Version Number:

Team Members :

Team No:

Module: Model Based System Engineering



./



Learning Report

–

Embedded C

**Document History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
|  | 08/03/2021 | Rajat Kumar Singh |  |  |  |

**Contents**

**ACTIVITY 1**–COMPILATION APPROACH……………………………………………………………………………………………………………..4

1.1- MAKE FILE ................................................................................................. 4**ERROR!BOOKMARK NOT DEFINED.**

1.2- STARTUP CODE………………………………………………………………………………………………………………………5

1.3- LINKER FILE…………………………………………………………………………………………………………………………….8 1.4- DEBUGGING TECHNIQUES………………………………………………………………………………………………………………9

**ACTIVITY 2**–IMPLEMENTATION OF PROTOCOLS USING STM IDE ........................................................................ 10

2.1 GPIO……………………………………………………………………………………………………………………………………………………10

2.2 EXTI……………………………………………………………………………………………………………………………………………………. .12 2.3 ADC……………………………………………………………………………………………………………………………………………………...14 2.4 SPI………………………………………………………………………………………………………………………………………………………..16 2.5 UART……………………………………………………………………………………………………………………………………………………19

**ACTIVITY 3**– PROJECT ON BCM MODULE .......................................................................................................... 21

# Activity 1 – COMPILATION APROACH

This is the complete compilation process of the sample program for ARM Cortex Mx processor based boards. Following are the compilation stages of a C program:

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:

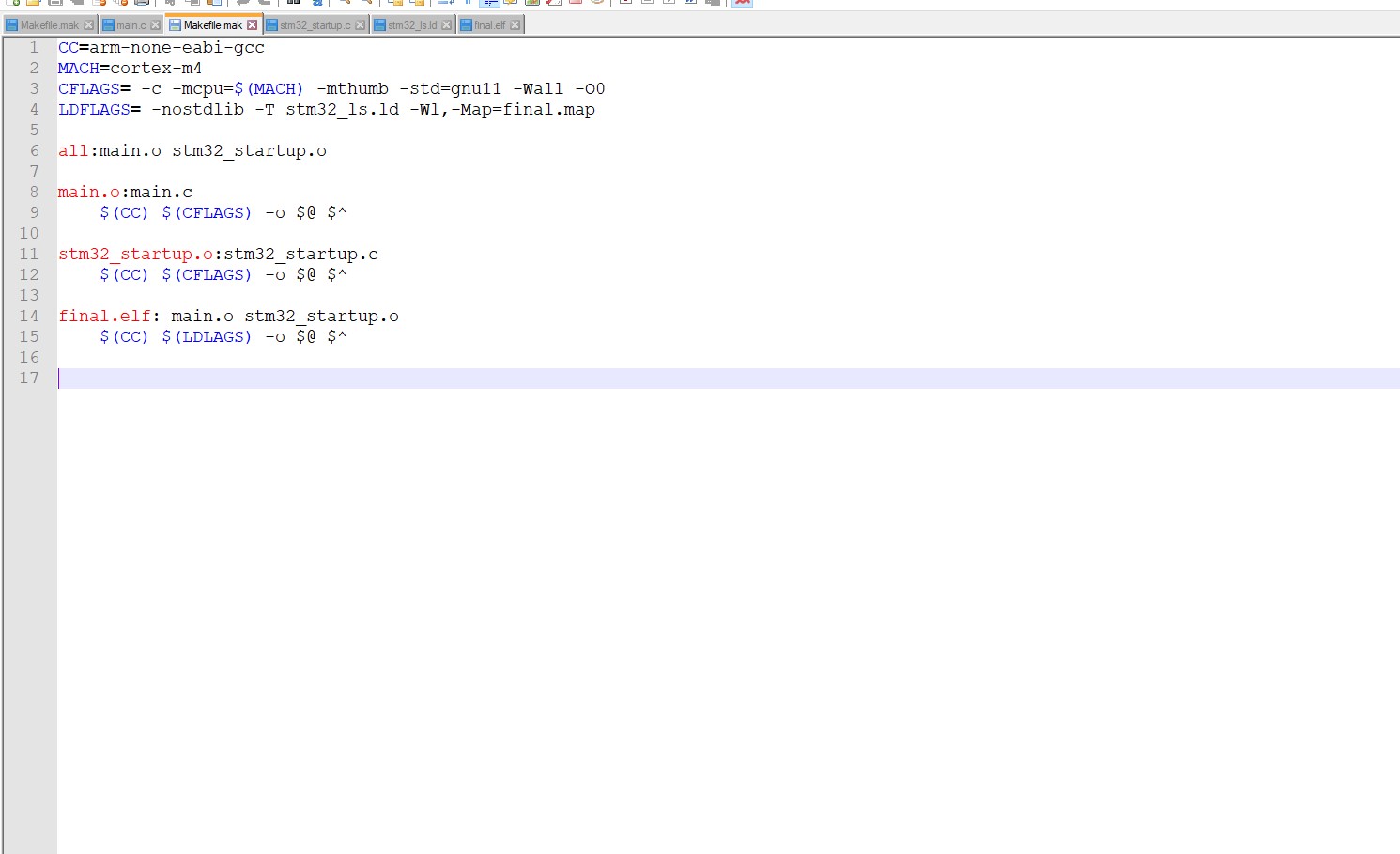


Fig 1.1.1 make file

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

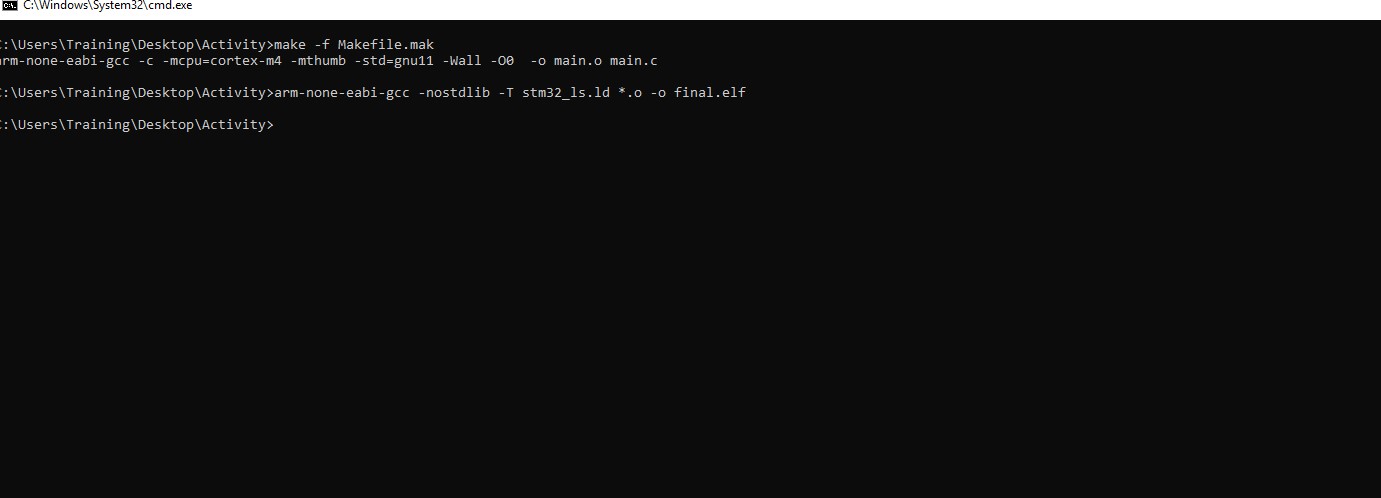


Fig 1.1.2 Make command

* **-mcpu=cortex-m4** is used to select our cortex-m4 processor which is used
* **-mthumb** is used to generate the code that executes in ARM state
* **main.o** is the target file
* **main.c** is the dependency

## 1.2- STARTUP CODE

* The startup file is responsible for setting up the right environments to run the code in main.c file.
* Some part of the startup code is target (processor) dependent.
* Role of startup file:
  1. Create a MCU specific vector table for microcontroller.
  2. To write a startup code which initializes .data and .bss section in SRAM.
  3. Call main()

