Gizmo: The Next Frontier in Autonomous Robotics - Building on Falcon's Legacy

Milestone 2

BIA Capstone Project

MGMT – 6134 – 23F

Prepared For:

Professor Mark Bueno

Professor Mona Abou Taka

Prepared By: **PC51 Technologies**

Wilberto Mejia - 1138669 (S32)

Apple Dela Cruz - 1080609 (S31)

Jhay-Ar Aguilar - 1105965 (S32)

Dean Paul Martin - 1091458 (S32)

Anjeline Sayoc - 1104192 (S32)

Roben Woo - 1080811 (S31)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. PROJECT SUMMARY

1.1 INTRODUCTION

Team Digi Destined initiated the project *Going Into Automation: Building an Arduino-based Intelligent Robot To Avoid Obstacles*. The project focused on developing an autonomous four-wheeled robot using the ESP32 wireless module and the Arduino integrated development environment which enables the robot car to make decisions through perception algorithms as part of the BIA Capstone Project (Summer 2023). The scope of the project was to create Falcon, a four-wheel car, with ESP 32 controller that has the following capabilities: line tracking and obstacle detection and avoidance. Due to time constraints, Team Digi Destined only completed Falcon with line tracking and obstacle detection abilities.

The project opens the door to a new area of knowledge. Integrated Systems and Micro Controllers show the path to AI that can further lead to wide opportunities to the society in fields such as, health, security, and education. The opportunity of getting the experience related to robotics inspired the current team, PC51 Technologies, to improve the robot functionalities and meet current needs in society while Fanshawe provides the steps into this wide and important field. Moving forward, PC51 Technologies aims to expand Falcon's capabilities to include obstacle detection and avoidance, while also introducing a new feature for light tracing. These enhancements will introduce **Gizmo**.

PC51 Technologies embarks on a journey where Falcon's legacy merges seamlessly with Gizmo's potential, creating a bridge between the past and the future. This project is illuminated by the promise of enhanced autonomy - offering solutions to challenges that society may not fully comprehend at this time. The transition from Falcon to Gizmo symbolizes PC51 Technologies' commitment to pushing the boundaries of what this intelligent robot can achieve. This project will also showcase the team's collective effort to keep pace with the rapid advancements in the field and to contribute to the cutting-edge innovations that shape society.

Gizmo, with its newfound capabilities, stands as a testament to the team's dedication to progress. This exploration in the field of robotics represents a significant stride forward. Fanshawe, as an institution, proudly supports this project as it continues to take bold steps into a wide and important field that will benefit the society.

1.2 PROJECT MILESTONE SUMMARY

High-Level Milestone Timeline				
Milestone	**Description**	**Start Date**	**Status**	**Completion Date**
1	Inception	11 September 2023	Completed	27 September 2023
2	Analysis of Deliverables	28 September 2023	On-Going	18 October 2023
3	Design of Deliverables		Not Started	TBA
4	Construction, Results, and Discussion of Deliverables		Not Started	TBA
Final Report	Final Report and Evaluations		Not Started	TBA

TABLE 1: PROJECT MILESTONE SUMMARY

Summary Project Status	
Project Start Date	September 11, 2023
Estimated Project End Date	December 8, 2023
Impacted Process	
Potential Financial Impact	

TABLE 2: PROJECT STATUS SUMMARY

Milestone Event Table			
Milestone	**Status**	**Due Date**	**Expected Completion Date**
Milestone 1	Completed	September 27, 2023	September 27, 2023
Milestone 2	In Progress	October 18, 2023	October 18, 2023
Milestone 3	To Be Done	November 8, 2023	
Milestone 4	To Be Done	November 29, 2023	
Final Milestone	To Be Done	December 8, 2023	

TABLE 3: PROJECT MILESTONE SCHEDULE

2. WEB SEARCH

Research about automatic robot cars led to the discovery of a diverse and dynamic field at the intersection of technology and transportation. Autonomous vehicles are frequently used to describe automatic robot vehicles. The team conducted research on the following topics:

- **Freenove Robot Car Tutorial and YouTube Videos** – The team watched video demonstrations and tutorials related to Freenove automatic robot cars on YouTube. These videos provide an opportunity to see how these robotic vehicles operate in real-life scenarios, such as robot cars navigating through various environments, following predefined paths, avoiding obstacles, and perhaps even performing tasks like picking up objects or drawing patterns. These videos give a visual understanding of what these robots are capable of, making it easier to grasp their functionalities and potential applications.

- **ESP32 and microcontroller** The team researched these two topics because these items are included in Freenove robot kits, which are both important components in their respective contexts. The ESP32 is primarily focused on providing wireless connectivity and is suitable for IoT applicaions, while the microcontroller in Freenove robot kits is tailored for robot control and automation tasks, making it an integral part of building and programming autonomous robot cars.

- **Arduino Software (IDE)** – The group's robot programmers installed this software on their machines. This is the main IDE used to write and upload code for the Arduino board needed to test Gizmo. This software tool is a user-friendly, versatile, and widely adopted software tool for programming Arduino boards and compatible microcontrollers.

- **Sensors** – The group discovered three crucial sensors that will be integrated into the Gizmo: ultrasonic wave sensors for obstacle detection and distance measurement around the car robot; photoresistors as light sensors to help with following and detecting light; and line tracking sensors (PCF8574), specially created for tracking line routes on the ground or following a predefined path.

- **Camera** - A crucial step in maximizing the robot's visual capabilities is the team's research into how to capture live footage utilizing the camera attached to the chassis kit.

3. USE CASE DIAGRAM

The use case diagram illustrates the core functionalities and interactions within Gizmo, featuring the ESP32-WROVER microcontroller board. This diagram provides a visual representation of key actors, including the autonomous control system (ESP32-WROVER) and the associated use cases that define how the car kit's motors are autonomously controlled, enabling it to navigate, avoid obstacles, and follow predefined paths. The diagram showcases the car kit's ability to perform tasks independently based on sensor data and programmed logic, highlighting its versatility and automation capabilities.

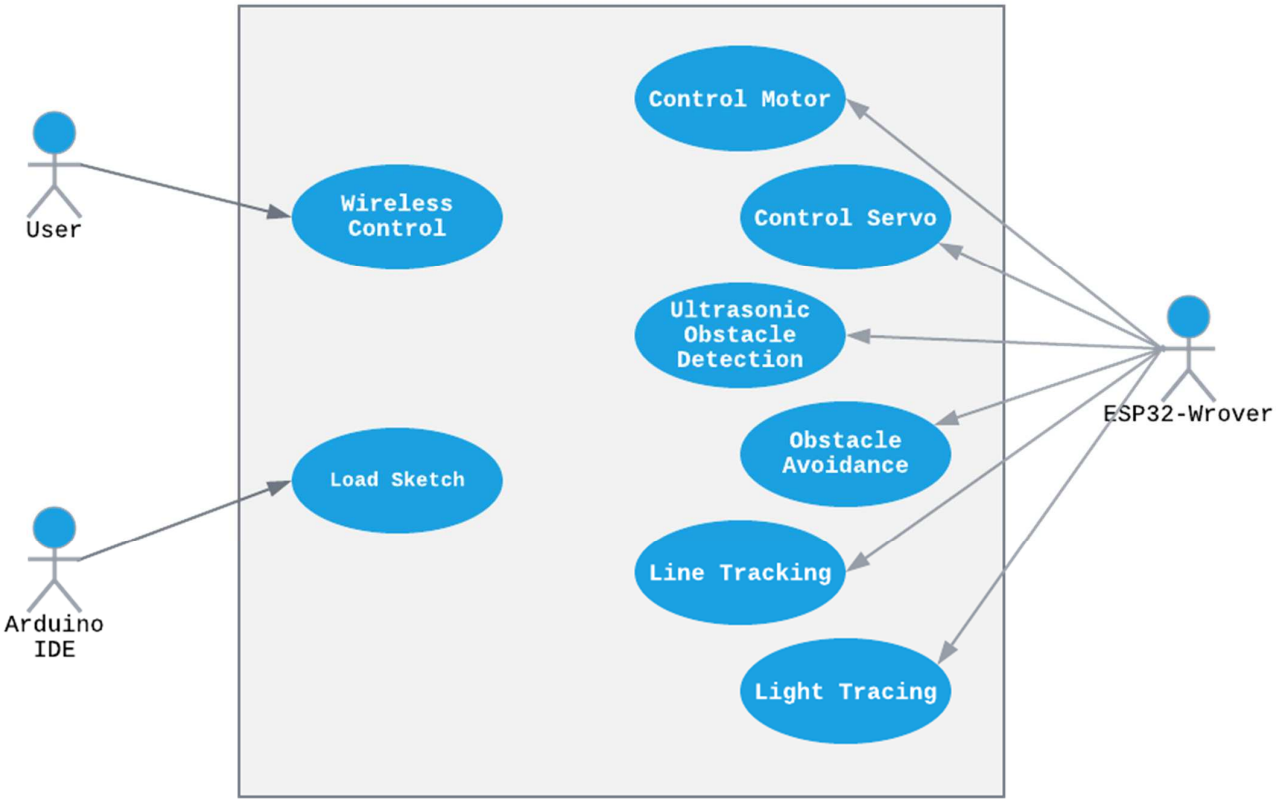


FIGURE 1: USE CASE DIAGRAM

Actors:

1. **System User**: This actor represents the person who interacts with Gizmo, either by use of a remote control or by creating or editing codes to control and observe its movements.

