Demo APP for MicroLifeDeviceSDK (WatchBP Office) - iOS

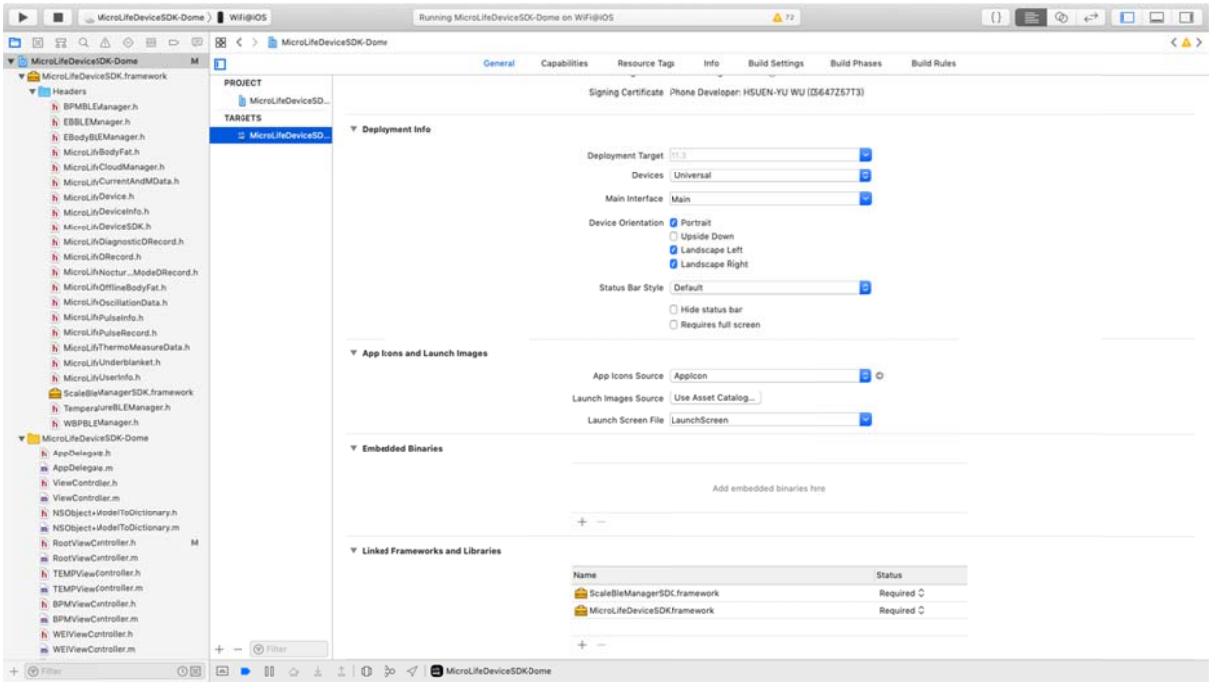
Table of Contents

Chapter 1 Development Environment

This user manual serves as a quick introduction to MicroLifeDeviceSDK / APIs and shows how to integrate into an iOS Demo App.

1.1. First of all, add MicroLifeDeviceSDK.framework into a development project. The minimum supported version is iOS 10. The compatible version of Xcode IDE is 10.2.1.

1.2. Under TARGETS / General / Linked Frameworks and Libraries, add MicroLifeDeviceSDK.framework.



1.3. Under TARGETS / Build Settings / Enable, set Bitcode to NO.

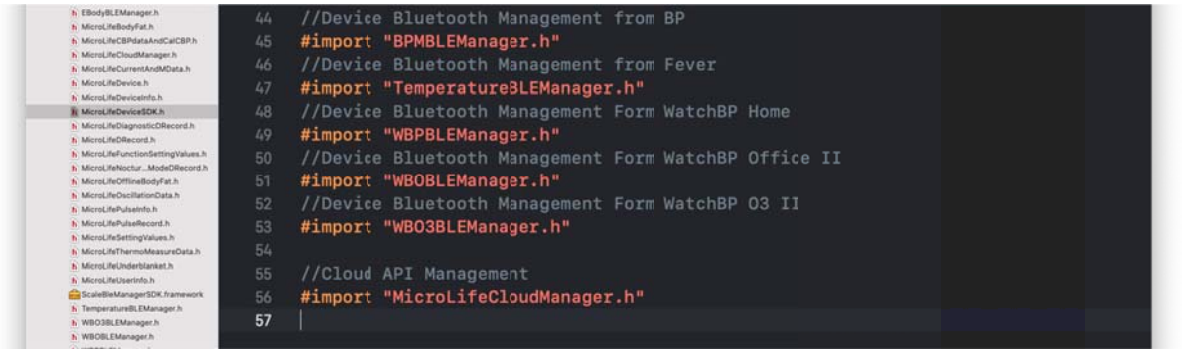
1.4. Under TARGETS / Build Settings / Other Linkeder Flags, use the linker flag “-all_load”.

1.5. Import header file as bellows.

```
#import <MicroLifeDeviceSDK/MicroLifeDeviceSDK.h>
```

Chapter 2 Entry Point and Bluetooth LE Protocol (BLEManager)

The BLEManager is responsible for managing the Bluetooth communication. Refer to the file "MicroLifeDeviceSDK/MicroLifeDeviceSDK.h" for suitable device models. They are named like "XXXBLEManager".



