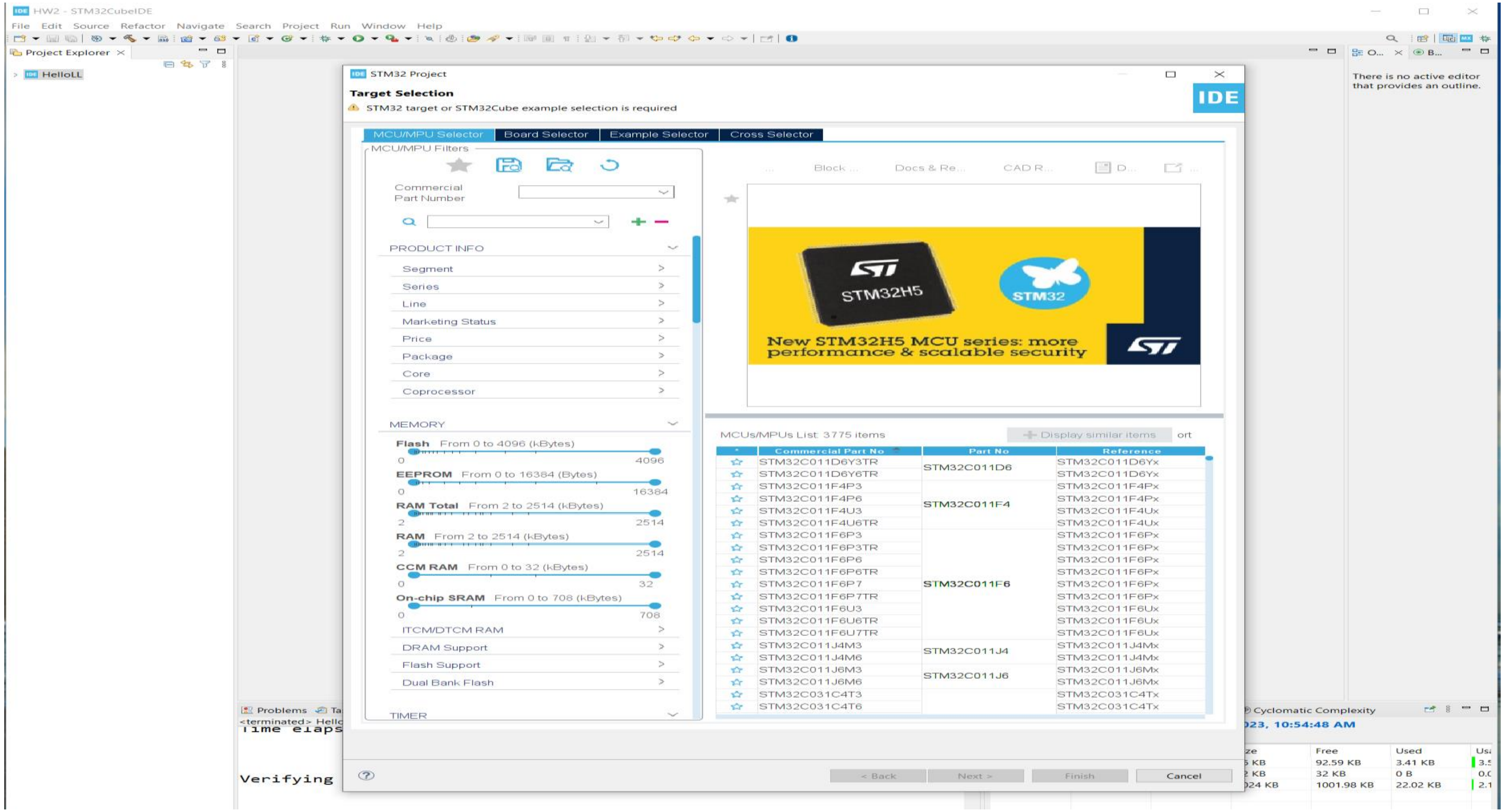
UCSD Embedded C Assignment 3

By

Hsuankai Chang

hsuankac@umich.edu

Step 1. Startup STM32CubeIDE and create new STM32 project



Target Selection

⚠ STM32 target or STM32Cube example selection is required

MCU/MPU Filters

Commercial Part Number

PRODUCT INFO

- Segment >
- Series >
- Line >
- Marketing Status >
- Price >
- Package >
- Core >
- Coprocessor >

MEMORY

- **Flash** From 0 to 4096 (kBytes) — 0 … 4096
- **EEPROM** From 0 to 16384 (Bytes) — 0 … 16384
- **RAM Total** From 2 to 2514 (kBytes) — 2 … 2514
- **RAM** From 2 to 2514 (kBytes) — 2 … 2514
- **CCM RAM** From 0 to 32 (kBytes) — 0 … 32
- **On-chip SRAM** From 0 to 708 (kBytes) — 0 … 708
- ITCM/DTCM RAM >
- DRAM Support >
- Flash Support >
- Dual Bank Flash >

TIMER

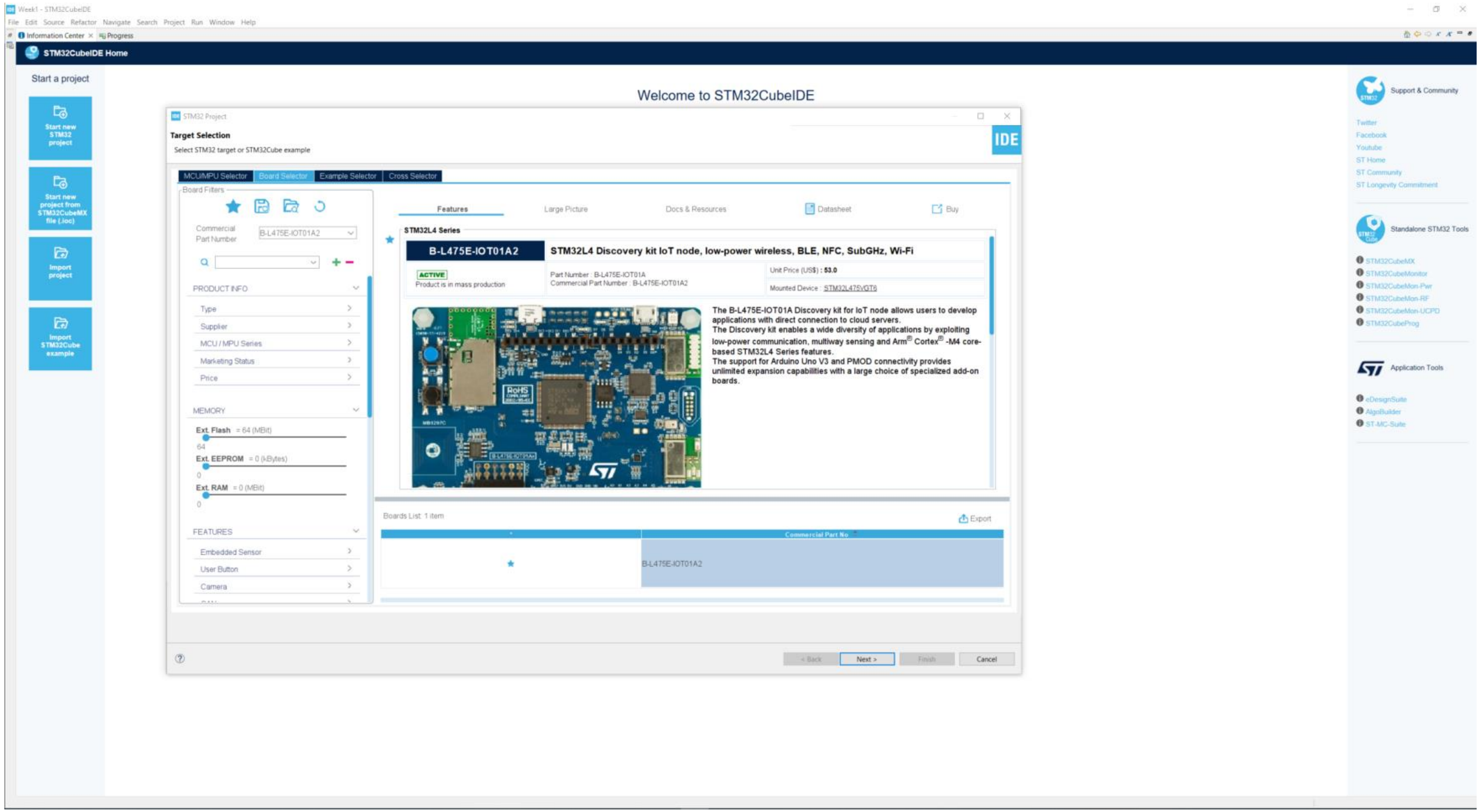
MCUs/MPUs List: 3775 items

*	Commercial Part No	Part No	Reference
☆	STM32C011D6Y3TR	STM32C011D6	STM32C011D6Yx
☆	STM32C011D6Y6TR		STM32C011D6Yx
☆	STM32C011F4P3	STM32C011F4	STM32C011F4Px
☆	STM32C011F4P6		STM32C011F4Px
☆	STM32C011F4U3		STM32C011F4Ux
☆	STM32C011F4U6TR		STM32C011F4Ux
☆	STM32C011F6P3	STM32C011F6	STM32C011F6Px
☆	STM32C011F6P3TR		STM32C011F6Px
☆	STM32C011F6P6		STM32C011F6Px
☆	STM32C011F6P6TR		STM32C011F6Px
☆	STM32C011F6P7		STM32C011F6Px
☆	STM32C011F6P7TR		STM32C011F6Px
☆	STM32C011F6U3		STM32C011F6Ux
☆	STM32C011F6U6TR		STM32C011F6Ux
☆	STM32C011F6U7TR		STM32C011F6Ux
☆	STM32C011J4M3	STM32C011J4	STM32C011J4Mx
☆	STM32C011J4M6		STM32C011J4Mx
☆	STM32C011J6M3	STM32C011J6	STM32C011J6Mx
☆	STM32C011J6M6		STM32C011J6Mx
☆	STM32C031C4T3		STM32C031C4Tx
☆	STM32C031C4T6		STM32C031C4Tx

```
<terminated> Hello
Time elaps
Verifying
```

```
023, 10:54:48 AM
ze      Free         Used       Usa
5 KB    92.59 KB     3.41 KB    3.5
2 KB    32 KB        0 B        0.0
024 KB  1001.98 KB   22.02 KB   2.1
```

Step 2. Access board selector and type in the board you use, click Next



Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: B-L475E-IOT01A2

Ext. Flash = 64 (MBit)

Ext. EEPROM = 0 (kBytes)

Ext. RAM = 0 (MBit)

STM32L4 Series

B-L475E-IOT01A2 STM32L4 Discovery kit IoT node, low-power wireless, BLE, NFC, SubGHz, Wi-Fi

ACTIVE
Product is in mass production

Part Number : B-L475E-IOT01A
Commercial Part Number : B-L475E-IOT01A2

Unit Price (US$) : **53.0**

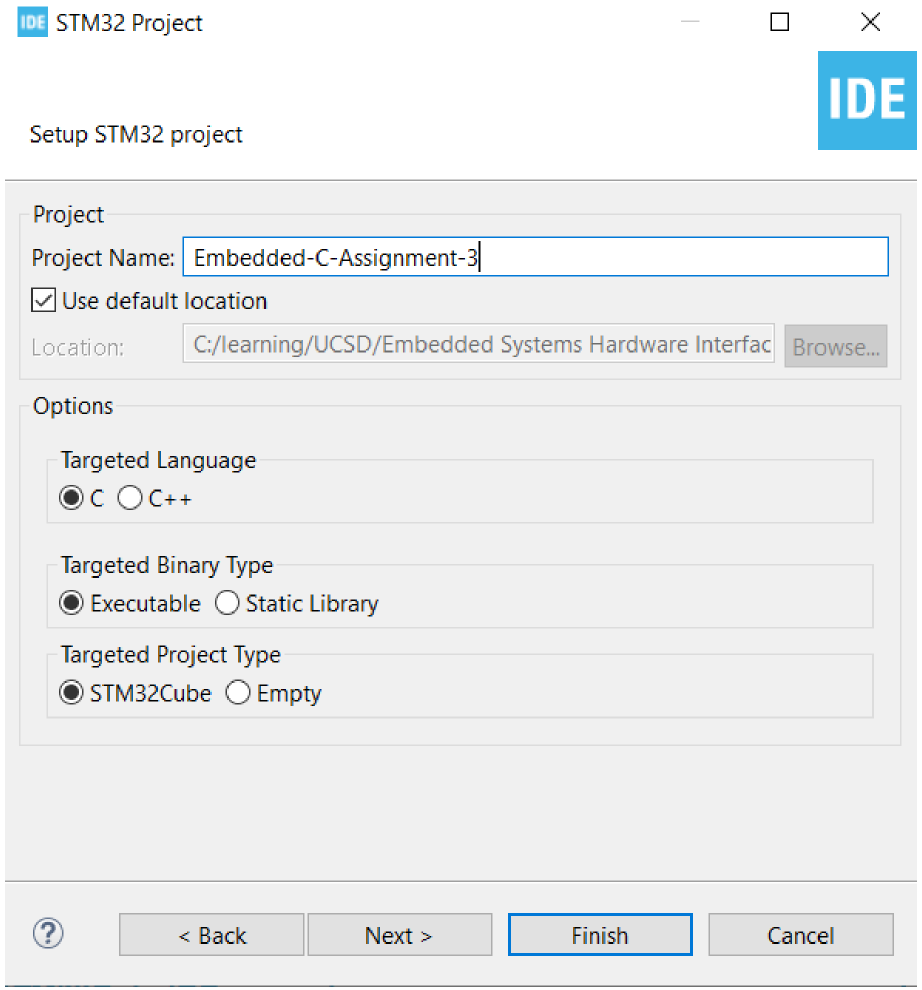
Mounted Device : STM32L475VGT6

The B-L475E-IOT01A Discovery kit for IoT node allows users to develop applications with direct connection to cloud servers.
The Discovery kit enables a wide diversity of applications by exploiting low-power communication, multiway sensing and Arm® Cortex® -M4 core-based STM32L4 Series features.
The support for Arduino Uno V3 and PMOD connectivity provides unlimited expansion capabilities with a large choice of specialized add-on boards.