2. **ESP32-Wrover**: This actor represents the ESP32-WROVER microcontroller board, which serves as the central control unit of the car kit.

3. **Arduino IDE**: This actor represents the Arduino Integrated Development Environment, which is used by the user or developer to write, edit, upload, and debug code (sketch) programs for Gizmo.

Wireless Control Use Case	
Name:	Wireless Control
Actor:	User
Description:	Describes how the User interacts with the Gizmo to control its movements and functionalities wirelessly. This use case encompasses two sub-use cases: "Control via Wi-Fi" and "Control via IR."
Successful Completion:	**Control via Wi-Fi:** 1. The User launches the control application on their computer or mobile device. 2. The control device establishes a Wi-Fi connection with Gizmo. 3. The User interacts with the control interface on the device, sending commands to control the car kit's movements (e.g., forward, backward, left, right) or trigger specific functionalities. 4. The control device transmits control commands via Wi-Fi to Gizmo. 5. Gizmo interprets the commands and instructs the relevant components to execute the desired actions. 6. Gizmo responds to the control commands and performs the requested actions (e.g., moving in the specified direction). **Control via IR:** 1. The User utilizes the remote-control device (IR remote) pointed at the car kit's IR receiver. 2. The User presses buttons on the remote control, which generates IR signals. 3. The IR signals are transmitted to the car kit's IR receiver. 4. Gizmo decodes the received IR signals and maps them to specific control commands. 5. Based on the decoded IR signals, Gizmo instructs the relevant components to execute the desired actions. 6. Gizmo responds to the control commands sent via IR and performs the requested actions.
Alternative / Exceptions:	• In cases where there is interference or loss of signal during wireless control (both Wi-Fi and IR), the car kit may not respond as expected. The User may need to re-establish the connection or ensure an unobstructed line of sight for IR control. • If there are issues with the Wi-Fi network or the control application, the User may experience connectivity problems, leading to a loss of control. • In the case of IR control, if the remote-control device malfunctions or the IR signals are not received correctly, the control commands may not be executed as intended.
Precondition:	• Gizmo is powered on and operational. • The control devices (computer, mobile device, or remote control) are within the Wi-Fi range (for Wi-Fi control) or have a clear line of sight to the car kit's IR receiver (for IR control). • For Wi-Fi control, the control devices are connected to the same Wi-Fi network as the car kit. • The necessary software and applications for control are installed and running on the control devices.
Postcondition:	• Gizmo has executed the control commands as per the User's input, either via Wi-Fi or IR control.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors.

	• It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • Gizmo's motors are in good working condition and are capable of responding to control signals. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning. • Control devices (computer, mobile device, or remote control) are fully functional, and their Wi-Fi or IR capabilities are working correctly.

Load Sketch Use Case	
Name:	Load Sketch
Actor:	Arduino IDE
Description:	Illustrates the interaction between the Arduino IDE and Gizmo's microcontroller to load a custom user-written code (sketch) onto Gizmo. This process includes compiling the sketch and ensuring that all necessary libraries, including those required for the Gizmo's specific functionalities, are integrated into the sketch.
Successful Completion:	1. The User opens the custom sketch within the Arduino IDE, which may include code for controlling Gizmo's motors, sensors, and other functionalities. 2. The Arduino IDE initiates the Load Sketch use case when the User selects the "Upload" option for the Gizmo microcontroller. 3. The Arduino IDE communicates with the Gizmo microcontroller through the selected interface (e.g., USB) and uploads the compiled sketch to the microcontroller's memory. 4. Once the upload is complete, the Gizmo microcontroller stores the new sketch in its memory and is ready to execute the user-defined code.
Alternative / Exceptions:	• If the sketch contains errors or is incomplete, the Arduino IDE may generate errors, preventing the sketch from being uploaded to Gizmo. The User must address these errors before attempting to load the sketch again. • In case of communication errors or hardware issues between the Arduino IDE and Gizmo, the upload process may fail. The User should check the connection and troubleshoot any issues to ensure a successful sketch upload.
Precondition:	• The Arduino IDE is installed and operational on the User's computer. • The User has created or obtained a custom sketch for Gizmo, which includes the necessary code to control the car's functions and utilizes the required libraries. • The Gizmo's microcontroller is connected to the User's computer through an appropriate interface, such as USB.
Postcondition:	• The custom sketch, along with the required libraries, is successfully loaded onto Gizmo's microcontroller, allowing the microcontroller to perform the specified functions as defined in the sketch.
Assumptions:	• It is assumed that the custom sketch provided by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE.

Control Motor Use Case	
Name:	Control Motor
Actor:	ESP32-WROVER
Description:	Gizmo operates without direct user intervention and can perform various movements, including moving forward, moving backward, turning, and stopping based on preprogrammed logic and sensor input.
Successful Completion:	1. The ESP32-WROVER initiates the Control Motor use case by executing preprogrammed logic and control algorithms. 2. The Gizmo processes sensor data from various sensors, such as ultrasonic sensors or line-tracking sensors, to determine the car's environment and current conditions. 3. Based on the sensor data and predefined control algorithms, Gizmo calculates the desired motor actions. a. If an obstacle is detected in front of the car, the Gizmo adjusts the motor control signals to navigate around the obstacle, avoiding collisions. b. If a line-tracking sensor detects a deviation from a predefined path, the Gizmo adjusts the motor control signals to follow the path. c. Gizmo can adjust motor control signals to achieve movements such as moving forward, moving backward, turning, and stopping based on the desired behavior. 4. Gizmo communicates with the motor drivers and adjusts the voltage levels supplied to the motors according to the calculated motor control signals. 5. As a result of the adjusted voltage levels, the motors respond by performing the requested actions, such as moving the car in the specified direction or stopping its movement. 6. Gizmo continues to autonomously control the motors based on sensor data and predefined algorithms, adapting to changing environmental conditions as necessary.
Alternative / Exceptions:	• Depending on the sensor input and predefined logic, the Gizmo may perform different motor control actions, allowing the car kit to adapt to various scenarios. • If Gizmo encounters a critical error or malfunction during autonomous control that prevents it from executing the predefined control algorithms, it should implement a safety protocol, such as stopping the motors, to ensure the safety of the car kit.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control.
Postcondition:	• Gizmo autonomously responds to its environment by controlling the motors to achieve predefined behaviors. • The car kit remains in the specified motor state until Gizmo determines a behavior change is necessary based on sensor data and control algorithms.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • Gizmo's motors are in good working condition and are capable of responding to control signals. • All sensors, including ultrasonic sensors, line-tracking sensors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

Control Servo Use Case	
Name:	Control Servo
Actor:	ESP32-WROVER
Description:	Gizmo independently controls the servos to achieve specific movements or positions, contributing to the car kit's autonomous navigation and functionality.
Successful Completion:	1. The ESP32-WROVER initiates the Control Servo use case as part of its autonomous operation. 2. Gizmo processes sensor data and other inputs to determine the desired positions or movements for the servos. 3. Based on the sensor data and predefined control algorithms, Gizmo calculates the required servo positions or movements. 4. The Gizmo communicates with the servo driver circuits and adjusts the servos to achieve the calculated positions or movements. 5. The servos respond by moving to the specified positions or executing the designated movements autonomously.
Alternative / Exceptions:	• Depending on the sensor input and predefined logic, Gizmo may perform different servo control actions, allowing the car kit to adapt to various scenarios. • If Gizmo encounters errors or malfunctions during servo control that prevent it from executing the predefined control algorithms, it should implement safety protocols or error handling mechanisms to ensure safe operation and system stability.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control. • The servos are connected to Gizmo.
Postcondition:	• The servos have been autonomously controlled by Gizmo to achieve the desired positions or movements. • The car kit's autonomous operation continues with the servos maintaining their positions or making further adjustments as needed.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • Gizmo's servos are in good working condition and are capable of responding to control signals. • All sensors, including ultrasonic sensors, line-tracking sensors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

