CE 6308 / CS 6396 / EEDG 6308

Real-Time Systems

Assignment 1

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 it cross the bridge safely.

Requirements

- Problem 1 – We were asked to modify the Pre-Debug task to accelerate the Ego Vehicle when 'W' key is pressed.
- Problem 2 – To automate the movement of Ego Vehicle and to make it cross the bridge safely

Designated Sensors and Actuators

- **Sensors:** DeviceRegsitry.pixels, DeviceRegistry.compass, DeviceRegistry.speedometer, DeviceRegistry.microphone
- **Actuators:** DeviceRegistry.speedControl, DeviceRegistry.brakeControl, DeviceRegistry.steeringControl, DeviceRegistry.transmitterControl

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 Debug_Task(),
        new InitialDirectionControl(),
        new ReachDestinationControlTask(),
        new TurnLeft(),
        new TurnRight(),
        new AlarmControl()
};
```

Sensors Used

```
devices.pixels
devices.compass
devices.microphone
```

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

Actuators Used

```
devices.speedControl
devices.steeringControl
devices.transmitterControl
```

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

```
devices.memory
```

```
STATE_INIT_DIRECTION_CONTROL = 0;
STATE_VEHICAL_DIRECTION_DONE = 1;
STATE_READY_FOR_STRAIGHT_MOVE = 2;
STATE_CHECK_FOR_GRASS = 3;
STATE_GRASS_DETACTED = 4;
STATE_READY_FOR_LEFT_MOVE = 5;
STATE_LEFT_MOVE_DONE = 6;
STATE_READY_TO_MOVE_TOWARDS_BRIDGE = 7;
STATE_GRASS_DETACTED_NEAR_BRIDGE = 8;
STATE_READY_FOR_RIGHT_MOVE = 9;
STATE_RIGHT_MOVE_DONE = 10;
STATE_BOAT_ALARM_DETACTED = 11;
STATE_TX_CONTROL_AND_READY_MOVE_TOWARDS_DEST = 12;
STATE_MOVE_TOWARDS_DEST = 13;
STATE_DESTINATION_STOP = 14;
```

Task 0: `Debug_Task()`

This task is made to move the Ego Vehicle manually and getting used to the AVL environment. We are moving the vehicle forward using the 'W' key.

```
public class Debug_Task : TaskInterface
  {
    public void Execute(DeviceRegistry devices) {
        if (Input.GetKey("w"))
        {
           /** Problem 1: AVL Environment **/

            devices.speedControl[0] = 1f;
            devices.speedControl[1] = 10f;

        }
```

Task 1: `InitialDirectionControl()`

Using this task, the vehicle will take its initial position by aligning itself towards the north direction.

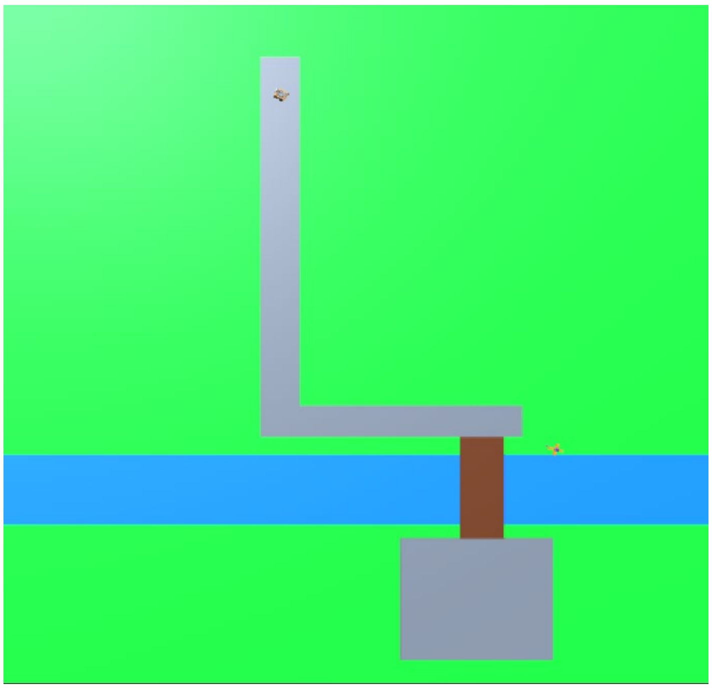
Here, we are allowing a direction correction window of 1 degree (-0.5 to +0.5) for the vehicle to align itself towards the north direction. It will take a left or right turn according to value of **`compass`**. The state of vehicle is `STATE_INIT_DIRECTION_CONTROL`.

