Learning Report – Embedded C

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
	23/02/2021	Srushti Wilson Rangari		Dr.Vivek Kaundal, Bhargav N	First Draft
	25/02/2021	Srushti Wilson Rangari		Dr.Vivek Kaundal, Bhargav N	Second Draft
	27/02/2021	Srushti Wilson Rangari		Dr.Vivek Kaundal, Bhargav N	Final Draft

Contents

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:

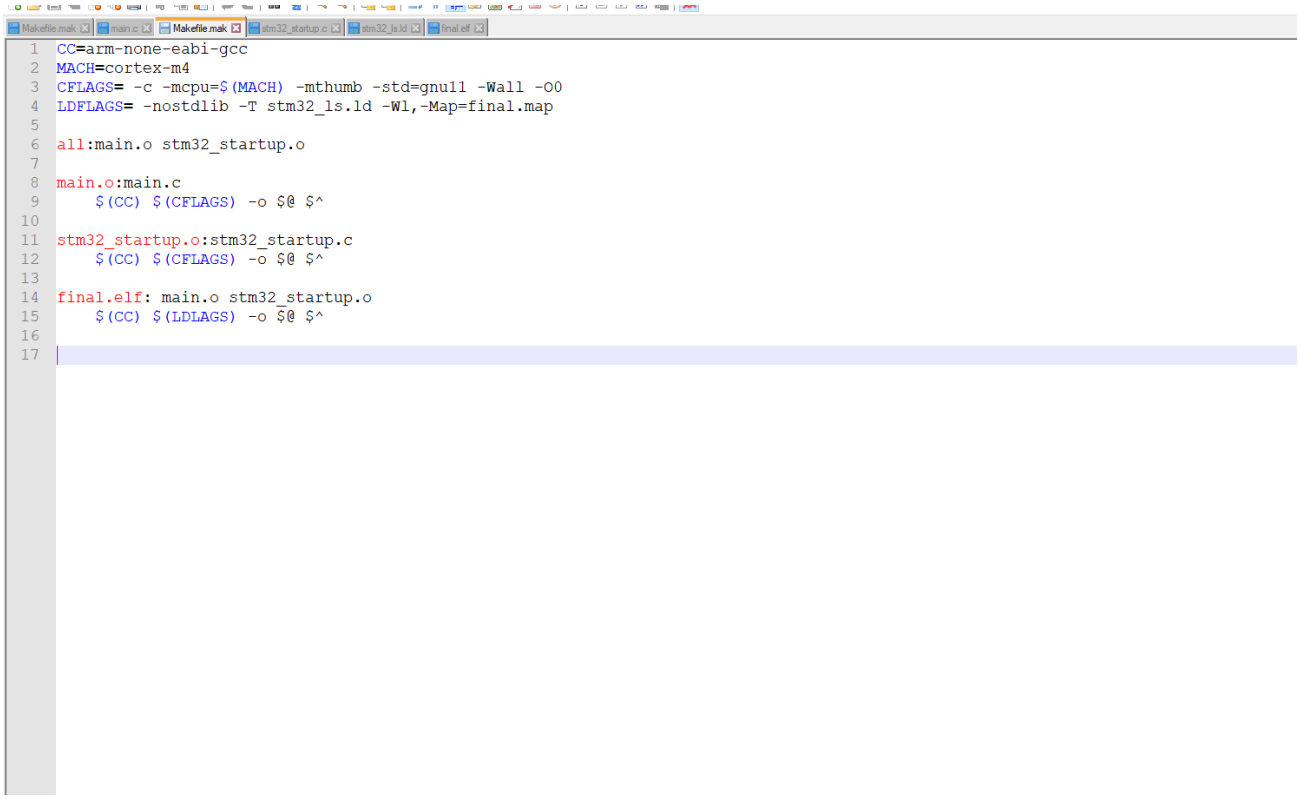


Fig 1.1.1 make file

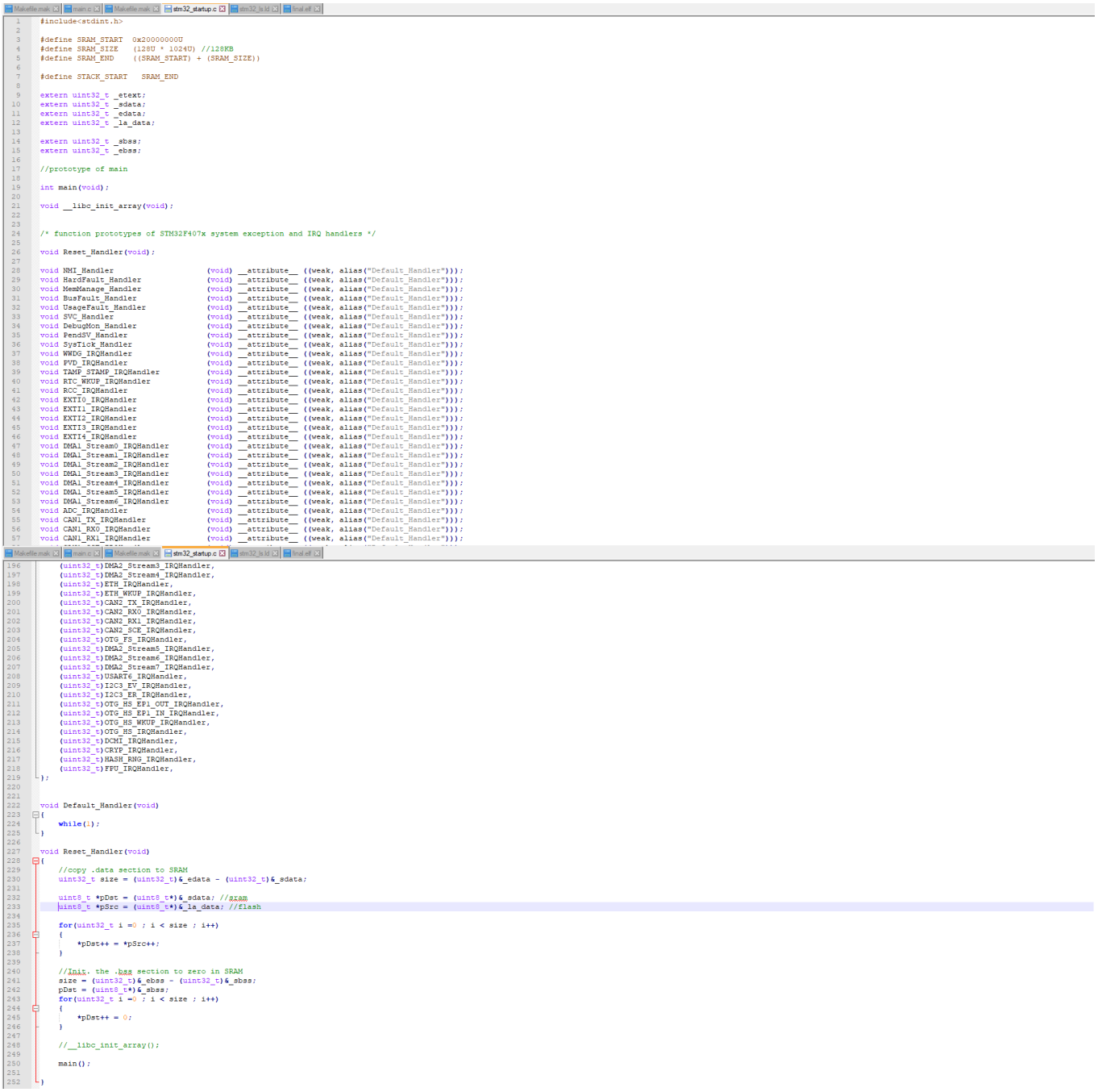
The command to run this make file in the command prompt is:

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()



```c
#include<stdint.h>

#define SRAM_START  0x20000000U
#define SRAM_SIZE   (128U * 1024U) //128KB
#define SRAM_END    ((SRAM_START) + (SRAM_SIZE))

#define STACK_START   SRAM_END

extern uint32_t _etext;
extern uint32_t _sdata;
extern uint32_t _edata;
extern uint32_t _la_data;

extern uint32_t _sbss;
extern uint32_t _ebss;

//prototype of main

int main(void);

void __libc_init_array(void);


/* function prototypes of STM32F407x system exception and IRQ handlers */

void Reset_Handler(void);

void NMI_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void HardFault_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void MemManage_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void BusFault_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void UsageFault_Handler             (void) __attribute__ ((weak, alias("Default_Handler")));
void SVC_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void DebugMon_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void PendSV_Handler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void SysTick_Handler                (void) __attribute__ ((weak, alias("Default_Handler")));
void WWDG_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void PVD_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void TAMP_STAMP_IRQHandler          (void) __attribute__ ((weak, alias("Default_Handler")));
void RTC_WKUP_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void RCC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI0_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI1_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI2_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI3_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI4_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream0_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream1_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream2_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream3_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream4_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream5_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream6_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void ADC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_TX_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_RX0_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_RX1_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
```

```c
    (uint32_t)DMA2_Stream3_IRQHandler,
    (uint32_t)DMA2_Stream4_IRQHandler,
    (uint32_t)ETH_IRQHandler,
    (uint32_t)ETH_WKUP_IRQHandler,
    (uint32_t)CAN2_TX_IRQHandler,
    (uint32_t)CAN2_RX0_IRQHandler,
    (uint32_t)CAN2_RX1_IRQHandler,
    (uint32_t)CAN2_SCE_IRQHandler,
    (uint32_t)OTG_FS_IRQHandler,
    (uint32_t)DMA2_Stream5_IRQHandler,
    (uint32_t)DMA2_Stream6_IRQHandler,
    (uint32_t)DMA2_Stream7_IRQHandler,
    (uint32_t)USART6_IRQHandler,
    (uint32_t)I2C3_EV_IRQHandler,
    (uint32_t)I2C3_ER_IRQHandler,
    (uint32_t)OTG_HS_EP1_OUT_IRQHandler,
    (uint32_t)OTG_HS_EP1_IN_IRQHandler,
    (uint32_t)OTG_HS_WKUP_IRQHandler,
    (uint32_t)OTG_HS_IRQHandler,
    (uint32_t)DCMI_IRQHandler,
    (uint32_t)CRYP_IRQHandler,
    (uint32_t)HASH_RNG_IRQHandler,
    (uint32_t)FPU_IRQHandler,
};


void Default_Handler(void)
{
    while(1);
}

void Reset_Handler(void)
{
    //copy .data section to SRAM
    uint32_t size = (uint32_t)&_edata - (uint32_t)&_sdata;

    uint8_t *pDst = (uint8_t*)&_sdata; //sram
    uint8_t *pSrc = (uint8_t*)&_la_data; //flash

    for(uint32_t i =0 ; i < size ; i++)
    {
        *pDst++ = *pSrc++;
    }

    //Init. the .bss section to zero in SRAM
    size = (uint32_t)&_ebss - (uint32_t)&_sbss;
    pDst = (uint8_t*)&_sbss;
    for(uint32_t i =0 ; i < size ; i++)
    {
        *pDst++ = 0;
    }

    //__libc_init_array();

    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:

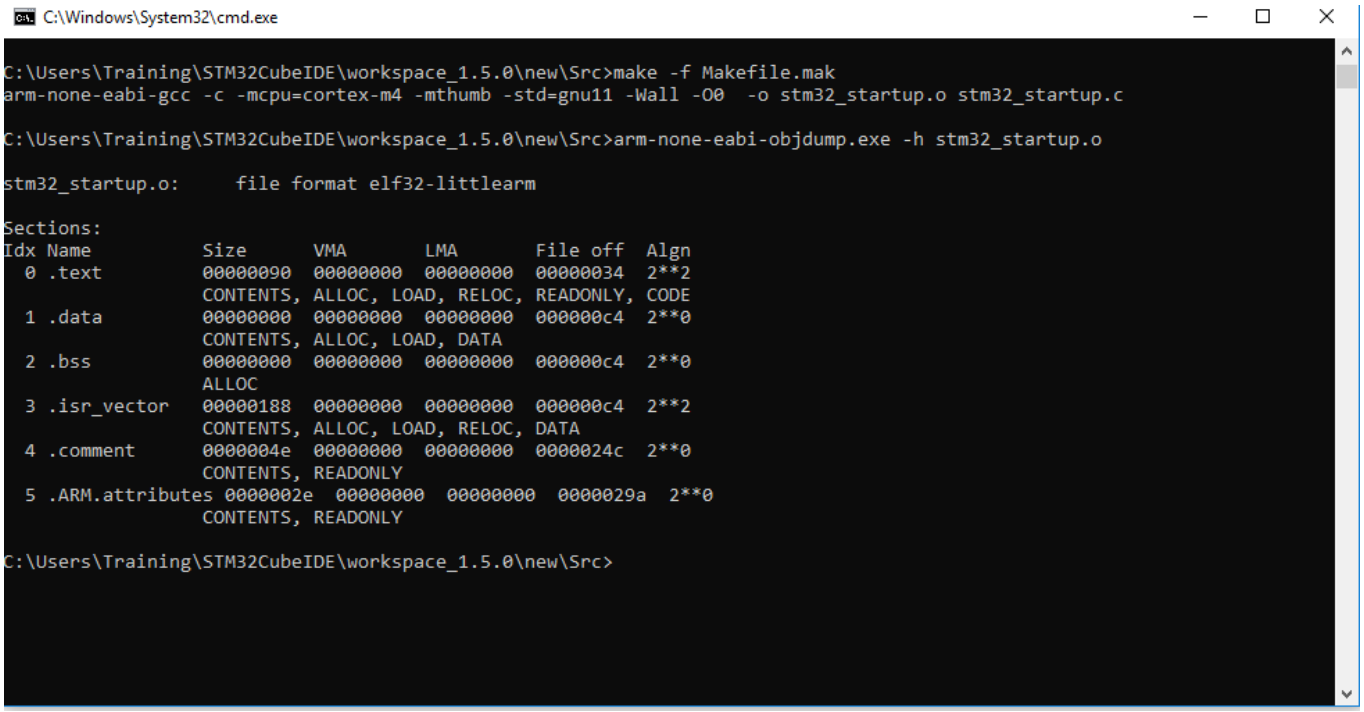
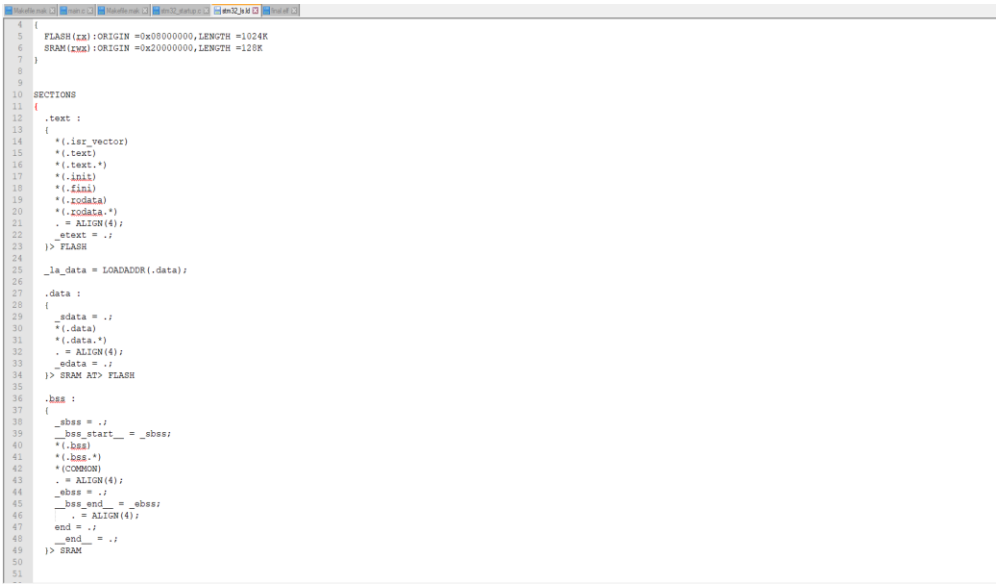


Fig 1.2.2: Startup command

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)



```
{
  FLASH(rx):ORIGIN =0x08000000,LENGTH =1024K
  SRAM(rwx):ORIGIN =0x20000000,LENGTH =128K
}


SECTIONS
{
  .text :
  {
    *(.isr_vector)
    *(.text)
    *(.text.*)
    *(.init)
    *(.fini)
    *(.rodata)
    *(.rodata.*)
    . = ALIGN(4);
    _etext = .;
  }> FLASH

  _la_data = LOADADDR(.data);

  .data :
  {
    _sdata = .;
    *(.data)
    *(.data.*)
    . = ALIGN(4);
    _edata = .;
  }> SRAM AT> FLASH

