

GENESIS - Learning Outcome & Mini-project Summary Report



LTTTS
GLOBAL
ENGINEERING
ACADEMY



L&T Technology Services



Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved	Remarks/Revision Details
		Akshansh Mishra PS No- 99003753			

Contents

CONTENTS	3
1. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)	5
1.1 MODULES USED.....	5
1.2 PROJECT TITLE : CALCULATOR.....	5
1.3 TOPIC AND SUBTOPICS.....	5
1.4 OBJECTIVES & REQUIREMENTS.....	5
1.4.1 High Level requirement Analysis –.....	5
1.4.1 Low Level Requirement Analysis -.....	6
1.5 DESIGN	6
1.5.1 High level diagram: Deployment Diagram.....	6
1.5.2 LLR Diagram: Composite diagram.....	7
1.5.3 LLR Diagram: Profile Diagram -.....	8
1.5.3 LLR Diagram : Communication Diagram -.....	8
1.5.4 LLR Diagram: Object diagram –.....	9
1.5.5 LLR Diagram: Activity diagram -.....	10
1.5.6 LLR Diagram: Deployment diagram -.....	11
1.5.7 LLR Diagram: UML Class diagram.....	11
1.5.8 LLR Diagram: State diagram.....	12
1.6 TEST PLAN	12
1.6.1 High Level Test Plan.....	12
1.6.2 Low Level Test Plan.....	13
1.7 IMPLEMENTATION SUMMARY	13
1.8 VIDEO SUMMARY	ERROR! BOOKMARK NOT DEFINED.
1.9 GIT LINK.....	13
1.10 GIT DASHBOARD.....	13
1.10.1 Badges.....	13
1.10.2 Git Inspector.....	14
1.10.3 Setup for Build.....	15
1.10.4 Outcome of the Build	15
1.10.5 Setup for Code Quality.....	16
1.10.6 Outcome of Code Quality.....	16
1.10.7 Setup for Unity Testing	17
1.10.8 Outcome of Unity Testing.....	17
1.11 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS.....	18
1.12 SUMMARY.....	18
1.12.1 Outcomes:.....	18
1.13 CHALLENGES FACED AND HOW WERE THEY OVERCOME	18
2. PYTHON PROGRAMMING PROJECT	19
2.1 MODULES USED.....	19
2.2 PROJECT TITLE: RETRIEVE DATA FROM MULTIPLE EXCEL SHEET.....	19
2.3 TOPIC AND SUBTOPICS	19
2.4 OBJECTIVES:.....	19
2.5 REQUIREMENTS:.....	19
2.5.1 High Level requirement analysis.....	19
2.5.2 Low Level requirement Analysis -.....	19
2.6 DESIGN	20

2.6.1 Use Case LLR Diagram.....	20
2.6.2 Object HLR Diagram.....	20
2.7 IMPLEMENTATION SUMMARY	21
2.8 VIDEO SUMMARY	ERROR! BOOKMARK NOT DEFINED.
2.9 GIT LINK.....	21
2.10 GIT DASHBOARD	21
2.11 SUMMARY -.....	21
2.11.1 Outcomes:.....	21
2.12 CHALLENGES FACED AND HOW WERE THEY OVERCOME	22
3.EMBEDDED C.....	23
3.1 MODULE:.....	23
3.2 OBJECTIVES & REQUIREMENTS.....	23
3.2.1 HIGH LEVEL REQUIREMENTS:.....	23
3.2.2 LOW LEVEL REQUIREMENTS:.....	23
3.3 TEST PLAN.....	23
3.5 IMPLEMENTATION SUMMARY	25
3.6 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS.....	26
3.7 SUMMARY.....	26
4.0 MINIPROJECT -4 [INDIVIDUAL] – KERNEL PROGRAMMING AND DEVICE DRIVERS -.....	27
4.1 MODULE :	27
4.2 TOPIC AND SUBTOPICS:	27
4.3 OBJECTIVES & REQUIREMENTS:.....	27
4.3.1 Requirements:.....	28
4.4 IMPLEMENTATION SUMMARY:	28
4.4.1 Hands-on Activity that are implemented are as follow:.....	28
4.4.2 User space code:.....	28
4.4.3 kthread examples:.....	28
4.5 GIT LINK:.....	29
4.6 GIT DASHBOARD	29
4.7 SUMMARY:.....	29
4.8 CHALLENGES FACED AND HOW WERE THEY OVERCOME:.....	30

1. Software Development Life Cycle (SDLC)

1.1 Modules Used

Modules used in this project are SDLC and C programming.

1.2 Project title : Calculator

Modules linked to the mini project Ex – Linux, SDLC and C.

1.3 Topic and Subtopics

- The core steps of SDLC is being implemented.
- The features of Calculator are implemented.
- The testing has been done for each function.
- Introduction about SDLC
- C Programming
- Code Analysis
 1. CPP Check
 2. Valgrind
- Testing
 1. Unity Testing
- Makefile
- V Model
- Agile Model
- Git Hub

1.4 Objectives & Requirements

1.4.1 High Level requirement Analysis –

- Any calculator must be efficient.
- Any calculator must have a user-friendly interface.
- It should also be accurate in terms of results.
- It should be able to perform multiple functions.
- It must be cost efficient.

ID	Description	Status
HLR 01	Basic Arithmetic Calculation	Implemented
HLR 02	Trigonometric Calculation	Implemented
HLR 03	Dimension Conversion	Implemented

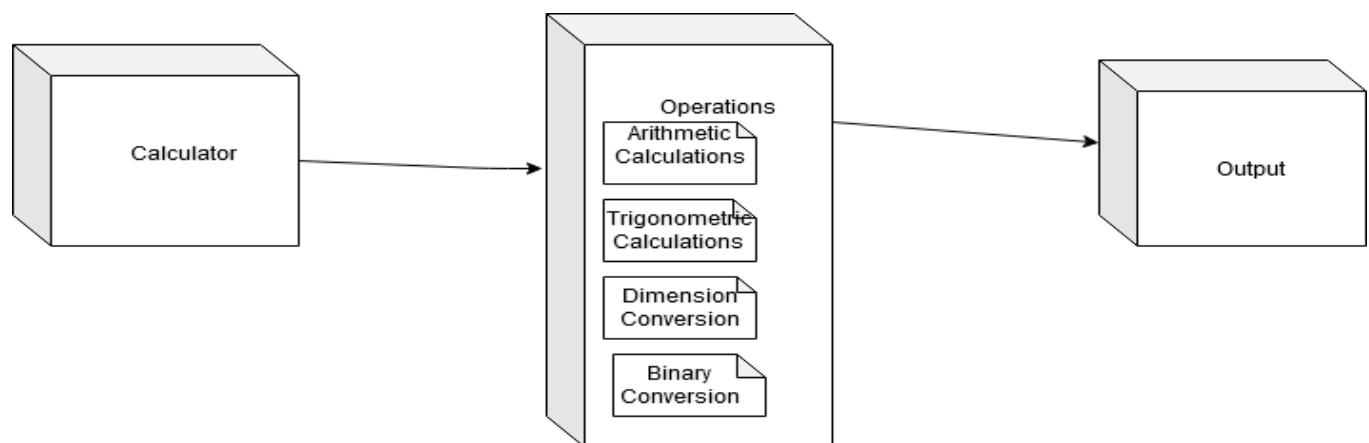
HLR 04	Bin https://www.yammer.com/lnttsgroup.onmicrosoft.com/#/Threads/show?threadId=1165927078649856 ary Conversion	Implemented
--------	--	-------------

