LA Project Report

Year IV, AIA English, Group 30342&30341

Students: Malita Alin, Tudor Ada, Mosnegutu Vlad

1. Initial Conditions

Figure 1.1 Google Earth view

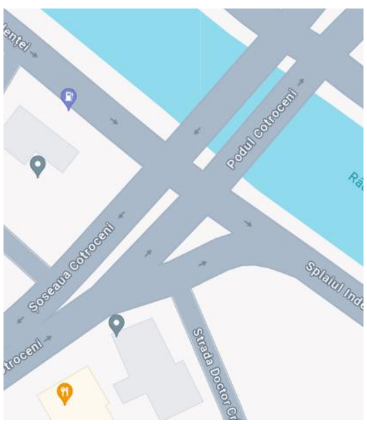

Figure 1.2 Intersection 1 (Pin1)

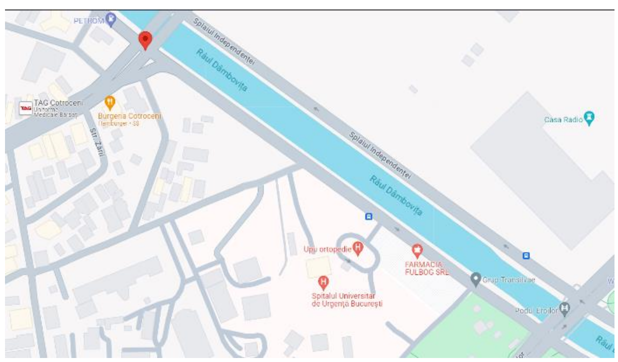

Figure 1.3 Connection Street

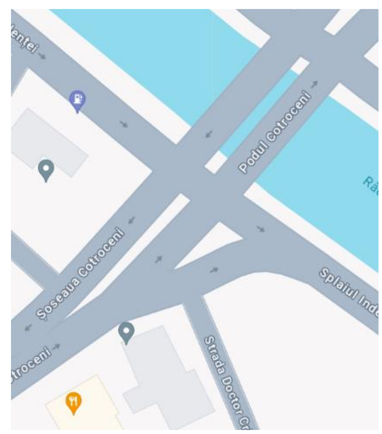

Figure 1.4 Intersection 2 (Pin2)

Based on the given Intersections, the prospect of the project is the representation, modelling and control of the intersection using the OETPN_OERTPN java framework. Below the system is simplified through illustration fig1.5.

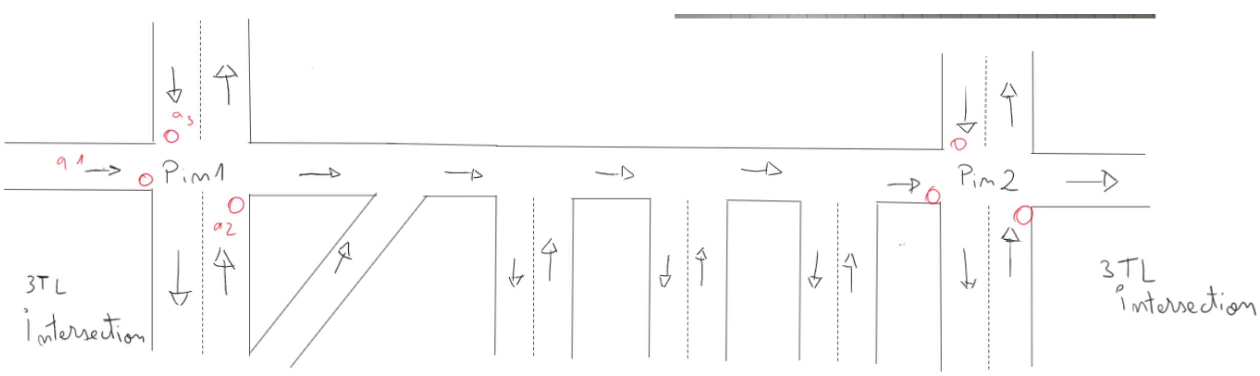

Figure 1.5 Illustrated Intersection System

2. Guard Mapping

2.1 Intersections - Transitions

Since both intersections are extremely similar, their transitions are the same, with the exception of transition t_4n and place P_4N present only for Intersection1.

