./

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](#_Toc69485049)

[CALCULATOR [Team] 4](#_Toc69485050)

[Module/s 4](#_Toc69485051)

[Topic and Subtopics 4](#_Toc69485052)

[Objectives & Requirements 4](#_Toc69485053)

[Design 6](#_Toc69485054)

[Test Plan 7](#_Toc69485055)

[Implementation Summary 8](#_Toc69485056)

[Video Summary 8](#_Toc69485057)

[Git Link 8](#_Toc69485058)

[Git Dashboard 8](#_Toc69485059)

[Git Inspector Summary 9](#_Toc69485060)

[Git Unit Testing Summary: 9](#_Toc69485061)

[Code Quality and Valgrid check: 10](#_Toc69485062)

[Individual Contribution & Highlights 10](#_Toc69485063)

[Challenges faced and how were they overcome 10](#_Toc69485064)

[Future Scope (If applicable) 10](#_Toc69485065)

[Miniproject - Finding Words in a Document and Printing it in a new File [Individual] 11](#_Toc69485066)

[Module: 11](#_Toc69485067)

[Topic and Subtopics 11](#_Toc69485068)

[Objectives & Requirements 11](#_Toc69485069)

[Test Plan 12](#_Toc69485070)

[Implementation Summary: 12](#_Toc69485071)

[Git Link 14](#_Toc69485072)

[Summary 14](#_Toc69485073)

[Challenges faced and how were they overcome 14](#_Toc69485074)

[Miniproject -2 [Team/Individual] 15](#_Toc69485075)

[Module/s 15](#_Toc69485076)

[Topic and Subtopics 15](#_Toc69485077)

[Objectives & Requirements 15](#_Toc69485078)

[Design 15](#_Toc69485079)

[Test Plan 15](#_Toc69485080)

[Implementation Summary 15](#_Toc69485081)

[Git Link 15](#_Toc69485082)

[Git Dashboard 15](#_Toc69485083)

[Summary 15](#_Toc69485084)

[Individual Contribution & Highlights 15](#_Toc69485085)

[Summary 15](#_Toc69485086)

[Challenges faced and how were they overcome 15](#_Toc69485087)

# 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

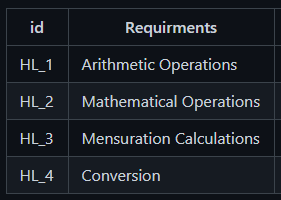
1. Took state structural model and class behavioral model for the low-level requirement and Deployment behavioral model for the High-level requirement.

* Testing:

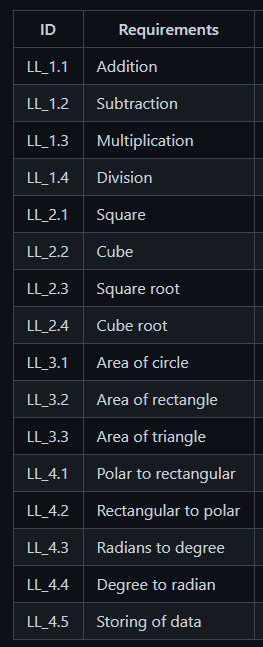
1. Testing types functional testing and nonfunctional testing and learnt that Nonfunctional testing has no impact on the features.
2. In the functional testing, we have different types namely Boundary value test, Requirement based test, Scenario based test.

