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
1	20-4-2021	Kopparapu Jyothi Swaroopa Rani			

Table of Contents

Miniproject -1 [Team]

Module

SDLC and C Programming.

Topic and Subtopics

It is almost impossible for us to imagine mathematics without a calculator. An electronic calculator is typically a portable electronic device used to perform calculations ranging from basic arithmetic to complex mathematics. Calculators are used in a comprehensive mathematics curriculum to increase the quality of student learning experience. Without calculators, advanced math courses, such as Calculus, would require much longer time to solve. The calculators we know today were not invented until the 1970s, and the use of smartphones as calculators did not begin until at least the late nineties.

DIFFERENT TYPES OF CALCULATORS

ABACUS:

Abacus is the first tool created specifically for use in mathematical computations. It was invented by Sumerians and Egyptians in 2500 BC. The abacus is a table of successive columns with beads or stones representing a single unit, which could be used for addition or subtraction.

CONS:1. It is not useful for multiplication or division.

PASCAL CALCULATOR:

Pascal Calculator was invented in 1642 by a French inventor and mathematician Blaise Pascal. It performed calculations through a clockwork-type of mechanism and was lauded for attempting arithmetic calculations which was previously thought impossible.

CONS:

1. Production of these type of calculators was difficult.
2. Bulky in size.

MECHANIC CALCULATOR:

Curt Herzstark invented the first handheld, mechanical calculator in 1945. In 1970, a company in Japan invented the first digital pocket calculator. Companies like Texas Instruments adapted the design of the Japanese device and enhanced it by creating the graphing calculators we know today.

SCIENTIFIC CALCULATOR:

The first scientific calculator was invented in 1968. The HP-35, introduced on February 1, 1972, was the first pocket calculator and the world's first handheld scientific calculator. Texas Instruments (TI), after the production of several units with scientific notation, introduced a handheld scientific calculator on January 15, 1974. TI-30 series is one of the most widely used scientific calculators in classrooms.

SMARTPHONE:

With the invention of the first smartphone in 1995, individuals began to replace expensive digital calculators with the multi- use device. This required even the most sophisticated calculator designs to be upgraded to remain relevant in the market.

4W & 1H

WHAT?

Calculators are devices that are designed to do simple to complex calculations. Simplest calculators can do arithmetic operations like addition, subtraction, multiplication, and division. While sophisticated calculators can handle exponential operations, roots, logarithms, solve quadratic equations, trigonometric functions, hyperbolic functions, etc.

WHERE?

Calculators are tools that can be used by anyone, can also be a tool for learning mathematics when used appropriately. Calculators are used for educational to business purposes. Calculators benefits students from kinder garden to University level.

WHEN?

Calculators can be used when we find it difficult to solve a problem, solve complex calculations, etc. Since they are very easy to carry, we can make use of calculator anytime. Calculators can be used to crosscheck the result that we obtained.

WHY?

Calculators can provide much more precise, accurate results without any error when compared to the calculations made by user. Calculators are designed in such a way that anybody can make use of it easily.

HOW?

Calculators are designed in such a way to make user extremely easy to calculate, ergonomic, small, etc. The operations can be selected by pressing the respective buttons provided on the calculator, etc. are some of the symbols that are printed on the calculators extending from simple to scientific.

SWOT ANALYSIS:

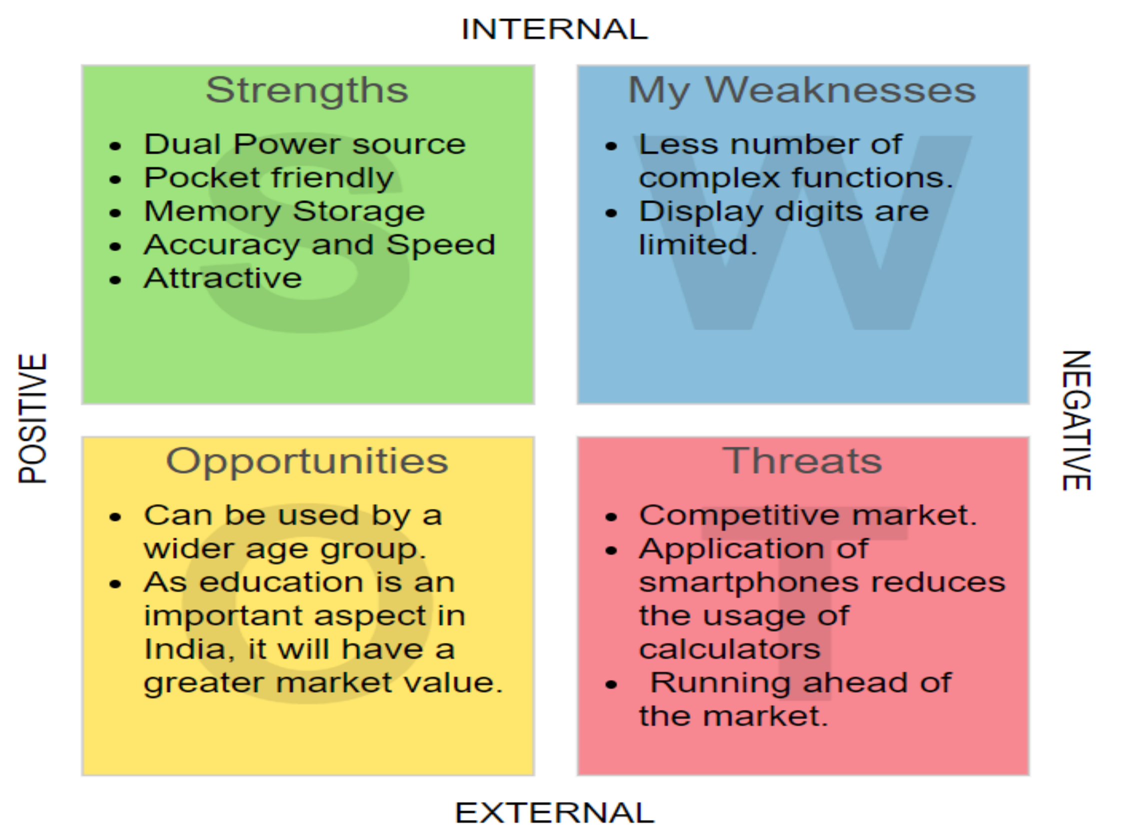

Figure 1:SWOT Analysis of a Calculator

Objectives & Requirements

HIGH LEVEL REQUIREMENTS

ID	Description
HLR_1	Arithmatic Operations
HLR_2	Trigonometric Operations
HLR_3	Logarithmic Operations and cube root
HLR_4	Mathprint
HLR_5	Roots & Power
HLR_6	Memory Storage
HLR_7	Binary to Decimal
HLR_8	Complex Numbers

Table 1:High Level Requirements of Calculator

LOW LEVEL REQUIREMENTS

Requirement	Description
Binary to Decimal Conversion	Take the input in form of only 1s and 0s as long through keypad and accordingly give the output as an int.
Root and Power	Take input as a double and find the square root and give output in double type. To find power take inputs as int for both number and the
Arithmetic Operations	
Addition	Input validation:check the ASCII value range of the user input numbers.Input type: integer, float.
	Operation: Take two inputs from the user and check the data type.
	If the inputs are in float data type; the results will be in floating point.If the inputs are in integer data type;
	the result will be in integer data type.If the inputs are in combination of integer as well as floating type;
	then the result should be in floating type.
Subtraction:	Input validation: check the ASCII value range of the user input numbers.Input type: integer, float.
	Operation: Take two inputs from the user and check the data type.Sign of both the input values must be considered and accordingly the
	result should be in floating type.
Multiplication:	Input validation: Check the ASCII value range of the user input numbers. Also check the sign of the user input numbers.
	Input type: integer, float. Operation: Take two inputs from the user and check the data type.Sign of both the input values must be
	considered and accordingly the multiplication operation must be performed. If the inputs are in float data type; the results will be in
	floating point.If the inputs are in integer data type; the result will be in integer data type.If the inputs are in combination of integer
	as well as floating type; then the result should be in floating type.
Division:	Input Validation: Check the ASCII value range of the user input numbers.Also check the sign of the user input numbers.
	Divide by zero is not possible. Input type: integer, float.Operation: Take two inputs from the user and check the data type.
	Sign of both the input values must be considered and accordingly the division operation must be performed.
	If the inputs are in float data type; the results will be in floating point.If the inputs are in integer data type; the result will be in
	are in combination of integer as well as floating type; then the result should be in floating type.
Memory Storage	A history button is created which shows the last five stored results.When the user hits the HISTORY button it will display the last five
	stored value. Operation: Works with arrays.
Complex mode	It is used to calculate with the real and imaginary numbers in a single mode.
Math Print	It takes the input, calculates it and shows the result as well as the input together.
Trignometric Functions	Finding values for sin(), cos(),tan(),sec(),cosec(),cot() functions.
Exponential functions	To perform logarithmic functions and cube root functions.

