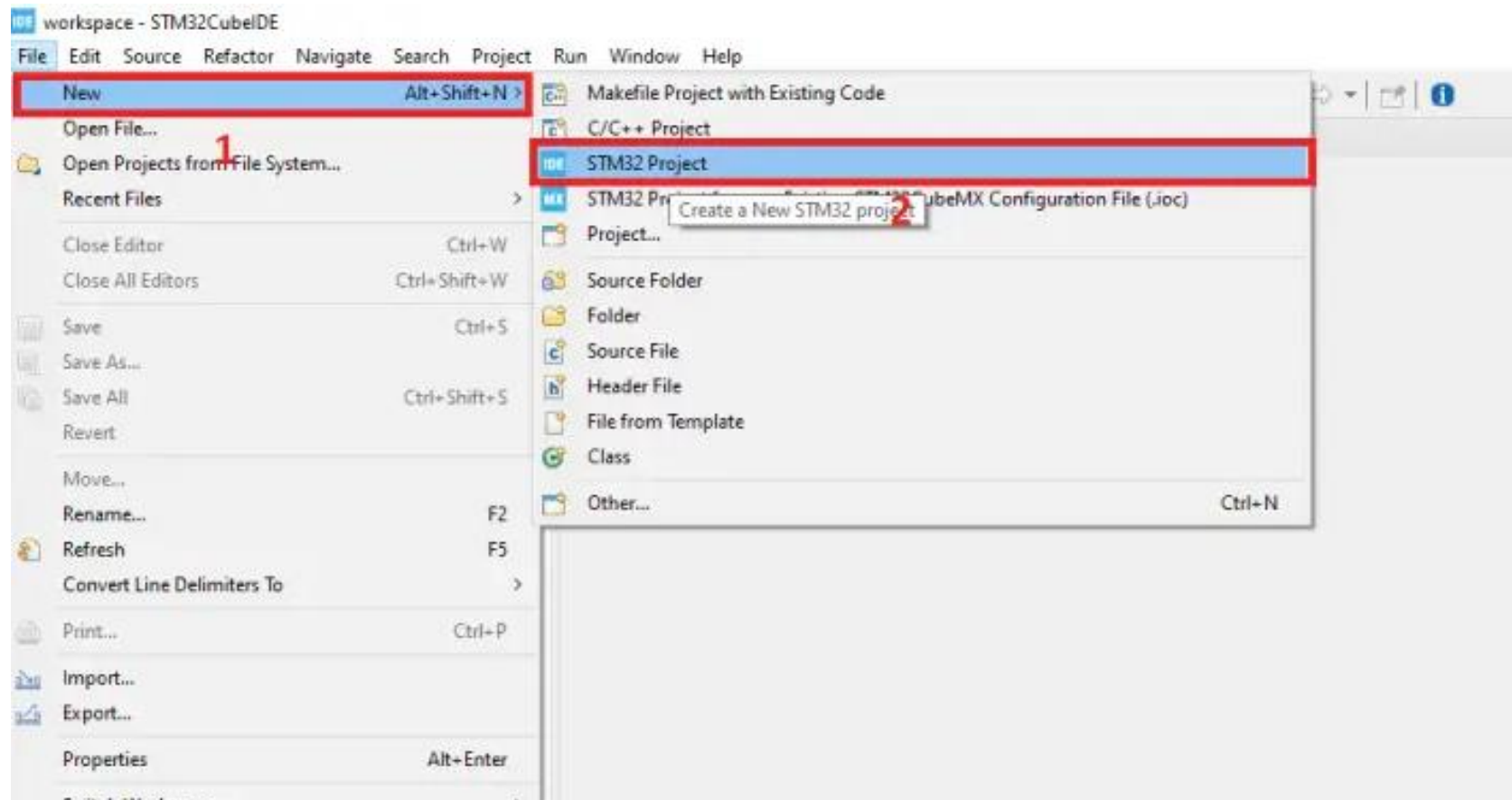
