

Demo APP for MicroLifeDeviceSDK (Android)

Table of Contents

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

Chapter1 Development Environment

1.1 The supported SDK version is as follow:

```
compileSdkVersion 26
buildToolsVersion '26.0.3'

defaultConfig {
    minSdkVersion 19
    targetSdkVersion 26
    versionCode 1
    versionName "1.3"
}
```

1.2 Add the library “sdk-release.aar” into the “libs” directory.

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

```
compile(name:'sdk-release', ext:'aar')
compile(name:'scalelesdk-v1.4.0', ext:'aar')
```

Chapter2 Entry Point and Bluetooth LE Protocol

The “ChoseActivity” is the entry point of the sample application. The “WBPTestActivity” is dedicated to the device WatchBP Home A (Bluetooth LE).

```

<activity
    android:name=".BPMTestActivity"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden" />
<activity
    android:name=".WeightTestActivity"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden" />
<activity
    android:name=".BtTestActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".WBPTestActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".ChoseActivity"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".ConnectionActivity">

```

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

```

//Initialize the connection SDK
Global.wbpProtocol = WBPProtocol.getInstance
    ( aty: this, isSimulation: false, isPrintLog: true, Global.sdkid_WBP);
Global.wbpProtocol.setOnConnectStateListener(this);
Global.wbpProtocol.setOnDataResponseListener(this);
Global.wbpProtocol.setOnNotifyStateListener(this);
Global.wbpProtocol.setOnWriteStateListener(this);

```

2.1.1 The “setOnConnectStateListener()” is to get the connection status of device.

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

2.1.2 The “setOnNotifyStateListener()” is to get the data which is response from device.

2.1.3 The “setOnWriteStateListener()” is to get the data which is sent to device.

2.2 The “isEnabledBt()” or “ isSupportBluetooth()” is to check if the smartphone’s Bluetooth is enabled or not. The “isSupportBluetooth()” will prompt a warning message to inform user to turn on Bluetooth if it is disabled.

Chapter3 WatchBP APIs

3.1. Instance of Bluetooth LE Protocol :

3.1.1. Interface :

	public static *Protocol getInstance(Activity aty, boolean isSimulation, boolean isPrintLog, String sdkid)
Definition	Initialize Bluetooth LE Protocol for WatchBP Home A device
Parameter	Activity aty : name of activity or this boolean isSimulation : is simulator or device boolean isPrintLog : is printing log or not String sdkid : SDK ID of designated device
	<pre>//Initialize the connection SDK Global.bpmProtocol = BPMProtocol.getInstance (aty: this, isSimulation: false, isPrintLog: true, Global.sdkid);</pre>

3.2. Connection State and Result :

3.2.1. Interface :

	public void setOnConnectStateListener(OnConnectStateListener l)
Definition	The “setOnConnectStateListener()” is to get the connection status of device.

3.2.2. Delegate :

	void onBtStateChanged(boolean isEnabled)
Definition	The “onBtStateChanged()” is to monitor the state of Enabled or Disabled.

	void onScanResult(String mac, String name, int rssi)
Definition	This is to get Bluetooth information of devices which discovered in the vicinity.
Parameter	macAddress: MAC of device name: device name RSSI: RSSI

	void onConnectionState(ConnectState state)
Definition	The “onConnectionState()” is to monitor the status of connection.
Parameter	<pre>public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] }</pre>

3.3. Device scanning or discovery :

3.3.1. Interface :

	public void startScan(int timeout)
Definition	The “startScan()” is for device scanning or discovery. The result will be shown with the “onScanResult”.
Parameter	int timeout

	public void stopScan()
Definition	Terminate the scanning process.

