |
| 34 | 1 | 1 | 1 | 0 | 5 | TRUE |
| 35 | 1 | 1 | 1 | 1 | 10 | TRUE |
| 36 | 0 | 0 | 0 | 0 | 10 | TRUE |
| 37 | 0 | 0 | 0 | 0 | 0 | TRUE |
| 38 | 1 | 1 | 1 | 1 | 0 | FALSE |
| 39 | 0 | 1 | 0 | 1 | 5 | FALSE |
| 40 | 1 | 0 | 0 | 0 | 10 | TRUE |

## **5.2 Evaluation of Smart System Behavior**

This section presents two key metrics used to evaluate the behavior of the smart traffic light system during the simulation.

The first metric is the contextual match rate, which measures how accurately the system's decisions are aligned with real-time traffic and pedestrian conditions. A high match rate indicates effective adaptation to dynamic inputs.

The second metric is the number of activations per traffic light, indicating how often a green light was assigned to each lane. This analysis reflects the system’s prioritization logic, particularly its responsiveness to pedestrian congestion (with the pedestrian light receiving the highest number of activations).

The combination of these two metrics allows us to assess whether the system meets the desired behavioral requirements and to examine its ability to respond in a precise, flexible, and context-aware manner to changing real-time conditions.

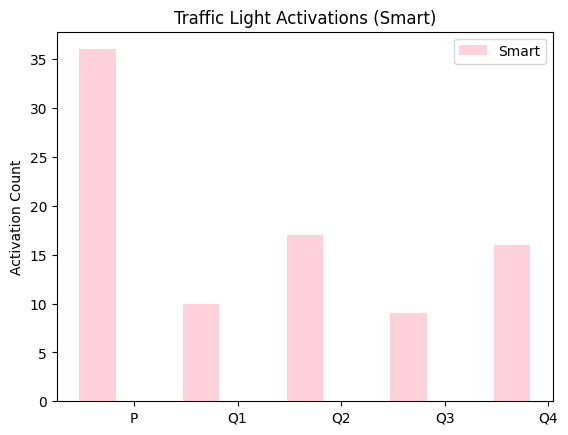
The following figure presents the contextual match rate achieved by the smart system during simulation.



*Figure 18: Contextual Match Rate*

As shown in the chart, the smart system achieved a contextual match rate of approximately 87%, indicating that in most cases, the selected traffic light action aligned well with the actual traffic and pedestrian conditions. This high percentage reflects the system’s ability to respond accurately and appropriately to dynamic, real-time inputs, rather than following a fixed schedule.

The following chart shows how many times each traffic light was activated by the smart system during the simulation run.



*Figure 19: Traffic Light Activations*

The chart shows that the pedestrian traffic light (P) was activated more than any other light during the simulation. This finding indicates that the smart system tends to prioritize pedestrians, especially when there is crowding at the crosswalk and particularly under severe weather conditions. Notably, 91.67% of the system's decisions were labeled as “good.” The other traffic lights were activated based on actual needs in each lane, using data collected from sensors and the camera.

## **5.3 Prediction Accuracy and Model Evaluation**

To assess the effectiveness of our supervised learning model (Random Forest), we evaluated its performance across all predicted outputs using a held-out test set. The results demonstrate strong accuracy for most traffic control attributes, particularly those concerning vehicle lanes, while also highlighting areas for future optimization.

**Prediction Accuracy by Output Variable:**

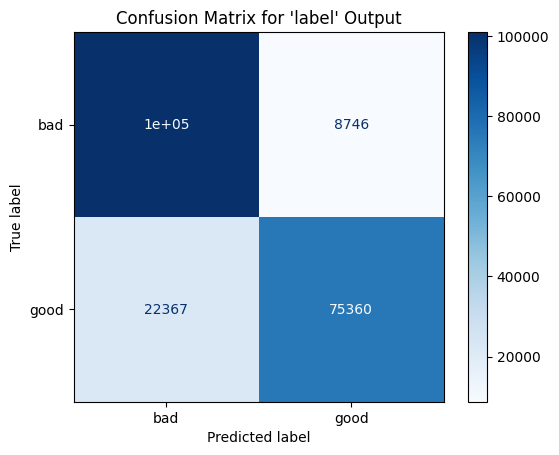
| **Output Variable** | **Accuracy** |
| --- | --- |
| priority | 0.85 |
| traffic\_light\_lane\_1 | 0.93 |
| traffic\_light\_lane\_2 | 0.92 |
| traffic\_light\_lane\_3 | 0.93 |
| traffic\_light\_lane\_4 | 0.92 |
| traffic\_light\_pedestrian | 0.85 |
| label (good/bad) | 0.85 |

