INVOTEK

Jurnal Inovasi Vokasional dan Teknologi

http://invotek.ppj.unp.ac.id/index.php/invotek

Vol. 21, No. 3, Oct 2021

ISSN: 1411 – 3411 (p)
ISSN: 2549 – 9815 (e)

Improvement of autonomous railway monitoring robot prototype as fault detection on railway

Ismail Rokhim[1*], **Disa Nabila**[1], **Adhitya Sumardi Sunarya**[1]

[1] Department of Manufacturing Automation and Mechatronic, Politeknik Manufaktur Bandung
Jl. Kanayakan No. 21 Dago, Bandung, Jawa Barat, Indonesia-40135

*Corresponding author: *ismailrokhim@polman-bandung.ac.id* Doi: https://doi.org/10.24036/invotek.v21i3.865

This work is licensed under a Creative Commons Attribution 4.0 International License

Abstract

One of the causes of accidents on trains to date has been caused by damage of railways such as cracked rails, broken rails and other rail defects. Currently, the inspection of the railway line is still carried out manually by the railway inspection officer or Petugas Penilik Jalan Rel (PPJ). A prototype of Autonomous Railways Monitoring Robot (ARMR) is a prototype of inspection robot that can perform rail line checks automatically and be monitored remotely to detect any faults on the railway. The robot detects the width of the rail fault by reading the number of pulse train sent by the encoder and the depth of the rail fault by reading the time delay signal received by the distance sensor. The result of rail fault and location are subsequently delivered to users by utilizing several features of the Internet of Things (IoT). Faults detected by ARMR prototype, but it is unable to distinguish a fault and rail connection. To overcome this weakness, a camera is used that will capture the shape of the fault or rail connection. The results of the experiment at a speed of 20 cm / s, the robot is able to detect the width of the rail fault of 78 mm with an error value of 1.67%, and a fault width of 5 mm with an error value of 2.48%. The robot also able to distinguish faults and connections with 72 % accuracy for the comparison of the distance between the ARMR prototype and the distance shown on google maps.

Keywords: Autonomous Railway Monitoring Robot, Internet of Things, Distance Sensor, Encoder.

1. Introduction

The train is a transportation that is able to carry a high number of passengers so that it is suitable for mass transportation. The maintenance of railway facilities and infrastructure is regulated and regulated in such a way by the Minister of Transportation of the Republic of Indonesia, namely in the Regulation of the Minister of Transportation No. 32 of 2011 concerning Standards and Procedures for Maintenance of Railway Infrastructure. The maintenance and inspection of the rails is currently carried out directly by the railroad inspector or also known as the Road Inspector Officer (PPJ). PPJ directly checks the condition of the rails by bringing equipment to detect the rails, including: a waterpass for checking the straightness and flatness of the rail, a tool used to detect defects on the rails that are not visible and an iron ruler for checking the width of gaps in the connection The inspection of broken rails is still carried out directly (visually) by rail inspectors [1].

The length of railroads in Indonesia reaches 5368 km [1], and the current inspections are still not effective. It can be seen from the results of the investigation data, the NTSC's 2019 railway accidents were 85.7% of the causes of train accidents caused by derailments, and the 2019 railway accident category was 75% of the control and supervision of the existing train facilities and infrastructure at a maximum. In a previous study, the Autonomous Railways Monitoring Robot was developed, which is a prototype inspection robot that detects damage to railroad tracks that can move automatically to detect faults on railroad tracks. This prototype can detect rail faults with a minimum width of 3 mm at a maximum speed of 15 cm/s and a minimum width of 5 mm at a maximum speed of 60 cm/s, with a maximum error of 7.4% or equal to 1.3 cm. The faults detected by the robot are in the form of faults in the connection area or faults as rail damage [2]. Robots with a similar mission were also developed using ultrasonic sensors and GPS to detect rail faults and their fault points, the robot was moved manually via radio control, while telemetry, GPS, and mission planner software

Received date 12/11/2020, Revised date 02/04/2021, Accepted date 31/10/2021

were integrated with each other used as controllers in autonomous mode. FPV cameras are used to monitor the state of the railroad tracks directly. Reports or notifications of the location of damage to the railroad tracks are accessed by the official website [3]. Research to detect rail faults has also been carried out using accelerometers on the X-axis and Z-axis displays. Rail fault information is displayed graphically on the monitor screen which is processed from an excel database. The location of the point of occurrence of rail damage is measured in longitude and latitude coordinates. The real condition of the rails can also be seen on the monitor screens of the two vision cameras and the surrounding environmental conditions using a remote vision camera [4].

The development of the Autonomous Railways Monitoring Robot (ARMR) prototype in this study is intended to improve the ability to detect rail damage compared to previous studies. The combination of laser sensor[9] and encoder[10] is used to detect the width and depth of the fault. The use of cameras is used to obtain visual information on the real condition of the types of rail damage, types of fractures or cracks.

2. Method

System development refers to the SDLC (Software Delveopment Life Cycle) Waterfall research system development method. The SDLC waterfall model is often also called a linear sequential model or a classic lifeline. The waterfall model is a sequential or sequential flow approach from software. The following are the stages of the waterfall model:

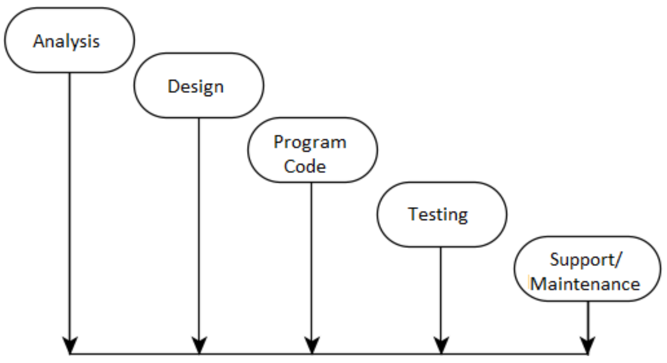


Figure 1. Stages of Waterfall Model System Development

1. Analysis
   The process of searching for hardware and software information domains, such as required functions, on tools and user interfaces. Analyze development needs and analyze system characteristics.
2. Design
   The design process is translating software requirements so that they can be implemented into programs according to the needs mentioned in the previous stage.
3. Program Code Generation
   Coding process which is the implementation of the design phase. There is information processing obtained from input (tool input), as well as features in the user interface (web).
4. Testing
   System testing process. In the form of direct testing (demo tools) and data testing with actual data.
5. Support or Maintenance This stage has not been implemented.