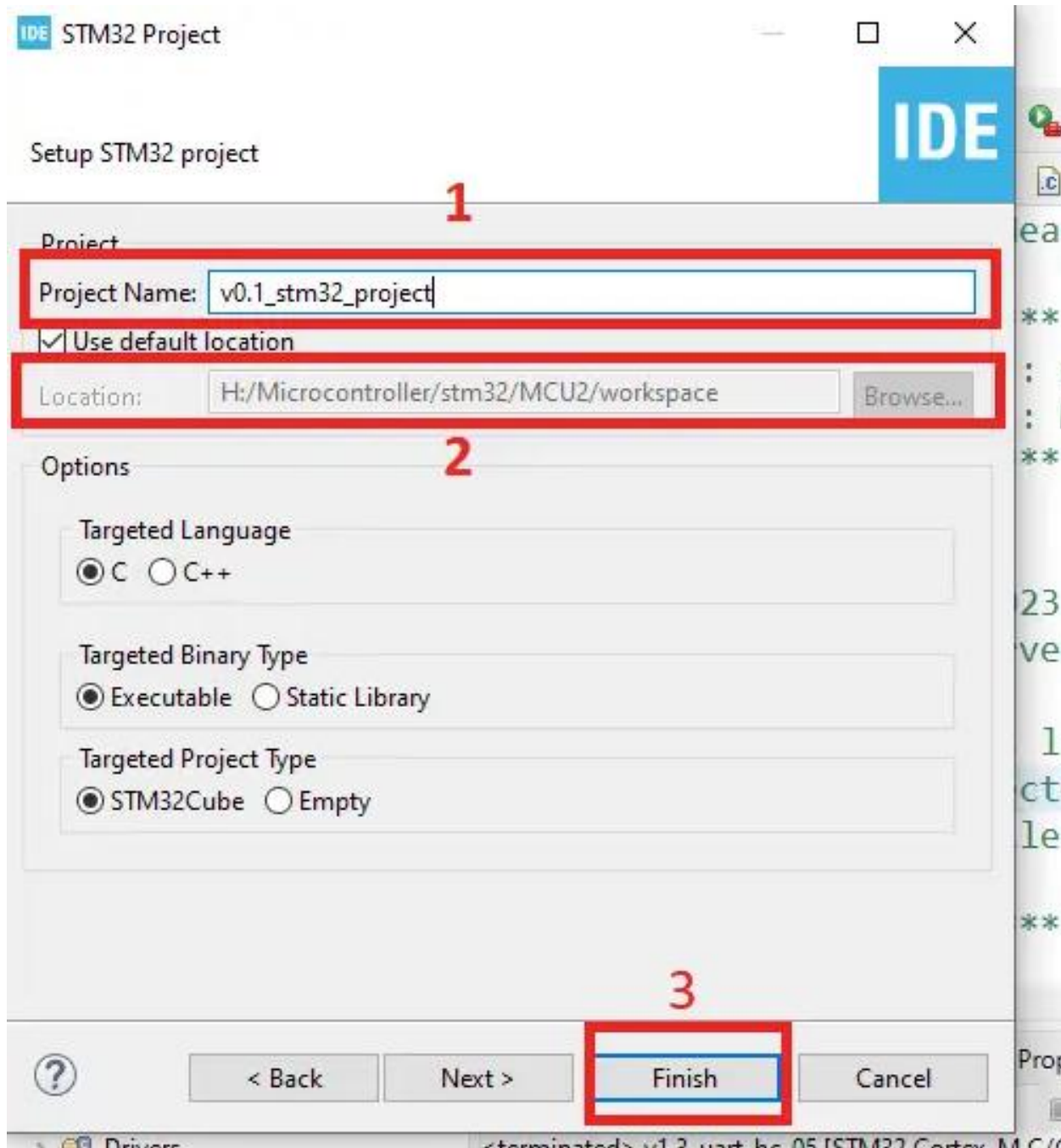


