Prototyping and Systems Engineering Hochschule Hamm-Lippstadt Summer Semester 2023 Group A5 20th of June 2023

1st Deundara Palliya Pisula Mayan Guruge
deundara-palliya-pisula-mayan.guruge@stud.hshl.de

2nd Masrur Jamil
masrur-jamil.prochchhod@stud.hshl.de

3rd Mehedi Hasan
Mehedi.hasan@stud.hshl.de

4thJiban-Ul Azam Chowdhury Shafin
jiban-ul-azam-chowdhury.shafin@stud.hshl.de

***Abstract*—This document serves as a comprehensive documentation of the Prototype created by our team. The purpose of this paper is to provide an in-depth exploration of the project, including the hardware design, electronic components utilized, and the algorithm implemented to ensure the robot meets its intended objectives. Moreover, it covers all the tasks, experiments, and tests conducted throughout the entire semester.**

I. INTRODUCTION

The paper focuses on the prototype built by our team. The main intention of this project was to come up with an autonomous car that will drive on a given track, detect and avoid obstacles depending on their color. After an obstacle is detected the car will either push through the obstacle or move around the obstacle depending on the obstacle's color.

II. TASKS 1 AND 2

As an initial step we were assigned the task of creating system engineering models based on the following requirements.

- Develop an autonomous vehicle that can drive autonomously on a given track, based on line detection.
- Being able to detect obstacles, remove and drive around obstacles based on identifying it's colour.
- Optimizing the speed
- Different routings, driving in an oval or eight pattern.
- Parking

For Task 2 we had to develop a first prototype.This prototype was to be developed using TinkerCAD and had to showcase the overall circuit design and outer-design of our autonomous car.

III. TASK 3

On our Last Task, before we started our lab practicals on developing our autonomous car, we had to finalize our SysML diagrams and showcase our finished TinkerCad designs and also create a paper prototype of our autonomous car design.

IV. SYSTEM MODELLING DIAGRAMS

A. Block Diagram

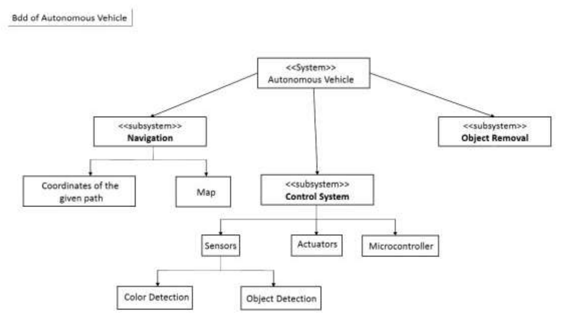

Bdd of Autonomous Vehicle
<<System>> Autonomous Vehicle
<<subsystem>> Navigation
<<subsystem>> Object Removal
Coordinates of the given path
Map
<<subsystem>> Control System
Sensors
Actuators
Microcontroller
Color Detection
Object Detection

Fig. 1. Block Diagram

Each block in the diagram represents a specific function or component of the autonomous vehicle system. The arrows indicate the flow of information or control between the blocks. This block diagram gives an overview of how the different subsystems and components work together to enable the autonomous vehicle to follow a black line, detect objects, and navigate around them based on color detection.

Autonomous Vehicle: This is the main block representing the entire autonomous vehicle system. It encompasses all the other blocks and is responsible for controlling the overall functioning of the vehicle.

Navigation: This block focuses on the vehicle's navigation capabilities. It receives information from two sources: "Coordinates of the Given Path" and "Map." These inputs help the navigation subsystem determine the vehicle's desired path and destination.

Coordinates of the Given Path: This block provides specific coordinates or instructions regarding the desired path the vehicle should follow. It helps the navigation subsystem understand the route it needs to take.

Map: The "Map" block represents a map of the environment where the vehicle operates. It provides essential information about the black line is present. The navigation subsystem uses this information to plan the vehicle's path.

Control System: The control system subsystem is responsible for managing the vehicle's movements and actions based on the inputs received from various sources. Sensors: This block represents the sensors used by the autonomous vehicle. Sensors gather data about the surroundings, such as detecting objects and detecting colors. They provide important information for decision-making.

Actuators: Actuators are devices responsible for converting signals from the control system into physical actions. Examples include motors that control the vehicle's movement, steering mechanisms, and braking systems.

Microcontroller: The microcontroller is the brain of the control system. It receives inputs from the sensors, processes them, and sends commands to the actuators. It handles the decision-making process and controls the vehicle's behavior.

Color Detection: This block represents the functionality of detecting colors. The sensor(s) feed data to this block, which analyzes the color information to make decisions. It may, for example, help the vehicle identify the color of the object.

Object Detection: This block focuses on detecting objects in the vehicle's path. The sensor(s) provide input to this block, which analyzes the data and identifies any obstacles or objects that the vehicle needs to avoid or navigate around.

Object Removal: This block represents the subsystem responsible for handling object removal or avoidance. It receives input from the object detection block and makes decisions on how to navigate around or remove objects from the vehicle's path.

