./

Learning Report – Embedded C – Hardware + Programming + Testing



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1.0 | 29/12/2020 | SHREYA KORI |  |  |  |
| 1.1 | 03/01/2021 | SHREYA KORI |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Table of Contents

[Table of Contents 3](#_Toc60494291)

[1. ACTIVITY 1: BUILD PROCESS 4](#_Toc60494292)

[**1.1. main.c 4**](#_Toc60494293)

[2. ACTIVITY 2: LINKER SCRIPT 6](#_Toc60494294)

[3. ACTIVITY 3: HEADER FILE 7](#_Toc60494295)

[**3.1 MCU- Specific Header File 7**](#_Toc60494296)

[**3.2 GPIO-Specific Header File 10**](#_Toc60494297)

[**3.3 Source File: 12**](#_Toc60494298)

[4. ACTIVITY 4: 17](#_Toc60494299)

[**GITHUB LINK: https://github.com/99003163/Embedded\_C 17**](#_Toc60494300)

[REFEERENCES: 17](#_Toc60494301)

# ACTIVITY 1: BUILD PROCESS

## **main.c**

#include<stdio.h>

int main()

{

int num1=5;

int num2=10;

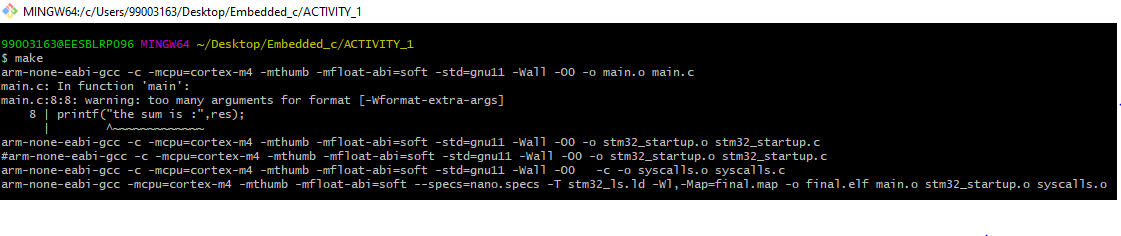
int addition;

addition = num1+num2;

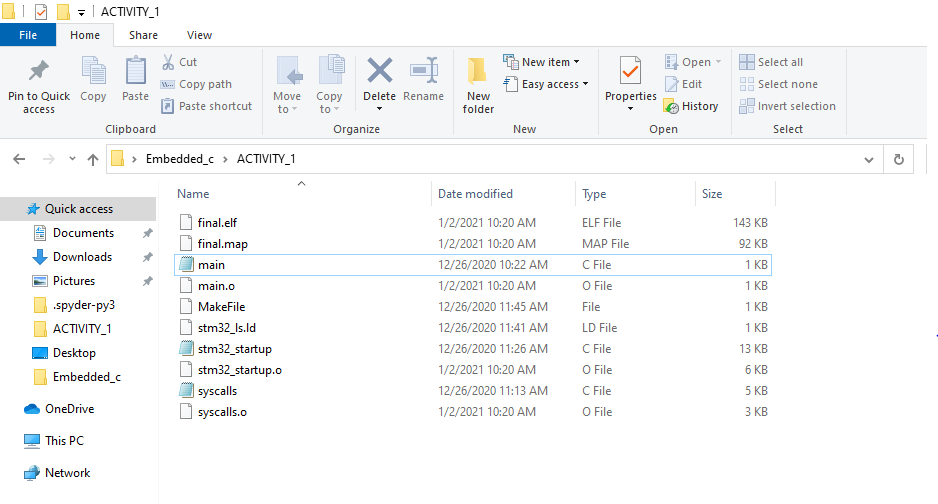
printf("the sum is :",addition);

return 0;

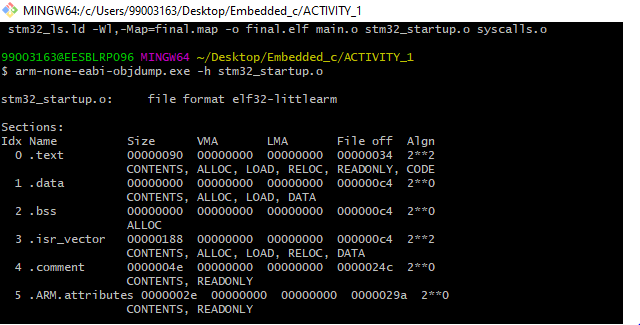
}



**Fig. 1: Make using GIT Bash**



**Fig. 2: Activity Folder**



**Fig. 3: Memory Organization**

# ACTIVITY 2: LINKER SCRIPT

ENTRY(Reset\_Handler)

