Department of Computer Engineering

COMP413 - Internet of Things

Final Project Report

Instructor

Abdulkadir KÖSE

Project

Real-Time Adaptive Traffic Light Control Using Vehicle Detection

Group 6

Abdullah Azzam Amrulah

Enes Yaviç

Ertuğrul Sarıtekin

Mert Atalay Aktürk

1. Introduction

Urban traffic congestion has become a major challenge in modern cities due to increasing vehicle ownership and limited road infrastructure. Inefficient traffic signal control leads to longer waiting times, increased fuel consumption, and higher emissions. Traditional traffic light systems usually rely on fixed-time schedules, which cannot adapt to real-time traffic conditions and often result in poor performance during peak and off-peak hours.

Recent advances in computer vision and artificial intelligence have enabled new approaches for intelligent traffic management. By analyzing live traffic conditions, traffic signal behavior can be adjusted dynamically to improve traffic flow. In this project, an intelligent traffic monitoring system is developed to estimate traffic density at a road junction using live video streams and object detection techniques.

1.1 Problem Statement

Conventional traffic signal systems lack awareness of real-time traffic density and vehicle distribution across different lanes. As a result, some lanes remain congested while others are underutilized. Installing physical sensors such as loop detectors or deploying dedicated cameras can be costly and difficult to maintain. Therefore, there is a need for a flexible, software-based solution that can analyze traffic conditions using existing infrastructure and provide accurate traffic density information.

1.2 Objectives

The main objectives of this project are as follows:

- To analyze live traffic video streams from a real road junction
- To detect and count vehicles in different traffic lanes using computer vision
- To estimate traffic density in real time
- To provide traffic data through a software interface
- To design a system suitable for industrial and urban-scale applications

1.3 Project Scope and Contributions

This project focuses on a local traffic monitoring system that operates without cloud dependency. A live video feed from an existing traffic junction is processed locally, and vehicle counts are extracted using an object detection model. The system exposes traffic density data through a local server interface, which can be integrated with traffic control systems in the future.

The main contributions of this project include:

- A real-time traffic density estimation system using live video streams
- Lane-based vehicle counting using spatial analysis
- A modular software architecture suitable for further expansion
- An industrial-oriented design that prioritizes reliability and performance

2. System Model

This section describes the overall design and architecture of the proposed smart traffic monitoring system. The system is designed to analyze live traffic conditions at a road junction using computer vision techniques and to estimate traffic density in real time. The overall system architecture is shown in Figure 1.

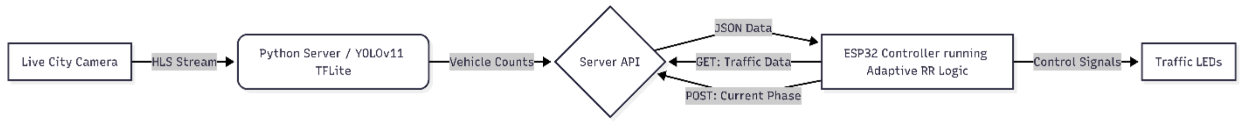


Figure 1. Overall architecture of the proposed traffic monitoring system.

2.1 Overall System Architecture

The proposed system consists of four main components:

1. Live traffic video source
2. Local processing unit
3. Traffic analysis module
4. Traffic data output interface

A live video stream from an existing traffic junction is used as the input source. This video stream is processed locally without relying on cloud services. The local processing unit runs traffic analysis software, which detects vehicles and estimates traffic density for each lane. The processed traffic data is then made available through a local server interface for further use.

This modular architecture allows the system to be easily extended or integrated with traffic control hardware in future developments.

2.2 Data Flow and Processing Pipeline

The system follows a sequential data processing pipeline, as illustrated in Figure 1.

First, a live video stream is obtained from a public traffic monitoring source. The video stream is continuously read frame by frame. Each frame is passed to the traffic analysis module, where vehicle detection is performed using a deep learning–based object detection model.

After vehicle detection, the system calculates the center point of each detected vehicle and assigns it to a specific traffic lane based on predefined spatial boundaries. The number of vehicles in each lane is then counted to estimate traffic density.

Finally, the lane-based vehicle counts are stored in a shared data structure and made available through a local application programming interface. This allows other system components, such as traffic light controllers or monitoring tools, to access the traffic data in real time.

