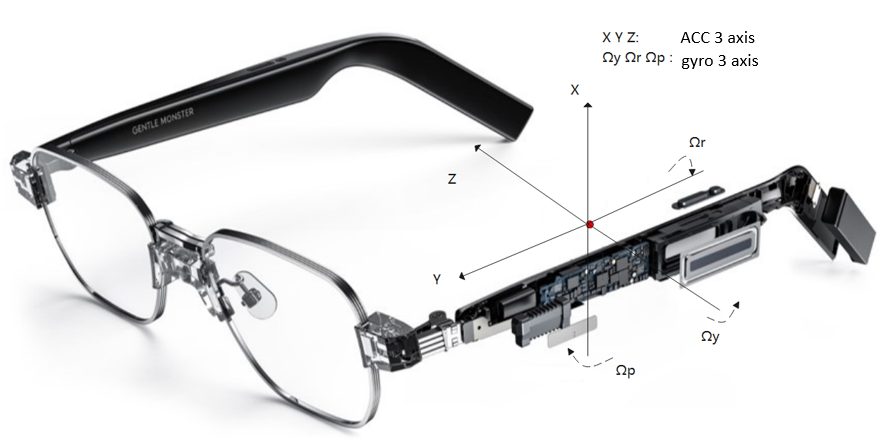
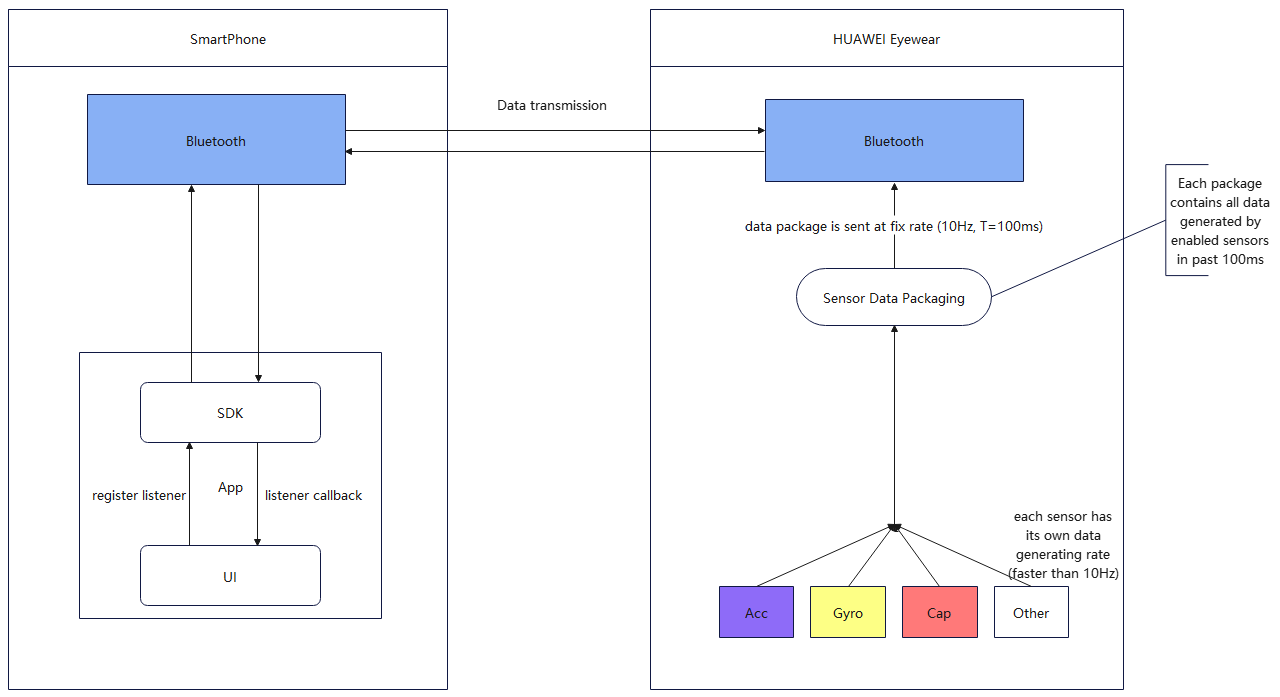
**HUAWEI Smart Glasses SDK Guide**

# Eyeglass Sensor Orientation Diagram and Data Interaction Flowchart

* Sensor Orientation Diagram

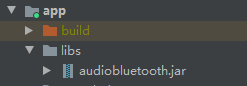


* Data exchange process



# SDK Access Guide

* Add dependency: Place the SDK file audiobluetooth.jar in the libs directory of the app module.



Add the following configuration to the app/build.gradle file to depend on the JAR package in the libs folder.

implementation fileTree (dir: 'libs', include: ['\*.jar'])

Note: it cannot be compileOnly fileTree. (dir: 'libs', include: ['\*.jar'])

* Permission statement: Add Bluetooth and location permissions to the AndroidManifest.xml file for scanning devices.

