iOS杰理蓝牙OTA开发说明

- 对应的芯片类型：AC692x，BD29
- APP开发环境：iOS平台，iOS 10.0以上，Xcode 13.4以上
- 对应于苹果商店上的APP: **【杰理OTA】**
- 源码连接： https://github.com/Jieli-Tech/iOS-JL_OTA
- 杰理OTA对外开发文档：https://doc.zh-jieli.com/Apps/iOS/ota/zh-cn/master/index.html

声明

1. 本项目所参考、使用技术必须全部来源于公知技术信息，或自主创新设计。
2. 本项目不得使用任何未经授权的第三方知识产权的技术信息。
3. 如个人使用未经授权的第三方知识产权的技术信息，造成的经济损失和法律后果由个人承担。

APP版本

版本	日期	编辑	修改内容
V3.2.0	2023年1月11日	陈冠杰	重构UI页面，整理项目架构，新增自动化测试/广播音箱模块
v2.0.0	2021年10月14日	凌 煊峰	蓝牙库新增根据ble地址对升级设备的回连；重写ota demo
v1.5.0	2021年09月08日	冯 洪鹏	优化自定义蓝牙SDK的接入方式
v1.2	2020年12月09日	冯 洪鹏	更新文档
v1.1	2020年04月20日	冯 洪鹏	增加升级的错误回调
v1.0	2019年09月09日	冯 洪鹏	OTA升级功能

概述

本文档是为了后续开发者更加便捷移植杰理OTA升级功能而创建。

1、导入JL_BLEKit.framework

将JL_OTA项目的*JL_BLEKit.framework*导入Xcode工程项目里，添加*Privacy - Bluetooth Peripheral Usage Description*和*Privacy - Bluetooth Always Usage Description*两个权限。

2、SDK具体使用的两种方式

第一种，使用自定义的蓝牙连接API进行OTA（对应BleManager文件夹）：所有BLE的操作都自行实现，SDK只负责对OTA数据包解析。

从而实现OTA功能。

第二种，使用JL_BLEKit.framework内的蓝牙连接API进行OTA（对应SDKBleManager文件夹）：完全使用SDK。

工程中已通过BleHandle文件夹内的JLBleHandler类进行了统筹区分，具体可以参考实际源码。

开发普通OTA升级流程只需要参考Views文件夹中的NormalUpdate文件内容即可。

2.1、使用自定义的蓝牙连接API进行OTA

参考Demo：「JL_OTA项目的 BleManager」

1、支持的功能：

- BLE设备握手连接；
- 获取设备信息；
- OTA升级能实现；

- 注意：相对于2.2中描述的所有BLE操作都需自行实现；

2、会用到的类：

- **JL_Assist**：部署SDK类；(必须)
- **JL_ManagerM**：命令处理中心，所有的命令操作都集中于此；(必须)
- **JLModel_Device**：设备信息存储的数据模型；(必须)

3、BLE参数：

- **【服务号】**：AE00
- **【写】特征值**：AE01
- **【读 】特征值**：AE02

2.1.1、初始化SDK

```
        /*--- JLSDK ADD ---*/
        _mAssist = [[JL_Assist alloc] init];
        _mAssist.mNeedPaired = _isPaired;              //是否需要握手配对
        /*--- 自定义配对码(16个字节配对码) ---*/
        //char pairkey[16] = {0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04};
        //NSData *pairData = [NSData dataWithBytes:pairkey length:16];
        _mAssist.mPairKey    = nil;              //配对秘钥（或者自定义配对码
pairData)
        _mAssist.mService    = FLT_BLE_SERVICE; //服务号
        _mAssist.mRcsp_W     = FLT_BLE_RCSP_W;  //特征「写」
        _mAssist.mRcsp_R     = FLT_BLE_RCSP_R;  //特征「读」
```

2.1.2、BLE设备特征回调

```
#pragma mark - 设备特征回调
- (void)peripheral:(CBPeripheral *)peripheral
didDiscoverCharacteristicsForService:(CBService *)service
             error:(nullable NSError *)error
{
    if (error) { NSLog(@"Err: Discovered Characteristics fail."); return; }

    /*--- JLSDK ADD ---*/
    [self.mAssist assistDiscoverCharacteristicsForService:service
Peripheral:peripheral];
}
```

