A Project on

Web Interface Based Automation of Toll Collection System Using Computer Vision

Submitted by

Name	Student ID
Imran Khan Prince	1943557025
Md Imdadul Haque Milon	1943557012
Md Kabirul Islam	1943557030

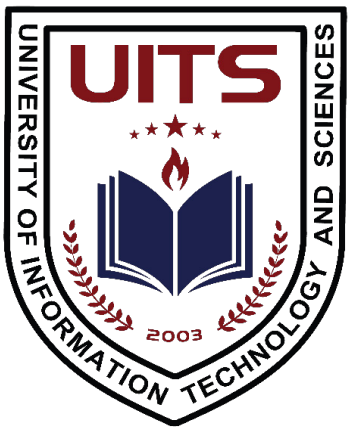
Submitted to

Mr. Md. Qamarul Hasan

Lecturer

Department of Electrical and electronic Engineering (EEE)

University of Information Technology and Sciences (**UITS**)



Department of Electrical and Electronic Engineering (**EEE**)

July 19, 2023

UITS

Future will be better than thy past

University of Information Technology & Sciences

An initiative of PHP Family

Web Interface Based Automation of Toll Collection System Using Computer Vision

This Project is presented in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering (B.Sc. in EEE) Spring-2023

University of Information Technology and Sciences (**UITS**)

WEB INTERFACE BASED AUTOMATION OF TOLL COLLECTION SYSTEM USING COMPUTER VISION

Submitted To

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING (EEE)

UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES (UITS)

In partial fulfillment of the requirements for the award of the degree

Of

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING (EEE)

Submitted By:

Name	Student ID
Imran Khan Prince	1943557025
Md Imdadul Haque Milon	1943557012
Md Kabirul Islam	1943557030

Under the Supervision of

Mr. Md. Qamarul Hasan
Lecturer
Department of Electrical and electronic Engineering (EEE)
University of Information Technology and Sciences (UITS)

Dhaka, Bangladesh.

July 19, 2023

Student's Declaration:

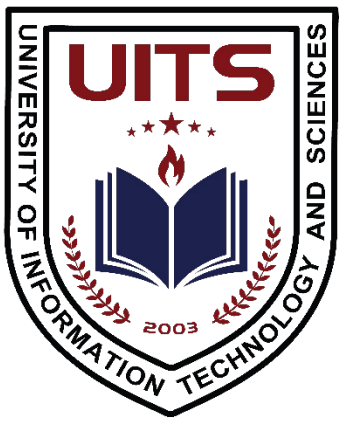
We, the undersigned, declare that this "**Web Interface Based Automation of Toll Collection System Using Computer Vision**" project report is the result of our original work as a group. The data and information presented in this report have been obtained from reliable sources, and any opinions or findings expressed herein are those of our group.

We confirm that we have appropriately acknowledged all the sources of information and materials used in this report and that all the citations and references are correctly included and appropriately formatted. Furthermore, we affirm that this report has not been submitted previously in part or in full to any other educational institution or for any other purpose. This report is only intended for the academic evaluation of our final-year project.

We understand that any act of plagiarism, misrepresentation, or dishonesty will result in severe academic and/or disciplinary actions.

Signed,

Student's Signature	Student's Signature	Student's Signature
Imran Khan Prince	Md. Imdadul Haque Milon	Md. Kabirul Islam
ID:1943557025	**ID:**1943557012	**ID:**1943557030



University of Information Technology and Sciences (UITS)

Certification

This is to certify that I have supervised the preparation and submission of the project report entitled " **Web Interface Based Automation of Toll Collection System Using Computer Vision** " by **Imran Khan Prince**, **Md. Imdadul Haque Milon**, and **Md. Kabirul Islam** in partial fulfillment of the requirements for the project.

I hereby acknowledge that I have reviewed the content and quality of the project report, and I believe it is a comprehensive and suitable representation of the work carried out during the project.

I further certify that to the best of my knowledge; the project report is the original work of the aforementioned group members and that it has not been submitted elsewhere for any other academic or non-academic purpose.

Approved By:

_____

Supervisor's Signature

Mr. Md. Qamarul Hasan

Acknowledgments

First and foremost, we would like to thank our project supervisor, **Mr. Md. Qamarul Hasan** Sir, for his guidance, encouragement, and support throughout the project. His expertise and knowledge in the field of Image Processing have been invaluable in the successful completion of this project. He has been a constant source of inspiration to us. We consider ourselves extremely fortunate for having the opportunity to learn and work under his supervision over the entire period.

We are very much thankful to all of our faculty members of **EEE Department of UITS**. They helped us a lot through our project. Without their help and support, we could not complete our work. We also thank our department for providing us with a great opportunity and platform to make our effort a success.

We would like to express our appreciation to our families and friends who have provided us with moral support and encouragement throughout the project.

Finally, we would like to thank all the participants of the project who have dedicated their time and effort to make this project a success.

We acknowledge that without the support and guidance of these individuals, this project would not have been possible.

Thanks all.

.

ABSTRACT

This project addresses traffic congestion challenges in Bangladesh's mega projects by introducing a novel toll collection solution based on computer vision technology. The system comprises three modules: the Camera and Detection Module, using Python and OpenCV to capture video footage and extract license plate information with Optical Character Recognition (OCR); the Web Server and Verification Module, powered by Python Flask, which verifies the license plate against a database and maintains comprehensive records; and the Hardware Module, integrating an Arduino Uno microcontroller to control toll booth gates based on verification status received from the Web Server. By automating toll collection, this system minimizes human inefficiencies, leading to improved traffic flow, reduced congestion, and enhanced transportation management. The project report provides an in-depth analysis of the system's design, methodology, and implementation, demonstrating its effectiveness in enhancing toll collection efficiency and revolutionizing traffic management in Bangladesh's mega projects. By leveraging computer vision and web-based technologies, this innovative approach holds great promise for creating smoother traffic, reducing travel times, and elevating the overall user experience in the region's transportation network.

Table of Contents

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Bangladesh, with its booming infrastructure development and numerous mega projects, faces the daunting challenge of managing traffic congestion at toll booths. These toll booths, often manned by human operators, contribute to significant delays, long queues, and frustrating traffic bottlenecks. The inefficiencies and errors associated with traditional toll collection systems demand innovative solutions to alleviate congestion and enhance overall transportation efficiency. Dhaka elevated expressway, Padma Bridge and Jatrabari Flyover [1] are just a few examples of the recent mega transportation projects. These projects have been constructed to save time. These projects are built with debt from banks and to repay that debt we have there is a toll collection booth located behind every one of those mega projects. Those toll collection booths are maintained by humans and due to human inefficiency, most of the time there is a long line of cars behind every toll booth [2].

The prevalence of traffic congestion at toll booths in Bangladesh underscores the need for a more streamlined and automated toll collection process. Existing manual systems are prone to human errors, including misidentification of license plates, slow transaction processing, and inadequate data management. These inefficiencies not only cause delays but also hinder accurate vehicle tracking, which is crucial for effective traffic management and monitoring. The government is addressing the problem and finding a solution for that [3].

To address these challenges, the project "Web Interface Based Automation of Toll Collection System Using Computer Vision" aims to revolutionize toll collection processes in Bangladesh's mega projects. By leveraging the power of computer vision technology and intelligent automation, the project seeks to replace manual toll collection with a more efficient, accurate, and reliable system.

The project's core objective is to automate the toll collection process, thereby minimizing human intervention and associated errors. The integration of computer vision techniques allows for automated license plate recognition, ensuring swift and accurate identification of vehicles passing through toll booths. The use of optical character recognition (OCR) enables efficient extraction of number plate information, facilitating seamless vehicle tracking and verification.

By implementing this automated toll collection system, the project endeavors to achieve several significant benefits. Firstly, it aims to enhance traffic flow by reducing congestion at toll booths, leading to reduced travel times and improved overall transportation efficiency. Secondly, the system's automated verification capabilities offer improved accuracy and security, eliminating the possibility of unauthorized access or fraudulent activities. Thirdly, the project seeks to enhance data management by maintaining comprehensive records of passing vehicles, enabling effective traffic analysis and monitoring.

The project's significance extends beyond immediate traffic management benefits. The successful implementation of an automated toll collection system has the potential to revolutionize transportation systems in Bangladesh. It sets a precedent for deploying advanced technologies to address complex traffic challenges, positioning the country at the forefront of intelligent transportation solutions. Moreover, the project's outcomes can inspire similar initiatives in other regions facing similar traffic congestion issues.

By leveraging computer vision technology and intelligent automation, the "Web Interface Based Automation of Toll Collection System Using Computer Vision" project aims to improve traffic flow, reduce congestion, and enhance transportation efficiency in Bangladesh's mega projects. Through innovative solutions, the project endeavors to eliminate the drawbacks of traditional toll collection systems and pave the way for a more streamlined and effective toll management infrastructure.

1.2 PROBLEM STATEMENT

In the context of Bangladesh's mega projects, the conventional toll collection systems in place at toll booths contribute significantly to traffic congestion, leading to delays, long queues, and inefficiencies in transportation. The human-operated toll collection process is prone to errors, resulting in increased wait times for commuters and hindrances to overall traffic flow. These challenges highlight the need for an innovative solution to streamline toll collection and alleviate the associated traffic congestion.

The existing toll collection systems suffer from several key problems. Firstly, the reliance on human operators introduces inefficiencies and errors during the manual identification and verification of vehicles passing through the toll booths. These errors include

misidentification of license plates, incorrect data entry, and delays in transaction processing. Secondly, the lack of an automated verification process leads to difficulties in distinguishing between registered and unauthorized vehicles, further exacerbating traffic congestion. Thirdly, the absence of a centralized database and real-time information sharing makes it challenging to track passing vehicles and analyze traffic patterns effectively.

The limitations of the current toll collection systems have a direct impact on transportation efficiency, economic productivity, and the overall quality of life for commuters. The congestion and delays experienced at toll booths not only result in increased travel times but also cause fuel wastage, environmental pollution, and unnecessary stress for road users. Furthermore, the lack of accurate and timely traffic data hampers effective traffic management and impedes efforts to optimize road infrastructure. Therefore, a pressing problem exists in the toll collection systems of Bangladesh's mega projects, requiring an innovative solution that can automate the collection process, improve verification accuracy, enhance data management, and alleviate traffic congestion. By addressing these challenges, a more efficient and streamlined toll collection system can be established, facilitating smoother traffic flow, reducing delays, and enhancing overall transportation efficiency in the country's mega projects.

The proposed project, "Web Interface Based Automation of Toll Collection System Using Computer Vision," aims to bridge this gap by implementing an automated toll collection system that utilizes computer vision technology for license plate recognition, web-based server architecture for verification, and hardware integration for seamless gate control. By mitigating the existing problems and enhancing the overall toll collection process, this project seeks to significantly reduce traffic congestion, optimize transportation efficiency, and improve the overall commuter experience in Bangladesh's mega projects.

1.3 OBJECTIVE

The objective of the project, "Web Interface Based Automation of Toll Collection System Using Computer Vision," is to develop an efficient and automated toll collection system that utilizes computer vision technology, web-based server architecture, and hardware integration to address the challenges associated with traffic congestion at toll booths in Bangladesh's mega projects. The project aims to achieve the following objectives:

- **Automate Toll Collection:** Develop a system that automates the process of toll collection by utilizing computer vision techniques to extract license plate information from captured video footage. Implement Optical Character Recognition (OCR) algorithms to accurately recognize and extract the number plate text.
- **Improve Verification Accuracy:** Establish a web-based server module that verifies the extracted license plate information against a database of registered vehicles. Develop an efficient data structure and algorithms to ensure accurate verification of passing vehicles, distinguishing between authorized and unauthorized vehicles.
- **Enhance Data Management:** Create a comprehensive database system that maintains records of registered vehicles, passing times, and dates. Implement efficient data storage and retrieval mechanisms to enable real-time information sharing and analysis of traffic patterns.
- **Mitigate Traffic Congestion:** Minimize traffic congestion at toll booths by automating the toll collection process, reducing human errors, and optimizing the flow of authorized vehicles. Improve overall traffic management efficiency and enhance the commuting experience for road users.
- **Ensure Seamless Gate Control:** Integrate hardware components, including an Arduino Uno microcontroller, with the system to enable seamless gate control based on the verification status received from the web server. Develop reliable communication protocols for efficient interaction between the software and hardware modules.
- **Evaluate System Performance:** Conduct thorough performance evaluations to assess the accuracy, speed, and reliability of the automated toll collection system. Measure the system's effectiveness in reducing traffic congestion, minimizing delays, and improving transportation efficiency.

By achieving these objectives, the project aims to revolutionize the toll collection process in Bangladesh's mega projects, mitigating traffic congestion, optimizing transportation efficiency, and enhancing the overall commuting experience for road users

.

Chapter 2

LITERATURE REVIEW

A toll collection system is a mechanism that collects fees from vehicles using a road, bridge, or tunnel. It is an essential component of intelligent transportation systems (ITS) that aim to improve traffic management, safety, and efficiency. However, traditional toll collection systems that rely on manual operations or physical tokens have several drawbacks, such as long waiting times, fuel wastage, high accident risks, and environmental pollution. Therefore, there is a need for developing automated and smart toll collection systems that can reduce these problems and enhance the user experience.

One of the promising technologies that can enable automated and smart toll collection systems is computer vision. Computer vision is the field of study that deals with the acquisition, processing, analysis, and understanding of visual information from images or videos. Computer vision can be used to perform various tasks related to toll collection, such as vehicle detection, classification, identification, tracking, and verification. By using computer vision techniques, toll collection systems can achieve higher accuracy, speed, security, and convenience.

In this literature review, we will survey some of the existing works on computer vision-based toll collection systems. We will focus on the following aspects: **(1) the camera and detection module**, **(2) the web server and verification module**, and **(3) hardware module**. We will also compare and contrast the different approaches, methods, and challenges in each aspect. Finally, we will summarize the main findings and identify the research gaps and future directions.

2.1 CAMERA AND DETECTION MODULE

The camera and detection module is responsible for capturing the images or videos of the vehicles passing through the toll plaza and extracting the relevant information from them. The main tasks involved in this module are: (1) camera selection and placement, (2) image preprocessing and enhancement, (3) region of interest (ROI) extraction, (4) optical character recognition (OCR), and (5) vehicle re-identification.

2.1.1 Camera Selection and Placement

The choice of camera type and location is crucial for obtaining high-quality images or videos of the vehicles. The camera should have sufficient resolution, frame rate, field of view, and sensitivity to capture clear and stable images or videos in various lighting and

weather conditions. The camera should also be placed at an optimal angle and distance from the vehicles to avoid occlusion, distortion, or reflection.

Several studies have proposed different types of cameras for toll collection systems. For example, [4] used day/night infrared (IR) cameras to capture the images of the vehicle and its number plate in both day and night scenarios. [5] used a high-definition (HD) camera mounted on a gantry to capture the front view of the vehicle. [6] used a web camera connected to a Raspberry Pi board to capture the rear view of the vehicle.

2.1.2 Image Preprocessing and Enhancement

The images or videos captured by the camera may suffer from various degradations due to noise, blur, illumination variation, shadow, or weather effects. Therefore, image preprocessing and enhancement techniques are applied to improve the quality and visibility of the images or videos before further processing. Some of the common techniques include: (1) grayscale conversion, (2) histogram equalization or normalization, (3) noise reduction or filtering, (4) edge detection or sharpening, and (5) contrast or brightness adjustment. Several studies have applied different image preprocessing and enhancement techniques for toll collection systems. For example, [4] applied grayscale conversion, histogram equalization, and median filtering to enhance the contrast and reduce the noise of the IR images. [5] applied grayscale conversion, histogram normalization, and Gaussian filtering to improve the brightness and smoothness of the HD images. [6] applied grayscale conversion, histogram equalization, and Canny edge detection to highlight the edges and contours of the web camera images.

2.1.3 Region of Interest Extraction

The region of interest (ROI) extraction is the process of locating and cropping the specific part of the image or video that contains useful information for toll collection. The most important ROI for toll collection is the license plate or number plate of the vehicle. The license plate contains alphanumeric characters that uniquely identify the vehicle owner and registration details. The license plate can be used to verify the vehicle identity, check its validity, and charge its fee accordingly. Several studies have proposed different methods for ROI extraction for toll collection systems. For example, [1] used a cascade classifier based on Haar-like features to detect the license plate region from the IR images. [5] used a convolutional neural network (CNN) based on YOLOv3 architecture to detect the license

plate region from the HD images. [6] used a contour analysis based on Hough transform to detect the license plate region from the web camera images.

