Proiect Practica 2020-2021

UNIVERSITATEA DIN CRAIOVA

Facultatea de Automatica, Calculatoare si Electronica

Specializarea : Automatica si Informatica Aplicata

Grupa : 2.3A

Practica de domeniu 2020-2021

Modelarea si simularea vehiculelor in Matlab/Simulink.

Membri echipei:

Penoiu Dennis-Gabriel

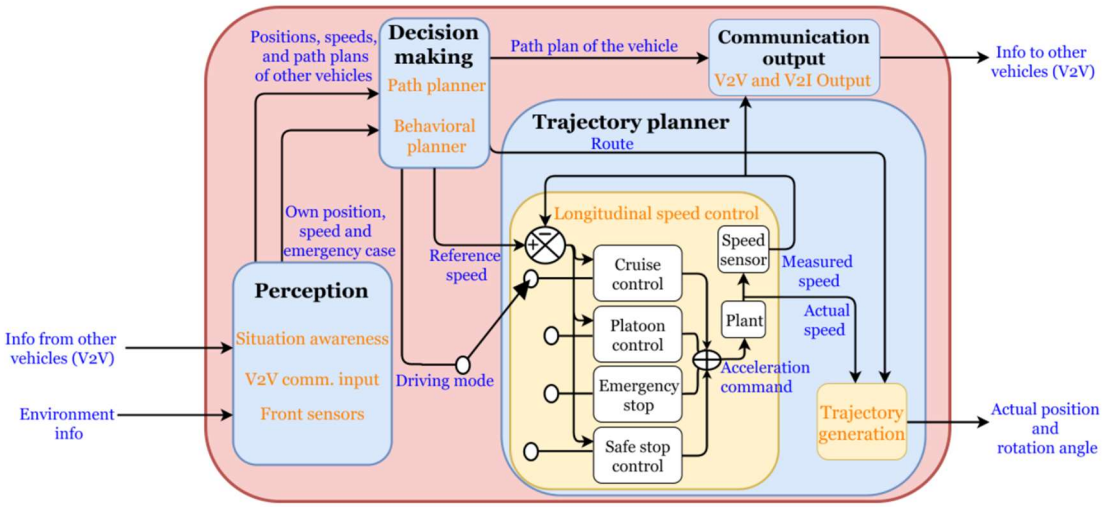
Robu Grigore Andrei

Coordonator,

Prof. Dr. Ing. Andreea Iacob

Craiova, 2021

Structura unui vehicul autonom în MOBATSim constă din diferite niveluri de abstractizare. Motivul acestei abstractizări este de a permite utilizatorului să acceseze și să personalizeze cu ușurință componentele de nivel scăzut, intrările / ieșirile acestora și algoritmii de decizie și control implementați prin interfețe clare între aceste componente și subsisteme prezentate în figură. Există patru interfețe principale; Percepție, luare de decizii, planificator de traiectorie și rezultate de comunicare.



Interfața de percepție

Este formată din trei componente principale; Senzori frontali, intrare de comunicare V2V și conștientizare a situației. Vehiculele percep mediul în două moduri, și anume prin senzorii frontali și componentele de comunicație V2V. Componenta de conștientizare a situației procesează datele percepute pentru a evalua cazul de urgență al vehiculului și situația relativă a acestuia în funcție de celelalte vehicule din jur.

Interfața Planificator de Traiectorie

Se compune dintr-un subsistem de control al vitezei longitudinale și o componentă de generare a traiectoriei. Controlul vitezei longitudinale este format din patru părți principale. Aceste părți funcționează în funcție de comanda modului de conducere creată de componenta planificator comportamental a interfeței de luare a deciziilor. Există în principal patru moduri de conducere disponibile: controlul vitezei de croazieră, controlul plutonului, controlul opririi în siguranță și oprirea de urgență.

Controlul vitezei de croazieră este modul în care nu există vehicul în față. În acest mod, vehiculul accelerează pentru a atinge viteza maximă. Atâta timp cât senzorul din față nu detectează niciun vehicul din față sau componenta V2I a vehiculului nu primește un mesaj de oprire înainte de o intersecție, vehiculul continuă să meargă la viteza maximă.

Când vehiculul trebuie să urmeze un vehicul mai lent, controlul plutonului preia pentru a genera comenzile de accelerație pentru a forma un pluton stabil. Există o serie de algoritmi care pot fi implementați pentru urmărirea unui vehicul și păstrarea unei distanțe de siguranță. În acest moment, MOBATSim oferă trei variante: spațiu constant prin PID, timp de avans constant prin PID și timp de

progres constant de MPC. Toți acești algoritmi de control pot fi reglați pentru a implementa vehicule mai agresive sau mai robuste după comportamente.

Controlul sigur al opririi începe în cazul în care vehiculul primește un mesaj de oprire și trebuie să se oprească înainte de o intersecție în cazul în care celelalte vehicule au dreptul de trecere pentru a traversa intersecția. Ajută vehiculele să încetinească încet înainte de un nod de oprire desemnat.

