# Hryfi ne SDK description

# V 1.4

[Hryfi ne SDK description 1](#_Toc216870694)

[V 1.4 1](#_Toc666005099)

[The Xcode integration instructions 1](#_Toc862930729)

[1. Some instructions (refer to demo) All instructions need to be registered first, according to key, and then return the corresponding Dic information instructions. See the attachment for all key instructions 3](#_Toc1319694112)

[11, and the dial is correlated 15](#_Toc887834168)

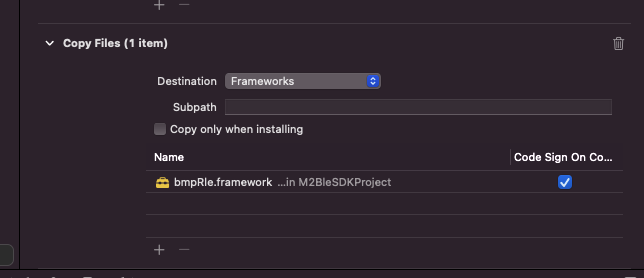
[Attachment I: All key instructions 16](#_Toc1112482220)

# The Xcode integration instructions

## Put the LHZL BleSdk.framework Pull it into the project

## grasp bmpRle.framework Pull into the project, the dial push related,

And were added at the copu F iles



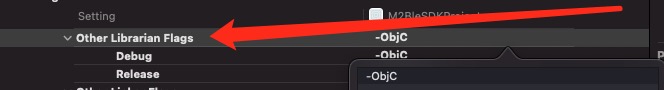
## Add Bluetooth related permission description and network permissions to the project infoPlist



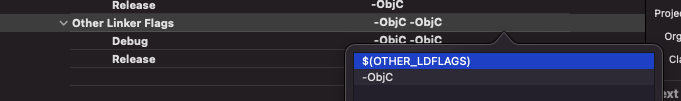




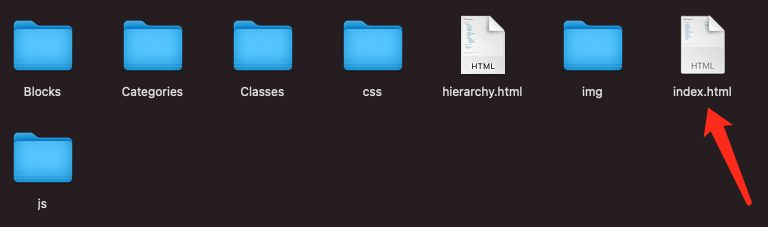
## Add other Librarian flags to join-Objc in project Bulid Setting

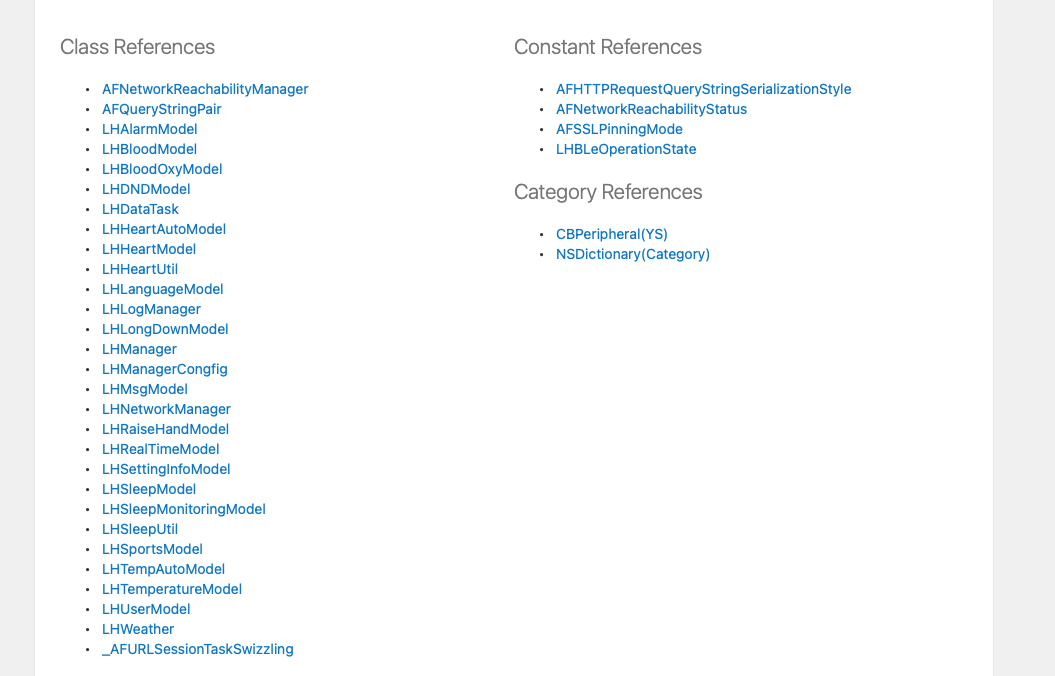


## Add the $ (OTHER \_ LDFLAGS) -Objc in the other Linker Flags



## All class references, index of html file in mo open doc file.html, in the browser open can see all information about the available models





