Unity3DBrainLinkProSDK V1.0.0

Date: 0621 2022

Author: Liang Fang

SDK Version: 1.0.0

MCU: 3.1

更新记录：

1. V1.0.0　Unity3D 移动端 SDK

目录

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

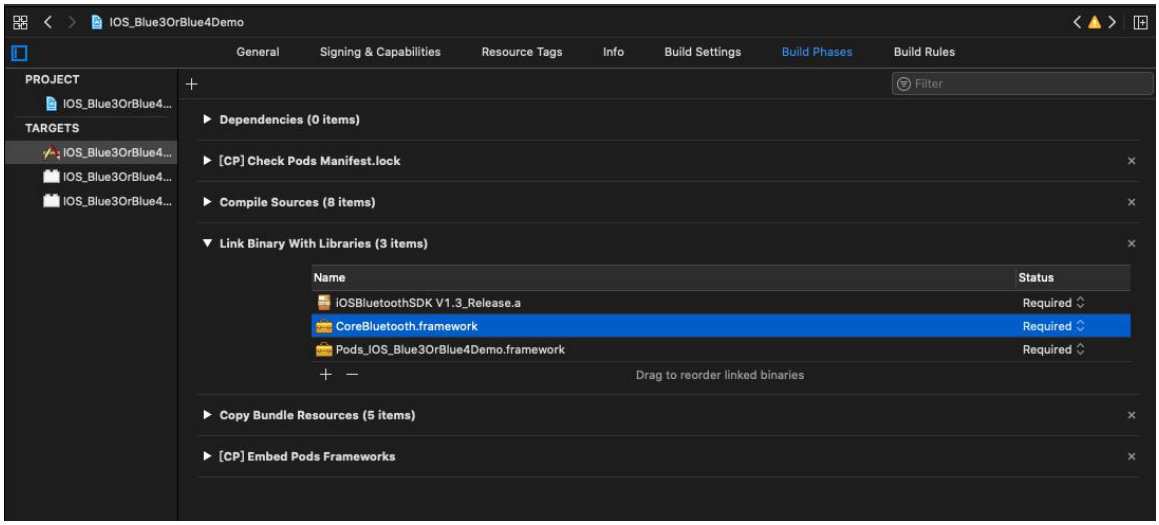
iOS 端配置：

第一步:

1.1 在 Xcode 项目里 TARGETS 一 Build Phases 导入 IOS 系统框架库如下

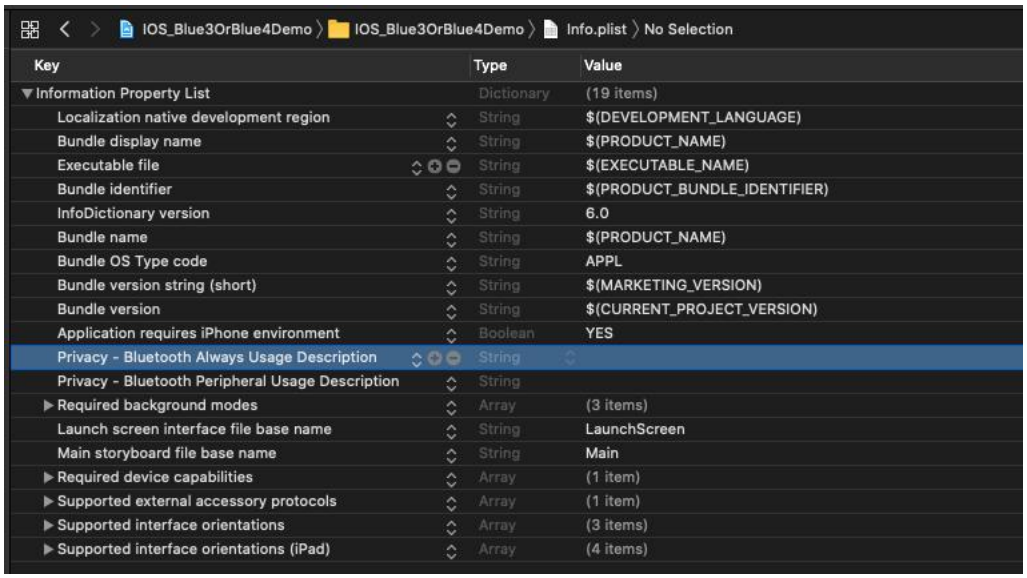
- CoreBluetooth.framework

如图：

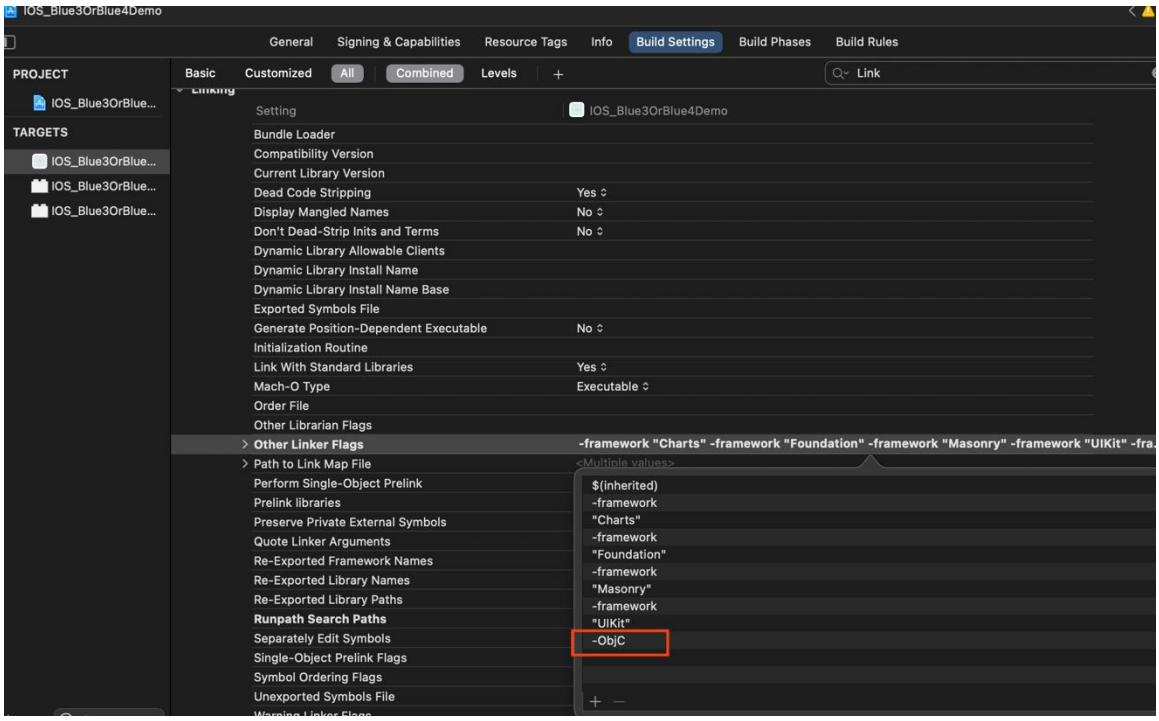


在 Info.plist 里添加蓝牙权限 （IOS13 需要添加蓝牙权限 Privacy — Bluetooth Always Usage Description，Privacy — Bluetooth Peripheral Usage Descriptio）

