Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 6 No. 5 (2022) 749 - 760 ISSN Media Electronic: 2580-0760

Vehicle Detection Monitoring System using Internet of Things

Yani Nurhadryani[1], Wulandari[2*], Muhammad Naufal Farras Mastika[3]
[1,2,3]Department of Computer Science, Faculty of Mathematics and Natural Science, IPB University
[1]yani_nurhadryani@apps.ipb.ac.id, [2]wulandari.ilkom@apps.ipb.ac.id*, [3]muhammadnaufalfm@gmail.com

Abstract

The overcapacity of vehicle numbers is one of the significant causes of the traffic congestion problem on Indonesia roadway. The government applies a One-way system (SSA) as one proposed solution to unravel the congestion. However, several congestion points are still found during the SSA implementation. Thus, this research offers an alternative method to detect congestion using IoT technology. The system automatically enumerates the number, classifies the type, and computes the speed averages of vehicles to identify the severity of congestion based on the Indonesian Highway Capacity Manual (IHCM) published by the Ministry of Public Works 2014. We utilize ultrasonic sensors to detect the vehicles and send the data to the server in real-time. The research successfully develops an IoT system for traffic congestion detection. Communication between nodes and API can be done well. Data exchange involving encryption and decryption with AES-256 is successfully done. Website application developed in this research is successfully show the severity level of the congestion and their vehicle numbers. The average accuracy of the system is 78,97%. The system detected more vehicles than actual numbers due to the misreading value of the sensors.

Keywords: Congestion, IHCM, Internet of Things, Vehicle detection, Web monitoring

1. Introduction

Congestion is a significant problem faced by Indonesia. Currently, Indonesia occupies the second position as the most congested country globally in 2017 [1]. Based on [2], congestion is a phenomenon when the smoothness of traffic on existing roads decreases. Congestion is commonly caused by the inability of a road to accommodate passing vehicles. The excess volume of vehicles compared to the capacity of a road causes congestion on a road segment. Road growth in Indonesia from 2011 to 2016 was 6.9%. On the other hand, vehicle growth from 2011 to 2016 in Indonesia was 52.26% for cars and 51.32% for motorcycles [3], [4].

The losses caused by traffic jams are not small. Congestion can increases fuel consumption [5], [6]. That research result is in accordance with [7] that was conducted in Bogor City. The result showed that the cost of purchasing fuel for cars increased from Rp. 13.933 to Rp. 19.171 and motorcycles from Rp. 5.082 to Rp. 7.172. Apart from the economic aspect, traffic jams are also detrimental to social, cultural, and mental health aspects [8]–[10]. The data shows that 85.8% experience stress when experiencing traffic jams. In addition, 63% of students and college students admitted experiencing delays due to traffic jams [7]. Based on these problems, it is necessary to develop a system to detect congestion points in traffic. The congestion detection system is designed to avoid more severe congestion.

In the previous research, a method of a vehicle to vehicle (V2V) communication [11], [12] with the CoTEC [13] and the Vanet algorithm [14], [15] algorithm was used to detect congestion. Those methods utilize communication between vehicles to notify the smooth condition of a road segment. However, those methods take a long time to implement. This method requires each vehicle to have a communication device to convey and receive messages. Various new regulations and regulations must be rolled out first so that motor vehicle manufacturers can equip the latest vehicles with this communication tool. This method is unsuitable for application due to the severity of the congestion impact experienced. The solution needed is a system that can be implemented quickly and inexpensively.

Internet of Things (IoT) technology can be utilized to reduce the congestion problem [16]–[19]. IoT is a comprehensive and open network of intelligent objects that have the ability to organize and share information,

Accepted: 18-05-2022 | Received in revised: 30-08-2022 | Published: 31-10-2022

data automatically, and resources, react, and do something as a result of conditions and changes in the environment [20]–[22]. IoT is a form of evolution from an internet that provides a major leap for collecting, analyzing, and distributing data that can be turned into information, knowledge, and, perhaps, trust [23].

Much research was conducted to solve the congestion problem using the embedded system and IoT. [24] developed an automatic vehicle counting using the image. The system could count the vehicle real-time by utilizing Raspberry Pi. [25] utilized sound and infrare (IR) sensors to reduce the traffic density. This study aims to develop an Internet of Things (IoT) based system to detect congestion. We intend to implement a low-cost model for the users to monitor the traffic density level. The system will automatically record the number, classify the type, and calculate the average speed of passing vehicles. We utilize ultrasonic sensors to detect vehicles. This research is expected to help reduce congestion by using an IoT-based system that can detect the level of congestion on the road.

2. Research Method

This study assumes that the road conditions follow the PKJI standard [26]. The calculation of density standards is based on the PKJI standard. The classification of vehicles is divided into three types, namely motorcycles, light vehicles, and heavy vehicles. A light vehicle is a vehicle that has a length between 2.5 meters to 5.5 meters. Meanwhile, a heavy vehicle has a length longer than 5.5 meters. The road is also assumed to be free of non-motorized vehicles and pedestrians. The nodes will be placed in an area that is not exposed to other than motorized vehicles to avoid the pedestrian or non-motorized vehicles detected. Nodes also are placed on each required lane segment. For example, if a path has two lanes, it requires two nodes.

The research method is carried out in stages and begins with analysis and model development. Thus, after the analysis is made, materials and tools are prepared. In this research, we are doing node development and web development. After those phases were done, system integration was conducted. The integrated system will be installed at a pre-determined location. The system will be tested and evaluated in the testing phase. The result of this research is a ready-to-use IoT prototype for recording, classifying types, and calculating the average number of vehicle speeds.

2.1 Analysis and Model Development

The first stage of this research is gathering the system requirements and developing the detection model. In this stage, we analyze the road conditions, calculate road capacity, determine road service standards, and design a system diagram.

Several road conditions that were considered are shoulder condition, shoulder width, and road traffic. This stage also determines the sensors required in the development and road capacity calculation. Road capacity is calculated according to [26]. Commonly, road capacity became the standard for calculating road services. The result of this phase is the capacity of each lane. The equation for calculating the road capacity is shown in Equation 1.

$$C = C_0 \times FC_{LJ} \times FC_{PA} \times FC_{HS} \times FC_{UK} \quad (1)$$

with C is road capacity (passenger car unit (pcu)/hour), C_0 is base capacity (pcu/hour), FC_{LJ} is capacity adjustment factor related to lane width or traffic lane, FC_{PA} is capacity adjustment factor related to direction separation, only on the undivided road, FC_{HS} is capacity adjustment factor related to side resistance class for the road with roadside, and FC_{UK} is a capacity adjustment factor related to city size.

The level of road service will determine whether a road is declared smooth or jammed. The level of road service uses the standard degree of saturation. Degree of saturation (DS) refers to the ratio of current to capacity. The formula for calculating the degree of saturation is shown in Equation 2 [26].

$$DS = Q/C \quad (2)$$

with *C* is road capacity (pcu/hour), and *Q* is traffic flow per unit time (pcu/hour).