3.3.2. Delegate :

	void onConnectionState(ConnectState state)
Definition	The “onConnectionState()” is to monitor the status of scanning.
Parameter	<pre> public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] } </pre>

3.4. Connection :

3.4.1. Interface :

	public void connect(String macAddress)
Definition	Connect to device with MAC address.
Parameter	macAddress: MAC of device

3.4.2. Delegate :

	void onConnectionState(ConnectState state)
Definition	The “onConnectionState()” is to monitor the status of connection.
Parameter	<pre> public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] } </pre>

3.5. Bonding :

3.5.1. Interface :

	public void bond(String macAddress)
Definition	Binding specified device by MAC
Parameter	macAddress: MAC of device

3.5.2. Delegate :

	void onConnectionState(ConnectState state)
--	--

Definition	The “onConnectionState()” is to monitor the status of connection.
Parameter	<pre>public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] }</pre>

3.6. Disconnection :

3.6.1. Interface :

	public void disconnect()
Definition	Disconnect device.

3.6.2. Delegate :

	void onConnectionState(ConnectState state)
Definition	The “onConnectionState()” is to monitor the status of disconnection.
Parameter	<pre>public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] }</pre>

3.7. Read usual mode history data from BPM :

3.7.1. Interface :

	public void readUsualModeHistoryData()
Definition	Read usual mode history data from BPM

3.7.2. Delegate :

	void onResponseReadUsualModeHistory(DRecord dRecord,boolean isNullData)
Parameter	data : usual mode history data isNullData : is Null Data

3.8. Read diagnostic mode history data from BPM :

3.8.1. Interface :

	public void readDiagnosticModeHistoryData()
Definition	Read diagnostic mode history data from BPM

3.8.2. Delegate :

	void onResponseReadDiagnosticModeHistory(DiagnosticDRecord dRecord,boolean isNullData)
Parameter	dRecord : diagnostic mode history data isNullData : is Null Data

3.9. Clear selected mode history data of the BPM :

3.9.1. Interface :

	public void clearHistoryData(boolean clearUsualMode,boolean clearDiagnosticlMode,boolean clearNocturnalMode)
Definition	Clear selected mode history data of the BPM
Parameter	clearUsualMode : clear Usual Mode clearDiagnosticlMode : clear Diagnosticl Mode clearNocturnalMode : clear Nocturnal Mode

3.9.2. Delegate :

	void onResponseClearSelectedModeHistory(boolean isSuccess)
Parameter	isSuccess : True or False

3.10. Clear current mode history data of the BPM :

3.10.1. Interface :

	public void clearCurrentModeHistoryData()
Definition	Clear current mode history data of the BPM

3.10.2. Delegate :

	void onResponseClearSelectedModeHistory(boolean isSuccess)
Parameter	isSuccess : True or False

3.11. Disconnect the Bluetooth with BPM :

3.11.1. Interface :

	public void disconnectWBP()
Definition	Disconnect the Bluetooth with BPM

3.11.2. Delegate :

	void onConnectionState(ConnectState state)
Definition	The "onConnectionState()" is to monitor the status of connection with WBP device.

Parameter	<pre>public enum ConnectState { ScanFinish, //Scan finish Connected, //Connect success Disconnect, //Disconnect ConnectTimeout, //Connection timeout ScaleWake, //Scale Wake [EBodyProtocol limited] ScaleSleep //Scale Sleep [EBodyProtocol limited] }</pre>
-----------	---

3.12. Write device Time to BPM :

3.12.1. Interface :

	public void writeDeviceTime()
Definition	Write device Time to BPM

3.12.2. Delegate :

	void onResponseWriteDeviceTime(boolean isSuccess)
Parameter	isSuccess : True or False

3.13. Write a new user ID to BPM :

3.13.1. Interface :

	public void writeUserID(String ID)
Definition	Write a new user ID to BPM
Parameter	ID : User ID ◦

3.13.2. Delegate :

	void onResponseWriteUserID(boolean isSuccess)
Parameter	isSuccess : True or False

3.14. Read nocturnal mode setting :

3.14.1. Interface :

	public void readNocturnalModeSetting()
Definition	readNocturnalModeSetting

3.14.2. Delegate :

	void onResponseReadNocturnalModeSetting(DeviceInfo deviceInfo)
Parameter	deviceInfo : nocturnal mode setting

3.15. Change nocturnal mode setting :

3.15.1. Interface :

	public void changeNocturnalModeSetting(boolean open,int year,int month,int day,int hour)
Definition	Change nocturnal mode setting Note: This command requires matching hardware to set. You can use "readDeviceIDAndInfo(MicroLifeDeviceInfo.openNocturnalMod

	e)" to check if the device supports it.
Parameter	open : Nocturnal ON/OFF year : year month : month day : day hour : hour