# Some instructions (refer to demo) All instructions need to be registered first, according to key, and then return the corresponding Dic information instructions. See the attachment for all key instructions

Similar to this form LHManager addObserver: self identifier: kLHDeviceSupportInfo mainThread: YES actionBlock: ^ (id observer, NSDictionary \* dictionary) are the content of the callback, the returned dic key is kLHDeviceSupportInfo, you can get Value

### Register the Bluetooth status and device connection status

-(void )registerCallBack{

\_\_weak typeof (self ) weakself = self ;

/// Register the Bluetooth status callback

[LHManager addObserver:self identifier:kLHCentralManagerState mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

CBCentralManager \* devicesSate =   [dictionary objectForKey :kLHCentralManagerState ];

switch (devicesSate.state ) {

case CBManagerStatePoweredOn :

NSLog (@ "\* \* \* \* open");

**break ;**

case CBManagerStatePoweredOff :

NSLog (@ "\* \* \* \* off");

[weakself.navigationController popViewControllerAnimated:NO ];

**break ;**

**default :**

**break ;**

}

}];

/// Register the device status callback status callback

[LHManager addObserver:self identifier:kLHDevicesState mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

/\*\*

\* Device connection status

\* NSDictionary Set the information key value, value: 0: in connection, 1, connection was successful, 2, binding failed, 3, connection was disconnected

\*/

[weakself.navigationController popViewControllerAnimated:NO ];

NSNumber \*num =   [dictionary objectForKey :kLHDevicesState ];

switch (num.intValue ) {

**case 0:**

NSLog (@ "\* \* \* \* is connecting");

**break ;**

**case 1:**

NSLog (@ "\* \* \* \* Bluetooth is successful");

**break ;**

**case 2:**

NSLog (@ "\* \* \* \* binding failed");

**break ;**

**case 3:**

NSLog (@ "\* \* \* \* Bluetooth Disconnect");

**break ;**

**case 4:**

NSLog (@ "\* \* \* \* First binding");

/// First binding, watch pop-up window, APP to clear the historical data of the day

weakself.title =[NSString stringWithFormat :@"%@(%@)",[LHManager sharedInstance].connectPeripheral.name ,[LHManager sharedInstance].connectPeripheral.mac] ;

**break ;**

**case 5:**

NSLog (@ "\* \* \* \* non-first binding");

/// Consent to binding, non-first time, no popup

weakself.title =[NSString stringWithFormat :@"%@(%@)",[LHManager sharedInstance].connectPeripheral.name ,[LHManager sharedInstance].connectPeripheral.mac] ;

**break ;**

**default :**

**break ;**

}

}];

### 2, the watch actively sent to the app command callback registration

-(void )commandSentByBracelet{

[LHManager addObserver:self identifier:kLHDeviceFindphone mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, start looking for a phone (looking for a phone, you can play a sound) 1, end looking for a phone (stop looking for a phone, you need to stop playing a sound)

NSLog (@ "- - - - - -to find a mobile phone to open and close- - -%@ - - - -", dictionary);

}];

[LHManager addObserver:self identifier:kLHMusicControl mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, play 1, pause, 2, previous song 3, next song

NSLog (@ "- - - - - -hand ring controls app music playback, but it is only an agreement, actually can not control- - -%@ - - - -", dictionary);

}];

[LHManager addObserver:self identifier:kLHDeviceControlPhoneCamera mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, open your camera interface 1, take photos, 2, turn off the camera interface

NSLog (@ "- - - - -hand ring control app photo- - - -%@ - - - -", dictionary);

}];

/// Bluetooth support function information class, can save the returned model, according to the model to judge whether the support function, see the model definition

[LHManager addObserver:self identifier:kLHManagerCongfig mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

self .managerCongfig = dictionary[kLHManagerCongfig];

NSLog (@ "Bluetooth supports function information class managerCongfig- - - - - - - - - - - -%@", self f.managerCongfig);

}];

}

### 3, search for the device callback, return to the device list, and select the device for connection

[LHManager addObserver:self identifier:kLHScanType  mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - - - - - - - - - - -search equipment- - - - - -dic%@", dictionary);

NSMutableArray <CBPeripheral \*> \*    per =   [dictionary objectForKey :kLHScanType ];

weakself.devieceArray = per;

[weakself.tableview reloadData ];

}];

### 4. Select the device to connect, and the successful connection callback will be displayed above

[[LHManager sharedInstance] connectWithPeripheral:per];

### 5. Disconnect

# warning If disconnected, call- (void) clearData first, otherwise it will reconnect

/// clear data

/// clear data

[self .lhBleManager clearData];

/// disconnect

[self .lhBleManager disConnect];

### 6. Get the supported function information of the device

/// Bluetooth device support function information, sdk send, only need to register the callback (important), the device support function class

