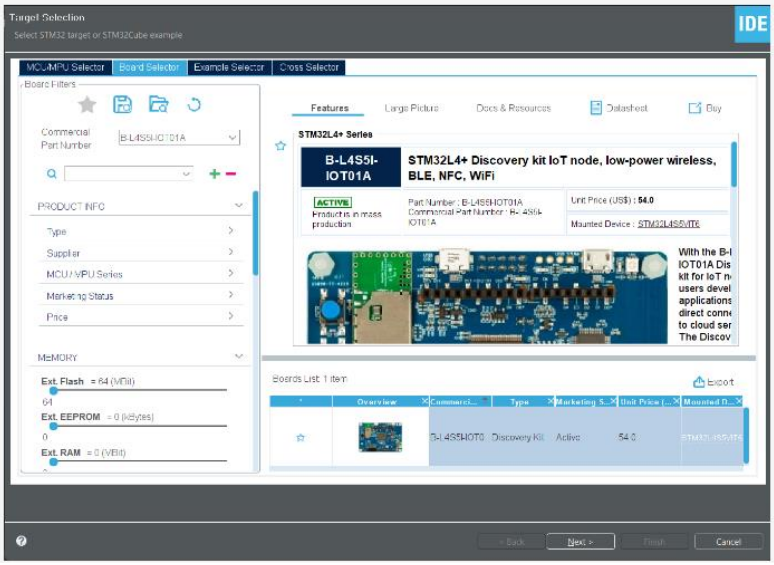
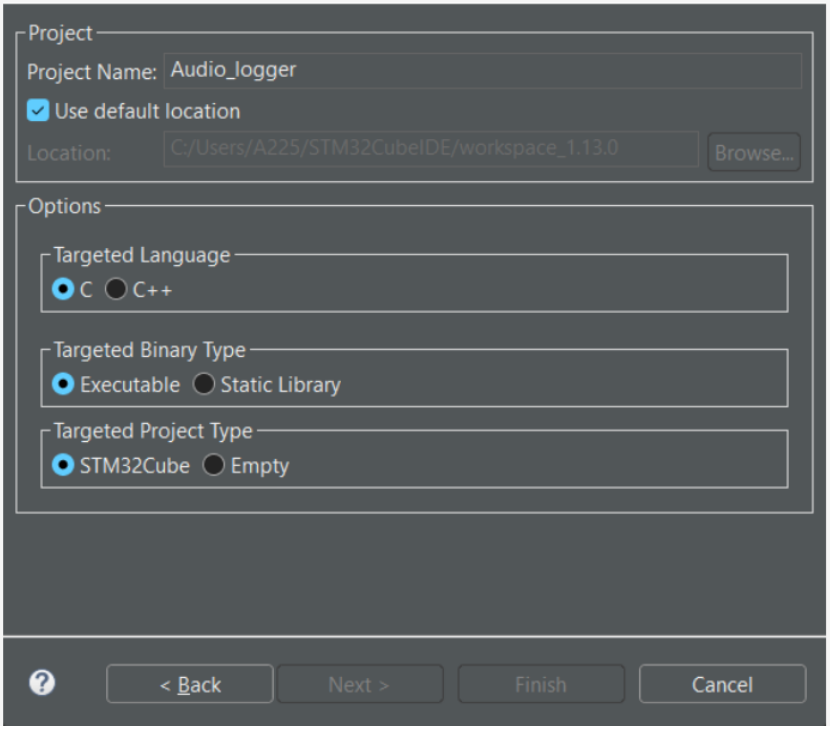
Creating New Project:

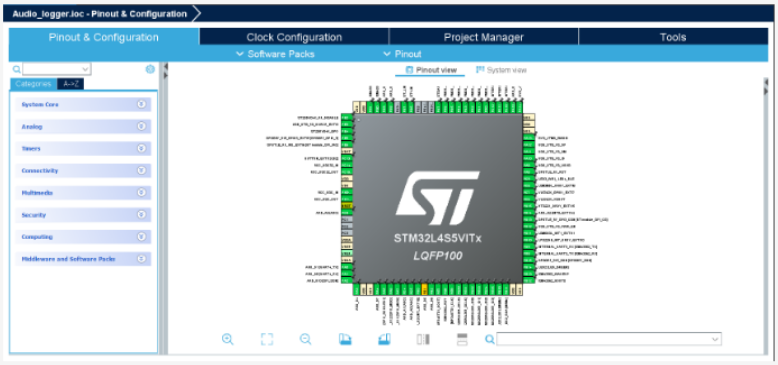
To create a project, go to File > New Project. There will be a new window opened for you to select the board or MCU. In this project, we will use B-L4S5I-IOT01A the part is also linked below. Go to Board Selector and select the board by typing in the commercial part number (B-L4S5I-IOT01A) and click Next.



After clicking Next, type the project name (In this case: Audio_logger) and click on Finish.

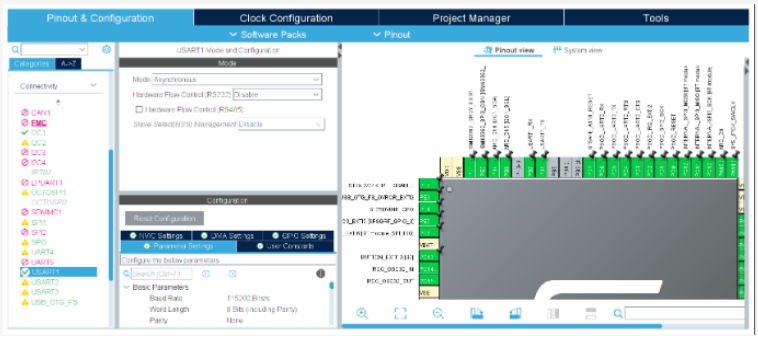


After clicking finish, the STM32 CUBE IDE will automatically download the software packages that your board requires. After the download is complete you will see a workspace something like this (a .ioc file will be opened):



Assigning ports:

In this project, we will be sending messages to our console from the device (STM32 Board). STM32 Board uses UART-based communication. Now, to enable UART in your board go to connectivity > USART1 and check if it is in Asynchronous Mode. Now, you will see a warning symbol near USART1. To remove that please reset PB3, PB4, and PB5. After doing so, you will see that the warning is now gone, and a green tick near the USART1 is visible.



The codes of each section are given below:

1. The User Include Section

Copy Code

```
#include "stdio.h"
#include "string.h"
```

```
23 /* USER CODE BEGIN Includes */
24 #include "stdio.h"
25 #include "string.h"
26 /* USER CODE END Includes */
```

2. The Private Macro Section

Copy Code

```
#define DATA_INPUT_USER 512
#define AXIS_NUMBER 1
```

```
38 /* USER CODE BEGIN PM */
39 #define DATA_INPUT_USER 512
40 #define AXIS_NUMBER 1
41 /* USER CODE END PM */
```

3. The Private Variables Section

Copy Code

```
float mic_data[DATA_INPUT_USER * AXIS_NUMBER] = {0};
```

3. The Private Variables Section

Copy Code

```
float mic_data[DATA_INPUT_USER * AXIS_NUMBER] = {0};
```

```
61 /* USER CODE BEGIN PV */
62 float mic_data[DATA_INPUT_USER * AXIS_NUMBER] = {0};
63 /* USER CODE END PV */
```

4. The Private Functions Sections

Copy Code

```
void fill_mic();
void Log();
```

```
81 /* USER CODE BEGIN PFP */
82 void fill_mic();
83 void Log();
84 /* USER CODE END PFP */
```

5. The User Code Begin 3 inside while loop

Copy Code

```
Log();
```

5. The User Code Begin 3 inside while loop

Copy Code

```
Log();
```

```
140   while (1)
141   {
142     /* USER CODE END WHILE */
143
144     /* USER CODE BEGIN 3 */
145       Log();
146   }
147   /* USER CODE END 3 */
```

6. The User Code Begin 4

Copy Code

```
void fill_mic() {
        for (int i = 0; i < DATA_INPUT_USER; i++) {
                mic_data[AXIS_NUMBER * i] = HAL_ADC_GetValue(&hadc1);
        }
}

void Log(){
        HAL_ADC_Start(&hadc1);
        fill_mic();
        for (int i = 0; i < DATA_INPUT_USER; i++) {
                printf("%.2f",mic_data[AXIS_NUMBER * i]);
                printf(" ");
        }
        printf("\r\n");
```

