Interfacing a Temperature & Humidity Sensor

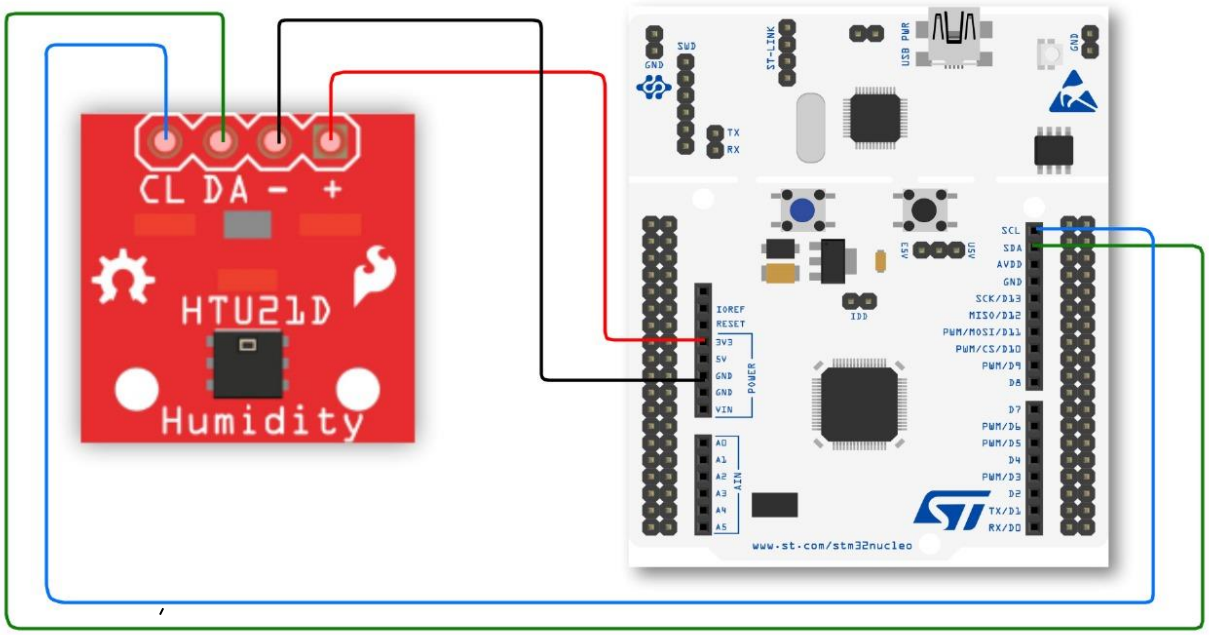
Objective:

The Objective of this experiment is to interface a temperature & humidity sensor to an STM32 microcontroller.

Requirements:

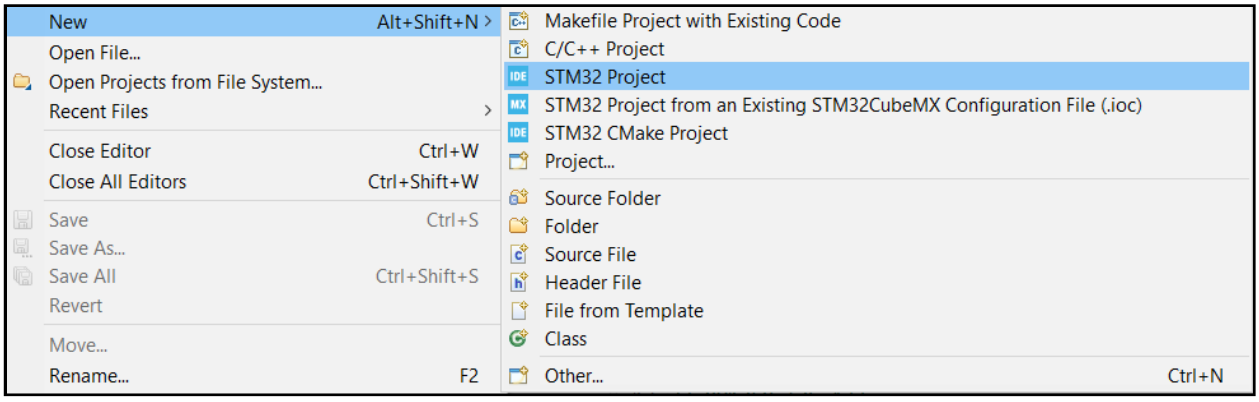
1. STM32 Cube IDE software.
2. HTU21DF Sensor (I2C).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.

Connection Diagram:

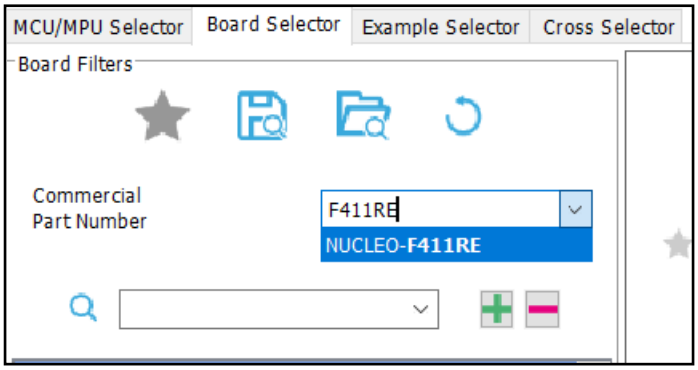


Procedure:

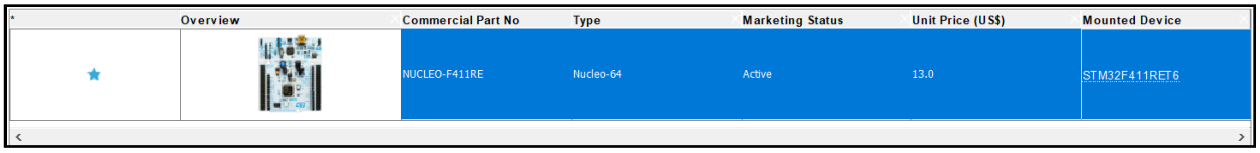
1. Click on **File→New→STM32 Project** to start your project on Cube IDE.



