可穿戴式设备开发

蓝牙耳蜗的设计与应用

上海交通大学 智能物联网研究中心

*这篇报告为整个蓝牙耳蜗一期结题报告

我所做的部分为一部分安卓 app 编写以及 GAIA 协议相关。
请通过目录移步至具体部分查看。

目　录

一、 绪论

1. 课题目的

通过本课题对于蓝牙耳机进行初步的开发，了解蓝牙耳机的声音传输协议、远程 音频控制协议、蓝牙的串口通信协议以及手机 APP 开发。本部分开发目前处于第一阶段，需要实现的内容包括安卓端和 IOS 端界面编程、蓝牙数据传输速率测试、蓝牙板与电脑端通信、蓝牙搜索配对和维持、晶振频率改变和校准。

2. 课题要求

- 了解蓝牙耳机音频传输、控制协议，验证蓝牙耳机音频传输功能
- 通过蓝牙串口在不影响音频功能的基础上实现指令传输及对于蓝牙远程设备的控制
- 通过电脑端连接 RS232 实现对传输指令的验证
- 通过电脑端连接 RS232 实现对 GPIO 口输出功能的控制可行性验证
- 通过配置手机端和蓝牙板之间的协议和处理，探究数据传输与音频输出同时进行的可行性
- 通过接受蓝牙端的反馈计算当前可得到的数据传输速率
- 通过改进协议内容探究改进数据传输速率的可行性
- 在 IOS 系统和接口下连接蓝牙端
- 探究在 IOS 借口下实现 GAIA 协议的可行性
- 改进安卓端的界面设计
- 在安卓端为下一步开发提供更多界面接口

批注 [Office1]: 改一下要求

3. 硬件说明

- 耳机配置配置: P4 的开关为 MIC 端，将实现耳机插孔的声音输出
- 麦克风配置：配置 P3 的开关（两个）为 LIN 端，将实现话筒插孔的声音输入:配置 P3 的开关为 MIC 端，将通过实验板的话筒（P5）实现声音输入
- 蓝牙芯片模式

  编译成功后，按住电源键不放可以更改蓝牙耳机的模式，分为如下三种模式:
  1. 未连接状态:红色灯不断闪烁，蓝牙芯片未连接
  2. 可连接/搜索状态:红色灯与绿色灯交替闪烁

批注 [Office2]: 正文内容转移到此处

3. 已连接状态:绿色灯不断闪烁，蓝牙芯片已配对连接，可实现音频、控制等功能

4. 协议说明

- HSP(Handset Profile) 蓝牙耳机与其他蓝牙设备（如手机）的通信协议。连接和配置好后，蓝牙耳机将作为远程设备的音频输入和输出接口。
- A2DP(Advanced Audio Distribution Profile) A2DP 技术能够采用耳机内的芯片来堆栈数据，从而达到声音的高清晰度，用于制作蓝牙立体声耳机
- AVRCP(Audio/Video Remote Control Profile) 用于提供控制 TV、Hi-Fi 设备等 的标准接口。AVRCP 定义了如何控制流媒体的特征，包括暂停、停止、启动重 放、音量控制及其它类型的远程控制操作（一般用于耳机等设备显示手机播放 曲目，控制手机音量、播放、暂停等）
- SPP(Serial Port Profile) 蓝牙串行端口基于 SPP 协议，能在蓝牙设备之间创建串口进行数据传输。
- GAIA GAIA 功能是基于 SPP 的蓝牙通讯协议，开启此功能后，就可以手机终端上控制蓝牙模块，并获取蓝牙模块的状态。
- GATT GATT 协议是基于 BLE（Bluetooth Low Energy）的蓝牙协议，使用 GATT 建立连接两个设备之间的 BLE 连接，比普通蓝牙连接具有低功耗特点，但同一边缘设备通过 GATT 协议仅能与一个中央设备连接。相比 GAIA 协议 GATT 协议仅提供很少的命令格式。

二、 课题分析

1. 整体性分析

系统整体框架如图 1 所示，在系统中，我们将使用移动手机及 PC 端对于整体系统 进行控制指令、音频、数据等的传输。其中，手机将通过蓝牙连接实现 UART 控制、 音频播放等功能，PC 将通过 RS232 实现控制指令与数据传输。此外，针对于具备蓝 牙的 PC 也将能够应用系统实现蓝牙连接从而完成 UART 控制、音频播放的功能。

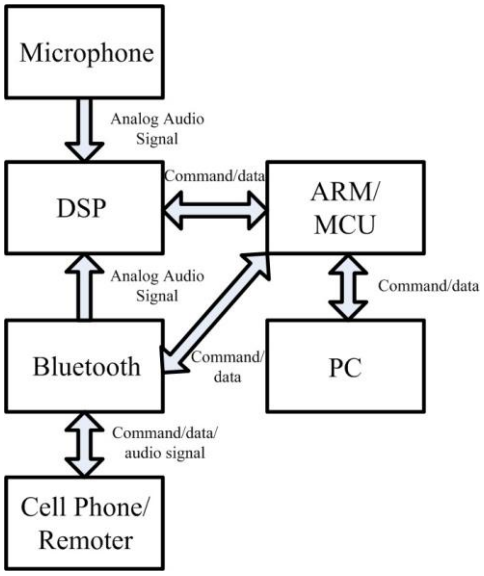


Figure 2.1: 系统整体框架图

2. 可行性分析

根据芯片的开发供应商，在 demo 的例程 sink 中，已经实现 Figure 2.1 中有关音频控制的相关内容，实现了蓝牙的连接、音频的传输等功能。目前需要实现手机对于蓝牙 设备的远程控制。考虑现实中 Apple 耳机可通过耳机上的按钮实现音乐的播放、暂停 等功能以控制手机而不影响正常功能，因而通过手机控制远程的蓝牙设备而不影响 音频功能具备可行性。

考虑协议 AVRCP 以及 GAIA，前者为远程蓝牙设备(例如:蓝牙耳机)控制手机的相关协议，后者为手机控制远程蓝牙设备的协议(基于 SPP，为已经封装好的协议)。因而，若需要实现 GAIA 协议规定以外的手机远程控制指令，可通过 SPP 按照新的协议标准进行远程控

三、 开发内容

1. 实现功能展示

A. 蓝牙搜寻配对及维持

手机与蓝牙芯片配对后将记录下蓝牙芯片的设备 ID
App 中可选择连接过的蓝牙设备 ID，通过 BlueToothAdapter 可在应用内部与设备建立连接，并建立 Socket，进行稳定的通讯。

