./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69564833)

[CALCULATOR [Team] 6](#_Toc69564834)

[Module/s 6](#_Toc69564835)

[Topic and Subtopics 6](#_Toc69564836)

[Objectives & Requirements 6](#_Toc69564837)

[Design 8](#_Toc69564838)

[Test Plan 9](#_Toc69564839)

[Implementation Summary 10](#_Toc69564840)

[Video Summary 10](#_Toc69564841)

[Git Link 10](#_Toc69564842)

[Git Dashboard 10](#_Toc69564843)

[Git Inspector Summary 11](#_Toc69564844)

[Git Unit Testing Summary: 11](#_Toc69564845)

[Code Quality and Valgrid check: 12](#_Toc69564846)

[Individual Contribution & Highlights 12](#_Toc69564847)

[Challenges faced and how were they overcome 12](#_Toc69564848)

[Future Scope (If applicable) 12](#_Toc69564849)

[Miniproject - Finding Words in a Document and Printing it in a new File [Individual] 13](#_Toc69564850)

[Module: 13](#_Toc69564851)

[Topic and Subtopics 13](#_Toc69564852)

[Objectives & Requirements 13](#_Toc69564853)

[Test Plan 14](#_Toc69564854)

[Implementation Summary: 14](#_Toc69564855)

[Git Link 16](#_Toc69564856)

[Summary 16](#_Toc69564857)

[Challenges faced and how were they overcome 16](#_Toc69564858)

[Miniproject -BCM features [Team] 17](#_Toc69564859)

[Module linked to the BCM features is Embedded C. 17](#_Toc69564860)

[Topic and Subtopics 17](#_Toc69564861)

[Objectives & Requirements 17](#_Toc69564862)

[Design and TestPlan: 17](#_Toc69564863)

[Implementation Summary 20](#_Toc69564864)

[Individual Contribution & Highlights 20](#_Toc69564865)

[Summary 20](#_Toc69564866)

[Miniprojector -CANOE an CAPL Scripting[Team] 21](#_Toc69564867)

[Module linked to the CANOE, CAPL scripting 21](#_Toc69564868)

[Objectives & Requirements 21](#_Toc69564869)

[Design 21](#_Toc69564870)

[Test Plan 22](#_Toc69564871)

[Implementation Summary 22](#_Toc69564872)

[Individual Contribution & Highlights 24](#_Toc69564873)

[Challenges faced and how were they overcome 24](#_Toc69564874)

[Miniproject -Evolved MBSE with HIL testing [Team] 25](#_Toc69564875)

[Module involved in Evolved MBSE with HIL testing are MATLAB, SIMULINK, STATEFLOW 25](#_Toc69564876)

[Topic and Subtopics 25](#_Toc69564877)

[Objectives & Requirements 25](#_Toc69564878)

[Design 26](#_Toc69564879)

[Low Level Design of Hatch Control(INDIVIDUAL): 27](#_Toc69564880)

[Test Plan 27](#_Toc69564881)

[Implementation Summary 29](#_Toc69564882)

[Individual Contribution & Highlights 30](#_Toc69564883)

[Challenges faced and how were they overcome 30](#_Toc69564884)

TABLE OF FIGURES:

[Figure 1 High Level Requirement of conversion from degree to radians and vice versa 8](#_Toc69564885)

[Figure 2 State diagram for Low level requirement of conversion from degree to radians and vice versa 8](#_Toc69564886)

[Figure 3 High level requirement test plan 9](#_Toc69564887)

[Figure 4 Git Badges 10](#_Toc69564888)

[Figure 5 Git Inspector Summary 11](#_Toc69564889)

[Figure 6 Git Unit testing summary 11](#_Toc69564890)

[Figure 7 Code quality Summary 12](#_Toc69564891)

[Figure 8 Output of the code 14](#_Toc69564892)

[Figure 9 Main code snippet 15](#_Toc69564893)

[Figure 10 Accessing the main file snippet 15](#_Toc69564894)

[Figure 11 Output files getting generated script for the given inputs snippet 16](#_Toc69564895)

[Figure 12 Pin Configurations selected for the features 17](#_Toc69564896)

[Figure 13 Code Snippet for Hatch Control 18](#_Toc69564897)

[Figure 14 Microcontroller implementation of hatch control 18](#_Toc69564898)

[Figure 15 Code snippet for Exterior Lighting 18](#_Toc69564899)

[Figure 16 Implementation of exterior lighting in Controller 19](#_Toc69564900)

[Figure 17 Code snippet for Interior lighting 19](#_Toc69564901)

[Figure 18 Image of the microcontroller implementation of interior lighting(PinkLight) 19](#_Toc69564902)

[Figure 19 Code snippet for Wiper control 20](#_Toc69564903)

[Figure 20 Microcontroller implementation of Wiper Control 20](#_Toc69564904)

[Figure 21 CANOE implementation of BCM Feature 21](#_Toc69564905)

[Figure 22 Dashboard of CANOE Project for BCM feature 22](#_Toc69564906)

[Figure 23 Database with different messages and signals 23](#_Toc69564907)

[Figure 24 Trace window where we can see the signal outputs 23](#_Toc69564908)

[Figure 25 CAPL scripting for Door 24](#_Toc69564909)

[Figure 26 High level Design of BCM features of MBSE 26](#_Toc69564910)

[Figure 27 Low level design of Hatch opening of Evolved MBSE 27](#_Toc69564911)

[Figure 28 Hatch opening sensor conversion to digital 27](#_Toc69564912)

[Figure 29 Algorithm for Hatch control low level requirement of evolved MBSE. 29](#_Toc69564913)

TABLE OF TABLES:

[Table 1 Test Plan for high level and low-level requirement of Conversion 10](#_Toc69564914)

[Table 2 Python Test input and Output 14](#_Toc69564915)

[Table 3 Test Plan for CANOE BCM feature 22](#_Toc69564916)

[Table 4 High level requirements of evolved MBSE 25](#_Toc69564917)

[Table 5 Low level Requirements of Evolved MBSE 26](#_Toc69564918)

[Table 6 High level Requirement Test plan of Evolved MBSE 28](#_Toc69564919)

[Table 7 Low level test plan of Hatch control Evolved MBSE 29](#_Toc69564920)

# CALCULATOR [Team]

## Module/s

Modules linked to the calculator project is SDLC.

### Topic and Subtopics

* Software Life Cycles:

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

Learnt about Release cycles and 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 gitDesktop and through git commands.
3. Learnt about how to generate a 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

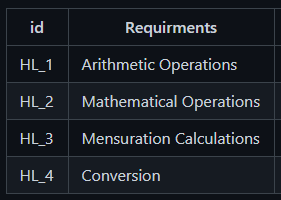
1. Took state structural model and class behavioral model for the low-level requirement and Deployment behavioral model for the High-level requirement.

* Testing:

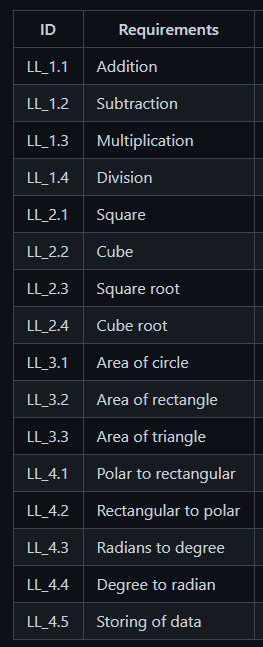
1. Testing types functional testing and nonfunctional testing and learnt that Nonfunctional testing has no impact on the features.
2. In the functional testing, we have different types namely Boundary value test, Requirement based test, Scenario based test.

## Objectives & Requirements

“High level requirements” –



“Low level requirements”-



“Objective”-

A calculator is a machine which allows people to do math operations more easily. For example, most calculators will add, subtract, multiply, and divide. Some also do square roots, and more complex calculators can help with calculus and draw function graphs. Calculators are found everywhere.

