1.  Integration

## 1.1 Download

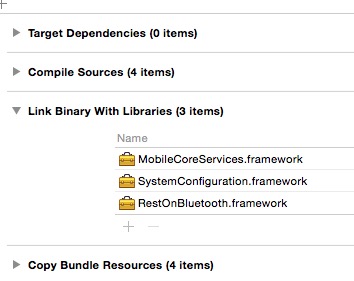
Download MilkyBluetooth.zip file and then unzip it

## 1.2 Import SDK

1）After un-zip the file, there are 3 folders as doc，sdk，demo.

2）mport the SDK\MilkyBluetooth.framework into your project

3）then add MobileCoreService.framework, SystemConfiguration.framework in the Link Binary with Libraries.



# **2.  Application function**

## 2.1  Header files description

**MilkyHistoryDataModel.h ： History data, Generate report from Dot.**

#import <Foundation/Foundation.h>

@interface MilkyHistoryDataModel : NSObject

//Start Date "yyyy-MM-dd"

@property(nonatomic,strong)NSString \*date;

//Time Zone

@property(nonatomic,strong)NSNumber \*timezone;

//Total sleep duration, except the falling sleep time and the awake time.

@property(nonatomic,strong)NSNumber \*duration;

// Start time of sleep by using the Unix timestamp (timeIntervalSince1970)

@property(nonatomic,strong)NSNumber \*startTime;

//Sleep score

@property(nonatomic,strong)NSNumber \*score;

//Time Duration of light sleep(minutes)

@property(nonatomic,strong)NSNumber \*light;

// Time Duration of middle sleep(minutes)

@property(nonatomic,strong)NSNumber \*rem;

// Time Duration of deep sleep(minutes)

@property(nonatomic,strong)NSNumber \*deep;

// Time Duration of awake(minutes)

@property(nonatomic,strong)NSNumber \*wake;

//Percentage of the deep sleep time

@property(nonatomic,strong)NSNumber \*MdDeepSleepPerc;

//Percentage of the middle sleep time

@property(nonatomic,strong)NSNumber \*MdRemSleepPerc;

//Percentage of the light sleep time

@property(nonatomic,strong)NSNumber \*MdLightSleepPerc;

//Percentage of awake time

@property(nonatomic,strong)NSNumber \*MdWakeSleepPerc;

//Duration time of the awake before get up(Minutes)

@property (nonatomic,strong)NSNumber \*MdWakeUpTime;

//Times of awake

@property(nonatomic,strong)NSNumber \*wake\_times;

// TimeStep(60) not available at this moment

@property(nonatomic,strong)NSNumber \*timeStep;

//Total record(one record per one minute)

@property(nonatomic,strong)NSNumber \*recordCount;

/\*Stop Mode

\* 0: By normal stop operation

\* 1: By Auto

\* 2: By PowerOff operation on the device

\* 3: Stop by Error(e.g. Low Battery、System fault)

\*/

@property(nonatomic,strong)NSNumber \*stopMode;

//The duration time of different status(second)

@property(nonatomic,strong)NSArray \*statusValueArray;

/\*sleepStateArray Sleep state level，describe：

0 awake

1 light sleep

2 middle sleep

3 deep sleep

\*/

@property(nonatomic,strong)NSArray \*sleepStateArray;

//Motion Intensity（float type, output range:0-255）

@property(nonatomic,strong)NSArray \*motionIntensityArray;

//Time of falling sleep

@property(nonatomic,strong)NSNumber \*asleepTime;

//Sleep efficiency（Percentage of sleep time across total time）

@property (nonatomic,strong)NSNumber \*SleepEfficient;

//Body movement level(-2:very few:monitor failure；-1:less than normal:data lost；+0:Normal；+1:more than normal: Restless；+2:too many: seriously restless)

@property (nonatomic,strong)NSNumber \*bodyMoveLevel;

//Version of algorithm

@property(nonatomic,strong)NSString \*arithmeticVer;

//Sleep Curve array（float type, output range:0.0-3.0）

@property (nonatomic,strong)NSArray \*SleepCurveArray;

