

CHAPTER 4 – RESULTS AND DISCUSSIONS

This chapter provides an in-depth look of the results and discussions that follows by the well-organized stages of development, calibration, and testing, in accordance with the V-model methodology. These discussions build upon the design concepts that were set forth in the previous chapter. The V-model, a cornerstone in the system development and engineering methodologies, has played a crucial role in directing the systematic approach to research.

4.1 Hardware and Software Development

The Hardware and Software development stage involves the parallel development of hardware and software components, following the specified requirements and system architecture. Hardware Development involves operations such as designing circuits, constructing prototypes, and conducting hardware testing. Software Development involves the tasks of coding, conducting unit testing, and ensuring that the product fulfills the required requirements. Both developments have the objective of preparing by utilizing component development, unit analysis, and integration build.

4.1.1 Component Development

Concerning hardware, this requires the detail workable specification on hardware. The workable specification is needed to start producing a prototype for the design so that the design specification is well-drafted out [31]. The schematic diagram works as a workable specification in this system, which is outlining all electrical connections that includes the arrangement of cables, components, and connecting points. Figure 4.1 shows the schematic diagram of the system created using the EasyEDA software.

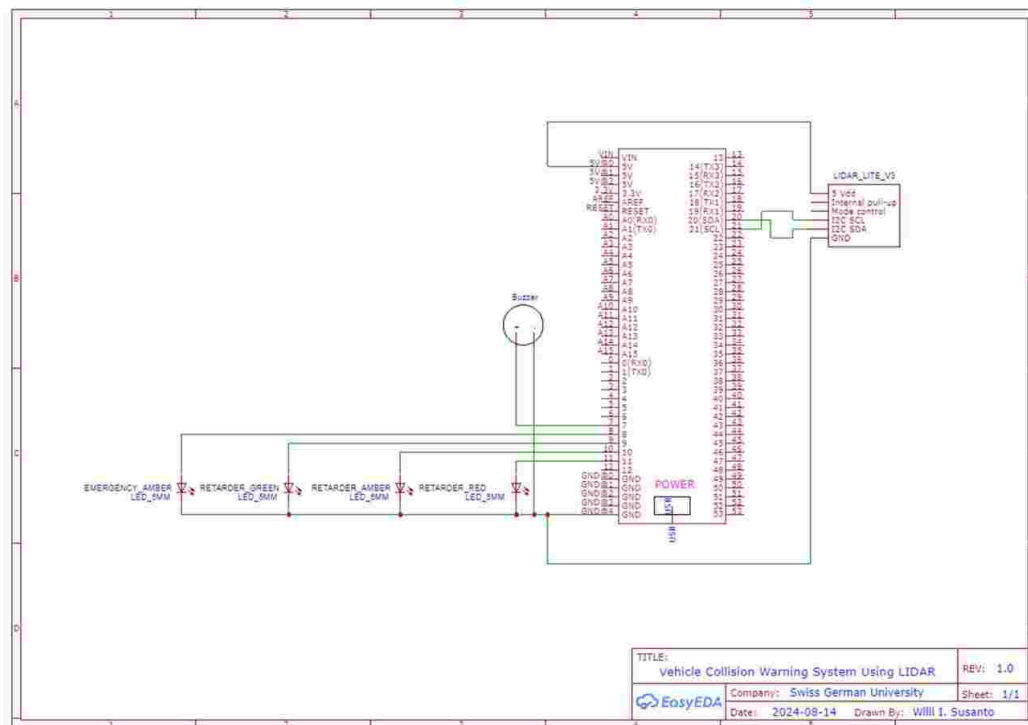


Figure 4.1 Schematic Diagram of Vehicle Collision Warning System

The wiring diagram serves as a crucial plan that directs the assembly and integration of the hardware components, guaranteeing that all physical connections are organized in a manner that supports the functionality and dependability of the system. The hardware assembly can be executed and understood by referring to the blueprint guidelines, shows in Figure 4.2.

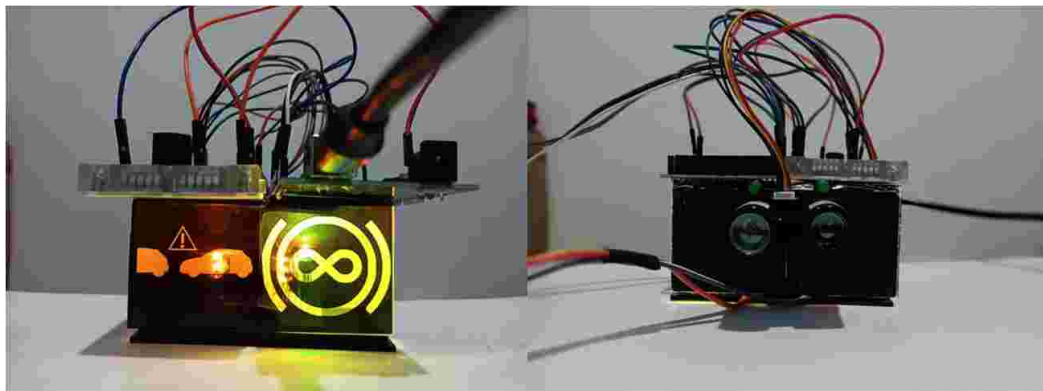


Figure 4.2 Vehicle Collision Warning System Assembly

Concurrently, software development involves creating a calibration program that is essential for correlating the hardware's output with the expected results based on specified standards. This process guarantees the accuracy and performance of the system. The calibration program is divided according to a single system of the sensor and actuators. Sensor calibration is the process of ensuring the accuracy of the Garmin LIDAR Lite v3 through the comparison of the measurements generated through a manual distance check using a roll meter instrument with the distance result presented in the Arduino serial monitor. The calibration of actuators, particularly the buzzer as an audible signal and LEDs as a visual signal, is an essential process to guarantee their consistent and accurate functioning. This methodology involves a systematic procedure of iteratively activating and deactivating the actuators multiple times. By engaging in this iterative process, it becomes feasible to evaluate and adjust the performance characteristics of the actuators.

4.1.2 Unit Analysis and Test

In the unit analysis and test phase, every individual system component, including sensors and actuators, gets thorough testing to verify that it complies to design standards and functions accurately before starting integration. Calibration at this stage involves adjustment the responses of the components to guarantee accuracy and consistency in their measurements and operations. The sensor would go through testing to guarantee that it accurately observes environmental variables in accordance with established standards, while an actuator would be tested to validate that its response to control signals exactly corresponds to the planned action. The calibration procedure for the vehicle collision warning system involves verifying the actuators (LED and Buzzer) and aligning the precision of the LIDAR sensor.

1. Actuators Calibration

The initial stage of the calibration process for a vehicle collision warning system involves calibrating the actuators, specifically the LED and buzzer. The method of calibration is using a specific Arduino program. The flowchart of the actuators calibration is process specified in the Figure 4.3.

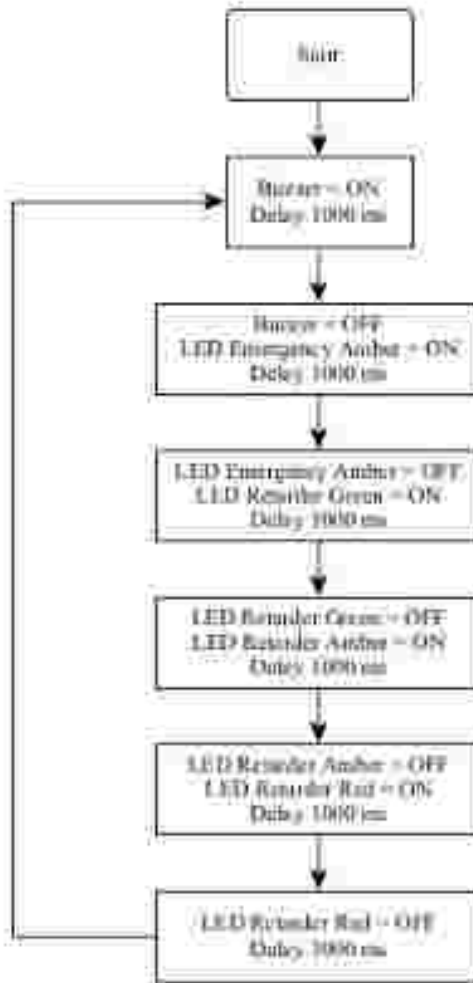


Figure 4.3 Actuator Calibration Flow Procedure

The result of actuator calibration is presented in Table 4.1, as indicated by the flow procedure.

Table 4.1 Calibration Test for Buzzer and LED

No.	Actuator		Step 1	Step 2	Step 3	Step 4	Step 5
1	Buzzer		ON	OFF	OFF	OFF	OFF
2	LED Emergency Brake		OFF	ON	OFF	OFF	OFF
3	LED Retarder	Green	OFF	OFF	ON	OFF	OFF
4	LED Retarder	Amber	OFF	OFF	OFF	ON	OFF
5	LED Retarder	Red	OFF	OFF	OFF	OFF	ON

2. LIDAR Sensor Calibration

The second phase is calibrating the LIDAR sensor, a crucial aspect for the accurate operation of the collision warning system. The distance sensor utilizes a roll meter to accurately measure the distance, which is then compared to the recorded result in the serial monitor. The outcome of the calibration is presented in Table 4.2.

