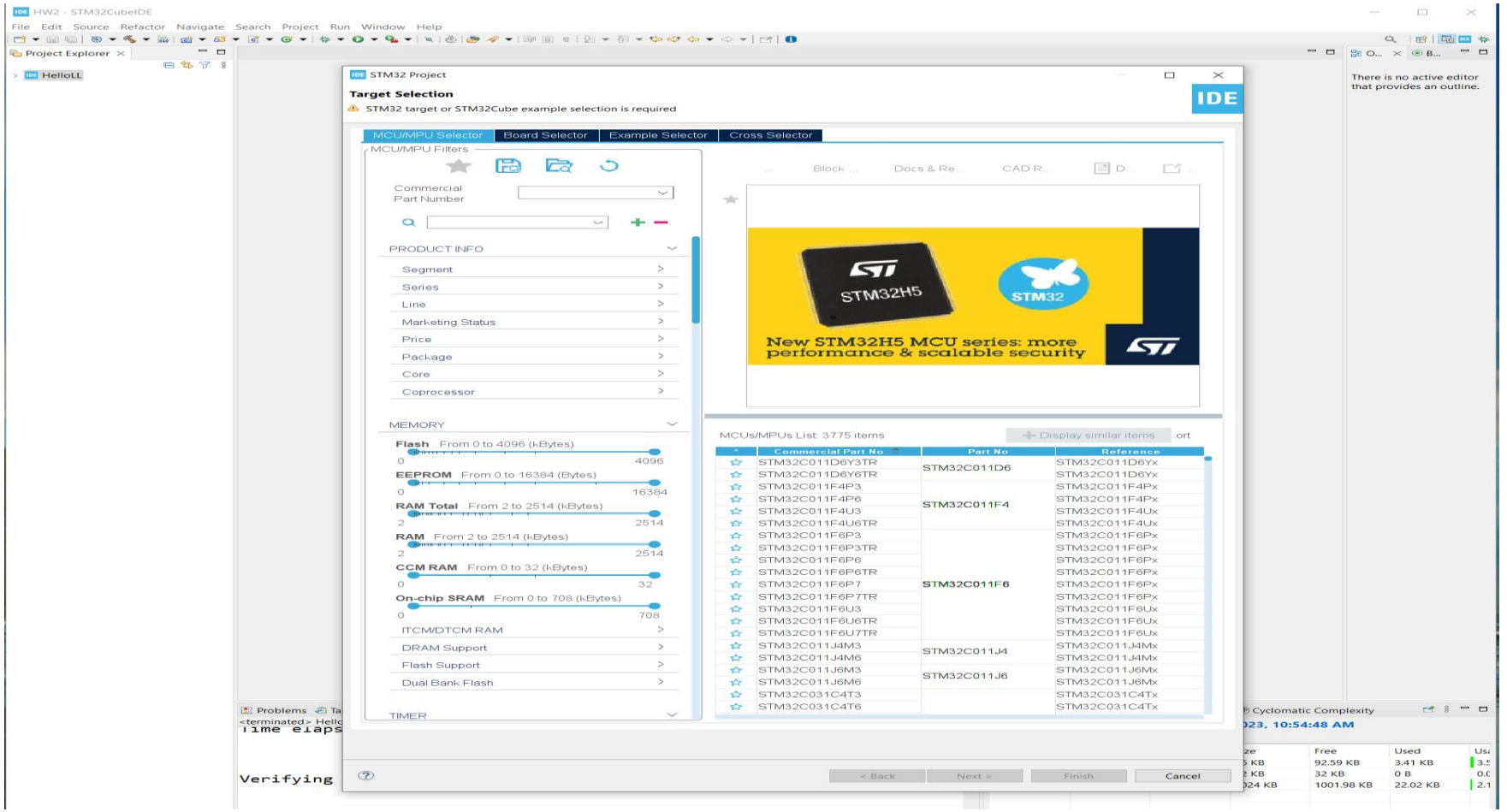
UCSD Embedded C Assignment 8

By

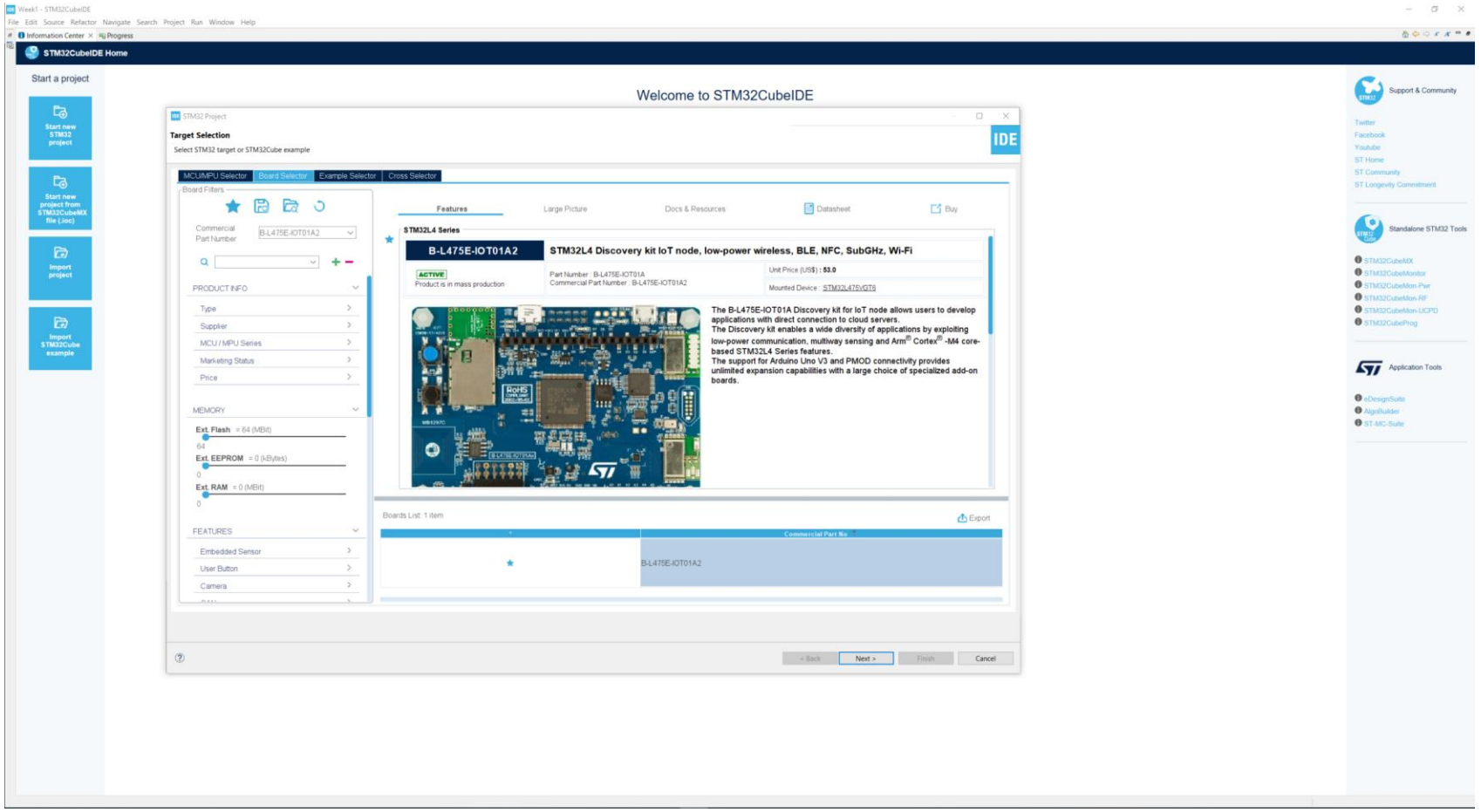
