Create BarCode Reading Application Demo in Android

BarCode Scanner is an app for decoding one-dimensional and two-dimensional code. By using the phone's camera, this app will quickly scan and recognize the information of one-dimensional code and two-dimensional code.

Supported barcode types

- 1D Barcodes: Code 11, Code 39, Code 93, Code 128, Codabar, Interleaved 2 of 5, Patch Code, EAN13, EAN8, UPC-A, UPC-B, Plus 2, Plus 5, Intelligent Mail, Planet.
- 2D Barcodes: Data Matrix, PDF417, QRCode, MicroQR

1 . ZXing ("Zebra Crossing")

ZXing (pronounced "zebra crossing") is an open-source, multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages. Our focus is on using the built-in camera on mobile phones to scan and decode barcodes on the device, without communicating with a server. However the project can be used to encode and decode barcodes on desktops and servers as well. We currently support these formats:

- UPC-A and UPC-E
- EAN-8 and EAN-13
- Code 39
- Code 93
- Code 128
- ITF
- Codabar
- RSS-14 (all variants)
- QR Code
- Data Matrix
- Aztec ('beta' quality)
- PDF 417 ('alpha' quality)

This library is divided into several main components which are actively supported:

2 . Zbar on Android with the NDK

We've had experience writing an automobile VIN (Code 39) scanner on both iPhone and Android. In the case of the iPhone we leverage the ZBar library and the Android the ZXing library. Unfortunately, Android has been a tremendous amount of work. There was a tremendous amount of fracturing with the camera drivers to code around to tune the image capture. But that's another story...

For the image processing the ZXing library on Android with was simply not as fast or accurate as capturing Code 39 barcodes (no opinion on QR, etc.) as ZBar on the iPhone. This left an interesting customer service dilemma as Android would be subpar. We looked at a few options and settled on an interesting proposition.

Android recently introduced the NDK (Native Development Kit) that allows you to build native code from C/C++. Interesting... We realized that there was a good chance that we could get the ZBar library running on Android with the NDK (at least for the image processing). We did have to rewrite the build/make system as the NDK does not have full make/configure support. But, after investing time in this we were able to build the ZBar library on the NDK and leverage their JNI

wrapper to invoke this code on Android.

So far the results have been excellent. The ZBar library on Android is 50% faster in recognition for Code 39 and a lot more accurate.

We'll be donating an example project to the ZBar project once we can remove the client specific portions.

Also, just be aware this only applies for Code 39. We have no idea how the image processing quality compares for the other formats.

3 . Payed SDK

1 :- **SD-TOOLKIT® Barcode Reader SDK**

http://www.sd-toolkit.com/products_sdtbarcode_android.php

2 :- **QuickMark SDK**

http://www.quickmark.cn/en/basic/SDK.asp

3 :- **Barcode Scanner SDK**

http://www.scandit.com/pricing/

4 :- **Barcode Scanner SDK**

http://shopsavvy.mobi/developers/

5 :- **Mobile Barcode SDK**

http://www.accusoft.com/barcodemobiledownload.htm