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

Final Report

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 CHARTER

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.SCOPE OF WORK

Going further from the previous project, there are two main functions that are part of or scope for the current project:

- Obstacle avoidance while driving in route: : Addition of the functionality of avoiding obstacles by temporarily going around the obstacle and going back to continue moving through the line route.
- Follow a moving light: The second big improvement of the autonomous robot is including the functionality to detect and follow a moving light, including the capability to move forward, backward, and turn sideways to follow the light target without the need of a tracking line route.

1.3. DELIVERABLES

The Project is also expected to provide the following Deliverables:

1. **Improved Autonomous Functionality**: Enhance the robot's autonomous capabilities to include obstacle avoidance, allowing it to navigate around obstacles and return to the tracking line.
2. **Target Tracing Feature:** Develop a feature that enables the robot to detect and trace light sources, allowing it to identify and follow light patterns while maintaining a defined distance from the source.
3. **User-Friendly Code**: Create clear and well-documented code that is easily understandable by both the project team and potential future project teams.
4. **Live Demonstration**: Conduct a live demonstration of the robot's improved functionalities to showcase its capabilities and functionality to stakeholders.
5. **Documentation**: Provide comprehensive project documentation, including design documents, technical specifications, and user manuals, to facilitate understanding and future maintenance.
6. **Testing and Validation**: Perform rigorous testing and validation of the robot's new features, as well as the existing features, to ensure they work as intended and meet project goals.
7. **Feature Expansion**: Addition of more features to the robot, such as additional sensors or functionality to enhance its versatility and usefulness.
8. **Presentation and Final Report**: Deliver a comprehensive project presentation and a final report summarizing the project's objectives, methodologies, achievements, and future recommendations.
9. **Project Handover**: Prepare the necessary documentation and training materials to hand over the project to a future group, ensuring proper continuity and knowledge transfer.

The project manager, **Wilberto Mejia**, is a key member of the **PC51 Technologies team** and is responsible for managing the development of each project from start to finish. The preparation of comprehensive project plans, attentive oversight of budgets, timetables, and scopes, and ongoing evaluation of project performance are all part of the project manager's duties. By taking a proactive approach, the project manager makes sure that any deviations from the project's course are quickly handled, upholding the highest standards of project excellence. The project manager's steadfast commitment and knowledge make him or her an essential factor in the team's project success.

PC51 Technologies is a team of six teammates.

Core Team Members	
Name	**Student ID / Section**
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 1: TEAM COMPOSITION

Following are the stakeholders involved in this project.

Name	Role
Professor Marc Bueno	Project Mentor
Professor Mona Abou Taka	Project Mentor
Professor Vinicius Rodrigues De Moraes	Project Sponsor

TABLE 2: KEY STAKEHOLDERS

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	Completed	18 October 2023
3	Design of Deliverables	19 October 2023	Completed	8 November 2023
4	Construction, Results and Discussion of Deliverables	9 November 2023	Completed	29 November 2023
Final Report	Final Report and Evaluations	30 November 2023	Completed	8 December 2023

TABLE 3: PROJECT MILESTONE SUMMARY

SPONSOR ACCEPTANCE

Approved by the Project Sponsor:

______________________________ Date: ____________________

2. PROBLEM STATEMENT

Currently, the robot car is lacking two crucial features for autonomous self-driving robots:

1. Avoidance of obstacle while in route

   The robot car can navigate autonomously through a line track, however, it lacks the capability to detect an obstacle while moving in that route. It cannot autonomously go around obstacles and go back to the route it is taking.

   Without this ability, the robot car is lacking a crucial feature that is important to autonomous self-driving robots. In the real world, we see obstacles like traffic lights, buildings, and any small or big items every day that can hinder its movement.

2. Following a moving target

   The robot car does not have the capability to follow a target which is in motion, which is also one of the important features a robot car should have in terms of autonomous driving.

   In developing this ability, it opens potential opportunities for future growth with regards to autonomously completing tasks. For now, we are developing this via incorporating light tracing functionalities, utilizing the robot car's photoresistors.

3. REQUIREMENTS GATHERING

3.1.1. FUNCTIONAL REQUIREMENTS

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.

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:
  - Slow down or stop the car.
  - Steer the car away from the obstacle.

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.

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.

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.
- DC2 - The distance calculation shall provide accurate and real-time information regarding the proximity of obstacles to the car.

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.

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.

3.1.2. NON-FUNCTIONAL REQUIREMENTS

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.

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.

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.

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.

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.

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.

4. ROLES

ROLES	Description	Resource
Project Manager (Leader)	- Main point of contact between stakeholders, sponsors, and the development team - Lead the team members, define project goals, communicate with stakeholders, and see a project through to its closure - Organizes, plans, and executes the project - Conducts meeting or project checkpoints	Wilberto Mejia
Associate Project Manager	- Assists the project manager in overseeing and managing the project	Anjeline Sayoc
Scrum Master	- Removing obstacles that interfere with the Scrum Team's progress. - Helping everyone to understand Scrum methodology and how to use it. - Taking care that the different scrum events are done properly.	Anjeline Sayoc
Product Owner	- Making sure the product goal is clearly communicated and understood. - Representing the stakeholders needs in the product backlog. - Getting the major advantage of the product through the scrum team.	Wilberto Mejia
Business / Requirements Analyst	- Understanding and analyzing the business strategies, goals, and requirements - Utilize and compile charts, tables, and other elements of data visualization - Work closely with others throughout the business hierarchy to communicate their findings and help implement changes - Gather, analyze, document, and review system requirements and ensure they are clear, feasible, and complete	Roben Woo Wilberto Mejia
Robot Programmer	- Utilizes and programs using TinkerCAD or Arduino - Codes the robot programs according to requirements - Create unit test cases - Perform unit and system testing according to test cases - Assembles and manages the physical hardware of robot	Anjeline Sayoc Dean Paul Martin Jhay-Ar Aguilar
Quality Assurance Analyst / Tester	- Analyzes and checks the quality of the robot - Creates test cases for unit and system integration testing - Performs the testing of the robot according to the final requirements	Roben Woo Wilberto Mejia Apple Dela Cruz
Documents Specialist	- Consolidate and review document completeness - Proofread documents - Ensure documents quality and availability	Apple Dela Cruz Roben Woo Wilberto Mejia

TABLE 4: TEAM COMPOSITION

5. PLAN WITH EVENT TABLE

The plan for the project is divided into multiple parts over a timeline of 3 months starting from 5th September 2023. The events are as follows:

Summary Project Status	
Project Start Date	September 11, 2023
Estimated Project End Date	December 8, 2023
Impacted Process	None
Potential Financial Impact	None

TABLE 5: PROJECT STATUS SUMMARY

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

TABLE 6: PROJECT MILESTONE SCHEDULE

6. RELEVANCE/SIGNIFICANCE

The Freenove Autonomous Robot Car is a versatile DIY robot kit that can be used in a variety of educational, hobbyist, and practical applications. Here are some of the significance and relevance of autonomous robot cars in the context of robotics and technology:

Education and Learning: Autonomous robot cars, designed for educational purposes, can be highly relevant in teaching students and enthusiasts about robotics, programming, and autonomous systems. They provide hands-on experience and a tangible platform for learning STEM (Science, Technology, Engineering, and Mathematics) concepts.

Skills Development: Assembling, programming, and experimenting with an autonomous robot car can help individuals develop valuable skills in electronics, coding, problem-solving, and project management, which are highly relevant in today's technology-driven world.

Innovation and Creativity: Such robot cars can inspire innovation and creativity among hobbyists and learners. Users can customize and experiment with these platforms to create unique applications, contributing to the evolution of autonomous technology.

Research and Development: Making an autonomous autonomous robot cars serve as platforms for researchers and developers to explore various sensors, algorithms, and technologies related to autonomous navigation, computer vision, machine learning, and artificial intelligence. This is highly relevant for advancements in autonomous systems.

Practical Applications: Depending on its capabilities, an autonomous robot car can have practical applications in fields like agriculture (for crop monitoring), logistics (for automated delivery), or research (for data collection). This practicality makes it relevant to specific industries.

Consumer Products: If the Freenove Autonomous Robot Car is designed for the consumer market, it may provide entertainment or utility features for consumers, such as home automation, surveillance, or interactive experiences.

The relevance and significance of the Freenove Autonomous Robot Car would depend on its specific features, capabilities, target audience, and applications.

7. RESOURCES/GANTT CHART

7.1.1. RESOURCES

The following the needed resources for this project:

Hardware Resources:

- Freenove 4WD Car Kit for ESP32-WROVER (Compatible with Arduino IDE)
- 2 Pack Rechargeable Battery 3.7V 2600mAh Large Capacity Flat Top Li-ion Battery
- Battery Charger
- WIFI
- Laptop
- Mobile Device

Software Resources:

- Integrated Development Environment (IDE) for programming the microcontroller (Arduino IDE)
- Freenove app for Android
- Programming languages (C/C++)
- Simulation and modeling software (if used for testing)
- Documentation and project management tools (e.g., Microsoft Word, project management software)

Human Resources:

- Project Manager
- Associate Project Manager
- Business / Requirements Analyst
- Robot Programmer
- Quality Assurance Analyst / Tester
- Documents Specialist
- Faculty or mentor guidance and support
- Potential collaboration with experts or researchers in the field

Communication and Collaboration:

- Communication tools and platforms for team meetings and discussions
- Access to libraries, research materials, and online resources

Safety Equipment:

- Safety gear, if required for working with certain hardware components

Project Documentation:

- Templates and tools for creating project documentation, reports, and presentations

Presentation and Demonstration Equipment:

- Audiovisual equipment and materials for presenting the project to stakeholders

7.1.2. GANTT CHART

See Appendix 1

8. CLIENT SIGN-OFF

I, **Vinicius Rodrigues**, after a thorough review and comprehensive understanding of the Project Charter for the **" Gizmo: The Next Frontier in Autonomous Robotics - Building on Falcon's Legacy,"** hereby formally signify my endorsement and approval for the project to advance in accordance with the conditions defined in the charter.

I acknowledge the alignment of the objectives, scope, and timeline in the project charter with the capstone project goals and expectations.

By affixing my signature below, I reaffirm my unwavering commitment to actively support the project's success.

Client Name: Vinicius Rodrigues

Date: 27/09/2023

Signature: ____

9. 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.

10. 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.

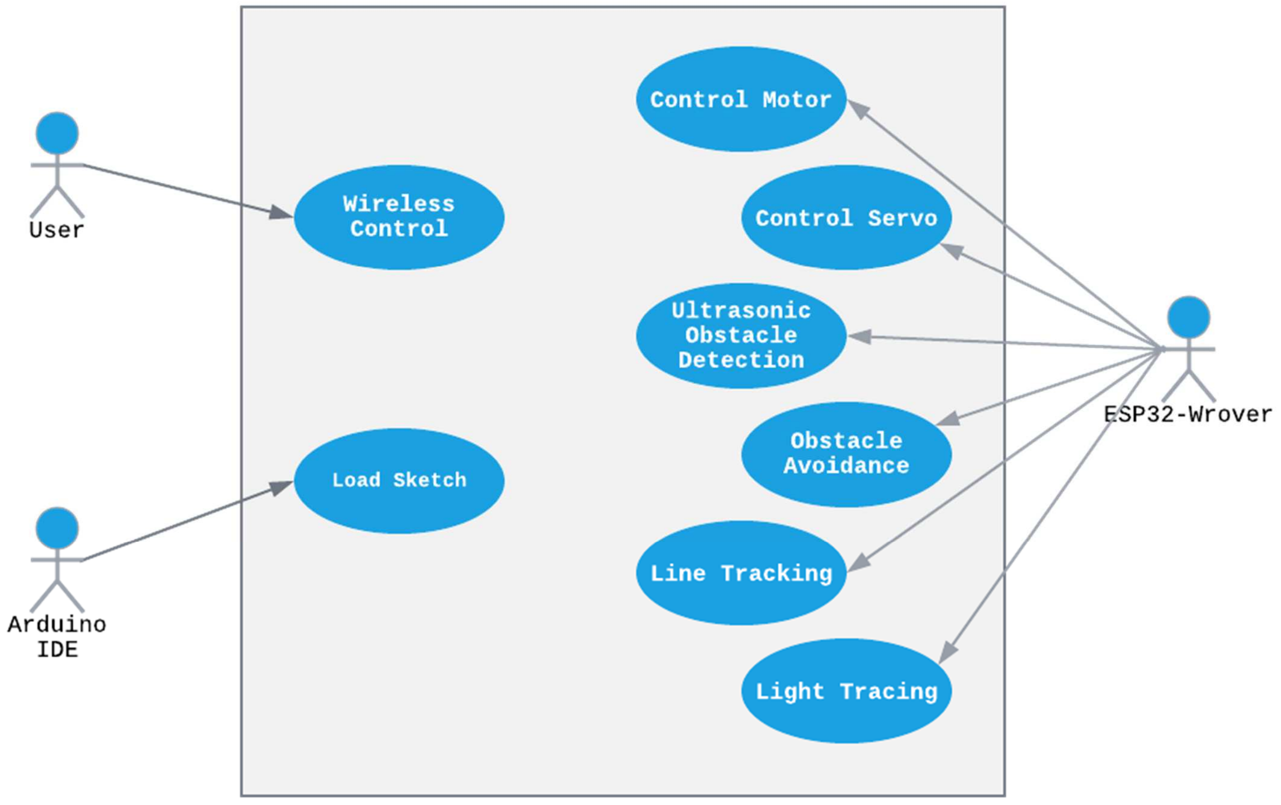


FIGURE 1: USE CASE DIAGRAM

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.

