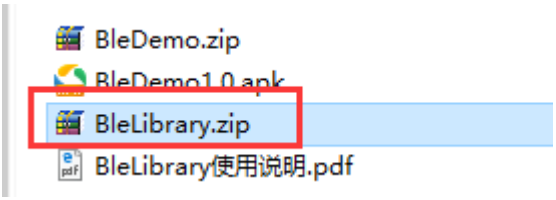
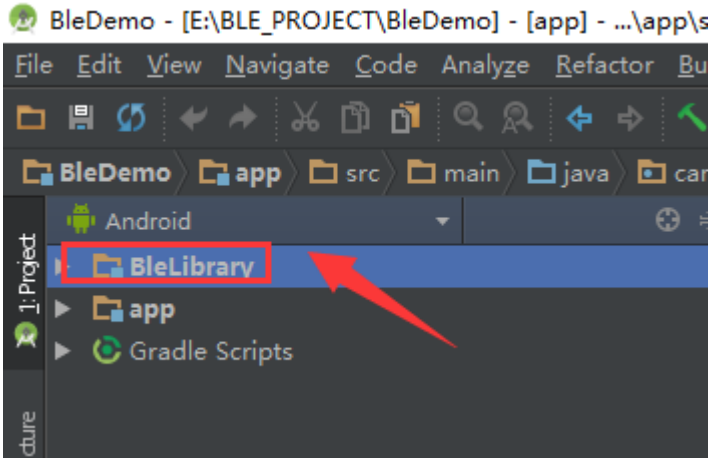
BleLibrary 使用说明

使用平台: Android studio
Android SDK API： 大于等于 level 18

一、在 android studio 中导入 BleLibrary 库；



解压 BleLibrary.zip 压缩包将其导入 android studio 项目中作为库使用。导入后如下所示：



二、在你的项目中的配置文件 AndroidManifest.xml 中添加如下：

```
<!—权限 -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-feature
      android:name="android.hardware.bluetooth_le"
        android:required="true" />

<!—在 application 标签之间注册服务 -->
<service    android:name="com.inuker.bluetooth.library.BluetoothService" />
```

（说明：由于 android 6.0 以上设备使用蓝牙 4.0 需要使用定位权限，所以需要把定位权限添加上）

三、蓝牙开关

判断手机是否支持低功耗蓝牙：

```
if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
                Toast.makeText(this, "手机不支持低功耗蓝牙", Toast.LENGTH_SHORT).show();
                return;
        }

final BluetoothManager mBluetoothManager = (BluetoothManager)
getSystemService(Context.BLUETOOTH_SERVICE);
        if (mBluetoothManager.getAdapter() == null) {//取得蓝牙适配器
            Toast.makeText(this, "你的手机不支持蓝牙 4.0", Toast.LENGTH_SHORT).show();
            return;
        }
```

打开蓝牙：`ClientManager.getClient(this).openBluetooth();`

关闭蓝牙：`ClientManager.getClient(this).closeBluetooth();`

监听蓝牙的开关：

```
    ClientManager.getClient(this).registerBluetoothStateListener(mBluetoothStateListener);

    private final BluetoothStateListener mBluetoothStateListener = new BluetoothStateListener() {
        @Override
        public void onBluetoothStateChanged(boolean openOrClosed) {
            // true 打开，false 关闭
        }
    };
```

四、设备扫描

```
private void scanBle() {
    SearchRequest request = new SearchRequest.Builder()
                .searchBluetoothLeDevice(3000, 1)// 先扫 BLE 设备 1 次，每次 3s
                .build();
    ClientManager.getClient(this).search(request, new SearchResponse() {

        @Override
        public void onSearchStarted() {
          //开始扫描
```

```
        }

        @Override
        public void onDeviceFounded(SearchResult result) {
            //扫描得到附近蓝牙设备信息如（地址 mac，设备名字 name，距离 rssi 等）
        }

        @Override
        public void onSearchStopped() {
            //扫描结束
        }

        @Override
        public void onSearchCanceled() {
            //取消
        }
    });
}
```