Hsuankai Chang

hsuankac@umich.edu

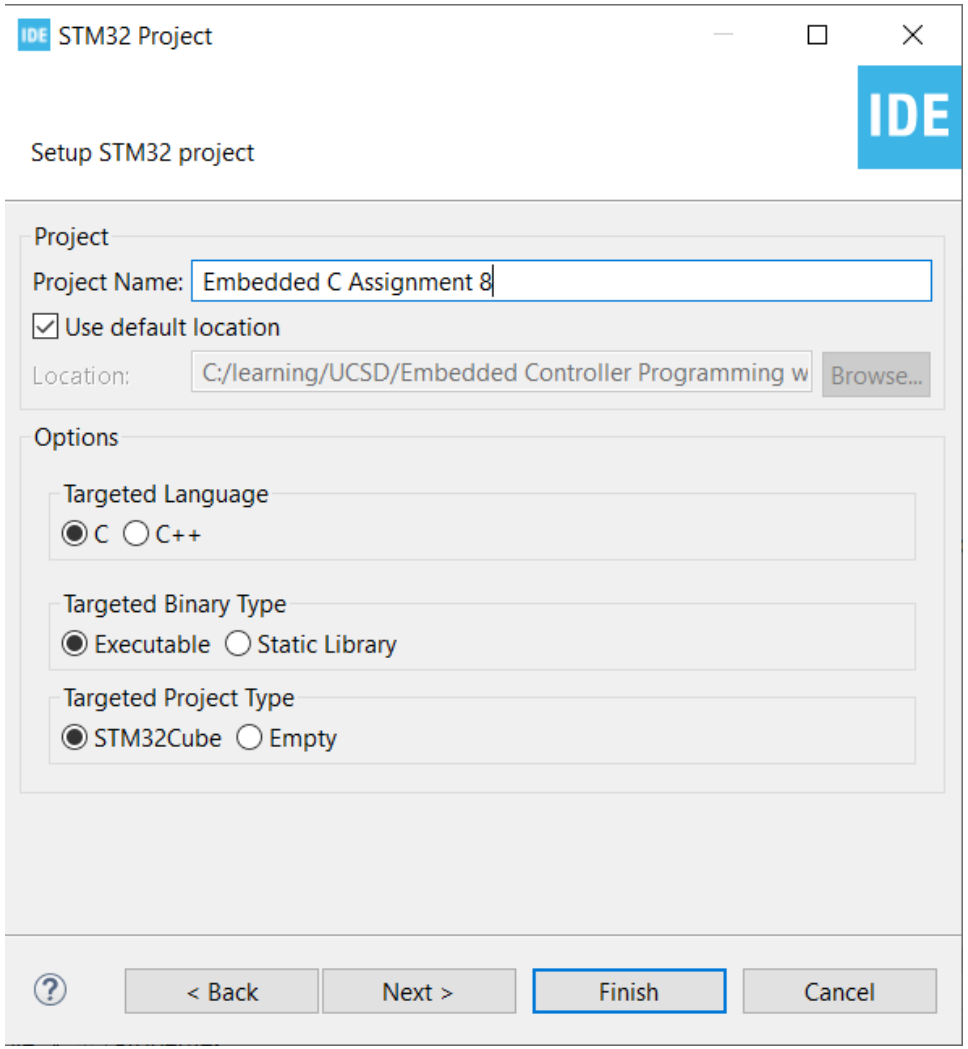
Step 1. Startup STM32CubeIDE and create new STM32 project



