

## Anomaly Detection: Design of an ML based Anomaly Dete

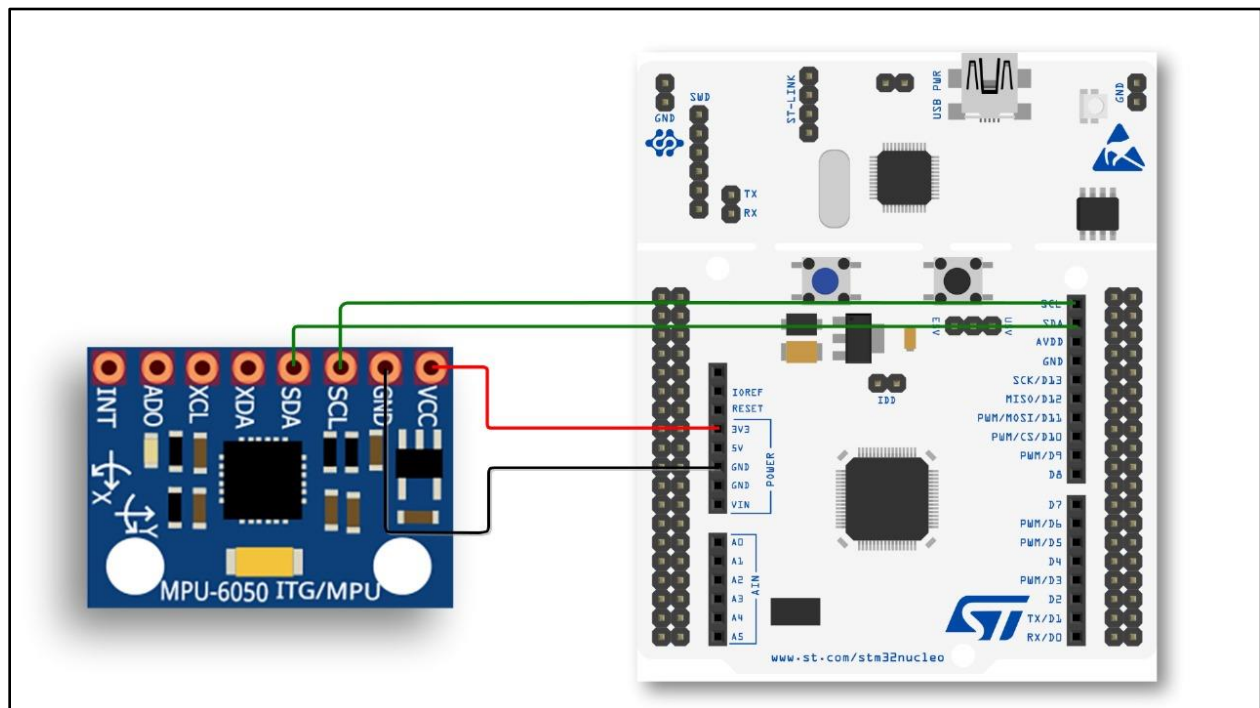
### Objective:

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:

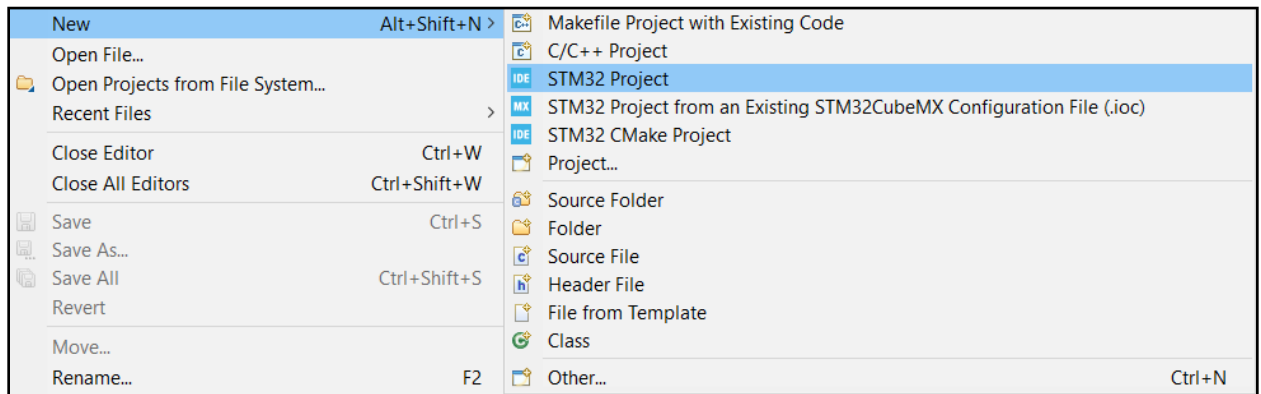
1. STM32 Cube IDE software.
2. Light Sensor (I2C).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.

