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. 7 No. 3 (2023) 430 - 436 ISSN Media Electronic: 2580-0760

Implementation of Self Driving Car System with HSV Filter Method Based on Raspberry & Arduino Serial Communication

Kelvin Kristian Roestamadji[1], Florentinus Budi Setiawan[2], Leonardus Heru Pratomo[3], Slamet Riyadi[4]
[1,2,3,4]Department of Electrical Engineering, Faculty of Engineering, Universitas Katolik Soegijapranata, Indonesia
[1]kristian.kelvin29@gmail.com, [2]f.budi.s@unika.ac.id, [3]leonardus@unika.ac.id, [4]riyadi@unika.ac.id

Abstract

The development of technology in the transportation sector at this time is increasingly crucial. So the company innovates to create a car that can run itself with a high level of security. In this study, we designed an autonomous drive system for a 1:10 scale RC car using the main components in the form of a Raspberry Pi 4 and a Raspberry Pi camera as image processing for automatic control of an self driving car. Then the Arduino Nano, BTS7960, and Driver L298N components are used to regulate the movement of the DC motor. In this article, the control strategy of this self-driving car will be shown which will be implemented to detect lanes as a guide to walk autonomously. This study uses the HSV color filer method with morphology techniques to detect the path to be passed. This study resulted in a path detection that was very accurate and operated in real-time when compared to the CNN method using sampling paths to be passed that had previously been researched. After the path is detected, the interconnection between the mini computer and the microcontroller will work to synchronize the path detection and motor movement. In trials and hardware implementations carried out in the self-driving car laboratory with artificial intelligence, it can work according to the algorithm created with a success rate of 90%.

Keywords: self driving car; HSV;rRaspberry; autonomous

1. Introduction

The *self-driving car* is one of the technologies that are very promising to become a research prospect for *artificial intelligence* or what we usually know as *artificial intelligence* will become a technological revolution that will shape *self-driving cars*[1],[2]. Several technologies have been implemented for the development of this automatic moving car such as *AGV (Automated Guided Vehicle)* or the *self-driving car* itself, usually, the development of making this self-driving car uses *GPS technology*, *LiDAR*, *image processing*, *computer vision*, and many more[3]. Using technology such as *LIDAR*, *GPS*, *Pattern Guide*, and ultrasonic, has several advantages and has more information when implemented, for example, information on signs, obstacles, pedestrians, and others. We compare it with the use of *computer vision* or *image processing* which only uses input in the form of a camera that can detect what is in front of it.[4]-[6]

In the development that has been made by the CNN (*Convolutional Neural Network*) method, it has a road sample training method to be able to pass a *self-driving car* or it can be confirmed by this method that it will require a lot of sample training depending on the path to be traversed[7],[8] . This research designs a *Self-Driving Car* that will run in real-time with a path detection method using an *HSV* color filter with *morphology* and *Gaussian blur* techniques to perfect the detection of the path that the *self-driving car* will pass[9]-[13]. With the use of this method, we can freely operate this *self-driving car* without having to train the sample first.

This research uses a *raspberry pi 4b mini-computer* which will be operated to process image data or we can call it *image processing*. Then for the movement of the *self-driving car*, it uses a DC motor as a steering and driver. *Raspberry* will be connected serially with *Arduino* for *image processing* and drive interconnection[14],[15]. In this study, we developed a *self-driving car* in real time [16] library [14],[15]. It will be combined according to the purpose of making this AGV which makes it easier to detect the location of the AGV [16].

2. Research Methods

The flow of research conducted by the author is shown in Figure 1. Based on Figure 1, this research method includes a literature study where the theories used will

Accepted: 10-10-2022 | Received in revised: 28-03-2023 | Published: 01-06-2023

be implemented, Design and Manufacture of Prototypes, Results, and Discussion, as well as conclusions based on the results of tests that have been carried out Literature studies are conducted to identify the problem and find various solutions to solve the problem. After knowing the problem, the *self-driving car* that has been designed and implemented follows the solution to the problems found. the advantages of applying the *HSV* method to detect paths with the addition of pre-processing which will then be interconnected serially with the Arduino nano as a steering control on this *self-driving car* robot. Researchers make *self-driving* this way for transportation needs as the distribution of goods in the industry. This study also simulates a *self-driving car* for testing if errors are found.