2.1 System Overview

The ARMR prototype has five parts, namely interface, cloud, controller, input and actuator. An overview of the system in general is shown in Figure 2.

P-ISSN: 1411-3414
E-ISSN: 2549-9815

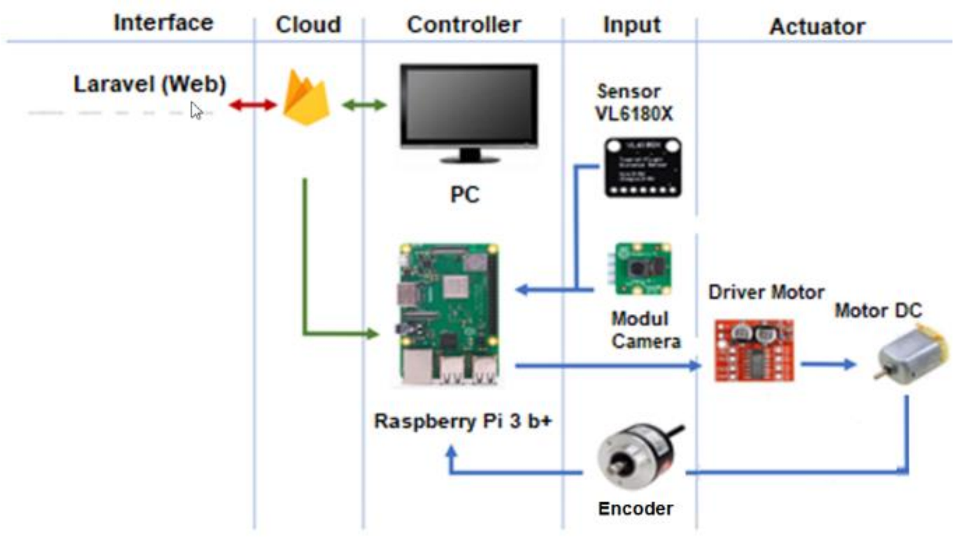


Figure 2. System architecture

The interface system uses the Laravel framework which functions as an intermediary between users and the system as well as for monitoring the movement of the prototype in real time. The information displayed on the website is the movement of the prototype which can be monitored remotely using the Google Maps feature to determine the position of the prototype and the location of the fault on the rail. The interface uses the cloud as a database storage medium with firebase. The controller used is a raspberry Pi 3 b+ as an actuator controller in the form of a motor and data receiver from sensor input. The proximity sensor used is VL6180X, as a detector of fault depth and railroad connections. The encoder is used to measure the width of the fault and generate data from the accumulated pulses sent by the encoder. The camera functions as a picture taker of the fault conditions detected on the rails.

2.2 Analysis of the Developed System

The system developed in this study is an ARMR prototype with an increased ability to detect rail damage, which consists of two detections of rail sections, namely fault detection and rail connection. The faults detected are in the form of faults as rail damage. The detected fault can be known with information on the depth and width of the fault in millimeters. Next, the ARMR prototype was able to distinguish rail joints and fractures. Rail joints are no longer detected as rail faults like the previous system, but as rail joints. Rail faults and connections can be found with location information and descriptions on the interface. The following scenarios are required by the user:

1. Information on the size of the depth of the fault
2. Fault width size information
3. Able to distinguish rail faults and joints

2.3 System Design

The design process consists of the steps taken to produce software so that it can be implemented into a program according to the needs that have been mentioned in the previous stage. The system consists of electrical and informatics parts in the form of program code.

2.3.1 Electrical Design

The electrical design used by the ARMR prototype is shown in Figure 3. In detecting the size of the fault depth and detecting the rail connection using the VL6180X sensor. The VL6180X sensor uses I2C communication to the raspberry pi on pin GPIO 2 as serial data (SDA) and pin 3 GPIO as serial clock (SCL) which carries data information between I2C and the controller. Devices connected to the I2C Bus system can be operated as master and slave. Master is a device that initiates data transfer on the I2C Bus by generating a start signal, ending data transfer by forming a stop signal, and generating a clock signal. While the slave is a device that is addressed by the master.

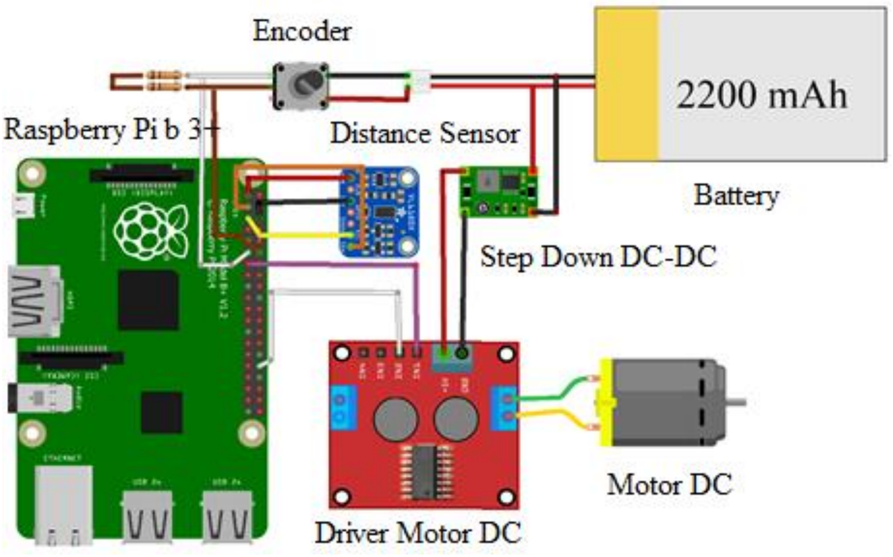


Figure 3. Electrical design of prototype

The encoder is used to measure the width of the fault depth and generate real-time data from the accumulated encoder pulses for monitoring the position of the ARMR prototype.