Table 4.2 Calibration Test for Garmin LIDAR Sensor v3

No.	Actual Range	System Read	Standard	Result	Remark
1	55 cm	56 ~ 59 cm	± 5 cm	Max +4 cm	Within Spec
2	90 cm	93 ~ 95 cm	± 5 cm	Max +5 cm	Within Spec
3	280 cm	284 ~ 285 cm	± 5 cm	Max +5 cm	Within Spec
4	565 cm	566 ~ 568 cm	± 5 cm	Max +3 cm	Within Spec
5	1520 cm	1522 ~ 1524 cm	± 5 cm	Max +4 cm	Within Spec

According to the calibration result in Table 4.1, it is evident that the sensor and actuators are functioning effectively. The LIDAR sensor generates an output with a variance of ± 5 cm, which matches the specifications of the sensor. Meanwhile, the actuators are functioning properly according to the commands of the calibration program. Thoroughly calibrating and testing each unit separately allows for the early identification and resolution of any faults. This is crucial for ensuring the reliability of complex systems, as the performance of individual components can greatly affect the overall behavior of the system.

4.1.3 Integration Build

System development regularly takes place at a per-feature or per-component level. In order to achieve a comprehensive solution, it is necessary to merge each individual subsystem into an integrated compiled design, which is then tested as an integrated entity. This stage involves the integration of multiple hardware and software components to form a comprehensive system. It guarantees the coordinated functioning of all components as intended. In order to integrate the system, there are three steps involved in the decision-making process: the kinematics algorithm, the perception

algorithm, and the full system that combines both kinematics and perception algorithms.

4.1.3.1 Kinematics Algorithm

The Kinematics algorithm approaches the specified target value as either true or false in a strict sense, following 3-second rule guidance from education campaign from the Toll Road Business Entities, The Ministry of Public Works and Public Housing. With considering the calculation process while LIDAR is taking distance value, the Arduino program have been set with 3 condition if, shown in the Table 4.3.

Table 4.3 Kinematics Algorithm System Decision

No.	Time-to-Collision (sec)	Interval	Actuator	
			LED Emergency Brake	Buzzer
1.	> 2	1 sec	Off	Off
2.	$1 > \text{TTC} \leq 2$	1 sec	Amber	Off
3.	≤ 2	1 sec	Amber	On

4.1.3.2 Perception Algorithm

The perception algorithm relies on Fuzzy Logic to calculate the appropriate brake action with not a simply true or false target. As more and more situations require decisions that cannot be answered with a simple yes or no, using fuzzy logic to support decisions becomes more necessary [32]. The perception algorithm relies on Fuzzy Logic to calculate the appropriate brake action based on the distance to the next vehicle and the relative speed. The construction of Fuzzy Logic will be carried out using Matlab, specifically by utilizing the 'fuzzyLogicDesigner' function with the Mamdani type. The Mamdani model is well-suited for a driver assistance system designed to mitigate collisions by providing brake suggestions based on the distance to the next vehicle and relative speed. This is because the Mamdani method is effective at dealing with uncertainty and complicated associations between variables. One advantage of fuzzy logic in order to formalize human reasoning is that the rules are set in natural language [33]. The Mamdani technique is known for its intuitive rule-based structure, which replicates human decision-making by utilizing normal language terminology. This

characteristic makes it understandable and simple to build into systems that required human-like thinking, such as driver assistance systems.

The Mamdani fuzzy inference system works by converting crisp inputs into fuzzy sets, utilizing these sets in a series of IF-THEN rules to produce fuzzy outputs, and afterward integrating these outputs into a unified set that is later converted back into a crisp, actionable final result through defuzzification. Figure 4.4 displays the overview diagram of the Mamdani fuzzy logic system.

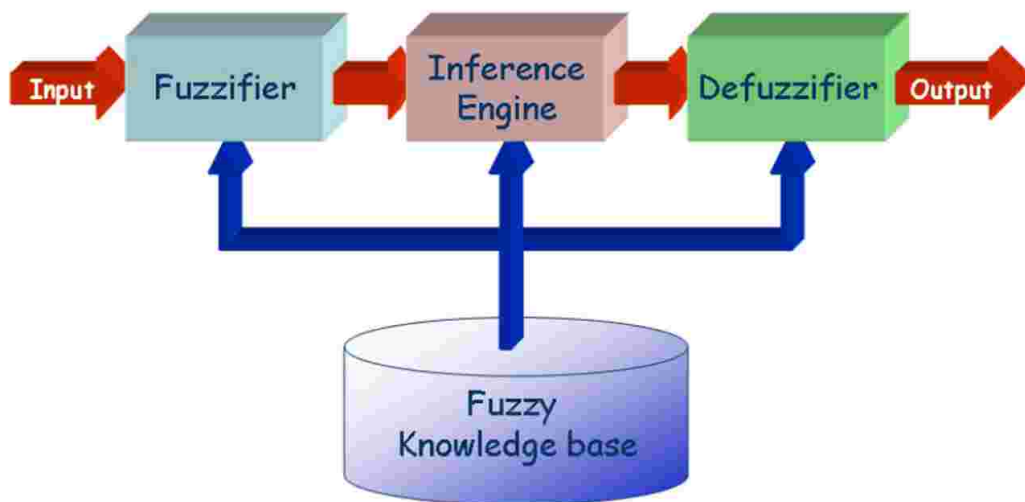


Figure 4.4 Overview Diagram of a Fuzzy System

In order to create a fuzzy logic system that provides suggestions for brake decision according to the distance to the next vehicle and the relative speed, here is a detailed and step-by-step approach to developing the fuzzy system:

1. Step 1: Define Inputs and Outputs

The development of the Fuzzy system starts with the identification of the important variables that have an effect on the system. The inputs for this model are the Distance to the Next Vehicle and the Relative Speed between the car and the car ahead. The output provides the Brake Suggestion, which indicates the recommended form of braking action. Each variable should have a crisp value

within a specific range. Table 4.4 provides detailed information about the specific crisp values for each variable.

Table 4.4 Fuzzy System Input-Output

Variable	Specification	Range
Distance to Next Vehicle	Max 4000 cm	0 – 4000
Relative Speed	Max 4000 cm/s	0 – 4000
Brake Suggestion	Max 100% pressing brake	0 – 100

Using the Fuzzy Logic Toolbox in MATLAB, generate a new Fuzzy Logic System and specify the inputs (Distance to Next Vehicle and Relative Speed) and output (Brake Suggestion) through the assigned ranges.

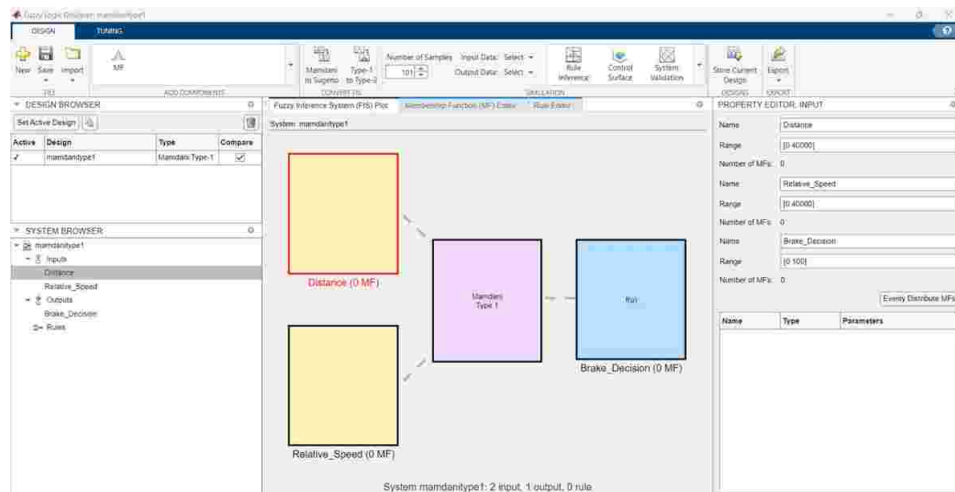


Figure 4.5 Fuzzy System Input-Output (Matlab GUI)

2. Step 2: Fuzzification

Subsequently, the fuzzification process involves the conversion of precise, numerical inputs into fuzzy inputs. This is accomplished by assigning each input variable to a collection of linguistic terms, each of which is represented by a fuzzy membership function (MF). Linguistic words are represented using geometric shapes such as triangles or trapezoids. Fuzzification is a fundamental first process in fuzzy logic systems, enabling them to manage complex and

unclear data by utilizing a reasoning approach that emulates human decision-making. Table 4.5 displays the membership function (MF) of the variable.

Table 4.5 Fuzzy System Membership Function

Variable	Membership Function	Type	Parameters
Distance (to Next Vehicle)	Close	Trapezoidal	[0 0 500 1000]
	Fair	Triangular	[500 1500 2500]
	Far	Trapezoidal	[2000 2500 4000 4000]
Relative Speed	Slow	Trapezoidal	[0 0 500 1000]
	Medium	Triangular	[500 1500 2500]
	Fast	Trapezoidal	[2000 2500 4000 4000]
Brake Suggestion	Safe	Triangular	[0 15 30]
	Light	Triangular	[20 35 50]
	Moderate	Triangular	[40 55 70]
	Significant	Triangular	[60 80 100]

The Membership Function type has been defined utilizing Trapezoidal and Triangular shapes, not only due to their popularity but also because of their excellent performance. According to Nasution's research in 2011, the type is simple and give a good controller performance and easy to handle [34]. According to the research, both the triangular and trapezoidal membership function types have no overshoot, have a fast speed response, and good steady state accuracy.

Afterwards that, the crisp value was developed in compliance with the established criteria, and it is presently being integrated into the fuzzy system utilized in Matlab. A determination is reached for the distribution of values based on the estimated value. Figure 4.6 shows the input and output of the fuzzy system built in Matlab.

