

Demo APP for MicroLifeDeviceSDK (Android)

Table of Contents

Chapter 1	Development Environment
Chapter 2	Entry Point and Bluetooth LE Protocol
Chapter 3	Bluetooth LE Protocol APIs
Chapter 4	Oxygen APIs
Chapter 5	User Interface of Demo App
Chapter 6	Functionality of Demo App

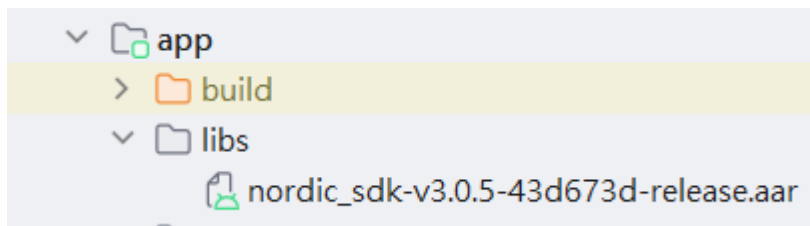
Chapter1 Development Environment

1.1 The supported SDK version is as follow:

```
android {
    namespace "com.mlc.nordic_sdk"
    compileSdkVersion 36

    defaultConfig { DefaultConfig it ->
        applicationId "com.mlc.nordic_sdk"
        minSdk 29
        targetSdk 36
        versionCode 1
        versionName "2.2.0"
    }
}
```

1.2 Add the library “nordic_sdk_vxxx-xxxxxxx.aar” into the “libs” directory.



1.3 In the “build.gradle”, add the description as bellows.

```
//aar (sdk required)
implementation fileTree(dir: "libs", include: ["*.aar", "*.jar"], exclude: [])

//nordic scan (sdk required)
implementation libs.scanner

//nordic connect (sdk required)
implementation libs.ble
implementation libs.ble.ktx
implementation libs.ble.common
implementation libs.ble.livedata

//open csv (sdk required)
implementation (libs.opencsv)

//livedata
implementation libs.runtime.livedata
```

Chapter2 Entry Point and Bluetooth LE Protocol

The “ChoseActivity” is the entry point of the sample application. The “SPO2Activity” is dedicated to the device of general Oxygen Monitor (Bluetooth LE).

```
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.ChoseActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".activity.BPMActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.SPO2Activity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.PFMActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.ThermoActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.BaseActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
    <activity
        android:name=".activity.BGMAActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
    <activity
        android:name=".activity.WBPActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
</application>
```

2.1 Initialize the instance “bluetoothManager”. This is to fulfill Bluetooth LE features and connection sequence.

```
bluetoothManager = BluetoothManager.getInstance(activity = this, listener = this)
```

2.1.1 OnIMBluetoothLEListener: For Bluetooth progress.

```
interface OnIMBluetoothLEListener {
    3 Usages 5+ Implementations
    fun onScanResult(
        device: BluetoothDevice,
        deviceName: String,
        deviceType: DeviceType?,
        macAddress: String?
    )

    5+ Implementations
    fun onConnectionState(connectState: ConnectState)

    5+ Implementations
    fun onConnectionState(connectState: ConnectState, state: Int)

    3 Usages 5+ Implementations
    fun onReceivedBleDataResult(data: List<Byte>, head: Int? = null)

    2 Usages 5+ Implementations
    fun onResponseSWRevision(swRevision: String)

    2 Usages 5+ Implementations
    fun onResponseFWRevision(fwRevision: String)

    2 Usages 5+ Implementations
    fun onResponseHWRevision(hwRevision: String)

    2 Usages 5+ Implementations
    fun onBtStateChanged(isEnable: Boolean)

    2 Usages 5+ Implementations
    fun onRetryFailed(msg: String)
}
```

2.2 Initialize the instance “bpmProtocol”. This is to fulfill retrieving BPM commands and parsing data from the BPM device.

```
// Input protocol key
4 Usages
private var spo2Protocol: SP02Protocol? = SP02Protocol.getInstance( sdkid = "")
```

2.2.1 SPO2Protocol. OnDataResponseListener: For SPO2 Protocol progress

```
interface OnDataResponseListener {
    4 Usages 2 Implementations
    fun onResponseSpo2Data(spo2Data: Spo2Data)
    2 Usages 2 Implementations
    fun onResponseSpo2Limit(spo2Limit: Spo2Limit?)
}
```