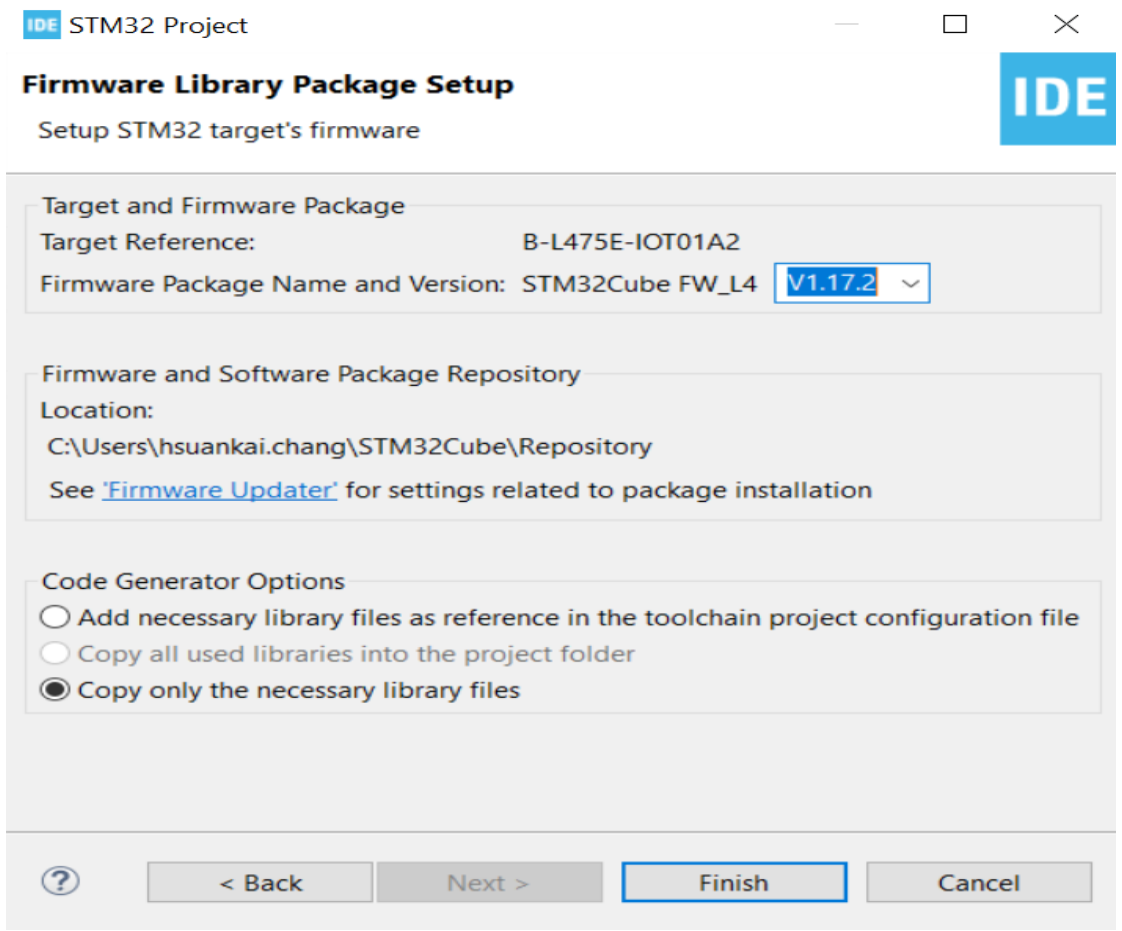
Step 2. Access board selector and type in the board you use, click Next



