./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1. |  | Abhishek Guria/  99003726 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69567481)

[APPLIED SDLC CALCULATOR 5](#_Toc69567482)

[Module:- 5](#_Toc69567483)

[INTRODUCTION 5](#_Toc69567484)

[MY PRODUCT: “SEMI-ENGINEERING CALCULATOR” 5](#_Toc69567485)

[RESEARCH 5](#_Toc69567486)

[REQUIREMENTS: 6](#_Toc69567487)

[LOW LEVEL REQUIREMENTS: 8](#_Toc69567488)

[DESIGN 10](#_Toc69567489)

[HIGH LEVEL DESIGN 10](#_Toc69567490)

[LOW LEVEL DESIGN 10](#_Toc69567491)

[INBUILT CONSTANTS AND THEIR OPERATIONS:- 12](#_Toc69567492)

[BEHAVIOURAL DIAGRAMS:- 13](#_Toc69567493)

[Implementation Summary 14](#_Toc69567494)

[Git Link 14](#_Toc69567495)

[Git Dashboard 14](#_Toc69567496)

[Unit Testing 14](#_Toc69567497)

[Test plan for power of given function:- 14](#_Toc69567498)

[Test plan for Trigonometric functions:- 16](#_Toc69567499)

[Individual Contribution & Highlights:- 17](#_Toc69567500)

[Challenges Faced And How Were They Overcome:- 19](#_Toc69567501)

[EMBEDDED C : HARDWARE + PROGRAMMING + TESTING 20](#_Toc69567502)

[MODULES:- 20](#_Toc69567503)

[Topics and Subtopics 20](#_Toc69567504)

[INTRODUCTION TO THE SOFTWARE : STM32CUBEIDE 20](#_Toc69567505)

[Objectives & Aim:- 22](#_Toc69567506)

[Requirements:- 22](#_Toc69567507)

[Design 23](#_Toc69567508)

[Activity 1 – COMPILATION APROACH 23](#_Toc69567509)

[1.2- STARTUP CODE 26](#_Toc69567510)

[1.3- LINKER SCRIPT 28](#_Toc69567511)

[1.4- DEBUGGING TECHNIQUES 29](#_Toc69567512)

[Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE 31](#_Toc69567513)

[2.1 GPIO: 31](#_Toc69567514)

[2.2 EXTI: 33](#_Toc69567515)

[Test Plan 34](#_Toc69567516)

[2.3 ADC 35](#_Toc69567517)

[2.4 SPI 37](#_Toc69567518)

[2.5 UART 40](#_Toc69567519)

[Implementation summary 42](#_Toc69567520)

[Advanced Python Programming Summary Report 47](#_Toc69567521)

[Modules:- 47](#_Toc69567522)

[Objective :- 47](#_Toc69567523)

[Requirements :- 47](#_Toc69567524)

[**Defining the idea for project :** 47](#_Toc69567525)

[Detailed Requirements:- 48](#_Toc69567526)

[DESIGN:- 49](#_Toc69567527)

[Structral Diagram 49](#_Toc69567528)

[Behavioral Diagram 50](#_Toc69567529)

[SOURCE CODE:- 51](#_Toc69567530)

[Code Quality:- 54](#_Toc69567531)

[GITHUB LINK:- 54](#_Toc69567532)

[Summary 54](#_Toc69567533)

[Learning Report – Embedded Linux & Kernel Device Drivers Programming 55](#_Toc69567534)

# MINIPROJECT 1:- APPLIED SDLC CALCULATOR

# Module:-

Module used in this project is visual studio code.

**SYSTEM/ SOFTWARE DEVELOPMENT**

## INTRODUCTION

We have attempted to build the software development cycle of a semi-engineering calculator. It includes the numerous high-level and low-level requirements of the various features involved in semi-engineering such as log, antilog, exponents roots and polynomial functions. It then explores about the various test cases that should be taken in consideration such that all requirements are taken care of. In the end, we build the codes that follow all the requirements and design efficiently as per them.

## MY PRODUCT: “SEMI-ENGINEERING CALCULATOR”

Our system is more or less semi advanced which basically able to do desirable calculations of daily basis from bottom to top level i.e. it includes every basic operation which eats lots of time in scheduled timings.

## RESEARCH

There are different types of calculators with respect to the cost and its features. It is plotted in the graph shown below which shows the range of calculators for highest cost and lowest feature and vice versa. The range we have chosen is for a medium range cost and feature. It is similar to a programmable scientific calculator. It will be more efficient for the multipurpose use.

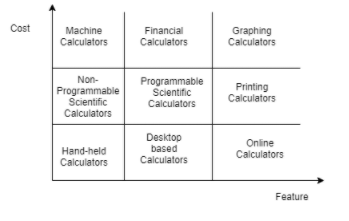
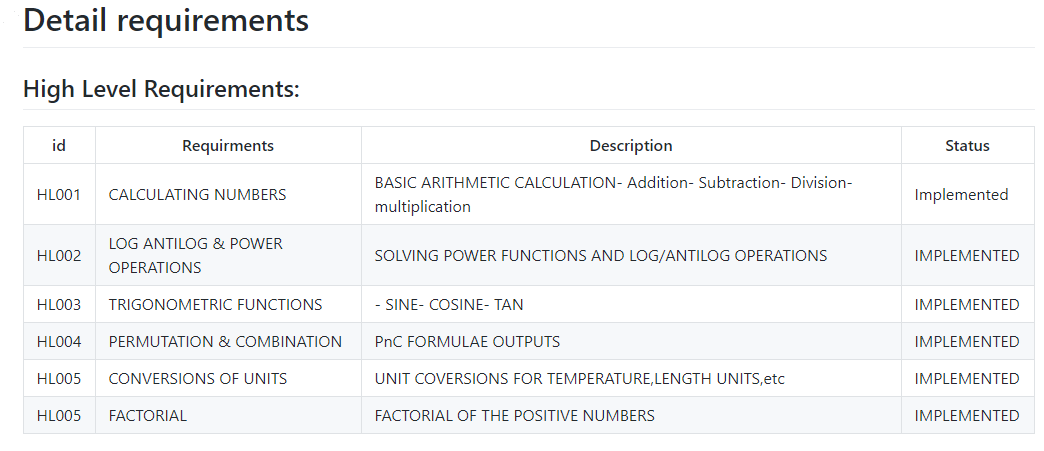


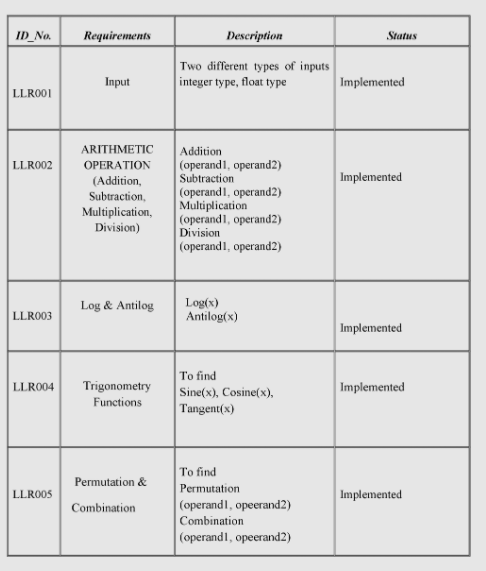
Figure1: Cost vs Features

## REQUIREMENTS:

The calculator build by us includes basic arithmetic functions, trigonometric functions and exponential functions. It also consists of some other functions including power, log, antilog, permutation and combination functions.



## LOW LEVEL REQUIREMENTS:



# ACTIVITY 2: AGILE METHODOLOGY

**Theme-Calculator**

The theme is designing a calculator with certain features according to the specific requirements.

The target customers for the calculator are school and college going students for semi-engineering and various other branches.

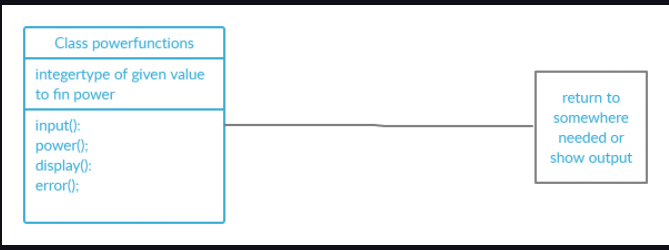
according to the requirements provided are:

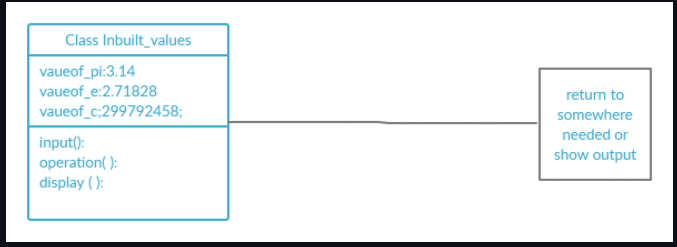
1. To find out power of particular value
2. Trigonometric Ratios.
3. Complex Functions(i.e log antilog, and constants).

**User Story**

# DESIGN

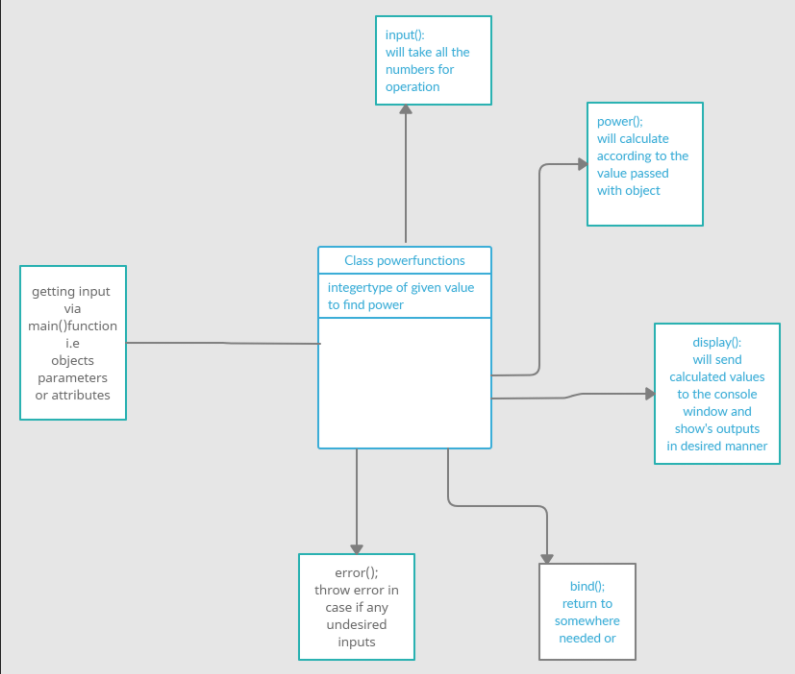
## HIGH LEVEL DESIGN



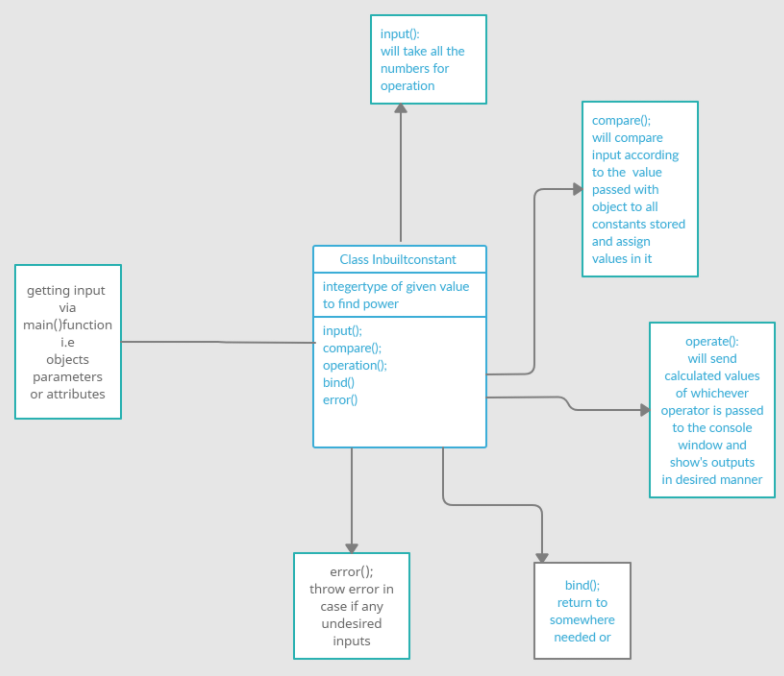
****

**Figure 1&2: High level Diagram**

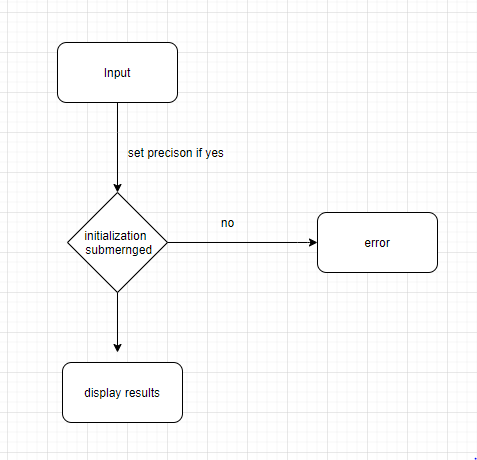
## LOW LEVEL DESIGN

****

# INBUILT CONSTANTS AND THEIR OPERATIONS:-

****

# BEHAVIOURAL DIAGRAMS:-



**Figure 3: For digit precision**



**Figure 4: Diagram for Constants**

## 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)

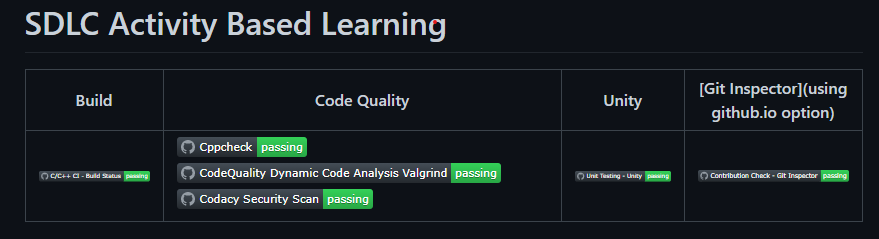
Ensure code quality and clean code and description practices

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

### Git Link

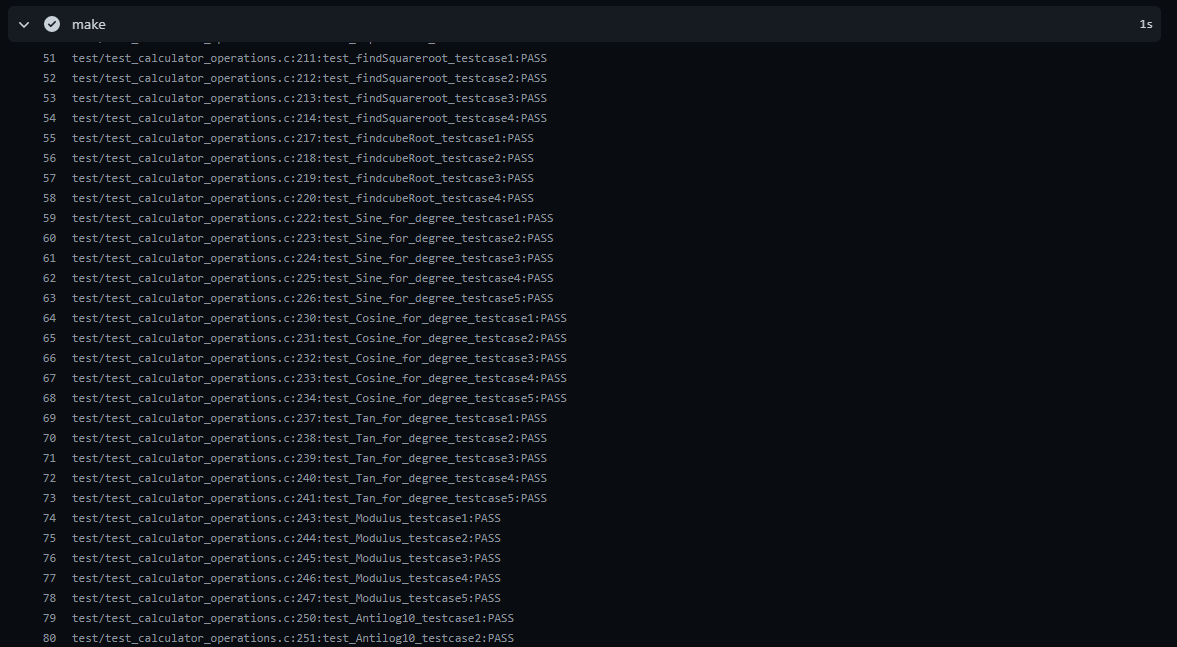
<https://github.com/99003721/AppliedSDLC_Calculator_N1>

