

# **Demo APP for MicroLifeDeviceSDK (Android)**

## **Table of Contents**

<b>Chapter 1</b>	<b>Development Environment</b>
<b>Chapter 2</b>	<b>Entry Point and Bluetooth LE Protocol</b>
<b>Chapter 3</b>	<b>Bluetooth LE Protocol APIs</b>
<b>Chapter 4</b>	<b>WBP APIs</b>
<b>Chapter 5</b>	<b>User Interface of Demo App</b>
<b>Chapter 6</b>	<b>Functionality of Demo App</b>
	<b>Appendix</b>

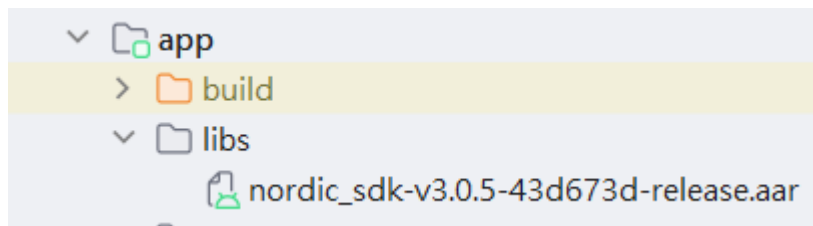
## Chapter1 Development Environment

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 “WBPAActivity” is dedicated to the device of general Blood Pressure 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.BPMAActivity"
        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.PFMAActivity"
        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.WBPAActivity"
        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 “wbpProtocol”. This is to fulfill retrieving WBP commands and parsing data from the WBP device.

```
// Input protocol key
this.wbpProtocol = WBPProtocol.getInstance(
    sdkid = "",
    listener
)
```

## 2.2.1 WBPProtocol. OnDataResponseListener: For WBP Protocol progress

```

interface OnDataResponseListener {
    2 Usages 2 Implementations
    fun onResponseWBPPReadUsualModeHistory(dRecordUsual: DRecordUsual) //00h
    1 Usage 2 Implementations
    fun onResponseWBPPReadUsualModeHistoryEachMeasurement(
        totalPackageNumber: Int,
        packageNumber: Int,
        dRecordUsualEach: DRecordUsualEach
    ) //00h-F1
    1 Usage 2 Implementations
    fun onResponseWBPPReadDiagnosticModeHistory(dRecordDiagnostic: DRecordDiagnostic) //01h
    2 Usages 2 Implementations
    fun onResponseWBPPClearSelectedModeHistory(success: Boolean) //02h
    2 Usages 2 Implementations
    fun onResponseWBPPClearCurrentModeHistory(success: Boolean) //03h
    2 Usages 2 Implementations
    fun onResponseWBPPWriteDeviceTime(success: Boolean) //05h
    1 Usage 2 Implementations
    fun onResponseWBPPWriteNewUserId(success: Boolean) //06h
    1 Usage 2 Implementations
    fun onResponseWBPPReadNocturnalModeSetting(nocturnalInfo: NocturnalInfo) //07h-01
    1 Usage 2 Implementations
    fun onResponseWBPPChangeNocturnalModeSetting(success: Boolean) //07h-02
    1 Usage 2 Implementations
    fun onResponseWBPPReadMeasurementSetting(measurementSetting: MeasurementSetting) //08h-00
    1 Usage 2 Implementations
    fun onResponseWBPPWriteMeasurementSetting(success: Boolean) //08h-01
    1 Usage 2 Implementations
    fun onResponseWBPPReadDeviceIDAndInfo(deviceInfo: DeviceInfo) //0Bh
    1 Usage 2 Implementations
    fun onResponseWBPPReadDeviceTime(deviceTime: DeviceTime) //0Ch
    1 Usage 2 Implementations
    fun onResponseWBPPReadUserIDAndVersionData(userIDAndVersionData: UserIDAndVersionData) //0Dh
    1 Usage 2 Implementations
    fun onResponseWBPPReadNocturnalModeHistory(dRecordNocturnal: DRecordNocturnal) //0E
    1 Usage 2 Implementations
    fun onResponseWBPPReadSerialNumber(serial_number: String) //0Fh-00
    2 Usages 2 Implementations
    fun onResponseWBPNACK(msg: String)
    4 Usages 2 Implementations
    fun onWriteWBPPCommand(byteArray: ByteArray, nextCommand: String)
}

```

## 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&lt;Byte&gt;,     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_BPM3G, MLC_BPM4G, MLC_BPM5G,     4 Usages     MLC_Thermo, MLC_SP02, MLC_PF2,     5 Usages     MLC_WS, MLC_BGM, MLC_WBP,     1 Usage     WAG_ALL, WAG_BPM4G }</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_BPM3G, MLC_BPM4G, MLC_BPM5G,     4 Usages     MLC_Thermo, MLC_SP02, MLC_PF2,     5 Usages     MLC_WS, MLC_BGM, MLC_WBP,     1 Usage     WAG_ALL, WAG_BPM4G }</pre>	

	fun stopScan()	
Definition	Terminate the scanning process.	

