VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FINAL REPORT

ADVANCED CONTROL THEORY

PROF. VINODH KUMAR E

SLOT: F1

EEE4018

REVIEW

TOPIC:

DISTRIBUTED TRAFFIC LIGHT CONTROL SYSTEM

SUBMITTED BY:

LAKSHYA PORWAL – 17BEI0034

KANAV BHATNAGAR - 17BEI0056

INDEX

ABSTRACT

The signal controllers of traffic network constitute a network of computers. They can distribute their computation realizing an efficient traffic control without any central management. However the optimal control inputs can be also calculated by a single traffic controller if the traffic network contains few intersections. The control aim is to relieve traffic congestion, reduce travel time and improve homogenous traffic flow in urban traffic area using distributed control architecture.

Transportation systems are complex dynamic systems that are hard to be modeled exactly. The project presents an improved discrete event simulation model of traffic light control on a single intersection. Model is developed using SIMULINK toolbox provided by MATLAB. We proposed the improved model using controller with three input switches and three output switches combined with traffic signal logic block. In this control system, the fixed time controller algorithm is located within traffic signal logic block.

INTRODUCTION

Traffic controller systems

A traffic signal is typically controlled by a controller mounted inside a cabinet.Some electro-mechanical controllers are still in use (New York City still had 4,800 as of 1998, though the number is lower now due to the prevalence of the signal controller boxes. However, modern traffic controllers are solid state. The cabinet typically contains a power panel, to distribute electrical power in the cabinet; a detector interface panel, to connect to loop detectors and other detectors; detector amplifiers; the controller itself; a conflict monitor unit; flash transfer relays; a police panel, to allow the police to disable the signal; and other components.

In the United States, controllers are standardized by the NEMA, which sets standards for connectors, operating limits, and intervals.The TS-1 standard was introduced in 1976 for the first generation of solid-state controllers.

Solid state controllers are required to have an independent conflict monitor unit (CMU), which ensures fail-safe operation. The CMU monitors the outputs of the controller, and if a fault is detected, the CMU uses the flash transfer relays to put the intersection to FLASH, with all red lights flashing, rather than displaying a potentially hazardous combination of signals. The CMU is programmed with the allowable combinations of lights, and will detect if the controller gives conflicting directions a green signal, for instance.

In the late 1990s, a national standardization effort known as the Advanced transportation controller (ATC) was undertaken in the United States by the Institute of Transportation Engineers. The project attempts to create a single national standard for traffic light controllers. The standardization effort is part of the National Intelligent transportation system program funded by various highway bills, starting with ISTEA in 1991, followed by TEA-21, and subsequent bills. The controllers will communicate using National Transportation Communications for ITS Protocol (NTCIP), based on Internet Protocol, ISO/OSI, and ASN.1.

Fixed time control

Pedestrian traffic signal in Taiwan, featuring a "Walking green man" below a countdown display where the "Red Man" once stood.

In traffic control, simple and old forms of signal controllers are what are known as electro-mechanical signal controllers. Unlike computerized signal controllers, electro-mechanical signal controllers are mainly composed of movable parts (cams, dials, and shafts) that control signals that are wired to them directly. Aside from movable parts, electrical relays are also used. In general, electro-mechanical signal controllers use dial timers that have fixed, signalized intersection time plans. Cycle lengths of signalized intersections are determined by small gears that are located within dial timers. Cycle gears, as they are commonly known, range from 35 seconds to 120 seconds. If a cycle gear in a dial timer results in a failure, it can be replaced with another cycle gear that would be appropriate to use. Since a dial timer has only one signalized intersection time plan, it can control phases at a signalized intersection in only one way. Many old signalized intersections still use electro-mechanical signal controllers, and signals that are controlled by them are effective in one way grids where it is often possible to coordinate the signals to the posted speed

limit. They are however disadvantageous when the signal timing of an intersection would benefit from being adapted to the dominant flows changing over the time of the day.Coordinated control: Diagram demonstrating that when traffic lights are synchronised for traffic travelling in one direction (green arrows), the traffic in the other direction is not necessarily synchronised (blue arrows).Attempts are often made to place traffic signals on a coordinated system so that drivers encounter a green wave, a long string of green lights (the technical term is progression). The distinction between coordinated signals and synchronized signals is very important. Synchronized signals all change at the same time and are only used in special instances or in older systems. Coordinated (progressed) systems are controlled from a master controller and are set up so lights "cascade" (progress) in sequence so platoons of vehicles can proceed through a continuous series of green lights. A graphical representation of phase state on a two-axis plane of distance versus time clearly shows a "green band" that has been established based on signalized intersection spacing and expected vehicle speeds. In some countries (e.g. Germany, France and the Netherlands), this "green band" system is used to limit speeds in certain areas. Lights are timed in such a way that motorists can drive through without stopping if their speed is lower than a given limit, mostly 50 km/h (30 mph) in urban areas. This system is known as "grüne Welle" in German, "vague verte" in French, or "groene golf" in Dutch (English: "green wave"). Such systems were commonly used in urban areas of the United States from the 1940s, but are less common today. In the UK, Slough in Berkshire had part of the A4 experimented on with this. Many US cities set the green wave on two-way streets to operate in the direction more heavily traveled, rather than trying to progress traffic in both directions. But the recent introduction of the flashing yellow arrow (see Traffic-light signalling and operation) makes the lead-lag signal, an aid to progression, available with protected/permissive turns. In modern coordinated signal systems, it is possible for drivers to travel long distances without encountering a red light. This coordination is done easily only on one-way streets with fairly constant levels of traffic. Two-way streets are often arranged to correspond with rush hours to speed the heavier volume direction. Congestion can often throw off any coordination, however. On the other hand, some traffic signals are coordinated to prevent drivers from encountering a long string of green lights. This practice discourages high volumes of traffic by inducing delay yet preventing congestion. Speed is self-regulated in coordinated signal systems; drivers traveling too fast will arrive on