Project creation in Stm32Cube IDE



Select STWINBX1 board



Open MX and clear all Pinouts

SETTING UP LEDS

Find the pins and set them up as GPIO Output

PH10	n/a	n/a	n/a	Low	Output Push Pull	No pull-up and no p...	Medium	n/a	LED2[ORANGE]	<input checked="" type="checkbox"/>
PH12	n/a	n/a	n/a	Low	Output Push Pull	No pull-up and no p...	Medium	n/a	LED1[GREEN]	<input checked="" type="checkbox"/>

Set up FreeRTOS

Download the middleware X-CUBE-FREERTOS

▼ STMicroelectronics.X-CUBE-FREERTOS	✓	1.3.1	
▼ CMSIS RTOS2	✓	10.6.2	
▼ RTOS2	✓		
Core	✓	10.6.2	TZ_Non_Supported ▼
Heap	✓	10.6.2	Heap_4 ▼

Then

Pinout & Configuration

Clock Configuration

▼ Software Packs

▼ Pinout

Categories A-Z

Analog >

Timers >

Connectivity >

Multimedia >

Computing >

Middleware ▼

1

2

3

4

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V2

Configuration

Reset Configuration

Events FreeRTOS Heap Usage

Timers and Semaphores Mutexes

User Constants Tasks and Queues

Config parameters Include parameters Advanced settings

1 Configure the below parameters :

Search (Ctrl+F)

ENABLE_MPU Disabled

ENABLE_FPU Disabled

Kernel settings

USE_PREEMPTION Enabled

CPU_CLOCK_HZ SystemCoreClock

TICK_RATE_HZ 1000

MAX_PRIORITIES 56

Create tasks

☒ CMSIS RTOS2

Enable RTOS

Configuration

Reset Configuration

☒ Config parameters

☒ Include parameters

☒ Advanced settings

☒ User Constants

☒ CMSIS RTOS2

Tasks and Queues

Timers and Semaphores

Mutexes

Events

FreeRTOS Heap Usage

Tasks

Task name	Priority	Stack size (W...	Entry Func...	Code Generation ...	Parameter	Allocation	Buffer Na...	Control Block ...
LED1_task	osPriorit...	128	LED1_tas...	Default	NULL	Dynamic	NULL	NULL
LED2_task	osPriorit...	128	LED2_tas...	Default	NULL	Dynamic	NULL	NULL

Add

Delete

Queues

MX

Edit

X

Task name

LED2_task

Priority

osPriorityBelowNormal

▼

Stack size (Words)

128

▲▼

Entry Function

LED2_task_function

Code Generation Option

Default

▼

Parameter

NULL

Allocation

Dynamic

▼

Buffer Name

NULL

Control Block Name

NULL

OK

Cancel

Change timebase for SYS from SysTick to TIM3 (any available TIM)

SYS Mode and Configuration	
Mode	
Timebase Source	TIM3

Multitask toggle LED

```
void LED1_task_function(void *argument)
{
    /* USER CODE BEGIN LED1_task */
    /* Infinite loop */
    for(;;)
    {
        HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
        osDelay(500);
    }
    /* USER CODE END LED1_task */
}
```

Make sure both tasks toggle different LEDs

SET UP SPI

Set up the spi to mode 0

Mode Full-Duplex Master

Hardware NSS Signal Disable

Hardware RDY Signal Disable

Configuration

Reset Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

Clock Parameters

Prescaler (for Baud Rate)	32
Baud Rate	4.0 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

Autonomous Mode

State	Disable
-------	---------

CRC Parameters

CRC Calculation	Disabled
-----------------	----------

Advanced Parameters

NSSP Mode	Enabled
NSS Signal Type	Software
Fifo Threshold	Fifo Threshold 01 Data
Nss Polarity	Nss Polarity Low
Master Ss Idleness	00 Cycle
Master Inter Data Idleness	00 Cycle
Master Receiver Auto Susp	Disable
Master Keep Io State	Master Keep Io State Disable
IO Swap	Disabled
Ready Master Management	Internal
Ready Signal Polarity	High

