PETRI NET MODELLING
FOR
LEGO MINDSTROM NXT

Date: 02/11/2015

Developed by:

Roland Schurig

PreranaShamsundar Punjabi

Table of Contents

Executive Summary

Lego Mindstrom is an educational product designed by Lego to build easy Robots. The NXT brick is the brain of any Lego Mindstrom Robot. It's an intelligent, computer-controlled Lego brick that lets a Robot come alive and perform different operations. The Robot consists of a digger, 2 tractor wheels and a freewheel, a light sensor and a compass sensor and 3 actuators. The Robot consists of a USB port or Bluetooth that can be used to upload programs into the internal memory of NXT.

Aim of this project is to program NXT such that the Robot can follow the way points configured and return to the base in the end. In case, the Robot comes across a ball on its way (between any two waypoints), it picks the ball up and delivers it to '*L*' location and returns to the next configured waypoint.

The Robot will be programmed to start from the Initial point '*I (0,0)*' and go to the base point 'B' after which the Robot will pass through the pre-defined set of way points until the last waypoint is reached. The *'node.wp'* is used for configuring the waypoints. Once the last waypoint is reached the Robot must return to the Base point and stop. On its way from Base to the Final waypoint if the Robot detects a ball (In this case: the light sensor's value increases more than 50), it lowers its digger and picks up the ball. If the robot has successfully picked up the ball it will make a BIP sound. Once the Robot picks the ball it shall deliver the ball to a Location '*L*' and then return to the next configured waypoint.

For the purposes of this project we have designed an Interpreted Petri Net model using Tina.**Tina2Lego** translator is used in-order to translate the PN model to IPN model. Next the **PetriNXT** tool is used to translate the *'xxx.net'* files that were generated using Tina to '*xxx.rdp'* file. Once we have a working '*xxx.rdp'* file, this is loaded into the Robot and the program is tested.

The Aim of second part of the lab project is to program two Robots that can run their task in parallel and that can co-ordinate with each other. The Robots communicated with each other using Bluetooth Messages.

1 Model Description – [Single Robot Model Design]

Design Description:

The Main Aim of this project is to program the NXT. When a set of pre-configured waypoints are provided, the Robot will move from one way point to the other until it reaches the last configured waypoint.

The guidelines for the program are as follows:

- The Robot is initialized at *'I (0,0)'* using the command *'INIT_NAVE'*, that
  - Calibrates the compass
  - Initializes the navigation (all coordinates are set to 0)
  - Reads the *'node.wp'* file (contains the list of pre-defined waypoints)
- *'ETALON_CAPT_LUMIERE'* command is used to calibrate the light sensor
- The Robot then moves to the Base location when the calibration is done
- Once the Robot is ready, it moves from one waypoint to the other as defined in *'node.wp'* file
  - If the light sensor value increases above or equal to 50, it indicates that the Robot has encountered a ball along the path
  - The Robot then moves closer to the ball and lowers the digger to catch the ball
  - If the Robot is unsuccessful in catching the ball then it just moves to next way point and rise up the digger
  - If the Robot is successful in catching the ball, it then delivers the ball to a location *'L'* (the delivery point)
  - At location *'L'* the Robot rises up its digger and releases the ball and returns to the last travelled way point, to continue its journey.
  - The above described steps are followed until the last way point is reached
- Once the last waypoint is encountered, the Robot moves back to the Base and Stops

The way points are configured using a waypoint editor; a '*node.wp*' file is created where we specify the different coordinates.

