Project Report – Team TrafficLight

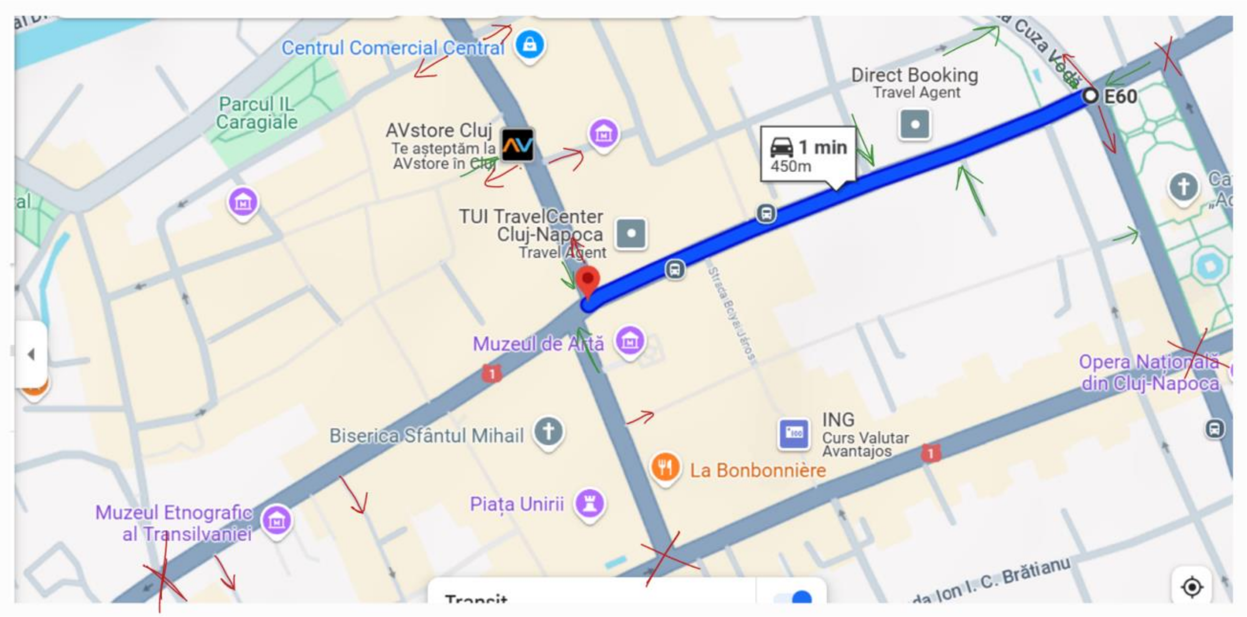
1. Project Requirements:

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.