The result of DS is used to determine the service level of the road. The standard of road service is categorized into six levels, namely A, B, C, D, E, and F. Each level of road service has a lower limit value. Table 1 shows the range for DS to service level, respectively. The system diagram is designed to fulfill the purpose of making this system. The communication scheme between the nodes and the API is determined along with the encryption used.

Table 1. Relation between DS and service level [27]

Service level	DS	Description
A	[0, 0.20]	Smooth
B	(0.20, 0.45]	Crowded smoothly
C	(0.45, 0.70]	Crowded
D	(0.70, 0.85]	Crowded slowly
E	(0.85, 1.00]	Congested
F	(1.00, ∞]	Total congested

2.2 Material and Tools Preparation

This paper aims to develop a prototype of the use of IoT technology in recording the number, classification, and average speed of vehicles. Two ultrasonic sensors were used to detect vehicles. A Raspberry Pi 3 Model B is utilized as a computational device to calculate vehicles' numbers and classification. Two breadboards were used as connection boards for sensors and cables. A 16 GB Micro SD Card is used as a storage media for the Raspberry Pi. The hardware that will be used as a

DOI: https://doi.org/10.29207/resti.v6i5.4082

development environment and a server is a personal computer with specifications of Intel Core i3-3217U 1.80 GHz, 8 GB RAM, and 750 GB HDD. The software needed for development is MySQL 14.14 as a DBMS, NodeJS as a programming language for APIs, ExpressJS as a REST framework, Python 3.6 as a programming language for nodes, and Postman as an API testing application.

2.3 Node Development Phase

The node development phase consists of design, development, and testing. During the development, several iterations occur between development and testing. At this stage, the output of this phase is the assembled node. The design stage aims to design the required nodes and determine the detection method, calculation method for detecting the vehicle speed, and vehicle classification method. The sensor used is an ultrasonic sensor [28]. The arrangement of the sensors corresponds to that applied by [29], as shown in Figure 1.

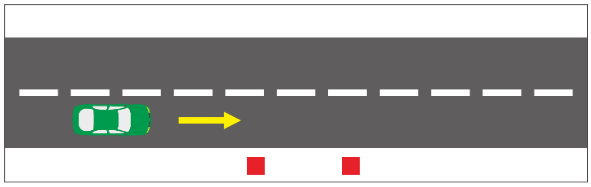

Figure 1. The sensor arrangement of congestion traffic detection

The installed node will be used to detect and indicate a passing vehicle. The distance between the sensor and a vehicle is used to indicate a passing vehicle. The detection method is based on changing conditions based on the given conditions [30]. An initial time is stated when the vehicle entered the state. The calculation of the vehicle's speed utilizes the law of Uniform Motion. This study assumes that passing vehicles are not accelerating suddenly. The formula for calculating speed is shown in Equation 3 [31].

$$v_v = \frac{S_v}{\Delta t} \quad (3)$$

with $v_v$is the speed of the vehicle (m/s), S_v is vehicle mileage, and Δt is the travel time (s). Vehicle mileage is the distance an object moves from one point to another. Travel time is the difference between the start time at the starting point and the end time at the endpoint when an object moves.

In addition to calculating speed, vehicle classification also uses the Uniform Motion Law. This study classifies vehicles based on the length of the vehicle. By utilizing the speed obtained in Equation 3, the length of the vehicle can be obtained by using Equation 4 [31].

$$l_v = v_v \times \Delta t \quad (4)$$

where l_v is the length of the vehicle (m), v_v is the speed of the vehicle (m/s), and Δt is the travel time (s).

At the development stage, the prepared design and equipment will be applied at this stage. The algorithm is implemented using the Python 3.6 programming language by utilizing the gpiozero library to read input from the ultrasonic sensor. The testing stage serves to check the results of the development carried out. Tests are carried out starting from the sensor's accuracy in detecting the distance, classifying the vehicle, and calculating the vehicle's speed.

2.4 Web Development Phase

This stage is carried out simultaneously with the node development stage. This stage will produce an API for the system. The API is used for the nodes can send detection results to the database quickly and safely. This phase consists of three stages: API design stage, Web development stage, and Web testing and evaluation. The API design stage will produce a database schema and an API schema—the use case of each existing role also designed in this stage. The user roles that can access the API are device and administrator. Each role has its limitations in accessing the API. The design determined in the previous stage is applied at this stage. Development is divided according to the number of existing modules. Web development uses javascript programming language with React library for front-end development, NodeJS with ExpressJS framework for API development, and MySQL for DBMS. The web testing and evaluation stage tests the development results carried out. The test is divided into two tests: the API and the front-end. The API test is carried out to test the success of the API in receiving the response and returning the proper request, which is a test carried out manually with the Postman application and automatically with the mocha framework. The front-end test is carried out to test the success of the reliability of the front-end.

2.5 Integration system testing

This stage will integrate the node and APIs that have been developed separately. The integration stage is essential for this development because the success of IoT depends on the successful integration of nodes and APIs. After performing the integration between the node and the API, testing is carried out. The test is divided into two tests: communication and field tests. Communication testing is carried out to ensure communication between the API and the node is successfully done. Testing is carried out as data exchange with an agreed cryptographic scheme, the tool for receiving the response given by the API, and the API in accepting the request given by the node. At this stage, the data sent by the tool is still in dummy form. Field testing will be carried out by installing roadside tools and APIs on the server. The difference with the communication test is that the data sent to the API results from the direct recording made by the node. The field test is expected to see the algorithm's accuracy for

DOI: https://doi.org/10.29207/resti.v6i5.4082

recording the number, classifying, and recording the speed of passing vehicles. Once the integration test is successful, the API and node are ready for final installation. The node will be placed in each lane of the road. The API will be installed on a server that has been provided with a public domain so that nodes can access it.

3. Result And Discussion

3.1 Analysis and Model Development

The road used as the test site is Babakan Tengah Road, Babakan, Dramaga, Bogor, West Java (Figure 2). The road is located around IPB University. Jalan Babakan Tengah Road has a busy to very crowded level. The road has one-track road and can be traversed by two vehicles with different lanes. Crowds usually occur in the morning and evening. Many vehicles are parked on the side of the Babakan Tengah Road, which causes the effective lane width to decrease from each side.

Figure 2. Babakan Tengah road, IPB Dramaga, Bogor

The calculation of the capacity of the Babakan Tengah road is based on [26]. The effective lane width is 2.5 m, roadside width is around 0.9 m, the type of road is 2/2 with roadside TT, the side obstacle level is medium (S), and the area size is under 0.1 million people. Based on those data, C_0 number s 2900, FC_{LJ} is 0.56, FC_{PA} is 1, FC_{HS} is 0.956, and FC_{UK} is 0.86. The capacity calculation, according to PKJI (2014) is obtained by using Equation 1.

$$C_{bateng} = C_0 \times FC_{LJ} \times FC_{PA} \times FC_{HS} \times FC_{UK} \quad (5)$$

$$C_{bateng} = 2900 \times 0.56 \times 1 \times 0.956 \times 0.86 \quad (6)$$

$$C_{bateng} = 1335{,}188 \text{ pcu/hour}$$

with C_{bateng} is road capacity for Babakan Tengah Road.