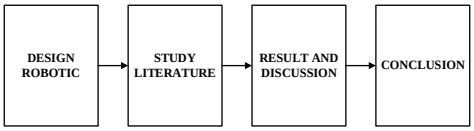

Figure 1. Research Flowchart

2.1 Planning Robot *Self-Driving*

The components for implementing a *self-driving car* include controllers and microcontrollers that are used, namely *Raspberry Pi 4 mini computers*, *Arduino Nano*, BTS7960, L298N drivers, potentiometers, and other supporting components. The L298N driver is used to control the DC motor at the front as steering on a *self-driving car*. The BTS7960 driver is used to control a DC motor as a forward or backward drive on a *self-driving car*. *Raspberry Pi 4* functions to run the main program whose job is to turn on the camera, process images captured by the camera, apply the *HSV* indicator, control the L298N driver, and finally translate the potentiometer connected to the steering into digital data which is then sent serially to the *Arduino nano*. The *Arduino Nano* functions to translate digital data received from the *Raspberry Pi 4* and align the program to operate the BTS7960 driver so that the self-driving car can move forward following the track.

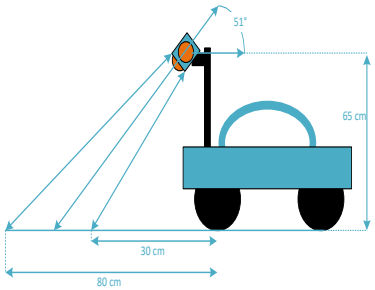

Figure 2. *Self-Driving Car* Camera Placement

In designing a *self-driving car* prototype using a power supply in the form of a 12V battery with a capacity of 12AH as a *Raspberry Pi* voltage source. The Raspberry Pi camera v2 is mounted on the front of the car using a handmade acrylic base. The Raspberry Pi camera holder is placed 510 to the floor with a height of 65 cm above the floor. This installation is intended so that the Raspberry Pi camera can capture the track pattern lines in 1 frame. Pictures of camera placement can be seen in Figure 2.

2.2 Path Detection Algorithm With *HSV* Method

In this study, the *HSV* filter was used to detect differences in the color of the lane markings and the color of the oad. This *HSV* is an indicator to determine the color of the outline of the track used. In this research, the level of accuracy and ability to recognize lanes using the *HSV* color filter mechanism is very suitable when implemented in this *self-driving car*. The composition of *HSV* includes *Hue*, *Saturation*, and *Value*. Where *Hue* is a collection of primary colors such as red, yellow, green, and others. As for *saturation*, it is purity or it can be called the strength of the color. Whereas this *value* is the brightness level with a value of 0% - 100% if the value is getting smaller then the color will look darker and vice versa. *HSV* values can also be obtained from the conversion of *RGB* values. A clear picture of *HSV* can be seen in Figure 3.

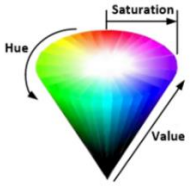

Figure 3. *HSV* Composition

RGB to *HSV* conversion can be obtained using equations (1), (2), and (3).

$$h = tan\left[\frac{3(g-b)}{(r-g)+(r-b)}\right] \quad (1)$$

$$s = 1 - \left[\frac{min(r,g,b)}{v}\right] \quad (2)$$

$$v = \frac{r+g+b}{3} \quad (3)$$

With this normalized value, the equation for converting the value from *RGB* to *HSV* value is equation (4), (5), and (6).

$$v = max\ (R,G,B) \quad (4)$$

$$S = \begin{cases} 0 & jika\ v > 0 \\ v - \frac{\min(R,G,B)}{v} & jika\ v = 0 \end{cases} \quad (5)$$

$$H = \begin{cases} 0 & jika\ s = 0 \\ \frac{60\ x\ (g-b)}{S\ x\ V} & jika\ v = R \\ 60\ x\left[2 + \frac{(b-r)}{S\ x\ V}\right] & jika\ v = G \\ 60\ x\left[4 + \frac{(r-g)}{S\ x\ V}\right] & jika\ v = B \end{cases} \quad (6)$$

To use the *HSV* filter, we have to calibrate each road

DOI: https://doi.org/10.29207/resti.v7i3.4579

condition and different lighting conditions, so we use the threshold function to set the *HSV* value which will be implemented in *self-driving cars*.

