Prototype of Autonomous Line-Following Vehicle

1st Rubayet Kamal
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
rubayet.kamal@stud.hshl.de

2nd Yuming Wang
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
yuming.wang@stud.hshl.de

3rd Mert Yavas
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
mert.yavas@stud.hshl.de

4th Joseph Asare
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
joseph-asare.owusu@stud.hshl.de

***Abstract*—This project report details the design and implementation of a line-following autonomous vehicle prototype. The vehicle's components were designed using Solidworks and Fusion 360, and subsequently manufactured through laser cutting and 3D printing. The prototype employs line sensors to follow a predefined path, while integrated ultrasonic sensors enable real-time obstacle detection. These sensors allow the vehicle to assess its surroundings and make informed decisions to avoid collisions. Furthermore, the system includes color sensors to identify and react to different obstacle colors, enhancing its adaptability and decision-making capabilities. This project demonstrates the vehicle's ability to operate autonomously in dynamic environments, showcasing the integration of multiple sensor technologies to create a robust and intelligent navigation system proficient in both line-following and obstacle management.**

***Index Terms*—System Modelling Language (SysML) Polylactic acid (PLA)**

I. Introduction

This paper focuses on building a prototype using knowledge gained from previous semesters. The main objective is to build a line following car with the ability to detect and avoid obstacles based on colors detected. After the color is evaluated the car should be able to perform an action based on the color.

Section II details the analysis of the line-following robot using SysML. In Section III, the design process is outlined, explaining how the project evolved from concept to execution. Section IV describes the components used, listing and explaining the roles of the sensors and actuators integral to the design, with an emphasis on their inter-dependencies.

Once the design and wiring and soldering of components were completed, programming commenced. This is covered in Section V, which explains the algorithms and includes code snippets demonstrating the execution of various functions.

Section VI describes the simulation of the prototype using UPPAAL-5.0.0, while Section VII presents the results obtained in the real world. The final section, Section VIII, concludes the paper, outlining the project's outcomes and potential future work.

II. Analysis using SysML

SysML supported the "specification, analysis, design, verification and validation of our Project. There are different SysML diagrams. In this project we used requirement diagrams and sequence diagrams to detail the processes gone through in this project" [1].

A. Requirements

The requirement was to develop an autonomous vehicle capable of driving autonomously on a specified track using line detection. Several critical requirements were established and visualized through a requirement diagram, which offers a graphical representation of the requirements, using Visual Paradigm Online [2]. As shown in Figure 1. the primary objective was to create a prototype of an autonomous vehicle, which encompasses three main areas: Obstacle Management, Track Management, and Direction Management.

Within Obstacle Management, an additional requirement, Colour Management, was derived, emphasizing the vehicle's need to detect obstacles and perform functions based on their color. This area also included a refinement that specifies the distance at which the vehicle must stop and begin evaluating obstacles, ensuring timely and accurate obstacle handling.

Track Management necessitated the derivation of a requirement focused on Speed Optimization. This requirement was crucial because the vehicle needed to adjust its speed dynamically whenever it makes right or left turns, ensuring smooth and efficient navigation on the track.

Direction Management led to the development of a requirement indicating that the vehicle should be capable of executing different routing strategies. This requirement was further detailed into four specific types of routing that the vehicle should manage, showcasing its versatility in navigation.

By integrating these requirements, the vehicle prototype aims to be a sophisticated autonomous system, proficient in obstacle detection and response, track navigation with speed adjustments, and versatile routing capabilities.

B. Requirement Diagram

Requirement diagram is used to illustrate our goals and requirements for the prototype. As shown in Figure 1, our main requirement is autonomous vehicle, from here we have three sub-requirements which are track management, obstacles management, and direction management. From obtacles management, as you can see, braking distance refines it. One test case is used to verify the color management.

C. State-Machine Diagram

State machine is a tool we use to analyze different states of the prototype. It shows complete state-dependent reactive behavior of given elements [1]. In Figure 2, we describe the

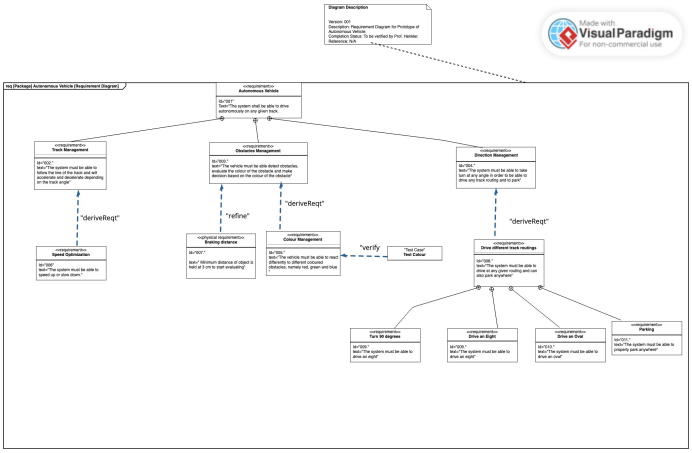

Fig. 1. Requirement Diagram

whole state changes of our prototype through its lifetime where the car will follow the line, and do object detection, color detection. For example,when the power is on, the car will first go in move forward state, here the color detection will be activated, then if red is detected, the car will then go to red state. There is one special case for blue state. Here is one condition choosing, which means if there is one object within the range set , the car will go back to go forward state, otherwise the car will go to stop state.

