Report – Model Based System Engineering

GLOBAL ENGINEERING ACADEMY

Genesis

L&T Technology Services

Name: V.S.SHANDHIYA
PS No: 99002477
Module: Model Based System Engineering

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	22-sep-2020	SHANDHIYA.V.S, 99002477	99002481, 99002486	Prithvi, Bhargav N	
2	23-sep-2020	SHANDHIYA.V.S, 99002477	99002481, 99002486	Prithvi, Bhargav N	
3	26-sep-2020	SHANDHIYA.V.S, 99002477	99002481, 99002486	Prithvi, Bhargav N	

Contents

TABLE OF FIGURES:

ACTIVITY 1
MODEL BASED SYSTEM ENGINEERING

ACTIVITY 1: MODEL BASED SYSTEM ENGINEERING

MBSE IN AUTOMITIVE INDUSTRY

INTRODUCTION:

Commercial automobiles have turned into very complex high-technology products in a relatively short time span. Different factors contribute to this complexity. One of them is the increasing number of vehicle functionalities supported by software, electronics and mechatronic technologies, a trend that does not seem to slow down.

REQUIREMENTS:

- To develop future autonomous vehicles, companies must adopt model-based systems engineering (MBSE) methodologies.
- MBSE is a formalized application of modeling to support system requirements, analysis, design and validation and verification (V&V) activities, beginning in the conceptual design phase and continuing through Out development and later lifecycle phases.
- With the MBSE approach, engineers can capture the needed complexity and behaviors of advanced real-time features for autonomous vehicles; it enables development teams to understand an area of interest or concern and provide unambiguous communication among interested parties.
- Now more than ever, the MBSE approach is necessary to address the complex multidisciplinary, multi-domain process of autonomous vehicle development, which must meticulously orchestrate mechanical, electrical and electronic, and software and controls engineering.

DESIGN ARCHITECTURE:

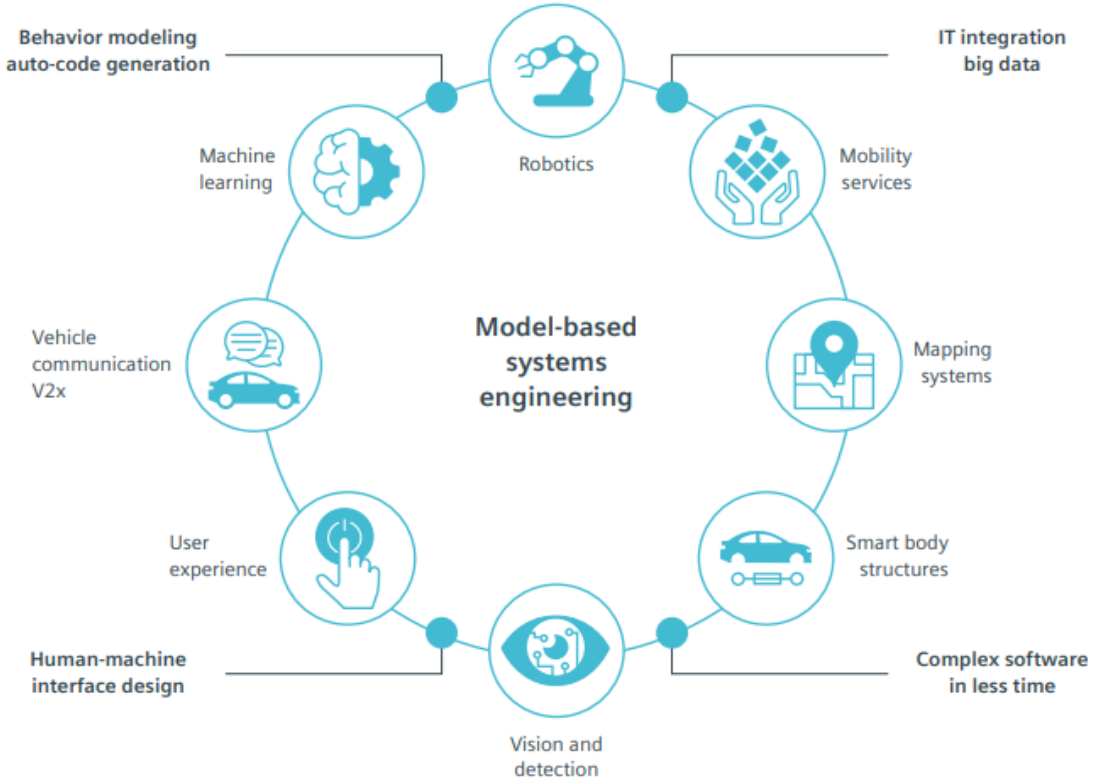


Figure 1-Design Architechture

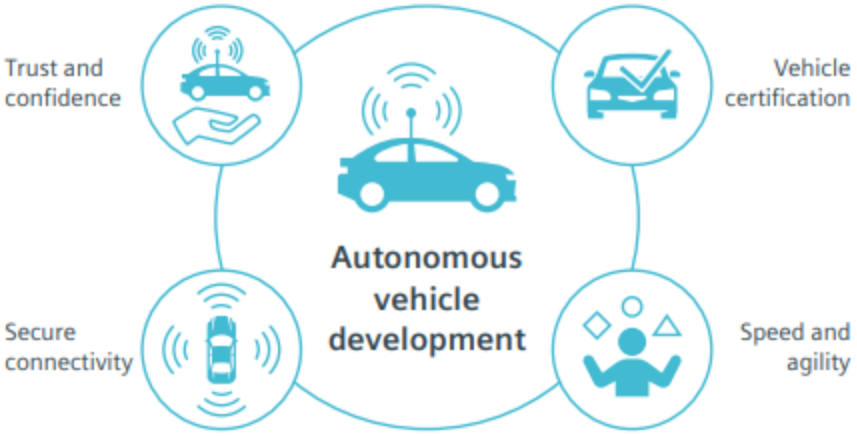


Figure 2-Automation model

MBSE economic analysis methodology: process flow

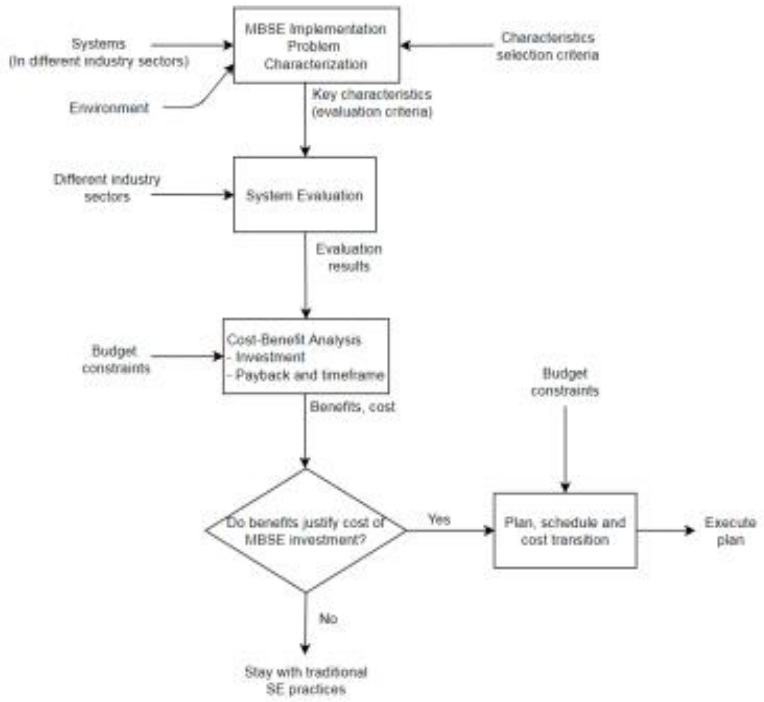


Figure 3-Process Flow

COST VS TIME CURVE:

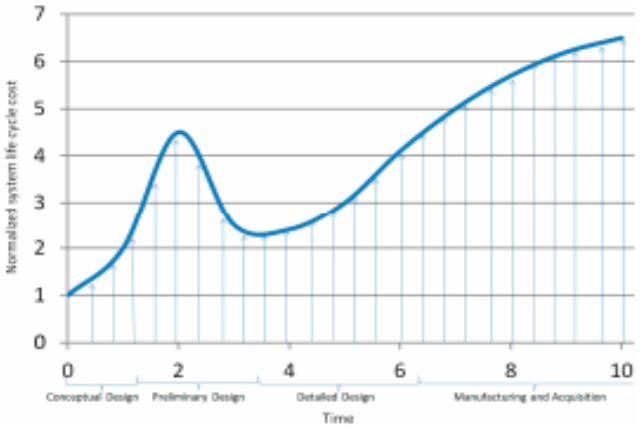


Figure 4-Cost vs Time graph

MBSE TOOLS

Commercial MBSE + SysML Tool Reviews

1. Enterprise Architect - MBSE + SysML Support (Sparx Systems)
2. MagicDraw - MBSE + SysML Support (No Magic)
3. Cameo Systems Modeler - MBSE + SysML Support (No Magic)
4. Visual Paradigm (Visual Paradigm)
5. Rational Rhapsody Designer (IBM)

MODELLING TOOLS BASED ON THEIR SPECIFIC JOB

1. **Brainstorming-*Requirements***

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people.

- Document Analysis
- Focus Group
- Interface analysis
- Interview
- Observation
- Prototyping
- Requirement Workshops

2. **IBM Rhapsody -*System Medalling and sub-system modelling***

IBM Rhapsody for Systems Engineers is an integrated, model-driven systems engineering environment for complex projects. It uses Systems Modeling Language (SysML) and Unified Modeling Language (UML) to enable rapid requirements analysis and visual, model-driven design.

3. **Cameo Systems Modeler- *System Modelling***

No Magic's Cameo Systems Modeler is a Model-Based Systems Engineering (MBSE) solution in one easy-to-use package, enabling single users or an entire engineering team to create, collaborate, and manage systems requirements and designs.

4. **Capella Open Source MBSE tool-*Function Simulation tool***

Capella is an Open Source MBSE tool implementing the Arcadia method. It is a comprehensive, extensible and field-proven MBSE solution focusing on the design of systems architectures. Capella is strongly driven by the current practices and concerns of system engineering practitioners and can be easily mastered by those familiar with SysML language.

5. Enterprise Architect with SysML plugin-*Analysis Models*

With advanced modeling capabilities, low cost and a wealth of innovative features, Enterprise Architect combined with MDG Technology for SysML is the premier team-based modeling environment for the System Engineer.

6. Papyrus-*Test Simulation Models*

Papyrus is aiming at providing an integrated and user-consumable environment for editing any kind of EMF model and particularly supporting UML and related modeling languages such as SysML and MARTE.

7. CORE-*Hardware Simulation Model*

CORE provides engineers with a powerful solution for building highly complex system models with rich connectivity across domains. Supported by a robust simulation engine, CORE provides end-to-end coverage of the system development process from requirements to V&V.

8. ARTiSAN Studio-*System integration models*

Artisan Studio, provides complete support for OMG UML and SysML in a single, integrated toolset. You can create consistent, high quality models for systems and software engineers to communicate requirements, design decisions and alternatives across the entire team, regardless of their location. This enables your team to Work-as-One™, modeling systems and software throughout the entire project lifecycle.

3. MIL, SIL, PIL and HIL testing

- MIL, SIL, PIL and HIL testing come in the verification part of Model-Based Design approach after you have recognized the requirement of the component/system you are developing and they have been modelled in simulation level (e.g. Simulink platform).
- Before the model is deployed into the hardware for production, few verification steps take place (in which MIL, SIL, PIL and HIL comes) which are listed below.

