Introduction

The IEC 61499 standard is a Distributed Industrial Process Measurement and Control Systems modelling language. Mainly IEC 61499 standard gives a general model for distributed system. This standard ensures reusability, portability, reconfigurability, interoperability for a distributed automation system. The Function Block Development Kit(FBDK) software provides the resource for building a distributed system according to the IEC 61499 standard.

Objectives

The objective of this assignment is to

- Learn the nature of industrial automation systems
- Build distributed systems according to IEC 61499 standard
- Learn centralized to decentralized control
- Design system at high level
- Use object-oriented development methodology
- Use UML Modelling tools
- Apply Model-View-Controller design pattern
- Learn Event-driven system architectures
- Components encapsulation (composite FB) and reusability, reconfigurability, interoperability
- Learn about application domains of IEC61499 standard

Requirements

- Function Block Development Kit(FBDK) software
- Methodology related to Model-View-Controller design pattern
- UML Modelling tools

Task 1.a.i. Traffic Light Control system (centralized approach)

Steps:

For developing this system, the following step are taken (in Figure 1). For sketch step the scenario of the traffic lights control system was provided which is shown in Figure 2 with some minor modification (Road3 and Road 4 title is added for simple representation). All other steps are gone through by using the FBDK software package.

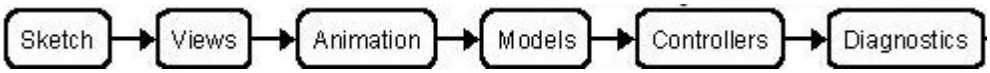

*Figure 1:*Process step

.

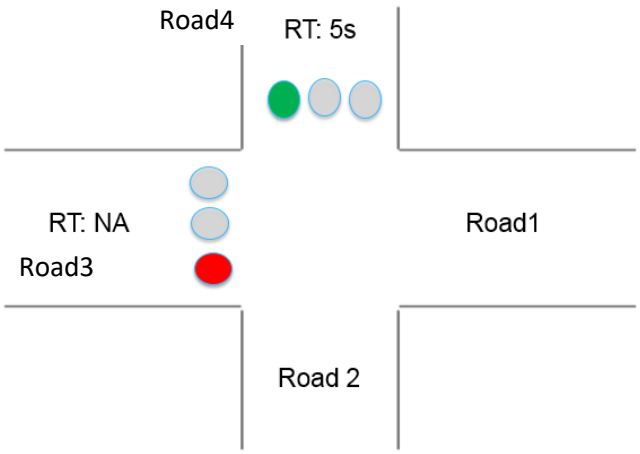

Figure 2. Traffic light system

Architectures for the traffic lights in the centralized design has global HMI, global VIEW and centralize CONTROL (Figure 3).

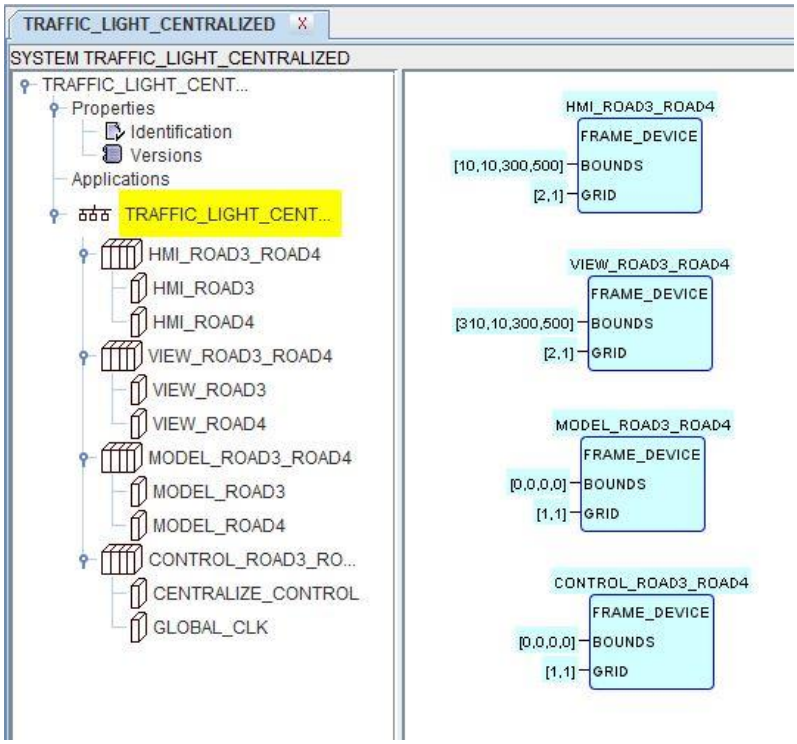

Figure 3: Architecture

View Layer

The view layer is created (Figure 4 and Figure 5) by using OUT_BOOL to show the status of the traffic light.

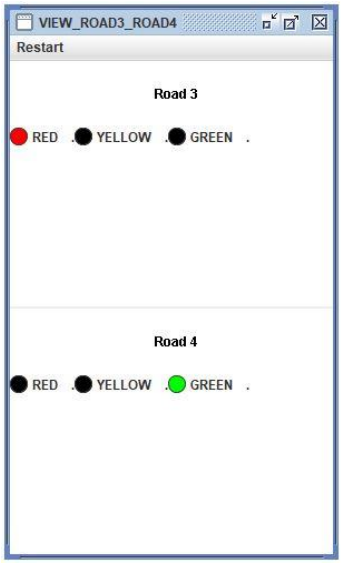

Figure 4: View Layer

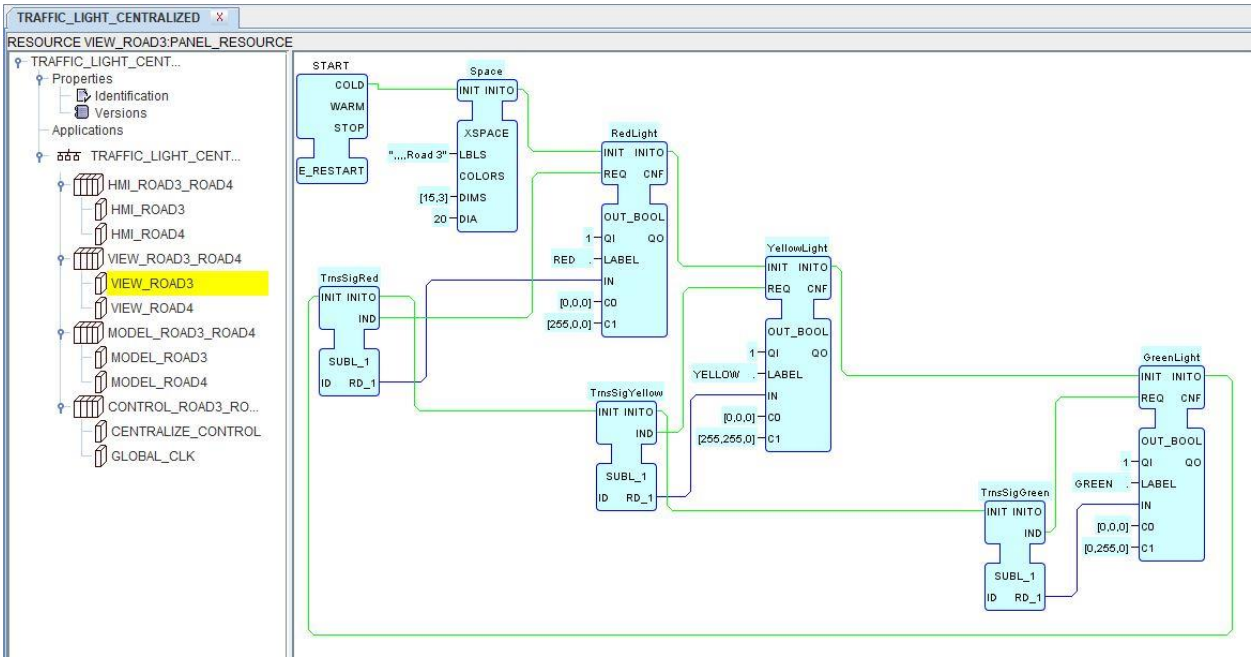

Figure 5: View layer for Road 3

HMI Layer

In HMI layer, OUT_ANY and IN_ANY is used for showing the remaining time and to input the desired on-time for green and yellow light respectively (Figure 6 and Figure 7).

The HMI of this system is designed to:

- Enter the desired on-time for green and yellow light for each road.
- Display the status of each semaphore (green, yellow, red) lights

- Display the remaining time (RT, see Figure 2), in seconds, for the current light. The remaining time is shown only for green and yellow light.