3.15.2. Delegate :

	void onResponseChangeNocturnalModeSetting(boolean isSuccess)
Definition	Change nocturnal mode setting Note: This command requires matching hardware to set. You can use "readDeviceIDAndInfo(MicroLifeDeviceInfo.openNocturnalMode)" to check if the device supports it.
Parameter	isSuccess : True or False

3.16. Erase all of device measure & error times :

3.16.1. Interface :

	public void eraseAllOfDeviceMeasureAndErrorTimes()
Definition	Erase all of device measure & error times

3.16.2. Delegate :

	void onResponseEraseAllOfDeviceMeasureAndErrorTimes(boolean isSuccess)
Parameter	isSuccess : True or False

3.17. Read device ID and info from BPM :

3.17.1. Interface :

	public void readDeviceIDAndInfo()
Definition	Read device ID and info from BPM

3.17.2. Delegate :

	void onResponseReadDeviceInfo(DeviceInfo deviceInfo)
Parameter	deviceInfo : device ID and info

3.18. Read device Time from BPM :

3.18.1. Interface :

	public void readDeviceTime()
Definition	Read device Time from BPM

3.18.2. Delegate :

	void onResponseReadDeviceTime(DeviceInfo deviceInfo)
--	--

Parameter	deviceInfo : device Time
-----------	--------------------------

3.19. Read user ID and version data from BPM :

3.19.1. Interface :

	public void readUserAndVersionData()
Definition	Read user ID and version data from BPM

3.19.2. Delegate :

	void onResponseReadUserAndVersionData(User user, VersionData versionData)
Parameter	user : user ID verData : version data

3.20. Read Nocturnal mode history data from BPM :

3.20.1. Interface :

	public void readNocturnalModeHistoryData()
Definition	Read Nocturnal mode history data from BPM

3.20.2. Delegate :

	void onResponseReadNocturnalPatternHistory(NocturnalModeDRecord dRecord,boolean isNullData)
Parameter	dRecord : Nocturnal pattern history data isNullData : is Null Data

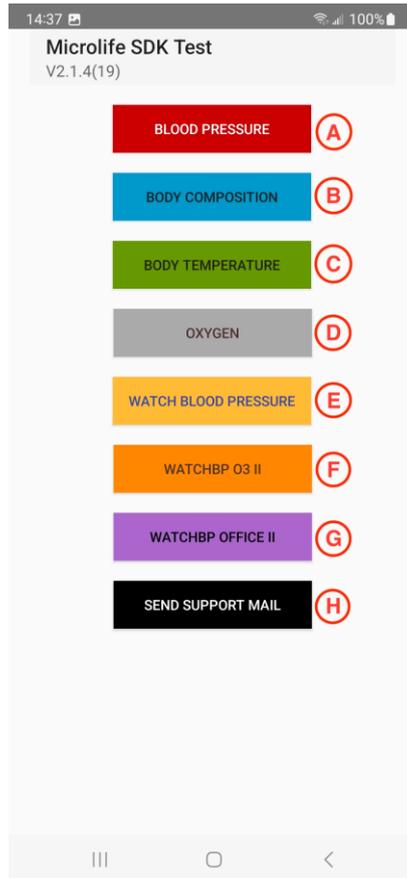
Chapter4 User Interface of Demo App

4.1. Version:

4.1.1. Demo App version as V2.1.4, the (19) is build count.

4.2. Getting Started :

Start the app and then select the button “WATCH Blood Pressure ” / “E” to communicate with the designate device WatchBP Home A.