2.1. Initiate a BLEManage and register the delegation of BLEManager. The DataResponseDelegate is dedicated for response messages of BLEManager.

2.2. Once the BLEManager is initiated, the BLEManager checks the status of Bluetooth by BLEManagerCellPhoneBluetoothDidUpdateState.

2.3. While turning on Bluetooth, BLEManager runs the scan/ discovery procedure automatically by BLEManagerDidDiscoverBluetoothDeviceMacAddress to get devices' information in the vicinity.

2.4. Call the bindingDevice:(NSData *)macAddress to bind an activated device. The status of binding can be found by the following section 2.5. The BLEManager will automatically connect to a bound device while it is activated.

2.5. Connection status：

2.5.1. Connection / binding：

- (void) * BLEManagerDidConnectDevice;

2.5.2. Fail of connection：

- (void) * BLEManagerDidFailToConnectDevice;

2.5.3. Disconnection：

- (void) * BLEManagerDidDisconnectDevice;

Chapter 3 APIs of BLEManager

By utilizing the APIs of BLEManager, can transfer data between device and App via Bluetooth. The APIs includes two parts: one is for fundamental Bluetooth and another is dedicated for communication with a designated device / BPM (Blood Pressure Monitor) with specific commands and protocol.

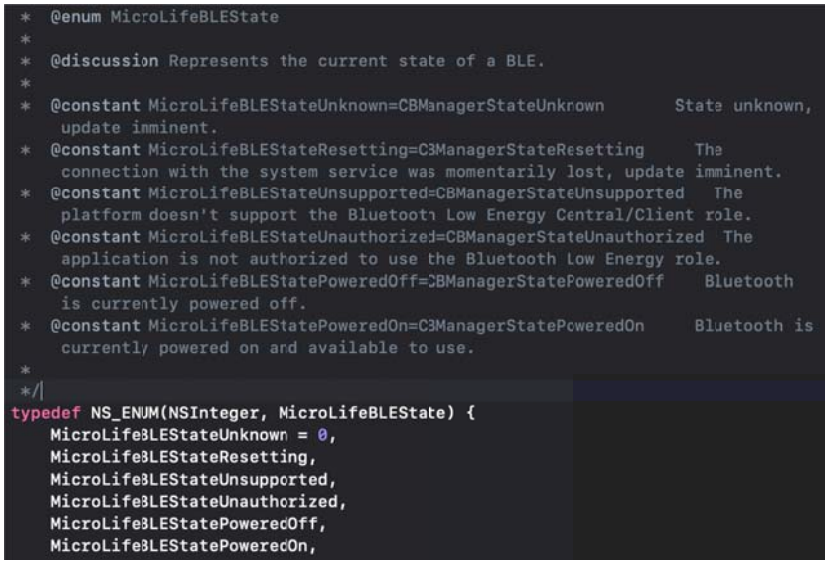
3.1. Device searching :

3.1.1.

	+(instancetype)shareInstanceWhithAuthorizationkey:(NSString *)key Target * Names:(NSArray *)target * Names
Definition	Searching for device with default / customized name
Parameter	Key: Authorization code target * Names: device name which can be included single or multiple
	`self.aBPMBLEManager = [BPMBLEManager shareInstanceWhithAuthorizationkey:SDKkey TargetBPMNames:@[@"A6 BT",@"A6 BASIS PLUS BT",@"A7 TOUCH BT",@"B3 BT",@"B6 Connect",@"A6",@"Progress"]];` `self.aBPMBLEManager.dataResponseDelegate = self;`

3.1.2.

	- (void) * BLEManagerCellPhoneBluetoothDidUpdateState:(MicroLifeBLEState)state;
Definition	Response for the shareInstanceWhithAuthorizationkey (). This is to manage Bluetooth status of cellphone.
Parameter	status: Bluetooth status of cellphone

```
 *  @enum MicroLifeBLEState
 *
 *  @discussion Represents the current state of a BLE.
 *
 *  @constant MicroLifeBLEStateUnknown=CBManagerStateUnknown       State unknown,
     update imminent.
 *  @constant MicroLifeBLEStateResetting=CBManagerStateResetting     The
     connection with the system service was momentarily lost, update imminent.
 *  @constant MicroLifeBLEStateUnsupported=CBManagerStateUnsupported   The
     platform doesn't support the Bluetooth Low Energy Central/Client role.
 *  @constant MicroLifeBLEStateUnauthorized=CBManagerStateUnauthorized  The
     application is not authorized to use the Bluetooth Low Energy role.
 *  @constant MicroLifeBLEStatePoweredOff=CBManagerStatePoweredOff    Bluetooth
     is currently powered off.
 *  @constant MicroLifeBLEStatePoweredOn=CBManagerStatePoweredOn     Bluetooth is
     currently powered on and available to use.
 *
 */
typedef NS_ENUM(NSInteger, MicroLifeBLEState) {
    MicroLifeBLEStateUnknown = 0,
    MicroLifeBLEStateResetting,
    MicroLifeBLEStateUnsupported,
    MicroLifeBLEStateUnauthorized,
    MicroLifeBLEStatePoweredOff,
    MicroLifeBLEStatePoweredOn,
```

	- (void) * BLEManagerDidDiscoverBluetoothDeviceMacAddress:(NSData *)macAddress Name:(NSString *)name RSSI:(NSNumber *)RSSI;
Definition	Response for the shareInstanceWhithAuthorizationkey (). This is to manage information of devices that are discovered in the vicinity.
Parameter	macAddress: MAC address of device name: device name RSSI: RSSI

3.2. Binding / Pairing：

3.2.1.

