./

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](#_Toc69784743)

[Miniproject -1 [Team] 5](#_Toc69784744)

[Module: 5](#_Toc69784745)

[Topic and Subtopics: 5](#_Toc69784746)

[Objective: 5](#_Toc69784747)

[Requirements: 5](#_Toc69784748)

[Design 7](#_Toc69784749)

[Test Plan 9](#_Toc69784750)

[Implementation Summary 9](#_Toc69784751)

[Video Summary 9](#_Toc69784752)

[Git Link 10](#_Toc69784753)

[Git Dashboard 10](#_Toc69784754)

[Summary 10](#_Toc69784755)

[Individual Contribution & Highlights 14](#_Toc69784756)

[Summary 14](#_Toc69784757)

[Challenges faced and how were they overcome 14](#_Toc69784758)

[Future Scope (If applicable) 14](#_Toc69784759)

[Miniproject -2 [Team] 15](#_Toc69784760)

[Module 15](#_Toc69784761)

[Topic and Subtopics 15](#_Toc69784762)

[Objectives 15](#_Toc69784763)

[Requirements 15](#_Toc69784764)

[Components Used: 16](#_Toc69784765)

[Design 17](#_Toc69784766)

[Circuit Implementation: 18](#_Toc69784767)

[Test Plan 19](#_Toc69784768)

[Implementation Summary 20](#_Toc69784769)

[Git Link 21](#_Toc69784770)

[Git Dashboard 21](#_Toc69784771)

[Summary 21](#_Toc69784772)

[Challenges faced and how were they overcome: 22](#_Toc69784773)

[Miniproject -3 [Individual] 23](#_Toc69784774)

[Module 23](#_Toc69784775)

[Topic and Subtopics 23](#_Toc69784776)

[Objectives 24](#_Toc69784777)

[Requirements 24](#_Toc69784778)

[Design 24](#_Toc69784779)

[Test Plan: 26](#_Toc69784780)

[Implementation Summary 28](#_Toc69784781)

[Git Link 28](#_Toc69784782)

[Git Dashboard 28](#_Toc69784783)

[Summary 29](#_Toc69784784)

[Challenges faced and how were they overcome 29](#_Toc69784785)

[Future Scope 29](#_Toc69784786)

[Miniproject -4 [Individual] 30](#_Toc69784787)

[Module 30](#_Toc69784788)

[Topic and Subtopics 30](#_Toc69784789)

[Objectives & Requirements 30](#_Toc69784790)

[Test Plan 31](#_Toc69784791)

[Implementation Summary 31](#_Toc69784792)

[Git Link 32](#_Toc69784793)

[Git Dashboard 32](#_Toc69784794)

[Summary 32](#_Toc69784795)

[Challenges faced and how were they overcome 32](#_Toc69784796)

# Miniproject -1 [Team]

## **Module:**

“SDLC (System Development Life Cycle)”

### Topic and Subtopics:

* ***V-Model***
* C Programming
* Makefile
* Unit Testing
* Version Control (via GitHub)
* ***Agile-Methodology***
* Theme
* Epic
* User- Story

All these topics had been implemented using V-model.

## **Objective:**

Designing a basic calculator that performs basic functions as well as some specific functions as per requirements.

## **Requirements:**

***High Level Requirement:***

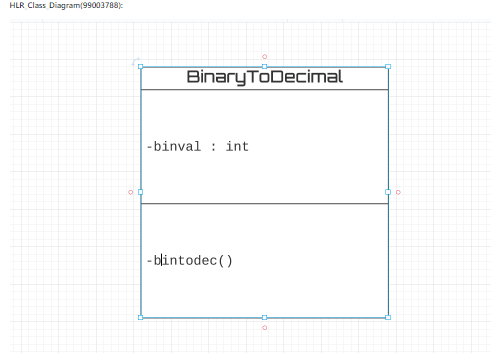
1. Maximum input digits a user can enter is up to 12 digits.
2. Arithmetic Operations
3. Exponential and Logarithmic Operations.
4. Square roots and powers.
5. Permutation and Combinations.
6. Finding area of different geometrical shapes like circle, square, rectangle, triangle.
7. Finding volume of different geometrical shapes like cube, cylinder, cone, sphere.
8. Trigonometric and Inverse Trigonometric Functions.
9. Measurements conversion (km, cm, inch, etc.).
10. weight conversion (kilogram, liter, etc.).
11. Mixed to improper fractions.
12. Cube and cube root.
13. Temperature conversion.
14. Arithmetic operations of fractions. (+,-,\*,/) .
15. Calculation of Reminder.
16. Calculation of simple interest.
17. Binary to decimal conversion

***Low Level Requirement:***

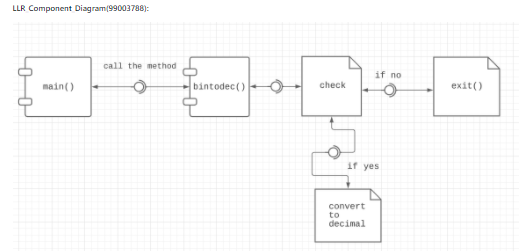
1. Exponential operation (result=b^x where b=base and x=exponent. Input type: integer and float) Logarithmic operation (result=log(x) where the input type of x is integer or float) .
2. Area of different geometrical shapes (The input data type used in this scenario are integer, float and long).
3. Volume of different geometrical shapes (The input data type used in the scenario are integer and float.)
4. Permutation and Combination (result=nPr; input type: integer and result=nCr; input type: integer).
5. Using float for Decimal values.
6. Using int for Integer values.
7. Using Double.
8. Calculation of reminder using modulus operator.
9. Using arithmetic operators for calculation of simple interest.