五、停止扫描

```
ClientManager.getClient(this).stopSearch();
```

六、搜索到蓝牙之后连接蓝牙设备服务

```
BleRequest    mBleRequest    =    new BleRequest();

mBleRequest.connect(mContext, bleMac, new BleConnectResponse() {
            @Override
            public void onResponse(int code, BleGattProfile data) {
                if (code == REQUEST_SUCCESS) {
                    //成功连接
                }else {
                     //连接失败
                }
            }
        });
```

七、监听服务连接状态

注册：ClientManager.getClient(this).registerConnectStatusListener(bleMac,mConnectStatusListener);

注销：ClientManager.getClient(this).unregisterConnectStatusListener(bleMac, mConnectStatusListener);

```
/**
 * 连接状态监听
 */
private final BleConnectStatusListener mConnectStatusListener = new BleConnectStatusListener() {
    @Override
    public void onConnectStatusChanged(String mac, int status) {
        isConnected = (status == STATUS_CONNECTED);
          // status=0 表示连接成功，其他值表示断开
    }
};
```

八、连接服务成功之后打开蓝牙通知并且同步数据

打开通知 ：

打开默认服务通知（接收设备发送上来的数据）

```
mBleRequest.openDeafultNotify(mContext, bleMac, new BleDefaultNotifyListener() {
                    @Override
                    public void onOpen(boolean isOpen) {
                        if (!isOpen){
                        //没有打开就从新打开
                        }
                    }
                });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleDefaultNotifyListener	interface	是否成功打开通知回调	

打开另一个服务通知（这个服务通知专门接收实时的用户状态和坐姿状态）

```
        leRequest.openOtherNotify(mContext, bleMac, new BleOtherNotifyListener() {
                    @Override
                    public void onOpen(boolean isOpen) {
                        if (!isOpen){
                            //没有打开就从新打开
                        }
                    }
                });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleOtherNotifyListener	interface	回调	

监听设备发送上来的数据：

```
/* 每次连接都会数据进行同步*/
BLE.setOnBleSynListener(new BleSynDataListener() {
    @Override
    public void onFirstResponse(String result) {
        // result 数据格式是 json,包含了电池电量，马达震动标志位以及固件版本号
        (说明：这个数据是连接之后设备第一份发送来的数据信息)
      //{“code":"0"," battery":"2," monitorflag ":" F5"," firmwareversion ":" WZP01.0.1",}
      //后面有说明
    }

    @Override
    public void startSynTime(boolean isStart) {
          //开始同步
    }

    @Override
    public void sendFinishSyn() {
        mBleRequest.finishSyn(mContext, bleMac, new BleRequest.SynFinishListener() {
            @Override
            public void finishSyn(boolean isFinish) {
                //数据同步完成
            }
        });
    }

    @Override
    public void onHistoryStepSyn(String result) {
      // Log.d(TAG, "历史步数... " + result);
//{"code":"0","year":"2017","month":"4","day":"24","hour":"22","minute":"33","second":"46","step":0}
 //如果有多天历史步数数据将会多次回调 后面有说明 （demo 中的查看历史步数数据就是将这里返回的历史步数存起来传递到另一个页面查看）
    }

    @Override
    public void onHistoryUserStatusSyn(String result) {
        // Log.d(TAG, "历史用户状态... " + result);

 //{"code":0,"month":4,"day":24,"rows":[ {"hour":2,"minute":2,"status":"走"},{"hour":3,"minute":3,"status":"跑"}]}
 //如果有多天历史用户状态数据将会多次回调 后面有说明（demo 中的查看历史用户状态数据就是将这里返回的历史步数存起来传递到另一个页面查看）
    }
```

```
});

    @Override
    public void onHistorySittingStatusSyn(String result) {
        // Log.d(TAG, "历史坐姿返回信息... " + result);
//{"code":"0","year":"2257","month":"5","day":"5","sitting":100,"forward":200,"backward":300,"leftLeaning":400,"rightLeaning":500}
    }
```

连接第一次返回数据说明：

{“code":"0"," battery":"2," monitorflag ":" F5"," firmwareversion ":" WZP01.0.1",}

