Learning Report – MBSE

GLOBAL ENGINEERING ACADEMY

GENSIS

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	21/09/20	Milind Mohapatra	Prassana Kumar S(99002671) Sachin C Mirji(99002673)	To be: Pagala Prithvi Sekhar, Srinivas K, Bhargav N	Initial draft
2	23/09/20	Milind Mohapatra	Prassana Kumar S(99002671) Sachin C Mirji(99002673)	To be: Pagala Prithvi Sekhar, Srinivas K, Bhargav N	
3	25/09/20	Milind Mohapatra	Prassana Kumar S(99002671) Sachin C Mirji(99002673)	To be: Pagala Prithvi Sekhar, Srinivas K, Bhargav N	
4	26/09/20	Milind Mohapatra	Prassana Kumar S(99002671) Sachin C Mirji(99002673)	To be: Pagala Prithvi Sekhar, Srinivas K, Bhargav N	

Table of Contents

TABLE OF FIGURES **5**

ACTIVITY 1: REVOLUTIONIZING MBSE BY GULFSTREAM AEROSPACE **7**

ORIGIN 7
CHALLENGES FACED UP TILL 2003 8
WHAT HAPPENED AFTER 2003 8
TOOLS AND SOFTWARE USED BY GULFSTREAM AEROSPACE 8

ACTIVITY 2: EQUATION MODELLING **11**

LOGICAL EXPRESSION: 11
TRIGONOMETRIC EXPRESSION: 12
WATER LEVEL: 12
FIRST ORDER DIFFERENTIAL EXPRESSION: 13
SECOND ORDER DIFFERENTIAL EXPRESSION: 13

ACTIVITY 3: MATLAB **14**

ALGORITHM DEVELOPMENT 14
ALGORITHM 14
Import the data *14*
Create output variable *14*
Create table *14*
Allocate imported array to column variable names *14*
Clear temporary variables *14*
Assign independent variables for all the sensors *14*
Time Normalization *15*
Linear acceleration data analysis *15*
Clear no longer required variables *16*
Gravity sensor data analysis *16*
CUSTOM FUNCTION 18
Sensor Fusion *18*
Clear no longer required variables *18*
Accessing ride quality *18*
Putting together all data *19*
Write all the data to a csv file named "ALL_DATA_LOGS" *19*
AUTOMATION SCRIPT 20

Import data *20*
Find number of occurrences of a word *20*
Display *20*
MATLAB ONRAMP COURSE CERTIFICATE 21

ACTIVITY 4: SIMULINK 22

IN CLASS ACTIVITIES: 22
INSTRUMENT CLUSTER: 22
Test Plan: *27*
DEBUGGING 29
Passing Simulation: *29*
Show label *31*
Adding breakpoint: *33*
Understanding Execution Order *35*
Sampling Time (Timing Legend) *36*
Logging *38*
Highlight to Source and Destination *40*
Floating Viewer/ Scope *41*
DIFFERENT TYPES OF SUBSYSTEM 43
Atomic Subsystem: *43*
Non Atomic subsystem: *43*
Triggered Subsystem: *43*
Enabled subsystem: *45*
Variant subsystem: *46*
SIMULINK ONRAMP COURSE CERTIFICATE 48

ACTIVITY 5: STATEFLOW 49

HVAC 49
DEBUGGING TECHNIQUES IN STATEFLOW CHART 49
Breakpoint *49*
Pause time *50*
STATEFLOW ONRAMP COURSE CERTIFICATE 51

Table of Figures

Figure 1 Gulfstream G650 (Image courtesy of Gulfstream) 7
Figure 2 A Gulfstream jet forward fuselage system (Image courtesy of Gulfstream)....8
Figure 3 A fastener projected onto the curved surface of a door (Image courtesy of Gulfstream) 9
Figure 4 Logical expression block diagram 11
Figure 5 Trigonometric expression block diagram 12
Figure 6 Water level block diagram 12
Figure 7 First order differential expression block diagram 13
Figure 8 Second order differential equation block diagram 13
Figure 9 Rate of change of acceleration 16
Figure 10 Rate of change of perceived gravity 17
Figure 11 Bumps on the road 18
Figure 12 Matlab Onramp certificate 21
Figure 13 Instrument cluster dashboard 23
Figure 14 Automotive console 23
Figure 15 Fuel Gauge subsystem 24
Figure 16 Speed(km/hr) subsystem 24
Figure 17 Speed(miles/hr) subsystem 25
Figure 18 RPM subsystem from speed(km/hr) 25
Figure 19 RPM subsystem from speed(miles/hr) 25
Figure 20 Left and right indicator subsystem 26
Figure 21 Parking indicator subsystem 26
Figure 22 Test signals 27
Figure 23 Speed 28
Figure 24 RPM, fuel, indicator and parking 28
Figure 25 Simulink model for Temperature conversion (C to F) 29
Figure 26 Passing simulation using Pause time 30
Figure 27 Scope values 30
Figure 28 Label display on signal line 31
Figure 29 Label values during simulation 32
Figure 30 Breakpoint addition in signal line 33
Figure 31 Breakpoint Execution 34
Figure 32 The execution is paused at breakpoint 35
Figure 33 Execution order of the system 36
Figure 34 Colored Legend representation 37
Figure 35 Logging creation at signal line 38
Figure 36 Log compare of two signals 39
Figure 37 Highlighting source and destination 40
Figure 38 Floating Viewer Test Model 41
Figure 39 Floating Viewer /Scope in Signal Lines 42
Figure 40 Triggered Subsystem 44
Figure 41 Enabled and triggered subsystem 44
Figure 42 Scope of enabled and triggered subsystem 45
Figure 43 Enabled Subsystem 46

Figure 44 Variant subsystem 47
Figure 45 Simulink onramp certificate 48
Figure 46 Breakpoint 49
Figure 47 Pause time 50
Figure 48 Stateflow onramp certificate 51

Activity 1: Revolutionizing MBSE by Gulfstream Aerospace

Origin

Gulfstream aerospace company is an American company and a wholly owned subsidiary of General Dynamics. They design, develop, manufacture, market and service business jet aircrafts such as the Gulfstream G650 model shown in figure 1. Gulfstream has produced more than 2000 aircrafts since 1958.

Figure 1 Gulfstream G650 (Image courtesy of Gulfstream)

Challenges faced up till 2003

- Historically been low volume, highly custom, hand built interiors
- Relied on highly skilled craftsmen to interpret engineering drawings with limited content
- Model based design process was needed to support higher volume production

What happened after 2003

In 2003, Gulfstream Aerospace implemented MBD (Model Based Design) and have since achieved a series of remarkable successes. Developing and FAA-certifying the industry's first and currently only electronic 3D MBTD System and the G650 aircraft being the first completely 3D MBTD system designed and certified are a few of those successes.

Between the 2D drawing approach for the G550 model and the MBD approach for the G650 model, the standard parts and part numbers were reduced by more than 50 percent. Even more amazingly, the number of outfitting clips, angles and brackets were reduced from around 450 down to only 6. The reduction of part numbers alone generated huge savings in the supply chain management.

Tools and software used by Gulfstream Aerospace

1. **CATIA V5**: To provide a true 3D electronic representation of the aircraft, the engineering team modeled everything, including fasteners, veneer, hoses and so on, as the nominal geometries using CATIA V5. The 3D digital models allowed the team to increase the part reuse during the design phase, minimize technician's subjective customization on the shop floor and ultimately reduce part numbers by more than 50 percent. Nowadays, the model-based interior design has been expanded to the entire airplane fleet.

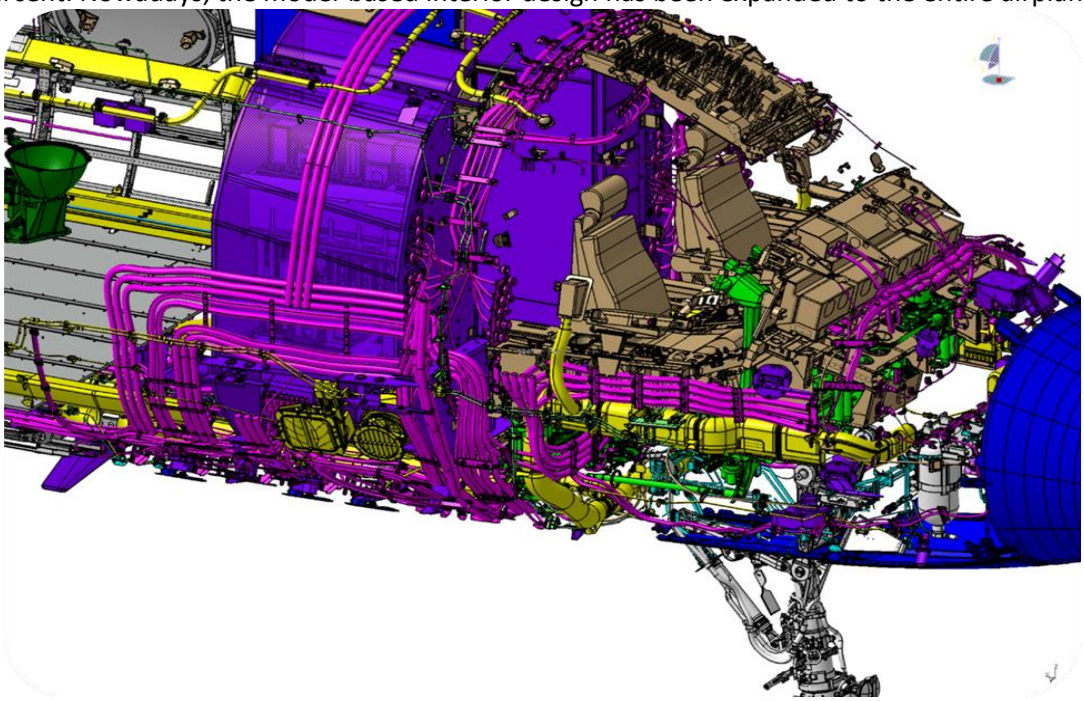


Figure 2 A Gulfstream jet forward fuselage system (Image courtesy of Gulfstream)

2. **ProjectWorks:** Using this software, along with multiple high-resolution laser projectors, Gulfstream projects the fastener map directly onto the airplane's body. The projectors coordinate with each other to bend and reshape the image to make sure that all the fasteners are correctly located on the curved surface. With the digital model projection, you can modify the model or instructions in real time and they will be instantly updated and mapped correctly onto the airplane, without having to wait for the redrawn physical nylon pieces anymore. One tip here is that rather than using actual protrusions, Gulfstream only models the fasteners as points and vectors with attributes such as types, tooling and pre-drill information. This can prevent the model heaviness from being loaded onto computers.

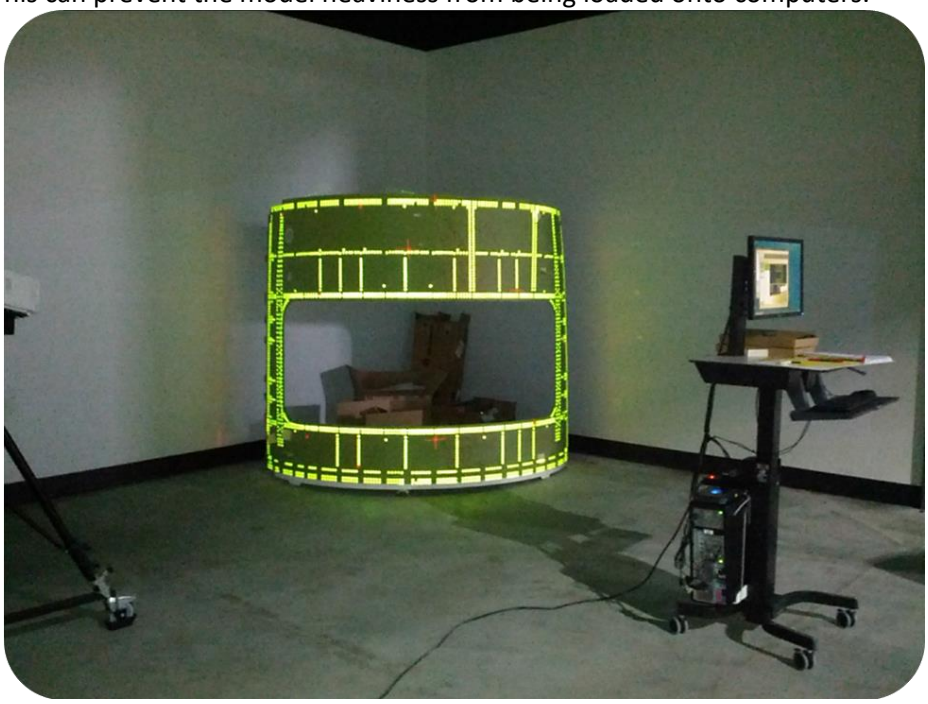


Figure 3 A fastener projected onto the curved surface of a door (Image courtesy of Gulfstream)

3. **Gulfstream Symmetry™ Flight Deck: A** fully featured model based systems engineering tools and program support simulator.
4. **Matlab and Simulink:** Simulink, Aerospace Blockset, and Simulink Coder to model and simulate the digital flight-control system and aircraft dynamics in real time.
5. **MSC Nastran**
6. **Patran**
7. **Apex**
8. **HyperWorks suite**
9. **Abaqus**
10. **SmarTeam**

MODEL IN LOOP

The actual model of the aircraft is designed to capture the most real world data readings during a simulation. Then the controlled is designed and modelled so that it can control the model of the aircraft as per requirement.

SOFTWARE IN LOOP

Once the model is verified in MIL, the next stage is Software in Loop, where the code is generated from the controller model and controller block is replaced with this code. This gives clarity on whether the controller logic is hardware implementable. In case the input output logs match with that of MIL, we move to the next step.

PROCESSOR IN LOOP

The controller model, i.e, the logic block in SIL is loaded onto an embedded processor and is run in a closed loop simulation with simulated model of aircraft and dynamics from the MIL.

HARDWARE IN LOOP

Once the above steps are successful a lab scale model of the actual hardware is built and tested while being controlled by the embedded processor from the PIL step.

Activity 2: Equation Modelling

Logical Expression:

((a<b) && !(b>c)) || ((a==c) && !(b==2))

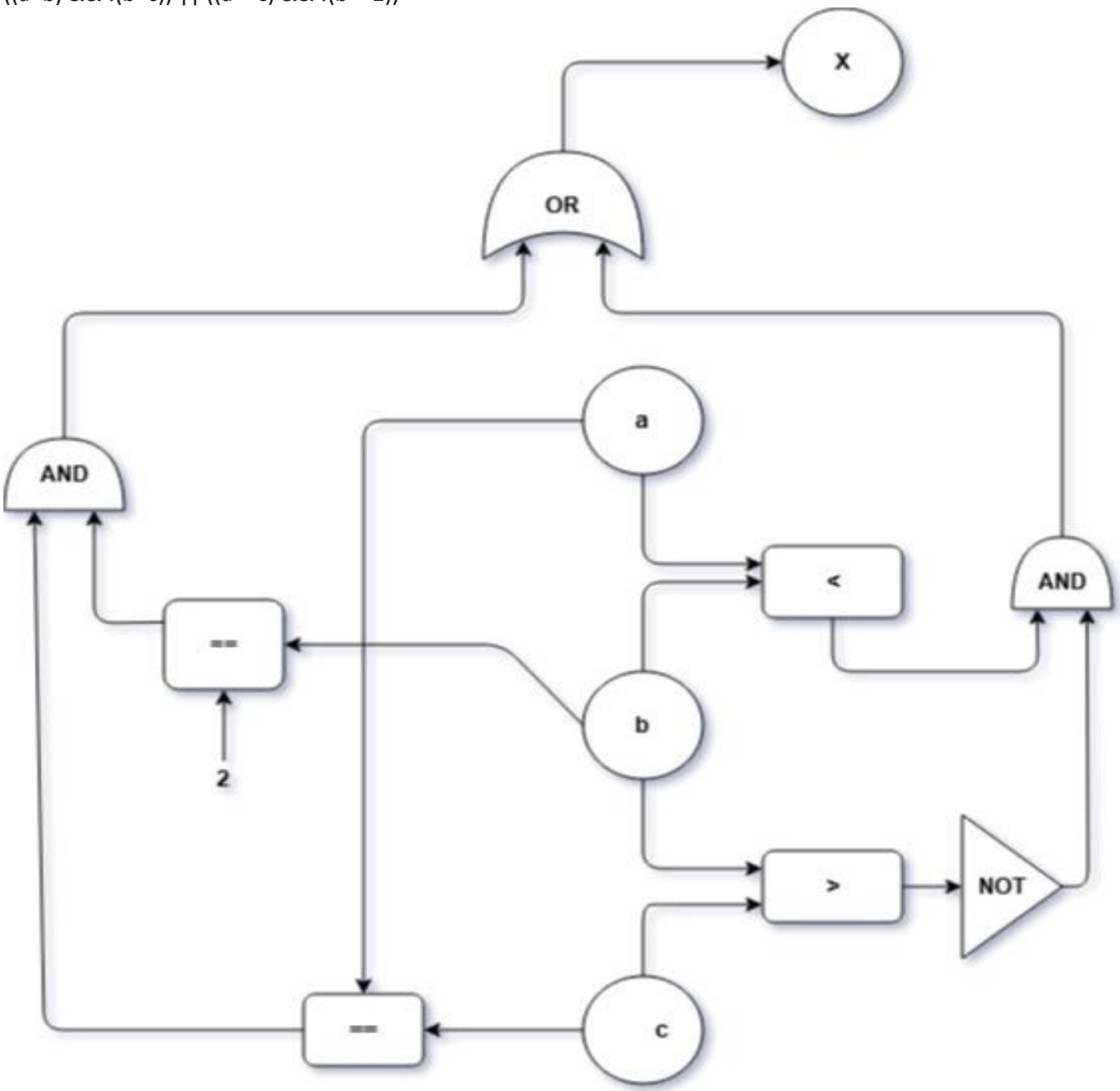


Figure 4 Logical expression block diagram

Trigonometric Expression:

v = 1/3(pi*r^2*h)