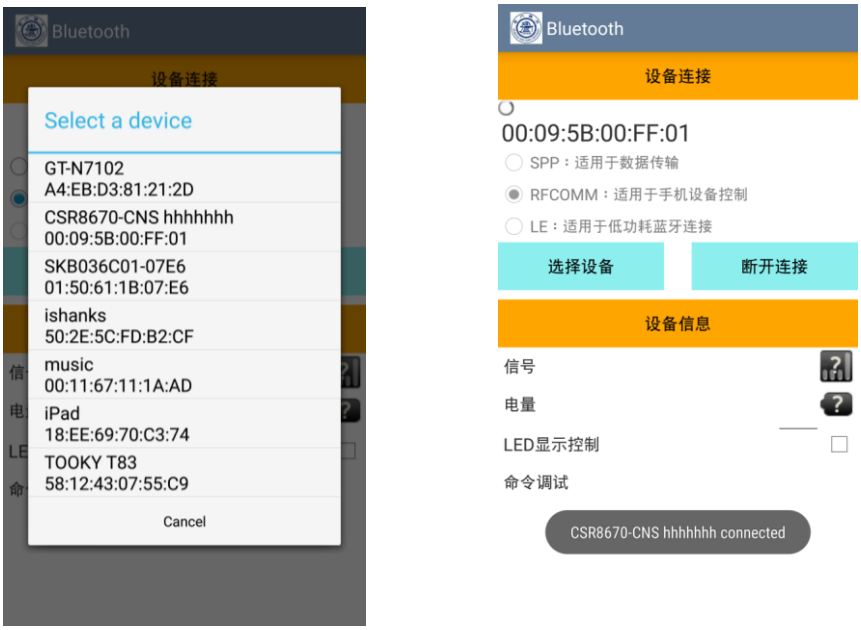


Figure 3.1: 蓝牙配对维持功能界面

B. 语音通讯功能

经验证蓝牙板可以接受手机电话信号。可以使用蓝牙班的耳机和手机的话筒共同通话，且不需要进行额外设置。

C. 命令控制功能

蓝牙板及手机端均使用 GAIA 库，手机端封装 GAIA 命令并通过建立的蓝牙 socket 发送。
蓝牙板根据 GAIA 协议解析命令并做出响应，并向手机端发送 GAIA ACK，手机端解析 ACK 了解命令是否成功执行。(根据 GAIA 协议对某些命令附加以额外数据)
以实现命令： 设备音量控制
连接信号查询
设备电量查询
命令由 GAIA 定义，命令格式如 Figure 3.2 所示

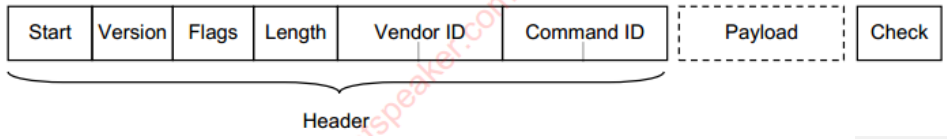


Figure 3.2: GAIA 包格式

D. 数据传输功能

停等传输下 1.92KB/s，无丢包。
非停等传输下 2.5KB/s，更接近数据传输峰值，但存在 3%左右的丢包率。
文件传输与音频播放互不影响。

E. 安卓 app

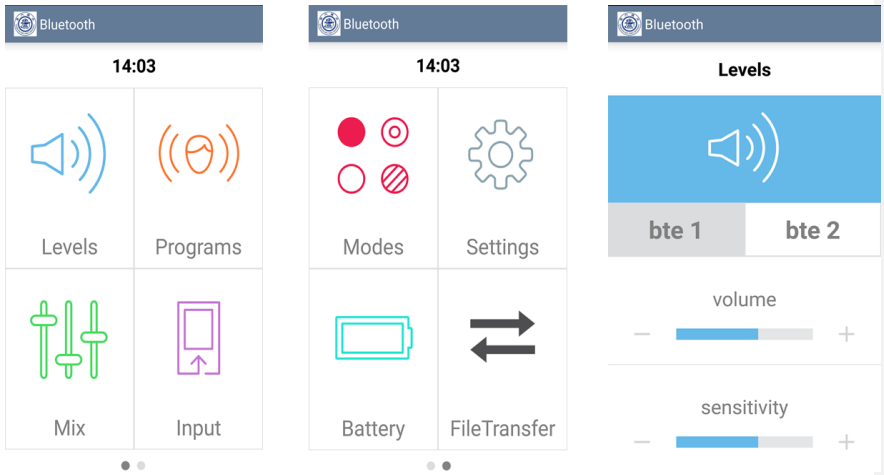


Figure 3.3: Android 应用界面

F. IOS app

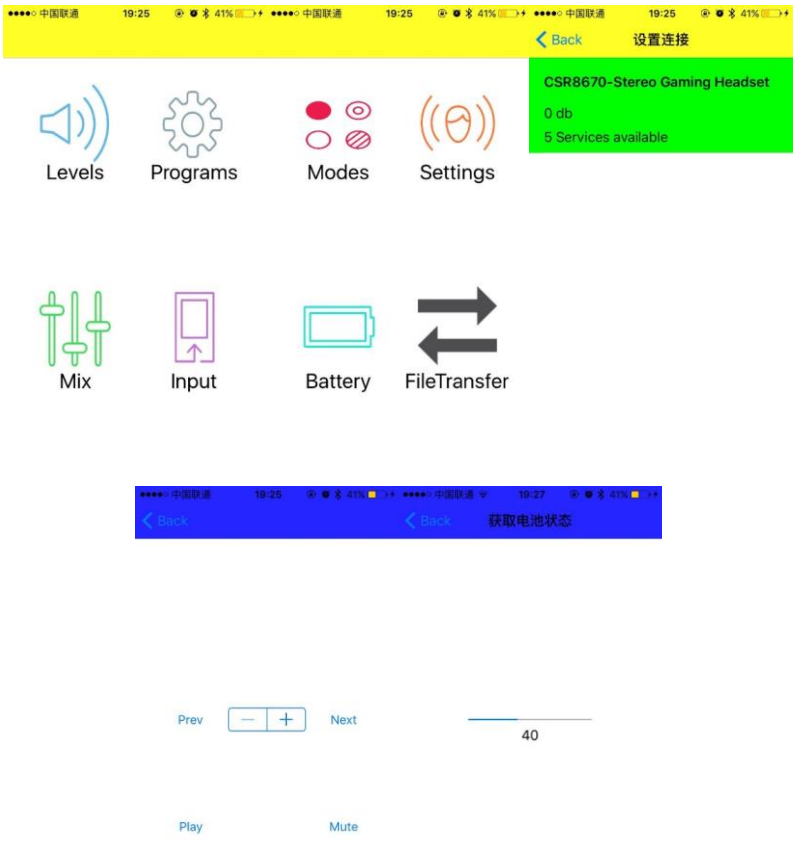


Figure 3.4：IOS 应用界面

2. 开发板配置

A. 开发板初始配置

- 系统使用需激活 sink 程序的 GAIA 功能，若使用正常的蓝牙耳机操作，则可按照
- 如下列图所示的操作进行:

- 使用 BlueLab 打开 sink 程序
- 连接实验板，通过 PSTool 进行 Merge，如 Figure 3.5 所示

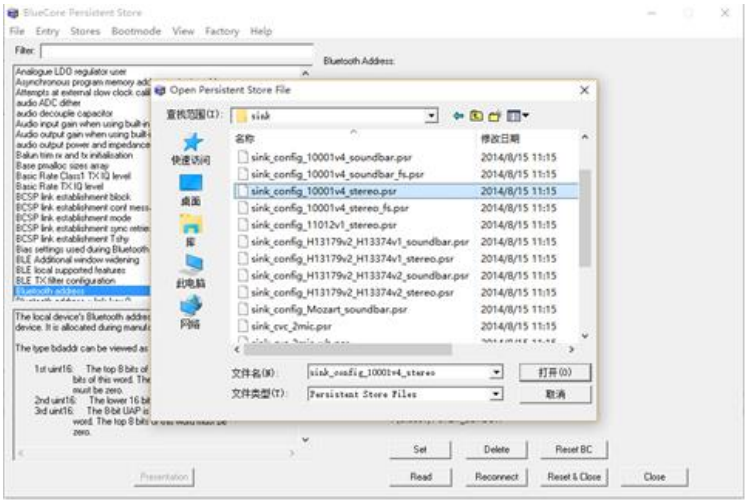


Figure 3.5: 使用 PSTool 实现 Merge

- 更改 address 为 00025b00ff01，如 Figure 3.6 所示

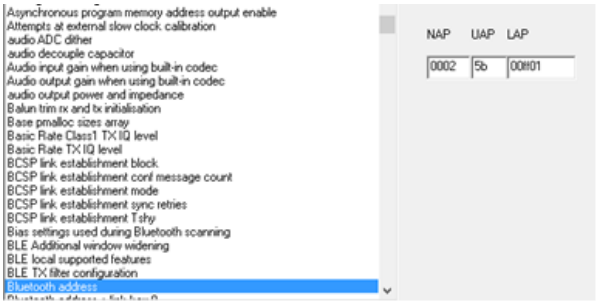
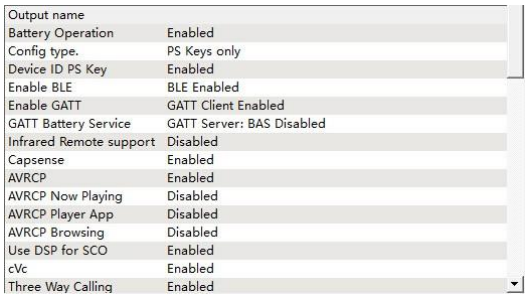


Figure 3.6: 使用 PSTool 更改 address

- 在 PSTool 中搜索 security 并更改 Module security code 的变量为 87fc 90bc 9072 228c bcb7 204a fd01 5e20 a34e d68e fd1f 80a7 51cc 62ef e27b 9db3，如 Figure3.7 所示

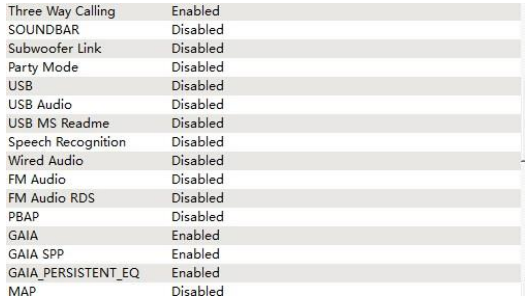
Figure 3.7: 使用 PSTool 更改 Module security code

- 采用 Headset 模式(蓝牙耳机)，针对于程序属性进行更改，更改后配置如 Figure3.8-3.10 所示



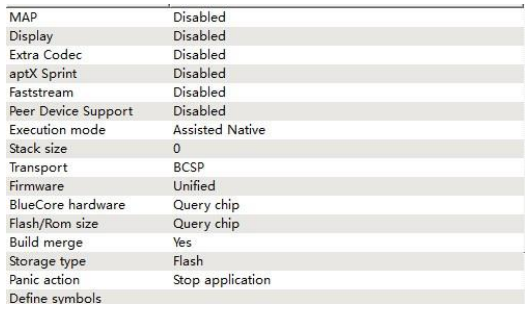
Output name	
Battery Operation	Enabled
Config type.	PS Keys only
Device ID PS Key	Enabled
Enable BLE	BLE Enabled
Enable GATT	GATT Client Enabled
GATT Battery Service	GATT Server: BAS Disabled
Infrared Remote support	Disabled
Capsense	Enabled
AVRCP	Enabled
AVRCP Now Playing	Disabled
AVRCP Player App	Disabled
AVRCP Browsing	Disabled
Use DSP for SCO	Enabled
cVc	Enabled
Three Way Calling	Enabled

Figure 3.8: 程序配置(上)



Three Way Calling	Enabled
SOUNDBAR	Disabled
Subwoofer Link	Disabled
Party Mode	Disabled
USB	Disabled
USB Audio	Disabled
USB MS Readme	Disabled
Speech Recognition	Disabled
Wired Audio	Disabled
FM Audio	Disabled
FM Audio RDS	Disabled
PBAP	Disabled
GAIA	Enabled
GAIA SPP	Enabled
GAIA_PERSISTENT_EQ	Enabled
MAP	Disabled

Figure 3.9: 程序配置(中)



MAP	Disabled
Display	Disabled
Extra Codec	Disabled
aptX Sprint	Disabled
Faststream	Disabled
Peer Device Support	Disabled
Execution mode	Assisted Native
Stack size	0
Transport	BCSP
Firmware	Unified
BlueCore hardware	Query chip
Flash/Rom size	Query chip
Build merge	Yes
Storage type	Flash
Panic action	Stop application
Define symbols	

Figure 3.10: 程序配置(下)

- 在 Libraries 处添加 gaia spp 库，否则将无法编译通过，如 Figure 3.8 所示运行程序即可，运行过程中此时可能需要按住板子上的开机键