FIGURE 1: WAYPOINT EDITOR

The developed PetriNet model is as shown below:

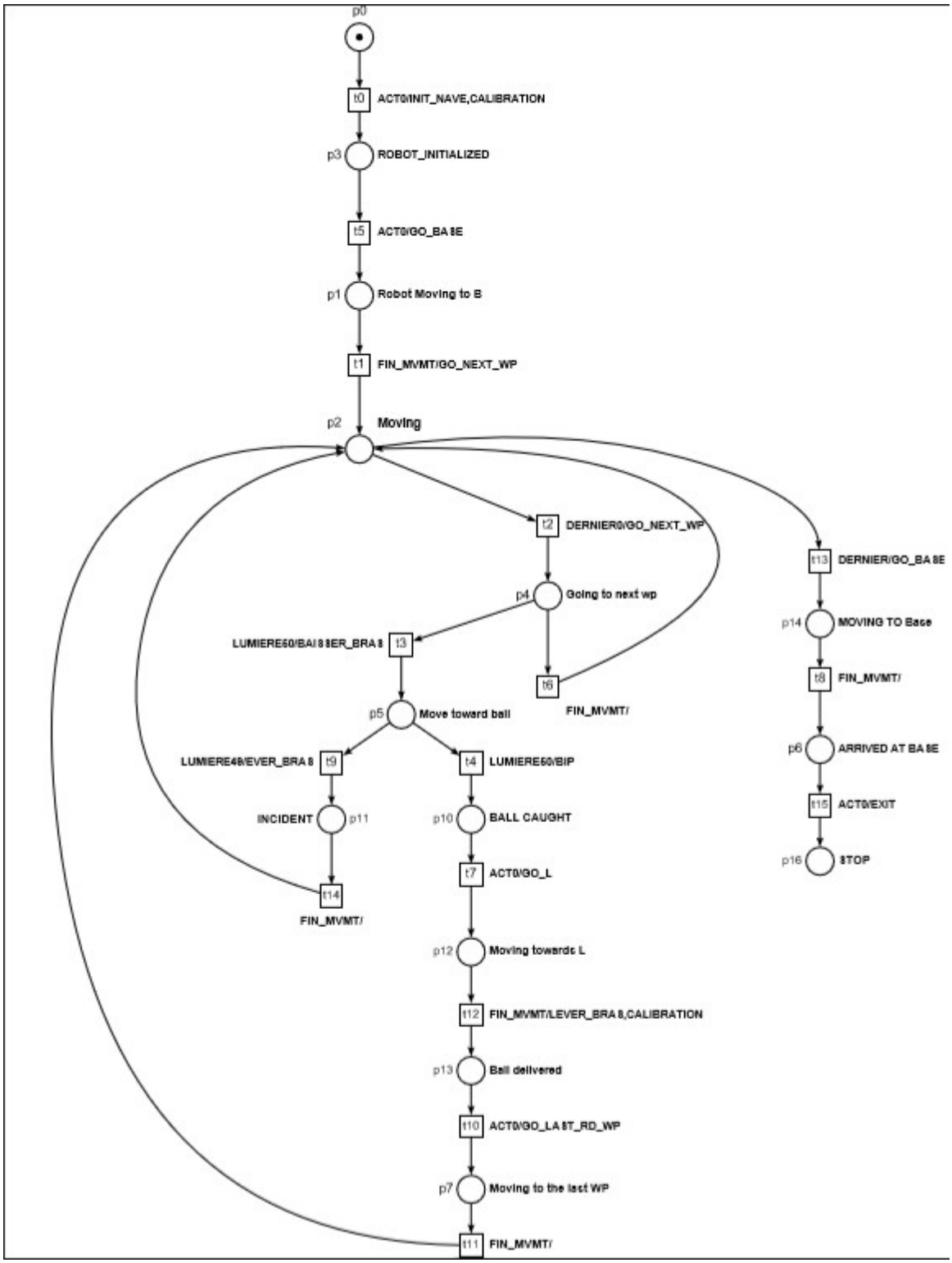

FIGURE 2: ROBOT MODEL

Once the PetriNet model is run successfully then using the ***'textify'*** option in Tina a text file of the graphical representation is created. The generated file is in *'xxx.net'* format.

```
File  Edit  View  Tools  Help
tr t1 : {FIN_MVMT/GO_NEXT_WP} [0,w[ p1 -> p2
tr t3 : {LUMIERE50/BAISSER_BRAS} [0,w[ p4 -> p5
tr t4 : {LUMIERE50/BIP} [0,w[ p5 -> p10
tr t9 : {LUMIERE49/EVER_BRAS} [0,w[ p5 -> p11
tr t2 : {DERNIER0/GO_NEXT_WP} [0,w[ p2 -> p4
tr t7 : {ACT0/GO_L} [0,w[ p10 -> p12
tr t12 : {FIN_MVMT/LEVER_BRAS,CALIBRATION} [0,w[ p12 -> p13
tr t13 : {DERNIER/GO_BASE} [0,w[ p2 -> p14
tr t0 : {ACT0/INIT_NAVE,CALIBRATION} [0,w[ p0 -> p3
tr t5 : {ACT0/GO_BASE} [0,w[ p3 -> p1
tr t15 : {ACT0/EXIT} [0,w[ p6 -> p16
tr t11 : {FIN_MVMT/} [0,w[ p7 -> p2
tr t8 : {FIN_MVMT/} [0,w[ p14 -> p6
tr t10 : {ACT0/GO_LAST_RD_WP} [0,w[ p13 -> p7
tr t14 : {FIN_MVMT/} [0,w[ p11 -> p2