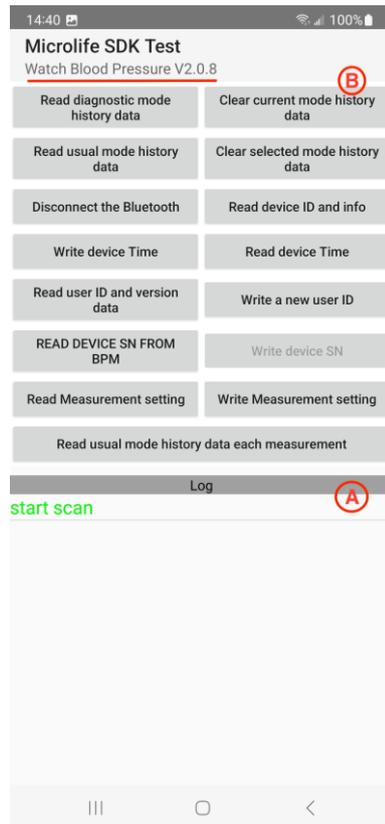


4.3. Operation Sequence :

4.3.1. The scanning (discovery) is automatically run to discover devices in the vicinity.

4.3.2. If a device is bonded, it will be connected accordingly. If not, the “bindingDevice” can be used to run bonding process.

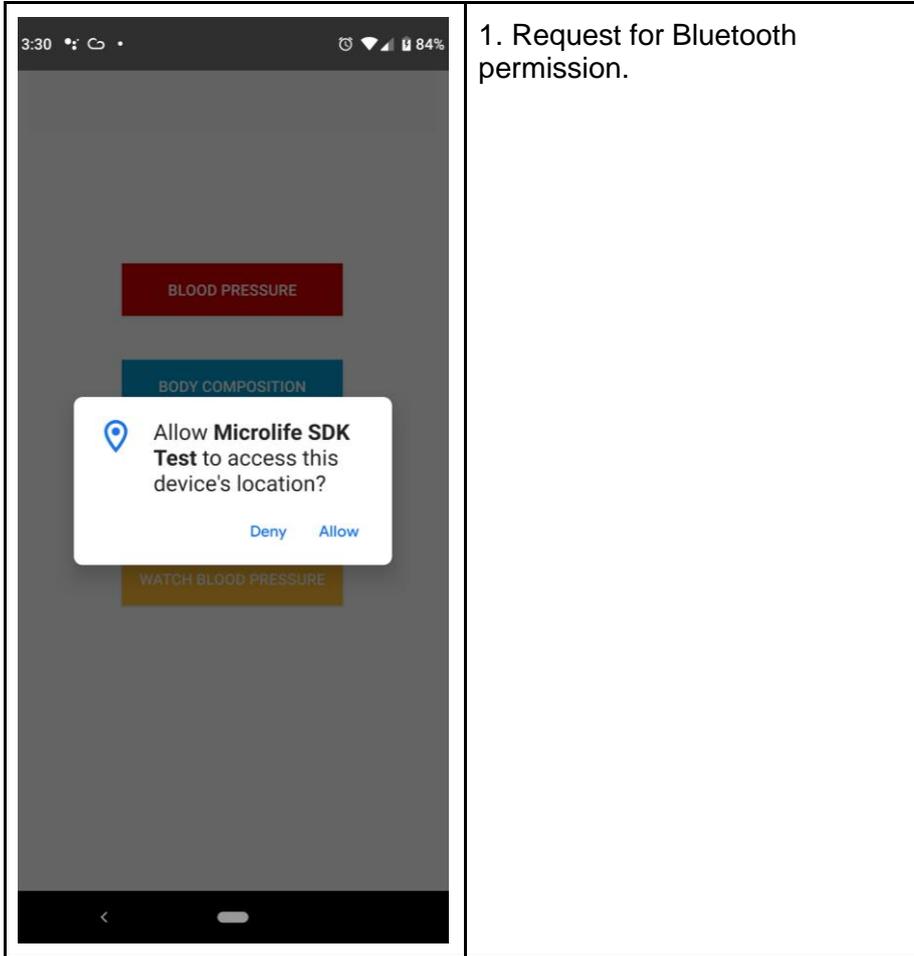
4.4. Operating Interface and Sequence :



- 4.4.1. SDK version : V2.0.8
- 4.4.2. Region A : The log window is used to display information about communication handshake between App and device.
- 4.4.3. Region B : This part is to communicate with the device WatchBP Home A by different functions / commands such as data transferring, synchronization and so on.
- 4.4.4. Refer to “WBPTTestActivity” from the demo application (sample code) to get more detailed.