Table 2:Low Level Requirements of Calculator

Design

HIGH LEVEL DESIGN:
STATE MACHINE

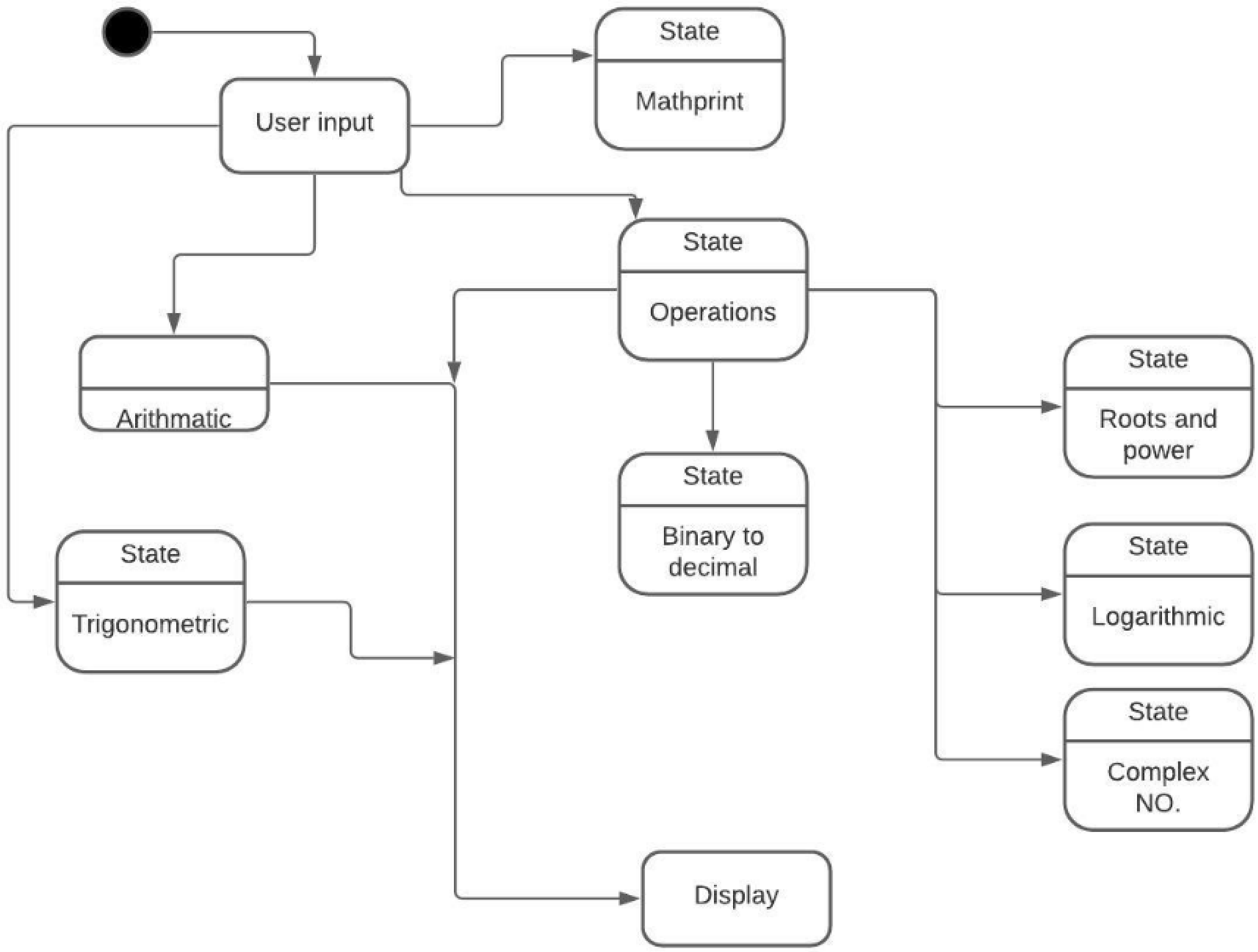

Figure 2:High Level-State flow diagram

CLASS

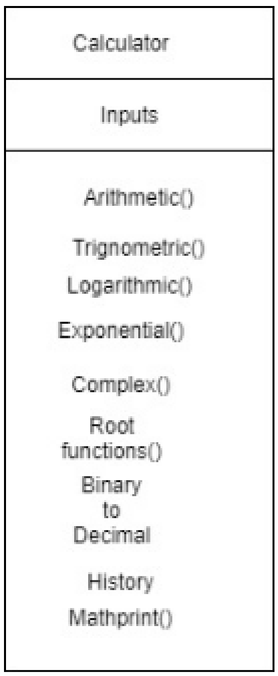

Figure 3:High Level-Class Diagram

LOW LEVEL DESIGN
DEPLOYMENT DIAGRAM

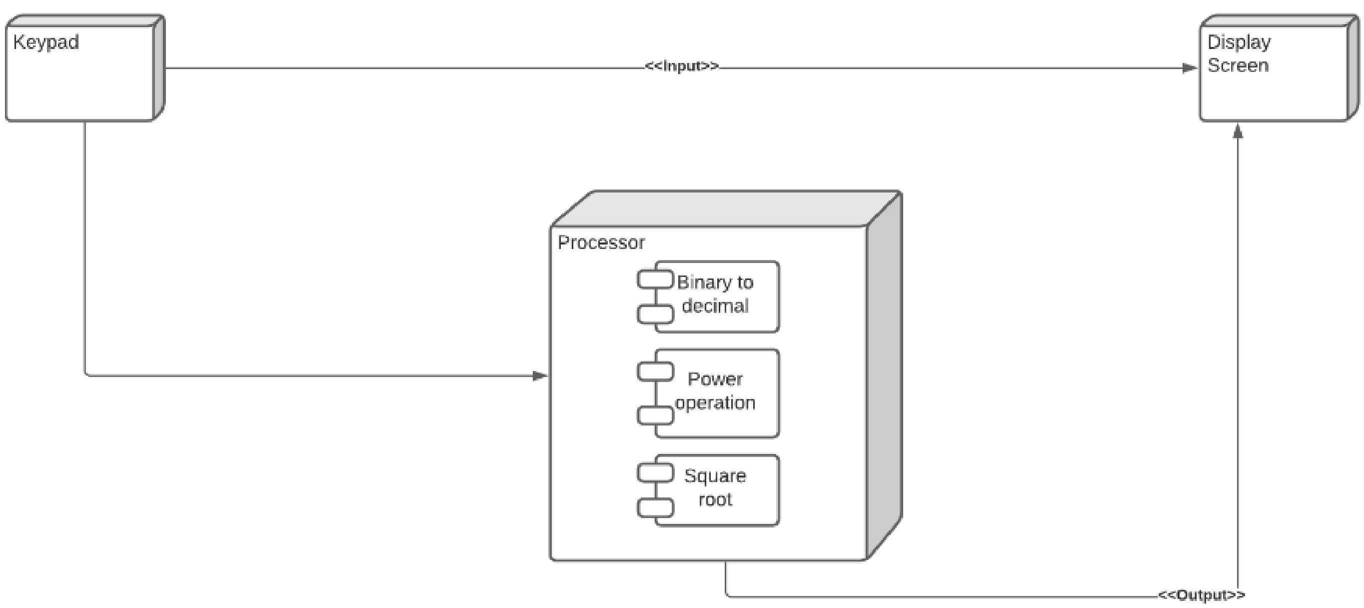

Figure 4:Low Level Deployment Diagram

COMPONENT DIAGRAM

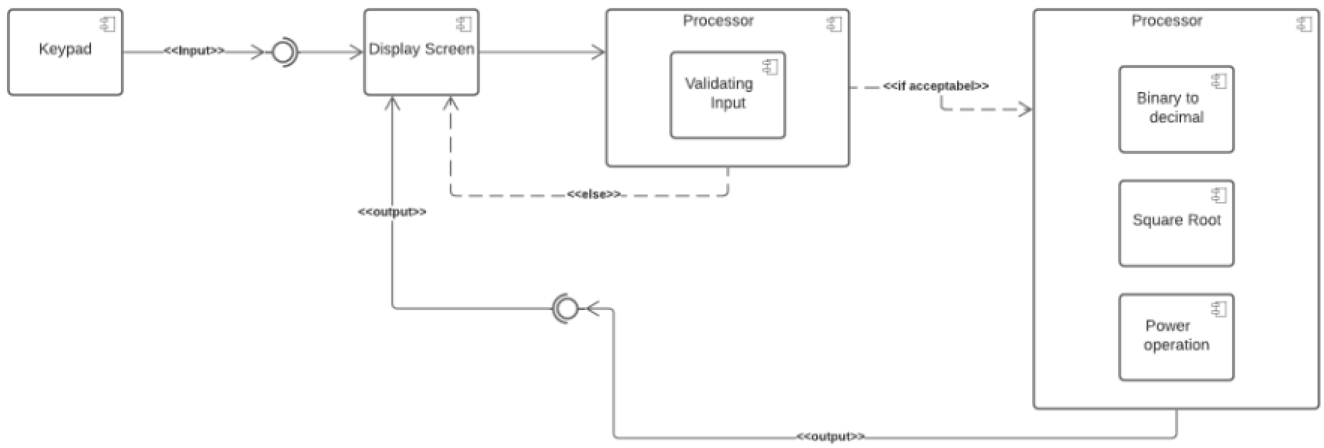

Figure 5:Low Level Component Diagram

USE CASE DIAGRAM

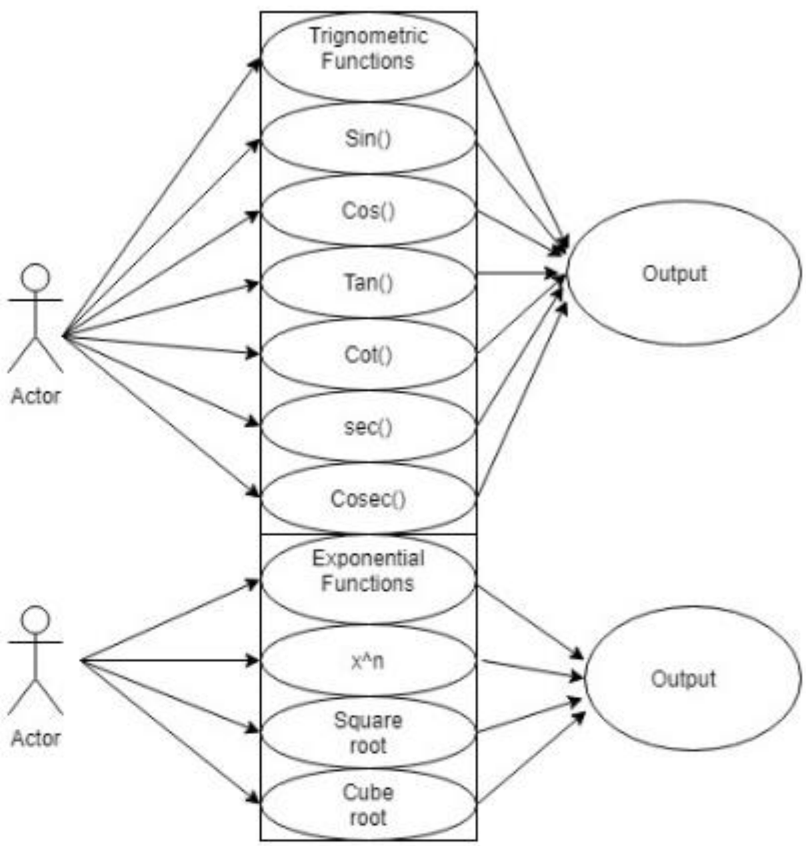

Figure 6:Low Level Use Case Diagram

OBJECT DIAGRAM

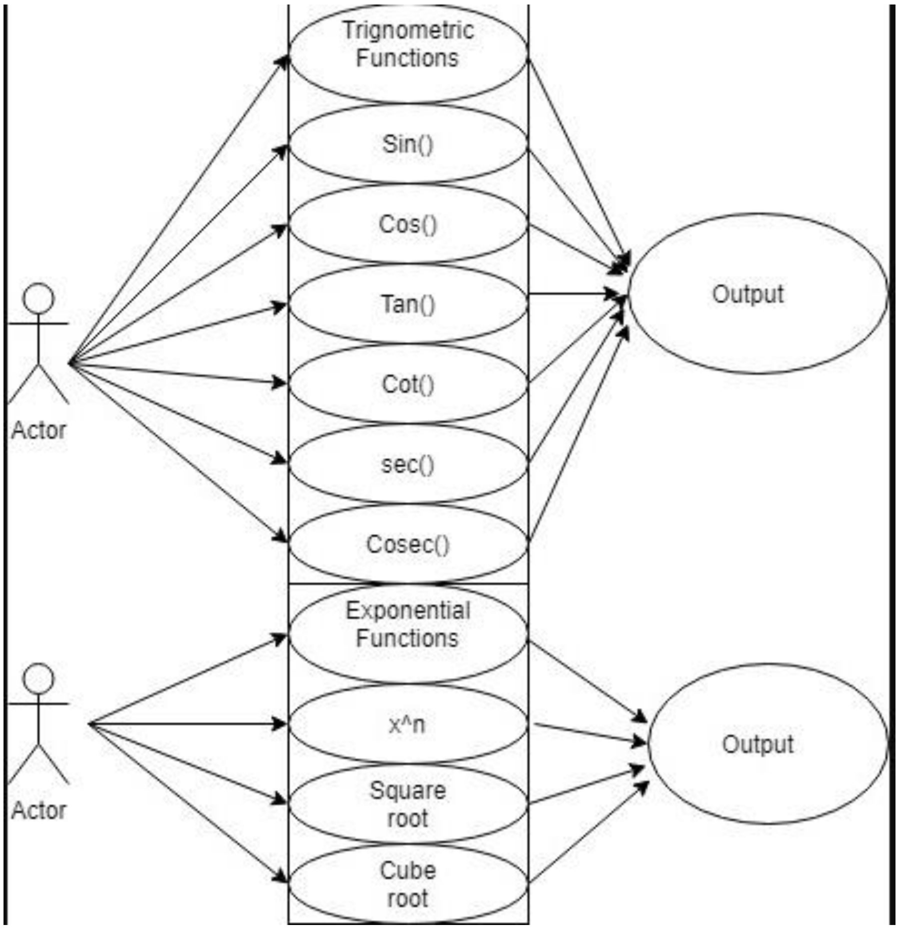

Figure 7:Low Level Object Diagram

SEQUENCE DIAGRAM

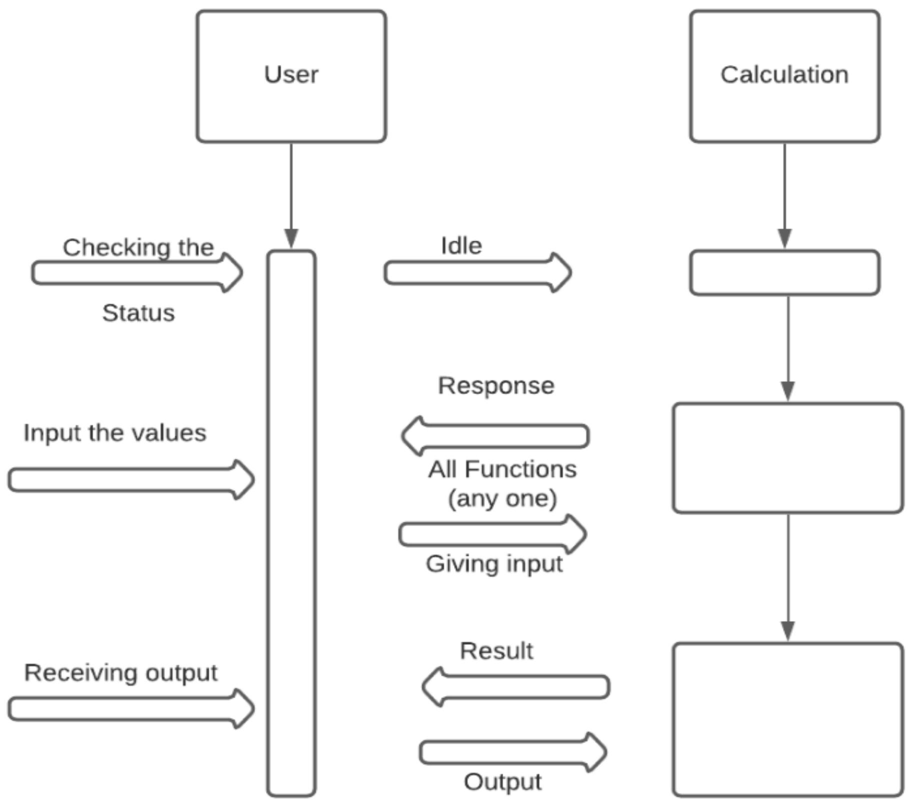

Figure 8:Low Level Sequence Diagram

STATE MACHINE

Figure 9:Low Level State Machine Diagram

Test Plan

TEST PLAN

Test_id	Description	Expected input	Expected Output
LLR_1 _ Arithmetic Operations	It contains all the basic arithmetic operations		
	Addition:1)The user input must be validated. The sign of the user input must also be validated.	Integer, integer	Integer, integer
	2) The floating point input must provide a floating point results.	Float, Float	Float
	3) A combination of floating point input and integer input must provide a floating point output.	Float, integer Or Integer, float	Float
	4) If result exceeds by 14 digits	Input 1=10 digits Input 2=6 digits	Out of range
	5) If the first input is a negative number and second input is positive number or vice-versa.	Input1= -ve greater Input2= +ve smaller	Negative
	If the negative input is greater than the positive input then the output must be negative.	Or Input1= +ve smaller Input2= -ve greater	Negative
	Subtraction: 1)The user input must be validated. The sign of the user input must be validated.	Integer or floating point input.	pass
		Alphanumeric	Error
	2) If both the input is of integer type or floating type then the output must be integer or floating type.	Integer, integer	Integer
		Or Float, float	Float
	3)If both the input sign is negative then the output must be the additive of both the values.	Input1= -ve Input2= -ve	Output=-(input1+input2)
	4)If the result exceeds 14 digits then the display unit must show out of bound or out of range	Input1=more than 14 digits	Output= result out
	Multiplication:1) The user input must be validated. The sign of the user input must be validated.	Integer or floating point input.	Pass
		Alphanumeric input	Error
	2) If both the input is of integer type or floating type then the output must be integer or floating type.	Integer, integer	Integer

