./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1. | 16-04-2021 | Darpan Patil |  |  |  |
| 2 | 17-04-2021 | Darpan Patil |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69361300)

[Miniproject -1 [Team] 4](#_Toc69361301)

[Modules used 4](#_Toc69361302)

[Topic and Subtopics 4](#_Toc69361303)

[Objectives & Requirements 5](#_Toc69361304)

[Low Level Requirement 6](#_Toc69361305)

[Design 11](#_Toc69361308)

[Test Plan 26](#_Toc69361309)

[Implementation Summary 28](#_Toc69361310)

[Video Summary 28](#_Toc69361311)

[Git Link 28](#_Toc69361312)

[https://github.com/99003738/AppliedSDLC\_N3.git 28](#_Toc69361313)

[Git Dashboard 28](#_Toc69361314)

[Summary 29](#_Toc69361315)

[Summary 35](#_Toc69361316)

[Challenges faced and how were they overcome 35](#_Toc69361317)

[Miniproject -2 PYTHON [Individual] 26](#_Toc69361318)

[Module:- Python 26](#_Toc69361319)

[Topic and Subtopics 26](#_Toc69361320)

[Objectives & Requirements 27](#_Toc69361321)

[Design 27](#_Toc69361322)

[Low Level Design 27](#_Toc69361323)

[Implementation Summary 28](#_Toc69361324)

[Git Link 28](#_Toc69361325)

[Git Dashboard](#_Toc69361326)

[Summarys 28](#_Toc69361327)

[Individual Contribution & Highlights 28](#_Toc69361328)

[Summary 28](#_Toc69361329)

[Challenges faced and how were they overcome 28](#_Toc69361330)

[Miniproject -3 MBSE [TEAM] 29](#_Toc69361318)

[Module:- MBSE 29](#_Toc69361319)

[Objectives & Requirements 29](#_Toc69361321)

[Design 30](#_Toc69361322)

[Low Level Design 30](#_Toc69361323)

[Implementation Summary 31](#_Toc69361324)

[Git Link 31](#_Toc69361325)

[Summarys 32](#_Toc69361327)

[Challenges faced and how were they overcome 40](#_Toc69361330)

[Miniproject -4: Embedded C [Team] 32](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919098)

[Module/s: 33](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919099)

[Requirements: 33](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919100)

[Design: 35](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919101)

[Individual Contribution & Highlights: 36](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919102)

[Summary 37](file:///C:\Users\99003767\Downloads\GENESIS%20-%20%20Mini-project%20Summary%20Report.docx#_Toc67919103)

[Miniproject -5 CANOE AND CAPL SCRIPTING [TEAM] 38](#_Toc69361318)

[Module:- MBSE 38](#_Toc69361319)

[Objectives & Requirements 39](#_Toc69361321)

[Design 40](#_Toc69361322)

[Low Level Design 40](#_Toc69361323)

[Implementation Summary 41](#_Toc69361324)

[Summarys 41](#_Toc69361327)

[Miniproject -6 CONTROL SYSTEMS [TEAM] 42](#_Toc69361318)

[Module:- MBSE 42](#_Toc69361319)

[Objectives & Requirements 42](#_Toc69361321)

[Design 45](#_Toc69361322)

[Low Level Design 46](#_Toc69361323)

[Implementation Summary 52](#_Toc69361324)

[Summarys 53](#_Toc69361327)

**TABLE OF FIGURES**

[Figure 1 :Activity diagram 8](#_Toc69716774)

[Figure 2: Activity Diagram 9](#_Toc69716775)

[Figure 3: Deployment Diagram 10](#_Toc69716776)

[Figure 4 : Sequential Diagram 11](#_Toc69716777)

[Figure 5 : Class diagram 12](#_Toc69716778)

[Figure 6 : Activity diagram 13](#_Toc69716779)

[Figure 7:Use case diagram 14](#_Toc69716780)

[Figure 8: Sequential Diagram 15](#_Toc69716781)

[Figure 9: Composite diagram 16](#_Toc69716782)

[Figure 10 :Class Diagram 17](#_Toc69716783)

[Figure 11:Use case diagram 18](#_Toc69716784)

[Figure 12: State Diagram 19](#_Toc69716785)

[Figure 13:Test plan 20](#_Toc69716786)

[Figure 14:Test plan 21](#_Toc69716787)

[Figure 15:Git Dashboard 22](#_Toc69716788)

[Figure 16:Folder Structure 23](#_Toc69716789)

[Figure 17:Contributors List 23](#_Toc69716790)

[Figure 18: GIT Inspector summary 24](#_Toc69716791)

[Figure 19:Commits 25](#_Toc69716792)

[Figure 20: Build log 26](#_Toc69716793)

[Figure 21:Unit Testing 26](#_Toc69716794)

[Figure 22:Python implementation 30](#_Toc69716795)

[Figure 23:Python Implementation 31](#_Toc69716796)

[Figure 24:MBSE output 34](#_Toc69716797)

[Figure 25:Pin Configuration 35](#_Toc69716798)

[Figure 26: Remote control code 36](#_Toc69716799)

[Figure 27: Motion detection 37](#_Toc69716800)

[Figure 28 37](#_Toc69716801)

[Figure 29 38](#_Toc69716802)

[Figure 30: Mirror and Steering control 39](#_Toc69716803)

[Figure 31: Alcohol detection 40](#_Toc69716804)

[Figure 32: 4 nodes 42](#_Toc69716805)

[Figure 33: before ignition 43](#_Toc69716806)

[Figure 34: ignition ON and Seat belt reminder 43](#_Toc69716807)

[Figure 35: Reminder OFF after seat belt bulckled 44](#_Toc69716808)

[Figure 36: Airbag deployed due to sudden deceleration and crash 44](#_Toc69716809)

# 

# Mini project -1 [Team]

## Modules used

SDLC (Software Development Life Cycle) and C Programming Modules are used in the project.

Project Title

Mini Calculator

### Topic and Subtopics

* Introduction about SDLC
* C Programming
* Git Hub
* Code Analysis

Val grind

CPP check

* Testing
* Unity Testing
* Features of Calculator
* Core Steps of SDLC
* Testing has been done for each function.
* Make file.
* V Model
* Agile Model

## Objectives & Requirements

Objective: -  The simplest calculators can do only addition, subtraction, multiplication, and division. More sophisticated calculators can handle exponential operations, roots, logarithms, trigonometric functions, and hyperbolic functions So we are designing a calculator which contains all the functionalities

Of simplest calculator and sophisticated calculator.

High Level Requirement

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Description |
| HL\_01 | Basic operations | Addition,Subtraction,Multiplication and Division |
| HL\_02 | Trignometry | All trigonometric functions |
| HL\_03 | Conversions | Basic conversions like decimal to octal and hexadecimal. |
| HL\_04 | BMI | Calculate BMI taking height and weight into consideration. |

## Low Level Requirement

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Description |
| LL\_01 | Precision | 2 decimal places |
| LL-02 | Input for trigonometric functions | Should be in radians |
| LL\_03 | Conversion options | Decimal,Octal,Binary and Hexadecimal |
| LL\_04 | BMI | Input for height in feet and weight in kg. |

