青橙无线眼镜SDK使用说明

1. Author：James
2. Shenzhen QC.wireless Technology Co., Ltd.
3. Version：1.0.0

更新记录:

1. （2025/07/23）扫描,连接,测量指令
2. （2025/07/23）增加设置指令

1.介绍

1.1SDK的作用

向合作伙伴公司提供可与青橙无线设备一起使用的Android眼镜SDK，该设备为主要眼镜或其他设备提供基本功能和高级功能。 该文档旨在解释API的使用上下文，功能等。 预期读者和阅读建议附件1中显示了本文中预期的读者和读者建议。

阅读者	作用
软件架构工程师	架构分析和技术指导
android开发工程师	具有一定的android开发能力，了解Ble，wifi相关开发技术

2.API 说明

2.0接入条件

> Android 5.0以上版本，Bluetooth 4.0以上.

2.1 SDK需要的权限

```
    //网络权限
    <uses-permission android:name="android.permission.INTERNET" />
    //蓝牙相关权限
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
/>
    //存储相关权限
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

2.2接入条件

- 青橙无线眼镜
- 青橙无线SDK及文档

2.3.1API

扫描设备

```
    //开始
    BleScannerHelper.getInstance().scanDevice(final Context context,
UUID mUuid, final ScanWrapperCallback scanCallBack);
    //结束
    BleScannerHelper.getInstance().stopScan(Context context)
    //指定设备扫描
    BleScannerHelper.getInstance().scanTheDevice(final Context context,
final String macAddress, final OnTheScanResult scanResult)
```

设备连接:BleOperateManager.getInstance()

```
    //直连

 BleOperateManager.getInstance().connectDirectly(smartWatch.deviceAddres
s)
    //扫描连接

 BleOperateManager.getInstance().connectWithScan(smartWatch.deviceAddres
s)
    //断开
    BleOperateManager.getInstance().unBindDevice()
    //重连
    BleOperateManager.getInstance().setNeedConnect(boolean needConnect)
    //关闭蓝牙时调用
    BleOperateManager.getInstance().setBluetoothTurnOff(false)
    BleOperateManager.getInstance().disconnect()
```

```
    //打开系统蓝牙监听
    BleOperateManager.getInstance().setBluetoothTurnOff(true)

    请注意 sample 的MyApplication的注册监听，要先注册才能收到蓝牙的连接，断开事件
```

设备连接:BleOperateManager.getInstance()

```
  //注册眼镜事件上报监听
  LargeDataHandler.getInstance().addOutDeviceListener(100,
deviceNotifyListener)

   inner class MyDeviceNotifyListener : GlassesDeviceNotifyListener() {

        @RequiresApi(Build.VERSION_CODES.O)
        override fun parseData(cmdType: Int, response:
GlassesDeviceNotifyRsp) {
            when (response.loadData[6].toInt()) {
                //眼镜电量上报
                0x05 -> {
                    //当前电量
                    val battery = response.loadData[7].toInt()
                    //是否在充电
                    val changing = response.loadData[8].toInt()
                }
                //眼镜通过快捷识别
                0x02 -> {
                    if (response.loadData.size > 9 &&
response.loadData[9].toInt() == 0x02) {
                        //要设置识别意图: eg 请帮我看看眼前是什么，图片中的内容
                    }
                    //获取图片缩略图
                    LargeDataHandler.getInstance().getPictureThumbnails
{ cmdType, success, data ->
                        //请将data存入路径,jpg的图片
                    }
                }
```

```
0x03 -> {
    if (response.loadData[7].toInt() == 1) {
        //眼镜启动麦克风开始说话
    }
}
//ota 升级
0x04 -> {
    try {
        val download = response.loadData[7].toInt()
        val soc = response.loadData[8].toInt()
        val nor = response.loadData[9].toInt()
        //download 固件下载进度 soc 下载进度 nor 升级进度
    } catch (e: Exception) {
        e.printStackTrace()
    }
}

0x0c -> {
    //眼镜触发暂停事件，语音播报
    if (response.loadData[7].toInt() == 1) {
        //to do
    }
}

0x0d -> {
    //解除APP绑定事件
    if (response.loadData[7].toInt() == 1) {
        //to do
    }
}
//眼镜内存不足事件
0x0e -> {

}
//翻译暂停事件
```

```
                0x10 -> {

                }
                //眼镜音量变化事件
                0x12 -> {
                    //音乐音量
                    //最小音量
                    response.loadData[8].toInt()
                    //最大音量
                    response.loadData[9].toInt()
                    //当前音量
                    response.loadData[10].toInt()

                    //来电音量
                    //最小音量
                    response.loadData[12].toInt()
                    //最大音量
                    response.loadData[13].toInt()
                    //当前音量
                    response.loadData[14].toInt()

                    //眼镜系统音量
                    //最小音量
                    response.loadData[16].toInt()
                    //最大音量
                    response.loadData[17].toInt()
                    //当前音量
                    response.loadData[18].toInt()

                    //当前的音量模式
                    response.loadData[19].toInt()

                }
            }
        }
    }
```

2.3.2功能列表:

同步时间

```
        LargeDataHandler.getInstance().syncTime { _, _ -> }
```

眼镜电量

```
    //添加电量监听
     LargeDataHandler.getInstance().addBatteryCallBack("init") { _,
response ->