2.3 Communication and Deployment Design

The proposed system is designed to operate locally and does not depend on cloud-based communication. All video processing and traffic analysis tasks are performed on a local processing unit. A lightweight local server is used to expose traffic data through a simple interface.

Low-power wide-area network technologies such as LoRa were not used in this project. The system is designed with an industrial-oriented perspective, where continuous power availability and high data throughput are assumed. Since video processing requires high bandwidth and low latency, local wired or wireless network communication is more suitable than low-power communication technologies. This design choice improves system reliability and ensures real-time performance for traffic monitoring applications.

2.4 Design Constraints and Assumptions

Several design assumptions were made during the development of the system:

- A stable power supply is available at the deployment location
- A live traffic video feed is accessible

- Local computing resources are sufficient for real-time video processing
- The system operates in a controlled urban environment

These assumptions reflect realistic conditions for industrial and smart city traffic monitoring systems. While the system is currently implemented as a monitoring solution, it can be extended to directly control traffic signals in future work.

3. Hardware Design

This section describes the hardware components used in the proposed traffic monitoring and control system. The hardware design focuses on simplicity, reliability, and suitability for industrial and smart city environments.

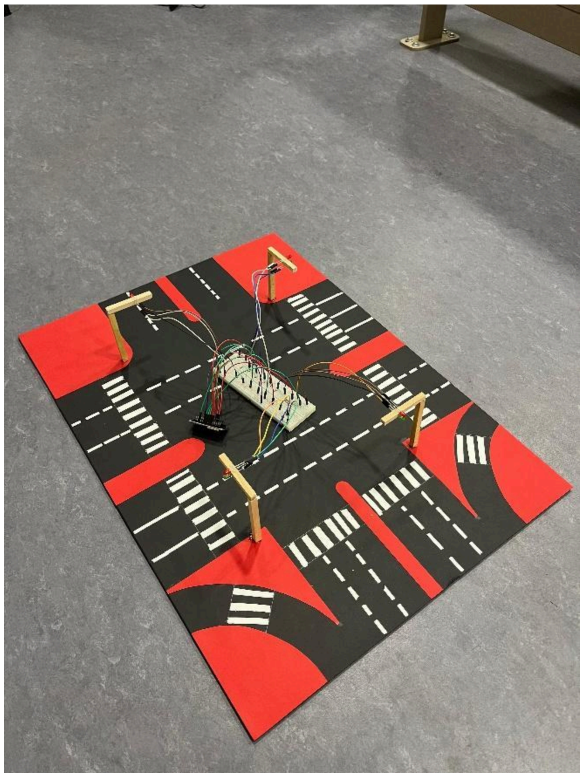


Figure 2. Hardware components used in the proposed system.

3.1 Hardware Components Overview

The main hardware component used in this project is the ESP32 microcontroller. The ESP32 is responsible for receiving traffic density data and controlling traffic light indicators. Due to its built-in wireless capabilities, and sufficient processing power for control tasks, the ESP32 was suitable for this project.

Traffic light indicators are represented using LEDs corresponding to red and green signals. These LEDs simulate real-world traffic lights and allow the system to demonstrate adaptive traffic signal behavior based on traffic density information.

A local processing unit (such as the laptop we used) is used to handle video processing and traffic analysis tasks. This separation ensures that computationally intensive operations, such as video decoding and object detection, do not overload the embedded hardware.

Full list of hardware components:

- **x1** ESP32 Dev Kit V1: *Main microcontroller for logic and Wi-Fi connectivity.*
- **x4** LEDs (Red): *Traffic light indicators for Stop.*
- **x4** LEDs (Green): *Traffic light indicators for Go.*
- **x8** Resistors (220Ω): *Current limiting for LEDs.*
- **x1** Breadboard & Wires: *For circuit assembly.*
- **x1** Host Computer: *Laptop, PC, or Raspberry Pi 4 to run the Python CV server.*

Cost of equipment:

- LEDs: 1.5 TL
- Resistor: 2 TL
- Breadboard: 45 TL
- Wires: 35 TL

Total: 83.5 TL

3.2 Deployment Environment

The system is designed for deployment in an urban traffic environment where continuous power supply and stable local networking are available. The live traffic video feed is accessed through https://tv.kayseri.bel.tr, Osman Kavuncu Boulevard camera to be exact, eliminating the need for additional camera installations.

