Project Report – Team TrafficLight

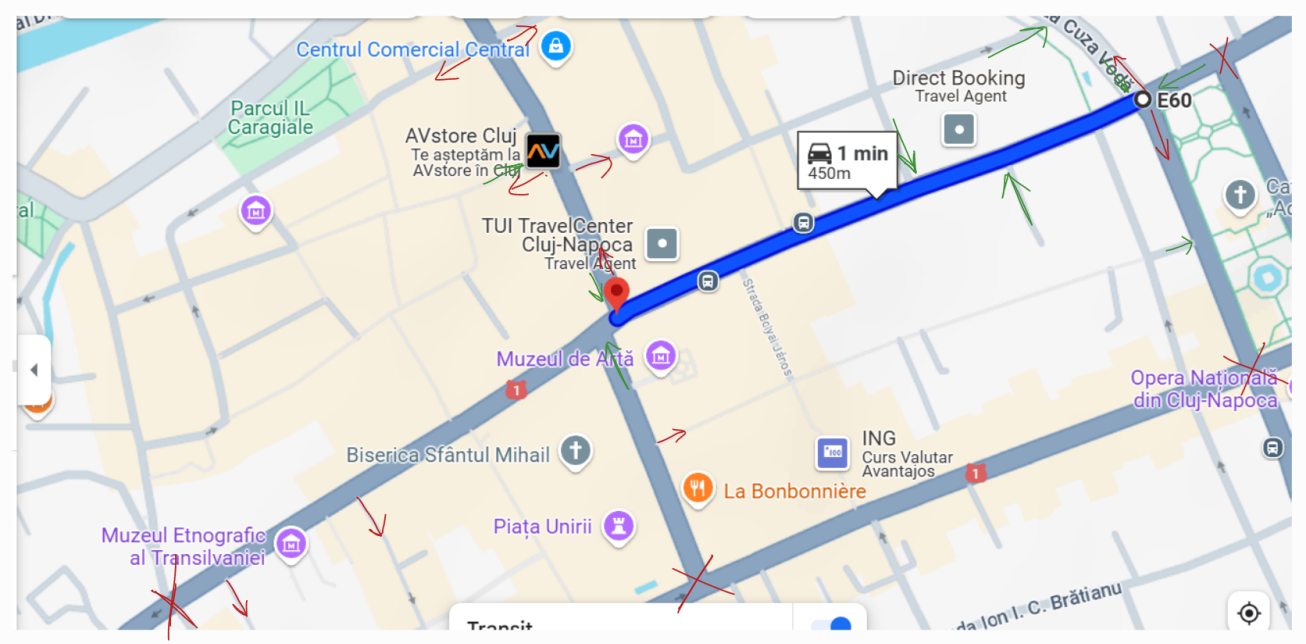
1. Project Requirements:

According to the map given to each team, develop a controller for each intersection (plant), that controller is a closed-loop one (with the in (1..n) input channels that is connected to its intersection’s output channels op(1..n) and an Intersections (with the OPs output channels). The controller must have dynamic delays feature to extend the time of the green light in case of a traffic jam (using asynchronous transitions in Lanes to send the signal and in Controller to receive it just like Project session 4).

The implemented lanes should include Bus lanes, Bus stations, and Taxi station (Project session 5) also the Priority cars should be able to cross the traffic light and not wait for the green light (this feature should be implemented in all traffic lights: the ones in the intersection and the pedestrian traffic lights.