tr t6 : {FIN_MVMT/} [0,w[ p4 -> p2
pl p1 : {Robot Moving to B}
pl p2 : Moving
pl p5 : {Move toward ball}
pl p4 : {Going to next wp}
pl p10 : {BALL CAUGHT}
pl p11 : INCIDENT
pl p12 : {Moving towards L}
pl p13 : {Ball delivered}
pl p14 : {MOVING TO Base}
pl p0 (1)
pl p3 : ROBOT_INITIALIZED
pl p6 : {ARRIVED AT BASE}
pl p16 : STOP
pl p7 : {Moving to the last WP}
net ROBOT50FINAL
```

FIGURE 3: TEXT FILE VERSION OF THE DESIGNED MODEL

This *'xxx.net'* file is given as an input to the **PetriNXT** Tool that will convert the **PN model** into **IPN model**. Here each condition is given a value and it is mapped to the relevant actions that need to be performed in case the condition is satisfied.

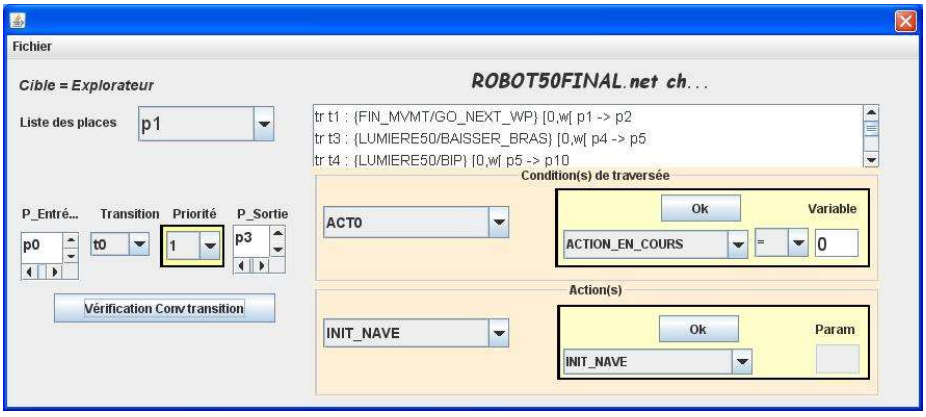

FIGURE 4: CONVERT THE PN MODEL INTO IPN MODEL

The PetriNXT tool generates an *'xxx.rdp'* file.

In order to check the designed model is functioning as expected a simulator is used wherethe behavior of the developed designed is evaluated for different scenarios.