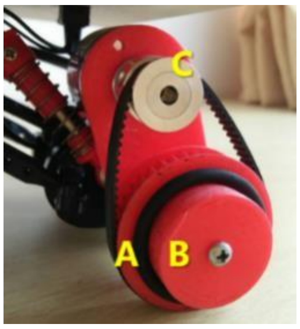


Figure 4. Rear left wheel of ARMR prototype

It is known that wheel A is the rear left ARMR prototype wheel. Wheel A on the robot is connected to the encoder through the timing belt to measure the rotational speed of the wheel. Pulse counts are required to obtain accurate fault width measurements.

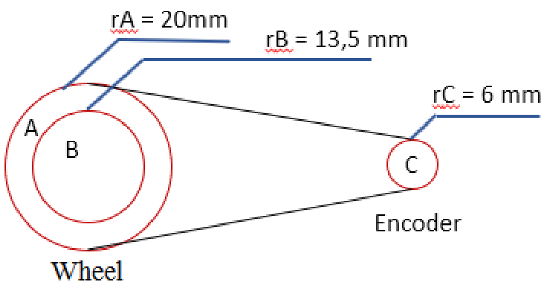


Figure 5. Calculation of encoder pulses

P-ISSN: 1411-3414
E-ISSN: 2549-9815

The pulse is obtained from the following calculations,

$$\text{Wheel Circumference} = 2\pi r \tag{1}$$

$$n_A = n_B \tag{2}$$

$$\frac{\text{Wheel B circumference}}{\text{Wheel A circumference}} = \frac{84{,}82}{125{,}66} = \frac{1}{1{,}5}$$

$$\frac{d_B}{d_A} = \frac{1}{1{,}5}$$

$$\frac{d_A}{d_C} = \frac{n_C}{n_A} \tag{3}$$

$$Pulse = \frac{1}{1{,}5} \text{ x wheel C circumference} = 25$$

$$Linear\ resolution = \frac{pulse}{PPR} = \frac{25}{100} = 0{,}25$$

Linear resolution is obtained from the pulse value divided by PPR (Pulse per Revolution) from the specification of the encoder used, which is 100. Linear resolution becomes a variable to detect the size of the fault width on the rail fault which will then be used in the program code using the python language.

2.3.2 Program Code Design

The control system on the ARMR prototype is in the form of an open loop control, which is input from the VL6180X sensor as a fault detector and is followed by a realtime program. The system flow diagram is shown in Figure 6. The user must select the speed, start-end location and press the start button so that the data enters the database and the raspberry executes the program. First, the motor is active and the VL6180X sensor works to detect the range between the sensor and the rail whether it passes the limit determining the rail connection, otherwise it will proceed to the limit value for the fault.

Next the ARMR prototype will execute commands to measure the depth and width of the fault. For faults, after the motor stops, the fault will be captured and then stored along with other data in the database, thus displaying real-time information along with the movement of the ARMR prototype until it reaches the final destination.

P-ISSN: 1411-3414
E-ISSN: 2549-9815

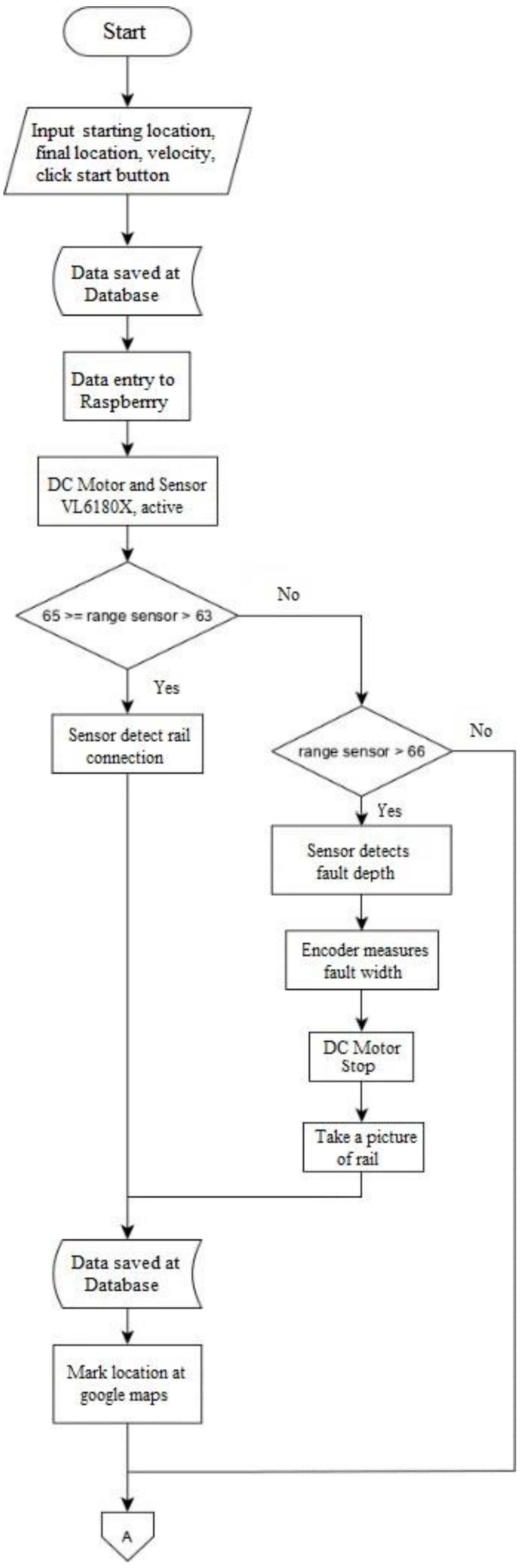

Start
Input starting location, final location, velocity, click start button
Data saved at Database
Data entry to Raspberry
DC Motor and Sensor VL6180X, active
65 >= range sensor > 63
No
Yes
Sensor detect rail connection
range sensor > 66
No
Yes
Sensor detects fault depth
Encoder measures fault width
DC Motor Stop
Take a picture of rail
Data saved at Database
Mark location at google maps
A