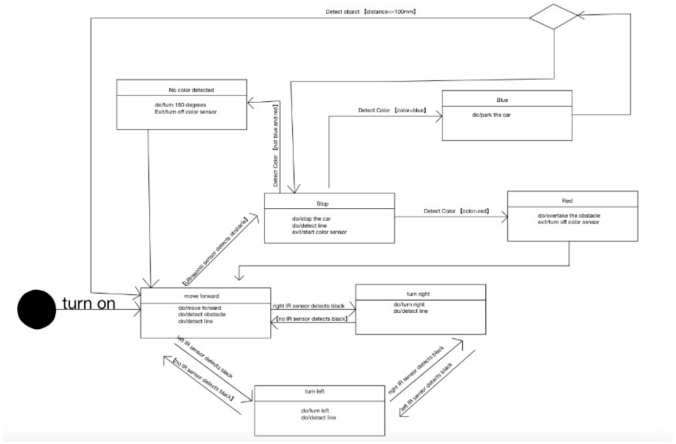

Fig. 2. State Machine Diagram

D. Activity Diagram

Activity diagram describes scenarios describing required activities [1]. As shown in Figure 3, When the power is turned on, the car will first move forward, and then the obstacle management will be activated. If there is an object detected, the car will then go to stop activity, otherwise, the car will go to track management activity which ensures the car follows the line. After the stop activity, the color evaluation activity is processed. Here we use color sensor to realize the goal of color detection. After this activity, the car will then go back to obstacle management.

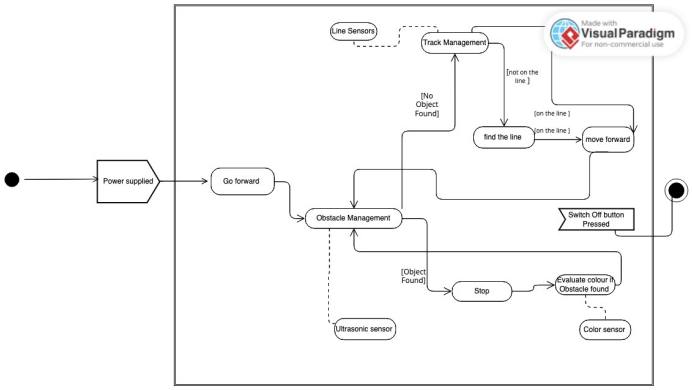

Fig. 3. Activity Diagram

E. Block Definition Diagram

A Block Definition Diagram(BDD) is a structural diagram that shows system components, their contents, interfaces, and relationships [1]. In our prototype, as shown in Fig. 4, we use a 3200 mah battery, power bank, DC motor, ultrasonic sensor, color sensor, and IR sensor. The relationship between these components is shown with a diamond and an arrow. A filled-in diamond means that there is a 'Composition' relationship. This means that if the component above is destroyed, the ones below too will not exist. Fig. 4 outlines an illustration of the parts and values used in this project and the components' relationship. For example, in the block sensor subsystem, there are three blocks composed: color sensor, ultrasonic sensor, and infrared sensor.

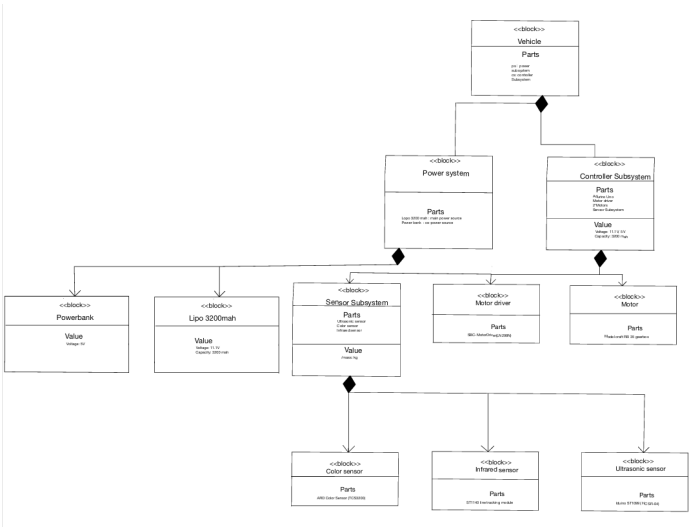

Fig. 4. Block Definition Diagram

F. Sequence Diagram for Colour Evaluation

Sequence Diagram for Color sensor evaluation shows the order of the color evaluation behavior. In this case, as shown in Fig. 5. there are three similar procedures for color detection. For blue detecting example, the user will first give order to

system to detect blue for 3 times, then the system will wait for the data from the color sensor, if the color is detected, the data will be sent back from system to User.

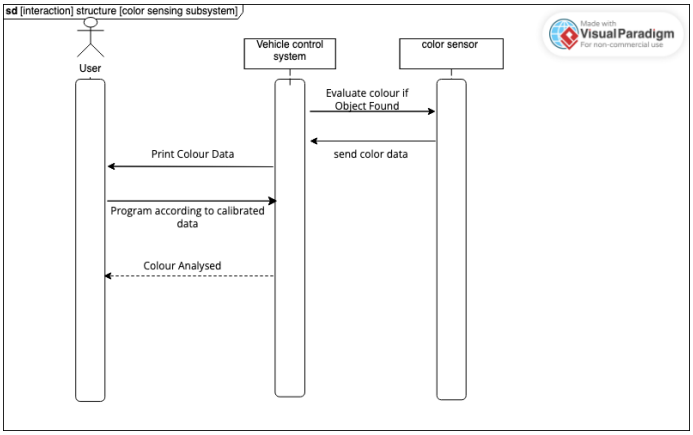

Fig. 5. Sequence Diagram For Color Sensing

G. Sequence Diagram for Following Line

The sequence Diagram for the Following Line describes the order of the line following management. In this case, as shown in Fig. 6, the User, the Car, the Control system, right IR sensor, and left IR sensor are showed. The sequence between these parts can also be checked. For example, when the left IR sensor sends a signal to the control signal, after receiving the signal, the system will send data to the car to make sure it makes a slightly left-shift behavior.

