Anomaly Detection: Design of an ML based Anomaly Dete

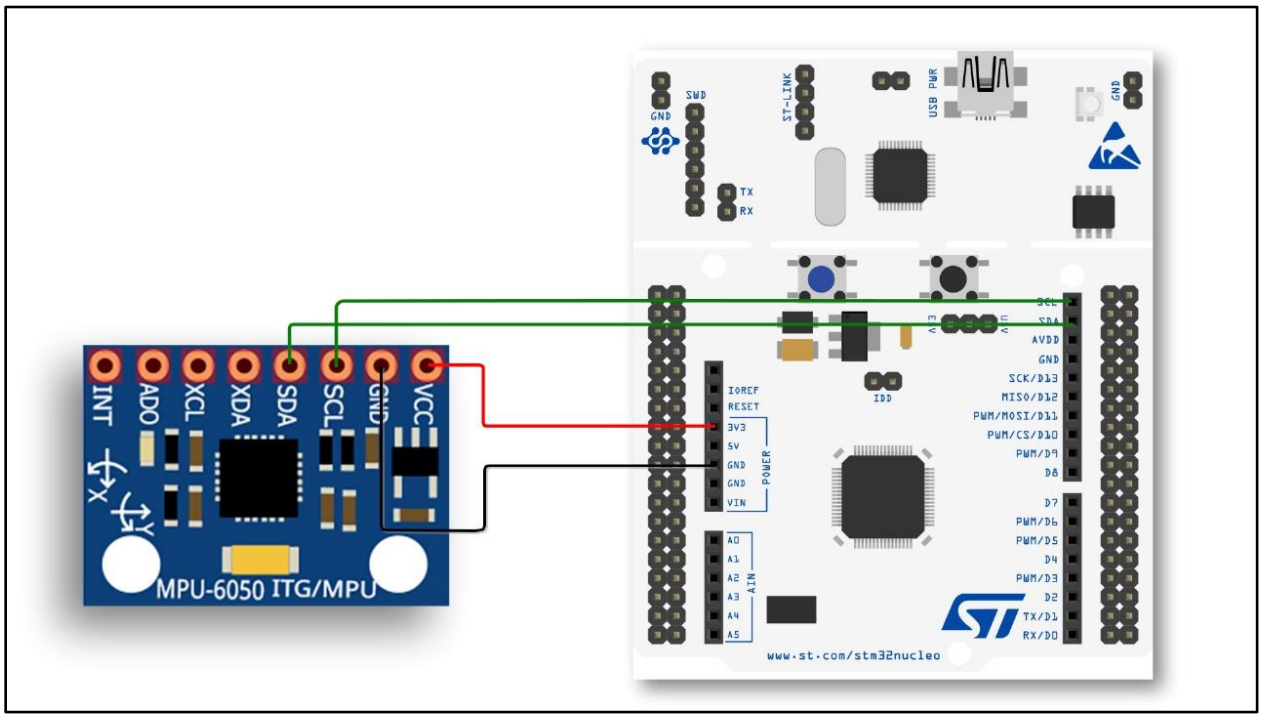
Objective:

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:

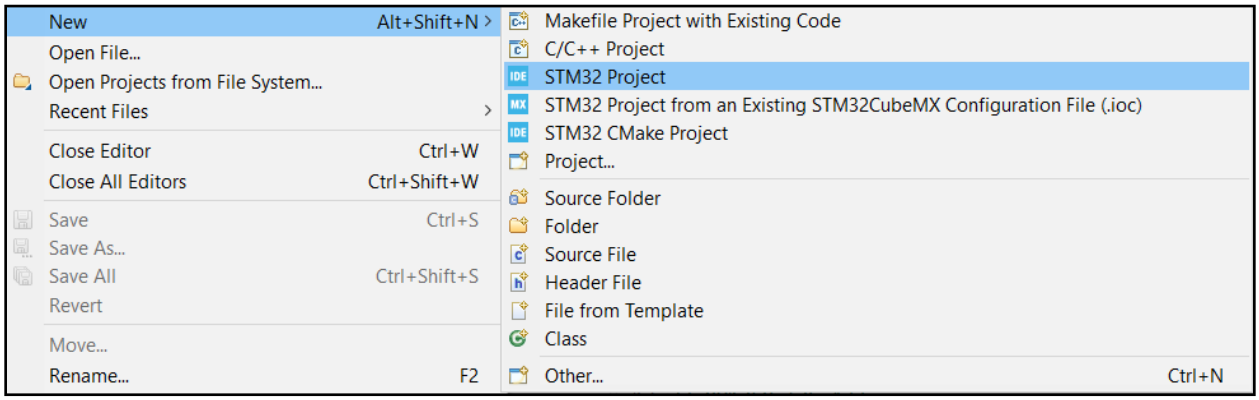
1. STM32 Cube IDE software.
2. Light Sensor (I2C).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.

Connection Diagram:

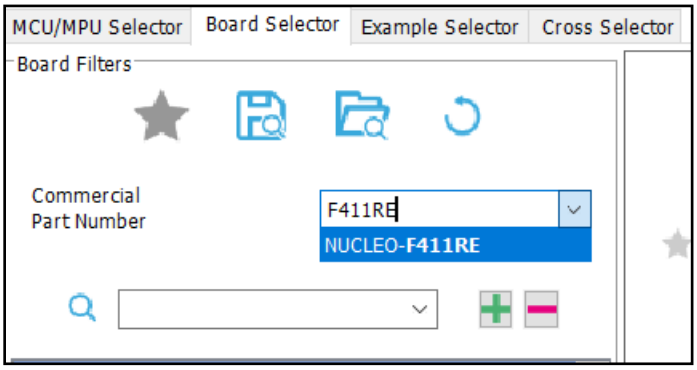


Procedure:

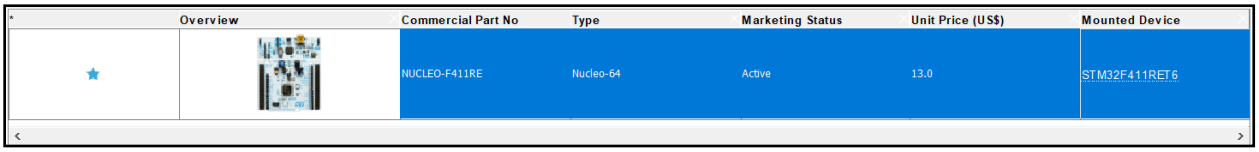
1. Click on **File→New→STM32 Project** to start your project on Cube IDE.



