Design a Datalogger code to send light sensor data from STM32 to NanoEdge AI Studio on PC

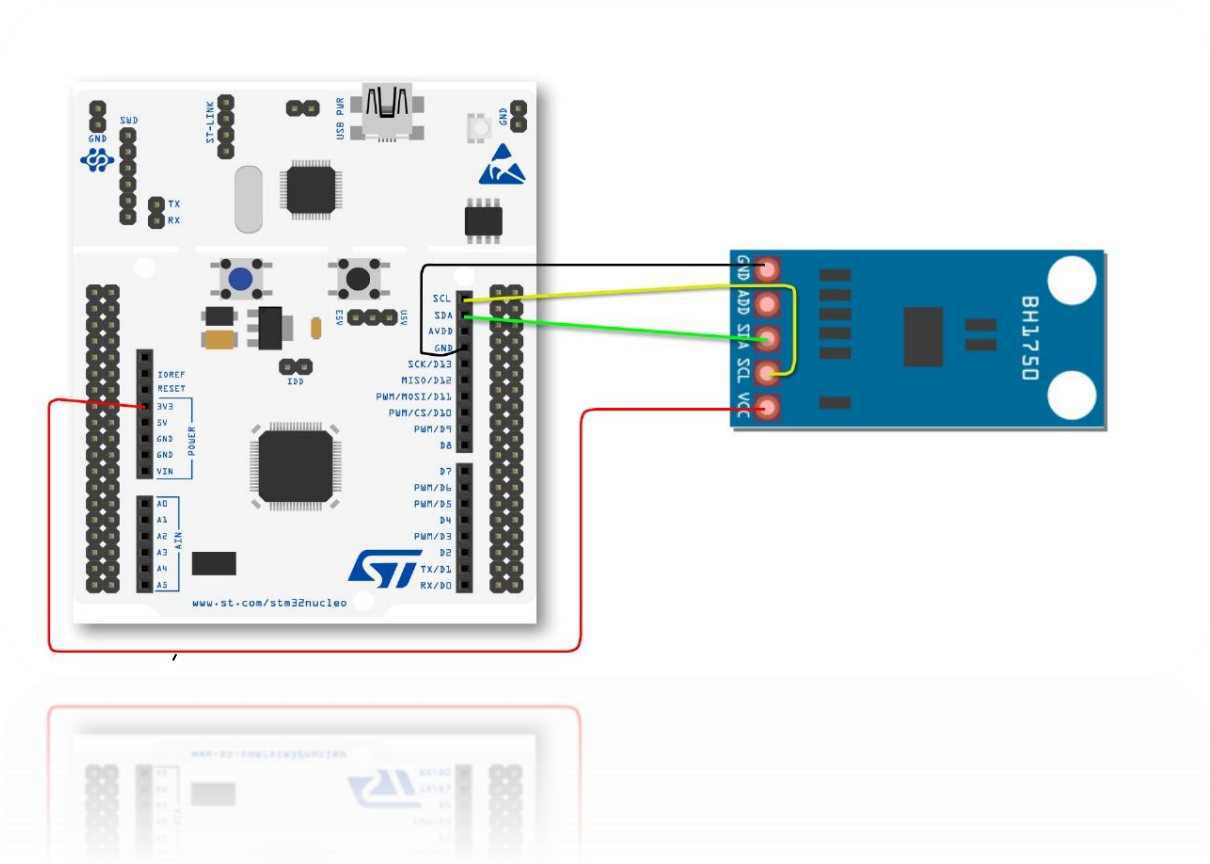
Objective:

The Objective of this experiment is to interface a light sensor to an STM32 microcontroller and deploy the Machine Learning Model built using the NanoEdge AI Studio into the microcontroller. This will give the microcontroller the ability to make a decision on the device itself based on classification on real time light sensor data.

Requirements:

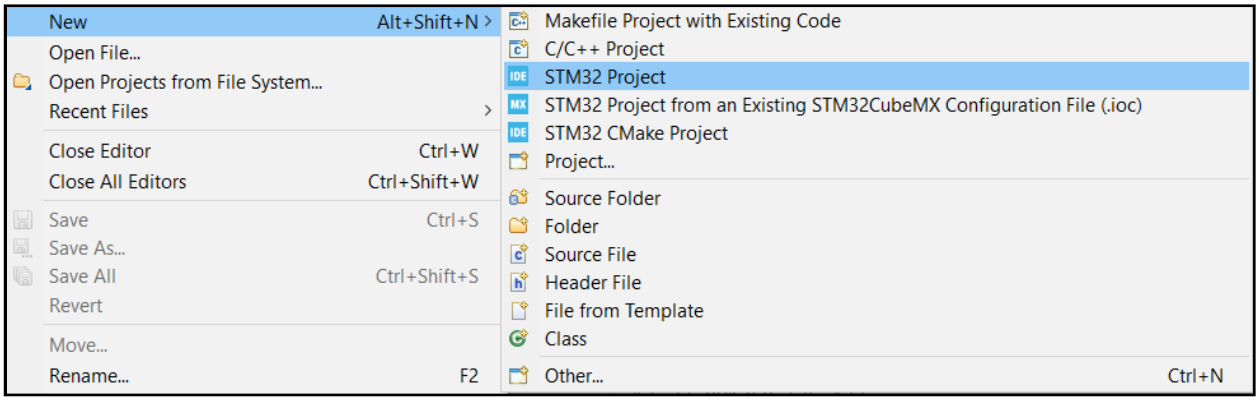
1. STM32 Cube IDE software.
2. Light Sensor (I2C).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.

