

## EXPERIMENT: 3

### STTS751 TEMPERATURE SENSOR INTERFACING WITH DEV BOARD/NODE

#### What will you learn from this module:

- Interfacing with the help of I2C protocol.
- Temperature measurement using STTS751 sensor and nrf dev board/node.
- Configuration of overlay file, device tree and prj file for enabling hardware device.

#### Requirements:

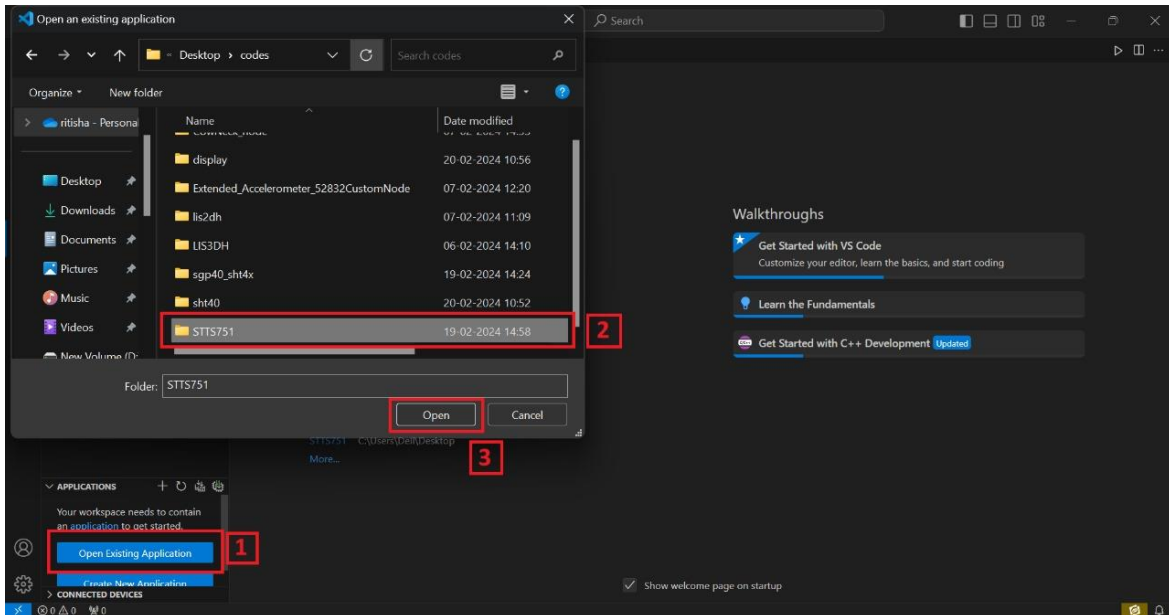
- nRF connect desktop software.
- nRF Command line tools.
- Visual studio code.
- USB cable.
- nRF52832 Development Board/Node.
- STTS751 Sensor.
- TTL Device.

#### Prerequisites:

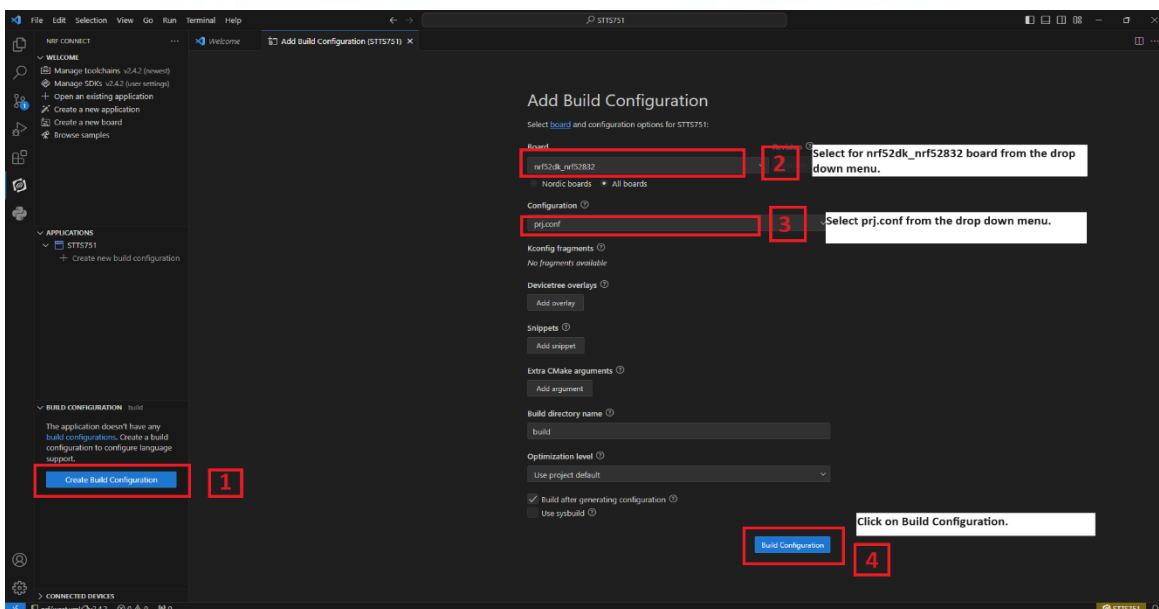
- Basic knowledge of C/C++.
- Basic knowledge of communication protocol.
- Basic project setup.