```
if (devices.compass[0] > Constants.COMPASS_POS_OFFSET_NEAR_NORTH)
{
    // Left
    devices.steeringControl[0] = Constants.STEERING_CONTROL_IDX0;
    devices.steeringControl[1] = Constants.COMPASS_NEG_OFFSET_NEAR_NORTH;
}
else if (devices.compass[0] < Constants.COMPASS_NEG_OFFSET_NEAR_NORTH)
{
    // Right
    devices.steeringControl[0] = Constants.STEERING_CONTROL_IDX0;
    devices.steeringControl[1] = Constants.COMPASS_POS_OFFSET_NEAR_NORTH;
}
```

```
else
{
    devices.steeringControl[0] = Constants.STEERING_CONTROL_NORMAL;
}

if((devices.compass[0] > Constants.COMPASS_NEG_OFFSET_NEAR_NORTH) &&
   (devices.compass[0] < Constants.COMPASS_POS_OFFSET_NEAR_NORTH))
{
     devices.memory[0] = Constants.STATE_VEHICAL_DIRECTION_DONE;
}
```

<u>**Dependencies**</u>**:**
`DeviceRegistry.compass`,
`Device.memory`



As soon as the vehicle aligns to North direction it's state changes to `STATE_VEHICAL_DIRECTION_DONE.`

The next state is `STATE_READY_FOR_STRAIGHT_MOVE` where the vehicle is now ready to move forward on the straight path.

```
if((devices.compass[0] > Constants.COMPASS_NEG_OFFSET_NEAR_NORTH) &&

   (devices.compass[0] < Constants.COMPASS_POS_OFFSET_NEAR_NORTH) &&
   (Constants.STATE_VEHICAL_DIRECTION_DONE == devices.memory[0]))
{
        devices.steeringControl[0] = Constants.STEERING_CONTROL_IDX0;
        devices.steeringControl[1] = Constants.STEERING_CONTROL_NORMAL;
        devices.memory[0] = Constants.STATE_READY_FOR_STRAIGHT_MOVE;
}
```

Task 2: `ReachDestinationControlTask()`

Responsibilities of this task is to direct the Ego Vehicle towards the goal.

After setting the direction, the vehicle will accelerate towards the goal with the speed of `SPEED_2F`.

```
if (Constants.CURRENT_STATE == devices.memory[0])
{
    devices.speedControl[0] = Constants.STEERING_CONTROL_IDX0;
    devices.speedControl[1] = Constants.SPEED_2F;
    devices.memory[0] = Constants.NEXT_STATE;
}
```

`ReachDestinationControlTask` will execute every time a state is changed. For example, when we change from state `STATE_READY_FOR_STRAIGHT_MOVE` to `STATE_READY_FOR_LEFT_MOVE` the control goes back to **`ReachDestinationControlTask.`**

Dependencies:
`DeviceRegistry.speedControl`,
`DeviceRegistry.memory`
`DeviceRegistry.transmitterControl`

Task 3: `TurnLeft()`

Using this task, the vehicle will take a 90 Degree Left turn, with a direction correction window of 2 degree (+89 to +91). Initially its status is `STATE_READY_FOR_LEFT_MOVE` and after the left turn is completed, the state changes to `STATE_LEFT_MOVE_DONE`. When the state is `STATE_LEFT_MOVE_DONE`, the control goes back to **`ReachDestinationControlTask`** and the vehicle starts moving forward.

Dependencies:
`DeviceRegistry.compass,`
`DeviceRegistry.steeringControl,`
`DeviceRegistry.memory`

```
if (Constants.STATE_GRASS_DETACTED == devices.memory[0])
{
    devices.steeringControl[0] = Constants.STEERING_CONTROL_IDX0;
    devices.steeringControl[1] = Constants.STEERING_CONTROL_FAST_LEFT;

    if((devices.compass[0] > Constants.COMPASS_NEG_91_DEGREE) &&
      (devices.compass[0] < Constants.COMPASS_NEG_89_DEGREE))
    {
        devices.memory[0] = Constants.STATE_READY_FOR_LEFT_MOVE;
    }
}
```

Task 4: `TurnRight()`

The vehicle will take 90 Degree Right turn, allowing a direction correction of 2 degree (-1 to +1). Here, the state of ego vehicle changes from `STATE_READY_FOR_RIGHT_MOVE` to `STATE_RIGHT_MOVE_DONE`, while sending the control back to **`ReachDestinationControlTask`** for transmission of alarm signal by the ego vehicle for crossing the bridge and for moving forward when the boat arrival alarm is not detected.

