Sambo **Op EN SDK** description document

Author	Date	Version	Modify / Add / Delete
Cris_Peng	201 . 9 / 11 / 6	2.0.1.10	1, increase the robot type query interface (3.7.10)
Cris_Peng	201 . 9 / 10 / 28	2.0.1.10	1.Pixie robot arm control protocol change: from the original HandMotionManager to WingMotionManager control, the corresponding motion control class was changed to RelativeAngleWingMotion , AbsoluteWingHandMotion , NoAn gleWingMotion (corresponding protocol update such as 3.5 wings motion control) 2.Modification of the Elf Wheel Non-Angle Control Protocol: For details, see 3.6.1 Mainly Field Modification 3.Compatible with D head motor control protocol, D head motion control, please use DRelativeAngleHeadMotion , DAbsoluteAngleHeadMotion , D N oAngleHeadMotion (three classes with the use of D funds SDK consistent use of documents inside, just like the name has changed, the head elf version control, see article 3.4) HeadMotionManager corresponds added D paragraph robot control method doDNoAngleMo tion of , DoDAbsoluteAngleMotion , DoDRelativeAngleMotion
Cris_Peng	201 . 9 / 6 / 3	2.0.1.8	1.Add desktop motion control protocol (3.15)
Cris_Peng	201 . 9 / 3 / 7	2.0.1.7	1. The client's heartbeat is maintained in the child thread to avoid being blocked by the main thread 2.Add desktop robot face tracking switch and status callback (3.14)
Cris_Peng	2018/ 10 / 26	2.0.1.4	1.Add wheel movement callback (see 3.6.4 for details) 2, add the gyroscope data callback
Cris_Peng	201 8/ 10 / 10	2.0.1.3	1.Added recognition language switching (3.2.16 is only applicable to foreign versions) 2.Add prohibition of semantic request pre-configuration (3.11.9 is

			only applicable to foreign versions) 3, update the pull video stream related methods, remove the MediaManager management class, replace with HDCameraManager
Cris_Peng	2018/ 9 / 12	2.0.1.2	1. Add anti-killing configuration (3.11.8) 2, the App the suspension system hidden button method (3.7.9)
Cris_Peng	2018/8/3	2.0.1.1	1. Whether the wake-up interface is added through voice wake-up or wake-up caused by other reasons (touch, call SpeechManager.doWakeUp method), please refer to 3.2.5 for changes . 2. Add misrecognition mode. At present, the main control will actively intercept and filter the recognition of a word. After setting the misrecognition mode, the master no longer intercepts the recognition of a word (see 3.11.7 Misrecognition Mode) 3, add the interface to cancel the last recognition (currently used internally) This update needs to be valid for system versions after 2018/8/3
Cris_Peng	6/21/2018	2.0.1.0	Identification interface return: Add identification start, identification end, identification error return (for domestic version only) For details, please see 3.2.13, 3.2.14, 3.2.15

table of Contents

SDK version change description

Version No. 2.0.1.10:

1, increase the robot type query interface (3.7.10)

Version No. 2.0.1.9:

1, elf robot arm control protocol change: the original HandMotionManager changed WingMotionManager control, motion control corresponding class instead RelativeAngleWingMotion, AbsoluteWingHandMotion, NoAngleWingMotion (corresponding protocol updates as 3.5 wings Motion Control)
2. Modification of the Elf Wheel Non-Angle Control Protocol: For details, please see 3.6.1 The main modification is the constant modification of the motion mode.
3. Compatible with D head motor control protocol, D head motion control, please use DRelativeAngleHeadMotion
, DAbsoluteAngleHeadMotion, D N oAngleHeadMotion (three classes with the use of D funds SDK consistent use of documents inside, just like the name has changed, the head elf version control, see article 3.4)
HeadMotionManager correspondingly added D robot control methods doDNoAngleMotion, doDAbsoluteAngleMotion, doDRelativeAngleMotion

Version No. 2.0.1.8:

1, add desktop robot control protocol (3.15)

Version No. 2.0.1.7:

1. The client's heartbeat is maintained in the child thread to avoid being blocked by the main thread
2, add desktop robot face tracking switch, status callback (Desktop Robot)

Version No. 2.0.1.4:

1, add the wheel motion callback
2, add the gyroscope data callback

Version No. 2.0.1.3:

1. Added recognition language switching (3.2.16 is only applicable to foreign versions)
2. Add prohibition of semantic request pre-configuration (3.11.9 is only applicable to foreign versions)
3, update the pull video stream related methods, remove the MediaManager management class, replace with HDCameraManager

Version No. 2.0.1.2:

1, the background music or killing configuration
2, add another floating button to hide the function, Activity jump within the same APP will not flash again

Version No. 2.0.1.1:

1. Add the wake-up interface to wake up by voice or other reasons (touch, call the SpeechManager.doWakeUp method)
2. Add misrecognition mode. At present, the main control will actively intercept and filter the recognition of a word. After setting the misrecognition mode, the master no longer intercepts the recognition of a word
3, add the interface to cancel the last recognition (currently used internally)

Version No. 2.0.1.0:

1, modify the SDK's package called com.sanbot.opensdk and SDK named SanbotOpenSDK.

2, increase recognition start, end recognition, recognition errors returns an interface (for details see 3.2.13,3.2.14,3.2.15)

Version No. 2.0.0:

3, modified the SDK core communication mechanism , to replace the SDK written language .

4, modify the communication mechanism of the projector, the projector part of the interface has been adjusted.

5, increases the barrier detection interface.

6, now Activity is also required in the onCreate method call **register (Class subClass)** method , the Activity of Class register objects to the SDK in.

The version number 1.1.8:

1, increases the acquisition gyro sensor data interface.

2, the hand movement control of the relative angular movement, the absolute angular movement increases simultaneously control two hands together methods motion.

3, increase the custom global command word and custom semantic function.

4, correct the touch event callback one and wheel-free angular movement of the document description.

5, voice recognition callback adds open semantic function.

6, facial recognition callback interface now returns all people can recognize facial data .

7, fix a projector problem Interface failure.

8, repair facial recognition in Activity is the callback will occur after the destruction of bug.

9, repairing high frequency transmission instruction, the switching Activity may occur when the page is the ANR of bugs.

10, repair SDK could lead to the collapse of the master bug.

11, increasing the display /hide the system F loatBar function interface.

Version number 1.1.7:

1, increases the module Movement manager, allows for free running , automatic charge control function .

2, increase the preprocessor directive function , before the recording, the voice mode , truncated configuration functions are all integrated into the unified command in the pretreatment.

3, remove the power manager, obtaining the robot charge and interface state of charge moved to the system manager.

4, now Service needs in the onCreate method call **Regi STER (Class subClass)** method , the Service's Class registration object to the SDK in.

5, increases the pause, continue, speech synthesis is stopped, obtaining a master version number , obtaining infrared range a result, the black line switch filtered luminance white light is provided , monitor home security alarm interface.

6, is compatible with the future of semantic function, identifying discourse callback interfaces change .

1 Overview

This document is intended for Android developers.

This document is used to guide developers to quickly access the Sambo Open SDK. This SDK provides Android applications with functions related to controlling robot voice and actions .

The description of the name of the robot involved in the document: Section B = Pokemon ;Section D = King Kong; Desktop Robot = Mini Pokemon

2 access process

2.1 Development Environment

This SDK can adopt Android Studio or Eclipse and other development tools to develop, recommend Android Studio software version is 1.5 or more versions.

JDK requires the use of JDK7 and above versions, Android SDK's minSdkVersion minimum is 11 .

2.2 configuration instructions

In order to meet the development needs in different environments, we provide three different SDK versions, respectively: Sanbot OpenSDK_XXX.aar , JarA , JarB .

Wherein JarA and JARB distinguished: JARB not provide BindBaseActivit Y , TopBindActivity and BindBaseService these base classes that are needed BindBaseUtil class implements the communication with the robot.

2.2..1 AAR EDITION SDK configuration instructions

First , connect the provided Sanbot OpenSdk_XXX.aar (XXX is the version number) and gson-2.2.4.jar file copied to the M odule the libs folder .

In A PP M odule of build.gradle file in the folder add the following elements:

```
repositories { flatDir { dirs 'libs' } }


dependencies {    compile (name: ' Sanbot OpenSdk_XXX', ext: 'aar')

}
```

2.2.2 JAR version of the SDK configuration instructions

1, please Jar A folder (or JarB folder , depending on your specific needs using the A section Jar package or B section Jar package) src / libs directory files copied to all of your project in the libs folder .

2, please Jara (or JarB) folder src / res directory under the resource file with your project resource files were merged.

3, please in your project AndroidManifest.x ml file add the following two permissions:

```
<uses-permission android: name = "android.permission.CAMERA" />
<uses-permission android: name = "android.permission.INTERNET" />
<uses-permission android: name = "android.permission.ACCESS_WIFI_STATE" />
<uses-permission android: name = "android.permissi on.ACCESS_NETWORK_STATE" />
```

4, if you're using Android Studio tool for development, please remember to configure so file read path , in M odule of build.gradle file in the folder and add the following contents:

```
sourceSets.main { jniLibs.srcDir 'libs' }
```

2.3 confusing instructions

For applications that integrate the Sambo SDK, please note that the methods related to the Sambo SDK should not be confused. The obfuscation method is as follows:

```
- Keep class . COM qihancloud . Opensdk. ** { *; }
-keep class com.sanbot.opensdk. ** {*;}
-keep class com.sunbo.main. ** {*;}
```

Ensure SDK's class can not be confused with, otherwise it will appear on the robot control failure and other runtime exception

2.4 matters needing attention

1. At present the development process requires the use of R & D systems we provide for testing, formal system needs 2.2 1 before and later versions to support existing features.
2. The system does not include the Google service framework, which means that all services provided by Google cannot be used, such as Google Maps, Google Login, etc. Please developers try to avoid using the corresponding SDK of Google Android open platform . 3. Please carefully monitor system boot broadcast, broadcast transmission system boot time is to repair instead issued for the first time into the desktop system boot, before use, please carefully consider this will not affect your program. 4. Robot stops after operating for some time in the user, automatically return to the desktop, if you do not want your App in the open position is return to the desktop, please let your App to call Android system SDK open the screen lit function . The reference code is as follows:

```
getWindow (). setFlags (WindowManager.LayoutParams. FLAG_KEEP_SCREEN_ON ,
WindowManager.LayoutParams. FLAG_KEEP_SC REEN_ON );
```

3 Coded Access Details

3.1 SDK's basic description

The SDK only supports controlling robots in Activity and Service components .

3.1.1 Activity inherited class described

All Acitivity classes in the project please inherit BindBaseActivity class or TopBaseActivity class, where TopBaseActivity class is a subclass of BindBaseActivity . The similarities and differences between BindBaseActivity and TopBaseActivity are as follows:

Same point: both need to override an abstract method onMainServiceConnected () (only called once when Activity is created).

Differences: The default status bar is displayed at the top of the page inherited from the TopBaseActivity class, and the setBodyView method should be used when setting the layout. The style of the Application should be the style of NoActionBar .

Note: If you want to control the robot as soon as the Activity starts, the corresponding control code must be placed in onMainServiceConnected (), otherwise it will be invalid in other life cycles! And need OnCreate use in the Register (XX Act ivity .Class) registered XX Activity , the Register method must be placed super.onCreate () before calling .