The ESP32 is assumed to be deployed near the traffic control hardware, where it can receive traffic data through local communication and control the traffic light signals accordingly. This deployment model aligns with industrial traffic management systems, where centralized processing units support distributed control devices.

3.3 Hardware Design Justification

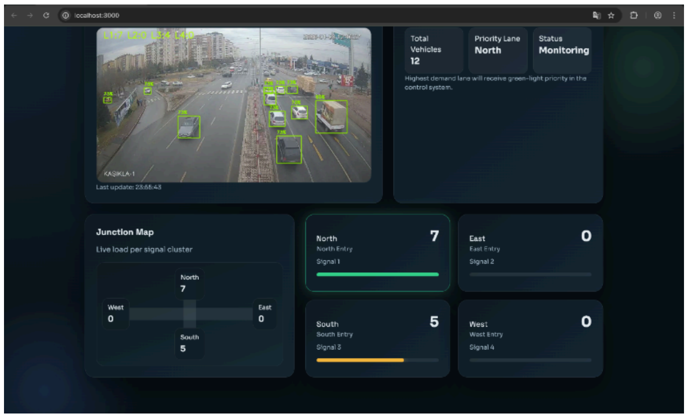
The choice of separating traffic analysis and control hardware was made to improve system performance and reliability. Video-based traffic analysis requires significant computational resources, which are not suitable for direct execution on low-power microcontrollers.

Low-power communication technologies such as LoRa were intentionally not used in this project. The system is designed with an industrial-oriented perspective, where real-time performance, higher data throughput, and low latency are prioritized over ultra-low power consumption. As a result, local wired or wireless communication methods are more appropriate for this application.

This hardware design allows the system to be scaled effectively and supports future integration with real traffic control infrastructure.

4. Software Design

This section describes the software architecture, data processing workflow, and traffic analysis logic used in the proposed system. The software is designed to operate locally, process live traffic video streams in real time, and provide traffic density information through a lightweight interface.



Highest demand lane will receive green-light priority in the control system.

Figure 3. Vehicle detection

4.1 Software Architecture

The software architecture follows a modular design, where each component is responsible for a specific task. The main software components include:

- Video stream acquisition module
- Traffic analysis and vehicle detection module
- Lane-based traffic density estimation module
- Local server and data interface module

The software architecture is lightweight and modular, allowing the inference server and trained model to be deployed on embedded platforms such as a Raspberry Pi for edge-based operation.

The simplified software architecture is shown in Figure 4. This modular approach improves maintainability and allows individual components to be updated or replaced without affecting the entire system.

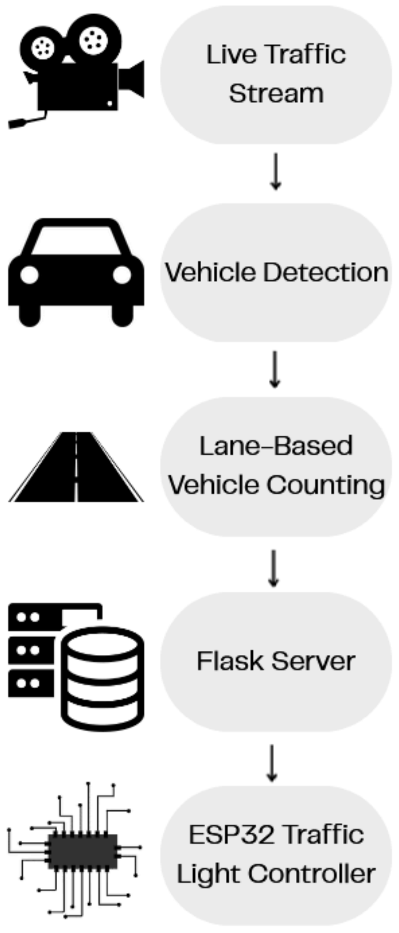


Figure 4. Software Architecture Diagram (simplified)

4.2 Video Stream Acquisition

The system uses a live traffic video stream obtained from https://tv.kayseri.bel.tr/osman-kavuncu-bulvari as mentioned before. The video stream is accessed programmatically and decoded frame by frame using a computer vision library.

Since the video stream uses temporary access tokens, the system includes a mechanism to automatically refresh the stream URL when the connection expires. This ensures continuous operation without manual intervention. Also, each video frame is forwarded to the traffic analysis module for further processing.