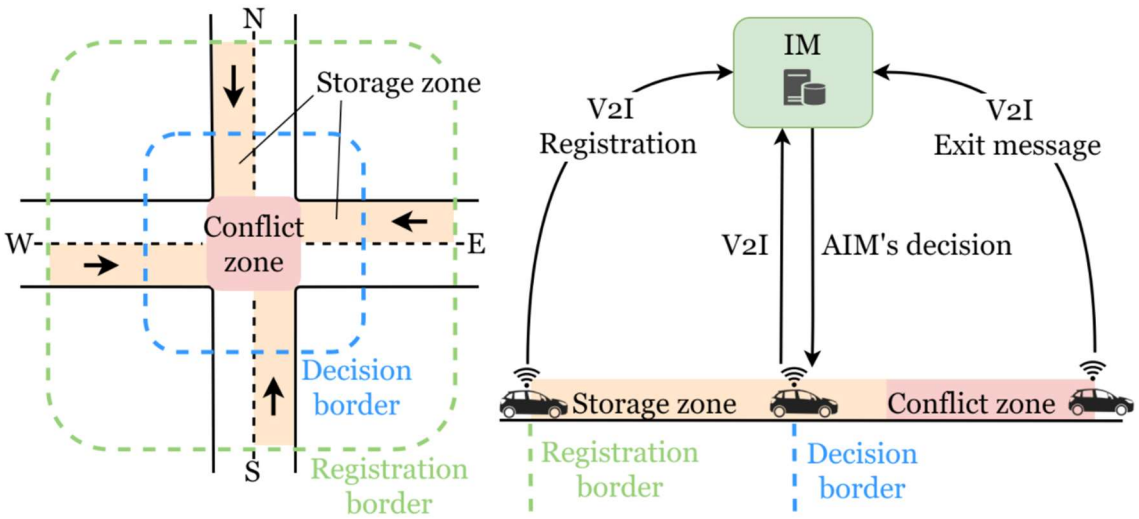
Oprirea de urgență folosește frâne complete atunci când vehiculul din față se oprește brusc sau un alt vehicul se oprește brusc. Apoi partea de oprire de urgență generează o comandă de decelerare constantă.

Traseul selectat și viteza longitudinală sunt luate ca intrări de componenta de generare a traiectoriei pentru a crea mișcările longitudinale și de rotație. În scopul simulării se creează un vector de poziție care constă dintr-o poziție a coordonatelor x-y-z și un unghi de rotație.

