

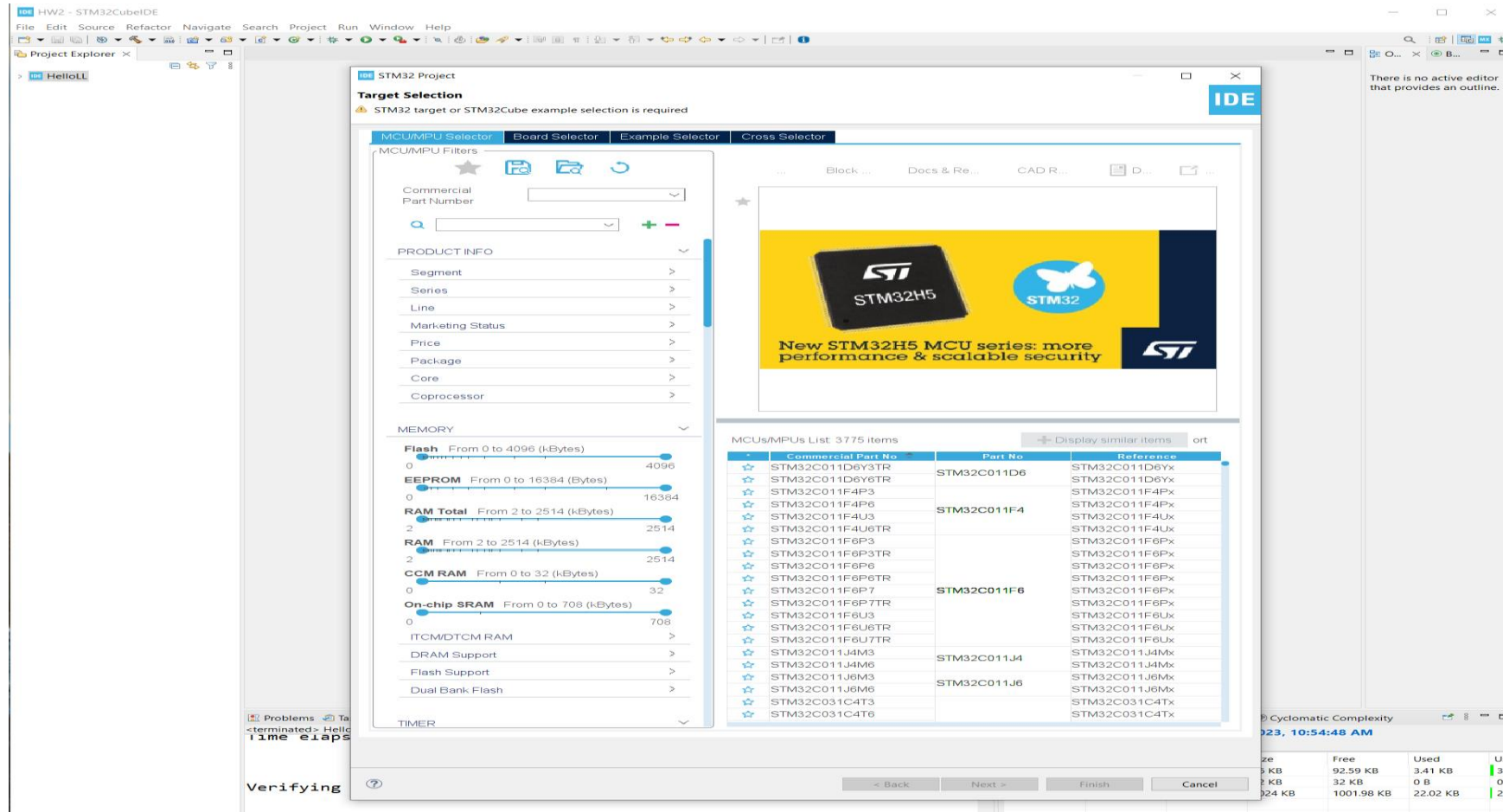
UCSD Embedded C Assignment 8

By

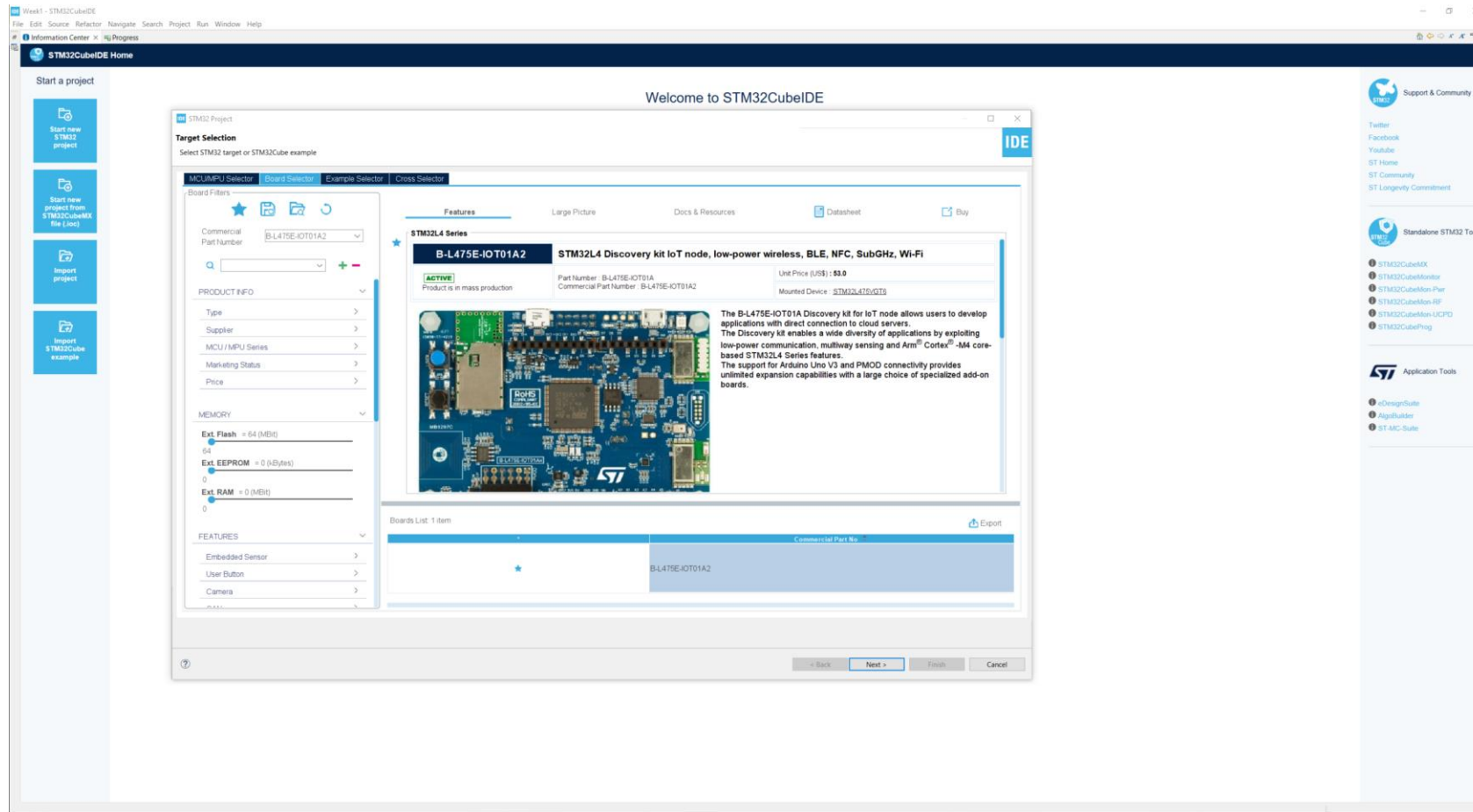
Hsuankai Chang

hsuankac@umich.edu

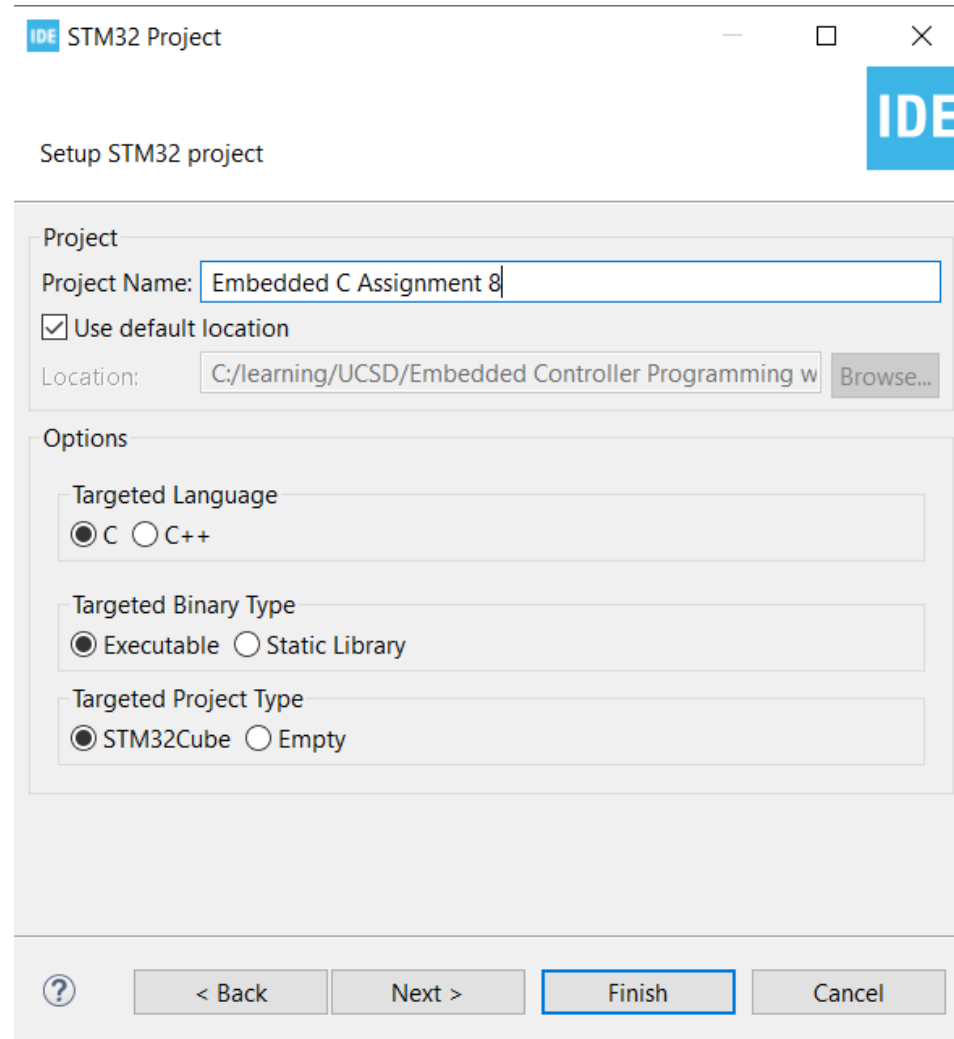
Step 1. Startup STM32CubeIDE and create new STM32 project



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



Step 3. Enter the project name then click Next



The image shows a 'Setup STM32 project' dialog box from an IDE. The window title is 'IDE STM32 Project'. The dialog is titled 'Setup STM32 project'. It has two main sections: 'Project' and 'Options'. In the 'Project' section, the 'Project Name' field contains 'Embedded C Assignment 8'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:/learning/UCSD/Embedded Controller Programming w' with a 'Browse...' button next to it. The 'Options' section contains three groups of radio buttons: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish' (which is highlighted with a blue border). A 'Cancel' button is also present.