3.1.2 Service class inheritance explained

If you need to control the robot in the Service , you can inherit the BindBaseService class. To inherit BindBaseService class, you need to overload an abstract method onMainServiceConnected () (only called once when the service is created), and you need to register **XXServic e** with **register (XXService.Class)** in the OnCreate method **. The register** method must be placed in **super.onCreate ( )** before the call **;** . If it is necessary to control the robot as soon as the **Service** is started, the corresponding control code must be in onMainServiceConnected (), otherwise it is invalid.

Sample code:

```
public class SampleService extends BindBaseService {
...
  @Override
   protected void onMainServiceConnected () {
  control code
   }
...
  @Override public void onCreate () {

                        register (SampleService. class ); super .onCreate ();              }       ...


}
```

.3..1..3 B, paragraph Jar package particularly described

B models Jar package is to solve when a developer can not inherit from BindBaseActivity and BindBaseService how when controlling a robot of the problem and provide the SDK. In the Jar package of section B, the BindBaseUtil class is replaced . This class has the following methods:

.1, **BindBaseUtil (A ctivity Activity)** or **BindBaseUtil (-Service-Service)**

The above two methods are the construction methods of B indBaseUtil .

2, **public void connectService ()**

Call this method , beginning with the robot to establish a connection master, only success to establish a connection with the robot perform communication. This method requires the Activity of () onResume life- cycle method invocation or Service of ON the Create () life cycle method invocation.

3, **public void breakConnection ()**

Call this method to disconnect from the robot master . This method requires the Activity of onStop () Life calls or cycle approach in Service of onDestroy () Life call the cycle method.

4, **public void setOnConnectedListener (OnConnectedListener onConnectedListener)**

Method to monitor whether a connection has been established with the master. When calling the **connectService ()** method to establish a connection with the master, the connection process is an asynchronous operation , and this method will be called back after the connection is successful . If you need to perform some operations as soon as the connection with the master is successfully established , you can put related code in this callback.

Please note : connectService () and breakConnection () Please strictly in accordance with the above is required into the corresponding components of the life cycle , remember, do not forget to call breakConnection () method, also do not comply with the requirements of the life-cycle approach calls connectService () And breakConnection () method ! !! !!

5, **public void register (Class subClass)**

Please in Service and Activity components of Oncreate ( ) method call this method, the current Service or Activity of the class object is registered, this method must be placed super.onCreate () method called before. If it is not registered or the registered object is wrong , the SDK's preprocessing instruction function may not work properly.

3.1.3 Method Return Value OperationResult is described

All of the robot control method will return **OperationResult** type of parameters, the parameters for feedback the method performs the result: the implementation is successful , the implementation of the cause of failure , execution success after the return value.

OperationResult in there are the following three methods:

I public NT getErrorCode () : instruction execution result, 1 indicates success, the other is less than 0 the value indicates failed .

String getDescription public () : Instruction Description result of execution, if the instruction fails, will instruction execution cause of the failure will be specifically described.

public String getResult () : additional field, generally used to store feedback data after instruction execution.

3.2 Voice Control

Access process:

1, application SpeechManager objects ,application codes as

```
SpeechManager speechManager = ( SpeechManage r ) getUnitManager (FuncConstant.
SPEECH_MANAGER );
```

2, use application of speechManager the object , call the appropriate methods for control.

.3.2..1 Speech Synthesis

Method signature:

public OperationResult startSpeak (String text)

Method description :

The speech synthesis method is used to control the robot to speak the specified text content . The language of the synthesized speech is related to the current system language setting.

Sample code:

```
speechManager.startSpeak (" This is a test text ");
```

.3.2.2 speech synthesis (designated synthesis parameters)

Method signature:

public OperationResult startSpeak (String text, SpeakOption speakOption)

Method description :

The overload method of the speech synthesis method can specify the language , speed and intonation of the synthesized speech .

Parameter Description:

SpeakOption has the following parameters :

languageType indicates the language of the synthesized speech. The value range is LAG_CHINESE (Chinese

), LAG_ENGLISH_US (English _ the United States).

S peed represents synthesized speech of the speech rate, in the range of 0 ~ 100 .

Intonation indicates the tone of the synthesized speech, and the value ranges from 0 to 100 .

Sample code:

```
SpeakOption speakOption = new SpeakOption (); speakOption.setSpeed (70);
OperationResult operationResult = speechManager .startSpeak ( " This is a test statement " , speakOption);
```

3.2.3 Sleep

Method signature:

public OperationResult doSleep ()

Method description :

Put the machine to sleep and stop talking.

Sample code:

```
speechManager. doSleep () ;
```

3.2.4 Wake

Method signature:

public OperationResult doWakeUp ()

Method description :

Actively put the machine into the awake state . This method can only be called in Activity . Calling in Service will not take effect.

Sample code:

```
speechManager. doWakeUp () ;
```

3.2.5 wake-sleep state callback

Method signature :

public void setOnSpeechListener (SpeechListener speechListener)

Method description :

setOnSpeechListener is a universal callback interface for voice control . Parameters passed determine the callback interface is a callback function which belongs . This callback method is triggered when the robot goes to sleep or wakes up .

Sample code:

```
speechManager .setOnSpeechListener ( new WakenListener () { @Override public void onWakeUp () { // wake event callback


      } @Override public void onSleep () { // sleep event callback } @Override public void onWakeUpStatus ( boolean b) { // wake event callback b == true represents the event waked up by the wake word       } }) ;
```

.3.2..6 identified utterance callback

Method signature :

public void setOnSpeechListener (SpeechListener speechListener)

Method description :

Please note : as compatible follow-up of semantic features, this interface from 1.1.7 onwards undergone changes .

This interface provides all robot-recognized characters except wake words to third-party applications. During this process, the third-party application can decide whether to enable the truncation function.). Be sure to note is that robots do not speak the user's sleep state to do the identification.

By default truncation is in the closed state, such as the need for this activity open truncation, please AndroidManifest.xml increase following documents preprocessing instructions (on preprocessing command description may refer to 3.11 Section) : <Meta-Data Android : name = "RECOGNIZE_MODE" a ndroid: v alue = "1" />

Note : The execution time of the code in the onRecognizeResult callback method should not exceed 500ms, otherwise it will affect the normal work of the entire system , and it is not recommended to do truncation in Service .

Parameter Description:

Robot recognize a word callback implementation is RecognizeListener the interface, in this interface, the need to implement the following methods:

boolean onRecognizeResult (Grammar grammar);

Users can get the text recognized by the robot through the grammar.getText () method .

The return value of onRecognizeResult is a boolean. The return value is set by the developer according to the actual business needs. When the return value is true, it means that the robot no longer responds to the text, and false otherwise.

Grammar objects included in the robot recognition to the user to speak of content as well as user words were semantically parsing result, the current semantics only supports the use of hearing fly semantic engine of the system , Grammar explanation classes, please refer to ″ Sambo open semantic documentation″ .

Sample code:

```
speechManager.setOnSpeechListener (new RecognizeListener () {
 @Override
 public boolean onRecogni zeResult ( Grammar grammar ) { System.out.print ("
The text recognized by the robot is " + grammar.getText () );          return true ;
 } });
```

3.2.7 voice volume callback

Method signature :

public void setOnSpeechListener (SpeechListener speechListener)

Method description :

When the robot to recognize sounds of the surroundings, this is the callback method, to identify the returned sound volume size . This method can be used to judge the user's speaking state and obtain the magnitude of the user's speaking voice . Only when the robot is awake, the method will be called back.

Parameter Description:

Robot recognize a word callback implementation is RecognizeListener the interface, in this interface, the need to implement the following methods:

void onRecognizeVolume (int volume);

volume value returned is the robot recognition to the sound volume size , in the range of 0 to 30 .

3.2.8 if the robot is speaking

Method signature :

public OperationResult isSpeaking ()

Method description :

This method is used to determine if the current robot is speaking.

Return value description:

The getResult () method of OperationResult will return the query result. If the returned value is "1", it means that the robot is talking , and "0" is the opposite.

Sample code :

```
OperationResult operationReusult = speechManager . IsSpeaking () ;
i f ( operationResult.getResult (). equals ( " 1 " )) {
// TODO robot is talking
}
```

3.2.9 pause speech synthesis

Method signature :

public OperationResult pauseSpeak ()

Method description :

This method is used to pause the speech synthesis of the current robot .

3.2.10 Continue speech synthesis (overseas edition does not support this method)

Method signature :

```
public OperationResult resumeSpeak ()
```

Method description :

This method is used to resume the speech synthesis that the current robot has paused .

3.2.11 Stop speech synthesis (overseas edition does not support this method)

Method signature :

```
public OperationResult stopSpeak ()
```

Method description :

This method is used to stop the speech synthesis of the current robot .

.3.2..12 speech synthesis status callback

Method signature :

```
public void setOnSp eechListener (SpeechListener speechListener)
```

Method description :

This callback method is triggered when the robot is synthesizing speech .

Parameter Description:

Speech synthesis state callback implementation is SpeakListener the interface, in this interface, the need to implement the following method: **void onSpeakStatus (SpeakStatus speakStatus);**

onSpeakStatus method is the callback during speech synthesis, wherein SpeakStatus class progress parameter indicates the statement is currently synthesized progress , in the range 0 < = Percent < = 100 .

Sample code:

```
speechManager . setOnSpeechListener ( new SpeakListener () {

    @Override
    public void onSpeakStatus ( SpeakStatus speakStatus ) {
    System.out.print ( " Synthesizing speech, progress is: "   +
    speakStatus.getProgress () );    }

});
```

.3.2..1.3 -recognition-start status callback

Method signature :

```
public void setOnSpeechLis tener (SpeechListener speechListener)
```

Method description :

This method will be called back when the robot starts to recognize

Parameter Description:

Robot recognize a word callback implementation is RecognizeListener the interface, in this interface, the need to implement the following methods:

```
@Override
public void onStartRecognize () {
```

```
        Log. I ( "Cris" , " State callback when starting recognition " ) ;
}
```

.3.2..1.4 recognition of the end status callback

Method signature :

public void setOnSpeechListener (SpeechLis tener speechListener)

Method description :

This method will be called back when the robot finishes recognition

Parameter Description:

Robot recognize a word callback implementation is RecognizeListener the interface, in this interface, the need to implement the following methods:

```
@Override
public void onStopRecognize () {
        Log. I ( "Cris" , " State callback when end recognition " ) ;
}
```

.3.2..1.5 recognition error status callback

Method signature :

public void setOnSpeechListener (SpeechListener speechListe ner)

Method description :

This method will be called back when the robot recognizes an error

Parameter Description:

Robot recognize a word callback implementation is RecognizeListener the interface, in this interface, the need to implement the following methods:

```
@Override
public void onError ( int engine , int errorCode ) {
        Log. I ( "Cris" , "onError: engine =" + engine + "errorCode =" + errorCode ) ;
}
```

E ngine's current recognition engine is 0 for Xunfei and 1 for Dubi

The error codes identified by E rrorCode . For example , when engine == 0 , errorCode == 20005 , it means that no one is currently speaking. For other error codes, please check Xunfei's error code list.

.3.2..1.6 wake up and recognize a language select:

Method signature:

public OperationResult doWakeUp (WakeUpOption option)

Method description :

Actively put the machine into the awake state . This method can only be called in Activity . Calling in Service will not take effect. Foreign customers select the language of different countries need to wake up to identify, in addition to switching the default language, the new wake-up time is now passed **WakeUpOption** , **WakeUpO ption** able to configure the robot recognition languages.

