GENESIS - Learning Outcome & Mini-project Summary Report

LTTS
GLOBAL
ENGINEERING
ACADEMY

L&T Technology Services

Details

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved	Remarks/Revision Details
	16/04/2021	Vipul Sakunia PS No- 99003723			

Contents

CONTENTS **3**

TABLE OF FIGURES 6

APPLIED SDLC-CALCULATOR [TEAM] **9**

MODULE 9
Topic and Subtopics *9*
OBJECTIVES & REQUIREMENTS 9
REQUIREMENTS 9
INTRODUCTION 9
RESEARCH 10
COST OF THE CALCULATOR 10
FEATURES THAT ARE INTEGRATED IN OUR PROJECT CALCULATOR ARE: 10
DEFINING OUR SYSTEM 10
4WH 10
DETAIL REQUIREMENTS 11
DESIGN 15
TEST PLAN 18
IMPLEMENTATION SUMMARY 19
Git Link *19*
Git Dashboard *19*
Summary *20*
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 28
Summary *28*
Challenges faced and how were they overcome *29*
Future Scope *29*

PYTHON [INDIVIDUAL] **30**

MODULE 30
Topic and Subtopics *30*
OBJECTIVES & REQUIREMENTS 30
HIGH LEVEL REQUIREMENT ANALYSIS 30
LOW LEVEL REQUIREMENT ANALYSIS 30
CODE AND OUTPUT 31
TEST PLAN 34
Git Link *35*
Git Dashboard *35*
Summary *35*
Challenges faced and how were they overcome *35*

EMBEDDED-C [TEAM] **36**

MODULE 36
Topic and Subtopics *36*
OBJECTIVES & REQUIREMENTS 36
DESIGN AND TEST PLAN: 36
IMPLEMENTATION SUMMARY 40
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 40
Summary *40*
Challenges faced and how were they overcome *40*

MBSE [TEAM] 41

OBJECTIVES & REQUIREMENTS 41
DESIGN 42
TEST PLAN 47
IMPLEMENTATION OUTPUT 48
IMPLEMENTATION SUMMARY 50
Summary 50
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 50
Challenges faced and how were they overcome 51

AUTOMOTIVE SYSTEMS AND OVERVIEW-MBD EVOLVED[TEAM] 52

FEATURES 52
OBJECTIVES & REQUIREMENTS 52
HIGH LEVEL REQUIREMENTS: 52
LOW LEVEL REQUIREMENTS: 52
DESIGN 53
FEATURE MODEL 54
TEST PLAN 58
HIL Testing for TPMS and Exterior Lights 58
IMPLEMENTATION 59
MODEL ADVISOR REPORT 59
MCDC REPORT 59
AUTOMATED OUTPUT RESULT 60
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 61
Summary 61
Challenges faced and how were they overcome 61

CANOE AND CAPL SCRIPTING [TEAM] 62

MODULE 62
TOPICS AND SUBTOPICS 62
OBJECTIVES & REQUIREMENTS 62
DESIGN 62
TEST PLAN 63
IMPLEMENTATION SUMMARY 64
SharePoint Link 64
SharePoint Dashboard 64
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 65
Summary 65
Challenges faced and how were they overcome 65

CONTROL SYSTEMS [TEAM & INDIVIDUAL] 66

MODULE 66
TOPICS AND SUBTOPICS 66
OBJECTIVES & REQUIREMENTS 66
REQUIREMENTS 67
DESIGN FOR LONGITUDINAL CONTROL 68
PLANT MODEL WITH P, PI, PD AND PID CONTROLLER 68
Analysis 71
PLANT MODEL WITH LEAD LAG COMPENSATOR 72
TRANSFER FUNCTION: 72

TOOL ANALYSIS 72
Analysis for Lead Compensator *77*
PLANT MODEL WITH FUZZY CONTROLLER 77
LONGITUDINAL CONTROL WITH DISTURBANCE USING PI, PD, PID CONTROL 78
Analysis *82*
SharePoint Link *83*
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 83
Pitch Control for Drone *83*
MATH ANALYSIS 83
PITCH CONTROL OF DRONE WITH CONTROLLERS 83
TOOL ANALYSIS 83
ANALYSIS AND TIME RESPONSE RESULT 87
SharePoint Link *88*
Implementation Summary *88*
SUMMARY 88
INDIVIDUAL CONTRIBUTION & HIGHLIGHTS 88
Challenges faced and how were they overcome *88*

Table of Figures

Figure 1 HLR Structural 15
Figure 2 LLR Sequence Diagram 16
Figure 3 LLR Structural 17
Figure 4 Power function UML 18
Figure 5 Git dashboard 20
Figure 6 Git Dashboard 1 20
Figure 7 Inspector Summary 21
Figure 8 Setup Git 22
Figure 9 Build 22
Figure 10 Built Action 23
Figure 11 Setup Alignment 23
Figure 12 Cpp check 24
Figure 13 Cpp Run Check 24
Figure 14 Issues 25
Figure 15 Unit Testing setup 26
Figure 16 Unit testing Outcome 26
Figure 17 Unit Testing Outcome2 28
Figure 18 Code Python 32
Figure 19 Output Python 1 33
Figure 20 Output Python 2 34
Figure 21 Git dashboard Python 35
Figure 22 Pin Configuration 36
Figure 23 Code Snippet for Hatch Control 37
Figure 24 Microcontroller implementation of Hood control from switch 37
Figure 25 Code snippet for Exterior Lighting 38
Figure 26 MC implementation of Exterior lights using LDR sensor 38
Figure 27 Code snippet for Interior lighting 39
Figure 28 Image of the microcontroller implementation of interior lighting (Pink Light) 39
Figure 29 Code snippet for Wiper control 39
Figure 30 Microcontroller implementation of Wiper Control 39
Figure 31 High Level MBSE Design 42
Figure 32 MBSE Input 43
Figure 33 MBSE Output 43
Figure 34 Dashboard 44
Figure 35 MBSE Plant 44
Figure 36 Engine Status/Key Ignition 45
Figure 37 Power Outlet 45
Figure 38 Gear Control 46
Figure 39 Gear Control State flow 46
Figure 40 Seat Belt reminder MBSE 47
Figure 41 SBR Sate flow Implementation MBSE 47
Figure 42 Output for Manual Implementation 1 49
Figure 43 Output for Manual Implementation 2 50

Figure 44 High level Design 53
Figure 45 Low Level design 53
Figure 46 Evolved Plant 54
Figure 47 Seat Belt reminder model 54
Figure 48 Gear Control for Input to SBR 55
Figure 49 Gear Implementation State Flow 55
Figure 50 SBR low level model 56
Figure 51 State flow for SBR 56
Figure 52 Algorithm SBR 57
Figure 53 SWOT for SBR 58
Figure 54 HIL testing for TPMS 58
Figure 55 HIL testing for Exterior Lights 59
Figure 56 Model advisor for whole model 59
Figure 57 MCDC report 60
Figure 58 Automated Output for SBR and Gear 60
Figure 59 Dashboard Canoe 62
Figure 60 Speedometer 63
Figure 61 Mode switch, Break and Throttle control 63
Figure 62 Trace Canoe 64
Figure 63 SharePoint Dashboard 65
Figure 64 Longitudinal Control 66
Figure 65 Plant Model PID 68
Figure 66 PID scope output 69
Figure 67 Bode Plot P 69
Figure 68 Bode Plot PI 70
Figure 69 Bode Plot PD 70
Figure 70 Bode Plot PID 71
Figure 71 Main TF response 74
Figure 72 Pole Zero Map of main TF 74
Figure 73 Response with Lead-Lag Controller 75
Figure 74 Lead Lag Pole Zero Map 76
Figure 75 Bode Plots 76
Figure 76 Fuzzy Logic Implementation 77
Figure 77 Fuzzy Bode Plot 78
Figure 78 Fuzzy Output 78
Figure 79 PID with Disturbance 79
Figure 80 Disturbance with PI Output 79
Figure 81 Disturbance with PD Output 80
Figure 82 Disturbance with PID Output 80
Figure 83 Bode Plot PI 81
Figure 84 Bode Plot PD 81
Figure 85 Bode Plot PID 82
Figure 86 Step Response Drone 86
Figure 87 Response with Controllers 86
Figure 88 Bode Plots for Drone PID 87

Figure 89 Pole Zero Map 88

Applied SDLC-CALCULATOR [Team]

Module

Modules linked to the calculator project is Applied SDLC and C Programming.

Topic and Subtopics

The topics which I mainly focused on are UML diagrams, different software life cycles, usage of software tools (VScode), creation and installation of make file, learning about how to use GitHub and how to do continuous integration of the team code, SWOT Analysis and 4 WH questions to justify the objective.

- Software Life Cycles:

  Mainly focused on Agile model, V model and their advantages, disadvantages and how it affects the cycle of development.

  Learnt about release cycles & development cycles. In development life cycles, we have different stages named as Pre-alpha, Alpha, Beta, Release Candidate.
- GITHUB and GIT Desktop:
  1. How to make the repositories.
  2. How to push and pull the code in and out of the repositories using git Desktop and through git commands.
  3. Learnt about how to generate Make file in VScode.
  4. Continuous Integration in GITHUB and learnt about GIT inspector
  5. Learnt about how to raise issue and solve the issues and learnt about the difference between branch and main and raised a request to merge the branch in to main.
- UML diagrams:
  1. UML- "unified modeling language" learnt there are 2 categories namely structural, behavioral and chose different models in structural, behavioral model which fits according to the objective
  2. Took state structural model and class behavioral model for the low-level requirement and Deployment behavioral model for the High-level requirement.

- Testing:
  1. Testing types functional testing and nonfunctional testing and learnt that Nonfunctional testing has no impact on the features.
  2. In the functional testing, we have different types namely Boundary value test, Requirement based test, Scenario based test.

Objectives & Requirements

Requirements

The calculator build by us includes basic arithmetic functions, trigonometric functions and exponential functions. It also consists of some other functions including power, log, antilog, permutation and combination functions.

Introduction

We have attempted to build the software development cycle of a semi-engineering calculator. It includes the numerous high-level and low-level requirements of the various features involved in semi-engineering such as log,

antilog, exponents roots and polynomial functions. It then explores about the various test cases that should be taken in consideration such that all requirements are taken care of. In the end, we build the codes that follow all the requirements and design efficiently as per them.

Research

There are different types of calculators with respect to the cost and its features. It is plotted in the graph shown below which shows the range of calculators for highest cost and lowest feature and vice versa. The range we have chosen is for a medium range cost and feature. It is similar to a programmable scientific calculator. It will be more efficient for the multipurpose use.

Cost of the Calculator

- Our Calculator will sum around 800 Rupees, having all necessary features required for all students
- Cost and features mainly focus on most of the important requirements we researched for a calculator.

Features that are integrated in our project calculator are:

- Will perform basic Addition, Subtraction, division, Multiplication with validation
- Also, it can perform long division approach with remainder and quotient as a result
- Addition of Hexadecimal number
- Calculation of GCD
- Basic Trigonometry operation like Sine, Cos, Tan
- performing the power function i.e. calculating the power of the number
- Integrated with operation like logarithmic and Antilog calculation
- Factorial of numbers and Exponential
- Mean of the numbers
- Conversion of Units

Defining Our System

Our system is more or less semi advanced which basically able to do desirable calculations of daily basis from bottom to top level i.e. it includes every basic operation which eats lots of time in scheduled timings.

4WH

Who:

- School students, college personnel of Science and Engineering, Professionals at industrial level

What:

- Calculate problems in Science, engineering & mathematics. In some aspects of astronomy, physics and chemistry.

When:

- In situations that require quick access to certain mathematical functions (trigonometry, logarithmic, etc.)

Where:

- Widely used in both educational and professional settings (Institutions, laboratories).

How:

- Our calculator is a simple tool that any user can efficiently use at the time of most basic as well as complex calculations.

Detail requirements

High Level Requirements:

ID	Requirements	Description	Status
HL001	CALCULATING NUMBERS	BASIC ARITHMETIC CALCULATION- ADDITION- SUBTRACTION- DIVISION- MULTIPLICATION	IMPLEMENTED
HL002	LOG ANTILOG & POWER OPERATIONS	SOLVING POWER FUNCTIONS AND LOG/ANTILOG OPERATIONS	IMPLEMENTED
HL003	TRIGONOMETRIC FUNCTIONS	SINE- COSINE- TAN	IMPLEMENTED
HL004	PERMUTATION & COMBINATION	PERMUTATION & COMBINATION FORMULAE OUTPUTS	IMPLEMENTED
HL005	CONVERSIONS OF UNITS	UNIT COVERSIONS FOR TEMPERATURE, DEGREE, etc.	IMPLEMENTED
HL005	FACTORIAL	FACTORIAL OF THE POSITIVE NUMBERS	IMPLEMENTED
HL007	LCM & GCD	LCM and GCD OF THE POSSIBLE REAL NUMBERS	IMPLEMENTED
HL008	ROOTS	SQUARE AND CUBE ROOTS OF THE POSITIVE NUMBERS	IMPLEMENTED

Low Level Requirements:

Arithmetic Operations:

ID	Requirements	Description	Status
LL001	Addition	Taking two numbers as input and adding them	Implemented
LL002	Subtraction	Taking two numbers as input and subtracting them	Implemented
LL003	Multiply	Taking two numbers as	Implemented

		input and multiplying them	
LL004	Divide	Taking two numbers as input and dividing them	Implemented

Log and Antilog operations

ID	Requirements	Description	Status
LL003	Logarithm	Taking one number as input with base as 10 and finding its log output	Implemented
LL004	Anti-logarithm	Taking one number as input with base as 10 and finding its antilog output	Implemented

Trigonometric Operations

ID	Requirements	Description	Status
LL005	Sine Functions	Finding the sine of a number	Implemented
LL006	Cos Functions	Finding the cos of a number	Implemented
LL007	Tan Functions	Finding the tan of a number	Implemented

Permutation and Combination

ID	Requirements	Description	Status
PER_1	n<R	shows error when N<R	Implemented
PER_2	Formulae calculation Permutation	finds permutation using the formulae	Implemented
COM_1	n<R	shows error when N<R	Implemented
COM_2	Formulae calculation combination	finds combination using the formulae	Implemented

Conversion of Units

ID	Requirements	Description	Status
COU_2	Temperature conversion	converts user input from Celsius to Fahrenheit	Implemented
COU_3	Degree to radian conversion	converts one user input from degree to rad at a time	Implemented

Factorial and Power

ID	Requirements	Description	Status
Fac_1	Factorial	Finding factorial of the given number	Implemented
pre_1	Library	Math library file will be required in coding section.	Implemented
pre_2	power function	Requirement of power function while executing this section of program.	Implemented
pre_3	Data type	Must contain best suited data types else show	Implemented
pre_3	Boundary limit	Limit exceeds.	Implemented

LCM and GCD Functions

ID	Requirements	Description	Status
LL0011	LCM	Finding the least common multiple of the possible real numbers	Implemented
LL0012	GCD	Finding the greatest common devisor of the possible real numbers	Implemented

Precedence

ID	Requirements	Description	Status
BOD_1	Bracket Type	For more than 1 operation, brackets ((),[],{}) should be identified and executed first.	Implemented
BOD_2	Bracket order 1	The bracket order (), {}, [] is to be followed.	Implemented
BOD_3	Bracket order 2	For more than 1 operation, powers and roots should be identified and executed after brackets.	Implemented
BOD_4	Bracket order 3	For more than 1 operation, Division should be identified and executed after exponents and roots.	Implemented
BOD_5	Bracket order 4	For more than 1	Implemented

		operation, Multiplication should be identified and executed after Division.	
BOD_6	Bracket order 5	For more than 1 operation, Multiplication should be identified and executed after Division.	Implemented
BOD_7	Bracket order 6	For more than 1 operation, Addition should be identified and executed after Multiplication.	Implemented
BOD_8	Bracket order 7	For more than 1 operation, subtraction should be identified and executed after Addition.	Implemented

Store and Recall Functions

ID	Requirements	Description	Status
BOD_1	Stored value	- Capable of storing values for 50 days	Implemented
BOD_2	MC function	- Pressing 'MC' and 0 clears the memory	Implemented
BOD_3	Memory storage	- Stores up to 5 memories.	Implemented
BOD_4	Memory style	- Show all memories in a first in last out style after every click	Implemented

Precision

ID	Requirements	Description	Status
pre_1	Multiplication power	When multiplication of 10 digit it should show in power of ten	Implemented
pre_2	divided by 0	When value is divided by zero then zero then it should through error.	Implemented
pre_3	Boundary Precision	user puts value of more than 10 then it must not except more than that and print limit exceeds as an output.	Implemented
COM_3	Formulae calculation combination	finds combination using the formulae.	Implemented

COM_1	n<R	shows error when N<R	Implemented

Design

High Level Designs:

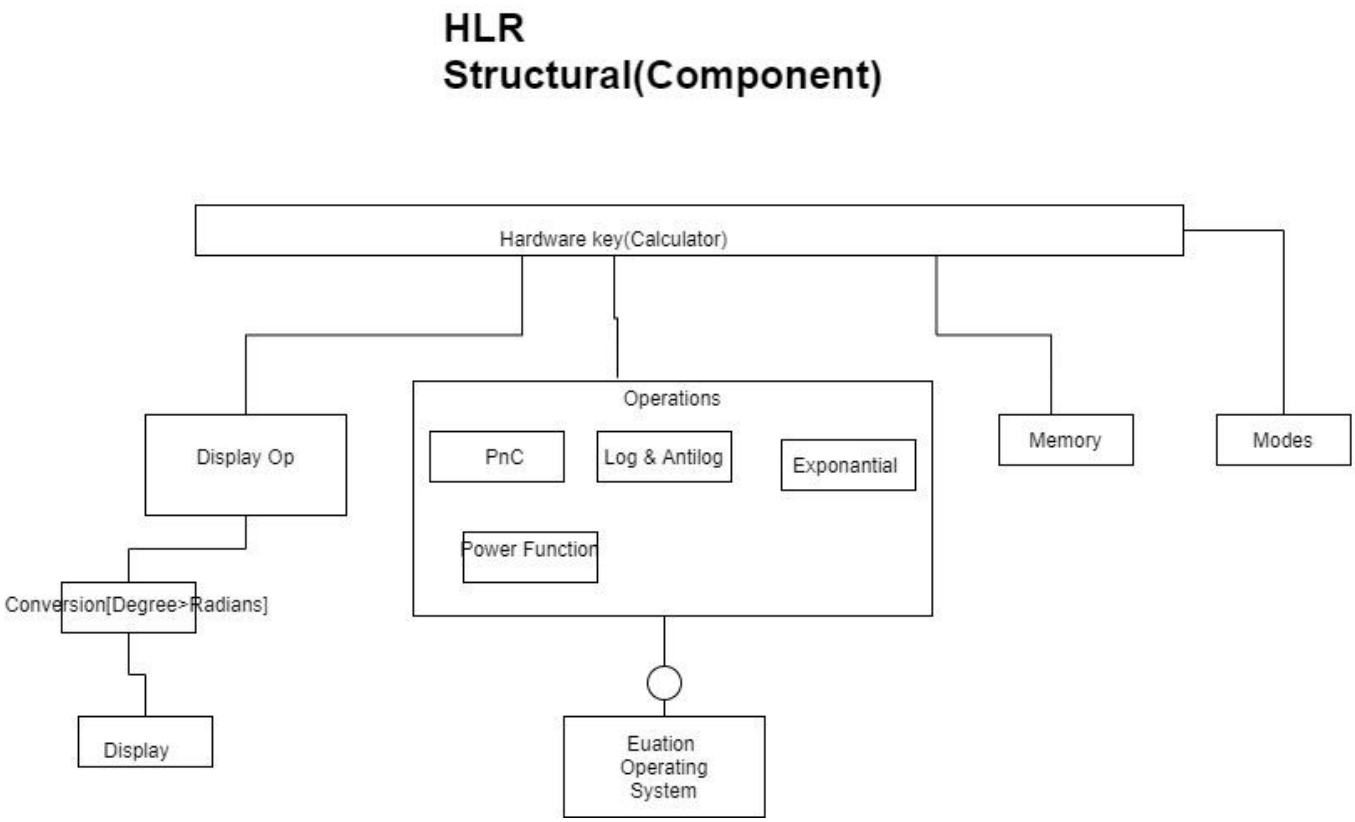


Figure 1 HLR Structural

Low Level Designs:

Sequence Diagram for LLR

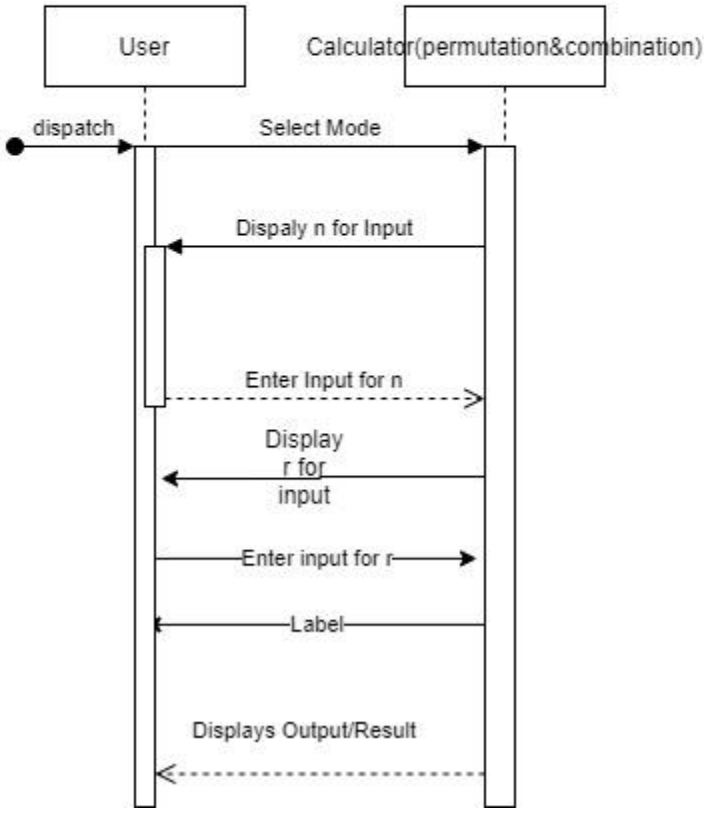


Figure 2 LLR Sequence Diagram

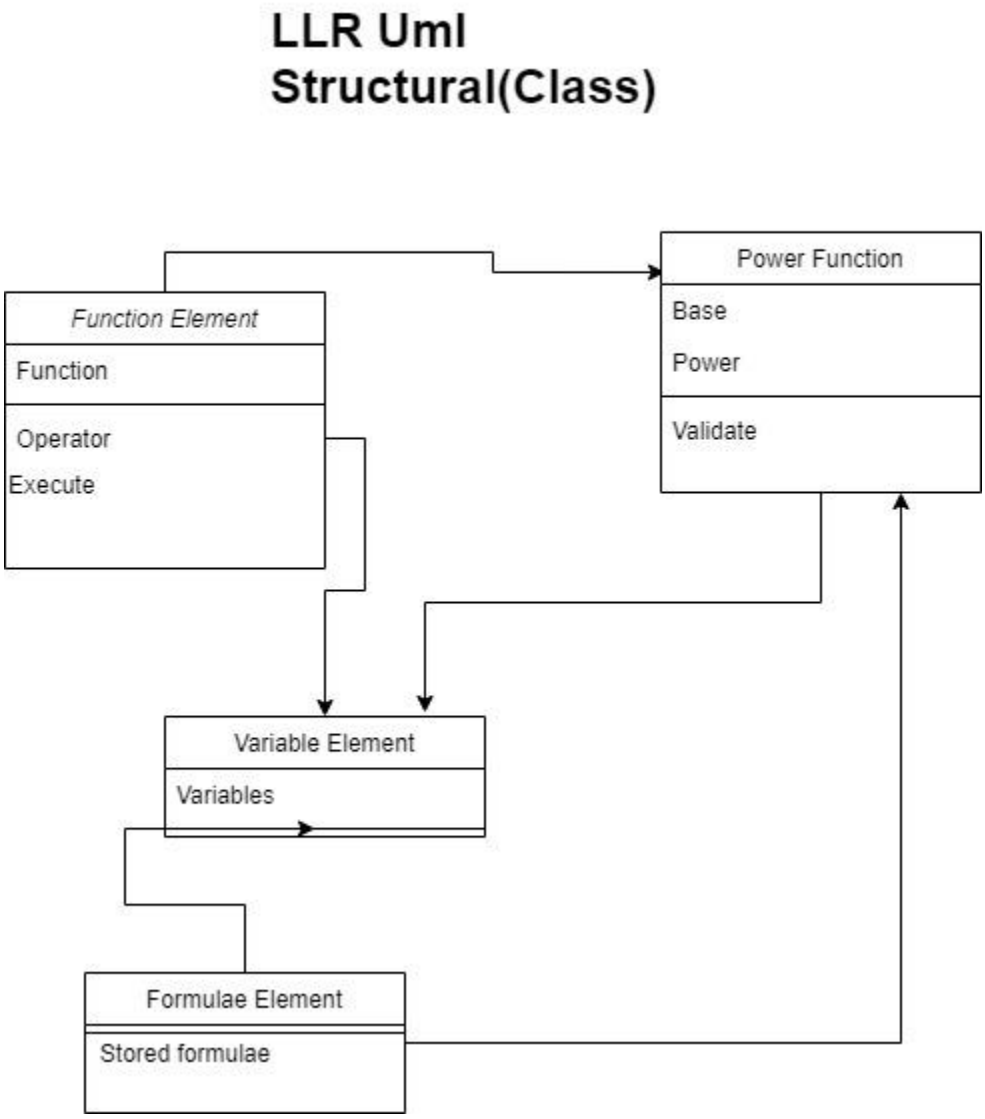


Figure 3 LLR Structural

Behavioral – Object Diagram

Power Function

Figure 4 Power function UML

Test Plan

High-Level and Low-Level test plan:

ID	Description	Input	Expected Output	Actual Output	Test Type
HLT_1	When n greater than or equal to r for nPr	n=3 r=2	3	3	Scenario
HLT_2	Power fxn F(x)= x^r Where r is greater than equals to 0	x=2 r=3	8	8	Requirement
HLT_3	Conversion of temperature unit	- Stores up to 5 memories.	f	f	Requirement
HLT_4	Conversion of degree to radian	- Show all memories in a first in last out style after every click	degree	radian	radian
LLT_1	When n<r for nPr	n=2 r=3	Error	Error	Scenario
LLT_2	Finding the square of given number i.e. find power of number will be 2	100^2	10000	10000	logic based
LLT_3	Finding the power of given number i.e. find power of number will be any number	2^5	32	32	logic based
LLT_4	Finding the power of big number i.e. find power of	3^20	3.84*10^9	3.84*10^9	Boundary

	number which results in digits more than 20				based
LLT_5	find permutation using the formula	3,2	6	6	logic based
LLT_6	finding permutations of large numbers	180,179	Error: large result	Error: large result	Boundary based
LLT_7	Permutation of numbers when n<r	10,11	Error	Error	Requirement based

Implementation Summary

The implementation part of the project consists of the software implementation of the product, Calculator, as documented in the requirements and design section of the life cycle. It consists of various C files in the and header files that link the multiple files. It also consists of the Make file which is used to build run and clean all the multiple files and check for the test cases that are formulated according to the requirement specified.