MEMORY

{

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

}

# ACTIVITY 3: HEADER FILE

## **3.1 MCU- Specific Header File**

**#include** <stdint.h>

**#ifndef** INC\_STM32F4XX\_H\_

**#define** INC\_STM32F4XX\_H\_

**#define** \_\_vo **volatile**

/\* other definitions\*/

**#define** ENABLE 1

**#define** DISABLE 0

**#define** GPIO\_PIN\_SET ENABLE

**#define** GPIO\_PIN\_RESET DISABLE

/\* Defining macros for the various memory \*/

**#define** FLASH\_ADDR 0x80000000U

**#define** SRAM1\_ADDR 0x20000000U

**#define** SRAM2\_ADDR 0x2001C000U

**#define** ROM\_ADDR 0X1FFF0000U

**#define** SRAM\_ADDR SRAM1\_ADDR

/\* Defining macros for bus system \*/

**#define** AHB1\_ADDR 0x40020000U

**#define** AHB2\_ADDR 0x50000000U

**#define** APB1\_ADDR 0x40000000U

**#define** APB2\_ADDR 0x40010000U

**#define** PERI\_ADDR APB1\_ADDR

/\* Defining macros for peripherals hanging on AHB1 Bus \*/

**#define** GPIOA\_ADDR (AHB1\_ADDR + 0x0000U)

**#define** GPIOB\_ADDR (AHB1\_ADDR + 0x0400U)

**#define** GPIOC\_ADDR (AHB1\_ADDR + 0x0800U)

**#define** GPIOD\_ADDR (AHB1\_ADDR + 0x0C00U)

**#define** GPIOE\_ADDR (AHB1\_ADDR + 0x1000U)

**#define** GPIOF\_ADDR (AHB1\_ADDR + 0x1400U)

**#define** GPIOG\_ADDR (AHB1\_ADDR + 0x1800U)

**#define** GPIOH\_ADDR (AHB1\_ADDR + 0x1C00U)

**#define** GPIOI\_ADDR (AHB1\_ADDR + 0x2000U)

**#define** RCC\_ADDR (AHB1\_ADDR + 0x3800U)

/\*Defining the macros for peripherals which are hanging on to APB1 bus\*/

**#define** SPI2\_I2S2\_ADDR (APB1\_ADDR + 0X3800U)

**#define** SPI3\_I2S3\_ADDR (APB1\_ADDR + 0X3C00U)

**#define** USART2\_ADDR (APB1\_ADDR + 0X4400U)

**#define** USART3\_ADDR (APB1\_ADDR + 0X4800U)

**#define** UART4\_ADDR (APB1\_ADDR + 0X4C00U)

**#define** UART5\_ADDR (APB1\_ADDR + 0X5000U)

**#define** I2C1\_ADDR (APB1\_ADDR + 0X5400U)

**#define** I2C2\_ADDR (APB1\_ADDR + 0X5800U)

**#define** I2C3\_ADDR (APB1\_ADDR + 0X5C00U)

**#define** CAN1\_ADDR (APB1\_ADDR + 0X6400U)

**#define** CAN2\_ADDR (APB1\_ADDR + 0X6800U)

/\*Defining the macros for peripherals which are hanging on to APB2 bus\*/

**#define** USART1\_ADDR (APB2\_ADDR + 0X1000U)

**#define** USART6\_ADDR (APB2\_ADDR + 0X1400U)

**#define** SPI1\_ADDR (APB2\_ADDR + 0X3000U)

**#define** RCC ((RCC\_GPIO\_Reg\_def\_t\*)RCC\_ADDR)

/\* GPIO Peripheral Registers \*/

**typedef** **struct**

{

uint32\_t MODER;

uint32\_t OTYPER;

uint32\_t OSPEEDR;

uint32\_t PUPDR;

uint32\_t IDR;

uint32\_t ODR;

uint32\_t BSRR;

uint32\_t LCKR;

uint32\_t AFR[2]; //AFRL[0](Low Register) & AFRH[1](High Register)

