iOS配置：

导出Xcode文件中UnityAppController.mm添加

#import "Blue4Manager.h"

extern "C"{

    void SendSettings(char \*settings){

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

        [[Blue4Manager shareInstance] writeToProAWithCode:data];

    }

    void Connect()

    {

        //蓝牙连接设置

        [Blue4Manager logEnable:YES];

        [[Blue4Manager shareInstance] configureBlueNames:@[@"BrainLink\_Pro",@"BrainLink\_Lite",@"BrainLink",@"BrainLink\_Lite\_P",@"Brainlink\_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]);

  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] connectBlue4];

    }

    void disConnect(){

        [[Blue4Manager shareInstance] disConnectBlue4];

    }

}

在Info.plist中添加

<key>NSBluetoothAlwaysUsageDescription</key>

<string>需要您的同意,才能访问蓝牙来连接脑电波设备</string>

<key>NSBluetoothPeripheralUsageDescription</key>

<string>需要您的同意,才能访问蓝牙来连接脑电波设备</string>

Bitcode设置为No