字段说明：

字段	类型	说明	备注
code	String	Code=”0”表示返回成功，无其他值	
battery	String	2 表示百分之 2%	
monitorflag	String	F5 表示打开，F0 表示关闭	
firmwareversion	String	固件版本号	

返回历史步数数据说明：

{"code":"0","year":"2017","month":"4","day":"24","hour":"22","minute":"33","second":"46","step":0}

字段说明：

字段	类型	说明	备注
code	String	Code=”0”表示返回成功，无其他值	回调没执行表示数据没返回；
year	String	年	
month	String	月	
day	String	日	
hour	String	时	
minute	String	分	
second	String	秒	
step	Int	步数	

返回历史用户状态数据说明：

{"code":0, "year":"2017","month":4,"day":24,"rows":[{"hour":2,"minute":2,"status":"走"},{"hour":3,"minute":3,"status":"跑"}]}

字段说明：

字段	类型	说明	备注

code	Int	0:返回成功 无其他值	
year	int	年	
month	int	月	
day	Int	日	
rows[]	Aray	数组坐姿点数	
hour	Int	时	
minute	Int	分	
status	String	状态	未知、坐/站立、躺、走、跑

返回历史坐姿数据说明：

{"code":"0","year":"2257","month":"5","day":"5","sitting":100,"forward":200,"backward":300,"leftLeaning":400,"rightLeaning":500}

字段说明：

字段	类型	说明	备注
code	Int	0:返回成功 无其他值	
year	String	年	
month	String	月	
day	String	日	
sitting	int	坐正时间，单位：秒	
forward	Int	前倾时间，单位：秒	
backward	Int	后倾时间，单位：秒	
leftLeaning	Int	左倾时间，单位：秒	
rightLeaning	int	右倾时间，单位：秒	

实时监听用户状态和坐姿状态：

```
BLE.setOnUserStatusAndSittingStatusListener(new BleUserStatusAndSittingStatusListener() {
        @Override
        public void onStatus(String userStatus, String sittingStatus) {
            // String status = "用户状态:" + userStatus +    "  ——  坐姿:" + sittingStatus;
        }
    });
```

接口回调 onStatus(String userStatus, String sittingStatus) 说明：

(userStatus:未知 、坐/站立、躺 、走、跑; sittingStatus:、未知 、正坐 、偏左 、偏右 、偏前 、偏后

这个用户状态和坐姿状态在 demo 的标题栏显示！)

(注:在页面活动生命周期结束之时需要关闭服务通知 mBleRequest.closeDeafultNotify(this,bleMac);)

九、发送指令

1、 读取电池电量

```
mBleRequest.getBattery(mContext, bleMac, new BleBateryListener() {
                    @Override
                    public void onBattery(int level) {
                        // "获取电池电量"   level （如 50 表示 50%）
                    }
                });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleBateryListener	interface	回调	

2、 读取马达震动标志

```
mBleRequest.getMotorShockFlag(mContext, bleMac, new BleMotorFlagListener() {
    @Override
    public void onMotor(String motorFlag,int second) {
            // "读取马达震动 " + motorFlag + " " + second + "s");
            (说明 motorFlag 为 F0 表示关闭 ，为 F5 表示打开，second 表示触发震动时间)
    }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleMotorFlagListener	interface	回调	

3、 关闭马达震动

```
mBleRequest.setMotorShock(mContext, bleMac, false, 0, new BleSetMotorShockListener() {
                    @Override
                    public void onSetMotorShock(boolean isSet) {
                        //设置成功 true: 设置成功 ， false :设置失败
                    }
                });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleSetMotorShockListener	interface	设置马达震动回调	
false	Boolean	表示关闭	
time	Int	0 秒此参数无用	关闭马达震动，不用

			传时间

4、 打开马达震动设置参数

```
mBleRequest.setMotorShock(mContext, bleMac, true, time, new BleSetMotorShockListener() {
    @Override
    public void onSetMotorShock(boolean isSet) {
        //设置成功
    }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleSetMotorShockListener	interface	设置马达震动回调	
true	Boolean	表示打开	
time	Int	最大只能设置 120 秒	打开马达震动要设置时间

5、 激活设备命令

```
mBleRequest.setEnableDevice(mContext, bleMac, new BleEnableDeviceListener() {
        @Override
        public void onEnable(boolean isSet) {
            //激活成功
        }
});
```