- Here, I am taking an example of designing a Controller for a DC motor and putting the code generated from the model of the Controller in a supported System-on-Chip.

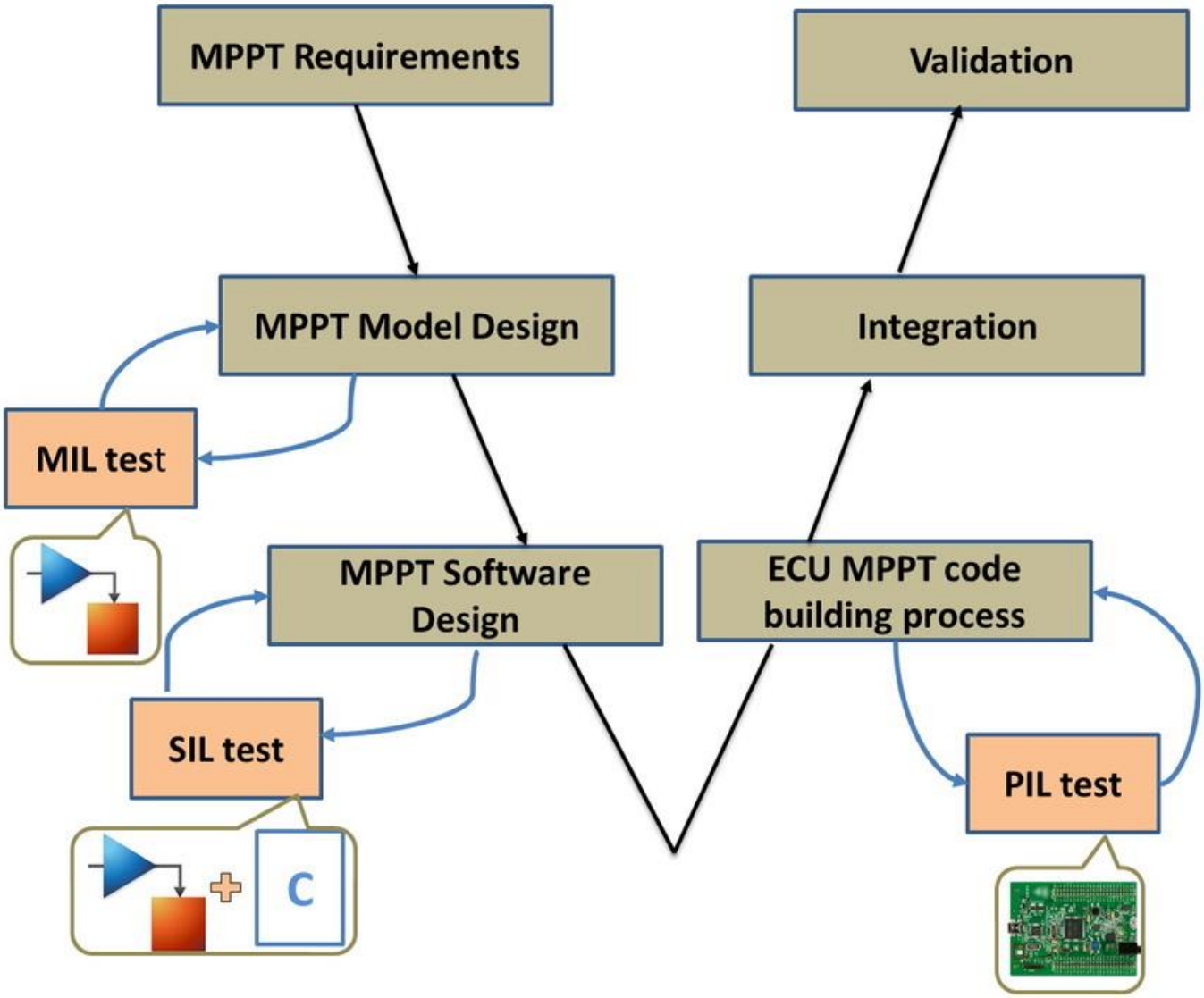


Figure 5- Development of V model

ACTIVITY 2: APPLIED LEARNING

ACTIVITY 2: APPLIED LEARNING

LOGICAL AND RELATIONAL OPERATORS:

Problem Statement:

Scenario:

Design an Automatic Car Headlight Switcher, which turns ON/OFF and switches between low and high beam depending on the intensity of light on the vehicle.

- Input Required:
  - Intensity of Light
    - ✓ None
    - ✓ Low
    - ✓ Medium
    - ✓ High
  - Power: ON/OFF

- Outputs Required:
  - Headlight: ON/OFF
  - Beam: Low beam and High beam
  - Operations to be performed:

Solution:

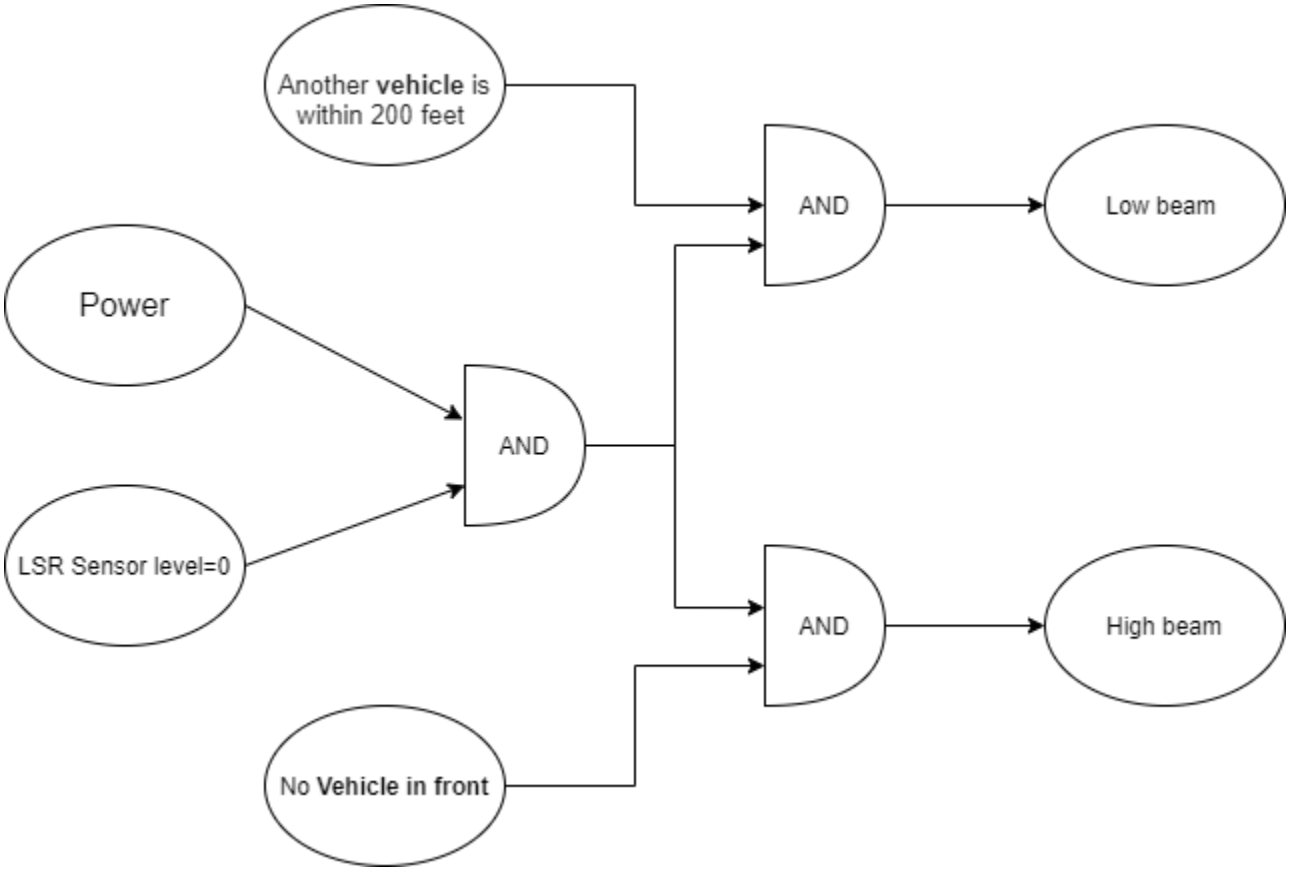


Figure 6-Logical and Relational Operators

TRIGONOMETRIC OPERATORS

Design an equation to find out the height(h) and volume(v) of a right-angled cone, given its radius(r) and slant height(s). Given r=6cm, s=10cm.

- Inputs Required:
  - Radius (r)
  - Slant height (s)
  - Theta
- Outputs Required:
  - Height (h)
  - Volume of the cone
  - Operations to be performed:

Output:

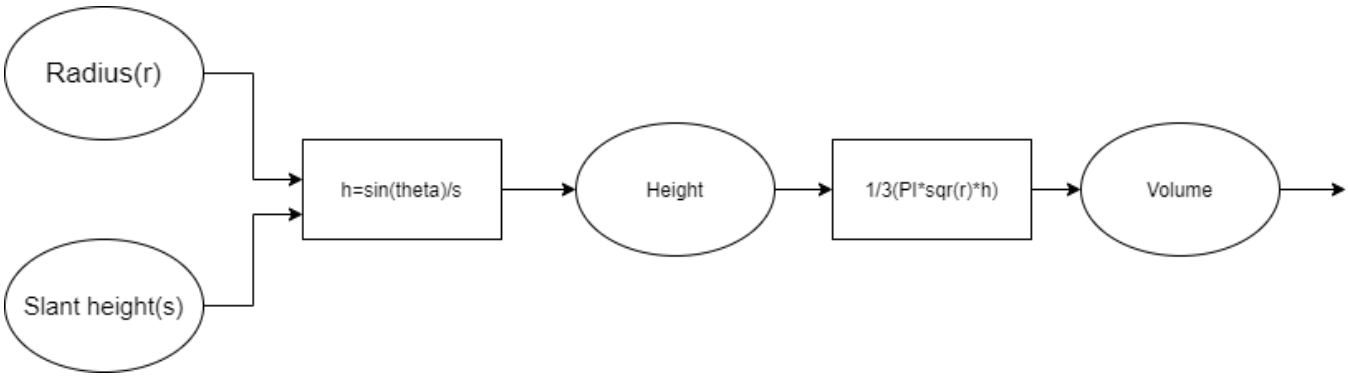


Figure 7- Trigonometric Operator

DIFFERENTIAL EQUATION:

FIRST ORDER

INPUT=R
OUTPUT=C
EQUATION:
C(s)=R(s)(1/(s+1) +H(s))

