Toll gate systems with YOLO-CNN based using Raspberry Pi for vehicle class detection

Achmad Musyaddad Romadoni[1], *Pandu* Agusti Putra[1], *Delia* Riza Mayora[1], *Muhammad* Daffa Jawadil Ubaid[2], *Nur* Vidia Laksmi B.[1*], *Reza* Rahmadian[1], *Ayusta* Lukita Wardani[1], *Aditya* Chandra Hermawan[1], *Alfarid* Hendro Yuwono[1]

[1]Department of Electrical Engineering, Faculty of Vocational Studies, Universitas Negeri Surabaya, Surabaya, Indonesia
[2]Department of Informatics Management, Faculty of Vocational Studies, Universitas Negeri Surabaya, Surabaya, Indonesia

Abstract. Traffic congestion at toll gates remains a significant issue, primarily due to manual vehicle identification and payment processes. This paper aims to design a prototype of an automated toll gate system integrated with IoT that can automatically detect vehicle classes with the You Only Look Once (YOLO)-Convolutional Neural Networks (CNN) based, and supports digital payment methods via RFID and QRIS. The system is built using a Raspberry Pi 3B+ as the controller, equipped with an RFID-RC522 module and an MG90S servo motor as the gate actuator. A 604 dataset of vehicle images was used to train the YOLOv3 model, achieving a mean Average Precision (mAP) of 75.41%. Experimental results show that the system can read RFID data reliably and receive QRIS payment callbacks in real time through integration with Flask and Ngrok web applications. The implementation demonstrates that all components work in an integrated and responsive performances for vehicle detection, identity verification, and gate control. This system presents strong potential as an efficient solution to reduce queues and improve traffic flow at toll gates.

1 Introduction

Freeway road infrastructure is a vital component in supporting social and economic mobility, as it facilitates the rapid and efficient distribution of goods and services [1]. However, in several major cities in Indonesia, traffic congestion frequently occurs at toll gate areas, both at entry and exit points [2]. This congestion leads to negative impacts such as fuel wastage, increased pollution, and logistical delays [3]. From an economic perspective, losses are calculated based on vehicle operating costs, travel time, and emissions [4].

Currently, the automatic toll gate (GTO) system in Indonesia is divided into two types: Single-class GTO and Multi-class GTO. However, in multi-class GTO systems, vehicle classification is still performed manually by operators in the control room [5]. This manual process is one of the main causes of long queues at toll gates. In addition, insufficient e-toll balance often causes delays, as occurred at the Sei Rampah toll gate in October 2018 [6].

Various previous researches have attempted to address many issues using artificial intelligence and machine learning approach to optimize their systems [7-9]. A paper by Zhang et. al [10] demonstrated that the YOLO algorithm is capable of detecting vehicles in real time. Research by Darmadi et al. [11] also confirmed that YOLO is effective in identifying vehicles (cars, buses, and trucks) on the Jakarta–Cikampek toll road. In addition, a study by Taufiq [12] conducted a simulation of vehicle class detection at toll gates using the YOLO algorithm, showing that it can effectively classify vehicles based on the number of axles. However, most of these studies remain limited to algorithm testing and have not yet been integrated into a fully functional prototype system that can be practically implemented at toll gates.

Based on the aforementioned issues [1–12], this paper aims to design a solution that can reduce congestion at freeway gate entry and exit points. The proposed solution is the development of a prototype system capable of automatically detecting vehicle classes using the YOLO (You Only Look Once) algorithm-Convolutional Neural Networks (CNN) based. Additionally, the system provides an alternative payment method via QRIS when the user's e-toll balance is insufficient. The system is developed using a Raspberry Pi as the main processing unit to operate the toll gate system. The concept of the Internet of Things (IoT) [13] is also added in these systems, including the RFID-RC522 module, MG90S servo motor, and interconnected communication that enables automatic data exchange without direct human intervention.