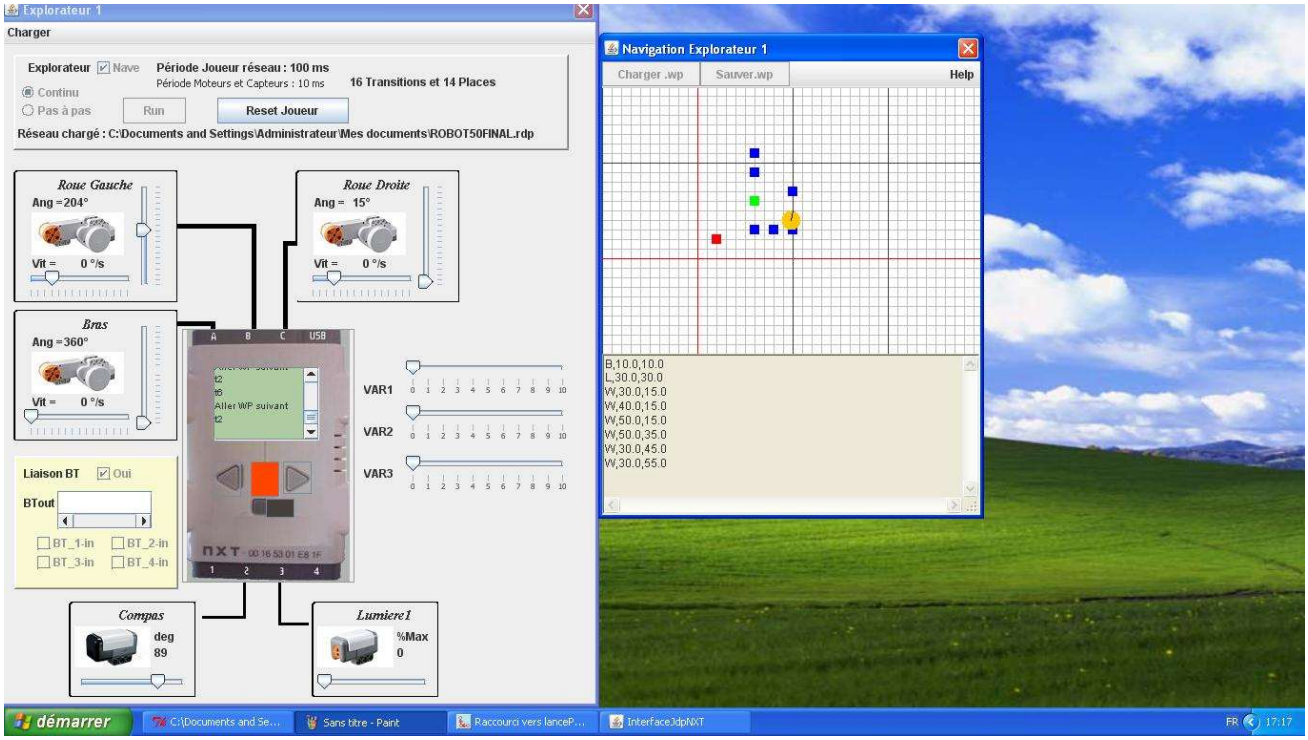

FIGURE 5: SIMULATION FOR THE ROBOT

After successful simulation the *'xxx.rdp'* file can be loaded into the NXT's memory and hence forth be executed.

2 Reachability Analysis– [Single Robot Model]

The Reachability Analysis of the developed Model is obtained from Tina, which confirms that the Model is **Bounded**, **Live** and **Reversible**. The below screenshot has been captured to indicate the same.

The detailed report that was generated during the analysis has been attached for further reference.