### 3.5 Disconnection

	fun disconnectGatt()	
Definition	Disconnect device.	

## Chapter4 WBP APIs

### 4.1 Instance of WBP Protocol

#### 4.1.1 Interface :

	<pre>fun getInstance(     sdkid: String,     listener: OnDataResponseListener? = null ): WBPProtocol?</pre>
Definition	Initialize WBP Protocol
Parameter	<p>sdkid : SDK ID of designated device</p> <p>listener : This is to obtain the parsed WBP information.</p>
	<pre>// Input protocol key this.wbpProtocol = WBPProtocol.getInstance(     sdkid = "",     listener )</pre>

## 4.2 Read usual mode history data from WBP :

### 4.2.1 Interface :

	fun readUsualModeHistory(): ByteArray
Definition	Read usual mode history data.

### 4.2.2 Delegate :

	fun onResponseWBPRReadUsualModeHistory( dRecordUsual: DrecordUsual )
Parameter	<pre>public final data class DRecordUsual(     public final var mode: Int,     public final var historyMeasurementNumber: Int,     public final var mData: List&lt;MData&gt; )</pre>

### 4.3 Read usual mode history each measurement data from WBP :

#### 4.3.1 Interface :

	<pre>fun readUsualModeHistoryEachMeasurement (     selectUser: Long ): ByteArray?</pre>
Definition	Read usual mode history each measurement data.

#### 4.3.2 Delegate :

	<pre>fun onResponseWBPRadUsualModeHistoryEachMeasurement(     totalPackageNumber: Int,     packageNumber: Int,     dRecordUsualEach: DrecordUsualEach )</pre>
Parameter	<p>(1) total package number, package number: If “total package number” &gt;=2, there should be multi packages to send. Distinguish each package by “package number” and wait data transmit until receive last package which “package number” is equal to “Total package number”. Only “DRecord” will be divided by multi package.</p> <p>(2) dRecordUsualEach</p> <pre>public final data class DRecordUsualEach(     public final var mode: Int?,     public final var historyMeasurementNumber: Int?,     public final var mDatumUsualEaches: List&lt;MDataUsualEach&gt; )</pre>

#### 4.4 Read diagnostic mode history from WBP :

##### 4.4.1 Interface :

	fun readDiagnosticModeHistory(): ByteArray
Definition	Read diagnostic mode history.

##### 4.4.2 Delegate :

	fun onResponseWBPreadDiagnosticModeHistory( dRecordDiagnostic: DRecordDiagnostic )	
Parameter	<pre> public final data class DRecordDiagnostic(     public final var mode: Int,     public final var historyMeasurementNumber: Int,     public final var mDatumDiagnostics: List&lt;MData&gt;,     public final var avgAll: BPAvg,     public final var avgM: BPAvg,     public final var avgE: BPAvg,     public final var diagMemoryEndIndex01b: Int,     public final var diagMemoryEndIndex02b: Int,     public final var diagMemorySet01b: Int,     public final var diagMemorySet02b: Int,     public final var diagDayCount: Int,     public final var diagTimes: Int,     public final var diagOver: Int ) </pre>	

#### 4.5 Clear selected mode history of WBP :

##### 4.5.1 Interface :

	<pre>fun clearSelectedModeHistory(     usual: Boolean = false,     diagnostic: Boolean = false,     nocturnal: Boolean = false ): ByteArray</pre>
Definition	Clear selected mode history.

##### 4.5.2 Delegate :

	<pre>fun onResponseWBPClearSelectedModeHistory(success: Boolean)</pre>
Parameter	success : True of False

#### 4.6 Clear current mode history of WBP :

##### 4.6.1 Interface :

	<pre>fun clearCurrentModeHistory(): ByteArray</pre>
Definition	Clear current mode history.

##### 4.6.2 Delegate :

	<pre>fun onResponseWBPClearCurrentModeHistory(success: Boolean)</pre>
Parameter	success : True of False

## 4.7 Disconnect Bluetooth from WBP :

### 4.7.1 Interface :

	fun disconnectBluetooth (): ByteArray
Definition	Disconnect Bluetooth.

### 4.7.2 Delegate :

	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>



#### 4.8 Write device time to WBP :

##### 4.8.1 Interface :

	fun writeDeviceTime(): ByteArray
Definition	Write device time to WBP.

##### 4.8.2 Delegate :

	fun onResponseWBPWriteDeviceTime(success: Boolean)
Parameter	success : True of False

#### 4.9 Write new user ID to WBP :

##### 4.9.1 Interface :

	fun writeNewUserId(id: String = "b1001"): ByteArray
Definition	Write new user ID to WBP.