## Design

“System Level and subsystem level UMLs – Structural and Behavioral”

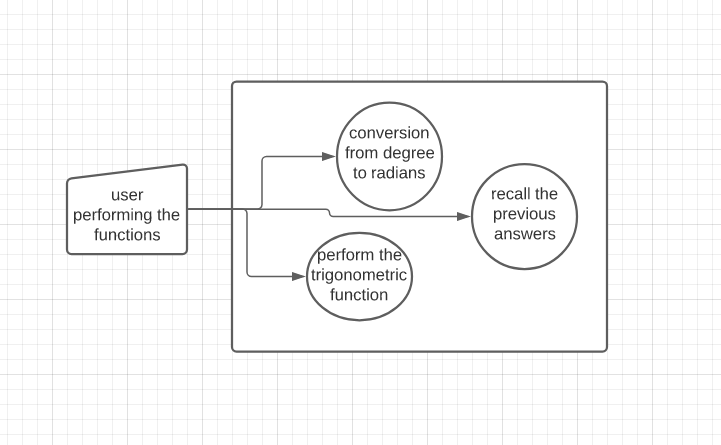
****

Figure High Level Requirement of conversion from degree to radians and vice versa

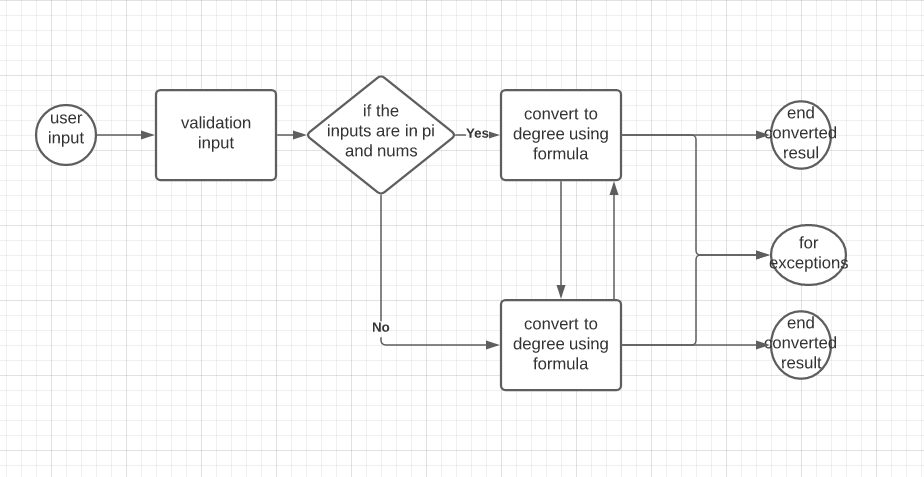
****

Figure State diagram for Low level requirement of conversion from degree to radians and vice versa

## Test Plan

“Integration level and unit level in the template”



Figure High level requirement test plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirements | Description | Given Input | Expected output | Actual output | Type of Test |
| H\_L 1 | Input taken as string for conversion | adfjh | Invalid input | Invalid input | Scenario based |
| L\_L 1 | Input given is degree to into convert radian | 90 | 1.5708 | 1.5708 | Scenario based |
| L\_L 2 | Input given is radian to convert in to degree | 1.5708 | 90 | 90 | Scenario based |

Table Test Plan for high level and low-level requirement of Conversion

## Implementation Summary

After deciding the features from each team mate, we started to build the header files, source files and test plans in Vscode as per the UML diagrams we designed and after that every ones contribution has been pushed in to GitHub in to the main branch after doing their tasks in their respective branches. So, after completing implementation of the whole requirements in Vscode we have created a make file and tested it and pushed it into GitHub, done the CPP check and added some badges to the repository which checks the code in the repository meets the required standard

### 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

“Link to the repo”- [99003724/LTTS722-727 (github.com)](https://github.com/99003724/LTTS722-727)

### Git Dashboard

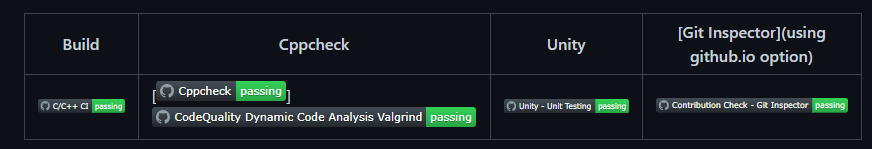


Figure Git Badges

Git Inspector Summary:

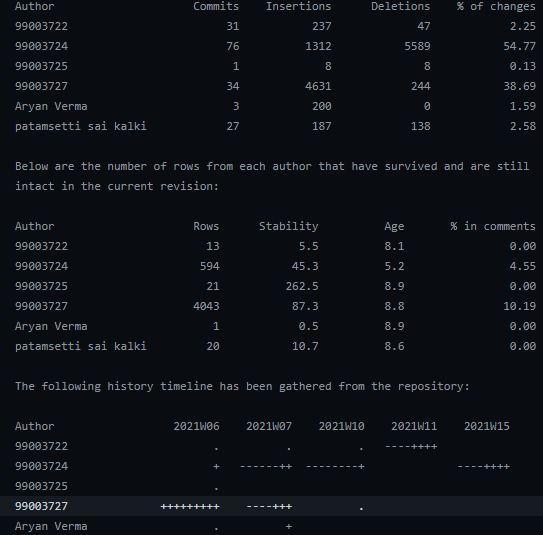


Figure Git Inspector Summary

## Git Unit Testing Summary:

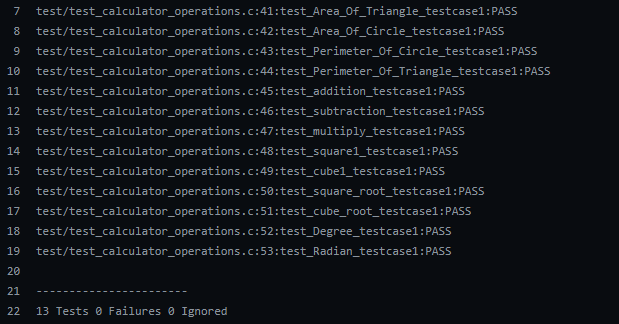


Figure Git Unit testing summary

## Code Quality and Valgrid check:

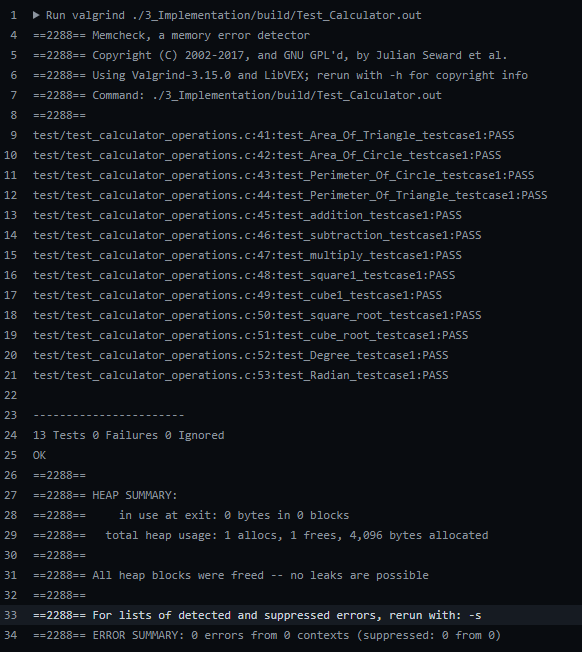


Figure Code quality Summary

## Individual Contribution & Highlights

Conversion of degree to radians and degree to radians. Helped the Team in integrating the code and helped to create a make file.

