LAB#2 GUIDE

Rapid Wireless Sensing enabled by NXP’s latest MCX W23 wireless MCU and Low-Power Magnetic Wakeup Sensors

Hands-on Lab#2 Guide

Contents

Hands-On Lab#2 Overview:

Tamper detection using low-power magnetic wakeup sensor:
Securing Your Assets with MCXW23 Wireless MCU

Brief Description

This example uses **FRDM-MCXW23** and **NMH1000** magnetic switch sensor (using sensor expansion board) to demonstrate autonomous detection of tampering/theft/abuse on device using low-power magnetic wakeup feature and transmit ALERT message via BLE wireless UART.

Details

The core idea is to leverage advanced sensor technology to protect high-value or secure assets. For example, smart meters can monitor for tampering, enhancing home security by detecting unauthorized access to safes or lockers. Personal medical devices can be safeguarded against misuse, and laptops or tablets can be protected from theft. In industrial settings, warehouse theft detection and machine tampering alerts can significantly reduce losses and improve security. This technology can benefit even simple applications like door open/close detection.

Solution

This example demonstrates ease-of-enablement using NXP's FRDM-MCXW23 and sensors development ecosystem to accelerate prototyping for your multiple such applications.

The **Lab#2** solution comprises of

- **MCX W23** for transmitting tamper detection alerts via BLE.
- Hall effect magnetic switch sensor **NMH1000** for low-power magnetic wakeup.

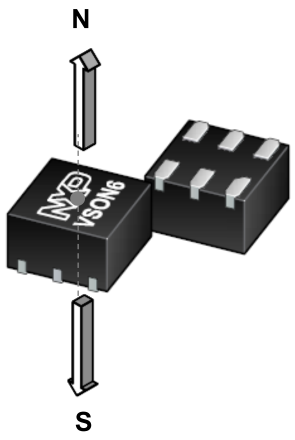
Recommended Sensor

NMH1000: Hall-Effect Magnetic Switch

The **NMH1000** is a **Hall effect magnetic field switch** with small physical size. The switch is most sensitive to **a vertical field passing through** the top-to-bottom surfaces of the package. The switch operates at low supply voltage with **ultra-low power consumption**. The switch detects the absence of a magnetic field and outputs either High or Low state upon the comparison with pre-set thresholds.

Features:

- Z- axis Hall effect switch for axial magnets
  North & South polarity sensitivity
- Low operating voltage:
  1.2 V min
- Ultra low current:
  75 nA typ. @ 1 Hz sample rate
- Industrial operating temperature range:
  -40 °C to 85 °C
- Selections by hardware pin / register:
  - ➢ Threshold
  - ➢ Sample Rate
  - ➢ Standalone Mode or I²C User Mode
- 2-state output via hardware pin / register



DFN 1.4 x 1.4 x 0.85 mm

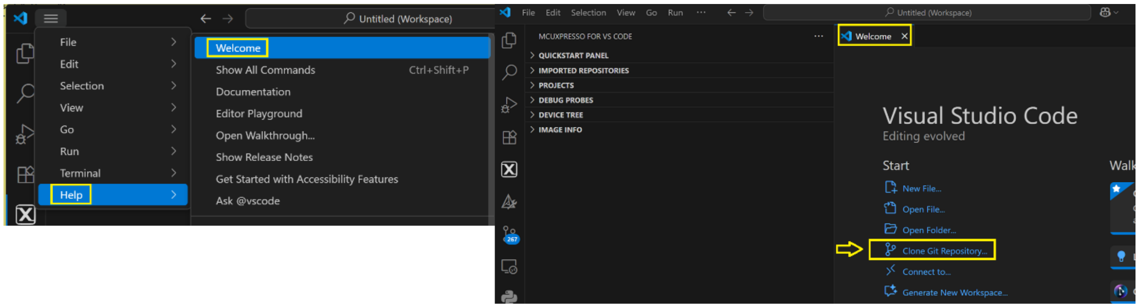
Target Applications

- Door or window open / closed detection
- Laptop lid open / closed detection
- IOT applications
- Electronic system wake-up
- Drug delivery device
- CGM application
- Insulin pump flow control
- Metering anti-tampering
- Reed switch replacement

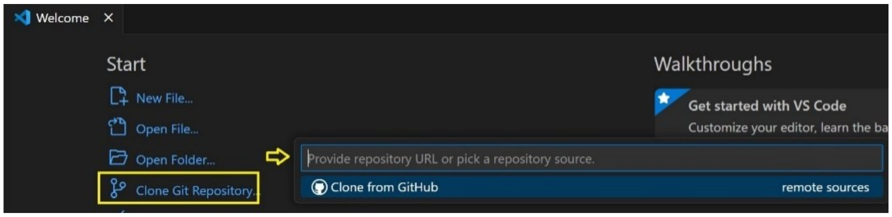
Get Hands-on Workshop Lab Examples from ACH

Note: If you have completed Lab#1, you can skip steps 1) to 6) and start from 7).

1) Open VS Code and select "Clone Git Repository" option on the “Welcome” page. Find the “Welcome” page under “Help” tab.

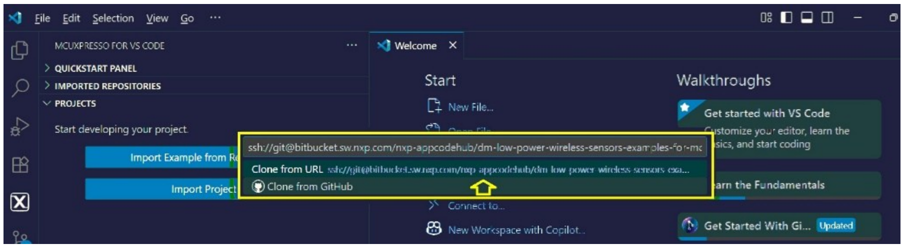


2) VS Code will open a pop-up to allow you to enter the Git repository URL.