Ultrasonic Obstacle Detection Use Case	
Name:	Ultrasonic Obstacle Detection
Actor:	ESP32-WROVER
Description:	Gizmo utilizes ultrasonic sensors to detect obstacles in the car's path and make decisions to navigate around them, contributing to the car kit's autonomous obstacle avoidance capabilities.
Successful Completion:	1. The ESP32-WROVER initiates the Ultrasonic Obstacle Detection use case as part of its autonomous operation. 2. Gizmo activates the ultrasonic sensors to emit ultrasonic waves into the surrounding environment. 3. The ultrasonic sensors measure the time it takes for the emitted waves to bounce off nearby obstacles and return as echoes. 4. Based on the time taken for the echoes to return and knowing the speed of sound, Gizmo calculates the distances to the detected obstacles. 5. Gizmo processes the distance data to determine if any obstacles within a predefined range could obstruct the car's path. a. If obstacles are detected within the predefined range Gizmo generates control commands to navigate the car around the obstacles. b. The control commands may include adjustments to the motor speeds or directions to avoid collisions. c. Gizmo communicates with the motor control system to execute the generated control commands and navigate the car safely around detected obstacles. 6. The car kit continues its autonomous operation, periodically performing ultrasonic obstacle detection to ensure obstacle avoidance throughout the journey.
Alternative / Exceptions:	• Depending on the specific sensor data and predefined logic, Gizmo may adapt its navigation strategy to different scenarios, allowing the car kit to avoid obstacles effectively. • If Gizmo encounters errors or malfunctions during ultrasonic obstacle detection or if it fails to calculate distances accurately, it should implement safety protocols or error handling mechanisms to ensure safe operation and obstacle avoidance.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control. • Ultrasonic sensors are properly connected to the ESP32-WROVER
Postcondition:	• Gizmo has autonomously detected obstacles using ultrasonic sensors and successfully navigated the car around them to ensure safe and obstacle-free movement.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • Gizmo's servos are in good working condition and are capable of responding to control signals. • All sensors, including ultrasonic sensors, line-tracking sensors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

Obstacle Avoidance Use Case	
Name:	Obstacle Avoidance
Actor:	ESP32-WROVER
Description:	Gizmo employs sensor data to detect obstacles in the car's path and executes control algorithms to navigate around them, contributing to the car kit's autonomous obstacle avoidance capabilities.
Successful Completion:	1. The ESP32-WROVER initiates the Obstacle Avoidance use case as part of its autonomous operation. 2. Gizmo activates the sensors, including ultrasonic sensors for detecting obstacles and line-tracking sensors for tracking paths. 3. The ultrasonic sensors measure the distances to nearby obstacles by emitting ultrasonic waves and analyzing the returning echoes. 4. The line-tracking sensors monitor the ground for tracking lines or detecting deviations from the intended path. 5. Gizmo processes the sensor data to identify the presence and locations of obstacles and the position of the tracking lines. 6. Based on the sensor data and predefined control algorithms, Gizmo control commands to navigate the car around obstacles and maintain its path. 7. The control commands may include adjustments to motor speeds, directions, or servo orientations to avoid collisions or realign with the desired path. 8. Gizmo communicates with the motor control system and servos to execute the generated control commands and ensure obstacle avoidance and path following. 9. The car kit continues its autonomous operation, periodically performing obstacle avoidance and path tracking to maintain safe and accurate navigation.
Alternative / Exceptions:	• Depending on the specific sensor data and predefined logic, Gizmo adapts its navigation strategy to different scenarios, allowing the car kit to avoid obstacles effectively and respond to complex environments. • If Gizmo encounters errors or malfunctions during obstacle avoidance, sensor data inaccuracies, or control algorithm failures, it should implement safety protocols or error handling mechanisms to ensure safe operation and obstacle avoidance.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control. • Sensors, including ultrasonic sensors, line-tracking sensors, or any other relevant sensors, are properly connected to Gizmo.
Postcondition:	• Gizmo has autonomously detected obstacles and successfully navigated the car around them to ensure obstacle-free movement while maintaining its intended path.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • Gizmo's servos are in good working condition and are capable of responding to control signals. • All sensors, including ultrasonic sensors, line-tracking sensors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

Line Tracking Use Case	
Name:	Line Tracking
Actor:	ESP32-WROVER
Description:	Gizmo uses sensor data to track lines on the ground and maintain a predefined path, contributing to the car kit's autonomous navigation and following capabilities.
Successful Completion:	1. The ESP32-WROVER initiates the Line Tracking use case as part of its autonomous operation. 2. Gizmo activates the line-tracking sensors to scan the ground and detect lines or path markers. 3. The line-tracking sensors continuously monitor the ground surface and transmit data to Gizmo. 4. Gizmo processes the sensor data to identify the positions and orientations of the detected lines or markers. 5. Based on the sensor data and predefined control algorithms, Gizmo generates control commands to ensure that the car kit remains on the path defined by the lines. a. Control commands may include adjustments to motor speeds, directions, or servo orientations to keep the car kit aligned with the lines. 6. Gizmo communicates with the motor control system and servos to execute the generated control commands and maintain accurate line tracking. 7. The car kit continues its autonomous operation, periodically performing line tracking to stay on the predefined path and follow lines accurately.
Alternative / Exceptions:	• Depending on the specific sensor data and predefined logic, Gizmo may adapt its navigation strategy to different line-following scenarios, allowing the car kit to respond to complex line patterns or intersections. • If the Gizmo encounters errors or malfunctions during line tracking, sensor data inaccuracies, or control algorithm failures, it should implement safety protocols or error handling mechanisms to ensure safe operation and path following.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control. • Line-tracking sensors are properly connected to Gizmo.
Postcondition:	• Gizmo has autonomously tracked lines on the ground and successfully maintained its predefined path, ensuring accurate navigation and following capabilities.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • All sensors, including line-tracking sensors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

Light Tracing Use Case	
Name:	Light Tracking
Actor:	ESP32-WROVER
Description:	Gizmo uses sensor data to track lines on the ground and maintain a predefined path, contributing to the car kit's autonomous navigation and following capabilities.
Successful Completion:	1. The ESP32-WROVER initiates the Light Tracing (Autonomous) use case as part of its autonomous operation. 2. Gizmo activates the photoresistors to detect ambient light intensity in their respective directions. 3. The photoresistors continuously monitor the light intensity and transmit data to the Gizmo. 4. Gizmo processes the sensor data to determine the direction of the light source relative to the car's current orientation. 5. Based on the sensor data and predefined control algorithms, Gizmo generates control commands to adjust the car's orientation to face the light source. a. Control commands may include adjustments to servo orientations or motor directions to steer the car toward the light source. 6. Gizmo communicates with the motor control system and servos to execute the generated control commands, ensuring that the car kit follows the direction of the light source autonomously. 7. The car kit continues its autonomous operation, periodically performing light tracing to maintain alignment with the light source.
Alternative / Exceptions:	• Depending on the specific sensor data and predefined logic, Gizmo may adapt its light-following strategy to different scenarios, allowing the car kit to follow light sources effectively, regardless of their position or movement. • If Gizmo encounters errors or malfunctions during light tracing, sensor data inaccuracies, or control algorithm failures, it should implement safety protocols or error handling mechanisms to ensure safe operation and accurate light tracing.
Precondition:	• Gizmo is powered on and in a ready state. • Gizmo is operational and has access to sensor data for navigation and control. • Photoresistors are properly connected to Gizmo.
Postcondition:	• Gizmo has autonomously traced a light source and successfully adjusted the car's orientation to follow the direction of the light source, enabling light tracing capabilities.
Assumptions:	• It is assumed that the custom sketch loaded by the User is functional and free of critical errors. • It is assumed that the necessary libraries are available and correctly installed within the Arduino IDE. • All sensors, including photoresistors, and any other sensors used, are operational and provide accurate data. • Gizmo has a stable and reliable power supply, providing sufficient energy to the motors, servos, and sensors for proper functioning.