	Multiplication:1) The user input must be validated. The sign of the user input must be validated.	Integer or floating point input.	Pass
		Alphanumeric input	Error
	2) If both the input is of integer type or floating type then the output must be integer or floating type.	Integer, integer	Integer
		Or Float, float	float
	3)If both the values are negative the output must have a positive sign.	Input1=-ve Input2=-ve	Output=+ve
	If one input is positive and other one is negative then the resultant must have negative sign.	Input1=+ve Input2=-ve	Output=-ve
	Division:1) The user input must be validated. The sign of the user input must be validated.	Integer or floating point input.	Pass
		Alphanumeric input	Error
	2) If both the input is of integer type or floating type then the output must be integer or floating type.	Integer, integer	Integer
		Or Float, float	float
	3) If both the values are negative the output must have a positive sign.	Input1=-ve Input2=-ve	Output=+ve
	If one input is positive and other one is negative then the resultant must have negative sign.	Input1=+ve Input2=-ve	Output=-ve
	4) If the denominator is zero then the display unit must show error.	Input1 = digit Input2 = zero	Error
	If the numerator is zero it must display infinite	Input1=zero Input2= digit	infinite

LLR_2_Memory_Storage	1)It must display the last five results when the user hits the history button.	History	Last five results
	2) The history operation starts storing the results from first after switching on the calculator.	OFF	No result
		ON History	All the results are removed.
LLR_3_Binary_To_Decimal	To convert a binary number to decimal number.	Value= 10010	18
LLR_4_Square_Root	To find the square root of the input value.	Value= 196	14
LLR_5_Power	To find the nth power of the given input.	Values= 12,3	1728
LLR_6_Math_Print	For the calculation 2 no.s should be given	inputs and output both should be displayed at one time e.g. 2,3	2+ 3 ----- 5
LLR_7_Complex_Number	For executing this 4 no.s should be given as input 2 for reals and 2 for imaginary	First complex no.:- Real 2, Imag 3 Second complex no.:- Real 3,Imag 4	5+4i
LLR_8_Trignometric_Functions	1) Sine function- Positive sign of sine function results positive	Sin(30)	0.5
	Negative sign of sine function results negative	Sin(-30)	-0.5
	2) Cosine function- Positive sign of cos function results positive	Cos(30)	0.8660254037844
	Negative sign of cos function results positive	Cos(-30)	0.8660254037844
	3) Tangent function- Positive sign of cos function results positive	Tan(45)	1
	Negative sign of cos function results negative	Tan(-45)	-1
	4) Secant function- Positive sign of sec function results positive	Sec(30)	6.48292123496
	Negative sign of sec function results positive	Sec(-30)	6.48292123496
	5) CoSec function- Positive sign of cosec function results positive	coSec(30)	2
	Negative sign of cosec function results negative	coSec(-30)	-2
	6) Cotangent function- Positive sign of cot function results positive	Cot(45)	1
	Negative sign of cot function results negative	Cot(-45)	-1

Table 3:Test Plan of a Calculator

Implementation Summary

“Section focused toward’ s implementation aspects. Here it is only core summary while all the details are in the Git Repo

Note: The GitHub private repo should be documented (Readme.md files at each folder level)

Ensure code quality and clean code and description practices

Mandatory: To add the GitHub user - **stepin654321** as a contributor to the repo”

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/99003783/T7_SDLC_CALC

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	Unit Testing - Unity passing	Contribution Check - Git Inspector passing

Figure 10:GIT Dashboard

Summary

Workflow

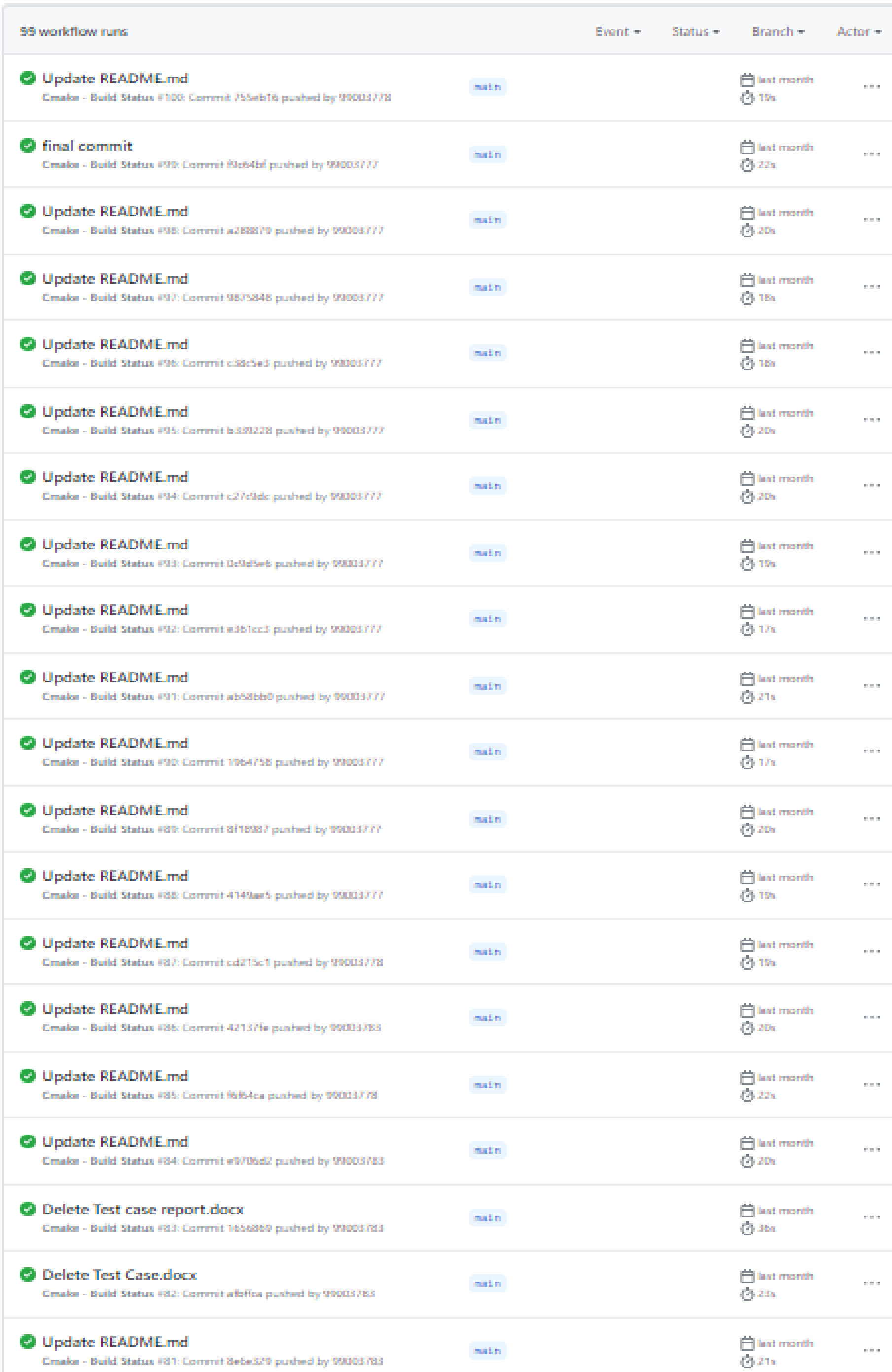

Figure 11:GIT Workflow

Git inspector summary

```
22  The following historical commit information, by author, was found:
23
24  Author                    Commits    Insertions     Deletions    % of changes
25  99003729                        1            24           270            6.13
26  99003777                       43           343           145           10.17
27  99003778                       18           468            19           10.15
28  99003783                       46           962           350           27.34
29  99003785                      113          1118          1100           46.22
```

Figure 12:GIT Inspector Summary

Build

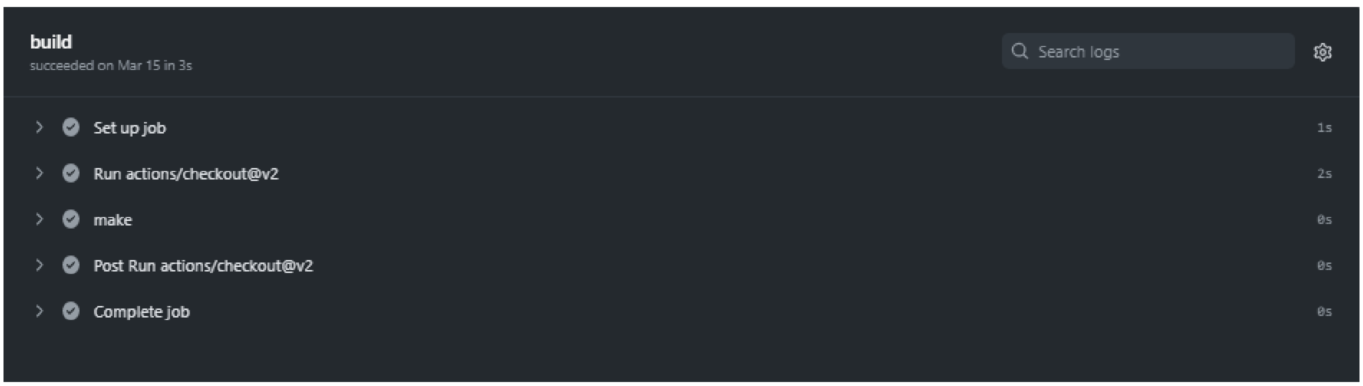

Figure 13:GIT Build

Code quality and Issues or Bug Tracking

```
test/test_calculator_operations.c:73:test_add:PASS
test/test_calculator_operations.c:74:test_add_testcase2:PASS
test/test_calculator_operations.c:75:test_subtract:PASS
test/test_calculator_operations.c:76:test_multiply:PASS
test/test_calculator_operations.c:77:test_divide:PASS
test/test_calculator_operations.c:78:test_modulus:PASS
test/test_calculator_operations.c:80:test_Sin_func1:PASS
test/test_calculator_operations.c:81:test_Cos_func1:PASS
test/test_calculator_operations.c:82:test_Cosec_func1:PASS
test/test_calculator_operations.c:83:test_Sec_func1:PASS
test/test_calculator_operations.c:84:test_Tan_func1:PASS
test/test_calculator_operations.c:85:test_Cot_func1:PASS
test/test_calculator_operations.c:87:test_Sin_func2:PASS
test/test_calculator_operations.c:88:test_Cos_func2:PASS
test/test_calculator_operations.c:89:test_Cosec_func2:PASS
test/test_calculator_operations.c:90:test_Sec_func2:PASS
test/test_calculator_operations.c:91:test_Tan_func2:PASS
test/test_calculator_operations.c:92:test_Cot_func2:PASS
test/test_calculator_operations.c:94:test_Sin_func3:PASS
test/test_calculator_operations.c:95:test_Cos_func3:PASS
test/test_calculator_operations.c:96:test_Cosec_func3:PASS
test/test_calculator_operations.c:97:test_Sec_func3:PASS
test/test_calculator_operations.c:98:test_Tan_func3:PASS
test/test_calculator_operations.c:99:test_Cot_func3:PASS
test/test_calculator_operations.c:101:test_cube_root:PASS
test/test_calculator_operations.c:106:add_complex1:PASS
test/test_calculator_operations.c:107:add_complex2:PASS
test/test_calculator_operations.c:108:sub_complex1:PASS
test/test_calculator_operations.c:109:sub_complex2:PASS
test/test_calculator_operations.c:110:mul_complex1:PASS
test/test_calculator_operations.c:111:mul_complex2:PASS
test/test_calculator_operations.c:112:div_complex1:PASS
test/test_calculator_operations.c:113:div_complex2:PASS
```