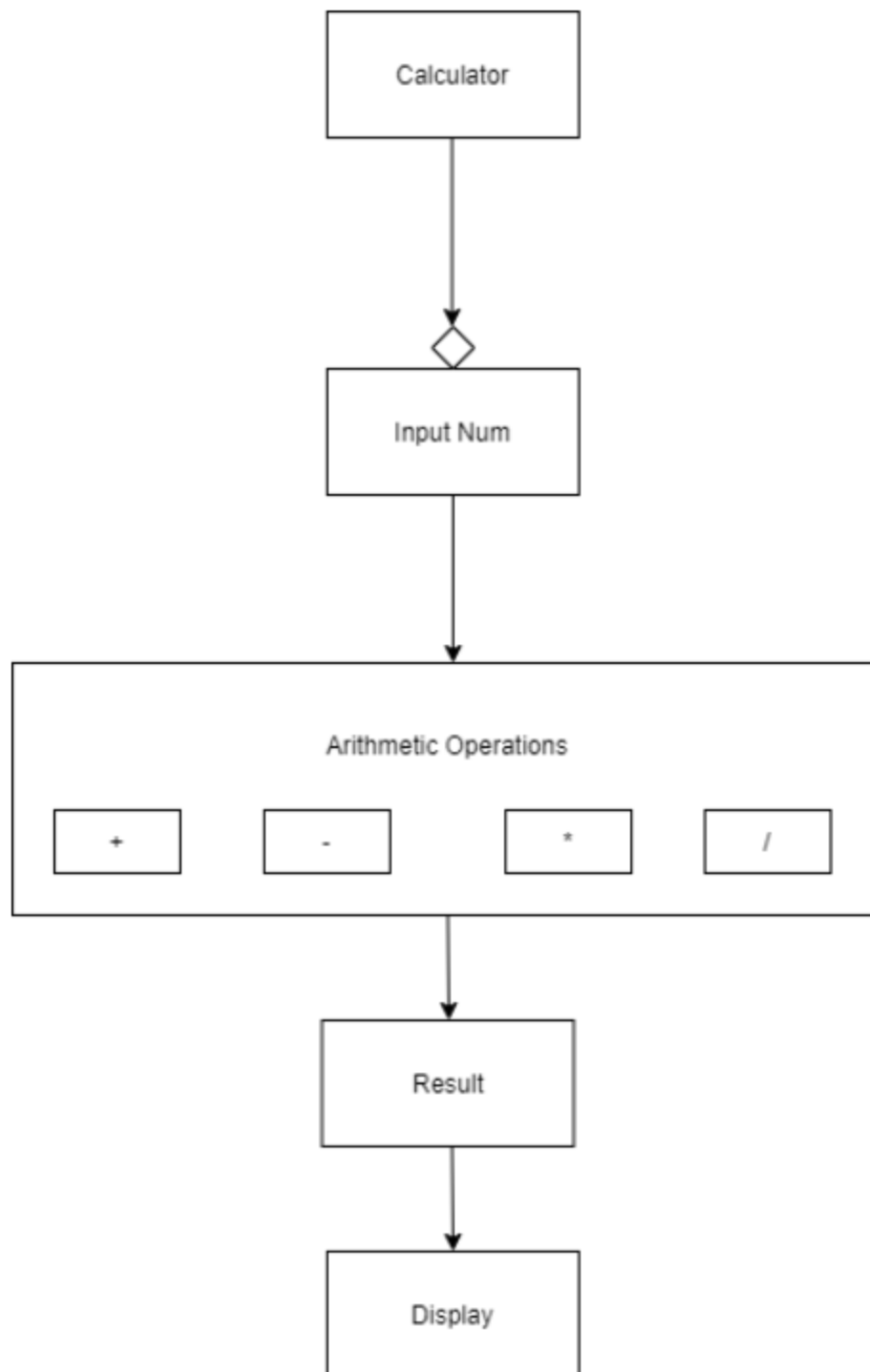
1.4.1 Low Level Requirement Analysis -

ID	Description	Status
LLR01	For Arithmetic Conversion Addition Subtraction Multiplication Division modulus	Implemented
LLR02	For Trigonometric Conversion, Sine, Cosine, Tan, Cot, Sec, Co-sec.	Implemented
LLR03	For Binary Conversion, Binary to decimal and hex, Decimal to hex and binary, Conversion range of word size.	Implemented
LLR04	Dimension Conversion, Length conversion, Mass conversion, Temperature conversion, Floating values.	Implemented