/// Bluetooth support function information class, can save the returned model, according to the model to judge whether the support function, see the model definition

[LHManager addObserver:self identifier:kLHManagerCongfig mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

self .managerCongfig = dictionary[kLHManagerCongfig];

NSLog (@ "Bluetooth supports function information class managerCongfig- - - - - - - - - - - -%@", self f.managerCongfig);

}];

### 7, the callback registration actively returned by the Bluetooth device

/ / It is best to register in the camera operation column, the watch actively sends the instructions related to the camera, and the action of the app to the watch (, @ "camera ready", @ "mobile phone photo finished", @ " mobile phone end photo) is the same process, but the interaction between the two sides

-(void )commandSentByBracelet{

[LHManager addObserver:self identifier:kLHDeviceFindphone mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, start looking for a phone (looking for a phone, you can play a sound) 1, end looking for a phone (stop looking for a phone, you need to stop playing a sound)

NSLog (@ "- - - - - -to find a mobile phone to open and close- - -%@ - - - -", dictionary);

}];

[LHManager addObserver:self identifier:kLHMusicControl mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, play 1, pause, 2, previous song 3, next song

NSLog (@ "- - - - - -hand ring controls app music playback, but it is only an agreement, actually can not control- - -%@ - - - -", dictionary);

}];

[LHManager addObserver:self identifier:kLHDeviceControlPhoneCamera mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 0, open your camera interface 1, take photos, 2, turn off the camera interface

NSLog (@ "- - - - -hand ring control app photo- - - -%@ - - - -", dictionary);

}];

/// Bluetooth support function information class, can save the returned model, according to the model to judge whether the support function, see the model definition

[LHManager addObserver:self identifier:kLHManagerCongfig mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

self .managerCongfig = dictionary[kLHManagerCongfig];

NSLog (@ "Bluetooth supports function information class managerCongfig- - - - - - - - - - - -%@", self f.managerCongfig);

}];

}

### 8, Health data callback registration (see demo)

/// Register the callback "Get all the health data" and "Get the health data of the day", the single measurement results will also be returned in the following callback,

-(void )notifiHealthData{

[LHManager addObserver:self identifier:kLHRealtimeSports mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

LHRealTimeModel \* realTimeModel =   [dictionary objectForKey :kLHRealtimeSports ];

NSLog (@ "\* \* \* \* Real-time step data- - - - - -%@", realTimeModel);

}];

[LHManager addObserver:self identifier:kLHSYHelthData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSNumber \*code =   [dictionary objectForKey :kLHSYHelthData ];

NSLog (@ "\* \* \* \* The start or end of health data synchronization callback- - - - - -%d", code.intValue);

}];

[LHManager addObserver:self identifier:kLHSportData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* sportArray =   [dictionary objectForKey :kLHSportData ];

NSLog (@ "\* \* \* \* sports data- - - - - -%@", sportArray);

}];

[LHManager addObserver:self identifier:kLHSleepData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* sleepArray =   [dictionary objectForKey :kLHSleepData ];

NSLog (@ "\* \* \* \* Sleep data- - - - - - -%@", sleepArray);

}];

[LHManager addObserver:self identifier:kLHHeartData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* heartArray =   [dictionary objectForKey :kLHHeartData ];

NSLog (@ "\* \* \* \* Heart rate data- - - - - -%@", heartArray);

}];

[LHManager addObserver:self identifier:kLHBloodData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* bloodArray =   [dictionary objectForKey :kLHBloodData ];

NSLog (@ "\* \* \* \* Blood pressure data- - - - - -%@", bloodArray);

}];

[LHManager addObserver:self identifier:kLHBloodOxyData mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* bloodOxyArray =   [dictionary objectForKey :kLHBloodOxyData ];

NSLog (@ "\* \* \* \* Blood Oxygen data- - - - - -%@", bloodOxyArray);

}];

}

### 9. Unbind the equipment and restore the factory settings

* 1, Unbind the device

/ / Popup prompts the user: Please find the device in [Settings] - - -> [Bluetooth] and unbind

[self.lhBleManager unBindDevice];

* 2. Return to the factory settings

[self.lhBleManager resettingDevice];

### 10, other instructions are sent to reference demo

The if ([per isEqualToString: @ "Get the settings information"]) {

/// Bluetooth setting information, sdk to send, only need to register the callback (important), you can save the returned model, set the synchronization to modify the model

[LHManager addObserver:self identifier:kLHSettingCofig mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

self .settingInfoModel = dictionary[kLHSettingCofig];

NSLog (@ "Bluetooth Settings information settingInfoModel- - - - - - - - - - - - - - - - - - - - - -%@", self f.settingInfoModel );

}];

[self .lhBleManager getSettingConfig];

} else if ([per isEqualToString: @ "Get the alarm clock list"]) {

/// Alarm clock list, save up, set up when using

[LHManager addObserver:self identifier:kLHGetAlarm mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSArray \* array = dictionary[kLHGetAlarm ];