### Connection Diagram:

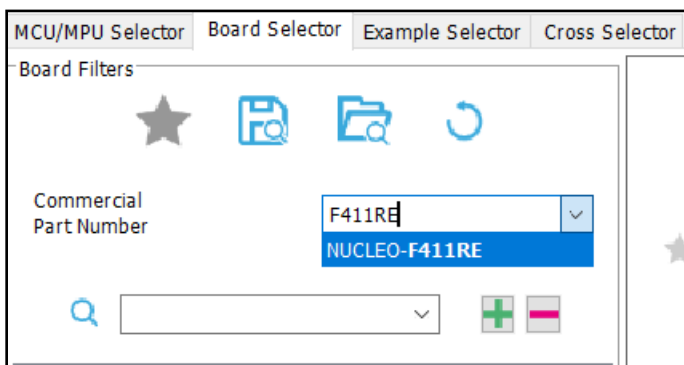


## Procedure:



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



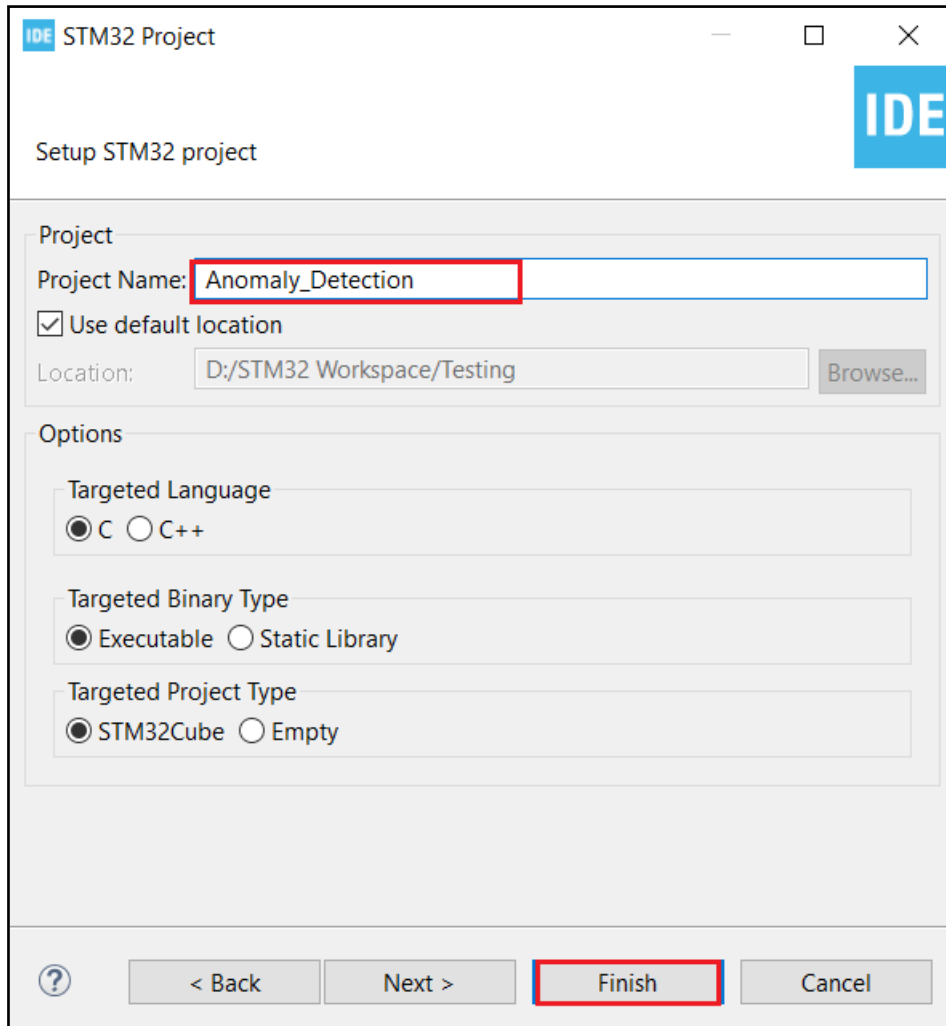
2. A **Target Selection** window will open. Click on **Board Selector**, where you need to select the microcontroller board you are working with.  
(NB: If you are having Nucleo-F401RE, you have to select the said Commercial Part Number)



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**.

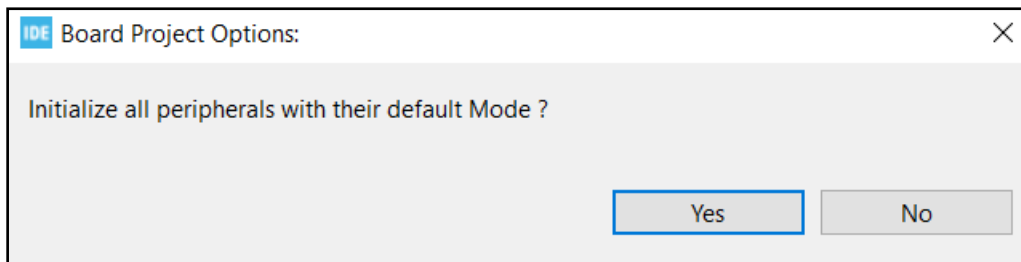
Overview		Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
		NUCLEO-F411RE	Nucleo-64	Active	13.0	STM32F411RET6

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**.



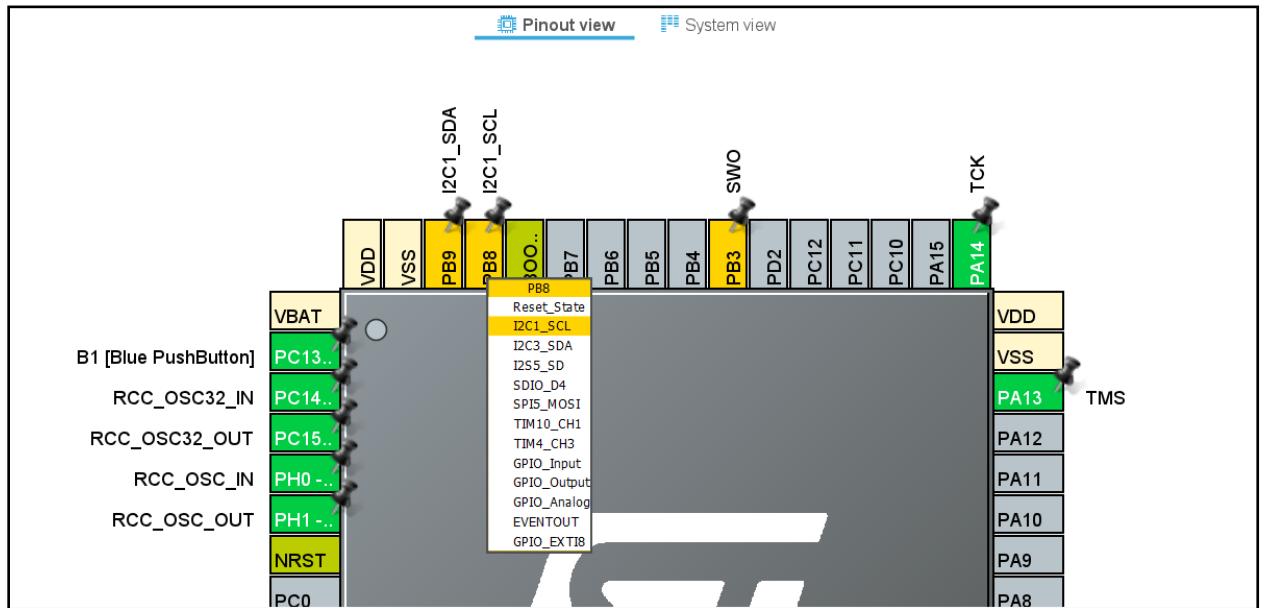
The image shows the 'STM32 Project' setup window in the IDE. The window title is 'IDE STM32 Project'. The main heading is 'Setup STM32 project'. Under the 'Project' section, the 'Project Name' is 'Anomaly\_Detection', which is highlighted with a red box. The 'Use default location' checkbox is checked. The 'Location' is 'D:/STM32 Workspace/Testing', with a 'Browse...' button next to it. Under the 'Options' section, there are three groups of radio buttons: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', and 'Finish' (highlighted with a red box), followed by a 'Cancel' button.

