REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

ROBOKART HOME LIVE

20011030 — Muhammed KASAP

SENIOR PROJECT

Advisor
Assist. Prof. Dr. Furkan ÇAKMAK

Haziran 2024

ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to my advisor, Assist. Prof. Dr. Furkan ÇAKMAK, who assisted me from the initial to the final stages of this project and did everything possible to ensure its successful completion.

His enthusiastic and curious attitude towards the project's developments positively influenced me, and even in situations where there was no progress, his cheerful demeanor calmed me down and prevented me from avoiding communication.

I am grateful to him for his guidance and mentorship throughout this project.

Muhammed KASAP

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

HTTP	Hypertext Transfer Protocol
ROS	Robot Operation System
UI	User Interface
Wi-Fi	Wireless Fidelity

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

ROBOKART HOME LIVE

Muhammed KASAP

Department of Computer Engineering
Senior Project

Advisor: Assist. Prof. Dr. Furkan ÇAKMAK

The RoboKart Home Live project offers a new way to play racing games. You can control the vehicle on your phone and see the game map in the real world. Our game connects to a robot through our app, turning the robot into the game's vehicle. Using the ROS operating system, we send messages to the robot to move in real life. With our mobile app made with Flutter and Firebase, you can control the robot, make and choose maps, and compare your scores with others after time-based races.

Keywords: ROS, Flutter, Firebase, Game, Racing Game, Real World, Robotics

ÖZET

ROBOKART HOME LIVE

Muhammed KASAP

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr. Ögr. Üyesi Furkan ÇAKMAK

RoboKart Home Live projesi, yarış oyunlarına yeni bir soluk getirerek telefon ekranında aracı hareket ettirmeyi ve haritayı gerçek dünyada görebilmeyi mümkün hale getiriyor. Elimizdeki uygulamadan bir robota bağlanarak, robotu oyunumuzun aracı haline getiren bu oyun, ROS işletim sistemi vasıtasıyla gerçek dünya üzerinde hareket eden robota gerekli mesajları yolluyor. Flutter ve Firebase ile yapılan mobil uygulama sayesinde robotu hareket ettiren kullanıcılar, harita oluşturma, harita seçme ve zamana karşı yaptıkları yarış sonunda diğer kullanıcıların skorlarını ve kendi skorlarıyla karşılaştırma imkânı buluyor.

Anahtar Kelimeler: ROS, Flutter, Firebase, Oyun , Yarış Oyunu, Gerçek Dünya, Robotik

1 Introduction

The fast evolution of technology is bringing up new perceptions regarding robotics and mobile gaming. This project works on the development of an innovative system, Robokart Homelive, using Robot Operating System (ROS) and Flutter. The system integrates robot control with a racing game that can be enjoyed by the user on their phone.

This project is inspired by the interest in exploring the combination of robotics with gaming for increasing user experience. By making use of ROS for communication with the robots and controlling them and Flutter for building the mobile application, we aim to provide users with an opportunity to control real robots while enjoying their actions in the racing game on the app.

This is not only a typical robotics application for entertainment but also a fresh approach in combining robotics with mobile gaming, realizing the power and joy of building user experiences between ROS and Flutter. Our goal is to ensure that combining ROS and Flutter for creating interactive and fun experiences is both possible and thrilling.

2
Literature Review

The open-source platform ROS is continuously growing; with its help, and with ubiquitous Wi-Fi connectivity or the assistance of the complementary Android communication mode, it is possible to hook up your Android device with your toy robot. There has been real testing, and it has been shown that there are ROS libraries available on Android. A robot's webcam, using ROS nodes, can be used to stream live images to an Android device and show it on the screen for the user[1].

Web pages were largely static for quite some time. Updates were infrequent and users had to refresh the page to view new content. In current times, real-time updates likely are some of the most important requirements of the day specially in fields like stock market and healthcare. We now have video conferencing and VoIP services that allow us to communicate online, thanks to real-time data transfer.

HTTP connections To get data from an HTTP connection, the client such as a web browser needs to ask one or more times. It listens on a port, fetches the data and stops listening on that port. HTTP is a protocol that connects to the server only when requested and does not have an undertaking. It also means HTTP is stateless, it contains no memory of the previous request[2].

