DCS Project Report

- Team Finishers -

Teacher: As.dr.ing. Dahlia Al-Janabi

Students: Paula-Ilinca Sucala
Maria-Diana Moldovan

Year 4, AIA English, Group 30342

Jan 2024

Table of Contents

1. Introduction

1.1 Requirements

The report represents the detailed description of the Finishers team project, involving one given intersection and a roundabout, which require control through a traffic light system.
The project is composed of 4 main sections:

- **Specifications**: the illustration of the intersection and the roundabout through map and drawings.
- **Design**: presentation of the OETPN models for the plants, controllers and connection street, including place types and guard maps. Also includes the component diagram of the system.
- **Implementation**: Git repository link to all the components of the project, including code, drawings and the report.
- **Testing**: The demonstration of the functionality of the OETPN models and control by testing 2 requested scenarios .

1.2 Specification

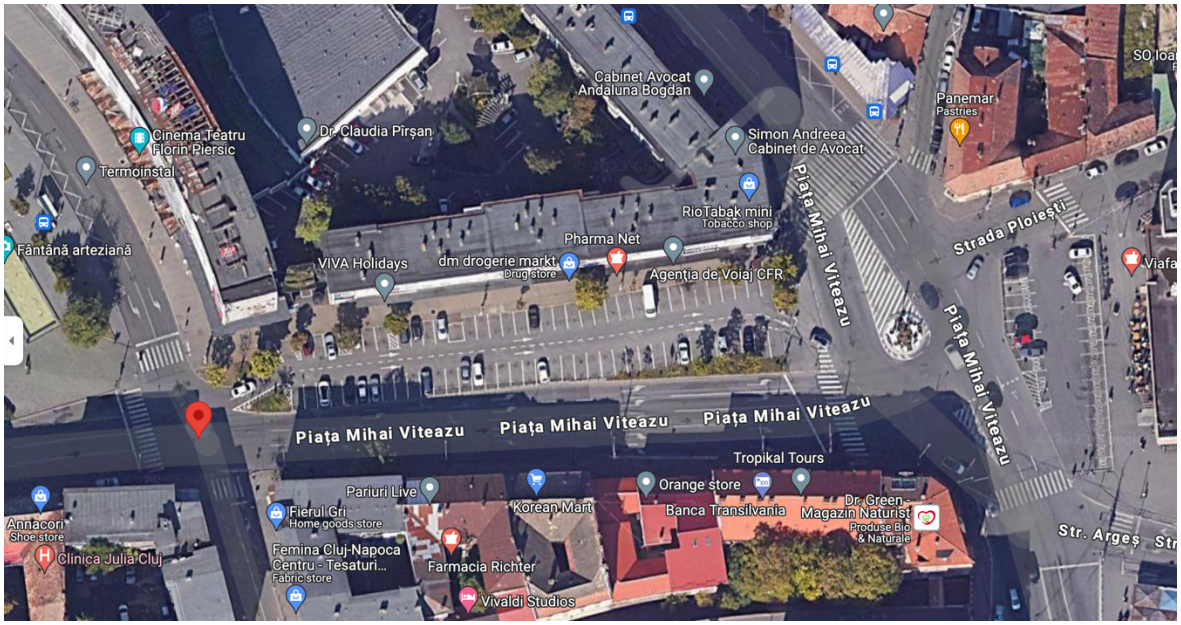


Figure 1.1 Google Earth View

Figure 1.2 Intersection, middle street and the roundabout

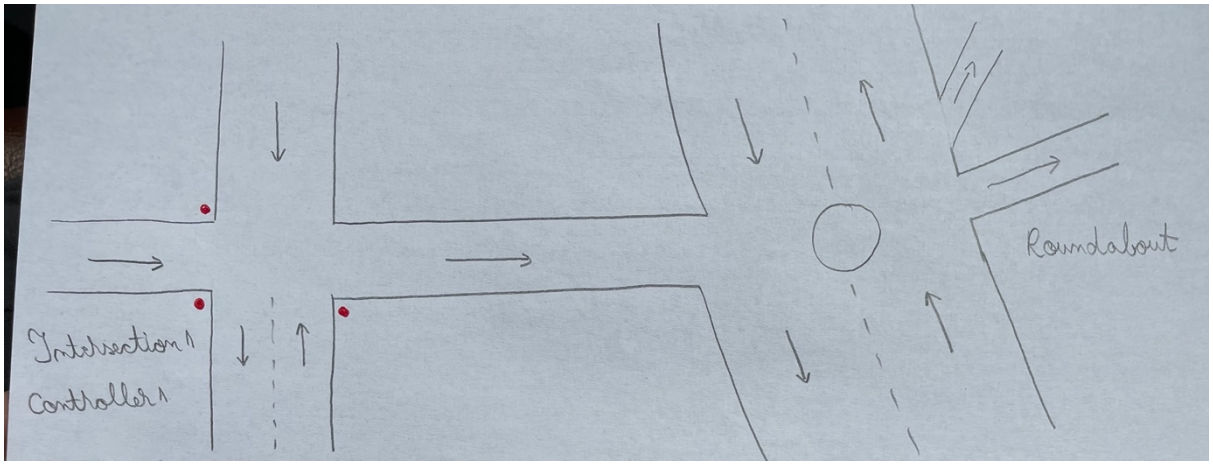


Fig 1.3 Simplified drawn map

The purpose of the project is to represent, model and control the draen map using OETON_OERTPN java framework.
The red dots from „Intersection1" represent the position of control places, meaning the traffic lights. They will help us to do the traffic flow inside the intersection. The traffic lights of the connection between middle street and roundabout are ignored.