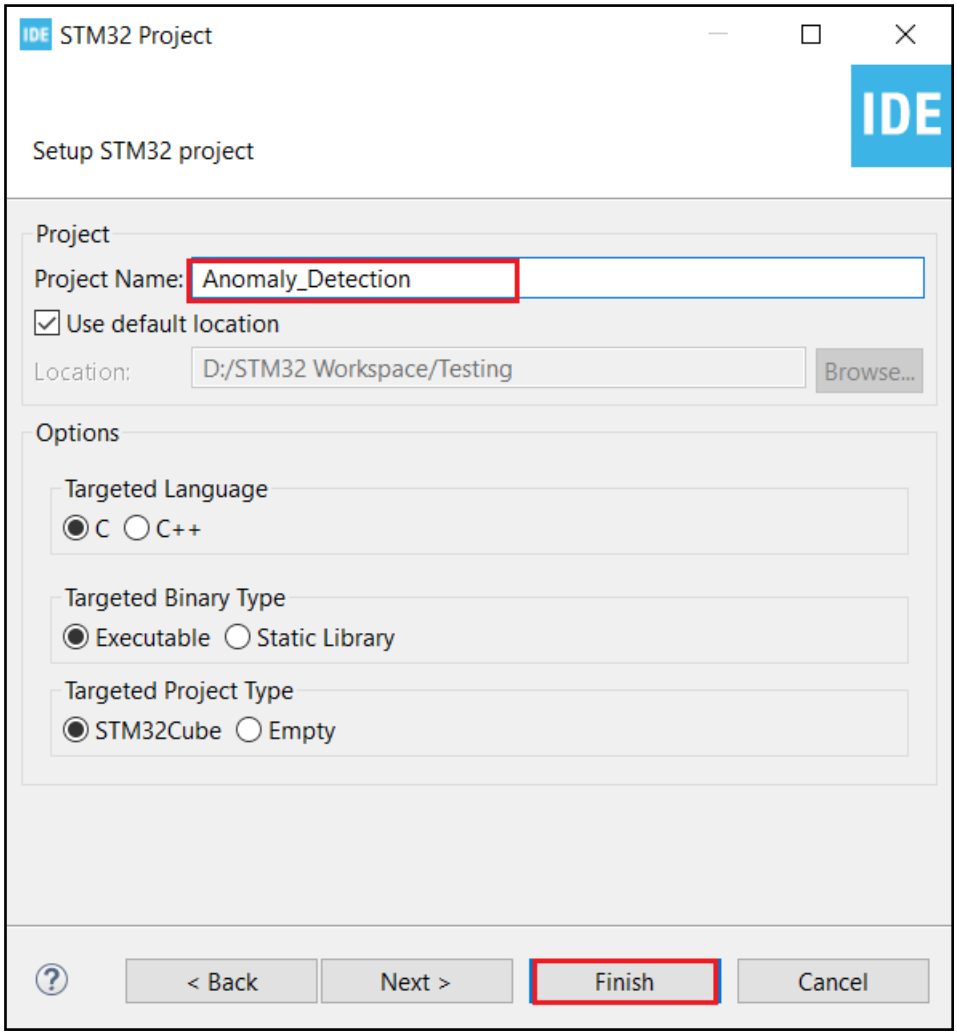
2. A **Target Selection** window will open. Click on **Board Selector**, where you need to select the microcontroller board you are working with.
(**NB:** If you are having Nucleo-F401RE, you have to select the said Commercial Part Number)



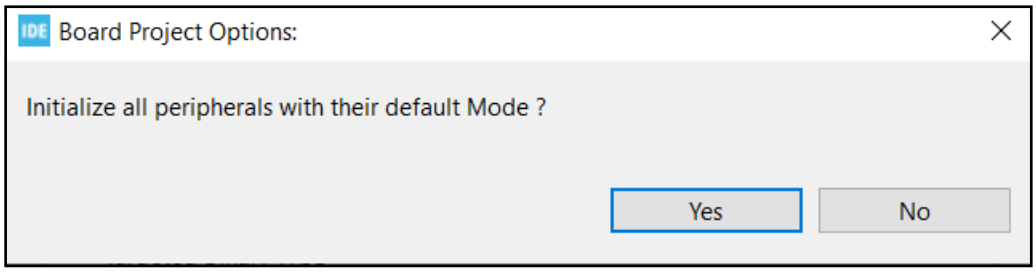
3. After this on the right-hand side of the window, under **Board List** you will see the board you have selected. Click on the board and then click on **Next.**



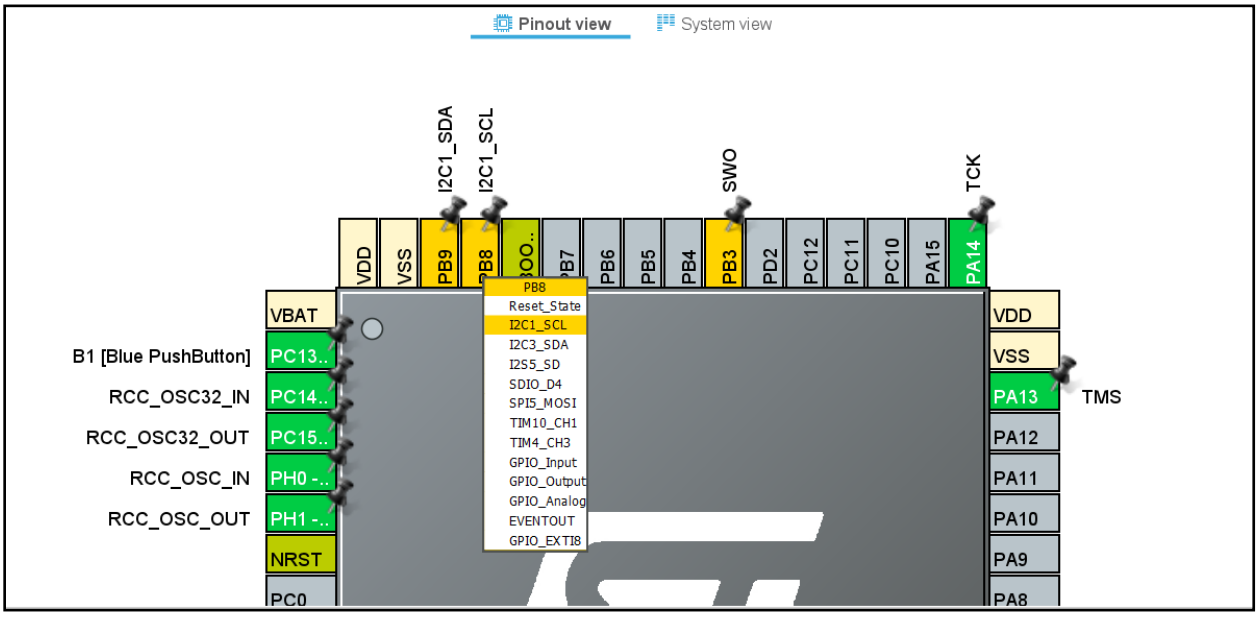
4. In the next window give your project a name **"Anomaly_Detection"**, rest of the things will remain by default as it is for now. Click on **Finish.**