2 System overview

This paper employs a systems engineering approach to design and test a prototype of an automated toll gate system based on IoT, with vehicle classification using the YOLO-CNN based. The development process

* Corresponding author: nurvidialaksmi@unesa.ac.id

includes the selection and integration of hardware components such as a Raspberry Pi as the central controller, an RFID module for user identification, and a servo motor as the gate barrier actuator. Software implementation involves real-time image processing based on YOLO, along with a digital payment system using QRIS. All components are designed and tested to operate in an integrated, real-time, aiming to accurately detect vehicle classes and support efficient transaction processes at toll gates.

2.1 Data collection

The sample used in this system consists of 604 images of vehicles from various classes, which have been annotated using the LabelImg application. Annotation was performed by assigning bounding boxes to each vehicle according to its class, enabling the dataset to be used for training a YOLO-based object detection model. The vehicle class dataset was obtained from an open-source repository provided by Roboflow.

The annotated image data was used for two main stages: training and testing the detection model. The dataset was split with a composition of 80% (483 images) for training and 20% (121 images) for testing. The purpose of this split is to ensure that the model can learn optimally from the training data and then be objectively evaluated using previously unseen testing data, in order to obtain accurate and comprehensive detection performance measurements.

2.2 Hardware components

2.3.1 Raspberry Pi 3B+

Raspberry Pi 3B+ is a single-board minicomputer designed to provide an affordable, energy-efficient, and capable computing solution. This device is suitable for a wide range of applications such as programming, automation, the Internet of Things (IoT), and system prototyping. Raspberry Pi is chosen for its compact form factor—about the size of a credit card—while still offering functionality and features comparable to conventional computers [14].

One of the programming languages that can be used on the Raspberry Pi is Python. Python is a high-level programming language known for its clear and simple syntax. Additionally, Python is supported by a wide range of libraries that offer extensive functionality, enabling the development of various types of applications.

In this system, the Raspberry Pi serves as the main processing unit that controls and executes all automation processes of the toll gate system. Its module specifications include: a Broadcom BCM2837B Cortex-A53 64-bit processor with a clock speed of 1.4GHz, 1GB of LPDDR2 SDRAM, and connectivity support such as dual-band Wi-Fi (2.4GHz and 5GHz), Bluetooth 4.2, Ethernet, and a 40-pin General-Purpose Input/Output (GPIO) header. The block diagram of the Raspberry Pi 3B+ is shown in Fig. 1.

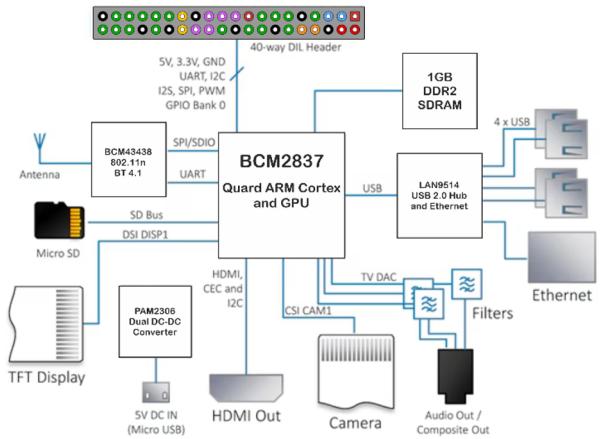

Fig 1. Block diagram of raspberry Pi 3B+.

2.3.2 RFID-RC522

RFID (Radio Frequency Identification) is a technology that utilizes radio waves to automatically identify data [15]. It is a module used to read and write RFID tags [16]. The RFID module contains a microchip equipped with an antenna, allowing it to identify and transmit data. This antenna enables the chip to send identification information to a reader.

In this system, RFID technology is utilized as part of the e-toll system, where an RFID tag embedded in the vehicle is used for automatic identification when passing through the toll gate. The module specifications are as follows: operating voltage of 2.5VDC–3.3VDC, data transfer rate of 10Mbps, current consumption of 13mA–26mA, frequency of 13.56 MHz, and communication protocols such as I2C, SPI, and UART.

2.3.3 Motor servo MG90S

The MG90S servo motor is a device used to convert electrical energy into mechanical energy. In a DC servo motor, this energy transformation occurs through the interaction between two magnetic fields, one of which is generated by a permanent magnet. The servo motor has the capability to move or rotate objects with high precision, particularly in terms of angular position, acceleration, and speed [17].