Figure 14:Code Quality

Unit Testing

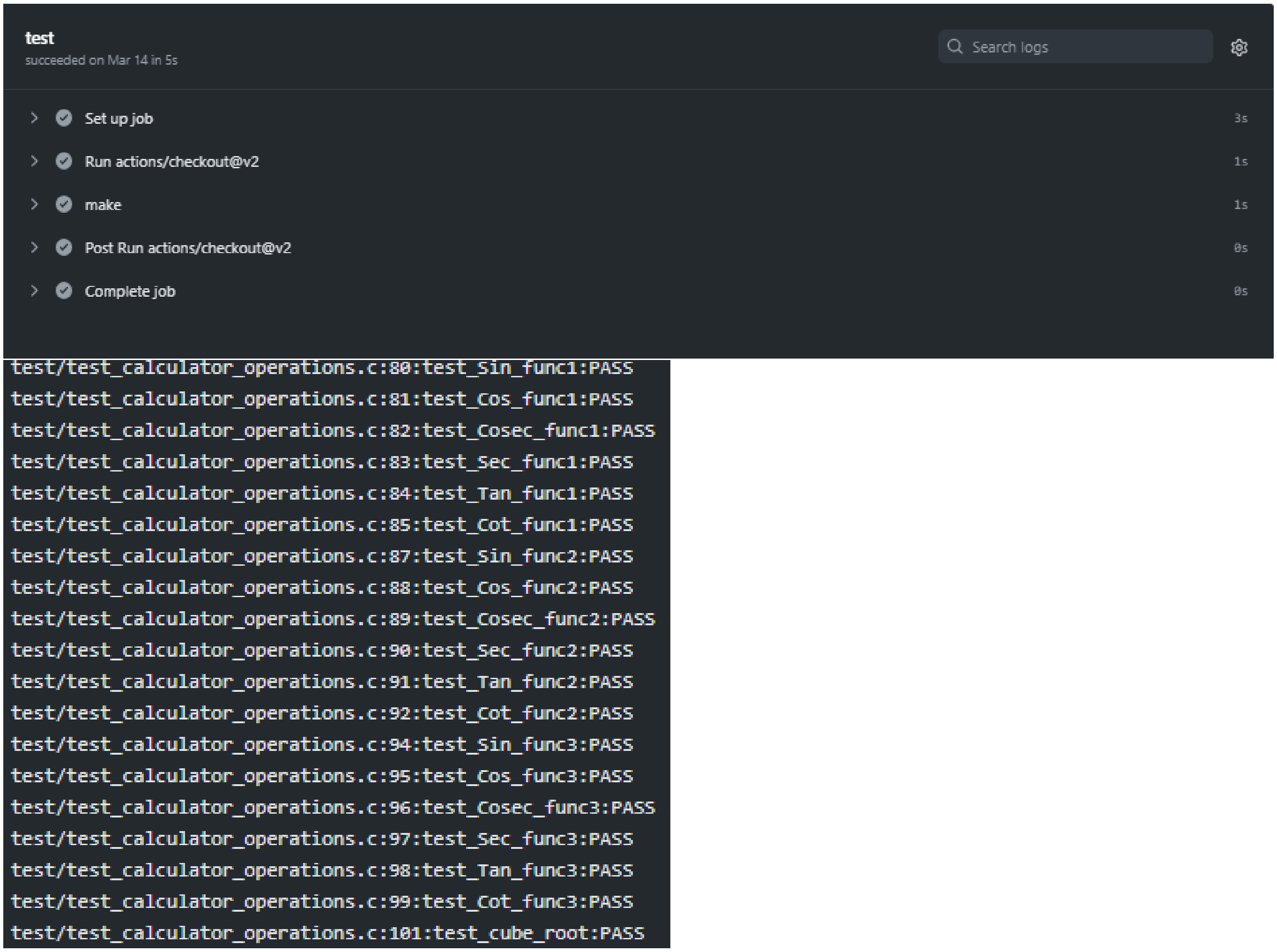

Figure 15:Unit Testing

Individual Contributions and Highlights

PS No.	Name	Features	Issuess Raised	Issues Resolved	No Test Cases	Test Case Pass
99003777	Aman Srivastava	Binary to decimal, squareroot,power	3	3	3	3
99003778	kopaarapu jyothi	Trignometric,logarithemic and cube root	2	2	19	19
99003783	Amiya Kumar Panda	Complex Function and MathPrint	2	3	8	8
99003785	Sourav Dey	Arithmetic Operation and Memory Storage	3	3	6	6

Summary

Hence, Calculator is designed which is portable and user friendly and operations are arithmetic, trigonometric, root and power functions.

Challenges faced and how were they overcome

1. At first, we were facing problem with make file later it was overcome by some research work in that topic.
2. Unable to get the test cases passed later some changes have done on test_calculator_operations.c file and it shows all the test cases in terminal.
3. Initially, I'm not able to work with make file, I overcome the issue by doing an activity in personal repository.
4. Faced issues during compilation of program and working with GitHub. Our colleagues helped us with clearing of these issues.

Miniproject -2 [Individual]

Module

Advanced Python Programming

Topic and Subtopics

Lists, Tuple, Dictionaries and Sets, Regular Expressions
OOPS Concepts- Classes, Objects, Inheritance, Polymorphism

Requirements

High Level Requirements

ID	Requirements	Description
HL_1	Searching the word	Search the word from the input file given by the user
HL_2	writing	Write the 10 characters before and after the searched word in the new text file.
HL_3	Extracting user defined data	Write required data in the text file

Low Level Requirements

ID	Requirements	Description
LL_1	Searching the word	Using the user inputs, the word to be searched through the input file
LL_2	Searching the word in every line of text file	The data to be searched is defined by the user and then searched throughout the entire text file
LL_3	Writing the data into the new text file	Data given as an input by the user searches in every line and the 10 characters before and after is extracted too and then added in a new text file along the searched word. And the name of the output file will be the searched word. Ex: If we are searching for the word software, the output file will be software.txt

Table 4:High Level and Low-Level Requirements of word search and outputfile.txt

Design

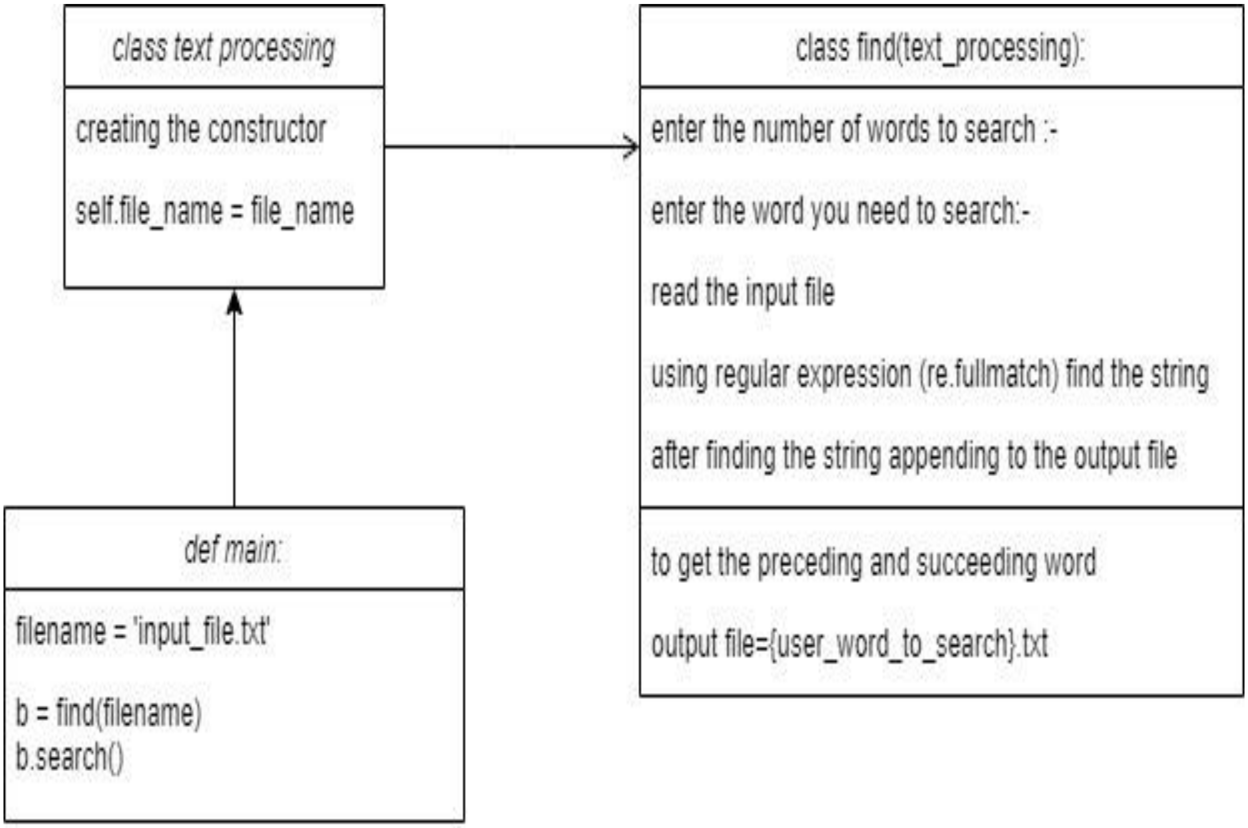

Figure 16:Design of word search

```python
# Importing RegEx module
import re


# Creating Class
class file_search:
    # Creating method and it contains self, string arguments.
    def search_word(self, string):
        # creating attribute
        self.string = string
        # Opening input text file
        input_file = open("input.txt", 'rt')
        #  Reading text in the input file
        text = input_file.read()
        #  finding the string, irrespective of cases
        result = re.findall(string, text, re.M | re.I)
        i = 0
        x = []
        m = 0
        while i < len(result):
            # Searching first string of the output file
            if i == 0:
                f = re.search(result[i], text)
                i += 1
                k = f.span()
                m = k[1]
                # Appending the strings to output file
                x.append(text[k[0]-9:k[0]]+' '+string+' '+text[k[1]+1:k[1]+9])
            else:
                # Searching remaining strings of the output file
                f = re.search(result[i], text[m+1:])
                i += 1
                k = f.span()
                m = k[1]
                # Appening the strings to output file
                x.append(text[k[0]-9:k[0]]+' '+string+' '+text[k[1]+1:k[1]+9])
        y = []
        y.append(str(len(result)))
        for z in range(1, len(x)+1):
            y.append(str(z) + ": " + x[z-1].strip())
        #  Creating output file
        dest = string+".txt"
        #  Opening output file
        with open(dest, 'a') as a:
            #  Writing the lines into output file
            a.writelines('\n'.join(y))
# Creating Main function
if __name__ == "__main__":
    #    creating object
    object_search = file_search()
    #  Enter the input string to search
    string = input("enter a string to search: ")
    #  Searching the required string
    object_search.search_word(string)
```

Figure 17:Python Code for word search and outputfile.txt

Test Plan

Test ID	Description	Input	Expected Output	Actual Output
ID_1	To print the count of the number of words to be printed, to print the word software and 10 characters before and after the word software. Also print everything in new file named **software.txt**	software	35 lines of text with the word 'software ' in each line .	35 lines of text with the word 'software ' in each line .
ID_2	To print the count of the number of words to be printed, to print the word license and 10 characters before and after the word software. Also print everything in new file named **license.txt**	license	87 lines of text with the word 'license ' in each line .	87 lines of text with the word 'license ' in each line
ID_3	To print the count of the number of words to be printed, to print the word work and 10 characters before and after the word software. Also print everything in new file named **work.txt**	work	64 lines of text with the word 'work ' in each line	64 lines of text with the word 'work ' in each line

Table 5:Test Plan for Word Search and Outputfile.txt

Implementation Summary

Hence, the word we searched and 10 characters before and after the word has been implemented in the following way:

```
35
1: 999 Free  software Foundati
2: GNU Libr software c Licens
3: 999 Free  software Foundati
4: GNU Libr software c Licens
5: 999 Free  software Foundati
6: s license software ser Gene
7: Version software ruary 19
8: ersion nu software ]
9: Version software ruary 19
10: ersion nu software ]
11: Version software ruary 19
12: ersion nu software ]
13: Version software ruary 19
14: ersion nu software ]
15: Version software ruary 19
16: ersion nu software ]
17: Version software ruary 19
18: ersion nu software ]
19: Version software ruary 19
20: ersion nu software ]
21: Version software ruary 19
22: ersion nu software ]
23: License,  software , hence
24: ersion nu software ]
25: License,  software , hence
26: opyright  software  1999 Fr
27: NU LESSER software PUBLIC L
28: opyright  software  1999 Fr
29: License,  software , hence
30: opyright  software  1999 Fr
31: DERED INA software R LOSSES
32: NU Librar software License,
33: 1999 Fre software e Founda
34: license,  software r Genera
35: 1999 Fre software e Founda
```

Figure 18:Implementation Summary

Implementation Summary

Hence, the python code is implemented to search for a word and print the output files named searchedword.txt using regular expressions

Challenges faced and how were they overcome

Initially, I was unable to use classes for this program and by doing some exercises on classes and objects and by going through some inline references I sorted out this problem.

Miniproject -3 [Team]

Module

Embedded C

Topic and Subtopics

Body Control Modules (BCM) of a car-Buzzer trigger system and wiper control system has been implemented using STM32 board, HAL and Embedded C coding.