（说明：激活是将 sensor 打开在没到用户使用之前，是没有历史数据的，不然，如果传感器一直开着，就一直会有数据产生，出厂的时候是未激活的，到用户手上激活了才可以使用，这时候使用设备才会产生数据）

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleEnableDeviceListener	interface	设置指令回调	

6、 取消激活设备命令

```
mBleRequest.setDisableDevice(mContext, bleMac, new BleDisableDeviceListener() {
        @Override
        public void onDisable(boolean isSet) {
            //取消激活成功
        }
});
```

（说明：取消激活是 sensor 关闭了，降低功耗）

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleDisableDeviceListener	interface	设置指令回调	

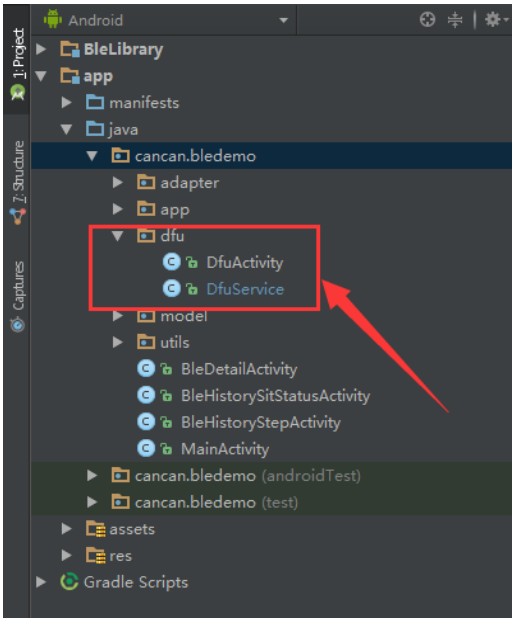
7、 发送空中升级模式命令

```
mBleRequest.setDfuModel(mContext, bleMac, new BleDfuModelListener() {
        @Override
        public void onDfuModel(boolean isDfu) {
            if (isDfu){
                //"发送升级指令成功
            }else {
                //"发送升级指令不成功，请重新发送。"
            }
        }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleDfuModelListener	interface	设置指令回调	

（说明：发送完空中升级命令之后，设备将会重启进入 DFU 空中升级模式，这时候需要重新扫描设备，此时将会扫描到名字为 DfuTrag 名字的设备，连接这个设备之后进行空中升级，空中升级详情参考 demo）；



8、 同步当前用户状态数据

```
mBleRequest. getCurrentUserStatus(mContext, bleMac, new BleCurrentUserStatusListener () {
        @Override
        public void onCurrentStatus(String result) {
            //返回当前坐姿数据，格式为 json
        }
    });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleCurrentUserStatusListener	interface	读取当前用户状态数据回调	

返回坐姿数据说明：

{"code":0,"month":4,"day":24,"rows":[{"hour":2,"minute":2,"status":"走"},{"hour":3,"minute":3,"status":"跑"}]}

字段说明：

字段	类型	说明	备注
code	Int	0:返回成功 无其他值	
month	int	月	
day	Int	日	
rows[]	Aray	数组坐姿点数	
hour	Int	时	
minute	Int	分	
status	String	状态	未知、坐/站立、躺、走、跑

9、 同步实时的步数

```
mBleRequest.getCurrentStep(mContext, bleMac, new BleCurrentStepListener() {
    @Override
    public void onStep(String result) {
        //  返回步数信息 --------> result 是 json 格式数据
//{"code":"0","year":"2017","month":"4","day":"24","hour":"22","minute":"33","second":"46","step":0}
    }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleCurrentStepListener	interface	读取当前步数回调	

返回步数数据说明：

{"code":"0","year":"2017","month":"4","day":"24","hour":"22","minute":"33","second":"46","step":0}

字段说明：

字段	类型	说明	备注
code	String	Code=”0”表示返回成功，无其他值	回调没执行表示数据没返回；
year	String	年	
month	String	月	
day	String	日	
hour	String	时	
minute	String	分	
second	String	秒	
step	Int	步数	

10、 坐姿校准