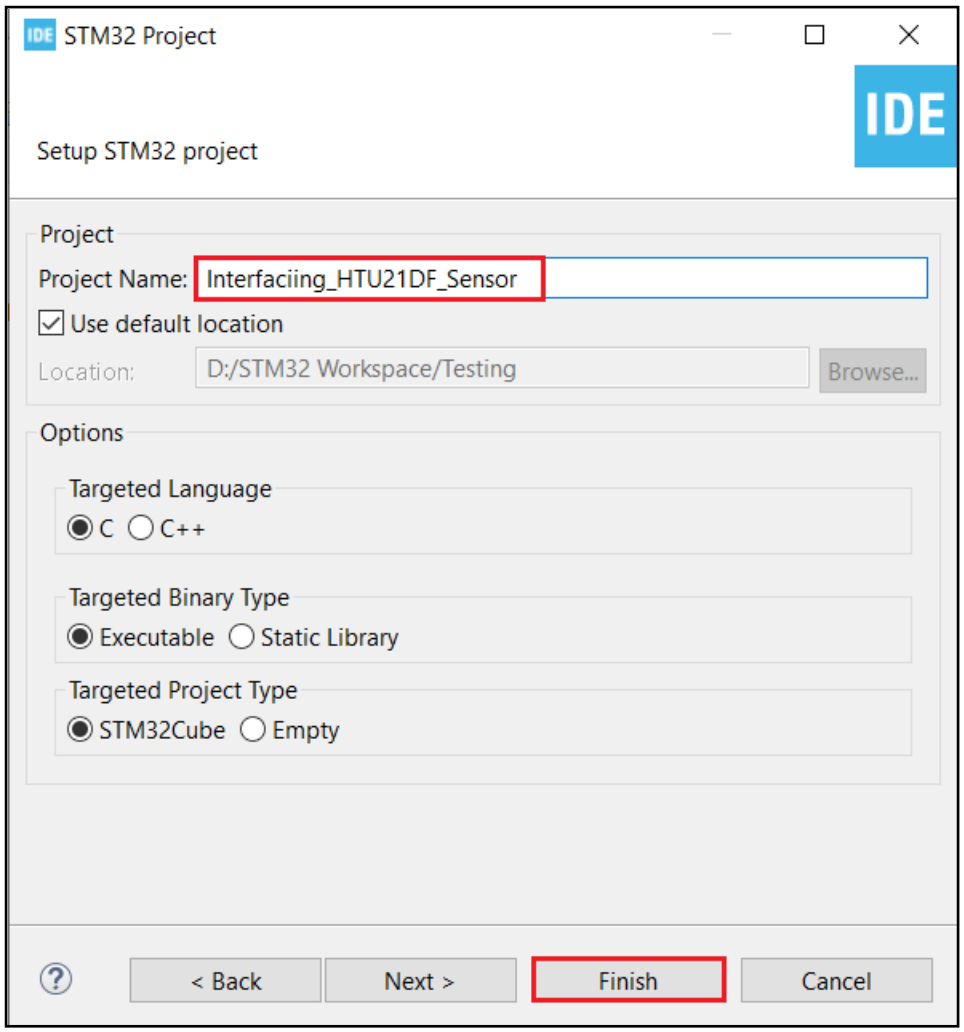
2. A **Target Selection** window will open. Click on **Board Selector**, where you need to select the microcontroller board you are working with.
   (**NB:** If you are having Nucleo-F401RE, you have to select the said Commercial Part Number)



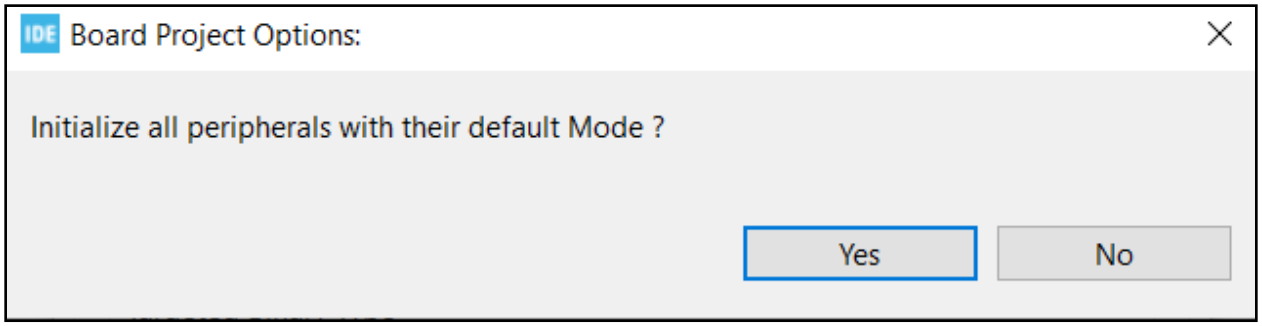
3. After this on the right-hand side of the window, under **Board List** you will see the board you have selected. Click on the board and then click on **Next.**