##### 4.9.2 Delegate :

	fun onResponseWBPWriteNewUserId(success: Boolean)
Parameter	success : True of False

#### 4.10 Read nocturnal mode setting from WBP :

##### 4.10.1 Interface :

	fun readNocturnalModeSetting(): ByteArray
Definition	Read nocturnal mode setting.

##### 4.10.2 Delegate :

	fun onResponseWBPWriteNewUserId(success: Boolean)
Parameter	success : True of False

#### 4.11 Change nocturnal mode setting of WBP :

##### 4.11.1 Interface :

	fun changeNocturnalModeSetting(nocturnalInfo: NocturnalInfo): ByteArray
Definition	Change nocturnal mode setting.
Parameter	<pre>public final data class NocturnalInfo(     public final var openNocturnal: Int,     public final var startYear: Int,     public final var startMonth: Int,     public final var startDay: Int,     public final var startHour: Int )</pre>

##### 4.11.2 Delegate :

	fun onResponseWBPChangeNocturnalModeSetting(success: Boolean)
Parameter	success : True or False

#### 4.12 Read measurement setting from WBP :

##### 4.12.1 Interface :

	fun readMeasurementSetting(): ByteArray
Definition	Read measurement setting.

##### 4.12.2 Delegate :

	fun onResponseWBPRadMeasurementSetting( measurementSetting: MeasurementSetting )
Parameter	<pre>public final data class MeasurementSetting(     public final var measurement_times: Int = 3,     public final var rest_time: Int = 0,     public final var interval_time_seconds: Int? = 15,     public final var exclude_average: Int = 0,     public final var sw_afib: Int )</pre>

## 4.13 Write measurement setting to WBP

	<pre>fun writeMeasurementSetting(     measurement_times: Int = 3,     rest_time: Int = 0,     interval_time: Int = 7,     exclude_average: Int = 0,     sw_afib: Int ): ByteArray</pre>
Definition	Write measurement setting.

## 4.13.1 Delegate :

	fun onResponseWBPWriteMeasurementSetting(success: Boolean)
Parameter	success : True of False

## 4.14 Read device ID and info from WBP

	fun readDeviceIDAndInfo(): ByteArray
Definition	Read device ID and info.

## 4.14.1 Delegate :

	fun onResponseWBPRReadDeviceIDAndInfo(deviceInfo: DeviceInfo)
Parameter	<pre>public final data class DeviceInfo(     public final var device_id: String,     public final var connectType: String = "",     public final var measurement_times: Int,     public final var error1: Int,     public final var error2: Int,     public final var error3: Int,     public final var error5: Int )</pre>

## 4.15 Read device time from WBP

	fun readDeviceTime(): ByteArray
Definition	Read device time.

## 4.15.1 Delegate :

	fun onResponseWBPreadDeviceTime(deviceTime: DeviceTime)
Parameter	<pre> public final data class DeviceTime(     public final var year: Int,     public final var month: Int,     public final var day: Int,     public final var hour: Int,     public final var minute: Int,     public final var second: Int ) </pre>

## 4.16 Read user ID and version data from WBP

	fun readUserIDAndVersionData(): ByteArray
Definition	Read user ID and version data.

## 4.16.1 Delegate :

	<pre> fun onResponseWBPreadUserIDAndVersionData(     userIDAndVersionData: UserIDAndVersionData ) </pre>
Parameter	<pre> public final data class UserIDAndVersionData(     public final var current_user_no: Int,     public final var user_id: String,     public final var fw_name: String,     public final var fw_date: String,     public final var max_user: Int,     public final var max_memory: Int,     public final var diagnostic_mode_afib: Boolean,     public final var usual_mode_afib: Boolean,     public final var nocturnal_mode: Boolean,     public final var bp_type: String,     public final var device_batt: Float,     public final var p_id: String,     public final var offset_year: Int,     public final var current_mode: Int ) </pre>

## 4.17 Read nocturnal mode history from WBP

	fun readNocturnalModeHistory(): ByteArray
Definition	Read nocturnal mode history.

## 4.17.1 Delegate :

	<pre>fun onResponseWBPReadNocturnalModeHistory(     dRecordNocturnal: DrecordNocturnal )</pre>
Parameter	<pre>public final data class DRecordNocturnal(     public final var data_index: Int,     public final var initial_year: Int,     public final var initial_month: Int,     public final var initial_date: Int,     public final var initial_hour: Int,     public final var initial_minute: Int,     public final var mDatumDiagnostics: List&lt;MDataNocturnal&gt; )</pre>

## 4.18 Read serial number from WBP

	fun readSerialNumber(): ByteArray
Definition	Read serial number.