## Setup and Configuration:

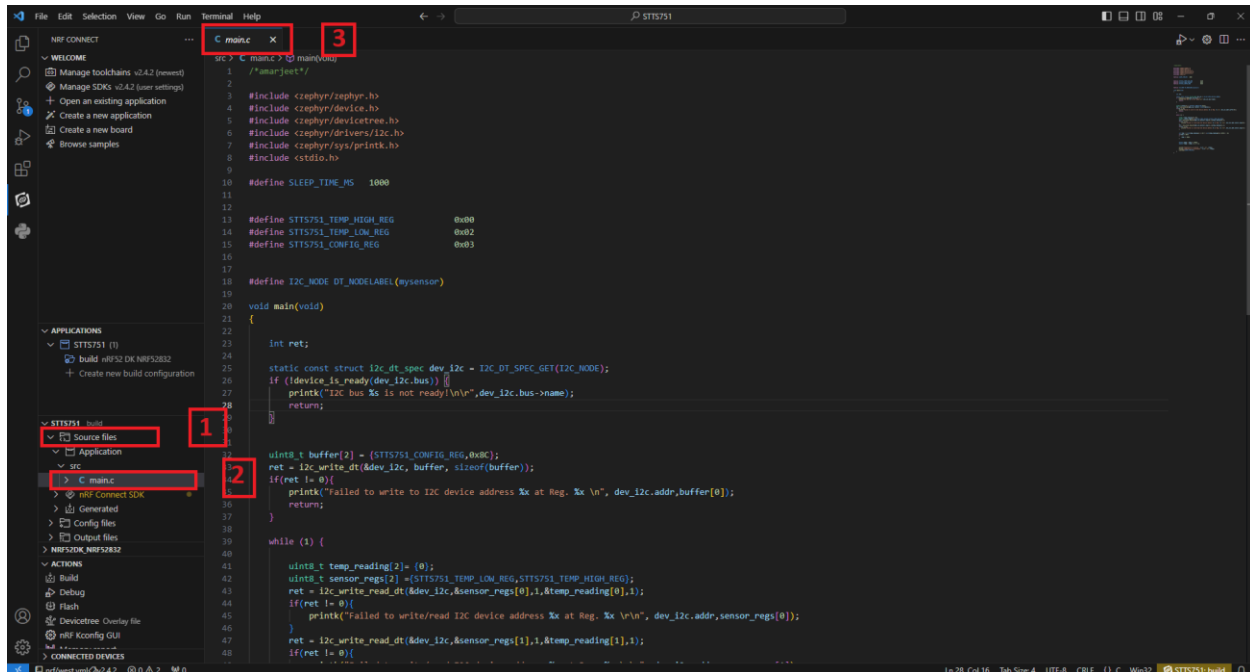
- Open VS Code and click on **Open Existing Application** [1] > click on **sht40** [2] > **Open** [3] as shown in the picture below.



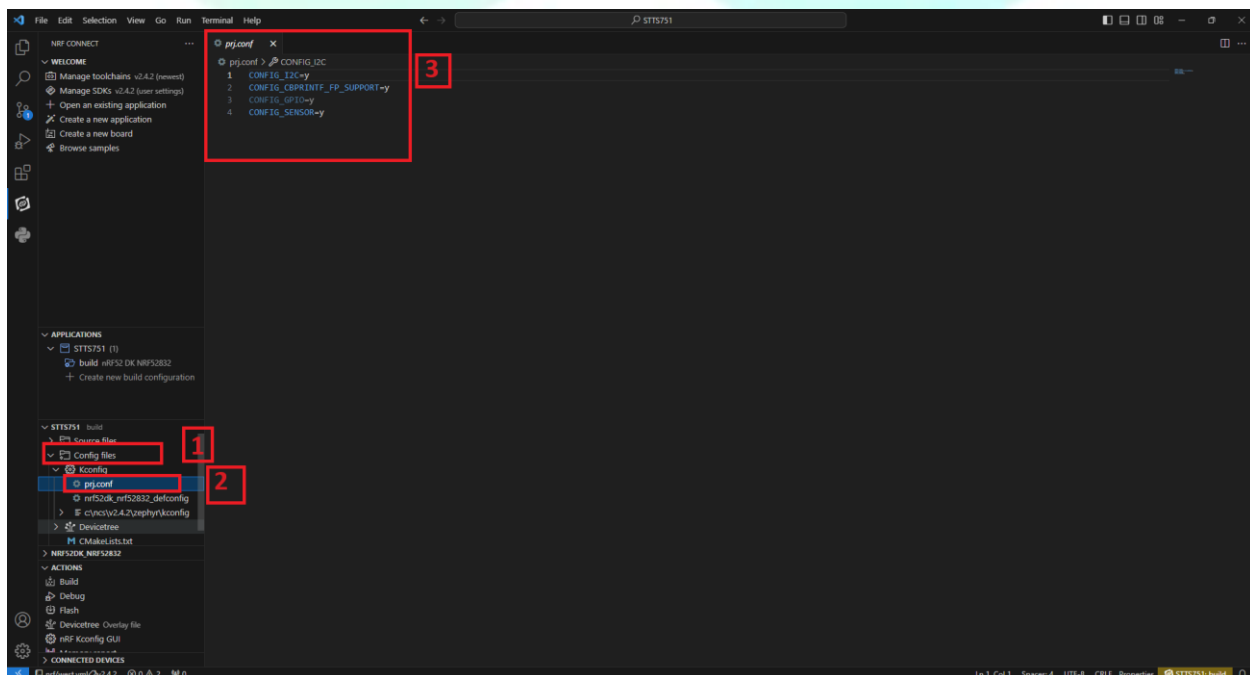
- Click on **Create new build configuration** [1]. Here you can change the board version, if you are using nRF52832, then select **nrf52dk\_nrf52832** [2] or you can change from dropdown menu for another version like nRF52833 etc.
- After that click on the Configuration and select **prj.conf** [3] from dropdown menu and then click on the **Build Configuration** [4] as shown below in the picture.