4. CLASS DIAGRAM

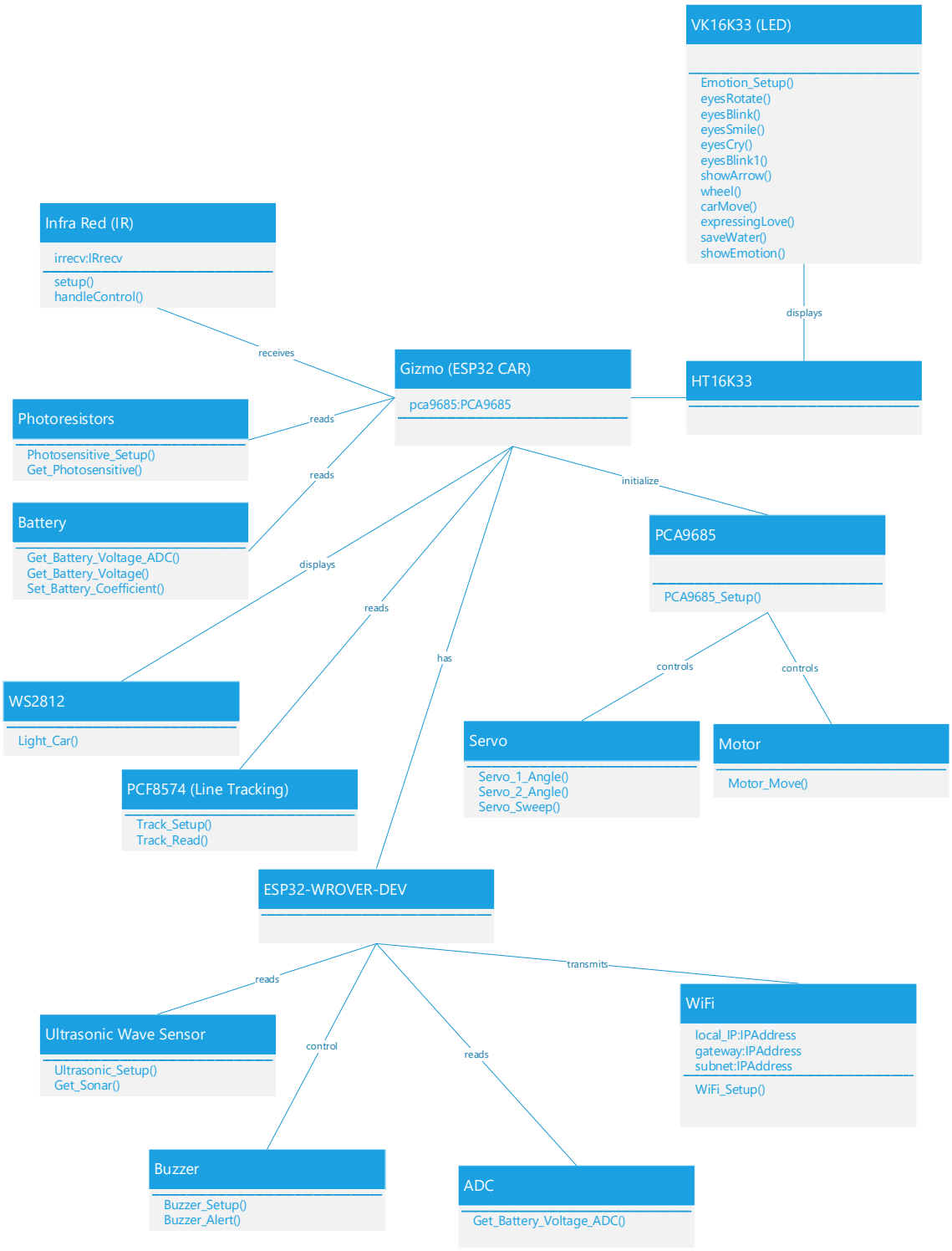


FIGURE 2: CLASS DIAGRAM

1.1 ENTITY AND ITS RELATIONSHIPS

- **Infra-Red (IR)** - The infrared module consists of two main components: the infrared remote control and the infrared receiver. The remote control is a device with buttons that emit infrared rays with different commands when pressed. The receiver, on the other hand, can detect and receive these infrared signals.

- **Photoresistors** - The photoresistor module is a light-sensitive component that changes its resistance based on the amount of light it receives. It consists of photoresistors, which are active components that decrease resistance when exposed to light. The module uses a circuit to detect changes in the resistance of the photoresistors, which in turn indicates the intensity of the light. By measuring the voltage changes in the circuit, the module can determine the brightness of the light source. This feature is used in the light-tracing car to control its movement toward the light source by reading the ADC (Analog-to-Digital Converter) values of the two photoresistors on the car's head.

- **Gizmo (ESP32 CAR)** - The ESP32 car is a smart car that can be controlled and programmed using the ESP32 development board. It has various components such as photoresistors, motors, ultrasonic sensors, servo motors, infrared sensors, and a buzzer. The car can be controlled through a client interface, which allows users to control the direction of the car, servo control, battery level, RGB LED color, and more.

- **VK16K33 (LED)** - This module represents the VK16K33 which is a chip that controls the LED matrix in Gizmo. It uses the HT16K33 chip to control the LED matrix. The LED matrix is divided into two sides, with the left side displaying a "+" symbol and the right side displaying an "o" symbol. The chip has an IIC address and IIC pins that need to be defined in the code.

- **HT16K33** - The HT16K33 module uses a chip to control the LED matrix.

- **Battery** - This module represents the battery, which is a device that stores and provides electrical energy. The battery is used in the Freenove 4WD Car Kit for ESP32.

- **PCA9685** - This module is PCA9685, which is a driver chip used to control motors and servos in Gizmo. It is controlled by the IIC chip and has multiple output channels. In the case of controlling motors, the PWM value of the corresponding output channels of PCA9685 is set to drive the motor. Similarly, in the case of controlling servos, the PWM value of the corresponding output channels of PCA9685 is set to drive the servo. The specific channels used for controlling motors and servos are mentioned in the document.

- **WS2812** - This module manages the WS2812, which is a type of LED that is used in the car. It is composed of 12 WS2812 LEDs, each of which can emit three basic colors: red, green, and blue. These LEDs can be controlled to emit colorful colors by inputting control signals through A0/WS2812. Each WS2812 LED supports 256-level brightness adjustment, allowing for a wide range of color options. The WS2812 LEDs are

connected in a cascading manner, with the DOUT of each WS2812 connected to the DIN of the next WS2812. This allows for easy control and synchronization of the LEDs.

- **PCF8574 (Line Tracking)** - This module manages the PCF8574, which is a chip that is used in the line tracking module of the Line Tracking Car. It is connected to the ESP32 and helps in obtaining the feedback value from the tracking module. The ESP32 reads the IO value of the PCF8574 to determine whether the three channels of the line tracking module are triggered. The obtained feedback value is then printed out through the serial port.

- **Servo** - This module manages the servo, which is a compact package that includes a DC motor, reduction gears, a sensor, and a control circuit board. It is commonly used to control motion in model cars, airplanes, robots, and other devices. Servos have a 180-degree range of motion and can output higher torque than a simple DC motor. The servo have three wire leads for electric power (positive and negative) and a signal line. The servo angle is controlled by a 50Hz PWM signal with a duty cycle in a certain range.

- **Motor** - This module controls and manages the motor, which is a device that converts electrical energy into mechanical energy. It consists of two parts: the stator and the rotor. The stator is the stationary part of the motor, usually the outer case, and it has terminals to connect to the power. The rotor is the rotating part, usually the shaft, and it can drive other mechanical devices to run. When a motor is connected to a power supply, it will rotate in one direction, and reversing the polarity of the power supply will make the motor rotate in the opposite direction.

- **ESP32-WROVER-DEV** - ESP32-WROVER-DEV is a module that is designed based on the PCB onboard antenna packaged ESP32-WROVER module. It is used in various projects and is compatible with the Arduino IDE. The module has hardware interfaces such as GPIO pins, LED indicators, a camera interface, a reset button, a boot mode selection button, and a USB port. It is a dual-core processor that can handle different tasks simultaneously.

- **Ultrasonic Wave Sensor** - The ultrasonic wave sensor is a module that uses ultrasonic waves to measure distances. It works by sending out ultrasonic waves and measuring the time it takes for the waves to bounce back after encountering an obstacle. By calculating the time difference, the distance between the sensor and the obstacle can be determined. The sensor consists of an ultrasonic transmitter and a receiver, which work together to send and receive the waves. The module is commonly used in applications such as obstacle avoidance in cars and distance measurement.

- **Wi-Fi** - The Wi-Fi module is a wireless communication technology that allows devices to connect to the internet or communicate with each other without the need for physical cables. It operates using radio waves and is commonly used in homes, offices, and public spaces to provide internet access. There are two modes of operation for Wi-Fi: station mode, where a device acts as a client and connects to a router network,

and AP mode, where a device creates a hotspot network for other devices to connect to.

- **Buzzer -** This module's purpose is to manage the buzzer, which is a sound component that is widely used in electronic devices such as calculators, electronic warning clocks, and alarms. There are two types of buzzers: active and passive.

  Active buzzer: It has an oscillator inside and will sound if it is supplied with power. It can only make a specific frequency of sound.

  Passive buzzer: It requires an external oscillator signal, usually using PWM with different frequencies, to make a sound. It can be controlled to make a sound with different frequencies. The resonant frequency of a passive buzzer is 2kHz, which means it is loudest when its resonant frequency is 2kHz.

- **ADC -** ADC stands for analog-to-digital converter. This module's purpose is an electronic-integrated circuit used to convert analog signals, such as voltages, into digital or binary forms consisting of 1s and 0s. The ADC on ESP32 has a range of 12 bits, which means it has a resolution of 2^12=4096. This allows it to divide the analog range (at 3.3V) into 4096 equal parts. The more bits an ADC has, the denser the partition of analog values will be, resulting in greater precision in the conversion.

