

IOSMacBrainLinkProSDKSpecialEdition V1.3.1

Date: 0820 2021

Author: Liang Fang

SDK Version: 1.3.1

MCU: 3.1



更新记录:

1. V1.3.1 增加新的蓝牙数据字段 temperature 和 heartrate
2. V1.3 修复 iOS13.5 系统回连失败的 bug, 同时支持 iOS 和 macOS
3. V1.2 支持多连接或单连接
4. V1.0 支持单连接

目录

IOSMacBrainLinkProSDKSpecialEditon V1.3.1	1
IOSMacBrainLinkProSDKSpecialEditon 开发指南	4
介绍	4
你的第一个项目: IOS_HZLBlue4Demo.....	4
IOSMacBrainLinkProSDKSpecialEditon V1.3.1 API 参考	10
HZLBlueData 参考	10
Blue4Manager 参考	12

IOSMacBrainLinkProSDKSpecialEditon 开发指南

介绍

本指南将教你如何使用 iOsmacBrainLinkProSDKSpecialEditon 从宏智力公司的硬件中获取脑电波数据。这将使您的 iOS 应用程序能够接收和使用脑波数据，如 BLEMIND 和 BLEGRAVITY，你可以通过蓝牙 宏智力公司的硬件 和文件资源 iOsmacBrainLinkProSDKSpecialEditon 来获取他们。

功能:

接收脑波数据。同一时刻可以连接一个或多个蓝牙设备。

文件包含：

- API 参考(此文档)
- SDK 静态库和头文件
- iOsmacBrainLinkProSDKSpecialEditon V1.3.1.a
- HZLBlueData.h
- Blue4Manager.h
- IOS_HZLBlue4Demo/ Mac_HZLBlue4.0Demo

支持的硬件设备：

- 有电量的数据格式
 - BrainLink_Pro
 - Jii

支持的 iOS / macOS 版本：

- iOS 9.0 + / macOS 10.9+

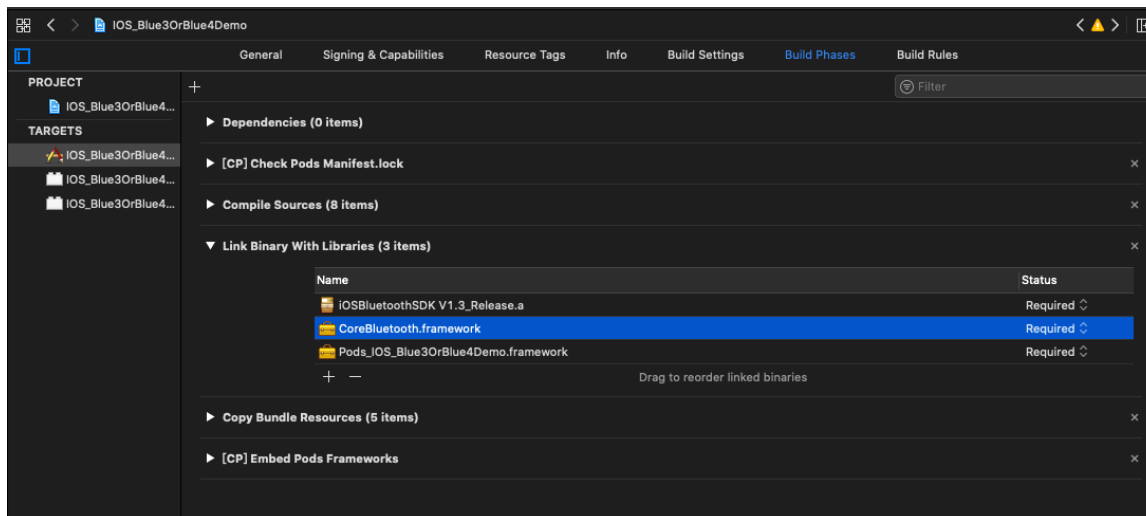
你的第一个项目: IOS_HZLBlue4Demo

第一步:

1.1 在 Xcode 项目里 TARGETS - Build Phases 导入 iOS 系统框架库如下

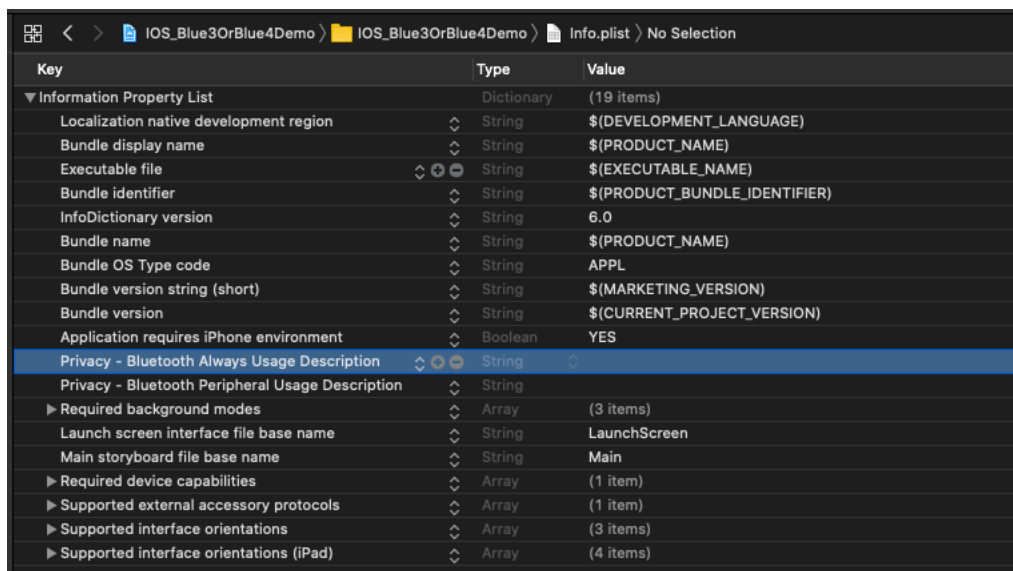
- CoreBluetooth.framework

如图：

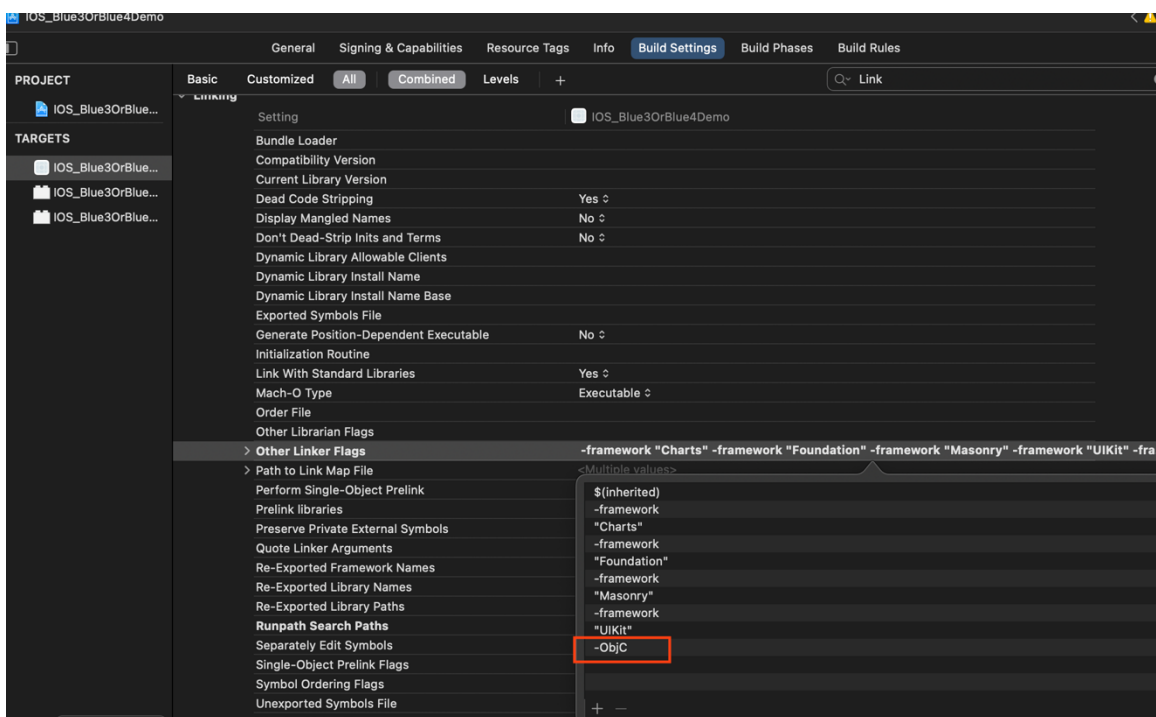


在 Info.plist 里添加蓝牙权限 (iOS13 需要添加蓝牙权限 Privacy - Bluetooth Always Usage Description , Privacy - Bluetooth Peripheral Usage Descriptio)