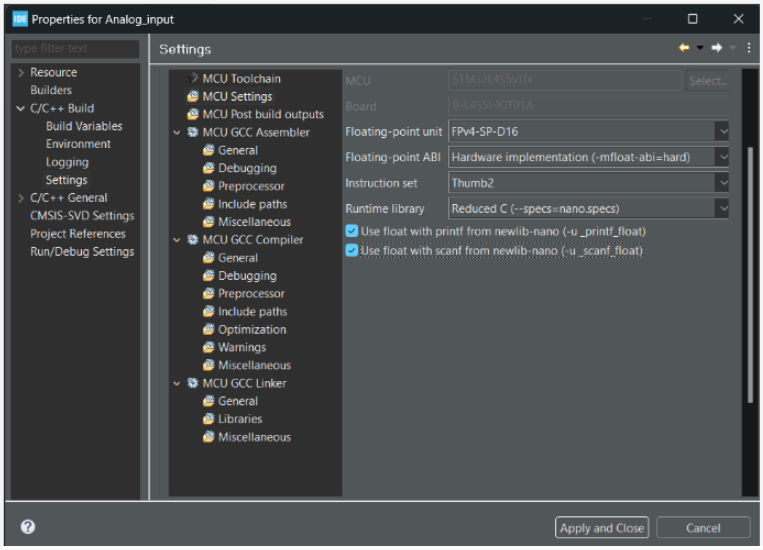
```
905 /* USER CODE BEGIN 4 */
906 void fill_mic() {
907     for (int i = 0; i < DATA_INPUT_USER; i++) {
908         mic_data[AXIS_NUMBER * i] = HAL_ADC_GetValue(&hadc1);
909     }
910 }
911
912 void Log(){
913     HAL_ADC_Start(&hadc1);
914     fill_mic();
915     for (int i = 0; i < DATA_INPUT_USER; i++) {
916         printf("%.2f",mic_data[AXIS_NUMBER * i]);
917         printf(" ");
918     }
919     printf("\r\n");
920     HAL_Delay(500);
921 }
922 int __io_putchar(int ch){
923     HAL_UART_Transmit(&huart1, (uint8_t *) &ch, 1, HAL_MAX_DELAY);
924     return ch;
925 }
926 /* USER CODE END 4 */
```

Understanding: The Include section is filled with stdio.h and string.h libraries because we are using both print statements as well as string datatype. The private variables section is added because we need a variable to store the data from the analog device, and the variable is an array. The private macro section defines 2 variables, data input user, and axis number. The data input is the number of data inputs in a single line, and the axis number is the number of axis of the input (An accelerometer will have three axis x, y, and z).

The User Code Begin 3 is the actual code that will run on the microcontroller device. The User Code Begin 4 is for transmitting characters (data) from the microcontroller to the console using UART and for two main functions of the projects. The log code is used to create the data log, and the fill mic is used to fill the array with the data inputs.

Removing the printf error:

Whenever you try to use printf or scanf statement there will be a compiler error, which simply means that the compiler does not allow using printf or scanf. So, to allow the compiler to recognize them, right-click on the project > Properties > C/C++ Build > Settings > MCU Settings and select both scanf and printf statements. The code will now look good.



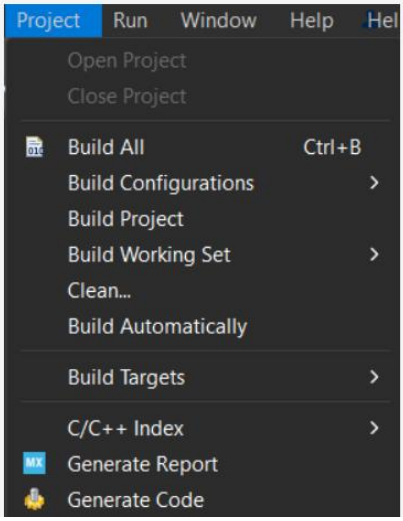
Click on Apply and Close.