Advancements in mobile technology have brought a significant change in our lives; we can control the gadgets with smartphones and computers even when we are at a distance. For example, farmers can water their greenhouses by their phones, and with the help of a smart home system, people can check and control home appliances from a remote place.

Unlike conventional security systems that require expensive hardware and maintenance, this system makes use of wireless communication solely. Users can maneuver the robot with the help of their Android devices, which is why additional controlling hardware for the robot is redundant. Using open-source software for the robot parts brings down the cost, whereas Wi-Fi communication makes it easy to use and widely compatible[3].

3
Feasibility

3.1 Technical Feasibility

For this work, ROS will be used for robot control and Flutter for the mobile application. The connection between ROS and Flutter will be done using a library called roslibdart, which enables ROS to communicate with Flutter. Additionally, websockets are going to be used to establish a real-time two-way connection between the mobile device and the robot so data can be exchanged, and control commands can be sent with the least latency.

3.2 Workforce and Time Planning

During this phase, the necessary workforce and time planning for the project were determined. Initially, learning ROS and Flutter is essential, which is expected to take approximately 15-20 days. Subsequently, the technology required to establish communication between Flutter and ROS should be researched and learned, taking another 15-20 days. Following this, the project will progress by first working in a simulation environment and then on a real robot. The entire process is projected to be completed within approximately 1.5 to 2 months. This entire process is illustrated in Figure 3.1 Gantt Chart.

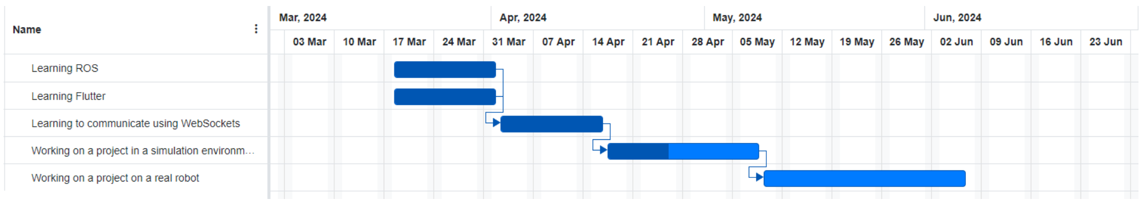


Figure 3.1 Gant Chart

3.3 Legal Feasibility

1. The project will be conducted in compliance with all legal regulations.

2. License terms and conditions related to the use of ROS and Flutter will be taken into account.

3. Legal protection will be ensured considering intellectual property rights of the project and licensing details of the software used.

3.4 Economic Feasibility

In the feasibility stage, a detailed cost analysis was conducted to outline the financial requirements of the project. The table breaks down the costs into three main categories: the expenses for learning ROS and Flutter, which amount to 399.98 TL; the costs for the required programs and libraries, which are assumed to be 0 TL; and the estimated expenditure for the necessary equipment for the robot, which exceeds $600.Thetotalcostfortheprojectismorethan399.98TLand600$, reflecting the investment in both learning resources and hardware. All these expenses are written in Table 3.1 Cost Table.

Table 3.1 Cost Table

Explanation	**Price**
Learning ROS and Flutter	399.98TL
Required Programs and Libraries	0TL
Necessary Equipments for the Robot	More than 600$
Total	***More than 399.98TL and 600$***

4
System Analysis

4.1 Objective

This project aims at creating a strong system for controlling robots with mobile devices using ROS (Robot Operating System) and Flutter, a tool to make mobile apps for interactive experiences. We will be using ROS for communication and control of the robot and Flutter for the development of a mobile app that can control the robot. This will enable remote robot control and gameplay, such as a virtual racing game.

The project will explore how WebSockets can be used for fast and reliable real-time communication between a mobile app and the robot. We hope to demonstrate how those fun and interactive ways for controlling robots in both virtual and real-world environments can be created by combining ROS, Flutter, and WebSocket technologies.

