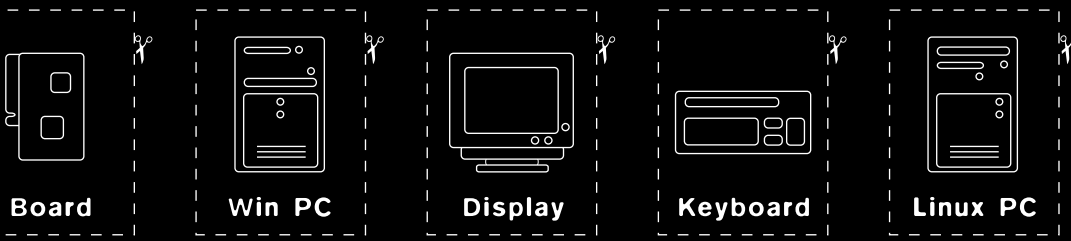




Special Linuxで電子工作シリーズ

LinuxでLEGO (後編)



有限会社ハンブルソフト
成松宏

本誌2001年3月号に掲載した「LinuxでLEGO(前編)」はいかがだったでしょうか。前回の記事では、LEGO MINDSTORMSをLinuxマシンから動かす方法を紹介しました。今回は、「nqc」を使ってLEGOを制御する方法を説明し、Web経由でカメラの向きを変えたり、画像も見られるというシステムを紹介します(写真1)。

nqcとは?

MINDSTORMSには、RCXのプログラムを行う環境として、Window上で動作するプログラム(画面1)が標準で付いてきます。解説アニメーションがあったり、ブロックを組み立てるようにプログラムを組めたりとなかなか楽しいのですが、毎度使用するにはちょっとツライものがあります。

そのため、このソフトに代わるものとして、「nqc」というアプリケーションが開発されています。これは、Dave Baum氏が開発したRCXをプログラムするためのソフトウェアで、Windows、Macintosh、UNIXなどで動作します(原稿執筆時点でのバージョンはVer.2.2r2)。nqcは「Cに似ているけどCじゃない(Not Quite C)」という言語で、RCXのバイトコードへのコンパイルを行えます。またコンパイラ機能の他に、RCXへのダウンロード機能なども備えています。nqcで得られる大きなメリットは、Emacsのc-modeでプログラムを編集してコマンドラインからダウンロードできることで、普通の感覚(?)でRCXのプログラムを組めるようになります。

- ・nqcの文法
すでに触れたように、nqcは「Not Quite C」という言語なの

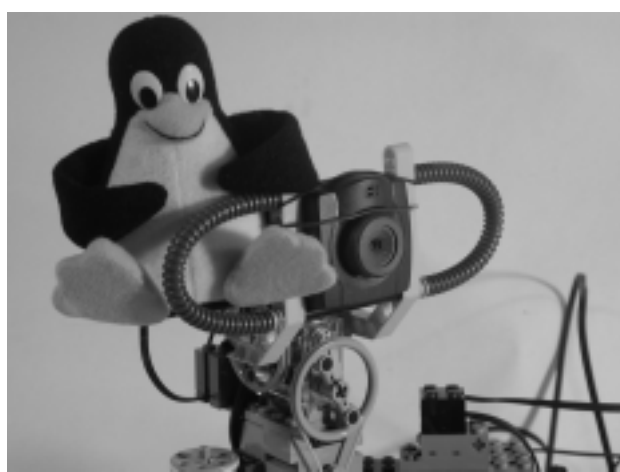


写真1 LEGO Controlled Camera



画面1 LEGO RIS付属のプログラム環境

で、C言語と異なる部分のみを列挙します(表1)。またインターネット上にさまざまな文書が存在し、日本語に翻訳された文書としては、「日本語nqcユーザズマニュアル(記事末のRESOURCE[1]を参照)」「nqcチュートリアル([2])」があります。英文では、nqcのWebページ([3])にnqcのマニュアル([4])とプログラミングガイド([5])のPDFファイルがあります。

・nqcのインストール

詳しいインストール方法は前編で書きましたので、そちらを参照してください。nqcのWebページからソースをダウンロードし、展開後makeを実行するだけです。

・nqcのプログラム例：sample.nqc

nqcで組んだ、モーターAを回転させるプログラム例を示し

表1 nqcとC言語の異なる部分

・使える型は整数のみ
・演算は代入演算(*=などのみ) ただし定数式はその限りではない。
・制御構造はif、while、repeatなど
・taskがあり、並行実行可能

リスト1 sample.nqc

```
1: task main()
2: {
3:   int t;
4:   t = 0;
5:   OnFwd(OUT_A);
6:   while(t<8){
7:     SetPower(OUT_A,t);
8:     PlaySound(SOUND_CLICK);
9:     Wait(100);
10:    Toggle(OUT_A);
11:    t++;
12:   }
13:   Float(OUT_A);
14: }
```

ます(リスト1)。内容は表2の通りです。

このプログラムは、nqcプログラムを使ってコンパイルし、ダウンロード後実行させることができます。また-Lオプションを付けることで、RCXのバイトコードへのコンパイル結果を見ることもできます(実行例1)。

LEGO VISION COMMAND

LEGOに「VISION COMMAND(写真2)という製品があります。これにはレゴブロックの突起がついたCMOSカメラ(写真3)と、カメラをコントロールするのに便利なLEGOブロックと、カメラコントロール用のWindowsソフトが入っています。カメラは5mのUSBケーブルでパソコンと接続されます。