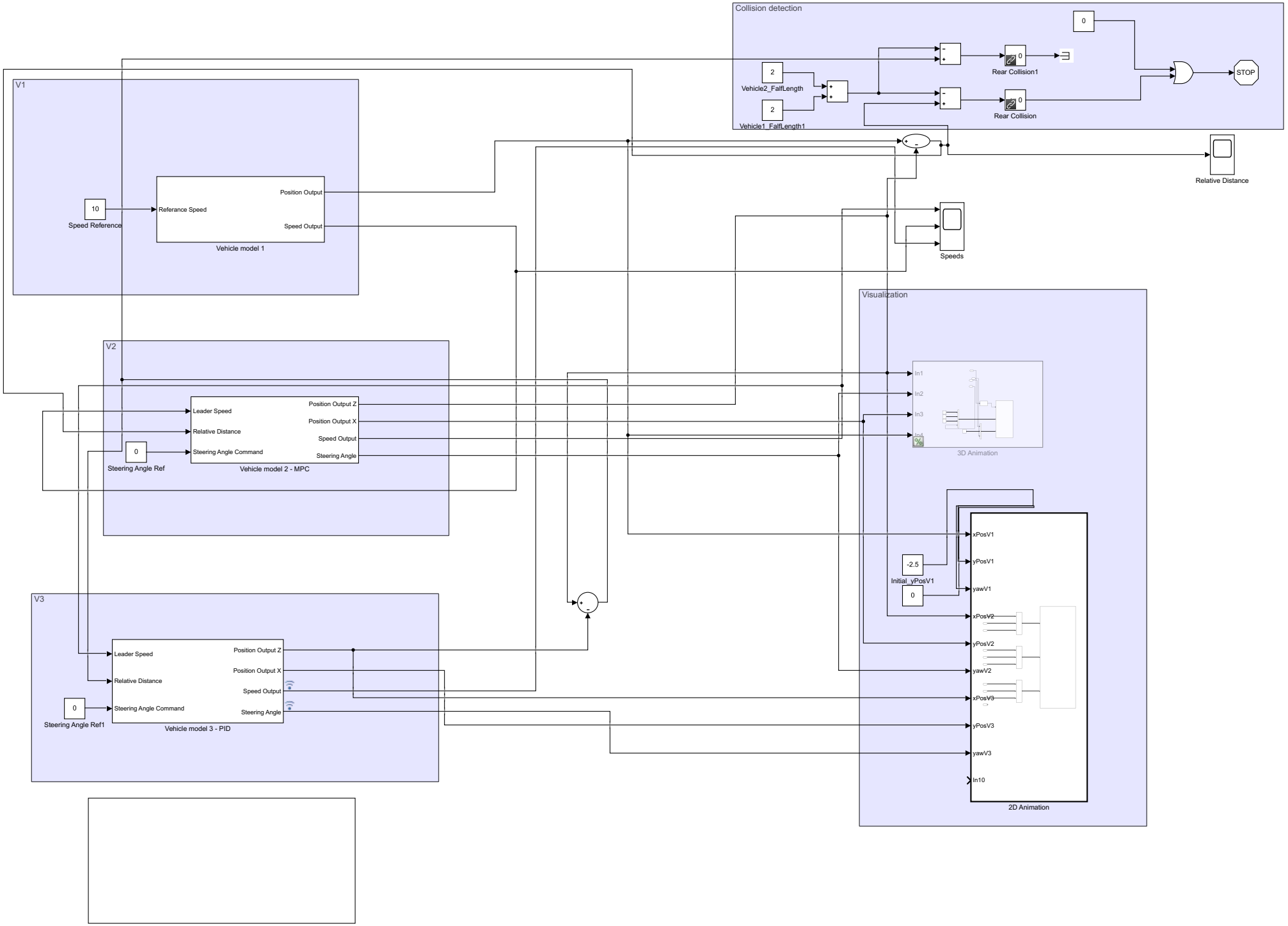
Interfață de ieșire de comunicare

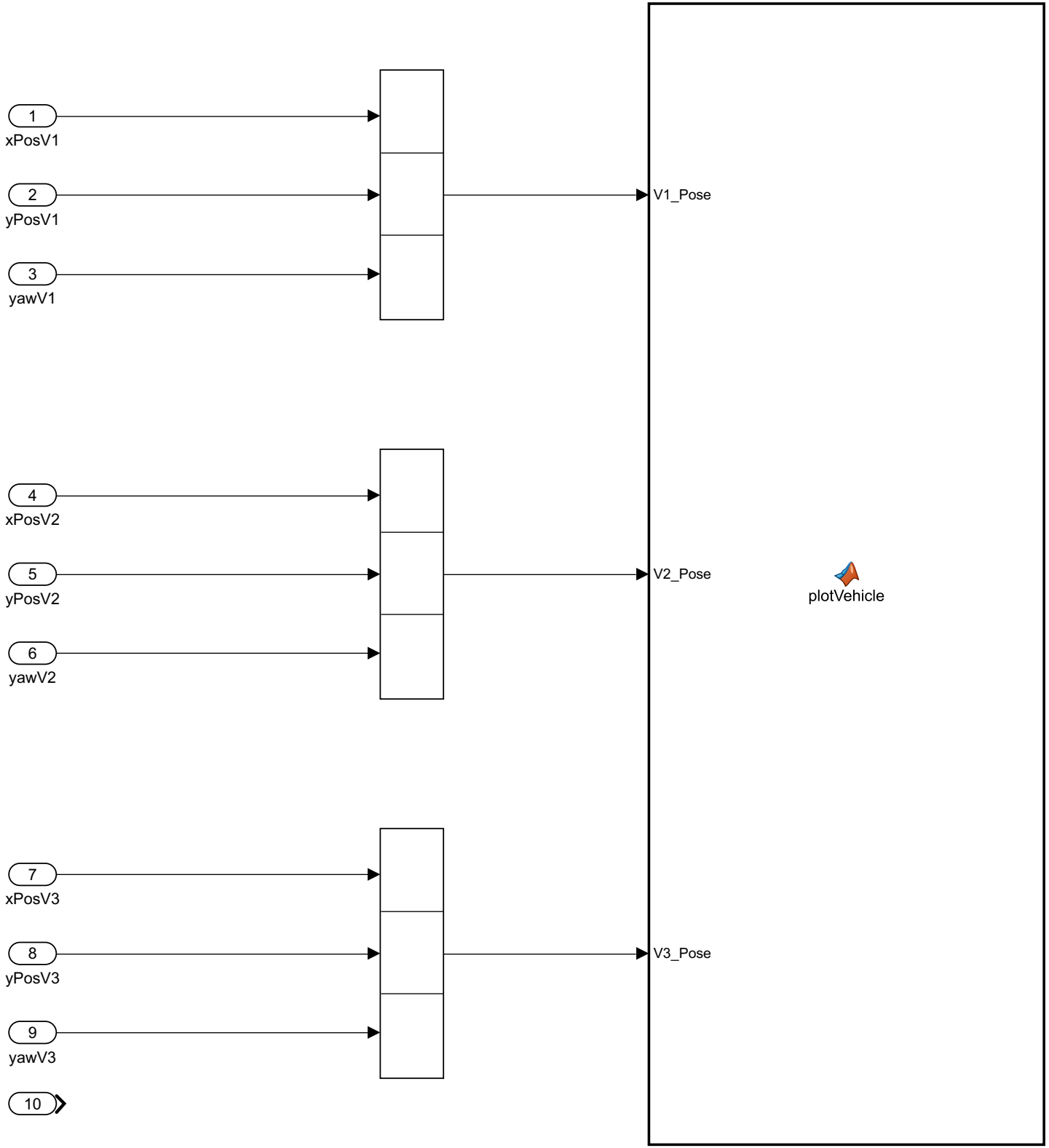
Valorile actuale ale planurilor de poziție, viteză și traseu ale vehiculului sunt apoi împachetate de componenta de ieșire de comunicație V2V și sunt partajate cu restul rețelei de vehicule conectate. Importanța partajării planurilor de parcurs ale vehiculelor este de a prevedea blocajele de trafic și de a alege o rută care să ajungă la nodul de destinație în cel mai scurt timp posibil.