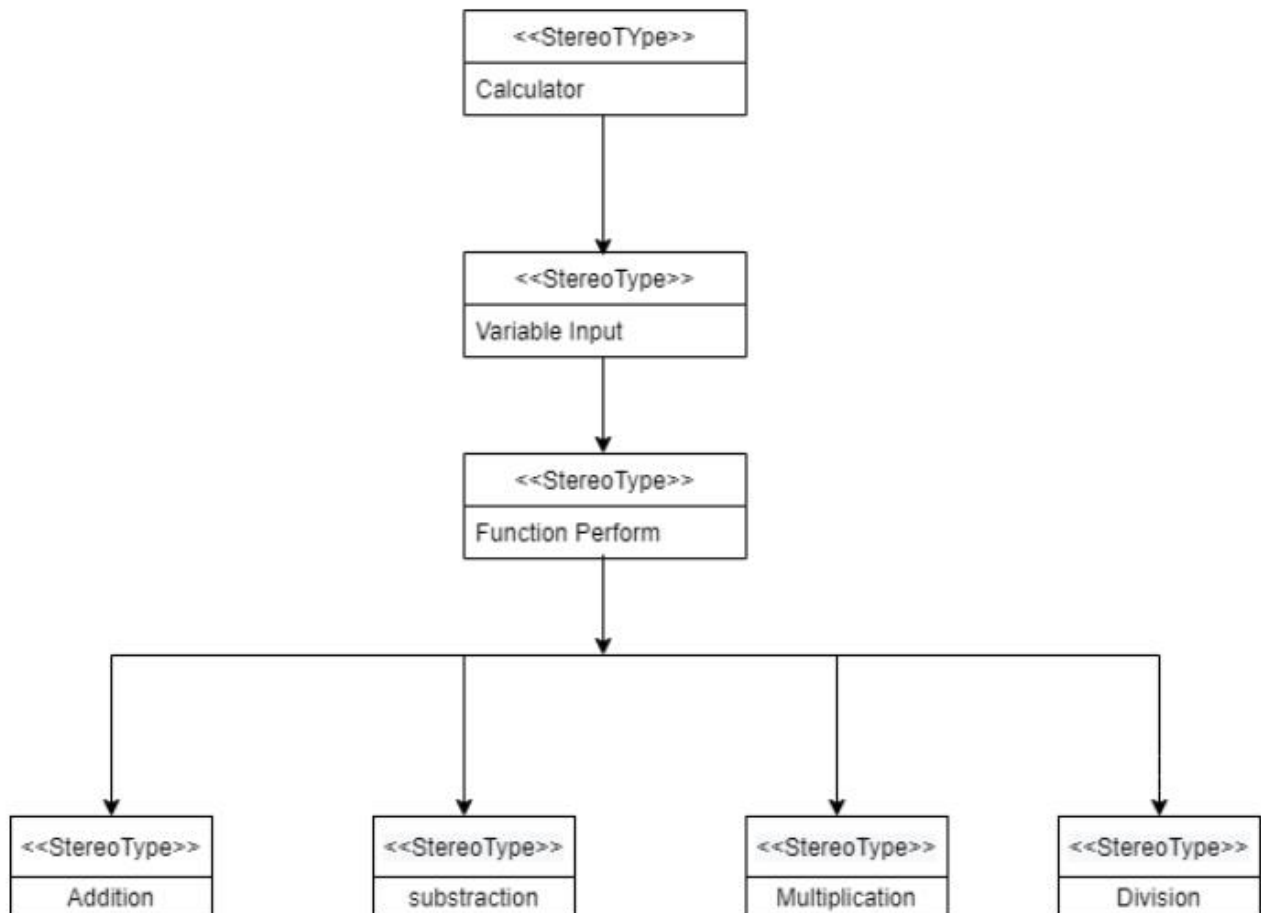
1.5 Design

1.5.1 High level diagram: Deployment Diagram

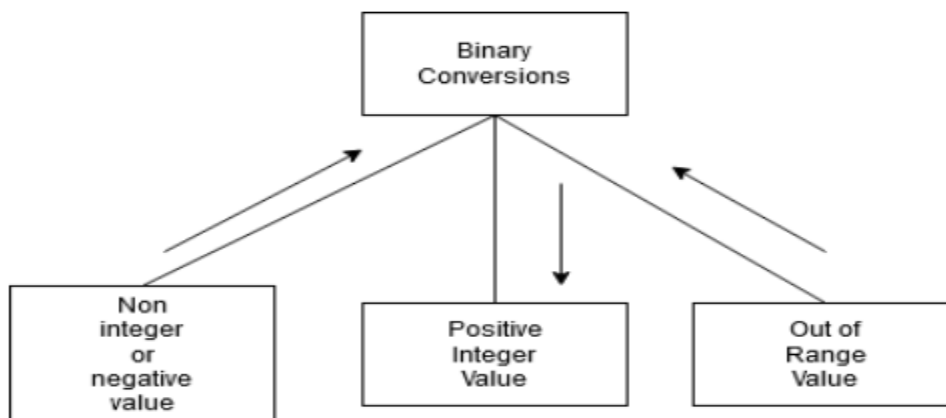


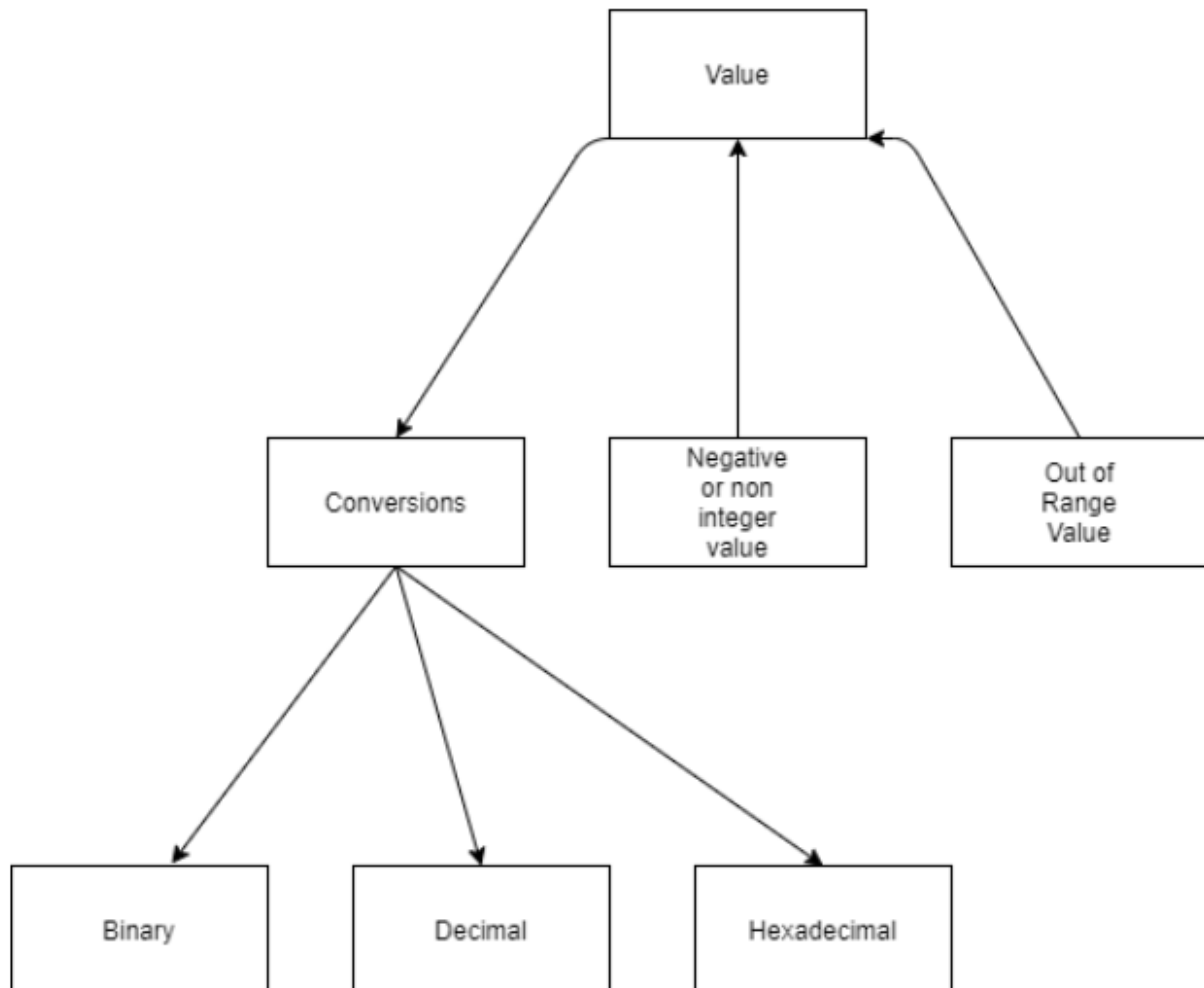
1.5.2 LLR Diagram: Composite diagram.