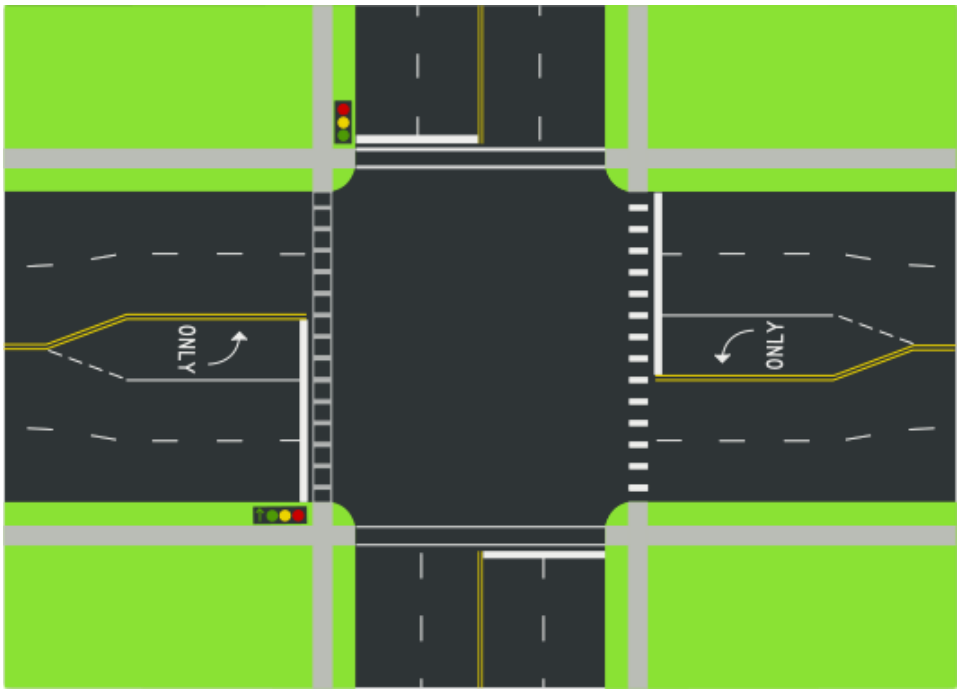
a red indication and end up stopping, drivers traveling too slowly will not arrive at the next signal in time to utilize the green indication. In synchronized systems, however, drivers will often use excessive speed in order to get through as many lights as possible.

As we know all know that traffic light control systems are complex to handle. For this reason, many current methods do not have good theoretical bases. However, without modeling the system appropriately, the characteristic of transportation system cannot be identified.

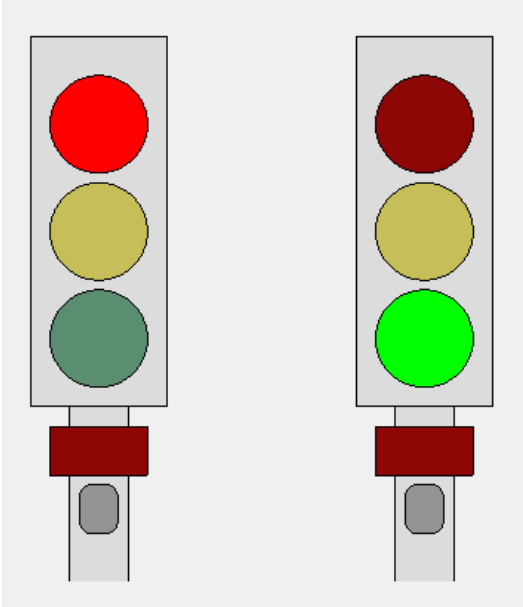
We have developed the single intersection model for implementation of the specific algorithm as discussed earlier.

A Distributed Traffic Control System

This model is for an intersection of two 1-way roads controlled using a distributed control system. In order to coordinate the traffic light state between the two charts, the two charts communicate with each other via messages. The two charts can thus be completely identical.

