**Assignment – 6**

**Exercise 1 gives you all necessary API so that you start receiving all simulated sensor data on ST BLE Sensor App. Make change in the program so that it can send actual HTS221 temperature sensor data rather than simulated data. Hind: Include necessary driver files as mentioned in Lab7T1. Add necessary API/helper function in app\_bluenrg\_ms.c**

**Implementation:**

**Step-1**: Start a new STM32Cube project

Go to file 🡺 New 🡺 STM32 Project

Graphical user interface, text, application, email

Description automatically generated

This will generate a popup box, go to Board Selector. In commercial part number enter your stm32 specific part name and select the board and next.

Graphical user interface, application

Description automatically generated

This will generate a new popup for creating a new project of the specified name and project setting 🡺 finish and for Initializing all peripherals with their default Mode select No.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

**Step-2:** Open Device Configuration Wizard

**A picture containing table

Description automatically generated**

**Step-3**: Select all configurations as shown above and click OK. It’ll add one additional Software Packs in configuration wizard**.**

1. Install STMicroelectronics X-Cube-BLE1
2. In Wireless BlueNRG-MS
   1. Select BlueNRG-MS Controller and Utils
   2. HCI\_TL 🡺 Basic
   3. HCI\_TL\_INTERFACE 🡺 UserBoard
3. In Device BLE Application
   1. Application 🡺 SensorDemoBLE

**Graphical user interface, table

Description automatically generated**

**Step-4:** Click on the newly added software pack and Enable both Wireless and Device BLE1 Application Mode.

Graphical user interface, text, application

Description automatically generated

Enable SPI3 Mode 🡺 Full Duplex Master as out SPBTLE-RF module is interfaced with protocol

Graphical user interface, text, application

Description automatically generated

Also enable USART1 for debugging, MODE 🡺 Asynchronous

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, email

Description automatically generated

**Step-5:** Install the ST BLE Sensor and couple your device. The temperature and pressure will appear, but they will be random numbers. To get the required parameters, follow the steps below.

**Step-6:** Include the Temperature and pressure file from BSP and configure app\_bluenrg\_ms.c for getting temperature and pressure from inbuilt sensors in stm32.

Include hts221 and lps22hb .c and .h file in Drivers 🡺 BSP 🡺 Components

And Include temperature and Pressure sensor related files in Drivers 🡺 BSP 🡺 B-L475-IOT01A1

And make sure to include these paths as following.

1. Alt+Enter or right click on project name and go to properties this will open a popup for properties of your BLE App.
2. C/C++ Build 🡺 Settings 🡺 MCU GCC Compiler 🡺 Include Paths

Graphical user interface, text, application

Description automatically generated

Add paths if not included and apply and close.

**Code:**Make changes only in app\_bluenrg\_ms.c file as following

**Step-7**: Add Print statement for printing the temperature and pressure values.

Graphical user interface, text, application

Description automatically generated

/\* USER CODE BEGIN Includes \*/

**#include** <stdio.h>

**#include** "stm32l475e\_iot01.h"

**#include** "stm32l475e\_iot01\_tsensor.h"

**#include** "stm32l475e\_iot01\_psensor.h"

**#include** <math.h>

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN PV \*/

**float** temp\_value = 0; // Measured temperature value

//float humid\_value = 0; // Measured humidity value

**float** pres\_value = 0; // Measured humidity value

uint8\_t msg1[] = "\r\n";

uint8\_t msg2[] = "Temperature ";

uint8\_t msg3[] = "Pressure ";

**Step-8**: Initialized Temperature and pressure

/\* USER CODE END PV \*/

**static** **void** **User\_Init**(**void**)

{

BSP\_PB\_Init(BUTTON\_KEY, *BUTTON\_MODE\_EXTI*);

BSP\_LED\_Init(*LED2*);

BSP\_COM\_Init(*COM1*);

BSP\_TSENSOR\_Init();

BSP\_PSENSOR\_Init()}

Graphical user interface, text, application, email

Description automatically generated

**Step-9:** Include the formula for temperature and Pressure to get the accurate result.

Text

Description automatically generated

/\*\*

\* @brief Set random values for all environmental sensor data

\* @param float pointer to temperature data

\* @param float pointer to pressure data

\* @retval None

\*/

**static** **void** **Set\_Random\_Environmental\_Values**(**float** \*data\_t, **float** \*data\_p)

{

temp\_value = BSP\_TSENSOR\_ReadTemp();

**int** tmpInt1 = temp\_value;

**float** tmpFrac = temp\_value - tmpInt1;

**int** tmpInt2 = **trunc**(tmpFrac \* 100);

\*data\_t = temp\_value;

//snprintf(data\_t, 100, "%d.%02d", tmpInt1, tmpInt2);

//printf("data\_t: %s\n", data\_t);

**printf**("temperature: %d.%02d\n",tmpInt1, tmpInt2);

pres\_value = BSP\_PSENSOR\_ReadPressure();

**int** presInt1 = pres\_value;

**float** presFrac = pres\_value - presInt1;

**int** presInt2 = **trunc**(presFrac \* 100);

\*data\_p = pres\_value;

//snprintf((char \*)data\_p, 100, "%d.%02d", presInt1, presInt2);

//printf("data\_p: %s\n", data\_p);

**printf**("Pressure: %d.%02d\n",presInt1, presInt2);

}

**Result:**

**Paired your device Mobile app(St Ble)**

**Graphical user interface, text, application

Description automatically generated**

**Tera term Output** :

**Graphical user interface, text, application

Description automatically generated**