În acest moment, componenta V2I este utilizată doar pentru a traversa intersecții. AIM (Gestionarea autonomă a intersecției) trimite și primește mesaje și folosește propriii algoritmi pentru a da dreptul de drum vehiculelor primite. O explicație mai detaliată a algoritmilor IM implementați va fi publicată ca o lucrare de conferință la IAV2019.



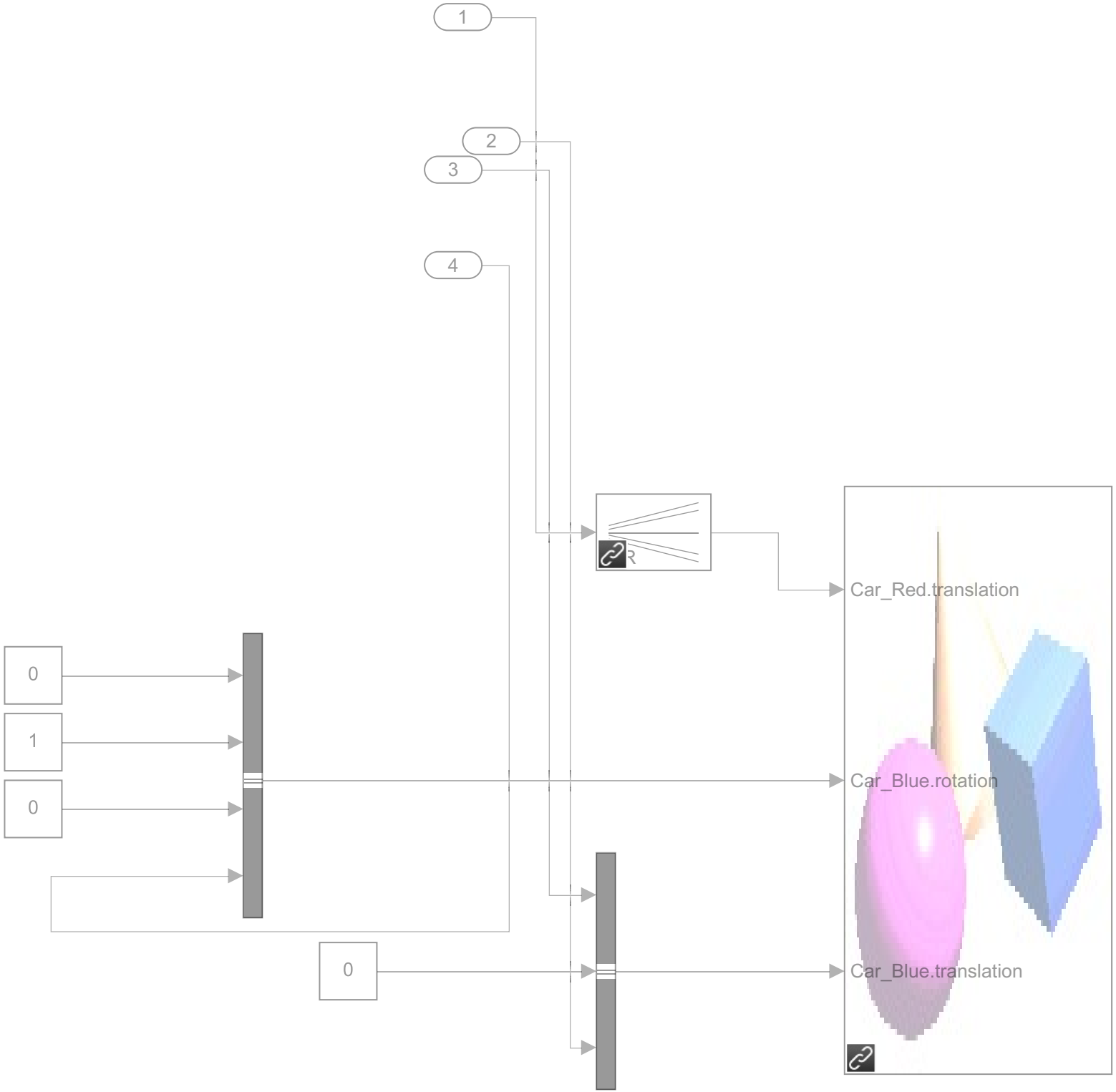
Proiect Practica AIA
Simulink vehicle




1
xPosV1
2
yPosV1
3
yawV1
4
xPosV2
5
yPosV2
6
yawV2
7
xPosV3
8
yPosV3
9
yawV3
10
V1_Pose
V2_Pose
V3_Pose
plotVehicle