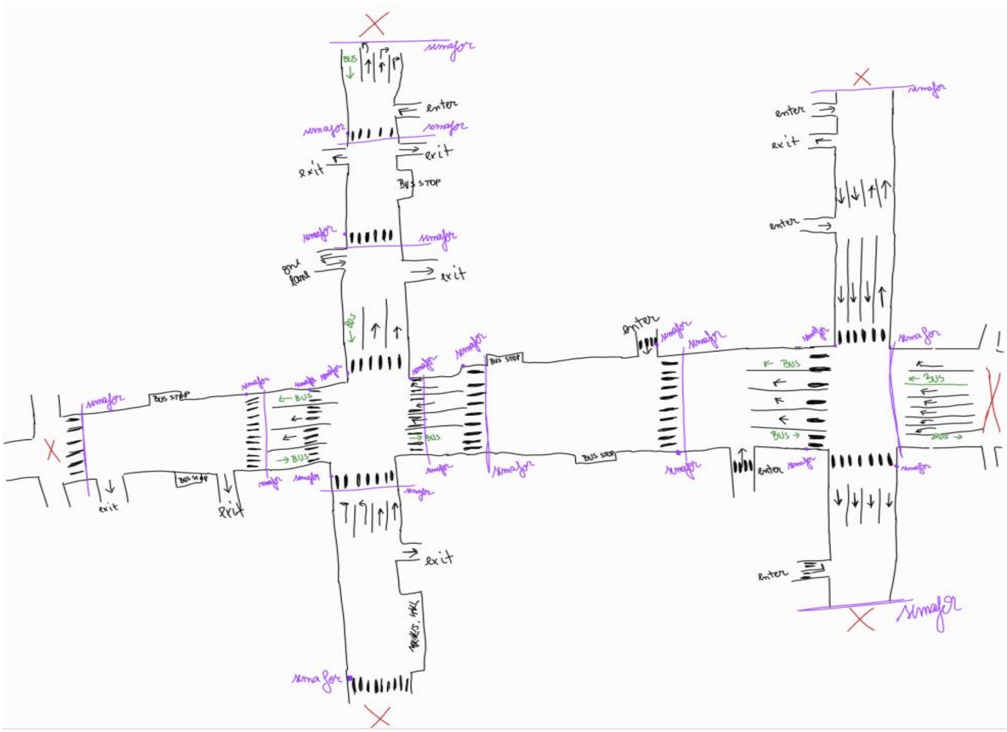
2. Specifications

2.1. Paste a screen shot of the entire given map

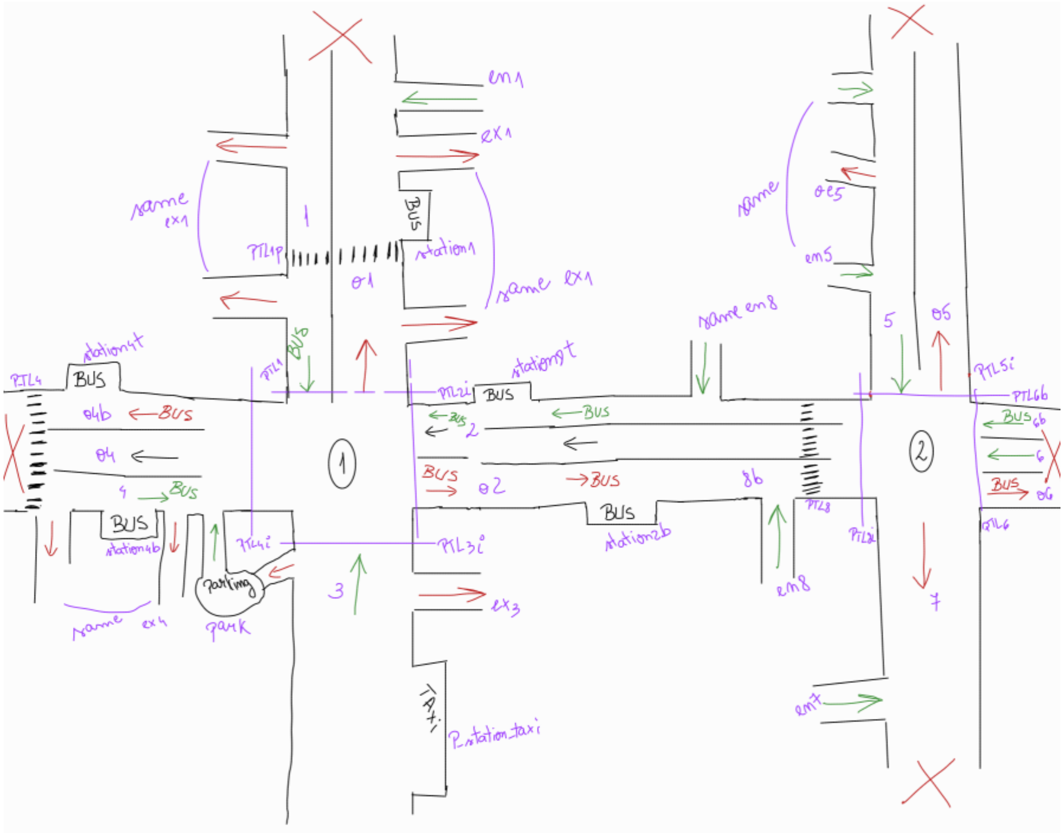


2.2. Draw a simplified one showing the intersections and the middle street that connects them, if the street has output and input lanes, they should be drawn and implemented at the end. The input and output lanes that are connected directly to the intersection should be fully implemented so as the middle 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).

