University of Dhaka

Department of Computer Science and Engineering

CSE-3113 : Microprocessor and Assembly Language Lab

Lab Report 6 : Bare-metal programming: Development of lower-level system library for unrestricted programming

Submitted By:

Name : Nur Jannat Meherin

Roll No : 37

Name : Nafisa Anzum

Roll No : 45

Submitted On :

July 22, 2023

Submitted To :

Dr. Upama Kabir

Dr. Md. Mustafizur Rahman

1 Introduction

In the ever-evolving landscape of embedded systems, the STM32F446RE microcontroller stands as a powerful and versatile component capable of driving innovation and efficiency across various applications. As the demand for more sophisticated and specialized functionalities increases, developers seek to harness the complete potential of the hardware to achieve optimal performance and resource utilization. This is where bare metal programming comes into play.

The STM32F446RE microcontroller, part of STMicroelectronics' STM32 family, boasts an ARM Cortex-M4 core, offering a robust platform for bare metal programming – a method that allows developers to write code that directly interacts with the hardware, bypassing the complexities of an operating system. By doing so, developers gain unprecedented control over system resources, real-time responsiveness, and reduced memory footprint, enabling the creation of highly efficient and tailor-made firmware solutions.

In this comprehensive report, we delve into the world of bare metal programming on the STM32F446RE microcontroller. We explore the fundamental concepts, advantages, and challenges of this programming approach and provide practical examples that demonstrate how to harness the true power of the microcontroller through direct hardware manipulation.

2 Objectives

Learn different networking commands:

2.1 General Objectives :

This lab assignment aims to:

- Understand and have hands-on training to build a microcontroller driver for unrestricted/unlimited programming.
- Envision enriching students to have the boldness to develop deep system-level programming for the use of any developed micro-controller. In this case, developers do not need to wait for a new release of hardware abstraction layer like HAL (in CubeIde) or CMSIS of Keil.
- Unveil and give a solid knowledge of the linker, loader, and makefile components and concepts.

- Describe the memory mapping and address generation for area and sections for the machine code.

2.2 Working Objectives :

The student will be able to :

- Configure the system clock and GPIO port as done in the micro-controller assignments.
- Recognize input and lightening and LED using a GPIO port as they did in the μC assignment.
- Compile and run any program that uses a clock and GPIO. However, students can extend their design or tools with the full support of SMT32xxx to the programmers.
- Learn a tutorial for an initial library or driver development for the STM32 microcontroller.

3 Commands :

- **arm-none-eabi-gcc :** If we want to check id gcc is available.
- **arm-none-eabi-gcc –version :** If we want to check the version of gcc
- **arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb main.c -o main.o :** If we want compile and assemble the single file(linking is not done here)
- **arm-none-eabi-gcc -S -mcpu=cortex-m4 -mthumb main.c -o main.s :** if we want to just Compile the single file(assemble and linking is not done here)
- **make :** The .o files will be created
- **arm-none-eabi-objdump.exe :** Commands to create list of different file format
- **arm-none-eabi-objdump.exe -h main.o :** Display the contents of the section headers
- **arm-none-eabi-objdump.exe -d main.o ¿ mainlog :** Create the mainlog file in the folder with the sections .
- **arm-none-eabi-objdump.exe -s main.o :** Display the full contents of all sections requested
- **make all :** Build all the files
- **rm-none-eabi-objdump.exe -D led.o ¿ ledlog :** Create the ledlog file in the folder with the sections
- **ls :** Show all the file names in a directory
- **touch stm32startup.c :** Create an empty file named stm32startup.c
- **vi Makefile :** Run the makefile
- **make clean :** To clean the unnecessary files.
- **arm-none-eabi-gcc -nostdlib -T stm32**$_ls.ld * .o - ofinal.elf$: $Makefinal.elffile$**arm-none-eabi-objdump.exe -h final.elf :** $Analyzethesections(alsoknow$

- **arm-none-eabi-nm.exe final.elf :** Symbol table maintained by the linker
- **make load :** This will load the code into the stm32 microcontroller.
- **arm-none-eabi-gdb.exe :** This will enable the gdb prompt
- **target remote localhost:3333 :** Connect OpenOCD server ti gdb client
- **monitor reset init :** Monitor reset initialization.
- **monitor flash write**$_{i}mageerasefinal.elf : Loadexecutablefinal.elfintothemicrocontroller$**mo**
- **monitor resume :** Run the program.
- **monitor halt :** Stop program execution.
- **monitor reset :** Reset the stm board.
- **monitor mdw 0x20000000 4 :** Value of the current task variable will be found.
- **quit :** Leave the GDb client and OpenOCD.
- **vim main.c :** Can edit the file in git bash
- **vim Makefile :** Can edit the file in git bash
- **rm-none-eabi-objdump.exe -h final.elf :** To show the standard library section

4 Code Explanation :

Let's analyze all the codes in the code files :

4.1 led.c :

```
#include<stdint.h>

// This includes the standard integer header, which provides definitions for fixed-wi

#include "stm32f446xx.h"

//This includes the device-specific header file for the STM32F446 microcontroller ser

#include "led.h"

//This includes the user-defined header file "led.h," which presumably contains macro


void led_init(uint8_t input_led, uint8_t output_led)

//This function is responsible for initializing the input and output pins connected t

{
        RCC->AHB1ENR |= (1<<0);

        // This line enables the clock for GPIO Port A by setting the 0th bit of the

        GPIOA->MODER |= (0<<(input_led*2)) | (1<<(output_led*2));

        // This line configures the mode of the specified GPIO pins for input and out

}

void led_on(uint8_t led_no)

//This function turns on the specified LED, given its index led_no.

{
        GPIOA->BSRR |= (1<<led_no);
        // This line sets the corresponding bit of the BSRR (Bit Set/Reset Register)

}

void led_off(uint8_t led_no)

//This function turns off the specified LED, given its index led_no.

{
        GPIOA->BSRR |= (1<<led_no) << 16;

        //This line sets the corresponding bit of the BSRR register in the GPIOA peri

}
```

4.2 led.h :

```
#ifndef LED_H_

//This is a preprocessor directive that checks if the macro LED_H_ is not defined. If

#define LED_H_

//This line defines the macro LED_H_. It is used to prevent multiple inclusions of th

void led_init(uint8_t input_led, uint8_t output_led);

//This function prototype declares a function named led_init that takes two uint8_t (

void led_on(uint8_t led_no);

//This function prototype declares a function named led_on that takes a single uint8_

void led_off(uint8_t led_no);

//This function prototype declares a function named led_off that takes a single uint8

#endif
```

4.3 main.c :

```
#include <stdio.h>
#include "led.h"
#include "systemClock.h"
#include "stm32f446xx.h"

// These lines include user-defined header files: "led.h" and "systemClock.h," as wel

int main(void)
{

//This is the main function, the entry point of the program, which is executed when t

        sysClockConfig();

        // This function is called to configure the system clock of the microcontroll

        led_init(1,4);

        // This function is called to initialize the LEDs. The function led_init() is

        while(1){

        // loop creates an infinite loop, which means the code inside the loop will r

                uint32_t in = GPIOA->IDR;

            // This line reads the input data register (IDR) of GPIOA, which is likel

                if((in>>1)&1)
                        led_on(4);

            // These lines checks if the second bit of the in variable (corresponding

                else
                        led_off(4);

            // If the condition in the previous line is false, this part of the code
            }

        }
```

4.4 main.h :

```
#ifndef MAIN_H_
#define MAIN_H_

// These are conditional inclusion guards, which prevent the contents of the header f


#define MAX_TASKS   5

// This line defines a macro MAX_TASKS with the value 5. It sets the maximum number o

/* some stack memory calculations */
#define SIZE_TASK_STACK          1024U
#define SIZE_SCHED_STACK         1024U

//These lines define macros for the size of task and scheduler stacks. They are set t

#define SRAM_START               0x20000000U

// This macro represents the starting address of the SRAM, which is 0x20000000U.

#define SIZE_SRAM                ( (128) * (1024))

// This macro calculates the size of the SRAM, which is 128 KB (128 * 1024 bytes).

#define SRAM_END                 ((SRAM_START) + (SIZE_SRAM) )

//  This macro represents the ending address of the SRAM by adding the size of the SR

#define T1_STACK_START           SRAM_END

// Starting address of Task 1 stack.

#define T2_STACK_START           ( (SRAM_END) - (1 * SIZE_TASK_STACK) )

// Starting address of Task 2 stack.

#define T3_STACK_START           ( (SRAM_END) - (2 * SIZE_TASK_STACK) )
```

```
Starting address of Task 3 stack.

#define T4_STACK_START           ( (SRAM_END) - (3 * SIZE_TASK_STACK) )

// Starting address of Task 4 stack.

#define IDLE_STACK_START         ( (SRAM_END) - (4 * SIZE_TASK_STACK) )

// Starting address of the Idle task stack.

#define SCHED_STACK_START        ( (SRAM_END) - (5 * SIZE_TASK_STACK) )

//  Starting address of the Scheduler stack.

#define TICK_HZ 1000U

// This line defines a macro TICK_HZ with the value 1000. It sets the frequency of th

#define HSI_CLOCK                         16000000U

// This macro represents the frequency of the HSI (High-Speed Internal) clock, //whic

#define SYSTICK_TIM_CLK                   HSI_CLOCK

// This macro sets the system tick timer clock to the same frequency as the HSI //clo

#define DUMMY_XPSR  0x01000000U

// This line defines a macro DUMMY_XPSR with the value 0x01000000U. It is used to //s

#define TASK_READY_STATE  0x00

// This macro represents the state of a task when it is ready to run.

#define TASK_BLOCKED_STATE  0XFF

// This macro represents the state of a task when it is blocked (not ready to run).

#define INTERRUPT_DISABLE()  do{__asm volatile ("MOV R0,#0x1"); asm volatile("MSR PRI
// This macro disables interrupts by setting the PRIMASK bit to 1 in the Control //(C

#define INTERRUPT_ENABLE()  do{__asm volatile ("MOV R0,#0x0"); asm volatile("MSR PRIM

// This macro enables interrupts by setting the PRIMASK bit to 0 in the Control (CONT

#endif /* MAIN_H_ */
```

