RaspberryPi活用6⑤

カメラモジュールの接続

RaspberryPiのカメラモジュールを以下のように接続する。

カメラモジュールの接続

「設定」→「Raspberry Piの設定」→「カメラ」 を有効する。

カメラモジュールの接続

映像は、Raspiのモニタ端子に接続されたモニタに映し出される。
（VNCのネット経由での映像は出ない）

カメラモジュールの接続

動画と静止画を保存するフォルダを準備する。

フォルダ名をCameraで作成する。

```
pi@raspberrypi:~ $ mkdir Camera
```

情報によれば、raspistillの動画はRaspiHDMIモニタに映し出され、プレビュー時間内（ディフォルト5s）の最終フレームを保存される。

```
pi@raspberrypi:~ $ cd Camera/
pi@raspberrypi:~/Camera $ raspistill -o image.jpg
```

```
pi@raspberrypi:~/Camera $ ls
image.jpg
pi@raspberrypi:~/Camera $
```

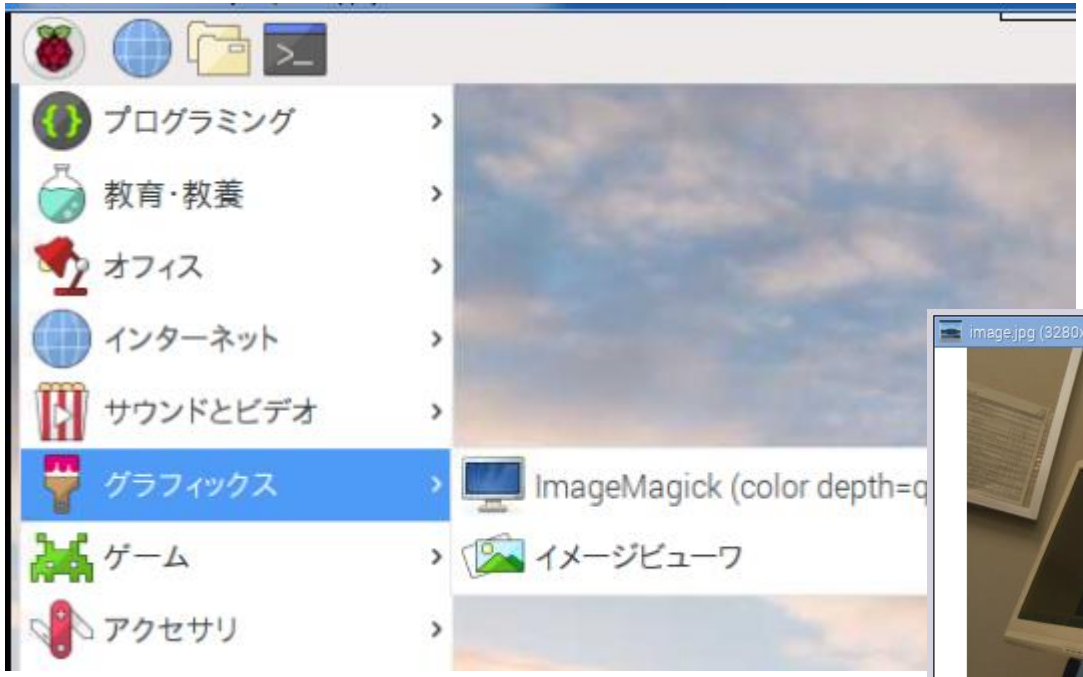
カメラを上下、左右反対に使用する場合は

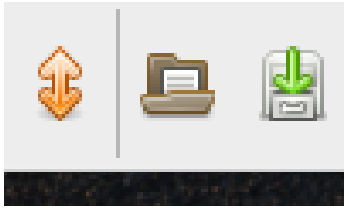
$raspistill -vf -hf -o image.jpg

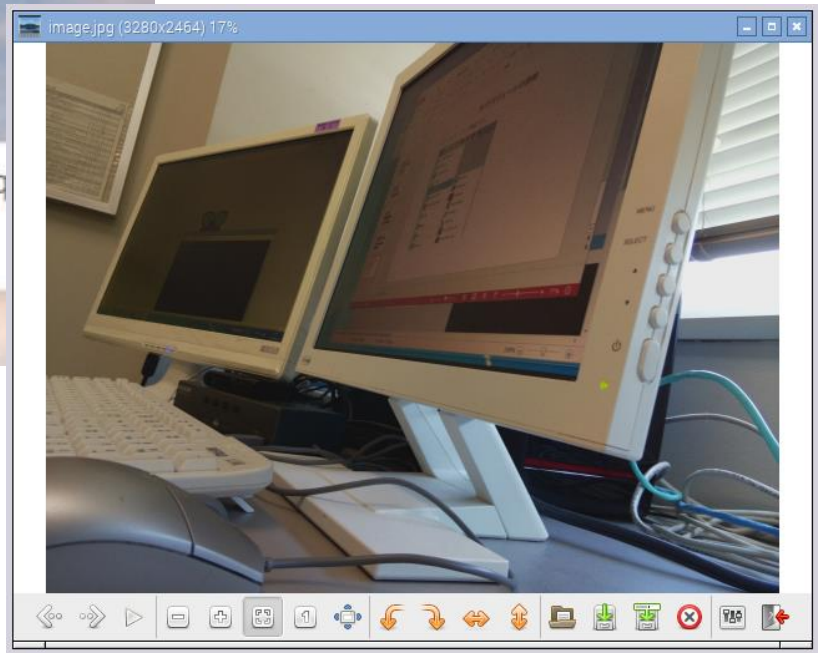
とする。

カメラモジュールの接続

イメージビューワにて確認する。ファイルを開くアイコンから保存したimage.jpgを開く。







カメラモジュールの接続

動画記録はディフォルトでH264フォーマットの５秒間の映像が記録される。-t オプションで10000 とすれば１０秒間の撮影となる。