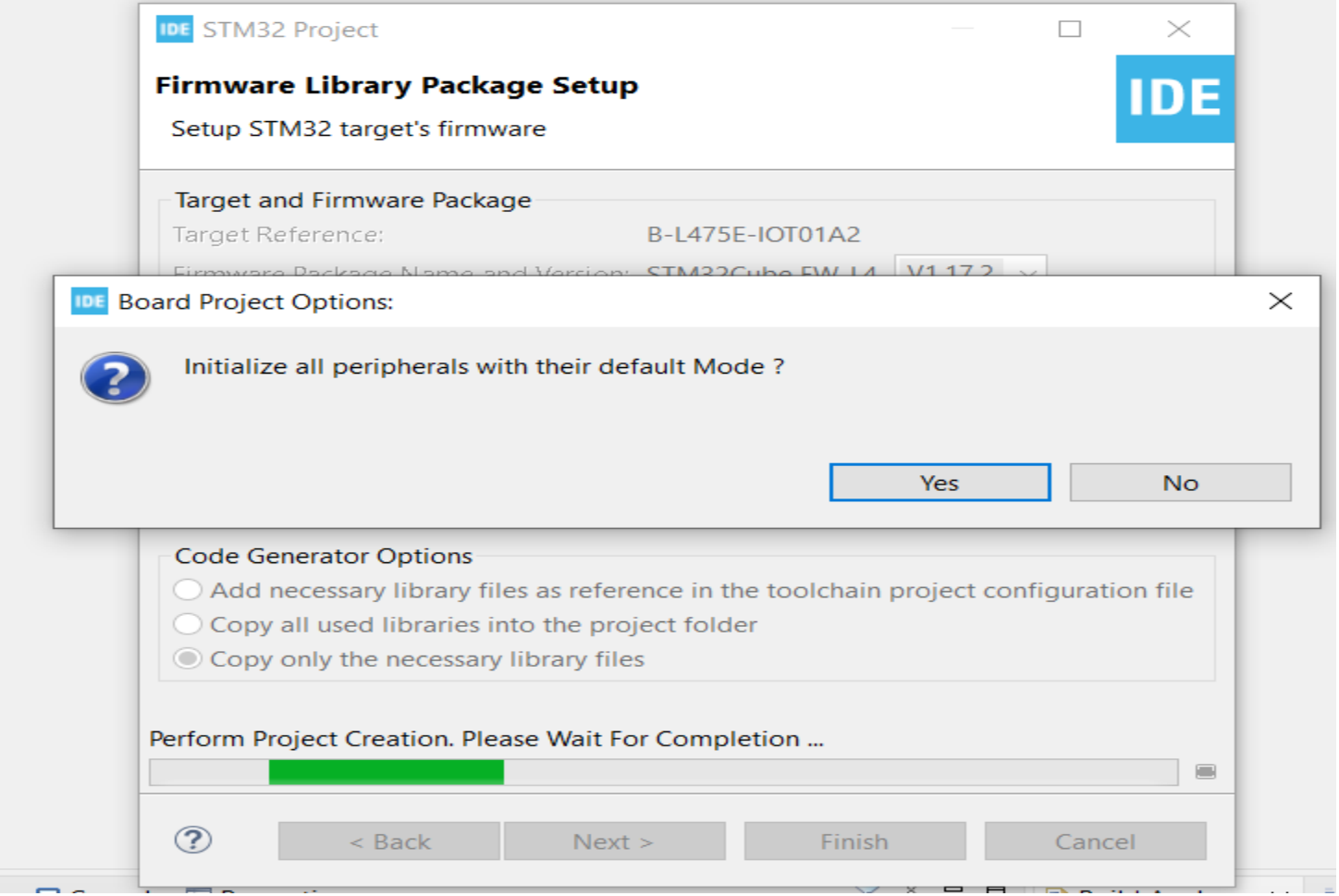
Step 3. Enter the project name then click Next