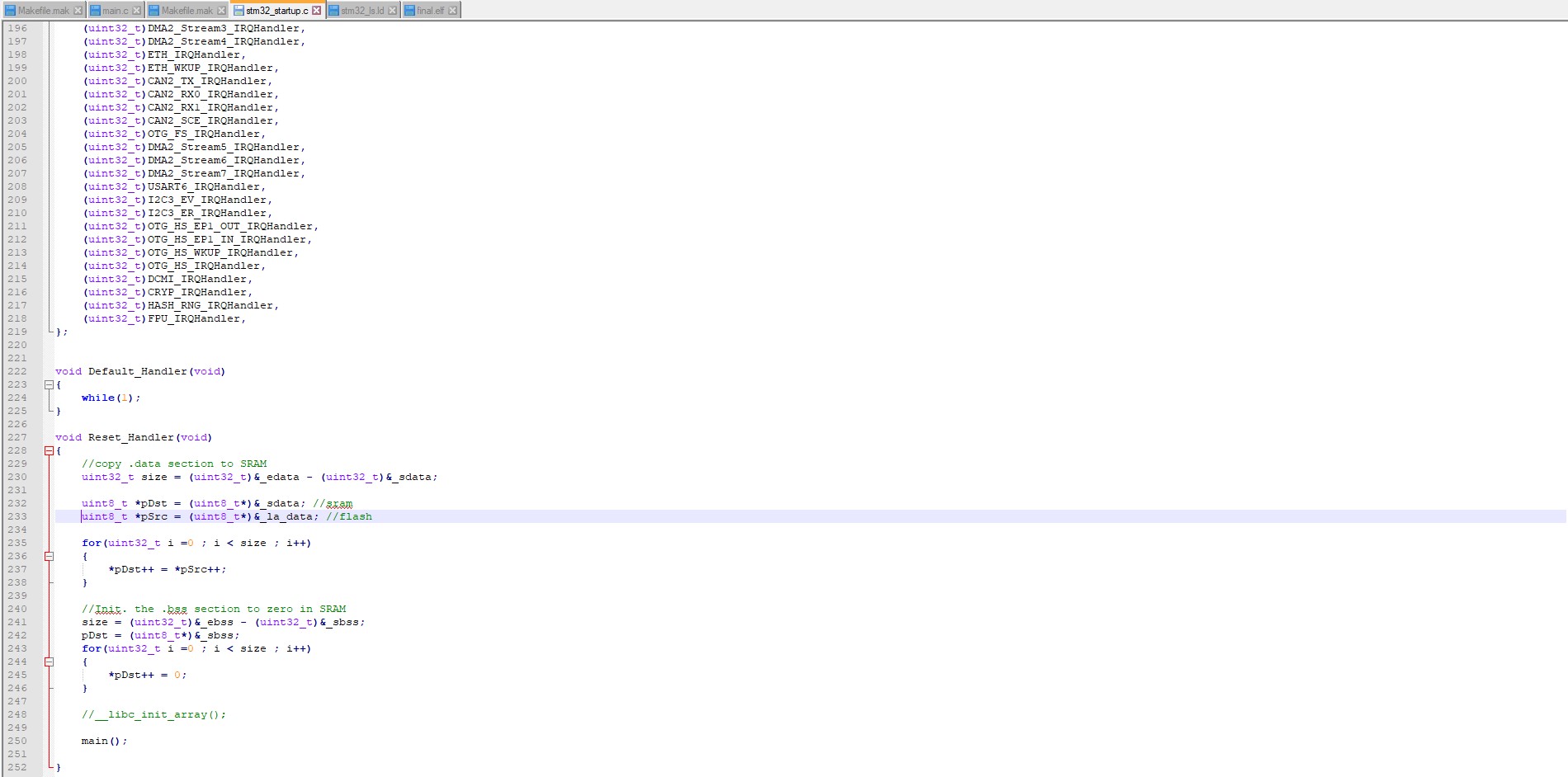
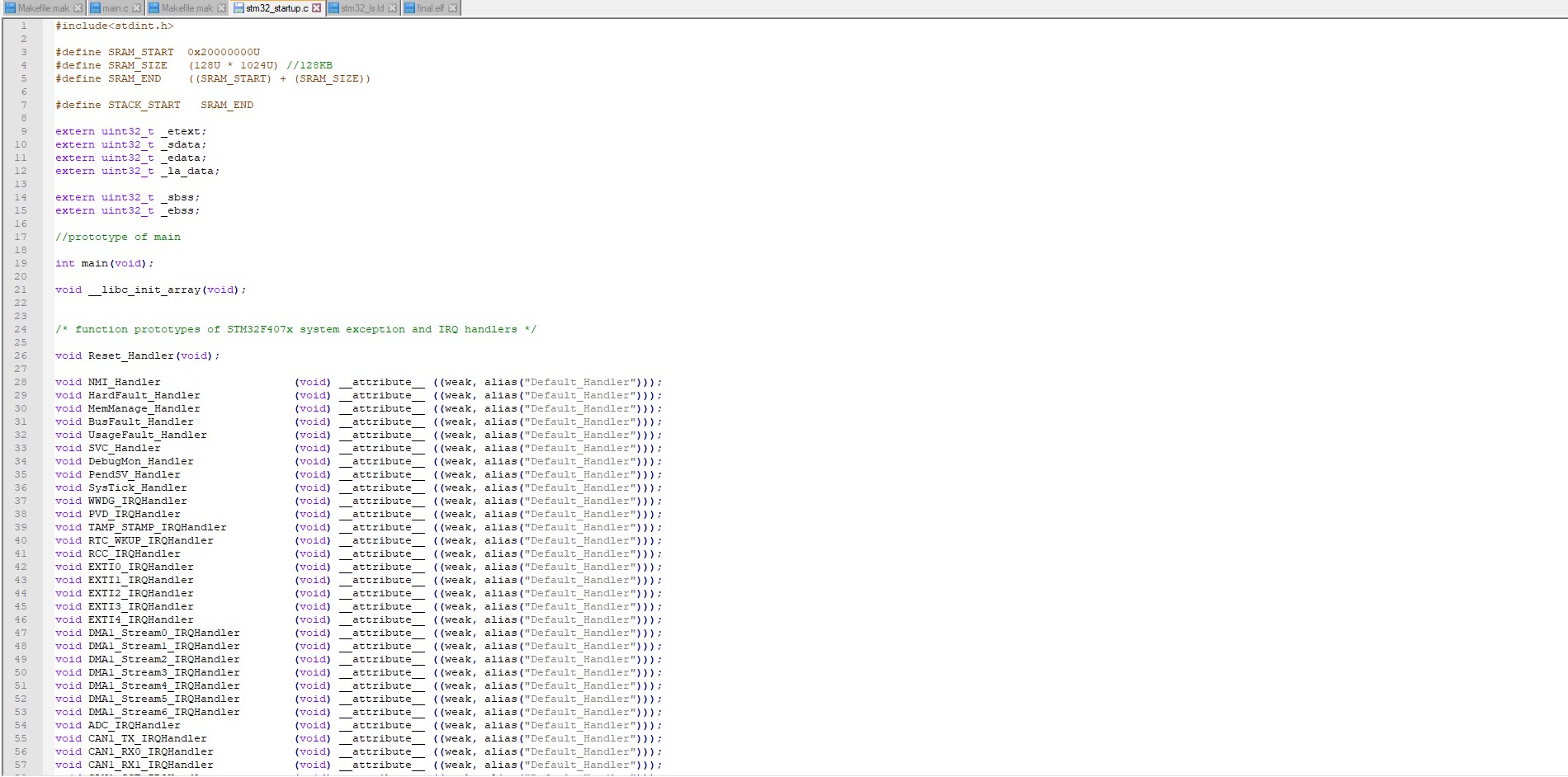


Fig 1.2.1 Startup code

In startup code we use variable attributes to store some variables in the user defined function.

Function attributes:

* Weak: Lets programmer override already defined weak function (dummy function) with the same function name.
* Alias: Lets programmer give any alias name for same function.

The startup.o file generated is of elf executable format, various sections of which are shown below:

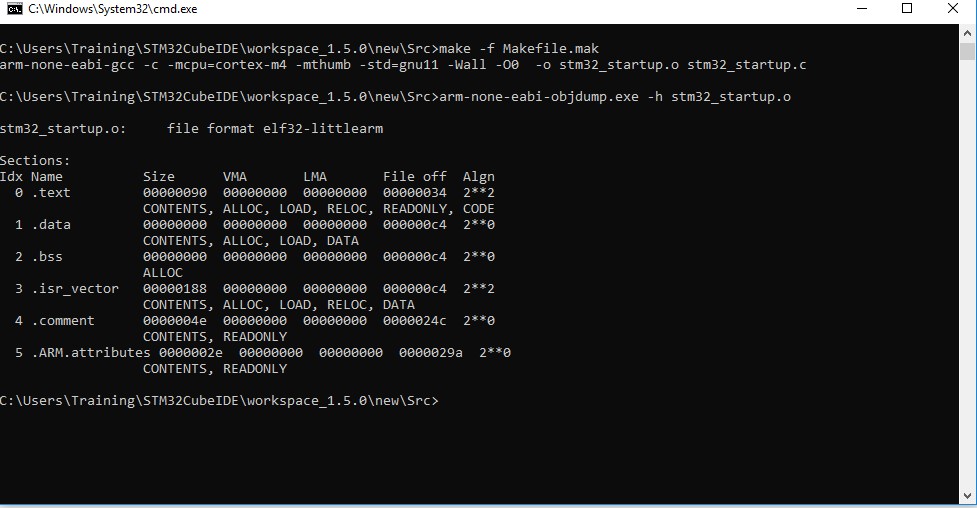


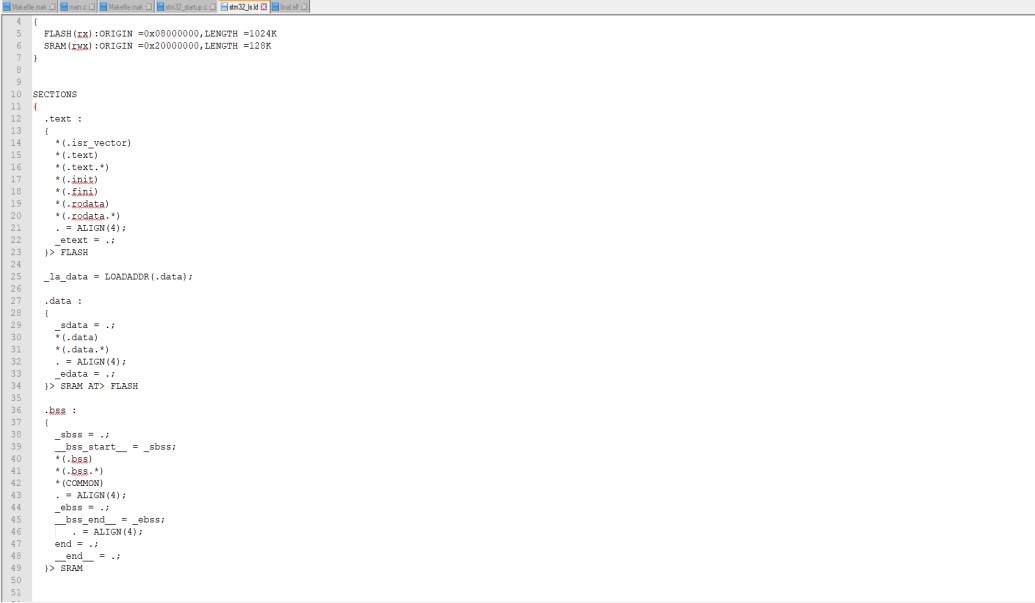
Fig 1.2.2: Startup command

## 1.3- LINKER SCRIPT

* Linkers take one or more object files or libraries as input and combines them to create a single executable file as output.
* Linker scripts decide how different sections of object file should be merged to create an output file.
* Reset handler is the entry point to the application
* Entry command is used to set the “Entry point address” information in the header of final elf file generated.

