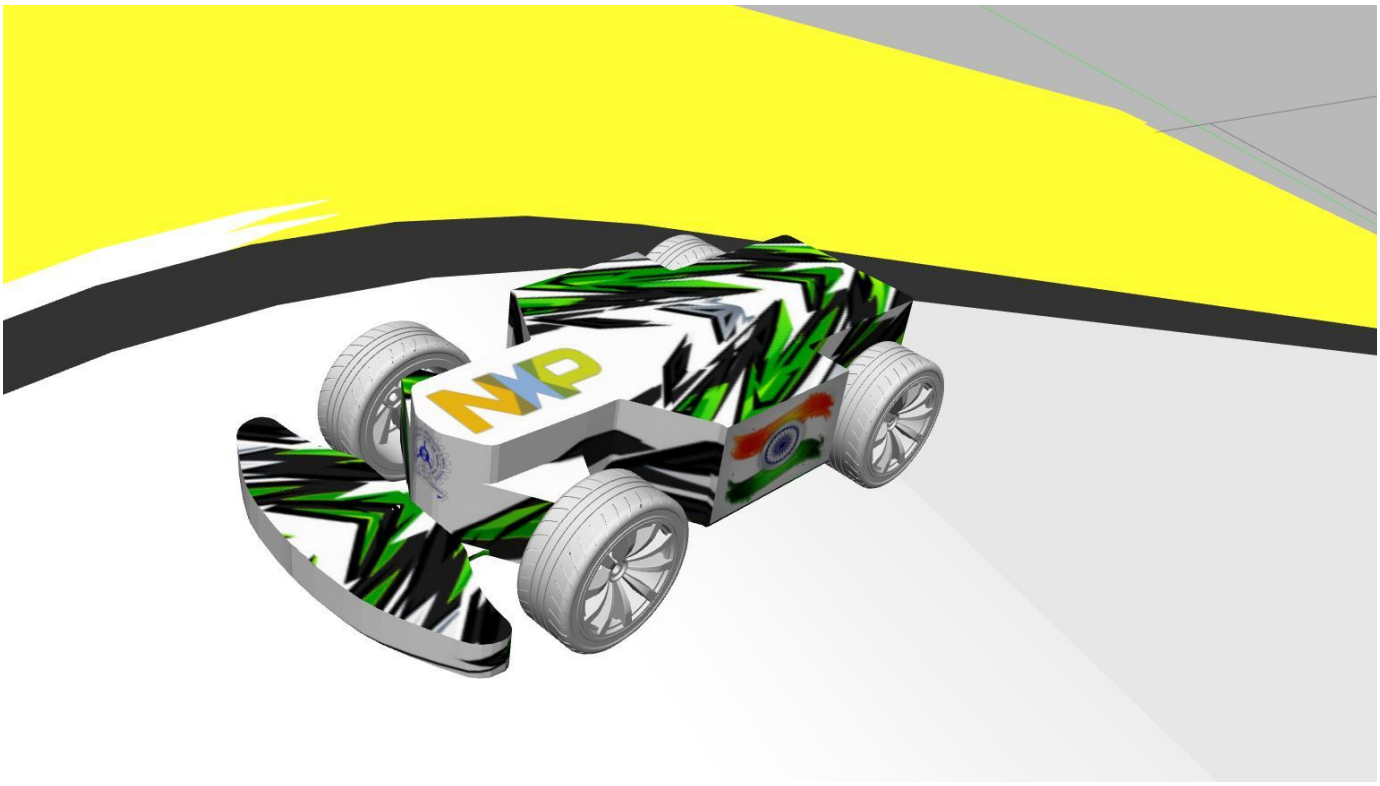
NXP AIM : Online Design Challenge

Team Name : Kuch_Bhi

S. No.	Team Members Name	College/Location	Branch	Current Semester
1.	Tanishq Chaudhary	Indian Institute of Technology Dhanbad	Electronics and Communication Engineering	5
2.	Sakshi Dwivedi	Indian Institute of Technology Dhanbad	Mining Engineering	5
3.	Harshit Sharma	Indian Institute of Technology Dhanbad	Computer Science and Engineering	5
4.	Rudra Pratap Singh Panwar	Indian Institute of Technology Dhanbad	Mechanical Engineering	5

NXP AIM Car Model Pictures

1.