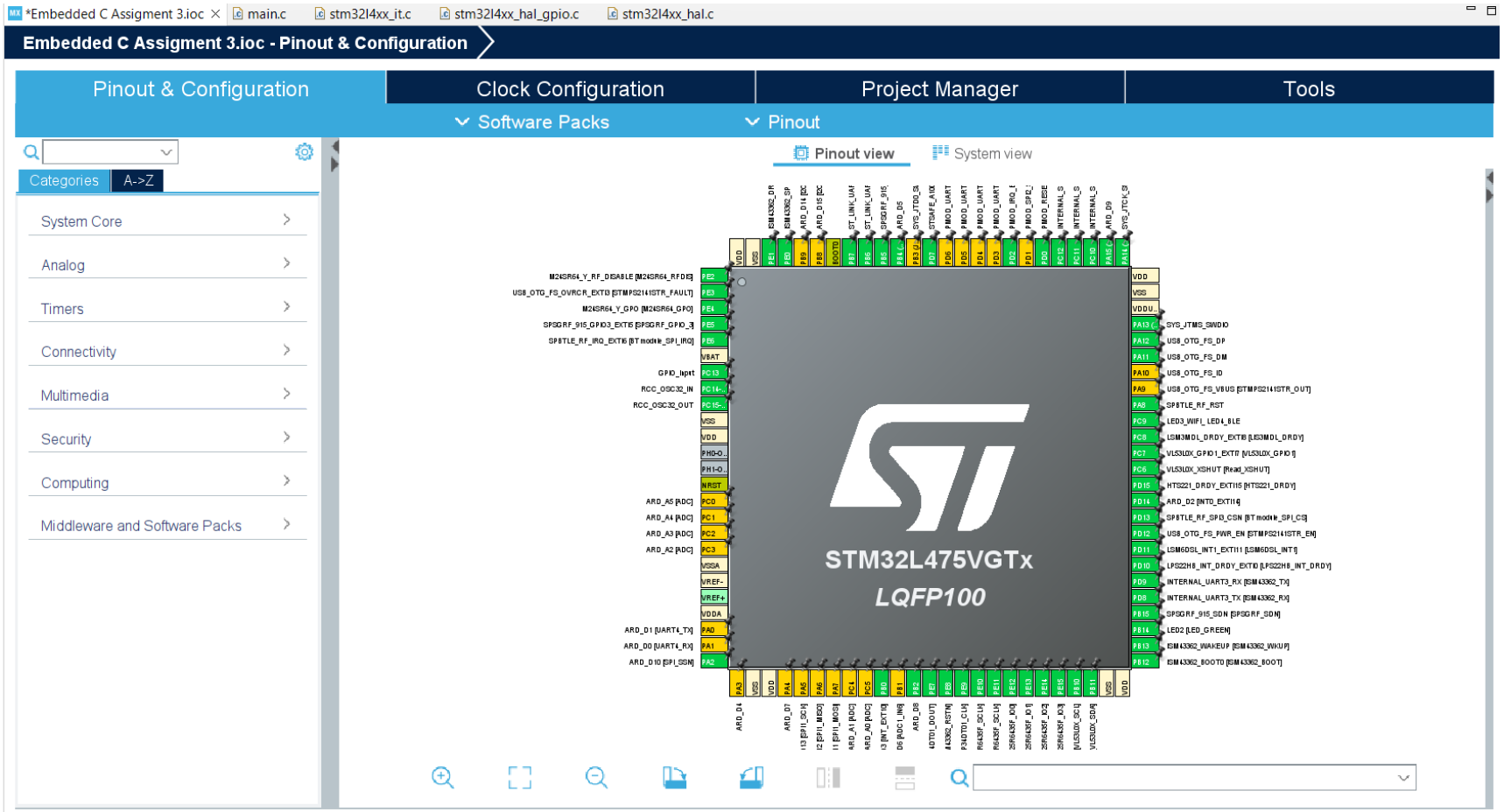
Step 4. See the firmware package name and version



