iOS 开发 ------ Bluetooth 4.0 开发

作者：HuixiaZhang
原文链接：https://ld246.com/article/1539224233710
来源网站：链滴

一.项目简介

简单介绍一下我们的项目,我们是做一个将手机作为中心去扫描外设(蓝牙铃铛),APP通过各种指令控制牙铃铛做出一些操作,框架用的是苹果提供的CoreBluetooth(基于蓝牙4.0,iOS6.0以上).其中该框架最要的是外设peripheral和中心central,在我们这手机就是central(中心),而需要连接的铃铛就是peripherl(外设),行了,直接上代码吧!

二.代码部分(自己封装的一个蓝牙管理类,重要的代码)

因为,蓝牙可能不止一个界面用到,所以我在此将它写成了一个单例.

```
// 蓝牙主要就是读写操作,需要蓝牙那边告知不同的UUID
#import "ACHLBluetooth.h"
#import <CoreBluetooth/CoreBluetooth.h>
#define ST_CAHRACTERISTIC @"xxxx"//读取数据ID
#define WT_CAHRACTERISTIC @"xxxx"//写入数据ID
#define SERVICE_UUID @"0000xxxx-0000-1000-8000-00805f9b34fb" // 服务uuid
@interface ACHLBluetooth () <CBCentralManagerDelegate,CBPeripheralDelegate>
@property (nonatomic, strong) CBCentralManager *centralMgr; // 中心
@property (nonatomic, strong) CBCharacteristic *writeCharacteristic;
@end

static ACHLBluetooth *blueManager = nil;
```

初始化,设置代理

```
blueManager.centralMgr = [[CBCentralManager alloc] initWithDelegate:self queue:nil];
```

判断是否开启蓝牙

```
#pragma mark ------ CBCentralManager代理方法 -------
// 检查蓝牙是否可用
- (void)centralManagerDidUpdateState:(CBCentralManager *)central {
    NSLog(@"检查蓝牙是否可用");
    switch (central.state) {
        case CBCentralManagerStatePoweredOn:
            // 可用，扫描所有的外设
            [self scanDevice];
            break;
        case CBCentralManagerStatePoweredOff:
            NSLog(@"蓝牙关闭了");
            break;
        default:
            break;
    }
}
// 扫描蓝牙
- (void)scanDevice {
    [blueManager.centralMgr scanForPeripheralsWithServices:nil options:nil];
}
```

扫描到的外设(centralManager的代理方法)

```
// 扫描外设连接
```

```
- (void)centralManager:(CBCentralManager *)central didDiscoverPeripheral:(CBPeripheral *)per
pheral advertisementData:(NSDictionary<NSString *,id> *)advertisementData RSSI:(NSNumbe
 *)RSSI {
    NSLog(@"扫描到%@的外设---%@",peripheral,advertisementData);
// 用1或2方法
// 1.通过通知(代理,block等)传到外面在外面连接
  //  [[NSNotificationCenter defaultCenter] postNotificationName:@"scanedDevice" object:nil
serInfo:@{@"central":central,@"advertisementData":advertisementData,@"peripheral":periphe
al}];

    //2.找到需要的蓝牙设备，停止搜素，保存数据(本页面连接)
//    if([[peripheral.identifier UUIDString] isEqualToString:IDENTIFIER]){
//        NSLog(@"扫描到名字为%@的蓝牙并开始连接",peripheral.name);
//
//        blueManager.discoveredPeripheral = peripheral;
//       连接蓝牙
//        [blueManager.centralMgr connectPeripheral:peripheral options:nil];
//
//    }else{
//
//        NSLog(@"未扫描到蓝牙");
//    }
}
```