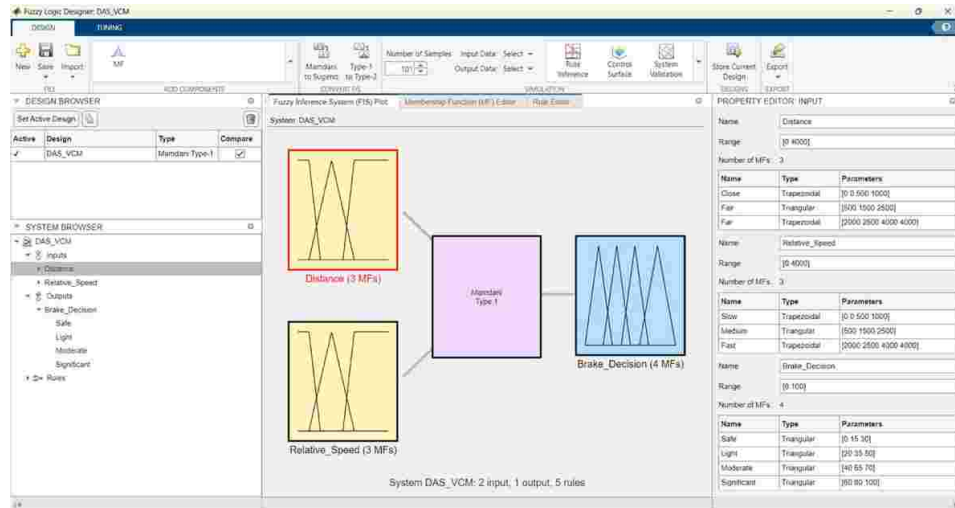


Figure 4.6 Fuzzy System Membership Function (Matlab GUI)

3. Step 3: Define Rule

When working with fuzzy logic, it is essential to determine the specific rule utilized by the inference engine. The rules are comprised of a set of IF-THEN statements that form a logic connection between the fuzzy inputs and the fuzzy outputs. The Rule Editor in Fuzzy GUI allows for the input of rules using the Add All Possible Rules function. Since that there are two variables, each with three membership functions, the total number of feasible rules is nine. Upon considering all alternative rules, the decision is made based on the basic logic as the first prediction. The following are the guidelines for constructing the Perception-based system:

System: DAS_VCM			
Add All Possible Rules Clear All Rules			
	Rule	Weight	Name
1	If Distance is Close and Relative_Speed is Slow then Brake is Moderate	1	rule1
2	If Distance is Fair and Relative_Speed is Slow then Brake is Light	1	rule2
3	If Distance is Far and Relative_Speed is Slow then Brake is Safe	1	rule3
4	If Distance is Close and Relative_Speed is Medium then Brake is Significant	1	rule4
5	If Distance is Fair and Relative_Speed is Medium then Brake is Moderate	1	rule5
6	If Distance is Far and Relative_Speed is Medium then Brake is Light	1	rule6
7	If Distance is Close and Relative_Speed is Fast then Brake is Significant	1	rule7
8	If Distance is Fair and Relative_Speed is Fast then Brake is Significant	1	rule8
9	If Distance is Far and Relative_Speed is Fast then Brake is Moderate	1	rule9

Figure 4.7 Fuzzy System Rules Mapping

4. Step 4: Fuzzy Inference Engine

Matlab fuzzy logic designer offers two types of inference engines: Mamdani and Sugeno. After the consideration on the previous discussion, it has been determined that the Mamdani type will be applied in the collision warning system because it is more compatible with the collision warning system.

The Mamdani fuzzy inference system operates by transform crisp inputs into fuzzy sets, using these sets in a sequence of IF-THEN rules to generate fuzzy outputs, and subsequently merging these outputs into a unified set that is subsequently converted back into a crisp, practical final output through defuzzification. Previous research experiments have confirmed that the Mamdani-type Fuzzy is the most suitable for linear systems, it is widely used in particular for decision support application [35].

It is necessary to configure Mamdani as a type of fuzzy system in the Matlab fuzzy logic designer graphical user interface (GUI). After that, the FIS plot will display the type on the block diagram according to the rules.

5. Step 5: Defuzzification

Defuzzification is the process by which fuzzy outputs are converted into distinct and precise results. The collision warning fuzzy system will employ the centroid method. As previously stated in the research, centroid defuzzification is the most widely used technique because, when it is used, the defuzzified values tend to move smoothly around the output fuzzy region [36].

In order to calculate the output using the centroid technique in a fuzzy logic system, it is necessary to determine the balance point of the output region of the fuzzy set. This is similar to calculating the centroid of a physical object. The centroid is determined by computing the weighted mean of all potential outputs, where the weights correspond to the membership values of these outputs in the fuzzy set.

In centroid method, the fuzzified value, $d(C)$, is defined as the value within the range of variable z for which the area under the graph of membership function C is divided into two equal subareas. For the discrete case, in which C is defined on finite universal set $[z_1, z_2, \dots, z_n]$, the formula is shown on Equation 4.1 [37].

$$d_{CA}(C) = \frac{\sum_{k=1}^n C(z_k)z_k}{\sum_{k=1}^n C(z_k)} \quad (4.3)$$

After completing the development of the fuzzy system from Step 1 to Step 5, the attention now turns to the final phase with testing and optimization. These are the final phases of fuzzy development.

1. Final Step 1: Implementation and Testing

Implementation and testing indicate to the operational use of the system, which include developing a graphical user interface (GUI) using Matlab's Rule Inference. This graphical user interface enables the dynamic adjustment of input values and provides real-time visualization of braking suggestions, effectively imitating real-world interactions. Figure 4.8 shows the operation of the Matlab's Rule Inference.

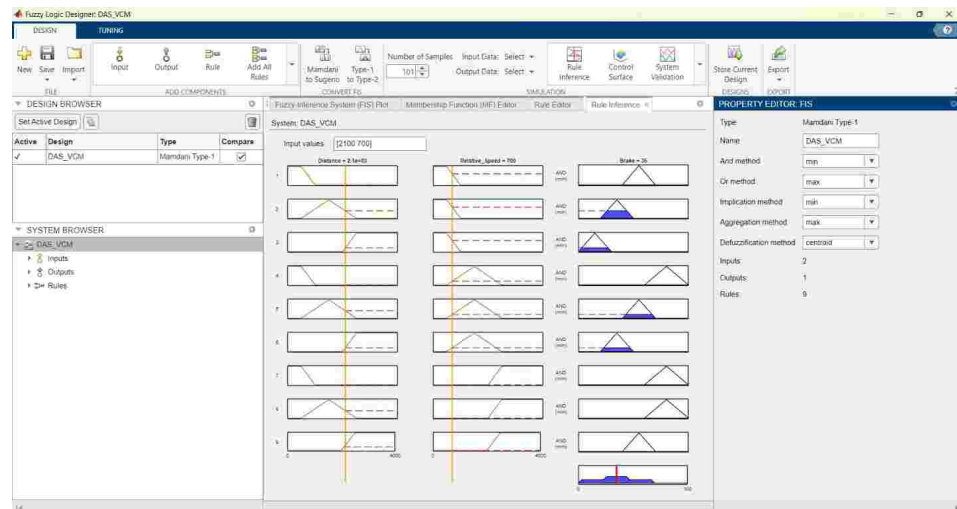


Figure 4.8 Matlab's Fuzzy System Rule Inference

From the Figure 4.8 above, it can be seen that the crisp input for distance is 2100 and the relative speed is 700, it means that the distance value is 2100 cm or equal to 2.1 m, and the relative speed is 700 cm/s or 0.7 m/s. The result from the fuzzy system is 35, which mean it needs 35% of brake. To verified that the system is working well, lets take a manual calculation of the fuzzy decision by step by step.

a. Fuzzification

Take the crisp input $x1$ and $y1$, then Determine the degree to which these inputs belong to each of the appropriate fuzzy sets.

Crisp input:

$$x1 = 2100; y1 = 700$$

Membership function:

$D1$ = Close; $D2$ = Moderate; $D3$ = Far

$V1$ = Slow; $V2$ = Medium; $V3$ = Fast

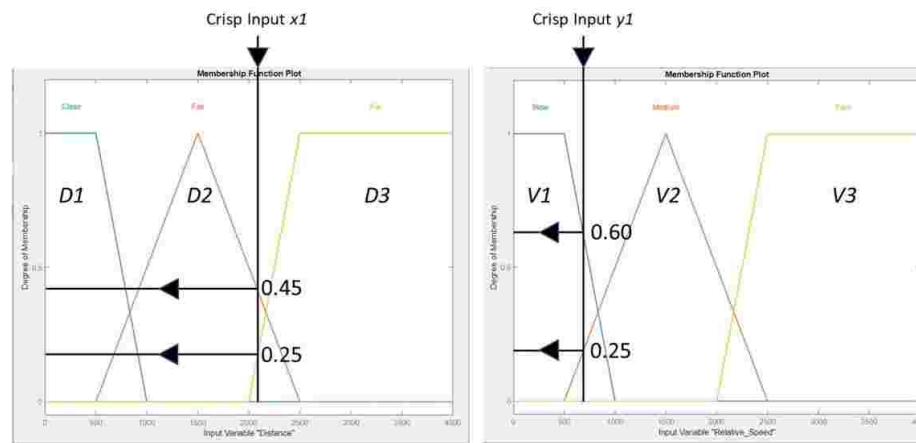


Figure 4.9 Fuzzification of the First Implementation System

From the Figure 4.9, it can be identified the estimation of the Degree of Membership (DOM), with the result as below:

$$DOM_{x=D2}=0.45$$

$$DOM_{x=D3}=0.25$$

$$DOM_{y=V1}=0.60$$

$$DOM_{y=V2}=0.25$$

b. Rule Evaluation

The following step involves calculating the Degree of Membership (DOM) using the fuzzified inputs for rule evaluation. This phase involves checking the intersection of the inputs in the relevant rules. The fuzzy operator used to derive a single DOM representation of the rule is the AND fuzzy operation, which involves the intersection of fuzzy sets. This operation is defined by the formula presented in Equation 4.2.

$$DOM_{A \cup B}(x) = \min[DOM_A(x), DOM_B(x)] \quad (4.2)$$

The specific details shown in Figure 4.10, which is a capture from the Rule Inference tab in Matlab's Fuzzy Logic Designer.