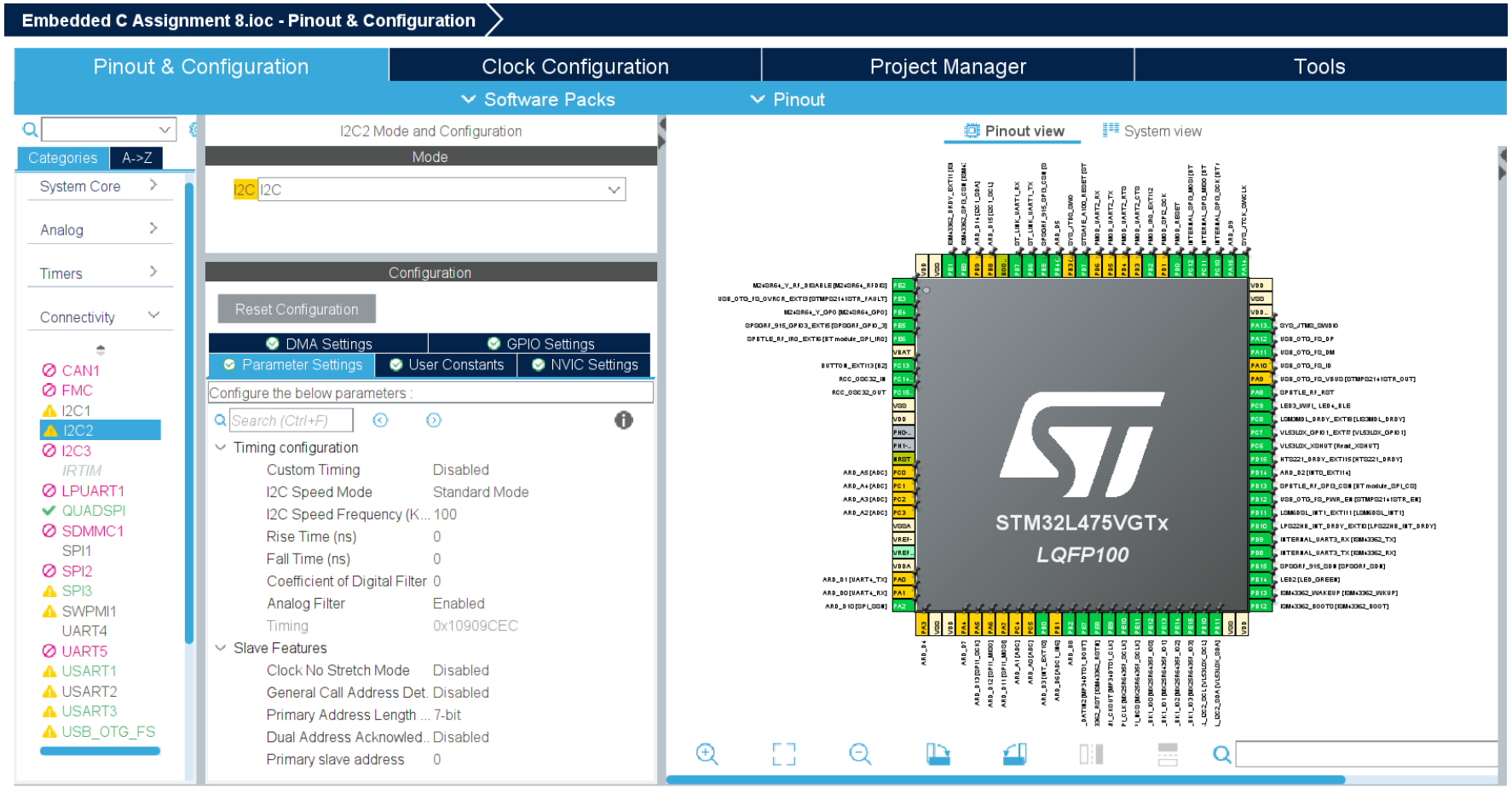
Step 5. Click yes to initialize all peripherals to default