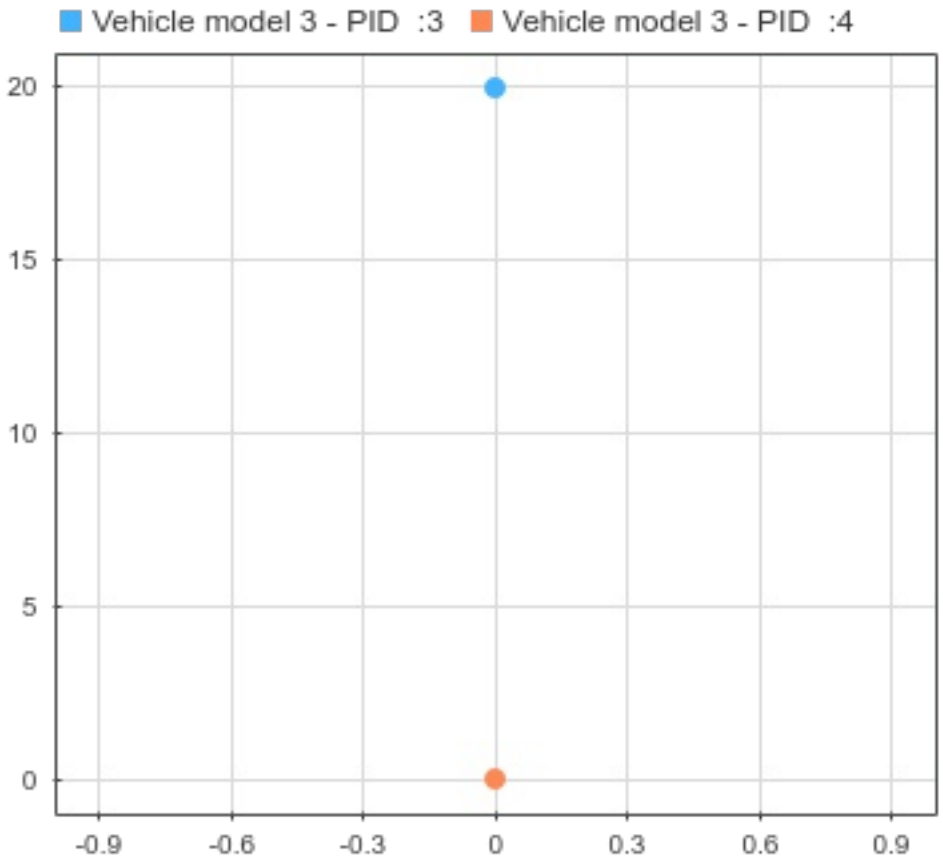
```
function plotVehicle(V1_Pose,V2_Pose,V3_Pose)

%%Clear previous Vehicle
cla;
%%Vehicle 1 - Pose
x1 = V1_Pose(1);
y1 = V1_Pose(2);
yaw1 = V1_Pose(3);

centerP = [x1;y1];
V1_HalfLengh = 4; %Length = 4m
V1_HalfWidth = 2; %Width = 2m

% Creating a rectangle
p1=[V1_HalfLengh; V1_HalfWidth];
p2=[V1_HalfLengh; -V1_HalfWidth];
p3=[-V1_HalfLengh; -V1_HalfWidth];
p4=[-V1_HalfLengh; V1_HalfWidth];
%Rotation Matrix
Rmatrix = [cos(yaw1) -sin(yaw1); sin(yaw1) cos(yaw1)];

%Rotated Points
p1r = centerP + Rmatrix*p1;
p2r = centerP + Rmatrix*p2;
p3r = centerP + Rmatrix*p3;
p4r = centerP + Rmatrix*p4;

%Connection points
Hitbox_V1 = [p1r p2r p3r p4r p1r];
cornersV1_x = transpose(Hitbox_V1(1,:));
cornersV1_y = transpose(Hitbox_V1(2,:));

%%Vehicle 2 - Pose
x2 = V2_Pose(1);
y2 = V2_Pose(2);
yaw2 = V2_Pose(3);

centerP = [x2;y2];
V2_HalfLengh = 4; %Length = 4m
V2_HalfWidth = 2; %Width = 2m
% Creating a rectangle
p1=[V2_HalfLengh; V2_HalfWidth];
p2=[V2_HalfLengh; -V2_HalfWidth];
p3=[-V2_HalfLengh; -V2_HalfWidth];
p4=[-V2_HalfLengh; V2_HalfWidth];

%Rotation Matrix
Rmatrix = [cos(yaw2) -sin(yaw2); sin(yaw2) cos(yaw2)];
%Rotated Points
p1r = centerP + Rmatrix*p1;
p2r = centerP + Rmatrix*p2;
p3r = centerP + Rmatrix*p3;
p4r = centerP + Rmatrix*p4;

%Connection points
Hitbox_V2 = [p1r p2r p3r p4r p1r];
cornersV2_x = transpose(Hitbox_V2(1,:));
cornersV2_y = transpose(Hitbox_V2(2,:));
```

