



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

Milestone 4

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

1. Project Summary..... 3

1.1 Introduction 3

1.2 Project Milestone Summary 3

2. Deployment Diagram 5

3. API Functions Used 6

4. Test Cases..... 7

5. Coding Standards 7

6. Gantt Chart..... 9

7. Recommendations 9

8. SUMMARY 9

9. References 10

APPENDIX 1 – Test Cases 11

APPENDIX 2 – Gantt Chart 19

APPENDIX 3 – Deployment Guide..... 21

APPENDIX 4 – USER Guide 24

LIST OF TABLES

Table 1: Project Milestone Summary..... 3

Table 2: Project Status Summary..... 4

Table 3: Project Milestone Schedule 4

LIST OF FIGURES

Figure 1: Deployment Diagram 5

Figure 2: IR Remote Functions 18

1. PROJECT SUMMARY

1.1 INTRODUCTION

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

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

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

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

1.2 PROJECT MILESTONE SUMMARY

High-Level Milestone Timeline				
Milestone	Description	Start Date	Status	Completion Date
1	Inception	11 September 2023	Completed	27 September 2023
2	Analysis of Deliverables	28 September 2023	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	Not Started	8 December 2023

TABLE 1: PROJECT MILESTONE SUMMARY

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

TABLE 2: PROJECT STATUS SUMMARY

Milestone Event Table			
Milestone	Status	Due Date	Expected Completion Date
Milestone 1	Completed	September 27, 2023	September 27, 2023
Milestone 2	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	In Progress	December 8, 2023	