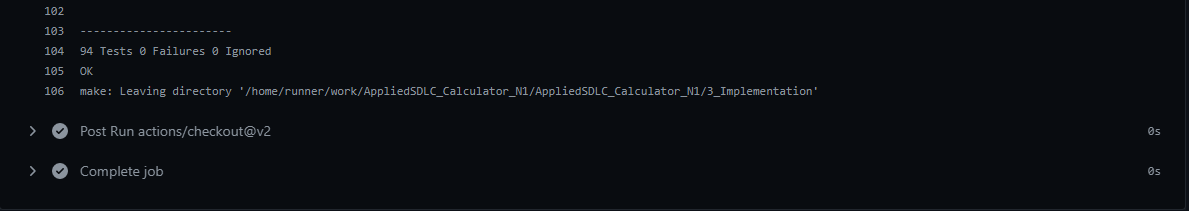
### Git Dashboard



****

# Unit Testing





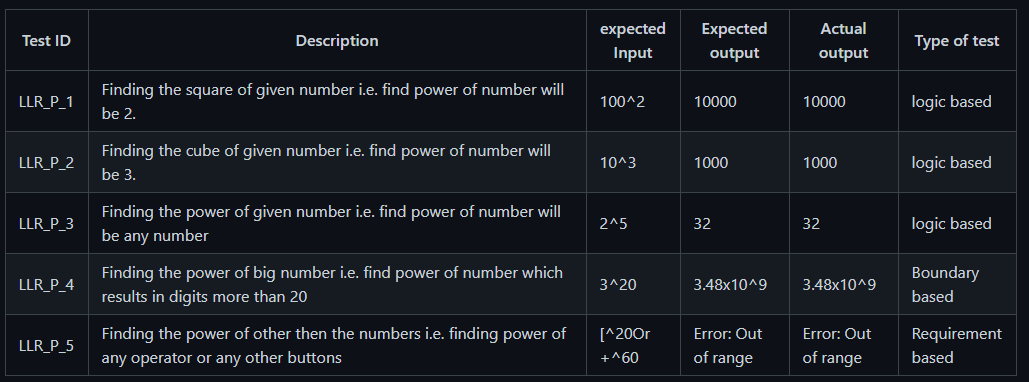
# Code quality:-



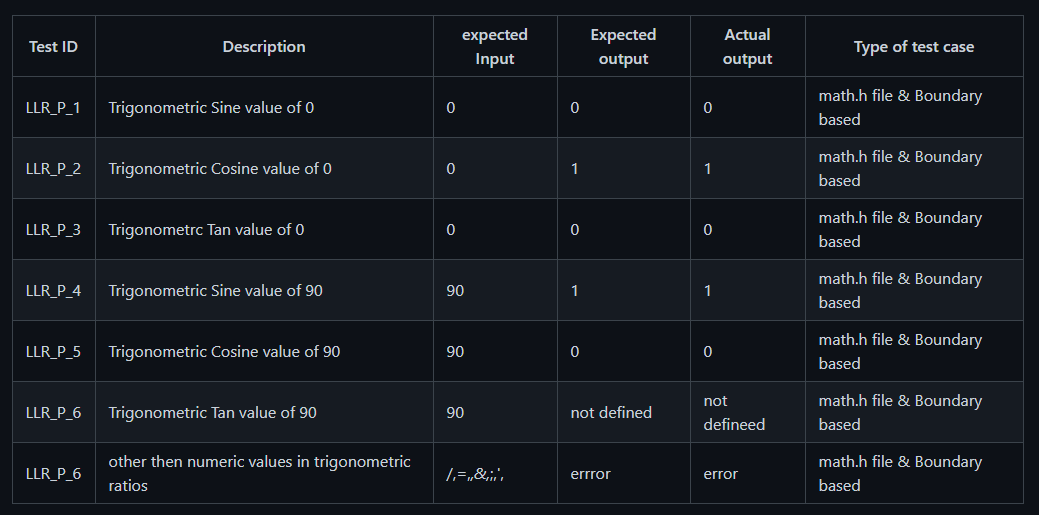




## Test plan for power of given function:-



## Test plan for Trigonometric functions:-



## Individual Contribution & Highlights:-

**Power functions:**

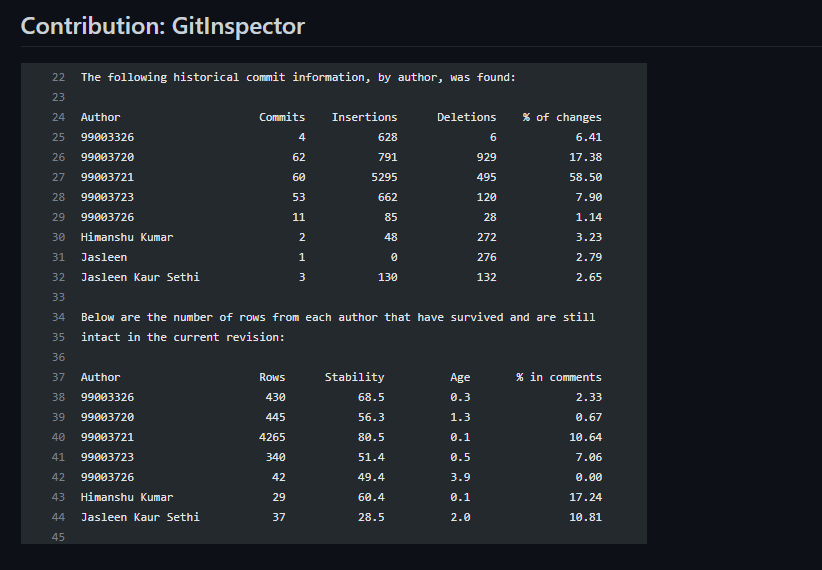
1. Find out the square & square root.
2. Find out the cube & cube root.
3. Answer in less than 8 digits
4. Less than 8-digit input

**Trigonometric Ratios:**

1. To find Sine
2. To find Cosine of given angle
3. To get tan value.

**Complex functions**:-

1. To find out the logarithmic values.
2. To find out the antilogarithmic values.
3. To use inbuilt constants and do operations on them.



# Challenges Faced And How Were They Overcome:-

* Running the make file as its resolved by defining its correct path(.out for linux and -lm for math functions)
* Syncronizing the VS code to github , colleague help to resolve the issue
* Making the function call in correct path
* Open git log while committing, thus went to github desktop and pulled origin and then pushed origin.
* Test case code for the boundary problem. Added code with the help of internet

### 

SUMMARY:-

* In this task the learning part or objective is to learn how to familiarize with the main tools to build projects like visual studio code and execute codes in it with the particular formats.
* as with different header modules as well as their definitions and finally end up completion of the project with the test cases to throughput the results of our project and to take out learning as how to build future projects in the same way as this project.

# EMBEDDED C : HARDWARE + PROGRAMMING + TESTING

# MODULES:-

The main usefull modules are listed below:-

* Use of STM32CubeIDE.
* Virtual studio code.

# Topics and Subtopics

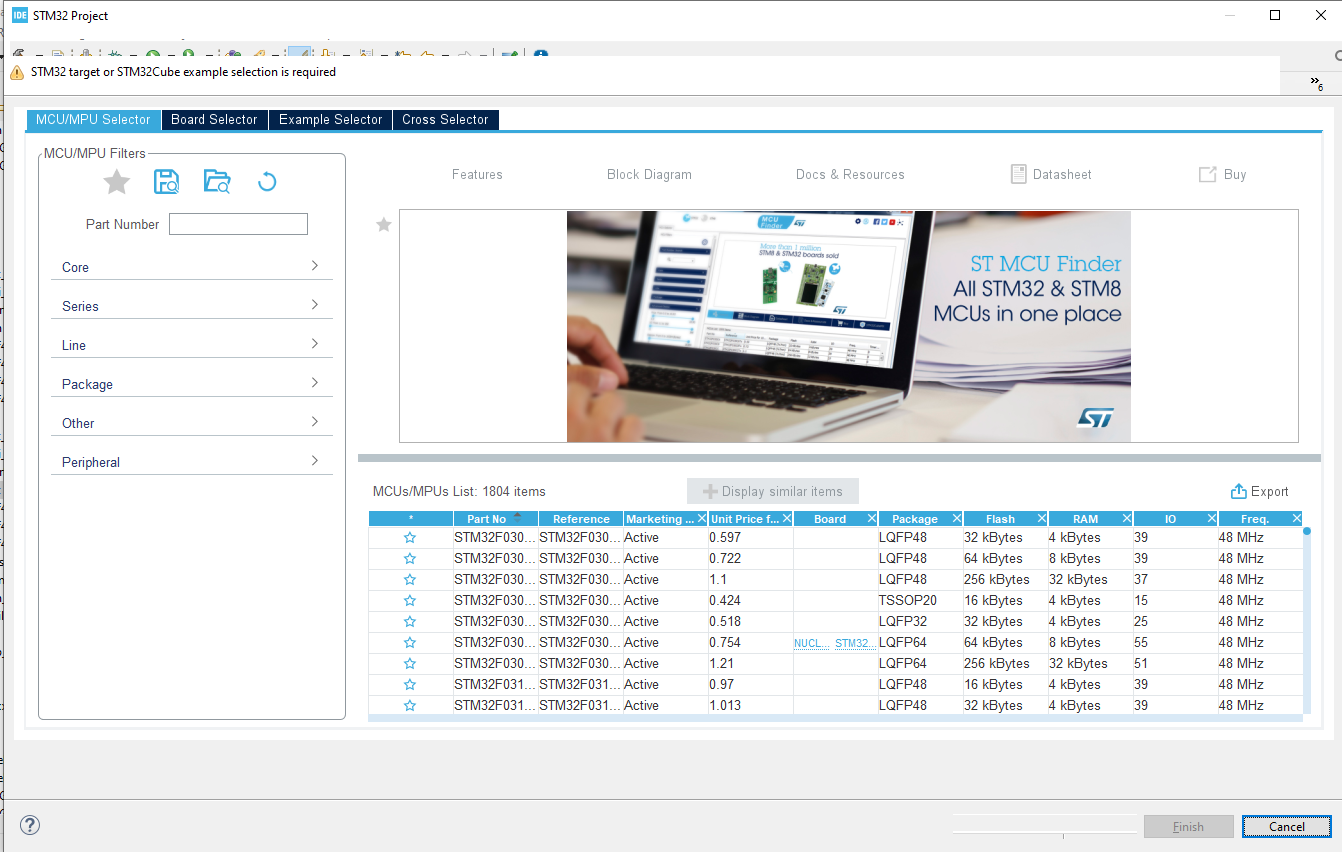
# INTRODUCTION TO THE SOFTWARE : STM32CUBEIDE

* STM32CubeIDE is a multi-OS development tool, which is part of the STM32Cube system program.
* STM32CubeIDE is a state-of-the-art C / C ++ development platform with boundary modification, code production, code integration, and debugging features for STM32 microcontrollers and microprocessors. Based on the Eclipse® / CDT framework and the GCC development tools, as well as the GDB debugging. Allows integration of hundreds of existing plugins that complete the Eclipse® IDE features.
* STM32CubeIDE combines STM32 configuration and project creation functionality from STM32CubeMX to provide the same tools experience and save installation time and development time. After the selection of the STM32 MCU or blank MPU, or a customized microcontroller or microprocessor from the board selection or model selection, the project is created and a code is generated. At any time during development, the user can revert to implementation and configuration or middleware and update the startup code without affecting the user code.
* STM32CubeIDE includes build and layout analysts that provide the user with useful information about project status and memory requirements.
* STM32CubeIDE also includes advanced and advanced debugging features including basic CPU registers, memory, and road-related registers, as well as a flexible live clock, Serial Wire Viewer interface, or error analyzer.

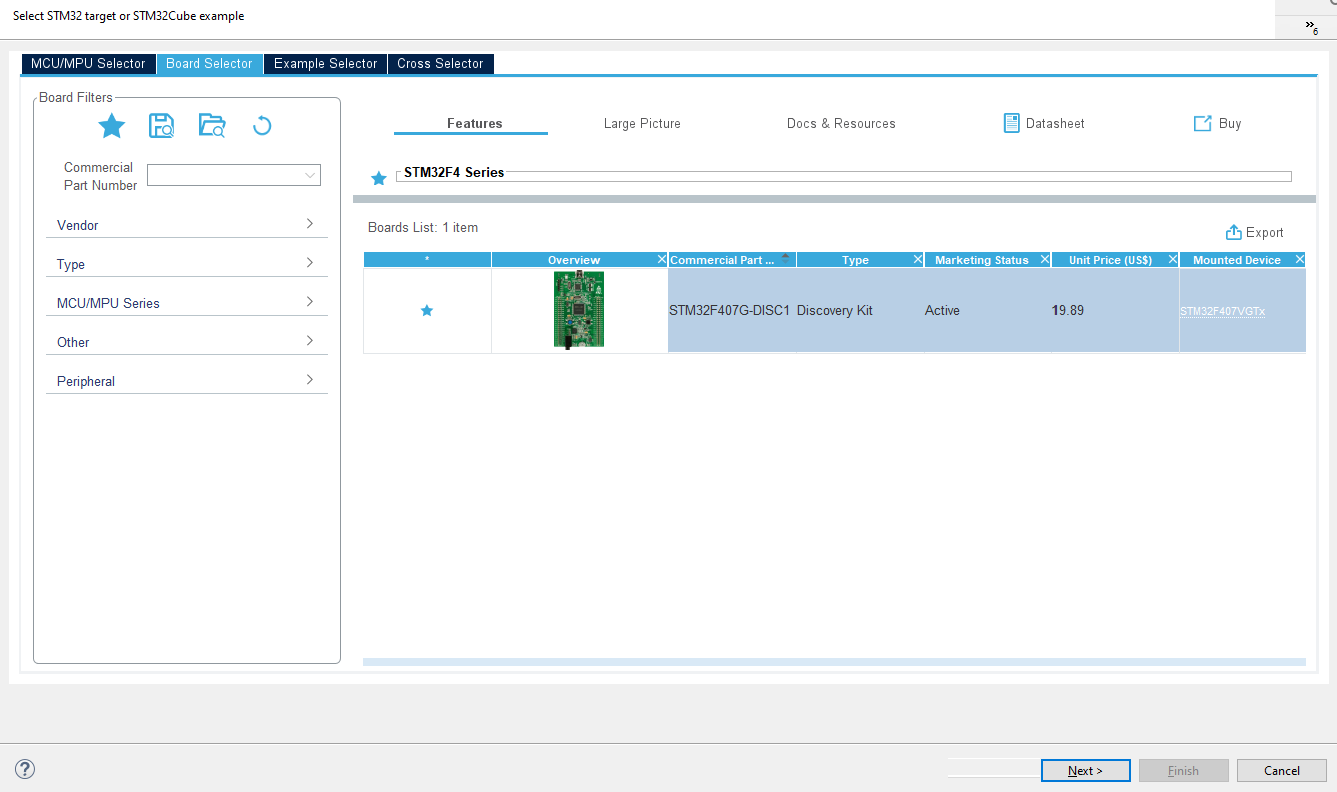
**GETTING STARTED WITH THE SOFTWARE:**

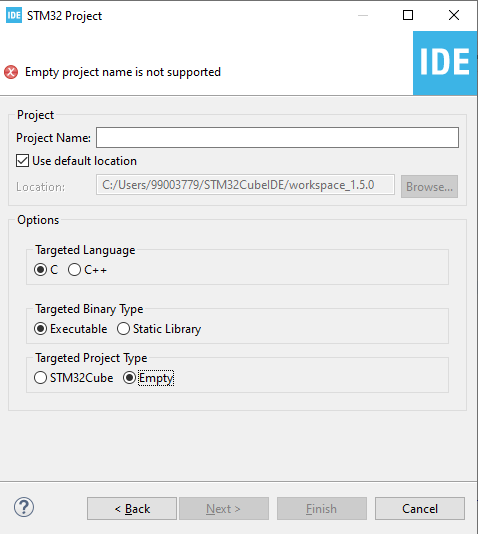
Start the software:

1. Goto file 🡪 New🡪 STM32 Project



1. Select the board and star it for frequent uses and start with the project .





Type the name of the project and start with the coding.

# Objectives & Aim:-

1. To build our own drivers according to our need and compile the same in different development environment tool kit to asses our code for driver.
2. To build and work upon the driver code and use it in the making of the body control module of the vehicles with using various sensors and actuators.

