# ZBar bar code reader



## Why another bar code reader?

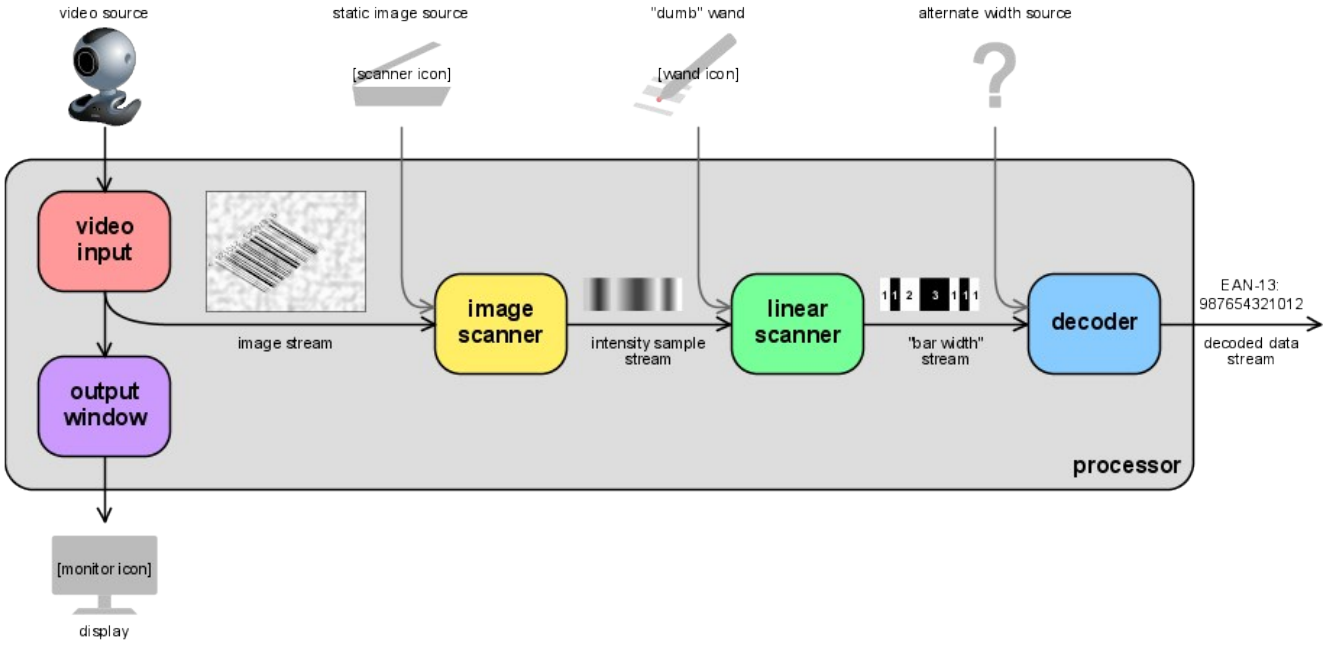
There are scores of commercial and shareware programs that read many different symbologies, but searching around a few years ago, the few open source readers I was able to find didn't meet my needs. This project aims to create a high performance, stable, robust library component with supporting infrastructure that makes it easy to use in a variety of applications. (and most importantly... "it sounded like fun at the time" :)

## How does it work?

A common design for a bar code "image scanner" is to apply digital image processing techniques to an image containing a bar code: exact details vary, but this usually involves several filter steps to cleanup noise, sharpen and enhance contrast, edge detection and shape analysis to determine symbol location and orientation, etc. Finally the data is extracted from this pristine image. All of these processing stages require CPU cycle and memory resources and are often sensitive to various "filter parameter" configurations which are difficult for end-users to understand and setup.

The ZBar library uses an approach closer to that used by "wand" and "laser" scanners: linear (1D) bar codes are designed to be decoded by a simple light sensor passing over the light and dark areas of a symbol. Taking advantage of this, the ZBar implementation makes linear scan passes over an image, treating each pixel as a sample from a single light sensor. The data is scanned, decoded and assembled on the fly.

Taking a cue from modern processing paradigms, ZBar further abstracts this idea into a layered streaming model. Processing is separated into independent layers with well defined interfaces, which can be used together or individually plugged into any other system. A high-level description of the modules is provided here:




video input
: Abstraction of a video device which produces a stream of images for scanning. The current release has interfaces to video4linux (versions 1 and 2). Support is in progress for VfW (Windows)

output window
: Simple abstraction of a display output window that can present a scanned image to the user and accept input in response. To maximize flexibility, the window may be opened and owned by the library, or attached to an application managed window embedded in a GUI. The current release supports basic X11 interfaces

(XVideo and XImage). Support is in progress for VfW (Windows).

image scanner
: Makes scan passes over a two-dimensional image to produce a linear stream of intensity samples. The input images may come from the video input module, or any external image source (such as an image file output by a flatbed scanner or digital camera). This module also incorporates the optional inter-frame consistency heuristic applied to a video stream.

linear scanner
: Scans a stream of abstract intensity samples to produce a "bar width" stream. The intensity samples could be pixel values from the built-in image scanner, pixel values from an alternate external image scanner, or even raw sensor samples from a "decoder-less" wand or laser sensor. The bars are detected and measured by applying some very basic 1D signal processing to the input sample stream

decoder
: The decoder searches a stream of bar widths for recognizable patterns and produces a stream of completely decoded symbol data. The current release implements decoding for EAN-13, UPC-A, UPC-E, EAN-8, Code 128 and Code 39 symbologies. Support is planned eventually for PDF-417 and EAN/UPC add-ons

processor
: One potential drawback of a completely independent modular approach is that it can take some coding to tie all of the modules together, complicating even simple applications. The high-level "processor" module connects all of the other modules to flexibly support many common uses. For example, this makes it easy to pop up a window (or not) and scan for bar codes from video or image sources with very few lines of code. The included sample applications: `zbarcam` and `zbarimg` are two examples of how this can be done (UTSL).

widgets
: For applications which already have a GUI, it does not always make sense to open a separate window for reading bar codes. To facilitate tighter integration between the reader and an existing GUI, the library also comes with ready-made "widgets" for various popular toolkits (currently Qt4, GTK+-2.0 and PyGTK2). The test programs in the distribution are good examples of how you can use these widgets to quickly incorporate a bar code reader widget into your application.

language interfaces
: With the performance sensitive image processing done in C, library wrappers for Perl (currently available) and Python make building a bar code application fast and easy!

plugin
: Support has been started for a NPAPI (Mozilla) plugin, which is supported by popular linux browsers and OpenOffice.org (among others).

*spadix@users.sourceforge.net*



Last modified: Fri Jul 15 11:18:33 PDT 2011