11. CLASS DIAGRAM

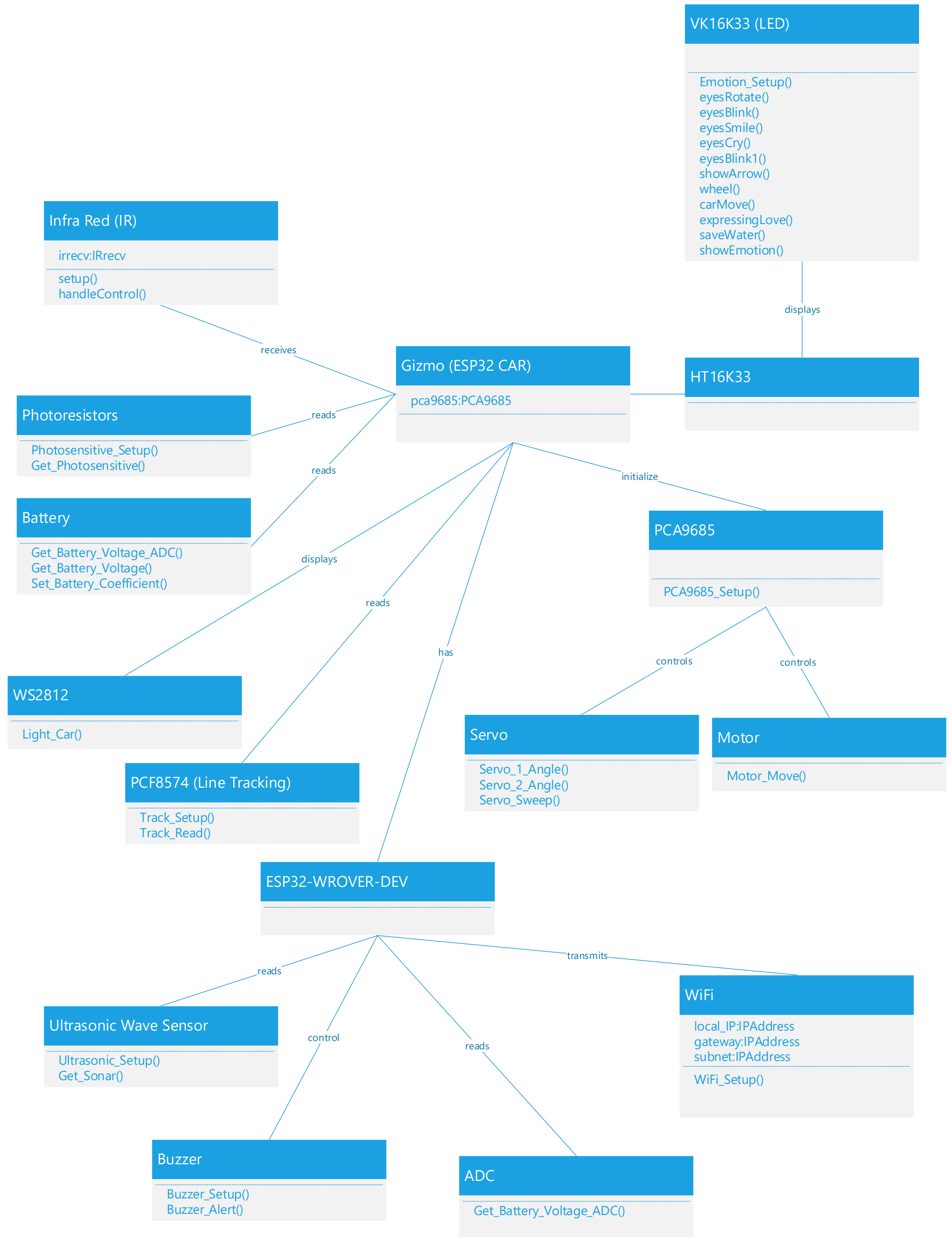


FIGURE 2: CLASS DIAGRAM

11.1. 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.

12. SYSTEM ARCHITECTURE

12.1. 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.

12.2. 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.

12.3. 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.

12.4. 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.

12.5. 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.

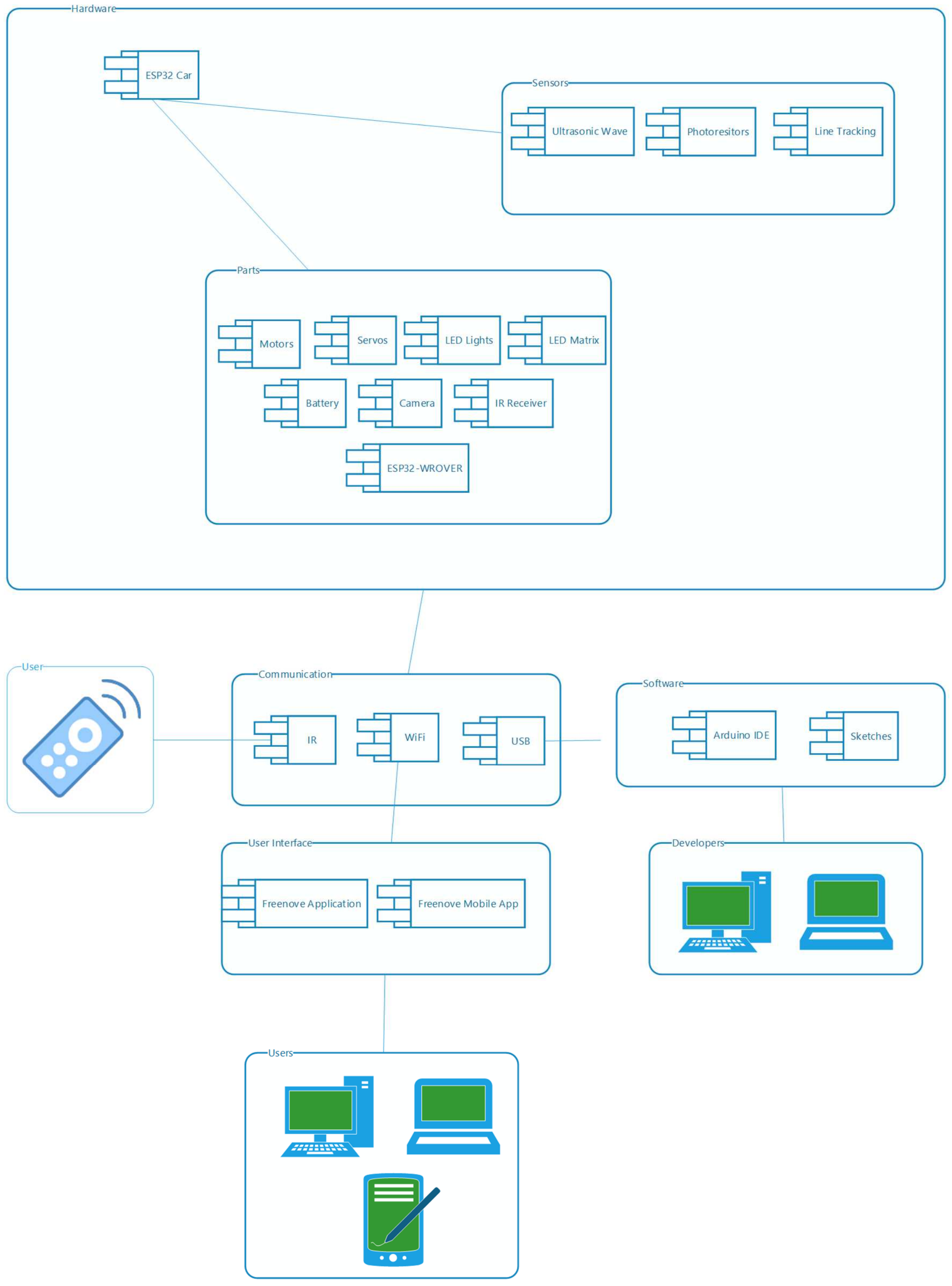


FIGURE 3: SYSTEM ARCHITECTURE

13. 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 2.

14. API FUNCTIONS USED

1. **IR (Infrared):**
    - Infrared communication involves transmitting and receiving data using infrared light. It's similar to how remote controls for TVs work.
    - While not explicitly referred to as an API, it involves communication protocols for encoding and decoding data using infrared signals.
    - This communication may require specific libraries or protocols to define how data is modulated onto the IR signal and how the receiver should decode and interpret this data.
2. **Wi-Fi:**
    - Wi-Fi allows the car to connect to a local wireless network, enabling remote control and data exchange.
    - Wi-Fi communication involves well-defined protocols and standards (e.g., TCP/IP) for network communication.
    - To interact with Wi-Fi networks, the software may use Wi-Fi libraries or APIs that provide functions for connecting to networks, transmitting data, and managing network configurations.
3. **USB:**
    - USB (Universal Serial Bus) is a common hardware interface for connecting devices to computers or other microcontrollers.
    - Although not explicitly mentioned as an API, using USB typically involves USB communication protocols and potentially software libraries.

- Libraries or APIs can manage USB connections, allowing data transfer, programming the microcontroller, and debugging.

4. **Freenove Application (Desktop and Mobile):**

   - The Freenove Application provides a user interface for controlling and monitoring the car.
   - APIs for handling user inputs might include libraries for GUI development, allowing the creation of buttons, sliders, and other interactive elements.
   - Sensor data visualization would involve using charting or visualization libraries to display data graphically.
   - Communication with the ESP32-WROVER over Wi-Fi may involve using networking libraries or protocols to send and receive commands and data between the application and the car.

5. **Executable File (Main.exe) and Python Script (main.py):**

   - Main.exe and main.py are components of the desktop application.
   - Communication with the ESP32-WROVER over Wi-Fi or other interfaces may require network communication libraries or APIs.
   - Python may use libraries like **socket** for network communication.

6. **Arduino IDE:**

   - While not explicitly an API, the Arduino IDE serves as the development environment for programming the ESP32 Car.
   - The IDE provides a set of libraries and APIs for working with hardware components, such as sensors, motors, servos, and LEDs.
   - Developers can use these libraries to control the hardware and define the behavior of the car.