The source files contain the implementation of the various high-level requirements specified including the following functions:

- Arithmetic Operations
- Trigonometric Operations
- Roots
- Log and Antilog functions
- Permutation and combination
- Conversion of units
- Factorial and power functions

Git Link

https://github.com/99003721/AppliedSDLC_Calculator_N1

Git Dashboard

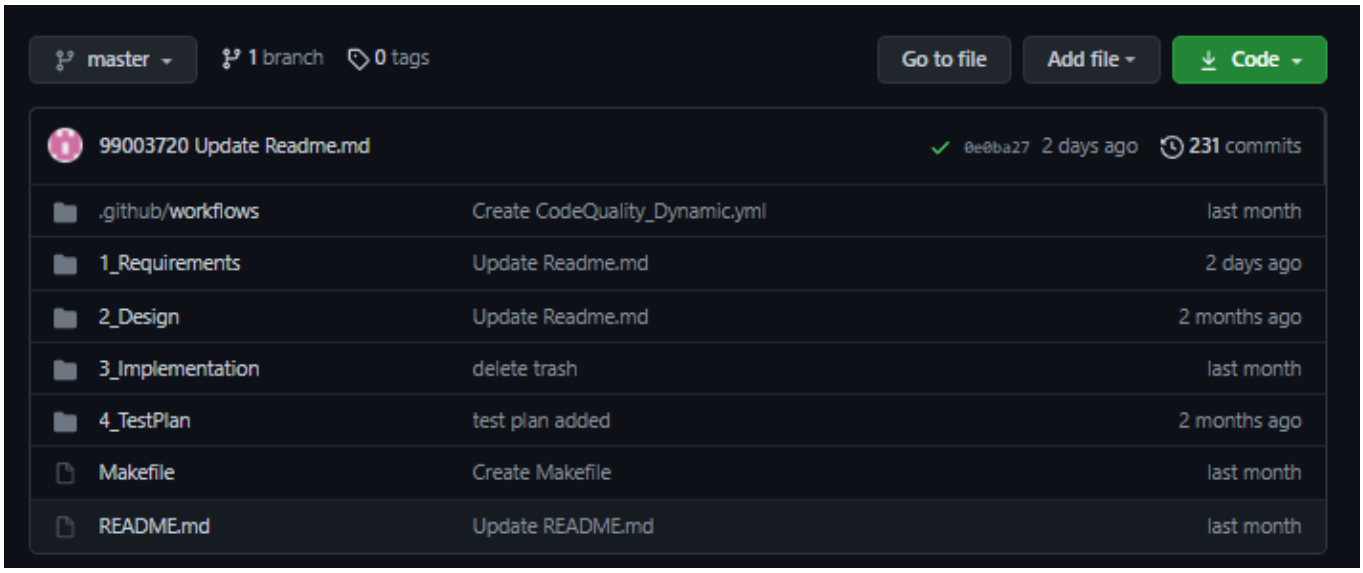


Figure 5 Git dashboard

Build	Code Quality	Unity	[Git Inspector](using github.io option)
C/C++ CI - Build Status passing	Cppcheck passing CodeQuality Dynamic Code Analysis Valgrind passing Codacy Security Scan passing	Unit Testing - Unity passing	Contribution Check - Git Inspector passing

Contributors List and Summary

PS No.	Name	Features	Issuess Raised	Issues Resolved	No Test Cases	Test Case Pass
99003720	Jasleen Kaur Sethi	Feature: Arithmetic Operation, Store,Recall, Combination, LCM,GCD	7 No	6 No	38 No	38 No
99003721	Himanshu Kumar	Feature: Exponentail,Factorial,cuberoot,Log/Antilog	7 No	6 No	23 No	23 No
99003723	Vipul Sakunia	Feature: Permutation,Power,Conversion-Degree, Temperature	3 No	3 No	20 No	20 No
99003726	Abhishek Guria	Feature: Trigonometry,root,log,antilog	1 No	2 No	25 No	25 No

Figure 6 Git Dashboard 1

Summary

This project started with the research on the various calculators in the market, its age and its features. Then the requirements for our custom calculator was documented in the form of high level and low-level requirements. After the formulation of requirements, the design was made using the UML diagrams for the visual understanding of how the calculator software must be executed. After that, all the features required by the calculator are created in the form of C codes. The codes are then tested with all the possible test cases and error detection and correction is done. All the multiple files along with the headers are then executed and run with the help of a Makefile. Various building, CPP Checks and Unity testing is then done on the codes and badges for these workflows are finally attained.

Git inspector summary

```
The following historical commit information, by author, was found:

Author                    Commits    Insertions      Deletions    % of changes
99003326                        4           628              6            6.38
99003720                       69           805            939           17.56
99003721                       62          5297            497           58.34
99003723                       55           665            121            7.91
99003726                       12            86             29            1.16
Himanshu Kumar                  2            48            272            3.22
Jasleen                         1             0            276            2.78
Jasleen Kaur Sethi              3           130            132            2.64

Below are the number of rows from each author that have survived and are still
intact in the current revision:

Author                    Rows      Stability          Age       % in comments
99003326                   430           68.5          4.5                2.33
99003720                   452           56.1          5.4                0.66
99003721                  4265           80.5          0.3               10.64
99003723                   341           51.3          4.9                7.04
99003726                    40           46.5          8.3                0.00
Himanshu Kumar              29           60.4          4.6               17.24
Jasleen Kaur Sethi          37           28.5          6.4               10.81

The following history timeline has been gathered from the repository:

Author                  2021W06    2021W07    2021W10    2021W11    2021W15
99003326                      .                  +++
99003720                      .     -+++++   -----+++   ---+++++  ----+++++
99003721             +++++++++       --++     --++++         -+
99003723                      .  --+++++++         ++          +
99003726                      .         ++                      .
Himanshu Kumar                                      -
Jasleen                                             -
Jasleen Kaur Sethi            .                     .
Modified Rows:             5157        834       3906         25          9
```

Figure 7 Inspector Summary

Build

Setup done

```yaml
name: C/C++ CI - Build Status

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: make
      run: make -C 3_Implementation/ all
```

Figure 8 Setup Git

Outcome

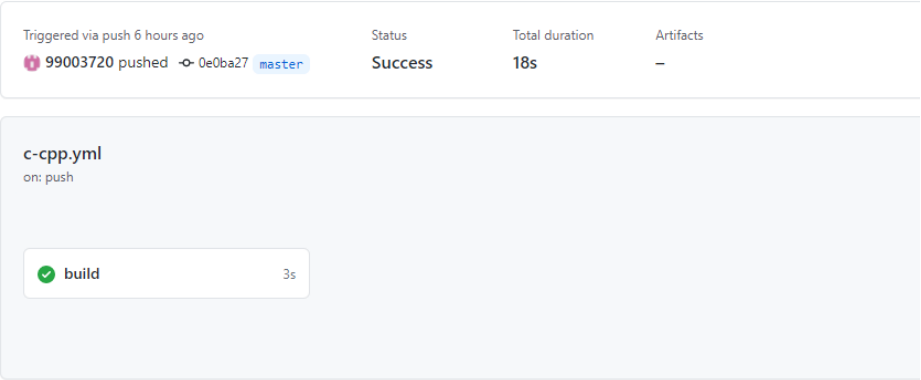


Figure 9 Build

```
Set up job
Run actions/checkout@v2
make
1  ▶ Run make -C 3_Implementation/ all
4  make: Entering directory '/home/runner/work/AppliedSDLC_Calculator_N1/AppliedSDLC_Calculator_N1/3_Implementation'
5  mkdir build
6  gcc project_main.c src/calculator_operations.c src/calc_operat.c -Iinc -Iunity  -o build/Calculator.out.out -lm
7  make: Leaving directory '/home/runner/work/AppliedSDLC_Calculator_N1/AppliedSDLC_Calculator_N1/3_Implementation'
Post Run actions/checkout@v2
Complete job
```

Figure 10 Built Action

Code quality and Issues or Bug Tracking

Setup Alignment

```yaml
name: CodeQuality Dynamic Code Analysis Valgrind
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: apt install dependency
      run: |
        sudo apt-get -y install valgrind
        sudo apt-get -y install libcunit1 libcunit1-doc libcunit1-dev
    - name: make test
      run: make -C 3_Implementation/ test
    - name: Valgrid
      run: valgrind ./3_Implementation/build/Test_Calculator.out
```

Figure 11 Setup Alignment

Outcome

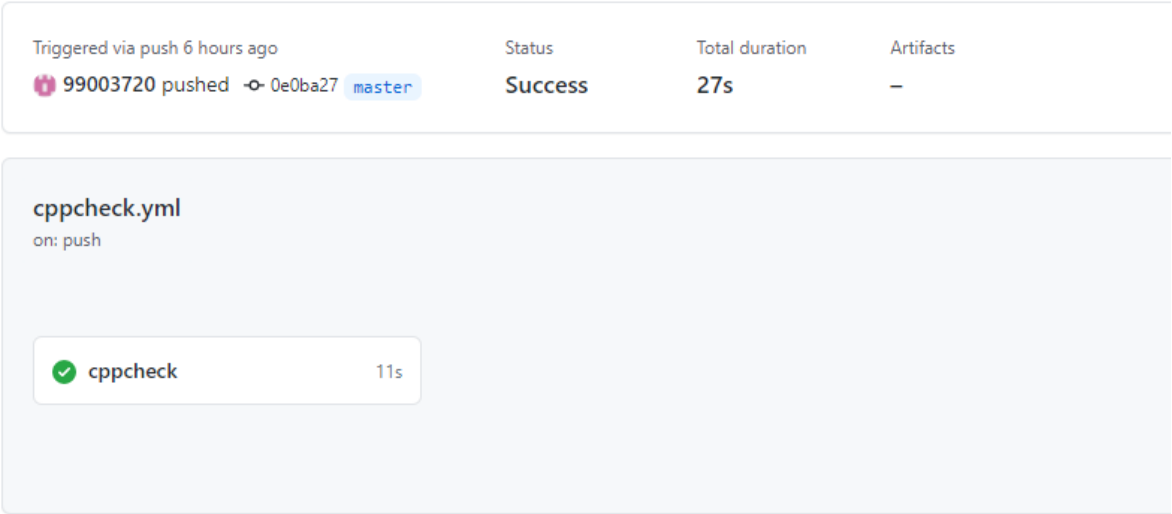


Figure 12 Cpp check

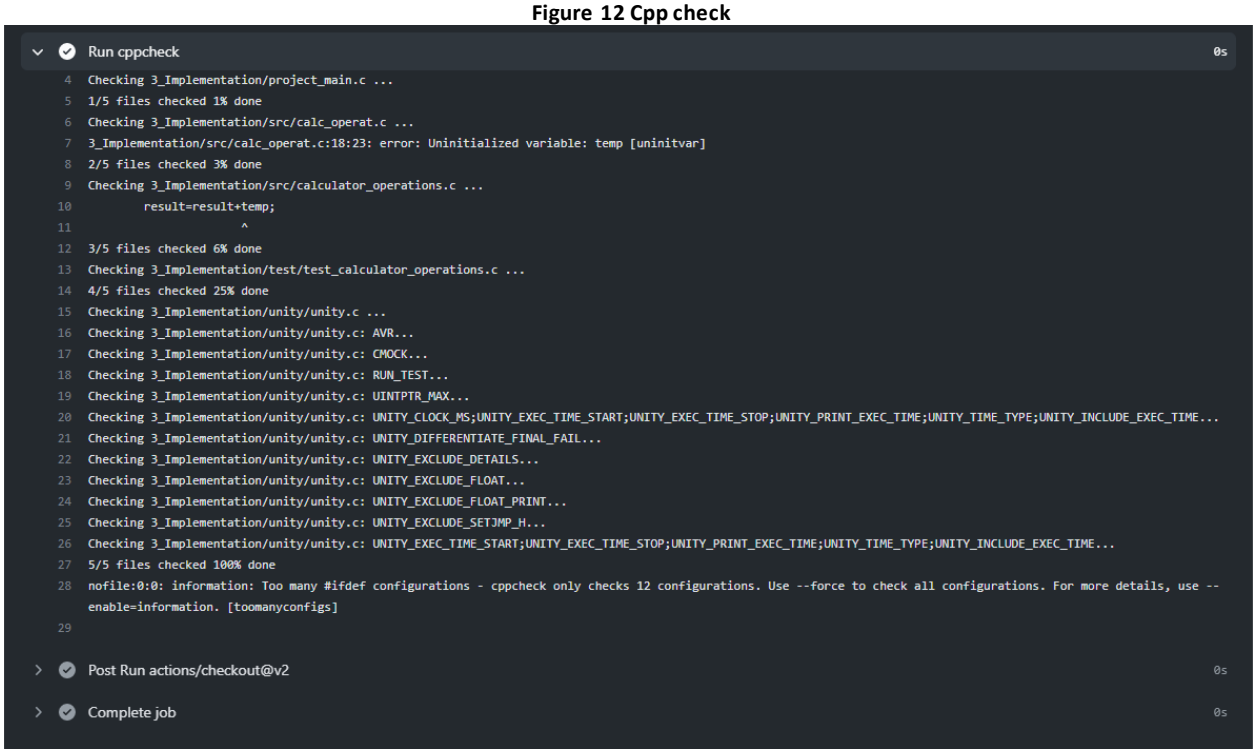


Figure 13 Cpp Run Check

Issues

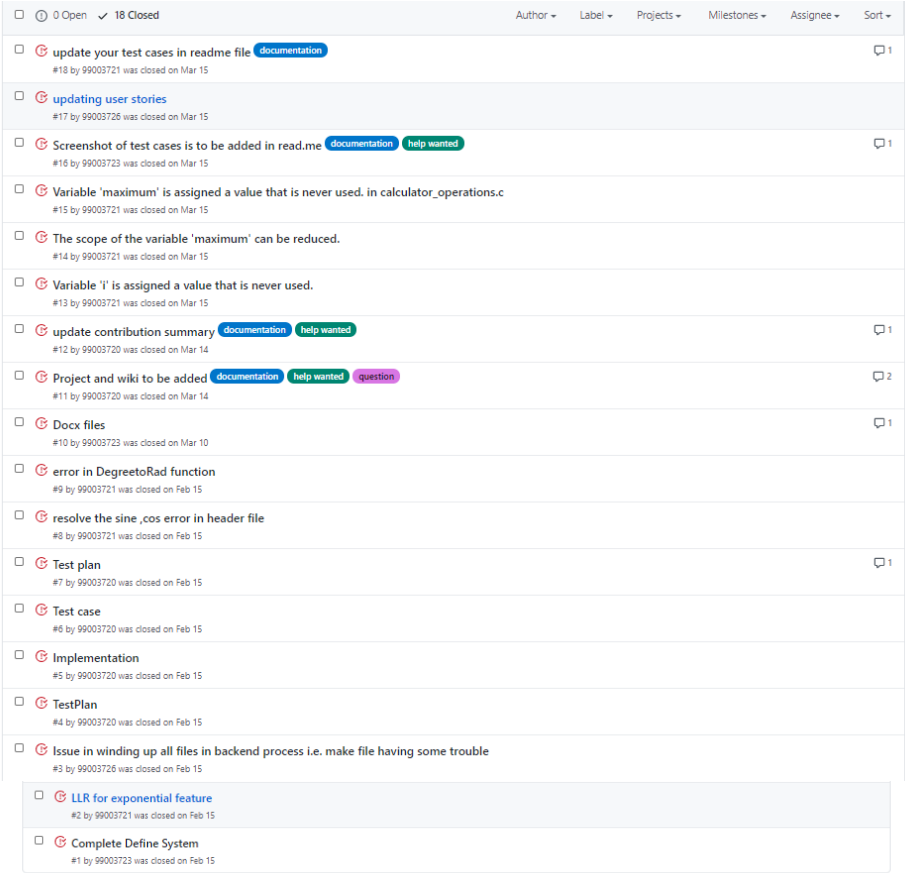


Figure 14 Issues

Unit Testing

Setup Alignment

17 lines (13 sloc) | 252 Bytes

```yaml
name: Unit Testing - Unity

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: make
      run: make -C 3_Implementation/ test
```

Figure 15 Unit Testing setup

Outcome

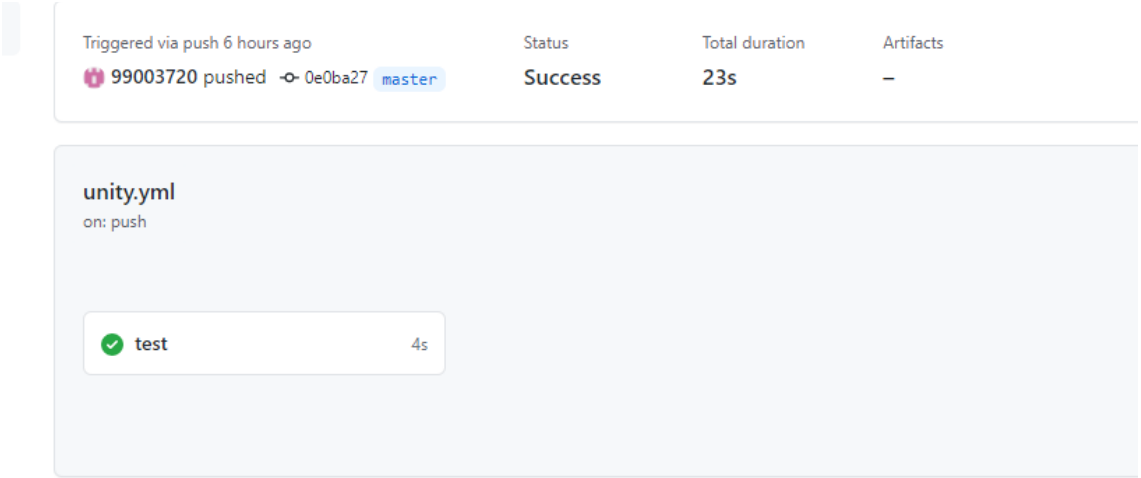


Figure 16 Unit testing Outcome

make

```
▶ Run make -C 3_Implementation/ test
make: Entering directory '/home/runner/work/AppliedSDLC_Calculator_N1/AppliedSDLC_Calculator_N1/3_Implementation'
mkdir build
gcc src/calculator_operations.c src/calc_operat.c test/test_calculator_operations.c unity/unity.c -Iinc -Iunity  -o build/Test_Calculator.out -lm
./build/Test_Calculator.out
test/test_calculator_operations.c:153:test_add:PASS
test/test_calculator_operations.c:154:test_add_testcase2:PASS
test/test_calculator_operations.c:155:test_add_testcase3:PASS
test/test_calculator_operations.c:156:test_add_testcase4:PASS
test/test_calculator_operations.c:157:test_add_testcase5:PASS
test/test_calculator_operations.c:159:test_subtract:PASS
test/test_calculator_operations.c:160:test_subtract_testcase2:PASS
test/test_calculator_operations.c:161:test_subtract_testcase3:PASS
test/test_calculator_operations.c:162:test_subtract_testcase4:PASS
test/test_calculator_operations.c:163:test_subtract_testcase5:PASS
test/test_calculator_operations.c:164:test_subtract_testcase6:PASS
test/test_calculator_operations.c:166:test_multiply:PASS
test/test_calculator_operations.c:168:test_divide:PASS
test/test_calculator_operations.c:169:test_divide_test2:PASS
test/test_calculator_operations.c:170:test_divide_test3:PASS
test/test_calculator_operations.c:171:test_divide_test4:PASS
test/test_calculator_operations.c:172:test_divide_test5:PASS
test/test_calculator_operations.c:174:test_lcm:PASS
test/test_calculator_operations.c:175:test_lcm_case2:PASS
test/test_calculator_operations.c:176:test_lcm_case3:PASS
test/test_calculator_operations.c:177:test_lcm_case4:PASS
test/test_calculator_operations.c:179:test_combination:PASS
test/test_calculator_operations.c:180:test_combination_testcase2:PASS
test/test_calculator_operations.c:181:test_combination_testcase3:PASS
test/test_calculator_operations.c:182:test_combination_testcase4:PASS
test/test_calculator_operations.c:183:test_combination_testcase5:PASS
test/test_calculator_operations.c:184:test_combination_testcase6:PASS
test/test_calculator_operations.c:186:test_permutation:PASS
test/test_calculator_operations.c:187:test_permutation_testcase2:PASS
test/test_calculator_operations.c:188:test_permutation_testcase3:PASS
test/test_calculator_operations.c:189:test_permutation_testcase4:PASS
test/test_calculator_operations.c:190:test_permutation_testcase5:PASS
test/test_calculator_operations.c:191:test_permutation_testcase6:PASS
test/test_calculator_operations.c:194:test_factorial_testcase1:PASS
test/test_calculator_operations.c:195:test_factorial_testcase2:PASS
test/test_calculator_operations.c:196:test_factorial_testcase3:PASS
test/test_calculator_operations.c:197:test_factorial_testcase4:PASS
test/test_calculator_operations.c:198:test_factorial_testcase5:PASS
test/test_calculator_operations.c:200:test_LOG_testcase1:PASS
test/test_calculator_operations.c:201:test_LOG_testcase2:PASS
test/test_calculator_operations.c:202:test_LOG_testcase3:PASS
test/test_calculator_operations.c:207:test_exponential_testcase4:PASS
test/test_calculator_operations.c:208:test_exponential_testcase5:PASS
test/test_calculator_operations.c:211:test_findSquareroot_testcase1:PASS
test/test_calculator_operations.c:212:test_findSquareroot_testcase2:PASS
test/test_calculator_operations.c:213:test_findSquareroot_testcase3:PASS
test/test_calculator_operations.c:214:test_findSquareroot_testcase4:PASS
test/test_calculator_operations.c:217:test_findcubeRoot_testcase1:PASS
test/test_calculator_operations.c:218:test_findcubeRoot_testcase2:PASS
test/test_calculator_operations.c:219:test_findcubeRoot_testcase3:PASS
test/test_calculator_operations.c:220:test_findcubeRoot_testcase4:PASS
test/test_calculator_operations.c:222:test_Sine_for_degree_testcase1:PASS
test/test_calculator_operations.c:223:test_Sine_for_degree_testcase2:PASS
test/test_calculator_operations.c:224:test_Sine_for_degree_testcase3:PASS
test/test_calculator_operations.c:225:test_Sine_for_degree_testcase4:PASS
test/test_calculator_operations.c:226:test_Sine_for_degree_testcase5:PASS
test/test_calculator_operations.c:230:test_Cosine_for_degree_testcase1:PASS
test/test_calculator_operations.c:231:test_Cosine_for_degree_testcase2:PASS
test/test_calculator_operations.c:232:test_Cosine_for_degree_testcase3:PASS
```

```
test/test_calculator_operations.c:244:test_Modulus_testcase2:PASS
test/test_calculator_operations.c:245:test_Modulus_testcase3:PASS
test/test_calculator_operations.c:246:test_Modulus_testcase4:PASS
test/test_calculator_operations.c:247:test_Modulus_testcase5:PASS
test/test_calculator_operations.c:250:test_Antilog10_testcase1:PASS
test/test_calculator_operations.c:251:test_Antilog10_testcase2:PASS
test/test_calculator_operations.c:252:test_Antilog10_testcase3:PASS
test/test_calculator_operations.c:253:test_Antilog10_testcase4:PASS
test/test_calculator_operations.c:254:test_Antilog10_testcase5:PASS
test/test_calculator_operations.c:256:test_AntilogBase2_testcase1:PASS
test/test_calculator_operations.c:257:test_AntilogBase2_testcase2:PASS
test/test_calculator_operations.c:258:test_AntilogBase2_testcase3:PASS
test/test_calculator_operations.c:260:test_power:PASS
test/test_calculator_operations.c:261:test_power2:PASS
test/test_calculator_operations.c:262:test_power3:PASS
test/test_calculator_operations.c:263:test_power4:PASS
test/test_calculator_operations.c:264:test_power5:PASS
test/test_calculator_operations.c:266:test_degreetorad:PASS
test/test_calculator_operations.c:267:test_degreetorad2:PASS
test/test_calculator_operations.c:268:test_degreetorad3:PASS
test/test_calculator_operations.c:269:test_degreetorad4:PASS
test/test_calculator_operations.c:270:test_degreetorad5:PASS
test/test_calculator_operations.c:272:test_tempconv:PASS
test/test_calculator_operations.c:273:test_tempconv2:PASS
test/test_calculator_operations.c:274:test_tempconv3:PASS
test/test_calculator_operations.c:275:test_tempconv4:PASS
test/test_calculator_operations.c:276:test_tempconv5:PASS

-----------------------
94 Tests 0 Failures 0 Ignored
OK
make: Leaving directory '/home/runner/work/AppliedSDLC_Calculator_N1/AppliedSDLC_Calculator_N1/3_Implementation'
```

Figure 17 Unit Testing Outcome2

Individual Contribution & Highlights

The features implemented by me are:

Permutation and Combination

- Requirements and design for the permutation.
- Implemented code for permutation of two positive numbers

Power

- Requirements and design for the power functions
- Implemented code for power of two positive numbers

Conversion of Units

- Requirements and design for conversion of temperature and degree.
- The low-level design for the operations
- Various test cases for the operations including all kinds of requirements.

Summary

- Through this project, I learnt about team work and managing the team. Communication between the team members was the most essential part of working in a team so as to execute and complete all the requirements in the given deadline.
- I improved my C programming coding skills with number of different functions implemented by me and my team members.
- I improved my leadership skills by ownership and accountability of work done.

Challenges faced and how were they overcome

1. Running the make file as its resolved by defining its correct path (.out for Linux and -lm for math functions)
2. Synchronizing the VS code to GitHub, colleague help to resolve the issue
3. Making the function call in correct path
4. Open git log while committing, thus went to GitHub desktop and pulled origin and then pushed origin.
5. Test case code for the boundary problem. Added code with the help of internet

Future Scope

Every project can be well executed and properly defined when it is designed and deployed as per the Software Development Life Cycle.

Python [Individual]

Module

The modules linked to the mini project is Python.

Topic and Subtopics

Python Programming

- Strings
- Loops
- Exception Handling
- Classes and Objects
- Hacker Rank- Python Learning
- Solo Learn – Python
- Pytest

Tools

- PyCharm
- Notepad++
- VScode
- PEP8

Objectives & Requirements

Problem Statement was to find the five input words in the given source file i.e., .txt file and give the output in a new file by creating it. The output must contain one word before and after the input word (desired word).

High Level requirement analysis

ID	Requirements	Description	Status
HL_1	Searching Word	Search the word from the text file given by the user.	Implemented
HL_2	Writing	Write the words after and before of the searched word in new text file.	Implemented
HL_3	Extracting user defined data	Write required data in the text file.	Implemented

Low Level requirement analysis

ID	Requirements	Description	Status
LL_1	Search Word	The user inputs the word to be searched through the text file.	Implemented

LL_2	Searching word in every line of the text file.	The data to be searched is defined by the user and then searched throughout the entire text file.	Implemented
LL_3	Writing the data into new text file.	Data given as an input by the user is searches in every line and the word after and before is extracted too and then added in a new text file along the searched word.	Implemented

Code and Output

word_finder.py > ...