### Challenges faced and how were they overcome

* Faced the challenge in merging of the individual branch to the main branch git has thrown some errors and I solved it by manually editing the code as a solution which was suggested by git itself.
* Faced the challenge in making make file then fixed the environmental path variable and gave the correct command in vscode command prompt to generate the make file.

### Future Scope (If applicable)

### 

# Miniproject - Finding Words in a Document and Printing it in a new File [Individual]

## Module:

Modules linked to Finding Words in a Document and printing it in a new file is PYTHON.

### Topic and Subtopics

* Hacker Rank:

Done python intermediate course and did the activities in the course.

* Solo Learn:

Done the Python solo learn course and learnt some advanced topics in the solo learn course.

* Pycharm:

Learnt the PYCHARM tool to write and run the code in the Pycharm environment and linking it to GITHUB.

* Notepad++:

Learnt the NOTEPAD++ tool to write and run the code in the Notepad++ environment and linking it to GITHUB.

* VSCODE:

Used Vscode to write, Execute and push the code in to GITHUB.

* PEP8 Tool:

Used to check the quality of code.

* PYTEST:

Learnt what is PYTEST, performed some activities using PYTEST and performed some testing using pytest commands.

* Regular Expressions:

1. Learnt about how to use regular expressions in finding a string in a file or document.
2. Learnt the shortcuts in regular expressions to use it in the code and make life simpler.

* OOP’s Concepts:

1. Polymorphism
2. Abstraction
3. Inheritance
4. Encapsulation
5. Method
6. Object
7. Class

* Exception handling & file handling.

## Objectives & Requirements

Create a function to take input file from the user, 5 key words will be asked to search in the files from the user and 5 files should be generated with the information about how many times the word is repeated and should print the words before and after the word given by the user.

High level Requirement:

Give the string as an input and the code should generate a separate file with the word.

Should print how many times the word is getting repeated.

Low level Requirements:

1. Give a non-string as an input the code should throw an exception.
2. Give the input as uppercase or lower case it has to take the input and find the string in the file without the problem of uppercase and lower case and has to generate the files.

## Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Repetition | Before word | After word |
| Ok | 1 | Is | Then |
| key | 0 | No word | No word |

Table Python Test input and Output

For the key word is we got 253 repetitions and 253 before words and after words you can see the below snippet to observe the result.

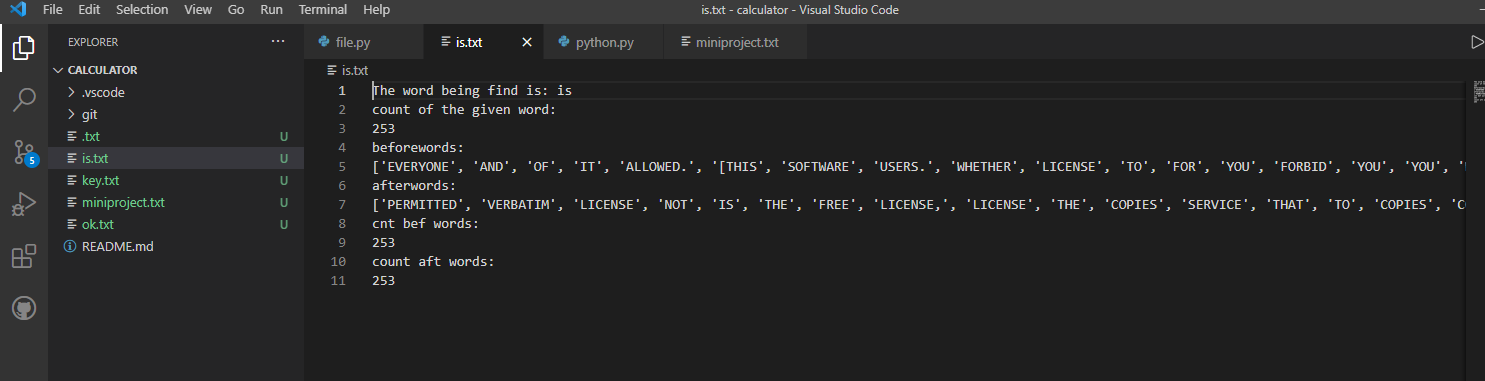


Figure Output of the code

## Implementation Summary:

In vscode I have written the 2 scripts one contains the main file and the other script contains the import function in which I will be importing the main script and I used inheritance concept to execute the code.

In the main script I used regex to find the words in the given input file and had put the function in the while loop so that it can run the code according to the user string inputs.