2.1.4 Optical Character Recognition

Optical character recognition (OCR) is the process of converting the image of text into machine-readable text. OCR is applied to the license plate image to extract the alphanumeric characters that represent the vehicle number. OCR can be performed using various techniques, such as template matching, feature extraction, segmentation, classification, or deep learning. Several studies have applied different OCR techniques for toll collection systems. For example, [4] used a template matching based on correlation coefficient to recognize the license plate characters from the IR images. [5] used a feature extraction based on histogram of oriented gradients (HOG) and a classification based on support vector machine (SVM) to recognize the license plate characters from the HD images. [6] used a segmentation based on connected components and a classification based on K-nearest neighbors (KNN) to recognize the license plate characters from the web camera images.

2.1.5 Vehicle Re-identification

Vehicle re-identification is the process of matching the vehicle image with a database of previously seen vehicles. Vehicle re-identification can be used to enhance the accuracy and security of toll collection systems by verifying the vehicle identity, detecting the vehicle duplication or fraud, and tracking the vehicle movement. Vehicle re-identification can be performed using various attributes, such as color, model, type, or logo of the vehicle. Several studies have proposed different methods for vehicle re-identification for toll collection systems. For example, [4] used a Siamese model based on CNNs to compare the vehicle attributes from the IR images with a database of real-world vehicle images from VeRi776 dataset. [5] used a feature extraction based on HOG and a classification based on SVM to classify the vehicle type from the HD images into four categories: car, bus, truck, or motorcycle. [6] used a template matching based on normalized cross-correlation to match the vehicle logo from the web camera images with a database of predefined logos.

2.2 WEB SERVER AND VERIFICATION MODULE

The web server and verification module is responsible for receiving and processing the information from the camera and detection module and providing the feedback and service to the user. The main tasks involved in this module are: (1) web server design and implementation, (2) data structure and database management, (3) verification algorithm and logic, and (4) user interface and communication.

2.2.1 Web Server Design and Implementation

The web server is the software component that handles the requests and responses between the camera and detection module and the user. The web server should have sufficient capacity, speed, security, and reliability to handle multiple concurrent requests and responses in real time. The web server should also support various protocols, formats, and methods for data transmission and communication. Several studies have proposed different web servers for toll collection systems. For example, [4] used a Python Flask server to create a web server and hosted it on localhost. The Flask server is a lightweight and flexible framework that allows easy development and deployment of web applications using Python. [5] used a Java Servlet server to create a web server and hosted it on Amazon Web Services (AWS). The Servlet server is a robust and scalable framework that allows dynamic generation and processing of web content using Java. [6] used a Node.js server to create a web server and hosted it on Heroku. The Node.js server is a fast and efficient framework that allows event-driven and non-blocking input/output operations using JavaScript.

2.2.2 Data Structure and Database Management

The data structure and database management is the process of organizing, storing, retrieving, and updating the data related to toll collection. The data may include the vehicle information, such as number plate, type, color, model, logo, owner name, registration date, validity status, fee amount, etc. The data may also include the transaction information, such as passing time, passing date, verification status, payment method, payment confirmation, etc. The data structure and database management should ensure the consistency, integrity, security, and availability of the data. Several studies have proposed different data structures and databases for toll collection systems. For example, [4] used two lists as data structures: one for car database and another for passed car database. The car database stored all the registered cars with their number plates as keys and their owners' names as values. The passed car database stored the passed cars with their number plates as keys and their passing times and dates as values. [5] used two tables as data structures: one for vehicle information

table and another for transaction information table. The vehicle information table stored the vehicle details with their number plates as primary keys and their types, colors, models, logos, owners' names, registration dates, validity statuses, and fee amounts as attributes. The transaction information table stored the transaction details with their transaction IDs as primary keys and their number plates as foreign keys referencing the vehicle information table. The transaction information table also stored their passing times, passing dates, verification statuses, payment methods, and payment confirmations as attributes. [6] used MongoDB as a database management system. MongoDB is a cross-platform and document-oriented database that allows flexible and dynamic schema design using JSON-like documents. MongoDB stored the vehicle and transaction information in a single collection with their number plates as unique identifiers and their other details as fields.

2.2.3 Verification Algorithm and Logic

The verification algorithm and logic is the process of checking and validating the vehicle information and transaction information for toll collection. The verification algorithm and logic should ensure the correctness, completeness, and timeliness of the verification process. The verification algorithm and logic should also handle various scenarios, such as valid or invalid vehicles, registered or unregistered vehicles, paid or unpaid vehicles, etc.

Several studies have proposed different verification algorithms and logics for toll collection systems. For example,

[4] used a simple verification algorithm that received a GET request from the camera and detection module with the license plate text as a parameter. The algorithm then searched the license plate text in the car database list and returned a verification status as "verified" or "not verified" to the camera and detection module. The algorithm also appended the license plate text and the current time and date to the passed car database list.

[5] used a complex verification algorithm that received a POST request from the camera and detection module with the license plate image as a parameter. The algorithm then performed OCR on the license plate image to extract the license plate text. The algorithm then searched the license plate text in the vehicle information table and retrieved the vehicle details. The algorithm then checked the validity status and fee amount of the vehicle and returned a verification status as "valid" or "invalid" to the camera and detection module. The algorithm also inserted a new record to the transaction information table with the

transaction ID, number plate, passing time, passing date, verification status, payment method, and payment confirmation.

[6] used a hybrid verification algorithm that received a GET request from the camera and detection module with the license plate text and vehicle logo as parameters. The algorithm then searched the license plate text in the MongoDB collection and retrieved the vehicle details. The algorithm then compared the vehicle logo with the predefined logos in the database and returned a verification status as "matched" or "unmatched" to the camera and detection module. The algorithm also updated the existing record in the MongoDB collection with the passing time, passing date, verification status, payment method, and payment confirmation.

2.2.4 User Interface and Communication

The user interface and communication is the process of providing the feedback and service to the user based on the verification status and transaction information. The user interface should be clear, intuitive, informative, and user-friendly. The communication should be fast, secure, and reliable.

Several studies have proposed different user interfaces and communication methods for toll collection systems. For example, [4] used two endpoints for web server communication: one for "/api/verify" and another for "/". In the verify endpoint, the web server received the GET request from the camera and detection module and returned the verification status as a JSON response. In the "/" endpoint, the web server served a list of passed cars as a HTML page. [5] used a graphical user interface (GUI) for user feedback and service. The GUI displayed the vehicle image, the license plate image, the vehicle details, the transaction details, and the verification status on a monitor screen. The GUI also provided a payment option for the user to choose between cash or card payment. The GUI communicated with the web server using HTTP protocol. [6] used a mobile application for user feedback and service. The mobile application allowed the user to view the toll historical data, get toll news feeds, and make toll payments using QR code scanning. The mobile application communicated with the web server using WebSocket protocol.

2.3 Hardware Module

The hardware module is responsible for controlling the physical devices that are involved in toll collection, such as gate, sensor, motor, LED, buzzer, etc. The hardware module should be compatible, responsive, and robust.

Several studies have proposed different hardware modules for toll collection systems. For example, [4] used an Arduino Uno board with other motors and sensors as the hardware module. The Arduino Uno board received a serial communication from the camera and detection module with the verification status as a text message. Based on the verification status, the Arduino Uno board sent a signal to open or close the gate using a servo motor. [5] used a Raspberry Pi board with other devices as the hardware module. The Raspberry Pi board received a wireless communication from the web server with the verification status as a JSON message. Based on the verification status, the Raspberry Pi board controlled the gate using a relay module, the LED using a GPIO pin, and the buzzer using a PWM pin. [6] used a NodeMCU board with other components as the hardware module. The NodeMCU board received a MQTT communication from the web server with the verification status as a topic message. Based on the verification status, the NodeMCU board controlled the gate using a stepper motor, the LED using a digital pin, and the buzzer using an analog pin.

2.4 SUMMARY AND FUTURE DIRECTIONS

In this literature review, we have surveyed some of the existing works on computer vision based toll collection systems. We have focused on the following aspects: (1) camera and detection module, (2) web server and verification module, and (3) hardware module. We have compared and contrasted the different approaches, methods, and challenges in each aspect.

From the literature review, we can observe that computer-vision based toll collection systems have several advantages over traditional toll collection systems, such as higher accuracy, speed, security, and convenience. However, there are also some limitations and challenges that need to be addressed, such as:

The camera and detection module may face difficulties in capturing and processing the images or videos of the vehicles in complex scenarios, such as occlusion, distortion, reflection, illumination variation, shadow, weather effects, etc. - The web server and verification module may face difficulties in handling and verifying the vehicle information

and transaction information in real time, especially when there are multiple concurrent requests and responses.

The hardware module may face difficulties in controlling and coordinating the physical devices that are involved in toll collection, such as gate, sensor, motor, LED, buzzer, etc. Therefore, some of the possible future directions for computer vision-based toll collection systems are:

Developing more robust and adaptive camera and detection methods that can cope with various challenges in image or video acquisition and processing. - Developing more efficient and secure web server and verification methods that can handle large-scale data transmission and communication. Developing more compatible and responsive hardware methods that can integrate various devices and technologies for toll collection.

Chapter 3

METHODOLOGY

3.1 Brief Review of Used Technologies

3.1.1 Software:

Python

Figure 1: Python logo

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991 [7]. Python is widely used in various domains, including web development, data analysis, artificial intelligence, scientific computing, and automation. It has become one of the most popular programming languages due to its ease of use and versatility. Python Language has been used in this project for all of the modules except the hardware. The Key Features of Python are,

Readability: Python emphasizes code readability, making it easy for both beginners and experienced developers to write and understand code. Its syntax uses indentation and whitespace to define code blocks, making it visually intuitive.

Versatility: Python can be used for a wide range of applications, from writing simple scripts to building complex web applications and machine learning models. It offers extensive libraries and frameworks that cater to diverse programming needs.

Interpreted Language: Python is an interpreted language, meaning that the code is executed line by line by the Python interpreter. This allows for rapid development and debugging without the need for compilation.

Object-Oriented: Python supports object-oriented programming, enabling developers to create and use classes and objects, making code modular and reusable.

Rich Standard Library: Python comes with a rich standard library that provides a wide range of built-in modules and functions for performing various tasks, such as file handling, networking, and data manipulation.

Community Support: Python has a large and active community of developers who contribute to its continuous improvement. This community-driven aspect ensures a wealth of resources, tutorials, and documentation available for learners and developers.

Python's simplicity, readability, and extensive standard library have made it a go-to choice for developers across various industries. In our project, Python serves as the core programming language for implementing the different modules, including the camera and detection module, and web server module. Python's ability to handle complex tasks efficiently and its ease of integration with other technologies, it has enabled us to build a comprehensive and functional toll-collection system. Additionally, Python's vast ecosystem, including libraries like OpenCV, Flask, and PySerial, has allowed us to take advantage of existing tools and accelerate the development process. Overall, Python's combination of readability, versatility, and community support has been instrumental in the success of our project, providing a solid foundation for creating a reliable and efficient toll collection system with a web-based interface and computer vision capabilities.

OpenCV-python

Figure 2: Python OpenCV logo

OpenCV, short for "Open-Source Computer Vision Library," is a powerful and popular library used for computer vision and image processing tasks in Python [8]. It was originally developed by Intel and is now maintained by a large community of developers. OpenCV is designed to provide easy-to-use tools and functions for working with images and videos, making it an excellent choice for beginners and experts alike. This library was used for our project in the identification module. Key Features and capabilities of the library are,

Image and Video Processing: OpenCV allows you to read, write, and manipulate images and videos with ease. You can load images from files, capture video streams from cameras, and apply various operations like resizing cropping, and color transformations.

Computer Vision Algorithms: OpenCV offers a wide range of computer vision algorithms, such as object detection, face recognition, image segmentation, and feature matching. These algorithms help you analyze and understand the content of images.

Image Filtering and Enhancements: OpenCV provides tools for filtering and enhancing images. You can apply filters like blurring and sharpening, adjust brightness and contrast, and perform histogram equalization to improve the quality of images.

Feature Detection and Description: OpenCV enables you to detect and describe features in images, making it useful for tasks like image registration, object tracking, and augmented reality applications.

Machine Learning Integration: OpenCV seamlessly integrates with machine learning libraries like scikit-learn and TensorFlow, allowing you to combine computer vision tasks with advanced machine learning techniques.

OpenCV is a fundamental and widely used library in our project. It plays a crucial role in the camera and detection module, where it is used to capture video from the camera source, detect license plate regions using the Haar cascade classifier, and perform various image processing operations. By leveraging OpenCV's extensive functionalities, we have been able to build an efficient and accurate system for detecting and extracting license plate information from video streams. Its ease of use, cross-platform compatibility, and community support have made it an invaluable tool for our computer vision tasks. Overall, OpenCV empowers our project with powerful image and video processing capabilities, making it possible to implement a reliable and efficient toll collection system with a web-based interface and computer vision features.

Cascade Classifier

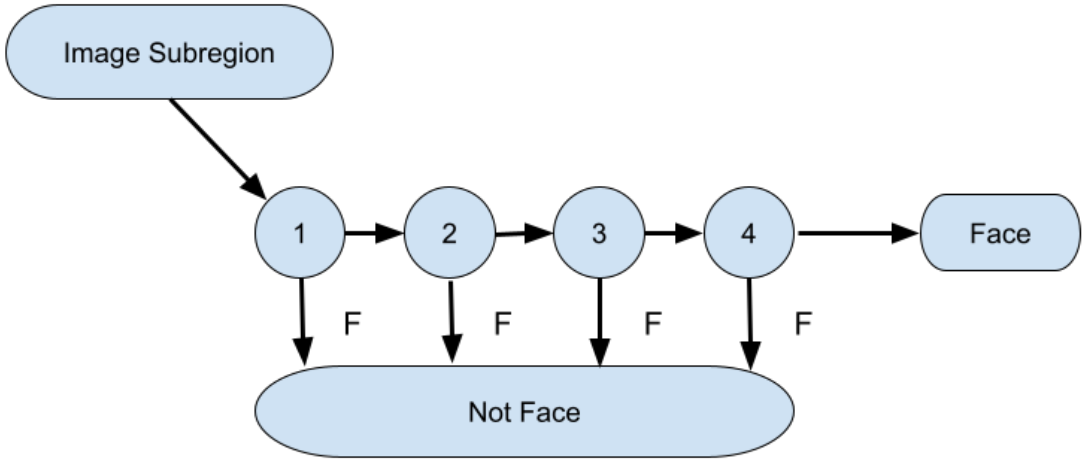


Figure 3: Cascade Classifier

A cascade classifier is a machine learning-based object detection technique used to identify specific objects or patterns in images or video frames. It was first introduced by Viola and Jones in 2001 and has since become widely used for real-time object detection tasks [9]. Cascade classifiers are especially popular for applications like face detection, pedestrian detection, and, in this project's context, license plate detection

Cascade classifiers are based on the AdaBoost algorithm, which combines multiple weak classifiers (simple decision rules) into a strong classifier. These weak classifiers are organized in a cascaded structure, where each stage of the cascade progressively filters out regions that are unlikely to contain the target object, leading to faster and more efficient detection. The training process of a cascade classifier involves providing positive samples (images containing the target object) and negative samples (images without the target object). The algorithm learns a set of features that best distinguish between positive and negative samples. During the detection phase, the cascade classifier sequentially evaluates these features, and if a region fails to pass a stage, it is rejected as a potential object candidate.

In this project, the Haar cascade classifier is a crucial component for license plate detection. By leveraging OpenCV's built-in Haar cascade classifier for license plate detection, the project benefits from a robust and efficient method for identifying regions of interest in the captured video frames. Cascade classifiers are fast compared to other deep learning models.

The cascade classifier helps filter out non-license plate regions, focusing the OCR process only on potential license plate areas. This step significantly improves the speed and accuracy of the license plate recognition, enabling real-time processing in a toll collection scenario. Moreover, the cascade classifier contributes to the overall efficiency of the project by minimizing false positives, reducing unnecessary OCR attempts, and optimizing the toll collection system's performance. By utilizing the Haar cascade classifier we have incorporated a well-established and effective object detection technique. Its ability to detect license plate regions accurately and swiftly aligns perfectly with the project's objective of automating the toll collection process, enhancing traffic flow, and providing an efficient and user-friendly system for users.