# Requirements:-

In hardware: -

1. STM32XXXX series board
2. Connecting cables
3. Bread board
4. Connecting wires
5. Sensors
6. Actuators

Software

1. Visual studio code
2. STM32CubeIDE

# Design

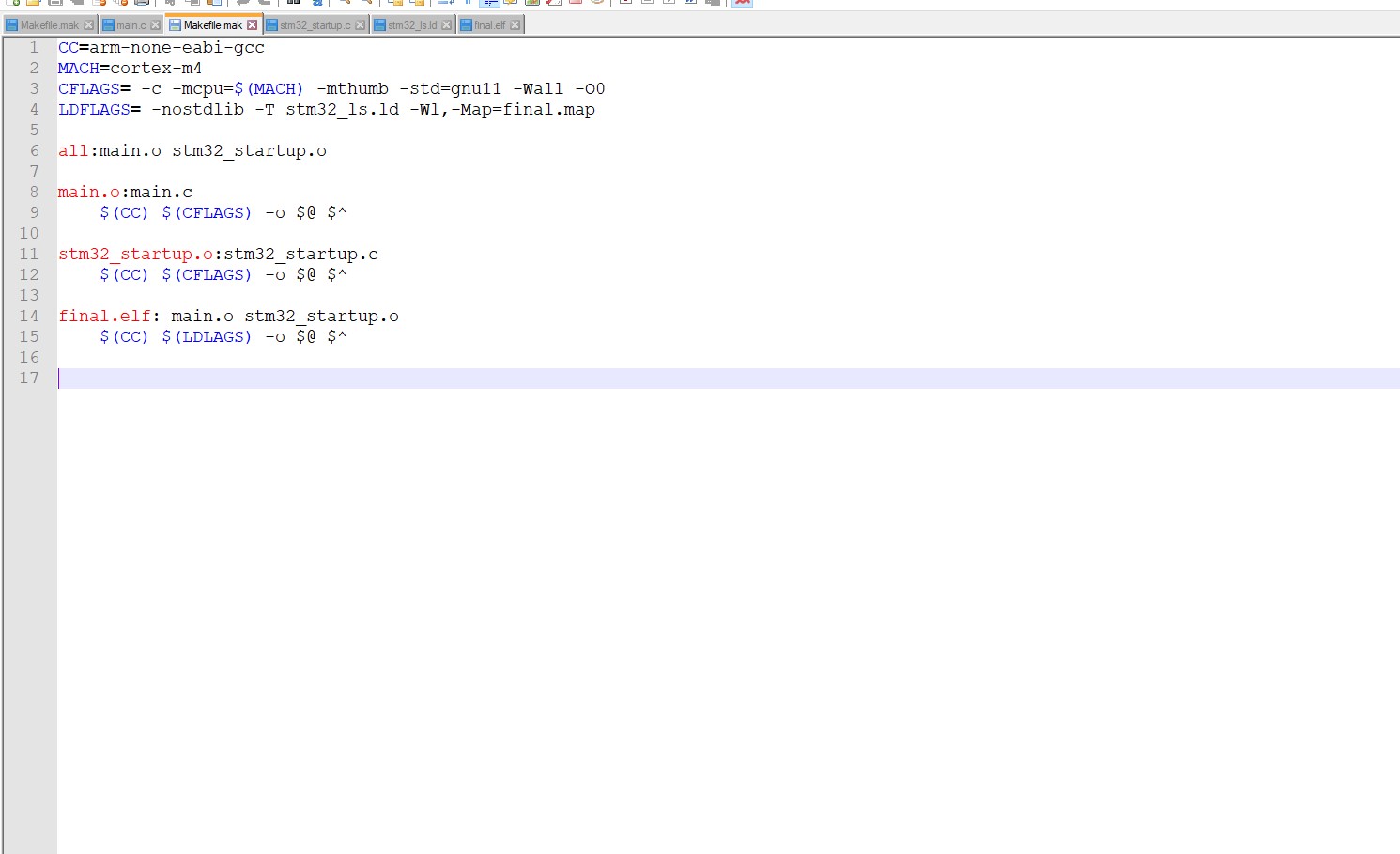
## Activity 1 – COMPILATION APROACH

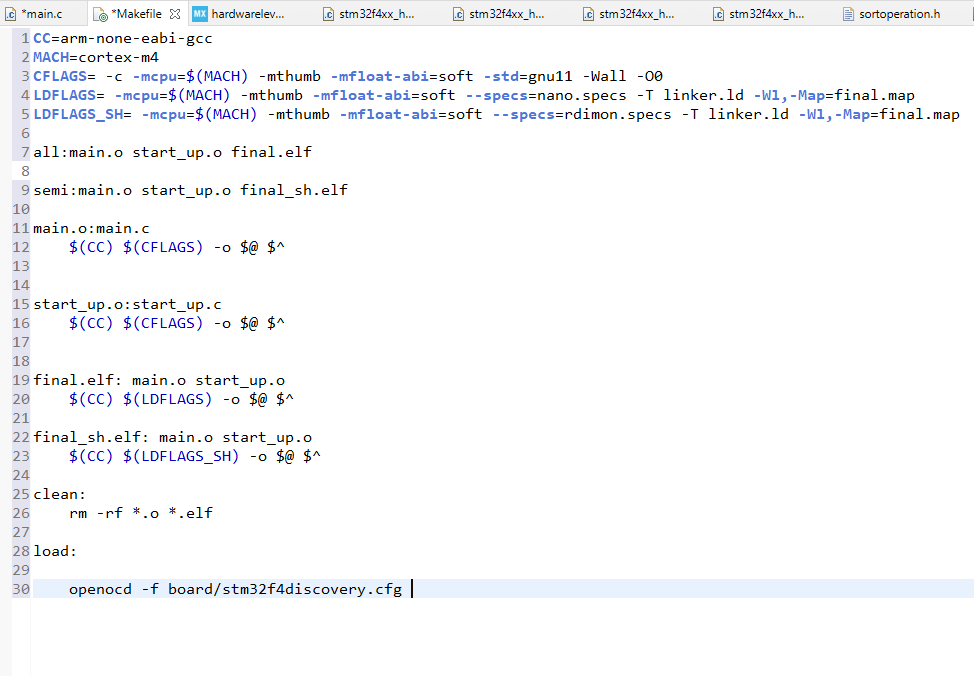
This is a complete process for integrating the ARM Cortex Mx processor sample system. The following are the integration stages of plan C:

1. Preprocessor stage
2. Compilation stage
3. Assembly stage
4. Linking stage

**1.1- MAKE FILE**

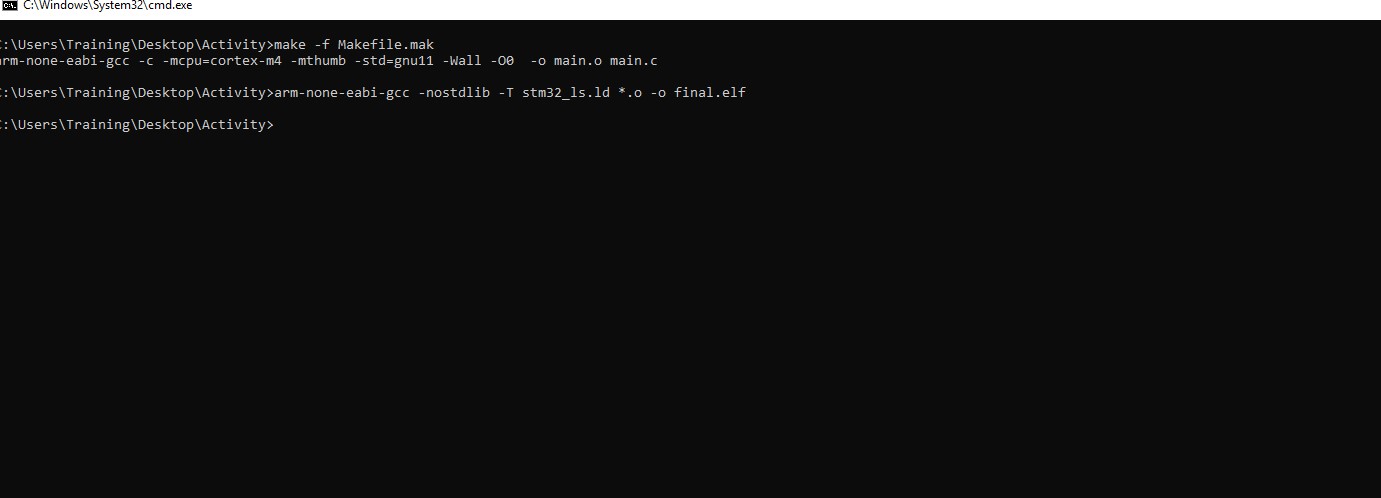
Below is the make file for the sample program:





**Fig 1. Make file**

The command to run this make file in the command prompt is:



**Fig 2 Make command**

* -mcpu = cortex-m4 used to select our cortex-m4 processor used
* -mthumb is used to generate output code in ARM state
* main.o is a redirect file
* main.c dependence

## 1.2- STARTUP CODE

• The start up file is responsible for setting the correct code usage fields in the main.c file.

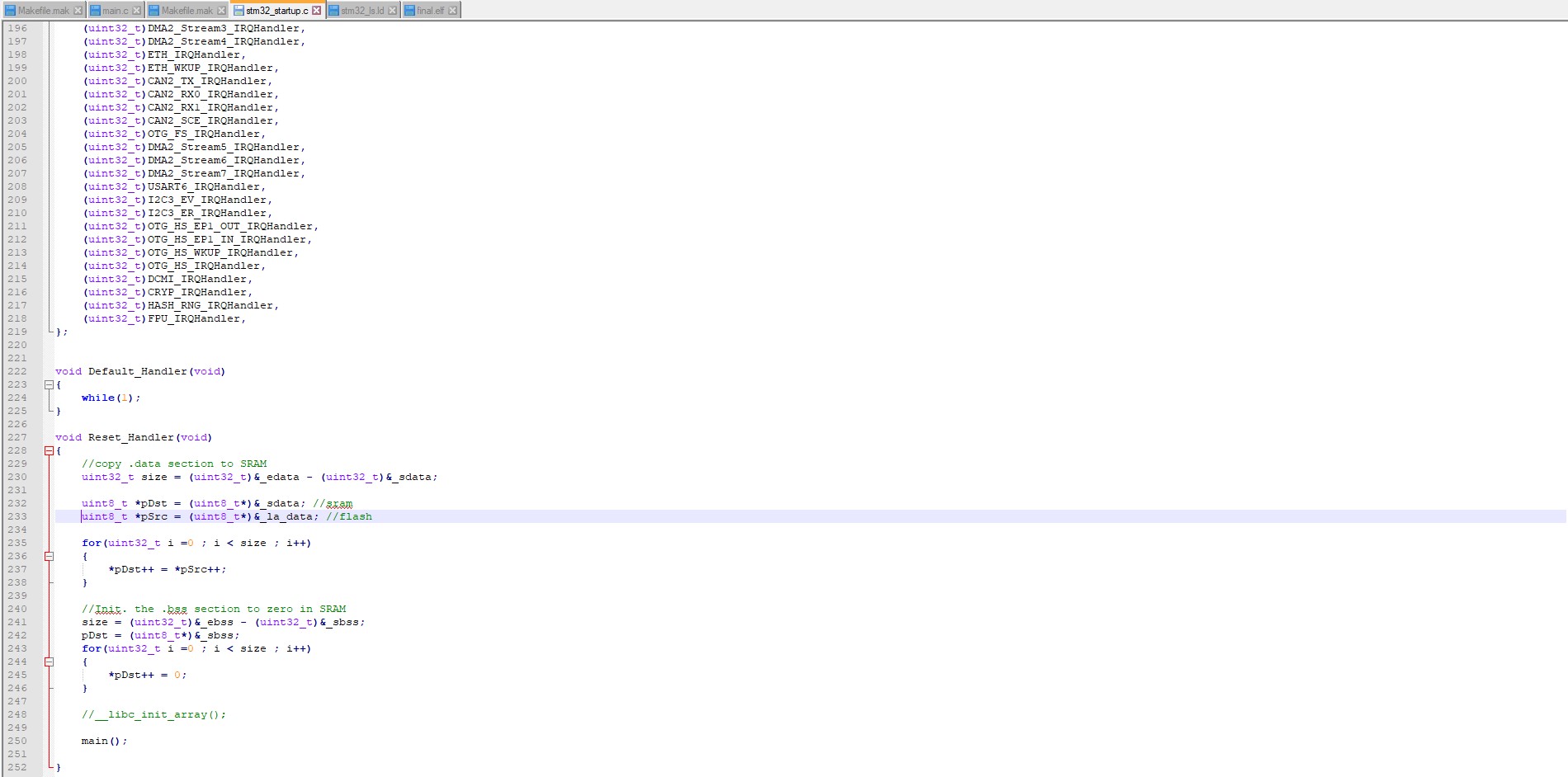
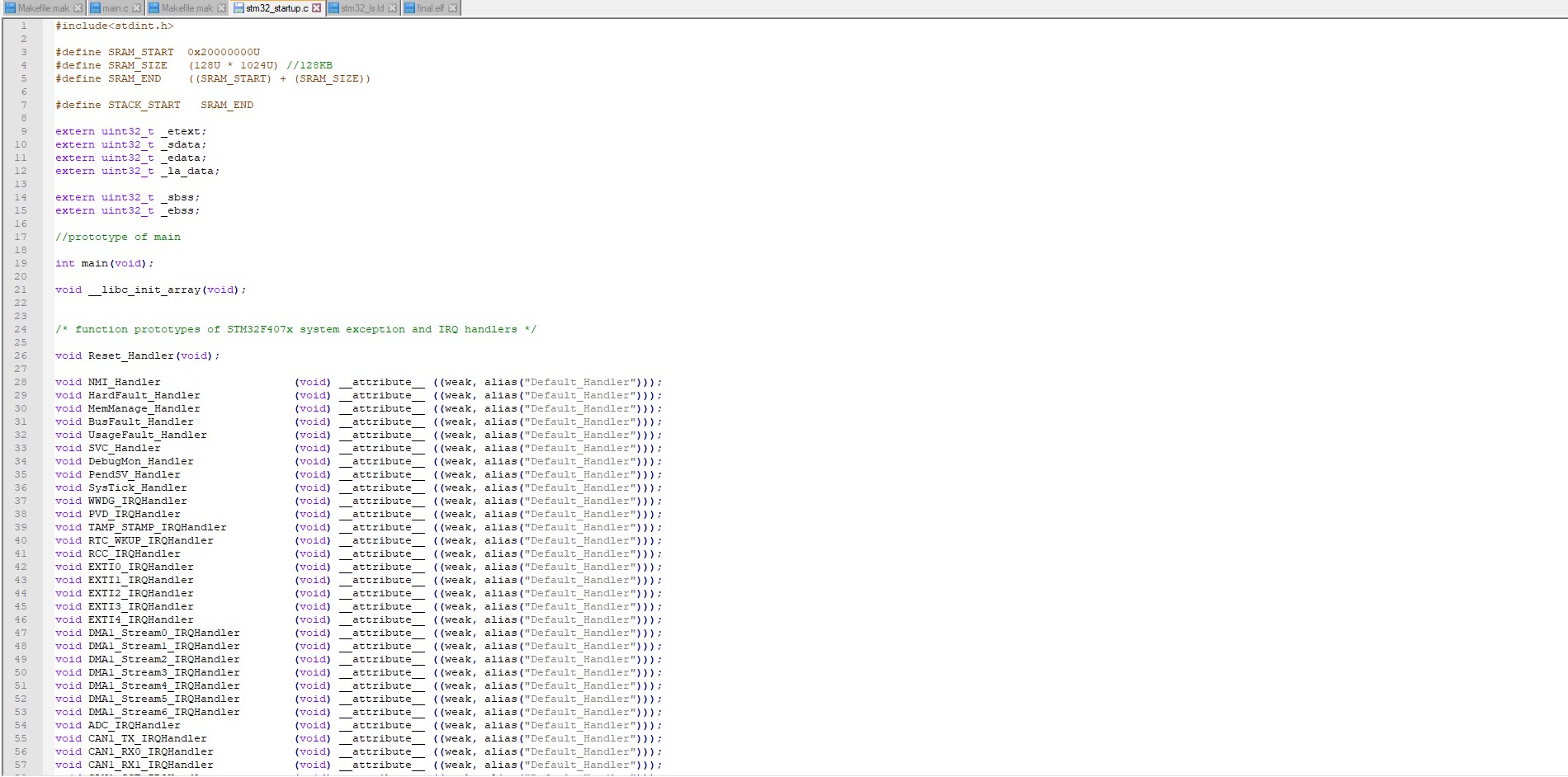
• Part of the start code depends on the direction (processed)

• Role of startup file:

1. Create a special MCU microcontroller vector table.

2. Write the startup code that starts the data section and .bss in SRAM.

3. Call the main ()



**Fig 3. Startup code**

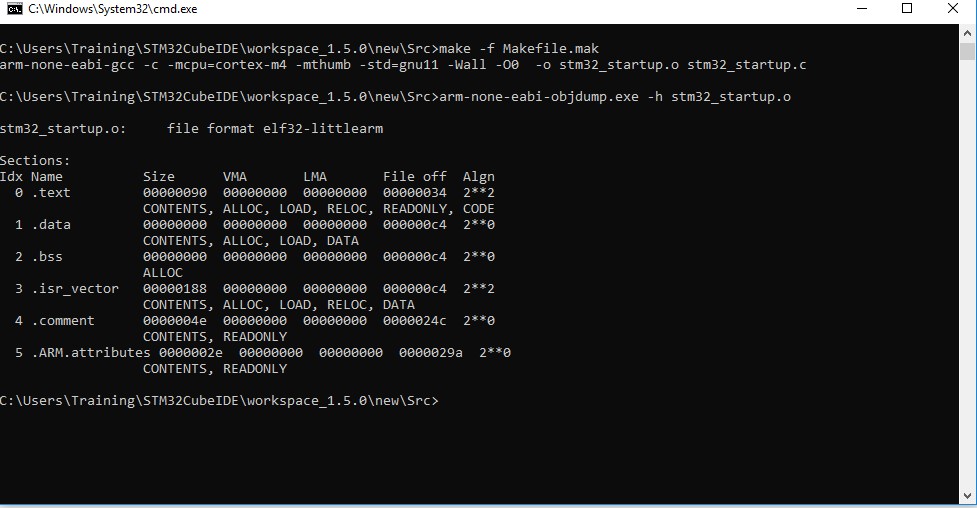
In the startup code we use the attributes to keep the variable in the user-defined function.