	- (void)bindingDevice:(NSData *)macAddress
Definition	Binding a specified device by MAC
Parameter	macAddress: MAC address of device

3.2.2.

	- (void) * BLEManagerDidConnectDevice;
Definition	Response for the bindingDevice() with the status Connection

	- (void) * BLEManagerDidFailToConnectDevice
Definition	Response for the bindingDevice() with the status Fail of Connection

	- (void) * BLEManagerDidDisconnectDevice;
Definition	Response for the bindingDevice() with the status Disconection

3.3. MAC address of binding device：

3.3.1.

	- (NSData *)getBindingDevice;
Definition	Get MAC address of a binding device or nil.

3.4. Unbinding：

3.4.1.

	- (void)unBindingDevice;
Definition	Unbinding device

3.4.2.

	- (void) * BLEManagerDidFailToConnectDevice;
Definition	Response for the unBindingDevice()

3.5. Disconnection：

3.5.1.

	- (void)disconnectDevice;
Definition	Disconnect with device

3.5.2.

	- (void) * BLEManagerDidFailToConnectDevice;
Definition	Response for the disconnectDevice ()

3.6. Searching for devices ：

3.6.1.

	- (void)reScan;
Definition	Search for devices repeatedly

3.6.2.

	- (void) * BLEManagerDidDiscoverBluetoothDeviceMacAddress:(NSData *)macAddress Name:(NSString *)name RSSI:(NSNumber *)RSSI;
Definition	This is to get information of devices which are discovered in the vicinity.
Parameter	macAddress: MAC of device name: device name RSSI: RSSI

3.7. Stop searching for devices：

3.7.1.

	- (void)stopScan;
Definition	Stop searching

3.7.2. Null.

Chapter 4 WatchBP Office APIs

4.1. Read all history or current data：

4.1.1. Interface：

	-(void)readAllHistorys;
Definition	Read all history or current data

4.1.2. Delegate：

	-(void)WBOBLEManagerResponseReadAllHistorys:(MicroLifeDRecord *)data;
Parameter	data：History data

4.2. Read central BP memory data by index：

4.2.1. Interface：

	- (void)readCBPDataWithIndex:(int)index Dformat:(Dformat)dformat;
Definition	4.2. Read central BP memory data by index
Parameter	index：CBP memory index dformat：Data format NoCBPRaw：No CBP raw data LowCBPRaw：low resolution CBP data FullCBPRaw：full CBP raw data

4.2.2. Delegate：

	-(void)WBOBLEManagerResponseReadCBPData:(MicroLifeCBPdataAndCalCBP *)data IsNoData:(BOOL)isNoData;
Parameter	cRecord：CBP data and CalCBP data isNullData：True or False

4.3. Clear all history data：

4.3.1. Interface：

	- (void)clearAllHistorys;
Definition	Clear all history data

4.3.2. Delegate：

	- (void)WBOBLEManagerResponseClearHistory:(BOOL)isSuccess;
Parameter	isSuccess：True or False

4.4. Disconnect the Bluetooth device：

4.4.1. Interface：

	- (void)disconnect;
Definition	Disconnect the Bluetooth device

4.4.2. Delegate：

	- (void)WBOBLEManagerDidDisconnectDevice;

4.5. Read user ID and version data

4.5.1. Interface：

	- (void)readUserAndVersionData;
Definition	Read user ID and version data

4.5.2. Delegate：

	- (void)WBOBLEManagerResponseReadUserAndVersionData:(MicroLifeUserInfo *)user VersionData:(MicroLifeDeviceInfo *)verData;
Parameter	user：user ID verData：version data

4.6. Write user ID ：

4.6.1. Interface：

	- (void)writeUserID:(NSString *)ID;
Definition	Write user ID to device
Parameter	ID：User ID。

4.6.2. Delegate：

	- (void)WBOBLEManagerResponseWriteUserID:(BOOL)isSuccess;
Parameter	isSuccess：True or False

4.7. Read BPM setting values：

4.7.1. Interface：

	- (void)readSettingValues;
Definition	Read BPM setting values

4.7.2. Delegate：

	- (void)WBOBLEManagerResponseReadSettingValue:(MicroLifeSettingValues *)settingValues;
Parameter	settingValues： BPM setting values

4.8. Write BPM setting values：

4.8.1. Interface：

	- (void)writeSettingValues:(MicroLifeSettingValues *)settingValues;
Definition	Write BPM setting values to device
Parameter	settingValues：BPM setting values

4.8.2. Delegate：

	- (void)WBOBLEManagerResponseWriteSettingValues:(BOOL)isSuccess;
Parameter	isSuccess：True or False

4.9. Read device ID and info ：

4.9.1. Interface：

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info

4.9.2. Delegate：

	- (void)WBOBLEManagerResponseReadDeviceIDAndInfo:(MicroLifeDeviceInfo *)deviceInfo;
Parameter	deviceInfo：Device ID and info