  .bss :
  {
    _sbss = .;
    __bss_start__ = _sbss;
    *(.bss)
    *(.bss.*)
    *(COMMON)
    . = ALIGN(4);
    _ebss = .;
    __bss_end__ = _ebss;
     . = ALIGN(4);
    end = .;
    __end__ = .;
  }> SRAM
```

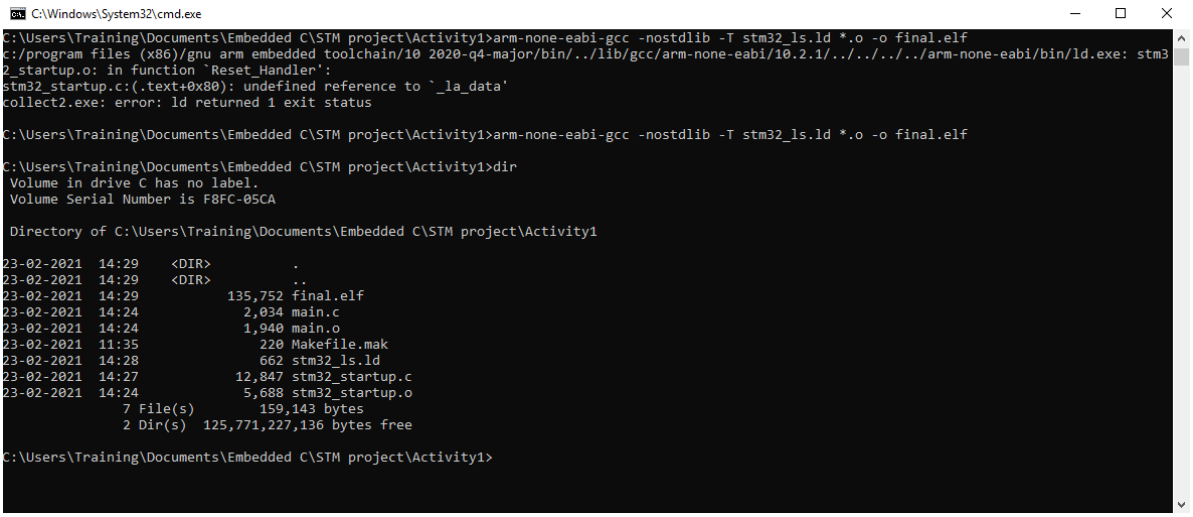


Fig 1.3.1 command to generate final.elf file

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.

2.1 GPIO:

Toggling LED at pin PD12 at GREEN_LED_GPIO_PORT. Serial wire is enabled at pin PA13.

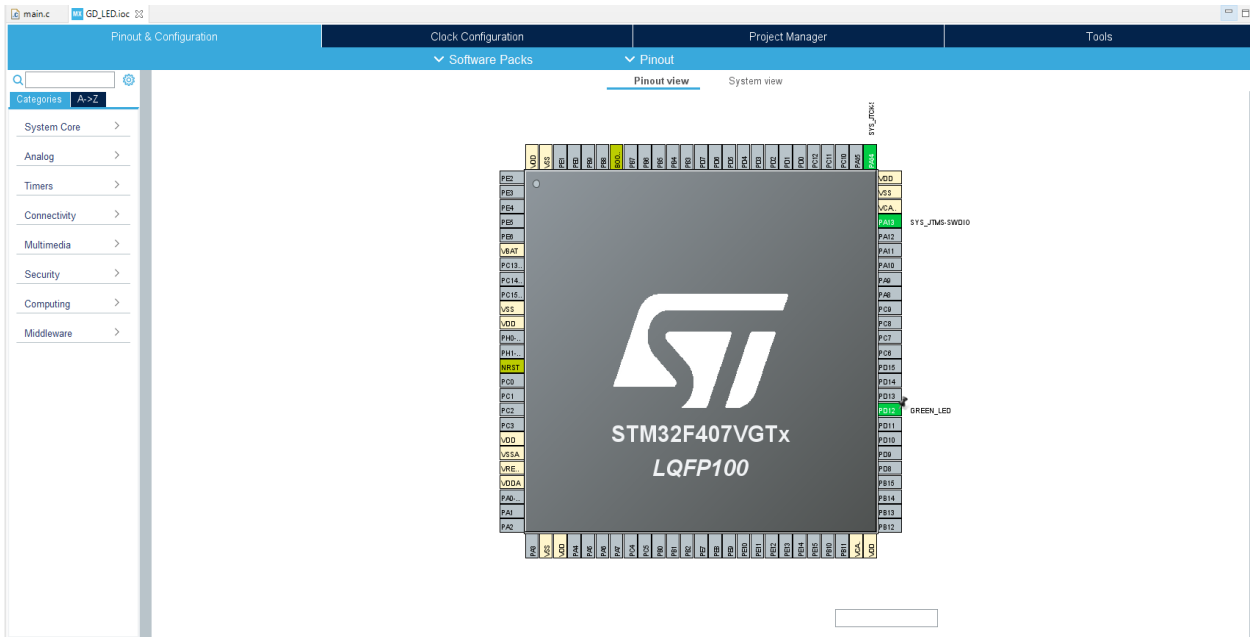


Fig: 2.1.1 GPIO pin configuration

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
    HAL_Delay(500);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

Fig: 2.1.2 GPIO configuration code

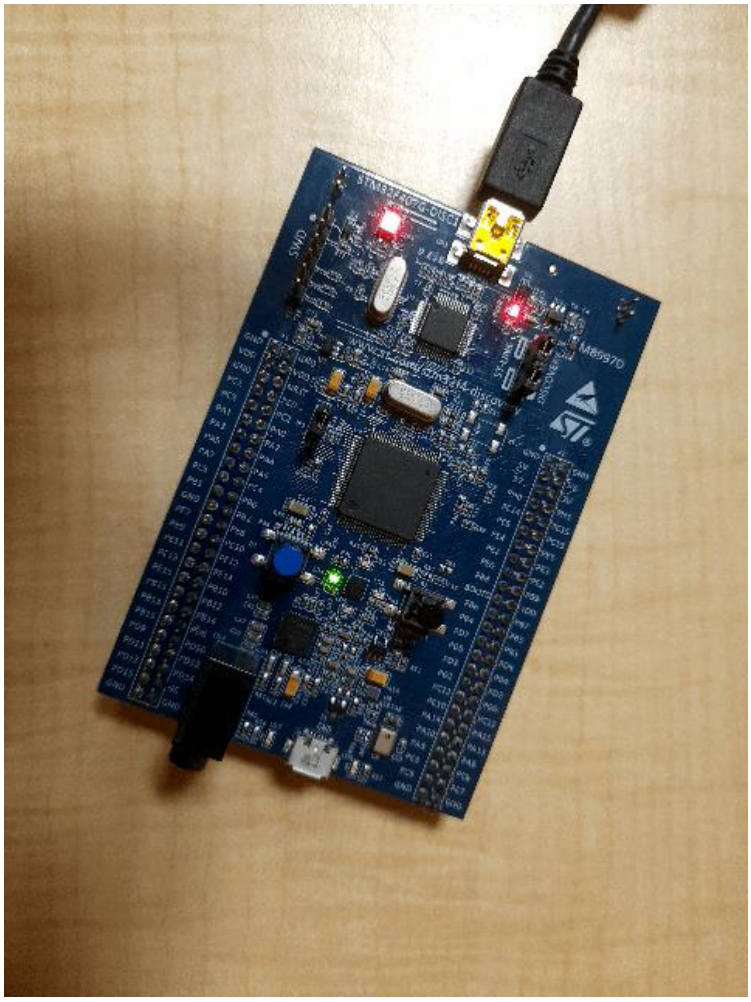


Fig: 2.1.2 LED toggling

2.2 EXTI:

Blue button at PA0 works as an external interrupt.
When the blue button is pressed the Green LED at pin PD12 toggles.

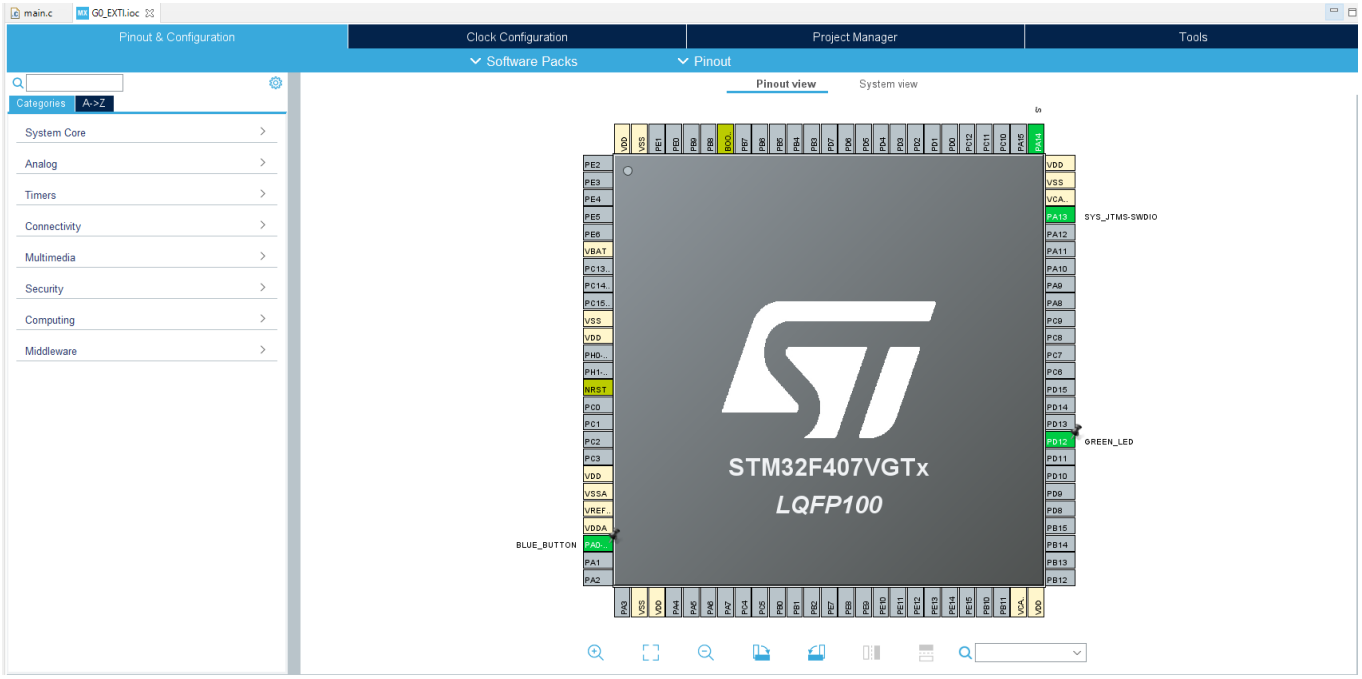


Fig: 2.2.1 EXTI pin configuration

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.

main.c | G0_EXTI.ioc