5. SYSTEM ARCHITECTURE

1.2 HARDWARE COMPONENTS

- **Robot Car Body (ESP32 Car):** This is the physical platform that integrates all the hardware and tool components and allows the car to move and use all its functions.

- **ESP32-WROVER**: The ESP32-WROVER microcontroller module serves as the main brain (CPU) of the car, handling movements, function controls, and internal and external communication.

- **Motors**: These motors control the movement of the car wheels and their direction.

- **Servos**: The servo is used for controlling specific actions like steering or movement of sensors like the ultrasonic sensors.

- **LED Lights (WS2812):** These LEDs can be used for visual indicators and serve aesthetic effects.

- **LED Matrix**: The LED matrix can display text, light patterns, or sensor data.

- **Battery**: Stores electricity to power the entire car system.

- **Camera:** The camera captures images or video for various applications, such as computer vision.

- **IR Receiver:** The IR receiver component receives infrared signals from a remote control or other IR s, allowing the car to receive commands or data via external infrared communication.

- **Sensors**:
- **Ultrasonic Wave Sensors**: Used for obstacle detection and distance measurement around the car robot.
- **Photoresistors**: Light sensors that can be used for following and detecting light.
- **Line Tracking Sensors (PCF8574)**: Used for tracking line routes on the ground or following a specific path.

1.3 SOFTWARE COMPONENTS:

- **Arduino IDE**: This is the development environment for programming the ESP32 Car and controlling the hardware components' functions and behavior.

- **Sketches**: Custom software developed using Arduino IDE to control the motors, servo, and LEDs, and interact with sensors.

1.4 COMMUNICATION MODULES:

- **IR (Infrared)**: Used for communication between the car and a remote control or another device via infrared.

- **Wi-Fi**: Allows the car to connect to a local Wi-Fi network for remote control and data exchange.

- **USB:** Provides a versatile communication interface for programming, data transfer, debugging, and user interaction between the robot and external devices such as computers or microcontrollers.

1.5 USER INTERFACE:

- **Freenove Application**: A mobile or desktop application developed for controlling and monitoring the car. It can include features like driving controls, sensor data visualization, and more.
- **Executable File (Main.exe):** For desktop application (Windows).
- **Python Script (main.py):** For desktop application (cross-platform).
- **Freenove Mobile Application:** For mobile application.

1.6 SYSTEM ARCHITECTURE DESCRIPTION

1. The ESP32-WROVER serves as the central controller, running the custom firmware developed in Arduino IDE.
2. The sketches control the motors, servo, and LEDs, and communicate with the various sensors to gather data.
3. Sensor data (from ultrasonic sensors, photoresistors, and line tracking sensors) is processed within the ESP32 and can be used for decision-making and control.
4. The ESP32-WROVER communicates with the Freenove Application using Wi-Fi, allowing users to control the car remotely and receive real-time sensor data.
5. The Freenove Application can be installed on a mobile device or a computer (Windows, macOS, or Linux). It offers a user-friendly interface for driving the car, visualizing sensor data, and sending commands.
6. In addition to Wi-Fi communication, the car can also communicate using IR with a remote control or another IR-enabled device for basic control.
7. The LED lights (WS2812) and LED matrix can display emotions, sensor data, or visual effects as needed.

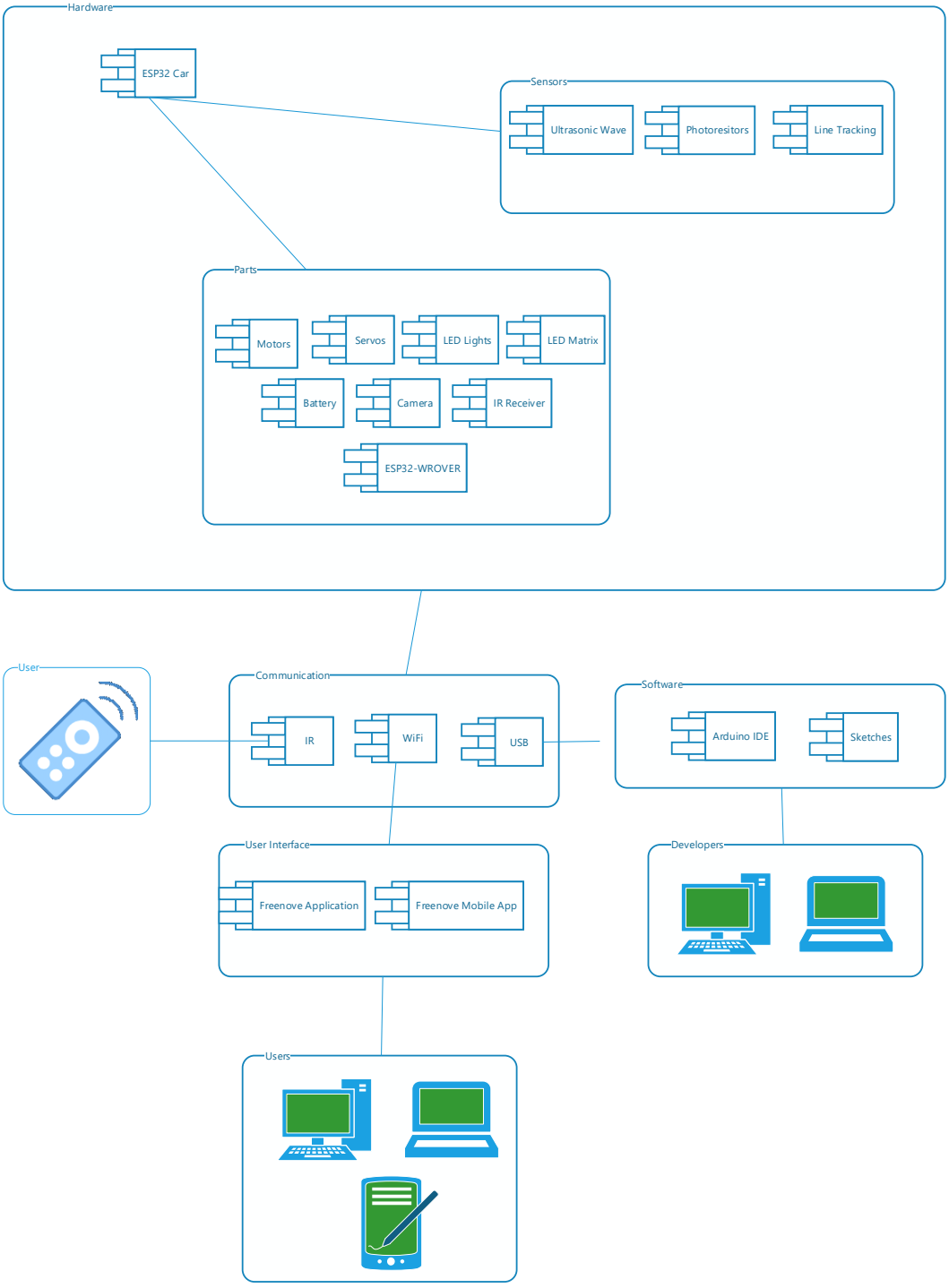


FIGURE 3: SYSTEM ARCHITECTURE

6. PROTOTYPE

User Story ID	As a <type of user>	I want <to perform some task>	so that I can <achieve some goal>
1	Gizmo Operator	Gizmo to autonomously detect and avoid obstacles in its path	Gizmo can continue its route following tasks efficiently and without human intervention.
2	Gizmo Operator	Gizmo to be capable of detecting and tracking a moving light source.	Gizmo can actively follow the light source by adjusting its movements, including moving forward, backward, and turning sideways.
3	Gizmo Operator	To see continuous improvements in the existing features of Gizmo.	I can make the most of the robot's capabilities and enjoy a better user experience.

See Appendix 1. * Prototypes for User Stories 2 and 3 will be available on Milestone 3.

7. TEST CASES

See Appendix 2. Test cases for User Stories 2 and 3 will be available on Milestone 3.

8. GANTT CHART

See Appendix 3.

9. REFERENCES

Freenove (n.d.). *Tutorial.pdf*. https://github.com/Freenove/Freenove_4WD_Car_Kit_for_ESP32/Blob/master/Tutorial.pdf

Freenove Videos (n.d.). *Freenove Videos*. https://www.youtube.com/@Freenove/videos

Arduino (2023). *Arduino IDE*. https://www.arduino.cc/

APPENDIX 1 – PROTOTYPE

Line Tracking with Obstacle Avoidance (obstacle circle)

if it is on track

Sensor scan 60 90 120 degrees

If there is an Obstacle head stop

Sensor scan

If it is too near move back

If it is too far move forward

Then Stop at a certain distance.