Chapter5 Functionality of Demo App

5.1. Bluetooth authorization :



1. Request for Bluetooth permission.

5.2. Pairing / Bonding :

	<ol style="list-style-type: none"> 1. There is a message to confirm the pairing bonding procedure between device and cellphone if they haven't bonded yet. 2. Once the procedure is done, choose any function/ command to do communication with WatchBP Home A device. 3. The green part is from "onScanResult".

5.3. Command: Write a new user ID to BPM

The screenshot shows the 'Microlife SDK Test' application interface. At the top, it says 'Watch Blood Pressure'. Below this is a grid of buttons for various functions: 'READ USUAL MODE HISTORY DATA', 'READ DIAGNOSTIC MODE HISTORY DATA', 'CLEAR SELECTED MODE HISTORY DATA', 'CLEAR CURRENT MODE HISTORY DATA', 'DISCONNECT THE BLUETOOTH', 'WRITE DEVICE TIME', 'WRITE A NEW USER ID', 'READ NOCTURNAL MODE SETTING', 'CHANGE NOCTURNAL MODE SETTING', 'ERASE ALL OF DEVICE MEASURE AND ERROR TIMES', 'READ DEVICE ID AND INFO', 'READ DEVICE TIME', 'READ USER ID AND VERSION DATA', and 'READ NOCTURNAL MODE HISTORY DATA'. Below the buttons is a 'Log' window displaying the following text:

```

year=0, month=0, day=0, hour=0, minute=0,
second=0}
NOTIFY : 4D511A0B44E8A7A34299E3424D00
000301000002
NOTIFY : 000003000005000094
WRITE : Write a new user ID
WRITE : writeUserID : 029331549LQ
WRITE : 4DFF1806003032393333313534394C
5100000000
WRITE : 000000000000DB
onResponseWriteUserID : true
NOTIFY : 4D5103068128
    
```

1. The command “Write a new user ID” is to write a new user ID to device.

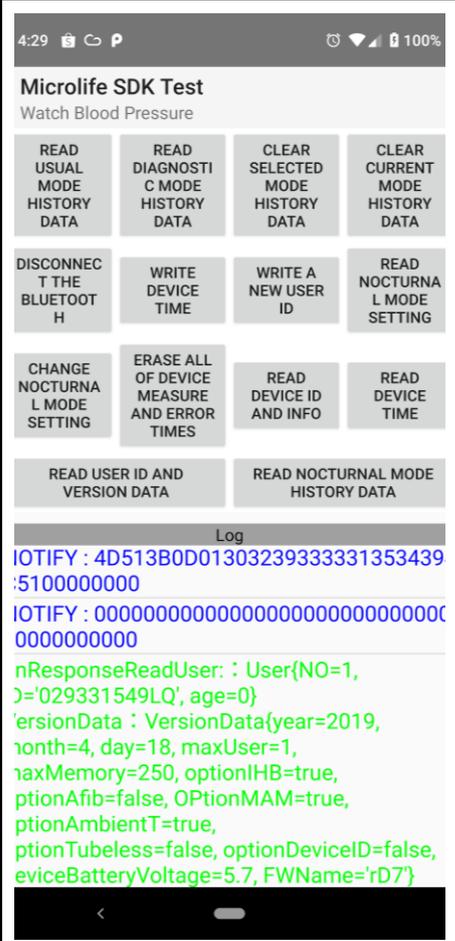
2. WRITE :writeUserID : 029331549LQ
The ID “029331549LQ” is made up of ASCII code.

3.onResponseWriteUserID : true :
This means that the writing/ sending procedure is successful.