You, 2 months ago | 1 author (You)

```python
import re
```

You, 2 months ago • wordstarted

```python

xas = int(input("enter the no. for words: \n"))


class special_character():
    def __init__(self):

        self.string = open('input.txt').read()  # opening input file
        self.new_string = re.sub(r'[^a-zA-Z\n]', ' ', self.string)
        # regex for replacing all non alphabetic characters with whitespace
        self.text_file = open("sample.txt", "w")
        # creating and writing the new string in sample file
        self.n = self.text_file.write(self.new_string)
        self.text_file.close()  # closing the sample file

    def input_word(self):

        self.count = 0
        self.findword = input("Enter the word:\n")
        # input for the word to be search
        pattern = re.compile(self.findword, re.I | re.M)
        # pattern for the word making it insensitive of the case
        hand = open(r'sample.txt')
        handle = hand.read()
        self.wordfinder = re.findall(pattern, handle)
        self.file_list = handle.split()  # spliting the handle

    def creative_file(self):
        create_file = self.findword + '.txt'
        # creating the file of same name of the search word and writing
        file_gen = open(create_file, 'w+')
        for i in range(len(self.file_list)):
            wordfound = re.match(self.findword, self.file_list[i], re.I | re.M)
            if wordfound:
                self.count += 1
```

Figure 18 Code Python

```
                sent = (self.file_list[i-1] + ' ' + self.file_list[i] + ' ' +
                        self.file_list[i+1])
                file_gen: IO g one word before and after the input search word
                file_gen.write(str(sent) + '\n')
                file_gen.write(str(self.count) + ' :')

        file_gen.write('word repeated:' + str(len(self.wordfinder)) + '\n')
for v in range(xas):
        obj = special_character()
        obj.input_word()
        obj.creative_file()
```

```
3.680753044\pythonFiles\lib\python\debugpy\launcher'
enter the no. for words:
5
Enter the word:
software
Enter the word:
license
Enter the word:
document
Enter the word:
GNU
Enter the word:
version
PS C:\Users\99003723\Desktop\Python\Mini_project>
```

```
MINI_PROJECT
demo.py
document.txt        U
GNU.txt             U
input.txt
license.txt         U
README.md
sample.txt          U
software.txt        U
version.txt         U
word_finder.py
```

Figure 19 Output Python 1

```
Free Software Foundation
1 :most software are
2 :free software to
3 :the software is
4 :designated software packages
5 :Free Software Foundation
6 :free software we
7 :free software and
8 :the software and
9 :Finally software patents
10 :GNU software including
11 :free software developers
12 :free software only
13 :free software For
14 :any software library
15 :of software functions
16 :such software library
17 :a software library
18 :for software interchange
19 :free software distribution
20 :of software distributed
21 :distribute software through
22 :Free Software Foundation
23 :Free Software Foundation
24 :Free Software Foundation
25 :For software which
26 :Free Software Foundation
27 :Free Software Foundation
28 :free software and
29 :of software generally
30 :OTHER SOFTWARE EVEN
31 :free software that
32 :free software you
33 :Free Software Foundation
```

Figure 20 Output Python 2

Test Plan

ID	Description	Input	Output
T_01	Should get entered word output as a txt file and number of repetitions	software	35
T_02	Should ask the user to enter the number of	Run the program	Enter the no. of words

	inputs		

Git Link

https://github.com/99003723/Mini_project

Git Dashboard

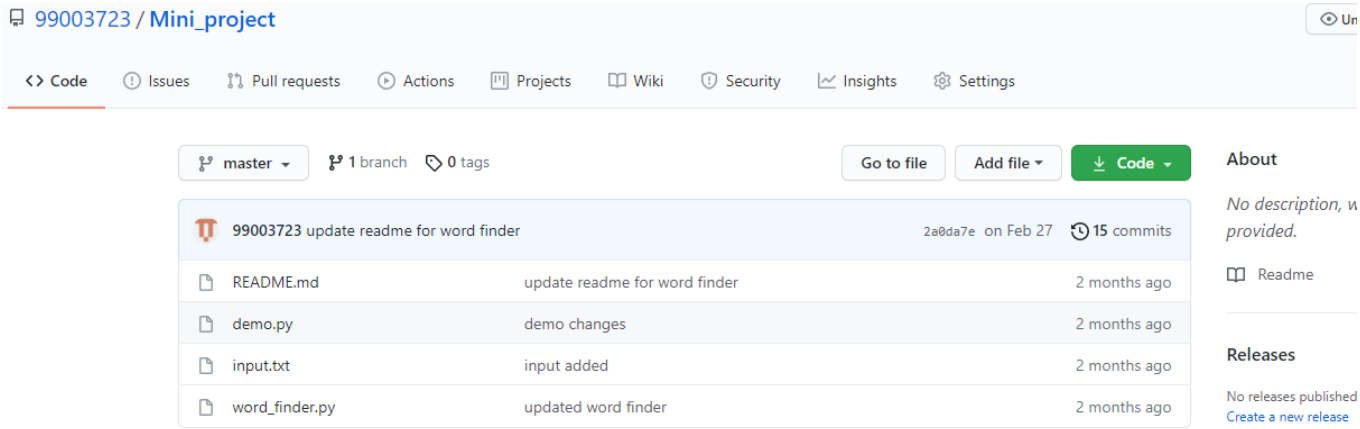


Figure 21 Git dashboard Python

Summary

The program asks the user to input the number of words, he/she wants to search. The output is provided to the user as a txt file.

I learned to work in different code editors like pycharm and VScode and how to build and execute a program by constructing classes and structures of code. Different test methods to check the quality of code was implemented to get a standardized finish according to the industry levels.

Challenges faced and how were they overcome

- Executing the code using class structure was confusing but with project work it got cleared.
- Coding in different editors was challenging at first, but as the work progressed I achieved a flow to approach the problem. Applying SDLC also helped to get a clear idea about the project.

Embedded-C [Team]

Module

Module linked to the BCM features is Embedded C.

Topic and Subtopics

- Make Start Up file
- Linker Script
- Debugging techniques
- Driver library creation & HAL library
- Stm32cube-IDE

Objectives & Requirements

- Implementation of BCM Features in STM32cube-IDE and dump in to microcontroller(ARM-cortex-M4) and verify its functionality.
    1. Hood release control
    2. Exterior lighting
    3. Automatic Wiper control
    4. Internal lighting

Design and Test Plan:

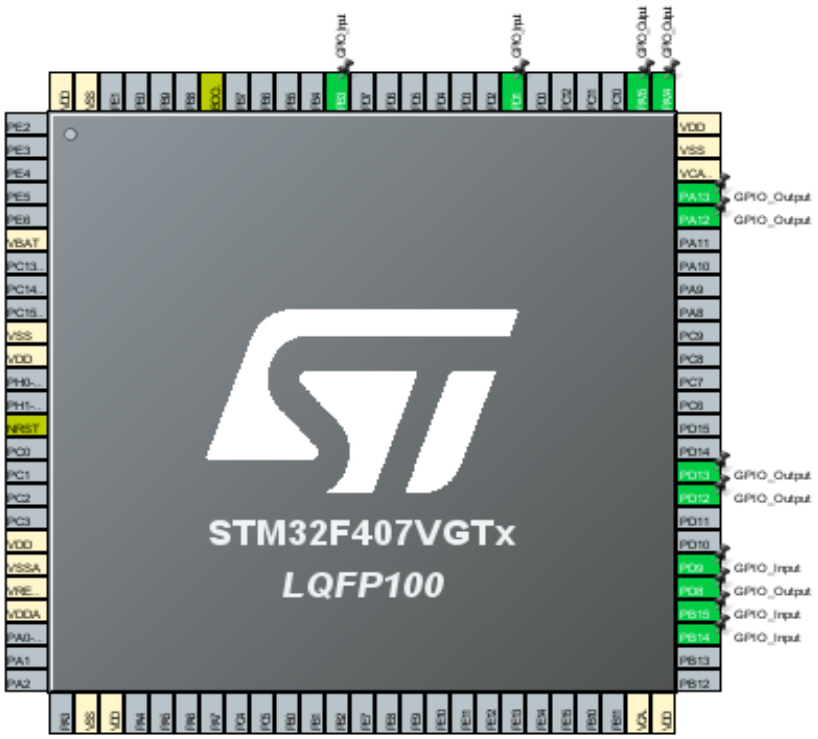


Figure 22 Pin Configuration

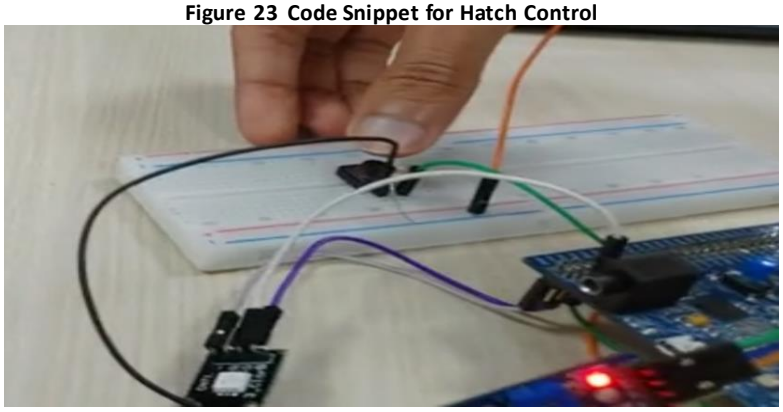
Above mentioned features are implemented in the code snippet given below:

1. Hood release control

```
/*************Manual Opening of back trunk***********/
   if (!(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_9)))
    {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);

    }
    else
    {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        //HAL_Delay(2000);
    }
```

Figure 23 Code Snippet for Hatch Control



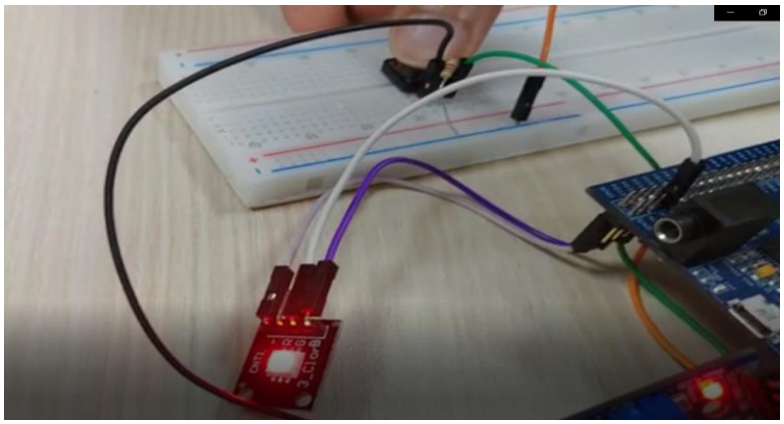


Figure 24 Microcontroller implementation of Hood control from switch

2. Exterior Lighting Using LDR

```
if(!(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14)))
  {
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET);
  }
  else
  {
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
  }
```

Figure 25 Code snippet for Exterior Lighting

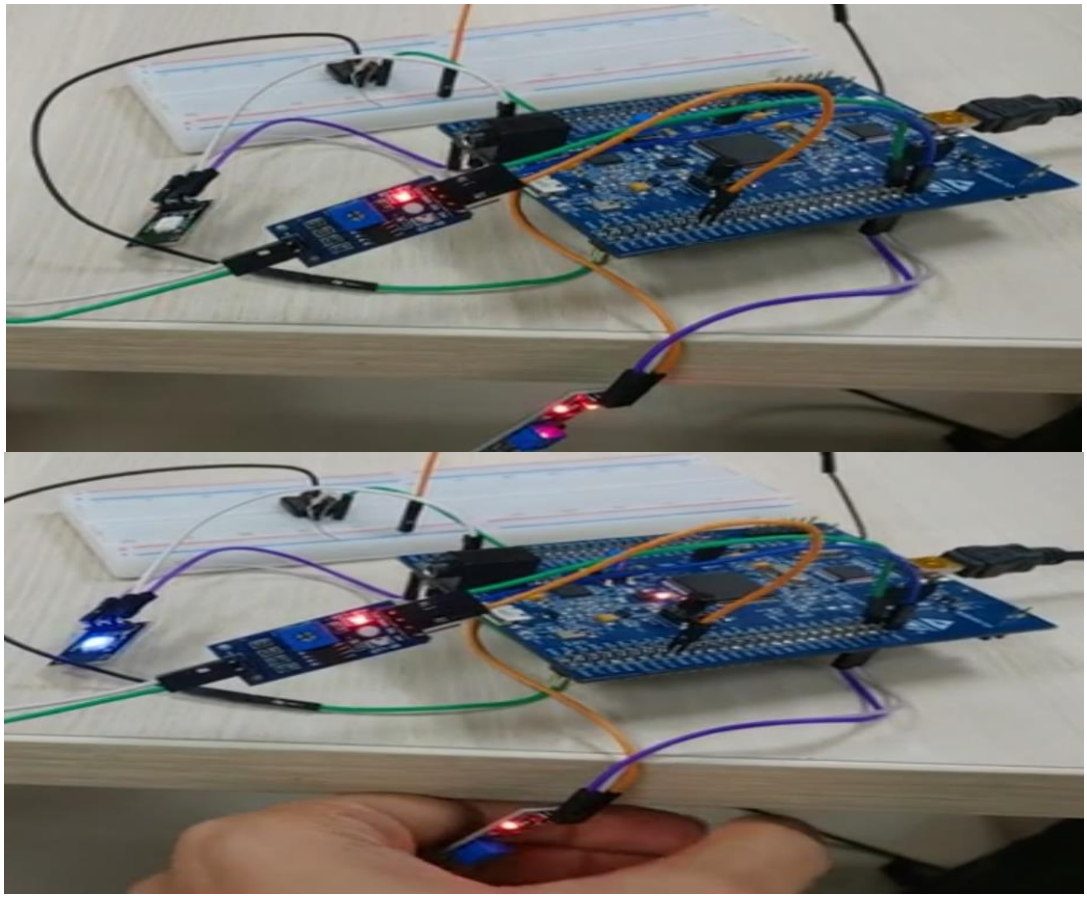


Figure 26 MC implementation of Exterior lights using LDR sensor

3. Interior light switching using switch

```
if (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3)==GPIO_PIN_RESET)  /*Internal light Switching*/
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_SET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_SET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,GPIO_PIN_SET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,GPIO_PIN_SET );
}
else
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_RESET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,GPIO_PIN_RESET );
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,GPIO_PIN_RESET );
}
```

Figure 27 Code snippet for Interior lighting

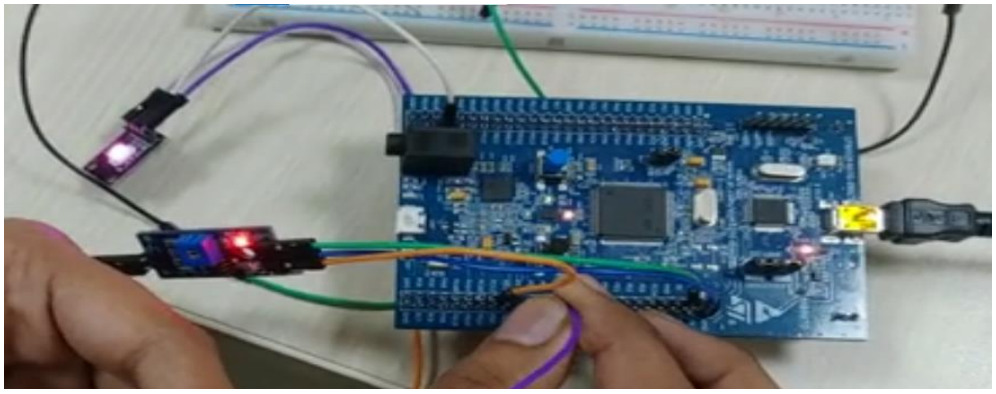


Figure 28 Image of the microcontroller implementation of interior lighting (Pink Light)

4. Automatic wiper control

```
if (HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_1) == 0)              /*Wiper control system*/
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_SET );
}
else
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_RESET );
}
```

Figure 29 Code snippet for Wiper control

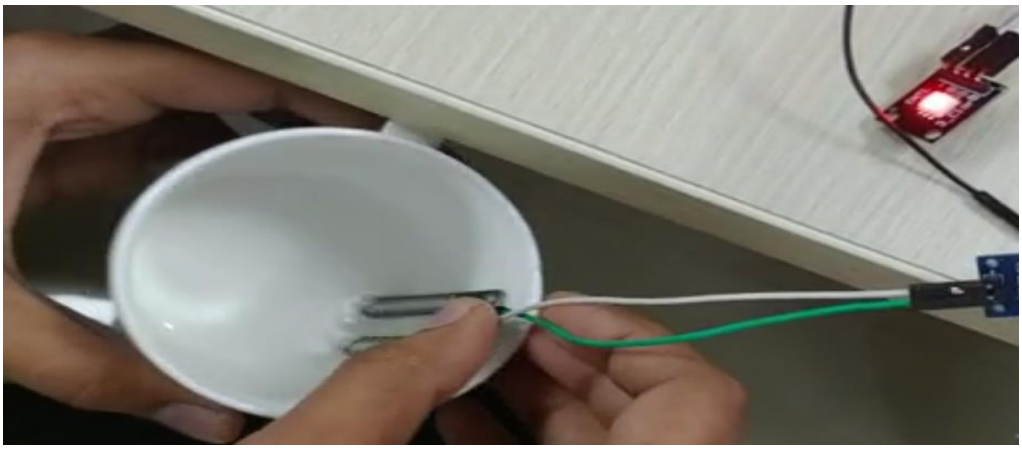


Figure 30 Microcontroller implementation of Wiper Control

Implementation Summary

Connected the STM32 Board and made some configuration so that HAL libraries are added in to the micro controller and did the PIN configuration according to the features and wrote the program accordingly and verified it in the micro controller.

Individual Contribution & Highlights

Code contribution in final code – Designing the Hood release & Exterior lighting.

Summary

Outcomes:

1. STM32f407 discovery board programming.
2. Programming the features BCM module: Hood release and exterior lights using LDR sensor.

Challenges faced and how were they overcome

Challenges	Resolution
Creation of drivers	HAL Driver and later created own driver also.
Integration of model	Using IDE

MBSE [Team]

To implement different Body Control Module (BCM) feature present in a car using MATLAB and Simulink modelling. To test the respective feature and whole integrated system using Manual testing and Automated testing.

Here GMC Sierra truck is taken as a reference to build the model.

Sr. no.	Module
1.	MATLAB & Simulink
2.	Logical operations.
3.	State flow

BCM and comfort module should have following features:

1. Engine status (key position)
2. Seat belt reminder
3. Power outlet
4. Gear control
5. Steering control
6. Exterior lighting
7. Interior lighting
8. Cruise control system
9. Tire pressure management system
10. Wiper control
11. Power window
12. Hood Release

Objectives & Requirements

High Level requirements for Engine status, Power Outlet, Gear Control, Seat Belt reminder:

ID	Requirements
HLR_ES_1	Ignition Key position
HLR_ES_2	Engine state output through lamp with different colors.
HLR_PO_1	Engine Status
HLR_PO_2	Power Outlet Push button to switch.
HLR_PO_3	Power Outlet display for output.
HLR_GC_1	Engine Status
HLR_GC_2	Radio button to change the gear position.
HLR_GC_3	Gear display for output.
HLR_SBR_1	Engine Status
HLR_SBR_2	Toggle Switch to input the position of seat belt.
HLR_SBR_3	Should remind the driver to fasten seat belt through bickering lamp.
HLR_SBR_4	Gear Mode Status.

Low Level requirements:

ID	Requirements
LLR_ES_1	4 positions of key ignition for engine status OFF, ACC, ON/RUN, START
LLR_ES_2	Key position different outputs for different features so as to integrate the model accordingly.
LLR_PO_1	Outlet should be OFF if engine status is OFF.
LLR_PO_2	Outlet should be ON if engine status is ACC.
LLR_PO_3	Outlet should be ON if engine status is ON/RUN.
LLR_PO_4	Outlet should be ON if engine status is START.
LLR_GC_1	For active gear control Engine Status should be at START mode.
LLR_GC_2	4 gear positions Park, Neutral, Drive, Reverse.
LLR_GC_3	Gear output for display in dashboard.
LLR_GC_4	Gear output for Seat belt reminder.
LLR_GC_5	Gear transition through state flow from the input of gear mode from the user.
LLR_SBR_1	Engine status should be in ACC, ON/RUN or START for SBR to activate.
LLR_SBR_2	Buckle sensor should give output as a reminder to the user through display if seat belt not buckled.
LLR_SBR_3	Buckle sensor should stop reminding when seat buckle is closed.
LLR_SBR_4	SBR should be activated when the gear is in any position other than park.
LLR_SBR_5	Gear output for input Seat belt reminder.
LLR_SBR_6	Seat belt reminder implementations through state flow.

Design

High Level Design:

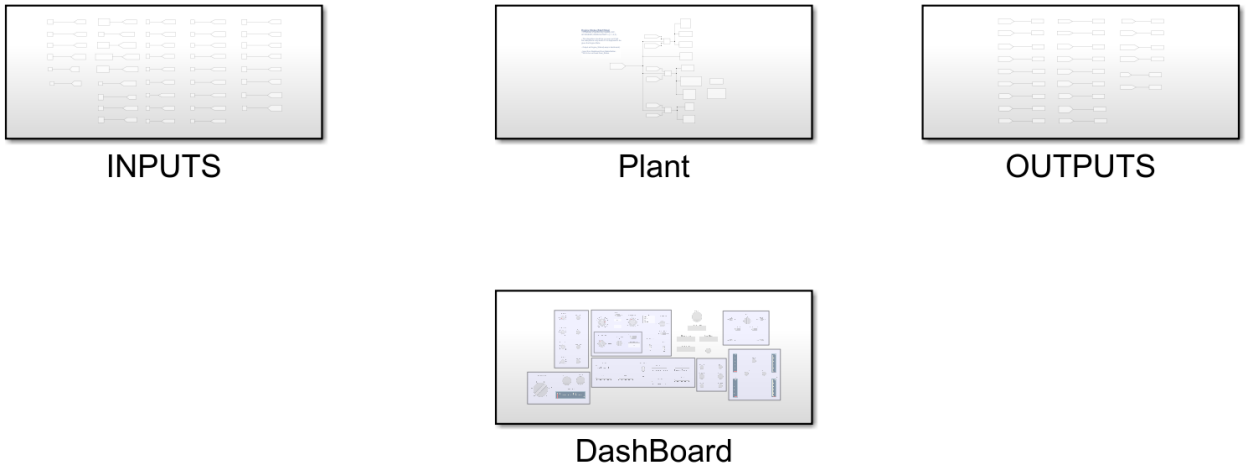


Figure 31 High Level MBSE Design

Inputs:

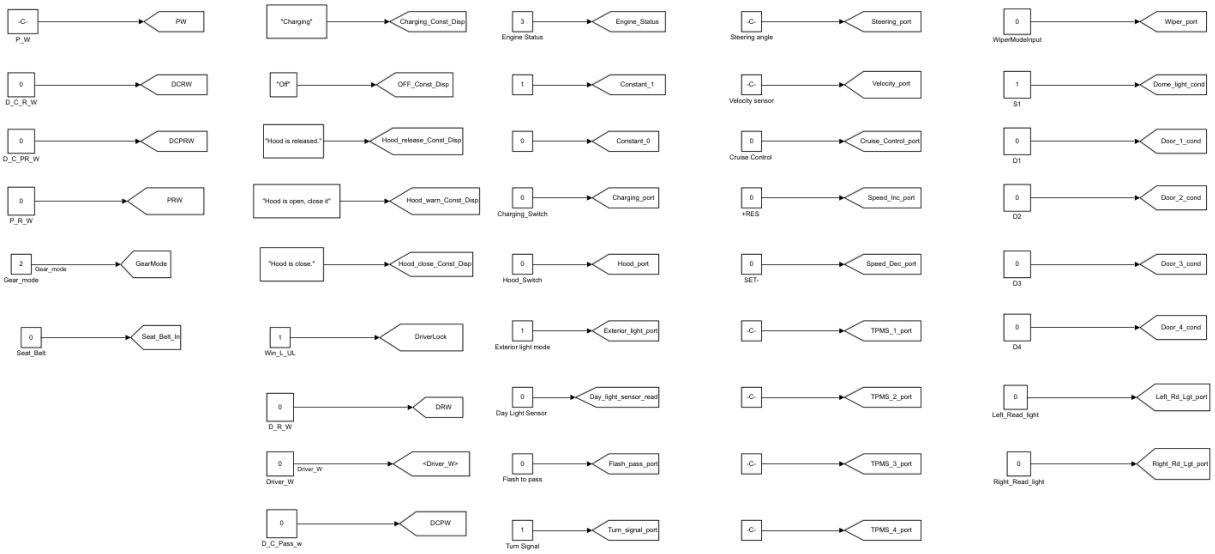


Figure 32 MBSE Input

Outputs:

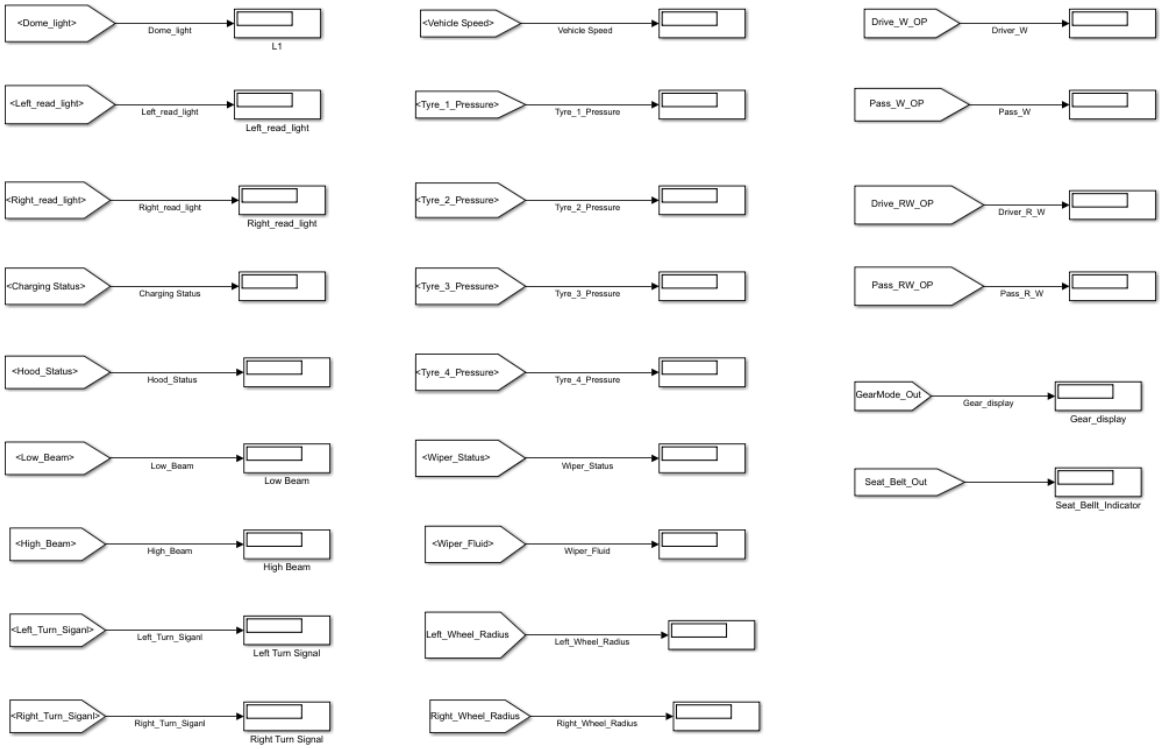


Figure 33 MBSE Output

Dashboard:

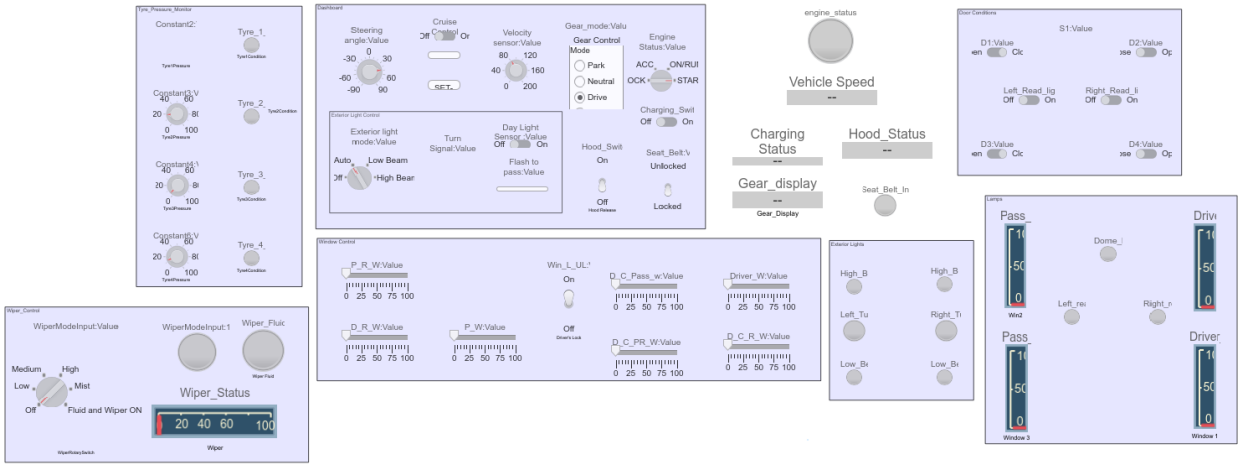


Figure 34 Dashboard

Plant:

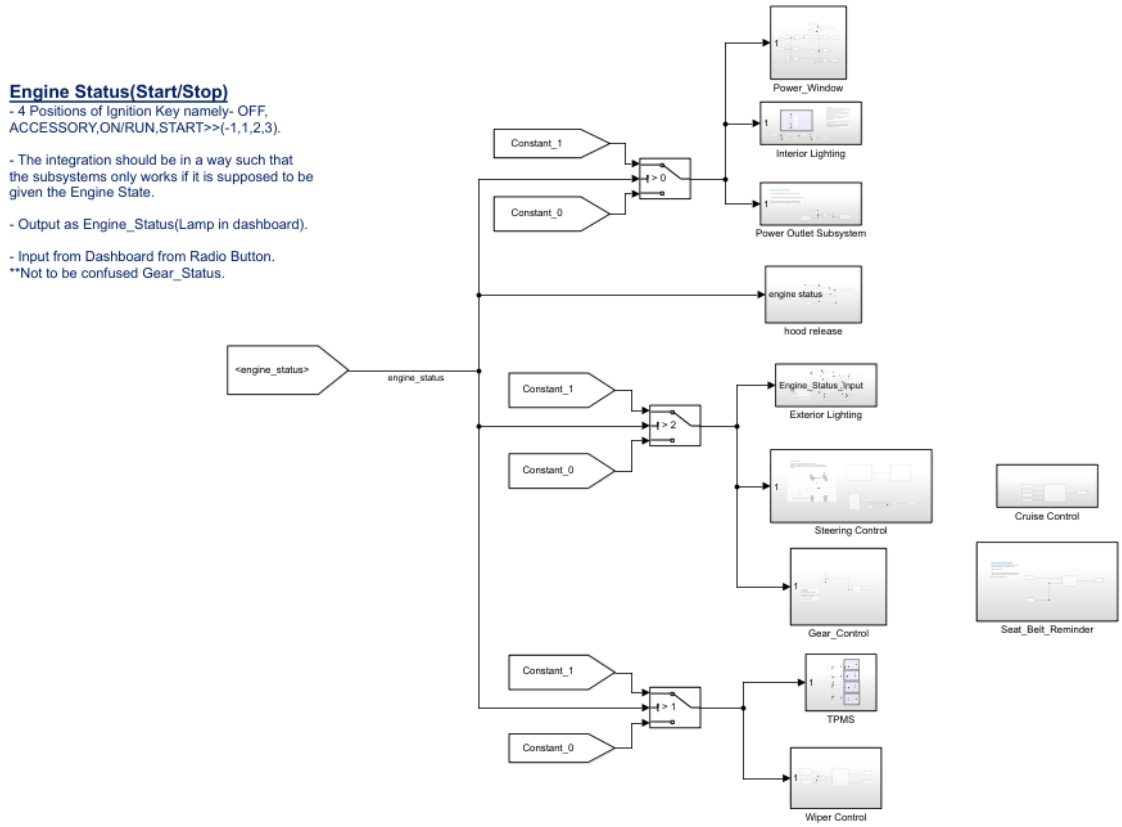


Figure 35 MBSE Plant

Low Level Design for Individual System:

1. Engine Status/Key Ignition Position:

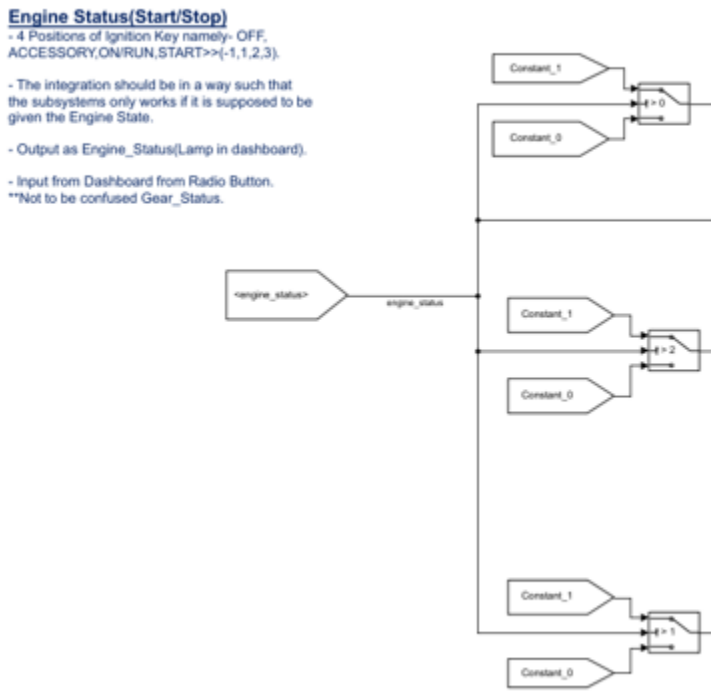
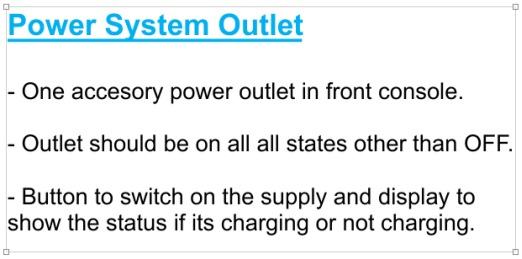


Figure 36 Engine Status/Key Ignition

2. Power Outlet:



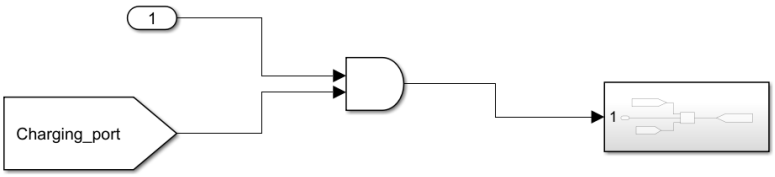


Figure 37 Power Outlet

3. Gear Control:

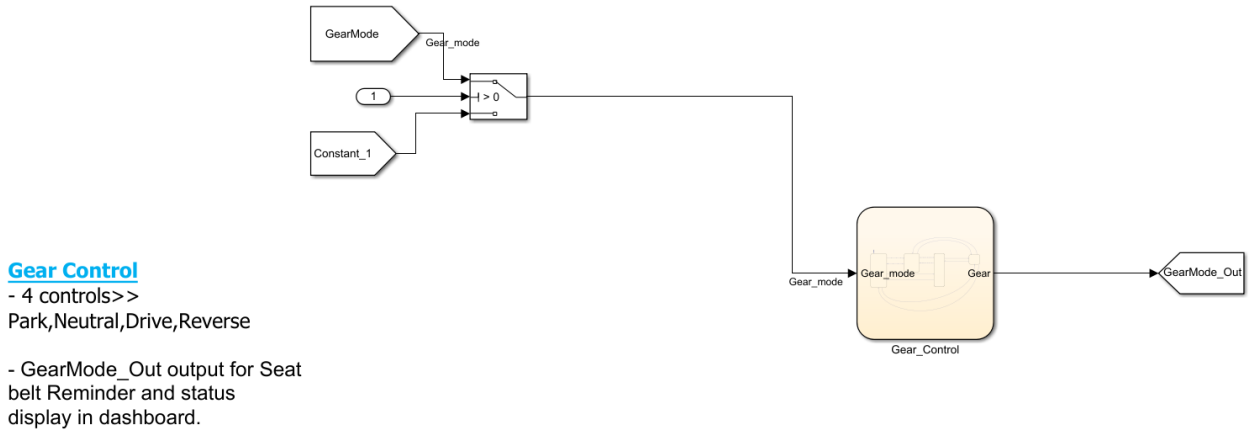


Figure 38 Gear Control

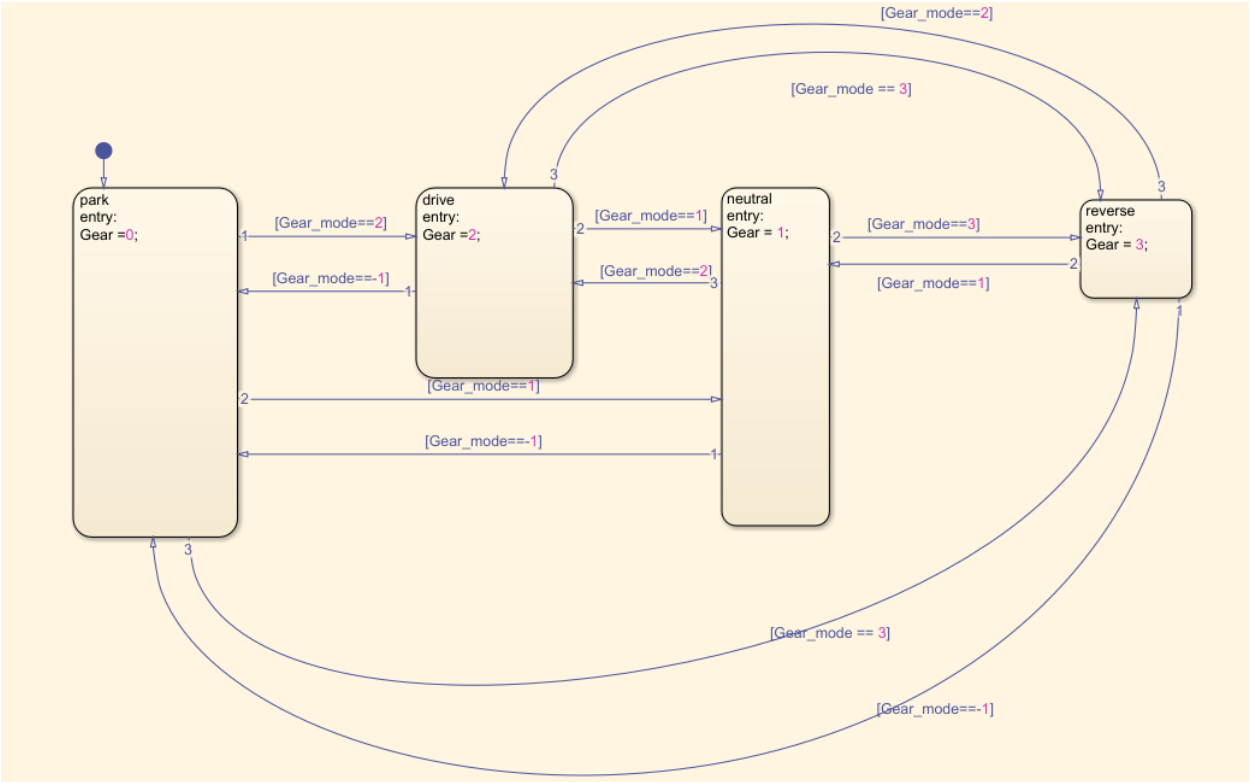


Figure 39 Gear Control State flow

Gear positions Output:

- Park= 0
- Neutral= 1
- Drive= 2
- Reverse= 3

4. Seat Belt Reminder:

Seat Belt Reminder

- Lamp indicator for reminding the user about the status of the seat belt. (Seat_Belt_Out)

- Input from the Gearmode_Out so as to only remind when the gear position is in Drive(2) or Reverse(3).

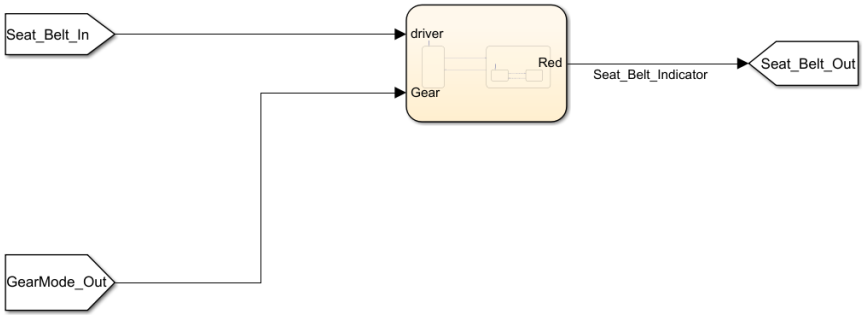


Figure 40 Seat Belt reminder MBSE

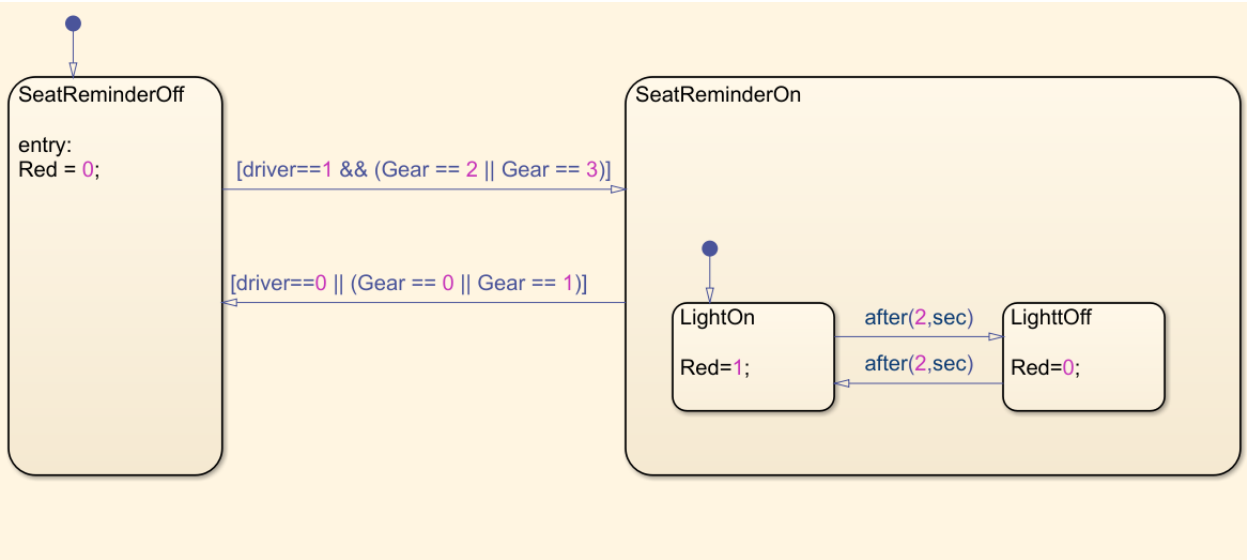


Figure 41 SBR Sate flow Implementation MBSE

Test Plan

ID	Input	Output
ES_1	Change knob position from STOP to START	Green light in lamp
PO_1	Engine status in ACC and charging mode slider on	Charging on display
PO_2	Engine status in START and charging mode slider OFF	Not Charging

GC_1	In Park Position	1 in Gear display
GC_2	Engine in Start position and gear In Neutral Position	2
GC_3	In Reverse Position	3
SBR_1	Engine in START mode Gear Position in Park and seat belt toggle in locked	No blinking of lamp
SBR_2	Engine START Gear at Drive Seat belt unlocked	Blinking of lamp

Implementation Output

- The figure below shows the following outputs:
    1. Engine status is shown which is in ON/RUN position.
    2. Charging Port Is ON.
    3. Seat belt indicator is OFF as seat belt is in locked position.
    4. Gear position is neutral position i.e., Gear display output as 1.

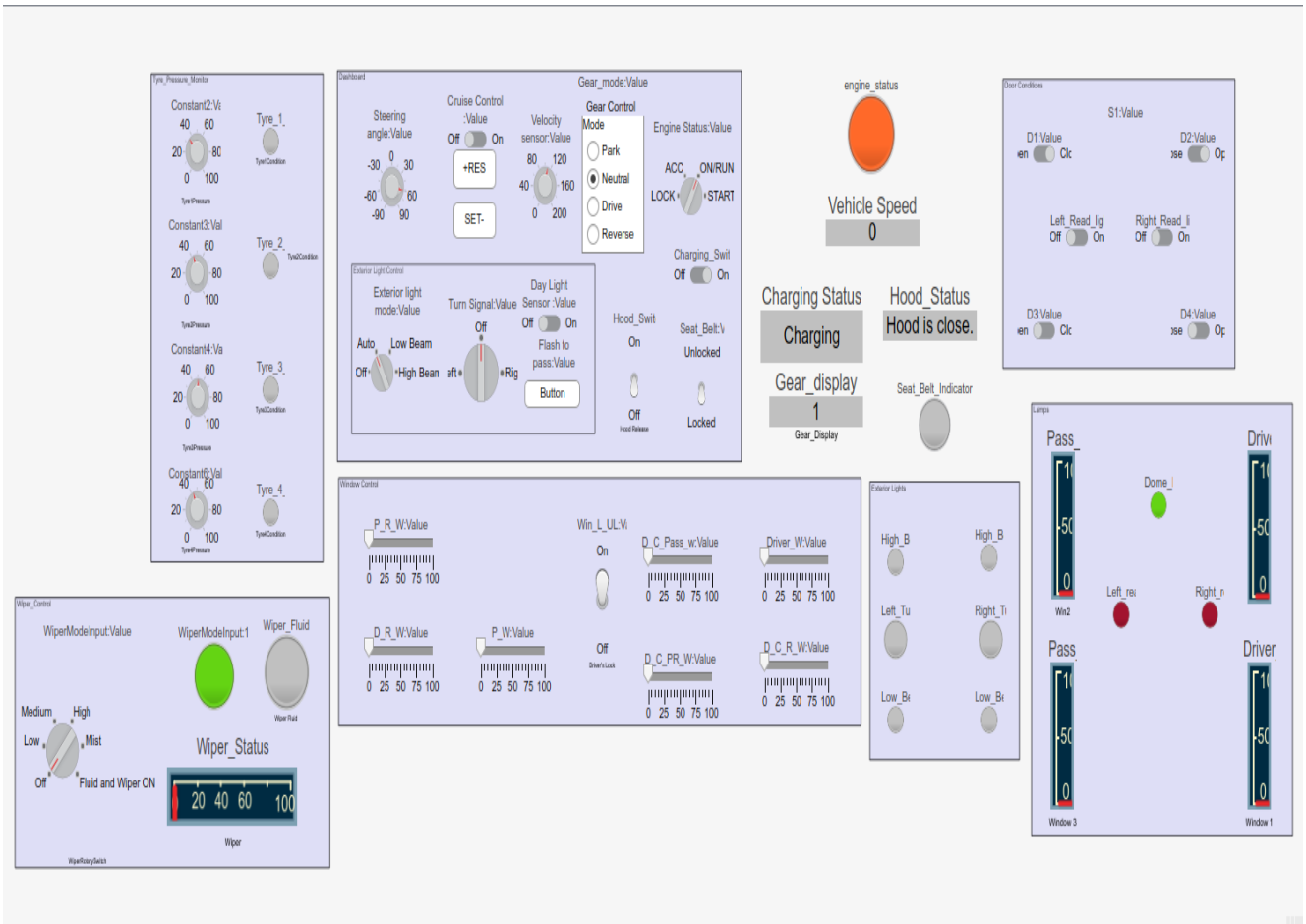


Figure 42 Output for Manual Implementation 1

- The figure below shows the following outputs:
    1. Engine status is shown which is in START position.
    2. Charging Port Is ON.
    3. Seat belt indicator is ON as seat belt is in unlocked position.
    4. Gear position is Drive position i.e., Gear display output as 2.

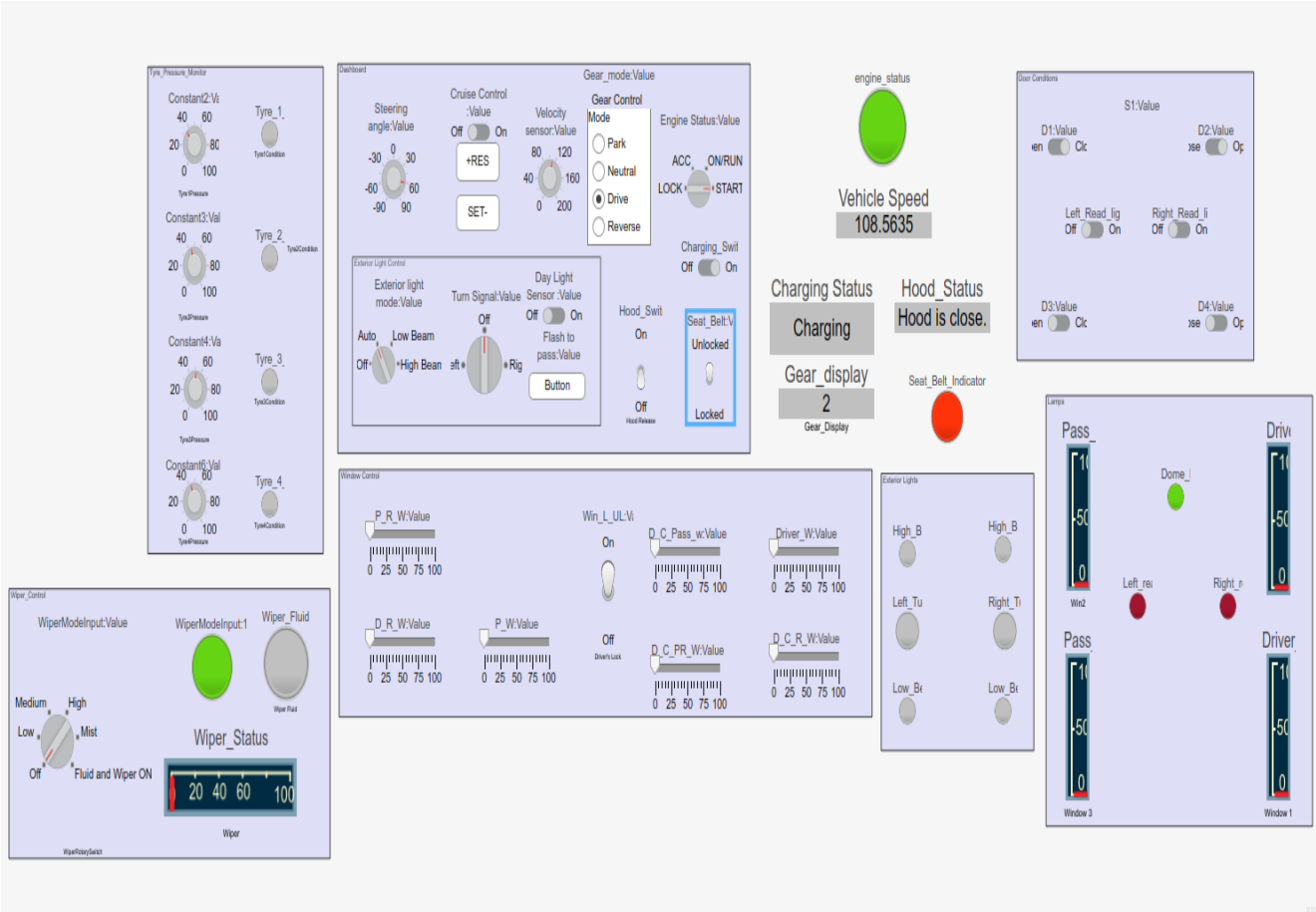


Figure 43 Output for Manual Implementation 2

Implementation Summary

All features are integrated in such a way that they work only when corresponding Engine status is ON. Designing of the BCM features i.e. Engine status, Power Outlet, Gear Control and Seat belt reminder using MATLAB and Simulink. Designing of the manual testing model and automated testing models. Integration and manual testing was done.

Summary

We went through an extensive research about the BCM features present in GMC Sierra. We tried to build the models in Simulink with the help of state flow. Proper conventions were maintained while building the models. We also did manual as well as testing of our individual models. Integration testing was also performed combining all the features of the group members.

Individual Contribution & Highlights

Role in team	Description
Subsystem designing	Designing of the subsystems – Engine Status, Power Outlet, Gear Control, Seat belt reminder.

Gathering requirement	Requirements and test cases.
Integration	Integration of subsystems input & outputs.

Challenges faced and how were they overcome

Challenges	Resolution
Requirement	Using 2020-gmc-Sierra-Owners-Manual
Integration of whole model	Was overcome with the help of each team members clear idea of implementation from the start.

Automotive systems and overview-MBD Evolved[Team]

The automotive central body control module (BCM) is a processor-based power distribution center that supervises and controls functions related to the car body such as lights, windows, security, door locks and access control, and various comfort controls. The features implemented here are Power window, Tire Pressure Management System, Cruise Control, Exterior Lights and seatbelt reminder. HIL testing for TPMS and exterior lights.

Features

Seat Belt Reminder:

Seat Belt Reminder (SBR) sensor mat detects occupants in passenger and rear seats. This information is then used by a seat belt reminder system in the vehicle to trigger a warning light and/or an audible chime reminding unbuckled seat occupants to fasten their seat belts. Research studies indicate that the risk of dying in a crash could be reduced by about 60% by using the seat belt and by more, when belts and air bags are combined (WHO 2004). While most drivers wear seat belts in the fronts of cars, a significant proportion involved in crashes are unrestrained, even in countries with highest seat belt use.

Objectives & Requirements

The reasons drivers indicate most often as to why they do not use a seat belt include short trips, forgetfulness, in a rush, and discomfort. Basic seat belt reminder system –The reminder or warning system can consist of a visual signal (a blinking icon or text display) or an acoustic signal of varying pitch. A combination of both signal types is often used for the front seat belts. A warning light and a tone activated immediately after the vehicle is started and continues if a driver is not belted, with the light persisting for at least 60 seconds before getting solid indication.

High Level Requirements:

ID	Requirements
HLR_SBR_1	Should remind the driver to fasten seat belt through display.
HLR_SBR_2	Engine Status.
HLR_SBR_3	Seat Sensor and Buckle Sensor
HLR_SBR_4	Gear Mode Status.

Low Level Requirements:

ID	Requirements
LLR_ SBR_1	Seat Sensor should be activated when driver seats upon it.
LLR_ SBR_2	Buckle sensor should give output as a reminder to the user through display and chime if seat belt not buckled.
LLR_ SBR_3	Buckle sensor should stop reminding when seat buckle is closed.
LLR_ SBR_4	SBR should be activated when the gear is in any position other than park.
LLR_ SBR_5	Engine status should be in ACC, ON/RUN or START for SBR to activate.

