

## 1.0 Overall Project and Hardware Description

### 1.1 Project Description

#### 1.1.1 Background

In this project, an audio frequency spectrum analyzer was designed by using STM32 Nucleo-F446RE development board. This low-cost frequency spectrum analyzer can be used to analyze the frequency content of input audio signals. Frequency spectrum of an input signal possesses vital information, and the same has been employed for various applications ranging from engineering to biomedical, such as speech recognition, electrocardiogram (ECG) signal analysis, fault diagnosis of induction motors, music analysis and so on.

Analyzing the spectra of electrical signals provides information about dominant frequency, power, distortion, harmonics, bandwidth and other spectral components, which are not easily detectable in time-domain waveforms. An example of block diagram of spectrum analyzer is shown in Figure 1.

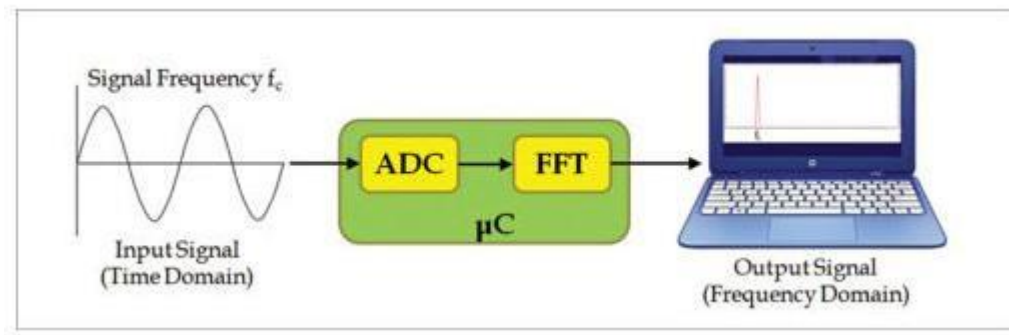


Figure 1: An example of block diagram of spectrum analyzer

#### 1.1.2 Project Flow

The project flow begins with the input audio signals connection from the microphone. The audio signal will be amplified by the amplifier according to the resistant value of the potentiometer. Next, these analog input audio signals will be converted into digital signals through Analog-to-Digital Converter (ADC). These ADC values/signals will be used as the inputs for Complex Fast Fourier Transform (CFFT). To be specific, `arm_cfft_f32` function is invoked to convert the time domain input into frequency domain complex data output. Then, the magnitude of these complex numbers will be calculated using `arm_cmplx_mag_f32` function and converted into the corresponding LED value through some mathematical calculations. Finally, these LED values will be display through 8x8 LED matrix. Every column of 8x8 LED module represents a frequency range and the number of blinking LED in a column will decrease as the signal frequency increases.

## **1.2 Hardware Requirement and Connection**

### **1.2.1 Hardware Components**

There are several hardware components used to build the audio frequency spectrum analyzer in this project. The main component is the STM32 Nucleo-F446RE development board with a mini USB cable. This board has an ARM cortex M4 core running at maximum operating frequency of 180MHz. In addition, it has 512 Kbytes of flash memory and 128 Kbytes of SRAM. Furthermore, it has two push buttons (USER and RESET) and three LEDs which include USB connection (LD1), user LED (LD2) and power LED (LD3).

Next, KY-037 microphone sensor module is required to receive audio signals. There are four input/output pins namely, digital output signal (DO), positive supply voltage (+V), ground (GND) and also analog output signal (AO). The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module. The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value. The sensitivity of microphone module can be controlled by adjusting the potentiometer.

The third component required is the MAX7219 LED dot matrix module. This module used MAX7219 chip to drive 8x8 LED matrix which can be used in various electronic display panels. This 8x8 LED matrix has two power lines (VCC and GND) and three data lines (DIN, CS and CLK). This LED matrix module has an operating voltage ranges from 4.7C-5.3V and an operating current ranges from 320mA up to a maximum value of 2A.

