iOSBluetoothSDK V1.3_Release

Date: 0723, 2020

Author: Liang Fang

SDK Version: 1.3

MCU: 2.3

Update record:

1. V1.3 fix the bug of failed callback in iOS13.5 system
2. V1.2 supports multiple or single connections
3. V1.0 supports single connection

Content

iOSBluetoothSDK Development Guide

Introduction

This guide will teach you how to use HZLBlueTooth SDK for iOS to write iOS applications that can acquire brainwave data from Macrotellect 's Hardware（BrainLink Pro & BrainLink Lite）. This will enable your iOS apps to receive and use brainwave data such as BLEMIND and BLEGRAVITY acquired via Bluetooth, Macrotellect 's Hardware and File source encapsulated as HZLBlueTooth. HZLBlueTooth SDK for iOS supports upgrading Hardware

Function:
Receive brainwave data. One or more Bluetooth devices can be connected at the same time.

Files included:

- API Reference (this document)
- SDK static library and headers
- iOSBluetoothSDK V1.3_Release.a
- HZLBlueData.h
- Blue3OrBlue4Manager.h
- iOSBluetoothSDK V1.3_Release Demo

Supported devices:

- Data format with power
  - BrainLink_Pro
  - Jii
- Data format without power
  - BrainLink_Lite
  - Mind Link

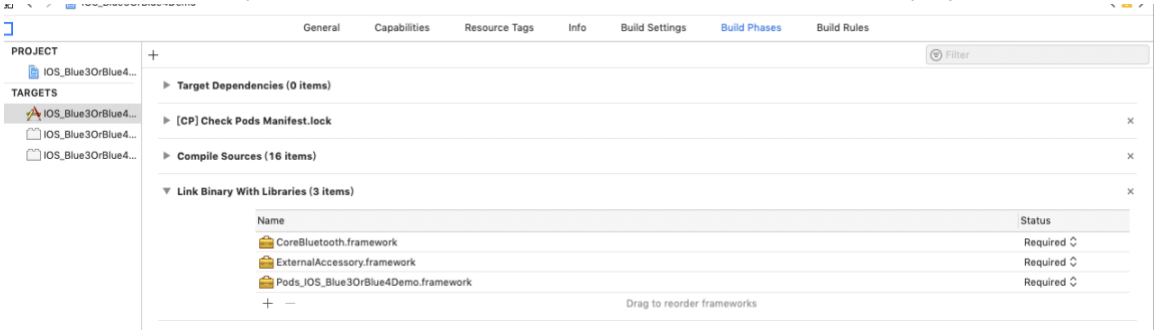
iOS Verrsion :

- iOS 9.0 +

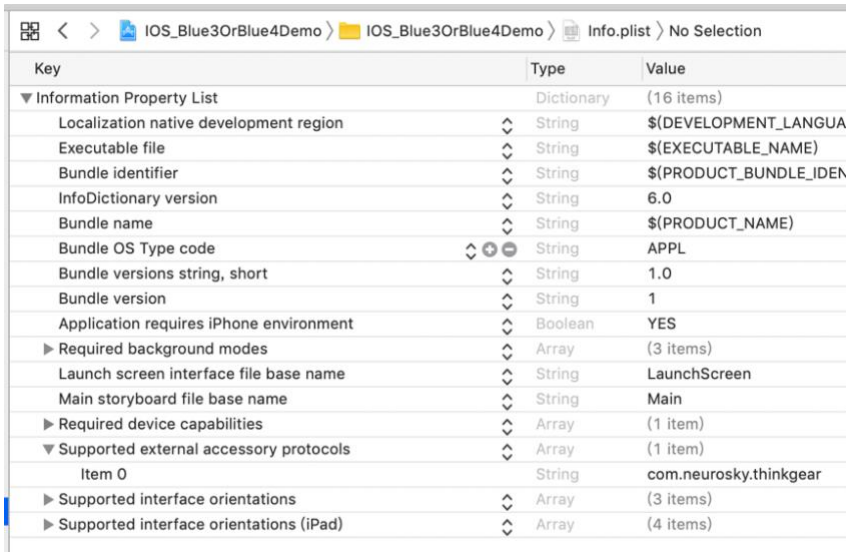
Your First Project: iOSBluetoothSDK V1.3_Release Demo