2. Design

2.1 Roundabout

2.1.1 Drawing and Petri Net

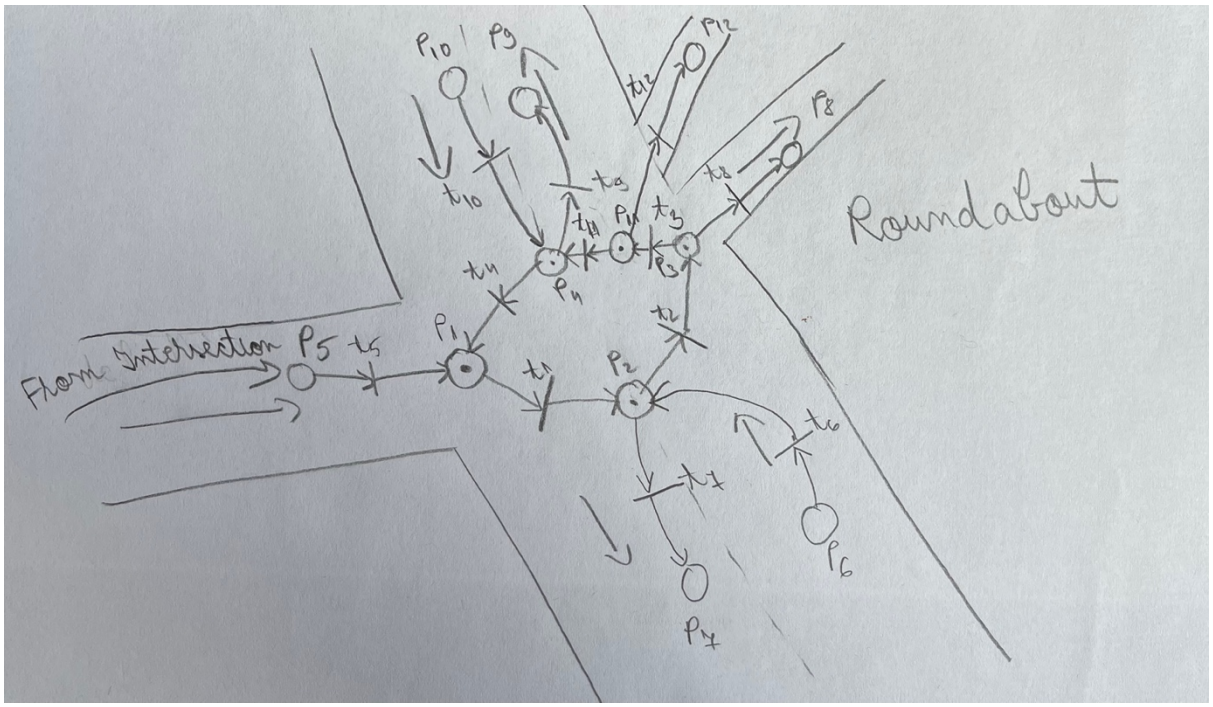


Fig 2.1.1 Drawn by hand roundabout with locations and transitions

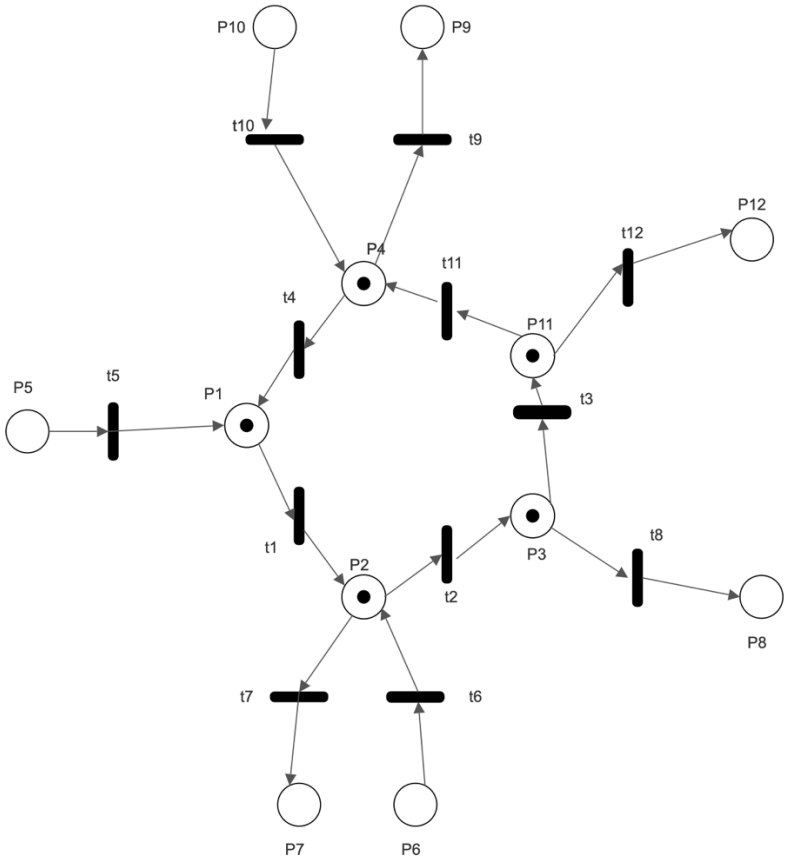


Fig 2.1.2 Petri Net for Roundabout

2.1.2 Places and transitions

- **Place types:**

DataCarQueue (Lanes): P_1, P_2, P_3, P_4, P_{11}
DataCar: P_5, P_6, P_7, P_8, P_9, P_{10}, P_{12}

- **Grd and map:**

t_1 : (P_1.HaveCarForMe) AND (P_2.CanAddCars)
P_1.PopElementWithTargetToQueue(P_2)

t_2 : (P_2.HaveCarForMe) AND (P_3.CanAddCars)
P_2.PopElementWithTargetToQueue(P_3)

t_3 : (P_3.HaveCarForMe) AND (P_{11}.CanAddCars)
P_3.PopElementWithTargetToQueue(P_{11})

t_4 : (P_4.HaveCarForMe) AND (P_1.CanAddCars)
P_4.PopElementWithTargetToQueue(P_1)

t_{11} : (P_{11}.HaveCarForMe) AND (P_4.CanAddCars)
P_{11}.PopElementWithTargetToQueue(P_4)

t_5 : ($P_5 \neq$ NULL) AND (P_1.CanAddCars)
P_1.AddElement(P_5)

t_6 : ($P_6 \neq$ NULL) AND (P_2.CanAddCars)
P_2.AddElement(P_6)

t_{10} : ($P_{10} \neq$ NULL) AND (P_4.CanAddCars)
P_4.AddElement(P_{10})

t_7 : (P_2.HaveCarForMe)
P_2.PopElementWithTarget (P_7)

t_8 : (P_3.HaveCarForMe)
P_3.PopElementWithTarget (P_8)

t_9 : (P_4.HaveCarForMe)
P_4.PopElementWithTarget (P_9)

t_{12} : (P_{11}.HaveCarForMe)
P_{11}.PopElementWithTarget (P_{12})