B. CSR8670 硬件烧录步骤

- 打开 Blueflash,连接设备并点击 stop processor

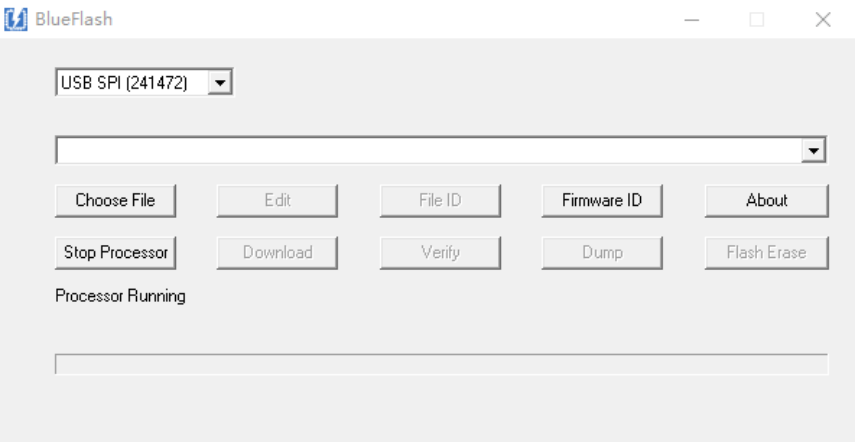


Figure 3.11:连接设备

- 点击 Flash Erase,勾选 Erase Persistent Store 后点击 Erase Full Chip 擦除全部

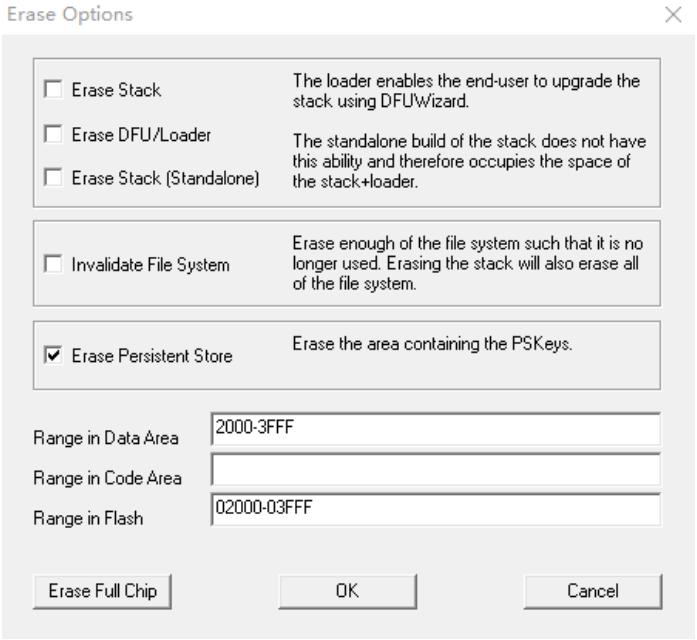


Figure 3.12:擦除数据

- 点击 choose file 依次选择下述两个文件——loader_unsigned.xpv 和 stack_unsigned.xpv，点击 download 进行烧录,上述两文件位于 ADK4.0 根目录下的 firmware\assisted\unified\gordon

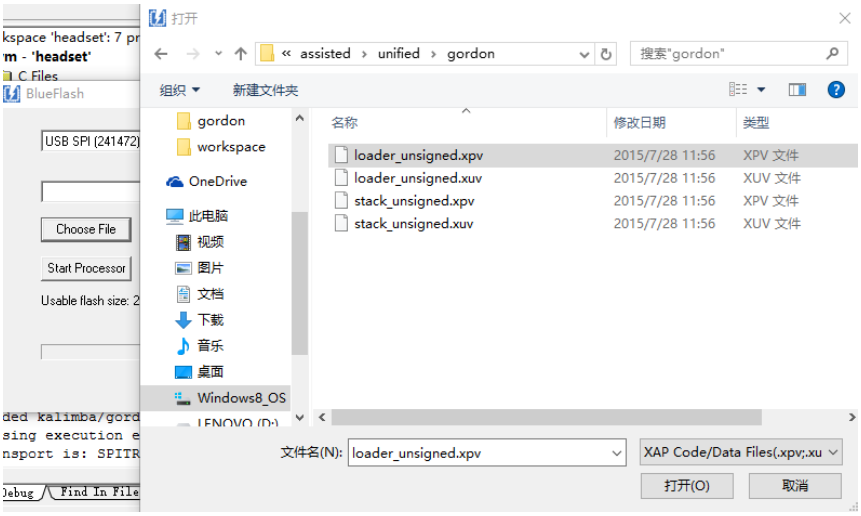


Figure 3.13:开始烧录

- 点击 start processor,关闭 Blueflash
- 打开 Pstool,连接设备，点击 file 依次 merge 下述两个文件——sink_system_csr8670.psr 以及 headset_gaming_CNS10001v4.psr,完成后点击 Reset&Close

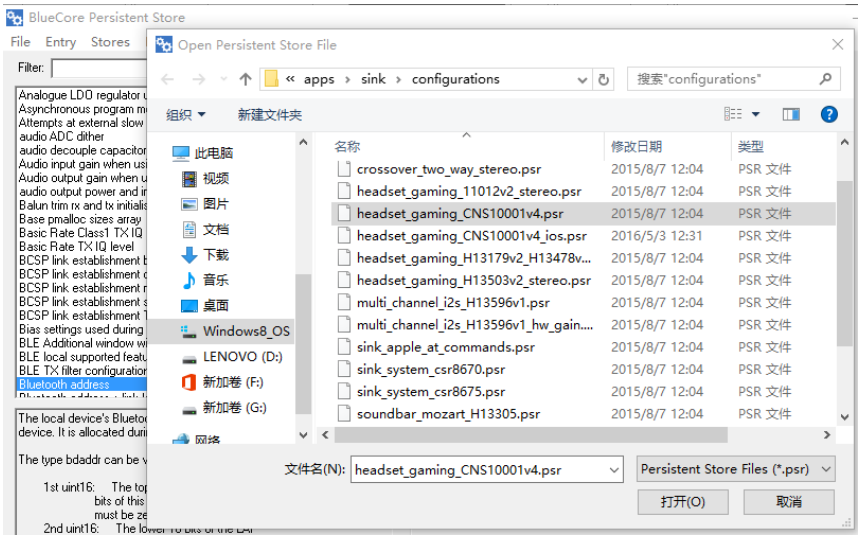


Figure 3.14:merge

- 打开 ADK4.0 xIDE,点击 project 加载 sink 项目

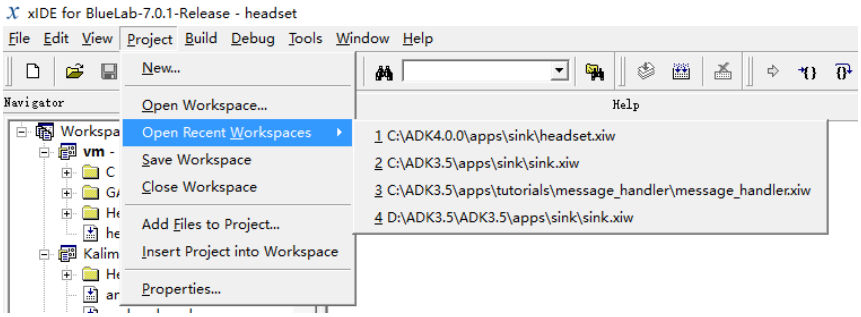


Figure 3.15:加载工程

- 点击 run 进行硬件烧录，期间需按住开发板左数第二个按钮直至完成烧录
- 烧录完成后红灯持续闪烁，此时长按第二个键直至红绿灯交替闪烁进入搜索配对模式，此时可以在手机上打开蓝牙搜索设备，完成配对后开发板将呈现绿灯持续闪烁的状态
- 使用 sink configuration tool 可以进行配置，使得开发板在上电后直接进入搜索配对模式，步骤如下：
  打开 sink configuration tool 并连接设备
  点击左侧 bluetooth,选择 connection management 下的 at power on，勾选 Enter Discoverable Mode at Power On