Figure 6. System Flowchart

P-ISSN: 1411-3414
E-ISSN: 2549-9815

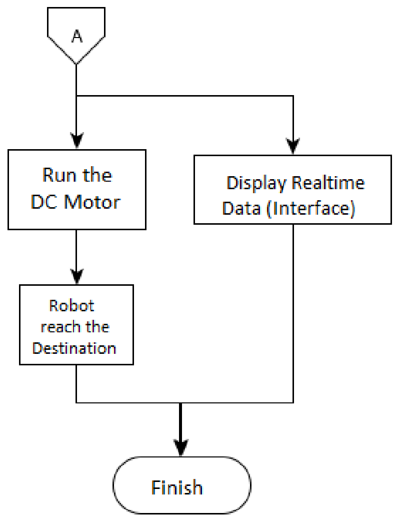
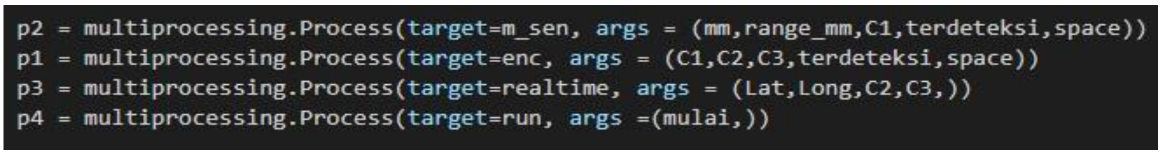

A
Run the DC Motor
Display Realtime Data (Interface)
Robot reach the Destination
Finish

Figure 7. Realtime ARMR Prototype Flowchart

2.4 Program Code Generation

In making the program, the author uses the python language and the multiprocessing module, which is a module that allows to maximize the use of the core on the raspberry pi so that running programs can work in parallel. Multiprocessing is capable of executing functions on new subprocesses. Each parameter will be serialized and sent to a new subprocess. The return function is serialized by the subprocess and returned to the main process.



```
p2 = multiprocessing.Process(target=m_sen, args = (mm,range_mm,C1,terdeteksi,space))
p1 = multiprocessing.Process(target=enc, args = (C1,C2,C3,terdeteksi,space))
p3 = multiprocessing.Process(target=realtime, args = (Lat,Long,C2,C3,))
p4 = multiprocessing.Process(target=run, args =(mulai,))
```

Figure 8. Multiprocessing in Python Functions

3. Results and Discussion

In previous studies, there was no information on the width of the fault and the depth of the fault, the information sent by the robot was only rail damage. By using the VL6180X type proximity sensor and camera installation, these weaknesses have been overcome. With the photo capture sent by the robot, cracks or broken rails can also be distinguished.

3.1 Rail Fault Test

ToF (Time-of-Flight) is a method for measuring the distance between the sensor and the object. The distance calculation is obtained from the difference in transmission when the signal is sent and received back by the sensor. The signal is a packet consisting of microwaves with a unique pattern, so the sensor will be able to recognize the signal.

The VL6180X sensor is a laser-based proximity sensor that produces a specific distance measurement. In principle, the laser beam is fired at the nearest object, reflected to the detector, and this module calculates the laser travel time (when it was fired and when it was received by the detector). The light pulse is directed at an object and the reflection is detected a time (T) so to produce data in the form of distance it is calculated as follows:

$$D = \frac{cT}{2} \tag{4}$$

Description: D = sensor distance to the object (m)
c = speed of light (m/s)
T = travel time (s)

The limit value for the distance between the VL6180X sensor and the rail is used for testing the sensor against faults at a certain speed. The iron used is hollow iron with a size of 30 mm x 15 mm and coated with black paint. Next, the depth of the fault is the sum of the sensor-rail distance, which is 63 mm plus a height of 15 mm so that the sensor distance to the fault is 78 mm as shown in Figure 9.

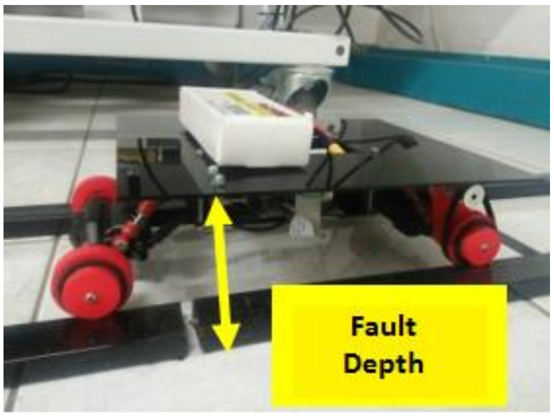


Figure 9. Fault depth

In dynamic conditions there are three variations of speed, namely 20cm/s, 30 cm/s and 45 cm/s. The test results of the VL6180X sensor readings at the depth of the rail fracture are dynamically shown in Figure 10.

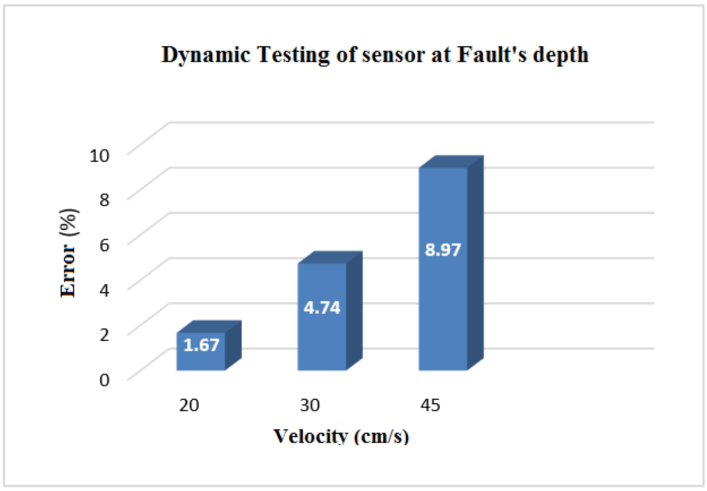


Figure 10. Sensor testing at fault depth

From Figure 10, it can be seen that the smallest error value of the VL6180X sensor readings against rail fractures dynamically at a speed of 20 cm/s, which is 1.67%. So that the speed of 20 cm/s is used as the speed for detecting rail faults.