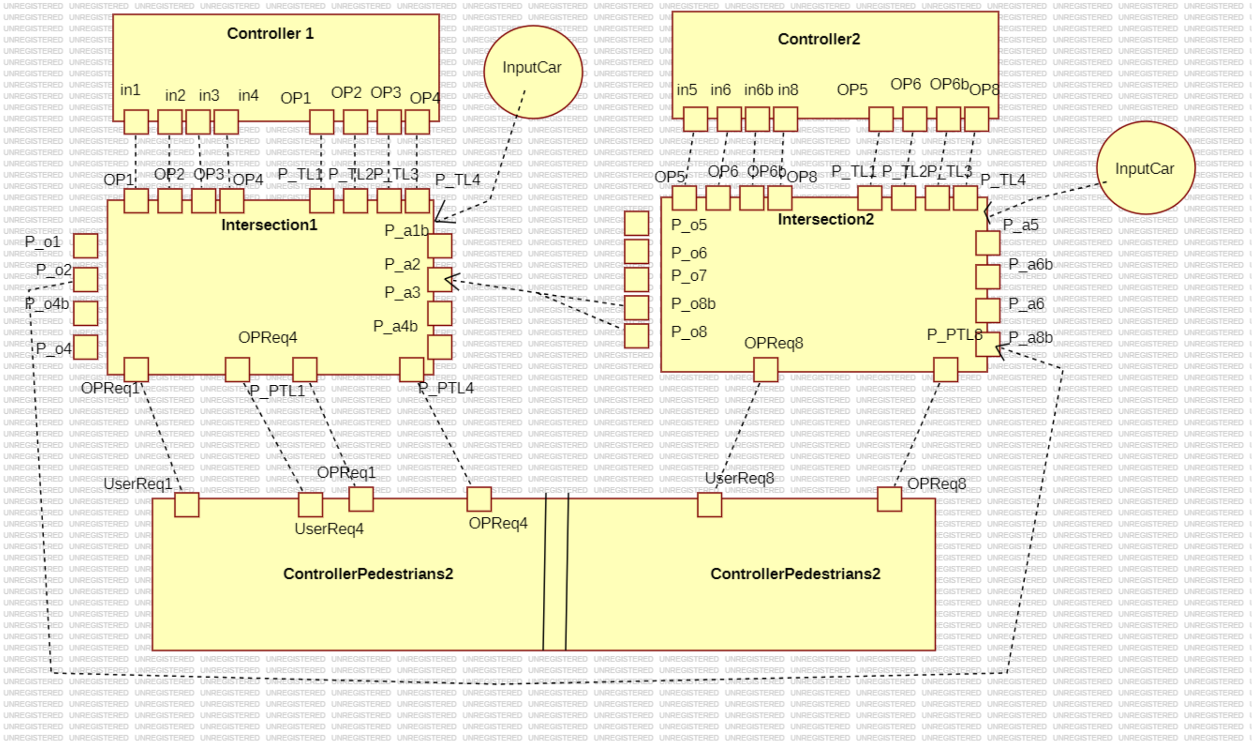
a. More complex drawing



b. Simplified Version



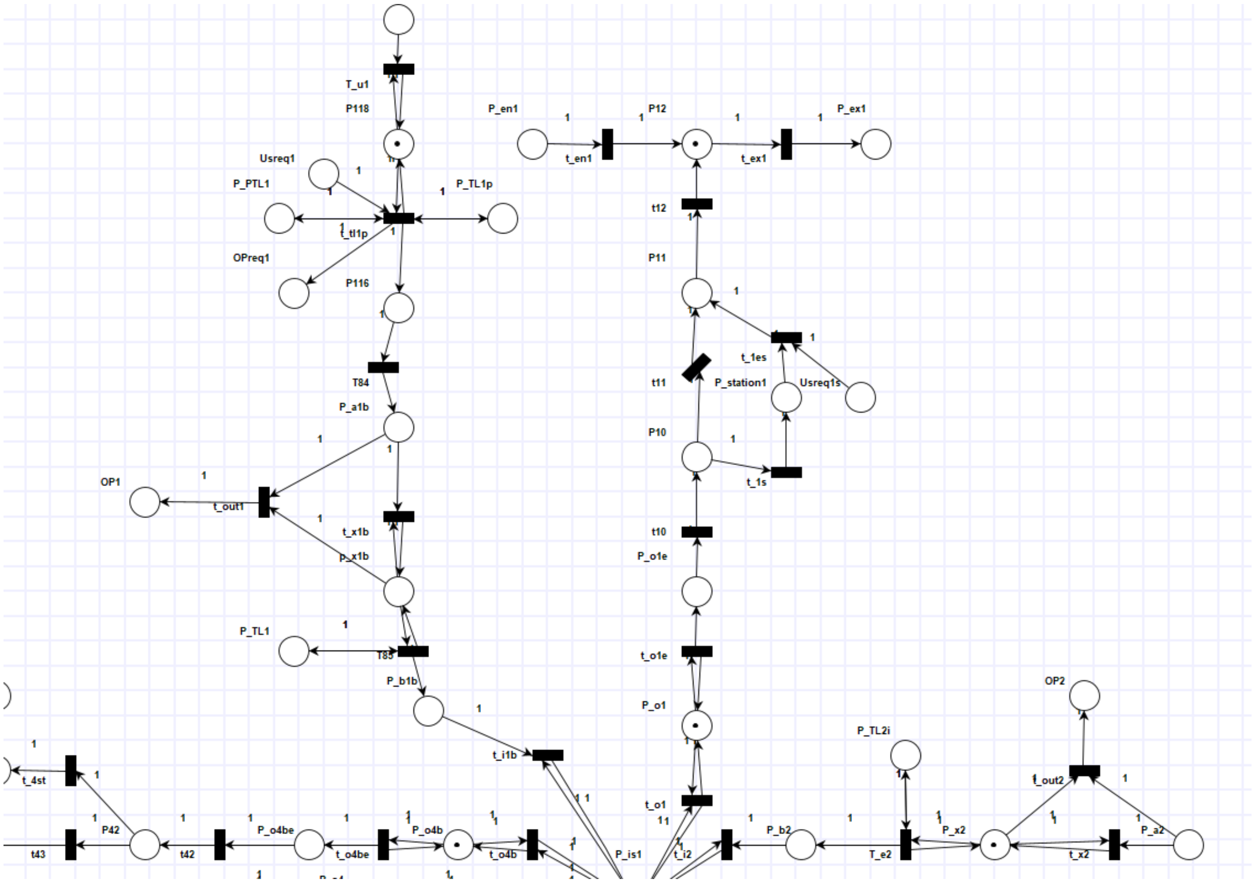
2.3. Draw the component diagram (using UML drawing tool) for the entire system (depending on your implementation, each OETPN is considered a component) and show the names of the input and output channels.

