# Unity3DBrainLinkProSDK V1.0.5

Date: 0402 2023

Author: Liang Fang SDK Version: 1.0.5

MCU: 4.201

更新记录：

1. V1.0.5 手动扫描
2. V1.0.4 增加喜好度和角度值的开关
3. V1.0.3 增加新的蓝牙数据字段 HRV
4. V1.0.1 Android SDK 优化，添加详细开发步骤
5. V1.0.0 Unity3D移动端SDK

**目录**

[Unity3DBrainLinkProSDK V1.0.5 1](#_Toc131342450)

[Unity3D BrainLinkProSDK开发指南 4](#_Toc131342451)

[介绍 4](#_Toc131342452)

[iOS端配置: 5](#_Toc131342453)

[Android端配置: 13](#_Toc131342454)

[Unity3D BrainLinkProSDK V1.0.5 iOS端 API参考 16](#_Toc131342455)

[HZLBlueData 参考 16](#_Toc131342456)

[Blue4Manager 参考 19](#_Toc131342457)

[Unity3D BrainLinkProSDK V1.0.5 Android端 API参考 21](#_Toc131342458)

## Unity3D BrainLinkProSDK开发指南

### 介绍

本指南将教你如何使用Unity3D BrainLinkProSDK从宏智力公司的硬件中获取脑电波数据。这将使您的移动端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Unity3D BrainLinkProSDK来获取他们。

**功能:**

接收脑波数据。

**文件包含：**

* API参考(此文档)
* Asset/iOS
* Asset/Android

**支持的硬件设备：**

* 有电量的数据格式
  + BrainLink\_Pro

**支持的iOS / Android版本：**

* iOS 9.0 + / Android SDK 26+

**支持的Unity3D版本：**

* Unity2019+

**Unity3D中使用方法：**

* 在场景中新建物体“ThinkGearManager”，挂载ThinkGearMnanger的脚本
* 调用方法见Demo场景以及Demo.cs脚本

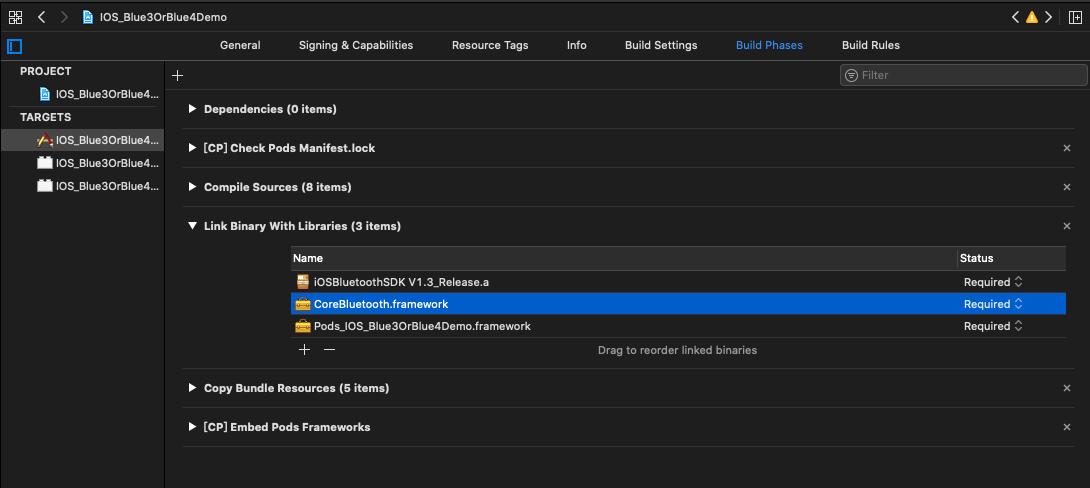
### iOS端配置:

**第一步:**

1.1 在Xcode项目里 TARGETS – Build Phases 导入IOS系统框架库如下

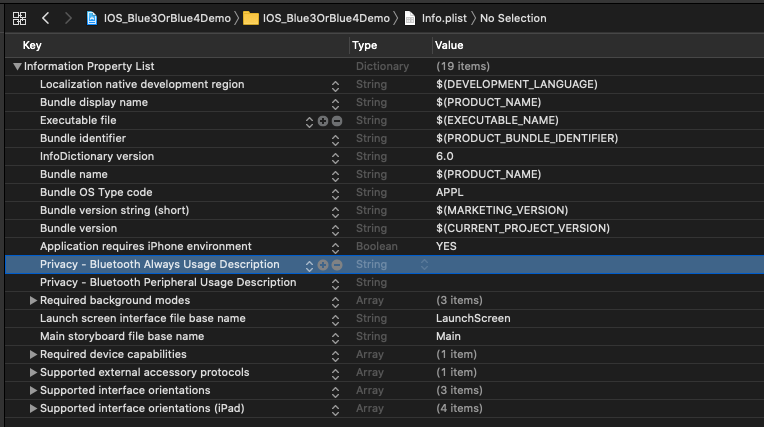
* CoreBluetooth.framework

如图：

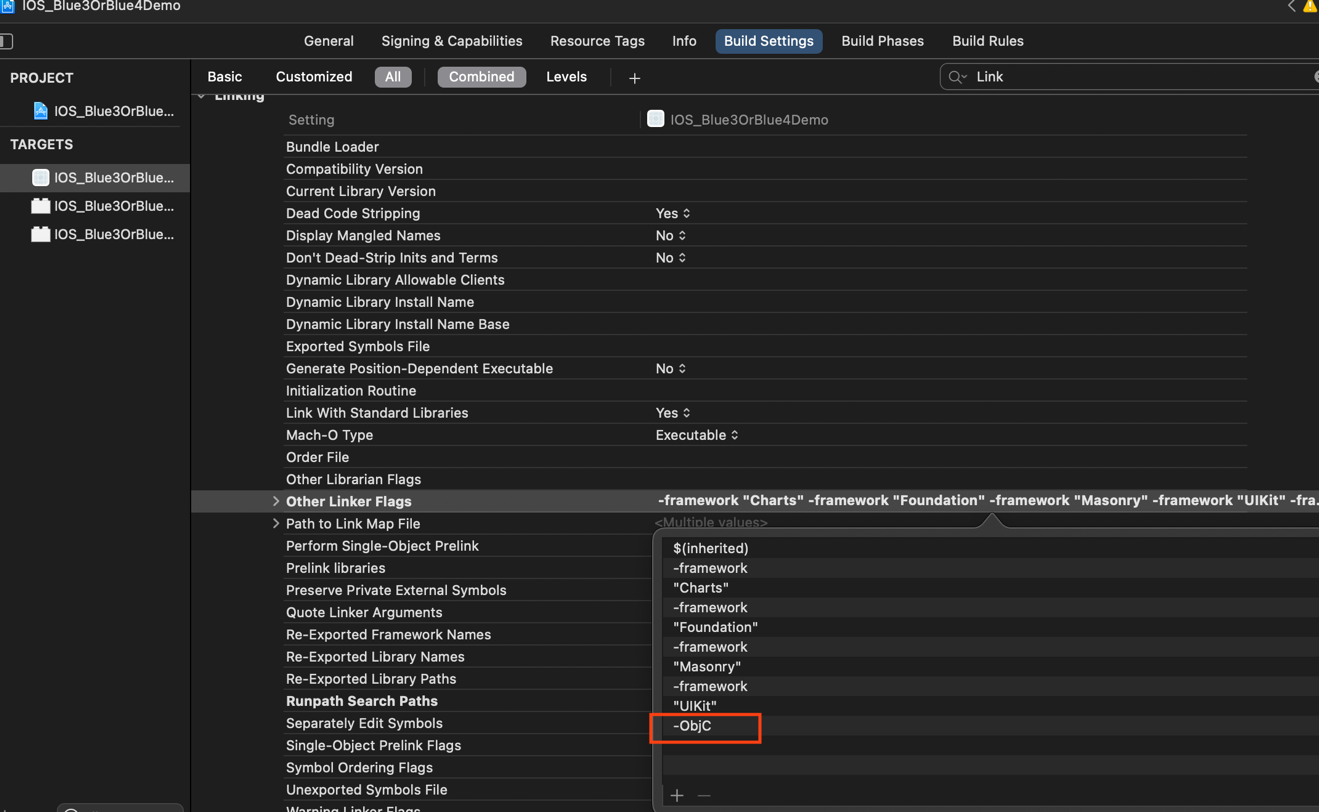
****

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：