## UML Diagrams(High level)

Activity Diagram

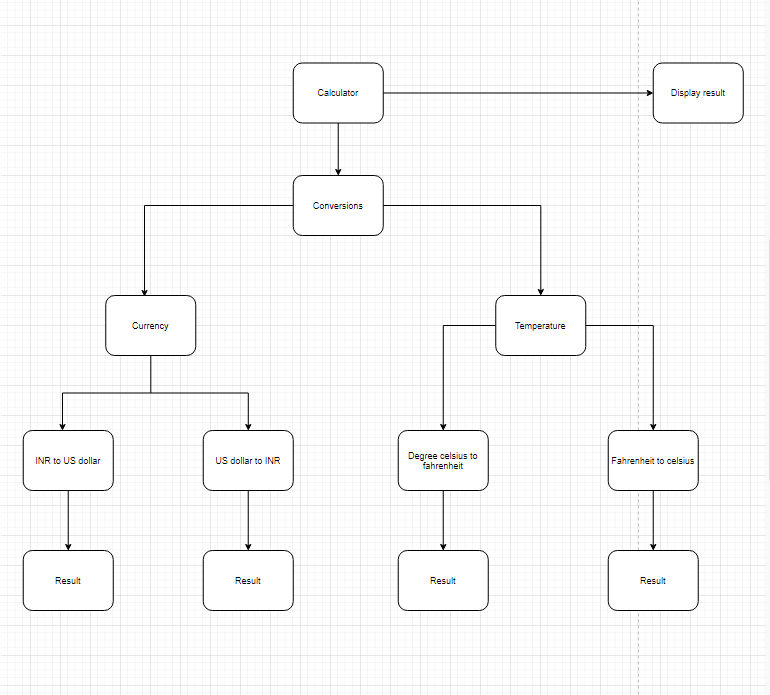


Figure 1 :Activity diagram

## 

Figure 2: Activity Diagram

Deployment diagram

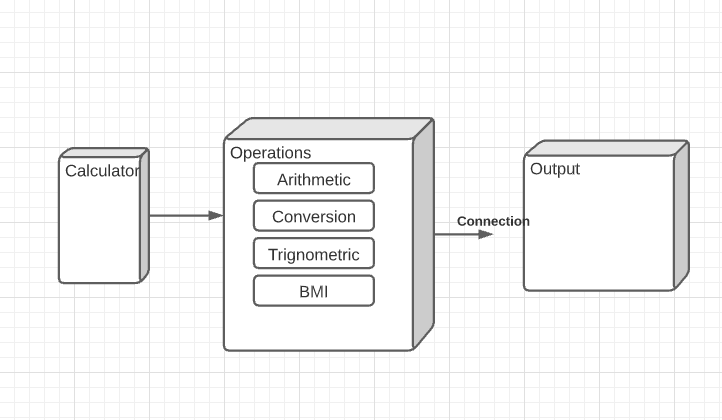


Figure 3: Deployment Diagram

## Sequential Diagram

## 

Figure 4 : Sequential Diagram

## UML Diagrams(Low level)

Class diagram



Figure 5 : Class diagram

Activity diagram

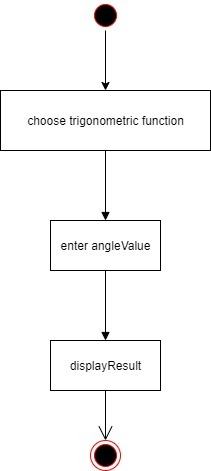


Figure 6 : Activity diagram

Use Case Diagram

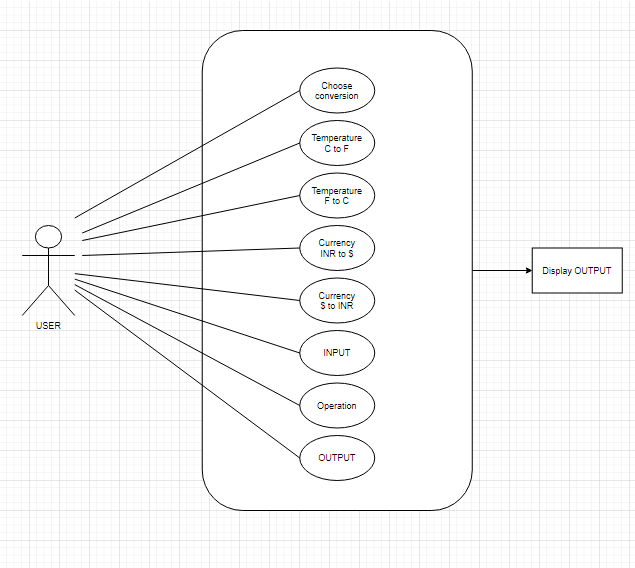


Figure 7:Use case diagram

Sequential Diagram

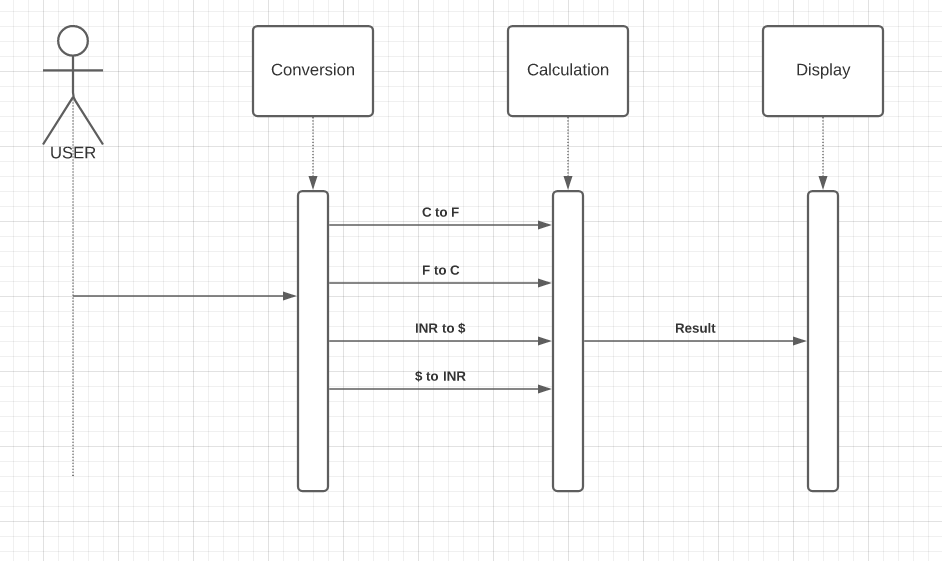


Figure 8: Sequential Diagram

Composite Diagram



Figure 9: Composite diagram

Class Diagram

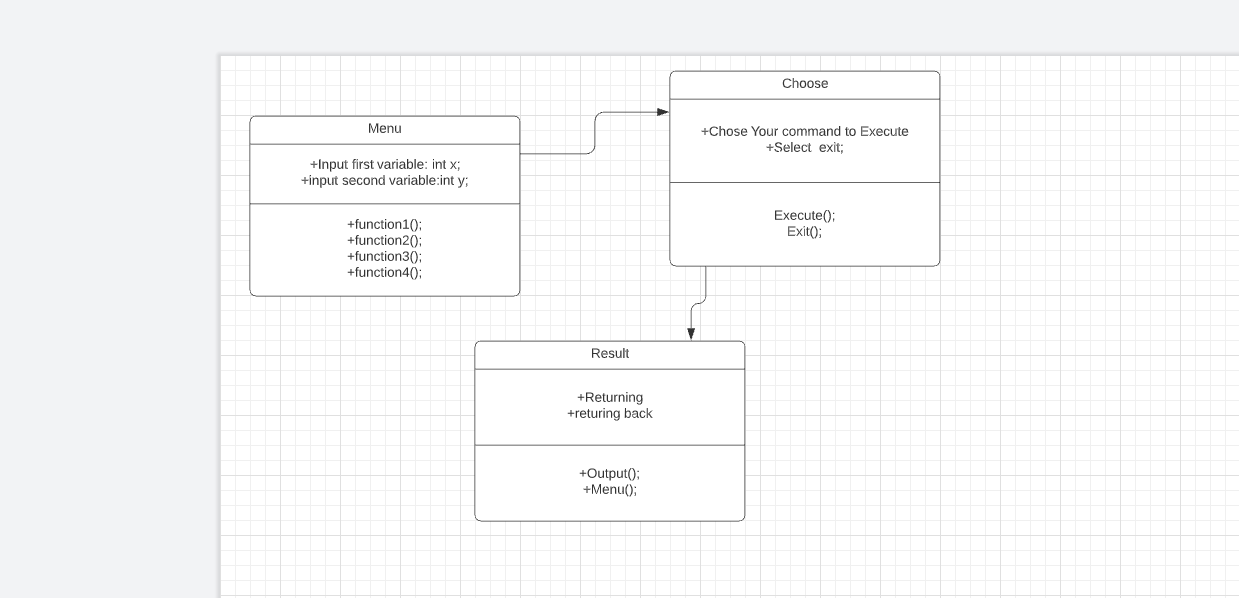


Figure 10 :Class Diagram

Behavioral Use case Diagram:

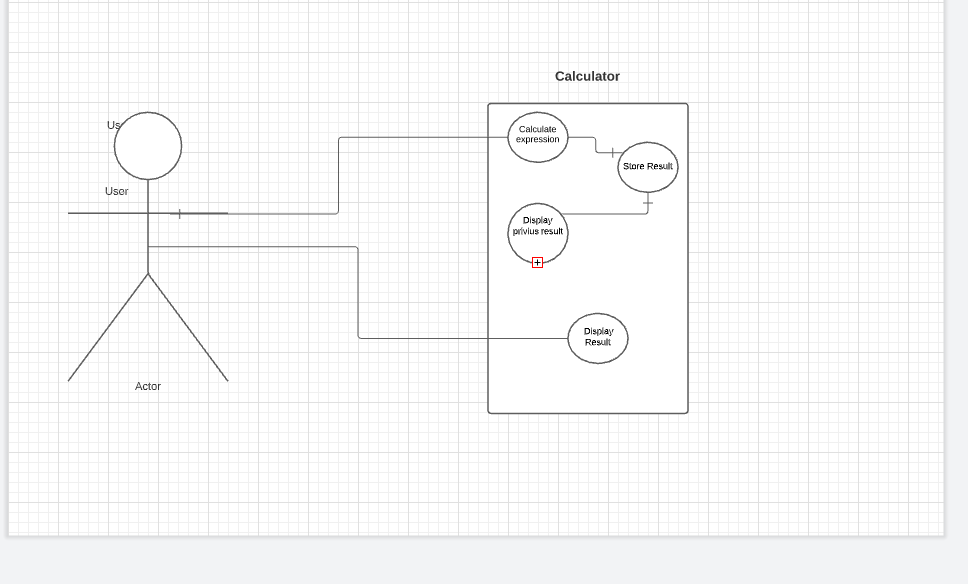


Figure 11:Use case diagram

Beahvioural State Diagram

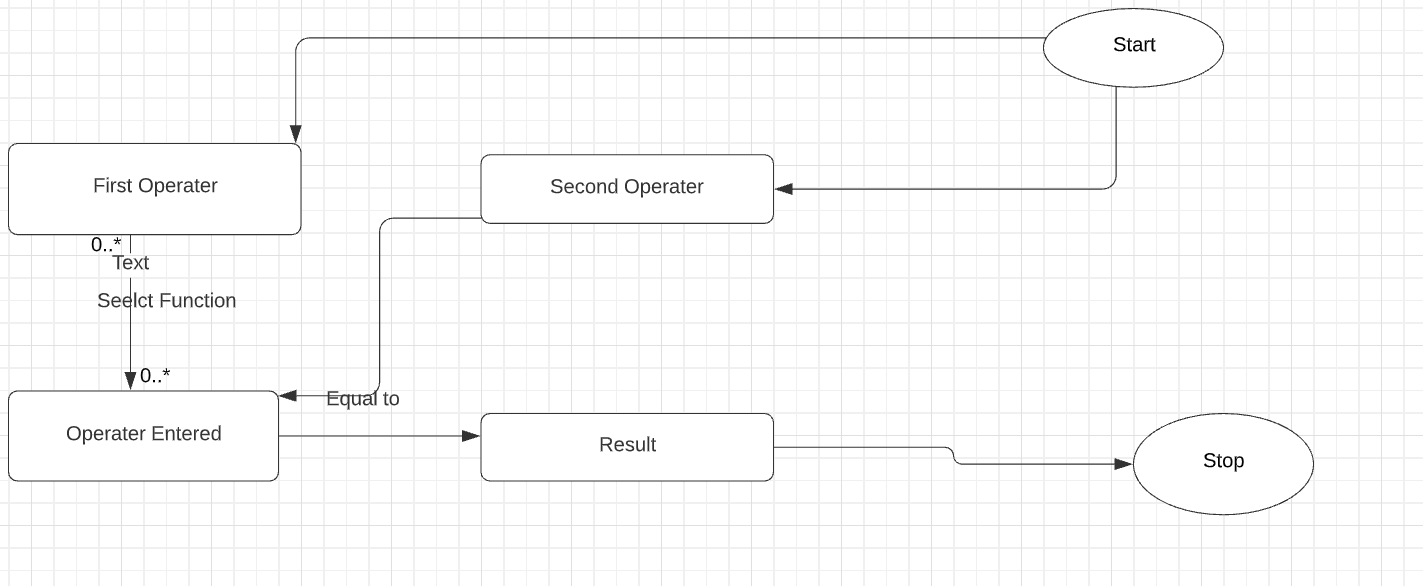


Figure 12: State Diagram

Test Plan

## High level test plan

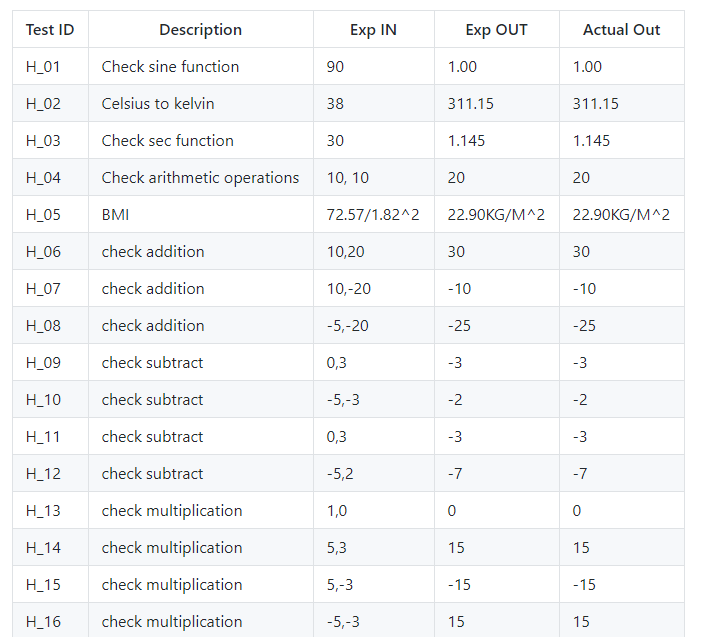
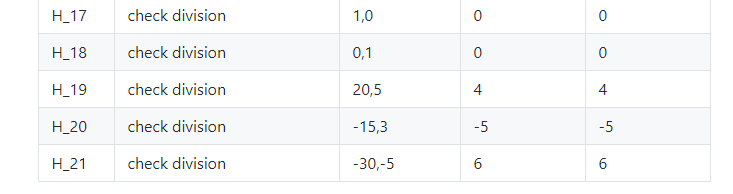


Figure 13:Test plan



Low level test plan

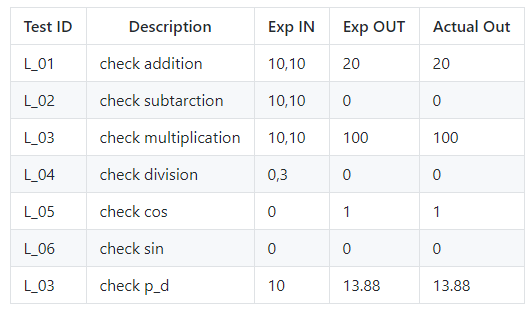


Figure 14:Test plan

## 

## Implementation Summary

Now days calculator is contributing in each one of us life, some of us using for very basic arithmetic calculation and some of using for calculating such a complex problem which take so many hours to calculate by manually. Hence in market there are various categories of calculator available based on your requirements. Some of them are made to be very specific in term of their using and some of them are used by different-different class of people who are using it. Like students, graduate students, business man, local shops and etc. If define this device in very short then we can say it is a device that performs arithmetic operations on numbers. Implementation folder has all the source files, header files, test files for different features of the calculator.