Figure 6:HMI Layer

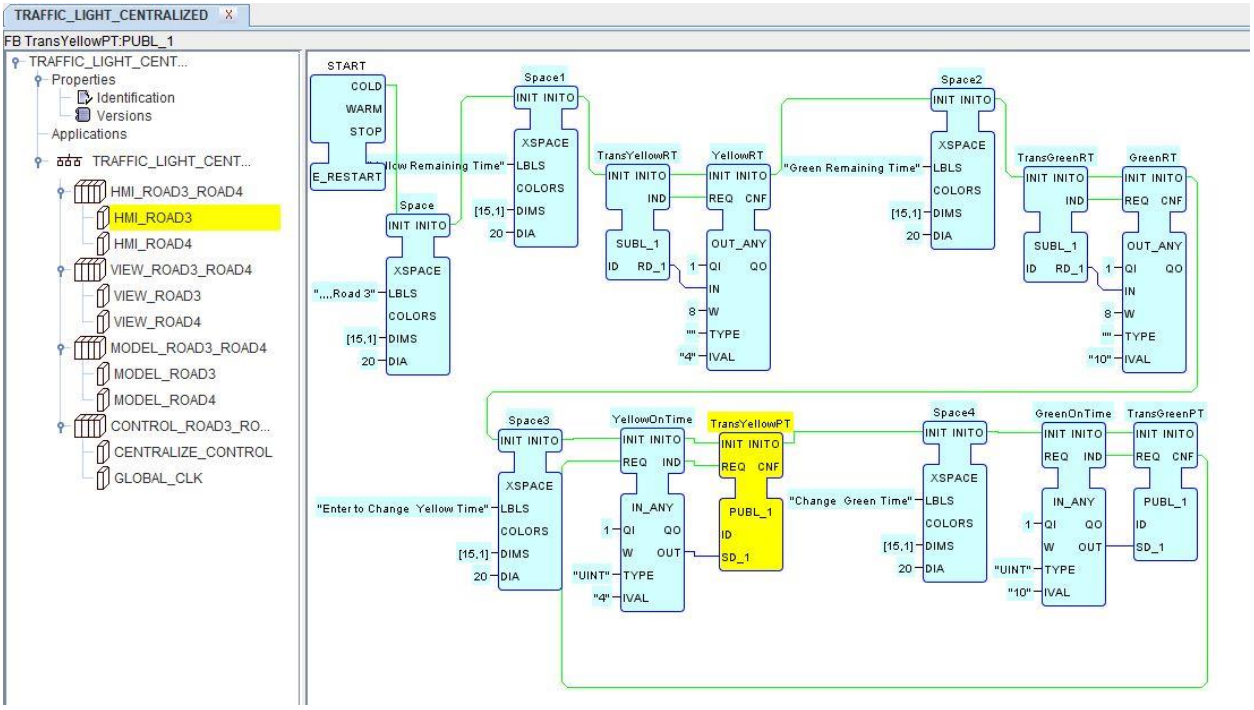

Figure 7: HMI layer for Road 3

Model Layer

In Model layer, a composite block is used. This layer is kept simple because the physical lights only take on/off signals (Figure 8).

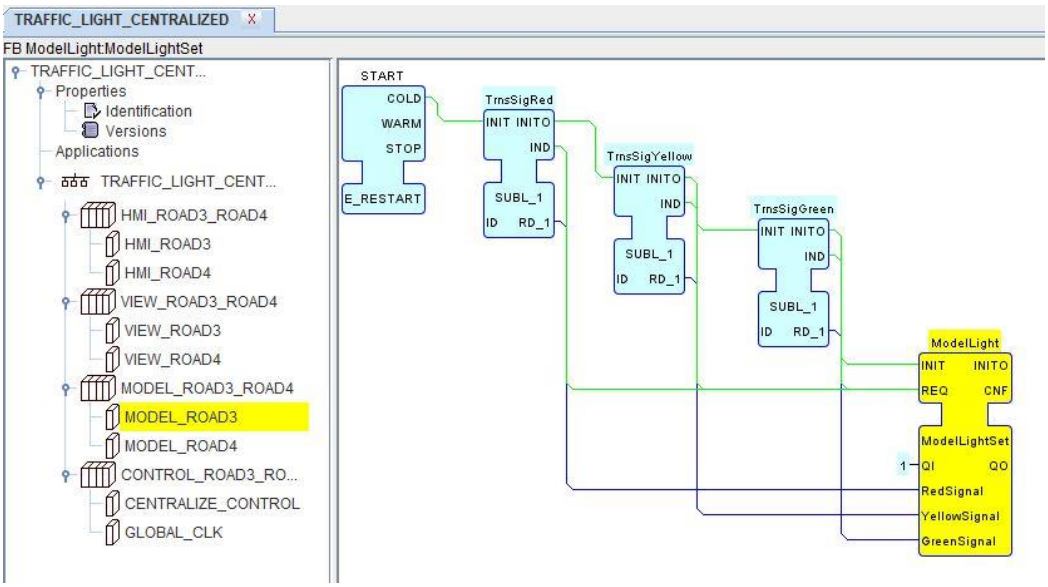

Figure 8: Model layer

Control Layer

For control layer, a basic block named SEMAPHORE_CU is designed to

- take input of the desired on-time for green and yellow lights
- take input of clock pulse
- send out the on/off signal for the red, yellow, green lights
- send out remaining on-time for green and yellow lights
- go through the on-time for green and yellow light, it gives the “control” of the road to the other semaphore.
- be configured with different times for their respective green and yellow lights

The block with created algorithm and the ECC are shown in Figure 9 and Figure 10. The created algorithm are described below:

- For every Semaphore request, it trigger the REQ algorithm, which set the desired on-time for green and yellow light. (Figure: 9(b))
- For every clock pulse, it triggers either RT_GREEN or RT_YELLOW depending upon the grad condition for decreasing the RT value. (Figure: 9(c & d))
- If the RT_YELLOW become zero, then CLK triggers the SEMA_SIG algorithm for turn on the Red light and assign a semaphore signal to give the “control” of the road to the other semaphore. (Figure: 9(e))