Live **->** Our model is deadlock-free

Reversible **->** Our model can go back to its initial state from any marking

Bounded **->** There is a maximal number of tokens for all markings

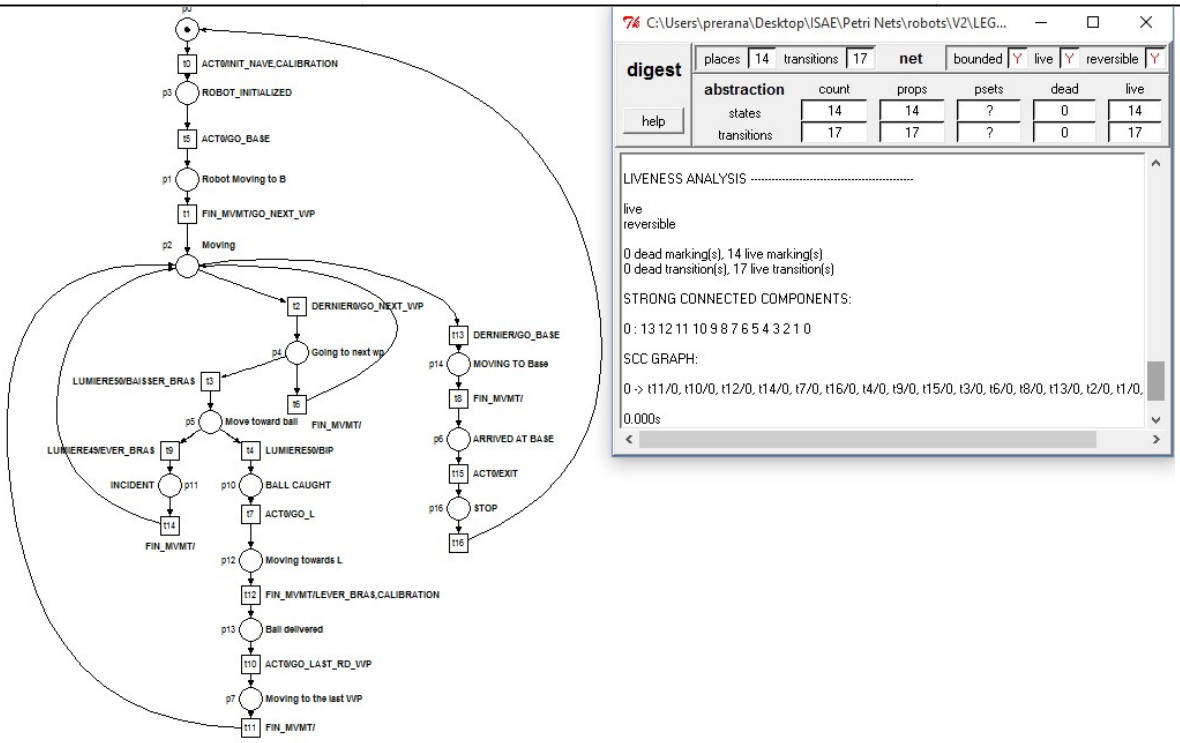

FIGURE 6: REACHABILITY ANALYSIS OF THE ROBOT MODEL

3 Challenges Encountered– [Single Robot Model]

There were few interesting challenges encountered in the development of this Model. The changes that have been adapted are mentioned below:

1. The initial design that was developed did not include the condition *'FIN_MVMT'* (this condition indicates that navigation is finished). Due to this itwas difficult to identify if the Robot has finished navigating and caused error during the simulation
2. The condition *'PRISE'* (this condition indicates that the ball was caught and digger was lowered) was used in the initial design to detect that the ball was caught. But during the simulation process we could not use this signal to test our model as the tool did not support this option. Therefore the condition *'LUMIERE1'*was used instead to simulate that a ball has been detected on the path.

4 Conclusion– [Single Robot Model]

The Robot model was designed and successfully simulated and executed on to the hardware. For the purposes of visualization a video of the execution is taken.

The ***EMS_Lego_Roland_Prerana.zip*** contains the files that were generated and used for developing this model along with the simulation video.