Syntax: Entry(symbol\_name)

Entry(Reset\_Handler)



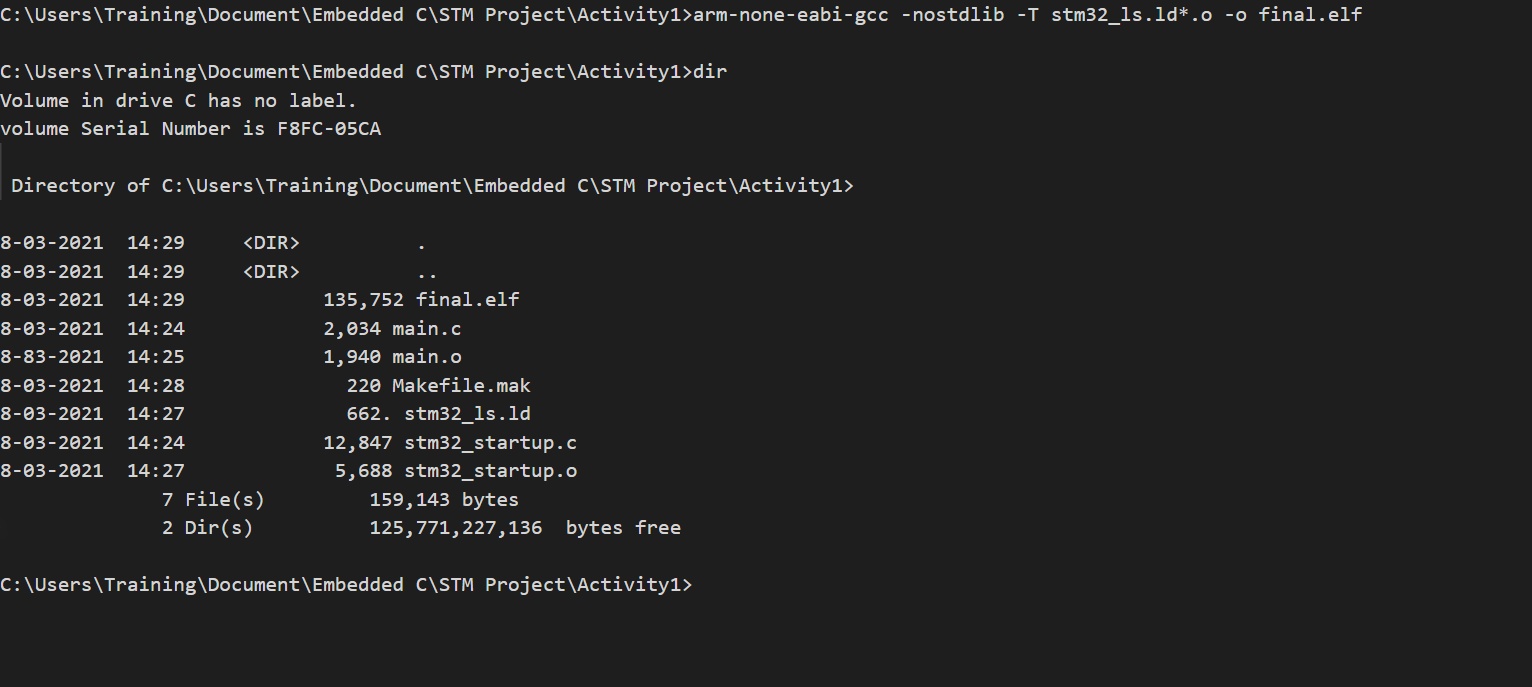


Fig 1.3.1 command to generate final.elf file

## 1.4- DEBUGGING TECHNIQUES

* The STM32F407VG is embedded with on chip debugger for debugging the code.
* The OCD ON-Chip Debugger aims to provide debugging, in system programming and boundary scan testing for embedded target devices.
* OCD is a free and opensource host application allows you to program, debug, and analyze your applications using GDB.
* It supports various target boards based on different processor architecture.

# Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

Implementation of protocols for STM32F407VG microcontroller featuring ARM32 bit ARM-cortex M4 with FPU core using HAL library.

## 2.1GPIO:

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

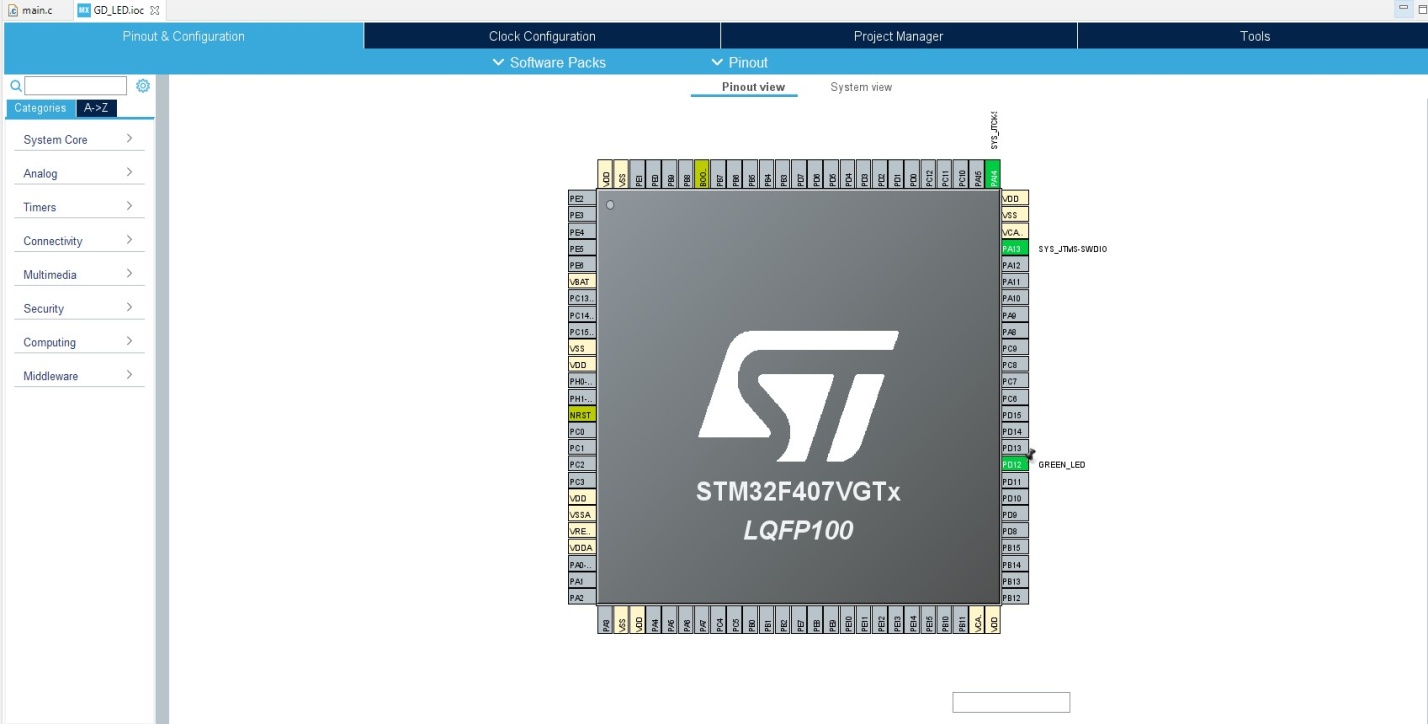


Fig: 2.1.1 GPIO pin configuration

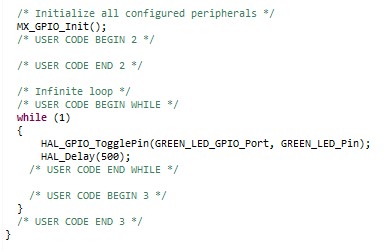


Fig: 2.1.2 GPIO configuration code

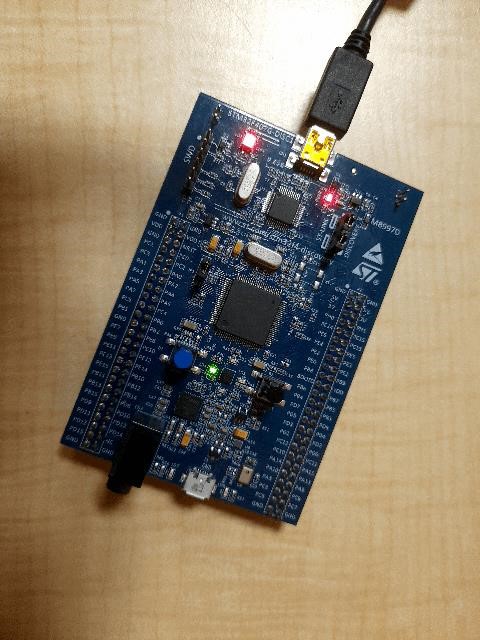


Fig: 2.1.2 LED toggling

## 2.2EXTI:

Blue button at PA0 works as an external interrupt.

When the blue button is pressed the Green LED at pin PD12 toggles.