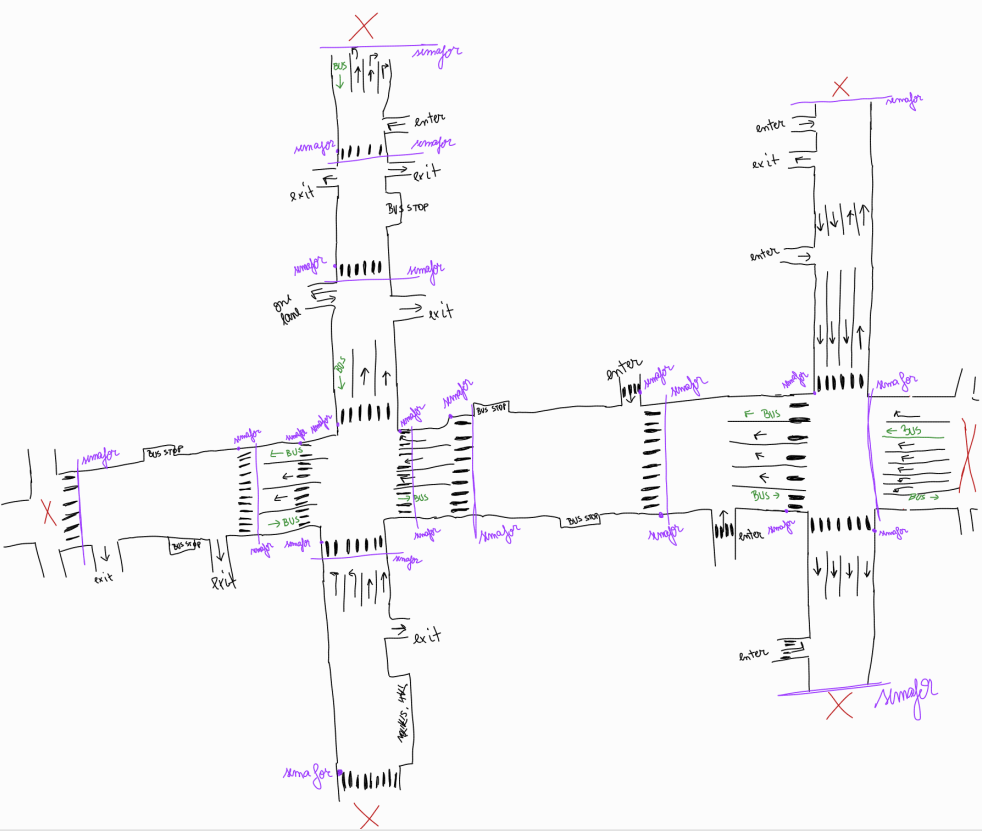
1. Specifications

2.1. Paste a screen shot of the entire given map

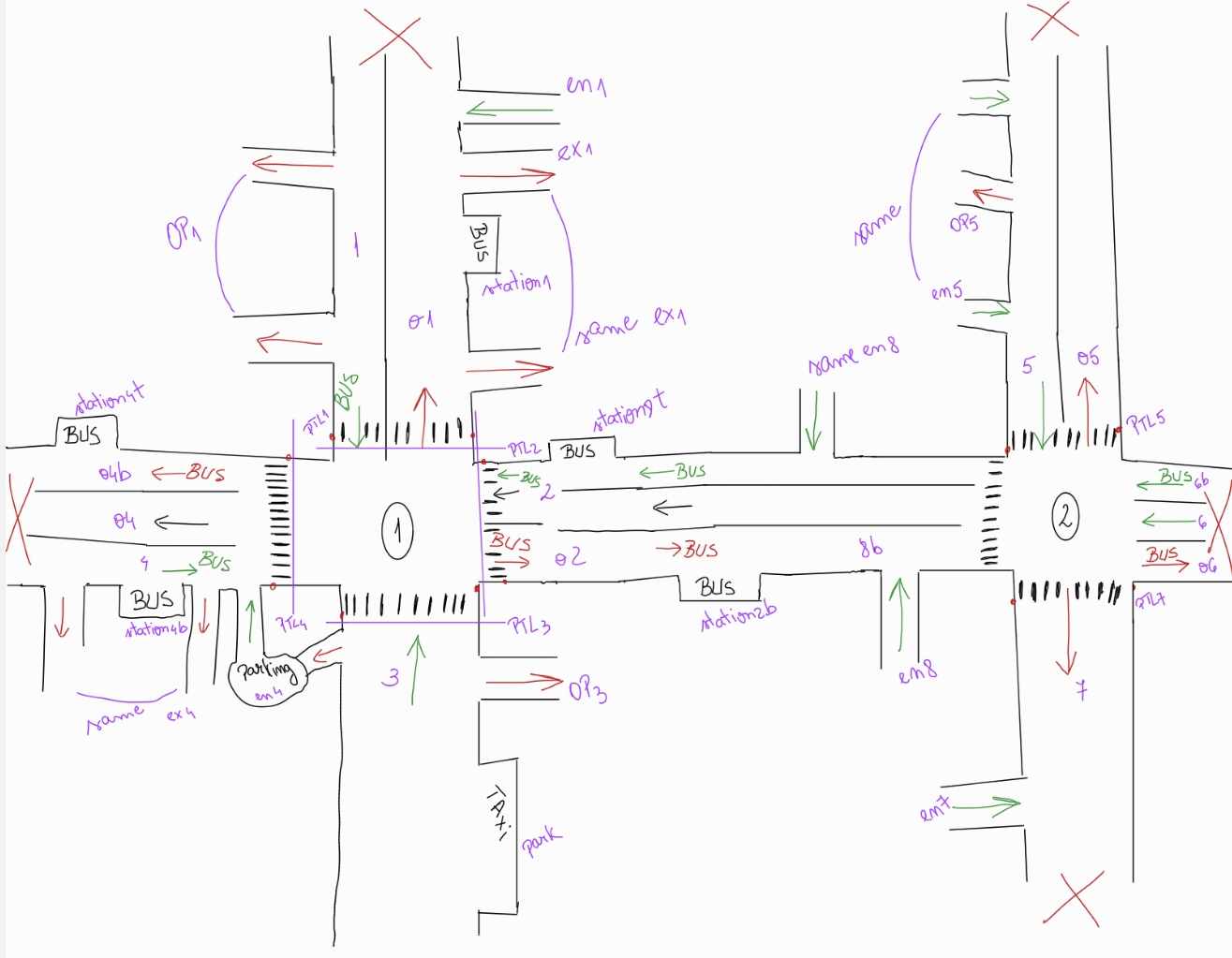


2.2. Draw a simplified one showing the intersections and the middle street that connects them, if the street has output and input lanes, they should be drawn and implemented at the end. The input and output lanes that are connected directly to the intersection should be fully implemented so as the middle street, if there exist exits and entrances, bus lanes, bus stations and taxi stations with pedestrian traffic lights if existed (their implementation should end until the nearest intersection or the roundabout that is not part of your given map).

a. More complex drawing



b. Simplified Version



2.3. Draw the component diagram (using UML drawing tool) for the entire system (depending on your implementation, each OETPN is considered a component) and show the names of the input and output channels.

