

# **Demo APP for MicroLifeDeviceSDK (Body Temperature / Thermometer) - Android**

## **Table of Contents**

- Chapter 1    Development Environment**
- Chapter 2    Entry Point and Bluetooth LE Protocol**
- Chapter 3    Bluetooth LE Protocol & APIs**
- Chapter 4    Thermometer APIs**
- Chapter 5    User Interface and Functionality**
- Chapter 6    Demo App**

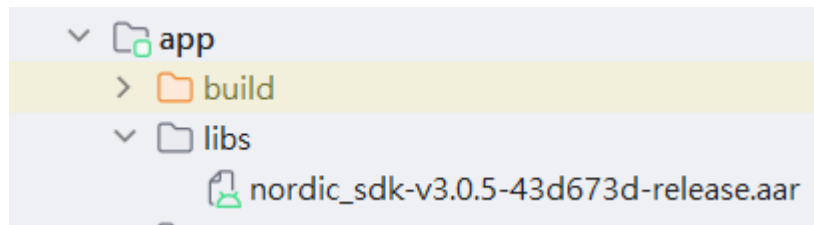
## Chapter 1 Development Environment

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"
    }
}
```

- 1.1. Add the library “nordic\_sdk\_vxxx-xxxxxxx.aar” into the “libs” directory.



- 1.2. In the “build.gradle(.kts)”, 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
```



## Chapter 2 Entry Point and Bluetooth LE Protocol

The “ChoseActivity” is the entry point of the sample application.

The “ThermoActivity” is dedicated to Thermometer.

```

        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.SP02Activity"
    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.BGMActivity"
    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 “ThermoProtocol”. This is to fulfill Bluetooth LE features and connection sequence.

### Activity :

```
class ThermoActivity : ComponentActivity(), OnIMBluetoothLEListener, ThermoProtocol.OnDataResponseListener {
    val TAG = "ThermoActivity"
    22 Usages
    private val viewModel by viewModels<ThermoActivityViewModel>()
    2 Usages
    private var bluetoothManager: BluetoothManager? = null
    2 Usages
    private var deviceName = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        initParam()

        setContent {
            ThermoScreen(viewModel = viewModel)
        }
    }

    1 Usage
    private fun initParam() {
        bluetoothManager = BluetoothManager.getInstance(activity = this, listener = this)
        viewModel.setBleReceiveManager(bluetoothManager)

        viewModel.setOnDataResponseListener(this)
    }
}
```

- 2.1.1 OnIMBluetoothLEListener: For Bluetooth progress.

```

public interface OnIMBluetoothLEListener {
    5+ Implementations
    public abstract fun onScanResult(device: android.bluetooth.BluetoothDevice): kotlin.Unit?

    5+ Implementations
    public abstract fun onConnectionState(connectState: com.microlife.device.sdk.bluetooth.BluetoothState): kotlin.Unit?

    5+ Implementations
    public abstract fun onConnectionState(connectState: com.microlife.device.sdk.bluetooth.BluetoothState): kotlin.Unit?

    5+ Implementations
    public abstract fun onReceivedBleDataResult(data: kotlin.ByteArray): kotlin.Unit?

    5+ Implementations
    public abstract fun onResponseSWRevision(swRevision: kotlin.String): kotlin.Unit?

    5+ Implementations
    public abstract fun onResponseFWRevision(fwRevision: kotlin.String): kotlin.Unit?

    5+ Implementations
    public abstract fun onResponseHWRevision(hwRevision: kotlin.String): kotlin.Unit?

    5+ Implementations
    public abstract fun onBtStateChanged(isEnable: kotlin.Boolean): kotlin.Unit?

    5+ Implementations
    public abstract fun onRetryFailed(msg: kotlin.String): kotlin.Unit?
}

```

### 2.1.2 ThermoProtocol.OnDataResponseListener: For Thermo Protocol progress

```

public interface OnDataResponseListener {
    1 Usage 2 Implementations
    public abstract fun onResponseDeviceInfo(macAddress: kotlin.String): kotlin.Unit?

    1 Usage 2 Implementations
    public abstract fun onResponseUploadMeasureData(thermoMeasureData: kotlin.ByteArray): kotlin.Unit?