/\*sleepCurveStatusArray，as below describe：

0 Normal

1 Initialization

2 Breath stop(not available for Dot)

3 Heart Pulse stop(not available for Dot)

4 Body movement

5 leaving bed(not available for Dot)

6 turnover(not available for Dot)

\*/

@property (nonatomic,strong)NSArray \*sleepCurveStatusArray;

//Mark up for the abnormal data(Data valid flag，short type,0-valid；bit1-body movement too less；bit2-body movement too less)

@property (nonatomic,strong)NSNumber \*flaginvalid;

// Score Deduction due to too long sleep time

@property (nonatomic,strong)NSString \*md\_sleep\_time\_decrease\_scale\_long;

// Score Deduction due to too short sleep time

@property (nonatomic,strong)NSString \*md\_sleep\_time\_decrease\_scale\_short;

// Score Deduction due to times of awake

@property (nonatomic,strong)NSString \*md\_wake\_cnt\_decrease\_scale;

// Score Deduction due to body movement (Restlessness)

@property (nonatomic,strong)NSString \*md\_body\_move\_decrease\_scale;

// Score Deduction due to deep sleep

@property (nonatomic,strong)NSString \*md\_perc\_deep\_decrease\_scale;

// Score Deduction due too long falling sleep time

@property (nonatomic,strong)NSString \*md\_fall\_asleep\_time\_decrease\_scale;

// Score Deduction due to sleeping time (too late)

@property (nonatomic,strong)NSString \*md\_start\_time\_decrease\_scale;

// Score Deduction due to good sleeping (ratio of middle sleep/deep sleep)

@property (nonatomic,strong)NSString \*benignSleepLow;

// Score Deduction due to sleeping efficiency

@property (nonatomic,strong)NSString \*sleepaceEfficientLow;

// Score Deduction due to too less of body movement

@property (nonatomic,strong)NSString \*bodyMoveLow;

//Original data,Value0;

@property(nonatomic,strong)NSArray \*originFeatureValue0Array;

//Original data,Value1;

@property(nonatomic,strong)NSArray \*originFeatureValue1Array;

//Original Data-tag

@property(nonatomic,strong)NSArray \*originTagArray;

@end

**BLEHelper.h ：the interfacing of the Bluetooth communication of the device,**

@property(nonatomic,readonly)CBPeripheral \*peripheral;// Bluetooth peripheral

@property(nonatomic,readonly)CBCharacteristic \*readCharactertic;//Read Characteristic

/\*

share：it need to be invoked at the beginning initWithPeripheral:readCharactertic: ，for the initialization of Bluetooth parameter

Parameter：

peripheral (Bluetooth peripheral)

readCharactertic ((Read Characteristic)

\*/

+(BLEHelper \*)share;

// opposite function of “share”

+(void)deshare;

/\*

initWithPeripheral：Initialization， the “share” need to be invoked with this peripheral for the initialization of Bluetooth parameters

parameter：

peripheral (Blue peripheral)

readCharactertic (Bluetooth Read Characteristic)

\*/

- (void) initWithPeripheral:(CBPeripheral \*)peripheral readCharactertic:(CBCharacteristic \*)readCharactertic;

**BleManager.h: Bluetooth connection scanning, receive the Bluetooth communication data**

// Whether the device is connected with Bluetooth or not

- (BOOL)isConnected;

//Peripheral scanning

-(void)scanfPeripheralWithWithSuccess:(void(^)(NSString \* ID, NSString \*name, CBPeripheral \*peripheral))success;

// Peripheral connection

-(void)bleConnectWithPeripheral:(CBPeripheral \*)peripheral

success:(void (^)(void))success

failure:(void (^)(NSString \*error))failure;

// Clear the request of Bluetooth connection

- (void)clearConnectRequestHandle;

/\*\*

\* Disconnect the connection of the Bluetooth \*/

-(void)bleDisconnectPeripheralWithsuccess:(void (^)(void))success;

**NewBleServiceInterface.h: function interfacing with Dot**

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "BLEHelper.h"

@class BTHNetworkAssistHandle;

@class SLPDownloadHistoryDataAssist;

@class SLPBHTRequestHelper;

@class NewBlePackage;

@class NewBleRequest;

@class BLEHelper;

@interface NewBleServiceInterface : NSObject

/\*

\*Time calibration

@timstamp 时间戳

@timezone 时区偏移

@specialTime 特殊时令 DST?