A diagram of a computer network

Description automatically generated

1. Design

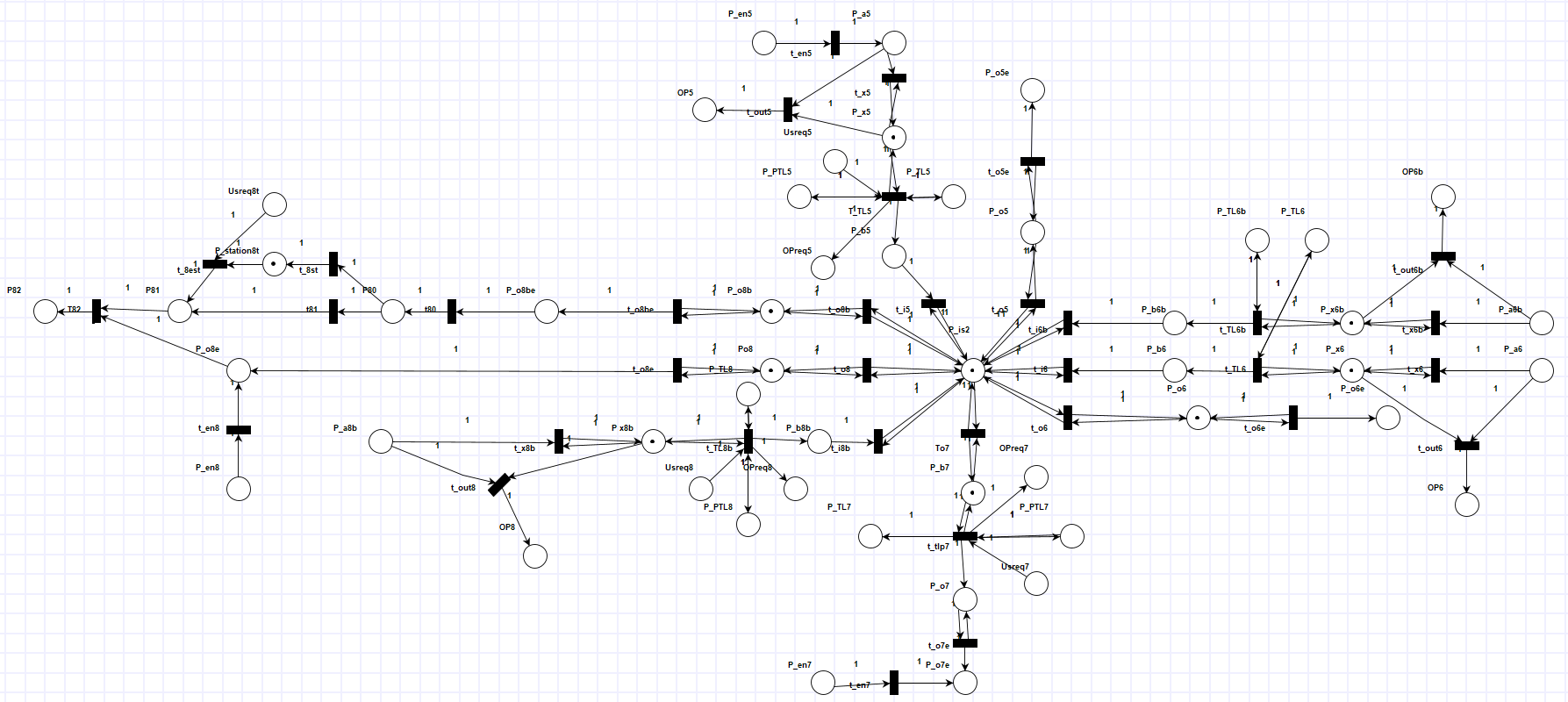
3.1. Draw the OETPN model for the Plant (the intersections, input and output lanes, and the middle street).

a. Intersection 1 (Pin 1)

A diagram of a network

Description automatically generated with medium confidence

b. Intersection 2 (Pin 2)



3.2. Write the Place types, grd&map for the entire map structure OETPN (group the similar transitions together).

a. Intersection 1

* Place Types:
  + - * + Input Places

DataCar: P\_a1b, P\_b1b, P\_a2, P\_b2, P\_a3, P\_b3, P\_u4, P\_a4b

DataCarQueue: P\_x1b, P\_x2, P\_park, P\_x3, P40, P\_station4b, P41, P\_x4b, P\_b4b

DataString: green, full, P\_TL1, UserReq1, P\_PTL1, P\_TL2, UserReq2, P\_PTL2, P\_TL3, UserReq3, P\_PTL3, P\_TL4, P\_PTL4, UserReq4b, UserReq4

DataTransfer: OPReq1, OP1, OPReq2, OP2, OPReq3, OP3, OPReq4, OP4

* + - * + Output Places

DataCar: P\_o1e, P\_en1, P\_ex1, P\_o2e, P\_en4, P\_ex4, P\_o4be, P44,

DataCarQueue: P\_o1, P10, P\_station1, P11, P12, P\_o2, P20, P\_station2b, P21, P\_o4, P\_o4e, P\_o4b, P42, P\_station4t, P43

DataString: UserReq1s, UserReq2b, UserReq4t

DataTransfer: P22,

* + - * + P\_Is1 : DataCarQueue
* Grd&map

T\_x1b: P\_a1b != Null && P\_x1b CanAddCars && P\_a1b IsBus || P\_a1b IsPriorityCar

P\_x1b.AddElement(P\_a1b)

T\_out1: P\_a1b != Null && P\_x1b CanNotAddCars

OP1.SendOverNetwork(“full”)

T\_TL1b: P\_TL = green && P\_x1b HaveCar && P\_PTL1 = P\_PTL1 && UserReq1 != Null