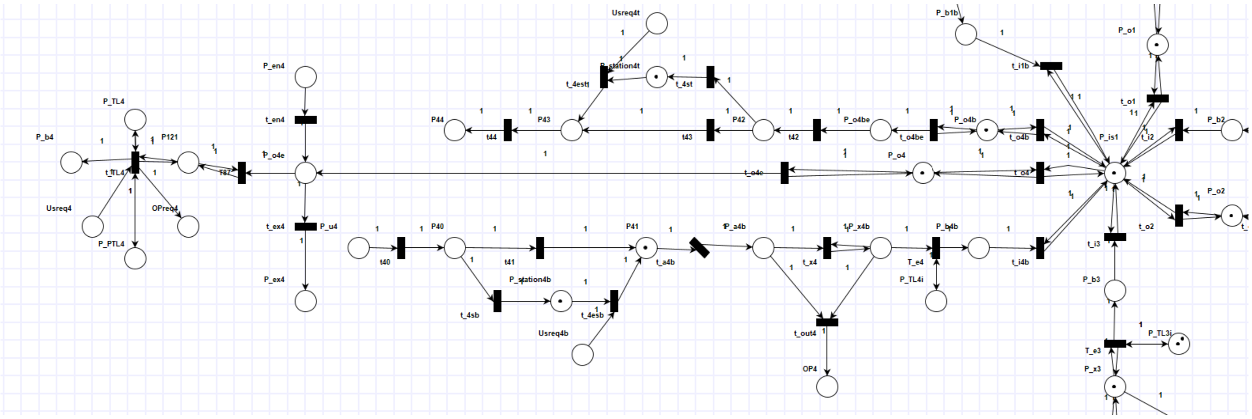


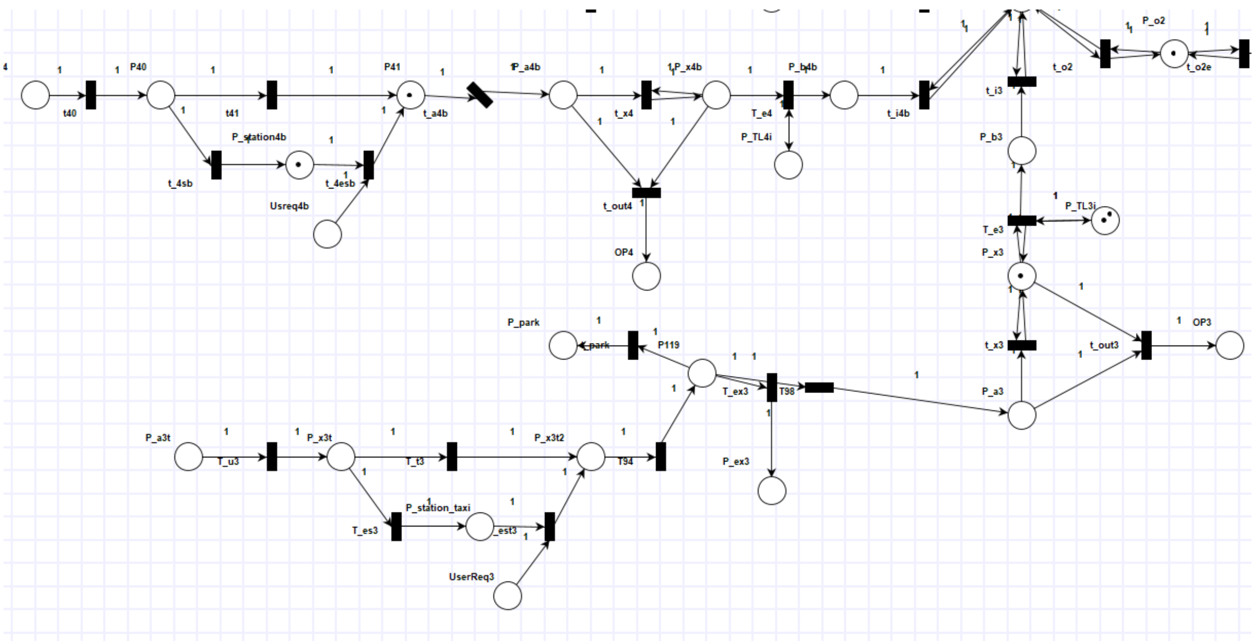
3. Design

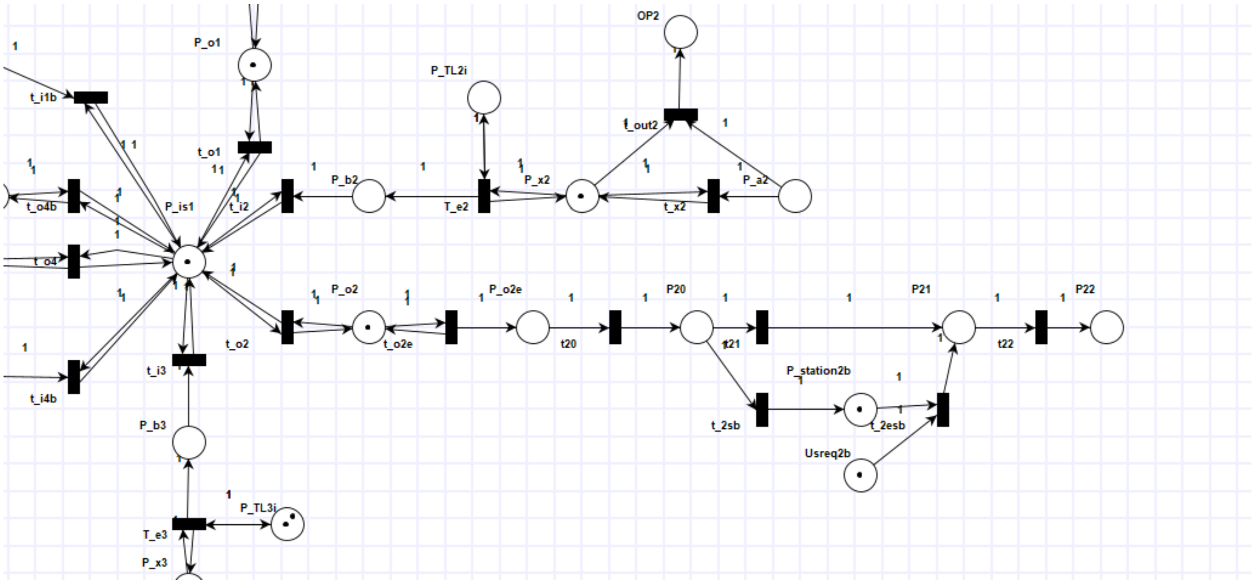
3.1. Draw the OETPN model for the Plant (the intersections, input and output lanes, and the middle street).

a. Intersection 1 (Pin 1)