### 

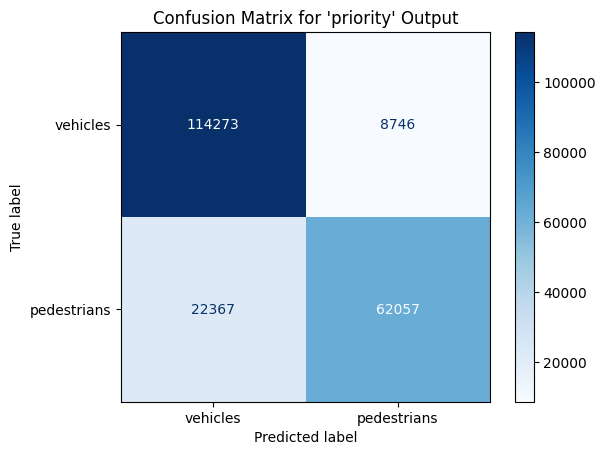
### **Model Evaluation Metrics**

| **Metric** | **Value** | **What It Means** |
| --- | --- | --- |
| Precision | 0.9297 | When the model decides to activate something (like a green light or giving priority), it's right about 93% of the time. This means it makes very few false alarms. |
| Recall | 0.9171 | Out of all the correct answers that should have been found, the model catches around 91%. So it's pretty good at not missing important signals. |
| F1-Score | 0.9154 | This is a combination of precision and recall. A score around 91% means the model is generally balanced, it's accurate and consistent, but there’s still a bit of room to improve. |

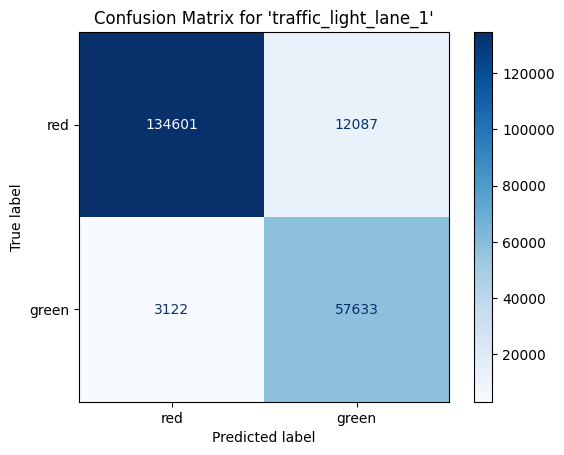
**Confusion Matrices for Features :**

****

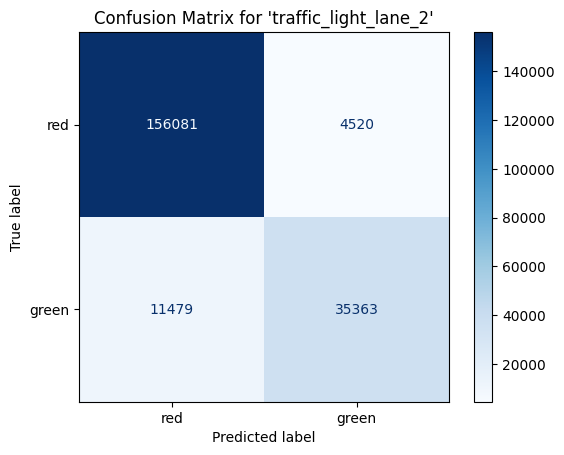
*Figure 20: Confusion Matrix for the ‘label’ Output*

****

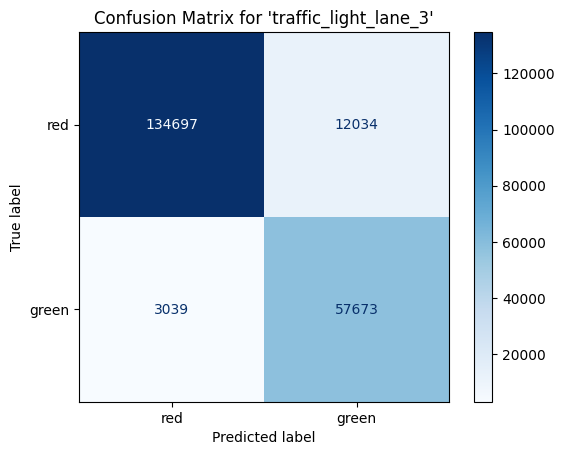
*Figure 21: Confusion Matrix for the priority Output*