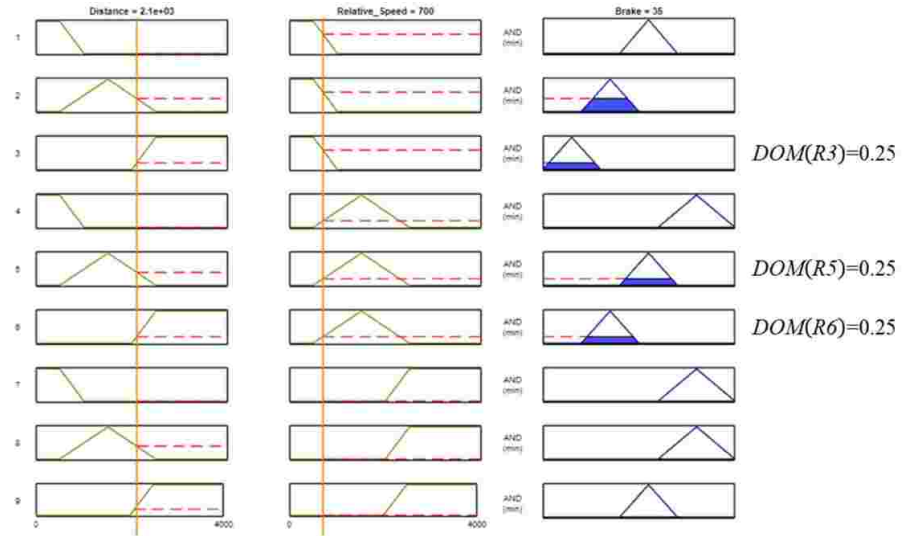


Figure 4.10 Rule Inference Setup for the Rule Evaluation

Figure 4.10 illustrates that the intersection inputs are determined by Rule 2, Rule 3, Rule 5, and Rule 6. Following that, the result of the DOM computed in the following calculation.

$$DOM_{D2 \cup V1}(R2) = \min[0.45, 0.60] = 0.45$$

$$DOM_{D3 \cup V1}(R3) = \min[0.25, 0.60] = 0.25$$

$$DOM_{D2 \cup V2}(R5) = \min[0.45, 0.25] = 0.25$$

$$DOM_{D3 \cup V2}(R6) = \min[0.25, 0.25] = 0.25$$

c. Aggregation of the Rule Outputs

In the preceding stage of rules evaluation, the Degree of Membership (DOM) of the outputs generated by the relevant rules has been determined. During the aggregation process, the outputs of all rules will be unified. The process will aggregate the membership functions of all rule consequents that have been previously clipped or scaled, resulting in a unified fuzzy set $\sum DOM$. The result of the fuzzy set shown in Figure 4.11.

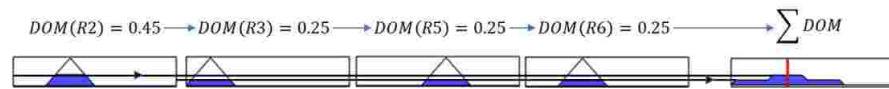


Figure 4.11 Aggregation of the Outputs of Fuzzy System

Based on the aggregate result, there are only three Membership of Function (MF) that intersect in the fuzzy set. These MFs are Safe, Light, and Moderate. The range of each variable of the three membership functions is determined by calculating the mean of the parameter.

$$z_{1(safe)} = \frac{0+15+30}{3} = 15$$

$$z_{2(light)} = \frac{20+35+50}{3} = 35$$

$$z_{3(moderate)} = \frac{40+55+70}{3} = 55$$

d. Defuzzification

The defuzzification step is where the Fuzzy's final calculation is performed. Fuzziness facilitates the evaluation of rules, nevertheless, the final result of a fuzzy system must be a crisp number. The defuzzification procedure takes the aggregated output fuzzy set as input and produces a single numerical output. The defuzzification process in this computation involves employing the centroid approach, which utilizes Equation 4.1. The crisp number of the output is calculated as given in the calculation below.

$$d_{CA}(C) = \frac{\sum_{k=1}^n C(z_k)z_k}{\sum_{k=1}^n C(z_k)} \quad (4.4)$$

where $C(z_k)$ is the degree contributions of each membership z_k , then

$$DOM(z_1) = DOM(R3) = 0.25$$

$$DOM(z_2) = DOM(R2) = 0.45$$

$$DOM(z_3) = DOM(R5) = 0.25$$

$$d_{CA}(C) = \frac{DOM(z_1)z_1 + DOM(z_2)z_2 + DOM(z_3)z_3}{DOM(z_1) + DOM(z_2) + DOM(z_3)}$$

$$d_{CA}(C) = \frac{0.25(15) + 0.45(35) + 0.25(55)}{0.25 + 0.45 + 0.25}$$

$$d_{CA}(C) = \frac{3.75 + 15.75 + 13.75}{0.95} = \frac{33.25}{0.95} = 35$$

The fuzzy decision result output, calculated manually, is $d_{CA}(C) = 35$. It generates the identical outcome as the system has been completed. This result indicates that the Matlab system works well according to the basics theory. The next phase of the fuzzy development can be executed using Matlab, as it operates proficiently. The next step will involve testing the implementation under various conditions, followed by evaluating the results for the purpose of optimizing and fine-tuning.

2. Final Step 2: Optimization and Fine-Tuning

The last stage of the fuzzy development process corresponds to optimizing and fine-tuning the fuzzy system. The process involves thorough sampling evaluation within many scenarios to guarantee that the system performs as expected. According to a previous study published in the IEEE journal regarding the tuning of Fuzzy Logic Controllers (FLCs), the tuning process of FLCs have no systematic procedure, requiring a trial-and-error approach involving adjustments to the fuzzy rules and the mapping of the membership functions until satisfactory results are achieved [38].

The strategy will also be applied to the tuning process in this research. The fine-tuning procedure involves the utilization of a trial-and-error approach to improve the fuzzy logic rules and membership functions until satisfactory results are achieved. The iterative process is crucial for improving the accuracy and user-friendliness of the system, guaranteeing that the fuzzy logic system not only fulfills but outperforms the realistic requirements of real-world driving situations. This step concludes the development process, resulting in a reliable and trusted system that is prepared for operation. These are the sequential process of trial and fine-tune that the researcher went through to optimize the fuzzy system.

a. Trial 1: Rules Adjustment

This stage involves reviewing a set of existing rules that define the system's decision-making process. Through the analysis of system implementation in different scenarios, modifications are implemented to the existing rules in order to accurately reflect for the unique requirements of real-world driving conditions. During this step, the main focus is to enhance the "THEN" part of the fuzzy rules, which define the system's output or braking decision. The modification involves readjusting these rules to ensure that outputs are configured to more extreme values, such as significant braking action or a safe condition. The outcome of the rules modification is shown in Figure 4.12.

System: DAS_VCM_2			
Add All Possible Rules Clear All Rules			
	Rule	Weight	Name
1	If Distance is Close and Relative_Speed is Slow then Brake is Moderate	1	rule1
2	If Distance is Fair and Relative_Speed is Slow then Brake is Safe	1	rule2
3	If Distance is Far and Relative_Speed is Slow then Brake is Safe	1	rule3
4	If Distance is Close and Relative_Speed is Medium then Brake is Significant	1	rule4
5	If Distance is Fair and Relative_Speed is Medium then Brake is Significant	1	rule5
6	If Distance is Far and Relative_Speed is Medium then Brake is Light	1	rule6
7	If Distance is Close and Relative_Speed is Fast then Brake is Significant	1	rule7
8	If Distance is Fair and Relative_Speed is Fast then Brake is Significant	1	rule8
9	If Distance is Far and Relative_Speed is Fast then Brake is Significant	1	rule9

Figure 4.12 Fuzzy System Rules Mapping Adjustment

Three rules, especially rule2, rule5, and rule9, have been adjusted as a result of fine-tuning. As explained earlier, the output of the rules has been adjusted to reflect extreme values. Specifically, rule2 changes from Light to Safe, while rule5 and rule9 are adjusted from Moderate to Significant. The objective of generating these extreme values in the output rules is to enable the centroid method of defuzzification, which computes the mean value of the aggregated outputs, to produce a well-balanced and realistic brake suggestion. This method ensures that the brake suggestions are evenly distributed, enabling for all braking actions to be based on the average output of the centroid method.

b. Trial 2: MF Adjustment

Subsequent to the rules adjustment, consideration goes to optimizing the membership functions of the fuzzy sets. During the tuning implementation trial of the fuzzy logic system, the modification of the membership functions is carried out, with especially focus on the output membership functions. This adjustment becomes essential when it is noticed that the current membership functions do not efficiently translate fuzzy inputs into real-world brake decisions. Significantly, the system's responsiveness and accuracy are limited since the centroid approach for defuzzification prevents the extreme values (minimum and maximum) required for specific driving circumstances from being obtained.