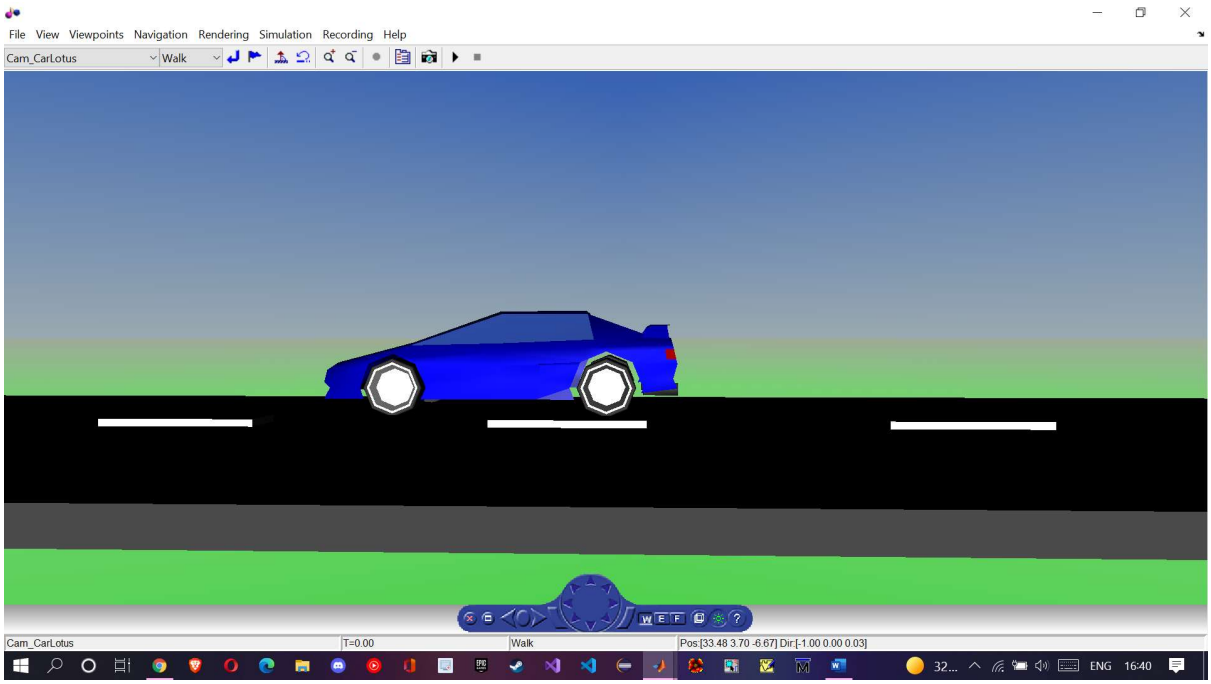
```
%%Vehicle 1 - Pose
x3 = V3_Pose(1);
y3 = V3_Pose(2);
yaw3 = V3_Pose(3);

centerP = [x3;y3];
V3_HalfLengh = 4; %Length = 4m
V3_HalfWidth = 2; %Width = 2m

% Creating a rectangle
p1=[V3_HalfLengh; V3_HalfWidth];
p2=[V3_HalfLengh; -V3_HalfWidth];
p3=[-V3_HalfLengh; -V3_HalfWidth];
p4=[-V3_HalfLengh; V3_HalfWidth];
%Rotation Matrix
Rmatrix = [cos(yaw3) -sin(yaw3); sin(yaw3) cos(yaw3)];

%Rotated Points
p1r = centerP + Rmatrix*p1;
p2r = centerP + Rmatrix*p2;
p3r = centerP + Rmatrix*p3;
p4r = centerP + Rmatrix*p4;

%Connection points
Hitbox_V3 = [p1r p2r p3r p4r p1r];
cornersV3_x = transpose(Hitbox_V3(1,:));
cornersV3_y = transpose(Hitbox_V3(2,:));

%% Plot Vehicle and the road
plot(cornersV1_x,cornersV1_y,'g');  %Vehicle 1 rectangle
plot(x1,y1,'*'); %Vehicle 1 center

plot(cornersV2_x,cornersV2_y,'r');  %Vehicle 2 rectangle
plot(x2,y2,'o'); %Vehicle 2 center

plot(cornersV3_x,cornersV3_y,'b');  %Vehicle 3 rectangle
plot(x3,y3,'x'); %Vehicle 3 center

upperLine = plot([0 1000],[5 5],'Color','blue');
midLine = plot([0 1000],[0 0],'--','Color','blue');
lowerLine = plot([0 1000],[-5 -5],'Color','blue');
%Adjust Axis
axis([x3-40 x3+40 y3-20 y3+20]); %Camera following V3 as ago vehicle
```