IDE STM32 Project

Setup STM32 project

Project

Project Name: Embedded C Assignment 8

☒ Use default location

Location: C:/learning/UCSD/Embedded Controller Programming w Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel

Step 4. See the firmware package name and version



The image shows a Windows-style dialog box titled "STM32 Project" with a subtitle "Firmware Library Package Setup". The subtitle also includes the instruction "Setup STM32 target's firmware". The dialog is divided into three sections: "Target and Firmware Package", "Firmware and Software Package Repository", and "Code Generator Options". In the first section, the "Target Reference" is "B-L475E-IOT01A2" and the "Firmware Package Name and Version" is "STM32Cube FW_L4 V1.17.2", with the version part highlighted by a blue selection box. The second section shows the "Location" as "C:\Users\hsuankai.chang\STM32Cube\Repository" and includes a link to the "Firmware Updater". The third section contains three radio button options for code generation, with the last option, "Copy only the necessary library files", being selected. At the bottom, there are buttons for "?", "< Back", "Next >", "Finish" (which is highlighted with a blue border), and "Cancel".

IDE 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

Firmware and Software Package Repository

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

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

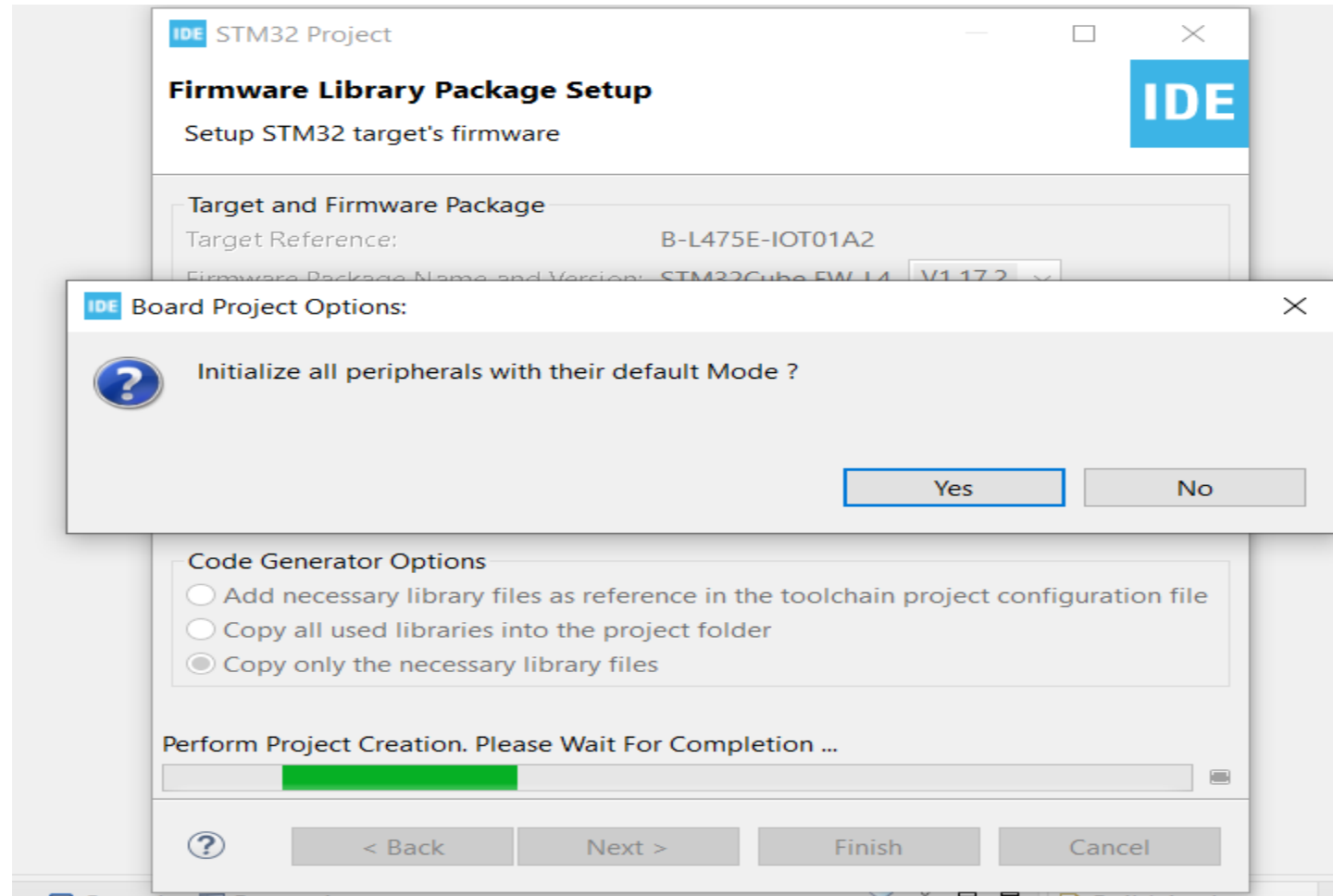
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

