Assignment 4 - Robotic Networks in Balancing

Problem Statement:

• Environment and Robot Definition
– Number of Robots N = 8
– Fixed Left Robot Position R1 − (0, 0)
– Fixed Right Robot Position R8 − (14, 0)
– Magnitude of Maximum Robot Velocity Vmax = 0.15m/s
• Each Moving Robot Ri can access robot pose data for robots Ri−1 and Ri+1 at every instance

• At each instance (as frequently as possible) command a velocity vector (Vxi., Vyi) for every moving robot Ri such that the robots are balanced (equidistant from each other).

Theory:

There are 8 bots of which the two located at ends are fixed and 6 bots in between adjust position to balance. The bots can access the left and right bot velocity so an algorithm is written for them to assist in its movement to attain a balanced position. (equidistant from left and right bot).

The velocity determined by the algorithm was in terms of the velocities in the x and y directions. The drive bots take inputs as unicycle model so we need conversions to their velocities which is provided in the form of linear and angular velocities. The max velocity is restricted to 0.15m/s. Function 'velocity convert' is used to convert velocities along cartesian coordinates to linear and angular velocities. The following helper code provided was used in the program balancing.py

– /odom Contains robot's position and velocity (in local frame) data ,
– /left_odom Contains left robot's position and velocity (in local frame) data
– /right_odom Contains right robot's position and velocity (in local frame) data
– /cmd_vel Command robot's velocity.
The function to convert velocities in the skeleton code works well if the robot orientation and the desired direction of movement differ by less than ± 90°. When the difference is more than 90, the bot can move in reverse direction preferably without turning in a circle with negative linear velocity.

Result :

The balancing.py program is created and the plots for X-coordinate Vs time & X Vs Y coordinate was plotted. The output.txt files are generated and saved in folder to refer the data of Bots at various instances:

Time taken by bots to reach balanced position:

Bot2= 26.5secs, Bot 3= 25.4 secs, Bot4= 25.7secs, Bot 5=25.03 secs, Bot6=23.6 secs