記録中の動画は先ほど同様HDMIコネクタ出力モニターに映る。

$ raspivid -t 10000 -o video.h264

Raspiに直接接続されたモニタに１０秒間の映像が映るとともに、video.h264のファイル名で記録される。

```
pi@raspberrypi:~/Camera $ raspivid -t 10000 -o video.h264
pi@raspberrypi:~/Camera $ ls video*
video.h264
pi@raspberrypi:~/Camera $
```

カメラモジュールの接続

omxplayerを使ってvideo.h264を実行すれば、同様にコネクタ出力モニターに1920x1080p（ディフォルト解像度）の録画された動画が表示される。

```
pi@raspberrypi:~/Camera $ omxplayer video.h264
Video codec omx-h264 width 1920 height 1080 profile 100 fps 25.000000
Subtitle count: 0, state: off, index: 1, delay: 0
V:PortSettingsChanged: 1920x1080@25.00 interlace:0 deinterlace:0 anaglyph:0 par:
1.00 display:0 layer:0 alpha:255 aspectMode:0
have a nice day ;)
pi@raspberrypi:~/Camera $
```

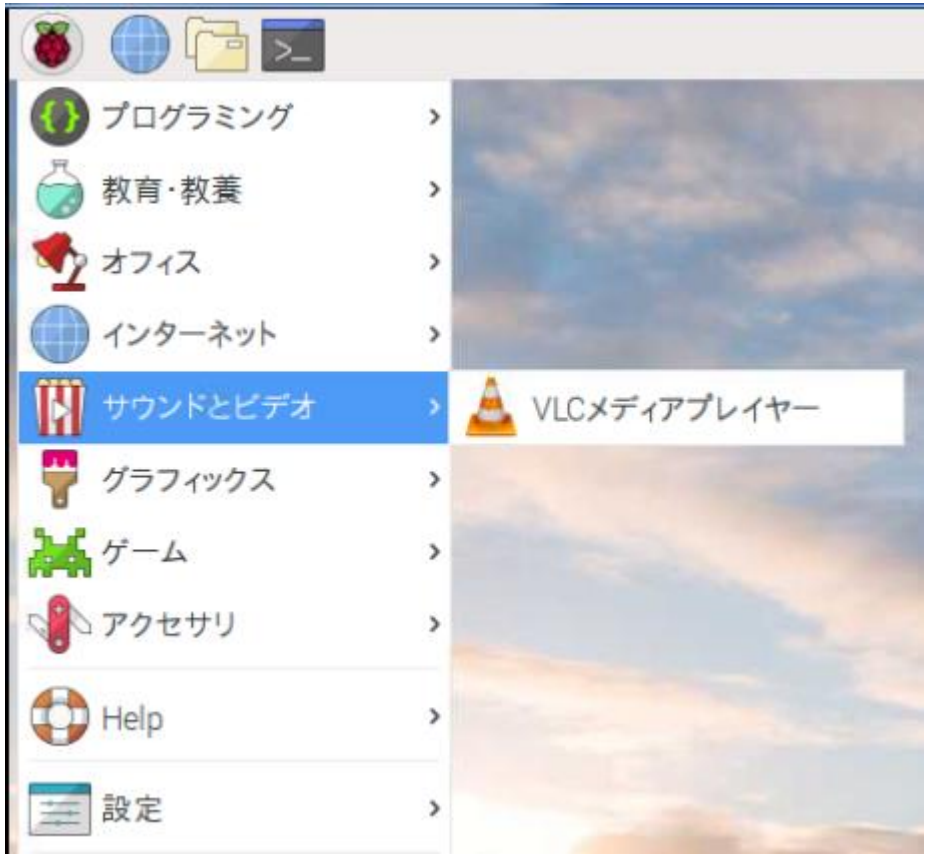
VNCクライアント画面には映らないので、直接接続したモニター画面でチェックのこと。

