e-Yantra Robotics Competition - 2020-21
Nirikshak Bot
Task 4C - Theme and Implementation Analysis

<Team-ID>

Team Leader Name	Suyash Kumar Srivastava
College	IIT(ISM) DHANBAD
Team Leader Email	suyash20191@gmail.com
Date	12/01/2021

Scope

Q1. State the scope of the theme assigned to you. (5)

< Teams should briefly explain in their own words the theme assigned. What in your opinion is the purpose of such an application? You may use figures/diagrams to support your answer (Make neat and labelled diagrams).
Answer format: Text - limit: 100 words. >

ANSWER-1

The scope of our theme is to develop a Quality Assurance System to monitor the Quality of a series of interconnected 2DOF ball balancing platforms, being continuously fed by balls from a ball dispenser. The ball is Expected to reach the destination point (different for different balls) by following a fixed and predetermined path through the mazes which is done by automating the movement of platforms using control logic.

The challenge is to complete the given task in the shortest possible amount of time with least penalties. The theme encourages us to use various techniques like image processing, algorithm building, robotic simulation, etc.

As for the application of ball balancing platforms, they can be readily used in understanding control system applications. It can be used in testing the different aspects of controls, as the non-linearities increases with the increase in associated degree of freedom.

Also the platforms can be employed in the purpose of segregating different kinds of spherical objects by focusing on the specific differentiating property either by computer vision or weight/pressure detection.

Testing your knowledge (Theme and Rulebook analysis)

Q2. Consider the following dictionary written in ball_details.json file:

```
{
   “red”         : [“T3_CB1”],
   “green”       : [“T2_CB2”, “T1_CB1”],
   “blue”        : [“T1_CB3”, “T3_CB3”]
}
```

Based on the dictionary given above, write the correct Collection Box for the following sequence of balls dispensed by BD: (5)

< This question is to check if you have understood how to interpret the ball_details.json file correctly. Hence fill in the answers carefully in the table below>

Sequence	Color	Collection Box Name
4th	Green	**T1_CB1**
5th	Blue	**T3_CB3**
2nd	Blue	**T1_CB3**
3rd	Red	**T3_CB1**
1st	Green	**T2_CB2**

Q3. Consider the JSON configuration given in Q2.

a) **What are the ENTRY and EXIT cell coordinates used by the *first green ball* for all the tables it is passing through? (2)**
b) **What are the ENTRY and EXIT cell coordinates used by the *second blue ball* for all the tables it is passing through? (2)**
c) **What are the ENTRY and EXIT cell coordinates used by the *first red ball* for all the tables it is passing through? (2)**

< This question is to check if you have understood the Arena section of the Rulebook. Write your answers point wise for (a), (b) and (c)>

ANSWER-3

(a)

S.NO	PLATFORM	ENTRY	EXIT
1	T4	(0,5)	(9,4)
2	T2	(0,4)	(9,5)

(b)

S.NO	PLATFORM	ENTRY	EXIT
1	T4	(0,5)	(9,4)
2	T3	(4,9)	(0,4)

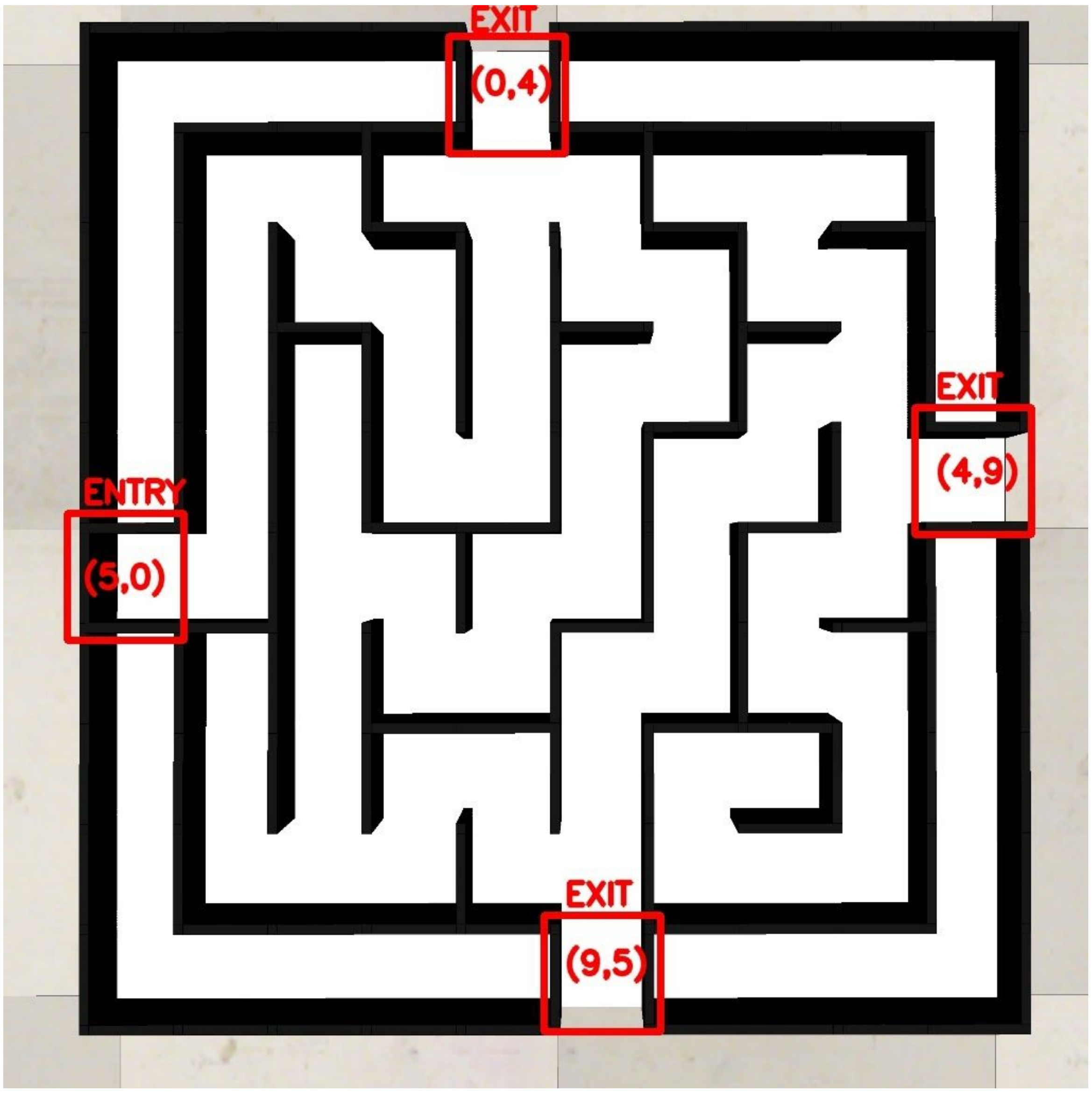
(c)