B. Requirement Diagram

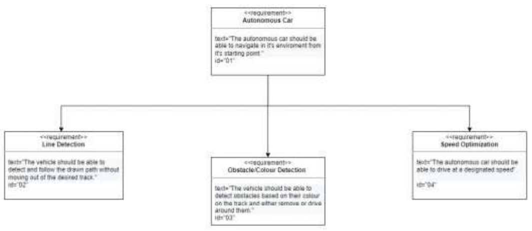

<<requirement>>
Autonomous Car
text="The autonomous car should be able to navigate in it's enviroment from it's starting point."
id="01"
<<requirement>>
Line Detection
text="The vehicle should be able to detect and follow the drawn path without moving out of the desired track."
id="02"
<<requirement>>
Obstacle/Colour Detection
text="The vehicle should be able to detect obstacles based on their colour on the track and either remove or drive around them."
id="03"
<<requirement>>
Speed Optimization
text="The autonomous car should be able to drive at a designated speed"
id="04"

Fig. 2. Requirement Diagram.

This requirement diagram gives an overview of the key objectives and functionalities that need to be fulfilled by the autonomous vehicle system. It outlines the core requirements related to navigation, line detection, obstacle detection, color detection, and speed optimization.

Autonomous Car: This is the main requirement block representing the overall goal of the autonomous car system. The text associated with this block states that the car should be able to navigate in its environment from its starting point and run itself till the end point. This requirement sets the primary objective for the autonomous car system.

Line Detection: This block represents the requirement for line detection. The text associated with this block states that the vehicle should be able to detect and follow the drawn path without moving out of the desired track. This requirement focuses on the car's ability to track and follow a black line accurately.

Obstacle/Color Detection: This block represents the requirement for obstacle and color detection. The text associated with this block states that the vehicle should be able to detect obstacles on the track and either remove the obstacle or drive around it based on its color. This requirement emphasizes the car's capability to identify and handle objects in its path using color detection.

Speed Optimization: This block represents the requirement for speed optimization. The text associated with this block states that the autonomous car should be able to drive itself at a designated speed. This requirement focuses on the car's ability to optimize its speed while navigating the environment.

C. State Machine Diagram

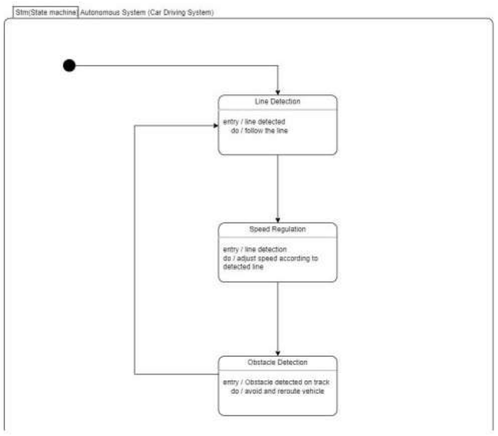

Stm(State machine) Autonomous System (Car Driving System)
Line Detection
entry / line detected
do / follow the line
Speed Regulation
entry / line detection
do / adjust speed according to detected line
Obstacle Detection
entry / Obstacle detected on track
do / avoid and reroute vehicle

Fig. 3. State Machine Diagram.

The state machine diagram describes the different states the autonomous vehicle can be in and how it transitions between these states based on line detection and obstacle detection. It starts by detecting the line, then regulates its speed accordingly, and if an obstacle is detected, it reroutes to avoid it. The vehicle continuously loops back to line detection to ensure it remains on track.

Start: The state machine diagram begins with the "Start" state, which represents the initial state of the autonomous vehicle. When the vehicle starts, it enters this state.

Line Detection: From the "Start" state, the vehicle transitions to the "Line Detection" state. This state represents the process of the vehicle detecting the black line on the track. The entry action "line detected" signifies that the vehicle has successfully detected the line. The do action "follow the line" indicates that the vehicle will start following the detected line.

Speed Regulation: From the "Line Detection" state, the vehicle transitions to the "Speed Regulation" state. This state focuses on adjusting the vehicle's speed according to the detected line. The entry action "line detection" shows that the vehicle has detected the line again (maybe it keeps checking while following the line). The do action "adjust speed according to the detected line" means that the vehicle will modify its speed based on the detected line's position or other factors.

Obstacle Detection: From the "Speed Regulation" state, the vehicle transitions to the "Obstacle Detection" state. This state represents the process of the vehicle detecting an obstacle on the track. The entry action "obstacle detected on track" indicates that the vehicle has encountered an obstacle. The do action "avoid and reroute vehicle" means that the vehicle will take measures to avoid the obstacle and find an alternative route.

Looping Back to "Line Detection": After the "Obstacle Detection" state, the vehicle loops back to the "Line Detection" state. This loop allows the vehicle to continuously detect and follow the black line while also checking for obstacles periodically.

D. Activity Diagram

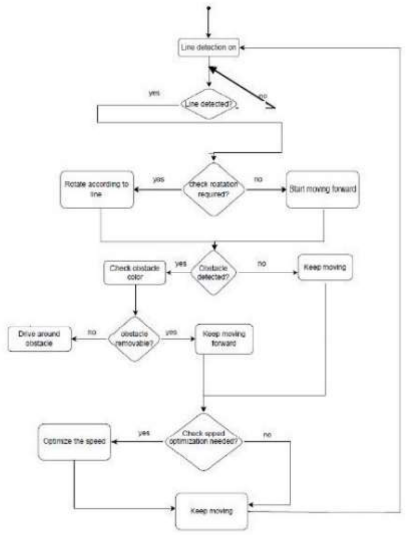


Fig. 4. Activity Diagram.