Design

High level:

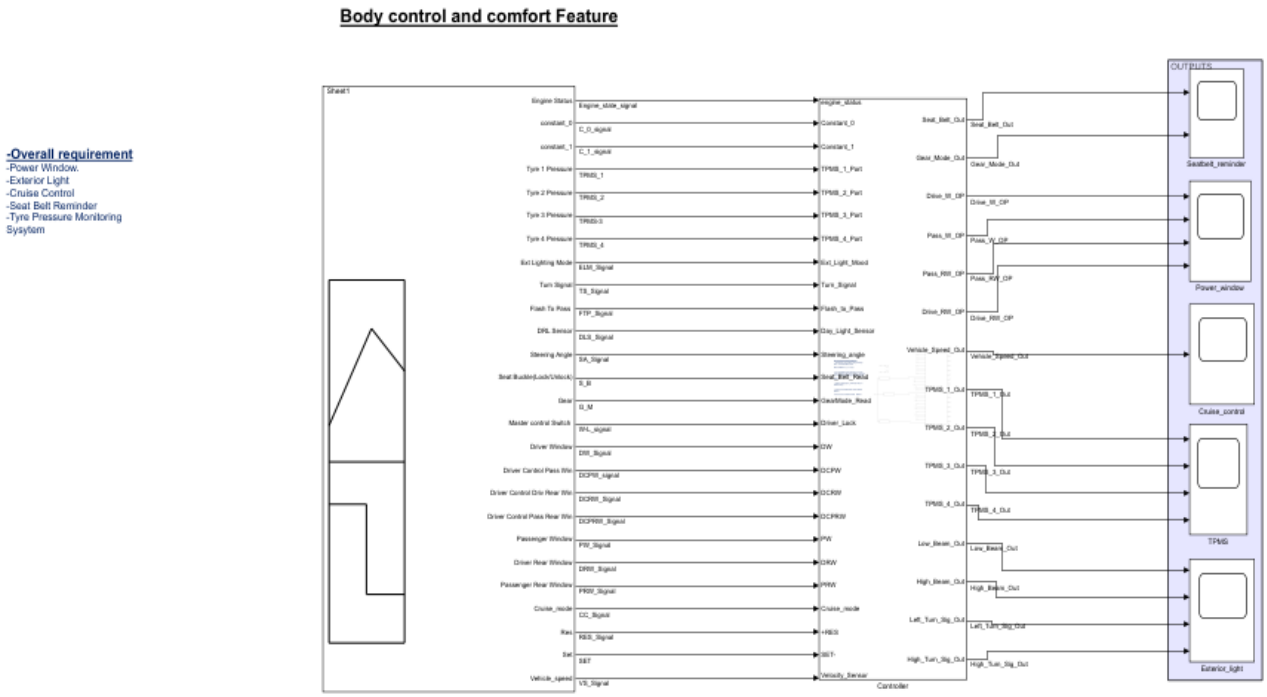


Figure 44 High level Design

Low Level:

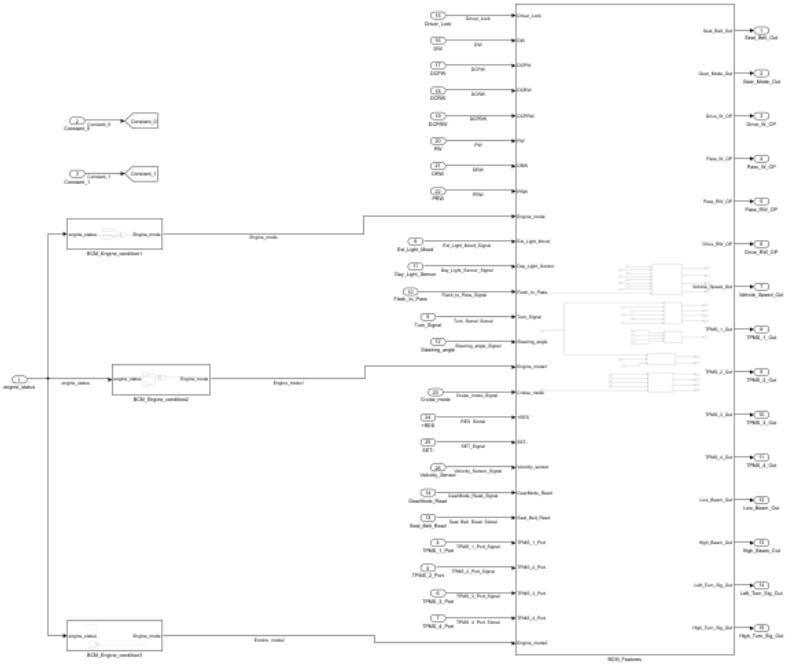


Figure 45 Low Level design

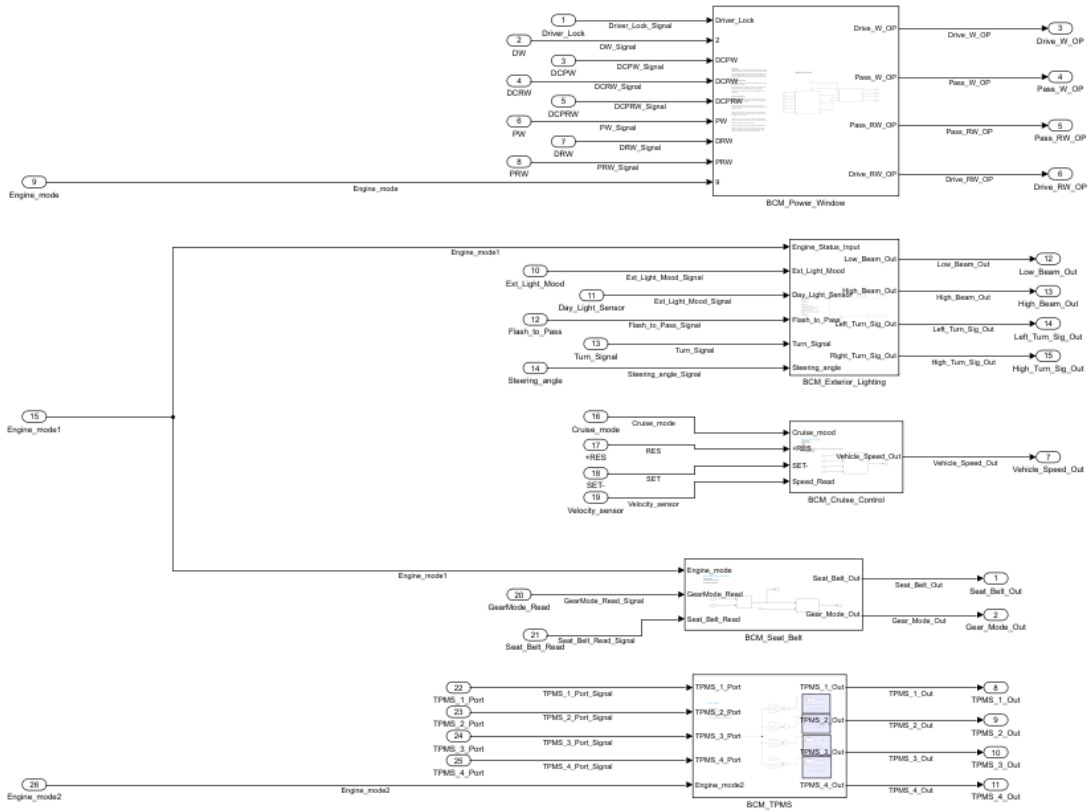


Figure 46 Evolved Plant

Feature Model

Seat Belt Reminder:

Passenger seat not considered.

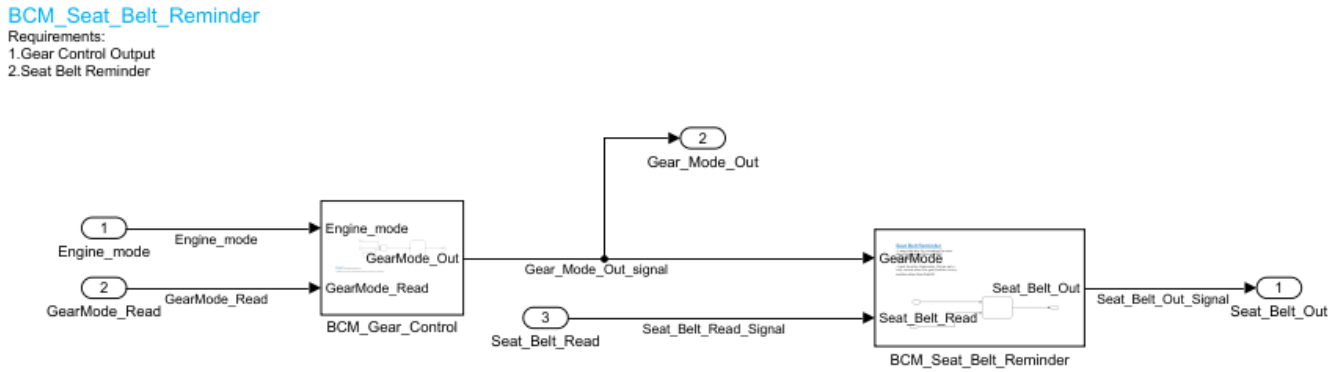
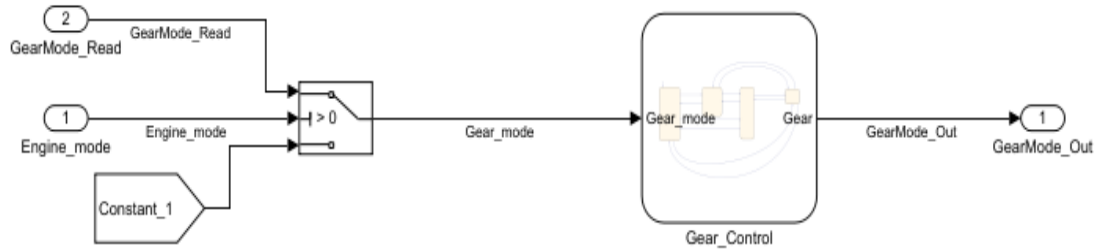


Figure 47 Seat Belt reminder model



Gear Control

- 4 controls>> Park,Neutral,Drive,Reverse

- GearMode_Out output for Seat belt Reminder and status display in dashboard.

Figure 48 Gear Control for Input to SBR

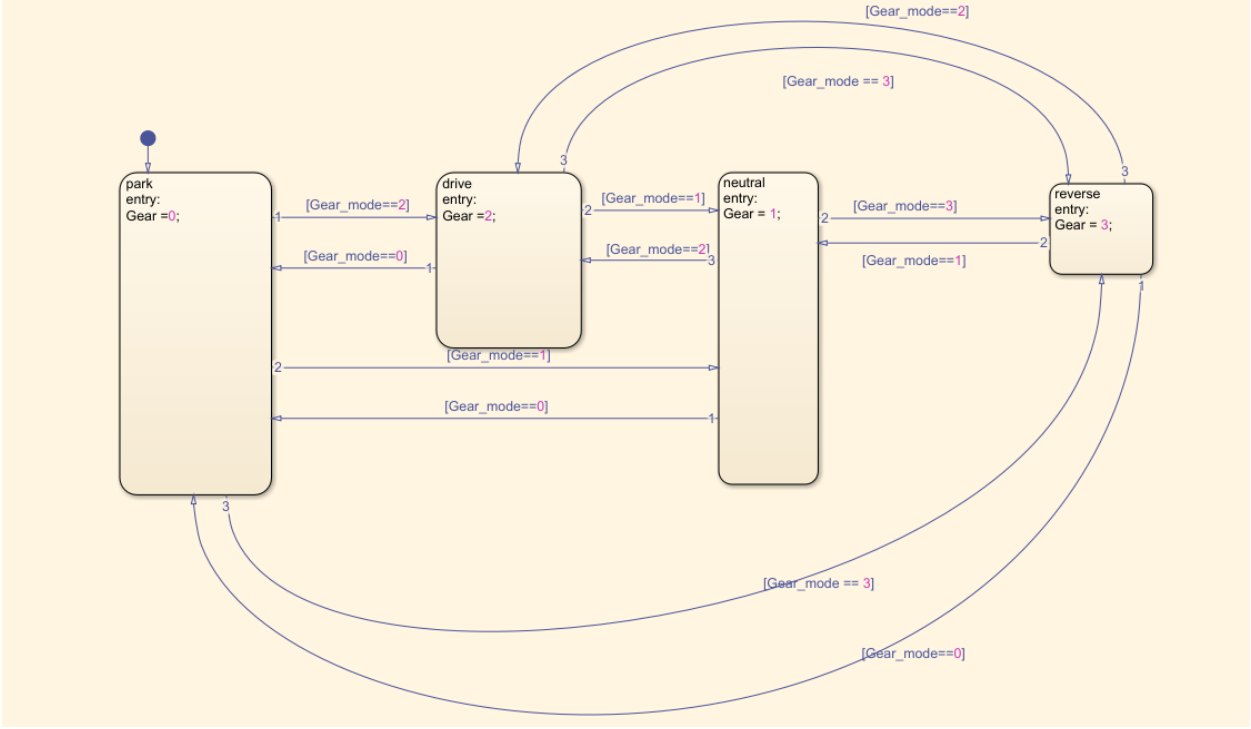


Figure 49 Gear Implementation State Flow

Seat Belt Reminder

- Lamp indicator for reminding the user about the status of the seat belt. (Seat_Belt_Out)

- Input from the Gearmode_Out so as to only remind when the gear position is any position other than Park(0).

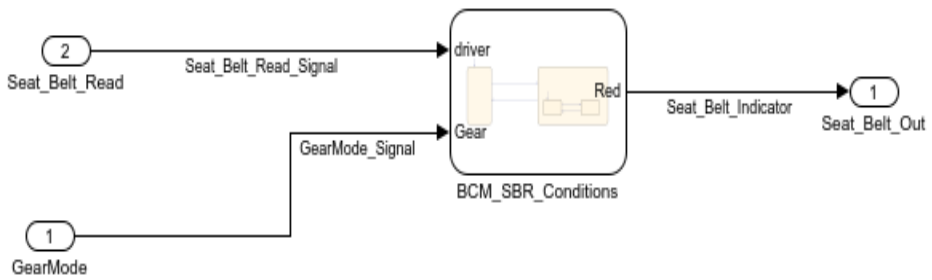


Figure 50 SBR low level model

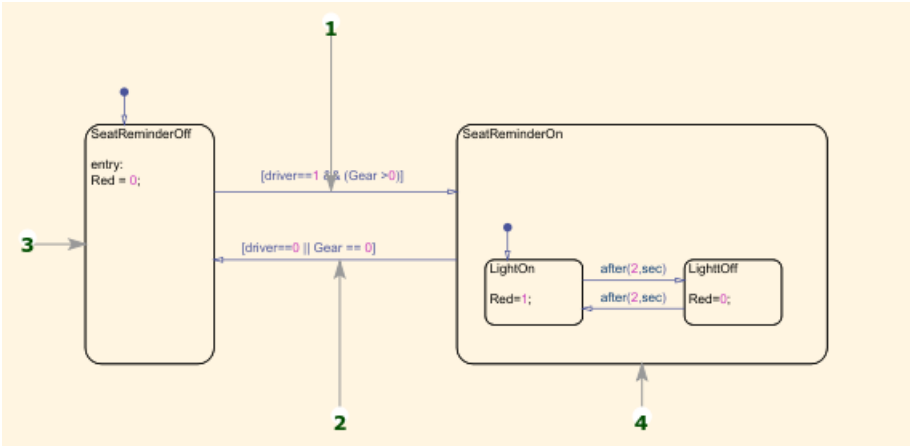


Figure 51 State flow for SBR

Algorithm:

- Seat Belt Reminder

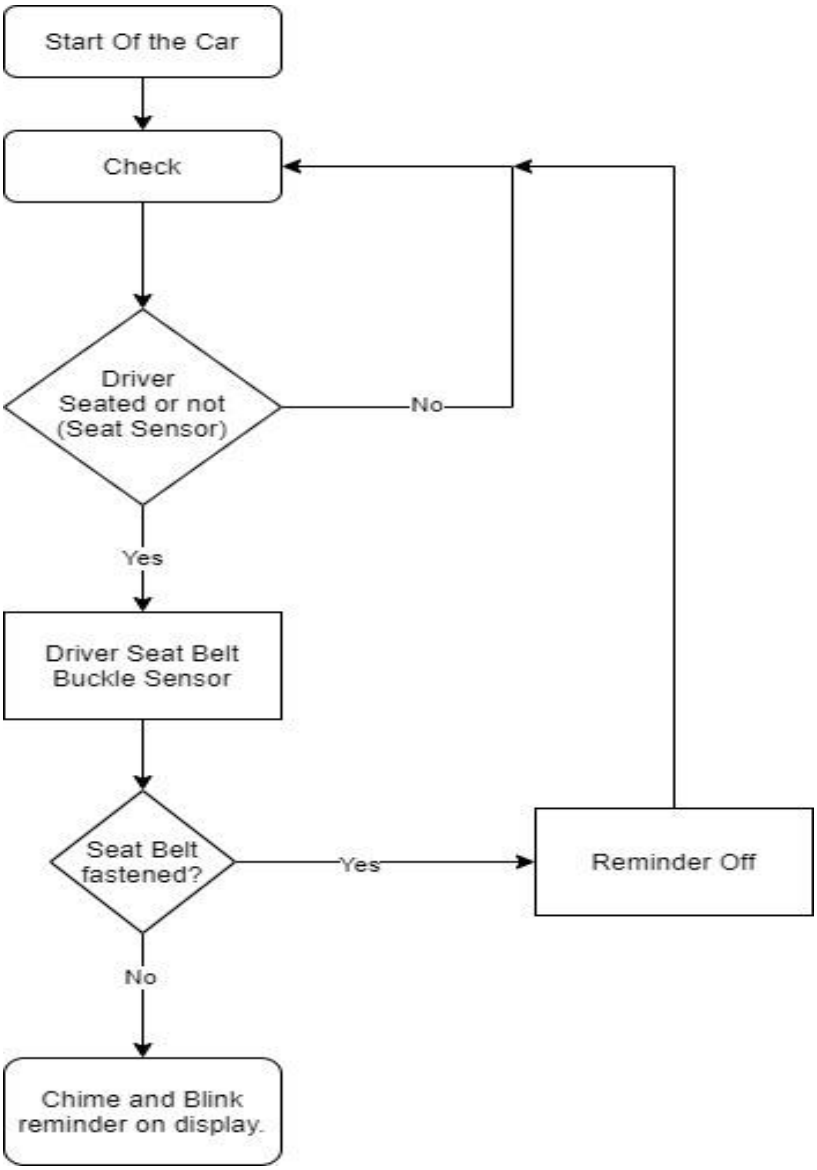


Figure 52 Algorithm SBR

SWOT Analysis:

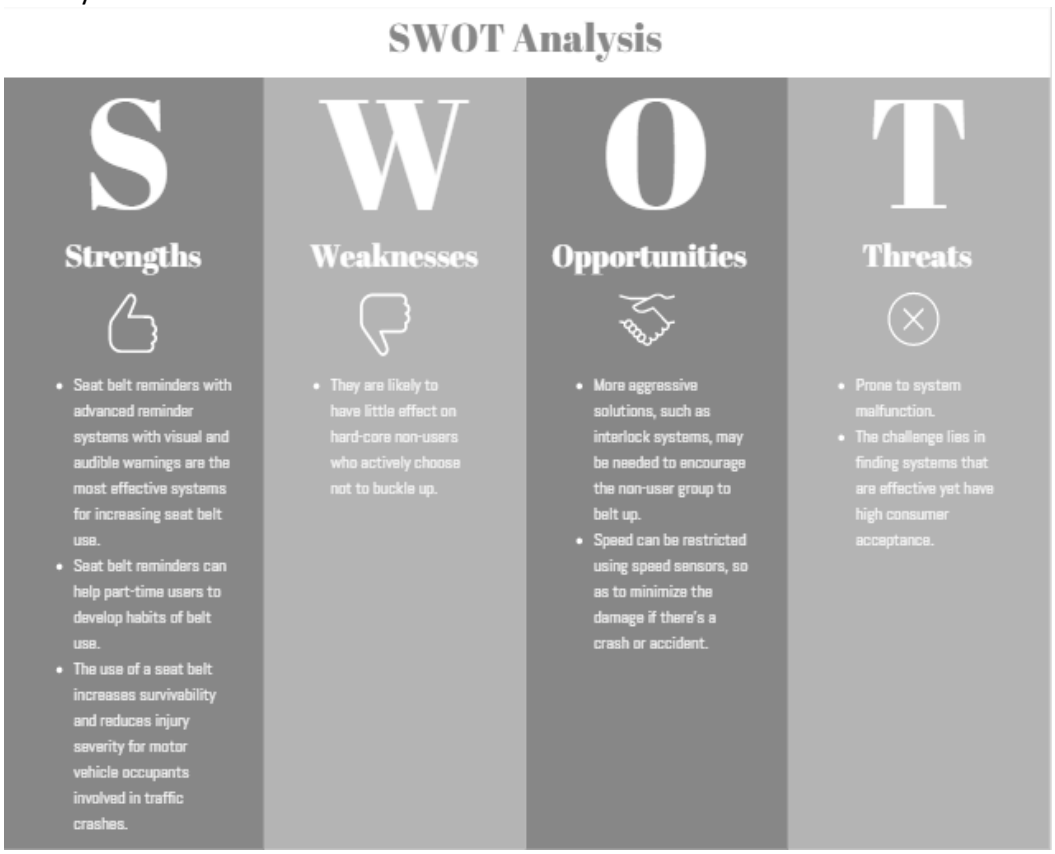


Figure 53 SWOT for SBR

Test Plan

Refer Excel sheet for the test plan: Reference link for test cases excel sheet

HIL Testing for TPMS and Exterior Lights

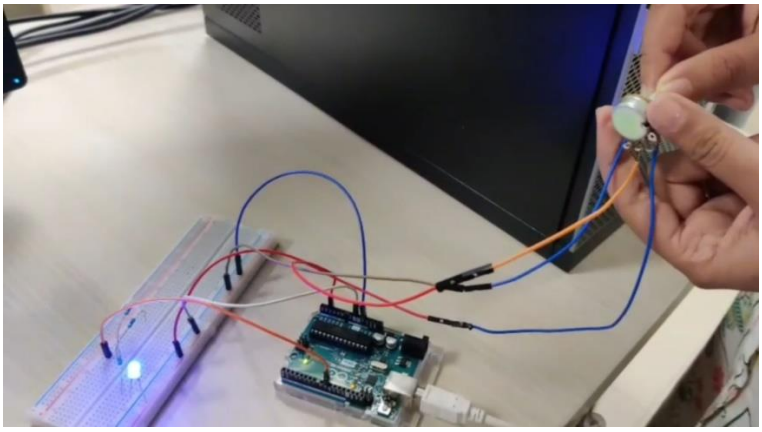


Figure 54 HIL testing for TPMS

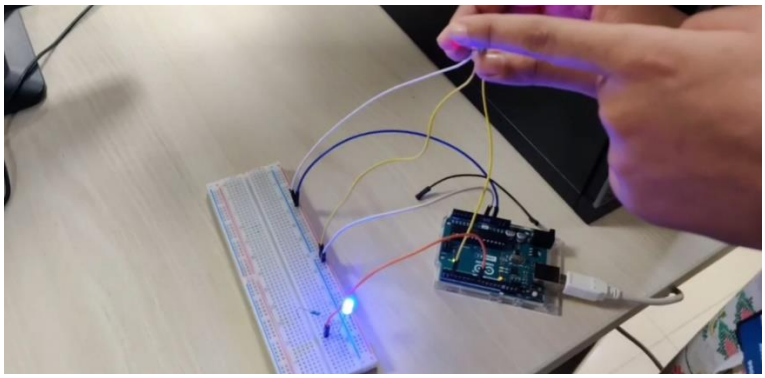


Figure 55 HIL testing for Exterior Lights

Implementation

Model Advisor Report

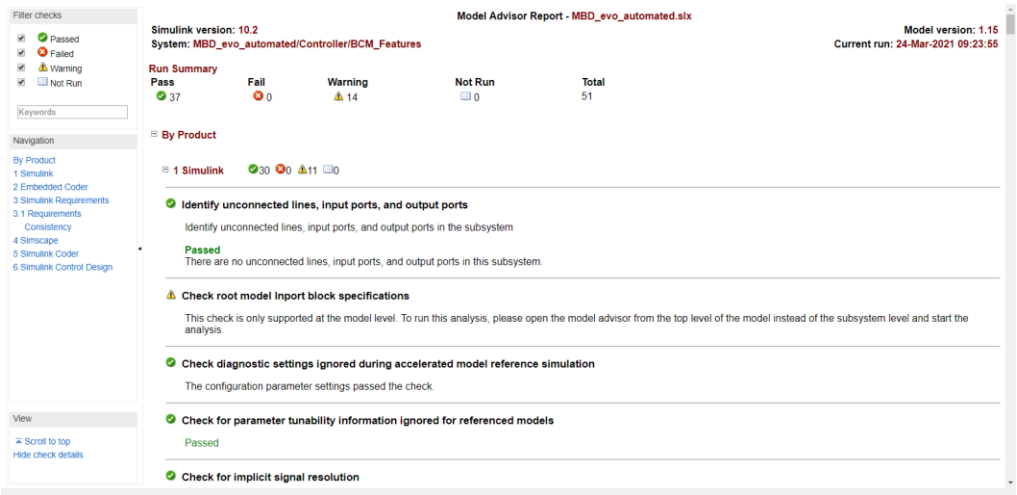


Figure 56 Model advisor for whole model

MCDC Report

Model Hierarchy/Complexity	Run 1							
		Decision		Condition		MCDC		Execution
1. MBD_evo_automated	68	100%		100%		100%		100%
2. . . . Controller	67	100%		100%		100%		100%
3. . . . . BCM_Engine_condition1	1	100%		NA		NA		100%
4. . . . . BCM_Engine_condition2	1	100%		NA		NA		100%
5. . . . . BCM_Engine_condition3	1	100%		NA		NA		100%
6. . . . . BCM_Features	64	100%		100%		100%		100%
7. . . . . BCM_Cruise_Control	14	100%		100%		100%		NA
8. . . . . BCM_CC_Conditions	14	100%		100%		100%		NA
9. . . . . SF: Controller/BCM_Features/BCM_Cruise_Control/BCM_CC_Conditions	13	100%		100%		100%		NA
10. . . . . SF: power_on	7	100%		100%		100%		NA
11. . . . . BCM_Exterior_Lighting	4	100%		100%		100%		100%
12. . . . . BCM_FT_Front_lights		NA		100%		100%		100%
13. . . . . Compare To Constant1		NA		100%		NA		100%
14. . . . . Compare To Constant2		NA		100%		NA		100%
15. . . . . Compare To Constant3		NA		100%		NA		100%
16. . . . . BCM_FT_Turn_Lights	2	100%		100%		100%		100%
17. . . . . Compare To Constant5		NA		100%		NA		100%
18. . . . . Compare To Constant7		NA		100%		NA		100%
19. . . . . BCM_Power_Window	7	100%		NA		NA		100%
20. . . . . BCM_PW_Master_Control_Switch	3	100%		NA		NA		100%
21. . . . . BCM_PW_Windows	4	100%		NA		NA		100%
22. . . . . BCM_Seat_Belt	27	100%		100%		100%		100%
23. . . . . BCM_Gear_Control	17	100%		NA		NA		100%
24. . . . . Gear_Control	16	100%		NA		NA		NA
25. . . . . SF: Controller/BCM_Features/BCM_Seat_Belt/BCM_Gear_Control/Gear_Control	15	100%		NA		NA		NA
26. . . . . BCM_Seat_Belt_Reminder	10	100%		100%		100%		NA
27. . . . . BCM_SBR_Conditions	10	100%		100%		100%		NA
28. . . . . SF: Controller/BCM_Features/BCM_Seat_Belt/BCM_Seat_Belt_Reminder/BCM_SBR_Conditions	9	100%		100%		100%		NA
29. . . . . SF: SeatReminderOn	4	100%		NA		NA		NA
30. . . . . BCM_TPMS	12	100%		100%		NA		100%
31. . . . . Compare To Constant		NA		100%		NA		100%
32. . . . . Compare To Constant1		NA		100%		NA		100%
33. . . . . Compare To Constant2		NA		100%		NA		100%
34. . . . . Compare To Constant3		NA		100%		NA		100%
35. . . . . Enabled Subsystem_1	2	100%		NA		NA		NA
36. . . . . Enabled Subsystem_2	2	100%		NA		NA		NA
37. . . . . Enabled Subsystem_3	2	100%		NA		NA		NA
38. . . . . Enabled Subsystem_4	2	100%		NA		NA		NA

Figure 57 MCDC report

Automated Output Result

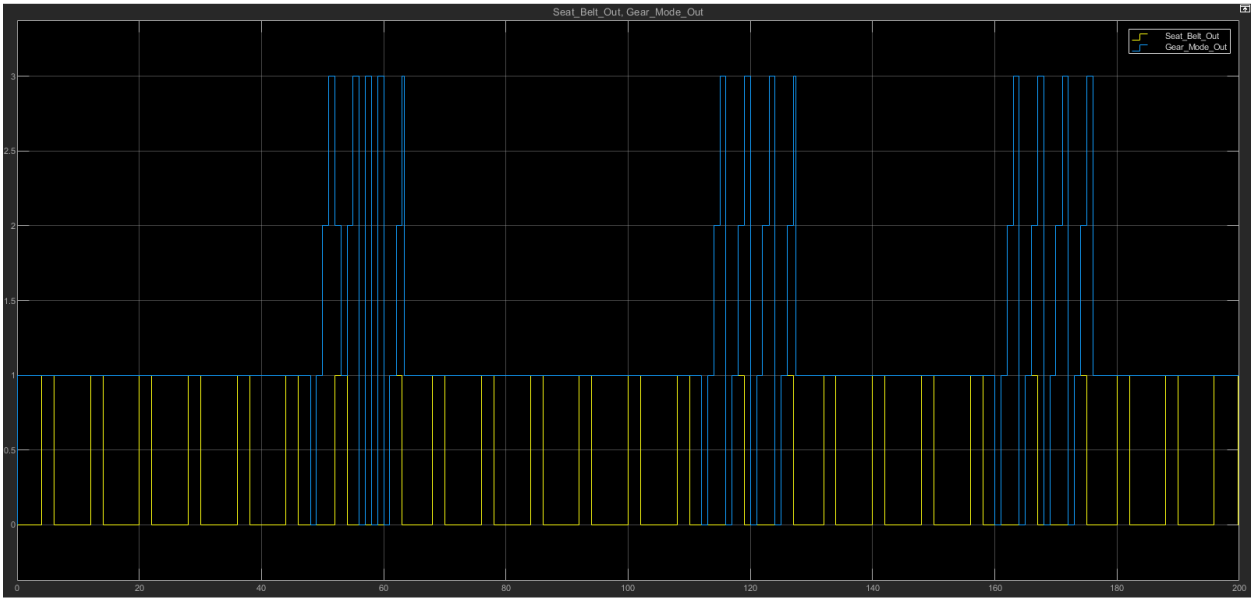


Figure 58 Automated Output for SBR and Gear

Individual Contribution & Highlights

Seat Belt Reminder Implementation and automation of model.

Summary

- Applications of MAAB rules
- MCDC coverage report
- Automated HIL model
- Research on BCM features and creation of requirement datasheet
- Model Advisor report for designed MATLAB model
- Algorithm scripting- Sports Tracker
- Industry standard documentation

Challenges faced and how were they overcome

Challenges	Outcomes
MCDC Coverage report	Debugging errors
Datasheet creation	Thorough research
Automation of model	Simulink

Canoe and CAPL scripting [Team]

Module

Canoe and CAPL Scripting for Launch Control.