Connection Diagram:

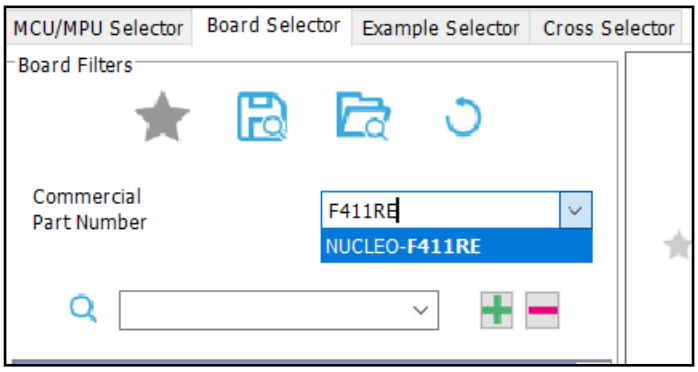


Procedure:

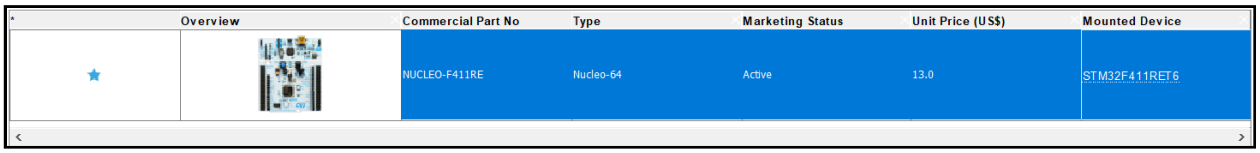
1. Click on **File→New→STM32 Project** to start your project on Cube IDE.



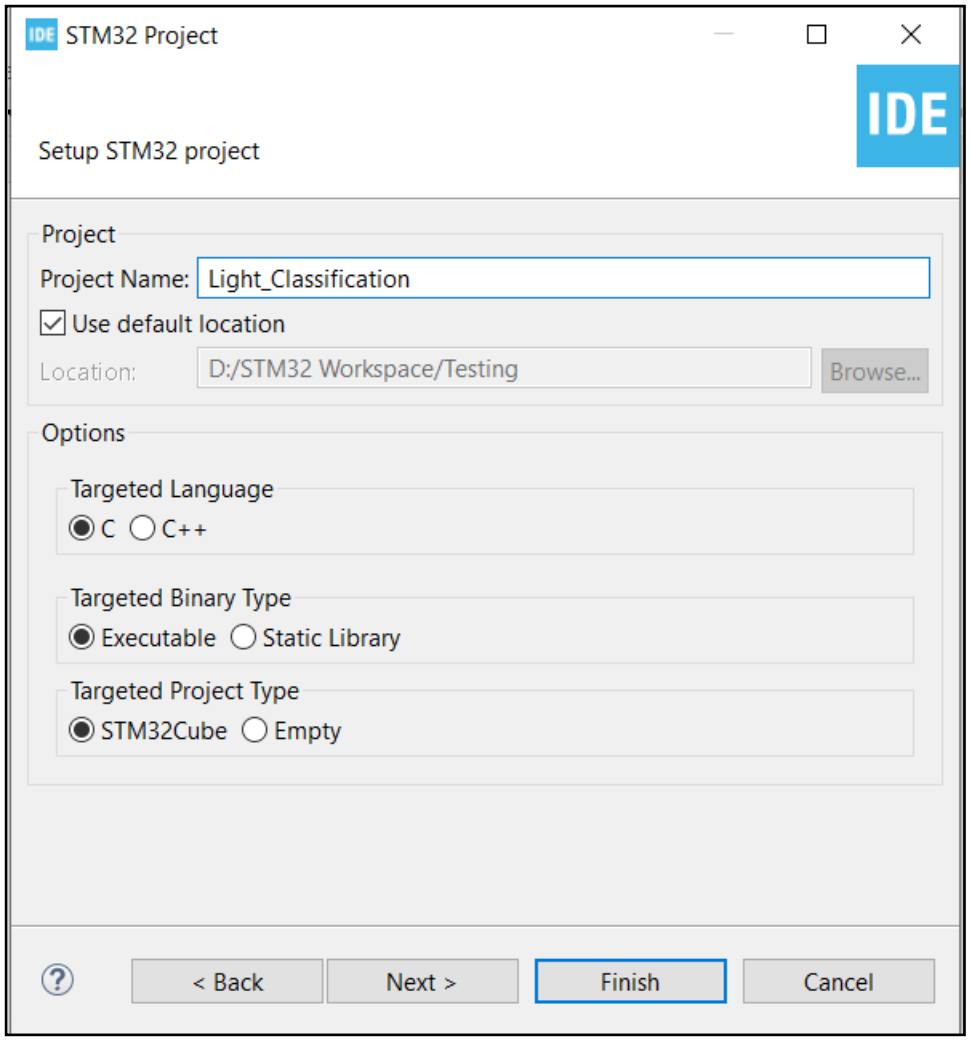
2. A **Target Selection** window will open. Click on **Board Selector**, where you need to select the microcontroller board you are working with.
(**NB:** If you are having Nucleo-F401RE, you have to select the said Commercial Part Number)



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.**

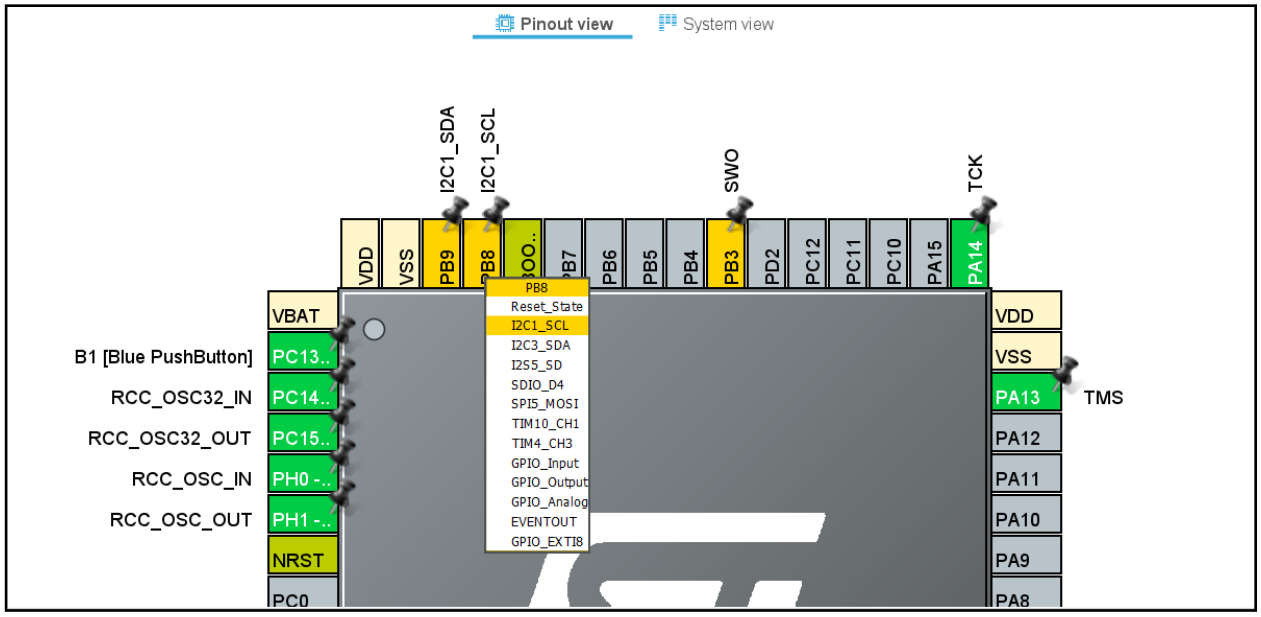


4. In the next window give your project a name, rest of the things will remain by default as it is for now. Click on **Finish.**