Boards List: 1 item

*	Commercial Part No
★	B-L475E-IOT01A2

Step 3. Enter the project name then click Next



STM32 Project

Setup STM32 project

Project

Project Name: Embedded-C-Assignment-3

☑ Use default location

Location: C:/learning/UCSD/Embedded Systems Hardware Interfac

Options

Targeted Language

● C ○ C++

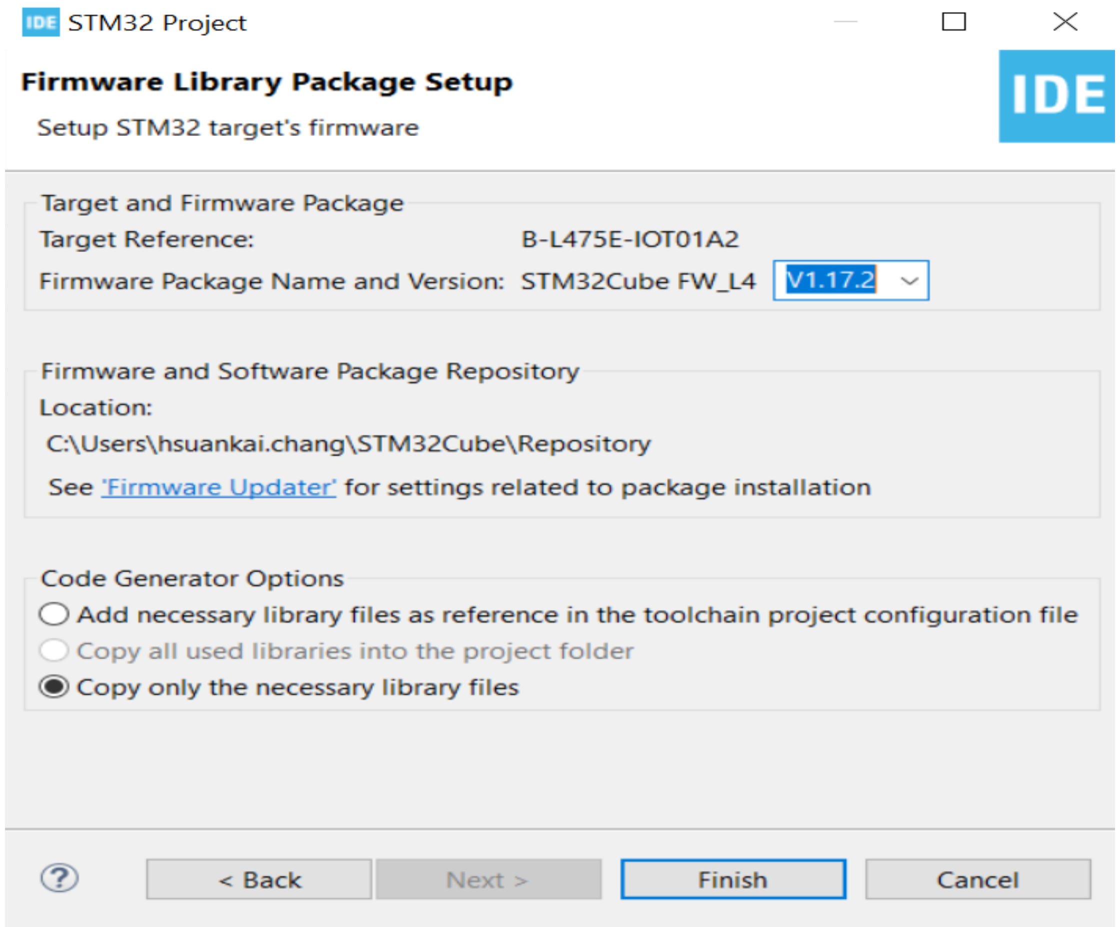
Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

Step 4. See the firmware package name and version



Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: B-L475E-IOT01A2

Firmware Package Name and Version: STM32Cube FW_L4 V1.17.2

Firmware and Software Package Repository

Location:

C:\Users\hsuankai.chang\STM32Cube\Repository

See 'Firmware Updater' for settings related to package installation

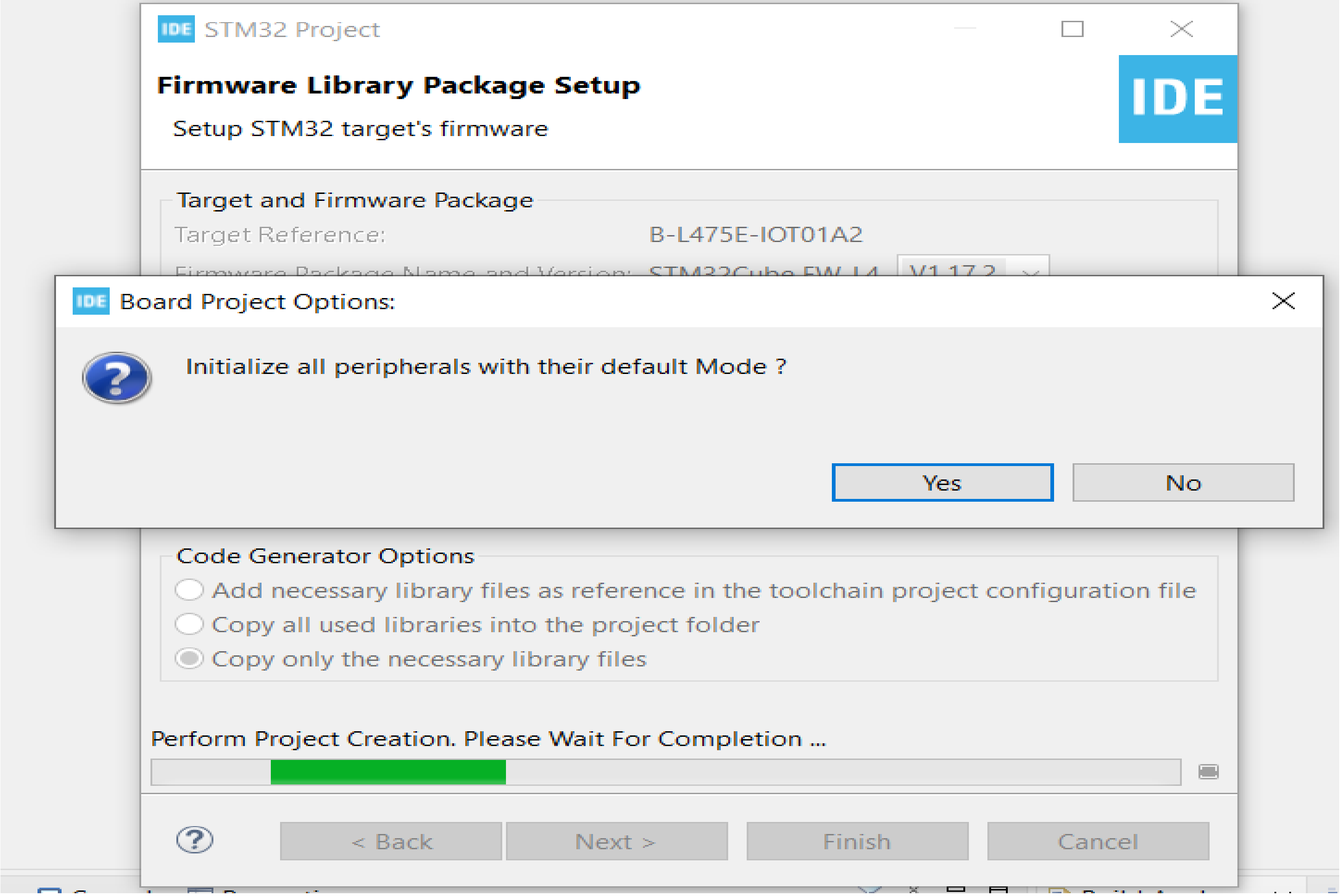
Code Generator Options

○ Add necessary library files as reference in the toolchain project configuration file

○ Copy all used libraries into the project folder

● Copy only the necessary library files

Step 5. Click yes to initialize all peripherals to default



STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: B-L475E-IOT01A2

Firmware Package Name and Version: STM32Cube FW_L4 V1.17.2

Board Project Options:

Initialize all peripherals with their default Mode ?

Code Generator Options

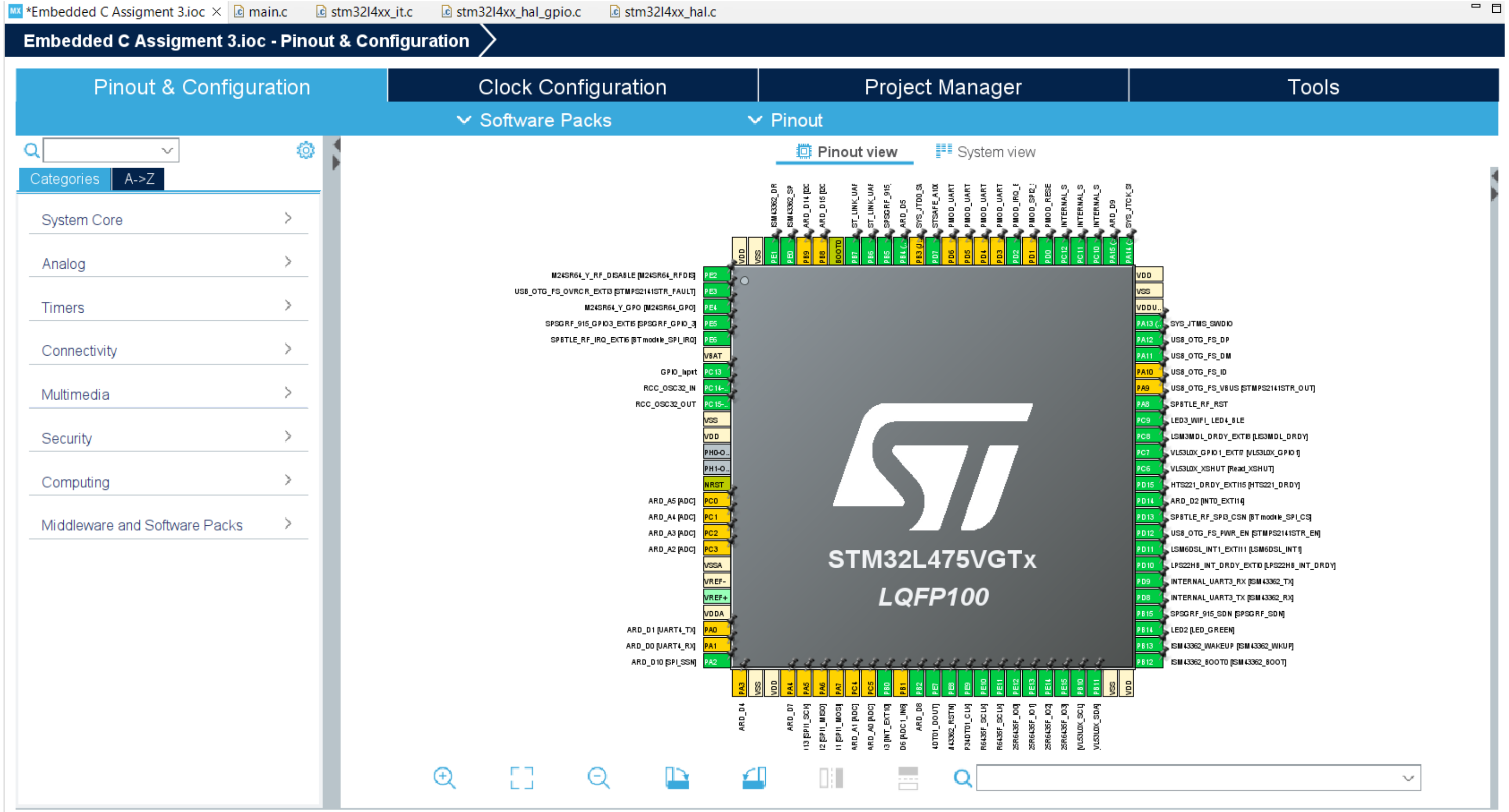
○ Add necessary library files as reference in the toolchain project configuration file