2.3. Pre-Processing HSV color filters

The use of pre-processing on the *HSV* color filter serves to perfect the marking lines that are read by this pre-processing using *morphology*. This *morphology* is an image or *image processing* technique using the basic principles of mathematical *morphology* and three basic operations, namely AND, OR, and NOT. The benefit of using the *morphology* process is to remove noise in the detected image. The following is an example of binary logic operations in the *morphology* process in Figure 4.

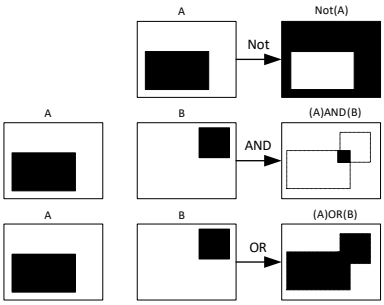

Figure 4. *Morphological* Logic

Then we use the *Dilation* process in the morphology technique which functions to add a pixel in the scope of the image that is read by the image, by placing one by one the center of the arrangement of elements for each background pixel. *Dilation* notation can be written in the equation (7).

$$g(x,y) = f(x,y) \oplus SE \tag{7}$$

For the original pixel that has the same value as the element arrangement, the neighborhood pixel value will change to the element arrangement or can be said to increase the pixel size so that the image being read becomes thick and can repair broken images when read, especially at the edges of objects. An example can be seen in Figure 5.

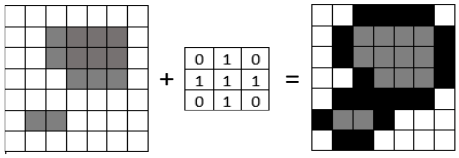

Figure 5. *Dilation* Process

Then the closing process in the *morphology* technique is to carry out erosion and dilation operations using the arrangement of elements in equation (8)

$$f(x,y) \circ SE = (f(x,y) \oplus SE) \ominus SE \tag{8}$$

This Closing method is used to repair pixels that retain their original shape by closing the perforated pixel holes. An example can be seen in Figure 6.

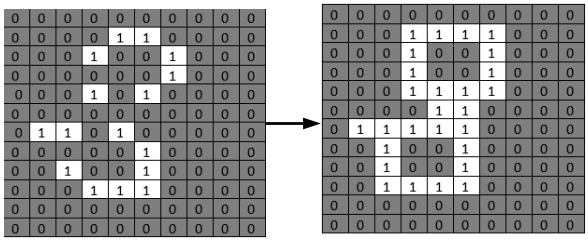

Figure 6. Closing Process

2.4. *Gaussian Blur*

The *Gaussian* filter function is to remove noise when detecting track pattern lines by changing the image to appear black pixels according to the amount of noise present. The blurred image obtained by convoluting the *Gaussian kernel* is shown in equation (9) and (10).

$$G(x,y) = F(x,y) * H(x,y) \tag{9}$$

$$(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{X^2+Y^2}{2\delta^2}} \tag{10}$$

In equations (9) and (10), G(x,y) represents the smoothed image, F(x,y) represents the input image, and H(x,y) represents the selected *Gaussian* filter.

2.5. *Raspberry Pi & Arduino Nano* Serial Communication

Serial communication is a method used to transfer data via a data cable or pin that connects two different or the same devices. In this study, Serial Communication was used between the *Arduino* microcontroller and the *Raspberry Pi mini computer*. Microcontroller is a device that is used to control several instruments at once, such as sensors, motors, servos, and so on. Meanwhile, a microcomputer is a device that works in the same way as a normal computer, except that it is smaller in size. This serial communication cannot be separated from the use of TX and RX pins. TX stands for Transmitter (sender), while RX stands for Receiver (receiver). In this *self-driving car*, the division of tasks in serial communication used raspberry pi as the sender and Arduino as a receiver. The Raspberry Pi's task as a sender is to send data obtained from the potentiometer attached to the steering which has been translated into digital data which is then sent to Arduino so that Arduino will process the digital data to drive the bts7960 driver which will move in harmony with the steering mounted on the connected device. different. Sending from Raspberry Pi to Arduino uses serial communication. So what you have to do is connect the TX pin on the raspberry pi to the RX pin on the Arduino, because the raspberry pi is the data sender and the Arduino is the receiver. Serial communication is very good to use because there is no delay in sending or receiving data. After all, it is directly connected serially and operates in real-time. Illustrate the serial connection between Raspberry and Arduino, it can be seen in Figure 7.