Start: The activity diagram begins with the "Start" activity, which represents the starting point of the autonomous vehicle. The vehicle is started, and the process moves forward from this point.

Line Detection: From the "Start" activity, the process moves to the "Line Detection" activity. Here, the vehicle turns on the line detection functionality. If the line is not detected, the process continues with the "Continue Searching for Line" activity, indicating that the vehicle will keep searching for the line.

Line Detected: If the line is detected, the process moves to the "Line Detected" activity. Here, the vehicle checks if any rotation is required based on the detected line. If rotation is needed, the process moves to the "Rotate According to Line" activity, indicating that the vehicle will rotate to align itself with the line. If no rotation is required, the process moves to the "Move Forward" activity, indicating that the vehicle will start moving forward.

Obstacle Detection: From the "Move Forward" activity, the process moves to the "Obstacle Detection" activity. Here, the vehicle checks if any obstacles are detected. If no obstacles are detected, the process goes back to the "Move Forward" activity, indicating that the vehicle will continue moving forward.

Obstacle Detected: If an obstacle is detected, the process moves to the "Obstacle Detected" activity. Here, the vehicle checks the color of the obstacle and decides if it is removable or not. If the obstacle is removable, the process goes back to the "Move Forward" activity, indicating that the vehicle will continue moving forward to remove the obstacle. If the obstacle is not removable, the process moves to the "Drive Around Obstacle" activity, indicating that the vehicle will navigate around the obstacle.

Speed Optimization: From the "Move Forward" or "Drive Around Obstacle" activities, the process moves to the "Speed Optimization" activity. Here, the vehicle checks if speed optimization is needed. If speed optimization is required, the process moves to the "Optimize Speed" activity, indicating that the vehicle will adjust its speed accordingly. If speed optimization is not needed, the process goes back to the "Move Forward" or "Drive Around Obstacle" activities, indicating that the vehicle will continue moving at the current speed.

Looping Back: After the "Speed Optimization" activity, the process loops back to the "Line Detection" activity. This loop allows the vehicle to continuously perform line detection, obstacle detection, and speed optimization while moving forward or navigating around obstacles. The process continues until the power button is switched off.

E. Uppaal Model

The Uppaal model consists of three Templates, the whole car system, obstacle detection and line detection. The Car template consists of two states line detection and obstacle detection.
In the line detection state if the line is not detected, the car reroutes itself till the line is detected and comes back to the path. When the car is on the line it keeps moving on the path till an obstacle is detected. If an obstacle is detected the Uppaal model depicts that the car will reroute itself away from

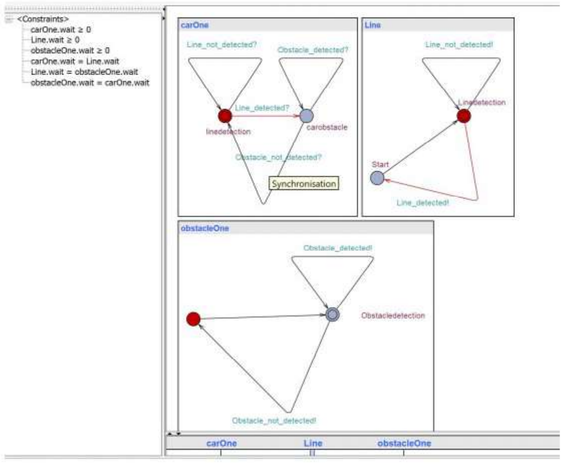
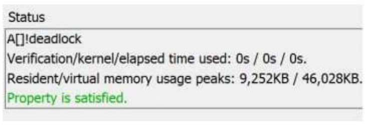


Fig. 5. Uppaal Model.

the obstacle path and move away from the obstacle and come back to the path and keep moving forward on a loop.
After successfully creating the Uppaal model, it was verified for errors and no deadlocks were found.



```
Status
A[]!deadlock
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 9,252KB / 46,028KB.
Property is satisfied.
```

Fig. 6. Deadlock Verification.

V. HARDWARE COMPONENTS USED FOR THE PROTOTYPE

The Prototype incorporates several electronic components, each serving specific functions and possessing distinct properties. Here is a summary of these components:

- Arduino Uno.
- Motor Driver (L298N)
- 2 IR Sensors.
- Colour Sensor.
- 2 Geared Motors
- Battery
- Ultrasonic sensor

A. Arduino Uno

Fig. 7. Arduino Uno

The Arduino Uno was selected as the primary micro controller for the Prototype, responsible for gathering sensor data, processing it, and initiating appropriate actions through the actuators for navigation purposes. This development board incorporates the Atmega328 microcontroller from Atmel's megaAVR family, featuring an adapted architecture with an 8-bit RISC processor core. The board is equipped with a total of 14 digital GPIO pins, of which 6 support PWM output, along with 6 analog input pins. The communication interface with the PC is established through the USB connection.