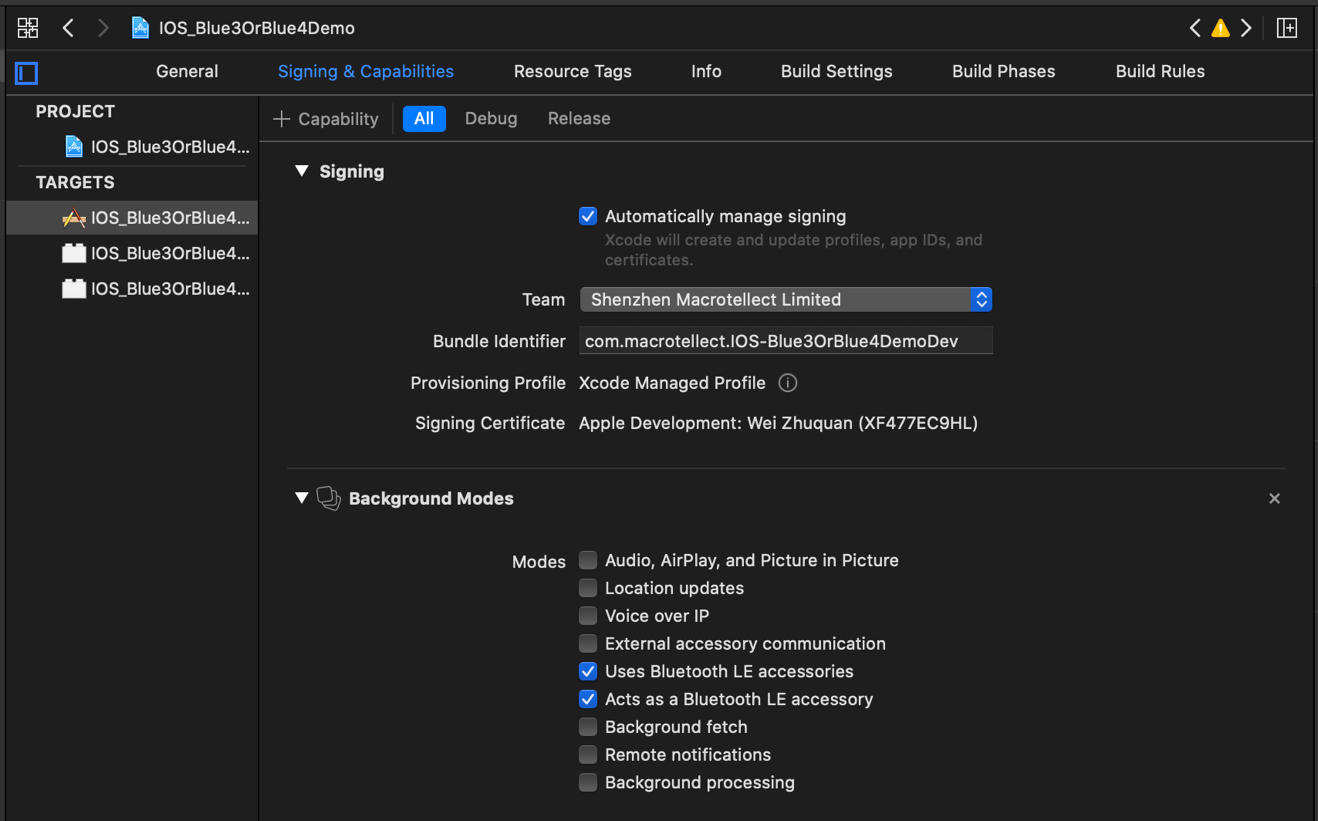
****

在所有TARGETS Build Settings中的link--->Other Linker Flags 中加入：-ObjC，Bitcode设置为No



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:**

**在UnityAppController.mm文件中**

导入头文件

#import "Blue4Manager.h"