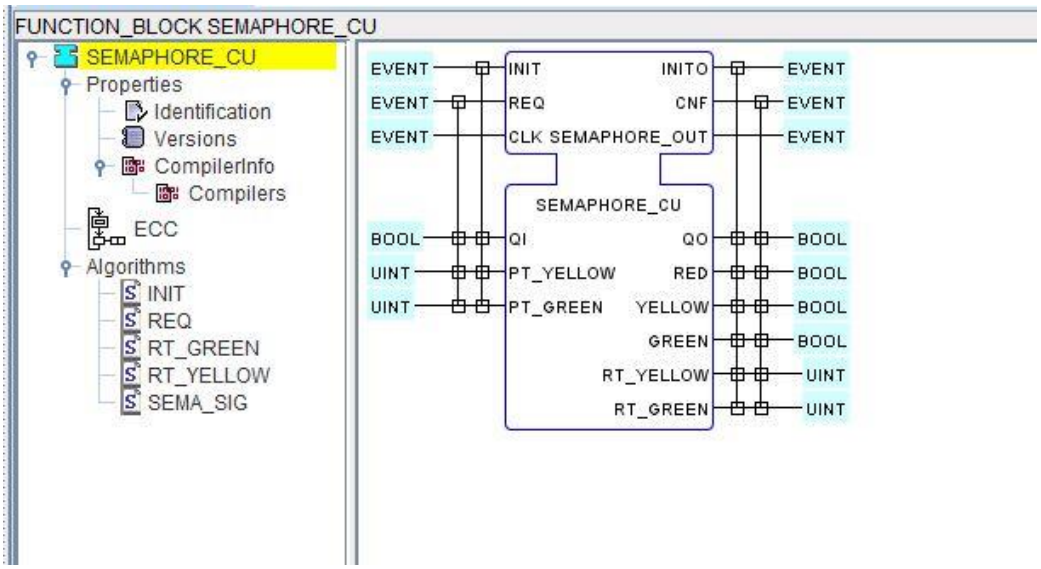

Figure: 9(a): Basic block

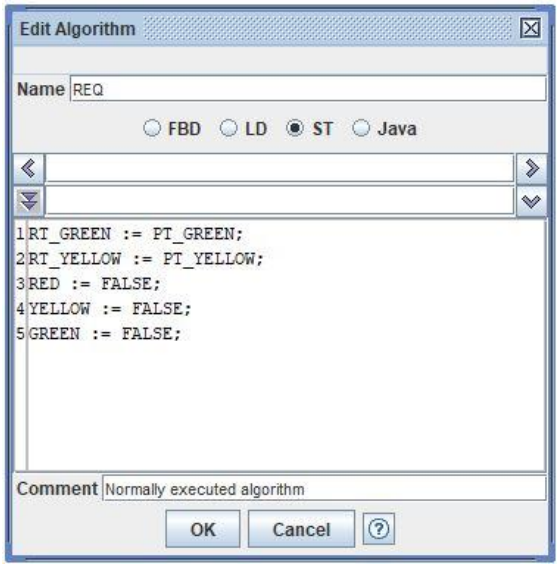

Figure: 9(b): REQ algorithm

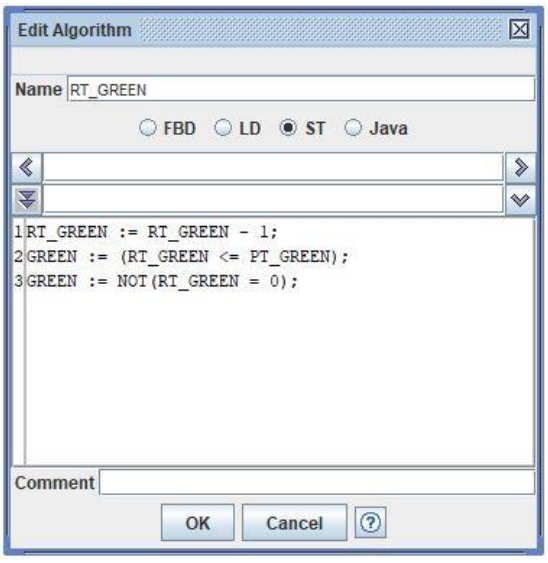

Figure: 9(c): RT_GREEN algorithm

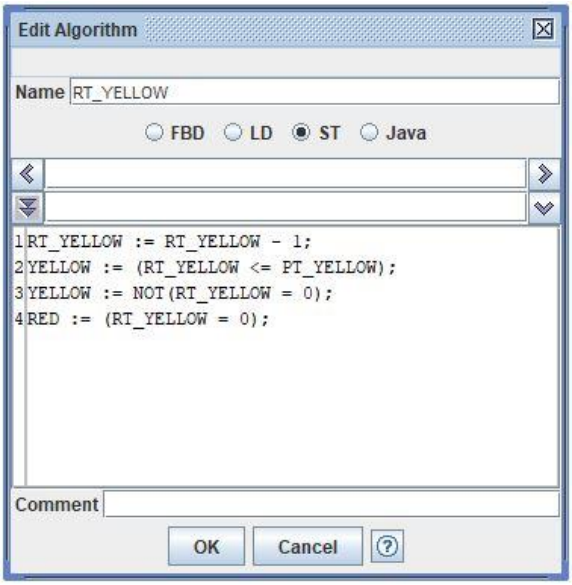

Figure: 9(d): RT_YELLOW algorithm

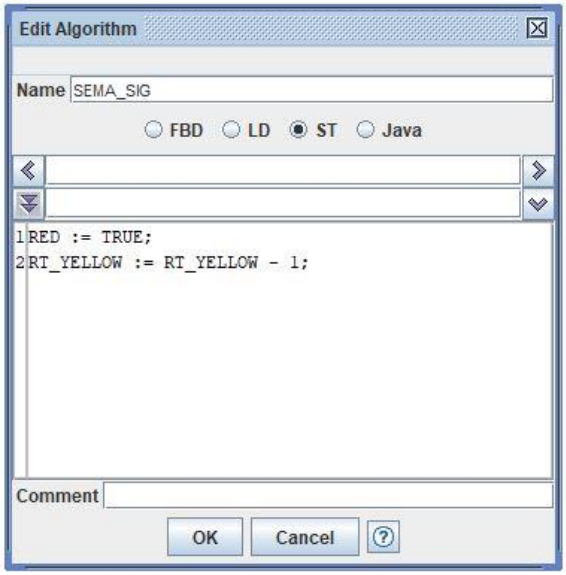

Figure: 9(e): SEMA_SIG algorithm

Figure 9: Basic block: SEMAPHORE_CU

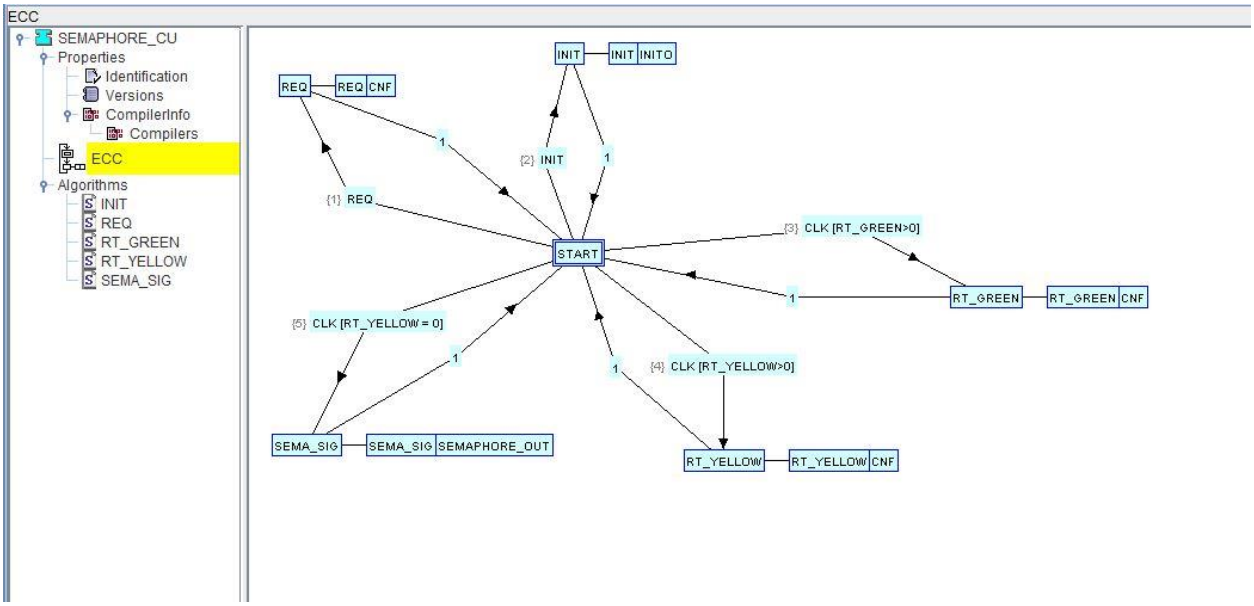

Figure 10: ECC of SEMAPHORE_CU

So, this SEMAPHORE block is used for controlling each road, Semaphore1 for controlling Road3 and Semaphore2 for controlling Road4, shown in Figure 11. For implementing a centralize control, a global clock is used, and no other separated frame device is introduced.

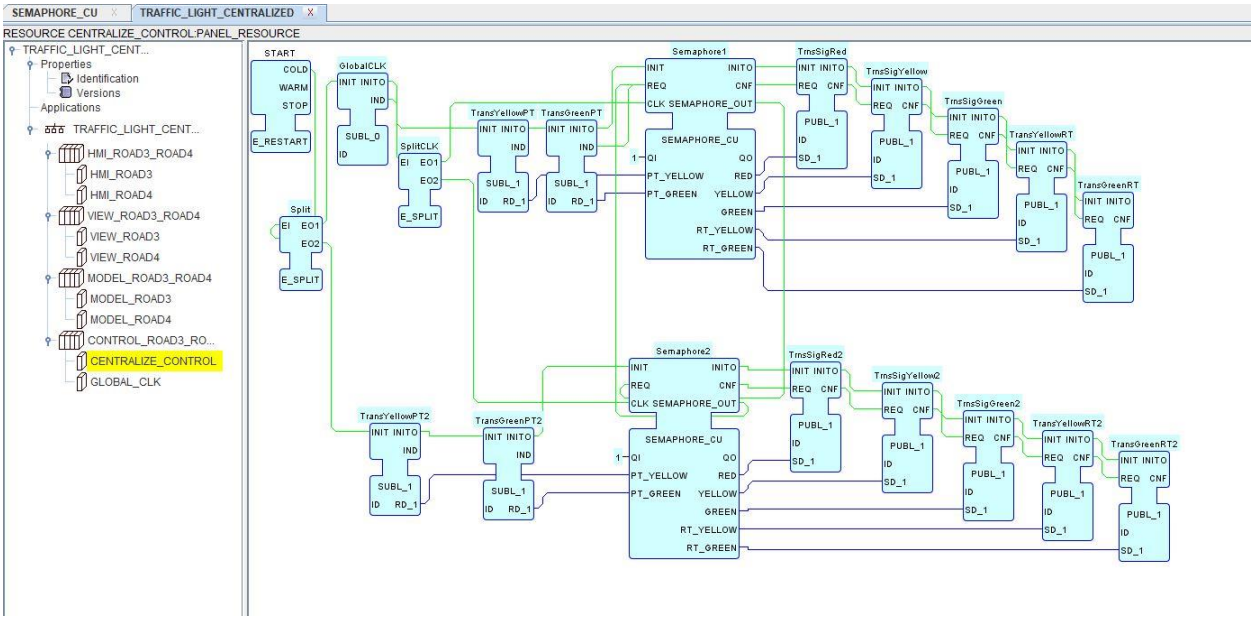

Figure 11: Centralize control layer

ULM Diagram:

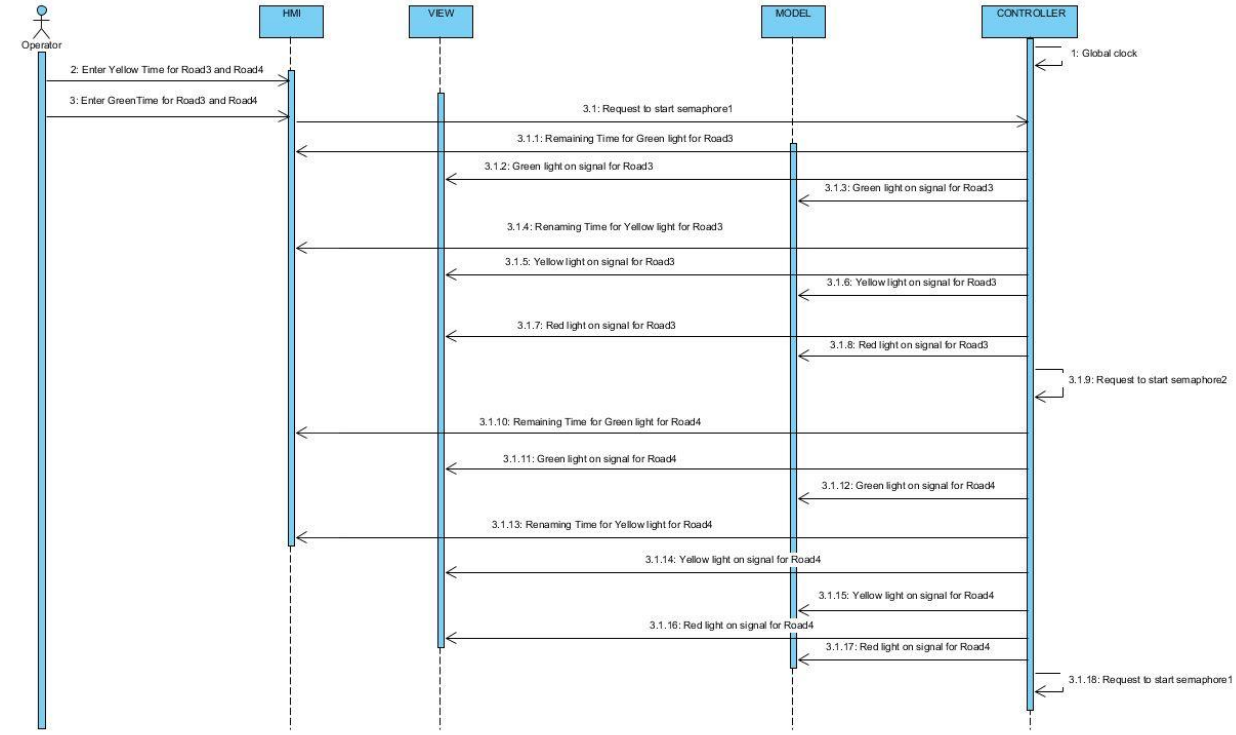

Figure 12: Sequence diagram for Traffic Light Control system (centralized)

Task 1.a. ii. Traffic Light Control system (distributed approach)

Steps

For developing this system, similar steps are taken as in centralized approach. For the traffic lights in the distributed approach design, the option 1 architectures has been used. (Figure 3).