## 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”

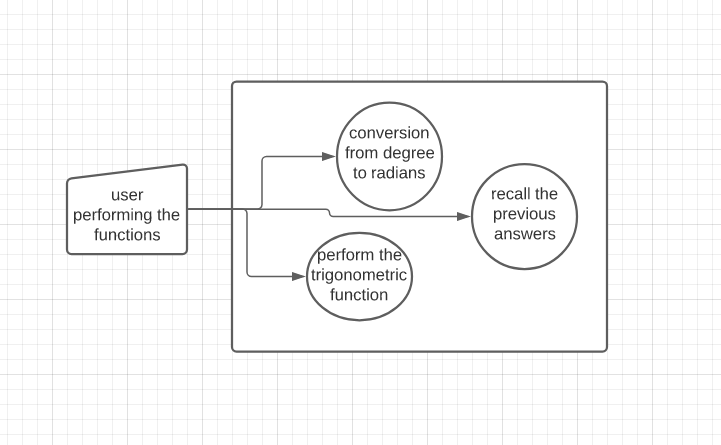
****

Fig1. High Level Requirement of conversion from degree to radians and vice versa

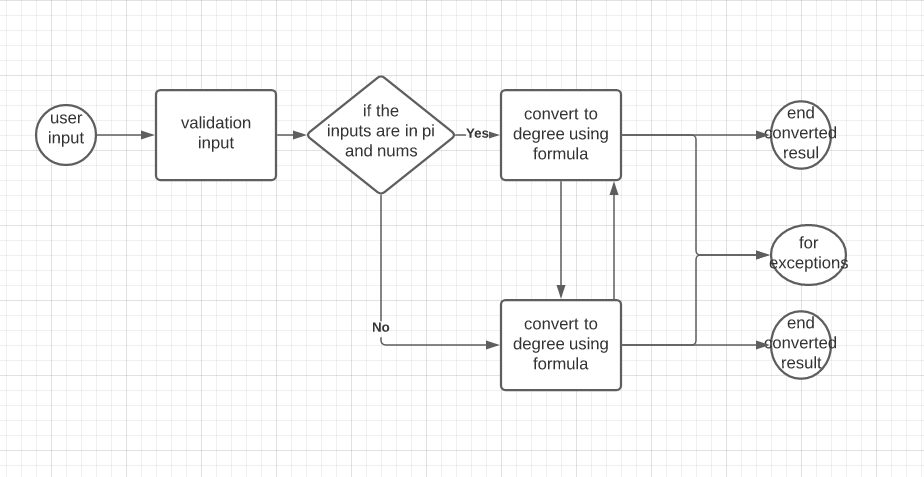
****

Fig2.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”



Fig3. 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 |

Table1. 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

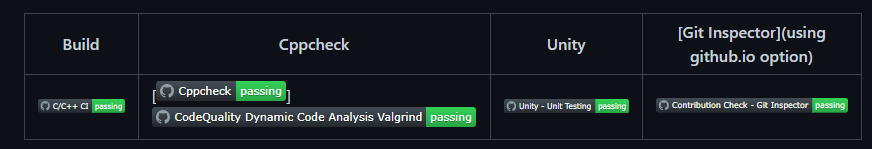


Fig4.Git Badges

Git Inspector Summary:

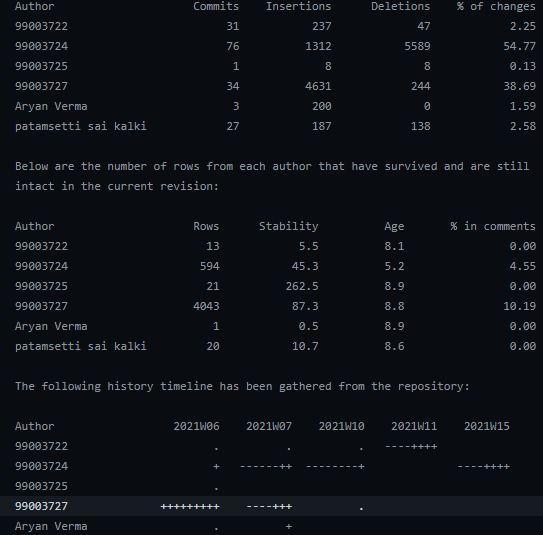


Fig5.Git Inspector Summary

## Git Unit Testing Summary:

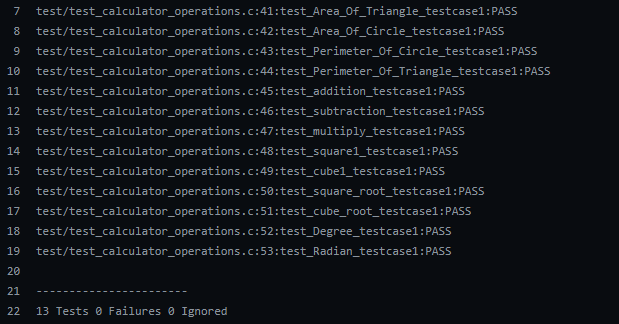


Fig6.Git Unit testing summary

## Code Quality and Valgrid check:

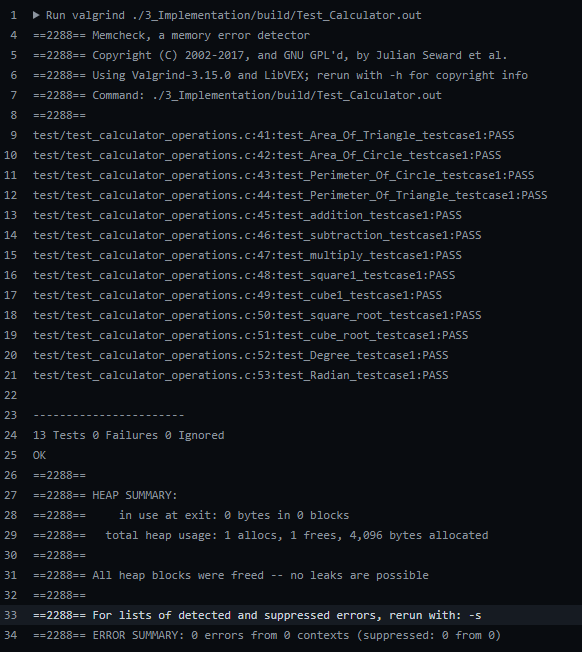


Fig7.Code quality Summary

## 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 |

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.