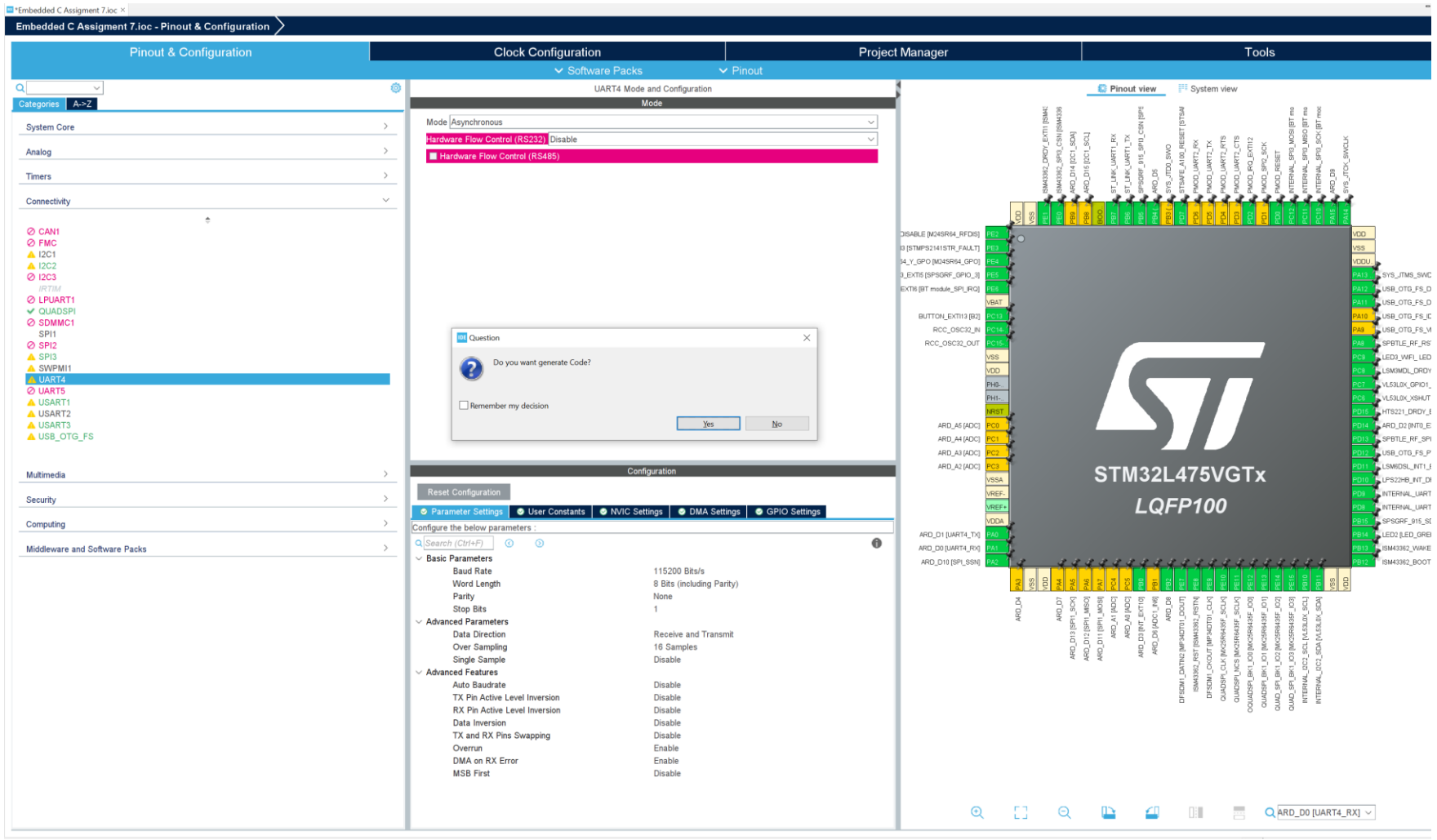
Step 6. When in .ioc file, click System view