Sample code:

WakeUpOption currently supports the following languages:

LAG_ARABIC_INTERNATIONAL // Arabic

LAG_CHINESE_HK // Hong Kong, China

LAG_DANISH //Danish

LAG_ENGLISH_US //English

LAG_FRENCH_FRANCE //French

LAG_GERMAN //German

LAG_ITALIAN //Italian

LAG_JAPANESE //Japanese

LAG_KOREAN //Korean

LAG_POLISH //Polish

LAG_PORTUGUESE_PORTUGAL //Portuguese

LAG_SPANISH_SPAIN //Spanish

LAG_TURKISH //Turkish

```
WakeUpOption = WakeUpOption new new WakeUpOption () ;
wakeUpOption.setLanguageType (WakeUpOption. LAG_ARABIC_IN ternational ) ;
speechManager .doWakeUp (wakeUpOption) ;
```

3.3 hardware control

Access process:

1, application HardWareManager objects ,application codes as

```
HardWareManager hardWareManager = ( HardWareManager ) getUnitManager (FuncConstant.
HARDWARE_MANAGER );
```

2, use application of hardWareManager the object , call the appropriate methods for control.

.3..3..1 Control White Light

Method signature :

public Op erationResult switchWhiteLight (boolean isOpen)

Method description :

Used to turn white light on or off.

Sample code :

```
hardWare Manager. switchWhiteLight ( true ) ;
```

.3..3.2 Control LED lamp

Method signature :

public OperationResult setLED (LED led)

Method description :

Of the robot LED lamp is controlled .

Parameter Description:

The construction method of the LED class is as follows :

```
LED ( byte part, byte mode, byte delayT ime, byte randomCount)
```

The part parameter indicates the position of the LED light on the robot,

The mode parameter indicates the control mode of the LED light,

The delayTime is only used when the LED light is in the blinking mode, which indicates the blinking interval of the LED light (please note that the delayTime value ranges from 1 to 255 in 100ms).

randomCount is only used when the LED light is in the random color blinking mode, which indicates the number of random colors blinking (the value of randomCount is 1~7).

The following is a description of the values that the part parameter may pass :

LED.PART_ALL All **LED** lights of the robot

LED. PART_WHEEL robot chassis of the **LED** light

LED.PART_LEFT_HAND LED light of the left wing of the robot

LED. PART_RIGHT_HAND LED light of the right wing of the robot

LED . PART_LEFT_HEAD LED light on the left side of the robot head

LED . PART_ RIGHT _HEAD robot head and right side **LED** lamp

The following is a description of the values that the mode parameter may pass :

LED. MODE_CLOSE turns off the **LED** light

LED. MODE_WHITE white **LED** light

LED. MODE_RED red **LED** light

LED. MODE_GREEN green **LED** light

LED . MODE_PINK pink **LED** light

LED. MODE_PURPLE purple **LED** light

LED. MODE_BLUE blue **LED** light

LED . MODE_YELLOW yellow **LED** light

LED. MODE_FLICKER_WHITE blinking white **LED** (**delayTime** can control the **LED** blinking interval)

LED. MODE_FLICKER_RED flashing red **LED** light

LED . MODE_FLICKER_GREEN flashing green **LED** lights

LED. MODE_FLICKER_PINK flashing of pink **LED** lights

LED. MODE_FLICKER_PURPLE flashing purple **LED** light

LED. MODE_FLICKER_BLUE blinking blue **LED** lights

LED. MODE_FLICKER_YELLOW flashing yellow **LED** lights

LED . MODE_FLICKER_RANDOM random color flashing **(randomCount** can control the number of colors of **LED** random flashing **)**

Sample code :

```
hardWare Manager. setLED ( new LED (LED. PART_ALL , LED. MODE_FLICKER_RANDOM , 10,3) ) ;
```

3.3.3 Touch event callback

Method signature :

```
public void setOnHareWareListener (HardWareListener hareWareListener)
```

Method description :

When the robot touch sensor is triggered when , the touch event callbacks occur.

Parameter Description:

Touch event achieved is TouchSensorListener the interface, in this interface, the need to implement the following methods:

void onTouch (int part);

Wherein the part in the range of 1 to 13 .

1	Chin right sensor	**2**	Chin left side sensor
3	Chest, the right side of the sensor	**4**	Chest left side sensor
5	Left head sensor	**6**	Right back sensor
7	Back left sensor	**8**	Back right sensor
9	Left hand sensor	**10**	Right hand sensor
11	Head top middle sensor	**12**	The head of the right front of the sensor
13	Overhead left front sensor		

Sample code :

```
hardWareManager.setOnHareWareListener (new TouchSensorListener () {
    @Override
    public void onTouch (int part) {
        if (part == 11 || part == 12 || part == 13) {
            touchTv.setText (" You touched the head " );
        }
    }
});
```

.3..3..4 sound source localization

Method signature :

public void setOnHareWareListener (HardWareListener hareWareListener)

Method description :

When the robot is awakened by voice , the sound source localization event will callback.

Parameter Description:

Sound source localization event achieved is VoiceLocateListener the interface, in this interface, the need to implement the following methods:

void voiceLocateResult (int angle);

angle indicates a sound source relative to the head in front of the offset angle , and is calculated in a clockwise direction.

Sample code :

```
hardWareManager.setOnHareWareListener (new VoiceLocateListener () {
    @Override
    public void voiceLocateResult (int angle) {
        // TODO obtains the sound source positioning angle for processing } });
```

.3..3..5 P the IR detector

Method signature :

public void setOnHareWareListener (HardWareListener hareWareListener)

Method description :

Robot 's front and behind , respectively, have a PIR module, when the robot PIR module detects that someone from the robot in front of or behind the pass or leave time, PIR detection event callbacks occur.

Parameter Description:

P the IR detection event implemented is PIRListener the interface, in this interface, the need to implement the following methods:

void onPIRCheckResult (boolean isChecked, int part);

When is Checked is true , it means that the PIR has detected someone passing, and when it is false , it means that it has detected that the person has left or before the robot. part parameter indicates an infrared sensor portion ,the value in the range of 1~2 . 1 represents the PIR on the front of the robot , and 2 represents the PIR on the back of the robot .

Sample code :

```
    hardWareManager.setOnHareWareListener (new PIRListener () {
     @Override
     public void onPIRCheckResult ( boolean isCheck, int part) {
         System. out.print ((part == 1? " front ": " behind ") + "PIR is triggered ");
     }
});
```

.3..3..6 infrared sensor callback (infrared ranging)

Method signature :

public void setOnHareWareListener (HardWareListener hareWareListener)

Method description :

Infrared sensor callback function currently returns the robot 18 is th measuring infrared emitters amount result of the distance of the obstacle. Robot infrared emitter position schematically in FIG as shown:

Parameter Description:

An infrared sensor associated functions implemented is InfrareListener the interface, in this interface, the need to implement the following methods:

void infrareDistance (int part, int distance);

infrareDistance is infrared distance result method , Part parameter indicates an infrared sensor portion, a specific value of the figure identifies the parts-one correspondence, in the range of 1 to 17, respectively represent 1 to 17 No. infrared ranging module. distance indicates the distance from the measurement module to the obstacle , in centimeters.

Sample code :

```
hardWareManager .setOnHareWareListener ( new InfrareListen er () {
    @Overr ide
    public void infrareDistance ( int part, int distance) {
        if (distance! = 0) { System.out.print ( " Part " + part + " The
distance is " + distance);          }     } });
```

.3..3..7 is turned on / off off black line filter

Method signature :

```
public OperationResult switchBlackLineFilter (boolean isOpen)
```

Method description :

Comparative ground presence wide black lines or time slot, infrared distance will affect the function of the robot can work normally , so as to stop the movement of the wheels of the robot. If you need to prevent the robot from being disturbed in this environment and continue to move, you need to turn on the black line filtering function. Please note : Enabling the black line filtering function will cause the robot's anti- fall function to fail. Please avoid enabling the black line filtering function in an environment where there is a risk of the robot falling .

.3..3..8 provided white light luminance

Method signature :

```
public OperationResult setWhiteLightLevel (int level)
```

Method description :

Set the brightness of the white light of the robot. The value of level ranges from 1 to 3, which respectively represents the brightness level of the white light is energy saving , soft and bright .

3.3.9 gyro-related functions callbacks

Method signature :

```
public void setOnHareWareListener (HardWareListener hareWareListener)
```

Method description :

This callback interfaces used to obtain the robot on the apparatus for measuring gyroscope deflection angle data.

Parameter Description:

Gyro -related functions to achieve is GyroscopeListener the interface, in this interface, the need to implement the following methods:

```
void gyroscopeData (float driftAngle, float elevat ionAngle, float rollAngle);
```

The above method is a callback of gyroscope measurement results, driftAngle represents yaw angle data, elevationAngle represents pitch angle data , and rollAngle represents roll angle data.

Sample code :

```
hardWareManager .setOnHareWareListener ( new GyroscopeListener () {
    @Override
    public void gyroscopeData ( float driftAngle, float elevationAngle,
float rollAngle) {        // TODO returns gyroscope measurement data } });
```

3.3.10 wall failure detection

Method signature:

public void setOnHareWareListener (HardWareListener hareWareListener)

Method description:

The robot's wheel chassis, wings, and some distance detectors in front of and behind the body. When these sensors detect obstacles nearby, they will report it.Wall barrier state .

Parameter Description:

The barrier detection event implements the ObstacleListener interface. In this interface, the following methods need to be implemented:

void onObsta cleStatus (boolean status);

Callback parameters status : true , obstacle avoidance false , environmental safety

Sample code:

```
hardWareManager . setOnHareWareListener ( new ObstacleListener () {
@Override public void onObstacleStatus ( boolean b) { if (b) { Log.w (
"info" , "onObstacleStatus: avoiding obstacles! " ); } else {
Log.i ( " info " , " onObstacleStatus: Barrier-free around. " );          }
    } });
```

3..4 head movement control

Access process:

1, application HeadMotionManager objects ,application codes as

```
HeadMotionManager headMotionManager = ( HeadMotionManager ) getUnitManager
(FuncConstant. H EA DMOTION_MANAGER );
```

2, use application of headMo tionManager the object , call the appropriate methods for control.

.3..4..1 relative angular movement

Method signature :

public OperationResult doRelativeAngleMotion (RelativeAngleHeadMotion relativeAngleHeadMotion)

Method description :

Controlling the robot head relative to the current position, the specified direction rotation movement angle section.

Parameter Description:

RelativeAngleHeadMotion This class has two parameters, Action and angle. a ction represents the robot's head movement mode, and angle represents the relative angle of the robot's head movement .