2.1.3、BLE更新通知特征的状态

```
#pragma mark - 更新通知特征的状态
- (void)peripheral:(CBPeripheral *)peripheral
didUpdateNotificationStateForCharacteristic:(nonnull CBCharacteristic
*)characteristic
             error:(nullable NSError *)error
{
    if (error) { NSLog(@"Err: Update NotificationState For Characteristic
fail."); return; }

    /*--- JLSDK ADD ---*/
    __weak typeof(self) weakSelf = self;
    [self.mAssist assistUpdateCharacteristic:characteristic
Peripheral:peripheral Result:^(BOOL isPaired) {
        if (isPaired == YES) {
            weakSelf.lastUUID = peripheral.identifier.UUIDString;
            weakSelf.lastBleMacAddress = nil;

            weakSelf.mBlePeripheral = peripheral;
            /*--- UI配对成功 ---*/
            [JL_Tools post:kFLT_BLE_PAIRED Object:peripheral];
        } else {
            [weakSelf.bleManager cancelPeripheralConnection:peripheral];
        }
    }];
}
```

2.1.4、BLE设备返回的数据

```
#pragma mark - 设备返回的数据 GET
- (void)peripheral:(CBPeripheral *)peripheral didUpdateValueForCharacteristic:
(CBCharacteristic *)characteristic
             error:(NSError *)error
{
    if (error) { NSLog(@"Err: receive data fail."); return; }

    /*--- JLSDK ADD ---*/
    [self.mAssist assistUpdateValueForCharacteristic:characteristic];
}
```

2.1.5、BLE设备断开连接

```
#pragma mark - 设备断开连接
- (void)centralManager:(CBCentralManager *)central didDisconnectPeripheral:
(CBPeripheral *)peripheral
                 error:(nullable NSError *)error
{
    NSLog(@"BLE Disconnect ---> Device %@ error:%d", peripheral.name,
(int)error.code);
    self.mBlePeripheral = nil;

    /*--- JLSDK ADD ---*/
    [self.mAssist assistDisconnectPeripheral:peripheral];

    /*--- UI刷新，设备断开 ---*/
    [JL_Tools post:kFLT_BLE_DISCONNECTED Object:peripheral];
}
```

2.1.6、手机蓝牙状态更新

```
//外部蓝牙，手机蓝牙状态回调处，实现以下：
#pragma mark - 蓝牙初始化 Callback
- (void)centralManagerDidUpdateState:(CBCentralManager *)central
{
    _mBleManagerState = central.state;

    /*--- JLSDK ADD ---*/
    [self.mAssist assistUpdateState:central.state];

    if (_mBleManagerState != CBManagerStatePoweredOn) {
        self.mBlePeripheral = nil;
        self.blePeripheralArr = [NSMutableArray array];
    }
}
```

2.1.7、获取设备信息 （BLE连接且配对后必须执行一次）

```
  [self.mAssist.mCmdManager cmdTargetFeatureResult:^(NSArray * _Nullable array)
{
        JL_CMDStatus st = [array[0] intValue];
        if (st == JL_CMDStatusSuccess) {
            JLModel_Device *model = [weakSelf.mAssist.mCmdManager
outputDeviceModel];
            JL_OtaStatus upSt = model.otaStatus;
            if (upSt == JL_OtaStatusForce) {
                NSLog(@"---> 进入强制升级.");
                if (weakSelf.selectedOtaFilePath) {
```

```
                    [weakSelf
otaFuncWithFilePath:weakSelf.selectedOtaFilePath];
                } else {
                    callback(true);
                }
                return;
            } else {
                if (model.otaHeadset == JL_OtaHeadsetYES) {
                    NSLog(@"---> 进入强制升级：OTA另一只耳机.");
                    if (weakSelf.selectedOtaFilePath) {
                        [weakSelf
otaFuncWithFilePath:weakSelf.selectedOtaFilePath];
                    } else {
                        callback(true);
                    }
                    return;
                }
            }
            NSLog(@"---> 设备正常使用...");
            [JL_Tools mainTask:^{
                /*--- 获取公共信息 ---*/
                [weakSelf.mAssist.mCmdManager
cmdGetSystemInfo:JL_FunctionCodeCOMMON Result:nil];
            }];
        } else {
            NSLog(@"---> ERROR：设备信息获取错误!");
        }
    }];
```