}GPIO\_Reg\_def\_t;

/\* RCC Registers\*/

**typedef** **struct**

{

\_\_vo uint32\_t RCC\_CR;

\_\_vo uint32\_t RCC\_PLLCFGR;

\_\_vo uint32\_t RCC\_CFGR;

\_\_vo uint32\_t RCC\_CIR;

\_\_vo uint32\_t RCC\_AHB1RSTR;

\_\_vo uint32\_t RCC\_AHB2RSTR;

\_\_vo uint32\_t RCC\_AHB3RSTR;

uint32\_t RESERVED0;

\_\_vo uint32\_t RCC\_APB1RSTR;

\_\_vo uint32\_t RCC\_APB2RSTR;

uint32\_t RESERVED1[2];

\_\_vo uint32\_t RCC\_AHB1ENR;

\_\_vo uint32\_t RCC\_AHB2ENR;

uint32\_t RESERVED2[2];

\_\_vo uint32\_t RCC\_AHB3ENR;

\_\_vo uint32\_t RCC\_APB1ENR;

\_\_vo uint32\_t RCC\_APB2ENR;

\_\_vo uint32\_t RCC\_AHB1LPENR;

\_\_vo uint32\_t RCC\_AHB2LPENR;

\_\_vo uint32\_t RCC\_AHB3LPENR;

\_\_vo uint32\_t RCC\_APB1LPENR;

\_\_vo uint32\_t RCC\_APB2LPENR;

\_\_vo uint32\_t RCC\_BDCR;

\_\_vo uint32\_t RCC\_CSR;

\_\_vo uint32\_t RCC\_SSCGR;

\_\_vo uint32\_t RCC\_PLLI2SCFGR;

\_\_vo uint32\_t RCC\_PLLSAICFGR;

\_\_vo uint32\_t RCC\_DCKCFGR;

} RCC\_GPIO\_Reg\_def\_t;

/\* GPIO Clock Enable \*/

**#define** GPIOA\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<0

**#define** GPIOB\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<1

**#define** GPIOC\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<2

**#define** GPIOD\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<3

**#define** GPIOE\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<4

**#define** GPIOF\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<5

**#define** GPIOG\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<6

**#define** GPIOH\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<7

**#define** GPIOI\_PE\_CLOCK\_ENABLE() RCC->RCC\_AHB1ENR |= 1<<8

/\* GPIO Clock Disable

#define GPIOA\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOB\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOC\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOD\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOE\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOF\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOG\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOH\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)

#define GPIOI\_PE\_CLOCK\_DISABLE() RCC->RCC\_AHB1ENR &= ~(1<<0)\*/

/\* GPIO Clock Reset \*/

**#define** GPIOA\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<0);RCC->RCC\_AHB1RSTR &= ~(1<<0);}**while**(DISABLE)

**#define** GPIOB\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<1);RCC->RCC\_AHB1RSTR &= ~(1<<1);}**while**(DISABLE)

**#define** GPIOC\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<2);RCC->RCC\_AHB1RSTR &= ~(1<<2);}**while**(DISABLE)

**#define** GPIOD\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<3);RCC->RCC\_AHB1RSTR &= ~(1<<3);}**while**(DISABLE)

**#define** GPIOE\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<4);RCC->RCC\_AHB1RSTR &= ~(1<<4);}**while**(DISABLE)

**#define** GPIOF\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<5);RCC->RCC\_AHB1RSTR &= ~(1<<5);}**while**(DISABLE)

**#define** GPIOG\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<6);RCC->RCC\_AHB1RSTR &= ~(1<<6);}**while**(DISABLE)

**#define** GPIOH\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<7);RCC->RCC\_AHB1RSTR &= ~(1<<7);}**while**(DISABLE)

**#define** GPIOI\_PE\_CLOCK\_RESET() **do**{RCC->RCC\_AHB1RSTR |= (1<<8);RCC->RCC\_AHB1RSTR &= ~(1<<8);}**while**(DISABLE)