Make sure that the SPI pins MATCHES THE IIS3DWB datasheet

Pin Name	Signal on Pin	Pin Context Assign...	Pin Privilege access	GPIO output level	GPIO mode	GPIO Pull-up/Pull-d...	Maximum output sp...	Fast Mode	User Label	Modified
PD3	SPI2_MISO	n/a	n/a	n/a	Alternate Function ...	No pull-up and no p...	Low	n/a		<input type="checkbox"/>
PI1	SPI2_SCK	n/a	n/a	n/a	Alternate Function ...	No pull-up and no p...	Low	n/a		<input type="checkbox"/>
PI3	SPI2_MOSI	n/a	n/a	n/a	Alternate Function ...	No pull-up and no p...	Low	n/a		<input type="checkbox"/>

Then set up the GPIO

Pin Name	Signal on Pin	Pin Context Assign...	Pin Privilege access	GPIO output level	GPIO mode	GPIO Pull-up/Pull-d...	Maximum output sp...	Fast Mode	User Label	Modified
PF12	n/a	n/a	n/a	Low	Output Push Pull	Pull-up	Medium	n/a	CS_DWB[IIS3DWB]	<input checked="" type="checkbox"/>
PF15	n/a	n/a	Non-privileged access	n/a	External Interrupt M...	No pull-up and no p...	n/a	n/a	INT1[IIS3DWB]	<input checked="" type="checkbox"/>
PH10	n/a	n/a	n/a	Low	Output Push Pull	No pull-up and no p...	Medium	n/a	LED2[ORANGE]	<input checked="" type="checkbox"/>
PH12	n/a	n/a	n/a	Low	Output Push Pull	No pull-up and no p...	Medium	n/a	LED1[GREEN]	<input checked="" type="checkbox"/>

And interrupt

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI Line15 interrupt	<input checked="" type="checkbox"/>	0	0

PLEASE MAKE SURE IT MATCHES THE ISS3DWB PINS IN DATASHEET

SETUP IIS3DWB

Download the middleware X-CUBE-MEMS1

▼ STMicroelectronics.X-CUBE-MEMS1	✓	11.2.0	▼
> <i>Exposed APIs</i>			
▼ <i>Device</i> MEMS1_Applications		11.2.0	
Application			Not selected ▼
> <i>Board Part</i> AccGyr		5.6.0	
> <i>Board Part</i> AccMag		5.7.0	
▼ <i>Board Part</i> Acc	✓	1.5.0	
LIS2DW12			Not selected ▼
LIS2DH12			Not selected ▼
IIS2DLPC			Not selected ▼
AIS2DW12			Not selected ▼
AIS328DQ			Not selected ▼
AIS3624DQ			Not selected ▼
H3LIS331DL			Not selected ▼
IIS2ICLX			Not selected ▼
AIS2IH			Not selected ▼
LIS2DU12			Not selected ▼
IIS3DWB	✓	1.3.0	SPI ▼
LIS2DUX12			Not selected ▼
ST1VAFE3BX			Not selected ▼

Go to the middleware and set up this

☒ Board Part Acc

Configuration

Reset Configuration

☒ Parameter Settings ☒ Platform Settings

Platform proposal

BSP

Name	IPs or Components		Found Solutions	BSP API
IIS3DWB_CS	GPIO:Output	<input type="checkbox"/>	PF12 [CS_DWB[IIS3DWB]]	Unknown
IIS3DWB BUS IO driver	SPI:Full-Duplex Master	<input type="checkbox"/>	SPI2	BSP_BUS_DRIVER

CHECK WHO_AM_I REGISTER TO SEE IF SENSOR ON

In a new file **vib_io.c**

Add the includes needed

```
#include "vib_io.h"
#include "iis3dwb_reg.h"
#include "steval_stwinbx1_bus.h"
#include "main.h" // for CS_DWB_GPIO_Port, CS_DWB_Pin

extern SPI_HandleTypeDef hspi2;
static stmdev_ctx_t dev_ctx;
```

SPI Write Wrapper

```
/* SPI write wrapper: assert CS, send reg & data, de-assert CS */
static int32_t drv_write(void *handle, uint8_t reg, uint8_t *bufp, uint16_t len) {
    HAL_GPIO_WritePin(CS_DWB_GPIO_Port, CS_DWB_Pin, GPIO_PIN_RESET);
    BSP_SPI2_Send(&reg, 1);
    BSP_SPI2_Send(bufp, len);
    HAL_GPIO_WritePin(CS_DWB_GPIO_Port, CS_DWB_Pin, GPIO_PIN_SET);
    return 0;
}
```