4.5 stm32-startup.c :

```
#include <stdint.h>

// This line includes the standard integer types header file, stdint.h, which provide

#define SRAM_START  0x20000000U

//his macro represents the starting address of the SRAM, which is 0x20000000U.

#define SRAM_SIZE   (128U * 1024U)

// This macro calculates the size of the SRAM, which is 128 KB (128 * 1024 //bytes).

#define SRAM_END    ((SRAM_START) + (SRAM_SIZE))

// This macro represents the ending address of the SRAM by adding the size of the SRA

#define STACK_START   SRAM_END

//This line defines a macro STACK_START, which sets the starting address of the stack

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

/* These are external references to symbols defined in the linker script. They repres

extern uint32_t _ebss;
extern uint32_t _sbss;

int main(void);




/* function prototypes of STM32F446xx system exception and IRQ handlers */

void Reset_Handler(void);
void NMI_Handler                                         (void) __attribute__ ((weak,
void HardFault_Handler                                 (void) __attribute__ ((weak, a
void MemManage_Handler                                 (void) __attribute__ ((weak, a
void BusFault_Handler                                 (void) __attribute__ ((weak, al
void UsageFault_Handler                         (void) __attribute__ ((weak, alias("D
void SVC_Handler                                         (void) __attribute__ ((weak,
void DebugMon_Handler                                 (void) __attribute__ ((weak, al
void PendSV_Handler                                   (void) __attribute__ ((weak, al
void SysTick_Handler                                  (void) __attribute__ ((weak, al
void WWDG_IRQHandler                                 (void) __attribute__ ((weak, ali
void PVD_IRQHandler                                 (void) __attribute__ ((weak, alia
void TAMP_STAMP_IRQHandler                         (void) __attribute__ ((weak, alias
void RTC_WKUP_IRQHandler                         (void) __attribute__ ((weak, alias("
void FLASH_IRQHandler                                 (void) __attribute__ ((weak, al
void RCC_IRQHandler                                 (void) __attribute__ ((weak, alia
void EXTI0_IRQHandler                                 (void) __attribute__ ((weak, al
void EXTI1_IRQHandler                                 (void) __attribute__ ((weak, al
void EXTI2_IRQHandler                                 (void) __attribute__ ((weak, al
void EXTI3_IRQHandler                                 (void) __attribute__ ((weak, al
void EXTI4_IRQHandler                                 (void) __attribute__ ((weak, al
void DMA1_Stream0_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream1_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream2_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream3_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream4_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream5_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void DMA1_Stream6_IRQHandler                 (void) __attribute__ ((weak, alias("Defa
void ADC_IRQHandler                                 (void) __attribute__ ((weak, alia
void CAN1_TX_IRQHandler                         (void) __attribute__ ((weak, alias("D
void CAN1_RX0_IRQHandler                         (void) __attribute__ ((weak, alias("
void CAN1_RX1_IRQHandler                         (void) __attribute__ ((weak, alias("
void CAN1_SCE_IRQHandler                         (void) __attribute__ ((weak, alias("
void EXTI9_5_IRQHandler                         (void) __attribute__ ((weak, alias("D
void TIM1_BRK_TIM9_IRQHandler                 (void) __attribute__ ((weak, alias("Def
void TIM1_UP_TIM10_IRQHandler                 (void) __attribute__ ((weak, alias("Def
void TIM1_TRG_COM_TIM11_IRQHandler         (void) __attribute__ ((weak, alias("Defaul
void TIM1_CC_IRQHandler                         (void) __attribute__ ((weak, alias("D
void TIM2_IRQHandler                                 (void) __attribute__ ((weak, ali
void TIM3_IRQHandler                                 (void) __attribute__ ((weak, ali
void TIM4_IRQHandler                                 (void) __attribute__ ((weak, ali
void I2C1_EV_IRQHandler                         (void) __attribute__ ((weak, alias("D
void I2C1_ER_IRQHandler                         (void) __attribute__ ((weak, alias("D
void I2C2_EV_IRQHandler                         (void) __attribute__ ((weak, alias("D
void I2C2_ER_IRQHandler                         (void) __attribute__ ((weak, alias("D
void SPI1_IRQHandler                                  (void) __attribute__ ((weak, al
void SPI2_IRQHandler                                 (void) __attribute__ ((weak, ali
void USART1_IRQHandler                          (void) __attribute__ ((weak, alias("D
void USART2_IRQHandler                          (void) __attribute__ ((weak, alias("D
void USART3_IRQHandler                           (void) __attribute__ ((weak, alias("
void EXTI15_10_IRQHandler                   (void) __attribute__ ((weak, alias("Defau
void RTC_Alarm_IRQHandler                    (void) __attribute__ ((weak, alias("Defa
void OTG_FS_WKUP_IRQHandler             (void) __attribute__ ((weak, alias("Default_H
void TIM8_BRK_TIM12_IRQHandler           (void) __attribute__ ((weak, alias("Default_
void TIM8_UP_TIM13_IRQHandler            (void) __attribute__ ((weak, alias("Default_
void TIM8_TRG_COM_TIM14_IRQHandler         (void) __attribute__ ((weak, alias("Defaul
void TIM8_CC_IRQHandler                  (void) __attribute__ ((weak, alias("Default_
void DMA1_Stream7_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void FMC_IRQHandler                     (void) __attribute__ ((weak, alias("Default_H
void SDIO_IRQHandler                     (void) __attribute__ ((weak, alias("Default_
void TIM5_IRQHandler                     (void) __attribute__ ((weak, alias("Default_
void SPI3_IRQHandler                     (void) __attribute__ ((weak, alias("Default_
void UART4_IRQHandler                    (void) __attribute__ ((weak, alias("Default_
void UART5_IRQHandler                    (void) __attribute__ ((weak, alias("Default_
void TIM6_DAC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_
void TIM7_IRQHandler                     (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream0_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream1_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream2_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream3_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream4_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void CAN2_TX_IRQHandler                  (void) __attribute__ ((weak, alias("Default_
void CAN2_RX0_IRQHandler                 (void) __attribute__ ((weak, alias("Default_
void CAN2_RX1_IRQHandler                 (void) __attribute__ ((weak, alias("Default_
void CAN2_SCE_IRQHandler                 (void) __attribute__ ((weak, alias("Default_
void OTG_FS_IRQHandler                   (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream5_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream6_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void DMA2_Stream7_IRQHandler             (void) __attribute__ ((weak, alias("Default_
void USART6_IRQHandler                   (void) __attribute__ ((weak, alias("Default_
void I2C3_EV_IRQHandler                  (void) __attribute__ ((weak, alias("Default_
void I2C3_ER_IRQHandler                  (void) __attribute__ ((weak, alias("Default_
void OTG_HS_EP1_OUT_IRQHandler           (void) __attribute__ ((weak, alias("Default_
void OTG_HS_EP1_IN_IRQHandler            (void) __attribute__ ((weak, alias("Default_
void OTG_HS_WKUP_IRQHandler              (void) __attribute__ ((weak, alias("Default_
void OTG_HS_IRQHandler                   (void) __attribute__ ((weak, alias("Default_
void DCMI_IRQHandler                     (void) __attribute__ ((weak, alias("Default_
void FPU_IRQHandler                      (void) __attribute__ ((weak, alias("Default_
void SPI4_IRQHandler                      (void) __attribute__ ((weak, alias("Default
void SAI1_IRQHandler                      (void) __attribute__ ((weak, alias("Default
void SAI2_IRQHandler                      (void) __attribute__ ((weak, alias("Default
void QuadSPI_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handl
void HDMI_CEC_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handl
void SPDIF_RX_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handl
void FMPI2C1_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handl
void FMPI2C1_error_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handl


uint32_t vectors[] __attribute__((section(".isr_vector")))   = {

/* This is the Interrupt Vector Table (IVT), an array of function pointers that will

        STACK_START,
        (uint32_t)Reset_Handler,
        (uint32_t)NMI_Handler,
        (uint32_t)HardFault_Handler,
        (uint32_t)MemManage_Handler,
        (uint32_t)BusFault_Handler,
        (uint32_t)UsageFault_Handler,
        0,
        0,
        0,
        0,
        (uint32_t)SVC_Handler,
        (uint32_t)DebugMon_Handler,
        0,
        (uint32_t)PendSV_Handler,
        (uint32_t)SysTick_Handler,
        (uint32_t)WWDG_IRQHandler,
        (uint32_t)PVD_IRQHandler,
        (uint32_t)TAMP_STAMP_IRQHandler,
        (uint32_t)RTC_WKUP_IRQHandler,
        (uint32_t)FLASH_IRQHandler,
        (uint32_t)RCC_IRQHandler,
        (uint32_t)EXTI0_IRQHandler,
        (uint32_t)EXTI1_IRQHandler,
        (uint32_t)EXTI2_IRQHandler,
        (uint32_t)EXTI3_IRQHandler,
        (uint32_t)EXTI4_IRQHandler,
        (uint32_t)DMA1_Stream0_IRQHandler,
        (uint32_t)DMA1_Stream1_IRQHandler,
        (uint32_t)DMA1_Stream2_IRQHandler,
        (uint32_t)DMA1_Stream3_IRQHandler,
        (uint32_t)DMA1_Stream4_IRQHandler,
        (uint32_t)DMA1_Stream5_IRQHandler,
        (uint32_t)DMA1_Stream6_IRQHandler,
        (uint32_t)ADC_IRQHandler,
        (uint32_t)CAN1_TX_IRQHandler,
        (uint32_t)CAN1_RX0_IRQHandler,
        (uint32_t)CAN1_RX1_IRQHandler,
        (uint32_t)CAN1_SCE_IRQHandler,
        (uint32_t)EXTI9_5_IRQHandler,
        (uint32_t)TIM1_BRK_TIM9_IRQHandler,
        (uint32_t)TIM1_UP_TIM10_IRQHandler,
        (uint32_t)TIM1_TRG_COM_TIM11_IRQHandler,
        (uint32_t)TIM1_CC_IRQHandler,
        (uint32_t)TIM2_IRQHandler,
        (uint32_t)TIM3_IRQHandler,
        (uint32_t)TIM4_IRQHandler,
        (uint32_t)I2C1_EV_IRQHandler,
        (uint32_t)I2C1_ER_IRQHandler,
        (uint32_t)I2C2_EV_IRQHandler,
        (uint32_t)I2C2_ER_IRQHandler,
        (uint32_t)SPI1_IRQHandler,
        (uint32_t)SPI2_IRQHandler,
        (uint32_t)USART1_IRQHandler,
        (uint32_t)USART2_IRQHandler,
        (uint32_t)USART3_IRQHandler,
        (uint32_t)EXTI15_10_IRQHandler,
        (uint32_t)RTC_Alarm_IRQHandler,
        (uint32_t)OTG_FS_WKUP_IRQHandler,
        (uint32_t)TIM8_BRK_TIM12_IRQHandler,
        (uint32_t)TIM8_UP_TIM13_IRQHandler,
        (uint32_t)TIM8_TRG_COM_TIM14_IRQHandler,
        (uint32_t)TIM8_CC_IRQHandler,
        (uint32_t)DMA1_Stream7_IRQHandler,
        (uint32_t)FMC_IRQHandler,
        (uint32_t)SDIO_IRQHandler,
        (uint32_t)TIM5_IRQHandler,
        (uint32_t)SPI3_IRQHandler,
        (uint32_t)UART4_IRQHandler,
        (uint32_t)UART5_IRQHandler,
        (uint32_t)TIM6_DAC_IRQHandler,
        (uint32_t)TIM7_IRQHandler,
        (uint32_t)DMA2_Stream0_IRQHandler,
        (uint32_t)DMA2_Stream1_IRQHandler,
        (uint32_t)DMA2_Stream2_IRQHandler,
        (uint32_t)DMA2_Stream3_IRQHandler,
        (uint32_t)DMA2_Stream4_IRQHandler,
        0,                                                                         //
        0,
        (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,
        0,                                                                        //C
        0,
        (uint32_t)FPU_IRQHandler,
        0,
                0,
                (uint32_t)SPI4_IRQHandler,
                0,
                0,
                (uint32_t)SAI1_IRQHandler,
                0,
                0,
                0,
                (uint32_t)SAI2_IRQHandler,
                (uint32_t)QuadSPI_IRQHandler,
                (uint32_t)HDMI_CEC_IRQHandler,
                (uint32_t)SPDIF_RX_IRQHandler,
                (uint32_t)FMPI2C1_IRQHandler,
                (uint32_t)FMPI2C1_error_IRQHandler
        };


        void Default_Handler(void)
        {
        /* This is the default handler function that is called when an interrupt does not hav

                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;
                uint8_t *pSrc = (uint8_t*) &_etext;

                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;
                }




                //call the init function of std library

                // call main() function

                main();

         /* This is the Reset Handler function, which is the first function called after the
        The remaining part of the code (starting from //copy .data section to SRAM) is the in
        }
```