DOI: https://doi.org/10.29207/resti.v7i3.4579

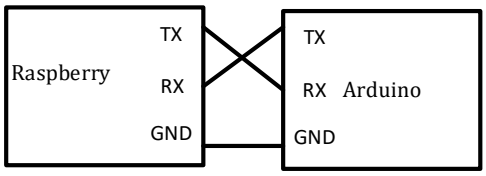

Figure 7. Serial Communication

2.6. *Ackermann* Model *Self-Driving Car*

The *Ackermann* system model on self-driving cars is designed so that no slip occurs between the ground and the wheels. Using the *Ackermann* method is very suitable to be implemented in this self-driving car because this car uses a movement system like conventional cars in general. The *Ackermann* model can be seen in Figure 8.

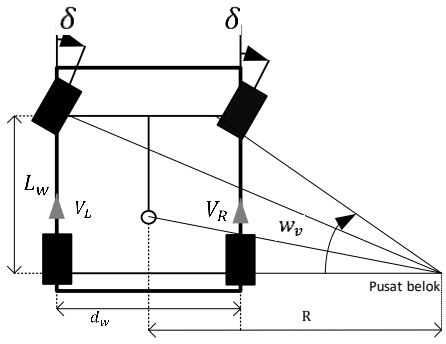

Figure 8. *Ackermann* Model

The model allows specifying the radius of rotation (R)

From the steering angle (δ) and rotation the angular velocity value must be adopted by each traction wheel $(\omega_{r\_R}, \omega_{r\_L})$. The linear speed of each driving wheel is expressed as the function of vehicle speed and radius of the curve in equations (10) and (11).

$$V_L = \omega_V \left(R + \frac{d_w}{2}\right) \quad (10)$$

$$V_R = \omega_v \left(R - \frac{d_w}{2}\right) \quad (11)$$

The radius of the curve depends on the wheelbase and rudder angle in equation (12).

$$\left(\boldsymbol{R} - \frac{\boldsymbol{L_w}}{\boldsymbol{tan\delta}}\right) \quad (12)$$

In equation (12) is the substitution of equations (10) and (11) we get the angular velocity at each driving wheel in equations (13) and (14).

$$\omega_{r\_L} = \frac{L_w + d_w/2.tan\delta}{L_w}\omega_v \quad (13)$$

$$\omega_{r\_R} = \frac{L_w - d_w/2.tan\delta}{L_w}\omega_v \quad (14)$$

2.7. *Self-Driving Car* Work Process

The initial step in the active camera flowchart is in the form of a video that records in real-time. Then the camera detects the track pattern. If the pattern is not detected, the program will not run and will return to the initial process. If a pattern is detected, then the process of applying the HSV indicator works by processing or filtering the required color then a Gaussian filter to smooth or remove noise in the path edge detection process so that the results are better. Then the next process when it has been successfully filtered is adding a supporting program that functions to keep the car's position in the middle of the track. The results of the filtering and supporting programs will be connected directly to the driving motor which will move according to the path that has been processed by the camera. To see the flowchart of the self-driving car work process can be seen in Figure 9.

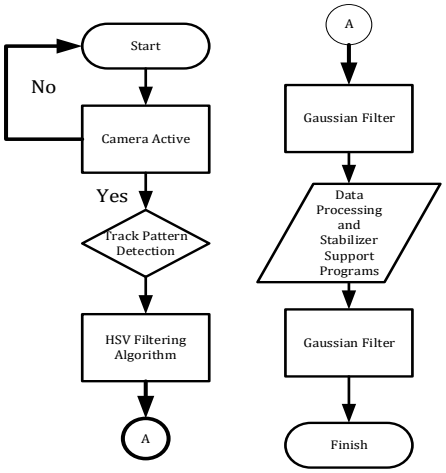

Figure 9. Self-Driving Cars Work Flowchart

3. Results and Discussions

The test results of this tool were carried out in a closed room using a Raspberry Pi 4B mini computer with a Raspberry v2 camera sensor as input that functions to read the path and for steering using a DC motor and this self-driving car moves with the rear wheels. The results of the self-driving car prototype can be seen in Figure 10.