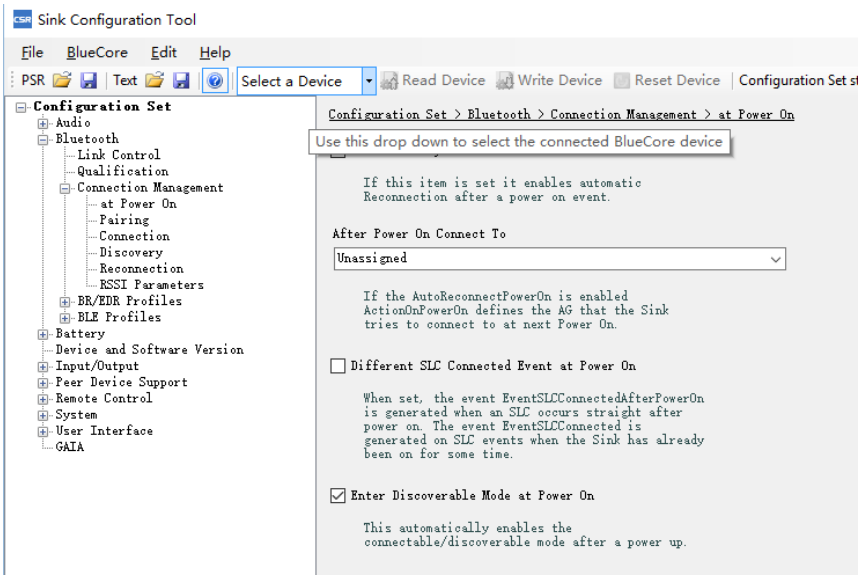


Figure 3.16:搜索配对

- 点击工具栏上的 write device，再点击 reset device，重新烧录硬件代码后即可生效

- *ios 端连接 BLE 需配置蓝牙芯片的 BLE start bonding 事件，步骤如下
- 打开 sink configuration tool 并连接设备
- 点击左侧 user interface 下的 user events,进入用户事件界面，选择 ble start bonding 事件，将其与任意逻辑输入绑定，Button timing 项对应按键时长，可配置不同按键时长以将一个按键与多个时间进行绑定

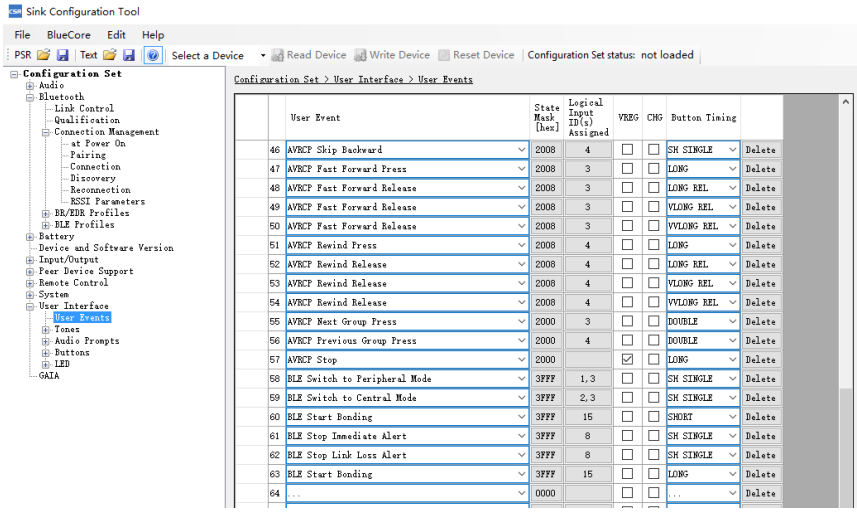


Figure 3.17:按键绑定

- 点击左侧 user interface,选择 Buttons 下的 translation，可修改按键与逻辑输入值之间的映射，开发板上右侧的 5 个按键从右往左依次对应 PIO0~PIO4，左侧两个键从左到右为预设的 CHG 和 VREG，对应 PIO25 和 PIO24，建议将用户时间绑定至右侧 5 个键

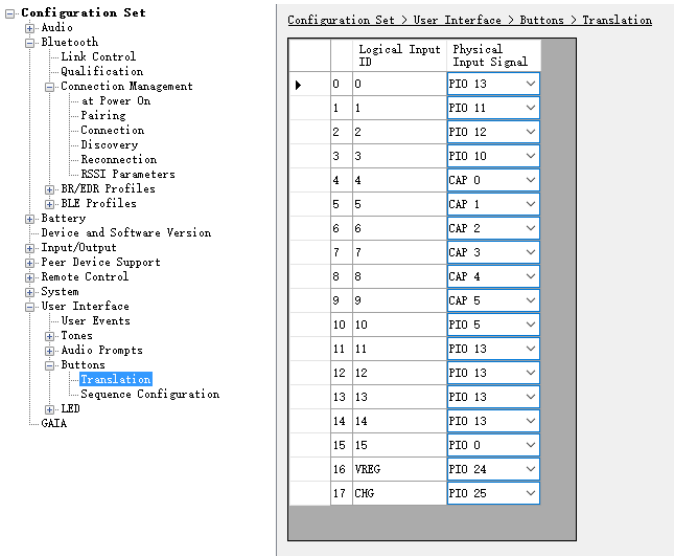


Figure 3.18:修改映射

- 完成事件绑定以及按键映射的配置后,先后点击 write device 和 reset device，重新烧录后即可生效
- 在蓝牙芯片进入搜索配对模式后可以点击实现设定的按键触发 ble start bonding 事件，此时观察 xIDE 中 bluestack 一栏，若出现 DM pdu (type 0x0357) is not recognised 说明 bonding 尚未完成，继续点击直至出现 DM_HCI_WRITE_SCAN_ENABLE_CFM,表明 bonding 已成功，此时手机向开发板发起配对即可成功连接

```
---- 17:40:41.386 ------------------
DM pdu (type 0x0357) is not recognised.
Content =
    0x57, 0x03, 0x10, 0x00, 0x20, 0x00, 0x30, 0x00, W... .0.
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ........
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, ........
    0x00, 0x00                                      ..
-----------------------------------
```

```
---- 17:42:12.513 ------------------
DM_HCI_WRITE_SCAN_ENABLE_CFM
    type = 0x0e1a (3610)
    phandle = 0x0010 (16)
    status = 0x00 (0)
-------------------------------------
```

