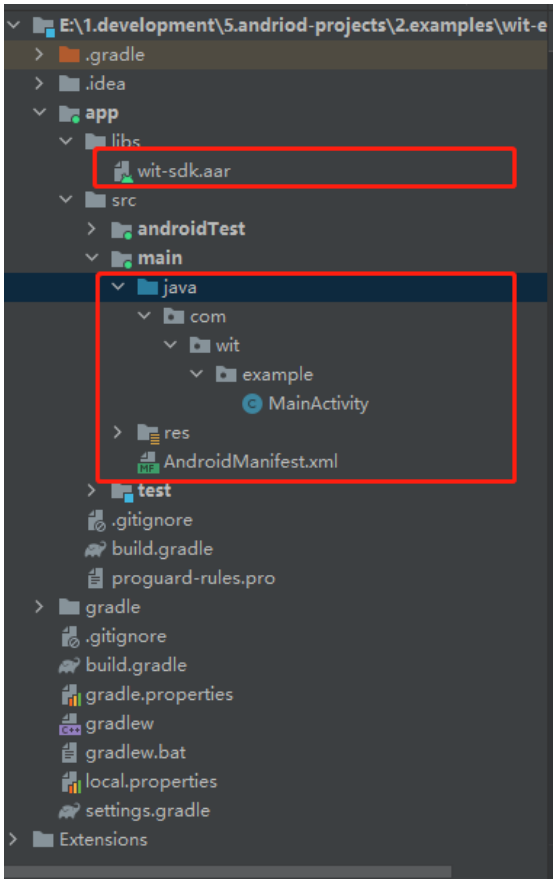
BWT901CL 2.0 Quick Guide

Sample code Instruction

1. This example demonstrates how to use the Android Bluetooth 2.0 SDK developed by WitMotion
2. This example will demonstrate how to search and connect Bluetooth 2.0 sensors, and control the sensors
3. Please be familiar with the use of WitMotion Bluetooth 2.0 before using the SDK, and understand the protocol of the sensor

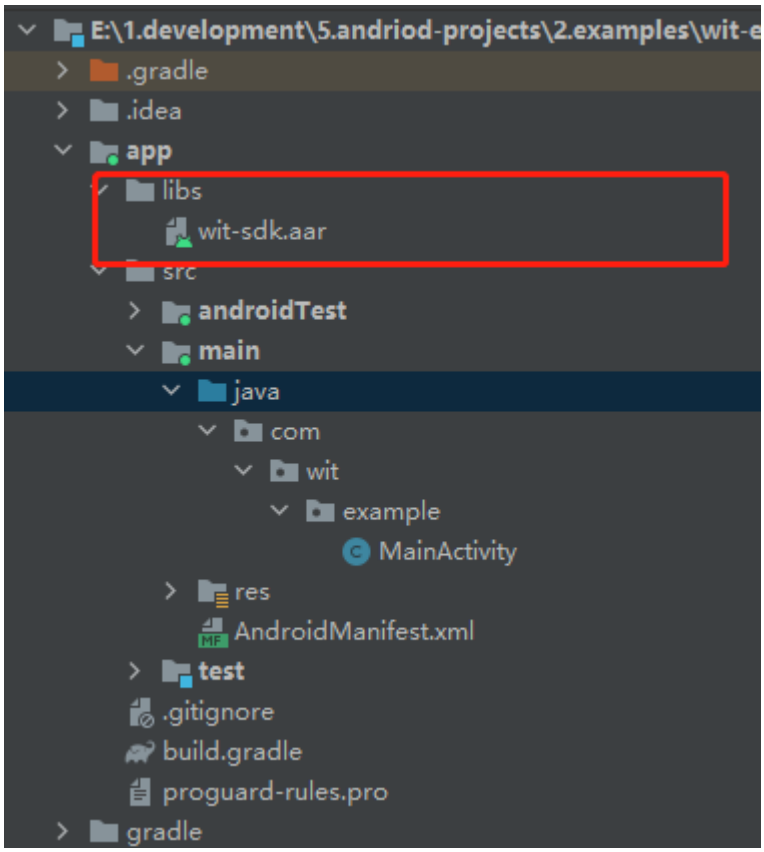
Sample code list

libs: project dependency files
java: project source code
MainActivity: The main page of the project, the routine has only one activity, and all the logic code is in this activity