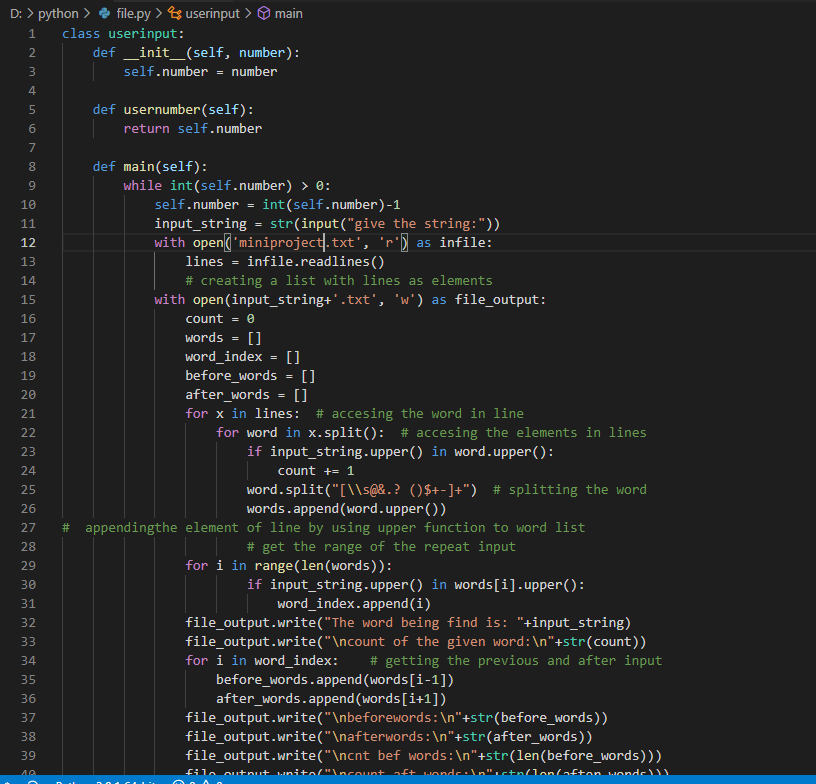


Figure Main code snippet

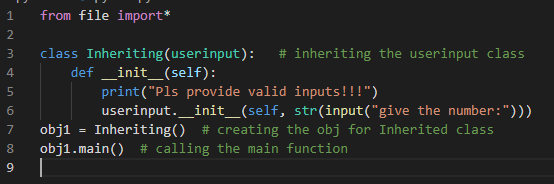


Figure Accessing the main file snippet

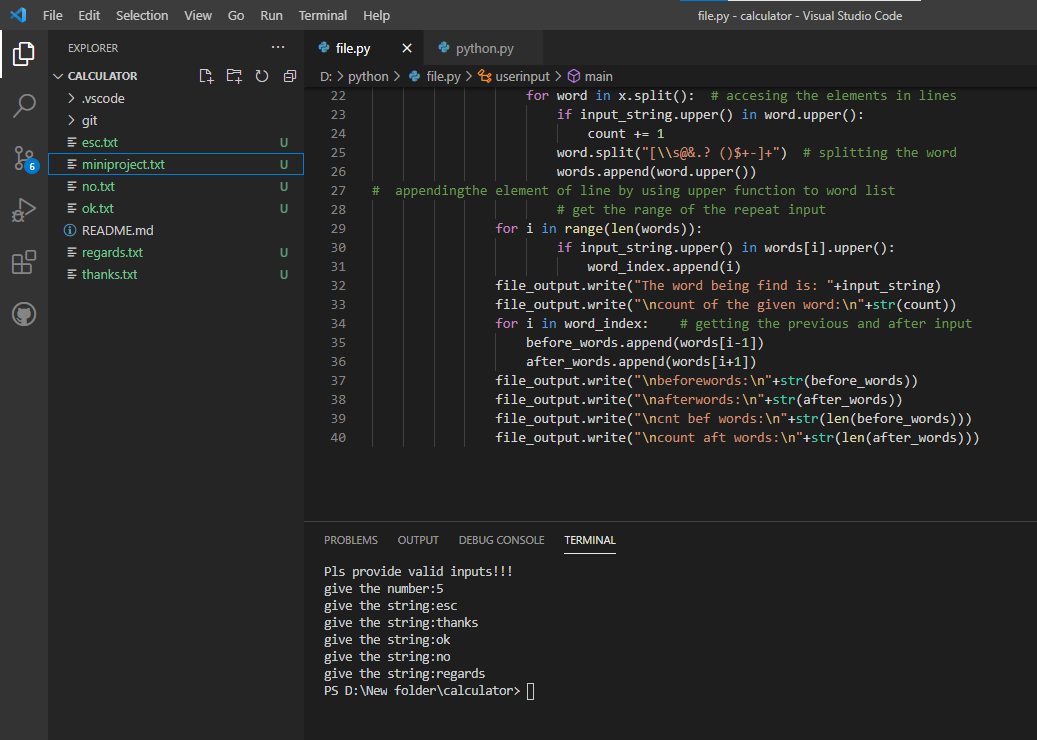


Figure Output files getting generated script for the given inputs snippet

### Git Link

[99003727/python-module-shiva: for python module (github.com)](https://github.com/99003727/python-module-shiva)

### Summary

Used the concepts of file accessing in solo learn and regular expressions helped me in creating the main function script and the OOP’s concepts helped me in creating the second script to access the files from the main script. So after the code execution the code will ask the user like how many inputs he wants and accordingly the code will ask the user to give the corresponding string inputs and the code will generate the text files according to the given strings by the user.

### Challenges faced and how were they overcome

* Was unable to import the main script file in to another script and unable to inherit the classes present in the main script solved it by writing import\* which access all the files in the main script.

# Miniproject -BCM features [Team]

## Module linked to the BCM features is Embedded C.

### Topic and Subtopics

* Make Start Up file
* Linker Script
* Debugging techniques
* HAL library
* Stm32cube-IDE
* file

