SUNMI Scanning SDK

Why to use the scanning SDK encapsulated by SUNMI?

SUNMI provides the scanning SDK that fits your devices. Compared to the open-source solutions currently in use, SUNMI's scanning SDK has the following advantages:

1. A high recognition rate. Through a large number of tests simulating real-life scenarios, SUNMI's scanning SDK has increased the recognition rate by 74% on average, compared to common solutions based on the ZXing open-source project.
2. A speed almost 100% higher than ZXing's solutions when scanning 1D barcodes.
3. Easy to use. With five lines of code, you are able to add scan function in your project.
4. Support scanning 17 types of codes, and more code types will be added later.
5. Perfectly fitted to SUNMI's devices. The combination of hardware and software ensures highly efficiency and stability.

How to use SUNMI's scanning SDK?

You have two options:

1. Use your App to call the barcode scanning module integrated by SUNMI's UI system for scanning, and get return values. This method is very easy to use.
2. Write your own camera interface and call the scanning SDK encapsulated by SUNMI to parse images. This method is relatively complicated but provides higher degrees of freedom.

The first method:

To make development easy, SUNMI OS has a built-in module for scanning. You can call it by startActivityForResult(), and then receive return values of the scan results by onActivityResult().

You can refer to the following code:

```
/*
Create an Intent where you want to start scanning; Call the scanning module by startActiityForResult().
*/
        Intent intent = new Intent("com.summi.scan");
        intent.setPackage("com.sunmi.sunmiqrcodescanner")
/*
        The method has the same function as above.
        Intent intent = new Intent("com.summi.scan");
        intent.setClassName("com.sunmi.sunmiqrcodescanner",
        "com.sunmi.sunmiqrcodescanner.activity.ScanActivity");

        Control the functional options by passing parameters.
        Add configurations only when needed, as all parameters have a default value.
        intent.putExtra("CURRENT_PPI", 0X0003);// The current resolution
        //The best for M1 & V1 is 800*480.
        //PPI_1920_1080 = 0X0001;
        //PPI_1280_720 = 0X0002;
        //PPI_BEST = 0X0003;
        intent.putExtra("PLAY_SOUND", true);// Play a voice prompt after scanning; Default: True
        intent.putExtra("PLAY_VIBRATE", false);// Vibrate after scanning; Default: False. Note: Only M1 supports this configuration currently; V1 does not.
        intent.putExtra("IDENTIFY_MORE_CODE", false);// Identify multiple 2D barcodes on the screen; Default: False
        intent.putExtra("IS_SHOW_SETTING", true);// Show the SETTINGS button in the upper right corner; Default: True
        intent.putExtra("IS_SHOW_ALBUM", true);// Show the SELECT PHOTOS FROM GALLERY button; Default: True
```

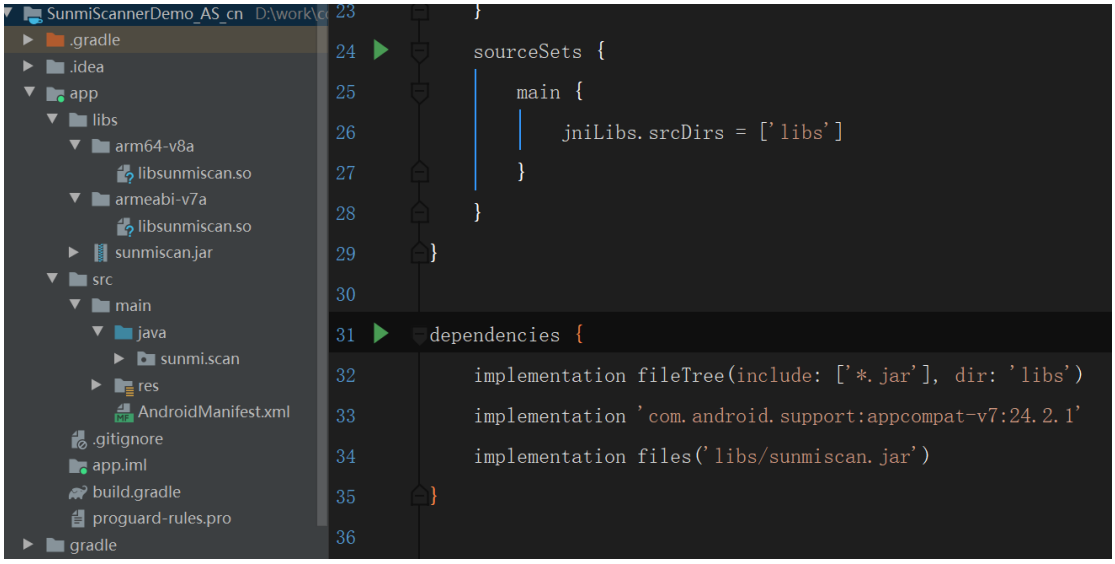
```
*/
        startActivityForResult(intent, START_SCAN);
```