2.1.8、固件OTA升级

```
   //升级流程：连接设备-->获取设备信息-->是否强制升级-->(是)则必须调用该API去OTA升级;
     //
|_______>(否)则可以正常使用APP;

// 设置代理
@interface JLUpdateViewController () <JLBleManagerOtaDelegate>
- (void)viewDidLoad {
    [super viewDidLoad];

    // 设置ota升级过程状态回调代理
    [JLBleManager sharedInstance].otaDelegate = self;
}

/**
 *  选择文件后，点击启动OTA升级
 */
- (IBAction)updateBtnFunc:(id)sender {
```

```
    if (![JLBleManager sharedInstance].mBlePeripheral) {
        self.updateSeekLabel.text = @"";
        [DFUITools showText:@"请先连接设备" onView:self.view delay:1.0];
        return;
    }

    /*--- 获取设备信息 ---*/
    [[JLBleManager sharedInstance] otaFuncWithFilePath:_selectFilePath];
}

#pragma mark - JLBleManagerOtaDelegate

/**
 *  ota升级过程状态回调
 */
- (void)otaProgressWithOtaResult:(JL_OTAResult)result withProgress:
(float)progress {
    if (result == JL_OTAResultUpgrading || result == JL_OTAResultPreparing) {
        if (result == JL_OTAResultPreparing) self.updateLabel.text = @"校验文件
中";
        if (result == JL_OTAResultUpgrading) self.updateLabel.text = @"正在升
级";
    } else if (result == JL_OTAResultPrepared) {
        NSLog(@"---> 检验文件【完成】");
    } else if (result == JL_OTAResultReconnect) {
        NSLog(@"---> OTA正在回连设备... %@", [JLBleManager
sharedInstance].mBlePeripheral.name);
        [[JLBleManager sharedInstance] connectPeripheralWithUUID:[JLBleManager
sharedInstance].lastUUID];
        [self otaTimeClose];//关闭超时检测
    } else if (result == JL_OTAResultReconnectWithMacAddr) {
        NSLog(@"---> OTA正在通过Mac Addr方式回连设备... %@", [JLBleManager
sharedInstance].mBlePeripheral.name);
        JLModel_Device *model = [[JLBleManager
sharedInstance].mAssist.mCmdManager outputDeviceModel];
        [JLBleManager sharedInstance].lastBleMacAddress = model.bleAddr;
        [[JLBleManager sharedInstance] startScanBLE];

        [self otaTimeClose];//关闭超时检测
    } else if (result == JL_OTAResultSuccess) {
        NSLog(@"--->升级成功.");
    } else if (result == JL_OTAResultReboot) {
        NSLog(@"--->设备重启.");
    } else {
        // 其余错误码详细 Command+点击JL_OTAResult 查看说明
        NSLog(@"ota update result: %d", result);
    }
}
```

2.2、使用SDK内的蓝牙连接API进行OTA

参考Demo：「JL_OTA项目的 SDKBleManager文件夹」

1、支持的功能：

- BLE设备的扫描、连接、断开、收发数据、回连功能；
- BLE设备过滤；
- BLE设备握手连接；
- BLE连接服务和特征值设置；
- 获取设备信息；
- OTA升级能实现；

2、会用到的类：

- **JL_BLEUsage**：可设置BLE过滤、握手、参数；
  查看蓝牙状态；（详情看*2.2.1*，*2.2.2*，*2.2.3*，*2.2.4*，*2.2.5*）
- **JL_Entity**：BLE设备的模型类，记录设备的相关信息（如名字、UUID、UID、PID等）；
- **JL_BLEMultiple**：BLE扫描、连接、断开、回连；
- **JL_ManagerM**：获取设备信息、OTA操作；

2.2.1、初始化SDK

```
        self.mBleMultiple = [[JL_BLEMultiple alloc] init];
        self.mBleMultiple.BLE_FILTER_ENABLE = YES;          // 过滤非杰理蓝牙设备
        // self.mBleMultiple.filterKey = nil;                 // 一般情况赋值nil
即可
        self.mBleMultiple.BLE_PAIR_ENABLE = YES;
        /*--- 自定义配对码(16个字节配对码) ---*/
        //char pairkey[16] = {0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04,
        //                    0x01,0x02,0x03,0x04};
        //NSData *pairData = [NSData dataWithBytes:pairkey length:16];
        // self.mBleMultiple.pairKey    = nil;             // 配对秘钥（或者自定义
配对码pairData）
        self.mBleMultiple.BLE_TIMEOUT = 7;
```