## Objectives & Requirements

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

## Design and TestPlan:

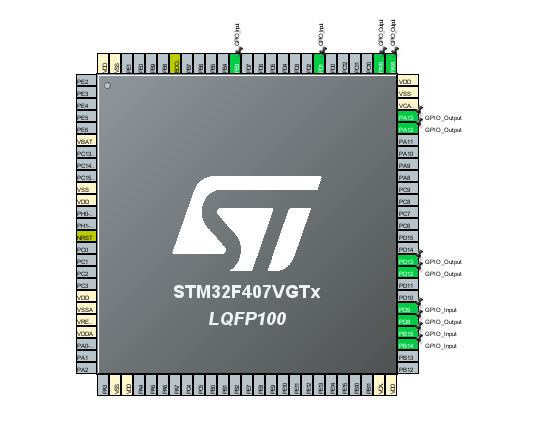


Figure Pin Configurations selected for the features

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

1. Hatch open

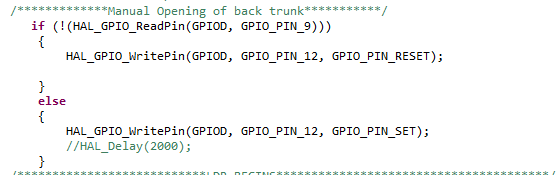


Figure Code Snippet for Hatch Control

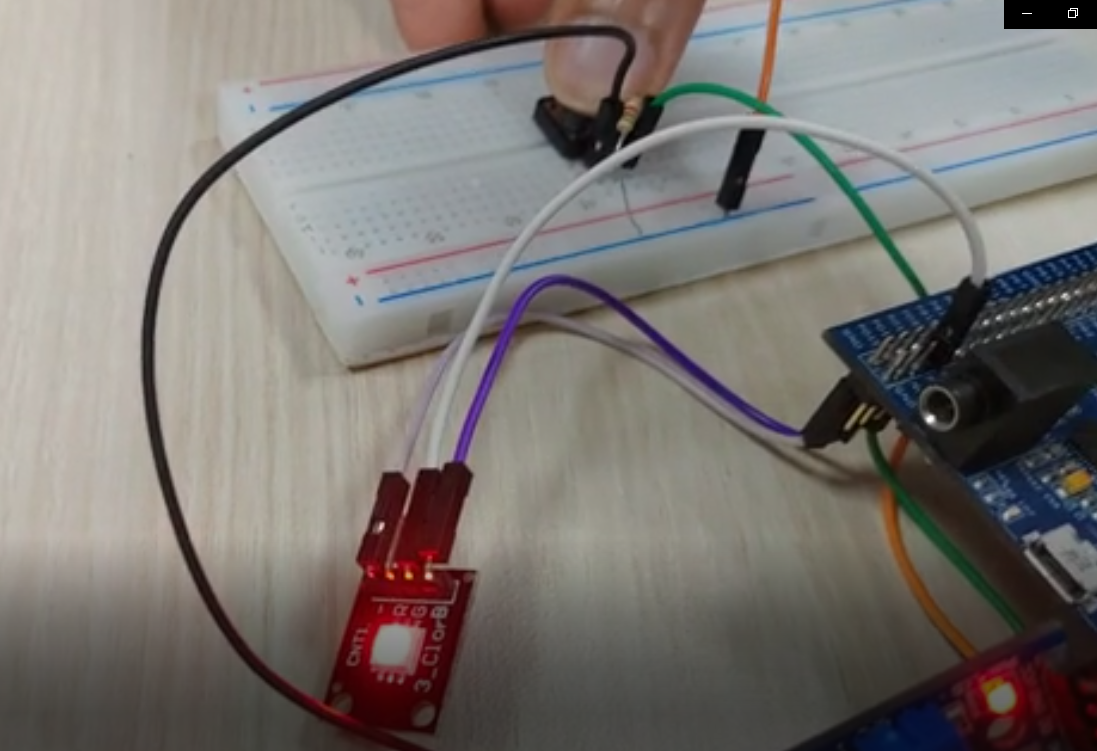


Figure Microcontroller implementation of hatch control

1. Exterior Lighting Using LDR

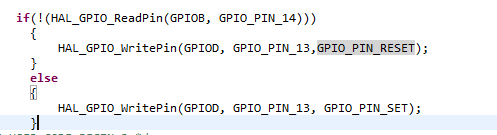


Figure Code snippet for Exterior Lighting

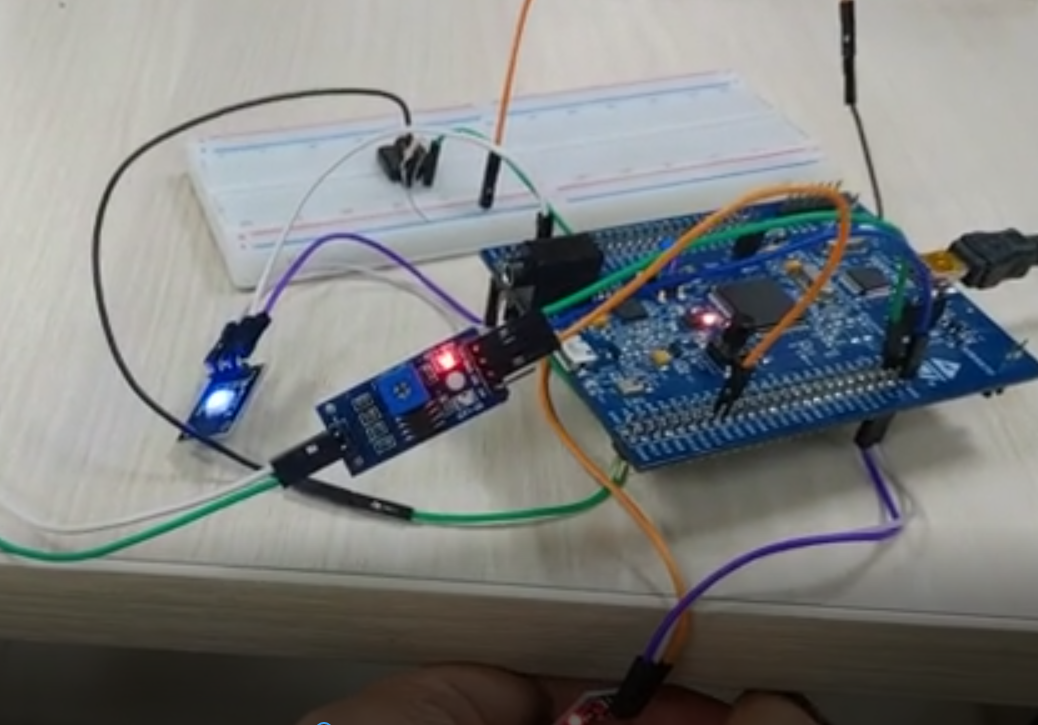


Figure Implementation of exterior lighting in Controller

1. Interior light switching using switch

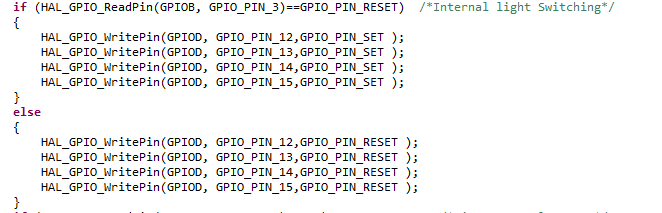


Figure Code snippet for Interior lighting

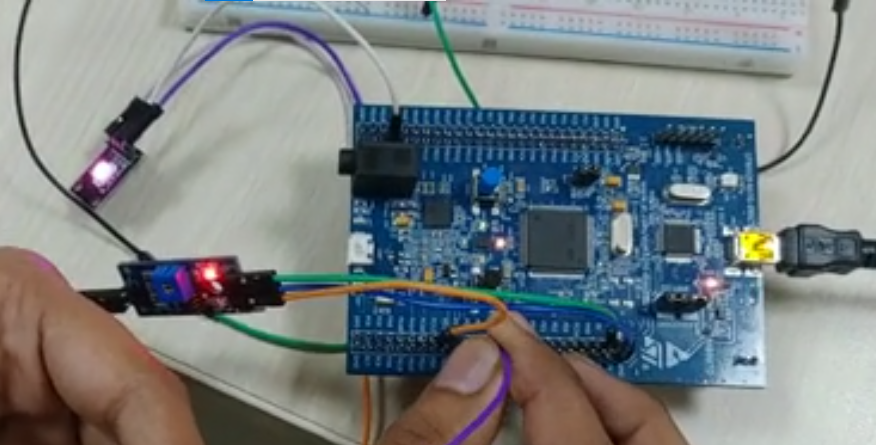


Figure Image of the microcontroller implementation of interior lighting(PinkLight)

1. Automatic wiper control

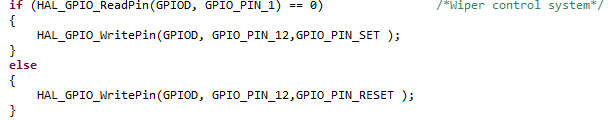


Figure Code snippet for Wiper control



Figure 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 Wiper system & Interior lighting.

### Summary

Outcomes:

1. STM32f407 discovery board programming.
2. Programming the features BCM module: Automatic Wiper System, Interior lighting

# Miniprojector -CANOE an CAPL Scripting[Team]

## Module linked to the CANOE, CAPL scripting

## Objectives & Requirements

Implementation of BCM features like door signal, cruise control and Dash board lighting in CANOE using CAPL scripting.

High Level Requirements:

1. Cruise control (99003729)
2. Door locking
3. Dash Board Lighting (99003722)

Low Level requirements for Door locking:

* Door locking:

Here the Door ECU will be receiving the signal from the Cruise control ECU and if the speed gets increased more than 15m/s^2 the doors will Lock automatically.

* In the Park mode (i.e. if the user presses P button and if there is no acceleration) the door unlocks the feature is taken from tesla manual.
* If the car is in acceleration and if the user wants to unlock the door while it’s in motion the user can’t unlock the door.
* If the Door is locked and if the user tries to open the door can’t be opened and in the same way if they user can open the door when lock is open.

## Design

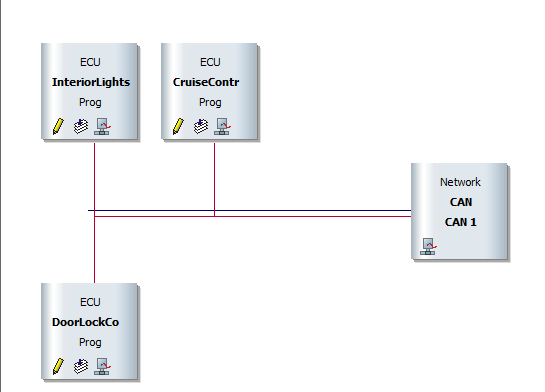


Figure CANOE implementation of BCM Feature



Figure Dashboard of CANOE Project for BCM feature

## Test Plan

|  |  |  |
| --- | --- | --- |
| **INPUTS** | **Expected Output** | **Actual output** |
| Press input “P” | Door should unlock | Door getting Unlocked |
| Press input “L” | Door should Lock | Door gets locked |
| Try to open door when its locked | Door shouldn’t open | Door doesn’t open |
| when speed>15m/s^2 | Car doors should get locked | Car doors locked |

Table Test Plan for CANOE BCM feature

## Implementation Summary

Firstly, as a team we chose a recent car manual and decided individual features which have interconnection between them and created a data base of messages and signals according to the features in the CANOE tool and added 3 Nodes connected to each other and each have contributed their CAPL scripting for the respective chosen ECU, after that we created PANELS for DOOR closing, Cruise control controller, Dash board lighting indication. So, after that we configured that panel to the CAPL scripting with the system variables and added the system variable sin the CAPL scripting to make the functionality working properly.