S.NO	PLATFORM	ENTRY	EXIT
1	T4	(0,5)	(9,4)
2	T3	(4,9)	(9,5)

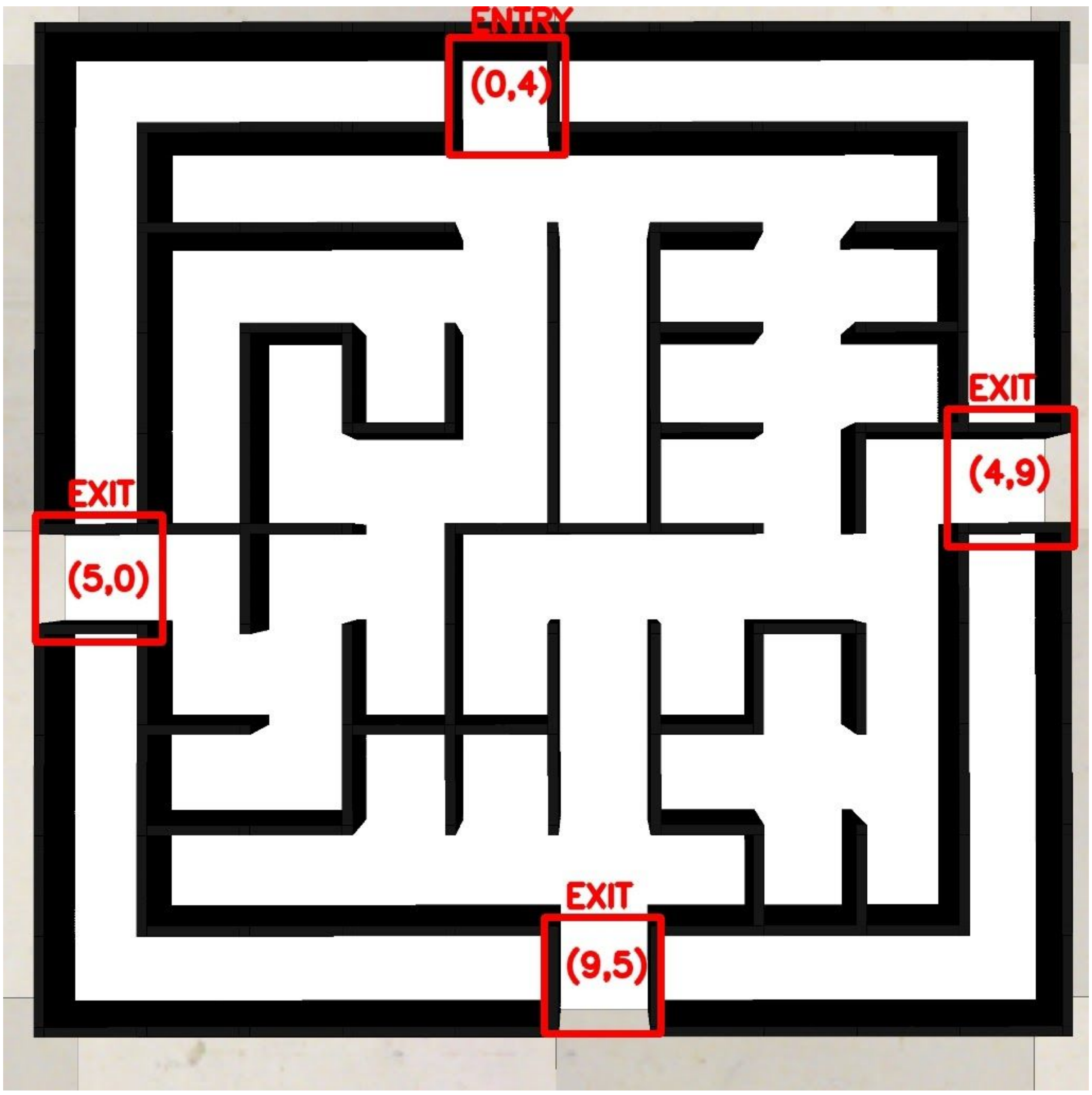
Q4. Download the *task_4c_maze_images.zip* file from this link (from Task 4C page). The images have been named maze_t1.jpg, maze_t2.jpg and so on (according to the Theme Run Requirements part under Theme Description section of the Rulebook). Generate these mazes on the single Platform Table one by one according to the resultant maze images shown in Figure 10 and 12 of Arena section in CoppeliaSim and capture a top-view screenshot for all of them. (4)

< Make sure to carve the respective EXIT points for all the mazes on Platform Table. Paste all the screenshots in this document. All the screenshot images should be properly labelled with ENTRY and EXIT clearly marked>