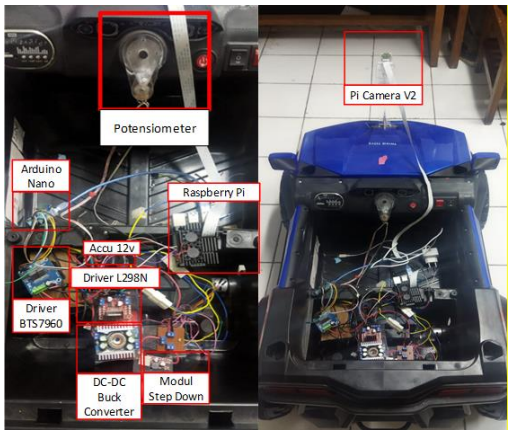

Figure 10. *Self-Driving Car* Prototype

DOI: https://doi.org/10.29207/resti.v7i3.4579

This self-driving car was tested on straight roads and turning roads as shown in the Figure 11.

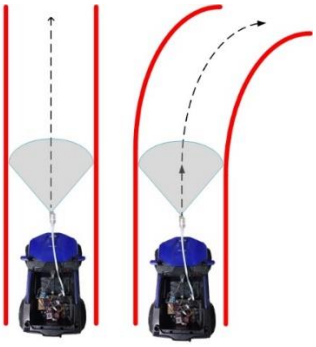

Figure 11. *Self-Driving Car* Trial Track

3.1. *HSV* Filter Trial Results

Testing this filter must first calibrate min to max and max to min. In this experiment, the min-to-max calibration is given a value of 0, and the max-to-min calibration test is given a max value of 179 for H, 255 for S, and V. By shifting the max or min values until a change occurs in the pattern object. Table 1 shows the H (hue) threshold test. Table 2 shows the S (saturation) threshold test. Table 3 shows the V (value) threshold test. The discussion is a basic explanation, of the relationships and generalizations shown by the results. The description answers the research question. If there are doubtful results then present them objectively.

Table 1. *Threshold Hue* Test

Calibration	H *min*	Hue *max*	White Color	Black Color
Min to Max	0	135	Detected	Not Detected
Max to *Min*	137	179	Not Detected	Detected

Table 2. *Threshold Saturation* Test

Calibration	S *min*	S *max*	White Color	Black Color
Min to Max	0	70	Detected	Not Detected
Max to *Min*	72	255	Not Detected	Detected

Table 3. *Threshold Value* Test

Calibration	V *min*	V *max*	White Color	Black Color
Min to Max	0	53	Detected	Not Detected
Max to *Min*	60	255	Not Detected	Detected

Tables 1 through 3 use the thresholding method where this process requires setting the appropriate value so that an accurate value is found. Then a trial is carried out by changing the value of min to max and max to min. The method is required to determine the black or white color of the path.

3.2. *Self-Driving Car* Functionality Testing

Functional testing is carried out to determine the effect of different light intensities. In this test, two conditions were tried, namely indoors and outdoors. The indoor test used TL lamp lighting while the outdoor test directly used lighting from the sunlight without any obstacles.

Table 4. Functionality Testing

Threshold Saturation	*Threshold Value*	*Indoor Condition*	*Outdoor Condition*
0-20	200-255	Undetected Path	Undetected Path
10-50	130-255	Detected Path	*Noise* Detected
70-255	53-106	Detected Path	Detected Path

Table 4 shows that setting a threshold value under different location conditions can affect the perception of the color filter process being performed. But if the threshold saturation value is 70-255 and the threshold value is 53-106 then the system can adapt to indoor and outdoor lighting conditions.