In this system, the MG90S servo motor functions as an actuator to automatically open and close the toll gate barrier once the payment has been completed. The module specifications are as follows: operating voltage of 4.8VDC–6VDC, maximum rotation speed of 0.10 sec/60°, rotation angle up to 180 degrees, PWM range of 0.5 ms to 2.5 ms, and a maximum torque of 2.5 kg-cm at 6VDC.

Pulse Width Modulation (PWM) is a technique used to generate an analog signal using digital devices by adjusting the pulse width within a fixed period. The ratio between the active signal duration and the total period is called the duty cycle, which influences the average output value. PWM can modulate the pulse width to provide analog control over digital outputs, such as controlling a DC motor. In this system, PWM is used to control the rotation angle of a servo motor, where the pulse width determines the movement angle, such as 0°,

45°, 90°, 135°, and 180°, and is calculated using a specific formula. [18]:

$$\text{PWM} = \text{PWMmin} + \left(\frac{Servo\ Angle}{\text{Max}\,Angle} \times (PWM_{max} - PWM_{min})\right) \quad (1)$$

2.3 Software components / logic system

2.3.1 YOLO

YOLO is an algorithm designed to detect and recognize various objects in images in real-time [19]. YOLO is a part of Artificial Intelligence (AI) technology that utilizes CNN to perform real-time object detection [20]. The network works by dividing the input image into several regions or grids, then predicting bounding boxes along with the probability of object presence within each grid. In this system, YOLO is used to detect vehicles and automatically classify them into their respective categories.

Here, a confusion matrix is used to evaluate the classification performance of the model. The confusion matrix allows for the calculation of precision and recall for each class. In classification tasks, based on the combination of actual (ground truth) and predicted classes, there are four possible outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

Based on the results of the confusion matrix, precision and recall can be calculated using the following equations: [21].

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Precision indicates how accurately the model classifies relevant objects, while recall measures the model's ability to identify all relevant objects from the ground truth. From these two metrics, a Precision–Recall (P–R) curve can be constructed, where recall is plotted on the horizontal axis and precision on the vertical axis.

Mean average precision (mAP) is the average of the Average Precision (AP) values, which are calculated based on the precision scores of each correctly detected relevant item. If a relevant item is not detected, its precision is considered to be zero. The formula for calculating mAP is as follows: [21].

$$\text{mAP} = \frac{1}{c}\sum_{i}^{c} = 1\text{AP}i \quad (4)$$

The equation for calculating Average Precision (AP) using the 11-point interpolation method is as follows:

$$\text{AP} = \frac{1}{11}\sum_{r\in\{0;0.1;\ldots;1\}} \text{p}_{interp}\ (\text{r}) \quad (5)$$

$$p_{int\,erp} = \max_{r:\tilde{r}\geq\tilde{r}_n} p\,(\tilde{r}) \quad (6)$$

The equation for calculating average precision (AP) using the all-point interpolation method is given as follows:

$$AP = \sum_{n=0}(r_{n+1} - r_n)\ p_{interp}(r_{n+1}) \quad (7)$$

$$p_{interp}(r_{n+1}) = \max_{r:\tilde{r}\geq+\tilde{r}_n 1} p\,(\tilde{r}) \quad (8)$$

Where:

c : Number of classes
R : Relevant items retrieved by the system
p : Precision
p_{interp} : Interpolated precision
r : Recall
$p(\tilde{r})$: Precision value at a given recall.

The block diagram of the object detection system based on YOLO (You Only Look Once) is shown in Figure 2. This figure illustrates the main pipeline starting from the input image of a vehicle, which is then processed by a Convolutional Neural Network (CNN) using the Darknet-53 architecture for feature extraction. The extracted features are then passed to the YOLO detection head, which simultaneously predicts the bounding box coordinates, confidence scores, and object class labels. These predictions are then passed through a post-processing stage, including Non-Maximum Suppression (NMS) to eliminate duplicate detections and threshold filtering to remove detections below a defined confidence score. The final output consists of the detected object positions (bounding boxes), confidence levels, and class labels.