Job qualifications:

• Weakness: Allows the app to override a previously defined weak function (dummy function) with the same function name.

• Alias: Lets programmer provides any alias name for the same function.

A startup.o file made in usable format, the various components of which are shown below:



**Fig 4: Startup command**

## 1.3- LINKER SCRIPT

• Linkers take one or more object files or libraries as input and combine them to create a single usable file as output.

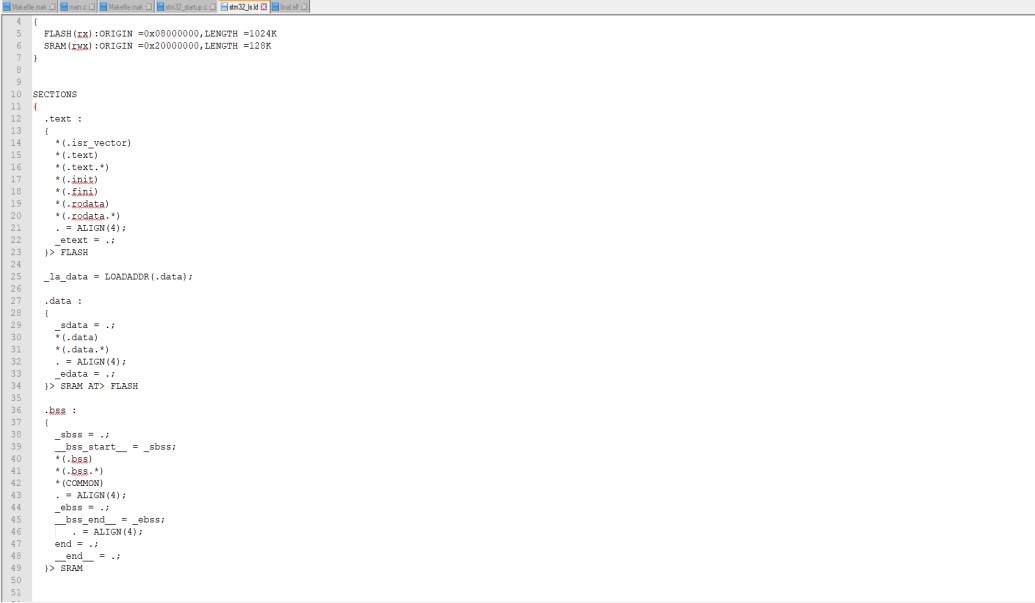
• Linker scripts determine how different categories of object files should be compiled to create an output file.

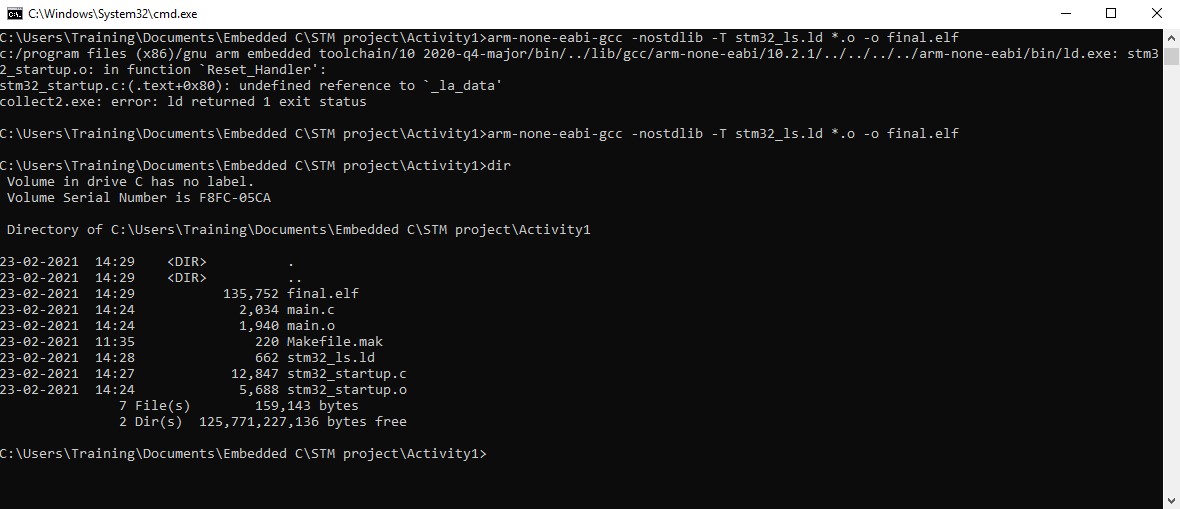
• Reset holder is the login of the application

• The login command is used to set the "login address address" information in the last elf file file created.

Syntax: Login (brand\_sign)

Login (Reset\_Manager)





**Fig 5: command to generate final.elf file**

## 1.4- DEBUGGING TECHNIQUES

• STM32F407VG embedded in chip debugger to fix code error.

• OCD ON-Chip Debugger aims to provide bug fixes, program settings and test parameters for embedded targeted devices.

• OCD is a free and resource-based application that allows you to edit, correct, and analyze your plans using GDB.

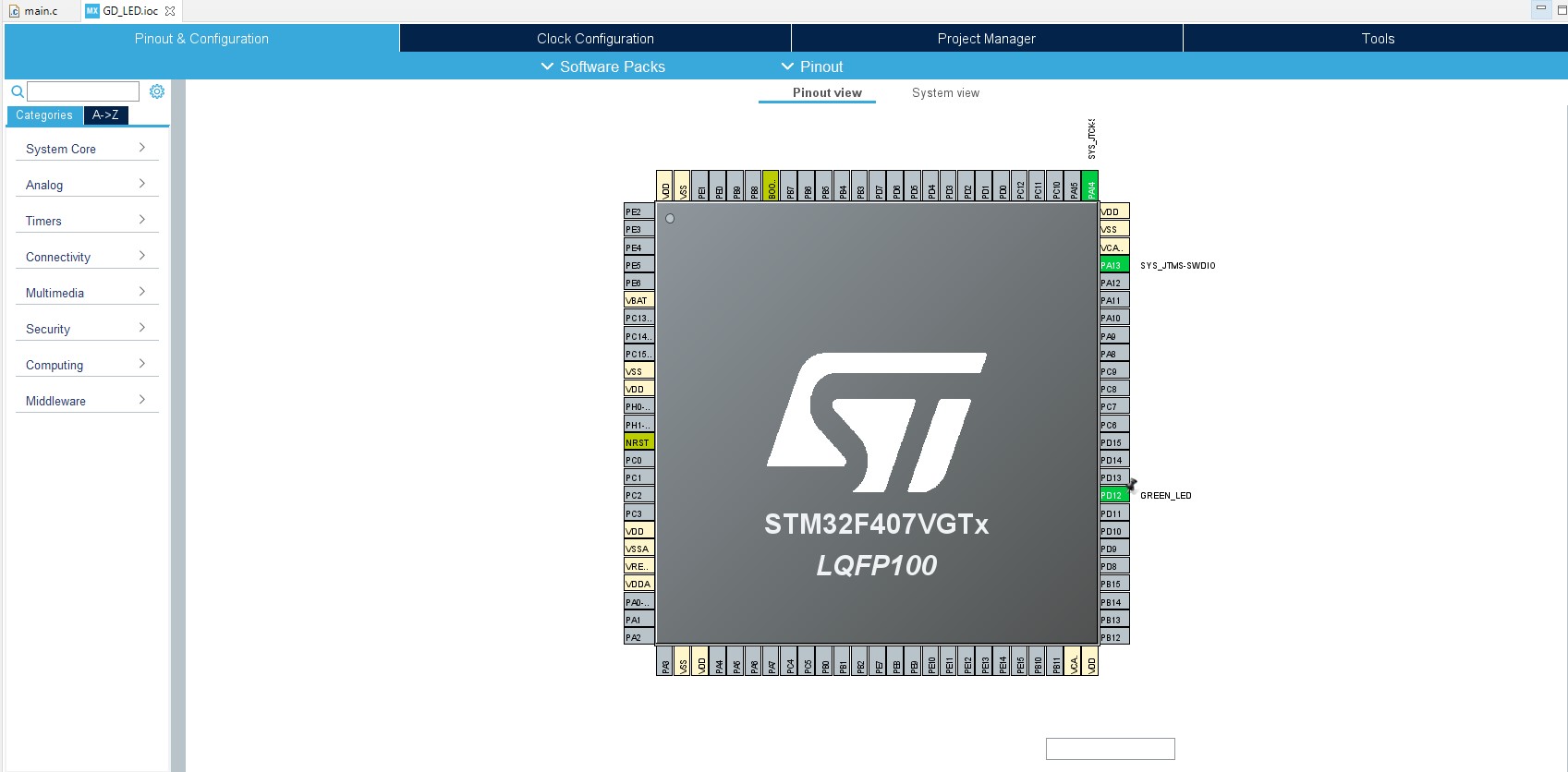
• Supports various target boards based on different processor properties.

# Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

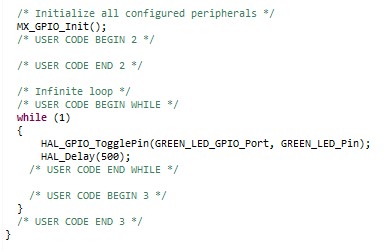
Protocol implementation for STM32F407VG microcontroller featuring ARM32 bit ARM – cortex M4 having FPU core using HAL Library

## 2.1 GPIO:

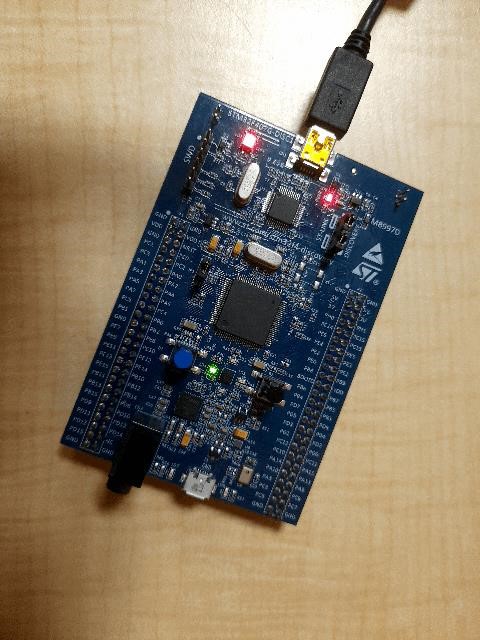
Toggling LED at pin PD12 at GREEN\_LED\_GPIO\_PORT. Serial wire is enabled at pin PA13.



**Fig: 6: GPIO pin configuration**



**Fig: 7 : GPIO configuration code**

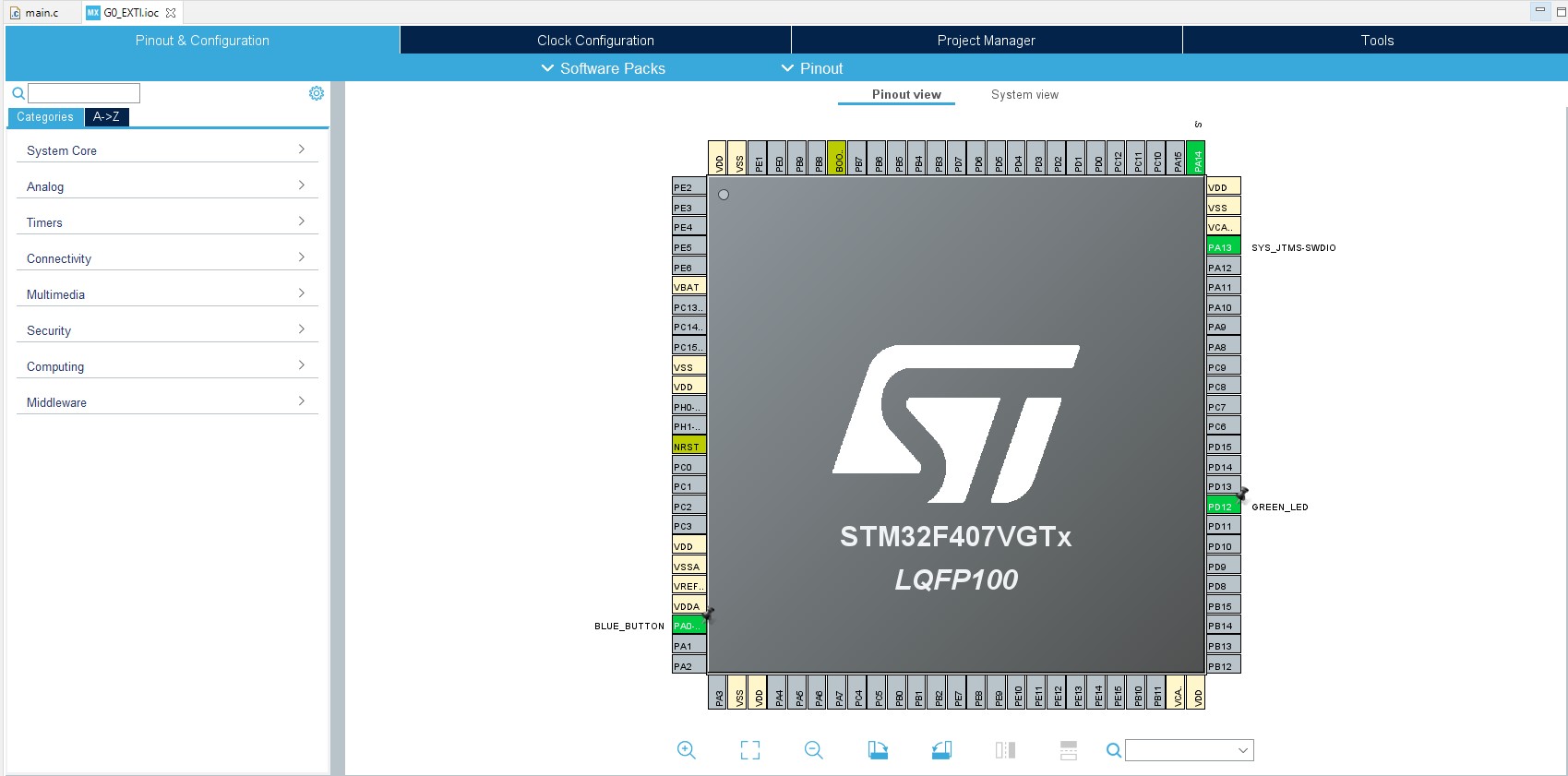


**Fig: 8: LED toggling**

## 2.2 EXTI:

The blue button on the PA0 acts as an external distraction.