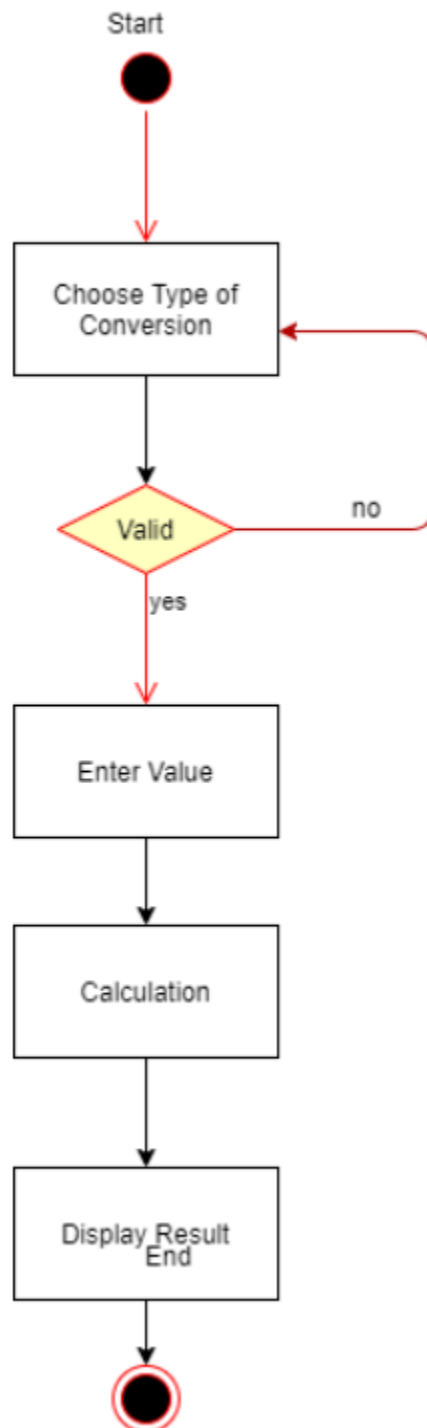
1.5.3 LLR Diagram: Profile Diagram -



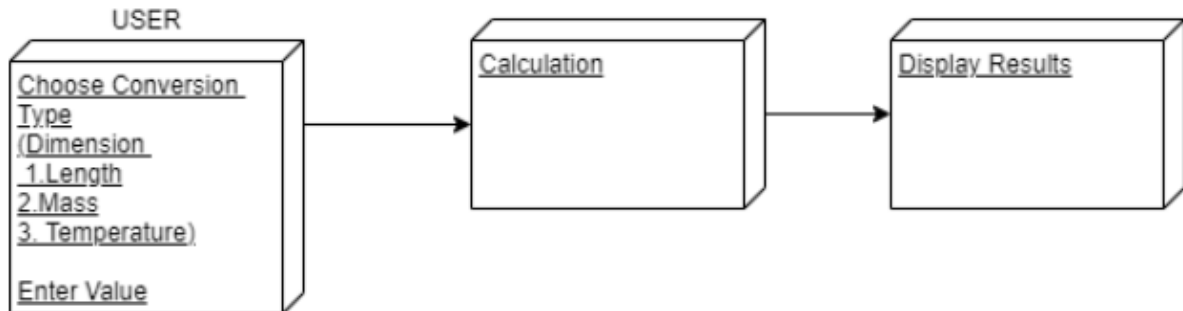
1.5.3 LLR Diagram : Communication Diagram -



1.5.4 LLR Diagram: Object diagram –

1.5.5 LLR Diagram: Activity diagram -

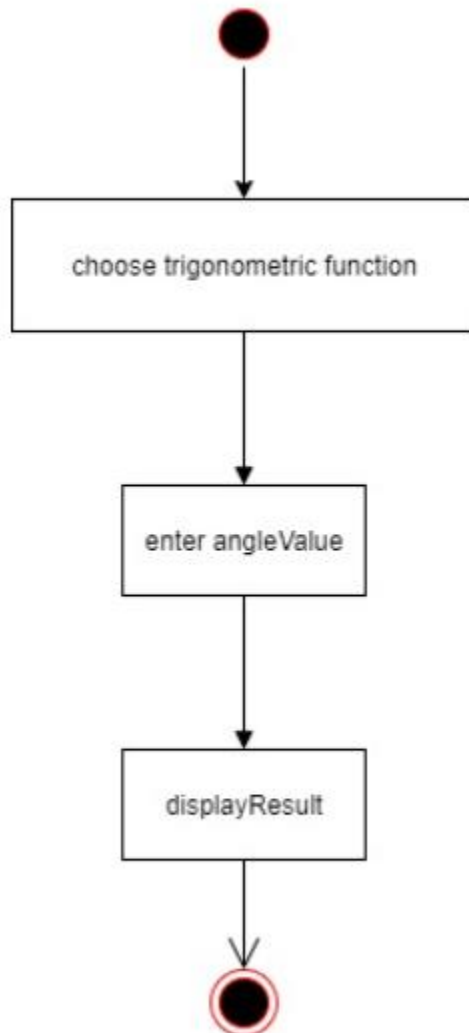
1.5.6 LLR Diagram: Deployment diagram -



1.5.7 LLR Diagram: UML Class diagram.



1.5.8 LLR Diagram: State diagram.



1.6 Test Plan

1.6.1 High Level Test Plan

Test ID	Description	Expected input	Expected Output	Actual Output	Type of Test
H_01	Perform Trigonometric Calculation	4	Perform Trigonometric Calculations Based on the Input	Getting right output	Scenario Based

1.6.2 Low Level Test Plan

Test Id	Input	Expected output	Actual output	Status(pass/fail)
T1	Sine 30	0.5		
T2	111(in binary)	7(decimal), 7(hex)		
T3	gm to Kg(1000g)	1 Kg		
T4	Addition (15,8)	23		
T5	Division (18,9)	2		

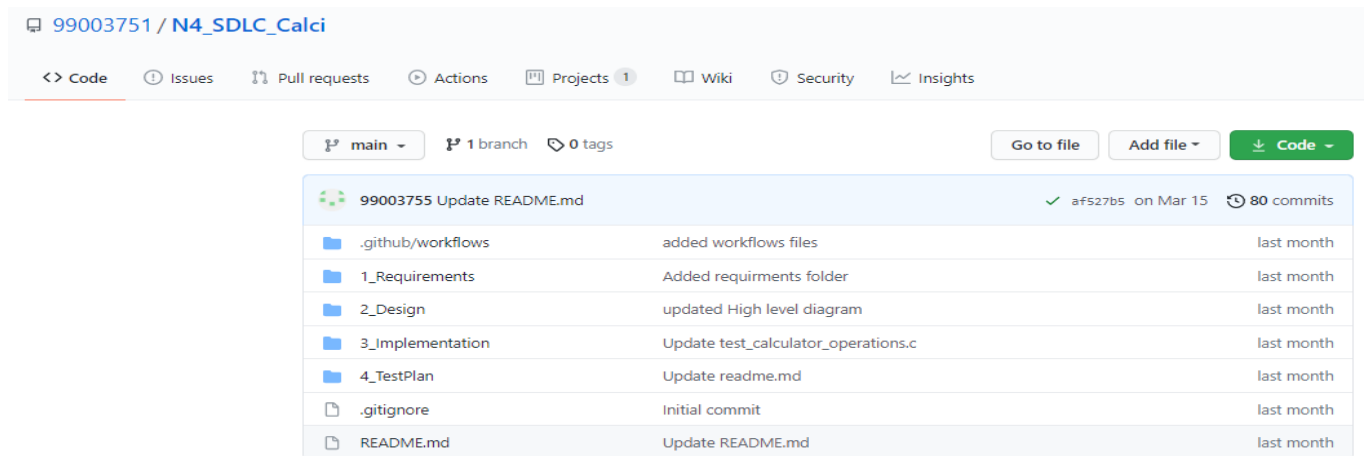
1.7 Implementation Summary

It is a basic calculator that will allow users to perform operations in Mathematics Addition, Subtraction, Multiplication, Division, Trigonometry, Factorial, Area, Volume etc. However, the input has to be in the form "number1 operator1 number2 operator2 number3" (i.e 2+4*10). The input values can be from any integer to even a number with decimals. Moreover, this calculator is smart enough to operate multiplication/division before addition/subtraction, in another word it is implemented with the order of precedence logic.