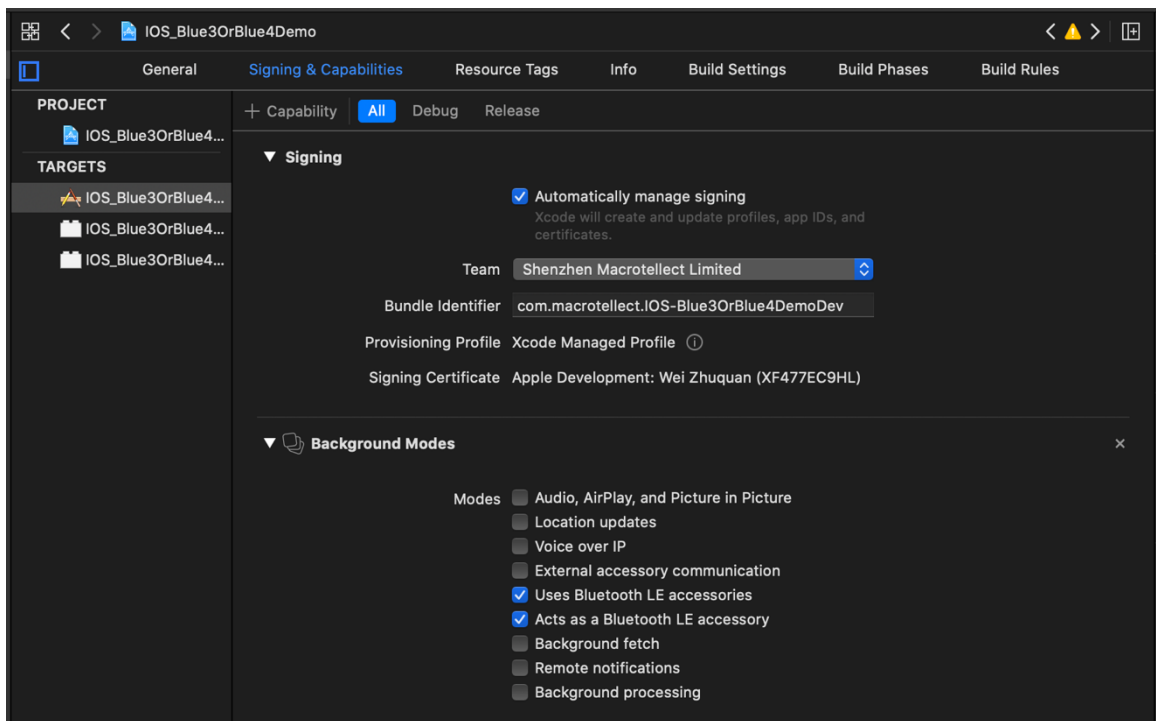
如图：



在TARGETS Build Settings中的link--->Other Linker Flags 中加入：-ObjC



1.2 如果你想让蓝牙可以在后台运行，请如下设置，不需要则不必设置
如图：



第二步:

导入头文件

```
#import "HZLBlueData.h"
```

```
#import "Blue4Manager.h"
```

功能: 接收数据

//蓝牙连接

```
NSArray *blueNames = @[@"BrainLink",@"BrainLink_Pro",@"jii@jii-***"];  
[Blue4Manager logEnable:YES];  
[[Blue4Manager sharedInstance] configureBlueNames:blueNames ableDeviceSum:6];
```

//连接蓝牙成功回调

```
__weak FactoryViewController *weakSelf = self;  
[Blue4Manager sharedInstance].blueConBlock = ^( NSString *markKey){  
    //判断连接的设备 123456 分别对应 ABCDEF  
    if ([markKey isEqualToString:@"1"]){  
        NSLog(@"A 设备蓝牙连接成功");  
    }  
    .....  
    else if ([markKey isEqualToString:@"6"]){  
        NSLog(@"F 设备蓝牙连接成功");  
    }  
};
```

//蓝牙断开回调

```
[Blue4Manager sharedInstance].blueDisBlock = ^( NSString *markKey){  
    //判断断开的设备  
    if ([markKey isEqualToString:@"1"]){  
        NSLog(@"A 设备蓝牙断开");  
    }  
    .....  
    else if ([markKey isEqualToString:@"6"]){  
        NSLog(@"F 设备蓝牙断开");  
    }  
};
```