添加属性

NSMutableArray \*\_refreshDevices;

在文件末尾添加方法

extern "C"{

void SendSettings(char \*settings){

NSData \*data = [[NSString stringWithCString:settings encoding:NSUTF8StringEncoding] dataUsingEncoding:NSUTF8StringEncoding];

[[Blue4Manager shareInstance] writeToProAWithCode:data];

}

void SetWhiteList(char \*whiteList){

\_refreshDevices = [NSMutableArray array];

NSString \*list = [NSString stringWithCString:whiteList encoding:NSUTF8StringEncoding];

NSArray \*blueNames = [list componentsSeparatedByString:@","];

//蓝牙连接设置

[Blue4Manager logEnable:YES];

[[Blue4Manager shareInstance] configureBlueNames:blueNames ableDeviceSum:1];

[[Blue4Manager shareInstance] bluePermission:^(BluePermission\_state state, NSString \*log) {

NSLog(@"BluePermission\_state====%@",log);

if (state == Blue4\_Unauthorized) {

//未授权

}

else if (state == Blue4\_PoweredOn) {

//蓝牙 开

[\_refreshDevices removeAllObjects];

[[Blue4Manager shareInstance] scanBlue4WithScannedWithScanTime:122 blue4DeviceBlock:^(ScannedDevice \*blue4Device) {

BOOL isFind = false;

for (ScannedDevice \*ss in \_refreshDevices) {

if ([ss.peripheral.identifier.UUIDString isEqualToString: blue4Device.peripheral.identifier.UUIDString]) {

isFind = true;

}else{

}

}

if(!isFind){

[\_refreshDevices addObject:blue4Device];

NSLog(@"==name===%@", blue4Device.name);

NSLog(@"==identifier===%@", blue4Device.peripheral.identifier);

}

NSString \*nameIdentifierRiss = [NSString stringWithFormat:@"%@,%@,%d",blue4Device.name,blue4Device.peripheral.identifier,blue4Device.RSSI.intValue];

UnitySendMessage("ThinkGearManager", "DeviceFound", [nameIdentifierRiss cStringUsingEncoding:NSUTF8StringEncoding]);

}];

}

else if (state == Blue4\_PoweredOff) {

//蓝牙 关

}

}];

[Blue4Manager shareInstance].endScanBlue4Block = ^{

NSLog(@"终止扫描");

};

//蓝牙连接成功

[Blue4Manager shareInstance].blueConBlock = ^(NSString \*markKey) {

if ([markKey isEqualToString:@"1"]) {

//判断连接的设备

NSLog(@"A设备 蓝牙 连接成功");

UnitySendMessage("ThinkGearManager", "ReceiveContentState", "yes");

}

};

//蓝牙断开回调

[Blue4Manager shareInstance].blueDisBlock = ^(NSString \*markKey){

if ([markKey isEqualToString:@"1"]) {

//判断连接的设备

NSLog(@"A设备 蓝牙 断开");

UnitySendMessage("ThinkGearManager", "ReceiveContentState", "no");

// UnitySendMessage("ThinkGearManager", "ReceiveBlueToothType", "");

}

};

//第一个设备(A)数据回调 其他设备数据回调如hzlblueDataBlock\_B与hzlblueDataBlock\_A的写法相同

[Blue4Manager shareInstance].hzlblueDataBlock\_A = ^(HZLBlueData \*blueData, BlueType conBT, BOOL isFalseCon) {

if (conBT == BlueType\_Pro) {

if (blueData.bleDataType == BLEMIND) {

//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

// UnitySendMessage("ThinkGearManager", "ReceiveBlueToothType", "4\_0");