Here inc folder contains all header files with ‘.h’ extension. It contains prototype of all functions.

* The src folder holds all the source file with ‘.c’ extension. It has definition of all function whose prototype is define in inc folder.
* The test folder contain test.c file for testing of source code based on requirement, scenario, and boundary.
* The unity folder contains file which holds prototype and definition of the standard unity test case functions.
* And then there is Makefile

### Video Summary

“Please upload a short video on the repo for the walkthrough of the project (Team/Individual) less than 7min and less than 30MB File Size. Start is the Standard opening slide with title of miniproject + Team members followed by the walkthrough ”

### Git Link

https://github.com/99003765/T6final\_Calci

### Git Dashboard

Badges:-



Figure 15:Git Dashboard

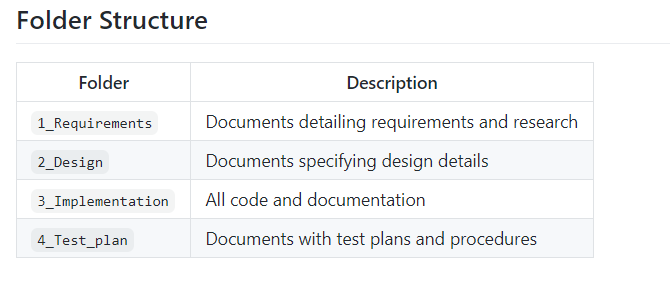


Figure 16:Folder Structure

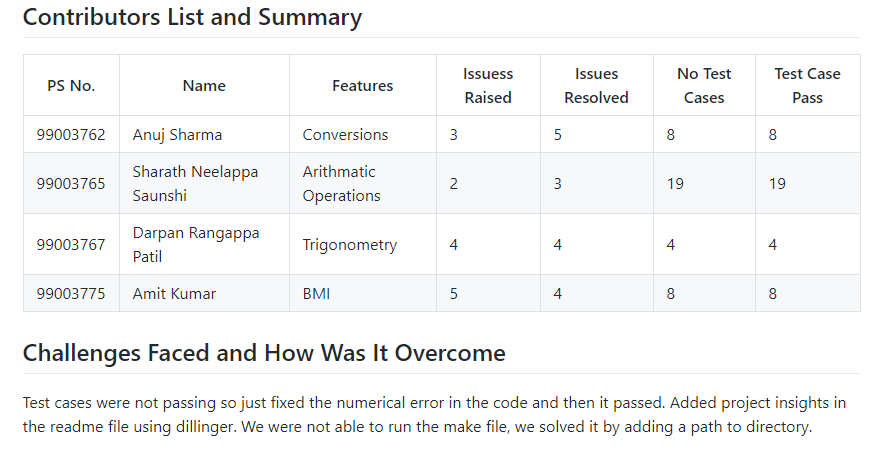


Figure 17:Contributors List

### 

### Summary

It is a simple electronic hardware/software device that is capable of performing the simple calculations such as addition, subtraction, multiplication, division, trigonometry number, binary to decimal conversion and other similar conversions and BMI etc.

#### Git inspector summary

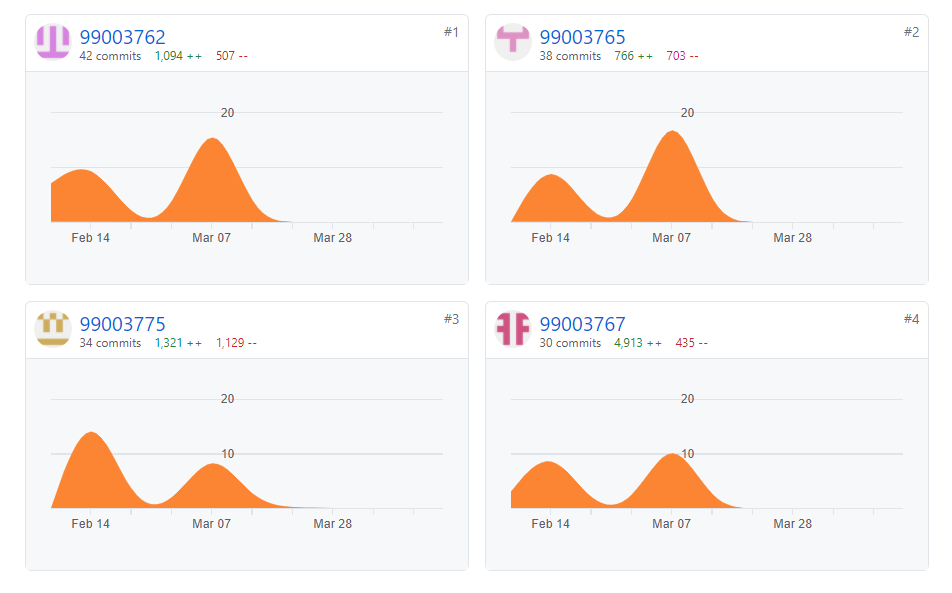


Figure 18: GIT Inspector summary

#### 

Figure 19:Commits

#### Build

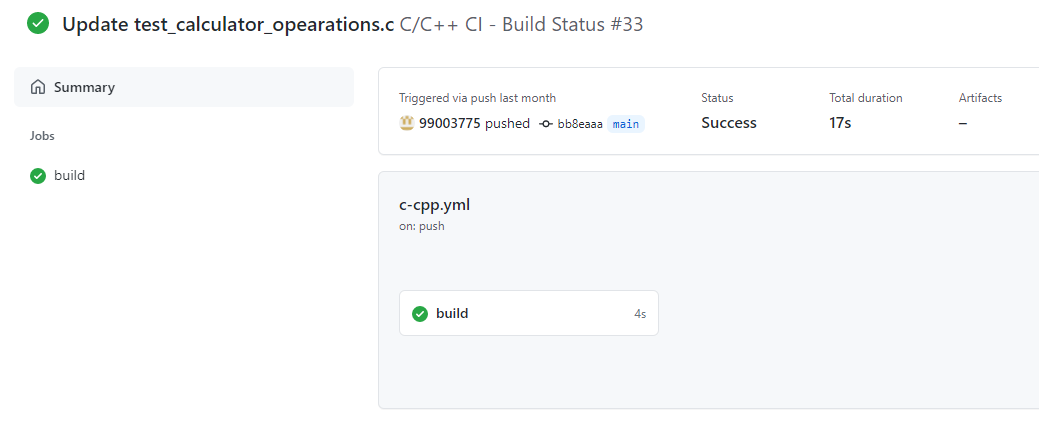


Figure 20: Build log

#### Unit Testing



Figure 21:Unit Testing

#### Individual Contribution & Highlights

* In Calculator Project, I have done Trigonometric operations.
* Implemented the feature in C language
* Implemented all possible test cases for the same
* High level and Low-Level Requirements are listed
* Plotted UML (Structural and Behavioral) diagrams for High level and Low-Level Requirements
* Listed 4W1H for the Project
* Issues are raised and respective issues are solved.
* Contributed in overall workflow and in project implementation.

### Summary

### The main motto is to design a calculator with certain features according to the specific requirements. The target customers for the designed calculator are students, shopkeepers, banking executives and engineers.

Technical:

* Improved implementation of C Concepts
* Practical Implementation of SDLC life Concepts
* Source code management(GitHub)

Soft skills:

* Project management
* Conflict management

### Challenges faced and how were they overcome

* Running the make file as its resolved by defining its correct path(.out for linux and -lm for math functions)
* Synchronizing the VS code to GitHub, colleague help to resolve the issue
* Making the function call in correct path
* Open git log while committing, thus went to GitHub desktop and pulled origin and then pushed origin.
* Test case code for the boundary problem. Added code with the help of internet
* Integration problem since lots of header file was there. Changes made in header file to remove the multiple occurrence error.

### 

# Miniproject -2 [Individual]

## Module Used

In this Module we used python language for project.

Project title : Counting the number of word defined by user and print words before and after it.

### Topic and Subtopics

## Objectives

## I am having collection data in form of a file. This program takes a user input as a keyword and search the occurrence of the word in the file and assembles all corresponding data from file and prints the word after and before it and creates a separate file for it.

## Requirements

|  |  |
| --- | --- |
| ID | Description |
| HL\_01 | To take user defined input and search for particular word in the given file |
| HL\_02 | Create a new file with the name as defined by user |
| HL\_03 | Print the word before and after it |
| HL\_04 | Print the number of occurences |

Test cases: Not Applicable.