EasyOCR

Figure 4: OCR

EasyOCR is a Python library for optical character recognition (OCR) that provides a simple and user-friendly interface for recognizing text from images and video frames. It is built on top of the Tesseract OCR engine, making it easy to integrate into Python projects for text extraction tasks [10]. Its Key Features and Capabilities are,

Multilingual Support: EasyOCR supports a wide range of languages, making it suitable for text recognition in diverse scenarios. It allows users to recognize text in multiple languages, including English, Chinese, Spanish, French, and many more.

Simplified API: EasyOCR offers an intuitive and straightforward API, enabling users to perform OCR with just a few lines of code. The library handles the complexities of image pre-processing and text extraction, making it accessible to beginners and experienced developers alike.

Pretrained Models: EasyOCR comes with pre-trained models for various languages and tasks. These models have been trained on large datasets, allowing for accurate and reliable text recognition without requiring extensive training on custom data.

Real-Time Performance: One of the key advantages of EasyOCR is its real-time performance, which allows for efficient text recognition from images and video streams. This feature is particularly valuable for applications that require instant feedback and processing, as in our project's license plate recognition module.

In this project, EasyOCR serves as a critical component for recognizing the characters on the detected license plate regions. By leveraging EasyOCR's pre-trained models and multilingual support, the project gains the capability to accurately recognize text from license plates in various languages. This ensures that the toll collection system can efficiently handle a diverse range of vehicles and license plate formats, enhancing its usability and applicability. The simplified API of EasyOCR enables seamless integration with the project's existing codebase, allowing for smooth implementation and hassle-free text extraction from license plate images. With its real-time performance, EasyOCR enables swift text recognition, contributing to the overall efficiency of the toll collection system. It ensures that the license plate verification process is fast and responsive, providing a smooth experience for users passing through the toll plaza.

Requests

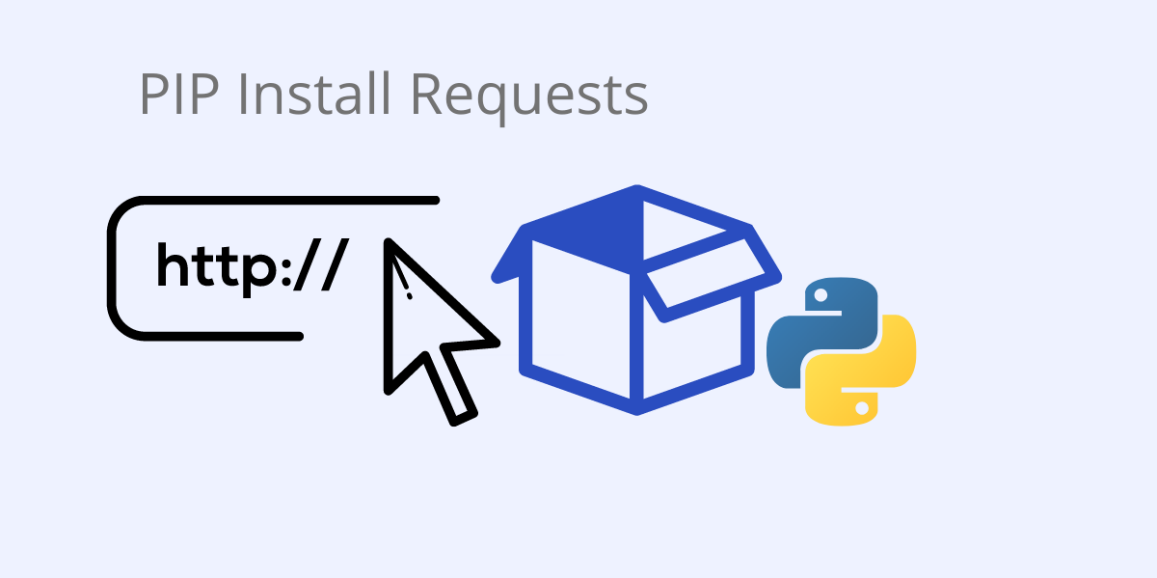


Figure 5: Request library python

The requests module in Python is a versatile and powerful third-party library used for making HTTP requests [11]. In the context of our project, the requests module plays a crucial role in facilitating seamless communication between our web server and other components of the toll collection system. Key Features and Advantages are,

Ease of Use: The requests module provides a simple and user-friendly API for sending HTTP requests and handling responses. Its intuitive syntax allows developers, even beginners, to quickly grasp its functionality and incorporate it into their Python applications effortlessly.

HTTP Methods Support: The module supports various HTTP methods, including GET, POST, PUT, DELETE, and more. This versatility enables us to interact with web services, APIs, and websites, making it a fundamental tool for creating a robust and interactive web application.

JSON and Form Data Support: requests simplify working with JSON and form data by automatically encoding and decoding data when sending and receiving HTTP requests. This streamlines data exchange between the web server and other components of the system.

The requests module in Python has proven to be an invaluable addition to our project. Its ease of use, a wide range of capabilities, and strong community support have enabled us to seamlessly interact with external APIs, manage HTTP requests efficiently, and establish secure and reliable communication between our web server and other components. By integrating the requests module into our Python application, we have simplified the process of making HTTP requests and handling responses, saving time and effort in the development process. Its flexibility and customization options have empowered us to tailor our HTTP requests to suit the specific needs of our toll collection system, enhancing its functionality and user experience. Overall, the requests module has been instrumental in ensuring smooth and efficient communication between different components of our toll collection system, contributing significantly to the success of our project.

PySerial

Figure 6: PySerial logo

PySerial is a widely-used Python library that provides a simple and powerful interface for working with serial ports[12]. In the context of our project, PySerial plays a critical role in establishing communication between the hardware module (Arduino Uno) and the web server, enabling seamless control of the gate opening and closing based on the verification status received from the server.

Serial Communication Support: PySerial facilitates easy communication with serial devices, such as microcontrollers, sensors, and other hardware components. It allows us to send and receive data through the serial port, making it an essential tool for interacting with our Arduino Uno microcontroller in the hardware module.

Configurable Baud Rates and Settings: The library enables us to configure essential parameters, such as baud rates, data bits, stop bits, and parity, to ensure reliable and error-free communication between the Python script and the Arduino Uno.

Non-Blocking Operation: PySerial supports non-blocking operation, allowing us to implement asynchronous communication. This feature prevents the Python script from getting blocked while waiting for data from the serial port, ensuring smooth and responsive system behavior.

PySerial has proven to be an indispensable component of our project. Its ability to facilitate serial communication with the Arduino Uno microcontroller has empowered us to control the gate opening and closing based on the verification status received from the web server. By integrating PySerial into our Python script, we have gained a reliable and efficient means of interacting with serial Hardware modules gate controlling arduino, ensuring seamless communication between the hardware module and the computer. Its cross-platform compatibility, configurable settings, non-blocking operation, and exceptional error handling have contributed significantly to the overall functionality and stability of our toll collection system. Overall, PySerial has proved to be a powerful and user-friendly Python library for serial communication, enhancing the capabilities of our project and reinforcing the seamless integration between hardware and software components.

Flask

Figure 7: Flask Logo

Flask is a popular and lightweight Python web framework used for building web applications and APIs [13]. In the context of our project, Flask serves as the backbone of the web server module, providing a simple and efficient way to create and manage the server-side functionality. Key Features and Advantages of the library are,

Minimalistic and User-Friendly: Flask is known for its minimalist design, making it easy for developers to get started quickly. Its simplicity allows for rapid development without unnecessary boilerplate code, making it ideal for small to medium-sized projects like ours.

Extensible and Modular: Despite its minimalistic nature, Flask is highly extensible and modular. It allows developers to integrate various extensions and libraries as needed, enabling the addition of new features and functionalities with ease.

Routing and URL Handling: Flask simplifies URL routing, making it straightforward to map specific URL patterns to corresponding Python functions. This feature allows us to define endpoints for handling incoming requests from the camera and detection module as well as the hardware module.

Templating Engine: Flask comes with a Jinja2 templating engine, allowing for the creation of dynamic HTML templates. This enables us to generate dynamic web pages, display real-time data, and respond to user interactions effectively.

Built-in Development Server: Flask includes a built-in development server, making it convenient for testing and debugging our web application locally without the need for additional server configurations.

RESTful API Support: Flask provides built-in support for creating RESTful APIs, which enables us to expose specific functionalities of our web server to external clients, such as the camera and hardware modules.

Flask has proven to be an excellent choice for our project. Its minimalist design and user-friendly API have allowed us to rapidly develop the web server module, streamlining the process of integrating the camera and detection module and hardware module into a cohesive system. With Flask's extensibility and modular nature, we have been able to seamlessly incorporate additional functionality, such as creating RESTful APIs and implementing custom authentication mechanisms. This adaptability has empowered us to tailor the web server to suit the specific requirements of our toll collection system. Overall,

Flask's ease of use, routing capabilities, templating engine, and support for RESTful APIs have been instrumental in creating a reliable, efficient, and user-friendly web server. By choosing Flask as the web framework for our project, we have achieved a scalable and maintainable solution, providing a seamless and responsive user experience to all stakeholders involved in the toll collection process.

HTML & Bootstrap

Figure 8: Html+ Bootstrap Logo

HTML (Hypertext Markup Language) and Bootstrap are essential tools for building modern and responsive web pages. They play a vital role in the front-end development of web applications, including our project. HTML is the standard markup language used to create the structure and content of web pages. It consists of a series of elements, each represented by tags, which define the layout and formatting of the page's content. HTML allows developers to create headings, paragraphs, lists, images, forms, and more, providing the building blocks for displaying information on a web page[14]. And, Bootstrap is a popular front-end framework that simplifies the process of designing responsive and mobile-friendly web pages. It includes a collection of pre-designed HTML, CSS, and JavaScript components and templates, allowing developers to create visually appealing and consistent web layouts quickly.[15]

HTML and Bootstrap are powerful tools in the front-end development of our project. HTML allows us to create the structure and content of our web pages, while Bootstrap enables us to design responsive and visually appealing user interfaces quickly. By leveraging the semantic structure of HTML and the pre-designed components of Bootstrap, we can create a user-friendly and consistent web interface for our toll collection system. The combination of HTML and Bootstrap empowers us to build a modern and accessible web application, ensuring seamless interaction with users across different devices and

screen sizes. Overall, these technologies contribute significantly to the success of our project by providing an intuitive and visually engaging user experience.

MySQL Database

Figure 9: MySQL Logo

MySQL is an open-source relational database management system (RDBMS) widely used for storing and managing structured data [16]. In the context of our project, we've planned to implement MySQL to serve as the backend database for storing information related to registered cars and their passing records. Due to time constrain we weren't able to do that.

3.1.2 Hardware:

Arduino Uno

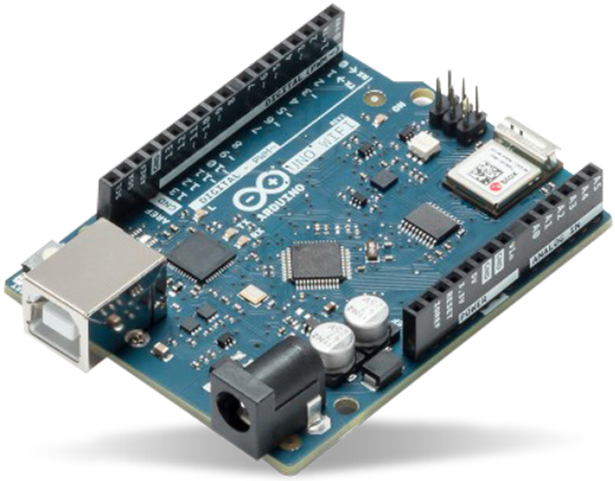


Figure 10: Arduino Uno

Arduino Uno is an open-source microcontroller development board based on the ATmega328P microcontroller[17]. It is widely used in various electronics and robotics projects due to its simplicity, versatility, and ease of use. In our project, the Arduino Uno serves as a crucial hardware module responsible for controlling the gate opening and closing based on the verification status received from the web server.

Specifications of Arduino Uno:

Table 1: Specifications of Arduino Uno

Microcontroller	ATmega328P – 8-bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Pinouts of the Arduino Uno Board:

Table 2: Pinouts of Arduino Uno

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power the microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.

Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Arduino Uno plays a vital role in our project, serving as the hardware module responsible for interfacing with the web server and controlling the gate's opening and closing. By utilizing Arduino Uno's GPIO pins, we can connect motors and sensors to control the gate's physical movement. The Arduino IDE, coupled with the vast community support and extensive ecosystem, enables us to program the Arduino Uno easily and integrate it seamlessly into our toll collection system. The simplicity and accessibility of Arduino Uno have made it a popular choice for many electronics projects, including ours, where it acts as a reliable and efficient bridge between the web-based interface and the physical hardware components. Overall, Arduino Uno's capabilities, affordability, and widespread popularity make it an ideal choice for the hardware module of our toll collection system, ensuring smooth communication and effective control of the gate based on the verification status received from the web server.

HC-SR04 sonar sensor

Figure 11: Sonar Sensor

The HC-SR04 ultrasonic sensor is a popular distance-measuring module commonly used in various electronic projects, including robotics and automation [18]. In our project, the HC-SR04 sensor serves as a crucial component in the hardware module, providing distance-measuring capabilities. The HC-SR04 ultrasonic sensor is a widely used distance-measuring module known for its accurate non-contact measurement capabilities. By emitting ultrasonic pulses and measuring the time taken for the pulses to bounce back from obstacles, the HC-SR04 provides reliable and precise distance measurements within a wide range, making it suitable for various applications. Its straightforward interface, low power consumption, and cost-effectiveness further contribute to its popularity in electronics projects, including our "Web Interface Based Automation of Toll Collection System Using Computer Vision." In our project, the HC-SR04 sensor integrated with the Arduino Uno serves as a crucial component for proximity detection at the toll gate, ensuring safe and efficient toll collection operations. This sensor makes sure that no cars can pass without toll and also the toll bar doesn't collide with the car.

The Pinout of HC-SR04:

Table 3: Pinouts of Sonar Sensor

Pin Name	Description
VCC	Power supply (5V)
GND	Ground (0V)
Trig	Trigger Input
Echo	Echo Output

Servo motor

Figure 12: Servo Motor

The servo motor is a commonly used electromechanical device that converts electrical signals into rotational motion [19]. In the context of our project, the servo motor serves as a crucial component in the hardware module, allowing controlled movement of the gate based on the verification status received from the web server. The servo motor is a compact and lightweight electromechanical device that offers precise angular positioning and operates in a closed-loop control system. It can be easily interfaced with microcontrollers using a control signal, such as PWM, to specify the desired position or angle of rotation. The servo motor's variable speed and torque, along with its wide range of applications, make it a versatile choice for tasks requiring controlled and accurate movements. In our "Web Interface Based Automation of Toll Collection System Using Computer Vision" project, the servo motor plays a crucial role in controlling the gate's movement based on the verification status received from the web server. Its precise positioning ensures reliable gate access for authorized vehicles, contributing to the system's seamless toll collection and enhanced security.

In our project, the servo motor is utilized to control the gate's movement based on the verification status received from the web server. When the server confirms the validity of a vehicle's license plate, it sends a signal to the hardware module, which then activates the servo motor to open the gate. Conversely, if the verification fails, the servo motor is triggered to close the gate. By leveraging the precise positioning capabilities of the servo motor, we ensure that the gate opens and closes accurately and reliably, providing seamless access to authorized vehicles and maintaining the security of the toll collection system.

Table 4: Pinouts of servo Motor

Pin Name	Description
Signal (S)	Control signal input (PWM)
Power (VCC)	Power supply (4.8V to 6V)
Ground (GND)	Ground (0V)

With that the brief review of used components and software section is concluded.