4.6 Makefile :

```
CC=arm-none-eabi-gcc

/* This line defines the variable CC as arm-none-eabi-gcc, which is the cross-compile

MACH=cortex-m4

/* This line defines the variable MACH as cortex-m4, which indicates the target ARM C

CFLAGS= -c -mcpu=$(MACH) -mthumb -std=gnu11 -Wall -o0

/* This line sets the compilation flags for the C compiler :
-c: Indicates that only compilation should be performed, not linking,  -mcpu=£(MACH):

LFLAGS= -nostdlib -T stm32_ls.ld -Wl,-Map=final.map

/* This line sets the linker flags :
 -nostdlib: Informs the linker not to use standard system libraries during the linkin


all:main.o led.o systemClock.o stm32_startup.o final.elf

/*  This line defines the first target all, which is the default target when you run

main.o:main.c
        $(CC) $(CFLAGS) -o $@ $^

 /*  This target defines a rule for building the object file main.o from main.c. : £(

led.o:led.c
        $(CC) $(CFLAGS) -o $@ $^

systemClock.o:systemClock.c
        $(CC) $(CFLAGS) -o $@ $^


stm32_startup.o:stm32_startup.c
        $(CC) $(CFLAGS) -o $@ $^
 /* Similarly, this target builds the stm32_startup.o object file from stm32_startup.

final.elf:main.o led.o systemClock.o stm32_startup.o
        $(CC) $(LFLAGS) -o $@ $^

 /* This target builds the final executable final.elf by linking the object files tog


clean:
        rm -rf *.o *.elf

 /* This target clean is used to remove the object files and the final executable, ef

load:
        openocd -f board/stm32f4discovery.cfg

 /*  This target load is not a default target. It seems to be used to load the compil
```

4.7 stm32F446xx.h :

```
/*
* Data Structure for Reset and Clock Control Registers (RCC), Address Range: 0x4002 3
typedef struct
{
uint32_t volatile CR;                                   /* Offset: 0x00 (R/W) Clock Con
uint32_t volatile PLLCFGR;                            /* Offset: 0x04 (R/W) PLL Configur
uint32_t volatile CFGR;                             /* Offset: 0x08 (R/W) Clock Configura
uint32_t volatile CIR;                                  /* Offset: 0x0C (R/W) Clock In
uint32_t volatile AHB1RSTR;                     /* Offset: 0x10 (R/W) AHB1 Peripheral Res
uint32_t volatile AHB2RSTR;                     /* Offset: 0x14 (R/W) AHB2 Peripheral Res
uint32_t volatile AHB3RSTR;                     /* Offset: 0x18 (R/W) AHB3 Peripheral Res
uint32_t volatile reserved0;
uint32_t volatile APB1RSTR;                     /* Offset: 0x20 (R/W) APB1 Peripheral Res
uint32_t volatile APB2RSTR;                     /* Offset: 0x24 (R/W) APB2 Peripheral Res
uint32_t reserved1[2];
uint32_t volatile AHB1ENR;                            /* Offset: 0x30 (R/W) AHB1 Periphe
uint32_t volatile AHB2ENR;                            /* Offset: 0x34 (R/W) AHB2 Periphe
uint32_t volatile AHB3ENR;                            /* Offset: 0x38 (R/W) AHB3 Periphe
uint32_t reserved2;
uint32_t volatile APB1ENR;                            /* Offset: 0x40 (R/W) APB1 Periphe
uint32_t volatile APB2ENR;                            /* Offset: 0x44 (R/W) APB1 Periphe
uint32_t reserved3[2];
uint32_t volatile AHB1LPENR;                     /* Offset: 0x50 (R/W) AHB1 Peripheral Cl
uint32_t volatile AHB2LPENR;                     /* Offset: 0x54 (R/W) AHB2 Peripheral Cl
uint32_t volatile AHB3LPENR;                     /* Offset: 0x58 (R/W) AHB3 Peripheral Cl
uint32_t reserved4;
uint32_t volatile APB1LPENR;                     /* Offset: 0x60 (R/W) APB1 Peripheral Cl
uint32_t volatile APB2LPENR;                     /* Offset: 0x64 (R/W) APB2 Peripheral Cl
uint32_t reserved5[2];
uint32_t volatile BDCR;                             /* Offset: 0x70 (R/W) Backup Domain C
uint32_t volatile CSR;                                  /* Offset: 0x74 (R/W) Clock Co
uint32_t reserved6[2];
uint32_t volatile SSCGR;                             /* Offset: 0x80 (R/W) Spread Spectru
uint32_t volatile PLLI2SCFGR;                      /* Offset: 0x84 (R/W) PLLI2S Configurat
uint32_t volatile PLLSAICFGR;                      /* Offset: 0x88 (R/W) PLLSAI Configurat
uint32_t volatile DCKCFGR;                            /* Offset: 0x8C (R/W) Dedicated Cl
uint32_t volatile CKGATENR;                     /* Offset: 0x90 (R/W) Clocks Gated Enable
uint32_t volatile DCKCFGR2;                     /* Offset: 0x94 (R/W) Dedicated Clocks Co
```

```
} RCC_t;
#define RCC ((RCC_t *)0x40023800)


/*
* Data Structure for GPIO port
*/
typedef struct
{
uint32_t volatile MODER; /* Offset: 0x00 (R/W) Mode Register */
uint32_t volatile OTYPER; /* Offset: 0x04 (R/W) Output Type Register */
uint32_t volatile OSPEEDR; /* Offset: 0x08 (R/W) Output Speed Register */
uint32_t volatile PUPDR; /* Offset: 0x0C (R/W) Pull-up/Pull-down Register */
uint32_t volatile IDR; /* Offset: 0x10 (R/W) Input Data Register */
uint32_t volatile ODR; /* Offset: 0x14 (R/W) Output Data Register */
uint32_t volatile BSRR; /* Offset: 0x18 (R/W) Bit Set/Reset Register */
uint32_t volatile LCKR; /* Offset: 0x1C (R/W) Configuration Lock Register */
uint32_t volatile AFRL; /* Offset: 0x20 (R/W) Alternate Function Low Register */
uint32_t volatile AFRH; /* Offset: 0x24 (R/W) Alternate Function High Register */
} GPIO_t;


#define GPIOA ((GPIO_t *)0x40020000)

/* GPIOA is a macro defined to be a pointer to the base address of GPIO port A. ((GPI

#define GPIOB ((GPIO_t *)0x40020400)
#define GPIOC ((GPIO_t *)0x40020800)
#define GPIOD ((GPIO_t *)0x40020C00)
#define GPIOE ((GPIO_t *)0x40021000)
#define GPIOF ((GPIO_t *)0x40021400)
#define GPIOG ((GPIO_t *)0x40021800)
#define GPIOH ((GPIO_t *)0x40021C00)

/* Similarly, the other macros are defined as Port A */
```

