Aobo Service Robot
SDK Interface
Description Document

Document Change Record

Number	Change Description	Author	Version	Date	Note
1	Single draft	Sheet	V1.0	2019/08/10	
2	update	Sheet	V1.1	2019/09/06	Improve content
3	update	Sheet	V1.3	2019/10/12	Add peripherals
4	update	Sheet	V1.5	2023/11/12	Add voice

Catalogue

1. Document Description

This SDK is designed for the navigation of Aobo hardware version 2.60, and the firmware of the motherboard is used after August 1st, 2019. The SDK used in this document is for use only. If you are unsure, please consult our technical after-sales service.

1.1 File configuration description

Compiler, it is recommended to use AS
Using the aar file provided by the project, add in the build.gradle of the project
Implementation (name:'aobobotsdk-v1.1 ', ext:'aar')
Give program permissions

```
<uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission     android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

The dependency library uses Eventbus as the event publishing--subscription bus. Other parameter settings can refer to the demo

2. Navigation interface description

2.1 Chassis and radar navigation service connection

2.1.1 Initialize management class

The robot management is a singleton mode.
AoboRobotManeger aoboRobotManeger
AoboRobotManeger.getInstance();

The following usage methods are all based on this management class, and if there are special methods will be pointed out

2.1.2 Connecting robot
Methods for connecting robots
aoboRobotManeger.connect();
The status of the connection can be notified using onEventMainThread to receive event status notifications.

2.1.3 Get the current map

The control for obtaining the map is RPMapView, and the initialization of the map is as follows
aoboRobotManeger.stopGetMap(true);
aoboRobotManeger.setAutoUpdateMap(true);
aoboRobotManeger.getMap(this, mapView);
For specific and detailed usage, please refer to the demo.

2.1.4 Instruction for robot movement

aoboRobotManeger.moveBy(MoveDirection direction);
[Function]
The instructions for moving are

[Parameter]
Direction of movement
Forward MoveDirection.FORWARD

Backward MoveDirection.BACKWARD
Turn left MoveDirection.TURN_LEFT
Turn right MoveDirection.TURN_RIGHT

2.1.5 Robot go to designated location

aoboRobotManeger.moveToPoint(Pose pose , String mapname,boolean is stackoa)

[Function]

The parameter is Pose, which means the point to be reached

[Parameter]

The mapname is the name of the map, which can be ignored. The isstakos parameter is whether to let the robot follow the virtual orbit

2.1.6 Robot clearing the map

aoboRobotManeger.clearMap();

[Function]

Robot clears current map data

2.1.7 Robot saves map

aoboRobotManeger.saveMap(String mapPath,String mapName)

[Function]

Save map

[Parameter]

Save two parameters of the map, the first being the saved path and the second being the saved name. (Used in conjunction with saving maps)

2.1.8 Robot loading map

aoboRobotManeger.loadMap(String mapPath,String mapName)

[Function]

Load map locally

[Parameter]

Save two parameters of the map, the first being the saved path and the second being the saved name. (Used in conjunction with loading maps)

2.1.9 Does the robot update the map

aoboRobotManeger.setAutoUpdateMap(boolean isupdata)

[Function]

Normally, When a robot enters, map updates will be enabled. If it is a loaded map, the updated map can be paused.

[Parameter]

Isupdate is true, update map; false, do not update

2.1.10 Robot repositioning

aoboRobotManeger.reLocation();

[Function]

When the data returned by the robot's radar scan does not match the contour of the map, the repositioning function can be used.

[Parameter]

2.1.11 Robot rotation angle

aoboRobotManeger.turnRound(short degree)

[Function]

Robot rotation angle

[Parameter]

The angle is an integer, ranging from 0 to 360.