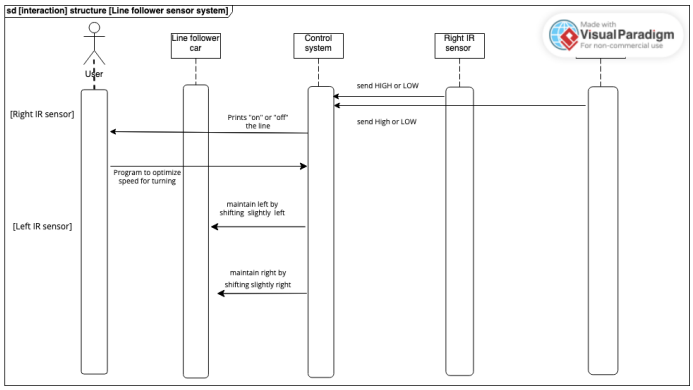

Fig. 6. Sequence Diagram For Following Line

H. Sequence Diagram for Obstacle Management

Sequence Diagram for Obstacle management shows how we realize the function of avoiding obstacles, and the sequence of it. For example, in Fig. 7, when the Ultrasonic sensor sends a signal to the control system, the control system will then start analyze color by sending a signal to the color sensor. The return signal from the color sensor will be sent back to control system.

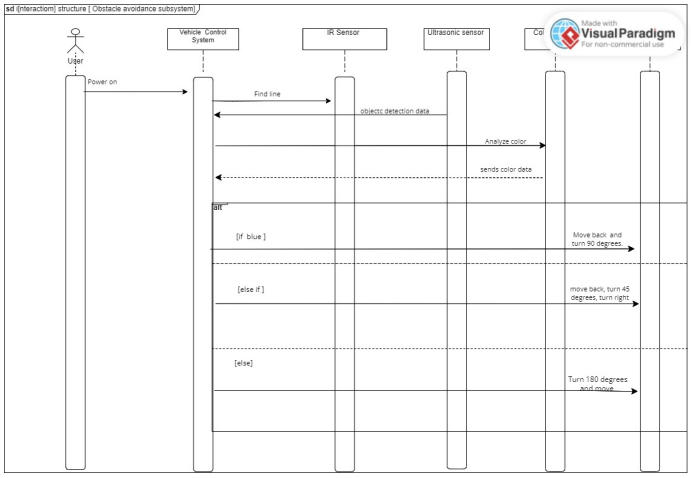

Fig. 7. Sequence Diagram Showing Interaction Between Components Used In Ensuring Object Avoidance

III. Design

SolidWorks [3] and Fusion 360 [4] were employed in designing various essential components of the prototype. The designs were then exported as .dxf files for laser cutting, with wood serving as the base material, and as .stl files for 3D printing using PLA. Fig. 8. displays the design in drawing mode whereas Fig. 9. presents the comprehensive 3D views from different perspectives of the designed prototype.

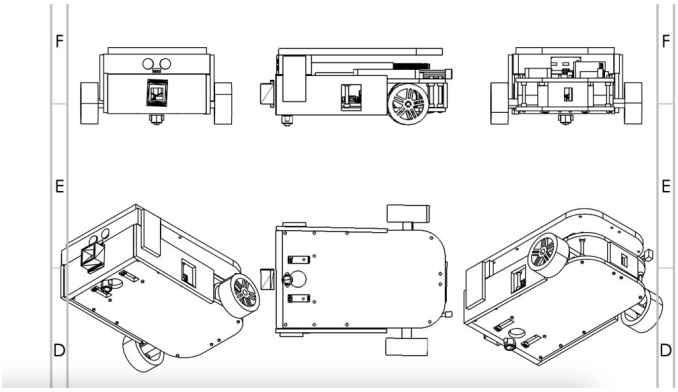

Fig. 8. Drawing

A number of different parts were designed separately to make this possible. These parts will be discussed in the upcoming subsection, detailing how the shapes of these parts played a crucial role in the prototype.

A. Base, Body and Cover

The design process began with the creation of the base, a crucial component influenced by several key considerations. First and foremost, the ST 1140 sensor functions optimally within a specific range, necessitating precise placement to ensure stable performance. To achieve this, we crafted two holes tailored to fit our line sensor accurately. Additionally, the front wheel, a ball-bearing wheel, required specific radius

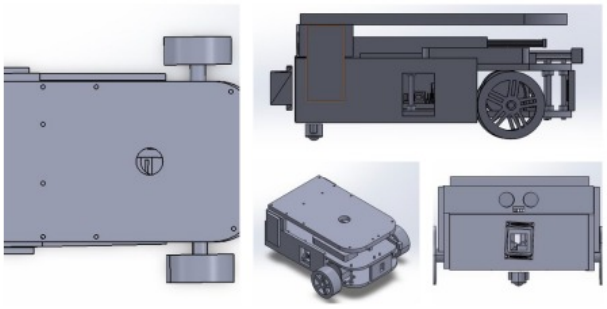

Fig. 9. 3D View

and height specifications. To accommodate this, we designed a hole with a radius slightly smaller than the wheel's, ensuring a secure fit while maximizing its height. All components were then screwed in place, providing stability and allowing for easy modifications when needed.