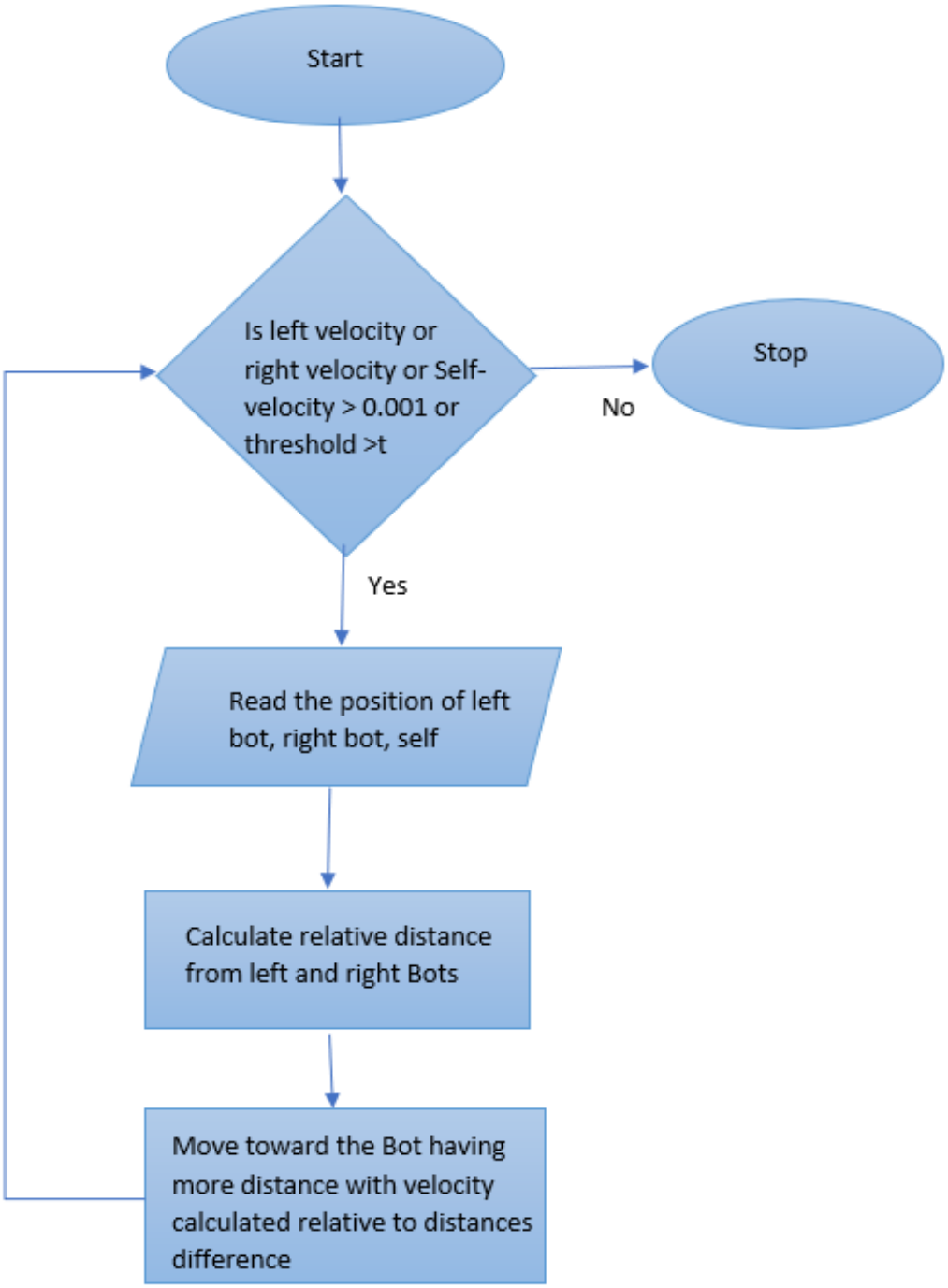