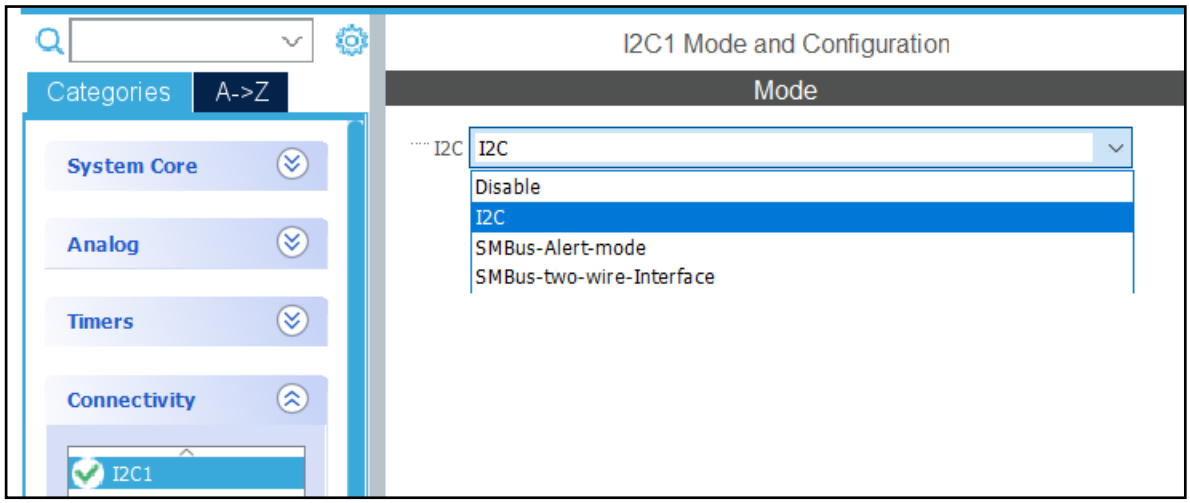


5. Cube IDE will ask if you want to initialize all peripherals with their default mode, click on **Yes.**

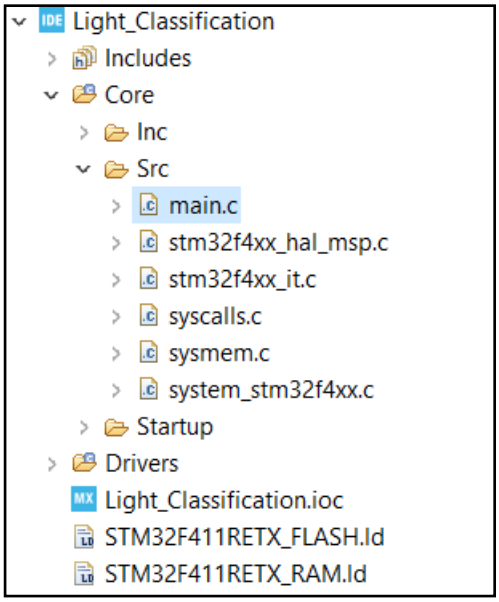
6. In the **Pinout & Configuration** tab, click on **PB8** pin and select it as an **I2C1_SCL** and **PB9** pin as an **I2C1_SDA**.



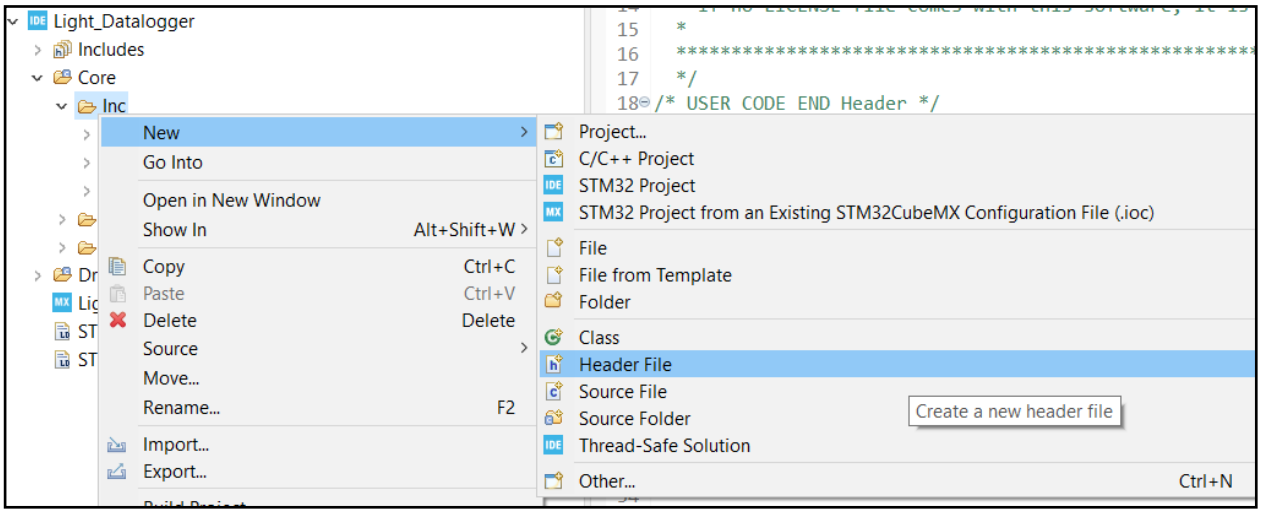
7. Next on the left-hand side under **Categories →Connectivity**, select I2C1 and enable it.