3.2 Rail Fault Width Test

To prove the ability of the VL6180X sensor to detect the width of the fault, a test was carried out to detect the width of the fault with variations in the width of the fault. Figure 11 shows the results of fault detection based on variations in the width of the fault by adjusting the robot's movement at a speed of 20 cm/s.

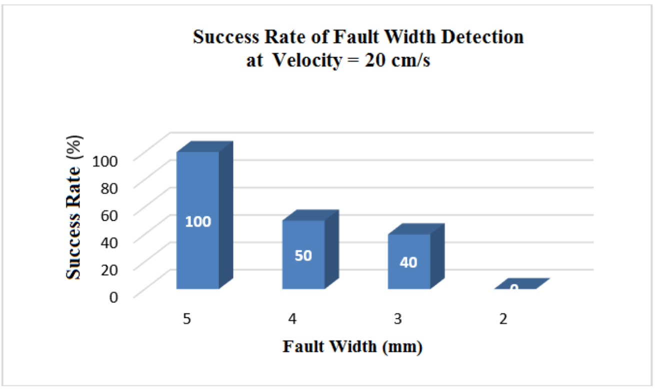


Figure 11. Success rate of fault width detection at a speed of 20 cm/s

From the above test on a fault width of 5 mm the success rate of fault detection is 100%, at a width of 4 mm the success rate decreases to 50%. Meanwhile, at 3 mm fault width, the success rate is only 40% and at 2 mm fault width is not detected at all. Therefore, the size of the fault width of 5 mm is used because it has a 100% success rate. The width of the fracture being tested is the distance between two rails with a width of 5 mm, as shown in Figure 12.

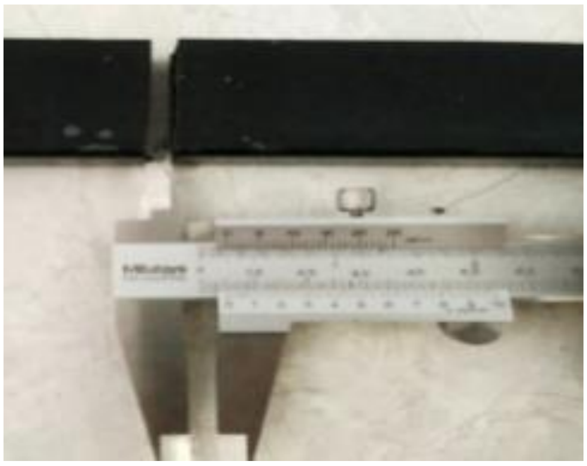


Figure 12. Fault width

The width of the fault was tested with three variations of speed, namely 20cm/s, 30 cm/s and 45 cm/s. The results of testing the width of the fault on variations in the speed of the robot's movement are shown in Figure 13.

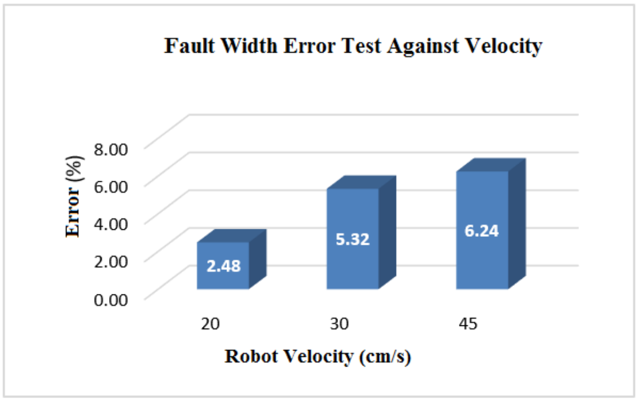


Figure 13. Testing Fault width against robot speed

P-ISSN: 1411-3414
E-ISSN: 2549-9815

It can be seen that the speed affects the measurement of the width of the rail fault. The highest error value from testing the width of the fault occurs at a speed of 45 cm/s at 6.24%. At lower speeds, the error value tends to decrease. At a speed of 30 cm/s, the error value is 5.32% and at a speed of 20 cm/s, the error value is 2.48%. So that the speed of 20 cm/s is used as the speed to detect the width of the rail fault.

3.3 Rail Connection Test

The rail connection is made by combining 2 hollow pieces with wood as an intermediary with a width of 5 mm. The detected rail connection is shown in Figure 14.

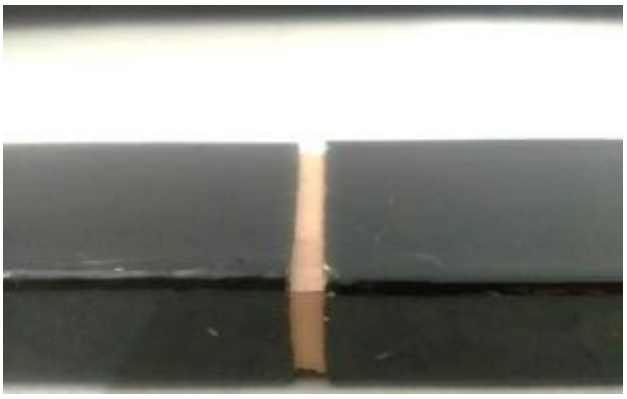


Figure 14. Rail connection

The rail connection test is carried out in the same way as in the fault detection test. To determine the accuracy of the VL6180X sensor against the rail connection, measurements were carried out as before, namely the actual distance between the sensor and the connection was 65 mm. Next, ten samples of data were taken for each speed. In dynamic conditions there are three variations of speed, namely 20 cm/s, 30 cm/s and 45 cm/s. The results of this test are shown in Figure 15.

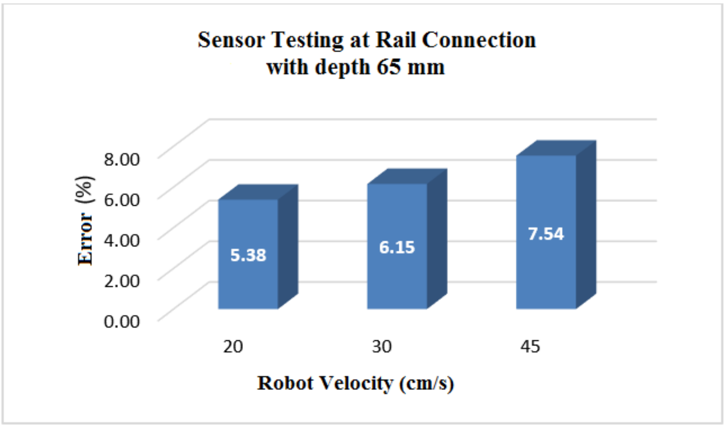


Figure 15. Testing sensor redings on robot speed

From the test results, it is known that the highest error value from testing the width of the fault is at a speed of 45 cm/s at 7.54%. At lower speeds, the error value tends to decrease. At a speed of 30 cm/s the error value is 6.15% and at a speed of 20 cm/s the error is 5.38%. So the speed of 20 cm/s is used as the speed to detect the rail connection.