/\* GPIO Peripheral \*/

**#define** GPIOA ((GPIO\_Reg\_def\_t\*)GPIOA\_ADDR)

**#define** GPIOB ((GPIO\_Reg\_def\_t\*)GPIOB\_ADDR)

**#define** GPIOC ((GPIO\_Reg\_def\_t\*)GPIOC\_ADDR)

**#define** GPIOD ((GPIO\_Reg\_def\_t\*)GPIOD\_ADDR)

**#define** GPIOE ((GPIO\_Reg\_def\_t\*)GPIOE\_ADDR)

**#define** GPIOF ((GPIO\_Reg\_def\_t\*)GPIOF\_ADDR)

**#define** GPIOG ((GPIO\_Reg\_def\_t\*)GPIOG\_ADDR)

**#define** GPIOH ((GPIO\_Reg\_def\_t\*)GPIOH\_ADDR)

**#define** GPIOI ((GPIO\_Reg\_def\_t\*)GPIOI\_ADDR)

**#endif** /\* INC\_STM32F4XX\_H\_ \*/

## **3.2 GPIO-Specific Header File**

**#include** "stm32f4xx.h"

**#ifndef** INC\_STM32F4XX\_GPIO\_DRIVER\_H\_

**#define** INC\_STM32F4XX\_GPIO\_DRIVER\_H\_

/\* GPIO Pin Configuration\*/

**typedef** **struct**

{

uint8\_t GPIO\_Pin\_Number;

uint8\_t GPIO\_PinMode;

uint8\_t GPIO\_Pin\_Speed;

uint8\_t GPIO\_Pin\_PuPd\_Control;

uint8\_t GPIO\_Pin\_OP\_Type;

uint8\_t GPIO\_Pin\_Alt\_Fun\_Mode;

}GPIO\_PIN\_CONFIG\_T;

/\* GPIO Handle Structure \*/

**typedef** **struct**

{

GPIO\_Reg\_def\_t \*pGPIOx;

GPIO\_PIN\_CONFIG\_T PIN\_CONFIG;

}GPIO\_HANDLE\_T;

/\* GPIO pin numbering \*/

**#define** GPIO\_PIN\_NUMBER\_0 0

**#define** GPIO\_PIN\_NUMBER\_1 1

**#define** GPIO\_PIN\_NUMBER\_2 2

**#define** GPIO\_PIN\_NUMBER\_3 3

**#define** GPIO\_PIN\_NUMBER\_4 4

**#define** GPIO\_PIN\_NUMBER\_5 5

**#define** GPIO\_PIN\_NUMBER\_6 6

**#define** GPIO\_PIN\_NUMBER\_7 7

**#define** GPIO\_PIN\_NUMBER\_8 8

**#define** GPIO\_PIN\_NUMBER\_9 9

**#define** GPIO\_PIN\_NUMBER\_10 10

**#define** GPIO\_PIN\_NUMBER\_11 11

**#define** GPIO\_PIN\_NUMBER\_12 12

**#define** GPIO\_PIN\_NUMBER\_13 13

**#define** GPIO\_PIN\_NUMBER\_14 14

**#define** GPIO\_PIN\_NUMBER\_15 15

/\* GPIO Operating Modes \*/

**#define** GPIO\_PIN\_MODE\_IN 0

**#define** GPIO\_PIN\_MODE\_OUT 1

**#define** GPIO\_PIN\_MODE\_ALT 2

**#define** GPIO\_PIN\_MODE\_ANALOG 3

**#define** GPIO\_PIN\_MODE\_RT 4

**#define** GPIO\_PIN\_MODE\_FT 5

**#define** GPIO\_PIN\_MODE\_RFT 6

/\* GPIO pin possible output speeds\*/

**#define** GPIO\_PIN\_SPEED\_LOW 0

**#define** GPIO\_PIN\_SPEED\_MEDIUM 1

**#define** GPIO\_PIN\_SPEED\_FAST 2

**#define** GPIO\_PIN\_SPEED\_HIGH 3

/\* GPIO\*/

**#define** GPIO\_PIN\_PUPD\_CONTROL\_0 0

**#define** GPIO\_PIN\_PUPD\_CONTROL\_1 1

**#define** GPIO\_PIN\_PUPD\_CONTROL\_2 2

**#define** GPIO\_PIN\_PUPD\_CONTROL\_3 3

/\* GPIO Pin Output Types \*/

**#define** GPIO\_OP\_TYPE\_PP 0

**#define** GPIO\_OP\_TYPE\_OD 1

/\* GPIO pin pull up and pull down configuration \*/

**#define** GPIO\_NO\_PUPD 0

**#define** GPIO\_PIN\_PU 1

**#define** GPIO\_PIN\_PD 2

**void** **GPIO\_PeriClockControl**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t EnorDi);

**void** **GPIO\_Init**(GPIO\_HANDLE\_T \*pGPIOHandle);

**void** **GPIO\_DeInit**(GPIO\_Reg\_def\_t \*pGIOx);

uint8\_t **GPIO\_ReadFromInputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber);