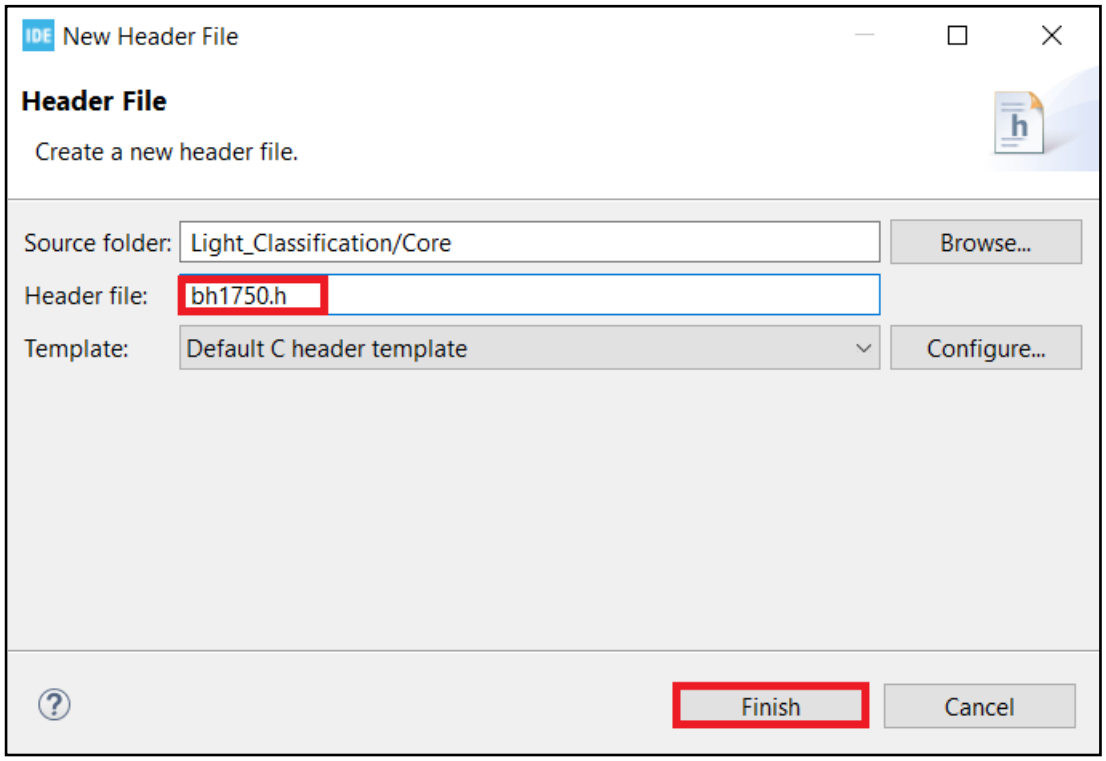
8. Press C**trl+S** to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example, I have named my project as (Light_Classification) **Light_Classification→Core →Src→main.c (**double click to load the code).



9. Now open your project tree **Light_Classification→Core →Inc.** Right click on your **Inc** folder and create a new **Header File**.



10. Name the Header File as **bh1750.h** and select on **Finish**.



11. Below is the code snippets, please put your code in the appropriate places in the **bh1750.h** file.

```c
#ifndef INC_BH1750_H_
#define INC_BH1750_H_

#include "stdio.h"

// BH1750 I2C Address
#define BH1750_ADDR 0x23 // BH1750 I2C address


// Function prototypes
void BH1750_Init(I2C_HandleTypeDef *hi2c);
float BH1750_ReadLux(I2C_HandleTypeDef *hi2c);

// Error Status
HAL_StatusTypeDef Transmit_Err, Receive_Err;


// BH1750 initialization
void BH1750_Init(I2C_HandleTypeDef *hi2c) {
    uint8_t cmd[] = {0x10}; // Power on
    Transmit_Err = HAL_I2C_Master_Transmit(hi2c, BH1750_ADDR << 1, cmd, sizeof(cmd), HAL_MAX_DELAY);
    if(Transmit_Err != HAL_ERROR){
          printf("\r\n");
          printf("BH1750 has been initialized");
          printf("\r\n");
    }
}

// Reading Light Intensity from BH1750 sensor
float BH1750_ReadLux(I2C_HandleTypeDef *hi2c) {
    uint8_t data[2];
    HAL_I2C_Master_Receive(hi2c, BH1750_ADDR << 1, data, sizeof(data), HAL_MAX_DELAY);

    uint16_t lux = (data[0] << 8) | data[1];
    return (float)lux / 1.2;
}
#endif
```

12. 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.

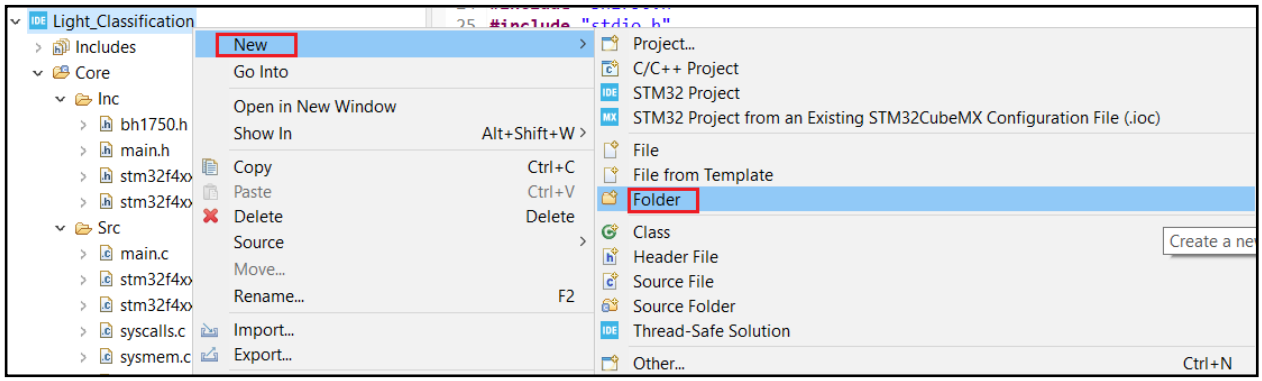
```c
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "bh1750.h"
#include "stdio.h"
#include "string.h"
#include "knowledge.h"
#include "NanoEdgeAI.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define DATA_INPUT_USER 256
#define AXIS_NUMBER 1
#define CONFIRMATIONS_NB (uint32_t)(3)
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
float light;
float light_buffer[DATA_INPUT_USER * AXIS_NUMBER] = {0};
float output_class_buffer[CLASS_NUMBER]; // Buffer of class probabilities
const char *id2class[CLASS_NUMBER + 1] = { // Buffer for mapping class id to class name
    "unknown",
    "Flashlight_Working",
    "Flashlight_Not_Working",
    "No_Flashlight",
};
```