4.10. Read device time ：

4.10.1. Interface：

	- (void)readDeviceTime;
Definition	Read device time

4.10.2. Delegate：

	- (void)WBOBLEManagerResponseReadDeviceTime:(MicroLifeDeviceInfo *)deviceInfo;
Parameter	deviceInfo：Device Time

4.11. Write device Time

4.11.1. Interface：

	- (void)writeDeviceTime;
Definition	Write device Time to device

4.11.2. Delegate：

	-(void)WBOBLEManagerResponseWriteDeviceTime:(BOOL)isSuccess;
Parameter	isSuccess：True or False

4.12. Read BPM function setting value：

4.12.1. Interface：

	- (void)readFunctionSettingValue;
Definition	Read BPM function setting value

4.12.2. Delegate：

	-(void)WBOBLEManagerResponseReadFunctionSettingValue:(MicroLifeFunctionSettingValues *)functionSettingValues;
Parameter	functionSettingValues：BPM function setting value

4.13. Read BT module name：

4.13.1. Interface：

	- (void)readBTModuleName;
Definition	Read BT module name of device

4.13.2. Delegate：

	-(void)WBOBLEManagerResponseReadBTModuleName:(NSString *)BTModuleName;
Parameter	BTModuleName：BT Module Name

4.14 Start/Stop remote measurement：

4.14.1 Interface：

	- (void)startRemoteMeasurementWhtihCBPFunction:(Dformat)dformat;
Definition	Start remote measurement
Parameter	dformat：Data format which BPM sent out. It is same as what APP requested. NoCBPRaw：No CBP raw data LowCBPRaw：low resolution CBP data (sampling rate =16Hz) FullCBPRaw：full CBP raw data (sampling rate=256Hz)

	- (void)stopRemoteMeasurement;
Definition	Stop remote measurement

4.14.2 Delegate：

	- (void)WBOBLEManagerResponseStartRemoteMeasurement:(Dformat)D_format;
Definition	Start remote measurement
Parameter	dformat：Data format which BPM sent out. It is same as what APP requested. NoCBPRaw：No CBP raw data LowCBPRaw：low resolution CBP data (sampling rate =16Hz) FullCBPRaw：full CBP raw data (sampling rate=256Hz)

	- (void)WBOBLEManagerResponseremoteMeasurementStatusEvery5secondsWithSTATUS:(STATUS)status MeasurementNumber:(int)measurementNumber TotalMeasurementNumber:(int)totalMeasurementNumber Countdown:(int)countdown TotalMeasuretime:(int)totalMeasuretime;

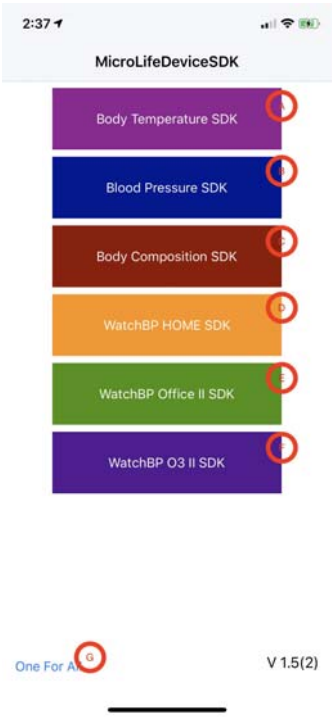
Definition	Send remote measurement status every 5 seconds
Parameter	status： MeasurementWait：device wait countdown for next measurement. MeasurementStart：device is start BP measurement, MeasurementStop：manual press I/O to stop measurement measurementNumber：Send current measurement number in auto mode. totalMeasurementNumber：Send total measurement number in auto mode. countdown：Send current countdown time in auto mode. totalMeasuretime：Send total measurement time (seconds) in auto mode. Total measurement are count between 1st measurement to last measurement. (exclude rest time)

	- (void)WBOBLEManagerResponseMeasurementResultsForEachMeasurement:(MicroLifeCurrentAndMData *)data HistoryMeasurementNumber:(int)historyMeasuremeNumber CurrentMeasurementTimes:(int)currentMeasurementTimes AverageCalculationWhenMeasurement:(BOOL)isAverage;
Definition	Send measurement results for each measurement
Parameter	dRecord：CurrentAndMData historyMeasuremeNumber：The history measurement times store in memory. currentMeasurementTimes：Send current measurement times to APP. isAverage：Send Average calculation when measurement to APP.

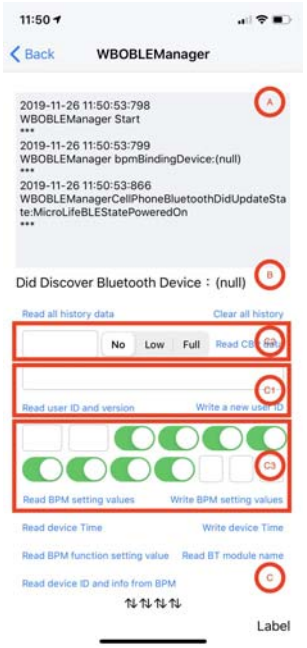
Chapter 5 User Interface & Functionality of WatchBP Office

5.1. Getting Started :

Start the app and then select the button “WatchBP Office II SDK” / “E” to communicate with the designate device.



5.2. Operating Interface and Sequence：



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

5.2.2. Region B：The display is for current device with MAC address.

5.2.3. Region C：This part is for all functionalities of WatchBP Office device.

C1：Read CBP data by different mode such as Low/ Full. Before running this, the function “Read all history data” shall be performed firstly.