In this study, the Hue value was Hmin = 135 and Hmax = 179, the saturation value was Smin = 70 and Smax = 255, and the Value value was Vmin = 53 and Vmax = 106 so that the tool can detect the specified path. In this trial process using the thresholding method to distinguish between black and white. The determination of the HSV value can be seen in Figure 12.

(a) (b) (c)

Figure 12. *Self-Driving Car* Trial Path

The output is the final result window which has gone through a thresholding process and a gaussian filter with a closing technique that produces good color filtering without noise. After going through the process of determining the HSV value, the final result will be seen in Figure 13.

Figure 13. Closing Process Result

DOI: https://doi.org/10.29207/resti.v7i3.4579

At this stage, a self-driving car trial was carried out using the HSV algorithm on the trajectory. In this process, the car can run automatically following the track in real time. The movement of the car is determined from the tracking process using the (x, y) axis which utilizes the blue center point to set the conditions for the car's movement. If the angle of the red line is greater to the right then the car will turn right. If the angle of the red line is greater to the left then the car will turn left. If the angles are the same, the car will go straight. The tracking results and experiments can be seen in Figure 14 and Figure 15.

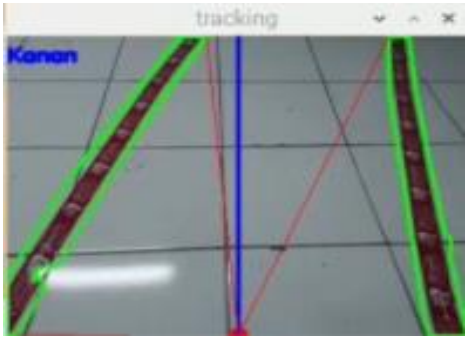

Figure 14. *Self-Driving Car* Moving Test

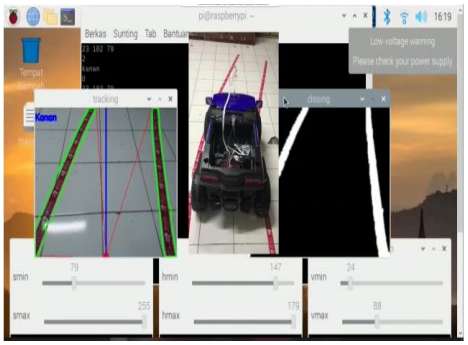

Figure 15. Overall Test

The test results above can run according to the algorithm whether operated in an open or closed room. The algorithm that has been made in this test is that this self-driving car can detect the path correctly and for the motion mechanism to be properly connected between the microcontroller and the Raspberry Pi 4 mini-computer.

4. Conclusion

The path detection algorithm with the HSV color filter method can work well when detecting a path when a self-driving car moves along a straight path or a turning lane. To test the road movement of the self driving car, it is carried out in a closed room with sufficient lighting. The results of reading the path and movement of the self driving car will be immediately displayed on the monitor screen of this mini computer. This algorithm has been developed and researched and has a reading and movement accuracy rate of 90% when compared to an automated guided vehicle that uses an infrared sensor. The self driving car is more reliable because the self driving car is not disturbed by light. And when compared to the CNN method, the method used by researchers is more effective and runs immediately in real time and no path samples are required for its operation. In the future, this research will be used for industrial needs or automotive needs which will then be applied in Indonesia

Acknowledgment

Thank you to the Soegijapranata Catholic University for providing a place to conduct this research.