○ Copy all used libraries into the project folder

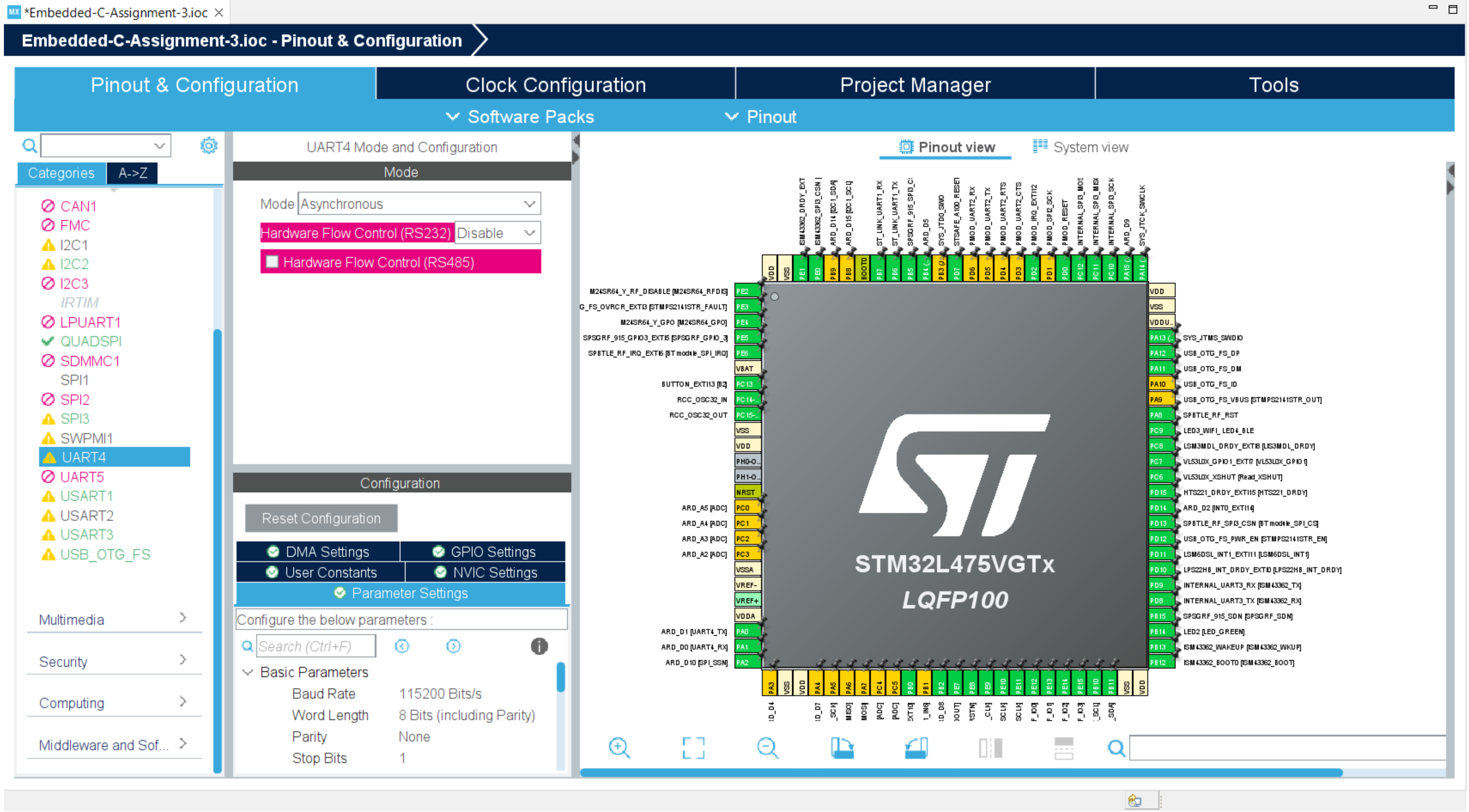
● Copy only the necessary library files

Perform Project Creation. Please Wait For Completion ...

Step 6. When in .ioc file, click Pinout & Configurations


*Embedded C Assigment 3.ioc
main.c
stm32l4xx_it.c
stm32l4xx_hal_gpio.c
stm32l4xx_hal.c
Embedded C Assigment 3.ioc - Pinout & Configuration
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Software Packs
Pinout
Categories
A->Z
System Core
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware and Software Packs
Pinout view
System view
STM32L475VGTx
LQFP100
M24SR64_Y_RF_DISABLE [M24SR64_RFDIS]
USB_OTG_FS_OVRCR_EXTI3 [STMPS2141STR_FAULT]
M24SR64_Y_GPO [M24SR64_GPO]
SPSGRF_915_GPIO3_EXTI5 [SPSGRF_GPIO_3]
SPBTLE_RF_IRQ_EXTI6 [BT module_SPI_IRQ]
GPIO_Input
RCC_OSC32_IN
RCC_OSC32_OUT
ARD_A5 [ADC]
ARD_A4 [ADC]
ARD_A3 [ADC]
ARD_A2 [ADC]
ARD_D1 [UART4_TX]
ARD_D0 [UART4_RX]
ARD_D10 [SPI_SSN]
SYS_JTMS_SWDIO
USB_OTG_FS_DP
USB_OTG_FS_DM
USB_OTG_FS_ID
USB_OTG_FS_VBUS [STMPS2141STR_OUT]
SPBTLE_RF_RST
LED3_WIFI_ LED4_BLE
LSM3MDL_DRDY_EXTI8 [LIS3MDL_DRDY]
VL53L0X_GPIO1_EXTI7 [VL53L0X_GPIO1]
VL53L0X_XSHUT [Read_XSHUT]
HTS221_DRDY_EXTI15 [HTS221_DRDY]
ARD_D2 [INT0_EXTI14]
SPBTLE_RF_SPI3_CSN [BT module_SPI_CS]
USB_OTG_FS_PWR_EN [STMPS2141STR_EN]
LSM6DSL_INT1_EXTI11 [LSM6DSL_INT1]
LPS22HB_INT_DRDY_EXTI0 [LPS22HB_INT_DRDY]
INTERNAL_UART3_RX [ISM43362_TX]
INTERNAL_UART3_TX [ISM43362_RX]
SPSGRF_915_SDN [SPSGRF_SDN]
LED2 [LED_GREEN]
ISM43362_WAKEUP [ISM43362_WKUP]
ISM43362_BOOT0 [ISM43362_BOOT]
ARD_D4
ARD_D7

Step 7. User Story 1: Create a CLI (Command Line Interface) on UART1 that prompts you to enter a 1 for polling, a 2 for interrupt, and a 3 for DMA. Double check we have enabled USART1 and UART4



*Embedded-C-Assignment-3.ioc

Embedded-C-Assignment-3.ioc - Pinout & Configuration

UART4 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

□ Hardware Flow Control (RS485)

Configuration

Configure the below parameters :

Basic Parameters

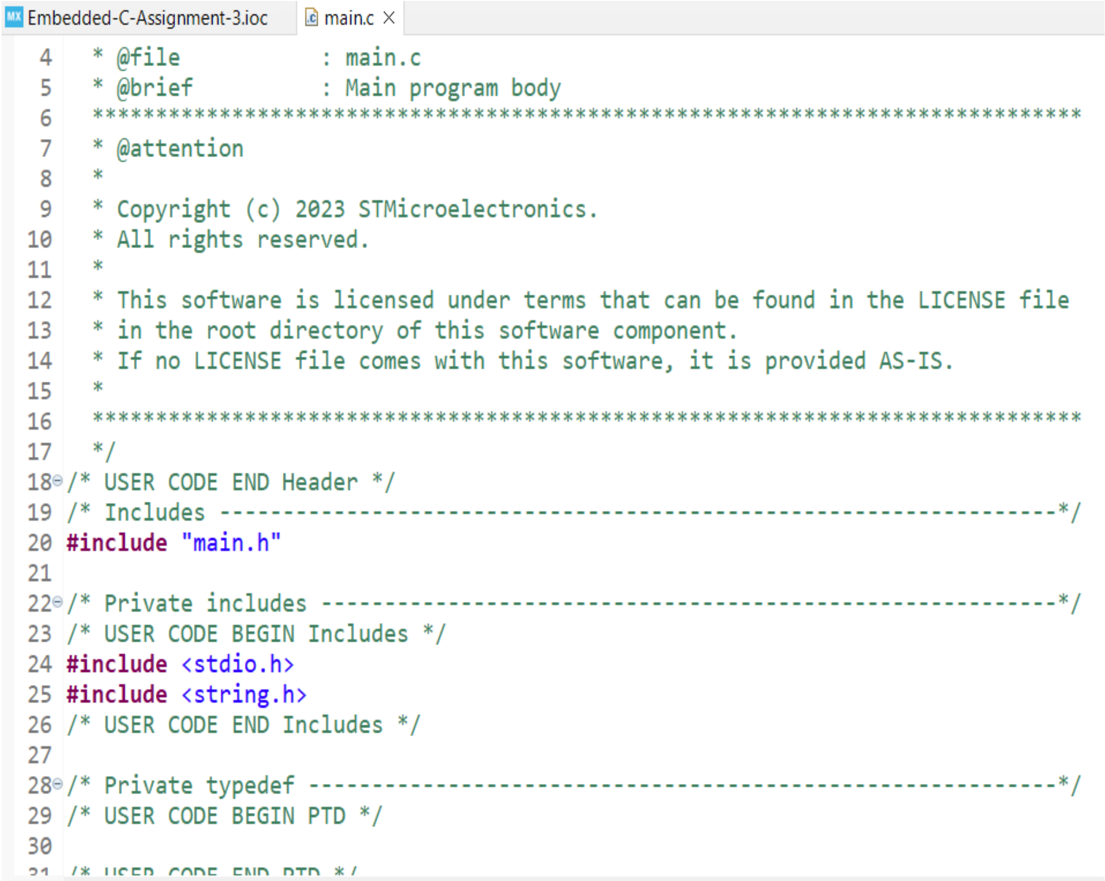
Baud Rate: 115200 Bits/s

Word Length: 8 Bits (including Parity)