In an effort to deal with this issue, the membership functions for the braking decision are adjusted by expanding the range of these functions beyond the original 0 to 100% scale. The minimum values are modified to be lowered below 0%, while the maximum values are expanded above 100%. The adjustment of the membership function for the brake decision, to be shown in Figure 4.13.

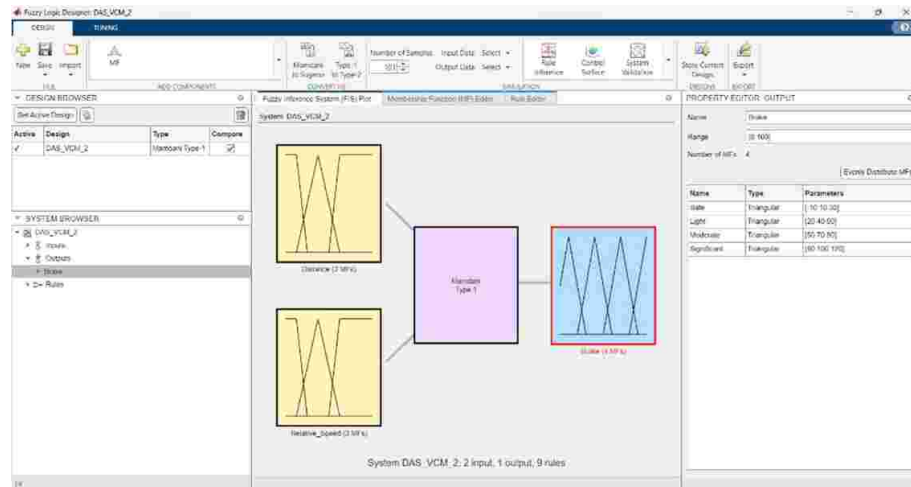


Figure 4.13 Fuzzy System Membership Function Adjustment

The expanded adjustment only modifies the minimum and maximum values of the Brake Decision membership function, while the membership functions of the other variables remain unchanged. The Brake Decision membership function extension enables the system to successfully collect and respond to more severe situations, guaranteeing that the fuzzy logic controller can handle a wider range of braking intensities. This modification guarantees that the output can be transformed into a more precise braking action, hence improving the overall efficiency and dependability of the system in real-world driving experiences.

4.1.3.3 Kinematics and Perception Algorithms

The integration of the sensor and actuator for single system kinematics and perceptions already done developed, the final step of the integration build phase is the complete system integration, which is combine the two integrated system into the final system. The final system integration started with translating the Fuzzy System build in Matlab into the Arduino code, when it is done integration code to the Kinematics and Perception Algorithm integration system should be worked together. The Kinematics and Perception algorithm integration flow is shown in the Figure 4.14.

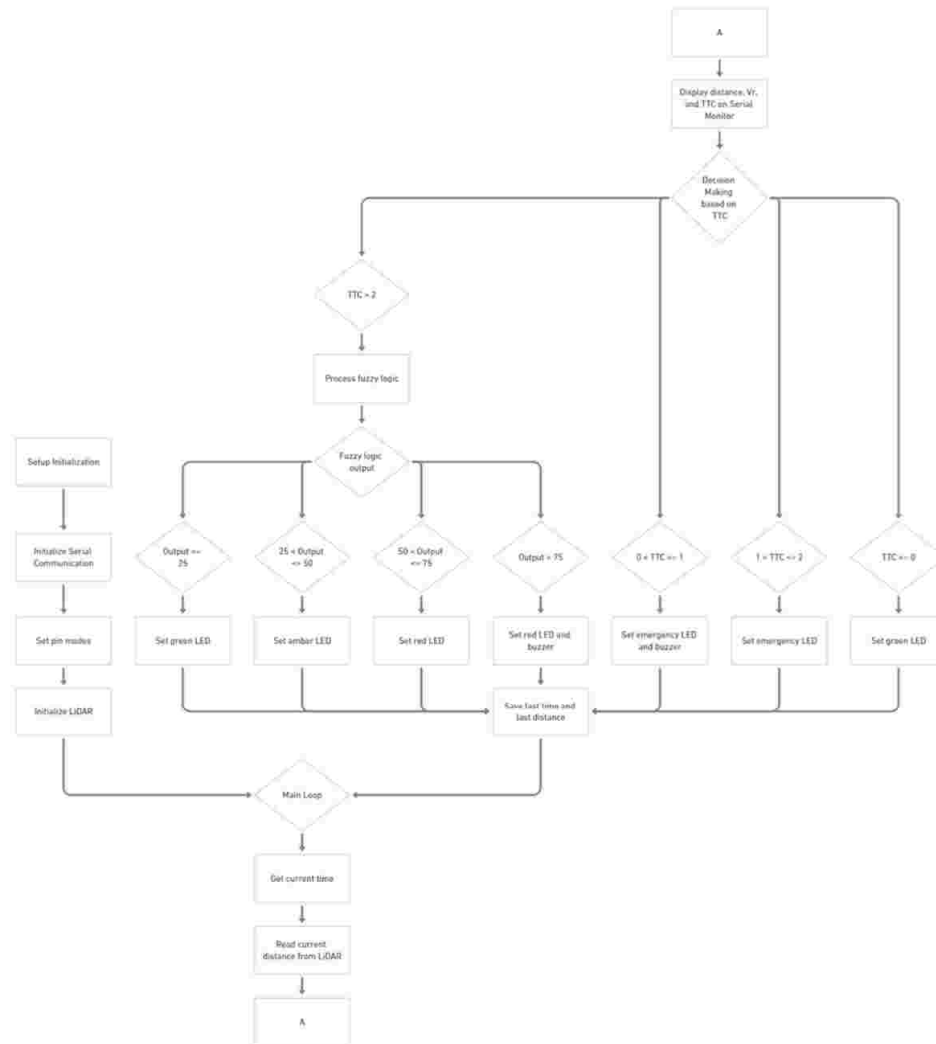


Figure 4.14 Driver Assistance Collision Warning System with Kinematics and Perception Algorithm Flowchart

The flowchart describes the operating logic of an Arduino-based Driver Assistance Collision Warning System that using Kinematics and Perception Algorithms. The primary decision is made by checking the Time-to-Collision (TTC). If the Time-to-Collision (TTC) is less than 2 seconds, the system will employ a kinematics algorithm approach to determine the output. While the Time-To-Collision (TTC) exceeds 2 seconds, the system employs fuzzy-perception-based logic to calculate the suitable output.

4.1.3.4 Integration Test

After the integration build of the components into single compiled system, kinematics-perception-based system, it shall be going to the integration testing. In the integration testing phase, the objective is to verify the seamless interaction between the kinematics unit (Time-to-Collision (TTC) threshold) and the perception unit (Fuzzy Logic). This phase ensures that there is no interference between the two algorithms. During the TTC threshold, only the Emergency brake indicator will be activated, and vice versa. When the TTC is above the threshold, only the Brake Suggestion indicator will be displayed, based on the Fuzzy Logic. The complete integration of these elements is crucial for the overall efficiency and reliability of the collision warning system.

Test Condition:

The collision warning driver assistance system has been placed at the top of the model remote control vehicle and directed towards the rigid barrier. Subsequently, the remote control vehicle approaches the rigid barrier, leading to a variation of the distance value. The position should be in the various conditions in order to accurately represent a variety of warning possibilities.

Acceptance Condition:

The system warning indication must accurately reflect the system's algorithm. The output warning of the system should align with the system's logic when making kinematics-based or perception-based decisions. The system decision truth table is shown in the Table 4.6.

Table 4.6 Collision Warning System Integration Test Truth Table

No.	TTC (sec)	Decision Algorithm	Indicator
1.	0	Kinematics	Brake – Green
2.	$0 < \text{TTC} < 1$	Kinematics	Retarder – Amber, Buzzer
3.	$1 < \text{TTC} < 2$	Kinematics	Retarder – Amber
4.	$\text{TTC} > 2$	Perception	Brake – Green/Amber/Red

Integration Test Result:

Throughout the integration test, the sensor's reactions were carefully observed and documented as the remote-controlled automobile got closer the barrier. The responses were compared to the expected results as determined by the Table 4.6. The activity testing of the Integration Test is shown in the Figure 4.15.

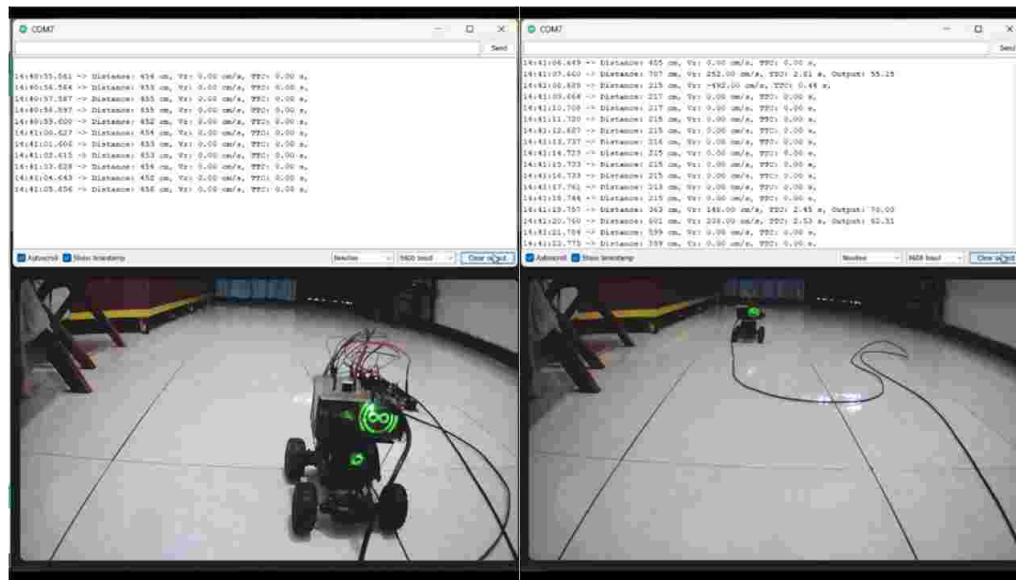


Figure 4.15 Integration Test of the Collision Warning System

The integration test has gone well, satisfying all the acceptance criteria. Therefore, the system proved the capacity to make accurate decisions throughout various scenarios, as indicated by the result shown in Table 4.7.