15. DEPLOYMENT DIAGRAM

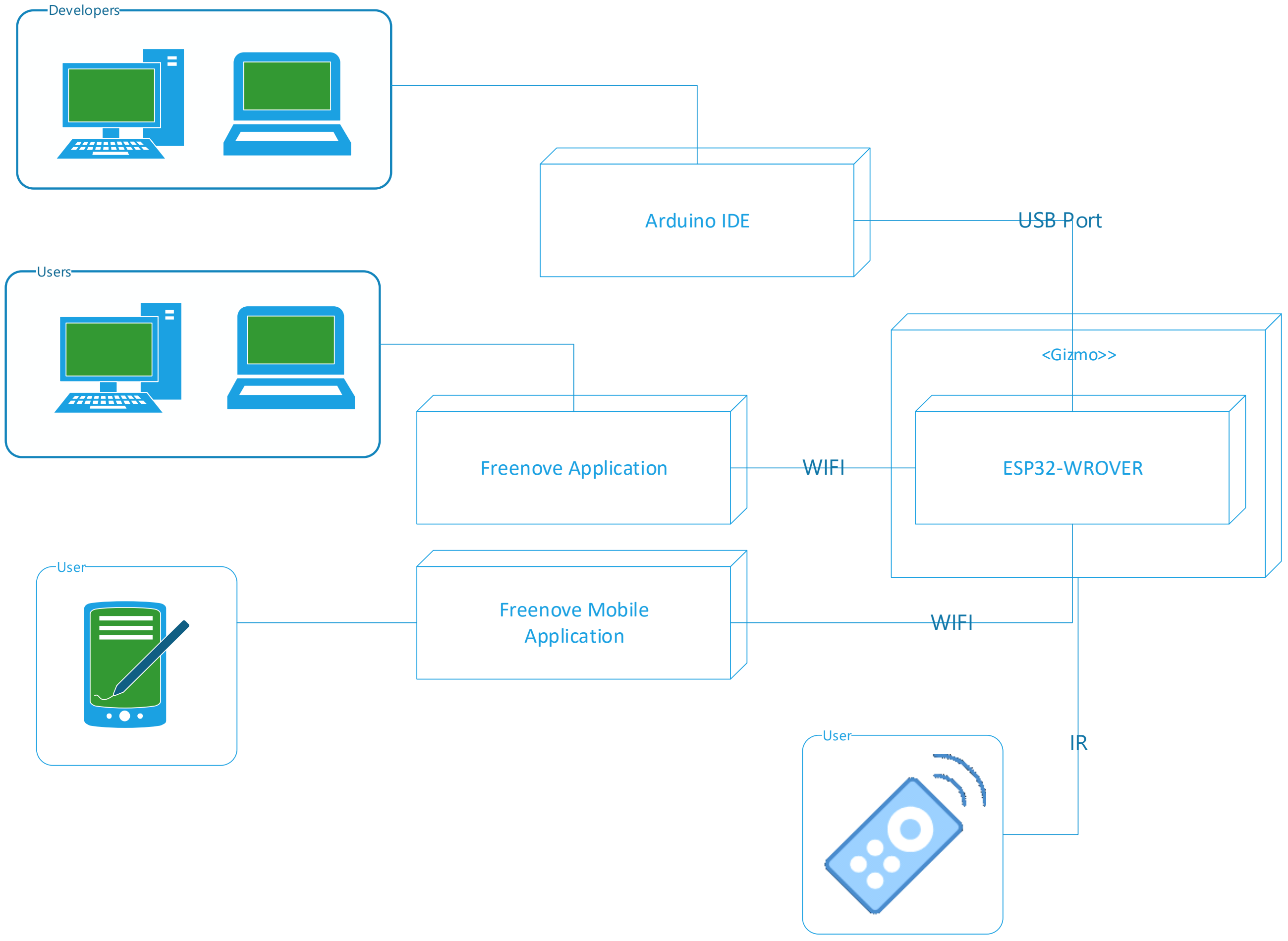


FIGURE 4: DEPLOYMENT DIAGRAM

- Gizmo ESP32 Wrover is the physical hardware where the sketch is deployed.
- The developer's computer or laptop can communicate with Gizmo through the USB port using the Arduino IDE for sketch deployment.
- The Freenove Application can be installed on a computer/laptop to control Gizmo via Wi-Fi.
- The Freenove Mobile Application can be installed on a mobile device to control Gizmo via Wi-Fi.
- An IR Remote Control device can also be used to operate Gizmo.
- Gizmo's hardware communicates with the computer/laptop via USB for sketch deployment.
- Gizmo's hardware communicates with the computer/laptop and mobile devices over Wi-Fi for remote control and operating Gizmo functions.

16. TEST CASES

See Appendix 3.

17. REQUIREMENTS TRACEABILITY MATRIX

See Appendix 4.

18. CODING STANDARDS

In software development, adherence to a well-defined coding standard is essential to fostering teamwork, enhancing code maintainability, and ensuring the long-term viability of a project. The following coding standard outlines a set of best practices and principles used in the enhancement of Gizmo. The characteristics of readability, robustness, and clarity in code are essential to the creation of high-quality software.

1. **Comments:**
    - Use comments to explain the purpose of code sections, functions, and variables for improved readability.
2. **Constants and Naming:**
    - Define constants for parameters like pin numbers with clear names.
    - Use CamelCase for function names and follow a consistent variable naming convention.
3. **Indentation and Whitespace:**
    - Consistently indent code for a clear structure.
    - Use spaces around operators for better readability.
4. **Modularity and Functions:**
    - Organize code into functions with specific responsibilities to promote modularity.
    - Create separate functions for initializing different modules.
5. **Error Handling:**
    - Implement error checking and handling where appropriate, ensuring robustness in code.
6. **Delay Usage:**
    - Use delays judiciously, especially in functions like **Servo_Sweep**, to avoid blocking code execution for extended periods.

7. **Git Version Control:**
    - Initialize a Git repository at the project's root.
    - Adopt a branching strategy (e.g., feature branches, release branches) and use clear branch names.
    - Write descriptive commit messages and make each commit a logically atomic change.
    - Use pull requests for collaborative development and conduct code reviews before merging.
    - Choose a merging strategy (merge commits, rebase, or squash merges).
    - Tag releases with version numbers.
    - Include a **.gitignore** file to exclude unnecessary files.
    - Document Git conventions and workflows in the project's README.
    - Handle merge conflicts effectively, encouraging communication within the team.
8. **README Documentation:**
    - Keep the README up-to-date with information on cloning, setup, and contribution guidelines.
9. **Remote Repositories:**
    - If collaborating, use remote repositories (e.g., GitHub, GitLab) for sharing code and collaboration.
10. **Coding Style Consistency:**
    - Maintain coding style consistency within the team or community, adhering to established conventions and practices.

19. PRODUCT / SPRINT BACKLOG

See Appendix 5.

20. DEPLOYMENT GUIDE

See Appendix 6.

21. USER GUIDE

See Appendix 7.

22. RECOMMENDATIONS

The team recommends exploring Gizmo's potential application in humanitarian efforts, specifically for landmine detection in conflict-affected regions like Colombia. By enhancing Gizmo's capabilities to systematically cover predefined areas, incorporating sensors such as a metal detector and transmitting specific landmine locations to a central database, the robot could significantly contribute to the safety of communities and aid organizations involved in mine-clearance efforts. With access to relevant statistical data from the Colombian government, Gizmo has the potential to be a valuable asset supporting ongoing initiatives to make affected regions safer.

Looking ahead, the team proposes key enhancements for Gizmo's future development. This includes refining obstacle avoidance and exploration by updating the code to detect polygons, expanding beyond the current calibration for round obstacles. Integrating a camera and ultrasonic wave sensors will further enhance obstacle avoidance, providing a more comprehensive understanding of the environment. Moreover, integrating obstacle avoidance and light tracing to track sketches in both the WIFI car and Infrared car sketches can make Gizmo more versatile and adaptable to different scenarios. These enhancements collectively aim to elevate Gizmo's performance in its existing applications and pave the way for more sophisticated tasks across diverse domains.

23. SUMMARY

This BIA Capstone Project, "Gizmo: The Next Frontier in Autonomous Robotics - Building on Falcon's Legacy," stands as a detailed and thorough guide designed for users keen on incorporating particular features into the Freenove ESP32-WROVER robot car. It takes a user-friendly approach, offering step-by-step implementation details, expected results, and related test cases for a hands-on and practical experience. The document revolves around three user stories: autonomous obstacle avoidance, light source tracking, and continuous feature improvements.

The first user story takes users through the process of creating a robot car adept at autonomously detecting and avoiding obstacles. The guide covers connecting distance sensors, uploading code for interpreting sensor data, implementing motor control, and testing the robot car in an obstacle-rich environment. The anticipated result is a robot car that adeptly navigates its surroundings, detects obstacles, and dynamically alters its path to prevent collisions. The second user story delves into crafting a robot car with the capability to detect and track a moving light source. This involves connecting a light sensor, uploading code for interpreting changes in light intensity, implementing motor control or servo mechanisms, and testing the robot car in an environment with a moving light source. The expected outcome is a robot car proficient in tracking a dynamic light source and adjusting its position or orientation accordingly. Lastly, the third user story encourages a culture of continuous improvement in the existing features of the ESP32-WROVER robot car. Users are guided to select a specific feature for enhancement, analyze the current implementation, introduce changes, and rigorously test the updated feature for noticeable improvements.

The appendices offer additional resources such as a Gantt chart, deployment guide, troubleshooting tips, and online support resources.

The project recommendation suggests future applications for Gizmo, including potential use in humanitarian efforts such as landmine detection, further highlighting its versatility and societal impact. In essence, this Capstone Project stands as a comprehensive and pragmatic resource, providing users of varied experience levels with the tools to explore and maximize the potential of the Freenove ESP32-WROVER robot car.

24. GITHUB URL

https://github.com/rwoofanshawe/gizmo-mgmt6134

25. REFERENCES

GitHub. (n.d.). ***Gizmo: The Next Frontier in Autonomous Robotics - Building on Falcon's Legacy.*** https://github.com/rwoofanshawe/gizmo-mgmt6134/tree/master/Freenove

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

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

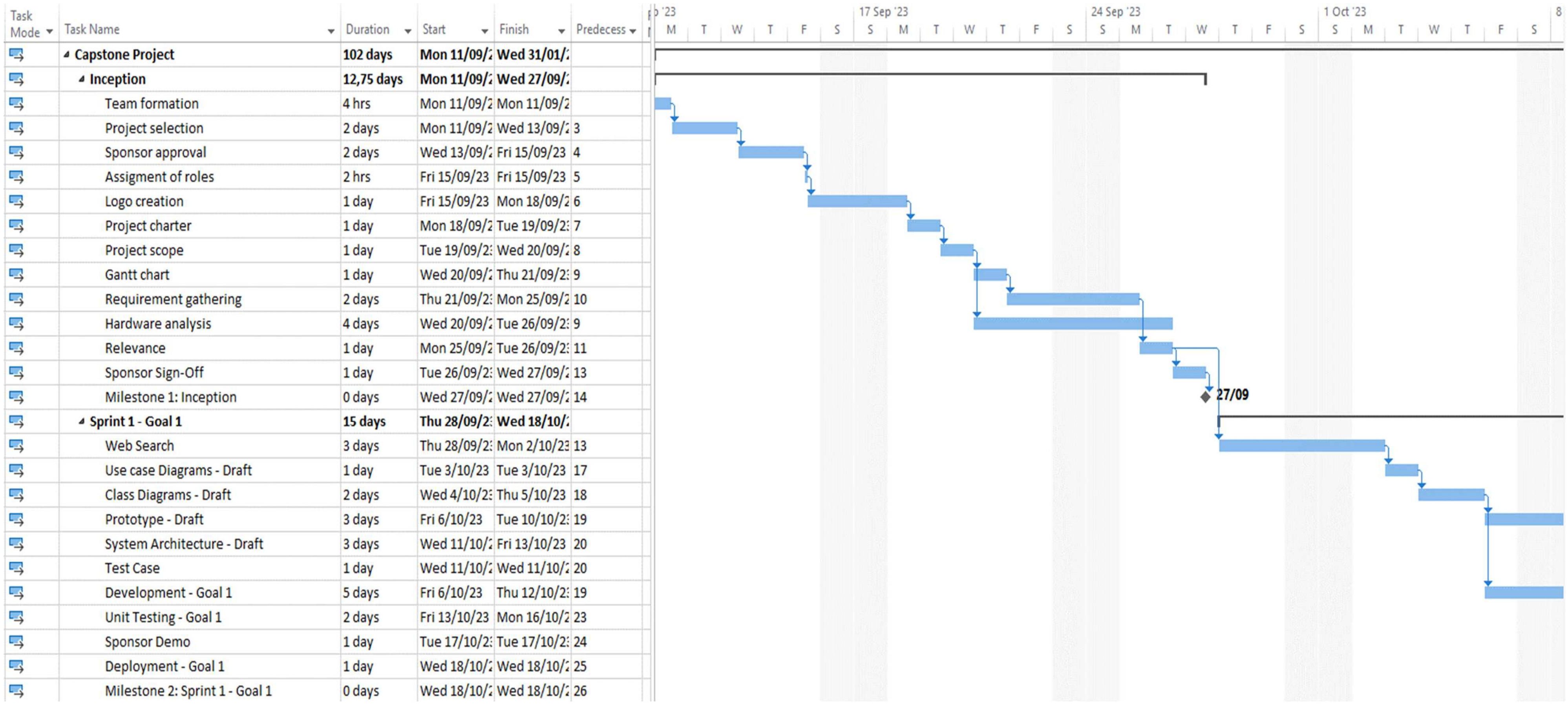
Freenove Videos (n.d.). *Freenove Videos*. https://www.youtube.com/@Freenove/videos