2.2 Intersection

2.2.1 Drawing and Petri Net

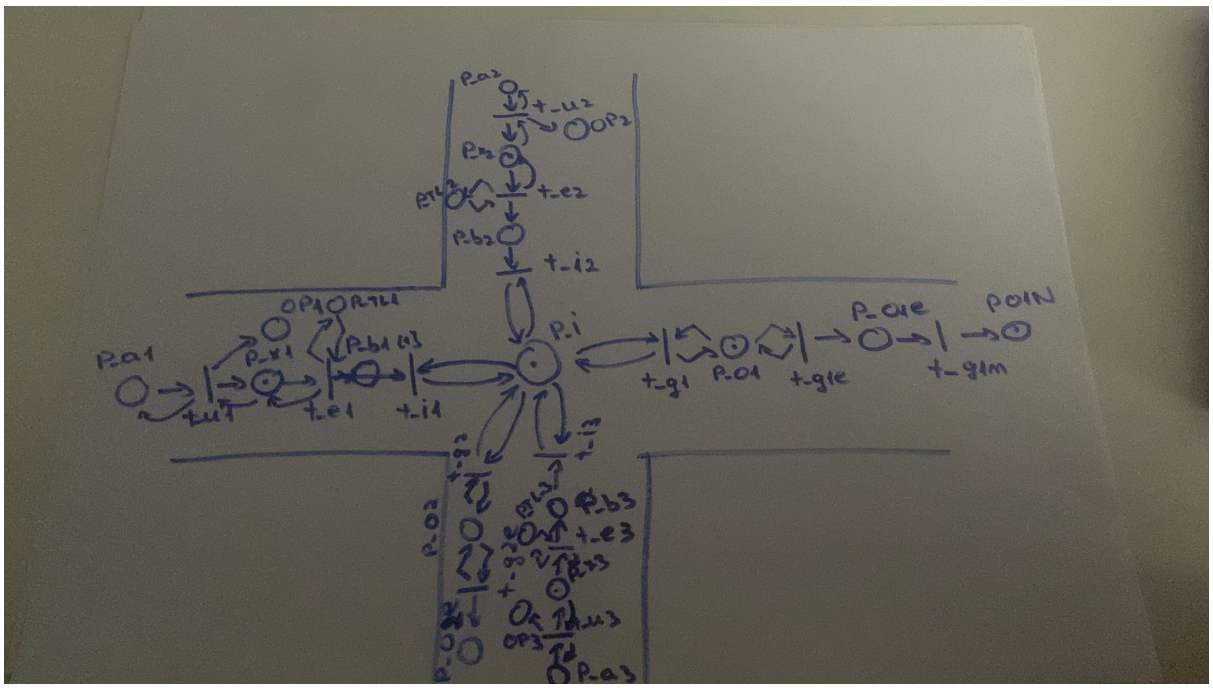


Fig 2.2.1 Drawn by hand intersection with locations and transitions

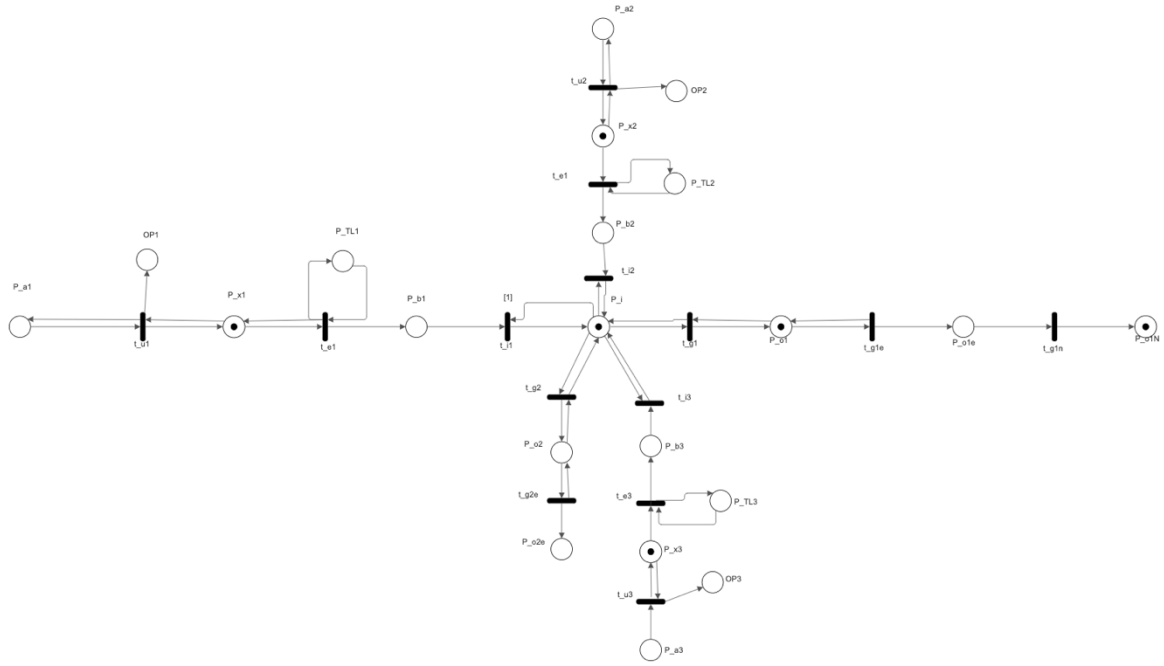


Fig 2.2.2 Petri Net for Intersection

2.2.2 Places and transitions

- **Place types:**

DataCar: P_a1, P_a2, P_a3, P_b1, P_b2, P_b3, P_o1e, P_o2e
DataTransfer: OP1, OP2, OP3, PO1N
DataCarQueue: P_x1, P_x2, P_x3, P_i, P_o1, P_o2
DataString: P_TL1, P_TL2, P_TL3

- **Grd and map:**

t_u1: input place: P_a1, P_x1
grd1: (m(P_a1) ≠ φ AND m(P_x1).CanAddCars));
map1: m(P_a1).addElement() (m(P_x1))
grd2: (m(P_a1) ≠ φ AND m(P_x1).CanNotAddCars));
map2: m(P_a1).Move() (m(P_a1)) ; m(OP1) .SendOverNetwork(full)
(same logic applies to t_u2 and t_u3)

t_e1: input place: P_x1, P_TL1
grd: (m(P_x1).HaveCar AND m(P_TL1) = green);
map: m(P_x1).PopElementWithoutTarget() (m(P_b1)); m(P_TL1).Move() m(P_TL1)
(same logic applies to t_e2 and t_e3)

t_i1: input place: P_b1, P_I
grd: (m(P_b1) ≠ φ AND m(P_I).CanAddCars));
map: m(P_b1).AddElement() (m(P_I))
(same logic applies to t_i2 and t_i3)

t_g2: input place: P_I, P_o2
grd: (m(P_I).HaveCar And m(P_o2).CanAddCars));
map: m(P_I).PopELementWithTargetToQueue() (m(P_o2))
(same logic applies to t_g1)

t_g2e: input place: P_o2
grd: (m(P_o2).HaveCar));
map: m(P_o2).PopElementWithoutTarget() (m(P_o2e))
(same logic applies to t_g1e)

t_g1N: input place: P_01e
grd: (m(P_01e) ≠ φ);
map: m(P_01e).SendOverNetwork() (m(P_O1N) = m(P_O1N))