The body and cover were designed with identical shapes using SolidWorks and then laser-cut from a .dxf file. The base and cover were constructed from 6 mm thick wood, while the body, which supports more weight, was made from 10 mm thick wood. This variation in thickness ensured structural integrity where it was most needed. For flexibility in adjustments, the base and body are connected using stick supports, which are detailed further in the next subsection. These supports fit into holes positioned at various corners, allowing for a stable yet adjustable assembly. Fig. 10. below illustrates these adjustments and components, showcasing the careful consideration given to each design element.

Fig. 10. 3D view of base, body or cover

B. Stick support

The stick support plays a crucial role in facilitating easy modifications to the wiring and other components. These supports function as finger joints, allowing for straightforward connection and disconnection between the base and the body. We utilized a total of four sticks, one of which is shown in Figure 11 below. Each stick was 3D printed using PLA, with a production time of approximately 10 minutes per stick.

C. Colour Sensor Cover

The TCS3200 color sensor is highly sensitive to ambient light [5], making calibration challenging as the values fluctuate with changes in surrounding light. To address this issue, a color sensor cover was designed to ensure that the photo-resistor only receives light from the target source, effectively blocking out ambient light. The cover has a length of 3 cm, which is the optimal distance for the color sensor to provide accurate results according to the manufacturer's specifications. This cover was mounted on the front face of the prototype, parallel to the ultrasonic sensor, ensuring that the vehicle stops at the exact distance where the color sensor can accurately evaluate the color. Figure 12 illustrates the discussed part.

Fig. 11. Stick Support

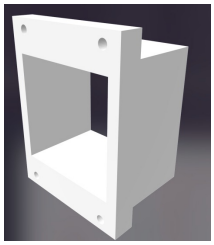

Fig. 12. Colour Sensor Cover

D. Ultrasonic Holder

The Ultrasonic sensor, HC SR-04, can detect objects within a 45-degree range from its face [6], making its stability crucial. An unstable sensor may cause the prototype to fail in stopping at the desired distance. Moreover, a moving sensor may detect noise, even without an object in front of it. To address this challenge, a stabilizing part was designed in Fusion 360 and 3D printed using PLA. Figure 13 shows the designed stabilizing part.

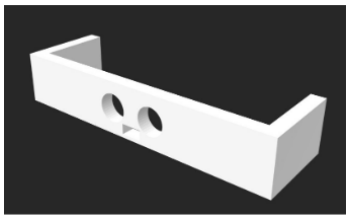

Fig. 13. Ultrasonic Holder

E. 2 * Motor Holder

Two 9V DC motors were used to drive the wheels of the prototype. Ensuring the wheels' stability is crucial to maintain a consistent height relative to the ground, preventing misalignment or bending of the vehicle. To secure the motors and ensure wheel stability, a specific part was designed in SolidWorks and then 3D printed. These parts feature 2.5 mm screw holes, allowing for easy screw attachment to the base. Figure 14 shows the designed part.

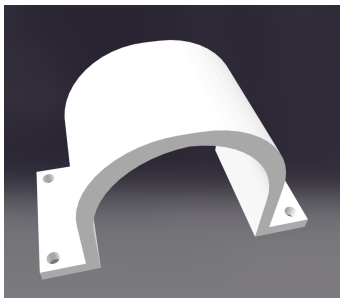

Fig. 14. Motor Holder

IV. Components

Various hardware components were used in this project with each having distinct properties and performing specific functions. Components incorporated are mentioned below along with some of their schematic. Schematics shown in this paper are created with TinkerCAD [7]. Figure 15 below, shows the complete schematic including all components.

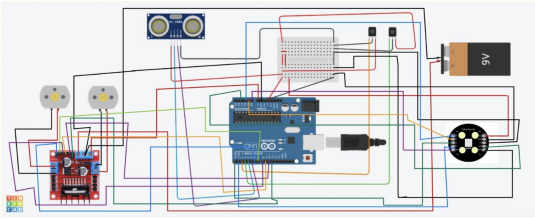

Fig. 15. Complete Schematic

A. Arduino UNO

"The Arduino Uno serves as the brain of our prototype, responsible for gathering data from sensors, processing it, and initiating appropriate actions through actuators for navigation purposes. This development board incorporates the Atmega328 microcontroller from the Atmel microcontroller family, featuring an 8-bit RISC processor core. The board is equipped with 14 digital GPIO pins and 6 analog pins, as shown in Figure 16. Communication with the PC is established through a USB connection" [8].

Key features of the Arduino Uno include "the Atmega328P microcontroller, an 8-bit RISC processor core, 14 digital GPIO pins, and 6 analog pins. These features enable the Arduino Uno to effectively manage sensor data and control actuators, making it an integral part of the prototype's operation" [8].

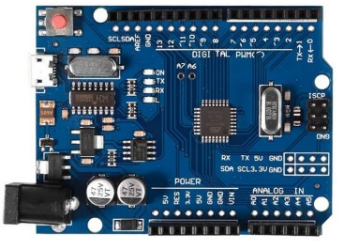

Fig. 16. Arduino Uno

B. Battery

Conrad 7.4V 3000 mAh 20C Eco-Line LiPo battery, as shown in Fig. 17, was used to power L298N. The battery has a weight of 210 g according to the manufacturer [9].

Fig. 17. LiPo Battery

C. 9V DC Motors

Two 9V DC Motors as shown in the Fig. 18. [10]. were the primary tool to run back wheels. These motors were powered from the battery as the Arduino does not have enough powering capability. This is where L298N came into use and will be discussed in the next subsection.