2.1.12 Robot return pile charging

aoboRobotManeger.goHome();

[Function]

This method is return the robot to the charging station for charging, and the state of whether it returns can be obtained in the event. Specific reference demo

[Parameter**]**

2.1.13 Robot cancellation action

aoboRobotManeger.cancelAction();

[Function]

This method is to cancel the current navigation action of the robot

[Parameter]

2.1.14 Patrol operations of robots

aoboRobotManeger.partol(List<Location> locationList)

[Function]

This provides a simple robot patrol method

[Parameter]

Use the Localtion's point collection, without direction, stops navigation as stopPartol()

2.1.15 Virtual wall operation of robot

The virtual wall operation is directly added to the map, and in the added state, mapView. setVirtualWallIndicator (rawPoint);

Selecting two points on the map is to add a virtual wall, For specific examples, please refer to the demo

2.1.16 Robot status monitoring

Need to monitor robot status can inherit AoboNaviListener interface

void getUpdateMapStatus(GetUpdateMapStatusEvent var1);

[Function]

Map update status of robots

[Parameter]

GetUpdateMapStatusEvent Get update events

void getRobotHealthInfo(RobotHealthInfoEvent var1);

[Function]

Robot Health Information

[Parameter]

void getWallUpdate(WallUpdateEvent var1);

[Function]

Virtual wall status of robot

[Parameter]

void getRobotStatusUpdate(RobotStatusUpdateEvent var1);

[Function]

Robot battery positioning information

[Parameter]

void getRobotPoseUpdate(RobotPoseUpdateEvent var1);

[Function]

Robot location information

[Parameter]

void getMoveActionUpdateEvent(MoveActionUpdateEvent var1);

[Function]

Robot movement information

[Parameter]

void getMapUpdateEvent(MapUpdateEvent var1);

[Function]

Robot map data information

[Parameter]

void getLaserScanUpdateEvent(LaserScanUpdateEvent var1);

[Function]

Robot radar update information

[Parameter]

void getConnectionLostEvent(ConnectionLostEvent var1);

[Function]

Robot connection loss information

[Parameter]

void getConnectionFailedEvent(ConnectionFailedEvent var1);

[Function]

Robot connection fail information

[Parameter]

void getConnected(ConnectedEvent var1);

[Function]

Robot connection information

[Parameter]

For specific usage, please refer to the demo

3. Voice interface description

3.1 3.0 Communication Protocol（Developed based on AoboRobot3.0）

3.1.1 Brief introduction

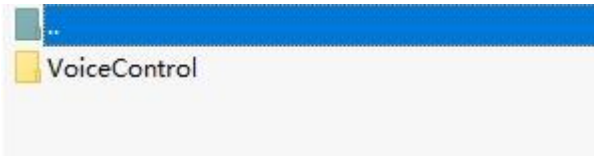
In order to facilitate users to directly call the robot's voice function, some interfaces are open to users, including microphone wake-up, voice recognition and synthesis, distance query and setting of ultrasonic waves, and simple machine motion control (based on interface definitions)

3.1.2 Integration Guide

Prepare the content. First, install the demo on the screen. The demo needs to be added to the AoboRobot homepage and opened from the homepage to receive communication data (calling interface services, secondary development applications must be opened from the homepage)

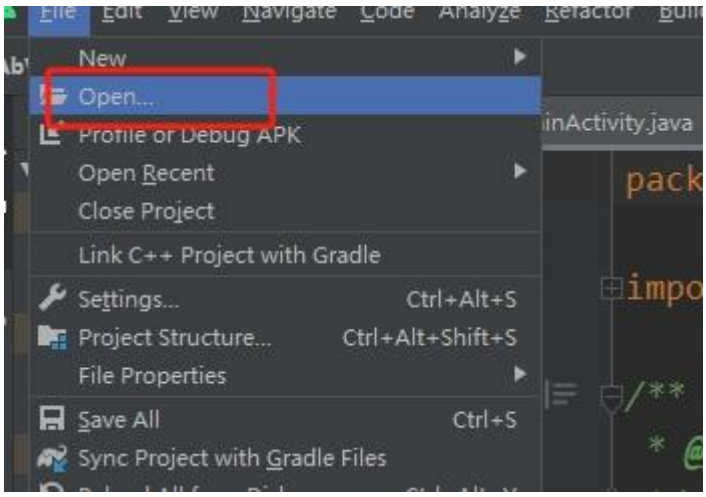
3.1.3 Demo Run Steps:

Directly use the provided file, the SDK includes a simple and runnable demo



（Import project）

Open Android Studio，In the menu bar: File--->open--->demo, Unzip SDK path.



Taking the Android Studio integrated development tool as an example, it is recommended to directly use a real machine for testing.