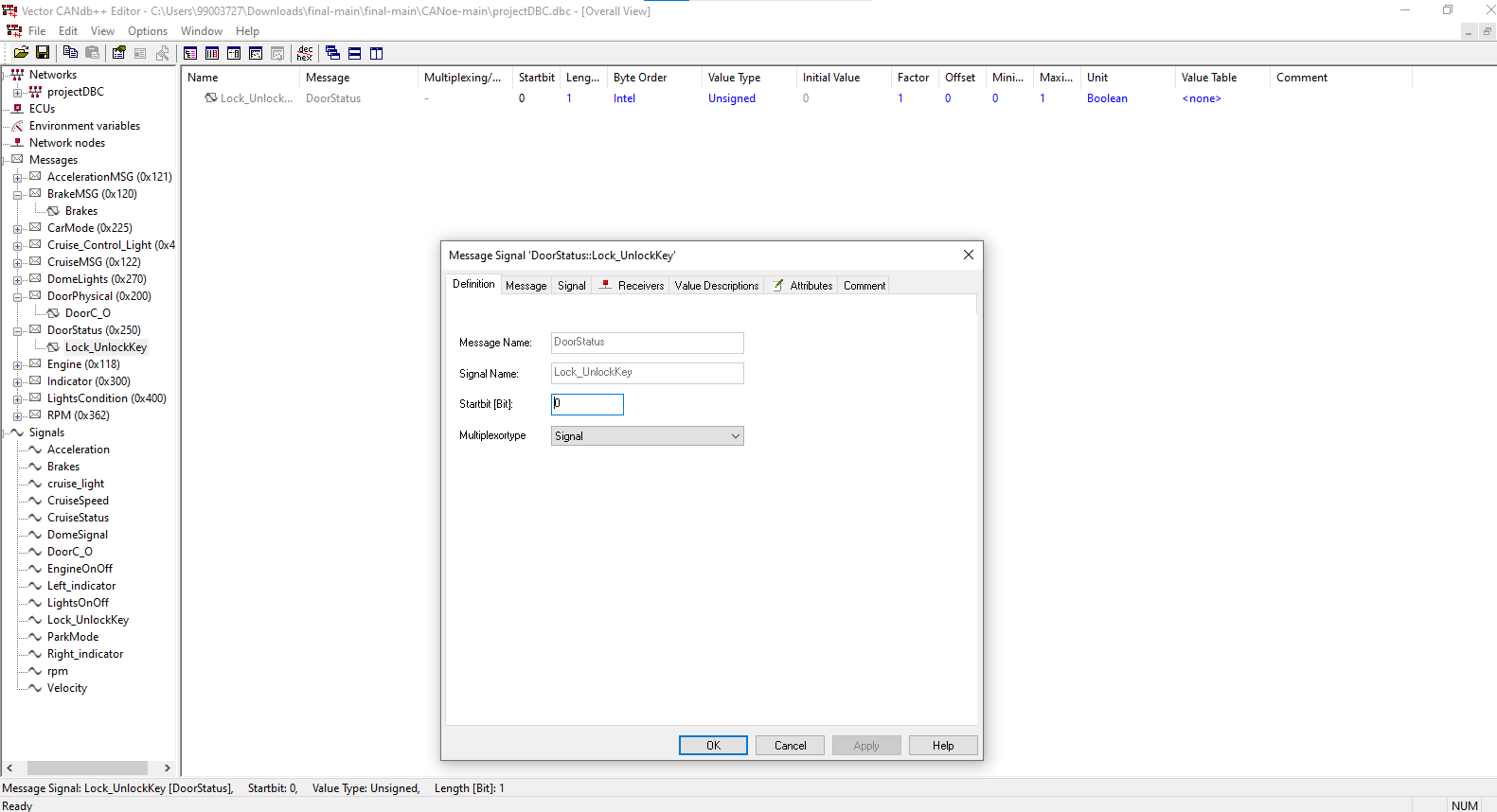


Figure Database with different messages and signals

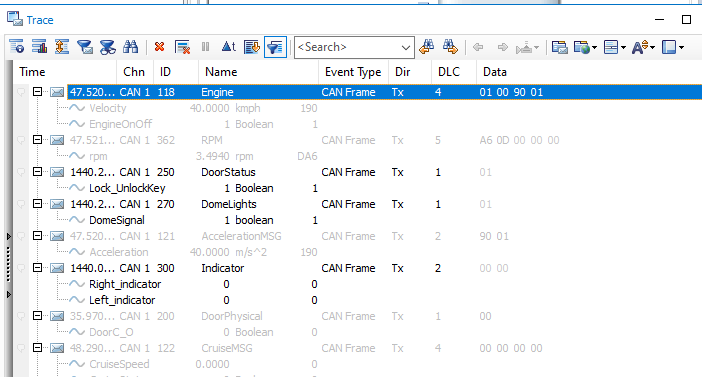


Figure Trace window where we can see the signal outputs

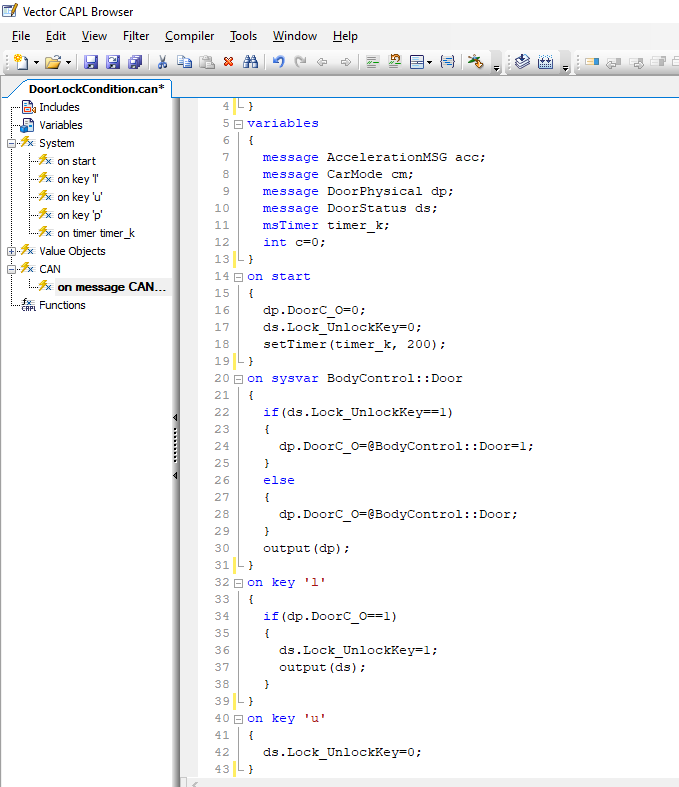


Figure CAPL scripting for Door

## Individual Contribution & Highlights

Door locking and unlocking, Door opening and closing has been implemented in the CAPL script. Features I chose are PARK mode-the doors should unlock in this mode, CAR doors should get locked if the speed of the vehicle increases more than 15m/s^2, CAR DOOR should not open if the door is in lock condition.

### Challenges faced and how were they overcome

* CAPL scripting was difficult to write and overcame this problem by referring the CAPL programming manual.
* Creating the panels was bit difficult and assigning it to the system variables was little hard to understand then searched in CANOE help and Googled some information like how to proceed.

# Miniproject -Evolved MBSE with HIL testing [Team]

## Module involved in Evolved MBSE with HIL testing are MATLAB, SIMULINK, STATEFLOW

### Topic and Subtopics

* HIL testing
* MAAB GUIDELINES
* Research
* Model Advisor
* Modelling Techniques
* MC/DC coverage
* SIL testing

## Objectives & Requirements

Implementation of BCM features like Cruise control, Hatch control, AC control, Sunroof control and Wiper control in MATLAB using Simulink and State flow and take one feature from the previous mentioned features to do HIL testing in ADRUINO UNO microcontroller board.

High Level Requirements:

|  |  |
| --- | --- |
| ID | Description |
| H\_CC\_01 | Cruise control subsystem- The Cruise system was introduced to reduce the driver fatigue for long drives. It is a closed loop control systems. |
| H\_HD\_02 | Opening of the back door through physical switch and proximity sensor |
| H\_SUN\_03 | Opening & closing Sun roof |
| H\_FW\_04 | Enabling front wind shield wiper |
| H\_RW\_05 | Enabling rear wind shield wiper |
| H\_AC\_06 | Main requirement is to provide comfort for the people on the vehicle cabin. This feature includes the process of controlling the temperature based on the user comfortable |