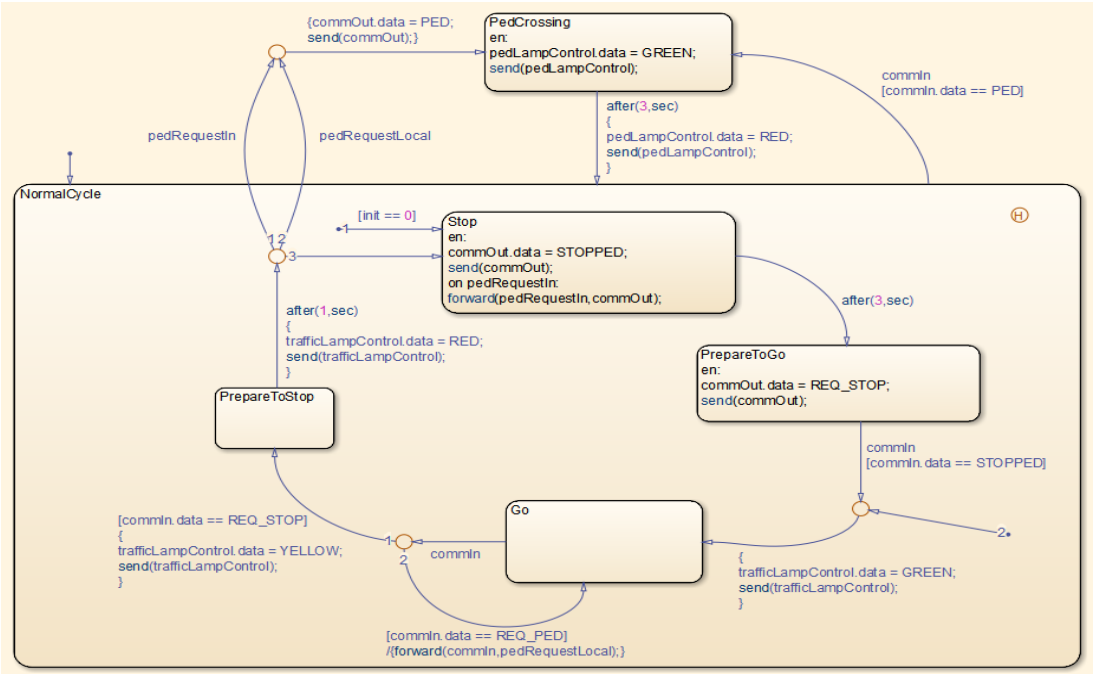


A simple GUI to control the traffic signals has been created using MATLAB. The user can press the pedestrian request button under the pedestrian light to request a pedestrian crossing.



The controller for each road is represented by a traffic light controller subsystem. In this example, these are represented using the "Traffic Light 1" and "Traffic Light 2" subsystems.

The main logic in each subsystem is represented by a "Controller" chart which describes the various states of the traffic signal.



MOTIVE

1. Increasing the traffic handling capacity of roads
2. Reducing collisions and waiting time for both vehicles and pedestrians[17]
3. Encouraging travel within the speed limit to meet green lights
4. Reducing unnecessary stopping and starting of traffic - this in turn reduces fuel consumption, air and noise pollution, and vehicle wear and tear
5. Reducing travel time
6. Reducing driver frustration and road rage

SOFTWARE USED

MATLAB

MATLAB® combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. It includes the Live Editor for creating scripts that combine code, output, and formatted text in an executable notebook.

SIMULINK

Simulink is a MATLAB-based graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries

STATEFLOW

Stateflow is a control logic tool used to model reactive systems via state machines and flow charts within a Simulink model. Stateflow uses a variant of the finite-state machine notation established by David Harel, enabling the representation of hierarchy, parallelism and history within a state chart.

ABOUT DIFFERENT FUNCTION/BLOCKS USED IN OUR MODEL

Subsystem, Atomic Subsystem, Nonvirtual Subsystem, Code Reuse Subsystem

A subsystem block contains a subset of blocks or code within an overall model or system. The subsystem block can represent a virtual subsystem or a nonvirtual subsystem.

In nonvirtual subsystems, you can control when the contents of the subsystem are evaluated. Nonvirtual subsystems are executed as a single unit (atomic execution). You can create conditionally executed nonvirtual subsystems that execute only when a transition occurs on a triggering, function-call, action, or enabling input.

A subsystem is virtual if the block is neither conditionally executed nor atomic. Virtual subsystems do not have checksums.

An Atomic Subsystem block is a subsystem block in which **Treat as atomic unit** is selected by default.

A CodeReuse Subsystem block is a subsystem block in which **Treat as atomic unit** is selected and **Function packaging** is set to Reusable function, specifying the function code generation format for the subsystem.

To create a subsystem, do one of the following:

- Copy a subsystem block from the Ports & Subsystems library into your model. Then add blocks to the subsystem by opening the subsystem block and copying blocks into it.
- Select all blocks and lines that make up the subsystem, and select **Diagram** > **Subsystem & Model Reference** > **Create Subsystem from Selection**. Simulink replaces the blocks with a subsystem block, along with the necessary Inport and Outport blocks to reflect signals entering and leaving the subsystem.

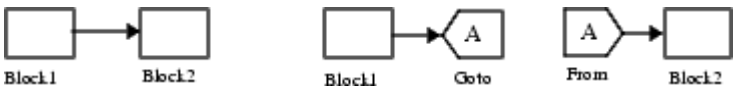
The number of input ports drawn on the subsystem block's icon corresponds to the number of Inport blocks in the subsystem. Similarly, the number of output ports drawn on the block corresponds to the number of Outport blocks in the subsystem.

FROM Block:

The From block accepts a signal from a corresponding Goto block, then passes it as output. The data type of the output is the same as that of the input from the Goto block. From and Goto blocks allow you to pass a signal from one block to another without actually connecting them. To associate a Goto block with a From block, enter the Goto block's tag in the **Goto Tag** parameter.

A From block can receive its signal from only one Goto block, although a Goto block can pass its signal to more than one From block.

This figure shows that using a Goto block and a From block is equivalent to connecting the blocks to which those blocks are connected. In the model at the left, Block1 passes a signal to Block2. That model is equivalent to the model at the right, which connects Block1 to the Goto block, passes that signal to the From block, then on to Block2.