4.2 Information Sources

The development of the Flutter and ROS project will be supported by different sources of information. The main sources will be the official documentation of both communities—Flutter and ROS. The Flutter documentation covers the building of mobile apps, details about widgets, state management, and how to use features specific to a platform. The ROS documentation covers how to control and communicate with robots, and it explains ROS packages, message passing, and node setup.

Additional resources to be used on the project will include tutorials, online forums, and community resources. These resources will provide practical examples, help with troubleshooting, and best practices. By using all these resources, the project will integrate ROS with Flutter apps, allowing for easy communication and control between mobile devices and robots.

4.3 Requirements

4.3.1 Technical Requirements

- Latest versions of ROS and Flutter.
- Adequate knowledge and proficiency in ROS and Flutter usage among project team members.
- Selection of a Mobile Device and a Robot
- Selection of a suitable ROS package for facilitating mobile device control.
- Availability of a compatible robot model for testing the mobile device control system.

4.3.2 Hardware Requirements

- A mobile device suitable for the project
- A robot adequate for the project
- A computer for writing the necessary codes

4.3.3 Training Requirements

- Training on ROS and Flutter, including learning the necessary libraries for communication with ROS on Flutter.
- Learning about Websockets for establishing connections between the robot and the mobile device.
- Understanding the transition from the simulation environment to the robot for seamless integration.

4.3.4 Questions

1. What are the advantages of communicating with ROS through a mobile device?
2. How will the integration of ROS and Flutter be achieved, and what benefits will this integration provide?
3. What types of data transmission methods can be preferred for controlling the robot from a mobile device?

5
System Design

The system design enables seamless communication between robots and mobile devices through the use of ROS and Flutter. WebSocket technology facilitates the free exchange of data and control commands in real time. This allows users to monitor and control robots remotely from the Flutter app, enhancing accessibility and flexibility.

The system also incorporates a module that converts ROS messages into images, enabling users to view what the robot is seeing through the Flutter app. Additionally, the system continuously tracks the robot's current position, creating a dynamic map on the app. This map allows users to visualize the robot's environment and monitor its movements in real time. With established communication, message conversion, and robot connectivity, the system is now ready for additional features and various applications.

5.1 Robot Features

The robot used in this project, Turtlebot3 WafflePi , comes equipped with several features that are essential for our application. These features include:

- LIDAR Sensor: For mapping and navigation.
- Camera: To capture real-time images and video streams.
- Wheel Encoders: For precise movement and position tracking.
- IMU (Inertial Measurement Unit): To measure the robot's orientation and acceleration.
- Single Board Computer (Raspberry Pi): To process sensor data and execute control algorithms.

A photograph of the Turtlebot3 WafflePi can be found in Figure 5.1.

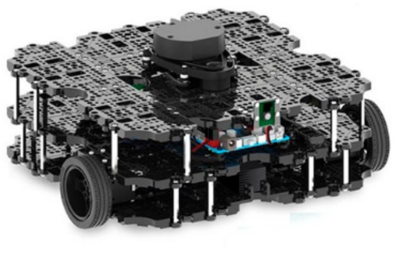


Figure 5.1 The Robot (Turtlebot3 WafflePi) which we connect in simulation

5.2 Communication between ROS and Flutter

The system uses ROS for controlling the robot operations and Flutter for providing the mobile interface. Connection between them is done using WebSockets, enabling rapid real-time data transfer.

- Initialization: The system initiates a connection to the ROS network when the Flutter app starts. It will include subscribing to topics that provide data that this app is interested in.
- Termination: When the connection is no longer required, it will take the necessary steps to shut down the connection to avoid resource leaks.

5.3 Connecting Robot (WebSocket)

The system will initialize a WebSocket connection to the robot's ROS network. This allows the app to send and receive real-time data. The robot's IP address and ROS topics to be subscribed to are defined.

- Initialization: The app connects to the ROS network at a specified URL.
- Topic Subscription: The app subscribes to specific ROS topics to receive data, such as the robot's camera feed.

5.4 Mapping and Navigation

The robot utilizes its LIDAR sensor and wheel encoders to create a map of its environment and navigate within it. This map is then used to plan paths and avoid obstacles.

- Mapping: The robot generates a map of its surroundings using the LIDAR sensor data.
- Navigation: Using the generated map, the robot plans paths and navigates to specified waypoints while avoiding obstacles.