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

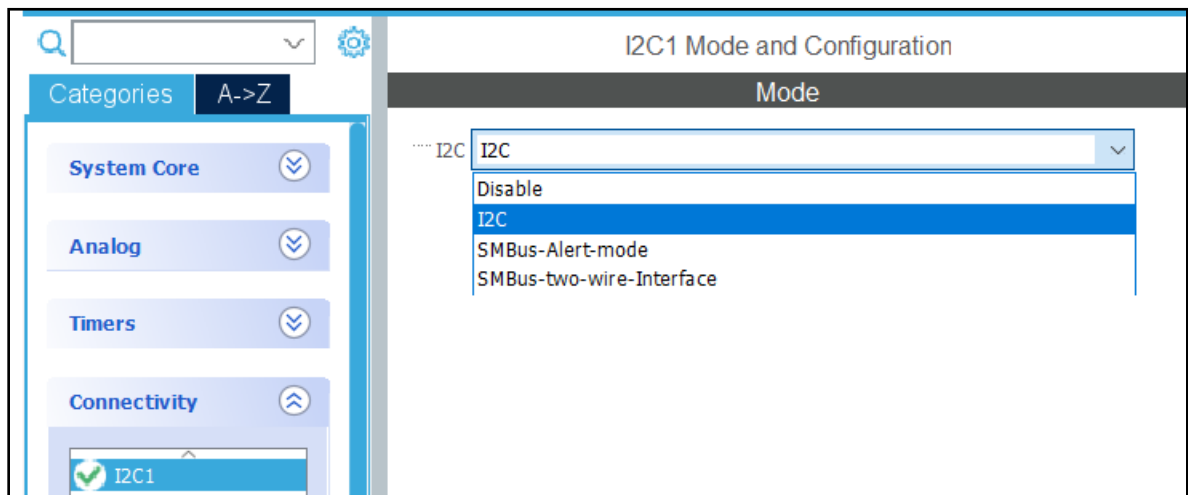


The image shows the 'Board Project Options' dialog box. The title is 'IDE Board Project Options:'. The main text is 'Initialize all peripherals with their default Mode ?'. At the bottom right, there are two buttons: 'Yes' (highlighted with a blue box) and 'No'.

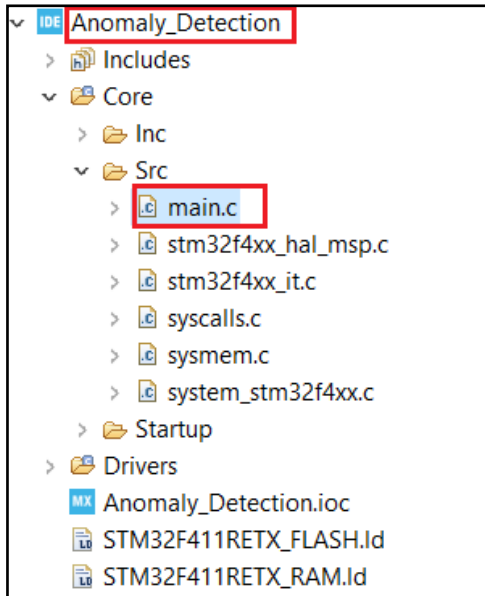
- In the **Pinout & Configuration** tab, click on **PB8** pin and select it as an **I2C1\_SCL** and **PB9** pin as an **I2C1\_SDA**.



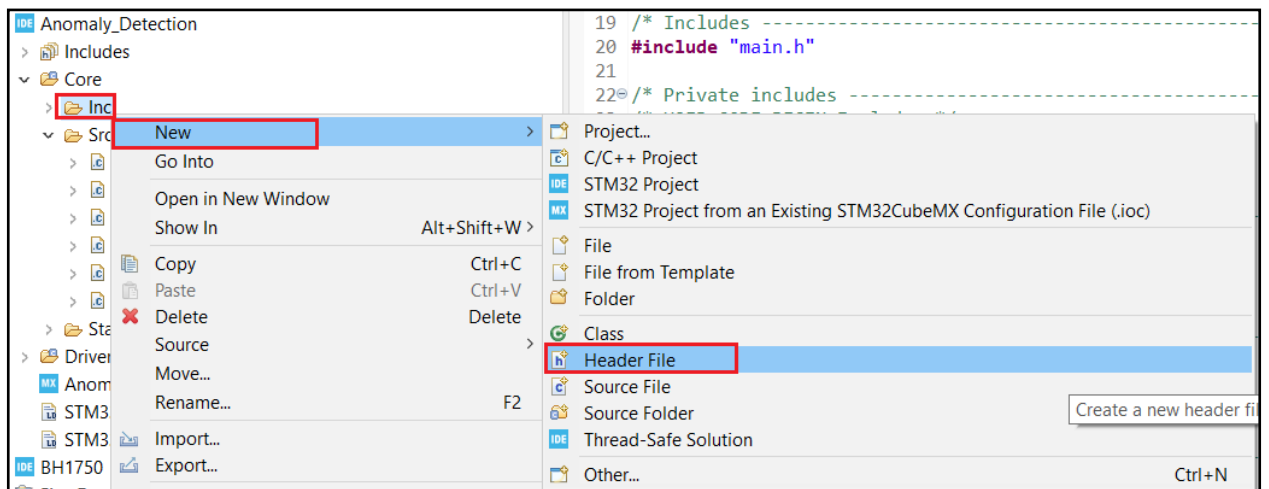
- Next on the left-hand side under **Categories** → **Connectivity**, select I2C1 and enable it.