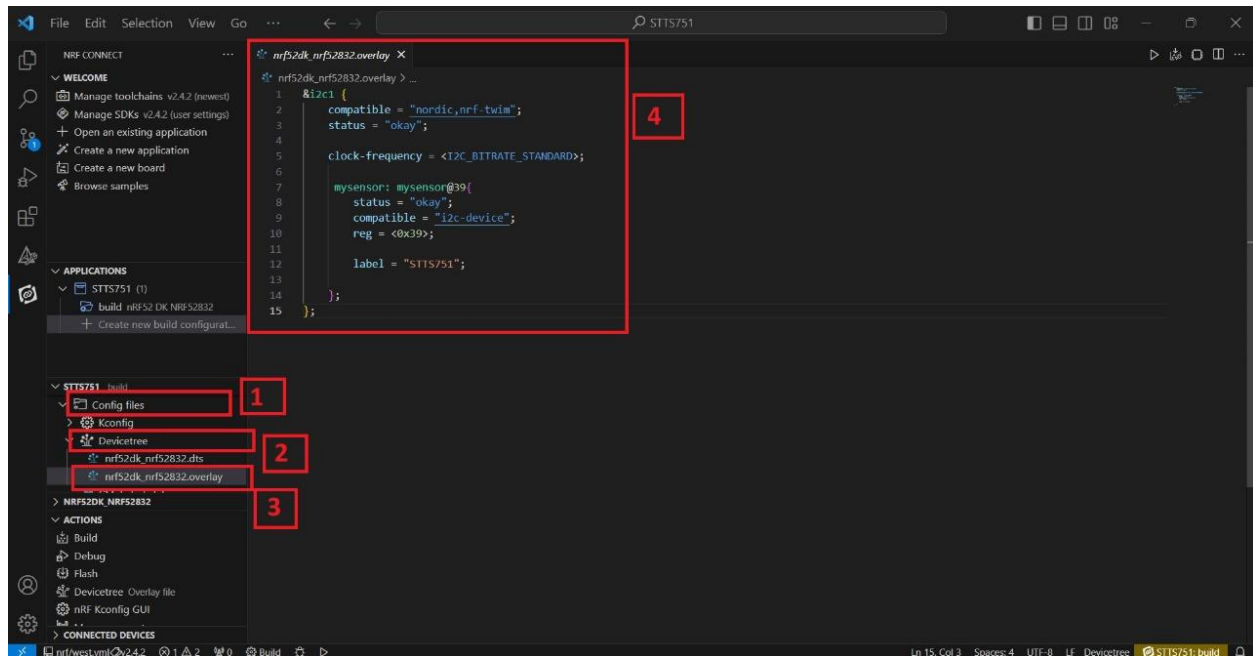
- Go to source file, click **source file [1]** > click on **Application** > click on **src** > click on **main.c [2]**.
- After Click on **main.c** file and you will see the code on your screen **[3]**.



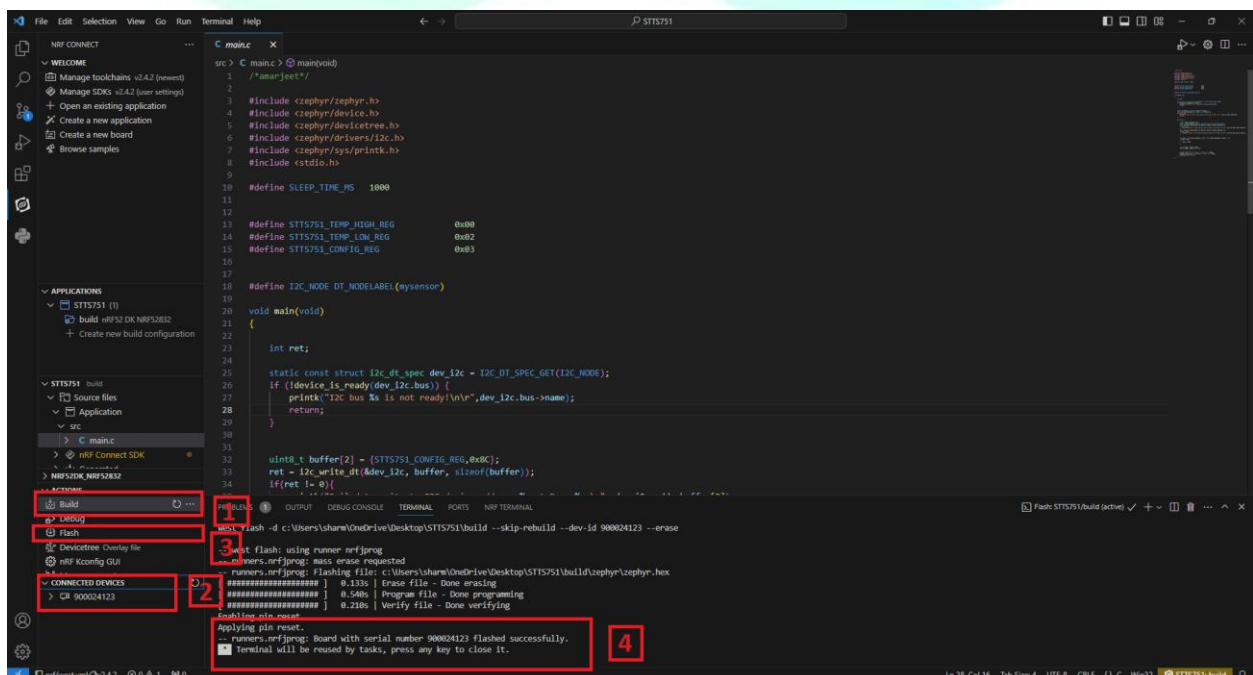
- To configure the prj configuration, click on **Config files [1]** > click on **Kconfig** > click on **prj.conf [2]**.
- The prj configuration will appear on your screen **[3]** as shown in the picture below.