```
/* USER CODE BEGIN PV */
uint8_t flag = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
      if(1==flag){
              HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
              HAL_Delay(500);
              flag = 0;
          }
    /* USER CODE BEGIN 3 */
  }
```

Fig: 2.2.1 EXTI configuration code

2.3 ADC

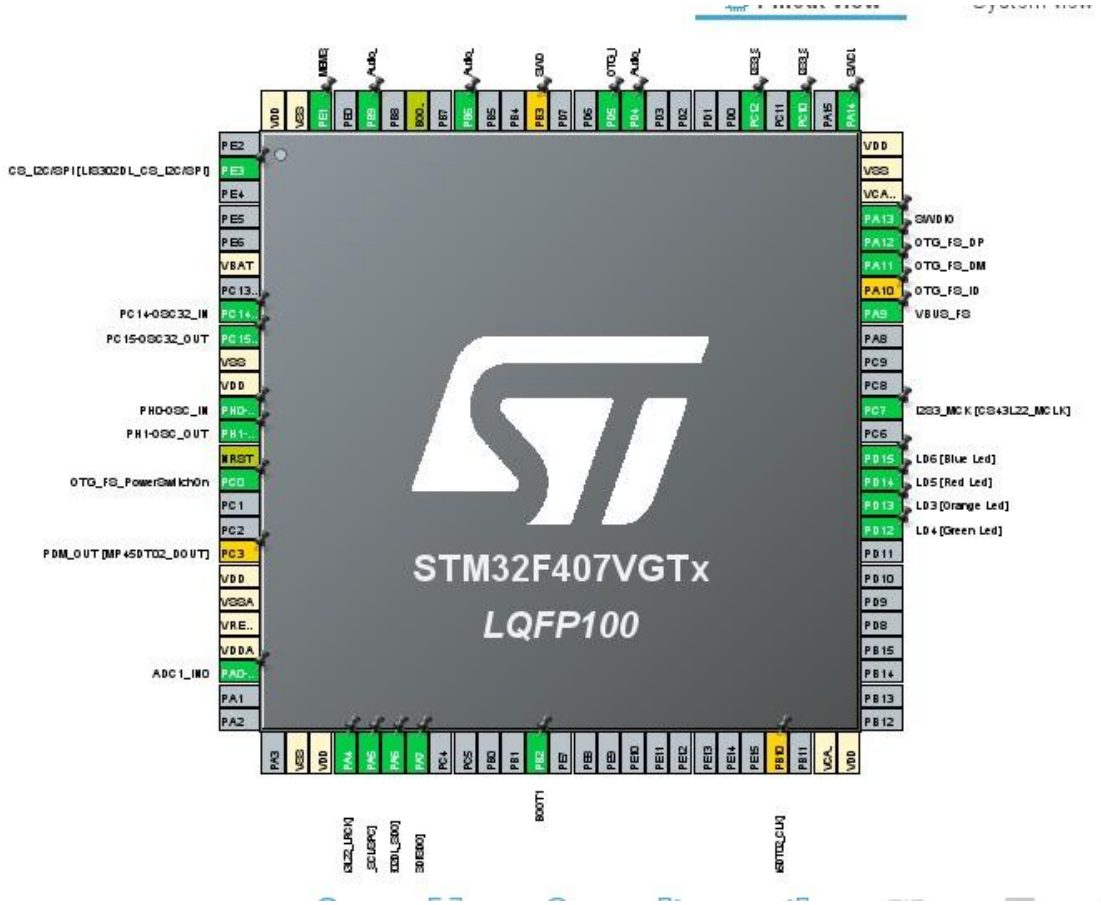


Fig: 2.3.1 ADC pin configuration

*main.c

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"


ADC_HandleTypeDef hadc1;

I2C_HandleTypeDef hi2c1;

I2S_HandleTypeDef hi2s3;

SPI_HandleTypeDef hspi1;


/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S3_Init(void);
static void MX_SPI1_Init(void);
static void MX_ADC1_Init(void);
void MX_USB_HOST_Process(void);


uint32_t adc_value;

int main(void)
{

  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
```

*main.c

```

  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_USB_HOST_Init();
  MX_ADC1_Init();

  while (1)
  {

    MX_USB_HOST_Process();


    HAL_ADC_Start(&hadc1);


    if(HAL_ADC_PollForConversion(&hadc1, 5)== HAL_OK){

        adc_value= HAL_ADC_GetValue(&hadc1);

    }


    HAL_ADC_Stop(&hadc1);
    HAL_Delay(100);

  }

}


void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
```

Fig: 2.3.2 ADC configuration code

2.4 SPI

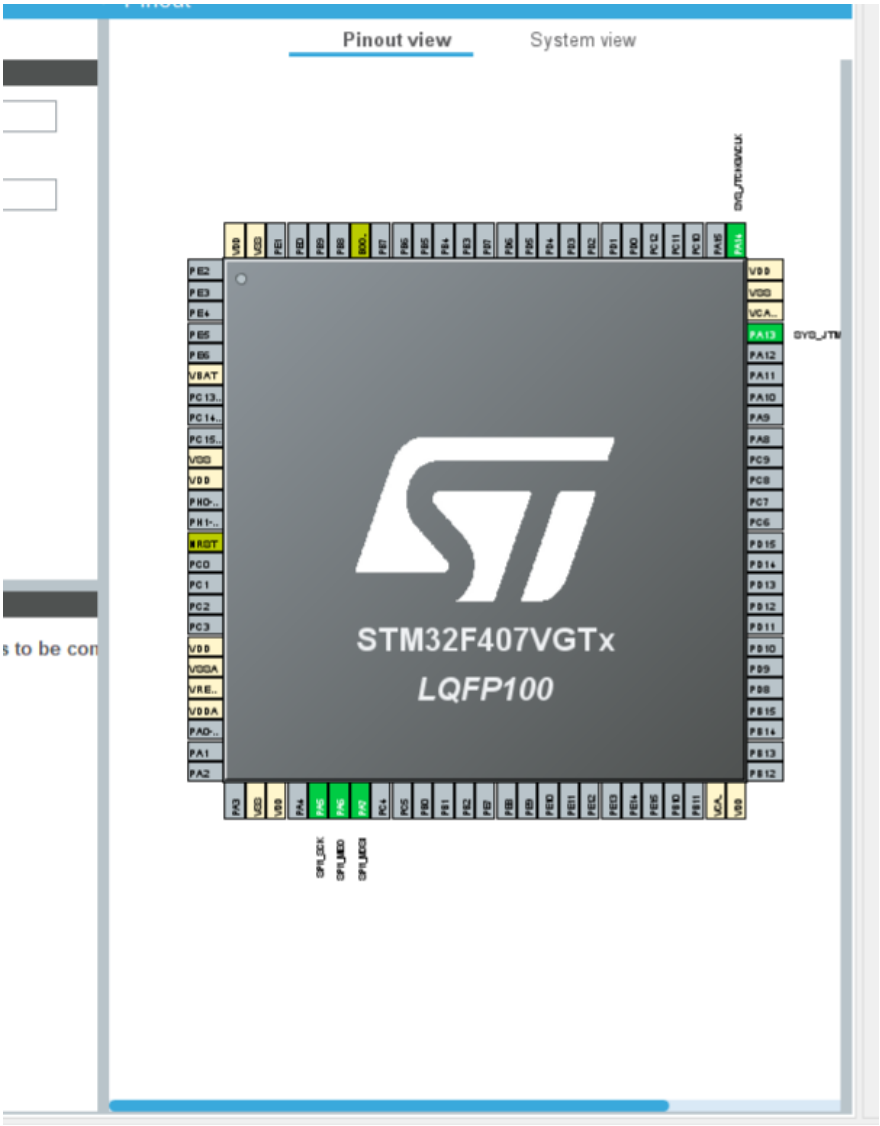
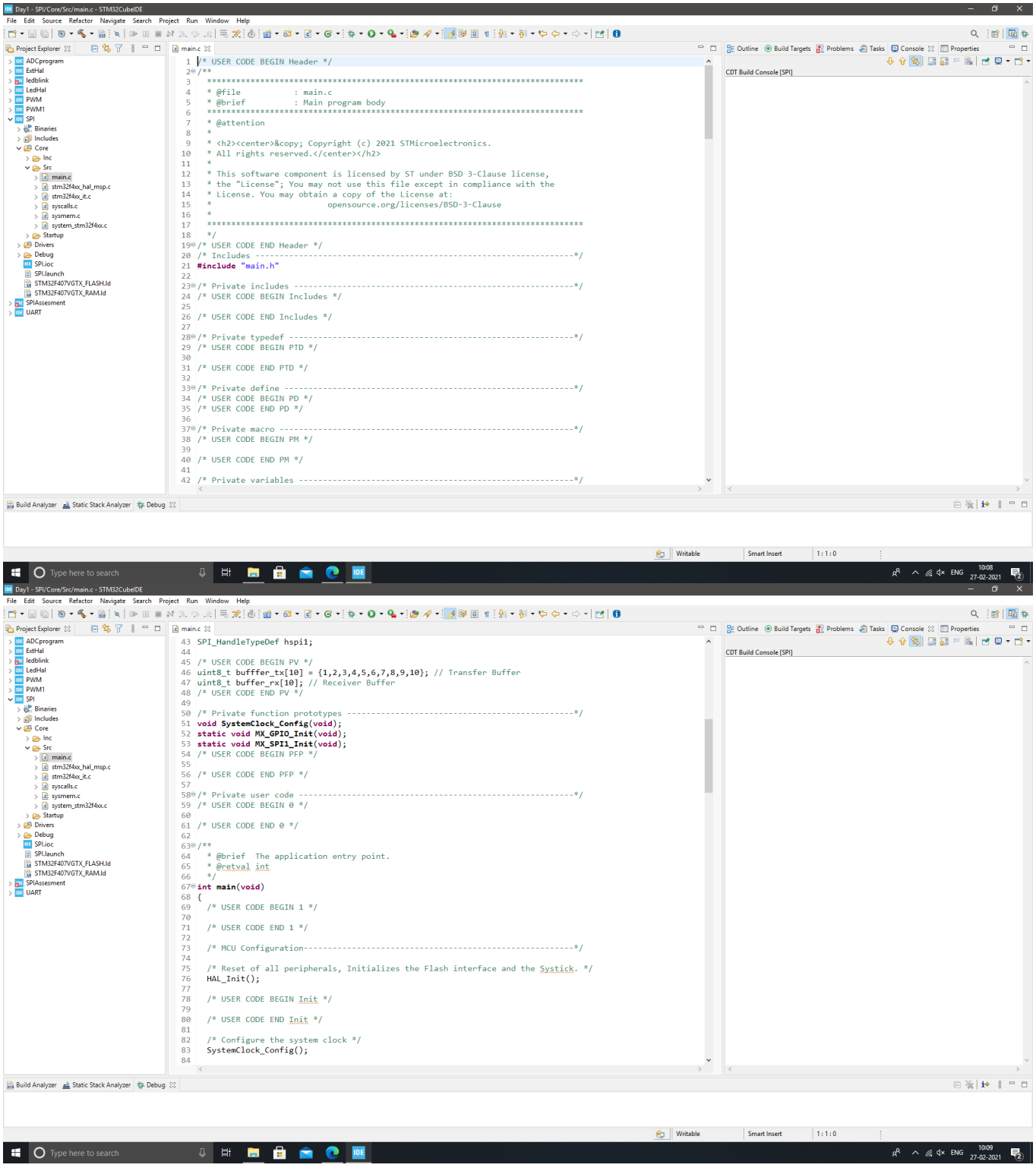