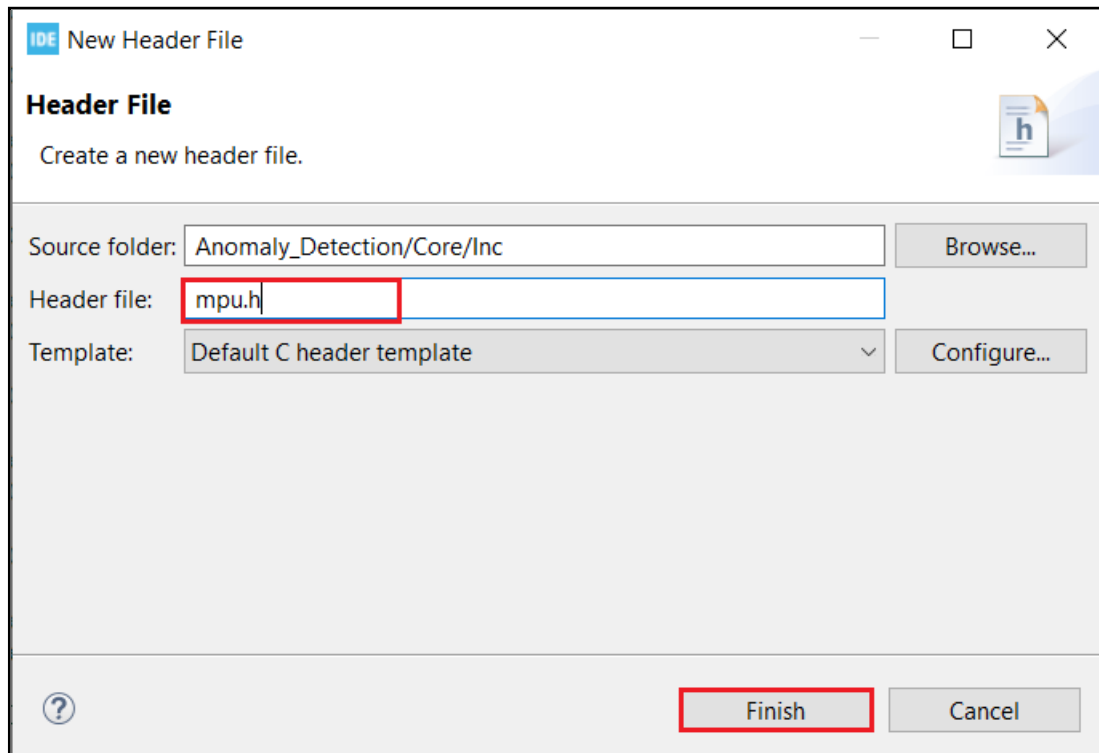
8. Press **Ctrl+S** to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example, I have named my project as (Anomaly\_Detection) **Anomaly\_Detection**→**Core** →**Src**→**main.c** (double click to load the code).



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.

```

2  #ifndef INC_MPU_H_
3  #define INC_MPU_H_
4
5  extern I2C_HandleTypeDef hi2c1;
6
7  #define MPU6050_ADDR          0xD0
8  #define MPU6050_PWR_MGMT_1    0x6B
9  #define MPU6050_ACCEL_XOUT_H   0x3B
10 #define MPU6050_ACCEL_YOUT_H   0x3D
11 #define MPU6050_ACCEL_ZOUT_H   0x3F
12 #define MPU6050_TEMP_OUT_H     0x41
13 #define MPU6050_GYRO_XOUT_H    0x43
14 #define MPU6050_GYRO_YOUT_H    0x45
15 #define MPU6050_GYRO_ZOUT_H    0x47
16
17 int16_t accel_data[3];
18 int16_t gyro_data[3];
19
20 float Ax, Ay, Az, Gx, Gy, Gz;
21
22 void MPU6050_Init(void)
23 {
24     uint8_t data;
25
26     // Wake up MPU6050
27     data = 0x00;
28     HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, MPU6050_PWR_MGMT_1, 1, &data, 1, HAL_MAX_DELAY);
29 }
30

```

```

31 void MPU6050_Read_Accel(int16_t* accel_data)
32 {
33     uint8_t buffer[6];
34
35     HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, MPU6050_ACCEL_XOUT_H, 1, buffer, 6, HAL_MAX_DELAY);
36
37     accel_data[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
38     accel_data[1] = (int16_t)((buffer[2] << 8) | buffer[3]);
39     accel_data[2] = (int16_t)((buffer[4] << 8) | buffer[5]);
40
41     /* AFS_SEL      Full Scale Range      LSB Sensitivity
42         0           ±2g                    16384 LSB/g
43         1           ±4g                    8192  LSB/g
44         2           ±8g                    4096  LSB/g
45         3           ±16g                   2048  LSB/g */
46
47     Ax = accel_data[0]/2048.0;
48     Ay = accel_data[1]/2048.0;
49     Az = accel_data[2]/2048.0;
50 }
51

```

```

52 void MPU6050_Read_Gyro(int16_t* gyro_data)
53 {
54     uint8_t buffer[6];
55
56     HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, MPU6050_GYRO_XOUT_H, 1, buffer, 6, HAL_MAX_DELAY);
57
58     gyro_data[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
59     gyro_data[1] = (int16_t)((buffer[2] << 8) | buffer[3]);
60     gyro_data[2] = (int16_t)((buffer[4] << 8) | buffer[5]);
61
62     /* FS_SEL      Full Scale Range      LSB Sensitivity
63         0           ± 250 °/s            131  LSB/°/s
64         1           ± 500 °/s            65.5 LSB/°/s
65         2           ± 1000 °/s           32.8 LSB/°/s
66         3           ± 2000 °/s           16.4 LSB/°/s */
67
68     Gx = gyro_data[0]/131.0;
69     Gy = gyro_data[1]/131.0;
70     Gz = gyro_data[2]/131.0;
71 }
72

```

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.

```
23  /* USER CODE BEGIN Includes */
24  #include "mpu.h"
25  #include "stdio.h"
26  #include "string.h"
27  #include "NanoEdgeAI.h"
28  /* USER CODE END Includes */
29
30  /* Private typedef -----*/
31  /* USER CODE BEGIN PTD */
32
33  /* USER CODE END PTD */
34
35  /* Private define -----*/
36  /* USER CODE BEGIN PD */
37  #define DATA_INPUT_USER 256
38  #define AXIS_NUMBER 3
39  #define NUMBER_LEARN 40
40  /* USER CODE END PD */
41
42  /* USER CODE BEGIN PV */
43
44  /* USER CODE END PV */
45
46  /* USER CODE BEGIN PFP */
47
48  /* USER CODE END PFP */
```

Code snippets highlighted in red boxes:

- Lines 24-27: `#include "mpu.h"`, `#include "stdio.h"`, `#include "string.h"`, `#include "NanoEdgeAI.h"`
- Lines 37-39: `#define DATA_INPUT_USER 256`, `#define AXIS_NUMBER 3`, `#define NUMBER_LEARN 40`
- Lines 54-55: `float mpu_buffer[DATA_INPUT_USER * AXIS_NUMBER] = { 0 };`, `uint8_t similarity = 0;`
- Lines 65-67: `void fill_mpu_buffer();`, `void Train(void);`, `void Inference(void);`