Fig. 18. DC Motor

D. L298N

The L298N Motor Controller was used to power both the DC motors and the Arduino Uno. Since the Arduino Uno cannot supply more than 5 volts, the controller is essential for operating the motors. The Motor Controller has three key inputs: the first is the voltage from the battery, the second is the ground connection shared with the Arduino and the

battery, and the third is the 5V input for the Arduino Uno [11]. Additionally, the controller allows for changing the direction of the motors and adjusting the speed of the vehicle analogously [11]. Fig. 19. shows how two DC motors and Arduino were connected to the motor controller.

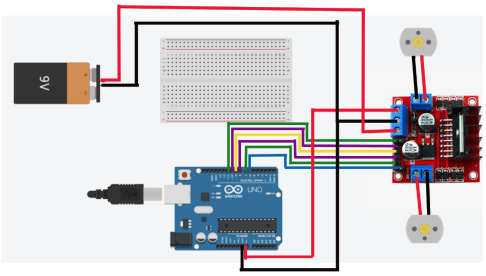

Fig. 19. L298N Motor Controller Schematic

E. HC SR-04

”This module is ideally suited for distance measurement in a range between 2 cm and 3 m. With a resolution of about 3 mm, distances can thus be measured by ultrasonic signal. If a signal (falling edge) is applied to the trigger input, a distance measurement is performed and output at the echo output as a PWM TTL signal. The ultrasonic distance sensor is particularly suitable for obstacle detection, distance measurement, as a level indicator and for industrial applications.

When triggered, the ultrasonic loudspeaker (transducer) emits an ultrasonic noise of maximum 200µs. The ultrasonic loudspeaker emits a 40 kHz signal. This means that 8 periods (edge changes) are emitted within the 200µs in which the sensor emits its ultrasonic noise. In order to arrive at these 8 periods of the 40 kHz signal mathematically.

Transmission of the ultrasonic signal is started when a 10µs long start signal (ActiveHigh) is received at the ”Trigger input pin”. After transmission, the signal is activated at the ”Echo output signal pin” (ActiveHigh). If the reflected signal is now picked up again at the microphone, the echo signal is deactivated again after detection. The time between activation and deactivation of the echo signal can be measured and converted to distance, as this also corresponds to how long it takes the ultrasonic signal to cover the distance between loudspeaker-¿ reflecting wall -¿ microphone in the air. The conversion is then made by approximating a constant air velocity - the distance is then half the distance traveled” [6].

Fig. 20 shows the schematic of ultrasonic sensor with Arduino UNO along with the line sensor (will be discussed in next subsection) [6].

F. ST 1140

”The sensor module detects whether there is a light-reflecting or light-absorbing surface in front of the sensor. The IR sensor emits an infrared light that hits an obstacle, it is reflected and received by the photo-diode. It has a detection

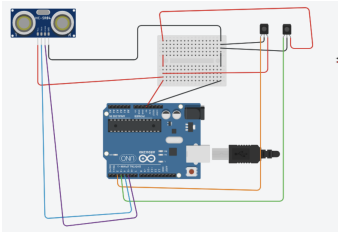

Fig. 20. Schematic of HC SR-04 and ST1140 with Arduino

range between 2 to 20cm. The sensitivity of both the receiver and the strength of the transmitter can be adjusted. This makes it possible to adjust the distance at which the sensor is triggered. This is done with Potentiometers” [12]. In this project, 2 IR sensors were used and were installed at the base of the prototype. They aid in line following and object detection. When a black line is detected, the sensor sends 1 or HIGH to the Microcontroller. On the other hand, when a white or any other line is detected, the sensor sends 0 or LOW to the Microcontroller. This was crucial in forming an algorithm to ensure the prototype followed the line. It can be seen in Fig. 20 represents the schematic of the Line Sensors connected to the Arduino.

G. TCS3200

”The TCS3200 chip is designed to detect the color of the light falling on it. It has an array of photodiodes (a matrix of 8x8, for a total of 64 sensors). These photodiodes are covered with four types of filters. Sixteen sensors have a RED filter and thus can only measure the red component in the incident light. Another sixteen have a GREEN filter and sixteen have a BLUE filter. Each visible color can be divided into three basic colors. So these three types of filtered sensors help measure the weighting of each of the primary colors in the incident light. The remaining 16 sensors have a clear filter. The TCS3200 converts the intensity of the incident lighting into a frequency. The output waveform is a square wave with 50% duty cycle. You can use the timer of a MCU to measure the period of the pulse to determine the frequency. The output of the TCS3200 is available on one line, which means that only 1 pin is available as output. The intensity of the red, green, blue, and clear channels can be determined by using the inputs, S2 and S3, which are used to select the sensors, to provide their output on the output line. That means the inputs S2 and S3 are used to control what kind of sensors give their signal to the output line. There are also two other inputs S0 and S1, which are used to regulate the output frequency. That means that with these inputs the intensity of the frequency of the channels RED, GREEN, BLUE, and CLEAR can be regulated” [5]. Fig. 21 shows a schematic of the color sensor TCS3200 when connected to Arduino UNO.