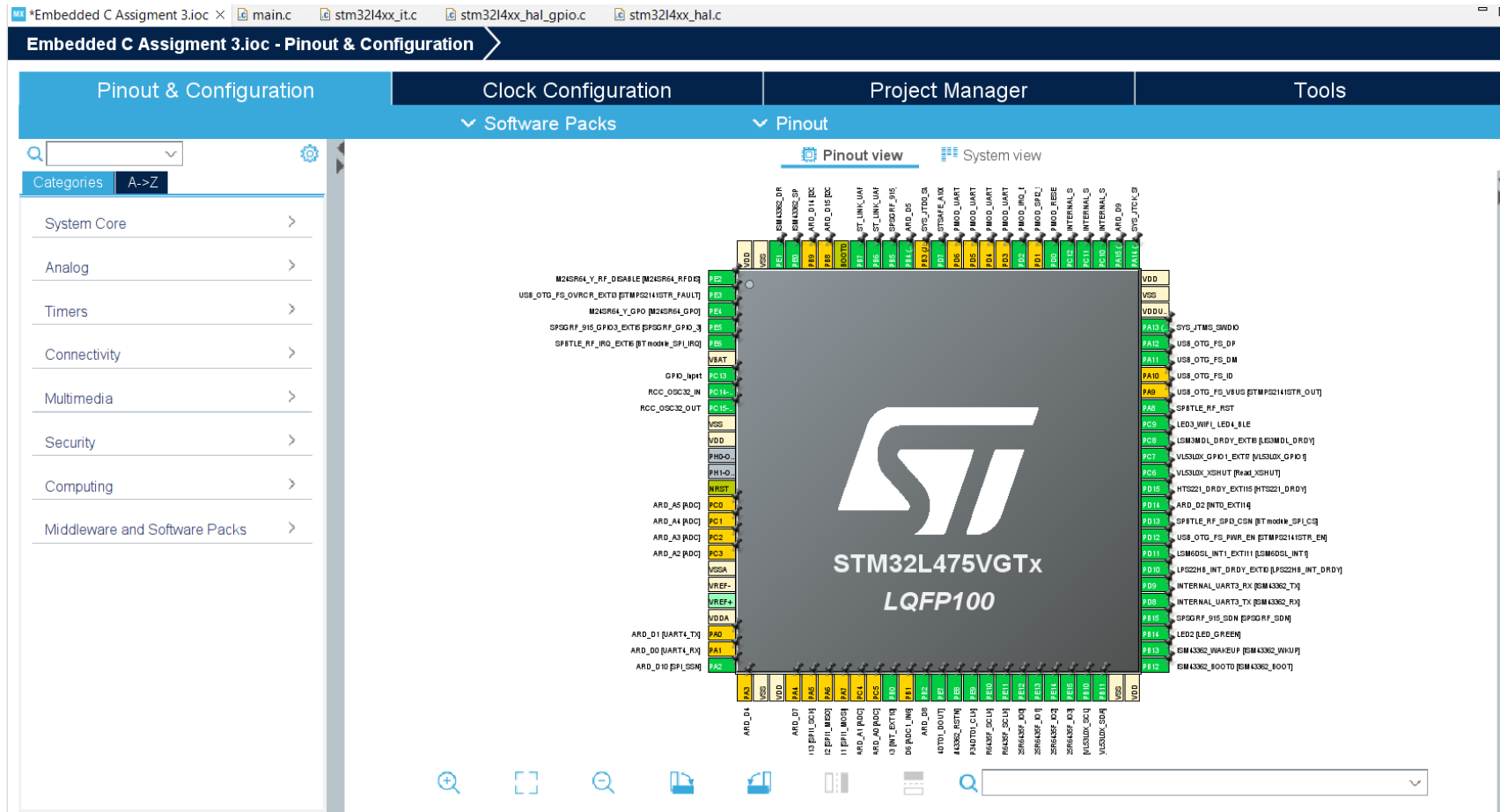
☒ Copy only the necessary library files

? < Back Next > Finish Cancel

Step 5. Click yes to initialize all peripherals to default



Step 6. When in .ioc file, click System view



Step 7. Check the setting for I2C2 to communicate with HTS221 sensor

Embedded C Assignment 8.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

I2C2 Mode and Configuration

Mode: I2C

Configuration

Reset Configuration

DMA Settings | GPIO Settings | Parameter Settings | User Constants | NVIC Settings

Configure the below parameters:

Search (Ctrl+F)

Timing configuration

- Custom Timing: Disabled
- I2C Speed Mode: Standard Mode
- I2C Speed Frequency (K...): 100
- Rise Time (ns): 0
- Fall Time (ns): 0
- Coefficient of Digital Filter: 0
- Analog Filter: Enabled
- Timing: 0x10909CEC

Slave Features

- Clock No Stretch Mode: Disabled
- General Call Address Det.: Disabled
- Primary Address Length ...: 7-bit
- Dual Address Acknowled.: Disabled
- Primary slave address: 0

Pinout view | System view

STM32L475VGTX LQFP100

Step 8. Generate code

Embedded C Assignment 7.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories: A-Z

- System Core
- Analog
- Timers
- Connectivity
 - CAN1
 - FMC
 - I2C1
 - I2C2
 - I2C3
 - IRTIM
 - LP_UART1
 - QUADSPI
 - SDMMC1
 - SPI1
 - SPI2
 - SPI3
 - SWPMI1
 - UART4
 - UART5
 - USART1
 - USART2
 - USART3
 - USB_OTG_FS
- Multimedia
- Security
- Computing
- Middleware and Software Packs

UART4 Mode and Configuration

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Hardware Flow Control (RS485): Enable

Question: Do you want generate Code?

☐ Remember my decision

Yes No

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters:

Search (Ctrl+F)