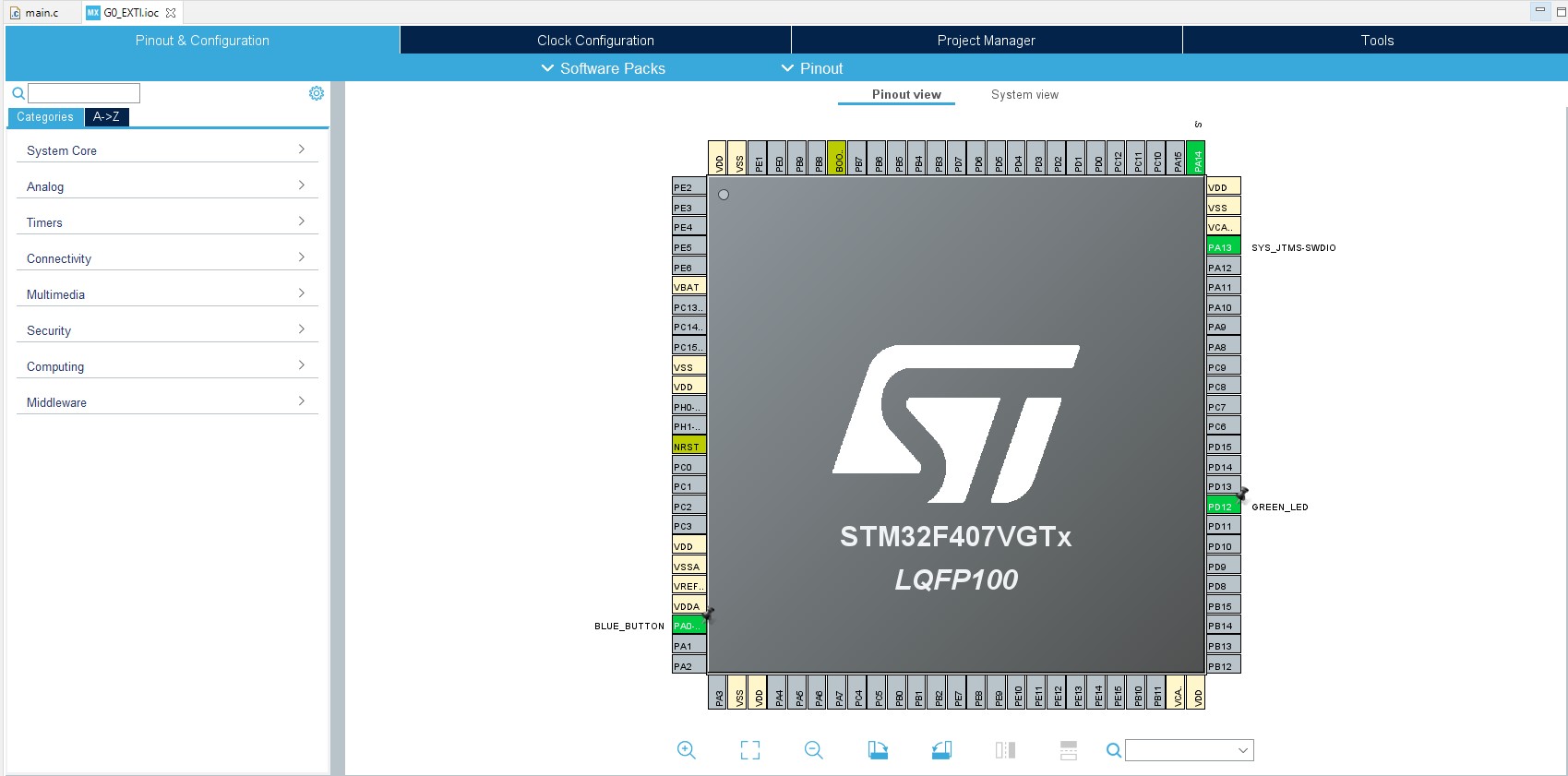


Fig: 2.2.1 EXTI pin configuration

In the main.c file a flag is initialized and if the flag == 1, the condition under the if loop executed to toggle the LED at PD12.

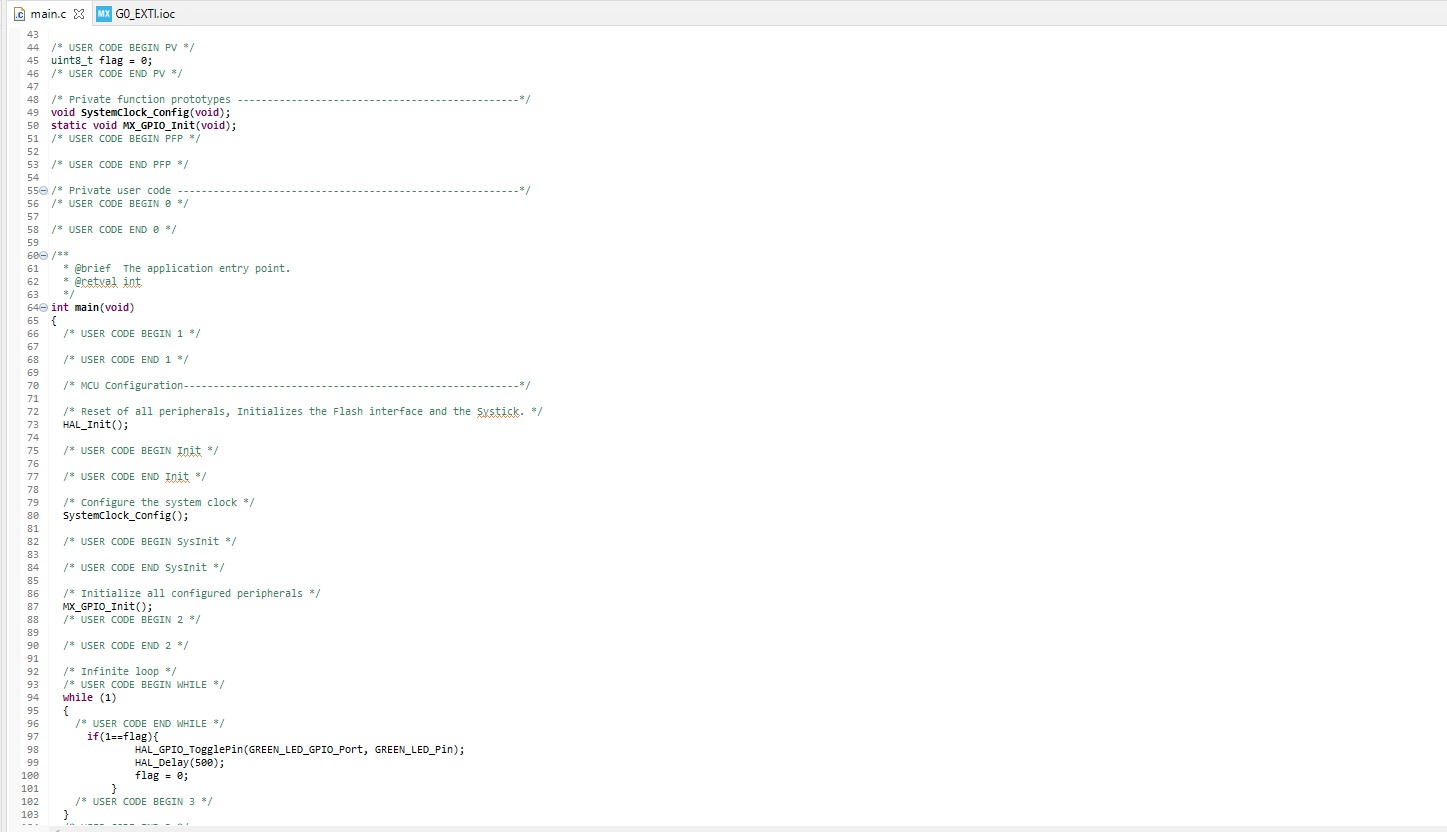


Fig: 2.2.1 EXTI configuration code

## 2.3 ADC

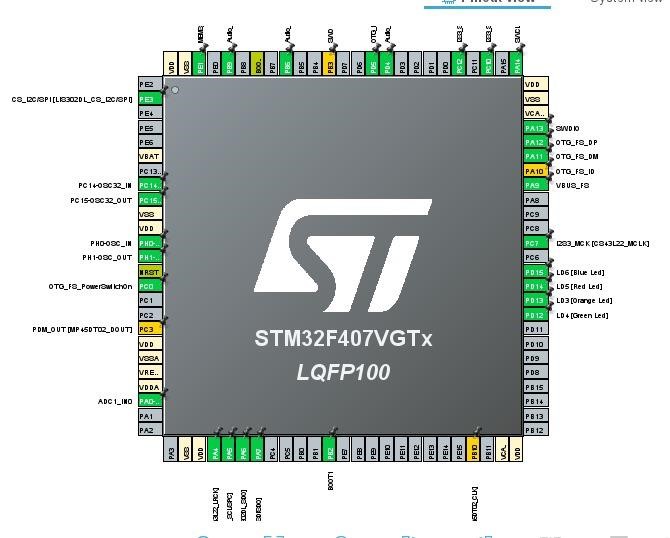
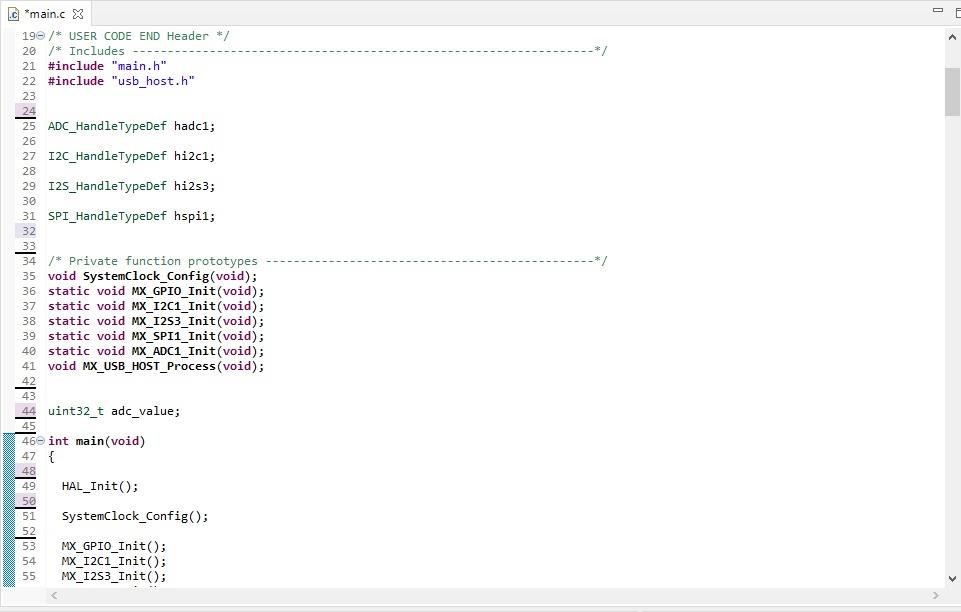
Fig: 2.3.1 ADC pin configuration