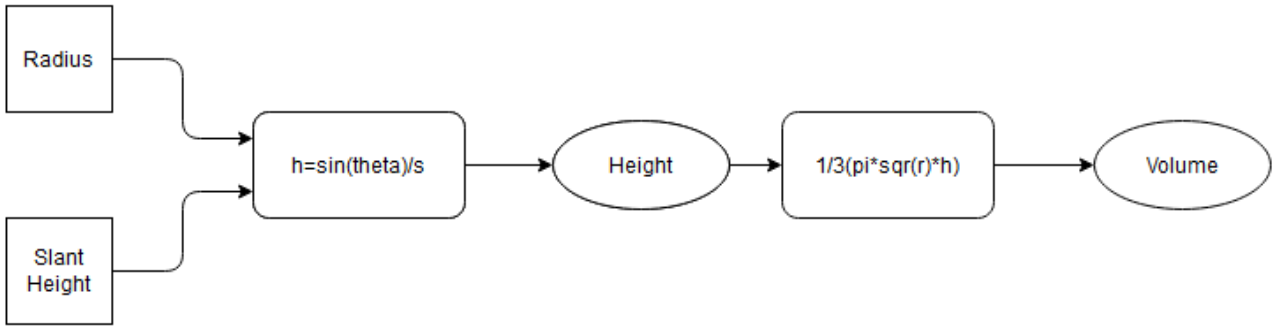


Figure 5 Trigonometric expression block diagram

Water level:

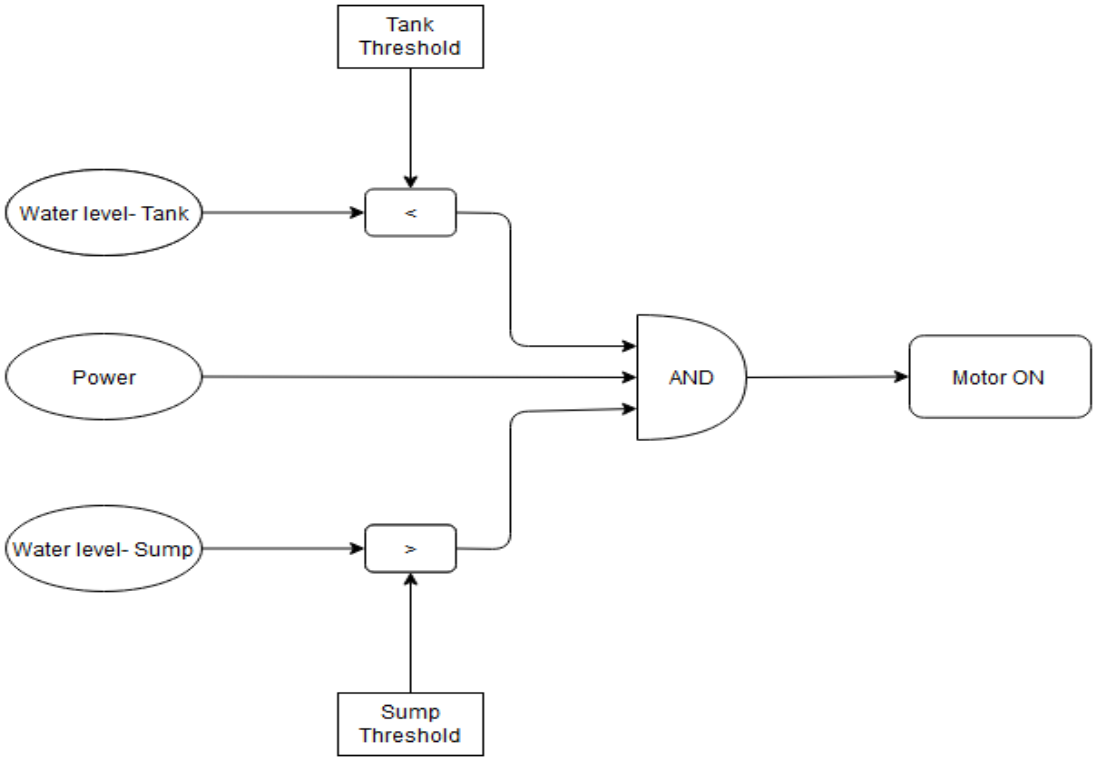


Figure 6 Water level block diagram

First Order Differential Expression:

y’ + a(x)y = y(x)

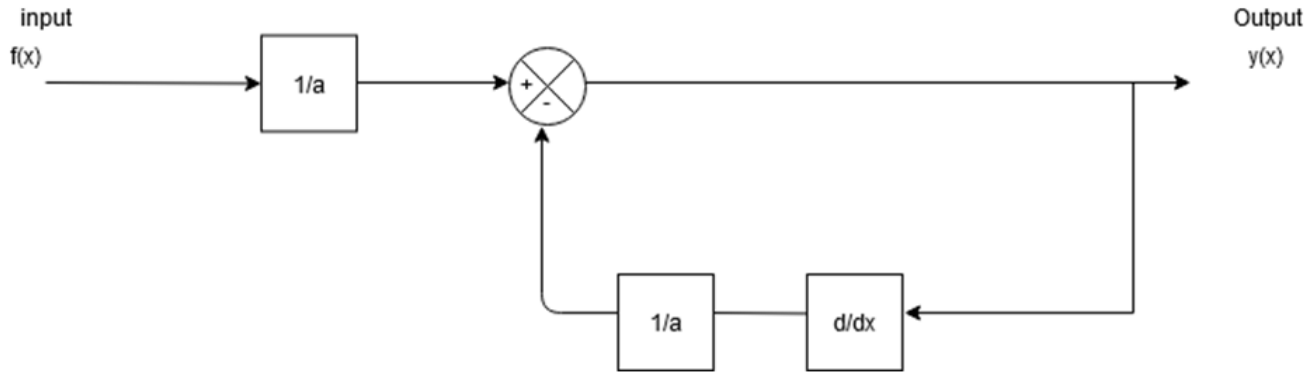


Figure 7 First order differential expression block diagram

Second Order Differential Expression:

y(t) = $-\left(\frac{B}{M}\right)y'(t) - \left(\frac{k}{M}\right)y(t) + \left(\frac{1}{M}\right)f(t)$

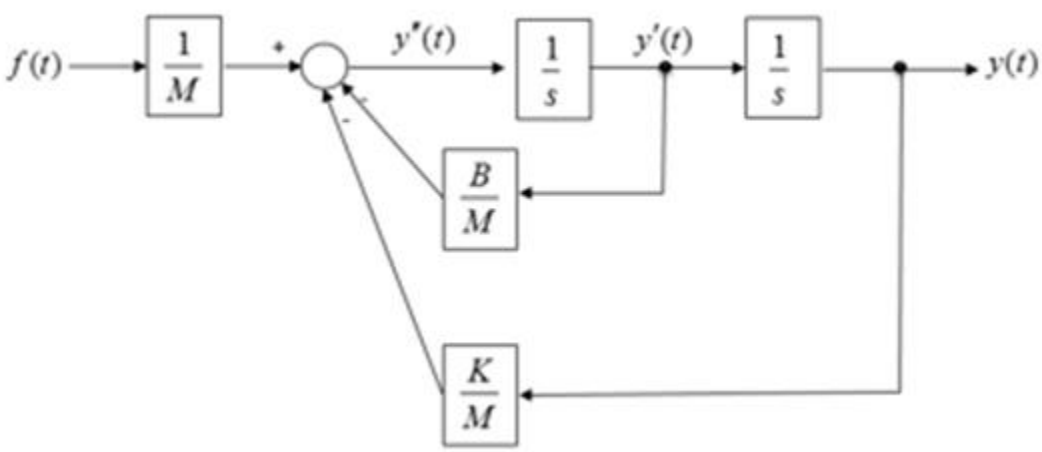


Figure 8 Second order differential equation block diagram

Activity 3: MATLAB

Algorithm development

Project Name: Bus ride quality estimation

Algorithm

Script for importing data from the following spreadsheet:

```
Worksheet: D:\Matlab\Genesis-Matlab\database\log.csv
@author : Milind Mohapatra
@PS no. : 99002670
```

Import the data

```
[acc] = xlsread('D:\MATLAB\Genesis-Matlab\database\log.csv','G2:I4457');
[gravity] = xlsread('D:\MATLAB\Genesis-Matlab\database\log.csv','D2:F4457');
[time] = xlsread('D:\MATLAB\Genesis-Matlab\database\log.csv','AD2:AD4457');
raw = [acc,gravity,time];
```

Create output variable

```
data = raw;
```

Create table

```
bus_data = table;
```

Allocate imported array to column variable names

```
bus_data.ACCNx = data(:,1);
bus_data.ACCNy = data(:,2);
bus_data.ACCNz = data(:,3);
bus_data.GRAVITYx = data(:,4);
bus_data.GRAVITYy = data(:,5);
bus_data.GRAVITYz = data(:,6);
bus_data.TIME = data(:,7);
```

Clear temporary variables

```
clearvars raw acc gravity time;
```

Assign independent variables for all the sensors

```
%Linear Acceleration on all three axes
ax = bus_data.ACCNx;
ay = bus_data.ACCNy;
az = bus_data.ACCNz;
%Gravity changes on all three axes
gx = bus_data.GRAVITYx;
gy = bus_data.GRAVITYy;
gz = bus_data.GRAVITYz;
%Time
t_temp = bus_data.TIME;
iteration = 4455;
```

Time Normalization

```
for i = 1:iteration
    t(i,:) = t_temp(i+1,1) - t_temp(i,1);
end

t_real(1,1) = 0;
for i = 1:iteration
    t_real(i+1,:) = t_real(i,:) + t(i,:);
end
%Time elapsed in seconds
t_real = t_real.*0.001;

%Clear no longer required variables
clearvars t_temp;
clearvars t;
```

Linear acceleration data analysis

```
%Change in acceleration every 0.1s on each axis
for i = 1:iteration
    axt(i,:) = ax(i+1,1)-ax(i,1);
end

for i = 1:iteration
    ayt(i,:) = ay(i+1,1)-ay(i,1);
end

for i = 1:iteration
    azt(i,:) = az(i+1,1)-az(i,1);
end

%Combining the acceleration change over all axis
acc_temp = sum(axt.^2 + ayt.^2 + azt.^2, 2);
acc_change = sqrt(acc_temp);
```

```
%Plot overall acceleration changes throughout journey
subplot(3,1,1);
plot(acc_change, 'm');
xlabel('Time in secs');
ylabel('Accleration in m/s^2');
title('RATE OF CHANGE OF ACCELERATION');
```

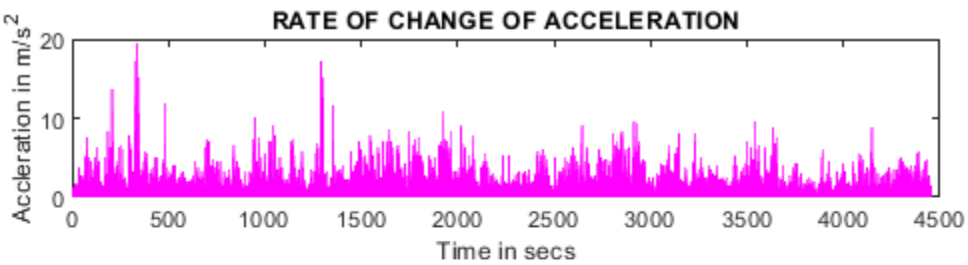


Figure 9 Rate of change of acceleration

Clear no longer required variables

```
clearvars acc_temp ax ay az;
clearvars axt ayt azt;
```

Gravity sensor data analysis

```
for i = 1:iteration
    gxt(i,:) = gx(i+1,1)-gx(i,1);
```

```
end

for i = 1:iteration
    gyt(i,:) = gy(i+1,1)-gy(i,1);
end

for i = 1:iteration
    gzt(i,:) = gz(i+1,1)-gz(i,1);
end

gvt_change = sum(gxt.^2 + gyt.^2 + gzt.^2, 2);

%Plot overall gravity changes throughout journey
subplot(3,1,2);
plot(gvt_change);
xlabel('Time');
ylabel('Gravity in m/s^2');
title('RATE OF CHANGE OF PERCIEVED GRAVITY');
```

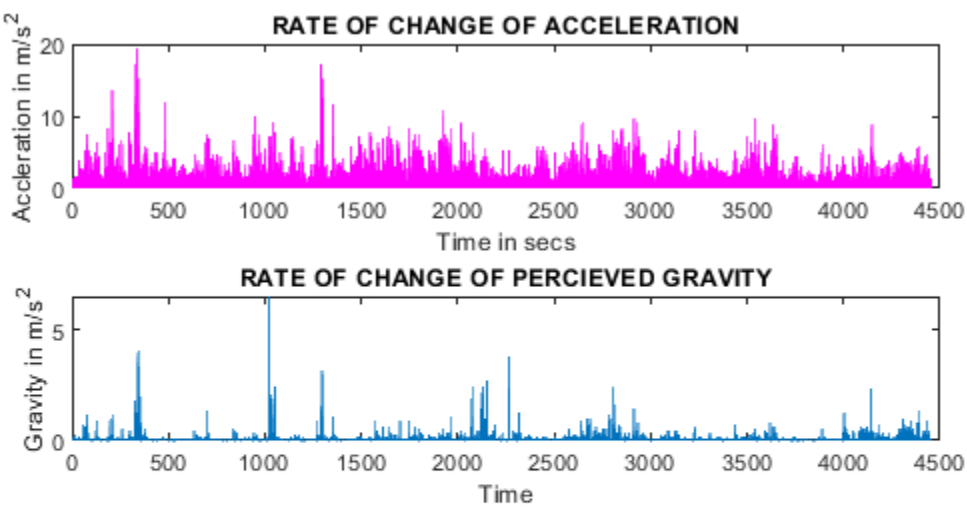


Figure 10 Rate of change of perceived gravity

Custom function

Sensor Fusion

```
%Use of custom function : fuse(arg1,arg2S)
final_data = fuse(acc_change, gvt_change);
subplot(3,1,3);
plot(final_data);
xlabel('Time');
ylabel('Bumps');
title('BUMPS ON THE ROAD');
```