Requirements

High Level Requirements

ID	Requirements
HL_1	Wiper control system
HL_2	Buzzer trigger system

Low Level Requirements

ID	Requirements
LL_1	Wiper ON
LL_2	Wiper OFF
LL_3	Buzzer ON
LL_4	Buzzer OFF

Design

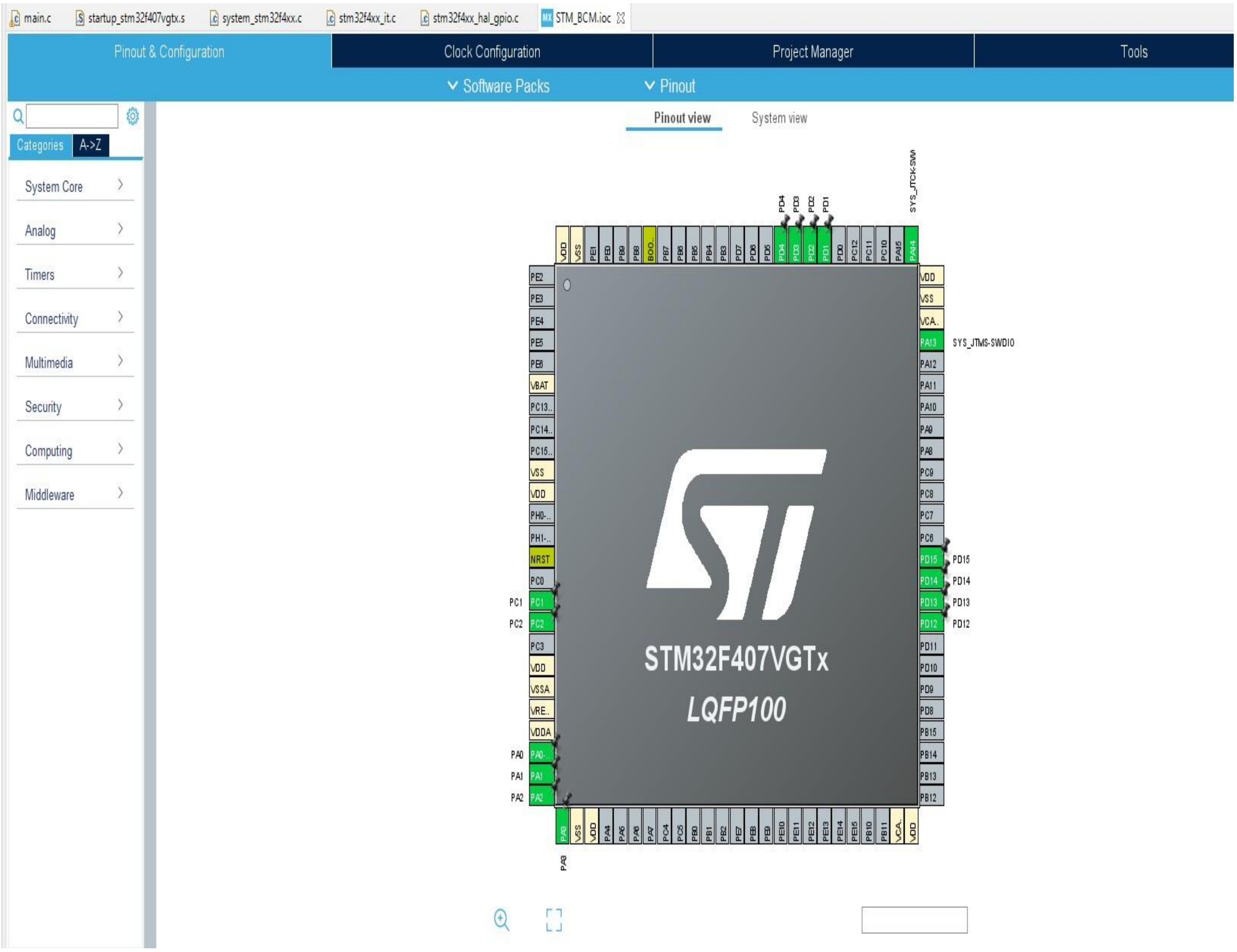

Figure 19:Pin Configuration of BCM Module Features

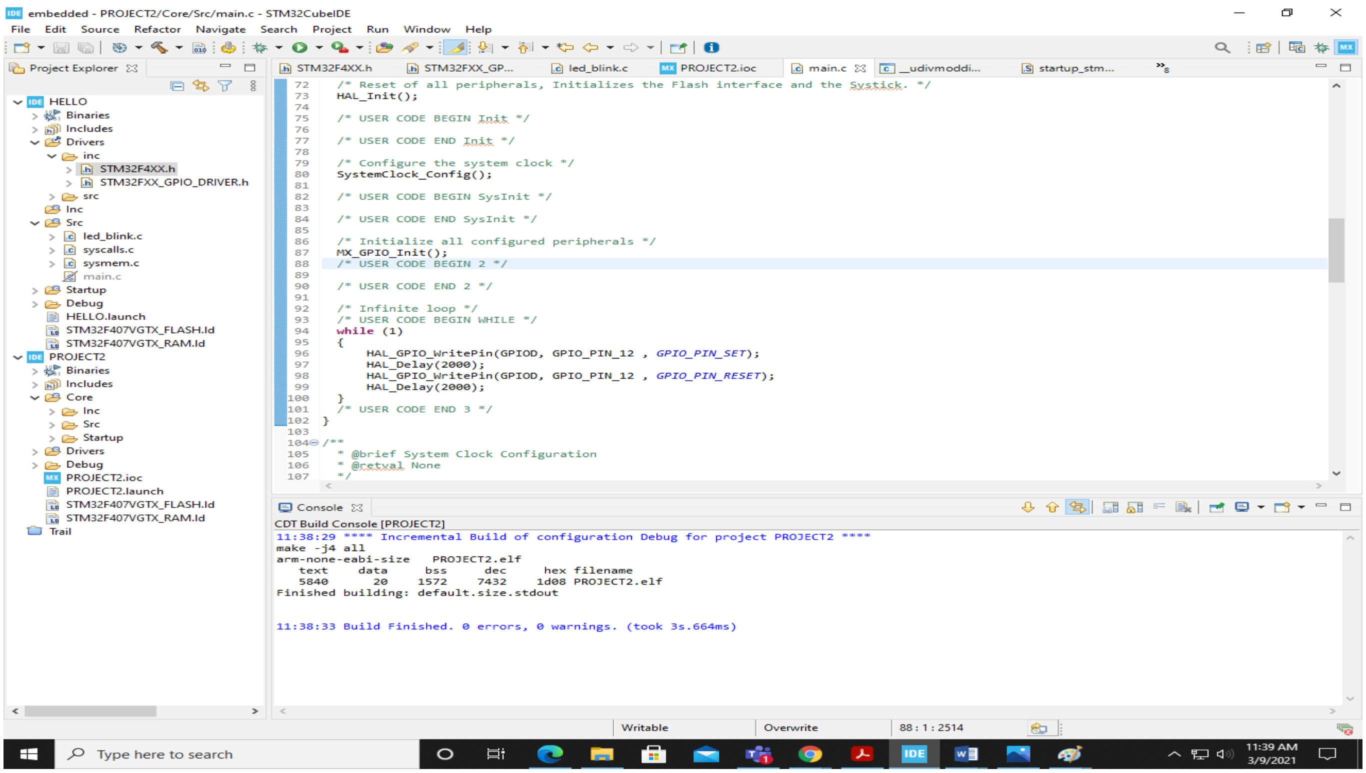

Figure 20:HAL Code for Buzzer trigger system

Figure 21:Hardware Setup for Buzzer Trigger System

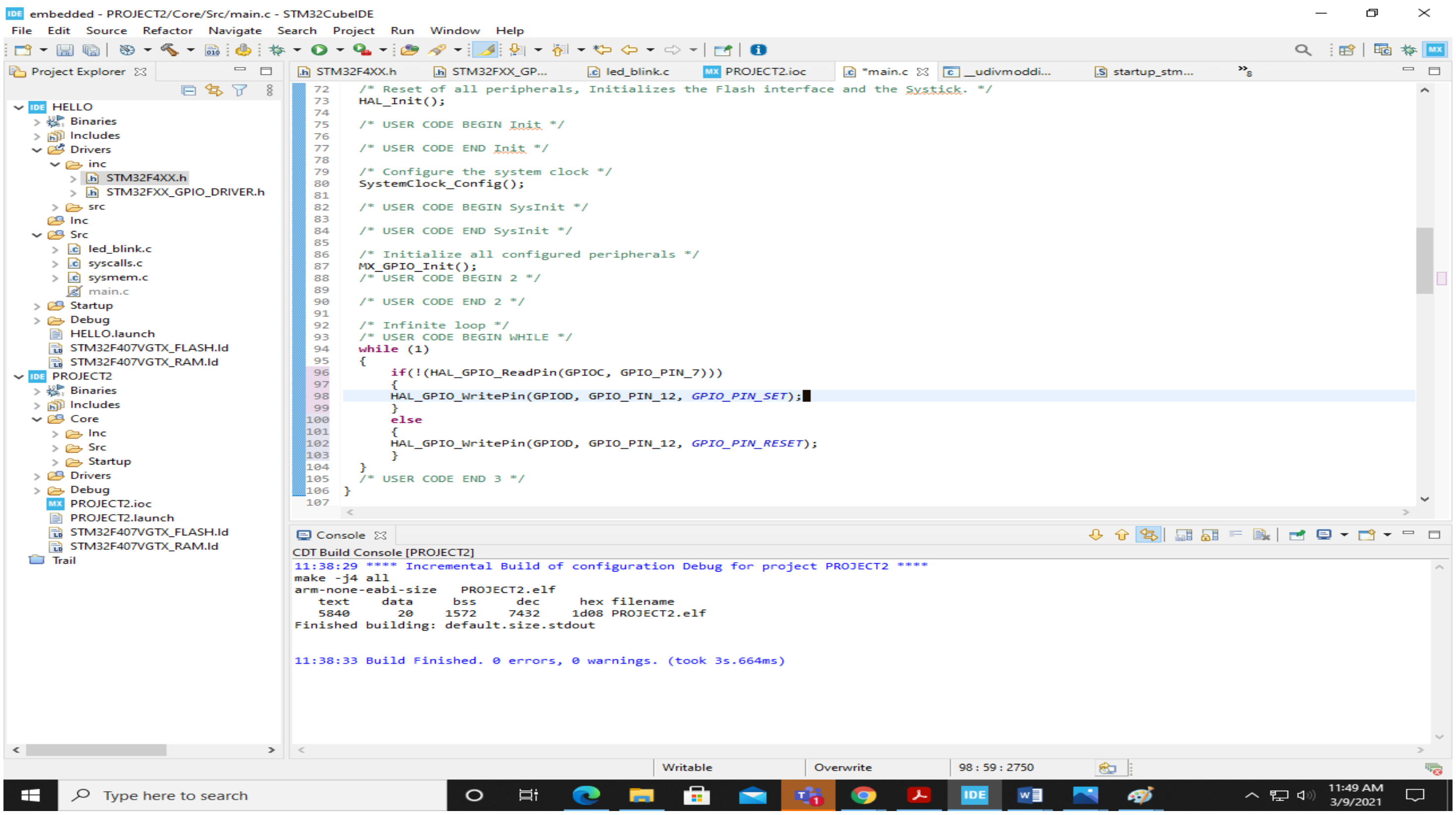

Figure 22:HAL Code for Wiper Control System

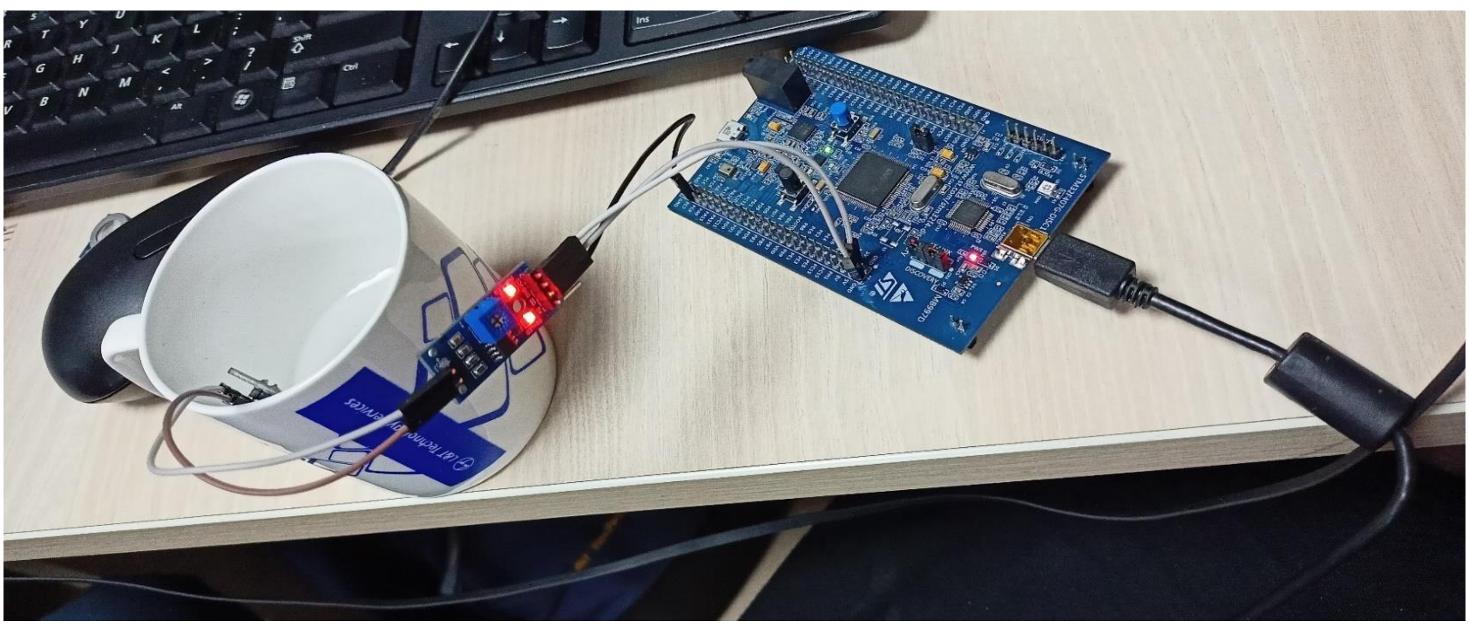

Figure 23:Hardware Setup for Wiper Control System

Test Plan

High Level Test Plan

ID	Requirements	Description	Input	Actual Output	Expected Output
HL_1	Wiper control system	Wiper should ON/OFF based on sensor conditions	Rain Drops	ON	ON
HL_2	Buzzer trigger system	Buzzer should detect whenever the object is detected	Person	ON	ON

Low Level Test Plan

ID	Requirements	Description	Input	Actual Output	Expected Output
LL_1	Wiper ON	Whenever the rain sensor senses the rain drops, then the wiper wipes	Rain Drops	ON	ON
LL_2	Wiper OFF	Whenever the rain sensor doesn't senses the rain drops, then the wiper stops wiping	No Rain Drops	OFF	OFF
LL_3	Buzzer ON	Whenever unauthorized person tries to access the car, then buzzer triggers	Person	ON	ON
LL_4	Buzzer OFF	Whenever unauthorized person doesn't access the car, then buzzer will not trigger	When there is no person	OFF	OFF

Table 6:High Level and Low-Level Test Plan of wiper and Sunroof Systems

Implementation Summary

Hence, Wiper control system and Buzzer trigger system was implemented using STM32 Board and Embedded C coding.