```
mBleRequest.setCalibrateSitPosition(mContext, bleMac, new BleCalibrateSitPositionListener() {
        @Override
        public void onCalibrate(boolean isCalibrate) {
            //校准返回，true 校准成功，false 发送指令失败
        }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleCalibrateSitPositionListener	interface	设置指令回调	

11、 坐姿校准清除

```
mBleRequest.setClearCalibrateSitPostion(mContext, bleMac, new BleClearCalibrateSitPositionListener() {
    @Override
    public void onClearCalibrate(boolean isSuccess) {
        if (isSuccess){
            //Toast.makeText(mContext,"坐姿校准清除",Toast.LENGTH_SHORT).show();
        }else {
            //Toast.makeText(mContext,"发送指令失败",Toast.LENGTH_SHORT).show();
        }
    }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleClearCalibrateSitPositionListener	interface	设置指令回调	

12、 重启 BLE 设备

```
mBleRequest.setBleReboot(mContext, bleMac, new BleRebootListener() {
        @Override
        public void onReboot(boolean isSuccess) {
            if (isSuccess){
                // finish();//发送重启命令成功之后，设备马达会震动一下，这时候，需要重新扫描 BLE 设备连接
            }else {
                //"发送重启命令不成功
            }
        }
    });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleRebootListener	interface	设置指令回调	

13、 清除设备数据

```
mBleRequest.setClearBleData(mContext, bleMac, new BleClearDataListener() {
    @Override
    public void onClearData(boolean isSuccess) {
        if (isSuccess){
            //发送清除数据命令成功
        }else {
            //发送清除数据命令失败
        }
    }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	

BleClearDataListener	interface	设置指令回调	

以下设置前后左右倾角度说明请看最后面的附录；

14、设置前倾角度

```
mBleRequest.setForwardAngle(mContext, bleMac, angle, new BleForwardAngleListener() {
        @Override
        public void onForwardAngle(boolean isSuccess) {
            if (isSuccess){
                //设置前倾角度成功
            }else {
                //发送指令失败
            }
        }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleForwardAngleListener	interface	设置指令回调	

15、 设置后倾角度

```
mBleRequest.setBackwardAngle(mContext, bleMac, angle, new BleBackwardAngleListener() {
        @Override
        public void onBackwardAngle(boolean isSuccess{
            if (isSuccess){
                //设置后倾角度成功
            }else {
                //发送指令失败
            }
        }
});
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleBackwardAngleListener	interface	设置指令回调	

16、 设置左倾角度

```
mBleRequest.setLeftAngle(mContext, bleMac, angle, new BleLeftAngleListener() {
        @Override
        public void onLeftAngle(boolean isSuccess) {
```

```
            if (isSuccess){
                //设置左倾角度成功
            }else {
                //发送指令失败
              }
        }
    });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleLeftAngleListener	interface	设置指令回调	

17、 设置右倾角度

```
mBleRequest.setRightAngle(mContext, bleMac, angle, new BleRightAngleListener() {
        @Override
        public void onRightAngle(boolean isSuccess) {
            if (isSuccess){
                //设置右倾角度成功
            }else {
                //发送指令失败
            }
        }
    });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleRightAngleListener	interface	设置指令回调	

18、 设置 BLE 昵称

```
    mBleRequest.setBleNickname(mContext, bleMac, name, new BleSetBleNickname() {
        @Override
        public void onSetNickname(boolean isSuccess) {
            if (isSuccess){
                //设置成功
            }else {
                //设置失败"
            }
        }
    });
```

参数说明：

参数	类型	说明	备注
mContext	Context	上下文	
bleMac	String	蓝牙地址	
BleSetBleNickname	interface	设置指令回调	

关于设置昵称说明：设置昵称的长度限制（中文最多五个，字幕最多十七个，可以中文字母混合，设置完成之后需要重启设备才能显示设置的昵称，不重启不影响指令的传输）

附录：

1. 关于设置倾斜角度说明：表示当用户倾斜多少度时，表示坐歪了。例如设置前倾角度为 30 时，当用户向前倾 30 度时表示坐歪。姿态会标记为前倾。