UnitySendMessage("ThinkGearManager", "ReceivePoorSignal", [[NSString stringWithFormat:@"%d",blueData.signal] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveBatteryCapacity", [[NSString stringWithFormat:@"%d",blueData.batteryCapacity] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveAttention", [[NSString stringWithFormat:@"%d",blueData.attention] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveMeditation", [[NSString stringWithFormat:@"%d",blueData.meditation] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveDelta", [[NSString stringWithFormat:@"%d",blueData.delta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveTheta", [[NSString stringWithFormat:@"%d",blueData.theta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowAlpha", [[NSString stringWithFormat:@"%d",blueData.lowAlpha] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighAlpha", [[NSString stringWithFormat:@"%d",blueData.highAlpha] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowBeta", [[NSString stringWithFormat:@"%d",blueData.lowBeta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighBeta", [[NSString stringWithFormat:@"%d",blueData.highBeta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowGamma", [[NSString stringWithFormat:@"%d",blueData.lowGamma] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighGamma", [[NSString stringWithFormat:@"%d",blueData.highGamma] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHeaetRate", [[NSString stringWithFormat:@"%d",[blueData.heartRate intValue]] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveTemperature", [[NSString stringWithFormat:@"%f",[blueData.temperature floatValue]] cStringUsingEncoding:NSUTF8StringEncoding]);

NSString \*hrvStr = @"";

if (blueData.HRV != nil) {

for (int i = 0; i < blueData.HRV.count; i++) {

if(i >= 1){

hrvStr = [hrvStr stringByAppendingString:[NSString stringWithFormat:@",%dms",[blueData.HRV[i] intValue]]];

}else{

hrvStr = [hrvStr stringByAppendingString:[NSString stringWithFormat:@"%dms",[blueData.HRV[i] intValue]]];

}

}

UnitySendMessage("ThinkGearManager", "ReceiveHRV", [hrvStr cStringUsingEncoding:NSUTF8StringEncoding]);

}

UnitySendMessage("ThinkGearManager", "ReceiveGrind4\_0", [[NSString stringWithFormat:@"%d",[blueData.grind intValue]] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveAp4\_0", [[NSString stringWithFormat:@"%d",blueData.ap] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHardwareversion4\_0", [blueData.hardwareVersion cStringUsingEncoding:NSUTF8StringEncoding]);

}

else if (blueData.bleDataType == BLEGRAVITY) {

UnitySendMessage("ThinkGearManager", "ReceiveXValue", [[NSString stringWithFormat:@"%d",blueData.xvlaue] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveYValue", [[NSString stringWithFormat:@"%d",blueData.yvlaue] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveZValue", [[NSString stringWithFormat:@"%d",blueData.zvlaue] cStringUsingEncoding:NSUTF8StringEncoding]);

}

else if (blueData.bleDataType == BLERaw) {

UnitySendMessage("ThinkGearManager", "ReceiveRawdata", [[NSString stringWithFormat:@"%d",blueData.raw] cStringUsingEncoding:NSUTF8StringEncoding]);

}

}

else if (conBT == BlueType\_Jii){

if (blueData.bleDataType == BLEMIND) {

UnitySendMessage("ThinkGearManager", "ReceiveAttention", [[NSString stringWithFormat:@"%d",blueData.attention] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveMeditation", [[NSString stringWithFormat:@"%d",blueData.meditation] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveBatteryCapacity", [[NSString stringWithFormat:@"%d",blueData.batteryCapacity] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveBlutToothType", "4\_0");

}

}

else if (conBT == BlueType\_Lite) {

// UnitySendMessage("ThinkGearManager", "ReceiveBlueToothType", "");

if (blueData.bleDataType == BLEMIND) {

UnitySendMessage("ThinkGearManager", "ReceivePoorSignal", [[NSString stringWithFormat:@"%d",blueData.signal] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveAttention", [[NSString stringWithFormat:@"%d",blueData.attention] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveMeditation", [[NSString stringWithFormat:@"%d",blueData.meditation] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveBatteryCapacity", [[NSString stringWithFormat:@"%d",0] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveDelta", [[NSString stringWithFormat:@"%d",blueData.delta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveTheta", [[NSString stringWithFormat:@"%d",blueData.theta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowAlpha", [[NSString stringWithFormat:@"%d",blueData.lowAlpha] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighAlpha", [[NSString stringWithFormat:@"%d",blueData.highAlpha] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowBeta", [[NSString stringWithFormat:@"%d",blueData.lowBeta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighBeta", [[NSString stringWithFormat:@"%d",blueData.highBeta] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveLowGamma", [[NSString stringWithFormat:@"%d",blueData.lowGamma] cStringUsingEncoding:NSUTF8StringEncoding]);