The capacity calculation above is for all lines—the road type 2/2 TT results in a capacity calculation for the entire lane. Thus, the calculation of the road capacity for each lane is shown in Equation 7. The calculation resulting from Equation 7 is 667,594 pcu/hour.

$$C_{b1} = \frac{C_{bateng}}{2} \quad (7)$$

The conversion of minutes for the road capacity is needed—the capacity value of every minute of the Babakan Tengah road as Equation 8. Based on Equation 8, the capacity value for each lane in Babakan Tengah Road for each minute is 11.126 pcu/minute.

$$C_{b\_minute} = \frac{C_{b1}}{60} \quad (8)$$

Once the capacity of each lane is obtained, the level of road service can be determined. The level of road service is based on the degree of saturation according to Equation 2. The equation for the degree of saturation for the Babakan Tengah road every minute and lane is shown in Equation 9.

$$DS_{Bateng} = \frac{Q_{Bateng}}{C_{b\_minute}} \quad (9)$$

with DS_{Bateng} is the degree of saturation of Jalan Babakan Tengah every 1 minute, and Q_{Bateng} is the traffic flow for Jalan Babakan Tengah every 1 minute (pcu/minute).

The system diagram of the Congestion Detection using IoT is shown in Figure 3. The HTTP/HTTPS protocol is used for communication between the Raspberry Pi (node) and the server (API).

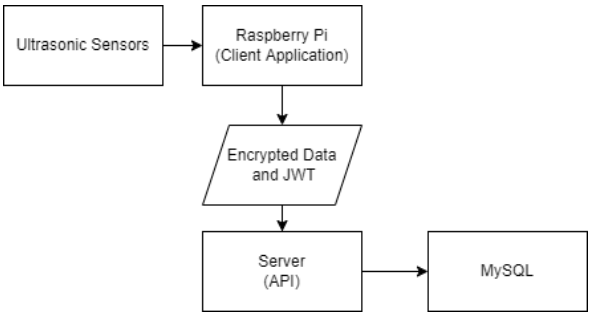

Figure 3. Block Diagram of the system

The data will be encrypted using the AES-256 algorithm to maintain the confidentiality of the data sent. The encryption and decryption keys use passwords automatically generated by the API during node registration. In addition to AES-256, each node must log in to get a valid JSON Web Token (JWT) from the API. To maintain data integrity, the data is equipped with a SHA512 hash. In addition to maintaining the confidentiality of data, this algorithm is implemented to ensure that data is transmitted by authorized nodes. This is because the administrator must first register the node to send data. Each node will be equipped username and password that is only known by the tool and the server.

The communication flow between the node and the API begins with sending data with a hash by the node in the

DOI: https://doi.org/10.29207/resti.v6i5.4082

form of JSON. JSON data is sent with JWT, which is obtained through the login process via headers. The API will process the data to be entered into the database. Before saving to the database, the API will check the authentication of the data obtained. The API will send the data storage success status to be received by the node. If the submitted data fails to be saved or is not accepted by the API, it is saved by sending it back with the following data. If the API successfully saves the submitted data, the node will delete the data. The communication process continues until the node is turned off.

3.2 Node Development Phase

The sensor's design for the lane is shown in Figure 4. The distance between the sensors (Su) is set at 30 cm. This reason is that the minimum length is not less than 30 cm. In addition, taking the distance between sensors that are not too far away prevents the vehicle from being detected in the second sensor because the vehicle shifts more than 1 meter. The marking of a vehicle by the sensor utilizes the distance between the vehicle and the sensor. The sensor is placed at an angle (α) of 90° to the vehicle's direction and the sensor's direction. The angle taken at 90° is set so that the sensor directly hits the vehicle's body, assuming the vehicle passes at an angle of 180° to the direction of the lane.

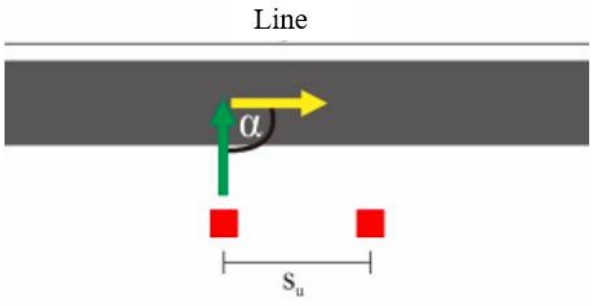

Figure 4. Distance and angle of the sensor's node

The marking of vehicles using sensors is divided into three conditions, as shown in Figure 5. The first condition (C1) is the condition of no vehicle. The first sensor is on in this state, and the second is off. The C1 stays put until the first sensor gets the vehicle through. Moving from C1 to the second state (C2) results in time ta. C2 is the condition of the vehicle found. The first sensor is off in this state, and the second is on. The C2 will stay put until the second sensor finds the vehicle found by the first sensor. The movement from C1 to C2 results in a time tb. The third condition (C3) is the vehicle's condition waiting for it to disappear. The first sensor is on in this state, and the second sensor is off. The C3 will stay put until the first sensor no longer finds the detected vehicle. The displacement of C3 to C1 gives the time tc.

Determining the sensor distance to a specific value and dividing vehicle detection into three conditions causes the formula for calculating vehicle speed to change. Using fixed distances from the sensors, ta, and tb, the formula for calculating vehicle velocity based on Equation 3 becomes Equation 10.

$$v_v = \frac{0.3}{(t_b - t_a)} \tag{10}$$

with v_v is vehicle speed (m/s), ta is the time to move from the first state to the second state (s), and tb is the transition time from the second state to the third state (s).

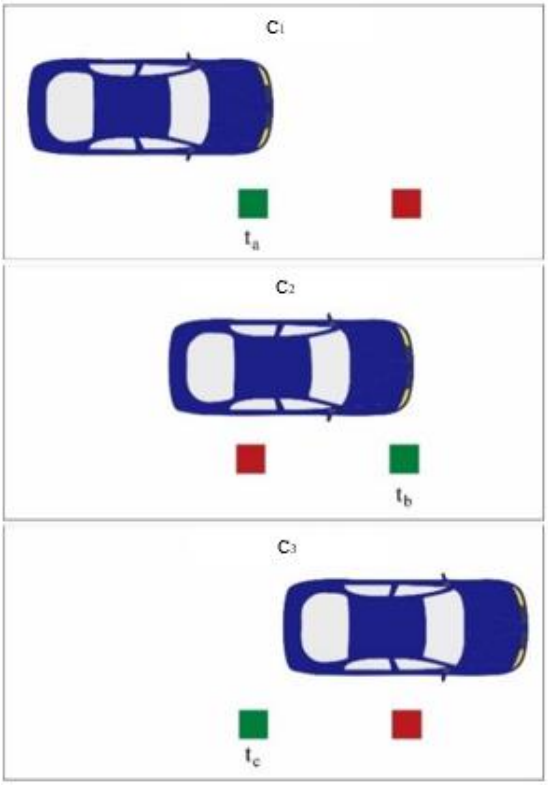

Figure 5. Condition state of vehicle's detection

In addition to the equation to calculate the speed, Equation 4 to calculate the length of the vehicle also changes. The vehicle length calculation uses the results from C2 and C3. The value of t uses the values of tb and tc. Because C1 uses the body of the vehicle, the length of the vehicle is reduced by the value of Su. The value of the length of the vehicle must be added to the value of Su. The equation for calculating the length of the vehicle becomes Equation 16.

$$l_v = 0.3 + \left(v_v \times (t_c - t_b)\right) \tag{11}$$

with l_v is length of the vehicle (m), t_c is transition time from the third state to the first state (s).