## Implementation

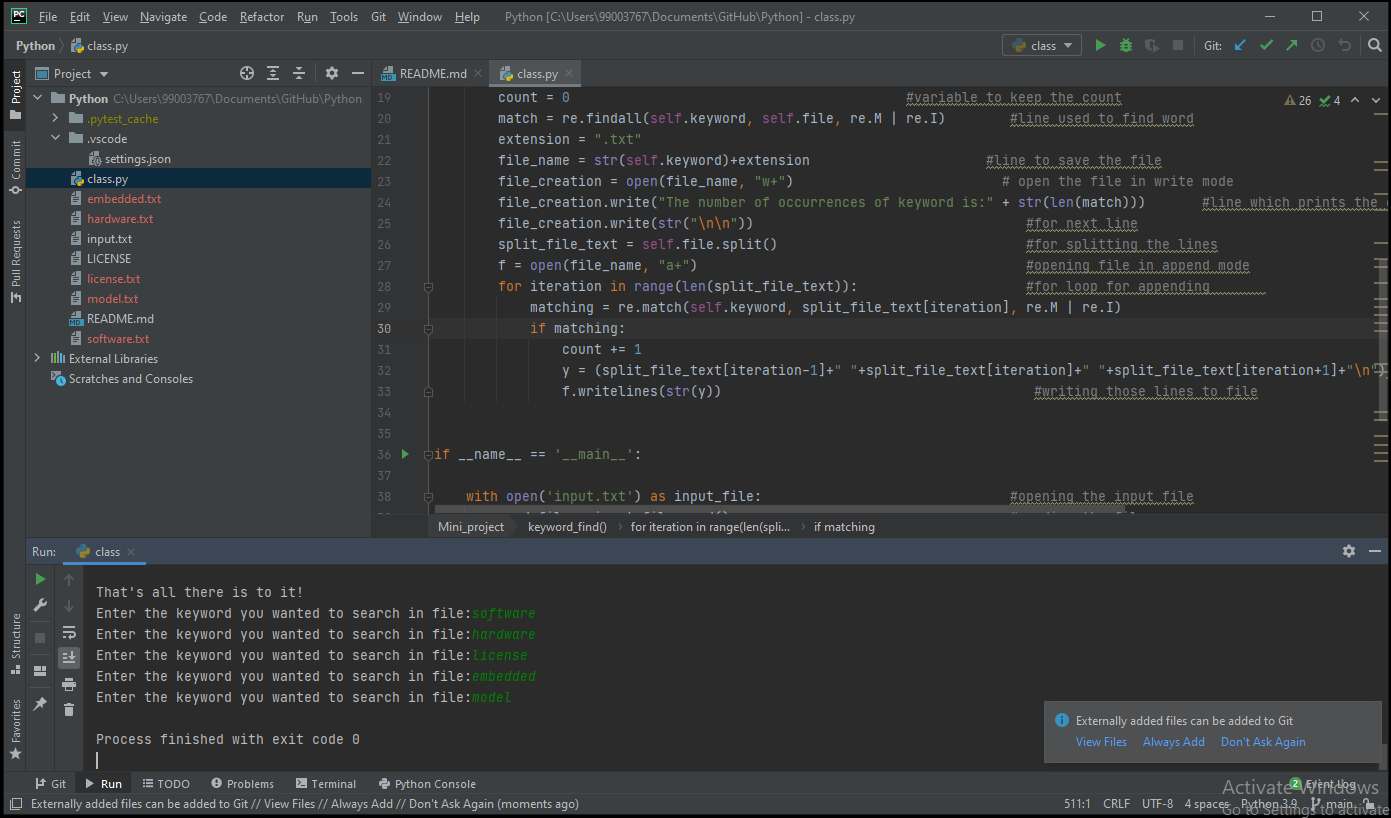


Figure 22:Python implementation

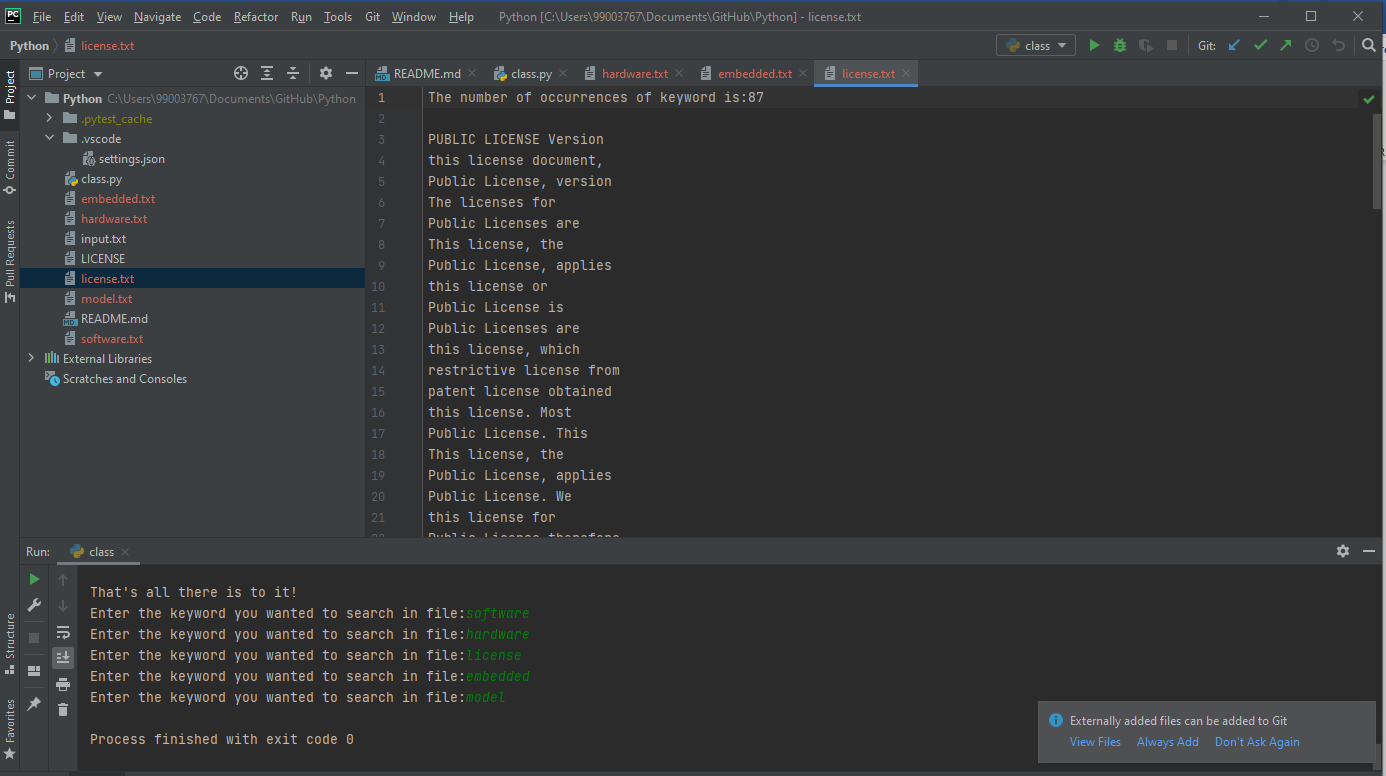


Figure 23:Python Implementation

## Implementation Summary

## I am having collection data in form of a file. This program takes a user input as a keyword and search the occurrence of the word in the file and assembles all corresponding data from file and prints the word after and before it and creates a separate file for it as you can see it in the above picture.

### Git Link

https://github.com/99003767/Python

### 

### Summary

Technical:

Improved implementation of Python Concepts

* Practical Implementation of Python Concepts
* Source code management

### Challenges faced and how were they overcome.

* Issues with regular expressions solved by practice.
* Committing to GitHub, pull and push in GitHub.
* System issues (crashing and Interfacing) .

# Miniproject -3-MBSE [Team]

## Modules used

Matlab and Simulink are the Modules are used in the project.

Project Title

Tire Pressure Monitoring System

## Objectives

## To monitor the pressure of all the tires and alert the driver if anything is wrong.

* To prevent tire burst which may lead to accident or loosing the control of vehicle.

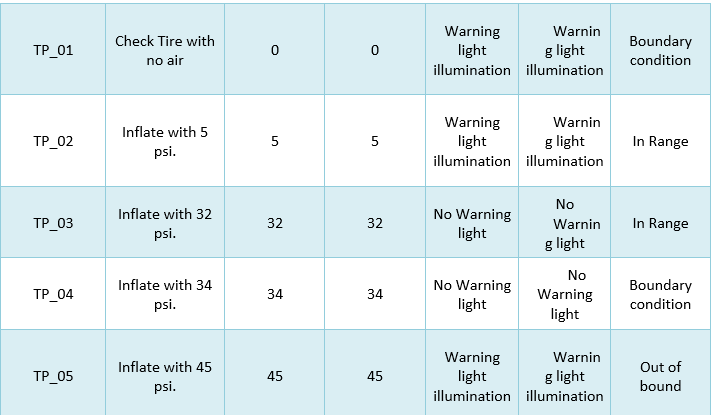
High level Requirements

|  |  |
| --- | --- |
| ID | Description |
| HL\_01 | To measure pressure of all four tires. |
| HL\_02 | Alert the driver if any of the tire has wrong pressure. |

Low level Requirements

|  |  |
| --- | --- |
| ID | Description |
| LL\_01 | Temperature range should be -25© to 95© |
| LL\_02 | Should be stable at this range of temperatures. |

Test Cases:



Implementation:

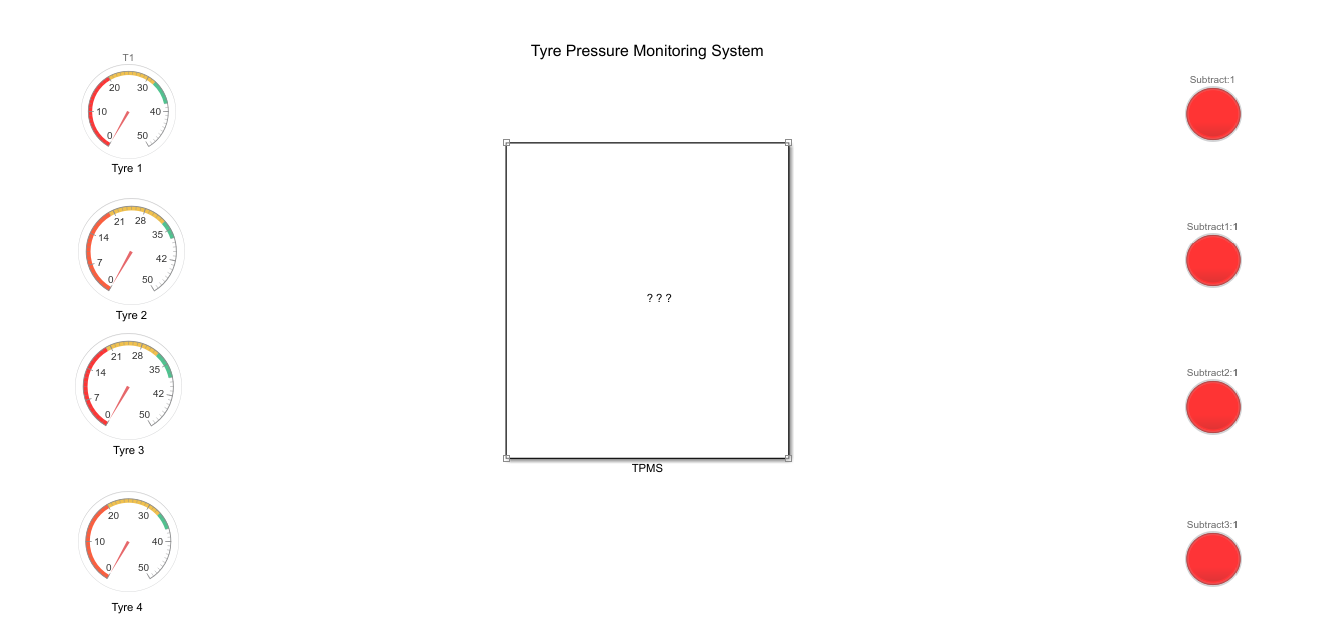


Figure 24:MBSE output

## Implementation Summary

* The system designed is able to detect the changes in the tyre pressure.
* If the pressure is below the standard pressure of 32 psi, it will alert the driver to see and fill the air in time.

Share point path:

## 

# 4.Miniproject : Embedded C [Team]

## **Module/s:**

|  |  |
| --- | --- |
| Sr. no. | Module |
|  | Stm32cube-IDE |
|  | HAL library |

## **Requirements:**

|  |  |
| --- | --- |
| Sr. no. | Requirement |
| 1. | Our designed system should have following features:   1. Remote(Lock/Unlock) control 2. Motion Detection(PIR sensor) 3. Mirror control and Steering control 4. Alcohol detection 5. Moisture sensing for AC |

## **4.2 Design:**

Our module was implemented using STM32f407VG microcontroller featuring 32 bit ARM-cortex -M4 with FPU core.