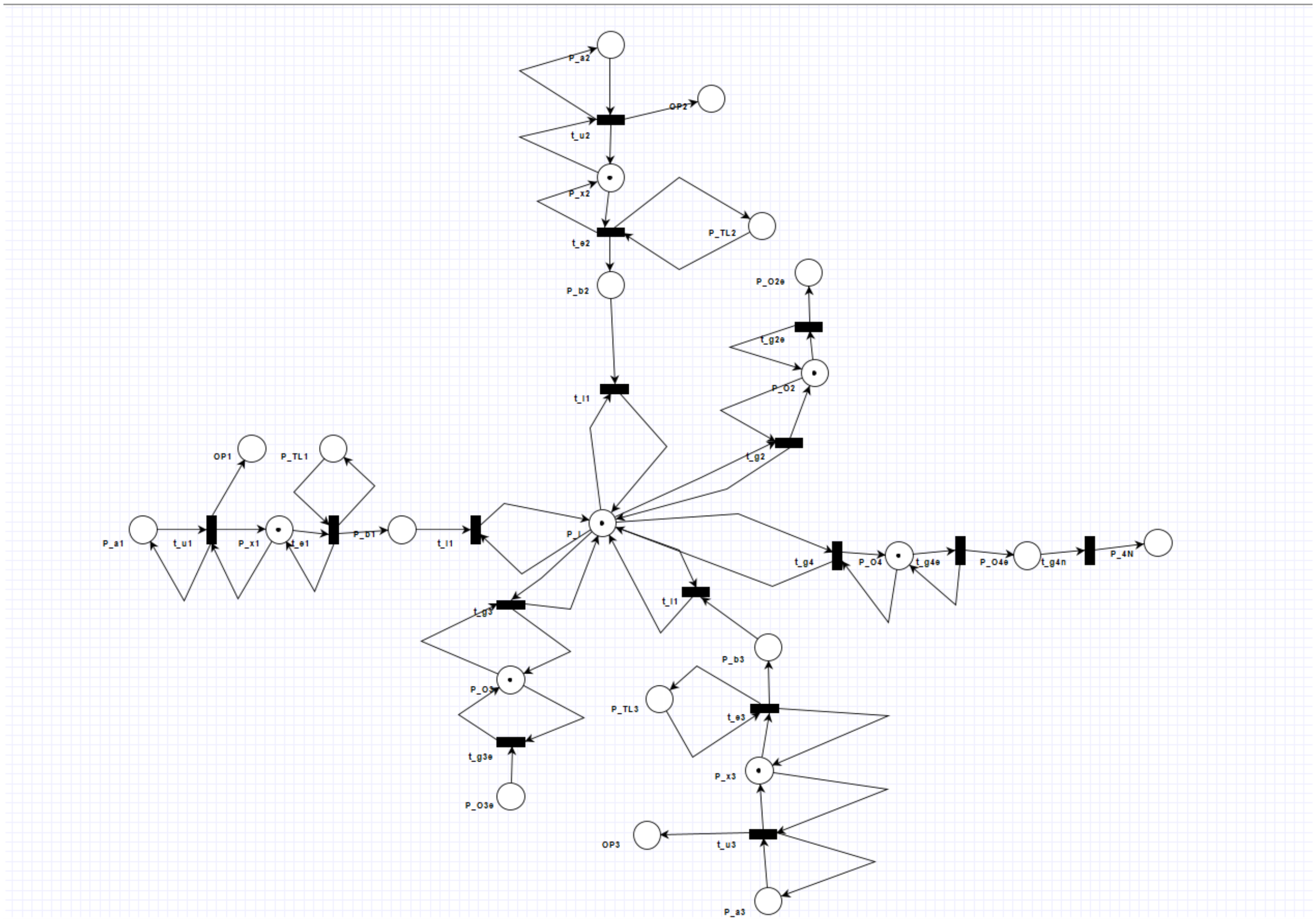

Figure 2.1.1 Intersection 1

PLACES:

- ➔ P__a1, P_a2, P_a3, P_b1,P_b2, P_b3, P_o2Exit, P_o3Exit, P_o4Exit = DataCar type
- ➔ OP1, OP2, OP3, P_4N = DataTransfer type
- ➔ P_x1, P_x2, P_x3, P_I, P_o2, P_o3, P_o4 = DataCarQueue type
- ➔ P_TL1, P_TL2, P_TL3 = DataString type

TRANSITIONS:

- 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_g3 and t_g4

- t_g2e: input place: P_o2
  grd: (m(P_o2).HaveCar));

  map: m(P_o2).PopElementWithoutTarget() (m(P_o2Exit))

same logic applies to t_g3e and t_g4e

- t_4N: input place: P_04e
  grd: (m(P_04e) ≠ ϕ);
  map: m(P_04e).SendOverNetwork() (m(P_4N) = m(P_4N)