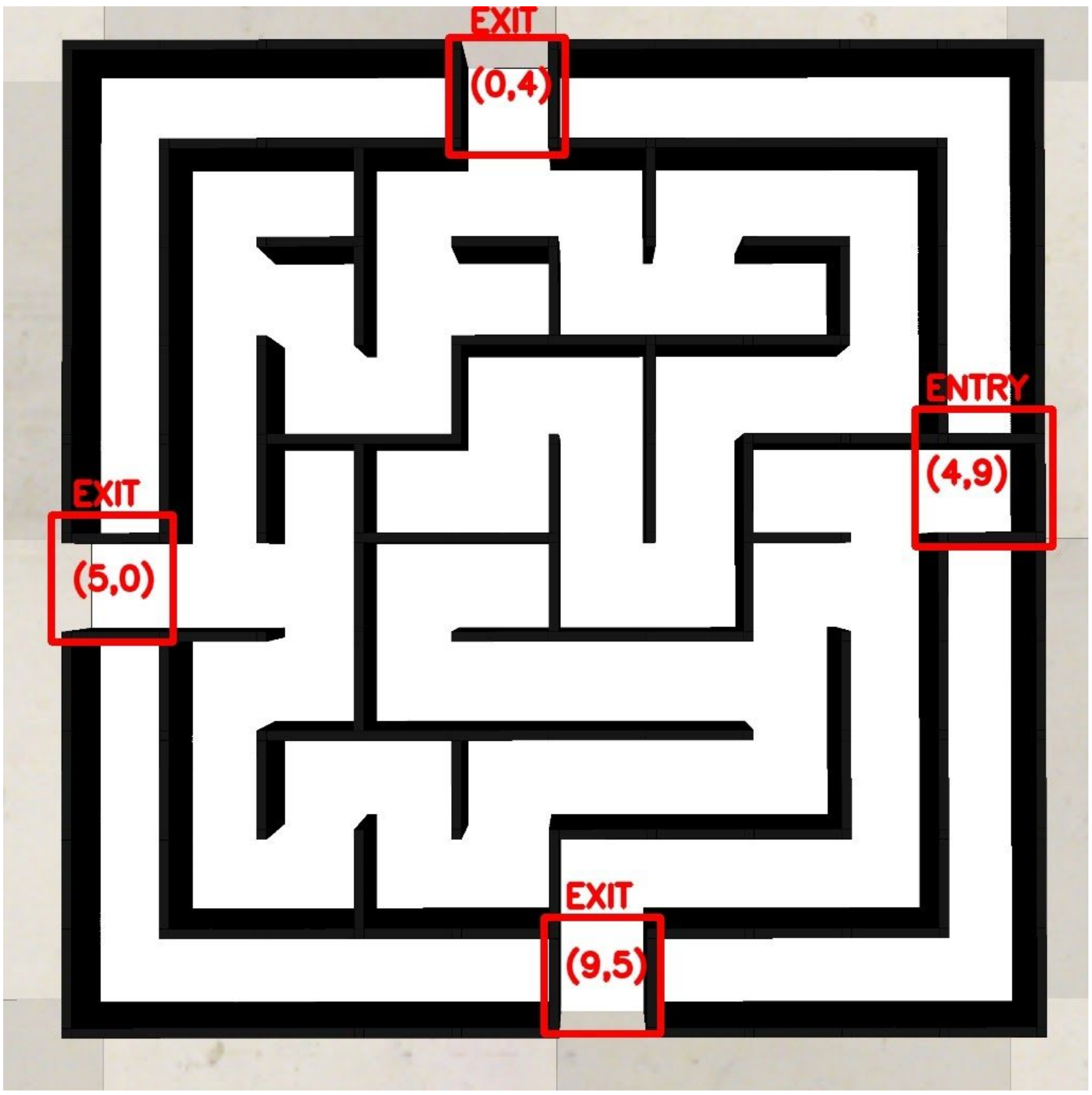
(a)maze_t1



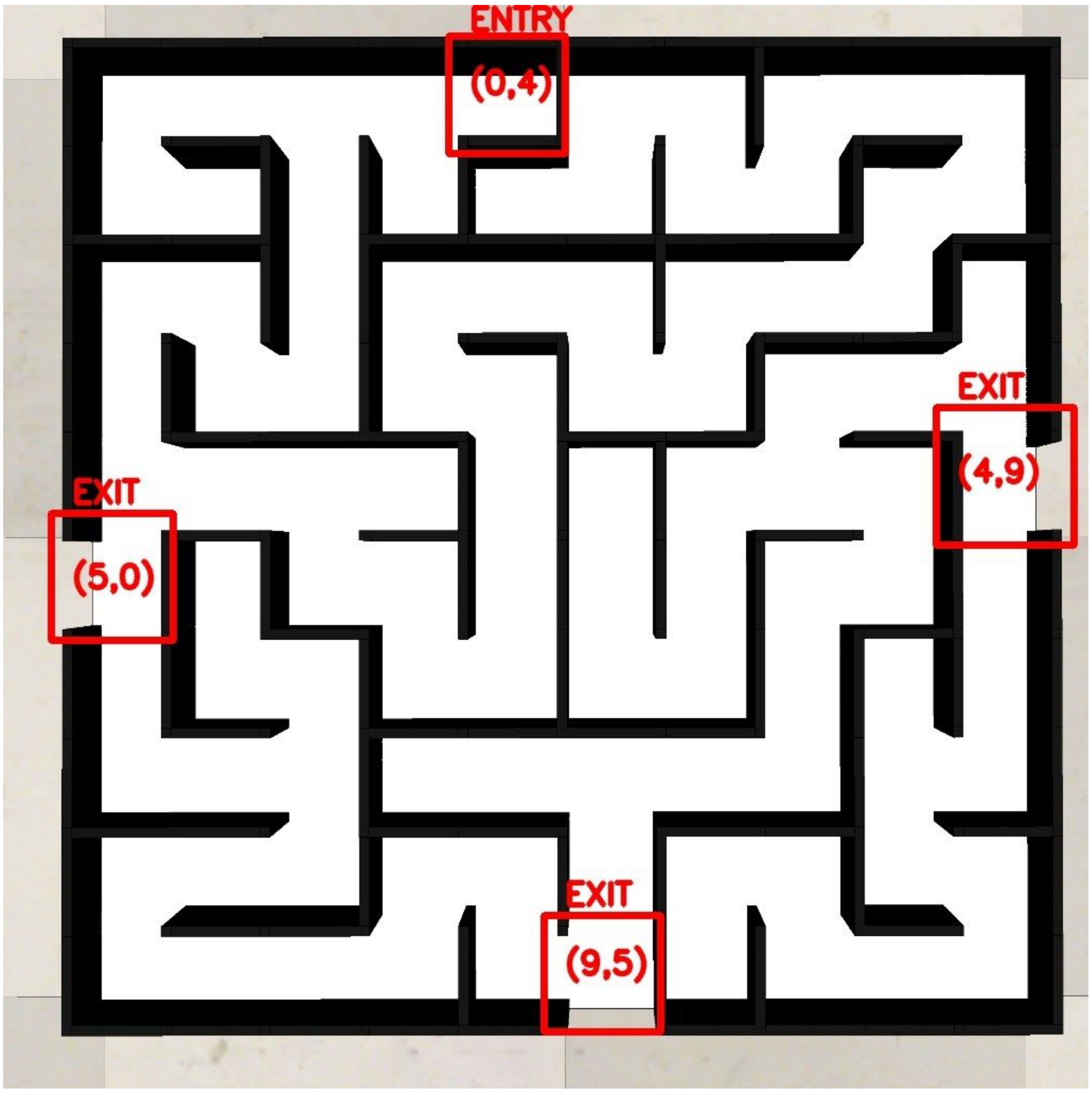
(b) maze_t2



(c) maze_t3



(d) maze_t4



Q5. Consider the following table showing the scenario for each ball and calculate the final score: (5)