Fig: 2.4.1 SPI Pin configuration


Day1 - SPI/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
main.c
Outline Build Targets Problems Tasks Console Properties
CDT Build Console [SPI]
ADCprogram
ExtHal
ledblink
LedHal
PWM
PWM1
SPI
Binaries
Includes
Core
Inc
Src
main.c
stm32f4xx_hal_msp.c
stm32f4xx_it.c
syscalls.c
sysmem.c
system_stm32f4xx.c
Startup
Drivers
Debug
SPI.ioc
SPI.launch
STM32F407VGTX_FLASH.ld
STM32F407VGTX_RAM.ld
SPIAssesment
UART
/* USER CODE BEGIN Header */
/**
* @file : main.c
* @brief : Main program body
* @attention
* <h2> © Copyright (c) 2021 STMicroelectronics.
* All rights reserved. </h2>
* This software component is licensed by ST under BSD 3-Clause license,
* the "License"; You may not use this file except in compliance with the
* License. You may obtain a copy of the License at:
* opensource.org/licenses/BSD-3-Clause
*/
/* USER CODE END Header */
/* Includes */
#include "main.h"
/* Private includes */
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef */
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define */
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro */
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables */
Build Analyzer Static Stack Analyzer Debug
Writable Smart Insert 1:1:0
Type here to search
ENG 10:08 27-02-2021
SPI_HandleTypeDef hspi1;
/* USER CODE BEGIN PV */
uint8_t buffer_tx[10] = {1,2,3,4,5,6,7,8,9,10}; // Transfer Buffer
uint8_t buffer_rx[10]; // Receiver Buffer
/* USER CODE END PV */
/* Private function prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
* @brief The application entry point.
* @retval int
*/
int main(void)
{
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
/* MCU Configuration */
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();
/* USER CODE BEGIN Init */
/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();
ENG 10:09 27-02-2021