Chapter3 Bluetooth LE Protocol APIs

3.1 Instance of Bluetooth LE Protocol

3.1.1 Interface :

	<pre>fun getInstance(activity: Activity, listener: OnIMBluetoothLEListener? = null): BluetoothManager?</pre>
Definition	Initialize Bluetooth LE Protocol
Parameter	activity : name of activity or this listener : get the Bluetooth status
	<pre>bluetoothManager = BluetoothManager.getInstance(activity = this, listener = this)</pre>

3.2 Read and Write command to Bluetooth Device

3.2.1 Interface :

	suspend fun writeCommand(data: ByteArray?)
Definition	write command to Bluetooth Device
Parameter	data : ByteArray command

3.2.2 Delegate :

	<pre>override fun onReceivedBleDataResult(data: List<Byte>, head: Int?)</pre>
Definition	This is to get List<Byte> from Bluetooth device.

3.3 Connection State and Result

3.3.1 Interface :

	<pre>override fun onScanResult(device: BluetoothDevice, deviceName: String, deviceType: DeviceType?, macAddress: String?)</pre>
Definition	This is to get Bluetooth information of devices which discovered in the vicinity.
Parameter	<pre>enum class DeviceType { 4 Usages ALL, MLC_ALL, 6 Usages MLC_BPM36, MLC_BPM46, MLC_BPM56, 4 Usages MLC_Thermo, MLC_SP02, MLC_PF2, 5 Usages MLC_WS, MLC_BGM, MLC_WBP, 1 Usage WAG_ALL, WAG_BPM46 }</pre>

	override fun onConnectionState(connectState: ConnectState)
Definition	The “onConnectionState()” is to monitor the status of connection.
Parameter	<pre>enum class ConnectState { 1 Usage StartScan, StopScan, ScanFinished, Connected, DeviceReady, 4 Usages Disconnect, ConnectFailed, 2 Usages Bonded, BondNone, Bonding, ERROR_133 }</pre>

	override fun onReceivedBleDataResult(data: List<Byte>, head: Int?)
Definition	This is to get List<Byte> from Bluetooth device.

	override fun onResponseSWRevision(swRevision: String)
Definition	This is used to obtain the software revision of the Bluetooth device.

	override fun onResponseFWRevision(fwRevision: String)
Definition	This is used to obtain the hardware revision of the Bluetooth device.

	override fun onBtStateChanged(isEnable: Boolean)
Definition	This is to monitor the state of Enabled or Disabled.

	override fun onRetryFailed(msg: String)
Definition	Return if sending the command fails.

3.4 Device scanning or discovery

3.4.1 Interface :

	fun startScan(scanDeviceType: DeviceType)	
Definition	The “startScan()” is for device scanning or discovery. The result will be shown with the “onScanResult”.	
Parameter	<pre>enum class DeviceType { 4 Usages ALL, MLC_ALL, 6 Usages MLC_BPM36, MLC_BPM46, MLC_BPM56, 4 Usages MLC_Thermo, MLC_SP02, MLC_PF2, 5 Usages MLC_WS, MLC_BGM, MLC_WBP, 1 Usage WAG_ALL, WAG_BPM46 }</pre>	

	fun stopScan()	
Definition	Terminate the scanning process.	

3.5 Disconnection

	fun disconnectGatt()	
Definition	Disconnect device.	

Chapter4 Oxygen APIs

4.1 Instance of Oxygen Protocol

4.1.1 Interface :

	<pre>fun getInstance(sdkid: String, dataResponseListener: OnDataResponseListener?): SPO2Protocol?</pre>
Definition	Initialize Oxygen Protocol
Parameter	<p>sdkid : SDK ID of designated device</p> <p>listener : This is to obtain the parsed Oxygen information.</p>
	<pre>// Input protocol key 4 Usages private var <u>spo2Protocol</u>: SPO2Protocol? = SPO2Protocol.getInstance(sdkid = "")</pre>

	<pre>fun onResponseSpo2Data(spo2Data: Spo2Data)</pre>
Definition	received oxygen data.
Parameter	<pre>data class Spo2Data(var <u>pul</u>: Int? = null, var <u>spo2</u>: Int? = null, var <u>pi</u>: Int? = null, var <u>points</u>: List<Int>? = null, var <u>error</u>: Int? = null)</pre>

4.2 Read Oxygen limit

4.2.1 Interface :

	fun readSpo2Limit(): ByteArray
Definition	Read oxygen limit.