C2：Read or Write User ID.

C3：Read/ Write BPM parameters/ setting values：

1. Highest inflation pressure of AUS mode： Valid parameter: 0(not setting), 160, 180, 200, 220, 240
2. Highest inflation pressure of Auto mode： Valid parameter: 0(not setting), 160, 180, 200, 220,

240.The device inflates the cuff using fuzzy logic to proper cuff pressure; the Highest Inflation Pressure is considered as a safeguard pressure.

3. SW_AUTO_hide：Set Show readings during rest time in auto mode. true:hide/false:show.
4. SW_SEL_silent：Beeper
5. SW_AUS_Hide：Set Show cuff pressure during deflation in AUS mode. true:hide/false:show.
6. SW_AVG_no_include_first：Set Average is include first memory data. true:is/false:is not.
7. SW_CBP：Set CBP measurement true:enabled/false:disabled.
8. SW_AFib：Set AFib measurement true:enabled/false:disabled.
9. SW_AMPM：Set 12/24-hour clock true:12-hour/false:24-hour.
10. SW_Kpa：Set Pressure unit: true:Kpa/false:mmHg.
11. RestTime：Rest time of auto mode.Start countdown base on rest time before 1st measurement in auto mode.
12. IntervalTime：Interval time of auto mode.Start countdown base on interval time before 2nd~6th measurement in auto mode.
13. AutoMeasureNumber: It’s number of measurements in auto mode.

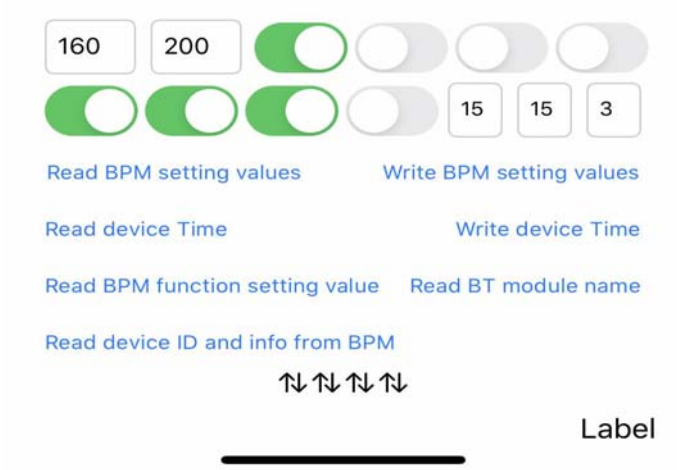
Note: Below are illegal settings

(1) AutoMeasureNumber=1 or 2 and SW_AVG_no_include_first=true

(2) AutoMeasureNumber=1 and SW_AFib =true

(3) SW_CBP=true, AutoMeasureNumber * IntervalTime >= 1800

For instance, the BPM setting values is included 13 parameters such as “160, 200, True, False, False, False, True, True, True, False, 15, 15, 3”



5.2.4. Refer to WBOViewController from the demo code (sample code) to get more detailed.

5.2.5. Operation Sequence：

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

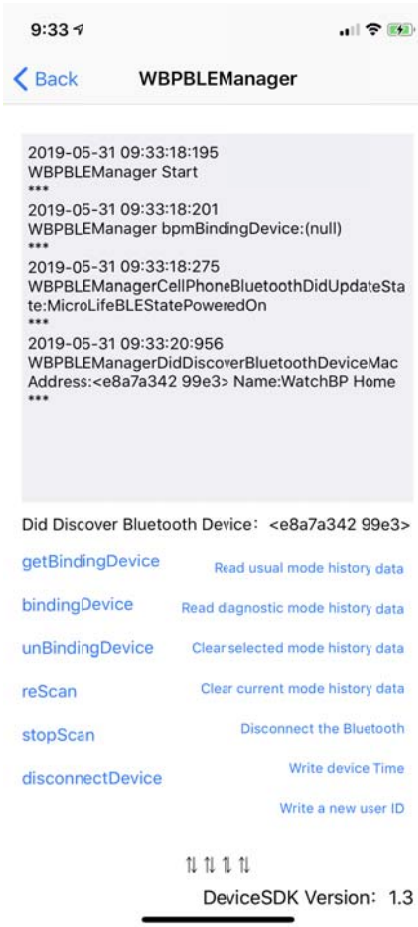
5.2.5.2. If a device is bonded, it will be connected accordingly. If not, the “bindingDevice” can be used to run bonding process.

5.2.5.3. Once the device is connected, select each function for communication with device.

Chapter 6 Demo App

Each “XXXBLEManager” has the same operation sequences and features. The following demonstration is mainly conducted by using the model WBPBLEManager. Refer to the sample code to see the complete application for dedicated devices.

6.1. Scanning / Discovery：



1. First of all, the WBPBLEManager starts Scanning / Discovery procedure. The detailed information will be displayed in Region A.