Figure 3.19:连接成功

3. 晶振频率更改调试方法

批注 [oc3]: //To do

4. 电脑端 UART 连接测试

A. 电脑端调试

我们针对于代码进行更改，通过电脑端连接调试时可以获取手机发送的指令信息，我们可以针对于获取的指令进行分析，进一步将指令代码与具体操作联系起来，如图 12 所示，为电脑端接收到的指令，为了更便于调试，我们可以针对于具体的指令进行分析。

针对于音量调整我们获取手机的信息为 cmd: 000a:0201 1

针对于警报灯的开启关闭我们获取的手机信息为 cmd: 000a:020d 13

针对于手机的主动播放示例声一我们获取的手机信息为 cmd: 000a:0209 1

```
cmd: 000a:c003 2
cmd: 000a:0201 1
cmd: 000a:c003 2
cmd: 000a:0201 1
cmd: 000a:c003 2
cmd: 000a:0201 1
cmd: 000a:c003 2
cmd: 000a:0201 1
cmd: 000a:c003 2
cmd: 000a:0201 1
cmd: 000a:c003 2
```

Figure 3.20: 电脑端调试结果

针对于其他内容此处不再详细列举，若为了方便调试及查找历史，我们将有能 力编写相应的日志生成代码。考虑到合作中手机对于蓝牙芯片的操作可能与示例不同，我们将重新设计新的 cmd 指令并对应到具体操作。

B. RS232 连接控制

- UART 控制 GPIO 主要通过电脑与板间采用 RS232 连接，RS232 的输入经过图 14 后， 连接到 CSR8670 的 UART 发送、接收口。经过 CSR8670 中 20-23 的 UART 端口，通过主 板逻辑，控制图 15 中的 PIO 口。由于根据板子的布设架构，如图 16 显示，可以通过观 察 LED 灯显示来进行验证。故而，整体方案的效果要求为:通过连接电脑的 RS232， 采用电脑可以对板子上的 LED 灯进行 UART 控制。

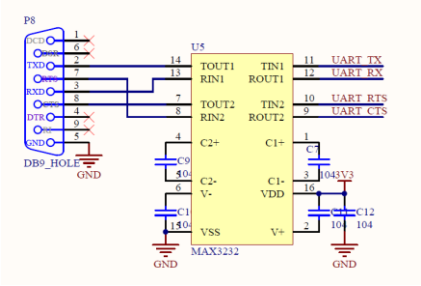


Figure 3.21: UART 输入

- 在我们的实验中，我们对 UART 数据传输逻辑以及 GPIO 口控制逻辑进行了整合,从而获得了 UART 控制 GPIO 口的方案。此外，根据 UART 传输信息的内容，也可设定相应的控制协议，完成对 GPIO 口的不同设计方案，例如:周期性闪灯、输出特定的波形等等。

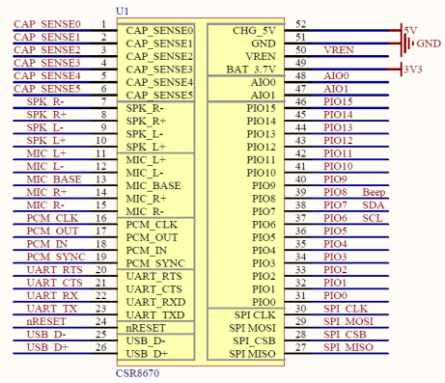


Figure 3.22: CSR8670

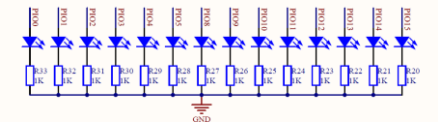


Figure 3.23: PIO 验证方案

- 为了更好的理解本部分的实际测试效果，我们拍摄了简短的小视频，由于无法嵌入 PDF 中，故单独发送视频效果以便验收。

5. 文件传输测试

A. 测试方法

通过停等与非停等两种方式对于文件传输速率进行了一个测试。测试中，我们均是使用 GAIA 协议中的 get_storage_partition_status (0x0610)指令（测试时任选一条指令即可），对其进行改造，在 GAIA 协议单条指令最长 270 字节的限制下，每

条携带 250 字节的数据段，由手机端连续不断地向蓝牙耳机发送。蓝牙耳机端在接收到指令后，根据情况返回状态信息。
若耳机端在一条指令处理途中收到了下条指令，则新指令会被解调并直接返回错误信息 insufficient_resource, 提示该条指令收到但无法得到处理，而解调过程也会对上一条指令的处理速度造成影响，拖慢整个文件传输过程的速度。这也是我们用两种测试方式的主要原因。

- 停等方法
  停等方法在手机端接收到耳机端返回的 ACK 后才进行下一条指令的发送。这样的方法可以确保文件的分包有序且完整的被耳机端所接收，但是实际上耳机端返回 ACK 的这段时间被浪费，对传输速率有一定影响。
  经过测试，停等方法得到的速率为 1.9KB/s 左右。
- 非停等方法
  非停等方法对耳机端从接收到指令到 ACK 发出过程的平均时长进行一个估计，手机端在一个分包发出并等待这样一个时长后直接发送下一个分包，从而达到传输速率的一个提高。
  易见，文件传输速率与等候间隙的关系是一个凸函数，我们通过逼近可以近似得到一个文件传输速率的峰值.经过测试，这个文件传输速率的峰值为 2.5KB/s 左右。
  在实际应用中，这种方法显然是有失妥帖的。耳机端的对于指令的接收和处理的耗时并非完全稳定，我们在测速过程中，在文件速率达到最大值时，是无法保证所有包均被接收到并处理的，实际为 97%左右。所以如果要使用这种方法进行文件传输，可能需要牺牲一定的指令有效载荷，加入 packet sequence 信息，并加入重发机制，保证数据完整性。

B. 实现方法

- 耳机端

在 sink_gaia.c 文件的 gaia_handle_command()函数中，加入对 GAIA_COMMAND_TYPE_DATA_TRANSFER 类型指令的处理，关于指令类型及指令 ID 的常量定义均在 ADK03.5/tools/include/profiles/BlueLab-6.5.2-Release/gaia.h 文件中。