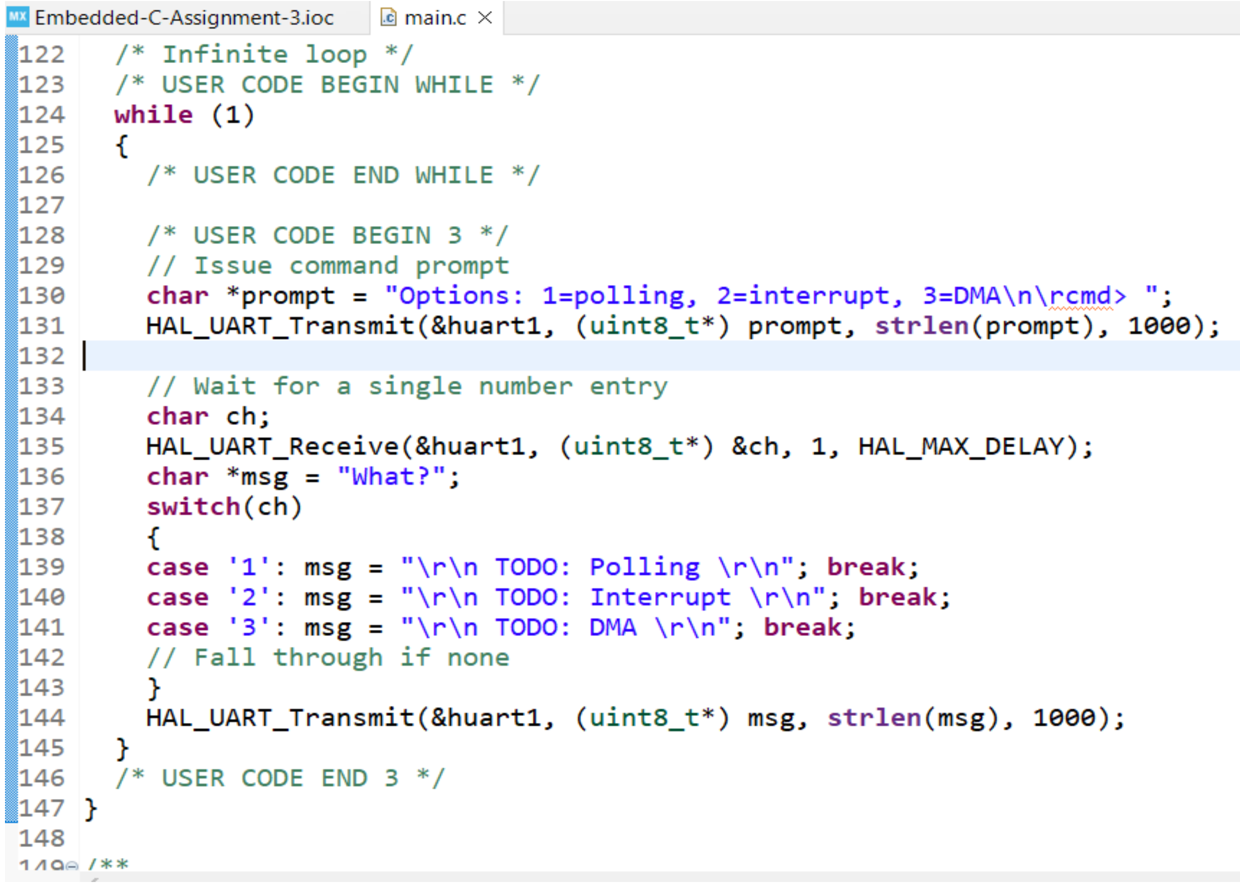
Parity: None

Stop Bits: 1

Step 8. User Story 1: Write the terminal code in main.c file

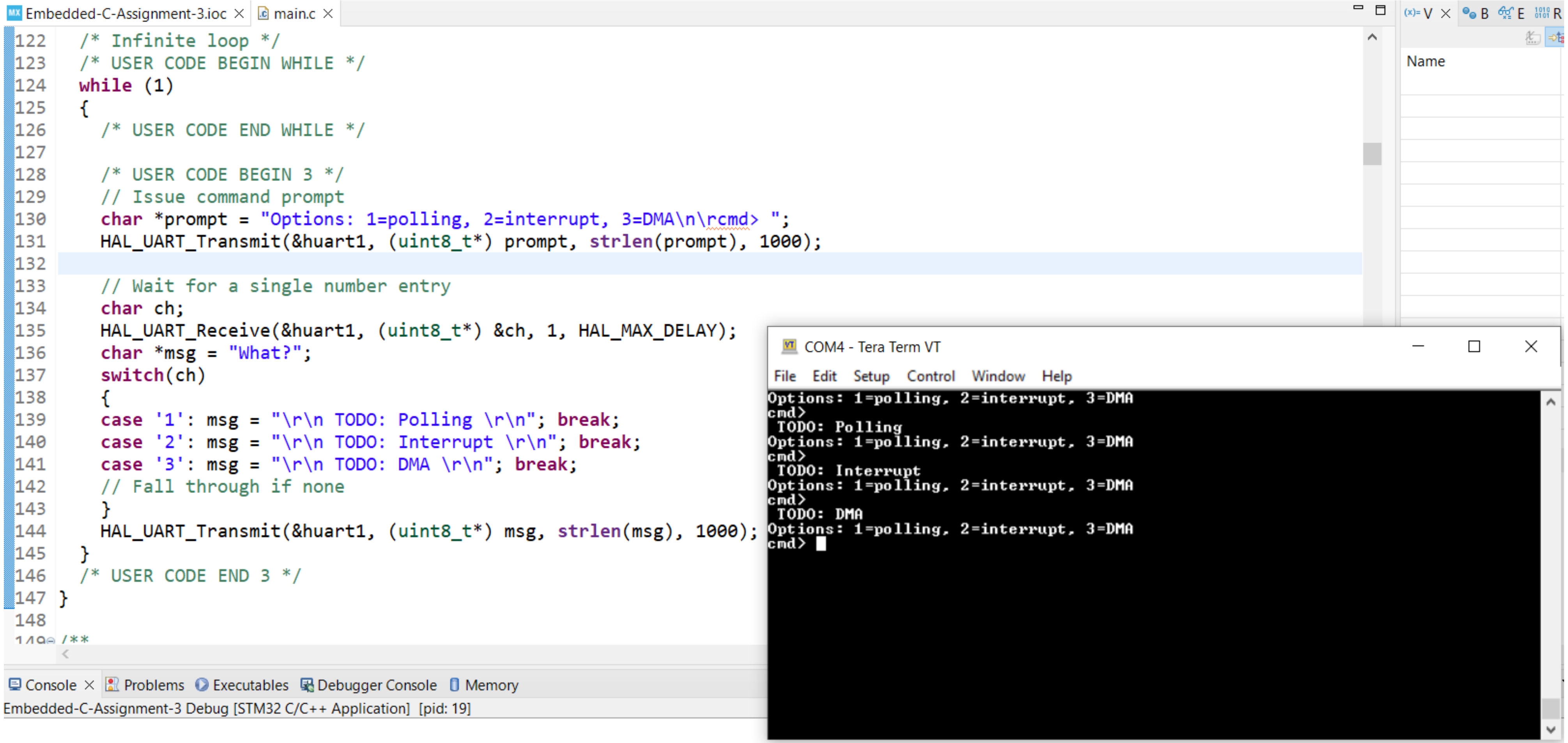


```c
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```



```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; break;
    case '3': msg = "\r\n TODO: DMA \r\n"; break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}

/**
```

Step 9. User Story 1: Build and run in debug mode. Open tera term and test is successful

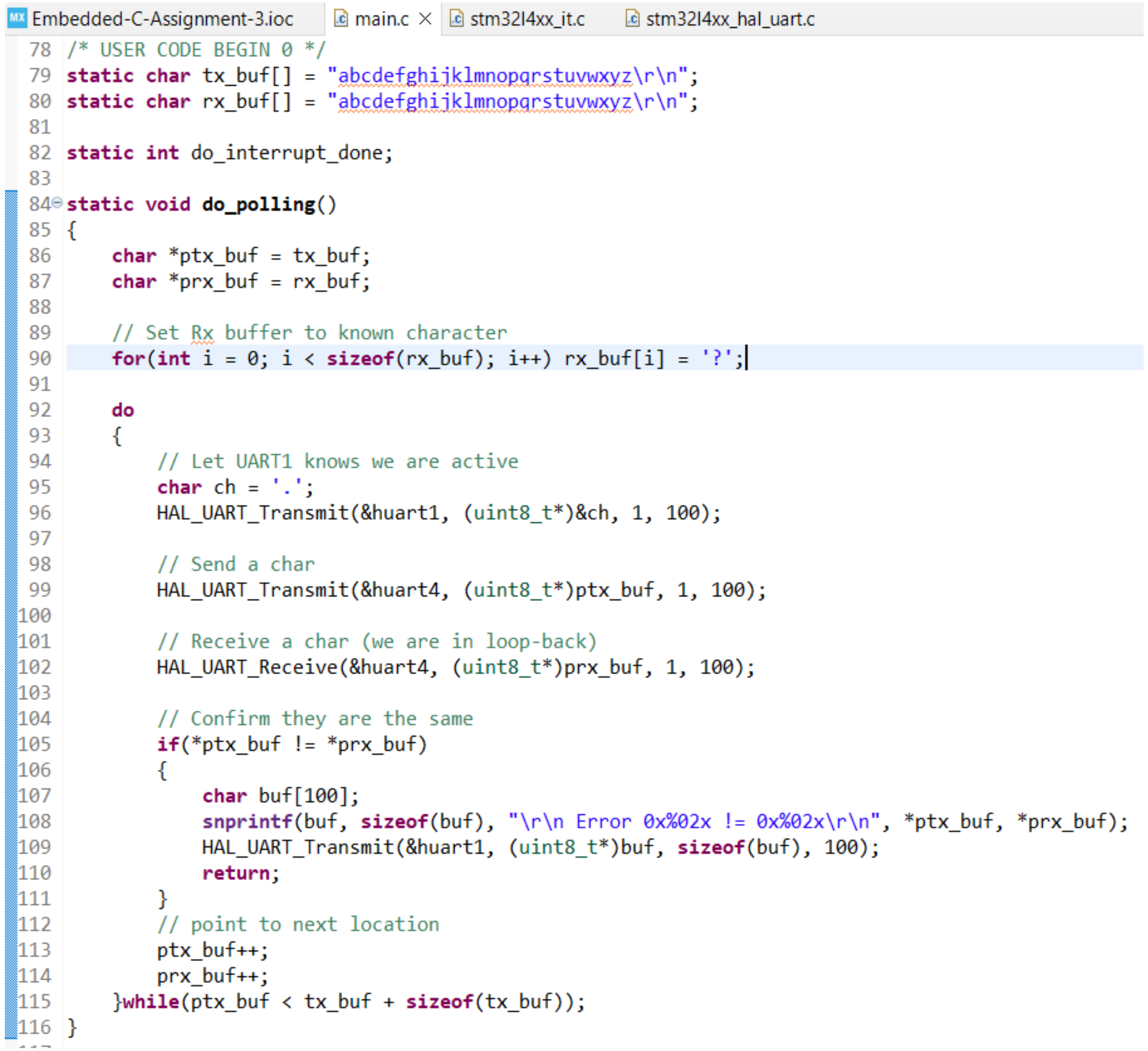


```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; break;
    case '3': msg = "\r\n TODO: DMA \r\n"; break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}

/**
```

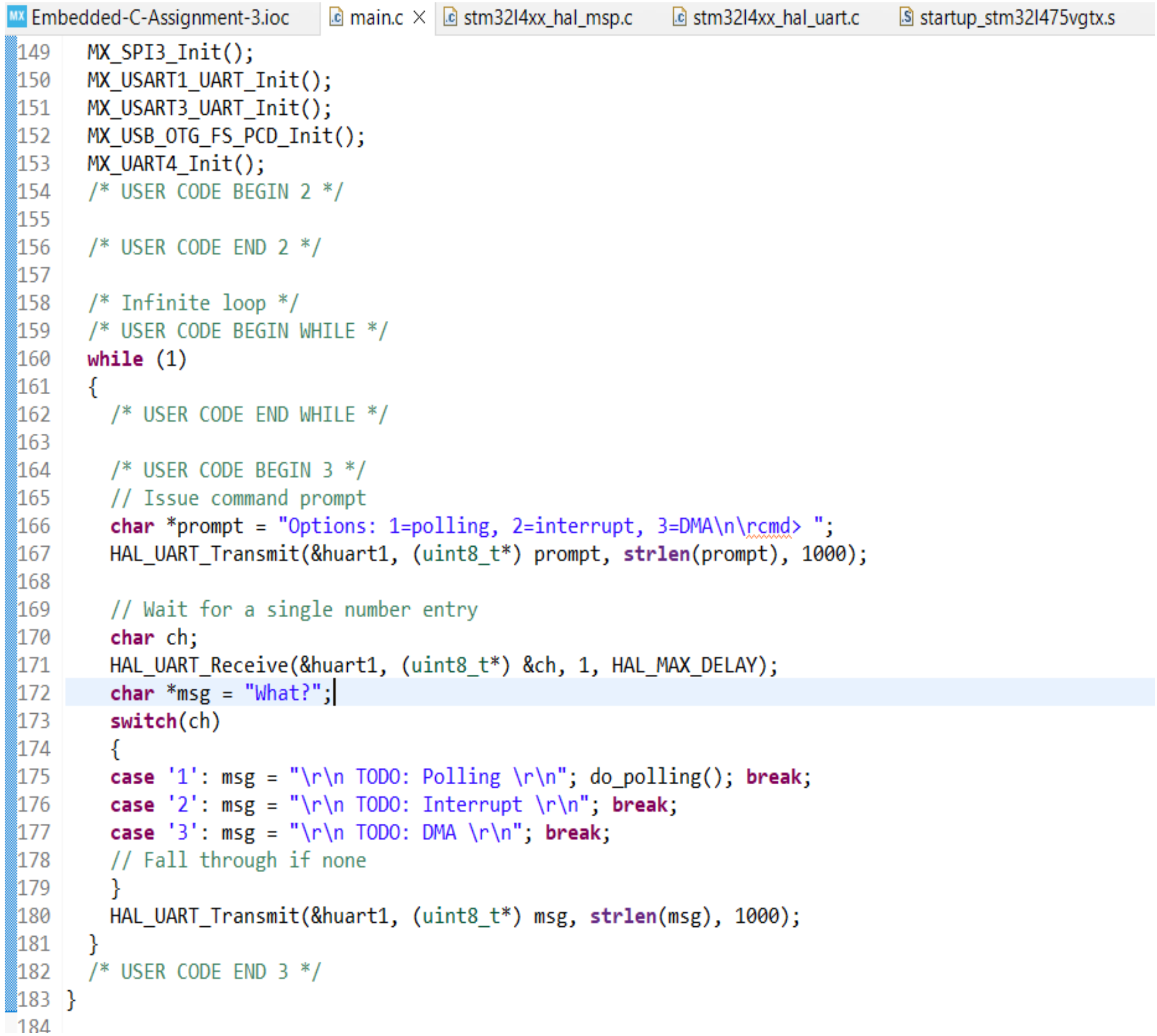
```
Options: 1=polling, 2=interrupt, 3=DMA
cmd>
 TODO: Polling
Options: 1=polling, 2=interrupt, 3=DMA
cmd>
 TODO: Interrupt
Options: 1=polling, 2=interrupt, 3=DMA
cmd>
 TODO: DMA
Options: 1=polling, 2=interrupt, 3=DMA
cmd>
```

```
Embedded-C-Assignment-3 Debug [STM32 C/C++ Application]  [pid: 19]

Download verified successfully
```

Step 10. User Story 2: Polling. When the user selects 1, use polling to transmit and receive the buffer. Write the code in main.c file

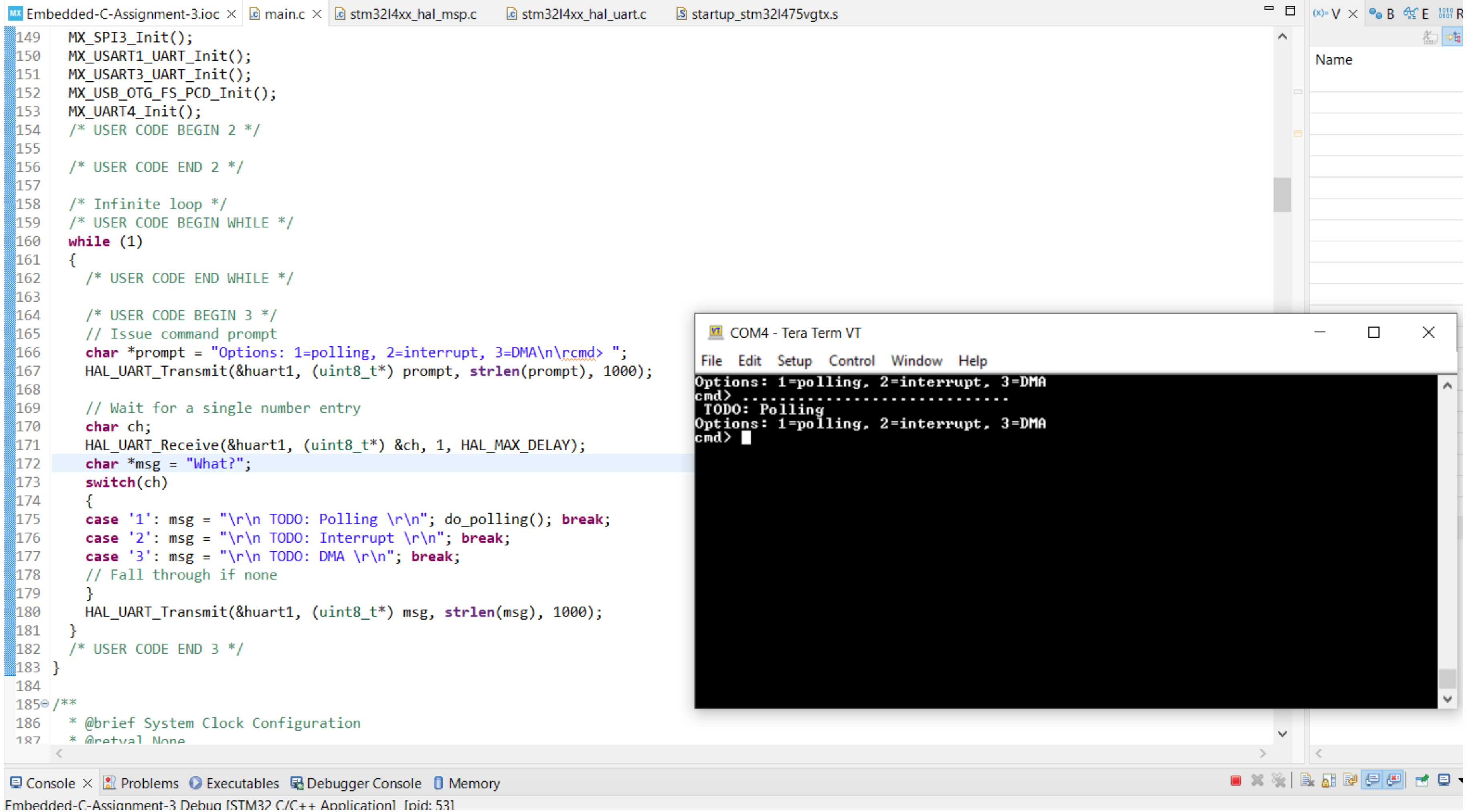