2.3 Controller

2.3.1 Drawing and Petri Net

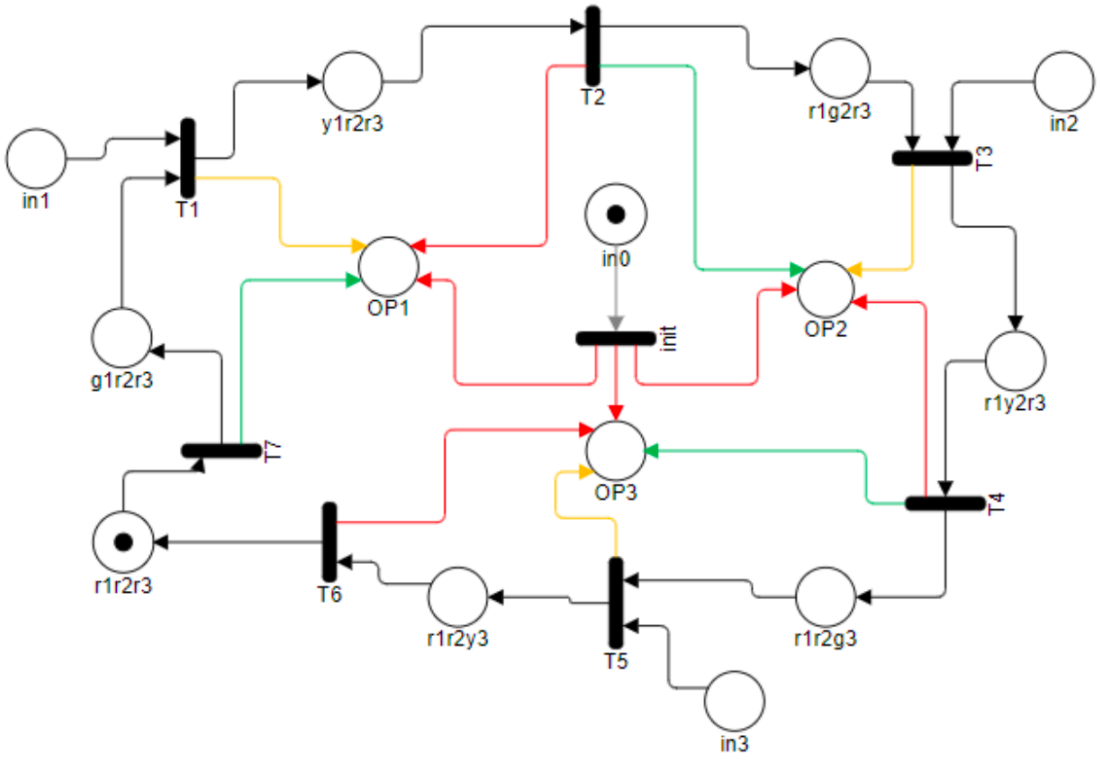


Fig 2.2.2 Petri Net for Controller

2.3.2 Places and transitions

- **Place types:**

DataString: in0, in1, in2, in3
DataTransfer: OP1, OP2, OP3
DataString: r1r2r3,g1r2r3,y1r2r3,r1g2r3,r1y2r3,r1r2g3,r1r2y3

- **Grd and map:**

init: input Place: in0
grd: (m(in0) ≠ φ);
map: m(in0).MakeNull; m(OP1).SendOverNetwork(in0)
m(OP2).SendOverNetwork(in0) m(OP3).SendOverNetwork(in0)

First Group of transitions:
T1 : input place: in1, g1r2r3
grd1: (m(in1) = φ And (m(g1r2r3) ≠ φ)); map1: m(g1r2r3).Move() (m(y1r2r3))
m(OP1).SendOverNetwork(yellow) DynamicDelay(Five)
grd2: (m(in1) ≠ φ And (m(g1r2r3) ≠ φ)); map2: m(g1r2r3).Move() (m(y1r2r3))
m(OP1).SendOverNetwork(yellow) DynamicDelay(Ten)
(t1, t3 and t5 follow the same logic)

Second Group of transitions:
t2 : input place: y1r2r3
grd: (m(y1r2r3) ≠ φ);
map: m(y1r2r3).Move() (m(r1g2r3)); m(OP1).SendOverNetwork(red)
m(OP2).SendOverNetwork(green)
(t2, t4 and t6 follow the same logic)