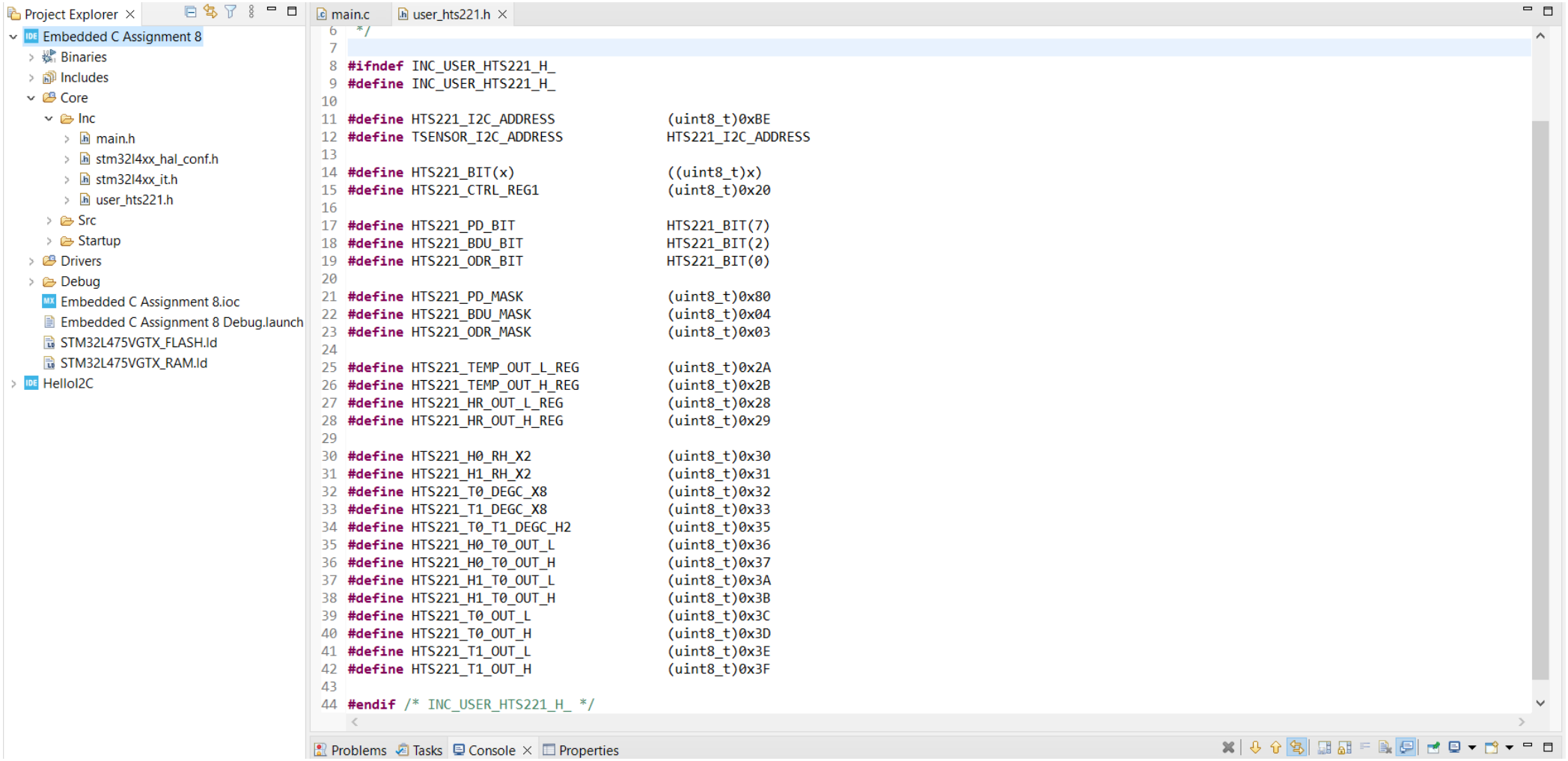
Step 7. Check the setting for I2C2 to communicate with HTS221 sensor



Step 8. Generate code



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



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

main.c × user_hts221.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 }
```

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

```
static void Temperature_Sensor_Init(uint16_t DeviceAddr)
{
     uint8_t tmp;

    /* Read CTRL_REG1 */
    tmp = Sensor_Read(DeviceAddr, HTS221_CTRL_REG1);

    /* Enable BDU */
    tmp &= ~HTS221_BDU_MASK;
    tmp |= (1 << HTS221_BDU_BIT);

    /* Set default ODR */
    tmp &= ~HTS221_ODR_MASK;
    tmp |= (uint8_t)0x01; /* Set ODR to 1Hz */

    /* Activate the device */
    tmp |= HTS221_PD_MASK;

    /* Apply settings to CTRL_REG1 */
    Sensor_Write_Multiple(DeviceAddr, HTS221_CTRL_REG1, tmp);
}

static void Humidity_Sensor_Init(uint16_t DeviceAddr)
{
    uint8_t tmp;

    /* Read CTRL_REG1 */
    tmp = Sensor_Read(DeviceAddr, HTS221_CTRL_REG1);

    /* Enable BDU */
    tmp &= ~HTS221_BDU_MASK;
    tmp |= (1 << HTS221_BDU_BIT);

    /* Set default ODR */
    tmp &= ~HTS221_ODR_MASK;
    tmp |= (uint8_t)0x01; /* Set ODR to 1Hz */

    /* Activate the device */
    tmp |= HTS221_PD_MASK;

    /* Apply settings to CTRL_REG1 */
    Sensor_Write_Multiple(DeviceAddr, HTS221_CTRL_REG1, tmp);
}
```

Step 14. Implement read data from temperature and humidity sensor

```c
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;
}
```

```c
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