     }
     //电量
     LargeDataHandler.getInstance().syncBattery()
}
```

读取眼镜版本信息

```
      LargeDataHandler.getInstance().syncDeviceInfo { _, response ->
                        if (response != null) {
                            //wifi 固件版本
                             response.wifiFirmwareVersion
                            //wifi 产品版本
                            response.wifiHardwareVersion
                            //蓝牙产品版本
                             response.hardwareVersion
                            //蓝牙固件版本
                             response.firmwareVersion
                        }
                    }
}
```

控制眼镜拍照

```
        LargeDataHandler.getInstance().glassesControl(
                        byteArrayOf(0x02, 0x01, 0x01)
                    ) { _, it ->
                        if (it.dataType == 1 && it.errorCode == 0) {
                            when (it.workTypeIng) {
                                2 -> {
                                    //眼镜正在录像
                                }
                                4 -> {
                                    //眼镜正在传输模式
                                }
                                5 -> {
                                    //眼镜正在OTA模式
                                }
                                1, 6 ->{
                                    //眼镜正在拍照模式
                                }
                                7 -> {
                                    //眼镜正在AI对话
                                }
                                8 ->{
                                    //眼镜正在录音模式
                                }
                            }
                        } else {
                            //执行开始和结束
                        }
                    }
```

控制眼镜录像

```
        //videoStart  true 开始录制   false 停止录制
                    val videoStart=true
```

```
val value = if (videoStart) 0x02 else 0x03
LargeDataHandler.getInstance().glassesControl(
    byteArrayOf(0x02, 0x01, value.toByte())
) { _, it ->
    if (it.dataType == 1) {
        if (it.errorCode == 0) {
            when (it.workTypeIng) {
                2 -> {
                    //眼镜正在录像
                }
                4 -> {
                    //眼镜正在传输模式
                }
                5 -> {
                    //眼镜正在OTA模式
                }
                1, 6 ->{
                    //眼镜正在拍照模式
                }
                7 -> {
                    //眼镜正在AI对话
                }
                8 ->{
                    //眼镜正在录音模式
                }
            }
        } else {
            //执行开始和结束
        }
    }
}
```

控制眼镜录音

```
//recordStart  true 开始录制   false 停止录制
val recordStart=true
val value = if (recordStart) 0x08 else 0x0c
LargeDataHandler.getInstance().glassesControl(
                byteArrayOf(0x02, 0x01, value.toByte())
            ) { _, it ->
                if (it.dataType == 1) {
                    if (it.errorCode == 0) {
                        when (it.workTypeIng) {
                            2 -> {
                                //眼镜正在录像
                            }
                            4 -> {
                                //眼镜正在传输模式
                            }
                            5 -> {
                                //眼镜正在OTA模式
                            }
                            1, 6 ->{
                                //眼镜正在拍照模式
                            }
                            7 -> {
                                //眼镜正在AI对话
                            }
                            8 ->{
                                //眼镜正在录音模式
                            }
                        }
                    } else {
                        //执行开始和结束
                    }
                }
            }
```

控制智能识图，上报缩略图

```
        //thumbnailSize  0..6
        val thumbnailSize=0x02
        LargeDataHandler.getInstance().glassesControl(
                        byteArrayOf(
                            0x02,
                            0x01,
                            0x06,
                            thumbnailSize.toByte(),
                            thumbnailSize.toByte(),
                            0x02
                        )
                    ) { _, it ->
                        if (it.dataType == 1) {
                            if (it.errorCode == 0) {
                                when (it.workTypeIng) {
                                    2 -> {
                                        //眼镜正在录像
                                    }
                                    4 -> {
                                        //眼镜正在传输模式
                                    }
                                    5 -> {
                                        //眼镜正在OTA模式
                                    }
                                    1, 6 ->{
                                        //眼镜正在拍照模式
                                    }
                                    7 -> {
                                        //眼镜正在AI对话
                                    }
                                    8 ->{
                                        //眼镜正在录音模式
                                    }
                                }
```

```
                            } else {
                                //触发AI拍照，上报缩略图会收到上报指令
                            }
                        }
                    }
```

BT配对

```
    //BT扫描
    BleOperateManager.getInstance().classicBluetoothStartScan()
    //BluetoothReceiver
        BluetoothDevice.ACTION_FOUND -> {
                val device =
                    intent.getParcelableExtra<BluetoothDevice>
(BluetoothDevice.EXTRA_DEVICE)
                if (device != null) {
                  //发现设备，当蓝牙地址和当前 BLE地址相等调用配对

 BleOperateManager.getInstance().createBondBluetoothJieLi(device)
                }
            }
```

眼镜音量控制

```
    //读取音量控制
      LargeDataHandler.getInstance().getVolumeControl { _, response ->
                        if (response != null) {
                            //眼镜音量 音乐最小值 最大值 当前值
                                    response.minVolumeMusic
                                    response.maxVolumeMusic
                                    response.currVolumeMusic
                            //眼镜电话 电话最小值 最大值 当前值
                                    response.minVolumeCall
```

```
                                response.maxVolumeCall
                                response.currVolumeCall
                            //眼镜系统 系统最小值 最大值 当前值
                                response.minVolumeSystem
                                response.maxVolumeSystem
                                response.currVolumeSystem
                            //眼镜当前的模式
                                response.currVolumeType
                        }
                    }
```

眼镜未同步的媒体数量

```
    LargeDataHandler.getInstance().glassesControl(byteArrayOf(0x02, 0x04))
{ _, it ->
    //it.imageCount    图片数量
    //it.videoCount    视频数量
    //it.recordCount   录音数量
            if (it.dataType == 4) {
            val mediaCount = it.imageCount + it.videoCount + it.recordCount
            if (mediaCount > 0) {
            //眼镜有多少个媒体没有上传
            } else {
            //无
            }
        }
}
```