2.4 Component Diagram

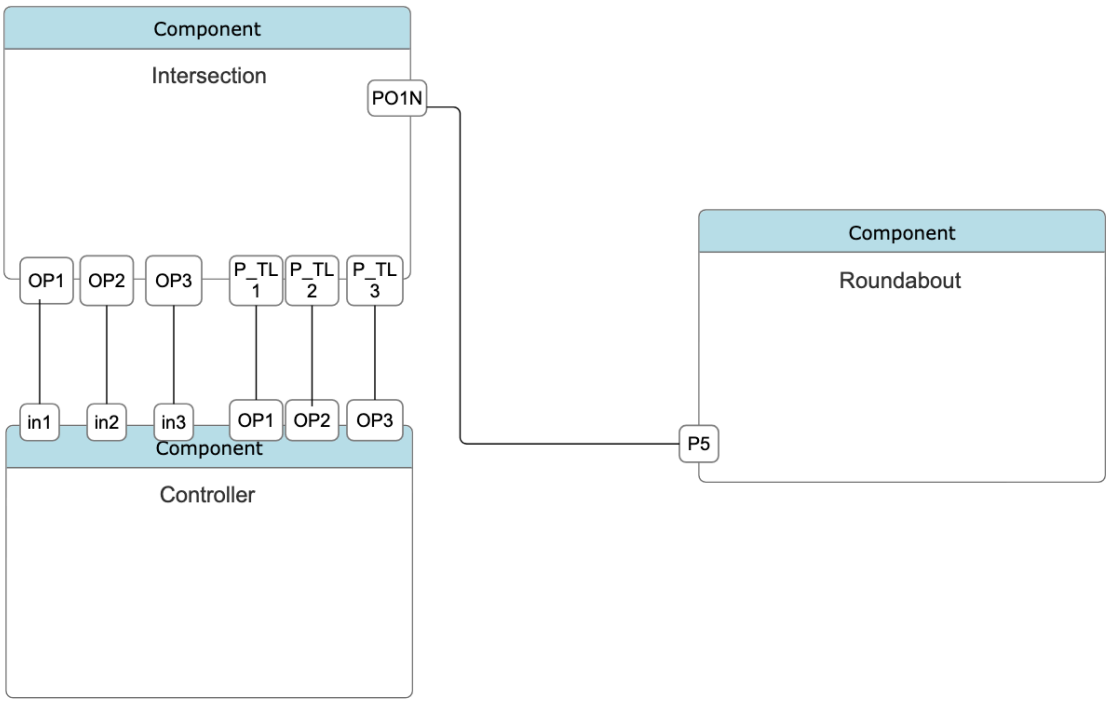


Fig 2.4.1 Component Diagram

3. Implementation

The repository link for the implementation of the project can be found here: https://github.com/sucalailinca/Finishers_Project