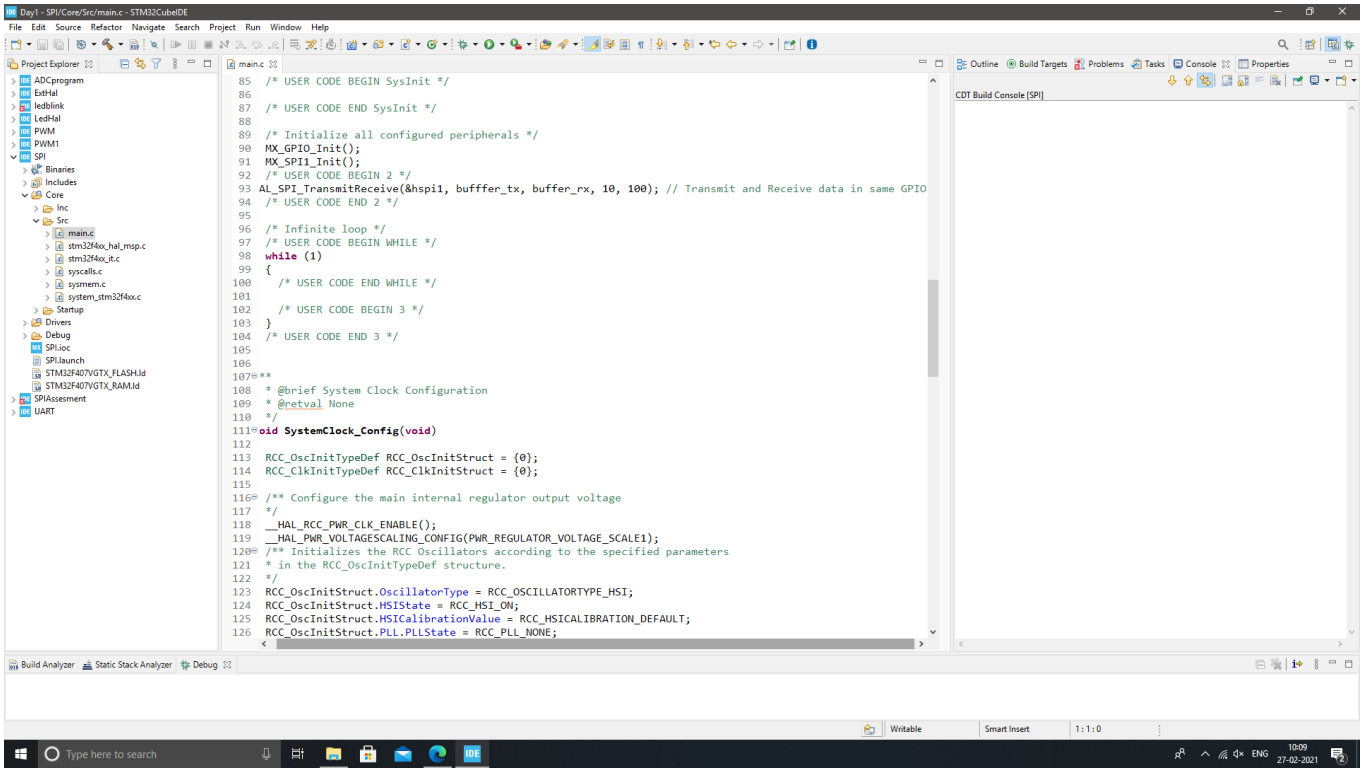


Fig: 2.4.2 SPI configuration code

2.5 UART

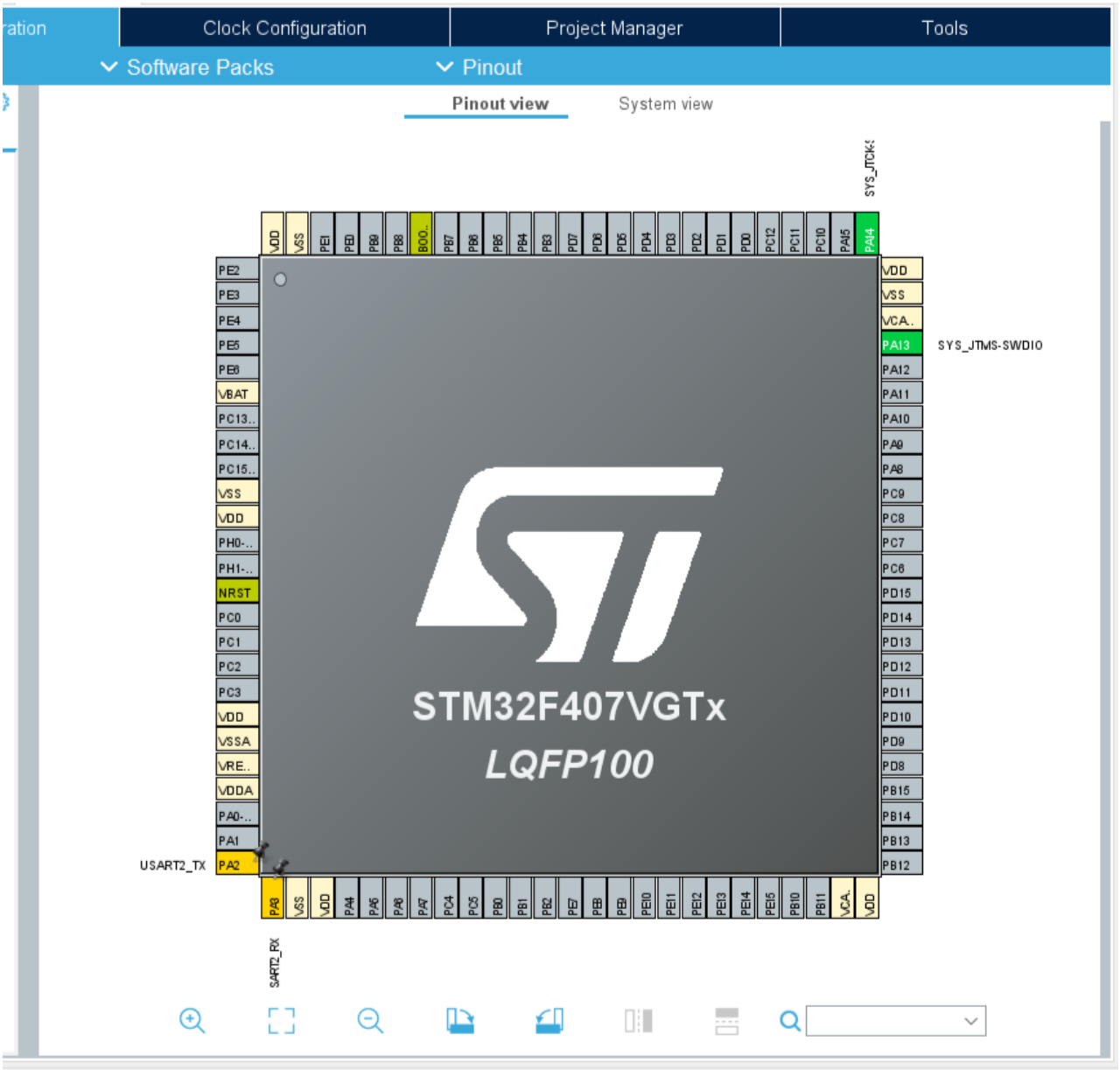
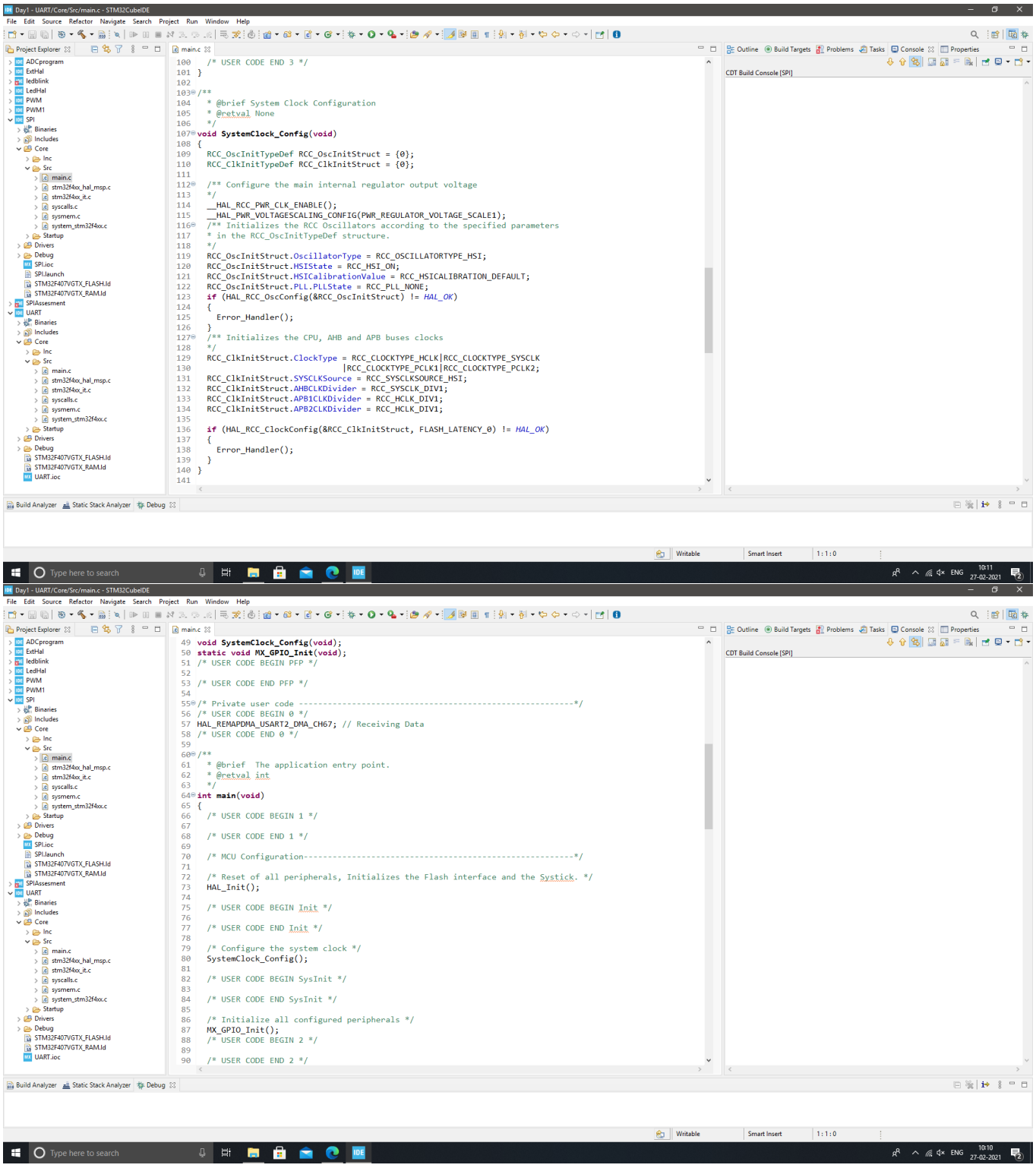


Fig: 2.5.1 UART Pin configuration




```c
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```c
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
HAL_REMAPDMA_USART2_DMA_CH67; // Receiving Data
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
```

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

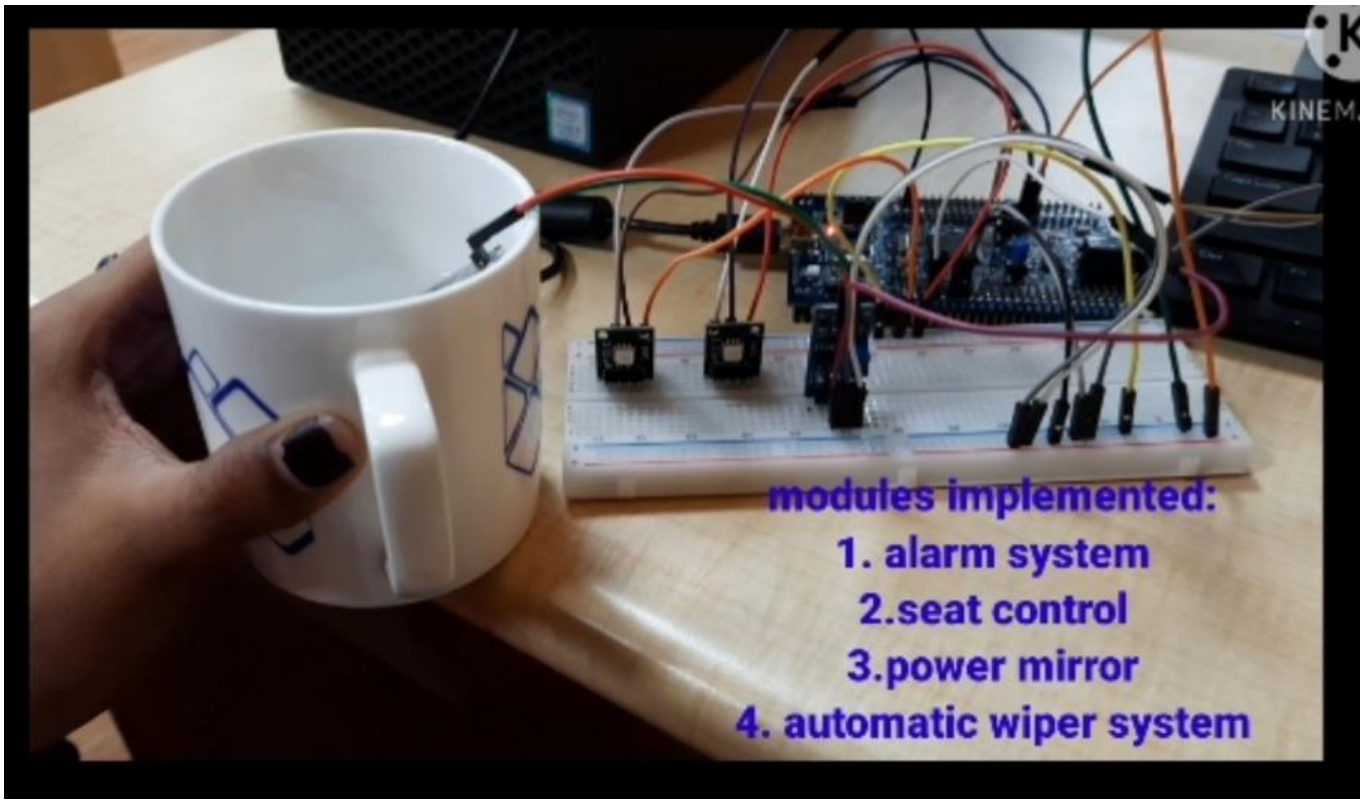


Fig 3.1 Features of BCM module

Below is the pinout and configuration of STM microcontroller used:

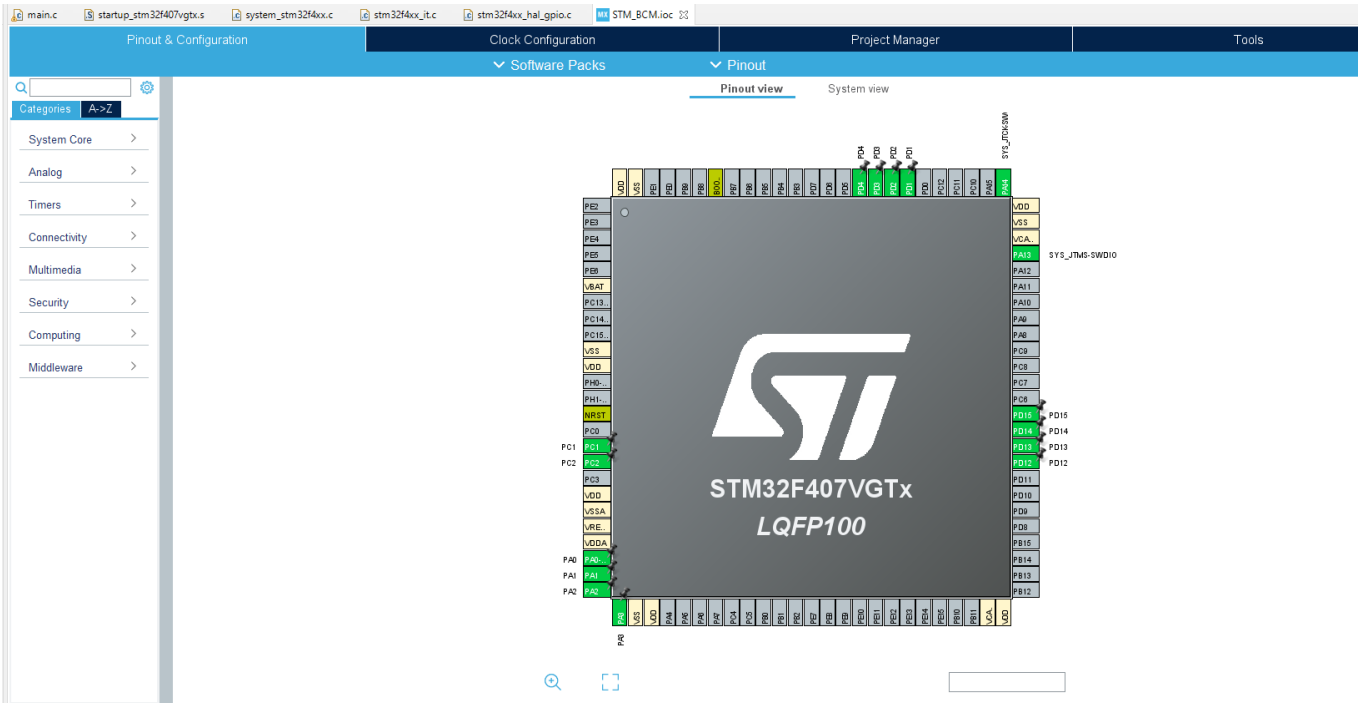


Fig 3.2 Pin configuration

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.

```
/*-----------------------------------------------------------------------------------------------------------------------------------------------*/
      if(HAL_GPIO_ReadPin(PA1_GPIO_Port, PA1_Pin) == 1)
      {


          if(HAL_GPIO_ReadPin(PA2_GPIO_Port, PA2_Pin)==0  && (HAL_GPIO_ReadPin(PA3_GPIO_Port, PA3_Pin)==0)){

            HAL_GPIO_TogglePin(PD1_GPIO_Port, PD1_Pin);
            HAL_Delay(900);

          }else if((HAL_GPIO_ReadPin(PA2_GPIO_Port, PA2_Pin)==1) && (HAL_GPIO_ReadPin(PA3_GPIO_Port, PA3_Pin)==0)){

              HAL_GPIO_TogglePin(PD2_GPIO_Port, PD2_Pin);
              HAL_GPIO_TogglePin(PD1_GPIO_Port, PD1_Pin);
              HAL_Delay(100);


          }else if(HAL_GPIO_ReadPin(PA2_GPIO_Port, PA2_Pin)==1 && HAL_GPIO_ReadPin(PA3_GPIO_Port, PA3_Pin)==1) {

              HAL_GPIO_TogglePin(PD2_GPIO_Port, PD2_Pin);
              HAL_Delay(100);
          }

      }else{
          //pass comment

      }
```

Fig 3.3 Alarm system-code

Implementation:

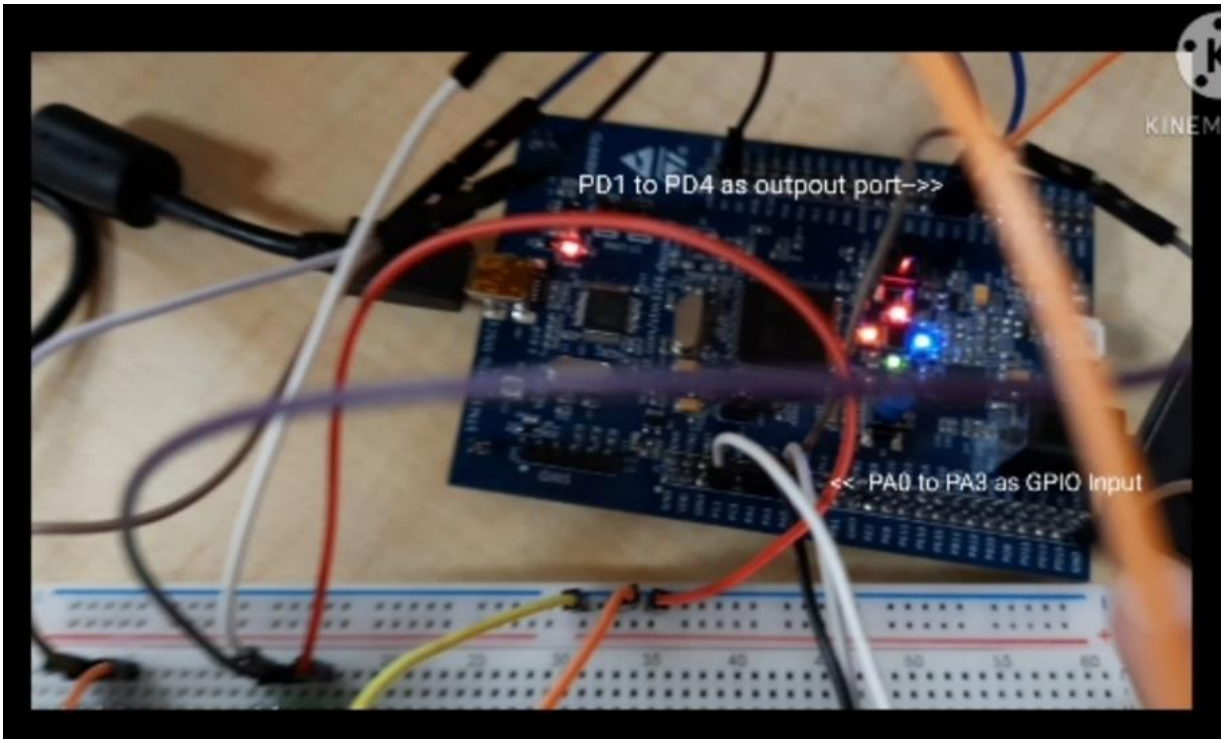


Fig 3.4 Alarm system - Implementation

2. 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

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

    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
    }
```

Fig 3.5 Seat control system- code

Implementation:

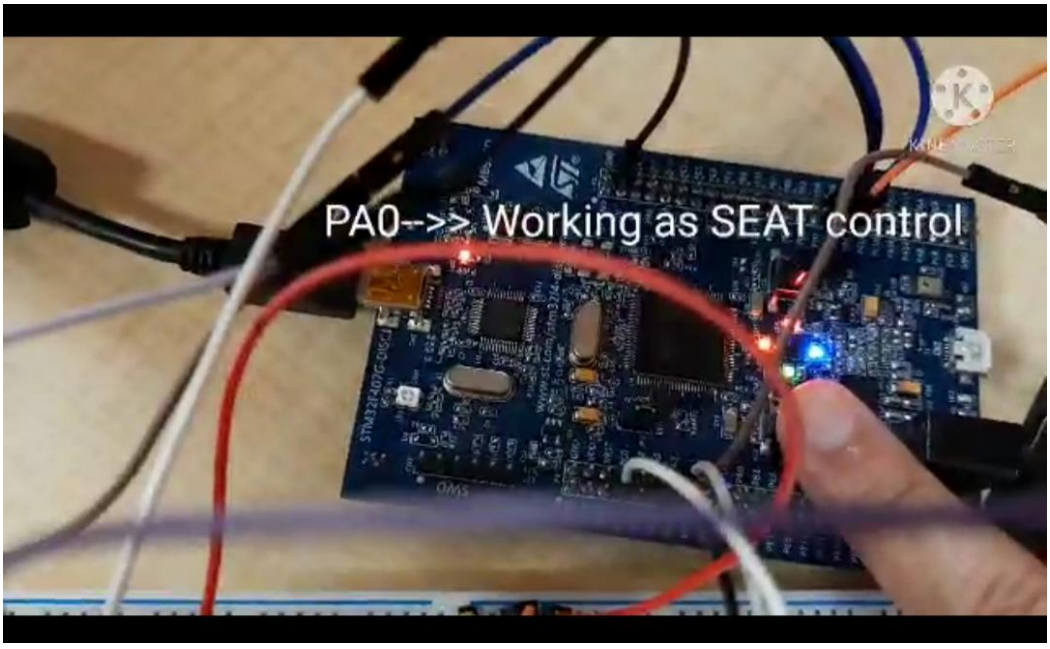


Fig 3.6 Seat control system- Implementation

3. 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.

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


if(HAL_GPIO_ReadPin(PA0_GPIO_Port, PA0_Pin)==1  && HAL_GPIO_ReadPin(PA3_GPIO_Port, PA3_Pin)==1){
        HAL_GPIO_TogglePin(PD12_GPIO_Port, PD12_Pin);
        HAL_Delay(100);
        HAL_GPIO_TogglePin(PD13_GPIO_Port, PD13_Pin);
        HAL_Delay(100);
       }

//while '}'
  }
```