3.2 Common method description

3.2.1 Wake up microphone array

3.2.1.1 Code controlled wake-up

Request data example

Tag: "Wakeup"

Function position direct: represents the beam direction. Taking a 4-meter array as an example, the values are 0, 1, and 2. 0 represents the wave velocity between the first and second microphone heads, 1 represents the intermediate wave velocity, and 2 represents the wave velocity between the third and fourth microphone heads. If it is a 6-meter array, the values are 0-5.

Code example: sendMessage("wakeup",direct);

Return result example

tag=4 message=wake up,angle=90,beam=1,score=1427.0

“tag=4” ：4 Represents wake-up type
“ wake up,angle=90,beam=1,score=1427.0 ” angle: Representing the awakening angle of this time, beam:Indicate the direction of microphone wave velocity, score:Indicate the score of this wake-up word (the higher the score, the more accurate the wake-up word is)

3.2.1.2 Voice wake-up

same with code wake-up

3.2.2 Speech recognition

Request data example

TAG：“startrecord”

Function position：None

Code example： sendMessage("startrecord","");

Return result example

tag=1 message=0

“tag=1”: The function position represents the flag bit for returning sound, "message=0", 0 represents the current sound size

tag=2 message=Hello

“ tag=2 ”： The function bit represents the flag bit for speech recognition, with "message=hello" and "hello" indicating the recognition content obtained by the array

3.2.3 Stop Speech Recognition

Request data example

TAG：“stoprecord”

Function position：None

Code example： sendMessage("stoprecord ","");

Return result example

tag = 2, message = stoprecord

“tag=2”: The function bit represents the flag bit for speech recognition
“stoprecord”: The function bit indicates that speech recognition has been stopped

3.2.4 Start Speech Synthesis

Return result example

TAG: “starttts”

Function position MSG: represents the content of the synthesized text in this time

Code example: sendMessage("starttts ",“How are you? I'm here”);

Return result example

tag=3 message=starttts

“tag=3: Function bit represents the flag bit for speech synthesis
“starttts”: Function bit indicates that speech synthesis has started

tag=3 message= ttsover

“tag=3: Function bit represents the flag bit for speech synthesis
“ttsover”: Function bit indicates completion of speech synthesis

3.2.5 Stop speech synthesis

Request data example

TAG: “ttstop”

Function position MSG: None

Code example: sendMessage("ttsstop",“”);

Return result example

tag=3　message= ttsstop

“tag=3：Function bit represents the flag bit for speech synthesis
“ttstop”：Function bit indicates pause of speech synthesis

3.3 Query the number of voice calls

Request data example

TAG：“queryaiui”

Function position none:

Code example：　sendMessage("queryaiui","");

Return result example

tag=4　message=9

“tag=4：Function bit represents the voice query function bit
“message=9”：Function bits, number of times of current calls
If the code query is not accurate, you can go to the backend settings of AoboRobot's main program to check

When using enhanced recognition, it is also possible to modify the content of the received recognition message

3.4 Open the specified interface

Request data example

TAG：“openpage”

Function position PAGENAME：Page name
The currently supported open pages include {explanation page, navigation page, settings page}, and the corresponding function bits are {pagevoicelead, pagevoiceexplain, pagesetting }

Code example：　sendMessage("openpage ",“pagevoicelead”);

Return result example

tag=5　message=open_pagevoiceexplain

“tag=5：Function bit indicates the function bit for opening the page
“message= open_pagevoiceexplain”：Indicates that the explanation page has been opened

4.Machine arm control interface description

To be updated

5.Face recognition interface description

To be updated

6.Peripheral

The robot peripherals include a fingerprint reader, a receipt printer, and the use of 4-in-1 (ID card, bank card, social security card, RFID card). Please refer to the project aobooutdevice for specific usage

6.1 Small ticket printer

The dependency library for the small ticket printer is printerlibs. When using it, the implementation project (': printerlibs') is required. The small ticket printer can be printed using both USB and serial ports. It is recommended to use USB for printing

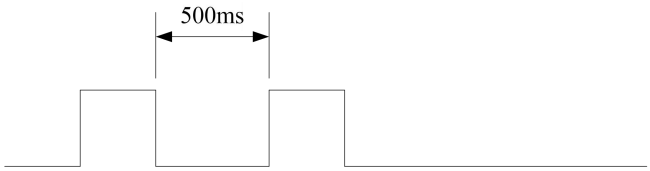
6.1.1 1.Print test

After powering on, press and hold the K1 button short contact on the board, release it, and the control board will print a test page.