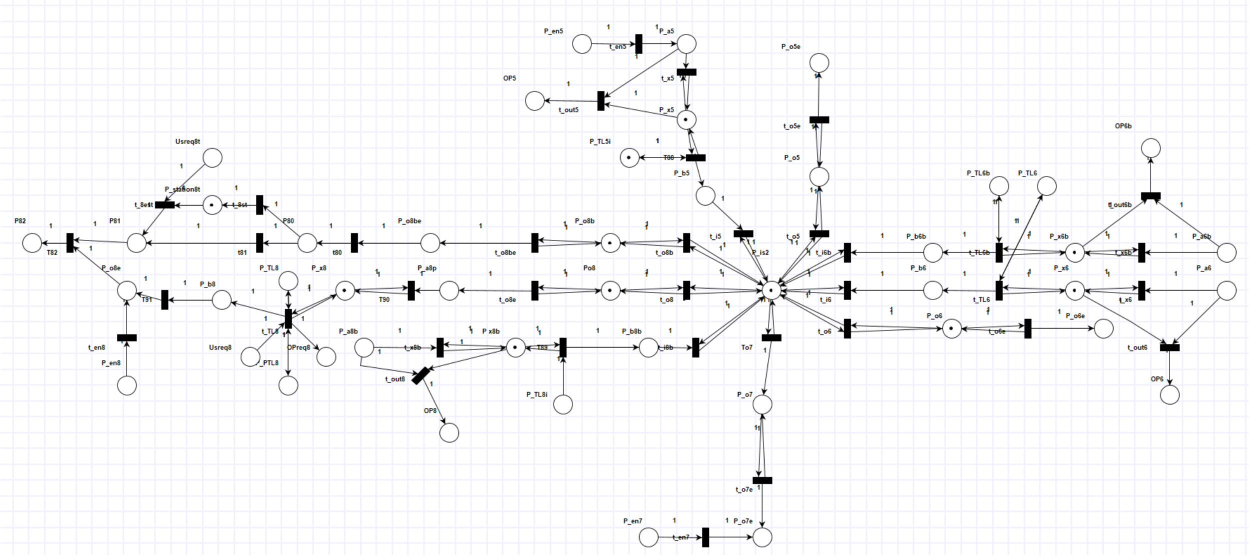





P40
t40
t41
P_station4b
t_4sb
t_4esb
Usreq4b
P41
t_a4b
P_a4b
t_x4
P_x4b
t_out4
OP4
T_e4
P_TL4i
P_b4b
t_i4b
t_i3
P_b3
T_e3
P_TL3i
P_x3
t_x3
P_a3
t_out3
OP3
t_o2
P_o2
t_o2e
P_park
t_park
P119
T_ex3
T98
P_ex3
P_a3t
T_u3
P_x3t
T_t3
P_x3t2
T94
T_es3
P_station_taxi
t_est3
UserReq3


P_o1
t_i1b
t_o1
t_o4b
P_is1
t_i2
P_b2
T_e2
P_TL2i
P_x2
t_out2
OP2
P_a2
t_x2
t_o4
P_o2
t_o2
t_o2e
P_o2e
t20
P20
t21
P21
t22
P22
P_station2b
t_2sb
t_2esb
Usreq2b
t_i4b
t_i3
P_b3
T_e3
P_TL3i
P_x3

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_a1, P_a1b, P_b1b, P_a2, P_b2, P_a3, P_b3, P_u4, P_a4b, P_a3t, P_park, P_4b4

DataCarQueue: P_x1b, P_x2, P_park, P_x3, P40, P_station4b, P41, P_x4b, P_b4b, P_x3t, P_station_taxi, P_x3t2

DataString: green, full, P_TL1, UserReq1, P_PTL1, P_TL2, UserReq2, P_PTL2, P_TL3, UserReq3, P_PTL3, P_TL4, P_PTL4, UserReq4b, UserReq4, P_TL1p, P_TL2i, P_TL3i, P_TL4i

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, P116, P_ex3

DataCarQueue: P_o1, P10, P_station1, P11, P12, P_o2, P20, P_station2b, P21, P_o4, P_o4e, P_o4b, P42, P_station4t, P43, P118, P119, P41

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_u1: P_a1 != Null && P118 CanAddCars
P118.AddElement(P_a1)

T84: P116 != Null && P_a1b CanAddCars
P_a1b.AddElement(P116)

T_x1b: P_a1b != Null && P_x1b CanAddCars || P_a1b IsBus || P_a1b IsProrityCar
P_x1b.AddElement(P_a1b)

T_out1: P_a1b != Null && P_x1b CanNotAddCars

OP1.SendOverNetwork(“full”)


T_TL1p: P_TL1p = green && P118 HaveCar && P_PTL1 = P_PTL1 && UserReq1 != Null

P116.AddElemeny(P118)

P_TL1p.Move(P_TL1p)

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: P119 != Null && P_park CanAddCars

P_park.PopElementWithTargert(P119)

t_e3: P_TL3i == green && P_x3 HaveCar

P+b3.AddElement(P_x3)