Team, T. A. (n.d.). Arduino style guide for writing content. Arduino Documentation.
https://docs.arduino.cc/learn/contributions/arduino-writing-style-guide

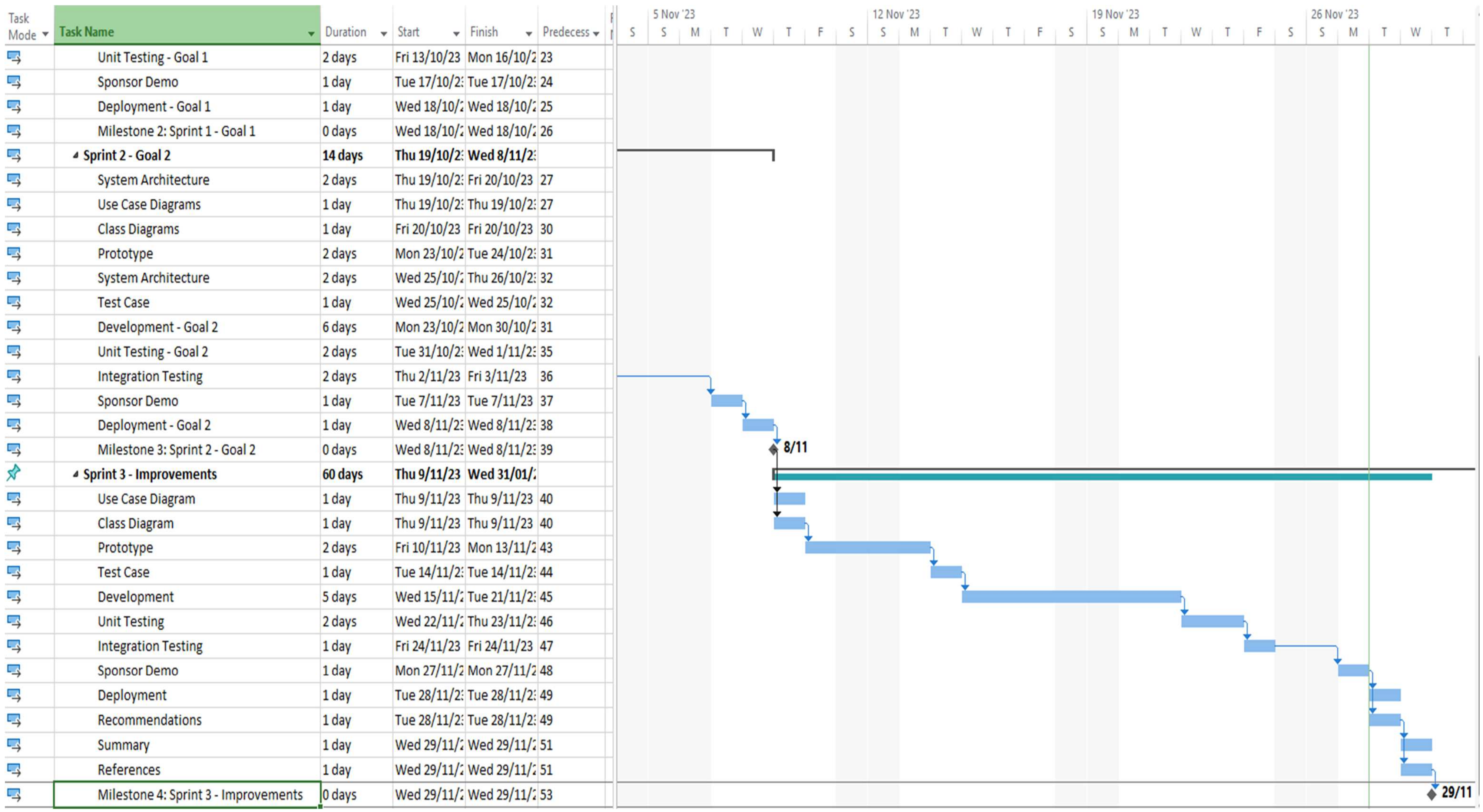
Team, T. A. (n.d.-a). Arduino style guide for creating libraries. Arduino Documentation.
https://docs.arduino.cc/learn/contributions/arduino-library-style-guide?queryID=undefined&amp;_gl=1%2A92kqy2%2A_ga%2AMTc5MjMwNzk4MC4xNjk0MTkyMDY4%2A_ga_NEXN8H46L5%2AMTY5OTk4NjQzMC4xMi4xLjE2OTk5ODY1NTQuMC4wLjA.

APPENDICES

APPENDIX 1 – 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 19/09/2	7
	Project scope	1 day	Tue 19/09/2	Wed 20/09/2	8
	Gantt chart	1 day	Wed 20/09/2	Thu 21/09/2	9
	Requirement gathering	2 days	Thu 21/09/2	Mon 25/09/2	10
	Hardware analysis	4 days	Wed 20/09/2	Tue 26/09/2	9
	Relevance	1 day	Mon 25/09/2	Tue 26/09/2	11
	Sponsor Sign-Off	1 day	Tue 26/09/2	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 28/09/2**	**Wed 18/10/2**	
	Web Search	3 days	Thu 28/09/2	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 4/10/2	Thu 5/10/23	18
	Prototype - Draft	3 days	Fri 6/10/23	Tue 10/10/2	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 12/10/2	19
	Unit Testing - Goal 1	2 days	Fri 13/10/23	Mon 16/10/2	23
	Sponsor Demo	1 day	Tue 17/10/2	Tue 17/10/2	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
	Unit Testing - Goal 1	2 days	Fri 13/10/23	Mon 16/10/2	23
	Sponsor Demo	1 day	Tue 17/10/2:	Tue 17/10/2:	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
	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 2 – 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.

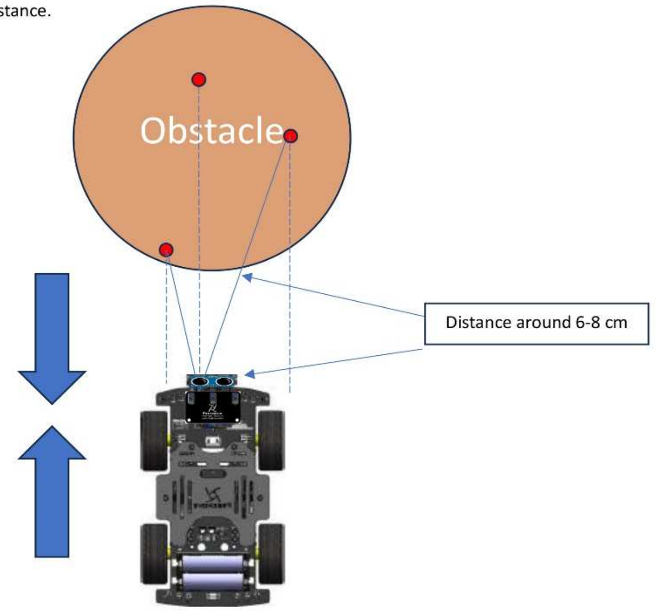


Illustration 1 (Obstacle Avoidance)

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.

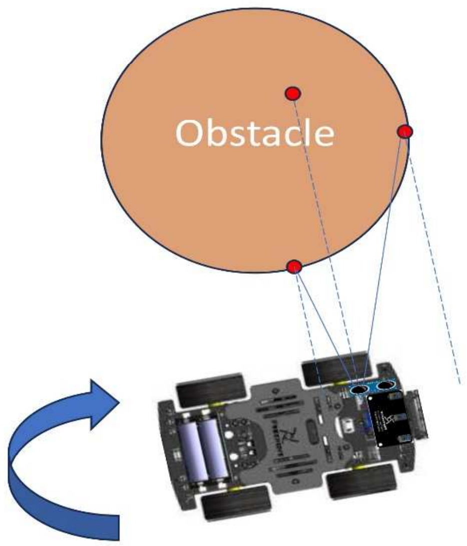


Illustration 2 (Obstacle Avoidance)

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.

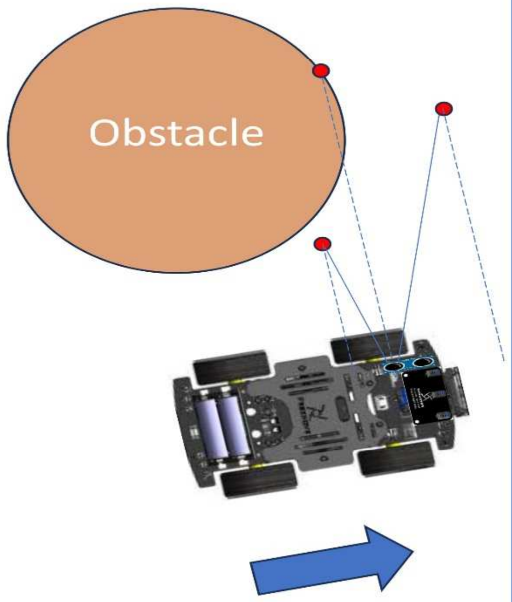


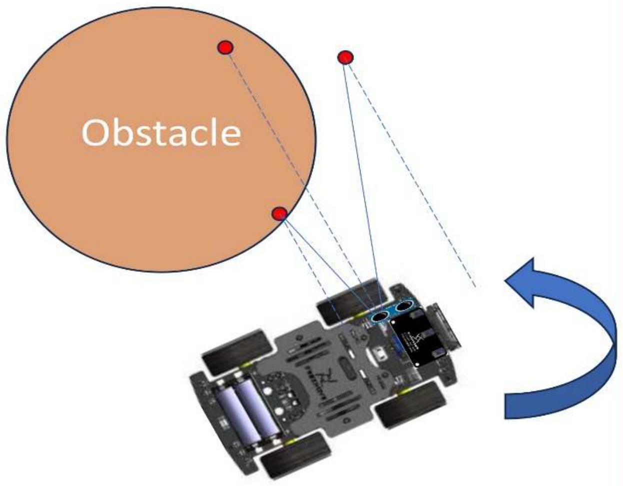
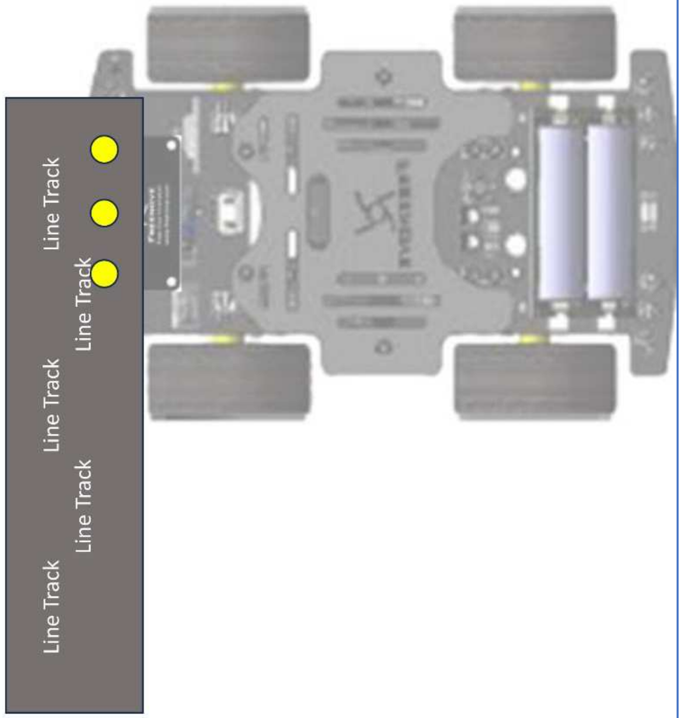
Illustration 3 (Obstacle Avoidance)	***Illustration 4 (Obstacle Avoidance)***
If the previous turn was left, Gizmo will turn to the right at a certain distance and then stop. If the previous turn was right, Gizmo will turn to the left at a certain distance and then stop.  Repeat move forward at a certain distance then stop then turn at a certain distance then stop. And stop when line track is located while moving and turning.	from obstacle avoidance, Gizmo detects 111 then stops.  Line Track Line Track Line Track Line Track Line Track