P\_b1b.PopElementWithoutTarget(P\_x1b)

P\_TL1.Move(P\_TL1)

P\_PTL1.Move(P\_PTL1)

OPReq1.SendOverNetwork(UserReq1)

T\_i1b: P\_b1b != Null && P\_Is1 CanAddCars

P\_Is1.AddElement(P\_b1b)

T\_o1: P\_Is1 HaveCarForMe && P\_o1 CanAddCars

P\_o1.PopElementWithTargetToQueue(P\_Is1)

T\_o1e: P\_o1 HaveCar

P\_o1e.AddElement(P\_o1)

T10: P\_o1e != Null && P\_o1e IsBus || P\_01e IsPriorityCar

P10.AddElement(P\_o1e)

T11: P10 != Null

P11.AddElement(P10)

T\_1s: P10 HaveBus

P\_station1.PopElementWithourTargetToQueue(P10)

T\_1es: P\_station1 HaveCar && UserReq1 != Null

P11.PopElementWithoutTargetToQueue(P11)

T12: P11.HaveCarForMe

P12.PopElementWithTargetToQueue(P12)

T\_en1: P\_en1 != Null && P12 CadAddCars

P12.AddElement(P\_en1)

T\_ex1: P12 HaveCar

P\_ex1.PopElementWithoutTargetToQueue(P12)

T\_out2: same t\_out1

T\_x2: same t\_x1b

T\_TL2: same T\_TL1b

T\_i2: same t\_i1b

T\_o2: same t\_o1

T\_o2e: same t\_o1e

T\_20: same t\_10

T\_21: same t\_11

T\_2sb: same t\_1s

T\_2esb: same t\_1es

T22: P\_21 HaveCarForMe

P22.SendOverNetwork(P21)

T\_i3: same t\_i1b

T\_TL3: same t\_Tl1b

T\_x3: same t\_x1b

T\_out3: same t\_out1

T\_park: P\_a3 != Null && P\_park CanAddCars

P\_park.AddElement(P\_a3)

T\_i4b: same t\_i1b

T\_TL4: same t\_TL1b

T\_x4: same t\_x1b

T\_out4: same t\_out1

T\_a4b: P41 HaveCar

P\_a4b.PopElementWithoutTarget(P41)

T\_o4: same t\_o1

T\_o4e: same t\_1oe

T\_en4: same t\_en1

T\_ex4: same t\_ex1

T\_o4b: same t\_o1

T\_o4be: same t\_1oe

T\_43: same t\_21

T\_4st: same t\_2st

T\_4est: same t\_4est

T\_44: t12

a. Intersection 2

* Place Types:
  + - * + Input Places:

DataCar: P\_en5, P\_a5, P\_b5, P\_a6b, P\_b6b, P\_a6, P\_b6, P\_a8b, P\_b8b

DataCarQueue: P\_x5, P\_x6b, P\_x6, P\_x8b

DataString: green, full, P\_TL5, P\_PTL5, UserReq5, P\_TL6b, P\_TL6, P\_LTL8, P\_PTL8, UserReq8

DataTransfer: OPReq5, OP5, OP6b, OP6, OPReq8, OP8

* + - * + Output Places:

DataCar: P\_o5e, P\_o6e, P\_o7, P\_en7, P\_o8be, P\_en8

DataCarQueue: P\_o5, P\_o6, P\_o7e, P\_b7, P\_o8b, P80, P\_station8t, P81, P\_o8, P\_o8e,

DataString: P\_TL7, P\_PTL7, UserReq7, UserReq8t

DataTransfer: OPReq7, P82

* + - * + P\_Is2: DataCarQueue
* Grd&map

T\_x5: same t\_x1b

T\_out5: same t\_out1

T\_TL5: same t\_TL1b

T\_i5: same t\_i1b

T\_o5: same t\_o1

T\_o5e: same t\_o1e

T\_en5: same t\_en1

T\_06: same t\_o4

T\_o6e: same t\_o4e

T\_i6:same t\_i2

T\_x6: same t\_x1b

T\_out6: same t\_out1

T\_TL6: P\_TL6 = green && P\_x6 HaveCar

P\_b6.PopElementWithoutTarget(P\_x6)