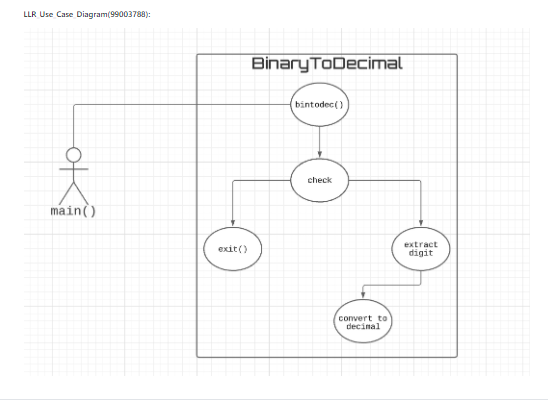
## **Design**

* High Level UML Diagram:

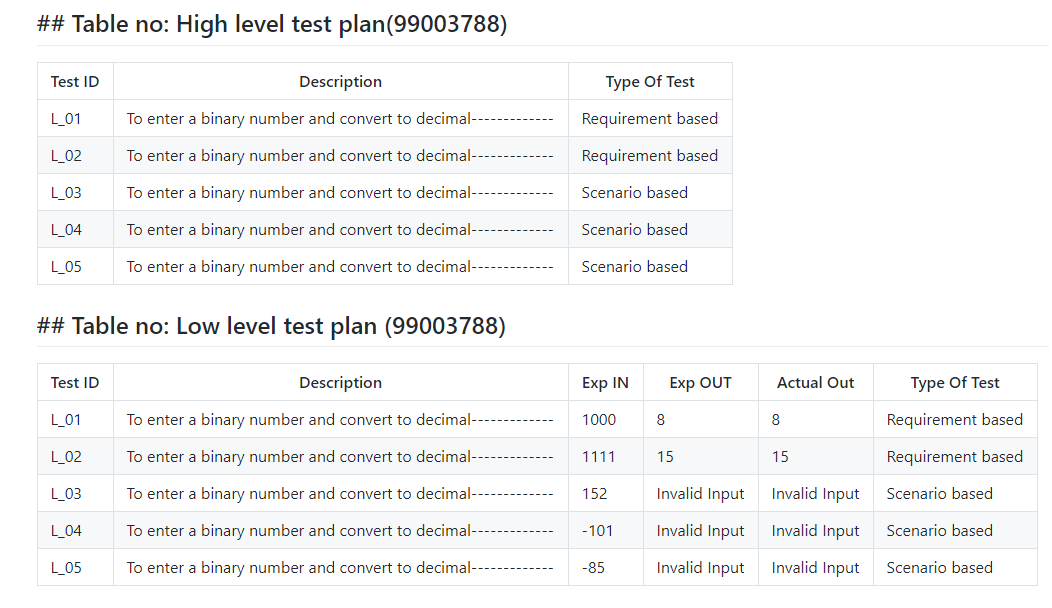


* Low Level UML Diagram:





## Test Plan



## Implementation Summary

Implementation folder had all source files, header files, test files for different features of the calculator such as Basic Arithmetic, Square root, cube root, exponent, logarithm, etc.

Here, **inc** folder holds all the header files with “.h” extension which contains prototype of all functions, structure definition, macro definition and definition of all the enumerators.

The **src** folder holds all the source files with “.c” extension which has definitions of all the functions whose prototype is defined in header files.

The **test** folder holds the ***test\_calculator\_operations.c*** file for cumulative testing of the source codes based on requirements, scenario and boundary.

Other than these folders, there is also a **unity** folder which holds prototypes and definition of the standard unity test case functions.

Also, there is **a Makefile** which builds, debugs using valgrind, check static and dynamic code quality, performs overall unit testing for all the codes together with the execution of single commands based on different defined targets.

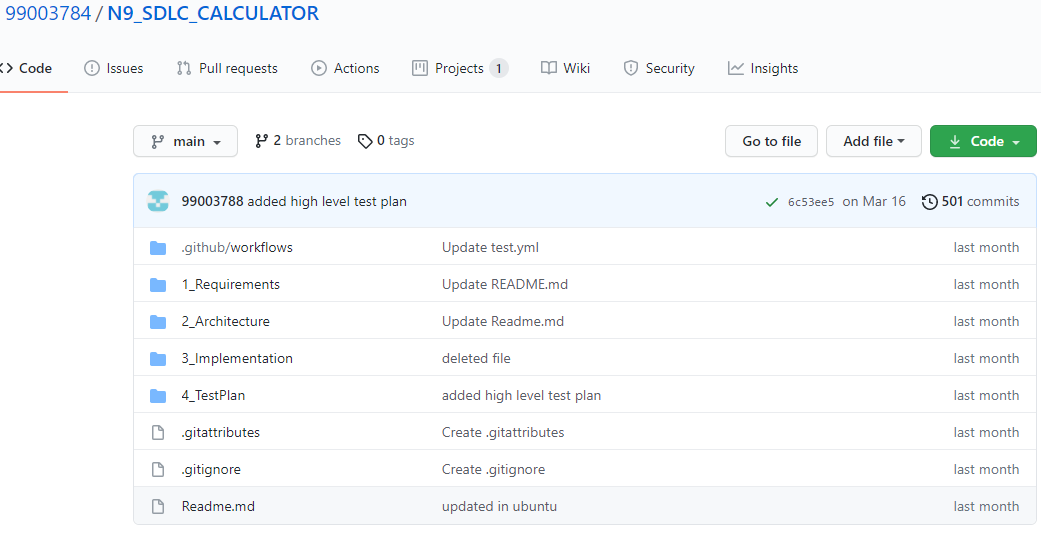
### Video Summary

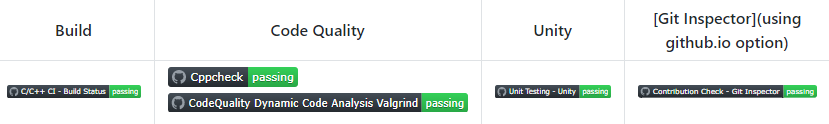
“Please upload a short video on the repo for the walkthrough of the project (Team/Individual) less than 7min and less than 30MB File Size. Start is the Standard opening slide with title of miniproject + Team members followed by the walkthrough”