1.8 Git Link

https://github.com/99003751/N4_SDLC_Calci.git

1.9 Git Dashboard



99003751 / N4_SDLC_Calci









<> Code Issues Pull requests Actions Projects 1 Wiki Security Insights

main 1 branch 0 tags Go to file Add file Code

99003755 Update README.md ✓ af527b5 on Mar 15 80 commits

File	Description	Time
.github/workflows	added workflows files	last month
1_Requirements	Added requirments folder	last month
2_Design	updated High level diagram	last month
3_Implementation	Update test_calculator_operations.c	last month
4_TestPlan	Update readme.md	last month
.gitignore	Initial commit	last month
README.md	Update README.md	last month

1.9.1 Badges

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

1.9.2 Git Inspector

```

✓ Run gitinspector

1  ▶ Run source ~/.profile
11 gitinspector 0.5.0dev
12 Copyright © 2012-2015 Ejwa Software. All rights reserved.
13 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
14 This is free software: you are free to change and redistribute it.
15 There is NO WARRANTY, to the extent permitted by law.
16
17 Written by Adam Waldenberg.
18 Running for commit : c5a56714c268adcb9032a3542b6a9ceff0ecb741
19 ===== Git Inspector =====
20 Statistical information for the repository 'N4_SDLC_Calci' was gathered on
21 2021/03/15.
22 The following historical commit information, by author, was found:
23
24 Author                Commits  Insertions  Deletions  % of changes
25 99003751                15       636        65         11.63
26 99003753                11       155        34          3.13
27 99003754                17      4304        41         72.06
28 99003755                22       772        23         13.18
29
30 Below are the number of rows from each author that have survived and are still
31 intact in the current revision:
32
33 Author                Rows     Stability  Age         % in comments
34 99003751                540      84.9       0.0         10.00
35 99003753                104      67.1       0.0          6.73
36 99003754               4141     96.2       0.0          9.64
37 99003755                677     87.7       0.0         15.81
38

```

1.9.3 Setup for Build

```
name: TEST - Build Status


on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:


    runs-on: ubuntu-latest

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



1.9.4 Outcome of the Build

 Summary

Jobs

 build

Triggered via push last month

 99003755 pushed  af527b5 main

Status

Success

Total duration


23s

Artifacts

–

makerun.yml

on: push

 build


8s

1.9.5 Setup for Code Quality


```
19 lines (15 sloc) | 312 Bytes

1  name: Cppcheck
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  jobs:
10   cppcheck:
11
12     runs-on: ubuntu-latest
13
14     steps:
15     - uses: actions/checkout@v2
16     - name: Install cppcheck
17       run: sudo apt -y install cppcheck
18     - name: Run cppcheck
19       run: cppcheck 3_Implementation
```



1.9.6 Outcome of Code Quality

 Summary

Jobs

 build

Triggered via push 1 hour ago

 99003763 pushed  e8521d1 main

Status

Success

Total duration


44s

Artifacts

-

gcov.yml

on: push

 build 27s

1.9.7 Setup for Unity Testing

17 lines (13 sloc) | 248 Bytes