4.8 systemClock.c

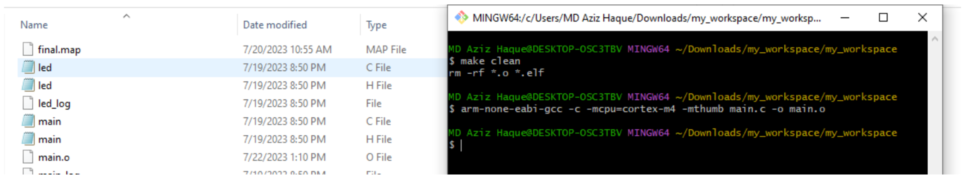
```
#include <stdint.h>
#include"systemClock.h"
#include"stm32f446xx.h"

/* These lines include necessary header files for the code to work. stdint.h provide


/**********System Clock Configuration*******/
void sysClockConfig(void){

/* This line defines the function sysClockConfig() which is used to configure the sy

        uint32_t *pwr_cr = (uint32_t *)0x40007000;
        uint32_t *flash_acr = (uint32_t *)0x40023C00;

        /* These lines declare two pointers (pwr_cr and flash_acr) to uint32_t, whic


        //1.enable HSE and wait for HSE to become ready
        RCC->CR |= (1<<16);
        while(!(RCC->CR & (1<<17)))

 /* These lines enable the High-Speed External (HSE) clock source and wait for it to

        //2.set the POWER ENABLE CLOCK and VOLTAGE REGULATOR
        RCC->APB1ENR |= (1<<28);
        *pwr_cr |= (3<<14);

 /* These lines enable the clock for the PWR peripheral and set the voltage scaling t

        //3.configure the FLASH PREFETCH and the LATENCY related settings
        *flash_acr = (1 << 9) | (1 << 10) | (1 << 8) | (5 << 0);

 /* This line configures the Flash memory access latency and enables the prefetch bu

        //4.configure the PRESCALARS HCLK, PCLK1, PCLK2
        //AHB PR
        RCC->CFGR = 0U;
        //APB1 PR
        RCC->CFGR |= (5<<10);
        //APB2 PR
        RCC->CFGR |= (4<<13);

/* These lines configure the prescalers for HCLK, PCLK1, and PCLK2. RCC->CFGR is the

        //5.configure the main PLL
        RCC->PLLCFGR = (PLL_M<<0) | (PLL_N<<6) | (PLL_P<<16) | (1<<22);

/* This line configures the main PLL (Phase-Locked Loop). The PLLCFGR register contr

        //6.enable the PLL and wait for it to become ready
        RCC->CR |= (1<<24);
        while(!(RCC->CR & (1<<25)));

/* These lines enable the main PLL and wait for it to become ready. Bit 24 is set to

        //7.select the CLOCK SOURCE and wait for it to be set
        RCC->CFGR |= (2<<0);
        while((RCC->CFGR & (3<<2)) != (2<<2));

/* These lines select the PLL as the system clock source and wait for it to be set.

}
```

4.9 systemClock.h

```
#define PLL_M 4

/* This constant is set to 4. It represents the division factor for the main PLL inpu

#define PLL_N 180

/* This constant is set to 180. It represents the multiplication factor for the VCO o

#define PLL_P 0 // PLLP = 2

/*  This constant is set to 0, which corresponds to PLLP = 2. It represents the divis

void sysClockConfig(void);

/*  This is a function prototype for the sysClockConfig() function. It declares the f
```

5 Experimental result

Some Snapshots of the Networking Commands can be seen in the following figures:

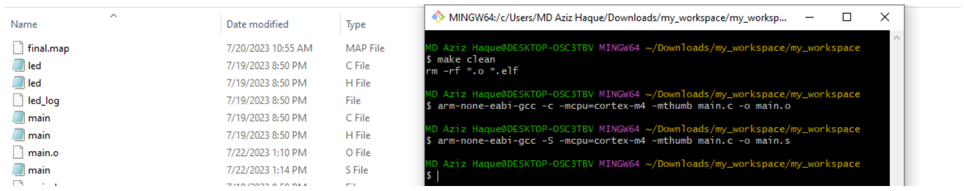


Name	Date modified	Type
final.map	7/20/2023 10:55 AM	MAP File
led	7/19/2023 8:50 PM	C File
led	7/19/2023 8:50 PM	H File
led_log	7/19/2023 8:50 PM	File
main	7/19/2023 8:50 PM	C File
main	7/19/2023 8:50 PM	H File
main.o	7/22/2023 1:10 PM	O File

MINGW64:/c/Users/MD Aziz Haque/Downloads/my_workspace/my_worksp...

```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ make clean
rm -rf *.o *.elf

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb main.c -o main.o

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ |
```

Figure 1: Compile and assemble (not link) main.o

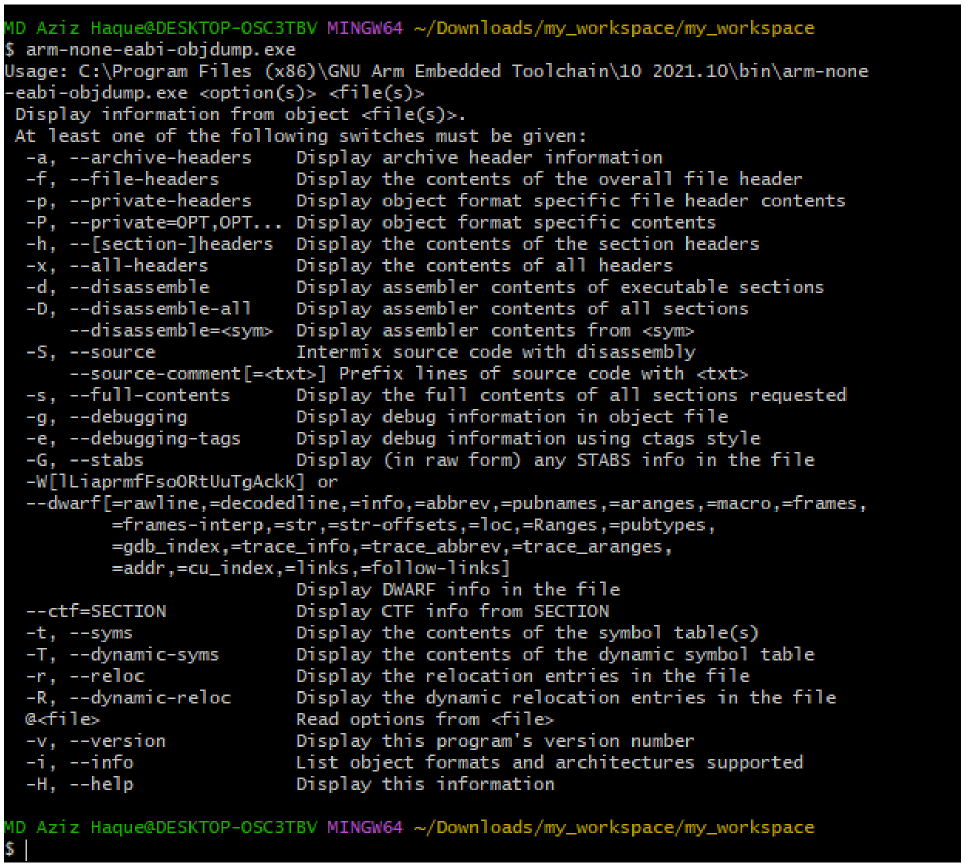


Name	Date modified	Type
final.map	7/20/2023 10:55 AM	MAP File
led	7/19/2023 8:50 PM	C File
led	7/19/2023 8:50 PM	H File
led_log	7/19/2023 8:50 PM	File
main	7/19/2023 8:50 PM	C File
main	7/19/2023 8:50 PM	H File
main.o	7/22/2023 1:10 PM	O File
main	7/22/2023 1:14 PM	S File

MINGW64:/c/Users/MD Aziz Haque/Downloads/my_workspace/my_worksp...