SPI Read Wrapper

```
/* SPI read wrapper: assert CS, send reg|0x80, read data, de-assert CS */
static int32_t drv_read(void *handle, uint8_t reg, uint8_t *bufp, uint16_t len) {
    reg |= 0x80;
    HAL_GPIO_WritePin(CS_DWB_GPIO_Port, CS_DWB_Pin, GPIO_PIN_RESET);
    BSP_SPI2_Send(&reg, 1);
    BSP_SPI2_Recv(bufp, len);
    HAL_GPIO_WritePin(CS_DWB_GPIO_Port, CS_DWB_Pin, GPIO_PIN_SET);
    return 0;
}
```

Initialize the IIS3DWB

```
/**
 * @brief Initialize the IIS3DWB vibration sensor and configure it for data acquisition.
 *
 * This function performs the following:
 * - Initializes the SPI interface for communication with the IIS3DWB sensor.
 * - Sets up the STM32 sensor driver context with board-specific read/write functions.
 * - Verifies the sensor is connected by reading the WHO_AM_I register.
 * - Resets the IIS3DWB sensor to default configuration.
 * - Configures the sensor for 3-axis measurement at 26.7 kHz, 2g full scale, with block data update.
 * - Sets the output data filter path to 6.3 kHz bandwidth.
 * - Enables the data-ready interrupt on INT1.
 *
 * @retval 0 Initialization and configuration successful.
 * @retval -1 SPI interface initialization failed.
 * @retval -2 IIS3DWB sensor not found (WHO_AM_I mismatch).
 */
```

```
int32_t vib_io_init(void)
{
    uint8_t whoamI, rst;

    /* 1) bring up the SPI bus */
    if (BSP_SPI2_Init() != BSP_ERROR_NONE) return -1;

    /* 2) hook our wrappers into the ST driver */
    dev_ctx.write_reg = drv_write;
    dev_ctx.read_reg = drv_read;
    dev_ctx.handle = &hspi2;
    dev_ctx.mdelay = HAL_Delay;

    /* 3) check device ID */
    iis3dwb_device_id_get(&dev_ctx, &whoamI);
}
```

```

if (whoamI != IIS3DWB_ID)                                return -2;          // WHO_AM_I_EXPECTED 0x7B

/* 4) reset sensor */
iis3dwb_reset_set(&dev_ctx, PROPERTY_ENABLE);
do {
    iis3dwb_reset_get(&dev_ctx, &rst);
} while (rst);

/* 5) basic configuration */
iis3dwb_block_data_update_set(&dev_ctx, PROPERTY_ENABLE);
iis3dwb_xl_data_rate_set (&dev_ctx, IIS3DWB_XL_ODR_26kHz);
iis3dwb_xl_full_scale_set(&dev_ctx, IIS3DWB_2g);
iis3dwb_xl_filt_path_on_out_set(&dev_ctx, IIS3DWB_LP_6kHz);

/* 6) enable data-ready interrupt on INT1 */
iis3dwb_int1_ctrl_t int1_ctrl;
iis3dwb_read_reg(&dev_ctx, IIS3DWB_INT1_CTRL, (uint8_t *)&int1_ctrl, 1);
int1_ctrl.int1_drdy_xl = 1;
iis3dwb_write_reg(&dev_ctx, IIS3DWB_INT1_CTRL, (uint8_t *)&int1_ctrl, 1);

return 0;
}

```

In a new file **vib_io.h**

Add the header

```

#ifndef VIB_IO_H
#define VIB_IO_H

#include <stdint.h>
#include "iis3dwb_reg.h"
#include "stm32u5xx_hal.h"

/* call once at startup. returns 0 on success */
int32_t vib_io_init(void);

#endif /* VIB_IO_H */

```

In **main.c**

Add check code in user code 2

```

/* USER CODE BEGIN 2 */
if(vib_io_init() != 0) HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin); // LED2[ORANGE] ON
else HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin); // LED1[GREEN] ON
/* USER CODE END 2 */