4.2.2 Delegate :

	fun onResponseSpo2Limit(spo2Limit: Spo2Limit?)
Parameter	<pre>data class Spo2Limit(var spo2UpperLimit: Int? = null, var spo2LowerLimit: Int? = null, var pulUpperLimit: Int? = null, var pulLowerLimit: Int? = null,)</pre>

4.3 Write Oxygen limit

4.3.1 Interface :

	<pre>fun writeSpo2Limit(spo2UpperLimit: Int, spo2LowerLimit: Int, pulUpperLimit: Int, pulLowerLimit: Int): ByteArray</pre>
Definition	Write oxygen limit.
Parameter	<p>spo2UpperLimit: Upper limit of SpO2 alarm, range 50–100.</p> <p>spo2LowerLimit: Lower limit of SpO2 alarm, range 50–100.</p> <p>pulUpperLimit: Upper limit of pulse rate alarm, range 5–250. Must be a multiple of 5 and cannot be 0.</p> <p>pulLowerLimit: Lower limit of pulse rate alarm, range 5–250. Must be a multiple of 5 and cannot be 0.</p>

4.4 Solve Data Result

4.4.1 Interface :

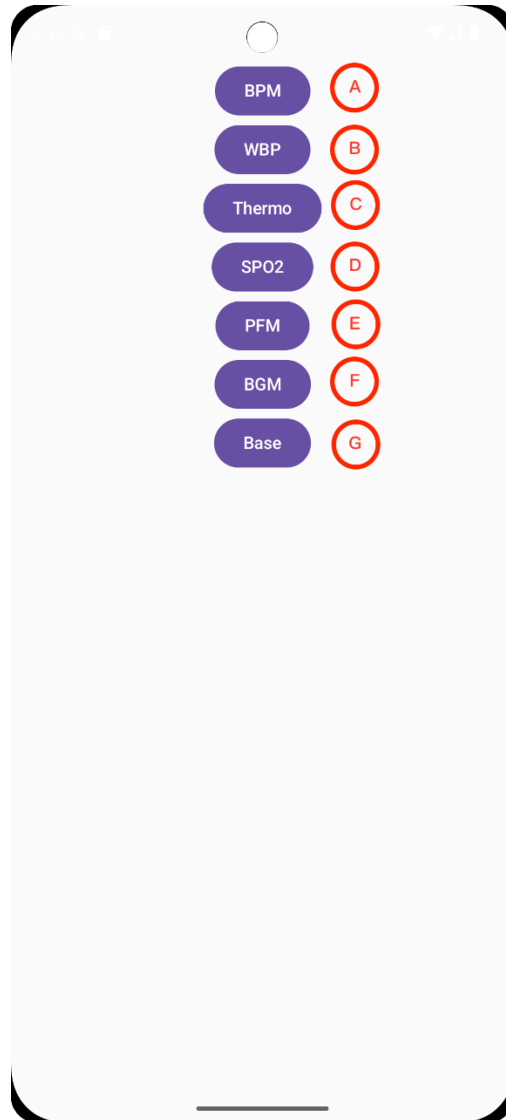
	<code>fun solveDataResult(data: String)</code>
Definition	Parses the List<Byte> received from the Bluetooth device.

	<code>fun solveDataResult(data: List<Byte>)</code>
Definition	Parses the List<Byte> received from the Bluetooth device.

Chapter5 User Interface of Demo App

5.1 Getting Started :

Start the app and then select the button “OXYGEN “ \ ” D” to communicate with the designate device Oxygen SPO2.



5.2 Operation Sequence :

The scanning (discovery) is automatically run to discover devices in the vicinity. If a device is bonded, it will be connected accordingly. If not, the “bindingDevice” can be used to run bonding process.

5.3 GUI Layout :

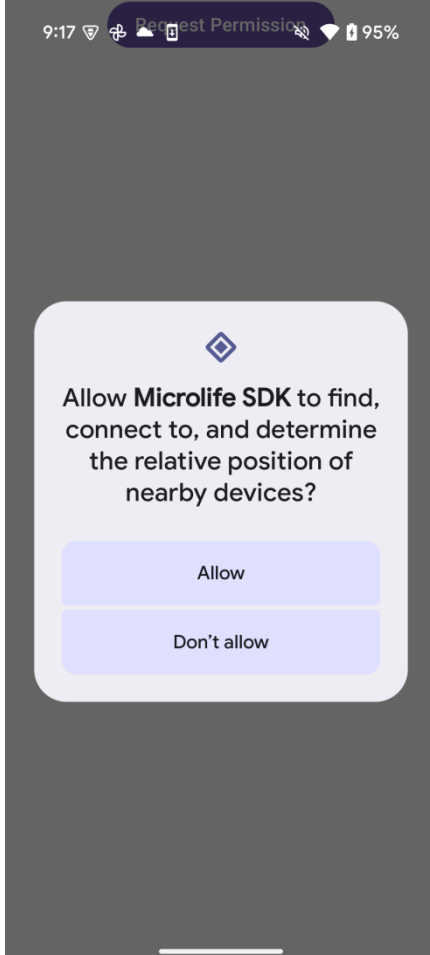


5.3.1 The log window is used to display result data information about communication handshake between App and communicate with the device Oxygen SPO2 by such as data transferring, synchronization and so on.