	CI	CP	CD	CT_4	CT_x	TB_4	TB_x	HP
1st	0	1	1	1	1	1	1	1
2nd	1	1	1	1	1	5	10	8
3rd	0	1	0	1	1	3	13	4
4th	1	0	0	1	1	10	20	15
5th	1	1	0	1	1	2	17	2

CM1	CM2	CM3	CM4
1	1	0	1

< Show your calculations in detail below>

ANSWER-5
1st:
(0*10)+(1*100)+(1*50)+(1*1*100)+(1*1*1*100)+(0*1*1*1*10)+(0*1*1*1*10)-(1*10)=340
2nd:
(1*10)+(1*100)+(1*50)+(1*1*100)+(1*1*1*100)+(1*1*1*5*10)+(1*1*1*10*10)-(8*10)=430
3rd:
(0*10)+(1*100)+(0*50)+(1*1*100)+(1*0*1*100)+(0*1*1*3*10)+(0*1*0*13*10)-(4*10)=160
4th:
(1*10)+(0*100)+(0*50)+(1*1*100)+(0*1*1*100)+(1*0*1*10*10)+(1*0*20*10)-(15*10)=-60
5th:
(1*10)+(1*100)+(0*50)+(1*1*100)+(1*0*1*100)+(1*1*1*2*10)+(1*0*1*17*10)-(2*10)=210

Total=340+430+160-60+210+3*50=1230 (ans)

Mechanism

Q6. Explain the mechanism that you used for your ball balancing platform. (5)

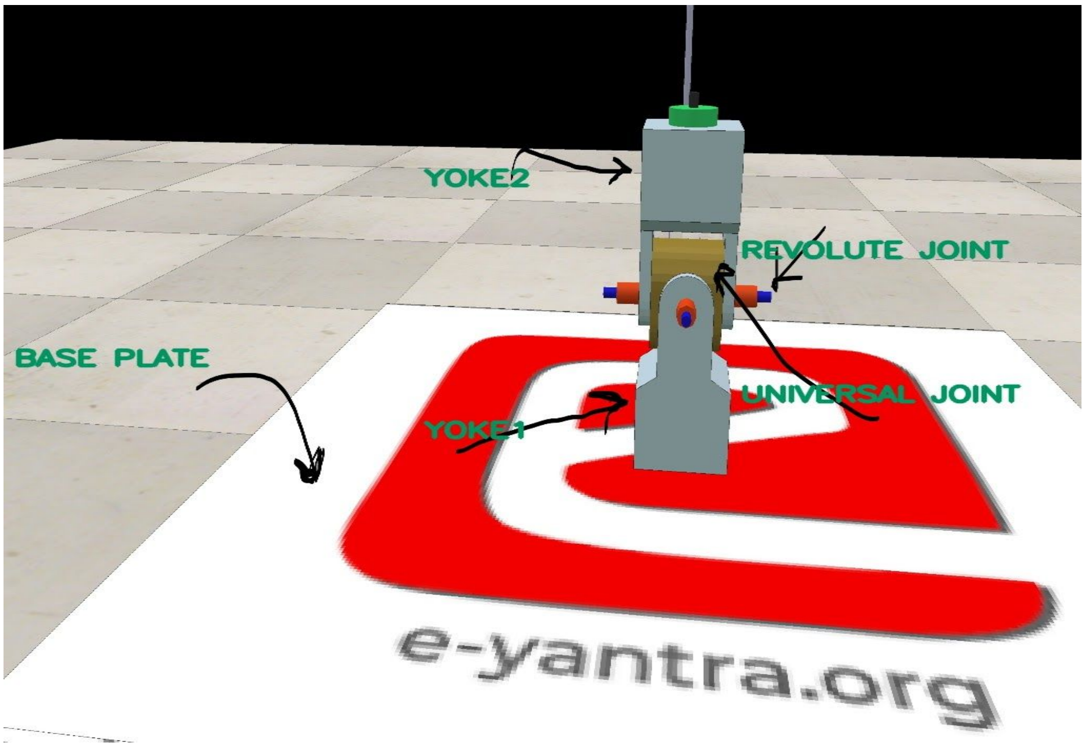
< You must explain the mechanical construction of your ball balancing platform and how have you connected all the different components provided to you. Make properly labelled diagrams to show the same. You may also use screenshots of the CoppeliaSim scene to demonstrate your mechanism.>

ANSWER-6

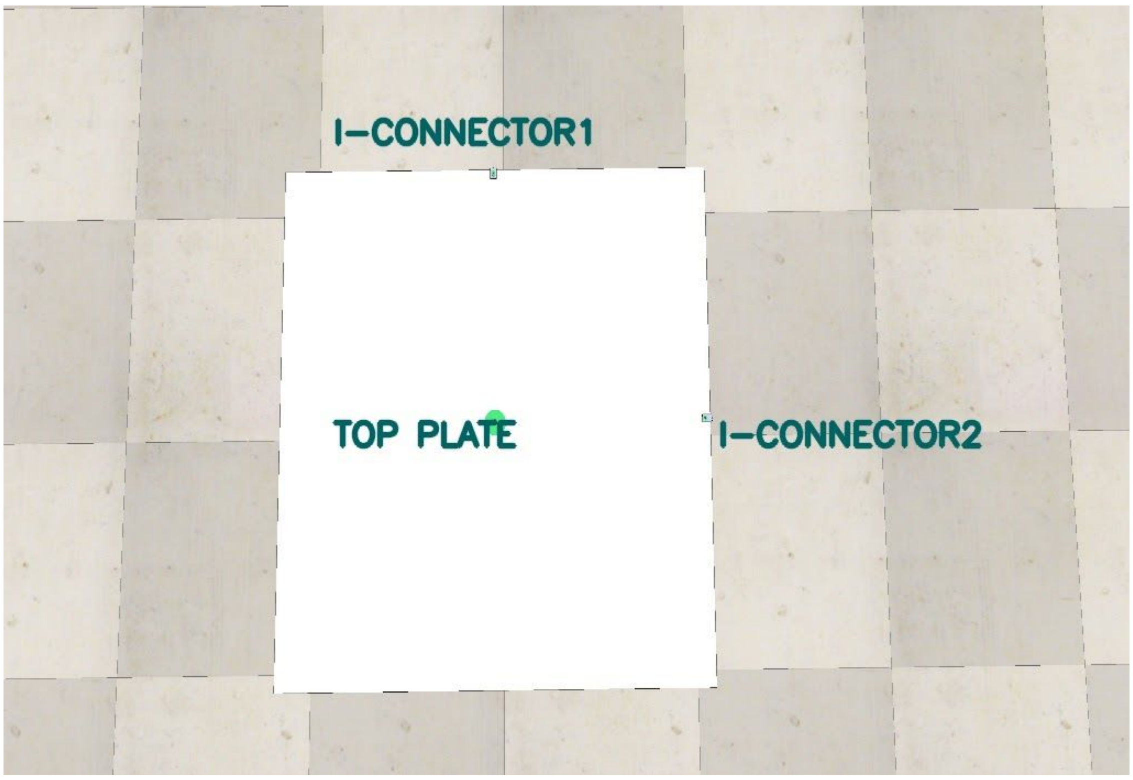
- For the mechanical construction of the ball balancing platform, we have used one top plate, one base plate, two yokes, one universal joint, six revolute joints, two servo motors, four servo holders, two servo fins, two connecting rods, two spherical joints, two I-connectors, several dummies and several force sensors.