5.5 Message to Image Conversion

The system would have an inbuilt mechanism to convert ROS messages to an image format so that they can be displayed on the Flutter app.

- Image Conversion: It will decode base64 encoded image data received from ROS topics and display them as an image in the Flutter app.

5.6 System Workflow

The overall workflow of the system integrates all the components discussed above to ensure smooth operation and user interaction.

- Data Acquisition: The robot collects data from its sensors, including the camera and LIDAR.
- Data Transmission: The data is transmitted to the Flutter app via WebSocket.
- Data Processing: The Flutter app processes the incoming data, converts necessary messages into images, and updates the user interface.
- User Interaction: Users can monitor the robot's environment and control its movements in real time through the Flutter app.

By following this structured system design, the project ensures robust communication and control mechanisms, providing a flexible and user-friendly interface for robot operation.

6
Implementation

In the Implementation section, the introduction of the application screens encountered before the robot is connected to the application will be provided.

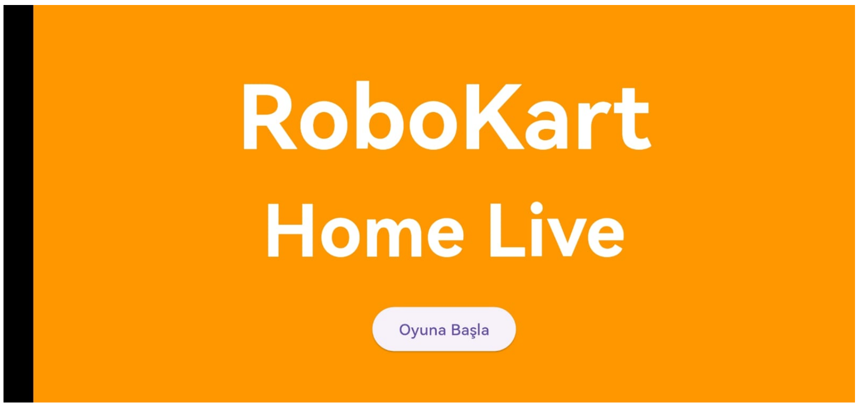


Figure 6.1 Login Screen

In Figure 6.1, we are greeted by a simple orange screen with a single button for entry, requiring no login information.