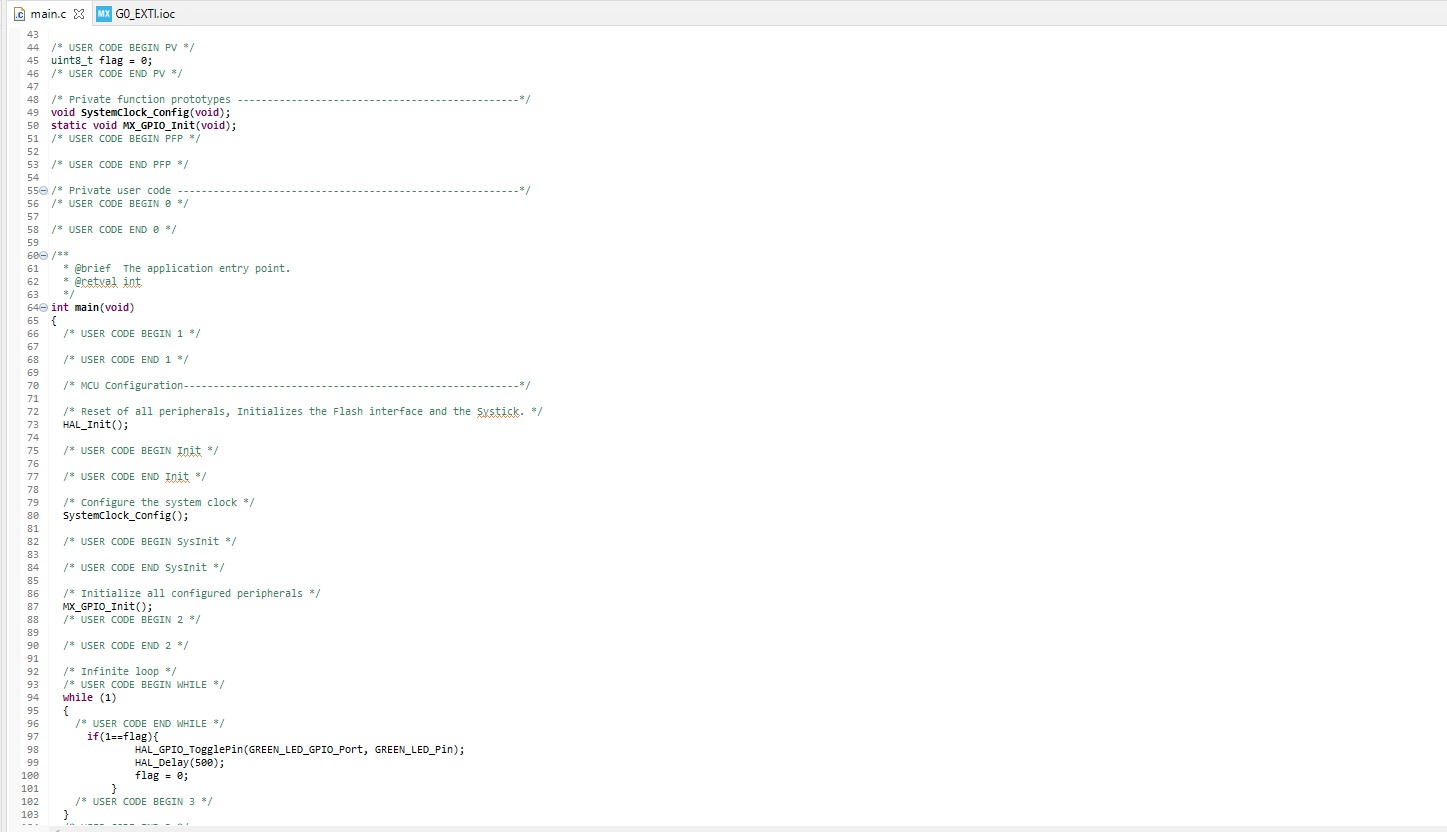
When the blue button is pressed Green LED on the PD12 pin.



**Fig 9: EXTI pin configuration**

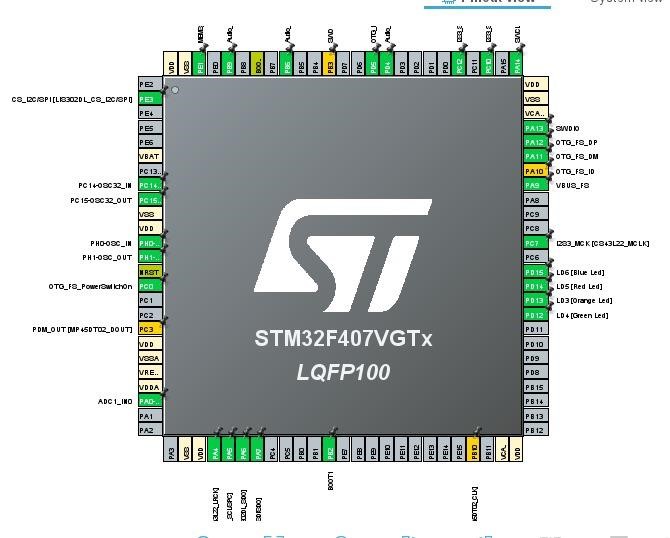
# Test Plan

In the main.c file the flag is started and if the flag == 1, the status below the if loop is made to convert the LED to PD12.



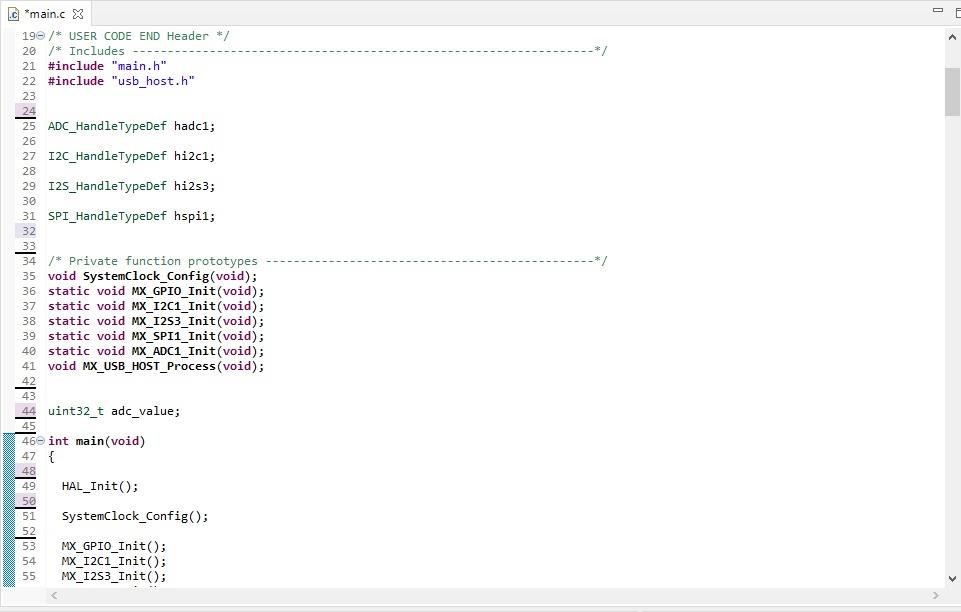
**Fig. 10: EXTI configuration code**

## 2.3 ADC

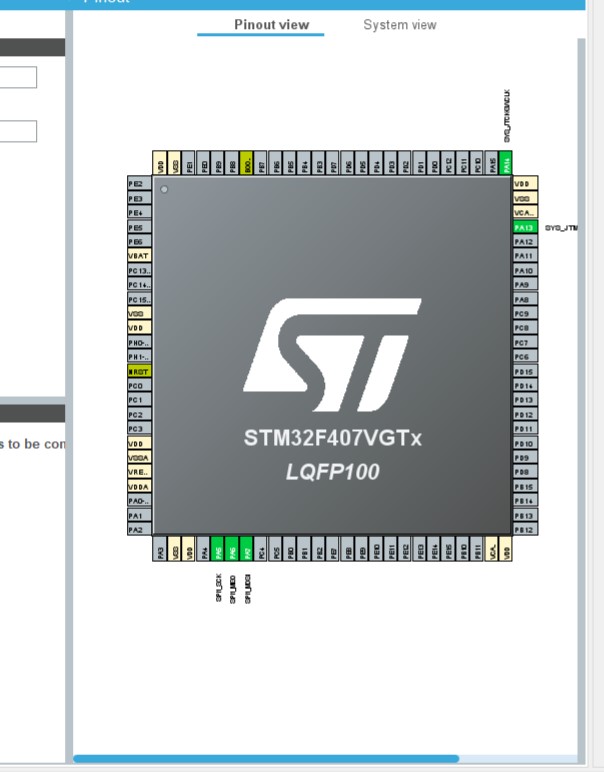


**Fig. 11: ADC pin configuration**

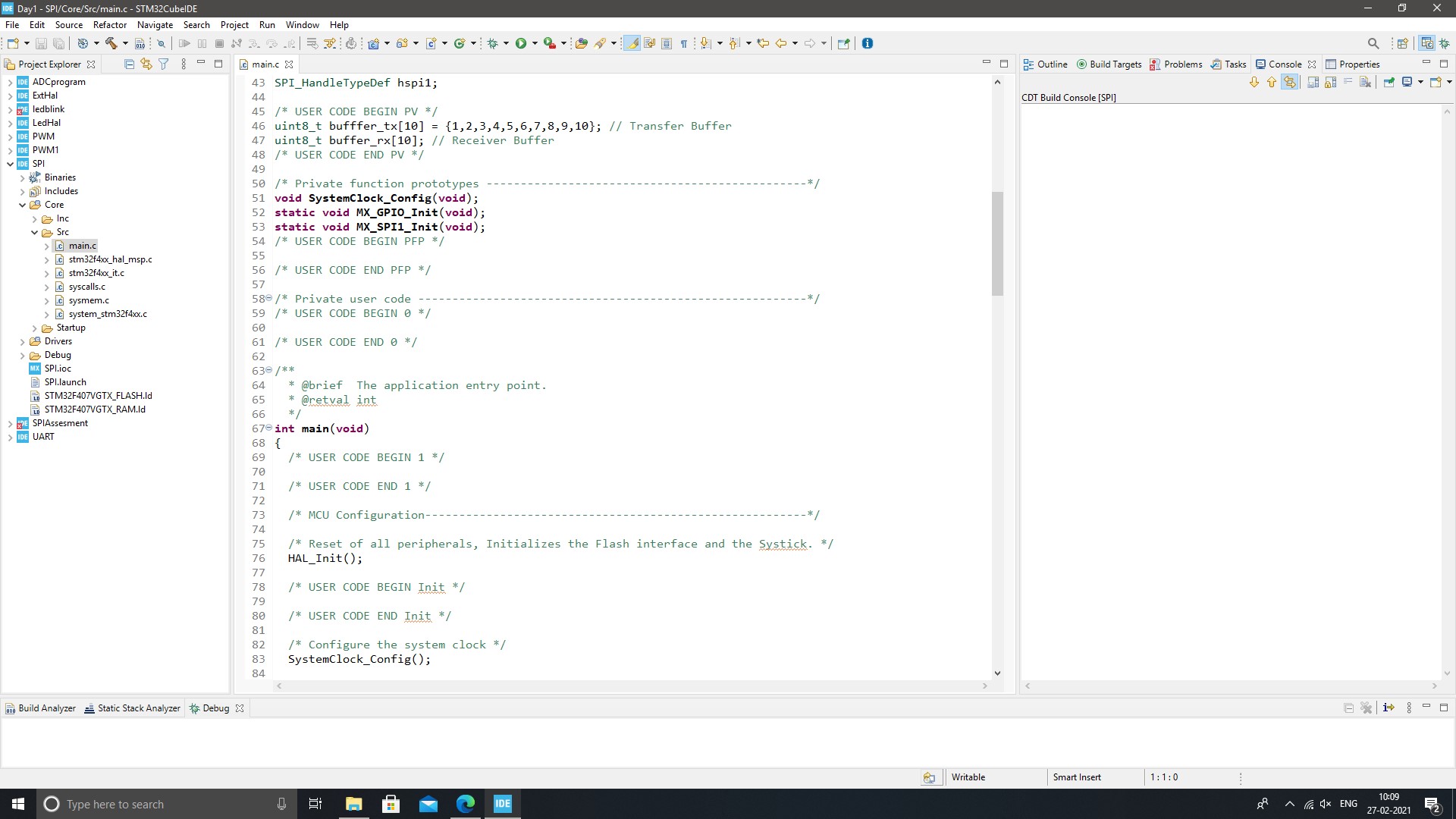
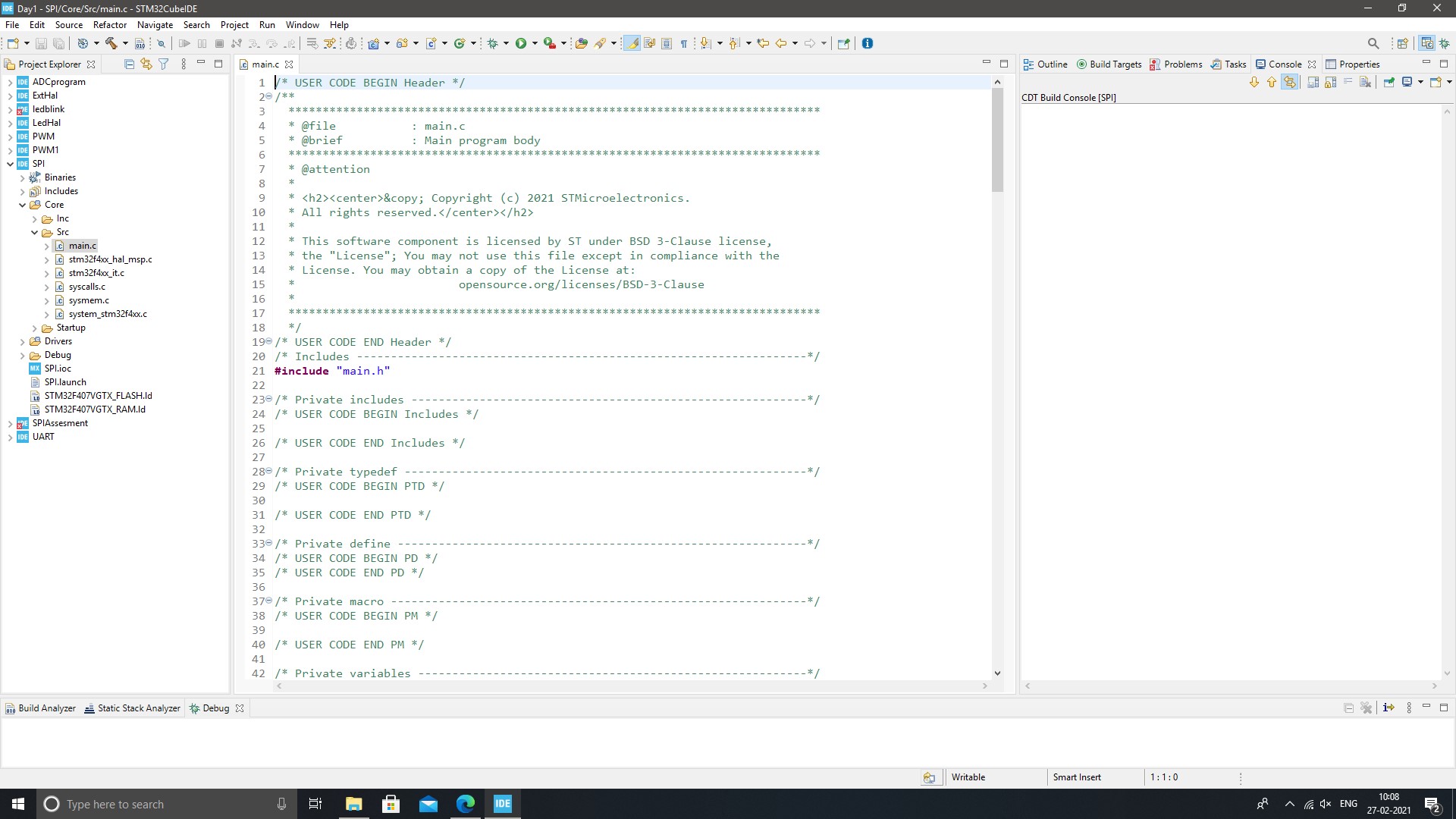
**Fig.12: configuration code**