B. Motor Driver (L298N)

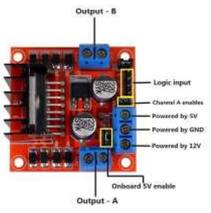


Fig. 8. Motor Driver

We were opted to use the SBC-MotoDriver2 (L298N) to regulate and power the two geared motors incorporated into our prototype. To control the motors, a driver IC is required. The driver IC acts as a switch, receiving signals from the Arduino Uno. When a high signal is received, the driver IC activates the switch, providing the motors with the required voltage for rotation. The SBC-MotoDriver2 offers two enable inputs, enabling the device to be enabled or disabled as needed.

C. 2 IR Sensors

Fig. 9. IR Sensors

Two infrared (IR) sensors are used in the car to detect and follow the line. The IR sensors are placed under the car and emit infrared light onto the surface. They then detect the reflection of the emitted light by analyzing two states high and low. In this way, the IR sensors can determine whether the car is positioned on the line or off the line. These sensors play an important role in keeping the car in the line.

D. Colour Sensor

A (TCS3200) color sensor was used to detect the color of the obstacles on the track and based on the color, either avoid the obstacle and move around the object or push through the obstacle.

Fig. 10. Colour Sensor

E. Geared Motors

Fig. 11. Geared Motors

In the prototype, we have integrated the Modelcraft RB 35 gearbox with motor. This motor model is capable of delivering a torque of 1:30, indicating a significant rotational force. It also has the speed reduction at the same rate as the torque, meaning the speed of the motor is reduced proportionally as the torque increases, maintaining a balanced performance.

F. Battery

Fig. 12. Battery

A rechargeable Polymer Li-Ion Battery is used in the prototype. This particular battery supplies the necessary power to the Arduino board, both motors and the sensors.

G. Ultrasonic Sensor

An ultrasonic sensor is used to detect obstacles in front of the car. It sends sound waves and measures the time taken for the sound waves to bounce back after hitting an obstacle. By calculating the distance based on the time of total travel, the ultrasonic sensor can determine if there is an obstacle near the car.

VI. PROTOTYPE DESIGN

A. TinkerCad

Tinkercad proved to be an invaluable tool for our car design project, offering a multitude of benefits that greatly facilitated

Fig. 13. Ultrasonic Sensor

our creative process. One of its most notable advantages is its accessibility as a web-based platform. By eliminating the need for complex software installations or configurations, Tinkercad provided us with a seamless experience. The user-friendly nature of Tinkercad played a pivotal role in our project. Its intuitive and straightforward interface made it an ideal choice, particularly for beginners in the realm of 3D modeling. We quickly grasped its functionalities, which enabled us to dive right into the design process with confidence. The ability to manipulate objects through scaling, rotating, and precise alignment empowered us with complete control over the shape and dimensions of our car design.

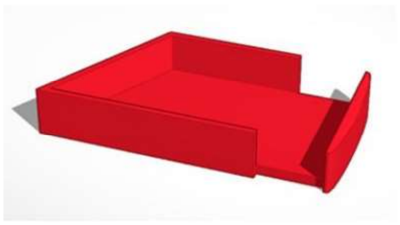


Fig. 14. 3D design in tinkerCAD

Tinkercad's diverse set of tools further facilitated our design journey. We leveraged these tools effectively to bring our car to life, shaping it exactly as we envisioned. For instance, the pointy side of our car served a specific purpose driven by our project goals. With the aim of overcoming obstacles, we deliberately designed the car with a pointy side to function as a means of removing obstacles. In cases where the car might fail to detect an obstacle, its pointy side would come into play, allowing it to effectively remove the obstacle and continue its path.

B. Rhinoceros

For the 2D design phase of our car project, we made the strategic decision to utilize Rhinoceros, a versatile software renowned for its precision and capabilities in creating detailed 2D designs. This choice was motivated by several factors, which we found advantageous for our specific project requirements. Initially, we contemplated utilizing 3D printing for our design. However, upon careful consideration, we realized that opting for 2D printing would provide us with greater flexibility and opportunities to innovate our design. One crucial aspect was the ease of cancellation or modification. With 2D printing, we could easily make changes or even cancel the printing process if necessary. In contrast, 3D printing would lock us into a particular design once initiated, limiting our ability to iterate or make significant alterations. This flexibility was paramount in our pursuit of an optimal car design.

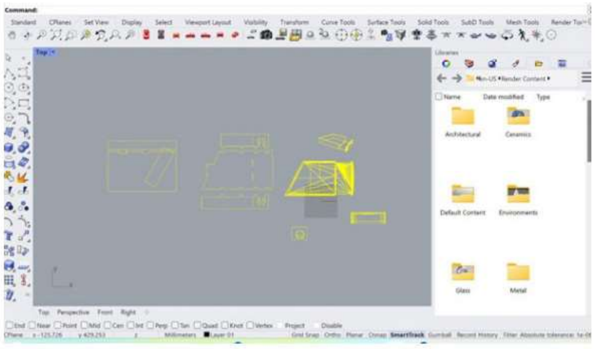


Fig. 15. Rhinoceros

Time efficiency also played a vital role in our decision-making process. We recognized that 3D printing could be considerably time-consuming, potentially leading to delays in our project timeline. Conversely, 2D printing and laser cutting offered a more streamlined workflow. By utilizing Rhinoceros, we could create a detailed 2D drawing of our car design efficiently. This allowed us to focus our time and effort on refining and perfecting the design without the added complexity and time constraints of 3D printing. Rhinoceros, with its robust set of features and precise design capabilities, proved to be an ideal tool for our 2D design requirements. Its user-friendly interface and comprehensive toolkit enabled us to create intricate and accurate 2D representations of our car design. This detailed 2D drawing served as the foundation for the subsequent laser cutting process, where we could transform our design into a physical shape.