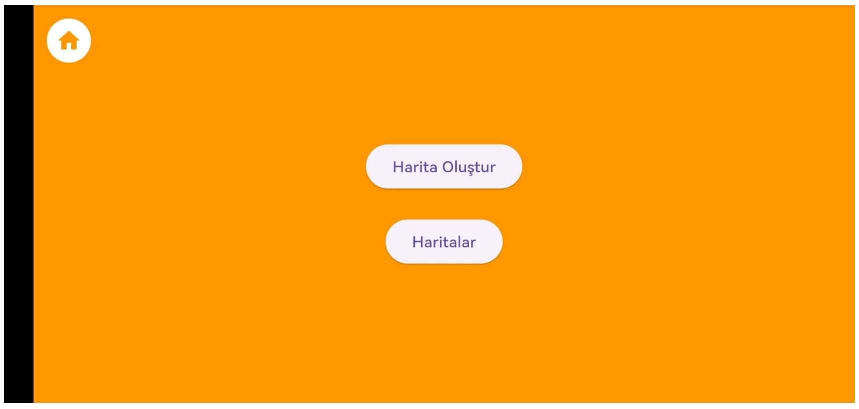


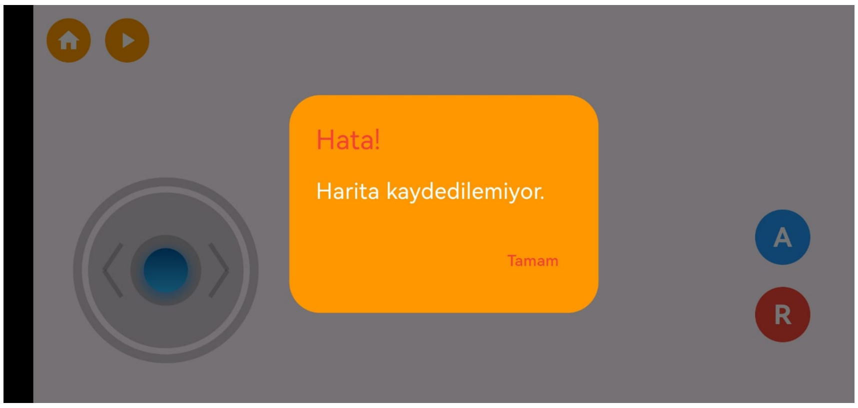
Figure 6.2 Menu Screen

In Figure 6.2, we are presented with a menu screen that has two options:

1. Harita Oluştur: During the map creation phase, you can drive the robot from its

current position to any desired location, and the area you drive through can be saved as a game map.

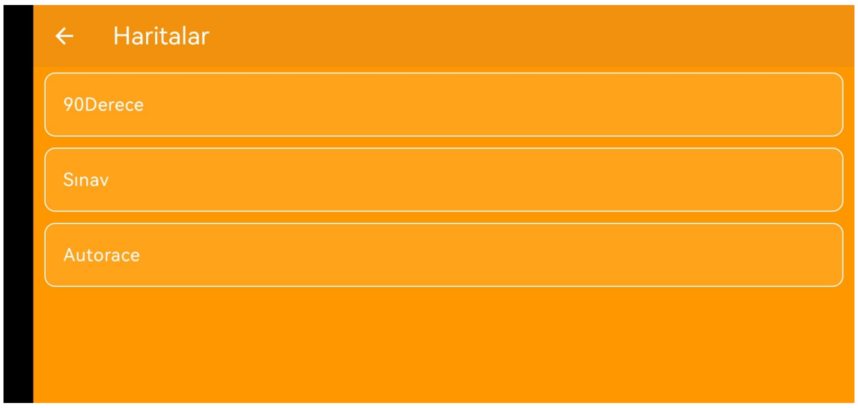
2. Haritalar: You can choose a map from previously created maps and then race on this selected map.


Hata!

Harita kaydedilemiyor.

Figure 6.3 Not-Connected Create Map Screen

In Figure 6.3, the error message encountered when trying to enter the map creation screen without yet being connected to the robot is shown.


Haritalar

90Derece

Sınav

Autorace

Figure 6.4 Map Select Screen

In Figure 6.4, the Map Selection Screen is displayed. There are three maps already saved:

- 90Derece
- Sınav
- Autorace

In Figure 6.5, the speed selection screen that appears after selecting a map is shown, where you must choose the robot's speed. Some considerations when making a speed selection include:

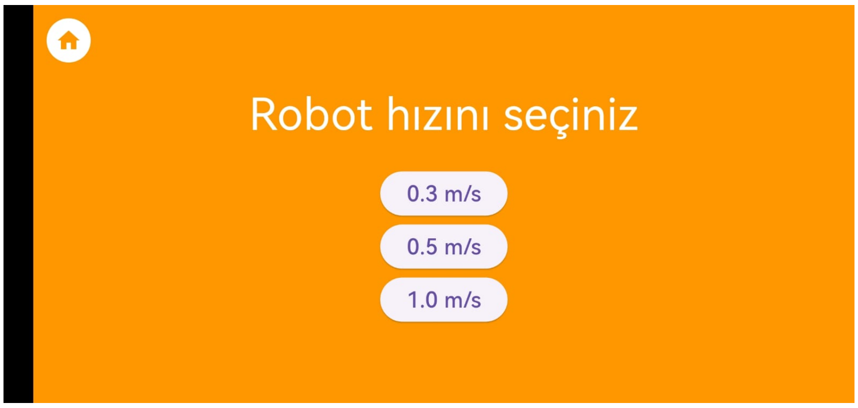
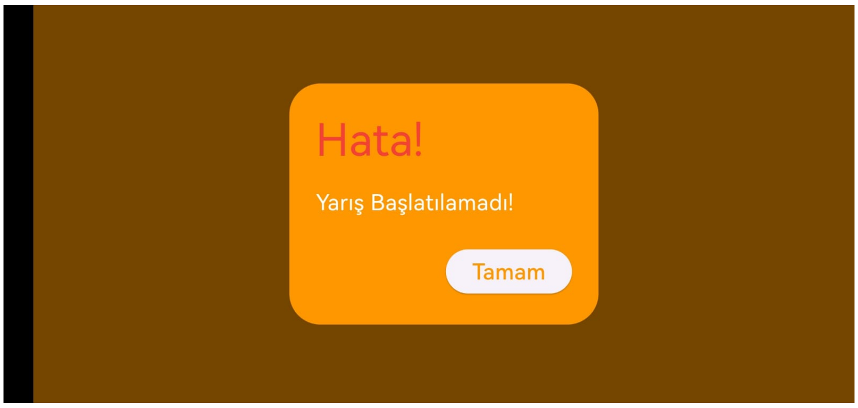