Challenges faced and how were they overcome

Initially, I'm not able to understand the embedded C code and was not able to write the code.
Later, after doing some exercises on embedded C coding, I was able to understand and write the code.

Miniproject -4 [Team]

Module

Model based system engineering (MBSE)

Topic and Subtopics

Body Control Module (BCM) of a car- wiper control system and power outlet has been implemented using MATLAB.
MATLAB-Onramp
Simulink-Onramp
Stateflow-OnRamp

Requirements

High Level Requirements

Wiper control system-Wiper must wipe based on conditions

Power Outlet-Power outlet should activate based on ignition

Low Level Requirements

a. 0-Wiper should OFF
b. 1-Wiper must wipe back and forth(1x)
c. 2-Wiper must wipe with low speed
d. 3-Wiper must wipe with medium speed
e. 4-Wiper must wipe with High speed.
f. Whenever the ignition is ON, Power outlet should be ON
g. Whenever the ignition is OFF, Power outlet should be OFF

Design

High Level Design

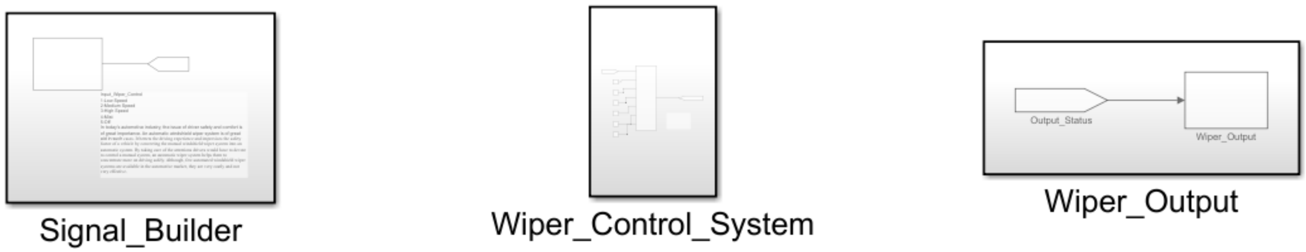

Figure 24:High Level Design of Wiper Control System

Figure 25:High Level Design of Power Outlet

Low Level Design

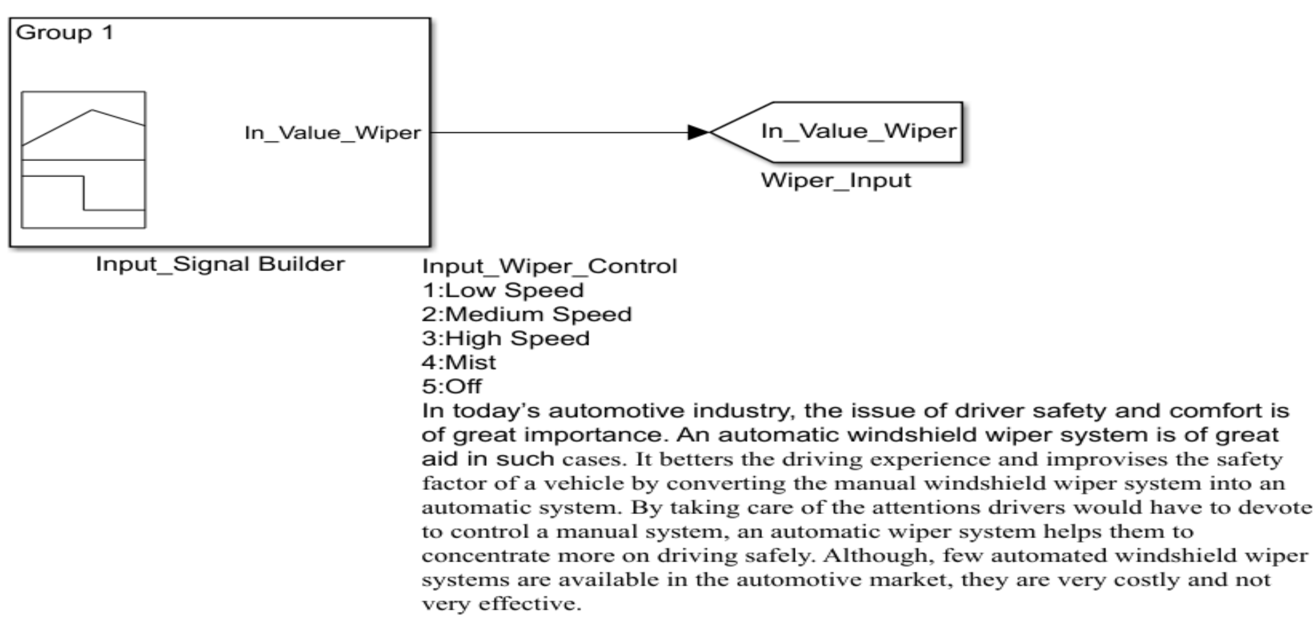

Figure 26:Low Level Design of Wiper Signal Builder

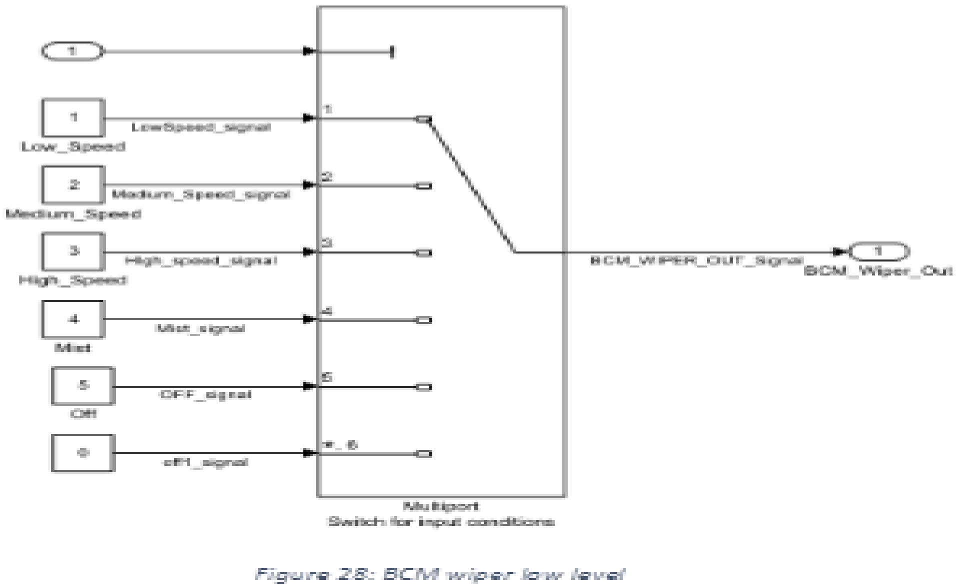

Figure 27:Low Level design of multiport switch for input conditions

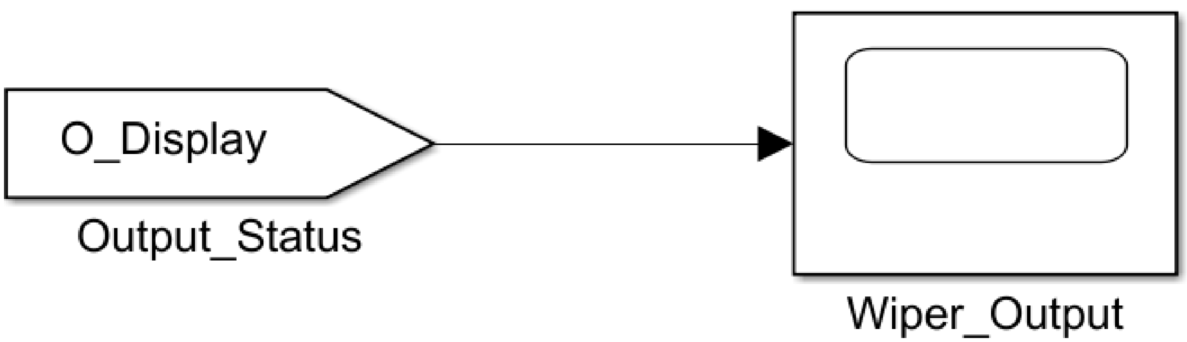

Figure 28:Low Level design of wiper output

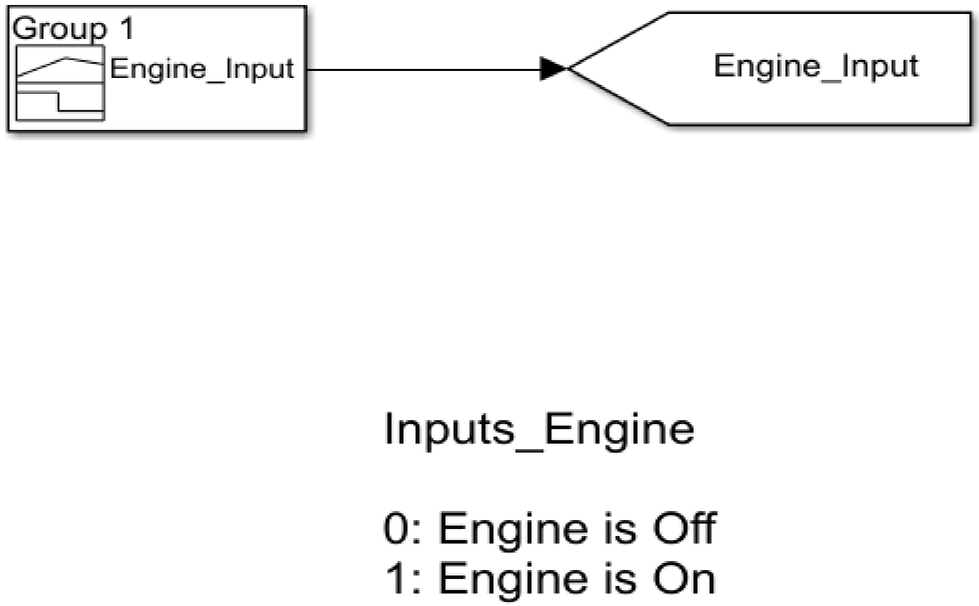

Figure 29:Low Level Design of Power Outlet Signal Builder

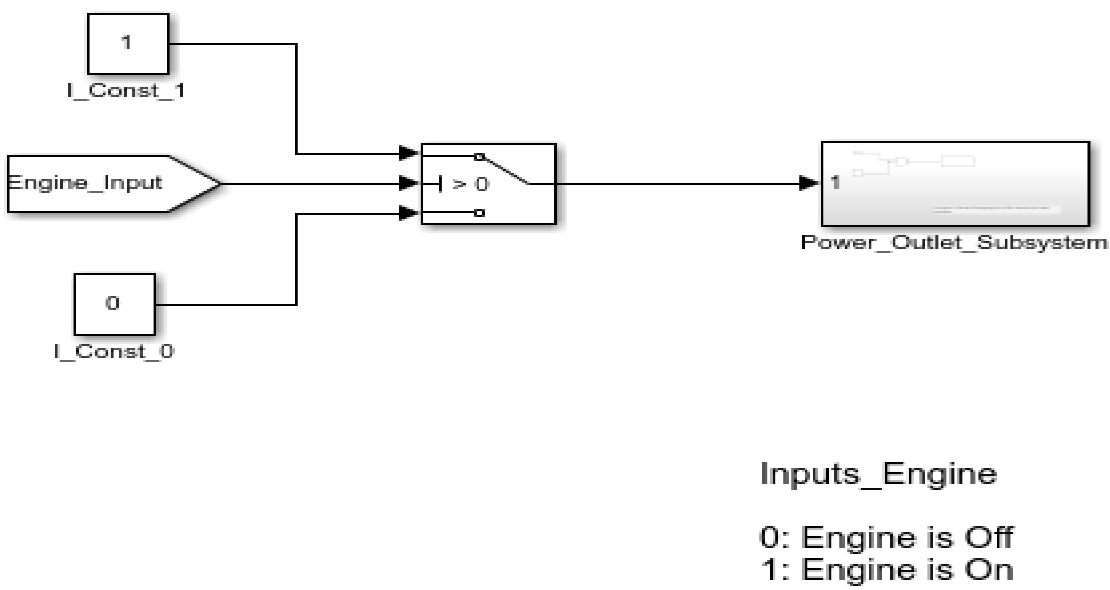

Figure 30:Low Level Design of Power Outlet Engine input

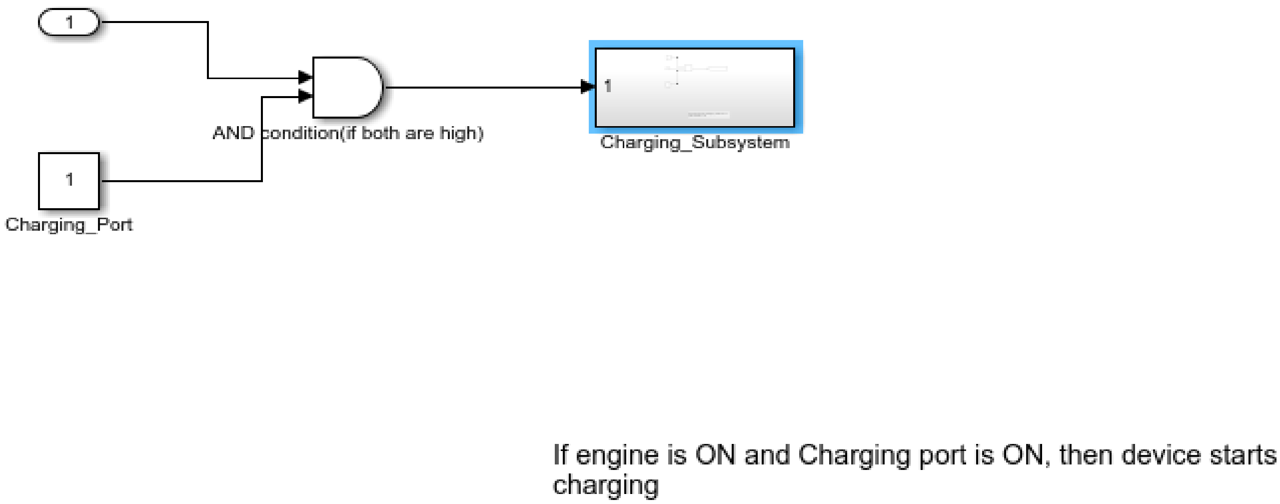

Figure 31:Low Level Design power outlet charging subsystem

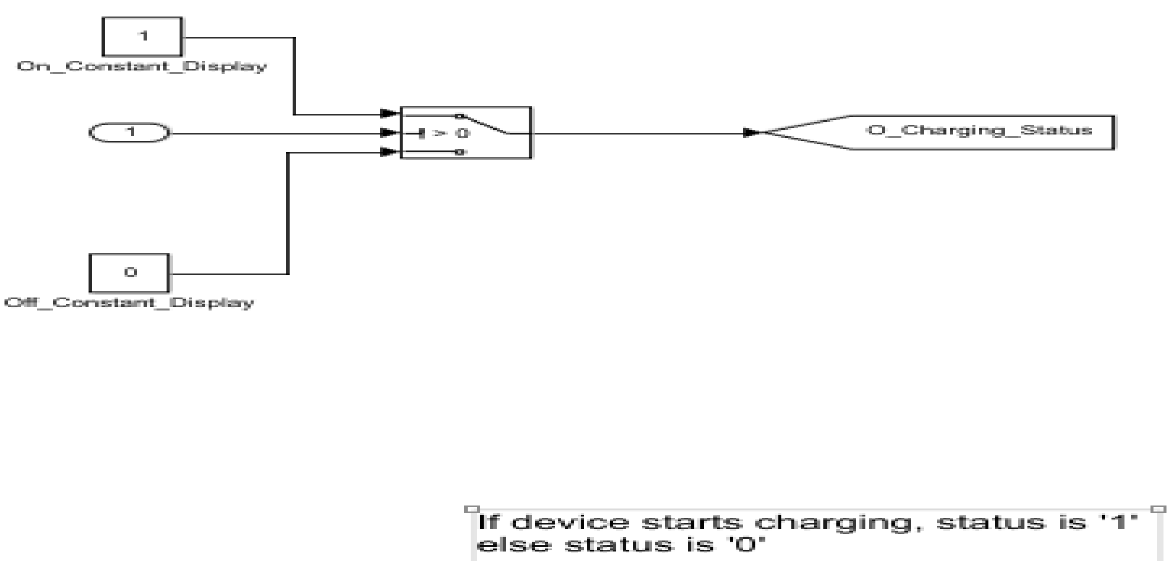

Figure 32:Power Outlet Output

Test Plan

High Level Test Plan

ID	Description	Input	Expected Output	Actual Output	Result
HLT_FW_01	Enabling front wind shield wiper	Front Wiper should be on	Front Wiper On	Front Wiper On	Pass