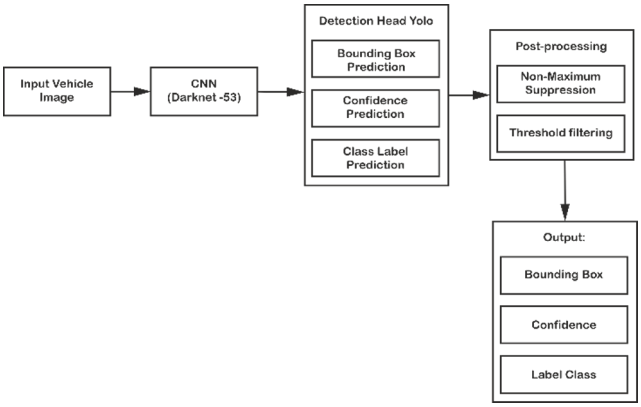

Fig. 2. YOLO block diagram.

2.3.2 QRIS

Quick Response Code Indonesian Standard (QRIS) is an electronic payment tool designed to facilitate transactions for the public [22]. By using QRIS, people can enjoy the convenience of making payments anytime and anywhere, even outside regular banking hours. On the customer side, transactions become more practical as they no longer need to carry a wallet. With just a

smartphone and an internet connection, payments can be made easily and quickly.

There are many payment gateways used in Indonesia, one of which is Midtrans. Midtrans is very easy to integrate into websites or mobile applications. It provides APIs for more flexible integration with any platform. Midtrans is equipped with transaction and data security systems to ensure the safety of all transactions [23].

In this system, QRIS is used as an alternative payment method at the automated toll gate, particularly for users whose e-toll balance is insufficient. With the availability of QRIS as an option, the system becomes more flexible and supports the broader implementation of cashless payments across various segments of society.

2.3.3 Freeway gate system interface

Fig. 3 shows the display of the toll gate system. In this paper, it was developed using Flask, a micro web application framework written in the Python programming language [24]. Flask is known for being lightweight and flexible, allowing for rapid development of web applications with a minimalist and modular design. It supports numerous third-party extensions that can add functionality according to system requirements. With this approach, Flask is used to provide a web interface that displays vehicle detection results, QRIS payment status, and gate control operations.

.

Fig 3. System interface.

To allow the locally hosted Flask website to be accessed by third-party services such as Midtrans, Ngrok is used as a tunneling tool. Ngrok converts the local IP address into a public URL that can be accessed over the internet, thereby enabling Midtrans to send payment response data to the local server via the HTTP POST method [19]. In addition, Ngrok simplifies the process of system demonstration and testing without the need for a permanent hosting server. The use of Ngrok is an effective solution for integrating local systems with cloud-based services in real-time.

2.4 Freeway gate system block diagram

Fig. 4 illustrates the block diagram of the tool, depicting the prototype system of a freeway gate based on the Raspberry Pi 3B+. The system begins with the input process in the form of vehicle data (Classes 1–5), detected using the YOLO algorithm with AI technology. The RFID card serves as the primary payment method, providing tariff data and gate origin information. If the balance on the RFID card is insufficient, the system will automatically switch to the QRIS (Midtrans) method as an alternative payment option. All inputs are processed by the Raspberry Pi 3B+, which acts as the central control unit of the system. Vehicle detection is used to determine the class of the vehicle, while information from the RFID or QRIS is used for verification and toll payment.

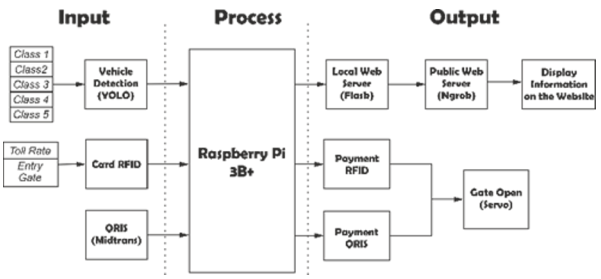

Fig 4. Block diagram of the toll gate systems.