There may be variations between boards with different names of short-circuit points.

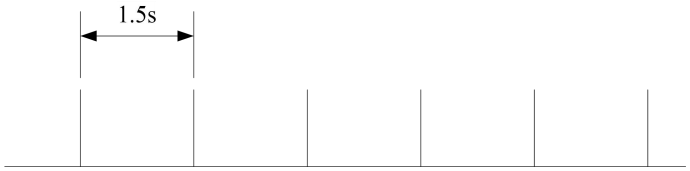
6.1.2 2. Flashing indicator light

These pictures show the waveform of the LED on the thermal control panel. The vertical line represents the number of LED flashes, 500ms represents a pause time, 400ms represents the duration of the flash, and 1.5s represents the duration of the stop after the LED flashes.

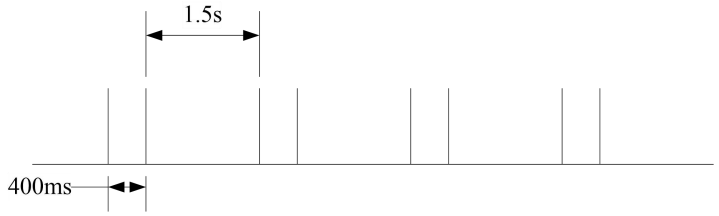
Power on:



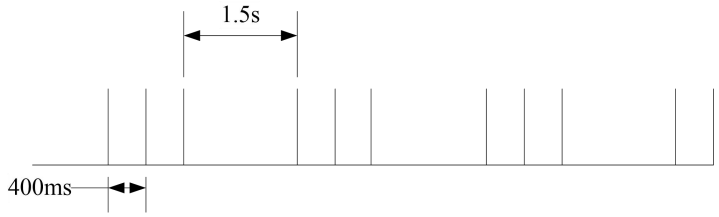
Normal work:



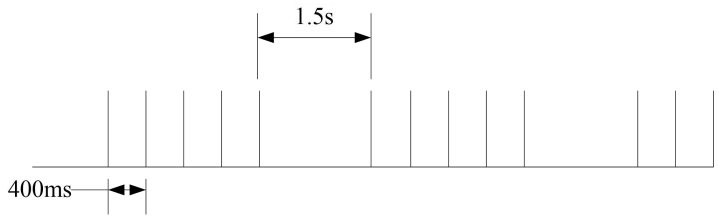
No printer detected:



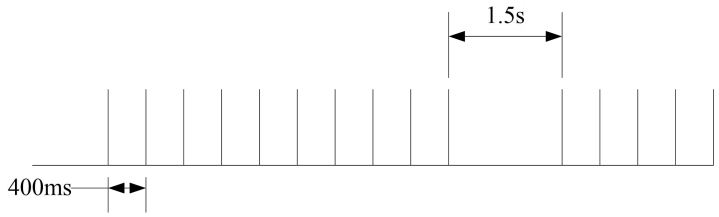
printer out of paper:



The heating element of the printer core is overheated::



Chinese font library chip not detected::



6.1.3 Introduction to printer methods

public void CaysnPos_Close(Pointer handle);

[Function]

Close port

[Parameter]

Printer handle

public int CaysnPos_ResetPrinter(Pointer handle);

[Function]

The method is to reset the printer

public int CaysnPos_SetPrintSpeed(Pointer handle, int nSpeed);

[Function]

Set the speed of the printer

[Parameter]

The parameter is nSpeed, which is the printer's printing speed in millimeters per second

public int CaysnPos_QueryPrinterStatus(Pointer handle, int timeout);

[Function]

Real time query of printer status (cash box, paper, error)

[Parameter]

Timeout is the millisecond timeout period that returns the printer status, - 1 indicates query failure. Detailed status information can be accessed through macro definitions

ublic int CaysnPos_QueryPrintResult(Pointer handle, int timeout);

[Function]

Query printing results

[Parameter]

Timeout is the timeout time, and the return value returns the corresponding status. Each value is defined as follows

Value	Definition
0	Printing successful
-1	Port closed
-2	Write failed
-3	Read failure

-4	The printer is unresponsive
-101	other error
-102	The printer is offline
-103	The printer is out of paper

public int CaysnPos_SetPrintDensity(Pointer handle, int nDensity);

[Function]

Set printing concentration,

[Parameter]

The parameter nDensity sets the printing concentration, and each value is defined as follows

Value	Definition
0	light
1	medium
2	concentrated

public int CaysnPos_FeedLine(Pointer handle, int numLines);

[Function]

Printer feed specified number of lines

[Parameter]

The number of lines to enter for parameter numLines, and the return value only indicates whether the instruction was successfully written. Returning true indicates successful writing, while returning false indicates failed writing.