4.3 Vehicle Detection and Traffic Analysis

Vehicle detection is performed using a deep learning–based object detection model. The model processes each video frame and identifies vehicles present in the scene.

For every detected vehicle, the bounding box coordinates are extracted, and the center point of the bounding box is calculated. This information is used to determine the vehicle’s position within the traffic junction.

Figure 3 shows an example of vehicle detection results, where detected vehicles are highlighted and labeled in the video frame.

4.4 Lane-Based Traffic Density Estimation

To estimate traffic density, the video frame is divided into four logical regions representing traffic lanes. The lane boundaries are defined using fixed horizontal and vertical reference lines.

Each detected vehicle is assigned to a lane based on the position of its center point. The total number of vehicles in each lane is counted in real time. These counts represent the traffic density values used by the system.

This approach allows the system to adapt to different traffic distributions and provides accurate lane-based traffic information without requiring physical sensors.

4.5 Adaptive Traffic Light Control Logic

The system includes an adaptive traffic light control mechanism that dynamically adjusts signal durations based on real-time traffic density. Vehicle counts obtained from the video-based detection module are used as inputs to control logic.

For each lane, the detected vehicle count is translated into a corresponding green light duration using a rule-based algorithm. Lanes with higher traffic density are assigned longer green phases, while lanes with lower traffic density receive shorter durations. This approach allows the system to respond immediately to changing traffic conditions instead of relying on fixed-time schedules.

The traffic control logic is implemented on the embedded controller, enabling real-time decision-making without dependence on external servers. This design demonstrates how computer vision-based traffic monitoring can be directly integrated with traffic signal control systems.

4.6 Local Server and Data Interface

A lightweight local server is implemented to expose traffic density data to other system components. The server provides a simple interface that returns the current vehicle count for each lane.

This design allows the traffic data to be easily accessed by traffic light controllers, monitoring applications, or future system extensions. Since the server operates locally, the system does not rely on cloud connectivity, improving reliability and reducing latency.

4.7 TinyML and Model Selection

In this project, a YOLOv11-based object detection model was used for vehicle detection. YOLOv11 was selected because it provides strong real-time object detection performance and already includes vehicle-related classes, reducing the amount of training required.

A custom dataset was created specifically for the selected traffic intersection. A Python script was developed to automatically capture screenshots from the live traffic camera at different times of the day, resulting in approximately 300 images. These images were manually annotated using the Roboflow platform, where vehicles were labeled to generate the final training dataset.

The YOLOv11 model was trained using this labeled dataset to improve detection accuracy for the specific camera angle and traffic conditions. After training, the resulting model was converted into TensorFlow Lite format. This conversion enables lightweight and efficient inference suitable for edge-oriented systems.

Although training was performed using standard deep learning tools, the final deployment follows TinyML principles by using a compact TensorFlow Lite model optimized for efficient execution. The inference pipeline is designed to run locally without cloud dependency, supporting real-time processing.

The trained model and Python-based inference server are designed to be deployable on embedded platforms such as a Raspberry Pi, enabling edge-based traffic analysis in practical smart city deployments.

4.8 Dataset Preparation

A custom dataset was created to train the vehicle detection model for the selected traffic intersection. A Python script was developed to capture screenshots from the live traffic camera feed at different times of the day. This process resulted in approximately 300 images representing various traffic conditions.

The collected images were annotated using the Roboflow platform, where vehicles were manually labeled. The labeled dataset was then used to train the YOLOv11 object detection model, allowing the model to adapt to the specific camera angle and road layout.

5. Results and Discussion

This section presents the results obtained from the proposed traffic monitoring system and discusses system behavior, observed outcomes, and design challenges. The system was evaluated using a live traffic video feed from a real urban junction.

5.1 Experimental Setup

The system was tested using a live traffic video stream obtained from an existing city traffic monitoring source. The video stream was processed locally on a personal computer running the traffic analysis software.

The system continuously analyzed incoming video frames and detected vehicles in real time. Traffic density was estimated by counting the number of detected vehicles in each predefined lane region. The resulting traffic data was accessed through a local server interface and used to simulate adaptive traffic signal behavior.

5.2 Observed Traffic Patterns

During testing, the system successfully detected vehicles under different traffic conditions, including low-density and high-density traffic scenarios. The number of vehicles varied significantly across lanes, demonstrating the limitations of fixed-time traffic signal systems.