//蓝牙数据回调

```
[Blue4Manager sharedInstance].hzlblueDataBlock_A = ^(HZLBlueData *blueData, BlueType conBT,  
BOOL isFalseCon) {  
    if (conBT == BlueType_Pro) {
```

```

        if (blueData.bleDataType == BLEMIND) {
            weakSelf.ALabl.text = [NSString stringWithFormat:@"sigal:%d att:%d med:%d
            ele:%d ap:%d del:%d theta:%d lowAlp:%d highAlp:%d lowBe:%d highBe:%d lowGa:%d highGa:%d version:%d
            grid=%d",blueData.signal,blueData.attention,blueData.meditation,blueData.batteryCapacity,blueData.
            ap,blueData.delta,blueData.theta,blueData.lowAlpha,blueData.highAlpha,blueData.lowBeta,blueData.
            highBeta,blueData.lowGamma,blueData.highGamma,blueData.hardwareVersion,blueData.grind];

            //信号值为 0 即佩戴了蓝牙设备
            //注：如果连接了蓝牙设备而未佩戴，信号值为大于 0 且小于或等于 200
            if(blueData.signal == 0){
                weakSelf.ASignalIV.image = [UIImage
            imageNamed:@"signal_zhengChang"];
            }else{
                weakSelf.ASignalIV.image = [UIImage imageNamed:@"signal3.png"];
            }
        }
        else if (blueData.bleDataType == BLEGRAVITY) {
            weakSelf.ACirlleLabl.text = [NSString stringWithFormat:@"x:%d y:%d
            z:%d",blueData.xvlaue,blueData.yvlaue,blueData.zvlaue];
        }
        else if (blueData.bleDataType == BLERaw) {
            weakSelf.ARawLabl.text = [NSString stringWithFormat:@"raw:%d
            eye:%d",blueData.raw,blueData.blinkeye];
        }
    }
    else if (conBT == BlueType_Jii){
        if (blueData.bleDataType == BLEMIND) {
            weakSelf.ALabl.text = [NSString stringWithFormat:@"sigal:%d att:%d med:%d
            ele:%d
            ap:%d",blueData.signal,blueData.attention,blueData.meditation,blueData.batteryCapacity,blueData.ap
            ];

            //信号值为 0 即佩戴了蓝牙设备
            //注：如果连接了蓝牙设备而未佩戴，信号值为大于 0 且小于或等于 200
            if(blueData.signal == 0){
                weakSelf.ASignalIV.image = [UIImage
            imageNamed:@"signal_zhengChang"];
            }else{
                weakSelf.ASignalIV.image = [UIImage imageNamed:@"signal3.png"];
            }
        }
    }
    else if (conBT == BlueType_Lite) {
        if (blueData.bleDataType == BLEMIND) {
            weakSelf.ALabl.text = [NSString stringWithFormat:@"sigal:%d att:%d
            med:%d del:%d theta:%d lowAlp:%d highAlp:%d lowBe:%d highBe:%d lowGa:%d
            highGa:%d",blueData.signal,blueData.attention,blueData.meditation,blueData.delta,blueData.t
            heta,blueData.lowAlpha,blueData.highAlpha,blueData.lowBeta,blueData.highBeta,blueData.lo

```



```

wGamma,blueData.highGamma];
        //信号值为 0 即佩戴了蓝牙设备
        //注：如果连接了蓝牙设备而未佩戴，信号值为大于 0 且小于或等于
200
        if(blueData.signal == 0){
            weakSelf.ASignalIV.image = [UIImage
imageNamed:@"signal_zhengChang"];
        }else{
            weakSelf.ASignalIV.image = [UIImage
imageNamed:@"signal3.png"];
        }
    }
    else if (blueData.bleDataType == BLERaw) {
        weakSelf.ARawLabl.text = [NSString stringWithFormat:@"raw:%d
eye:%d",blueData.raw,blueData.blinkeye];
    }
}

if (isFalseCon) {
    NSLog(@"A 设备假连接");
}
};
.....

[Blue4Manager sharedInstance].hziBlueDataBlock_F = ^(HZLBlueData *blueData, BlueType conBT,
BOOL isFalseCon) {
    .....
};

[[Blue4Manager sharedInstance] connectBlue4];

// 主动断开蓝牙
[[Blue4Manager sharedInstance] disconnectBlue4];

```

HZLBlueData 参考

Overview

该类是数据模型

Enum

```
typedef enum : NSUInteger {
    BlueType_NO = 0,
    BlueType_Lite,
    /*连接的是 BrainLink_Lite 数据格式设备 ,有 BLEMIND、BLERaw 类型数据 */
    BlueType_Pro,
    /*连接的是 BrainLink_Pro 数据格式设备 ,有 BLEMIND、BLEGRAVITY、BLERaw 类型数据 */
    BlueType_Jii,
    /*连接的是 Jii*/
}BlueType;

typedef NS_ENUM(NSUInteger,BLEDATATYPE){
    BLEMIND    =    0,           //脑波数据
    BLEGRAVITY,                 //重力数据
    BLERaw,                     //Raw 眨眼数据
};
```