Provide repository URL or pick a repository source.

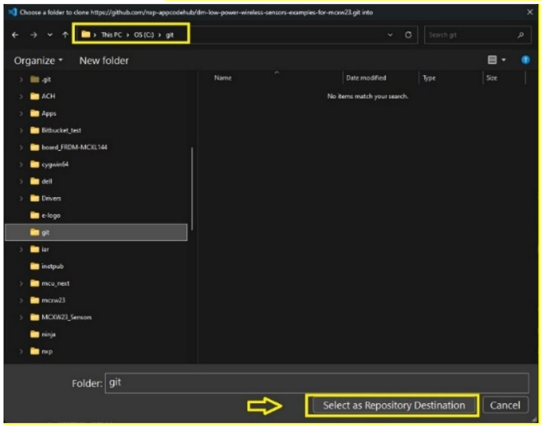
3) Copy the ACH gihub URL:"https://github.com/nxp-appcodehub/dm-low-power-wireless-sensors-examples-for-mcxw23.git" and select the the "Clone from URL" option.



ssh://git@bitbucket.sw.nxp.com/nxp-appcodehub/dm-low-power-wireless-sensors-examples-for-mc

Clone from URL ssh://git@bitbucket.sw.nxp.com/nxp-appcodehub/dm-low-power-wireless-sensors-exa...

4) It will ask for local destination directory where you would like to save the cloned repository, <u>create a "git" folder under C: drive and select that folder</u>.

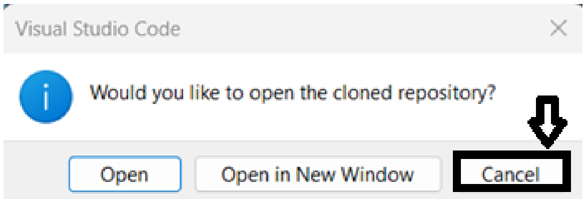


Choose a folder to clone https://github.com/nxp-appcodehub/dm-low-power-wireless-sensors-examples-for-mcxw23.git into

No items match your search.

Folder: git

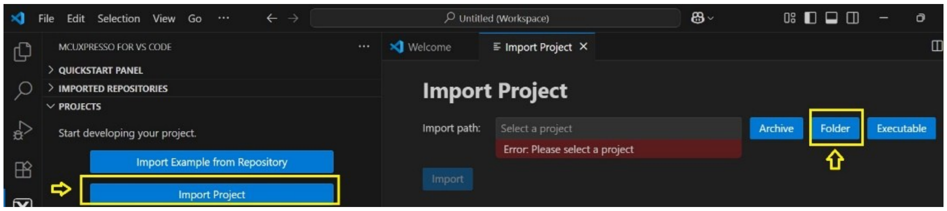
5) VS Code will ask whether you would like to open the cloned repository. Click on Cancel.



Would you like to open the cloned repository?

6) VS Code will clone the repository into the destination folder i.e. C:\git directory.

7) Click on "Import Project(s)" to start importing the chosen ACH project(s).



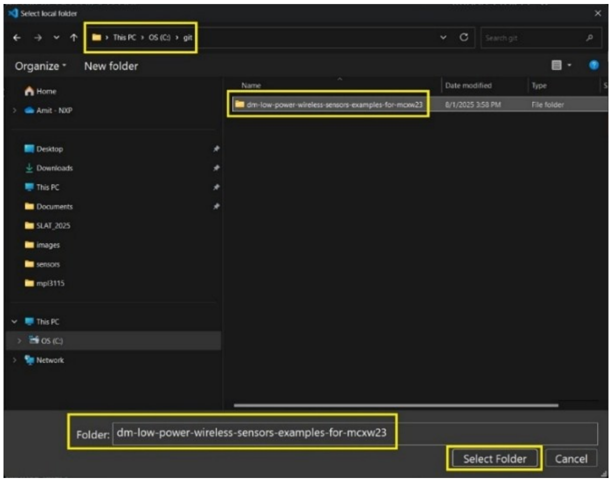
Import Project

Start developing your project.

Import path: Select a project

Error: Please select a project

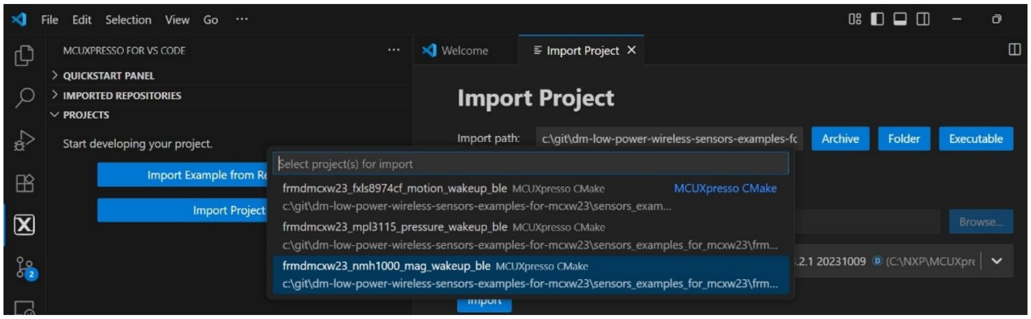
8) Select the cloned repository destination folder i.e. "C:/git/dm-low-power-wireless-sensors-examples-for-mcxw23".