Option 1: HMI and VIEW separated

Road 3:

HMI resource (remaining time, desired on-time), Control resource, Model resource, View resource (light status)

Road 4:

HMI resource (remaining time, desired on-time), Control resource, Model resource, View resource (light status)

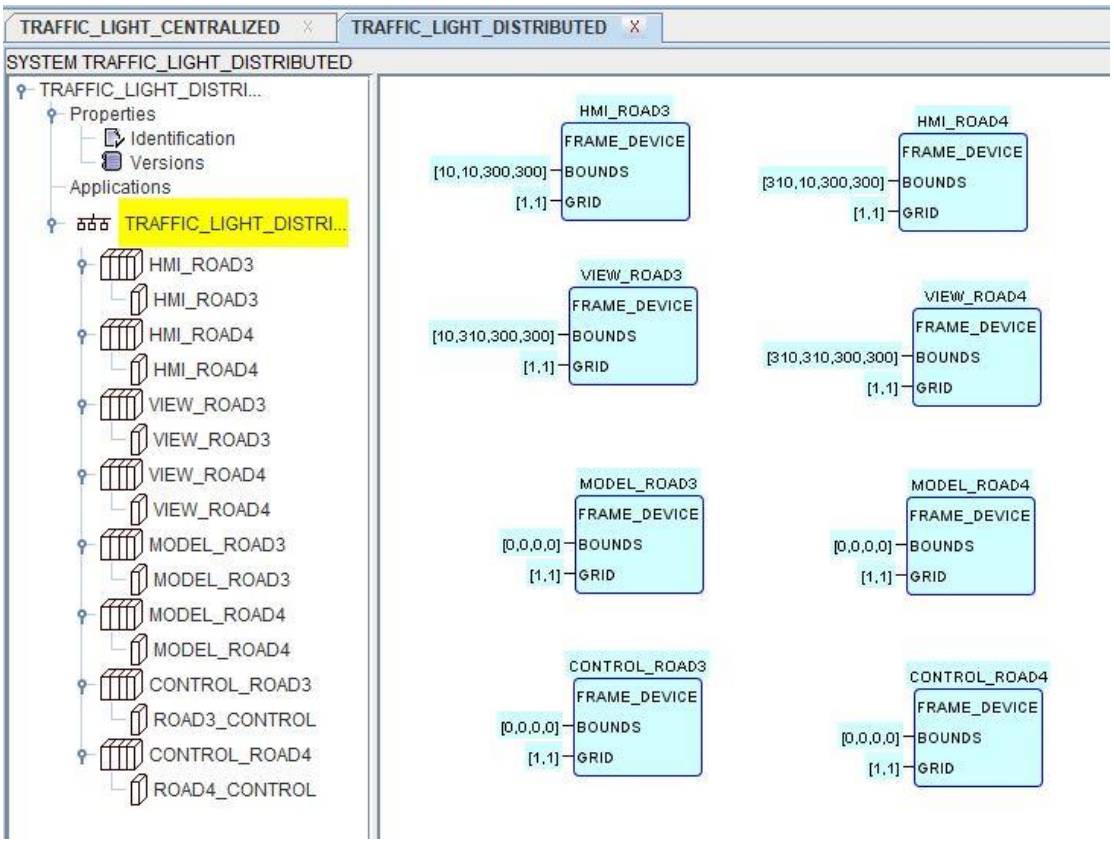

Figure 13: Architecture (distributed approach)

Changes in Layers

Mainly, the following changes are made:

- Introduce separate frame devices for every layer (Figure 13)
- HMI and VIEW layer for each Road is separated (Figure 14)
- Drastical changes are made in CONTROL layer which include adding separate clock for each Semaphore.

HMI and VIEW layer

The separated HMI and VIEW layer are shown in Figure 14.

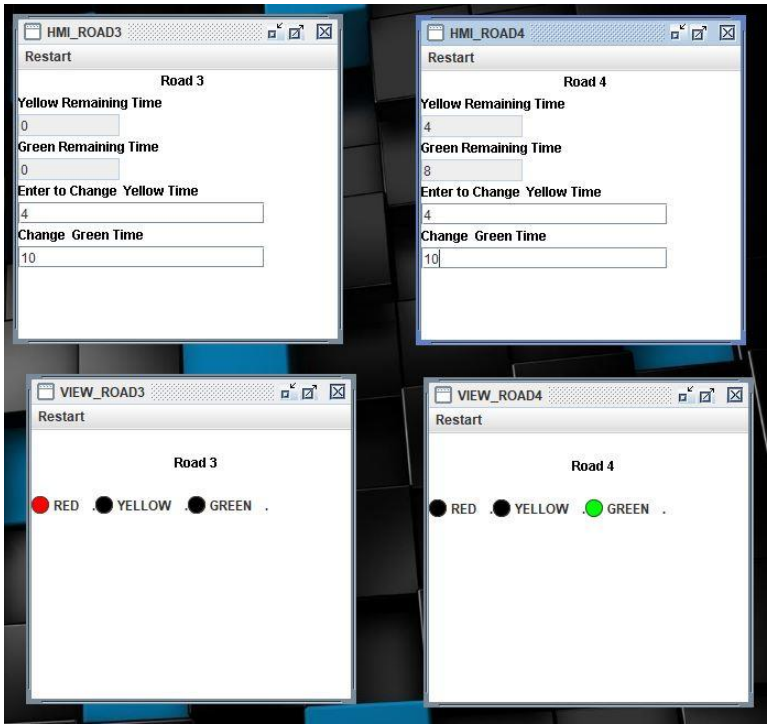

Figure 14: Separated HMI and VIEW layer

CONTROL layer

The SEMAPHORE block is used for controlling each road, Semaphore1 for controlling Road 3 and Semaphore2 for controlling Road 4. For implementing a distributed approach control, separate clock is used is introduced. For example, the Function block diagram for Road 3 is shown in Figure 15.

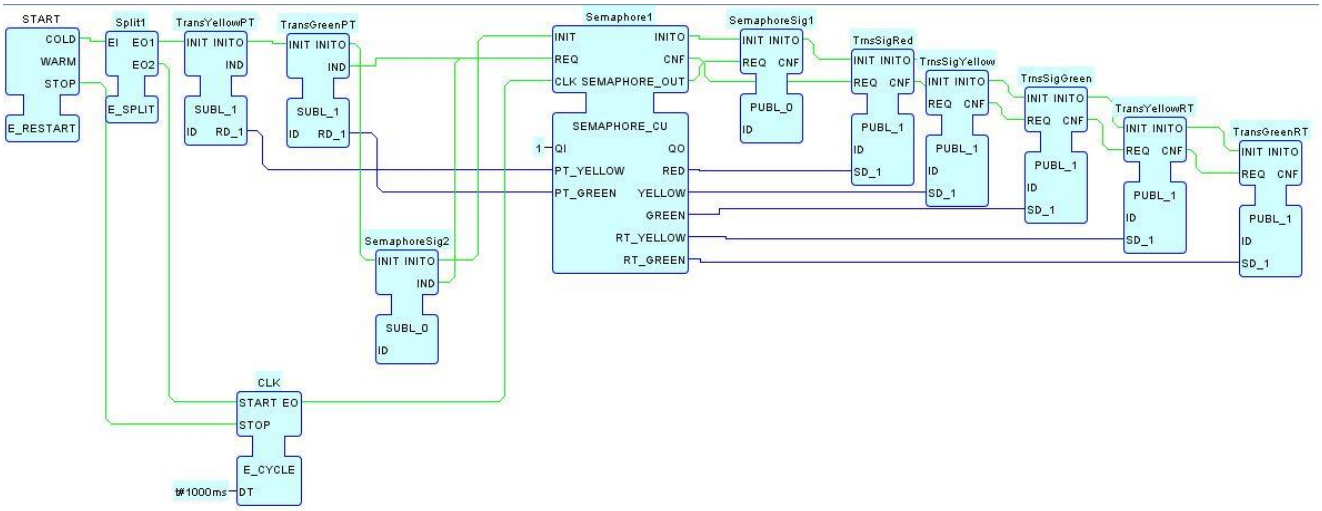

Figure 15:Control layer for Road 3 with own clock

ULM Diagram:

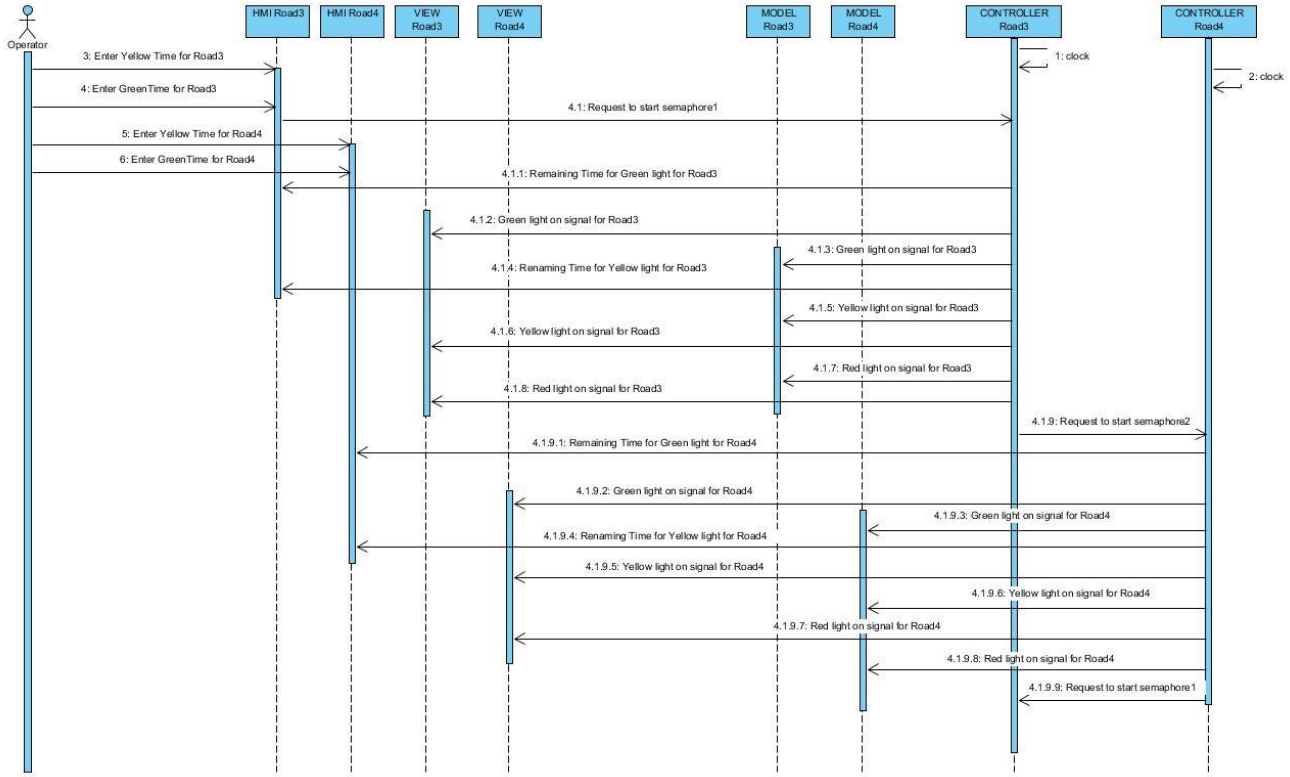

Figure 16: Sequence diagram for Traffic Light Control system (*distributed approach*)

Task 1.b Traffic Light Control system with counter

Steps

For creating this system (Figure 17), the Traffic Light distributed Control system is use as a base system and a pedestrian button is added with mentioned functionality.

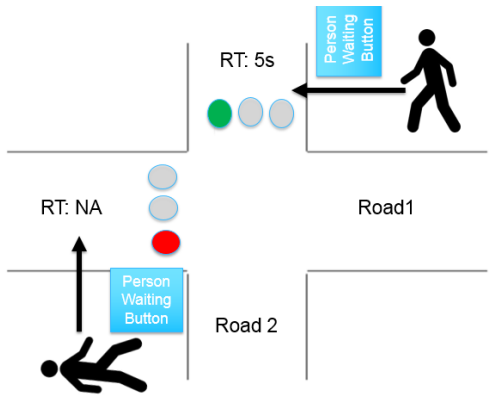

Figure 17. Traffic light system with pedestrian button

Changes in Layers

HMI layer

- Introduce pedestrian button in HMI layer (Figure 18)

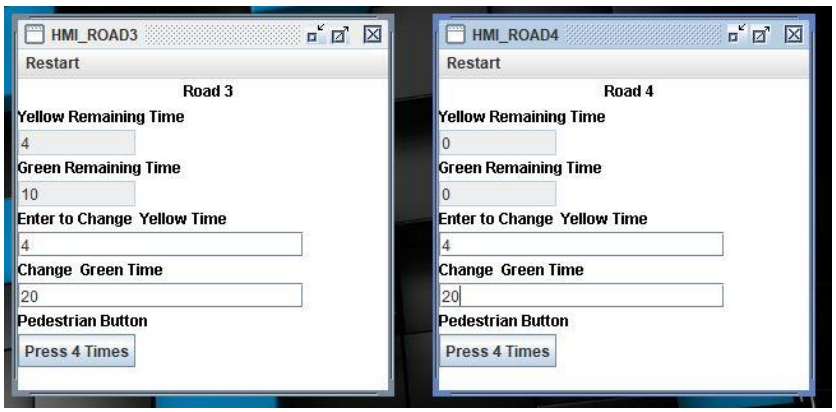

Figure 18: HMI with pedestrian button

Control Layer

- Modification in the designed Semaphore block (which is named as SEMAPHORE_BUTTON):
  - An internal variable is declared named BT_COUNR which count the button press (Figure 19),

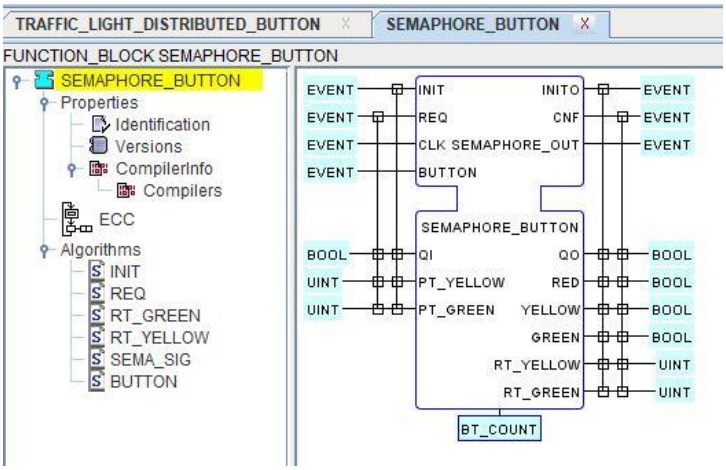

Figure 19: SEMAPHORE_BUTTON

- An algorithm is created, named BUTTON, using ST language, which trigger an event when the button is pressed four times and set the remaining time for the green light to four seconds (Figure 20).

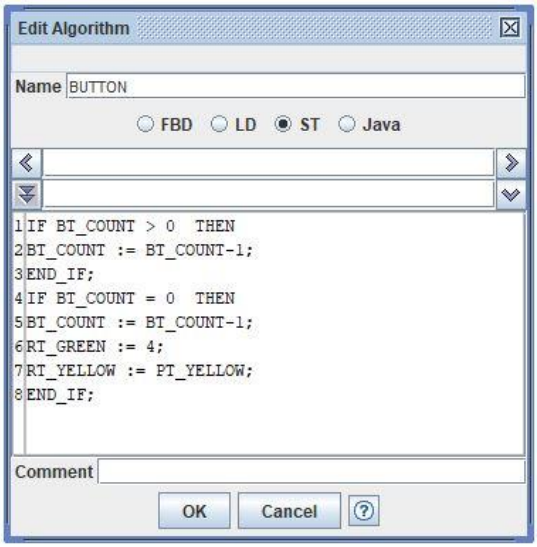

Figure 20: Button algorithm

- Then it switches to yellow and then to red accordingly the previous design (distributed approach).

Ultimately, if the push button has to be pressed four times then the green light has been ON for four seconds before it switches to yellow and then to red. The ECC is shown below:

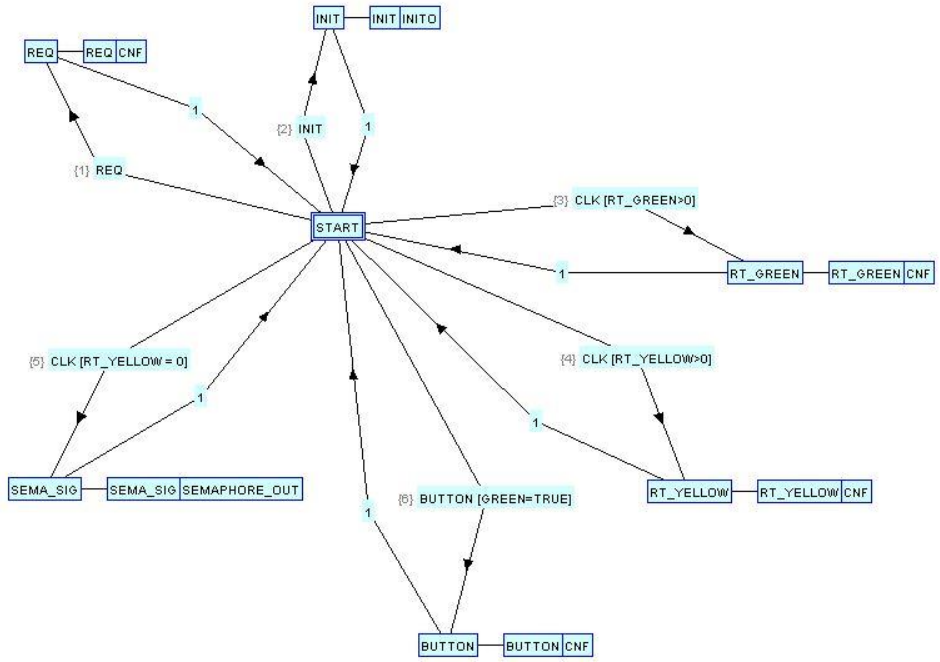

Figure 21: ECC for SEMAPHORE_BUTTON

ULM Diagram:

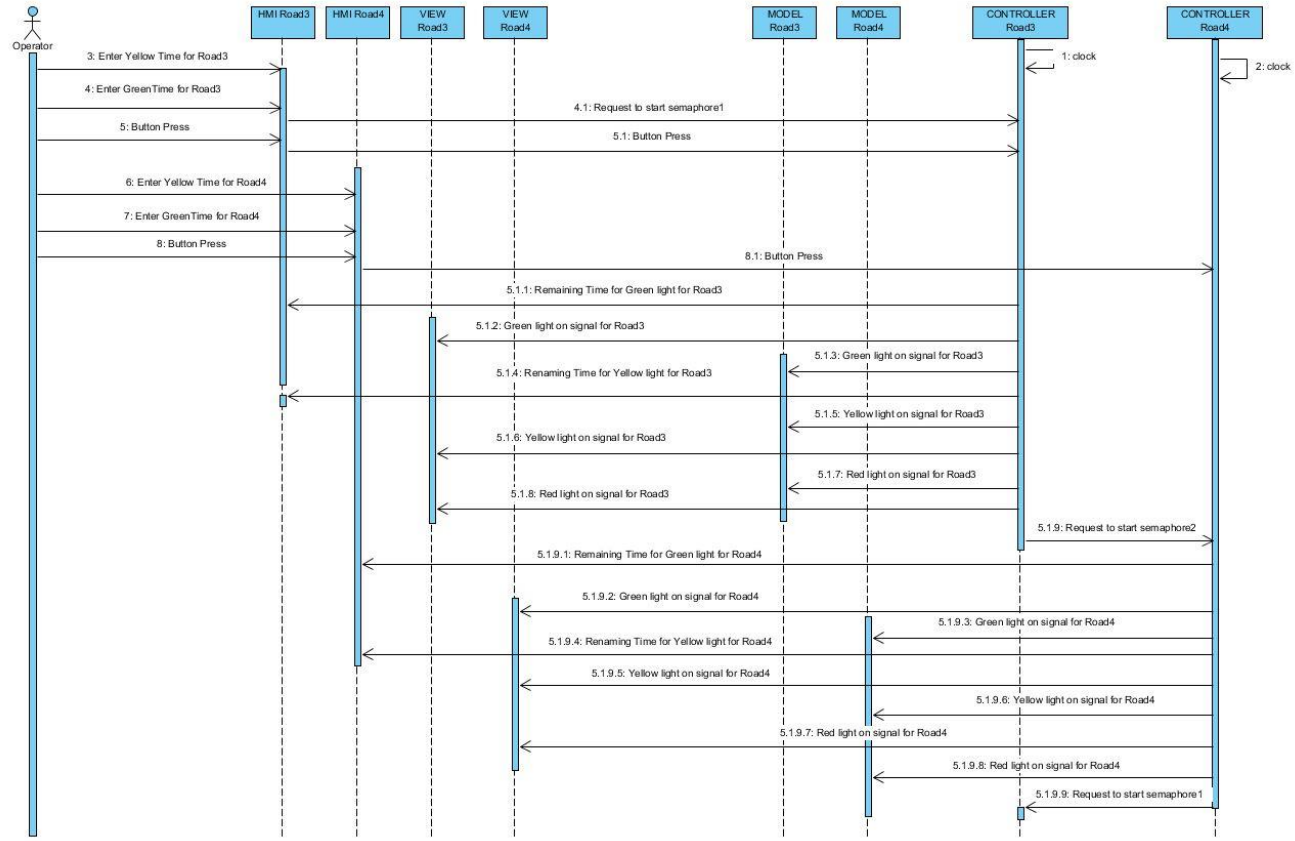

Figure 22: Sequence diagram for Traffic Light Control system (*distributed approach with counter*)

Task 2 Sorting system

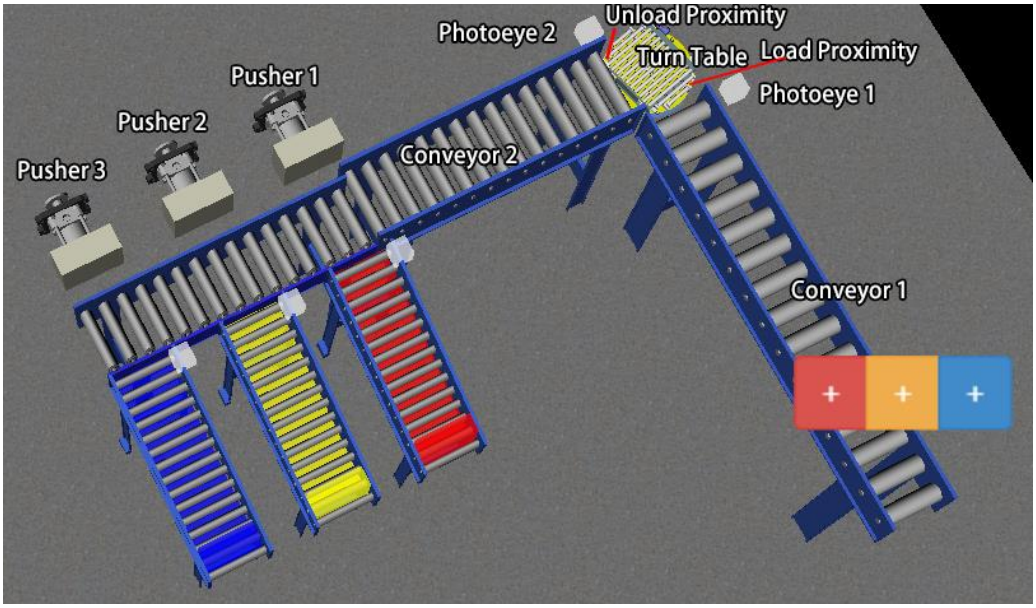

Figure 23 Sorting system

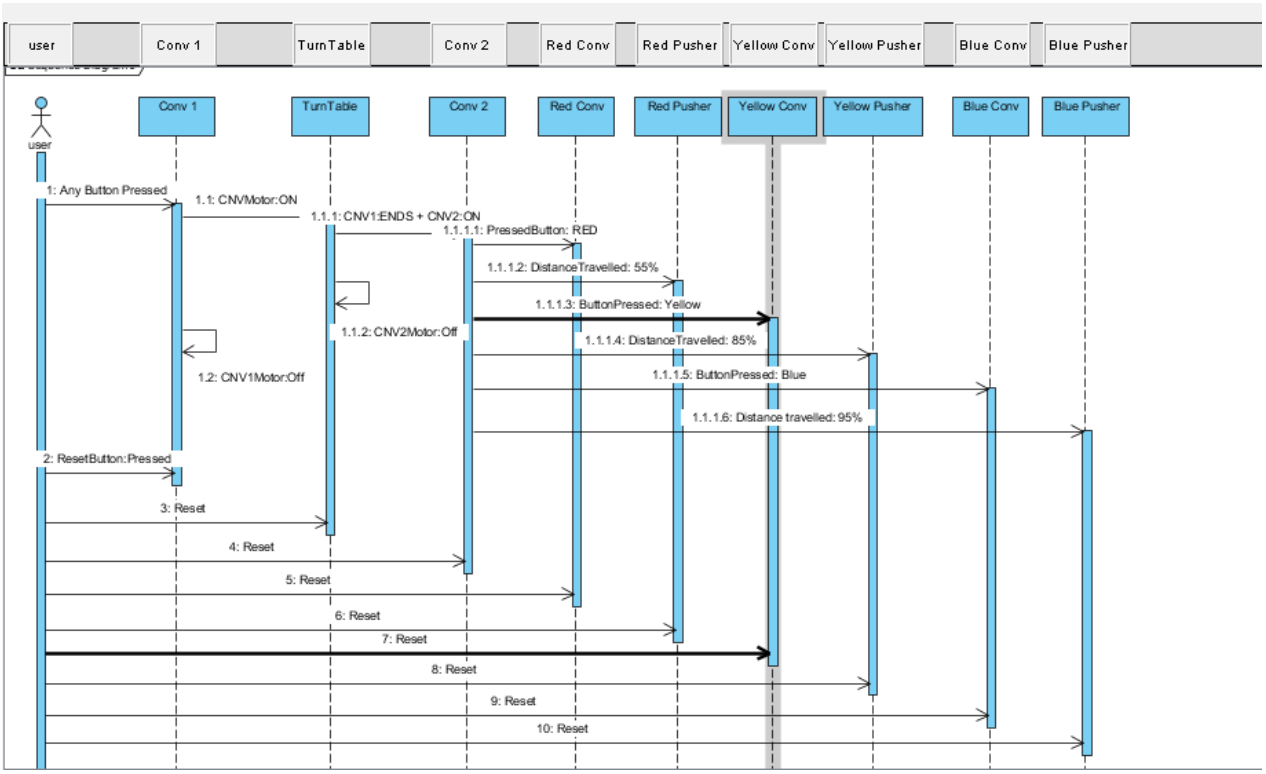

Figure 24: CNV System: Activity flow

IF statement block takes 2 inputs, works as AND gate and generate an event if both values are true. Compare55, Compare85 and Compare95 blocks compare the position of workpiece on cnv2 with a constant number for different colors and unloads the cnv2 accordingly. Distance travelled on the cnv2 is taken as a representation of sensor to sense the presence of right colored WP in front of pusher. As soon as right colored WP reaches in front of pusher, pusher pushes it to the next CNV and unloads the CNV2. Reset button, or any button to load the WP, clears the set values for all other logics.