C. Design and Schematics

Once our car design reached its completion, we proceeded to the next crucial step in our project: finalizing the prototype. To ensure a seamless integration of hardware components, we embarked on designing a comprehensive schematic that outlined the connections necessary for our design. This schematic served as a blueprint, guiding us in establishing the appropriate connections between various components.

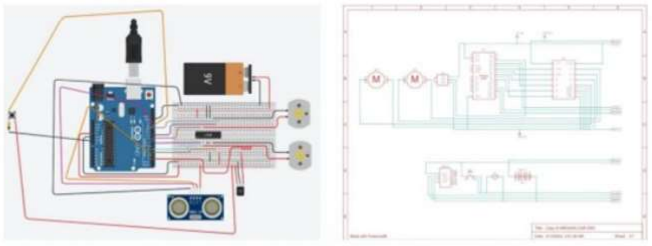


Fig. 16. Schematics of the prototype

To validate the feasibility and functionality of our schematic, we initially ran it through the Tinkercad simulation. This step allowed us to assess whether the connections and configurations met the required conditions and specifications. By simulating the design in Tinkercad, we gained valuable insights into how the car would operate based on the implemented code and the chosen ports for connecting the hardware components. This simulation provided us with a practical preview of the car's behavior, enabling us to identify any potential issues or areas for improvement.
The schematic itself played a crucial role in constructing our prototype. It visually represented the necessary connections between ports, facilitating the assembly process and ensuring that each component was correctly integrated into the design. By referring to the schematic, we were able to make precise and accurate connections, resulting in a well-structured and functional prototype.
Furthermore, the Tinkercad simulation was instrumental in helping us understand how the car would operate in practice. By running the simulation with the implemented code, we could observe how the car followed the instructions and responded to various inputs. This allowed us to fine-tune the code and make any necessary adjustments to ensure the desired performance of the car. In summary, the completion of our design led us to finalize the prototype by creating a comprehensive schematic of the required connections. The Tinkercad simulation served as a crucial testing ground, providing us with valuable insights into the behavior of the car based on the implemented code and hardware connections. The schematic, along with the simulation, guided us in constructing the prototype and ensuring that it operated according to our expectations. This iterative process of designing, simulating, and implementing helped us refine our design, leading to a functional and well-executed prototype.

D. Final Prototype

The combination of our design process resulted in the creation of the final prototype, which seamlessly incorporated the various components and designs developed using Tinkercad, Rhinoceros, Tinkercad simulation, and schematics. With the completion of laser cutting, we progressed to the crucial stage of connecting the ports, bringing the prototype closer to its intended form. The final prototype represented the harmonious integration of our design iterations across different platforms. The initial stages involved utilizing Tinkercad to create a 3D representation of our car design, allowing us to visualize and refine its structure. Concurrently, Rhinoceros provided us with the tools and capabilities to develop detailed 2D designs specifically tailored for laser cutting, enabling us to shape the physical components of the car with precision. Throughout the process, the Tinkercad simulation played a pivotal role in evaluating the performance of our design. By simulating the car's behavior based on the implemented code and hardware connections, we gained valuable insights into its functionality, identifying areas for improvement and making necessary adjustments.

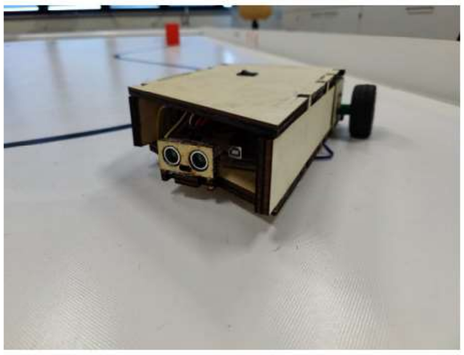


Fig. 17. Final Prototype

The schematics served as a crucial blueprint, guiding us in establishing the required connections between various ports and components. By referring to the schematics, we ensured the accurate and systematic integration of different elements, contributing to the overall functionality and coherence of the prototype. With the completion of laser cutting and the establishment of the necessary connections, we arrived at the final stage of the design process. We remained steadfast in our commitment to the original design, adhering to the vision and plan that we had set from the beginning. By staying true to our initial design concept and working diligently alongside it, we maintained consistency and coherence throughout the entire process.
In summary, the final prototype was the culmination of our collective efforts and the successful integration of designs developed through Tinkercad, Rhinoceros, Tinkercad simulation, and schematics. Laser cutting, connection of ports, and the unwavering commitment to our original design brought the prototype to its planned form. The combination of these design elements, along with our meticulous approach, resulted in a cohesive and functional final prototype that aligned with our initial vision.