****

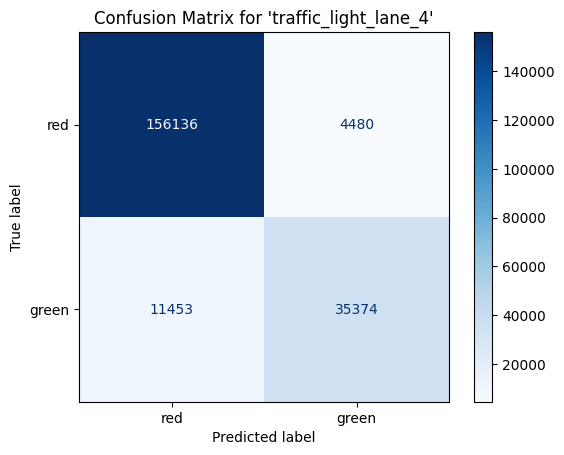
*Figure 22: Confusion Matrix for traffic light lane 1 Output*

****

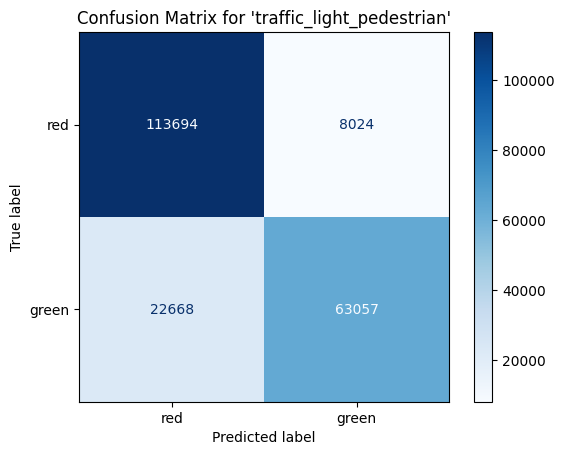
*Figure 23: Confusion Matrix for traffic light lane 2 Output*

****

*Figure 24: Confusion Matrix for traffic light lane 3 Output*

****

*Figure 25: Confusion Matrix for traffic light lane 4 Output*

****

*Figure 26: Confusion Matrix for pedestrian traffic light Output*

## 

## **5.4 Evaluation Against Project Goals**

The primary objective of the project was to develop a smart traffic light system that dynamically prioritizes pedestrians, particularly in situations involving environmental risks or traffic congestion that could endanger their safety or well-being. The system was tested in scenarios where pedestrians might be exposed to extreme weather conditions or gather in large numbers at the intersection, with the goal of evaluating its ability to respond appropriately to such contexts.

To assess its performance, 40 controlled simulation scenarios were executed, covering a wide range of conditions including variable weather, traffic congestion, and pedestrian presence. The results showed that the smart system aligned its decisions with contextual needs in 87% of the scenarios, demonstrating a strong capability to detect and respond to situations where pedestrian prioritization is genuinely required.

In addition, the pedestrian light ("P") was activated 36 times by the smart system, highlighting its sensitivity to real pedestrian needs. Accordingly, 91.67% of the scenarios labeled by the system were classified as "good", reflecting decisions that were both safe and effective.

The model’s overall performance showed high accuracy across all output variables. For example:

* 85% accuracy in predicting the need for pedestrian prioritization
* 93% accuracy in determining the state of the traffic light for lane 1
* An average F1 score of 91.5%, indicating a strong balance between precision and recall

It is important to emphasize that pedestrian prioritization did not come at the expense of proper vehicle traffic management. The underlying model demonstrated high accuracy across all output variables, including those related to vehicle signal control. This indicates the system’s ability to balance pedestrian safety with the efficient and continuous flow of vehicle traffic at the intersection.