NSLog (@ "Alarm clock list- - - - - - - - - - - - - - - - - - - - -%@", array);

}];

[self .lhBleManager getAlarm];

} else if ([per isEqualToString: @ "Set the alarm clock to"]){

[LHManager addObserver:self identifier:kLHSetAlarms mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether setting the alarm clock callback is successful- - - - -%@ - - - -", dictionary);

}];

/// Adjust the acquired alarm list, and create a new alarm clock if empty

LHAlarmModel \*model = [[LHAlarmModel alloc ] init ];

model.year = [[NSDate date ] ys\_ToString :@"yy"].longLongValue - 2000;

model.month = [[NSDate date ] ys\_ToString :@"MM"].longLongValue ;

model.day = [[NSDate date ] ys\_ToString :@"dd"].longLongValue ;

model.hour = [[NSDate date ] ys\_ToString :@"HH"].longLongValue ;

model.min = [[NSDate date ] ys\_ToString :@"mm"].longLongValue ;

model.number = 0;

model.none = 0;

model.d1 = 1;

model.d2 = 1;

model.d3 = 1;

model.d4 = 1;

model.d5 = 1;

model.d6 = 1;

model.d7 = 1;

NSMutableArray \*arr = [[NSMutableArray alloc ]initWithObjects :model, nil ];

[self .lhBleManager synAlarm:arr];

} else if ([per isEqualToString: @ "Get all health data"]) {

/ / Register the health data callback first

[self .lhBleManager getAllHelthData];

} else if ([per isEqualToString: @ "Get that day health data"]) {

/ / Register the health data callback first

[self .lhBleManager getTodayHelthData];

} else if ([per isEqualToString: @ "Set the target steps of"]){

[LHManager addObserver:self identifier:kLHSetAim mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether the setting number of target steps are successful- - - - -%@ - - - -", dictionary);

}];

/// Modify the LHSettingInfoModel according to the number of target steps taken in the setting

self .settingInfoModel.aim =  1000;

[self .lhBleManager aim: self .settingInfoModel.aim];

} else if ([per isEqualToString: @ "Set up the user information"]){

/// Modify the LHSettingInfoModel according to the user information obtained in the setting

[LHManager addObserver:self identifier:kLHUserModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the user information is successful- - - - - -%@ - - - -", dictionary);

}];

LHUserModel \* userModel =   self .settingInfoModel .usrModel ;

userModel.sex = 0;

userModel.age = 24;

userModel.height = 174;

userModel.weight = 75;

[self .lhBleManager synUsrInfo :userModel];

} else if ([per isEqualToString: @ "Set a sedentary reminder"]){

/// Modify the longDownModel in the LHSettingInfoModel according to the user information obtained in the setting

[LHManager addObserver:self identifier:kLHLongDownModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the sedentary reminder is successful- - - - -%@ - - - -", dictionary);

}];

LHLongDownModel \*model = self .settingInfoModel.longDownModel;

model.threshold = 150;

model.start = 8;

model.end = 22;

model. Time = 3; / / calculation formula: time \* 15 + 30. Note that the UI is divided into display minutes, say time = 1, and the UI display by formula = 45. Push back when setting up

model.open = NO ;

model.noonBreak = NO ;

model. The d1 = 1; / / switch, open to 1, close to 0, the same below

model.d2 = 1;

model.d3 = 1;

model.d4 = 1;

model.d5 = 1;

model.d6 = 1;

model.d7 = 1;

[self .lhBleManager synLongDown:model];

} else if ([per isEqualToString: @ "wristband real-time step switch"]){

[LHManager addObserver:self identifier:kSetRealTimeSports mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Is the real-time step switch of the bracelet successful- - - - -%@ - - - -", dictionary);

}];

/// Open or close the real-time step number upload switch value: 0 is closed, 1 is closed, it is recommended to enter the front desk is open, enter the background is closed

[self .lhBleManager setRealTimeSports:0];

} else if ([per isEqualToString: @ "Set the hand up screen"]){

[LHManager addObserver:self identifier:kSetRaiseHandModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the lighting of the hand lifting screen is successful- - - - -%@ - - - -", dictionary);

}];

/// Modify the LHRaiseHandModel in the LHSettingInfoModel according to the user information obtained in the setting

LHRaiseHandModel \*model = self .settingInfoModel.raiseHandModel;

model.open = NO ;

model.start = 8\*60;

model.end = 22\*60;

[self .lhBleManager raiseHand:model];

} else if ([per isEqualToString: @ "Set up the sleep monitoring"]){

/// Register for a successful callback

[LHManager addObserver:self identifier:kSetSleepMonitoringModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the setting of sleep monitoring is successful- - - - - -%@ - - - -", dictionary);

}];

/// Modify the LHSleepMonitoringModel in LHSettingInfoModel according to the user information obtained in the setting, you need to save the setting information first