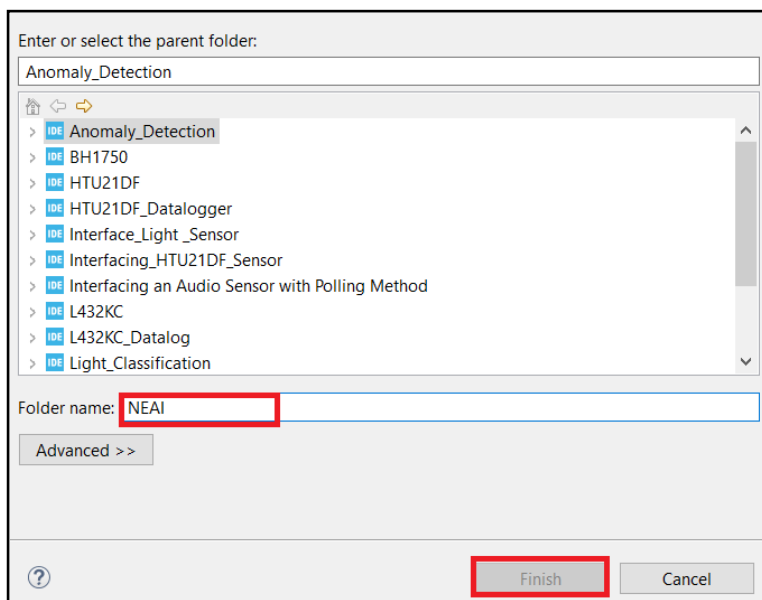
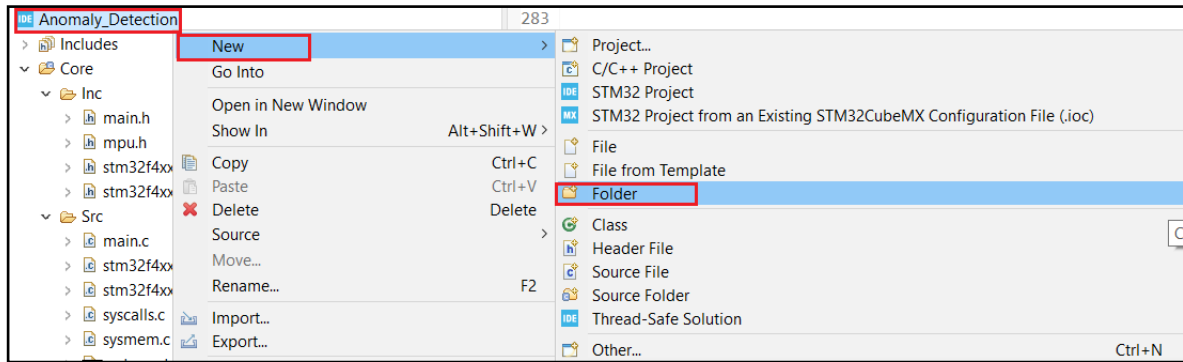
```
105  /* USER CODE BEGIN 2 */
106  MPU6050_Init();
107
108  enum neai_state error_code = neai_anomalydetection_init();
109  if (error_code != NEAI_OK) {
110      printf("Application Init Not Done!!!");
111  }
112  else
113      printf("Application Init Done!!!");
114
115  HAL_Delay (1000); // wait 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 }
```

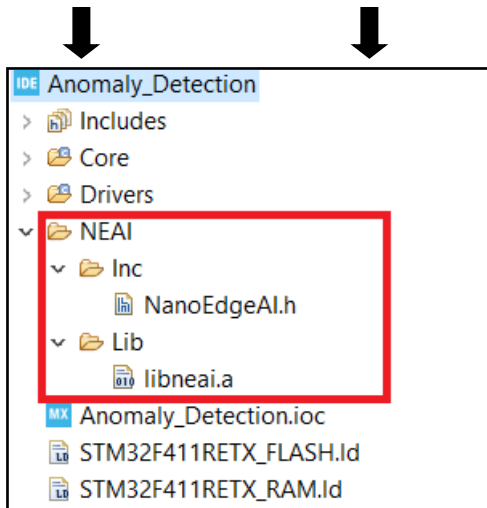
```

284 /* USER CODE BEGIN 4 */
285
286 void fill_mpu_buffer() {
287     for (int i = 0; i < DATA_INPUT_USER; i++) {
288         MPU6050_Read_Accel(accel_data);
289         mpu_buffer[AXIS_NUMBER * i] = Ax;
290         mpu_buffer[AXIS_NUMBER * i + 1] = Ay;
291         mpu_buffer[AXIS_NUMBER * i + 2] = Az;
292     }
293 }
294
295 void Train(void)
296 {
297     for (int i = 0; i < NUMBER_LEARN; i++) {
298         fill_mpu_buffer();
299         neai_anomalydetection_learn(mpu_buffer);
300         printf("Training Cycle No: ");
301         printf("%d", i);
302         printf("\r\n");
303         HAL_Delay(200);
304     }
305     printf("Training Done");
306 }
307
308 void Inference(void)
309 {
310     fill_mpu_buffer();
311     neai_anomalydetection_detect(mpu_buffer, &similarity);
312     printf("Similarity Score is: ");
313     printf("%d", similarity);
314     printf("\r\n");
315 }
316
317 int __io_putchar(int ch){
318     HAL_UART_Transmit(&huart2, (uint8_t *) &ch, 1, HAL_MAX_DELAY);
319     return ch;
320 }
321
322 /* USER CODE END 4 */

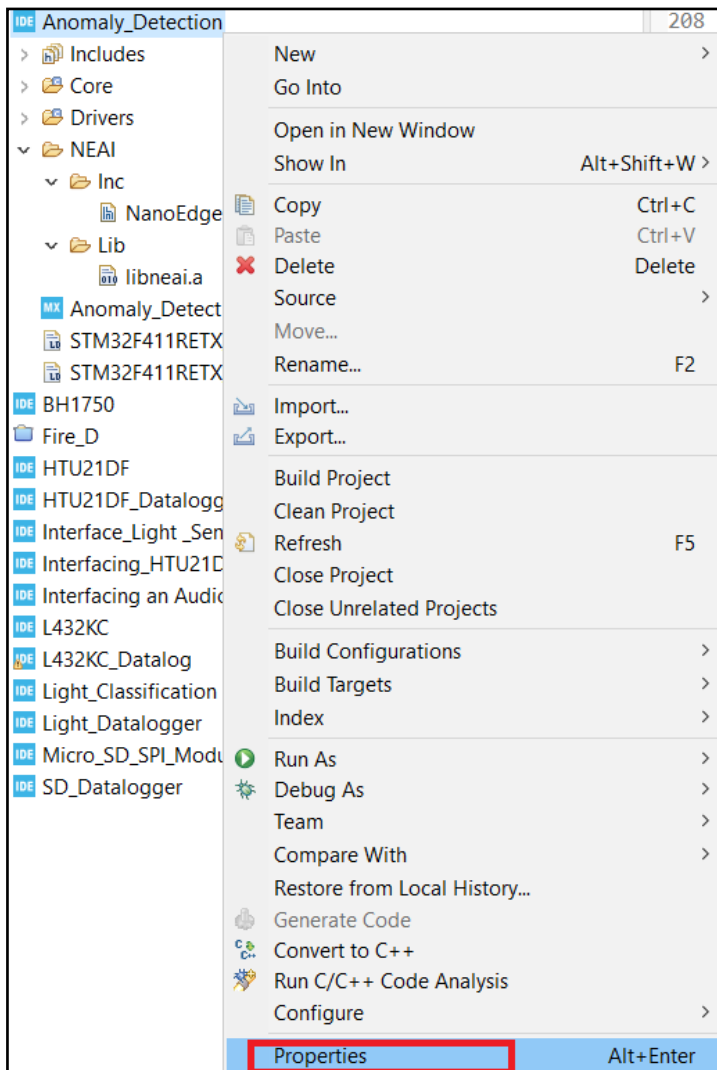
```