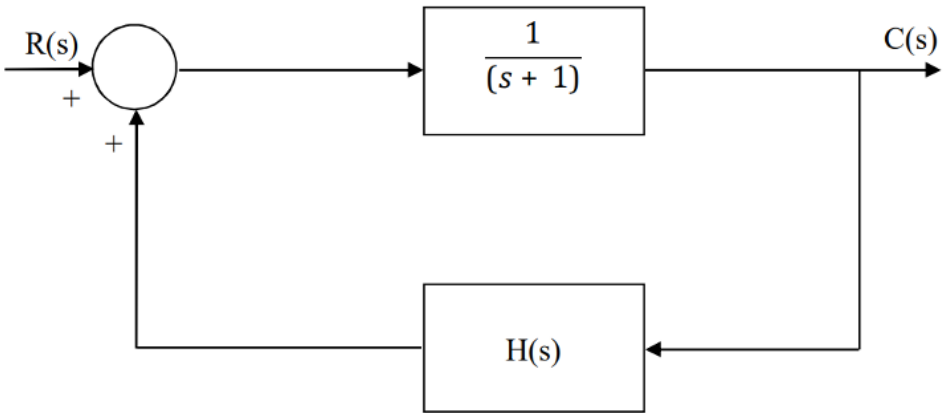


Figure 8- First Order Differential

SECOND ORDER

INPUT=U
OUTPUT=Y
EQUATION:
Y(S)=U(s)(C(Si-A) B+D

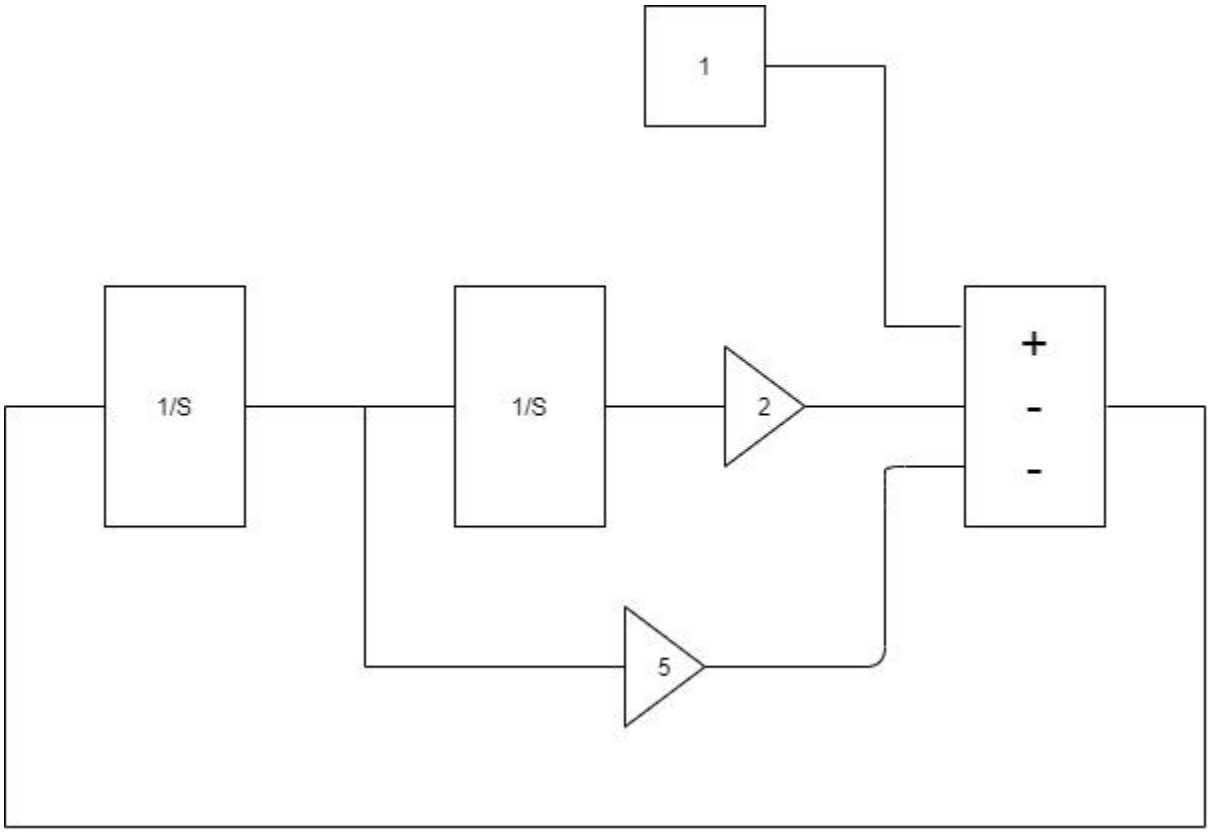


Figure 9- Second Order Differential

DIFFERENCE EQUATION

A linear **differential equation** is of first degree with respect to the dependent variable (or variables) and its (or their) derivatives. As a simple example, note dy/dx + Py = Q, in which P and Q can be constants or may be functions of the independent variable, x, but do not involve the dependent variable, y.

EQUATION:

X[K]=X[K-1] +2

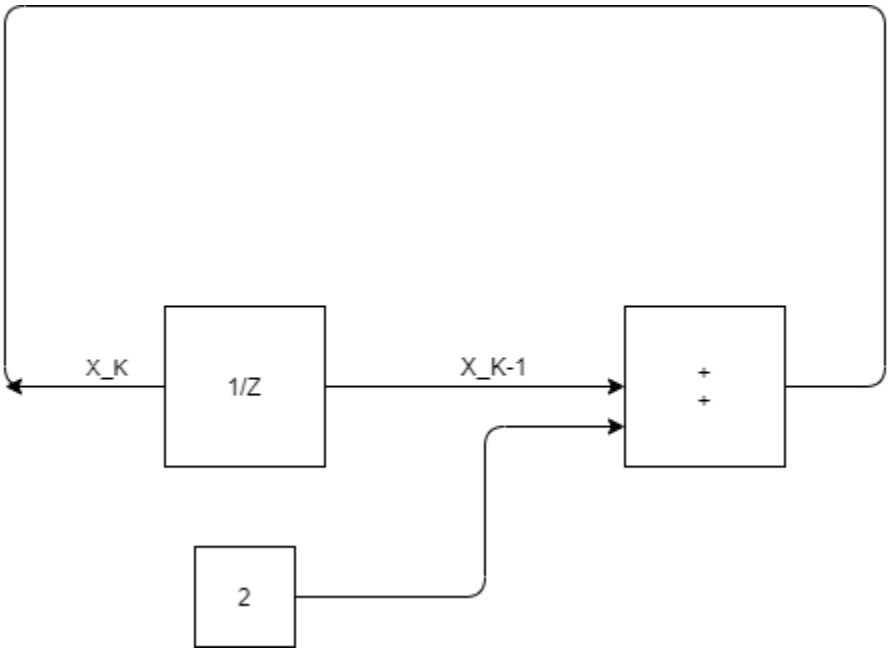


Figure 10- Difference Model

ACTIVITY 3: MATLAB

ACTIVITY 3: MATLAB

ALGORITHM DEVELOPMENT

Link for data sheet: Link1
Link for Script: Link2
Link for document: Link3

AUTOMATION SCRIPT

Link for document: Link

MATLAB ONRAMP

Figure 11- Matlab Onramp

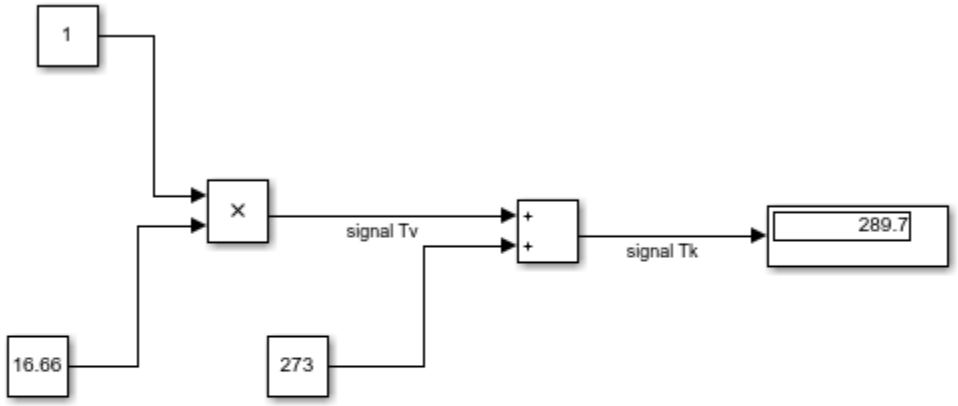
ACTIVITY 4: SIMULINK

ACTIVITY 4: SIMULINK

IN CLASS LEARNING

1.Voltage to Kelvin
2.Kelvin to Celcius
3.Celcius to Fahrenheit
4. Celcius to fahrenheit using ramp
5. Kelvin to celcius in ramp
6. Voltage to kelvin in ramp
7.Subsystems Implementation