Step 1:

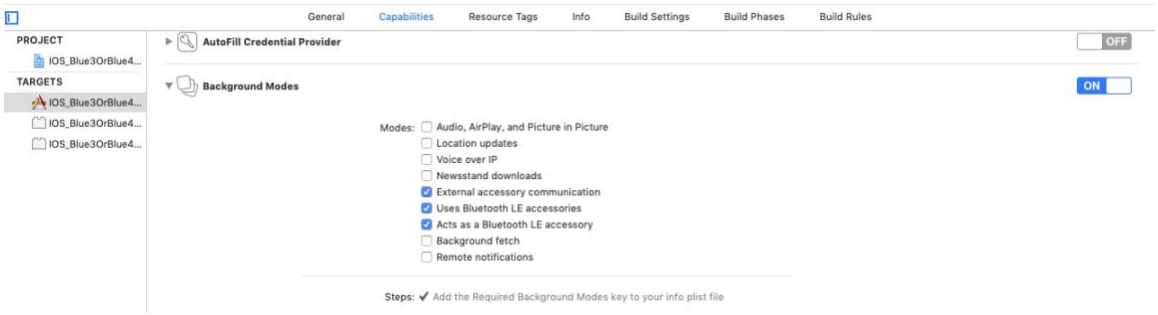
1.1 Import the IOS framework libraries CoreBluetooth.framework
and ExternalAccessory.framework in the Build Phases of TARGETS in the Xcode project:



Add com.neurosky.thinkgear In the Info.plist: (ios13 needs to add the Bluetooth permission privacy - Bluetooth always usage description and privacy - Bluetooth peripheral usage description)



1.2 If you want Bluetooth to work in the background, please set it as follows，
Don't set it if you don't need it:



Step 2:

Import header file

```
#import "HZLBlueData.h"
#import "Blue3OrBlue4Manager.h"
```

Function one：Receive data

```
// Bluetooth multi-connection
// blueNames: connectable device
NSArray *blueNames = @[@"BrainLink",@"BrainLink_Pro",@"jii@jii-***"];
[Blue3OrBlue4Manager logEnable:YES];
[[Blue3OrBlue4Manager shareInstance] configureBlueNames:blueNames ableDeviceSum:7];

// Connect bluetooth callback successfully
  __weak FactoryViewController *weakSelf = self;
[Blue3OrBlue4Manager shareInstance].blueConBlock =  ^(NSStrig * markKey){
      // Determine connected devices    123456 corresponds to ABCDEF respectively
      if ([markKey isEqualToString:@"1"]) {
            NSLog(@"Bluetooth device A is connected");
       }
      .....
      else if ([markKey isEqualToString:@"6"]){
            NSLog(@"Bluetooth device F is connected");
      }
};


// Bluetooth disconnect callback
[Blue3OrBlue4Manager shareInstance].blueDisBlock =  ^(NSStrig * markKey){
// Determine disConnected devices
      if ([markKey isEqualToString:@"1"]) {
          NSLog(@"Bluetooth device A is disConnected");
       }
```

```
            .....
            else if ([markKey isEqualToString:@"6"]){
                NSLog(@"Bluetooth device F is disConnected");
            }
        }
    };

  //蓝牙数据回调
    [Blue3OrBlue4Manager shareInstance].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];
                // when the signal value is 0, the bluetooth device is worn
                // note: if the bluetooth device is connected but not worn, Greater than 0 and less than or equal to 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.ACircleLabl.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];

                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.theta,blueData.lowAlpha,blueData.highAlpha,blueData.lowBeta,blueData.highBeta,blueData.lowGamma,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 device has a false connection");
        }
    };
    ......

    [Blue3OrBlue4Manager shareInstance].hzlblueDataBlock_G = ^(HZLBlueData *blueData, BlueType conBT, BOOL isFalseCon) {
        ......
    };


    [[Blue3OrBlue4Manager shareInstance] connectBlue3OrBlue4];

    // Active bluetooth disconnect
    [[Blue3OrBlue4Manager shareInstance]disConnectBlue3OrBlue4];
```