Topics and Subtopics

1. RPM
2. Gear Shift
3. Turbo Boost
4. Throttle
5. Launch Control mode
6. Break control

Objectives & Requirements

To implement launch control of McLaren 745LT, using Canoe and CAPL scripting.

Design

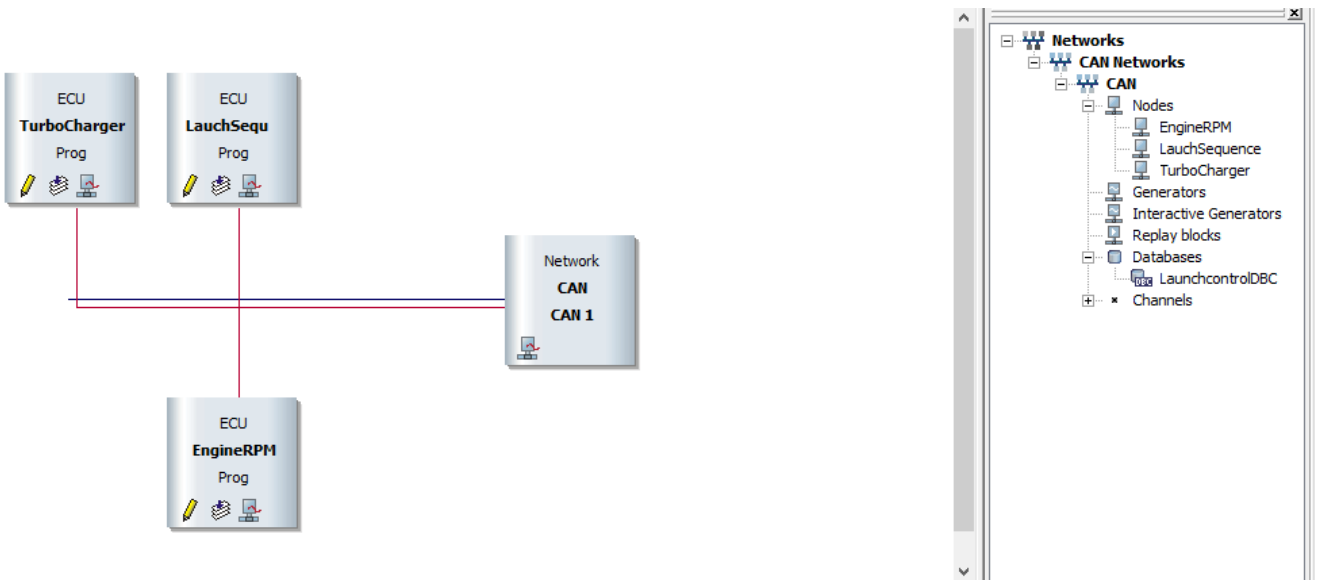


Figure 59 Dashboard Canoe

Figure 60 Speedometer

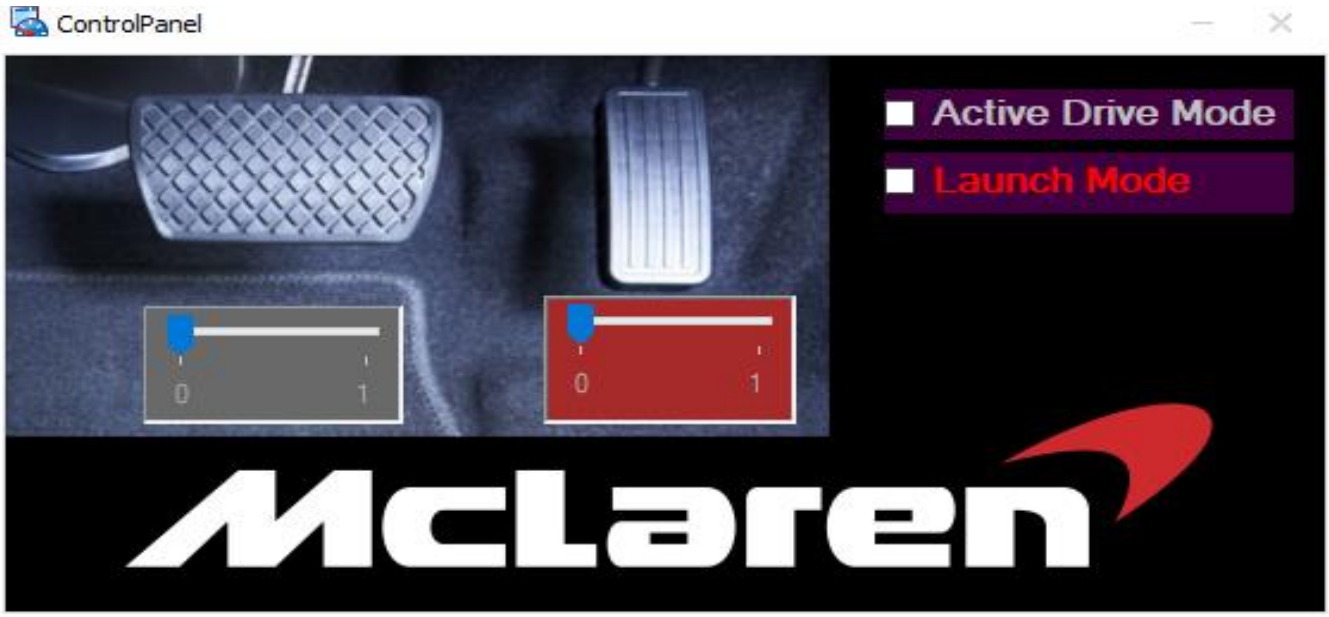


Figure 61 Mode switch, Break and Throttle control

Test Plan

Test ID	Description	Expected IN	Expected OUT	Actual Out	Test Type
LC_1	Lunch Control	1.Apply brake 2.Turn on Active Mode 3.Turn on Launch Control 4.Apply full Accelerator 5. Release	Launch Control Activated	Lunch Control Activated	Requirement Based

		Brake			
LC_2	Lunch Control	1.Apply brake 2.Active Mode OFF 3.Apply full Accelerator 4. Release Brake	Launch Control cannot be Activated	Launch control is not Activated	Scenario Based
LC_3	Launch Control	1.Turn on Active Mode 2.Turn on Launch Control 3.Apply full Accelerator	Launch Control cannot be Activated	Launch control is not Activated	Scenario Based
NM_01	Normal Mode Drive	Apply Accelerator for speeding up	Vehicle Speed increase	Vehicle Speed is increasing	Requirement Based
NM_02	Stopping	Apply Brake pedal	Vehicle speed decreases	Vehicle Speed is decreasing	Requirement Based

Implementation Summary

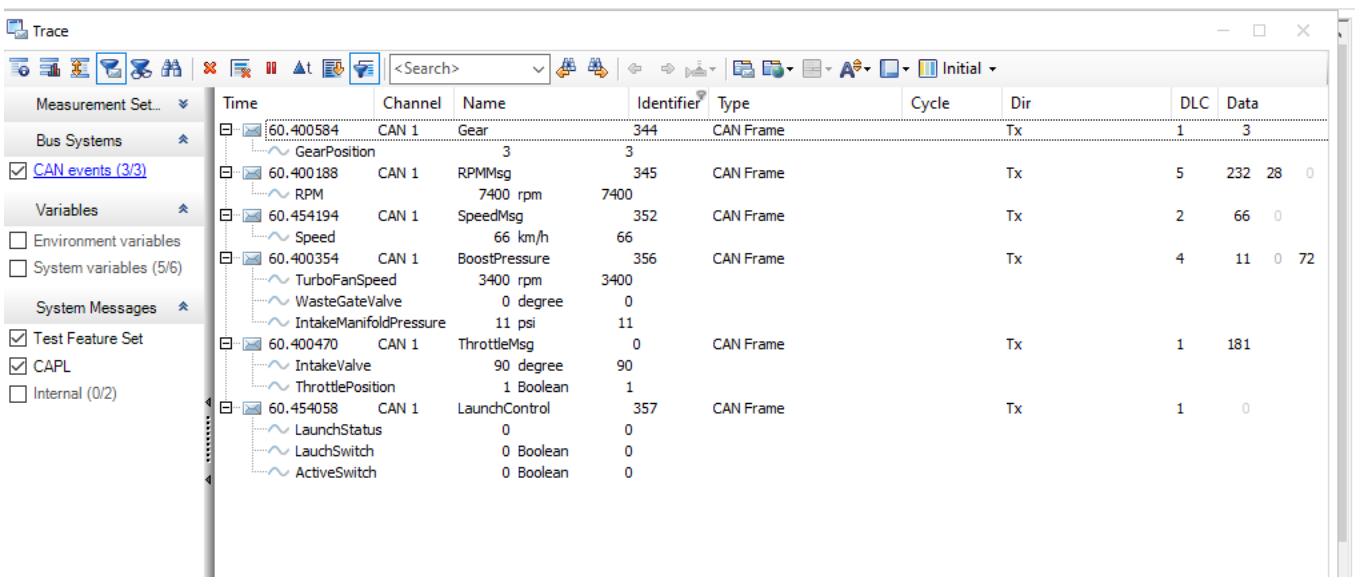


Figure 62 Trace Canoe

SharePoint Link

[Submission/Submission/BANGALORE/2002BLRGT04/99003745]

SharePoint Dashboard

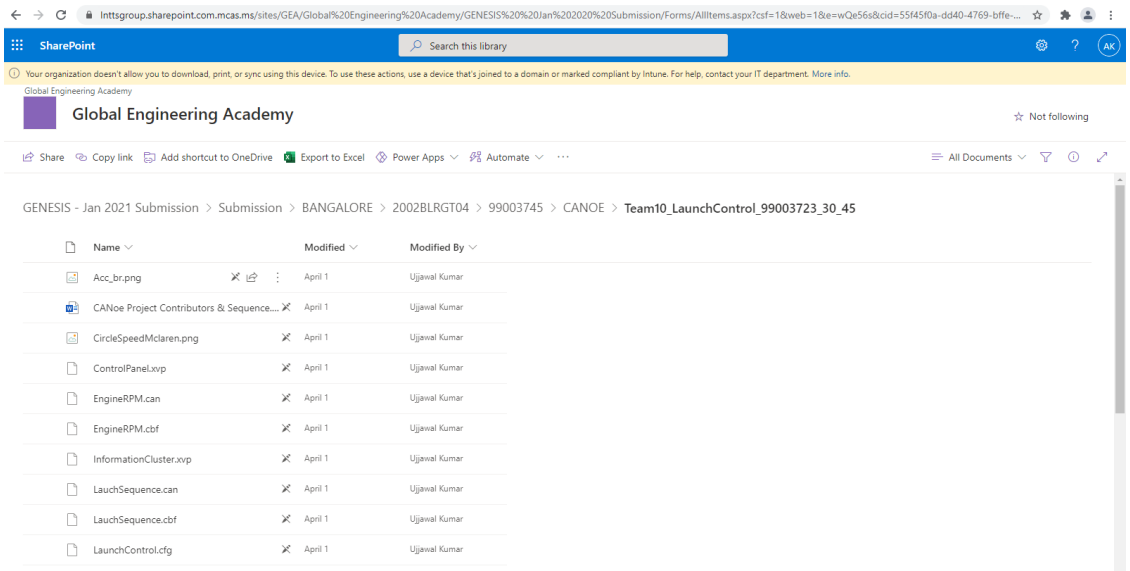


Figure 63 SharePoint Dashboard

Individual Contribution & Highlights

1. Code contribution for Throttle Position Control
2. Break Control
3. Mode Controller
4. Creating database and system variables.

Summary

Outcomes:

1. C language and scripting
2. CAN protocols
3. Logging, creation of database and system variables.
4. Tools- Canoe

Challenges faced and how were they overcome

S. No.	Challenges	Resolve
1.	Canoe Tool	Learning and tutorial videos
2.	System Integration	Changing part code and communication between nodes.

Control Systems [Team & Individual]

Module

Design of controllers for individual and group systems in MATLAB, Simulink and another similar open source software. Analyzing the system based on its speed, accuracy and stability.

Topics and Subtopics

1. System Plant and controller for drone control and longitudinal control of vehicle.
2. Controller design according to the requirement and order of the system.

Objectives & Requirements

The longitudinal dynamics of a vehicle is designed by referring the Newton's law of motion. As per the consideration of Newton's law of motion the force being exerted to the vehicle is the summation of different individual forces such as Viscous force, Aerodynamic force, braking force and Climbing force or the Downgrade force. This force is also called as gravitational force. These forces play a role of deciding factor for calculating the force exerted in the vehicle to move the vehicle or for the motion of the vehicle.

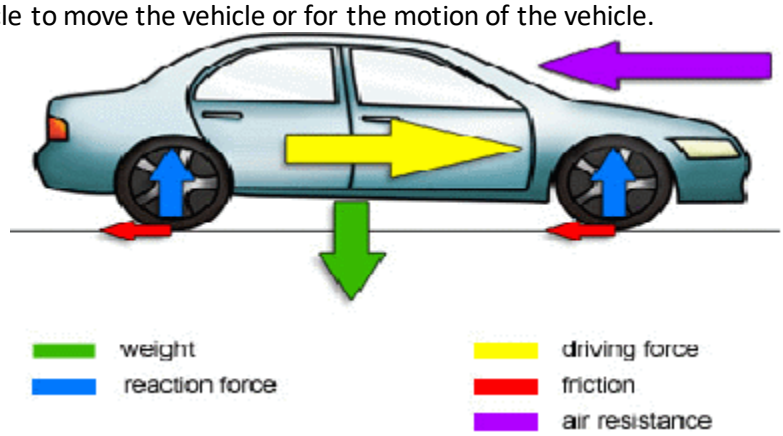


Figure 64 Longitudinal Control

Longitudinal dynamics of the vehicle as governed by the newton's law

$$F_d = M\frac{dv}{dt} + F_a + F_g$$

Where $M\frac{dv}{dt}$ is inertia force.
F_a is aerodynamic drag.
F_d is aerodynamic drag.
v_w is the wind gust speed.
M is the mass of vehicle and passenger.
θ is the road grade.
Ca is aerodynamic drag coefficient.

The state Variables

$$v' = \frac{1}{M}(F_d - C_a v^2)$$

$$F_d' = \frac{1}{T}C_1 u(t-T) - F_d$$

$$y = v$$

Differentiating the above equation

$$\frac{d}{dt}v' = \frac{1}{M}(-2C_a v\delta v + \delta F_d)$$
$$\frac{d}{dt}F_d' = \frac{1}{T}C_1\delta u(t-T) - F_d)$$
$$y = \delta v$$

The transfer function of the above equation is

$$\frac{\Delta v(s)}{\Delta u(s)} = \frac{\frac{C_1}{MT}}{(s+2C_a v/M)(s+\frac{1}{T})(s+\frac{1}{\tau})}$$

The parameters chosen are:
C_1 = 743; T=1s; τ =1s; M=1500kg; C_a = 1.19 N/(m/s^2); g =9.8 m/s^2.

Longitudinal control is used in cruise control system.

Individual system is for drone pitch control

A second order differential equation describing the rotational dynamics for the platform pitch angle can be written as:

θ'' = m/Jy

where m is the pitch moment of the wing, which is given by the following expression:
m =1/2(((V^2) ρ*S*c*Cm)
Cm = Cmac + Cmα*α + Cmδe*δe + Cmq*Q*(c/V)
Assuming that in steady hover flight θ = α and Cmac = 0 and substituting both m & Cm, we get

θ'' =ρ*(V^2) *S*c*2*Jy[Cmαθ + Cmδe*δe + Cmq*(c/V)dθ/dt]

Cmα = −0.145 and Cmδe = 0.65 Cmq = −10 α = 0

$$\frac{\theta(s)}{\delta_e(s)} = \frac{85}{s^2+40s+18}$$

```
%The derivative Cmα
% represents the variation of the pitching moment with respect to
% the angle-of-attack αThe derivative Cmα
% represents the variation of the pitching moment with respect to
% the angle-of-attack α.
% S 0.18m2 Wing Reference area
% c 0.3m Wing chord
% ρ 1.225kg/m3 Air density
% V 10m/s Wind velocity (airflow)
% Jy 0.0254kg.m y-axis moment of inertia
```

Requirements

1. Longitudinal control for vehicle's plant and controller design.

2. Pitch control for drone controller design and analysis.

Design for Longitudinal Control

Transfer function:

$$\frac{\Delta v(s)}{\Delta u(s)} = \frac{\frac{C_1}{MT}}{(s + 2C_a v/M)(s + \frac{1}{T})(s + \frac{1}{\tau})}$$

Roots of the equation are:
Pole 1: $-2C_a v/M$
Pole 2: -1/T
Pole 3: -1/τ
Order of the equation: Third order equation.