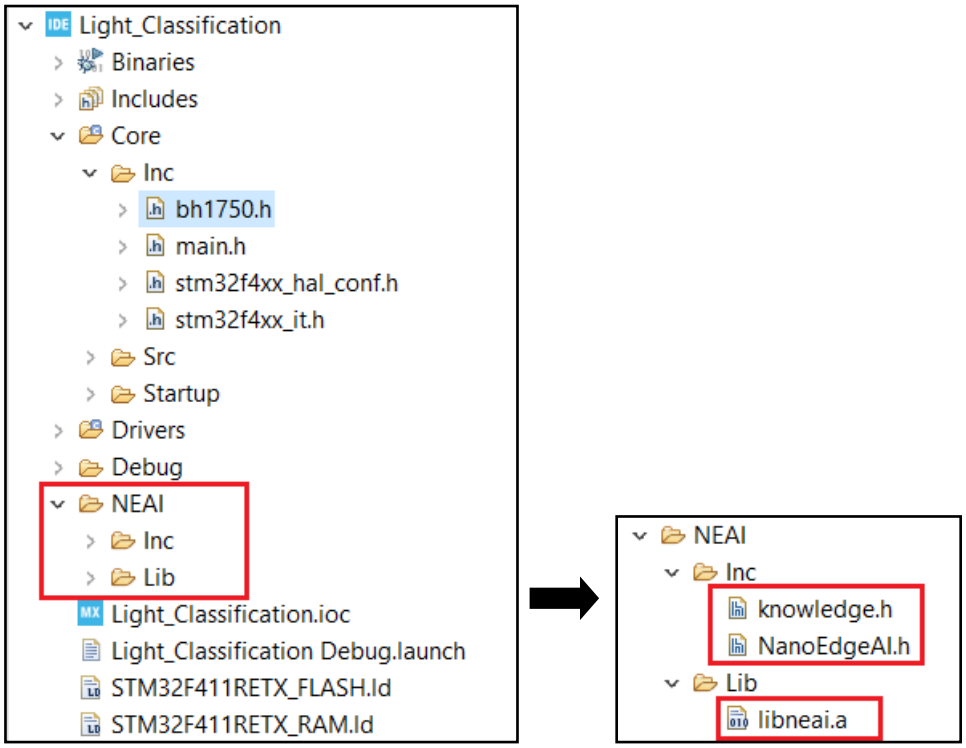
```c
/* USER CODE BEGIN PFP */
void fill_light_buffer();
void inference();

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  enum neai_state error_code = neai_classification_init(knowledge);
      if (error_code!=NEAI_OK)
      {
        printf("Knowledge initialization ERROR:");
        printf("%d", error_code);
      }
      else
      {
        printf("Knowledge initialization DONE:");
      }
```

```
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_I2C1_Init();
  /* USER CODE BEGIN 2 */

  BH1750_Init(&hi2c1);


  /* USER CODE END 2 */

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

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
```

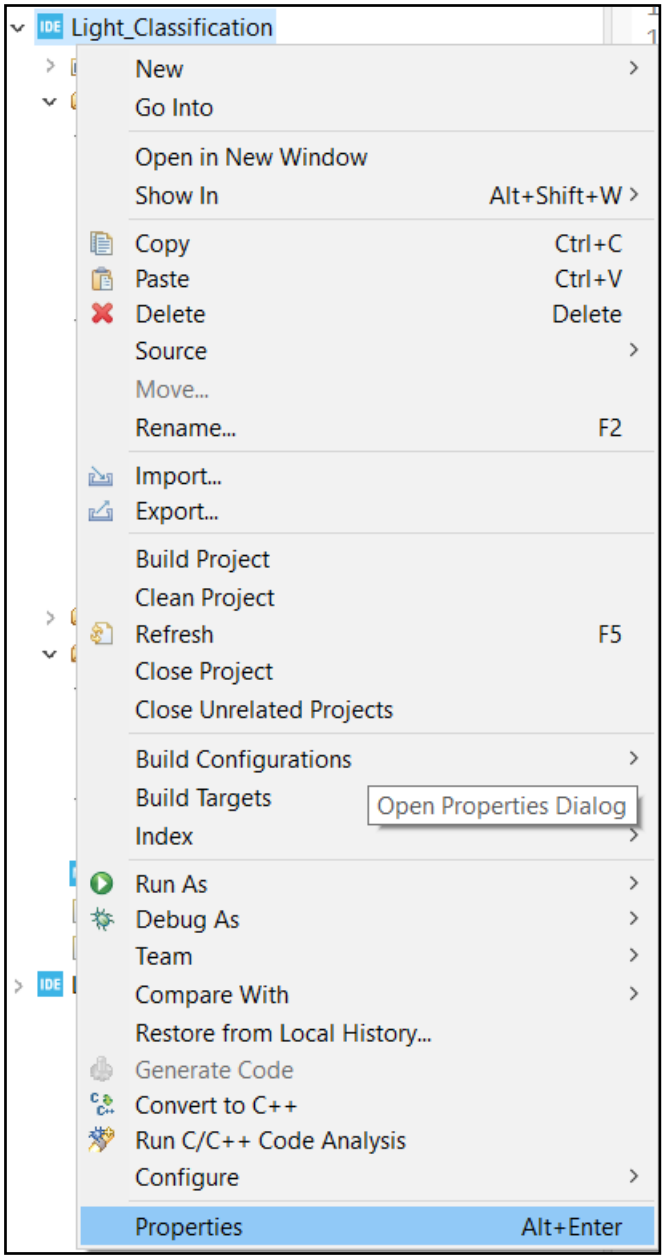
```c
/* USER CODE BEGIN 4 */
void fill_light_buffer() {
    for(int i = 0; i < DATA_INPUT_USER; i++){
        light = BH1750_ReadLux(&hi2c1);
        light_buffer[AXIS_NUMBER * i] = light;
        HAL_Delay(3);
    }
}

void inference()
{

  uint16_t i, id_class_t0, id_class_tn;
  fill_light_buffer();

  neai_classification(light_buffer, output_class_buffer, &id_class_t0);
  for (i = 0; i < CONFIRMATIONS_NB - 1; i++)
  {


    fill_light_buffer();
    neai_classification(light_buffer,output_class_buffer,&id_class_tn);
    if (id_class_t0 != id_class_tn)
    {
      break;
    }
    if (id_class_t0 == id_class_tn)
    {
        printf("Detected Class is:");
        printf(id2class[id_class_t0]);
        printf("\r\n");
    }
    else
    {
         printf("?");
         printf("\r\n");
    }
  }
}


int __io_putchar(int ch){
    HAL_UART_Transmit(&huart2, (uint8_t *) &ch, 1, HAL_MAX_DELAY);
    return ch;
}
```