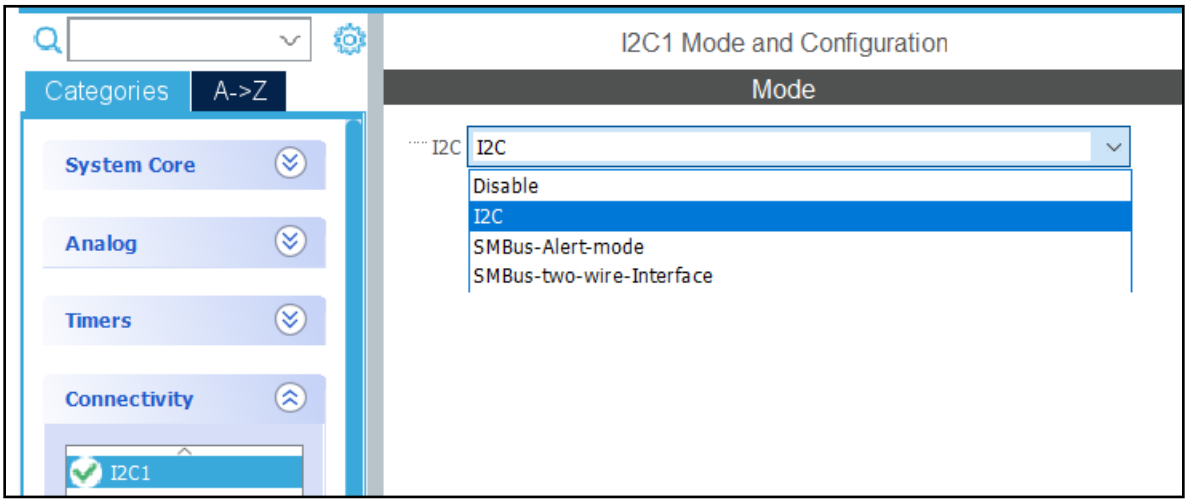
5. Cube IDE will ask if you want to initialize all peripherals with their default mode, click on **Yes.**



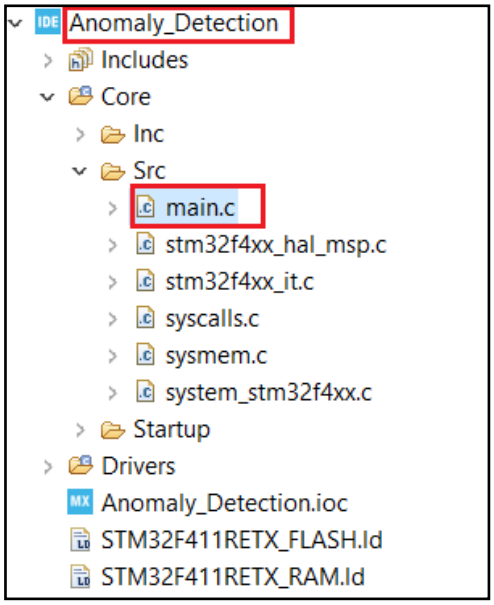
6. In the **Pinout & Configuration** tab, click on **PB8** pin and select it as an **I2C1_SCL** and **PB9** pin as an **I2C1_SDA**.



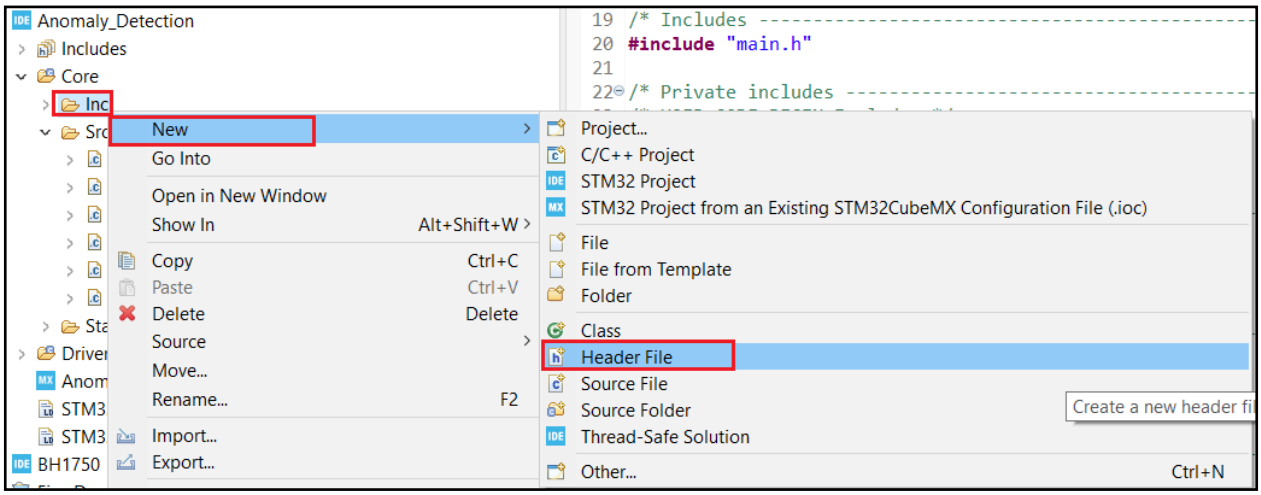
7. Next on the left-hand side under **Categories →Connectivity**, select I2C1 and enable it.