### **1.2.2 Hardware Connections**

This part explains the hardware connections between the development board to both microphone sensor and LED matrix modules. The VCC pin and GND pins of both modules are connected to 3.3V/5V and GND pins of the board. Next, the analog output signal (AO) pin of microphone is connected to AO pin of the board and the digital output signal (DO) pin is left unconnected since it is unused. The three data lines (DIN, CS and CLK) of the LED matrix module are connected to D11 (PA7), D10 (PB6) and D13 (PA5) of the board respectively. Figure 2 show the complete pin connections between the development board to both microphone sensor and LED matrix modules.

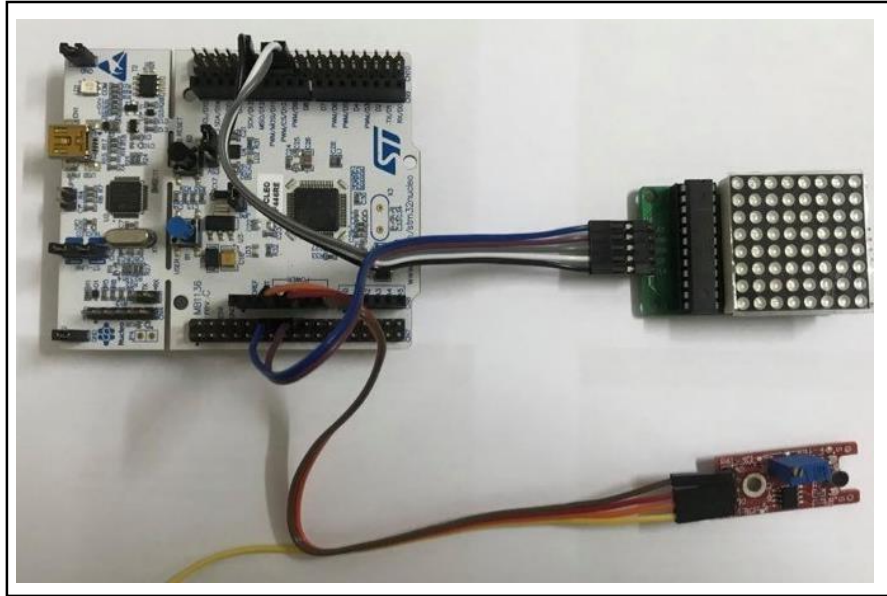


Figure 2: The pin connections between board to microphone and LED matrix modules

### 1.3 Software Requirement

Instead of using STM32CubeMX and Keil, this project is using Mbed OS, which is an open-source operating system for platforms using ARM microcontrollers. Mbed OS provides an abstraction layer for the microcontrollers it runs on, so that developers can focus on writing C/C++ applications. The written High Level Language (HLL) code will be loaded into the board through following description. A simple C code for printing “Hello World” is used in the example of Arduino with core ATmega328P. A binary file will be generated and loaded into the board by the compiler and assembler. In the case of Nucleo with core of ARM Cortex M4, two binary files are generated for the HLL code and Mbed OS. Then, both files are linked and loaded into the board. The benefit of doing this is that Mbed OS supports deterministic and multithreaded real time software execution with an RTOS.

Three additional software, which are Notepad++, command prompt and MobaXterm will be used to complete this project. Notepad++ is used to write or edit the source code “main.cpp”. Next, command prompt is used to compile the source code and download the compiled program into the targeted board. At first, the directory is changed to a directory which consists of the main source code. Then, mbed compile command will be invoked by specifying the name of build target which is NUCLEO\_F446RE and also the installed toolchain which is GCC\_ARM. The `flash` argument will automatically flash the compiled program onto the board. The third software, MobaXterm is used to serially communicate with the board and display the output message on the screen. To setup the connection to the board, click on “Session” button and choose “Serial”. Next, select COM3 for the serial port and choose 115200 (one hundred fifteen thousand two hundred) bits per second for the speed.

## 2.0 Install and set up Mbed OS

This section is separated into two parts. If you wish to look at the fully documentation for Mbed OS including the installation and setting up by the official party, it can be viewed from the section 2.1. If you wish to follow the steps that our team have done for this project, you can skip to the section 2.2 of this section.