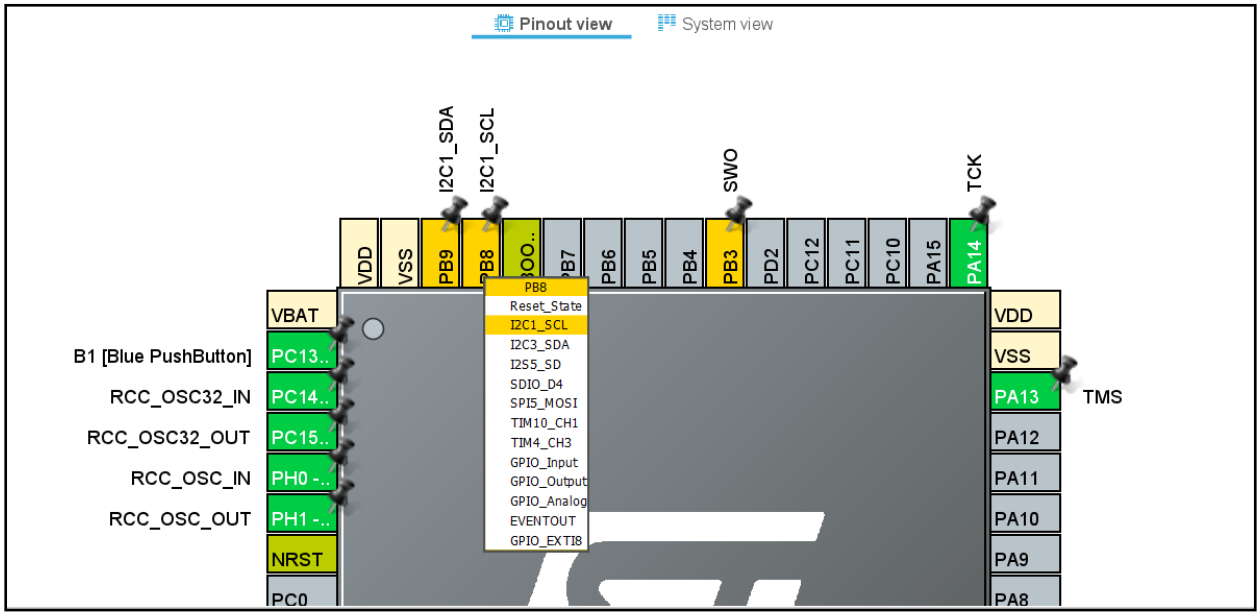
4. Give your project a name, rest of the things will remain by default as it is for now. Click on **Finish.**



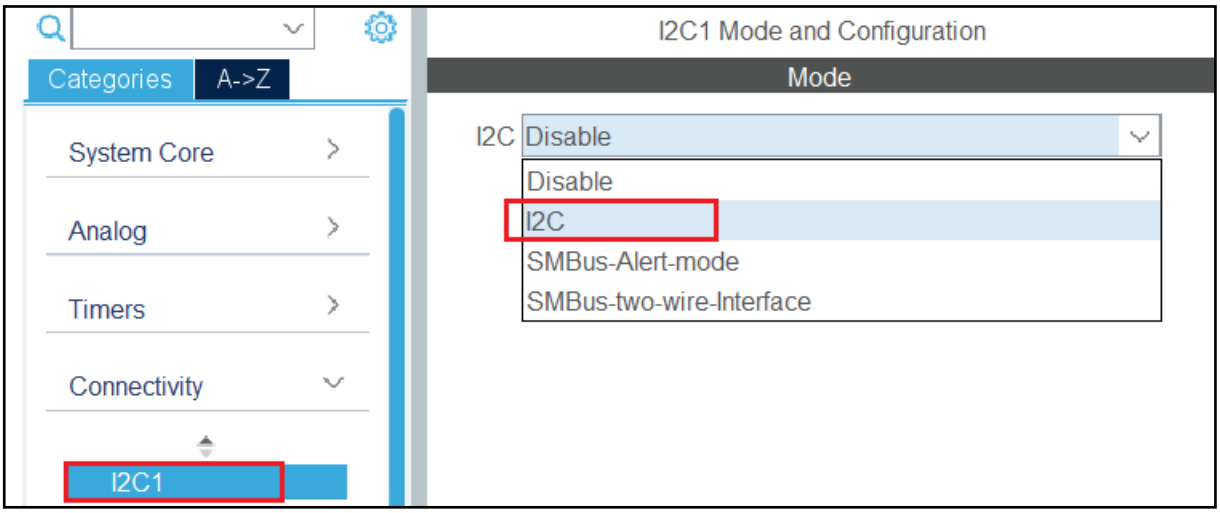
5. Cube IDE will ask if you want to initialize all peripherals with their default mode, click on **Yes.**



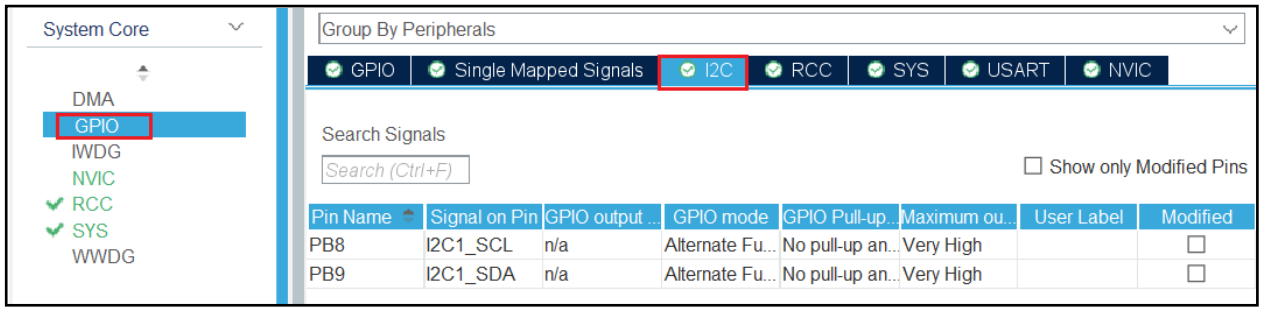
6. In the **Pinout & Configuration** tab, click on **PB8** pin and select it as an **I2C1_SCL** and **PB9** pin as an **I2C1_SDA**.