### Git Link

<https://github.com/99003784/N9_SDLC_CALCULATOR.git>

### Git Dashboard

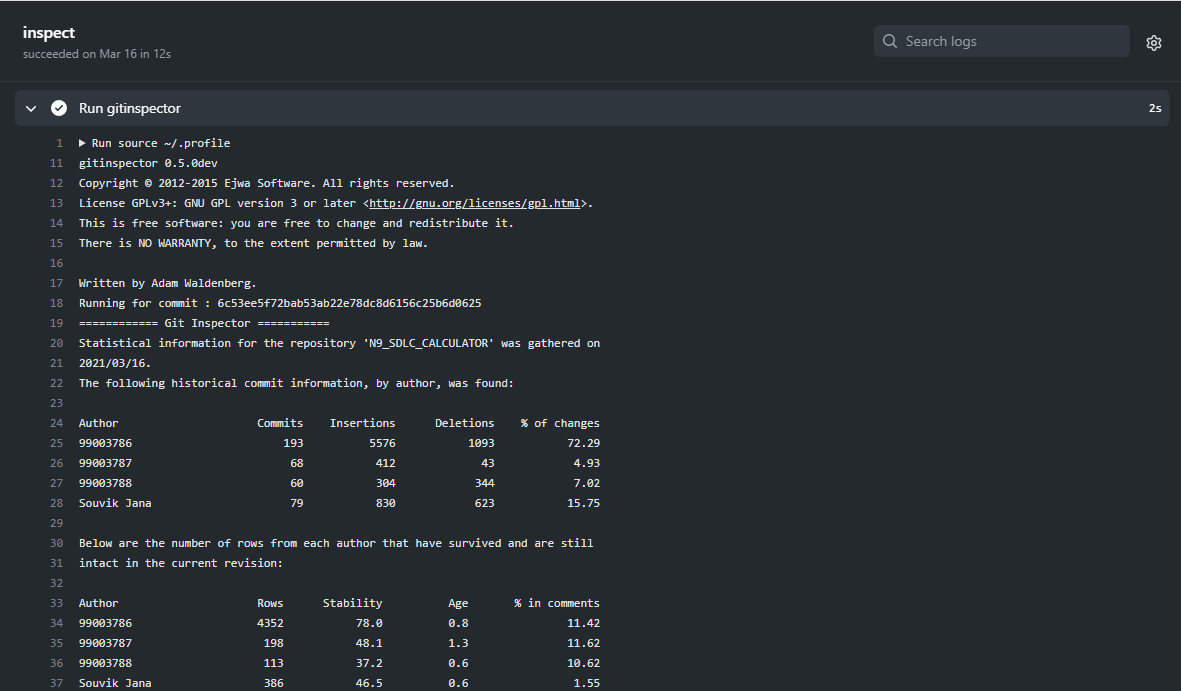
* ***GitHub Repo***
* ***Badges***

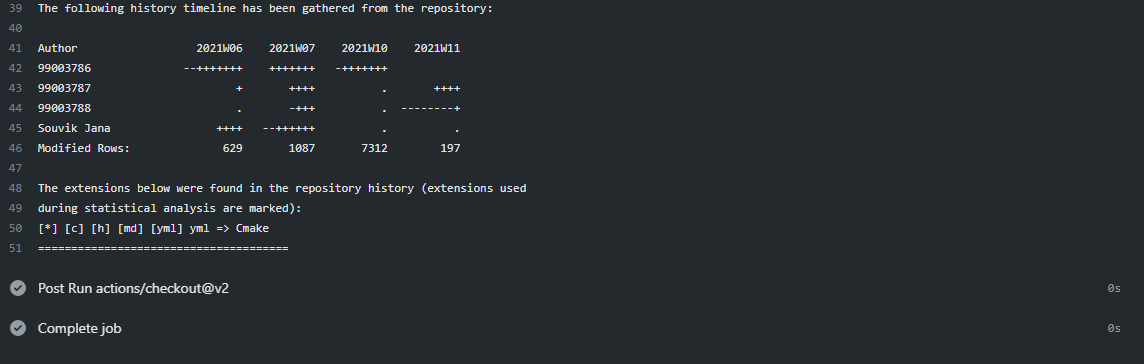
******

### Summary

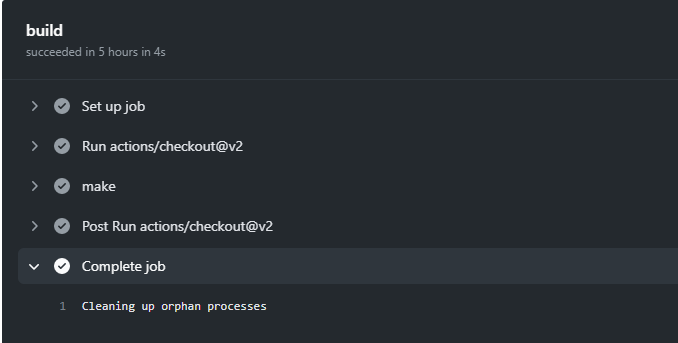
It is a simple electronic hardware/software device that can perform the simple calculations such as addition, subtraction, multiplication, division, calculating power of number, exponential function, logarithmic function, permutation and combination, trigonometry, inverse-trigonometric functions, factorial of a number, binary to decimal conversion etc.

#### Git inspector summary

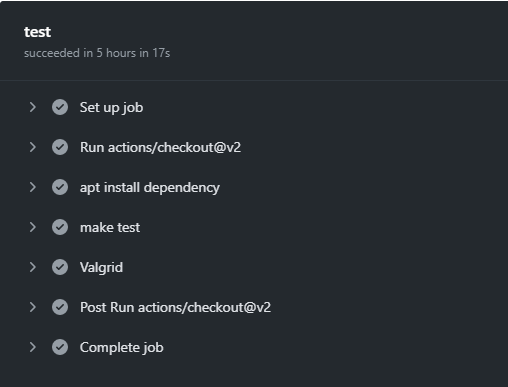


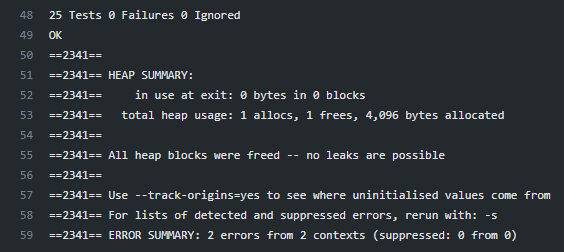


#### Build



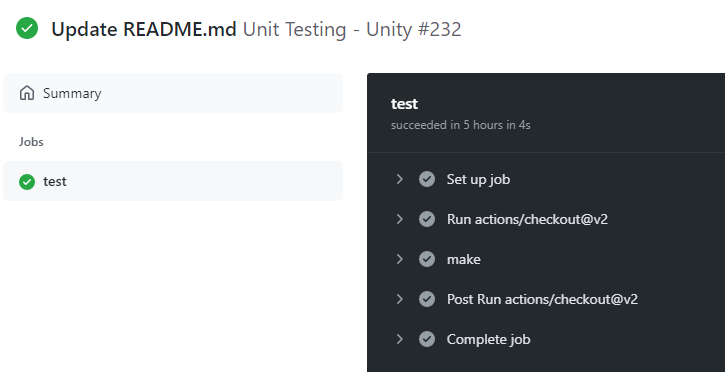
#### Code quality and Issues or Bug Tracking