3.4 Interface Test

Testing of rail fractures and joints is carried out simultaneously with interface testing to determine the comparison between the distance made on the prototype and the actual distance. The test scenario is from Ciroyom Station to Bandung Station which is shown on google maps. Figure 13 shows a map of the interfaces that are passed.

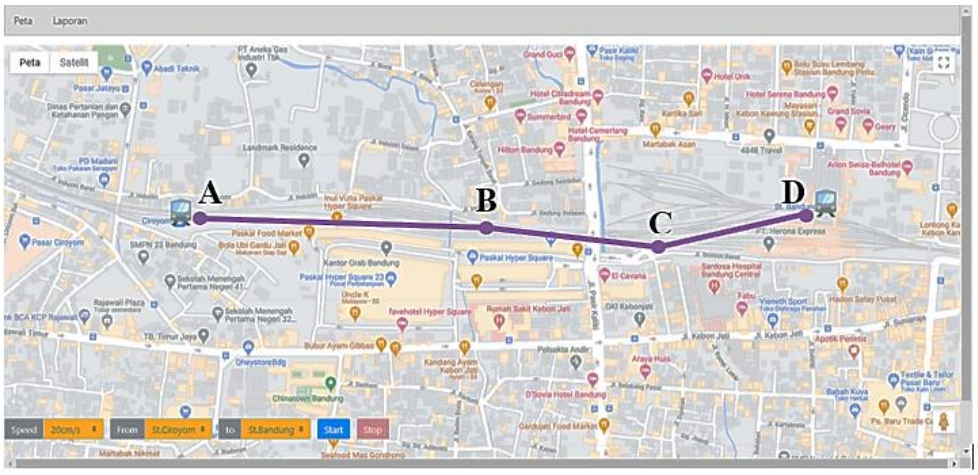


Figure 16. Interface map display and passed path

To find out the length of the path passed by the ARMR prototype, four points A to D were made following the rail line shown on google maps. Each point has a latitude and longitude position which is converted into UTM (Universal Transverse Mercator) to obtain a value in meters. Next, the position of each point is known by calculation using the following equation.

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \tag{5}$$

Value x as UTM Easting and y as UTM Northing. The D is the distance from the point A to B next. So that the results of the distance from one point to another are shown in table 1.

Table 1. Results of calculating the distance between points

Titik	Latatitude	Longitude	UTM Easting (m)	UTM Northing (m)	Jarak A-B (m)	Jarak B-C (m)	Jarak C-D m)
A	-691.413	1.075.902.879	786247.29	9234964.42	601.9	374.3	375.9
B	-691.413	107.595.731	786849.18	9234961.14			
C	-691.425.991	107.599.113	787223.09	9234944.71			
D	-69.139.511	1.076.024.977	787597.6	9234976.86			

From table 1 it is known the distance between points on the interface map, which is shown in Figure 17.

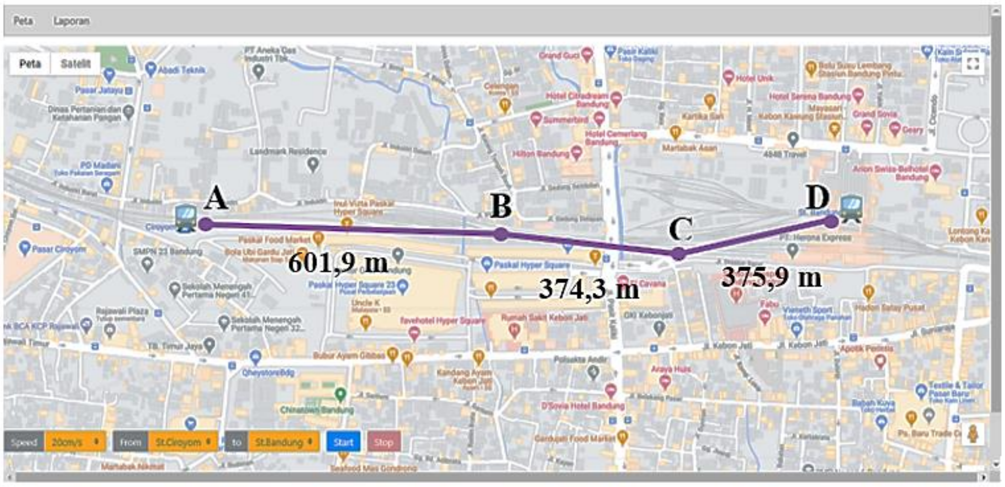


Figure 17. Distance between points on the map

P-ISSN: 1411-3414
E-ISSN: 2549-9815

The distance from point A to point D is 1352 meters. The path distance made in the prototype is 2.5 meters. So that we get a comparison between the distance of the path made in the prototype with the actual distance which is 2.5 meters: 1352 meters or equal to 1: 541 meters. Next, the results of the detection of faults and connections are displayed on the interface map menu where from the image three points of fault locations and one connection are shown which are marked with a red marking on the map interface, as shown in Figure 18.