実行例1 sample.nqcのコンパイルとダウンロード

```
$ nqc -d sample.nqc -run
Downloading Program: ...complete
Battery Level = 9.8 V
$ nqc -L sample.nqc

*** Task 0 = main
000 pwr   ABC, 7           13 07 02 07
004 dir   ABC, Fwd       e1 87
006 setv  var[0], 0      14 00 02 00 00
011 dir   A, Fwd        e1 81
013 out   A, On         21 81
015 jmp1  35            72 13 00
018 pwr   A, var[0]     13 01 00 00
022 plays 0             51 00
024 wait  100          43 02 64 00
028 dir   A, Flip       e1 41
030 sumv  var[0], 1     24 00 02 01 00
035 chk1  7 >= var[0], 18 95 42 00 07 00 00 e9 ff
043 out   A, Float     21 01

Total size: 45 bytes
```

表2 リスト1(sample.nqc)の解説

行数	説明
1行目	main taskを定義します。プログラムをダウンロード後RCXのRUNボタンを押すなどして実行させると、main taskが走ります。
3行目~4行目	変数tを宣言し、0を代入しています。
5行目	RCXのA端子に接続されたモーターをオンにし、順方向に回転させます。モーターの回転する方向は、電極ブロックの取り付け方向によって変わるので注意してください。
6行目	while文です。以下のブロックをtが8よりも小さい間繰り返します。
7行目	モーターを駆動する力を0~7の範囲で指定できます。0が最弱、7が最強になります。ここではAのモーターの力を変数tの値に設定しています。
8行目	クリック音を鳴らします。6種類の音を指定できます。
9行目	Waitで1/100秒単位で待つことができます。ここでは1秒待っています。
10行目	モーターAの回転の方向を逆転させます。
11行目	tの値を1つ増やします。
13行目	モーターAを止めます。止め方にはOffとFloatの2種類があります。Offだとブレーキがかかるように止まり、Floatは多少惰性で回転します。



写真2 VISION COMMAND

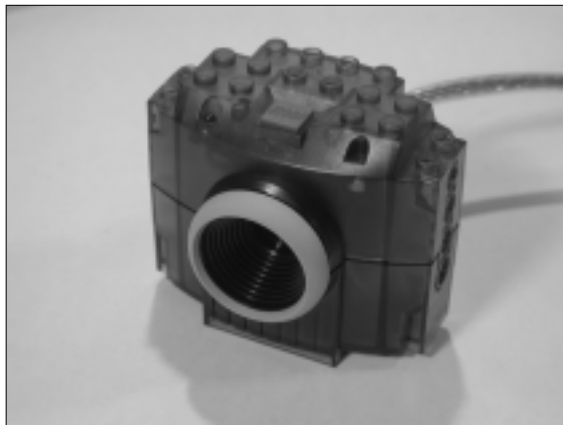


写真3 VISION COMMANDのカメラ

これをLinuxで使用できれば便利ですので、早速入手して調べてみました。「カーネル2.2.17」+「USB BACKPORT PATCH」でUSBを使用できるLinuxマシンを作成し、VISION COMMANDのUSBコネクタを差し込むと、logの情報から「vendor:Product」コードが「046d:0850」であることが分かりました。このコードを、「Linux USB プロジェクト」のWebページ[6]にある「USB Vendor/Device IDs list」で調べると、ベンダーはLogitech社、Productは該当なしと分かりました。Logitech社の「QuickCam Express」のProductコードが0840なので、同じような製品なのかもしれません。そこで、QuickCam Expressのドライバー-[7]を試してみましたが、残念ながらうまくいきませんでした。

CCDカメラとbttvドライバー

結局、VISION COMMANDのカメラの使用はあきらめ、部品だけ使うことにしました。カメラは以前購入していたCCDカメラ(写真4)を使用します。このカメラはビデオ信号を出力

するので、それをTVチューナー付きキャプチャーカードで読み取ります。キャプチャーカードはLinux上でTVを見るために以前購入したものを使用しました。現在では8000円程度で購入可能です。bt878チップを使っているので、bttvドライバで問題なく動作します。bttvドライバは、最近のディストリビューションでは最初から含まれている場合が多いと思います。うまくいかない場合は、bttvのWebページ[8]から新しいドライバをダウンロードして試してみるといいでしょう。「xawtv」プログラムで、TVやカメラの出力を見ることが出来ます。詳しくはxawtvのWebページ[9]を参照してください。

カメラマウントを作る

ここから、LEGOでカメラを動かす仕組みを作りますが、LEGOで思ったような機能をもった機構を組むのは結構大変です。これがまた、見かけと違って簡単にはいきません。まず単純な仕組みで試して感じをつかみ、応用し、テスト後にさらに改良するといった手順が必要です。少なくとも私はそう



写真4 CCDカメラ

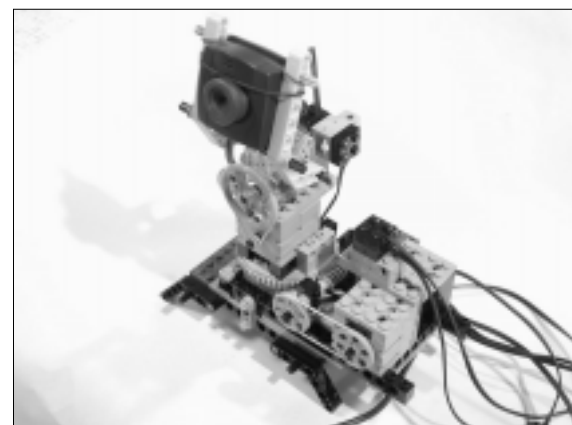


写真5 作成したカメラマウント

で、時間がかかります(まあ、そこも面白いところではありません)完成したカメラマウントを写真5に示します。モーター2個、ロータリーセンサを2個、タッチスイッチ2個のほか、VISION COMMANDのギアボックスやターンテーブルを使用しています。