uint16\_t **GPIO\_ReadFromInputPort**(GPIO\_Reg\_def\_t \*pGPIOx);

**void** **GPIO\_WriteToOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber,uint8\_t value);

**void** **GPIO\_WriteToOutputPort**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t value);

**void** **GPIO\_ToggleOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber);

**#endif** /\* INC\_STM32F4XX\_GPIO\_DRIVER\_H\_ \*/

## **Source File:**

**#include** "stm32f4XX\_GPIO\_driver.h"

**void** **GPIO\_PeriClockControl**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t EnorDi)

{

**if**(EnorDi == ENABLE)

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOH)

{

GPIOH\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOI)

{

GPIOI\_PE\_CLOCK\_ENABLE();

}

}

**else**

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOH)

{

GPIOH\_PE\_CLOCK\_RESET();

}

**else** **if**(pGPIOx == GPIOI)

{

GPIOI\_PE\_CLOCK\_RESET();

}

}

}

**void** **GPIO\_Init**(GPIO\_HANDLE\_T \*pGPIOHandle)

{

//1. configuring the mode

uint32\_t temp=0;

**if**(pGPIOHandle->PIN\_CONFIG.GPIO\_PinMode <= GPIO\_PIN\_MODE\_ANALOG )//non interrupt modes

{

temp = pGPIOHandle->PIN\_CONFIG.GPIO\_PinMode<<(2\*pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number);

pGPIOHandle->pGPIOx->MODER |= temp;

}

**else**

{

//interrupt mode FT, RT ,FTRT

}

//2. configuring the speed

uint32\_t temp1=0;

temp1 = pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Speed<<(2\*pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number);

pGPIOHandle->pGPIOx->OSPEEDR |= temp1;

//3. configuring the pu pd control

uint32\_t temp2=0;

temp2 = pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_PuPd\_Control<<(2\*pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number);

pGPIOHandle->pGPIOx->PUPDR |= temp2;

//4. configuring the output type

uint32\_t temp3=0;

temp3 = pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_OP\_Type<<(pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number);

pGPIOHandle->pGPIOx->OTYPER |= temp3;

uint32\_t tempA = pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number /8;

uint32\_t tempB = pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Number %8;

pGPIOHandle->pGPIOx->AFR[tempA] |= pGPIOHandle->PIN\_CONFIG.GPIO\_Pin\_Alt\_Fun\_Mode << (4\*tempB);

**if**(tempA == 0)

{

**if**(tempB == 0);

}

}

**void** **GPIO\_DeInit**(GPIO\_Reg\_def\_t \*pGIOx)

{

**if**(EnorDi == ENABLE)

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOH)

{

GPIOH\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOI)

{

GPIOI\_PE\_CLOCK\_ENABLE();

}

}

**else**

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOH)

{

GPIOH\_PE\_CLOCK\_ENABLE();

}

**else** **if**(pGPIOx == GPIOI)

{

GPIOI\_PE\_CLOCK\_ENABLE();

}// To be done

}

}

uint8\_t **GPIO\_ReadFromInputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber)

{

uint8\_t value;

value = (uint8\_t)((pGPIOx->IDR >> PinNumber) \* (0x00000001));

**return** value;

}

uint16\_t **GPIO\_ReadFromInputPort**(GPIO\_Reg\_def\_t \*pGPIOx)

{

uint16\_t value;

value = (uint16\_t)(pGPIOx->IDR);

**return** value;

}

**void** **GPIO\_WriteToOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber,uint8\_t value)

{

**if**(value == GPIO\_PIN\_SET)

{

pGPIOx->ODR |= (1 << PinNumber);

}

**else**

{

pGPIOx->ODR &= ~(1 << PinNumber);