References

[1] M. Daily, H. R. L. Laboratories, and S. Medasani, “Self-Driving Cars.”
[2] T. A. S. Nielsen and S. Haustein, “On sceptics and enthusiasts: What are the expectations towards self-driving cars?,” Transp. Policy, vol. 66, no. April 2017, pp. 49–55, 2018, doi: 10.1016/j.tranpol.2018.03.004.
[3] S. Royo and M. Ballesta-Garcia, “An overview of lidar imaging systems for autonomous vehicles,” Appl. Sci., vol. 9, no. 19, 2019, doi: 10.3390/app9194093.
[4] Y. Zein, M. Darwiche, and O. Mokhiamar, “GPS tracking system for autonomous vehicles,” Alexandria Eng. J., vol. 57, no. 4, pp. 3127–3137, 2018, doi: 10.1016/j.aej.2017.12.002.
[5] F. B. Setiawan, O. J. Aldo Wijaya, L. H. Pratomo, and S. Riyadi, “Sistem Navigasi Automated Guided Vehicle Berbasis Computer Vision dan Implementasi pada Raspberry Pi,” J. Rekayasa Elektr., vol. 17, no. 1, pp. 7–14, 2021, doi: 10.17529/jre.v17i1.18087.
[6] Florentinus Budi Setiawan, F. A. Kurnianingsih, Slamet Riyadi, and Leonardus Heru Pratomo, “Pattern Recognition untuk Deteksi Posisi pada AGV Berbasis Raspberry Pi,” J. Nas. Tek. Elektro dan Teknol. Inf., vol. 10, no. 1, pp. 49–56, 2021, doi: 10.22146/jnteti.v10i1.738.
[7] J. Naranjo-Torres, M. Mora, R. Hernández-García, R. J. Barrientos, C. Fredes, and A. Valenzuela, “A review of convolutional neural network applied to fruit image processing,” Appl. Sci., vol. 10, no. 10, 2020, doi: 10.3390/app10103443.
[8] S. O. Ali Chishti, S. Riaz, M. Bilal Zaib, and M. Nauman, “Self-Driving Cars Using CNN and Q-Learning,” Proc. 21st Int. Multi Top. Conf. INMIC 2018, 2018, doi: 10.1109/INMIC.2018.8595684.
[9] T. D. Do, M. T. Duong, Q. V. Dang, and M. H. Le, “Real-Time Self-Driving Car Navigation Using Deep Neural Network,” Proc. 2018 4th Int. Conf. Green Technol. Sustain. Dev. GTSD 2018, pp. 7–12, 2018, doi: 10.1109/GTSD.2018.8595590.
[10] C. Goyal, P. Lokeshwara Reddy, and P. Amalyal, “WITHDRAWN: Design & implementation of real time autonomous car by using image processing & IoT,” Mater. Today Proc., no. xxxx, 2020, doi: 10.1016/j.matpr.2020.08.060.
[11] B. C. Z. Blaga, M. A. Deac, R. W. Y. Al-Doori, M. Negru, and R. Danescu, “Miniature autonomous vehicle development on raspberry Pi,” Proc. - 2018 IEEE 14th Int. Conf. Intell. Comput. Commun. Process. ICCP 2018, pp. 229–236, 2018, doi: 10.1109/ICCP.2018.8516589.
[12] P. Zhang, T. Zhuo, W. Huang, K. Chen, and M. Kankanhalli, “Online object tracking based on CNN with spatial-temporal saliency guided sampling,” Neurocomputing, vol. 257, no. 2017, pp. 115–127, 2017, doi: 10.1016/j.neucom.2016.10.073.
[13] A. Merino, L. Puigví, L. Boldú, S. Alférez, and J. Rodellar, “Optimizing morphology through blood cell image analysis,” Int. J. Lab. Hematol., vol. 40, no. March, pp. 54–61, 2018, doi:

DOI: https://doi.org/10.29207/resti.v7i3.4579

10.1111/ijlh.12832.

[14] J. R. Cortes Leon, R. F. Martínez-Gonzalez, A. M. Medina, and L. A. Peralta-Pelaez, "Raspberry PI and Arduino UNO Working Together as a Basic Meteorological Station," Int. J. Comput. Sci. Inf. Technol., vol. 9, no. 5, pp. 97–104, 2017, doi: 10.5121/ijcsit.2017.9508.

[15] A. Aqthobilrobbany, A. N. Handayani, D. Lestari, Muladi, R. A. Asmara, and O. Fukuda, "HSV Based Robot Boat Navigation System," CENIM 2020 - Proceeding Int. Conf. Comput. Eng. Network, Intell. Multimed. 2020, pp. 269–273, 2020, doi: 10.1109/CENIM51130.2020.9297915.

[16] Z. Sun, "Vision Based Lane Detection for Self-Driving Car," Proc. 2020 IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. AEECA 2020, pp. 635–638, 2020, doi: 10.1109/AEECA49918.2020.9213624.

DOI: https://doi.org/10.29207/resti.v7i3.4579