Figure 3 shows an example of vehicle detection and lane-based traffic analysis. Detected vehicles are highlighted, and the estimated vehicle count for each lane is displayed on the video frame.

The results indicate that traffic density changes dynamically over time and differs across lanes, highlighting the importance of adaptive traffic control strategies.

5.3 System Performance

The system demonstrated stable real-time performance during continuous operation. Vehicle detection and lane assignment were performed without noticeable delay, allowing traffic density information to be updated in real time.

Using a lightweight TensorFlow Lite model enables efficient inference while maintaining acceptable detection accuracy. The modular software architecture ensured that video processing, traffic analysis, and data serving operated smoothly without interfering with each other.

5.4 Challenges and Design Decisions

Several challenges were encountered during system development. One major challenge was handling temporary video stream interruptions caused by expiring access tokens. This issue was addressed by implementing an automatic stream refresh mechanism.

Another challenge involved balancing detection accuracy and processing efficiency. Using a lightweight model helped maintain real-time performance while reducing computational load.

Low-power communication technologies such as LoRa were not used in this project. Since the system targets industrial-oriented traffic monitoring applications with continuous power availability and high data throughput requirements, local processing and communication were preferred. This design decision improved system responsiveness and reliability.

5.5 Discussion

The results demonstrate that video-based traffic analysis is an effective approach for estimating real-time traffic density. By using existing traffic camera infrastructure and local processing, the system avoids the need for additional hardware sensors.

While the current implementation focuses on traffic monitoring, the system can be extended to directly control traffic signals. The results obtained show strong potential for integration into smart city traffic management systems.

6. Limitations and Future Work

Although the proposed system successfully demonstrates real-time traffic density estimation using video analysis, some limitations should be acknowledged.

One limitation is the dependency on video quality. Poor lighting conditions, heavy rain, fog, or low-resolution streams may reduce vehicle detection accuracy. Occlusion between vehicles, especially during high traffic congestion, can also affect counting performance.

Another limitation is that the system currently relies on predefined lane boundaries. Changes in camera angle or junction geometry would require manual adjustment of lane divider parameters. Additionally, the system does not distinguish between different vehicle types such as cars, buses, or trucks.

The system was designed for local and industrial-style deployment; therefore, low-power communication technologies such as LoRa were not considered. This decision aligns with the project’s focus on continuous operation, high data throughput, and real-time processing rather than long-range, low-bandwidth communication.

6.1. Future Work

Several improvements can be considered for future development. Adaptive lane segmentation techniques could be introduced to automatically adjust lane boundaries based on camera position. Vehicle classification could also be added to provide more detailed traffic analysis.

Deploying the system on edge devices with hardware acceleration could improve efficiency and make the solution more scalable for real-world smart city applications.

7. Conclusion

In this project, a video-based smart traffic monitoring system was designed and implemented to estimate real-time traffic density at a road junction. The system processes a live traffic video feed, detects vehicles, and computes lane-based traffic density using local processing.

The proposed approach demonstrates that effective traffic analysis can be achieved through live feed. By leveraging existing traffic camera infrastructure and efficient machine learning models, the system provides a practical and scalable solution for intelligent traffic management.

The results show that adaptive traffic systems based on real-time data can outperform traditional fixed-time signal systems. Overall, the project successfully meets its objectives and highlights the potential of intelligent video-based solutions for smart city transportation systems.

REFERENCES

1. Ultralytics, *YOLO Object Detection Documentation*
   https://docs.ultralytics.com

2. TensorFlow, *TensorFlow Lite Documentation*
   https://www.tensorflow.org/lite

3. OpenCV, *Open-Source Computer Vision Library*
   https://opencv.org

4. Flask, *Flask Web Framework Documentation*
   https://flask.palletsprojects.com

5. Playwright, *Playwright Automation Framework*
   https://playwright.dev

6. Arduino, *ESP32 Development Documentation*
   https://docs.espressif.com

7. World Commission on Environment and Development, *Sustainable Development Goals*
   https://sdgs.un.org/goals

8. Kayseri Municipality, *Live Traffic Camera Feed*
   https://tv.kayseri.bel.tr

9. Roboflow, *Image Annotation Platform*
   https://roboflow.com/

Source code & video demo available here:

https://github.com/abdulazzam204/comp413-g6-smart-traffic-light