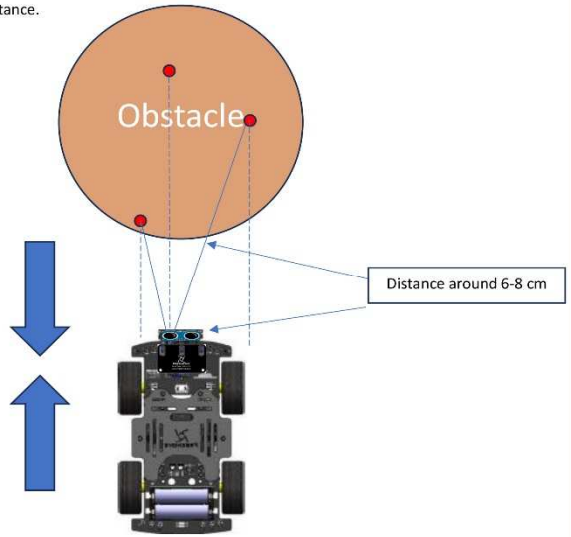


Illustration 1 (Obstacle Avoidance)

The sensor scans the object again.

Scans 60 90 120 degrees

Determine if the object is closer to the left or right.

If the object is on the left scanner turns 176 degrees, and then turns left at a certain distance.

If the object is on the right scanner turns 0 degrees, and then turns right at a certain distance.

then stops.

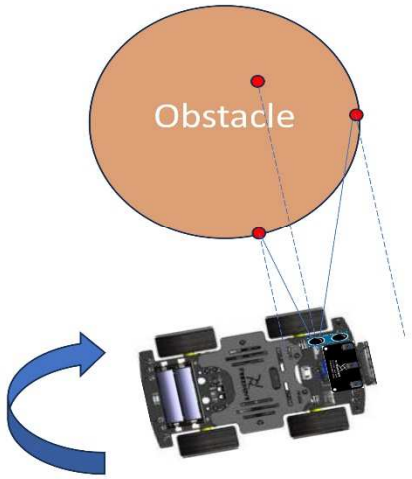


Illustration 2 (Obstacle Avoidance)

Before moving forward

Gizmo moves forward between 18-20 distances.

If distance is less than 18 Gizmo goes far

If distance is greater than 20 Gizmo goes near

Then loops to move forward between 18 to 20 distances.

Moves forward at a certain distance then stops.

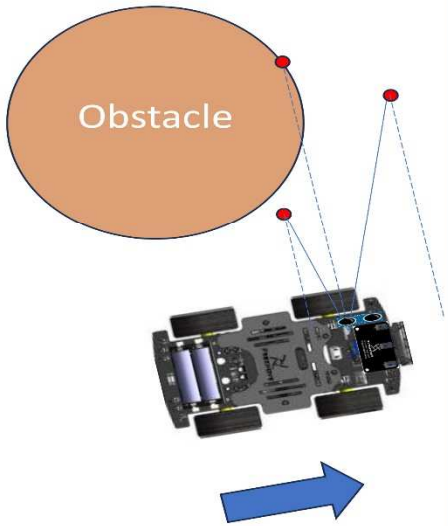
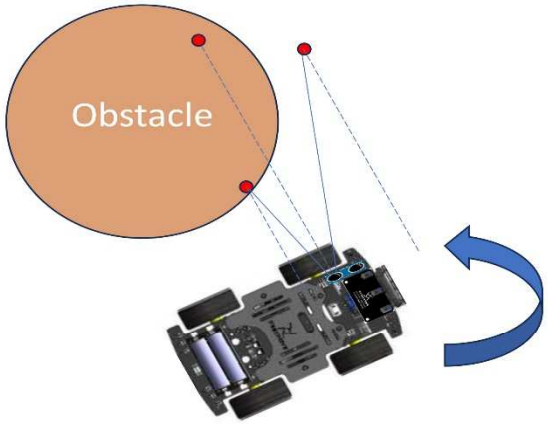


Illustration 3 (Obstacle Avoidance)

If the previous turn was left, Gizmo will turn to the right at a certain distance and then stop.

If the previous turn was right, Gizmo will turn to the left at a certain distance and then stop.



Repeat move forward at a certain distance then stop then turn at a certain distance then stop.

And stop when line track is located while moving and turning.

APPENDIX 2 – TEST CASES

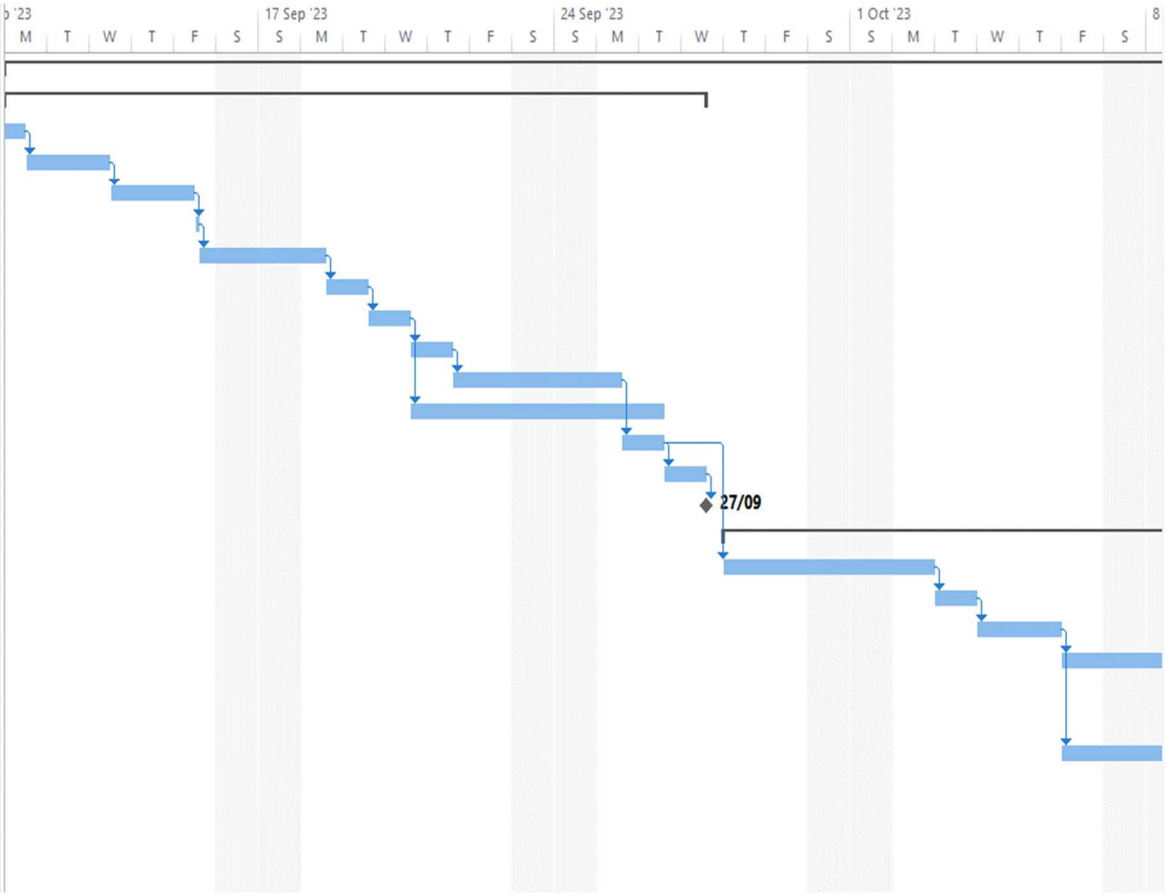
User Story 1: Gizmo to autonomously detect and avoid obstacles in its path

Test Case ID	Step No.	Operator Action / Input Specifications	Expected Results	Assumption / Operator Input	Status Pass / Fail	QC Comments / Actual Results
Pre-conditions: **1. Arduino IDE, USB-SERIAL CH340 (COMx), and necessary libraries installed in Gizmo Operator workstation.** **2. Gizmo battery is charged and installed in the battery compartment.** **3. Necessary sensors are integrated with Gizmo's car shield and ESP32-WROVER.**						
US1-001	**Loading of Obstacle Avoidance while in route sketch**					
	1	Connect your computer and Gizmo's ESP32 with a USB cable.	ESP2 has communication with the computer.	N/A		
	2	Open "07.1_Line_Tracking_with_Obstacle_Avoidance" folder in "Freenove_4WD_Car_Kit_for_ESP32\Sketches", double-click "07.1_Line_Tracking_with_Obstacle_Avoidance.ino".	Correct sketch selected.	Sketch is free of code errors		
	3	Select development board. Click Tolls on Menu bar, move your mouse to Board: "Arduino Uno", select ESP32 Arduino and then select ESP32 Wrover Module.	Correct development board selected.	N/A		
	4	Select serial port. Cilick Tools on Menu bar, move your mouse to Port and select COMx on your computer. The value of COMx varies in different computers, but it won't affect the download function of ESP32, as long as you select the correct one.	Correct serial port selected.	N/A		
	5	Click "Upload Using Programmer" and the program will be downloaded to Gizmo's ESP32.	Sketch successfully downloaded in Gizmo's ESP32.	N/A		
	6	A message "Done Uploading" and the console will have message "Leaving..., Hard resetting via RTS pin..."	Correct console output with no warnings or failures.	N/A		