    2 Implementations
    public abstract fun onResponseUploadCalibrate(calibrateParameter: kotlin.ByteArray): kotlin.Unit?

    2 Implementations
    public abstract fun onWriteCommand(byteArray: kotlin.ByteArray): kotlin.Unit?
}

```

ViewModel :

```

4 Usages
class ThermoActivityViewModel: ViewModel() {
    private val TAG = "ThermoActivityViewModel"

    1 Usage
    private var deviceName: String? = ""
    5 Usages
    private var bluetoothManager: BluetoothManager? = null
    2 Usages
    private var isDestroy = false
    2 Usages
    private var thermoProtocol: ThermoProtocol? = ThermoProtocol.getInstance(sdkey = THERMO_SDK_KEY)
    var state by mutableStateOf(value = ConnectState.Disconnect)

    //UI variable
    3 Usages
    private val _listData = MutableLiveData<List<String>>((value = emptyList()))
    val listData: LiveData<List<String>>
    |   get() = _listData

    fun setBleReceiveManager(bluetoothManager: BluetoothManager?) {
        |   this.bluetoothManager = bluetoothManager
        |
        |   startScan()
    }

    1 Usage
    fun setDeviceName(deviceName: String?) {
        |   this.deviceName = deviceName
    }

    1 Usage
    fun setOnDataResponseListener(onDataResponseListener: ThermoProtocol.OnDataResponseListener) {
        |   this.thermoProtocol?.setOnDataResponseListener(onDataResponseListener)
    }

```

2.1.3 The “setBleReceiveManager()”is to get Bluetooth response status from device.

2.1.4 The “setOnDataResponseListener()” is to get the response from device.

## Chapter 3 Bluetooth LE Protocol & APIs

### 3.1. Instance of Bluetooth LE Protocol :

#### 3.1.1. Interface :

	Private var *Protocol: *Protocol? = getInstance(String sdkid)
Definition	Initialize Bluetooth LE Protocol for Thermo device
Parameter	String sdkid : SDK ID of designated device
	<pre>private var thermoProtocol: ThermoProtocol? =     ThermoProtocol.getInstance(sdkid = THERMO_SDK_KEY)</pre>

### 3.2. Connection State and Result :

#### 3.2.1. Interface :

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

#### 3.2.2. Delegate :

	override fun onScanResult( device: BluetoothDevice, deviceName: String, deviceType: DeviceType?, macAddress: String? )
Definition	This is to get Bluetooth information of devices which



	discovered in the vicinity.
Parameter	device: BluetoothDevice deviceName: device name deviceType: device type <pre>enum class DeviceType {     ALL, MLC_ALL, MLC_BPM3G, MLC_BPM4G,     MLC_BPM5G, MLC_Thermo, 6 Usages     MLC_SPO2, MLC_PF2, 6 Usages     MLC_WS, MLC_BGM, MLC_WBP, 5 Usages     WAG_ALL, WAG_BPM4G 1 Usage }</pre> macAddress: device mac address