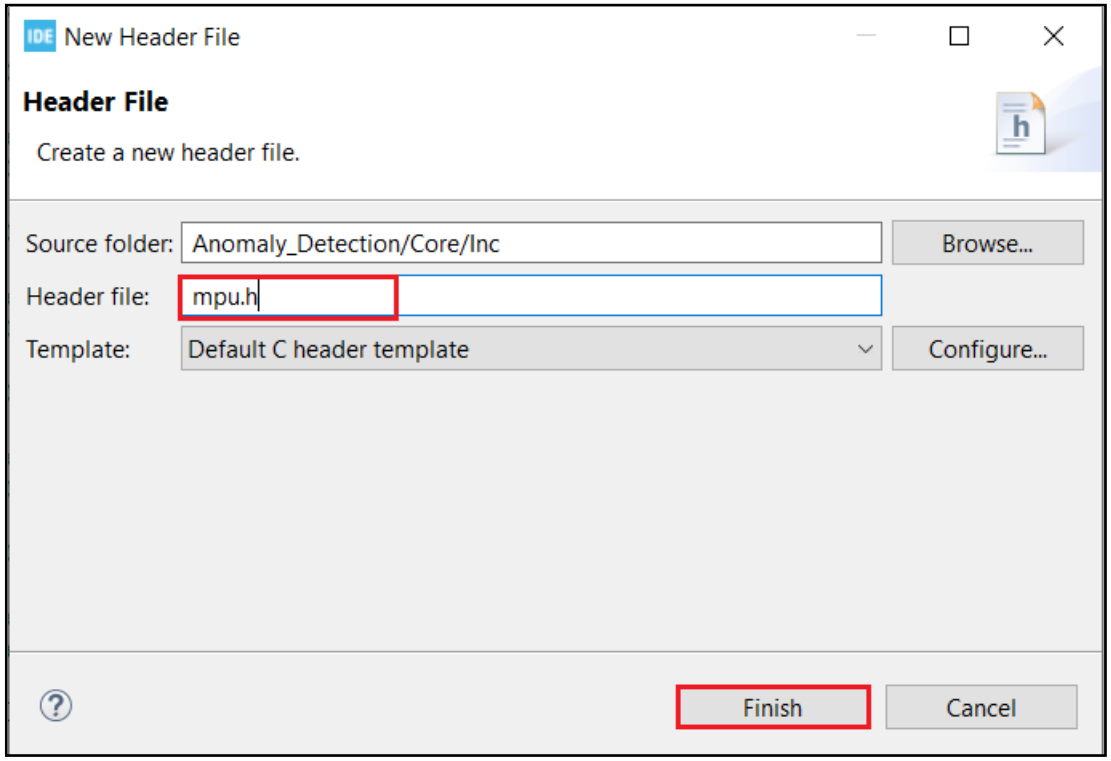
8. Press C**trl+S** to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example, I have named my project as (Anomaly_Detection) **Anomaly_Detection→Core →Src→main.c (**double click to load the code).



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



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



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

```
#ifndef INC_MPU_H_
#define INC_MPU_H_

extern I2C_HandleTypeDef hi2c1;

#define MPU6050_ADDR            0xD0
#define MPU6050_PWR_MGMT_1      0x6B
#define MPU6050_ACCEL_XOUT_H    0x3B
#define MPU6050_ACCEL_YOUT_H    0x3D
#define MPU6050_ACCEL_ZOUT_H    0x3F
#define MPU6050_TEMP_OUT_H      0x41
#define MPU6050_GYRO_XOUT_H     0x43
#define MPU6050_GYRO_YOUT_H     0x45
#define MPU6050_GYRO_ZOUT_H     0x47

int16_t accel_data[3];
int16_t gyro_data[3];

float Ax, Ay, Az, Gx, Gy, Gz;

void MPU6050_Init(void)
{
  uint8_t data;

  // Wake up MPU6050
  data = 0x00;
  HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, MPU6050_PWR_MGMT_1, 1, &data, 1, HAL_MAX_DELAY);
}

```

```c
void MPU6050_Read_Accel(int16_t* accel_data)
{
  uint8_t buffer[6];

  HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, MPU6050_ACCEL_XOUT_H, 1, buffer, 6, HAL_MAX_DELAY);

  accel_data[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
  accel_data[1] = (int16_t)((buffer[2] << 8) | buffer[3]);
  accel_data[2] = (int16_t)((buffer[4] << 8) | buffer[5]);

/*  AFS_SEL       Full Scale Range       LSB Sensitivity
      0               ±2g                 16384 LSB/g
      1               ±4g                  8192 LSB/g
      2               ±8g                  4096 LSB/g
      3               ±16g                 2048 LSB/g */

    Ax = accel_data[0]/2048.0;
    Ay = accel_data[1]/2048.0;
    Az = accel_data[2]/2048.0;
}
```

```c
void MPU6050_Read_Gyro(int16_t* gyro_data)
{
  uint8_t buffer[6];

  HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, MPU6050_GYRO_XOUT_H, 1, buffer, 6, HAL_MAX_DELAY);

  gyro_data[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
  gyro_data[1] = (int16_t)((buffer[2] << 8) | buffer[3]);
  gyro_data[2] = (int16_t)((buffer[4] << 8) | buffer[5]);

 /*  FS_SEL       Full Scale Range       LSB Sensitivity
      0             ± 250 °/s             131 LSB/°/s
      1             ± 500 °/s            65.5 LSB/°/s
      2             ± 1000 °/s           32.8 LSB/°/s
      3             ± 2000 °/s           16.4 LSB/°/s */

    Gx = gyro_data[0]/131.0;
    Gy = gyro_data[1]/131.0;
    Gz = gyro_data[2]/131.0;
}
```