}

}

**void** **GPIO\_WriteToOutputPort**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t value)

{

pGPIOx->ODR = value;

}

**void** **GPIO\_ToggleOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx,uint8\_t PinNumber)

{

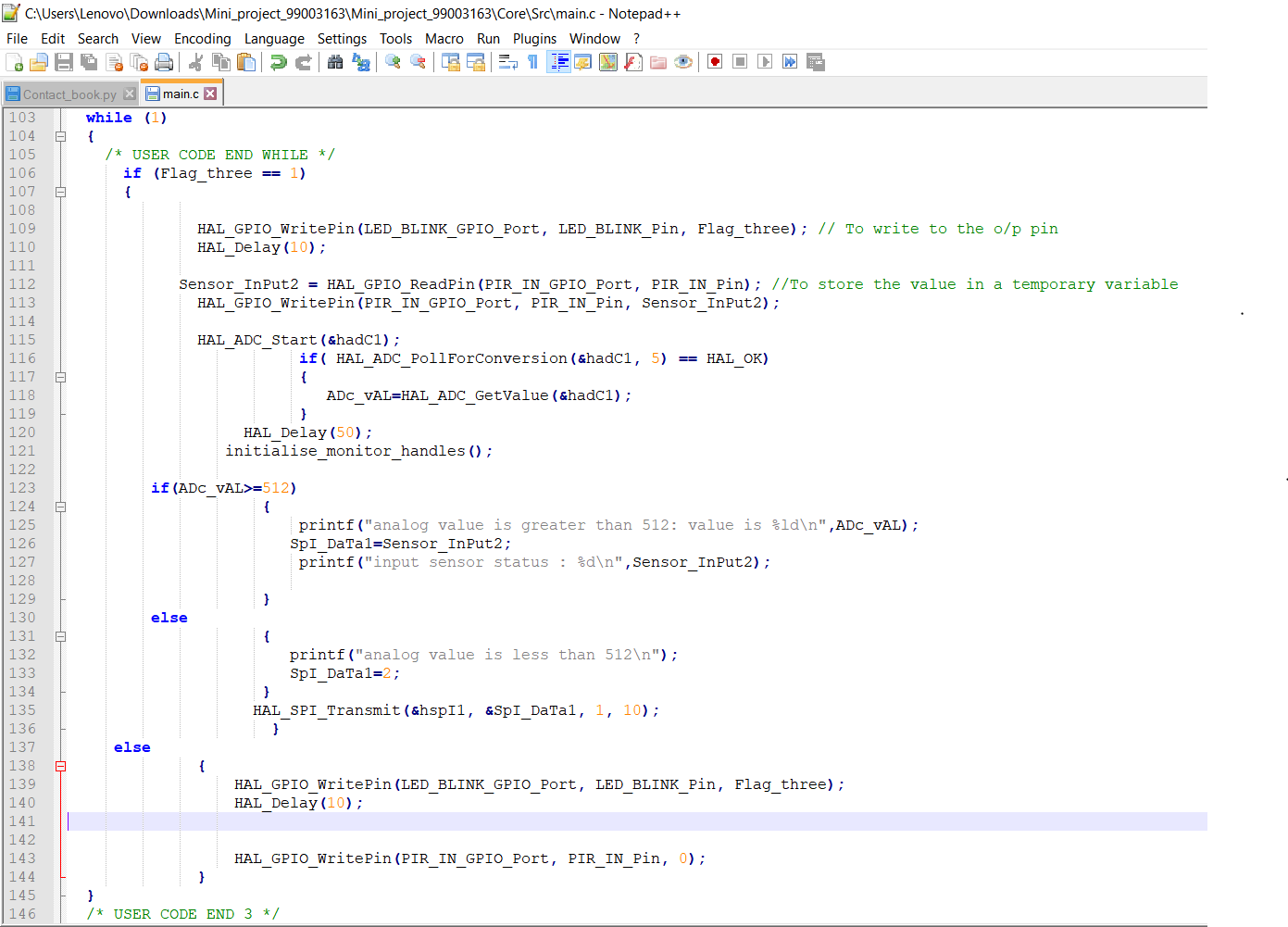
pGPIOx->ODR ^= (1<<PinNumber);