LHSleepMonitoringModel \*model = self .settingInfoModel.sleepMonitoringModel;

model.open = YES ;

model.start = 22 \* 60; / / Start minutes: 22: \* 60 minutes

model. End = 8 \* 60; / / End minutes: 8 \* 60 minutes

[self .lhBleManager sleepMonitoring:model];

} else if ([per isEqualToString: @ "Set the heart rate to automatically measure the"]){

[LHManager addObserver:self identifier:kLHHeartAutoMode mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the automatic measurement of heart rate is successful- - - - -%@ - - - -", dictionary);

}];

/// Modify the LHHeartAutoModel in LHSettingInfoModel according to the user information obtained in the setting, you need to save the setting information first

LHHeartAutoModel \*model = self .settingInfoModel.heartAutoModel;

model.open = NO ;

model.sleep = NO; / / sleep 0: off 1: on

model. Rate = 60; / / default 60, APP divided at 60:90:120 (tentative)

model.start = 8 \* 60; / / Start time several minutes: 22:00 \* 60 minutes

model. End = 22 \* 60; / / End minutes: 8 \* 60 minutes

[self .lhBleManager heartAuto:model];

} else if ([per isEqualToString: @ "Set the body temperature to automatically measure the"]){

[LHManager addObserver:self identifier:kLHTempAutoModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the automatic measurement of body temperature is successful- - - - -%@ - - - -", dictionary);

}];

/// Modify the LHTempAutoModel in LHSettingInfoModel according to the user information obtained in the setting, you need to save the setting information first

LHTempAutoModel \*model = self .settingInfoModel.tempAutoModel;

model.open = NO ;

model. Rate = 60; / / default 60, APP divided at 60:90:120 (tentative)

model.start = 8 \* 60; / / Start time several minutes: 22:00 \* 60 minutes

model. End = 22 \* 60; / / End minutes: 8 \* 60 minutes

[self .lhBleManager tempAuto:model];

} else if ([per isEqualToString: @ "Set disturb mode"]){

/// Modify the LHDNDModel in LHSettingInfoModel according to the user information obtained in the setting, you need to save the setting information first

[LHManager addObserver:self identifier:kLHDNDModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the non-disturb mode is successful- - - - -%@ - - - -", dictionary);

}];

LHDNDModel \*model = self .settingInfoModel.dndModel;

model.open = NO ;

model.start = 22 \* 60; / / Start minutes: 22: \* 60 minutes

model. End = 8 \* 60; / / End minutes: 8 \* 60 minutes

[self .lhBleManager dnd :model];

} else if ([per isEqualToString: @ "Set the vibration to turn on or off the"]){

[LHManager addObserver:self identifier:kLHShake mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the vibration opening or close is successful- - - - -%@ - - - -", dictionary);

}];

BOOL shake = self .settingInfoModel .shake ?0:1;

[self .lhBleManager shake :shake];

} else if ([per isEqualToString: @ "Turn the photo"]){on or off

[LHManager addObserver:self identifier:kLHCamera mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Open or close whether the camera is closed successfully- - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager camera:YES ];

} else if ([per isEqualToString: @ "Set up the message push"]){

[LHManager addObserver:self identifier:kLHMsgModel mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -Set whether the message push is successful- - - - - -%@ - - - -", dictionary);

}];

/// Modify the LHMsgModel in LHSettingInfoModel according to the user information obtained in the setting, you need to save the setting information first

LHMsgModel \*mm = self .settingInfoModel.msgModel;

mm. The call = 1; / / If the bleCongfig supports 3.0 Bluetooth, this UI setting is hidden and does not allow the user to open the mm. Can call only =0

if (self .lhBleManager.bleCongfig.supportBle3) {

mm.call = 0;

}

mm.msg = 1;

mm.qq = 1;

mm.wechat = 1;

mm.facebook = 1;

mm.twitter = 1;

mm.skype = 1;

mm.line = 1;

mm.whatsapp = 1;

mm.kakaotalk = 1;

mm.instagram = 1;

[self .lhBleManager setMsgModel:mm];

} else if ([per isEqualToString: @ "Set up the language"]){

[LHManager addObserver:self identifier:kLHSetLuange mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether the setting language is successful- - - - - -%@ - - - -", dictionary);

}];

// Set the language If the luangeMode of the LHManagerCongfig is 1, set it to the languageArray language list of the LHSettingInfoModel to obtain the setting information. In the LHSettingInfoModel, the language is the index corresponding to the current language, otherwise the default is 1: Chinese 0: English

if (self .managerCongfig.luangeMode == 1) {

if (self .managerCongfig.languageArray.count >0) {

LHLanguageModel \*model =  self .managerCongfig .languageArray [0];

[self .lhBleManager setLanguage :model.index ];

}

**}else {**

/ / Default is 1: Chinese: 0: English

[self .lhBleManager setLanguage:1];

}

} else if ([per isEqualToString: @ "Unit system switching (value 0x01 British 0x00 metric)"]){

if (self .managerCongfig.supportUnit == 0) {

NSLog (@ "Not supported");