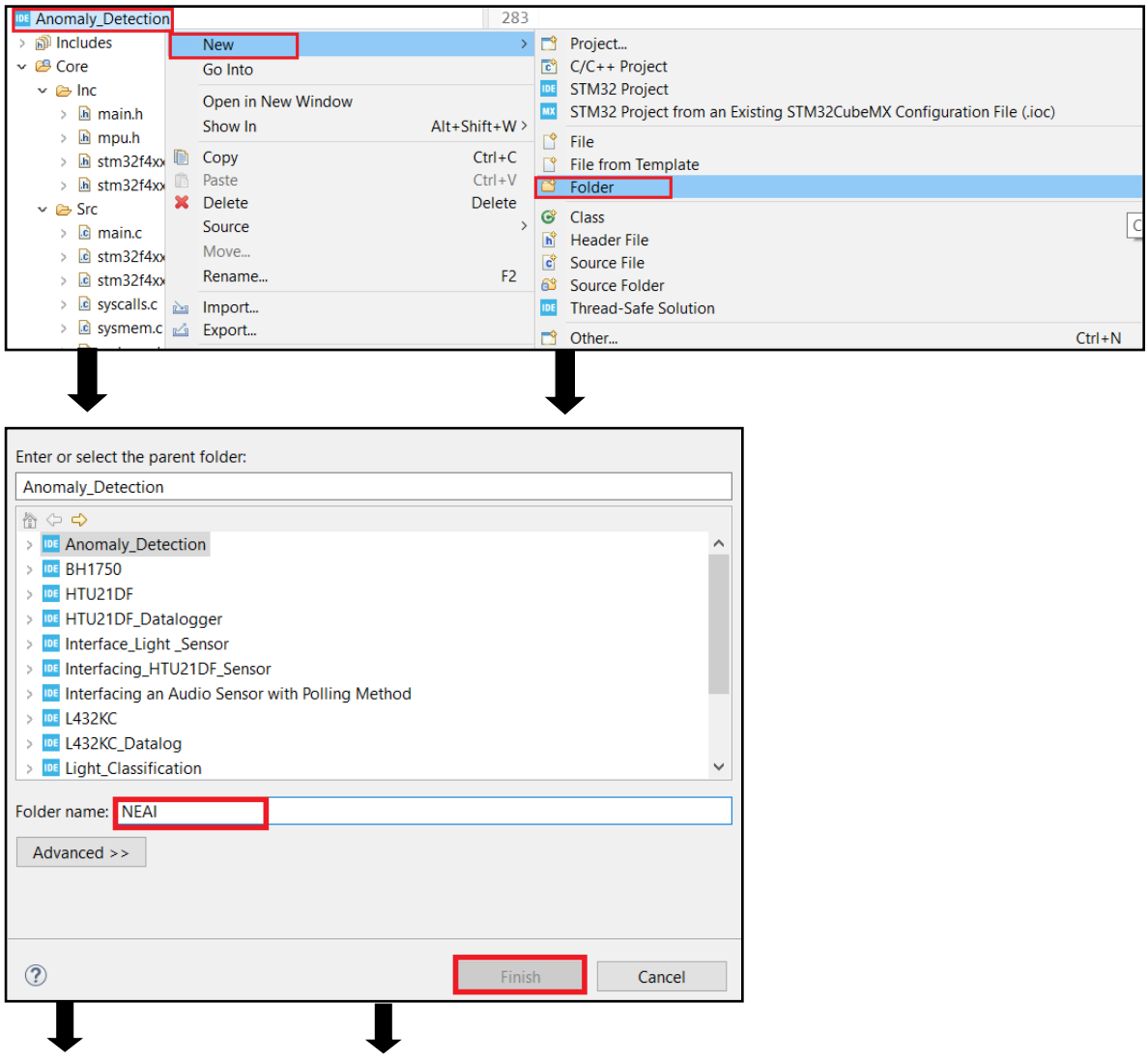
12. Cube IDE automatically generates a code format based on the configurations you have done. Cube IDE uses HAL libraries. Below is the code snippets, please put your code in the appropriate places in the **main.c** file.

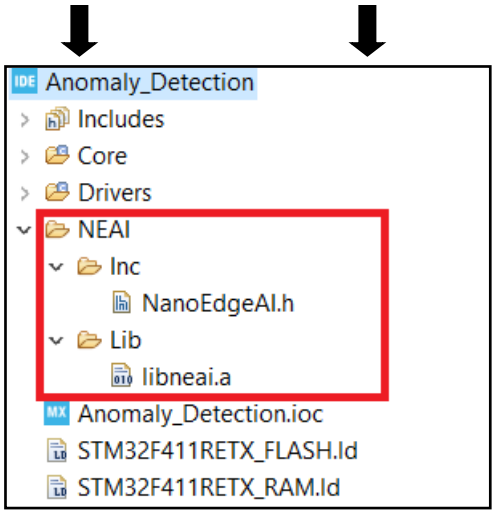
```
/* USER CODE BEGIN Includes */
#include "mpu.h"
#include "stdio.h"
#include "string.h"
#include "NanoEdgeAI.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define DATA_INPUT_USER 256
#define AXIS_NUMBER 3
#define NUMBER_LEARN 40
/* USER CODE END PD */

/* USER CODE BEGIN PV */

float mpu_buffer[DATA_INPUT_USER * AXIS_NUMBER] = { 0 };
uint8_t similarity = 0;

/* USER CODE END PV */

/* USER CODE BEGIN PFP */
void fill_mpu_buffer();
void Train(void);
void Inference(void);
/* USER CODE END PFP */
```

```
105   /* USER CODE BEGIN 2 */
106   MPU6050_Init();
107
108    enum neai_state error_code = neai_anomalydetection_init();
109        if (error_code != NEAI_OK) {
110          printf("Application Init Not Done!!!");
111        }
112        else
113          printf("Application Init Done!!!");
114
115    HAL_Delay (1000);  // wait for 1 sec
116    Train();
117
118
119
120   /* USER CODE END 2 */
121
122   /* Infinite loop */
123   /* USER CODE BEGIN WHILE */
124   while (1)
125   {
126       Inference();
127     /* USER CODE END WHILE */
128
129     /* USER CODE BEGIN 3 */
130   }
131   /* USER CODE END 3 */
132 }
```