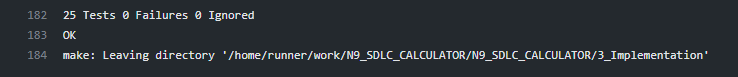
******



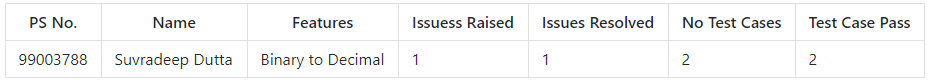
#### Unit Testing

“Unit Testing setup alignment with test plans and summary of outcome”





## Individual Contribution & Highlights

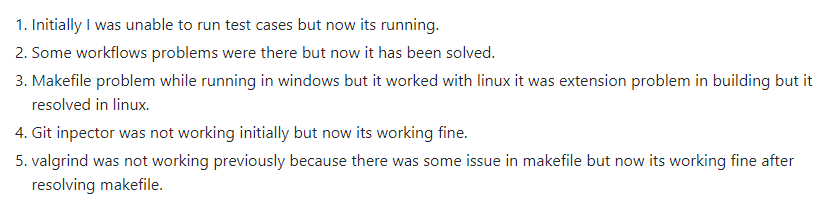


The function that I have implemented is conversion form binary to decimal. Here, first we need to accept a number and check whether it is greater than equal to 0 or not. If not then print “Invalid input”. If yes then the next step is to check whether the number is binary or not. If not the print “invalid Input”. If the number is binary, covert the binary number to its decimal equivalent. Finally print the decimal value.

### Summary

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

### Challenges faced and how were they overcome



### Future Scope (If applicable)

# Miniproject -2 [Team]

## Module

Embedded C

### Topic and Subtopics

* Driver API Development (Custom)

##### Sub-Topics

* GPIO
* ADC
* SPI, UART, I2C
* External interrupt.
* Debugging using STM-Board.
* Driver API Development (Hardware Level Abstraction - HAL)

##### Sub-Topics

* GPIO
* ADC
* External interrupt.
* Debugging using STM-Board.

## Objectives

To implement different CAR Module Features using STM32f407VG Microcontroller featuring 32-bit ARM-M4 with FPU-core.

CAR-Module Features: -

* Power Window
* Door Lock/Unlock

## Requirements

##### High-Level Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **Requirements** | **Description** | **Status** |
| 1 | Power Window | Indicates whether the window(s) of the cars is/are open or close. | Implemented |
| 2 | Door Lock/Unlock | Indicates whether the door(s) of the car is/are open or closed. | Implemented |

Table 5. High Level Requirements

##### Low-Level Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **Requirements** | **Description** | **Status** |
| 1 | When car window(s) is/are open. | GPIO pin PD13 is set and the LED on the discovery board glows. | Implemented |
| 2 | When car window(s) is/are closed. | GPIO pin PD13 is reset and the LED on the discovery board is off. | Implemented |
| 3 | When the door(s) of the car is/are open. | GPIO pin PD14 is reset and buzzer buzzes. | Implemented |
| 4 | When the door(s) of the car is/are close | GPIO pin PD14 is set and the buzzer is silent. | Implemented |

Table 6. Low-Level Requirements

## **Components Used:**

**a)** STM32f407VG Microcontroller

**b)** Breadboard

**c)** LED

**d)** LDR Sensor

**e)** Soil Sensor

**f)** PIR Motion detection sensor

**g)** RGB Color Sensor

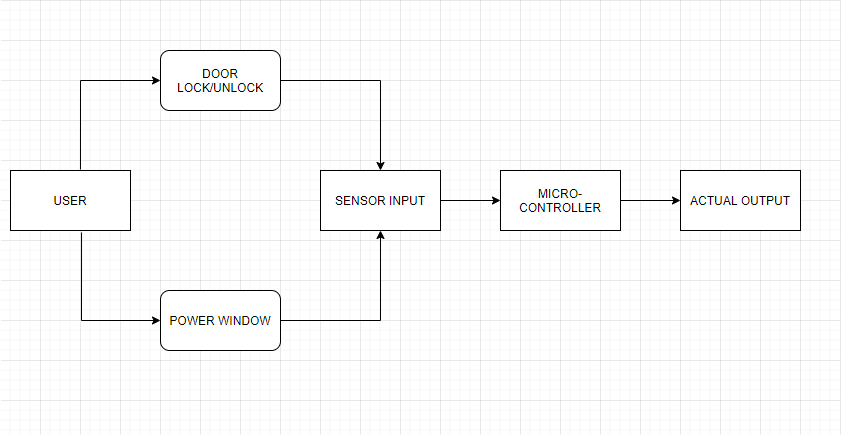
**h)** Potentiometer Sensor

**i)** Ring Buzzer Sensor

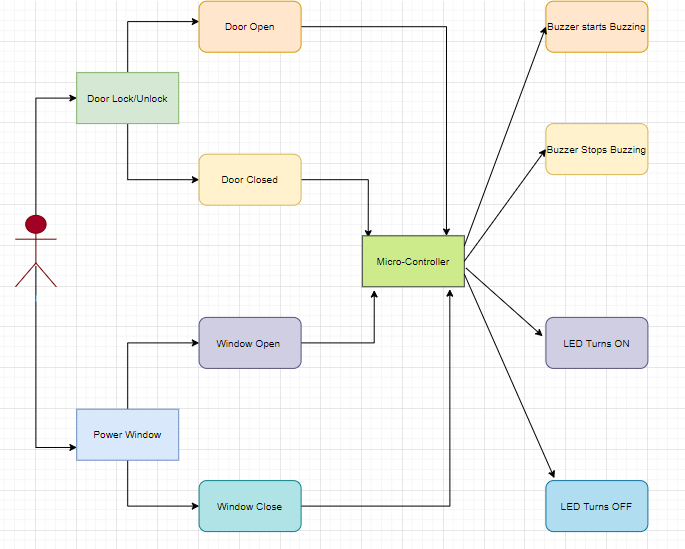
**j)** Jumper Wires

## Design

* High Level Requirements:

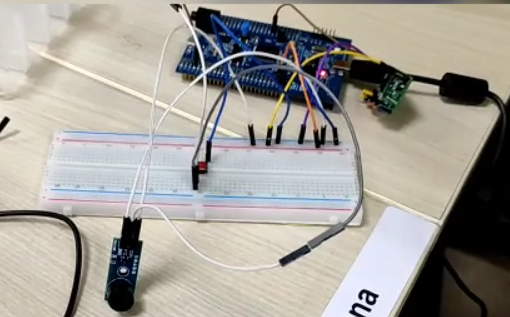


* Low Level Requirements:

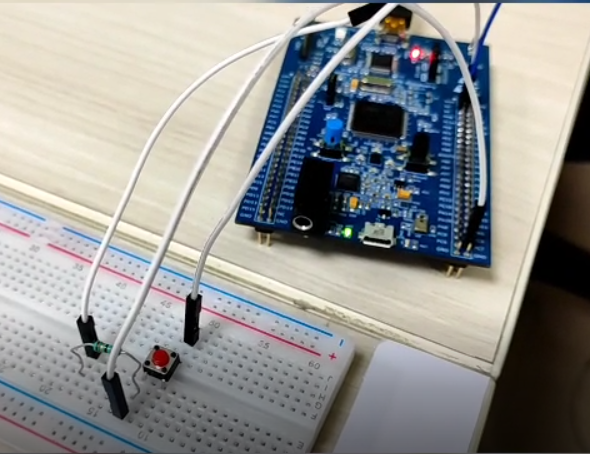


## Circuit Implementation:

* Door Lock/Unlock



* Power Window



## Test Plan

#### High-Level Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Number** | **Description** | **Expected Input** | **Expected Output** | **Actual Out** | **Type of Test** |
| 1 | Door lock/unlock | Door(s) is/are open/close | Buzzer in the car should start/stops buzzing | Buzzer connected to the discovery board start/stops buzzing. | Requirement based |
| 2 | Power window | Window(s) is/are open/close | Led in the car starts/stops glowing. | Green LED on the discovery board starts/stops glowing | Requirement based |

Table 7. High-Level Testing

#### Low-Level Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Number** | **Description** | **Expected Input** | **Expected Output** | **Actual Out** | **Type of Test** |
| 1 | Door lock/unlock | Door is open | Buzzer in the car should start buzzing | GPIO Pin PD 14 is reset and the buzzer connected to the discovery board start buzzing | Requirement based |
| 2 | Door lock/unlock | Door is close | Buzzer in the car is silent. | GPIO Pin PD 14 is set and the buzzer connected to the discovery board is silent | Requirement based |
| 3 | Power window | Window is open | Led in the car starts glowing. | GPIO PIN PD 13 is set and the green LED on the discovery board glows. | Requirement based |
| 4 | Power window | Window is close | Led in the car is off. | GPIO PIN PD 13 is reset and the green LED on the discovery board stops glowing. | Requirement based |

Table 8. Low -Level Testing

## Implementation Summary

Multiple features of the car using STM32f407 discovery board has been implemented. Here we have assigned certain pins for GPIO input and output by board configuration in STM32CubeIDE. Then we have generated a high-level abstraction code and then we use certain GPIO HAL functions, ADC HAL functions and configured.

We have also used the same LED’s for different feature indications.

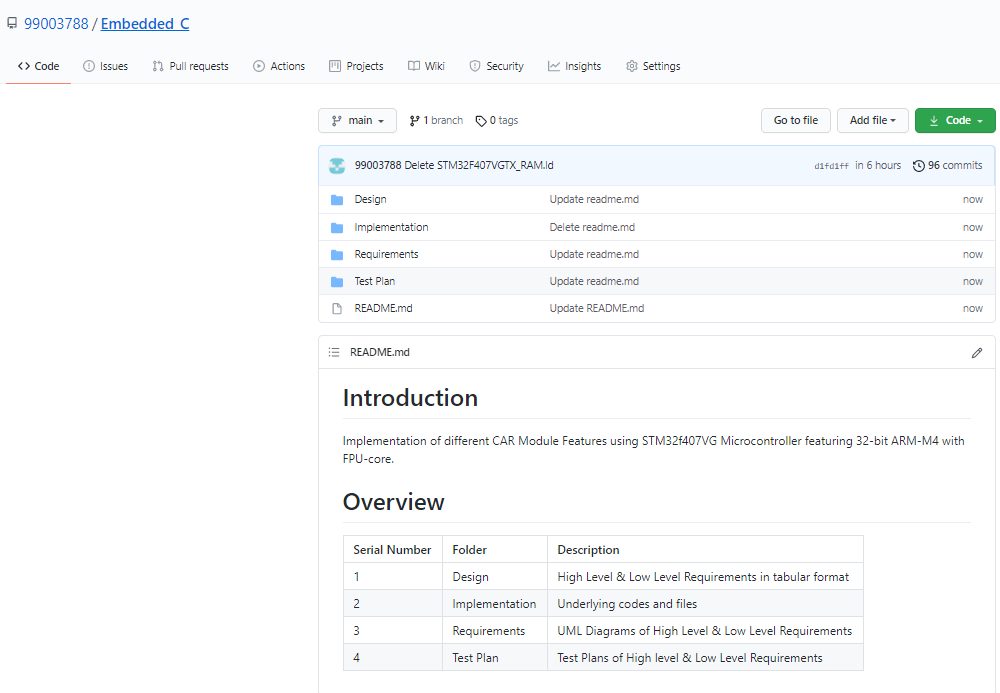
Features that I have implemented and pin configurations are as follows: -

* Power Window:
* PD13: Denotes the window(s) status (SET (1): Open, RESET (0): Closed)
* Door Lock/Unlock:
* PD14: Denotes the window(s) status (SET (1): Close, RESET (0): Open)

### Git Link

https://github.com/99003788/Embedded\_C.git

### Git Dashboard



### Summary

In this project, the features that has been selected are namely:

* Power window
* Sunroof Control
* Interior Lighting
* Door Lock/Unlock
* Seat Control, and
* Wiper Control

Out of these six, the features that I have implemented are:

* Power Window, and
* Door lock/Unlock

For power window control, I have used a switch along with the discovery board. When the switch is pressed GPIO PIN PD13 is set and the green LED on the discovery board turns on indicating that the car window(s) is/are open. Similarly, when the switch is released the GPIO PIN PD 13 is reset and the green LED on the discovery board turns off indicating that the window(s) is/are close.

For door lock/unlock feature, I have used the discovery board along with a buzzer and a switch. When the switch is pressed, the GPIO PIN PD 14 is reset and the buzzer starts buzzing indicating that the door(s) is/are open. Similarly, when the switch is released, the GPIO PIN PD 14 is set and the buzzer stops buzzing indicating the door(s) is/are closed.