Sample code dependencies

This routine depends on the wit-sdk project. If you want to port it to your own app, please port wit-sdk.aar in the lib folder in the routine to your app



Add the following dependencies to build.gradle

```
dependencies {
    // This dependency must be added
    implementation fileTree(include: ['*.jar', '*.aar'], dir: 'libs')

    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

// must add this dependency
implementation fileTree(include: ['*.jar', '*.aar'], dir: 'libs')

Project permissions

This project needs to be connected to Bluetooth, so you need to obtain permission from the customer. If you need to port it to your app, please make sure that your app also applies for the following permissions

1. AndroidManifest.xml main manifest file permission declaration
Please declare the following permissions in the program main manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.wit.example">
    <uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Witexampleble5"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2. Code application permission

Before using bluetooth, please call WitBluetoothManager.initInstance to initialize the bluetooth manager. When the bluetooth manager is initialized, the bluetooth manager will apply for permission from the user

```
/**
     * activity when created
     *
     * @author huangyajun
     * @date 2022/6/29 8:43
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize the bluetooth manager, here will apply for bluetooth permission
        WitBluetoothManager.initInstance(this);

        // start search button
        Button startSearchButton = findViewById(R.id.startSearchButton);
        startSearchButton.setOnClickListener((v) -> {
            startDiscovery();
        });

        // stop search button
        Button stopSearchButton = findViewById(R.id.stopSearchButton);
        stopSearchButton.setOnClickListener((v) -> {
            stopDiscovery();
        });

        // Plus meter calibration button
        Button appliedCalibrationButton = findViewById(R.id.appliedCalibrationButton);
        appliedCalibrationButton.setOnClickListener((v) -> {
            handleAppliedCalibration();
        });

        // Start magnetic field calibration button
        Button startFieldCalibrationButton = findViewById(R.id.startFieldCalibrationButton);
        startFieldCalibrationButton.setOnClickListener((v) -> {
            handleStartFieldCalibration();
        });

        // end magnetic field calibration button
```

```
        Button endFieldCalibrationButton = findViewById(R.id.endFieldCalibrationButton);
        endFieldCalibrationButton.setOnClickListener((v) -> {
            handleEndFieldCalibration();
        });

        // read 03 register button
        Button readReg03Button = findViewById(R.id.readReg03Button);
        readReg03Button.setOnClickListener((v) -> {
            handleReadReg03();
        });

        // auto refresh data thread
        Thread thread = new Thread(this::refreshDataTh);
        destroyed = false;
        thread.start();
    }
```

Search device

After applying for permission, please use Bluetooth manager start searching for bluetooth devices

Search for Bluetooth

The following is the code to search for Bluetooth. You need to get the Bluetooth manager, and then call the startDiscovery method. If you need to get the device found by Bluetooth management, you need to call registerObserver. After the device is found, it will notify you of the found Bluetooth device through the onFoundSPP method of registerObserver.

```
/**
     * Start searching for devices
     *
     * @author huangyajun
     * @date 2022/6/29 10:04
     */
    public void startDiscovery() {

        // close all devices
        for (int i = 0; i < bwt901clList.size(); i++) {
            Bwt901cl bwt901cl = bwt901clList.get(i);
            bwt901cl.removeRecordObserver(this);
            bwt901cl.close();
```

```
        }

        // clear all devices
        bwt901clList.clear();

        // start searching for bluetooth
        try {
            // get bluetooth manager
            WitBluetoothManager bluetoothManager = WitBluetoothManager
            . getInstance();
            // Monitor communication signals
            bluetoothManager.registerObserver(this);
            // start searching
            bluetoothManager.startDiscovery();
        } catch (BluetoothBLEException e) {
            e.printStackTrace();
        }
    }
```

After starting the search, the onFoundSPP method will be called when the device is searched. Here, the device is connected immediately after the device is searched.