Table 4.7 Integration Test Result of Collision Warning System

Test No.	TTC (cm/s)	Fuzzy Output	Decision Algorithm	Indicator	Judgment
1.	2.81	55.25	Perception	Brake – Red	Pass
2.	0.44	-	Kinematics	Retarder, Buzzer	Pass
3.	2.45	70.00	Perception	Brake – Red	Pass
4.	2.53	62.51	Perception	Brake – Red	Pass

In overall, the integration test for the system under the specified conditions was declared Pass. The system demonstrated reliable and accurate decision-making capabilities throughout the testing scenarios, therefore proving its conformity to the given acceptance criteria.

4.2 System Verification

4.2.1 System Test

The system testing phase for the collision warning system aims to validate the complete and integrated architecture of the system, ensuring it meets the specified requirements and functions correctly. The objective is to evaluate the overall system performance by testing all integrated components and their interactions. Test cases include verifying the accuracy of distance data transfer from the LIDAR to the Arduino Mega 2560, checking the correctness of speed and TTC calculations, and ensuring the fuzzy logic system generates appropriate warning responses. Additionally, it will be tested the edge cases such as sudden changes in distance and speed to ensure robustness. This phase ensures that data from the LIDAR sensor is accurately processed by the kinematics unit to calculate relative speed and Time-to-Collision (TTC), and then correctly interpreted by the fuzzy logic system in the perception unit.

Acceptance Condition:

The system's calculation of relative speed and time-to-collision should be accurate. Additionally, the system's output must comply to the logic of the system, whether it is making a decision based on kinematics or perception algorithms.

Data Source:

The data was acquired from the integration test, which included Time-to-Collision (TTC) numbers and other the relevant variables.

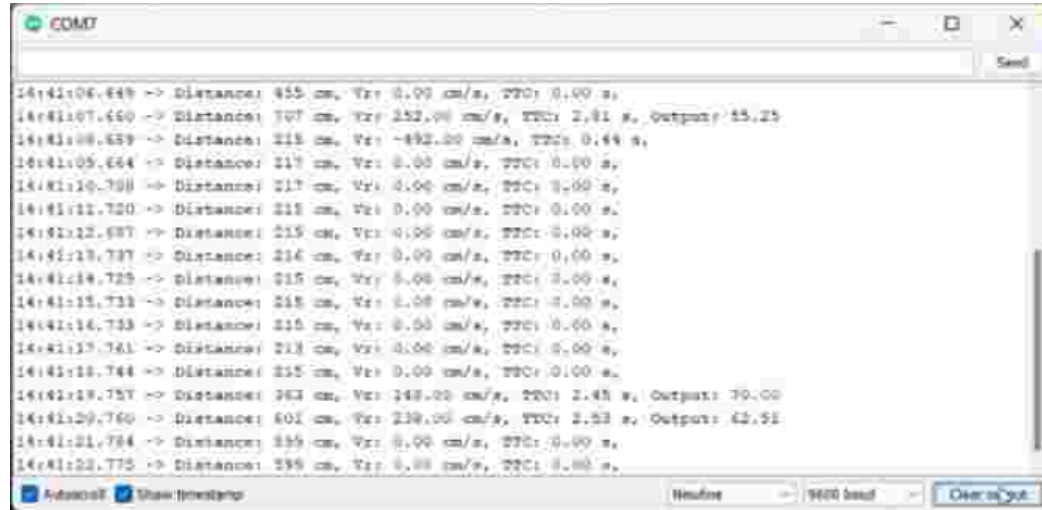


Figure 4.16 Integration Test Serial Monitor Result of Collision Warning System

Calculation:

TTC Calculation: utilising fundamental formulas to validate the system's outputs. The Time-to-Collision (TTC) can be calculated using Equation 2.2, which involves the relative speed (v) and the distance (d). The term relative speed (v) is defined by Equation 2.1, where d represents the distance travelled during time t , and d represents the distance travelled during time t .

Fuzzy Output Calculation: involves utilizing the rule interference in the Matlab Fuzzy Logic Designer to input the crisp values of Distance and Relative Speed into the Matlab Fuzzy System.

1. Test No. 1

$$d_1=455 \text{ cm}, d_2=707 \text{ cm}, t_1=06.649 \approx 7 \text{ s}, t_2=07.660 \approx 8 \text{ s}$$

$$v_r = \frac{707-455}{8-7} = 252 \text{ cm/s}$$

$$TTC = \frac{707}{252} = 2.81 \text{ s}$$

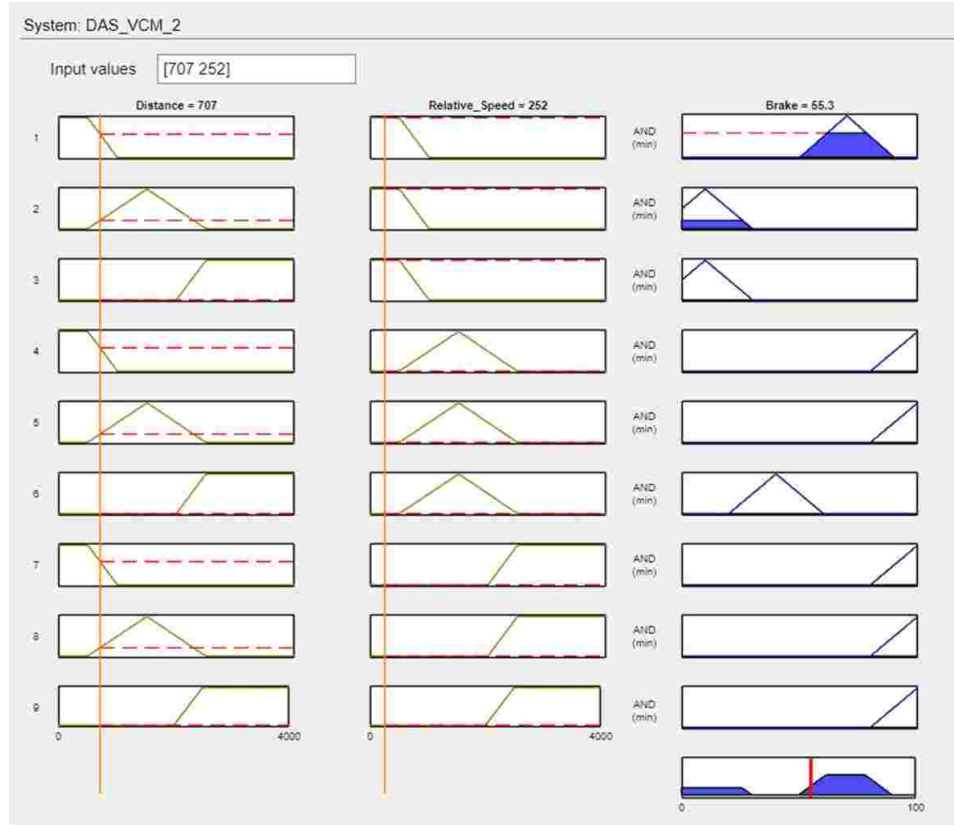


Figure 4.17 Test No. 1 Fuzzy Output Result

2. Test No. 2

$$d_1=707 \text{ cm}, d_2=215, \text{ cm}, t_1=07.660\approx 8 \text{ s}, t_2=08.689\approx 9 \text{ s}$$

$$v_r = \frac{215-707}{9-8} = -492 \text{ cm/s (closer)}$$

$$TTC = \frac{707-215}{492} = 0.44 \text{ s}$$

3. Test No. 3

$$d_1=215 \text{ cm}, d_2=363 \text{ cm}, t_1=18.744\approx 19 \text{ s}, t_2=19.757\approx 20 \text{ s}$$