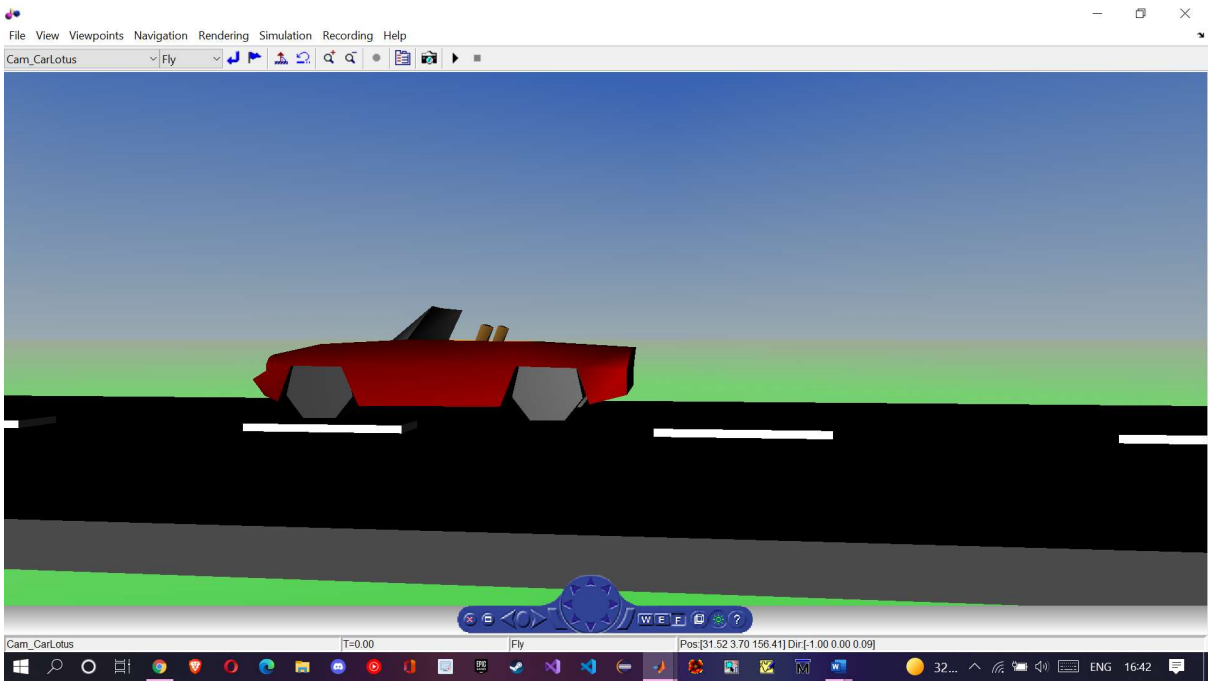

1
2
3
4
0
1
0
0
Car_Red.translation
Car_Blue.rotation
Car_Blue.translation

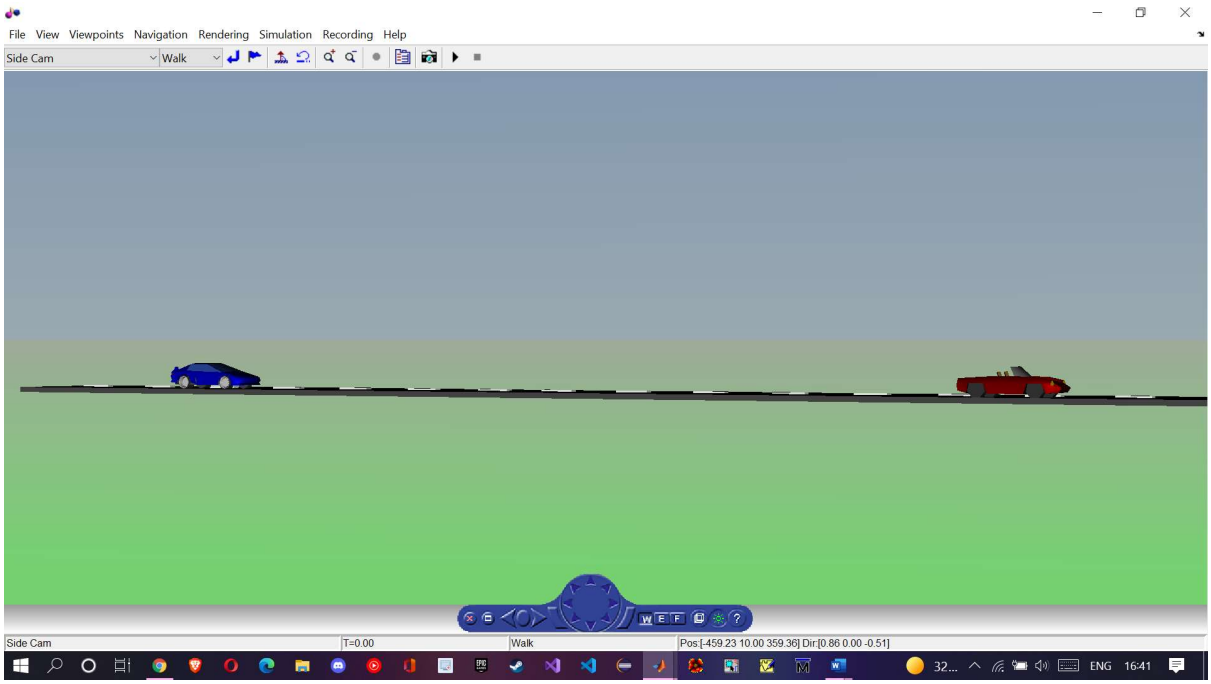
Vehicle model 3 - PID :3

Steering Angle Ref1:Value




File View Viewpoints Navigation Rendering Simulation Recording Help
Cam_CarLotus
Walk
Cam_CarLotus
T=0.00
Walk
Pos:[33.48 3.70 -6.67] Dir:[-1.00 0.00 0.03]
ENG
16:40


File View Viewpoints Navigation Rendering Simulation Recording Help