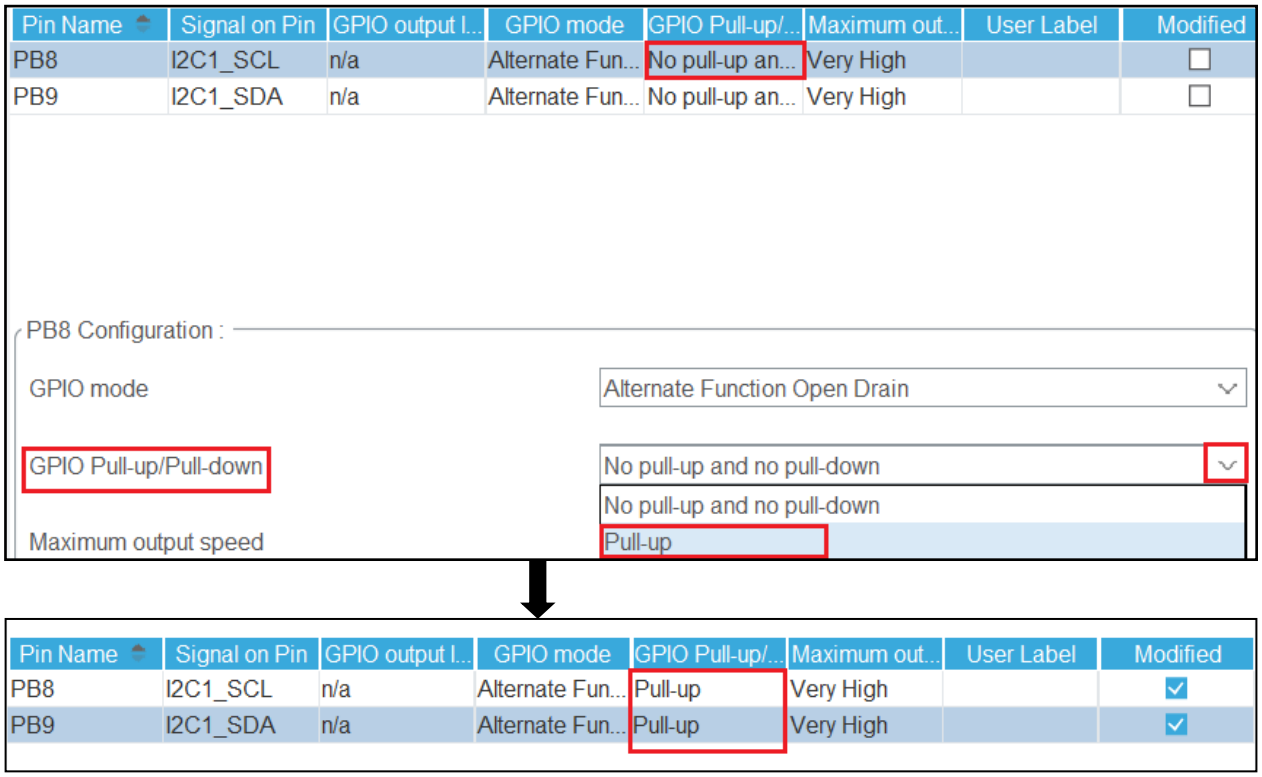
7. Next on the left-hand side under **Categories →Connectivity**, select I2C1 and enable it.



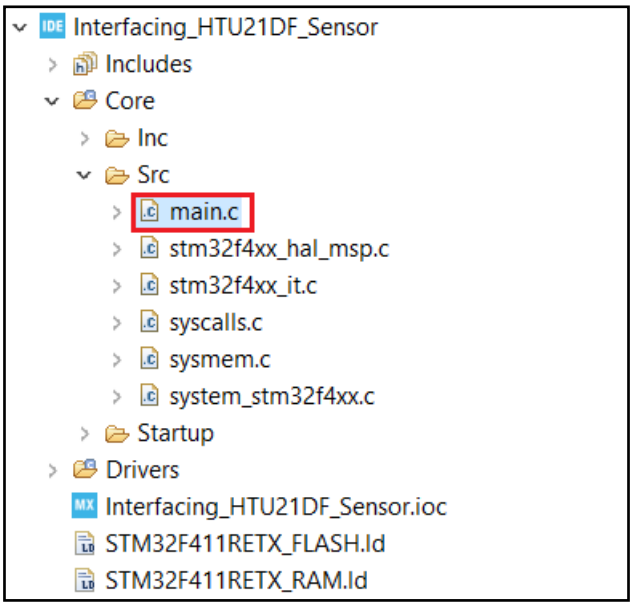
8. Now go to **System Core → GPIO** and select **I2C** option.



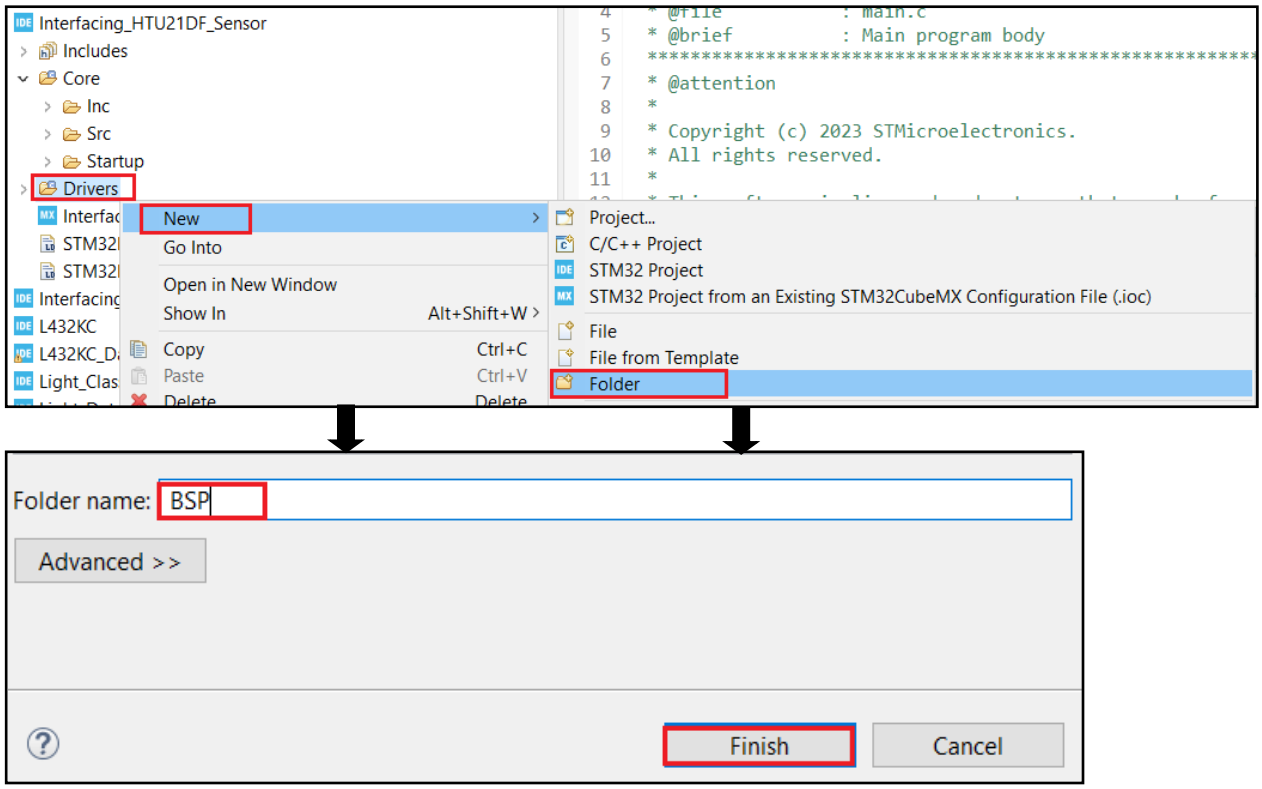
9. After selecting **I2C** option, change the GPIO pin **PB8 and PB9** as **Pull-up.**