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**.

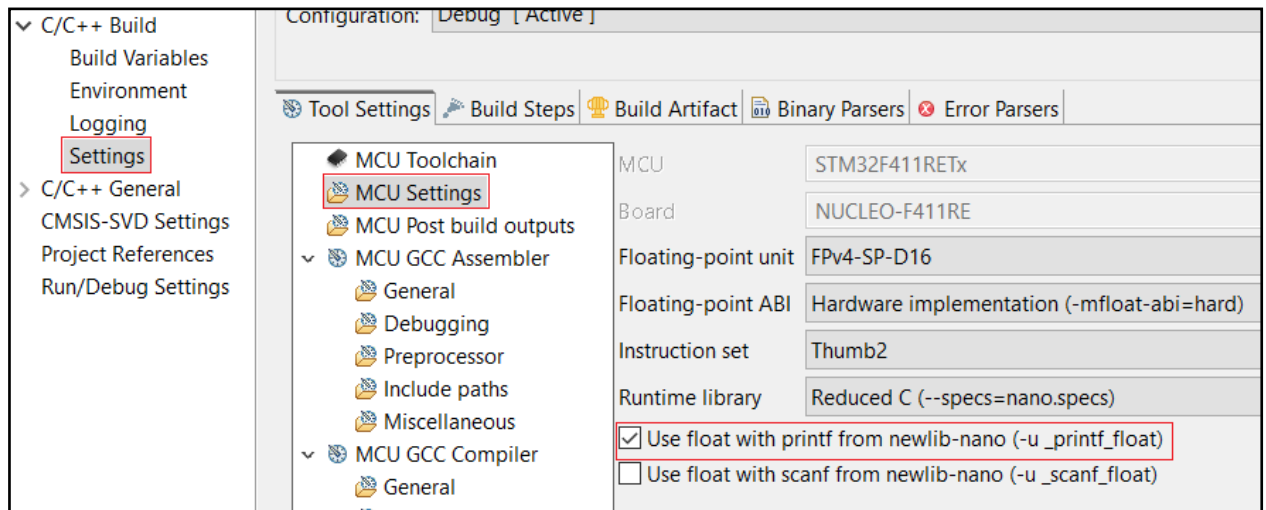




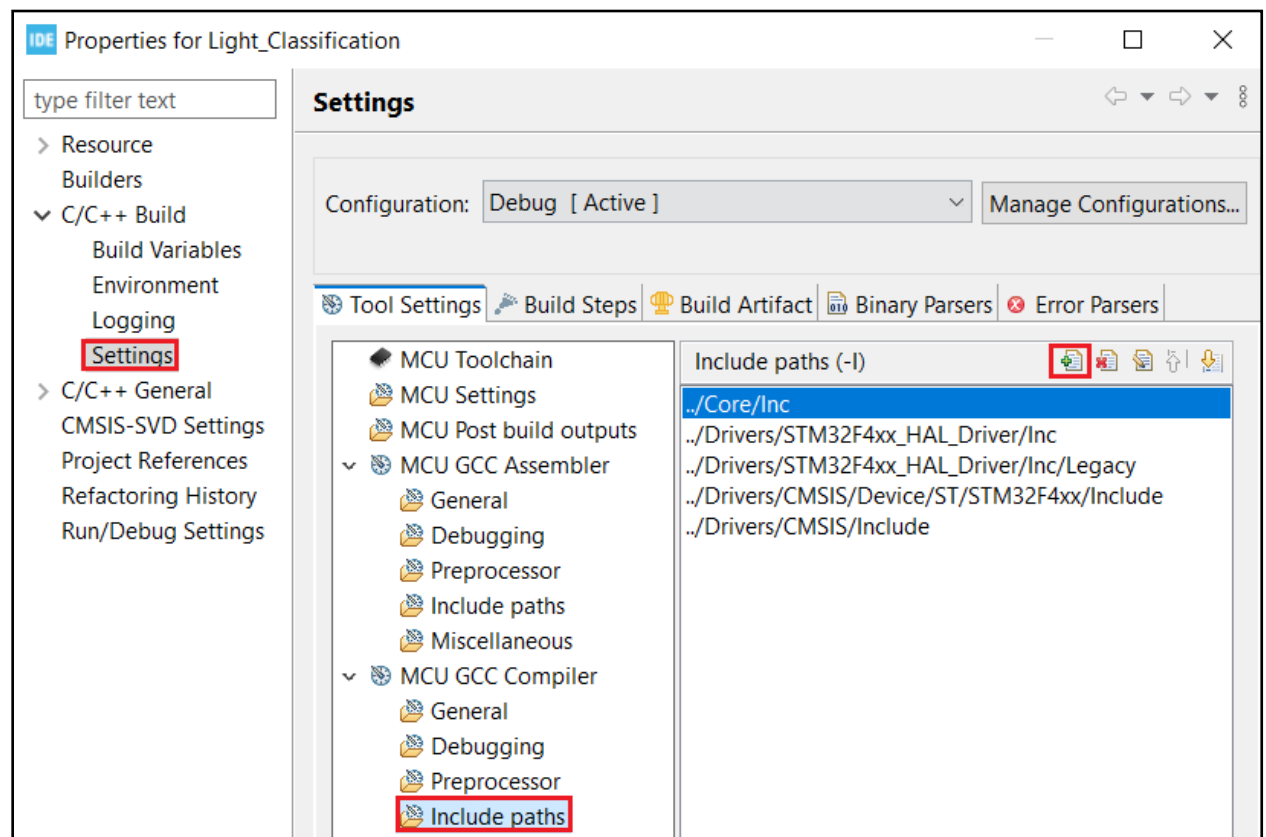
14. Now right click on your project tree, go to **Properties**.