List of files included is as given below:

- Node.wp – waypoints that were configured
- Tina Reversibility Check
- .nrd, .net, rdp, .txt files that were generated during the execution of this project
- A video that shows the simulation results for the developed model

5 Model Description – [Co-operative Robots]

Design Description:

The Main Aim of this project is to program two Robots that can communicate between each other via Bluetooth. The design for Robot-1 '*R1*' remains the same except after delivering the ball at 'L' it shall send a Bluetooth message to Robot-2 '*R2'*. *'R2'*will wait for the message from *'R1'* and then collect the ball from location *'L'* and deliver the same at its base location *'B2'*. This way parallelism is achieved and two Robots communicate with each other.

The guidelines for the program are as follows:

- The Robot-1 and Robot-2 is initialized at *'I1 (0,0)*' and *'I2 (0,0)*'location respectively, using the command *'INIT_NAVE'*, that
  - Calibrates the compass
  - Initializes the navigation (all coordinates are set to 0)
  - Reads the *'node.wp'* file (contains the list of pre-defined way points)
- *'ETALON_CAPT_LUMIERE'* command is used to calibrate the light sensor
- The Robots then moves to their Base location when the calibration is done
- Once the Robot-1 is ready, it moves from one way point to the other as defined in *'node.wp'* file
  - If the light sensor value increases above or equal to 50, it indicates that the Robot-1 has encountered a ball along the path
  - The Robot-1 then moves closer to the ball and lowers the digger to catch the ball
  - If the Robot-1 is unsuccessful in catching the ball then it just moves to next way point and rise up the digger
  - If the Robot-1 is successful in catching the ball, it then delivers the ball to a location *'L'* (the delivery point)
  - At location *'L'* the Robot rises up its digger and releases the ball. The Robot-1 additionally now broadcasts a Bluetooth message ('*ENVOI_BT*-1') indicating that the ball has been delivered to location *'L'* and then Robot-1 returns to the last travelled way point, to continue its journey
  - When Robot-2 receives the message '*BT_1 as true'*then it moves to location *'L'* its light sensor values increases more than 50 as it approaches the ball
  - Robot-2 picks up the ball and then delivers it to the Base-2 location
  - Once the Robot-2 delivers the ball to Base-2 location, it sends out a Bluetooth message to Robot-1 indicating that the location *'L'* is now free
  - The above described steps are followed until the last way point is reached
- Once the Robot-1 reaches the last waypoint, it broadcasts another Bluetooth message ('*ENVOI_BT*-3') indicating that all way points are covered and that the Robot will stop
- Robot-2 stops its execution once a *'BT_3'*is received

To accomplish this, we have one Tina model for each Robot. Robot R2 behavior is rather simple, it starts by going from its initial point to its base and then sends a message (BT_2) to R1. At base, R2 will wait for the messages coming from R1:

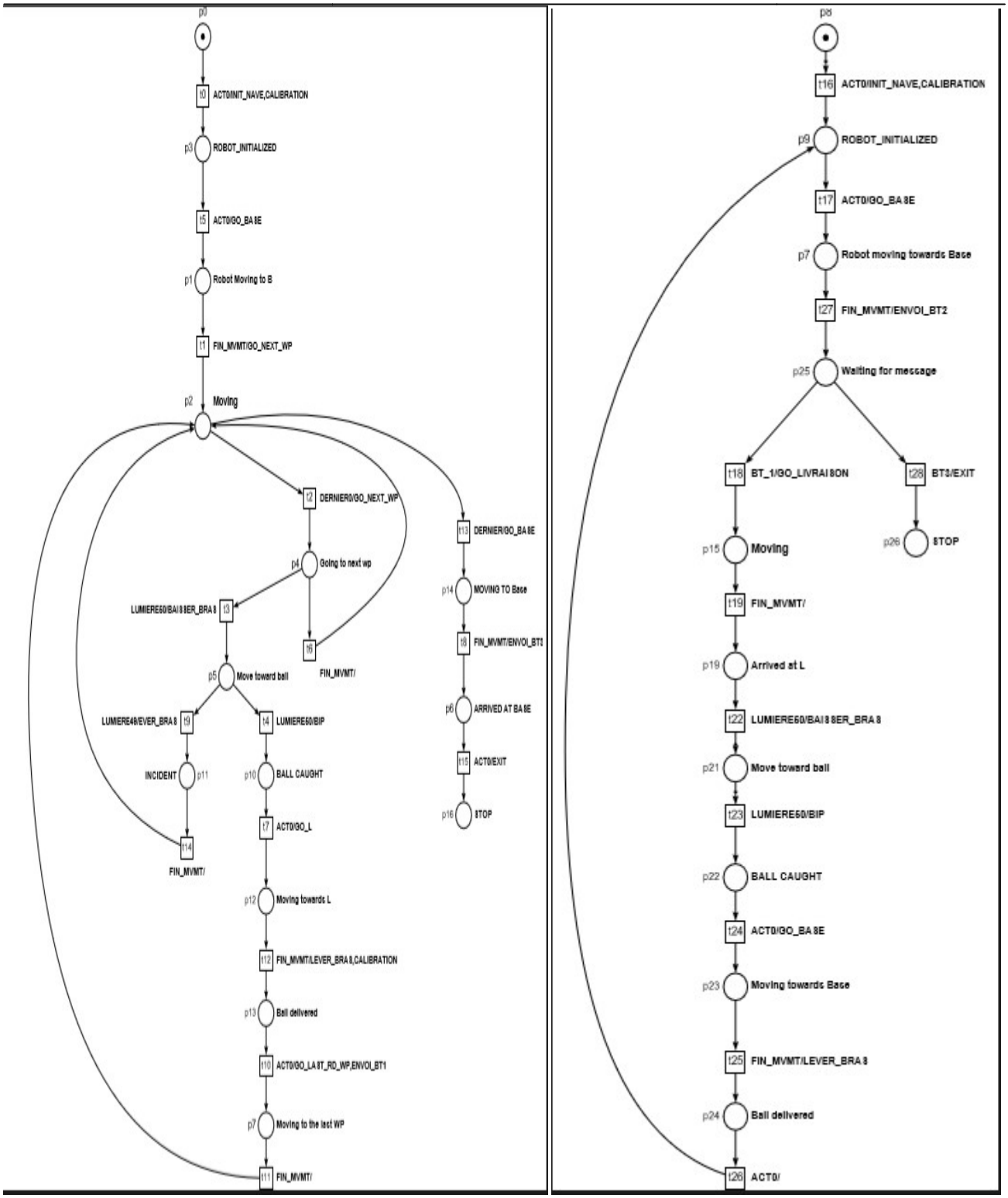

FIGURE 7: TINA MODELS FOR CO-OPERATIVE ROBOTS

[ROBOT-1 AND ROBOT-2 RESPECTIVELY]

To implement this, 2 Tina Models are developed. One for Robot R1 and the other for Robot R2.Similar to the previously explained process, the two models are separately configured and mapped to generate the *'xxx.rdp'* files. Finally using the simulator the scenario for both the Robots was executed successfully.

6 Challenges Encountered – [Co-Operative Robot Models]

This part of the project was rather simple to develop as most of the difficulties were already faced in the first part. There is only one small thing that we had come across, that is:

> The Robot-2 model design returns the ball at base-2 location as there is no command to direct the Robot-2 to go back to the initial location *'I-2 (0,0)'*. We tried simulating a default way point with the co-ordinates (0,0), but as there is no command that can direct the Robot to a particular way point it was difficult to go ahead with the plan.

7 Conclusion– [Co-Operative Robot Models]

The Robot model was designed and successfully simulated. For the purposes of visualization a video of the execution is also taken.