```
/**
     * This method will be called back when a Bluetooth 2.0 device is found
     *
     * @author huangyajun
     * @date 2022/6/29 8:46
     */
    @Override
    public void onFoundSPP(BluetoothSPP bluetoothSPP) {
        // Create a Bluetooth 2.0 sensor connection object
        Bwt901cl bwt901cl = new Bwt901cl(bluetoothSPP);

        // Avoid duplicate connections
        for (int i = 0; i < bwt901clList.size(); i++) {
            if (Objects.equals(bwt901clList.get(i).getDeviceName(),
            bwt901cl.getDeviceName())) {
                return;
            }
        }

        // add to device list
        bwt901clList.add(bwt901cl);
```

```
        // register data record
        bwt901cl.registerRecordObserver(this);

        // open the device
        try {
            bwt901cl.open();
        } catch (OpenDeviceException e) {
            // Failed to open device
            e.printStackTrace();
        }
    }
```

Stop searching

After starting the search, the Bluetooth manager will not stop the search by itself, you need to call its stopDiscovery method to stop the search

```
/**
     * Stop searching for devices
     *
     * @author huangyajun
     * @date 2022/6/29 10:04
     */
    public void stopDiscovery() {
        // stop searching for bluetooth
        try {
            // get bluetooth manager
            WitBluetoothManager bluetoothManager = WitBluetoothManager
            .getInstance();
            // unregister listening for bluetooth
            bluetoothManager.removeObserver(this);
            // stop searching
            bluetoothManager.stopDiscovery();
        } catch (BluetoothBLEException e) {
            e.printStackTrace();
        }
    }
```

Receive sensor data

Acquire data

The sensor data can be obtained through the getDeviceData method. getDeviceData needs to receive a key value, which is stored in the WitSensorKey class

```
/**
 * Get a device's data
 *
 * @author huangyajun
 * @date 2022/6/29 11:37
 */
private String getDeviceData(Bwt901cl bwt901cl) {
    StringBuilder builder = new StringBuilder();
    builder.append(bwt901cl.getDeviceName()).append("\n");

    builder.append(getString(R.string.accX)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AccX)).append("g \t");
    builder.append(getString(R.string.accY)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AccY)).append("g \t");
    builder.append(getString(R.string.accZ)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AccZ)).append("g \n");
    builder.append(getString(R.string.asX)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AsX)).append("°/s \t");
    builder.append(getString(R.string.asY)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AsY)).append("°/s \t");
    builder.append(getString(R.string.asZ)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AsZ)).append("°/s \n");
    builder.append(getString(R.string.angleX)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AngleX)).append("° \t");
    builder.append(getString(R.string.angleY)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AngleY)).append("° \t");
    builder.append(getString(R.string.angleZ)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.AngleZ)).append("° \n");
    builder.append(getString(R.string.hX)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.HX)).append("\t");
    builder.append(getString(R.string.hY)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.HY)).append("\t");
    builder.append(getString(R.string.hZ)).append(":").append(bwt901cl.
    getDeviceData(WitSensorKey.HZ)).append("\n");
    builder.append(getString(R.string.versionNumber)).append(":").append(bwt901cl.
```

```
            getDeviceData(WitSensorKey.VersionNumber)).append("\n");
            return builder.toString();
        }
```

Record data

When the device is turned on, you can call the registerRecordObserver method of bwt901cl, when the sensor data is updated, bwt901cl will call the onRecord method to notify you to record the data

```
/**
         * This method will be called back when a Bluetooth 2.0 device is found
         *
         * @author huangyajun
         * @date 2022/6/29 8:46
         */
        @Override
        public void onFoundSPP(BluetoothSPP bluetoothSPP) {
            // Create a Bluetooth 2.0 sensor connection object
            Bwt901cl bwt901cl = new Bwt901cl(bluetoothSPP);

            // Avoid duplicate connections
            for (int i = 0; i < bwt901clList.size(); i++) {
                if (Objects.equals(bwt901clList.get(i).getDeviceName(),
                bwt901cl.getDeviceName())) {
                        return;
                }
            }