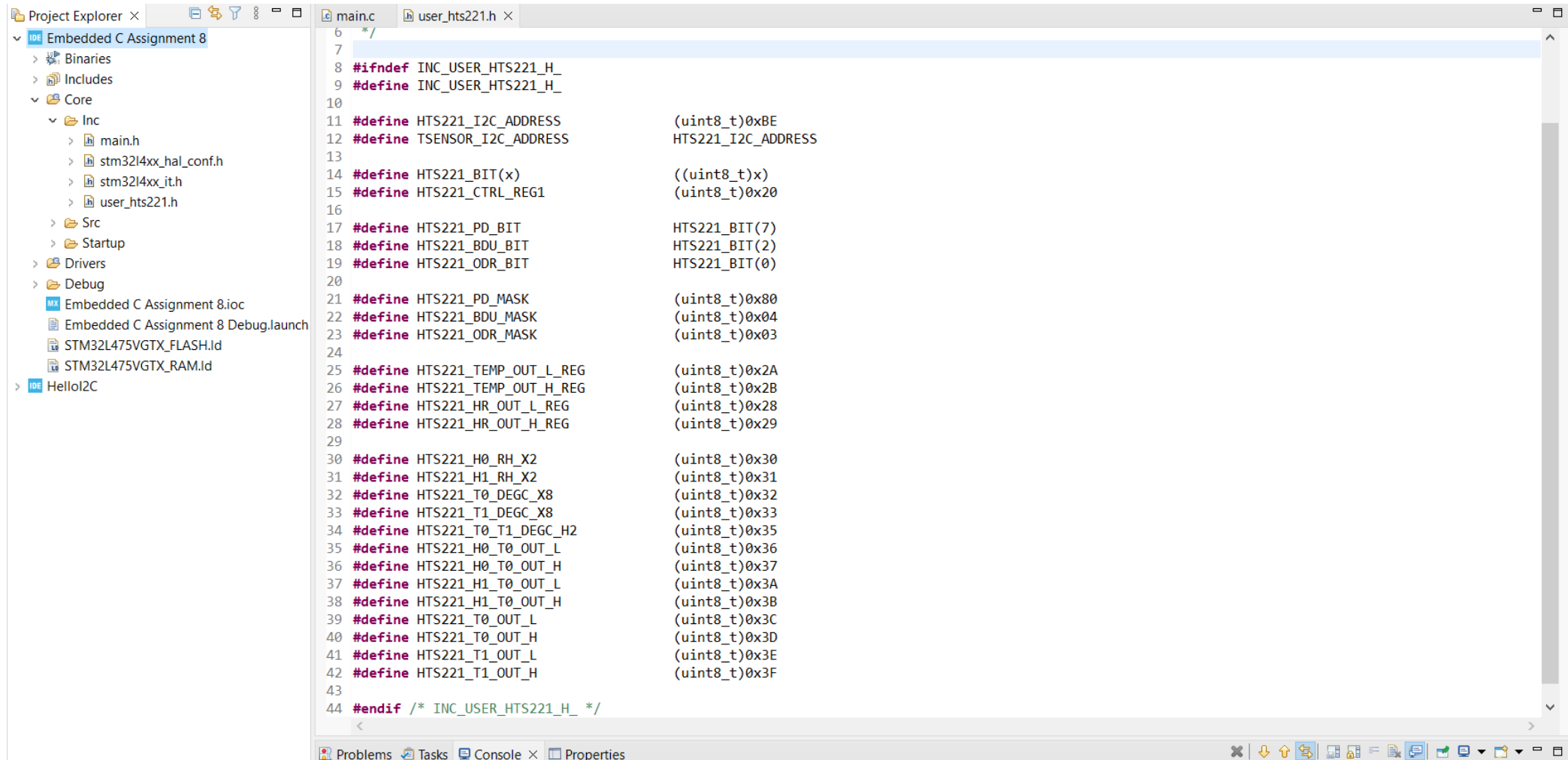
- Basic Parameters
 - Baud Rate: 115200 Bits/s
 - Word Length: 8 Bits (including Parity)
 - Parity: None
 - Stop Bits: 1
- Advanced Parameters
 - Data Direction: Receive and Transmit
 - Over Sampling: 16 Samples
 - Single Sample: Disable
- Advanced Features
 - Auto Baudrate: Disable
 - TX Pin Active Level Inversion: Disable
 - RX Pin Active Level Inversion: Disable
 - Data Inversion: Disable
 - TX and RX Pins Swapping: Disable
 - Overrun: Enable
 - DMA on RX Error: Enable
 - MSB First: Disable

Pinout view | System view

STM32L475VGTx LQFP100

ARD_D0 [UART4_RX]

Step 9. In core/include folder, create user_hts221.h file to store necessary macros



The screenshot shows an IDE with two windows. The left window, 'Project Explorer', displays the project structure for 'Embedded C Assignment 8'. The 'Core' folder is expanded, showing the 'Inc' subfolder which contains 'main.h', 'stm32l4xx_hal_conf.h', 'stm32l4xx_it.h', and 'user_hts221.h'. The right window, 'main.c', shows the content of 'user_hts221.h'. The file contains a series of macro definitions for the HTS221 sensor, including address, bit, and mask definitions, and a conditional compilation block for the 'INC_USER_HTS221_H' macro.

```
6  */
7
8  #ifndef INC_USER_HTS221_H_
9  #define INC_USER_HTS221_H_
10
11 #define HTS221_I2C_ADDRESS      (uint8_t)0xBE
12 #define TSSENSOR_I2C_ADDRESS  HTS221_I2C_ADDRESS
13
14 #define HTS221_BIT(x)          ((uint8_t)x)
15 #define HTS221_CTRL_REG1      (uint8_t)0x20
16
17 #define HTS221_PD_BIT          HTS221_BIT(7)
18 #define HTS221_BDU_BIT        HTS221_BIT(2)
19 #define HTS221_ODR_BIT         HTS221_BIT(0)
20
21 #define HTS221_PD_MASK         (uint8_t)0x80
22 #define HTS221_BDU_MASK       (uint8_t)0x04
23 #define HTS221_ODR_MASK       (uint8_t)0x03
24
25 #define HTS221_TEMP_OUT_L_REG  (uint8_t)0x2A
26 #define HTS221_TEMP_OUT_H_REG (uint8_t)0x2B
27 #define HTS221_HR_OUT_L_REG   (uint8_t)0x28
28 #define HTS221_HR_OUT_H_REG   (uint8_t)0x29
29
30 #define HTS221_H0_RH_X2        (uint8_t)0x30
31 #define HTS221_H1_RH_X2        (uint8_t)0x31
32 #define HTS221_T0_DEGC_X8      (uint8_t)0x32
33 #define HTS221_T1_DEGC_X8      (uint8_t)0x33
34 #define HTS221_T0_T1_DEGC_H2   (uint8_t)0x35
35 #define HTS221_H0_T0_OUT_L     (uint8_t)0x36
36 #define HTS221_H0_T0_OUT_H     (uint8_t)0x37
37 #define HTS221_H1_T0_OUT_L     (uint8_t)0x3A
38 #define HTS221_H1_T0_OUT_H     (uint8_t)0x3B
39 #define HTS221_T0_OUT_L        (uint8_t)0x3C
40 #define HTS221_T0_OUT_H        (uint8_t)0x3D
41 #define HTS221_T1_OUT_L        (uint8_t)0x3E
42 #define HTS221_T1_OUT_H        (uint8_t)0x3F
43
44 #endif /* INC_USER_HTS221_H_ */
```