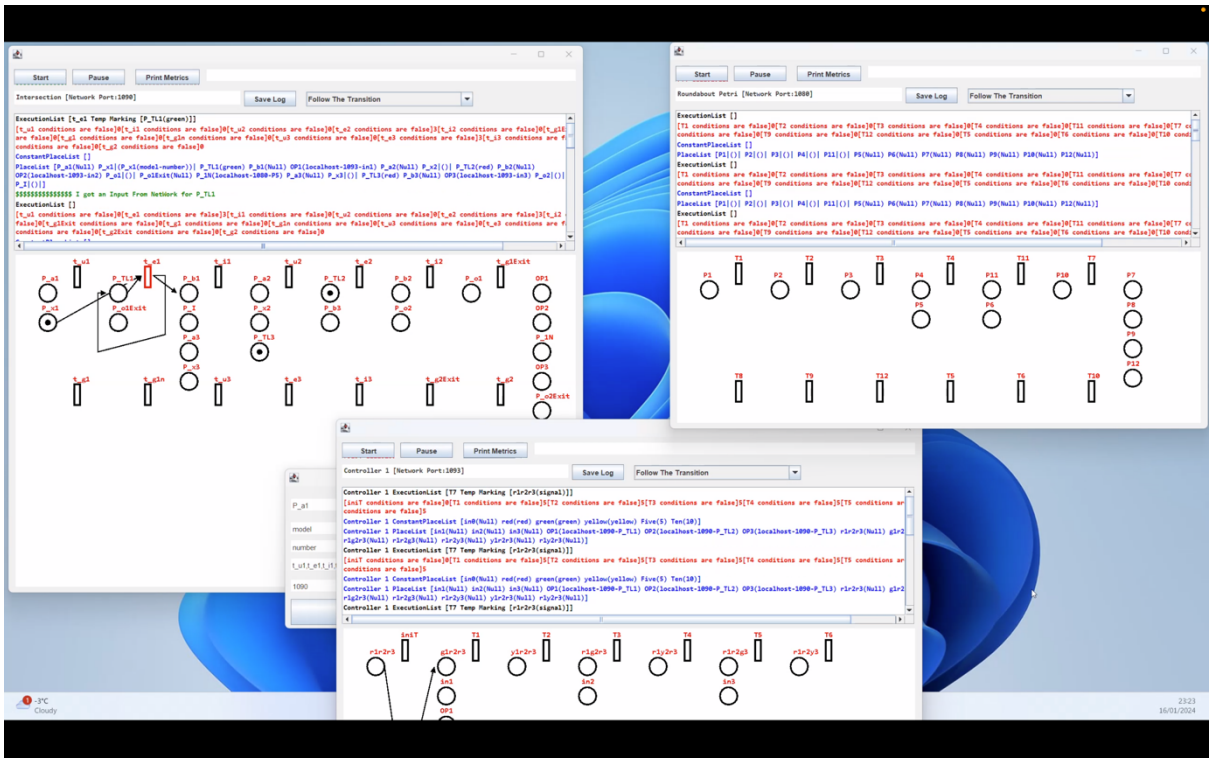
4. Testing

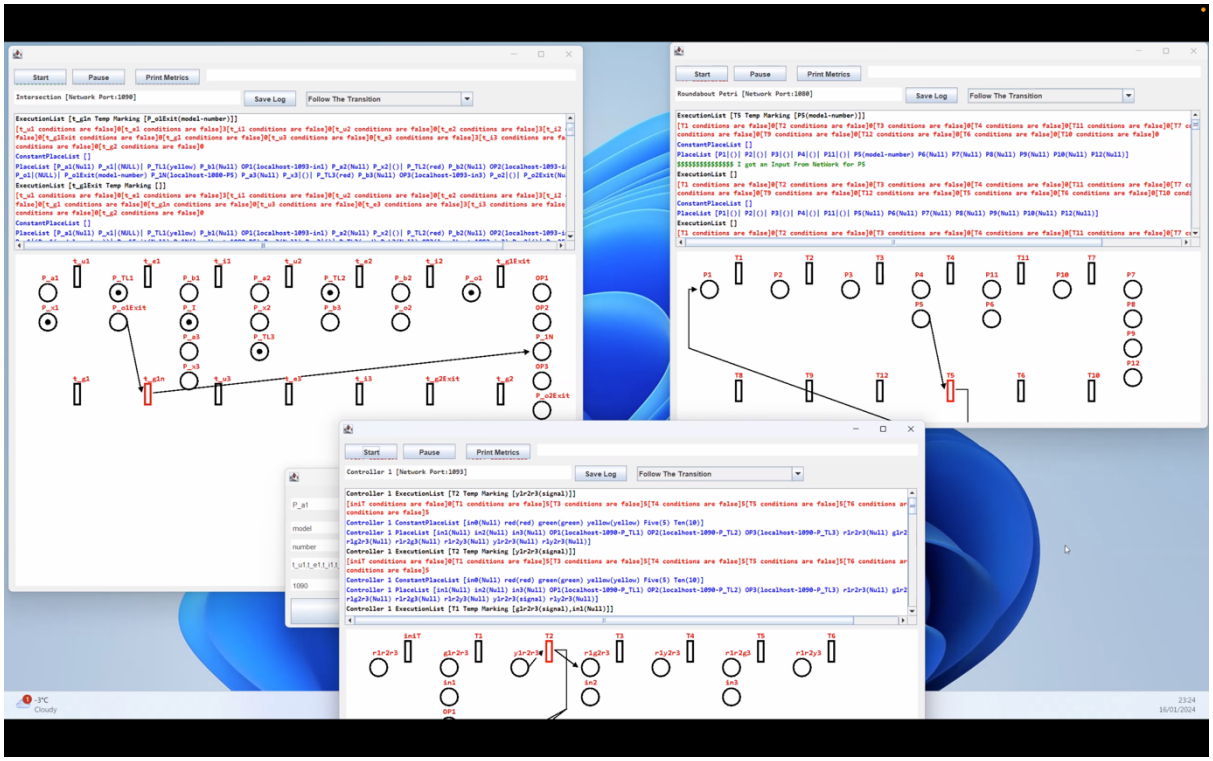
Testing part includes two kind of tests:

