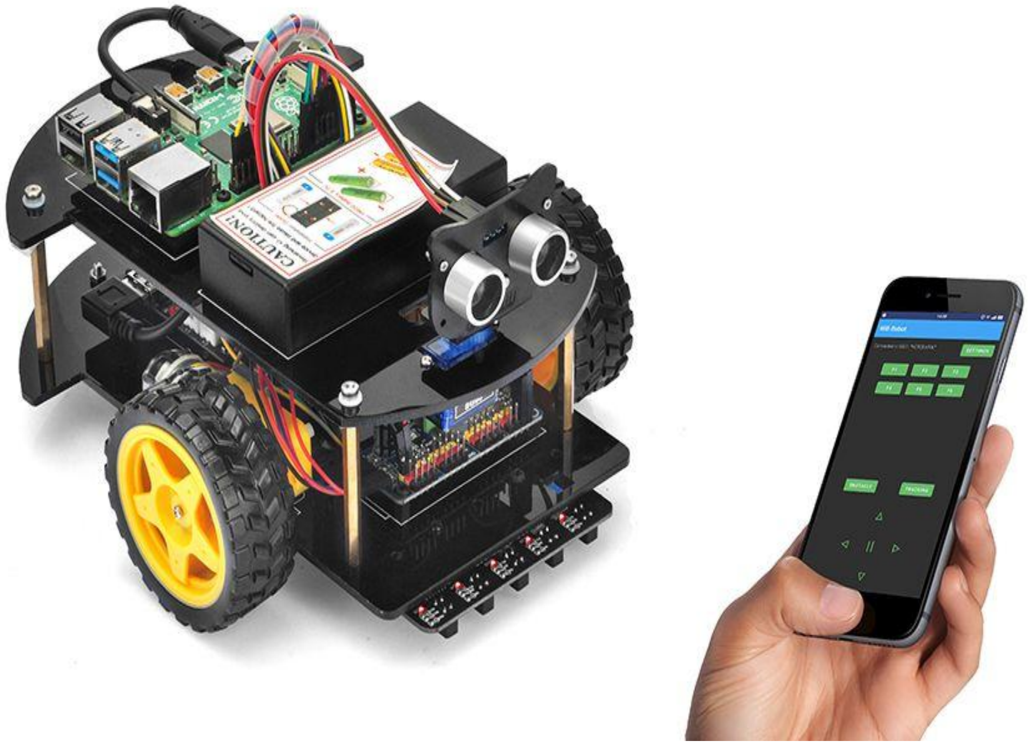
Final Project
Osoyoo Raspberry Pi Car V2



Palomar College - CSCI 212 MACHINE ORG / ASSEMBLY LANGUAGE

Group #11: Anh Tran & Kristin Stacy

INTRODUCTION

Our group's goal is to implement three tasks for OSOYOO Pie CAR V2.0 in C code.

Task 1: Add additional functionality to control the Leader robot acceleration/deceleration.

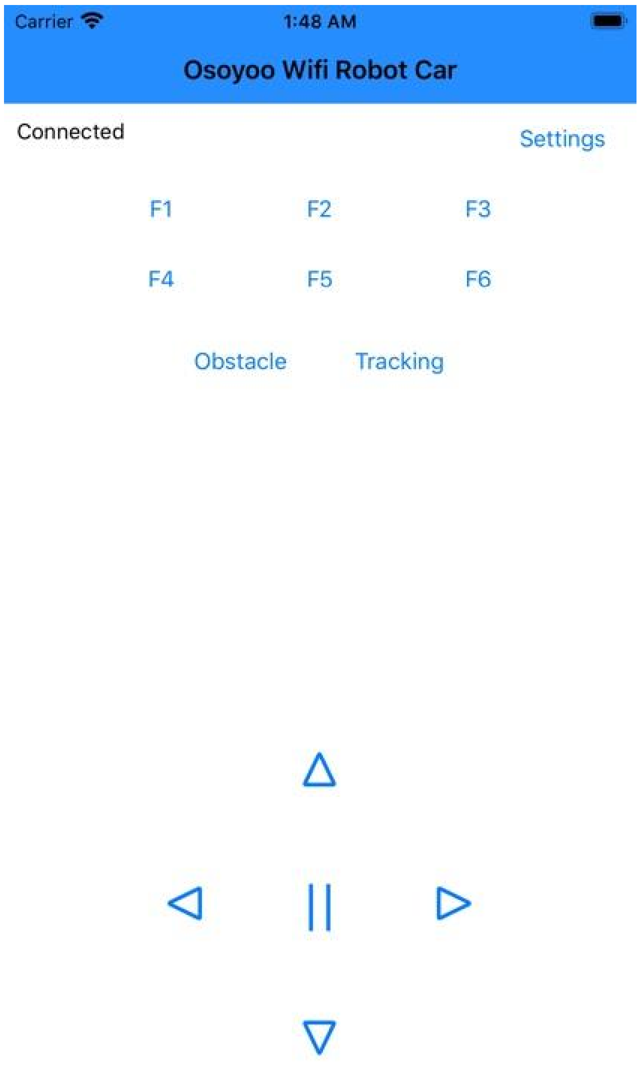
Task 2: Add the Obstacle avoidance functionality controlled via the "Obstacle" button on the phone app. Pressing the "Obstacle" button will toggle the robot obstacle avoidance capability on and off. When it is off, the Leader robot will be manually controlled via the phone app. When it is on, the robot should autonomously try to avoid any obstacle, return to the trajectory it was on prior to adjusting for the obstacle, and continue straight at a constant speed until the user toggle the Obstacle mode back to manual.