UnitySendMessage("ThinkGearManager", "ReceiveHighGamma", [[NSString stringWithFormat:@"%d",blueData.highGamma] cStringUsingEncoding:NSUTF8StringEncoding]);

}

else if (blueData.bleDataType == BLERaw) {

UnitySendMessage("ThinkGearManager", "ReceiveRawdata", [[NSString stringWithFormat:@"%d",blueData.raw] cStringUsingEncoding:NSUTF8StringEncoding]);

}

}

if (isFalseCon) {

NSLog(@"A设备假连接");

}

};

[[Blue4Manager shareInstance] connectBlue4WithIsAuto:NO];

}

void Scan(){

[\_refreshDevices removeAllObjects];

[[Blue4Manager shareInstance] scanBlue4WithScannedWithScanTime:122 blue4DeviceBlock:^(ScannedDevice \*blue4Device) {

BOOL isFind = false;

for (ScannedDevice \*ss in \_refreshDevices) {

if ([ss.peripheral.identifier.UUIDString isEqualToString: blue4Device.peripheral.identifier.UUIDString]) {

isFind = true;

}else{

}

}

if(!isFind){

[\_refreshDevices addObject:blue4Device];

NSLog(@"==name===%@", blue4Device.name);

NSLog(@"==identifier===%@", blue4Device.peripheral.identifier);

}

NSString \*nameIdentifierRiss = [NSString stringWithFormat:@"%@,%@,%d",blue4Device.name,blue4Device.peripheral.identifier,blue4Device.RSSI.intValue];

UnitySendMessage("ThinkGearManager", "DeviceFound", [nameIdentifierRiss cStringUsingEncoding:NSUTF8StringEncoding]);

}];

}

void ConnectDevice(char \*identifier){

for (ScannedDevice \*device in \_refreshDevices) {

if ([device.peripheral.identifier.UUIDString isEqualToString:[NSString stringWithCString:identifier encoding:NSUTF8StringEncoding]]) {

[[Blue4Manager shareInstance] manuallyConnetBlue4ForScannedDevice:device];

}

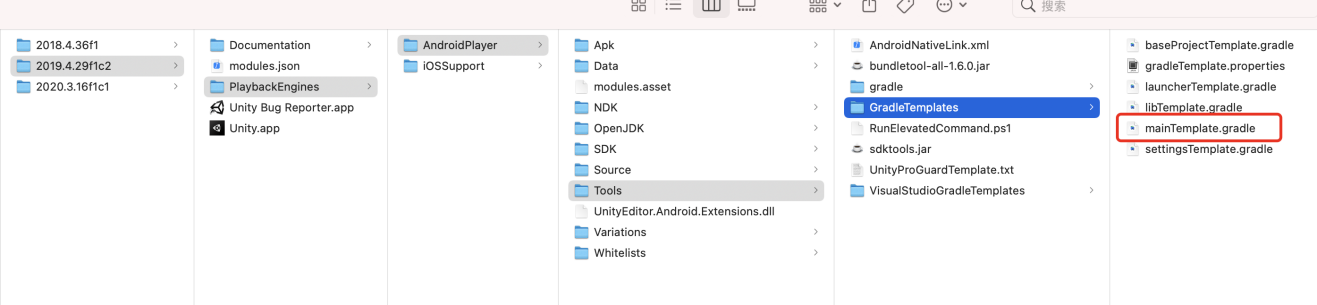
}

}

}

### Android端配置:

1.在unity安装目录中找到mainTemplate.gradle



添加：implementation 'androidx.appcompat:appcompat:1.1.0'