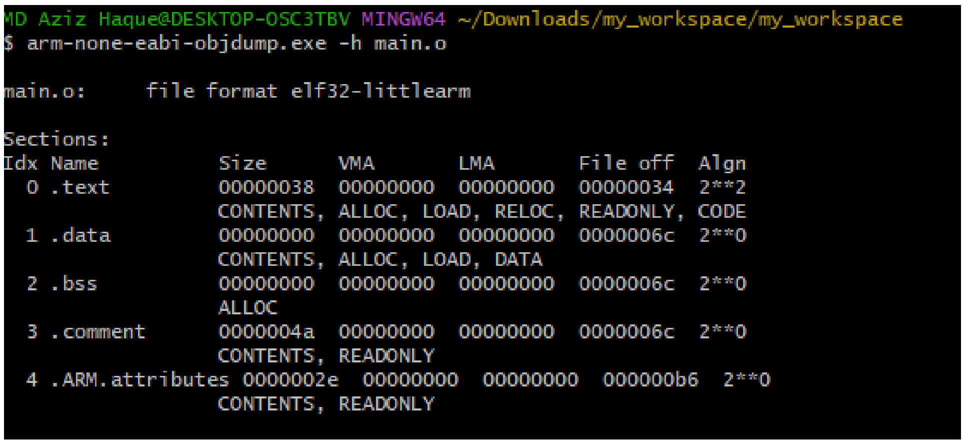
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ make clean
rm -rf *.o *.elf

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb main.c -o main.o

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-gcc -S -mcpu=cortex-m4 -mthumb main.c -o main.s

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ |
```

Figure 2: Generation of Assembly file main.s



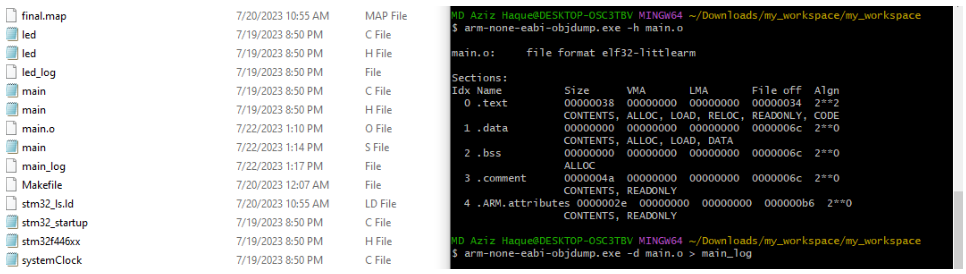
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-objdump.exe
Usage: C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2021.10\bin\arm-none
-eabi-objdump.exe <option(s)> <file(s)>
 Display information from object <file(s)>.
 At least one of the following switches must be given:
  -a, --archive-headers    Display archive header information
  -f, --file-headers       Display the contents of the overall file header
  -p, --private-headers    Display object format specific file header contents
  -P, --private=OPT,OPT... Display object format specific contents
  -h, --[section-]headers  Display the contents of the section headers
  -x, --all-headers        Display the contents of all headers
  -d, --disassemble        Display assembler contents of executable sections
  -D, --disassemble-all    Display assembler contents of all sections
      --disassemble=<sym>  Display assembler contents from <sym>
  -S, --source             Intermix source code with disassembly
      --source-comment[=<txt>] Prefix lines of source code with <txt>
  -s, --full-contents      Display the full contents of all sections requested
  -g, --debugging          Display debug information in object file
  -e, --debugging-tags     Display debug information using ctags style
  -G, --stabs              Display (in raw form) any STABS info in the file
  -W[lLiaprmfFsoORtUuTgAckK] or
  --dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
          =frames-interp,=str,=str-offsets,=loc,=Ranges,=pubtypes,
          =gdb_index,=trace_info,=trace_abbrev,=trace_aranges,
          =addr,=cu_index,=links,=follow-links]
                           Display DWARF info in the file
  --ctf=SECTION            Display CTF info from SECTION
  -t, --syms               Display the contents of the symbol table(s)
  -T, --dynamic-syms       Display the contents of the dynamic symbol table
  -r, --reloc              Display the relocation entries in the file
  -R, --dynamic-reloc      Display the dynamic relocation entries in the file
  @<file>                  Read options from <file>
  -v, --version            Display this program's version number
  -i, --info               List object formats and architectures supported
  -H, --help               Display this information

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ |
```

Figure 3: Display information from the object file



```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000038  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000006c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000006c  2**0
                  ALLOC
  3 .comment      0000004a  00000000  00000000  0000006c  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002e  00000000  00000000  000000b6  2**0
                  CONTENTS, READONLY
```

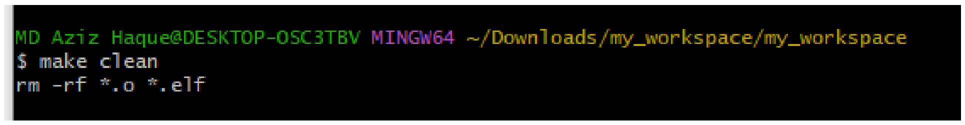
Figure 4: See the new section in main.o



final.map	7/20/2023 10:55 AM	MAP File
led	7/19/2023 8:50 PM	C File
led	7/19/2023 8:50 PM	H File
led_log	7/19/2023 8:50 PM	File
main	7/19/2023 8:50 PM	C File
main	7/19/2023 8:50 PM	H File
main.o	7/22/2023 1:10 PM	O File
main	7/22/2023 1:14 PM	S File
main_log	7/22/2023 1:17 PM	File
Makefile	7/20/2023 12:07 AM	File
stm32_ls.ld	7/20/2023 10:55 AM	LD File
stm32_startup	7/19/2023 8:50 PM	C File
stm32f446xx	7/19/2023 8:50 PM	H File
systemClock	7/19/2023 8:50 PM	C File

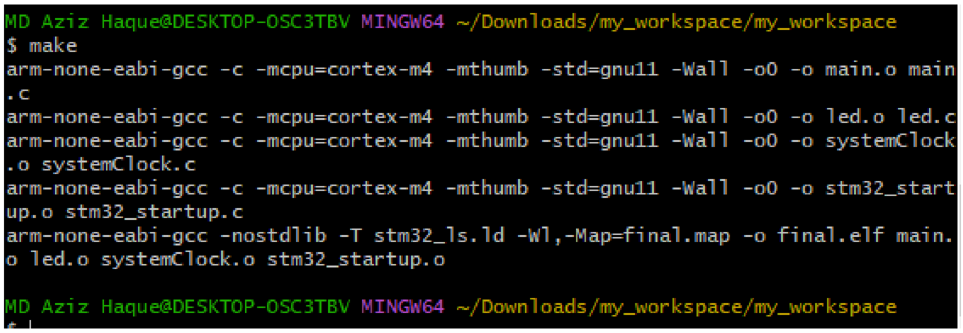
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000038  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000006c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000006c  2**0
                  ALLOC
  3 .comment      0000004a  00000000  00000000  0000006c  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 0000002e  00000000  00000000  000000b6  2**0
                  CONTENTS, READONLY