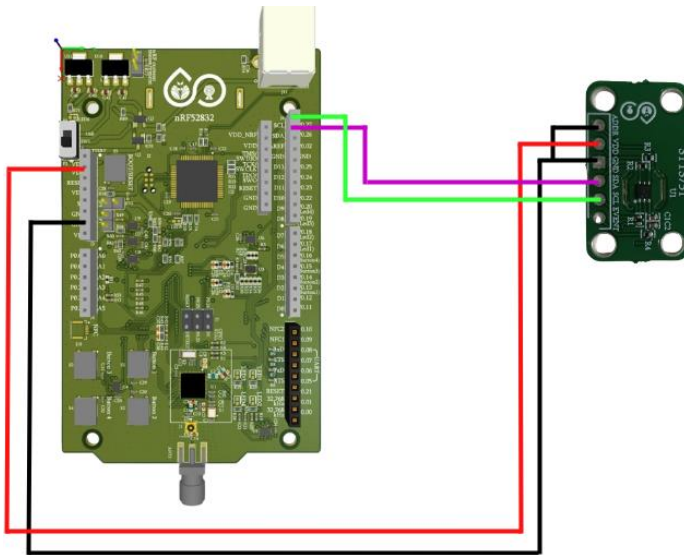
- To configure the i2c protocol, you have to enable it in the **.overlay** file.
- Click on the **Config files [1]** > click on **Kconfig** > click on **Devicetree [2]** > click on **nrf52dk\_nrf52832.overlay [3]**.
- The .overlay file will appear on your screen and add the given code to the .overlay file as shown in the picture given below [4].



- Click on **Build [1]** configuration again and check the **CONNECTED DEVICES [2]**.
- If device id is visible, then **Flash [3]** the code in Dev Kit.
- If **flashed successfully [4]** message is displayed on serial terminal, then flash process is complete.