<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

Note: For the development environment of Android 12+, the following Bluetooth permissions need to be specified and dynamically applied for in the code.

<uses-permission android:name="android.permission.BLUETOOTH\_CONNECT" />  
<uses-permission android:name="android.permission.BLUETOOTH\_SCAN" />  
<uses-permission android:name="android.permission.BLUETOOTH\_ADVERTISE" />

* The procedure for invoking the AudioBluetoothApi interface is as follows:

1. Initialization: init
2. Device scanning: startSearch
3. Connect device: connect
4. Register the listening and receiving of device data: registerListener
5. Sending instruction: sendCmd
6. Disconnect the device: disconnect

**The following table lists the detailed API statements.**

|  |  |  |  |
| --- | --- | --- | --- |
| method Name | Input Parameter | Return/Calback Return | Purpose |
| init | Context: system context of the SDK caller, which cannot be empty.  InitResultCallback callback: initialization result callback. This parameter can be null. | void  InitResultCallback.onResult:  boolean true Initialization completed false Initialization failed | Initializes the Bluetooth module. All operations depend on the initialization of the Bluetooth module. If the initialization is not complete, will throw AudioBluetoothException when any other method is invoked |
| deInit | / | void | Anti-init method |
| connect | String mac: MAC address of the device to be connected.  ConnectionStateListener listener: The value can be null. Returns the connection status | void  ConnectionStateListener:  DISCONNECT: The device data channel is disconnected.  Connecting, device data channels are being connected.  CONNECTED: The device data channel is connected. | Connects to a device. The device can call this interface in any state. The expected result is that the device and app can establish a data communication channel. |
| isConnected | String mac: indicates the MAC address to which the connection status is to be queried. | bool  true: The data channel is connected; false: The data channel is not connected. | Query the device connection status. If the return value of this status is true, instruction interaction can be performed. Otherwise, instruction interaction cannot be performed. |
| disconnect | String mac: MAC address of the device to be disconnected.  ConnectionStateListener listener: The value can be null. The connection status is the callback. | / | Disconnect the device. The device can call this API in any state. The expected result is that the device disconnects the data interaction channel with the app. |
| searchDevice | FoundCallback callback: device callback | void  FoundCallback:  void onFound(DeviceInfo deviceInfo);  void onFinish(); | Start device scanning to find the glasses to be matched near the phone where the app is installed. |
| stopSearch | / | void | Stop Device Scanning |
| sendCmd | String mac: MAC address of the device,  int cmdType: instruction type, IRspListener<Object> listener: response listening callback | void  void onSuccess(T object);  void onFailed(int errorCode); | Send instructions. 21 types of instructions can be sent. For the instruction type declaration, see the Cmd enumeration definition table. |
| registerListener | String mac: MAC address of the listening device.  INotifyListener listener: data listener | void  void onNotify(ReceiveDataEvent result); | Registering the Listener for Reporting Data |
| unregisterListener | String mac: MAC address of the listening device. | void | Unregister the listener |

**The following table lists the supported instructions encapsulated by the Cmd enumeration class.**

|  |  |  |
| --- | --- | --- |
| No. | Command Name | Enumerated Value |
| 1 | Querying the Version Number | VERSION |
| 2 | Querying the Current Status of the UX State Machine | STATE\_GET |
| 3 | emulate box close | BOX\_CLOSE |
| 4 | emulate box opening | BOX\_OPEN |
| 5 | emulate Box Entry | INBOX |
| 6 | emulate Box Exit | OUTBOX |
| 7 | emulate long press | PRESS\_3S |
| 8 | emulate press for 2 seconds | PRESS\_DOWN |
| 9 | emulate press lift | PRESS\_UP |
| 10 | emulate forward slide | SLIDE\_UP |
| 11 | emulate back slide | SLIDE\_DOWN |
| 12 | emulate wear | IN\_EAR |
| 13 | emulate remove | OUT\_EAR |
| 14 | emulate Left side Double Click | DC |
| 15 | emulate Right side Double Click | DC\_PEER |
| 16 | Reads acc data. | READ\_ACC |
| 17 | Reading Gyro Data | READ\_GYRO |
| 18 | Reading the sliding cap value | READ\_SLIDE\_CAP |
| 19 | Reading the cap values of wearing and pressing | READ\_WEAR\_CAP |
| 20 | Enable Sensor Data Reporting | SENSOR\_DATA\_UPLOAD\_OPEN |
| 21 | Disable Sensor Data Reporting | SENSOR\_DATA\_UPLOAD\_CLOSE |

**Description of abbreviations:**

|  |  |
| --- | --- |
| Acronyms | Full name in English |
| Acc | acceleration |
| gyro | gyroscope |
| cap | capacitance |

**Example of sending an instruction:**

AudioBluetoothApi.sendCmd("", Cmd. VERSION.getType(),listener);

**Sensor data reporting and analysis:**

When the registerListener method is used to register the listener for reporting of sensor data, the following method is used to parse the sensor data in the callback method onNotify(ReceiveDataEvent result):

byte[] appData = result.getAppData();

SensorData sensorData = SensorDataHelper.genSensorData(appData);

getUi().onSensorDataChanged(sensorData);

**Note: Sensor data structure description**[**See below**](#_二．传感器说明)**.**

* SDK Usage Example

For details, see SampleProject.