Name	Date modified	Type
dm-low-power-wireless-sensors-examples-for-mcxw23	8/1/2025 3:58 PM	File folder

Folder: dm-low-power-wireless-sensors-examples-for-mcxw23

9) Import Lab#2 "frmdmcxw23_nmh1000_mag_wakeup_ble" project into VS code workspace:



Import Project

Start developing your project.

Import path: c:\git\dm-low-power-wireless-sensors-examples-fc

Select project(s) for import

frmdmcxw23_fxls8974cf_motion_wakeup_ble MCUXpresso CMake MCUXpresso CMake
c:\git\dm-low-power-wireless-sensors-examples-for-mcxw23\sensors_exam...

frmdmcxw23_mpl3115_pressure_wakeup_ble MCUXpresso CMake
c:\git\dm-low-power-wireless-sensors-examples-for-mcxw23\sensors_examples_for_mcxw23\frm...

frmdmcxw23_nmh1000_mag_wakeup_ble MCUXpresso CMake
c:\git\dm-low-power-wireless-sensors-examples-for-mcxw23\sensors_examples_for_mcxw23\frm...

.2.1 20231009 (C:\NXP\MCUXpre

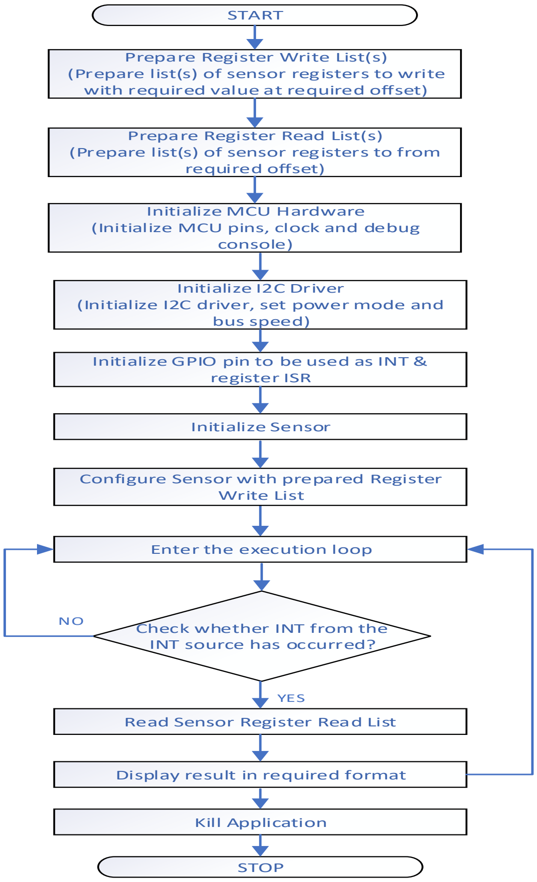
10) Select the toolchain version: ARM GNU Toolchain.

11) The selected "frmdmcxw23_nmh1000_mag_wakeup_ble " project is now imported on the VS Code workspace.

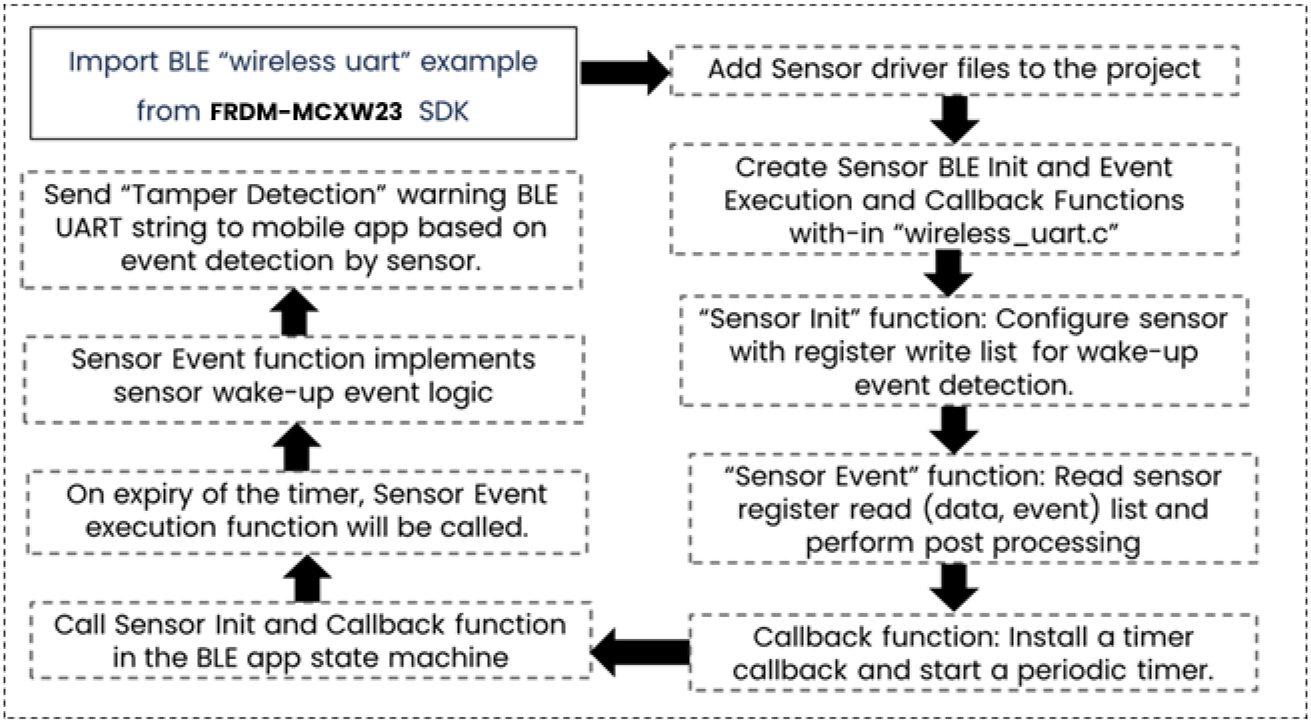
Code Walkthrough

Typical ISSDK example code flow:

1) Prepare register write list.
2) Prepare register read list.
3) Initialize MCU Hardware (Pins, Clock etc.)
4) Initialize COMM (I2C, SPI) driver
5) Initialize GPIO pin to be user as INT and register ISR.
6) Initialize Sensor, read WHOAMI
7) Configure Sensor with prepared register write list
8) Enter execution loop:
   i. Check whether interrupt source has occurred.
   ii. If yes, then read register read list
   iii. Display result or upload into TSA table
   iv. Execute post processing or other functions.


START
Prepare Register Write List(s)
(Prepare list(s) of sensor registers to write with required value at required offset)
Prepare Register Read List(s)
(Prepare list(s) of sensor registers to from required offset)
Initialize MCU Hardware
(Initialize MCU pins, clock and debug console)
Initialize I2C Driver
(Initialize I2C driver, set power mode and bus speed)
Initialize GPIO pin to be used as INT & register ISR
Initialize Sensor
Configure Sensor with prepared Register Write List
Enter the execution loop
NO
Check whether INT from the INT source has occurred?
YES
Read Sensor Register Read List
Display result in required format
Kill Application
STOP

Learn how to convert SDK wireless example to wireless sensing example:



Import BLE "wireless uart" example from **FRDM-MCXW23** SDK

Add Sensor driver files to the project

Create Sensor BLE Init and Event Execution and Callback Functions with-in "wireless_uart.c"

"Sensor Init" function: Configure sensor with register write list for wake-up event detection.

"Sensor Event" function: Read sensor register read (data, event) list and perform post processing

Callback function: Install a timer callback and start a periodic timer.

Call Sensor Init and Callback function in the BLE app state machine

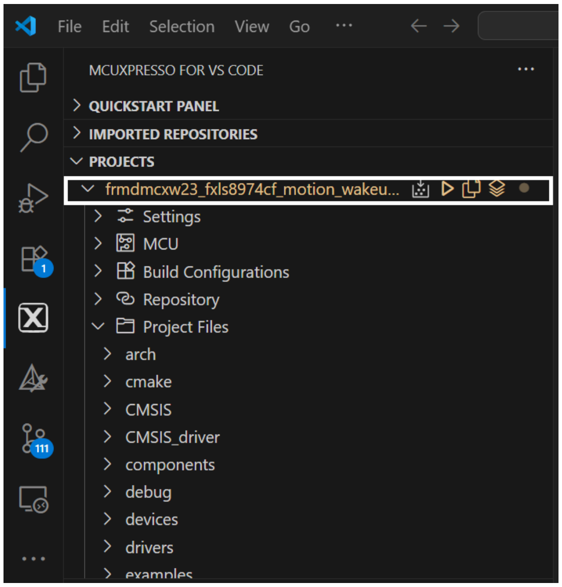
On expiry of the timer, Sensor Event execution function will be called.

Sensor Event function implements sensor wake-up event logic

Send "Tamper Detection" warning BLE UART string to mobile app based on event detection by sensor.

Compile Project:

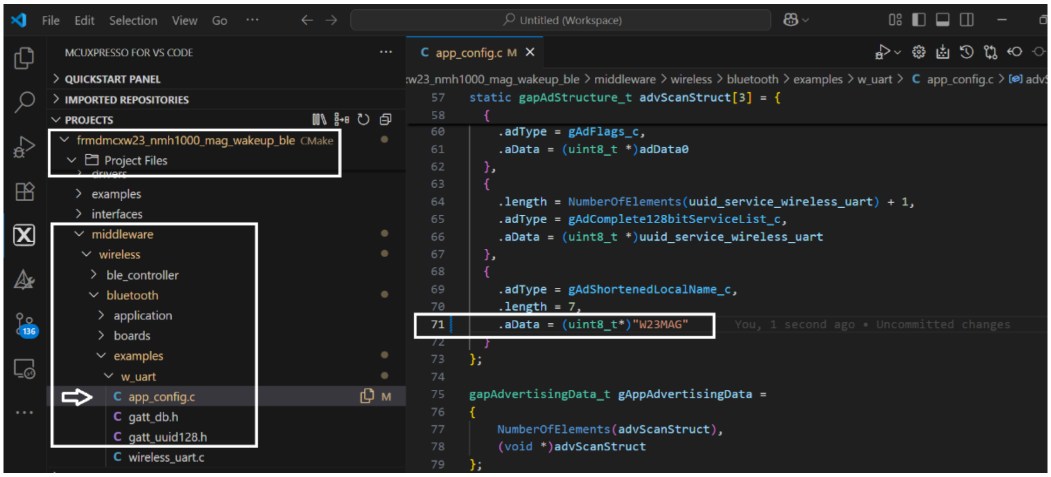
1) Chose "frmdmcxw23_nmh1000_mag_wakeup_ble" project" on your VS Code workspace:



```
⌄ frmdmcxw23_fxls8974cf_motion_wakeu...
  > Settings
  > MCU
  > Build Configurations
  > Repository
  ⌄ Project Files
    > arch
    > cmake
    > CMSIS
    > CMSIS_driver
    > components
    > debug
    > devices
    > drivers
    > examples
```

2) Before compiling the project, go to Explorer view and open app_config.c file (available under "Project Files->middleware->wireless->Bluetooth->examples->w_uart" folder).