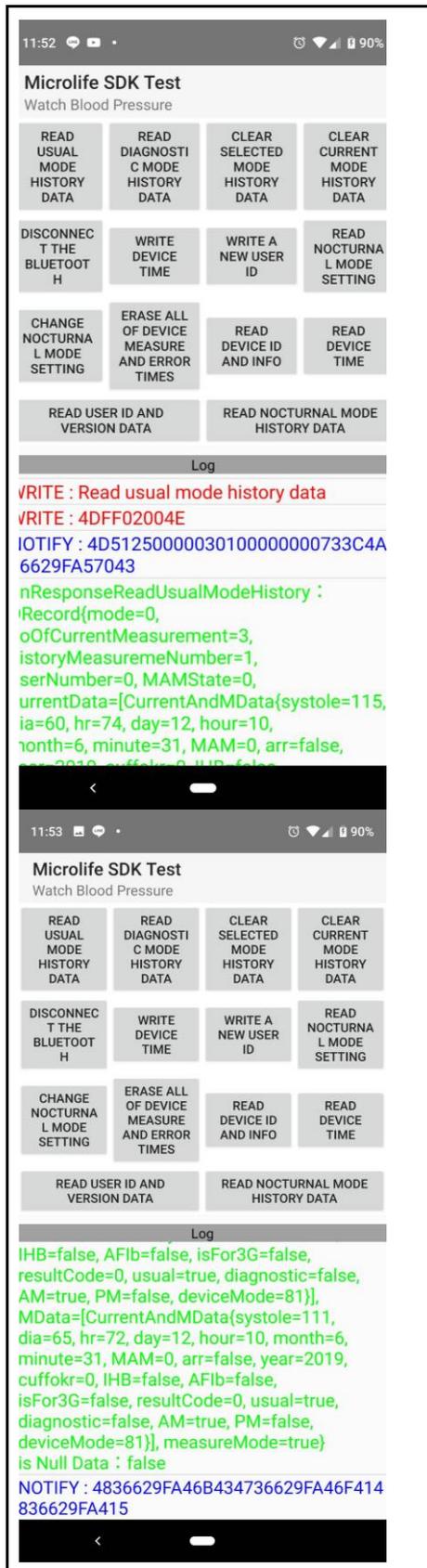
4. It can be verified by utilizing the command “Read user ID and version data”.

5. The **red** part is the command and communication protocol that is sent to device. The **blue** part is the raw data from device via Bluetooth. The **green** part is from “onResponseWriteUserID” which is decoded from the raw data.

5.4. Command: Read user ID and version data from BPM

	<ol style="list-style-type: none"> 1. The command “Read user ID and version data” is to get user ID and device information as below. 2.onResponseReadUser: : User{NO=1, ID='029331549LQ', age=0} VersionData : VersionData{year=2019, month=4, day=18, maxUser=1, maxMemory=250, optionIHB=true, optionAfib=false, optionMAM=true, optionAmbientT=true, optionTubeless=false, optionDeviceID=false, deviceBatteryVoltage=5.7, FWName='rD7'} 3.ID = user ID 4.VersionData = Device information
--	---

5.5. Command: Read usual mode history data from BPM



1. This is to get usual mode history data.
2. onResponseReadUsualModeHistory :
`DRecord{mode=0, noOfCurrentMeasurement=3, historyMeasuremeNumber=1, userNumber=0, MAMState=0, currentData=[CurrentAndMData{systole=115, dia=60, hr=74, day=12, hour=10, month=6, minute=31, MAM=0, arr=false, year=2019, cuffokr=0, IHB=false, AFib=true, isFor3G=false, resultCode=0, usual=true, diagnostic=false, AM=true, PM=false, deviceMode=81}, CurrentAndMData{...}], MData=[CurrentAndMData{systole=111, dia=65, hr=72, day=12, hour=10, month=6, minute=31, MAM=0, arr=false, year=2019, cuffokr=0, IHB=false, AFib=false, isFor3G=false, resultCode=0, usual=true, diagnostic=false, AM=true, PM=false, deviceMode=81}], measureMode=true}`
3. MData = The history measurement times store in memory.
4. currentData = The current measurement times store in memory.