タッチセンサは、角度の基準位置を知るのに使います。ロータリーセンサは回転量を1/16回転単位で返します。また、電源投入後に基準の位置を覚えてやらないとカメラの方向が分かりません。センサを4個使用していますので、タッチスイッチは並列に接続して「OR」をとった状態にしています。これらは、nqcのチュートリアルで知ったテクニックです。

タッチセンサの取り付け方には最後まで苦労しました。ターンテーブルが回転していくと、取り付けたプーリーが写真6のタッチセンサ1を押します。最初、プーリーだけでは厚みが不足し、センサに検知されませんでした。プーリーにLEGOに付属するゴムの輪を巻いてやっとセンサに検出されるようになりました。センサの値は、RCXでVIEWボタンを押すことで確認できます。リモコンとVIEWボタンでの表示を利用

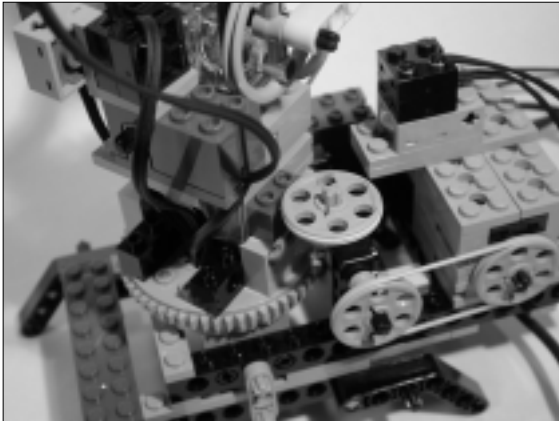


写真6 タッチセンサ1

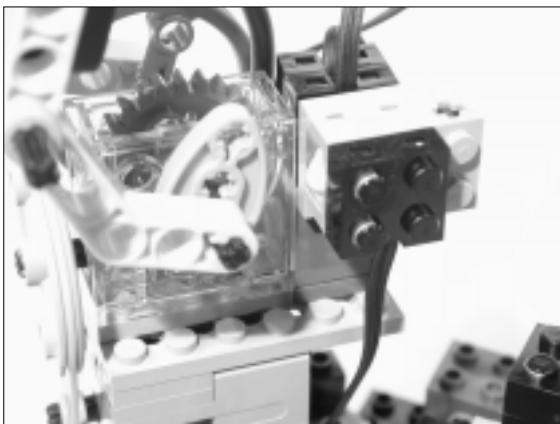


写真7 タッチセンサ2

すると、センサの調整が楽に行えます。

カメラの上下方向の回転軸が回転すると、軸に取り付けたカムが写真7のタッチセンサ2を押します。カメラの向きはy軸まわりで360度、上下方向で110度程度動かせます。しかし、角度を大きく変えると配線の処理が大変で、配線が外れたり絡まったりします。今回使用したターンテーブルは回転軸が開いていて配線を通せるので、その点でかなり有利です。

nqcプログラム

リスト2に、カメラマウントをコントロールするnqcプログラムを示します。プログラムは3つのtaskから構成されています。「task main」はセンサの設定と初期化を行います。「task servo_x」と「servo_y」はモーターを指定した位置まで動かすtaskで、Perlプログラムから直接起動されます。詳しい内容説明は表3を参照してください。

このプログラムが用意できたら、以下のコマンドでコンパイルとダウンロードができます。

```
$ nqc -d camera.nqc
```

プログラムに誤りがある場合には、エラーメッセージとエラーのあった位置が表示されますので修正してください。ダウンロードできたら、RCXのRunボタンを押すとカメラが位置合わせを行い止まります。

また、下のようにnqcに「-run」オプションをつけてやると、プログラムをダウンロード後すぐに走らせてくれますので、プログラム開発中は便利です。

```
$ nqc -d camera.nqc -run
```

カメラの向きを変えるPerlプログラム

カメラの向きを変えるにはリスト3のPerlプログラムを使用します。このプログラムの内容は表4の通りです。

プログラム内容を確認してもらったところで、使い方を説明しましょう。まずRCXにリスト2(camera.nqc)をダウンロードして実行させておきます。そして、以下のように指定するとカメラマウントが動きます。

```
$ perl camera.pl -x 400
$ perl camera.pl -y -20
$ perl camera.pl -x 650 -y -110
```

カメラの画像は、とりあえずxawtvで見ると良いでしょ

リスト2 camera.nqc

```

1: #define Motor_X    OUT_A
2: #define Motor_Y OUT_B
3: #define Rotate_X  SENSOR_1 // 回転センサ
4: #define Rotate_Y  SENSOR_2 // 回転センサ
5: #define Datums    SENSOR_3 // タッチセンサ
6: #define X_PA1    (56*16/2)
7: #define Y_PA1    (24*16/2)
8:
9: int xt,yt; // 目標値
10:
11: task main()
12: {
13:   SetSensor(Datums, SENSOR_TOUCH);
14:   SetDirection(Motor_X+Motor_Y,OUT_REV);
15:   SetPower(Motor_X+Motor_Y,3);
16:   On(Motor_X+Motor_Y);
17:   Wait(50);
18:   Off(Motor_X+Motor_Y);
19:
20:   SetDirection(Motor_X, OUT_FWD);
21:   On(Motor_X);
22:   until(Datums == 1);
23:   Off(Motor_X);
24:   SetSensor(Rotate_X, SENSOR_ROTATION);
25:
26:   SetPower(Motor_X, 7);
27:   SetDirection(Motor_X, OUT_REV);
28:   On(Motor_X);
29:   until(Rotate_X > X_PA1);
30:   Off(Motor_X);
31:
32:   SetDirection(Motor_Y, OUT_FWD);
33:   On(Motor_Y);
34:   until(Datums ==1);
35:   Off(Motor_Y);