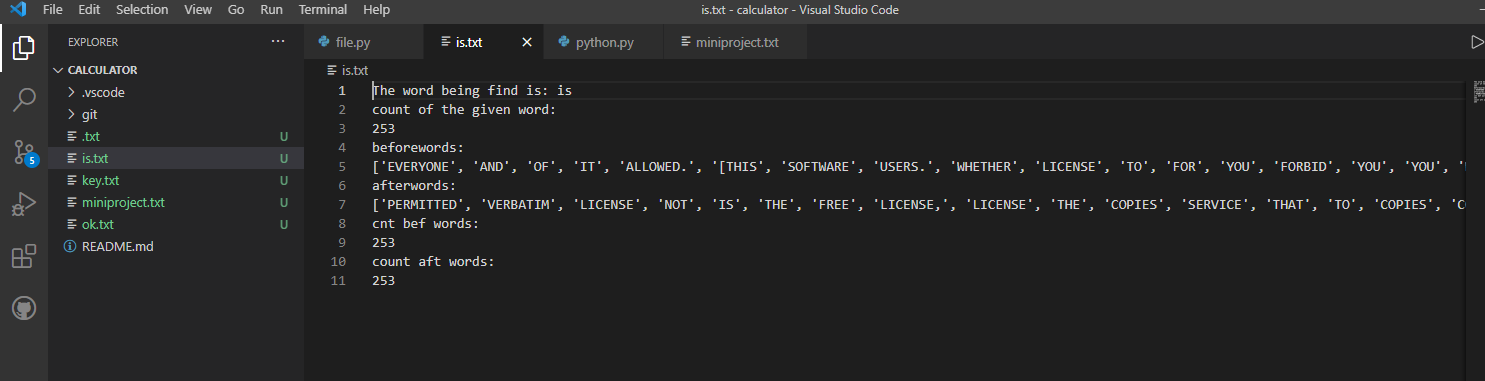


Fig8.Output of the code

## 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.

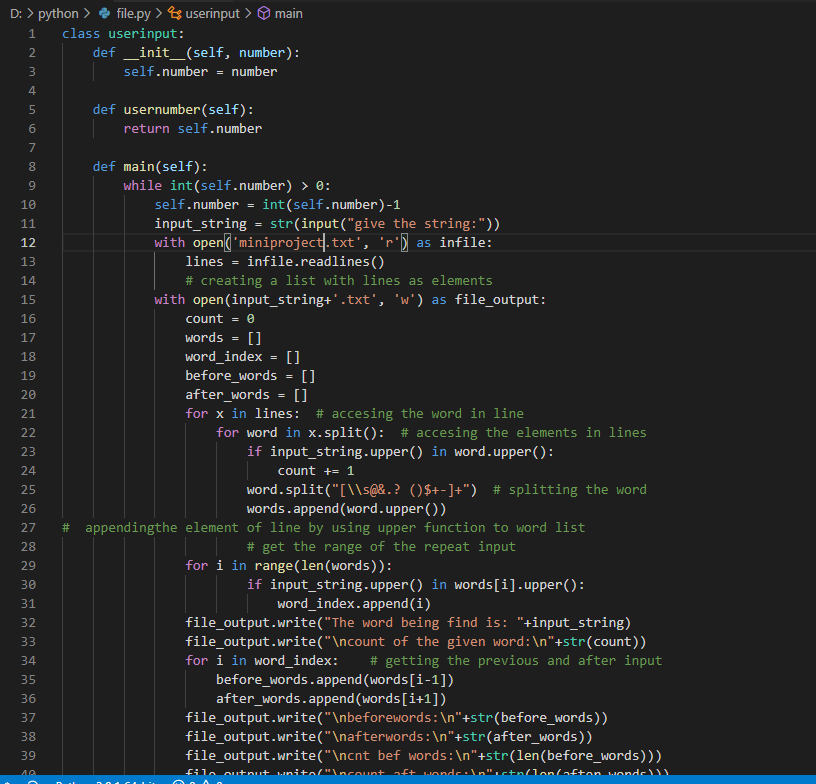


Fig9.Main code snippet

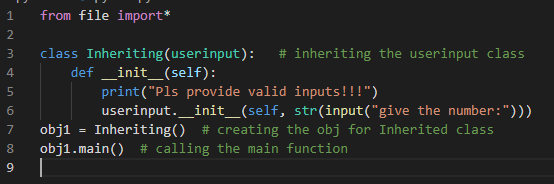


Fig10.Accesing the main file snippet

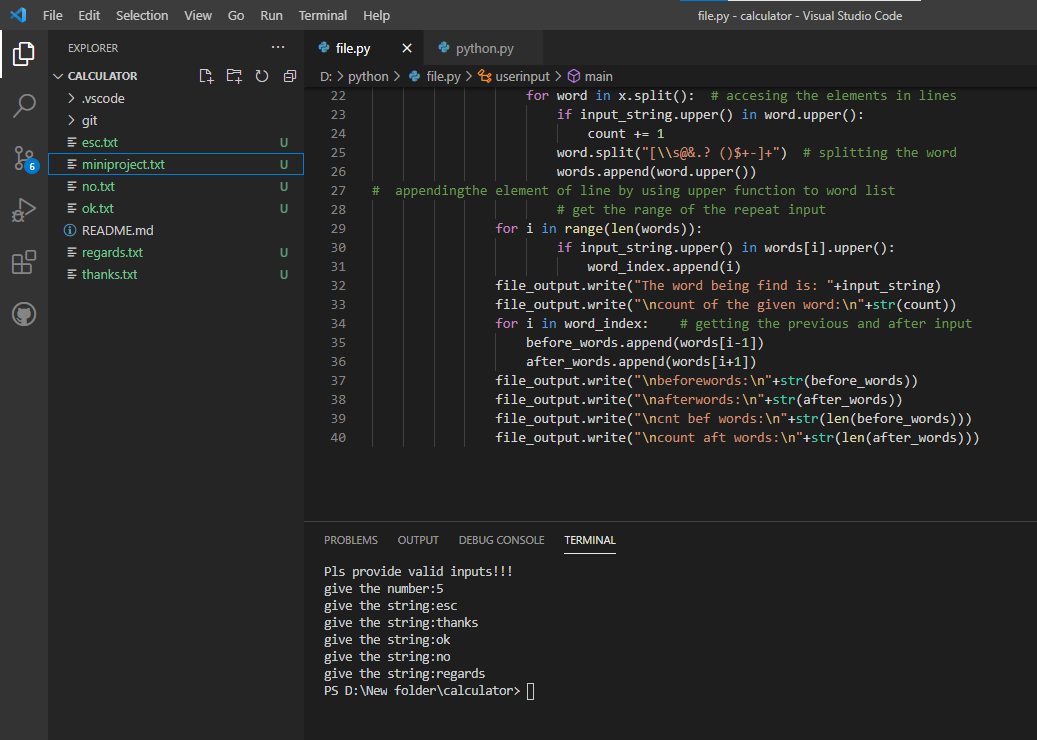


Fig11.Output files getting generated script for the given inputs snippet

### 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 file
* Start Up file
* Linker Script
* Debugging techniques
* 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. Hatch Opening
  2. Exterior lighting
  3. Automatic Wiper control
  4. Internal lighting

## Design

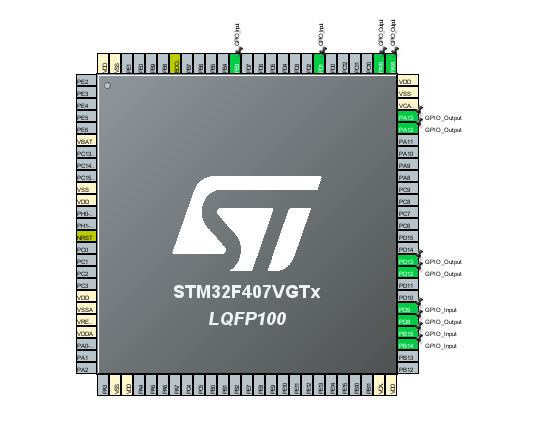


Fig12.PinConfigurations selected for the features