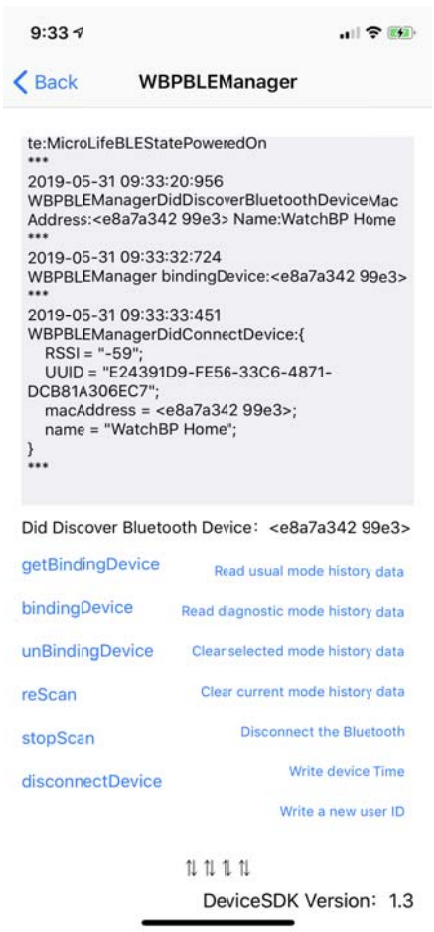
2. The WBPBLEManager bpmBindingDevice: * : (null) indicates that the remote device wasn’t bonded yet.

3. The WBPBLEManagerCellPhoneBluetoothDidUpdateState:MicroLifeBLEStatePoweredOn indicates that the power status of Bluetooth from cell phone is On.

4. The WBPBLEManagerDidDiscoverBluetoothDeviceMacAddress:

	<e8a7a342 99e3> Name:WatchBP Home indicates that the device with MAC address <e8a7a342 99e3> is discovered nearby. 5. In Region B, the “Did Discover Bluetooth Device：<e8a7a342 99e3>” stands for that the discovered/ selected device can be used in next steps such as Connection or Bonding. 6. In Region C, click the function “bindingDevice” to run bonding process.

6.2. Connection：

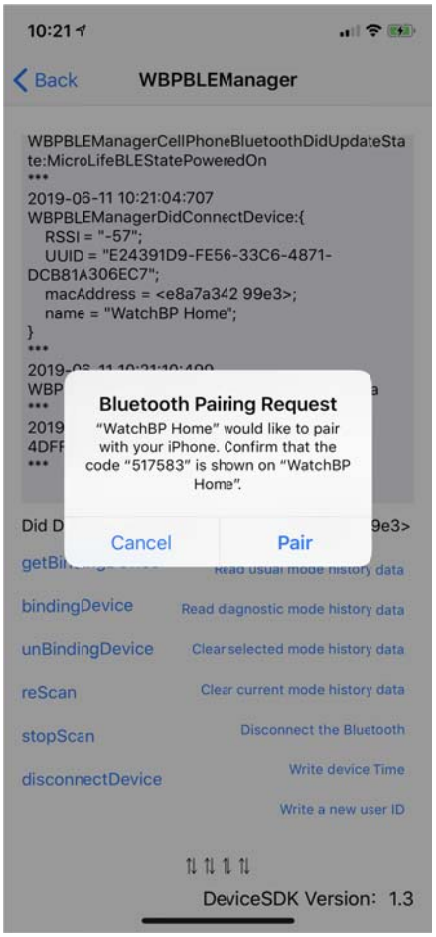


1. The information about the device bonding will be shown “WBPBLEManager bindingDevice:<e8a7a342 99e3>” in Region A.

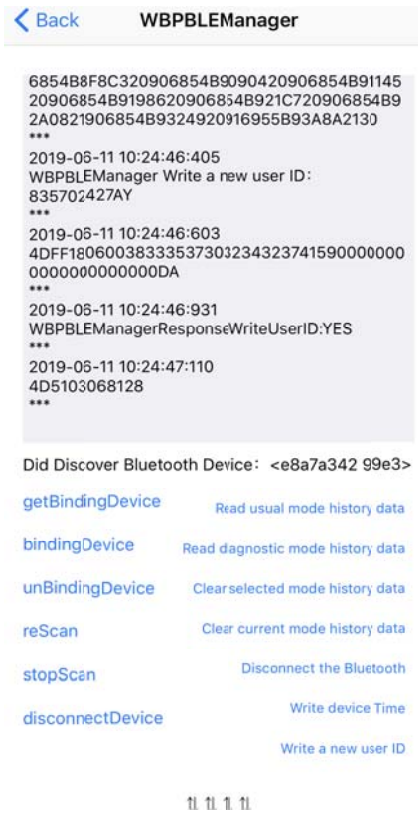
2. The following indicates that the device is connected.

WBPBLEManagerDidConnectDevice:{
RSSI = "-68";
UUID = "E24391D9-FE56-33C6-4871-DCB81A306EC7";
macAddress = <e8a7a342 99e3>;
name = "WatchBP Home";
}