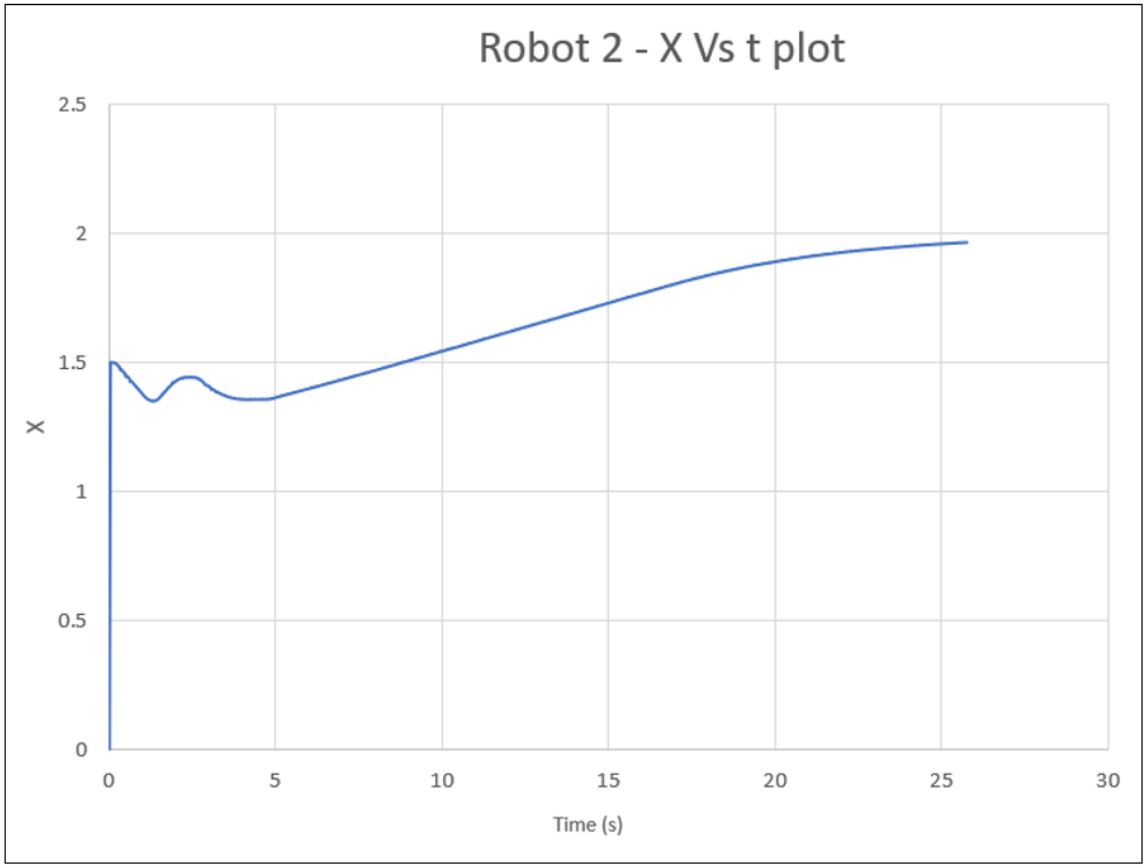
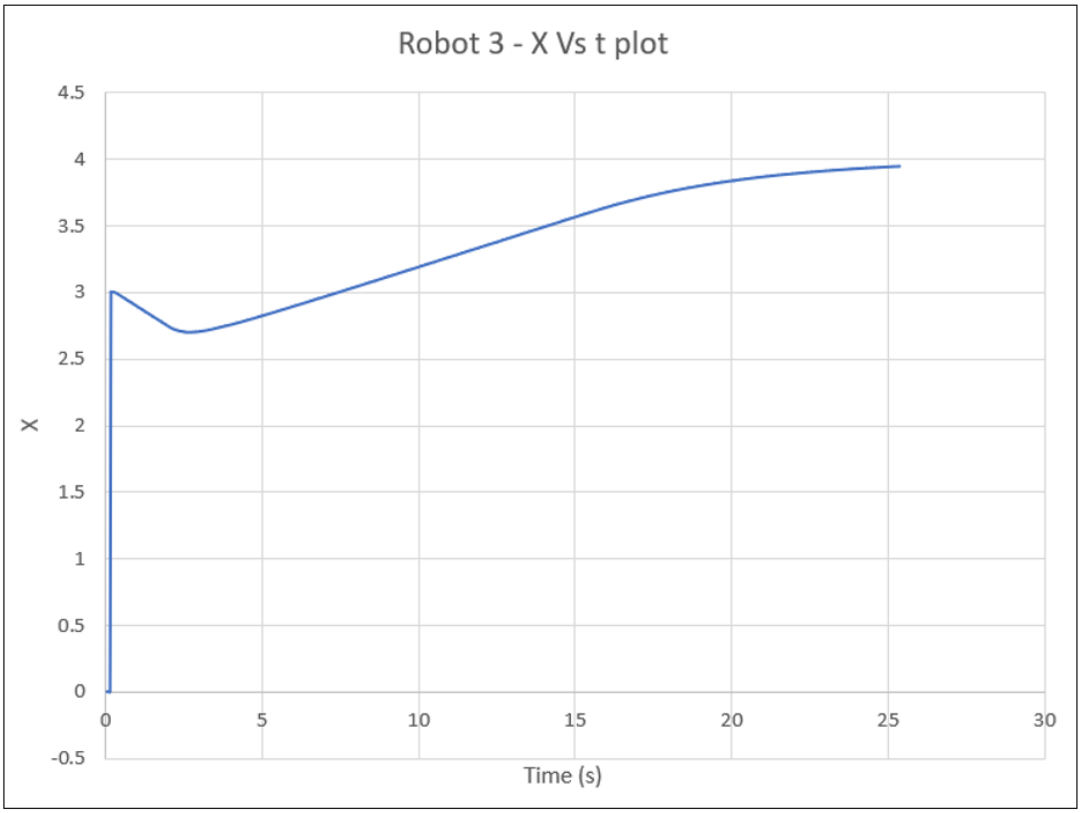