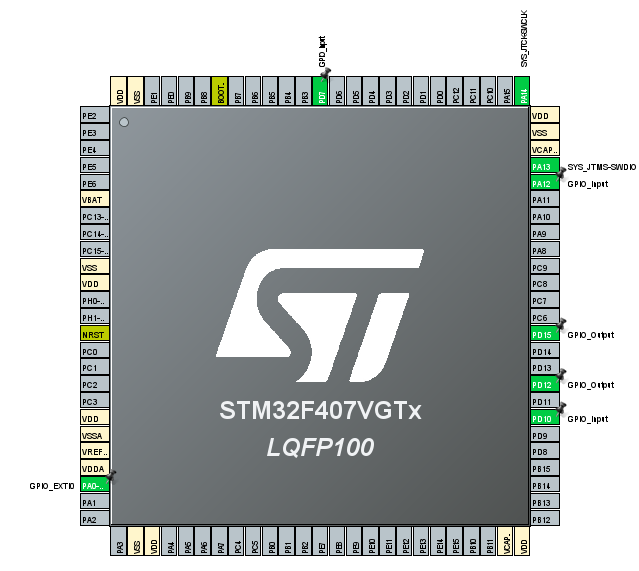


Figure :Pin Configuration

**4.3 Testing Result:**

**1)Remote(Lock/Unlock) Control:** Here the Remote control module is referred and when the button is pushed on the controller LED gets blinked. This refers that the door should open when remote button is pressed

Code:

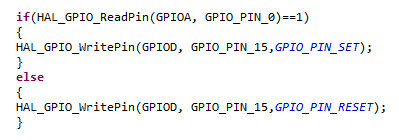


Figure 26: Remote control code

**Result**

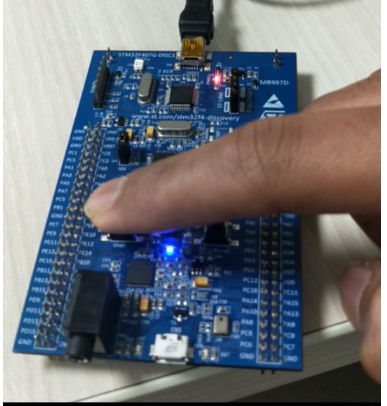
****

Figure 26 :Execution

**2)Motion Detection(PIR sensor):** PIR sensors allow you to sense motion,used to detect whether a human has moved in or out of the sensors range. If a person enters the car the seat is automatically adjusted and we showed it by blinking LED.

**Code:**

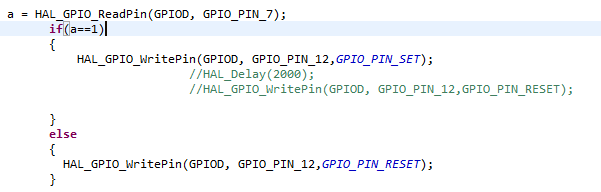
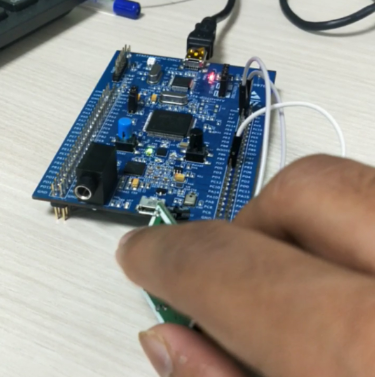


Figure 27: Motion detection

**Result:**

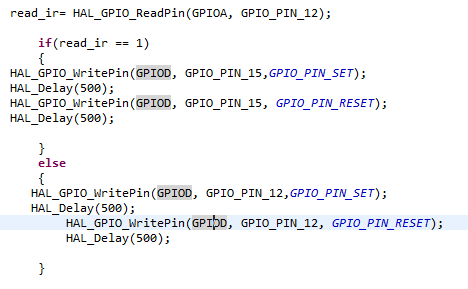
****

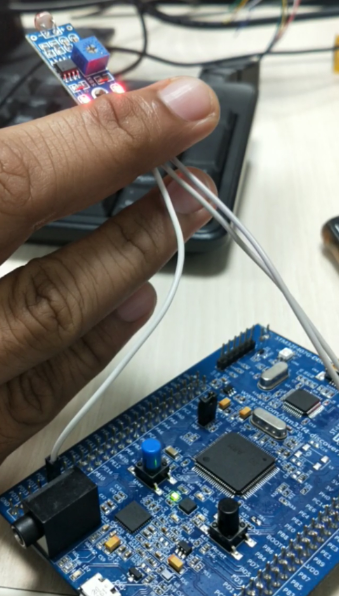
Figure

**3)** **Mirror control and Steering control**:

If the car gets started the 2 mirrors should get unfold themselves and if the engine is off the steering should get locked.Here we shown that by using LDR sensor.

**Code:**



****

Figure

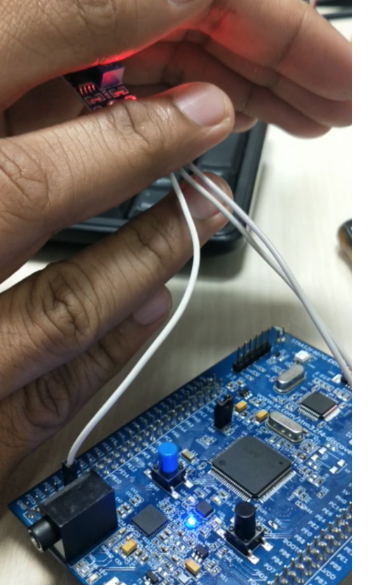
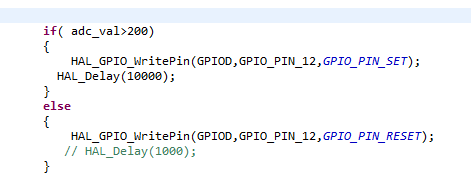
****

Figure 30: Mirror and Steering control

**4)** **Alcohol detection**

If a person consuming alcohol enters the car then it is detected using MQ7 sensor and the LED is blinked as output.

**Code:**



**Result:**

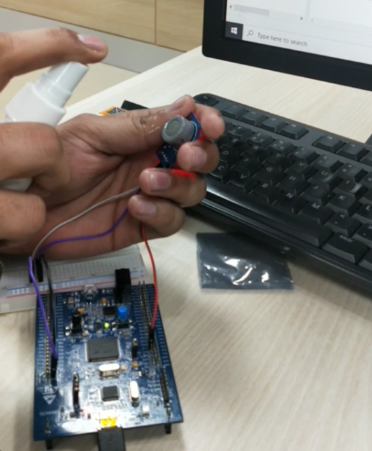
****

Figure 31: Alcohol detection

## **Individual Contribution & Highlights:**

|  |  |
| --- | --- |
| Role in project team | |
| 1. Integrator | Integration of source code files into single file |
| 1. Code contribution in functional code | Designing the Remote control and Motion Sensor system |

### Summary

**Outcomes:**

1. Stm32f407 discovery board programming.
2. Programming the features BCM module: Remote control and Motion Sensor system

**5.Miniproject -5 : Canoe and CAPL scripting [Team]**

## **5.1Module/s:**

|  |  |
| --- | --- |
| Sr. no. | Module |
|  | CAPL scripting |
|  | C Programming |

## **Requirements:**

1. Module should have following features for Airbag system:
2. Ignition control.
3. Seat belt control .
4. Airbag control.

**Functional requirements:**

1. When Ignition is turned ON ,Seat belt reminder should turn On and give the warning.
2. When Seat belt is buckled then the reminder should turn off.
3. If any crash is detected the accelerometer is high and there is sudden decelaration and airbag is deployed .
4. Airbag is deployed if and only if the seat belt is buckled.
   1. **Implementation:**

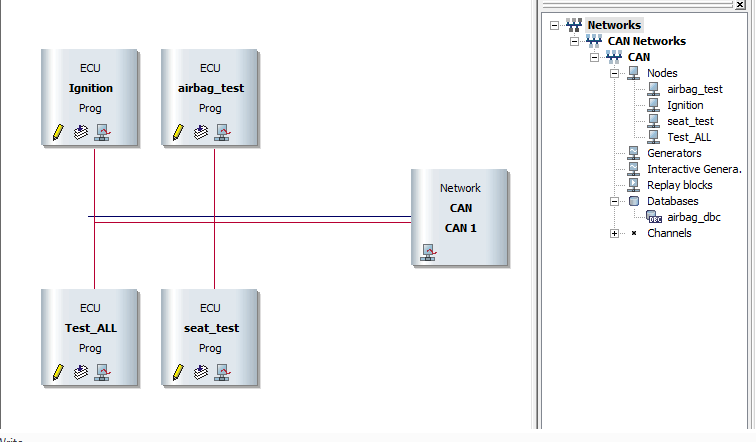
****

Figure 32: 4 nodes

* 1. **Result:**

1. **Ignition:**

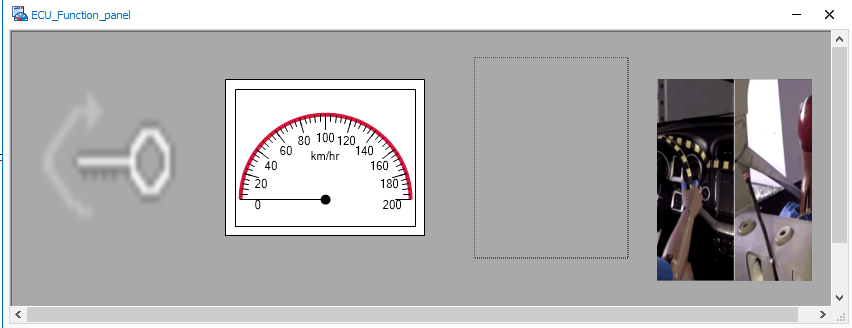
****

Figure 33: before ignition



Figure 34: ignition ON and Seat belt reminder

1. **Seat Belt Control:**

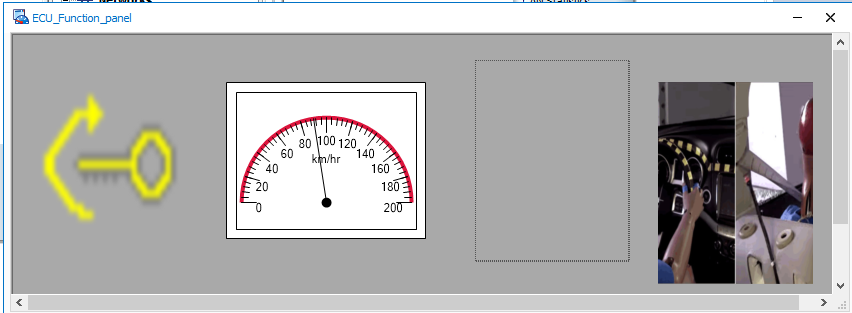


Figure 35: Reminder OFF after seat belt bulckled

1. **Airbag Deployment:**

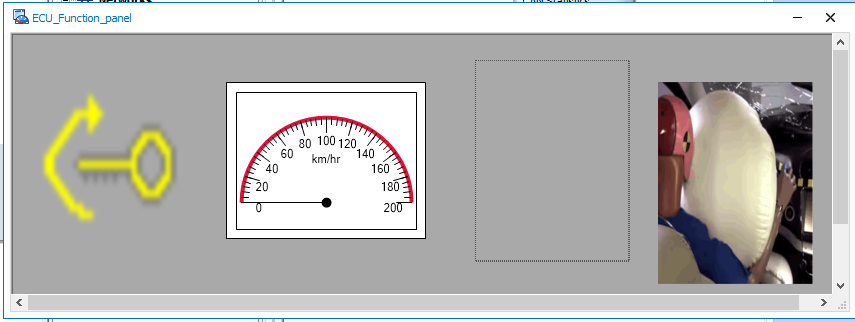


Figure 36: Airbag deployed due to sudden deceleration and crash

## **Individual Contribution & Highlights:**

|  |  |
| --- | --- |
| Role in project team | |
| 1. integrator | 1. Integration of all script and panel 2. Creating database file. 3. Creating test file. |
| 1. Code contribution in functional code | 1. Designing the control of Airbag system. 2. Writing script for Airbag module. 3. Creating the system variables. |

### Summary

**Outcomes:**

1. CAPL scripting hand on.
2. Implementation of lane keeping aid in CANoe tool.
3. Simulation of working panels.

### Challenges faced and how were they overcome

|  |  |
| --- | --- |
| Challenges | Resolved |
| 1. Jargons in CAPL scripting | Help section in CANoe tool |
| 1. Integration | Common database file and some changes in scripts. |

**6.Miniproject -6: Control System [Team]**

## **Module/s:**

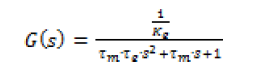
|  |  |
| --- | --- |
| Sr. no. | Module :Motor Control Analysis |
| 1 | Simulink Modeling |
| 2 | Model analysis |

## 

# 6.1 Introduction:

Here we have taken 4 systems to analyze they are BLDC with PID controller, PMSM with PID controller, Induction motor with PID controller and PMSM with PWM controller.