VOLTAGE TO KELVIN



REQUIREMENTS
I/P=Tv
O/p=Tk
Logic:
Tv=(200/12)+273

Figure 12- Voltage to Kelvin

KELVIN TO CELSIUS

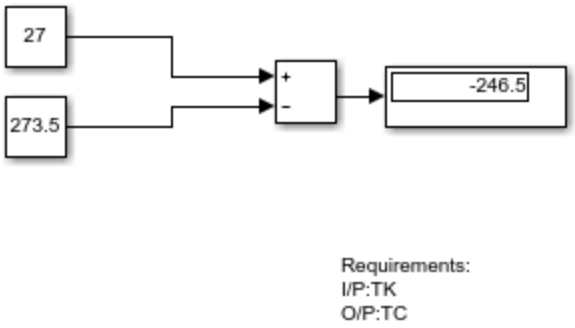


Figure 13- Kelvin to Celcius

CELCIUS TO FAHRENHEIT

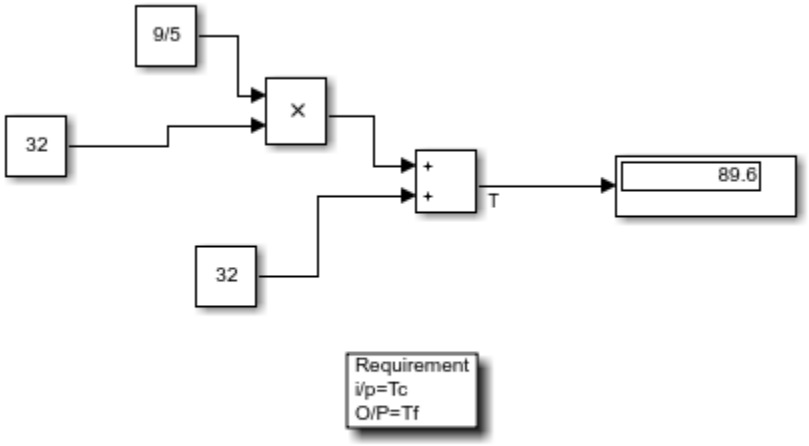


Figure 14- Celcius to Fahrenheit

CELCIUS TO FAHRENHEIT USING RAMP

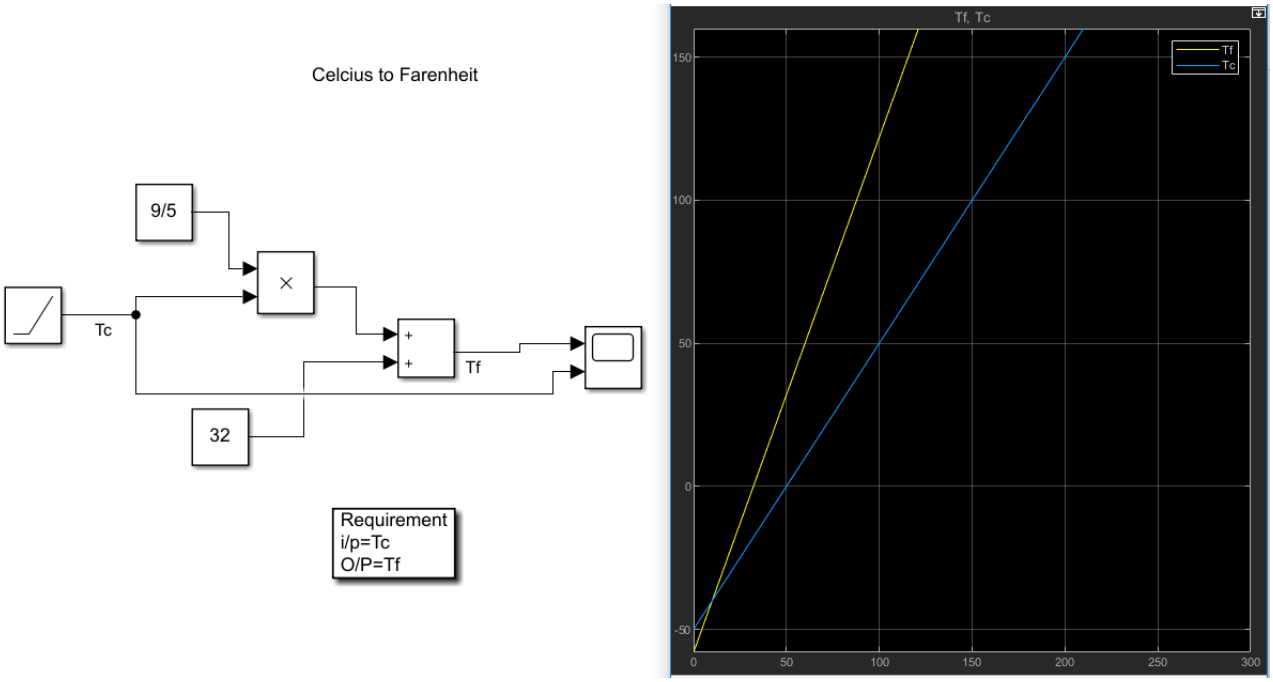


Figure 15- Celcius to Fahrenheit using Ramp

KELVIN TO CELCIUS IN RAMP

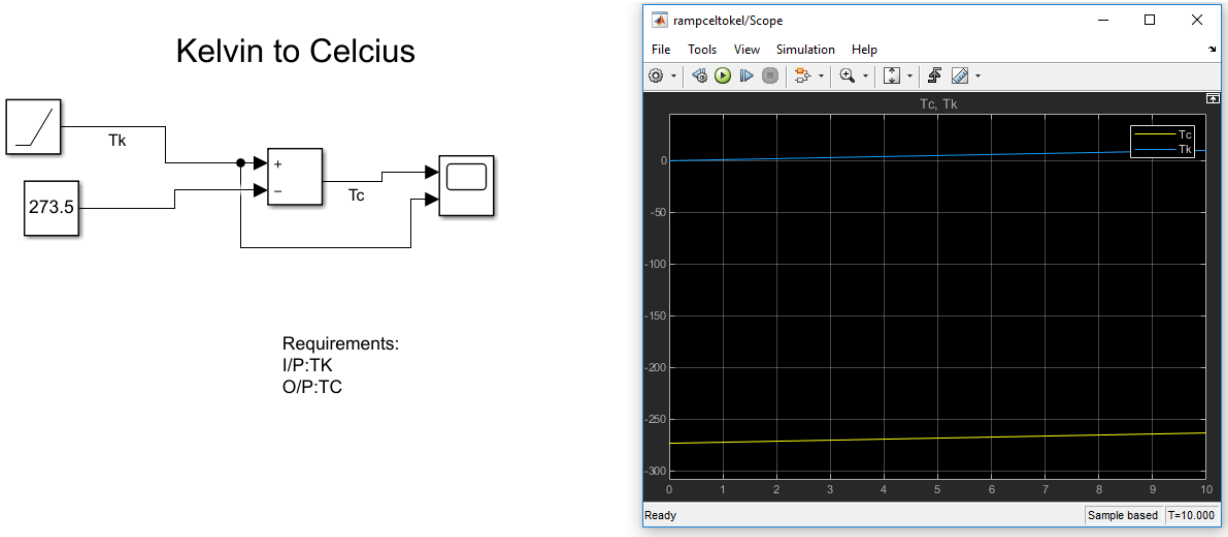


Figure 16- Kelvin to Celcius in Ramp

VOLTAGE TO KELVIN IN RAMP

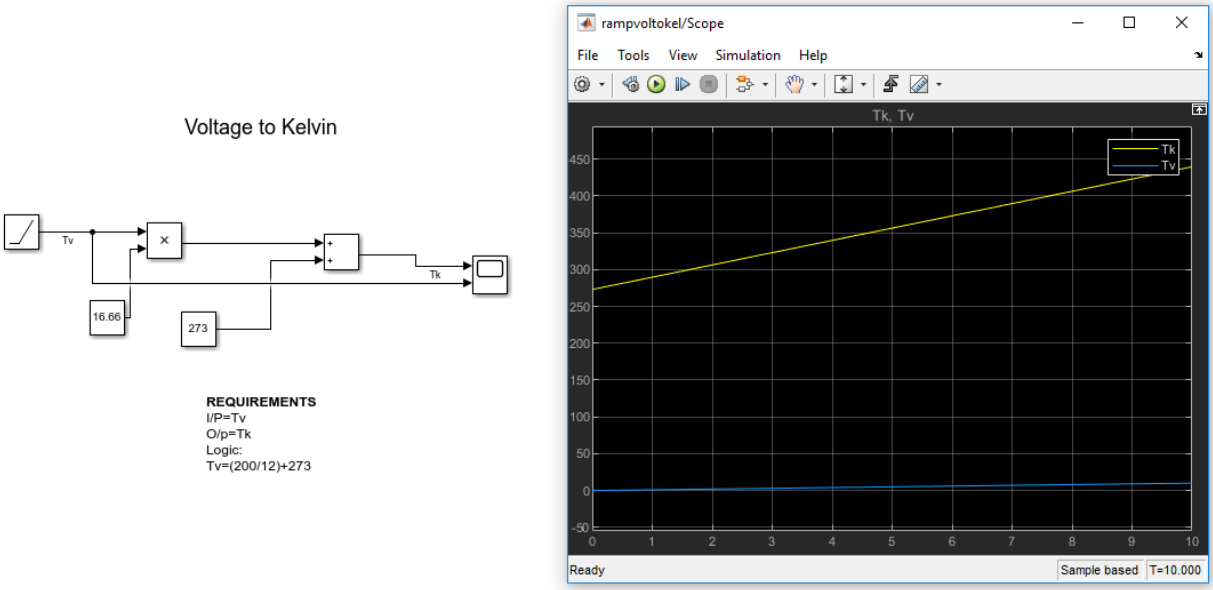


Figure 17-Voltage to Kelvin in Ramp

SUBSYSTEMS IMPLEMENTATION

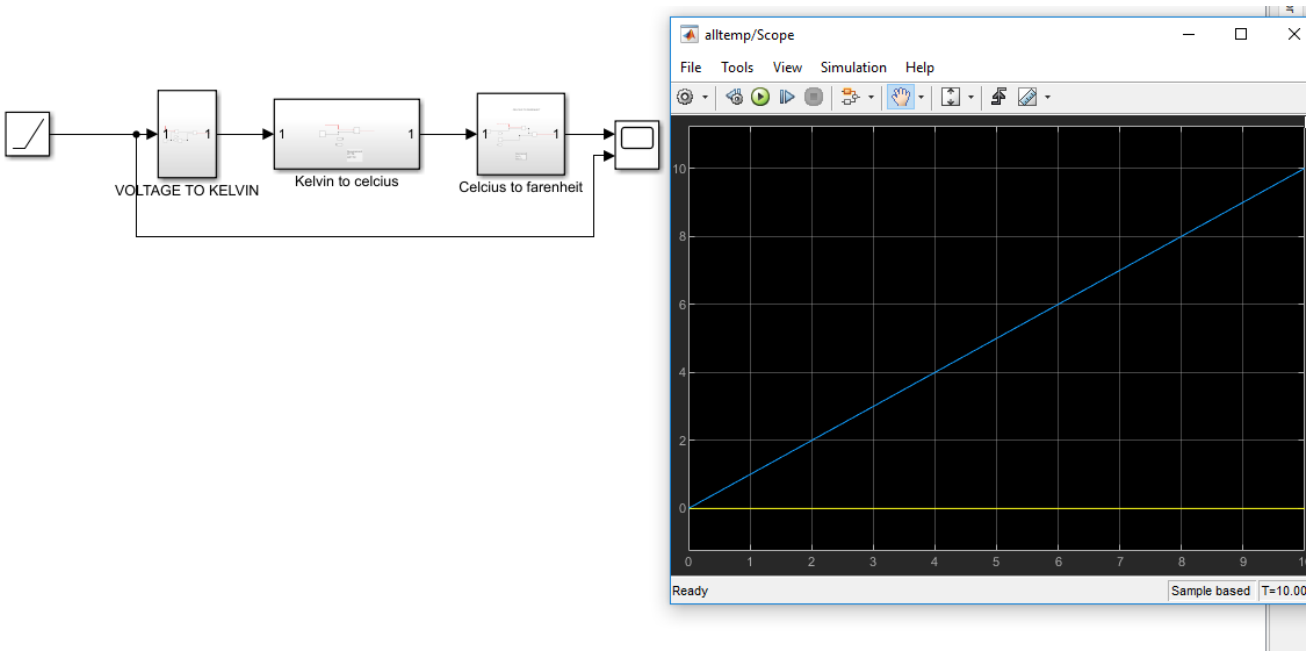


Figure 18- Subsystems Implementation

Designing Activities:

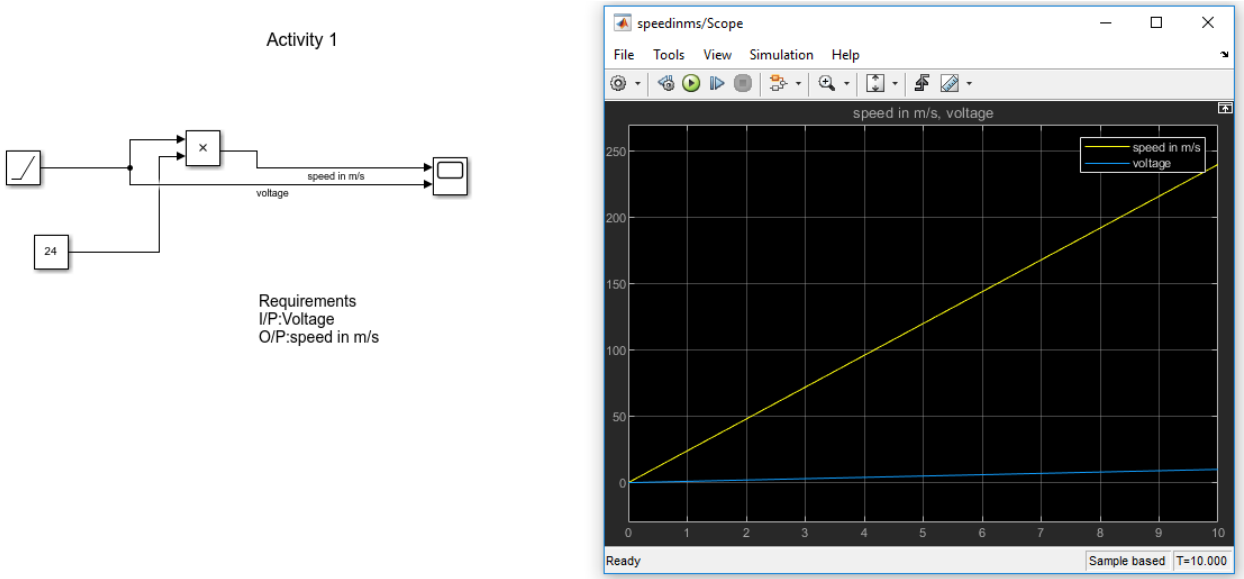


Figure 19- Voltage to Speed in m/s

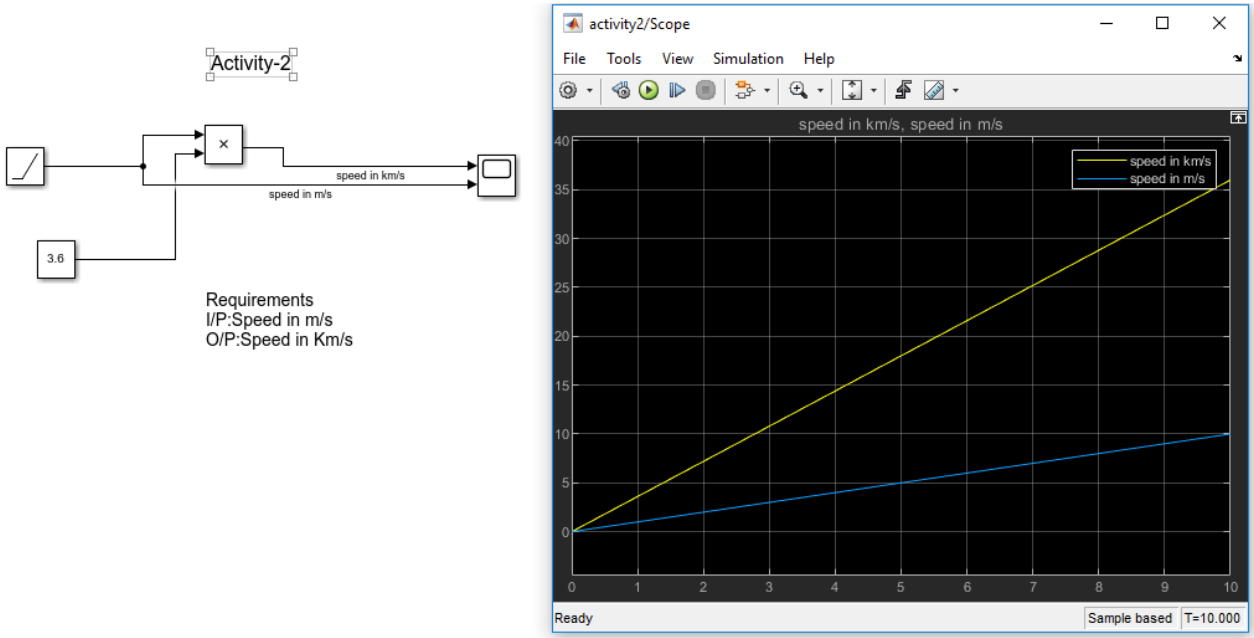


Figure 20- Speed in m/s to Speed in Km/s

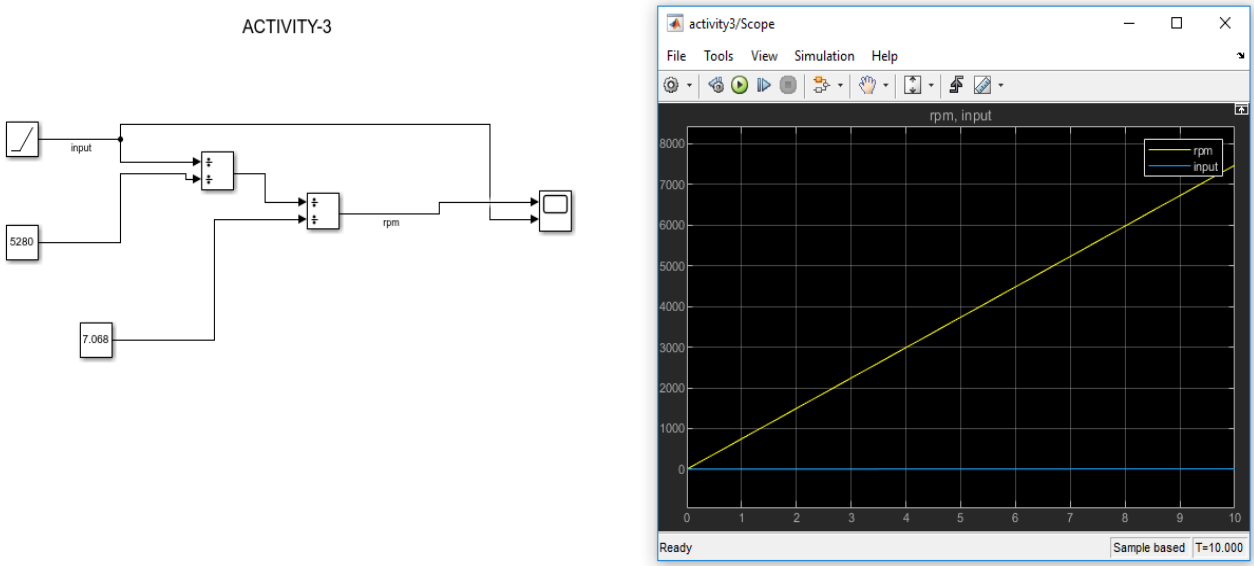


Figure 21- Speed in Km/s to RPM

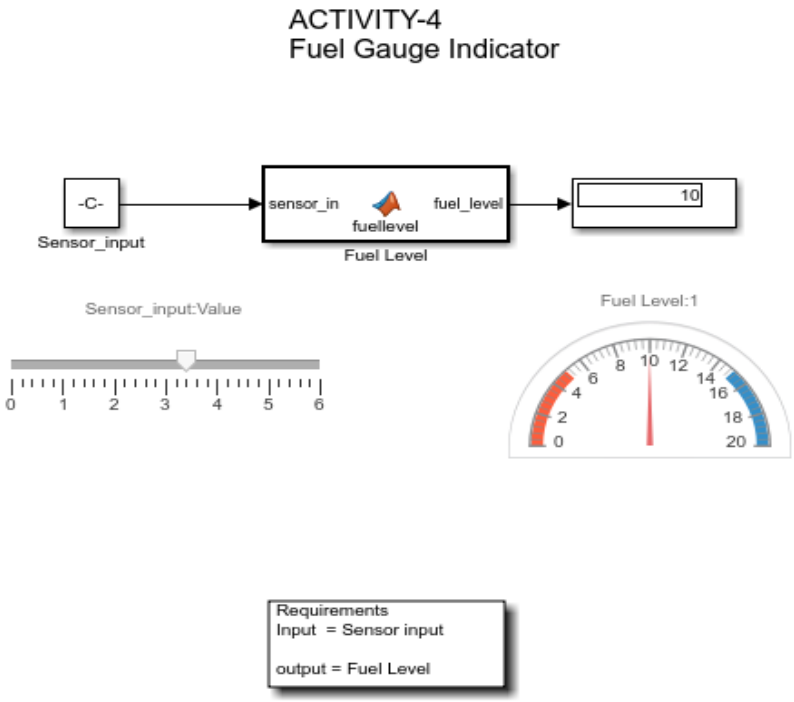


Figure 22- Fuel Gauge Indicator

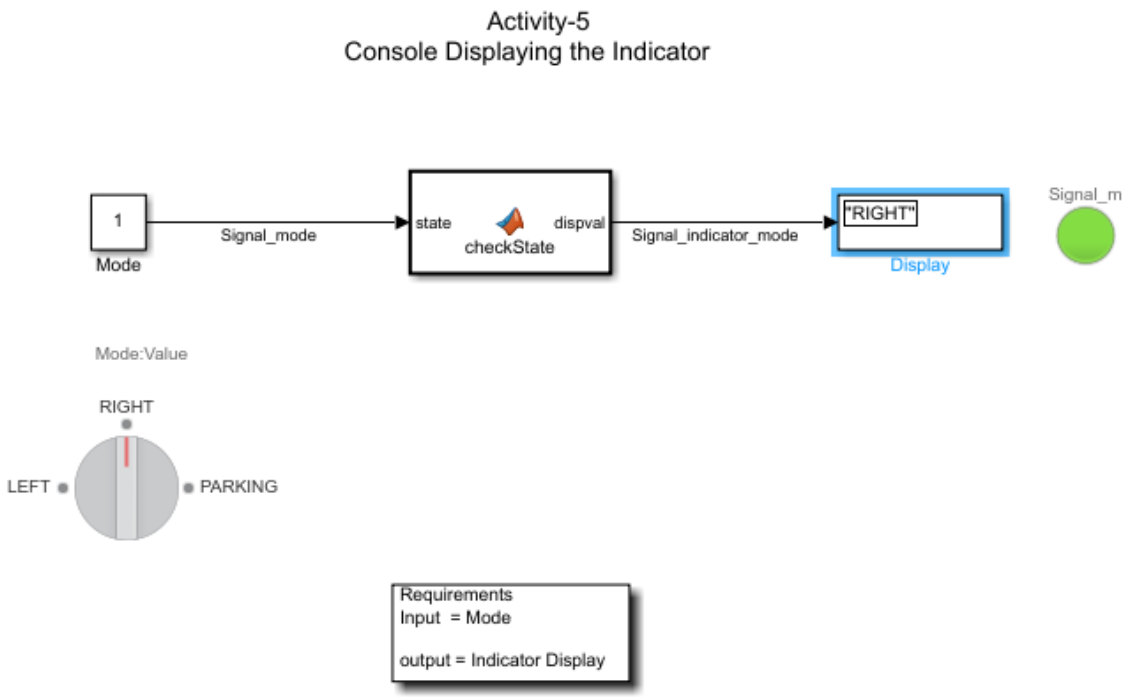


Figure 23- Signal Indicator

ACTIVITY-6
SUBSYSTEMS

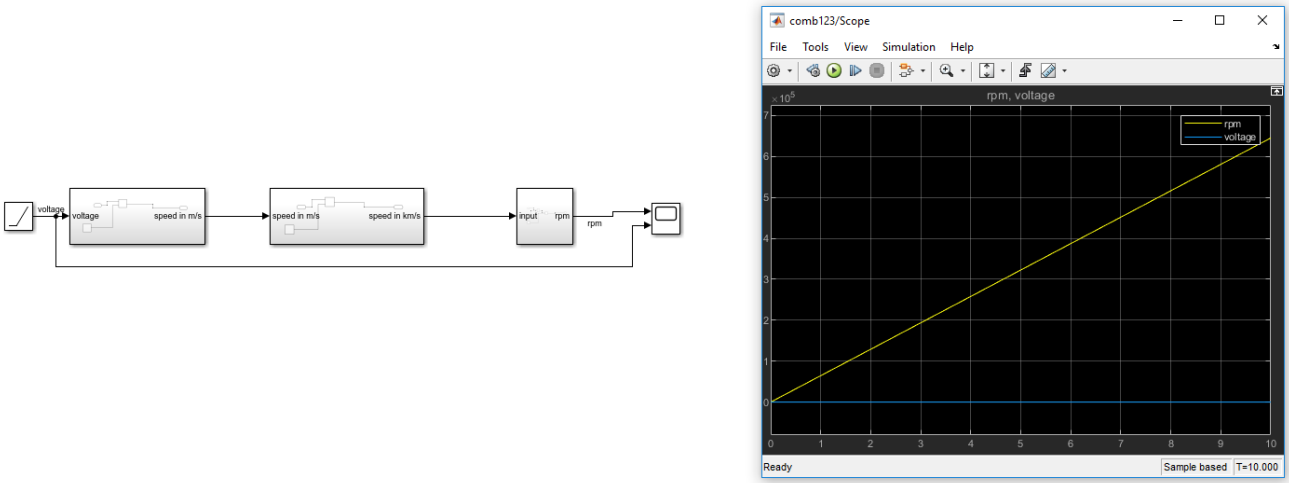


Figure 24- Subsystems

INSTRUMENT CLUSTER:

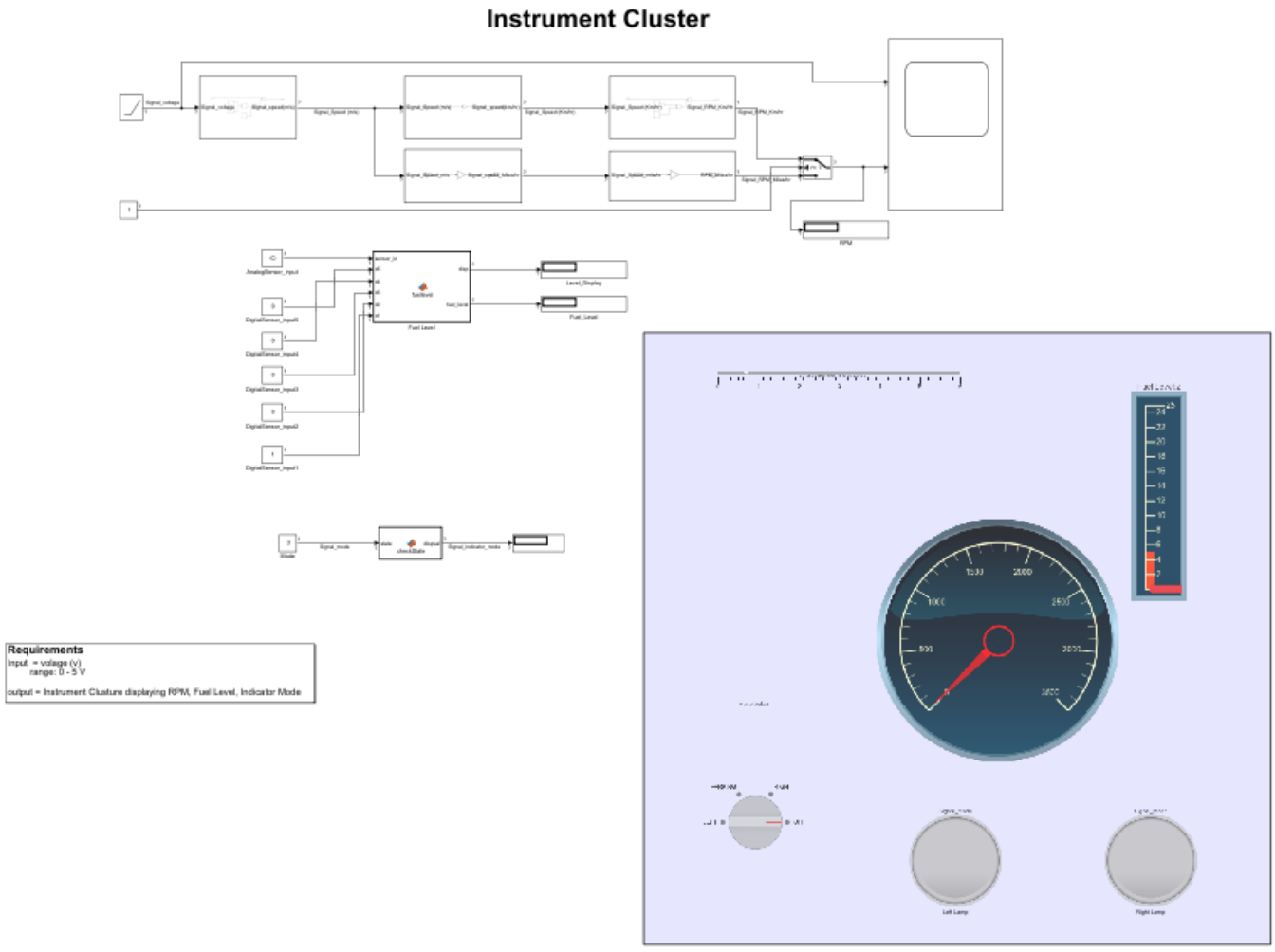


Figure 25- Instrument Cluster-2

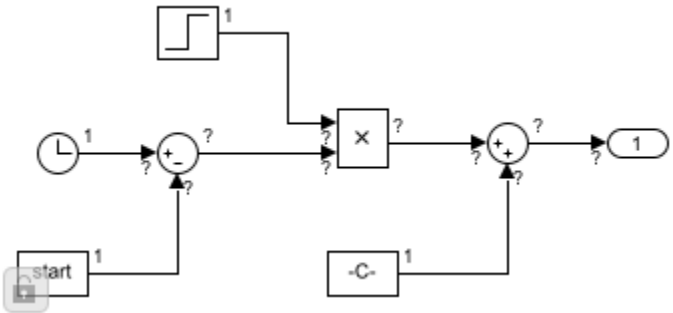


Figure 26- Instrument Cluster-3

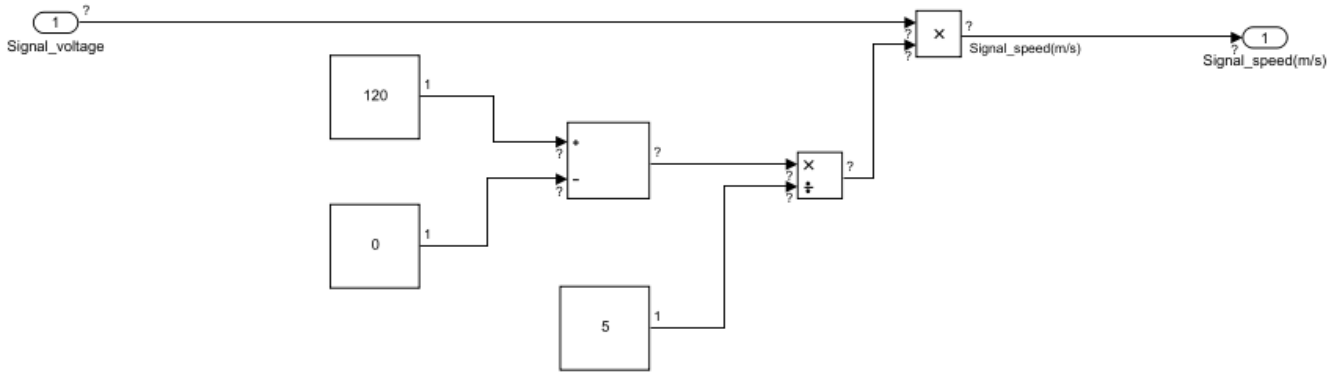


Figure 27- Instrument Cluster-4

Figure 28- Instrument Cluster-5

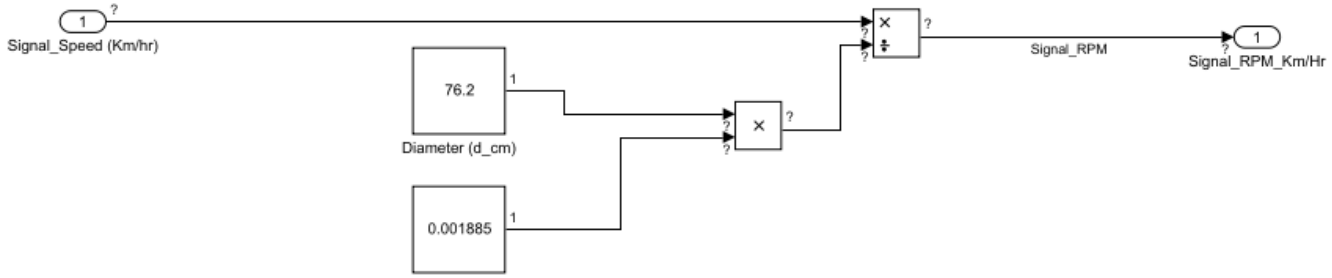


Figure 29- Instrument Cluster-6

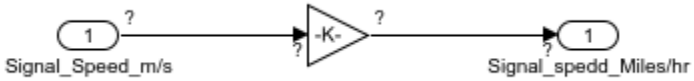


Figure 30- I Instrument Cluster-7

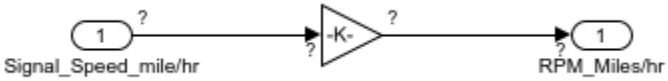


Figure 31- Instrument Cluster-8

LINK: Link

DEBUGGING TECHNIQUES:

Techniques which have used:

1. Adding Break points
2. Displaying Labels
3. Stepping
4. Execution Order
5. Sample Time
6. Logging
7. Highlight source and Destination
8. Floating Scope