After the data is processed, the Raspberry Pi displays vehicle information and payment status on a local web server using Flask, which is then forwarded to a public web server via Ngrok for online access and display on a website. If the payment is successful—whether via RFID or QRIS—the Raspberry Pi sends a signal to the servo motor to automatically open the gate. Thus, this system integrates vehicle detection, digital payment, and gate automation into a unified Internet of Things (IoT)-based solution.

Fig. 5 illustrates the overall system flowchart diagram, starting from the vehicle approaching the toll gate, image capture, vehicle classification using the YOLO method, payment verification via e-toll (RFID) or QRIS, and automatic gate opening after successful payment verification. The system is also equipped with a balance-checking mechanism for the e-toll card; if the balance is insufficient, the user is directed to complete the payment through the QRIS method displayed on the system screen. Once the transaction is confirmed successful, the gate will open.

The system process begins when a vehicle approaches the toll gate. The system then captures an image or video of the vehicle entering the detection area. Next, the system performs vehicle classification using the YOLO method to determine the vehicle type and the corresponding toll fare. After that, the system processes the payment automatically using an e-toll card (RFID) and displays information such as vehicle type, entry gate, and toll fare to the user. The system checks whether the e-toll balance is sufficient for payment. If the balance is sufficient, the process continues by automatically opening the gate. However, if the balance is insufficient, the system directs the user to make a payment through an alternative method using QRIS. The user then scans the displayed QRIS code and completes the payment. Once the payment via either e-toll or QRIS is successfully verified, the gate will open automatically. Finally, the vehicle passes through the gate, and the system process is completed.

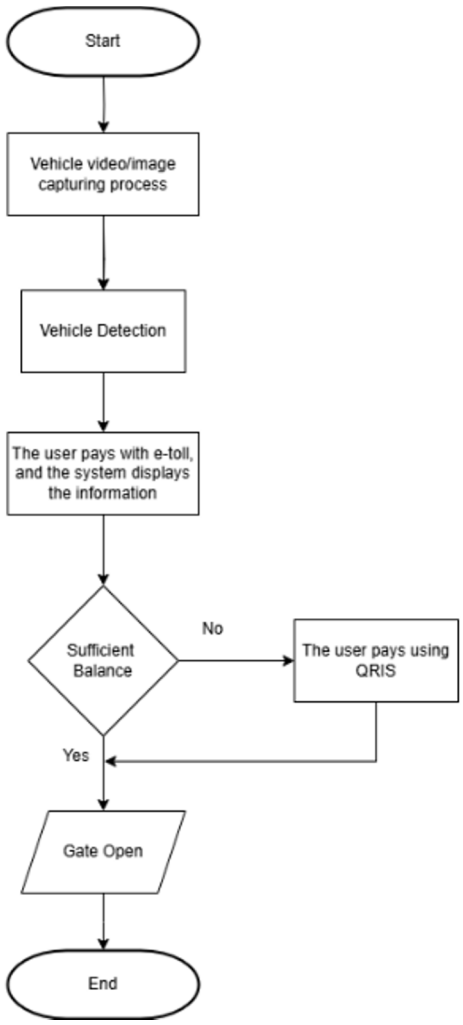

Fig. 5. Flowchart system.

3 Results and discussion

3.1 Vehicle object detection testing

Vehicle object detection testing was conducted using the mean Average Precision (mAP) metric at an Intersection over Union (IoU) threshold of 0.5 (mAP@0.50). Based on the evaluation results of the trained YOLO detection model, a mAP score of 75.41% was obtained. This result indicates that the model has a fairly good capability in recognizing and classifying vehicles into their respective categories based on the input images.

The model's performance was further analyzed based on five vehicle categories. The average precision (AP) value for each vehicle class is presented in Table 1 below.

Tabel 1. Average precision (AP) per vehicle category.

Vehicle Class	Precision (%)	Recall (%)	AP (%)
Class I	85.61	89.68	92,72
Class II	40.63	68.42	67,65
Class II	40.00	50.00	50,02
Class IV	62.50	83.33	81,44
Class V	76.92	83.33	85,20
mAP			75,41

The test results show that the model performs very well in recognizing Class I vehicles, achieving an Average Precision (AP) of 92.72%. It also performs fairly well on Class IV and V vehicles, with AP values of 81.44% and 85.20% respectively. For Class II, the AP value reached 67.65%, indicating that the model is still reasonably capable of distinguishing vehicles in this class, although there are some visual similarities with other classes. Meanwhile, the AP for Class III was 50.02%, which is lower compared to the other categories.