$$v_r = \frac{363-215}{20-19} = 148 \text{ cm/s}$$

$$TTC = \frac{363-215}{148} = 2.45 \text{ s}$$

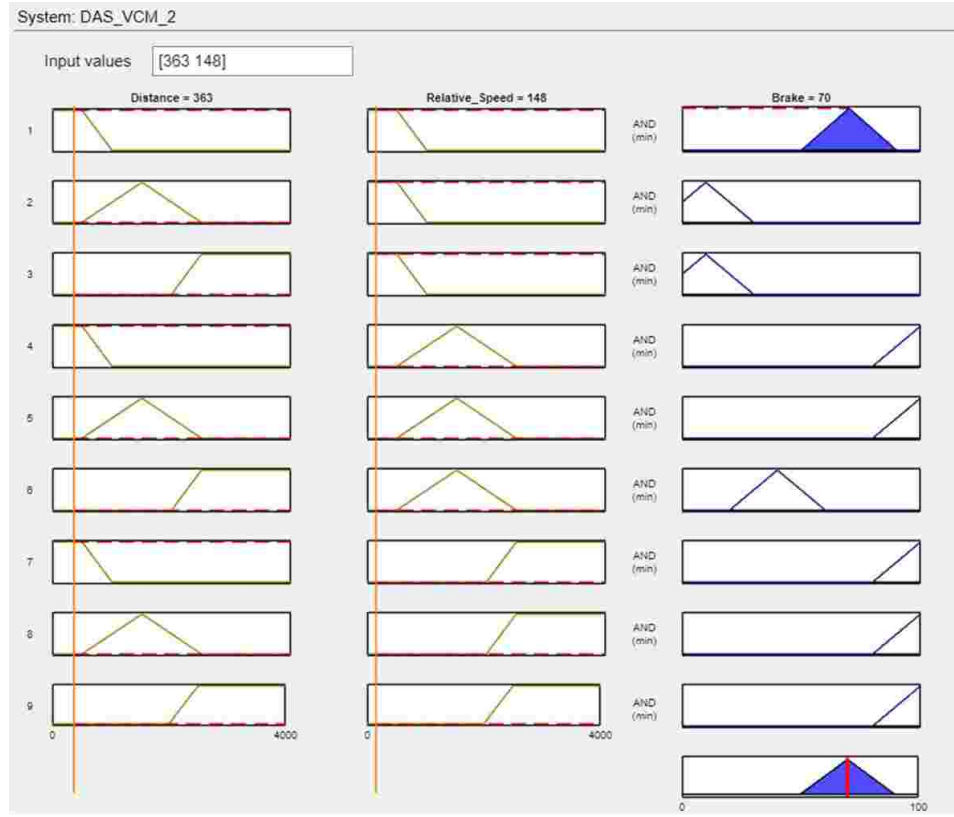


Figure 4.18 Test No. 3 Fuzzy Output Result

4. Test No. 4

$$d_1=363 \text{ cm}, d_2=601 \text{ cm}, t_1=19.757\approx 20 \text{ s}, t_2=20.760\approx 21 \text{ s}$$

$$v_r=\frac{601-363}{21-20}=238 \text{ cm/s}$$

$$TTC=\frac{363}{148}=2.53 \text{ s}$$

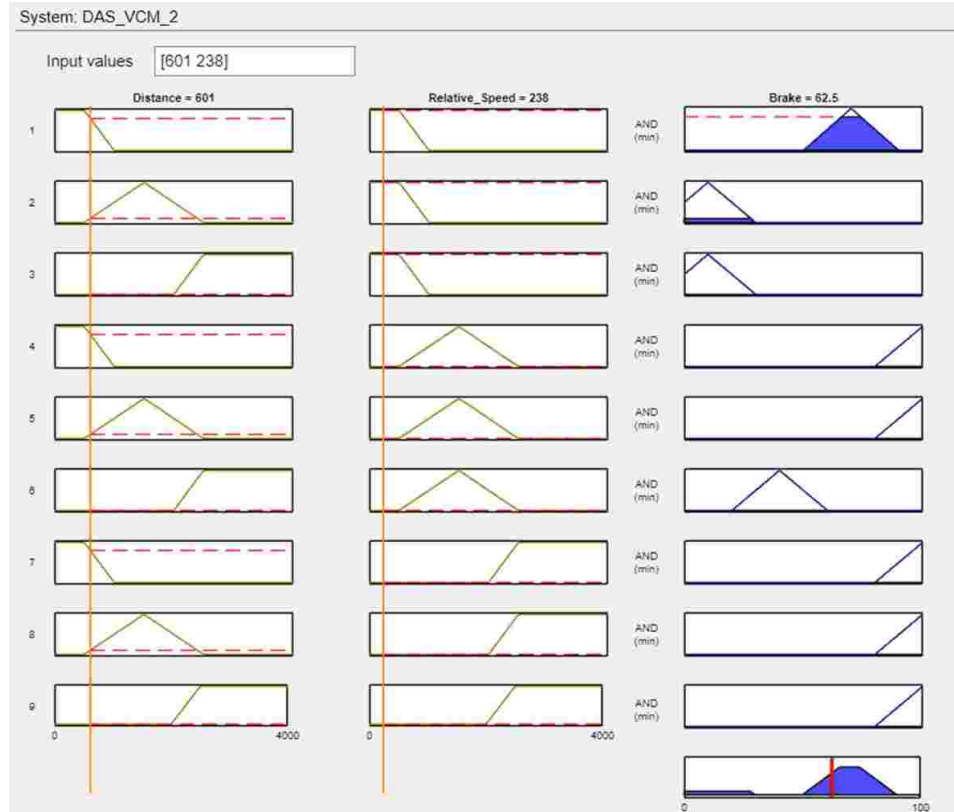


Figure 4.19 Test No. 3 Fuzzy Output Result

Testing Result:

The system test has been determined to be Pass. The manual calculations utilising fundamental formulas are compared to the automated calculations of the system, thereby approving the accuracy of the system's decision-making algorithms. The successful test achievement represents tangible evidence of the system's ability to function with precision and reliability in real-world circumstances, guaranteeing both safety and performance. The detail result is shown in Table 4.8.

Table 4.8 System Test Result of Collision Warning System

Test No.	System Output		Verification		Judgment
	TTC	Fuzzy Output	TTC Manual	Matlab Output	
1.	2.81 cm/s	55.25	2.81 cm/s	55.3	Pass
2.	0.44 cm/s	-	0.44 cm/s	-	Pass
3.	2.45 cm/s	70.00	2.45 cm/s	70	Pass
4.	2.53 cm/s	62.51	2.53 cm/s	62.5	Pass

4.2.2 Acceptance Test

The acceptance testing ensures the system meets end-user requirements and performs reliably under diverse environmental conditions. The collision warning system utilizes a LIDAR sensor as its primary system. Therefore, the acceptance test will specifically evaluate the performance of the LIDAR sensor. Based on the research from Park J. et al. from Hyundai Motor Company, there are 8 different environmental factors to examine the LIDAR performance: cover contamination, strong sunlight, high temperature, low temperature, vibration, interference, reflectivity of a target, and transitions between day and night [39]. In an effort to emphasize driver experience, the acceptance tests for the collision warning system will consist of real-world driving scenarios in the Indonesian environment. These tests will evaluate the system's performance through three assessments. The initial test focused on cover contamination, which includes fog, rain, and dust. The second test involves exposure to strong sunlight, while the third test focuses on the transition between day and night.

Test condition:

In the initial phase, cover contamination was simulated by the creation of foggy conditions. The chamber room was contaminated with artificial smoke to create a dense fog atmosphere, which effectively decreased visibility, simulating real-world fog circumstances. The purpose of this setup was to simulate the difficulties that a sensor may meet in settings with limited visibility, such as those experienced during dense fog on the road.

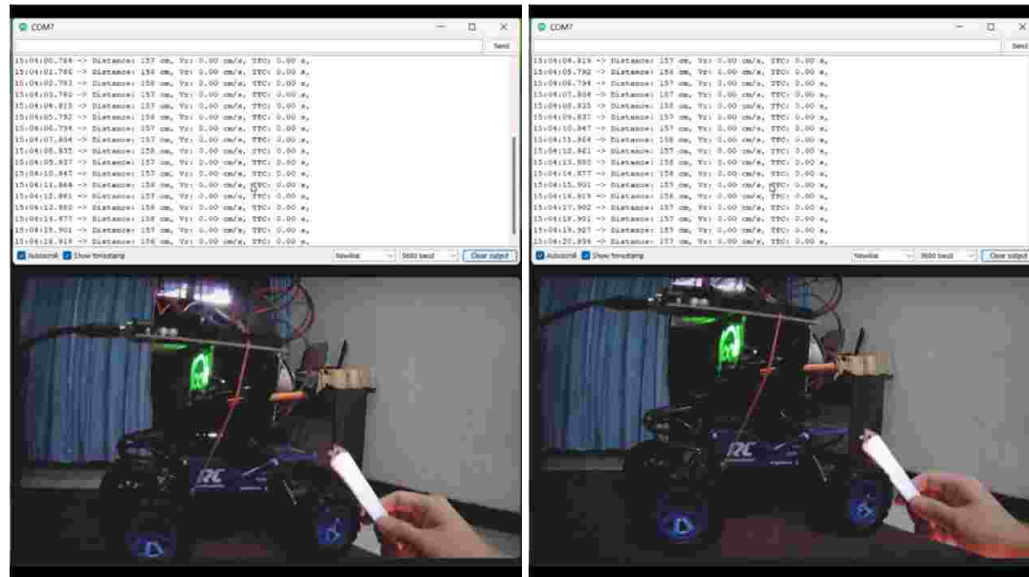


Figure 4.20 Acceptance Test – Cover Contamination (Fog)

The second phase involves the contamination of the cover by rain. Artificial raindrops was generated by a manual droplet creation mechanism built to replicate the process of rainfall. This required the act of applying water onto the sensor's visibility, essentially covering its surface with droplets that simulate those which it would come through in a rainfall.