## 4.18.1 Delegate :

	<pre>fun onResponseWBPReadSerialNumber(     serial_number: String )</pre>
Parameter	Example: WBP Reply serial number="10000000000000000000"

## 4.19 Solve Data Result

### 4.19.1 Interface :

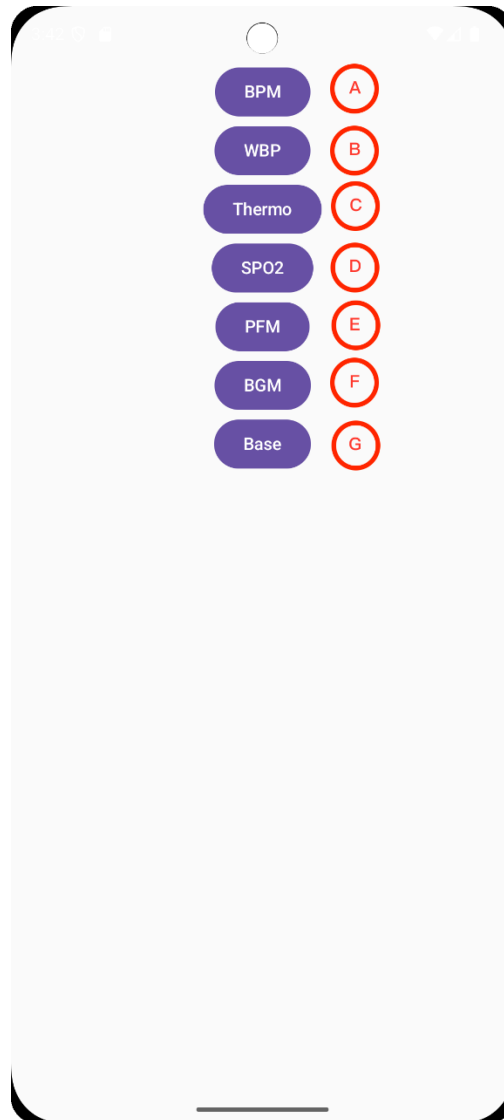
	<code>fun solveDataResult(data: String)</code>
Definition	Parses the List<Byte> received from the Bluetooth device.

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

## Chapter5 User Interface of Demo App

### 5.1 Getting Started :

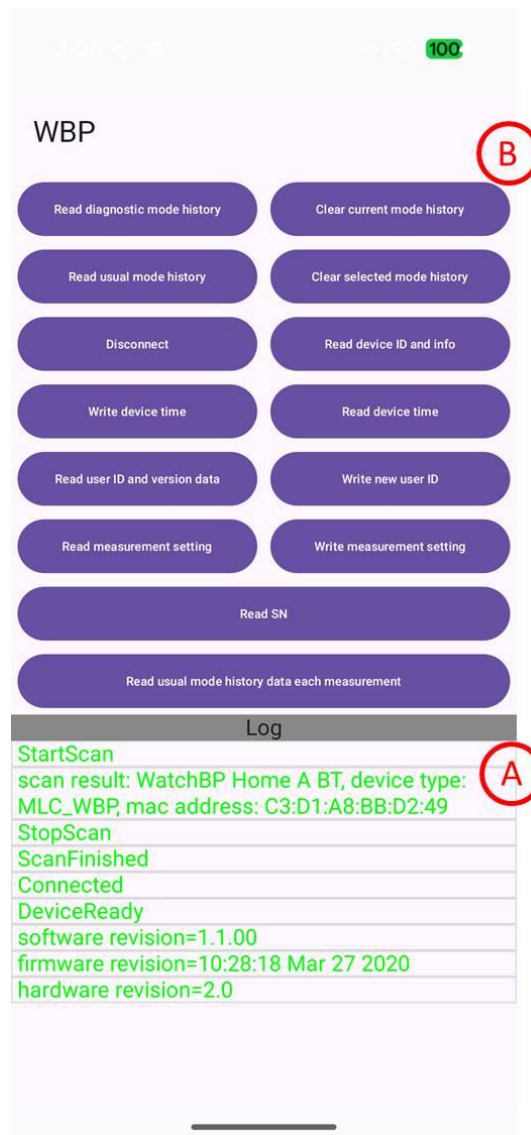
Start the app and then select the button “WatchBP” / “B” to communicate with the designate device WBP.