```c
/* USER CODE BEGIN 0 */
static char tx_buf[] = "abcdefghijklmnopqrstuvwxyz\r\n";
static char rx_buf[] = "abcdefghijklmnopqrstuvwxyz\r\n";

static int do_interrupt_done;

static void do_polling()
{
    char *ptx_buf = tx_buf;
    char *prx_buf = rx_buf;

    // Set Rx buffer to known character
    for(int i = 0; i < sizeof(rx_buf); i++) rx_buf[i] = '?';

    do
    {
        // Let UART1 knows we are active
        char ch = '.';
        HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

        // Send a char
        HAL_UART_Transmit(&huart4, (uint8_t*)ptx_buf, 1, 100);

        // Receive a char (we are in loop-back)
        HAL_UART_Receive(&huart4, (uint8_t*)prx_buf, 1, 100);

        // Confirm they are the same
        if(*ptx_buf != *prx_buf)
        {
            char buf[100];
            snprintf(buf, sizeof(buf), "\r\n Error 0x%02x != 0x%02x\r\n", *ptx_buf, *prx_buf);
            HAL_UART_Transmit(&huart1, (uint8_t*)buf, sizeof(buf), 100);
            return;
        }
        // point to next location
        ptx_buf++;
        prx_buf++;
    }while(ptx_buf < tx_buf + sizeof(tx_buf));
}
```



```c
  MX_SPI3_Init();
  MX_USART1_UART_Init();
  MX_USART3_UART_Init();
  MX_USB_OTG_FS_PCD_Init();
  MX_UART4_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; do_polling(); break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; break;
    case '3': msg = "\r\n TODO: DMA \r\n"; break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}
```

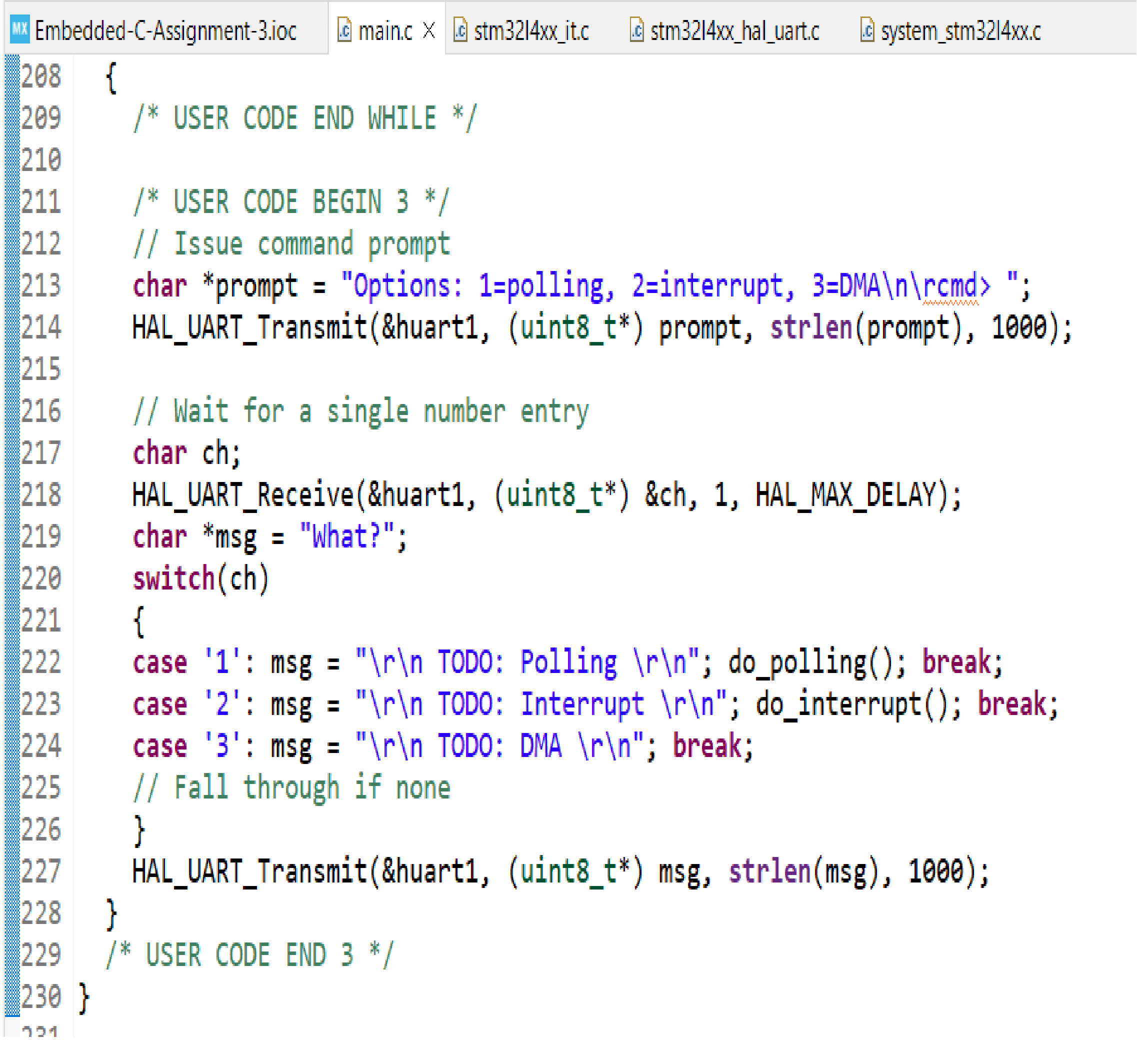
Step 11. User Story 2: Build and run the code, test is successful



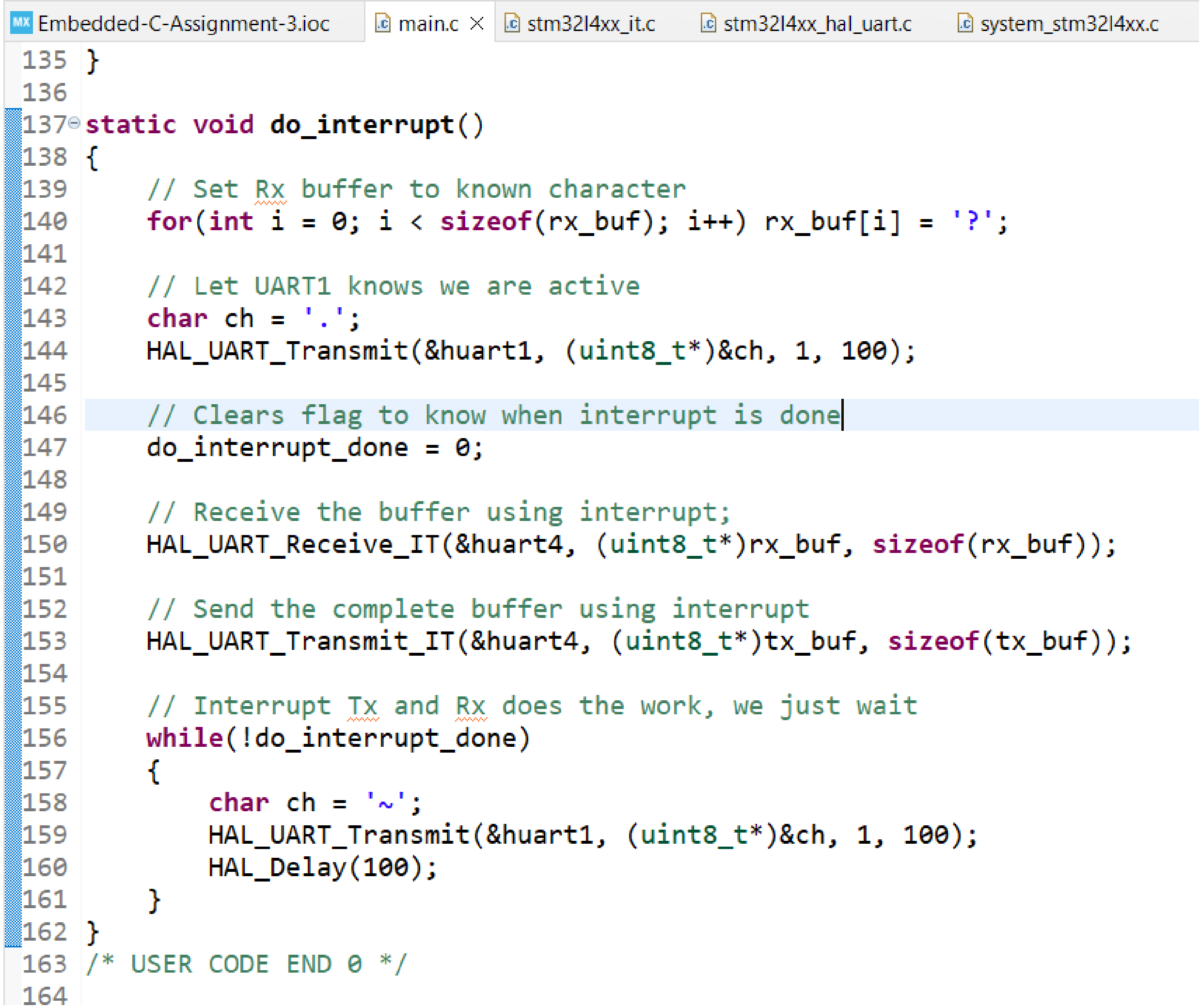
```c
  MX_SPI3_Init();
  MX_USART1_UART_Init();
  MX_USART3_UART_Init();
  MX_USB_OTG_FS_PCD_Init();
  MX_UART4_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; do_polling(); break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; break;
    case '3': msg = "\r\n TODO: DMA \r\n"; break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
```

```
Options: 1=polling, 2=interrupt, 3=DMA
cmd> ............................
 TODO: Polling
Options: 1=polling, 2=interrupt, 3=DMA
cmd> █
```

Step 12. User Story 3: Interrupts. When the user selects 2, use interrupts to transmit and receive the buffer. Write the code in main.c file

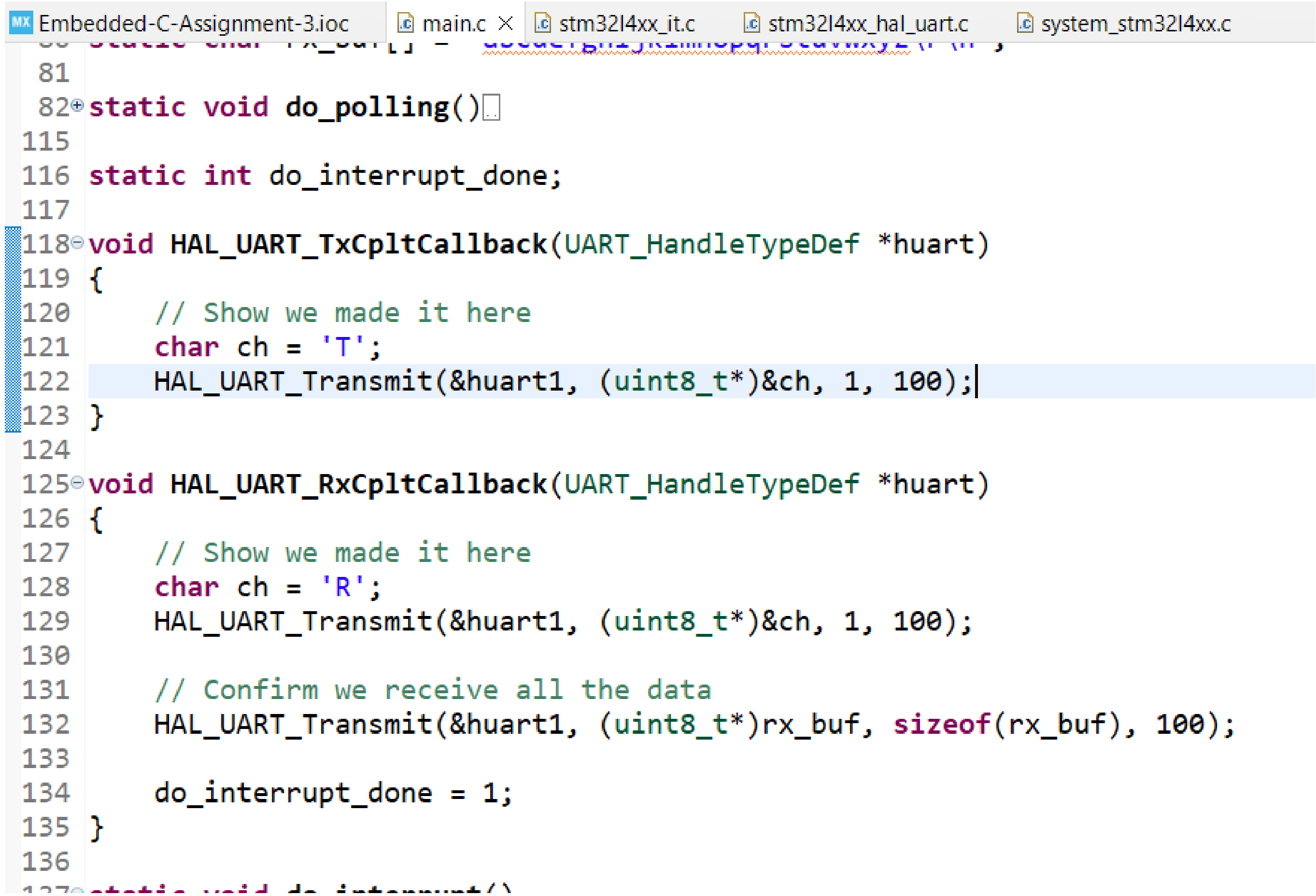