```

リスト2 続き

```

36:   SetSensor(Rotate_Y, SENSOR_ROTATION);
37:
38:   SetPower(Motor_Y, 7);
39:   SetDirection(Motor_Y, OUT_REV);
40:   On(Motor_Y);
41:   while(Rotate_Y > -Y_PA1/2);
42:   Off(Motor_Y);
43: }
44:
45: task servo_x ()
46: {
47:   int x,dx;
48:   x = Rotate_X;
49:   On(Motor_X);
50:   while(xt != x){
51:     dx = xt; dx -= x;
52:     if(dx > 0) SetDirection(Motor_X, OUT_REV);
53:     else      SetDirection(Motor_X, OUT_FWD);
54:     x = Rotate_X;
55:   }
56:   Off(Motor_X);
57: }
58:
59: task servo_y ()
60: {
61:   int y,dy;
62:   y = Rotate_Y;
63:   On(Motor_Y);
64:   while(yt != y){
65:     dy = yt; dy -= y;
66:     if(dy > 0) SetDirection(Motor_Y, OUT_FWD);
67:     else      SetDirection(Motor_Y, OUT_REV);
68:     y = Rotate_Y;
69:   }

```

表3 リスト2(camera.nqc)の解説

行数	説明
1行目～5行目	モーターとセンサの接続を定義しています。RCXのA端子にカメラを横方向に動かすモーター、B端子に縦方向を動かすモーターを接続します。センサ1の端子に横方向の回転センサ、センサ2の端子に縦方向の回転センサ、センサ3の端子に2つのタッチセンサを並列に接続します。
6行目～7行目	縦方向、横方向の回転センサの180度分にあたるカウント値を定義しています。歯車の歯の数が横方向が56枚、縦方向24枚、回転センサは1回転で16カウント進むので、このような計算になります。
9行目	モーターを動かす目標値を格納する変数を定義しています。この位置に定義すると、nqcによって変数0番と1番に割り当てられます。実際にどの変数に割り当てられたか調べるには、 \$ nqc -L camera.nqc として出力されるコンパイル結果を見ることで確認できます。この変数への値の設定はPerlプログラムで行います。
13行目	端子3につながっているセンサが、タッチセンサであることをRCXに教えます。
14行目～18行目	この後、2つのモーターを各タッチセンサがOnになるまで動かし、センサをリセット、適当な位置まで戻すという操作をします。しかし、最初の状態で2つのタッチセンサが両方ともOnになっているとうまくないので、0.5秒だけモーターを動かし、位置をずらします。この動作によって偶然2つのタッチセンサがOnになってしまう可能性もないわけではないのですが、そこまでは対応していません。
20行目～24行目	横方向のモーターをタッチセンサがOnになるまで動かし、センサの1番端子につながっているのが回転センサであるという設定を行っています。この設定により、回転センサーのカウント値が0にセットされます。
26行目～30行目	横方向のモーターを、ターンテーブルが180度回転するまで逆方向に回しています。
32行目～35行目	縦方向のモーターを、タッチセンサがOnになるまで回しています。
38行目～42行目	縦方向のモーターを90度回しています。
45行目～56行目	横方向の回転センサの値が、変数xtに格納された目標値と等しくなるまでモーターを動かします。モーターの向きは、回転センサの値と目標値xtの値の大小関係によって反転します。モーターのコネクタの向きが違っていると、モーターの回転方向が逆になりますので、その場合は52行目と53行目の「OUT_REV」と「OUT_FWD」を逆にしてください。いい加減なプログラムですが、RCXの動作が速いせいか、ちゃんと止まってくれます。
59行目～71行目	縦方向について、45行目～56行目と同様のことを行います。

表4 リスト3(camera.pl)の解説

行数	説明
2行目	2001年3月号(46ページ)で紹介した「legoLib.pl」ライブラリを読み込んでいます。
3行目~9行目	コマンドライン引数の処理を行っています。コマンドラインでは、「-x」 「-y」オプションに続いて、横方向と縦方向の目標値を与えます。横方向の可動範囲は360度ですので、値としては0~896となります。縦方向は110度程度ですので、0~120程度です。
11行目	legoLib.plで定義されているtrx関数を使って、横方向の目標値をRCXの変数0に書き込んでいます。
12行目	「task 1」、すなわちservo_xを起動しています。
15行目~16行目	11行目同様にRCXの変数1に縦方向の目標値を書き込み、「task 2」にあたる「servo_y」を起動しています。

リスト3 camera.pl

```

1: #!/usr/bin/perl
2: do 'legoLib.pl';
3: $usage = "Usage:$0 [-x num] [-y num]\n";
4: $#ARGV < 0 && die $usage;
5: while($_ = shift){
6:     if(/^~x$/) { $x = shift;}
7:     elsif(/^~y$/){ $y = shift;}
8:     else { die "Bad arg:$_\n$usage";}
9: }
10: if(defined($x)){
11:     &trx(1,0x14,0, 2, $x & 0xff , $x >> 8);
12:     &trx(1,0x71, 1); # start task 1
13: }
14: if(defined($y)){
15:     &trx(1,0x14,1, 2, $y & 0xff , $y >> 8);
16:     &trx(1,0x71, 2); # start task 2
17: }