Task 3: The Follower robot will attempt to follow the Leader robot autonomously, while maintaining a constant distance apart. As part of this task, the Follower robot will need to use the ultrasonic sensor onboard to find and to estimate the distance from the Leader robot. The Follower robot will adjust its speed accordingly to speed up/slow down in order to maintain the constant distance.

(ICD) INTERFACE CONTROL DOCUMENT

- OSOYOO V2.0 Raspberry Pi Robot Car basic framework
- Ultrasonic sensor to detect
- Tracking sensor module
- OSOYOO Wifi UDP Robot Car control APP

After connecting the OSOYOO Wifi UDP Robot Car control APP with Raspberry Pi through IP Address, the user can control the Robot Car through pressing buttons on the APP, which will send a UDP message to implement commands on Robot Car.



Button	**UDP message**
F1	F
F2	G
F3	H
F4	I
F5	J
F6	K
▲	A
▼	B
►	R
◄	L
I I	E
obstacle	O
tracking	T

Purpose of each interface:

- F1: Accelerate
- F2: Decelerate
- F3: Automatic Follow Leader mode
- F4 - F6: N/A
- ▲: Move forward
- ▼: Move backward
- ►: Turn right
- ◄: Turn left
- ||: Stop
- Obstacle: Automatic Obstacle Avoidance mode
- Tracking: Automatic Line Track mode

Description:

1. Accelerate

- Robot will speed up compare to its current movement’s velocity

2. Decelerate

- Robot will slow down compare to its current movement’s velocity

3. Automatic Follow Leader mode

- Robot car will use Ultrasonic sensor to detect the Leader robot and make automatic driving and follow the Leader robot.

4. Move forward
- Robot will move forward continuously with normal speed until prompt stop or change mode by user.
5. Move backward
- Robot will move backward continuously with normal speed until prompt stop or change mode by user.
6. Turn right
- Robot will turn right continuously with normal speed until prompt stop or change mode by user.
7. Turn left
- Robot will turn left continuously with normal speed until prompt stop or change mode by user.
8. Stop
- Simply stop car from any movements

9. Automatic Obstacle Avoidance mode
- Robot car will use Ultrasonic sensor to detect obstacles and make automatic driving and avoid collisions.

10. Automatic Line Track mode
- Robot car to automatically drive along a black line in white ground by using five IR tracking sensors to detect the line.

- The first function, **`go_Advance(int, int)`**, is used to make a robot moving forward has two inputs of the desired speed to travel.
- The **`go_Back(int, int)`** function is similar to the **`go_Advance`** function, except the robot moves backward.
- The **`go_left`** and **`go_Right`** function each have inputs of speed desired on each wheel to turn at a different angle and speed. When the function is called, they make one wheel go faster than the other wheels to make the robot rotate and change the direction.

(ADD) ALGORITHM DESCRIPTION DOCUMENT

The final code executed by the robot was modular. The code included functions for the purpose of navigating forward, turning, backward, avoiding obstacles, and manually controlled by the user. These functions were able to be repurposed at any time that the robot needed to accomplish the tasks.

- For the first task, we modify **`void udp_handler(void)`** method with case 'F' (button F1) corresponds with acceleration and case 'G' (button F2) corresponds with deceleration.
- We create a **`char prev_status`** for previous status to keep track of user's input. Inside each case, we create a nested switch statement to check the previous status **`(go_Back, go_Advance, go_Left or go_Right)`** of the robot in order to determine its current status in each state with different velocity.
- For the second task, by adding **`boolean obst_status`** to keep track of obstacle avoidance capability on/off. Declaring boolean **`obst_status`** set to true indicates manual controlled mode is on and automatic obstacle avoidance mode is off.

- In **`void udp_receiver(void)`** method, create a conditional statement to check if the user clicks on the "Obstacle" button. Case **`buf[0] == 'O'`** and obst_status is true, set **`obst_status`** to **`false`** and vice versa. Update switch statement in **`void udp_handler(void)`** method case **`'O'`** with a if statement to check **`obst_status`**. If **`obst_status`** is **`false`**, turn on automatic obstacle avoidance mode. If **`obst_status`** is **`true`**, turn off automatic obstacle avoidance by **`stop_car(fd)`** and prompt the user that the robot has returned to manual mode.
- Final task, create a void **`follow_leader()`** function and perform the opposite movement from **`obstacle_avoid()`** function. Instead of avoiding the obstacle, try to approach and remain a certain distance away from the object. Firstly, set the appropriate range between the Leader and the Follower by using **`distance()`** function returning a value greater than **`int OBJ_DISTANCE`**, if so return signal **`sts#`** to 1. **`o_val`** combines **`sts1, sts2,`** and **`sts3`** signals into a **`String`**. Check **`o_val`** data output if the robot detects an object. For example, case **`o_val`** equals **`"100"`** indicates the robot detects an object appearing in the far left.

Navigate the robot toward the object by implementing the **`go_Left`** function. In case **`"000"`**, which means an object has approached within allowed range, the Follower robot will automatically stop until further change in movement.

Performance Results

This video demonstrates the robot and the added functionality of acceleration and deceleration functions.

https://youtu.be/nEFqnYDcaGA

This video demonstrates the robot and the added functionality of Obstacle detection mode.

https://youtu.be/o_IpmystSDU

This video demonstrates follow the leader mode. Original task was to show one robot following the leader robot. We ran into technical problems when taking the robots to a new location and had wifi connectivity issues so instead we demonstrated a robot following an object rather than another robot.

https://youtu.be/sSC4SKzsC5k