Step 10. In main.c file, create private function prototypes

```
main.c × user_hsts221.h
55 UART_HandleTypeDef huart3;
56
57 PCD_HandleTypeDef hpcd_USB_OTG_FS;
58
59 /* USER CODE BEGIN PV */
60
61 /* USER CODE END PV */
62
63 /* Private function prototypes -----*/
64 void SystemClock_Config(void);
65 static void MX_GPIO_Init(void);
66 static void MX_DFSDM1_Init(void);
67 static void MX_I2C2_Init(void);
68 static void MX_QUADSPI_Init(void);
69 static void MX_SPI3_Init(void);
70 static void MX_USART1_UART_Init(void);
71 static void MX_USART3_UART_Init(void);
72 static void MX_USB_OTG_FS_PCD_Init(void);
73 /* USER CODE BEGIN PFP */
74 static void Temperature_Sensor_Init(uint16_t DeviceAddr);
75 static void Humidity_Sensor_Init(uint16_t DeviceAddr);
76 static void Sensor_Write_Multiple(uint8_t Addr, uint8_t Reg, uint8_t Value);
77 static uint8_t Sensor_Read(uint8_t Addr, uint8_t Reg);
78 static uint16_t Sensor_Read_Multiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length);
79 static HAL_StatusTypeDef I2Cx_WriteMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length);
80 static HAL_StatusTypeDef I2Cx_ReadMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length);
81 static float Read_Temp_Sensor(uint16_t DeviceAddr);
82 static float Read_Humidity_Sensor(uint16_t DeviceAddr);
83 /* USER CODE END PFP */
84
85 /* Private user code -----*/
86 /* USER CODE BEGIN 0 */
87
88 /* USER CODE END 0 */
89
90 /**
91  * @brief The application entry point.
92  * @retval int
```

Step 11. In main.c file, write the function definition for read/write to and from sensor using I2C

```
675 /* USER CODE BEGIN 4 */
676 static HAL_StatusTypeDef I2Cx_ReadMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length)
677 {
678     HAL_StatusTypeDef status = HAL_OK;
679
680     status = HAL_I2C_Mem_Read(i2c_handler, Addr, (uint16_t)Reg, MemAddress, Buffer, Length, 1000);
681
682     /* Check the communication status */
683     if(status != HAL_OK)
684     {
685         /* De-initialize the I2C communication bus */
686         HAL_I2C_DeInit(&hi2c2);
687     }
688     return status;
689 }
690
691 static HAL_StatusTypeDef I2Cx_WriteMultiple(I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length)
692 {
693     HAL_StatusTypeDef status = HAL_OK;
694
695     status = HAL_I2C_Mem_Write(i2c_handler, Addr, (uint16_t)Reg, MemAddress, Buffer, Length, 1000);
696
697     /* Check the communication status */
698     if(status != HAL_OK)
699     {
700         /* De-initialize the I2C communication bus */
701         HAL_I2C_DeInit(&hi2c2);
702     }
703     return status;
704 }
705
```

Step 12. In main.c file, create API function to use those read and write functions

```
706 static uint8_t Sensor_Read(uint8_t Addr, uint8_t Reg)
707 {
708     uint8_t read_value = 0;
709
710     I2Cx_ReadMultiple(&hi2c2, Addr, Reg, I2C_MEMADD_SIZE_8BIT, (uint8_t*)&read_value, 1);
711
712     return read_value;
713 }
714
715 static uint16_t Sensor_Read_Multiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length)
716 {
717     return I2Cx_ReadMultiple(&hi2c2, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, Buffer, Length);
718 }
719
720 static void Sensor_Write_Multiple(uint8_t Addr, uint8_t Reg, uint8_t Value)
721 {
722     I2Cx_WriteMultiple(&hi2c2, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, (uint8_t*)&Value, 1);
723 }
724
```

Step 13. Write functions for initialization. Since I2C initialization and GPIO pin alternate function configuration, priority setting, etc. already done in MX_I2C2_Init function, we don't need to do it again

```
5- static void Temperature_Sensor_Init(uint16_t DeviceAddr)
6 {
7     uint8_t tmp;
8
9     /* Read CTRL_REG1 */
0     tmp = Sensor_Read(DeviceAddr, HTS221_CTRL_REG1);
1
2     /* Enable BDU */
3     tmp &= ~HTS221_BDU_MASK;
4     tmp |= (1 << HTS221_BDU_BIT);
5
6     /* Set default ODR */
7     tmp &= ~HTS221_ODR_MASK;
8     tmp |= (uint8_t)0x01; /* Set ODR to 1Hz */
9
0     /* Activate the device */
1     tmp |= HTS221_PD_MASK;
2
3     /* Apply settings to CTRL_REG1 */
4     Sensor_Write_Multiple(DeviceAddr, HTS221_CTRL_REG1, tmp);
5 }
6
7- static void Humidity_Sensor_Init(uint16_t DeviceAddr)
8 {
9     uint8_t tmp;
0
1     /* Read CTRL_REG1 */
2     tmp = Sensor_Read(DeviceAddr, HTS221_CTRL_REG1);
3
4     /* Enable BDU */
5     tmp &= ~HTS221_BDU_MASK;
6     tmp |= (1 << HTS221_BDU_BIT);
7
8     /* Set default ODR */
9     tmp &= ~HTS221_ODR_MASK;
0     tmp |= (uint8_t)0x01; /* Set ODR to 1Hz */
1
2     /* Activate the device */
3     tmp |= HTS221_PD_MASK;
4
5     /* Apply settings to CTRL_REG1 */
6     Sensor_Write_Multiple(DeviceAddr, HTS221_CTRL_REG1, tmp);
7 }
8
```