Plant Model with P, PI, PD And PID controller

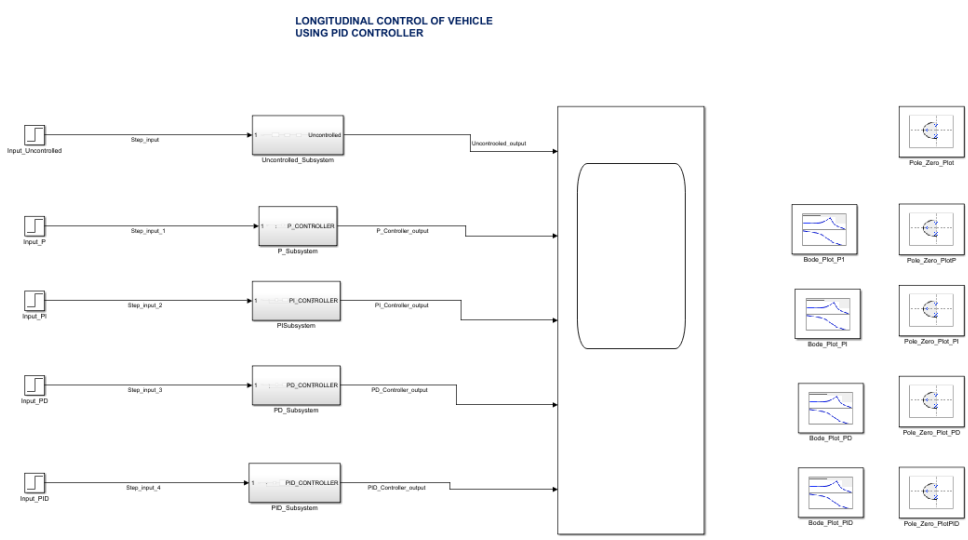


Figure 65 Plant Model PID

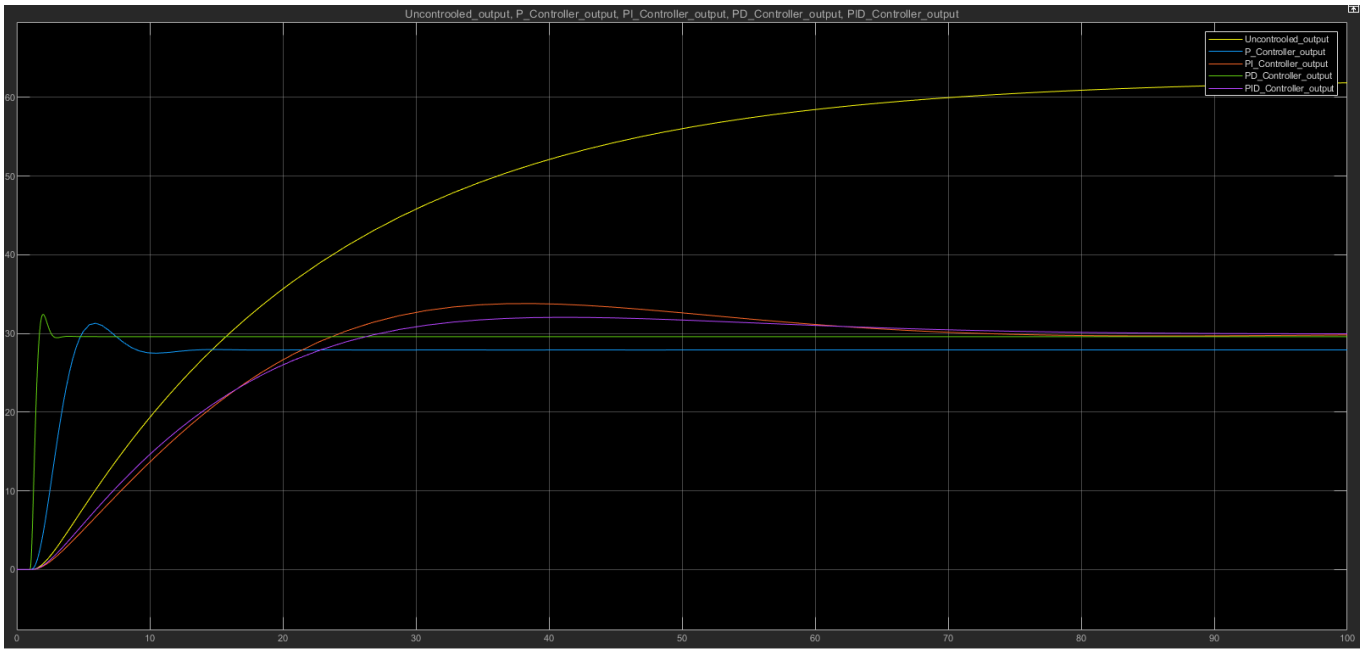


Figure 66 PID scope output

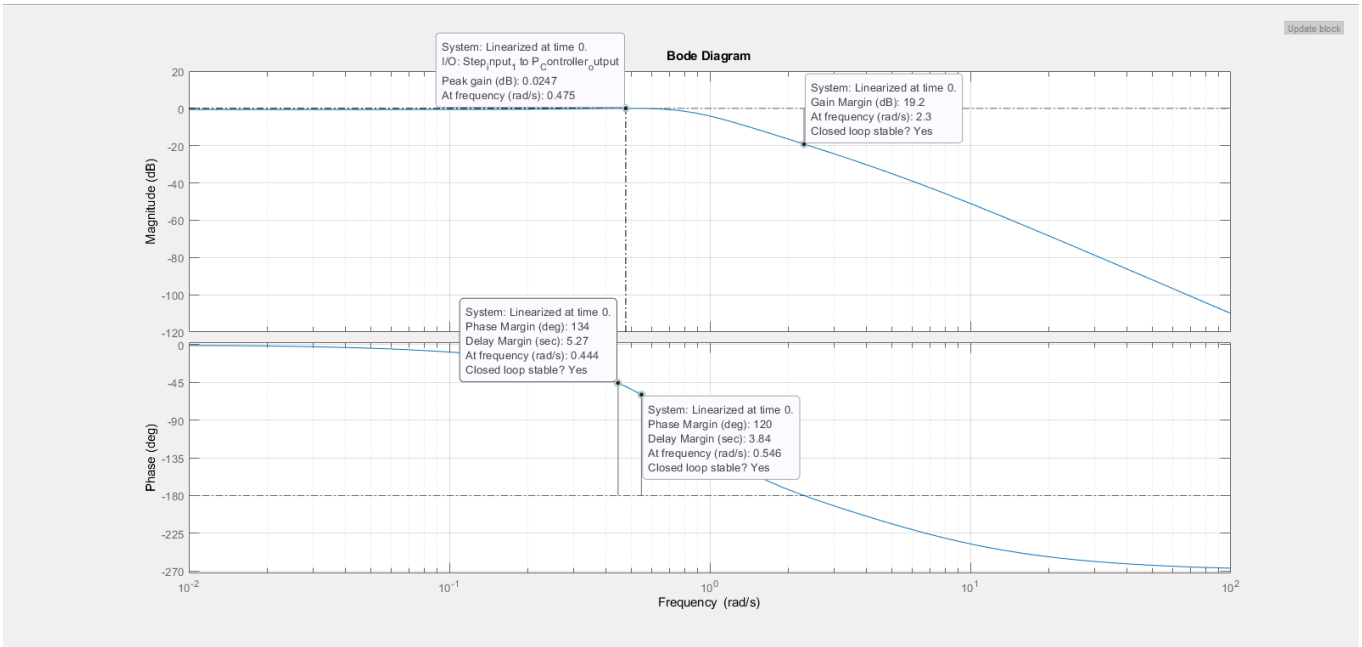


Figure 67 Bode Plot P

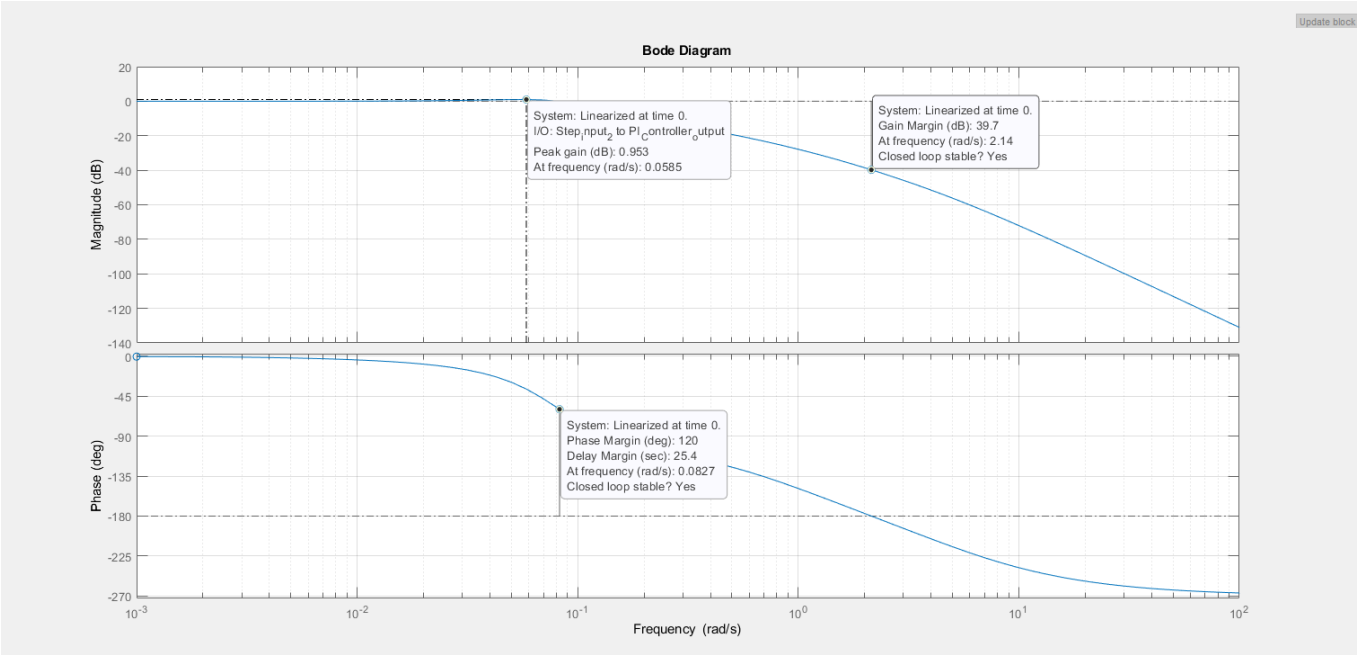


Figure 68 Bode Plot PI

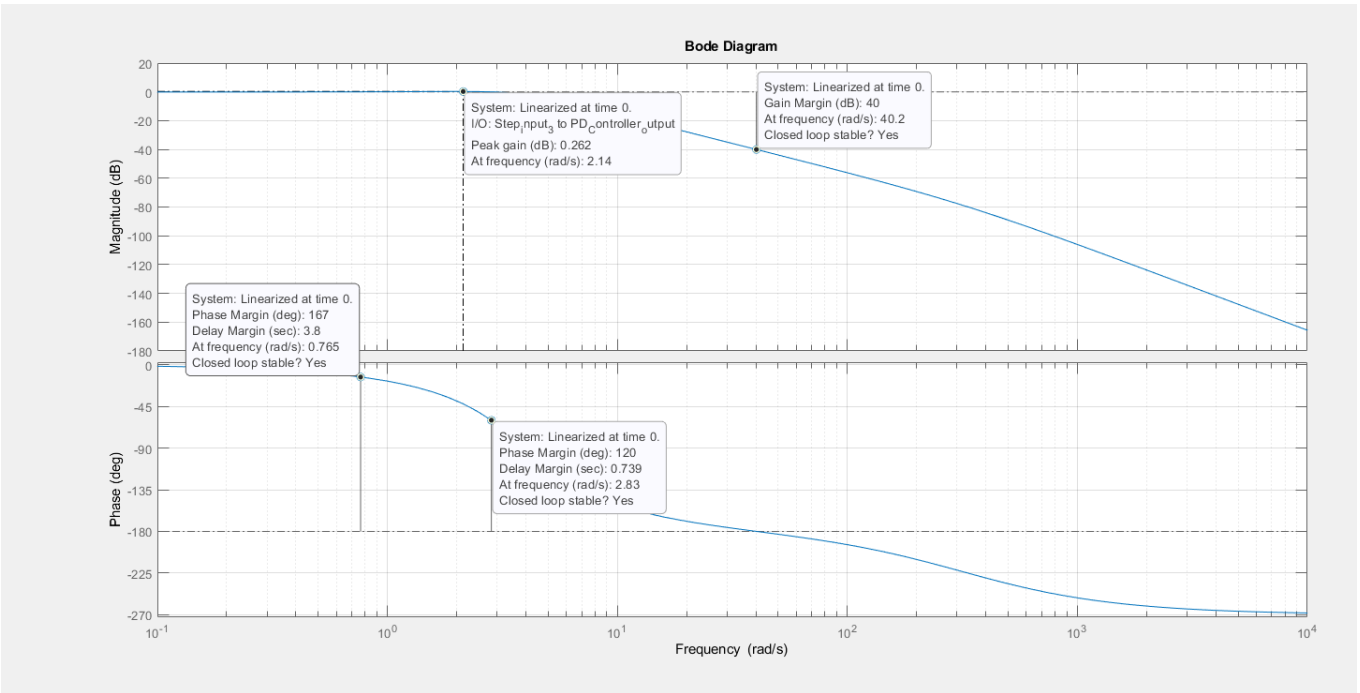


Figure 69 Bode Plot PD

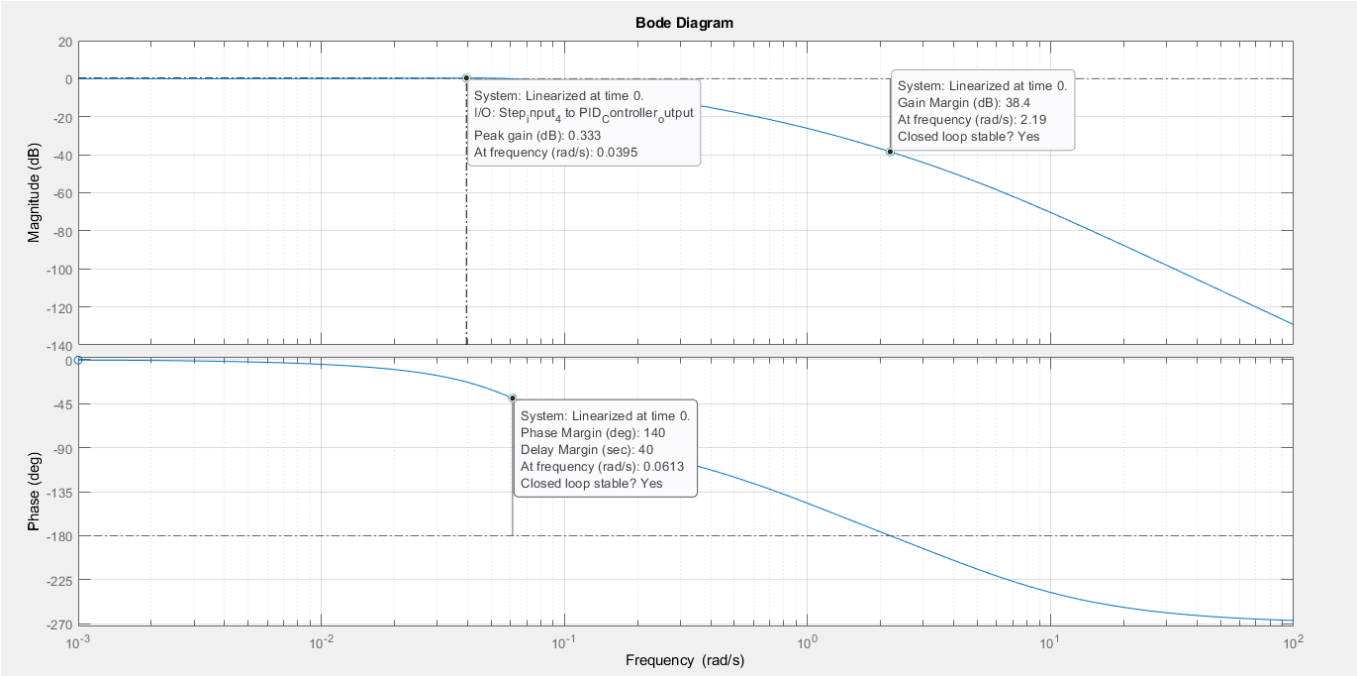


Figure 70 Bode Plot PID

Analysis

- P Controller
  Rise Time- 2.189s
  OS- 11.798%
  Settling time- 7.4s

- PI
  Rise Time- 16.417s
  OS- 13.068%
  Settling time-63.5s

- PD
  Rise Time- 4.62197s
  OS- 9.341%
  Settling time- 1.52s

- PID
  Rise Time- 17.780s
  OS- 6.989%
  Settling time- 66.2s

- Accuracy- PD controller is the accurate followed by P>PI>PID as setting for it is least among the others.

- Speed- PD>P>PI PID
  (As rise time is least for PD).

- Overshoot is maximum in PI>P>PD>PID
  Therefore, stability of PID is maximum followed by PD>P>PI.

- Poles- Poles are located on the left side of the s-plane in PID, making the system stable.

Plant Model with lead lag compensator

Lead Lag Controller for Longitudinal control of Vehicle

Team 3 Vipul Sakunia 99003723 Archita Modi 99003724 Ujjawal Kumar 99003745 Ravikumar M Pise 99003747

TRANSFER FUNCTION: 72
TOOL ANALYSIS 72
ANALYSIS FOR LEAD COMPENSATOR ..... **ERROR! BOOKMARK NOT DEFINED.**

Transfer Function:

[(C1/MT)/(s+2Cav) (s+1/T) (s+1/tau)]

Tool Analysis

```
C1=743;
T=1;
tau =0.2;
M=1500;
Ca=1.19;
v=30;
s=tf('s')

sys=tf([C1/M*T],[1,2*Ca*v/M]);
sys2=tf([1],[1,1/T]);
sys3=tf([1],[1,1/tau]);
sysf=sys2*sys3*sys
figure(1)
step(sysf)
title("Main TF Step response")

Gc=(s+.03)/(s+.04);
Gc1=(s+1)/(s+10);
lead=sysf*Gc*Gc1
p=pole(sysf)
z=zero(sysf)

figure(2)
pzmap(sysf)
title("Pole zero map of TF")
stepinfo(sysf)
```

```
Gcloseloop=lead/(1+lead)
p1=pole(lead)
z1=zero(lead)
figure(3)
opt = stepDataOptions('InputOffset',0,'StepAmplitude',230);
step(Gcloseloop,opt)
title("Longitudinal Lead Lag Response")
stepinfo(Gcloseloop)

figure(4)
pzmap(Gcloseloop)
title("Pole Zero Lead-Lag feedback")

% bode plots
figure(5);
subplot(2,2,1);
bode(sysf)
[Gm,Pm,Wcg,Wcp] = margin(sysf)
title('Bode of the OL TF');

subplot(2,2,2);
bode(Gcloseloop)
[Gm,Pm,Wcg,Wcp] = margin(Gcloseloop)
title('Bode of the CL TF');

% subplot(2,2,3);
% bode(lead)
% [Gm,Pm,Wcg,Wcp] = margin(lead)
% title('Bode of the LEAD TF');
```

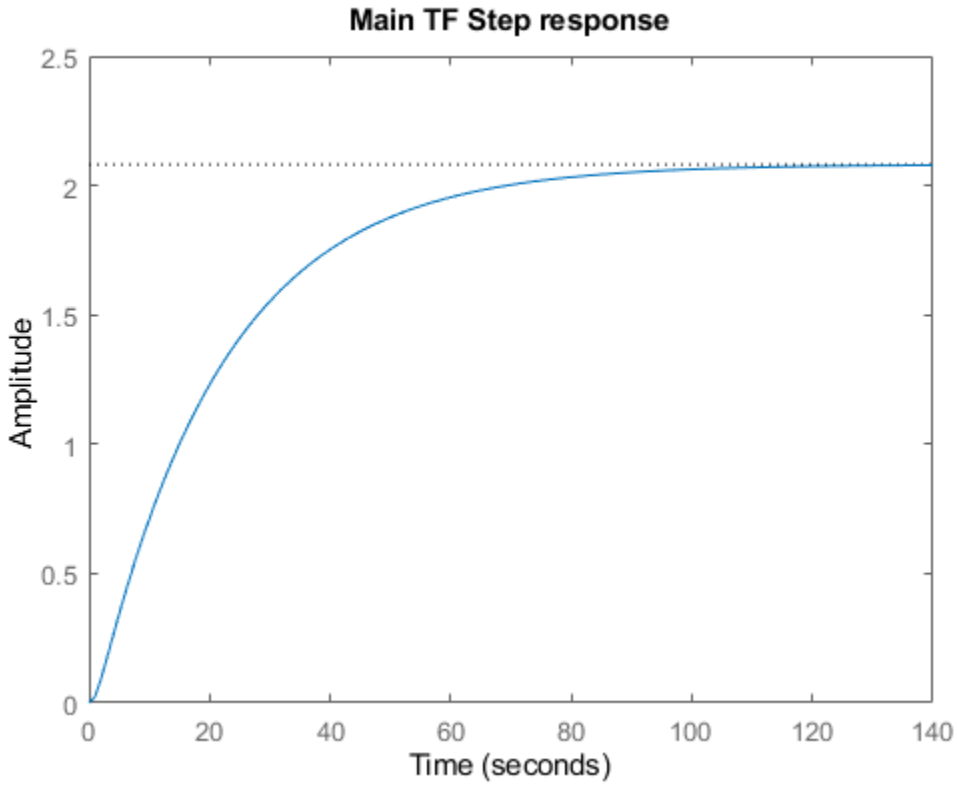


Figure 71 Main TF response

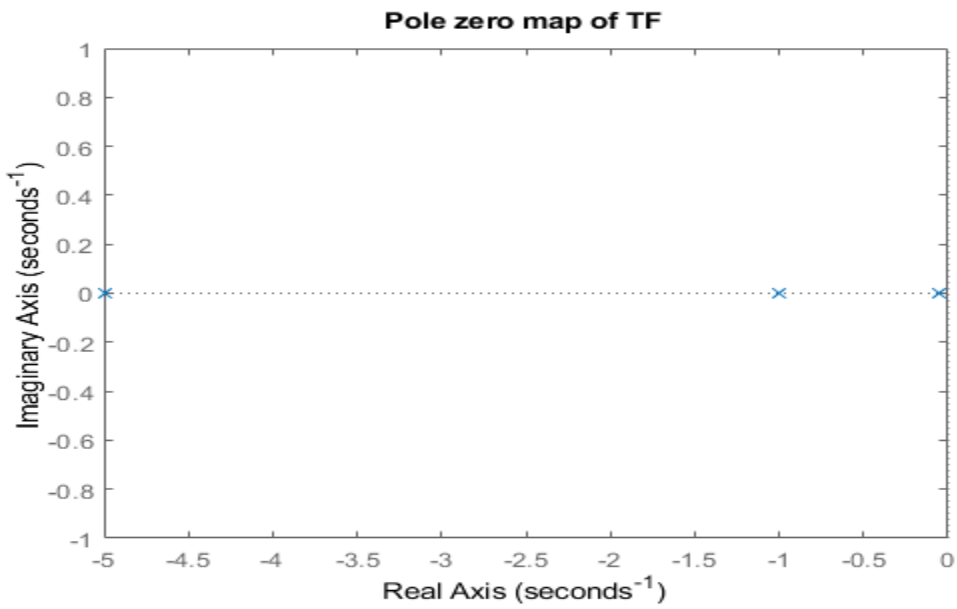


Figure 72 Pole Zero Map of main TF

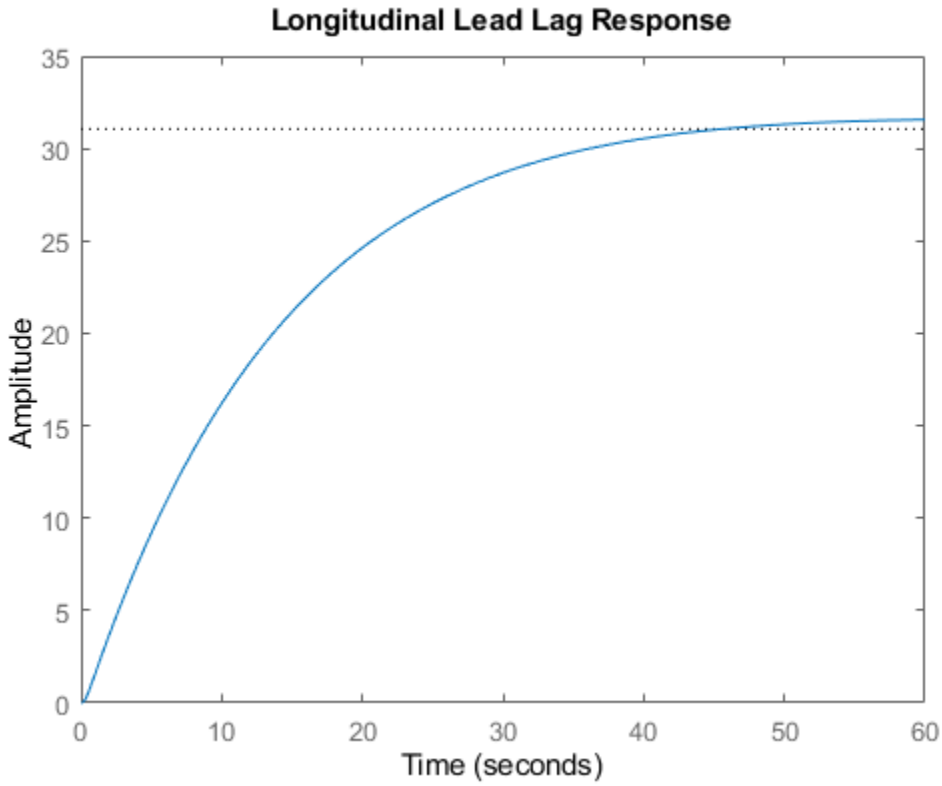


Figure 73 Response with Lead-Lag Controller

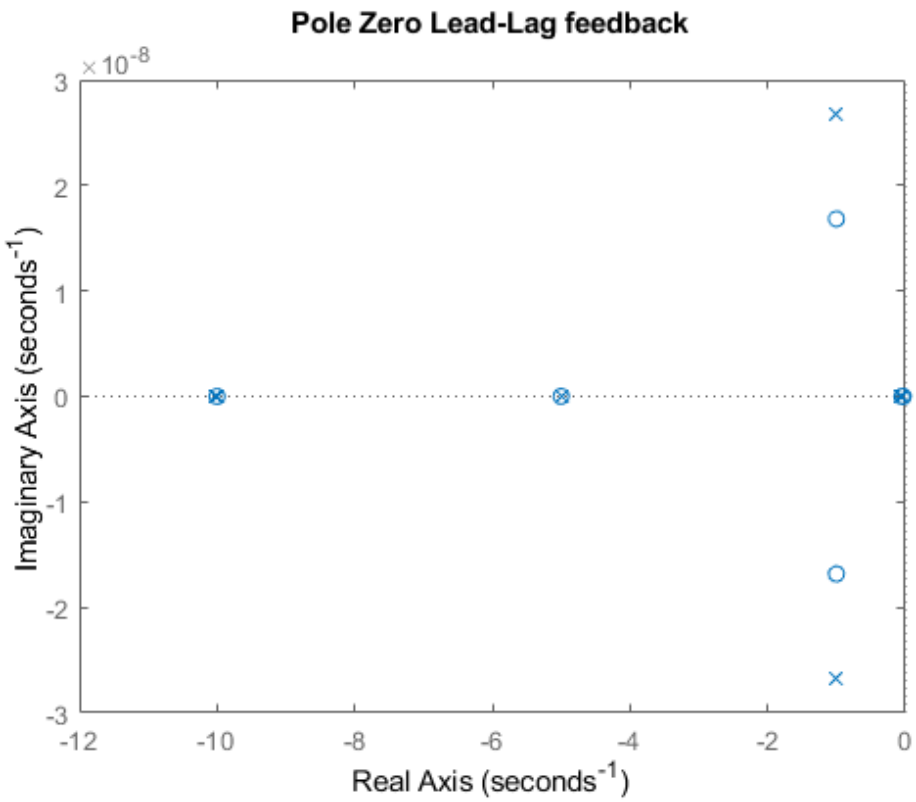


Figure 74 Lead Lag Pole Zero Map

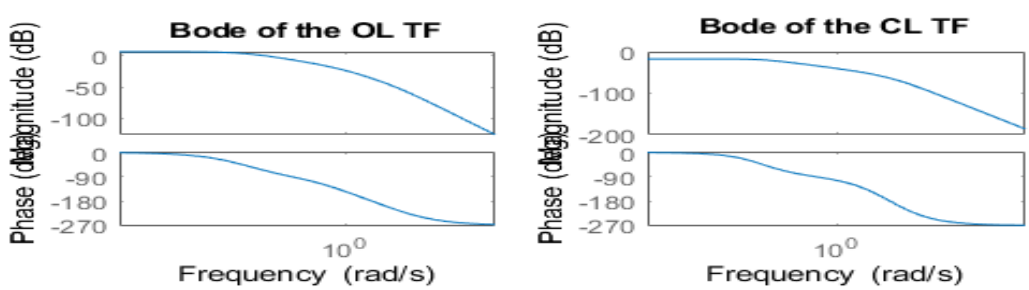


Figure 75 Bode Plots

Analysis for Lead Compensator

Risetime: 25.8096 Settling Time: 39.1411 Settling in: 0.1215 Settling Max: 0.1374 Overshoot: 1.7407 Undershoot: 0 Peak: 0.1374 Peak Time: 66.5908

Without Compensator-- Risetime: 46.2140 Settling Time: 83.4147 Settling Min: 1.8800 Settling Max: 2.0798 Overshoot: 0 Undershoot: 0 Peak: 2.0798 Peak Time: 153.8282

Speed-- speed of response is more in lead lag compensator Accuracy-- Lead Compensator provides higher accuracy Stability of system without compensator is high because overshoot is zero. The system is stable in both the cases from the frequency analysis as the gain margin, phase margin and the crossover frequency are all positive.

Plant Model with Fuzzy Controller

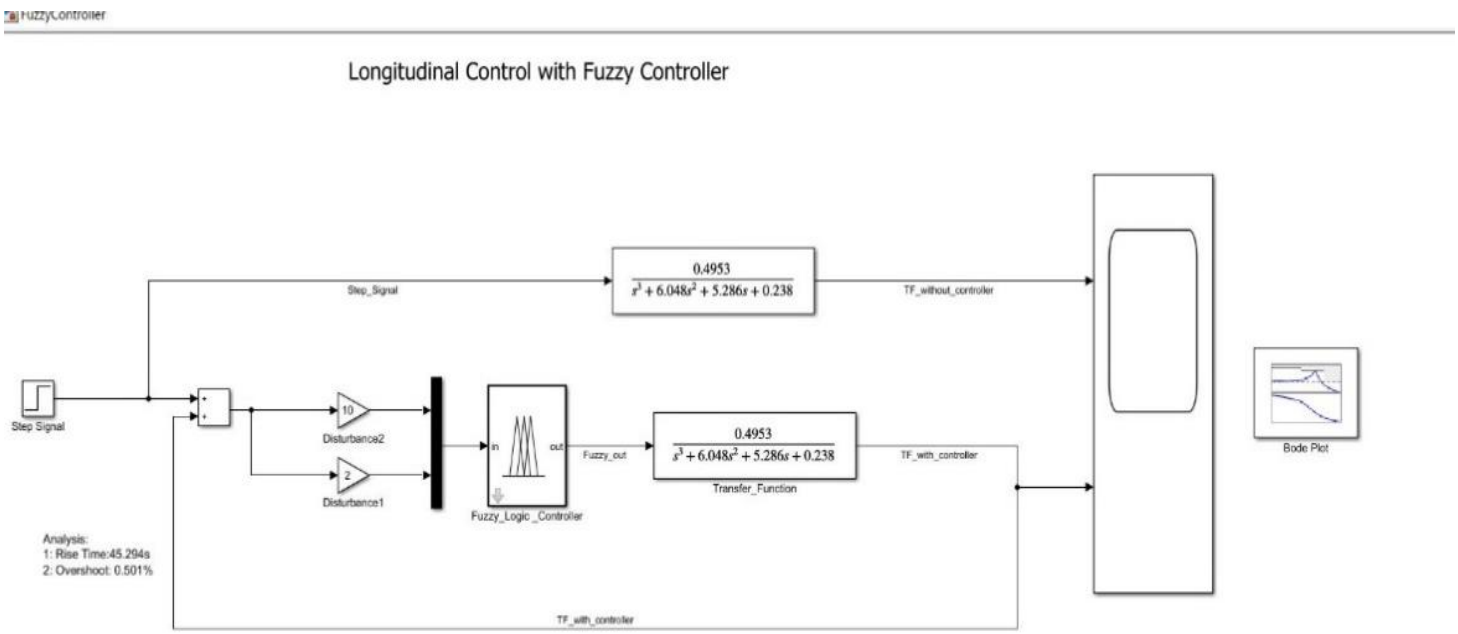


Figure 76 Fuzzy Logic Implementation

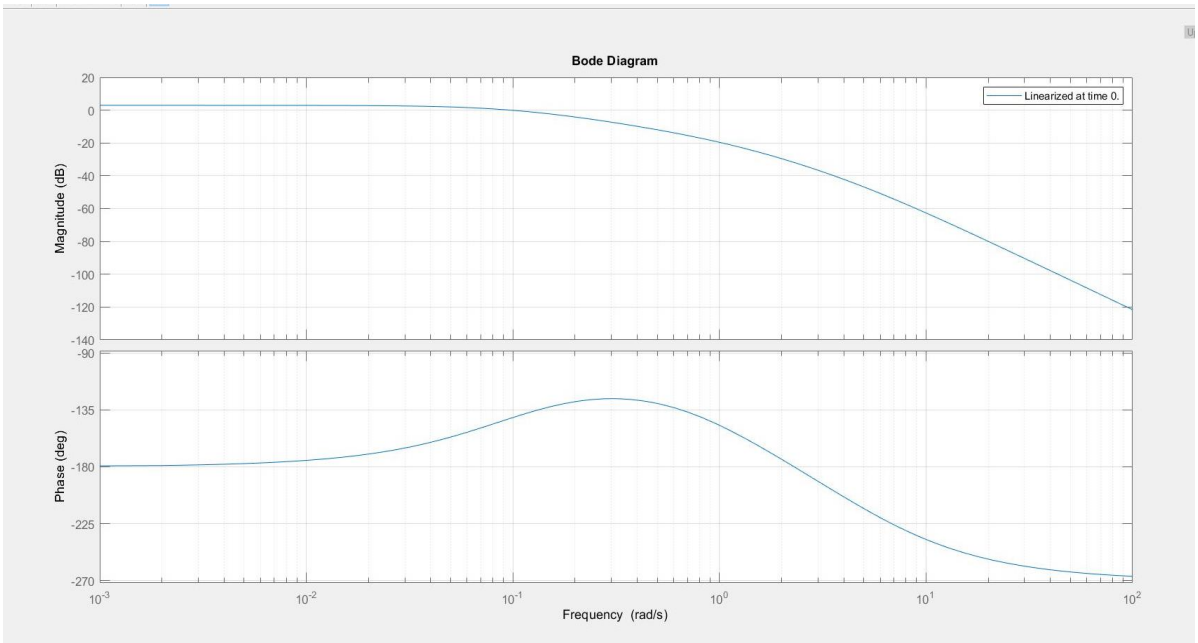


Figure 77 Fuzzy Bode Plot

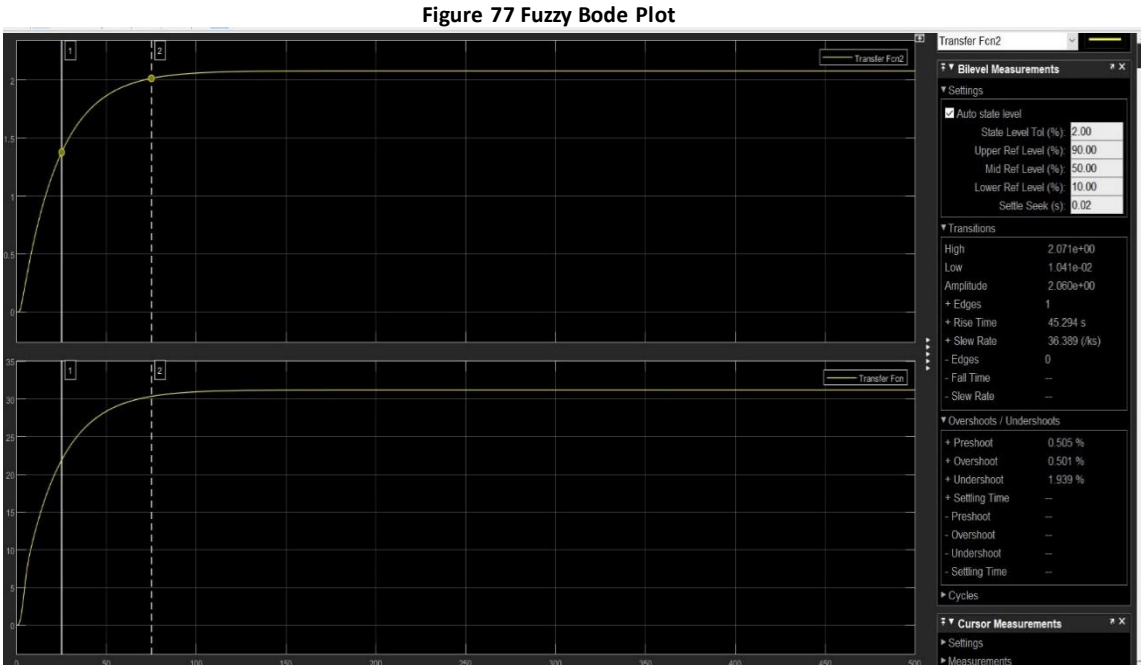


Figure 78 Fuzzy Output

- Rise Time: 45.294s
- Overshoot: 0.501%

Longitudinal Control with Disturbance using PI, PD, PID Control.