```
if (Constants.STATE_GRASS_DETACTED_NEAR_BRIDGE == devices.memory[0])
{

    devices.steeringControl[0] = Constants.STEERING_CONTROL_IDX0;
    devices.steeringControl[1] = Constants.STEERING_CONTROL_SLOW_RIGHT;

    if(devices.compass[0] > Constants.COMPASS_NEG_1_DEGREE &&
      devices.compass[0] < Constants.COMPASS_POS_1_DEGREE)
    {
        devices.memory[0] = Constants.STATE_READY_FOR_RIGHT_MOVE;
    }
}
```

Dependencies:
`DeviceRegistry.compass,`
`DeviceRegistry.steeringControl,`
`DeviceRegistry.memory`

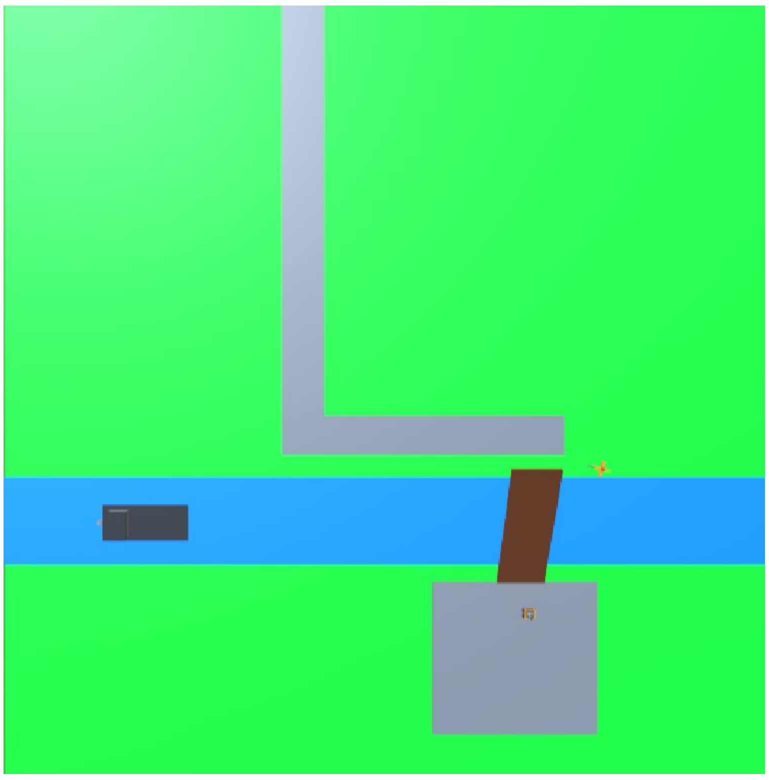
Task 5: `AlarmControl()`

The vehicle will listen to the boat arrival alarm using `devices.microphone` . If the alarm is **ON**, then the ego vehicle will wait and if it is **OFF**, the ego vehicle will cross the bridge by transmitting an alarm. The state of the ego vehicle will initially be `STATE_BOAT_ALARM_DETACTED` if the boat arrival alarm is ON. As soon as the boat alarm goes OFF, it will change to `STATE_TX_CONTROL_AND_READY_MOVE_TOWARDS_DEST`.

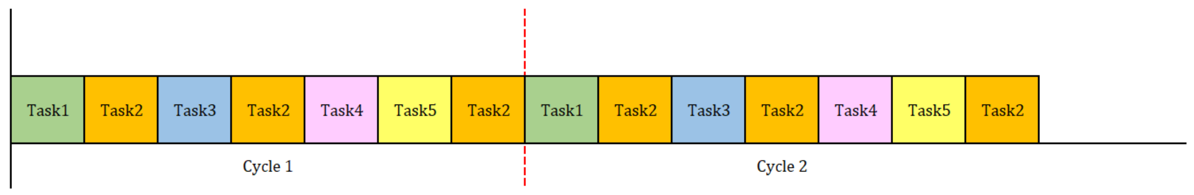
```
if (Constants.STATE_RIGHT_MOVE_DONE == devices.memory[0])
{
    if (Constants.BOAT_ALARM_DETACTED == devices.microphone[0])
    {
        devices.memory[0] = Constants.STATE_BOAT_ALARM_DETACTED;
    }
}

if (Constants.STATE_BOAT_ALARM_DETACTED == devices.memory[0])
{
    if (Constants.NO_ALARM_DETACTED == devices.microphone[0])
    {
devices.memory[0] = Constants.STATE_TX_CONTROL_AND_READY_MOVE_TOWARDS_DEST;
    }
}
```

Dependencies:
`DeviceRegistry.compass,`
`DeviceRegistry.steeringControl,`
`DeviceRegistry.memory`



Task Graph for 2 Cycles of Simulation



Task 1	InitialDirectionControl( )
Task 2	ReachDestinationControlTask( )
Task 3	TurnLeft( )
Task 4	TurnRight( )
Task 5	AlarmControl( )

Cycle 1

Task 1: InitialDirectionControl ( )

The system initially sets the position of EGO vehicle to North by checking the surroundings.

Task 2: ReachDestinationControlTask( )

EGO vehicle moves forward

Task 3: TurnLeft( )

EGO vehicle resets the direction to stay on the correct path

Task 2: ReachDestinationControlTask( )

EGO vehicle moves forward

Task 4: TurnRight( )

The EGO vehicle move forward till it reaches a dead end or a left turn

Task 5: AlarmControl( )

The EGO vehicle shifts the position to the left turn

Task 2: ReachDestinationControlTask( )

EGO vehicle moves forward

Cycle 2 is same as Cycle 1