3.2 System Design and Implementation

To finalize a system design for the project design parts had to be divided into 4 Module. Those are Detection and Identification Module, Web Verification server module, Hardware Module, and Web Interface module. These modules had to be integrated in a seamless way

that works in harmony and also it is really easy to use for the users. An Abstract Data Flow and Module Diagram for that would be,

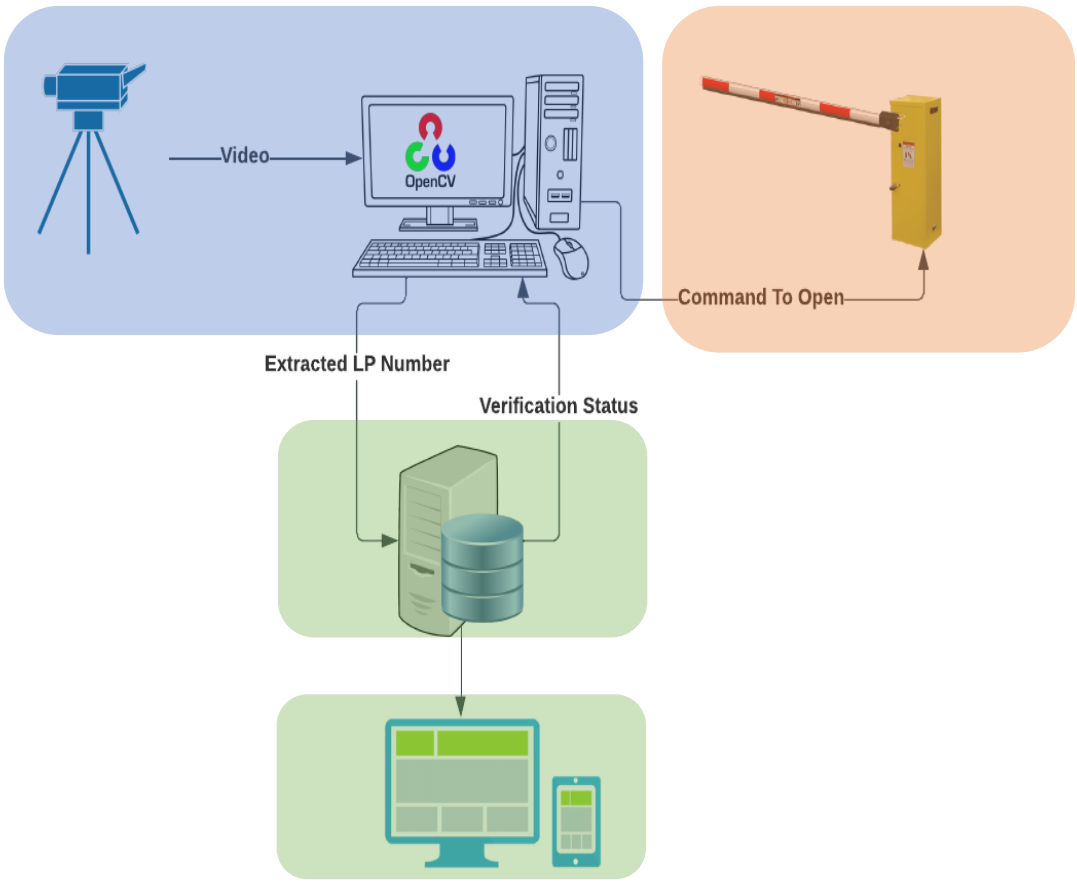


Figure 13: An Abstract Diagram of the system

Finding out the most suitable technology and Implementing it for each section so that each section works seamlessly is the challenge this methodology will address. For that we have throughout the project which works best module by module then we have integrated the whole system.

3.2.1 Module-1 – Camera and Detection Module:

To successfully toll a vehicle in a toll booth first we have to identify that vehicle and use that identification to bill the vehicle owner. In this module, we'll address the challenge of identifying a car or its owner and then bill him automatically. That identification process can be manual or automatic process using,

RFID:

It stands for Radio Frequency Identification. It is a technology that uses radio waves to automatically identify and track objects or individuals. RFID systems consist of three main components: RFID tags, RFID readers, and a backend database or system. RFID tags are small electronic devices that contain a unique identifier and can be attached or embedded in objects, products, or even people. These tags consist of an integrated circuit and an antenna. They come in various forms, such as adhesive stickers, cards, or even implantable chips. Each RFID tag carries a specific identification code that can be read wirelessly by RFID readers. RFID readers are devices that emit radio waves and capture the signals transmitted by RFID tags. These readers can be fixed or handheld and are equipped with antennas to communicate with the RFID tags within their range. When an RFID tag comes into proximity with an RFID reader, it receives the reader's radio wave signal, powers up, and transmits its unique identification code back to the reader.[20]

RFID is suitable for the Identification of a person. However, this technology is not well suited for us due to its manual nature. Out goal is to automate the process and RFID hinders that goal. To use RFID tag a user has to stop and then use it. This will introduce manual work and that is not suitable in a moving place like a road.

QR Code Scan:

A QR code, short for Quick Response code, is a two-dimensional barcode that consists of a matrix of black and white squares. QR codes were developed in 1994 by Denso Wave, a subsidiary of Toyota, and have gained widespread popularity due to their ability to store large amounts of information and their ease of use. QR codes can be scanned and read by smartphones, tablets, or dedicated QR code readers using the device's camera or a dedicated scanning application. When scanned, the encoded information within the QR code is decoded and processed by the scanning device.[21]

QR codes can store various types of data, including text, URLs, contact information, email addresses, phone numbers, and more. They can also be used to perform actions like opening a website, initiating a phone call, sending an email, or providing access to Wi-Fi networks. QR codes can contain up to several hundred alphanumeric characters, allowing for the storage of complex information. The square shape and the pattern of black and white squares within a QR code are designed to enable efficient scanning and error correction. The code contains alignment patterns, timing patterns, and error correction codes, which

help ensure accurate decoding even if the code is partially damaged or distorted. The use of QR codes has become widespread in various applications, including marketing, advertising, ticketing, product packaging, payment systems, and contactless check-ins. QR codes offer a convenient way to quickly access information or perform actions by simply scanning the code with a compatible device. QR codes have gained even more popularity with the increasing adoption of smartphones and the availability of QR code scanning apps. They provide a simple and efficient way to bridge the physical and digital worlds, enabling users to interact with the encoded information or perform actions without the need for manual input or typing. The QR code is a very popular method for billing but like the RFID due to its manual nature, we didn't use this solution.

Camera-Based Computer Vision:

Computer vision is a field of artificial intelligence and computer science that focuses on enabling computers or machines to gain a high-level understanding of visual data, similar to how humans interpret and analyze visual information. It involves the development and application of algorithms and techniques that allow computers to acquire, process, analyze, and interpret visual data from images or videos. The goal of computer vision is to extract meaningful information and insights from visual data to facilitate decision-making, automation, and understanding of the visual world. It aims to enable machines to "see" and comprehend the contents, context, and characteristics of visual inputs, such as images, videos, or live camera feeds.

Computer vision techniques will be employed in this project to enable the automated detection and recognition of license plates. Computer vision involves the development and application of algorithms to process, analyze, and interpret visual data from images or videos. The project will utilize image processing, machine learning, and deep learning methods to extract meaningful information from the captured video footage. These techniques will facilitate tasks such as license plate detection, character recognition, and verification against a database of registered vehicles. The application of computer vision will enhance the accuracy and efficiency of the toll collection system, enabling seamless automation and reducing traffic congestion.

Chosen Solution:

For this project, we've chosen OpenCV Python. Our primary objective is to set a camera and take a video input and then get a number plate which is a text from the output just as shown in the diagram,

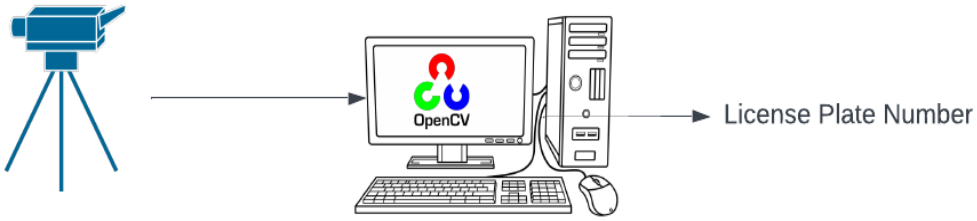


Figure 14: Detector Section

To reach that objective we have to code the algorithm in step by step manner. Those steps are given below,

Step-0 – Pre-processing: An image will have many frames with blurry or noisy foregrounds or objects that will hinder the efficiency of our Number plate recognition. So to avoid that we do some pre-processing to our image. Below is a given example of preprocessing.

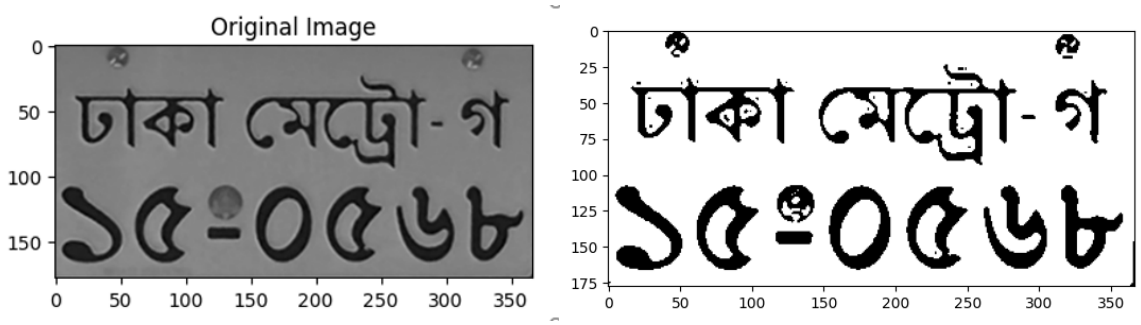


Figure 15: Pre Filtering of image

Step-1 – Select a Video Source: This project is a Realtime project so our video source also has to be Realtime and because of that we have limited options. We need a video camera for our purpose. We used our laptop's webcam for experimentation. To yield the best result A good-quality video camera is recommended.

Step-2 – Separate each frame of picture from the video: For that, we have a convenient method provided `.read()`. To use this method, we had to first create an object of the cap

using `cv2.VideoCapture.` This method returns an object of a video stream from a source. Our testing chosen source is our laptop's webcam.

Step-3 – Classifier Model to Detect Exact Region of image: There is plenty of different classifier models that can detect a license plate from an image. Those models are RNN, CNN, YOLO, etc deep learning models and architecture. We've used a cascade classifier for our project due to its speed and accuracy. A Russian Number Plate Recognizer was used to detect the license plate. In the figure below we can see an example of the classifier detecting the model. The detected region is marked with a purple bounding box in Figure-3.

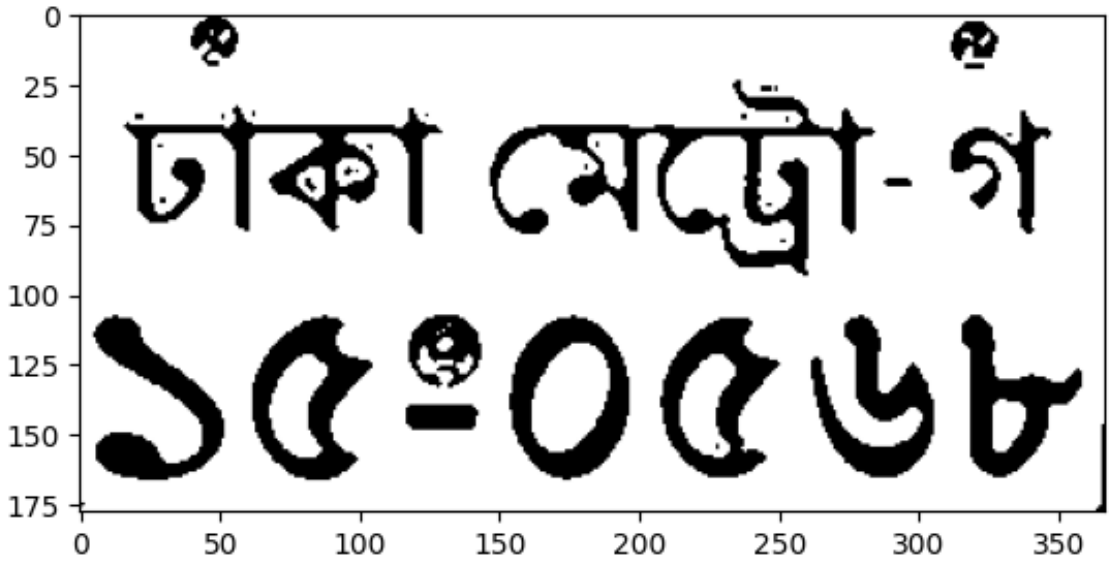
Figure 16: Detection of a License plate region from an image

Step-4 – Extracting the region of interest: Once we get the location of our license plate in the image matrix, we use a matrix manipulation technique to cut the exact region and copping out the rest of the region. In Figure-4,5 We can see the actual image vs the chopped image.

Figure 17: Raw Image

Figure 18: Extracted Image

Step-5 – Applying OCR: Once the ROI is extracted, we use EasyOCR Library to get the text from the license plate which was our objective. Easy OCR gives out the plate number. Which is an array of text. We take the text and concat it to keep it in a single line. Once the processing is done, we are ready to send the verified text to the next phase. The Inputs and outputs of the OCR function are in the figure below,



```
#easyocr extracting just image
image_array = np.array(thresh)

# Recognize characters in the image
result = reader.readtext(image_array, detail=False)
# text = ''.join([re.sub('[^a-zA-Z]+', '', r[1]) for r in result])
print(result)
```

```
['ঢাকা মেট্রো-গ', '১৫২০৫৬৮']
```

Figure 19: OCR extracts the text

Step-6 – Sending the text to Verify: Once we've extracted the and formatted the text the next step is to verify. We send the formatted text to our web verification server and that server responds with exists or not exist. That communication happens via http protocol and server has an api endpoint of `"/api/verify"`

Step-7 – Receiving the Verification Status from Verification Server: Once the server gets the text that was sent by step-6 and verifies that using its database of car_database and then if the car number is verified it sends a JSON response of `{exists: true}`. And Step-6 processes that information for step-8.

Step-8 – Sending the Gate Opening Signal: This section could be done in the server as well. Once the Detector app receives confirmation it sends a signal to the next step of hardware section.

So, A simple flow diagram of all of the processes is in the given Figure below

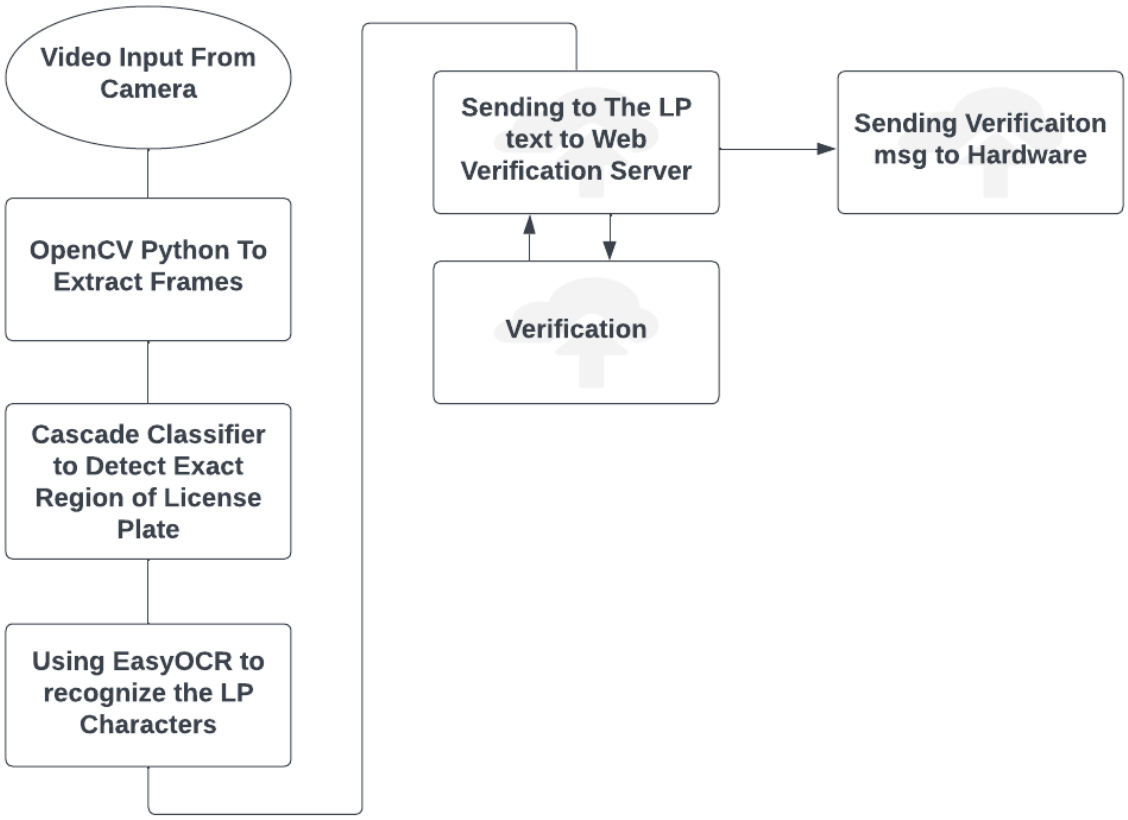


Figure 20: Flow Diagram of Detection Module

3.2.2 Module-2 – Web Verification Server:

Types of server choices:

In the development of our "Web Interface Based Automation of Toll Collection System Using Computer Vision," we explored different server-side technologies, each with its strengths and weaknesses. These technologies included Node.js, PHP, Django, and Flask, all of which play significant roles in web development and have their unique characteristics.