}

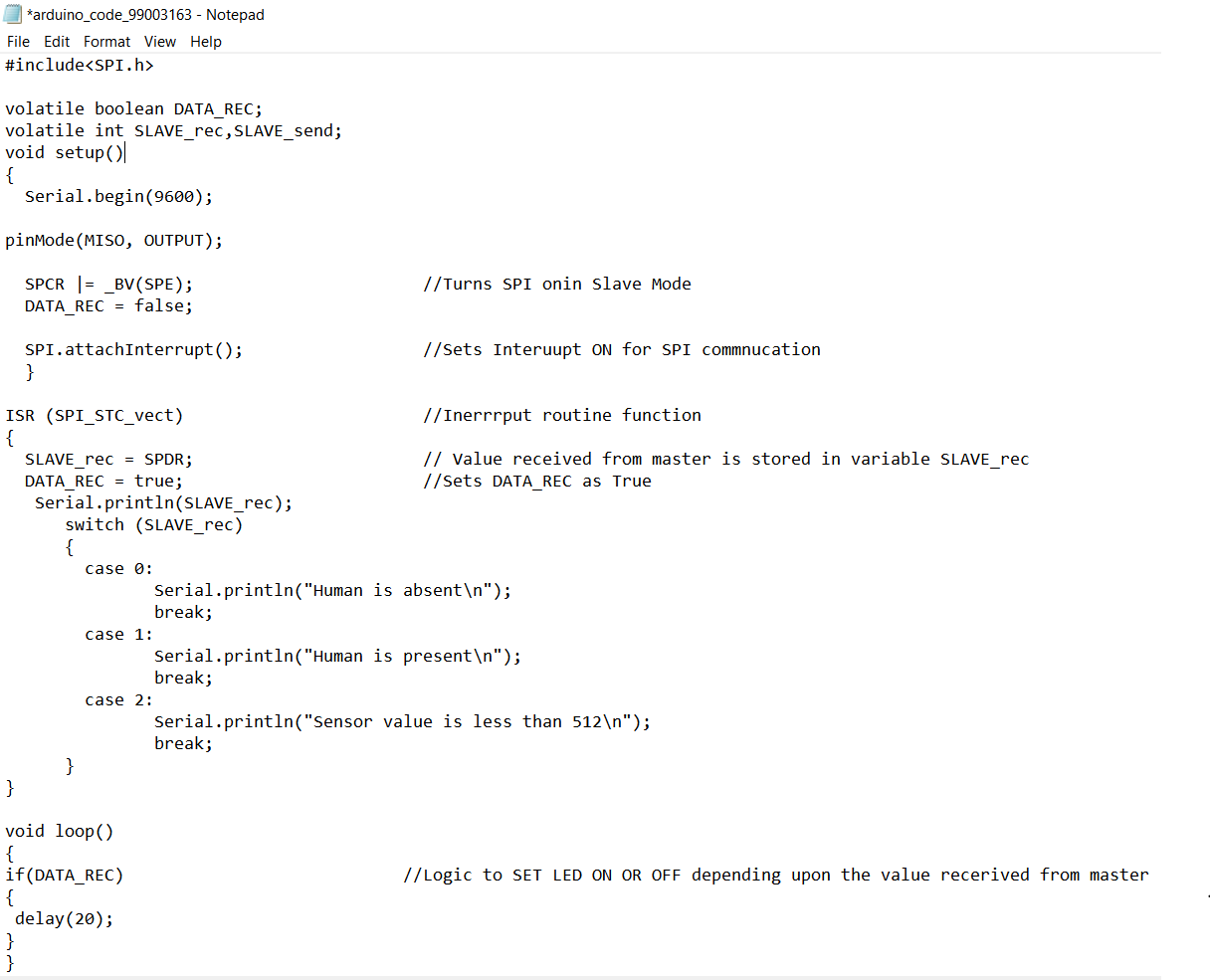
# ACTIVITY 4:

## **GITHUB LINK:** <https://github.com/99003163/Embedded_C>

## **LOGIC CODE**



## **ARDUINO CODE**



# REFEERENCES:

[1]. <http://web.cs.iastate.edu/~smkautz/cs227s13/labs/lab6/page04.html>

[2]. <https://youtu.be/2Hm8eEHsgls>

[3].<https://youtu.be/Bsq6P1B8JqI>