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

### 3.3. Device scanning or discovery :

#### 3.3.1. Interface :

	fun startScan()
Definition	The “startScan()” is for device scanning or discovery. The result will be shown with the “onScanResult”.
Parameter	None

	fun stopScan()
Definition	Terminate the scanning process.

#### 3.3.2. Delegate :

	override fun onConnectionState(connectState: ConnectState)
Definition	The “onConnectionState()” is to monitor the status of connection.

Parameter	<pre>enum class ConnectState {     StartScan, StopScan, ScanFinished, 1     Connected, DeviceReady,     Disconnect, ConnectFailed,     Bonded, BondNone, Bonding, ERROR_133 }</pre>
-----------	---

## 3.4. Disconnection :

## 3.4.1. Interface :

	fun disconnect()
Definition	Disconnect device.

## 3.4.2. Delegate :

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



## Chapter 4 Thermometer APIs

- 4.1. Received device information : Once Bluetooth connects to the thermometer successfully, the first time the thermometer sends the information.

	fun onResponseDeviceInfo(macAddress: String, workMode: Int, batteryVoltage: Float)
Parameter	macAddress : macAddress description  workMode : workMode description  batteryVoltage : batteryVoltage description

- 4.2. Received the current measurement result / data :

	fun onResponseUploadMeasureData(thermoMeasureData: ThermoMeasureData)
Parameter	thermoMeasureData : measurement result

- 4.3. Received the thermo calibrate parameter result / data :

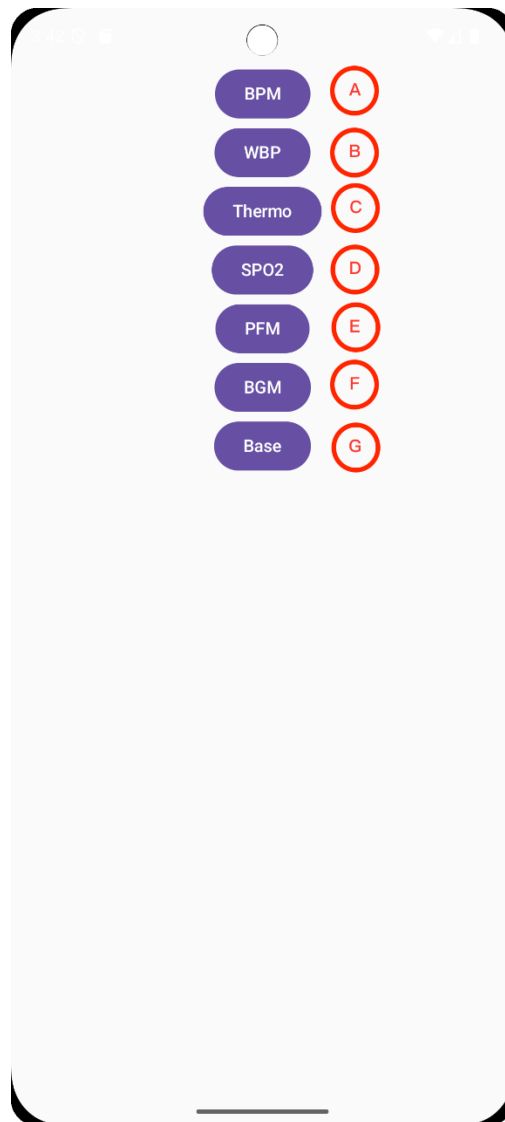
	fun onResponseUploadCalibrate(calibrateParameters: List<CalibrateParameter>)
Parameter	calibrateParameters : calibrate parameters result  <pre>data class CalibrateParameter(     val ca2_parameter: Int? = null,     val ca3_parameter: Int? = null,     val ca3_voltage: Int? = null,     val ca3_ambient: Int? = null, )</pre>



## Chapter 5 User Interface and Functionality

### 5.1 Getting Started :

Start the app and then select the button “BODY TEMPERATURE” / “C” to communicate with the designate device Thermometer.

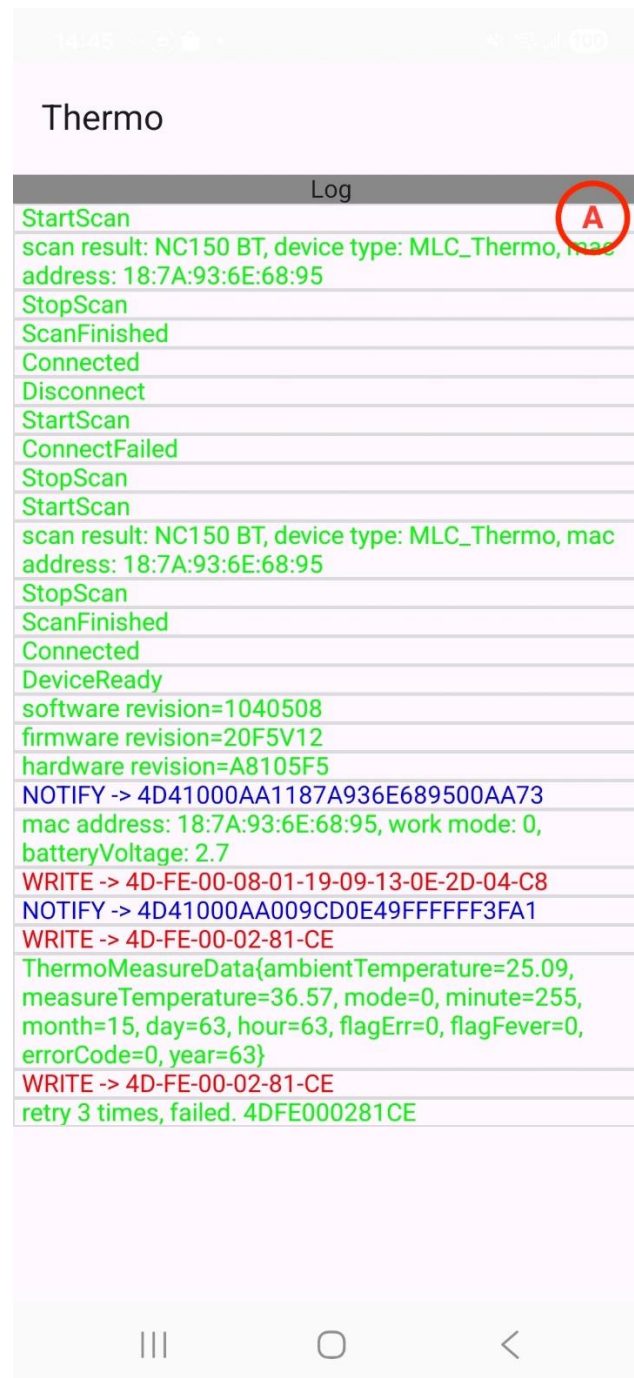


### 5.2 Operation Sequence :

The scanning (discovery) is automatically run to discover devices in the vicinity. If a device is bonded, it will be connected accordingly.