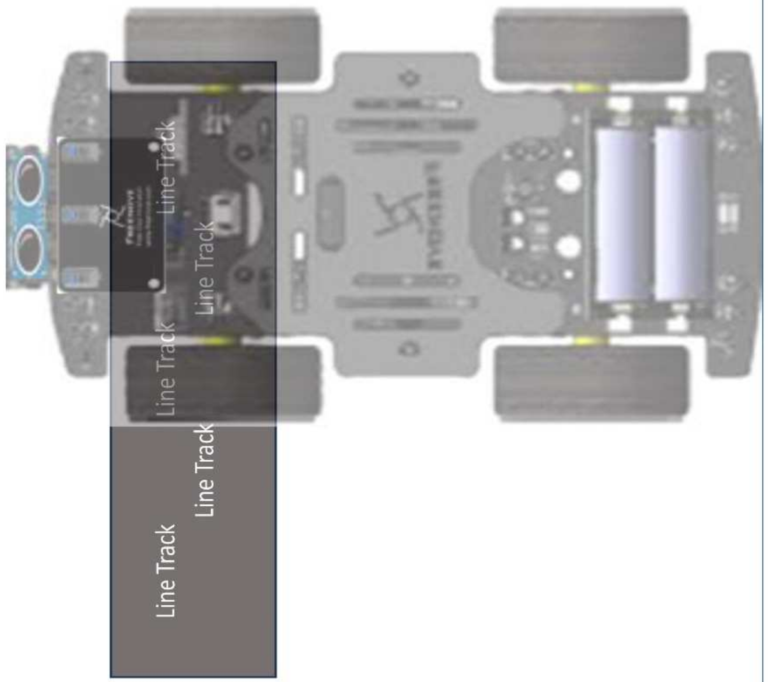
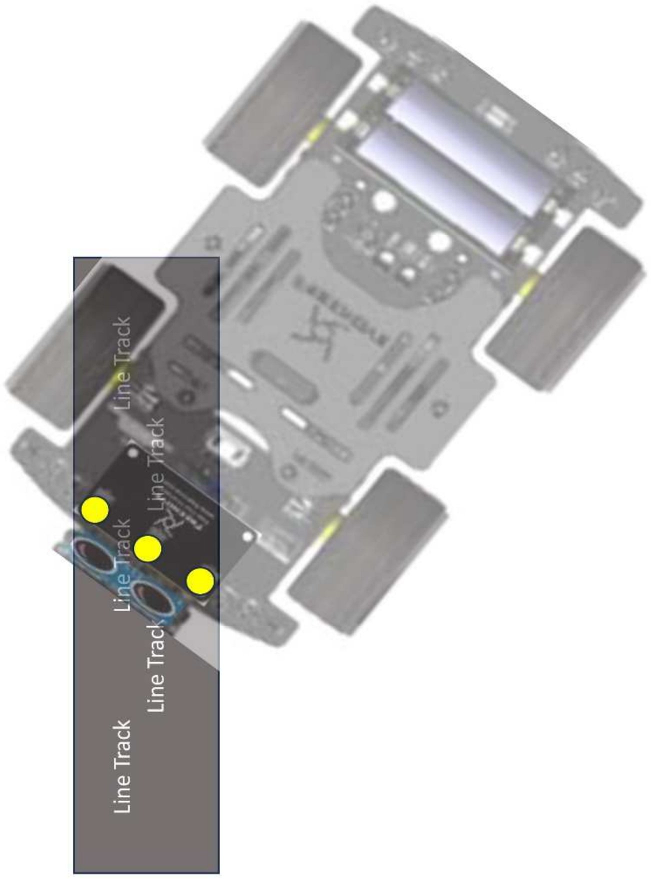
Illustration 1 (Return to Track)	***Illustration 2 (Return to Track)***
Gizmo move forwards until 000. 	Gizmo turns far from obstacle until 111 

Illustration 3 (Return to Track)

Move forward until 110 or 011.

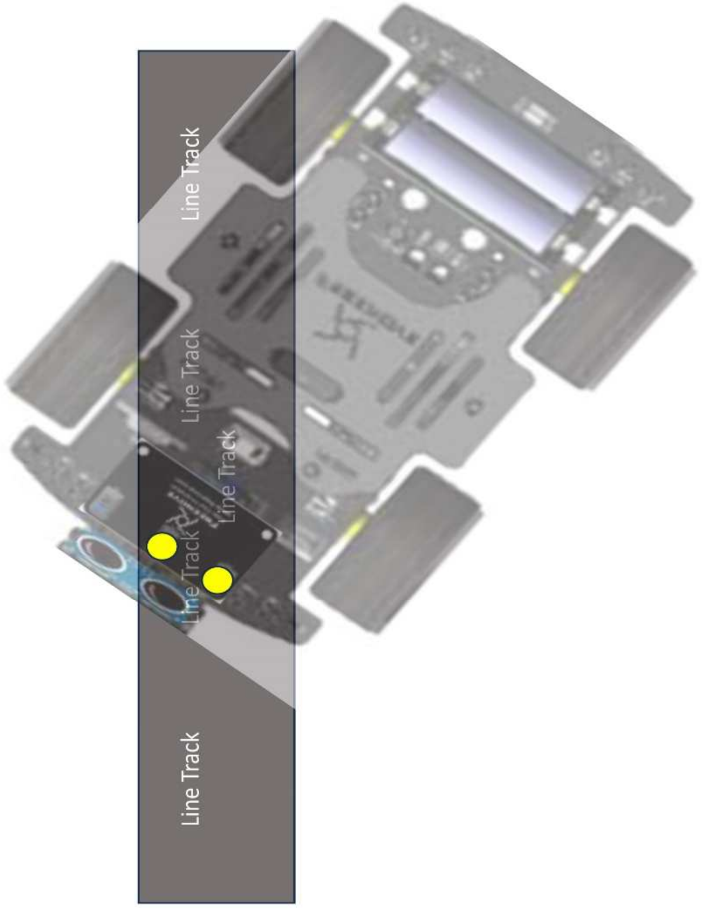
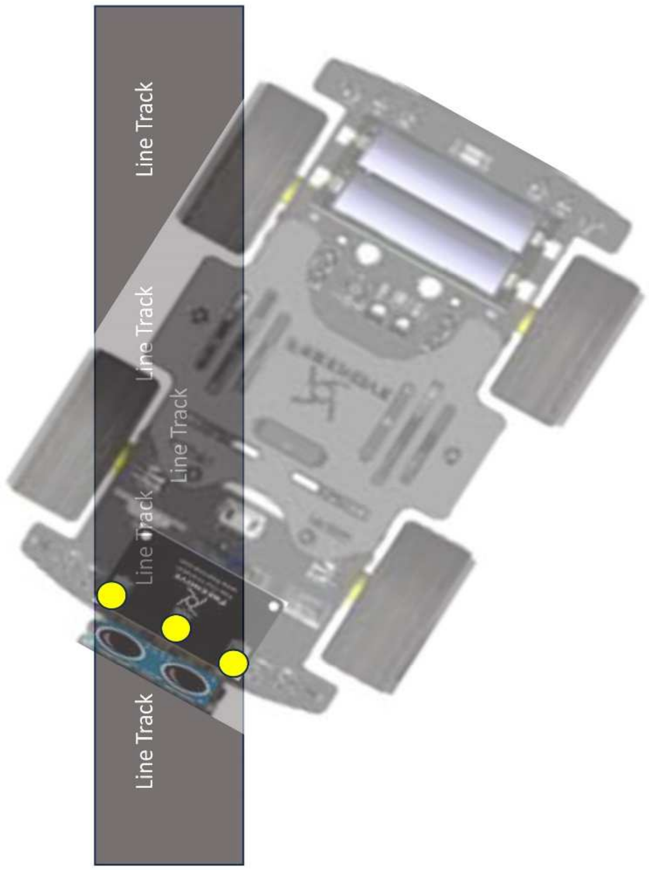


Illustration 4 (Return to Track)

Turn far from obstacle until 111.



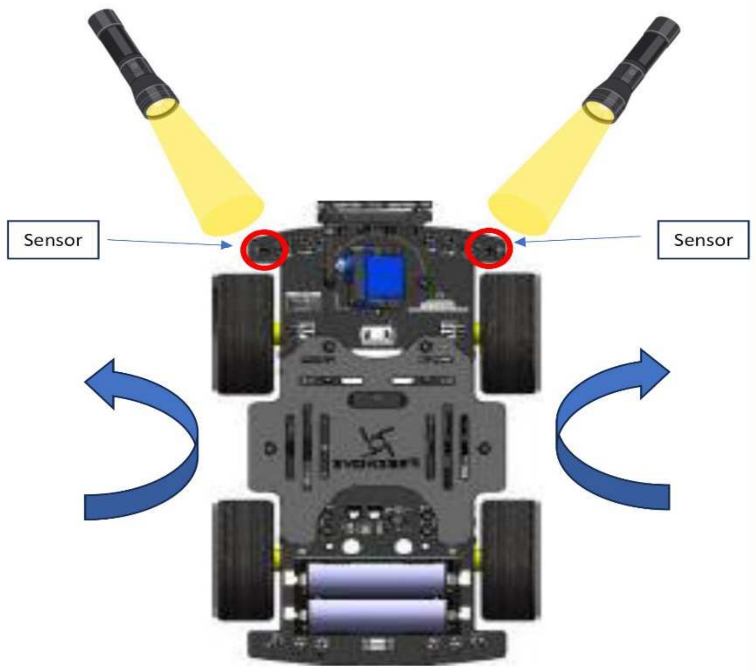
Repeat Gizmo turns far from obstacle until 111 and move forward until 110 or 011, until it is parallel to the line.

Same as Illustration 3 (Return to Track)

Light Tracing

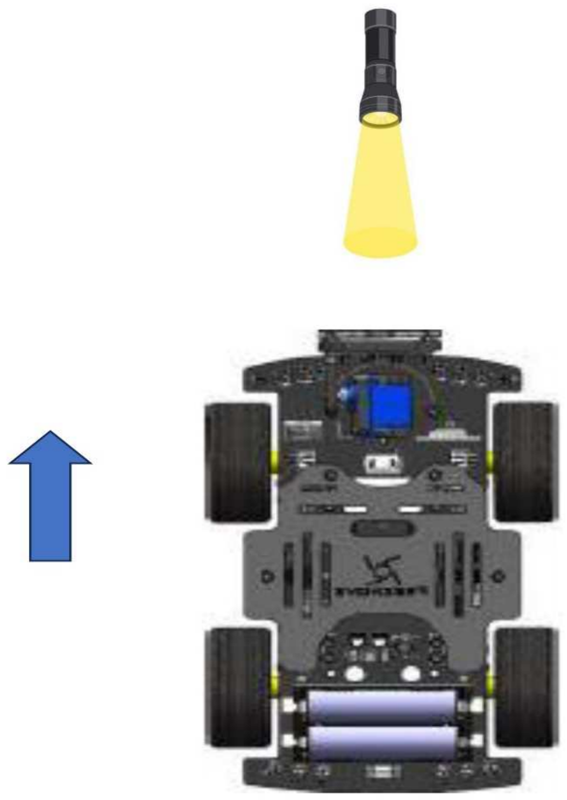
If the light is on the right sensor Gizmo rotates right

If the light is on the left sensor Gizmo rotates left



If the light is on the center of two sensor Gizmo will move forward

If there is no light Gizmo will stop



APPENDIX 3 – 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	**Pass**	
	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	**Pass**	
	3	Select development board. Click Tools on the Menu bar, move your mouse to Board: "Arduino Uno", select ESP32 Arduino and then select ESP32 Wrover Module.	Correct development board selected.	N/A	**Pass**	
	4	Select serial port. Click Tools on the 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	**Pass**	
	5	Click "Upload Using Programmer" and the program will be downloaded to Gizmo's ESP32.	Sketch successfully downloaded in Gizmo's ESP32.	N/A	**Pass**	Ensure that these libraries are added: Freenove_VK16K33_Lib_For_ESP32.zip, PCF8574.zip
	6	A message "Done Uploading" and the console will have the message "Leaving..., Hard resetting via RTS pin..."	Correct console output with no warnings or failures.	N/A	**Pass**	
	8	Unplug the USB cable from Gizmo.		N/A	**Pass**	

	9	Turn ON the power switch.	Gizmo successfully powered on.	N/A	**Pass**	
US1-002	**Line Tracking using predefined path**					
	1	Steps 1 to 9 of US1-001 successfully completed.		Use 04.2_Track_Car.ino	**Pass**	
	2	Scenario 1: With a straight-line path, Gizmo will travel from point A to point B and stop at the end of the track.	Gizmo will not derail from the path and stops at the end of the track.	N/A	**Pass**	Retested on new mat November 24, 2023 *Videos can be found in GDrive (US1-002-Scenario1)
	3	Scenario 2: With curve track, Gizmo will loop indefinitely.	Gizmo will not derail from the path.	N/A	**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-002-Scenario2)
US1-003	**Obstacle Detection**					
	1	Steps 1 to 9 of US1-001 successfully completed.		N/A	**Pass**	
	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.	**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive(US1-003-Step2)
US1-004	**Obstacle Detection and Avoidance**					
	1	Steps 1 to 9 of US1-001 successfully completed.		N/A	**Pass**	
	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 is rounded.	**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-003-Step2)
	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.		**Pass**	Retested on new mat November 24, 2023 *Videos can be found in GDrive (US1-004-Step3)

