1. References – Documentation – Example Codes – Youtube Videos – Github Repos

a. PlatformIO - mbed official dsp library with FFT example code (in the examples section): https://registry.platformio.org/libraries/mbed-mbed-official/mbed-dsp

Website Link for the example code - Adafruit Project Code - Fun with Fourier Transforms:
http://learn.adafruit.com/fft-fun-with-fourier-transforms?view=all

b. CMSIS-DSP Documentation: https://arm-software.github.io/CMSIS_6/latest/DSP/index.html

Example code demonstrating how an FIR filter can be used as a low pass filter: https://arm-software.github.io/CMSIS-DSP/latest/arm_fir_example_f32_8c-example.html

Real FFT Documentation: https://arm-software.github.io/CMSIS-DSP/latest/group__RealFFTF32.html

Frequency Bin Example Code (use of Complex FFT, Complex Magnitude, and Maximum functions): https://arm-software.github.io/CMSIS-DSP/latest/group__FrequencyBin.html

Videos/Example Codes

g. STM32 Fast Fourier Transform (CMSIS DSP FFT) - Phil's Lab #111:
https://www.youtube.com/watch?v=d1KvgOwWvkM

h. Fast Fourier Transform using the ARM CMSIS Library within the STM32 MCUs:
https://www.youtube.com/watch?v=Z3_YsXXgWzw

i. ARM CMSIS DSP FFT Library – Eli Hughes:
https://www.youtube.com/watch?v=glh0pPsyGpI

j. STM32 DSP CMSIS: Real-Time FFT| Python script to plot spectrogram in real-time:
https://www.youtube.com/watch?v=iU9Jb06Fpp4

k. CMSIS DSP Library FIR Low Pass Filter example:
https://www.youtube.com/watch?v=JXE3GzNDM2c

l. STM32 DSP CMSIS library: FIR filter in one line (2022 new method):
https://www.youtube.com/watch?v=jqCpaleqfls

m. How to use FFT on STM32F103C8T6 step-by-step in 10 minutes - STM32 Tutorial #6:
https://www.youtube.com/watch?v=wMGamrfvEH8

n. FFT Spectrum Analysis - Audio DSP On STM32 (24 Bit / 48 kHz):
https://www.youtube.com/watch?v=3WF4CGKoMas&t=720s

o. FFT on Nucleo Board with Kemet Vibration Sensor:
https://www.youtube.com/watch?v=WFoQAf8ER3o
Github Repo: https://github.com/yosoufe/ExperineceKemet/tree/master

p. [#13] FIR Filters - Audio DSP On STM32 (24 Bit / 48 kHz):
https://www.youtube.com/watch?v=n9Cy1xkEf1E&list=PLTNEB0-EzPluXh0d_5zRprbgRfgkrYxfO&index=1

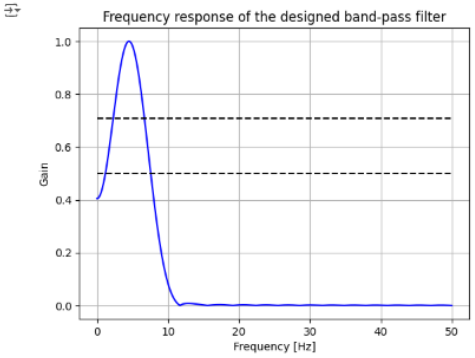
2. Sample Python Script to Compute Filter Coefficients (FIR Band Pass Filter Example):

```
# Filter specifications
sample_rate = 100  # Hz
nyquist_rate = sample_rate / 2
low_cutoff = 3  # Hz
high_cutoff = 6  # Hz
num_taps = 32  # Number of filter coefficients (taps)

# Design bandpass filter
coefficients = firwin(num_taps, [low_cutoff / nyquist_rate, high_cutoff / nyquist_rate], pass_zero=False)

# Frequency response of the filter
freq, response = freqz(coefficients, worN=8000)
plt.figure()
plt.plot(0.5 * sample_rate * freq / np.pi, np.abs(response), 'b')
plt.plot([0, 0.5 * sample_rate], [0.5, 0.5], 'k--')
plt.plot([0, 0.5 * sample_rate], [0.707, 0.707], 'k--')
plt.title('Frequency response of the designed band-pass filter')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Gain')
plt.grid()
plt.show()

# Print coefficients for use in embedded code
print("Filter coefficients:")
print(coefficients)
```




```
Filter coefficients:
[-0.00222951 -0.00465854 -0.0086943  -0.01488284 -0.02286169 -0.03121282
 -0.03763622 -0.03943306 -0.03419443 -0.02052307  0.00140974  0.02963956
  0.06060674  0.08975984  0.11244049  0.12484703  0.12484703  0.11244049
  0.08975984  0.06060674  0.02963956  0.00140974 -0.02052307 -0.03419443
 -0.03943306 -0.03763622 -0.03121282 -0.02286169 -0.01488284 -0.0086943
 -0.00465854 -0.00222951]
```