```

SETUP USART

Enable usart2 in connectivity, set up its mode to Asynchronous

✔ Parameter Settings

✔ User Constants

✔ NVIC Settings

✔ DMA Settings

✔ GPIO

Configure the below parameters :

Search (Ctrl+F)

- Basic Parameters
- Baud Rate

115200 Bits/s
- Word Length

8 Bits (including Parity)
- Parity

None
- Stop Bits

1
- Advanced Parameters
- Data Direction

Receive and Transmit
- Over Sampling

16 Samples
- Single Sample

Disable
- ClockPrescaler

1
- Fifo Mode

Disable
- Txfifo Threshold

1 eighth full configuration
- Rxfifo Threshold

1 eighth full configuration
- Autonomous Mode

Disable
- Advanced Features
- Auto Baudrate

Disable
- TX Pin Active Level Inversion

Disable
- RX Pin Active Level Inversion

Disable
- Data Inversion

Disable
- TX and RX Pins Swapping

Disable
- Overrun

Enable
- DMA on RX Error

Enable
- MSB First

Disable

Pin Name	Signal	Pin Cont...	Pin Privi...	GPIO o...	GPIO ...	GPIO P...	Maximu...	Fast M...	User Label	Modified
PA15 (JTDI)	USART...	n/a	n/a	n/a	Alternat...	No pull...	Low	n/a		<input type="checkbox"/>
PD5	USART...	n/a	n/a	n/a	Alternat...	No pull...	Low	n/a		<input type="checkbox"/>

Code example

```
static uint8_t tx_buffer[2000];
num = number;
snprintf((char *)tx_buffer, sizeof(tx_buffer), "Text %d \r\n", num);
HAL_UART_Transmit(&huart2, tx_buffer, strlen((char const *)tx_buffer), 1000);
```

SETUP FIFO

In init function, add initialization of fifo

```
iis3dwb_fifo_watermark_set(&dev_ctx, FIFO_WATERMARK); // 1. Set FIFO threshold (samples)
iis3dwb_fifo_mode_set(&dev_ctx, IIS3DWB_STREAM_MODE); // 2. Set FIFO mode to stop collecting data when FIFO is full
```

in interrupt handler

set a flag

```
if(GPIO_Pin == INT1_Pin){
    fifo_int_flag = 1; // Set the flag

} else {
    __NOP();
}
```

now to read the fifo,

you read the watermark flag, then read the entries and reset fifo

```
iis3dwb_fifo_status_get(&dev_ctx, &fifo_status);
```

```
/* read out all FIFO entries in a single read */
iis3dwb_fifo_out_multi_raw_get(&dev_ctx, fifo_data, num);
// Reset FIFO
iis3dwb_fifo_mode_set(&dev_ctx, IIS3DWB_BYPASS_MODE);
iis3dwb_fifo_mode_set(&dev_ctx, IIS3DWB_STREAM_MODE);
for (k = 0; k < num; k++) {
    iis3dwb_fifo_out_raw_t *f_data;

    /* print out first two and last two FIFO entries only */
    if (k > 1 && k < num - 2) continue;

    f_data = &fifo_data[k];

    /* Read FIFO sensor value */
```

```

datax = (int16_t *)&f_data->data[0];
datay = (int16_t *)&f_data->data[2];
dataz = (int16_t *)&f_data->data[4];
ts = (int32_t *)&f_data->data[0];

switch (f_data->tag >> 3) {
case IIS3DWB_XL_TAG:
    snprintf((char *)tx_buffer, sizeof(tx_buffer), "%d: ACC [mg]:\t%4.0f\t%4.0f\t%4.0f\r\n",
            k,
            iis3dwb_from_fs8g_to_mg(*datax),
            iis3dwb_from_fs8g_to_mg(*datay),
            iis3dwb_from_fs8g_to_mg(*dataz));
    HAL_UART_Transmit(&huart2, tx_buffer, strlen((char const *)tx_buffer), 1000);
    break;
default:
    break;
}
}

```