3.2 Servo motor testing

The testing was conducted to ensure that the MG90S servo motor could accurately respond to control signals (PWM) from the Raspberry Pi 3B+. The PWM signal was sent through GPIO 18 (pin 12) to control the rotation angle of the motor according to the system commands. The test results are presented in Table 2.

Tabel 2. Servo motor test results.

System Condition	Payment Status	PWM (µs)	Servo Rotation Angle (°)	Description
Vehicle detected	Unpaid	500	0°	Gate closed
Payment successful	Paid	1500	90°	Gate opened automatically
5 seconds after gate opened	Paid	500	0°	Gate closes automatically

Testing was carried out under two simulated conditions. First, when the payment status is confirmed as successful, the system sends a command to open the gate (rotation of approximately ±90°). Then, after a few moments, the servo receives a command to close the gate barrier again. The test results showed that the servo motor was able to operate precisely with stable rotation. No issues such as excessive vibrations or position inaccuracies were observed. This indicates that the communication between the Raspberry Pi and the servo via PWM signals is functioning effectively and reliably.

3.3 RFID reading test

The RFID-RC522 module was tested to detect RFID cards used as vehicle identifiers. The test was carried out by bringing the RFID card close to the reader within an effective range of 1–3 cm. The module was connected to the Raspberry Pi via the SPI interface, and a Python program was used to read the card's UID (Unique Identifier). The test results are presented in Table 3.

The test results showed that the RFID module was able to read card data accurately and reliably. The successfully read UID was used by the system to match vehicle information such as toll gate origin, fee, and vehicle identity. This data was also displayed on the web interface as a transaction log. During testing, no reading failures were found within the effective range.

Tabel 3. RFID reading test results.

Tag to Reader Distance	UID Read	Read Success	Description
1	Detected	100%	Optimal Reading
2	Detected	100%	Still Readable
3	Detected	100%	Maximum Effective Range
4	Not Read	0%	Out of Detection Range

3.4 QRIS testing

The testing was conducted to evaluate the system's ability to respond to payment transactions via QRIS integrated with the Midtrans service. Since the Flask web application runs locally on the Raspberry Pi, Ngrok was used as a tunneling tool to enable the system to receive callbacks directly from Midtrans. The test results are presented in Table 4.

Tabel 4. QRIS payment test results.

Scenario	Status Callback Midtrans	Gate Status	Description
QRIS Payment Successful	Success (HTTP 200)	Open	Gate opened automatically
QRIS Payment Failed	Failed (HTTP 400)	closed	Transaction cancelled
Connection Lost	Failed to Send	closed	The web did not receive the callback

The callback status from Midtrans is indicated by the HTTP response code, which represents the result of the transaction process. An HTTP code 200 (Success) signifies that the request has been successfully received and processed by the server, which in the context of this system means that the signal to open the gate was received and executed properly. Conversely, an HTTP code 400 (Failed) indicates that the request was invalid or failed to be processed by the server, resulting in the transaction being automatically canceled and the gate remaining closed.

The test results indicate that the system was able to receive transaction responses from Midtrans. When the payment status was confirmed as successful, the system immediately triggered a command to open the gate via the servo motor. This process ran smoothly, demonstrating that communication between Midtrans and the local system functioned properly. The QRIS system has proven to be successfully integrated as an automatic payment method in this toll gate prototype.

3.5 Toll gate system circuit configuration

Fig. 6 shows the design and implementation results of an automated toll gate system based on Raspberry Pi 3B+, integrated with the MG90S Servo Motor and RFID-RC522 sensor. The system was successfully tested to detect vehicles and automatically open the gate after the identification and payment processes were successfully completed.

In the toll gate system circuit, the Raspberry Pi 3B+ functions as the main control unit, managing communication between components, processing data from YOLO-based vehicle detection and RFID-RC522 reading. The processed data is displayed on a Flask-based web interface. Meanwhile, the gate-opening instruction is sent to the MG90S Servo Motor via a PWM signal from GPIO pin 18. The freeway gate system device is shown in Figure 7.