V. Algorithm

Our prototype is programmed using the Arduino IDE [13]. In this section, we discuss various functions of the

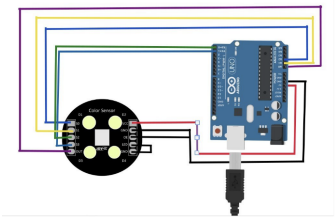

Fig. 21. Colour Sensor Schematic

car and the algorithms governing its behavior. The behaviors of the car are categorized into Line Following, Object Management, Color Evaluation, Overtake, Parking, and U-Turn. It is important to note that the prototype can perform all these functions on tracks with a white surface and a black line. However, for this paper, all results and simulations are demonstrated on a black surface with a white line. A Boolean variable called "whiteLine", as shown in Fig. 22., is created, which can be set to 'true' or 'false' to ensure the prototype performs the desired functions on the given track. The following link is where our GitHub repository can be found: https://github.com/RubayetKamal/Autonomous-Line-Following-Vehicle-Prototype

```
29
30  bool whiteLine = true;
31
```

Fig. 22. Code Snippet

A. Line Follower

When one IR sensor detects black, such as the left sensor, the autonomous vehicle turns right. This is achieved by increasing the speed of the left motor while simultaneously decreasing the speed of the right motor. This is shown in Fig. 23. The turning behavior to the left is implemented in the same manner, with the right motor speeding up and the left motor slowing down.

B. Object Management

The car must stop when it detects an object that is within 3 cm. It is shown in Fig. 24. This pause allows the color sensor to achieve a more accurate detection rate. If the sensor fails to detect an object or if the distance to the object is greater than 3 cm, the car continues moving while the sensor keeps detecting in the meantime. It should be noted that the color sensor starts evaluating only when an object is detected.

C. Colour Evaluation

As shown in Fig.25, the prototype reacts differently depending on the color of the objects it detects. The prototype will overtake the obstacle while detecting red, on the other hand, if the color of object is blue, the car will implement a parking procedure, if it is any other color, the car will turn 180 degrees.

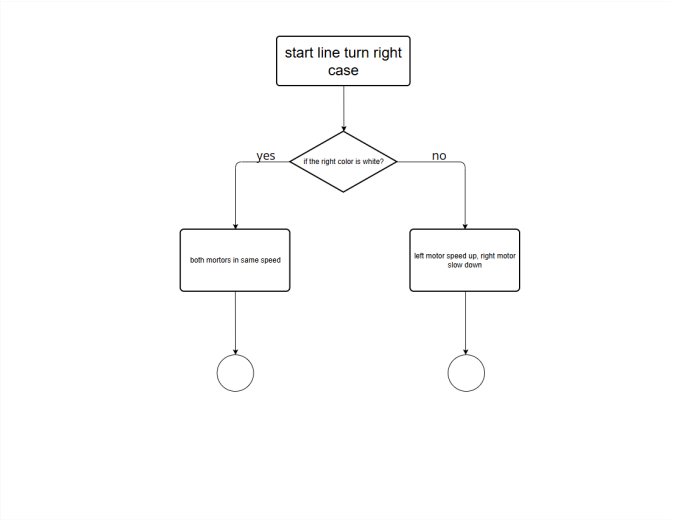

Fig. 23. Line Follower

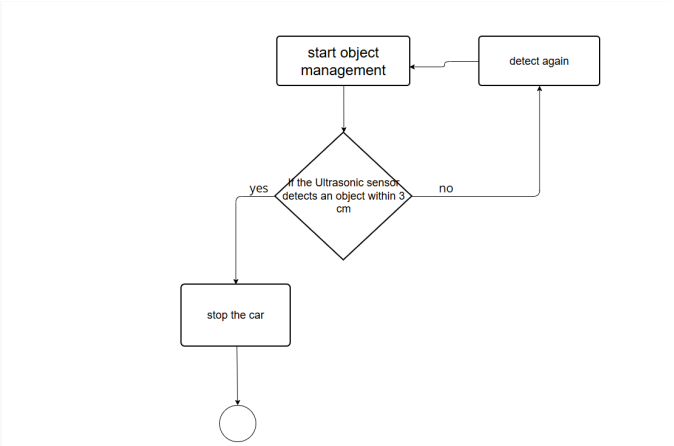

Fig. 24. Object Management

D. Overtake

Once the prototype detects red color, it will reverse. After this, the ultrasonic sensor sends signals again to ascertain whether there is an object within 100 cm. If it is true for this case, the car will turn left go forward then turn right until the IR sensor detects the line again. On the other hand, if it is false, the prototype will turn 180 degrees. This procedure is same as the subsequent case for Parking behavior. This is shown in Fig. 26.

E. Parking

As shown in Fig. 27, parking case is activated when the color sensor detects blue color. When blue is detected,The prototype will first reverse, then turn left 45 degrees, reverse and stop. In this case, if the ultrasonic sensor detects any object, the car will drive forward until detecting the line, otherwise the car will be in an idle state.