NodeJS:

Node.js is a server-side platform built on Chrome's V8 JavaScript engine. One of its key advantages is its asynchronous and non-blocking nature, which allows it to efficiently handle multiple concurrent connections without blocking the execution of other tasks. This makes Node.js highly suitable for applications requiring real-time data exchange or handling many connections simultaneously, such as chat applications or streaming services. Additionally, Node.js leverages JavaScript on both the server and client-side, which promotes code reusability and simplifies development for teams familiar with the language.

Moreover, Node.js benefits from a vast collection of open-source packages available through npm (Node Package Manager), enabling developers to quickly integrate additional functionalities into their applications.[22]

However, Node.js has some drawbacks to consider. While it excels at handling a large number of connections, it may face challenges with computationally intensive tasks due to its single-threaded event loop. Consequently, tasks that require significant processing power might slow down the server's response time. Furthermore, NodeJS is not suitable for us due to its Declarative nature. Its not simple to setup and use and it is hard to maintain and costly to host.

PHP:

PHP (Hypertext Preprocessor) is a popular server-side scripting language known for its simplicity and ease of learning. It is widely supported, with an extensive community and resources available for developers. PHP is particularly well-suited for rapid application development, making it an excellent choice for projects with time constraints or a need for quick prototyping. Additionally, PHP's language design allows for straightforward integration with HTML, simplifying the process of embedding dynamic content within web pages.[23]

However, PHP also has its drawbacks. Performance can be a concern for high-traffic applications since PHP traditionally runs on a synchronous, single-threaded model. This may result in performance bottlenecks, especially for applications with many concurrent users or computationally intensive operations. Due the the same reason as NodeJS we didn’t chose PHP for this project.

Django:

Django is a high-level Python web framework known for its robustness and feature-rich nature. It provides a comprehensive set of built-in features, including an Object-Relational Mapping (ORM) system and an admin interface, allowing developers to build sophisticated and data-driven web applications more rapidly. Django's emphasis on security also makes it a popular choice for applications that handle sensitive information.[24]

With its vast feature set, we didn’t choose Django setting up and then coding small scale things in Django is not worth the time. There is a lot of building things that are unnecessary for our usage.

Chosen Solution Flask

After careful consideration of the advantages and disadvantages of each server technology, we decided to use Flask for our toll collection system. Flask is a micro web framework for Python, known for its simplicity and minimalistic design. Its lightweight nature makes it easy to learn and use, making it an ideal choice for small to medium-sized projects where simplicity and speed of development are priorities.

Flask's flexible and unopinionated structure allows developers to choose their preferred tools and libraries for various tasks, offering the right balance between providing essential functionality and allowing developers the freedom to implement custom solutions. For our project, Flask's simple and easy-to-understand syntax aligned well with our team's expertise and the project's requirements, making it a clear and optimal choice for our web-based automation project. With Flask, we can efficiently implement the required functionalities without unnecessary complexity, achieving a seamless and effective toll collection system with web interface and computer vision capabilities.

From the first Module we are sending a licence plate number text that will be received and then verified in this Module. In simple terms we want everything like the figure,

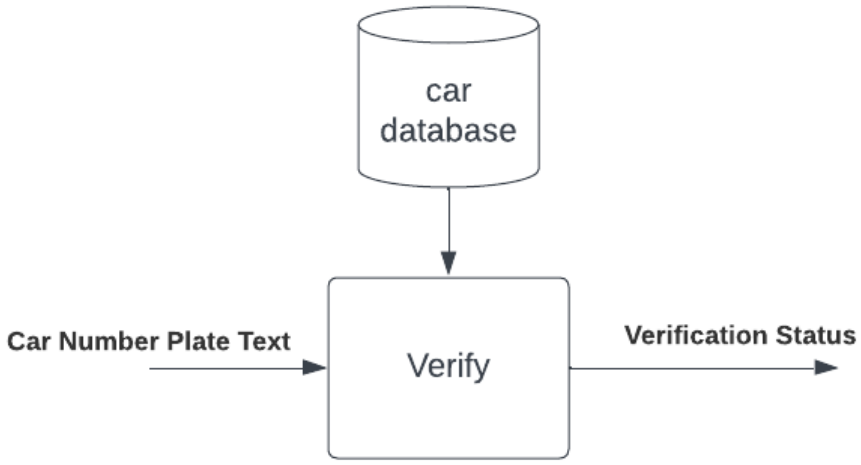


Figure 21: Simple block diagram of system needs

To Achieve the result we have to proceed step by step. Those steps are,

Step-1 – Receive Data in an Endpoint: Frist we need to receive the license plate number which is a text type data from the Detection module. To do that the server needs an endpoint where the Detection could send `GET` request and for that we create an API endpoint in

the location of `"/api/verify"`. The API receives a JSON object named `{text: "plate_number"}` And in that object the `plate_number` will be a string containing the license plate number.

Step-2 – Verification with Data Structure: This project uses a python list data structure that holds all the data of registered cars. The plate number we receive from the **Step-1** will be verified using the `car_number` database and if verified it'll subtract an amount from the user on the basis of `vehicle_type` field. The vehicle type field will have a string containing which type of vehicle this number plate has. We have a `Entry_database` and we also verify that if the same car has requested immediately before or not. Same car requesting again and again means it's false detection. When false detection happens we don't count that. We only count the first request of a single vehicle. And the balance will be deduced from the `Balance` field of the database. This balance field can be filled with advance payment gateway like **Bkash, Visa card** etc. For demonstration purpose we've hardcoded a balance to each user of 5000tk and the amount will be subtracted from that. For real time application this data structure will have to be migrated into a **MySQL** Database but due to time constrain this project did not used that. The data structure looks like below figure,

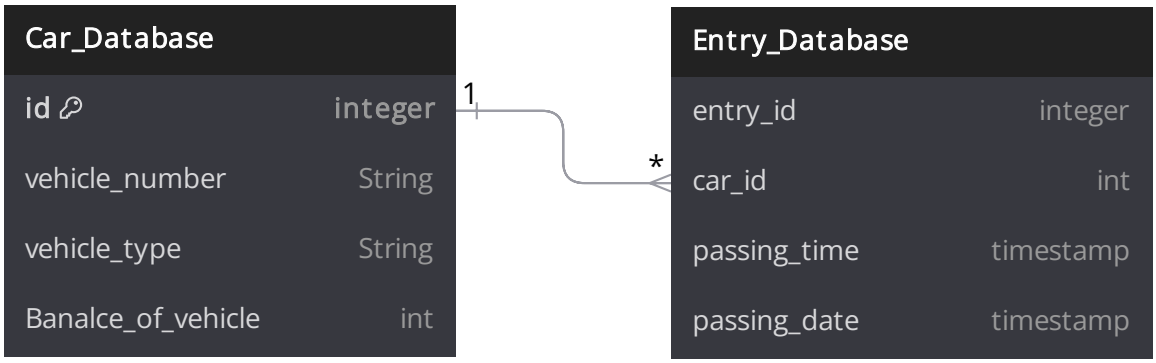


Figure 22: Relational Database Structure

Step-3 – Saving an Entry: Once a vehicle is verified in pseudo term it means the vehicle has passed the toll plaza. So we save an instance of vehicle from `Car_Database` into `Entry_Database` with some additional information like **passing time** and **passing date.** This relation in relational database management term is also known as Many to One relationship.

The total flow diagram of the section is in the figure below,

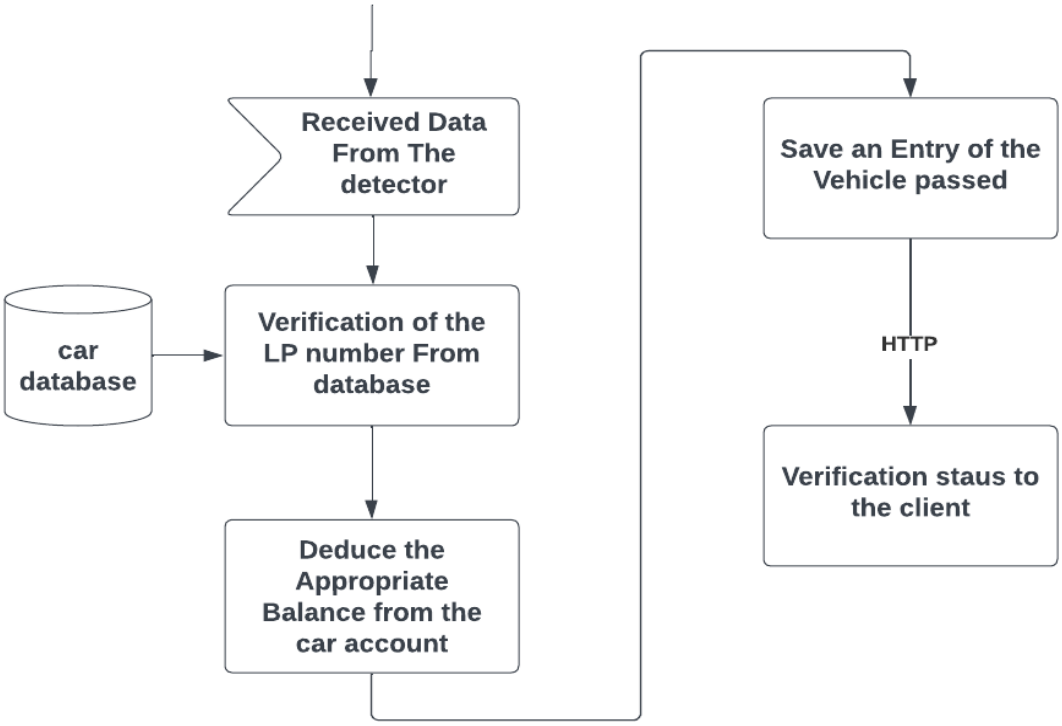


Figure 23: Flow diagram of Verification Server Section

3.2.3 Module-3 – Hardware:

Types of Hardwar Choices:

For hardware the choice was simple to receive the signal we need to use a microcontroller that can communicate in **Serial communication** with the computer and there wasn't much valid and easy choice other than Arduino. And for detection of vehicle and We could've used IR sensor or Sonar sensor. But availability and ease of use dictated the usage of sonar sensor. And for the raising the pole we've used a scaled model so the choice of which motor we use doesn't matter for actual application. However, a short description of the options is given below,

PLC:

Figure 24: Non Modular PLC

PLC, or Programmable Logic Controller, is a specialized industrial computer used to automate and control various processes in manufacturing and industrial environments. It is designed to withstand harsh conditions and is widely used in factories, plants, and production lines. PLCs receive input signals from sensors, process the data using a pre-programmed logic, and then generate output signals to control actuators and devices. They are highly reliable, scalable, and offer real-time control capabilities. PLCs play a crucial role in automating complex processes, improving efficiency, and ensuring safety in industrial settings. Their flexibility and programmability make them an essential component in modern industrial automation systems.[25] PLC will most likely be used in such application but for our development prototype we didn't used that because of its closeness and expensiveness.

ESP32:

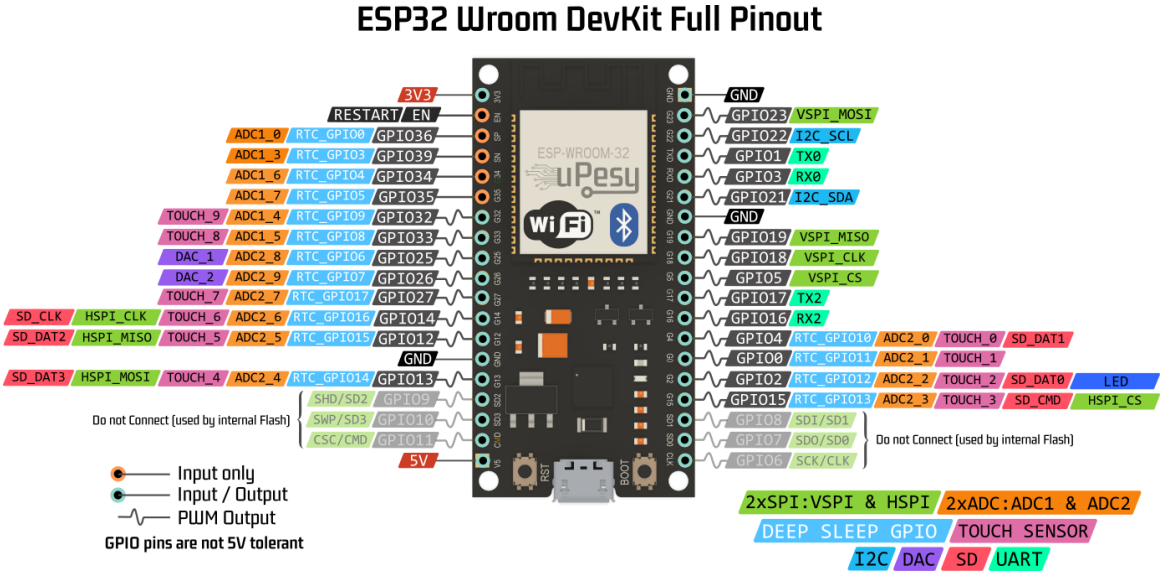


Figure 25: ESP32 Picture with pinout

The ESP32 is a versatile and powerful microcontroller module widely used in Internet of Things (IoT) applications. It integrates a dual-core processor, Wi-Fi, Bluetooth, and various GPIO pins for sensor interfacing. Its compact size and low power consumption make it suitable for portable and battery-powered projects. The ESP32's rich feature set, including its ability to support real-time operating systems and internet connectivity, has made it a popular choice among developers for creating smart devices, home automation systems, and remote monitoring solutions. With its ease of use and extensive community support, the ESP32 continues to drive innovation in the IoT landscape. In our application we didn't lean on this microcontroller due to we have already Arduino uno available. This Microcontroller is a little hard to set up with IDE and different sensor modules but it is far superior in terms of functionality and features. [26]

IR- Sensor:

Figure 26: An IR sensor

An IR (Infrared) sensor is a device that detects and measures infrared radiation emitted by objects. It is commonly used in various applications, including proximity sensing, motion detection, and temperature measurement. IR sensors consist of an IR emitter and an IR receiver, which work together to detect the presence or absence of an object based on the emitted and reflected infrared radiation. When an object is within the detection range of the IR sensor, it reflects the emitted infrared radiation back to the IR receiver. The sensor then interprets this reflected signal, and based on the intensity or time taken for the signal to return, it determines the presence or distance of the object. IR sensors find extensive use in automation systems, robotics, security systems, and electronics devices. Their non-contact nature, low power consumption, and ability to work in various lighting conditions make

them popular for a wide range of applications. IR sensors are valuable tools for enhancing automation, safety, and convenience in both industrial and consumer-oriented settings [27]. In our projects case coding IR sensor was inconvenient and we already had a **sonar sensor** from previous projects and because of that this project opted to use sonar sensor.