@specialTimeOffSet 特殊时令偏移 Time offset by DST?

\*/

+(void)checkDeviceTimeWithTimestamp:(UInt32)timstamp timeZoneOffSet:(int32\_t)timezone specialTime:(uint)specialTime specialTimeOffSet:(int32\_t)specialTimeOffSet;

/\*\*

\* Shortcut for Time calibration of Dot

\*/

+(void)milkyTimeCalibration;

/\*\*

\* Obtain the status of the Dot

\*

\* @param success status: 1：sampling in progress,0：no sampling taking place

\* @param failure query failure

\*/

+ (void)queryMilkyCollectStatusWithSuccess:(void (^)(NSInteger status))success

failure:(void (^)(void))failure;

/\*\*

\* Obtain the ID of the Dot

\*

\* @param success

\* @param fail

\*/

+(void)getMilkyIdWithSuccess:(void(^)(NSString \*milkyID))success fail:(void (^)())fail;

/\*

\*Device version number

\*/

+(void)newBleGetDeviceVersionWithSuccess:(void (^)(NSString \*version))success

failure:(void (^)(void))failure;

/\*

\*Battery status information

\*/

+(void)newBleGetDevicePowerStatusSuccess:(iBlock)success

failure:(void (^)(void))failure;

/\*

\*setting of the sleep monitoring

@hour 小时

@minutes 分

@times 时长

\*/

+(void)setMilkySleepMonitorWithHour:(uint)hour mintues:(uint)minutes duration:(uint)times WithSuccess:(void (^)(id responseObject))success fail:(void (^)())fail;

/\*\*

\* Alarm setting of Dot

\*

\* @param valid Valid of the alarm

\* @param offset Smart Alarm offset (Minutes)

\* @param hour Hour

\* @param min Minutes

\* @param weekArr Repeat date

\*/

+ (void)setAlarmInfoToMilkyWithValid:(int)valid andOffset:(int)offset andHour:(int)hour andMinutes:(int)min andWeekArr:(NSArray \*)weekArr andSuccess:(void (^)(id responseObject))success fail:(void (^)())fail;

/\*\*

\* Binding the current user ID with the Dot, which used to define the records of Dot belongings

\*/

+ (void)setUserIdToMilky:(int)userId WithSuccess:(void (^)(id responseObject))success fail:(void (^)())fail;

/\*\*

\* information of sleeping status

\*

\* @param success

\* ^(int isAsleep, int isAwake) (isAsleep 0:non-sleep; 1:in sleep; other:no effect) (isAwake 0:non-awake; 1:awake; other:no effect)

\* @param fail

\*/

+(void)getSleepingStatusInfoWithSuccess:(milkySleepingStatusInfoBlock)success fail:(void (^)())fail;

/\*

\*采集状态查询

\*/

+(void)checkStateCollection;

/\*\*

\* Change the sampling operation

\*

\* @param startOrStop 0x00:Stop sampling(if stopped，ignore this request) 0x01:start sampling(if already started, ignore this request) other：no effect

\* @param successBlock success return

\*/

+(void)changedSleepStatus:(int)startOrStop withSuccess:(changedMilkySleepStatusSuccessBlock)successBlock;

/\*

\*History data retrieving

@startTime

@startIndex

@count

\*/

+ (void)downloadHistoryDataFromTime:(UInt32)startTime

endTime:(UInt32)endTime

failed:(SLPBHTRequestFailed)failed

querySucceed:(SLPBHTHistoryDataQuerySucceed)querySucceed

dataRecieved:(SLPBHTHistoryDataRecieved)dataRecieved

completion:(SLPBHTRequestCompletion)completion;

## 3.  Connection flow of Dot

1）Tap the dot, Bluetooth function is active while there is a led lights up

2）Scan the dot from the phone, and establish the connection with the Dot.

3）If there is idle timeout, the bluetooth of the DOT will be disconnected, need to repeat above step 1 and 2 to resume the interfacing function.