### Challenges faced and how were they overcome:

* At first, we were facing issues with the STN32 Discovery Board because of which we needed to change the board.
* There were even few issues with our code even which were eliminated after going through the entire code properly and debugging.

# Miniproject -3 [Individual]

## Module

Python Programming

### Topic and Subtopics

#### Core-Topics

* Basic Python

##### Sub-Topics

* Data Types
* Arithmetic operations
* String operations
* Control structures

##### Sub-Topics

* If-else statements
* While loops
* For loops
* Nested Loops
* Functions

##### Sub-Topics

* Defining custom functions
* Pass by value
* Pass by reference
* Introduction to Library functions
* Data Structures

##### Sub-Topics

* List
* Tuple
* Set
* Dictionary
* Exceptional Handling

##### Sub-Topics

* Try, Except
* Finally
* pass keyword
* Date and time Library

##### Sub-Topics

* Date comparison
* Execution time analysis
* Excel file library (openpyxl)

##### Sub-Topics

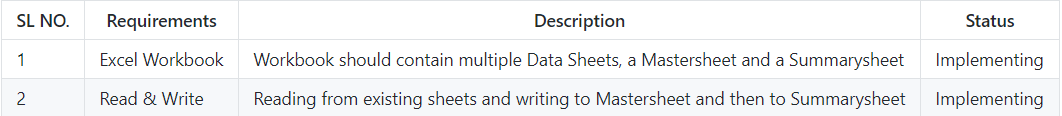
* Open multiple excel files
* Reading multiple excel files sheet by sheet.
* Mastersheet creation
* Writing to excel file

## Objectives

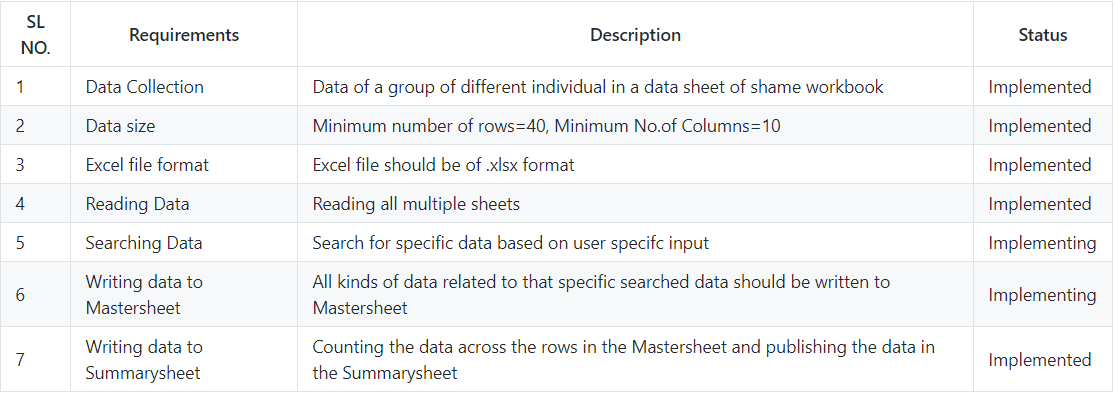
Analyzing data from 5 different excel spreadsheets and write specific data from all the excel sheets on a mastersheet and then counting number of entries in the mastersheet and write it in a summarysheet using Python Programming.

## Requirements

* High Level Requirements:

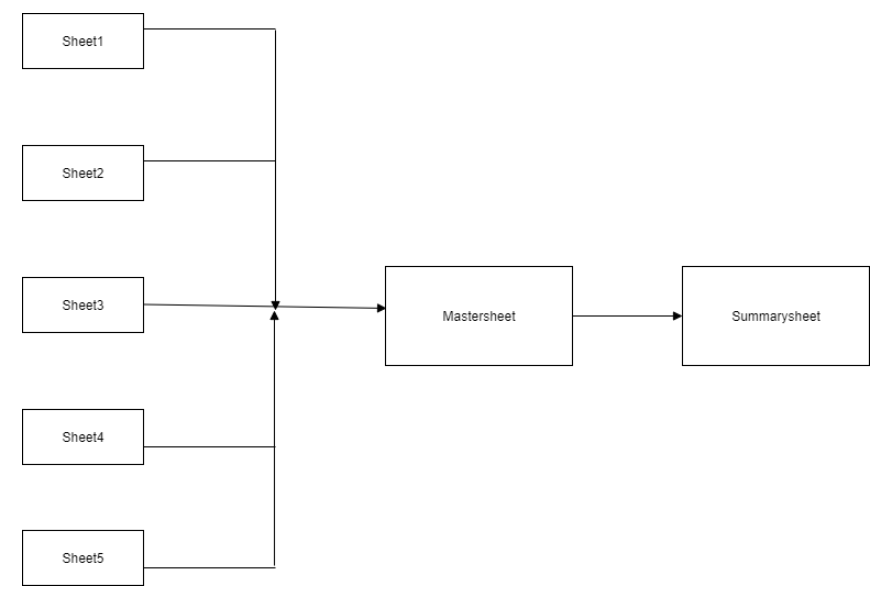


* Low Level Requirements:

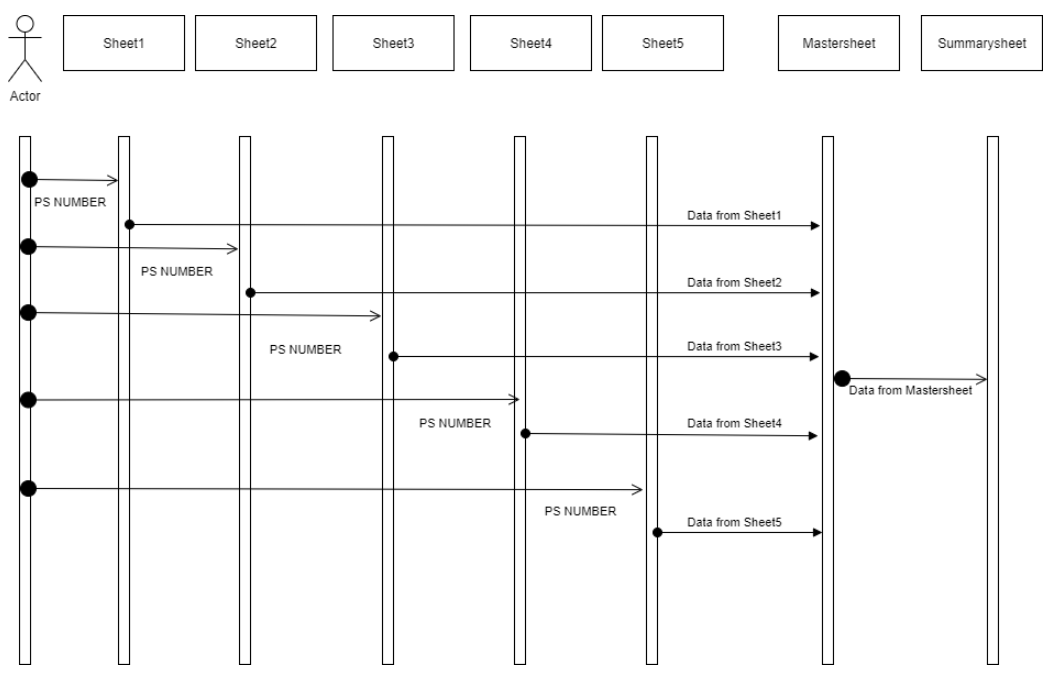


## Design

* High Level UML Diagram



* Low Level UML Diagram



## Test Plan:

#### High level Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl No.** | **Description** | **Expected Input** | **Expected Output** | **Actual Out** | **Type of Test** |
| 1 | To Search by PS NUMBER | Number of inputs you want: 2  Enter PS NUMBER: 62173517  Enter PS NUMBER: 44994897 | All the Data of 62173517 and 44994897 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | All the data of 62173517 and 44994897 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | Requirement based |

Table 11.High-Level Testing

#### Low Level Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl No.** | **Description** | **Expected Input** | **Expected Output** | **Actual Out** | **Type of Test** |
| 1 | To Search by PS NUMBER | |  | | --- | | Number of inputs you want: 3  Enter PS NUMBER: 62173517  Enter PS NUMBER: 44994897  Enter PS NUMBER: 20718405 | | All the Data of 62173517, 44994897 and 20718405 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | All the Data of 62173517, 44994897 and 20718405 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | Requirement based |
| 2 | To Search by PS NUMBER | Number of inputs you want: 2  Enter PS NUMBER: 62173517  Enter PS NUMBER: 44994897 | All the Data of 62173517 and 44994897 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | All the Data of 62173517 and 44994897 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | Scenario based |
| 3 | To Search by PS NUMBER | Number of inputs you want: 1  Enter PS NUMBER: 62173517 | All the Data of 62173517 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | All the Data of 62173517 along with all the headers are copied to master-workbook. Also, number of rows and columns in mastersheet are displayed in summarysheet. | Requirement based |
| 4 | To Search by PS NUMBER | Number of inputs you want: 0 | Only the headers from all the sheets will be copied to the mastersheet and total number of rows and columns will be displayed on the summarysheet | Only the headers from all the sheets will be copied to the mastersheet and total number of rows and columns will be displayed on the summarysheet | Requirement based |
| 5 | To Search by PS NUMBER | Number of inputs you want: 1  Enter PS NUMBER: 99003788  (the entered PS NUMBER is not present in any of the sheets) | Only the headers from all the sheets will be copied to the mastersheet and total number of rows and columns will be displayed on the summarysheet | Only the headers from all the sheets will be copied to the mastersheet and total number of rows and columns will be displayed on the summarysheet |  |

Table 12.Low-Level Testing

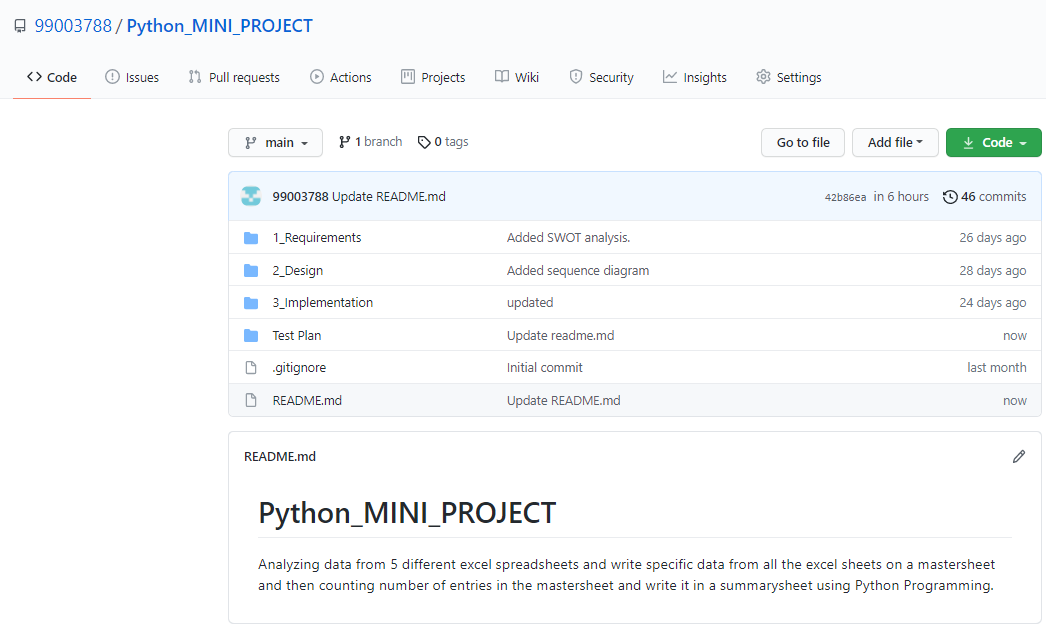
## Implementation Summary

Implementation folder has the python file with extension “.py” named **project.py**containing the code. It also contains the excel file which consists of 7 sheets namely “Sheet1”, “Sheet2”, “Sheet3”, “Sheet4”, “Sheet5”, “mastersheet” and “summarysheet”. “Sheet1” to “Sheet5” are populated manually. After we run compile and run the **project.py** file, data from “Sheet1” to “Sheet5” gets displayed on the mastersheet. The number of rows and columns on the mastersheet are displayed on the summarysheet.

### Git Link

<https://github.com/99003788/Python_MINI_PROJECT.git>

### Git Dashboard



**Git Summary**

There are four folders namely:

* 1\_Requirements:

Here there are two folders:

* HLR - It consists of the high-level requirement table.
* LLR - It consists of the low-level requirement table

There is also a readme.md file consisting of:

* Introduction
* 4W & 1H
* SWOT Analysis
* 2\_Design

Here there are two folders:

* HLR - It consists of the high-level requirement UML diagram.
* LLR – It consists of the low-level requirement UML diagram.
* 3\_Implementation

Here there are two files:

* project.py file consisting the code.
* Mini\_project.xlsx file consisting of the raw data.
* Test Plan

Here there is a readme.md file consisting of:

* High Level Test Plan
* Low Level Test Plan

### Summary

As per the objective, firstly I created an excel file that has five sheets. Then I started populating the sheets with data like NAME, PS NUMBER, Email, D.O.B, etc. The PS NUMBER column is common throughout all the five sheets.