The length of a vehicle is used to classify the vehicle. The length of the vehicle is obtained from observing and checking the websites of automotive companies in Indonesia. The collection of data obtained is processed

DOI: https://doi.org/10.29207/resti.v6i5.4082

into a threshold to classify a vehicle. Figure 6 shows the rules of vehicle classification.

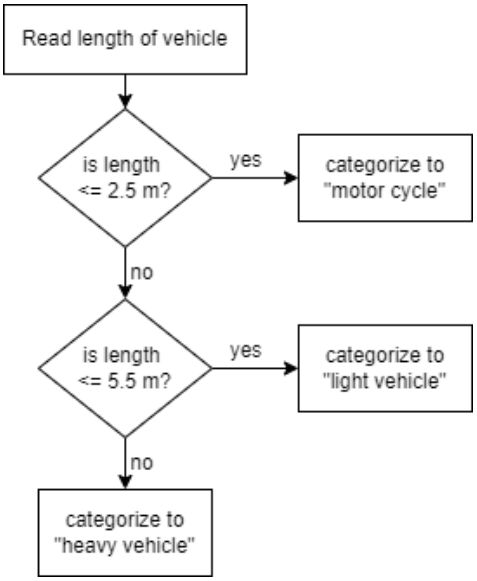

Figure 6. Vehicle classification rule

The node development stage produces a node with the application installed on the Raspberry Pi. The nodes are arranged according to the design stage. The application on the Raspberry Pi is used to process the data generated by the sensor. To ensure the data obtained from the sensors is accurate, the calibration process is executed. The first (U1) and second (U2) ultrasonic sensors were calibrated at 0.5 m to 3 m with a displacement of 0.5 m. Table 2 shows the calibration results of the U1 and U1 sensors. The calibration results show that the sensor value is relatively correct because the error rate is below 5% for all distances.

Table 2. Ultrasonic sensors calibration result

Distance between the object and sensor U1 (meter)			Distance between the object and sensor U2 (meter)		
Manual	Sensor	% Error	Manual	Sensor	% Error
0,5000	0,4985	0,2998	0,5000	0,4929	1,4202
1,0000	0,9916	0,8362	1,0000	0,9797	2,0266
1,5000	1,4864	0,9048	1,5000	1,4838	1,0805
2,0000	1,9836	0,8199	2,0000	1,9811	0,9449
2,5000	2,5014	0,0548	2,5000	2,4877	0,4934
3,0000	3,0075	0,2506	3,0000	2,9652	1,1596

The linear regression functions for sensors U1 and U1 obtained from the data are used to calculate the distance of the vehicle object to the U1 sensor. The arrangement of the nodes is shown in Figure 7. The U1 sensor is connected to GPIO pins 14 for triggers and 15 for echo. At the same time, the U1 sensor is connected to GPIO pins 20 for triggers and 21 for echo. The sensors are arranged in a row, with the distance between the sensors is 30 cm, according to Figure 7.

In the module on the client application, there are three modules developed for the node. The first module is a module for retrieving data. This module only accepts data generated by the sensor. The lag time between transmissions of a sensor is set to 0.01 seconds. This module is divided into three main functions, as shown in Figure 5:C1 (no object), C2 (object found), and C3 (till end). The data generated by each vehicle detection is used to classify and calculate vehicle speed.

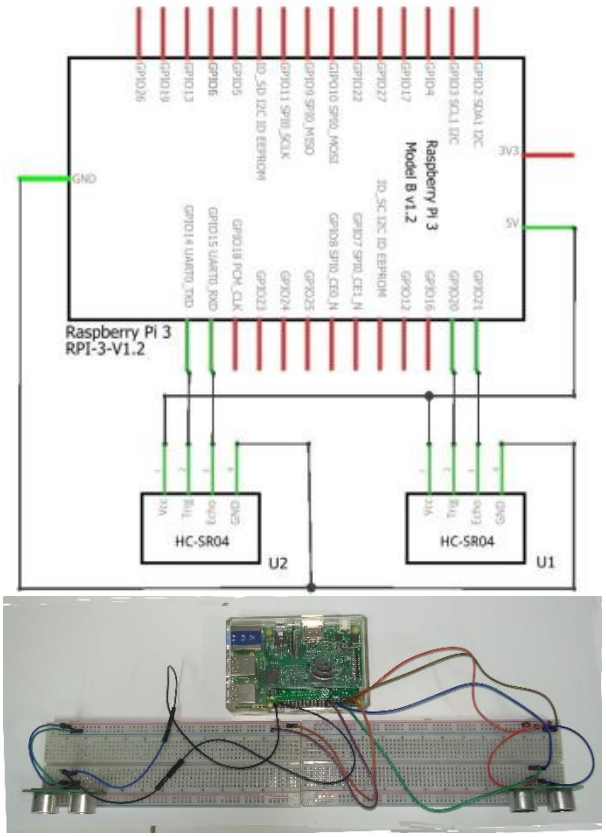

Figure 7. Circuit schematic of developed nodes and implementation result of the node

The second module is a module for classifying and calculating vehicle speeds. The data obtained from the previous module is used in this module. This module consists of two main functions: a function to classify vehicles (veh class) and a function to calculate the length and speed of a vehicle (veh dimension).

The third module is a module to calculate the number and average speed of motorbikes, light vehicles, and heavy vehicles and to calculate the degree of saturation of a lane according to Table 2. This module is run every minute. This module consists of two main functions: a function to calculate the number and average speed of motorcycles, light vehicles, and heavy vehicles (qty speed) and a function to calculate the degree of saturation (road status).

The fourth module is the data transfer module to the API. This module is called after the third module is finished. This module consists of two main functions: a function for sending data to the API (post func) and a

DOI: https://doi.org/10.29207/resti.v6i5.4082

function for encrypting data with the AES-256 algorithm and generating a hash of the data with the SHA512 algorithm (change to JSON)—implementation of the AES-256 and SHA512 algorithms using the pycrypto library in Python.

Table 3. Client application testing result

Module No.	Module Function	Number of Test Case			
		Total	Success	Failed	% Success
1	No object	3	3	0	100%
	Object found	6	6	0	100%
2	Veh class	3	3	0	100%
	Veh dimension	1	1	0	100%
	Qty speed	5	5	0	100%
3	Road status	8	8	0	100%
	Post func	4	4	0	100%
4	Change to json	1	1	0	100%

The system testing stage tests the reliability of the nodes that have been developed. The things that were tested were data encryption and data retrieval schemes. Based on Table 3, the client application is working well in accordance with the test.