HLT_RW_052	Enabling rear wind shield wiper	Rear Wiper should be on	Rear Wiper On	Rear Wiper On	Pass

Low Level Test Plan

ID	Description	Input	Expected Output	Actual Output	Result
LLT_LS_09	Enabling Low Speed Wiper	Low Speed Wiper should be on	Low Speed Wiper On	Low Speed Wiper On	Pass
LLT_HS_10	Enabling High Speed Wiper	High Speed Wiper should be on	High Speed Wiper On	High Speed Wiper On	Pass
LLT_M_11	Enabling Mist condition Wiper	Mist Condition Wiper should be on	Mist Condition Wiper On	Mist Condition Wiper On	Pass

Table 7:High Level and Low-Level Test Plan of MBSE Wiper Control System

Implementation Summary

Therefore, Implementation of Wiper control system and Power Outlet has been completed using MATLAB by using different blocks like switch, multiport switch, And block, sub systems, scope, display etc.,

Challenges faced and how they overcome

We find difficulty with **goto** and **from** blocks while converting from manual to automation method.

Miniproject -5 [Team]

Module

Automotive Protocols (CAN&UDS), Tool used is Canoe 10.0 SP7 and CAPL Scripting.

Topic and Subtopic

CAN and UDS

Requirements

High Level Requirements

Wiper control system-Wiper must wipe whenever rain falls

Sunroof System- Sunroof must OFF whenever rain falls

Low Level Requirements

When the ignition is ON

1. When the rain falls, then the wiper would be able to wipe, and sunroof should OFF automatically.
2. When there is no rain, it is up to user to turn ON the sunroof.

Design

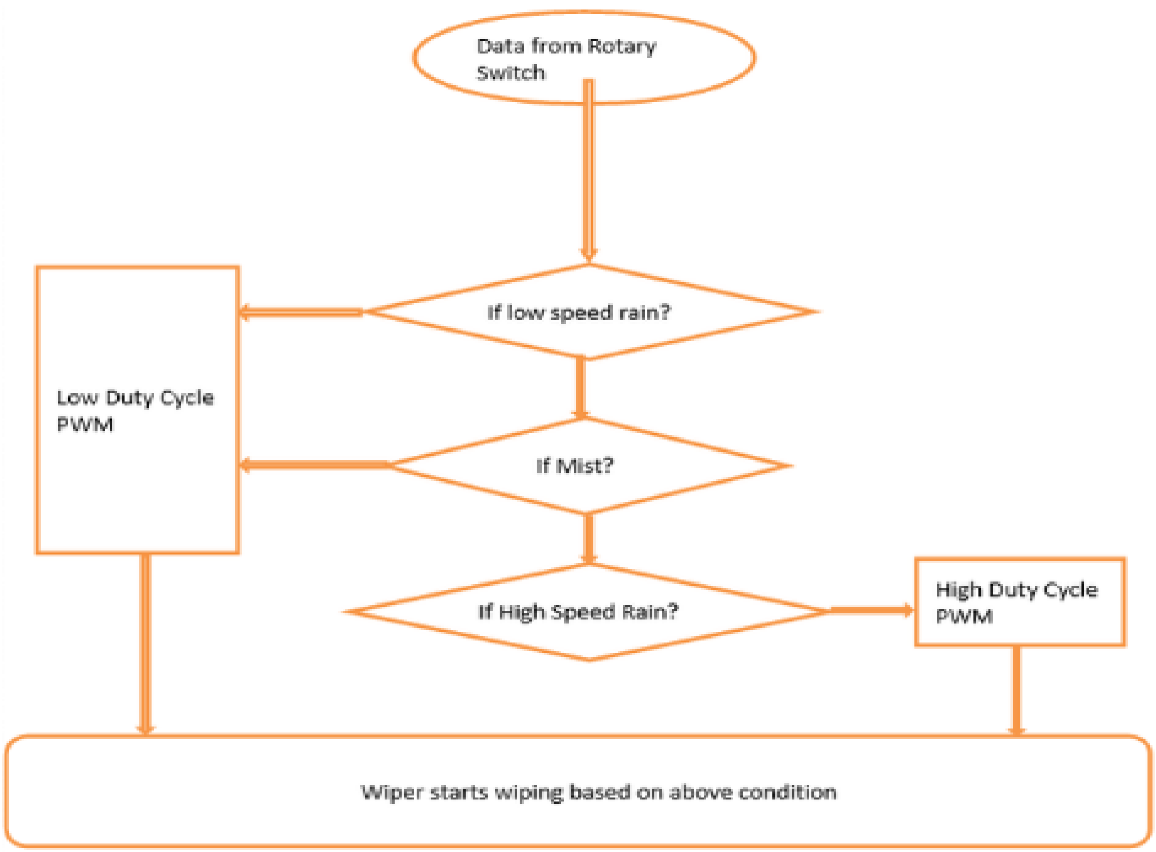

Figure 33:Design flow of wiper control system

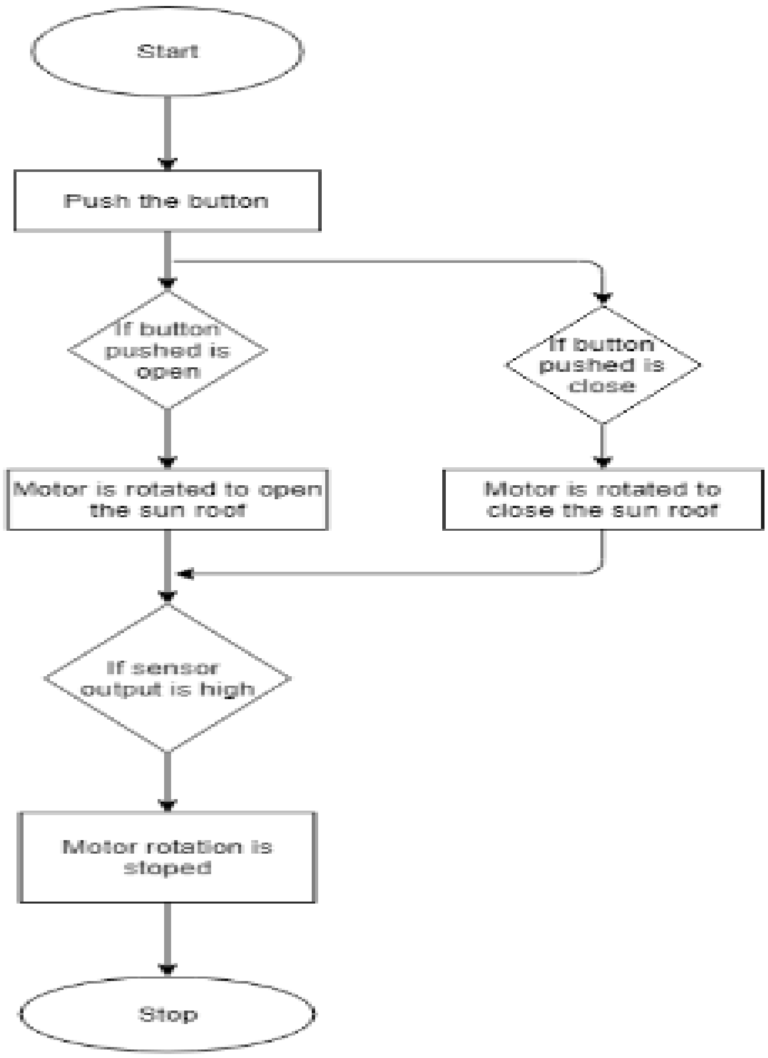

Figure 34: Design flow of sunroof system

Test Plan

ID	Requirements	Description	Input	Output
T_01	Wiper Control System	Wiper should ON, and Sunroof should OFF Whenever rain falls	Rain drops	Wiper ON Sunroof OFF
T_02	Sunroof System	Sunroof should on whenever the user hits on switch. And it should not close if there is an object detected	object	Sunroof ON