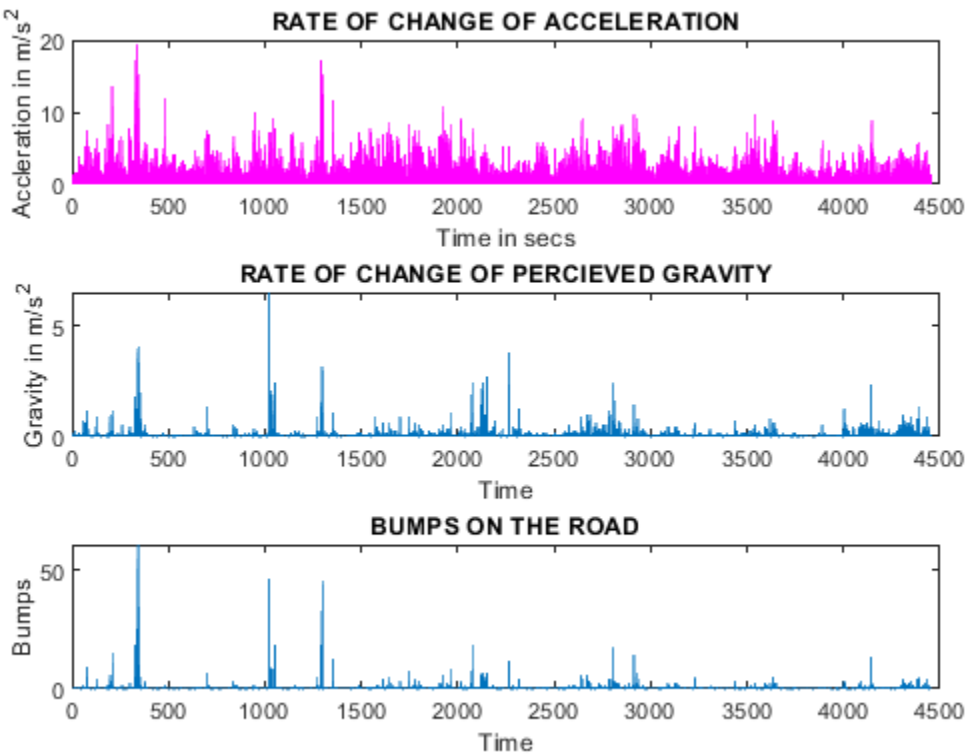


Figure 11 Bumps on the road

Clear no longer required variables

```
clearvars gx gy gx;
clearvars gxt gyt gzt;
```

Accessing ride quality

```
bump = 0;
smooth = 0;
threshold = 2;
for i = 1: iteration
    if final_data(i,:) > threshold
        bump = bump + 1;
    else
        smooth = smooth + 1;
    end
end

total_counter = bump+smooth;
percent_good_ride = (smooth/total_counter)*100;
percent_bad_ride = (bump/total_counter)*100;
percent_bad_ride = round(percent_bad_ride);
DISPLAY = sprintf('%d percent of the total ride was bad', percent_bad_ride);
disp(DISPLAY);
```

```
3 percent of the total ride was bad
```

Putting together all data

```
for i = i:size(final_data)
    null_matrix(i,:) = 0000;
end
alldata =
horzcat(data(2:end,1),data(2:end,2),data(2:end,3),data(2:end,4),data(2:end,5),data(2:end,6),data(2:end,7)
,null_matrix, acc_change,gvt_change,final_data);
```

Write all the data to a csv file named "ALL_DATA_LOGS"

```
csvwrite('ALL_DATA_LOGS.csv', alldata);
```

Published with MATLAB® R2020b

Matlab algorithm script

Automation script

```
% @author Milind Mohapatra
% @date 25.09.2020
% @description This script displays the number of occurrences of a given
% string in the given text file
```

Import data

```
clear all;
clc;
file = fopen('file.txt');
words = textscan(file,'%s');
fclose(file);
```

Find number of occurrences of a word

```
text = '';

for i=1:length(words{1})
    str=char(words{1}(i));
    text = string(text) + ' ' + string(str);
end
pattern = "for";
regex = regexp(text, pattern);
```

Display

```
disp("Number of occurrences of word 'for' in the text file")
length(regex)
```

```
Number of occurrences of word 'for' in the text file

ans =

    12
```

Published with MATLAB® R2020b

Matlab automation script

MATLAB OnRamp course certificate

MathWorks | Training Services

Course Completion Certificate

Milind Mohapatra

has successfully completed 100% of the self-paced training course

MATLAB Onramp

DIRECTOR, TRAINING SERVICES

21 September 2020

Figure 12 Matlab Onramp certificate

Activity 4: Simulink

In class activities:

- Celsius to Fahrenheit
- Celsius to Fahrenheit Subsystem
- Kelvin to Celsius
- Kelvin to Celsius subsystem
- Voltage to Kelvin
- Voltage to Kelvin Subsystem
- Voltage to Fahrenheit Subsystem

Instrument Cluster:

- Voltage to speed(m/s)
- Speed(m/s) to speed(km/hr)
- Speed(km/hr) to RPM
- Indicator
- Speed to RPM and Indicator
- Fuel gauge
- Automotive Console Final
- Automotive Console Test

Simulink models

Instrument Cluster

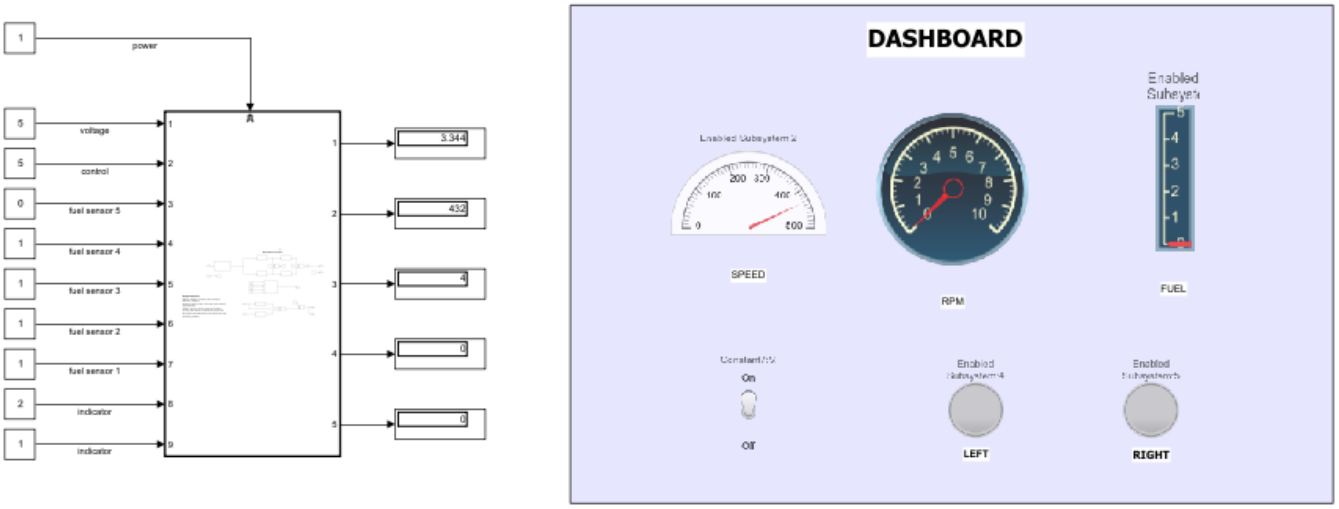
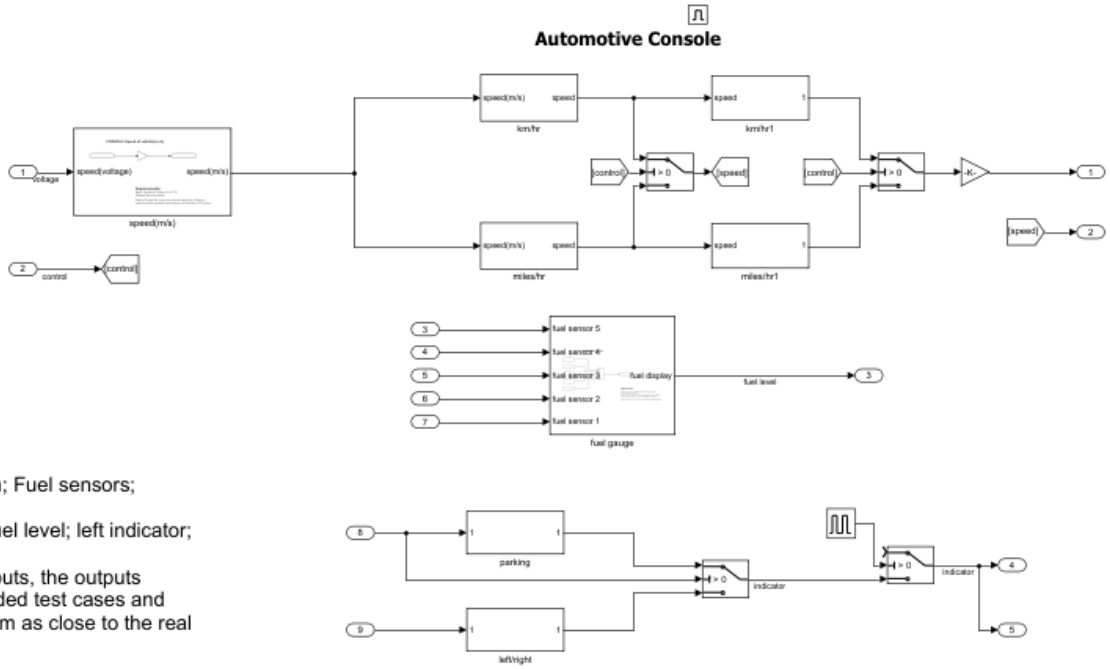
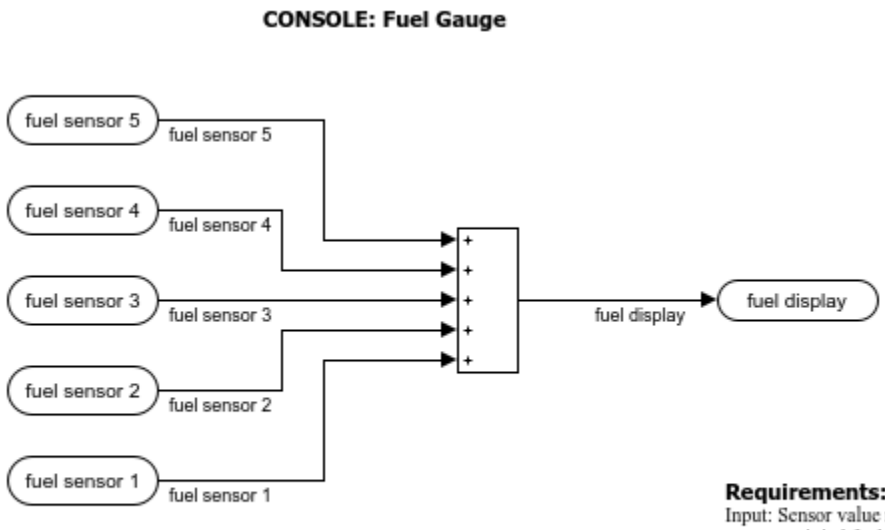


Figure 13 Instrument cluster dashboard



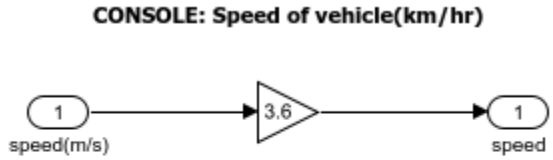
Requirements:
INPUT: Voltage; Location; Fuel sensors; Indicator; Parking
OUTPUT: Speed; rpm; fuel level; left indicator; right indicator
LOGIC: For the given inputs, the outputs should pass all the provided test cases and the system should perform as close to the real world as possible.

Figure 14 Automotive console



Requirements:
Input: Sensor value read from tank (consider 5 sensors)
Output: Digital fuel display
Logic: map the fuel in the tank to digital levels (eg: 0-5 litres fuel in tank corresponds to level 1 in digital fuel display)
Reference: https://www.youtube.com/watch?v=0du-QU1Q0T4

Figure 15 Fuel Gauge subsystem



Requirements:
Input: Speed in m/sec
Output: Speed in km/hr
Logic: Design the system to convert speed from m/sec to km/hr

Figure 16 Speed(km/hr) subsystem

CONSOLE: Speed of vehicle(miles/hr)

Requirements:
Input: Speed in m/sec
Output: Speed in miles/hr
Logic: Design the system to convert speed from m/sec to miles/hr

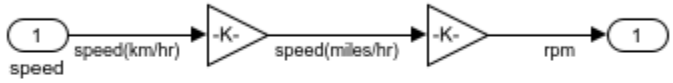
Figure 17 Speed(miles/hr) subsystem

CONSOLE: rpm

Requirements:
Input: Speed in km/hr
Output: RPM
Logic: Design the system to convert speed in km/hr to RPM
Reference: https://itstillruns.com/formula-rpm-5076999.html

Figure 18 RPM subsystem from speed(km/hr)

CONSOLE: rpm



Requirements:
Input: Speed in miles/hr
Output: RPM
Logic: Design the system to convert speed in km/hr to RPM
Reference: https://itstillruns.com/formula-rpm-5076999.html

Figure 19 RPM subsystem from speed(miles/hr)

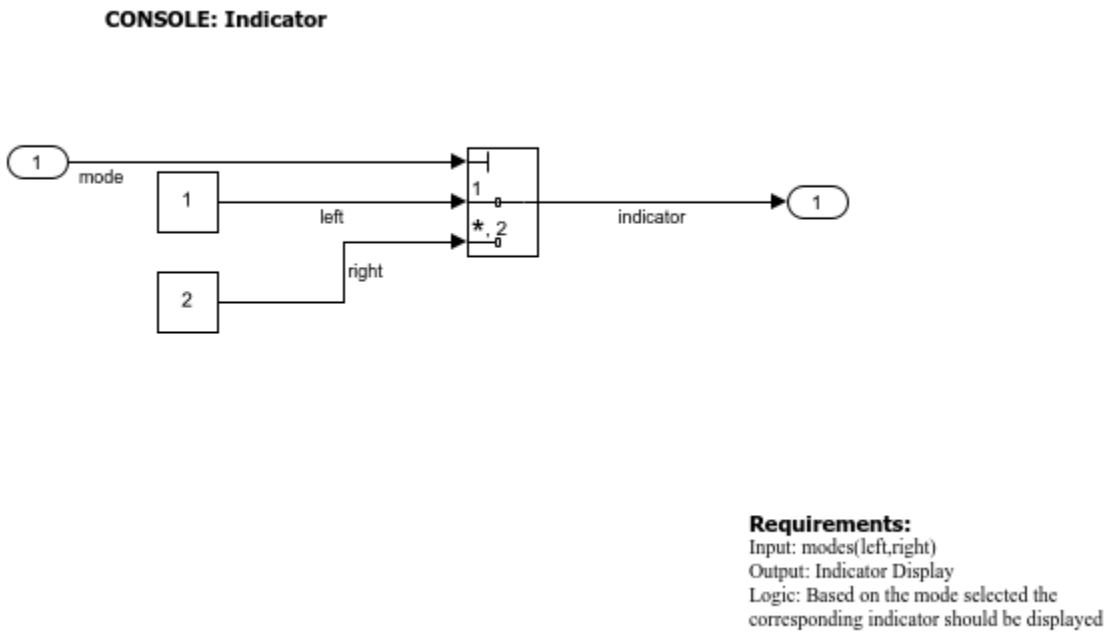


Figure 20 Left and right indicator subsystem

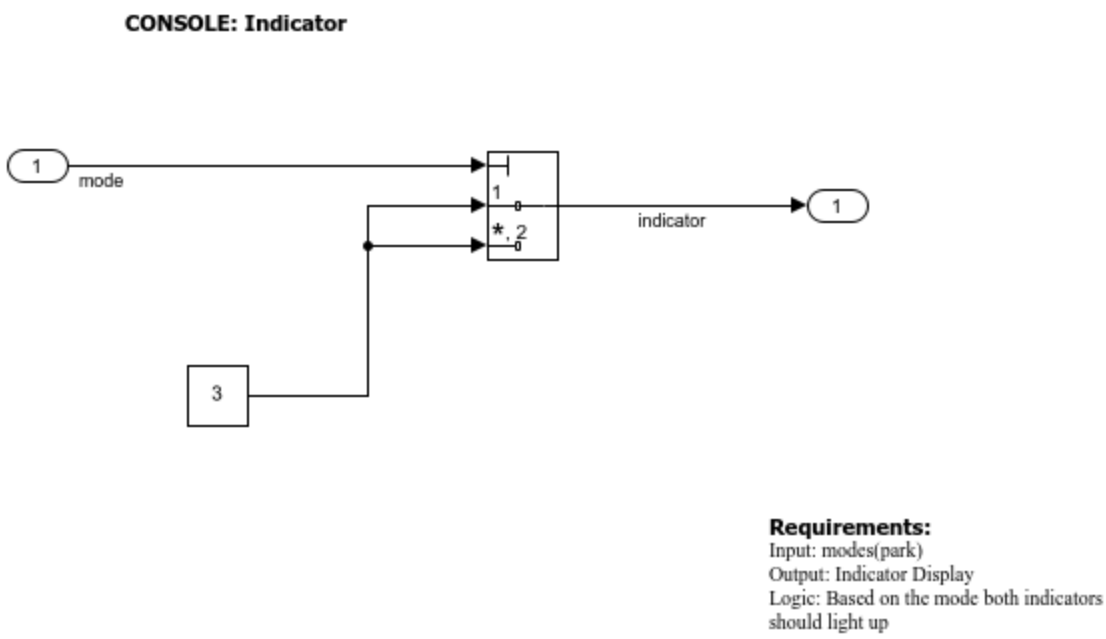


Figure 21 Parking indicator subsystem

Test Plan:

The Automotive Console was tested against the following inputs,

Time	Power	Voltage	Control	Fuel 1	Fuel 2	Fuel 3	Fuel 4	Fuel 5	Indicator 1	Indicator 2
0	1	0	0	0	0	0	0	0	1	1
1	1	0.5	0	1	0	0	0	0	2	2
2	1	1	0	1	1	0	0	0	1	1
3	1	1.5	0	1	1	1	0	0	2	2
4	1	2	0	1	1	1	1	0	1	1
5	1	2.5	0	1	1	1	1	1	2	2
6	1	3	1	1	1	1	1	1	1	1
7	1	3.5	1	1	1	1	1	1	2	2
8	1	4	1	1	1	1	1	1	1	1
9	1	4.5	1	1	1	1	1	1	2	2
10	1	5	1	1	1	1	1	1	1	1
11	0	5	1	1	1	1	1	1	1	1

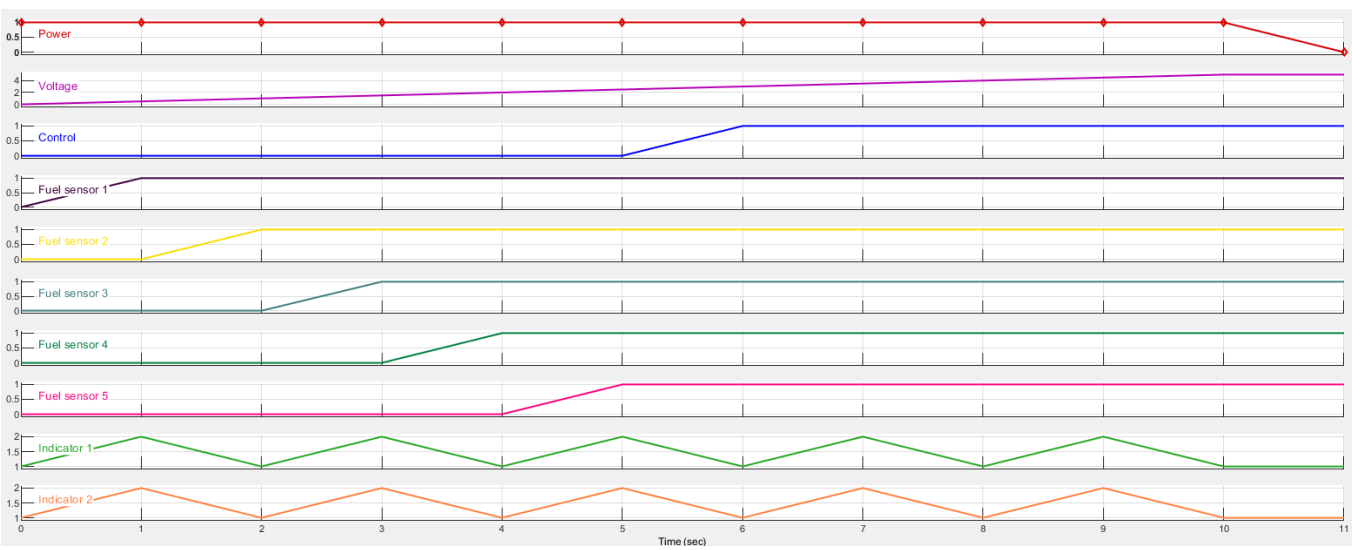


Figure 22 Test signals

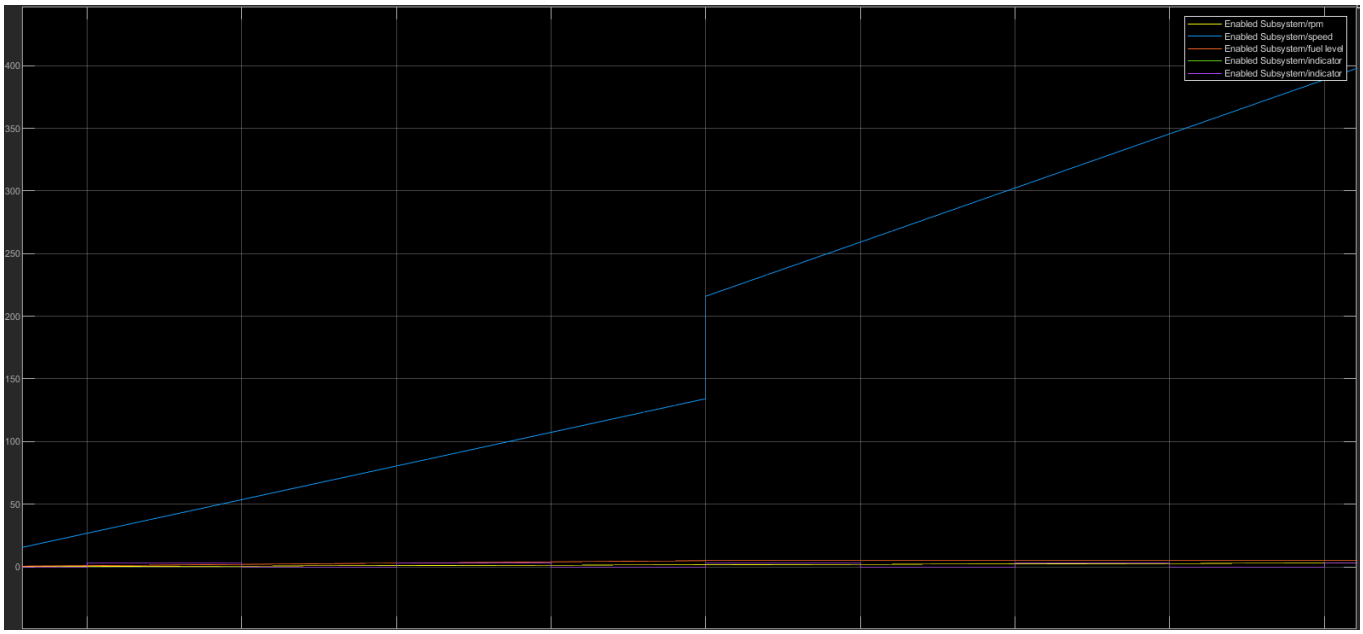


Figure 23 Speed

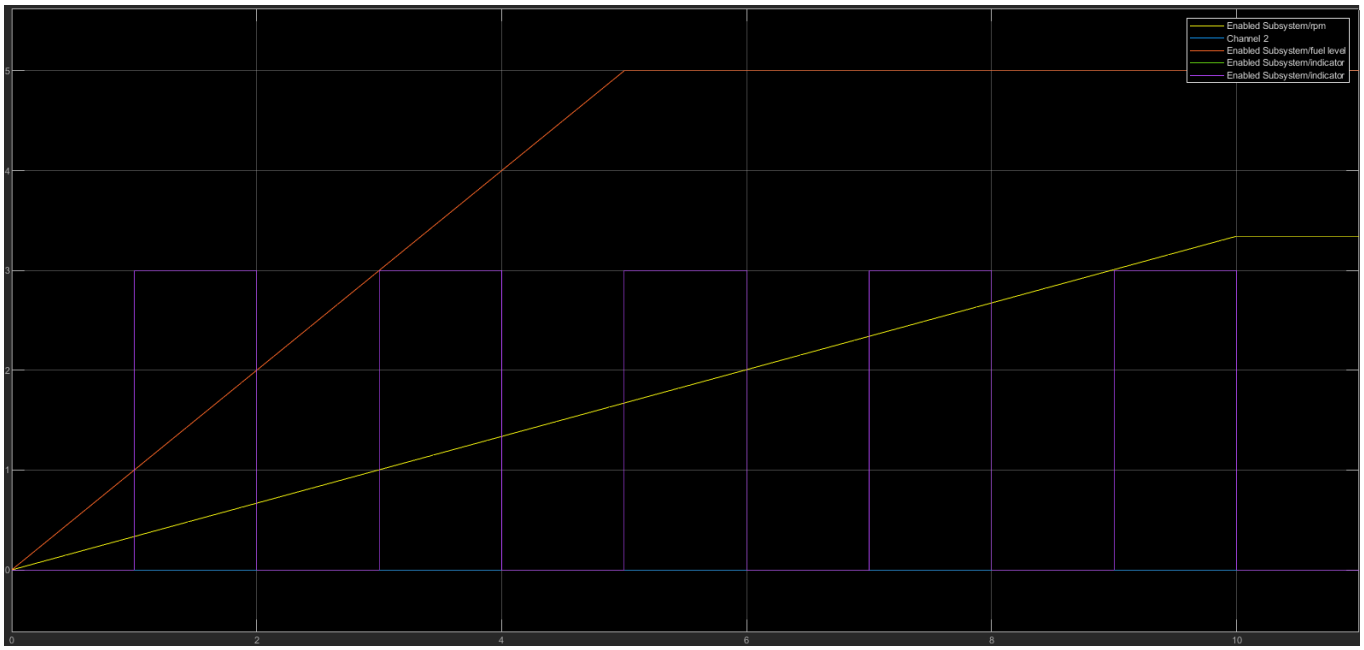


Figure 24 RPM, fuel, indicator and parking

Debugging

Model: Temperature conversion from Celsius to Fahrenheit

Simulink Model:

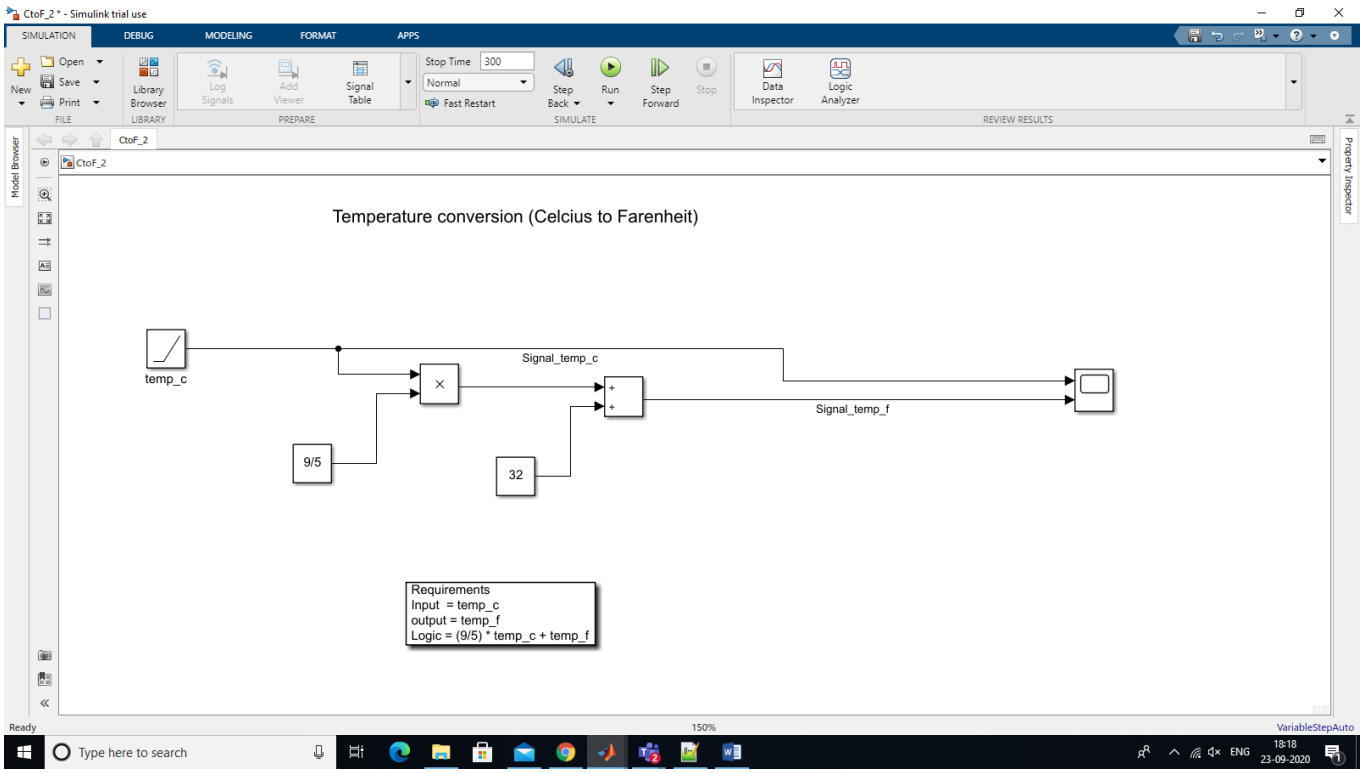


Figure 25 Simulink model for Temperature conversion (C to F)

Rquirements:

Input: Temperature in Celsius
Output: Temperature in Fahrenheit
temp_f = (9/5) * temp_c + 32

Passing Simulation:

The simulation can be passed by setting desired time to 'Pause time' in DEBUG tab.