```
1  name: Unit Testing - Unity
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  jobs:
10   test:
11
12     runs-on: ubuntu-latest
13
14     steps:
15     - uses: actions/checkout@v2
16     - name: make
17       run: make -C 3_Implementation/ test
```

1.9.8 Outcome of Unity Testing

✓ Update README.md Unit Testing - Unity #152

🏠 Summary

Jobs

✓ test

Triggered via push 1 hour ago

👤 99003763 pushed → e8521d1 [main](#)

Status

Success

Total duration

23s

Artifacts

—

unity.yml

on: push

✓ test

5s

1.10 Individual Contribution & Highlights

- Trigonometry functionalities implemented.
- Test case for the same is implemented.
- High level and low-level test cases is implemented for the same.
- Issue raised and the issue was solved.
- Helped during the workflow's implementation of the project.

Contributors List and Summary

PS Number	Name	Feature
99003751	Shrinidhi V Katti	Trigonometric Operation(Sin, Cos, Tan, Cot, Cos, Co-sec)
99003753	Akshansh Mishra	Arithmetic Operation(Addition, Subtraction, Multiplication, Division, Remainder)
99003754	Rishab Ostawal	Binary Conversion(Decimal, Hexadecimal)
99003755	Rohan Roy	Dimension Conversion(Length, Mass, Temperature)

1.11 Summary

1.11.1 Outcomes:

Technical:

- Improved implementation of 'C' concepts.
- Practical implementation of SDLC lifecycle.
- Source code management. (Github)

Soft skills:

1. Project management
2. Conflict management.

1.12 Challenges faced and how were they overcome

- Differentiation of high level and low level.
- Committing to GitHub, pull and push in GitHub.
- Converting pictures & tables into readme file.
- Cpp check and Unity testing.

2. Python Programming Project

2.1 Modules Used

Modules used in this project are Core and Advanced Python.

2.2 Project title: Retrieve data from multiple Excel sheet

2.3 Topic and Subtopics

2.4 Objectives:

To extract the data present in different spreadsheets in one excel file as required by the user.

2.5 Requirements:

2.5.1 High Level requirement analysis

Id	Requirements	Description	Status
HL1	Searching Data	Search all data from 5 sheets when user defines the PS number to be searched.	Implemented
HL2	writing to excel	Write all the data from different spreadsheet in one master sheet	Implemented
HL3	extracting user defined data	Write required data in the excel file.	Implemented
HL4	plotting the bar chart	plot the bar chart of the data present in the master sheet .	Implemented

2.5.2 Low Level requirement Analysis -

Id	Requirements	Description	Status
LL1	Search Parameters	The user defines the Name or PS Number or Email Id of the data to be searched	Implemented
LL2	Searching Data in excel file in every sheet	The data to be searched is defined by the user.	Implemented
LL3	writing the data into master sheet	Data defined by user has to be extracted from 5 different spreadsheets and put into one master sheet.	Implemented
LL4	Plotting the bar chart	plot the bar chart of the data present in the master sheet considering rows and columns.	Implemented

2.6 Design

2.6.1 Use Case LLR Diagram

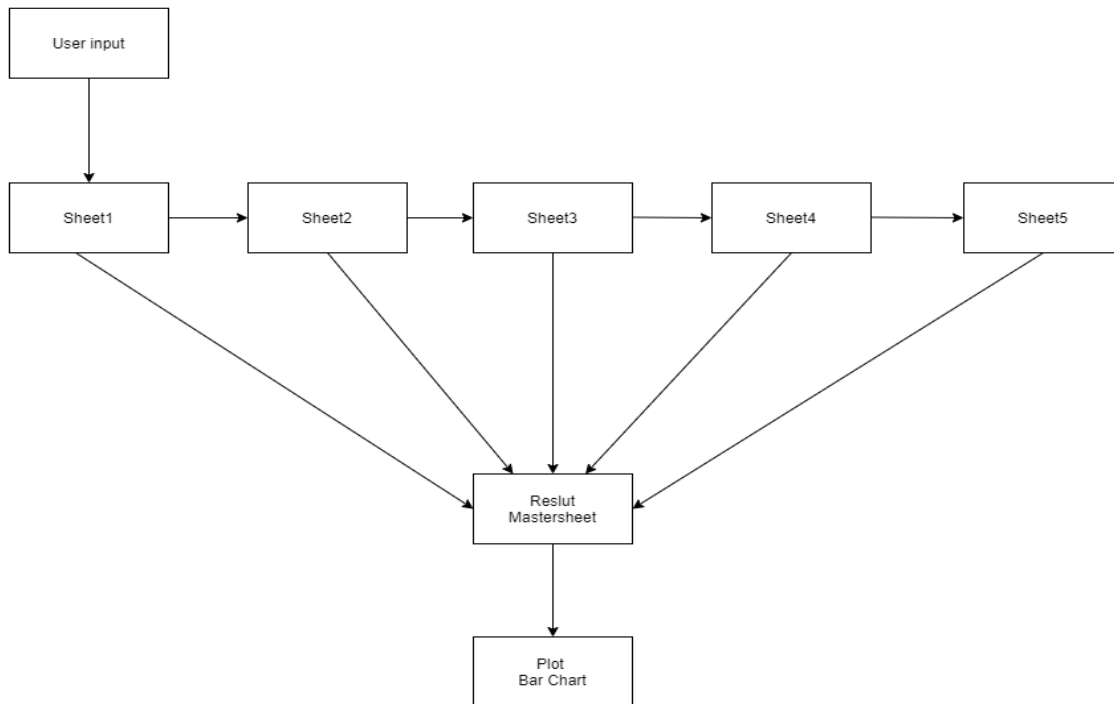


Figure : LLR Diagram

2.6.2 Object HLR Diagram

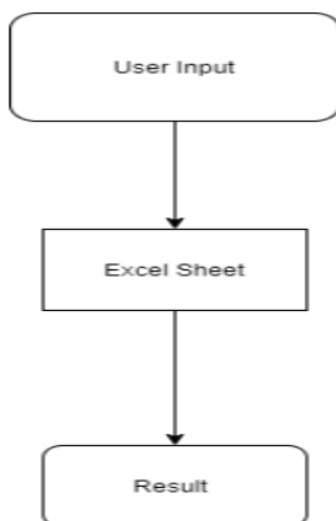


Figure : HLR Diagram

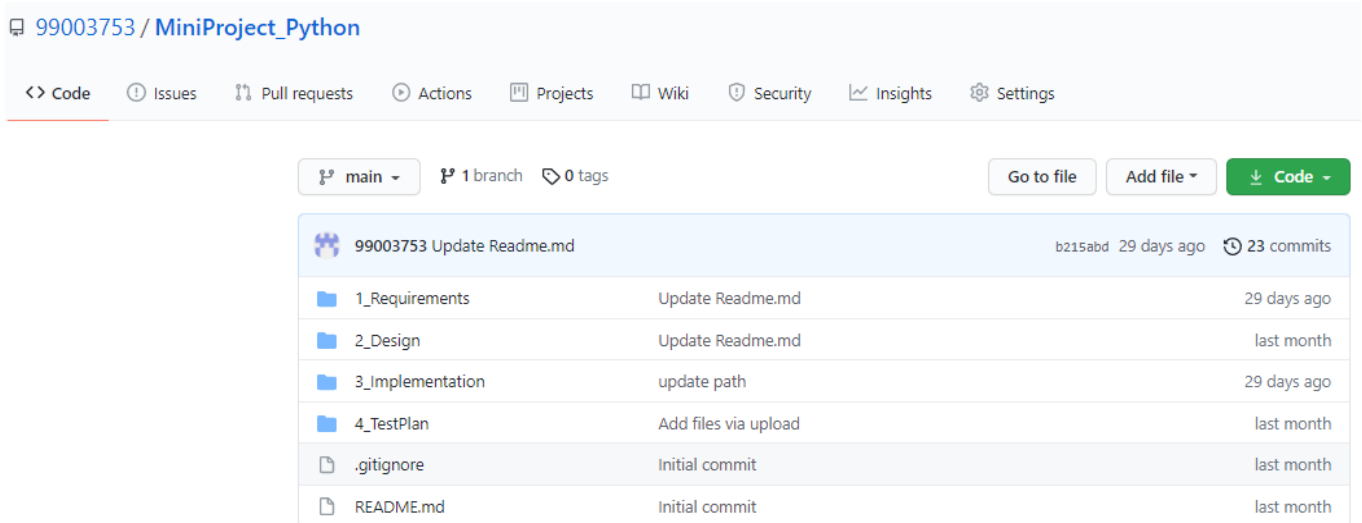