Devices:

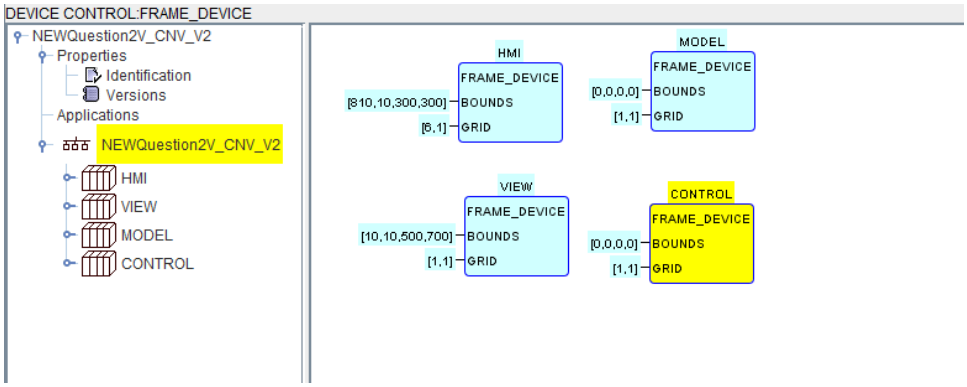

Figure 25: Devices

HMI

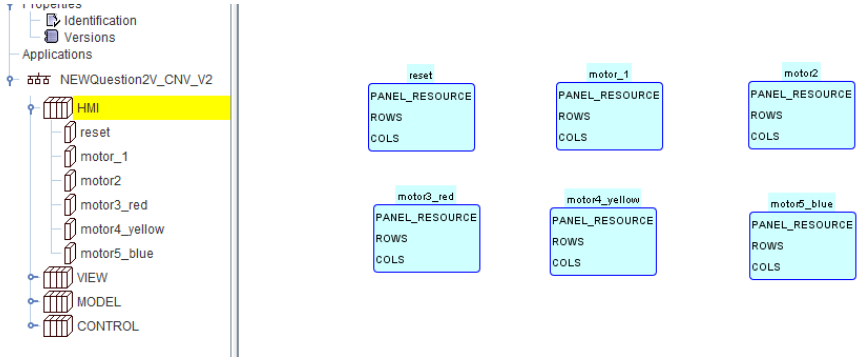

Figure 26: HMI Devices

HMI Device is used

- Moto On/OFF
- Restarting the whole system
- To place a workpiece and defines its color

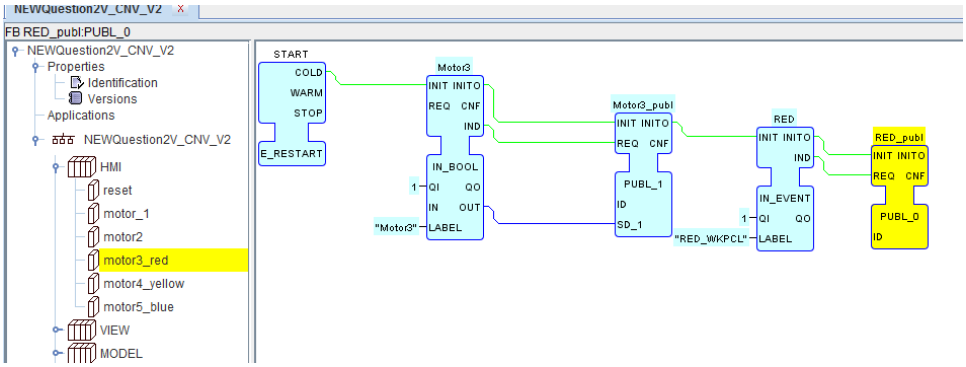

Figure 27: Motor 3 on off control and WP button

The resources inside HMI just have check buttons for motor status and buttons for WP and restarting. And as soon as a button is pushed, it publishes the changes for model device.

MODEL and CONTROL

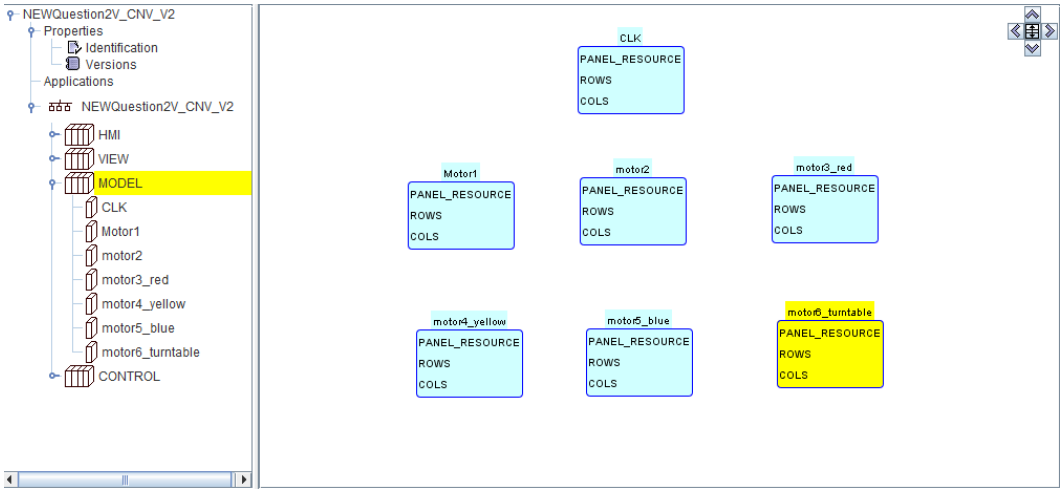

Figure 28: Model Devices

Model device has resources to control motor of conveyors and loading and unloading.

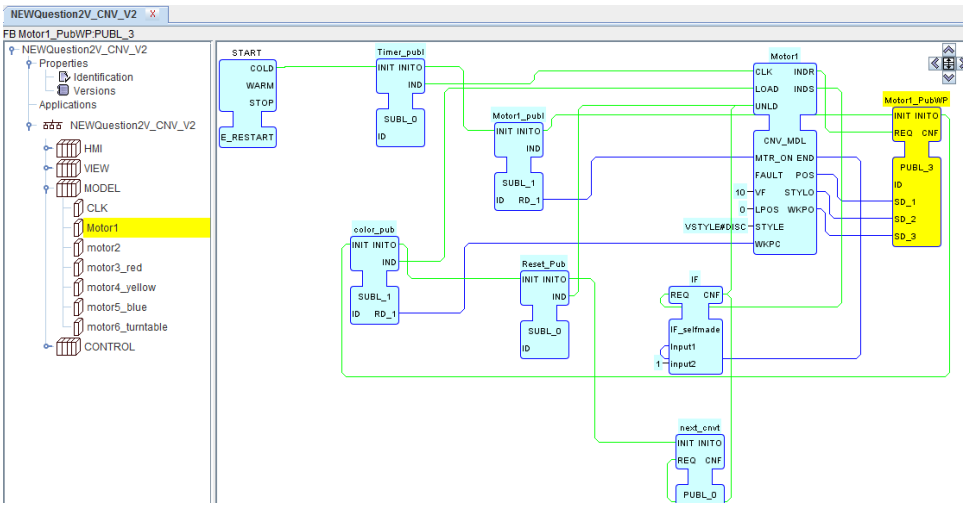

Figure 29: Motor 1 working

In the first conveyor, as soon as a button is pressed, the WP is generated, and its color depends upon the button pushed. The color decision is made in color_control of control device. The decision is then published back for the conveyor 1.

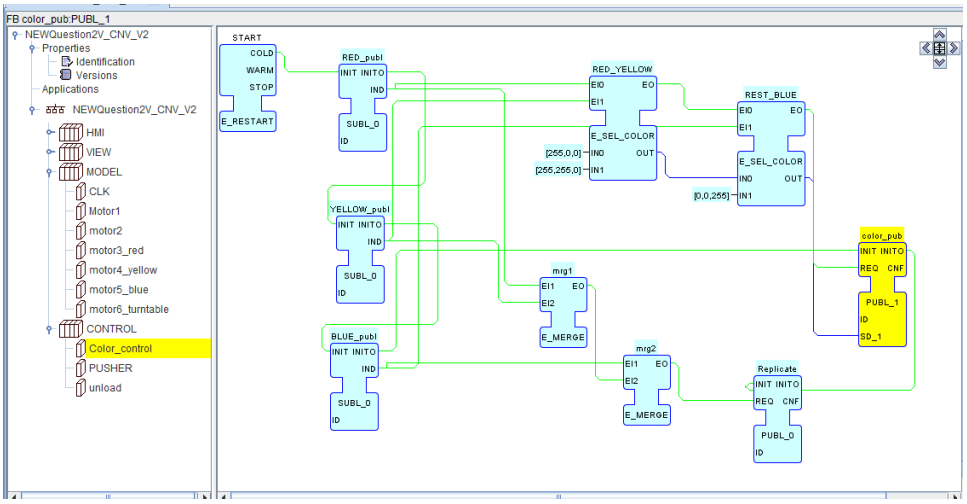

Figure30: WP color selection

As soon as the block reaches the end of the first conveyor, the conveyor is unloaded, and turntable is loaded.