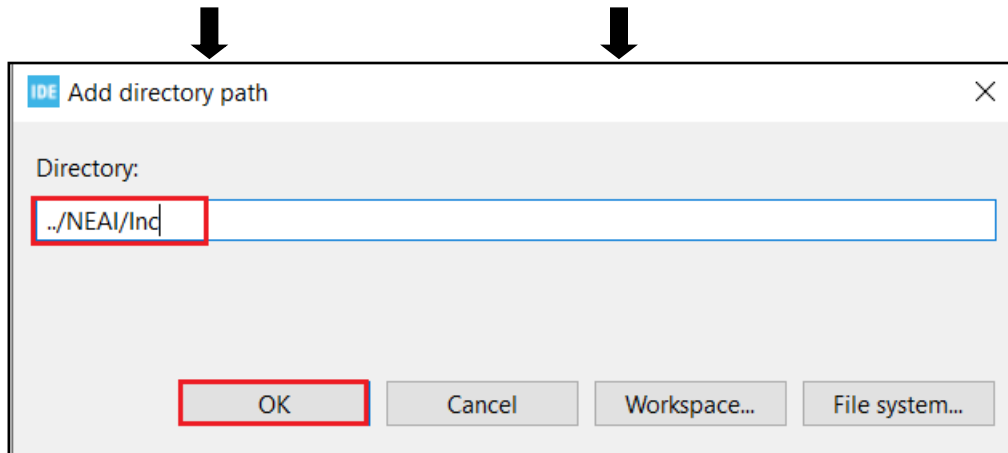


15. After that select **Settings**→**MCU Settings** and enable the **float printf**.

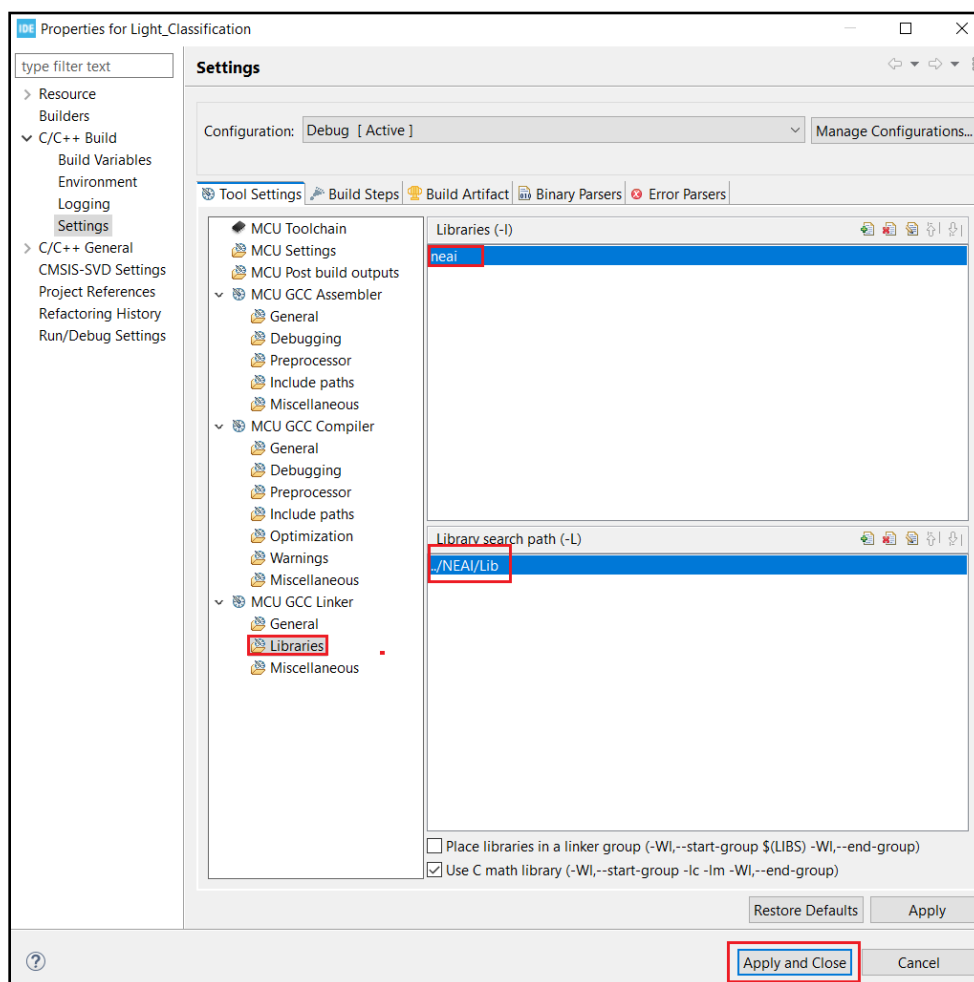



16. Next select **MCU GCC Compiler** →**Include paths** click on **Add** and in the new window select **Workspace** and select the **Inc** folder and click on **Ok**. The path of the **Inc** folder will be now added.