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

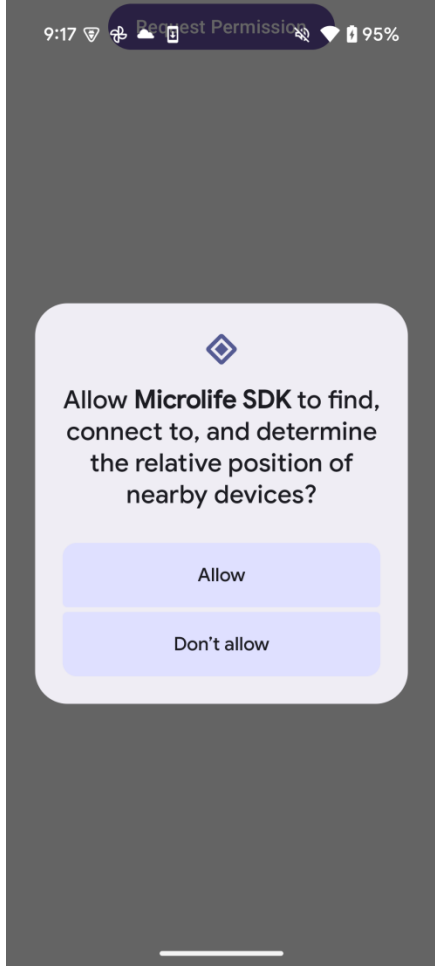
5.3.2 Region B : This part is to communicate with the device WBP by different functions / commands such as data transferring, synchronization and so on.

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

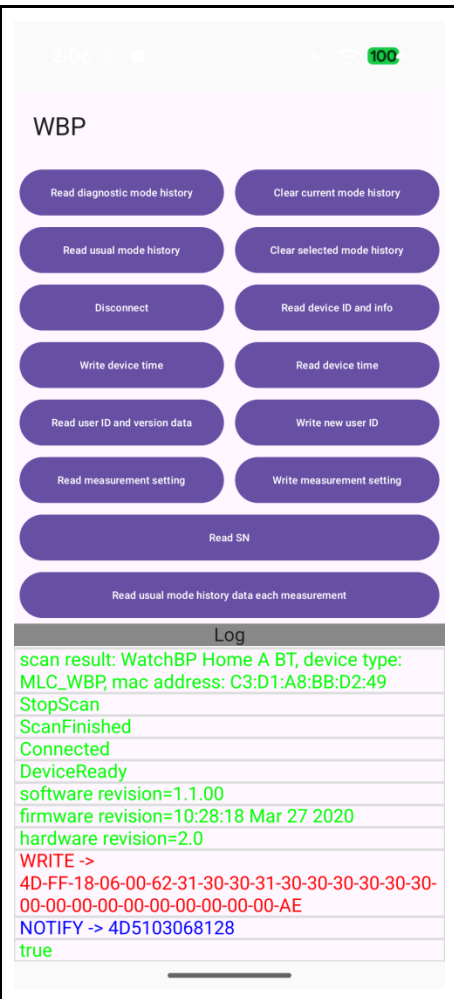


## Chapter6    Functionality of Demo App

### 6.1    Bluetooth authorization :

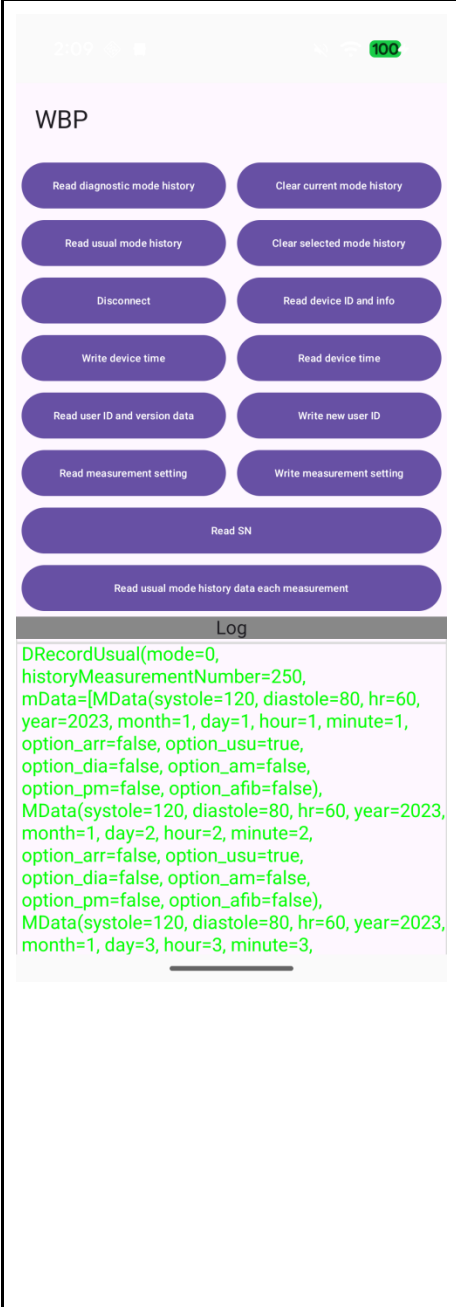
	<p>1.    Request for Bluetooth permission.</p>
--	--