13. Right click on the Light_Classification project and select **New→Folder**. Name the new folder as NEAI and click on **Finish**.You will be able to see a folder named NEAI inside of your Light_Classification project. Now right click on this newly created NEAI folder and create two separate new folders and name them Inc and Lib.

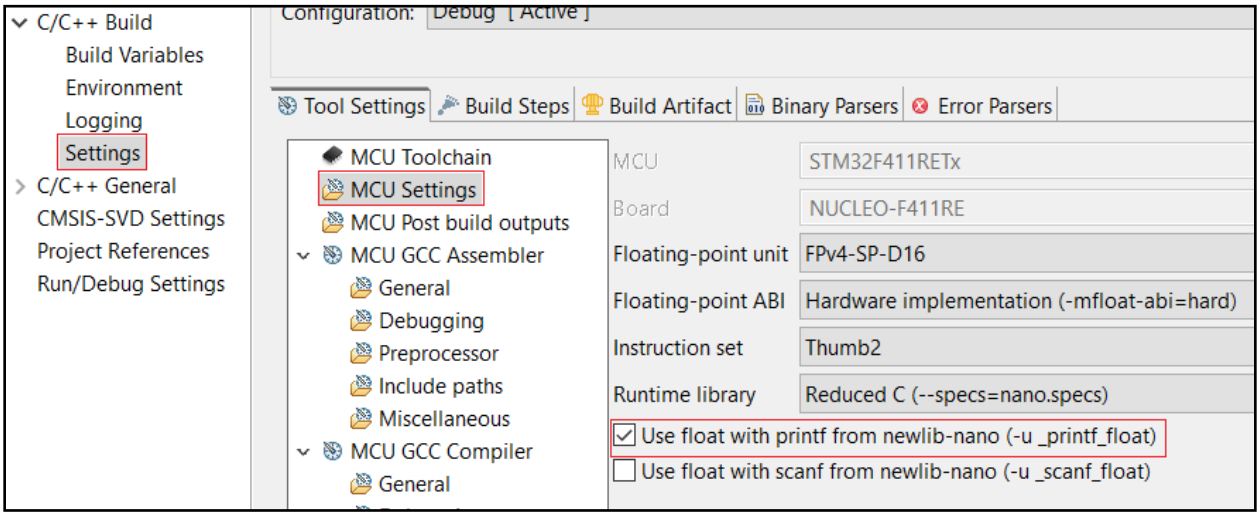




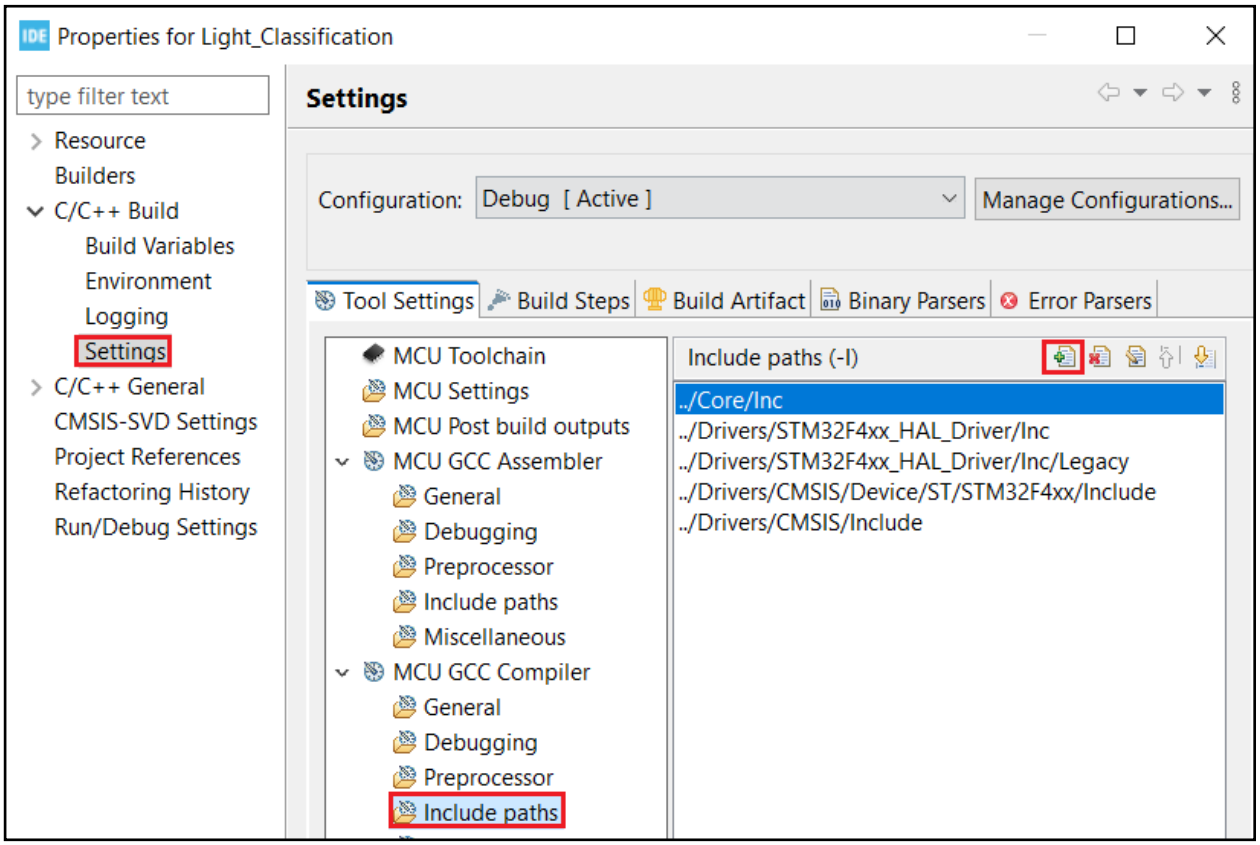
14. Now right click on your project tree, go to **Properties**