对于指令的处理如下图所示，之所以对指令 payload 长度进行分类是因为在传输末尾我们发送一条相同 ID 但是 payload 较短的指令用于标识传输结束，此时耳机端输出收到的总的包数，与手机端收到的 ACK 进行对照。而对于过程中的包我们仅作计数及返回 ACK 操作，以减少比较耗时的 IO 操作。

```
static bool gaia_handle_datatransfer_command(Task task, GAIA_UNHANDLED_COMMAND_IND_T *command)
{
    switch (command->command_id)
    {
    case GAIA_COMMAND_GET_STORAGE_PARTITION_STATUS:
        /*printf("command received\n");*/
        if (command->size_payload >= 100)
        {
            cnt += 1;
            gaia_send_success(command->command_id);
        } else if (command->size_payload > 10) {
            printf("%d\n",cnt);
        }
        else
            gaia_send_invalid_parameter(GAIA_COMMAND_GET_STORAGE_PARTITION_STATUS);
        return TRUE;
    case GAIA_COMMAND_OPEN_STORAGE_PARTITION:
```

Figure 3.24: 文件传输 sink 代码

- 手机端

停等方法

手机端主要采用 handler-message 的机制进行分包的发送，整个过程的每一次包发送在 handler 的 handleMessage()方法中进行。发送的开始由 button 触发，发送按钮按下后向 handler 发送一个 message，第一个包由此发出。在收到耳机端的 ACK 后，会立即向该 handler 发送一个 message，继续发送第二个包，由此开始循环往复。
我们测试了多组 5000 个包的连续发送，记录下发送开始与结束的时间戳，计算得停等方法的一个平均速率为 1.9KB/s。

非停等方法

非停等方法沿用上述的 handler，另开一个子线程定时向该 handler 发送 message 达到定时事件的效果。通过调整发送间隔逼近接收速率的最大值。最终我们测得在发送间隔在 95ms 左右(±2ms)时，文件传输速率可以达到 2.5(±0.1)KB/s 的峰值。

C. 接口

- 耳机端接收到的数据，正如在第二部分所展示的，目前可在 sink_gaia.c 文件中的 handle_datatransfer_command (Task task, GAIA_UNHANDLED_COMMAND_IND_T *command) 函数中的 case GAIA_COMMAND_GET_STORAGE_PARTITION_STATUS 下进行处理。command 的 payload 即是指令所携带的全部数据。
- 由于我们仅是测试速度所以简单使用了一条现有指令。实际使用中，若不想占用这些既有指令，可在 ADK3.5/tools/include/profiles/BlueLab-6.5.2-Release/gaia.h 中定义一条新的用户指令用于文件传输。为了规范，data transfer 类型的指令 ID 为 0x06xx。然后将新的指令的 case 加入到上

述函数中即可。

6. Android 操作流程

- **GAIA 板卡配置**

  - ADK3.5 开发：通过 GAIA 协议规定的控制协议内容，能够实现示例声音的播放以及声音的调 整、实验板指示灯的控制等功能。若需要有额外的功能补充，可自行定义通信协议并通过 SPP 串口信息传递实现远程控制功能。此项功能需要安卓手机安装 GAIA 控制 的相关软件。

  - ADK4.0 开发：ADK4.0 中自动集成了 GAIA_SPP 库，不需要再进行如 ADK3.5 所示的操作

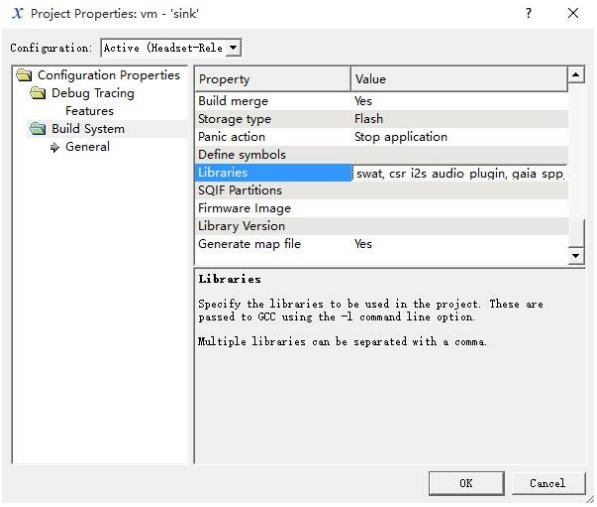


Figure 3.25: 添加 GAIA 库

批注 [oc4]: //To do

- **Android App 使用说明**

  - 手机连接开发板
    进入 设置-蓝牙 找到我们的设备 CSR8670-XXXX，点击进行配对（匹配失败请尝试多次），此时开发板已经可以作为蓝牙耳机使用进行音频播放

Figure 3.26:配对

- 打开 APP，进入 setting 界面，连接方式选择 SPP，点击选择设备选择已配对的 CSR8670

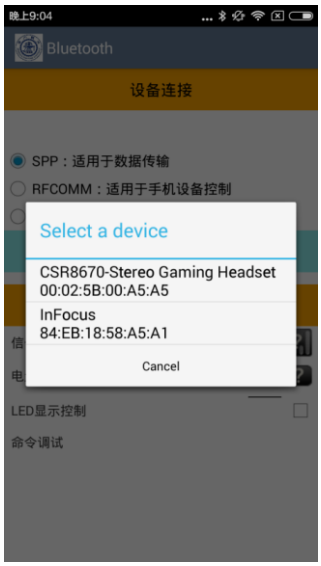


Figure 3.27:SPP 配对

- 选择设备后设备地址将显示在界面上方，点击连接设备，若屏幕下方出现 headset connected 的 toast 则说明连接成功，若失败则断开连接后重试，屏幕下方将显示蓝牙芯片的电量以及信号强度，LED 显示控制项可以控制开发板上的绿灯的亮暗

Figure 3.28:连接成功

- 音量界面（Levels）可以调节蓝牙芯片的输出音量，媒体音量在 0x00 到 0x0f 之间

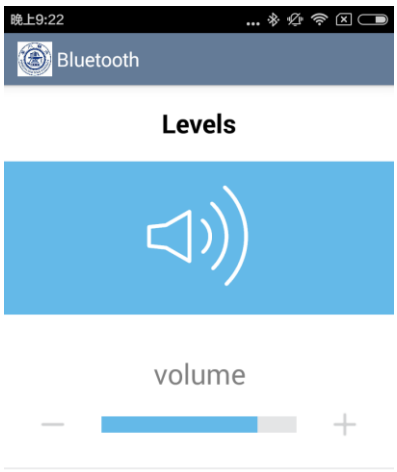


Figure 3.29:音量调节

- 文件传输界面可操作文件传输以及 UART 数据传输

- 文件传输时第一行输入为要发送的包的数量，每个包携带的数据为 250 字节，第二行为两个包之间的发送间隔，这个数值经过我们的测试在 95ms 左右时传输速率最高；点击 upload 开始传输，点击 stop 终止传输；下方为结果，第一行为开始发送时的时间戳，第二行为结束传送时的时间戳，第三行为收到的 ACK 数目，第四行为计算得的