3) On line 71, make sure to update "aData" field with a unique name (this will help to identify your board so that it doesn't clash with any other nearby board). Let me change to "NXP_A0" (try using NXPA01 onwards) but you can select any name you want (max length is 6 characters).



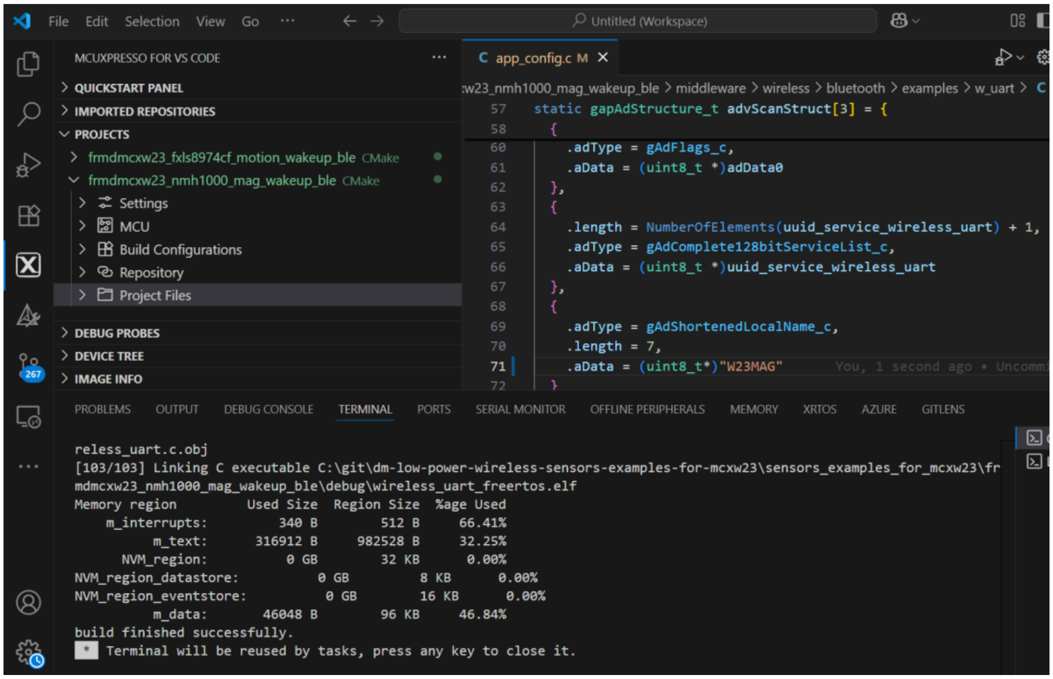
```
⌄ frmdmcxw23_nmh1000_mag_wakeup_ble CMake
  ⌄ Project Files
    > drivers
    > examples
    > interfaces
    ⌄ middleware
      ⌄ wireless
        > ble_controller
        ⌄ bluetooth
          > application
          > boards
          ⌄ examples
            ⌄ w_uart
              C app_config.c   M
              C gatt_db.h
              C gatt_uuid128.h
              C wireless_uart.c
```

:w23_nmh1000_mag_wakeup_ble > middleware > wireless > bluetooth > examples > w_uart > C app_config.c > adv

```
57  static gapAdStructure_t advScanStruct[3] = {
58    {
60      .adType = gAdFlags_c,
61      .aData = (uint8_t *)adData0
62    },
63    {
64      .length = NumberOfElements(uuid_service_wireless_uart) + 1,
65      .adType = gAdComplete128bitServiceList_c,
66      .aData = (uint8_t *)uuid_service_wireless_uart
67    },
68    {
69      .adType = gAdShortenedLocalName_c,
70      .length = 7,
71      .aData = (uint8_t*)"W23MAG"        You, 1 second ago • Uncommitted changes
72    }
73  };
74
75  gapAdvertisingData_t gAppAdvertisingData =
76  {
77      NumberOfElements(advScanStruct),
78      (void *)advScanStruct
79  };
```

4) Right click on chosen project and select "Pristine Build/Rebuild Project" to start clean build the project.

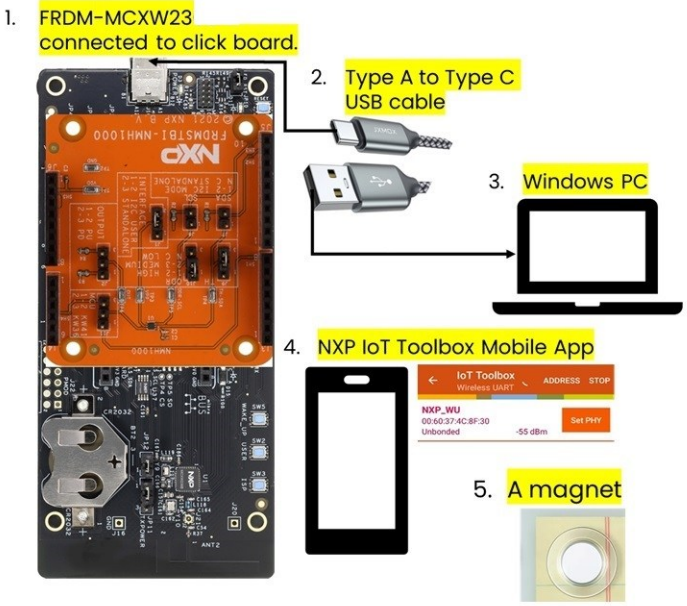
5) Confirm successful compilation of “frmdmcxw23_nmh1000_mag_wakeup_ble” project.



```
static gapAdStructure_t advScanStruct[3] = {
  {
    .adType = gAdFlags_c,
    .aData = (uint8_t *)adData0
  },
  {
    .length = NumberOfElements(uuid_service_wireless_uart) + 1,
    .adType = gAdComplete128bitServiceList_c,
    .aData = (uint8_t *)uuid_service_wireless_uart
  },
  {
    .adType = gAdShortenedLocalName_c,
    .length = 7,
    .aData = (uint8_t*)"W23MAG"        You, 1 second ago • Uncomm
  }
```