Connections:

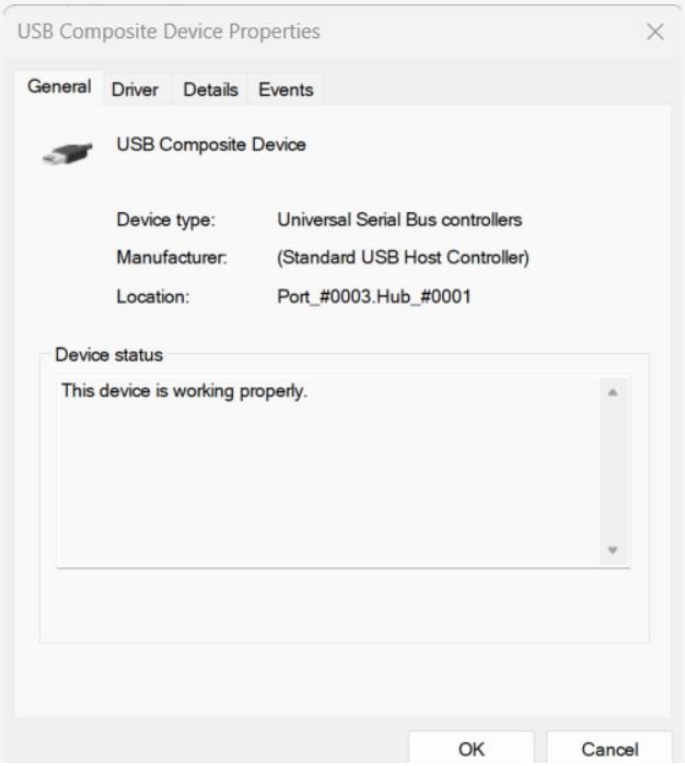
Scheme-it Embedded | Free Online Schematic and Diagramming Tool | DigiKey
Embedded Client cannot be loaded.
Minimum window size is 400x300. Current size is850x150

Building and Debugging:

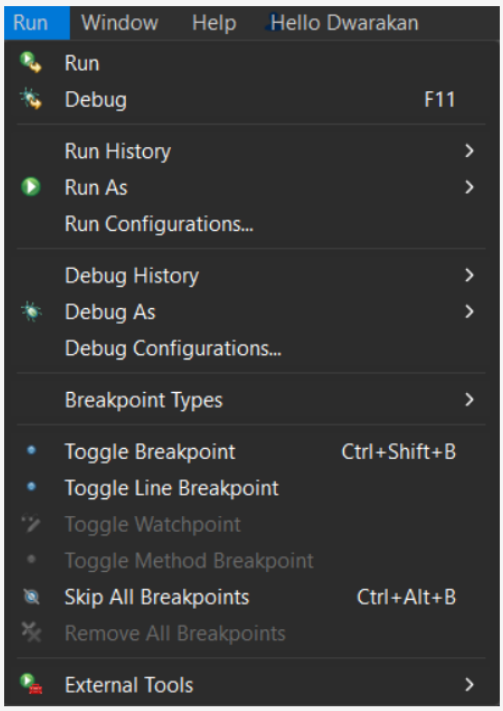
To build the project go to Project > Build Project. If you see no errors, then the code is ready to be debugged.



After building the project connect your microcontroller board and check the communication port in the device manager (Ex: COM7). The device name, in my case, can be different from yours. Do not panic; just check for the port number for any STM-based device.

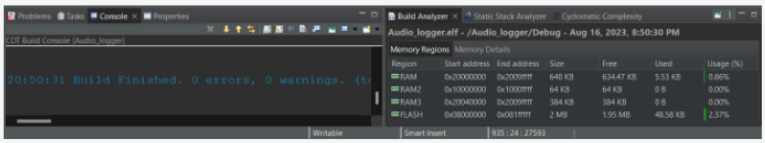


In this case, the port is 3. Now, debug the main.c file and check whether the console shows "Download verified successfully".