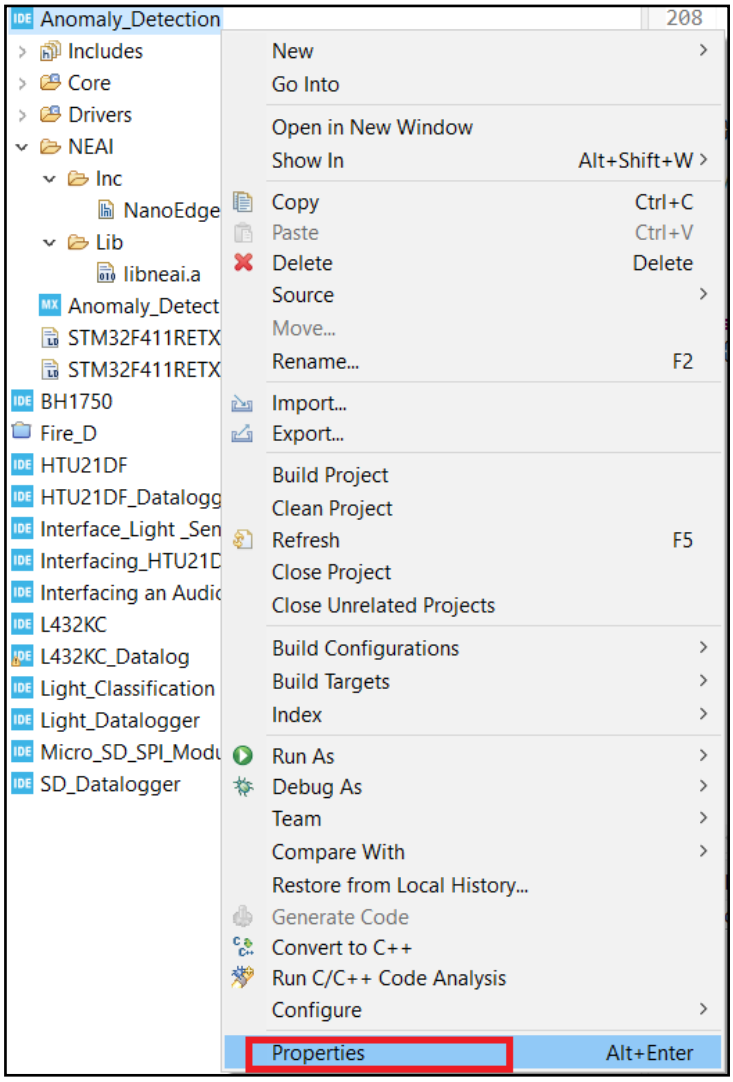
```
/* USER CODE BEGIN 4 */

void fill_mpu_buffer() {
    for (int i = 0; i < DATA_INPUT_USER; i++) {
      MPU6050_Read_Accel(accel_data);
      mpu_buffer[AXIS_NUMBER * i] = Ax;
      mpu_buffer[AXIS_NUMBER * i + 1] = Ay;
      mpu_buffer[AXIS_NUMBER * i + 2] = Az;
  }
}

void Train(void)

{
   for (int i = 0; i < NUMBER_LEARN; i++) {
   fill_mpu_buffer();
   neai_anomalydetection_learn(mpu_buffer);
   printf("Training Cycle No: ");
   printf("%d",i);
   printf("\r\n");
   HAL_Delay(200);
  }
   printf("Training Done");

}

void Inference(void)
{
  fill_mpu_buffer();
  neai_anomalydetection_detect(mpu_buffer, &similarity);
  printf("Similarity Score is: ");
  printf("%d",similarity);
  printf("\r\n");
}

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

/* USER CODE END 4 */
```

13. Right click on the **Anomaly_Detection** 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 **Anomaly_Detection** project. Now right click on this newly created NEAI folder and create two separate new folders and name them **Inc and Lib**.

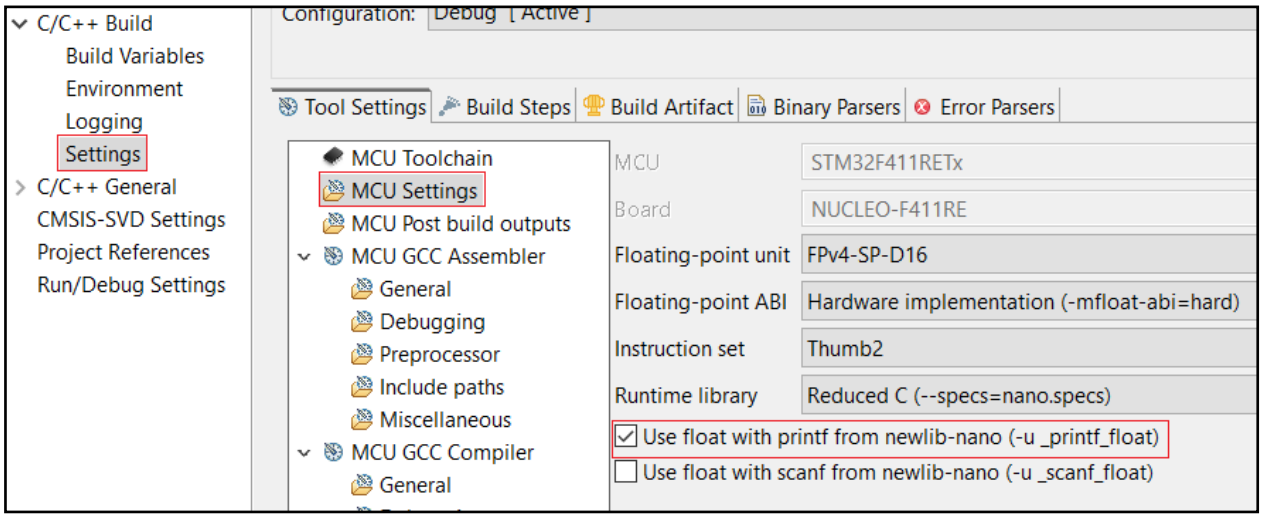




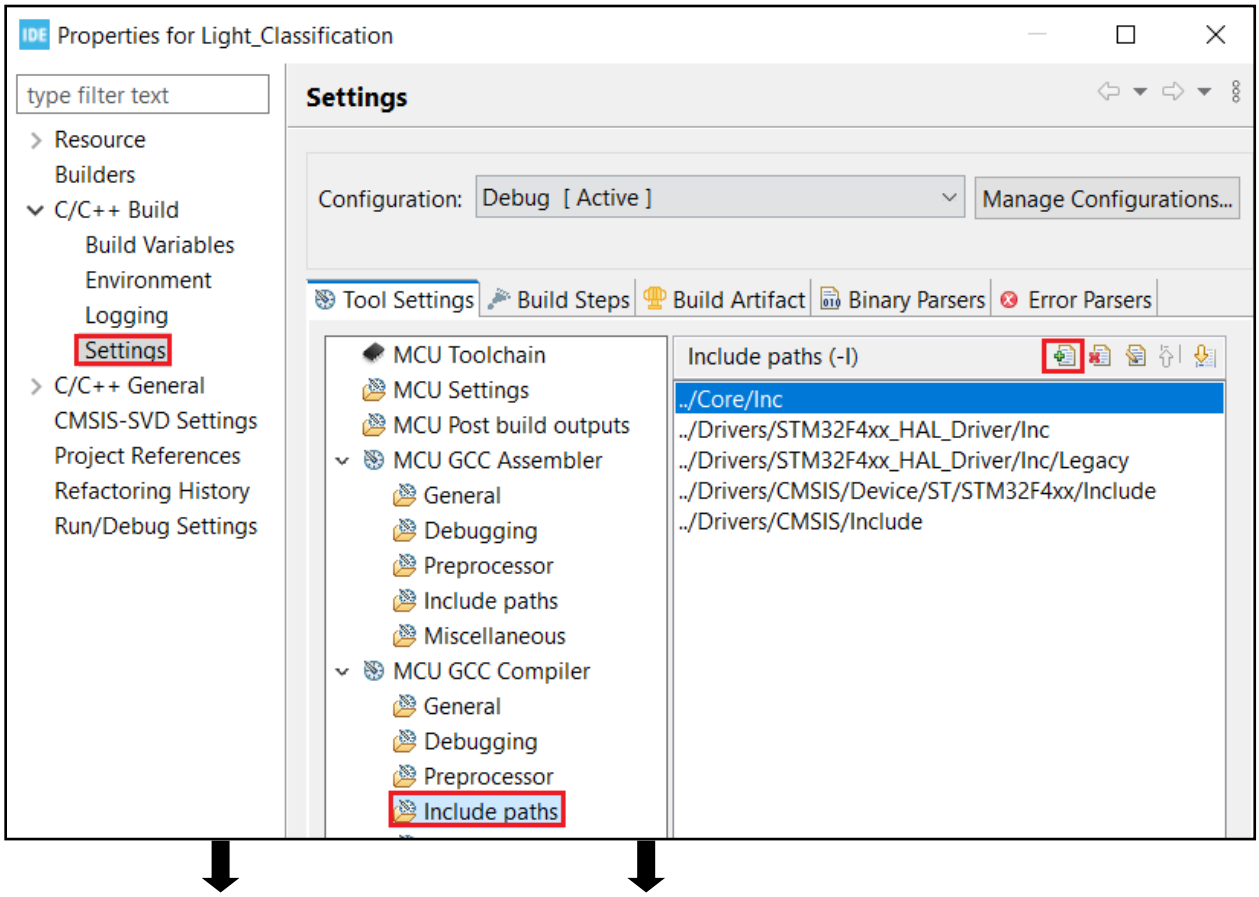
14. Now right click on your project tree, go to **Properties.**