Figure 3.30:数据传输

- 输入任意字符，点击 uart 可将内容发送至开发板，开发板接收到后将通过 uart 传送至电脑端，通过串口调试软件可以显示对应接口接收到的数据；同时可以通过 uart 对开发板进行一些控制，这里我们可以用 task 0b00 id 8021 来控制开发板上的一盏 LED 灯

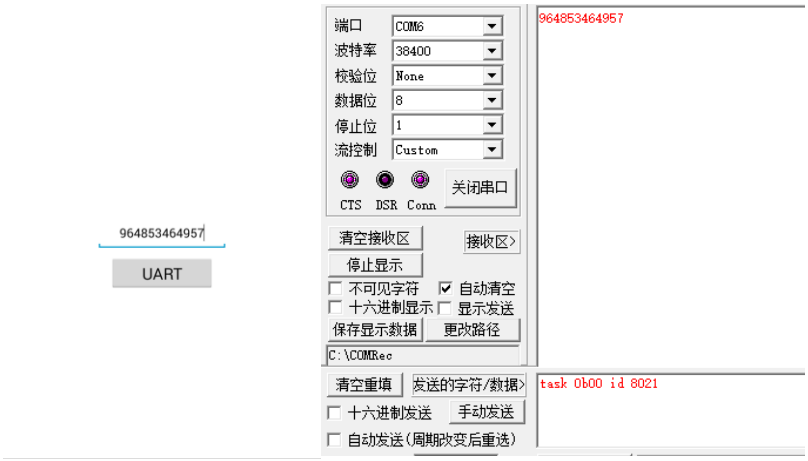


Figure 3.31:URAT

7. IOS 操作流程

A. 测试

我们针对于 IOS 平台进行了相应的测试，测试内容如下:

- IOS 平台能够通过蓝牙(传统蓝牙)连接蓝牙耳机并具备正常的音频功能
- IOS 平台能够通过 GATT 进行正常的蓝牙串口通信。我们具备将安卓端 GAIA 协议移植到 IOS 端的开发能力。在之前的开发中，我们已经成功向 IOS 端移植过安卓平台 (JAVA)的算法并测试成功。
- 针对于 IOS 接口，受到 IOS 接口封装的限制，目前只能通过 GATT 方式搭载 GAIA 协议

针对于上述技术，我们得出结论:能够在 IOS 平台上实现蓝牙耳蜗技术的开发与应用。同时，根据物联网中心目前的开发能力，我们在实现安卓端相关功能后具备 IOS 开发的能力，同时 IOS 平台具备连接此蓝牙耳蜗并实现相应控制功能的基础。在实际应用中，为了支持 BLE，针对于手机硬件需要相应的条件，即 iPhone4S 以下型号的手机可能无法正常运转耳蜗的相应功能。

批注 [oc5]: //To do

B. 操作流程

- 当开发板进入红绿灯交替闪烁的状态之后，打开手机蓝牙，搜索并连接开发板。

Figure 3.32:IOS 蓝牙搜索

- 此时打开 APP，点击 Settings 设置连接。

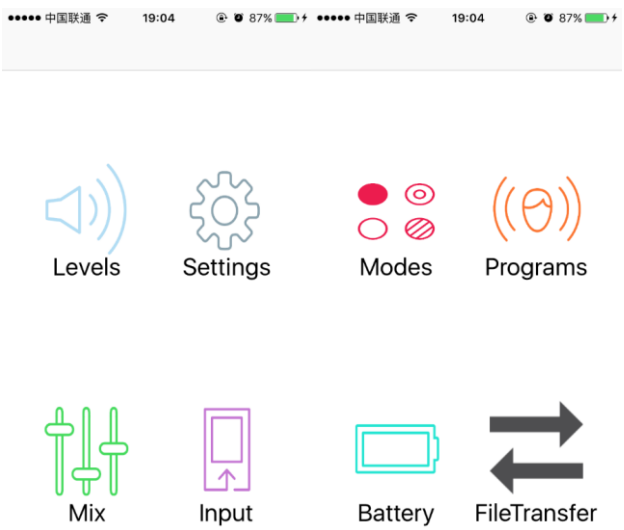


Figure 3.33:IOS 连接设置

- 点击开发板设置好的 bonding 键，直到 BlueStack 窗口出现 Write_Scan_Enable。此时点击设置，可以发现 CSR8670 设备，点击设备的名称进行配对，第一次进行配对会弹出配对提示，之后就不会再出现。

Figure 3.34:IOS 配对

- 确认连接成功后，返回主界面，在上一级菜单中选择 Levels 调节音量，选择 Battery 查看电量信息。

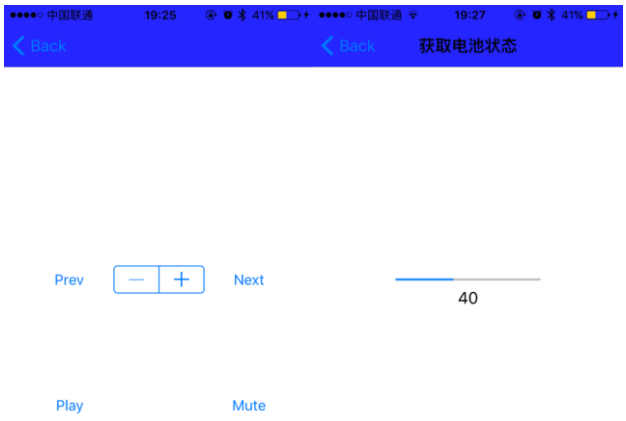


Figure 3.35:IOS 音量电量调节

四、 结论

- 可通过芯片实现音频播放
- 可通过手机对于蓝牙耳机进行远程的控制，包括获取电池状态，调整音量等功能，蓝牙耳机端可以通过返回信息反馈特定命令
- 针对于芯片能够将目前所处的状态以及手机发送的控制指令传输到电脑端进行
- 相应的核实验证，已验证蓝牙耳机可以与手机共同工作，测得 GAIA 协议状态下手机端传输数据最大速度为 2.5k/s，停等条件下则为 1.9k/s，距离目标下限仍有差距
- 探索使用蓝牙 4.0 并自定义蓝牙协议是否提升数据传输速率
- 测试得数据传输与接受音频信号可并行，数据传输并不影响音频信号质量（GAIA 协议下）
- Android、IOS 平台手机 APP 已实现基本功能，并留下相应接口以备下一阶段开发使用
- 探究得目前 IOS 端受其本身接口限制无法搭载 GAIA 协议
- 探究 IOS 端使用 GATT 协议实现 GAIA 部分命令
- UART 及蓝牙均可控制 CSR8670 的 GPIO 口的输入输出
- 成功以 BLE 连接手机端及蓝牙耳机端
- 通过焊接晶振完成晶振频率更改，在蓝牙板上更改了对应参数