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-objdump.exe -d main.o > main_log
```

Figure 5: Disassemble main.o to mainlog



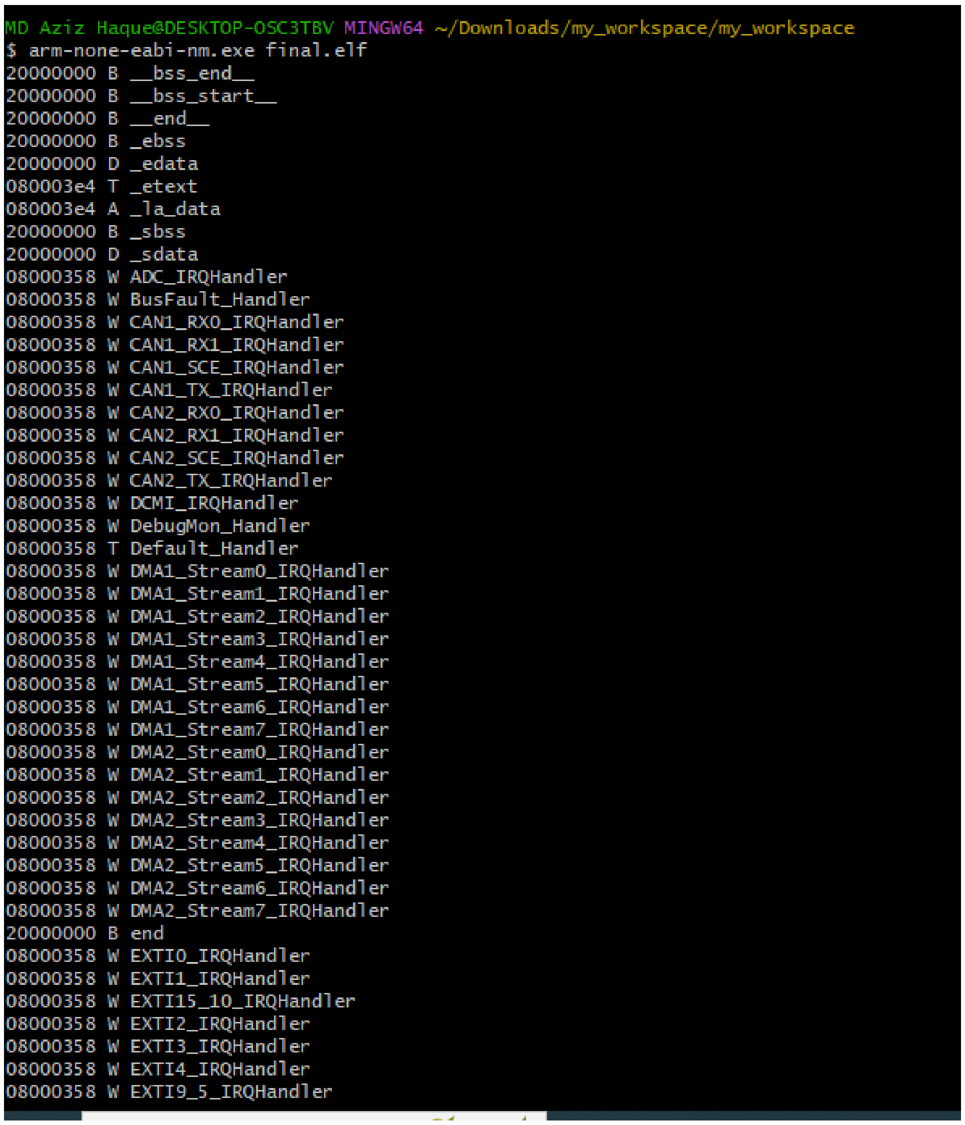
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ make clean
rm -rf *.o *.elf
```

Figure 6: Erase all the object files and executables



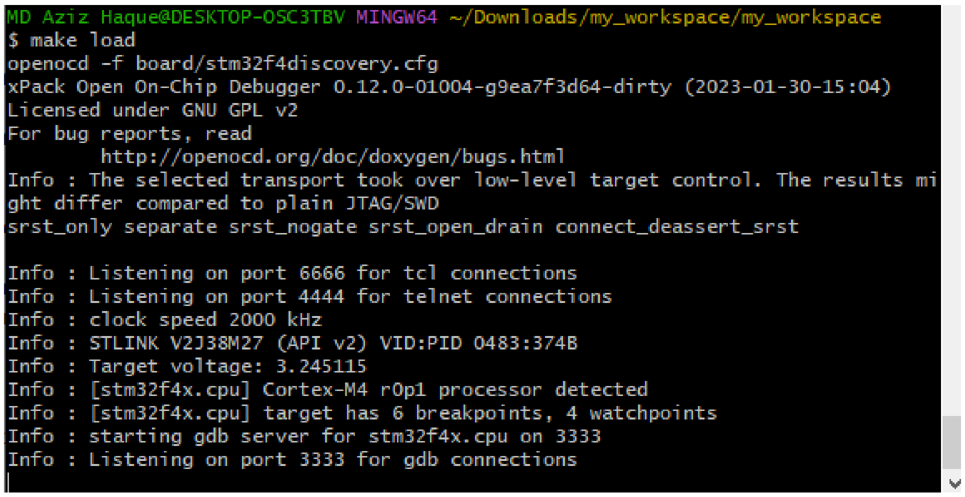
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ make
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -o0 -o main.o main
.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -o0 -o led.o led.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -o0 -o systemClock
.o systemClock.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -o0 -o stm32_start
up.o stm32_startup.c
arm-none-eabi-gcc -nostdlib -T stm32_ls.ld -Wl,-Map=final.map -o final.elf main.
o led.o systemClock.o stm32_startup.o

MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
```

Figure 7: Build all the files



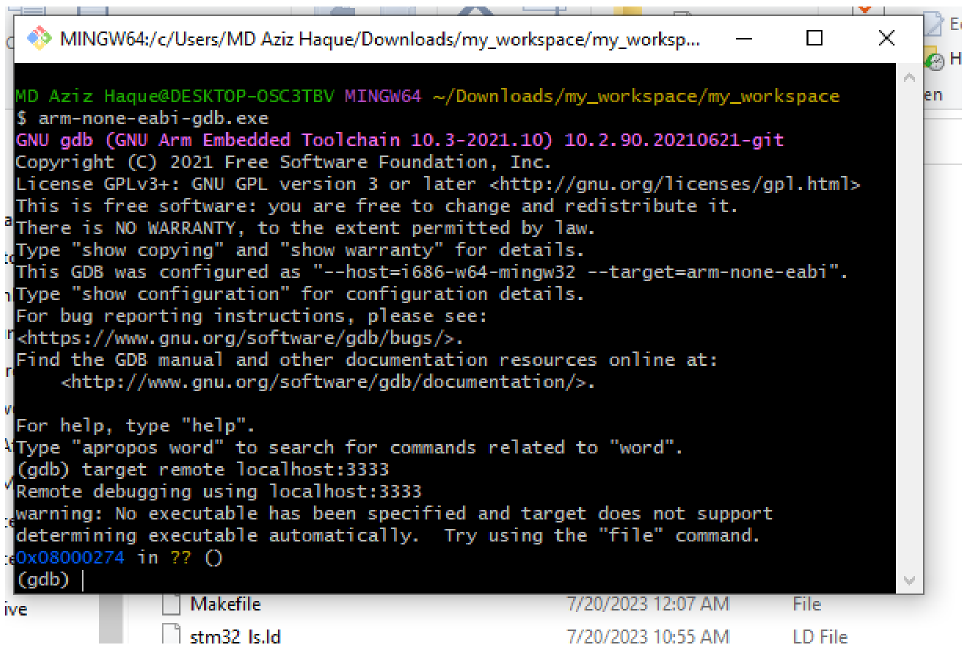
```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-nm.exe final.elf
20000000 B __bss_end__
20000000 B __bss_start__
20000000 B __end__
20000000 B _ebss
20000000 D _edata
080003e4 T _etext
080003e4 A _la_data
20000000 B _sbss
20000000 D _sdata
08000358 W ADC_IRQHandler
08000358 W BusFault_Handler
08000358 W CAN1_RX0_IRQHandler
08000358 W CAN1_RX1_IRQHandler
08000358 W CAN1_SCE_IRQHandler
08000358 W CAN1_TX_IRQHandler
08000358 W CAN2_RX0_IRQHandler
08000358 W CAN2_RX1_IRQHandler
08000358 W CAN2_SCE_IRQHandler
08000358 W CAN2_TX_IRQHandler
08000358 W DCMI_IRQHandler
08000358 W DebugMon_Handler
08000358 T Default_Handler
08000358 W DMA1_Stream0_IRQHandler
08000358 W DMA1_Stream1_IRQHandler
08000358 W DMA1_Stream2_IRQHandler
08000358 W DMA1_Stream3_IRQHandler
08000358 W DMA1_Stream4_IRQHandler
08000358 W DMA1_Stream5_IRQHandler
08000358 W DMA1_Stream6_IRQHandler
08000358 W DMA1_Stream7_IRQHandler
08000358 W DMA2_Stream0_IRQHandler
08000358 W DMA2_Stream1_IRQHandler
08000358 W DMA2_Stream2_IRQHandler
08000358 W DMA2_Stream3_IRQHandler
08000358 W DMA2_Stream4_IRQHandler
08000358 W DMA2_Stream5_IRQHandler
08000358 W DMA2_Stream6_IRQHandler
08000358 W DMA2_Stream7_IRQHandler
20000000 B end
08000358 W EXTI0_IRQHandler
08000358 W EXTI1_IRQHandler
08000358 W EXTI15_10_IRQHandler
08000358 W EXTI2_IRQHandler
08000358 W EXTI3_IRQHandler
08000358 W EXTI4_IRQHandler
08000358 W EXTI9_5_IRQHandler
```

Figure 8: Execution of final.elf



```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ make load
openocd -f board/stm32f4discovery.cfg
xPack Open On-Chip Debugger 0.12.0-01004-g9ea7f3d64-dirty (2023-01-30-15:04)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : The selected transport took over low-level target control. The results mi
ght differ compared to plain JTAG/SWD
srst_only separate srst_nogate srst_open_drain connect_deassert_srst

Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 2000 kHz
Info : STLINK V2J38M27 (API v2) VID:PID 0483:374B
Info : Target voltage: 3.245115
Info : [stm32f4x.cpu] Cortex-M4 r0p1 processor detected
Info : [stm32f4x.cpu] target has 6 breakpoints, 4 watchpoints
Info : starting gdb server for stm32f4x.cpu on 3333
Info : Listening on port 3333 for gdb connections
```

Figure 9: Load the file

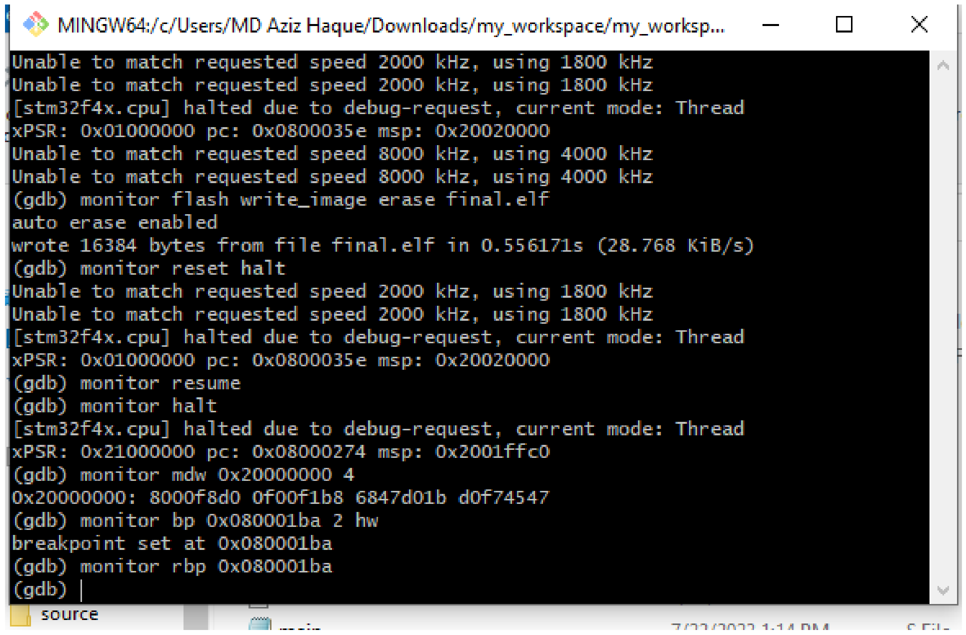


MINGW64:/c/Users/MD Aziz Haque/Downloads/my_workspace/my_worksp...

```
MD Aziz Haque@DESKTOP-OSC3TBV MINGW64 ~/Downloads/my_workspace/my_workspace
$ arm-none-eabi-gdb.exe
GNU gdb (GNU Arm Embedded Toolchain 10.3-2021.10) 10.2.90.20210621-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x08000274 in ?? ()
(gdb) |
```

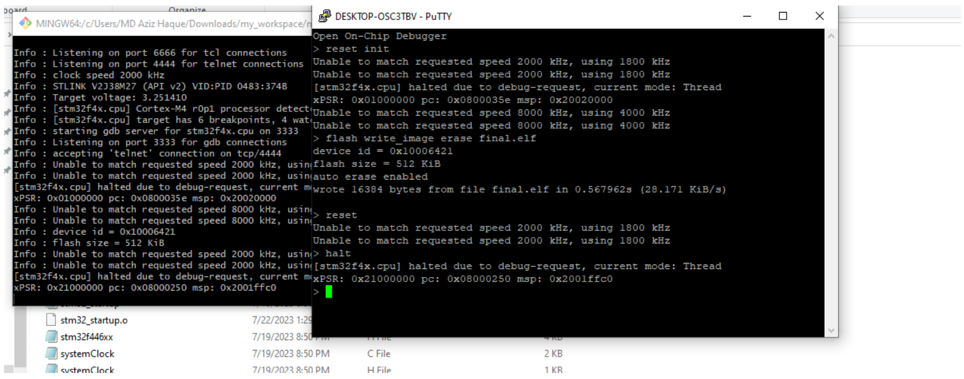
Makefile 7/20/2023 12:07 AM File
stm32 ls.ld 7/20/2023 10:55 AM LD File

Figure 10: Open GDB terminal



MINGW64:/c/Users/MD Aziz Haque/Downloads/my_workspace/my_worksp...

```
Unable to match requested speed 2000 kHz, using 1800 kHz
Unable to match requested speed 2000 kHz, using 1800 kHz
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800035e msp: 0x20020000
Unable to match requested speed 8000 kHz, using 4000 kHz
Unable to match requested speed 8000 kHz, using 4000 kHz
(gdb) monitor flash write_image erase final.elf
auto erase enabled
wrote 16384 bytes from file final.elf in 0.556171s (28.768 KiB/s)
(gdb) monitor reset halt
Unable to match requested speed 2000 kHz, using 1800 kHz
Unable to match requested speed 2000 kHz, using 1800 kHz
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800035e msp: 0x20020000
(gdb) monitor resume
(gdb) monitor halt
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x21000000 pc: 0x08000274 msp: 0x2001ffc0
(gdb) monitor mdw 0x20000000 4
0x20000000: 8000f8d0 0f00f1b8 6847d01b d0f74547
(gdb) monitor bp 0x080001ba 2 hw
breakpoint set at 0x080001ba
(gdb) monitor rbp 0x080001ba
(gdb) |
```



MINGW64:/c/Users/MD Aziz Haque/Downloads/my_workspace/

```
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 2000 kHz
Info : STLINK V2J38M27 (API v2) VID:PID 0483:374B
Info : Target voltage: 3.251410
Info : [stm32f4x.cpu] Cortex-M4 r0p1 processor detect
Info : [stm32f4x.cpu] target has 6 breakpoints, 4 wat
Info : starting gdb server for stm32f4x.cpu on 3333
Info : Listening on port 3333 for gdb connections
Info : accepting 'telnet' connection on tcp/4444
Info : Unable to match requested speed 2000 kHz, usin
Info : Unable to match requested speed 2000 kHz, usin
[stm32f4x.cpu] halted due to debug-request, current m
xPSR: 0x01000000 pc: 0x0800035e msp: 0x20020000
Info : Unable to match requested speed 8000 kHz, usin
Info : Unable to match requested speed 8000 kHz, usin
Info : device id = 0x10006421
Info : flash size = 512 KiB
Info : Unable to match requested speed 2000 kHz, usin
Info : Unable to match requested speed 2000 kHz, usin
[stm32f4x.cpu] halted due to debug-request, current m
xPSR: 0x21000000 pc: 0x08000250 msp: 0x2001ffc0
```

DESKTOP-OSC3TBV - PuTTY

```
Open On-Chip Debugger
> reset init
Unable to match requested speed 2000 kHz, using 1800 kHz
Unable to match requested speed 2000 kHz, using 1800 kHz
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800035e msp: 0x20020000
Unable to match requested speed 8000 kHz, using 4000 kHz
Unable to match requested speed 8000 kHz, using 4000 kHz
> flash write_image erase final.elf
device id = 0x10006421
flash size = 512 KiB
auto erase enabled
wrote 16384 bytes from file final.elf in 0.567962s (28.171 KiB/s)