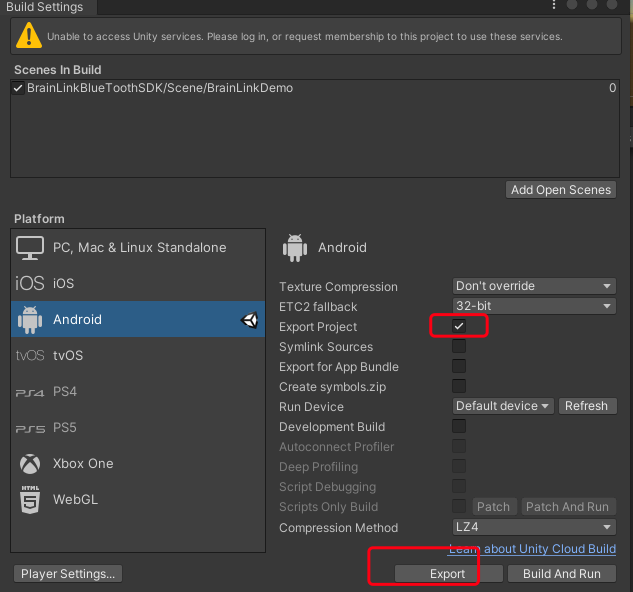
dependencies {

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation 'androidx.appcompat:appcompat:1.1.0'

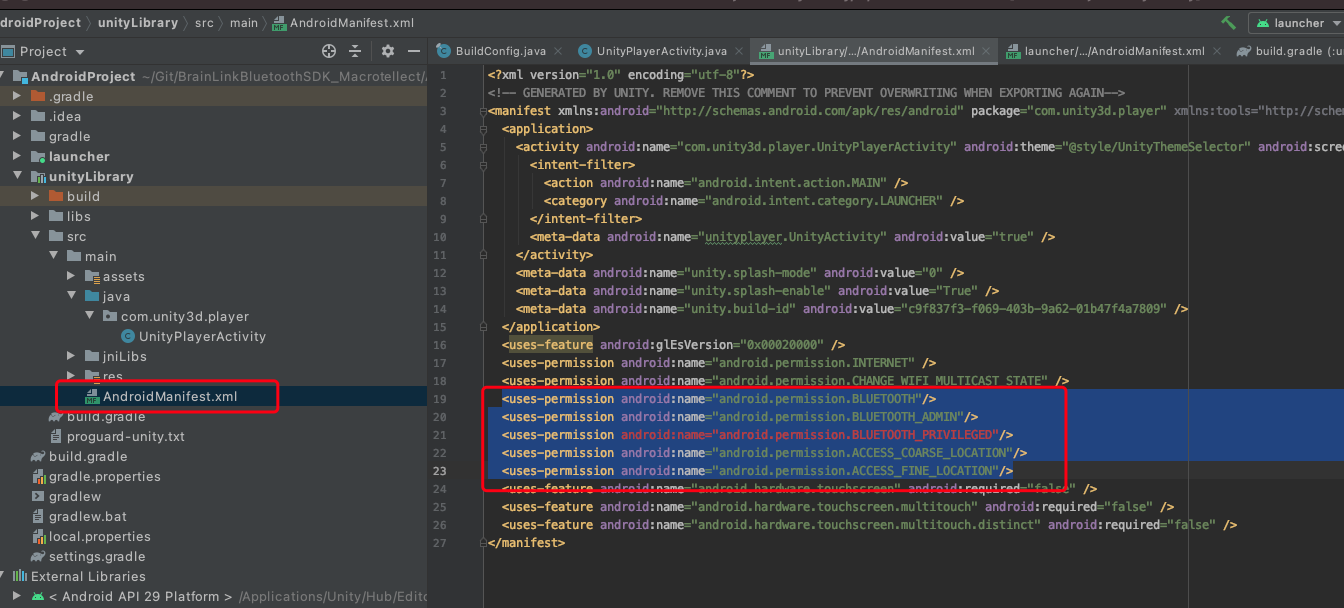
\*\*DEPS\*\*}

1. 导出unity的android项目 如下图



1. 在导出的android项目中，unityLibrary目录下的AndroidManifest.xml里添加 蓝牙、定位权限，如下

<uses-permission android:name="android.permission.BLUETOOTH"/>  
<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>  
<uses-permission android:name="android.permission.BLUETOOTH\_PRIVILEGED"/>  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>



在UnityPlayerActivity里添加授权代码，如下

requestPermissions();