The following is a description of the values that the action parameter may pass :

RelativeAngleHeadMotion .ACTION_STOP stop motion

RelativeAngleHeadMotion .ACTION_UP upward movement

RelativeAngleHeadMotion . ACTION_DOWN downward movement

RelativeAngleHeadMotion . ACTION_LEFT left movement

RelativeAngleHeadMotion . ACTION_RIGHT right movement

RelativeAngleH eadMotion . ACTION_LEFTUP left on the side of sports

RelativeAngleHeadMotion . ACTION_RIGHTUP right on the square movement

RelativeAngleHeadMotion . ACTION_LEFTDOWN left under side movement

RelativeAngleHeadMotion . ACTION_RIGHTDOWN under the right side of sports

Sample code :

```
RelativeAngleHeadMotion relativeAngleHeadMotion = new RelativeAngleHeadMot ion (
        RelativeAngleHeadMotion.ACTION_RIGHT, 30
);
headMotionManager.doRelativeAngleMotion (relativeAngleHeadMotion);
```

3.4.2 absolute angular motion

Method signature :

public OperationResult doAbsoluteAngleMotion (AbsoluteAngleHeadMotion absoluteAngleHeadMotion)

Method description :

Controlling the robot head is rotated to a specified angle at which location. The horizontal direction, the absolute angle in the range of 0 to 180 degrees , from the left to the right angle values sequentially increase . The absolute angle in the vertical direction ranges from 7 to 30, and the angle value increases from the lower value .

Parameter Description:

AbsoluteAngleHeadMotion This class has two parameters, Action and angle. A ction represents the robot head movement pattern, angle represents the robot head movement absolutely angle.

The following is a description of the values that the action parameter may pass :

AbsoluteAngleHeadMotion . ACTION_VERTICAL vertical motion

AbsoluteAngleHeadMotion . ACTION_HORIZONTAL horizontal motion

Sample code :

```
AbsoluteAngleHeadMotion absoluteAngleHeadMotion = new AbsoluteAngleHeadMotion (
        AbsoluteAngleHeadMotion.ACTION_HORIZONTAL, 130
);
headMotionManager.doAbsoluteAngleMotion (absoluteAngleHeadMotion);
```

3.4.3 horizontal center lock

Method signature :

pub lic OperationResult dohorizontalCenterLockMotion ()

Method description :

Controlling the robot head is rotated to the horizontal 90-degree position , and the robot in the horizontal direction to lock the motor. (After locking , the robot's head cannot be turned horizontally) .

Sample code :

```
headMotionManager.dohorizontalCenterLockMotion ();
```

.3..4..4 Absolute angular positioning operation

Method signature :

public OperationResult doAbsoluteLocateMotion (LocateAbsolute AngleHeadMotion locateAbsoluteAngleHeadMotion)

Method description :

Controlling the robot head is rotated to a specified angle at which position , and the completion of the decision whether to lock the rear head motor operation. The horizontal direction, the absolute angle in the range of 0 to 180 degrees , from the left to the right angle values sequentially increase . The absolute angle in the vertical direction ranges from 7 to 30, and the angle value increases from the lower value . The difference between this method and the head absolute angle movement method is that this method can decide whether to lock the motor . This method does not need to specify the rotation mode , but directly specifies the absolute angle value in the horizontal and vertical directions .

Parameter Description:

LocateAbsoluteAngleHead Motion This class has San Ge parameters, Action , horizontalAngle

And verticalAngle . A ction represents the robot head lock mode, horizontalAngle represents motion robot head horizontal direction absolute angle , verticalAngle represents robot head movement perpendicular to the direction of the absolute angle .

The following is a description of the values that the action parameter may pass :

LocateAbsoluteAngleHeadMotion . ACTION_NO_LOCK Motor is not locked

LocateAbsoluteAngleHeadMotion . ACTION_HORIZONTAL_LOCK motor horizontal direction locking

LocateAbsoluteAngleHeadMotion . ACTION_VERTICAL_LOCK Motor vertical lock

LocateAbsoluteAngleHeadMotion . ACTION_BOTH_LOCK motor horizontal and vertical directions are locked

Sample code :

```
LocateAbsoluteAngleHeadMotion locateAbsoluteAngleHeadMotion = new
LocateAbsoluteAngleHeadMotion (
        LocateAbsoluteAngleHeadMotion.ACTION_BOTH_LOCK, 30,15
);
headMotionManager.doAbsoluteLocateMotion (locateAbsoluteAngleHeadMotion);
```

.3..4..5 relative angular positioning operation

Method signature :

public OperationResult doRelativeLocateMotion (LocateRelativeAngleHeadMotion locateRelativeAngleHeadMotion)

Method description :

Controlling the robot head relative to the current position of the rotational angle , and the completion of the rotation decision whether to lock the rear head motor.

Parameter description :

LocateRelativeAngleHeadMotion this class has Wu Ge parameters, Action , horizontalAngle , VerticalAngle , horizontalDirection, and verticalDirection . A ction represents the robot head lock mode, horizontalAngle represents motion robot head horizontal direction relative to the angle , verticalAngle represents robot head movement perpendicular to the direction of relative angle , horizontalDirection represents the horizontal movement of the robot head direction , verticalDirection represents robot head vertical movement direction.

The following is a description of the values that the action parameter may pass :

LocateRelativeAngleHeadMotion . ACTION_NO_LOCK Motor is not locked

LocateRelativeAngleHeadMotion . ACTION_HORIZONTAL_LOCK motor horizontal direction locking

LocateRelativeAngleHeadMotion . ACTION_VERTICAL_LOCK Motor vertical lock

LocateRelativeAngleHeadMotion . ACTION_BOTH_LOCK motor horizontal and vertical directions are locked

The following is a description of the values that the horizonDirection parameter may pass :

LocateRelativeAngleHeadMotion . DIRECTION_LEFT horizontal left motion

LocateRelativeAngleHeadMotion . DIRECTION_ RIGHT horizontally to the right movement

LocateRelativeAngleHeadMotion . DIRECTION_ the UP perpendicular to the motion

LocateRelativeAngleHeadMotion . DIRECTION_ DOWN perpendicular to the motion

Sample code :

```
LocateRelativeAngleHeadMotion locateRelativeAngleHeadMotion = new
LocateRelativeAngleHeadMotion (
        LocateRelativeAngleHeadMotion.ACTION_BOTH_LOCK, (byte) 30, (byte) 15,
        LocateRelativeAngleHeadMotion.DIRECTION_LEFT
        , LocateRelativeAngleHeadMotion.DIRECTION_UP
);
MotionRateativeAngleMotionManagerDirectiveMove
```

3.5 wings motion control

Access process:

1, application Wing the MotionManager objects ,application codes as

```
Wing MotionManager wing MotionManager = ( Wing MotionManager ) getUnitManager
(FuncConstant. WING MOTION_MANAGER );
```

2, use application of Wing the MotionManager the object , call the appropriate methods for control.

.3..5..1 no angular movement

Method signature :

public OperationResult doNoAngleMotion (NoAngle Wing Motion noAngle Wing Motion)

Method description :

Control robot wings do no angle movements, such as raising his hand, wings homing and so on .

Parameter Description:

NoAngle Wing Motion this class has three parameters, Action, Part and Speed. A ction represents the robot wings rotation model, Part showing the robot wings location (left wing and right wing) , Speed represents the robot wings moving speed , in the range of 1 to 10.

The following is a description of the values that the action parameter may pass :

NoAngle Wing Motion . ACTION_UP wings upward rotation

NoAngle Wing Motion . Action_ DOWN wings downward rotation

NoAngle Wing Motion . Action_ STOP wings stop turning

NoAngle Wing Motion . ACTION_RESET Wings are returned to the default position (that is, the wings are vertically downward)

The following is a description of the values that the part parameter may pass :

NoAngle Wing Motion . PART_LEFT left wing operations

NoAngle Wing Motion .PART_RIGHT right wing operation

NoAngle Wing Motion .PART_BOTH left wing operate with

Sample code :

```
NoAngle Wing Motion noAngle Wing Motion = new NoAngle Wing Motion (NoAngle Wing
Motion. PART_BOTH ,
         5, NoAngle Wing Motion. ACTION_UP );
wing MotionManager.doNoAngleMotion (noAngle Wing Motion);
```

.3..5.2relative angular movement

Method signature :

public OperationResult doRelativeAngleMotion (RelativeAngle Wing Motion relativeAngle Wing Motion)

Method description :

Controlling the robot wings relative to the current position, the specified direction rotation movement angle section.

Parameter Description:

RelativeAngle Wing Motion this class have Si Ge parameters, Action, Part, Speed and angle. A ction represents the robot wings rotation model, Part showing the robot wings location (left wing and right wing) , Speed represents the robot wings moving speed , in the range of .1~.8, angle represents the robot wings rotated relative angle range It is 0~270, and the angle gradually increases in the counterclockwise direction.

The following is a description of the values that the action parameter may pass :

RelativeAngle Wing Motion . ACTION_UP wings rotated upwardly

RelativeAngle Wing Motion . Action_ DOWN wings to the rotation

The following is a description of the values that the part parameter may pass :

RelativeAngle Wing Motion . PART_LEFT left wing operations

RelativeAngle Wing Motion .PART_RIGHT right wing operation

RelativeAngle Wing Motion .PART_ BOTH right and left wings operate with

Sample code :

```
RelativeAngle Wing Motion relativeAngle Wing Motion = new R elativeAngle Wing
Motion ( RelativeAngle Wing Motion.PART_LEFT , 5,

RelativeAngle Wing Motion.ACTION_UP,
        30
);
wing MotionManager.doRelativeAngleMotion (relativeAngle Wing Motion);
```

3.5.3 absolute angular motion

Method signature :

public OperationResult doAbsoluteAngleMotion (AbsoluteAngle Wing Motion absoluteAngle Wing Motion)

Method description :

Controls the robot wings to rotate to the position where the specified angle is located.

Parameter Description:

AbsoluteAngle Wing Motion this class have San Ge parameter, Speed , Part and angle . speed represents the robot wings moving speed , in the range of .1 ~. 8, Part represents the robot wings location (left wing and right wing) , angle represents the robot wings rotation of the absolute angle, in the range of 0 to 270 , in the reverse The angle in the hour hand gradually increases.

The following is a description of the values that the part parameter may pass :

AbsoluteAngle Wing Motion . PART_LEFT left wing operations

AbsoluteAngle Wing Motion .PART_RIGHT right wing operation

AbsoluteAngle Wing Motion .PART_ BOTH right and left wings operate with

Sample code :

```
AbsoluteAngle Wing Motion absoluteAngle Wing Motion = new AbsoluteAngle Wing Motion
(
        AbsoluteAngle Wing Motion.PART_LEFT, 5,0
);
wing MotionManager.doAbsoluteAngleMotion (absoluteAngle Wing Motion);
```

3.6 wheel motion control

Access process:

1, application WheelMotionManager objects ,application codes as

```
WheelMotionManager wheelMotionManager = ( WheelMotionManager ) getUnitManager
(FuncConstant. WHEELMOTION_MANAGER );
```

2, use application of wheelMotionManager the object , call the appropriate methods for control.

.3..6..1 no angular movement

Method signature :

public OperationResult doNoAngleMotion (NoAngleWheelMotion noAngleWheelMotion)

Method description :

Control robot wheels do no angular motion, such as forward, backward , left to go and so on .

Parameter Description:

NoAngleWheelMotion this class have San Ge parameter, Speed , Action and DURATION . speed represents a robot wheel movement velocity , in the range of .1 ~ 10 , A ction represents robot wheel motion mode, duration represents the robot wheel motion specified

time , in units of 100 MS , when the duration is 0 when representing the robot will always in motion, Until you receive a stop command.