- **STEPS INVOLVED IN THE CONNECTION OF VARIOUS COMPONENTS**

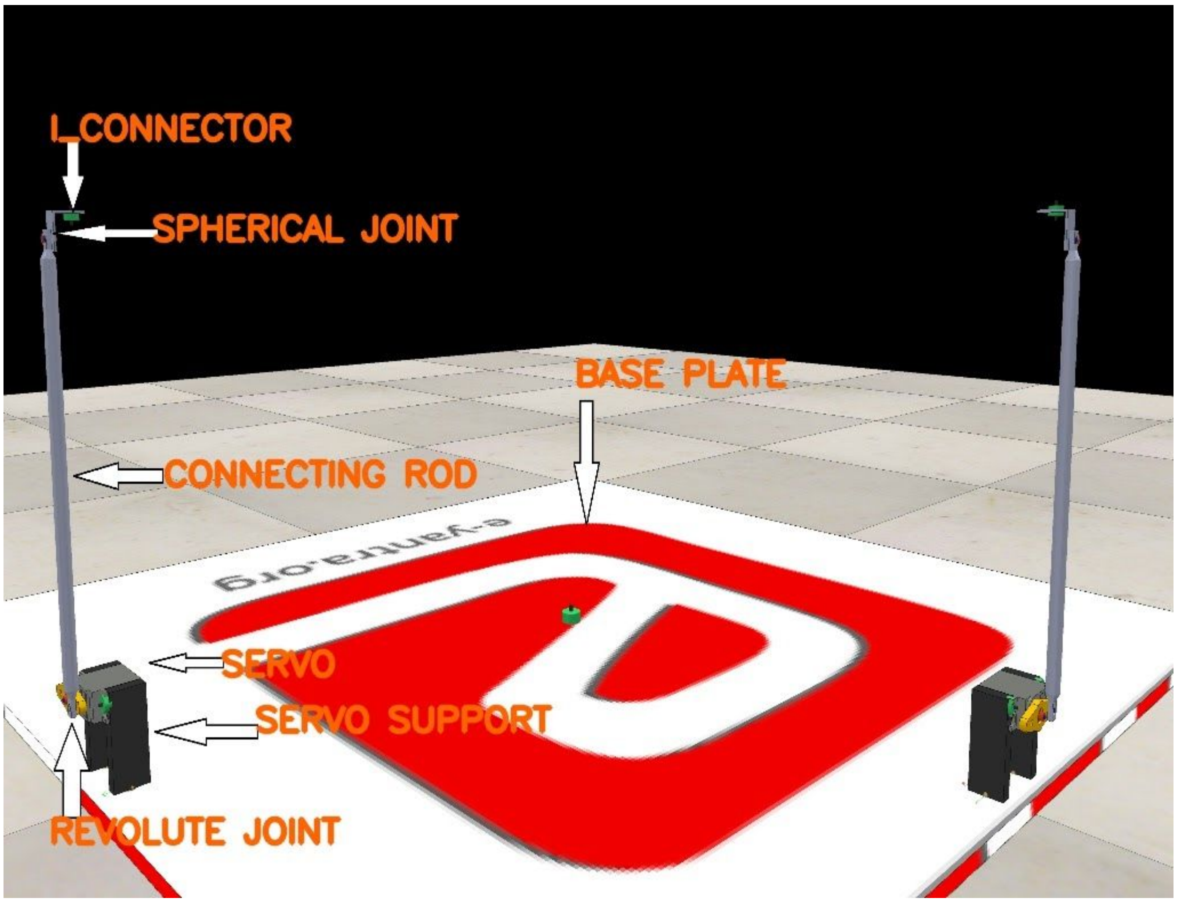
    - Imported the base plate plate model provided and placed it such that its center is at the position (0,0,5.5e-03). Then with the help of a force sensor connected a yoke directly above the point (0,0) and with its base resting on the X-Y plane. Then inserted a universal joint at its open end and placed two revolute joints in mutually perpendicular directions passing through the universal joint. Then we connected another yoke to one of these revolute joints with its open end pointing downwards. This setup provides a great amount of flexibility to manage the top plate. This step has shown below in the screenshot captured from coppelia sim.



- Added The top plate provided in the question above the yoke2 with the help of a force sensor. It is also made sure that this top plate is directly above the Base plate such that its center and base plate sensor differ only in their Z coordinates. We then added two I-Connectors to the top plate at positions (-3.5e-03 m,4.915e-01 m,3.13e-01 m) and (-4.9192e-01 m,-3.88e-03 m,3.13e-01 m). These are connected to the top plate with the help of two force sensors. The figure for this is shown below:

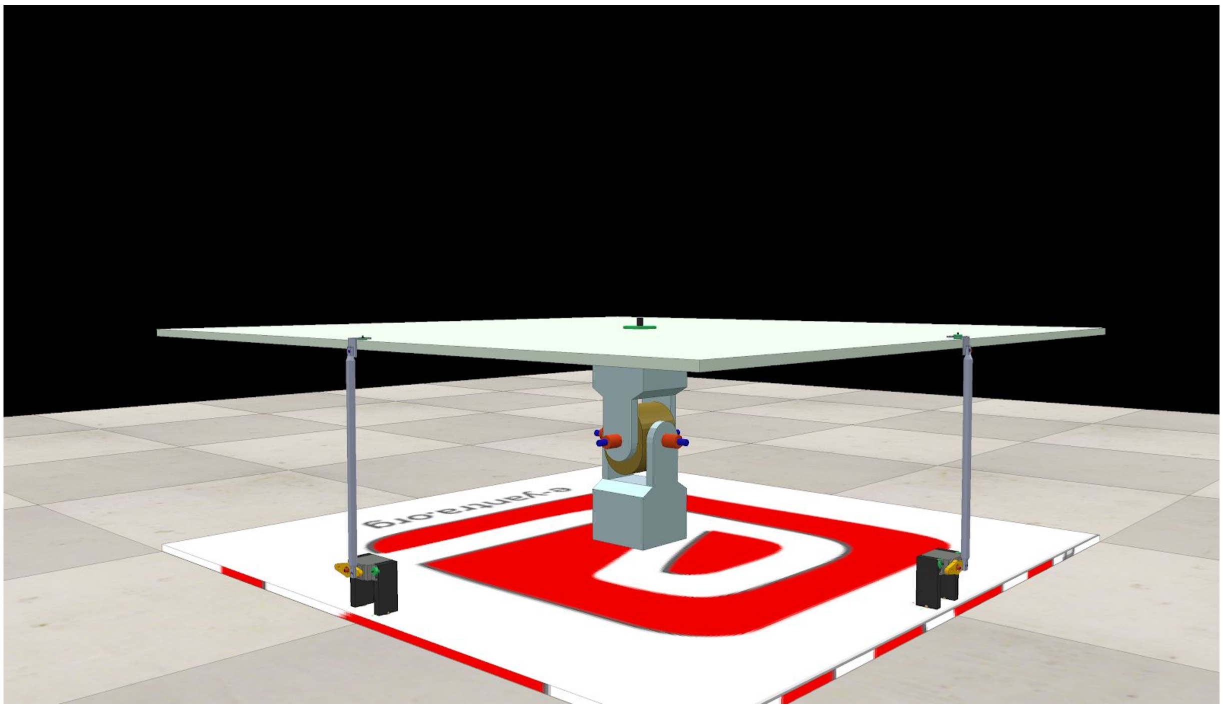


- In the third step, with the help of two spherical joints we connected two connecting rods , one to each of the I-connectors. At the other end of each of these rods we connected a servo fin with the help of revolute joints.Each of these servo fins were in turn connected to one servo motor (again by using a revolute joint). Each of these servos were in turn connected to Servo holders on either side with the help of force sensors. The last part of constructing the model was to connect these servo holders to the base plate with the help of force sensors. This marked the end of the construction of our ball balancing model. This step is shown in the figure below:



- **SCREENSHOT OF THE ENTIRE MODEL**

  Shown below is the screenshot of the entire model built by us. It contains all the parts described in the above points.



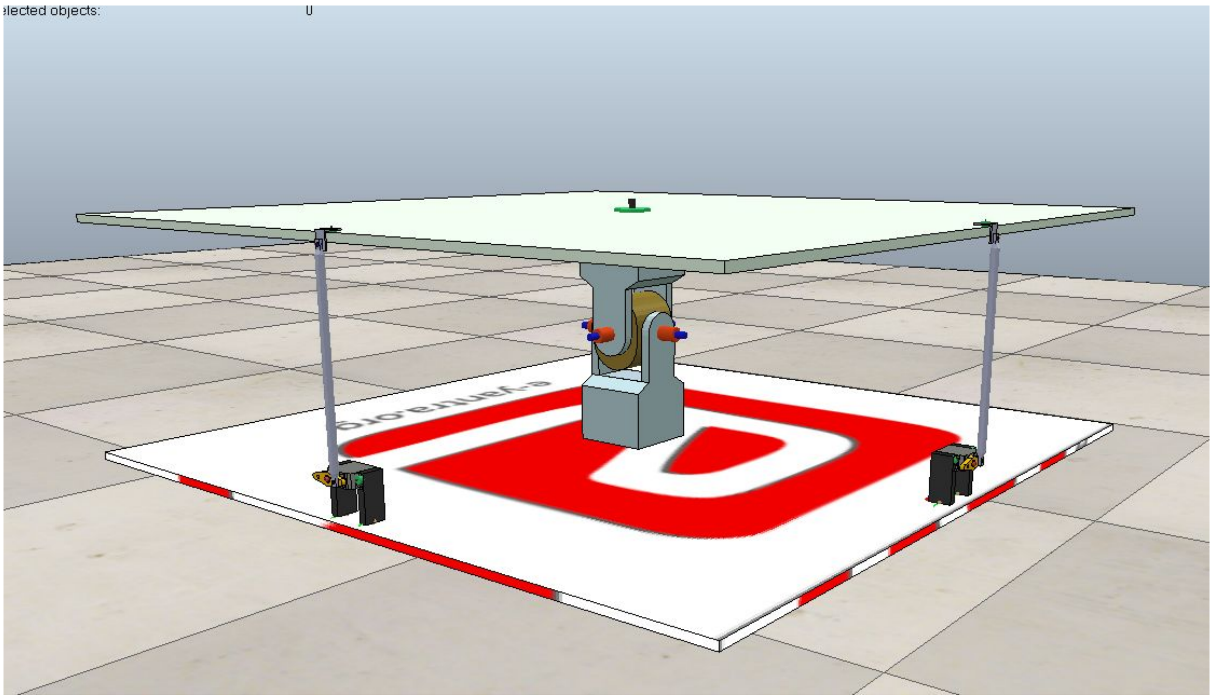
- **<u>BRIEF WORKING PRINCIPLE OF THE MODEL</u>**

  The state and velocity of the ball in x and y direction will be controlled by the rotation of revolute joints(both direction and magnitude) between servo fin and servo motor which in turn would be controlled by our control logic. Hence according to our logic we can control the position of the ball.