Step 14. Implement read data from temperature and humidity sensor

```
static float Read_Temp_Sensor(uint16_t DeviceAddr)
{
    int16_t T0_out, T1_out, T_out, T0_degC_x8_u16, T1_degC_x8_u16;
    int16_t T0_degC, T1_degC;
    uint8_t buffer[4], tmp;
    float tmp_f;

    Sensor_Read_Multiple(DeviceAddr, (HTS221_T0_DEGC_X8 | 0x80), buffer, 2);
    tmp = Sensor_Read(DeviceAddr, HTS221_T0_T1_DEGC_H2);

    T0_degC_x8_u16 = (((uint16_t)(tmp & 0x03)) << 8) | ((uint16_t)buffer[0]);
    T1_degC_x8_u16 = (((uint16_t)(tmp & 0x0C)) << 6) | ((uint16_t)buffer[1]);
    T0_degC = T0_degC_x8_u16 >> 3;
    T1_degC = T1_degC_x8_u16 >> 3;

    Sensor_Read_Multiple(DeviceAddr, (HTS221_T0_OUT_L | 0x80), buffer, 4);

    T0_out = (((uint16_t)buffer[1]) << 8) | (uint16_t)buffer[0];
    T1_out = (((uint16_t)buffer[3]) << 8) | (uint16_t)buffer[2];

    Sensor_Read_Multiple(DeviceAddr, (HTS221_TEMP_OUT_L_REG | 0x80), buffer, 2);

    T_out = (((uint16_t)buffer[1]) << 8) | (uint16_t)buffer[0];

    tmp_f = (float)(T_out - T0_out) * (float)(T1_degC - T0_degC) / (float)(T1_out - T0_out) + T0_degC;

    return tmp_f;
}
```

```
static float Read_Humidity_Sensor(uint16_t DeviceAddr)
{
    int16_t H0_T0_out, H1_T0_out, H_T_out;
    int16_t H0_rh, H1_rh;
    uint8_t buffer[2];
    float tmp_f;

    Sensor_Read_Multiple(DeviceAddr, (HTS221_H0_RH_X2 | 0x80), buffer, 2);

    H0_rh = buffer[0] >> 1;
    H1_rh = buffer[1] >> 1;

    Sensor_Read_Multiple(DeviceAddr, (HTS221_H0_T0_OUT_L | 0x80), buffer, 2);

    H0_T0_out = (((uint16_t)buffer[1]) << 8) | (uint16_t)buffer[0];

    Sensor_Read_Multiple(DeviceAddr, (HTS221_H1_T0_OUT_L | 0x80), buffer, 2);

    H1_T0_out = (((uint16_t)buffer[1]) << 8) | (uint16_t)buffer[0];

    Sensor_Read_Multiple(DeviceAddr, (HTS221_HR_OUT_L_REG | 0x80), buffer, 2);

    H_T_out = (((uint16_t)buffer[1]) << 8) | (uint16_t)buffer[0];

    tmp_f = (float)(H_T_out - H0_T0_out) * (float)(H1_rh - H0_rh) / (float)(H1_T0_out - H0_T0_out) + H0_rh;
    tmp_f *= 10.0f;

    tmp_f = ( tmp_f > 1000.0f ) ? 1000.0f
        : ( tmp_f < 0.0f ) ? 0.0f
        : tmp_f;

    return (tmp_f / 10.0f);
}
```