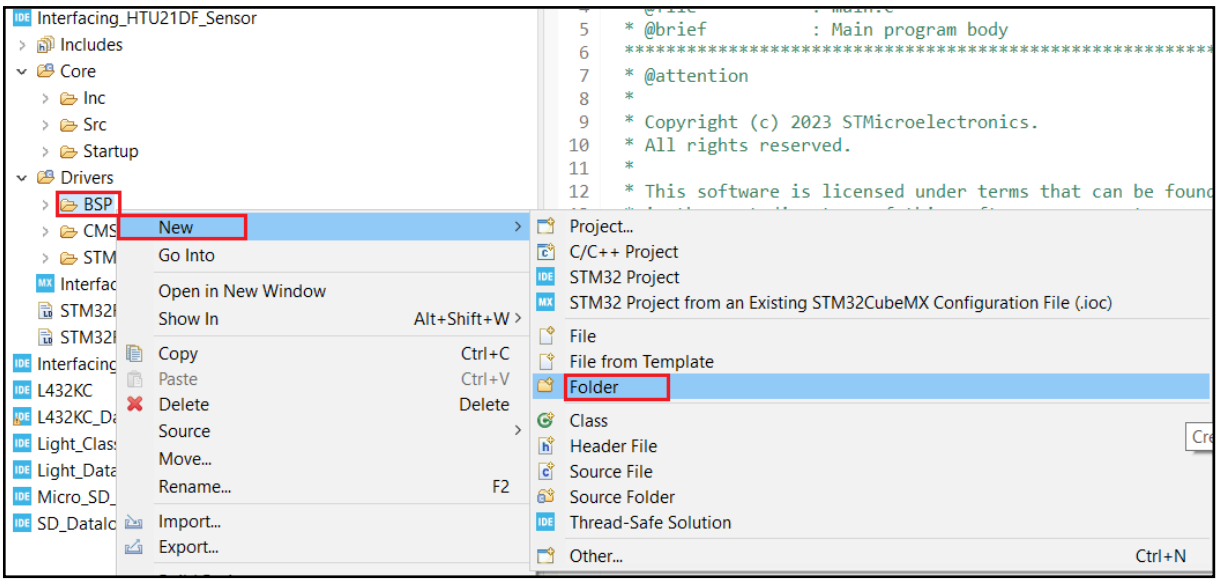
10. Press C**trl+S** to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example, I have named my project as (Interfacing_HTU21DF_Sensor) **Interfacing_HTU21DF_Sensor→Core →Src→main.c (**double click to load the code).



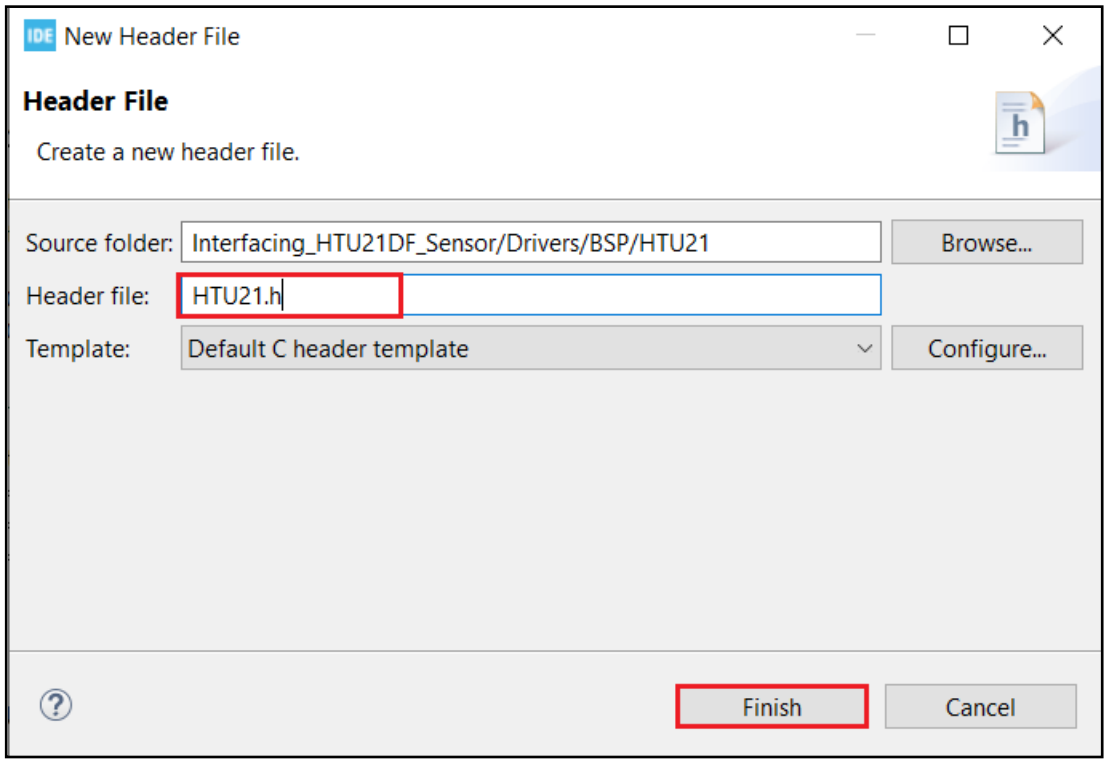
11. Open your project tree **Interfacing_HTU21DF_Sensor.** Right click on your **Drivers** folder and create a new **Folder** name as BSP.