Fig 4.6 Power mirror- code

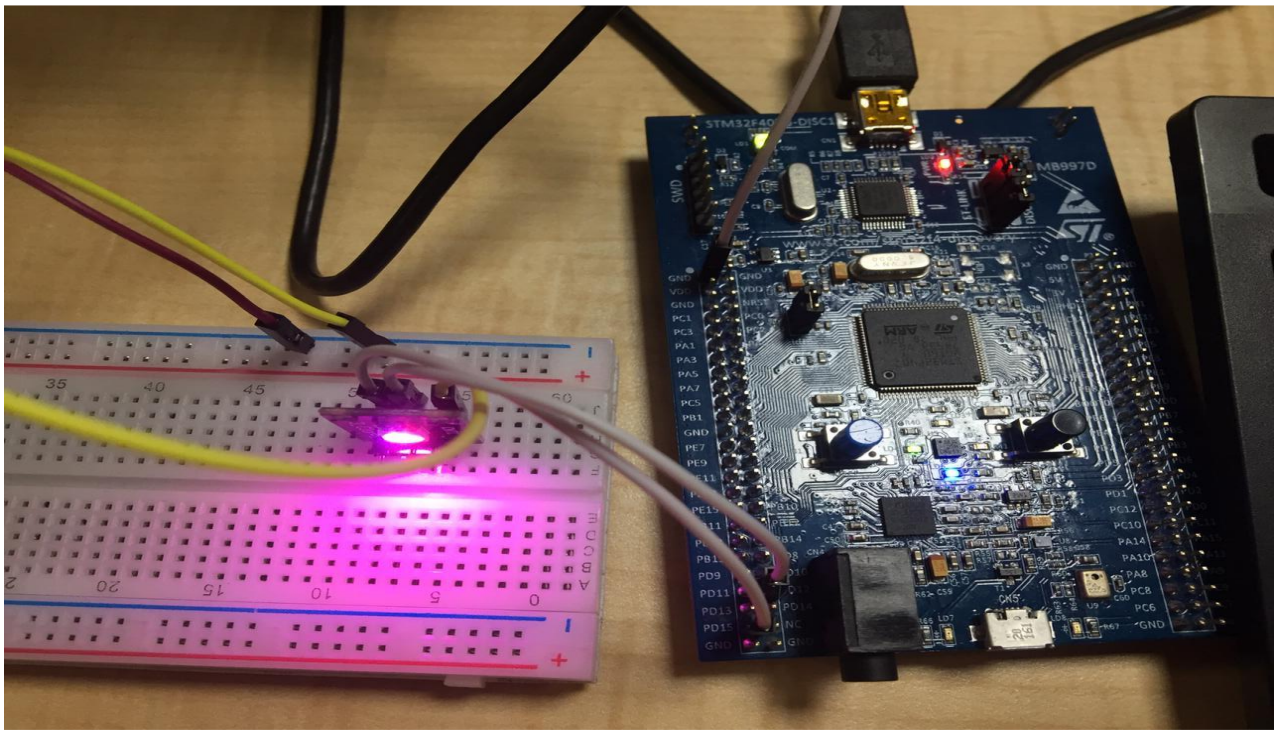


Fig 3.7 Power mirror- Implementation

4. 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.
 /*****************************************************************************************************************/
     if(HAL_GPIO_ReadPin(PC1_GPIO_Port, PC1_Pin)==1){

           HAL_GPIO_TogglePin(PD3_GPIO_Port, PD3_Pin);
           HAL_Delay(200);

     }else{
    //pass commment
     }
```

Fig 3.8 Automatic wiper system- code

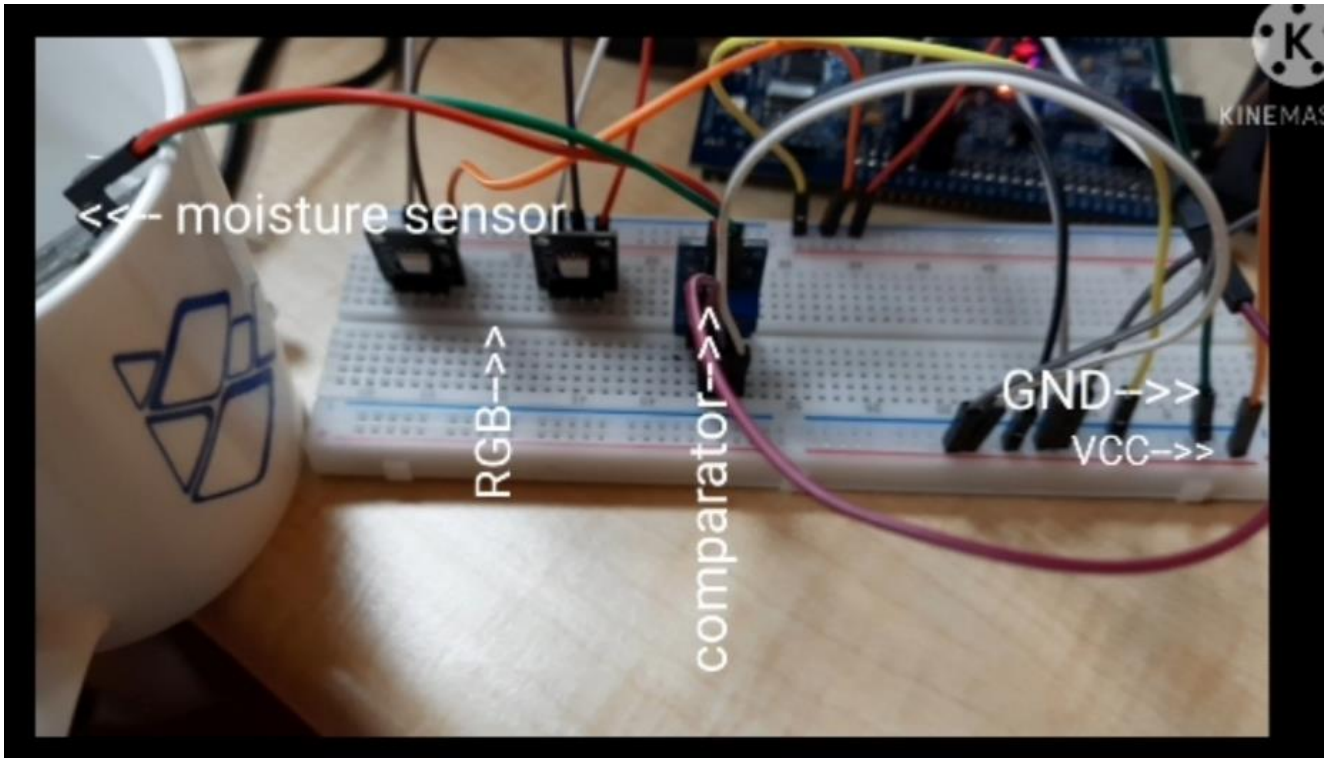


Fig 3.9 Automatic wiper system- Implementation

Final integrated project:

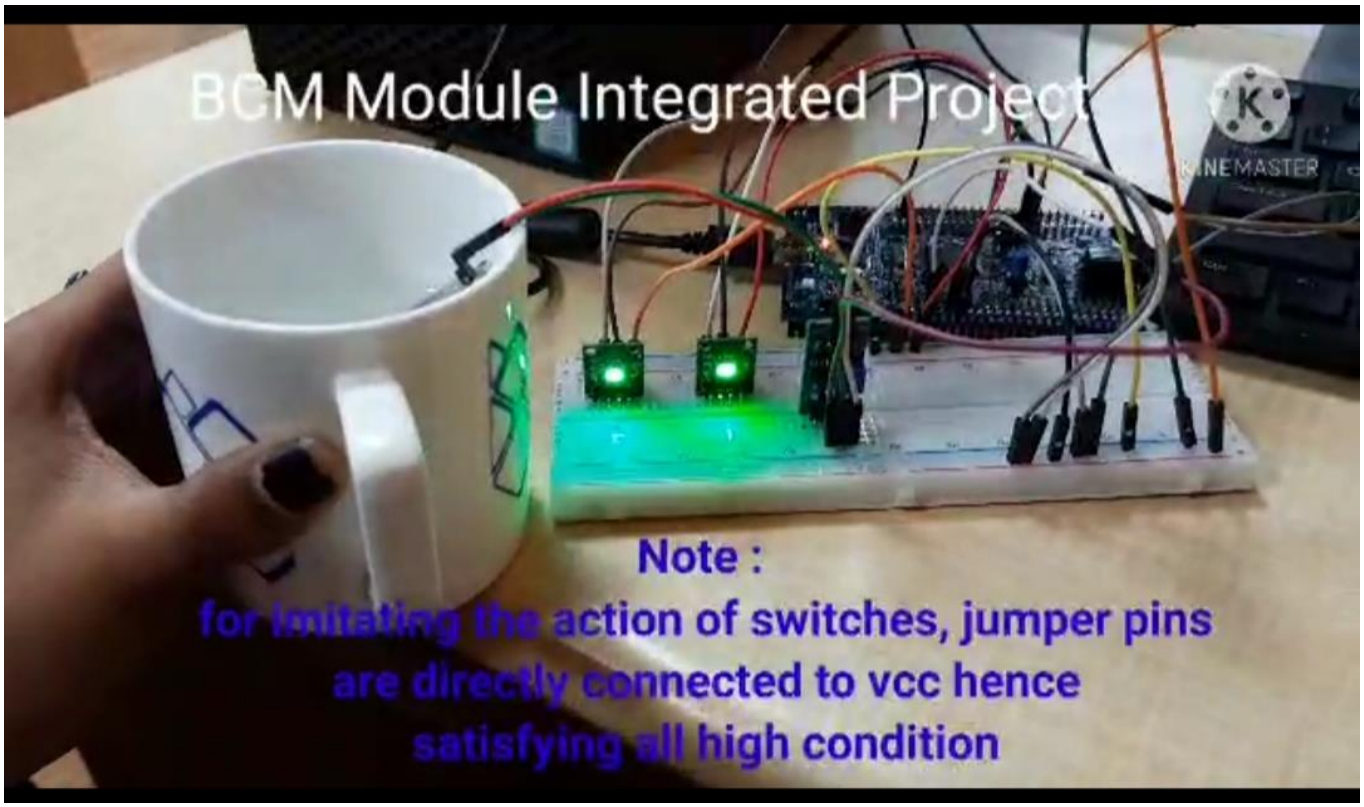


Fig 3.10: Final Integrated BCM