DEBUGGING

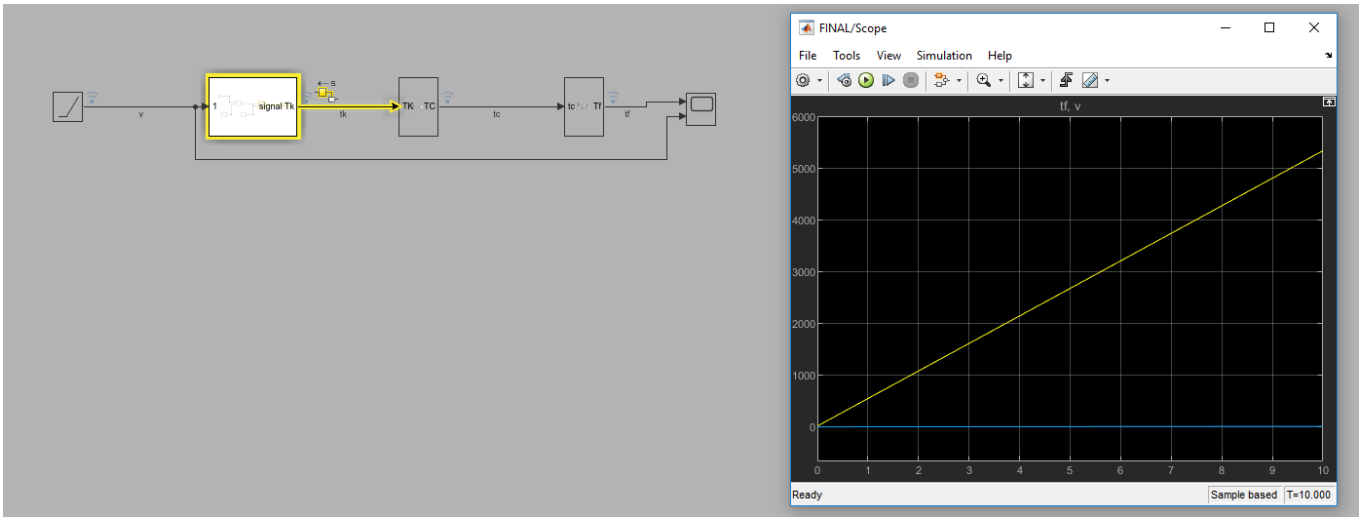


Figure 32-Highlighting

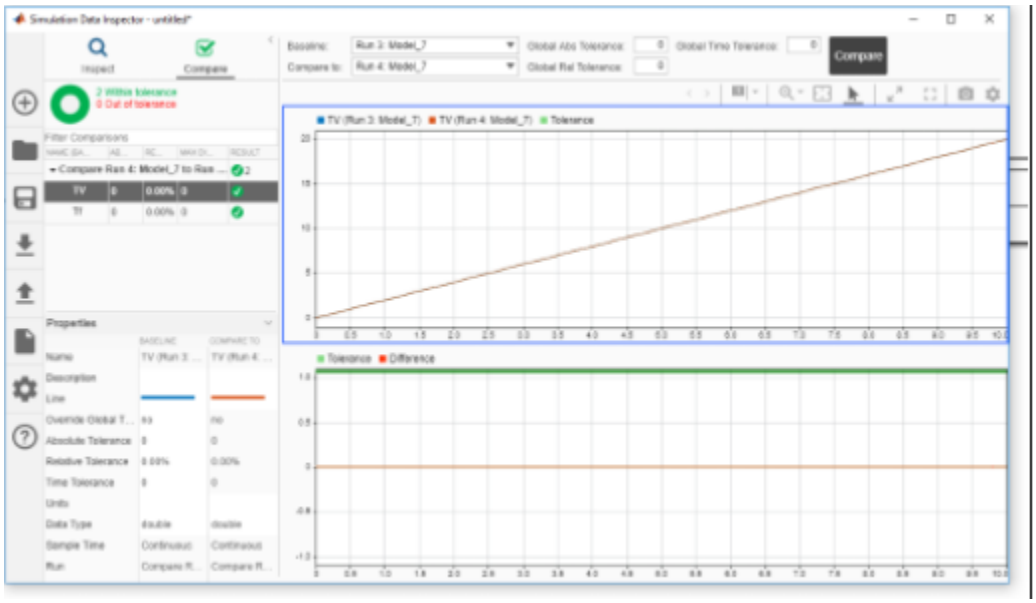


Figure 33- Simulation Inspector

MASKING

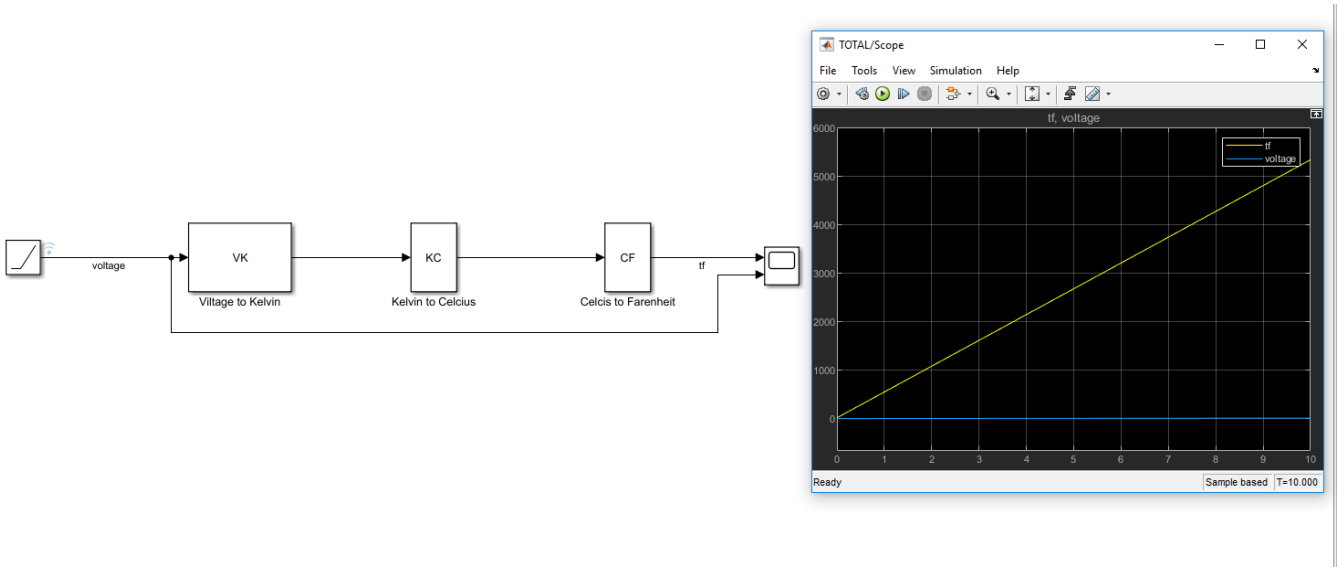


Figure 34- Masking

SUBSYSTEM

1.ATOMIC SUBSYSTEM:

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

- Nonvirtual subsystem – Control when the contents of the subsystem are evaluated as a single unit (atomic execution). Create conditionally executed subsystems that run only when an event occurs on a triggering, function-call, action, or enabling input.
- Virtual subsystem – Subsystem is neither conditionally nor atomically executed. Virtual subsystems do not have checksums. To determine if a subsystem is virtual, use the get param function for the Boolean block parameter IsSubsystemVirtual.
- An atomic subsystem is a Subsystem block with the block parameter **Treat as atomic unit** selected.
- A code reuse subsystem is a Subsystem block with the parameter **Treat as atomic unit** selected and the parameter **Function packaging** set to Reusable function, specifying the function code generation format for the subsystem.

2.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.

3.Enabled Subsystems:

The Enabled Subsystem block is a Subsystem block preconfigured as a starting point for creating a subsystem that executes when a control signal has a positive value.

Use Enabled Subsystem blocks to model:

- Discontinuities
- Optional functionality
- Alternative functionality

4.Variant Subsystems:

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*. You can programmatically switch the active variant of the Variant Subsystem block by changing values of variables in the base workspace, or by manually overriding variant selection using the Variant Subsystem block dialog. The *active variant* is programmatically wired to the Inport and Outport blocks of the Variant Subsystem by Simulink during model compilation.

IMPLEMENTATION:

1. Atomic Subsystem:

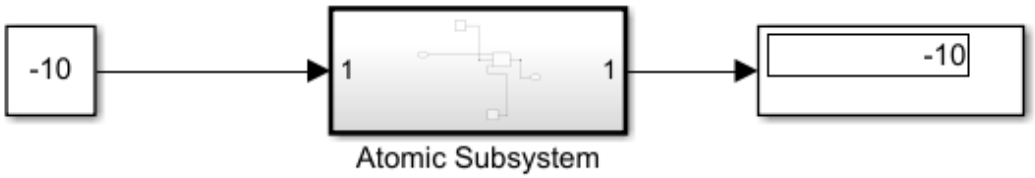
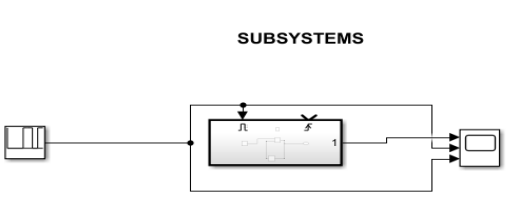


Figure 35- Atomic Subsystem

2. Triggered Subsystem:



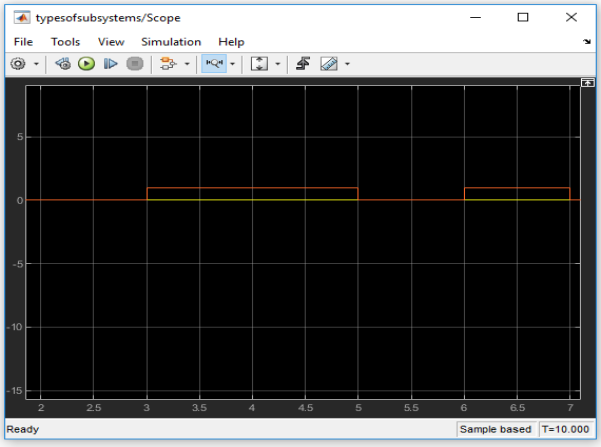


Figure 36- Triggered Subsystem

3. Variant Subsystem:

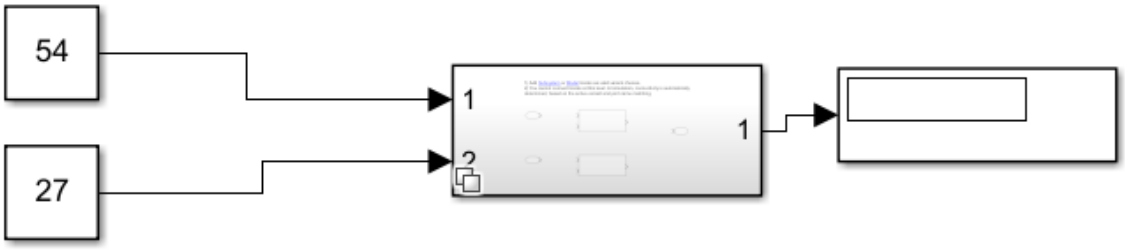


Figure 37- Variant Subsystem

4. Enabled Subsystem:

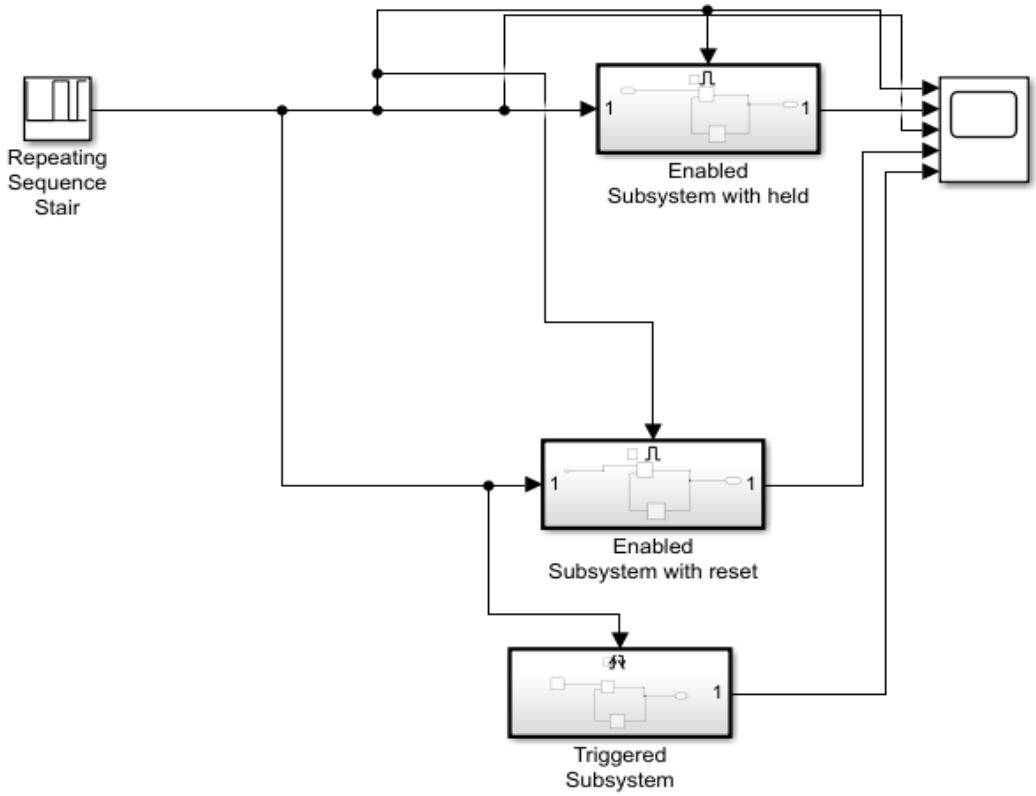


Figure 38- Enabled Subsystem

LINK for subsystems models: link

ACTIVITY 5: STATEFLOW

ACTIVITY 5: STATEFLOW

STATEFLOW MODELS

Requirements:

1. If ambient temperature is greater than user temperature :switch on the A/C.
2. If ambient temperature is lesser than user temperature : switch on the heater.
3. Else just do nothing.