Table 8:Test Plan of Canoe Wiper and sunroof systems

Implementation Summary

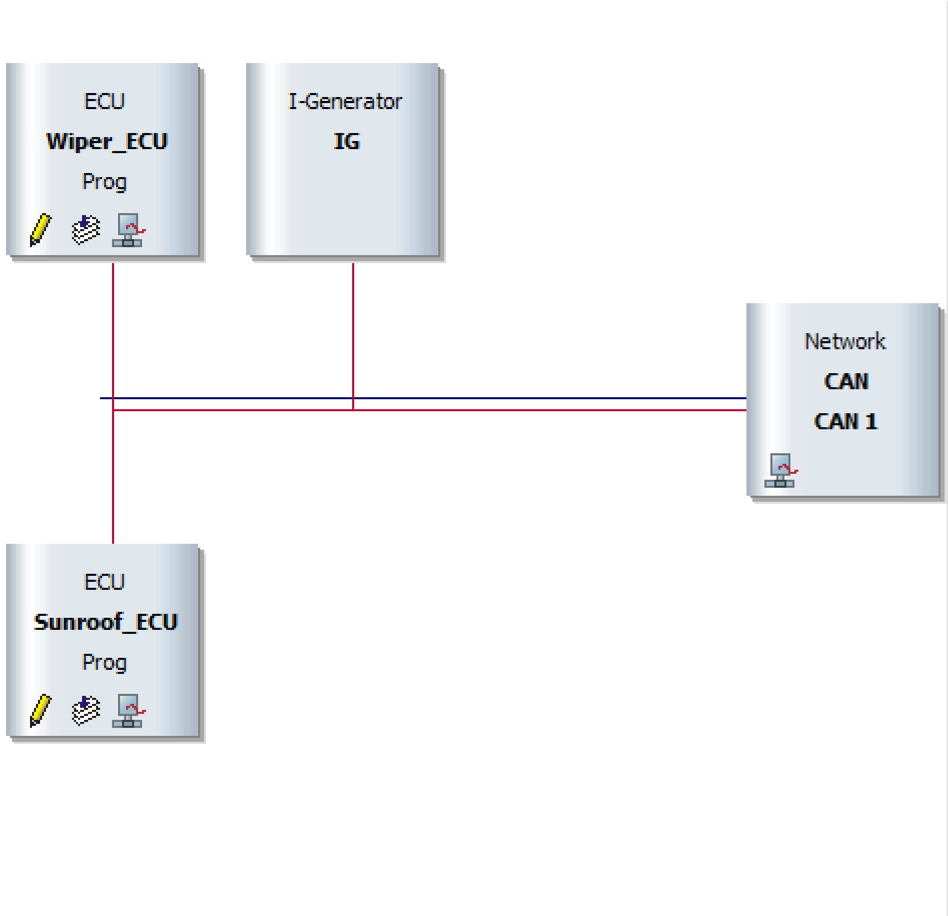

Figure 35:ECU's of Wiper control system and Sunroof System

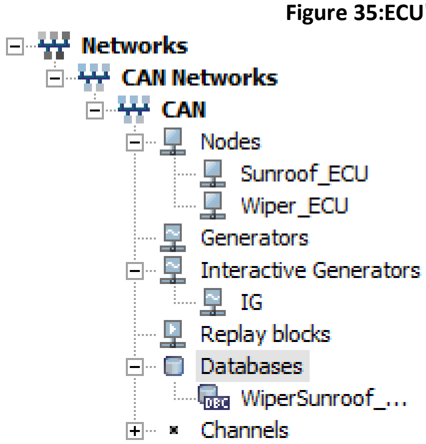

Figure 36:Nodes and Databases for wiper and sunroof

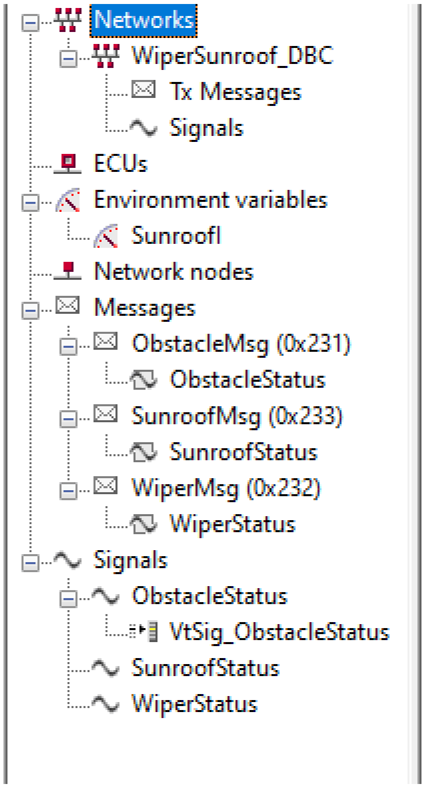

Figure 37:Messages and Signals for wiper and sunroof system

```
includes
{

}
variables
{
  message WiperMsg wp;
  int pos=10;
  int i=0;
  msTimer timer_wp;
}
on key 'k'    //To on the Wiper
{
  pos=10;
  wp.WiperStatus=pos;
  output(wp);
  setTimer(timer_wp,1);
}
on timer timer_wp
{
  if(pos==200)
  {
    pos=0;
  }
  else
  {
    pos=pos+10;
    wp.WiperStatus=pos;
  }
  output(wp);
  setTimer(timer_wp,1);
  i=0;
```

Figure 38:CAPL Script for wiper control system

```
includes
{

}

variables
{
  message SunroofMsg sun;
  message ObstacleMsg ob;
  message WiperMsg wi;
  int var = 10;
}
on key 'o'
{
 if(var<100)
{
  var=var+10;
  sun.SunroofStatus=var;
  output(sun);
}
}
on key 'c'
{
 if(var>0)
{
  var=var-10;
  sun.SunroofStatus=var;
  output(sun);
}
}
on message ObstacleMsg
{
  if(this.ObstacleStatus==1)
  {
    sun.SunroofStatus=100;
    output(sun);
  }
}
on message WiperMsg
{
  if(this.WiperStatus>0)
  {
    sun.SunroofStatus=0;
    output(sun);
```

Figure 39:CAPL Script for Sunroof System

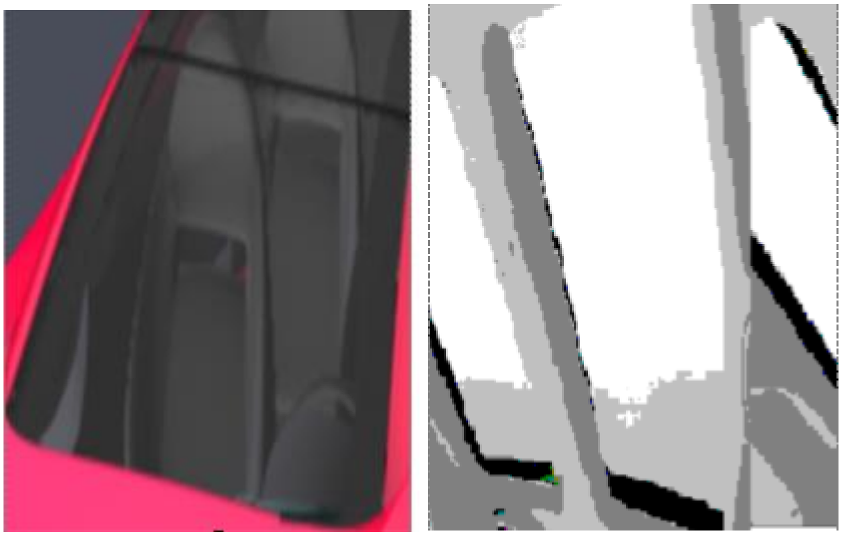

Figure 40:Output of Wiper and Sunroof Systems

Implementation Summary

Hence, Wiper control system and Sunroof system has been implemented in Canoe using databases, messages, signals, CAPL Scripting, control and display panels and Log file is generated.

Challenges faced and how they overcome

We find difficulty in working with signals, display and control panels. Later, by doing some exercises on these we are able to do the project.

Miniproject -6 [Individual]

Module

Intermediate C++

Topic and Subtopics

Memory Layout of a C++ program, Operator Overloading, Function Overloading, Templates, Friend function and virtual function, Classes and Objects, Inheritance and Polymorphism.

Requirements

High Level Requirements

ID	Requirements	Description
HL_01	Operations	User can perform different calculations like add, sub, mul and div, complex and fraction numbers
HL_02	Number types	To perform these operations user can be able to do for different data types

Table 9:Low Level Requirements of Arithmetic Operations, Complex numbers and fraction numbers

Low Level Requirements

ID	Requirements	Description
LL_01	Addition	Addition of two integer numbers, float numbers
LL_02	Subtraction	Subtraction of two integer numbers, float numbers
LL_03	Multiplication	Multiplication of two integer numbers, float numbers
LL_04	Division	Division of two integer numbers, float numbers
LL_05	Complex Numbers	Addition and Subtraction of two complex numbers
LL_06	Fraction Numbers	Addition and Subtraction of two Fraction numbers

Table 10:Low Level Requirements of Arithmetic Operations, Complex numbers and fraction numbers

Design

```
#ifndef __OPERAIION_H_
#define __OPERAIION_H_
#include<iostream>
using namespace std;
template <typename T>
class operation
{
private :
        T num1;
        T num2;
        T a;
        T b;

public:
    operation();
    operation(T n1,T n2);
    T add(T n1,T n2);
    T subtract(T n1,T n2);
    T multiply(T n1,T n2);
    T division(T n1,T n2);

};
using namespace std;
template <typename T>
operation<T>::operation()
{}
template <typename T>
operation<T>::operation(T n1,T n2):num1(n1), num2(n2)
{}
template <typename T>
T operation<T>::add(T n1,T n2)
{
    std::cout<<n1+n2<<endl;
    return n1+n2;

}
template <typename T>
T operation<T>::subtract(T n1,T n2)
{   std::cout<<n1-n2<<endl;
    return n1-n2;
}
template <typename T>
T operation<T>::multiply(T n1,T n2)
{    std::cout<<n1*n2<<endl;
    return n1*n2;
}
template <typename T>
T operation<T>::division(T n1,T n2)
{
    std::cout<<n1/n2<<endl;
    return n1/n2;
}
#endif
```

Figure 41:Design Flow of Arithmetic Operations

```
{
    complex<J> c3;
    J denominator;
    denominator=obj.real*obj.real+obj.imag*obj.imag;
    J T1;
    J T2;
    T1 = real*obj.real+imag*obj.imag;
    T1 = T1/denominator;
    T2 = imag*obj.real-real*obj.imag;
    T2 = T2/denominator;
    c3.real=T1;
    c3.imag=T2;
    return c3;

};
template<typename J>
complex<J>complex<J>::operator*(complex<J>&obj)
{
complex<J> c3;
c3.real=(real*obj.real)-(imag*obj.imag);
c3.imag=(real*obj.imag+imag*obj.imag);
return c3;
};
template<typename J>
complex<J>complex<J>::operator-(complex<J>&obj)
{
   complex<J> c3;
   c3.real=real-obj.real;
   c3.imag=imag-obj.imag;
   return c3;
};
template<typename J>
complex<J>complex<J>::operator+(complex<J>&obj)
{
    complex<J> c3;
    c3.real=real+obj.real;
    c3.imag=imag+obj.imag;
    return c3;
};
template<typename J>
void complex<J>::display()
{
    cout<<"Real "<<real<<" "<<"Imaginary "<<imag;
}
template <typename J>
J complex<J>::return_real_value()
{
    return real;

}
template <typename J>
J complex<J>::return_imag_value()
{
    return imag;

}
```

Figure 42:Design Flow of Complex Numbers

```
template <typename T>
Fraction<T>::Fraction(){}
template <typename T>
Fraction<T>::Fraction(T num,T den):numerator(num),denominator(den){}
template <typename T>
Fraction<T>::~Fraction(){}
template <typename T>
Fraction<T> Fraction<T>::operator+(const Fraction& ref)
{
Fraction<T> r;
r.numerator=(numerator*ref.denominator)+(denominator*ref.numerator);
r.denominator=denominator*ref.denominator;
return(r);
}
template <typename T>
Fraction<T> Fraction<T>::operator-(const Fraction& ref)
{
Fraction<T> r;
r.numerator=(numerator*ref.denominator)-(denominator*ref.numerator);
r.denominator=denominator*ref.denominator;
return(r);
}
template <typename T>
Fraction<T> Fraction<T>::operator*(const Fraction& ref)
{
Fraction<T> r;
r.numerator=numerator*ref.numerator;
r.denominator=denominator*ref.denominator;
return(r);
}
template <typename T>
Fraction<T> Fraction<T>::operator/(const Fraction& ref)
{
Fraction<T> r;
r.numerator=numerator*ref.denominator;
r.denominator=denominator*ref.numerator;
return(r);
}
template<typename T>
void Fraction<T>::printF(){
cout<<numerator<<"/"<<denominator<<endl;
}
template <typename T>
T Fraction<T>::return_denominator_value()
{
return denominator;
}
template <typename T>
T Fraction<T>::return_numerator_value()
{
return numerator;
}
#endif
```

Figure 43:Design flow of Addition of two fractions

Test Plan

ID	Requirements	Description	Input	Expected Output	Actual Output
T_01	Addition	Addition of two integer numbers, float numbers	2,3 1.3,0.4	5 2.7	5 2.7
T_02	Subtraction	Subtraction of two integer numbers, float numbers	5,4 6.5,3.1	1 3.4	1 3.4
T_03	Multiplication	Multiplication of two integer numbers, float numbers	1,2 0.2,1.2	2 0.24	2 0.24
T_04	Division	Division of two integer numbers, float numbers	6,3 2.4,1.1	2 2.181	2 2.181
T_05	Complex Numbers	Addition and Subtraction of two complex numbers	2,3 2,3	2+3i 2-3i	2+3i 2-3i
T_06	Fraction Numbers	Addition and Subtraction of two Fraction numbers	1/2+ 1/3 1/2+ 1/3	5/6 1/6	5/6 1/6

Table 11:Test Plan of Arithmetic Operations, Complex numbers and fraction numbers

Summary

Hence, Implementation of calculator using classes, objects, operator overloading, Templates has completed.

Challenges faced and how they overcome

Initially, I’m confused with operator overloading and polymorphism concepts. Later, by doing some exercises on these topics I’m able to do.