## 6.2 Command: Write a new user ID to WBP:

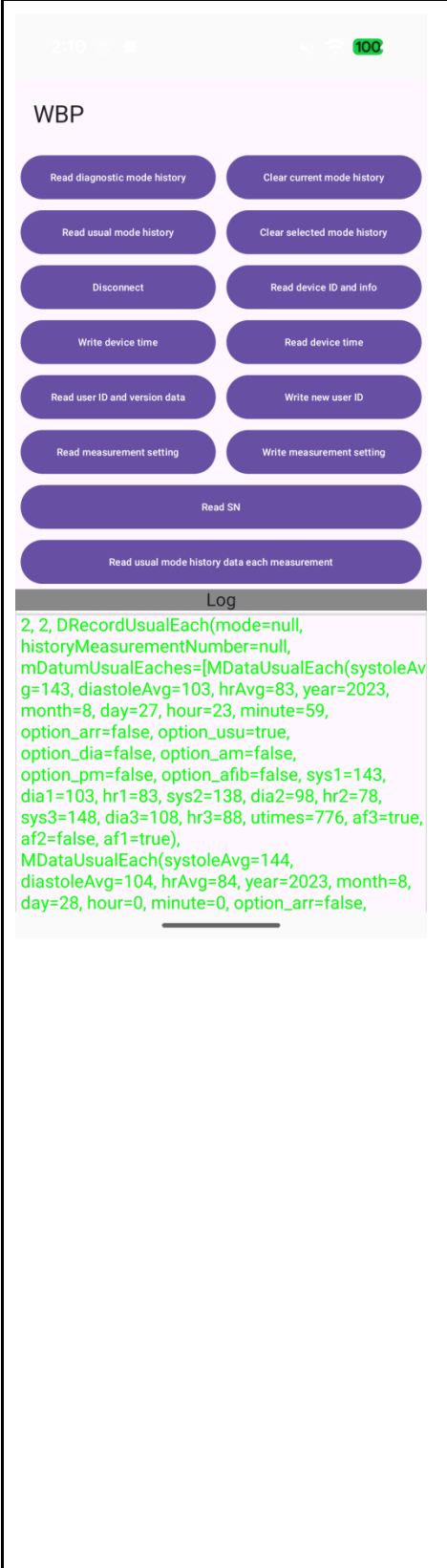
 <p>The screenshot shows the WBP app interface. At the top, there's a status bar with '100%' battery. Below it, the title 'WBP' is displayed. A grid of purple buttons includes: 'Read diagnostic mode history', 'Clear current mode history', 'Read usual mode history', 'Clear selected mode history', 'Disconnect', 'Read device ID and info', 'Write device time', 'Read device time', 'Read user ID and version data', 'Write new user ID', 'Read measurement setting', 'Write measurement setting', 'Read SN', and 'Read usual mode history data each measurement'. At the bottom, a 'Log' window displays the following text:</p> <pre> scan result: WatchBP Home A BT, device type: MLC_WBP, mac address: C3:D1:A8:BB:D2:49 StopScan ScanFinished Connected DeviceReady software revision=1.1.00 firmware revision=10:28:18 Mar 27 2020 hardware revision=2.0 WRITE -&gt; 4D-FF-18-06-00-62-31-30-30-31-30-30-30-30-30-30- 00-00-00-00-00-00-00-00-00-00-AE NOTIFY -&gt; 4D5103068128 true </pre>	<ol style="list-style-type: none"> <li>1. The command “WRITE USER” is to write a new user ID to WBP.</li> <li>2. The log “WRITE : write user“ and “Write : userID:b1001” are indicated that the App sends a command with an user ID. The ID is made up of ASCII code.</li> <li>3. The log “WBP:WriteUser -&gt; isSuccess = true” means that the writing/ sending procedure is successful.</li> <li>4. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from WBP via Bluetooth.</li> </ol>
---	---