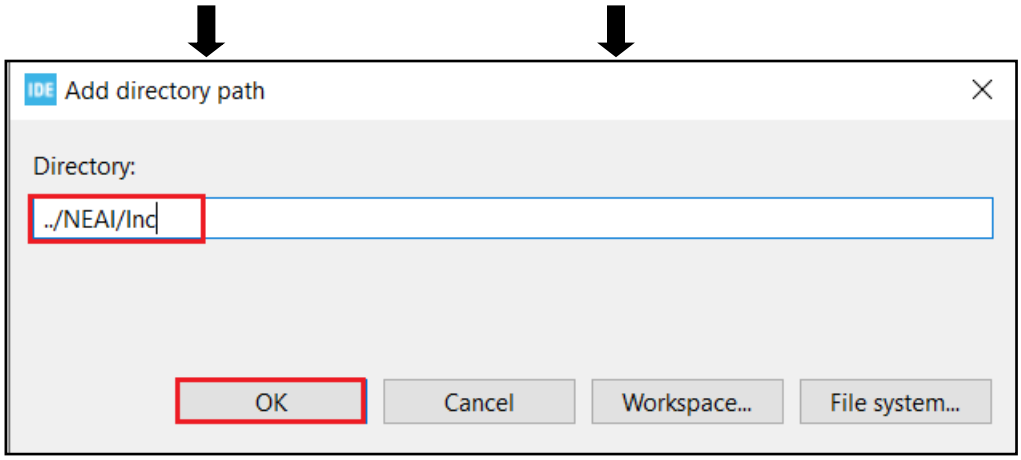


15. After that select **Settings→MCU Settings** and enable the **float printf.**

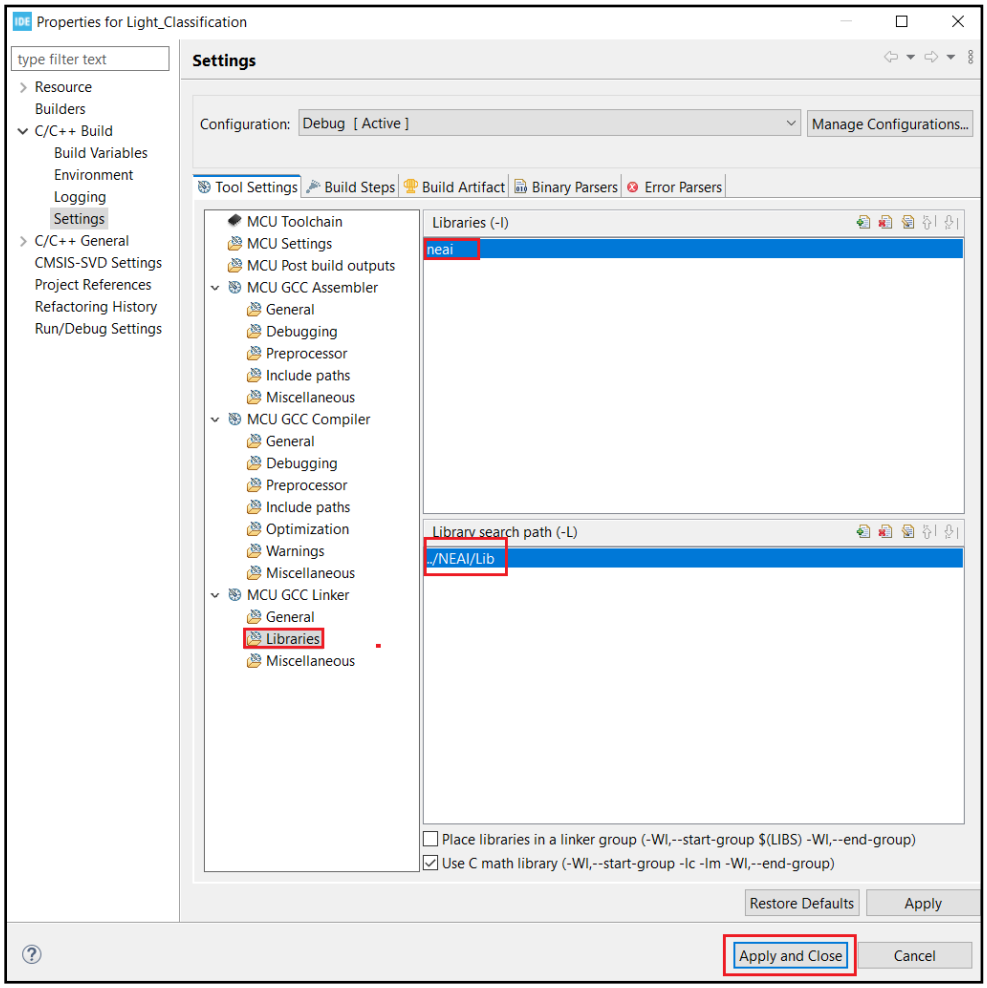


16. Next select **MCU GCC Compiler →Include paths** click on **Add** and in the new window select **Workspace** and select the **Inc** folder and click on **Ok.** The path of the **Inc** folder will be now added.