3.3 Web development phase

The result of the database design stage is the database schema and use case of each role. The development of this system requires eight tables in the database schema (Figure 8), namely roads, segments, paths, lanes, nodes, severity status, vehicle data, and administrators. Each lane has a node so that the relationship created is 1:1. Each node representing a lane sends the degree of saturation and traffic data for that lane every 1 minute. This condition underlies the table relationship between nodes with a severity status of one to many. Because each data degree of saturation contains lane traffic data, namely the number and average of each type of vehicle, the table relation between severity status and vehicle data is 1:1. The administrator table has no relation to any table because the table stores administrator data with the administrator, master administrator, and super administrator roles that are not related to any data.

A use case for each existing role is designed to develop the database, as shown in Figure 9. The use case is for master administrator data, access for administrator data, and access for traffic data. The administrator role can view data on roads, segments, nodes, and traffic. The master administrator role has access like administrator plus can add, delete, and update road data; add, delete, and update segment data; add, delete, and update data nodes; register and delete administrator data, and view traffic data. The super administrator role has access to

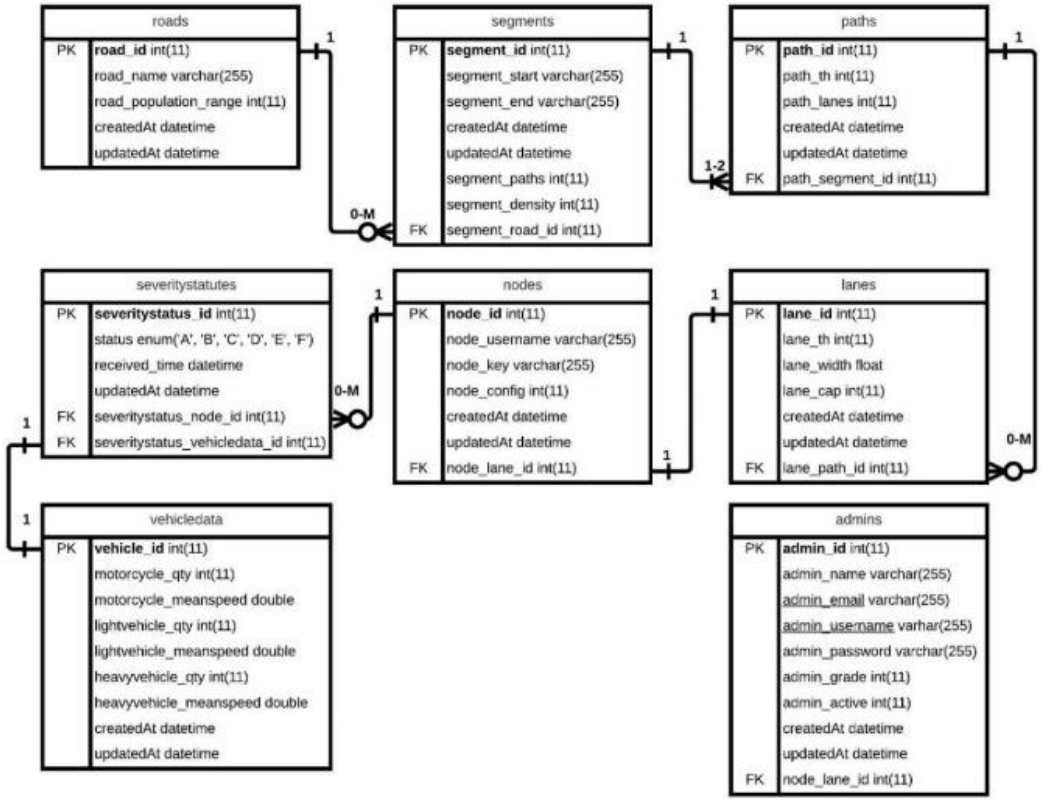

Figure 8. Database schema

DOI: https://doi.org/10.29207/resti.v6i5.4082

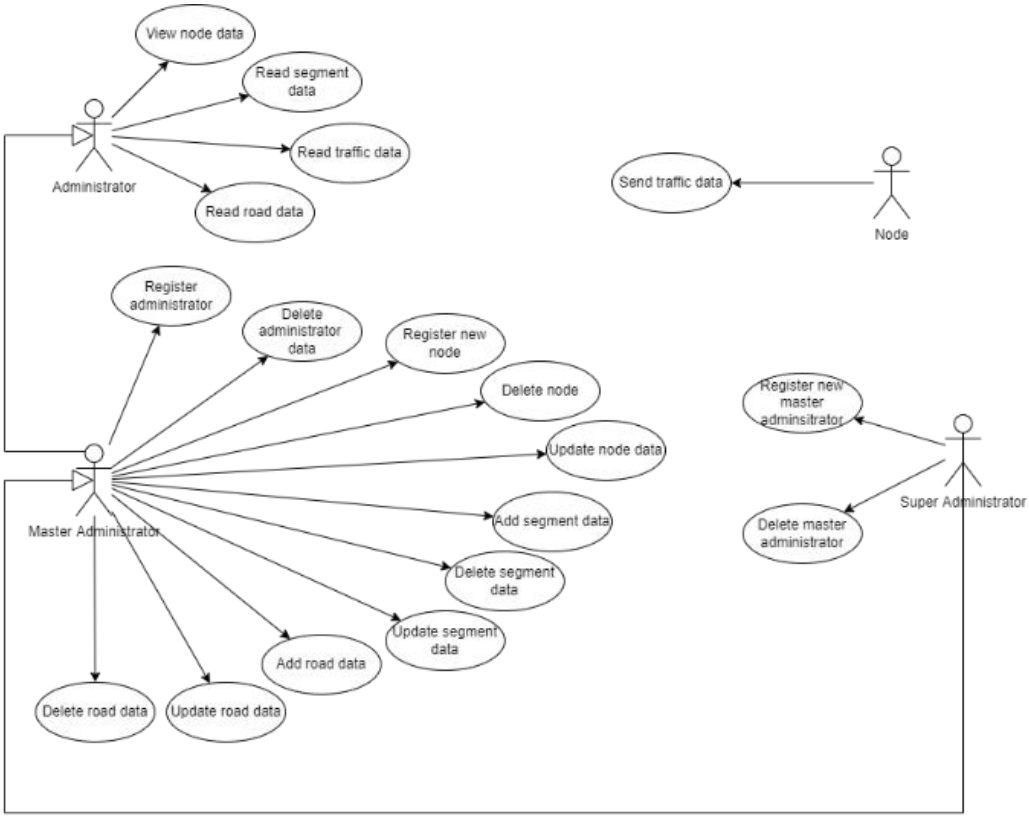

Figure 9. Use case diagram of the system

the master administrator and administrator, plus registering and deleting master administrator data. The role node can only send traffic data to the API.

broadly divided into four things: access for data paths, access for segment data, access for data nodes, access

The API development stage built the API according to a pre-determined design. The API is developed using the NodeJS programming language with MySQL DBMS. Several developed modules are the administrator module, road module, segment module, path module, lane module, node module, traffic data module, and home module. The administrator module consists of several features: administrator register, delete administrator data and login. This section applies to two roles, namely super administrator and administrator. In addition, communication between the front-end and the API in the administrator module to distinguish roles using tokens with JWT schemes.