### 6.3 Command: Read user ID and version data from WBP:


## 6.4 Command: Read usual mode history data from WBP

 <p>The screenshot shows the WBP application interface. At the top, there's a status bar with signal strength, Wi-Fi, and 100% battery. Below the title 'WBP', there are several purple buttons arranged in a grid: 'Read diagnostic mode history', 'Clear current mode history', 'Read usual mode history', 'Clear selected mode history', 'Disconnect', 'Read device ID and info', 'Write device time', 'Read device time', 'Read user ID and version data', 'Write new user ID', 'Read measurement setting', 'Write measurement setting', 'Read SN', and 'Read usual mode history data each measurement'. At the bottom, there's a 'Log' section displaying the following text:</p> <pre>DRecordUsual(mode=0, historyMeasurementNumber=250, mData=[MData(systole=120, diastole=80, hr=60, year=2023, month=1, day=1, hour=1, minute=1, option_arr=false, option_usu=true, option_dia=false, option_am=false, option_pm=false, option_afib=false), MData(systole=120, diastole=80, hr=60, year=2023, month=1, day=2, hour=2, minute=2, option_arr=false, option_usu=true, option_dia=false, option_am=false, option_pm=false, option_afib=false), MData(systole=120, diastole=80, hr=60, year=2023, month=1, day=3, hour=3, minute=3,</pre>	<ol style="list-style-type: none"> <li>1. The button “Read usual mode history” is to get usual mode history data.</li> <li>2. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from WBP via Bluetooth. The green part is the result after decoding with the raw data.</li> <li>3. The log “WBP : DRecordUsual(mode=0, historyMeasurementNumber=250, mData=[MData(systole=120, diastole=80, hr=60, year=2023, month=1, day=1, hour=1, minute=1, option_arr=false, option_usu=true, option_dia=false, option_am=false, option_pm=false, option_afib=false),... }]]” is included BP readings.</li> <li>4. Each BP reading has its own measurement date &amp; time, systolic, diastolic and pulse. For instance, the above-mentioned reading is DateTime 2023/1/1 01:01, Sys 120, Dia 80, Pulse 60.</li> </ol>
---	--

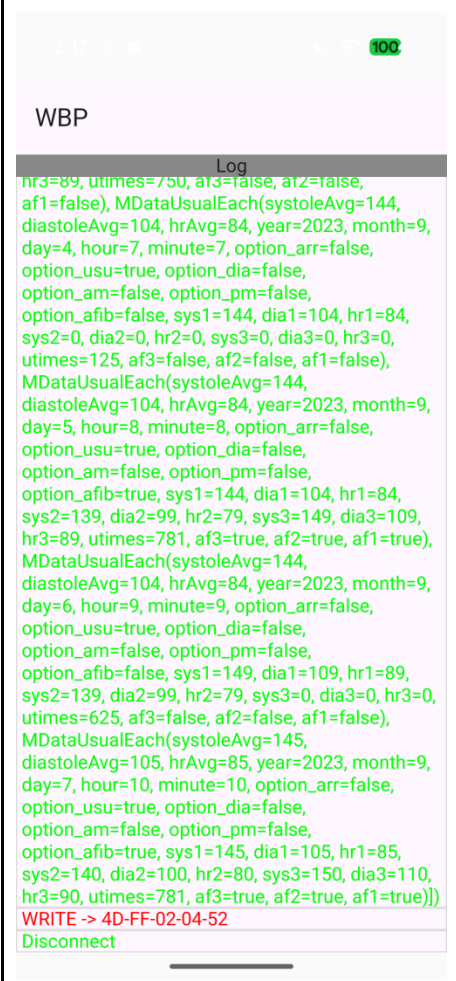
## 6.5 Command: Read usual mode history each measurement data from WBP:

 <p>WBP</p> <p>Read diagnostic mode history Clear current mode history</p> <p>Read usual mode history Clear selected mode history</p> <p>Disconnect Read device ID and info</p> <p>Write device time Read device time</p> <p>Read user ID and version data Write new user ID</p> <p>Read measurement setting Write measurement setting</p> <p>Read SN</p> <p>Read usual mode history data each measurement</p> <p>Log</p> <pre>2, 2, DRecordUsualEach(mode=null, historyMeasurementNumber=null, mDatumUsualEaches=[MDataUsualEach(systoleAvg=143, diastoleAvg=103, hrAvg=83, year=2023, month=8, day=27, hour=23, minute=59, option_arr=false, option_usu=true, option_dia=false, option_am=false, option_pm=false, option_afib=false, sys1=143, dia1=103, hr1=83, sys2=138, dia2=98, hr2=78, sys3=148, dia3=108, hr3=88, utimes=776, af3=true, af2=false, af1=true), MDataUsualEach(systoleAvg=144, diastoleAvg=104, hrAvg=84, year=2023, month=8, day=28, hour=0, minute=0, option_arr=false,</pre>	<ol style="list-style-type: none"> <li>1. The button “Read usual mode history each measurement” is to get usual mode each measurement history data.</li> <li>2. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from WBP via Bluetooth. The green part is the result after decoding with the raw data.</li> <li>3. The log “WBP : 2, 2, DRecordUsualEach(mode=null, historyMeasurementNumber=null, mDatumUsualEaches=[MDataUsualEach(systoleAvg=143, diastoleAvg=103, hrAvg=83, year=2023, month=8, day=27, hour=23, minute=59, option_arr=false, option_usu=true, option_dia=false, option_am=false, option_pm=false, option_afib=false, sys1=143, dia1=103, hr1=83, sys2=138, dia2=98, hr2=78, sys3=148, dia3=108, hr3=88, utimes=776, af3=true, af2=false, af1=true),... }]]” is included BP readings.</li> <li>4. Each BP reading has its own measurement date &amp; time, systolic, diastolic and pulse. For instance, the above-mentioned reading is DateTime 2023/8/27 23:59, systoleAvg 143, diastoleAvg 103, hrAvg 83.</li> </ol>
---	---