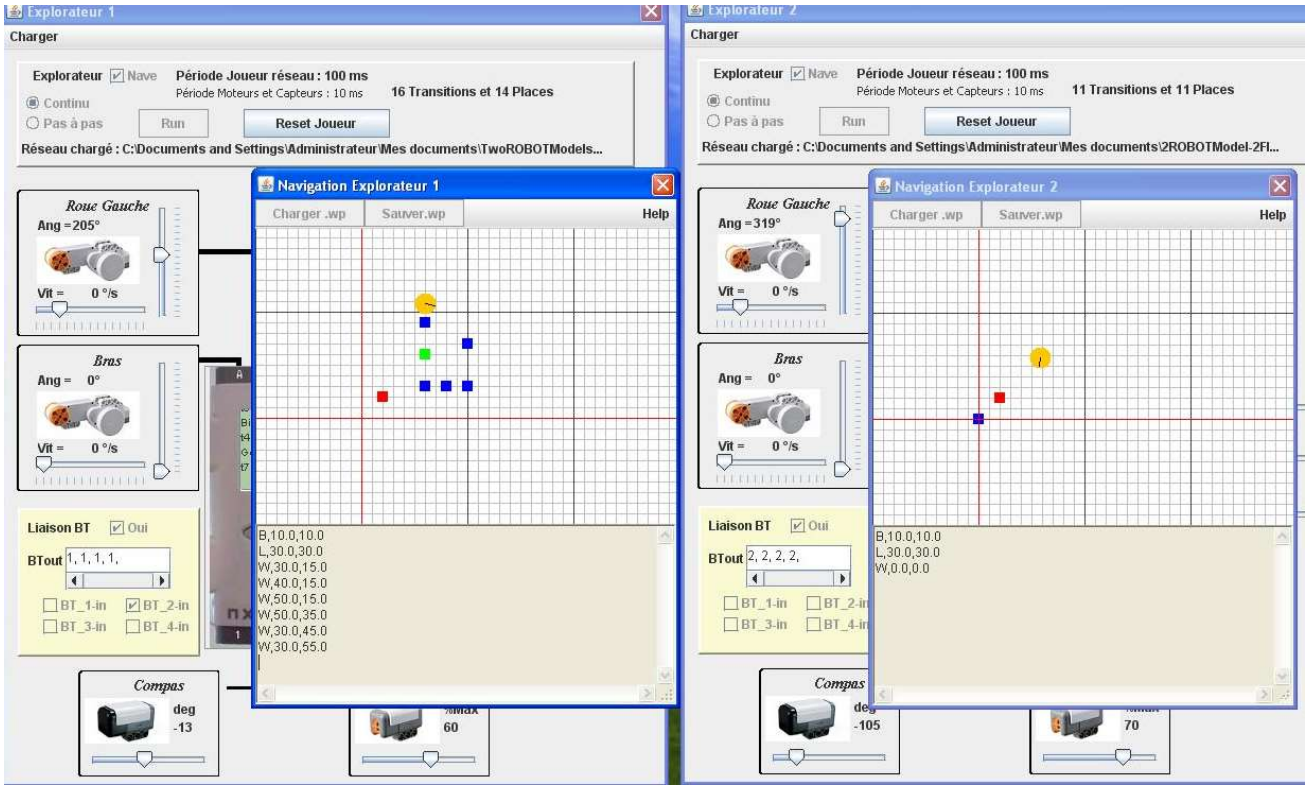

FIGURE 8: SIMULATION FOR CO-OPERATIVE ROBOTS

The ***EMS_Lego_Roland_Prerana.zip*** folder contains the files that were generated and used for developing this model along with the simulation video.

Video 1: https://www.dropbox.com/s/i7kq6j0d9pg2089/SimulationForSingleRobot.mp4?dl=0

Video 2: https://www.dropbox.com/s/8djj92uls2dfcry/Part2-Simulation-CooperativeModels.mp4?dl=0

List of files included is as given below:

- Node_2.wp – way points that were configured
- .nrd, .net, .rdp, .txt files that were generated during the execution of this project
- A video that shows the simulation results for the developed model