# **6. Challenges and Solutions**

Throughout the project, we encountered various challenges, both on the hardware and software sides. Each challenge required in-depth analysis and adjustments during development. Below are some of the key challenges we faced and how we addressed them:

1- Lack of Prior Experience with Hardware Components and Sensors

When we started the project, we had no prior hands-on experience working with IoT devices, ultrasonic sensors, or object-detection cameras. As software engineering students, most of our training focused on programming rather than on physical system components. Nevertheless, we dedicated significant time to self-learning through technical documentation, basic electronic connections, and understanding how to interface with physical devices. Through trial and error, reading reliable online resources, and consulting with our project advisor, we successfully managed to operate all hardware components and integrate them reliably into the system.

2-Creating a Realistic Simulation

Developing a simulation that looked realistic while still running smoothly in a web browser was a significant challenge. We had to learn how to work with Three.js, optimize the number of objects rendered, and create flowing yet simple animations, all while maintaining fast loading times and high performance.

3-Finding Relevant Research on Pedestrian-Priority Systems

During the initial research phase, we encountered a significant challenge in finding previous studies or projects with objectives similar to ours: a smart traffic light system that dynamically prioritizes pedestrians. Most existing research focused on optimizing vehicle flow or detecting pedestrians for alert purposes only, with very few works addressing the prioritization of pedestrians as a preferred group within an intelligent traffic control system. To overcome this challenge, we expanded our search to include related topics such as intersection safety, sensor-based traffic management systems, and the integration of pedestrians in smart urban transportation planning. From these diverse areas, we extracted key design principles, performance metrics, and evaluation methods, which helped us develop an original and innovative methodology tailored to the specific needs of our project.

# **7. Future Work**

The system developed in this project lays the groundwork for a dynamic and scalable smart traffic light infrastructure aimed at improving pedestrian safety in urban areas. Future development will focus on expanding the system’s scope, enhancing real-time decision-making, and integrating advanced communication technologies.

## **7.1 Multi-Intersection and Citywide Expansion**

The next stage will transition from a single-intersection setup to a citywide network of coordinated smart intersections. These intersections will share real-time data, such as pedestrian density and local weather conditions with nearby junctions or a central unit. This will enable context-aware decision-making across the city. To support this, a scalable data architecture and a secure, high-speed communication framework will be required.

## **7.2 Pedestrian Prioritization by Environment and Density**

Currently, the system detects harsh weather at a single intersection and adjusts signals accordingly. Future development will extend this capability to a network of intersections, incorporating demographic factors such as the presence of elderly people or children. An adaptive algorithm, combined with external data sources like real-time weather feeds and smart city infrastructure, will help determine when and where pedestrian priority should be applied..

## **7.3 Integration with V2X (Vehicle-to-Everything) Technology**

To improve road safety and coordination, the system will incorporate V2X (Vehicle-to-Everything) communication. This will enable real-time alerts for vehicles about nearby pedestrian activity or upcoming changes in traffic light status., allow dynamic speed adjustments, and support safer interactions for autonomous vehicles. Pedestrian smartphones or wearables will also send presence signals to enhance detection under low-visibility conditions.

## **7.4 Advanced Management and Analytics Interface for Municipal Authorities**

A centralized control interface will be developed for municipal traffic operators, providing real-time monitoring of smart intersections, historical traffic analysis, and manual override options for emergencies. It will also function as a strategic planning tool, offering visual insights and customizable reports to support the development of safer, more accessible urban infrastructure for pedestrians.

# **8. Conclusions**

The system developed in this project demonstrated strong performance in identifying traffic situations that require intelligent, context-aware responses. The model achieved 85% accuracy in detecting the need for pedestrian prioritization, 93% accuracy in determining the traffic light state for lane 1, and an average F1 score of 91.5%, reflecting a well-balanced trade-off between precision and recall. These results highlight the model’s ability to interpret and respond reliably to real-time input data.

Across 40 simulation scenarios covering a wide range of environmental and traffic conditions, the system achieved a contextual match rate of 87%, meaning that in most cases, the actions it selected were appropriate for the actual state of the intersection. The pedestrian light was activated 36 times in situations where prioritization was needed, and 91.67% of the scenarios were classified as “good”, indicating that the system consistently made safe and effective decisions.

These outcomes demonstrate the system’s ability to balance pedestrian safety with efficient traffic flow and underscore its potential for real-world implementation in smart transportation infrastructures.

# **9. References**

1. Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.<https://doi.org/10.1023/A:1010933404324>
2. Sidik, M. M. (2024, July 2). Various types of machine learning analysis. *Python in Plain English*.<https://python.plainenglish.io/various-types-of-machine-learning-analysis-a6e027a62db4>