2.7 Implementation Summary

The aim of the project is to extract the data present in different spreadsheets in one excel file as required by the user. The excel sheet consists of 5 spreadsheets. The user defines the data that needs to be searched on the basis of Name, PS Number and Email ID. The python program then reads the data corresponding to the particular data from different spreadsheets of excel. It then creates a master sheet and adds the data from all the sheets to it. In the end, it will plot the bar graph from the data present in the master sheet.

2.8 Git Link

https://github.com/99003753/MiniProject_Python

2.9 Git Dashboard



99003753 / MiniProject_Python

<> Code ! Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki 🛡️ Security 📊 Insights ⚙️ Settings

main 1 branch 0 tags Go to file Add file Code

File/Folder	Commit Message	Commit Hash	Time
99003753 Update Readme.md		b215abd	29 days ago
1_Requirements	Update Readme.md		29 days ago
2_Design	Update Readme.md		last month
3_Implementation	update path		29 days ago
4_TestPlan	Add files via upload		last month
.gitignore	Initial commit		last month
README.md	Initial commit		last month

2.10 Summary -

2.10.1 Outcomes:

Technical:

- Improved implementation of Python concepts.
- Practical implementation of SDLC lifecycle.
- Source code management. (GitHub)

Soft skills:

1. Project management
2. Conflict management.

2.11 Challenges faced and how were they overcome

- System issues(crashing and Interfacing).
- Differentiation of high level and low level.
- Committing to GitHub, pull and push in GitHub.
- Converting pictures & tables into readme file.

3.Embedded C

3.1 Module:

The modules used in this are SDLC, Embedded C and was implemented on the hardware STM32.

Topic and Subtopics

- The Car feature requirements for sub system was found.
- The window, seat and lighting system of car was developed.
- The code was dumped on the STM32 board.

3.2 Objectives & Requirements

3.2.1 HIGH LEVEL REQUIREMENTS:

- Adjustment of seat with the help of buttons in two direction
- Forward direction: when the input is 1
- Reverse direction: when the input is 0

3.2.2 LOW LEVEL REQUIREMENTS:

- When the input is 1 LED glows in clockwise direction
- When the input is 0 LED glows in anti-clockwise direction

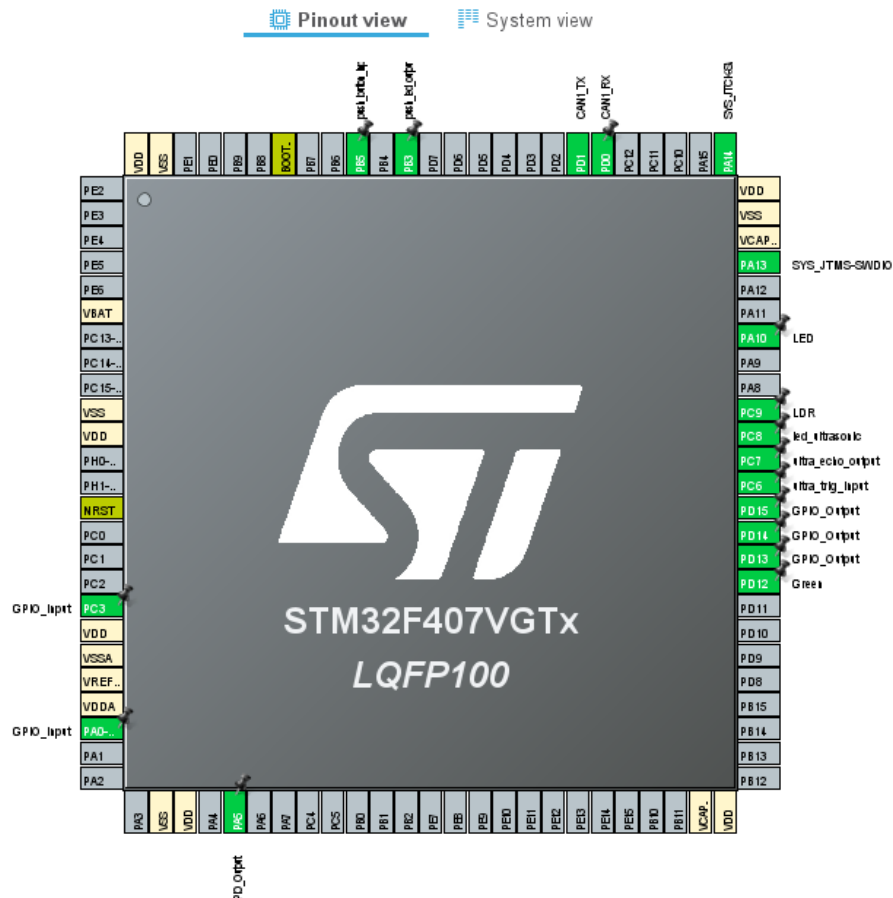
3.3 Test Plan

SL No	TEST_ID	Testing function	Expected input	Expected Output
1	HLT_1	Wiper control system.	When switch is pressed	Wiper starts
2.	HLT_2	Interior door light	When push button is pressed	Door light turns off indicating all doors are locked.
3.	HLT_3	Power window module	When switch is pressed	Window opens

4.	HLT_4	Seat belt warning system	When switch is open	Buzzer will be on
5.	HLT_5	Side mirror control system.	When button is pressed	Side mirror rotates (inwards and outwards)
6.	HLT_6	Automatic head light system.	Light intensity to LDR sensor	Head light turns on.

Table 3: High Level Test plans

3.4 Pin Config –



3.5 Implementation Summary

The code for seat system, Window system and lighting system was written by each team member respectively and finally was integrated at the end. The integrated code was dumped on the STM32 board. The individual control of the systems was working properly.