2.2.2、扫描设备

```
    /*--- 搜索蓝牙设备 ---*/
    [[JL_RunSDK sharedInstance].mBleMultiple scanStart];

    // 监听通知【kJL_BLE_M_FOUND】【kJL_BLE_M_FOUND_SINGLE】回调设备数组
    [JL_Tools add:kJL_BLE_M_FOUND Action:@selector(reloadTableView) Own:self];
    [JL_Tools add:kJL_BLE_M_FOUND_SINGLE Action:@selector(reloadTableView)
Own:self];
```

```
    // 获取设备数组
    - (void)reloadTableView {
        self.btEnityList = [JL_RunSDK
sharedInstance].mBleMultiple.blePeripheralArr;
        if ([JL_RunSDK sharedInstance].mBleEntityM && ![self.btEnityList
containsObject:[JL_RunSDK sharedInstance].mBleEntityM]) {
            [self.btEnityList insertObject:[JL_RunSDK
sharedInstance].mBleEntityM atIndex:0];
        }
        [self.subTableView reloadData];
    }
```

2.2.3、连接和断开设备

```
//API通过【JL_BLEMultiple】使用
/**
 连接设备
 @param entity 蓝牙设备类
 */
-(void)connectEntity:(JL_EntityM*)entity Result:(JL_EntityM_STATUS_BK)result;

/**
 断开连接
 */
-(void)disconnectEntity:(JL_EntityM*)entity Result:
(JL_EntityM_STATUS_BK)result;

/**
 *  BLE状态通知
 */
extern NSString *kJL_BLE_M_FOUND;                //发现设备
extern NSString *kJL_BLE_M_FOUND_SINGLE;         //发现单个设备
extern NSString *kJL_BLE_M_ENTITY_CONNECTED;     //连接有更新
extern NSString *kJL_BLE_M_ENTITY_DISCONNECTED; //断开连接
extern NSString *kJL_BLE_M_ON;                   //BLE开启
extern NSString *kJL_BLE_M_OFF;                  //BLE关闭
extern NSString *kJL_BLE_M_EDR_CHANGE;           //经典蓝牙输出通道变化
```

2.2.4、获取设备信息(必须)

```
[[JL_RunSDK sharedInstance] getDeviceInfo:^(BOOL needForcedUpgrade) {
    if (needForcedUpgrade) {
        NSLog(@"设备需要强制升级，请到升级界面选择ota升级文件进行升级！");
        [self startLoadingView:@"设备需要强制升级，请到升级界面选择ota升级文件进行升
级！" Delay:1.0];
    }
}];
```

2.2.5、开始OTA升级

```
   //升级流程：连接设备-->获取设备信息-->是否强制升级-->(是)则必须调用该API去OTA升级;
     //
|_______>(否)则可以正常使用APP;

// 设置代理
@interface JLUpdateViewController () <JL_RunSDKOtaDelegate>
// 设置ota升级过程状态回调代理
[JL_RunSDK sharedInstance].otaDelegate = self;

/**
 *  选择文件后，点击启动OTA升级
 */
- (IBAction)updateBtnFunc:(id)sender {
    if (![JL_RunSDK sharedInstance].mBleEntityM) {
        self.updateSeekLabel.text = @"";
        [DFUITools showText:@"请先连接设备" onView:self.view delay:1.0];
        return;
    }

    /*--- 获取设备信息 ---*/
    [[JL_RunSDK sharedInstance] otaFuncWithFilePath:_selectFilePath];
}

#pragma mark - JL_RunSDKOtaDelegate

/**
 *  ota升级过程状态回调
 */
- (void)otaProgressWithOtaResult:(JL_OTAResult)result withProgress:
(float)progress {
    if (result == JL_OTAResultUpgrading || result == JL_OTAResultPreparing) {
        if (result == JL_OTAResultPreparing) self.updateLabel.text = @"校验文件
中";
        if (result == JL_OTAResultUpgrading) self.updateLabel.text = @"正在升
级";
    } else if (result == JL_OTAResultPrepared) {
        NSLog(@"---> 检验文件【完成】");
    } else if (result == JL_OTAResultReconnect) {
        NSLog(@"---> OTA正在回连设备... %@", [JL_RunSDK
sharedInstance].mBleEntityM.mPeripheral.name);
    } else if (result == JL_OTAResultReconnectWithMacAddr) {
        NSLog(@"---> OTA正在通过Mac Addr方式回连设备... %@", [JL_RunSDK
sharedInstance].mBleEntityM.mPeripheral.name);
    } else if (result == JL_OTAResultSuccess) {
        NSLog(@"--->升级成功.");
    } else if (result == JL_OTAResultReboot) {
        NSLog(@"--->设备重启.");
```

```
    } else {
        // 其余错误码详细 Command+点击JL_OTAResult 查看说明
        NSLog(@"ota update result: %d", result);
    }
}
```