- HEATER AND COOLER SYSTEMS: -Chart

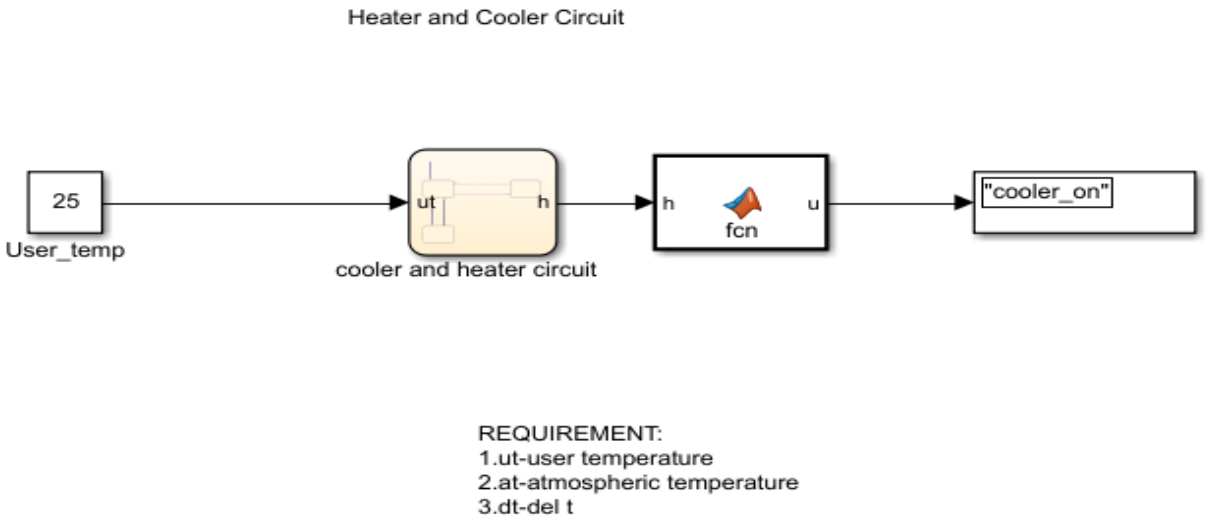


Figure 39- Heater Cooler - Chart

- Heater cooler-flow

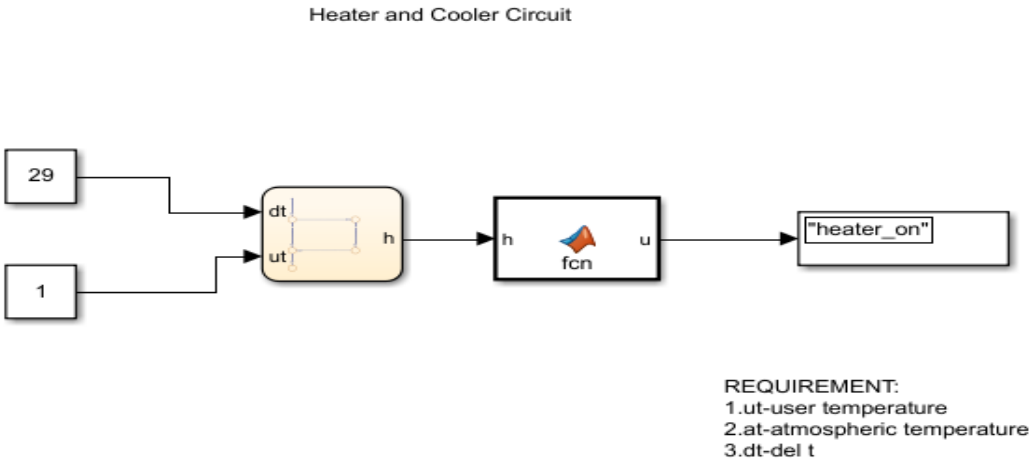


Figure 40- Heater Cooler- Flow

STATEFLOW DEBUGGING TECHNIQUES

1. Animation Speed
2. Break Points

1. ANIMATION:

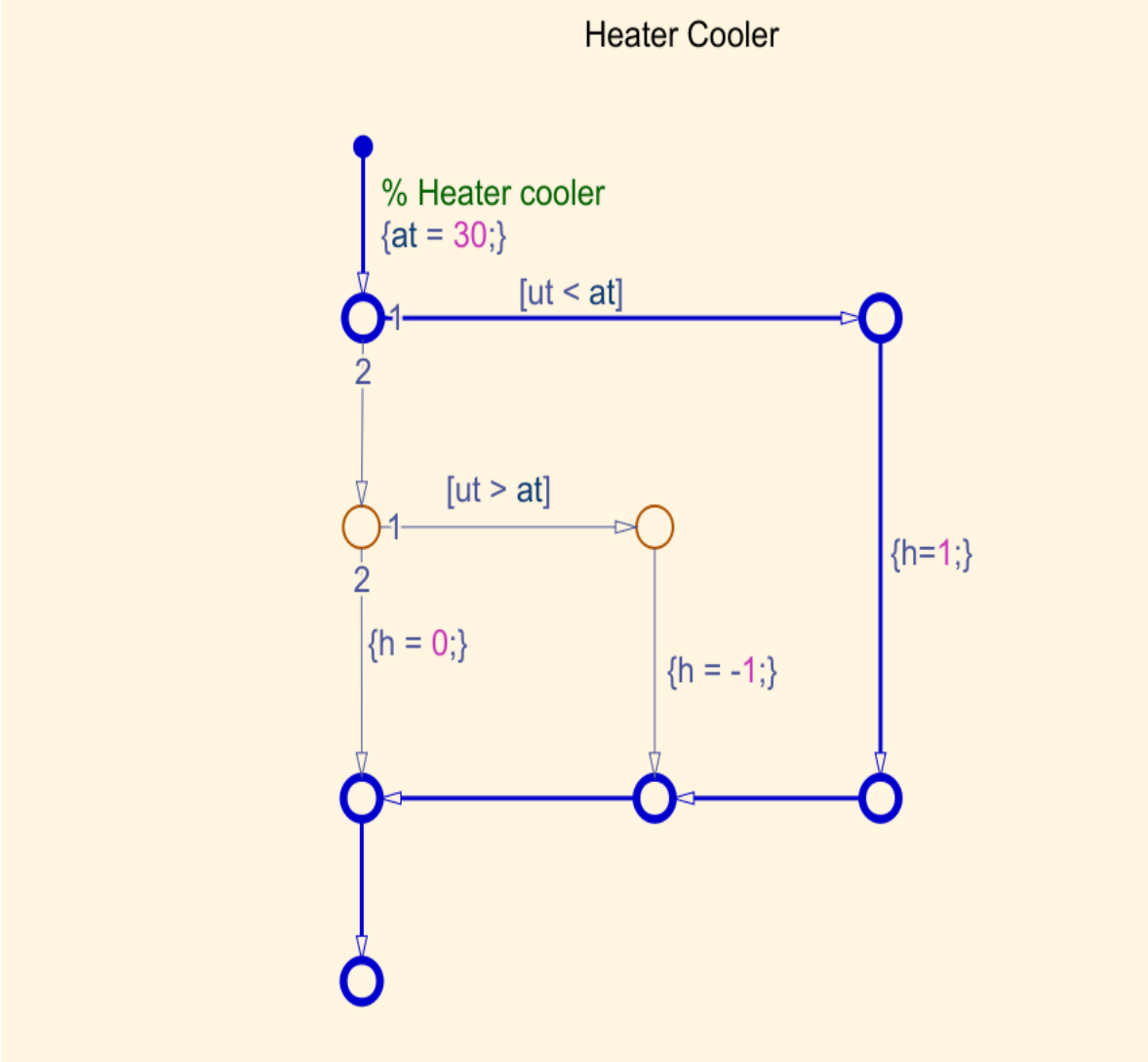


Figure 41- Animation

2. BREAKPOINTS:

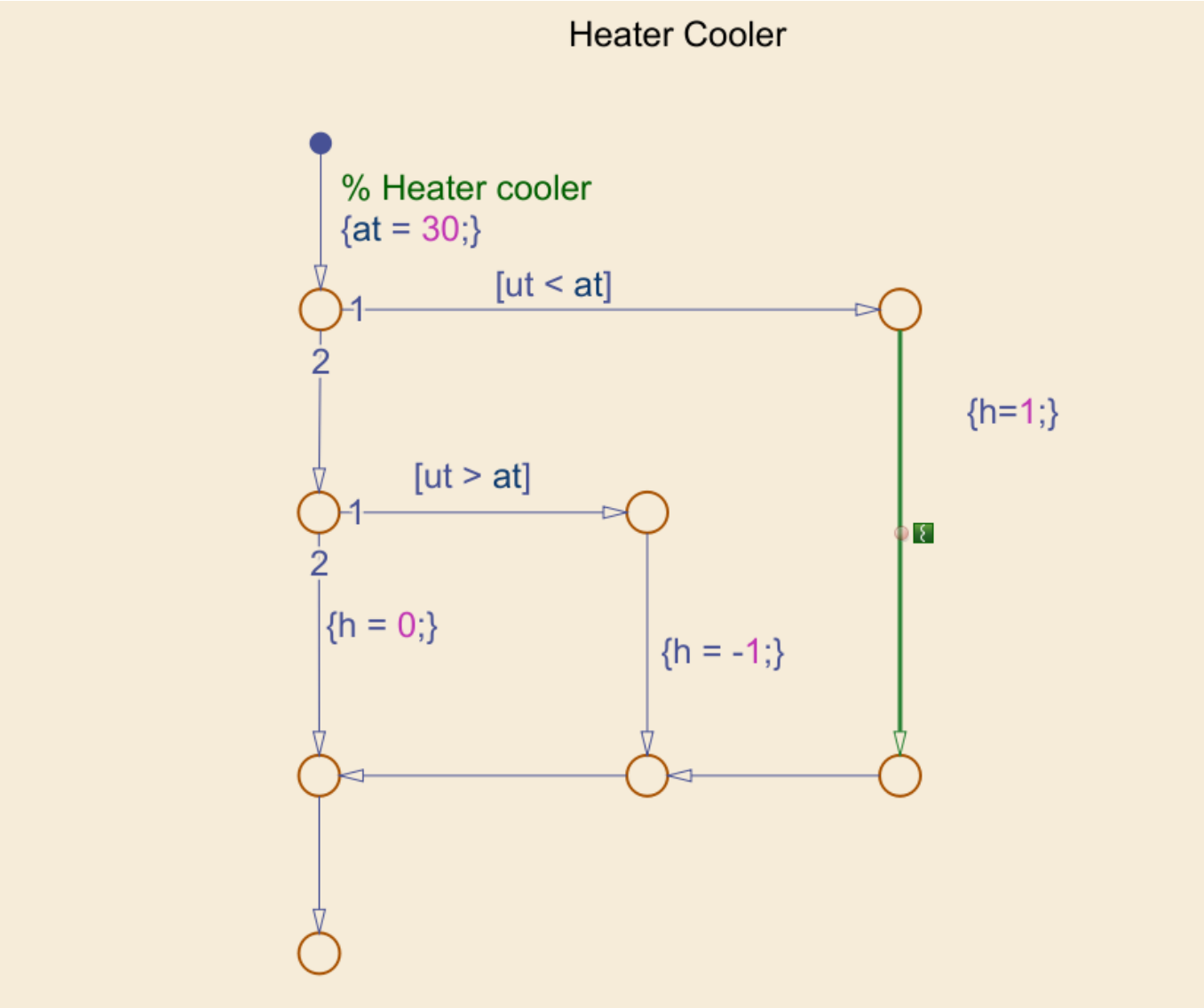


Figure 42- BreakPoints

STATEFLOW ONRAMP

Figure 43- Stateflow Onramp

REFERENCES:

[1]https://www.sae.org/news/2018/10/mbse-is-transforming-aerospace-engineering-systems-integration
[2]http://intercax.com/mbse-for-aerospace/
[3]https://militaryembedded.com/avionics/safety-certification/transitioning-do-178c-arp4754a-uav-using-model-based-design
[4]https://www.yammer.com/lnttsgroup.onmicrosoft.com/#/files/745401434112
[5] https://in.mathworks.com/help/matlab/matlab_prog/regular-expressions.html
[6]https://www.mathworks.com/help/simulink/slref/codereusesubsystem.html#parameter_d119e145990
[7] https://in.mathworks.com/help/simulink/ug/triggered-and-enabled-subsystems.html