The following is a description of the values that the action parameter may pass :

NoAngleWheelMotion . ACTION_FORWARD forward

NoAngleWheelMotion . ACTION_LEFT turns left in place

NoAngleWheelMotion . ACTION_RIGHT right standing around

NoAngleWheelMotion . ACTION_TURN_LEFT left turn

NoAngleWheelMotion . ACTION_TURN_RIGHT right turn

NoAngleWheelMotion . ACTION_STOP_TURN stop turning

NoAngleWheelMotion . ACTION_STOP_RUN stopped walking

Sample code :

```
NoAngleWheelMotion noAngleWheelMotion2 = new NoAngleWheelMotion (
        NoAngleWheelMotion.ACTION_FORWARD_RUN, 5,3000 );
wheelMotionManager.doNoAngleMotion (noAngleWheelMotion2);
```

.3..6.2 relative angular movement

Method signature :

public OperationResult doRelativeAngleMotion (RelativeAngleWheelMotion relativeAngleWheelMotion)

Method description :

Robot control wheel relative to the current position, the specified direction rotation movement angle section.

Parameter Description:

RelativeAngleWheelMotion this class have San Ge parameter, Speed , Action and angle . speed represents a robot wheel movement velocity , in the range of .1 ~ 10, A ction represents robot wheel motion mode, angle represents the robot wheel rotation relative angle .

The following is a description of the values that the action parameter may pass :

RelativeAngleWheelMotion . TURN_LEFT left turn

RelativeAngleWheelMotion . TURN_ RIGHT right turn

RelativeAngleWheelMotion . TURN_STOP stop rotation

Sample code :

```
RelativeAngleWheelMotion relativeAngleWheelMotion = new RelativeAngleWheelMotion (
        RelativeAngleWheelMotion.TURN_LEFT, 5,90
);
wheelMotionManager.doRelativeAngleMotion (relativeAngleWheelMotion);
```

3.6.3 from the movement

Method signature :

public OperationResult doDistanceMotion (DistanceWheelMotion distanceWheelMotion)

Method description :

Controls the robot to move a specified distance in the specified direction .

Parameter Description:

DistanceWheelMotion this class have San Ge parameter, Speed, Action and Distance. speed represents a robot wheel movement velocity, in the range of .1 ~ 10, A ction represents robot wheel motion mode, Distance represents the robot movement distance, in units of cm & lt.

The following is a description of the values that the action parameter may pass :

DistanceWheelMotion . ACTION_FORWARD_RUN forward

DistanceWheelMotion . ACTION_STOP_RUN stopped walking

Sample code :

```
DistanceWheelMotion distanceWheelMotion = new DistanceWheelMotion (
        DistanceWheelMotion. ACTION_FORWARD_RUN , 5,100
);
wheelMotionManager.doDistanceMotion (distanceWheelMotion);
```

.3..6..4 wheel motion state callback

Method description :

After the robot wheel moves, it will call back the wheel movement state

Parameter Description:

The state s of the callback is explained. When s = "0", it means that the motion is stopped. If it is not " 0", it is the motion state

Sample code :

```
wheelMotionManager.setWheelMotionListener (new
WheelMotionManager.WheelMotionListener () {
    @Override
    public void onWheelStatus (String s) {
        Log.i ("Cris", "onWheelStatus: s =" + s);
    }
});
```

3.7 system management

Access conditions : part of the function can only be made through the system signed app can access the following will require signing system can function access tagging.

Access process:

1, application SystemManager objects , application codes as

```
SystemManager systemManager = ( SystemManager ) getUnitManager (FuncConstant.
SYSTEM_MANAGER );
```

2, use application of systemManager the object , call the appropriate methods for control.

.3..7.1 acquisition device ID

Method signature :

public String getDeviceId () ;

Method description :

Get the device ID number of the current robot .

Sample code :

```
systemManager. getDeviceId ();
```

3.7.2 return screensaver animation interface

Method signature :

public OperationResult doHomeAction ()

Method description :

Let the robot display jump to the desktop screensaver animation interface.

Return value description:

OperationResult's ErrorCode may return ErrorCode . FAIL_APP_HAS_LOCKED error , which means that there is currently an app in the locked state, unable to return to the screen saver animation interface .

Sample code :

```
systemManager.doHomeAction ();
```

.3..7.3 expression control
Method signature :

public OperationResult showEmotion (EmotionsType emotion)

Method description :

The robot display the specified expression, this function only when the currently displayed page as the face when the interface will not take effect.

Parameter Description:

The following is a description of the values that the emotion parameter may pass :

EmotionsType . Arrogance Pride

EmotionsType . SURPRISE surprise

EmotionsType . WHISTLE whistle

EmotionsType . Laughter laugh

EmotionsType . GOODBYE goodbye

EmotionsType . SHY shy

EmotionsType . SWEAT Khan

EmotionsType . Snicker grin

EmotionsType . PICKNOSE pull the nose

EmotionsType . CRY cry

EmotionsType . ABUSE curse

EmotionsType . ANGRY angry

EmotionsType . KISS kiss

EmotionsType . SLEEP sleep

EmotionsType . SMILE Smile

EmotionsType . Grievance wronged

EmotionsType . QUESTION doubt

EmotionsType . FAINT halo

EmotionsType . PRIZE Chan

EmotionsType . The NORMAL default

Sample code :

```
systemManager. showEmotion ( EmotionsType. PRISE );
```

.3..7..4 Get Battery
Method signature :

```
public int getBatteryValue ()
```

Method description :

Used to get the battery level of the current robot .

Return value description:

The result field of the OperationResult returns the battery power. The power data is of type int.

.3..7..5 get battery status
Method signature :

```
public int getBatteryStatus ()
```

Method description :

Used to get the battery charging status of the current robot .

Return value description:

OperationResult the result field returns the robot charge state, result there are three results: **the SystemManager .STATUS_NORMAL** not the state of charge

The SystemManager .STATUS_CHARGE_PILE using a charging post robot charge in

The SystemManager . SATUS_CHARGE_LINE using power cord robot charge in

3.7.6 monitor home security alarm
Method signature :

```
public void setOnIDarlingListener (IDarlingListener iDarlingListener)
```

Method description :

If the security home turned up deployment strategy, when security home alarm event is triggered, the callback method will be called.

Parameter description :

Home security alarm callback implementation is IDarlingListener the interface, in this interface, the need to implement the following methods:

void onAlarm (int type);

type represents a home security alarm mode, when the type is 1 Shi , indicates that the current trigger the occlusion alarm , is 2 Shi , expressed triggered intrusion alarm , is 3 Shi , represented trigger a cross-border police .

3.7.7 to obtain a master version number
Method signature :

```
public String getMainServiceVersion ( )
```

Method description :

Get the master program version number of the current system .

.3..7..8 show / hide the system was suspended button

Method signature :

public OperationResult switchFloatBar (boolean isShow, String className)

Method description :

Show / hide the floating button of the current system . This method can only Activity in the call , Service call will not take effect.

Parameter Description:

isShow whether to show the floating button of the system

className The full name of the class of the current Activity , which can be obtained by the getClass (). getName () method .

.3..7..9 show / hide the system was suspended button (to avoid APP within Activity jump appears briefly)

Method signature :

public OperationResult switchFloatBarInApplication (boolean isShow)

Method description :

Show / hide the floating button of the current system .

Parameter Description:

isShow whether to show the floating button of the system

3.7.10 queries robot type

Method signature :

p ublic int getRobotType ();

Method description :

Get Robot Type

Method return description:

SystemManager. *ROBOT_TYPE_B* ; (B model, elf)
SystemManager. *ROBOT_TYPE_D* ; (D model, King Kong)
SystemManager. *ROBOT_TYPE_DESKTOP* ; (desktop robot, mini elf)

3.8 Multimedia Management

Access process:

1, application HDCameraManager objects ,application codes as

```
HDCameraManager hdCa meraManager = (HDCameraManager) getUnitManager
(FuncConstant.HDCAMERA_MANAGER);
```

2, use application of HDCA meraManager the object , call the appropriate methods for control.

.3..8.1 obtain audio and video streams callback

Method signature :

public void setMediaListen er (MediaListener mediaListener)

Method description :

Obtain audio and video streaming data through the robot's HD camera and microphone.

Note :

1, the microphone HD camera is not going to support echo cancellation, so when the local recording while playing there will be a sharp noise, use the audio need to pay attention.

Parameter description :

Obtain audio and video stream callback implementation is MediaStreamListener the interface, in this interface, the need to implement the following methods:

void getVideoStream (byte [] data);
void getAudioStream (byte [] data);

The above two methods correspond to the video stream data callback and audio stream data callback methods. For the processing of the acquired audio and video streams, please refer to the code provided in the Demo.

.3..8.2 open media stream

Method signature :

```
public OperationResult openStream (StreamOption streamOption)
```

Method description :

The method of opening the media stream , only after calling this method, can get the callback data of the audio and video stream.

Parameter Description:

StreamOption has the following parameters:

type indicates a video using the decoding mode may be a range of :

StreamOption . TYPE_HARDWARE_DECORD hard decoding (default value)

StreamOption . TYPE_SOFTWARE_DECORD soft decoding

isJustIframe indicates whether only return Iframe (please note that this is an English letter I, instead of the number 1) of the data ,the default value to false.

c hannel indicates the bitstream format of the returned video . The possible value ranges are:

StreamOption . MAIN_STREAM main stream , the returned video resolution is **1280 x 720** (default value)

StreamOption . SUB_STREAM sub stream , the returned video resolution is **640 * 480**

The example code is as follows:

```
StreamOption = StreamOption new new StreamOption () ;
streamOption.setChannel (StreamOption. MAIN_STREAM ) ;
streamOption.setDecodType (StreamOption. HARDWARE_DECODE ) ;
streamOption.setJustIframe ( to false ) ;
OperationResult OperationResult = hdCameraManager .openStream (streamOption) ;
int . Result = Integer valueOf ( operationResult.getResult ()) ;
if (result! = -1 ) { handleList .add (result) ; }
```

Opening a video stream now allows a maximum of two video streams to be pulled. The hdCameraManager.openStream (streamOption) method will return an OperationResult object when it is opened . According to the getResult () method of operationResult , you will get the instructions for opening the stream. Strings greater than 0 need to be converted to int. The purpose of the guide is to close the stream and need to pass in the change guide.

.3..8.3 Close media stream

Method signature :

public OperationResult closeStream (int handle)

Method description :

Method to close the media stream . If you have called the openStream method , you must remember that you must call this method to close the media stream when the interface is destroyed , otherwise it will cause unexpected risks such as memory leaks.

Example code

hdCameraManager.closeStream (handle)) , where the handle represents the guidelines returned when opening the stream

3.8.4 get recognition callback

Method signature :

public void setMediaListen er (MediaListener mediaListener)

Method description :

When the robot recognition to located when the front of the robot's face triggered this callback method. If the current user has entered information in the family member App, if the robot recognizes the corresponding user, it will return the information entered by the user together. Please note , the family members of the App entering the user does not necessarily mean that the robot can identify the user, recognition accuracy at the same time will be the current environmental light , face angle and other factors , which in no network environment without Will call back .