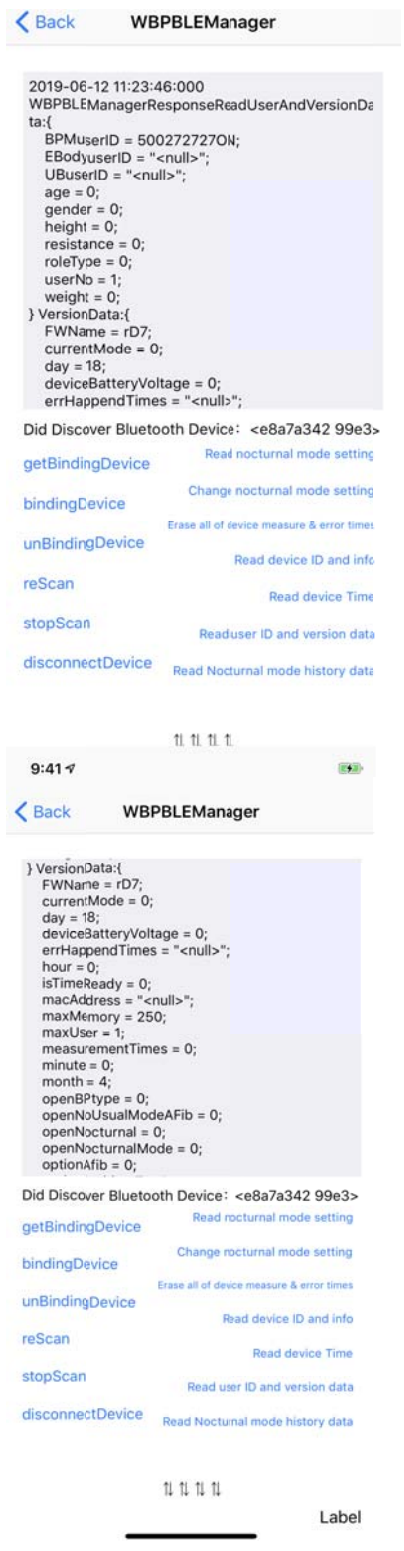
6.3. Bonding / Pairing：

	1. There is a message to confirm the bonding between device and cellphone if they haven't bonded yet. 2. Click "Pair" to run the bonding process.

6.4. Testing Command：Write a new user ID to BPM

	1. The command “Write a new user ID” is to write a new user ID to device. 2. The following “WBPBLEManager Write a new user ID：835702427AY”indicates that the writing value is “835702427AY” made up of ASCII code. 3. The response is as below: WBPBLEManagerResponse WriteUserID:YES. It means that the process is successful. 4. It can be checked by the command “Read user ID and version data”.

6.5. Testing Command：Read user ID and version data from BPM



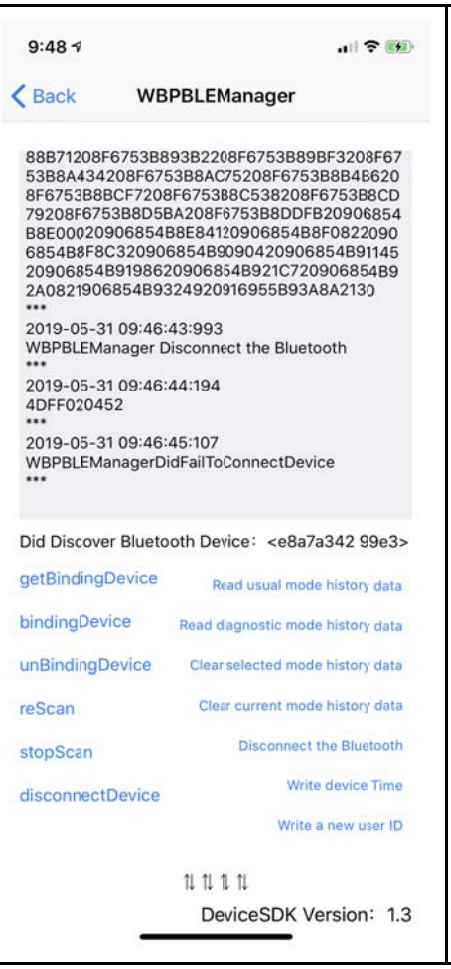
1. The command “Read user ID and version data” is to receive the information of user ID and device version.

2. The response is as below:

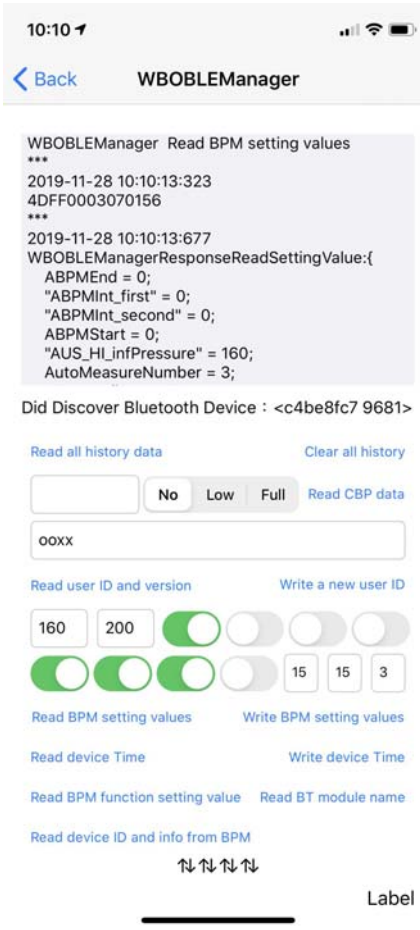
WBPBLEManagerResponseReadUserAndVersionData:{
BPMuserID = 500272727ON;
EBodyuserID = "<null>";
UBuserID = "<null>";
age = 0;
gender = 0;
height = 0;
resistance = 0;
roleType = 0;
userNo = 1;
weight = 0;
} VersionData:{
FWName = rD7;
currentMode = 0;
day = 18;
deviceBatteryVoltage = 0;
errHappendTimes = "<null>";
hour = 0;
isTimeReady = 0;
macAddress = "<null>";
maxMemory = 250;