The visibility of a Goto block tag determines the From blocks that can receive its signal. For more information, see Goto and Goto Tag Visibility. The block indicates the visibility of the Goto block tag:

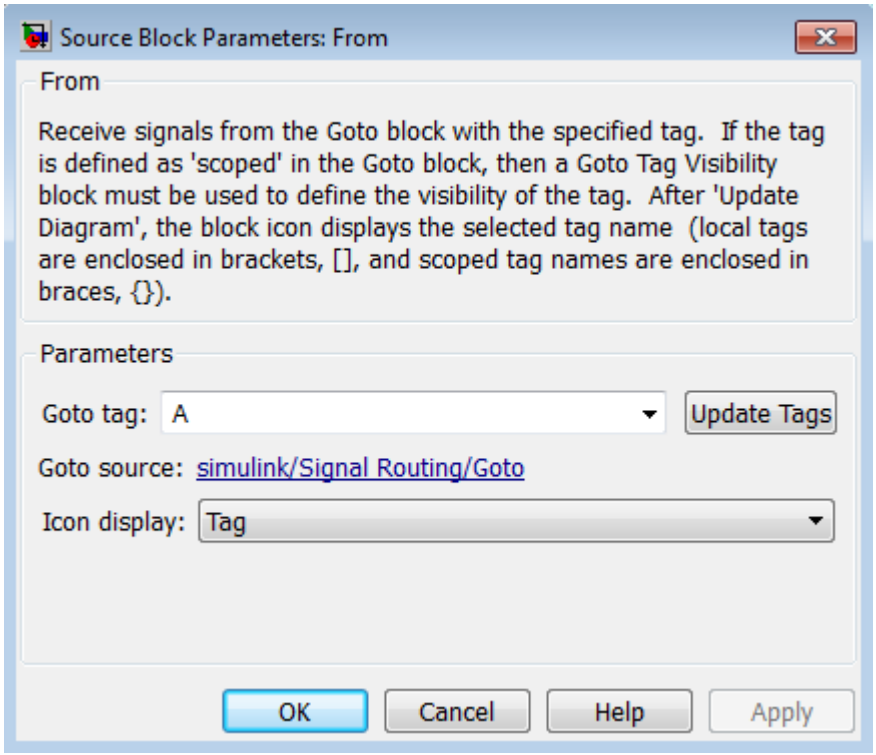
- A local tag name is enclosed in brackets ([]).
- A scoped tag name is enclosed in braces ({}).
- A global tag name appears without additional characters.

The From block supports signal label propagation.

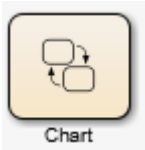
Data Type Support

The From block outputs real or complex signals of any data type that Simulink® supports, including fixed-point and enumerated data types.

For more information, see Data Types Supported by Simulink in the Simulink documentation.



Stateflow CHART:



A *finite state machine* is a representation of an event-driven (reactive) system. In an event-driven system, the system responds to an event by making a transition from one state (mode) to another. This action occurs as long as the condition defining the change is true.

A Stateflow chart is a graphical representation of a finite state machine. *States* and *transitions* form the basic elements of the system. You can also represent stateless flow charts.

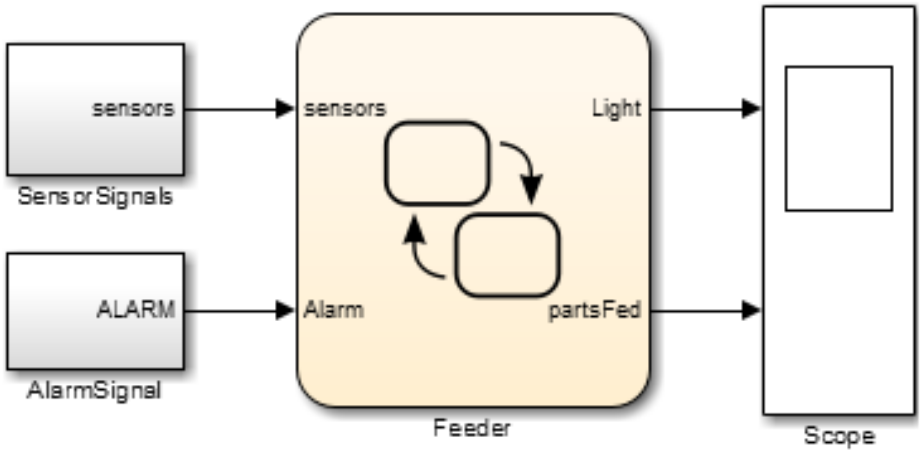
For example, you can use Stateflow charts to control a physical plant in response to events such as a temperature and pressure sensors, clocks, and user-driven events.

You can also use a state machine to represent the automatic transmission of a car. The transmission has these operating states: park, reverse, neutral, drive,

and low. As the driver shifts from one position to another, the system makes a transition from one state to another, for example, from park to reverse.

A Stateflow Chart can use MATLAB or C as the action language to implement control logic.

This block diagram represents a machine on an assembly line that feeds raw material to other parts of the line. It contains a chart, Feeder, with MATLAB as the action language.