P\_TL6.Move(P\_TL6)

T\_o7: same t\_o6

T\_TL7: same T\_TL5

T\_en7: same T\_en5

T\_o7e: P\_o7 HaveCar

P\_o7e.PopElementWithoutTarget(P\_o7)

T\_out8:same t\_out1b

T\_x8b: same t\_x1b

T\_TL8b: same t\_TL1b

T\_i8b: same t\_i1

T\_o8: same t\_o4

T\_o8e: same t\_o4e

T\_en8: same t\_en5

T\_o8b: same t\_o4b

T\_o8e: same t\_o4be

T\_80: same t\_42

T\_81: same t\_43

T\_8st: same t\_4st

T\_8est: same t\_4est

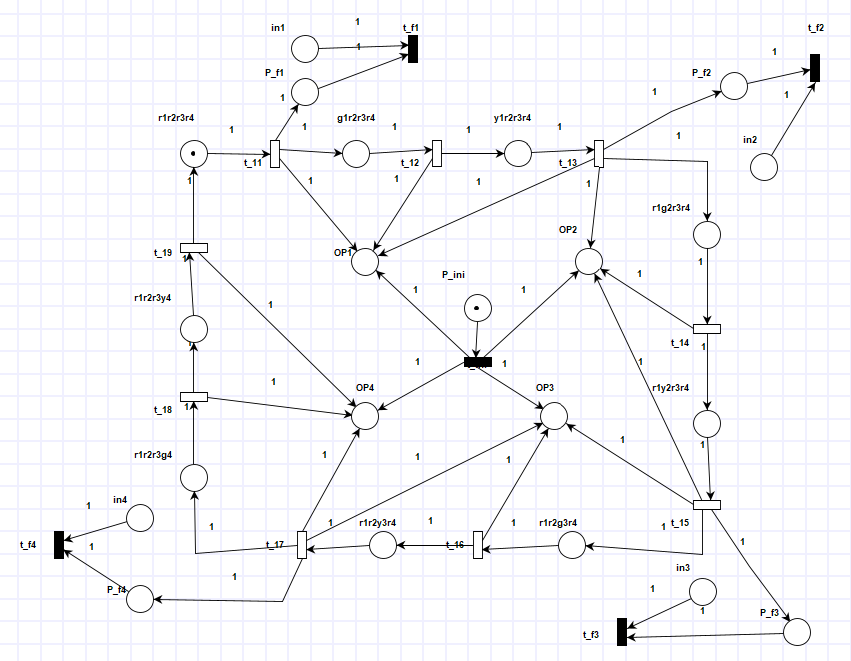
T\_82: P82 CanAddCars && P81 HaveCar || P\_o8e HaveCar

P82.SendOverNetwork(P81)

P82. SendOverNetwork(P\_o8e)

3.3. Draw the OETPN model for the controllers for the intersections and one pedestrian traffic light if existed.

a. Controllers – they are the same, so only one was drawn



b. ControllerPedestrians for one pedestrians crossing.

A diagram of a network

Description automatically generated

* 1. Write the Place types, grd&map for all the controllers OETPNs (group the similar transitions together).

a.Controller 1

* Place Types:

DataString: ini, red, green, yellow, r1r2r3r4, g1r2r3r4, y1r2r3r4, r1g2r3r4, r1y2r3r4, r1r2g3r4, r1r2y3r4, r1r2r3g4, r1r2r3y4, in1, in2, in3, in4, pf\_1, pf\_2, pf\_3, pf\_4

DataInteger: Five, Ten

DataTransfer: OP1, OP2, OP3, OP4

* Grd&map:

iniT: ini != Null

OP1.SendOverNetwork(ini)

OP2.SendOverNetwork(ini)

OP3.SendOverNetwork(ini)

OP4.SendOverNetwork(ini)

T\_11: r1r2r3r4 != Null

g1r2r3r4.Move(r1r2r3r4)

OP1.SendOverNetwork(green)

p\_f1.Move(r1r2r3r4)