```c
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; do_polling(); break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; do_interrupt(); break;
    case '3': msg = "\r\n TODO: DMA \r\n"; break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}
```



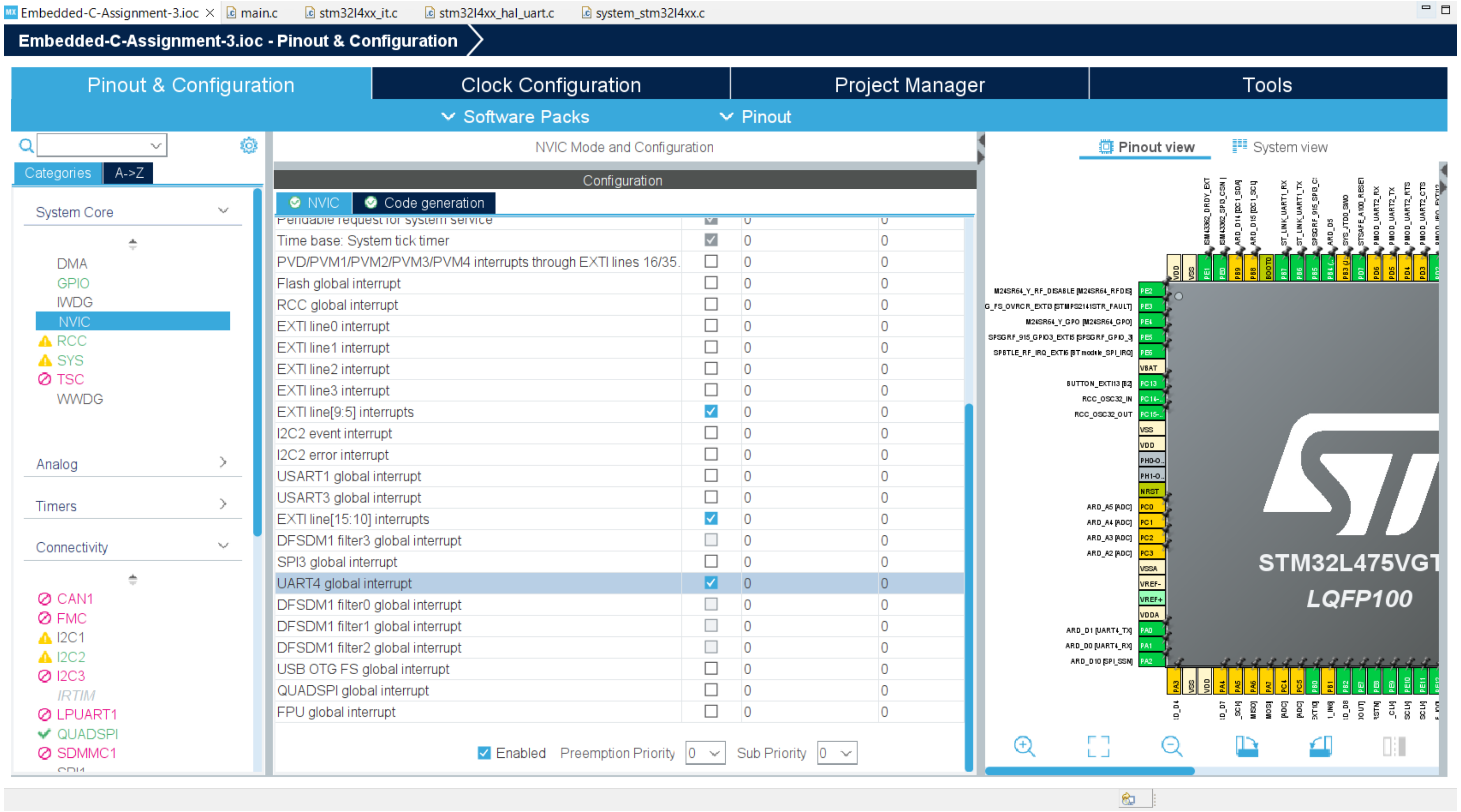
```c
}

static void do_interrupt()
{
    // Set Rx buffer to known character
    for(int i = 0; i < sizeof(rx_buf); i++) rx_buf[i] = '?';

    // Let UART1 knows we are active
    char ch = '.';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

    // Clears flag to know when interrupt is done
    do_interrupt_done = 0;

    // Receive the buffer using interrupt;
    HAL_UART_Receive_IT(&huart4, (uint8_t*)rx_buf, sizeof(rx_buf));

    // Send the complete buffer using interrupt
    HAL_UART_Transmit_IT(&huart4, (uint8_t*)tx_buf, sizeof(tx_buf));

    // Interrupt Tx and Rx does the work, we just wait
    while(!do_interrupt_done)
    {
        char ch = '~';
        HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);
        HAL_Delay(100);
    }
}
/* USER CODE END 0 */
```

Step 13. User Story 3: Write the Rx and Tx complete call back functions



```
static char rx_buf[] = "abcdefghijklmnopqrstuvwxyz\r\n";

static void do_polling()

static int do_interrupt_done;

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
    // Show we made it here
    char ch = 'T';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // Show we made it here
    char ch = 'R';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

    // Confirm we receive all the data
    HAL_UART_Transmit(&huart1, (uint8_t*)rx_buf, sizeof(rx_buf), 100);

    do_interrupt_done = 1;
}

static void do_interrupt()
```

Step 14. User Story 3: Don’t forget to setup the NVIC



NVIC Mode and Configuration

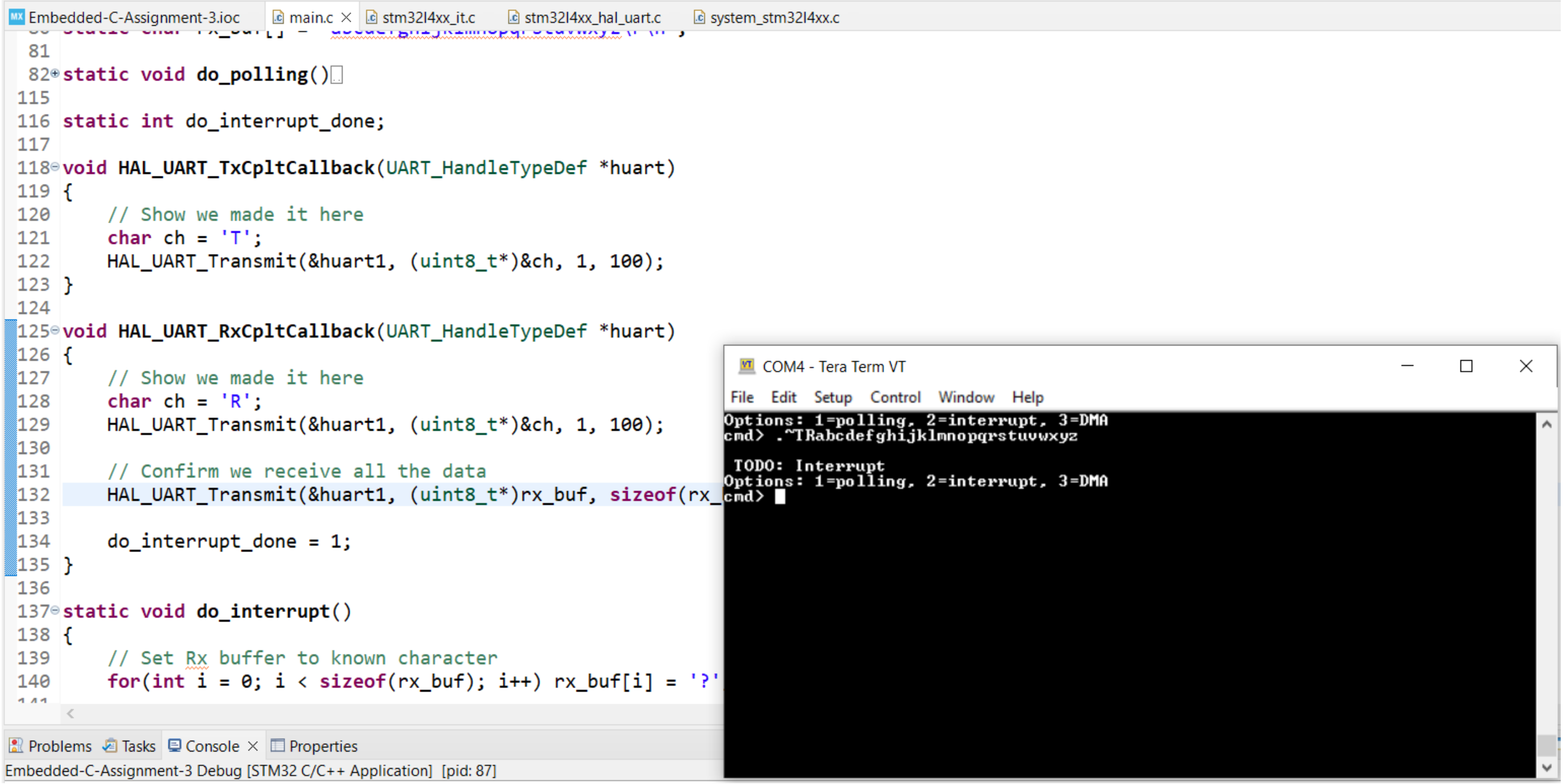
Configuration

NVIC | Code generation

Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/35.	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
EXTI line0 interrupt	☐	0	0
EXTI line1 interrupt	☐	0	0
EXTI line2 interrupt	☐	0	0
EXTI line3 interrupt	☐	0	0
EXTI line[9:5] interrupts	☑	0	0
I2C2 event interrupt	☐	0	0
I2C2 error interrupt	☐	0	0
USART1 global interrupt	☐	0	0
USART3 global interrupt	☐	0	0
EXTI line[15:10] interrupts	☑	0	0
DFSDM1 filter3 global interrupt	☐	0	0
SPI3 global interrupt	☐	0	0
UART4 global interrupt	☑	0	0
DFSDM1 filter0 global interrupt	☐	0	0
DFSDM1 filter1 global interrupt	☐	0	0
DFSDM1 filter2 global interrupt	☐	0	0
USB OTG FS global interrupt	☐	0	0
QUADSPI global interrupt	☐	0	0
FPU global interrupt	☐	0	0

☑ Enabled Preemption Priority 0 Sub Priority 0

Step 15. User Story 3: Build and run the code, test is successful

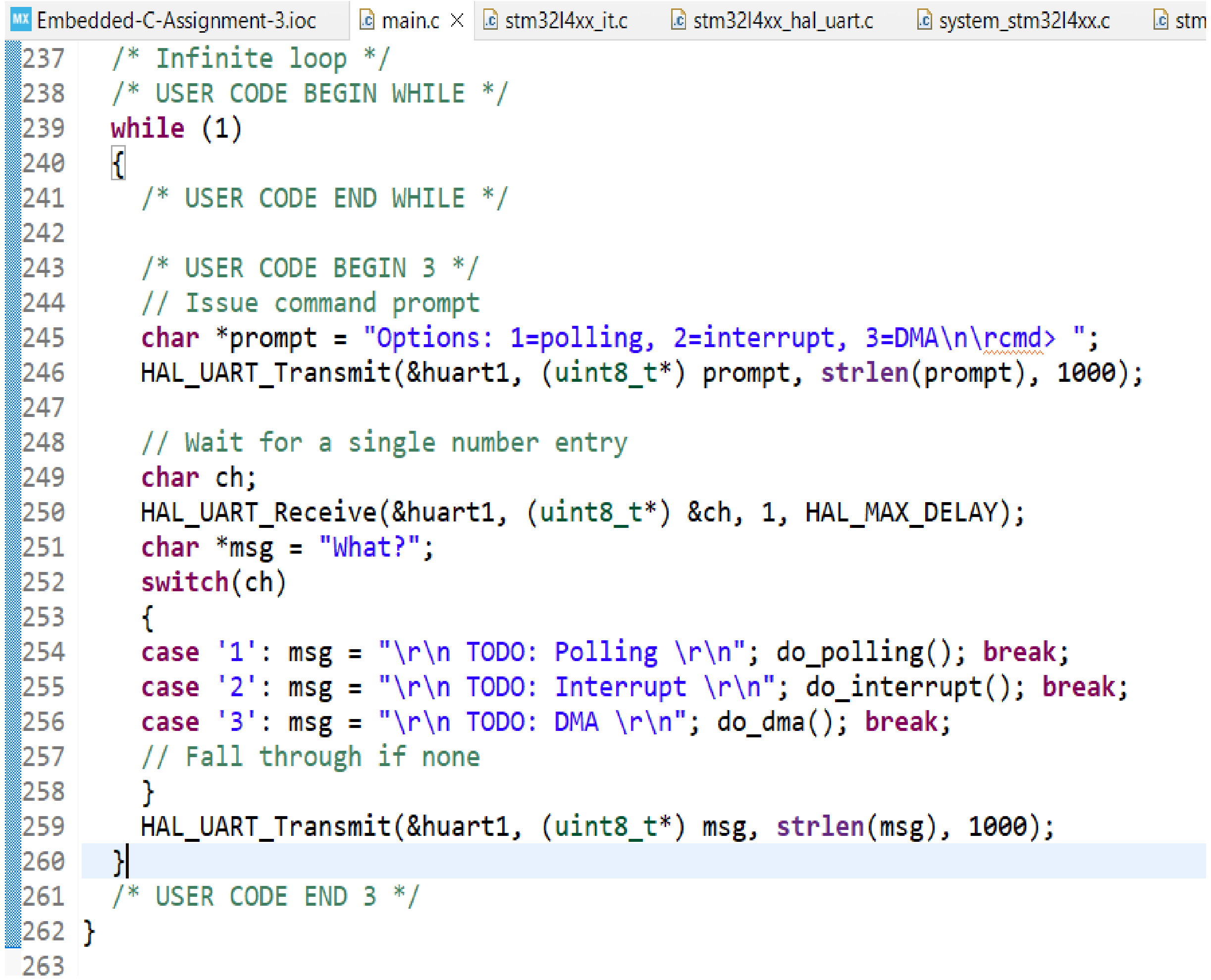