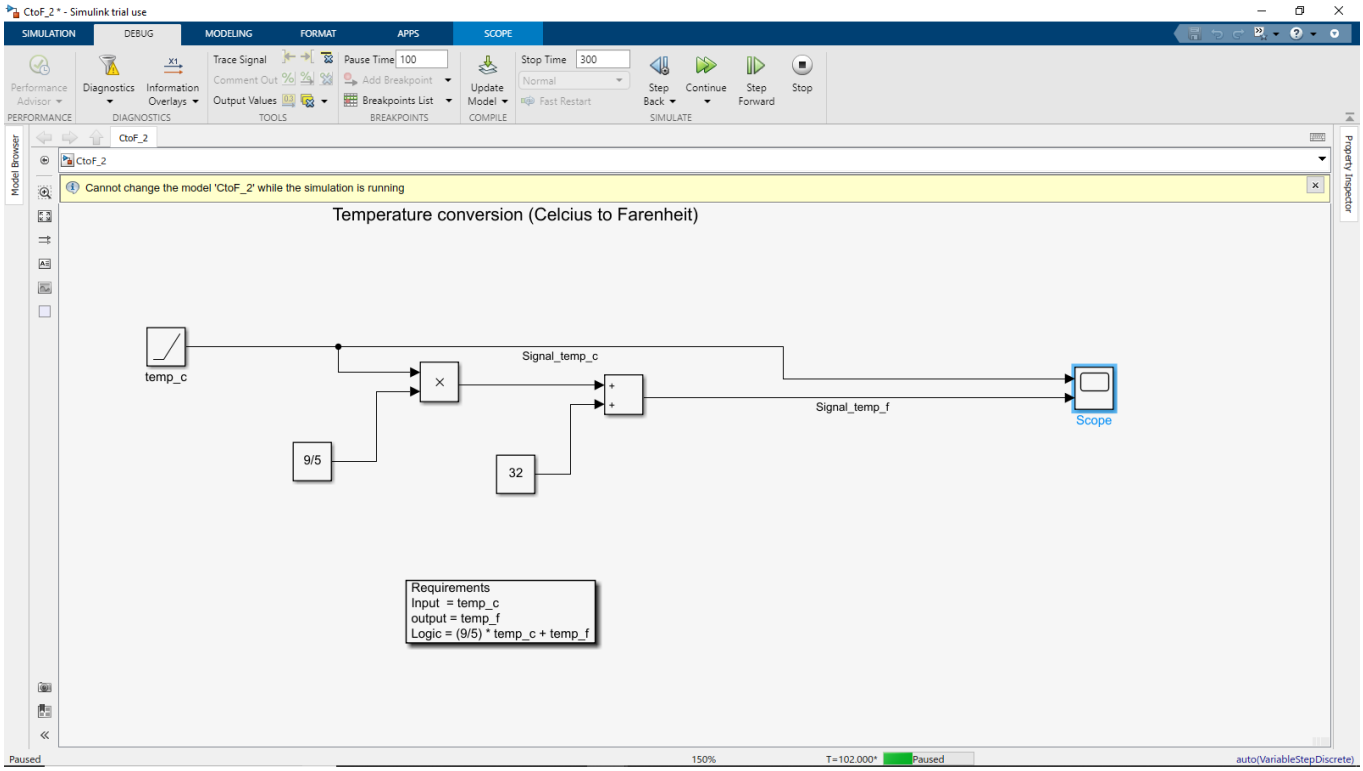


Figure 26 Passing simulation using Pause time

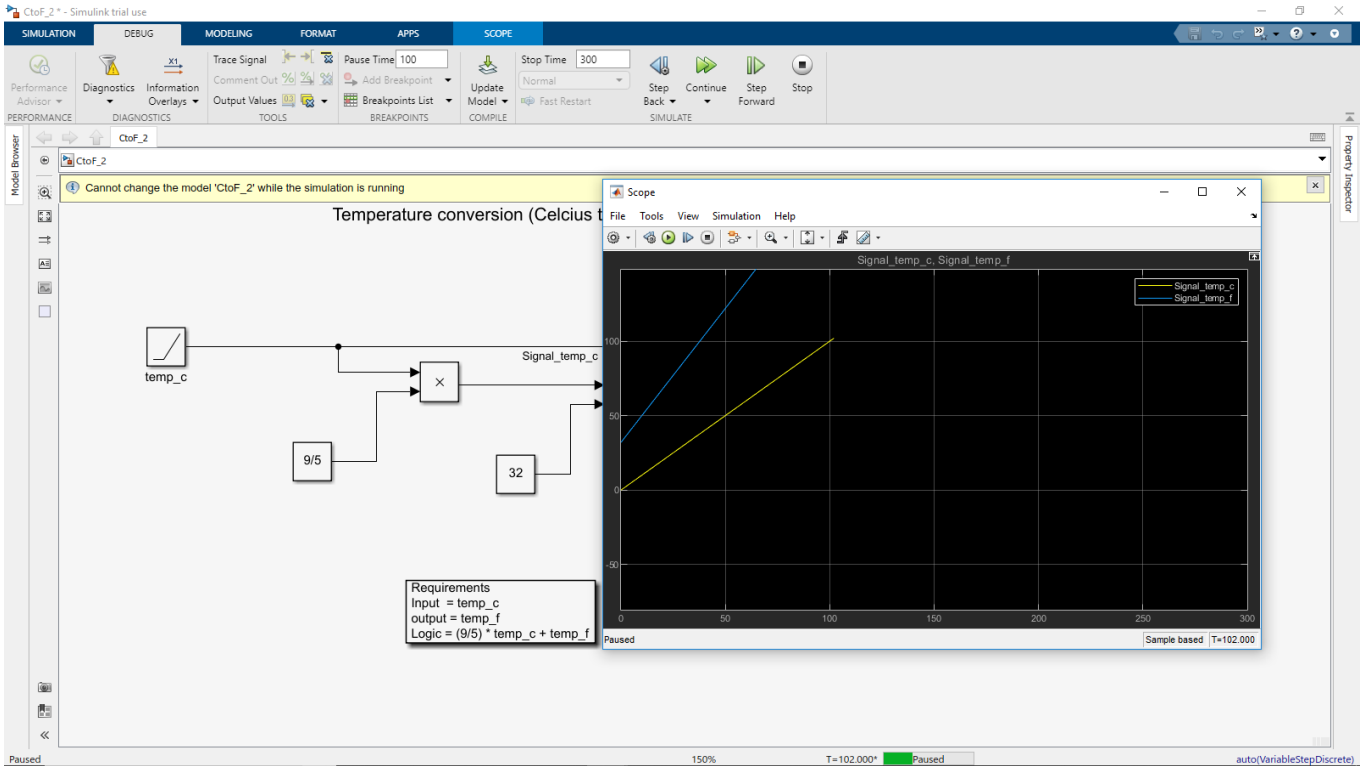


Figure 27 Scope values

Show label

Right click on signal line to add Label name.

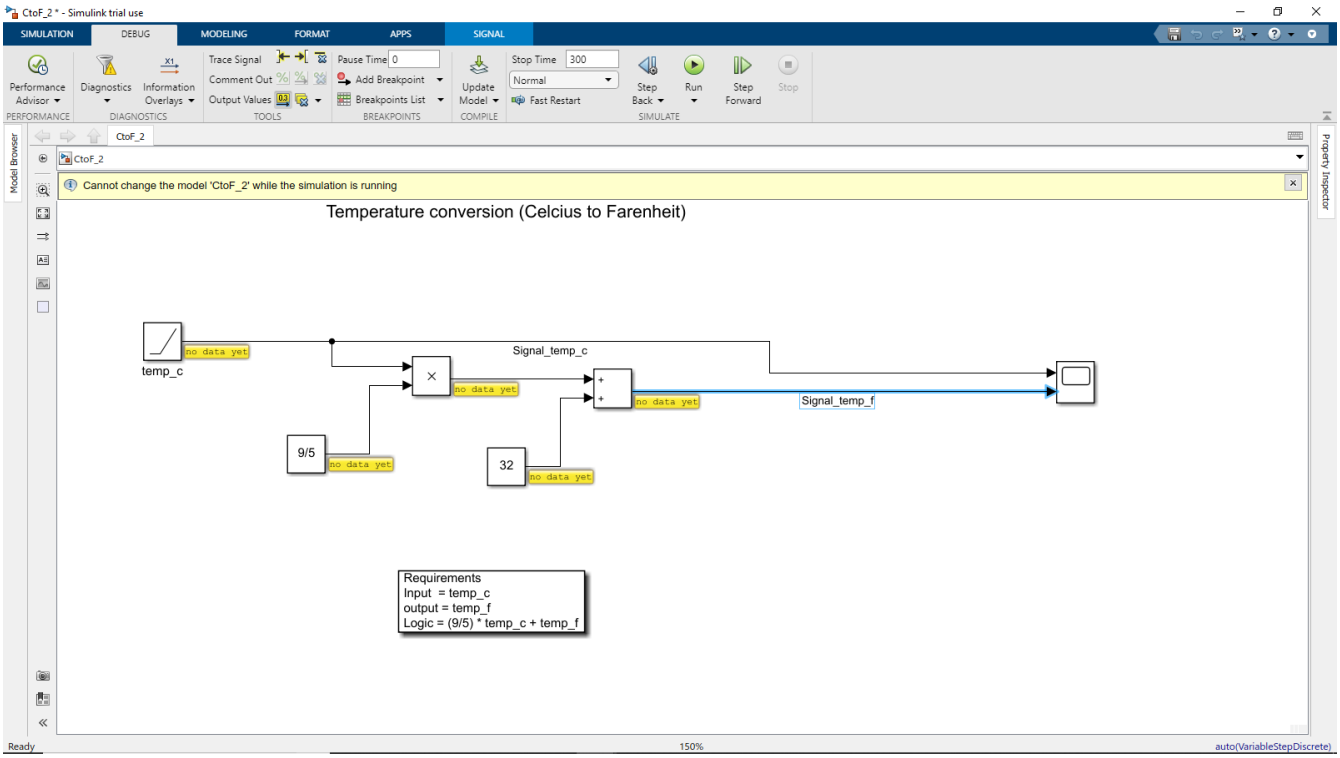


Figure 28 Label display on signal line

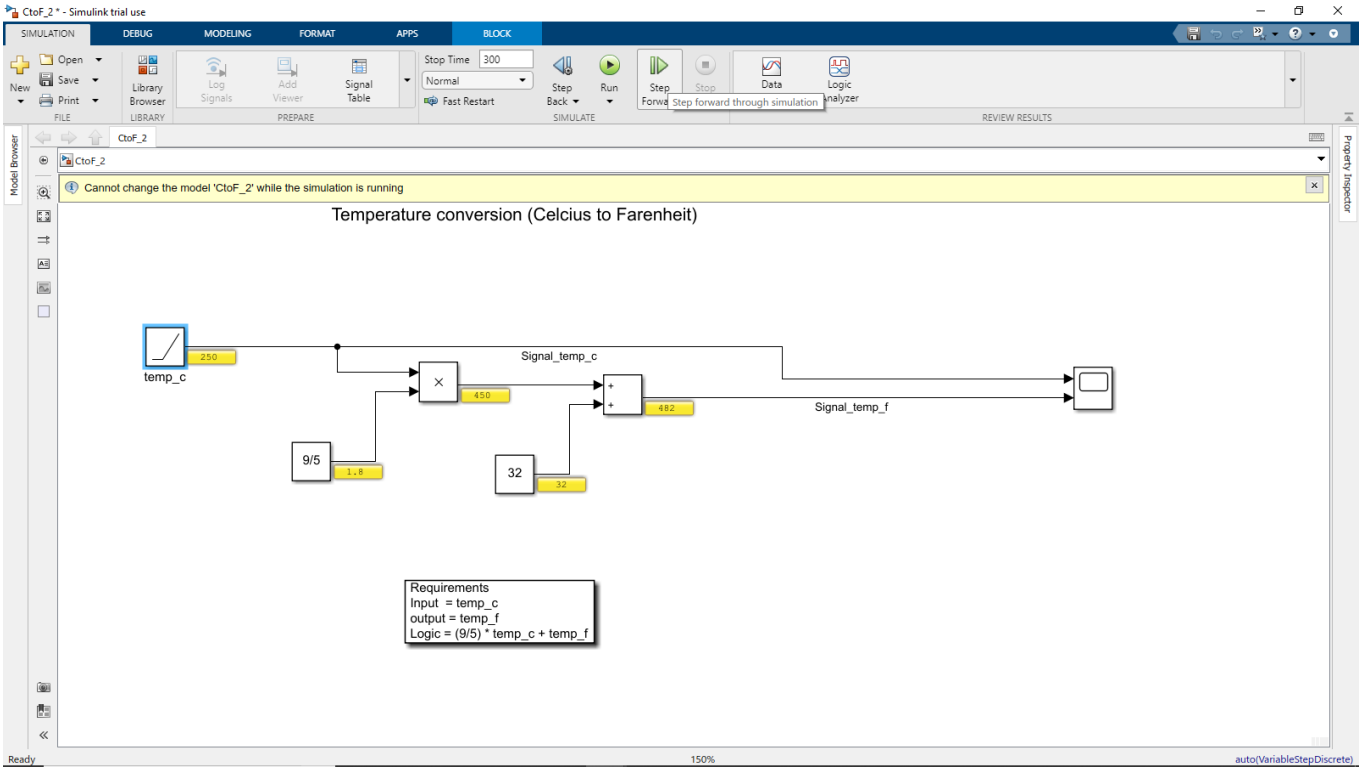


Figure 29 Label values during simulation

Adding breakpoint:

Breakpoints can be added to the signal lines to pause the execution.