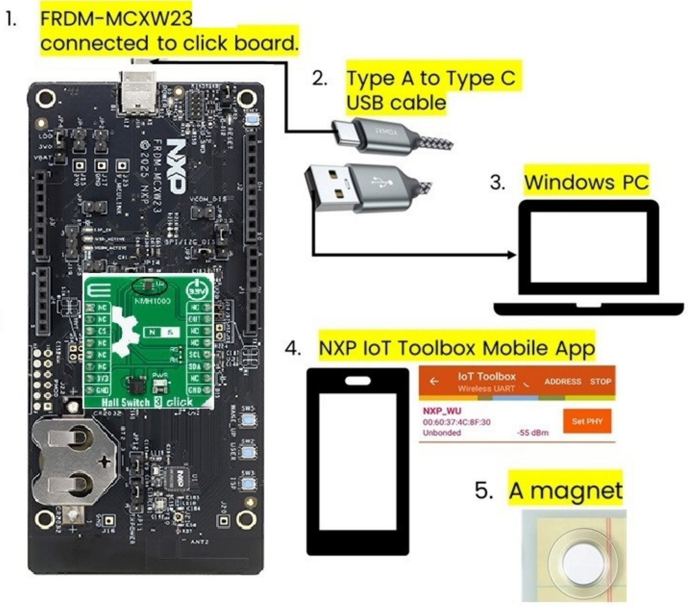
```
reless_uart.c.obj
[103/103] Linking C executable C:\git\dm-low-power-wireless-sensors-examples-for-mcxw23\sensors_examples_for_mcxw23\fr
mdmcxw23_nmh1000_mag_wakeup_ble\debug\wireless_uart_freertos.elf
Memory region         Used Size  Region Size  %age Used
    m_interrupts:         340 B        512 B     66.41%
          m_text:      316912 B     982528 B     32.25%
      NVM_region:          0 GB        32 KB      0.00%
NVM_region_datastore:          0 GB         8 KB      0.00%
NVM_region_eventstore:          0 GB        16 KB      0.00%
          m_data:       46048 B        96 KB     46.84%
build finished successfully.
 *  Terminal will be reused by tasks, press any key to close it.
```

Connect HW:



Hardware Setup with FRDMSTBI-NMH1000 shield board

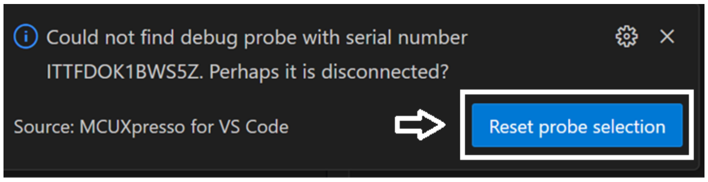
OR



Hardware Setup with Hall Switch 3 Click

Program the target Board:

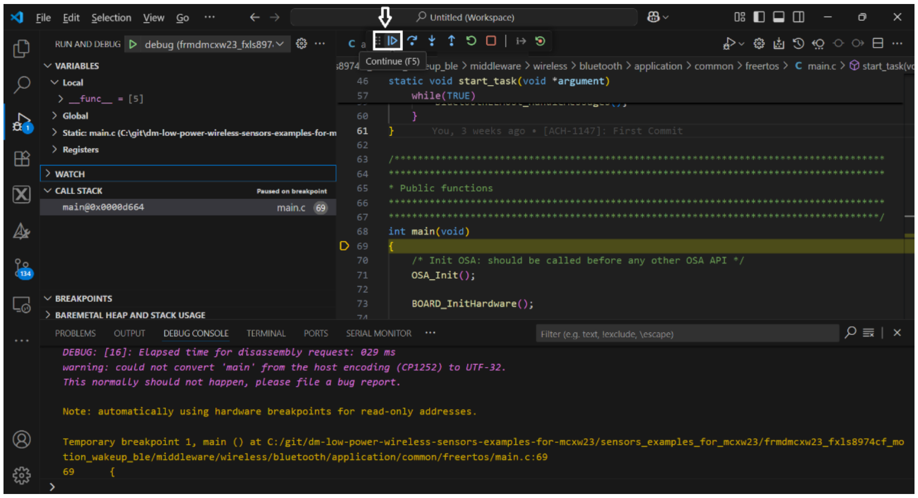
1) Right click on the "frmdmcxw23_nmh1000_mag_wakeup_ble" project and select "Debug".
2) In case VS code ask to reset debug probe, then click on "Reset probe selection" to proceed.



Could not find debug probe with serial number ITTFDOK1BWS5Z. Perhaps it is disconnected?

Source: MCUXpresso for VS Code

3) Debugger will start downloading the program to device. Once downloaded, you will see program curser in the main () function.