```

う。今回、リスト3では可動範囲のチェックを行っていないので、変な値を指定するとブロックが壊れそうになります。この点に気を付けて使ってください。または、独自に範囲のチェック機能を付けるのも良いでしょう。

ビデオ画像をwebに表示する

今回は、画像をWebページ上で見られるようにしたいので、そのための仕組みが必要になります。これは、カメラの画像を一旦ファイルに書き込み、そのファイルをCGIプログラムで出力すれば実現できます。「w3cam」というプログラムを使用すると、これらの作業が全部できてしまうらしいと聞いたので、今回はw3camを使用してみることにしました。

w3camのインストール

w3camは、Webページ上でvideo4linux対応デバイスの画像を見るためのCGIプログラムで、Apacheなどのhttpdサーバと組み合わせて使用します。

w3camのWebページ[10]から、「w3cam-0.6.6.tar.gz」をダウンロードします。以下のように適当なディレクトリに

リスト4 w3cam.cgi.scfに追加する設定

```

norm="1"          # NTSC
input="1"         # Composite1
format="2"        # jpeg
quality="60"      # jpeg quality
width=320
height=240

```

展開し、configure後、makeで完成です。

```

$ cd 作業用ディレクトリ
$ tar xovfz w3cam-0.6.6.tar.gz
$ cd w3cam
$ ./configure
$ make

```

この作業でできたプログラム「w3cam.cgi」と、設定ファイル「w3cam.cgi.scf」をCGI用のディレクトリにコピーします。ここでは共用のディレクトリにコピーしました。

```

$ su
Password:
# cp w3cam.cgi w3cam.cgi.scf /home/httpd/cgi-bin
# exit

```

w3cam.cgi.scfの設定

w3cam.cgi.scfにリスト4の設定を追加します。これらは、すべてw3cam.cgiの呼出時に指定できますが、ここで設定しておけばそれを省略できます。

httpdサーバの設定

w3camを動作させるには、ApacheなどのHTTPサーバが必要です。最近のほとんどのLinuxディストリビューションは、インストール時にHTTPサーバが動いている状態にできると思

表5 httpdの設定ファイル(LASER5 Linux 6.0の場合)

ファイル名	ディレクトリ	説明
httpd.conf	/etc/httpd	httpd設定ファイル
access.conf	/etc/httpd	httpd設定ファイル
srm.conf	/etc/httpd	httpd設定ファイル
access_log	/var/log/httpd	アクセスログ
error_log	/var/log/httpd	エラーログ

います。また動いているかどうかは、以下のコマンドで確認できます。

```
$ ps ax | grep httpd
```

これでhttpdというプロセスが表示されれば、Apacheサーバが動いています。動いていなければ、設定を行い動かすようにしましょう。

以下、LASER5 Linux 6.0での設定方法を説明していきます。他のディストリビューションでは異なる点があるかもしれませんが、ご了承ください。

以降の作業を行うために、Apache関連のファイルの場所を確認しておきます。表5に設定ファイルの場所を示します。このような設定ファイルは、locateコマンドで簡単に探すことができます。locateコマンドが動作しなかったり、データが古いと文句を言う場合には、rootになって

```
/etc/cron.daily/slocate.cron
```

を実行しておきましょう。

httpdとCGIを使えるようにする

w3cam.cgiとw3cam.cgi.scfのコピー先が、/home/httpd/cgi-binであれば、デフォルトでCGIが動作する設定になっている場合が多いと思います。そうならない場合、あるいは他のディレクトリにw3web.cgiをインストールする場合には、次のような指定がaccess.confにあることを確かめてください。

```
<Directory /home/httpd/cgi-bin>
Options ExecCGI
</Directory>
```

ついでに、srm.confで以下の設定が有効になっているか、確かめておきましょう。

```
# To use CGI scripts:
AddHandler cgi-script .cgi
```

w3cam.cgiを呼び出すhtml記述

リスト5に、w3cam.cgiを使用するためのHTMLファイル、「w3cam.html」の内容を示します。このファイルはメニューになっており、5つの異なるパラメータでw3cam.cgiを呼び出します(表6)。また画面2に、w3cam.cgiの出力をNetscapeブラウザで表示させた画面を示します。

リスト5 w3cam.html

```
1: <html>
2: <head>
3: <title>W3Cam Test</title>
4: </head>
5: <body bgcolor="#ffffff">
6: <h1>W3Camのテスト</h1>
7: <ul>
8: <li> <a href="/cgi-bin/w3cam.cgi?help">help</a>
9: <li> <a href="/cgi-bin/w3cam.cgi?quality=3">
10:   video(quality:3)</a>
11: <li> <a href="/cgi-bin/w3cam.cgi?quality=30">
12:   video(quality:30)</a>
13: <li> <a href="/cgi-bin/w3cam.cgi?quality=100">
14:   video(quality:100)</a>
15: <li> <a href="/cgi-bin/w3cam.cgi?quality=60&refresh=0.5">
16:   video(quality:60 refresh=0.1)</a>
17: <li> <a href="/cgi-bin/w3cam.cgi?mode=gui">
18:   GUI</a>
19: </ul>
20: </body>
21: </html>
```