public int CaysnPos_PrintTextA(Pointer handle, String str);

[Function]

Print text

[Parameter]

The parameter str is the string to be printed

public int CaysnPos_PrintTextInUTF8W(Pointer handle, WString str);

[Function]

Print text

[Parameter]

The parameter str is the string to be printed. Note that this function will convert the data to UTF8 encoding and send it.

There are several other methods for printing text, please refer to the demo code

public int CaysnPos_SetTextScale(Pointer handle, int nWidthScale, int nHeightScale);

[Function]

Set text magnification

[Parameter]

The parameter nWidthScale is the width amplification factor, and nHeightScale is the height amplification factor

public int CaysnPos_SetAsciiTextFontType(Pointer handle, int nFontType);

[Function]

Set English character font type

[Parameter]

The parameter nFontType is an English character font type, with each value defined as follows:

Value	Definition
0	Font A（12x24）
1	Font B（9x17）

public int CaysnPos_SetTextBold(Pointer handle, int nBold);

[Function]

Set bold text printing

[Parameter]

Is the parameter nBold bold? The values are defined as follows:

Value	Definition
0	Not bold
1	Bold

public int CaysnPos_SetTextUnderline(Pointer handle, int nUnderline);

[Function]

Set text underline

[Parameter]

The parameter nUnderline text is underlined, and each value is defined as follows:

Value	Definition
0	No underline
1	1 point underline
2	2 point underline

public int CaysnPos_SetTextUpsideDown(Pointer handle, int nUpsideDown);

[Function]

Set text inverted printing

[Parameter]

The parameter nUpsideDown is printed upside down, and the values are defined as follows:

Value	Definition
0	No Inverted printing
1	Inverted printing

public int CaysnPos_SetTextWhiteOnBlack(Pointer handle, int nWhiteOnBlack);

Set black and white reverse display

[Parameter]

The parameter nWhiteOnBlack is inverted in black and white, and the values are defined as follows:

Value	Definition
0	No black and white reverse display
1	Black and white reverse display

public int CaysnPos_SetTextRotate(Pointer handle, int nRotate);

[Function]

Set text rotation to 90 degrees for printing

[Parameter]

The parameter nRotate is for rotary printing, and the values are defined as follows:

Value	Definition
0	Non rotating printing
1	Rotating printing

public int CaysnPos_SetTextLineHeight(Pointer handle, int nLineHeight);

[Function]

Set row height

[Parameter]

Parameter nLineHeight, row height, range [1255]

public int CaysnPos_SetAsciiTextCharRightSpacing(Pointer handle, int nSpacing);

[Function]

Set the right blank space of ASCII characters,

[Parameter]

]Blank space on the right side of parameter nSpace, range [1255]

public int CaysnPos_SetKanjiTextCharSpacing(Pointer handle,int nLeftSpacing,int nRightSpacing);

[Function]

Set left and right blank space for Chinese text characters

[Parameter]

Parameter 1: nLeftSpacing with blank space on the right, range [1255]

Parameter nRightSpacing with blank space on the right, range [1255]

public int CaysnPos_SetSingleByteMode(Pointer handle);

[Function]

Set the printer to single byte encoding

[Parameter]

public int CaysnPos_SetCharacterSet(Pointer handle, int nCharacterSet);

[Function]

Set printer character set

[Parameter]
Parameter nCharacterSet printer character set, range [0,15]

public int CaysnPos_SetCharacterCodepage(Pointer handle, int nCharacterCodepage);
[Function]
Set character code page
[Parameter]
The parameter nCharacterCodepage is the character code page, with a range of [0,255]

public int CaysnPos_SetMultiByteMode(Pointer handle);
[Function]
Set the printer to multi byte encoding
[Parameter]

public int CaysnPos_SetMultiByteEncoding(Pointer handle, int nEncoding);
[Function]
Set the printer to multi byte encoding
[Parameter]
The parameter nEncoding is a multi byte encoding, with each value defined as follows:

Value	Definition
0	GBK
1	UTF8
2	
3	BIG5
4	SHIFT-JIS
5	EUC-KR

The commonly used methods are listed here, and other methods that need to be used can refer to the demo or the printerlibs.caysnpos_zh.txt file in the libs file

6.2 Fingerprint collector

Copy the fingerlibs directory to the corresponding libs directory of the Android Studio project.