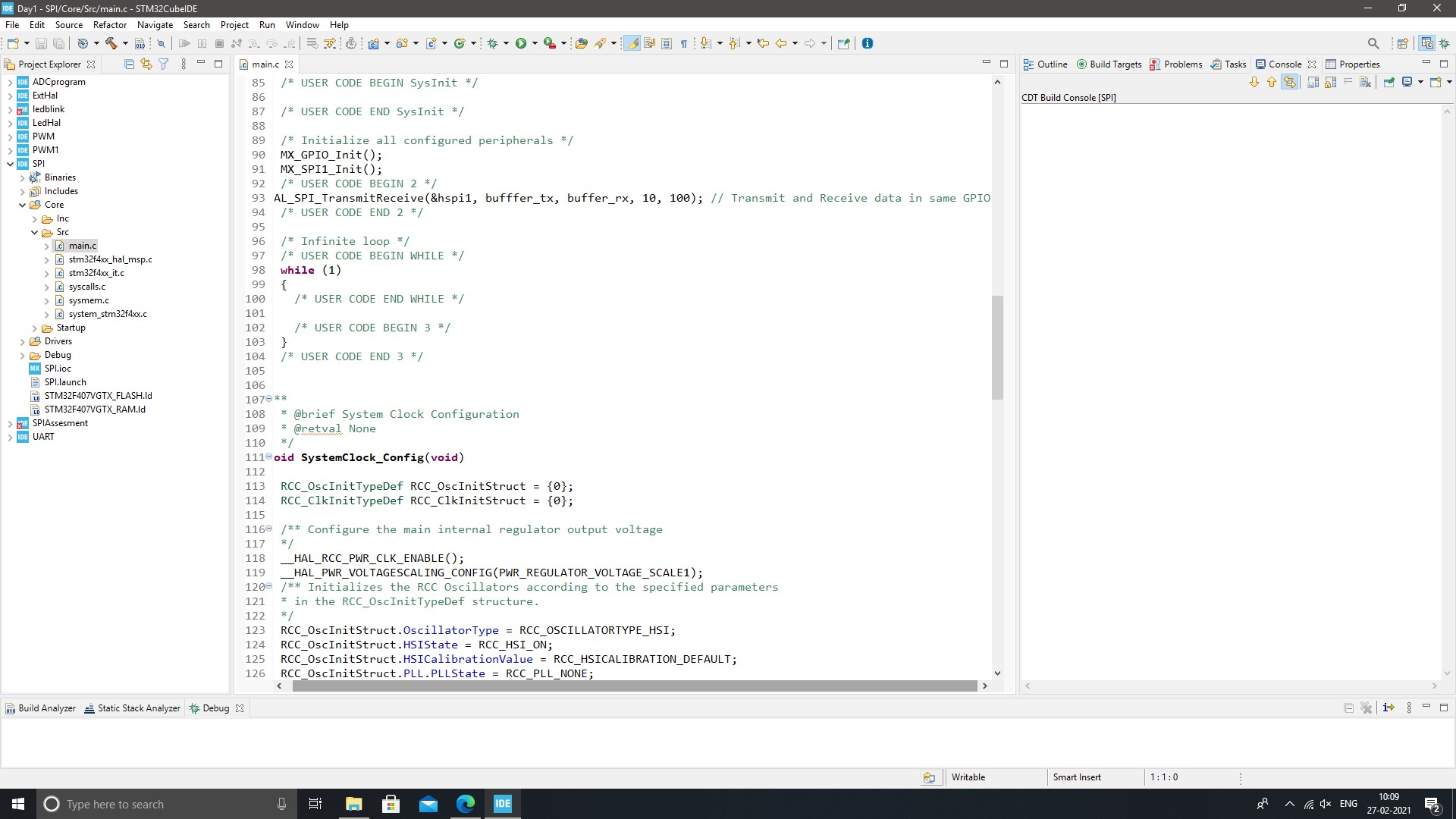


## 2.4 SPI



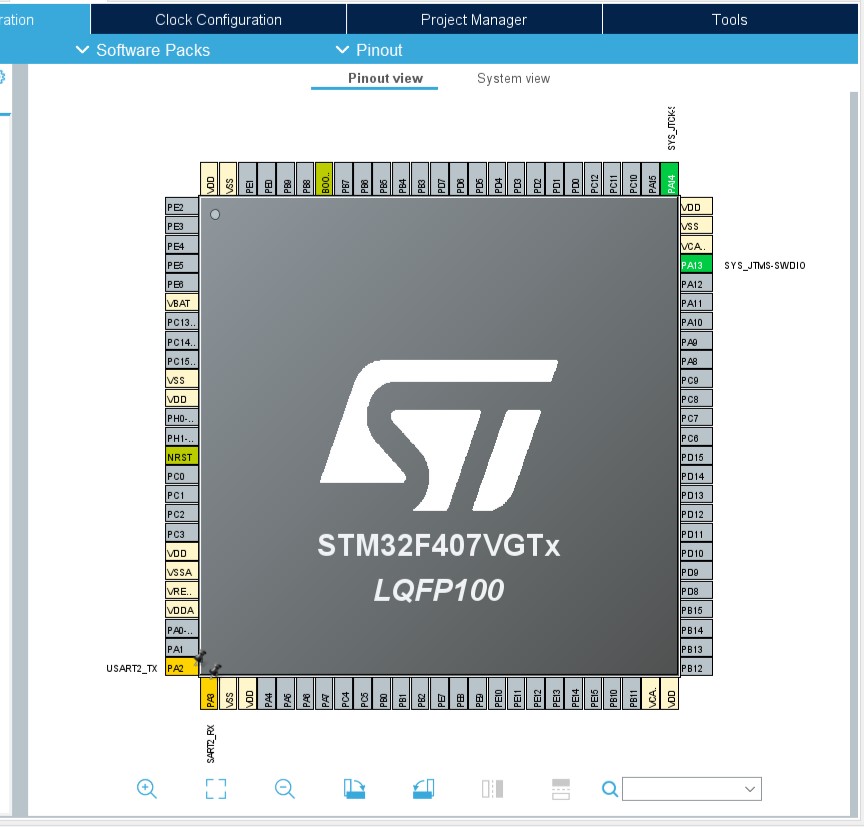
**Fig. 13: SPI Pin configuration**



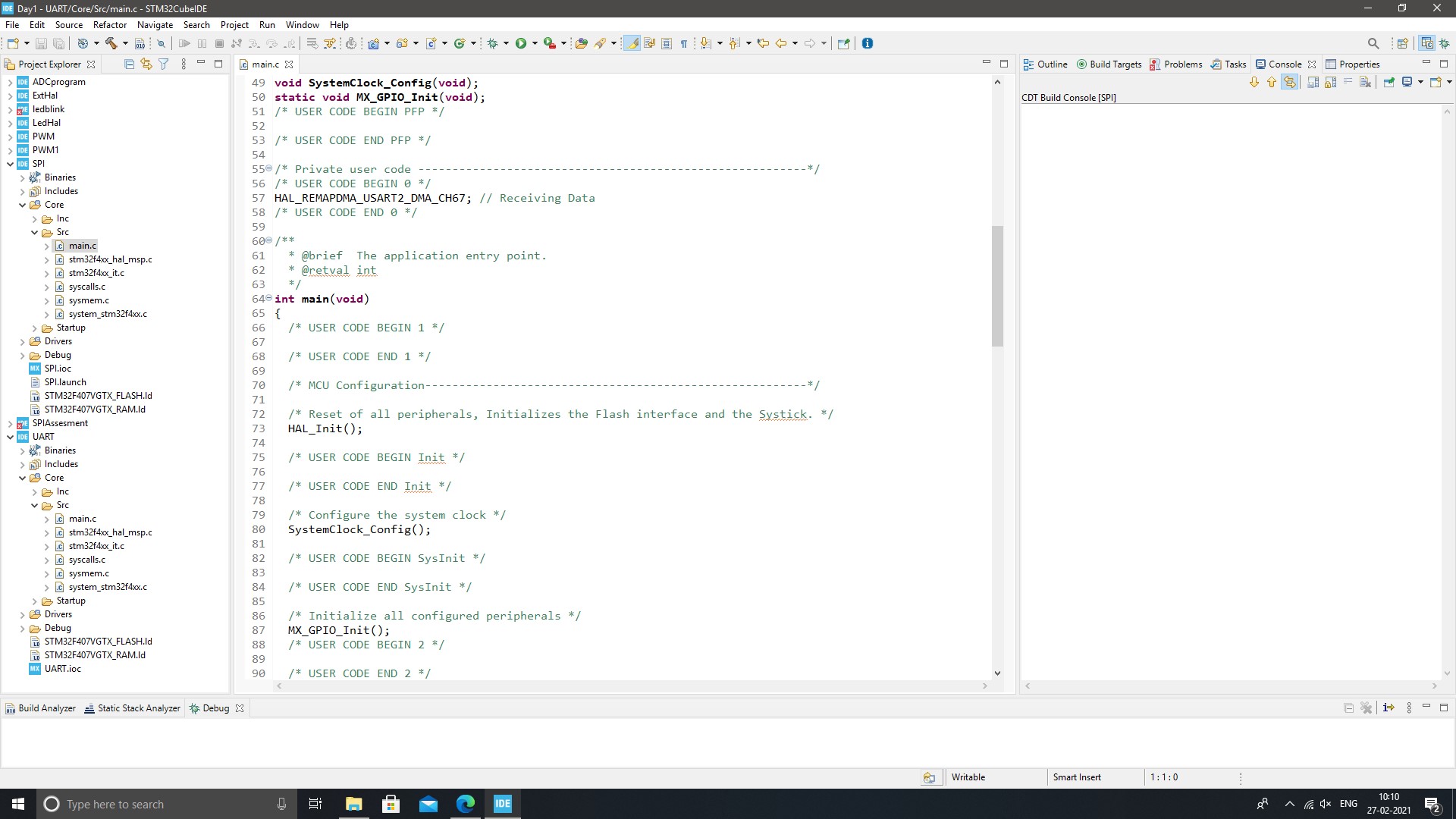
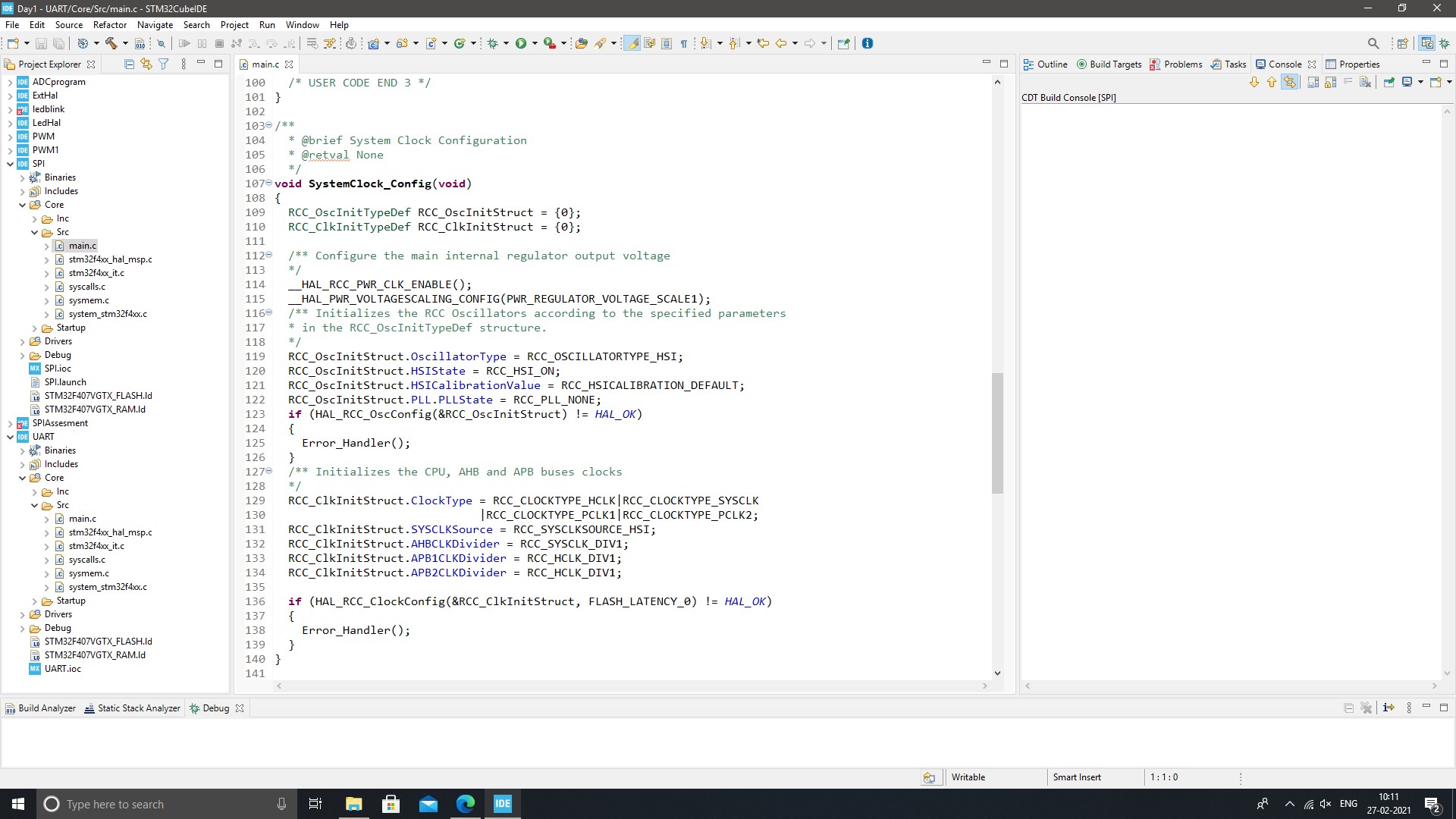


**Fig. 14: SPI configuration code**

## 2.5 UART



**Fig 15: UART Pin configuration**



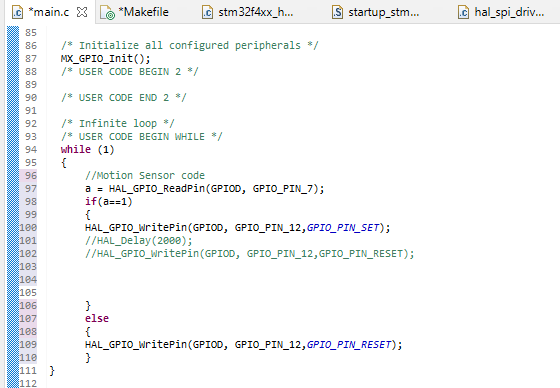
**Fig. 16: UART configuration code**

**ACTIVITY 3: BODY CONTROL MODULE (BCM) :**

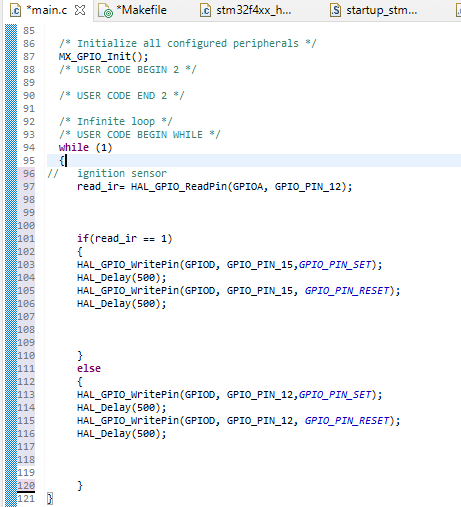
# Implementation summary

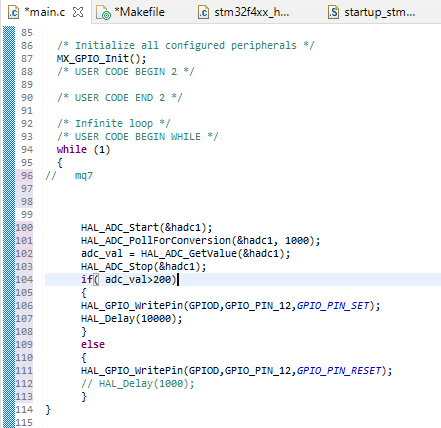
* module is a computer component in a car that monitors, controls and uses electrical devices tool was controlled by a different electrical component. There was a cooling module, an interior lighting module, a door lock module, and so on. The body control module combines all these different modules under a single system to work together instead of partitioning, a simple format for both production and problem solving. While there are many different types of body control modules, they are generally the same, the sensors for connecting, switching and automatic switching together in a single computer system. These components are subdivided into inputs, such as sensor data about temperature or speed, and the effects, or the way a computer control system responds to control motor performance. Inputs and outputs are further divided into analog and digital information types - analog signals used by modular continuous modules (such as oil pressure) and digital signals used for modules that can either turn on or off (such as headlights or oil indicator light).