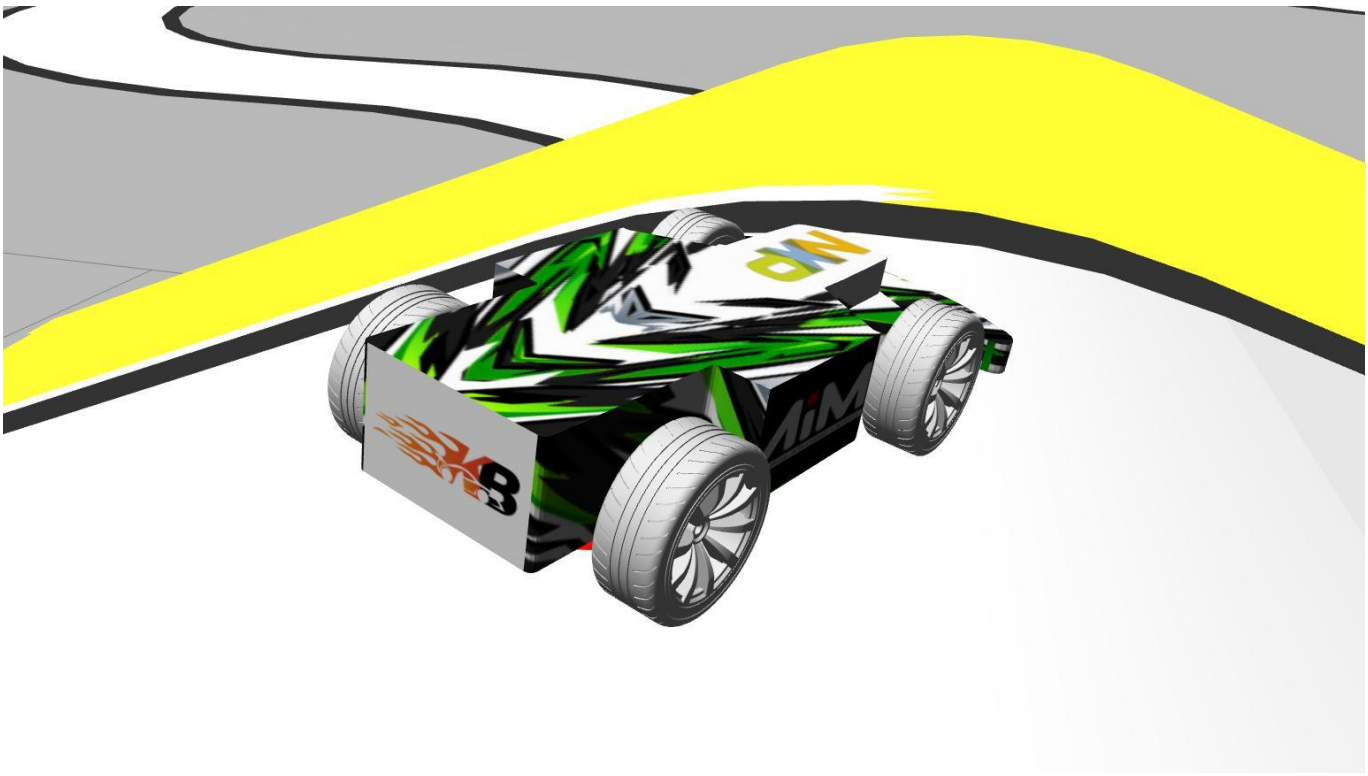


2.

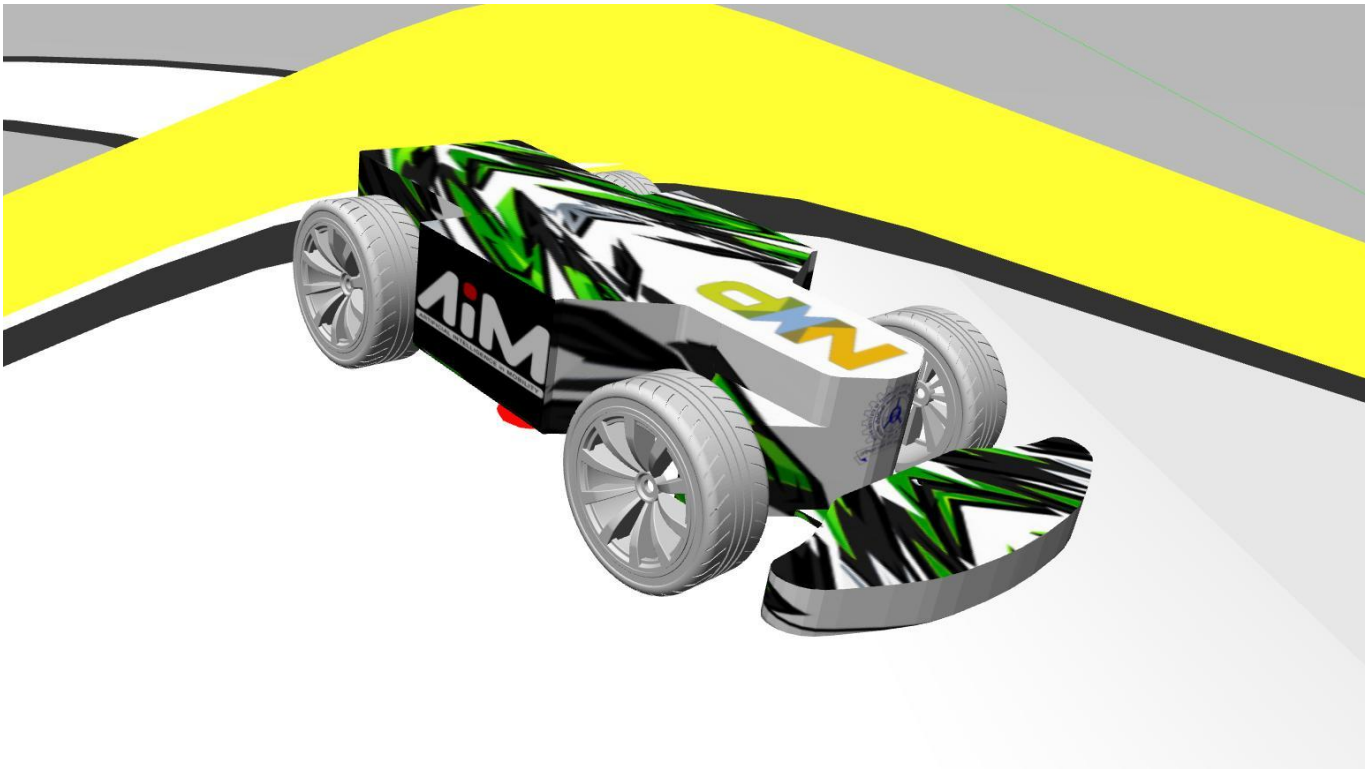
3.