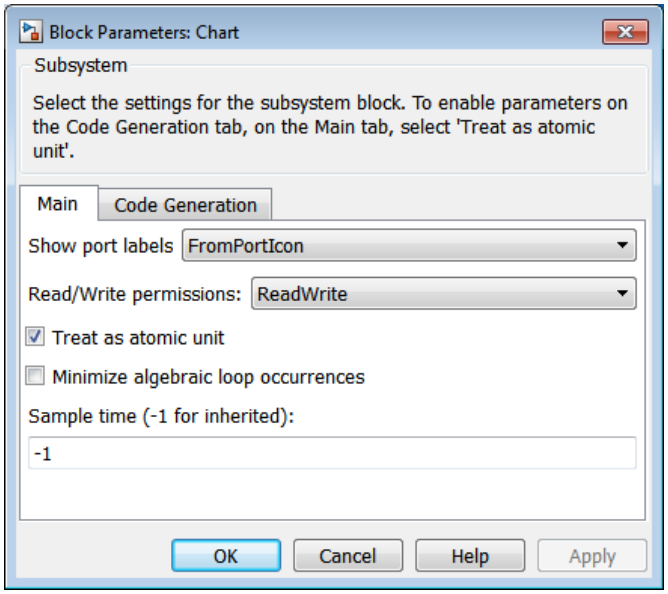


If you double-click the Feeder block in the model, the chart appears.

The Chart block accepts input signals of any data type that Simulink® supports, including fixed-point data and enumerated data types. For a description of data types that Simulink supports, refer to the Simulink documentation.

Floating-point inputs pass through the block unchanged. Boolean inputs to charts that use MATLAB as the action language pass directly as Boolean outputs. Boolean inputs to charts that use C as the action language are treated as double type.

You can declare local data of any type or size.



Outport

Create output port for subsystem or external output

Description

Outport blocks are the links from a system to a destination outside the system.

Simulink® software assigns Outport block port numbers according to these rules:

- It automatically numbers the Outport blocks within a root-level system or subsystem sequentially, starting with 1.
- If you add an Outport block, it is assigned the next available number.
- If you delete an Outport block, other port numbers are automatically renumbered to ensure that the Outport blocks are in sequence and that no numbers are omitted.

Outport Blocks in a Subsystem

Outport blocks in a subsystem represent outputs from the subsystem. A signal arriving at an Outport block in a subsystem flows out of the associated output port on that Subsystem block. The Outport block associated with an output port on a Subsystem block is the block whose **Port number** parameter matches the

relative position of the output port on the Subsystem block. For example, the Outport block whose **Port number** parameter is 1 sends its signal to the block connected to the topmost output port on the Subsystem block.

If you renumber the **Port number** of an Outport block, the block becomes connected to a different output port, although the block continues to send the signal to the same block outside the subsystem.

When you create a subsystem by selecting existing blocks, if more than one Outport block is included in the grouped blocks, Simulink software automatically renumbers the ports on the blocks.

The Outport block name appears in the Subsystem icon as a port label. To suppress display of the label, click the Outport block and select **Format** > **Hide Name**.

Initializing Outport Blocks in Conditionally Executed Contexts

To set initial conditions for an Outport block in a conditionally executed subsystem, use one of these approaches.

- Inherit initial values from input signals for the subsystem.
- Explicitly specify initial values

For details, see Specify or Inherit Conditional Subsystem Initial Values.

Top-level Outport Block in a Model Hierarchy

Outport blocks at the top-level of a model hierarchy have two uses: to supply external outputs to the base MATLAB® workspace, which you can do by using either **Configuration Parameters**pane parameters or the sim command, and to provide a means for analysis functions to obtain output from the system.

- To supply external outputs to the workspace, use the **Configuration Parameters** > **Data Import/Export** pane. For example, if a system has more than one Outport block and the save format is array, the following command

```
[t,x,y] = sim(...);
```

writes y as a matrix, with each column containing data for a different Outport block. The column order matches the order of the port numbers for the Outport blocks.