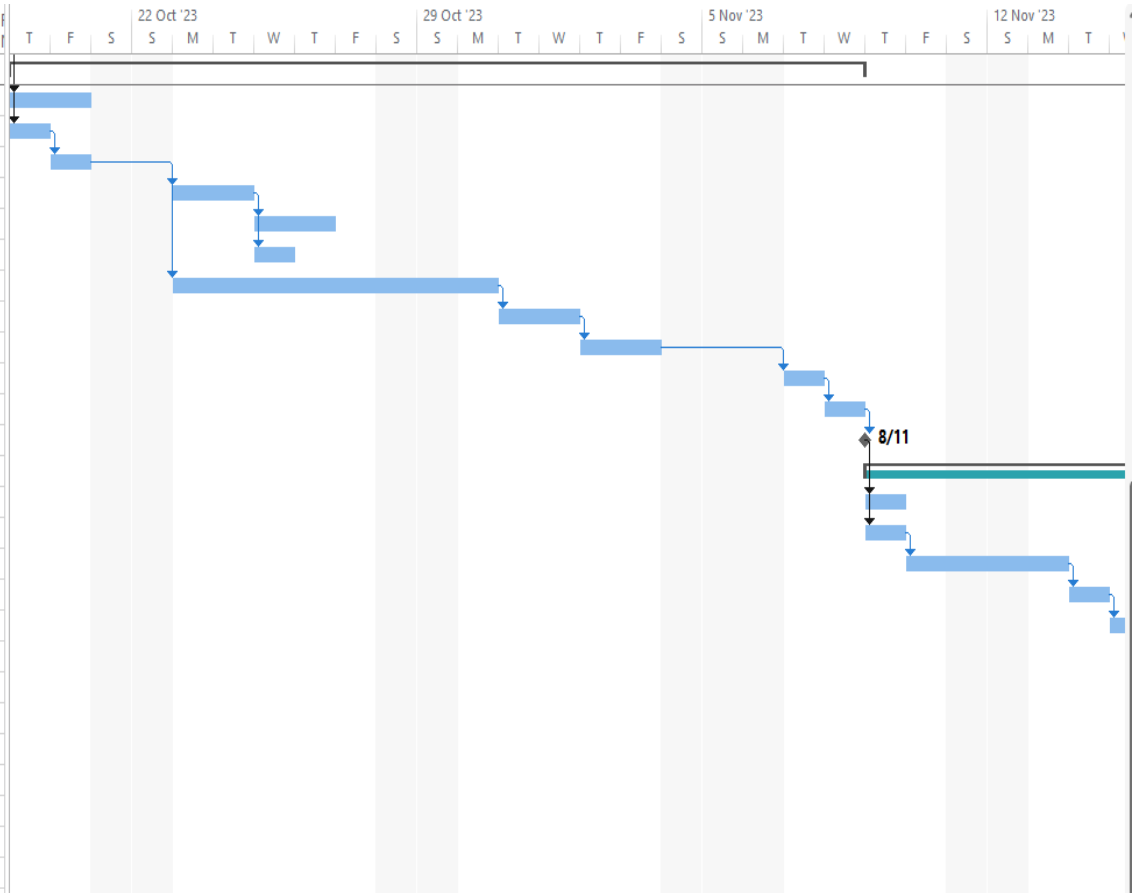
	7	Plug the Gizmo's ESP32 to the car shield.	ESP32 installed in car shield.	N/A		
	8	Unplug the USB cable from Gizmo.		N/A		
	9	Turn ON the power switch.	Gizmo successfully powered on.	N/A		
US1-002	**Obstacle Detection**					
	1	Steps 1 to 9 of US1-001 successfully completed.		N/A		
	2	While Gizmo is moving, it identifies an obstacle within the predefined distance using its front ultrasonic sensor.	Gizmo will stop and continue to scan the obstacle.	The obstacle has a circle shape or rounded. The object is 6cm to 8 cm away from the front of Gizmo's ultrasonic sensors.		
US1-003	**Obstacle Detection and Avoidance**					
	1	Steps 1 to 9 of US1-001 successfully completed.		N/A		
	2	While Gizmo is moving, it identifies an obstacle within the predefined distance using its front ultrasonic sensor.	Gizmo will stop and continue to scan the obstacle.	Obstacle has a circle shape or rounded. Object is 6 to 8 cm away from the front of Gizmo's ultrasonic sensors.		
	3	Gizmo will adjust its navigation to create distance from the obstacle.	Gizmo will move away if too near the obstacle or move closer if too far from the obstacle.			
	4	Gizmo will continue to scan the obstacle as it navigates within the perimeter of the object.	Gizmo will successfully navigate around the obstacle to avoid it.			

APPENDIX 3 – GANTT CHART

Task Mode	Task Name	Duration	Start	Finish	Predecess
	Capstone Project	**102 days**	**Mon 11/09/2**	**Wed 31/01/2**	
	Inception	**12,75 days**	**Mon 11/09/2**	**Wed 27/09/2**	
	Team formation	4 hrs	Mon 11/09/2	Mon 11/09/2	
	Project selection	2 days	Mon 11/09/2	Wed 13/09/2	3
	Sponsor approval	2 days	Wed 13/09/2	Fri 15/09/23	4
	Assigment of roles	2 hrs	Fri 15/09/23	Fri 15/09/23	5
	Logo creation	1 day	Fri 15/09/23	Mon 18/09/2	6
	Project charter	1 day	Mon 18/09/2	Tue 19/09/23	7
	Project scope	1 day	Tue 19/09/23	Wed 20/09/2	8
	Gantt chart	1 day	Wed 20/09/2	Thu 21/09/23	9
	Requirement gathering	2 days	Thu 21/09/23	Mon 25/09/2	10
	Hardware analysis	4 days	Wed 20/09/2	Tue 26/09/23	9
	Relevance	1 day	Mon 25/09/2	Tue 26/09/23	11
	Sponsor Sign-Off	1 day	Tue 26/09/23	Wed 27/09/2	13
	Milestone 1: Inception	0 days	Wed 27/09/2	Wed 27/09/2	14
	Sprint 1 - Goal 1	**15 days**	**Thu 28/09/23**	**Wed 18/10/2**	
	Web Search	3 days	Thu 28/09/23	Mon 2/10/23	13
	Use case Diagrams - Draft	1 day	Tue 3/10/23	Tue 3/10/23	17
	Class Diagrams - Draft	2 days	Wed 4/10/23	Thu 5/10/23	18
	Prototype - Draft	3 days	Fri 6/10/23	Tue 10/10/23	19
	System Architecture - Draft	3 days	Wed 11/10/2	Fri 13/10/23	20
	Test Case	1 day	Wed 11/10/2	Wed 11/10/2	20
	Development - Goal 1	5 days	Fri 6/10/23	Thu 12/10/23	19
	Unit Testing - Goal 1	2 days	Fri 13/10/23	Mon 16/10/2	23
	Sponsor Demo	1 day	Tue 17/10/23	Tue 17/10/23	24
	Deployment - Goal 1	1 day	Wed 18/10/2	Wed 18/10/2	25
	Milestone 2: Sprint 1 - Goal 1	0 days	Wed 18/10/2	Wed 18/10/2	26