Fig: 2.3.2 ADC configuration code



## 2.4 SPI

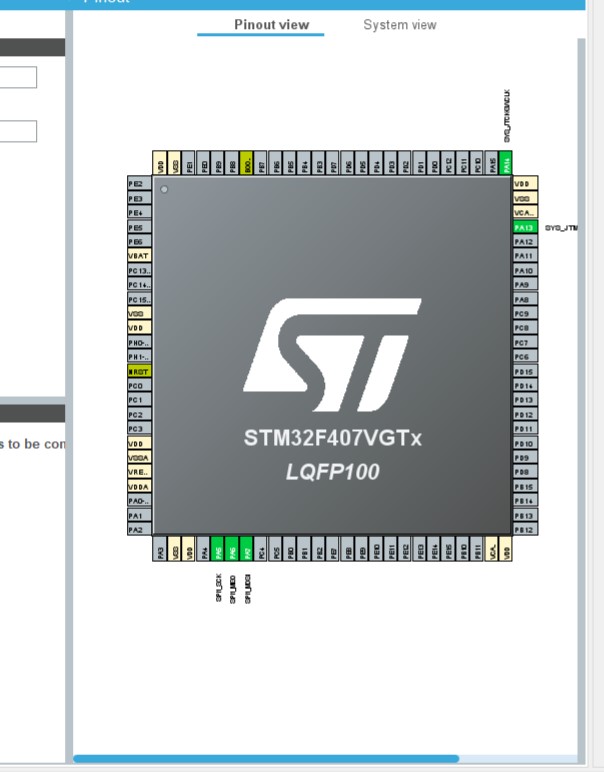
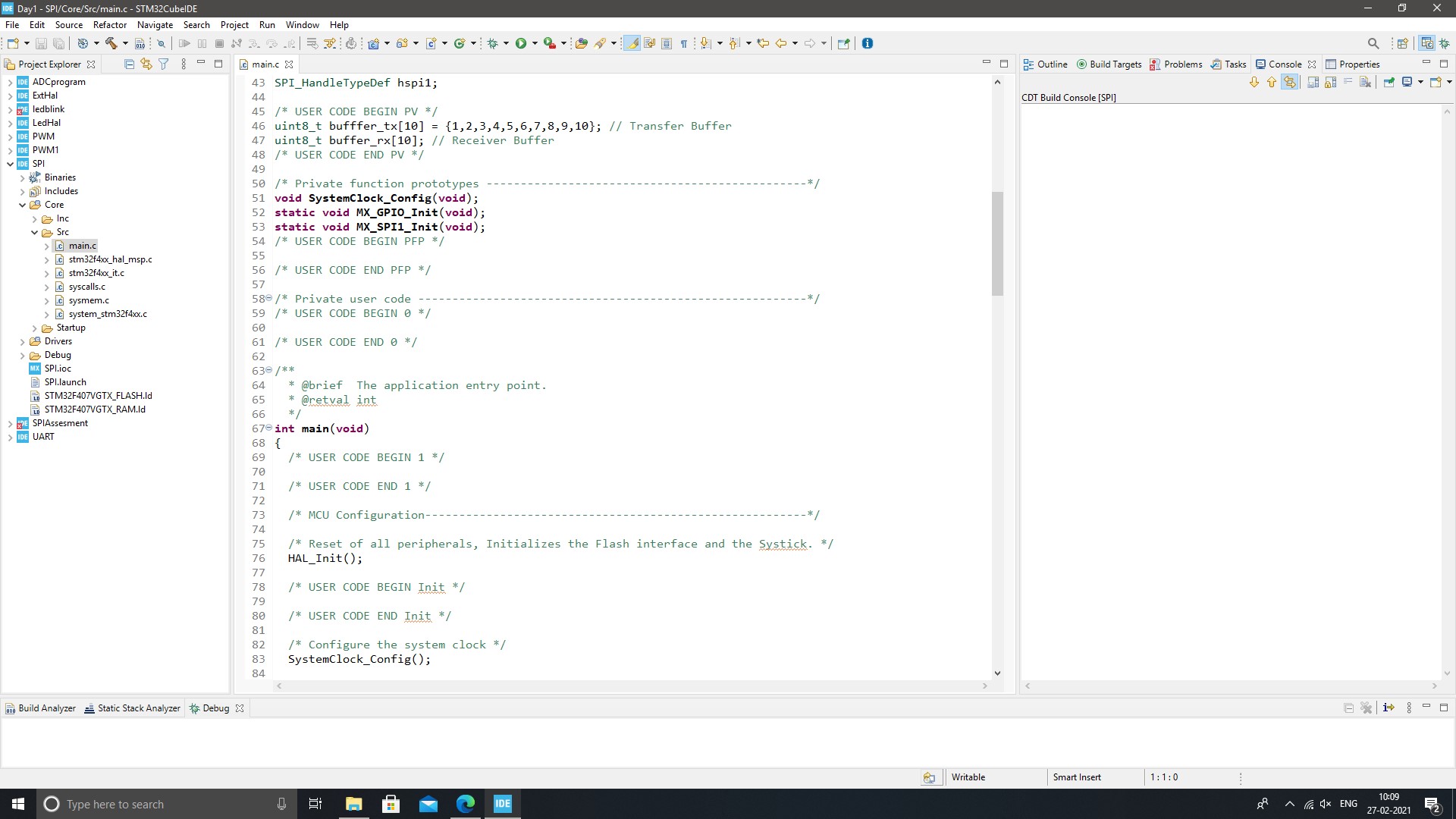
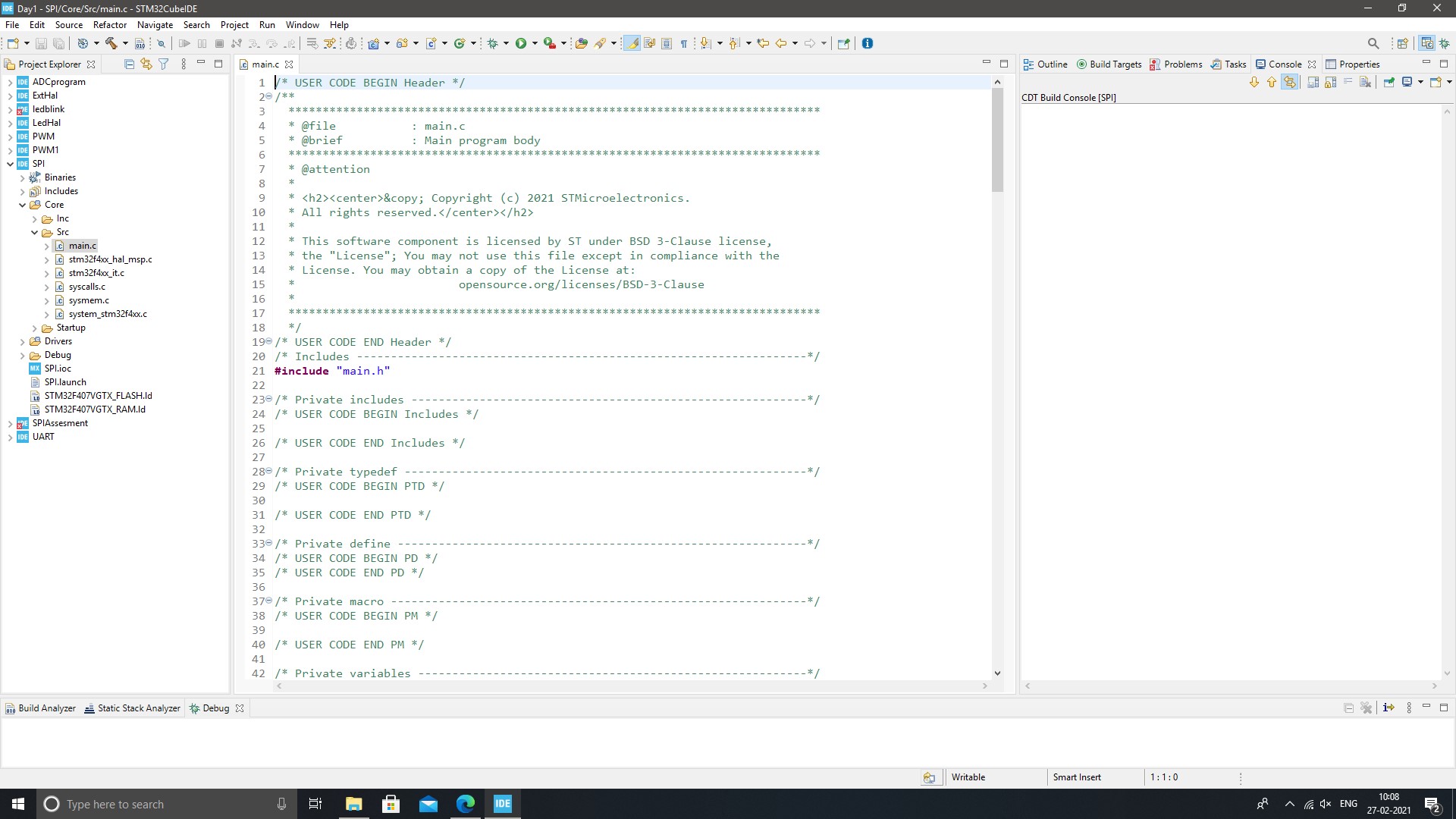