## ❖ PIN CONFIGURATION



### Board Pins -> Sensor Pins

GND -> GND

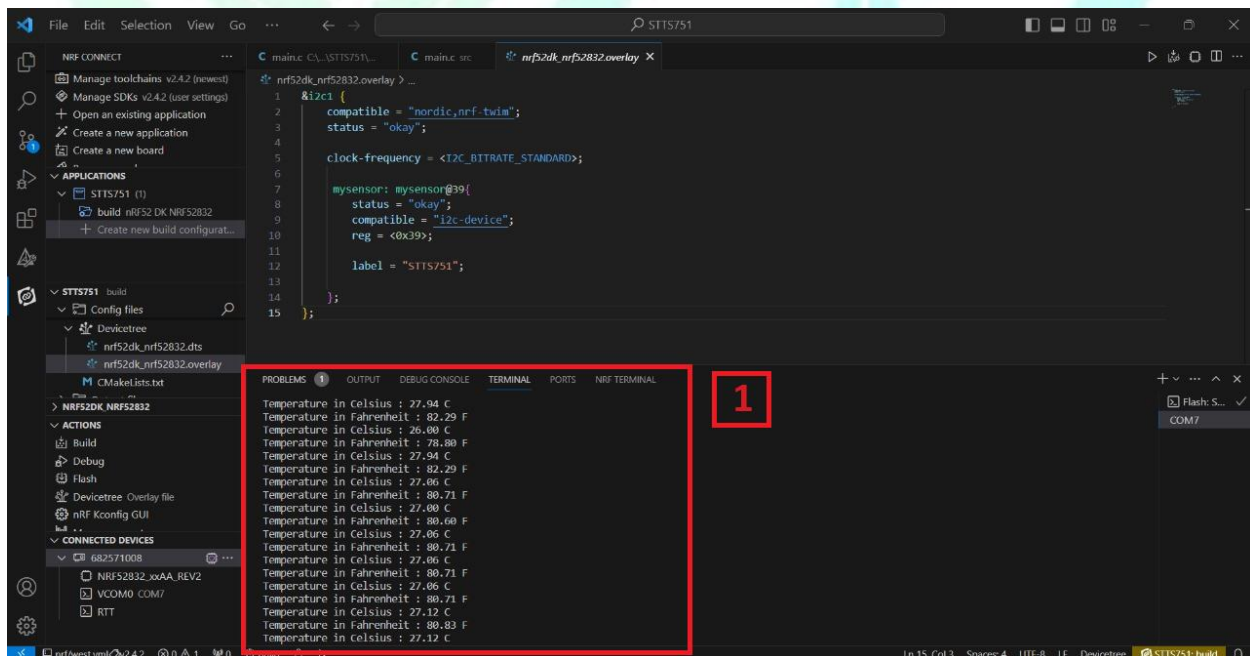
VDD -> VDD

PO.26 -> SDA

PO.27 -> SCL

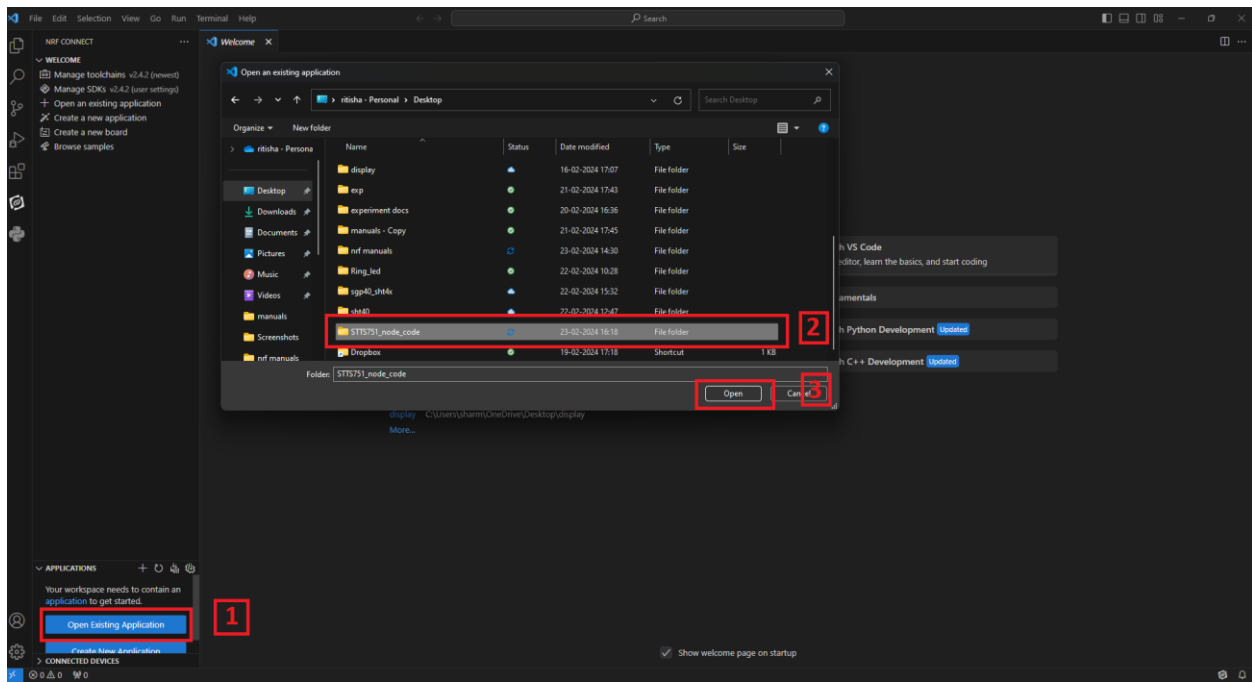
GND -> ADDR

## ❖ OUTPUT

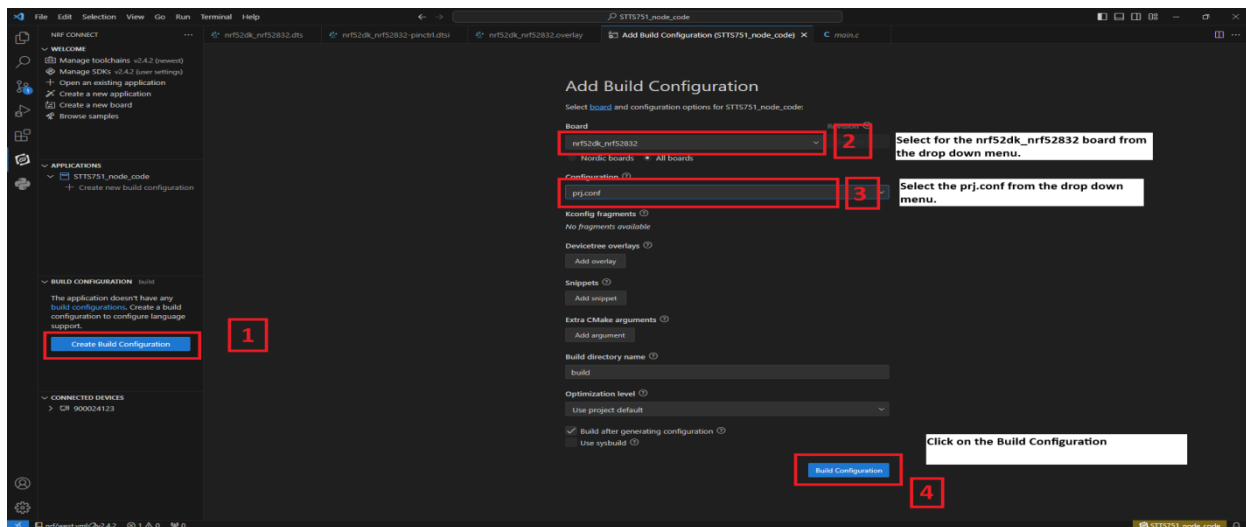


## INTERFACING WITH THE HELP OF NODE

- Open VS Code and click on **Open Existing Application** [1] > click on **STTS751\_node\_code** [2] > **Open** [3] as shown in the picture below.



- Click on **Create new build configuration** [1]. Here you can change the board version, if you are using nRF52832, then select **nrf52dk\_nrf52832** [2] or you can change from dropdown menu for another version like nRF52833 etc.
- After that click on the Configuration and select **prj.conf** [3] from dropdown menu and then click on the **Build Configuration** [4] as shown below in the picture.





- Go to source file, click **source file [1]** > click on **Application** > click on **src** > click on **main.c [2]**.
- After Click on **main.c** file and you will see the code on your screen.

```

src > C main.c
1 #include <zephyr/zephyr.h> // header files for the project
2 #include <zephyr/device.h>
3 #include <zephyr/devicetree.h>
4 #include <zephyr/drivers/i2c.h>
5 #include <zephyr/sys/printk.h>
6 #include <stdio.h>
7 #include <zephyr/zephyr.h>
8 #include <zephyr/drivers/gpio.h>
9 #include <zephyr/device.h>
10 #include <zephyr/pm/pm.h>
11 #include <hal/nrf_gpio.h>
12 #include <zephyr/drivers/sensor.h>
13
14 #define SLEEP_TIME_MS 1000 // Define the sleep time in milliseconds
15
16 #define STTS751_TEMP_HIGH_REG 0x00 // Define the register address for temperature high threshold in STTS751
17 #define STTS751_TEMP_LOW_REG 0x02 // Define the register address for temperature low threshold in STTS751
18 #define STTS751_CONFIG_REG 0x03 // Define the register address for configuration register in STTS751
19
20 /* Button variables */
21 static uint32_t time, last_time; //used for detecting long button press
22 static uint32_t button_counter=0; // Declare and initialize a static variable to store button counter value
23 int button_tick = 0; // Declare and initialize an integer variable to represent button tick
24
25 int sleep_flag=1; // Declare and initialize an integer variable to represent the sleep flag
26
27 #define I2C_NODE_DT_NODENAME(mysensor) // Define the I2C node label for the sensor "mysensor"
28
29 void enter_sleep(void) // Function to enter sleep mode
30 {
31
32
33     printk("Entering SYSTEM-OFF mode\n"); // Print a message indicating the entry into SYSTEM-OFF mode
34     NRF_POWER->SYSTEMOFF = 1; // Set the SYSTEMOFF bit in the NRF_POWER register to initiate system-off mode
35 }
36
37 void main(void) // main function