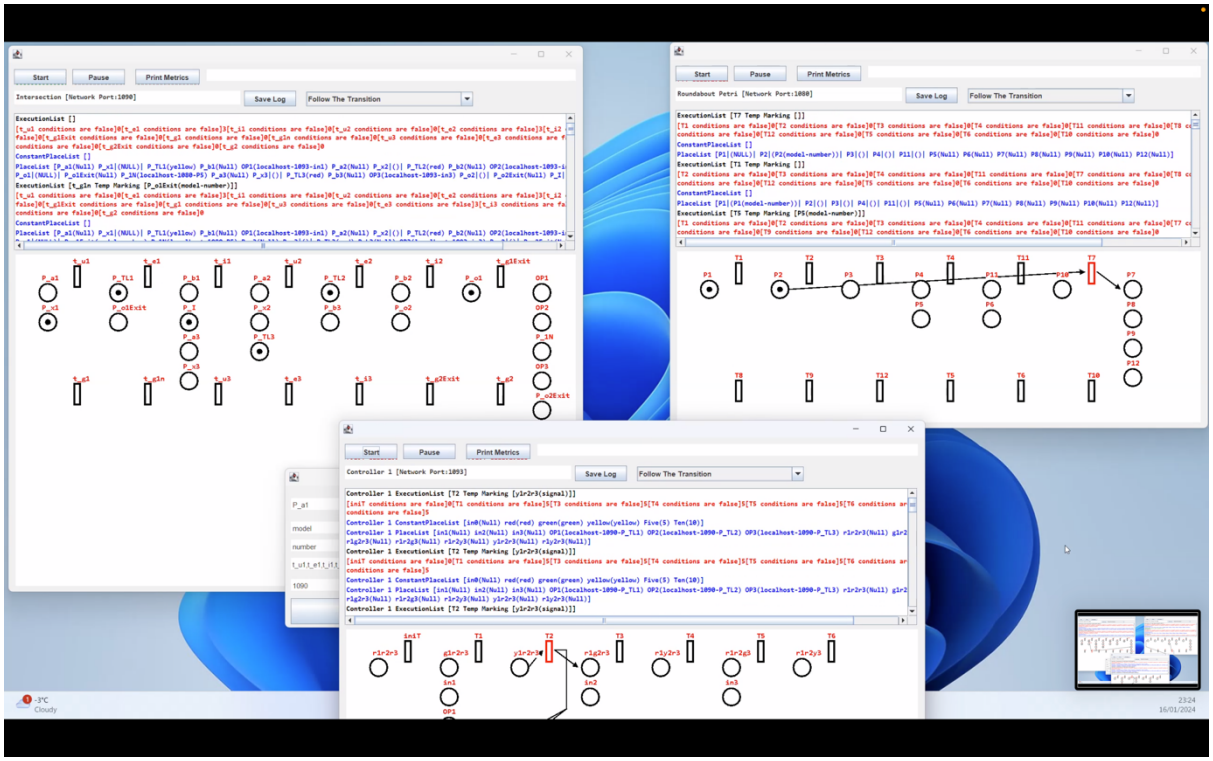
1. Car travelling through intersections
2. Car jam management