TABLE 3: PROJECT MILESTONE SCHEDULE

2. DEPLOYMENT DIAGRAM

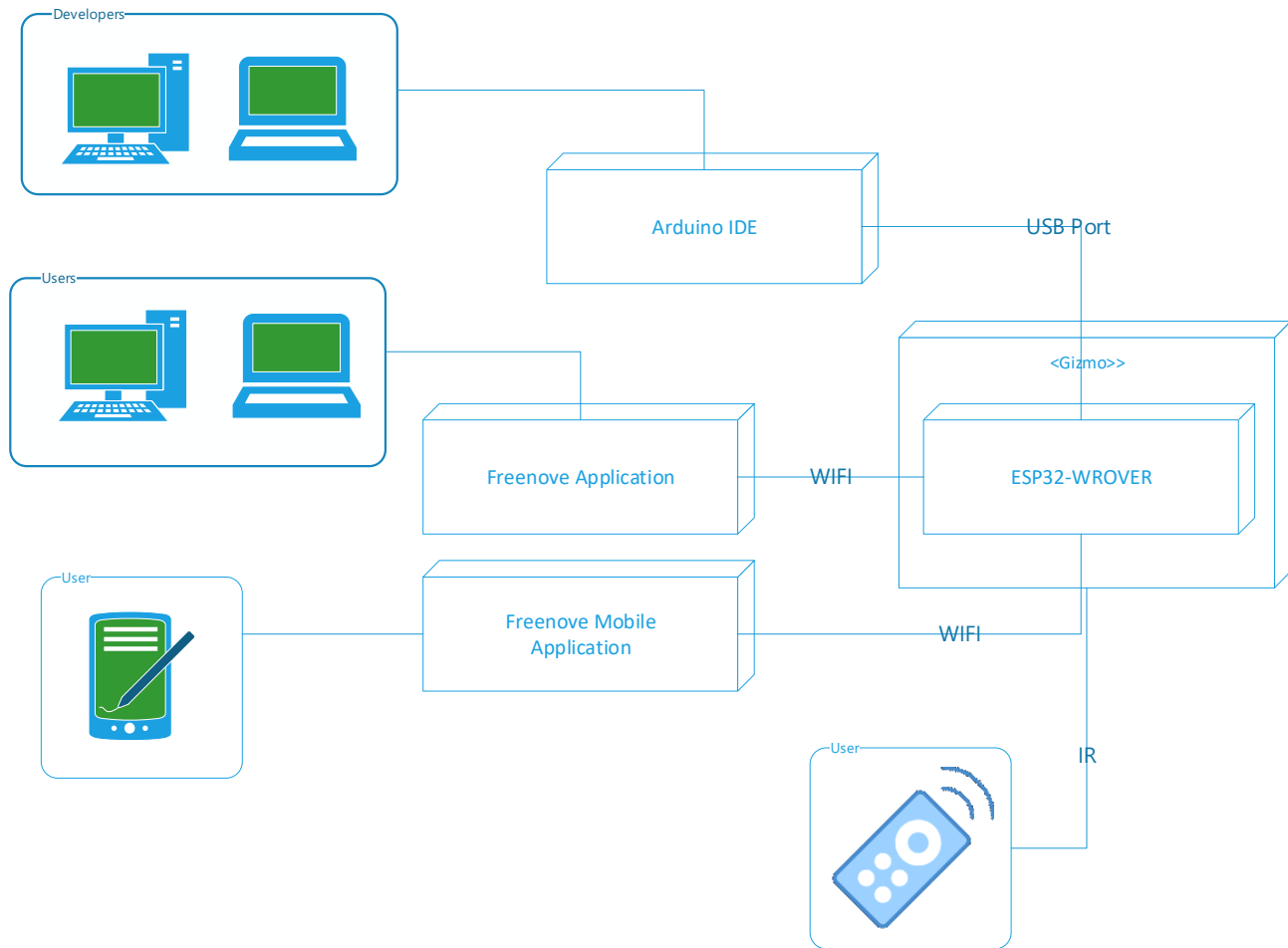


FIGURE 1: DEPLOYMENT DIAGRAM

- 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.

3. 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.

4. TEST CASES

See [Appendix 1](#).

5. 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.

6. GANTT CHART

See [Appendix 2](#).

7. 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.

8. 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.

9. REFERENCES

Freenove (n.d.). *Tutorial.pdf*.

https://github.com/Freenove/Freenove_4WD_Car_Kit_for_ESP32/Blob/master/Tutorial.pdf

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

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

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&gl=1%2A92kqy2%2A_ga%2AMTc5MjMwNzk4MC4xNjk0MTkyMDY4%2A_ga_NEXN8H46L5%2AMTY5OTk4NjQzMC4xMi4xLjE2OTk5ODY1NTQuMC4wLjA.

APPENDIX 1 – 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:						
<ol style="list-style-type: none"> 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

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

	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)

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

	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:						
<ol style="list-style-type: none"> 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	

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

	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: <ol style="list-style-type: none"> 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	

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

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	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	

