脑波数据：

- signal, 设备佩戴质量
- attention, 专注度
- meditation, 放松度
- delta,
- theta,
- lowAlpha,
- highAlpha,
- lowBeta,
- highBeta,
- lowGamma,
- highGamma,
- ap, 喜好度
- batteryCapacity, 电池电量百分比
- hardwareVersion, 设备固件版本
- grind

- grind 眨眼
- temperature 温度
- heartrate 心率

重力数据:

- xvlaue,
- yvlaue,
- zvlaue

Raw 眨眼数据:

- raw,
- blinkeye

注释：

连接 Jii , 只有 signal , attention , meditation , batteryCapacity , ap

连接 BrainLink_Lite , 只有 signal , attention , meditation , delta , theta , lowAlpha , highAlpha , lowBeta , highBeta , lowGamma , highGamma , raw , blinkeye

Instructions of some Instance Property

- signal: 信号值。当信号为 0 , 表示已经戴好 , 当信号值为大于 0 且小于等于 200 , 表示硬件和手机通过蓝牙已经连接
- batteryCapacity: 电池容量百分比
- ap: 喜好度
- hardwareVersion: 硬件版本。第一个版本值为 255, 当你更新硬件成功后, 硬件的版本值将会变小
- xvlaue : 重力传感器 X 轴值 前后摆动 俯仰角
- yvlaue : 重力传感器 Y 轴值 左右摆动 偏航角
- zvlaue : 重力传感器 Z 轴值 翅膀摆动 滚转角

Blue4Manager 参考

Overview

该类处理宏智力硬件与蓝牙设备之间的交互

Instance Property

蓝牙连接成功的回调

```
@property (nonatomic,copy)Blue4Connect blueConBlock;
```

蓝牙断开回调

```
@property (nonatomic,copy) BlueConnectdismiss blueDisBlock;
```

Note: 蓝牙设备按照连接顺序依次为 A B C D E F。

使用如上方式，比如有6个数据回调(hzlblueDataBlock_A,hzlblueDataBlock_B)，是为了保证数据的独立性，各个设备间的数据可以同时接受，互不影响。

蓝牙4.0设备最多可以连接6个，可以连接6个但是连接成功比较难。

如果要使用单连接，ableDeviceSum传入参数为1，只调用hzlblueDataBlock_A即可。

各个设备的数据回调

```
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_A;  
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_B;  
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_C;  
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_D;  
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_E;  
@property(nonatomic,copy)Blue4DataBlock hzlblueDataBlock_F;
```

各个设备连接状态

```
@property (nonatomic,assign)BOOL connected_A;  
@property (nonatomic,assign)BOOL connected_B;  
@property (nonatomic,assign)BOOL connected_C;  
@property (nonatomic,assign)BOOL connected_D;  
@property (nonatomic,assign)BOOL connected_E;  
@property (nonatomic,assign)BOOL connected_F;
```

Method

是否打印 log 默认不打印

```
+ (void)logEnable:(BOOL)enable;
```

初始化(单例)

```
+ (instancetype)shareInstance;
```

连接配置

参数说明：

blueNames: 可以连接的设备名称（蓝牙 4.0 设备）

```
NSArray *blueNames = @[@"BrainLink",@"BrainLink_Pro",@"jii@jii-***"];
```

1.jii@jii-表示可连接带 jii-前缀的设备名称 有 jii@表示是 jii 设备 @后面是设备名称 ***表示前缀相同即可

/*! @brief 连接配置(仅用于宏智力公司内部测试)

appSoleCode: app 唯一码

defaultBlueNames:默认的可连接蓝牙名称数组

ableDeviceSum: 可以连接的蓝牙设备个数

result: 返回可以连接的设备名

```
-(void)configureBlueNamesWithAppSoleCode:(NSString *)appSoleCode defaultBlueNames:(NSArray *)defaultBlueNames ableDeviceSum:(int)ableDeviceSum result:(void (^)(NSArray*))result;
*/
```

ableDeviceSum: 可以连接的蓝牙设备个数

```
-(void)configureBlueNames:(NSArray *)blueNames ableDeviceSum:(int)deviceSum
```

连接蓝牙设备

```
-(void)connectBlue4;
```

断开蓝牙设备

```
-(void)disconnectBlue4;
```

手动测试假连接（假连接定义：当 signal 等于 0，attention 和 meditation 的连续 10 个值不变的时候，认为是假连接，SDK 会断开当前设备的蓝牙连接，再次自动连接）

```
-(void)testAFalseCon:(BOOL)isTest; //手动测试 A 设备假连接
```

```
-(void)setTestToZero; //取消所有手动测试假连接
```