Figure 6.5 Speed Select Screen

- Your driving ability
- How curvy the map is
- The difficulty of controlling the robot

After these evaluations, you can enter the race by selecting a speed from the three options.


Hata!

Yarış Başlatılamadı!

Figure 6.6 Not-Connected Race Screen

In Figure 6.6, the error message encountered when entering the race without yet being connected to a robot is shown.

7 Experimental Results

In the Experimental Results section, the parts that were not covered in the implementation section will be explained in detail. These parts include:

- The Race
- Race End Screen
- Scoreboard

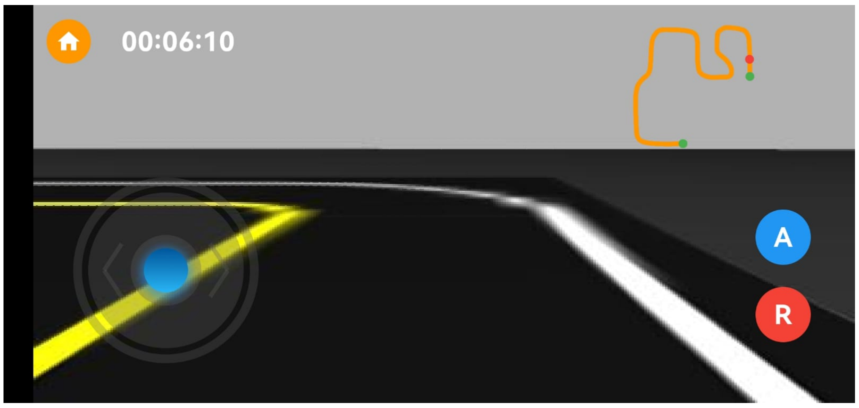


Figure 7.1 Race Screen

In Figure 7.1, you can see the Race screen. As you can observe, the race has started, and the visuals captured by the robot's camera are displayed on the mobile screen. The roads of the selected map (in this case, the selected map is "autorace") are indicated in the top right corner as a minimap. On the minimap, green dots indicate the start and finish points of the race, while the red dot indicates the current position of the robot, allowing you to track where the robot is heading at any given moment. The line formed by the merging of orange dots represents the path. Additionally, a timer written in white will measure how long it takes you to finish the race and will inform you of your time at the end of the race.

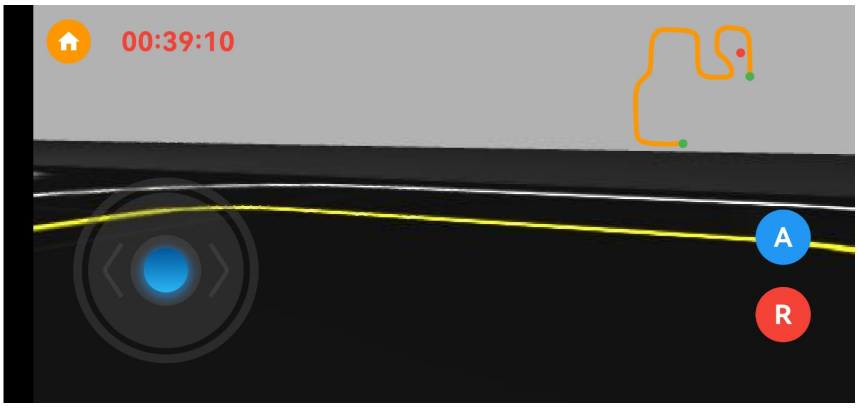


Figure 7.2 Out Of The Road Screen

In Figure 7.2, you again see the Race screen, but this time the robot has deviated from the track, which is indicated to the player by the timer turning red. Additionally, the timer is running at 10 times the normal speed to penalize the player for going off-track and to prompt them to return to the track.