```

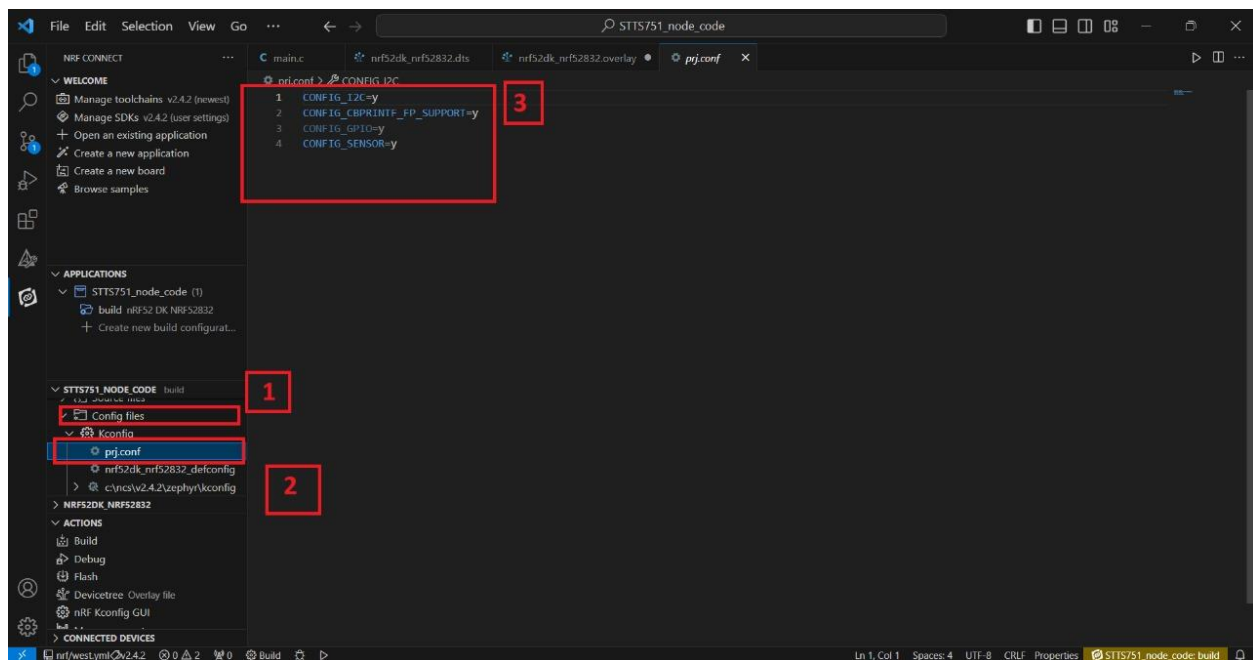
- To configure the i2c & UART protocols, you have to enable it in the **overlay file**.
- Click on the **Config files[1]** > click on **Kconfig** > click on **Devicetree** > click on **nrf52dk\_nrf52832.overlay [2]**.
- The overlay file will appear on your screen and add the given code to the **overlay file** as shown in the picture given below [3].

```

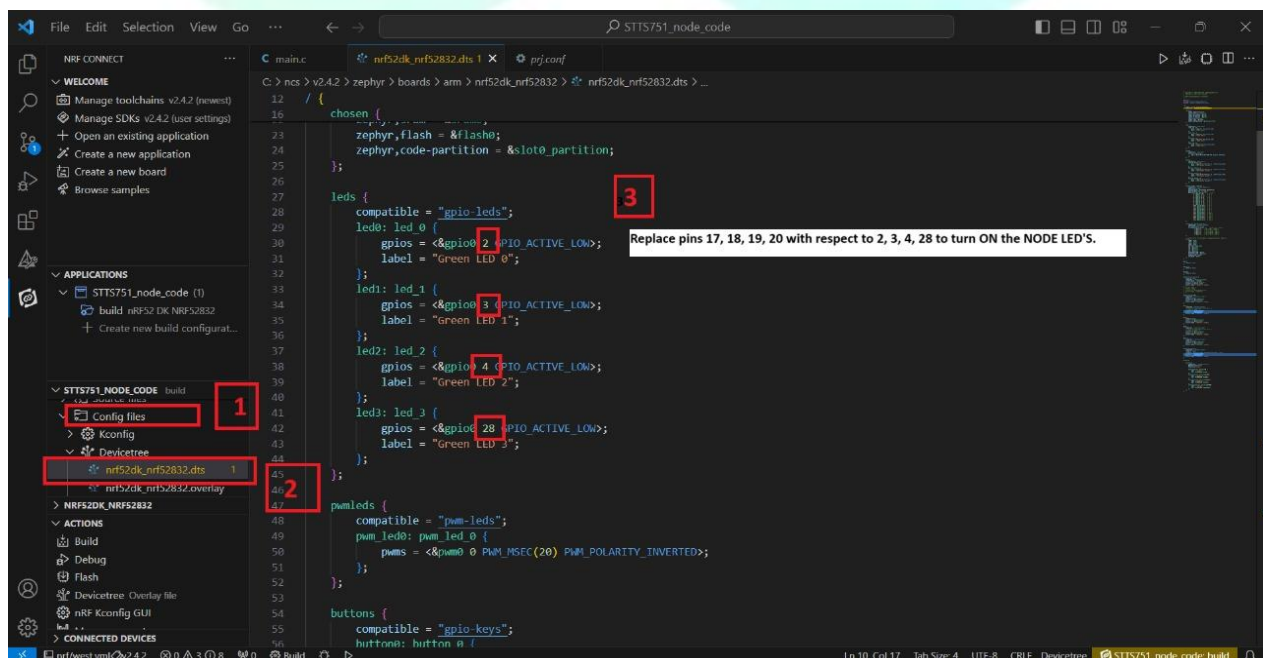
nrf52dk_nrf52832.overlay > pin-controller > uart0_default > group1
1 &pinctrl {
2     uart0_default: uart0_default {
3         group1 {
4             psels = <NRF_PSEL(UART_TX, 0, 24)>, //6
5             <NRF_PSEL(UART_RX, 0, 23)>, //8
6             <NRF_PSEL(UART_RTS, 0, 5)>,
7             <NRF_PSEL(UART_CTS, 0, 7)>;
8         };
9     };
10
11     uart0_sleep: uart0_sleep {
12         group1 {
13             psels = <NRF_PSEL(UART_TX, 0, 24)>, //6
14             <NRF_PSEL(UART_RX, 0, 23)>, //8
15             <NRF_PSEL(UART_RTS, 0, 5)>,
16             <NRF_PSEL(UART_CTS, 0, 7)>;
17             low-power-enable;
18         };
19     };
20
21     i2c1_default: i2c1_default {
22         group1 {
23             psels = <NRF_PSEL(TWIM_SDA, 0, 26)>, // 30
24             <NRF_PSEL(TWIM_SCL, 0, 25)>; // 31
25         };
26     };
27
28     i2c1_sleep: i2c1_sleep {
29         group1 {
30             psels = <NRF_PSEL(TWIM_SDA, 0, 26)>, //30
31             <NRF_PSEL(TWIM_SCL, 0, 25)>; //31
32             low-power-enable;
33         };
34     };
35
36     &i2c1 {
37         compatible = "nordic,nrf-twi";
38         status = "okay";
39     };
40 }