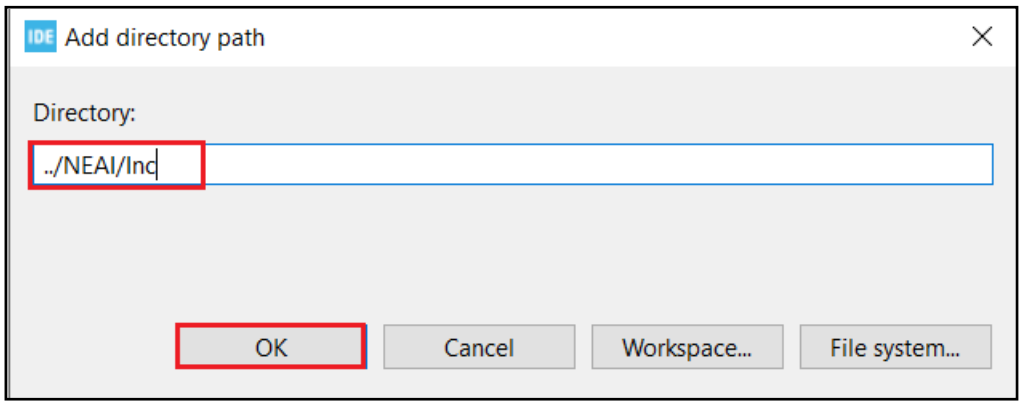


15. After that select **Settings→MCU Settings** and enable the **float printf.**

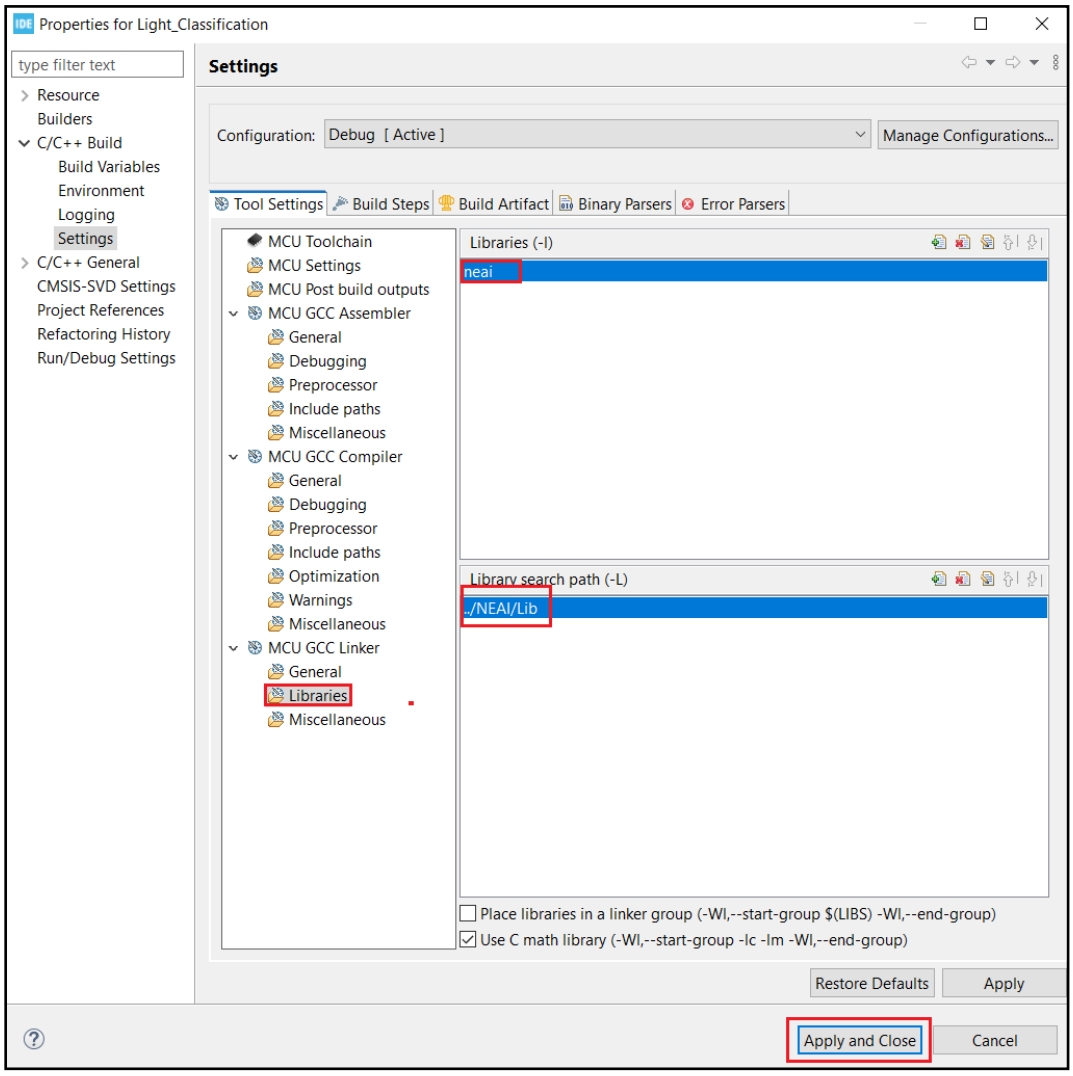


16. Next select **MCU GCC Compiler →Include paths** click on **Add** and in the new window select **Workspace** and select the **Inc** folder and click on **Ok.** The path of the **Inc** folder will be now added.