**return ;**

}

[LHManager addObserver:self identifier:kLHSetUnit mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether the unit system switching is successful- - - - - -%@ - - - -", dictionary);

}];

BOOL unitMode = self .settingInfoModel .unitMode ?0:1;

/ / Unit system switching (value 0x01 British system 0x00 metric system)

[self .lhBleManager setUnit :unitMode];

} else if ([per isEqualToString: @ "Wear mode 0: left hand 1: right hand"]){

[LHManager addObserver:self identifier:kLHSetWayOfWearing mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether the wearing method is successful- - - - - -%@ - - - -", dictionary);

}];

BOOL leftRight = self .settingInfoModel .leftRight ?0:1;

[self .lhBleManager setWayOfWearing:leftRight];

} else if ([per isEqualToString: @ "Switch over the time system (value 0x01 12 hours 0x00 24 hours)"]){

if (self .managerCongfig.supportTimeShow == 0) {

NSLog (@ "Not supported");

**return ;**

}

[LHManager addObserver:self identifier:kLHTimeShow mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether the switching time system is successful- - - - - -%@ - - - -", dictionary);

}];

/// Time display switch (value 0x01 12 hours 0x00 24 hours)

BOOL timeBy24 =   self .settingInfoModel .timeBy24 ?0:1;

[self .lhBleManager timeShow :timeBy24];

} else if ([per isEqualToString: @ "Set up the weather information"]){

if (self .managerCongfig.weatherSupport == 0) {

NSLog (@ "Not supported");

**return ;**

}

[LHManager addObserver:self identifier:kLHSetWeather mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - - -whether setting the weather information is successful- - - - -%@ - - - -", dictionary);

}];

LHWeather \*model = [LHWeather new ];

model.city\_name = @"shezhen";

model.temp = 26;

model.temp\_max = 37;

model.temp\_min = 10;

/ \* Reference value for current weather conditions (required transmission parameters)

0: Fine

1: Cloudy between sunny days

2: Cloudy

3: Yin

4: Shower

5: Rain

6: Thundershower

7: Snow

8: Haze

\*/

model.weather = 8;

[self .lhBleManager setWeatherInfo:model];

} else if ([per isEqualToString: @ "Start or turn off blood oxygen monitoring"]){

if (self .managerCongfig.bloodOxyMode == 0) {

NSLog (@ "Not supported");

**return ;**

}

[LHManager addObserver:self identifier:kLHBloodOxygenMeasurement mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

/// When the measurement is completed, the blood oxygen health data will get the last data display, a return represents the end of the measurement

// 1, on or off successfully, 0, on or off failed, 2, interrupted measurement due to error

NSLog (@ "- - - -whether the blood oxygen monitoring is successful- - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager bloodOxy:YES ];

} else if ([per isEqualToString: @ "Start or turn off the heart rate monitoring"]){

[LHManager addObserver:self identifier:kLHHeartMeasurement mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

/// The measurement is completed in the heart rate health data callback to get the last data display, there is a return represents the end of the measurement

// 1, on or off successfully, 0, on or off failed, 2, interrupted measurement due to error

NSLog (@ "- - - -Start or turn off heart rate monitoring successfully- - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager startHeart:YES ];

} else if ([per isEqualToString: @ "Start or close the blood pressure monitoring"]){

[LHManager addObserver:self identifier:kLHBloodMeasurement mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

/// The measurement is completed in the last data obtained in the blood pressure health data callback, and a return represents the end of the measurement

// 1, on or off successfully, 0, on or off failed, 2, interrupted measurement due to error

NSLog (@ "- - - -Start or close blood pressure monitoring successfully- - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager startBlood:YES ];

} else if ([per isEqualToString: @ "Start or close the temperature monitoring"]){

[LHManager addObserver:self identifier:kLHTempMeasurement mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

/// The measurement is completed in the last data obtained in the temperature health data callback, there is a return represents the end of the measurement

// 1, on or off successfully, 0, on or off failed, 2, interrupted measurement due to error

NSLog (@ "- - - -whether the temperature monitoring is successful- - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager startTemp:YES ];

} else if ([per isEqualToString: @ "Synchronize system time"]){

[LHManager addObserver:self identifier:KLHSynTime mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 1, setting succeeded, 0, setting failed

NSLog (@ "- - - -whether the synchronization system time is successful- - - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager synTime];

} else if ([per isEqualToString: @ "Start or end finding device"]){

[LHManager addObserver:self identifier:kLHSearchDevice mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

// 1, open or close successful, 0, open or off failed

NSLog (@ "- - - -Find whether the device is sent successfully- - - - - -%@ - - - -", dictionary);

}];

[self .lhBleManager searchDevice:YES ];

} else if ([per containsString: @ "Get the power quantity of"]){

[LHManager addObserver:self identifier:kLHPower mainThread:YES actionBlock:^(id observer, NSDictionary \*dictionary) {

NSLog (@ "- - -electricity- - - - - -%@ - - - -", dictionary);