Cam_CarLotus
Fly
Cam_CarLotus
T=0.00
Fly
Pos:[31.52 3.70 156.41] Dir:[-1.00 0.00 0.09]
ENG
16:42


File View Viewpoints Navigation Rendering Simulation Recording Help
Side Cam
Walk
Side Cam
T=0.00
Walk
Pos:[-459.23 10.00 359.36] Dir:[0.86 0.00 -0.51]
ENG 16:41

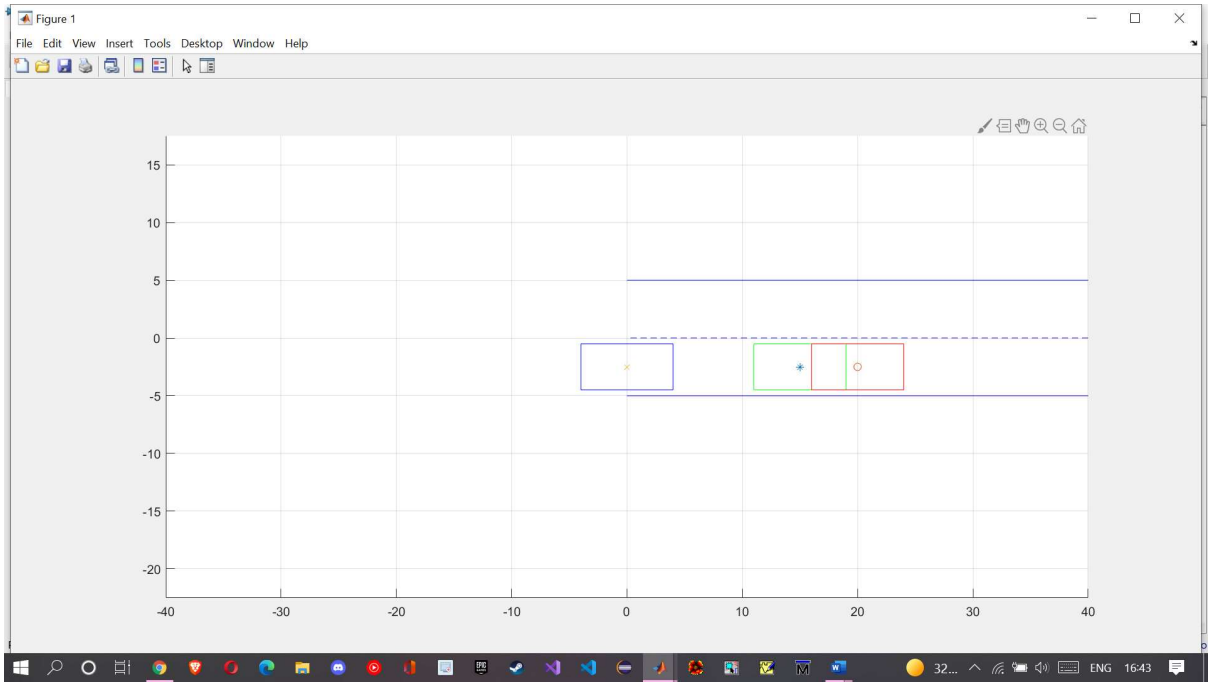


Figure 1
File Edit View Insert Tools Desktop Window Help
15
10
5
0
-5
-10
-15
-20
-40
-30
-20
-10
0
10
20
30
40
ENG 16:43