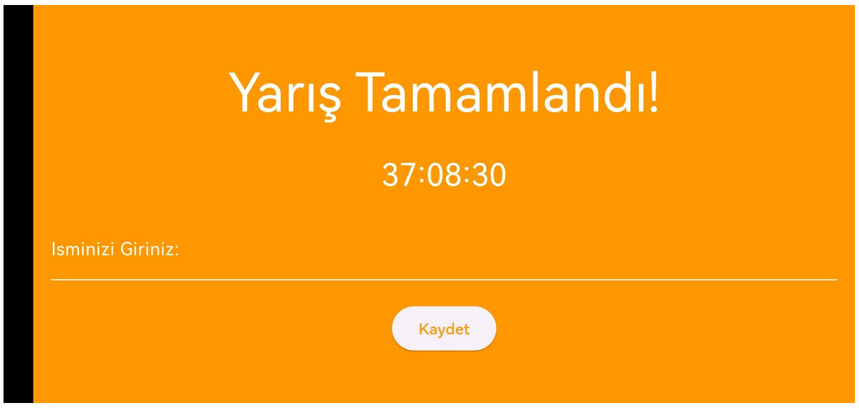
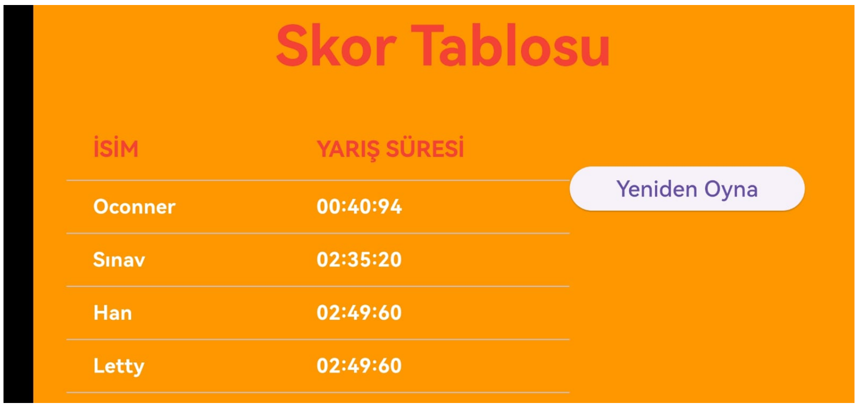


Figure 7.3 End of the Race Screen

In Figure 7.3, you see the race end screen where you can view your time and also see a space to enter your name. In this part, you enter your username to choose how your score should be displayed.

In Figure 7.4, you see the Scoreboard screen where you can see a leaderboard displaying your score against other players who have raced on the same map you just raced on. If you don't see yourself on the list, you can scroll down to find yourself. You can also choose to play again by clicking the "Play Again" button. However, remember that the physical robot should start the race at the starting point.