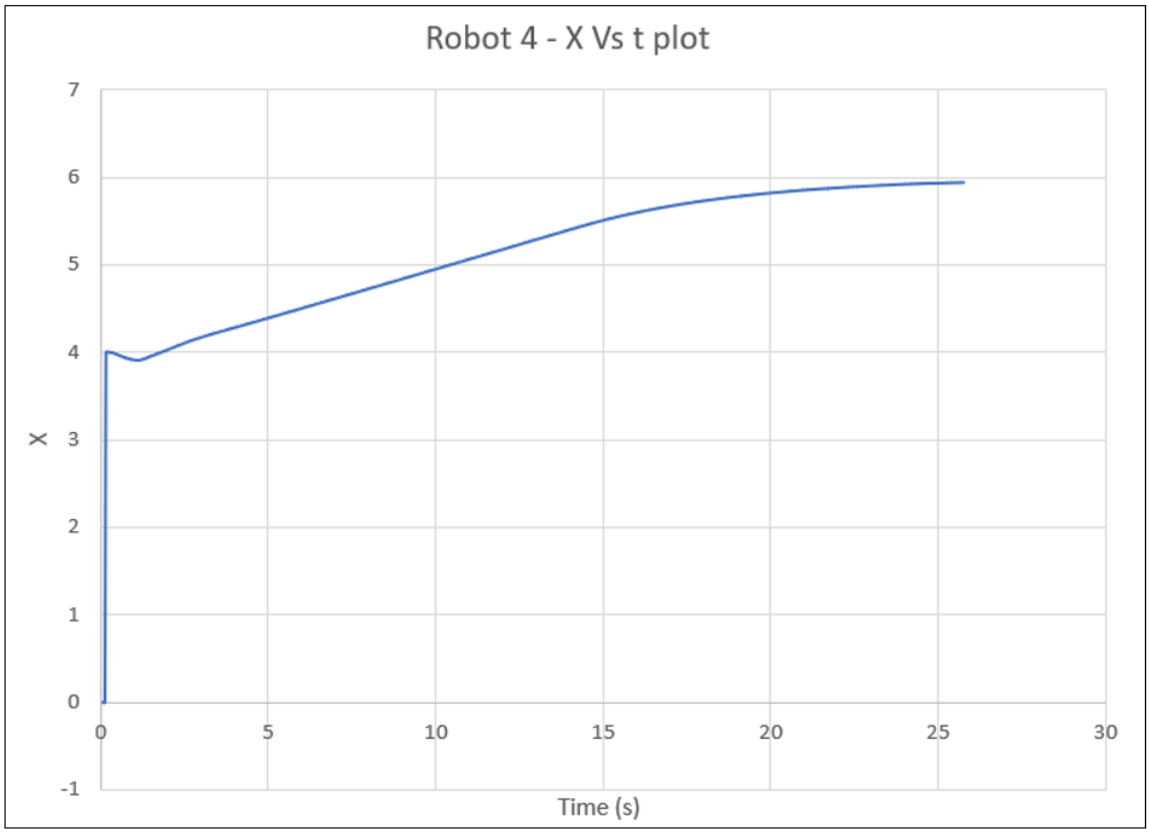