```c
static void do_polling()

static int do_interrupt_done;

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
    // Show we made it here
    char ch = 'T';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // Show we made it here
    char ch = 'R';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

    // Confirm we receive all the data
    HAL_UART_Transmit(&huart1, (uint8_t*)rx_buf, sizeof(rx_

    do_interrupt_done = 1;
}

static void do_interrupt()
{
    // Set Rx buffer to known character
    for(int i = 0; i < sizeof(rx_buf); i++) rx_buf[i] = '?'
```

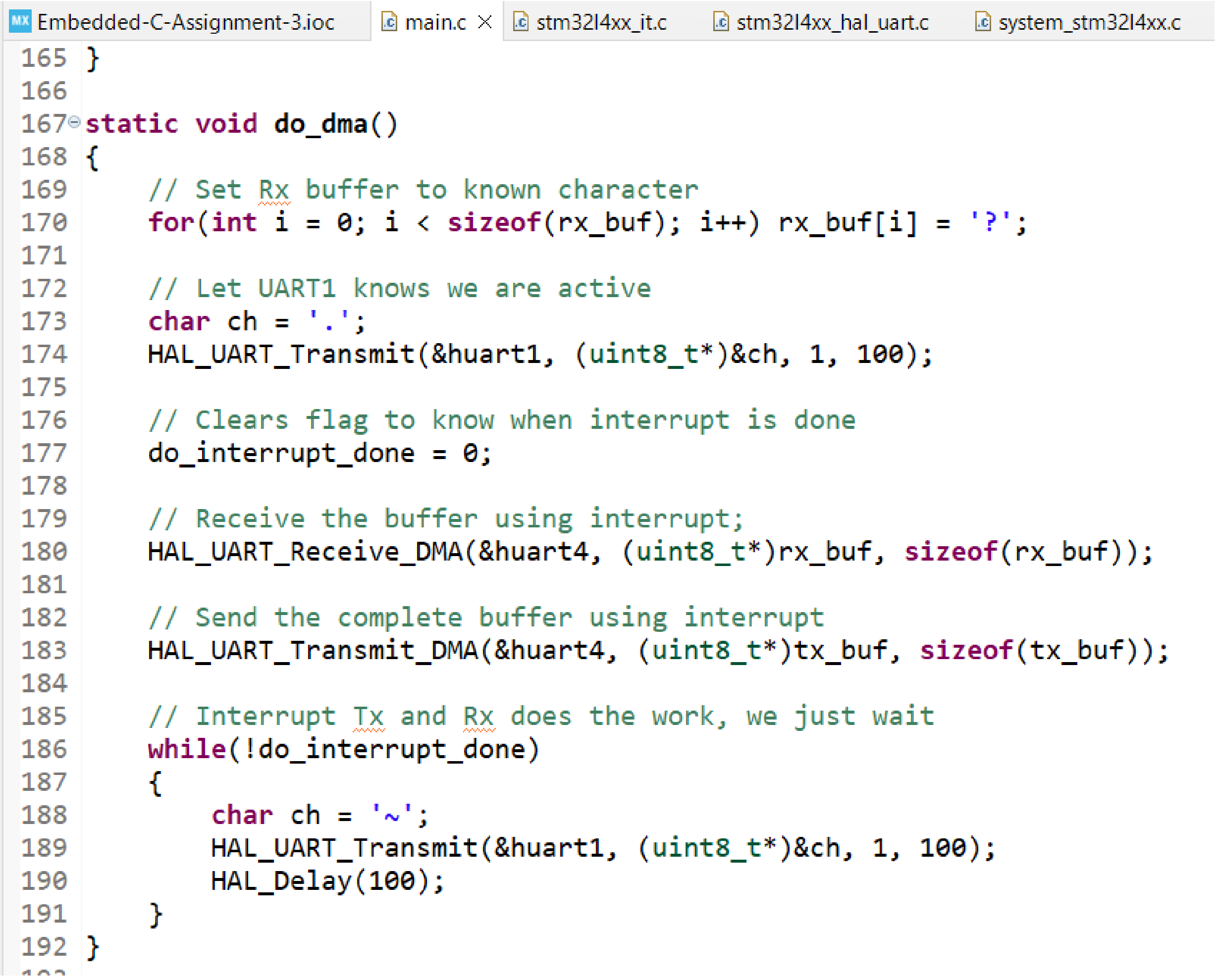
Embedded-C-Assignment-3 Debug [STM32 C/C++ Application] [pid: 87]

```
Options: 1=polling, 2=interrupt, 3=DMA
cmd> .~TRabcdefghijklmnopqrstuvwxyz

 TODO: Interrupt
Options: 1=polling, 2=interrupt, 3=DMA
cmd> 
```

Step 16. User Story 4: DMA. When the user selects 3, use DMA to transmit and receive the buffer. Write the code in main.c file

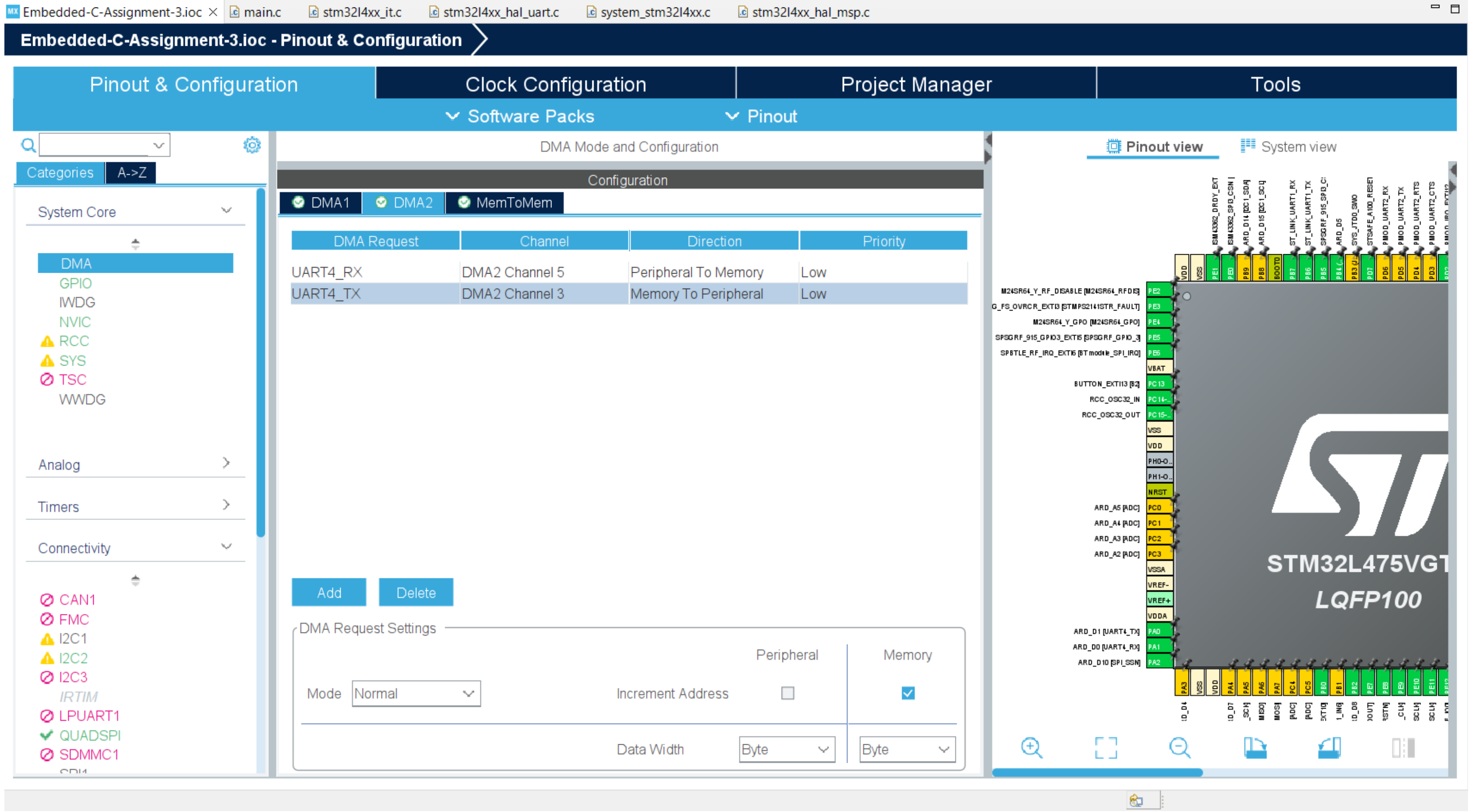


```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Issue command prompt
    char *prompt = "Options: 1=polling, 2=interrupt, 3=DMA\n\rcmd> ";
    HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);

    // Wait for a single number entry
    char ch;
    HAL_UART_Receive(&huart1, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    char *msg = "What?";
    switch(ch)
    {
    case '1': msg = "\r\n TODO: Polling \r\n"; do_polling(); break;
    case '2': msg = "\r\n TODO: Interrupt \r\n"; do_interrupt(); break;
    case '3': msg = "\r\n TODO: DMA \r\n"; do_dma(); break;
    // Fall through if none
    }
    HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
  }
  /* USER CODE END 3 */
}
```



```
}

static void do_dma()
{
    // Set Rx buffer to known character
    for(int i = 0; i < sizeof(rx_buf); i++) rx_buf[i] = '?';

    // Let UART1 knows we are active
    char ch = '.';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

    // Clears flag to know when interrupt is done
    do_interrupt_done = 0;

    // Receive the buffer using interrupt;
    HAL_UART_Receive_DMA(&huart4, (uint8_t*)rx_buf, sizeof(rx_buf));

    // Send the complete buffer using interrupt
    HAL_UART_Transmit_DMA(&huart4, (uint8_t*)tx_buf, sizeof(tx_buf));