}];

[self.lhBleManager getDevicePower];

}

The if ([per isEqualToString: @ "Unbind device"]){

/ / Popup prompts the user: Please find the device in [Settings] - - -> [Bluetooth] and unbind

[self .lhBleManager unBindDevice];

} else if ([per isEqualToString: @ "Restore the factory settings for"]){

[self .lhBleManager resettingDevice];

}

目录

[Hryfi ne SDK description 1](#_Toc827914815)

[V 1.3 1](#_Toc1217746792)

[The Xcode integration instructions 1](#_Toc1151177234)

[1) Put the LHZL BleSdk.framework Pull it into the project 1](#_Toc1155596015)

[2) grasp bmpRle.framework Pull into the project, the dial push related, 1](#_Toc260120637)

[3) Add Bluetooth related permission description and network permissions to the project infoPlist 1](#_Toc1718324414)

[4) Add other Librarian flags to join-Objc in project Bulid Setting 2](#_Toc518341242)

[5) Add the $ (OTHER \_ LDFLAGS) -Objc in the other Linker Flags 2](#_Toc1567582062)

[6) All class references, index of html file in mo open doc file.html, in the browser open can see all information about the available models 2](#_Toc1022334638)

[1. Some instructions (refer to demo) All instructions need to be registered first, according to key, and then return the corresponding Dic information instructions. See the attachment for all key instructions 3](#_Toc361601219)

[1. Register the Bluetooth status and device connection status 3](#_Toc52966723)

[2, the watch actively sent to the app command callback registration 4](#_Toc1153483603)

[3, search for the device callback, return to the device list, and select the device for connection 5](#_Toc1264034152)

[4. Select the device to connect, and the successful connection callback will be displayed above 5](#_Toc1713756540)

[5. Disconnect 5](#_Toc1055494216)

[6. Get the supported function information of the device 6](#_Toc1476364092)

[7, the callback registration actively returned by the Bluetooth device 6](#_Toc1225236806)

[8, Health data callback registration (see demo) 7](#_Toc334307359)

[9. Unbind the equipment and restore the factory settings 7](#_Toc886562161)

[10, other instructions are sent to reference demo 8](#_Toc1208697041)

[11, and the dial is correlated 15](#_Toc1523351114)

[Attachment I: All key instructions 16](#_Toc662133464)

### 11, and the dial is correlated

/// Push the custom wallpaper img way

-(void)pushCustomWallpaper:(UIImage\*)finalImage progress:(OTAProgressBlock)progress Complete:(OTACompleteBlock)completeBlock;

/// Push the custom wallpaper data mode

-(void)pushCustomWallpaperWihthData:(NSData \*)data  progress:(OTAProgressBlock)progress Complete:(OTACompleteBlock)completeBlock;

/// Disface push

/// @param data Dface Bin data

-(void)dailPush:(NSData \*)data  progress:(OTAProgressBlock)progress Complete:(OTACompleteBlock)completeBlock;

/// Push address book function

-(void)sendOtaContactData:(NSArray <HFContactModel \*>\*)Contacts progress:(OTAProgressBlock)progress Complete:(OTACompleteBlock)completeBlock

# Attachment I: All key instructions

/\*\*

\* Set up the weather key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*/

extern NSString \*const kLHSetWeather;

/\*\*

\* NSDictionary Set the message key value, value: CBPeripheral array

\*

\*/

extern NSString \*const kLHScanType ;

/\*\*

\* NSDictionary Set the information key value, value: CBManagerState

\*

\*/

extern NSString \*const kLHCentralManagerState;

/\*\*

\* Device connection status

\* NSDictionary Set information key value, value: 0: in connection, 1, successful connection, 2, binding, failed, 3, disconnected, 4, first binding, 5, non-first binding

\*

\*/

extern NSString \*const kLHDevicesState;

/\*\*

\* NSDictionary Set the information key value, value: LHSettingInfoModel

\*

\*/

extern NSString \*const kLHSettingCofig;

/\*\*

\* NSDictionary Watch in addition to the basic functions, to judge whether to support, value: LHManagerCongfig

\*

\*/

extern NSString \*const kLHManagerCongfig;

/\*\*

\* NSDictionary Set the information key value, value: NSarray <LHAlarmModel \*>

\*

\*/

extern NSString \*const kLHGetAlarm ;

/\*\*

\* NSDictionary Set information key value, value: @ (code) 0, setting succeeded, 1, setting failed

\*

\*/

extern NSString \*const kLHSetAlarms;

/\*\*

\* NSDictionary Set information key value, value: @ (code) 1, setting succeeded, 0, setting failed

\*

\*/

extern NSString \*const KGetAllHelthData;

/\*\*

\* Logo of synchronized health data

\* NSDictionary Set the information key value, value: 0, synchronization start 1, and synchronization end

\*

\*

\*

\*/

extern NSString \*const kLHSYHelthData;