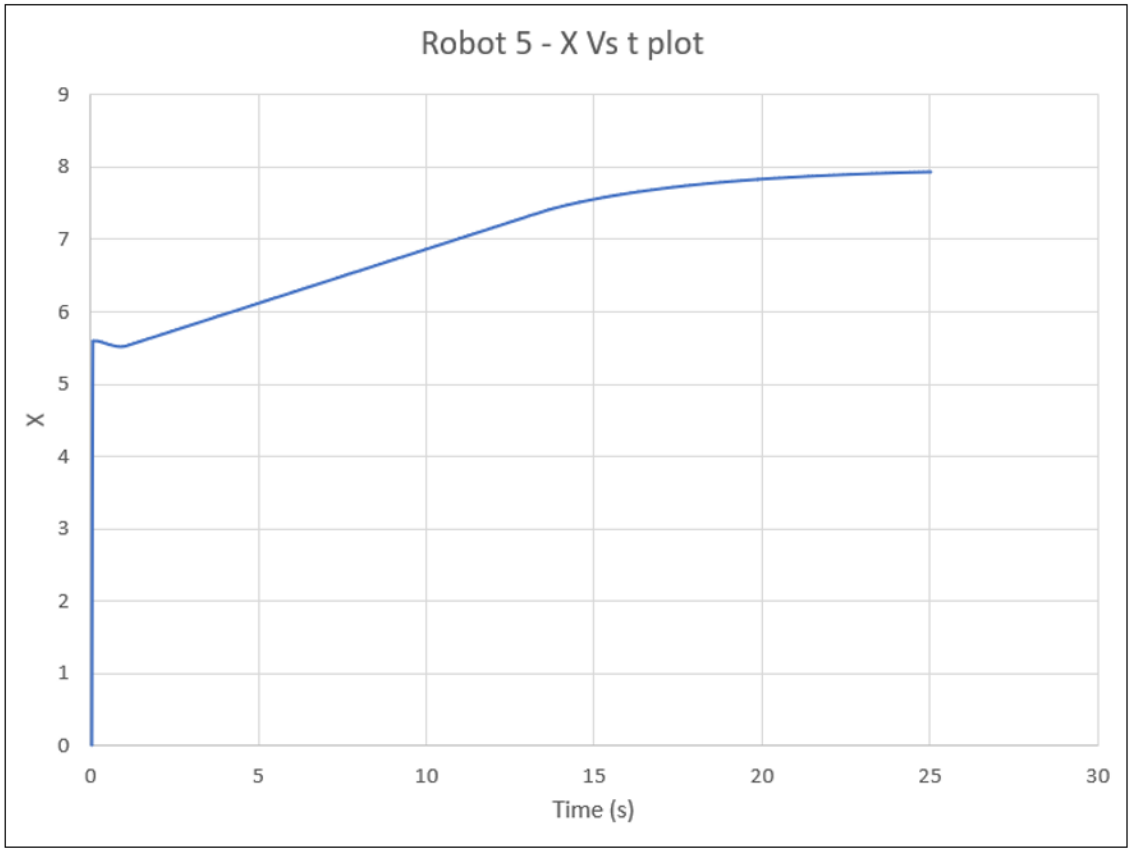