### 2.1 Official documentation for Mbed OS

The installation and setting up can be done by following the official guideline from the provided link:

[https://os.mbed.com/docs/mbed-os/v6.9/build-tools/install-and-set-up.html?fbclid=IwAR1UHjStM6sbA\\_OnucjxD\\_9TFU27OIYLVdkp23ivHHkTu9\\_QMKWxT06JxY](https://os.mbed.com/docs/mbed-os/v6.9/build-tools/install-and-set-up.html?fbclid=IwAR1UHjStM6sbA_OnucjxD_9TFU27OIYLVdkp23ivHHkTu9_QMKWxT06JxY)

### 2.2 Steps to install and setting up Mbed OS used in this project

The installation in this part also using the steps from the link in Part 1. But it is only running the necessary steps where the optional installation and setup is not taken.

1. You can go to the link as provided in section 2.1, and choose the installer based on your device operating system as shown in Figure3. (Note that our team is developing the project using operating system Windows 10)

#### Installers

The installers are a great way to get started with Mbed OS:

- [Windows installer.](#)
- [macOS installer.](#)

There is no installer for Linux; please follow the [manual installation guide](#).


 **Note:** The GNU Arm embedded toolchain (GCC) is bundled with the installers. If you want to compile using the Arm Compiler, visit the [supported compilers list](#) to see which versions you can use.

Figure 3: Mbed installer

2. From the link (section 2.1), install the dependencies as shown in Figure 4. (You are not required to do the installation if you already have it in the your device)

## 1. Install dependencies

### Instructions for Windows

1. Download and install [Python 3.7.x](#) (which includes `pip`).
2. Download and install [Git](#) (versions 1.9.5 or later are supported).
3. [Download and install Mercurial](#) (versions 2.2.2 or later are supported).

Figure 4: Dependencies installation

In this project, we are using Python 3.7.9 which can be installed from:

<https://www.python.org/downloads/release/python-379/>

We are choosing the Windows x86-64 executable installer for the Mercurial installation as shown in Figure 5. Please do select based on the device you are using.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		bcd9f22cf531efc6f06ca6b9b2919bd4	23277790	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		389d3ed26b4d97c741d9e5423da1f43b	17389636	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	4b544fc0ac8c3c94ea655211df9ade27	29305353	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		1094c8d9438ad1adc263ca57ceb3b927	8186795	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	60f77740b30030b22699dbd14883a4a3	7502379	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	7083fed513c3c9a4ea655211df9ade27	26940592	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	da0b17ae84d6579f8df3eb24927fd825	1348904	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		97c6558d479dc53bf448580b66ad7c1e	6659999	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		1e6d31c98c68c723541f0821b3c15d52	25875560	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		22f68f09e533c4940fc006e035f08aa2	1319904	<a href="#">SIG</a>

Figure 5: Mercurial installation

3. From the link (section 2.1), you need to install a compiler as shown in Figure 6. In this project we are using Keil as compiler, you are advised to choose the same compiler so that you can follow our guideline to setup easier.

### Compiler versions

You can build Mbed OS with the Arm Compiler and GNU Arm Embedded toolchains. The currently supported versions are:

Compiler	Download location	Name in Mbed CLI
Arm Compiler 6.15 (default ARM toolchain)	- A paid version is available as <a href="#">Arm Compiler 6.15 Professional</a> . - A paid version is also included in <a href="#">Keil MDK 5.33</a>	ARM
GNU Arm Embedded version 9 (9-2019-q4-major)	<a href="#">GNU Arm Embedded version 9 (9-2019-q4-major)</a>	GCC_ARM

Figure 6: Compiler installation

4. After the installation of compiler, open the command prompt and run the command. Note that you are required to use the compiler that you have installed on Step 3.

```
mbed config -G ARM_PATH "D:\Keil\Keil_v5\ARM"
```

This Figure 7 is showing the items that I have in the path of "D:\Keil\Keil\_v5\ARM".

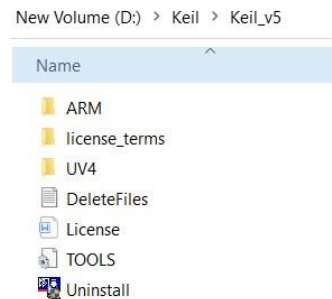


Figure 7: Content of folder path "D:\Keil\Keil\_v5\ARM".