             // add to device list
             bwt901clList.add(bwt901cl);

             // register data record
             bwt901cl.registerRecordObserver(this);

             // open the device
             try {
                  bwt901cl.open();
             } catch (OpenDeviceException e) {
                  // Failed to open device
                  e.printStackTrace();
             }
```

```
    }
```

```
/**
     * This method will be called back when data needs to be recorded
     *
     * @author huangyajun
     * @date 2022/6/29 8:46
     */
    @Override
    public void onRecord(Bwt901cl bwt901cl) {
        String deviceData = getDeviceData(bwt901cl);
        Log.d(TAG, "device data [ " + bwt901cl.getDeviceName() + "] = " + deviceData);
    }
```

Set up the sensor

Acceleration calibration

A summation calibration can be performed by calling AppliedCalibration of Bwt901cl. Remember to unlock the register before adding meter calibration

```
/**
     * Make all equipment plus meter calibrated
     *
     * @author huangyajun
     * @date 2022/6/29 10:25
     */
    private void handleAppliedCalibration() {
        for (int i = 0; i < bwt901clList.size(); i++) {
            Bwt901cl bwt901cl = bwt901clList.get(i);
            // unlock register
            bwt901cl.unlockReg();
            // send command
            bwt901cl.appliedCalibration();
        }
        Toast.makeText(this, "OK", Toast.LENGTH_LONG).show();
    }
```

Magnetic Field Calibration

You can control the start and end of magnetic field calibration by calling the startFieldCalibration and endFieldCalibration methods of Bwt901cl. After starting magnetic field calibration, please make 2-3 turns around each axis of the sensor's x, y, and z axes. If you don't know about magnetic field calibration, you can consult our support team.

Start magnetic field calibration

```
/**
     * Let all devices start magnetic field calibration
     *
     * @author huangyajun
     * @date 2022/6/29 10:25
     */
    private void handleStartFieldCalibration() {
        for (int i = 0; i < bwt901clList.size(); i++) {
            Bwt901cl bwt901cl = bwt901clList.get(i);
            // unlock register
            bwt901cl.unlockReg();
            // send command
            bwt901cl.startFieldCalibration();
        }
        Toast.makeText(this, "OK", Toast.LENGTH_LONG).show();
    }
```

End magnetic field calibration

```
/**
     * Let all devices end magnetic field calibration
     *
     * @author huangyajun
     * @date 2022/6/29 10:25
     */
```

```
    private void handleEndFieldCalibration() {
        for (int i = 0; i < bwt901clList.size(); i++) {
            Bwt901cl bwt901cl = bwt901clList.get(i);
            // unlock register
            bwt901cl.unlockReg();
            // send command
            bwt901cl.endFieldCalibration();
        }
        Toast.makeText(this, "OK", Toast.LENGTH_LONG).show();
    }
```

Read sensor register

This demonstration shows how to read the 03 register of the sensor. Use the sendProtocolData method to read the sensor data. After reading, you can get the register data through getDeviceData.

```
/**
     * Read 03 register data
     *
     * @author huangyajun
     * @date 2022/6/29 10:25
     */
    private void handleReadReg03() {
        for (int i = 0; i < bwt901clList.size(); i++) {
            Bwt901cl bwt901cl = bwt901clList.get(i);
            // The sendProtocolData method must be used, and the device will read the
            register value using this method
            int waitTime = 200;
            // The command to send the command, and wait 200ms
            bwt901cl.sendProtocolData(new byte[]{(byte) 0xff, (byte) 0xAA, (byte) 0x27,
            (byte) 0x03, (byte) 0x00}, waitTime);
            // get the value of register 03
            String reg03Value = bwt901cl.getDeviceData("03");
            // If it is read, reg03Value is the value of the register. If it is not read, waitTime
            can be enlarged, or read several times
            Toast.makeText(this, bwt901cl.getDeviceName() + " reg03Value: " +
        reg03Value, Toast.LENGTH_LONG).show();
        }
    }
```