	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.		**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-004-Step4)
US1-005	**Obstacle detection and avoidance within the predefined path**					
	1	Steps 1 to 9 of US1-001 successfully completed.		N/A	**Pass**	
	2	While Gizmo is moving at a predefined path (straight path), 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 is rounded.	**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-003-Step2)
	3	Gizmo will adjust its distance with the obstacle.	Gizmo will move away if too near the obstacle or move closer if too far from the obstacle.	There is only one obstacle within the predefined path.	**Pass**	Retested on new mat November 24, 2023 *Videos can be found in GDrive (US1-004-Step3)
	4	Gizmo will continue to scan the obstacle as it navigates within the perimeter of the object.	Gizmo will avoid obstacle as it navigate around it.		**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-004-Step4)
	5	Gizmo will scan and attempts to return to the predefined path.	Gizmo will return to the predefined path.		**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-005-Step5)
	6	Repeat steps 2 to 5 with the following scenarios: Curve path with 1 obstacle Straight line path with 2 or more obstacle Curve path with 2 or more obstacle	Gizmo will avoid 1 or more obstacles and will return to the predefined path.		**Pass**	Retested on new mat November 24, 2023 *Video can be found in GDrive (US1-005-Step6)

User Story 2: Gizmo to be capable of detecting and tracking a moving light source

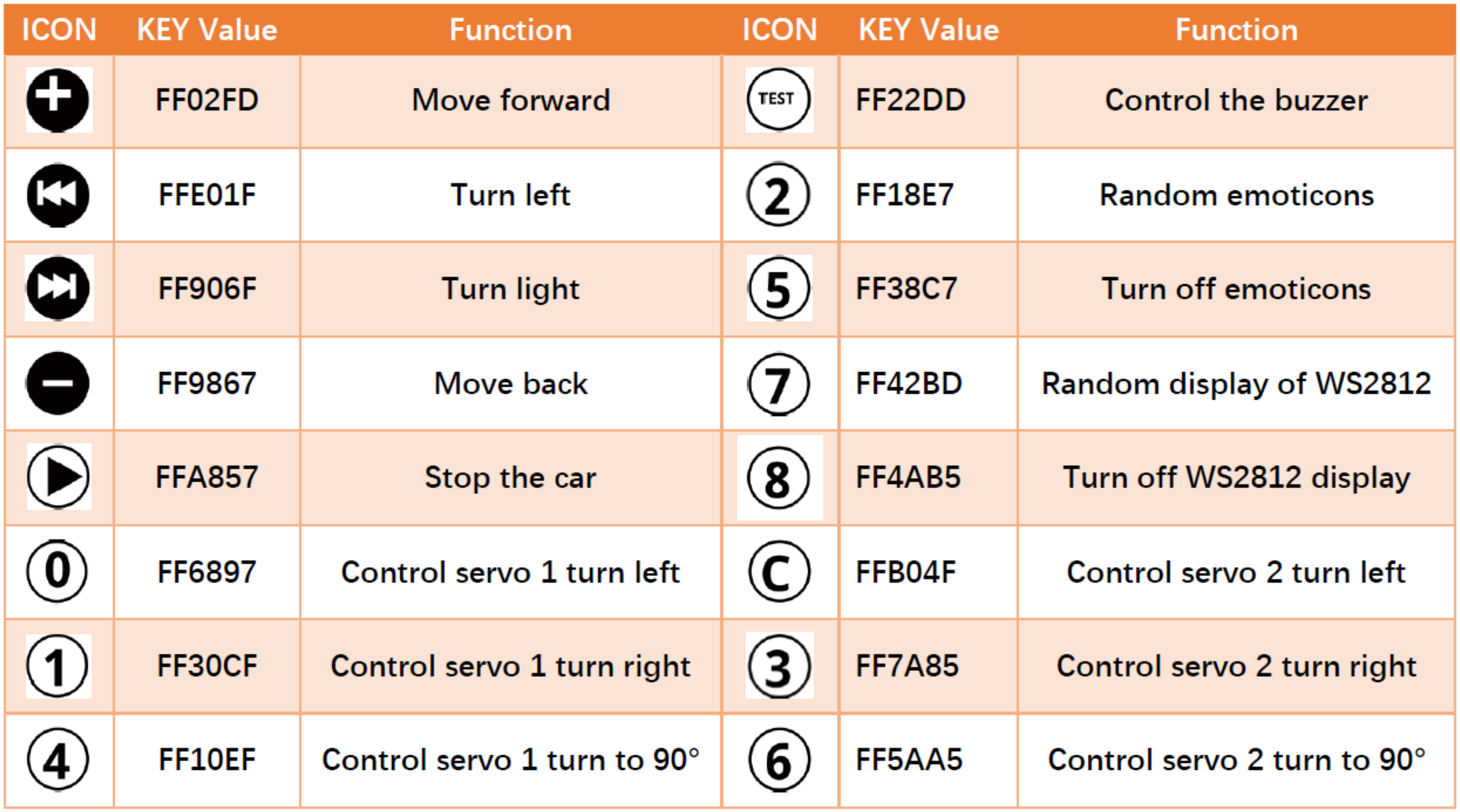
Test Case ID	Step No.	Operator Action / Input Specifications	Expected Results	Assumption / Operator Input	Status Pass / Fail	QC Comments / Actual Results
Pre-conditions: 1. **Arduino IDE, USB-SERIAL CH340 (COMx), and necessary libraries installed in Gizmo Operator workstation.** 2. **Gizmo battery is charged and installed in the battery compartment.** 3. **Necessary sensors are integrated with Gizmo's car shield and ESP32-WROVER.**						
US2-001	**Loading of Light Tracing sketch**					
	1	Connect your computer and Gizmo's ESP32 with a USB cable.	ESP2 has communication with the computer.	N/A	**Pass**	
	2	Open "03.3_Photosensitive_Car" folder in "Freenove_4WD_Car_Kit_for_ESP32\Sketches", double-click "03.3_Photosensitive_Car.ino".	Correct sketch selected.	Sketch is free of code errors	**Pass**	
	3	Select development board. Click Tolos on the Menu bar, move your mouse to Board: "Arduino Uno", select ESP32 Arduino and then select ESP32 Wrover Module.	Correct development board selected.	N/A	**Pass**	
	4	Select serial port. Click Tools on the 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	**Pass**	
	5	Click "Upload Using Programmer" and the program will be downloaded to Gizmo's ESP32.	Sketch successfully downloaded in Gizmo's ESP32.	N/A	**Pass**	
	6	A message "Done Uploading" and the console will have the message "Leaving..., Hard resetting via RTS pin..."	Correct console output with no warnings or failures.	N/A	**Pass**	
	7	Unplug the USB cable from Gizmo.		N/A	**Pass**	
	8	Turn ON the power switch.	Gizmo successfully powered on.	N/A	**Pass**	
US2-002	Light Tracing					
	1	Steps 1 to 9 of US2-001 successfully completed.		N/A	**Pass**	

	2	While Gizmo is moving, it identifies a light source at its right side within the predefined distance using its photoresistors (More bright light on the right turns right).	Gizmo will turn right.		**Pass**	*Video can be found in GDrive (US2-002)
	3	While Gizmo is moving, it identifies a light source at its left side within the predefined distance using its photoresistors (More Bright light on the left turns left).	Gizmo will turn left.	N/A	**Pass**	*Video can be found in GDrive (US2-002)
	4	While Gizmo is moving, it identifies a light source in front of the car within the predefined distance using its photoresistors (Less bright light on the left moves forward/Less bright light on the right moves backward).	Gizmo will move straight/backward.	N/A	**Pass**	*Video can be found in GDrive (US2-002)
	5	While Gizmo is moving, after turning or moving after the light source is identified, turn-off the light source.	Gizmo will stop.	N/A	**Pass**	*Video can be found in GDrive (US2-002)
US2-003	**Light tracing with obstacle detection and avoidance within the predefined path (WISHLIST)**					
	1	Steps of US1-005 successfully completed.		N/A	**Pass**	
	2	Gizmo to get off from the predefined path	Gizmo will enable light tracing capability.	N/A	**Pass**	
	3	Use a light source to guide Gizmo back to track	Gizmo will enable line tracking capability.		**Pass**	*Video can be found in GDrive (US2-003)

User Story 3: To see continuous improvements in the existing features of Gizmo

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.**						
US3-001	**Loading of sketch**					
	1	Connect your computer and Gizmo's ESP32 with a USB cable.	ESP2 has communication with the computer.	N/A	**Pass**	
	2	Open folder in "Freenove_4WD_Car_Kit_for_ESP32\Sketches", double-click sketch (*.ino) file.	Correct sketch selected.	Sketch is free of code errors	**Pass**	
	3	Select development board. Click Tools on the Menu bar, move your mouse to Board: "Arduino Uno", select ESP32 Arduino and then select ESP32 Wrover Module.	Correct development board selected.	N/A	**Pass**	
	4	Select serial port. Click Tools on the 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	**Pass**	
	5	Click "Upload Using Programmer" and the program will be downloaded to Gizmo's ESP32.	Sketch successfully downloaded in Gizmo's ESP32.	N/A	**Pass**	
	6	A message "Done Uploading" and the console will have the message "Leaving..., Hard resetting via RTS pin..."	Correct console output with no warnings or failures.	N/A	**Pass**	
	7	Unplug the USB cable from Gizmo.		N/A	**Pass**	
	8	Turn ON the power switch.	Gizmo successfully powered on.	N/A	**Pass**	

US3-002	**Gizmo functionalities**					
	1	Steps 1 to 9 of US3-001 successfully completed.		Correct sketch uploaded to perform the succeeding steps	**Pass**	
	2	Setup different surfaces and allow Gizmo scan obstacles. - Using puzzle mat - Using carpeted floor Using smooth floor or surface	Gizmo can scan obstacles and avoid them regardless of surfaces.	N/A	**Pass**	Carpeted floors should be plain design.
	3	Run Gizmo for Obstacle avoidance and line tracking for 30 minutes *** Due to current prototype state, this can only be achieved if Gizmo parts are in excellent condition**	Gizmo will continue detecting obstacle and able to track line without failure.	Battery used is fully charged.	**Partial Pass**	Gizmo had an issue with the Ultrasonic wave sensor for investigation.
	4	LED displays	Gizmo displays emotions or navigation visuals.	N/A	**Pass**	
	5	Multi-functional Infrared Car	Gizmo to perform functions using the IR remote. See Figure 2 for Remote Functions	Successfully configured IR car see Freenove Tutorial **Chapter 6.3 Multi-Functional Infrared Car**	**Pass**	*Video can be found in GDrive (US3-002-Step5)
	6	WiFi Car	Gizmo to perform functions using the Freenove Application via computer or mobile Device.	Successfully configured WiFi car see Freenove Tutorial **Chapter 7 WiFi Car**	**Pass**	*Video can be found in GDrive (US3-002-Step6)
	7	Perform Power Off	Power off Gizmo using power button.	N/A	**Pass**	



ICON	KEY Value	Function	ICON	KEY Value	Function
⊕	FF02FD	Move forward	TEST	FF22DD	Control the buzzer
⏮	FFE01F	Turn left	2	FF18E7	Random emoticons
⏭	FF906F	Turn light	5	FF38C7	Turn off emoticons
⊖	FF9867	Move back	7	FF42BD	Random display of WS2812
▶	FFA857	Stop the car	8	FF4AB5	Turn off WS2812 display
0	FF6897	Control servo 1 turn left	C	FFB04F	Control servo 2 turn left
1	FF30CF	Control servo 1 turn right	3	FF7A85	Control servo 2 turn right
4	FF10EF	Control servo 1 turn to 90°	6	FF5AA5	Control servo 2 turn to 90°

FIGURE 5: IR REMOTE FUNCTONS

APPENDIX 4 – 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 gets pingTime using pulseIn method.
8		Functional	Medium	TIM2	Gizmo accurately gets 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	RA2	Gizmo can continue to operate even there is an error in the code and will be able to be stopped using power off button.
17		Non-Functional	Low	RA3	Gizmo can be stopped using power off button.
18		Non-Functional	Low	UUS2	Gizmo will utilize LED matrix to display its actions.
19		Non-Functional	Low	UUS3	Gizmo can be powered off using the power button.
	USID-2 detecting and tracking a moving light source				
1		Functional	High	LD1	Gizmo will be able to detect light using its photoresistors.
2		Functional	High	LD2	Gizmo will be able to detect light using its photoresistors and evaluate ADC values.