17. Next go to **MCU GCC Linker → Libraries**. 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**.



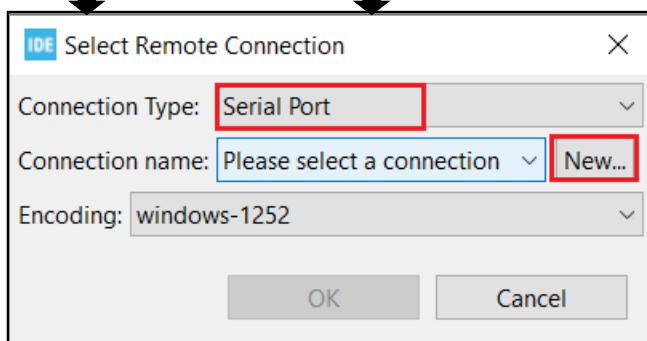
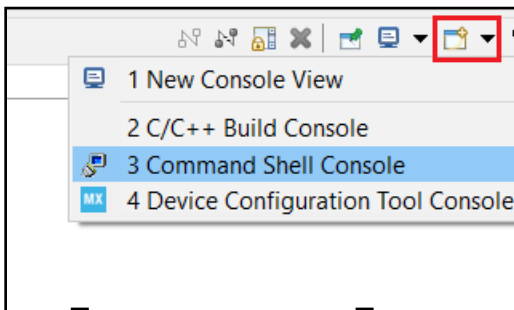
18. Now click on the **build** symbol  on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.

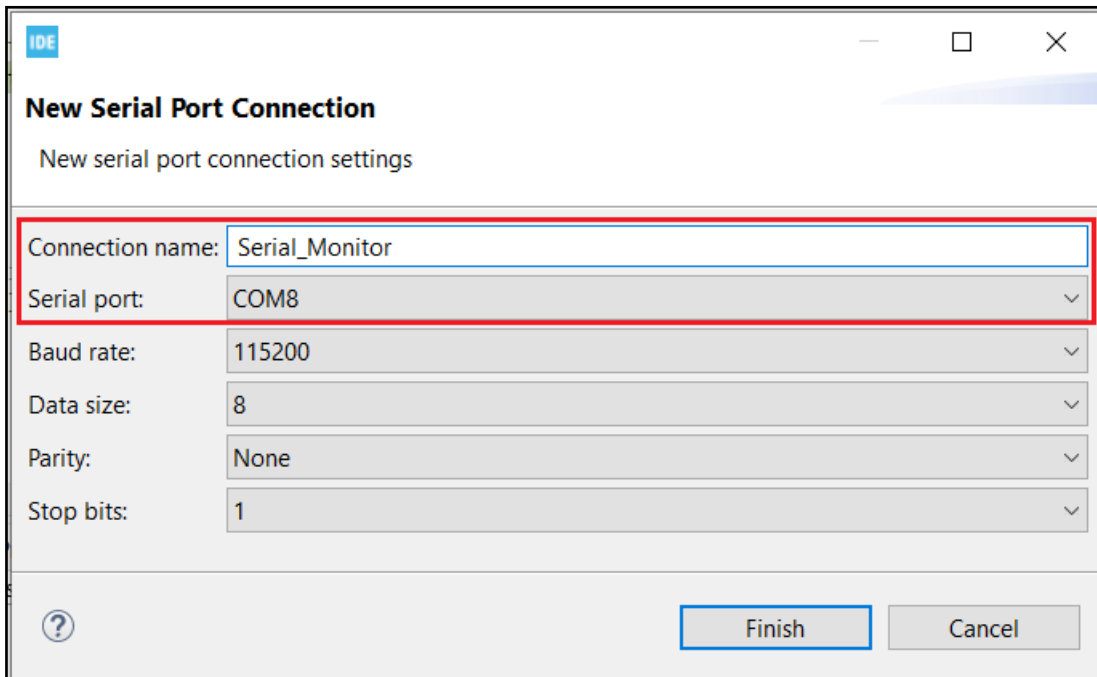
```
20:31:22 **** Incremental Build of configuration Debug for project Anomaly_Detection ****
make -j4 all
arm-none-eabi-size  Anomaly_Detection.elf
   text    data     bss     dec     hex filename
 27112    308    7588   35008   88c0 Anomaly_Detection.elf
Finished building: default.size.stdout

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.





The image shows a 'New Serial Port Connection' dialog box from an IDE. The dialog has a title bar with 'IDE' and standard window controls. The title is 'New Serial Port Connection' and the subtitle is 'New serial port connection settings'. The main area contains several fields: 'Connection name:' with the value 'Serial\_Monitor', 'Serial port:' with 'COM8', 'Baud rate:' with '115200', 'Data size:' with '8', 'Parity:' with 'None', and 'Stop bits:' with '1'. Each field has a dropdown arrow on the right. At the bottom, there is a question mark icon on the left, and 'Finish' and 'Cancel' buttons on the right. A red rectangle highlights the 'Connection name' and 'Serial port' fields.

IDE

### New Serial Port Connection

New serial port connection settings

Connection name: Serial\_Monitor

Serial port: COM8

Baud rate: 115200


Data size: 8

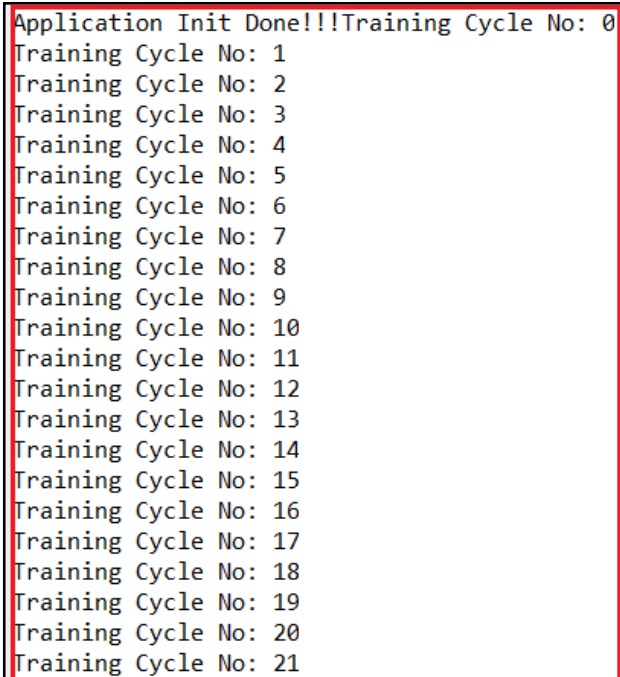
Parity: None

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**.



The image shows a console window with a red border. It contains the following text: 'Application Init Done!!!Training Cycle No: 0' followed by a list of 'Training Cycle No: 1' through 'Training Cycle No: 21'.

```
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
```



```
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 83
Similarity Score is: 85
Similarity Score is: 71
Similarity Score is: 87
Similarity Score is: 89
Similarity Score is: 90
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
Similarity Score is: 100
```

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,