String[] permissions = new String[]{

        android.Manifest.permission.*BLUETOOTH*,

        Manifest.permission.*ACCESS\_COARSE\_LOCATION*,

        Manifest.permission.*ACCESS\_FINE\_LOCATION*

};

private void requestPermissions() {

    ArrayList<String> permissionList = new ArrayList<>();

    for (int i = 0; i < permissions.length; i++) {

        if (ActivityCompat.*checkSelfPermission*(UnityPlayerActivity.this, permissions[i]) != PackageManager.*PERMISSION\_GRANTED*) {

            permissionList.add(permissions[i]);

        }

    }

    if (!permissionList.isEmpty()) {

        String[] mPermissions = permissionList.toArray(new String[permissionList.size()]);

        ActivityCompat.*requestPermissions*(UnityPlayerActivity.this, mPermissions, 100);

    }

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

    if (requestCode == 100) {

        for (int i = 0; i < grantResults.length; i++) {

            if (grantResults[i] == PackageManager.*PERMISSION\_DENIED*) {

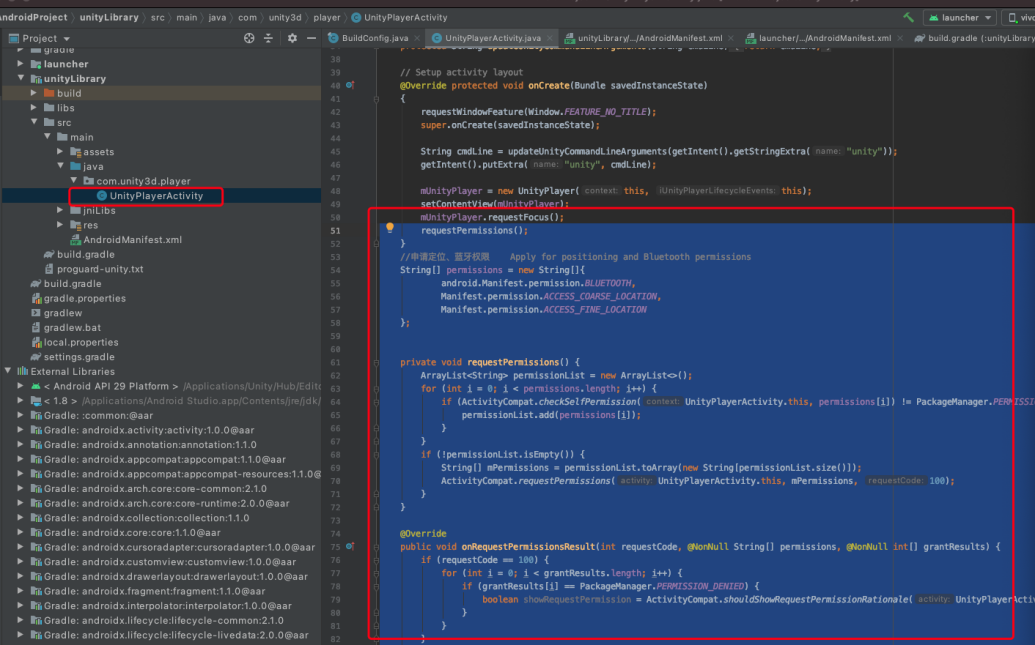
                boolean showRequestPermission = ActivityCompat.*shouldShowRequestPermissionRationale*(UnityPlayerActivity.this, permissions[i]);

            }

        }

    }

}



## Unity3D BrainLinkProSDK V1.0.5 iOS端 API参考

### 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,BLEDATATAYPE){

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
* HRV 心率变异性

**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

sdkVersion: sdk版本号

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 sdkVersion:(NSString \*)sdkVersion 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medition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

## Unity3D BrainLinkProSDK V1.0.5 Android端 API参考

UnityThinkGear.cs脚本中

SetBLLinstenner(string objectName) 此方法开启监听，参数为挂载接收回调方法脚本的物体名，本Demo中为ThinkGearManager，开启监听之后调用ConnectBluetooth()连接方法

回调方法在ThinkGearManager.cs脚本中ReceiveXX