The features contained in the road module are adding roads, deleting roads, editing road data, and viewing road data. The development of features on the API side is collected on a single controller. The features contained in the segment module are adding new segments, updating segment data, and deleting segment data. The side developed at this stage is only the API side. Segment data is tied to road data, so front-end development is combined with road module. The path module consists of several features: adding new paths, updating path data, and deleting segment data. The side developed at this stage is only the API side. The path module is closely related to the segment module. This module is only called by the functions contained in the segment module. The lane module only has one feature, which is adding lane data. Since the lane module is tied directly to the node, the side developed at this stage is only the API side. Front-end development for lane modules is automatically merged with development for node modules.

The part that is developed in the node module is the development of the API and the front-end. The features needed in the node module are adding nodes, logging in nodes, deleting nodes, and downloading configuration files. The development of features on the API side is collected on a controller. This stage also develops a feature to decrypt the data sent by the node. Decryption uses the AES-256 algorithm. The generated key will be saved in the configuration file. The data traffic module consists of a data traffic add feature and a data traffic view feature. The API and front-end of those features are developed. Severity status data and vehicle data are included in traffic data. The controller collects the development of features on the API side.

The modules developed in the home module are only on the front-end side because the features needed on the API side have been worked on in the previous module. The front-end view of the home module is divided into

DOI: https://doi.org/10.29207/resti.v6i5.4082

four essential parts (Figure 10). The website is used to show the congestion level of the road. The first part (point 1) displays the total number of lane data obtained by the system divided based on the congestion level. It will show the total number of the vehicles detected by the node and thus categorized it based on the severity level. The second part (point 2) displays the number of vehicle data of each type (motorcycles, light vehicles, and heavy vehicles) for each congestion level. The third part (point 3) displays the average speed of each type of vehicle (motorcycles, light vehicles, and heavy vehicles) for each congestion level. The fourth section (point 4) displays the system's total number of lane data every minute and the congestion level.

API testing is done automatically and manually. Tests are carried out to ensure the API can return a response according to the request given by the front-end and node. Based on the test results, the developed API is running well. After several tests and improvements were carried out, all the specified test cases were completed. In addition to testing the API module, testing was also carried out on data description. Tests are carried out to ensure that the data sent by the node can be retrieved as it was initially. In addition, testing is carried out to check the authenticity of the data (hash checking). The API must be able to distinguish the original data from the non-genuine data.

3.4 System Integration

This stage is carried out by merging the nodes with the API that was created in the previous stage. This stage ensures that the planned analysis of the system does not match the results that have been done.

3.5 Integration System Testing

The communication test uses dummy data as test material. The node will as if sending data to the API. The test results show that the node and API can communicate well. The API can properly decrypt the encrypted data sent by the node. The results of the response from the API can also be understood by nodes well (Table 4). The test results show that the node and API can communicate well after several tests and improvements have been carried out. The potential data lost daily due to communication between nodes and the API for all test cases is below 5% (72 of 1440 data).

Field tests were conducted to measure the algorithm's reliability in measuring and the algorithm's accuracy in calculating the number and classifying the types of passing vehicles. The field test was conducted on May 10, 2018. The field test was divided into three sessions: the first session at 14.01 – 15.01, the second session at 16.25 – 17.25, and the third session at 17.43 – 18.43. The road conditions during the test are as shown in Figure 6.

Table 4. Communication between node and API testing result

Test case	No.of repetitions	Average of sending time (s)	Potential data loss[a]
Data sent and stored	5	0,3251	8
Data failed to save	5	0,4218	11
Timeout[b]	5	1,0227	25

[a]Potential amount of data lost due to communication between nodes and API when sending data every day
[b]The specified waiting time is 300 ms

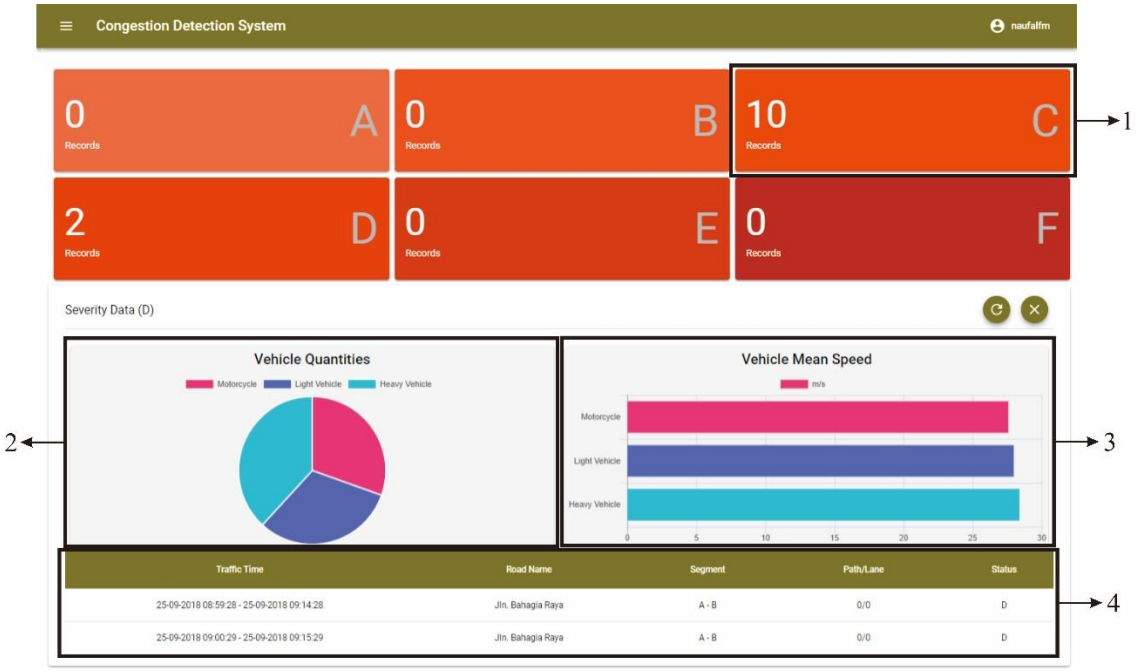

Figure 10. Home interface of the web application

DOI: https://doi.org/10.29207/resti.v6i5.4082

The average accuracy of detected vehicles by the system is 78,97%. The number of people passing on the roadside causes an error in vehicle classification. The developed node cannot distinguish between vehicle and non-vehicle objects. Non-vehicle objects are considered vehicle objects. The person passing by in Figure 11 is considered a vehicle object, in this case they were detected as motor cycle. Because the length of a human being is less than 2.5 meters, a human being is considered a motorbike according to the rules in Figure 6. In addition, the object of city transportation (angkot) is considered as two motorbikes. The wrong classification of angkot is caused by the opened door in the middle of the vehicle. The node recognizes it as two different objects. Heavy vehicles rarely pass through this route, so there is no data on large vehicles in this study. In the future, we will consider about implementing the system to a bigger lane. Table 5 shows the results of the field test conducted on Jalan Babakan Tengah, Babakan, Dramaga, Bogor. The error rate is relatively high because there is a significant data difference between the nodes and the manual.