Fig: 2.4.1 SPI Pin configuration



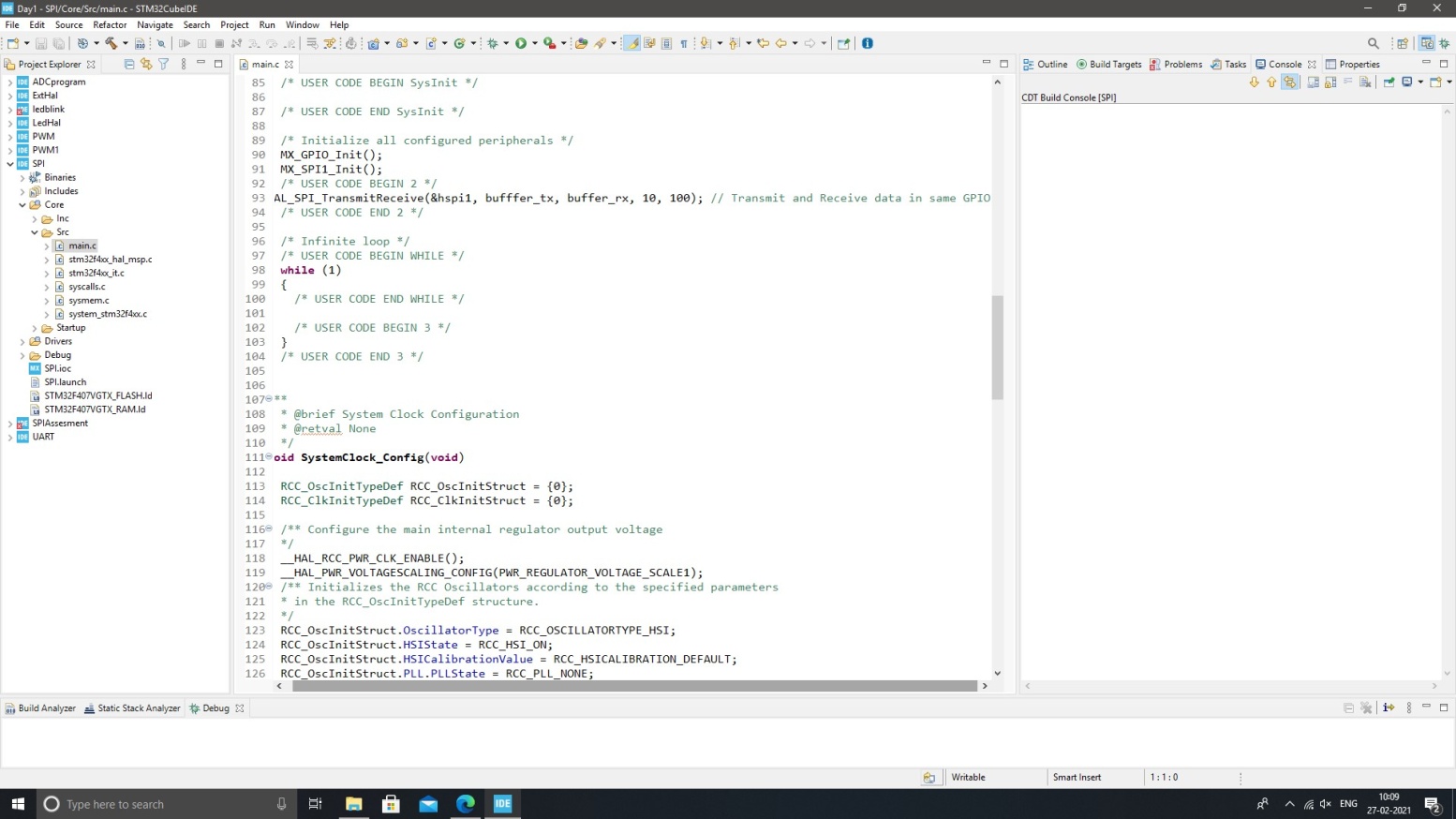


Fig: 2.4.2 SPI configuration code

## 2.5 UART

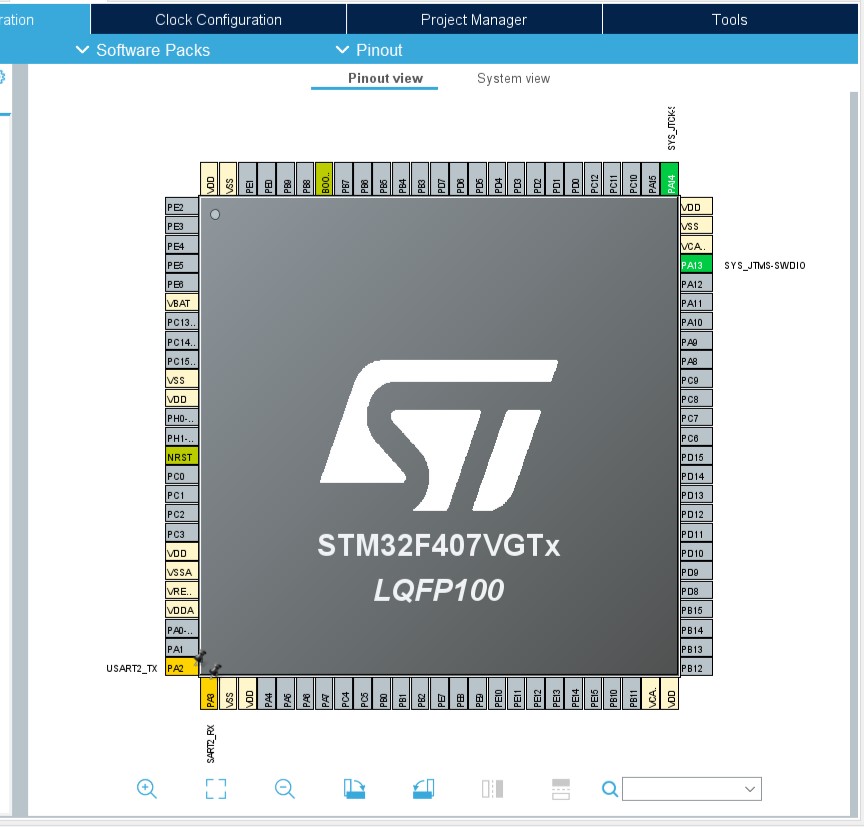


Fig: 2.5.1 UART Pin configuration

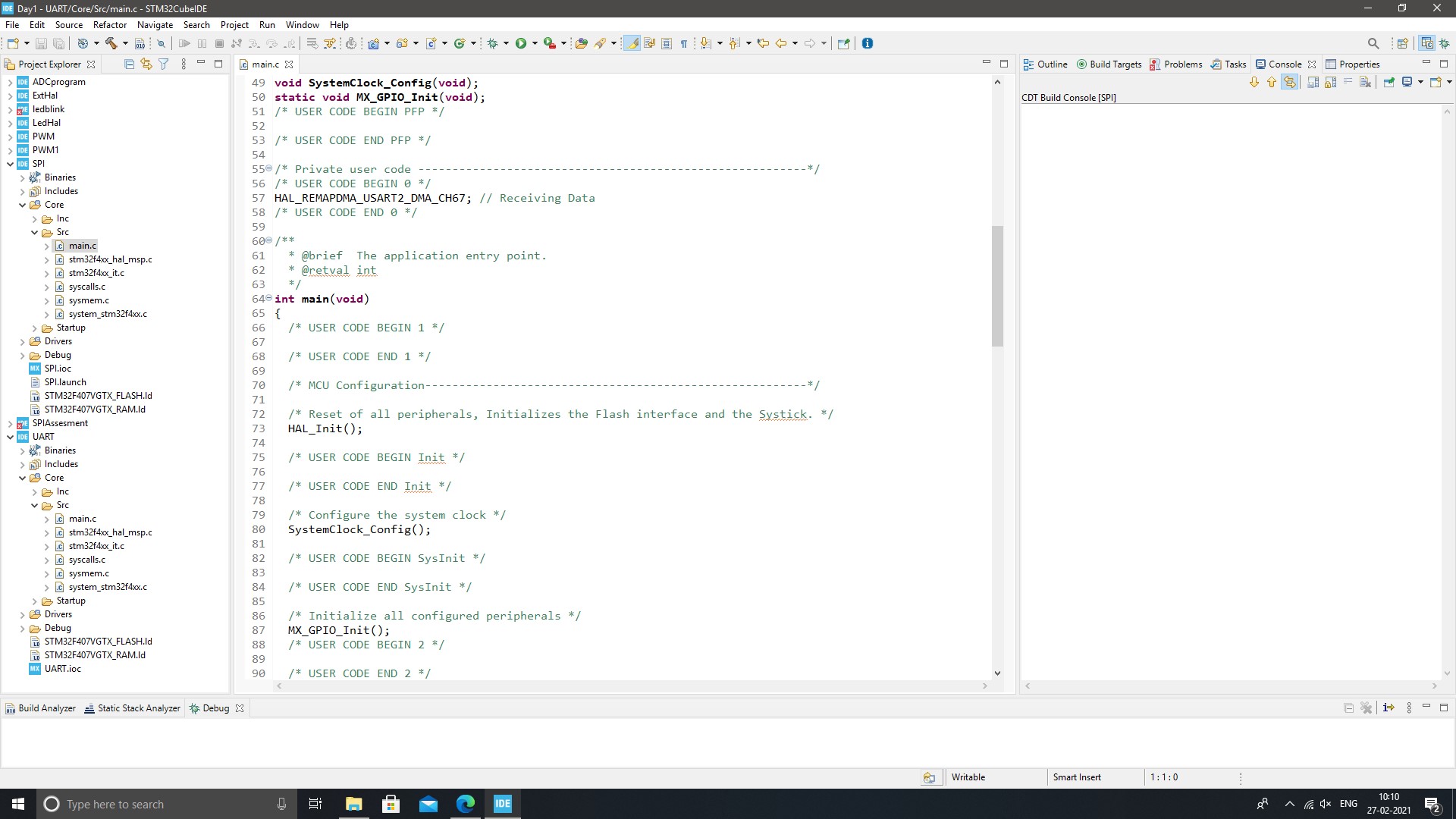
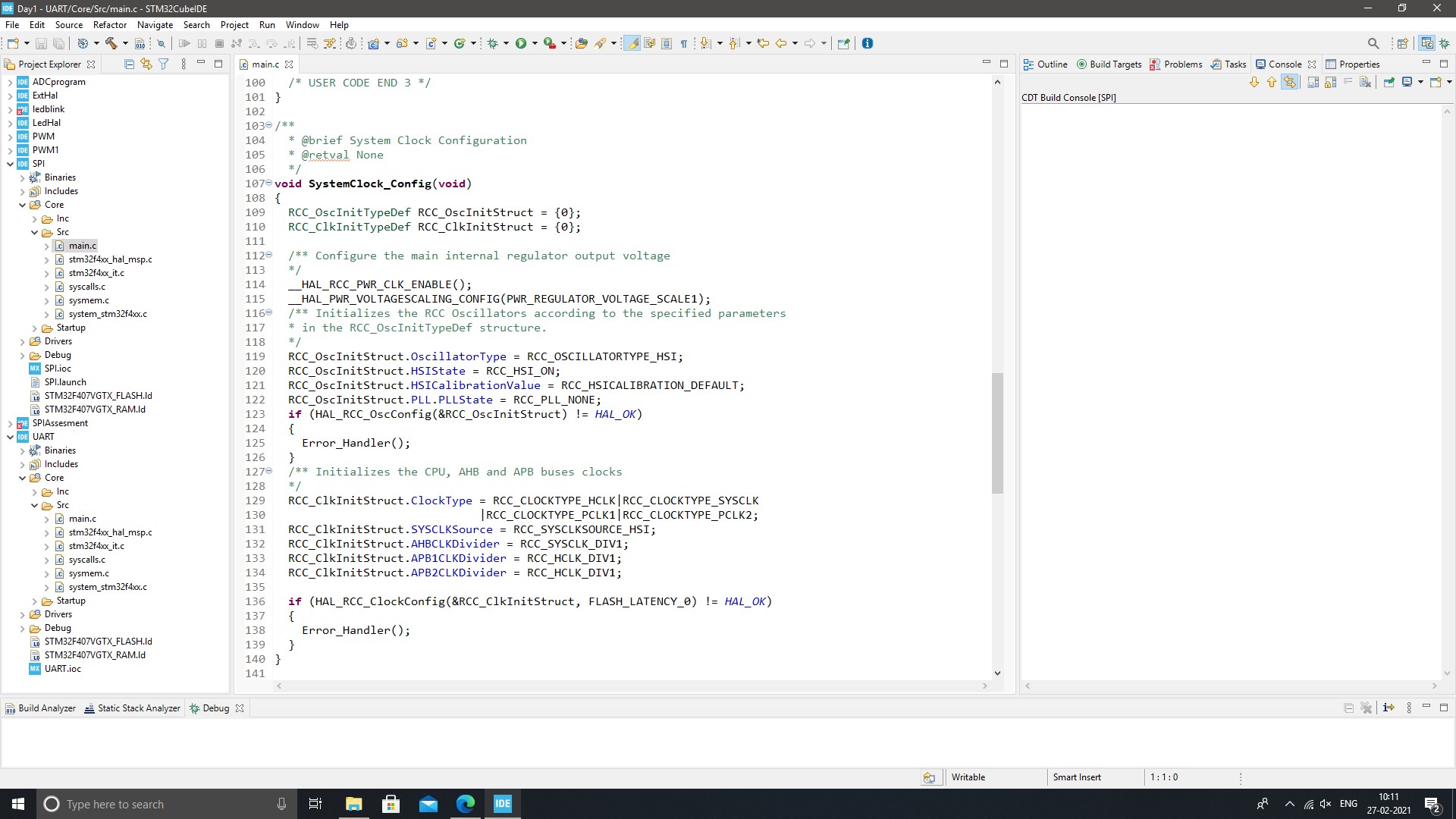