	maxUser = 1;
	measurementTimes = 0;
	minute = 0;
	month = 4;
	openBPtype = 0;
	openNoUsualModeAFib = 0;
	openNocturnal = 0;
	openNocturnalMode = 0;
	optionAfib = 0;
	optionAmbientT = 1;
	optionDeviceID = 0;
	optionDiagnosticModeAFib = 0;
	optionIHB = 1;
	optionMAM = 1;
	optionTubeless = 0;
	protocolVersion = 0;
	second = 0;
	year = 2019;
	}
	
	3. The “BPMuserID” stands for user ID and the value is “500272727ON”.
	
	4. The “VersionData” stands for device version included name of firmware “FWName” and its building date “day”, “month” and “year”.

6.6. Disconnection

	1. In order to disconnect the device completely, either “disconnectDevice” (Region C) or “Disconnect the Bluetooth” (Region D) is available. 2. The response is the following “WBPBLEManagerDidFailToConnectDevice” once the disconnection command is executed.

6.7. Testing Command：Read BPM setting values (WBOBLEManager)



1. The command “Read BPM setting values” is to read the setting values of BPM.

2. The response is as below:

WBOBLEManagerResponseReadSettingValue:{

ABPMEnd = 0;

"ABPMInt_first" = 0;

"ABPMInt_second" = 0;

ABPMStart = 0;

"AUS_HI_infPressure" = 160;

AutoMeasureNumber = 3;

"CBPInt_first" = 0;

"CBPInt_second" = 0;

"CBP_zone1_meas_off" = 0;

"CBP_zone2_meas_off" = 0;

"HI_infPressure" = 200;

IntervalTime = 15;

RestTime = 15;

"SW_AFib" = 1;

"SW_AMPM" = 1;

"SW_AUS_Hide" = 0;

"SW_AUTO_hide" = 1;

"SW_AVG_no_include_first" = 0;

"SW_CBP" = 1;

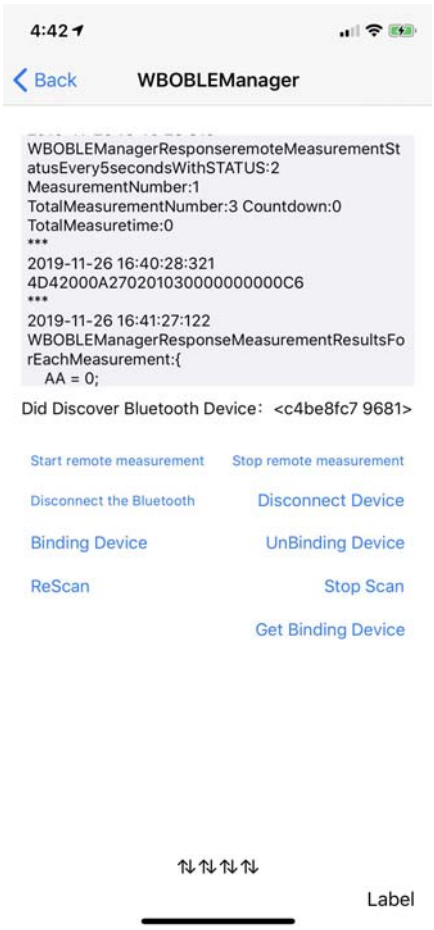
"SW_Kpa" = 0;

"SW_SEL_silent" = 0;

"SW_checkhide" = 0;

}

6.8. Testing Command：Start remote measurement / Stop remote measurement (WBOBLEManager)



1. The command “Start remote measurement” / “Stop remote measurement” is to run the process of remote control.

2. During measuring process, the relative status can be found by the following response:
WBOBLEManagerResponseremoteMeasurementStatusEvery5secondsWithSTATUS:1
MeasurementNumber:1
TotalMeasurementNumber:3
Countdown:15
TotalMeasuretime:0

3. Once the measurement is completed, the result is the following:
WBOBLEManagerResponseMeasurementResultsForEachMeasurement:{
AA = 0;
ABPM = 0;
AFIb = 0;
AM = 0;

	CPP = 60;
	CSBP = 111;
	IHB = 0;
	LB = 0;
	MAM = 0;
	MAP = 58;
	MCBP = 786;
	PM = 0;
	PVR = 202;
	SM = 0;
	arr = 0;
	condition = 1;
	cuffokr = 0;
	day = 26;
	deviceMode = 66;
	dia = 52;
	diagnostic = 0;
	errorCode = 0;
	errorMessage = "<null>";
	hour = 16;
	hr = 76;
	isFor3G = 0;
	minute = 27;
	month = 11;
	resultCode = 0;
	sys = 103;
	usual = 0;
	year = 2019;
	}
	HistoryMeasurementNumber:
	0
	CurrentMeasurementTimes:3

	AverageCalculationWhenMeasurement:YES
	。
	3.2.WBOBLEManagerResponseReadCBPData:{
	CBPdatas = (
	);
	CalCBPs = (
	5469,
	5461,
	5446,
	…...
	5492
	);
	dformat = 0;
	}