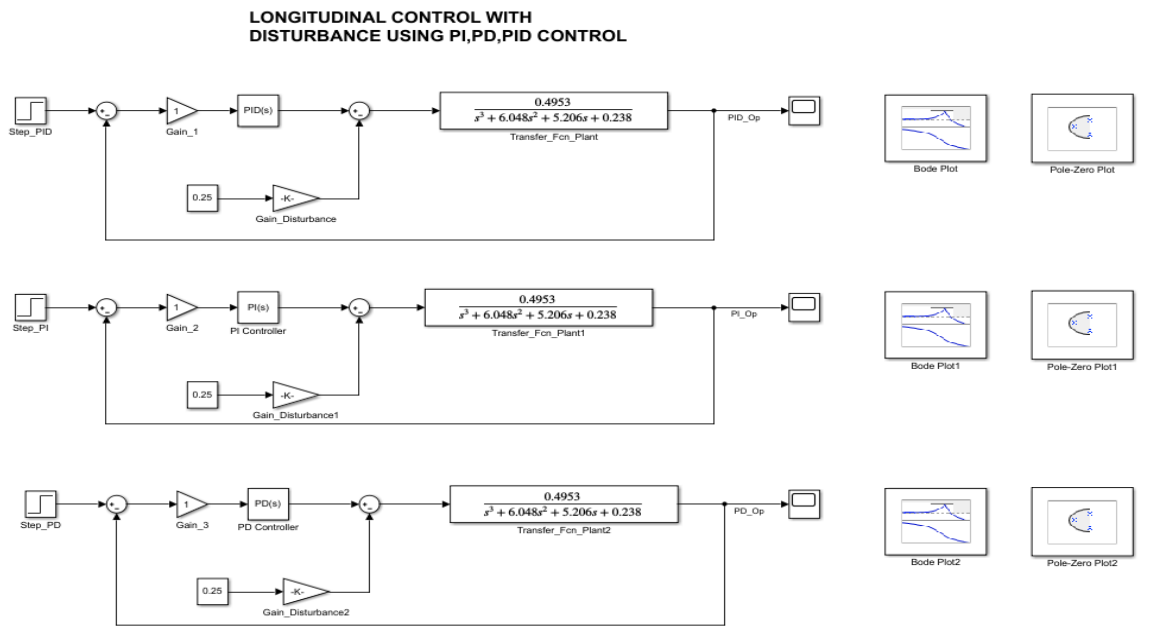


Figure 79 PID with Disturbance

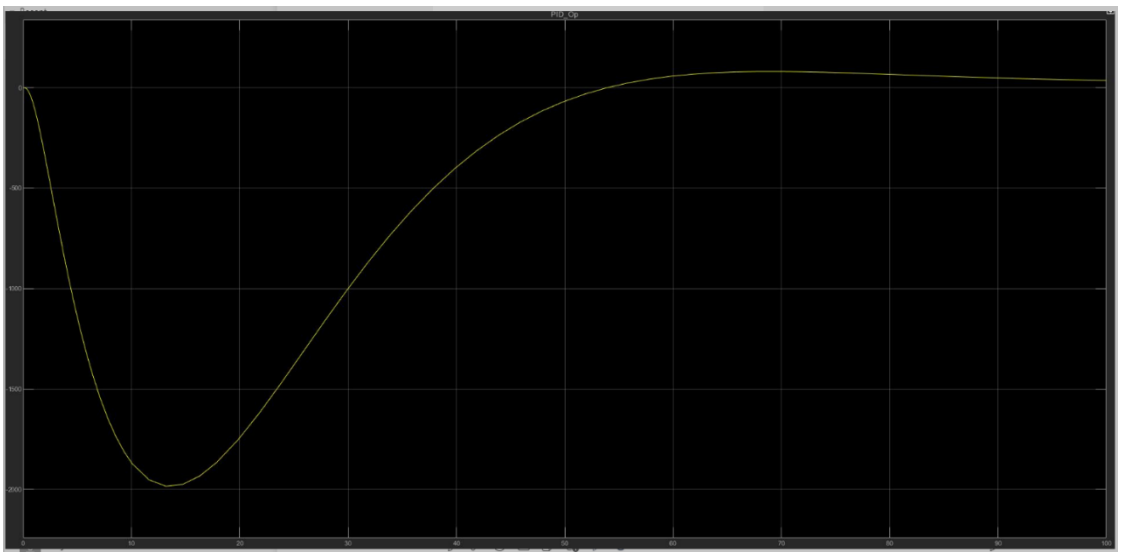


Figure 80 Disturbance with PI Output

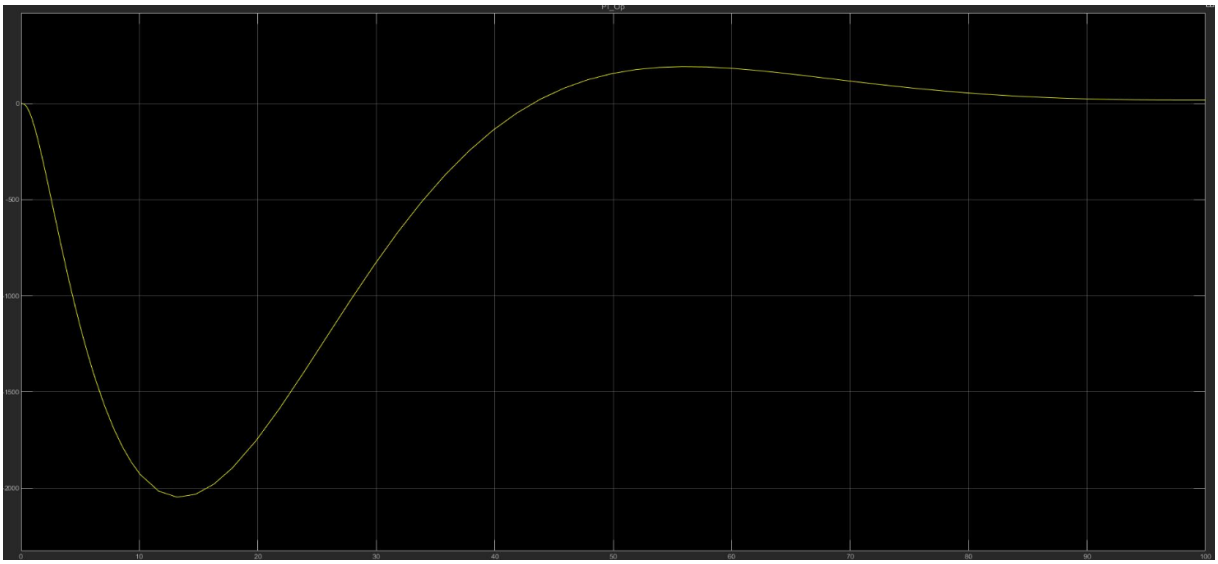


Figure 81 Disturbance with PD Output

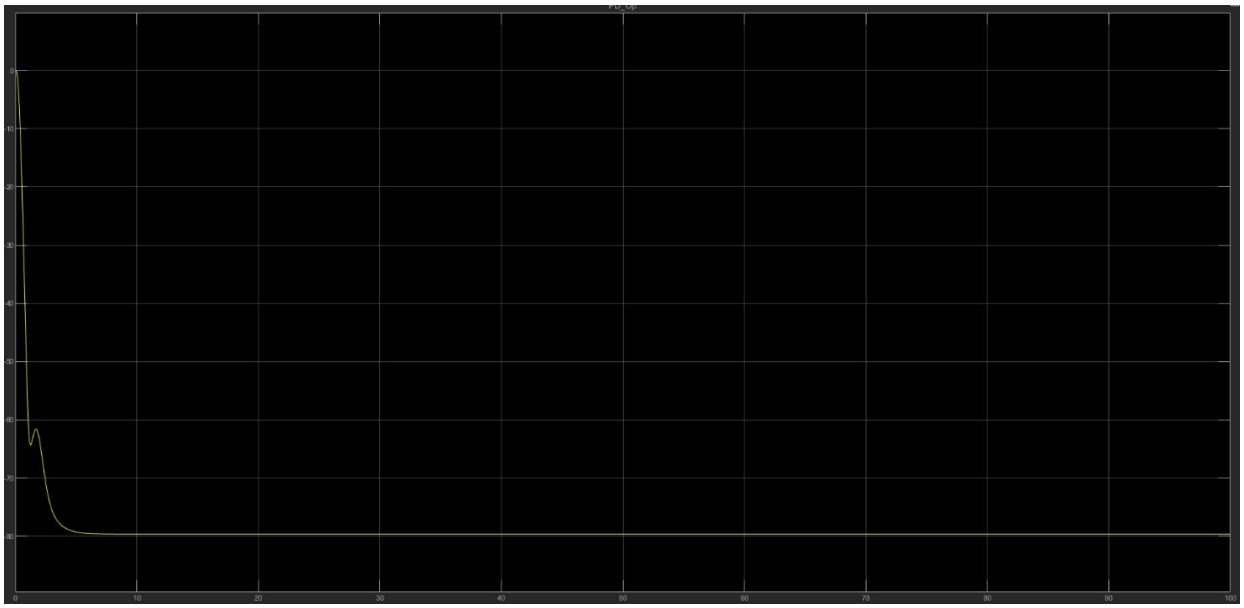


Figure 82 Disturbance with PID Output

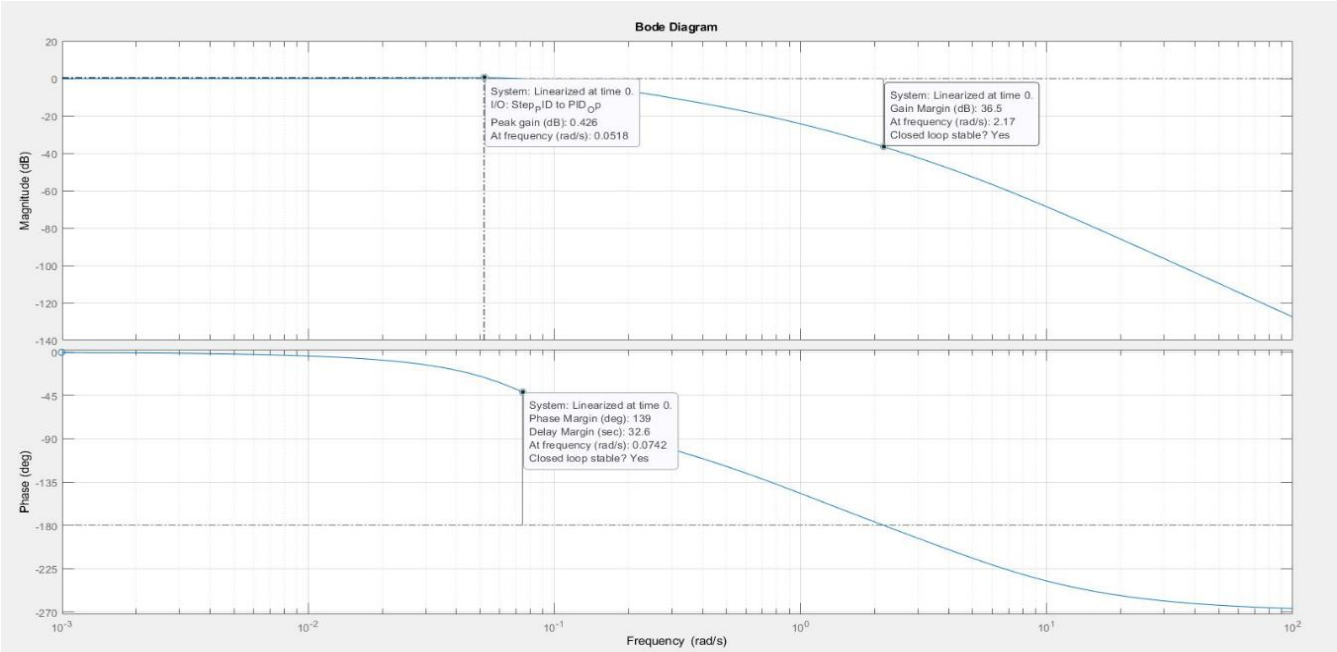


Figure 83 Bode Plot PI

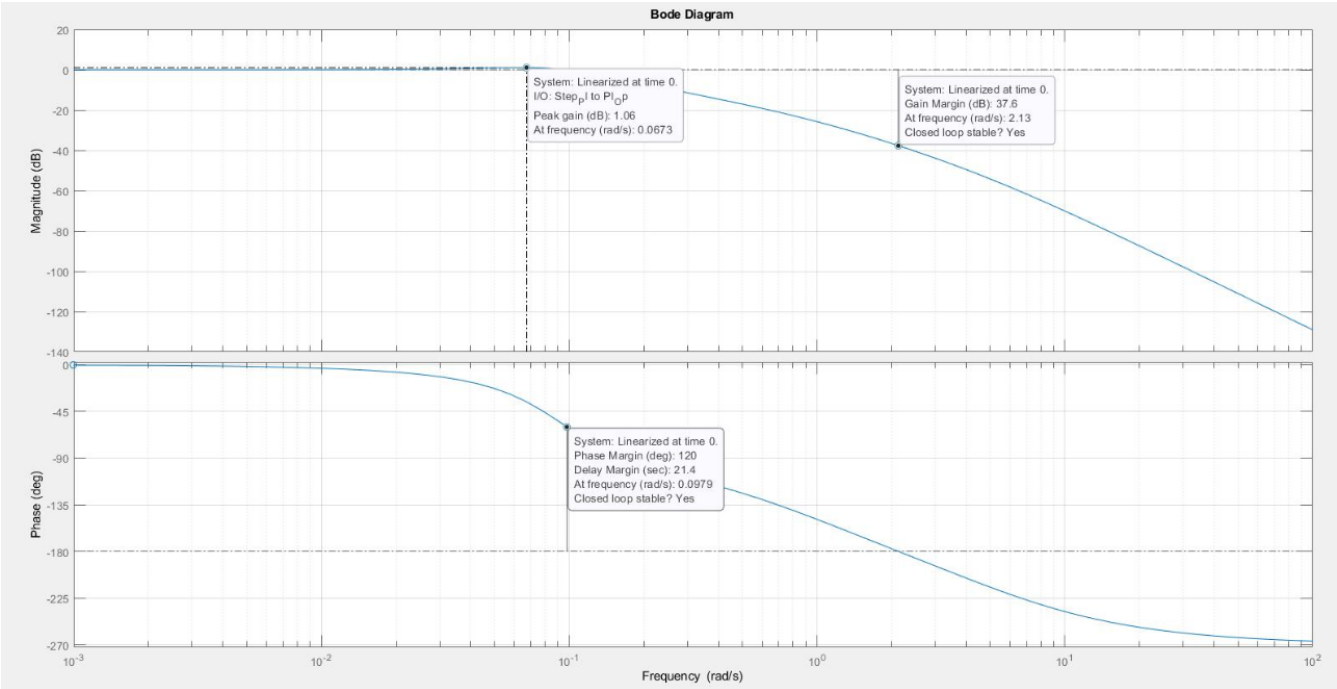


Figure 84 Bode Plot PD

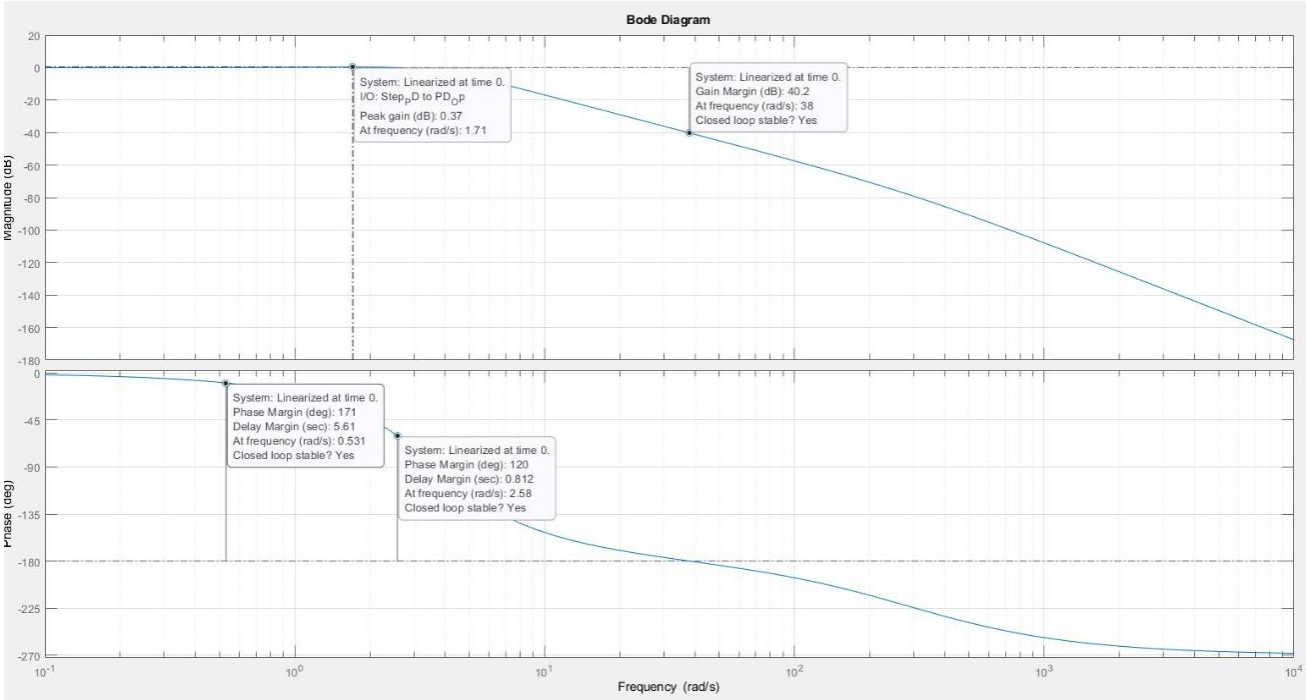


Figure 85 Bode Plot PID

Analysis

1. PID Controller
   Kp=0.8212
   Ki=0.070275 Kd-0.1688
   Rise time: 14.9 s
   Settling time: 58.6s Overshoot: 7.39%
2. PI Controller -0.70851
   Ki=0.0811
   Rise time: 13.7s
   Settling time: 55.3s Overshoot: 13.2%
3. PD Controller
   Kp=0.501
   Kd= 28.5792 Rise time: 13.7s
   Settling time: 55.3s
   Overshoot: 13.2%
4. Speed-PD-PI-PID
   System with PD, PI Controllers. speed is High because Rise time is less.
5. Accuracy=PD>=PI>PID
   System with PD, PI Controllers, Accuracy is more because Setting time is less.
   Stability PID>PD>PI

SharePoint Link

https://lnttsgroup.sharepoint.com/Submission/Forms/Longitudinal

Individual Contribution & Highlights

Lead-lag controller for Longitudinal control of vehicle.

Pitch Control for Drone

Math Analysis

roots= −22.7 ± 91.45i and −4.62

Pitch control of drone with controllers

TOOL ANALYSIS 83
ANALYSIS AND TIME RESPONSE RESULT 87
POLE MAPPING ..... ERROR! BOOKMARK NOT DEFINED.

Tool Analysis

```
numerator = 85;
denominator = [1,40,18];

sys_ol = tf(numerator,denominator);
sys_cl = feedback(sys_ol,1);

[GC_PI,info_PI] = pidtune(sys_ol,'PI');
sys_cl_PI = feedback(sys_ol * GC_PI,1);

[GC_PD,info_PD] = pidtune(sys_ol,'PD');
sys_cl_PD = feedback(sys_ol * GC_PD,1);

[GC_PID,info_PID] = pidtune(sys_ol,'PID');
sys_cl_PID = feedback(sys_ol * GC_PID,1);

% input response plots
figure(1);
subplot(2,2,1);
step(sys_ol);
title('Step of OL TF');

subplot(2,2,2);
step(sys_cl);
title('Step of CL TF');

% controller plots
figure(2);
subplot(2,2,1);
step(sys_cl)
title('CL Uncontrolled response of TF');
stepinfo(sys_cl)
```

```
% [zeta,wn]=damp(sys_cl)

subplot(2,2,2);
step(sys_cl_PI)
title('CL PI controlled response of TF');
stepinfo(sys_cl_PI)
[zeta,wn]=damp(sys_cl_PI)


subplot(2,2,3);
step(sys_cl_PD)
title('CL PD controlled response of TF');
stepinfo(sys_cl_PD)


subplot(2,2,4);
step(sys_cl_PID)
title('CL PID controlled response of TF');
stepinfo(sys_cl_PID)


% Bode plots
figure(3);
subplot(2,2,1);
bode(sys_ol)
title('OL TF');

subplot(2,2,2);
bode(sys_cl)
title('CL uncontrolled TF');

subplot(2,2,3);
bode(sys_cl_PID)
title('CL PID controlled TF');
```

```
ans =

  struct with fields:

        RiseTime: 0.7975
    SettlingTime: 1.4422
     SettlingMin: 0.7439
     SettlingMax: 0.8247
       Overshoot: 0
      Undershoot: 0
            Peak: 0.8247
        PeakTime: 2.6469


ans =
```

```
  struct with fields:

        RiseTime: 1.5488
    SettlingTime: 5.8888
     SettlingMin: 0.9142
     SettlingMax: 1.1421
       Overshoot: 14.2109
      Undershoot: 0
            Peak: 1.1421
        PeakTime: 3.3865


zeta =

    0.8811
    0.8811
   38.9665


wn =

    0.5865
    0.5865
    1.0000


ans =

  struct with fields:

        RiseTime: 0.0395
    SettlingTime: 0.1282
     SettlingMin: 0.9026
     SettlingMax: 1.0999
       Overshoot: 11.0251
      Undershoot: 0
            Peak: 1.0999
        PeakTime: 0.0812


ans =

  struct with fields:

        RiseTime: 1.7539
    SettlingTime: 6.3719
     SettlingMin: 0.9078
     SettlingMax: 1.0617
       Overshoot: 6.1683
      Undershoot: 0
```

```
      Peak: 1.0617
  PeakTime: 3.8941
```

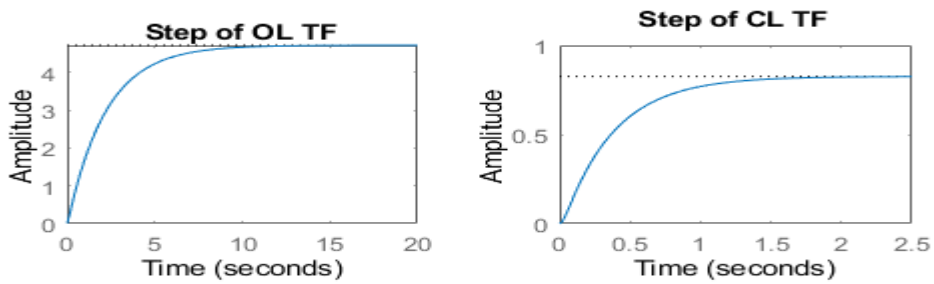


Figure 86 Step Response Drone

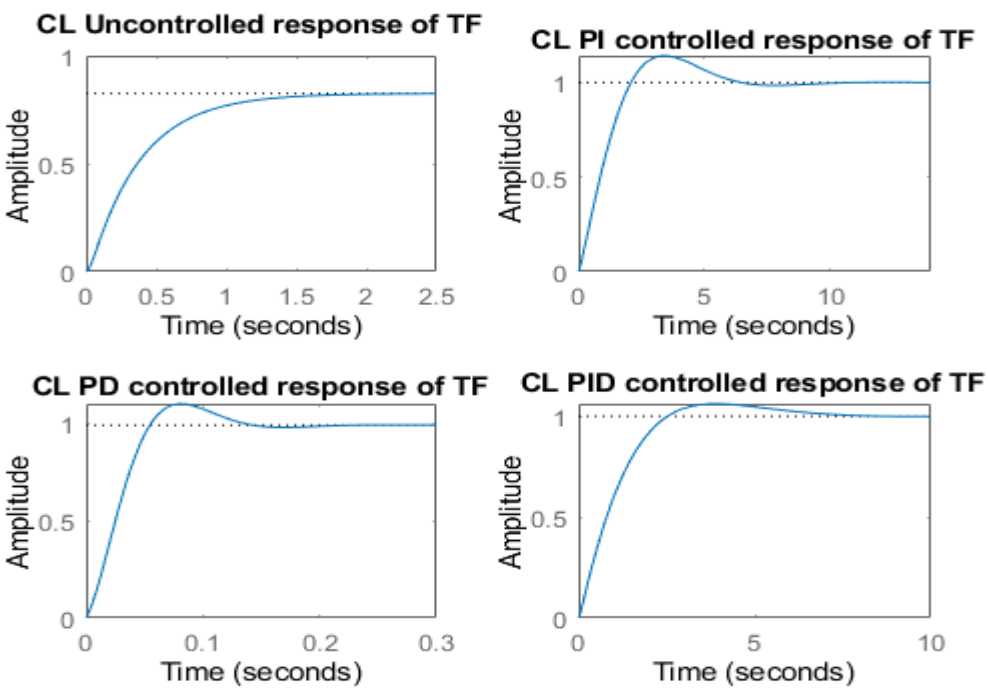


Figure 87 Response with Controllers

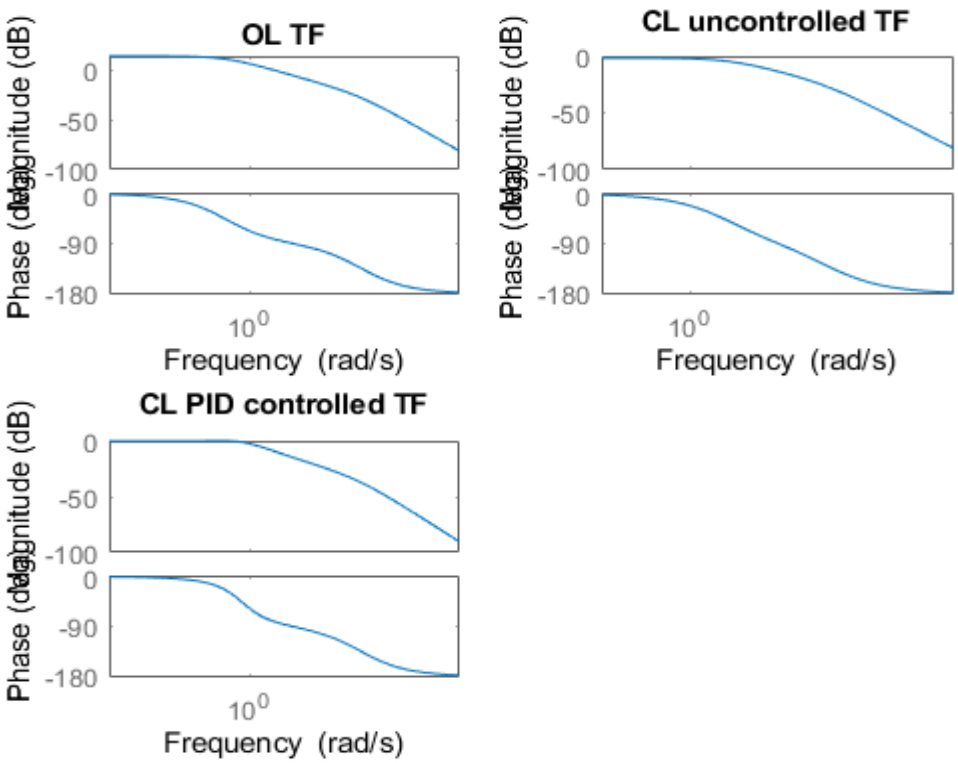


Figure 88 Bode Plots for Drone PID

Analysis and Time Response result

Rising time is increased so the speed is decreased. settling time is increased, accuracy decreased. Overshoot increased Peak is amplified of the system. Pole at origin is added. Eliminates the steady state error.

Rising time is decreased, speed of the system is increased. settling time is decreased, accuracy increased to a large extent.
Overshoot increased showing less stability from closed loop. Peak is amplified of the system but peak time decreased.
Zero at origin is added No affect to steady state error. Does not change the stability of the system.
Rising time is increased, speed of the system is decreased. settling time is increased, accuracy decreased.
Overshoot increased. Peak is amplified of the system.
The phase margin of PID controlled system is +ve Gain margin is infinite which make the system stable Can be cross checked by the roots and pole of the system as they lie on the negative x axis. Affects all the parameters of the system. Adds a pole & a zero at the origin. the steady state error of the system is reduced because of I controller.

Overall Analysis Stability- CL>PID>PD>PI Speed- PD>CL>PI>PID Accuracy- PD>CL>PI>PID

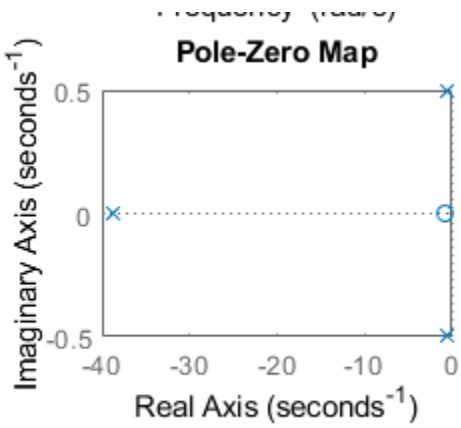


Figure 89 Pole Zero Map

SharePoint Link

https://lnttsgroup.sharepoint.com/sites/GEA/GlobalSubmission/Forms/DronePControl

Implementation Summary

Pitch control for drone by second order equation in open and closed loop in MATLAB script.

Summary

Outcomes:

1. Plant and controller design and control systems.
2. Migration of models and scripts in various open source software.
3. Analysis of system's speed, accuracy and stability.
4. Tools- OpenModelica, GNU Octave.

Individual Contribution & Highlights

1. Longitudinal Control for vehicle with lead-lag compensator.
2. Research for various controllers that can be used for the chosen plant.
3. Drone pitch control.
4. Migration of model and script in GNU octave and open modelica.

Challenges faced and how were they overcome

Challenges	Resolution
Equation and system knowledge	By going through various research papers
Scripting in MATLAB	Through examples and help documentation.