5. Create a folder (we are having the folder name as **project**), then go to the folder (**project**) in the command prompt and run these commands:

```
mbed new mbed-os-program
```

```
cd mbed-os-program
```

```
mbed ls -a
```

*Note that I am locating the project folder in D:\mbed, please do change this part based on your folder location.*

```
python3 -m pip install -r D:\mbed\project\mbed-os-program\mbed-os\requirements.txt
```

6. Go to the folder "D:\mbed\project\mbed-os-program" in command prompt and run this command.

*Note that I am locating the project folder in D:\mbed, please do change this part based on your folder location.*

```
mbed config -G MBED_OS_DIR D:\mbed\project\mbed-os-program\mbed-os
```

7. Go to the folder "D:\mbed\project" in command prompt and create a folder to import project (Example: blinky1).

8. Go to the folder "D:\mbed\project\ blinky1" in command prompt. Go to the link, get the command to import this sample project.

[https://os.mbed.com/teams/GigaDevice/code/GD32\\_example\\_blinky//file/8b84a0b4f817/stats\\_report.h/](https://os.mbed.com/teams/GigaDevice/code/GD32_example_blinky//file/8b84a0b4f817/stats_report.h/)

Choose *import with Mbed CLI* as show in the Figure 8.



Figure 8: Importing Mbed OS project (Searching for Import with Mbed CLI)

Then you can copy the command provided (**mbed import** [http://os.mbed.com/teams/GigaDevice/code/GD32\\_example\\_blinky/](http://os.mbed.com/teams/GigaDevice/code/GD32_example_blinky/)) as shown in Figure 9 and execute the command in the directory of path "D:\mbed\project\ blinky1".

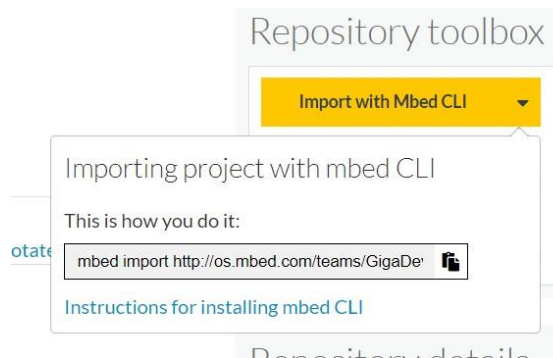


Figure 9: Get command to import Mbed OS project

The purpose of this is having this project as a base project and additional features can be added into this project.

9. Similarly as step 8, the command from the link is to add the **SerialStream.h** library. This library is added to enable the usage of serial printf. The details can be viewed in the coding later.

<https://os.mbed.com/users/MultipleMonomials/code/SerialStream/>

10. Similarly as step 8, the command from the link is to add the **arm\_math.h** and **arm\_const\_structs.h** library. This library is added so that we can use the CMSIS DSP.

[https://os.mbed.com/users/mbed\\_official/code/mbeddsp/docs/tip/arm\\_math\\_8h\\_source.html](https://os.mbed.com/users/mbed_official/code/mbeddsp/docs/tip/arm_math_8h_source.html)  
!

### 3.0 C++ codes

We are doing this project in the method where we are separating it into several small parts. The purpose of this is to ensure the functionality of each parts and simplify the debugging process. After the functionality of all basic parts are verified, we will integrate them together to achieve the objective of our project, frequency spectrum.

1. Ensure the ability of serial printf by viewing the result on the installed MobaXterm. The code is **Printf\_main.cpp** provided in the Codings folder of Github.
2. Ensure the ability of Nucleo 446RE board receive analog input by printing the value of analog data received in the installed MobaXterm. The code is **AnalogInput\_main.cpp** provided in the Codings folder of Github. *(Note : The connection to the hardware is important in this case)*
3. Ensure the ability of using CMSIS DSP. The code is **FFT\_main.cpp** provided in the Codings folder of Github.
4. Ensure the ability of providing data and controlling the 8x8LED Matrix Max Driver 7219. The code is **8x8LEDMatrix \_main.cpp** provided in the Codings folder of Github.  
*(Note: The connection to the hardware is important in this case)*
5. The final code for this frequency spectrum project is **main.cpp**, which is provided in the Codings folder of Github. *(Note : The connection to the hardware is important in this case)*