Now coming to the code, here I have used openpyxl library. I have also followed Object Oriented Programming approach and used two classes. In the first class named ***accept\_value*** I have created a method ***take\_vlaue ()*** that gives the user a choice to select number of input and then asking him/her to enter the input(s). In the next class named ***main*** that inherits the ***accept\_value*** class***,*** am creating a method ***my\_func ().*** In this method, firstly am loading the excel file. Then making use of the PS NUMBER(S) entered by the user am searching all the data related to that PS NUMBER(S) and the dumping the entire date collected from the 5 sheets into a existing sheet named ***mastersheet***. After that I am counting the number of rows and columns in the mastersheet and then reflecting the result into a sheet named summarysheet.

### Challenges faced and how were they overcome

* At first, I used panda library but I faced difficulty in understanding proceeding with the code. This issue was resolved by using openpyxl library instead.

### Future Scope

* Searching in huge amount of data needed to be analyzed.
* Code can be more optimized and automatic generation of summarysheet or masterbook can be added.
* This code as an API or GUI can be used in analysis of data in schools, colleges, MNCs, banks, etc. for maintaining records.

# Miniproject -4 [Individual]

## Module

“Kernel and Device Drivers”

### Topic and Subtopics

#### Core-Topics

* System Calls

##### Sub-Topics

* C programming
* Custom Kernel Image Generation
* Cross-Compilation (using Qemu)
* IPC (Inter- Process Communication)

##### Sub-Topics

* C programming
* Process
* Race Condition
* Semaphores
* Mutex
* Spinlock
* Wait Queue
* Threads
* Custom Modules

##### Sub-Topics

* Kfifo API
* List API
* Module development
* IOCTL

## Objectives & Requirements

#### Objective

* System calls -- echo back the given string
* System calls -- Traverse process list, print pid and ppid
* System calls -- Retrieve attributes of calling process

#### Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **SL No.** | **Requirements** | **Description** | **Status** |
| 1 | System call to echo string | The system call echo backs the given string. | Implemented |
| 2 | System call to traverse process list and print pid | System call traverses through process list and print pid of all the running process. | Implemented |
| 3 | System call to traverse process list and get attributes like state, priority and pid of the calling process | System call traverses through process list and retrieves state, priority and pid of calling process. | Implemented |

## Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL No.** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| 1 | Given string echo’s back | “Hello World!” passed as a command line argument | “Hello World!” (on serial console (Qemu)) | “Hello World!” | Requirement based |
| 2 | PID of all the running process is to be printed | Invoking executable file of user-space code on serial console | PID of all the running process. (on terminal) | PID of all the running process. | Requirement based |
| 3 | Attributes such as state, priority, PID of calling process is to be printed | Invoking executable file of user-space code on serial console | PID, State, Priority of the calling process. (on terminal) | PID, State, Priority of the calling process. | Requirement based |

Table 14.Test-Plan (Kernel Driver Development)

## Implementation Summary

For System Calls: -

* Generate new zImage after adding system calls definition and prototype. Adding its definition file name in Makefile.

-> make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- zImage

* Then mount the SD card and copy the output file of user space code in it and then unmount it by following commands: -

-> sudo mount –o loop, rw, sync rootfs.img /mnt/rootfs

-> sudo cp a.out rootfs.img /mnt/rootfs/home/root

->sudo umount rootfs.img /mnt/rootfs

* Then run Qemu using following command

-> qemu-system-arm -M vexpress-a9 -m 1024 -serial stdio \

-kernel zImage -dtb vexpress-v2p-ca9.dtb \

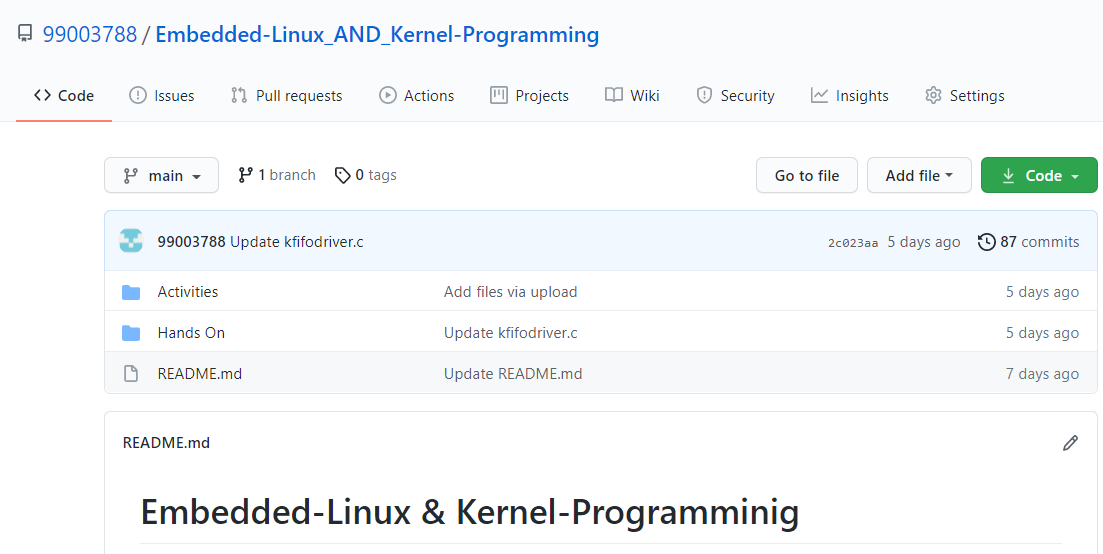
-sd rootfs.img -append "console=ttyAMA0 root=/dev/mmcblk0 rw"

* Then run the ./a.out file on Qemu and output is display either on VGA console or in serial console based on the system call. (give command line input only for first system call)
* System call displays the required output.

### Git Link

https://github.com/99003788/Embedded-Linux\_AND\_Kernel-Programming.git

### Git Dashboard



### 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.

### Challenges faced and how were they overcome

* Unable to directly access string in kernel space from userspace 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.