Parameter description :

Gets Face Recognition callback implementation is FaceRecognizeListener the interface, in this interface, the need to implement the following methods:

void recognizeResult (List < FaceRecognizeBean > faceRecognizeBean) ;

FaceRecognizeBean contains identification to the face in the imaging screen of location and personal information entered by the user in the family members (if any) , which contains the following parameters:

Private int w (face width of image recognition, the current default is **1280**)
Private String Birthday (identification of the user birthday information, user information has been entered in the required family members)
Private Double bottom (in the face of the screen in the bottom position occupied screen percentage ratio)
Private String gender (identified user gender information, user information needs of family members have been entered)
Private Double left (face in the screen, the left-most position of the screen occupied by the percentage ratio)
Private String the QLINK (pre left field)
Private Double right (in the face of the screen in the right side of the screen position occupied by the percentage ratio)
Private String the user (identified by user name information, user information has been entered in the required family members)
Private int H (face recognition image height , the current default is **720**)
Private Double top (face in the screen most top position occupied by the screen percentage ratio)

Sample code :

```
   hdCamera Manager .setMediaListener (new FaceRecognizeListener () {
    @Override
    public void recognizeResult ( List <FaceRecognizeBean>
faceRecognizeBean ) {
Log.i ("info", " Get face recognition data ");
    }
});
```

.3..8.5 from the capture video stream

Method signature :

public Bitmap getVideoImage ()

Method description :

You can call this method if you need to grab an image from the robot 's video stream . This method and 3.9.4 methods combine them together using , for gripping face.

Sample code :

```
hdCamera Manager.setMediaListener (new FaceRecognizeListener () {
 @Override
 public void recognizeResult (FaceRecognizeBean faceRecognizeBean ) { Bitmap
bitmap = mediaManager . getVideoImage () ; if (bitmap! = null) { Log.i ("info", "
got the human face Identify images "); }     } });
```

3.9 projector control

Access process:

1, application ProjectorManager objects ,application codes as

```
PojectorManager projectorManager = (ProjectorManager) getUnitManager
(FuncConstant.PROJECTOR_MANAGER);
```

2, use application of ProjectorManager the object , call the appropriate methods for control.

.3..9..1 Switch Projector

Method signature :

public OperationResult switchProjector (boolean isOpen)

Method description :

For controlling opening and closing of the projector , note that each opening and closing of the projector of the operation are at least the interval 12 is S or more , otherwise the instruction will not be executed .

Parameter Description:

isOpen true means the projector is turned on and vice versa

Sample code:

```
p ro jectorMananger.switchProjector (true);
```

.3..9.2 disposed projector mirror

Method signature :

public OperationResult setMirror (int value)

Method description :

Used to set the projector image .

Parameter Description:

Value ranges from 0 to 3, and has the following mirroring modes:

ProjectorManager. MIRROR_CLOSE 0 no flip, default mode

ProjectorManager. MIRROR_LR 1 Flip left and right

ProjectorManager. MIRROR_UD 2 Flip up and down

With your projector. MIRROR_ALL 3 up and down left and right are reversed

Sample code:

```
p ro jectorMananger. setMirror ( ProjectorManager.MIRROR_ALL );
```

.3..9.3 provided projector trapezoidal horizontal correction value

Method signature :

public OperationResult setTrapezoidH (int value)

Method description :

Used to set the projector keystone level correction value .

Parameter Description:

The valid value range is -30 ~ 30

Sample code:

```
p ro jectorMananger. setTrapezoidH ( 10 ) ;
```

.3..9.4 disposed projector trapezoidal vertical correction values

Method signature :

public OperationResult setTrapezoidV (int value)

Method description :

Used to set the keystone vertical correction value of the projector .

Parameter Description:

The valid value range is -20 ~ 30

Sample code:

```
P RO . jectorMananger setTrapezoid V ( 10 ) ;
```

.3..9.5 disposed projector contrast

Method signature :

public OperationResult setContrast (int value)

Method description :

Used to set the contrast of the projector screen .

Parameter Description:

The valid value range is -15 ~ 15

Sample code:

```
p ro jectorMananger. setContrast ( 10 ) ;
```

.3..9.6 disposed projector brightness

Method signature :

public OperationResult setBright (int value)

Method description :

Used to set the brightness of the projector screen .

Parameter Description:

The valid value range is -31 ~ 31

Sample code:

```
p ro jectorMananger. setBright ( 10 ) ;
```

.3..9.7 provided a projector color value

Method signature :

```
public OperationResult setColor (int value)
```

Method description :

For the projector screen the color value set .

Parameter Description:

The valid value range is -15 ~ 15

Sample code:

```
p ro jectorMananger. setColor ( 10 ) ;
```

.3..9.8 provided a projector screen saturation

Method signature :

```
public OperationResult setSaturation (int value)
```

Method description :

For a projector screen of saturation is set .

Parameter Description:

The valid value range is -15 ~ 15

Sample code:

```
p ro jectorMananger. setSaturation ( 10 ) ;
```

.3..9.9 provided a projector screen sharpness

Method signature :

```
public OperationResult setAcuity (int value)
```

Method description :

Used to set the sharpness of the projector screen .

Parameter Description:

The valid value range is 0 ~ 6

Sample code:

```
p ro jectorMananger. setAcuity ( 4 ) ;
```

.3..9.10 projector expert mode optical axis adjustment

Method signature :

```
public OperationResult setExpertAxis (int value)
```

Method description :

Used to set the optical axis of the projector expert mode .

Parameter Description:

Value 1 Open the setting 2 Increase the value 3 Decrease the value 4 Exit without saving 5 Save and exit the setting

Sample code:

```
p ro jectorMananger. setExpertAxis ( 1 ) ;
```

.3..9.1 1 projector expert mode phase adjustment

Method signature :

```
public OperationResult setExpertPhase (int value)
```

Method description :

Used to set the phase of the projector expert mode .

Parameter Description:

Value 1 Open the setting 2 Increase the value 3 Decrease the value 4 Exit without saving 5 Save and exit the setting

Sample code:

```
p ro jectorMananger. setExpertPhase ( 1 ) ;
```

.3..9.1 2 setting up the projector mode

Method signature :

```
public OperationResult setMode (int mode)
```

Method description :

Used to set the projector 's projection mode .

Parameter Description:

Value ProjectorManager.MODE_WALL 1 wall projection mode

ProjectorManager. MODE_CEILING 2 ceiling projection mode

Sample code:

```
P RO jectorMananger. the setMode ( ProjectorManager.MODE_CEILING ) ;
```

3..9.1 3 queries the projector data and status

Method signature :

```
public OperationResult queryConfig ( String configName )
```

Method description :

It is used to query whether the current projector is on or off , and related values such as sharpness, contrast, and color settings.

configName may have the following values:

With your projector. CONFIG_SWITCH query switching state **with your projector. CONFIG_TRAPEZOIDH** queries the horizontal keystone value **with your projector. CONFIG_TRAPEZOIDV** query vertical keystone value **with your projector. CONFIG_CONTRAST** query contrast value **with your projector. CONFIG_BRIGHT** query luminance value **with your projector. CONFIG_COLOR** query chroma value **with your projector. CONFIG_SATURATION** query saturation value **with your projector. CONFIG_ACUITY** query Sharpness value **ProjectorManager.**

CONFIG_MODE query mode **ProjectorManager. CONFIG_MIRROR** query image

Sample code:

```
= OperationResult OperationResult P RO jectorMananger. QueryConfig (
ProjectorManager.CONFIG_SWITCH );

i f ( operationResult.getResult (). equals ("1")) {

              Log.e ("info", "The projector is turned on ")

}
```

.3..9.14 projector interface callbacks

Method signature :

public void setOnProjectorListener (OnProjectorListener projectorListener)

Method description :

Projector-related listener interfaces for monitoring the projector expert mode adjustment error occurred when the situation .

Sample code:

```
projectorManager.setOnProjectorListener ( new ProjectorMana ger.ProjectorListener
     () {

        @Override

        public void expertModeError (ProjectorExpertMode projectorExpertMode) {

        // TODO projector expert mode adjustment error callback

        }

});
```

3.10 Modular movement function control

Access process:

1, application Modul is Ar the MotionManager objects ,application codes as

Modul ar MotionManager modul ar MotionManager = (Modul ar MotionManager) getUnitManager (FuncConstant. MODUL AR MOTION_MANAGER);

2, use application of Modul Ar the MotionManager the object , call the appropriate methods for control.

3.10.1 open / closed walk freely

Method signature :

public OperationResult switchWander (boolean isOpen)

Method description :

Controls the robot's free-running function on and off.

Return value description:

OperationResult the ErrorCode may be there the following return values :

ErrorCode.FAIL_MOTION_LOCKED current robots are performing other with the relevant sports function , leading to motion control is locked, unable to respond to commands .

ErrorCode.FAIL_NO_PERMISSION The current user group does not have permission to operate this function

ErrorCode.FAIL_IS_CHARGE The robot is currently charging, and the corresponding function cannot be turned on .

.3.10.2 is turned on / off automatically charged

Method signature :

```
public OperationResult switchCharge (boolean isOpen)
```

Method description :

Control the robot's automatic charging function on and off.

Return value description:

OperationResult the ErrorCode may be there the following return values :

ErrorCode.FAIL_MOTION_LOCKED current robots are performing other with the relevant sports function , leading to motion control is locked, unable to respond to commands .

ErrorCode.FAIL_NO_PERMISSION The current user group does not have permission to operate this function

ErrorCode.FAIL_IS_CHARGE The robot is currently charging, and the corresponding function cannot be turned on .

ErrorCode.FAIL_NO_CHARGE_PILE The robot did not find the charging pile. Please confirm whether the charging pile is plugged in or has been added to the smart circle.

.3.10..3 acquires walk freely switch state

Method signature :

```
public OperationResult getWanderStatus ()
```

Method description :

Determine whether the robot free walking function is turned on .

Return value description:

The getResult () method of OperationResult will return the query result. If the returned value is "1", it means that the loitering switch is turned on, and "0" means it is turned off.

Sample code :

```
OperationResult operationReusult = modul ar MotionManager .
GetWanderStatus ();
i f ( operationResult.getResult (). equals ( " 1 " )) {
// TODO loitering function is turned on
}
```

.3.10..4 acquires automatic charging switch state

Method signature :

```
public OperationResult getAutoChargeStatus ()
```

Method description :

Determine whether the robot's automatic charging function is turned on .

Return value description:

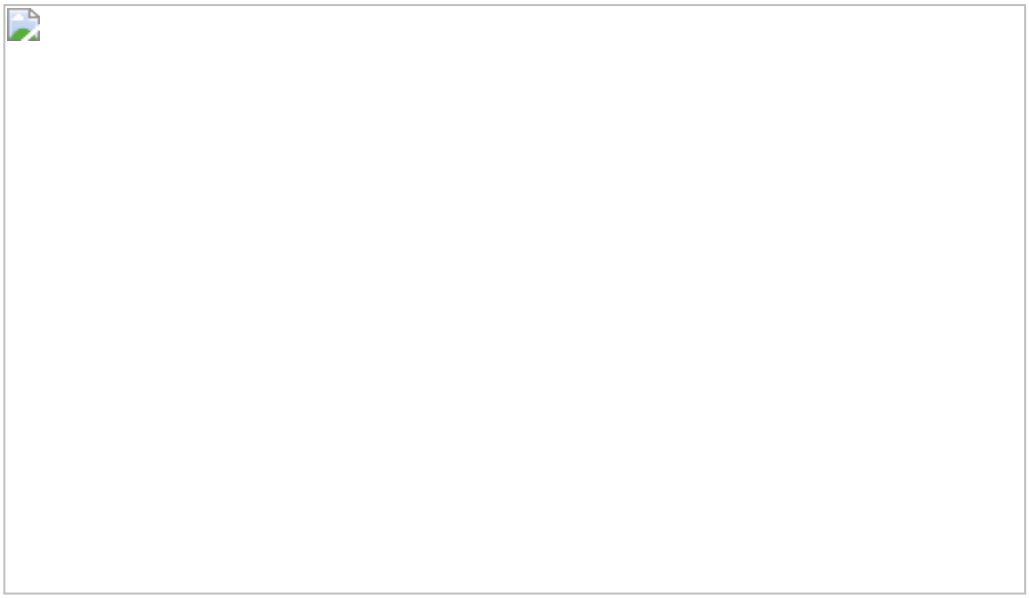
The getResult () method of OperationResult will return the query result. If the returned value is "1", it means that the automatic charging switch is turned on, and "0" means that it is turned off.