**Code Snaps:-**

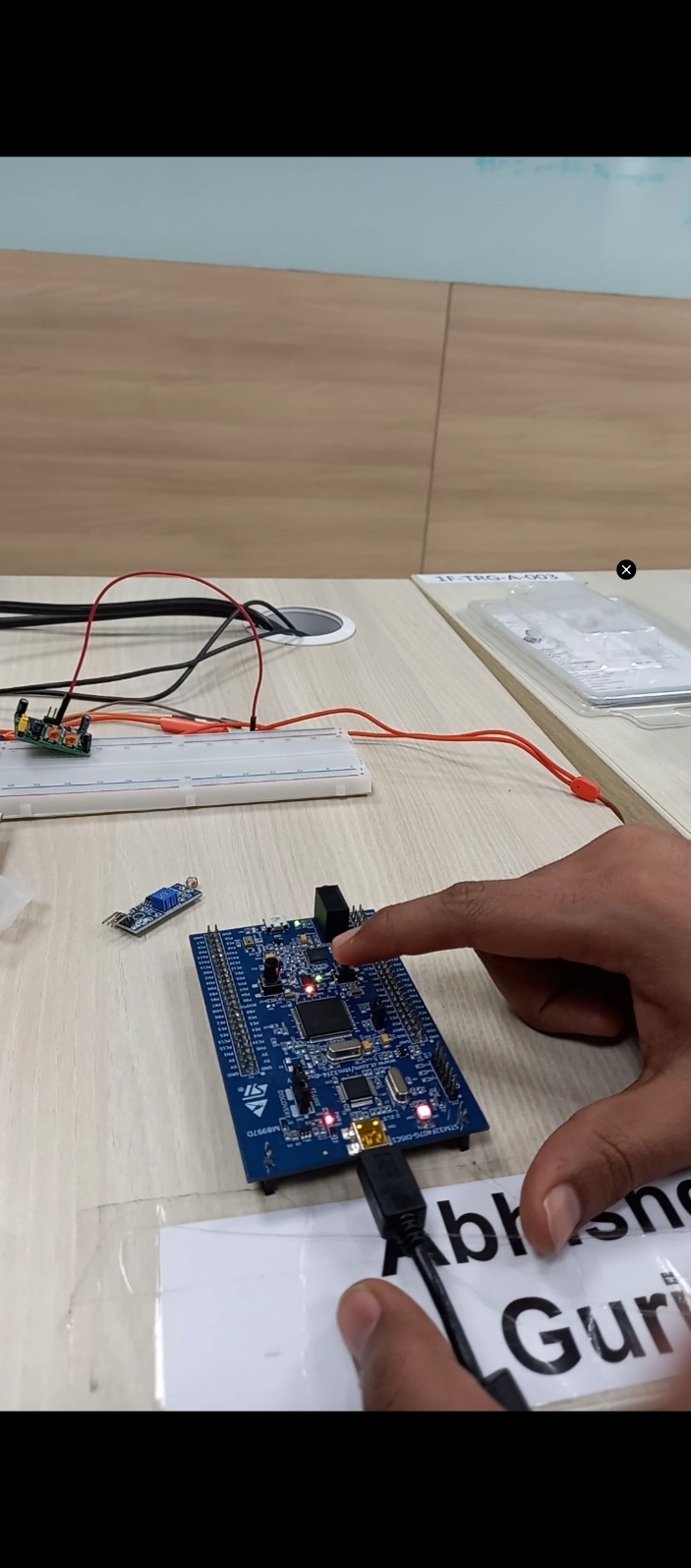
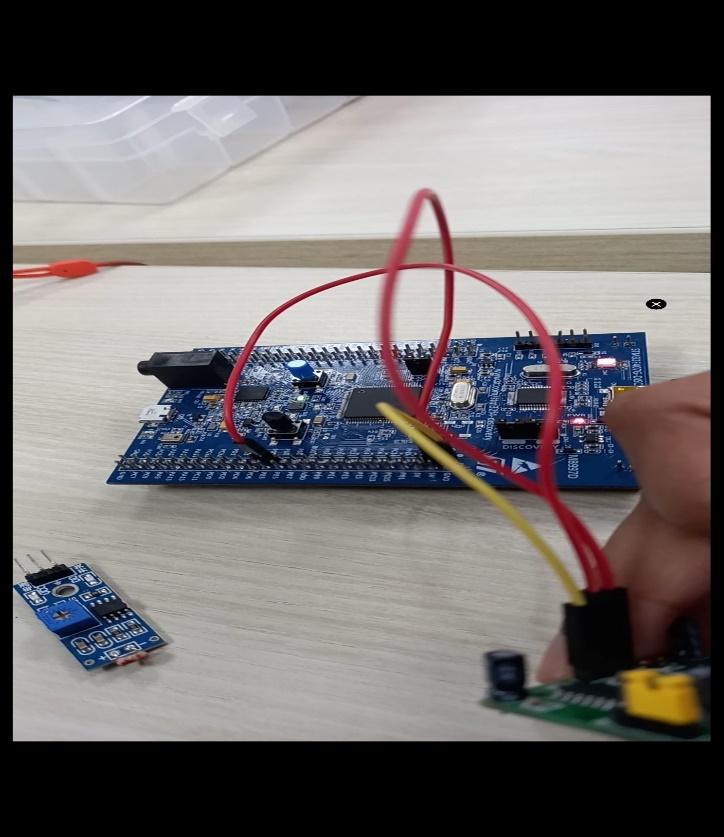


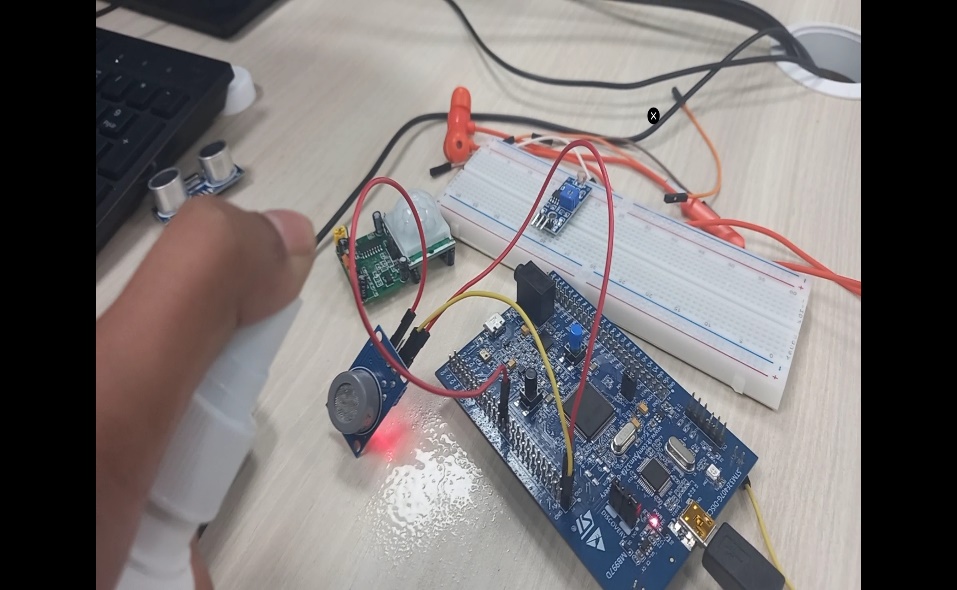
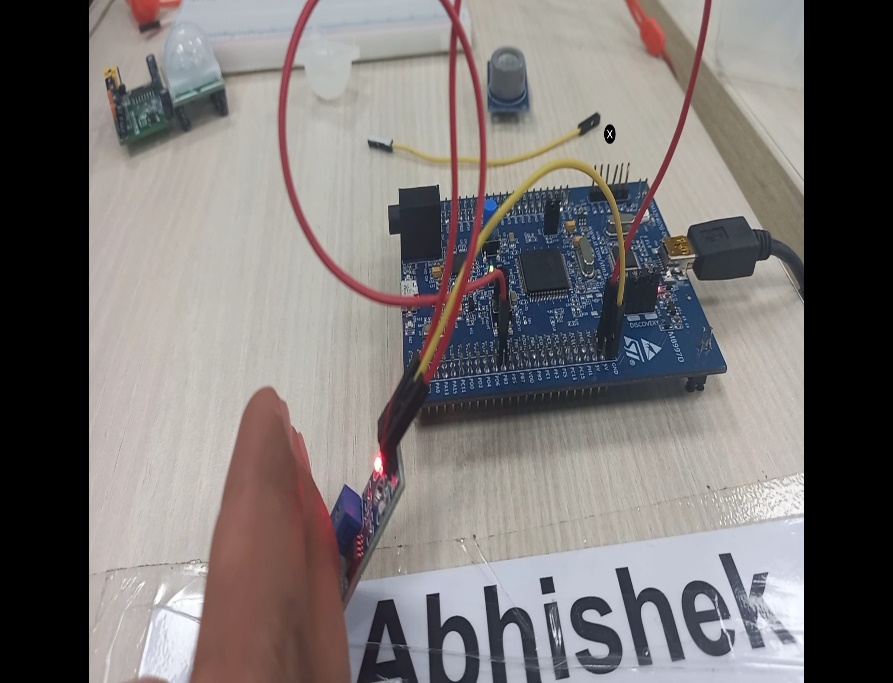




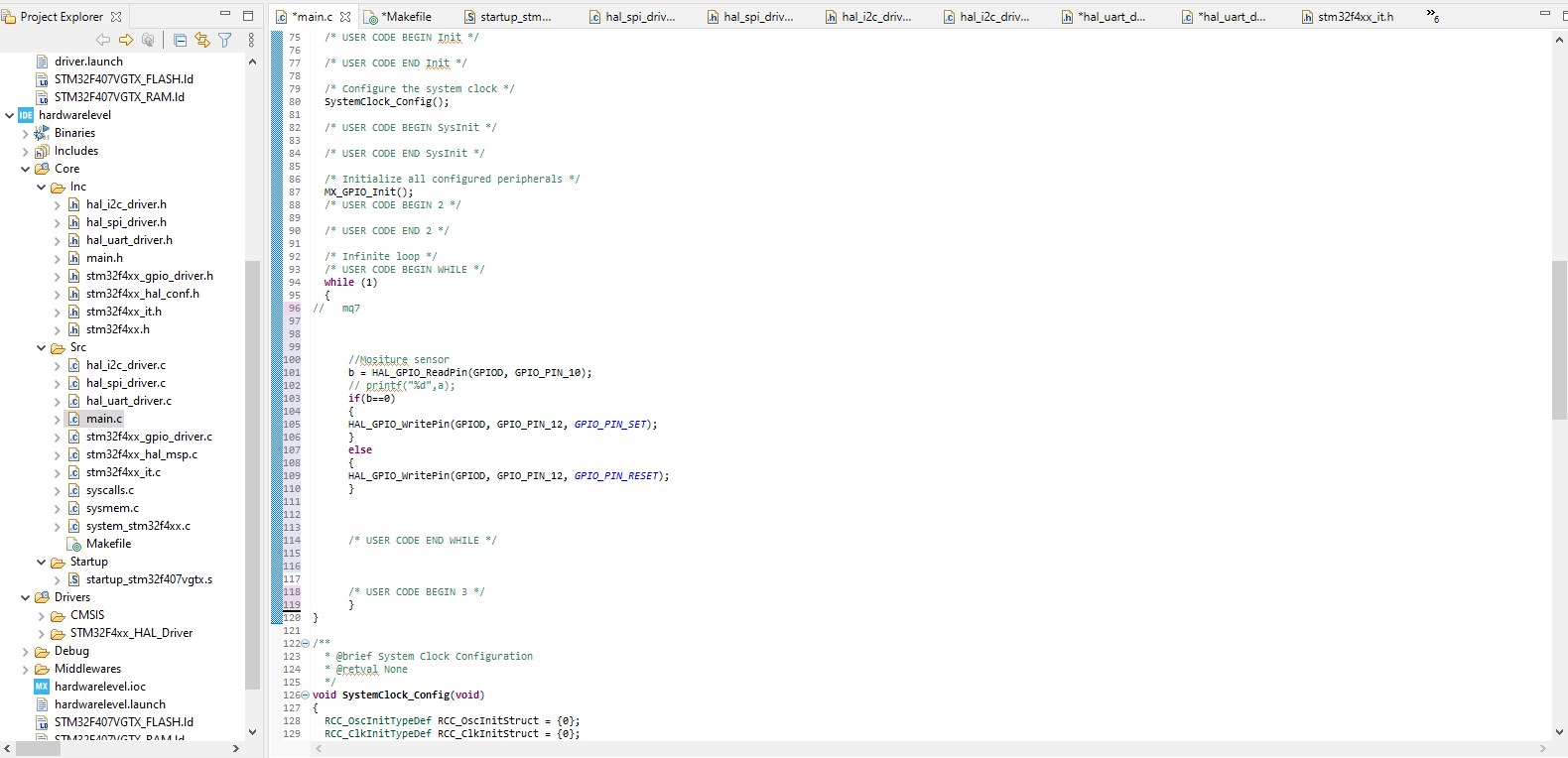


**Few Glimpses of the project:**



**Fig. 17: Project pictures**



**Fig. 18: Project Interface**

**SUMMARY :-**

* In this project the work is done over the testing of cross compilation on chip board i.e. to write code over different machine and compile over different development tool like STM32CubeIDE where a state-of-the-art C / C ++ development platform with boundary modification, code production, code integration, and debugging features for STM32 microcontrollers and microprocessors.
* In our project we have implemented six features on a microcontroller STM32f4 discovery board. In the project we integrated IR sensors, Alcohol sensor (mq7), LDR sensor, Moisture sensor, ignition sensor with microcontroller STM32f4 using GPIO protocol.
* While doing this project main part is STM32CubeIDE combines STM32 configuration and project creation functionality from STM32CubeMX to provide the same tools experience and save installation time and development time. After the selection of the STM32 MCU or blank MPU, or a customized microcontroller or microprocessor from the board selection or model selection, the project is created, and a code is generated. At any time during development, the user can revert to implementation and configuration or middleware and update the startup code without affecting the user code.

# Advanced Python Programming Summary Report

# Modules:-

Modules linked with this miniproject:-

* PyCharm environment IDE.

# Objective :-

To write program in python to extract data from sheets by giving one or more input and validate them and then to put the whole data of all the sheets of the same in Master sheet. Which will store them.

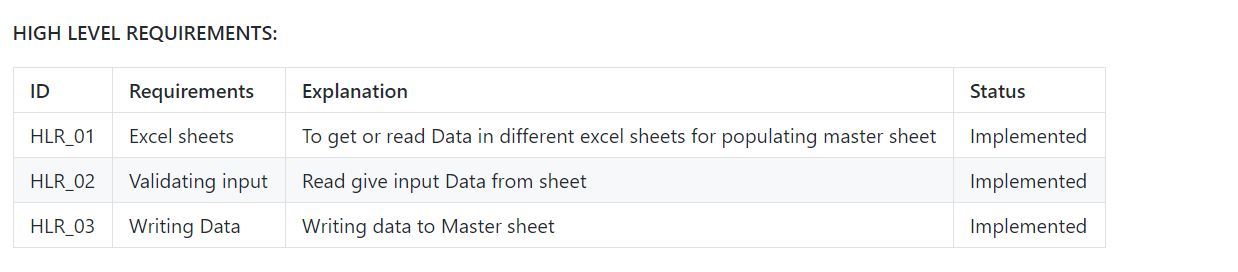
# Requirements :-

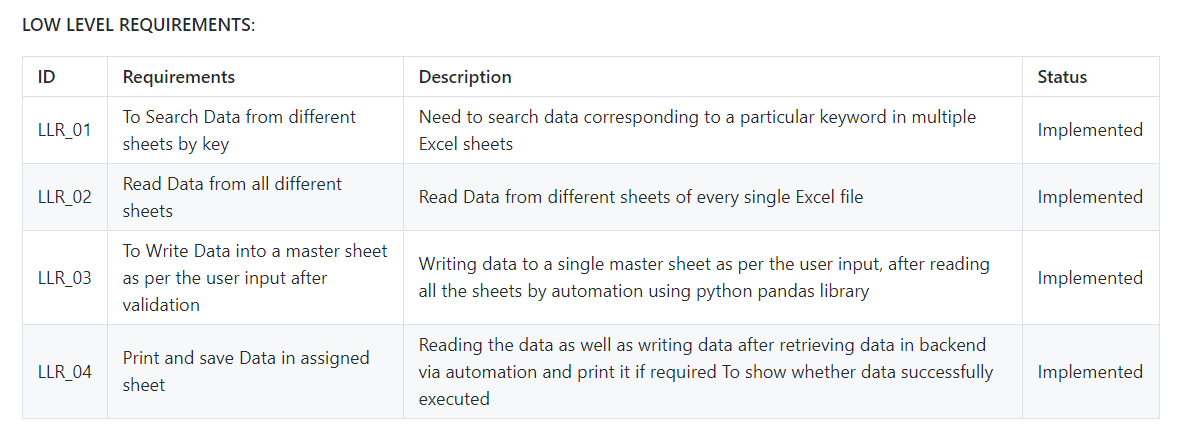
* Excel sheets with data populated with two to three attributes as unique key or primary key.

## **Defining the idea for project :**

Customized code is used to read and write data. An excel sheet made by hand containing 5 sheets and one master sheet. Here we search for the details of the person associated with the specific name and registration number on all 5 sheets. Once the data has been downloaded from the all sheets it will be printed on the master sheet. All its usage is used to read better search and write file. The code makes studying easier in the field of data science where there is a lot of data that needs to be extracted.