iOSBluetoothSDK V1.3 API Reference

HZLBlueData Reference

Overview
The HZLBlueData class is a data model

Enum
```
typedef enum : NSUInteger {
      BlueType_NO = 0,
      BlueType_Lite,
/*The current connection is the BrainLink_Lite data format device with BLEMIND and BLERaw type data*/
      BlueType_Pro,
/* The current connection is the BrainLink_Pro data format device with BLEMIND, BLEGRAVITY, and BLERaw type data*/
      BlueType_Jii,
/* The current connection is Jii*/
}BlueType;

 typedef NS_ENUM(NSUInteger,BLEDATATAYPE){
            BLEMIND   =    0,              // basic brain wave data
            BLEGRAVITY,                   // gravity data
            BLERaw,                       // blink data
};
```

Basic Brainwave Data：
- signal,
- attention,
- meditation,
- delta,
- theta,
- lowAlpha,
- highAlpha,
- lowBeta,
- highBeta,
- lowGamma,
- highGamma,
- ap,
- batteryCapacity,
- hardwareVersion,
- grind

Gravity Sensor Data:

- xvlaue,
- yvlaue,
- zvlaue

Raw& Blink Data:

- raw,
- blinkeye

Note：

When Jii is connected, only signal, attention, meditation, batteryCapacity, ap data type is available.

When BrainLink_Lite is connected, only signal，attention，meditation，delta，theta，lowAlpha，highAlpha，lowBeta，highBeta，lowGamma， highGamma，raw，blinkeye data type is available.

Instructions of some Instance Property

- **signal:** It represents the signal value of the Macrotellect 's Hardware. When the signal is 0, it means that the Macrotellect 's Hardware has been put on, and when the signal is greater than 0 and less than or equal to 200, it means that the Macrotellect 's Hardware is connected to the iPhone.
- **batteryCapacity:** In percentage terms. minimum value is 0, maximum value is 100
- **ap:** Appreciation value
- **hardwareVersion:** Hardware version. The first version value is 255，when you update the Macrotellect 's Hardware，the version value will be smaller.
- **xvlaue**： gravity value in The x axis (Pitching Angle)
- **yvlaue**： gravity value in The y axis (Yaw Angle)
- **zvlaue**： gravity value in The z axis (Roll Angle)

Blue3OrBlue4ManagerReference

Overview

The Blue3OrBlue4Manager class handles interaction between a Macrotellect's Hardware and an iOS device.

Instance Property

Successful callback of bluetooth connection

```
@property      (nonatomic,copy)Blue3OrBlue4Connect blueConBlock;
```

Bluetooth disconnect callback

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

Note: devices are connected in the order of A B C D E F. Six data callbacks (hzluedatablock_A ...) are used to ensure the independence of data. The data between various devices can be accepted at the same time without mutual influence. Bluetooth 4.0 devices can connect up to six, but it is difficult to connect successfully.
If you want to use a single connection, the input parameter of abledevicesum is 1. Only call hzluedatablock_A.

Data callback for every device.

```
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_A;
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_B;
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_C;
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_D;
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_E;
@property(nonatomic,copy)Blue3OrBlue4DataBlock    hzlblueDataBlock_F;
```

Connection status of every device

```
@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

Print log does not print by default

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

Initialization (singleton)

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

Parameter Configuration：

Parameter interpretation：

blueNames: **Device name (Bluetooth 4.0 device) or MFI (Bluetooth 3.0 device) that can be connected**

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

1.jii@jii-***： indicates the device name with the prefix of jii - that can be connected ，'jii@' means jii device. The content After @ is the device name. '***' indicates the prefix is the same.

2.BrainLink： Indicates the device name or MFI called brainlink that can be connected.

ableDeviceSum: Number of Bluetooth devices that can be connected

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

Connect bluetooth device

-(void)connectBlue3OrBlue4;

Disconnect bluetooth device

-(void)disConnectBlue3OrBlue4;

Manual test of false connection (definition of false connection: when signal is equal to 0 and the continuous 10 values of attention and mediation remain unchanged, it is considered to be a false connection. SDK will disconnect the Bluetooth connection of the current device and automatically connect again)

-(void)testAFalseCon:(BOOL)isTest; //Manual test A device false connection

-(void)setTestToZero;//Cancel all manual test false connections