5.3.2 Refer to “SPO2Activity.java” from the demo application (sample code) to get more detailed.

Chapter6 Functionality of Demo App


6.1 Bluetooth authorization :

	<p>1. Request for Bluetooth permission.</p>
--	---

6.2 Command: Read data from SPO2

 <p>SPO2</p> <p>Log</p> <pre> NOTIFY -> 81426440 Spo2Data(pul=66, spo2=100, pi=64, points=null, error=0) NOTIFY -> 802B2A292A2A2B2B2A2827 Spo2Data(pul=null, spo2=null, pi=null, points=[43, 42, 41, 42, 42, 43, 43, 42, 40, 39], error=0) NOTIFY -> 802524232120201F1F1F1F Spo2Data(pul=null, spo2=null, pi=null, points=[37, 36, 35, 33, 32, 32, 31, 31, 31, 31], error=0) NOTIFY -> 801F1F1F1F222A3C596D6E Spo2Data(pul=null, spo2=null, pi=null, points=[31, 31, 31, 31, 34, 42, 60, 89, 109, 110], error=0) NOTIFY -> 80604A3934302B26252526 Spo2Data(pul=null, spo2=null, pi=null, points=[96, 74, 57, 52, 48, 43, 38, 37, 37, 38], error=0) NOTIFY -> 8142643F Spo2Data(pul=66, spo2=100, pi=63, points=null, error=0) NOTIFY -> 802627272625232221201F Spo2Data(pul=null, spo2=null, pi=null, points=[38, 39, 39, 38, 37, 35, 34, 33, 32, 31], error=0) NOTIFY -> 801E1E1E1D1D1D1D1D1D1E Spo2Data(pul=null, spo2=null, pi=null, points=[30, 30, 30, 29, 29, 29, 29, 29, 29, 30], error=0) NOTIFY -> 801F24304A67726C584238 Spo2Data(pul=null, spo2=null, pi=null, points=[31, 36, 48, 74, 103, 114, 108, 88, 66, 56], error=0) NOTIFY -> 80352F2926252525262626 Spo2Data(pul=null, spo2=null, pi=null, points=[53, 47, 41, 38, 37, 37, 37, 38, 38, 38], error=0) NOTIFY -> 81426440 Spo2Data(pul=66, spo2=100, pi=64, points=null, error=0) NOTIFY -> 802524222120201E1E1E1D Spo2Data(pul=null, spo2=null, pi=null, points=[37, 36, 35, 33, 32, 32, 31, 31, 31, 31], error=0) </pre>	<p>onResponseSpo2Data :</p> <p>-> Spo2Data {SPO='100', Plus='66', Pi='64', spoDataArray=null}</p> <p>-> Spo2Data = OxygenResult{SPO='null', Plus='null', Pi='null', spoDataArray=[43, 42, 41, 42, 42, 43, 43, 42, 40, 39]}</p>
--	--

6.3 Command: Disconnect the Bluetooth

 <p>The screenshot shows the SPO2 app interface with a status bar at the top displaying 'SPO2' and a Bluetooth icon. Below the status bar is a 'Log' section containing a list of log entries. The entries show a sequence of 'NOTIFY' messages and 'Spo2Data' updates. The log ends with 'Disconnect'.</p> <pre> Log 45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 81FF7F00 Spo2Data(pul=255, spo2=127, pi=0, points=null, error=1) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 81FF7F00 Spo2Data(pul=255, spo2=127, pi=0, points=null, error=1) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) NOTIFY -> 802D2D2D2D2D2D2D2D2D2D2D2D2D2D2D Spo2Data(pul=null, spo2=null, pi=null, points=[45, 45, 45, 45, 45, 45, 45, 45, 45], error=0) Disconnect </pre>	<ol style="list-style-type: none"> 1. Disconnect the Bluetooth 2. If the oxygen device leaves the finger for a period of time, the bluetooth of the oxygen device will automatically disconnect. 3. The result is able to observe by the “onConnectionState(ConnectState state)” and its status can be found by the “ConnectState = Disconnect”.
---	---