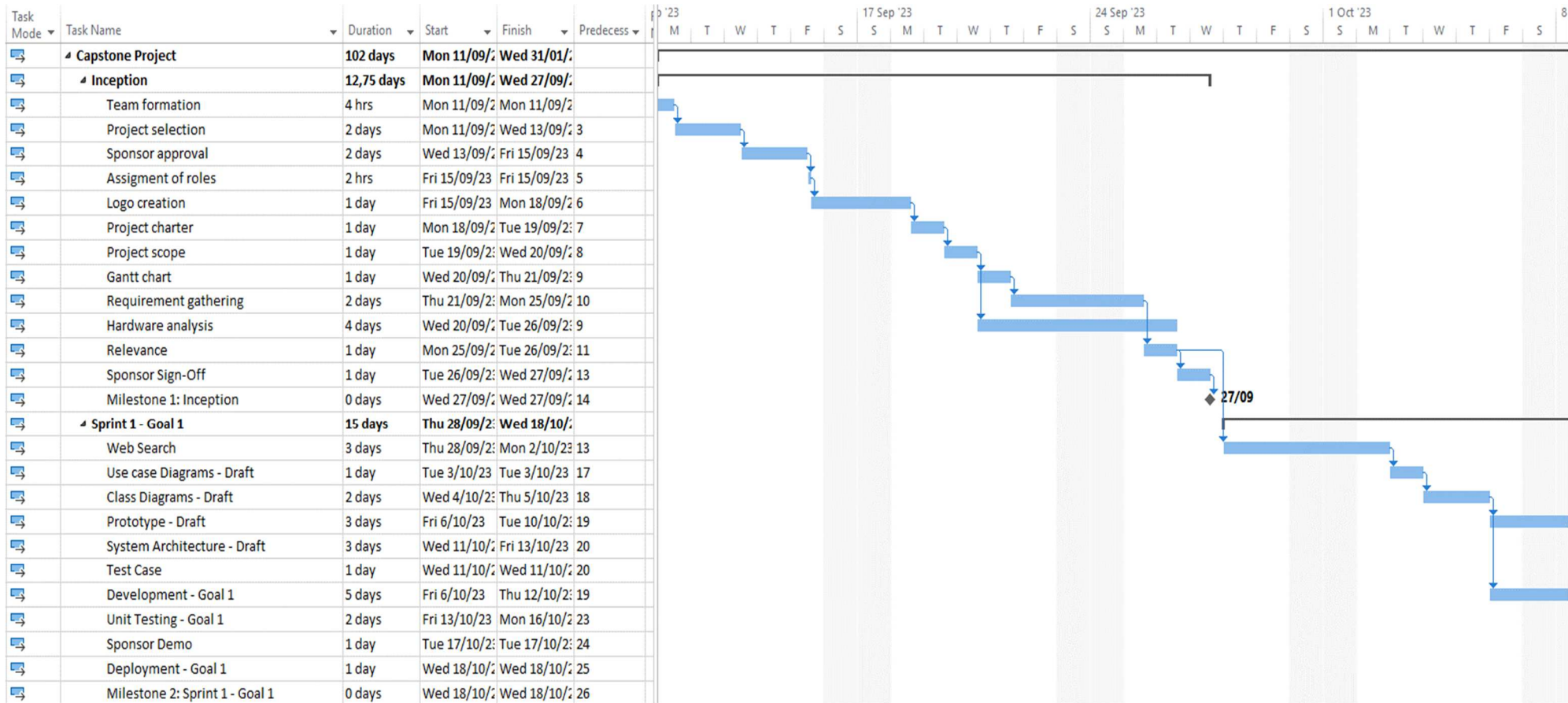
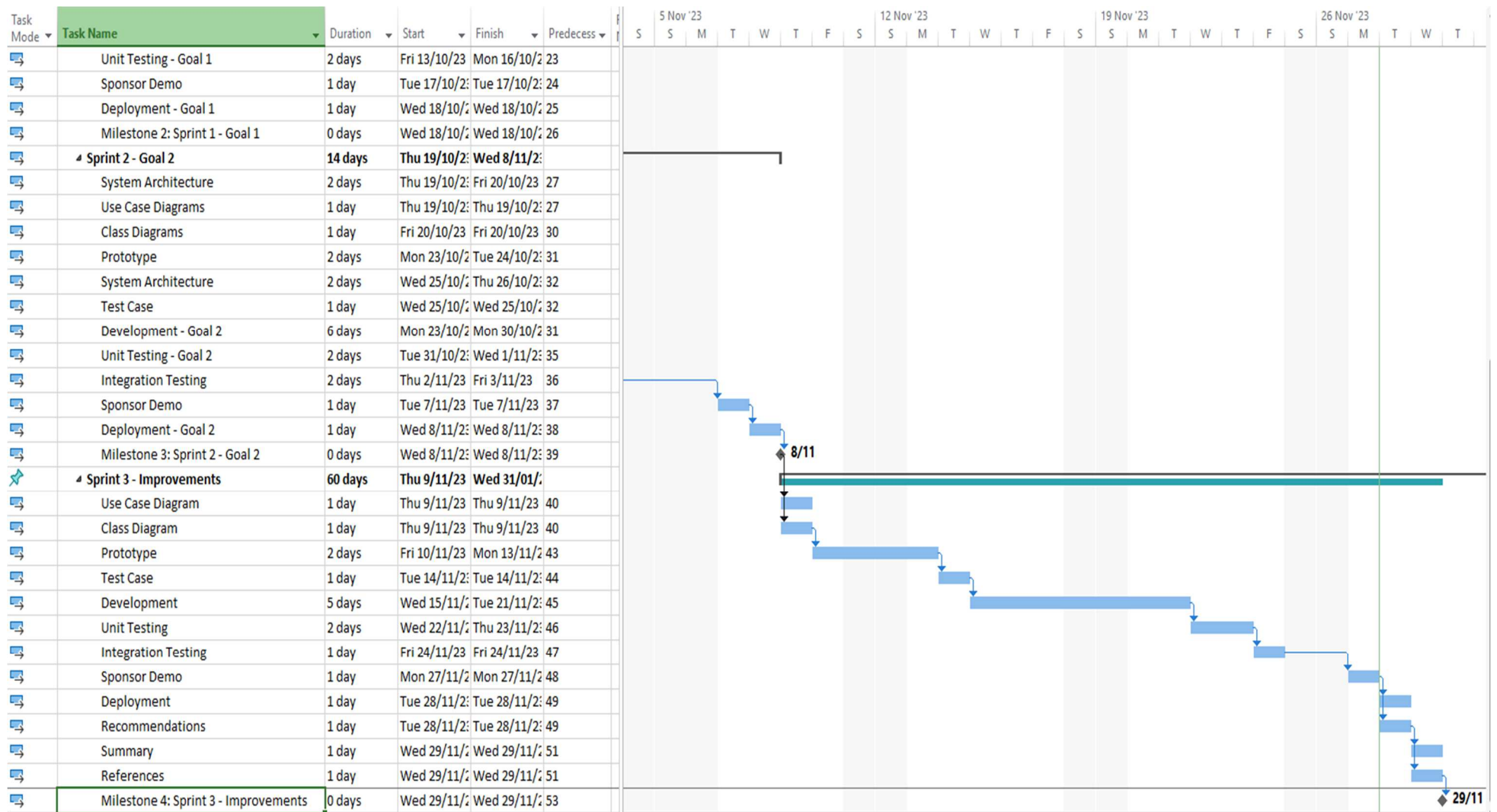
ICON	KEY Value	Function	ICON	KEY Value	Function
	FF02FD	Move forward		FF22DD	Control the buzzer
	FFE01F	Turn left		FF18E7	Random emoticons
	FF906F	Turn right		FF38C7	Turn off emoticons
	FF9867	Move back		FF42BD	Random display of WS2812
	FFA857	Stop the car		FF4AB5	Turn off WS2812 display
	FF6897	Control servo 1 turn left		FFB04F	Control servo 2 turn left
	FF30CF	Control servo 1 turn right		FF7A85	Control servo 2 turn right
	FF10EF	Control servo 1 turn to 90°		FF5AA5	Control servo 2 turn to 90°