3		Functional	High	LTB1	Gizmo will be able to detect light and be able to steer toward the light source.
4		Functional	High	LTB2	Gizmo will be able to detect light and be able to steer toward the light source.
5		Functional	Medium	ER3	Gizmo will be able to detect varying light source.
6		Functional	Medium	ER4	Gizmo will be able to detect varying light source and be able to steer toward the light source.
7		Non-Functional	Low	RA1	Gizmo is capable of light tracing without failure for more than 30 minutes of continuous operation.
8		Non-Functional	Low	RA2	Gizmo can continue to operate even there is an error in the code and will be able to be stopped using power off button.
9		Non-Functional	Low	RA3	Gizmo can be stopped using power off button.
10		Non-Functional	Low	UUS3	Gizmo can be powered off using the power button.
	USID-3 Improvements in the existing features of Gizmo				
		Non-Functional	Low	LLC1	Gizmo and its documentation shall adhere to the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
		Non-Functional	Low	LLC2	Copy of license shall be included in the capstone documentation.
		Non-Functional	Low	LLC3	Gizmo will be used solely for the BIA Capstone Project for Fall 2023.

		Non-Functional	Low	S1	Gizmo can accommodate integration of other features like LED lights, buzzer, and alike.
		Non-Functional	Low	S2	Gizmo can integrate to the code features like LED lights, buzzer, and alike.
		Non-Functional	Low	US1	Freenove application is user-friendly, easy to install and configure.

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**	

Category	Code	Requirement Group	Description
	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.
	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.

Category	Code	Requirement Group	Description
	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.
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**	

Category	Code	Requirement Group	Description
	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.

APPENDIX 5 – PRODUCT / SPRINT BACKLOG

ID	As a...	I want to be able to...	So that...	Priority	Status
BID001	User	Download and install Arduino IDE	I can create, edit, and save C++ codes	Sprint 1 / Milestone 2	DONE
BID002	User	Connect Gizmo to the computer using a USB cable	I can upload C++ codes for Gizmo's functionalities	Sprint 1 / Milestone 2	DONE
BID003	User	Turn Gizmo's power to ON	I can use and control Gizmo's behavior and movements	Sprint 1 / Milestone 2	DONE
BID004	User	Use Gizmo's ultrasonic sensors (circle shapes)	I can use it to detect circle-shaped obstacles in front of Gizmo	Sprint 1 / Milestone 2	DONE
BID005	User	Utilize Gizmo's ultrasonic sensors to emit waves	I can calculate the distance of the obstacle in front of Gizmo	Sprint 1 / Milestone 2	DONE
BID006	User	Combine Gizmo's direction and movement with obstacle detection calculation	Gizmo can move around the circle-shaped obstacle while detecting it ahead	Sprint 1 / Milestone 2	DONE
BID007	User	Use Gizmo's reflective optical sensors and line tracking sensors	I can use it to have Gizmo detect black-line routes on the ground for direction	Sprint 1 / Milestone 2	DONE
BID008	User	Utilize Gizmo's reflective optical sensors and line tracking sensors to emit infrared lights	I can use it to have Gizmo detect and compute the black-line route to move on	Sprint 1 / Milestone 2	DONE
BID009	User	Combine Gizmo's direction and movement with line track computation	Gizmo can move and traverse the black-line routes	Sprint 1 / Milestone 2	DONE
BID010	User	Integrate Gizmo's obstacle avoidance and line tracking capabilities (circle shapes)	I can use it to have Gizmo traverse a black-line route and avoid circle-shaped obstacles, temporarily moving out of the path, and then go back to the route	Sprint 2 / Milestone 3	DONE

BID011	User	Create a black-line route on a mat	I can use it to test Gizmo's obstacle avoidance with line tracking capabilities while in movement	Sprint 2 / Milestone 3	DONE
BID012	User	Use Gizmo's photoresistors	I can use it to have Gizmo detect the movement and direction of the light	Sprint 3 / Milestone 4	DONE
BID013	User	Utilize Gizmo's photoresistors to get ADC values from the detection of light	I can use it to have Gizmo determine and follow the direction of light	Sprint 3 / Milestone 4	DONE
BID014	User	Use Gizmo's ultrasonic sensors (rectangle shapes)	I can use it to detect rectangle-shaped obstacles in front of Gizmo	Sprint 3 / Milestone 4	DONE
BID015	User	Integrate Gizmo's obstacle avoidance and line tracking capabilities (circle and rectangle shapes)	I can use it to have Gizmo traverse a black-line route and avoid circle and rectangle shaped obstacles, temporarily moving out of the path, and then go back to the route	Sprint 3 / Milestone 4	DONE
BID016	User	Display Gizmo's eyes rotating via LED lights	I can add and see Gizmo's eye functionality to rotate	Sprint 3 / Milestone 4	DONE
BID017	User	Display Gizmo's eyes to blink via LED lights	I can add and see Gizmo's eye functionality to blink	Sprint 3 / Milestone 4	DONE
BID018	User	Display Gizmo's eyes to smile via LED lights	I can add and see Gizmo's eye functionality to smile	Sprint 3 / Milestone 4	DONE
BID019	User	Display Gizmo's eyes to cry via LED lights	I can add and see Gizmo's eye functionality to cry	Sprint 3 / Milestone 4	DONE
BID020	User	Install Freenove's mobile application via App Store (for iOS phones)	I can setup Gizmo to be controlled using Freenove's mobile application	Sprint 3 / Milestone 4	DONE
BID021	User	Install Freenove's mobile application via Google Play Store (for Android phones)	I can setup Gizmo to be controlled using Freenove's mobile application	Sprint 3 / Milestone 4	DONE

BID022	User	Install Freenove's desktop PC application	I can setup Gizmo to be controlled using Freenove's desktop PC application	Sprint 3 / Milestone 4	DONE
BID023	User	Set Gizmo's Wi-Fi connection password	Connect to Gizmo via Wi-Fi Access Point	Sprint 3 / Milestone 4	DONE
BID024	User	Set Gizmo's hotspot connection password	Connect to Gizmo via its Hotspot Network using ESP32	Sprint 3 / Milestone 4	DONE
BID025	User	Select Gizmo's network in the list of available connection networks in my mobile phone	I can establish network connection to access Gizmo	Sprint 3 / Milestone 4	DONE
BID026	User	Select Gizmo's network in the list of available connection networks in my desktop PC	I can establish network connection to access Gizmo	Sprint 3 / Milestone 4	DONE
BID027	User	Open and use Freenove's mobile application	Select 4WD Car for ESP32 which is Gizmo's device type	Sprint 3 / Milestone 4	DONE
BID028	User	Open and use Freenove's desktop PC application	Select 4WD Car for ESP32 which is Gizmo's device type	Sprint 3 / Milestone 4	DONE
BID029	User	Enter Gizmo's IP address in the mobile application	Set Gizmo's IP address to connect to	Sprint 3 / Milestone 4	DONE
BID030	User	Enter Gizmo's IP address in the desktop PC application	Set Gizmo's IP address to connect to	Sprint 3 / Milestone 4	DONE
BID031	User	Press the connect button in the mobile application	I can connect the mobile application to Gizmo's network	Sprint 3 / Milestone 4	DONE
BID032	User	Press the connect button in the desktop PC application	I can connect the desktop PC application to Gizmo's network	Sprint 3 / Milestone 4	DONE
BID033	User	Use Freenove's mobile application to use the control car running feature	I can control Gizmo's direction and movement using mobile application	Sprint 3 / Milestone 4	DONE
BID034	User	Use Freenove's mobile application to use the control camera feature	I can control Gizmo's camera angle and position using mobile application	Sprint 3 / Milestone 4	DONE

BID035	User	Use Freenove's mobile application to display Gizmo's LED emotion feature	I can show Gizmo's LED emotions using mobile application	Sprint 3 / Milestone 4	DONE
BID036	User	Use Freenove's mobile application to display Gizmo's RGB LED lights feature	I can show Gizmo's RBG LED emotions using mobile application	Sprint 3 / Milestone 4	DONE
BID037	User	Use Freenove's mobile application to produce Gizmo's buzzer feature	I can show Gizmo's buzzer sound using mobile application	Sprint 3 / Milestone 4	DONE
BID038	User	Use Freenove's desktop PC application to click the Turn Left button	Gizmo can turn left	Sprint 3 / Milestone 4	DONE
BID039	User	Use Freenove's desktop PC application to click the Turn Right button	Gizmo can turn right	Sprint 3 / Milestone 4	DONE
BID040	User	Use Freenove's desktop PC application to click the Forward button	Gizmo can move forward	Sprint 3 / Milestone 4	DONE
BID041	User	Use Freenove's desktop PC application to click the Backward button	Gizmo can move backward	Sprint 3 / Milestone 4	DONE
BID042	User	Use Freenove's desktop PC application to click the Up button	Gizmo's servo/camera angle turn to face up	Sprint 3 / Milestone 4	DONE
BID043	User	Use Freenove's desktop PC application to click the Down button	Gizmo's servo/camera angle turn to face down	Sprint 3 / Milestone 4	DONE
BID044	User	Use Freenove's desktop PC application to click the Left button	Gizmo's servo/camera angle turn to face left	Sprint 3 / Milestone 4	DONE
BID045	User	Use Freenove's desktop PC application to click the Right button	Gizmo's servo/camera angle turn to face right	Sprint 3 / Milestone 4	DONE
BID046	User	Use Freenove's desktop PC application to set Gizmo's RGB color	I can set and show Gizmo's RGB LED lights	Sprint 3 / Milestone 4	DONE

BID047	User	Use Freenove's desktop PC application to set Gizmo's RGB color mode	I can set and show Gizmo's RGB LED lights according to different modes	Sprint 3 / Milestone 4	DONE
BID048	User	Use Freenove's desktop PC application to display Gizmo's battery level	I can see and check Gizmo's battery level	Sprint 3 / Milestone 4	DONE
BID049	User	Put batteries in Remote Control	I can use the remote control to navigate Gizmo's movements via Infrared	Sprint 3 / Milestone 4	DONE
BID050	User	Click the power button on the Remote Control	I can use the remote control to use Gizmo's movements via Infrared	Sprint 3 / Milestone 4	DONE
BID051	User	Click the plus (+) button on the Remote Control	I can use the remote control to use Gizmo's move forward via Infrared	Sprint 3 / Milestone 4	DONE
BID052	User	Click the rewind (<<) button on the Remote Control	I can use the remote control to Gizmo's turn left via Infrared	Sprint 3 / Milestone 4	DONE
BID053	User	Click the forward (>>) button on the Remote Control	I can use the remote control to Gizmo's turn right via Infrared	Sprint 3 / Milestone 4	DONE
BID054	User	Click the minus (-) button on the Remote Control	I can use the remote control to Gizmo's to move back via Infrared	Sprint 3 / Milestone 4	DONE
BID055	User	Click the play (>) button on the Remote Control	I can use the remote control to Gizmo's to stop via Infrared	Sprint 3 / Milestone 4	DONE
BID056	User	Click the 0 button on the Remote Control	I can use the remote control to Gizmo's control servo 1 turn left via Infrared	Sprint 3 / Milestone 4	DONE
BID057	User	Click the 1 button on the Remote Control	I can use the remote control to Gizmo's control servo 1 turn right via Infrared	Sprint 3 / Milestone 4	DONE

BID058	User	Click the 2 button on the Remote Control	I can use the remote control to Gizmo's to turn on displaying random emoticons via Infrared	Sprint 3 / Milestone 4	DONE
BID059	User	Click the 3 button on the Remote Control	I can use the remote control to Gizmo's control servo 2 turn right via Infrared	Sprint 3 / Milestone 4	DONE
BID060	User	Click the 4 button on the Remote Control	I can use the remote control to Gizmo's control servo 1 turn 90-degree angle via Infrared	Sprint 3 / Milestone 4	DONE
BID061	User	Click the 5 button on the Remote Control	I can use the remote control to Gizmo's to turn off displaying random emoticons via Infrared	Sprint 3 / Milestone 4	DONE
BID062	User	Click the 6 button on the Remote Control	I can use the remote control to Gizmo's control servo 2 turn 90-degree angle via Infrared	Sprint 3 / Milestone 4	DONE
BID063	User	Click the 7 button on the Remote Control	I can use the remote control to turn on Gizmo's random display of WS2812 via Infrared	Sprint 3 / Milestone 4	DONE
BID064	User	Click the 8 button on the Remote Control	I can use the remote control to turn off Gizmo's random display of WS2812 via Infrared	Sprint 3 / Milestone 4	DONE
BID065	User	Click the C button on the Remote Control	I can use the remote control to Gizmo's control servo 2 turn left via Infrared	Sprint 3 / Milestone 4	DONE
BID066	User	Click the test button on the Remote Control	I can use the remote control to Gizmo's buzzer sound via Infrared	Sprint 3 / Milestone 4	DONE

APPENDIX 6 – DEPLOYMENT GUIDE

OVERVIEW

The purpose of this Deployment Guide is to describe in technical terms the steps necessary from running ESP32 code to uploading it to Assembled Hardware using Arduino and make it operational.