Chosen Solution: Arduino with Sonar sensor and Servo Motor

In this module our objective is to get a signal and safely open the gate so that no car gets harmed and no extra car can pass. To ensure latter feature we will program a logic where the Gate Will automatically close every time a car pass. But to ensure that somehow the gate bar doesn't damage the car we use a sonar sensor to detect if the car has passed or not. This process of Coding can be better understood by Seeing the logic diagram in the figure below,

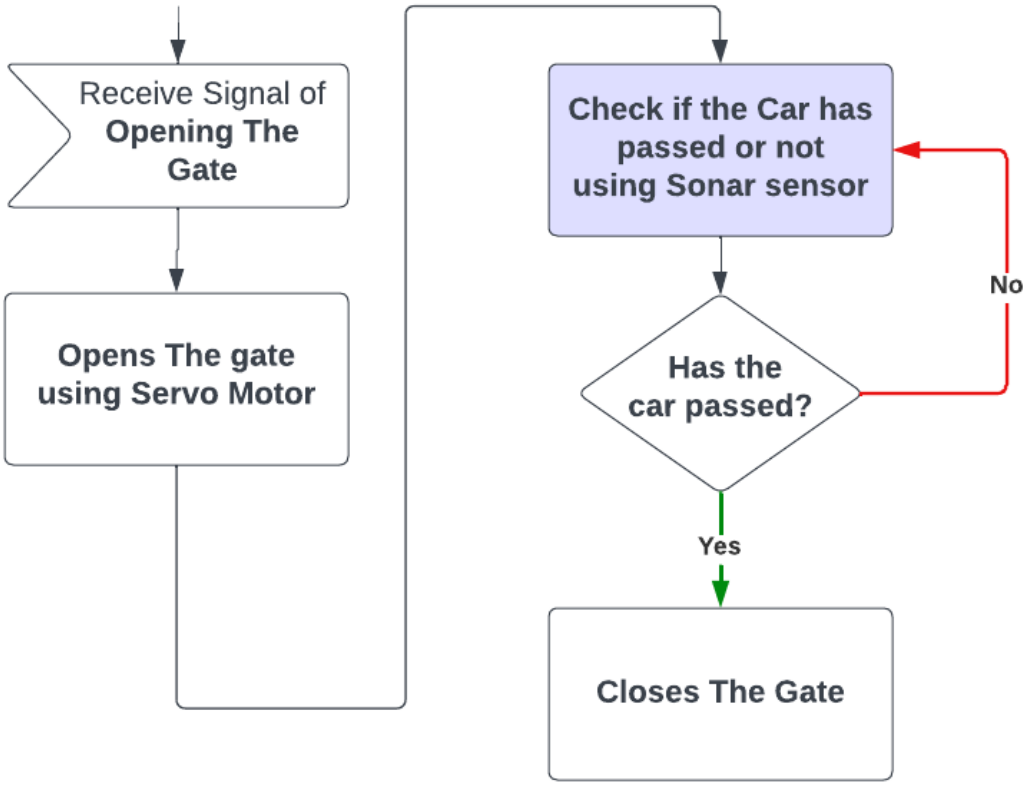


Figure 27: Logic Diagram of the Code

The connection setup of our Hardware section looks like below figures,

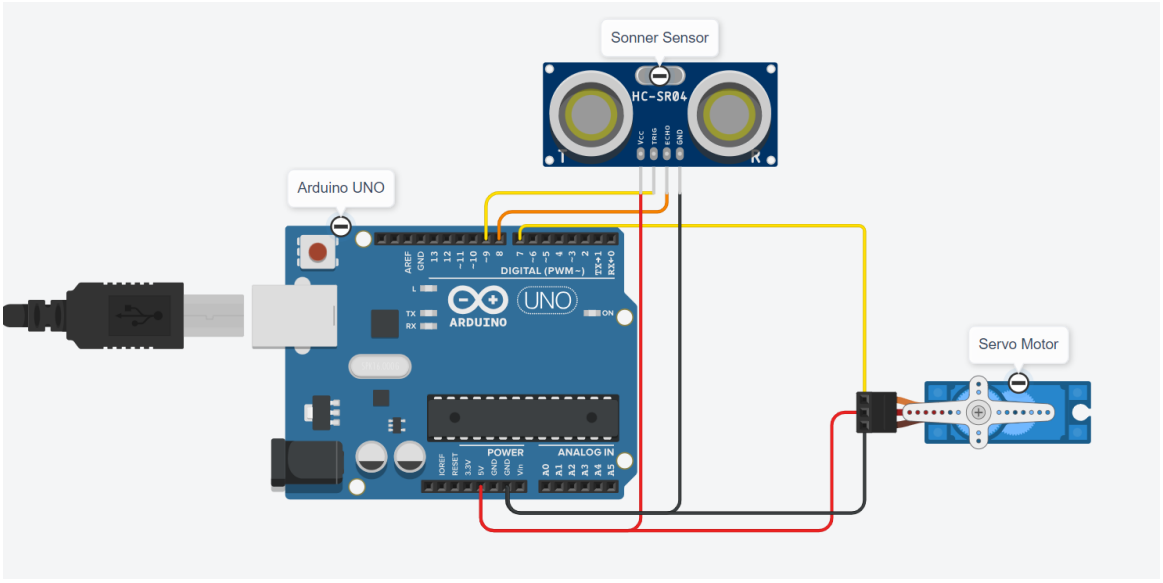


Figure 28: Tinker CAD simulation connection

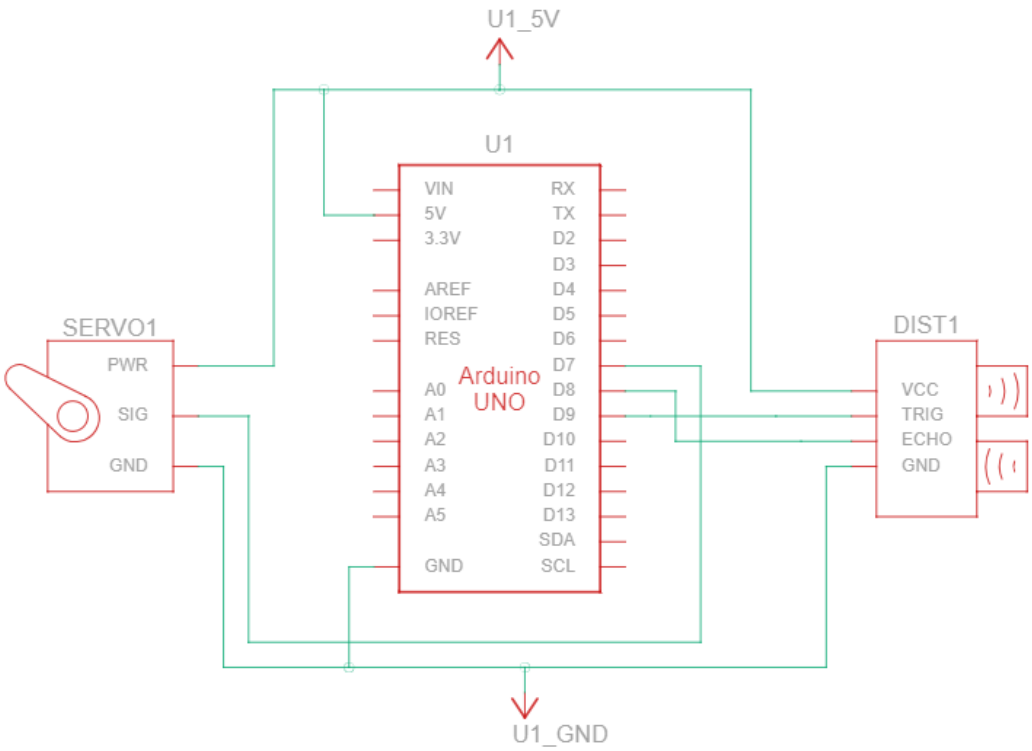


Figure 29: Pin Diagram of the connections

3.2.4 Module-4 – Web Interface Module:

To make a website we don't have any other choice than HTML and CSS. We've used a CSS library called bootstrap for better looking UI and save time. The Interface is served

through our previously setup **Flask** server from the verification module. That same server will serve the web interface to save complexity, expense and time. HTML and Bootstrap will be served through **Flask** templating engine which is **Jinja2.** The process is so simple for this module it is given in below figure

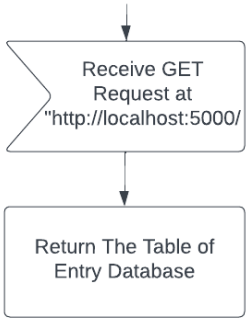


Figure 30: Flow Diagram of Interface Section

The interface looks like below figure,

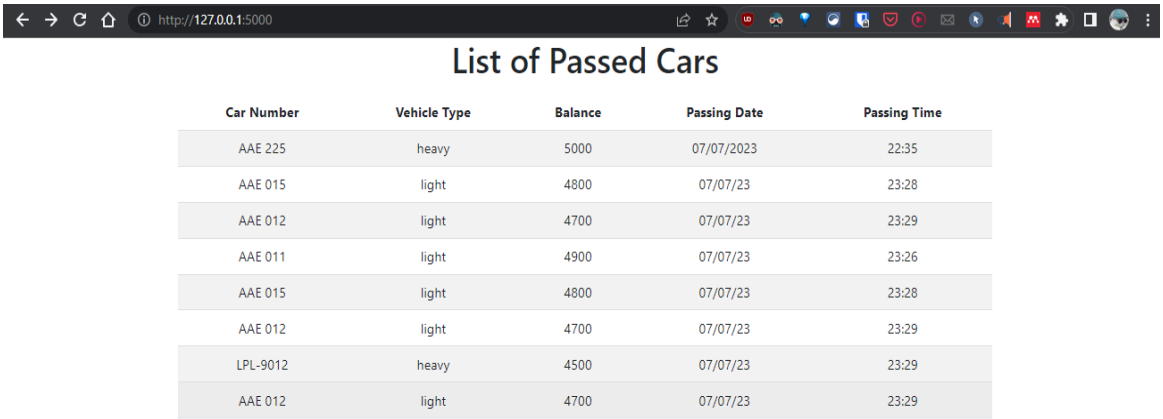


Figure 31: A Screenshot of the interface

3.3 INTEGRATION

The project has 4 Module but 3 Sections and Each section has their own tech stack. Those sections are “Arduino Hardware”, “Computer Vision Detector”, “Web Server”. Those section has to communicate each other seamlessly. The total file structure of the project is given in below figure

```
Project Folder
|
├── Arduino Hardware
```

```
│   └── gateControl
│       └── gateControl.ino
│
├── Computer Vision Detector
│   ├── model
│   │   └── haarcascade_russian_plate_number.xml
│   ├── number_plate.py
│   └── requirements.txt
│
└── Web Server
    ├── flask_app.py
    ├── requirements.txt
    └── templates
        └── index.html
```

The main project folder contains three main subfolders: "**Arduino Hardware**," "**Computer Vision Detector**," and "**Web Server**." The "Arduino Hardware" folder contains the Arduino code for the gate control module of the toll collection system, located in the "gateControl" subfolder. The "Computer Vision Detector" folder contains the necessary model file for the number plate detection, located in the "model" subfolder. The "number_plate.py" script contains the code for the computer vision detector module, and "requirements.txt" specifies the required Python dependencies. The "Web Server" folder contains the Flask web server application code in "flask_app.py." The "requirements.txt" file lists the required Python dependencies, and the "templates" subfolder contains the "index.html" file, which is the HTML template for the web interface.

3.3.1 Setup:

The Setup of the whole experimental process is given below,

1. Arduino Hardware:

- Downloaded and install the Arduino IDE from the official website (https://www.arduino.cc/en/software) to program the Arduino board.
- We made the structure with cardboard to replicate a toll booth in small scale.
- Connect the Arduino Uno board to the computer using a USB cable.
- opened the "gateControl.ino" file located in the "Arduino Hardware/gateControl" folder using the Arduino IDE.
- Then we upload the code to the Arduino Uno board.

2. Computer Vision Detector:

- Ensured that Python is installed on the computer.
- Installed the required Python dependencies by running the following command in the terminal or command prompt in the "Computer Vision Detector" folder:

```
pip install -r requirements.txt
```

3. Web Server:

- Ensuring Python installed on the computer.
- Installed the required Python dependencies for the web server by running the following command in the terminal or command prompt in the "Web Server" folder:

```
pip install -r requirements.txt
```

4. Computer Vision Model:

- The "Computer Vision Detector" uses the Haar Cascade model for number plate detection. The required model file "haarcascade_russian_plate_number.xml" is located in the "Computer Vision Detector/model" folder. This file can be found online. We made sure to keep it in place before addressing.

5. Web Interface:

- The web interface is built using Flask. Ensure that all the required files, including "flask_app.py" and the "templates" folder with "index.html," are present in the "Web Server" folder.

Once we completed the setup, the toll collection system is ready to operate with our scaled hardware. The Arduino hardware module controls the gate based on the received verification status from the web server. The computer vision detector module captures video from the camera source, extracts number plate text, and sends it to the web server for verification. The web server verifies the license plate with the database and returns the verification status to the hardware module for gate control. The web interface provides real-time information about passed cars and registered vehicles in the system.

3.3.2 Data Flow and Communication Protocol:

This project has a lot of part and all those parts has flowing data to pass information. Each section uses different modes of communication to communicate with another section, that communication diagram is given below,

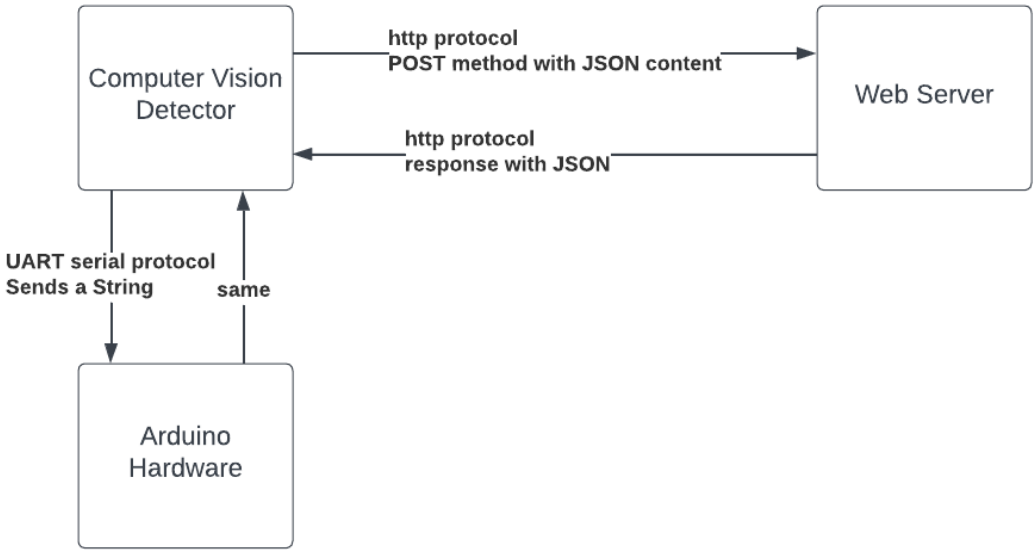


Figure 32: Communication Block Diagram with protocols

Chapter 4

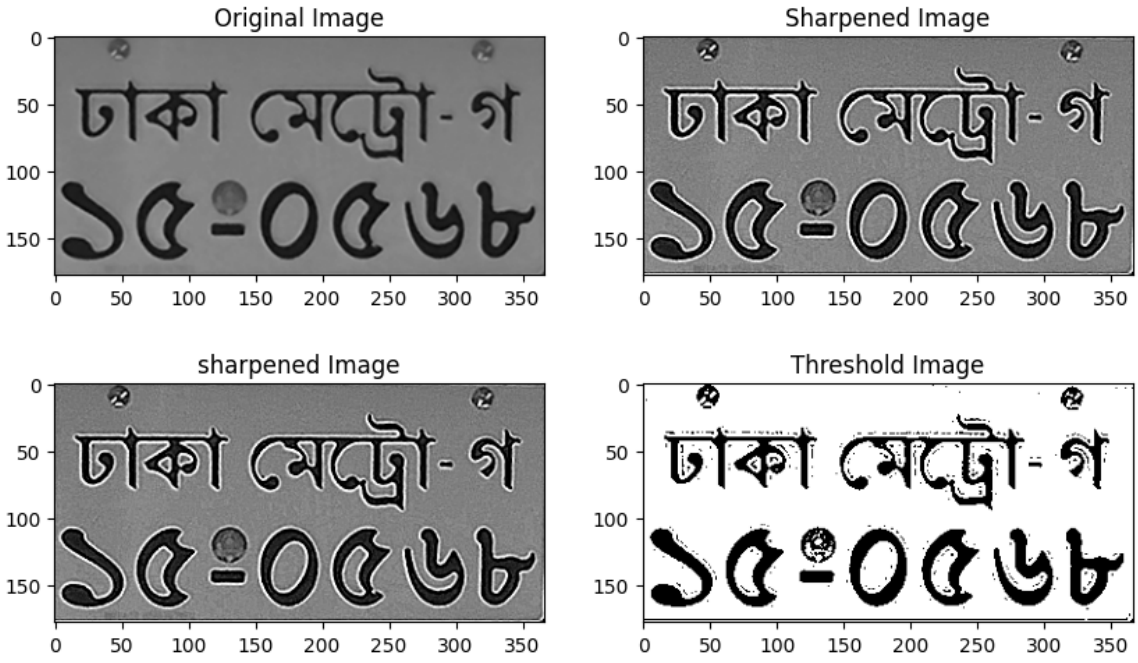
EXPERIMENTATIONS AND EXPERIMENTAL RESULTS