Task Mode	Task Name	Duration	Start	Finish	Predecess
	Sprint 2 - Goal 2	**14 days**	**Thu 19/10/2:**	**Wed 8/11/2:**	
	System Architecture	2 days	Thu 19/10/2:	Fri 20/10/23	27
	Use Case Diagrams	1 day	Thu 19/10/2:	Thu 19/10/2:	27
	Class Diagrams	1 day	Fri 20/10/23	Fri 20/10/23	30
	Prototype	2 days	Mon 23/10/2	Tue 24/10/2:	31
	System Architecture	2 days	Wed 25/10/2	Thu 26/10/2:	32
	Test Case	1 day	Wed 25/10/2	Wed 25/10/2	32
	Development - Goal 2	6 days	Mon 23/10/2	Mon 30/10/2	31
	Unit Testing - Goal 2	2 days	Tue 31/10/2:	Wed 1/11/2:	35
	Integration Testing	2 days	Thu 2/11/23	Fri 3/11/23	36
	Sponsor Demo	1 day	Tue 7/11/23	Tue 7/11/23	37
	Deployment - Goal 2	1 day	Wed 8/11/2:	Wed 8/11/2:	38
	Milestone 3: Sprint 2 - Goal 2	0 days	Wed 8/11/2:	Wed 8/11/2:	39
	Sprint 3 - Improvements	**60 days**	**Thu 9/11/23**	**Wed 31/01/**	
	Use Case Diagram	1 day	Thu 9/11/23	Thu 9/11/23	40
	Class Diagram	1 day	Thu 9/11/23	Thu 9/11/23	40
	Prototype	2 days	Fri 10/11/23	Mon 13/11/2	43
	Test Case	1 day	Tue 14/11/2:	Tue 14/11/2:	44
	Development	5 days	Wed 15/11/2	Tue 21/11/2:	45
	Unit Testing	2 days	Wed 22/11/2	Thu 23/11/2:	46
	Integration Testing	1 day	Fri 24/11/23	Fri 24/11/23	47
	Sponsor Demo	1 day	Mon 27/11/2	Mon 27/11/2	48
	Deployment	1 day	Tue 28/11/2:	Tue 28/11/2:	49
	Recommendations	1 day	Tue 28/11/2:	Tue 28/11/2:	49
	Summary	1 day	Wed 29/11/2	Wed 29/11/2	51
	References	1 day	Wed 29/11/2	Wed 29/11/2	51
	Milestone 4: Sprint 3 - Improvements	0 days	Wed 29/11/2	Wed 29/11/2	53



APPENDIX 3 – REQUIREMENTS TRACEABILITY MATRIX

REQT ID	BUSINESS REQUIREMENT	CATEGORY	PRIORITY	FUNCTIONAL / NON-FUNCTIONAL SPECIFICATION REFERENCE	ACCEPTANCE CRITERIA
	USID-1 Autonomously detects and avoids obstacles in its path				
1		Functional	High	OD1	The ultrasonic head is successfully installed in Gizmo.
2		Functional	High	OD2	The ultrasonic head scans at different angles while detecting obstacles.
3		Functional	High	CA1	Gizmo can avoid the obstacle and continue navigating the pre-defined path.
4		Functional	High	CA2	Gizmo slows down, stops or steers away from obstacles.
5		Functional	High	UWE1	Gizmo successfully transmits sound waves from its transmitter
6		Functional	High	UWE2	Gizmo successfully receives sound waves in its receiver.
7		Functional	Medium	TIM1	Gizmo accurately get pingTime using pulseIn method.
8		Functional	Medium	TIM2	Gizmo accurately get pingTime using pulseIn method.
9		Functional	Medium	DC1	Gizmo accurately calculate the distance using the formula distance = velocity * time / 2.
10		Functional	Medium	DC2	Gizmo accurately calculate the distance using the formula distance = velocity * time / 2.

11		Non-Functional	Low	ER1	Gizmo to function effectively on a variety of surfaces.
12		Non-Functional	Low	ER2	Gizmo has stable and reliable performance regardless of the surface type.
13		Non-Functional	Low	PE1	Gizmo is capable of performing obstacle avoidance and line tracking without failure for more than 30 minutes of continuous operation.
14		Non-Functional	Low	PE2	Gizmo is capable of performing obstacle avoidance and line tracking without failure for more than 30 minutes of continuous operation.
15		Non-Functional	Low	RA1	Gizmo is capable of performing obstacle avoidance and line tracking without failure for more than 30 minutes of continuous operation.
16		Non-Functional	Low	UUS2	Gizmo will utilize LED matrix to display its actions.
17		Non-Functional	Low	UUS3	Gizmo can be powered off using the power button.
	USID-2 detecting and tracking a moving light source				
	USID-3 Improvements in the existing features of Gizmo				
	**** RTM for User Stories 2 and 3 will be available on Milestone 3***				

Category	Code	Requirement Group	Description
FUNCTIONAL	OD	**Obstacle Detection**	
	OD1		The robot shall employ an ultrasonic ranging module to detect obstacles within a specified range around the car during moving in route.
	OD2		The ultrasound system gathers data from multiple directions, evaluates this data independently in each direction, and subsequently manages the car's actions to steer clear of obstacles while moving in its route.
FUNCTIONAL	CA	**Collision Avoidance**	
	CA1		The obstacle avoidance system shall actively control the car's movement to prevent collisions with detected obstacles.
	CA2		When an obstacle is detected, the system shall initiate one or more of the following actions to avoid collision: o Slow down or stop the car. o Steer the car away from the obstacle
FUNCTIONAL	UWE	**Ultrasonic Wave Emission**	
	UWE1		The ultrasonic ranging module shall be equipped to emit ultrasonic waves in a controlled manner.
	UWE2		The emitted ultrasonic waves shall propagate through the environment and interact with obstacles, causing them to reflect the waves back towards the module.
FUNCTIONAL	TIM	**Time Interval Measurement**	
	TIM1		The system shall precisely measure the time interval between the transmission of ultrasonic waves and the reception of their echoes.
	TIM2		The time difference, measured in microseconds or milliseconds, shall be a reliable indicator of the total travel time of the ultrasonic waves from transmission to reception
FUNCTIONAL	DC	**Distance Calculation**	
	DC1		The module shall utilize the measured time interval to calculate the distance to encountered obstacles based on the speed of sound in the environment.

Category	Code	Requirement Group	Description
	DC2		The distance calculation shall provide accurate and real-time information regarding the proximity of obstacles to the car.
FUNCTIONAL	LD	**Light Detection**	
	LD1		The car shall be equipped with two photoresistors, strategically placed at the front of the vehicle to detect variations in light intensity.
	LD2		The system shall utilize the Analog to Digital Converter (ADC) values obtained from the photoresistors to accurately measure the light intensity.
FUNCTIONAL	LTB	**Light Tracing Behavior**	
	LTB1		The car shall be programmed to respond to the detected light source by autonomously steering towards it.
	LTB2		The degree of steering shall be proportional to the difference in ADC values between the two photoresistors, ensuring precise alignment with the light source.
NON-FUNCTIONAL	ER	**Environmental Requirements**	
	ER1		The robot car shall be designed and calibrated to function effectively on a variety of surfaces, including but not limited to carpets, smooth floors, and rough outdoor terrains.
	ER2		The system shall adapt its driving parameters and behavior to ensure stable and reliable performance regardless of the surface type.
	ER3		The system shall demonstrate robust performance in varying lighting conditions, including low-light environments and areas with intense light sources.
	ER4		The smart car's light tracking behavior shall remain accurate and responsive, adjusting its steering in accordance with changes in light intensity, without significant deviations or errors caused by fluctuations in lighting conditions.
NON-FUNCTIONAL	LLC	**Legal and Licensing Compliance**	
	LLC1		All files, materials, and instructional guides utilized in the development and documentation of this capstone project, including those related to the Freenove 4WD smart car, shall adhere to the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
	LLC2		A copy of this license shall be prominently displayed in the project's documentation and provided with any derivative works.
	LLC3		The project team shall ensure that any resources, software, or materials utilized are not employed for commercial purposes and are used in strict compliance with the licensing terms and conditions specified.

Category	Code	Requirement Group	Description
NON-FUNCTIONAL	PE	**Performance Efficiency**	
	PE1		The system shall aim to maximize its operational time on a single battery charge under typical usage conditions.
	PE2		The system shall strive to provide responsive obstacle detection and avoidance capabilities to ensure the smart car can efficiently respond in dynamic environments.
NON-FUNCTIONAL	RA	**Reliability and Availability**	
	RA1		The smart car shall be designed to operate continuously without any system failures for the duration of its battery capacity, ensuring reliable performance throughout its operational cycle.
	RA2		The system shall include built-in fault detection mechanisms to promptly identify and recover from common errors or sensor malfunctions that may occur during the battery's operational capacity.
	RA3		In case of a critical system failure within the battery's operational capacity, there should be a straightforward and rapid system restart procedure that allows the smart car to resume normal operation.
NON-FUNCTIONAL	S	**Scalability**	
	S1		The system architecture shall be designed to allow for easy integration of additional sensors or modules to enhance the smart car's capabilities.
	S2		The software shall be modular and scalable to accommodate future upgrades and improvements without requiring significant code rewrites.
NON-FUNCTIONAL	UUS	**Usability and User Experience**	
	UUS1		The user interface for controlling the smart car shall be intuitive and user-friendly, ensuring that operators with varying levels of technical expertise can easily interact with the system
	UUS2		The system shall provide clear and informative feedback to the user, including obstacle detection alerts and status updates, through both visual and auditory cues.
	UUS3		The smart car shall be designed with a physical emergency stop button that is easily accessible to the operator, allowing for immediate manual intervention in case of unexpected behavior or emergencies.