ご購入はこちら

プロも使うYocto開発環境で初体験！

ダウンロード・データあります

ラズパイ時代のレベルアップ！ Myオリジナル Linuxの作り方

第35回　アプリをYoctoに組み込むときのエラーに出ないポイント

三ツ木 祐介

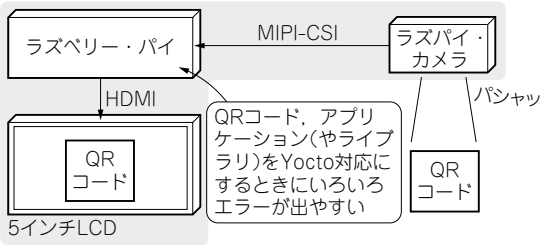

図1　提供されているパッケージ・ソフトウェア（ライブラリ/アプリケーション）**をYoctoに組み込むテクニックをQRコード読み取り実験で試す**

カメラで撮影したQRコードをオープンソース・ライブラリzbarcamで解析する．再掲

写真1　ラズベリー・パイでzbarcamを動かしてQRコード識別する実験で試す

パッケージ（今回はzbarcam）を含めてYoctoでLinuxを作成できるようになるのが今回からのキモ．再掲

ライブラリやアプリケーションなどのパッケージ・ソフトウェアをYoctoで生成するLinuxに組み込むときに起こる問題やその対策を，ラズベリー・パイ（Raspberry Pi）を使ったQRコード読み取り実験を題材にして紹介しています（**図1**，**写真1**）．

前回までで，QRコード読み取りソフトウェアzbar（zbarcam）を組み込んで，ラズベリー・パイ上で動作するLinuxをYoctoでビルドしたときのエラーを解析して対策を行いました．

さらに確認ポイントとして，エラーには現れない実機性能の低下やその対策を紹介しておきます．

これできちんとライブラリ/アプリケーションをYoctoで生成するLinuxに組み込むことができるようになります．

実機の処理性能についての問題＆対策

● 大量ログ出力による性能低下

ここまでの修正でzbarcamが作成されるようになり，実機上でも実行できるようになりました．

実際に動かしてみると，ウィンドウに本来表示されるはずのカメラからの映像が表示されずに，zbarのロゴマークだけが表示されました．

UARTコンソールから実行すると分かるのですが，実はデバッグ用のログが大量に出力されており，それが処理性能を大幅に低下させています（**リスト1**）．

ソースコードをgrepしてログを出力しているところを探すと，dprintf関数によって出力されていそうです．

```
$ grep -r 'scan:' .↵
./zbar/scanner.c:  dprintf(1, "scan
    : x=%d y=%d y0=%d y1=%d y2=%d",
```

さらにzbarのdprintfの実装を探してみるとデバッグ・プリント用の関数のようです．

```
$ grep -r 'dprintf' .↵
…（省略）…
./zbar/debug.h:#  define dprintf
                              (args...)
./zbar/debug.h:#  define dprintf(...)
./zbar/debug.h:#  define dprintf
          (level, format, args...) ¥
./zbar/debug.h:#  define dprintf
          (level, format, ...)     ¥
…（省略）…
```

debug.hの内容を確認してみます（**リスト2**）．

DEBUG_LEVELの定義がない場合は空の実装にな

第26回　NASの設定①…パーティションの作成（2019年3月号）
第27回　NASの設定②…ファイル共有してアクセスしてみる（2019年4月号）
第28回　ラズベリー・パイに日本語をしゃべらせる（2019年5月号）