T40: P_u4 IsBus && P_u4 && IsPriorityCar || P40 CanAddCars

P40.AddElement(P_u4)

T_4sb: P40 HaveBus

P_station4b.PopElementWithoutTrgetToQueue(P40)

T_4esb: P_station4b HaveCar

P41.PopElementWithoutTargetToQueue(P_station4b)

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, P_TL5i, P_TL8i
DataTransfer: OPReq5, OP5, OP6b, OP6, OPReq8, OP8, P82

- ♦ 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

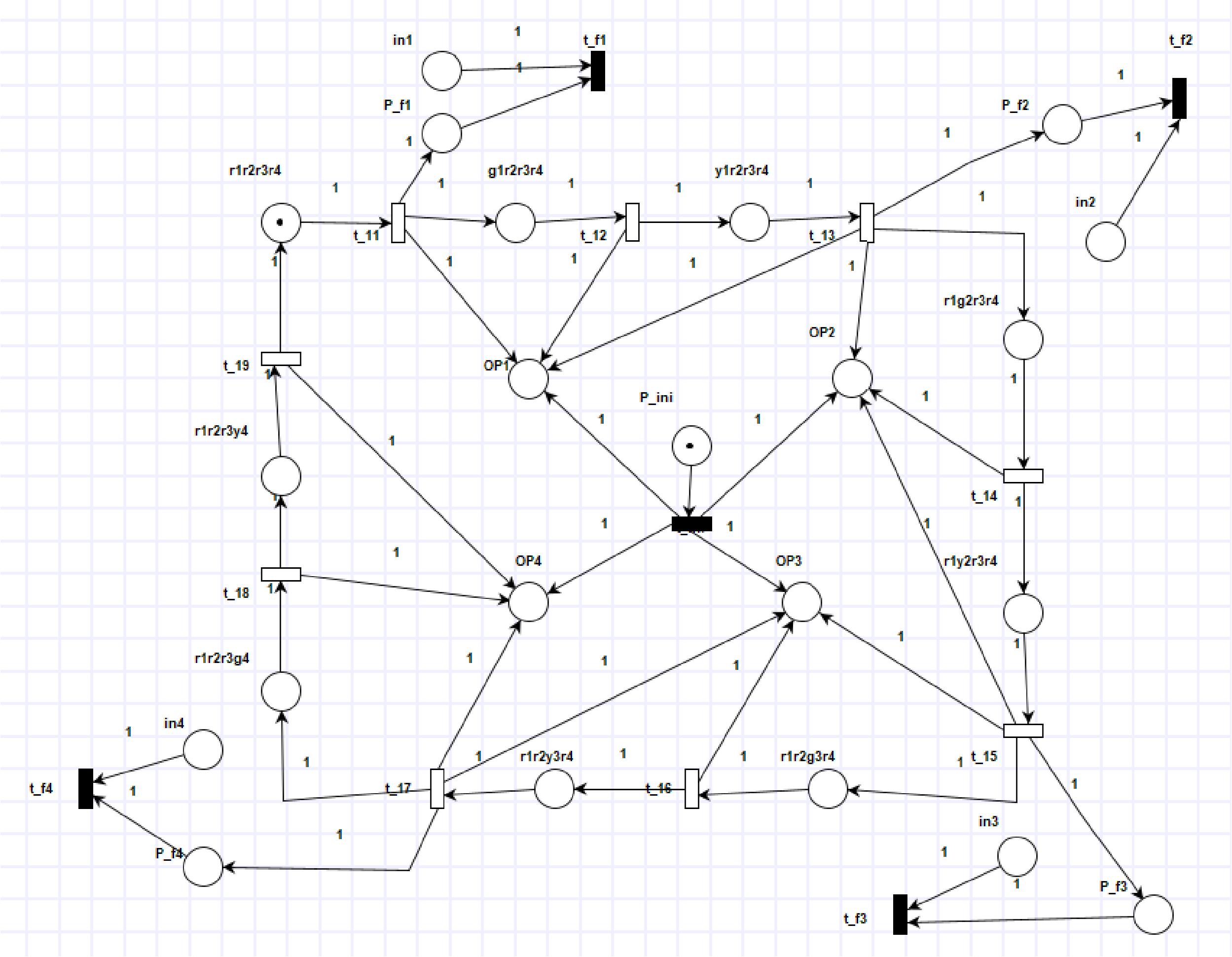
T_o7e: P_o7 HaveCar
P_o7e.PopElementWithoutTarget(P_o7)