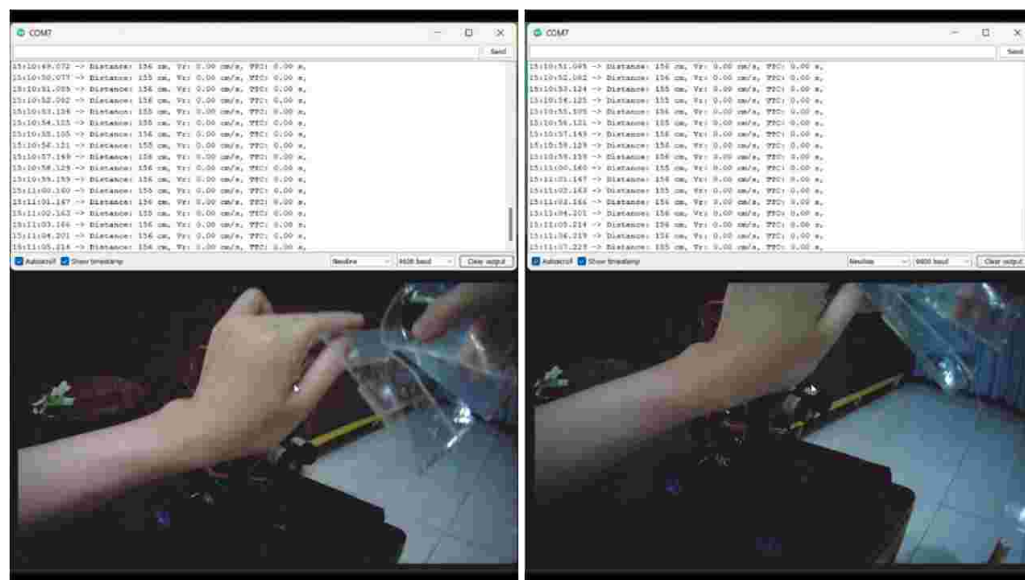


Figure 4.21 Acceptance Test – Cover Contamination (Rain)

The third phase focused on the contamination of covers by dust. In this situation, the sensor was covered by an acrylic covering that had been previously covered with powder to imitate a dusty condition. The purpose of this test was to assess the performance of the sensor while its optical components were blocked by dust particles.

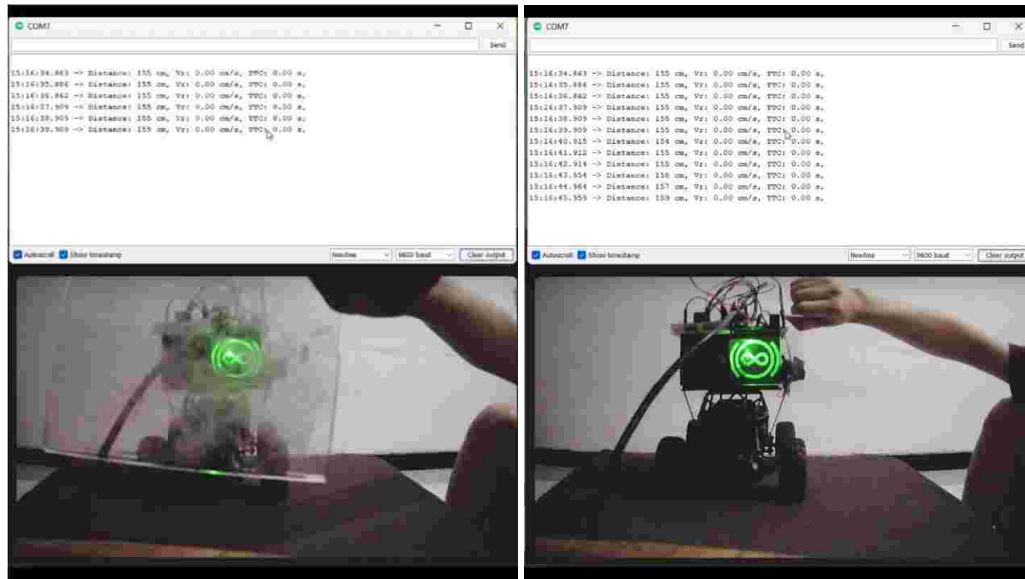


Figure 4.22 Acceptance Test – Cover Contamination (Dust)

The fourth test emulated rapid day-night transitions. The illumination in the chamber room performed two rapid cycles of transition, alternating between bright and dark. This simulation intended for imitating the sudden shifts in lighting that a sensor could encounter during the transition from day to night or vice versa. These tests assess the system's capacity to adapt and maintain accuracy under different lighting conditions, especially during the shift from day to night, to ensure reliable as well as consistent performance.

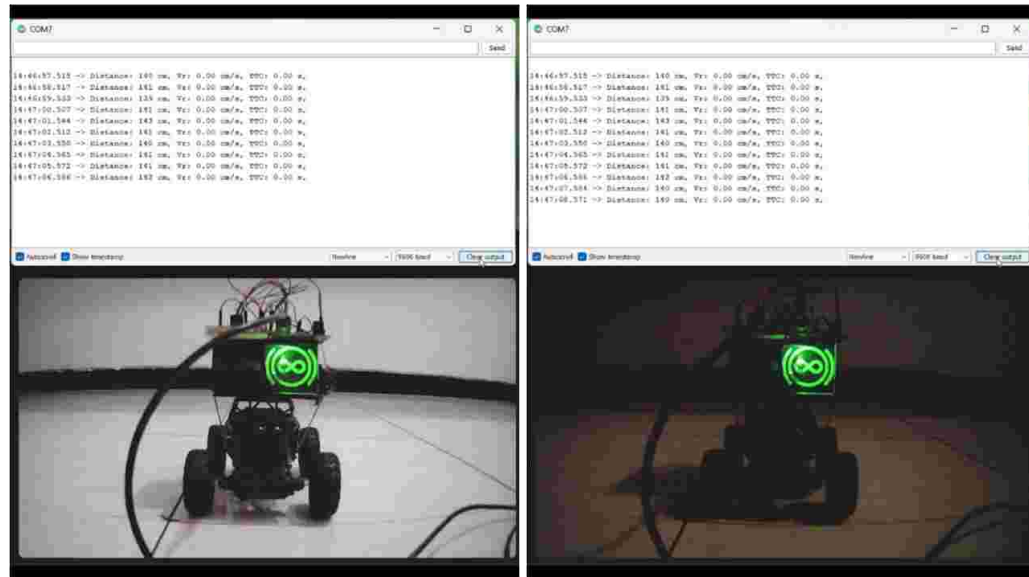


Figure 4.23 Acceptance Test – Day night Transition

At last, the fifth test was subjecting the sensor to strong sunlight conditions. A strong flashlight was used within the same chamber to illuminate the sensor from various angles, consequently replicating the impact of direct sunlight and glare. This test assesses the system's performance and reliability under bright, direct sunlight, guaranteeing its capacity to precisely identify and respond to collisions possibility.

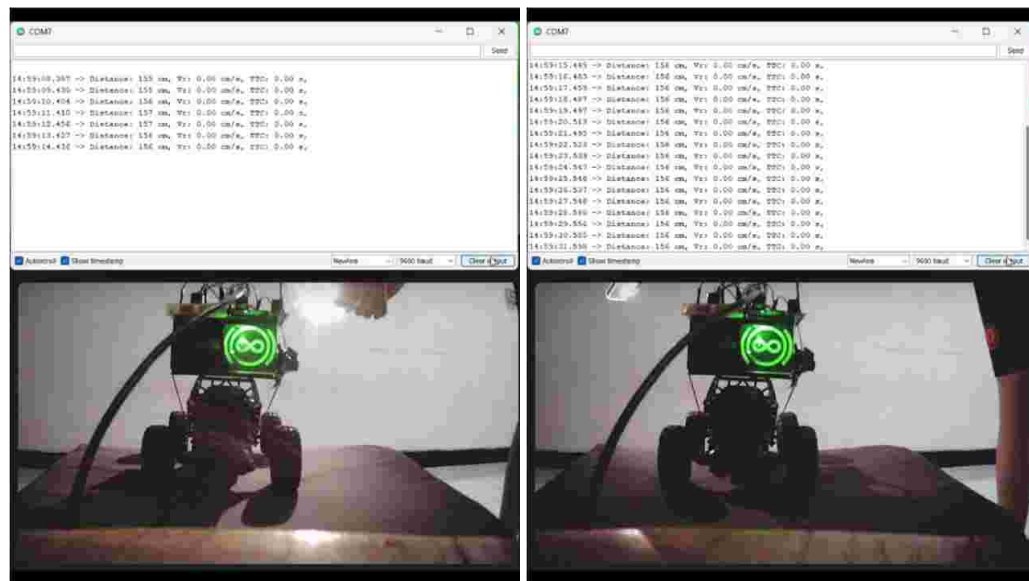


Figure 4.24 Acceptance Test – Strong Sunlight

Testing Result:

The sensor system was able to pass the acceptance test conducted under real-world condition simulations. The sensor system demonstrated great reliability and performance in all evaluated scenarios, fulfilling the acceptance criteria. The summary of the acceptance test result is shown in the Table 4.9.

Table 4.9 Acceptance Test Result for Collision Warning System

Test No.	Requirement + Test	Standard	Result	Remark	Judgment
1.	Fog	± 5 cm	Deviation 2 cm	No interference to sensor	Pass
2.	Rain	± 5 cm	Deviation 1 cm	No interference to sensor	Pass
3.	Dust	± 5 cm	Deviation 5 cm	No interference to sensor	Pass
4.	Strong Sunlight	± 5 cm	Deviation 2 cm	No interference to sensor	Pass
5.	Day-night transition	± 5 cm	Deviation 3 cm	No interference to sensor	Pass

The sensor successfully measured the distance in a simulation of fog conditions, hence ensuring safety in situations with limited visibility. While executing a rain cover contamination simulation, the sensor's functionality remained unaffected by the presence of water, ensuring accurate and reliable obstacle identification. The sensor successfully maintained its accuracy and performance even when subjected to dust cover contamination during the simulation. During the simulation of intense sunlight, the sensor successfully handled glare and bright light, while still maintaining accuracy and consistent detection abilities. Throughout the day-night transition simulation, the sensor simply adjusted to rapid changes in brightness, guaranteeing uninterrupted and accurate detection and decision-making. The satisfactory test results validate the system's readiness for implementation in various and demanding environments, guaranteeing safety and operational efficiency.