```
/******
```

```
if(HAL_GPIO_ReadPin(PA0_GPIO_Port, PA0_Pin)==1){  
  
    HAL_GPIO_TogglePin(PD12_GPIO_Port, PD12_Pin);  
    HAL_Delay(100);  
    HAL_GPIO_TogglePin(PD13_GPIO_Port, PD13_Pin);  
    HAL_Delay(100);  
    HAL_GPIO_TogglePin(PD14_GPIO_Port, PD14_Pin);  
    HAL_Delay(100);  
    HAL_GPIO_TogglePin(PD15_GPIO_Port, PD15_Pin);  
    HAL_Delay(100);  
  
}else{  
    //pass comment  
}
```

Figure : Basic logic for Implementation

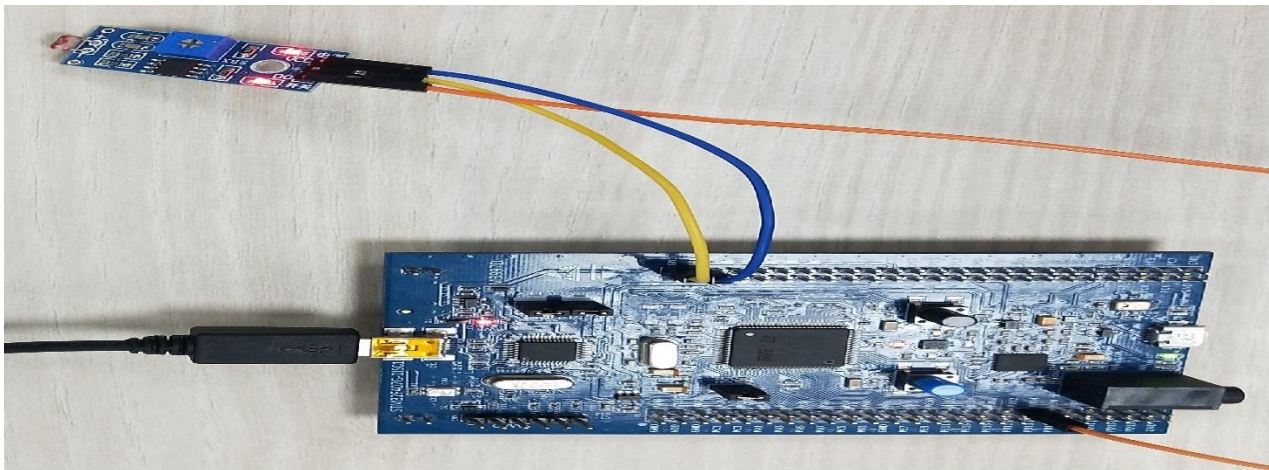


Figure : Use of the LDR sensor

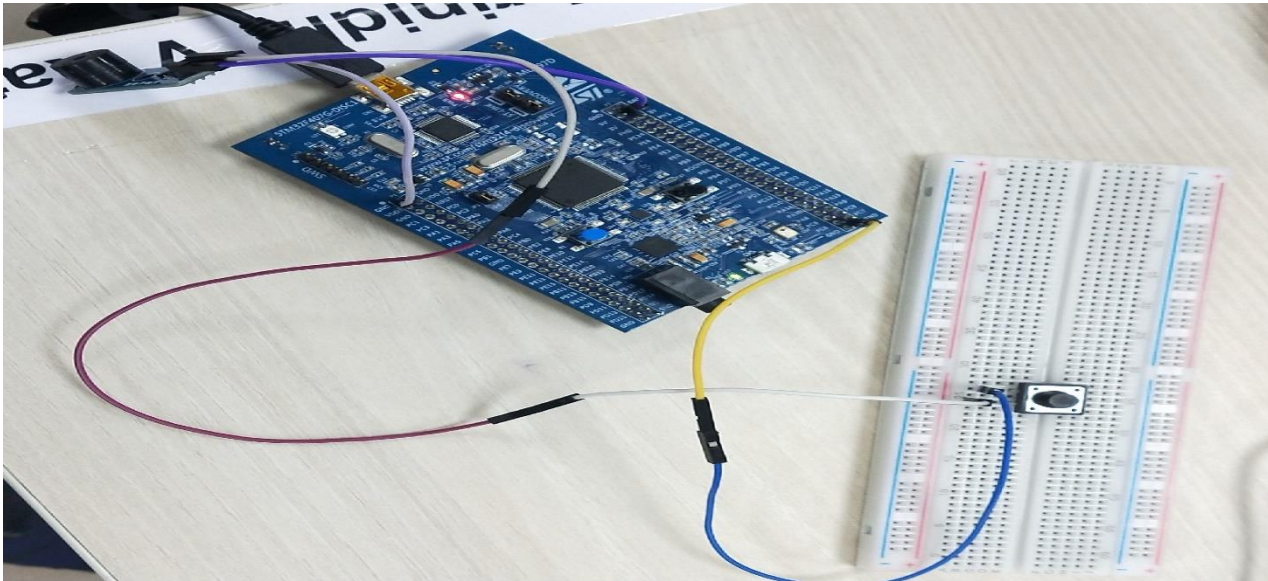


Figure : Switch and push button the Implementation

3.6 Individual Contribution & Highlights

1. Interior door light system.
2. Wiper Control module.
3. Power window.
4. Seat belt warning system.
5. Side mirror control system.
6. Automatic head light control system.

3.7 Summary

The code was written individually and integrated together by one of the teammate and code was dumped on the hardware. The output of each system was show in the form of LEDs.

Challenges faced and how were they overcome

- Controlling the function of Windows through code.
- With less components showing the output was challenging.

4.0 Miniproject -4 [Individual] – Kernel Programming and Device Drivers -

4.1 Module :

The modules used in this are Linux and Kernel Device drivers. Kernel programming is an advanced topic that requires in-depth study of the source code for the Linux kernel.

4.2 Topic and Subtopics:

- Basic Linux commands.
- Qemu Based Emulation.
- Creation of SD card.
- Building custom Kernel.
- Cross Compilation.
- Static and dynamic libraries.
- System calls.
 - Adding system calls in kernel space.
 - Invoking system calls from user space.
- Kernel modules.
 - In-Tree modules: Dynamic.
 - In-tree modules: static.
- Basics of Kernel Device Drivers.
- Registering Char Driver.
- Kernel Data Structure.
 - Kfifo API.
 - List API.
- IPC Kernel
 - Concurrency.
 - Kernel Threads.
 - Locking and Synchronization.
 - Mutex.
 - Semaphore.
 - Spinlocks.
 - Wait queues.
- IOCTL.
- Driver model.

4.3 Objectives & Requirements:

The main objective of this module is to apply the concepts of Linux kernel, kernel device drivers to develop:

- Custom kernel.
- Create char drivers.
- Developing cross compiled code for target qemu.
- Creating own system calls.

4.3.1 Requirements:

- Basic Linux commands.
- Programming in Linux Environment.
- Custom kernel.
 - zImage
 - vexpress-v2p-ca9.dtb
 - rootfs.img
- Operating system Basics.
- IPC concepts.
- Concurrency.
- File handling using system calls.
- Virtual Memory concept.

4.4 Implementation Summary:

4.4.1 Hands-on Activity that are implemented are as follow:

- Register char driver
- Register file operations
- Device Create, Class Create
- Read, write operations using global buffer
- Read, write operations using kfifo.
- ioctl operations, returning length/remaining space, reset operation
- ioctl operations - filling length/remaining space in structure
- synchronization in char driver - using wait queue

4.4.2 User space code:

- simple read, write
- multiple read, multiple write
- User space code for IOCTL operations

4.4.3 kthread examples:

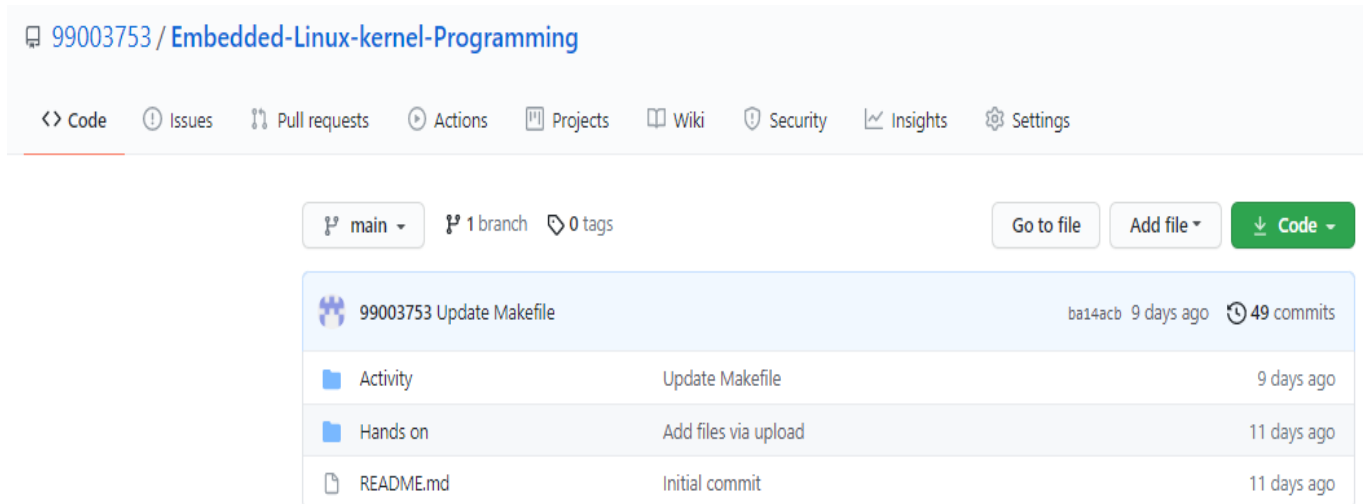
- simple two threads
- Race condition scenarios

- Mutual exclusion using semaphore, mutex, spinlock
- Synchronization using semaphores, wait queues
- Device Tree based platform driver code -- dummy UART
- Activity that are implemented are as follow:
- System calls -- echo back the given string.
- System calls—traverse process list print pid and ppid.
- System calls—length of string.
- System calls—taking simple parameter.
- IOCTL operation traverse the list.

4.5 Git Link:

<https://github.com/99003753/Embedded-Linux-kernel-Programming>

4.6 Git Dashboard



The screenshot shows the GitHub repository page for user 99003753, repository Embedded-Linux-kernel-Programming. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there are buttons for 'main' branch, '1 branch', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and 'Code'. The repository details show the commit hash 'ba14acb' from '9 days ago' with '49 commits'. A table lists the repository's history:

Commit Hash	Commit Message	Time Ago
ba14acb	Update Makefile	9 days ago
	Add files via upload	11 days ago
	Initial commit	11 days ago

4.7 Summary:

In this project custom system calls for a particular kernel is made by modifying internal syscalls.h, syscall.tbl, kernel /Makefile and its definition in c file in kernel folder of kernel source.

In user-space code of the system call a special system call number is mentioned to use the custom system call which is defined system call table (syscall.tbl). Finally, it's test on serial console and VGA console according to expected input and output.

4.8 Challenges faced and how were they overcome:

- Unable to directly access string in kernel space from user space and vice-versa – Using `copy_from_user()` and `copy_to_user()` solved this issue.
- Traversing through system process list was an issue- It was solved by using `for_each_process()` and `task_struct`.
- Traversing through node list was issue that was resolved using `list_for_each()` method.