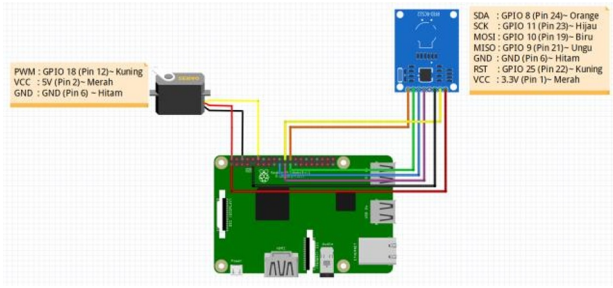

Fig 6. System hardware circuit.

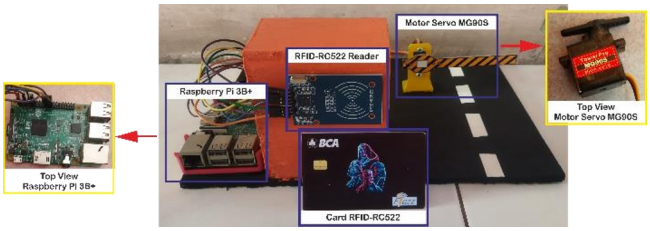

Fig 7. System device setup.

4 Conclusion

This system successfully designed and implemented a prototype of an automated toll gate system using the Raspberry Pi 3B+ as the main controller. The system is capable of classifying vehicle types using the YOLO algorithm and integrates digital payment through QRIS. It achieved a reasonably high vehicle detection accuracy of 75.41% mAP (mean average precision). Testing results indicate that all components operate in an integrated and responsive manner, including vehicle detection, payment processing, and gate control.

References

1. Z. Zhang, Z. Niu, Y. Liu, Y. Li, Adaptive analysis of freeway off-ramps incorporating heterogeneous traffic flows. Infrastructures. **10**, 4 (2025). https://doi.org/10.3390/infrastructures10040088
2. S. Syaiful, D. Maulani, D. Megawati, R. Gibran, A. K. Wardhani, Engineering model on traffic volume at bocimi toll gate. Journal of Applied Engineering