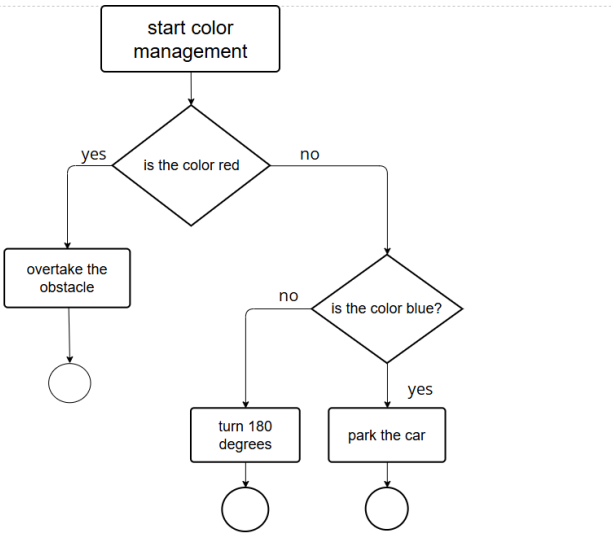

Fig. 25. Color Evaluation

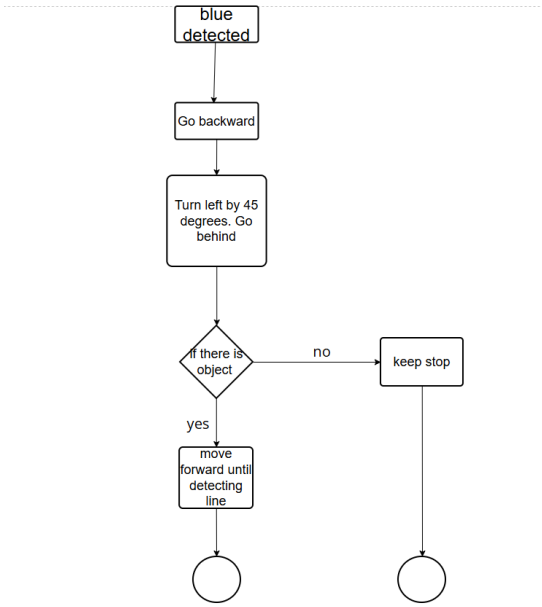

Fig. 27. Parking

Fig. 28.

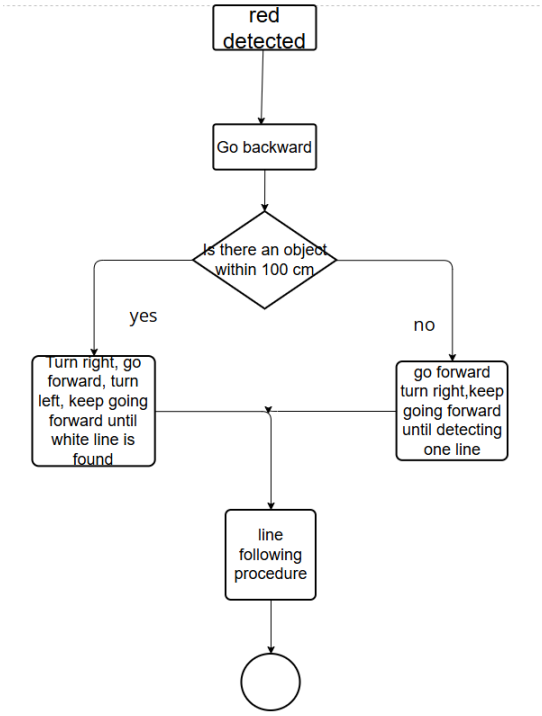

Fig. 26. Overtake

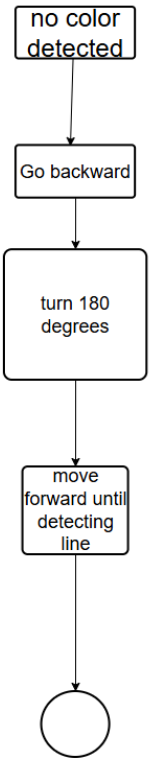

Fig. 28. U-turn

F. U-Turn

This mode is activated when the color sensor of the prototype sees any color apart from red and blue. When this occurs , the car turns 180 degrees and continues moving as shown in

VI. Simulation

All scenarios were simulated using UPPAAL-5.0.0. 4 templates were created: MyAuto, ObjectManagement, LeftHandle and RightHandle. MyAuto shows the overall picture of what the vehicle does. LeftHandle and RightHandle represent left and right line sensors respectively meaning the car will turn left if LeftHandle is triggerred and vice versa. Additionally, Object Management is the integration of the Ultrasonic Sensor with Colour Sensor. Fig. 29. is a screenshot of a simulation scenario in UPPAAL 5.0.0 [14].

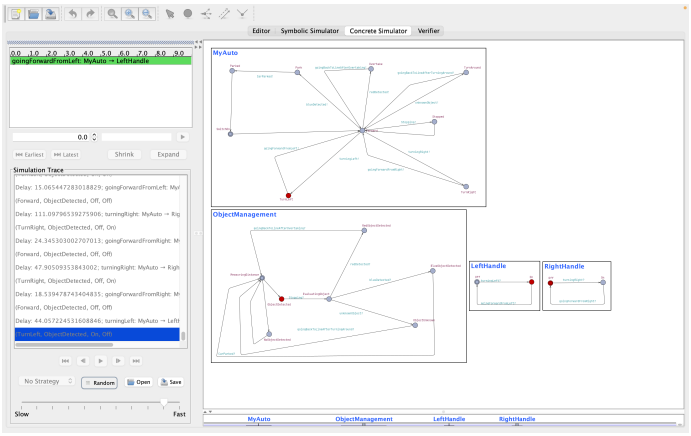

Fig. 29. Simulation in UPPAAL-5.0.0

VII. Result

A. Complete Design

Fig.30. shows the final prototype after everything had been successfully connected.

Fig. 30. Complete Look

B. Line Follower

Fig. 31. shows the car has been successfully programmed to follow line in a black track with white line.

C. Object Management

The Fig.32. shows that the car stops successfully when it detects an obstacle before starting to evaluate the colour.

D. Overtake

Fig. 33. shows the reaction of the car when it detects a red obstacle; the car reverses, turns left 45 degrees, goes forward, turns right at an angle and then keeps moving forward until it finds a line again.