Table High level requirements of evolved MBSE

Low Level Requirements:

|  |  |
| --- | --- |
| ID | Description |
| L\_CC\_01 | Cruise control should turn on only if vehicle speed is >40kmph. |
| L\_CC\_02 | It should hold the vehicle speed at the selected value. |
| L\_CC\_03 | Hold the speed with minimum surging. |
| L\_CC\_04 | Allow the vehicle to change speed. |
| L\_CC\_05 | Deactivate the control immediately after the brakes are applied. |
| L\_CC\_06 | Store the last set speed. |
| L\_HD\_07 | Stopping of the door through a button located on the bottom such that the servo motor stops at certain angle with the pulse width. |
| L\_HD\_08 | Servo motor should go extreme right when there is no handle door button pressed. |
| L\_HD\_09 | Servo motor should go extreme left when the user presses the stop button on the bottom of the door. |
| L\_HD\_10 | Driver can close and open the back door from the door switch which is next to him. |
| L\_AC\_11 | AC should turn on whenever the user turns on the switch |
| L\_AC\_12 | Based on the temperature set sensor should sense and maintain according to that. |
| L\_AC\_13 | User can change the speed of the based on his requirement |
| L\_AC\_14 | If the temperature is set then fan only will work. |
| L\_AC\_15 | User can change the direction of airflow. |
| L\_AC\_16 | User can turn off the AC. |
| L\_SUN\_17 | Operating motor used for sunroof panels |
| L\_LS\_18 | Enabling low speed wiper |
| L\_HS\_19 | Enabling High speed wiper |
| L\_MW\_20 | Enabling mist condition wiper |