12. Now right click on the **BSP** folder and create a new folder name as **HTU21.**



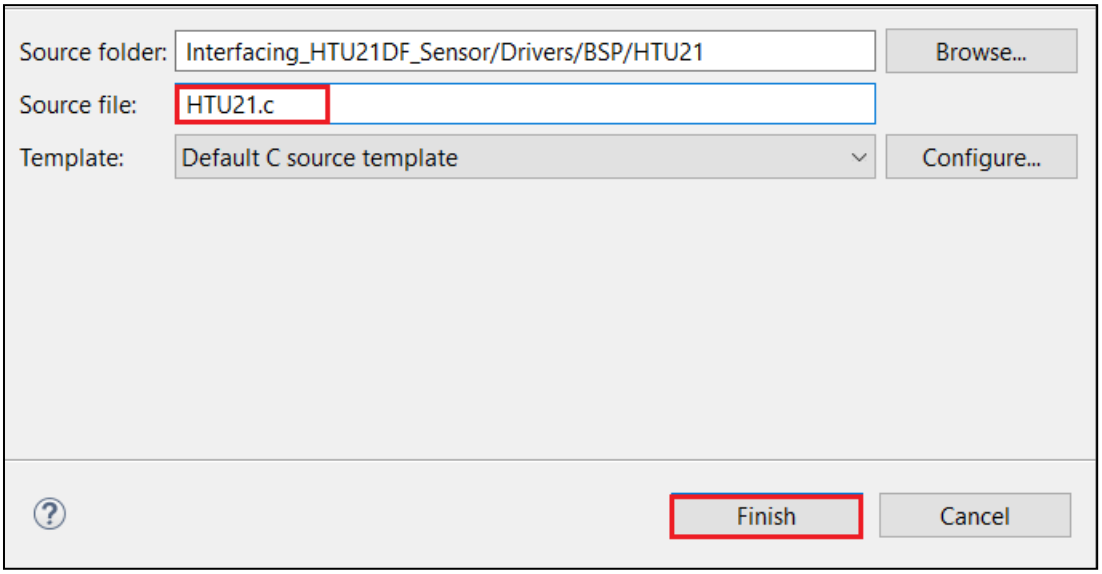
13. 13. Right click on the HTU21 folder and create a new Source file and the name the file as **HTU21.h** and select on **Finish.**



14. Below is the code snippets, please put your code in the appropriate places in the **HTU21.h** file.

```c

#ifndef SRC_HTU21_H_
#define SRC_HTU21_H_

#include "stm32f4xx_hal.h"
extern I2C_HandleTypeDef hi2c1;

/** Default I2C address for the HTU21D. */
#define HTU21_I2CADDR         (0x80)

/** Read temperature register. */
#define HTU21_READTEMP        (0xE3)

/** Read humidity register. */
#define HTU21_READHUM         (0xE5)

/** Write register command. */
#define HTU21_WRITEREG        (0xE6)

/** Read register command. */
#define HTU21_READREG         (0xE7)

/** Reset command. */
#define HTU21_RESET           (0xFE)

#ifdef __cplusplus
extern "C" {
#endif
char    HTU21_Init(void);
float   HTU21_GetTemp(void);
float   HTU21_GetHum(void);
void    HTU21_Reset(void);
#ifdef __cplusplus
}
#endif

#endif /* SRC_HTU21_H_ */

```

15. Right click on the HTU21 folder again and create a new source file and the name the file as **HTU21.c** and select on **Finish.**



16. Below is the code snippets, please put your code in the appropriate places in the **HTU21.c** file.

```c
#include "HTU21.h"

void HTU21_Reset()
{
    uint8_t Data=HTU21_RESET;

    HAL_Delay(15);
    HAL_I2C_Master_Transmit(&hi2c1, HTU21_I2CADDR, &Data, 1, 1000);
    HAL_Delay(15);

    }

char HTU21_Init()
{
    uint8_t check;
    uint8_t Data=HTU21_READREG;

    HAL_I2C_Master_Transmit(&hi2c1, HTU21_I2CADDR, &Data, 1, 1000);
    HAL_I2C_Master_Receive(&hi2c1, HTU21_I2CADDR, &check, 1, 1000);

    return check;
    }
```