连接断开,成功,失败

```
//连接成功
- (void)centralManager:(CBCentralManager *)central didConnectPeripheral:(CBPeripheral *)per
pheral {
    NSLog(@"蓝牙连接成功");
// 停止扫描
    [blueManager.centralMgr stopScan];
    if (self.delegate && [self.delegate respondsToSelector:@selector(connectSuccess)]) {
        [self.delegate connectSuccess];
    }
// 设置服务代理
    [blueManager.discoveredPeripheral setDelegate:self];
    [blueManager.discoveredPeripheral discoverServices:nil];

}

// 连接失败
- (void)centralManager:(CBCentralManager *)central didFailToConnectPeripheral:(CBPeriphera
 *)peripheral error:(NSError *)error {

    NSLog(@"蓝牙连接失败");
// 通过代理传出去提示用户
    if (self.delegate && [self.delegate respondsToSelector:@selector(connectError:)]) {
        [self.delegate connectError:error];
    }
}
// 断开连接了
- (void)centralManager:(CBCentralManager *)central didDisconnectPeripheral:(CBPeripheral *)
eripheral error:(NSError *)error  {
```

```
    NSLog(@"连接断开");
// 记得提示用户
    blueManager.discoveredPeripheral = nil;
}

------- CBPeripheral代理方法 ---------

//获取服务后的回调
- (void)peripheral:(CBPeripheral *)peripheral didDiscoverServices:(NSError *)error
{
    if (error)
    {
        NSLog(@"didDiscoverServices : %@", [error localizedDescription]);
        return;
    }

    for (CBService *s in peripheral.services)
    {
        [s.peripheral discoverCharacteristics:nil forService:s];
    }
}

//获取特征后的回调
- (void)peripheral:(CBPeripheral *)peripheral didDiscoverCharacteristicsForService:(CBService *
service error:(NSError *)error{
    if (error)
    {
        NSLog(@"获取服务特征出错: %@", [error localizedDescription]);
        return;
    }

    for (CBCharacteristic *c in service.characteristics)
    {

//        NSLog(@"-=-=-=--=-=-=%@",c.UUID.UUIDString);

        if ([c.UUID.UUIDString isEqualToString:ST_CAHRACTERISTIC]) {
            // 这里记得开启通知
            [peripheral setNotifyValue:YES forCharacteristic:c];

        }
        else if ([c.UUID.UUIDString isEqualToString:WT_CAHRACTERISTIC])
        {
            blueManager.writeCharacteristic = c;
        }else if ([c.UUID.UUIDString isEqualToString:GET_VERSION]) {
            // 开启通知
            [peripheral setNotifyValue:YES forCharacteristic:c];
        }
    }
}
//订阅的特征值有新的数据时回调
- (void)peripheral:(CBPeripheral *)peripheral didUpdateNotificationStateForCharacteristic:(CB
haracteristic *)characteristic
             error:(NSError *)error {
```

```
    if (error) {
        NSLog(@"Error changing notification state: %@",
              [error localizedDescription]);
    }
    [peripheral readValueForCharacteristic:characteristic];
}
// 特征值更新时回调
- (void)peripheral:(CBPeripheral *)peripheral didUpdateValueForCharacteristic:(CBCharacteristi
 *)characteristic error:(NSError *)error {
    NSLog(@"获取新的特征值了");
    if (error) {
        NSLog(@"获取新特征值时出错");
        return;
    }

    // 在这里根据不同的uuid 去做操作

    if ([characteristic.UUID.UUIDString isEqualToString:ST_CAHRACTERISTIC]) {
        NSLog(@"外设返回的数据为%@",characteristic.value);

        //转为字符串
        NSMutableString *strTemp = [NSMutableString stringWithCapacity:[characteristic.value l
ngth]*2];
        const unsigned char *szBuffer = [characteristic.value bytes];
        for (NSInteger i=0; i < [characteristic.value length]; ++i) {
            [strTemp appendFormat:@"%02lx",(unsigned long)szBuffer[i]];
           // 根据自己需求处理strTemp....
}
// 写入数据时回调
- (void)peripheral:(CBPeripheral *)peripheral didWriteValueForCharacteristic:(CBCharacteristic
)characteristic error:(NSError *)error {
    if (error) {
        NSLog(@"写入数据出错");
        return;
    } else {
        NSLog(@"恭喜你写入数据成功!");
    }
}
```

用户主动断开连接

```
// 断开连接
- (void)disconnect {
    [blueManager.centralMgr cancelPeripheralConnection:blueManager.discoveredPeripheral];
    blueManager.discoveredPeripheral = nil;
}
```

写入数据

```
// 写入数据
- (void)writeWithString:(NSString *)string {
// -----这个得看蓝牙那边,我在这写的只是我们这边的,并不适用所有.
    int length =(int)string.length / 2 ;
    SignedByte bytes[length];
```

```
    for (int i=0; i<length;i++) {
        int j=i*2;
        NSString *tmp=[string substringWithRange:NSMakeRange(j, 2)];
        unsigned int anInt;
        NSScanner * scanner = [[NSScanner alloc] initWithString:tmp];
        [scanner scanHexInt:&anInt];
        bytes[i] = anInt;
    }
//-----
    NSData *writeData = [NSData dataWithBytes:bytes length:length];
    if (blueManager.writeCharacteristic.properties & CBCharacteristicPropertyWrite) {
        [blueManager.discoveredPeripheral writeValue:writeData forCharacteristic:blueManager.
riteCharacteristic type:CBCharacteristicWriteWithResponse];
    } else {
        NSLog(@"不可写");
    }
}
```