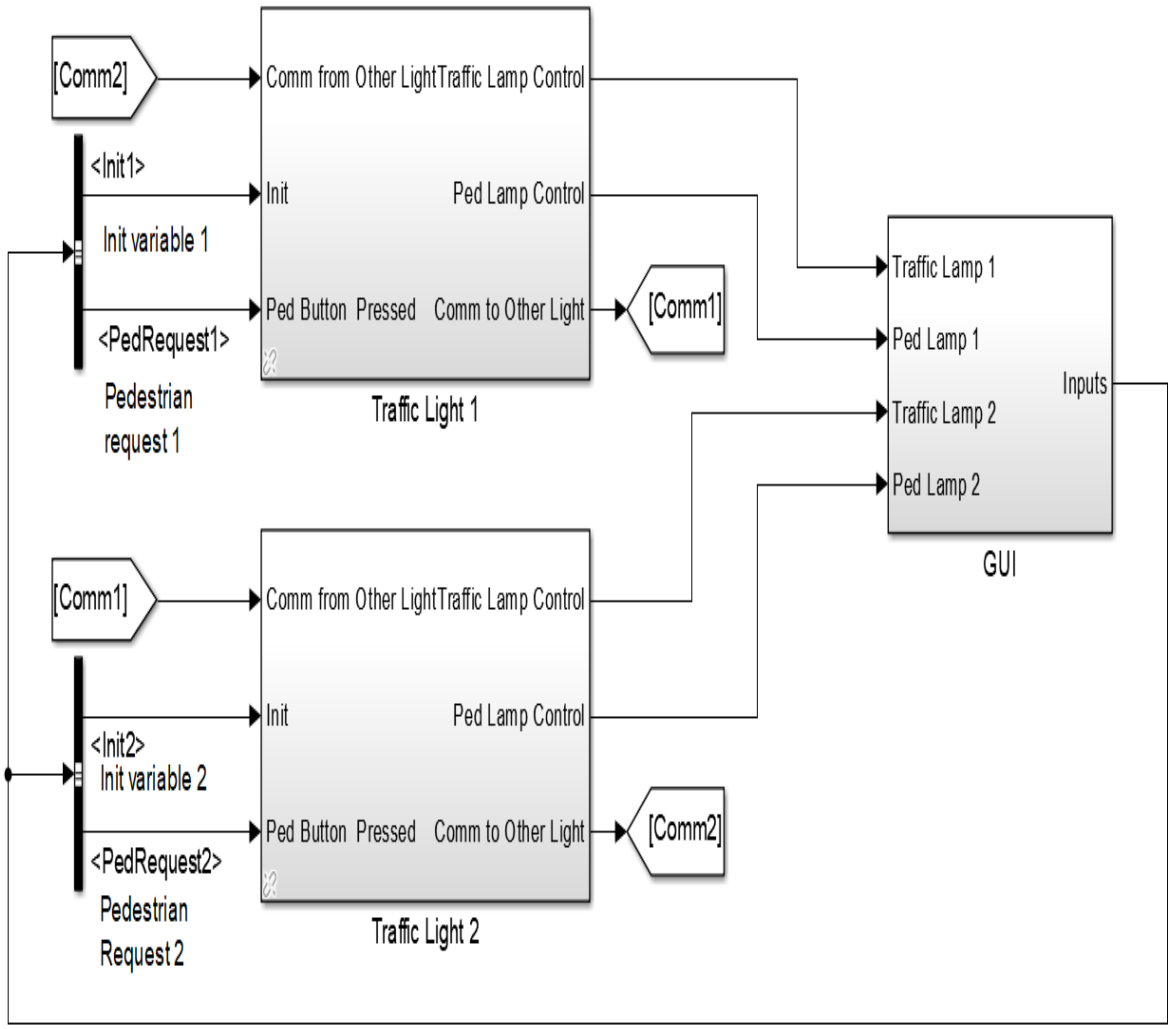
If you specify more than one variable name after the second (state) argument, data from each Outport block is written to a different variable. For example, if the system has two Outport blocks, to save data from Outport block 1

to speed and the data from Outport block 2 to dist, you could specify this command:

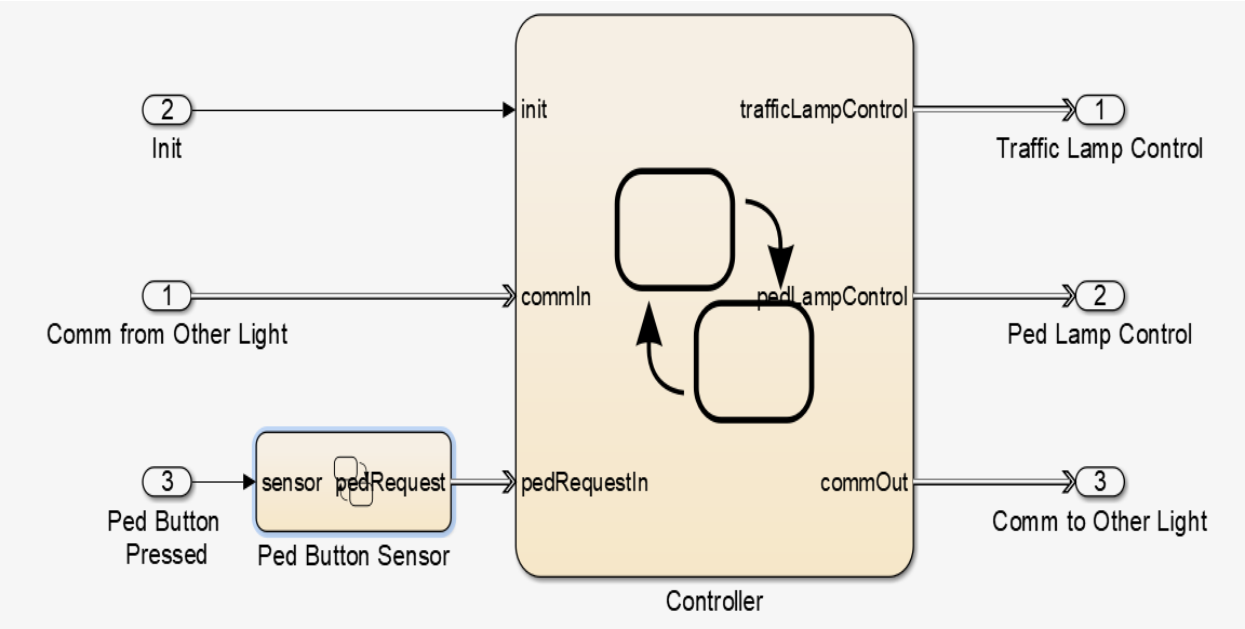
[t,x,speed,dist] = sim(...);