表6 リスト5(w3cam.html)の解説

行数	説明
8行目	helpを指定すると、w3cam.cgiの使用方法が表示されます。
9行目、11行目、13行目	「quality=数字」というパラメータで、出力するJPEGファイルの画質を指定できます。値を大きくすると画質がよくなりますが、ファイルサイズも大きくなります。値を小さくすると、ファイルサイズが小さくなりますが、画質も落ちます。最大は100です。
15行目	「refresh=秒」という指定で再表示を指示できます。リスト5では、0.5秒おきに画像が更新されます。また更新されている間、ブラウザにはデータが転送され続けます。更新を止めるには、ブラウザの停止(STOP)ボタンを押してください。
17行目	「mode=gui」という指定をすると、w3camの設定変更用のGUI付きで表示されるようになります。最初はこれいろいろ試してみるのもいいでしょう。



画面2 w3cam.cgi?mode=guiの出力

/dev/video0のパーミッション

w3camを使用すると、「/dev/video0がopenできない」というエラーが/var/log/httpd/error_logに出ます。httpdが/dev/video0を読み出せないのが当然です。rootになって、

```
chmod 666 /dev/video0
```

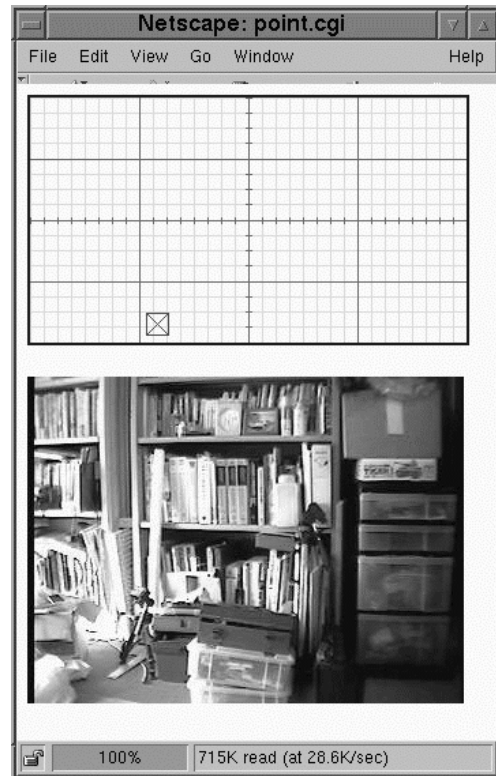
とコンソール上でコマンドを打ち込むと見られるようになります。しかし、Linuxを再起動後に試すとまた同じエラーが出てしまいます。「何が/dev/video0をいじってるんだらう?」と思い、

```
$ find /etc -type f |xargs grep /dev/video
```

と打って確認すると、「/etc/security/console.perms」というファイルが見つかりました。これは、ログイン時にコンソール関係のファイルのパーミッションを設定し、ログアウト時に戻すという機能の設定ファイルのようです。詳しくは「man console.perms」をご覧ください。

そういえば、/dev/video0のオーナーが自分になっているのが不思議ではありましたが。このファイルの<v41>の行を以下のように修正してやると、いつでもw3camが動くようになります。

```
#<console> 0600 <v41>          0600 root
<console> 0666 <v41>          0666 root
```



画面3 カメラをコントロールできるWebページ

仕上げ: Webページを作る

必要な材料が揃いましたので、いよいよ仕上げです。カメラの向きをコントロールできて、画像も見ることができるWebページを作ります。完成したページを画面3に示します。このページの構成と動作は図1の通りです。このページは、

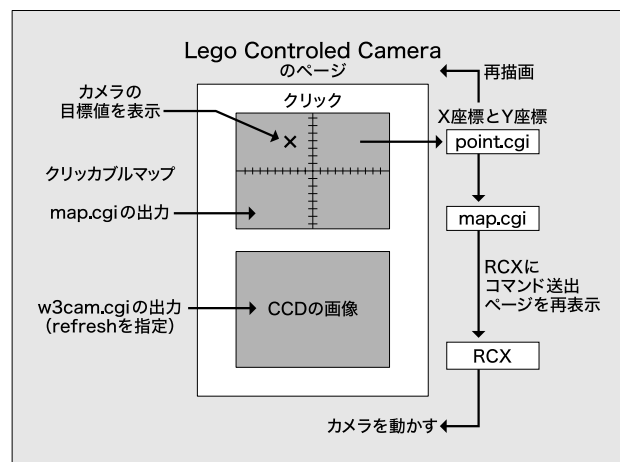


図1 Webページ構成と動作

リスト6 point.cgi

```

1: #!/usr/bin/perl
2:
3: do "cgiLib.pl";
4: &parse_from_data(data);
5: &out_html;
6: exit 0;
7:
8: sub out_html {
9:     $x = $data{'x'};
10:    $y = $data{'y'};
11:    &pr('Content-Type: text/html', "\n",
12:      '<html><head>',
13:      '<title>point.cgi</title>',
14:      '</head>',
15:      '<body bgcolor="white">',
16:      '<form action="point.cgi"',
17:      ' method="get">',
18:      '<input type="image" ',
19:      sprintf('src="map.cgi%s">',
20:        defined($x) ? "?x=$x&y=$y" : ""),
21:      '</form>',
22:      '',
23:      '</body></html>');
24: }