Skor Tablosu

İSİM	YARIŞ SÜRESİ
Oconner	00:40:94
Sınav	02:35:20
Han	02:49:60
Letty	02:49:60

Figure 7.4 Skorboard Screen

8 Performance Analysis

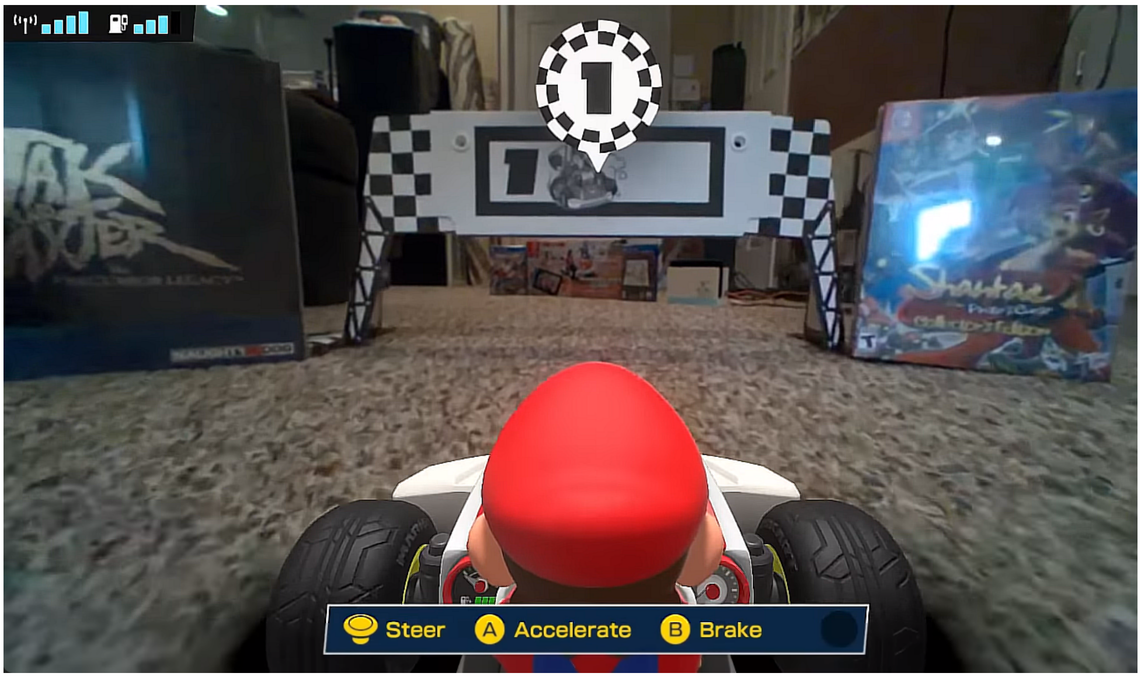


Figure 8.1 Mario Kart Live: Home Circuit Gameplay

A comparative table has been provided for Mario Kart Live: Home Circuit, which is similar to the RoboKart Home Live game, in terms of their features and A gameplay visual of the Mario Kart Live: Home Circuit game is given in Figure 8.1.

Features	**Mario Kart Live: Home Circuit**	**RoboKart Home Live**
Game Mechanics and Controls	Figure 7.3	Figure 8.1
Track Creation	Allowed	Allowed
Race Strategies and Obstacles	against other AR characters, dealing with in-game obstacles	against time
Physical Size and Flexibility of Gameplay	AR cards, Nintendo Switch, enough room to race	Robot, enough room to race

9 Conclusion

The RoboKart Home Live project offers a new gaming experience by using a remotely controlled robot. The racing game map is shown through the camera of the robot.

The mobile app was made in Flutter and Firebase. It uses WebSockets to send the speed commands of the robot. Users can select three different modes depending on the size of their environment.

For racing, the user could first drive the robot over the map to set it up. When the race begins, the user can view his rank in the leaderboard along with other users on the same map.

References

[1] A. E. Kırlı, M. B. Dilaver, and F. Çakmak, "Mobile robot control with android device sensors by using ros," *Mugla Journal of Science and Technology*, vol. 3, no. 1, pp. 31–34, 2017.

[2] B. Gupta and M. Vani, "An overview of web sockets: The future of real-time communication," *Int. Res. J. Eng. Technol. IRJET*, vol. 5, no. 12, 2018.

[3] G. Ersahin and H. Sedef, "Wireless mobile robot control with tablet computer," *Procedia-Social and Behavioral Sciences*, vol. 195, pp. 2874–2882, 2015.

Curriculum Vitae

FIRST MEMBER

Name-Surname: Muhammed KASAP
Birthdate and Place of Birth: 12.07.2002, Ordu
E-mail: muhammed.kasap@std.yildiz.edu.tr
Phone: 0507 155 00 96
Practical Training:

Project System Informations

System and Software: Ubuntu İşletim Sistemi, Mobil Telefon
Required RAM: 4GB
Required Disk: 2GB