__func__ = [5]

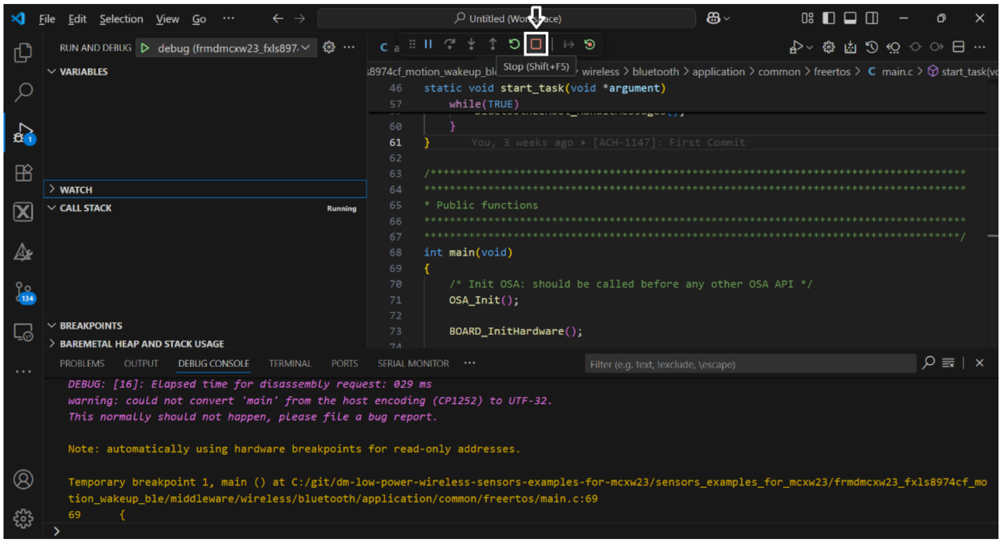
main@0x0000d664 main.c 69

s8974 ... up_ble > middleware > wireless > bluetooth > application > common > freertos > main.c > start_task(vo

```
static void start_task(void *argument)
    while(TRUE)
    }
}        You, 3 weeks ago • [ACH-1147]: First Commit

/*******************************************************************************
 *******************************************************************************
 * Public functions
 *******************************************************************************
 ******************************************************************************/
int main(void)
{
    /* Init OSA: should be called before any other OSA API */
    OSA_Init();

    BOARD_InitHardware();
```

```
DEBUG: [16]: Elapsed time for disassembly request: 029 ms
warning: could not convert 'main' from the host encoding (CP1252) to UTF-32.
This normally should not happen, please file a bug report.

Note: automatically using hardware breakpoints for read-only addresses.

Temporary breakpoint 1, main () at C:/git/dm-low-power-wireless-sensors-examples-for-mcxw23/sensors_examples_for_mcxw23/frmdmcxw23_fxls8974cf_mo
tion_wakeup_ble/middleware/wireless/bluetooth/application/common/freertos/main.c:69
69      {
```

4) Click on "Continue" button or press "F5" from your keyboard to resume running the downloaded program on device.
5) Now the program is downloaded on the device. you should see "white/bluish" LED blinking.
6) Click on "Terminate" button or press "Shift + F5" to terminate the debug session.



s8974cf_motion_wakeup_bl ... > wireless > bluetooth > application > common > freertos > main.c > start_task(vo

```
static void start_task(void *argument)
    while(TRUE)
    }
}        You, 3 weeks ago • [ACH-1147]: First Commit

/*******************************************************************************
 *******************************************************************************
 * Public functions
 *******************************************************************************
 ******************************************************************************/
int main(void)
{
    /* Init OSA: should be called before any other OSA API */
    OSA_Init();

    BOARD_InitHardware();
```

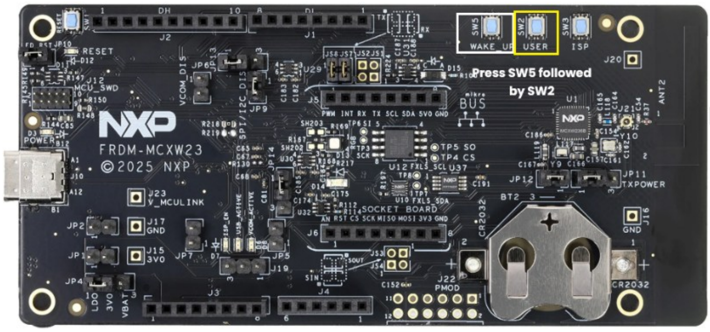
```
DEBUG: [16]: Elapsed time for disassembly request: 029 ms
warning: could not convert 'main' from the host encoding (CP1252) to UTF-32.
This normally should not happen, please file a bug report.

Note: automatically using hardware breakpoints for read-only addresses.

Temporary breakpoint 1, main () at C:/git/dm-low-power-wireless-sensors-examples-for-mcxw23/sensors_examples_for_mcxw23/frmdmcxw23_fxls8974cf_mo
tion_wakeup_ble/middleware/wireless/bluetooth/application/common/freertos/main.c:69
69      {
```

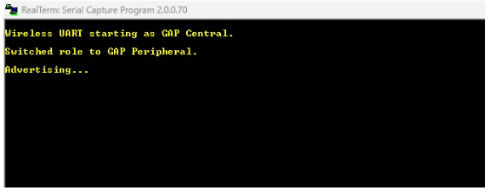
7) Disconnect the USB-C cable connected to FRDM-MCXW23 and reconnect. You should see "white/bluish" LED blinking

Start Advertising:

1) After that you should see white/bluish LED blinking and when you open a serial terminal (with baud rate 115200). You will see W UART starting as GAP central.