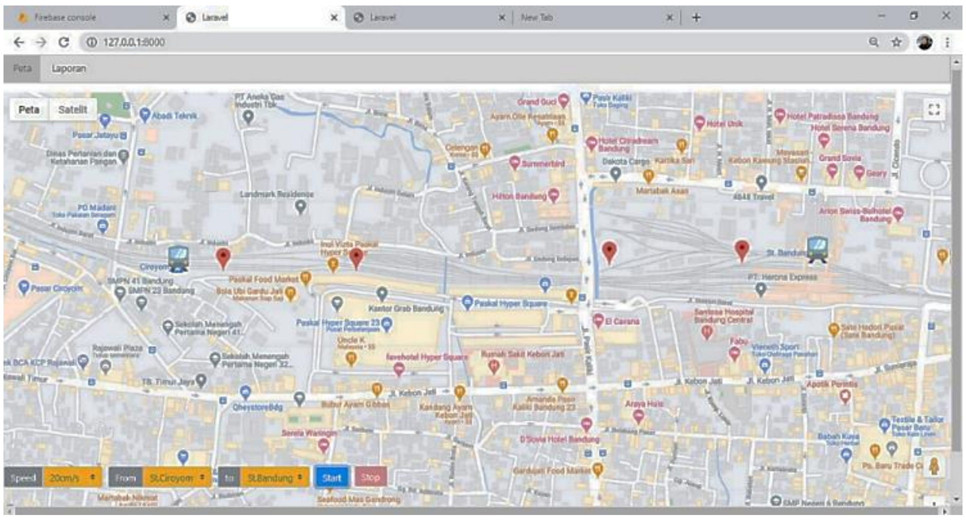
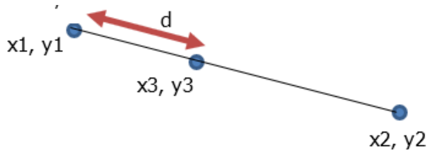


Figure 18. Display of detection results on the interface map

To compare the location of the faults and rail connections shown, the following calculations were carried out,



$$x3 = x1 + \frac{d}{D}(x2 - x1) \tag{6}$$

From the above equation, x3 and y3 are the latitude and longitude points from the fault location and detection rail connections. d is the distance from the closest point between points A to D to the point of fault location.

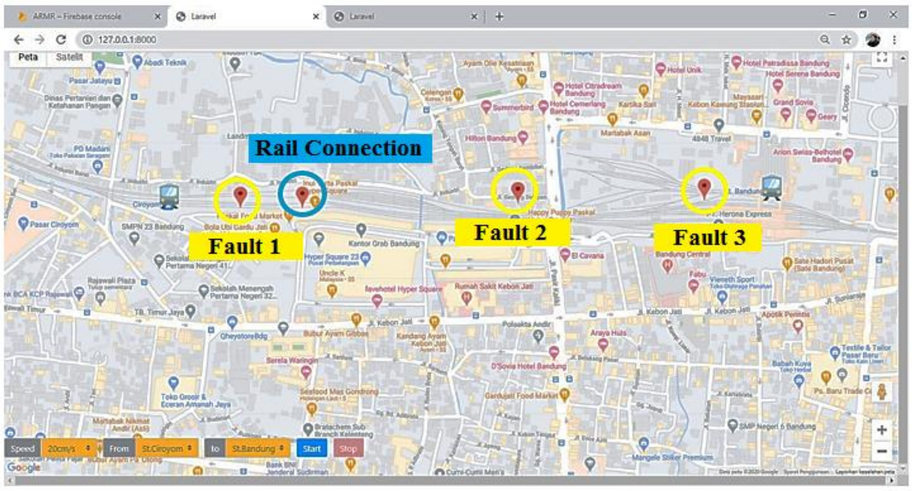
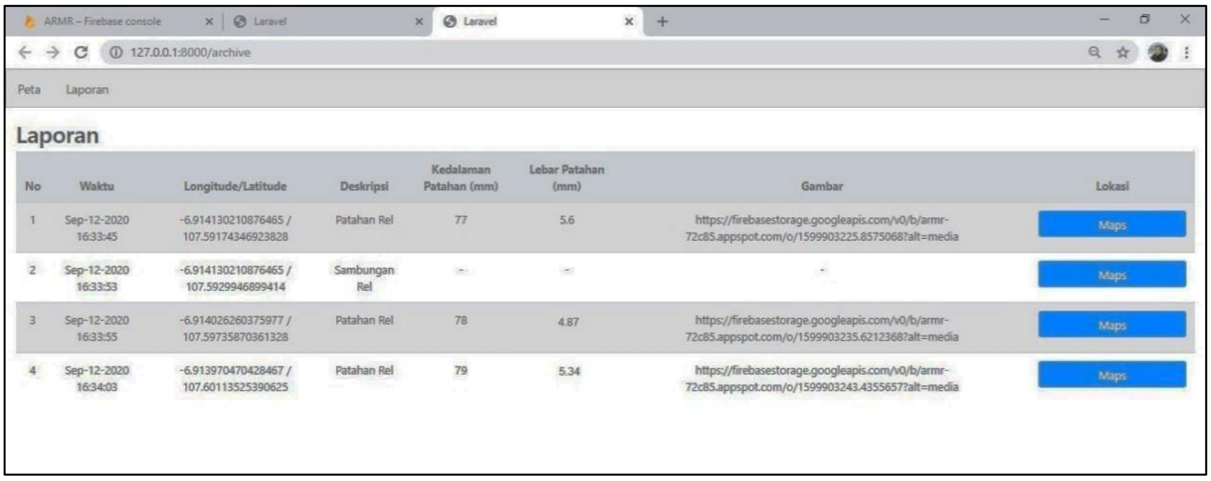


Figure 19. Detected faults and joints

P-ISSN: 1411-3414
E-ISSN: 2549-9815

Figure 16 shows the points of fault 1 (P1), connection (S1), fault 2 (P2) and fault 3 (P3). Next, in Figure 20, a report menu for the ARMR prototype robot is shown which contains information on the detected rail faults and connections.