```

グラフィックライブラリ「GD」を使って生成したクリックابلマップと、w3cam.cgiの画像からなっています。上部の方眼紙のような部分がクリックابلマップになっていて、ここをクリックしてやると、その作業を持ってpoint.cgiというCGIプログラムが呼び出されます(リスト6、表7)。point.cgiは、クリックされた座標を変換してRCXにコマンドを送り、再び同じページを表示します。このページのHTMLファイル、すなわちpoint.cgiが出力するHTMLファイルは、リスト7のようになります。表8もあわせて参照してください。

map.cgiの説明

map.cgiはクリックابلマップのイメージを描画します。いろいろ凝ったイメージが考えられますが、今回は簡単に方眼紙風の模様と、カメラの向きの目標値を表示させることにしました。リスト8に示します。

cgiLib.plの説明

cgiLib.pl(リスト9)は、CGIで良く使用する2つのサブルーチンを定義しています。parse_from_dataは、CGIに渡されたパラメータを環境変数に読み込むものです。prは引数に改行「\n」を付加して出力するだけのものです。HTMLファイルには引用符(「」)がたくさん含まれるので結構便利です。

リスト7 point.cgiが出力するhtml

```

1: <html>
2: <head>
3: <title>Lego Controlled Camera</title>
4: </head>
5: <body bgcolor="white">
6: <form action="point.cgi" method="get">
7: <input type="image" src="map.cgi?x=0&y=0">
8: </form>
9: 
10: </body>
11: </html>

```

表7 リスト6(point.cgi)の解説

行数	説明
3行目	ライブラリcgiLib.plを読み込んでいます。cgiLib.plではparse_from_dataとprの2つのサブルーチンを定義しています。リスト9(cgiLib.pl)を参照してください。
4行目	parse_from_dataでCGIに渡されたパラメータを、連想配列\$dataに読み込んでいます。
9行目~10行目	CGIに渡されたxとyパラメータを取り出しています。パラメータはmap.cgiから渡されます。
11行目~23行目	HTMLファイルを出力します。サブルーチンprは引数のリストを改行コードをはさみながら出力します。HTML文書では引用符(「」)を使用することが多い上に、改行も入れないと見にくいのでこのようなサブルーチンを使用しています。
19行目~20行目	クリックابلマップのイメージソースとして、map.cgiを指定していますが、point.cgiにxとyのパラメータを渡された場合、それをmap.cgiにも渡しています。map.cgiはパラメータで目標位置の描画を行います。

表8 リスト7の解説(point.cgiが出力するhtmlファイル)

行数	説明
6行目~8行目	クリックابلマップを定義しています。クリックابلマップに貼り付けるイメージは画像ファイルでもいいのですが、カメラが向いている方向も表示したかったので、map.cgiでGDを用いて動的に生成しています。7行目の「?x=0&y=0」の部分は実際にはクリックされた座標が入ります。
9行目	w3cam.cgiを呼び出しています。「refresh=0.1」で0.1秒ごとの更新を指定しています。ただし、実際には通信速度がネックで、そこまで速く更新されていないようです。

リスト8 map.cgi

```

1:  #!/usr/bin/perl
2:
3:  use GD;
4:  do "legoLib.pl";
5:  do "cgiLib.pl";
6:
7:  $w = 320;
8:  $h = 180;
9:  $xmax = 896;
10: $ymax = 130;
11: &parse_from_data(data);
12: $tx = $data{'x'};
13: $ty = $data{'y'};
14: &cameraMove($tx,$ty) if defined($tx);
15: &html_out;
16: exit 0;
17:
18: sub html_out {
19:     $im = new GD::Image($w,$h);
20:     $white = $im->colorAllocate(255,255,255);
21:     $blue = $im->colorAllocate( 0,155,150);
22:     $g0 = 220;
23:     $gray = $im->colorAllocate($g0,$g0,$g0);
24:     $red = $im->colorAllocate(255, 0, 0);
25:     $im->filledRectangle(0,0,$w-1,$h-1,$white);
26:     $a=2;
27:     $d=32;
28:     for($i=0;$i<$d;$i++){
29:         $x = $i*$w/$d;
30:         $im->line($x,0,$x,$h, $gray);
31:         $im->line($x,$h/2-$a,$x,$h/2+$a, $blue);
32:     }
33:     $d=16;
34:     for($i=0;$i<$d;$i++){
35:         $y = $i*$h/$d;
36:         $im->line(0,$y,$w,$y, $gray);
37:         $im->line($w/2-$a,$y,$w/2+$a,$y, $blue);
38:     }
39:     for($i=1;$i<4;$i++){
40:         $x = $i*$w/4;

```

リスト8 続き

```

41:     $y = $i*$h/4;
42:     $im->line($x,0,$x,$h, $blue);
43:     $im->line(0,$y,$w,$y, $blue);
44:     }
45:     $x = $data{'x'};
46:     $y = $data{'y'};
47:     if(defined($x) && defined($y)){
48:         $d = 8;$r = $d;
49:         $im->line($x-$d,$y-$d,$x+$d,$y+$d, $red);
50:         $im->line($x-$d,$y+$d,$x+$d,$y-$d, $red);
51:         $im->rectangle($x-$r,$y-$r,$x+$r,$y+$r,
52:             $red);
53:     }
54:     print "Content-Type: image/png\n\n";
55:     print $im->png;
56: }
57: sub cameraMove {
58:     local($tx,$ty) = @_;
59:     local($x) = $tx / $w * $xmax;
60:     local($y) = -$ty / $h * $ymax;
61:     &trx(1,0x14,0, 2, $x & 0xff, $x >> 8);
62:     &trx(1,0x71, 1); # start task 1
63:     &trx(1,0x14,1, 2, $y & 0xff, $y >> 8);
64:     &trx(1,0x71, 2); # start task 1
65: }