Table 5. Field testing result

Vehicle type		Number of vehicles for each session		
		1	2	3
Motor cycle	Node	373	360	415
	Manual	272	341	391
	Difference	121	19	24
Light vehicle	Node	0	0	0
	Manual	21	19	24
	Difference	21	19	24
Heavy vehicle	Node	0	0	0
	Manual	0	0	0
	Differences	0	0	0
Total	Node	373	480	524
	Manual	293	380	415
	Difference	80	100	109

The cause of the high error generated is because the distance between the sensors is short, which is 30 cm. The short distance causes another sensor captures the outgoing signal generated by one sensor. As shown in Figure 11, the signal generated by sensor U2 is captured by sensor U1. Nm is the normal line for the vehicle, N1 is the normal line for sensor U1, Wd is the incoming signal, and Wp is the reflecting signal. Signal capture error occurs because the ultrasonic sensor tolerates the angle of incidence with the sensor normal line of a signal (θ). The tolerance angle of the ultrasonic is 15°. When U2 transmits a sound signal with a slope of β, the signal is not reflected exactly 180°to the incoming signal. The angle of this phenomenon is referred to as angle of γ. The incorrect value of γ causes the reflected signal to be not captured by sensor U2 but by sensor U1. The reflected signal is within the tolerance range of the U1 sensor, so the U1 sensor receives the signal as input. This condition causes the vehicle detection based on Figure 4 to fail. This condition also applies to the sensor U2, with the same occurrence. This condition occurs when the value of γ is between 0° to 15°, which is not recommended.

One possible way to overcome this problem is to increase the distance between sensors (Su). The Su value needs to be determined so that the signals received by the two sensors do not overlap. As shown in Figure 11, the Su value is influenced by the distance between the sensor and the vehicle (Wd). Based on the Pythagorean law, the general equation for determining the value of Su is as in Equation 12.

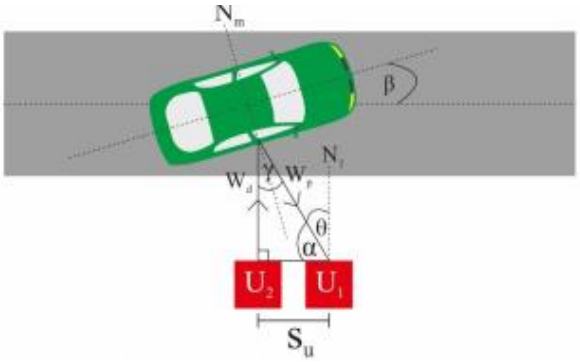

Figure 11. The condition of the signal capture error by a sensor

$$S_u = (W_p^2 - W_d^2)^{1/2} \quad (12)$$

The value of Wp cannot be ascertained because it is influenced by the vehicle's distance from the sensor or the value of Wd. The value of Wp must be replaced by something like Equation 13.

$$W_p = \frac{W_d}{\sin \alpha} \quad (13)$$

By combining Equations 12 and Equation 13, the equation to determine the value of Su becomes as in Equation 14.

$$S_u = W_d \left(\frac{1}{(\sin \alpha)^2} - 1\right)^{1/2} \quad (14)$$

This case occurs because the value of α is in the range of 75° to less than 90°. This case means that the minimum value is less than 75°. It can be seen in Equation 14 that the value of α is inversely proportional to the value of Su. Thus, the Equation 14 is derived to Equation 15. The Equation 15 is used to calculate the optimal Su value to avoid the overlapping signal problem. The Wd value entered the equation is the highest possible value or the farthest possible distance between the sensor and the vehicle.

$$S_u > W_d \left(\frac{1}{(\sin \alpha)^2} - 1\right)^{1/2} \quad (15)$$

with S_u is the distance between sensors (m), and W_d is distance between sensor and vehicle (m).

4. Conclusion

The recording of traffic data consisting of congestion level, number, and average vehicle speed per type was

DOI: https://doi.org/10.29207/resti.v6i5.4082

successfully carried out. Communication between nodes and API can be done well. Data exchange involving encryption and decryption with AES-256 went well. The data sent can also be appropriately verified from sources allowed to send data. However, the use of sensors on nodes has not been able to run correctly. The error rate in calculating the number of vehicles is still very high. Further development for reducing the error rate is needed, such as adding the boundary value to distinguish between human and vehicles, and finding the appropriate distance between the node and the lane.

References

[1] G. Cookson, "INRIX Global Traffic Scorecard," p. 44, 2018.

[2] S. Sumadi, "Kemacetan Lalulintas Pada Ruas Jalan Veteran Kota BrebeS," Universitas Diponegoro, Semarang, Indonesia, 2006. Accessed: May 19, 2022. [Online]. Available: http://eprints.undip.ac.id/15909/1/Sumadi.pdf

[3] BPS - Statistic Indonesia, "Statistical Yearbook of Indonesia 2014," Badan Pusat Statistik, 2014. Accessed: May 19, 2022. [Online]. Available: https://www.bps.go.id/publication/download.html?nrbvfeve=OGQyYzA4ZDlkNDFhYThjMDJmYWQyMmU3&xzmn=aHR0cHM6Ly93d3cuYnBzLmdvLmlkL3B1YmxpY2F0aW9uLzIwMTQvMDUvMDUvOGQyYzA4ZDlkNDFhYThjMDJmYWQyMmU3L3N0YXRpc3Rpay1pbmRvbmVzaWEtMjAxNC5odG1s&twoadfnoarfeauf=MjAyMi0wNS0xOSAxNzo1MjoxMg%3D%3D

[4] BPS - Statistic Indonesia, "Statistical Yearbook of Indonesia 2017," Badan Pusat Statistik, 2017. Accessed: May 19, 2022. [Online]. Available: https://www.bps.go.id/publication/download.html?nrbvfeve=YjU5OGZhNTg3ZjUxMTI0MzI1MzNhNjU2&xzmn=aHR0cHM6Ly93d3cuYnBzLmdvLmlkL3B1YmxpY2F0aW9uLzIwMTcvMDcvMjYvYjU5OGZhNTg3ZjUxMTI0MzI1MzNhNjU2L3N0YXRpc3Rpay1pbmRvbmVzaWEtMjAxNy5odG1s&twoadfnoarfeauf=MjAyMi0wNS0xOSAxODowNDoxOA%3D%3D

[5] S. Ezell, "Explaining International IT Application Leadership: Intelligent Transportation Systems," Jan. 2010, Accessed: May 19, 2022. [Online]. Available: https://trid.trb.org/view/911843

[6] D. Chen, J. Ignatius, D. Sun, M. Goh, and S. Zhan, "Impact of congestion pricing schemes on emissions and temporal shift of freight transport," *Transportation Research Part E: Logistics and Transportation Review*, vol. 118, pp. 77–105, Oct. 2018, doi: 10.1016/j.tre.2018.07.006.