/\*\*

\* Movement data key

\*

\*

\* NSDictionary Set the information key value, value: motion data array [LHSportsModel]

\*

\*/

extern NSString \*const kLHSportData ;

/\*\*

\* Blood pressure data for the key

\*

\*

\* NSDictionary Set the information key value, value: blood pressure data array [LHBloodModel]

\*

\*/

extern NSString \*const kLHBloodData ;

/\*\*

\* Blood oxygen data key

\*

\*

\* NSDictionary Set the information key value, value: Blood Oxygen data array [LHBloodOxyModel]

\*

\*/

extern NSString \*const kLHBloodOxyData ;

/\*\*

\* Heart rate data key

\*

\*

\* NSDictionary Set the information key value, value: heart rate data array [LHHeartModel]

\*

\*/

extern NSString \*const kLHHeartData ;

/\*\*

\* Sleep data key

\*

\*

\* NSDictionary Set the information key value, value: Sleep data array [LHSleepModel]

\*

\*/

extern NSString \*const kLHSleepData ;

/\*\*

\* Sleep data key

\*

\*

\* NSDictionary Set the information key value, value: body temperature data array [LHTemperatureModel]

\*

\*/

extern NSString \*const kLHTemperature ;

/\*\*

\* Real-time motion

\*

\*

\* Value: the LHRealTimeModel class

\*

\*/

extern   NSString \*const kLHRealtimeSports;

/\*\*

\* Set the target steps key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern   NSString \*const kLHSetAim ;

/\*\*

\* Set up the user information key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern   NSString \*const kLHUserModel;

/\*\*

\* Set up a sedentary reminder key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHLongDownModel;

/\*\*

\* Live steps key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kSetRealTimeSports;

/\*\*

\* Set the lifting screen key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kSetRaiseHandModel;

/\*\*

\* Set up the sleep monitoring key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kSetSleepMonitoringModel;

/\*\*

\* Set the heart rate to automatically measure the key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHHeartAutoMode;

/\*\*

\* Set the body temperature automatically measured key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHTempAutoModel;

/\*\*

\* Do not disturb the mode switch and set the key for the valid time period

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHDNDModel ;

/\*\*

\* Set the vibration on or off key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHShake ;

/\*\*

\* Turn the photo key on or off

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHCamera ;

/\*\*

\* Set up the message push key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHMsgModel ;

/\*\*

\* Set up the language key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHSetLuange;

/\*\*

\* Unit system switch (value 0x01 British system 0x00 metric system) key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHSetUnit ;

/\*\*

\* Wear mode 0: left hand 1: right hand key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHSetWayOfWearing;

/\*\*

\* Switch the time system (value 0x01 12 hours 0x00 24 hours) key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*

\*/

extern NSString \*const kLHTimeShow ;

/\*\*

\* Start or turn off the blood oxygen monitoring key

\*

\*

\* Value: @ (code) 1, turned on or off successfully, 0, turned on or off failed, 2, the measurement was interrupted due to an error

\*/

extern NSString \*const kLHBloodOxygenMeasurement;

/\*\*

\* Start off or turn off the heart rate monitoring key

\*

\*

\* Value: @ (code) 1, turned on or off successfully, 0, turned on or off failed, 2, the measurement was interrupted due to an error

\*/

extern NSString \*const kLHHeartMeasurement;

/\*\*

\* Start off or turn off the heart rate monitoring key

\*

\*

\* Value: @ (code) 1, turned on or off successfully, 0, turned on or off failed, 2, the measurement was interrupted due to an error

\*/

extern NSString \*const kLHHeartMeasurement;

/\*\*

\* Start or close the blood pressure monitoring key

\*

\*

\* Value: @ (code) 1, turned on or off successfully, 0, turned on or off failed, 2, the measurement was interrupted due to an error

\*/

extern NSString \*const kLHBloodMeasurement;

/\*\*

\* Start or close the temperature monitoring key

\*

\*

\* Value: @ (code) 1, turned on or off successfully, 0, turned on or off failed, 2, the measurement was interrupted due to an error

\*/

extern NSString \*const kLHTempMeasurement;

/\*\*

\* Synchronize the system time key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*/

extern NSString \*const KLHSynTime ;

/\*\*

\* Start or end by looking for the device key

\*

\*

\* Value: @ (code) 1, setting was successful, 0, setting failed

\*/

extern NSString \*const kLHSearchDevice;

/\*\*

\* Start or end by looking for the device key

\*

\*

\* Value: @ (code) 0, start looking for the phone 1, end looking for the phone

\*/

extern NSString \*const kLHDeviceFindphone;

/\*\*

\* Music play control key

\*

\*

\* Value: @ (code) 0, play 1, pause, 2, last song 3, next song

\*/

extern NSString \*const kLHMusicControl;

/\*\*

\* Hand loop control app photo key

\*

\*

\* Value: @ (code) 0, open your camera interface 1, perform photos, 2, close the camera interface

\*/

extern NSString \*const kLHDeviceControlPhoneCamera;