FIGURE 2: IR REMOTE FUNCTIONS

APPENDIX 2 – GANTT CHART



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



APPENDIX 3 – 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>

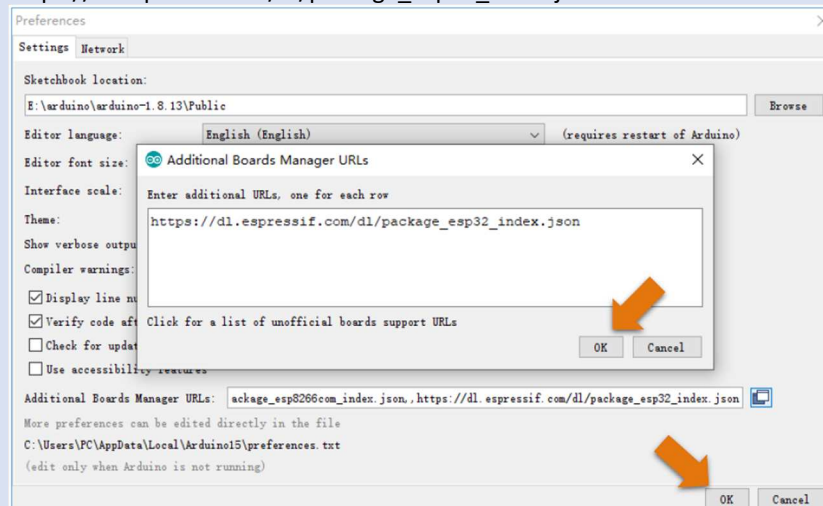
2. ESP32 Board Support:

Open the Arduino IDE.

Go to File > Preferences.

Enter the following URL into the "Additional Boards Manager URLs" field:

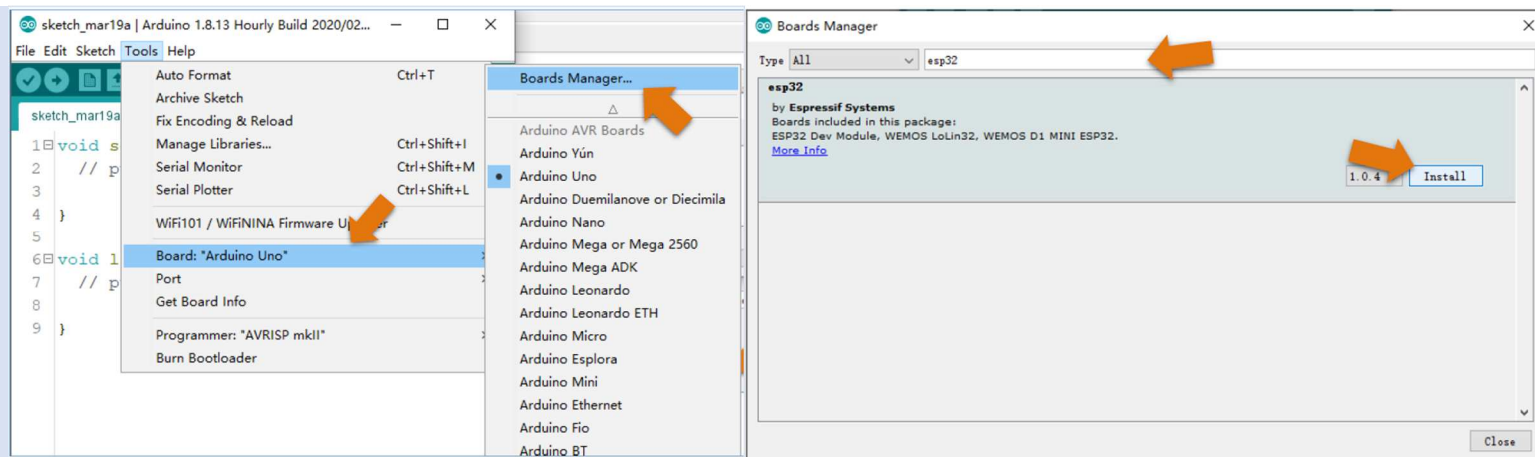
https://dl.espressif.com/dl/package_esp32_index.json



Click "OK" to close the Preferences window.

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

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



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:

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

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 4 – 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.](https://www.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.