4.1 EXPERIMENTATIONS:

We did a lot of experimentation before finalizing our project and building the project. Some of those experimentations are

4.1.1 Filtering

The image we got after classification through cascadc classifier was not clear due to our webcam being low resolution. We had to denoise and then apply some image processing technique to sharpen and make those image clear. Some examples for Filtering is given below,



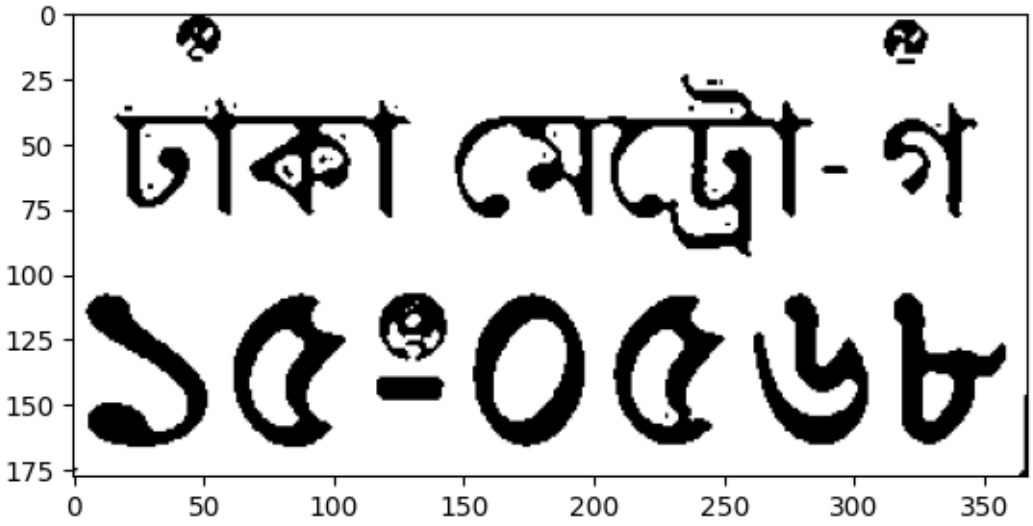


Figure 33: Applying Various Image Filtering Techniques

4.1.2 Testing OCR For English

We've used EasyOCR pretrained models to extract the text. However, EasyOCR for bangla isn't reliable enough for low resolution video. So to test our project first we have tried to extract English license plate text first. Some examples are given below,

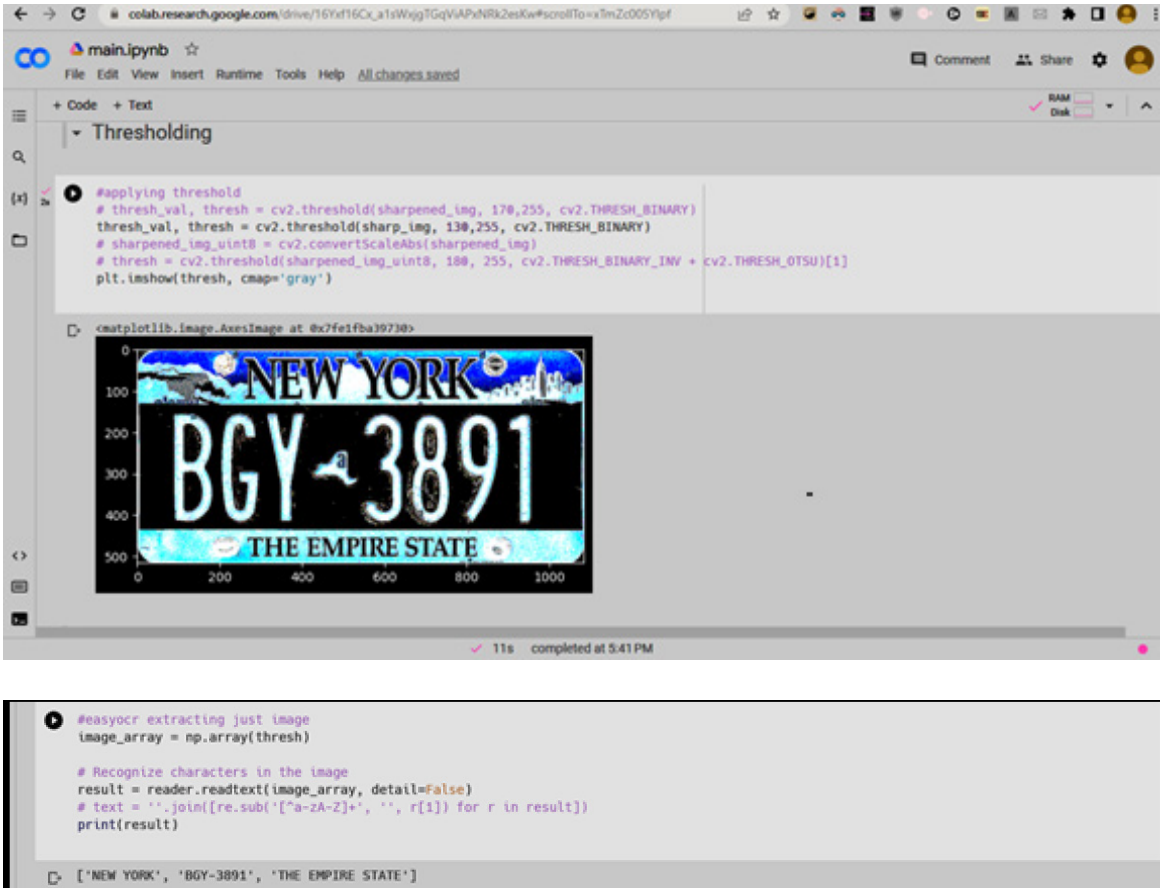
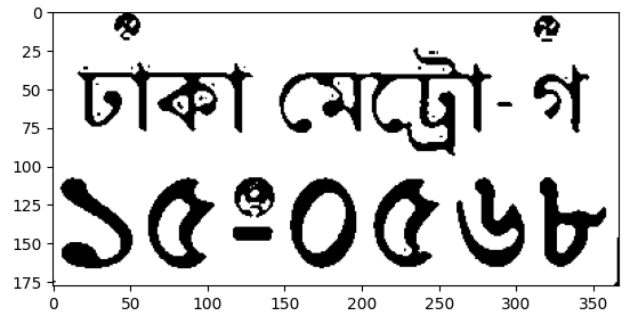


Figure 34: Extracting English text from an image

4.1.3 Testing OCR for Bangla

As mentioned before because of the unreliableness of bangla language for camera constrains we didn't implement Bangla in this project. But we did tested bangla language OCR and it worked Somewhat Accurately. The image is shown below,




```
#easyocr extracting just image
image_array = np.array(thresh)

# Recognize characters in the image
result = reader.readtext(image_array, detail=False)
# text = ''.join([re.sub('[^a-zA-Z]+', '', r[1]) for r in result])
print(result)
```

```
['ঢাকা মেট্রো-গ', '১৫-০৫৬৮']
```

Figure 35: Extracting Bangla text from an Image

4.1.4 Getting License Plate text from Video:

From a Printed paper or mobile phone screen we've hold this picture in front of the camera and we've attained somewhat accurate results. Both result and picture is given below,


```
License Plate:
License Plate: MD AAE 011
License Plate:
License Plate: MD AAE 011
License Plate:
License Plate: Md AAE 011
License Plate:
License Plate: d AAE 011
License Plate:
License Plate: HD AAE 011
License Plate: AAE 011
License Plate: AAE 011
License Plate: AAE 011
License Plate:
License Plate: AAE 011
License Plate:
License Plate:
License Plate: AAE 011
License Plate: AAE 013
```

Figure 36: Getting license Plate number from a Video Stream

4.2 Project Experimental Results:

For Hardware demonstration We've built a small scaled version with cardboard to show the concept and show the integration works fine here are some pictures and moments of that scaled replica,

4.2.1 Full Setup

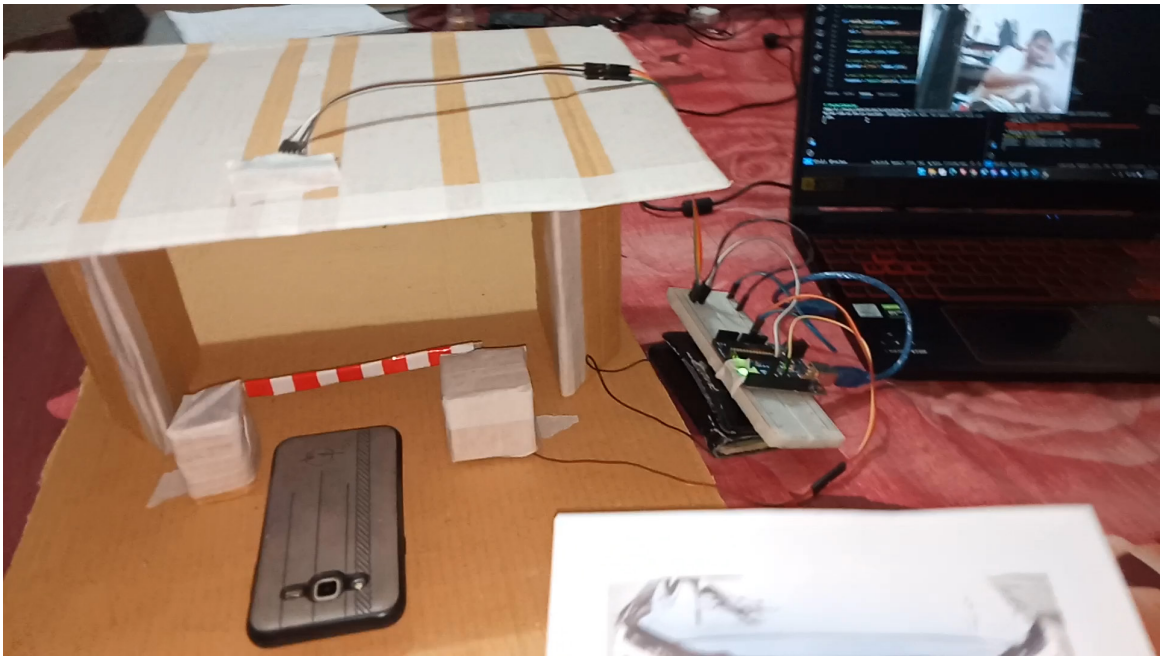


Figure 37: Scaled Setup of the project

In The setup we can see a replica of toll booth with its Arduino microcontorller. Next to that there is a laptop. In that laptop A **flask server** and a **OpenCV python** Program is running to detect and verify the license plate. We've printed some car pictures with some **valid and invalid** license plate numbers to check whether our system works as intended or not.

4.2.2 Valid License Plate:

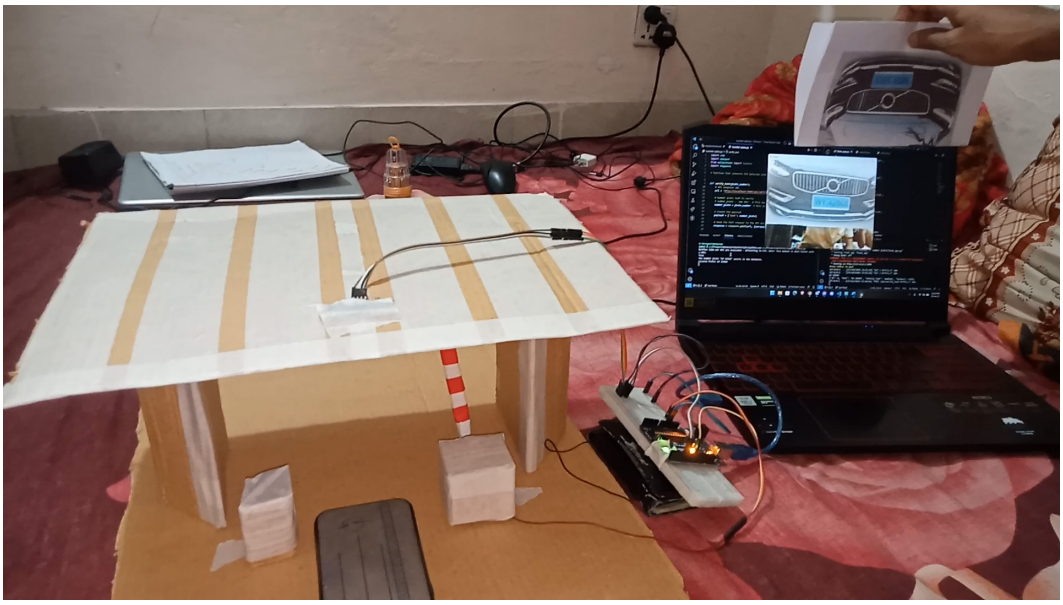


Figure 38: Project States when Given Valid License Plate Picture

In the picture we can see a printed verified car picture is hold in front of the camera and the camera detected the text and based on that it charged the user and opened the gate on the left side.

4.2.3 Invalid License Plate:

Invalid license plate could be due to many reasons. One is without balance another could be unregistered vehicle or same vehicle twice. These cases means the license plate number is invalid and the toll booth Traffic Arm should not open. We can see that in the picture given below,

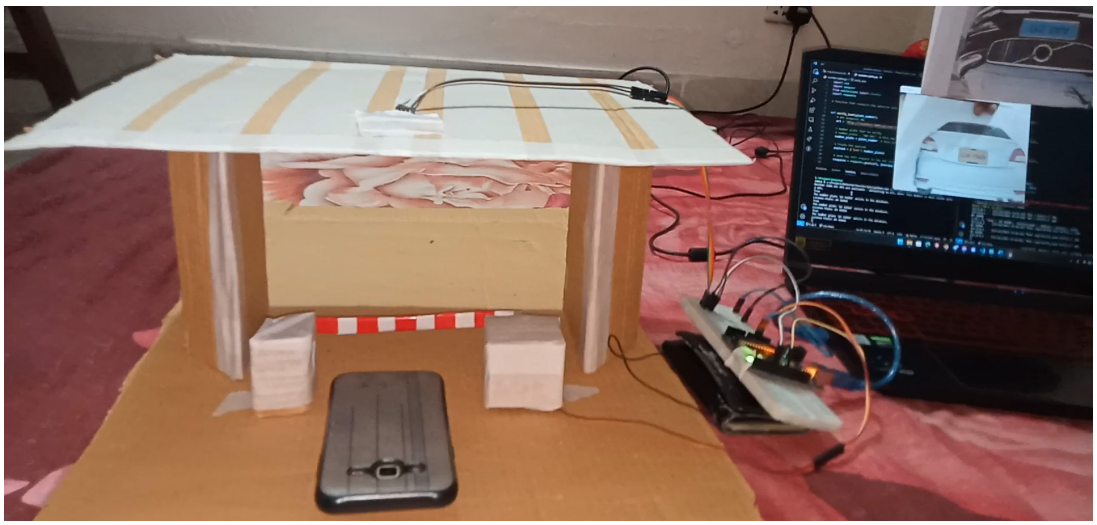


Figure 39: Project States when Given Invalid License Plate Picture

4.2.4 Web Interface:

The Web interface that shows Passed vehicle from Flask server its picture is given below

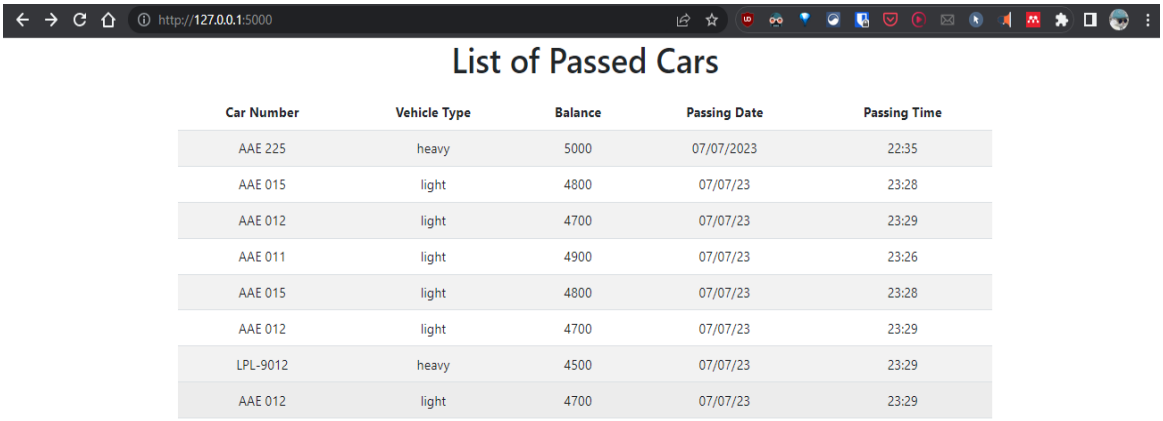


Figure 40: A screenshot of the web interface

\

Chapter 5

LIMITATIONS AND FURTHER DEVELOPMENT SCOPE

5.1 LIMITATIONS

This is just a conceptual model and by no means it is finished. This project has a lot of limitations to address those are,

5.1.1 License Plate Variations:

License plates can vary significantly in terms of size, color, font, and layout across different regions and countries. The system may face challenges in accurately detecting and recognizing license plates that deviate from the trained patterns, leading to potential errors or misclassifications. This project is only implemented on a single line license plate so putting a double line license plate may cause space related error sometimes.