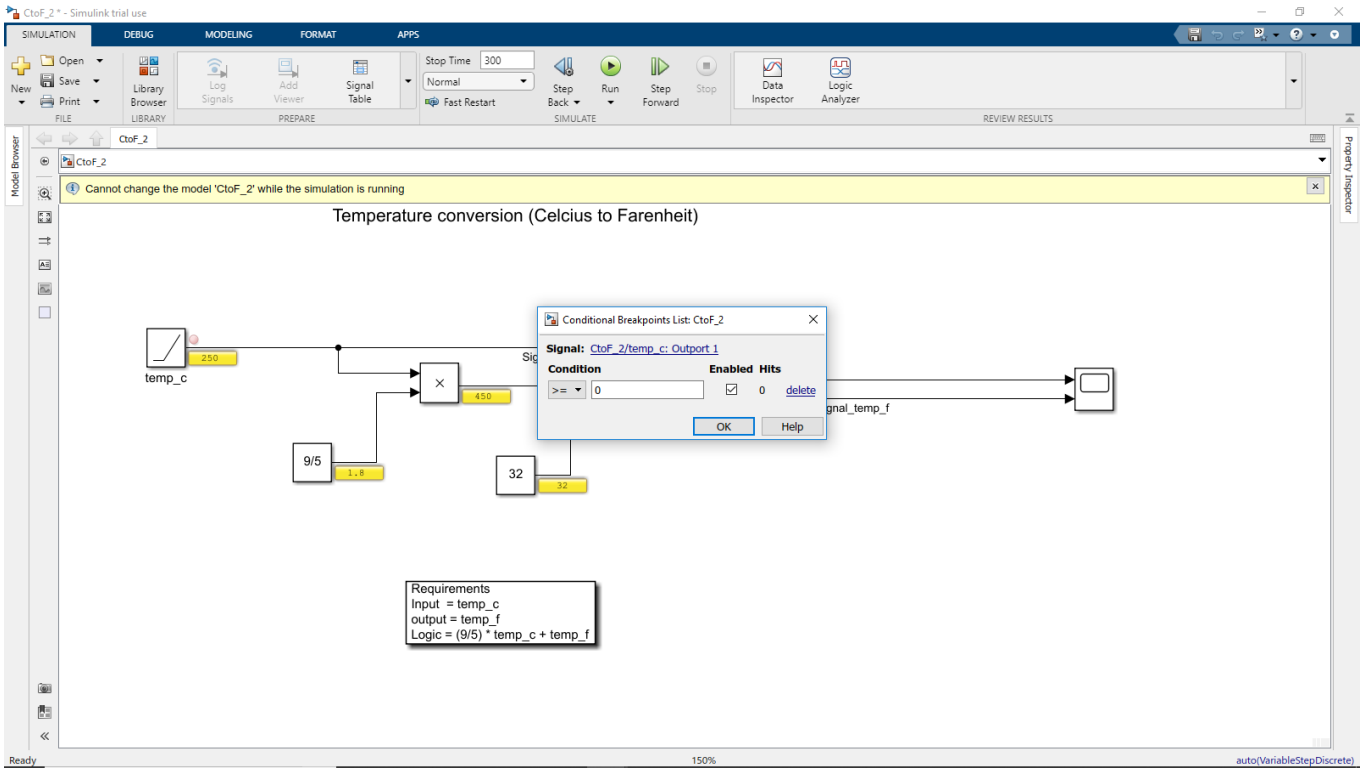


Figure 30 Breakpoint addition in signal line

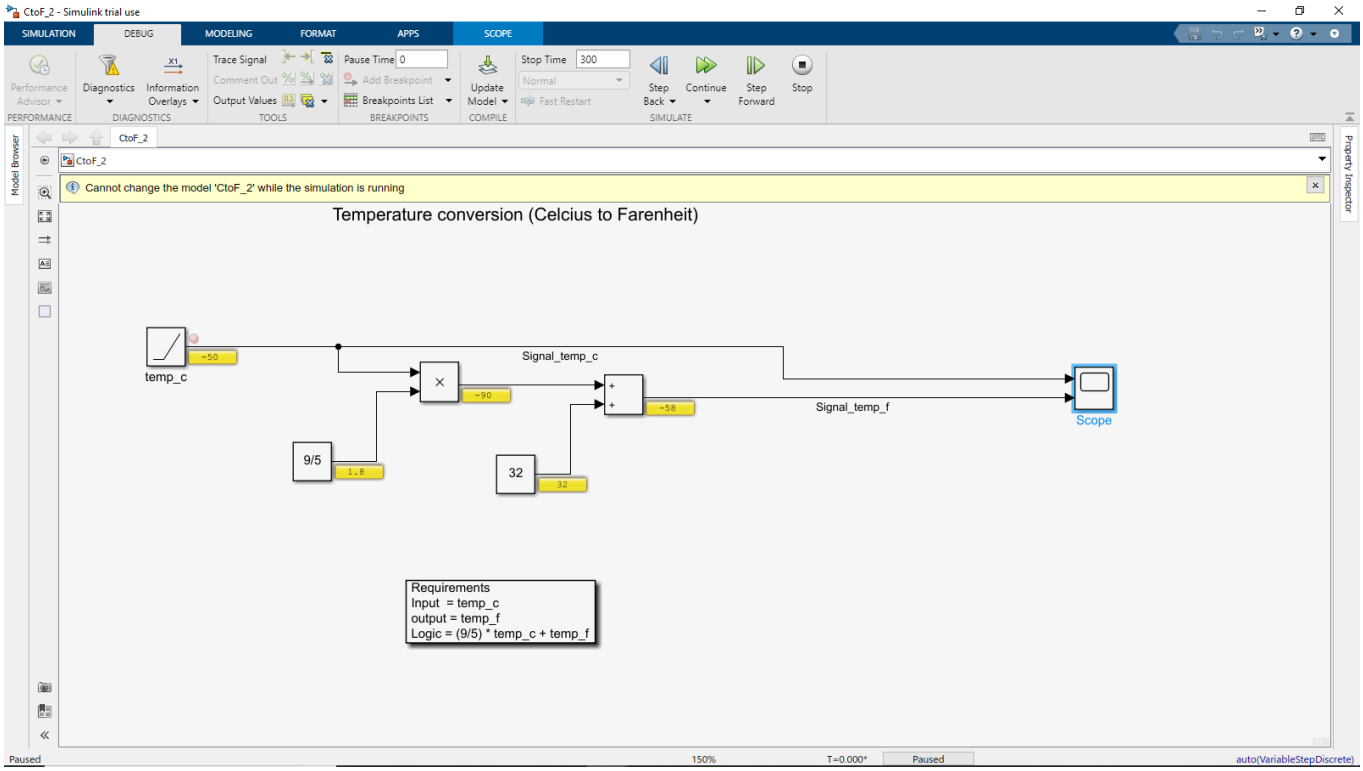


Figure 31 Breakpoint Execution

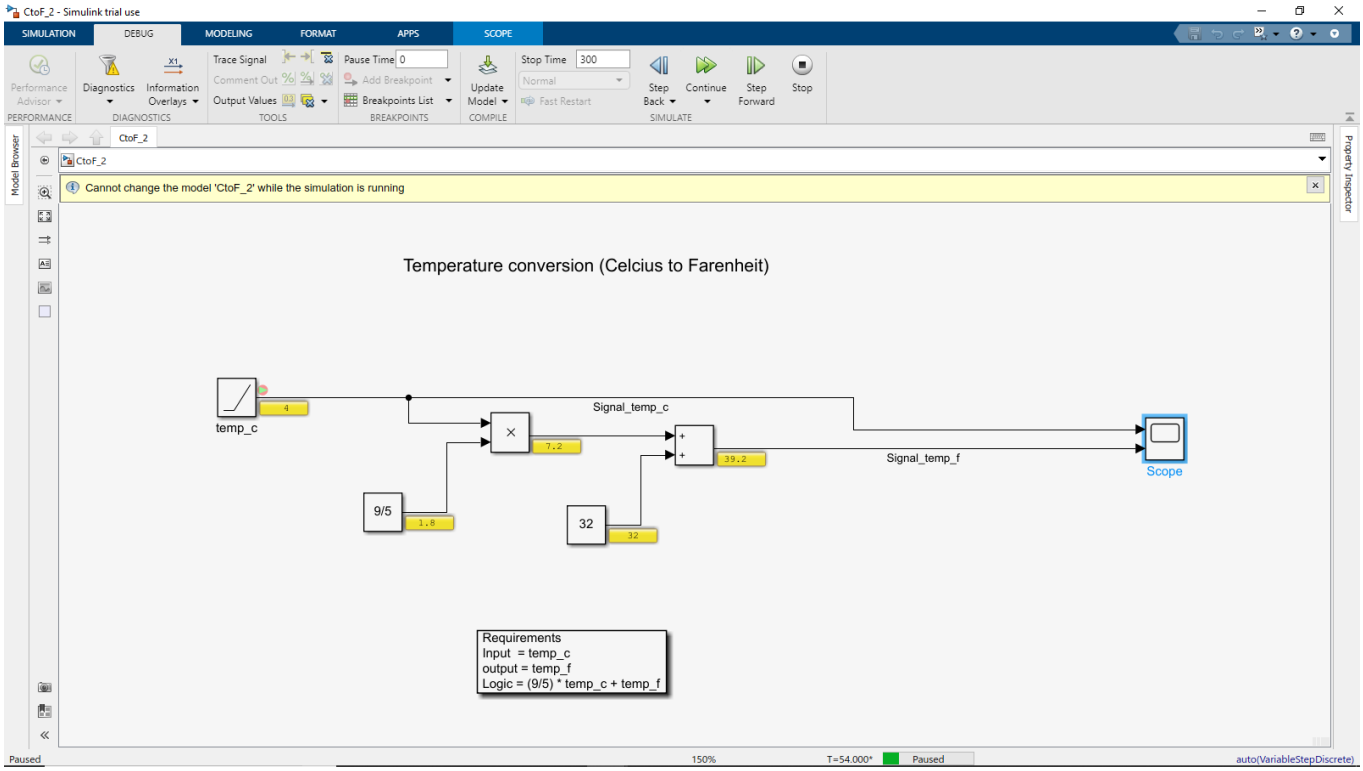


Figure 32 The execution is paused at breakpoint

Understanding Execution Order

Click on Information Overlays, to enable execution order for the system

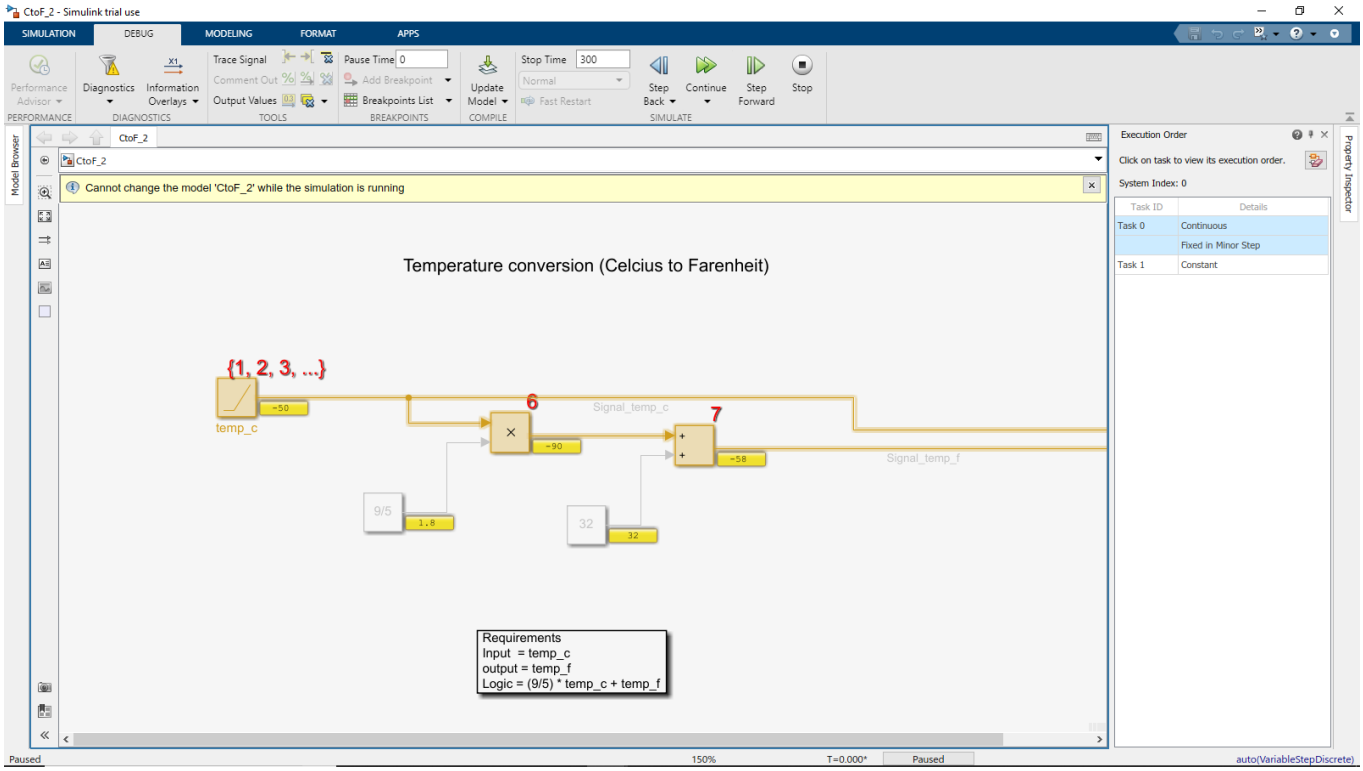


Figure 33 Execution order of the system

Sampling Time (Timing Legend)

Click on Information Overlays, to enable Legends and Colors. It differentiates constants, continuous and fixed input with different colors.

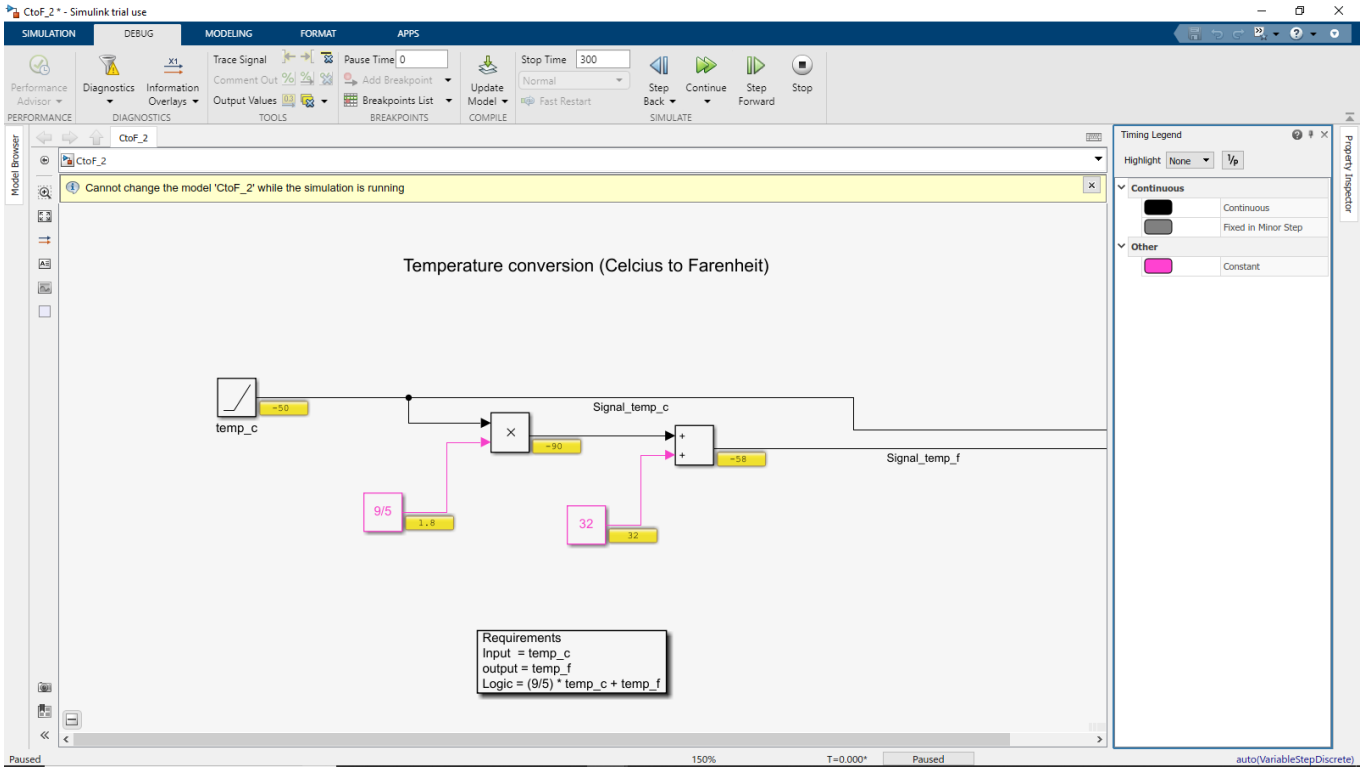


Figure 34 Colored Legend representation

Logging

Signal logging enables logging data for specified signals to the MATLAB workspace and the Simulation Data Inspector.

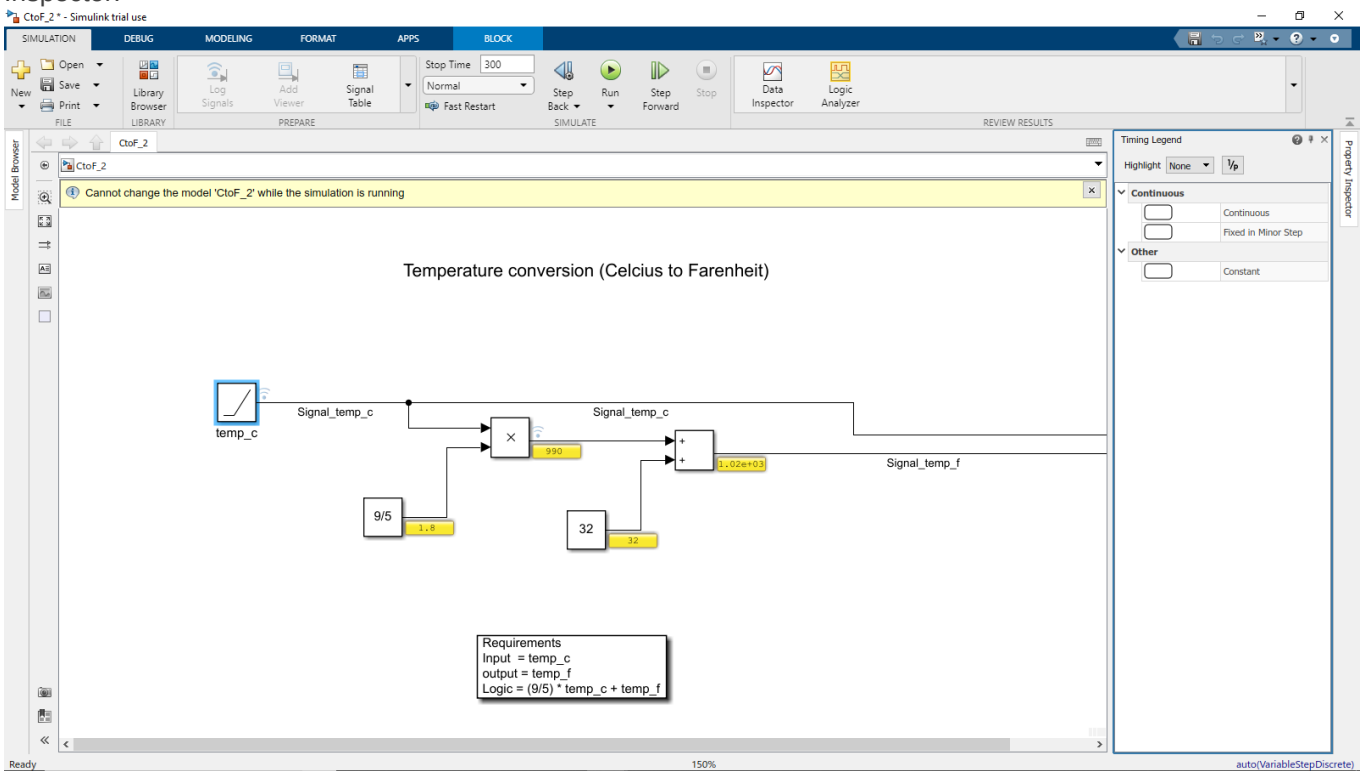


Figure 35 Logging creation at signal line

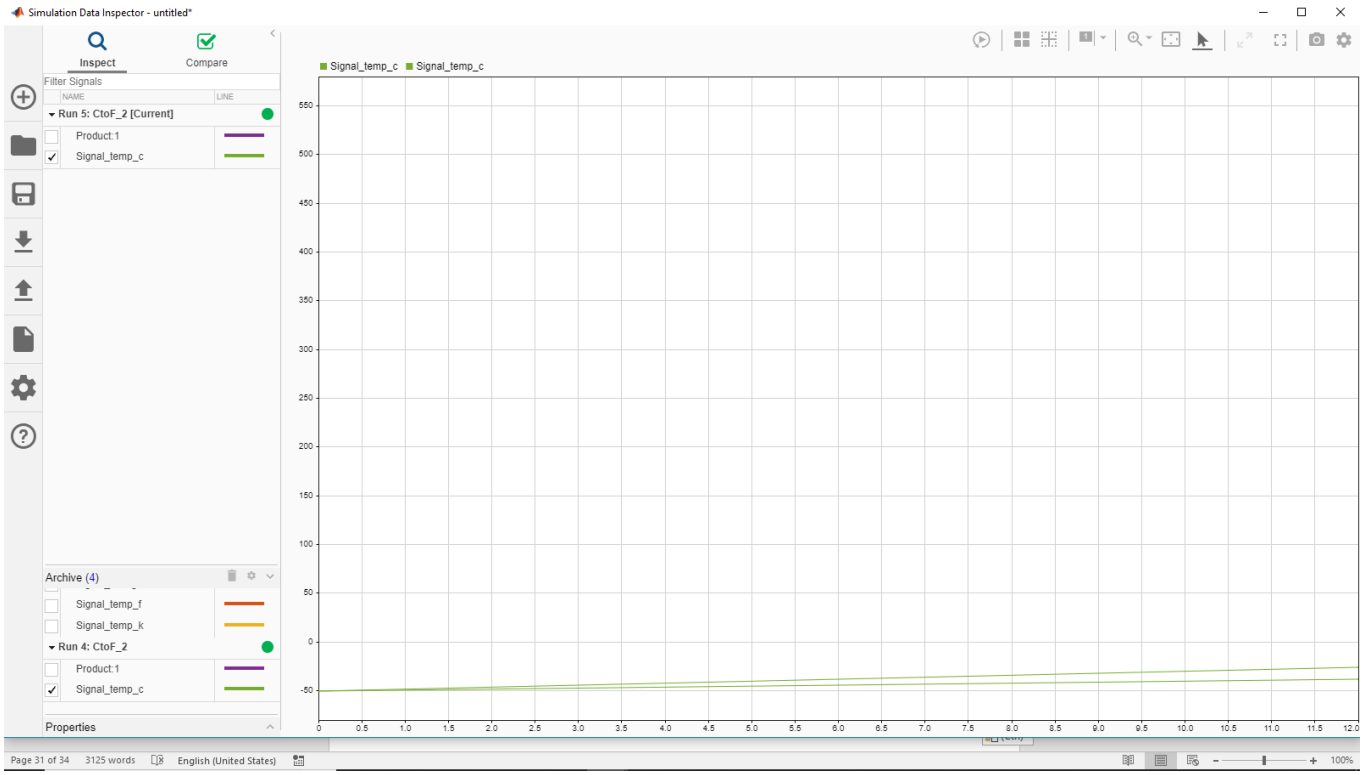


Figure 36 Log compare of two signals

Highlight to Source and Destination

Right click on any signal line to highlight the Source and Destination path.