4.

5.



6.



Architecture Details

The methodology we used here can be described as follows:-

At any point in time, a car can have the 3 possibilities that can be either it can see both vectors or just one vector or no vector. So for these 3 cases, we can define 3 conditions as follows:-

a. 2 vectors:- In this case, we used a PID controller, which works on the x-coordinates to control the amount of the steer. These coordinates can be mathematically expressed as :-

m0_x = (m0_x0 + m0_x1)/2
m1_x = (m1_x0 + m1_x1)/2

m_x = (m0_x +m1_x)/2

This m_x is considered as the current value in PID and the x-coordinate of the center of the frame window is considered as the desired goal.

b. 1 vector:- This case can be subdivided into 2 cases as follows:-

I. Inclined vector:- In this case, our steer is directly proportional to the slope of the vector. The proportionality constant is the steer_factor which is responsible for the max amplification of the slope factor.

II. Horizontal/Vertical vector:- In this case, we provide some constant speed depending on the situation.

c. 0 vector:- In this case, we provide some constant speed depending on the situation.

Algorithm Used & Any other Architecture Information

Our algorithm majorly consists of 2 divisions: the ground part and the bridge part. Both the parts are again divided into 3 sub-parts depending on the number of vectors visible in the pixy camera, i.e. 2 vectors, 1 vector, and no vector. As previously mentioned the functioning of each part in general, now we dive into the depth of each part according to the flow of code in the arena.

We are dividing our code into 2 parts based on the altitude of the cupcar we got by subscribing to the odometry of the vehicle. For the sake of convenience, we consider the round value of the altitude. Hence with the help of it, we can distinguish whether the cupcar is on the ground or a bridge.

So based on these we can go with such a flow of code:-

1. Altitude = 0.04: It means the cupcar is moving on the ground hence the 3 parts of the ground case will work here.

2. Altitude > 0.04: It means the car is not on the ground then it can be found in 2 places that are either upon bridges or on a ruddle. so we make a condition that checks whether the bridge is completed or not. Once both bridges get completed the change in altitude does not allow the code to work what is mentioned under the bridge algorithm on ruddle since there also altitude is greater than 0.04. But on ruddle the max altitude is not much, hence we can use the ground algorithms with a slight change in parameters.

3. There is a question about how it controls its speed upon the bridges? The answer to the question is simple: if the car reaches the bridge while going up and down, speed is proportional to the altitude of the cupcar, i.e. a linear relation between altitude to control speed. But the question is what is the proportionality factor? The factor while going up is much more than the factor while coming down. This is done so that car doesn't overspeed while moving down, and doesn't underspeed while going up. The perfect combination of these factors was determined through several test runs. The perfect combination of these factors doesn't allow the car to bump on the bridges.

4. Once the max height of the bridge is reached it marks the condition reached, which gets unmarked again when it reaches ground level after increasing the completed bridge count. Hence after completion of both the bridges the bridge count becomes 2 and then a change in altitude doesn't allow the

bridge algorithm to work. Hence at the time of ruddle, our ground algorithm only works. Since it also includes a condition when bridge count is 2 other than altitude is 0.04.

5. Now once both bridges are completed we increase the speed of the cupcar depending upon the path and its speed decreases once there is a change in altitude since it means a ruddle is reached.

External Libraries used:

1. **simple_pid:** This is used to provide the PID control to the steer in the case of 2 vectors.