```

- You need to enable sensor in prj file for communication as shown below.
- Click **Config files [1]** > then click on **Kconfig files** > click on **prj.conf [2]**.
- The **prj.conf** will appear on the screen **[3]** as shown below in the picture.

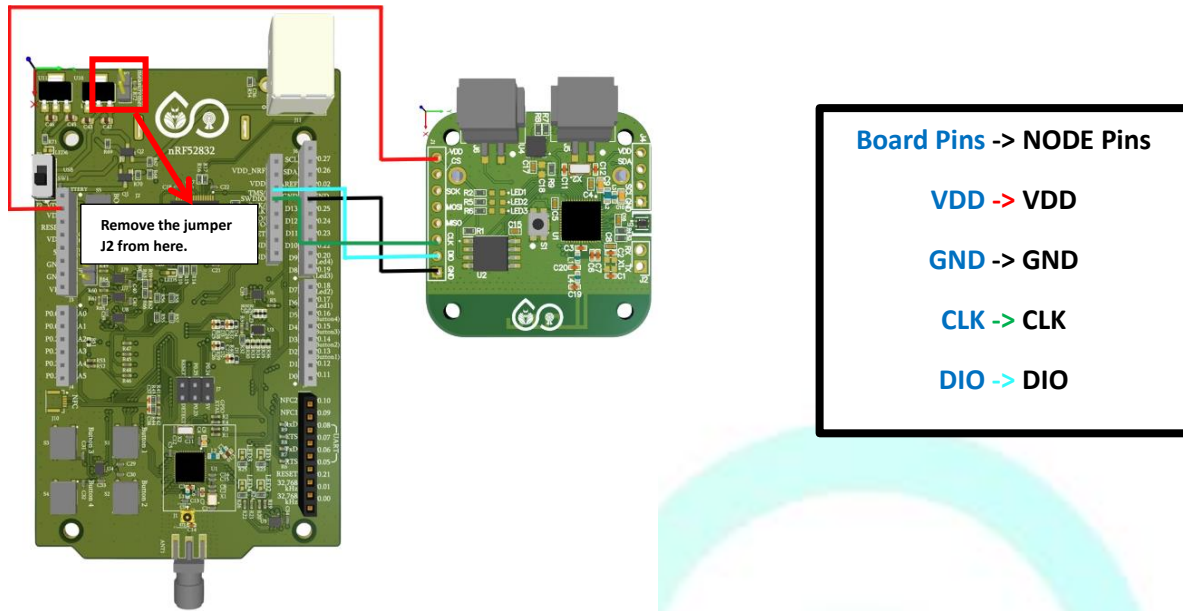


- You need to enable sensor in prj file for communication as shown below.
- Click **Config files [1]** > then click on **Devicetree** > click on **nrf52dk\_nrf52832.dts [2]**
- The **dts file** will appear on your screen and add the details in your **dts file** as shown in the picture given below [3].