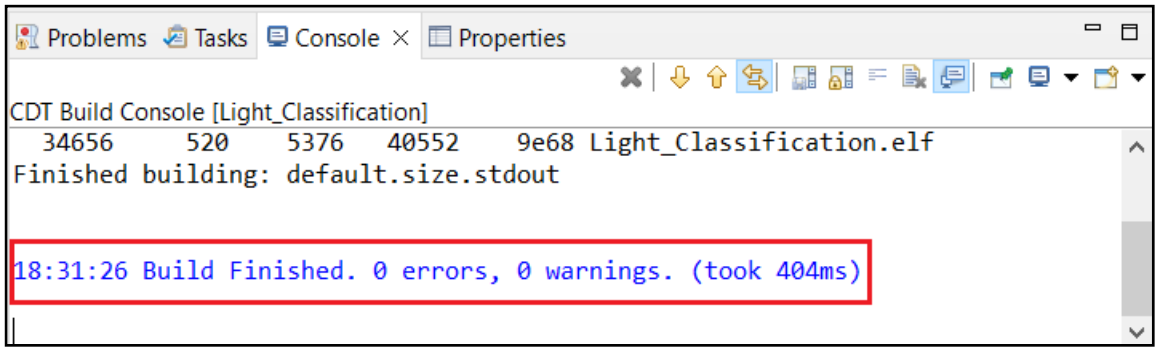




17. Next go to **MCU GCC Linker →Libraies**. In **Library search path (-L)** section, add the path of **Lib** folder similarly to the previous step of adding the **Inc** folder path. Next in the **Libraries(-l)** section click on **Add** and type **neai** then click on **Ok.** Then **Apply and Close.**

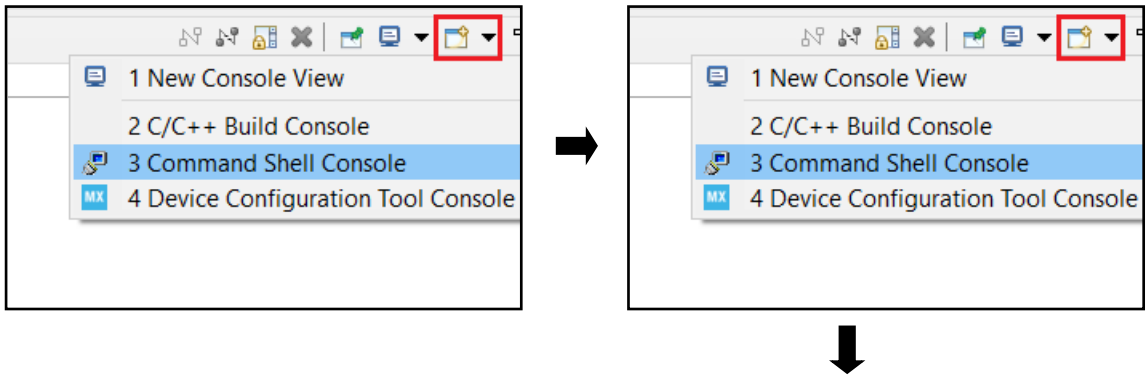


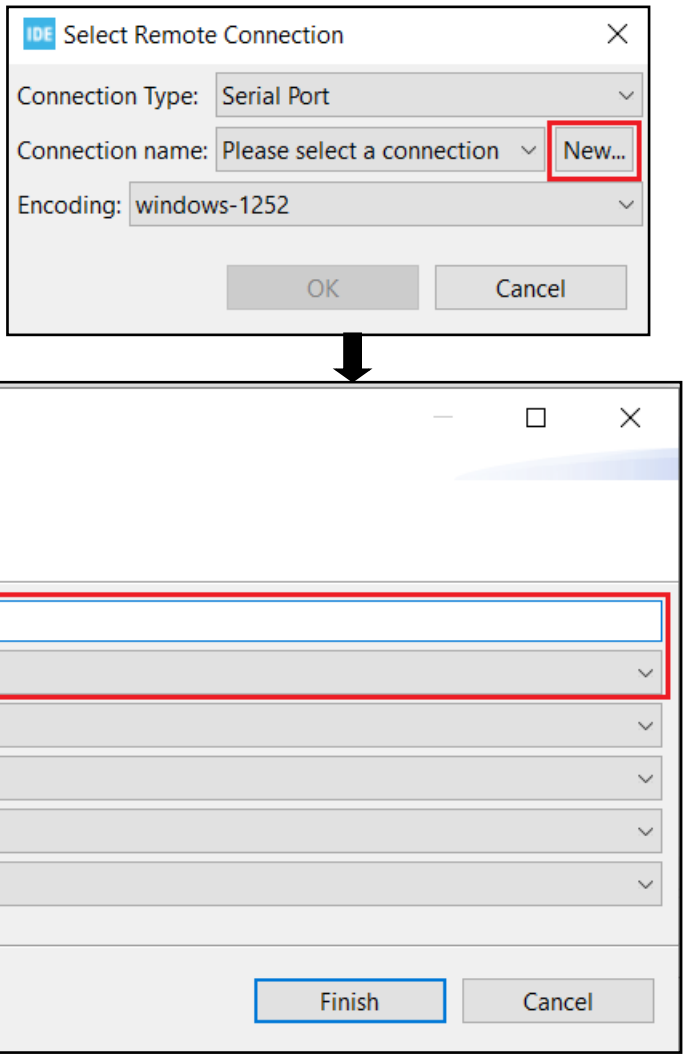
18. Now click on the **build** symbol  on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.



19. 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.

20. 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.




Select Remote Connection
Connection Type:
Serial Port
Connection name:
Please select a connection
New...
Encoding:
windows-1252
OK
Cancel
New Serial Port Connection
New serial port connection settings
Connection name:
Serial_Monitor
Serial port:
COM8
Baud rate:
115200
Data size:
8
Parity:
None
Stop bits:
1
Finish
Cancel

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

```
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
Detected Class is:No_Flashlight
DetDetected Class is:Flashlight_Working
Detected Class is:Flashlight_Working
Detected Class is:Flashlight_Working
Detected Class is:Flashlight_Working
Detected Class is:Flashlight_Working
Detected Class is:Flashlight_Working
Detected Class is:No_Flashlight
Detected Class is:Flashlight_Not_Working
Detected Class is:Flashlight_Not_Working
```

\

22. 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 color box for the proper understanding of the user. This document is for the use of education purpose only.