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.**

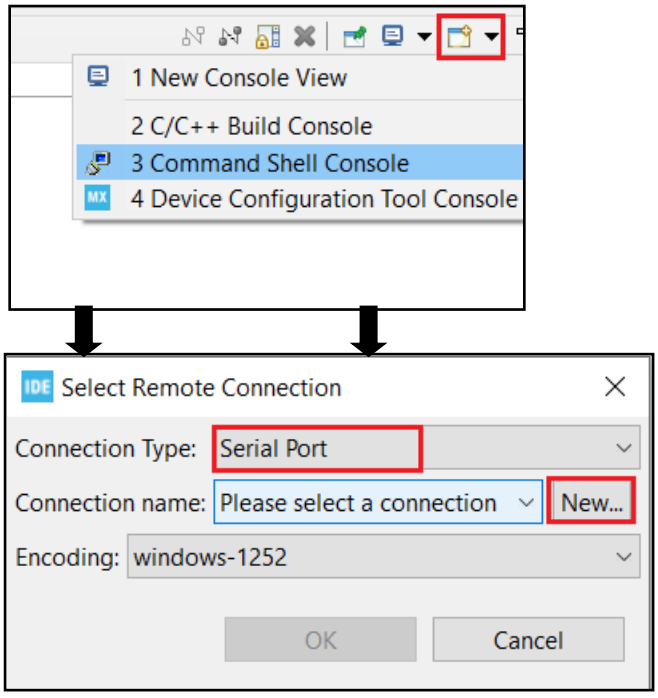


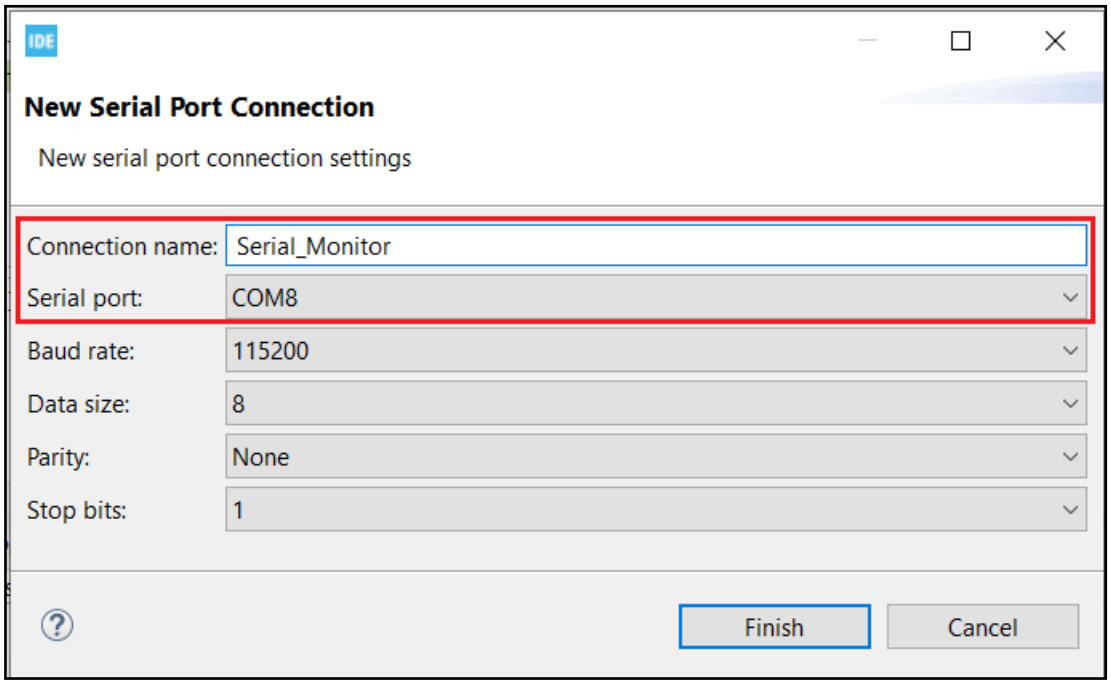
18. Now click on the **build** symbol  on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.

```
20:31:22 **** Incremental Build of configuration Debug for project Anomaly_Detection ****
make -j4 all
arm-none-eabi-size   Anomaly_Detection.elf
   text    data     bss     dec     hex filename
  27112     308    7588   35008    88c0 Anomaly_Detection.elf
Finished building: default.size.stdout


20:31:23 Build Finished. 0 errors, 0 warnings. (took 511ms)
```

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.

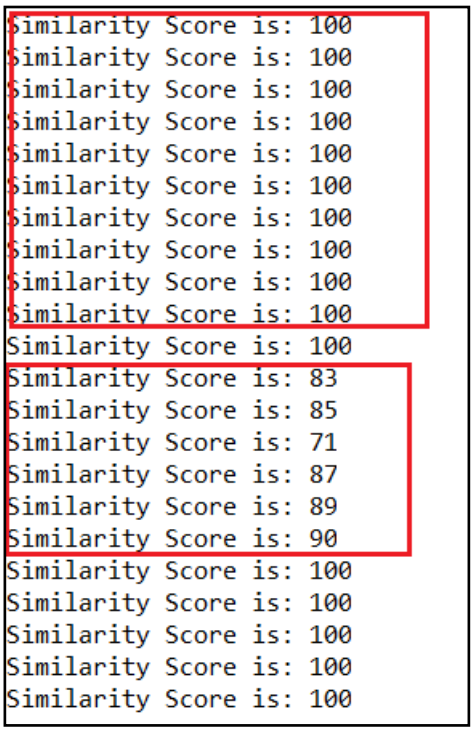
20. In the debug mode, go to the bottom right hand side corner, click on open console. Select the **Connection Type** as **Serial Port**, then click on **New.** In the new window, in **Connection name** give some name to your new connection, and select the **Serial port** correctly. Then click on **Finish**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





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

```
Application Init Done!!!Training Cycle No: 0
Training Cycle No: 1
Training Cycle No: 2
Training Cycle No: 3
Training Cycle No: 4
Training Cycle No: 5
Training Cycle No: 6
Training Cycle No: 7
Training Cycle No: 8
Training Cycle No: 9
Training Cycle No: 10
Training Cycle No: 11
Training Cycle No: 12
Training Cycle No: 13
Training Cycle No: 14
Training Cycle No: 15
Training Cycle No: 16
Training Cycle No: 17
Training Cycle No: 18
Training Cycle No: 19
Training Cycle No: 20
Training Cycle No: 21
```



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,