Science. **23**, 1, 99–111 (2025). https://doi.org/10.5937/jaes0-54055

3. M. Fattah, S. R. Morshed, A. A. Kafy, Insights into the socio-economic impacts of traffic congestion in the port and industrial areas of Chittagong city, Bangladesh. Transportation Engineering. **9** (2022). https://doi.org/10.1016/j.treng.2022.100122
4. X. Ye, X. Wang, S. D. Cafiso, A. Gracia, Readiness Evaluation of Freeways for Lane-Detection Performance of Lidar-Based Automated Vehicles: A Field Test Analysis. IEEE Transactions on Intelligent Transportation Systems. (2025). https://doi.org/10.1109/TITS.2025.3568591
5. U. Wiharja, S. Hartanto, Power and Waiting Time Efficiency at Automatic Toll Gates with the Contactless Card Payment Implementation. **42**, 1 (2025). https://doi.org/10.29037/2224-9028.1605
6. M. S. Vineetha et al, System for Automatic Toll Collection Using Image Processing and Raspberry PI. Lecture Notes in Network and Systems. (2025). https://doi.org/10.1007/978-3-031-88992-9_35
7. K. A. Tureta, A. Sabo, Y. Abdulrazak, Optimal Placement of Phasor Measurement Units on Shiroro 330 kV Grid Network using Binary Grey Wolf Optimization Algorithm. Vokasi Unesa Bull. Eng. Technol. Appl. Sci. **2**, 3, 444–459 (2025). https://doi.org/10.26740/vubeta.v2i3.38936
8. G. Oise, Enhancing Indoor Positioning Accuracy with Ant Colony Optimization and Dual Clustering. Vokasi Unesa Bull. Eng. Technol. Appl. Sci. **2**, 3, 516–530 (2025). https://doi.org/10.26740/vubeta.v2i3.39452
9. M. Ajuji, M. Dawaki, A. Mohammed, A. Ahmad, Estimating Residential Natural Gas Demand and Consumption: A Hybrid Ensemble Machine Learning Approach. Vokasi Unesa Bull. Eng. Technol. Appl. Sci. **2**, 3, 549–557 (2025). https://doi.org/10.26740/vubeta.v2i3.40135
10. Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, X. Guo, Real-Time Vehicle Detection Based on Improved YOLO v5. Sustainability. **14**, 19 (2022). https://doi.org/10.3390/su141912274
11. M. Manjusha, R. Niveditha, V. Sunitha, Estimation of YOLO algorithms for extracting traffic data using UAV and stationary camera. Innovative Infrastructure. **10**, 4 (2025). https://doi.org/10.1007/s41062-025-01898-y
12. P. P. Singh, S. Prasad, G. Kalita, J. Das, P. J. Mahanta, Light Weighted YOLO Vehicle Tracker for Toll Gate Using Centroid Algorithm. Lecture Notes in Electrical Engineering. (2025). https://doi.org/10.1007/978-981-97-3090-2_16
13. T. Sanislav, G. D. Mois, S. Zeadally, S. C. Folea, Energy Harvesting Techniques for Internet of Things (IoT). IEEE Access. **9**, 39530–39549 (2021). https://doi.org/10.1109/ACCESS.2021.3064066
14. S. Karthikeyan, R. A. Raj, M. V. Cruz, L. Chen, J. L. A. Vishal, V. S. Rohith., A Systematic Analysis on Raspberry Pi Prototyping: Uses, Challenges, Benefits, and Drawbacks. IEEE Internet of Things Journal. **10**, 16, 14397–14417 (2023). https://doi.org/10.1109/JIOT.2023.3262942
15. L. Profetto, M. Gherardelli, E. Iadanza, Radio Frequency Identification (RFID) in health care: where are we? A scoping review. Health Technology. **12**, 5, 879–891 (2022). https://doi.org/10.1007/s12553-022-00696-1
16. K. Mazumemdar, A. Ghosh, P. Singh, S. Peddakrishna, J. Kumar, RFID Antenna Deployment Model and Anti-Collision Algorithms to Enhance Vehicle Toll System Efficiency. Electronics (Switzerland). **14**, 7. https://doi.org/10.3390/electronics14071404
17. X. Zhang, Design and optimization of high-precision servo control system in precision positioning technology. Applied Mathematics and Nonlinear Sciences. **9**, 1 (2024). https://doi.org/10.2478/amns-2024-2958
18. S. Chandrappa, M. S. Guruprasad, H. N. N. Kumar, K. Raju, D. K. S. Kumar, An IoT-Based Automotive and Intelligent Toll Gate Using RFID. SN Computer Science. **4**, 2 (2023). https://doi.org/10.1007/s42979-022-01569-0
19. A. Kumar, N. Katal, A Lightweight YOLO Model for Detection of Disease from Optic Disc Region of Eye Fundus Imagery. Sensing and Imaging. **26**, 1 (2025). https://doi.org/10.1007/s11220-025-00575-9
20. M. L. Ali, Z. Zhang, The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. Computers. **13**, 12 (2024). https://doi.org/10.3390/computers13120336
21. X. Huang et al., SDES-YOLO: A high-precision and lightweight model for fall detection in complex environments. Scientific Reports. **15**, 1 (2025). https://doi.org/10.1038/s41598-025-86593-9
22. E. H. Muchtar et al., Quick response code Indonesia standard (QRIS) E-payment adoption: customers perspective. Cogent Business and Management. **11**, 1 (2024). https://doi.org/10.1080/23311975.2024.2316044
23. M. Hisyam, I. B. K. Manuaba., Integration Model of Multiple Payment Gateways for Online Split Payment Scenario. International Conference on Information Management and Technology, 122–126 (2022). https://doi.org/10.1109/ICIMTech55957.2022.9915168
24. A. Morchid et al., IoT-enabled fire detection for sustainable agriculture: A real-time system using flask and embedded technologies. Results in Engineering. **23** (2024). https://doi.org/10.1016/j.rineng.2024.102705