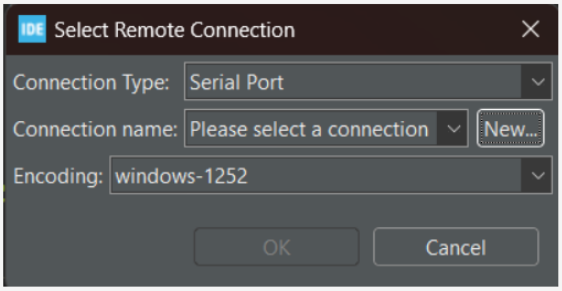


After debugging the code, you should use the ability of the device to use UART to send messages to your console, and for that, you have to create a console mentioning the port that your device is connected to. To do so, go to the console window below the workspace and select Command shell console under Open Console.

After debugging the code, you should use the ability of the device to use UART to send messages to your console, and for that, you have to create a console mentioning the port that your device is connected to. To do so, go to the console window below the workspace and select Command shell console under Open Console.

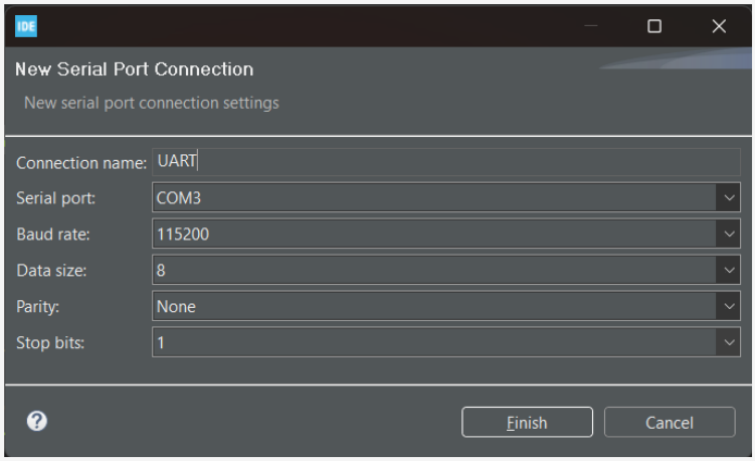


After clicking on it, you will see a new window opened for you to choose the port and name of the console. The connection type is serial port and click on "New".

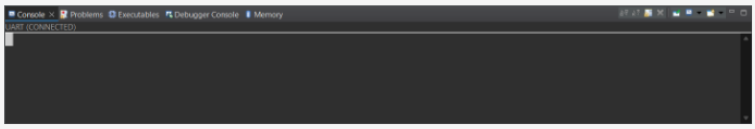


After clicking New, type the name of your port, select the serial port to which your device is connected, and click Finish.

After clicking New, type the name of your port, select the serial port to which your device is connected, and click Finish.



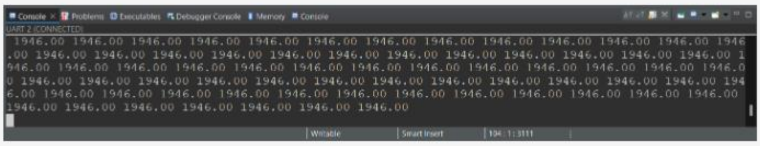
After clicking finish, you will see a new console created for you, and it will tell you that the console is connected.



Now, you are ready to resume the debugging. Click on "resume" which is available on the topmost taskbar. After that, you will see that your microcontroller is sending you the values from the analog device.

Result:

So, as we have completed everything, click on the resume button, and see the data inputs from your device. As you can see there are 512 data inputs per line. These data can be used for further interesting projects using NANOEDGE AI Studio (The next blog).



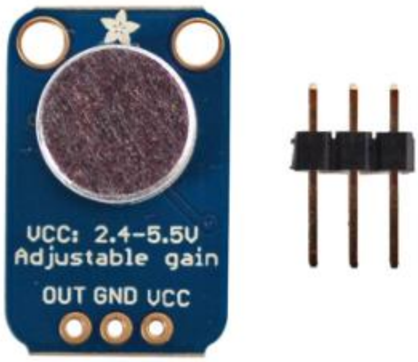
Key Parts and Components

Mfr Part # B-L4S5I-IOT01A

STM32L4+ DISCOVERY KIT IOT NODE,

STMicroelectronics

$54.00 Details



Mfr Part # 1063

AMP MICROPHONE ADJUSTABLE

Adafruit Industries LLC

$6.95 Details