Step 15. In main function, implement the driver code to read temp and humidity value then print it out on UART1

```
/* USER CODE BEGIN 2 */
float temp_value = 0;
float hum_value = 0;
char str_tmp[100] = "";

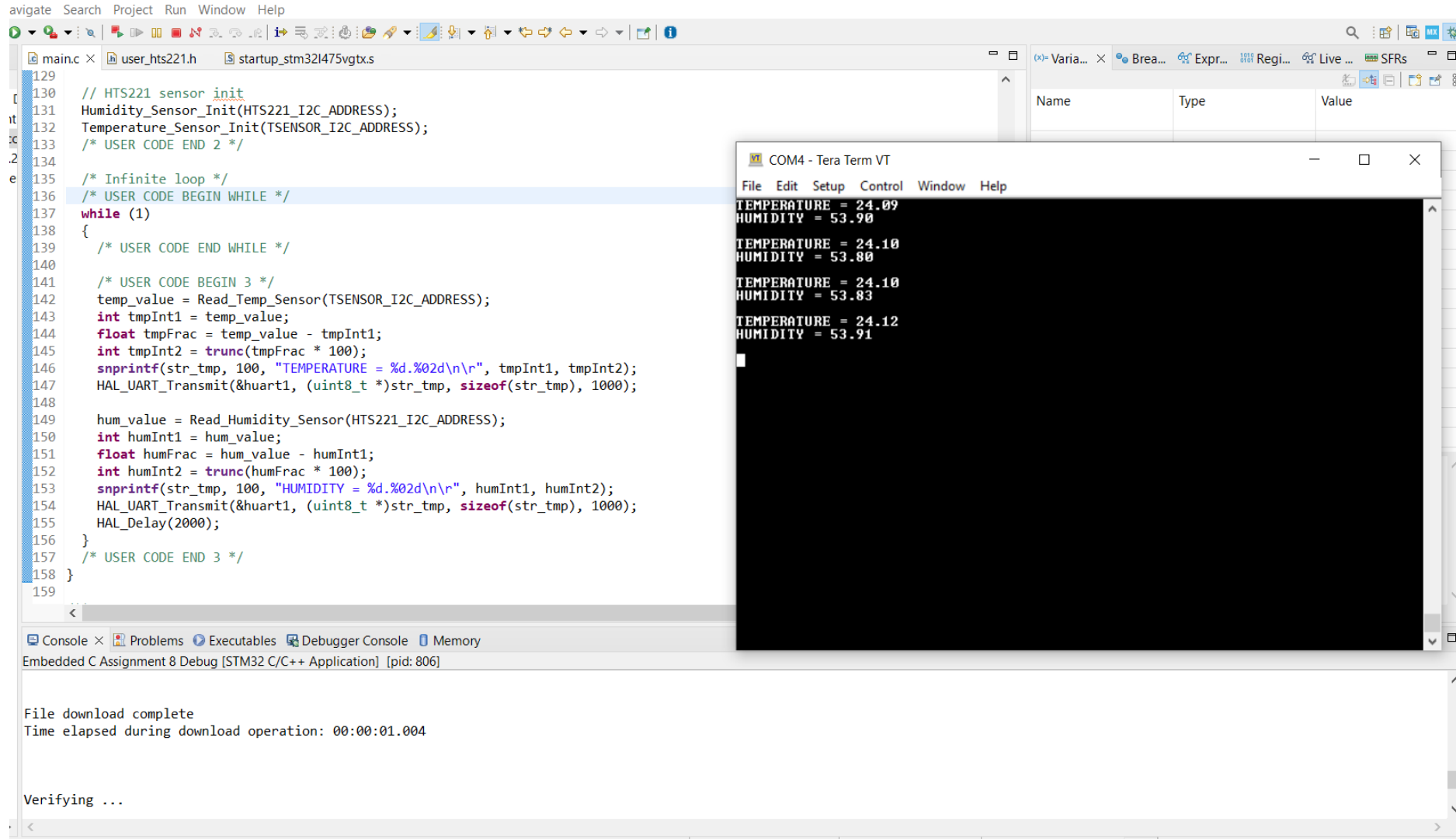
// HTS221 sensor init
Humidity_Sensor_Init(HTS221_I2C_ADDRESS);
Temperature_Sensor_Init(TSENSOR_I2C_ADDRESS);
/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    temp_value = Read_Temp_Sensor(TSENSOR_I2C_ADDRESS);
    int tmpInt1 = temp_value;
    float tmpFrac = temp_value - tmpInt1;
    int tmpInt2 = trunc(tmpFrac * 100);
    snprintf(str_tmp, 100, "TEMPERATURE = %d.%02d\n\r", tmpInt1, tmpInt2);
    HAL_UART_Transmit(&huart1, (uint8_t *)str_tmp, sizeof(str_tmp), 1000);

    hum_value = Read_Humidity_Sensor(HTS221_I2C_ADDRESS);
    int humInt1 = hum_value;
    float humFrac = hum_value - humInt1;
    int humInt2 = trunc(humFrac * 100);
    snprintf(str_tmp, 100, "HUMIDITY = %d.%02d\n\r", humInt1, humInt2);
    HAL_UART_Transmit(&huart1, (uint8_t *)str_tmp, sizeof(str_tmp), 1000);
    HAL_Delay(2000);
}
/* USER CODE END 3 */
}
```


Step 16. Compile and run in debug mode, open tera term and setup the terminal, then we can see the value



The screenshot shows an IDE with a C program for an HTS221 sensor. The code is in `main.c` and includes `user_hts221.h` and `startup_stm321475vgtx.s`. The program initializes the HTS221 sensor and enters an infinite loop that reads temperature and humidity data, formats it as strings, and transmits it via UART. A Tera Term window (COM4) is open, showing the received data. The data is formatted as `TEMPERATURE = %d.%02d\n\r` and `HUMIDITY = %d.%02d\n\r`. The output shows temperature values around 24.09 to 24.12 and humidity values around 53.80 to 53.91.

```
129
130 // HTS221 sensor init
131 Humidity_Sensor_Init(HTS221_I2C_ADDRESS);
132 Temperature_Sensor_Init(TSENSOR_I2C_ADDRESS);
133 /* USER CODE END 2 */
134
135 /* Infinite loop */
136 /* USER CODE BEGIN WHILE */
137 while (1)
138 {
139     /* USER CODE END WHILE */
140
141     /* USER CODE BEGIN 3 */
142     temp_value = Read_Temp_Sensor(TSENSOR_I2C_ADDRESS);
143     int tmpInt1 = temp_value;
144     float tmpFrac = temp_value - tmpInt1;
145     int tmpInt2 = trunc(tmpFrac * 100);
146     snprintf(str_tmp, 100, "TEMPERATURE = %d.%02d\n\r", tmpInt1, tmpInt2);
147     HAL_UART_Transmit(&huart1, (uint8_t *)str_tmp, sizeof(str_tmp), 1000);
148
149     hum_value = Read_Humidity_Sensor(HTS221_I2C_ADDRESS);
150     int humInt1 = hum_value;
151     float humFrac = hum_value - humInt1;
152     int humInt2 = trunc(humFrac * 100);
153     snprintf(str_tmp, 100, "HUMIDITY = %d.%02d\n\r", humInt1, humInt2);
154     HAL_UART_Transmit(&huart1, (uint8_t *)str_tmp, sizeof(str_tmp), 1000);
155     HAL_Delay(2000);
156 }
157 /* USER CODE END 3 */
158 }
159
```

COM4 - Tera Term VT

```
File Edit Setup Control Window Help
TEMPERATURE = 24.09
HUMIDITY = 53.90
TEMPERATURE = 24.10
HUMIDITY = 53.80
TEMPERATURE = 24.10
HUMIDITY = 53.83
TEMPERATURE = 24.12
HUMIDITY = 53.91
```

Console × Problems Executables Debugger Console Memory

Embedded C Assignment 8 Debug [STM32 C/C++ Application] [pid: 806]

File download complete
Time elapsed during download operation: 00:00:01.004

Verifying ...