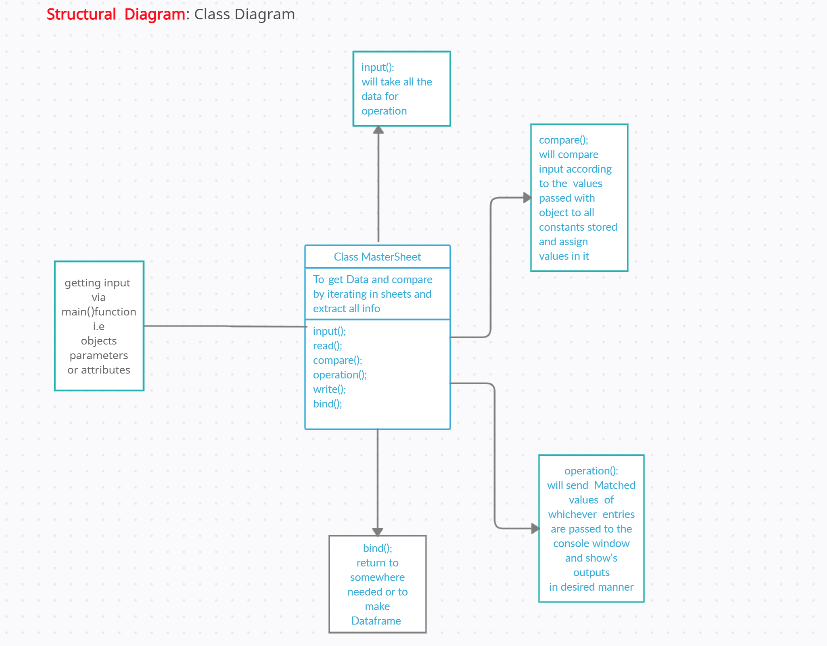
# Detailed Requirements:-



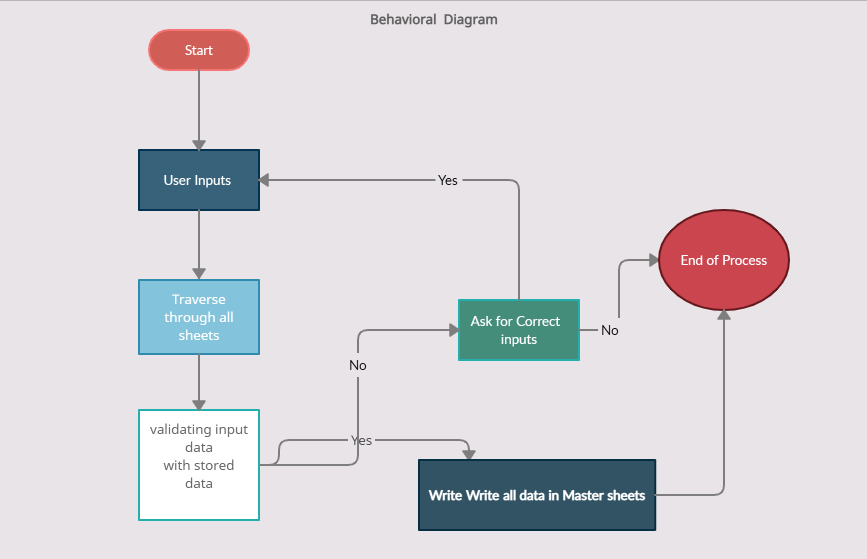


# DESIGN:-

## Structral Diagram



## Behavioral Diagram



# SOURCE CODE:-

import pandas as pd

import openpyxl as op

from openpyxl import Workbook

from openpyxl.utils.dataframe import dataframe\_to\_rows

def read\_path\_of\_files():

numberof\_sheets = int(input

("Enter Number of sheets To "

"read And extract data: "))

print("1. Enter the Paths of ", numberof\_sheets, " Excel sheets")

for i in range(1, numberof\_sheets + 1):

print('Path of sheet', i, ':')

filepath\_ofSheet.append(input())

def read\_files():

for fpath in (filepath\_ofSheet):

workbok1 = op.load\_workbook(fpath)

dff.append(pd.read\_excel(fpath))

namesheet = workbok1.sheetnames

print(namesheet)

allsheetnames.append(namesheet[0])

wb\_read.append(op.load\_workbook(fpath))

def write\_files\_in\_one():

with pd.ExcelWriter('auto5sheets1.xlsx') as writer:

for (df, a) in zip(dff, allsheetnames):

df.to\_excel(writer, sheet\_name=a, index=False)

def header\_in\_new\_sheets():

MasterSheet = Workbook()

wb\_read = op.load\_workbook("auto5sheets1.xlsx")

wsheet = MasterSheet.active

wsheet.title = 'output'

# sheets\_read = wb\_read.sheetnames

for sheet in wb\_read.sheetnames:

rs = wb\_read[sheet]

Maxrow = rs.max\_row

Maxcol = rs.max\_column

if sheet == allsheetnames[0]:

for j in range(1, Maxcol + 1):

value = rs.cell(row=1, column=j).value

wsheet.cell(row=1, column=j).value = value

else:

print("You are in now else block")

maxColmaster = wsheet.max\_column

for j in range(7, Maxcol + 1):

value = rs.cell(row=1, column=j).value

print(value)

maxColmaster = maxColmaster + 1

wsheet.cell(row=1, column=maxColmaster).value = value

MasterSheet.save('Openpyxl.xlsx')

wb\_read = op.load\_workbook("auto5sheets1.xlsx")

count = int(input("Enter how Many Data you want to read"))

for i in range(1, count + 1):

print('Enter Details for the ', i, ' Data')

FirstName = str(input('Enter First name '))

Email = str(input('Enter email Id'))

PS\_No = int(input('Enter PS Number '))

Datatoload = []

for sheet in wb\_read.sheetnames:

rs = wb\_read[sheet]

Maxrow = rs.max\_row

Maxcol = rs.max\_column

for i in range(2, Maxrow + 1):

if rs.cell(row=i, column=1).value == \

PS\_No and rs.cell(row=i, column=2).value == \

FirstName and rs.cell(

row=i, column=6).value == Email:

if sheet == allsheetnames[0]:

for j in range(1, Maxcol + 1):

Datatoload.append(rs.cell(row=i, column=j).value)

else:

for j in range(7, Maxcol + 1):

Datatoload.append(rs.cell(row=i, column=j).value)

df = pd.DataFrame(Datatoload)

df = df.T

for r in dataframe\_to\_rows(df, index=False, header=False):

wsheet.append(r)

MasterSheet.save('Openpyxl.xlsx')

if \_\_name\_\_ == '\_\_main\_\_':

filepath\_ofSheet = []

allsheetnames = []

dff = []

wb\_read = []

read\_path\_of\_files()

read\_files()

write\_files\_in\_one()

header\_in\_new\_sheets()

# D:\Python\python\_project\a\quiz\_1\_grades.xlsx

# D:\Python\python\_project\b\quiz\_2\_grades.xlsx

# D:\Python\python\_project\c\quiz\_3\_grades.xlsx

# D:\Python\python\_project\d\quiz\_4\_grades.xlsx

# D:\Python\python\_project\e\quiz\_5\_grades.xlsx

# 99003700 Richard Bennett Male 22 [richard.bennett@univ.edu](mailto:richard.bennett@univ.edu)

# 99003701 Timothy Parker Female 25 [timothy.parker@univ.edu](mailto:timothy.parker@univ.edu)

# 99003704 Michael Taylor Male 20 [michael.taylor@univ.edu](mailto:michael.taylor@univ.edu)

# Code Quality:-

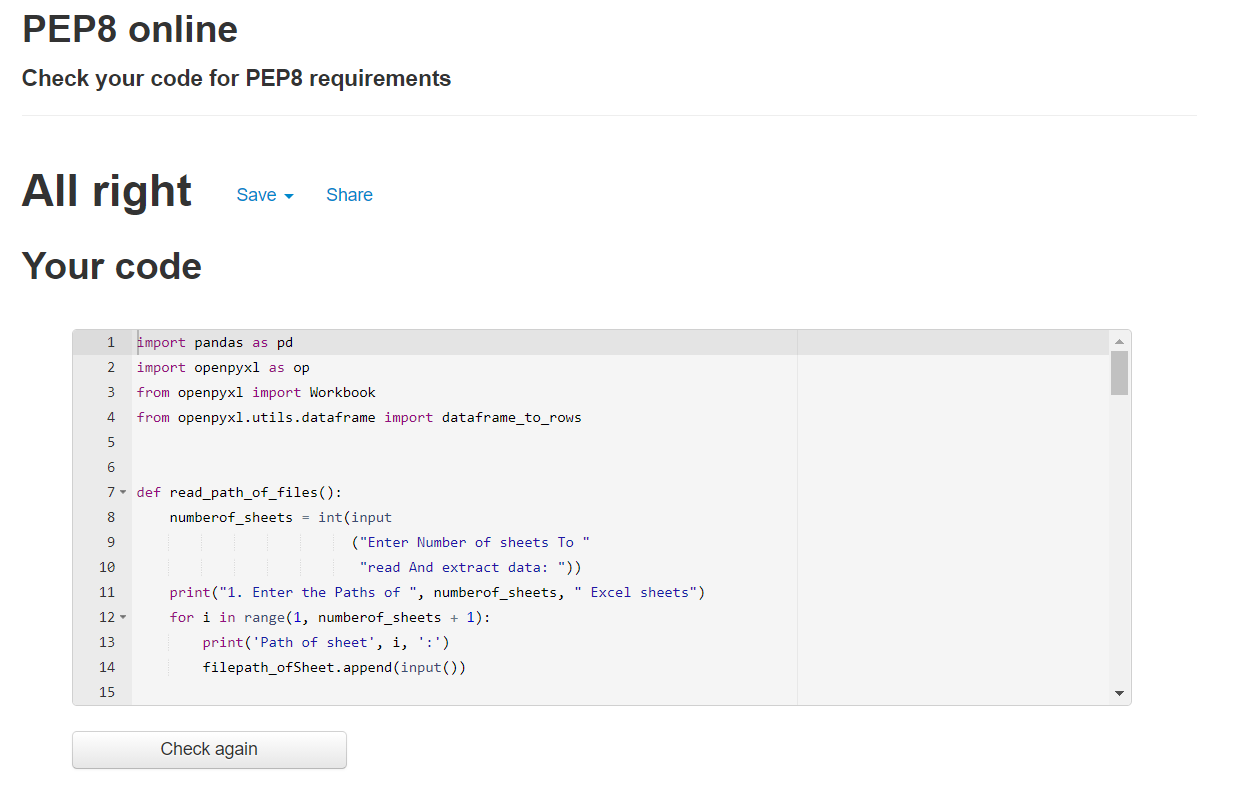


Figure 1 Code Quality

# GITHUB LINK:-

[https://github.com/99003726/ Miniproject\_Python.git](https://github.com/99003726/%20Miniproject_Python.git%20)

# Summary

In this project learning outcome is how to use frameworks of python to populate one excel sheets with few lines of code by using few common aspects which are as common unique identifiers among al the considered sheets .

# MiniProject 4 :- Embedded Linux & Kernel Device Drivers Programming

## Module/s

Modules linked to this miniproject is Ex –Embedded C, Linux, Kernel Device Drivers and OS programming.

### ****Topic and Subtopics****

1. Working with QEMU:-

* Setting up Qemu
* Building Custom Kernel
* Cross Compilation Techniques
* Booting Techniques

1. Kernel Programming and Device Drivers

* Introduction to Kernel
* Static and Dynamic Modules
* Kconfig entries
* Adding System Calls
* Simple Drivers
* File Operations
* Kernel Data structures
* Concurrency and Inter Process Communication (IPC)
* IOCTL Operations

## Objectives & Requirements

Objective:

The objective of this project is to implement the learning of the module in the following activities:

* Making a System call to echo back the given string
* Making a System Call to pass the Arguments
* Making a System call to traverse process list and print PID and PPID of a process and making a System call to retrieve the attributes of calling process
* Implementing IOCTL to echo back the string, implement two operations such that the string passed by one operation is retrieved by the other operation
* Applying mutual exclusion between first two threads by implementing three kernel threads. First two threads will write N nodes into a kernel list. The third thread waits for the first two threads and traverses the list while mutual exclusion is applied between the first two threads. Exit method cleanup the list nodes

Requirements:

High Level Requirement:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Description | Status |
| HLR\_01 | System Call to echo back string | The system call echoes back the given string on target from user space | Implemented |
| HLR\_02 | System Call to pass the argument | The System Call passes the argument from user space to Kernel Space | Implemented |
| HLR\_03 | System Call to traverse process list and print PID and PPID and retrieve attributes of the calling process | The System Call traverse the process list, prints PID and PPID and retrieve attributes of the calling process on the target window | Implemented |
| HLR\_04 | IOCTL to echo back string using two operations | Use IOCTL to echo back string by implementing 2 operations, one for passing string and other of for retrieving it. | Implemented |
| HLR\_05 | Threads implementation of 3 threads where 3rd third retrieves data for first 2 threads with list of N nodes and its traversal | Applying mutual exclusion to implement 3 threads where 2 write N nodes to threads and 3rd traverses the list. Exit module cleans up the node. | Implemented |

Low Level Requirement:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Description | Status |
| LLR\_01 | The string is echoed back to Target or the kernel space by system call | The string “abcdef” gets echoed to Qemu by creating a system call invoked by the user space. | Implemented |
| LLR\_02 | The Argument is passed back to the target | The Argument is passed back to the kernel space from user space | Implemented |
| LLR\_03 | The System Call traverse the process list, prints PID and PPID and retrieve attributes of the calling process on the target window | User invokes a system call to get PID, PPID and attributes of the calling process | Implemented |
| LLR\_04 | IOCTL used to echo back string with 2 operations | IOCTL to echo back string using 2 operations one to pass the argument and other to retrieve data and display it on Qemu. | Implemented |
| LLR\_05 | Mutual exclusion of three threads to traverse, list and clean up nodes | First two threads will write N nodes into a kernel list. The third thread waits for it and traverses the list. Mutual exclusion applied between the first two threads. Exit method cleanup the list nodes | Implemented |

## Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Input | Expected Output | Actual Output |
| TP\_01 | The string passed by the user is echoed to target space when invoked | “abcdef” as string passed in userspace | “abcdef” is printed in target space | “abcdef” is printed in target space using command dmesg |
| TP\_02 | The argument passed by user is printed to the kernel space | 1,2,3 as argument is passed by user | 1,2,3 is passed to the kernel space | 1,2,3 is printed in kernel space using command dmesg in kernel space only |
| TP\_03 | The PID, PPID and the attributes of the process is retrieved in target space when user space code is invoked | User space invocation for the system call | PID, PPID, Process State, Priority, RT Priority, Static Priority and Normal priority are printed in kernel space. | PID, PPID, process State, Priority, RT Priority, Static Priority and Normal priority are printed in kernel space. |
| TP\_04 | 3 threads are implemented. 1st and 2nd thread creates a list of N nodes, 3rd traverses the list after waiting for 2 threads to complete. | Invocation of the code | 3rd thread waits for 1st 2 threads to create a list through N nodes. 3rd thread then traverses the list and adds in Kfifo. | 3rd thread waits for 1st 2 threads to create a list through N nodes. 3rd thread then traverses the list and adds in Kfifo. |
| TP\_05 | 3 threads are implemented. 1st and 2nd thread creates a list of N nodes, 3rd traverses the list after waiting for 2 threads to complete by mutual exclusion. | Invocation of the code | 3rd thread waits for 1st 2 threads to create a list through N nodes by mutual exclusion. 3rd thread then traverses the list. Exit method cleans up all the nodes. | 3rd thread waits for 1st 2 threads to create a list through N nodes by mutual exclusion. 3rd thread then traverses the list. Exit method cleans up all the nodes. |

### 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/99003779/Embedded_linux_and_kernel_programming.git>

### Git Dashboard

## Individual Contribution & Highlights

“Brief on Contributions by you for Team”

### Summary

The whole project is based on kernel programming with understanding of embedded linux, device drivers, concept of system call and IOCTL i.e input output control.

The Linux Kernel Driver model is a combination of all the different types of drivers that were previously used in the kernel. It aims to add bus-specific drivers to bridges and devices by integrating data collection and operating in data that are globally accessible.

System calling is a system configuration program that asks for applications from the kernel, and strace is a powerful tool that allows you to follow a small layer between user processes and Linux kernel.

  Ioctl, which means "input output control" is a type of system call to a specific device. There are only a few system calls on Linux (300-400), which are not enough to display all the functions of unique devices. The driver can then specify ioctl which allows the user app to send orders.

### Challenges faced and how were they overcome

* System call implementation.
* Depth understanding of user space and kernel space

### Future Scope (If applicable):- NA