Table Low level Requirements of Evolved MBSE

## Design

High Level design

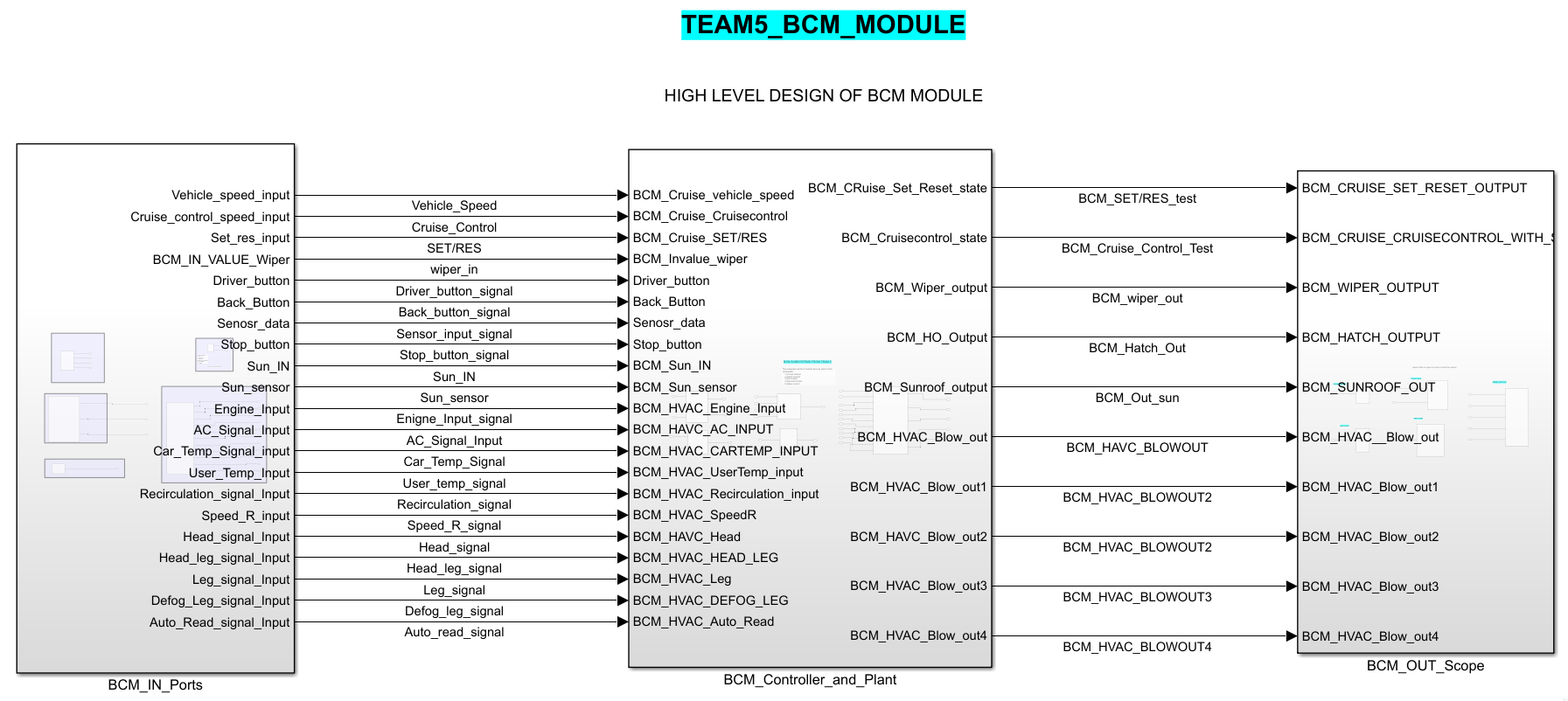


Figure High level Design of BCM features of MBSE

## Low Level Design of Hatch Control(INDIVIDUAL):

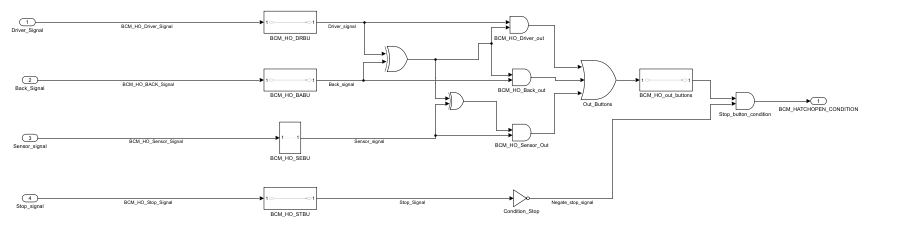


Figure Low level design of Hatch opening of Evolved MBSE

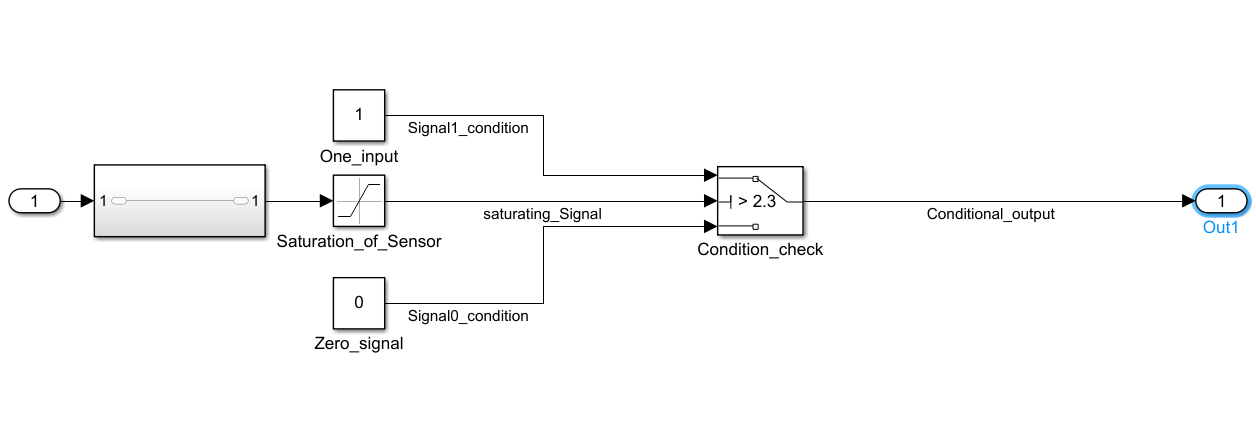


Figure Hatch opening sensor conversion to digital

## Test Plan

High Level Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Description | Input given | Output expected | Actual output | Type of Test | Result |
| HLT\_CR\_01 | A speed of 50kmph with cruise control switch in On state is given as input. | 50, Cruise On | 50 | 50 | Requirement based | Pass |
| HLT\_SUN\_02 | Open push button pressed | 16v | Sunroof open | Sunroof open | Requirement based | Pass |
| HLT\_SUN\_03 | Close push button pressed | 16v | Sunroof close | Sunroof close | Requirement based | Pass |
| HLT\_FW\_04 | Enabling front wind shield wiper | Front Wiper should be on | Front Wiper On | Front Wiper On | Requirement based | Pass |
| HLT\_RW\_05 | Enabling rear wind shield wiper | Rear Wiper should be on | Rear Wiper On | Rear Wiper On | Requirement based | Pass |
| HLT\_AC\_06 | To check whether ignition condition | Ign =0  Ac on =0 | Ac off | Ac off | Scenario based | Pass |
| HLT\_AC\_\_02 | When ignition on  To check the ac button condition | Ign = 0  Ac on =0 | Ac off | Ac off | Scenariobased | Pass |
| HLT\_AC\_03 | When ignition on  To check the ac button condition | Ign =0  Ac on =1 | Ac on | Ac on | Scenario based | Pass |
| HLT\_HD\_1 | A Button in the driver door and the sensor input should open the hatch at the back | DriverButton=0,  Sensordata=1 | Hatch open | Hatch  open | Scenario based | Pass |

Table High level Requirement Test plan of Evolved MBSE

Low level Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Description | Input given | Output expected | Actual output | Type of Test | Result |
| LLT\_HD\_01 | A Button on the back-door handle is pressed | Back Button=1 | Hatch opens | Hatch open | Scenario based | Pass |
| LLT\_HD\_02 | A Stop button on the bottom of the door should be pressed. | Stop Button=1 | Hatch closes | Door opening should be closed | Scenario based | Pass |
| LLT\_HD\_03 | A button on the back-door handle is pressed | Back door button=1 | Hatch opening  Should pause | Can’t be seen as motor is required | Scenario based | Failed |
| LLT\_HD\_04 | A Button on the bottom of the door should be pressed after pressing the back-door handle button | Bottom Button=1 | Hatch should close | Hatch closes | Scenario based | Pass |
| LLT\_HD\_05 | Sensor should detect the object and hatch should open | Sensor input=2.6 | Hatch open | Hatch open | Scenario based | Pass |
| LLT\_HD\_06 | Sensor should detect the object and hatch should open | Sensor input=2.1 | Hatch should not open | Hatch will not open | Boundary based | Pass |

Table Low level test plan of Hatch control Evolved MBSE

## Implementation Summary

* The whole model is designed in MATLAB R2020b.
* Simulink and state flow were used in building this project

Algorithm for Hatch control:

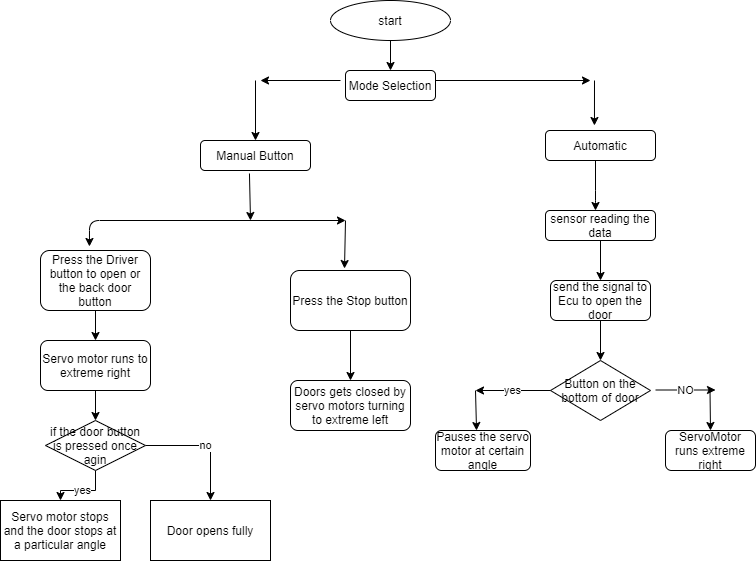


Figure Algorithm for Hatch control low level requirement of evolved MBSE.

Some research work is done before implementing all the features in the MATLAB like what sensors must be used and how the logic works in the real time so after doing some research I found the sensor for hatch opening using leg as an input to open the hatch uses proximity sensor so for this I have chosen optical proximity sensor and took the characteristics of that sensor and implemented the model using that sensor features.

Characteristics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proximity Sensors | range | I/P range | O/P response | O/P current |
| Capacitive | 7.2mm-8.8mm(<1cm) | <=200mA | <35ms | <0.3mA |
| Optical | 50mm-100mm(1cm-10cm) | (2.3 V to 3.6 V)  Response(200ms) | <35ms | <0.3mA |
| Inductive | 30mm | <10ms&(10v-36v) | <=0.3mss | <0.3mA |

Table 8 Proximity sensor comparision for Hatch control of Evolved MBSEomparison

References:

* <https://www.onsemi.cn/pdf/datasheet/noa3302-d.pdf>
* <https://www.gavazzionline.com/pdf/CA12CLC0.pdf>
* <https://docs.rs-online.com/e8de/0900766b80274a6e.pdf>

Sensor chosen is Optical proximity sensor NOA3302 because the other sensors do need metal objects to get detection and the other sensors work only for short range of distance.

A servo motor is used to open the door and works with 3 different pulse widths to 3 different ranges 0to90degrees, 90to180degrees, in between 0 to 180 with milliseconds duration.

MAAB guidelines are also used in implementing the HATCH Control in order to make someone to understand the model easier. Some of the Guidelines which I followed are given below.

1. na\_0008: Display of labels on signals
2. na\_0002: Appropriate implementation of fundamental logical and numerical operations
3. jc\_0141: Use of the Switch block
4. db\_0116: Simulink patterns for logical constructs with logical blocks
5. jc\_0201: Usable characters for Subsystem

## Individual Contribution & Highlights

Hatch opening is the individual contribution and I have converted the sensor data to digital data using a logical block and gave a protection to the sensor using saturation block.

## Summary

BCM features are implemented in the MATLAB Simulink and has been tested with different test cases and after completing the design we took single feature (HATCH CONTROL) and did the HIL testing in ADRUINO and tested the feature by using potentiometer for sensor data and varied it and got the expected output.

### Challenges faced and how were they overcome

Research work is little difficult to do and modelling the sensor data to digital and protecting the sensor against the high voltages was quite bit difficult but by trial and error I have designed the model.

# Miniproject -Control System [Individual and Team]

## Modules present in this project is Control System.

### Topic and Subtopics

* 1st order differential equation
* 2nd order differential equation
* Laplace transform
* State space
* Frequency domain analysis
* Time domain analysis
* Plant
* Controller
* Open loop control system
* Closed loop control system
* Initial Value theorem & Final value theorem

## Objectives

Objective of this Project is to understand how to analyze the system given to us and use different controllers to make the system stable if the system is unstable. It’s about learning how to analyze the given roots, where the roots are, How the roots behave when they are near, far from the imaginary axis. So, in this module I took an Individual system named thermometer (1st order system) and analyzed it. After analyzing the individual system, we took a one activity where Different motors are controlled with PID controller and PWM controller and should say which motor gives the stable response with the PID controller.

## Design

In designing the systems, I have used Scripting for individual and Simulink for team activity. So, in the individual I wrote a script for the first order equation and got the step response, Impulse response and plotted pole, zeroes of the thermometer and analyzed it for open loop and closed loop systems with and without controller.

For the Team activity, we took Induction motor, Permanent magnet synchronous motor, Brush less DC motor and 2 controllers named as PID, PWM.



## Test Plan

## Implementation Summary

### Git Link

### Git Dashboard

### Summary

#### Git inspector summary

#### Build

#### Code quality

#### Unit Testing

#### Issues

## Individual Contribution & Highlights

### Summary

### Challenges faced and how were they overcome