T_out8:same t_out1b
T_x8b: same t_x1b
T_TL8b: same t_TL1b

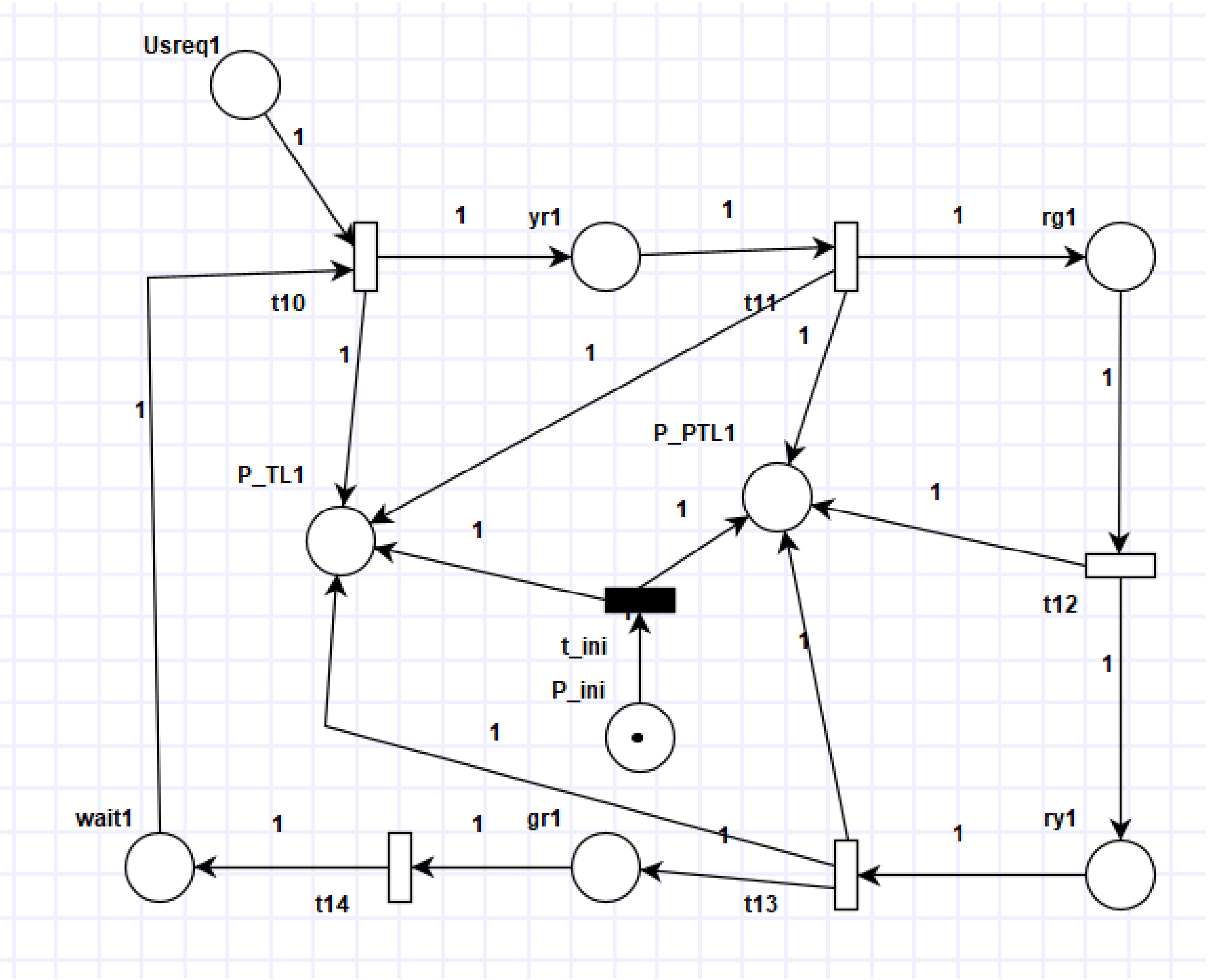
```
T_i8b: same t_i1

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)

T89: P_TL8i == green && P_x8b HaveCar
P_b8b.PopElementWithoutCar(P_TL8i)
P_TL8i.Move(P_TL8i)

t91: P_8b HaveCarForMe
P_o8e.PopElementWithTargetToQueue(P_b8)

t90: P_a8p != Null && P_x8 CanAddCars
P_x8.AddElement(P_a8p)
```

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.



3.4. 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, 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, wait8

DataTransfer: P_TL1, P_TL8, P_PTL1, P_PTL8

- Grd&map:

```
IniT: ini != Null
P_PTL1.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

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

5. 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.