T\_12: g1r2r3r4 != Null

y1r2r3r4.Move(g1r2r3r4)

OP1.SendOverNetwork(yellow)

T\_13: y1r2r3r4 != Null

r1g2r3r4.Move(y1r2r3r4)

OP1.SendOverNetwork(red)

OP2.SendOverNetwork(green)

p\_f2.Move(y1r2r3r4)

T\_14: r1g2r3r4 != Null

r1y2r3r4.Move(r1g2r3r4)

OP2.SendOverNetwork(yellow)

T\_15: r1y2r3r4 != Null

r1r2g3r4.Move(r1y2r3r4)

OP2.SendverNetwork(red)

OP3.SendOverNetwork(green)

p\_f3.Move(r1y2r3r4)

T\_16: r1r2g3r4 != Null

r1r2y3r4.Move(r1r2g3r4)

OP3.SendOverNetwork(yellow)

T\_17: r1r2y3r4 != Null

r1r2r3g4.Move(r1r2y3r4)

OP4.SendOverNetwork(green)

OP3.SendOverNetwork(red)

p\_f4.Move(r1r2y3r4)

T\_18: r1r2r3g4 != Null

r1r2r3y4.Move(r1r2r3g4)

OP4.SendOverNetwork(yellow)

T\_19: r1r2r3y4 != Null

r1r2r3r4.Move(r1r2r3y4)

OP4.SendOverNetwork(red)

T\_f1: p\_f1 != Null && in1 == Null || p\_f1 != Null && in1 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f2: p\_f2 != Null && in2 == Null || p\_f2 != Null && in2 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f3: p\_f3 != Null && in3 == Null || p\_f3 != Null && in3 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f4: p\_f4 != Null && in4 == Null || p\_f4 != Null && in4 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

b.Controller 2

* Place Types:

DataString: ini, red, green, yellow, r5r6r6br8, g5r6r6br8, y5r6r6br8, r5g6r6br8, r5y6r6br8, r5r6g6br8, r5r6y6br8, r5r6r6bg8, r5r6r6by8, in5, in6, in6b, in8, pf\_5, pf\_6, pf\_6b, pf\_8

DataInteger: Five, Ten

DataTransfer: OP5, OP6, OP6b, OP8

* Grd&map:

iniT: ini != Null

OP5.SendOverNetwork(ini)

OP6b.SendOverNetwork(ini)

OP6.SendOverNetwork(ini)

OP8.SendOverNetwork(ini)

T\_21: r5r6r6br8 != Null

g5r6r6br8.Move(r5r6r6br8)

OP5.SendOverNetwork(green)

p\_f5.Move(r5r6r6br8)

T\_22: g5r6r6br8 != Null

y5r6r6br8.Move(g5r6r6br8)

OP5.SendOverNetwork(yellow)

T\_23: y5r6r6br8 != Null

r5g6r6br8.Move(y5r6r6br8)

OP5.SendOverNetwork(red)

OP6.SendOverNetwork(green)

p\_f6.Move(y5r6r6br8)

T\_24: r5g6r6br8 != Null

r5y6r6br8.Move(r5g6r6br8)

OP6.SendOverNetwork(yellow)

T\_25: r5y6r6br8 != Null

r5r6g6br8.Move(r5y6r6br8)

OP6.SendverNetwork(red)

OP7.SendOverNetwork(green)

p\_f7.Move(r5y6r6br8)

T\_26: r5r6g6br8 != Null

r5r6y6br8.Move(r5r6g6br8)

OP7.SendOverNetwork(yellow)

T\_27: r5r6y6br8 != Null

r5r6r6bg8.Move(r5r6y6br8)

OP8.SendOverNetwork(green)

OP7.SendOverNetwork(red)

p\_f8.Move(r5r6y6br8)

T\_28: r5r6r6bg8 != Null

r5r6r6by8.Move(r5r6r6bg8)

OP8.SendOverNetwork(yellow)

T\_29: r5r6r6by8 != Null

r5r6r6br8.Move(r5r6rby8)

OP8.SendOverNetwork(red)