> reset
Unable to match requested speed 2000 kHz, using 1800 kHz
Unable to match requested speed 2000 kHz, using 1800 kHz
> halt
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x21000000 pc: 0x08000250 msp: 0x2001ffc0
>
```

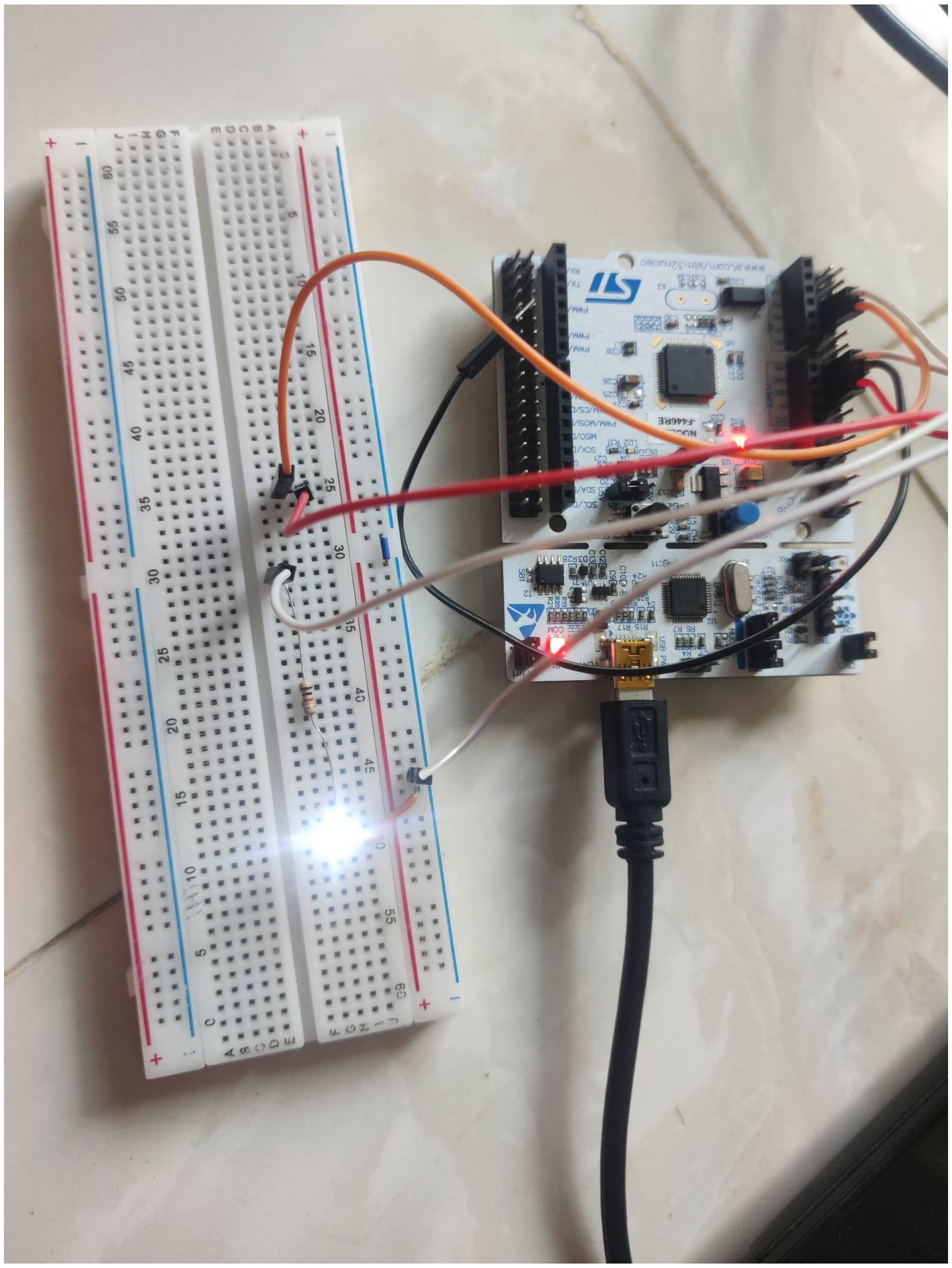
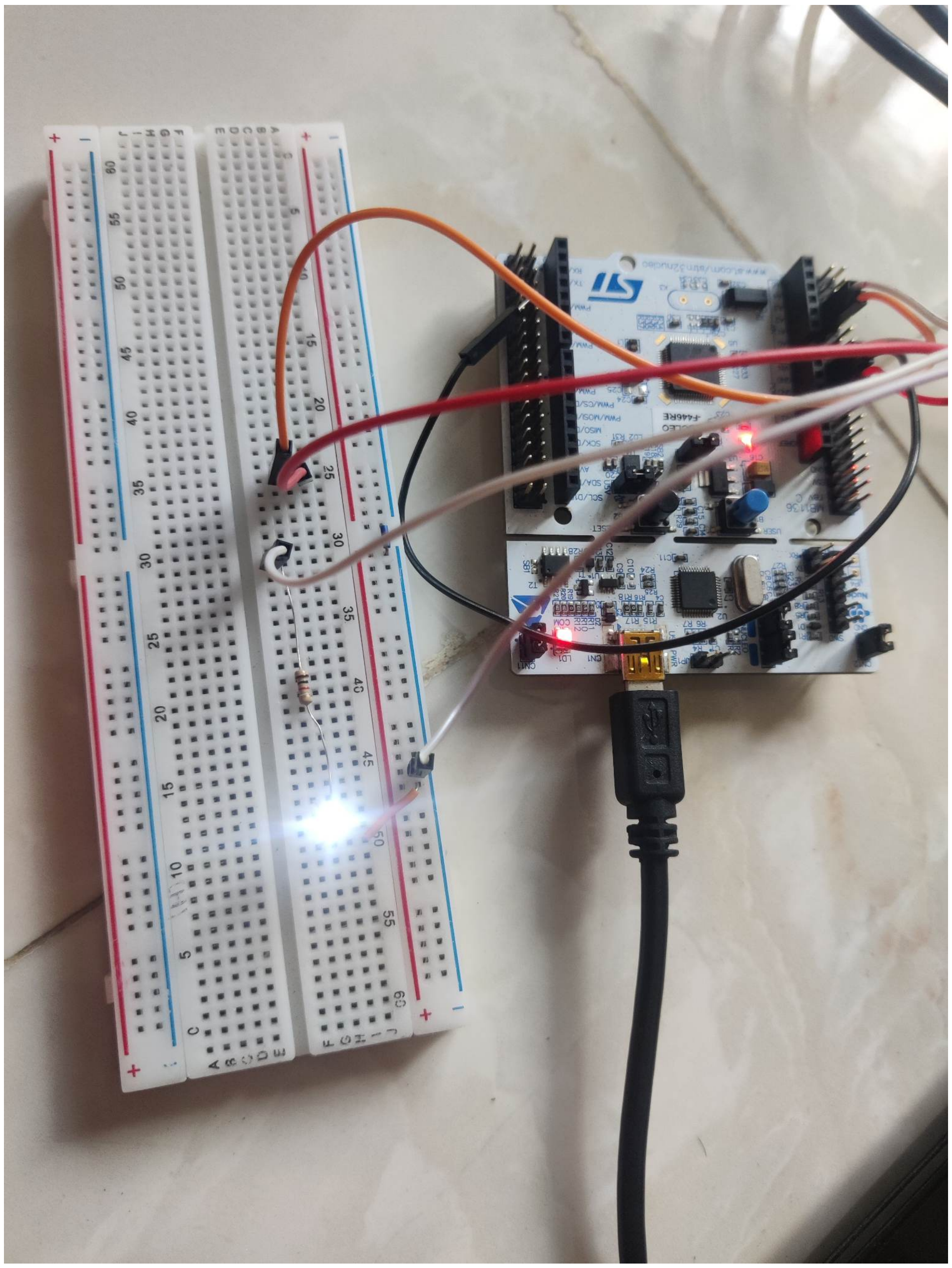


Figure 11: LED on in the brea board



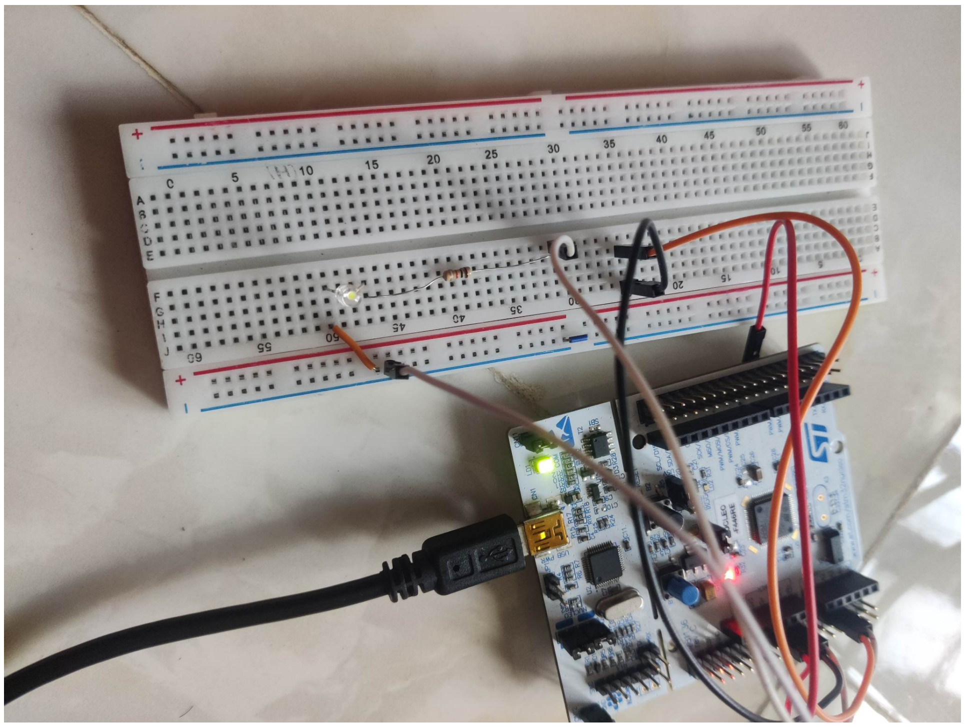


Figure 12: LED off in the bread board

6 Experience

1. We had to follow video tutorial for Understanding bare Metal Programming for Arm Embedded Syatem
2. We had to install multiple executable files to run the commands.
3. We had to install PuTTY
4. We used the knowledge we've earned in the Microcontroller Lab.

References

[1] YouTube : https://www.youtube.com/watch?v=qWqlkCLmZoE&list=
PLERTijJOmYrDiiWd10iRHY0VRHdJwUH4g&index=1

[2] Blog : https://www.googleadservices.com/pagead/aclk?
sa=L&ai=DChcSEwjK7vG5_6WAAxVngksFHQhBDY0YABAAGgJzZg&
ohost=www.google.com&cid=CAESbOD2nwGzoX2JANKZwNWI_
p-wELsBpTsqmxzz8Mm7lewkc5NB7WM4nyxYEBgaJVtJpKZcZoSM14impGwnlHkPd2Xd7Abckay13TMA7ZG
Z7K8d5NQFMYjdja0osBauWZ4j5tWwyCfeg&sig=AOD64_
2Tf2UKmGlfRYTJybf4UqikkTPJyw&q&adurl&ved=2ahUKEwiugO25_
6WAAxUJd2wGHTwhD74Q0Qx6BAgIEAE