PREREQUISITES:

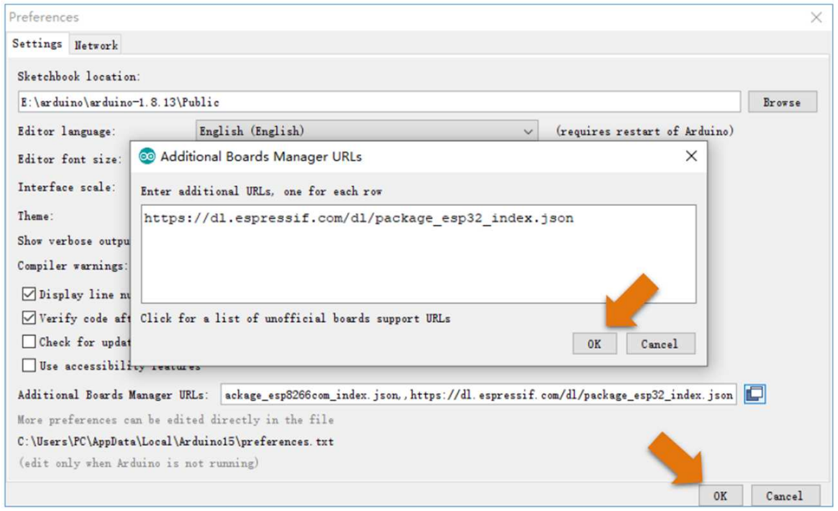
1. Arduino IDE:

Download and install the Arduino IDE from the official website.
https://www.arduino.cc/en/software

2. ESP32 Board Support:

Open the Arduino IDE.

Go to File > Preferences.
Enter the following URL into the "Additional Boards Manager URLs" field:
https://dl.espressif.com/dl/package_esp32_index.json



Preferences

Settings | Network

Sketchbook location: E:\arduino\arduino-1.8.13\Public

Editor language: English (English) (requires restart of Arduino)

Editor font size:

Interface scale:

Theme:

Show verbose outpu

Compiler warnings:

[✓] Display line nu

[✓] Verify code aft

[ ] Check for updat

[ ] Use accessibility features

Additional Boards Manager URLs

Enter additional URLs, one for each row

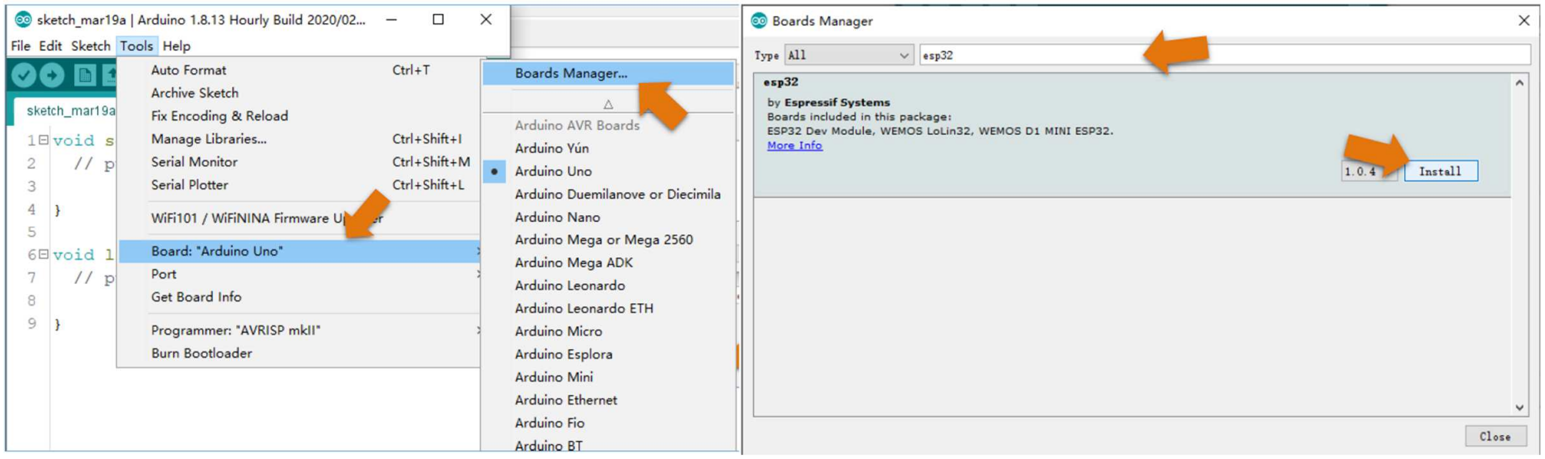
```
https://dl.espressif.com/dl/package_esp32_index.json
```

Click for a list of unofficial boards support URLs

Additional Boards Manager URLs: ackage_esp8266com_index.json,,https://dl.espressif.com/dl/package_esp32_index.json

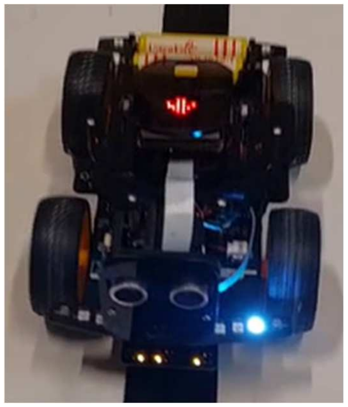
More preferences can be edited directly in the file
C:\Users\PC\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Click "OK" to close the Preferences window.
Go to Tools > Board > Boards Manager..., search for "esp32," and install the ESP32 board support package.



esp32
by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

3. Assembled Hardware



Battery is charged and installed in the battery compartment

Necessary sensors are integrated with car shield and ESP32-WROVER.

DEPLOYMENT STEPS:

1. Connect the Freenove ESP32 to your computer:

   Use a USB cable to connect the ESP32 to your computer.

2. Open Arduino IDE:

   Open the Arduino IDE on your computer.

3. Select ESP32 Board:

Go to Tools > Board and select your ESP32 board. Choose the specific ESP32 module you are using (e.g., ESP32-WROVER).

4. Select Port:

   Go to Tools > Port and select the port to which your ESP32 is connected.

5. Select Speed:

   Go to Tools > Upload Speed and select the speed you are using.

6. Write and Upload Your Code:

   Write or open your project code in the Arduino IDE.

   Make sure your code is compatible with the ESP32 platform.
   Click the right arrow button (Upload) to compile and upload your code to the ESP32.
   *For information about Arduino coding visit https://docs.arduino.cc/learn/

7. Monitor Serial Output (Optional)

   If your code includes serial output (e.g., Serial.begin()), open the Serial Monitor (Tools > Serial Monitor) to view debug messages.

8. Power Configuration (if necessary):

   Ensure that your ESP32 is powered correctly according to your project requirements.

9. Verify Operation:

   Verify that your project is operating as expected. Check for any errors or unexpected behavior.

TROUBLESHOOTING:

If you encounter issues during deployment, check the error messages in the Arduino IDE.

Ensure that your code is written to be compatible with the ESP32 platform.

Verify your connections and power supply.

ADDITIONAL NOTES:

Freenove may provide specific instructions or libraries for their ESP32 modules. Check the documentation or resources provided by Freenove for any additional steps or requirements.

APPENDIX 7 – USER GUIDE

1. INTRODUCTION

1.1 Overview

Welcome to the Freenove ESP32-WROVER robot car User Guide! This guide is designed around three user stories: autonomously detecting and avoiding obstacles, detecting, and tracking a moving light source, and continuous improvements in existing features.

1.2 Purpose

This guide aims to help you implement specific functionalities on your ESP32-WROVER robot car based on the provided user stories.

1.3 System Requirements

Ensure you have the ESP32-WROVER development board, a computer with the Arduino IDE installed, and additional peripherals required for specific user stories.

2. GETTING STARTED

2.1 Unboxing

Open the Freenove ESP32-WROVER package.

Confirm that all components, including the ESP32-WROVER board, USB cable, and user manual, are included.

2.2 Hardware Setup

Connect the ESP32-WROVER to a power source using the provided USB cable.

Attach any additional peripherals required for your specific user story.

2.3 Software Installation

Download and install the Arduino IDE from arduino.cc.

Open Arduino IDE.

3. USER STORY 1: AUTONOMOUS OBSTACLE AVOIDANCE

3.1 Background

Imagine creating a robot car capable of autonomously detecting and avoiding obstacles in its path using the ESP32-WROVER.

3.2 Objective

Build a robot car that can navigate its environment and avoid collisions by autonomously detecting obstacles.

3.3 Implementation Steps

Connect distance sensors (e.g., ultrasonic sensors) to the ESP32-WROVER.

Get sketch *07.1_Line_Tracking_with_Obstacle_Avoidance* from Github (https://github.com/rwoofanshawe/gizmo-mgmt6134)

Upload code to read sensor data and interpret it to detect obstacles.

Implement motor control to navigate around obstacles.

Test the robot car in an obstacle-rich environment.

3.4 Expected Outcome

The robot car should autonomously navigate its environment, detect obstacles, and change its path to avoid collisions.

4. USER STORY 2: LIGHT SOURCE TRACKING

4.1 Background

In this user story, we aim to create a robot car using the ESP32-WROVER that can detect and track a moving light source. This could be particularly useful for applications like solar tracking or following a light-emitting object.

4.2 Objective

Build a robot car capable of detecting a moving light source and adjusting its position or orientation accordingly.

4.3 Implementation Steps

Connect a light sensor (e.g., LDR or photodiode) to the ESP32-WROVER.

Get sketch *03.3_Photosensitive_Car* from Github (https://github.com/rwoofanshawe/gizmo-mgmt6134)

Upload code to read sensor data and interpret changes in light intensity.

Implement motor control or servo mechanisms to adjust the position of a sensor or actuator in response to changes in the light source's position.

Test the robot car in an environment with a moving light source.

4.4 Expected Outcome

The robot car should effectively track a moving light source, adjusting its position or orientation based on changes in light intensity.

5. USER STORY 3: CONTINUOUS FEATURE IMPROVEMENTS

5.1 Background

This user story focuses on continuous improvement in existing features of the ESP32-WROVER robot car. It encourages a mindset of ongoing refinement and enhancement.

5.2 Objective

Identify an existing feature of your ESP32-WROVER robot car and make continuous improvements to enhance its functionality, efficiency, or user experience.

5.3 Implementation Steps

Choose a specific feature or aspect of your ESP32-WROVER robot car for improvement.

Get sketches from Freenove_4WD_Car_Kit_for_ESP32\Sketches

Analyze the current implementation and identify areas for enhancement.

Implement changes or additions to improve the selected feature.

Test the updated feature to ensure it performs better than the previous version.

5.4 Expected Outcome

The selected feature should demonstrate noticeable improvement, whether in terms of performance, usability, or any other relevant aspect.

6. TROUBLESHOOTING

6.1 Common Issues

Issue: Connectivity Problems

Solution: Check power connections and ensure the ESP32-WROVER is properly connected. Verify Wi-Fi credentials and network availability.

Issue: Sensor Readings Inaccurate

Solution: Calibrate sensors and ensure they are placed correctly. Check for interference from other electronic devices.

6.2 Debugging Tips

Serial Debugging: Use the Serial Monitor in the Arduino IDE for debugging. Print relevant variables and messages to trace the execution of your code.

LED Indicators: Implement LED indicators in your code to signal specific states, aiding in debugging without the Serial Monitor.

6.3 Online Support Resources

If you encounter issues beyond the scope of this guide, seek assistance from online resources:

Freenove Forums: Visit the Freenove community forums to ask questions and share experiences.

Arduino Community: The Arduino community is a valuable resource for ESP32-related issues.