## 6.6 Command: Read diagnostic mode history from WBP

	<ol style="list-style-type: none"> <li>1. The button “Read diagnostic mode history” is to get diagnostic mode history data.</li> <li>2. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from WBP via Bluetooth. The green part is the result after decoding with the raw data.</li> <li>3. The log “WBP : DRecordDiagnostic(mode=0, historyMeasurementNumber=60, mDatumDiagnostics=[MData(systole=151, diastole=96, hr=61, year=2023, month=7, day=2, hour=7, minute=0, option_arr=false, option_usu=false, option_dia=true, option_am=true, option_pm=false, option_afib=true), ... ])” is included BP readings.</li> <li>4. Each BP reading has its own measurement date &amp; time, systolic, diastolic and pulse. For instance, the above-mentioned reading is DateTime 2023/7/2 07:00, Sys 151, Dia 96, Pulse 61.</li> </ol>
--	--

## 6.7 Command: Disconnect the Bluetooth

	<ol style="list-style-type: none"> <li>1. The button “DISCONNECT” is to terminate the connection of Bluetooth.</li> <li>2. The “DISCONNECT” sends a disconnected command to WBP, and then the disconnection will be executed by WBP.</li> </ol>
--	---

## Appendix

CurrentAndMData Structure / Parameter				
Property	Device	Type	Value	Description
AA=0	WBP	Integer	0, 1	Anti-artifact Detected
ABPM=0	WBP	Integer	0, 1	Ambulatory Blood Pressure Monitoring
AFIb=false	3G/4G	Boolean	false , true	Atrial Fibrillation Detection
AM=false	WBP	Boolean	false , true	Data measured at night with diagnostic mode (e.g., 4:00~12:00)
CPP=0	WBP	Integer	0 ~ 255	Central Pulse Pressure
CSBP=0	WBP	Integer	0 ~ 255	Carotid Systolic Blood Pressure
IHB=false	3G/4G	Boolean	false , true	Irregular Heartbeat Detection
LB=0	WBP	Integer	0, 1	Low Battery
MAM=0	3G/4G	Integer	0 ~ 3	Microlife Aerge Mode
MAP=0	WBP	Integer	0 ~ 255	Mean Arterial Pressure
MCBP=0	WBP	Integer	0 ~ 65535	Mean Central Blood Pressure
PM=false	WBP	Boolean	false , true	Data measured in the morning with diagnostic mode



				(e.g., 18:00~24:00)
PVR=0	WBP	Integer	0 ~ 65535	Pulse Variation Rate
SM=0,	WBP	Integer	0, 1	Start of a Manual Measurement
arr=false	3G/4G	Boolean	false , true	Detection of PAD or Afib
condition=0	WBP	Integer	0 ~ 24	Condition
Cuffokr=0	3G/4G	Integer	0, 1	Whether the wristband is tight: 3G detection mode: 2 - No detection, 4G detection mode: 0 - No tightness, 1 - There is tightness
day=7	WBP, 3G/4G	Integer	0 ~ 31	Date
deviceMode=58	WBP, 3G/4G	Integer	0x31 3G BPM, 0x3A 4G BPM, 0X51 WBP HomeA	Device type
dia=69	WBP, 3G/4G	Integer	0 ~ 255	Diastolic
diagnostic=false	WBP	Boolean	false , true	Data measured in diagnostic mode
errorCode=0	WBP	Integer	0 ~ 255	Error Code
hour=10	WBP, 3G/4G	Integer	0 ~ 24	Hour

hr=82	WBP, 3G/4G	Integer	0 ~ 255	Pulse
indexYear=0	WBP	Integer	0 ~ 99	offset Year
isFor3G=false	3G/4G	Boolean	false , true	is for 3G
minute=12	WBP, 3G/4G	Integer	0 ~ 59	Minute
month=2,	WBP, 3G/4G	Integer	1 ~ 12	Month
resultCode=0	WBP	Integer	0x01, 0x02, 0x03, 0x05, 0x42, 0x50, 0x51, 0x52	Result Code
systole=114	WBP, 3G/4G	Integer	0 ~ 255	Systolic
usual=false	WBP	Boolean	false , true	Data measured in usual mode
year=2024	WBP, 3G/4G	Integer	WBP: 2000~ 2050, 3G/ 4G: 2000 ~ 2048	Year