The sample code is developed in the standard MVP architecture mode. How to develop with the SDK can be based on the architecture mode that you are familiar with.

* Frequently Asked Questions
* **Cannot find the device?**

1. Check whether the glasses are in the scanable state. When the glasses are in the scanable state, the indicator on the charging cable flashes white.
2. Check whether the Bluetooth function is enabled on the phone. If not, turn on the Bluetooth function.
3. Check whether the permission statement is added. For details about the permission restrictions necessary for scanning devices, see the earlier description.
4. Check whether the location permission is disabled when you apply for the first time. If selected disable, you need to enable the location permission on Settings > Apps and Services > App Management page, find your project, go to the Permission page, and manually enable the location information permission. The permission setting paths may vary slightly for different phone brands, but the paths are similar.

* **All API calls in the SDK become invalid.**

Check whether the phone time is set to automatic. If the time is manually set and the time is set for one year later, the API may fail to be invoked. In this case, set the phone time to automatic obtaining mode.

* **Can I update my glasses firmware?**

Cannot to update the firmware of the glasses. Specified firmware versions are pre-installed in the glasses and cannot be upgraded or re-flushed.

# III. Sensor Data Description

* Sensor Data Field Descriptions

The sensor data reported by the glasses side is as follows:

|  |  |
| --- | --- |
| CommonSensorsDataSt structure description | |
| Field | Description |
| sensorType | Include Sensor Types  TOUCH:(0x01 << 0)  ACC: (0x01 << 1)  GYRO: (0x01 << 2)  SLIDE: (0x01 << 3)  PROX(cap):(0x01 << 4)  P :(0x01 << 5)  VIRTUAL :(0x01 << 6)  PPG :(0x01 << 7)  TEMP: (0x01 << 8) |
| time | System ticking clock, in ms |
| packageNumber | Package Count |
| volt | Current voltage, in mV. 1 volts (V) = 1000 millivolts (mV) |
| chargeCurrent | charging current |
| battPercent | Battery volume |
| temperature | Temperature, in degrees Celsius |
| halldata | Whether the current status is in the box or out of the box.  0: in box  1: out of the box |
| earside | Left frame leg or right frame leg  1: left leg  2: right leg |
| rolestate | Main leg or auxiliary leg  1: main leg  2: auxiliary leg |
| weardetect | Overall Wearing Status  1: wearing  0: not worn |
| knockDect | clicking event  1: Click.  2: double-click  3: three clicks  4: ear tapping  Note: Currently, glasses support only double-clicking events. |
| KnockScore | Knock score  Knock events judged by the knock score, such as double-click and triple-click |
| movementDetect | Not Used |
| touchevent | Not used at present |
|  |
| accelDataLen | ACC data length |
| accelData | Acceleration data set  Acc{  x, y, and z represent acceleration values of x, y, and z axes, respectively.  }  **The calculation formula and unit of accelData are as follows:**  Unit: m/s2. Conversion relationship: 4096 = 1 g. Value range: ±8 g (±32768). Note: Only the left leg supports this function. |
| acctimestamp | Acceleration timestamp (unit: us) |
| gyroDataLen | Gyro data length |
| gyroData | Gyro data set  Gyro{  roll, pitch, yaw  }  **The formula and unit of the gyroData data are as follows:**  Unit: deg/s. Conversion relationship: 16.384 = 1 dps. Value range: ±2000 dps. Note: Only the left leg supports this function. |
| capSensorDataLen | Capacitive Sensor Data Length  The six channels in the cap data are values related to the detection of wearing.  Raw is raw data. 0 to 5 correspond to channels 0 to 3 of cap1. And values of channels 3–4 of cap2.  channel data:  channel 0 2:  Sensed value of cap1 (raw: sensed value + noise value. Here, data is raw - noise value.)  channel 1 3:  Reference value of cap1. If the value exceeds the value, the value is 1. Otherwise, the value is 0.  channel 4: diff value of cap2 (the sensing value mentioned above)  channel 5: reference value of cap2 |
| capSensorData | Capacitive sensor data set, length 6 |
| capRawData | Capacitive sensor raw data set, with a length of 6 |