T\_f5: p\_f5 != Null && in5 == Null || p\_f5 != Null && in5 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f6: p\_f6 != Null && in6 == Null || p\_f6 != Null && in6 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f6b: p\_f6b != Null && in6b == Null || p\_f6b != Null && in6b != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

T\_f8: p\_f8 != Null && in8 == Null || p\_f8 != Null && in8 != Null

“”.DynamicDelay(Five)

“”DynamicDelay(Ten)

c.ControllerPedestrians

* Place Types:

DataString: ini, UserReq1, UserReq2, UserReq3, UserReq4, UserReq5, UserReq6, UserReq7, UserReq8, red, green, yellow, yr1, rg1, ry1, gr1, yr2, rg2, gr2, yr3, rg3, ry3, yr4, rg4, rg4, ry4, gr4, yr5, rg5, ry5, gr7, rg7, gr7, yr8, rg8, ry8, gr8, wait1, wait2, wait3, wait4, wait5, wait6, wait7, wait8

DataTransfer: P\_TL1, P\_TL2, P\_TL3, P\_TL4, P\_TL5, P\_TL6, P\_TL7, P\_TL8, P\_PTL1, P\_PTL2, P\_PTL3, P\_PTL4, P\_PTL5, P\_PTL6, P\_PTL7, P\_PTL8

* Grd&map:

IniT: ini != Null

P\_PTL1.SendOverNetwork(red)

P\_PTL2.SendOverNetwork(red)

P\_PTL3.SendOverNetwork(red)

P\_PTL4.SendOverNetwork(red)

P\_PTL5.SendOverNetwork(red)

P\_PTL6.SendOverNetwork(red)

P\_PTL7.SendOverNetwork(red)

P\_PTL8.SendOverNetwork(red)

t11: UserReq1 != Null && wait1 != Null

yr1.Move(wait1)

P\_TL1.SendOverNetwork(yellow)

t12: yr1 != Null

rg1.Move(yr)

P\_TL1.SendOverNetwork(red)

P\_PTL1.SendOverNetwork(green)

t13: rg1 != Null

ry1.Move(rg1)

P\_PTL1.SendOverNetwork(yellow)

t14: ry1 != Null

gr1.Move(ry1)

P\_TL1.SendOverNetwork(green)

P\_PTL1.SendOverNetwork(red)

t15: gr1 != Null

wait1.Move(gr1)

t21: same t11

t22: same t12

t23: same t13

t24: same t14

t25: same t15

t31: same t11

t32: same t12

t33: same t13

t34: same t14

t35: same t15

t41: same t11

t42: same t12

t43: same t13

t44: same t14

t45: same t15

t51: same t11

t52: same t12

t53: same t13

t54: same t14

t55: same t15

t61: same t11

t62: same t12

t63: same t13

t64: same t14

t65: same t15

t71: same t11

t72: same t12

t73: same t13

t74: same t14

t75: same t15

t81: same t11

t82: same t12

t83: same t13

t84: same t14

t85: same t15

1. Implementation

Repository link: <https://github.com/DCS-Lab-and-Project/final-project-andreea-ciubotaru.git>

1. Testing

5.1.

Send a Priority car from the 1st intersection, that should go through the middle street and exit from one of the exit lanes from the 2nd intersection without stopping at the red lights and if there is a bus lane, show that it can cross there as well. Attach screen shots showing how the car moves and at the end of the test, pause the intersection OETPN and click on the save log button, save it as test1\_intersection 1.txt and test1\_intersection 2.txt if you have implemented them in two separate OETPNs. Then add the text file/s to the repository.

5.2.

Traffic jam: for each intersection, create a traffic jam case by sending the maximum number of cars to the input lane of the intersection, start the controller, then send the last car. The controller should receive a signal from the plant (intersection) and the transition that is responsible for sending a yellow light to that lane where you input the cars to, should have changed the delay to 10 sec (it will be shown in the execution list) and it should return back to 5 sec when there is no signal in the in channel. Take screen shots of the execution and then pause the controller OETPN and click on the save log button, save it as test2.txt and add the text file to the repository.

5.3. In case you have bus lanes, bus stations, taxi stations do a test for them as well similar to project session 5.