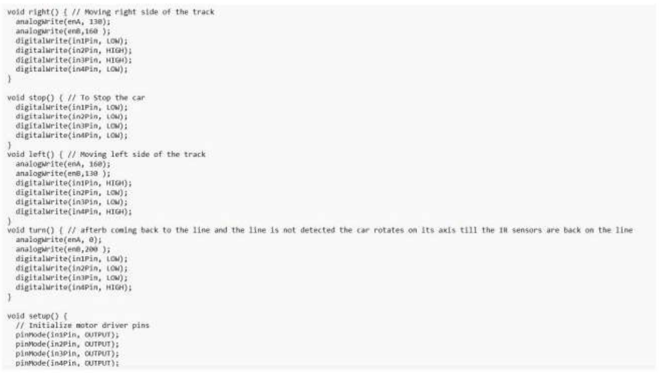
VII. Programming the Assembled Prototype

1. void forward(): This function is responsible for moving the car forward. It sets the appropriate signals on the motor driver pins to achieve forward motion. The specific steps involved in the forward movement are as follows:

- digitalWrite(in1Pin, LOW): Sets the in1Pin to a LOW state, ensuring that one side of the motor is turned off..
- digitalWrite(in2Pin, HIGH): Sets the in2Pin to a HIGH state, enabling the other side of the motor to rotate in a forward direction.
- digitalWrite(in3Pin, LOW): Sets the in3Pin to a LOW state, turning off the other side of the motor.
- digitalWrite(in4Pin, HIGH): Sets the in4Pin to a HIGH state, allowing the other side of the motor to rotate forward.

By activating the appropriate pins and deactivating the opposite pins, the function enables the car to move forward smoothly. 2. void forwardU(): This function is used when the car is coming back to the line and needs to move forward in its last forward direction. It provides the same functionality as the forward() function, but it also adjusts the speed of the car by using the analogWrite function to set the appropriate values for enA and enB pins. This allows the car to maintain a consistent speed while moving forward. 3. void right(): The right() function is responsible for moving the car to the right side of the track. It adjusts the signals on the motor driver pins to achieve a right turn. The steps involved in the right movement are as follows:

- analogWrite(enA, 130): Sets a specific speed value for the enA pin using analogWrite, controlling the speed of one side of the motor.
- analogWrite(enB, 160): Sets a specific speed value for the enB pin using analogWrite, controlling the speed of the other side of the motor.
- digitalWrite(in1Pin, LOW): Sets the in1Pin to a LOW state, turning off one side of the motor.
- digitalWrite(in2Pin, HIGH): Sets the in2Pin to a HIGH state, allowing the other side of the motor to rotate in a forward direction.
- digitalWrite(in3Pin, HIGH): Sets the in3Pin to a HIGH state, enabling the other side of the motor to rotate in a forward direction.
- digitalWrite(in4Pin, LOW): Sets the in4Pin to a LOW state, turning off the other side of the motor. By adjusting the speed and activating the appropriate pins, the function facilitates a smooth right turn for the car.



```
void right() { // Moving right side of the track
  analogWrite(enA, 130);
  analogWrite(enB,160 );
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, HIGH);
  digitalWrite(in3Pin, HIGH);
  digitalWrite(in4Pin, LOW);
}

void stop() { // To Stop the car
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, LOW);
  digitalWrite(in3Pin, LOW);
  digitalWrite(in4Pin, LOW);
}
void left() { // Moving left side of the track
  analogWrite(enA, 160);
  analogWrite(enB,130 );
  digitalWrite(in1Pin, HIGH);
  digitalWrite(in2Pin, LOW);
  digitalWrite(in3Pin, LOW);
  digitalWrite(in4Pin, HIGH);
}
void turn() { // afterb coming back to the line and the line is not detected the car rotates on its axis till the IR sensors are back on the line
  analogWrite(enA, 0);
  analogWrite(enB,200 );
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, LOW);
  digitalWrite(in3Pin, LOW);
  digitalWrite(in4Pin, HIGH);
}

void setup() {
  // Initialize motor driver pins
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
```

Fig. 18. Coding

4. void stop(): The stop() function is responsible for stopping the car by turning off all the signals on the motor driver pins. It sets all the pins (in1Pin, in2Pin, in3Pin, in4Pin) to a LOW state, effectively stop both sides of the motor and bringing the car to a complete stop situation. 5. void left(): The left() function is similar to the right() function, but it enables the car to move to the left side of the track. It adjusts the signals on the motor driver pins to achieve a left turn. The specific steps involved are opposite of the right() . By adjusting the speed and activating the appropriate pins, the function enables the car to make a smooth left turn. 6. void turn(): The turn()

function is used when the car has come back to the line and the line is not detected by the IR sensors. It causes the car to rotate on its axis until the IR sensors detect the line again. The specific steps involved are:

- analogWrite(enA, 0): Sets the enA pin to a 0 value using analogWrite, effectively stopping one side of the motor.
- analogWrite(enB, 200): Sets a specific speed value for the enB pin using analogWrite, controlling the speed of the other side of the motor.
- digitalWrite(in1Pin, LOW): Sets the in1Pin to a LOW state, turning off one side of the motor.
- digitalWrite(in2Pin, LOW): Sets the in2Pin to a LOW state, turning off the other side of the motor.
- digitalWrite(in3Pin, LOW): Sets the in3Pin to a LOW state, turning off the other side of the motor.
- digitalWrite(in4Pin, HIGH): Sets the in4Pin to a HIGH state, allowing the other side of the motor to rotate in a forward direction.