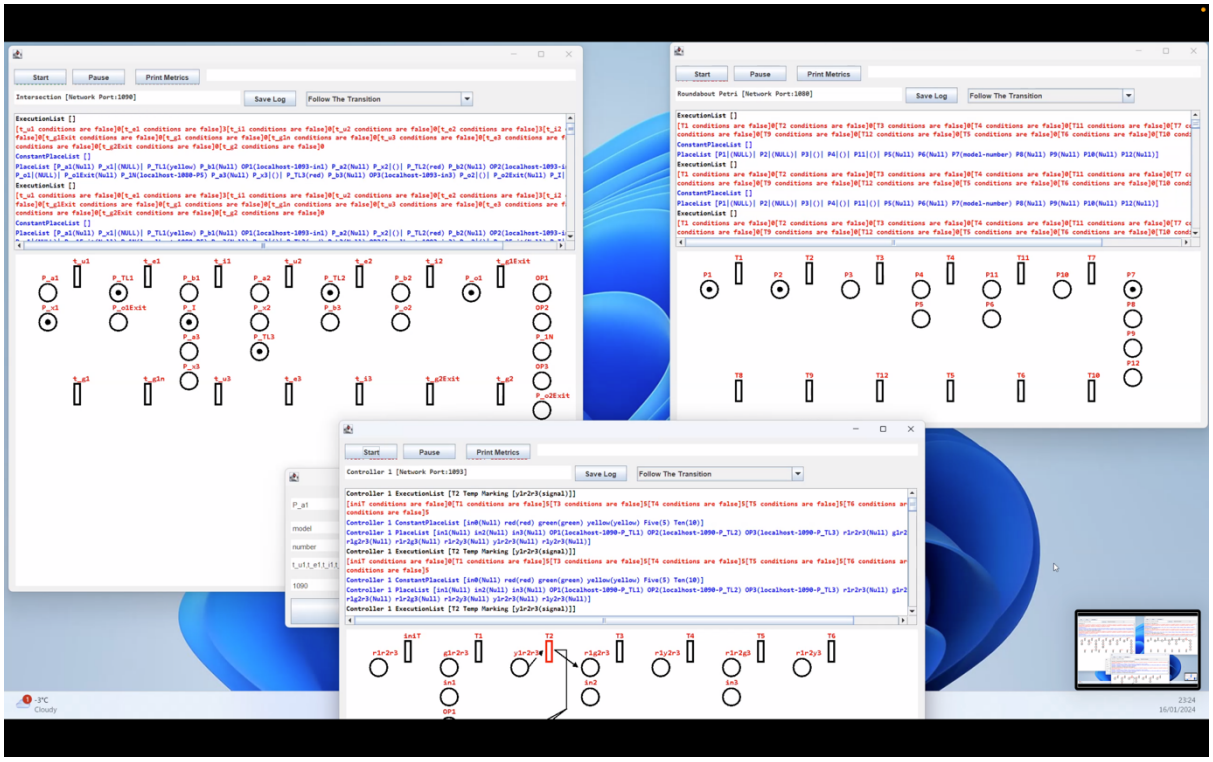
4.1 First Test

Send a car from the 1st intersection, that should go through the middle street and exit from one of the exit lanes from the roundabout. Screen shots showing how the car moves are attached. A video with the entire test can be found in the git repository form „implementation" chapter.









4.2 Second Test

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. Let the controller OETPN run until it reaches the same transition (2 loops) to show that the delay is changed back to 5 sec.