```

完成版を動かす

point.cgi、map.cgi、cgiLib.pl、legoLib.plを、同じWebブラウザでアクセスでき、CGIが実行できるディレクトリに配置してください。camera.nqcをRCXにダウンロードおよび実行後、Webブラウザでpoint.cgiにアクセスすると、画面3のようなページを見ることができるようになります。

表9 リスト8(map.cgi)の解説

行数	説明
3行目	GD.pmを使用しています。
4行目～5行目	ライブラリlegoLib.plとcgiLib.plを読み込んでいます。legoLib.plは前編(本誌2001年3月号)に掲載しております。
7行目～8行目	イメージの大きさを定義しています。
9行目～10行目	横方向と縦方向の最大可動範囲を定義しています。単位は回転センサの読み出し値です。横方向は360度(896単位)で、縦方向は3度(384単位)となります。縦方向は符号を反転してありますので、この場合横方向は360度、縦方向は121度まで動きます。
11行目	CGIに渡されたパラメータを連想配列\$dataに読み込んでいます。
12行目～14行目	パラメータxとyを読み出し、パラメータxが定義されている場合にはカメラを動かしています。
19行目	GDで指定したサイズのイメージを生成しています。
20行目～24行目	使用する色を確保しています。
25行目	描画を始める前にイメージ全体を白色で塗りつぶしています。
26行目～32行目	方眼紙の縦線と目盛を描画しています。
33行目～38行目	方眼紙の横線と目盛を描画しています。
39行目～44行目	方眼紙を縦横4分割する線を描画しています。
45行目～52行目	パラメータxとyが渡されている場合、その位置に印を描画しています。
53行目～54行目	作成したイメージデータを出力しています。
57行目～65行目	クリッカブルマップの座標系で渡された座標をカメラマウントの回転センサの座標に変換し、RCXへコマンドして送っています。リスト3(camera.pl)と同じですが、y座標は符号を反転させています。

完成

カメラの向きをコントロールできて、画像も見られるWebページは、これで完成です。自宅のLinux上でNetscapeブラウザを立ち上げて、カメラをコントロールしていると非常に楽しいです。小学生のころ(何年前だ?)、電気機器会社のショールームにリモコンで向きを変えられるカメラがあり、それで遊んでいたことを思い出しました。

今回は、Web経由でのコントロールを行いました。ローカルでよければ、Tcl/Tkなどでコントロールプログラムを作

成し、xawtvで画像を見れば、もっと素早い反応のプログラムが作れると思います。

なお、今回の実験(?)で、video4linuxを利用した面白い画像プログラムがたくさんあり、また簡単に作れるらしいことが分かりました。今後は、その方面でもいろいろ遊んでみるのも面白そうだと思っています。

リスト9 cgiLib.pl

```

1: sub parse_from_data
2: {
3:     local(*FORM_DATA) = @_;
4:     local($request_method,$query_string,
5:           @key_value_pairs,$key_value,$key,
6:           $value);
7:     $request_method = $ENV{'REQUEST_METHOD'};
8:     if($request_method eq "GET") {
9:         $query_string = $ENV{'QUERY_STRING'};
10:    } elsif ($request_method eq "POST") {
11:        read(STDIN,$query_string,
12:            $ENV{'CONTENT_LENGTH'});
13:    } else {
14:        die "Server uses unsupported method";
15:    }
16:    @key_value_pairs = split(/&/,$query_string);
17:    foreach $key_value (@key_value_pairs) {
18:        ($key,$value) = split(/=,$key_value);
19:        $value=~tr/+// ;
20:        $value=~
21:            s/%([\dA-Fa-f]{2})/pack("C",hex($1))/eg;
22:        if(defined($FORM_DATA{$key})) {
23:            $FORM_DATA{$key}=
24:                join("\0",$FORM_DATA{$key},$value);
25:        } else {
26:            $FORM_DATA{$key}=$value;
27:        }
28:    }
29: }
30:
31: sub pr
32: {
33:     foreach $line (@_) {
34:         print $line,"\n";
35:     }
36: }
37: 1;

```

R E S O U R C E

- [1] 日本語nqcユーザズマニュアル
<http://www.mi-ra-i.com/JinSato/MindStorms/nqc/nqcc-j.html>
- [2] nqcチュートリアル日本語訳
<http://www.cc.toin.ac.jp/EI/NQCj.html>
- [3] nqc Webページ
<http://www.enteract.com/~dbaum/nqc/index.html>
- [4] nqcマニュアル
http://www.enteract.com/~dbaum/nqc/doc/NQC_Manual.pdf
- [5] nqcプログラムガイド
http://www.enteract.com/~dbaum/nqc/doc/NQC_Guide.pdf
- [6] Linux USB Project
<http://www.linux-usb.org/>
- [7] Logitech Quickcam Express Driver
<http://www.bode.informatik.tu-muenchen.de/~acher/quickcam/quickcam.html>
- [8] bttv Webページ
<http://www.metzlerbros.de/bttv.html>
- [9] xawtv Webページ
<http://www.strusel007.de/linux/xawtv/>
- [10] w3cam Webページ
<http://www.hdk-berlin.de/~rasca/w3cam/>
- [11] LEGO MINDSTORMSパーフェクトガイド
翔泳社 古川剛編、Jin Sato、白川祐記、牧瀬哲郎、倉林大輔、
衛藤仁郎著
1999年 ISBN4-88135-769-7
<http://www.shoeisha.com/book/hp/pc/book/MindStorms/>