### 5.3 GUI Layout :

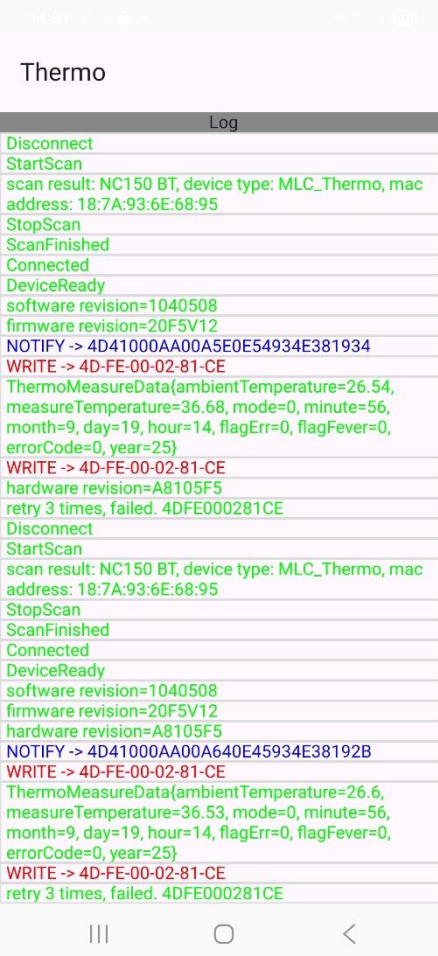


5.3.1 Region A : The log window is used to display information about communication handshake between App and device.

5.4 Refer to “BtTestActivity” from the demo application (sample code) to get more detailed.

## Chapter 1 Demo App

### 6.1. Scanning (Discovery), Connection, Data (Readings) :

 <p>Thermo</p> <p>Log</p> <p>Disconnect</p> <p>StartScan</p> <p>scan result: NC150 BT, device type: MLC_Thermo, mac address: 18:7A:93:6E:68:95</p> <p>StopScan</p> <p>ScanFinished</p> <p>Connected</p> <p>DeviceReady</p> <p>software revision=1040508</p> <p>firmware revision=20F5V12</p> <p>NOTIFY -&gt; 4D41000AA00A5E0E54934E381934</p> <p>WRITE -&gt; 4D-FE-00-02-81-CE</p> <p>ThermoMeasureData{ambientTemperature=26.54, measureTemperature=36.68, mode=0, minute=56, month=9, day=19, hour=14, flagErr=0, flagFever=0, errorCode=0, year=25}</p> <p>WRITE -&gt; 4D-FE-00-02-81-CE</p> <p>hardware revision=A8105F5</p> <p>retry 3 times, failed. 4DFE000281CE</p> <p>Disconnect</p> <p>StartScan</p> <p>scan result: NC150 BT, device type: MLC_Thermo, mac address: 18:7A:93:6E:68:95</p> <p>StopScan</p> <p>ScanFinished</p> <p>Connected</p> <p>DeviceReady</p> <p>software revision=1040508</p> <p>firmware revision=20F5V12</p> <p>hardware revision=A8105F5</p> <p>NOTIFY -&gt; 4D41000AA00A640E45934E38192B</p> <p>WRITE -&gt; 4D-FE-00-02-81-CE</p> <p>ThermoMeasureData{ambientTemperature=26.6, measureTemperature=36.53, mode=0, minute=56, month=9, day=19, hour=14, flagErr=0, flagFever=0, errorCode=0, year=25}</p> <p>WRITE -&gt; 4D-FE-00-02-81-CE</p> <p>retry 3 times, failed. 4DFE000281CE</p>	<p>1. Once starts the demo App, it will be conducting a scanning / discovery process. The device named NC150 BT is displayed in the vicinity. The connection can be created by the MAC address “18:7A:93:6E:68:95”.</p> <p>2. The received data (included Device information &amp; Readings) is as <b>Blue</b> part, and it can be decode in <b>Green</b> part. The following are the</p> <p>(a) Device information:</p> <p>THERMO : DeviceInfo -&gt; macAddress = 187A936E6895 , workMode = 0 , batteryVoltage = 2.7.</p> <p>(b) Readings:</p> <p>THERMO : UploadMeasureData -&gt; ThermoMeasureData = ThermoMeasureData{ambientTemperature=26.54, measureTemperature=36.68, mode=0, minute=56, month=9, day=19, hour=14, flagErr=0, flagFever=0, errorCode=0, year=25}.</p> <p>3. The <b>Red</b> part is ACK to inform device the process is complete.</p>
--	--