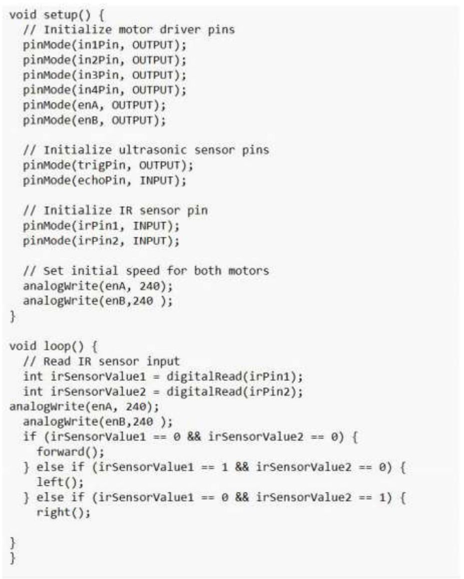
By adjusting the speed and activating the appropriate pins, the function enables the car to rotate until the IR sensors detect the line. Setup Function The setup function is called once during the initialization phase of the program. It is responsible for configuring the pin modes for the motor driver, ultrasonic sensor, and IR sensors. By setting the appropriate pin modes (input or output), the software ensures that the signals can be properly received or transmitted. Additionally, the setup function sets the initial speed values for the car's motors using the analogWrite function. These initial speed values can be adjusted to the specific requirements of the car.

The loop function is the main execution block of the program and runs continuously after the setup function. It implements the core logic of the line-following and obstacle-avoidance and detection behaviors. Here is the overall view of the loop function's operation:

- Read IR Sensor Inputs: The loop function reads the digital inputs from the IR sensors, specifically irSensorValue1 and irSensorValue2. These variables store the current states of the IR sensors, indicating whether the car is positioned on the line or off the line.
- Line-Following Behavior: Based on the inputs from the IR sensors, the code determines the appropriate movement for the car. If both ir sensors detect the white line (irSensorValue1 == 0 and irSensorValue2 == 0), the car moves forward by calling the forward() function. If only the left sensor detects the black line (irSensorValue1 == 1 and irSensorValue2 == 0), the car turns left by calling the left() function. If only the right sensor detects the black line (irSensorValue1 == 0 and irSensorValue2 == 1), the car turns right by calling the right() function.
- Obstacle Avoidance Behavior: The code incorporates obstacle detection using the ultrasonic sensor. It measures the distance to any obstacles by emitting ultrasonic waves and calculating the time it takes for the waves to bounce back. If an obstacle is detected (distance less than 10), the car performs a predefined evasive action. It turns left (left()), moves forward (forward()), turns right (right()), and then moves forward in the last forward direction (forwardU()). These actions are implemented with specific delays (delay()) to ensure proper navigation around the obstacle.
- Line Recovery: After avoiding the obstacle, the car moves forward using forward() function until both IR sensors detect the line before proceeding. When both ir sensor see the line the the loop breaks and execute the next stop() function. And the it goes forward using the forwarU() function then it turns until the right ir sensor (irSensorValue2 == 0) detect the line. It continuously checks the state of the IR sensors in a while loop. Once (irSensorValue2 == 0) sensors detect the line, the loop breaks, and the car resumes the line-following behavior.

The loop function continuously repeats this process, allowing the car to autonomously follow the designated line while avoiding obstacles.

The Line-Following Car with Obstacle Avoidance project demonstrates the successful implementation of a software system that enables a car to autonomously navigate along a specified line while avoiding obstacles. By utilizing a motor driver, ultrasonic sensor, and IR sensors, the car achieves reliable line tracking and obstacle detection capabilities.



```
void setup() {
  // Initialize motor driver pins
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);

  // Initialize ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Initialize IR sensor pin
  pinMode(irPin1, INPUT);
  pinMode(irPin2, INPUT);

  // Set initial speed for both motors
  analogWrite(enA, 240);
  analogWrite(enB,240 );
}

void loop() {
  // Read IR sensor input
  int irSensorValue1 = digitalRead(irPin1);
  int irSensorValue2 = digitalRead(irPin2);
analogWrite(enA, 240);
  analogWrite(enB,240 );
  if (irSensorValue1 == 0 && irSensorValue2 == 0) {
    forward();
  } else if (irSensorValue1 == 1 && irSensorValue2 == 0) {
    left();
  } else if (irSensorValue1 == 0 && irSensorValue2 == 1) {
    right();

}
}
```

Fig. 19. Functions

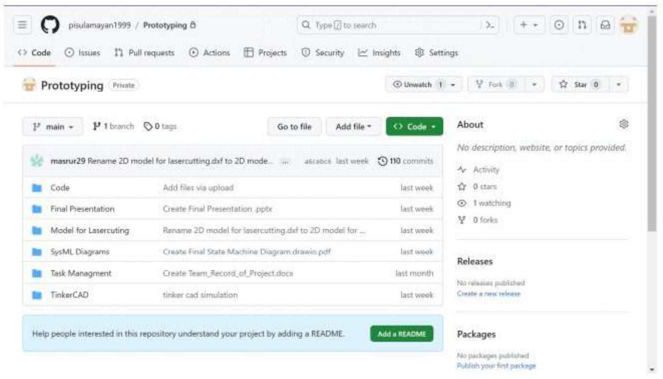
VIII. CONCLUSION

The prototype is successfully built up and is finally able to carry out the following tasks listed below.