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

1. Hatch open

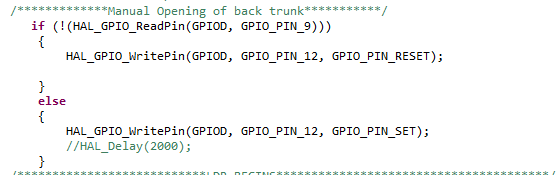


Fig13.Code Snippet for Hatch Control

1. Exterior Lighting Using LDR

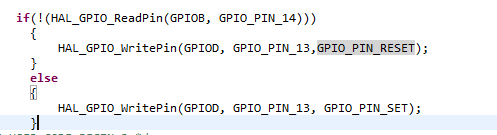


Fig14.Code snippet for Exterior Lighting

1. Interior light switching using switch

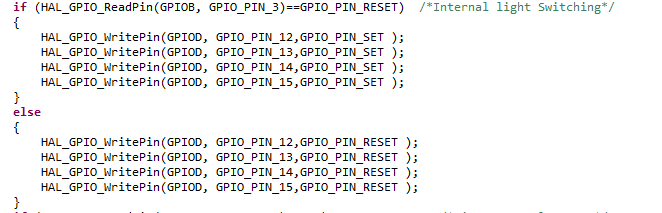


Fig15.Code snippet for Interior lighting

1. Automatic wiper control

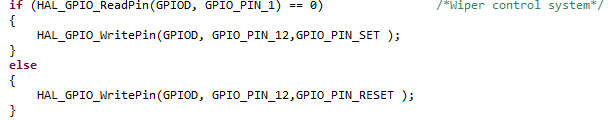


Fig16.Code snippet for Wiper control

## 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

# Miniproject -MBSE [Team]

## Module linked to the MBSE are MATLAB & Simulink.

## Objectives & Requirements

## Design

## 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