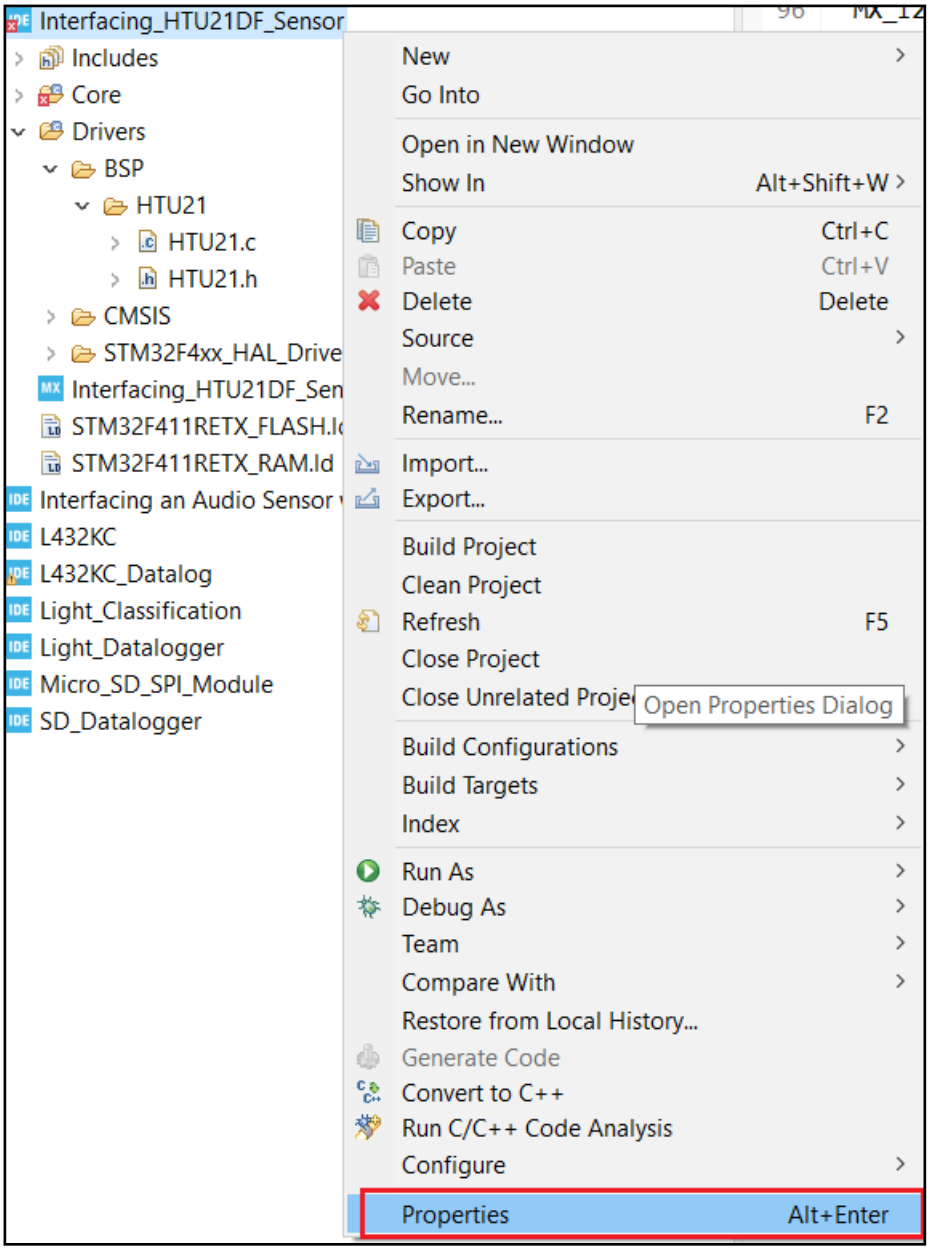
```c
float HTU21_GetTemp()
{
    uint8_t Temp_Data[2];
    uint8_t Data=HTU21_READTEMP;

    HAL_I2C_Master_Transmit(&hi2c1, HTU21_I2CADDR, &Data, 1, 1000);
    HAL_I2C_Master_Receive(&hi2c1, HTU21_I2CADDR, Temp_Data, 2, 1000);

    uint16_t Raw_Temp = (uint16_t)(Temp_Data[0] << 8 | (Temp_Data[1]& 0b11111100));

    float temp = Raw_Temp;
    temp *= 175.72f;
    temp /= 65536.0f;
    temp -= 46.85f;

    return temp;
    }

float   HTU21_GetHum()
{
    uint8_t Hum_Data[2];
    uint8_t Data=HTU21_READHUM;

    HAL_I2C_Master_Transmit(&hi2c1, HTU21_I2CADDR, &Data, 1, 1000);
    HAL_I2C_Master_Receive(&hi2c1, HTU21_I2CADDR, Hum_Data, 2, 1000);

    uint16_t Raw_Hum = (uint16_t)(Hum_Data[0] << 8 | (Hum_Data[1]& 0b11110000));

    float temp = Raw_Hum;
    temp *= 175.72f;
    temp /= 65536.0f;
    temp -= 46.85f;

    return temp;
    }
```

17. Cube IDE automatically generates a code format based on the configurations you have done. Cube IDE uses HAL libraries. Below is the code snippets, please put your code in the appropriate places in the **main.c** file.

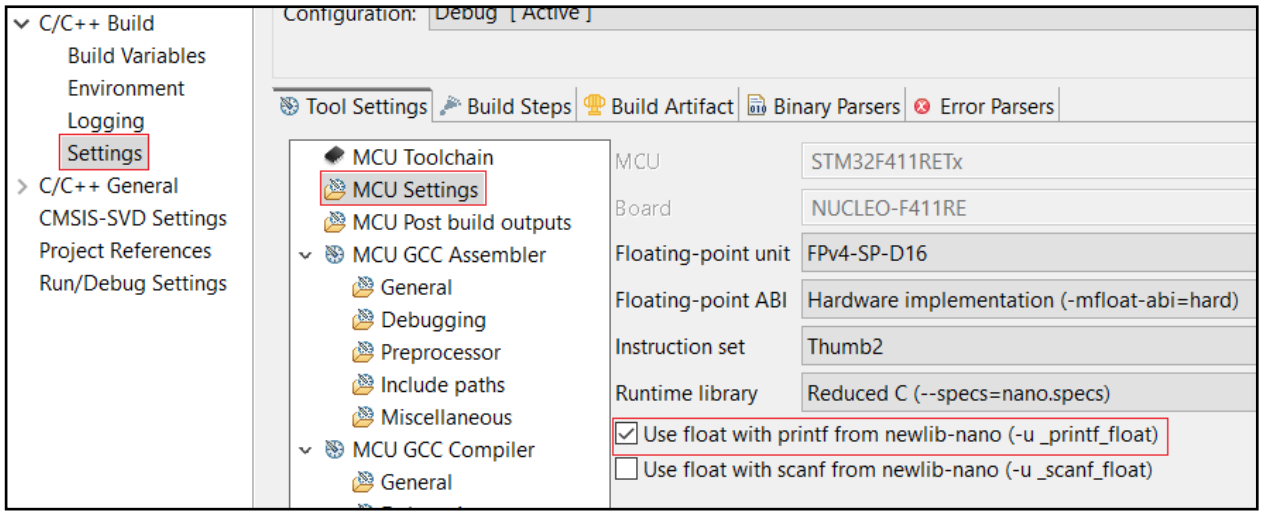
```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
#include "HTU21.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_I2C1_Init();
  /* USER CODE BEGIN 2 */
    HAL_Delay(1000);
    HTU21_Reset();

    while (HTU21_Init() != 0x2) {
        printf("Initializing HTU21 Sensor...!\r\n");
        HAL_Delay(100);
    }
    printf("HTU21 Sensor Initialization Success!\r\n");
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        float HTU21_Temp_Value = HTU21_GetTemp();
        printf("Temperature = %.2f\r\n", HTU21_Temp_Value);
        HAL_Delay(300);

        float HTU21_Hum_Value = HTU21_GetHum();
        printf("Humidity = %.2f\r\n", HTU21_Hum_Value);
        HAL_Delay(300);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

