**CE 6308 / CS 6396 / EEDG 6308**

**Real-Time Systems**

**Assignment 3**

**Spring 2023**

**Instructor: Farokh Bastani**

**Deep Padmani (DMP210005)**

**Rubina Parveen (RXL220014)**

**Roshni Johnson Nambiaparambil (RXN200022)**

**Venkat Kishan Bommali (VXB200009)**

**Sai Vishwanth Yalamanchili (SXY210012)**

**Objective**

The objective of this project is to move the Ego vehicle through the arena given and make the Ego vehicle park in the parking lot.

**Requirements**

* Problem 1 – We were asked to search for the empty parking spot in the parking lot.
* Problem 2 – We had to park the Ego vehicle safely in the parking lot.

**Designated Sensors and Actuators**

* **Sensors:** DeviceRegsitry.pixels, DeviceRegistry.compass, DeviceRegistry.speedometer, DeviceRegistry.lidar, DeviceRegistry.gps
* **Actuators:** DeviceRegistry.speedControl, DeviceRegistry.brakeControl, DeviceRegistry.steeringControl.

**Implementation**

For the implementation of this project, we created the 5 tasks given below using the sensors, actuators and state of the Ego Vehicle.

**Task List**

protected TaskInterface[] tasks = new TaskInterface[]   
    {  
        new InitialPosTask(),

new ObstacleDetect(),  
        new RightWallFollow(),  
        new WallFinder(),  
        new SearchParkingSlot(),  
        new ParkingTask(),

};

**Sensors Used**

devices.pixels

devices.compass

devices.lidar

devices.gps

The **device** is an instance of **DeviceRegistry**.

**Actuators Used**

devices.speedControl

devices.steeringControl

devices. brakeControl

To store the current state of the vehicle, we are using Ego Vehicle’s memory and we created the states as listed below.

STATE\_INITIAL\_ORIENTATION = 0;

STATE\_TAKING\_INITIAL\_POSITION = 1;

STATE\_INITIAL\_STATE\_REACHED = 2;

STATE\_INITIAL\_DIRECTION\_CALIBRATION = 3;

STATE\_MOVING\_FORWARD\_WITH\_3F\_SPEED = 4;

STATE\_OBJECT\_DETECTION = 5;

STATE\_OBJECT\_DETECTED = 6;

STATE\_WALL\_DETECTED = 7;

STATE\_CAR\_DETECTED = 8;

STATE\_MOVING\_FORWARD\_WITH\_1F\_SPEED = 9;

STATE\_LANE\_DETECTION = 10;

STATE\_TURNING\_RIGHT = 11;

STATE\_TURNING\_LEFT = 12;

STATE\_U\_TURN\_1 = 13;

STATE\_ U\_TURN\_2 = 14;

STATE\_PARKING\_DETECTED = 15;

STATE\_READY\_FOR\_PARKING = 16;

STATE\_PARKING\_EGO\_VEHICLE = 17;

STATE\_PARKING\_DONE = 18;

STATE\_PEDS\_DETECTED = 19;

devices.memory

**Task 1: InitialPosTask()**

Using this task, the vehicle will take its initial position by aligning itself away from the wall and near to the lane one. The vehicle will move left or right depending on its position and according to value of **compass**.

Using the states STATE\_INITIAL\_ORIENTATION = 0 and STATE\_TAKING\_INITIAL\_POSITION = 1 it will orient itself towards south and move towards lane one. Using the STATE\_INITIAL\_STATE\_REACHED = 2 and STATE\_INITIAL\_DIRECTION\_CALIBRATION = 3 it will start facing the east direction.

**Dependencies:**DeviceRegistry.compass, DeviceRegistry.gps, Device.memory, DeviceRegistry.steeringControl

A screenshot of a video game

Description automatically generated with medium confidence

*The vehicle has taken the initial position to move forward in the lane 1.*

**Task 2: ObstacleDetect()** Using this task, the vehicle can detect obstacles like pedestrians and cars, and at that time, it will wait until that object cannot clear the vehicle route. The vehicle will perform this task using lidar and pixel sensors, as shown below.

A screenshot of a video game

Description automatically generated

*Object Detection by ego vehicle.*

// return true if car in front of vehicle

//(0, 124, 255) (0, 255, 27) (172, 0, 255) (192, 192, 192) (101, 33, 0)

if((devices.pixels[5,4,0] == 0 && devices.pixels[5,4,1] == 124 && devices.pixels[5,4,2] == 255) ||

(devices.pixels[5,4,0] == 0 && devices.pixels[5,4,1] == 255 && devices.pixels[5,4,2] == 27) ||

(devices.pixels[5,4,0] == 172 && devices.pixels[5,4,1] == 0 && devices.pixels[5,4,2] == 255) ||

(devices.pixels[5,4,0] == 192 && devices.pixels[5,4,1] == 192 && devices.pixels[5,4,2] == 192) ||