Including fingerprint collector equipment class, control equipment class, constant class, exception handling class, algorithm processing class, etc. The corresponding types of each type are shown in the following figure:

Class Name	Type
com.zkteco.android.biometric.module.fingerprintreader.FingerprintSensor	Fingerprint collector equipment class

com.zkteco.android.biometric.module.fingerprintreader.FingerpringStatusCode	Error code
com.zkteco.android.biometric.module.fingerprintreader.exception.FingerprintException	Exception handling class
com.zkteco.android.biometric.core.utils.LogHelper	Logging tool class
com.zkteco.android.biometric.core.utils.ToolUtils	Auxiliary tools
com.zkteco.android.biometric.module.fingerprintreader.FingerprintCaptureListener	Collect listening event classes
com.zkteco.android.biometric.module.fingerprintreader.ZKFingerService	Algorithm processing class

6.2.1 Introduction to fingerprint collector methods

The FingerprintSensor. Class, operation refers to the fingerprint device class. For example, turning on the device, turning off the device, and starting procurement
Collect, stop collection, etc.
Public void open (int index)
[Function]
Connecting devices
[Parameters]
Index
The device index number, which is determined by the total number of connected collectors.
For example:
When the total number of collectors is 1, the value of index is 0;
When the total number of collectors is 2, the value of index is 0 or 1;
And so on and so on
[Return value]
[Note]
Error code thrown as an exception

public void close(int index)
[Function]
Turn off device
[Parameter**]**
index

The device index number, which is determined by the total number of connected collectors.
For example:
When the total number of collectors is 1, the value of index is 0;
When the total number of collectors is 2, the value of index is 0 or 1;
And so on and so on
[Return value]
[Note]
Error code thrown as an exception

public void setFingerprintCaptureListener(int index,
FingerprintCaptureListener listener)

[Function]

Set fingerprint collection listening events

[Parameter]

index

The device index number, which is determined by the total number of connected collectors.
For example:
When the total number of collectors is 1, the value of index is 0;
When the total number of collectors is 2, the value of index is 0 or 1;
And so on and so on

Listener

listening object
[Return value]
[Note]

public void startCapture(int index)

[Function]

Start taking images

[Parameter description]

index

The device index number, which is determined by the total number of connected collectors.
For example:
When the total number of collectors is 1, the value of index is 0;
When the total number of collectors is 2, the value of index is 0 or 1;
And so on and so on
[Return value]
[Note]
Asynchronous image retrieval, returning images and templates through the callback interface set by setFingerprintCaptureListener. (See Demo for details)

public void stopCapture (int index)

[Function]

Stop image retrieval (asynchronous)

[Parameter description]

index

The device index number, which is determined by the total number of connected collectors.

For example:

When the total number of collectors is 1, the value of index is 0;

When the total number of collectors is 2, the value of index is 0 or 1;

And so on and so on

[Return value]

[Note]

Stop image retrieval (asynchronous).

public static void destroy(FingerprintSensor fingerprintSensor)

[Function]

Destruction of resources

[Parameter description]

fingerprintSensor

Device operation instance object

[Return value]

[Note]

public static int getImageWidth()

[Function]

Obtain fingerprint image width.

[Parameter description]

[Return value]

Fingerprint image width.

[Note]

public static int getImageHeight()

[Function]

Obtain fingerprint image height

[Parameter description]

[Return value]

Fingerprint image height

[Note]

public static int getLastTempLen ()

[Function]

Obtain the length of fingerprint template data.

[Parameter description]

[Return value]

The length of fingerprint template data.

[Note]

public static void setFakeFunOn(int fakeFunOn)

[Function]

Set anti fake switch.