5.1.2 License Plate Language:

OCR library has pretrained models for each language separately and to use Bangla we only can use Bangla and to use English we only can use English. We've Implemented English Because of our camera resolution limitation so this current project can't work with bangla language. However, with a better camera it is really a simple job to change from English to Bangla.

5.1.3 System Cost and Maintenance:

Implementing a web interface-based automated toll collection system using computer vision requires substantial investment in hardware, software, and infrastructure. Additionally, ongoing maintenance and periodic updates may be necessary to ensure the system's optimal performance, which can incur additional costs.

5.1.4 Integration Complexity:

Integrating the automated toll collection system with existing transportation infrastructure, toll management systems, and other relevant systems can be complex. Ensuring seamless data exchange, compatibility, and interoperability may require substantial effort and coordination with different stakeholders.

5.1.5 Internet and Power Reliability:

Internet and power reliability pose significant limitations in our project, "Web Interface Based Automation of Toll Collection System Using Computer Vision." As the system heavily relies on real-time data exchange and communication between the computer vision

detector, web server, and hardware module, any disruptions in internet connectivity could impact the smooth functioning of the toll collection process. In areas with unstable or intermittent internet connectivity, there is a risk of data loss or delays in transmitting license plate information from the computer vision detector to the web server. This could result in inaccuracies in vehicle verification and gate control, leading to potential toll collection errors or unauthorized access. Similarly, power reliability is crucial for the continuous operation of the system. In the event of power outages or fluctuations, the hardware module, which controls the gate opening and closing, might fail to function correctly. This could lead to gate malfunctions, further exacerbating traffic congestion at the toll plaza.

5.2 SCOPES

In this section, we outline the scopes for enhancing the project, "Web Interface Based Automation of Toll Collection System Using Computer Vision." These scopes aim to refine the accuracy, security, user experience, and hardware efficiency of the toll collection system. There are many things we wanted to do but we couldn't due to time constrains and also budget constrains. By integrating those scopes this project could advance into a new height, we seek to elevate the system's performance and create a more efficient and reliable toll collection solution. Some Of the scopes are,

5.2.1 Using a YOLO Architecture and Model:

As part of our project enhancement, we propose the integration of the YOLO (You Only Look Once) model for license plate recognition. YOLO is an advanced object detection algorithm that offers real-time and highly accurate object localization in images and videos. By incorporating YOLO into the computer vision detector module of our toll collection system, we can achieve faster and more precise license plate identification, even in complex traffic scenarios. The YOLO model's unique architecture allows it to detect multiple objects in a single pass, reducing computational overhead and enabling real-time performance. With its ability to handle various scales and aspect ratios, YOLO excels at detecting license plates of different sizes and orientations, enhancing the system's adaptability to diverse vehicle types. Implementing the YOLO model will necessitate the acquisition of an extensive dataset of license plate images for training the network. We will fine-tune the YOLO model on this dataset, optimizing its performance specifically for license plate

recognition. By leveraging the power of deep learning and the YOLO algorithm, we aspire to significantly elevate the accuracy and efficiency of our toll collection system, ultimately providing a seamless and reliable experience for toll plaza management.

5.2.2 Using a Machine Learning Model to Improve Accuracy:

Implementing a machine learning model involves collecting a diverse and extensive dataset of license plate images. The selected deep learning algorithms, such as Convolutional Neural Networks (CNNs), will require training on this dataset to learn to recognize patterns in the images. The trained model can then be integrated into the computer vision detector module to enhance license plate recognition accuracy. Evaluating the model's performance through testing and fine-tuning the hyperparameters are essential steps in ensuring the system's optimal accuracy.

5.2.3 Better Encryption in Communication:

To achieve better encryption, upgrade the communication protocol to support TLS, which ensures secure data transmission through encryption and authentication. Configuring the web server with SSL certificates and enabling HTTPS will establish a secure connection between the clients and the server. Properly managing encryption keys and ensuring regular updates to the encryption mechanisms are vital to maintaining data security.

5.2.4 Better Interactive Web Interface:

To create a more interactive web interface, implement JavaScript frameworks like React or Vue.js to enable real-time updates and dynamic data visualization. Integrate user authentication using technologies like JSON Web Tokens (JWT) and Flask-Security to manage user access and permissions. Design an intuitive user interface with responsive design principles to ensure seamless usability across various devices.

5.2.5 Microcontroller-less Traffic Arm Control:

Research and identify suitable smart actuators with built-in controllers that can be directly controlled by the web server. Determine the compatibility and communication protocols required for seamless integration. Modify the hardware module to communicate with these advanced actuators, allowing direct control of the traffic arm without an intermediary microcontroller. Conduct extensive testing and validation to ensure the reliability and stability of this new hardware setup.

Chapter 6

Discussion and Conclusion

6.1 DISCUSSIONS

In the project "Web Interface Based Automation of Toll Collection System Using Computer Vision," we have successfully developed a comprehensive toll collection solution that leverages computer vision technology, a web server, and an Arduino Uno-based hardware module. As we delve into the discussion of our findings, we will analyze the outcomes, address the limitations, and explore the potential implications of our work.

6.1.1 Interpretation of Results:

Our computer vision detector module, integrated with the Haar Cascade model and EasyOCR library, exhibited promising results in license plate recognition. By capturing real-time video from a camera source, the system accurately extracted license plate text, allowing for seamless verification and toll collection. The use of EasyOCR proved effective in handling variations in license plate formats and achieving satisfactory accuracy. Our project does the whole process of verifying car, subtracting toll and showing it into user interfac without any manual intervention. So our primary objective of automating the process is achieved through this project.

6.1.2 Comparison with Previous Studies:

In comparison with traditional manual toll collection systems, our automated solution demonstrated significant advantages in terms of speed, efficiency, and reduction of human errors. While various computer vision-based toll collection systems exist, the integration of a web server and hardware module provides a centralized and scalable approach, enabling real-time data processing and gate control.

6.1.3 Limitations and Assumptions:

Despite the success of our project, we acknowledge some limitations. The system's performance may be impacted by challenging lighting conditions or obscured license plates due to factors like vehicle speed and angle. Additionally, reliance on internet connectivity for communication between modules could pose challenges in areas with unstable network coverage. Moreover, the hardware module's dependence on the Arduino Uno introduces constraints on the number of simultaneous operations.

6.1.4 Implications and Applications:

The project's successful implementation has significant implications for toll plaza management. The automated toll collection system can alleviate traffic congestion and reduce operational costs by eliminating the need for manual toll booths and personnel. Moreover, the seamless integration of computer vision technology with a web server facilitates real-time monitoring, data storage, and traffic analysis for toll authorities.

6.1.5 Future Directions:

To enhance the system further, we propose several future directions. Implementing advanced machine learning models, such as YOLO (You Only Look Once), could potentially boost license plate recognition accuracy and speed. Strengthening communication encryption protocols will bolster data security and user trust in the system. Additionally, exploring microcontroller-less traffic arm control solutions could streamline the hardware setup, making it more compact and efficient.

6.1.6 Project Objectives Achievement:

The project **successfully achieved its objectives** of automating toll collection using computer vision and web-based technologies. We have demonstrated the feasibility of a smart toll collection system that streamlines the process, improves accuracy, and enhances user experience. The "Web Interface Based Automation of Toll Collection System Using Computer Vision" project marks a remarkable advancement in revolutionizing toll plaza management. By combining cutting-edge technologies and innovative approaches, we have successfully developed an efficient, reliable, and user-friendly toll collection solution. Our computer vision detector module, equipped with the Haar Cascade model and EasyOCR library, demonstrated remarkable accuracy in recognizing license plate information. This breakthrough technology enables swift verification and toll collection, streamlining traffic flow and reducing human errors.

6.2 CONCLUSION

In comparison to traditional manual toll collection systems, our automated solution offers a host of advantages. It operates with enhanced speed, efficiency, and cost-effectiveness, eliminating the need for manual toll booths and personnel. Moreover, the integration of a web server empowers real-time monitoring, data storage, and traffic analysis for toll

authorities, improving overall toll plaza management. As we celebrate these achievements, it is essential to acknowledge the limitations we encountered during the project. The system's performance may be affected by challenging lighting conditions or obscured license plates, factors typical in real-world scenarios. Furthermore, internet connectivity dependency for communication could pose challenges in areas with unstable network coverage. Looking ahead, we envision exciting possibilities for enhancing the system further. Integrating advanced machine learning models, such as YOLO, could boost license plate recognition accuracy and speed, taking our project to new heights. Strengthening communication encryption protocols will ensure data security, building trust among users and stakeholders. Our project's success in automating toll collection through computer vision and web-based technologies demonstrates the potential to transform toll plaza management worldwide. This smart toll collection system's efficiency, accuracy, and user-friendliness signify a significant leap forward in the pursuit of smarter, more convenient, and sustainable transportation infrastructure. As we pave the way for the future, we remain committed to refining our solution and driving innovation in the toll collection domain.

References

LIST OF REFERENCES

[1] "Traffic congestions at Bangabandhu Expressway and Padma Bridge toll plazas | The Business Standard." https://www.tbsnews.net/bangladesh/traffic-congestions-bangabandhu-expressway-and-padma-bridge-toll-plazas-455242 (accessed Jul. 18, 2023).

[2] C. H. Lai, P. K. Hsiao, Y. T. Yang, S. M. Lin, and S. C. Candice Lung, "Effects of the manual and electronic toll collection systems on the particulate pollutant levels on highways in Taiwan," *Atmos Pollut Res*, vol. 12, no. 3, pp. 25–32, Mar. 2021, doi: 10.1016/J.APR.2021.01.020.

[3] "Padma Bridge automated toll collection ready for launch after Eid | The Business Standard." https://www.tbsnews.net/bangladesh/padma-bridge-automated-toll-collection-ready-launch-after-eid-655890 (accessed Jul. 18, 2023).

[4] M. S. Mathews, M. Prabu, A. Pitchai, D. Ben Roberts, and G. Rahul, "Improved Computer Vision-based Framework for Electronic Toll Collection," in *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE, Jan. 2022, pp. 213–216. doi: 10.1109/Confluence52989.2022.9734219.

[5] S. I. Popoola, O. A. Popoola, A. I. Oluwaranti, A. A. Atayero, J. A. Badejo, and S. Misra, "A cloud-based intelligent toll collection system for smart cities," *Communications in Computer and Information Science*, vol. 827, pp. 653–663, 2018, doi: 10.1007/978-981-10-8657-1_50/COVER.

[6] A. Suryatali and V. B. Dharmadhikari, "Computer vision based vehicle detection for toll collection system using embedded Linux," in *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, IEEE, Mar. 2015, pp. 1–7. doi: 10.1109/ICCPCT.2015.7159412.

[7] "What is Python? Executive Summary | Python.org." https://www.python.org/doc/essays/blurb/ (accessed Jul. 18, 2023).

[8] "OpenCV Course — Lesson 1: Introduction to OpenCV | by Machine Learning in Plain English | Medium." https://medium.com/@nerdjock/opencv-course-lesson-1-introduction-to-opencv-8ae8d967ce8 (accessed Jul. 18, 2023).

[9] "Haar Cascade Classifiers. Learn and implement Haar cascade… | by Om Rastogi | DataDrivenInvestor." https://medium.datadriveninvestor.com/haar-cascade-classifiers-237c9193746b (accessed Jul. 18, 2023).

[10] "Introduction to EASYOCR: A Simple and Accurate Python Library for Optical Character Recognition | by Arjun Gullbadhar | Level Up Coding." https://levelup.gitconnected.com/introduction-to-easyocr-a-simple-and-accurate-python-library-for-optical-character-recognition-62d98d94bcfc (accessed Jul. 18, 2023).

[11] "Python's Requests Library (Guide) – Real Python." https://realpython.com/python-requests/ (accessed Jul. 18, 2023).

[12] "pySerial — pySerial 3.4 documentation." https://pyserial.readthedocs.io/en/latest/pyserial.html (accessed Jul. 18, 2023).

[13] "Flask · PyPI." https://pypi.org/project/Flask/ (accessed Jul. 18, 2023).

[14] "HTML: HyperText Markup Language | MDN." https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Jul. 18, 2023).

[15] "Bootstrap · The most popular HTML, CSS, and JS library in the world." https://getbootstrap.com/ (accessed Jul. 18, 2023).

[16] "MySQL." https://www.mysql.com/ (accessed Jul. 18, 2023).

[17] "History of the iconic Arduino UNO microcontroller - Geeky Gadgets." https://www.geeky-gadgets.com/arduino-uno-microcontroller-13-12-2021/ (accessed Jul. 18, 2023).

[18] "HC-SR04 Ultrasonic Sensor : Pin Configuration, Working and Applications." https://www.elprocus.com/hc-sr04-ultrasonic-sensor-working-and-its-applications/ (accessed Jul. 18, 2023).

[19] "Introduction to Servo Motors." https://www.sciencebuddies.org/science-fair-projects/references/introduction-to-servo-motors (accessed Jul. 18, 2023).

[20] "Radio-frequency Identification (RFID) Technology - NextIAS." https://www.nextias.com/current-affairs/04-06-2022/radio-frequency-identification-rfid-technology (accessed Jul. 18, 2023).

[21] "What is Quick Response Code (QR Code)? - Definition from Techopedia." https://www.techopedia.com/definition/2915/quick-response-code-qr-code (accessed Jul. 18, 2023).

[22] "What Is Node.js and Why You Should Use It." https://kinsta.com/knowledgebase/what-is-node-js/ (accessed Jul. 19, 2023).

[23] "PHP: What is PHP? - Manual." https://www.php.net/manual/en/intro-whatis.php (accessed Jul. 19, 2023).

[24] "(3) What is Django? A high level python web framework. | LinkedIn." https://www.linkedin.com/pulse/what-django-high-level-python-web-framework-ocman-nazir-briet/ (accessed Jul. 19, 2023).

[25] "What is a PLC? Programmable Logic Controller." https://inductiveautomation.com/resources/article/what-is-a-PLC (accessed Jul. 19, 2023).

[26] "What is ESP32 and Why Is It Best for IoT Projects? - IoT Tech Trends." https://www.iottechtrends.com/what-is-esp32/ (accessed Jul. 19, 2023).

[27] "Infrared Sensor - How it Works, Types, Applications, Advantage & Disadvantage." https://electricalfundablog.com/infrared-sensor/ (accessed Jul. 19, 2023).

Appendices

APPENDICES

GITHUB REPOSITORY FOR PROJECT CODE

We are delighted to share the GitHub repository containing the source code and additional resources for the project "Web Interface Based Automation of Toll Collection System Using Computer Vision." This repository hosts the model, code, and any other necessary files related to the project, providing a comprehensive view of our work.

GitHub Repository URL:

```
https://github.com/ImranKhanPrince/Web-Interface-Based-Automation-of-Toll-Collection-System-Using-Computer-Vision
```

By visiting the above link, you will have access to all the files and resources that constitute the toll collection system, including the computer vision model, Python scripts, web server code, and other materials. We encourage you to explore the repository to gain a deeper understanding of the project's implementation and to utilize any components that may be relevant to your own endeavors. Should you have any questions or require further assistance, please feel free to reach out to us. We hope that this GitHub repository will foster collaboration, inspire new ideas, and contribute to the advancement of toll collection systems using computer vision and web-based technologies.

Thank you for your interest in our project, and happy exploring!