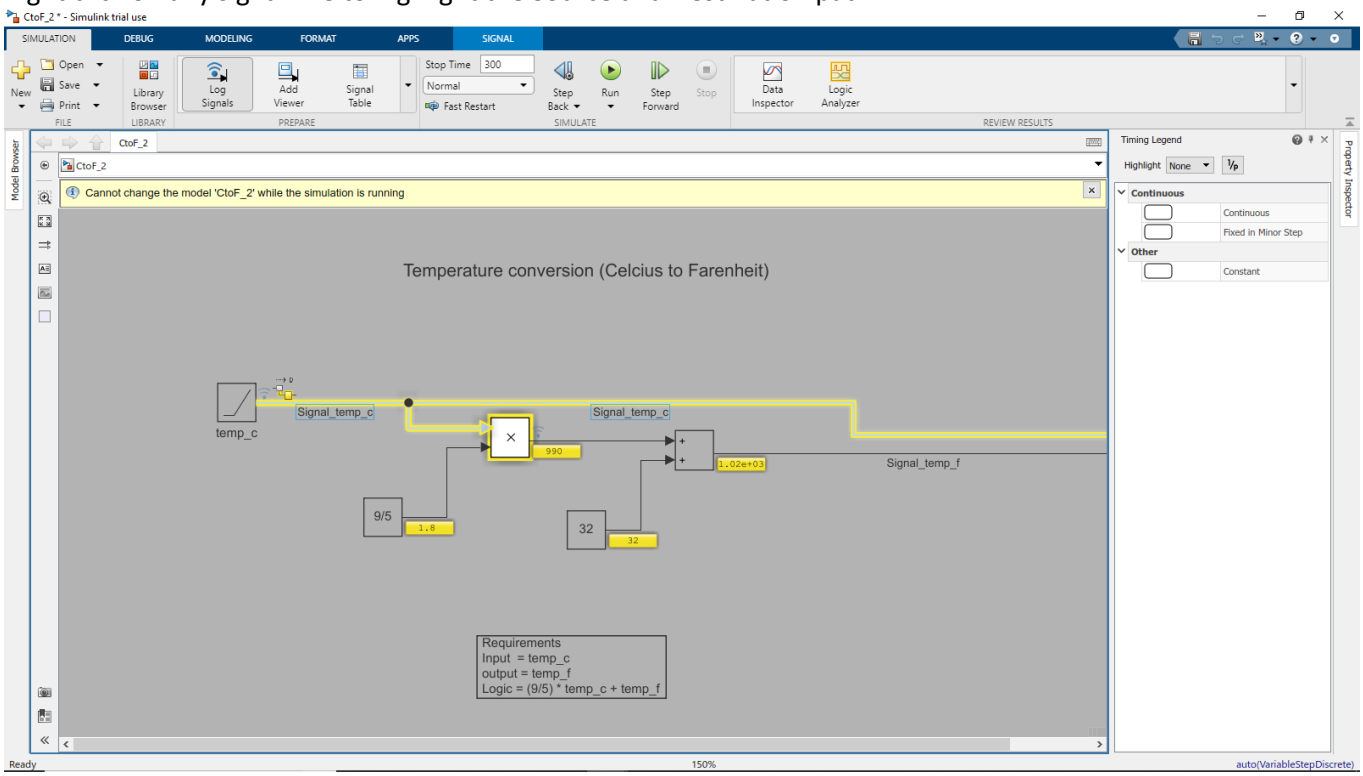


Figure 37 Highlighting source and destination

Floating Viewer/ Scope

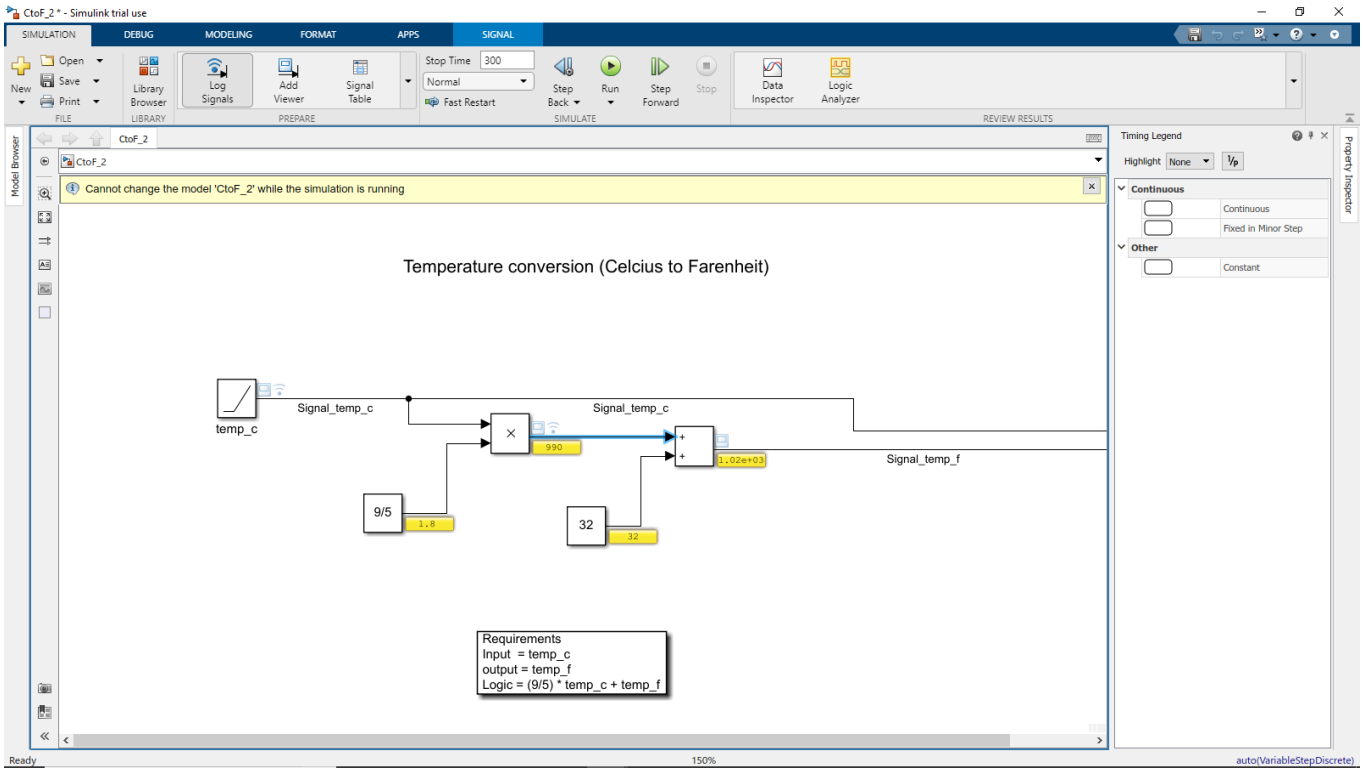


Figure 38 Floating Viewer Test Model

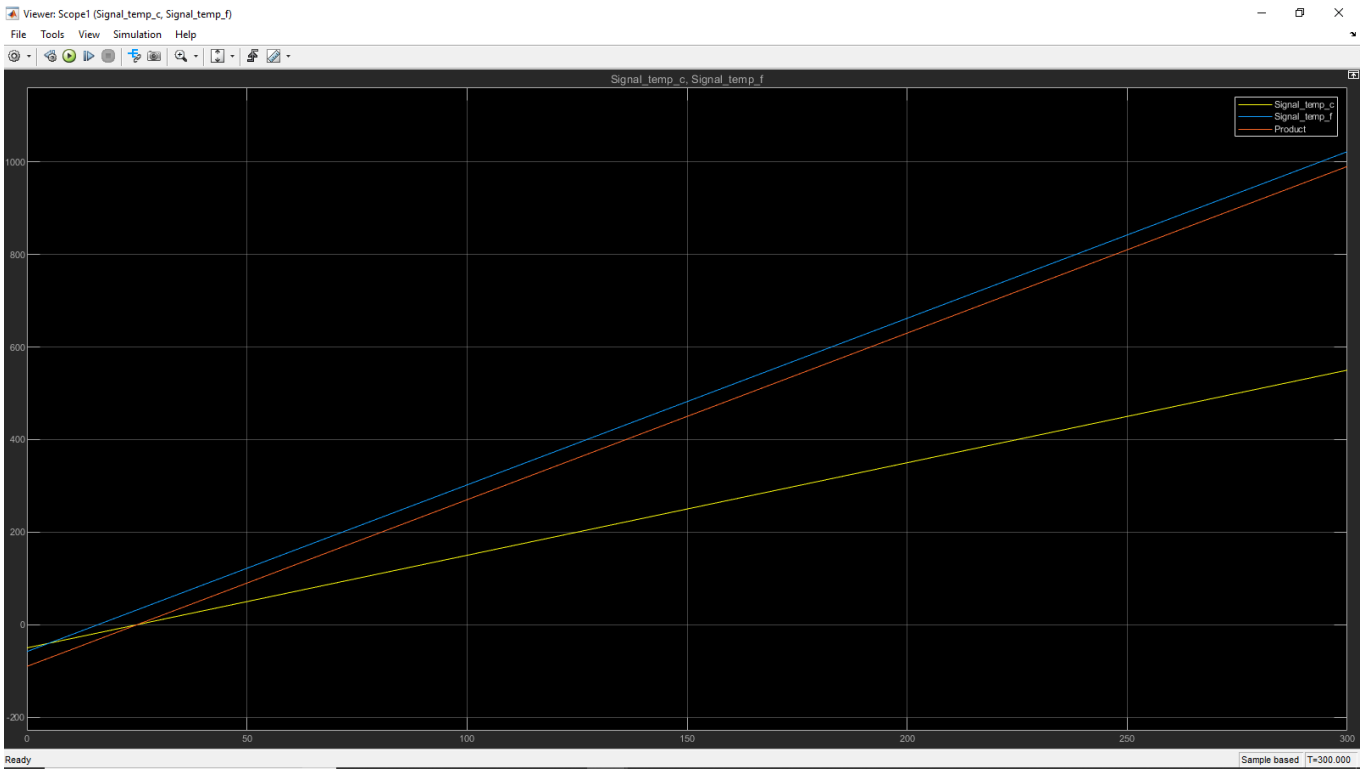


Figure 39 Floating Viewer /Scope in Signal Lines

Different Types of Subsystem

Atomic Subsystem:

Atomic Subsystem block is a Subsystem block that has its Treat as Atomic Unit parameter selected by default. A Subsystem can be created in these ways:

- Copy the Subsystem (or Atomic Subsystem) block from the Ports & Subsystems library into your model. You can then add blocks to the subsystem by opening the Subsystem block and copying blocks into its window.
- Select the blocks and lines that are to make up the subsystem using a bounding box, then choose Create Subsystem from the Edit menu. Simulink replaces the blocks with a Subsystem block. When you open the block, the window displays the blocks you selected, adding Inport and Outport blocks to reflect signals entering and leaving the subsystem.

Non Atomic subsystem:

Non-virtual subsystem Control when the contents of the subsystem are evaluated as a single unit (atomic execution). It creates a conditionally executed subsystems that run only when an event occurs on a triggering, function-call, action, or enabling input.

Triggered Subsystem:

A triggered subsystem is a conditionally executed atomic subsystem that runs each time the control signal (trigger signal):

- Either rises from a negative value to a positive value or zero, or rises from a zero value to a positive value.
- Either falls from a positive value to a negative value or zero, or falls from a zero value to a negative value.
- Rises or falls through or to a zero value.