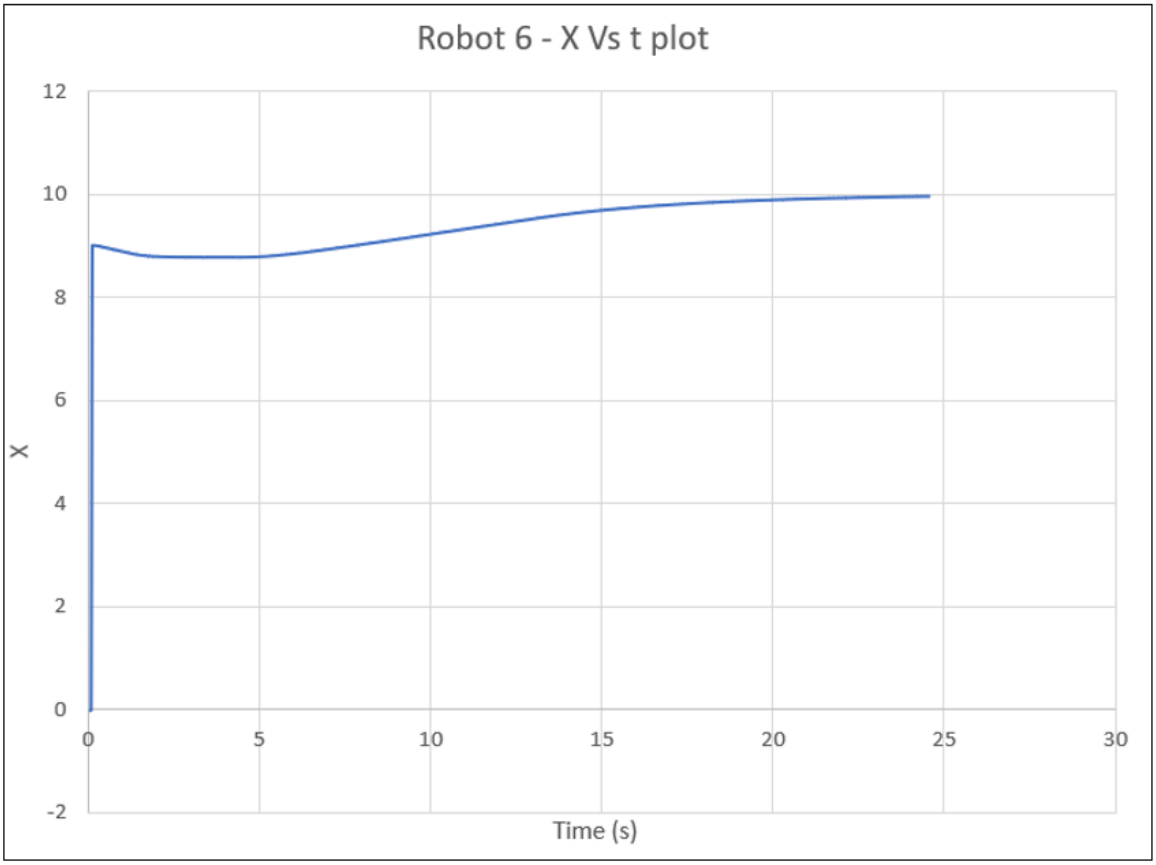
Fig. - Algorithm for Balancing.py

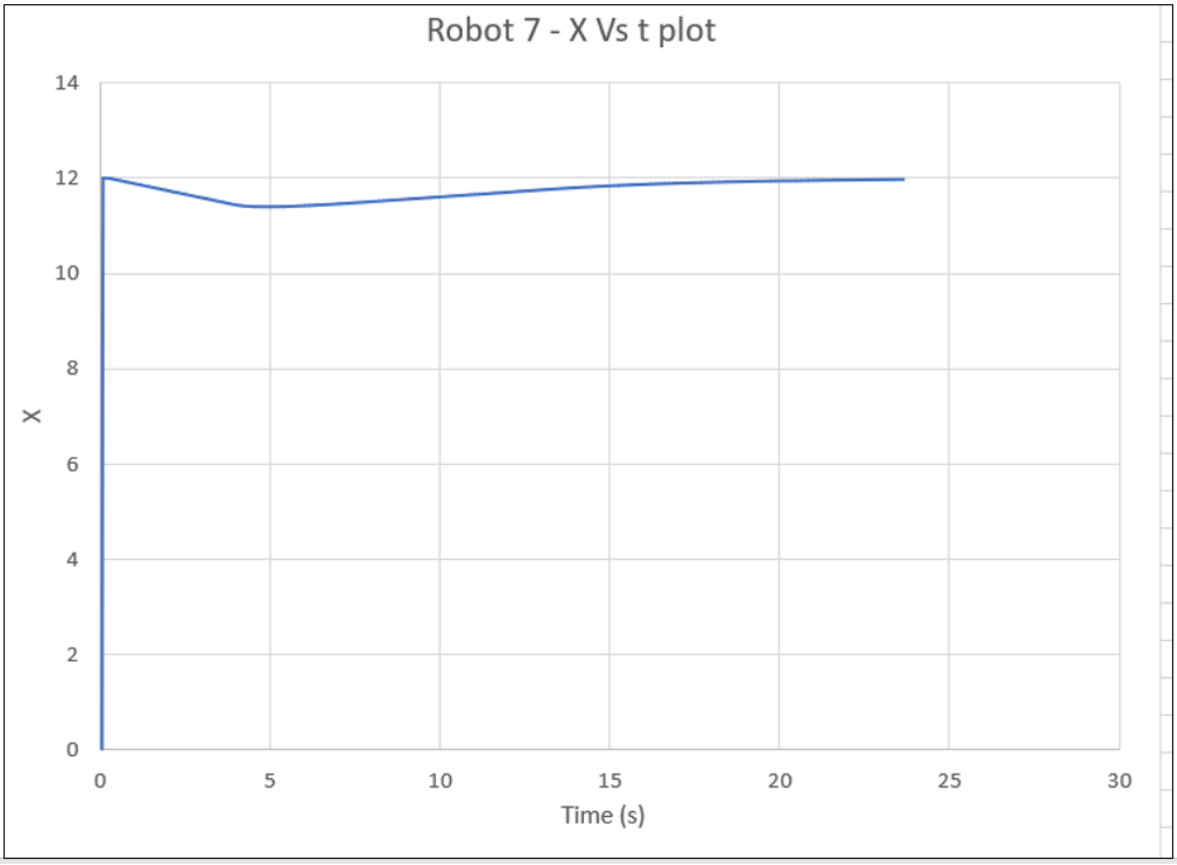

Robot 2 - X Vs t plot
2.5
2
1.5
1
0.5
0
X
0
5
10
15
20
25
30
Time (s)