如图：

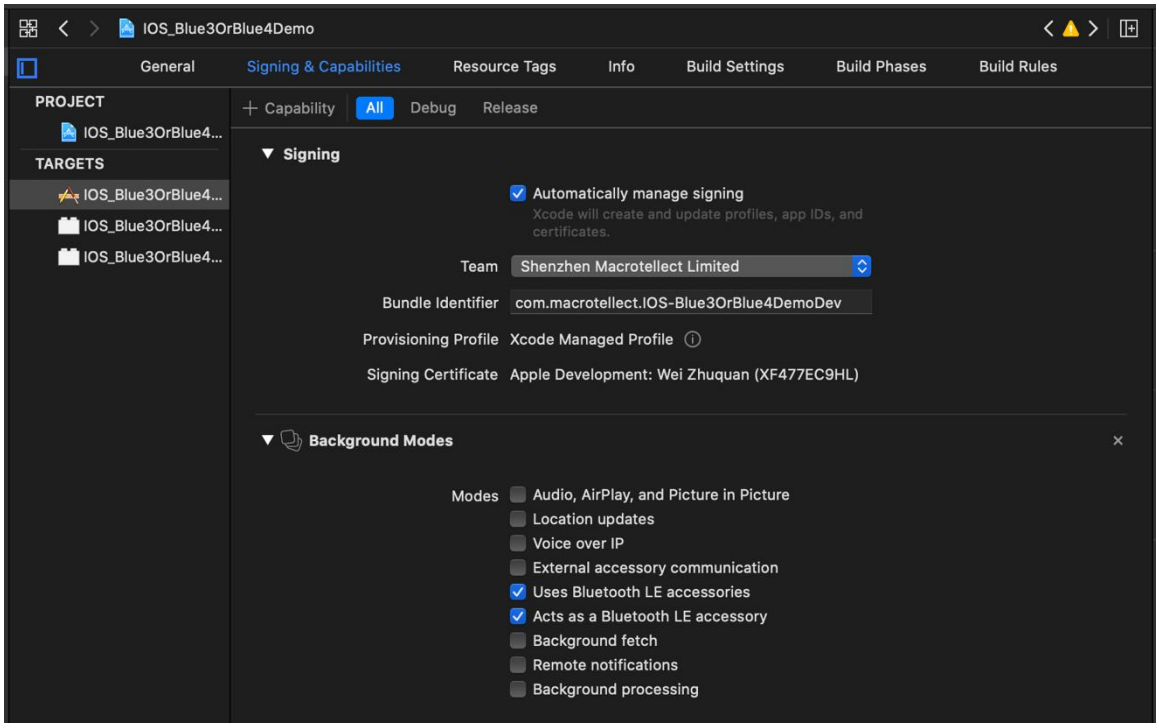


在TARGETS Build Settings中的link--->Other Linker Flags 中加入：-ObjC



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:

在UnityAppController.mm文件中

导入头文件

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

在文件末尾添加方法

```
extern "C"{
    void Connect()
    {
        //蓝牙连接设置
        [Blue4Manager logEnable:YES];

                                                    [[Blue4Manager shareInstance]
configureBlueNames:@[@"BrainLink_Pro",@"BrainLink_Lite",@"BrainLink",@"BrainLink_Lite_P",@"Brainl
ink_Lite",@"ROYWOS",@"BrainLink_Pink"] ableDeviceSum:1];
        //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
```

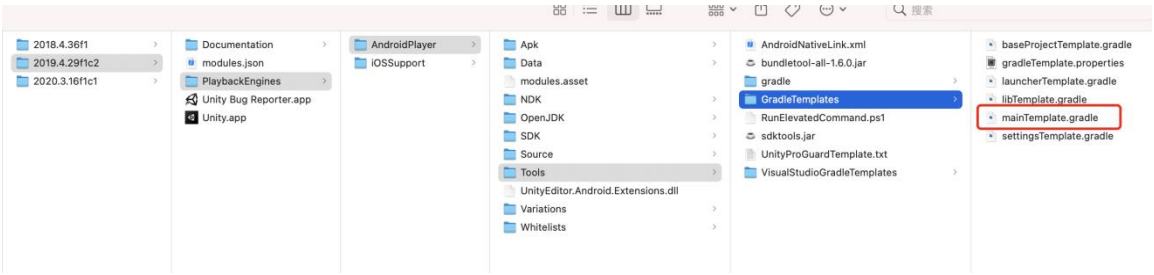
```
stringWithFormat:@"%d",blueData.highBeta] cStringUsingEncoding:NSUTF8StringEncoding]);
                                UnitySendMessage("ThinkGearManager", "ReceiveLowGamma", [[NSString
stringWithFormat:@"%d",blueData.lowGamma] cStringUsingEncoding:NSUTF8StringEncoding]);
                       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

                                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] connectBlue4];



}

void disConnect(){
```

```
        [[Blue4Manager shareInstance] disConnectBlue4];
    }

}
```