Fig: 2.5.2 UART configuration code

# Activity 3 – PROJECT ON BCM MODULE

BCM module was implemented using STM32f407VG microcontroller featuring 32 bit ARM-cortex M4 with FPU core.

This BCM module have following features:

1. Alarm system
2. Seat control
3. Power mirror
4. Automatic wiper system

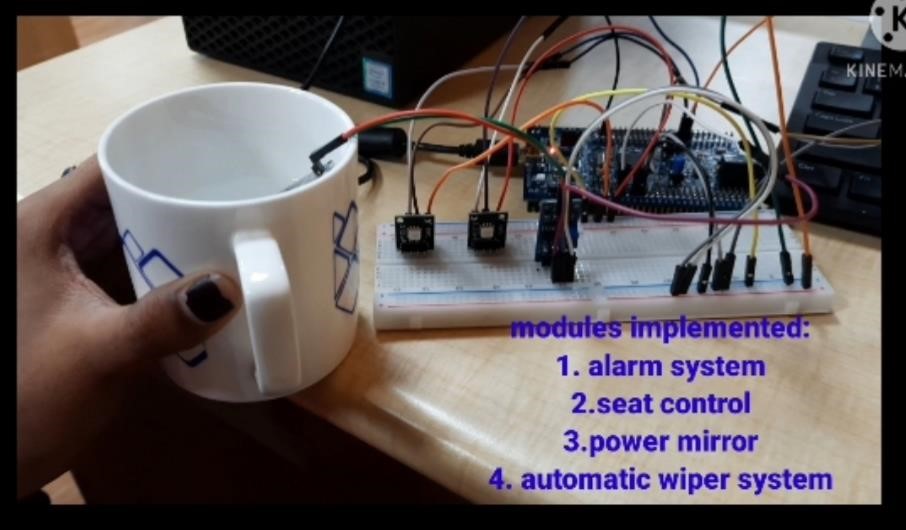


Fig 3.1 Features of BCM module

Below is the pinout and configuration of STM microcontroller used:

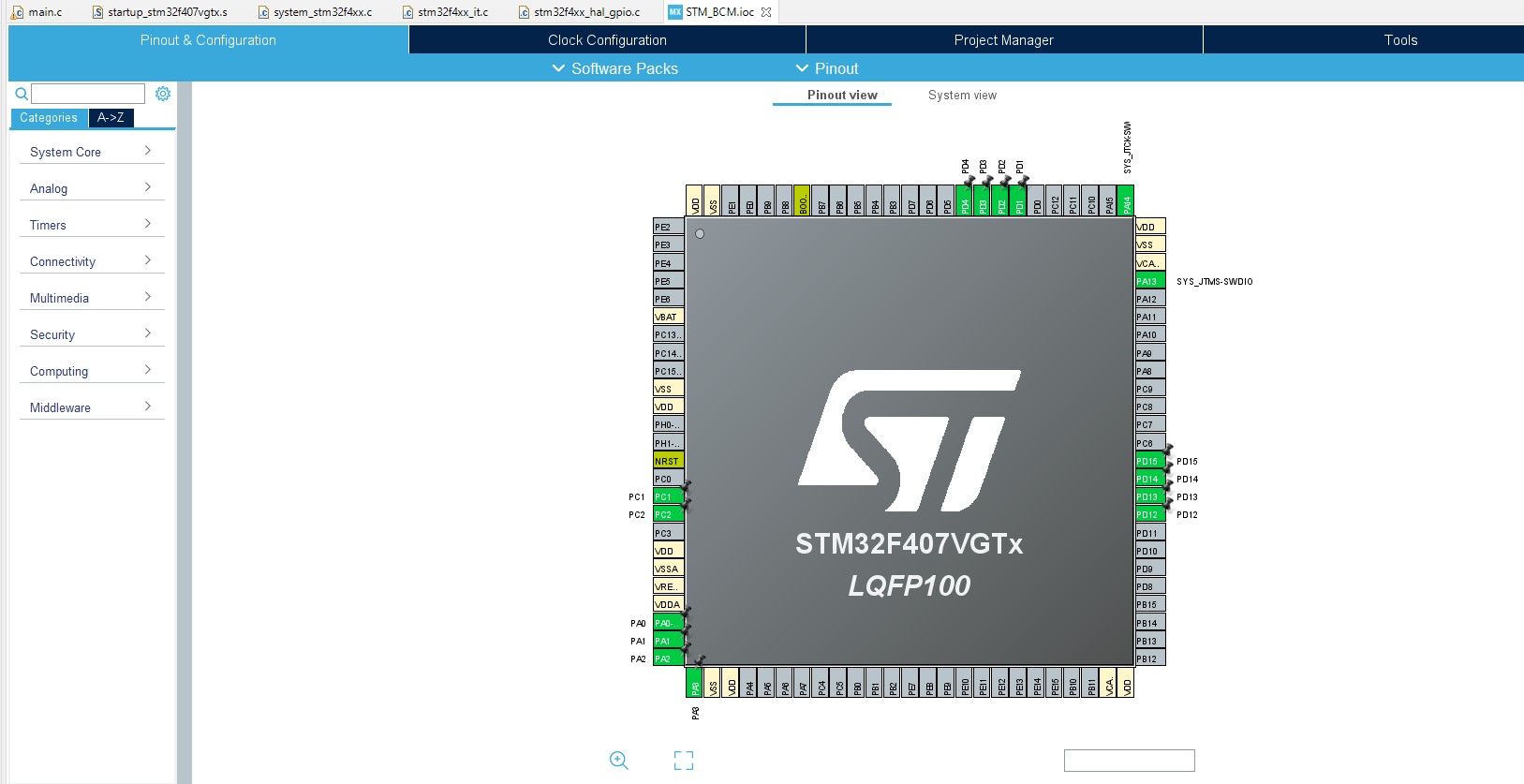


Fig 3.2 Pin configuration

1. **Alarm system:**

PA1: Denotes power status of car (0: OFF, 1: ON)

PA2: Denotes the door status (0: OFF, 1: ON)

PA3: Denotes the door status (0: door closed, 1: door open)

Alarm system uses two variables PA! and PA2 and Dashboard light as output. Door safety system checks whether door is locked or unlocked.

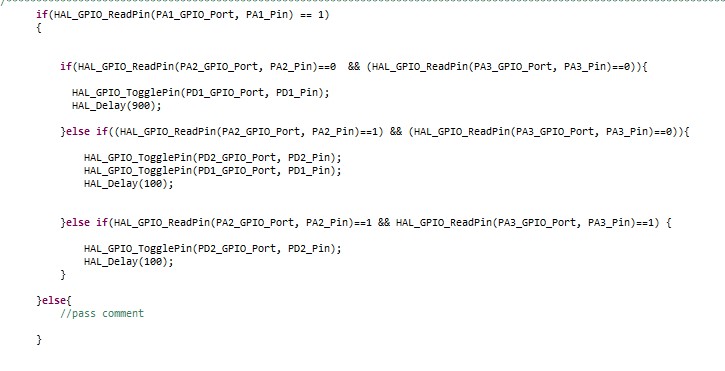


Fig 3.3 Alarm system-code **Implementation:**

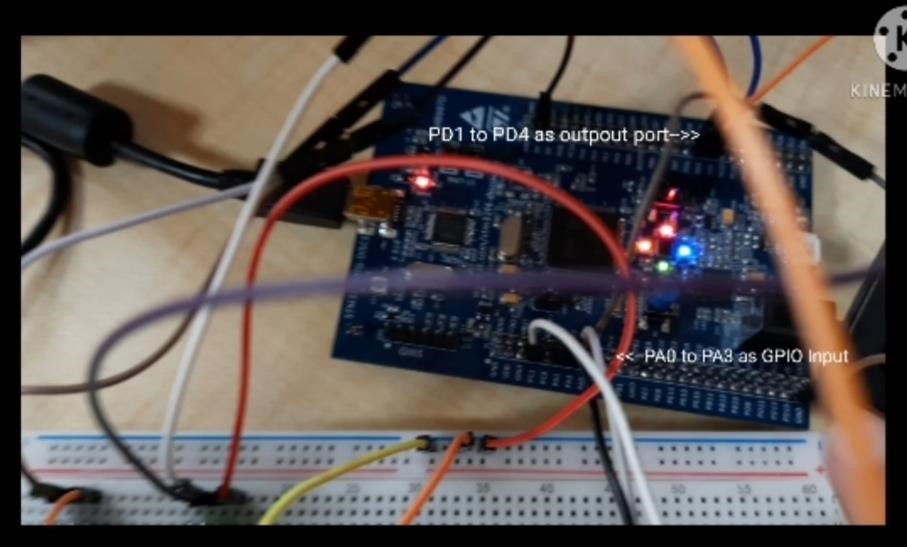


Fig 3.4 Alarm system - Implementation

1. **Seat control:**

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

When the input is 1 LED glows in clockwise direction

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

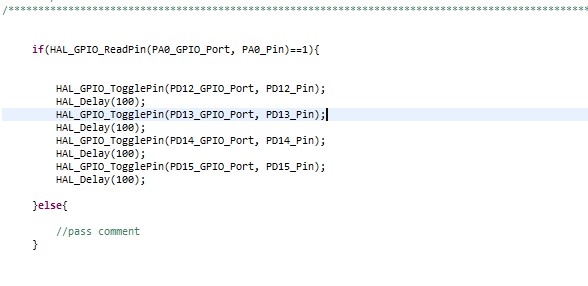


Fig 3.5 Seat control system- code  **Implementation:**



Fig 3.6 Seat control system- Implementation

1. **Power mirror:**

Two of the LEDs glow as a sign of mirrors unfolding when provided with power supply PD12 and PD13 are the output pins PA0 and PA3 are the input pins.