- Able to follow track.
- Able to detect obstacles.
- Able to avoid obstacles and move around them and return to back to the track.

IX. LINK TO THE GITHUB RELEASE

https://github.com/pisulamayan1999/Prototyping


pisulamayan1999 / Prototyping

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

Prototyping Private

Unwatch 1 · Fork 0 · Star 0

main · 1 branch · 0 tags · Go to file · Add file · Code

masrur29 Rename 2D model for lasercutting.dxf to 2D mode... ascabcs last week 110 commits

Code	Add files via upload	last week
Final Presentation	Create Final Presentation .pptx	last week
Model for Lasercuting	Rename 2D model for lasercutting.dxf to 2D model for ...	last week
SysML Diagrams	Create Final State Machine Diagram.drawio.pdf	last week
Task Managment	Create Team_Record_of_Project.docx	last month
TinkerCAD	tinker cad simulation	last week

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

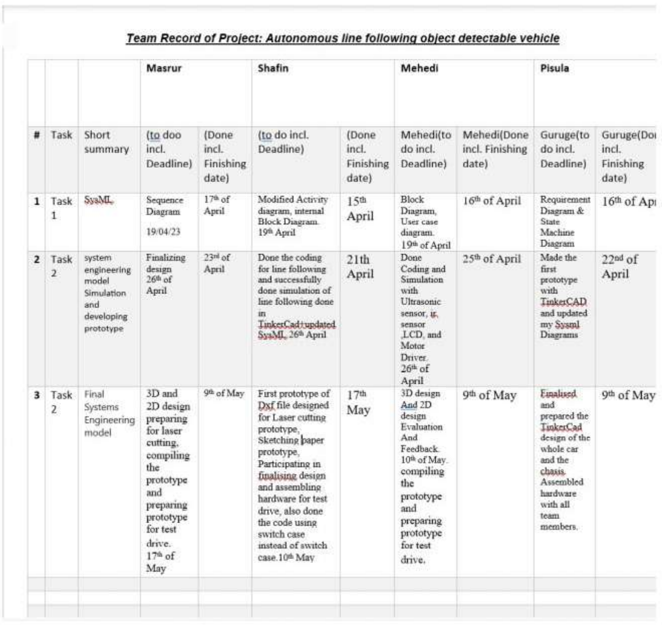
No releases published
Create a new release

Packages

No packages published
Publish your first package

Fig. 20. GitOverview

X. Individual Contribution


Team Record of Project: Autonomous line following object detectable vehicle

			Masrur		Shafin		Mehedi		Pisula	
#	Task	Short summary	(to doo incl. Deadline)	(Done incl. Finishing date)	(to do incl. Deadline)	(Done incl. Finishing date)	Mehedi(to do incl. Deadline)	Mehedi(Done incl. Finishing date)	Guruge(to do incl. Deadline)	Guruge(Do incl. Finishing date)
1	Task 1	SysML	Sequence Diagram 19/04/23	17th of April	Modified Activity diagram, internal Block Diagram. 19th April	15th April	Block Diagram, User case diagram. 19th of April	16th of April	Requirement Diagram & State Machine Diagram	16th of Ap
2	Task 2	system engineering model Simulation and developing prototype	Finalizing design 26th of April	23rd of April	Done the coding for line following and successfully done simulation of line following done in TinkerCad+updated SysML 26th April	21th April	Done Coding and Simulation with Ultrasonic sensor, ir sensor ,LCD, and Motor Driver. 26th of April	25th of April	Made the first prototype with TinkerCAD and updated my Sysml Diagrams	22nd of April
3	Task 2	Final Systems Engineering model	3D and 2D design preparing for laser cutting, compiling the prototype and preparing prototype for test drive. 17th of May	9th of May	First prototype of Dxf file designed for Laser cutting prototype, Sketching paper prototype, Participating in finalising design and assembling hardware for test drive, also done the code using switch case instead of switch case.10th May	17th May	3D design And 2D design Evaluation And Feedback. 10th of May. compiling the prototype and preparing prototype for test drive.	9th of May	Finalised and prepared the TinkerCad design of the whole car and the chasis. Assembled hardware with all team members.	9th of May

Fig. 21. Task Distribution

References

[1] https://mymoodle.hshl.de
[2] https://vdocuments.net/arduino-uno-datasheet569d589fca336.html?page=2
[3] https://app.diagrams.net/
[4] https://www.tinkercad.com/dashboard
[5] https://uppaal.org/
[6] https://www.rhino3d.com/de/

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Guruge, Pisula Mayan

Last Name, First Name

Lippstadt , 20.06.2023

Location, Date

[illegible]

Signature.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Prochchhod, Masrur Jamil

Last Name, First Name

Hamm, 20.06.2023

Location, Date

[illegible]

Signature.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Hasan, Mehedi

Last Name, First Name

Hamm, 20.06.2023

Location, Date

Mehedi Hasan

Signature.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Shafin, Jiban-Ul Azam Chowdhury

Last Name, First Name

Lippstadt, 20.06.2023

Location, Date

Jiban-Ul Azam Chowdhury Shafin

Signature.