### 3.0 Install and set up MobaXterm

This project is using the MobaXterm Home Edition V21.1 (Portable Edition) for serial printf which can be installed from this link:

<https://mobaxterm.mobatek.net/download-home-edition.html>

There are several steps you need to do for serial printing on MobaXterm, they can be viewed from the Figure 10. For step 3, the serial port is only available when you connect your board to your device (personal computer or laptop). For step 4, the speed (bps) shall be based on the baud rate of the board you are using. (115200 is the baud rate of Nucleo F446RE)



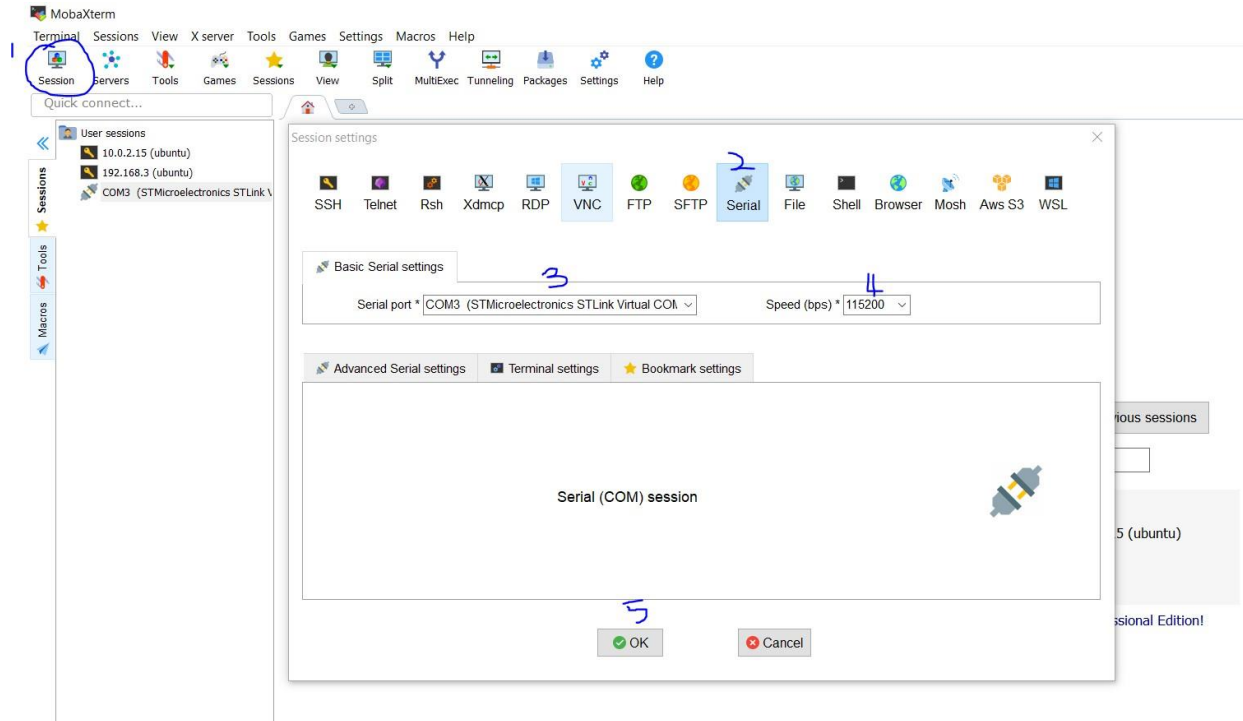


Figure 10: Steps to configure MobaXterm for serial printf

## 4.0 Compilation of project

After the required libraries are added, execute the command in the directory of path "D:\mbed\project\blinky1". This is to compile the code in main.cpp located in the path and flash it into the board (which is Nucleo F446RE in our project). Therefore, if you wish to start from the basic codes that we have provided, you can manually copy the content of the basic code into "D:\mbed\project\blinky1\main.cpp".

*Note that I am using Nucleo F446RE board in this project. Please do change this part in the command based on the board you are using. Note that it is not guaranteed where the provided will work perfectly in other board. Therefore, you are advised to use the Nucleo F446RE board as well.*

**mbed compile --target NUCLEO\_F446RE --toolchain GCC\_ARM --flash**