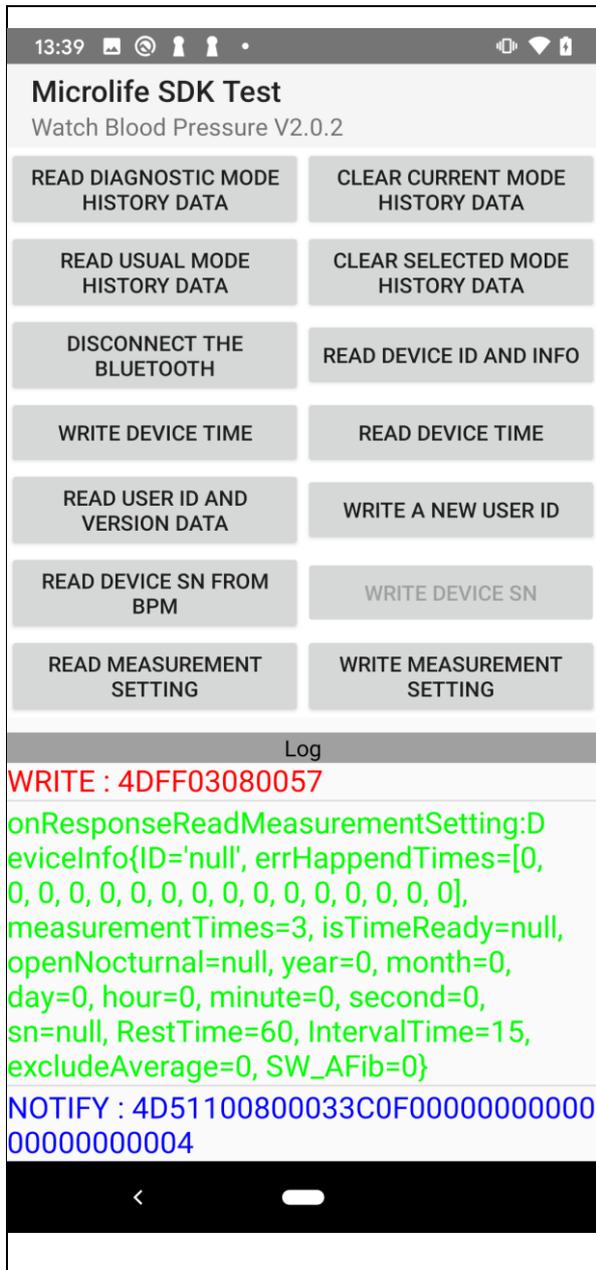
5.6. Command: Read usual mode history data each measurement

	<ol style="list-style-type: none"> 1. This is to get usual mode history data with 3 times each measurements. 2. The command is “4DFF0300F140” and it is sent to device (readUsualModeHistoryDataEachMeasurement()). 3.onResponseReadUsualModeHistoryEach: <pre>List<EachMData> = [EachMData{SYSTOLEavg=120, DIATOLEavg=80, HRavg=60, arr=false, usual=true, diagnostic=false, AM=false, PM=false, AFib=false, year=2022, month=1, day=1, hour=1, minute=1, sys1=120, dia1=80, hr1=60, sys2=115, dia2=75, hr2=55, sys3=125, dia3=85, hr3=65, uTimes=3, af3=false, af2=false, af1=true}, ...]</pre>
--	---

5.7. Command: Disconnect the Bluetooth

	<ol style="list-style-type: none"> 1. Disconnect the Bluetooth 2. The command is “4DFF020452” and it is sent to device. 3. The result is able to observe by the “onConnectionState(ConnectState state)” and its status can be found by the “ConnectState = Disconnect”.
--	--

5.8. Command: Read/Write Measurement Setting



1. The command “Read Measurement Setting” is to get WatchBP Home A device setting information as below.

2. onResponseReadMeasurementSetting: DeviceInfo{ID='null', errHappendTimes=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], measurementTimes=3, isTimeReady=null, openNocturnal=null, year=0, month=0, day=0, hour=0, minute=0, second=0, sn=null, RestTime=0, IntervalTime=15, excludAverage=0, SW_AFib=1}

	<p>3.The command “Write Measurement Setting” is to write new settings for WatchBP Home A device as below.</p> <p>4. WRITE: mTimes: 3, reset: 60, interval: 15, eAverage: 0, swAfib: 0</p> <p>5. onResponseWriteMeasurementSetting : true This means that the writing/ sending procedure is successful.</p> <p>4. It can be verified by utilizing the command “Read Measurement Setting”.</p>
--	--