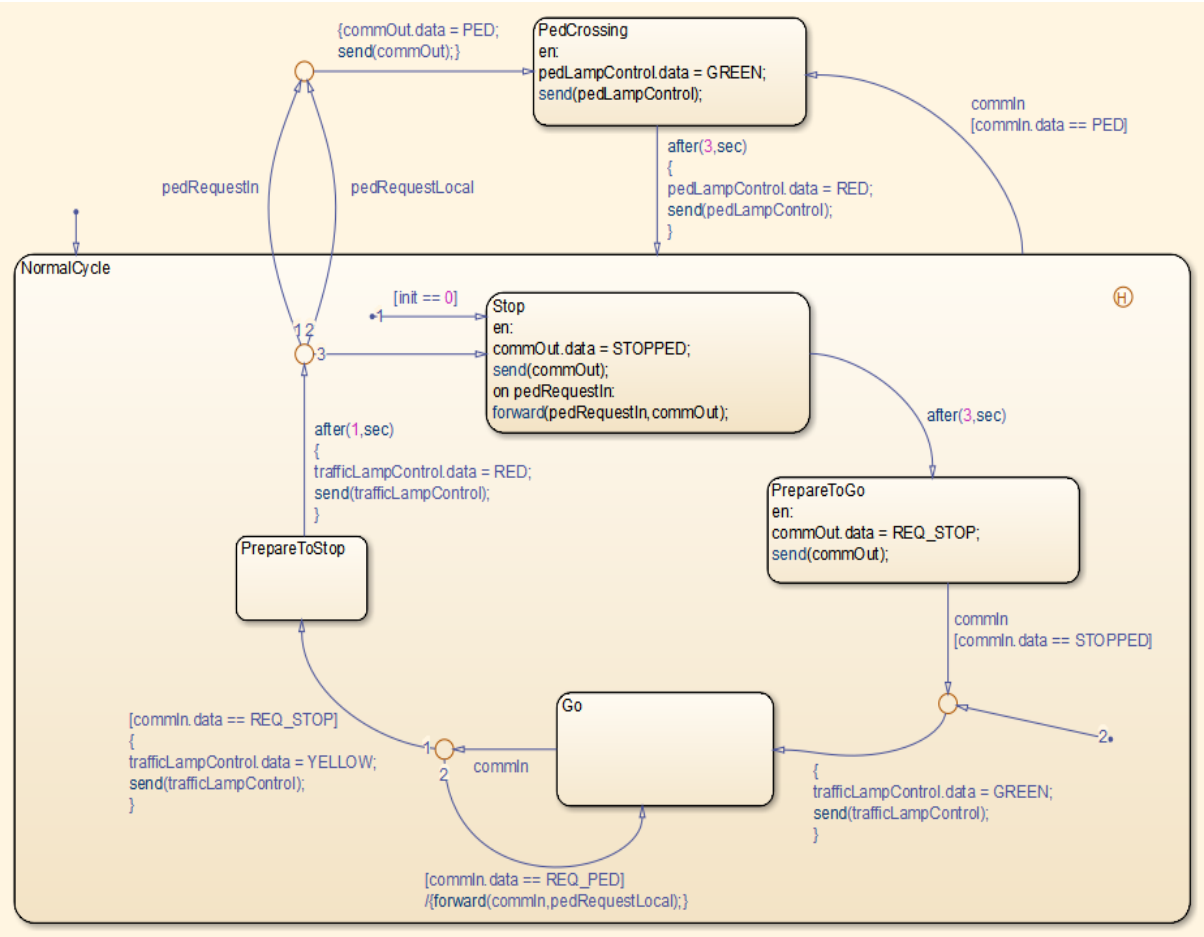
- To provide a means for the linmod and trim analysis functions to obtain output from the system.

SIMULINK MODEL

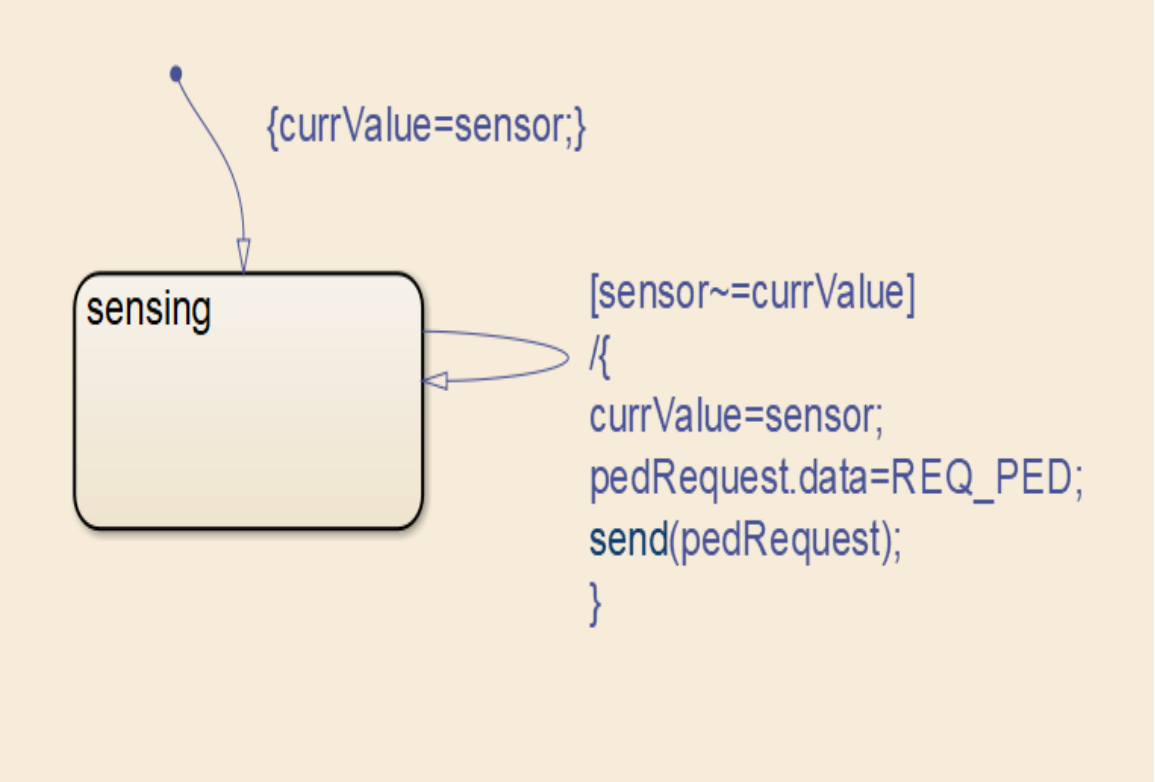


Stateflow Model:

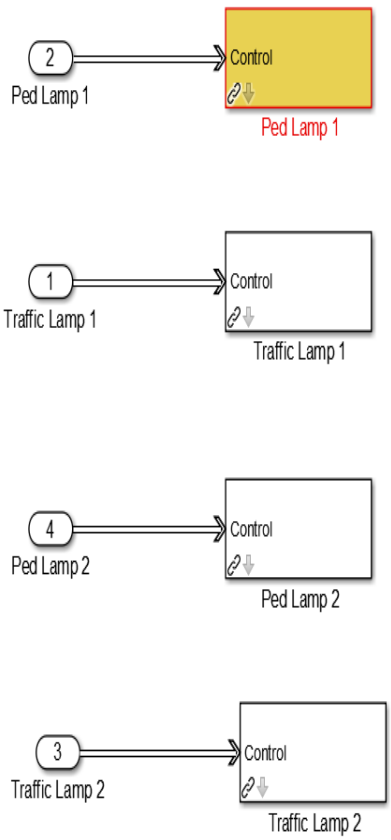


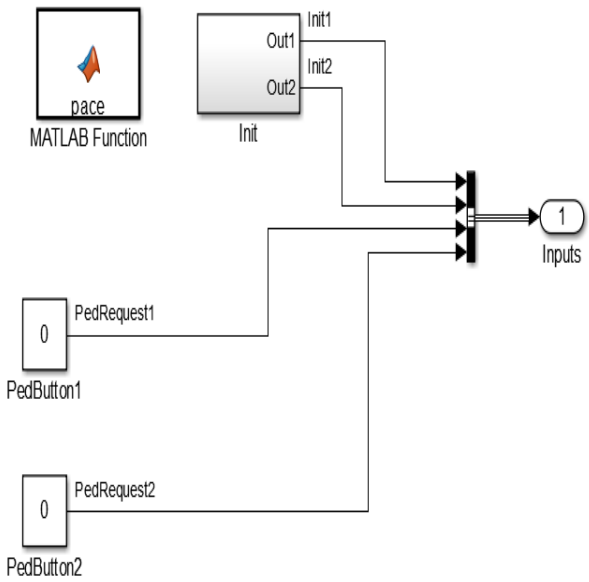


Sensing Block:



GUI Model





WORKING

There are 3 inputs- communication signal, initialization variable and Pedestrian Request to the subsystems of each traffic light.

The output of each traffic light subsystem are communication signal of the respective traffic light, Pedestrian control lamp and Traffic control lamp. These outputs can be displayed on our Graphical User Interface, which has 2 traffic lights, pedestrian lamp and a switch for Pedestrian Request.

The output of the traffic lights are then feedback to our subsystems for further communication between traffic lights.

Our traffic light subsystem consists of a controller made in stateflow. The controller consists of 4 states- Stop, prepare to go, Go, Prepare to stop

Stop- The red light is on in this state, it will be on when the other light is green and also when there is a pedestrian request.

Prepare to Go- The light remains red in this state, if a pedestrian request is there, it will send the request, after 3 seconds the pedestrian lamp will turn green and traffic lamps red.

Go- In this state the light turns green. The vehicles can move now for 3 seconds until the traffic light again becomes red.

Prepare to stop- The traffic light turns yellow in this state and the cars thus slowdown for the red signal to come.

Separate logic block for when Pedestrian request is there. The pedestrian request can be given any instant and it will go to this logic block where the Pedestrian lamp is turns green after 3 seconds and after 3 seconds again turns to red.

APPLICATIONS

Optimum control of fluctuating traffic volumes such as over saturated or unusual load conditions. • Improve the vehicular throughput

• Maximises the traffic flow

• Control the time intervals of the traffic light.

• Decrease delays of vehicles and environmental effects

• Increase intersection capacity at same time.

• Reduces the number of accidents

• Reduces the average waiting time at red lights

• Reduces wastage of time by a green light an empty road.

• Better traffic flow leading to greener environment

FUTURE POSSIBILITES

The model can be extended further for multiple intersection and will be extended further in research.

Sensible interactive design.

CONCLUSION

The project is a discrete event simulation model of traffic light control on single intersection. The improvement of model is to use the control with two input switches and two output switches combined with the traffic signal logic block. The simulation has shown the model is functioning properly.

REFERENCES

https://en.wikipedia.org/wiki/MATLAB

https://en.wikipedia.org/wiki/Traffic_light_control_and_coordination

https://www.researchgate.net/publication/261307591_A_Distributed_Algorithm_for_Adaptive_Traffic_Lights_Control

https://www.mathworks.com/help/matlab/

https://www.slideshare.net/fieyzad1/application-of-traffic-light