    // Interrupt Tx and Rx does the work, we just wait
    while(!do_interrupt_done)
    {
        char ch = '~';
        HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);
        HAL_Delay(100);
    }
}
```

Step 17. User Story 4: Setup the DMA in .ioc file

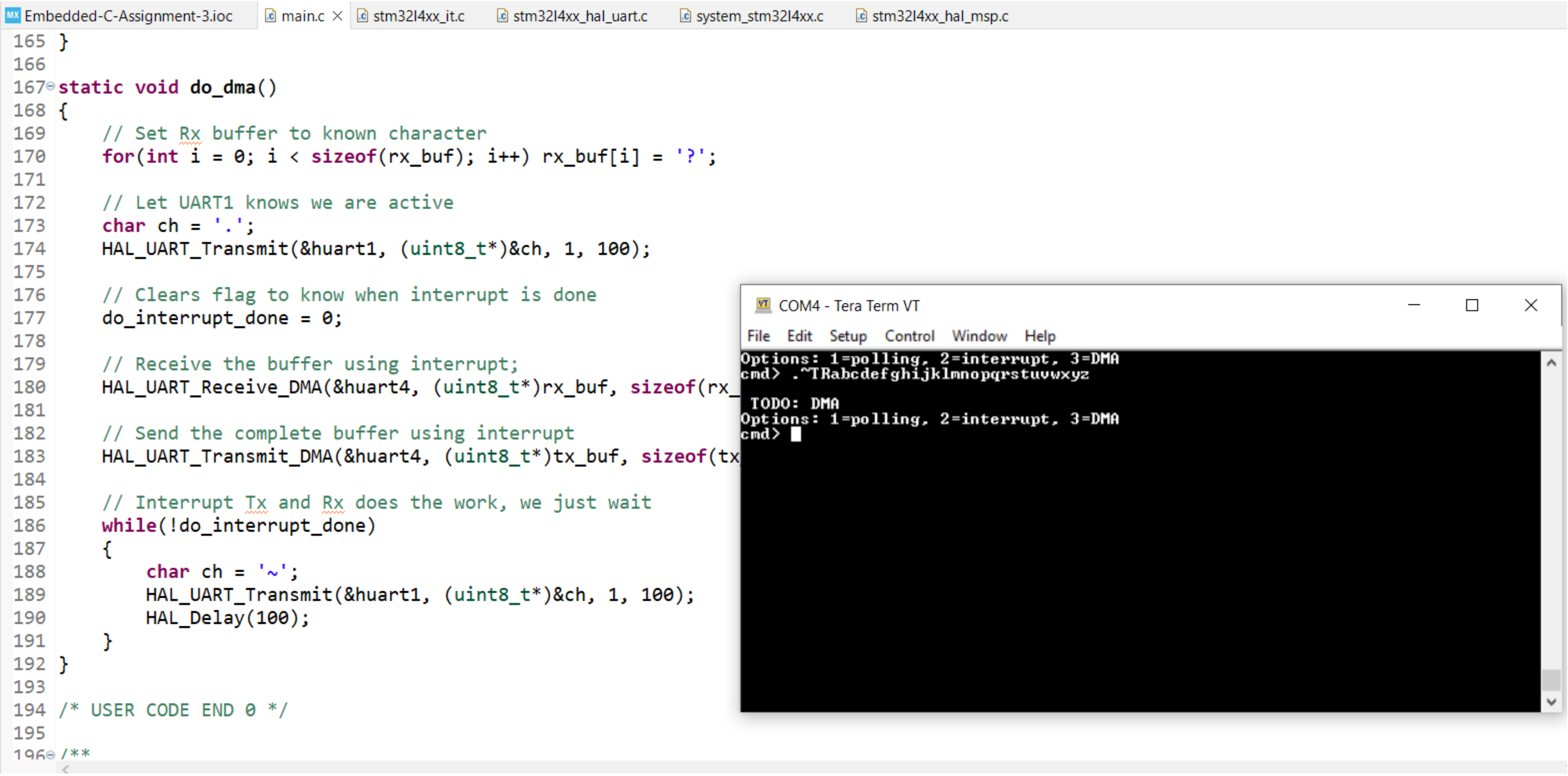


DMA Request	Channel	Direction	Priority
UART4_RX	DMA2 Channel 5	Peripheral To Memory	Low
UART4_TX	DMA2 Channel 3	Memory To Peripheral	Low

DMA Request Settings

		Peripheral	Memory
Mode: Normal	Increment Address	□	☑
	Data Width	Byte	Byte

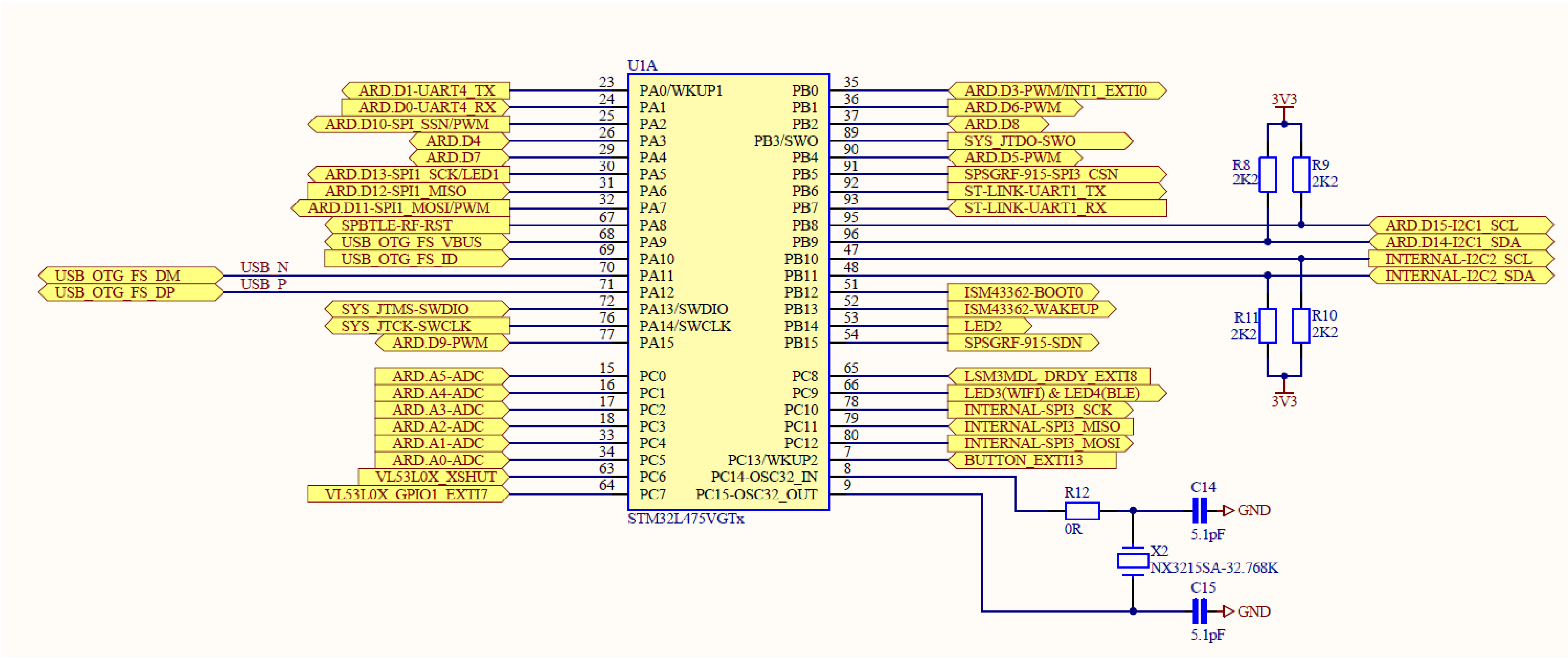
Step 18. User Story 4: Build and run the code, test is successful. Since DMA mode will call the same Tx and Rx call back functions, we do not need to code it up again



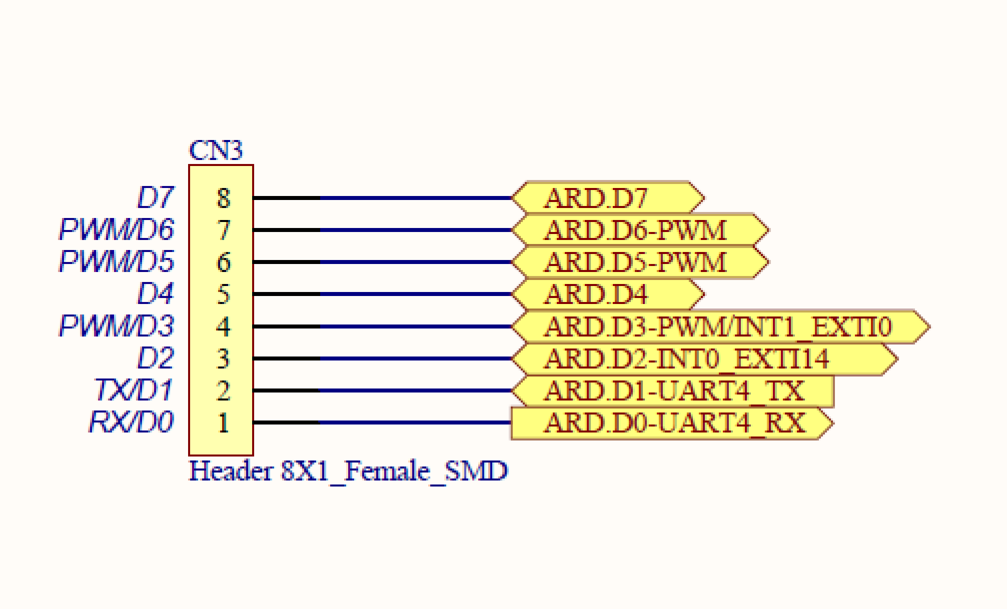
```c
}

static void do_dma()
{
    // Set Rx buffer to known character
    for(int i = 0; i < sizeof(rx_buf); i++) rx_buf[i] = '?';

    // Let UART1 knows we are active
    char ch = '.';
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);

    // Clears flag to know when interrupt is done
    do_interrupt_done = 0;

    // Receive the buffer using interrupt;
    HAL_UART_Receive_DMA(&huart4, (uint8_t*)rx_buf, sizeof(rx_

    // Send the complete buffer using interrupt
    HAL_UART_Transmit_DMA(&huart4, (uint8_t*)tx_buf, sizeof(tx

    // Interrupt Tx and Rx does the work, we just wait
    while(!do_interrupt_done)
    {
        char ch = '~';
        HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 100);
        HAL_Delay(100);
    }
}

/* USER CODE END 0 */

/**
```

```
Options: 1=polling, 2=interrupt, 3=DMA
cmd> .~TRabcdefghijklmnopqrstuvwxyz

 TODO: DMA
Options: 1=polling, 2=interrupt, 3=DMA
cmd>
```

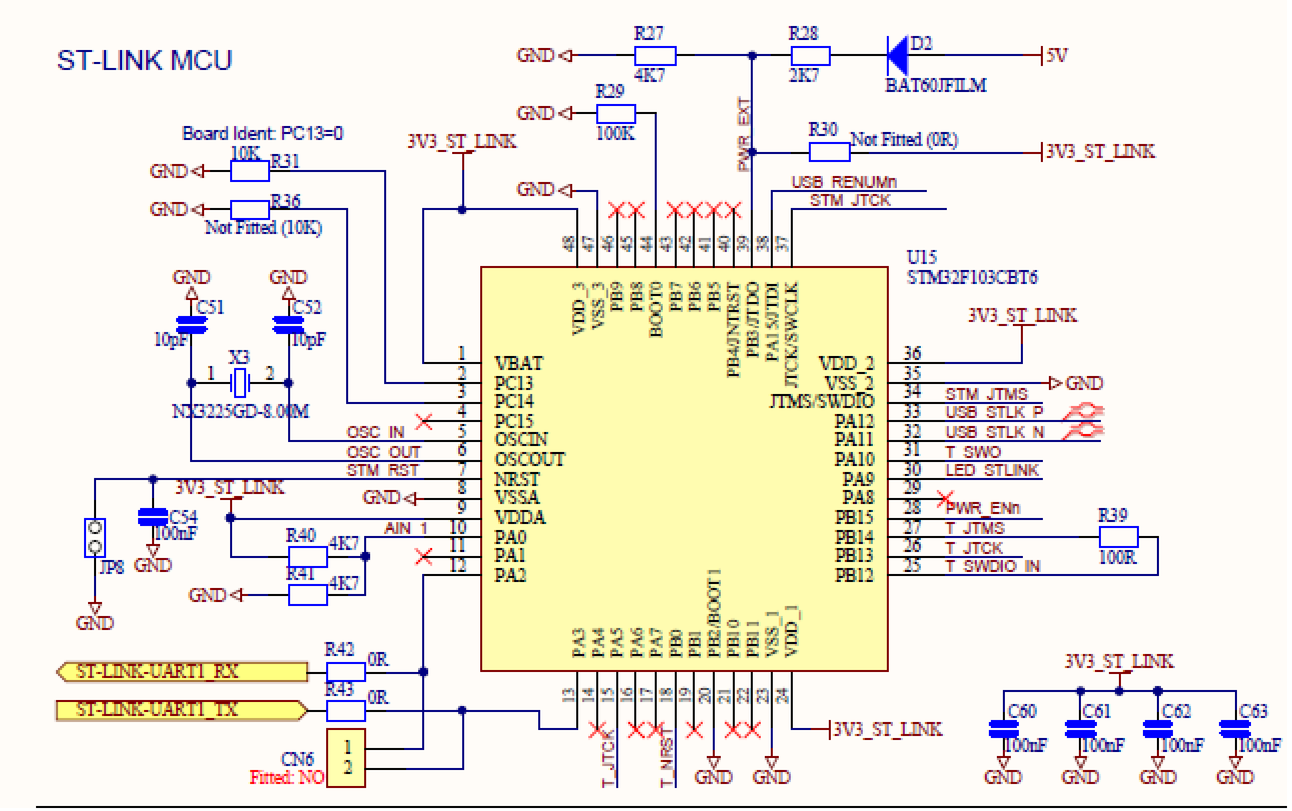
Appendix. Screenshot of STM32 Discovery Kit Schematic diagram where it shows the UART1 and UART4 signals



Appendix. Screenshot of STM32 Discovery Kit Schematic diagram where it shows the UART1 and UART4 signals



CN3
D7
PWM/D6
PWM/D5
D4
PWM/D3
D2
TX/D1
RX/D0
8
7
6
5
4
3
2
1
ARD.D7
ARD.D6-PWM
ARD.D5-PWM
ARD.D4
ARD.D3-PWM/INT1_EXTI0
ARD.D2-INT0_EXTI14
ARD.D1-UART4_TX
ARD.D0-UART4_RX
Header 8X1_Female_SMD



ST-LINK MCU
GND
R27
4K7
R28
2K7
D2
BAT60JFILM
5V
R29
100K
PWR_EXT
R30
Not Fitted (0R)
3V3_ST_LINK
Board Ident: PC13=0
10K
R31
R36
Not Fitted (10K)
USB_RENUMn
STM_JTCK
U15
STM32F103CBT6
C51
C52
10pF
X3
NX3225GD-8.00M
VBAT
PC13
PC14
PC15
OSCIN
OSCOUT
NRST
VSSA
VDDA
PA0
PA1
PA2
OSC_IN
OSC_OUT
STM_RST
AIN_1
JP8
C54
100nF
R40
R41
4K7
VDD_3
VSS_3
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4/JNTRST
PB3/JTDO
PA15/JTDI
JTCK/SWCLK
VDD_2
VSS_2
JTMS/SWDIO
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12
STM_JTMS
USB_STLK_P
USB_STLK_N
T_SWO
LED_STLINK
PWR_ENn
T_JTMS
T_JTCK
T_SWDIO_IN
R39
100R
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2/BOOT1
PB10
PB11
VSS_1
VDD_1
T_JTCK
T_NRST
ST-LINK-UART1_RX
ST-LINK-UART1_TX
R42
R43
0R
CN6
Fitted: NO
C60
C61
C62
C63
100nF