(devices.pixels[5,4,0] == 101 && devices.pixels[5,4,1] == 33 && devices.pixels[5,4,2] == 0)||

(devices.pixels[4,4,0] == 0 && devices.pixels[4,4,1] == 124 && devices.pixels[4,4,2] == 255) ||

(devices.pixels[4,4,0] == 0 && devices.pixels[4,4,1] == 255 && devices.pixels[4,4,2] == 27) ||

(devices.pixels[4,4,0] == 172 && devices.pixels[4,4,1] == 0 && devices.pixels[4,4,2] == 255) ||

(devices.pixels[4,4,0] == 192 && devices.pixels[4,4,1] == 192 && devices.pixels[4,4,2] == 192) ||

(devices.pixels[4,4,0] == 101 && devices.pixels[4,4,1] == 33 && devices.pixels[4,4,2] == 0)){

return true;

}

return false;

}

public bool checkPedestrains(DeviceRegistry devices)

{

// return true if pedestrain in front of vehicle

if((devices.pixels[5,4,1] == 0 && devices.pixels[5,4,0] == 255 &&devices.pixels[5,4,2] == 27) ||

(devices.pixels[4,4,1] == 0 && devices.pixels[4,4,0] == 255 && devices.pixels[4,4,2] == 27)){

return true;

}

return false;

}

**Task 3: RightWallFollow()**

We have used the right-hand rule or the wall follower strategy therefore this task moves the ego vehicle in a straight path in the lanes. The states of the Ego vehicle are as follows - STATE\_MOVING\_FORWARD\_WITH\_3F\_SPEED = 4; For moving in the lanes with speed 3f

STATE\_OBJECT\_DETECTION = 5; For detects if there is an object in the path

STATE\_MOVING\_FORWARD\_WITH\_1F\_SPEED = 9; For moving towards another lane with 1f speed

STATE\_LANE\_DETECTION = 10; Detects the other lanes

STATE\_U\_TURN\_1 = 13; Used while moving from lane 2 to 3 and 4 to 5

**Dependencies:**

*DeviceRegistry.lidar, DeviceRegistry.gps, Device.memory, DeviceRegistry.pixels*

A screenshot of a video game

Description automatically generated

*Ego Vehicle is moving in the forward direction using the right-hand rule.*

**Task 4: WallFinder()**

As the name suggests, the Ego vehicle is trying to find the outer walls. According to the GPS values, it will align itself away from the wall. Following states are used for this task:

STATE\_OBJECT\_DETECTED = 6; Indicates that something is in the path of the ego vehicle.

STATE\_WALL\_DETECTED = 7; Tells the Ego vehicle that the object is a wall

STATE\_TURNING\_RIGHT = 11; Taking a right turn away from the wall

STATE\_TURNING\_LEFT = 12; Taking a left turn away from the wall

STATE\_ U\_TURN\_2 = 14; Using this it will take a left turn after detecting the lane

**Dependencies:**   
*DeviceRegistry.compass, DeviceRegistry.gps, Device.memory, DeviceRegistry.steeringControl.*

A screenshot of a video game

Description automatically generated

*The Ego vehicle is taking a U turn to travel from lane 2 to lane 3*

**Task 5: SearchParkingSlot()**

Using this task, the Ego vehicle is searching for the empty parking slot using lidar sensors 3 and 4. If it detects the empty space, it stops there. The states used here are:

STATE\_OBJECT\_DETECTION = 5; Used for detecting the object

STATE\_PARKING\_DETECTED = 15; Indicates that an empty spot is detected

**Dependencies:**

*DeviceRegistry.steeringControl, DeviceRegistry.lidar, DeviceRegistry.memory,*

*DeviceRegistry.gps*

A screenshot of a video game

Description automatically generated

*The Ego vehicle has detected empty parking slot and is now starting to orient itself towards the parking slot.*

**Task 6: ParkingTask()**

We are using this task to park the Ego Vehicle into the empty parking slot. The states we used here are:

STATE\_READY\_FOR\_PARKING = 16; Using this state the ego vehicle will orient itself towards the parking slot.

STATE\_PARKING\_EGO\_VEHICLE = 17; The state helps the Ego vehicle to move forward while detecting the car in front of it using the lidar.

STATE\_PARKING\_DONE = 18; This is the state of the Ego vehicle when it is successfully parked.

**Dependencies:**   
*DeviceRegistry.compass, DeviceRegistry.steeringControl, DeviceRegistry.memory, DeviceRegistry.lidar, DeviceRegistry.gps*

A screenshot of a video game

Description automatically generated

*The Ego vehicle is parked.*

Task Graph for 2 cycles of simulation

**Cycle 1**

**A picture containing text, screenshot, software, line

Description automatically generated**

**Task 1:** **InitialPosTask ( )**

**Task 2:** **ObstacleDetect ( )**

**Task 3: RightWallFollow( )**

**Task 4: WallFinder( )**

**Task 5: SearchParking( )**

**Task 6: ParkingTask( )**

**Cycle 2 is same as Cycle 1**