Sample code :

```
OperationResult operationReusult = modul ar MotionManager .
GetAutoChargeStatus ();
i f ( operationResult.getResult (). equals ( " 1 " )) {
// TODO automatic charging function is turned on
}
}
```

3.11 preprocessing instructions

Pre- processing instruction refers Activity or Service component when establishing a connection with the master, then immediately execute instructions in force. Preprocessing instructions < Meta-Data > tag form disposed in the AndroidManifest .xml file , where arranged in < file application > , < Activity > and < -Service > tag next, each having a different scope , the following are the AndroidManifest .xml file in the preprocessing directives schematic :



The code snippet shown by the red arrow is the preprocessing instruction.

Arranged in < file application > tags in the preprocessing directives action domain current application at all the Activity and -Service , which is a global property , corresponding to each one activity or service are configured with the same pre-processing instructions ; and disposed < activity > tag preprocessing directives at will the current activity is visible in effect, the current activity of onStop method fails when invoked; arranged in < -Service > tag pre-processing of the next instruction will Service is effective when created, the Service of Invalid when the onDestroy method is called.

<application> the preprocessing directives will be < Activity > or < -Service > the same is covered preprocessing directives. If you need to < the Application > or < Service > using the following preprocessor directives, use caution, because the Service can the ability to run in the background, once configured preprocessor directive would have been due to the Service of survival and entry into force, it will have robot itself is Interaction logic and interactions with other third-party applications will have an impact. Therefore , if it is not necessary, avoid using preprocessing instructions under < application > or < service > .

Note : Due to the working mechanism of preprocessing directives to limitations in research and development system for the next app reload or unload behavior may occur robot due to the influence of preprocessing directives appear to work abnormal of the phenomenon , the phenomenon is only the app reload It is related to the uninstall behavior and will not affect the normal use of the app after it is put on the shelf .

.3..11..1 recording switch

Method signature :

```
<meta-data android: name = "CONFIG_RECORD" android: value = " true " />
```

Method description :

If the robot needs to use the recording function, this command must be configured . If this command takes effect successfully, the robot ear will light blue. Be sure to follow the configuration described above, otherwise it will lead to a master reboot, App with the master of the inter- connection is disconnected.

.3..11.2 recognition mode

Method signature :

```
<meta-data android: name = "RECOGNIZE_MODE" android: value = "1" />
```

Method description :

Set the mode of robot speech recognition. Currently, only truncation mode is supported . For details , please refer to the description in section 3.2.6.

3.11.3 speech patterns

Method signature :

```
<meta-data android: name = "SPEECH_MODE" android: value = "1" />
```

Method description :

Robot voice recognition built-in two modes: First, when the robot is awakened, if once there exceeds 10 S clock is not detected people speak, will automatically enter the sleep state , this mode is the default mode used by the robot ; two , the robot is awakened If, after a short time within (about 2 to .3 seconds or so) does not detect people speak , will immediately enter the sleep state, when the robot to recognize the user utterance to say, the same will enter the sleep state , in this mode, Each time the user completes a dialogue with the robot , they need to wake up the robot again to continue communicating with the robot.

When android: value = "1" when the robot will enter the above-mentioned mode II.

.3..11..4 touch response switch

Method signature :

```
<meta-data android: name = " FORBID_TOUCH " android: value = "true " />
```

Method description :

Configuration of this article after preprocessing directives, the user will still be able to listen to touch events, but the robot will not make any response to touch events, including the default behavior wake touch robots .

.3..11..5 the PIR responsive switch

Method signature :

```
<meta-data android: name = " FORBID_PIR " android: value = "true " />
```

Method description :

Configuration of this article after preprocessing directives, the user will still be able to listen to P IR is triggered events, but the robot will not be on the PIR triggering event in accordance with the default policy to execute (the default

strategy refers to listening to the robot back PIR is triggered , The robot will rotate 180 degrees) .

.3..11..6 speech wakeup response switch

Method signature :

```
<meta-data android: name = " FORBID_WAKE_RESPONSE " android: value = "true " />
```

Method description :

Configuration of this article after preprocessing instructions, the robot voice wake-up, in addition to ear green lights outside , will not make any default response behavior .

.3..11..7 misrecognized patterns

Method signature :

```
<meta-data android: name = " MISTAKENLY_IDENTIFIED " android: value = "true " />
```

Method description :

The master will filter the recognition of a word by default. After setting the preprocessing instruction, the master will not filter the recognition of a word, such as "Yes", "No", etc. This mode is best used in a quiet environment, and noisy environments will always be misidentified.

3.11.8 (will be killed when the background music) anti manslaughter mode

Method signature :

```
<meta-data android: name = " forbid_stop_music " android: value = "true " />
```

Method description :

In previous versions, the app that played music in the background was killed by default . For example, if the current page is playing music, and then jumps to another page, and the music is still playing, then this app will be killed. Now add a pre-processing mode. If you configure this mode, Will not be killed again, but the APP has to control the life cycle of the playback itself, so as not to affect the recognition effect

3.11.9 prohibits semantic request mode (only available in foreign editions)

Method signature :

```
<meta-data android: name = " forbid_request_semantics " android: value = " 1 " />
```

Method description :

The user configures the meta-data on the Activity corresponding to AndroidManifest . The main service will read this configuration. If it is set to 1, the main service does not make network requests, and the customer can make its own semantic request based on the text recognition returned by the main service.

3.12 Command words and semantics

3.12.1 custom global command word

Users can through certain words or phrases , the robot be controlled, these words or phrases, which we call the command word. For example , we can through the " lights " command word to control the robot to open the head of white light through the "play movie "word command controls the robot to open movie app and play movies. The command word is global, regardless of the system in any interface, always be able to take effect .

The SDK supports developers to customize global command words for invoking the Activity or Service specified by the developer . In use a custom global command word when , developers need to understand the following:

1, the global command support offline word recognition, word order can take effect in a non-networked state.

2, each app support custom 10 Article command words, the system supports 100 Article within custom global command word . Command words that exceed this limit are automatically ignored.

3, the command word is parsing order is uncertain, which means that if your app, and other developers who app when defining the same command word, it is likely that your command word will not take effect .

4, global command word will be pre-loaded when the system first starts, developers who if the development time for the global command word made changes need to restart the robot or restart the main service program to take effect.

The development process of custom global command words is as follows :

1> Add the following configuration in the AndroidManifest.xml file:

```
< Provider Android : Authorities = " < you are A PP package name > .gr Ammar"
Android : name = "COM. Sanbot .opensdk.utils.GrammarProvider" Android : exported
= "to true" />
```

Note that the content in the above brackets is replaced with the package name of the corresponding app.

2> In A ssets directory new under the GRA MMAR file , and in the new grammar file is created under a folder named globalgrammar.xml of files.

3> in globalgrammar.xml in preparation global command word of grammar .

```
< Grammars >
    < group >
        < command >
            < item > Play video </ item >
            < item > Open video </ item >
        </ command >
        < type > activity </ type >
        < class > com. Sanbot .librarydemo.VideoActivity < / class >
    </ group >
    < group >
        < command >
            < item > Start service </ item >
        </ command >
        < type > service </ type >
        < class > com. sanbot .librarydemo.MainService </ class >
    </ group >
</ Grammars >
```

Above it is a global command word syntax example, when the user says the robot " Play Video "or "open video "when , the robot will automatically open the COM. Sanbot .librarydemo.VideoActivity this page . Next, the XML tags in the example are described in detail.

< Grammars > --- the root tag of the grammar .

< Group > ----- <Group> in the definition of a set of command words and word order to start Activity or Service behavior definition.

< Command > ---- <Command> in contains a set having a command word of the same behavior.

< Item > ---- each <item> in contains a Tiao command word.

< type > ——— The object started when the command word is triggered . The optional values are activity and service .

< class > ——— The class name of the class that needs to be started . Note that you must fill in the complete class name information.

4> For the activity or service that needs to be started , add the following configuration in the androidmanifest.xml file :

a ndroid : exported = ”true” .

Is a global command word will be triggered by Intent passed into ACT ivity or Service , as developers need to get the word trigger command, you can refer to the following code

```
public class TestActivity extends BindBaseActivity { @Override public void onCreate (Bundle savedInstanceState) { super .onCreate (savedInstanceState);         String data = getIntent (). getStringExtra ( "content" );     } ..... }
```

.3.12 is.2 custom semantics

Identifying callback interface utterance (refer to 3.2.6 section) , the interface returned content in addition to said user, then the robot outside , but also to return the robot to recognize semantic parsing results. But this semantic parsing of support is imperfect in , often can not meet the developer for the needs of a variety of semantics specific scene. Thus , SDK provides developers a custom semantic method .

In use custom semantics time , developers need to understand the following:

1, since the semantics defined generally only in a networked using a state.

2, each app supports custom 20 Tiao semantics , over the limit on the number of semantics will be automatically ignored.

3, custom semantics only when the user of the app while in the foreground status to take effect . Therefore, in the Service in custom made listening semantics are unreliable behavior .

4, custom semantics when the system first starts will be pre-loaded, developers who if the development time for custom semantic made changes need to restart the robot or restart the main service program to take effect.

The development process of custom global command words is as follows :

1 > Add the following configuration in the AndroidManifest.xml file:

```
< Provider Android : Authorities = " < your the app package name > .gr Ammar" Android : name = "COM. Sanbot .opensdk.utils.GrammarProvider" Android : exported = "to true" />
```

Note that the content in the above brackets is replaced with the package name of the corresponding app .

2 > In A ssets directory new under the GRA MMAR file , and in the new grammar file is created under a folder named local grammar.xml the file.

3 > at the local Grammar .xml in preparation global command word syntax .

```
< Grammars >
    < group >
        < command >
            < item > Play video </ item >
            < item > Open video </ item >
        </ command >
        < topic > qh_video </ topic >
        < action > start </ action >
    </ group >
    < group >
        < command >
            < item > pause video </ item >
        </ command >
        < topic > qh_video </ topic >
        < action > pause </ ac t ion>
    </ group >
</ Grammars >
```

The above is a global command Words of example, when the user says the robot " Play Video "or "open video "when , the robot will automatically open the COM. Sanbot .librarydemo.VideoActivity this page . Next, the XML tags in the example are described in detail.

< Grammars > --- the root tag of the grammar .

< group > ——— <group> defines a set of semantics .

< Command > ---- <Command> in comprising a group having the same to PIC and the action of the semantics .

< Item > ---- each <item> in contains a bar phrases, when the robot recognizes this phrase, it would put it in accordance with the user-defined semantic rules for processing .

< topic > ——— defines the topic of semantics . In order to avoid mixing with the built-in semantics of the system, the suggested naming method is " company (individual) abbreviation _ topic name " . For details, please refer to ″ Sambo Open Semantic Explanation Document″ .

< action > ——— Defines semantic action . For details, please refer to ″ Sambo Open Semantic Explanation Document″ .