3. Embedded Challenge Tips *(Code Flow - if you are using FFT)*

Example/Sample Code Flow → [Collect Data from Sensor] →[**Data Acquisition Loop** Store the data in an array/buffer] → [Is it in the right format of Angular Rate?] → [Apply a filter here **if necessary**] → [Any manipulation needed to be done to the obtained data before FFT?] → [Apply FFT to the data blocks] → [Get the magnitudes and frequency values] → [Is it in the range to be a tremor?] →[Tremor Detected?! – Indicate through LED/Screen or anything else]

Notes:
- Make sure you have the right format of the output from the gyroscope. Refer the Page 10 of the Datasheet to know which Sensitivity Factor(Same as SCALING_FACTOR in recitation code) you have to consider. FS is dependent on CTRL_REG4 and ODR is dependent on CTRL_REG1 configuration. $Angular\ Rate = Raw\ Gyro\ Output * Sensitivity\ Factor$?

2.1 Mechanical characteristics

@ Vdd = 3.0 V, T = +25 °C, unless otherwise noted[a].

Table 4. Mechanical characteristics

Symbol	Parameter	Test condition	Min.[1]	Typ.[2]	Max.[1]	Unit
FS	Measurement range[3]	User-selectable		±245		dps
				±500		
				±2000		
So	Sensitivity[4]	FS = 245 dps	7.4	8.75	10.1	mdps/digit
		FS = 500 dps	14.8	17.50	19.8	
		FS = 2000 dps	59.2	70	79.3	
SoDr	Sensitivity change vs. temperature	From -40°C to +85°C		±2		%
DVoff	Digital zero-rate level[4]	FS = 245 dps	-25	±10	+25	dps
		FS = 500 dps	-37.5	±15	+37.5	
		FS = 2000 dps	-187.5	±75	+187.5	
OffDr	Zero-rate level change vs. temperature	FS = 245 dps		±0.03		dps/°C
		FS = 2000 dps		±0.04		
NL	Non linearity[3]	Best fit straight line	-5	0.2	+5	% FS
DST	Self-test output change	FS = 245 dps		130		dps
		FS = 500 dps		200		
		FS = 2000 dps		530		
Rn	Rate noise density	BW = 50 Hz		0.03		dps/ sqrt(Hz)
ODR	Digital output data rate			105/208/ 420/840		Hz
Top	Operating temperature range		-40		+85	°C

7.10 OUT_X_L (28h), OUT_X_H (29h)

X-axis angular rate data. The value is expressed as 2's complement.

7.11 OUT_Y_L (2Ah), OUT_Y_H (2Bh)

Y-axis angular rate data. The value is expressed as 2's complement.

7.12 OUT_Z_L (2Ch), OUT_Z_H (2Dh)

Z-axis angular rate data. The value is expressed as 2's complement.