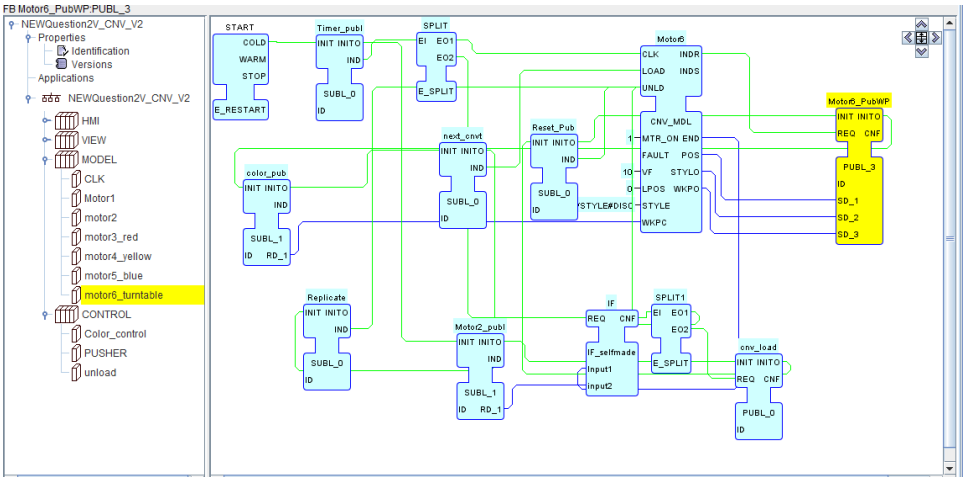

Figure 31: Turntable decision making

When turntable is loaded, it moves devices to the left to the end, and it checks the status of next conveyor, if the next conveyor is on, it then shifts the WP to the conveyor, otherwise waits for the motor2 to be turned on. When conveyor 2 is loaded, the devices starts moving on it to the left. It constantly publishes the position of the device for unload resource in control device. Which is there to make unloading decision for the conveyor.

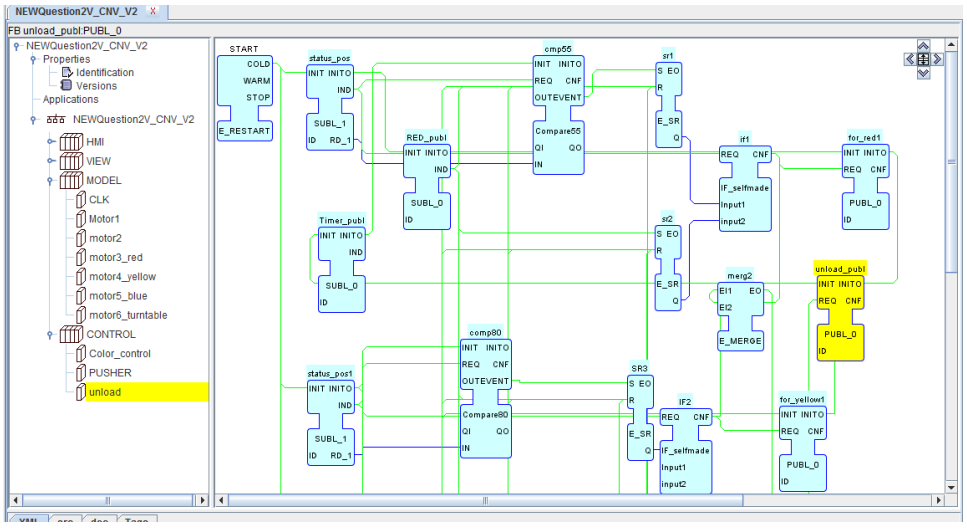

Figure 32: Motor 2 unloading decision block

When WP reaches in front of required pusher, the pusher slides and pushes the device to the next conveyor. It does not matter if the next conveyor is on or off. The pushers are controlled in pusher of control device.

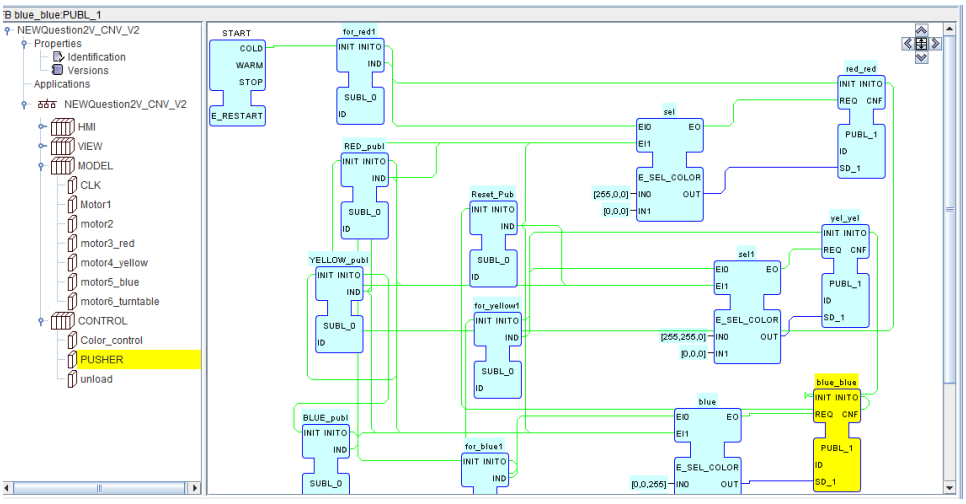

Figure 33: Pusher on/off decision making

As WP reaches the specific colored conveyor, then it is moved to the end if the conveyor is on, otherwise it stays there.

Reset button has priority over everything, as soon as it is pressed everything resets, and system starts over. In case, a there is a WP on conveyor and WP button is pressed again, the system unloads the WP right away and new WP appears.

VIEW

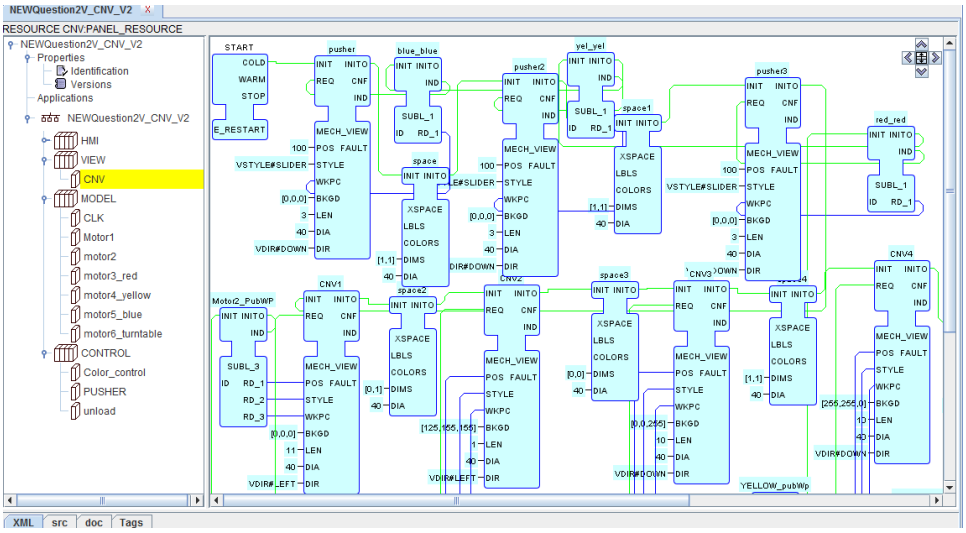

Figure 34: View Layer

There is only one device in view that is controlling the view for pusher, and conveyors and WP appears everything.

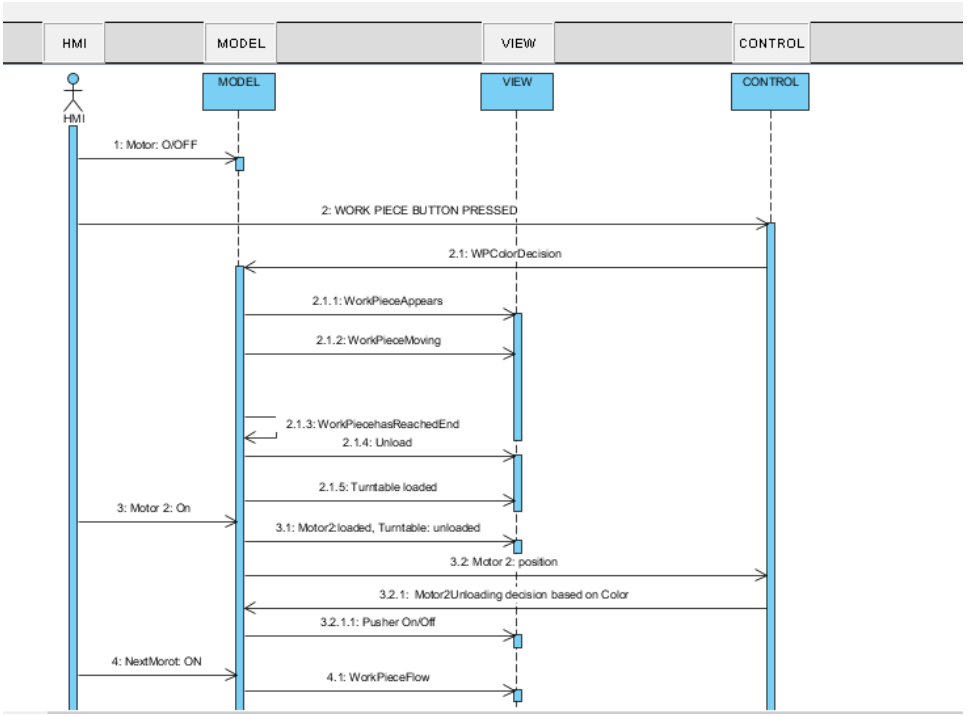

Figure 35: Conveyor System Sequence Diagram