3.13 Intelligent peripheral control

Access process:

1, application ZigbeeManager objects ,application codes as

```
ZigbeeManager zigbee Manager = (ZigbeeManager) getUnitManager
(FuncConstant.ZIGBEE_MANAGER);
```

2, use application of the ZigBee Manager objects , calling the appropriate method for controlling.

.3.13 is..1 acquires whitelist

Method signature :

```
public OperationResult getWhiteList ()
```

Method description :

For obtaining Zigbee whitelist list . The specific result will be returned in the callback method .

3.13.2 judge Zi gbee Are Ready

Method signature :

```
public OperationResult getNotifyReadyStatus ()
```

Method description :

Queries Zi gbee whether has been prepared ready , when zigbee ready to do after the other zigbee operations.

Return value description:

OperationResult of getResult () method will return query results, if the return value of "1" , represents Zigbee been successful initialization , "0" represents the Z-igbee has not been initialized successfully.

Sample code:

```
OperationResult operationResult = zigbeeManager . GetNotifyReadyStatus
();
if (operationResult.getResult (). equals ( "1" )) {
                // TODO NotifyReady
}
```

3.13.3 to obtain the list of devices
Method signature :

```
public OperationResult getZigbeeList ()
```

Method description :

Used to get a list of Zigbee devices . The specific result will be returned in the callback method .

.3.13 is..4 Add whitelist
Method signature :

```
public OperationResult addWhiteList (String command)
```

Method description :

For the Zigbee device is added to the whitelist .

3.13.5 delete whitelist
Method signature :

```
public OperationResult deleteWhiteList (String command)
```

Method description :

Used to remove Zigbee devices from the white list .

.3.13 is..6 Enable whitelist
Method signature :

```
public OperationResult switchWhtieList (boolean isOpen)
```

Method description :

Used to enable or disable the whitelist function .

Sample code:

```
zigbeeManager . switchWhtieList ( t rue);
```

.3.13 is..7 is provided apparatus allows time to join
Method signature :

```
public OperationResult setAllowJoinTime (int time)
```

Method description :

Set the time allowed for the device to join the gateway. When it is set to 0 , it means that the device is allowed to join the gateway at any time .

Parameter Description:

time The allowed time, in seconds.

Sample code:

```
zigbeeManager . setAllowJoinTime (200);
```

3.13.8 remove the device
Method signature :

```
public OperationResult deleteDevice (String command)
```

Method description :

Remove the device from the gateway.

.3.13 is..9 Empty whitelist
Method signature :

```
public OperationResult clearWhiteList ()
```

Method description :

Will the Z-igbee whitelist list empty.

.3.13 is.10 transmits Byte instruction
Method signature :

```
public OperationResult sendByteCommand (byte [] data)
```

Method description :

Direct transmission byte array type data to a Zigbee module, generally used for private communications device.

.3.13 is..11 transmits String command
Method signature :

```
public OperationResult sendCommand (String command)
```

Method description :

Directly send String type instructions to Zigbee module .

.3.13 is.12 is the Z igbee interfaces callback
Method signature :

```
public void setZigbeeListener ( ZigbeeListener zigbeeListener)
```

Method description :

Z igbee-related monitoring interface is used to obtain white list data, monitor Z igbee data update and status change.

Sample code:

```
zigbeeManager .setZigbeeListener ( new ZigbeeManager. ZigbeeListener () {
   @Override
    public void notifyWhiteList ( String whiteListData) {
        // Get whitelist data }     @Override     public void
notifyStatusChange ( String info) {      // Device type data will be returned
here }     @Override     public void notifyInfo ( String info) {
// Device list data is returned here
```

```
        }
});
```

3.1 4 Face Tracking and Active Greeting

Access process:

1, application FaceTrack Manager objects ,application codes as

```
FaceTrack Manager faceTrack Manager = ( FaceTrack Manager ) getUnitManager
(FuncConstant. FACETRACK_MANAGER );
```

2, use application of faceTrack Manager objects , calling the appropriate method for controlling.

.3..1.4..1 face tracking switch

Method signature :

```
public void switchFaceTrack ( boolean isOn )
```

Method description :

Directly send boolean-type commands to **FaceTrack Manager** . In the non-face interface, if you need to enable the face tracking function, you can use this method to open it. Similarly, if you need to turn off the face tracking function, you can pass in false .

Sample code:

```
faceTrackManager.switchFaceTrack ( true ) ;
```

3.1 4.2 face tracking status callback

Method signature :

```
public void setFaceTrackListener
( new FaceTrackManager.FaceTrackListener () { } )
```

Method description :

Face tracking status callback, it will only be called back if the face tracking switch is turned on in the current page

Sample code:

```
faceTrack Manager .setFaceTrackListener ( new FaceTrackManager.FaceTrackListener () {
@Override public void faceTrackStart () {                Log. i ( TAG ,
"faceTrackStart:" ) ; } @Override public void faceTrackComplete () {
Log. i ( TAG , "faceTrackComplete: " ) ; } @Override public void faceTrackFail () {
              Log. I ( TAG , " faceTrackFail: " ) ; } @Override public void
faceTrackLost () {                Log. I ( TAG , " faceTrackLost: " ) ; } @Override
public void startFindFace () {                Log. i ( TAG , "startFindFace:" ) ; } })
;
```

1, which startFindFace () method in the person standing in front of the robot triggers ir, is not detected when the person calls back, when the callback method, indicating that the robot was trying to find a human face by moving.

2, faceTrackStart () method explained that it had detected a face, a human face and begins tracking, a process which only one callback

.3, faceTrackComplete () method has been described face tracking and face in the middle of the screen, the user can make some face recognition logic operations, it is noted that this method may be triggered multiple times, it is possible to trace back the user moves Triggered again after the middle of the face.

4, faceTrackLost () method Description 10s no information in the face, the face has been lost, just the method only triggered faceTrackStart method will trigger after

5, faceTrack Fail () method only startFindFace after triggering method, 10s in the face is not looking to trigger, if 10s in the face to find this method does not trigger

.3..1.4..3 active switch hello

Method signature :

public void switchSayHello (boolean isOn)

Method description :

Directly send boolean-type instructions to **FaceTrack Manager** . If you need to use the active greeting function, please turn on the face tracking function first, because the active greeting function will only be triggered when the face is tracked until the face is in the middle of the screen.

Sample code:

```
faceTrackManager.switchSayHello ( true ) ;
```

true means open; false means close

.3..1.4..4 active greeting status callback

Method signature :

public void setUserNameDetectListener (new FaceTrackManager.UserNameDetectListener () { })

Method description :

The system will save the information face 5s, such as 5s in the face information is not automatically clear the face information stored, when the robot when in a non-speaking, non-wake-up, non-ban wheat state, will face the callback information.

Sample code:

```
faceTrackManager .setUserNameDetectListener ( new
FaceTrackManager.UserNameDetectListener () { @Override public void userName (String s)
{                Log. i ( TAG , "userName: $ s" + s) ; } }) ;
```

It should be noted that s may be an empty string

3.1 5 Desktop Motion Control Protocol

Access process:

1, application DesktopMotion Manager objects ,application codes as

```
DesktopMotion Manager desktop Manager = ( DesktopMotion Manager ) getUnitManager
(FuncConstant. DESKTOPMOTION_MANAGER );
```

2, use application of the **Desktop Manager** objects , calling the appropriate method for controlling.

3.1 5.1 stops moving

Method signature :

```
public OperationResult doStop ()
```

Method description :

Directly send instructions to the background service through the **desktop manager** object. The background service will stop the multi-axis movement after receiving the instruction.

Sample code:

```
desktopManager.doStop ();
```

.3..1.5.2 Desktop resetting movement

Method signature :

```
public OperationResult do Reset ()
```

Method description :

Directly send instructions to the background service through the **desktop Manager** object. The background service will reset after receiving the instruction.

Sample code:

```
desktopManager.doReset ();
```

.3..1.5..3 Desktop relative angular movement

Method signature :

```
public OperationResult doRelativeAngleMotion (RelativeAngleDesktopMotion
relativeAngleDesktopMotion )
```

Method description :

Directly send instructions to the background service through the **desktop manager** object. The background service receives the instructions and performs corresponding operations .

Speed has positive and negative, angle has no positive or negative

Vertical angle control range: 0-45

Horizontal angle control range: 0-160

Rolling angle control range: 0-160

Front and rear angle control range: 0-45

Sample code:

```
RelativeAngleDesktopMotion relativeAngleDesktopMotion = new RelativeAngleDesktopMotion () ;
relativeAngleDesktopMotion.setForwardDegree (( short ) 10 ) ; // Forward and backward motion angle 10
relativeAngleDesktopMotion.setForwardSpeed (( short ) 10 ) ; // Backward motion, speed control 1-60 , positive number direction after exercise, negative forward movement, backward movement is controlled by the speed of the positive and negative directions
relativeAngleDesktopMotion.setHorizontalDegree (( Short ) 10 ) ; // horizontal angle of movement 10
relativeAngleDesktopMotion.setHorizontalTurnSpeed (( Short ) 10 ) ; // speed control 1-60 to Left movement, positive number moves to the left, negative number moves to the right, the left and right movement direction is controlled by positive and negative speeds
relativeAngleDesktopMotion.setVerticalDegree (( short ) 10 ) ; // Vertical vertical movement angle 10
relativeAngleDesktopMotion.setVerticalSpeed (( short ) 10 ) ; / / Speed control 1-60 upward movement, positive number upward movement, negative number downward movement, positive and negative speed control up and down direction
relativeAngleDesktopMotion.setRollDegree (( short ) 10 ) ; // Rolling motion angle 10
relativeAngleDesktopMotion. setRollSpeed (( short ) 10 ) ; // Speed control 1-60 rolls to the right, positive numbers move to the right, negative numbers move to the left, and positive and negative speeds control the roll left and right movement direction
desktopMotionManager.doRelativeAngleMotion (relativeAngleDesktopMotion) ;
```

3.15.4 Desktop absolute angular motion

Method signature :

public OperationResult doAbsoluteAngleMotion (AbsoluteAngleDesktopMotion absoluteAngleDesktopMotion)

Method description :

Directly send instructions to the background service through the **desktop manager** object. The background service receives the instructions and performs corresponding operations.

Absolute angle control, speed can only be positive

Vertical angle control range: 0-45

Horizontal angle control range: 0-160

Rolling angle control range: 0-160

Front and rear angle control range: 0-45

Sample code:

```
AbsoluteAngleDesktopMotion absoluteAngleDesktopMotion = new AbsoluteAngleDesktopMotion ();
                absoluteAngleDesktopMotion.setForwardDegree ((short) 10);
                absoluteAngleDesktopMotion.setForwardSpeed ((short) 10);
           absoluteAngleDesktopMotion.setHorizontalDegree ((short) 10); // Horizontal motion angle 10
                absoluteAngleDesktopMotion.setHorizontalTurnSpeed ((short) 10);
           absoluteAngleDesktopMotion.setVerticalDegree ((short) 10); // Vertical vertical movement angle 10
                absoluteAngleDesktopMotion.setVerticalSpeed ((short) 10); absoluteAngleDesktopMotion.setRollDegree
           ((short) 10); // Rolling motion angle 10
                absoluteAngleDesktopMotion.setRollSpeed ((short) 10);
desktopMotionManager.doAbsoluteAngleMotion (absoluteAngleDesktopMotion);
```