Barcode Scanning with a Smartphone - Android (v2.1.x)

Barcode Scanning with a Smartphone

It is important to recognize that there are several fundamental differences in the capabilities of smartphones and tablets as barcode scanning devices. These differences result in a user experience different from purpose-built scanners, impacting the design of the mobile barcode scanning application.

These differences and the general impact they have on your application are the following:

- A smart phone does not have a dedicated hardware trigger. Without a hardware trigger, the application program is generally responsible for initiating the scanning process, which results in accessing the built-in camera, displaying a preview screen if required, and analyzing captured frames from the video stream for barcodes.
- A smartphone, unless otherwise configured, does not have an aimer. Generally, the application program provides a live-stream camera preview on the mobile device screen, thereby allowing the user to see what the camera sees and can then position the device over the barcode.
- Mobile device orientation may need to be considered. Most users hold and use a mobile device primarily in a portrait orientation for barcode scanning, which is generally sufficient. However, most mobile device cameras have a higher resolution along their landscape orientation. When scanning very long or dense barcodes, reorienting the device to landscape can be beneficial and even necessary.
- Image analysis and barcode decoding is performed in software on the mobile device which can be a CPU intensive task. It is highly recommended to only enable the symbologies and features of the SDK that your application needs.

The cmbSDK makes these differences as transparent as possible to the application developer and the user. By following a few simple guidelines, you can develop applications that work and behave the same when using for example an MX-1000 mobile terminal, or just the built-in camera of the device.

Mobile Device Triggering

Without a hardware trigger, mobile devices must use alternative methods to initiate barcode scanning. There are three common methods used:

- **Application or workflow driven trigger**: The application code or the business logic/workflow of the application starts the scanning process. In other words, when the user of the application has reached a point where a barcode needs to be scanned, the application invokes the scanning module. In simple programming terms, this is akin to calling a function like "startScanner()".
- **Virtual trigger**: The application program provides a button on the screen which the user can use to start/stop the scanning process. Depending on the application design, the user may be required to press and hold the virtual button to keep the scanner running. This is similar to the workflow driven method as the button from the user interface is used to invoke the scanning module.
- **Simulated trigger**: One of the buttons on the mobile device, typically the volume-down button, is used to simulate a hardware trigger. When the user presses and holds this button, the scanner starts/stops just like when a trigger is pulled on a purpose-built scanner. This method is not commonly used as users find it non-intuitive and inconvenient.

The cmbSDK supports all three methods, any of which (or multiple) can be used in an application.

Mobile Device Aiming

Unlike purpose-built scanners, mobile devices do not have a built-in aimer. Barcode aiming is generally accomplished by providing a live-stream preview from the camera on the mobile device display. The user can then reposition the device until the barcode appears in the field of view and is decoded.

The cmbSDK simplifies this task by providing a built-in preview control that can be displayed fullscreen, partial screen, and in either portrait or landscape orientation.

The cmbSDK also supports "passive" aimers: devices that attach to the mobile device or its case that use the LED flash of the device as a light source to project an aiming or targeting pattern. The advantage of these aimers is that an on-screen preview is no longer required, as the mobile device can project an aimer pattern similar to a purpose-built scanner. However, note that using the LED flash for general scanning illumination is not available because the mobile device flash is used for the aimer.

Mobile Device Orientation

Mobile devices support developing applications for either portrait orientation, landscape orientation, or auto-rotation between the two. The cmbSDK fully supports all three options for both the presentation of the barcode preview as well as the scan direction. Most barcodes can be scanned by a mobile device regardless of the orientation of the application and/or the mobile device.

For scanning "square" barcodes like QR, Data Matrix, and MaxiCode, any orientation can be used. However, for scanning long or dense barcodes like a large PDF417, using landscape orientation is recommended or even necessary. Mobile cameras have a higher resolution along the "height" of the image in portrait mode. For example, 1280x720 is a commonly used resolution. This means that scanning barcodes using portrait orientation provides 720 pixels of data along the horizontal axis, while landscape orientation provides 1280 pixels on the horizontal scan line for barcode decoding.

Mobile Device Performance

With multi-core CPUs and dedicated image processors, today's handheld devices have significant computing power and provide an ideal platform for efficient and cost-effective barcode decoding. It is still recommended for developers to optimize their barcode scanning applications. The cmbSDK is optimized specifically for mobile environments, but image analysis and barcode decoding is still a CPU intensive activity. Since these processes share the mobile device's CPU with the operating system, services, and other applications, developers are advised to limit their applications to only using the features of the SDK that they need.

Application optimizations include the following:

- Enable decoding only for the barcode types the application needs to scan.The cmbSDK supports the decoding of almost 40 different barcode types and subtypes, and while you can enable all of these, it can negatively impact performance and introduce unexpected errors
- More enabled symbologies can lead to slower performance and sluggish decoding, leaving the user with an inaccurate impression of the SDK’s capabilities.
- False reads are possible, especially when some of the weaker symbologies, like Code 25, are enabled without proper consideration and configuration of more advanced features. Advanced features like minimum code length and barcode location reduce false reads with the weak symbologies, but they also reduce performance, as they are not intended to be enabled and used at the same time.
- Using an optimal camera resolution. By default, the cmbSDK uses HD images (typically 1280x720) for barcode decoding. This resolution is sufficient for most barcodes, unless they are very small or dense. You can use a higher, full HD resolution, but these images are significantly larger, and thus require more time to analyze and decode.
- Using an appropriate decoder effort level.The SDK has a user-configurable effort level that controls how aggressively the SDK performs image analysis. Like most other settings, the SDK uses a default value (level 2) that is sufficient for most barcodes. Using a higher level can result in better decoding of poorer quality barcodes, but at the price of slower performance.

For these reasons, when the cmbSDK is initialized for use with the the mobile device's built-in camera, no barcode symbologies are enabled by default, and the application must only enable the symbologies it needs. As most barcode scanning applications only need to scan a handful of symbologies, this behavior encourages the developer to use the SDK in an efficient manner.