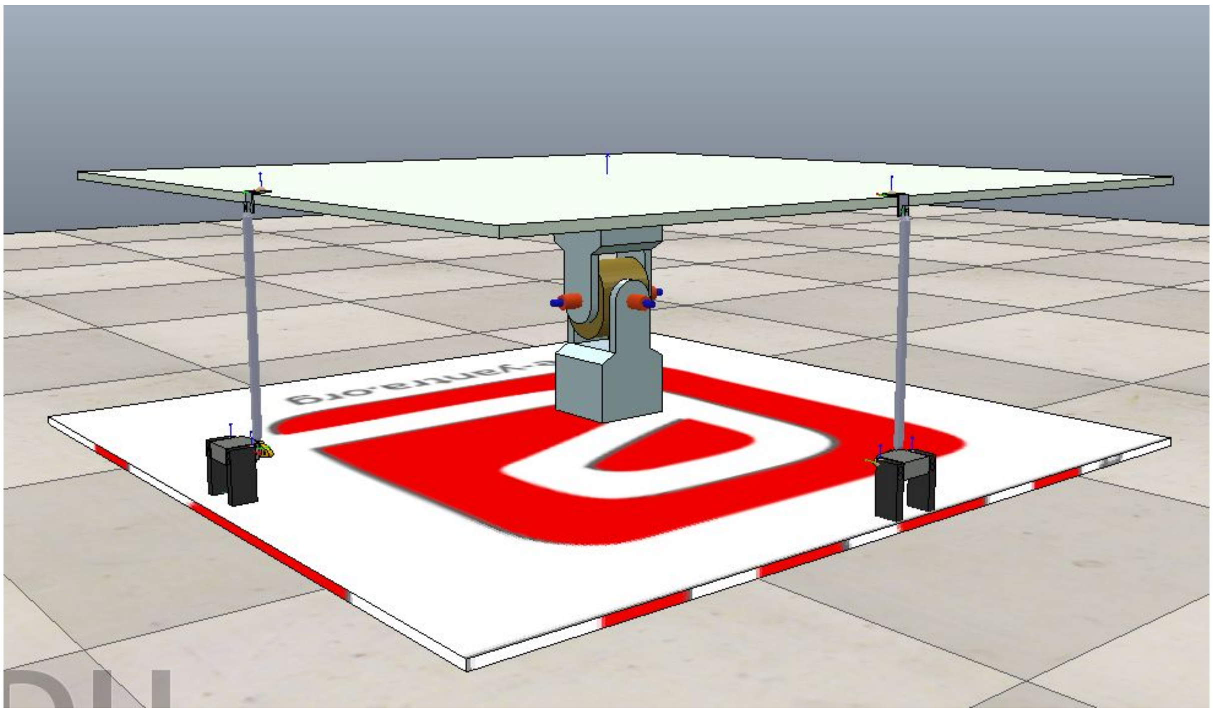
Q7. In Task 1C, you were given the task to design the ball balancing platform while in Task 3, you were given the task to use this ball balancing platform to control the position of the ball on top of it. How did your ball balancing platform change between these tasks? (5)
< Explain in brief how your design changed in the subsequent tasks. If your design did not change, then justify your reasons for the same.
Answer format: Text - limit: 100 words. .>

<u>ANSWER 7</u>

- Between task 1c and task 3, to align the top plate with the base plate we changed the orientation of servo motors (rotated by 180 degrees).
- We replaced the force sensor between the l_connector and connecting rod with spherical joints to allow better movement.
- To reduce the number of dummies we replaced the dummies connecting yoke to top plate, top plate to l connectors and servo motor to servo holder with force sensors and then used dummies to connect servo holder with base plate.
- We changed the hierarchy accordingly to fit in the changes in connections as suggested in remark of task 1 and then grouped the components wherever possible.



The updated model in accordance with task 3.



Model made in 1c.

Path Planning

Q8. What kind of path planning algorithm did you use for finding the shortest path for the given maze images (in Task 4A)? (5)

< Explain the logic behind the algorithm and the reason for your choice if any. You can use a pseudo-code and/or flowcharts to help elucidate your answer. >

ANSWER-8

We Used The Dijkstra's Shortest Path Finding algorithm For finding the shortest path from the start point to the destination point in different mazes.

Dijkstra was the most suitable algorithm for us primarily because of its low time complexity and simple implementation reasons. This also saved us from traversing all the possible paths and provided with the shortest path. This was a great boost from our Initial Brute Force Backtracking Implementation. Hence this was the most ideal choice for us.

LOGIC BEHIND DIJKSTRA'S ALGORITHM

- This algorithm is used to find the shortest path between a source vertex to all the vertices in the graph. It works well for both directed as well as undirected graphs and is a standard technique to find the shortest path.

- We start by assigning a distance value to all the vertices of the graph by INFINITY and assigning distance as 0 for the source.
- We kept two arrays for our implementation, one keeping track of the visited nodes and other containing the nodes to be visited.
- Then we iterate while there are nodes to visit, keeping the visited nodes container updated. For all the neighbours of the current node visited:
    - If the neighbour is already visited, we only update it's distance value if it is less than its current value.
    - Else we update it's distance value as sum of edge length and distance value of the node we are visiting and push it in the array of nodes to visit.
- This loop ends when there are no more nodes left to visit. If any node is not visited, then it is unreachable.
- Furthermore to obtain the path, a parent array is kept, which by backtracking starting from the target node gives the shortest path's coordinates.

PSEUDO-CODE

```
function Shortest_path(Graph, Source_vertex):
    N=Number of Nodes in the graph
    Dist[N]={INFINITY} //setting the distance of all the vertices to infinity initially
    Par[N] = {-1}
    toVisit = [], visited[N] = [False]
    toVisit.push_back(source)
    Dist[Source]=0

    while len(toVisit) != 0:
        curr = toVisit[0] // always taking the first node from toVisit array
```