[Parameter description]

fakeFunOn

0: indicates that anti-counterfeiting is turned off

1: Indicates enabling anti-counterfeiting

[Return value]

[Note]

public static int setFakeFunOn()

[Function]

Set anti fake switch

[Parameter description]

[Return value]

0: indicates that anti-counterfeiting is turned off

1: Indicates enabling anti-counterfeiting

[Note]

public static int getFakeStatus()

[Function]

Get the current fingerprint status

[Parameter description]

[Return value]

All 5 lower bits are 1, it indicates a true fingerprint: (value&0x1F)==31

Other: Suspected fingerprints

[Note]

Only support **SILK20R**，and **setFakeFunOn(1)**

[Function**]**

void captureOK(byte[] fpImage);

[Function]

Image acquisition successful

[Parameter]

fpImage

Fingerprint image

[Return value]

[Note]

void captureError(FingerprintSensorException e)

[Function]
Image acquisition failed
[Parameter]
[Return value]
Non
[Note]

void extractOK(byte[] fpTemplate);
[Function]
Collected template successfully
[Parameter**]**

fpTemplate

Fingerprint template
[Return value]
Non
[Note]

void extractError(int errno)
[Function]
Template extraction failed

[Parameter**]**

Please refer to the "Error List" for error code descriptions
[Return value]
[Note]

static public int verify(byte[] temp1, byte[] temp2)
[Function]
Compare two fingerprint templates

[Parameter**]**

temp1

The first fingerprint template data array

temp2

The second fingerprint template data array
[Return]
The matching score of the fingerprint template, with a return value range of 0 to 100.
Suggestion: When the return value is greater than the set threshold or default value (23), it can be considered that the two fingerprint templates have matched successfully.
[Note]
Please refer to the "Error List" for error code descriptions

static public int verifyId(byte[] temp, String id)

[Function]

Match the fingerprint template with the fingerprint template with the specified ID in the cache (i.e. 1:1 comparison), and return the matching score.

[Parameter]

temp

Fingerprint template data array

id

The ID of the matched target fingerprint template. The ID type is a string, and the string length cannot exceed 20 bytes.

For example： String id = “ZKTeco_20_01”。

[Return value]

The matching score of the fingerprint template, with a return value range of 0 to 100.

When the return value is greater than the set threshold or the default value (23), it indicates that the two fingerprint templates have been successfully matched.

static public int identify(byte[] temp, byte []idstr, int threshold, int count)

[Function]

Match the fingerprint template data with the cached fingerprint template data (i.e. 1: N comparison), and return count results with a matching score greater than threshold. If the actual number of results is less than count, return the actual number of results. For example, if count is set to 2, the return value may be 0,1,2.

[Parameter]

Temp

Fingerprint template data array

idstr

Returns the ID numbers and an array of scores after successful comparison. Suggest setting the array size to 50 * count.

Multiple results are arranged in descending order of matching scores, with each result including ID and match

The allocation score, ID, and matching score are separated by ’/t’, and the results are separated by ’/n’.

For example： String resultStrs= new String(idstr);

Such as the return value of resultStrs is ： “1001’ \t’ 95’ \n’ 1002’ \t’ 85’ \
n’ 1003’ \t’ 75”，Then it means returning three comparison results:
Result 1： id： 1001， score： 95；
Result 2： id： 1002， score： 85；
Result 3： id： 1003， score： 75；
Please refer to [Example Code] for detailed code.

threshold

Matching threshold, value range from 0 to 100 (note: this value will overwrite the previously set threshold)
Value).
The matching threshold is the boundary value used to determine the similarity between two fingerprint feature templates
The higher the setting, the stricter the requirement for similarity. Two fingerprint templates are compared and the returned value is greater than the set threshold, indicating successful comparison. Otherwise, the comparison fails. Usually, it can be flexibly adjusted according to the situation in specific applications.

Matching threshold description

Rejection rate	Misjudgment rate	Matching threshold	
1:N	1:1		
High	low	65	45
Middle	middle	55	35
Low	high	45	30

count

It is recommended to set the expected number of results to 1.
For example：
threshold set 50， count set 3。If only 2 templates match the matching threshold during the fingerprint template comparison process, the function returns a value of 2.
[Return value**]**
Successfully returned the actual number of comparison results. The return value may be less than count. 1: Represent 1 finger
Pattern template matching successful. 0: indicates that no fingerprint template matching was successful.