[7] R. D. Sapta, "Analisis dampak kemacetan lalu lintas terhadap sosial ekonomi pengguna jalan dengan contingent valuation method (CVM) (Studi kasus: Kota Bogor, Jawa Barat)," IPB University, Bogor, Indonesia, 2009. Accessed: May 19, 2022. [Online]. Available: http://repository.ipb.ac.id/handle/123456789/14161

[8] H. Yu, P. Wang, H. Zheng, J. Lio, and J. Liu, "Impacts of congestion on healthcare outcomes: an empirical observation in China," *Journal of Management Analytics*, vol. 7, no. 3, pp. 344–366, Apr. 2020, doi: https://doi.org/10.1080/23270012.2020.1731720.

[9] E. Bilotta, U. Vaid, and G. W. Evans, "Environmental Stress," in *Environmental Psychology*, Second edition., 2018. Accessed: Jun. 03, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119241072.ch4

[10] G. Li *et al.*, "Influence of traffic congestion on driver behavior in post-congestion driving," *Accident Analysis & Prevention*, vol. 141, p. 105508, Jun. 2020, doi: 10.1016/j.aap.2020.105508.

[11] Z. A. Abood, H. B. Taher, and R. F. Ghani, "Detection of Road Traffic Congestion Using V2V Communication Based on IoT," *Iraqi Journal of Science*, pp. 335–345, Jan. 2021, doi: 10.24996/ijs.2021.62.1.32.

[12] A. H. E. Fawal, A. Mansour, and M. Najem, "V2V Influence on M2M and H2H Traffics During Emergency Scenarios: Adaptive eNode-B for V2V Communications," *Global Advancements in Connected and Intelligent Mobility: Emerging Research and Opportunities*, 2020. https://www.igi-global.com/chapter/v2v-influence-on-m2m-and-h2h-traffics-during-emergency-scenarios/www.igi-global.com/chapter/v2v-influence-on-m2m-and-h2h-traffics-during-emergency-scenarios/232025 (accessed Jun. 03, 2022).

[13] R. Bauza and J. Gozalvez, "Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications," *Journal of Network and Computer Applications*, vol. 36, no. 5, pp. 1295–1307, Sep. 2013, doi: 10.1016/j.jnca.2012.02.007.

[14] D. Shrivastava and A. Agrawal, "A Survey Of Various Algorithms In congestion Detection In Vanet," vol. 3, no. 6, p. 5, 2014.

[15] S. Mohamed Hatim, S. Jamel Elias, N. Awang, and M. Y. Darus, "VANETs and Internet of Things (IoT): A Discussion," *IJEECS*, vol. 12, no. 1, p. 218, Oct. 2018, doi: 10.11591/ijeecs.v12.i1.pp218-224.

[16] S. Muthuramalingam, A. Bharathi, S. Rakesh kumar, N. Gayathri, R. Sathiyaraj, and B. Balamurugan, "IoT Based Intelligent Transportation System (IoT-ITS) for Global Perspective: A Case Study," in *Internet of Things and Big Data Analytics for Smart Generation*, vol. 154, V. E. Balas, V. K. Solanki, R. Kumar, and M. Khari, Eds. Cham: Springer International Publishing, 2019, pp. 279–300. doi: 10.1007/978-3-030-04203-5_13.

[17] N. B. Soni and J. Saraswat, "A review of IoT devices for traffic management system | IEEE Conference Publication | IEEE Xplore," *International conference on intelligent sustainable systems (ICISS)*, pp. 1052–1055, 2017.

[18] C.-H. Chen and K.-R. Lo, "Applications of Internet of Things," *ISPRS International Journal of Geo-Information*, vol. 7, no. 9, Art. no. 9, Sep. 2018, doi: 10.3390/ijgi7090334.

[19] P. Sadhukhan and F. Gazi, "An IoT based Intelligent Traffic Congestion Control System for Road Crossings," in *2018 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, Feb. 2018, pp. 371–375. doi: 10.1109/IC3IoT.2018.8668131.

[20] M. U.Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A Review on Internet of Things (IoT)," *IJCA*, vol. 113, no. 1, pp. 1–7, Mar. 2015, doi: 10.5120/19787-1571.

[21] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of Things (IoT): A Literature Review," *JCC*, vol. 3, no. 5, pp. 164–173, 2015.

[22] S. Sukode and S. Gite, "Vehicle Traffic Congestion Control & Monitoring System in IoT," *International Journal of Engineering Research*, vol. 10, pp. 19513–19523, Jan. 2015.

[23] D. Evans, "How the Next Evolution of the Internet Is Changing Everything," p. 11, 2011.

[24] S. Singh, B. Singh, Ramandeep, B. Singh, and A. Das, "Automatic Vehicle Counting for IoT based Smart Traffic Management System for Indian urban Settings," in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, Apr. 2019, pp. 1–6. doi: 10.1109/IoT-SIU.2019.8777722.

[25] G. Lakshminarasimhan, V. Parthipan, M. I. Ahmed, S. Harsha K Nvm, and D. D. Dhanasekaran, "Traffic Density Detection And Signal Automation Using Iot," *International Journal of Pure and Applied Mathematics*, vol. 116, no. 21, pp. 389–394, 2017.

[26] KEMENPU, "Pedoman Kapasitas Jalan Indonesia (Pkji)." KEMENPU, Jan. 01, 2014. Accessed: May 19, 2022. [Online]. Available: https://www.academia.edu/36420401/PEDOMAN_KAPASITAS_JALAN_INDONESIA_2014_LUAR_KOTA_

DOI: https://doi.org/10.29207/resti.v6i5.4082

[27] B. Titania, "Analisis Intensitas Bangunan Koridor Jalan Raya Cimahi berdasarkan Kapasitas Jalan," ITB, Bandung, Indonesia, 2017. Accessed: May 19, 2022. [Online]. Available: https://digilib.itb.ac.id/index.php/gdl/view/7424

[28] Y. Jo and I. Jung, "Analysis of Vehicle Detection with WSN-Based Ultrasonic Sensors," *Sensors*, vol. 14, no. 8, pp. 14050–14069, Aug. 2014.

[29] S. Taghvaeeyan and R. Rajamani, "Portable Roadside Sensors for Vehicle Counting, Classification, and Speed Measurement," *IEEE Trans. Intell. Transport. Syst.*, vol. 15, no. 1, pp. 73–83, Feb. 2014, doi: 10.1109/TITS.2013.2273876.

[30] S. Keawkamnerd, J. Chinrungrueng, and C. Jaruchart, "Vehicle Classification with low computation magnetic sensor | IEEE Conference Publication | IEEE Xplore," *2008 8th International Conference on ITS Telecommunications*, 2008, doi: 10.1109/ITST.2008.4740249.

[31] D. Halliday, R. Resnick, and J. Walker, *Fundamental of Physics 9th Edition*. Wiley, 2010.

DOI: https://doi.org/10.29207/resti.v6i5.4082