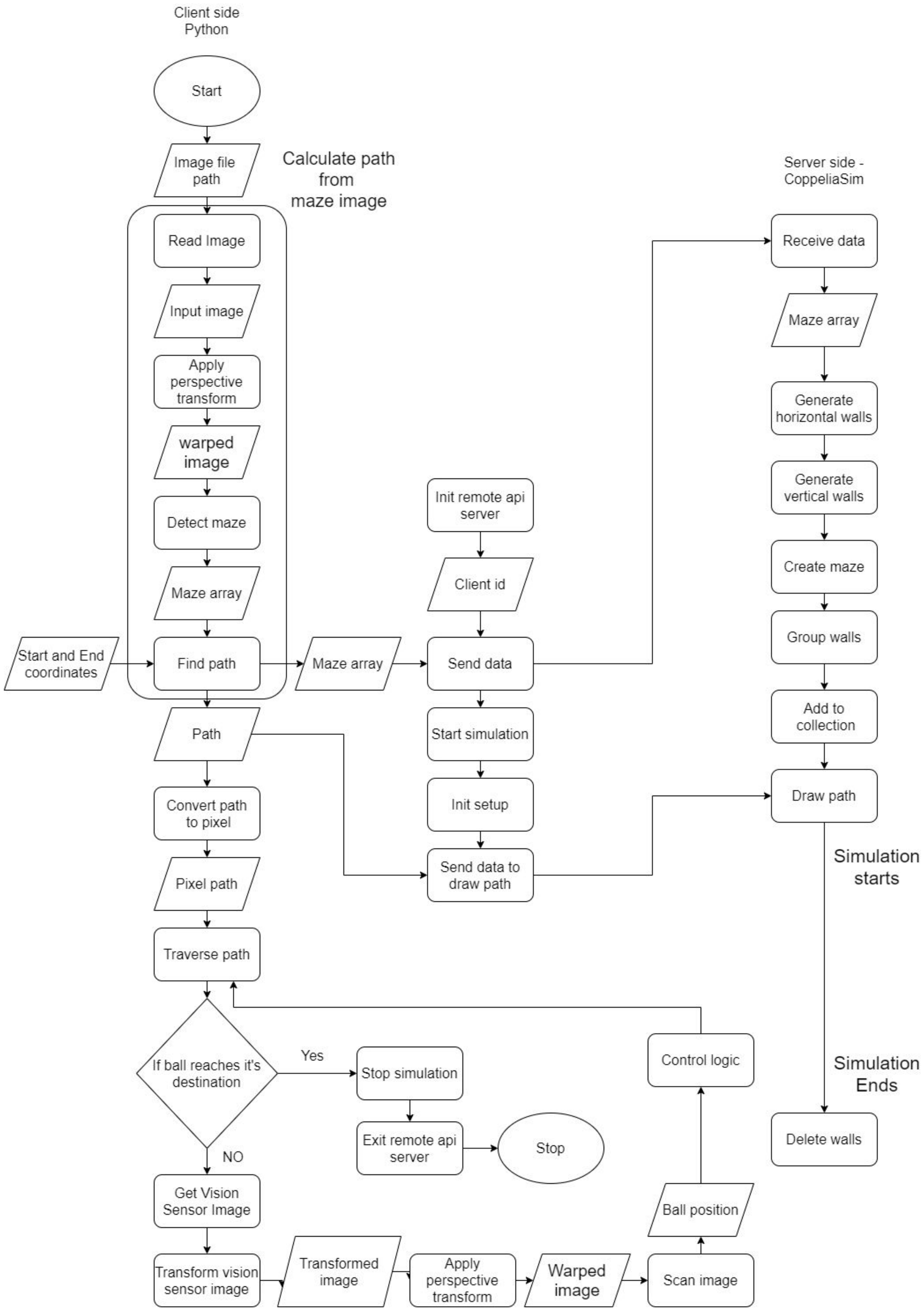
```
            if already visited:
                continue

            visited[curr] = True
            for neighbours of curr:
                if distance[neighbour] > distance[curr] + edge_len:
                    distance[neighbour] = distance[curr] + edge_len
                    toVisit.push_back(neighbour)
                    Par[neighbour] = Par[curr]

            del toVisit[0]

        shortestPath = []
        while par[tar] != -1:
            shortestPath.push_back(coord_of_tar)
            tar = Par[tar]

        reverse shortestPath
```

// This implementation uses dijkstra as a shortest path finding algo and backtracking to get the coordinates.

Algorithm Analysis

Q9. Draw a flowchart illustrating the algorithm/ strategy you propose to use for theme implementation. (7)

< The flowchart should elaborate on every possible function that you will be using for completing all the Theme Run.
Follow the standard pictorial representation used to draw the flowchart. >


Client side
Python
Start
Image file path
Calculate path from maze image
Read Image
Input image
Apply perspective transform
warped image
Detect maze
Maze array
Start and End coordinates
Find path
Maze array
Path
Convert path to pixel
Pixel path
Traverse path
If ball reaches it's destination
Yes
NO
Stop simulation
Exit remote api server
Stop
Get Vision Sensor Image
Transform vision sensor image
Transformed image
Apply perspective transform
Warped image
Scan image
Ball position
Control logic
Init remote api server
Client id
Send data
Start simulation
Init setup
Send data to draw path
Server side - CoppeliaSim
Receive data
Maze array
Generate horizontal walls
Generate vertical walls
Create maze
Group walls
Add to collection
Draw path
Simulation starts
Simulation Ends
Delete walls

Challenges

Q10. What are the major challenges that you have faced till now and the ones that you can anticipate in addressing this theme and how do you propose to tackle them?
(3)

< Answer format: Bullet points 1. Challenge 12. Challenge 2 3. Challenge 3, etc. >

- Challenge-1
  In task 1 we faced some degree of difficulty to tackle and learn all the OpenCV concepts and functions which were to be used in the task. Even after having done that we had to spend a lot of time adjusting the parameters to pass all the test cases. In 1a, the shapes had to be detected with a clear boundary that required tuning parameters. In 1b, the corner cases troubled us a lot, we had to assume the width of the border to be something and then code accordingly, we failed, next assumption and so on. Having had no prior experience of coppelia sim, Task1c also offered a lot of challenges initially, getting used to the hierarchy, parent child relationship, model import export, position adjustment, etc., were among them.

- Challenge-2
  In task 2 came with a new set of challenges. In task 2a, we had to get familiar with connecting and using Python Remote API to control the coppelia sim software, some commands were tricky and tedious, choosing the right threshold in the applyPerspectiveTransformImage() function was also challenging. In Task 2b, generating horizontal and vertical walls and establishing connection between the Coppelia sim and the Python Remote API went smoothly, but creating a maze by removing the required segment of a wall proved challenging. We had to cater a lot of corner cases. Specifically the rightmost Vertical Wall presented a lot of challenges and we had to dedicate a lot of effort to have it covered.

- Challenge-3
  In task3, we got to know the basics of PID controlling for the very first time. The resources given by the organisers proved to be of less importance and we had to resort to independent sources and a lot of confusion regarding how to CODE the concepts emerged. Once this passed, we had to dedicate quite some time to tune our parameters. This task had just too many variables to tune. We had to check for a wide range of values for each parameter and hope for the ball to reach the setpoint smoothly. Even after that we dedicated a lot of time in reducing the overshooting and the time required to get the ball steady. In all, it was quite challenging

- Challenge-4
  In task 4a, we had to choose from among various methods and models. Understanding and implementing them was challenging. We first tried the more basic and Brute Force approach of using Backtracking with memoization, it worked but it was awfully slow. We then moved on to complex but faster algorithms like dijkstra's. Also as the computation increased due to the addition of maze, many of

our teammates faced the challenge of awful lag due to their system limits and hence optimizing the code to get it to run on the test cases smoothly was quite challenging.

IMPORTANT:

- **The document you submit should be in YOUR OWN WORDS. To avoid any copyright violations, you must NOT copy phrases directly from manuals or the web.**
- **The team should NOT mail or upload the document anywhere else except on the portal.**
- **Teams failing to submit the document by the deadline will lose the marks for this task.**
- **e-Yantra WILL NOT entertain any request for extension of deadline for uploading the task.**
- **e-Yantra holds complete discretion to disqualify a team if any foul play is suspected.**