# 6.1.1 Brushless DC Motor with PID Controller Transfer Function:



Tm=mechanical constant

Te=Electrical time constant





Kt=Torque constant

Ke=electrical torque

## Brushless DC Motor with PID Controller:

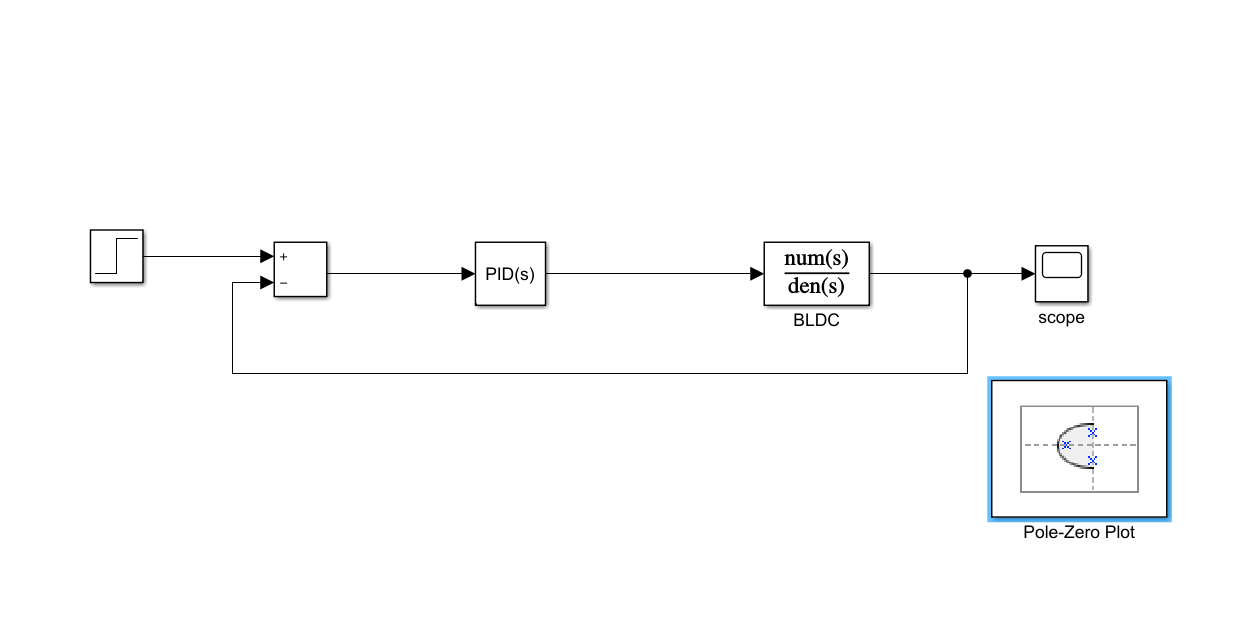


Fig1.BLDC Motor Model(Simulink)

In this system, we will be analyzing the poles and zeros of the system when the PID controller is added to the system.

* We got 1 pair of complex conjugate pole, 1 pole pair on the horizontal axis and 2 zeroes.
* We got 3 poles and 2 zeroes on the left side of the imaginary axis

So, 2 zeroes and 2 poles will nullify their effect and 1 pole will be on the left side so that we can say that system is stable.

### 6.1.2 Pole-Zero Map and Analysis:

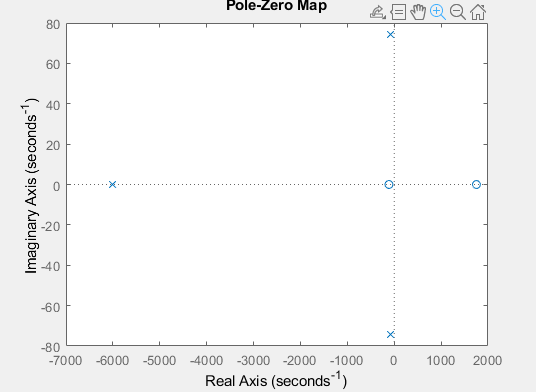


Fig2. Zero-pole plot of BLDC

Poles of BLDC system:

1.0e+03 \*

-6.0172 + 0.0000i

-0.0882 + 0.0744i

-0.0882 - 0.0744i

Zeroes of BLDC system:

1.0e+03 \*

1.7536

-0.1128

### 6.1.3 Output Graph:

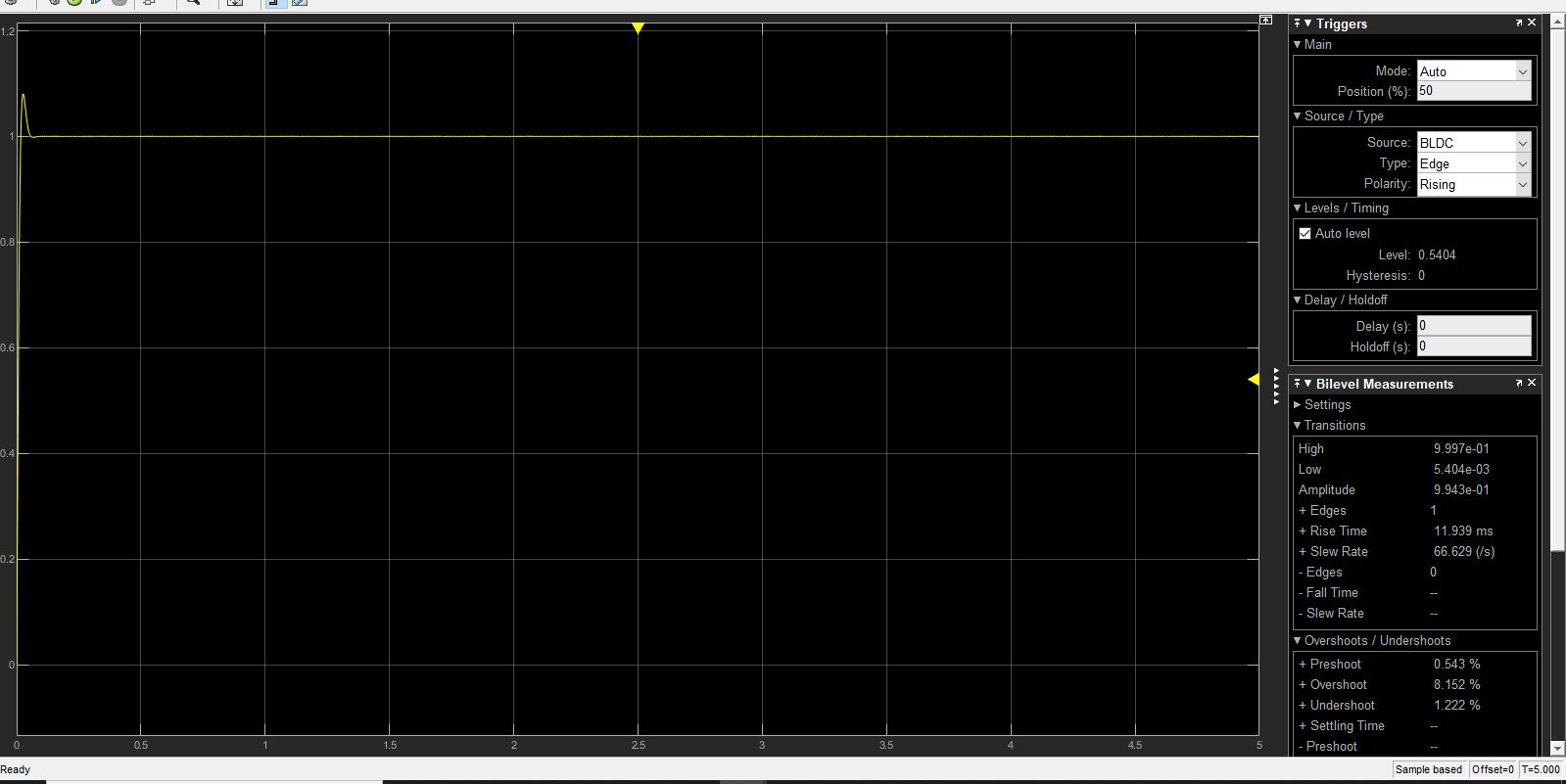


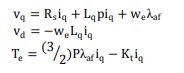
Fig3. Step Response of BLDC motor

After PID tuning we got the rise time as 11.9ms and overshoot is 8.152% as the tuning made the system parameters to adjust accordingly to get the stable system.

# 6.2 Permanent Magnet Synchronous Motor with PID Controller:

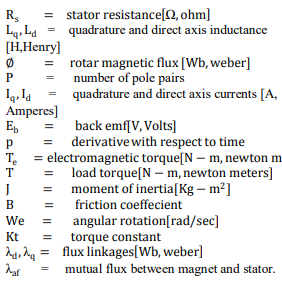
Transfer Function and Equation:

Equations:



Transfer Function:





## Permanent Magnet Synchronous Motor with PID Controller Model:

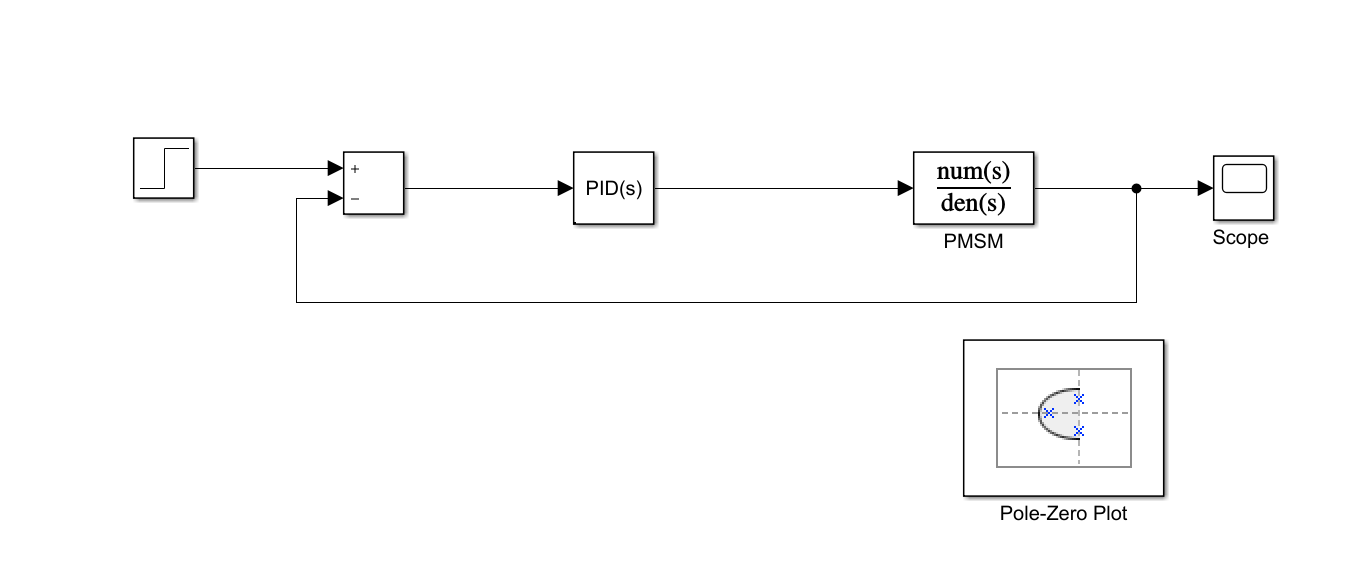


Fig4. PMSM with PID controller model

In this system, we will be analyzing the poles and zeros of the system when the PID controller is added to the system.