Receive the return parameters of the scan results by onActivityResult. You can refer to the following code:

```
        @Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && data != null) {
                Bundle bundle = data.getExtras();
                ArrayList result = (ArrayList<>) bundle .getSerializable("data");
                Iterator<> it = result.iterator();
                while (it.hasNext()) {
                        HashMap hashMap = it.next();
                        Log.i("sunmi", hashMap.get("TYPE"));// Scan type
                        Log.i("sunmi", hashMap.get("VALUE"));// Scan result
                }
        }
}
```

The second method:

1. Add the two files libsunmiscan.so and sunmiscan.jar to the project's libs folder according to the following levels, and add references of the scanning library to build.gradle.



2. Import relevant Java interfaces in the code.

```
import com.sunmi.scan.Config;

import com.sunmi.scan.Image;

import com.sunmi.scan.ImageScanner;

import com.sunmi.scan.Symbol;

import com.sunmi.scan.SymbolSet;
```

3. Initialize relevant parameters.

```
private ImageScanner scanner;

scanner = new ImageScanner();// Create a scanner


/* Create decoded images. Width and height are the width and height of the camera's preview resolution. Generally, the higher the resolution is, the clearer the image is but the lower the decoding speed is. A resolution no bigger than 1280*720 is recommended, like 640*480, 800*480, 1280*720. You can get the resolution supported by the current device by getSupportedPreviewSizes. The default format of the preview image is YCbCr_420_SP, and the parameter "Y800" means to take the "Y" component in YUV.

*/

Image source = new Image(previewSize_width, previewSize_height, "Y800");
```

```
        scanner.setConfig(0, Config.ENABLE_MULTILESYMS, 0);// Decode N barcodes at one time for the same image; 0: only one, 1: multiple

        scanner.setConfig( Symbol.QRCODE,Config.ENABLE, 1);// Identify QR codes; Default: Allow

        scanner.setConfig( Symbol.PDF417,Config.ENABLE, 1);// Identify PDF417 barcodes; Default (0): Deny

        scanner.setConfig(Symbol.DataMatrix, Config.ENABLE, 1);// Identify DataMatrix codes，Default (0): Deny

        scanner.setConfig(Symbol.AZTEC, Config.ENABLE, 1);// Identify AZTEC codes; Default (0): Deny
```

4. Decode images. If you use an Android camera, you can directly call the preview data to decode, by PreviewCallback.onPreviewFrame(byte[] data, Camera camera).

```
    source.setData(data); // Fill the raw data of the camera as the image data.

    int result = scanner.scanImage(source); // Decode.
```

If the images are of other formats, such as photos from your gallery, the format needs to be converted to BMP file format; If they are color images, they also need to be in grayscale, for currently the decoding library can only handle images in grayscale (i.e. the brightness value of each pixel point ranges from 0 to 255).

5. Get decoding results and barcode types

```
if(result >0){

        SymbolSet syms = scanner.getResults();

        for (Symbol sym : syms) {

                Log.i("sunmi", "Types:"+sym.getSymbolName());// Barcode Types, such as “EAN-8”

                Log.i("sunmi","Results:"+sym.getResult())// Strings of decoding results

        }

}
```

Additional Guidance

1. Types of code currently supported by SUNMI's scanning SDK include:

   - 1D Barcodes: EAN-8, EAN-13, UPC-A, UPC-E, Codabar, Code39, Code93, Code128, ISBN10, ISBN13, DataBar, DataBar Expanded, Interleaved 2 of 5
   - 2D Barcodes: QR Code , PDF417，DataMatrix，AZTEC

2. Remember to update the two files libsunmiscan.so and sunmiscan.jar when you update the decoding library.

3. Although the latest decoding library provides 64-bit version (under the arm64-v8a folder), devices on 64-bit platforms are also compatible with 32-bit ones (under the armeabi-v7a folder). You can choose the platform type in build.grande.

```
externalNativeBuild {
            cmake {
          //        abiFilters "arm64-v8a" // Choose arm64-v8a
              abiFilters "armeabi-v7a" // Choose armeabi-v7a
            }
      }
```