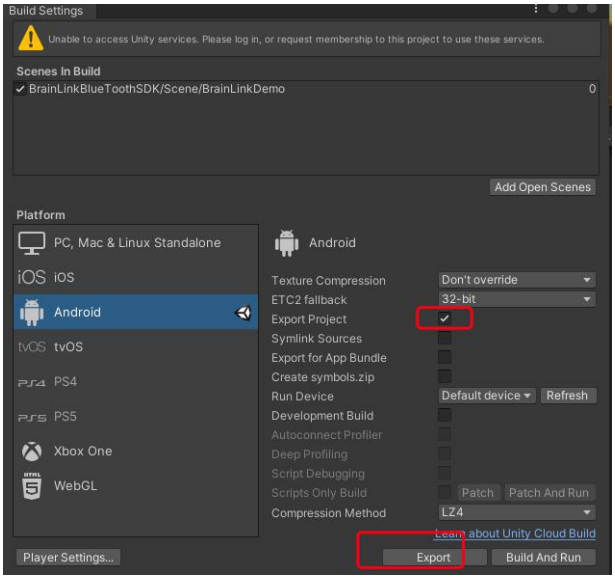
Android 端配置：

1.在 unity 安装目录中找到 mainTemplate.gradle



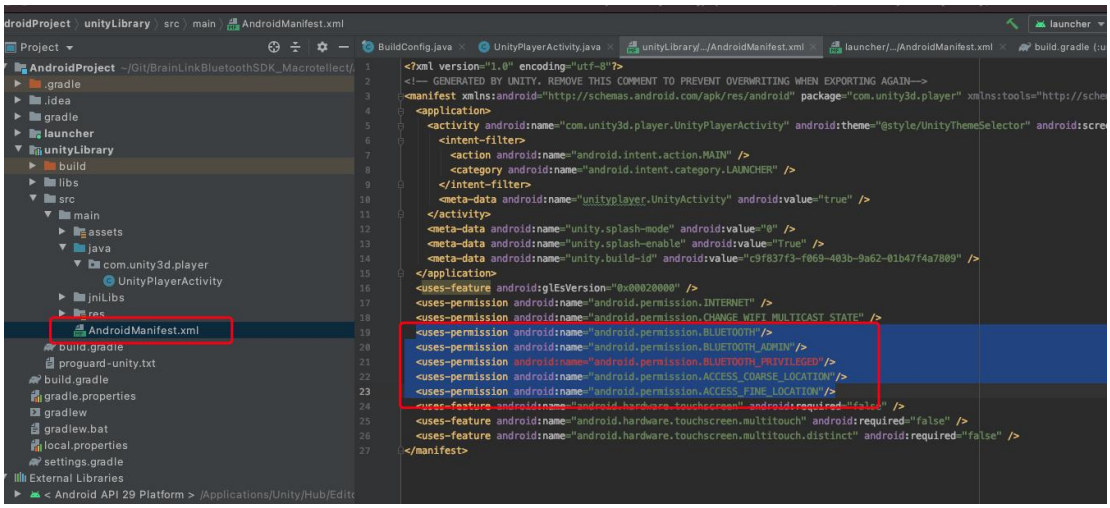
添加： implementation 'androidx.appcompat:appcompat:1.1.0'

2.导出 unity 的 android 项目 如下图



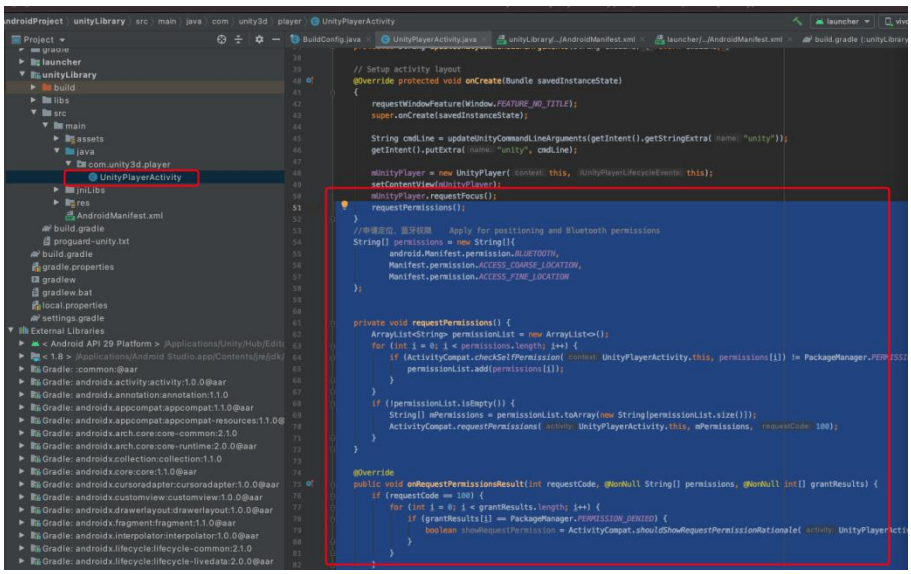
3.在导出的 android 项目中，unityLibrary 目录下的 AndroidManifest.xml 里添加 蓝牙、定位权限，如下

```
<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 V1.0.0 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

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(hzIblueDataBlock_A,hzIblueDataBlock_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hzIblueDataBlock_A即可。

各个设备的数据回调

@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_A;
@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_B;
@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_C;
@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_D;
@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_E;
@property(nonatomic,copy)Blue4DataBlock hzIblueDataBlock_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 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

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