Fig. 31. Car Following Line

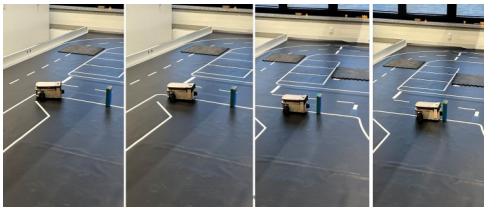

Fig. 32. Car Detecting Obstacle

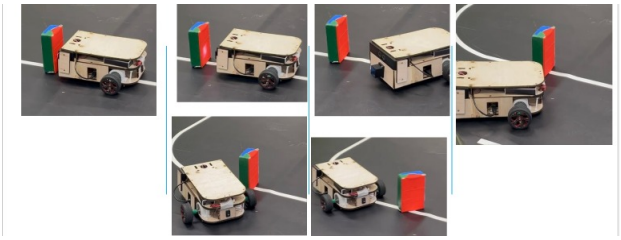

Fig. 33. Car Overtaking if Obstacle is red coloured

E. Parking

Fig. 34. below shows the vehicle parks if it sees a blue object within it's range. First it goes behind a little bit after which it turns right by approximately 45 degrees and then goes behind. It remains in this state until it sees an object within 100 cm from the front.

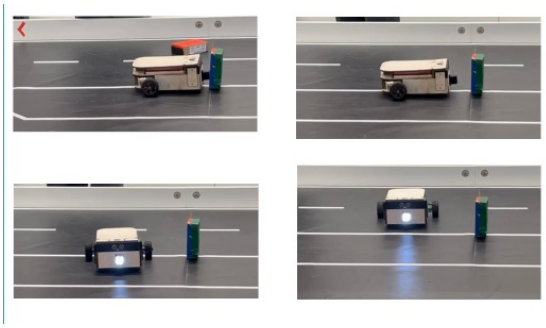

Fig. 34. Car to park if object is Blue

F. U-Turn

If the colour of the object is neither red nor blue, the vehicle will take a U-turn and start moving in the opposite direction. It does so by rotating for as long as it sees the white line again. This is shown in Fig. 35.

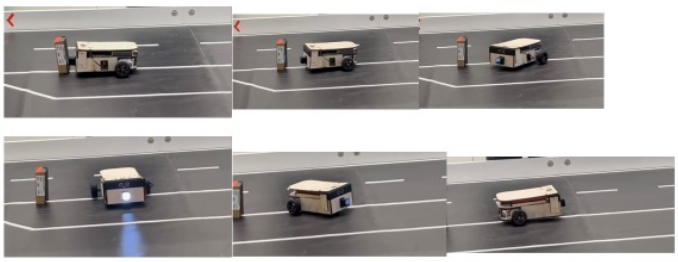

Fig. 35. Car taking U-turn for unknown coloured object

VIII. Conclusion

In this paper, we presented a prototype of a line-following autonomous vehicle capable of making self-decisions based on the color of obstacles. The results indicate that the line-following vehicle meets all the key requirements.

As part of future work, we plan to incorporate two more DC motors to replace the ball-bearing front wheel. Additionally, we aim to use an extra HCSR-04 ultrasonic sensor at the back. Both ultrasonic sensors will be connected to a servo motor, allowing the vehicle to monitor its surroundings and make more consistent decisions. To accommodate these new features, we plan to use a different microcontroller with more digital and analog pins.

Acknowledgment

This project has been possible solely because of the support of tools and expertise provided by Hamm-Lippstadt University of Applied Sciences. Special mention to Prof. Dr. S. Henkler and Mr. G. Wahrmann for their continuous guidance, dedication and feedback enabling us to to continually improve.

Appendix

All authors or team mates have contributed equally in ensuring the success of this project.

References

[1] T. Weilkiens, TotalBoox, and TBX, Systems Engineering with SysML/UML. Elsevier Science, 2011.
[2] "Online productivity suite," Visual Paradigm - Online Productivity Suite, https://online.visual-paradigm.com/ [Accessed Jun. 20, 2024].
[3] Solidworks 2024. Accessed: May 20, 2024. [Online]. Available: https://www.solidworks.com/
[4] Autodesk Fusion 360. (2024) Accessed: June 10, 2024. [Online]. Available: https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription
[5] Colorimeter,[Online].Available:https://www.joy-it.net/files/files/Produkte/SEN-Color/SEN-Color_Manual_2021-08-18.pdf [Accessed Jun. 27, 2024].
[6] Joy-IT, "KY-050 ultrasonic distance sensor," SensorKit,[Online].Available: https://sensorkit.joy-it.net/en/sensors/ky-050 [Accessed Jun. 27, 2024].
[7] TinkerCAD, https://www.tinkercad.com/dashboard [Accessed Jun. 20, 2024]
[8] "Uno R3," Arduino,[Online].Available:https://docs.arduino.cc/hardware/uno-rev3/. [Accessed Jun. 27, 2024]
[9] "Buy Conrad Energy Scale Model Battery Pack (LIPO) 7.4 V 5500 mah no. of cells: 2 20 C Softcase XT90," Conrad Electronic,[Online].Available:https://www.conrad.com/en/p/conrad-energy-scale-model-battery-pack-lipo-7-4-v-5500-mah-no-of-cells-2-20-c-softcase-xt90-1344152.html [Accessed Jun. 27, 2024].
[10] DC Motor, https://www.handsontec.com/dataspecs/775-Motor.pdf [Accessed Jun. 21, 2024]
[11] Handson Technology, "L298N motor driver module," Components101,[Online]. Available:https://components101.com/modules/l293n-motor-driver-module [Accessed Jun. 27, 2024].
[12] Joy-IT, "KY-033 linetracking sensor," SensorKit, https://sensorkit.joy-it.net/en/sensors/ky-033 [Accessed Jun. 27, 2024].
[13] Arduino IDE 2.3.2. Accessed: June 28, 2024. Available: https://www.arduino.cc/en/software
[14] UPPAAL 5.0.0, Accessed: June 20, 2024. Available: https://uppaal.org/downloads/