```
/* USER CODE BEGIN 4 */
int __io_putchar(int ch) {
    HAL_UART_Transmit(&huart2, (uint8_t*) &ch, 1, HAL_MAX_DELAY);
    return ch;
}
/* USER CODE END 4 */
```

18. Now right click on your project tree, go to **Properties**.



19. After that select **Settings→MCU Settings** and enable the **float printf.**

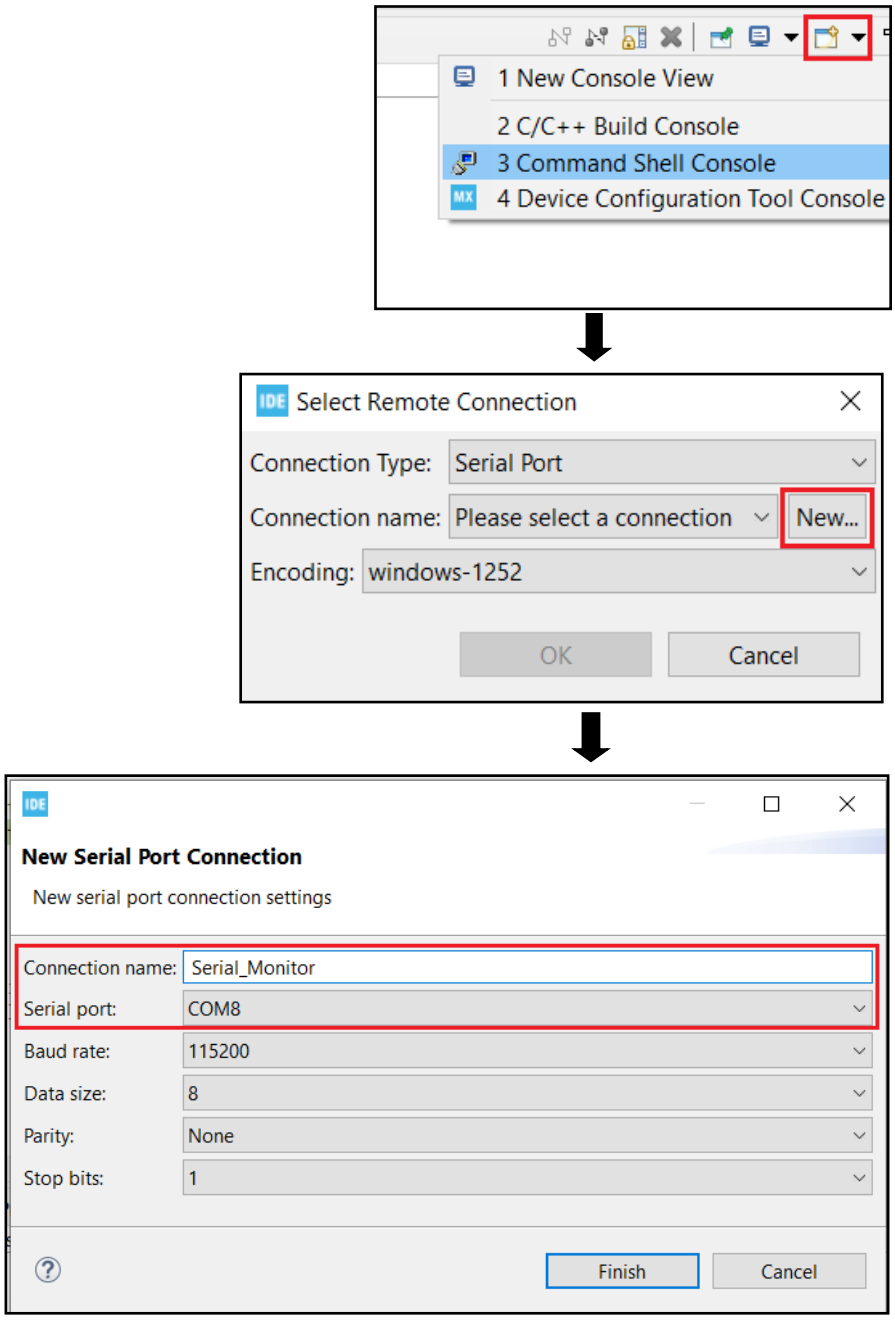


20. Now click on  the build symbol on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.

```
make -j4 all
arm-none-eabi-size   Interfacing_HTU21DF_Sensor.elf
   text    data     bss     dec     hex filename
  27612     476    2060   30148    75c4 Interfacing_HTU21DF_Sensor.elf
Finished building: default.size.stdout


14:33:44 Build Finished. 0 errors, 0 warnings. (took 736ms)
```

21. Next connect your STM32 board with your audio sensor connect to it to your PC and click on the **Debug**  icon to start the Debugging process. An **Edit Configuration** window will open, click on **OK,** without making any changes.

22. In the debug mode, go to the bottom right hand side corner, click on open console. Select the **Connection Type** as **Serial Port**, then click on **New.** In the new window, in **Connection name** give some name to your new connection, and select the **Serial port** correctly. Then click on **Finish** and then **Ok.** A console with the given name will be opened at the bottom of your screen.



23. Click on the **Resume** icon  to run your code. You should be able to see the value of light sensor in the **Console**.

```
Humidity = 44.53
Temperature = 28.16
Humidity = 44.53
Temperature = 28.16
Humidity = 44.57
Temperature = 28.16
Humidity = 44.57
Temperature = 28.15
Humidity = 44.66
Temperature = 28.16
Humidity = 44.66
Temperature = 28.15
Humidity = 44.66
Temperature = 28.14
Humidity = 44.70
Temperature = 28.15
Humidity = 44.74
Temperature = 28.15
Humidity = 44.74
Temperature = 28.13
Humidity = 44.79
Temperature = 28.12
Humidity = 44.79
Temperature = 28.12
Humidity = 44.79
Temperature = 28.11
```

24. Before moving out of the debugging mode, click on **Disconnect**  and close the console then click on the **Terminate**  icon. You will be moved out of the debugging mode.

Note: All important steps and parts are highlighted with a red colour box for the proper understanding of the user. This document is for the use of education purpose only.