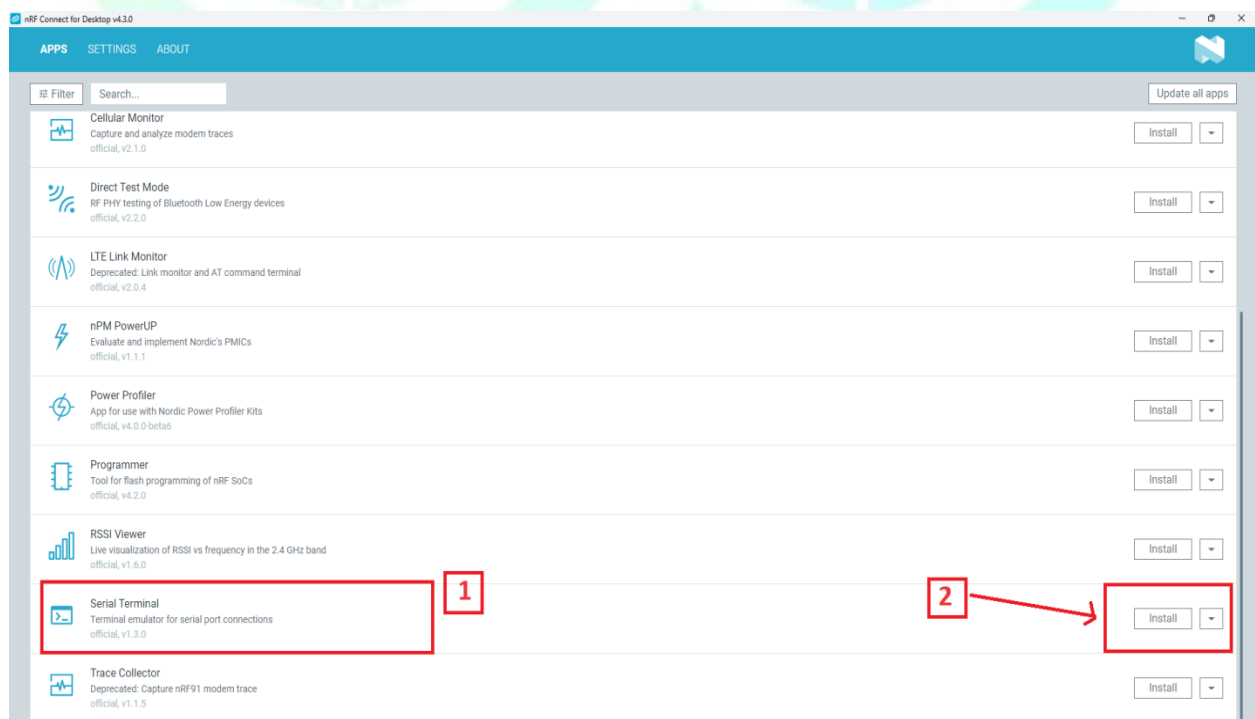




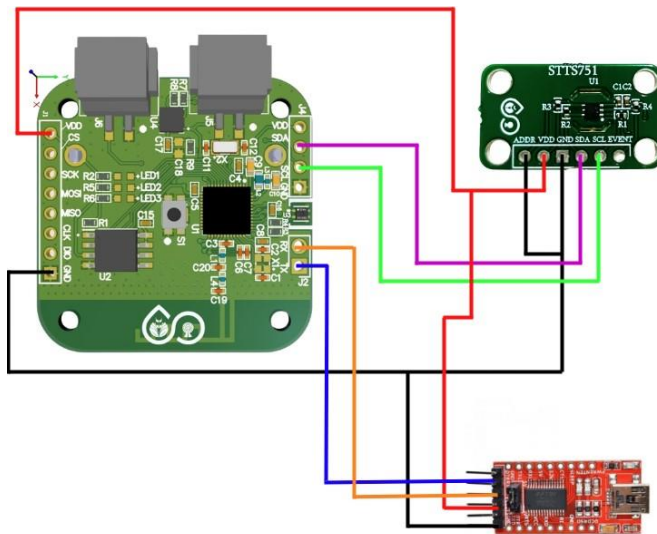
- For Node programming remove the jumper **J2** from the development board.
- Now flash the code with the help of nRF52832 development board as shown below in the figure.



- Firstly, you have to **Install [2]** the nRF **Serial Terminal [1]** in nRF Connect for Desktop application as shown below.



- Connect the **TTL Device** for UART communication so that the data must appear on the serial terminal.
- Connect the **TTL Device** as shown below in the picture.



**Node Pins -> TTL Pins**

**Tx -> Rx**

**Rx -> Tx**

**VDD -> VDD**

**GND -> GND**

**Node Pins -> Sensor Pins**

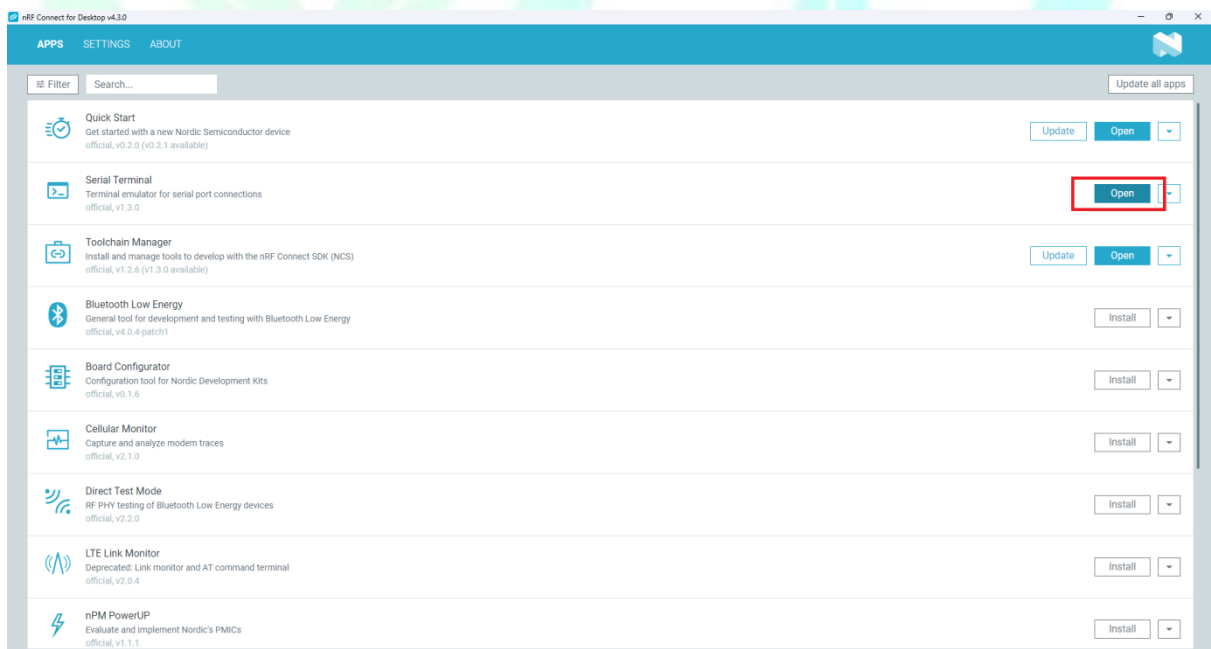
**SDA -> SDA**

**SCL -> SCL**

**VDD -> VDD**

**GND -> GND**

- After this, click on **Open** as shown below in the picture.



- Click on **Select Device [1]** > click on **FT232R USB UART [2]** as shown below in the picture.