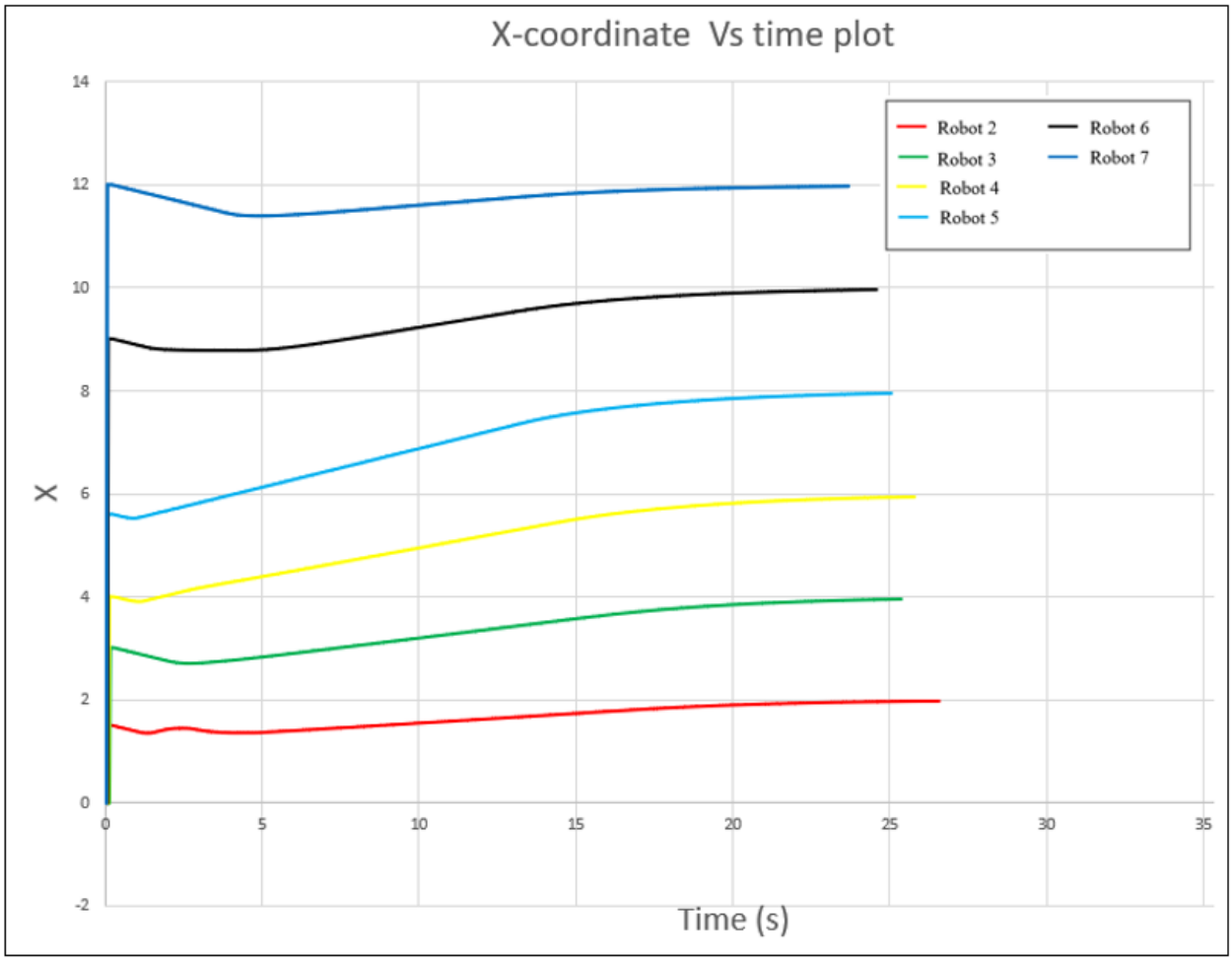

Robot 3 - X Vs t plot
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
X
0
5
10
15
20
25
30
Time (s)