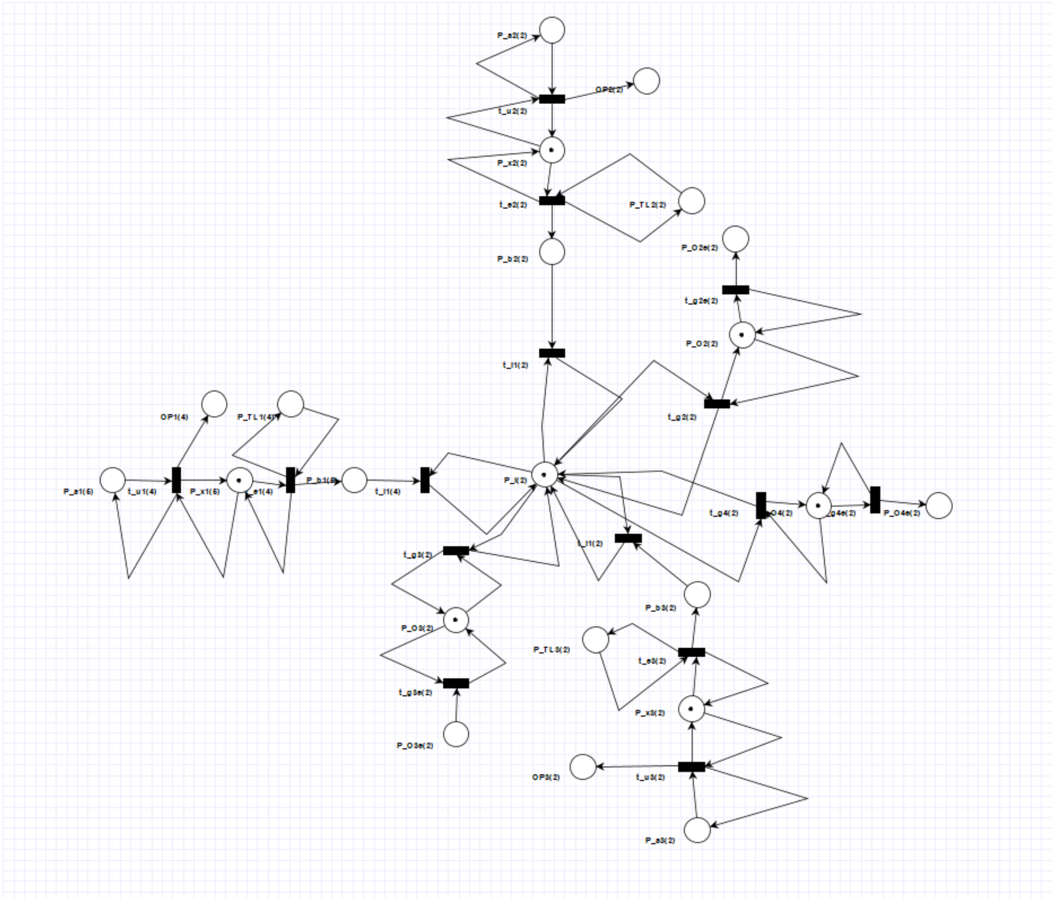

Figure 2.1.2 Intersection 2

Connecting Street – Transitions

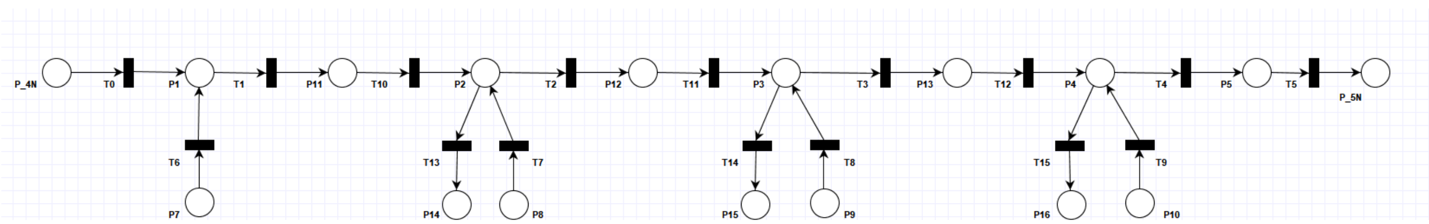

Figure 2.2.1 Connection Street PetriNet

PLACES:

→ P_4N, P5, P7, P8, ….P15,P16 = DataCar type

→ P_5N = DataTransfer type

→ P1, P2, p3, P4 = DataCarQueue type

TRANSITIONS:

→ First Group of transitions

- t0 : input place: P_4N

  grd: (m(P_4N) ≠ ϕ And m(p1).CanAddCars));

  map: m(P_4N).addElement() (m(p1))

- t6 : input place: p7
  grd: (m(p7) ≠ ϕ And m(p1).CanAddCars));

  map: m(p7).addElement() (m(p1))
- t7 : input place: p8
  grd: (m(p8) ≠ ϕ And m(p2).CanAddCars));

  map: m(p8).addElement() (m(p2))
- t8 : input place: p9
  grd: (m(p9) ≠ ϕ And m(p3).CanAddCars));

  map: m(p9).addElement() (m(p3))
- t9 : input place :p10
  grd: (m(p10) ≠ ϕ And m(p4).CanAddCars));

  map: m(p10).addElement() (m(p4))

t10, t11 and t12 will follow the same logic as the t0 ->t9 transitions

➔ Second Group

- t1 : input place: p1
  grd: (m(p1).HaveCarForMe);

  map: m(p1).PopElementWithTarget() (m(p11))
- t2 : input place: p2
  grd: (m(p2).HaveCarForMe);

  map: m(p2).PopElementWithTarget() (m(p12))
- t3 : input place: p3
  grd: (m(p3).HaveCarForMe);

  map: m(p3).PopElementWithTarget() (m(p13))
- t13 : input place: p2
  grd: (m(p2).HaveCarForMe);

  map: m(p2).PopElementWithTarget() (m(p14))
- t14 : input place: p3
  grd: (m(p3).HaveCarForMe);

map: m(p3).PopElementWithTarget() (m(p15))

transitions t1,t2,t3,t4 and t13,t14,t15 all have similar logic

➔ Transition t5: input place: p5
grd: (m(p5) ≠ ɸ);
map: m(p5).SendOverNetwork() (m(p5n) = m(p_a1)

2.3 Controllers - Transitions

Since the intersections are so similar, their controllers will have the same logic for their transitions and places

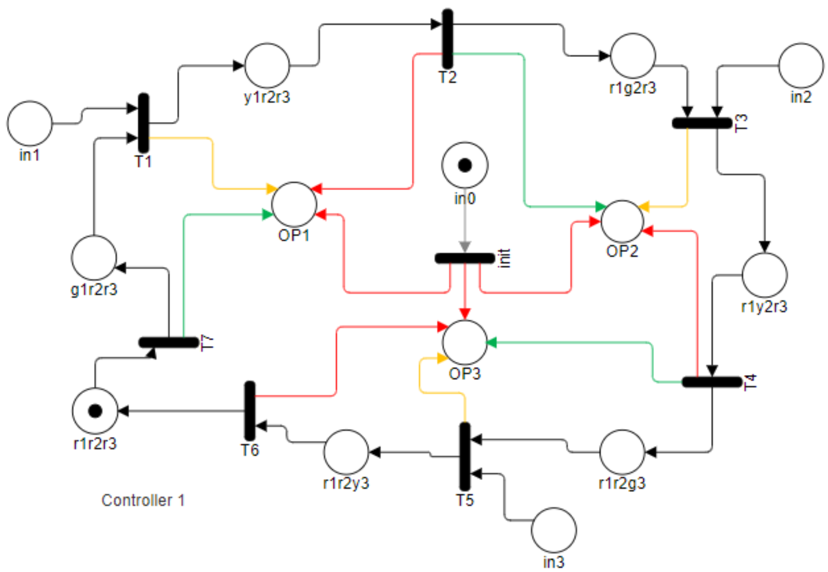

Figure 2.3.1 Controller 1 (same as Controller 2)

PLACES:

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

TRANSITIONS:

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

➔ First Group of transitions

- t2 : input place: in1, g1r2r3
grd1: (m(in1)= ɸ And (m(g1r2r3) ≠ ɸ));
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

3. Project Code

- ➔ Inside the Git repository, the location of the project file can be found at the following address: OETPN_OERTPN_Framework/New OETPN/New OETPN/src/ProjectLA
- ➔ Testing Folder offers all images and log files related to the 2 tests required for the project's system
- ➔ The Component Diagram is in the main folder of the project
- ➔ The Drawn_Petri_Nets folder holds all petri nets included in this project, created with diagram tools