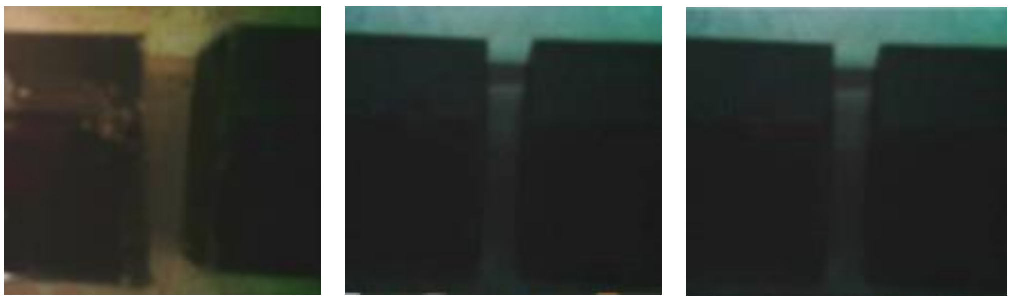


Laporan

No	Waktu	Longitude/Latitude	Deskripsi	Kedalaman Patahan (mm)	Lebar Patahan (mm)	Gambar	Lokasi
1	Sep-12-2020 16:33:45	-6.914130210876465 / 107.59174346923828	Patahan Rel	77	5.6	https://firebasestorage.googleapis.com/v0/b/armr-72c85.appspot.com/o/1599903225.8575068?alt=media	Maps
2	Sep-12-2020 16:33:53	-6.914130210876465 / 107.5929946899414	Sambungan Rel	-	-	-	Maps
3	Sep-12-2020 16:33:55	-6.914026260375977 / 107.59735870361328	Patahan Rel	78	4.87	https://firebasestorage.googleapis.com/v0/b/armr-72c85.appspot.com/o/1599903235.6212368?alt=media	Maps
4	Sep-12-2020 16:34:03	-6.913970470428467 / 107.60113525390625	Patahan Rel	79	5.34	https://firebasestorage.googleapis.com/v0/b/armr-72c85.appspot.com/o/1599903243.4355657?alt=media	Maps

Figure 20. Detection result report on interface

In Figure 20, a report table is displayed in the form of information on time, longitude and latitude, button-link map locations as well as a description of fault detection and rail connections. For fault detection in the image column, the url of the fault image is displayed using the Raspberry Pi camera module. The following pictures of fault 1, fault 2 and fault 3 are shown in Figure 21.


a. b. c.
Figure 21. Fault 1, b. Fault 2, c. Fault 3

The distance information from fault detection and rail connection is shown in Figure 22.

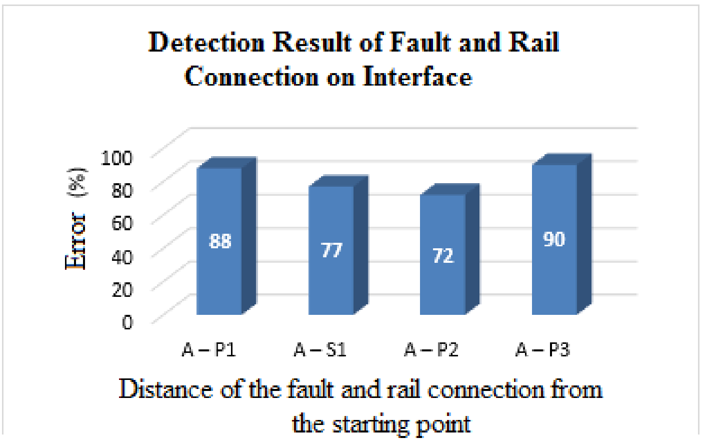


Figure 22. Fault 1, b. Fault 2, c. Fault 3

P-ISSN: 1411-3414
E-ISSN: 2549-9815

Figure 22 shows the results of the comparison of the distance on the interface map and the distance of the fault detected by the robot. The distance between the points in the table is the distance to the location of the rails where a fault is detected and the connection to point A is the starting point. It is known that the comparison of the distance traveled on the map with the actual distance is 1: 541. Then the distance traveled by the robot is the result of calculating the distance on the map divided by 541. Then the accuracy results of the faults and connections are detected where the smallest accuracy obtained is 72%.

4. Conclusion

Based on the development that has been done, the ARMR prototype robot is able to produce information on the size of the fault depth of 78 mm with an error value of 1.67% and information on the width of the fault 5 mm with an error value of 2.48% and can distinguish rail joints and faults and the accuracy results are 72% at comparison of the distance of the ARMR prototype with the distance shown by google maps.

References

[1] Kementrian perhubungan, "Menteri perhubungan republik indonesia," *Peraturan. Menteri Perhub. Republik Indones. Nomor Pm 115 Tahun 2018*, pp. 1–8, 2018.

[2] Siti Fatimah "Implementasi *Prototype Autonomous Railway Monitoring Robot* sebagai Pendeteksi Patahan pada Rel Kereta Api", Tugas Akhir D IV Politeknik Manufaktur Bandung, 2019.

[3] Rohmat Santoso, "*Autonomous Railways Monitoring Robot* Berbasis *Raspberry Pi* Sebagai Prototipe Robot Alat Bantu Petugas Inspeksi Rel Kereta Api", T. Elektronika, F. Teknik, U. N. Yogyakarta, 2019.

[4] Daryono Restu Wahono, "Mendeteksi Kondisi Rel Putus Menggunakan Akselerometer dan Kamera Visi", Puslit KIM-LIPI Serpong-Tanggerang, 2013.

[5] R. Sireesha, A. K. B, G. Mallikarjunaiah, and B. K. B, "*Broken Rail Detection System using RF Technology*," vol. 2, no. 4, pp. 11–15, 2015, doi: 10.1210/endo.137.10.8828507.

[6] R. Nicolau, "*Omnidirectional scanner using a time of flight senso*r by," no. February, 2018.

[7] P. Dana and B. Pemerintah, "Laporan Akhir Laporan Akhir," pp. 78–79, 2019.

[8] ST *Microelectronics*, "*Proximity and ambient light sensing* (ALS) *module* VL6180X (*Datasheet*)," *VL6180X datasheet*, no. *June*, pp. 1–87, 2016.

[9] A. Falamarzi, S. Moridpour and M. Nazem, "*A Review on Existing Sensors and Devices for Inspecting Railway Infrastructure*", Jurnal Kejuruteraan 31(1), 2019, pp 1-10 https://doi.org/10.17576/jkukm-2019-31(1)-01.

[10] N. Mahfuz, O. A. Dhali, S. Ahmed, and M. Nigar, "*Autonomous railway crack detector robot for Bangladesh: SCANOBOT*," in 5th IEEE Region 10 Humanitarian Technology Conference 2017, R10-HTC 2017, 2018, vol. 2018- Janua, pp. 524–527, doi: 10.1109/R10-HTC.2017.8289014.