2) On the FRDM-MCXW23 board, press SW5 followed by SW2, then you will see the profile shift to GAP Peripheral and now advertising.


Press SW5 followed by SW2



```
Wireless UART starting as GAP Central.
Switched role to GAP Peripheral.
Advertising...
```

Run NXP IoT Toolbox App on your mobile:

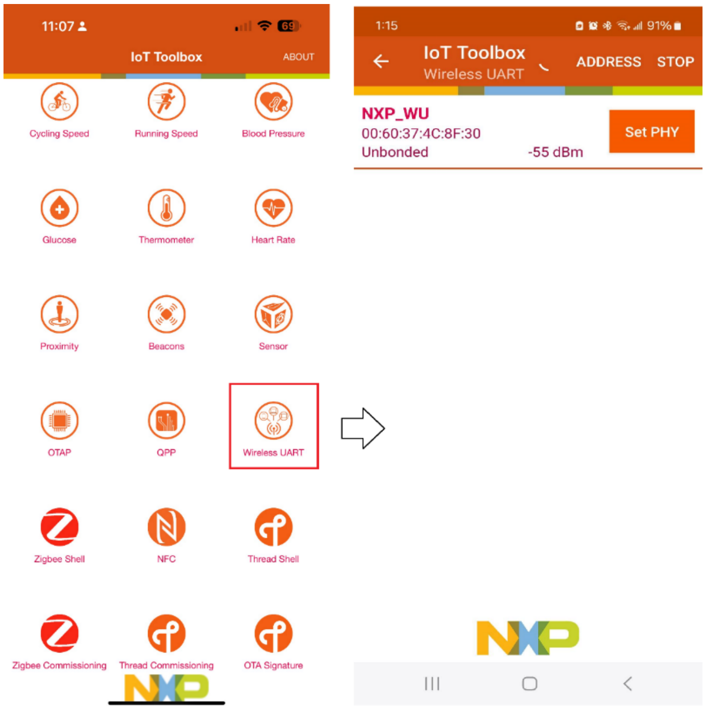
1) Open the installed NXP IoT Toolbox (available from Play Store and App Store) on your cell phone.

2) **Make sure on your mobile phone following setting is set:**

   Bluetooth connection is ON, and the location is ON.

3) Inside the application, click on the Wireless UART widget. Scan should start automatically.

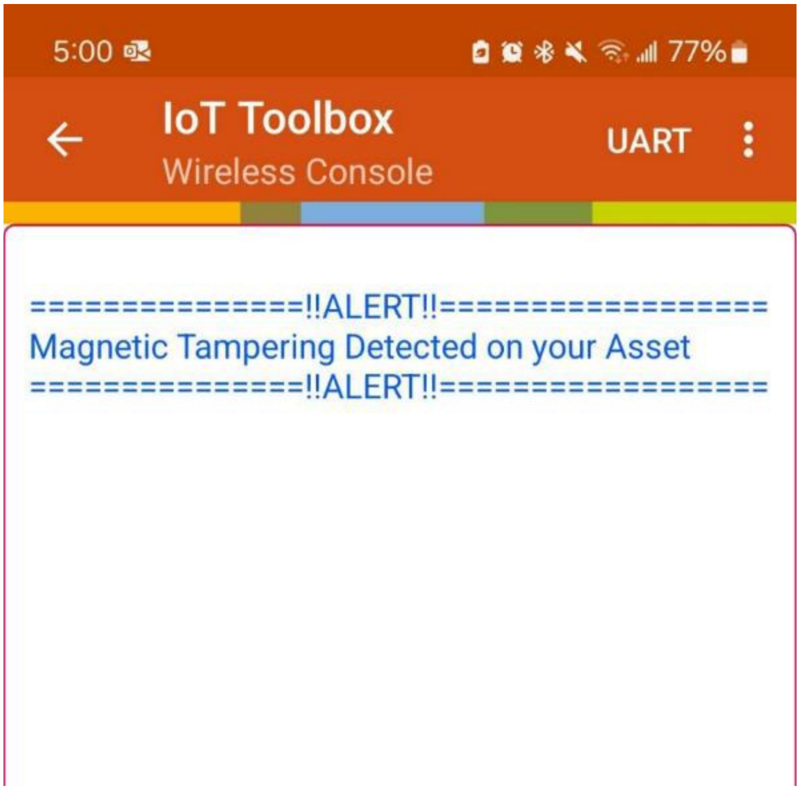
4) When a device called NXP_M0 or <your_id> appears, click on it. Your phone should now be connecting to the board.



NXP_WU
00:60:37:4C:8F:30
Unbonded -55 dBm

5) When you show magnetic tampering (put a magnet near to the NMH1000 on FRDMSTBI-NMH1000 or Hall Switch 3 click connected to FRDM-MCXW23), the NMH1000 magnetic switch sensor detects change in magnetic field and wakeup.

6) At that point, you will see Wireless UART app showing ALERT message as shown below:



```
===============!!ALERT!!==================
Magnetic Tampering Detected on your Asset
===============!!ALERT!!==================
```

7) FRDM-MCXW23 board will also show "RED" LED status. The "RED" LED status will continuously remain ON till magnetic tampering is detected.

Congratulation on successfully completing Hands-On Lab2

Congratulations!
You successfully
completed the
Hands-On Lab#2

Appendix

Revision History

Revision	Date	Description	Author
1.0	08/19/25	Creation	Amit Purohit