Otherwise, an error code will be returned
[Note]
Please refer to the "Error List" for error code descriptions
[Example code**]**
Refer to demo

static public int merge(byte[] temp1, byte[] temp2, byte[] temp3, byte[]temp)

[Function]

Merge 3 fingerprint templates into registration templates

[Parameter]

temp1

Pending registration fingerprint template 1

temp2

Pending registration fingerprint template 1

temp3

Pending registration fingerprint template 1

temp

Output registration fingerprint template

[Return value**]**

>0 other	Registration template length failed

[Note**]**

static public save (byte[] temp, String id)

[Function**]**

Save fingerprint template data to cache. Users save the fingerprint data they need in their database to the cache by calling the save method.

[Note**]**

The call to save must ensure that the fingerprint algorithm library has been initialized (init) successfully.

[Parameter**]**

teTemp

Incoming fingerprint template data array

id

Fingerprint temple id。

The ID type is a string, and the string length cannot exceed 20 bytes.

For example： String id = “ZKTeco_20_01”。

[Return value**]**

0	Success,Otherwise, return an error code

[Note]

static public int get (byte[] temp, String id)

[Function]

Obtain fingerprint template data for the specified ID in the cache.

[Parameter]

Temp

An array of fingerprint template data with a specified ID. Suggest setting the size of the array to1024 * 3 to ensure sufficient storage space for fingerprints

id

The fingerprint template ID that needs to be obtained.
The ID type is a string, and the string length cannot exceed 20 bytes.
For example： String id = “ZKTeco_20_01”。

[Return value]

>0	Success, the value is the length of the fingerprint template. Otherwise, an error code will be returned

[Note]

static public int del(String id)

[Function]

Delete registered fingerprint templates from memory

[Parameter]

id

The ID of the fingerprint template that needs to be deleted.
The ID type is a string, and the string length cannot exceed 20 bytes.
For example： String id = “ZKTeco_20_01”。

[Return value]

0	Success,Otherwise, return an error code

[Note]

Please refer to the "Error List" for error code descriptions

static public int clear ()

[Function]

Clear all fingerprint data from cache

[Parameter]

Nothing

[Return value]

0	Success,Otherwise, return an error code

[Note**]**
Please refer to the "Error List" for error code descriptions

static public int count()
[Function**]**
Get the current number of fingerprint templates stored in the cache

[Parameter**]**

Nothing
[Return value**]**
The number of fingerprint templates saved in the current cache

6.2.2 Fingerprint error list:

Error code	Remarks
-20001	Failed to open device
-20002	Device shutdown failed
-20003	Failed to obtain GPIO
-20004	Failed to set GPIO
-20005	Reading EEPROM failed
-20006	Failed to obtain image from USB
-20007	Failed to detect USB image
-20008	Insufficient input cache
-20009	Abnormal data reading
-20010	Fingerprint collection failed
-20011	Failed to decrypt image data
-20012	Failed to start collection thread
-20013	Stopping collection thread failed
-20014	Failed to initialize fingerprint device
-20015	Setting calibration parameters failed
ERR_NOT_FOUND -5000	Unable to find the designated ID

	fingerprint template
ERR_PARAM -5002	Parameter error
ERR_TEMPLATE -5003	Fingerprint template error
ERR_METHOD -5004	Method error

6.3 Four in One Equipment

The four in one device currently includes the reading of bank cards, ID cards, social security cards, and RFID. Both bank cards and social security cards are card inserted, while ID cards and RFID use contact reading.

6.3.1 Introduction to Project Deployment

In the example, the 4-in-1 includes different functions, so it exists as a single app separately. For details, please refer to the cra4app in aobooutdevice

IdCardActivity Corresponding to the use of ID card

SocialCardActivity Corresponding to the use of social security cards

CreditCardActivity CreditM1Activity Corresponding to the use of bank cards

DebitCardActivity Corresponding to the use of RFID cards

6.3.2 Introduction to Four in One Equipment Methods

public static IDCard dc_get_i_d_raw_info()

[Function]

IDCard ID card reading

[Parameter]

Nothing

[Return value]

IDCard class, which stores the information of the ID card

public static native String dc_cpureset_hex();

[Function]

reset

[Parameter]

Nothing

[Return value]

The returned string

public static native String dc_cpuapdu_hex(String var0);

[Function**]**

Read information for different cards based on their type

[Parameter**]**

Different functional parameter codes (please refer to the demo for details)

[Return value]

Return specific information