7.5 CTRL_REG4 (23h)

Table 30. CTRL_REG4 register

0	BLE	FS1	FS0	-	ST1	ST0	SIM

Table 31. CTRL_REG4 description

BLE	Big/little endian data selection. Default value 0. (0: data LSB @ lower address; 1: data MSB @ lower address)
FS1 - FS0	Full scale selection. Default value: 00 (00: 245 dps; 01: 500 dps; 10: 2000 dps; 11: 2000 dps)
ST1-ST0	Self-test enable. Default value: 00 (00: self-test disabled; Other: see Table)
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface; 1: 3-wire interface).

Table 22. DR and BW configuration setting

DR [1:0]	BW [1:0]	ODR [Hz]	Cutoff
00	00	100	12.5
00	01	100	25
00	10	100	25
00	11	100	25

Table 22. DR and BW configuration setting (continued)

DR [1:0]	BW [1:0]	ODR [Hz]	Cutoff
01	00	200	12.5
01	01	200	25
01	10	200	50
01	11	200	70
10	00	400	20
10	01	400	25
10	10	400	50
10	11	400	110
11	00	800	30
11	01	800	35
11	10	800	50
11	11	800	110

7.2 **CTRL_REG1 (20h)**

Table 20. CTRL_REG1 register

DR1	DR0	BW1	BW0	PD	Zen	Yen	Xen

Table 21. CTRL_REG1 description

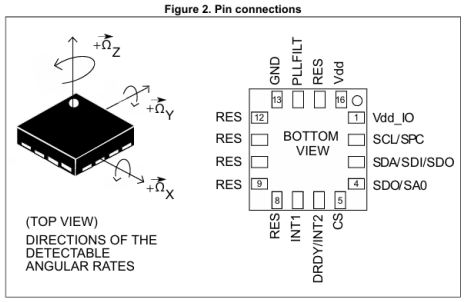
DR1-DR0	Output data rate selection. Refer to Table 22
BW1-BW0	Bandwidth selection. Refer to Table 22
PD	Power-down mode enable. Default value: 0 (0: power-down mode, 1: normal mode or sleep mode)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled; 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled; 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled; 1: X-axis enabled)

DR[1:0] is used to set ODR selection. **BW [1:0]** is used to set bandwidth selection.

In the following table (Table 22) all frequencies resulting in a combination of DR / BW bits are given.

1.1 **Pin description**

Figure 2. Pin connections



- Decide on which filters you want to use

    - Noise Reduction → Low Pass Filter
    - Frequency Range Detection → Band Pass Filter
    - Only use filters if **absolutely necessary. For example, try using the gyro output directly for further processing since it already passes through an inbuilt low pass filter with cutoffs mentioned in the tables from the datasheet.**
    - If using filters, try to get the correct coefficients for them. Try using scripts like the python script shown above.

- The FFT library functions generally suit block processing. (Operating on input/output buffers or arrays)
- Real FFT Library Function:
    - Example FFT Instance and Initialization → The FFT size typically must be a power of 2 (64, 128, 256, 512, etc..)

```
const int sampleSize = 128; // FFT size
float gyroData[sampleSize];
```

```
arm_rfft_fast_instance_f32 S;
arm_rfft_fast_init_f32(&S, sampleSize);
```

- FFT Computation → (Try both possible sizes for output, sampleSize and 2*sampleSize)

```
float output[2 * sampleSize]; // Output array must accommodate complex results
arm_rfft_fast_f32(&S, gyroData, output, 0); // '0' indicates forward FFT
```

- Magnitude Computation → (Try both possible sizes for magnitude, sampleSize/2 + 1 and sampleSize + 1)

```
// Array to hold the magnitude of the FFT output
float magnitude[sampleSize / 2 + 1];  // +1 for the DC component
```

```
// Compute the magnitude of the FFT output
arm_cmplx_mag_f32(output, magnitude, sampleSize / 2 + 1);
```

Structure of the FFT Output Array [Make sure the size is sampleSize or 2*sampleSize by trying both in your code and seeing the results!]

Array Size Issue:

Example → For N = 128 samples

- Gyro Data (time-domain input): 128
- FFT Output (frequency-domain output, requiring storage for complex numbers): Try 128 & 256
- FFT Magnitude (magnitude of the frequency components): 65 (Try 129 as well)