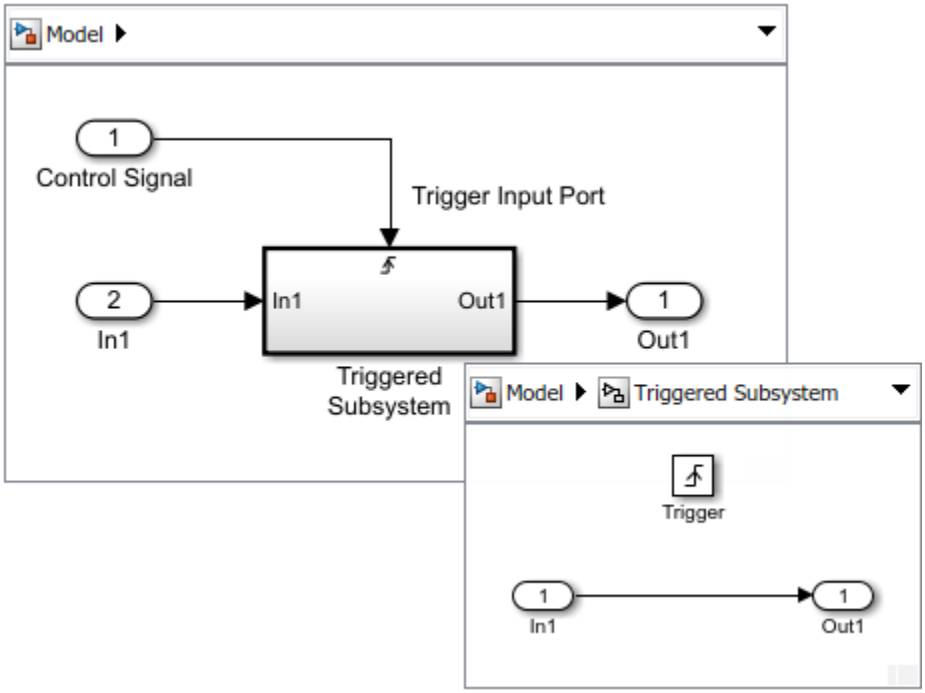


Figure 40 Triggered Subsystem

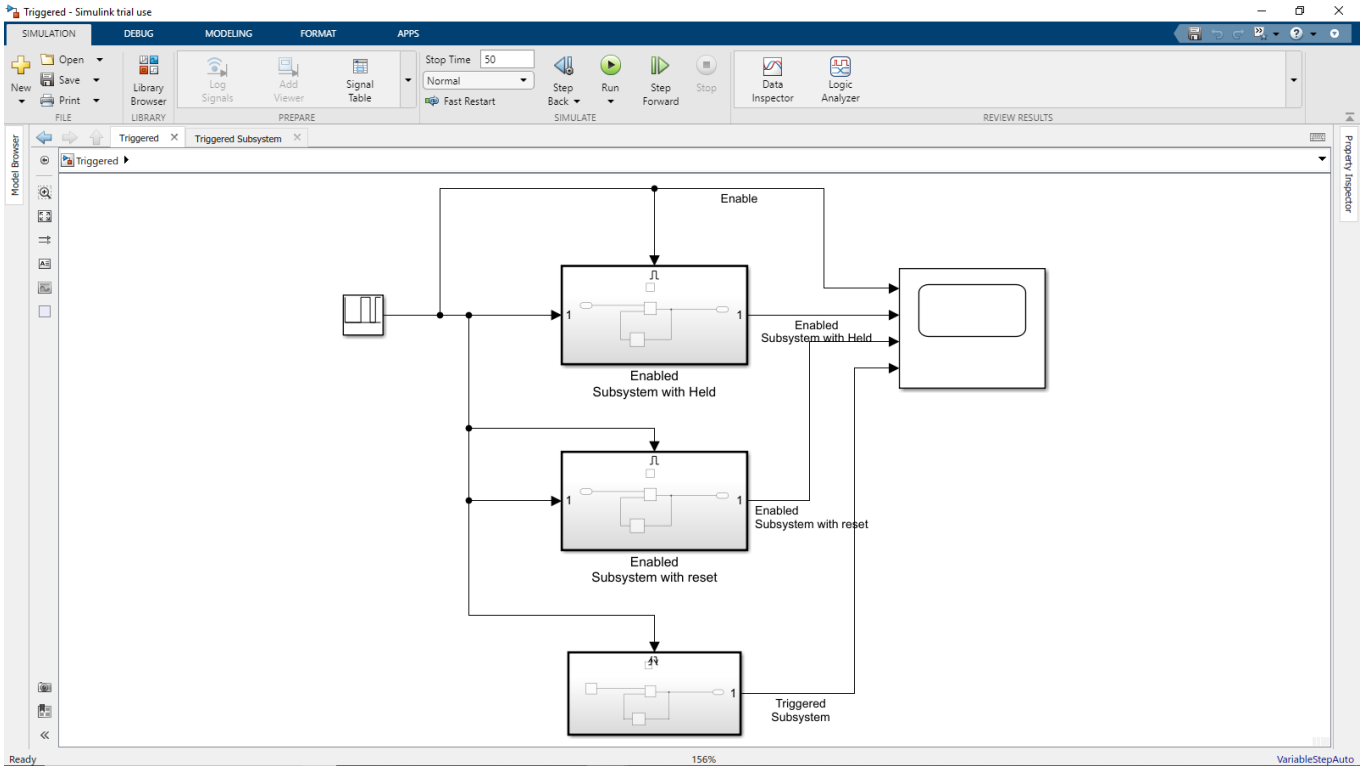


Figure 41 Enabled and triggered subsystem

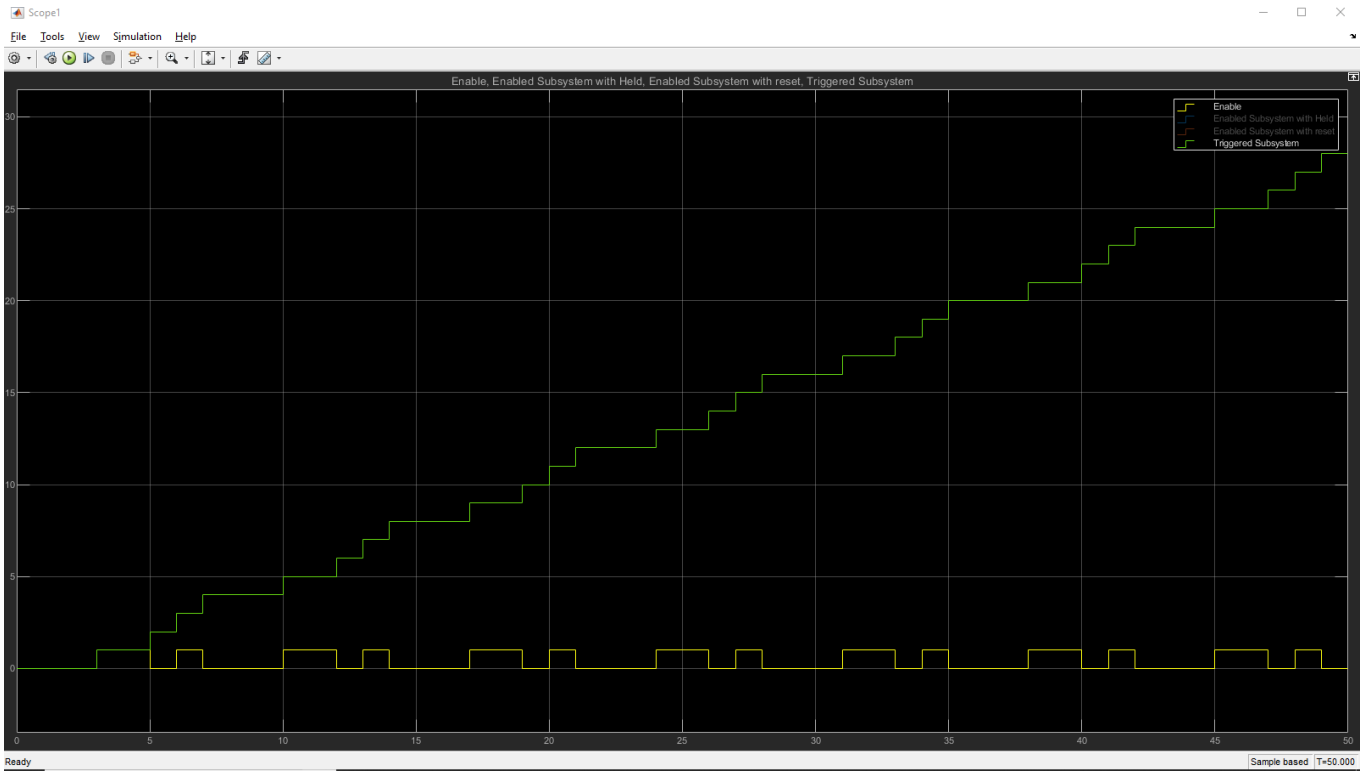


Figure 42 Scope of enabled and triggered subsystem

Enabled subsystem:

Using Enabled Subsystems: An enabled subsystem is a conditionally executed subsystem that runs once at each major time step while the control signal enabled subsystem has a positive value. If the signal crosses zero during a minor time step, the subsystem is not enabled or disabled until the next major time step.

The control signal can be either a scalar or a vector.

- If a scalar value is greater than zero, the subsystem executes.
- If any one of the vector element values is greater than zero, the subsystem executes.

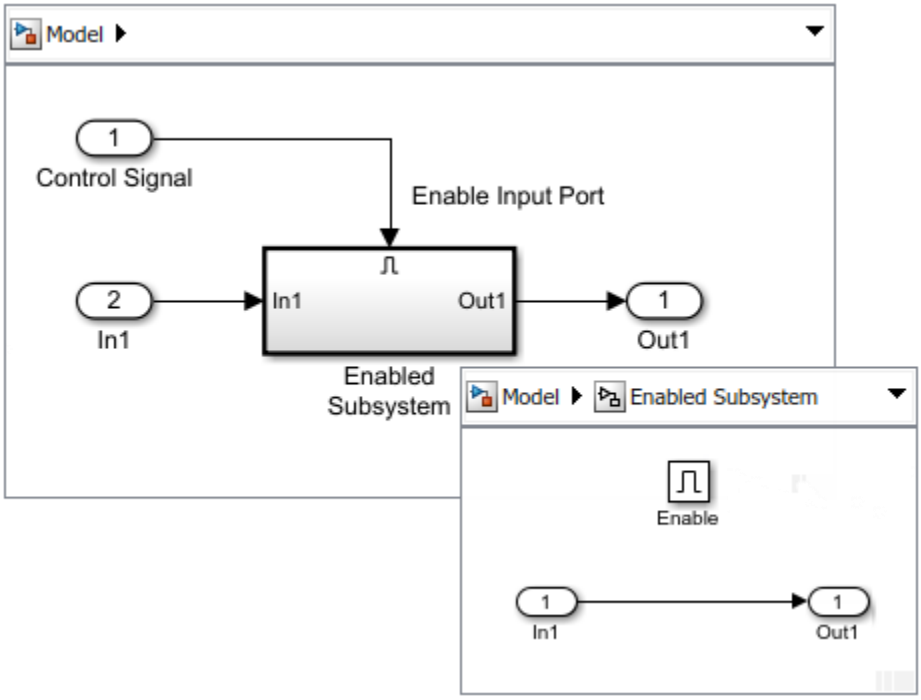


Figure 43 Enabled Subsystem

Variant subsystem:

- Variant subsystems provides multiple implementations for a subsystem where only one implementation is active during simulation.
- A Variant Subsystem block contains two or more child subsystems where one child is active during model execution. The active child subsystem is referred to as the active variant. The active variant is programmatically wired to the Inport and Outport blocks of the Variant Subsystem by Simulink during model compilation.
- To programmatically control variant selection, a Simulink. Variant object is associated with each child subsystem in the Variant Subsystem block dialog. Simulink. Variant objects are created in the MATLAB base workspace. These objects have a property named Condition, which is an expression, that evaluates to a boolean value and is used to determine the active variant child subsystem. For example, defining

  VSS_LINEAR_CONTROLLER=Simulink. Variant('VSS_MODE==1');

  in the base workspace creates a Simulink. Variant object where the constructor argument ('VSS_MODE==1') defines when the variant is active. Using the Variant Subsystem dialog, you then associate VSS_LINEAR_CONTROLLER with one of the child subsystems within the Variant Subsystem.

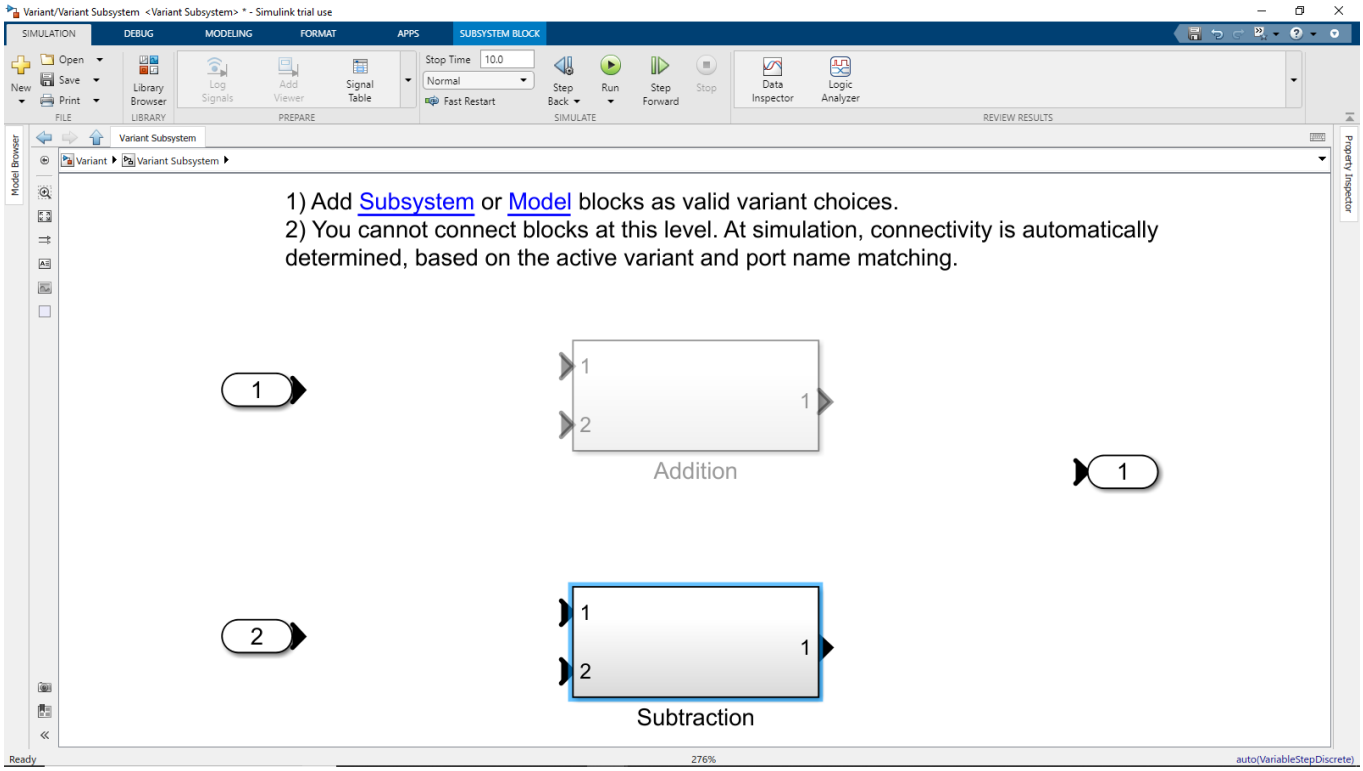


Figure 44 Variant subsystem

Simulink Onramp course certificate

Figure 45 Simulink onramp certificate

Activity 5: Stateflow

HVAC

Stateflow model

Debugging techniques in stateflow chart

Breakpoint

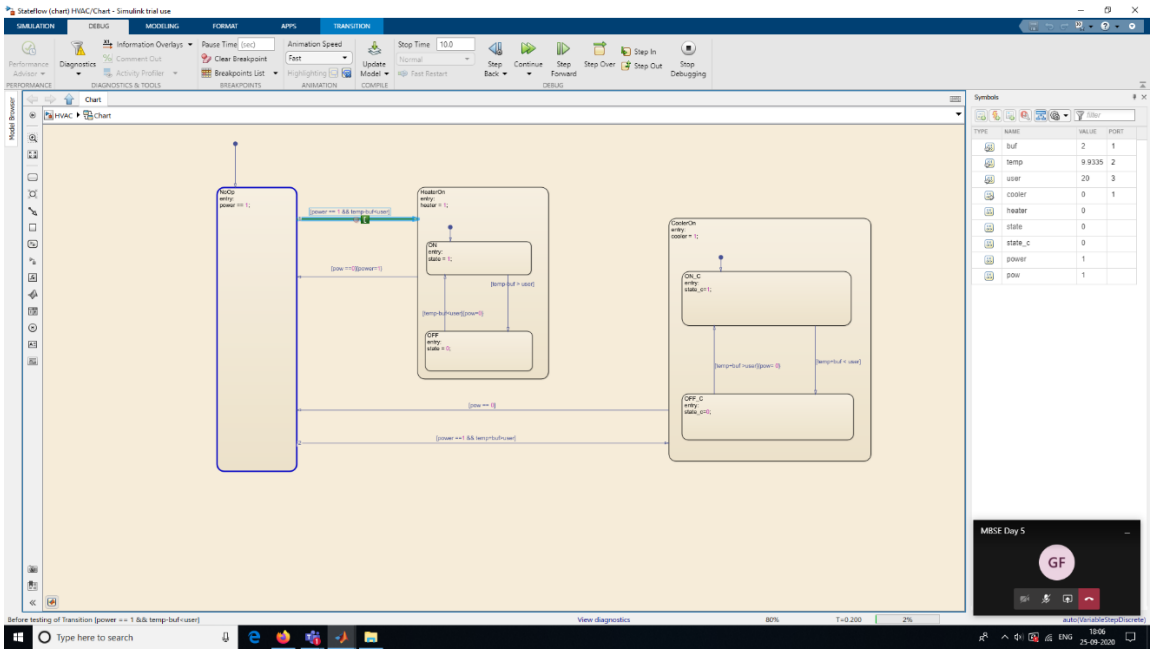


Figure 46 Breakpoint

Pause time

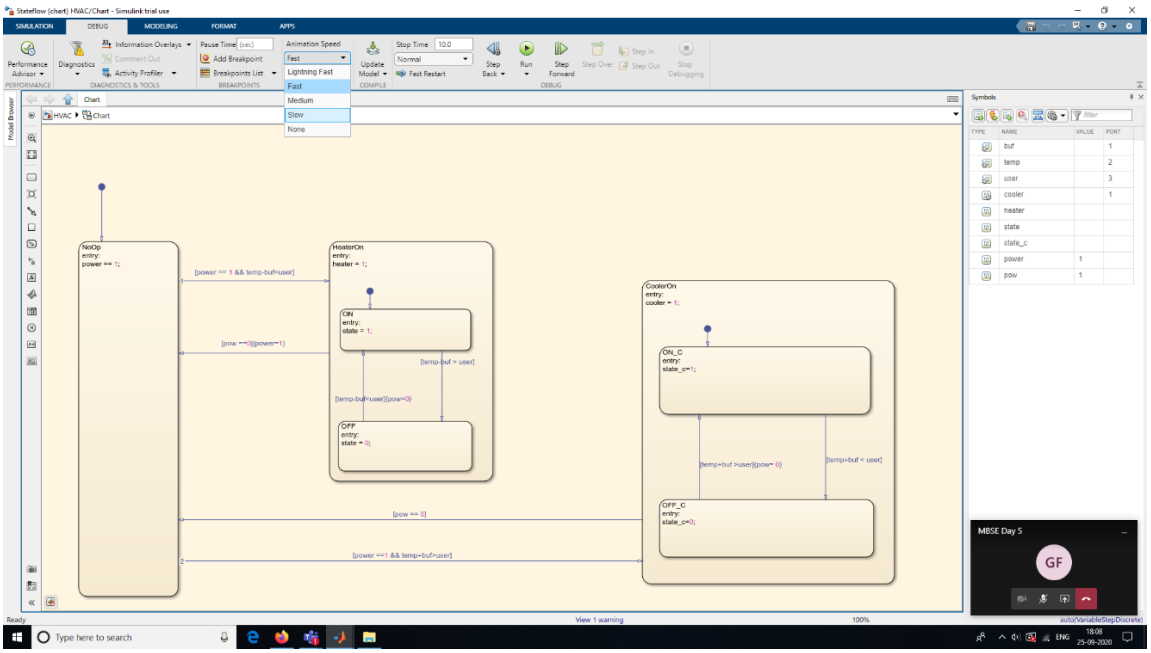


Figure 47 Pause time

Stateflow onramp course certificate

MathWorks | Training Services

Course Completion Certificate

Milind Mohapatra

has successfully completed 100% of the self-paced training course

Stateflow Onramp

DIRECTOR, TRAINING SERVICES

25 September 2020

Figure 48 Stateflow onramp certificate