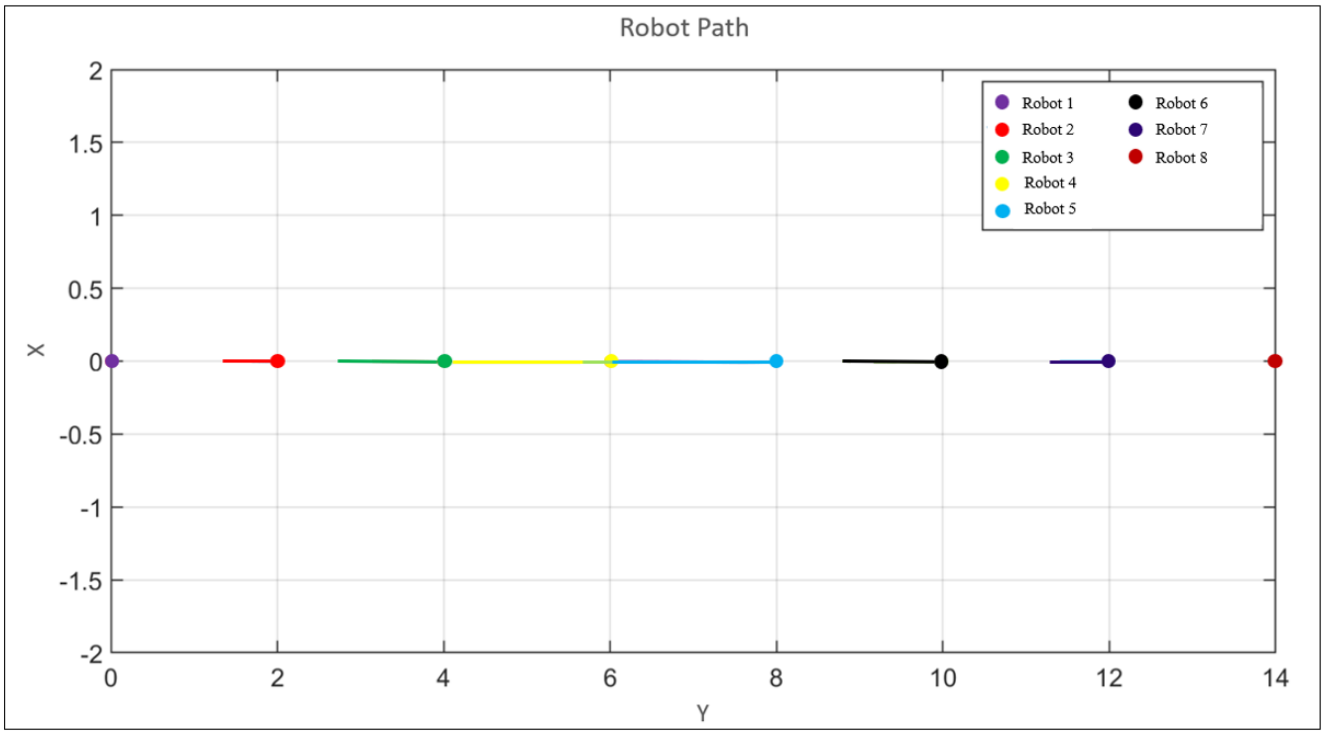

Robot 4 - X Vs t plot
X
Time (s)
7
6
5
4
3
2
1
0
-1
0
5
10
15
20
25
30


Robot 5 - X Vs t plot
X
Time (s)
9
8
7
6
5
4
3
2
1
0
0
5
10
15
20
25
30


Robot 6 - X Vs t plot
12
10
8
6
4
2
0
-2
X
0
5
10
15
20
25
30
Time (s)


Robot 7 - X Vs t plot
14
12
10
8
6
4
2
0
X
0
5
10
15
20
25
30
Time (s)


X-coordinate Vs time plot
Robot 2
Robot 3
Robot 4
Robot 5
Robot 6
Robot 7
X
Time (s)
14
12
10
8
6
4
2
0
-2
0
5
10
15
20
25
30
35


Robot Path
Robot 1
Robot 2
Robot 3
Robot 4
Robot 5
Robot 6
Robot 7
Robot 8
X
Y
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12
14