* We got 1 pair of complex conjugate pole, 1 pole pair on the horizontal axis and 3 zeroes.
* We got 3 poles and 2 zeroes on the left side of the imaginary axis and 1 pole and 1 zero on right side of the plane.

So, 2 zeroes and 2 poles will nullify their effect and 1 pole will be on the left side so that we can say that system is stable.

### 6.2.2 Pole-Zero map and Analysis:

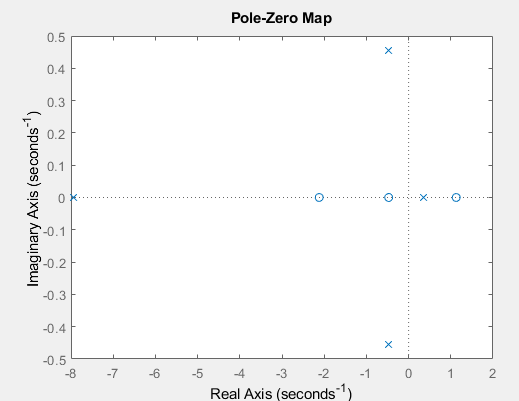


Fig5. Pole zero plot of PMSM

Poles of PMSM system:

-7.9492 + 0.0000i

-0.4733 + 0.4552i

-0.4733 - 0.4552i

0.3446 + 0.0000i

Zeroes of PMSM system:

-2.1189

1.1288

-0.4716

### 6.2.3 Output Graph:

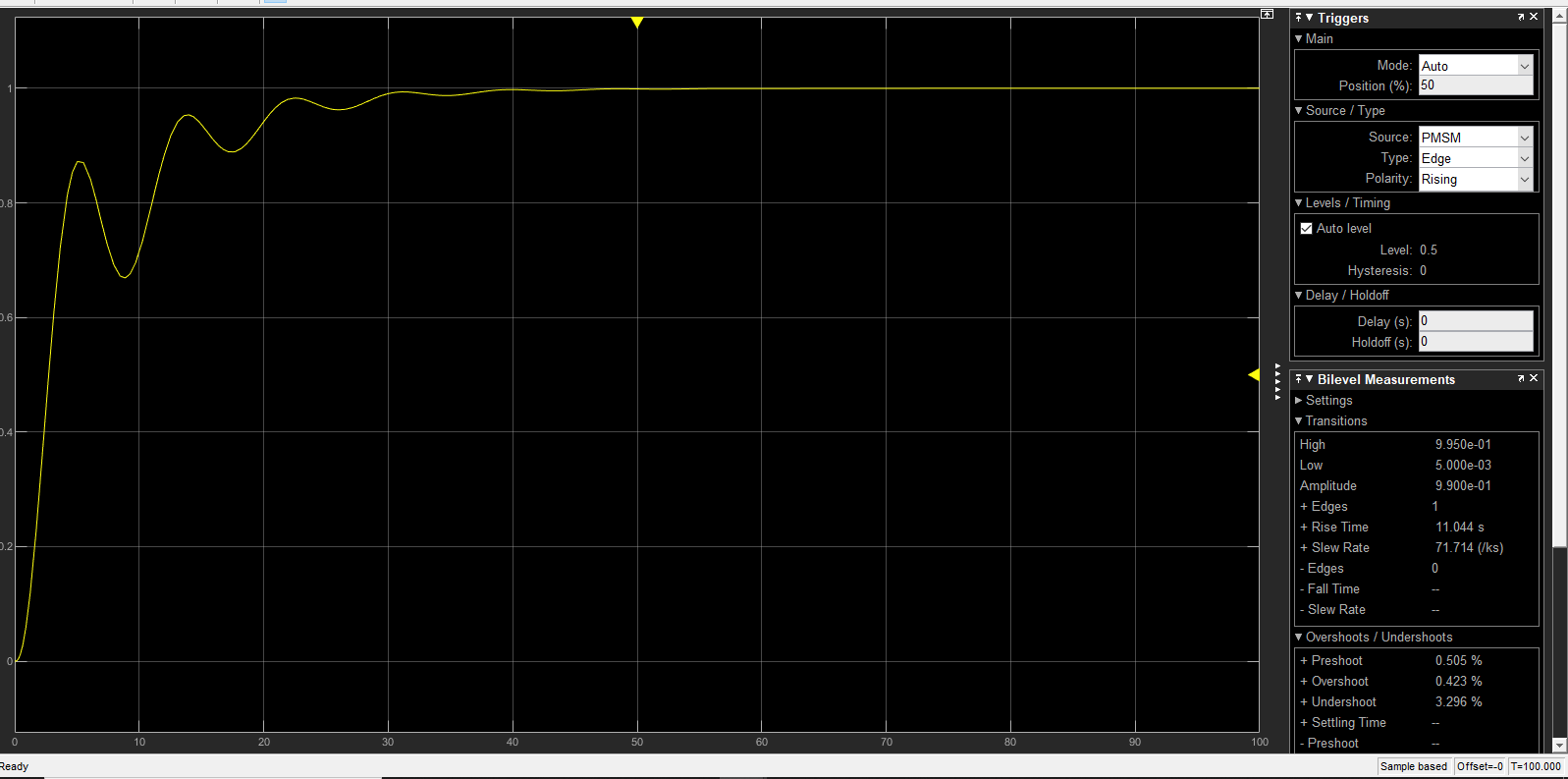
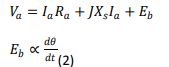


Fig6. PMSM Step Response

After PID tuning we got the rise time as 11.044ms and overshoot is 0.423% as the tuning made the system parameters to adjust accordingly to get the stable system.

# 6.3. Induction Motor with PID Controller Transfer function and Equations:



Transfer Equation:



Parameters:

 -Electrical Torque

Tm=Mechanical Torque

## Induction Motor with PID Controller:

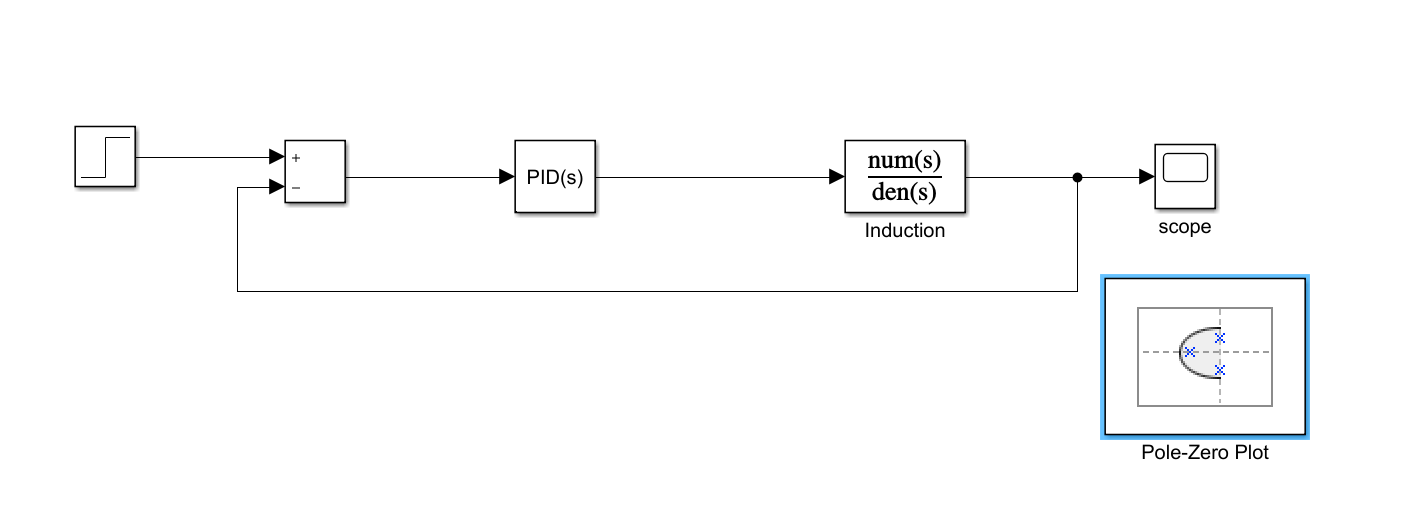


Fig7. Induction motor with PID model

In this system, we will be analyzing the poles and zeros of the system when the PID controller is added to the system.

* We got 1 pair of complex conjugate pole, 1 pole pair on the horizontal axis and 2 zeroes.
* We got 3 poles and 2 zeroes on the left side of the imaginary axis

So, 2 zeroes and 2 poles will nullify their effect and 1 pole will be on the left side so that we can say that system is stable.

### 6.3.1 Pole-Zero Map and Analysis:

In this system, we will be analyzing the poles and zeros of the system when the PID controller is added to the system.

* We got 4 poles and 3 zeroes on the left side of the imaginary axis .

So, 3 zeroes and 3 poles will nullify their effect and 1 pole will be on the left side so that we can say that system is stable.

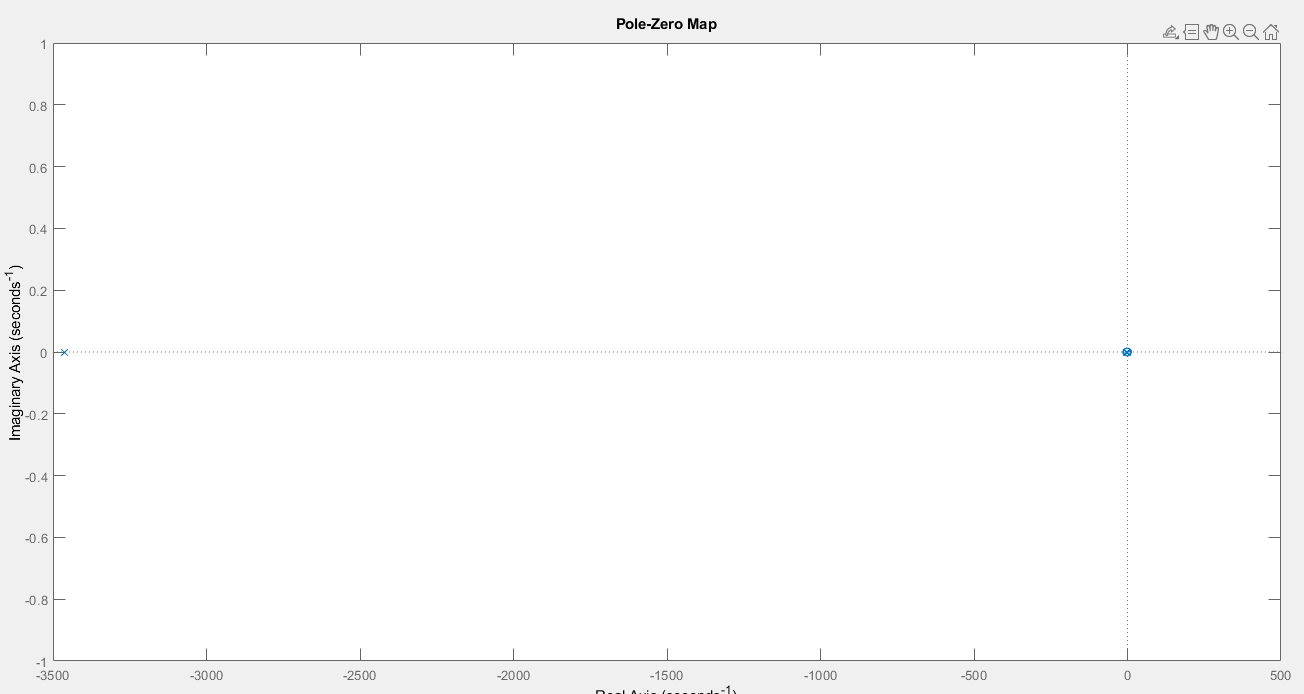


Fig8. Pole-plot of Induction motor with PID model

Poles of Induction Motor:

1.0e+03

-3.4615

-0.0023

0.0000

Zeroes of Induction Motor:

0

-2.7374

0.0118

### 6.3.2 Output Graph:

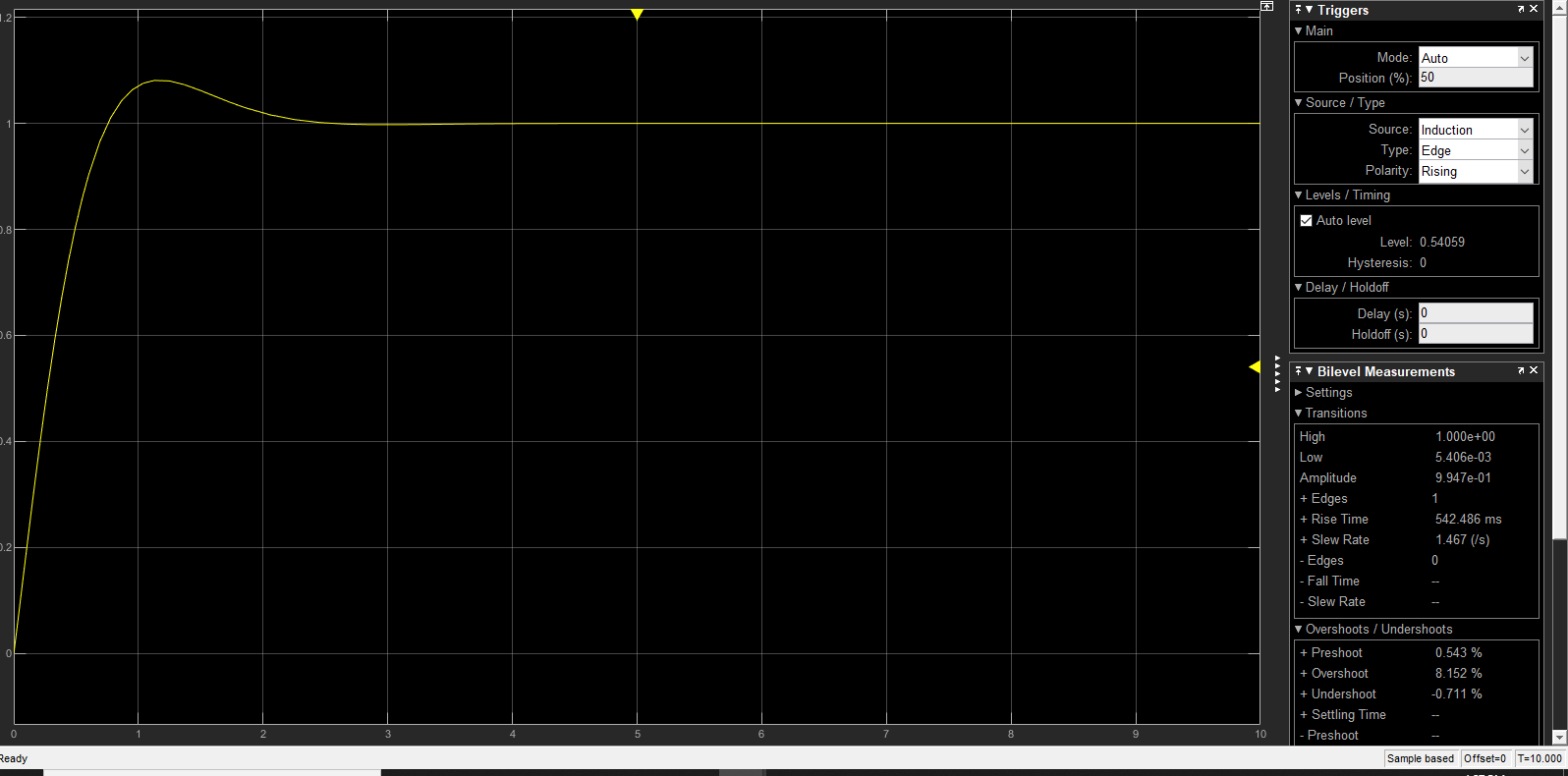


Fig9. Step Response of the PMSM after tuning

After PID tuning we got the rise time as 542.46ms and overshoot is 8.152% as the tuning made the system parameters to adjust accordingly to get the stable system.

# 6.4 PMSM Motor with PWM Controller:

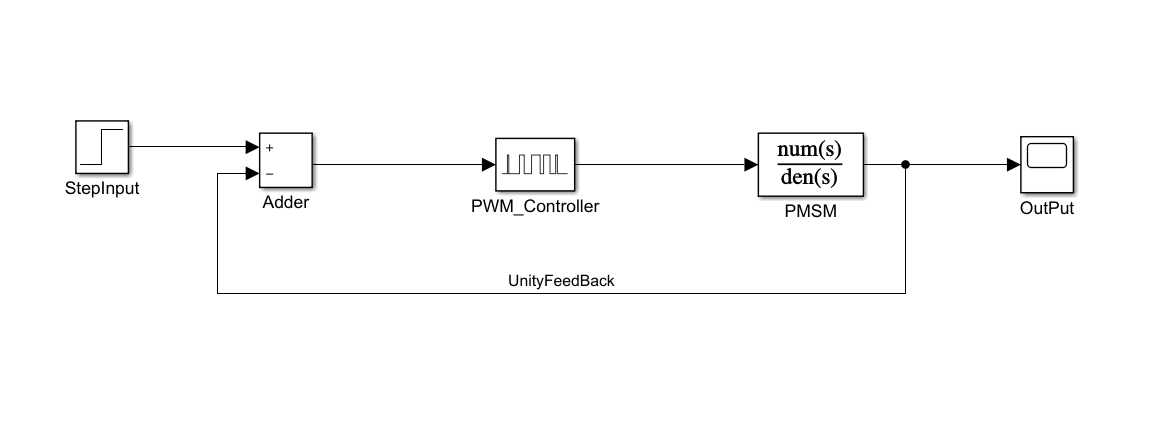


Fig10.PMSM Motor with PWM controller model

## **6.4.1 Output Graph:**

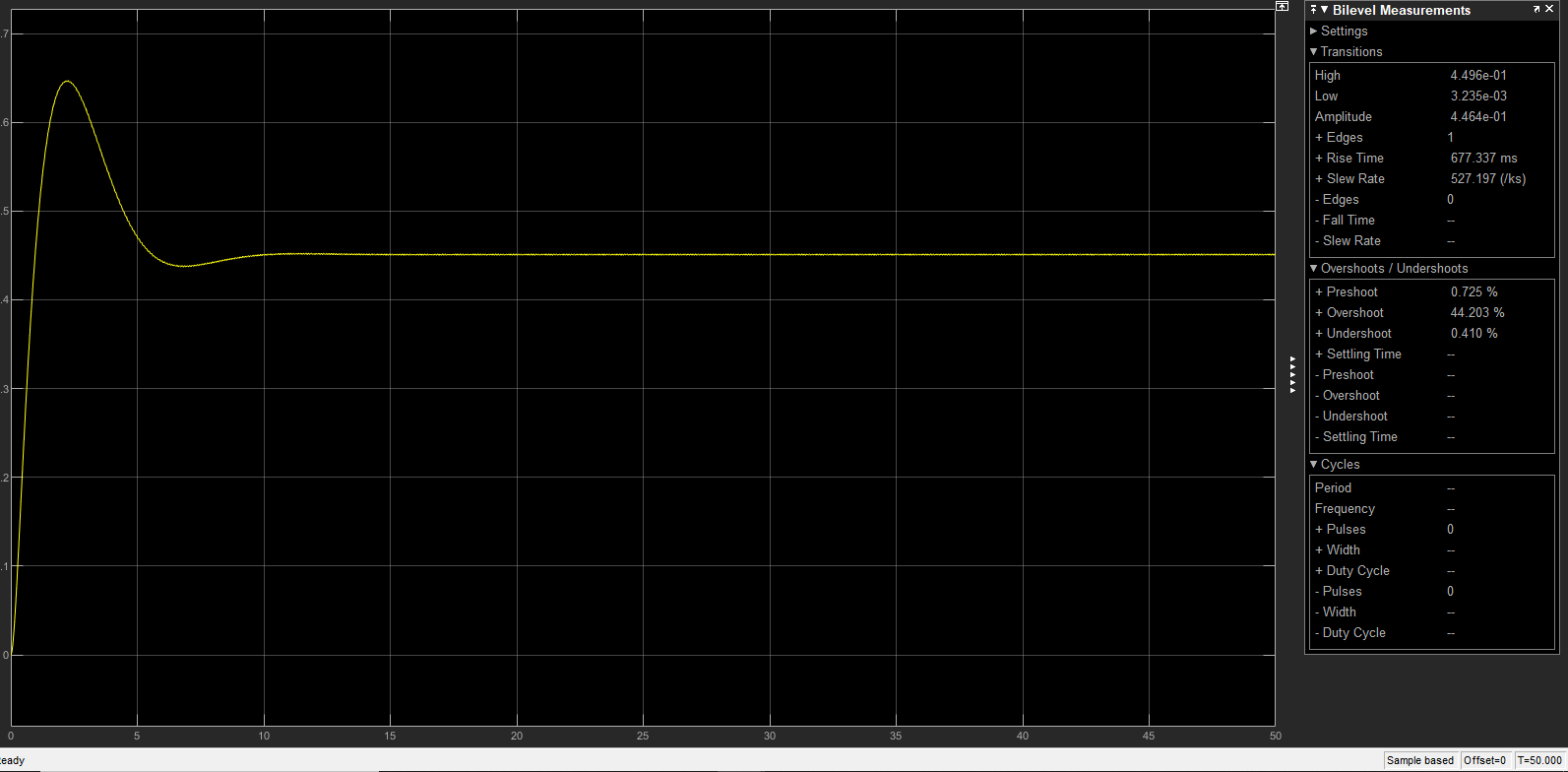


Fig11.PMSM Step Response with PWM controller

After PID tuning we got the rise time as 677.46ms and overshoot is 44% as the tuning made the system parameters to adjust accordingly to get the stable system.

# 6.5 Comparison Analysis of all systems:

By comparing the output graphs which includes Rise time, settling time, Over shoot of all the 4 above mentioned systems and by analyzing time domain of all the systems we can come to a conclusion that Permanent magnet synchronous motor with PID controller is more stable than any other systems mentioned above.

## **Individual Contribution & Highlights:**

|  |  |
| --- | --- |
| Role in project team | |
| 1.integrator | 1.Integration of Simulink model. |
| 2.Code contribution in functional code | 1.Creation of PMSM with PID models  2.PMSM With PID Analysis. |

### 6.7 Challenges faced and how were they overcome

|  |  |
| --- | --- |
| Challenges | Resolved |
| 1.Equations and Transfer functions | Referred IEEE papers and thesis |