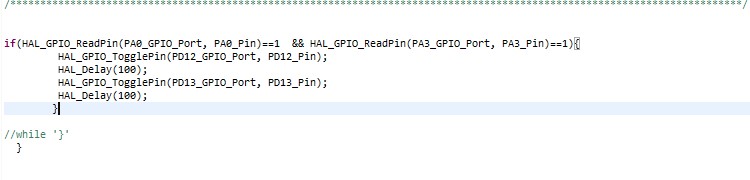


Fig 4.6 Power mirror- code

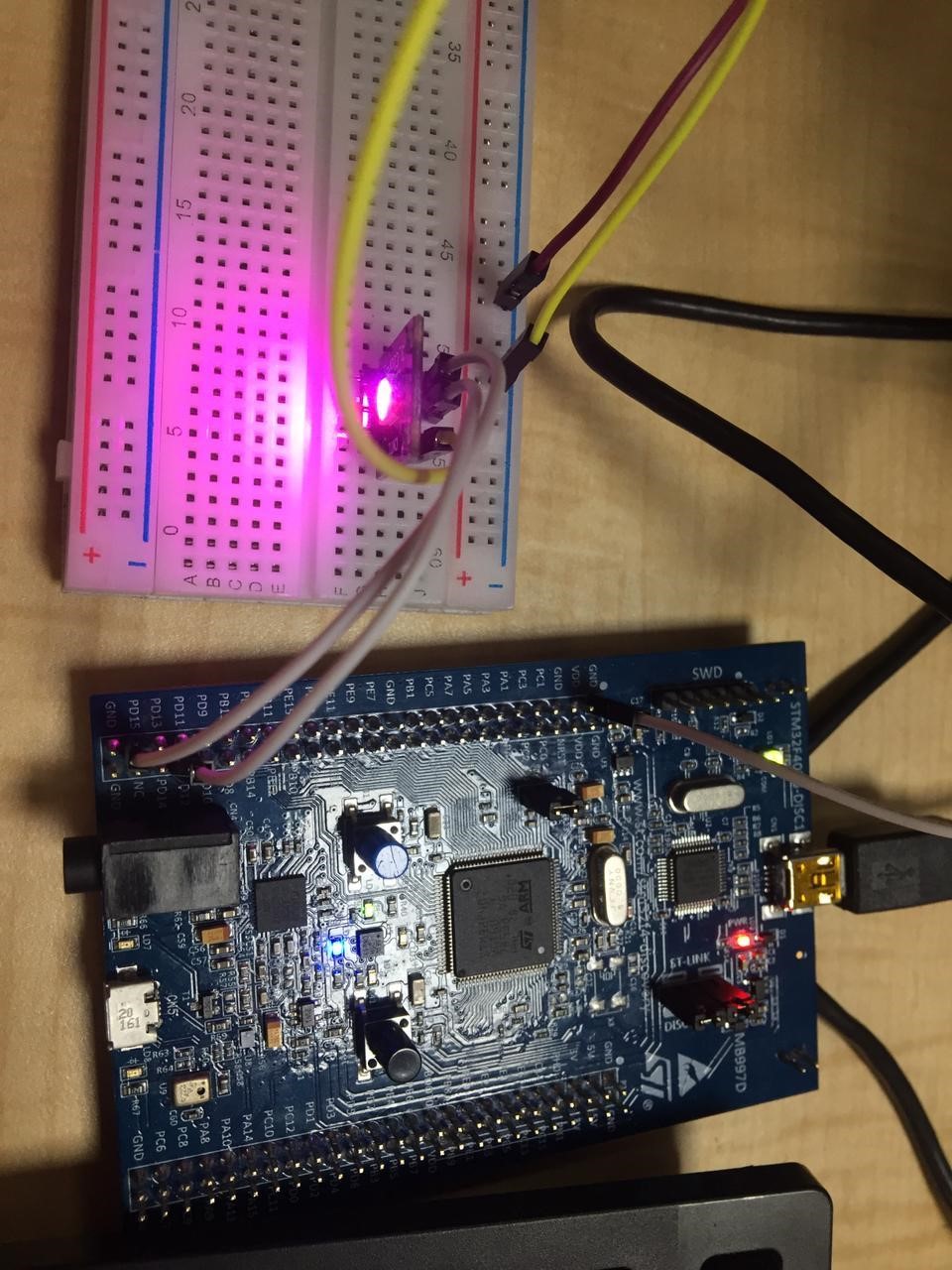


Fig 3.7 Power mirror- Implementation

1. **Automatic wiper system:**

Whenever the humidity sensor senses the presence of moisture over the windshield it sends the signal to turn ON the wipers.

The input was taken from humidity sensor at PC1

The output was given to the external led at pin PD3

The input was taken from humidity sensor at PC1 pin and output is given to the external led at pin PD3.

\* Whenever the humidity sensor senses presence of moisture over the wind shield it sends the signal to

turn ON the wipers.

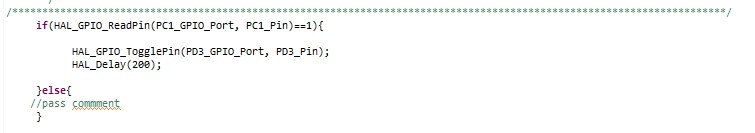


Fig 3.8 Automatic wiper system- code

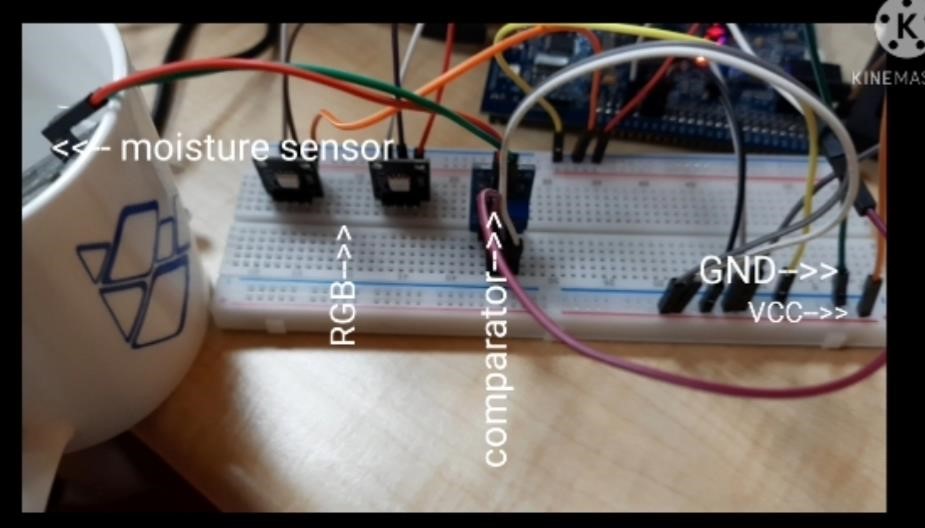


Fig 3.9 Automatic wiper system- Implementation

**Final integrated project:**

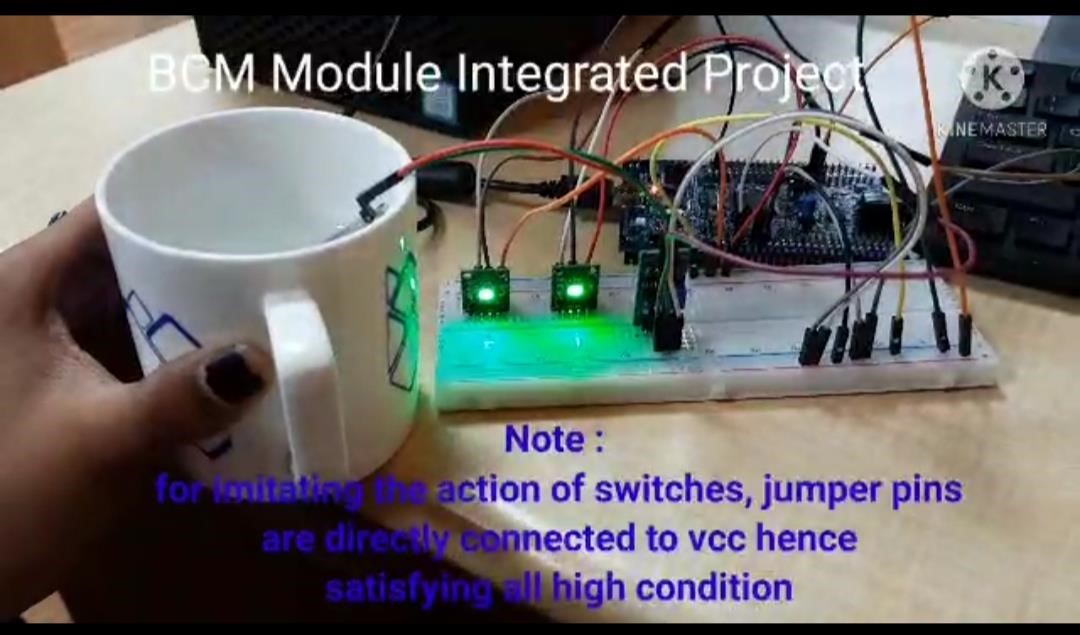


Fig 3.10: Final Integrated BCM