VNCクライアント画面で再生させるにはVLCプレイヤーが使用できる。

ただし、VNCでの画像はリアルな再生は望めない。

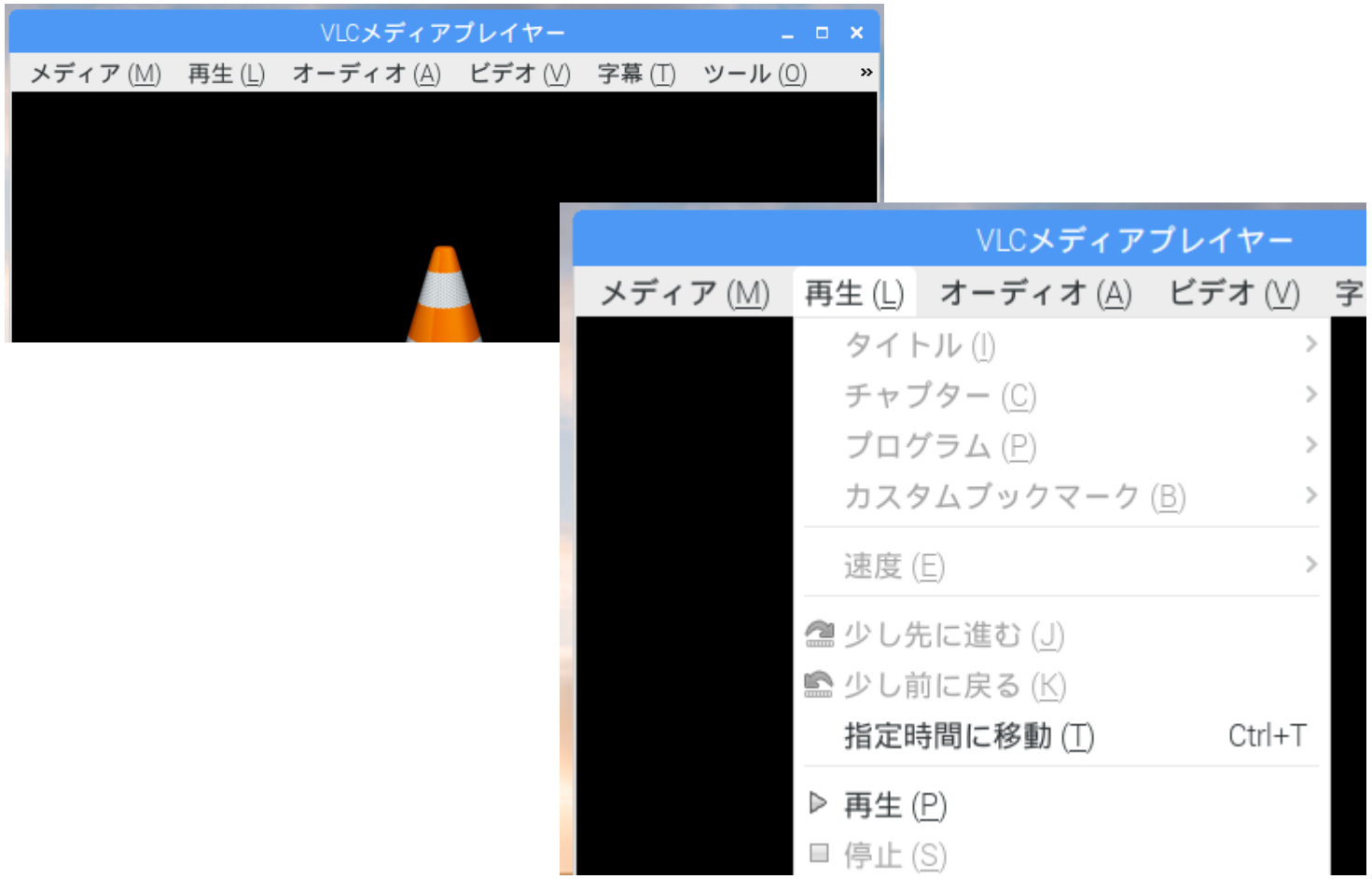
カメラモジュールの接続

バージョンを重ねるうち、VLCメディアプレイヤーは標準でインストールされるようになった。



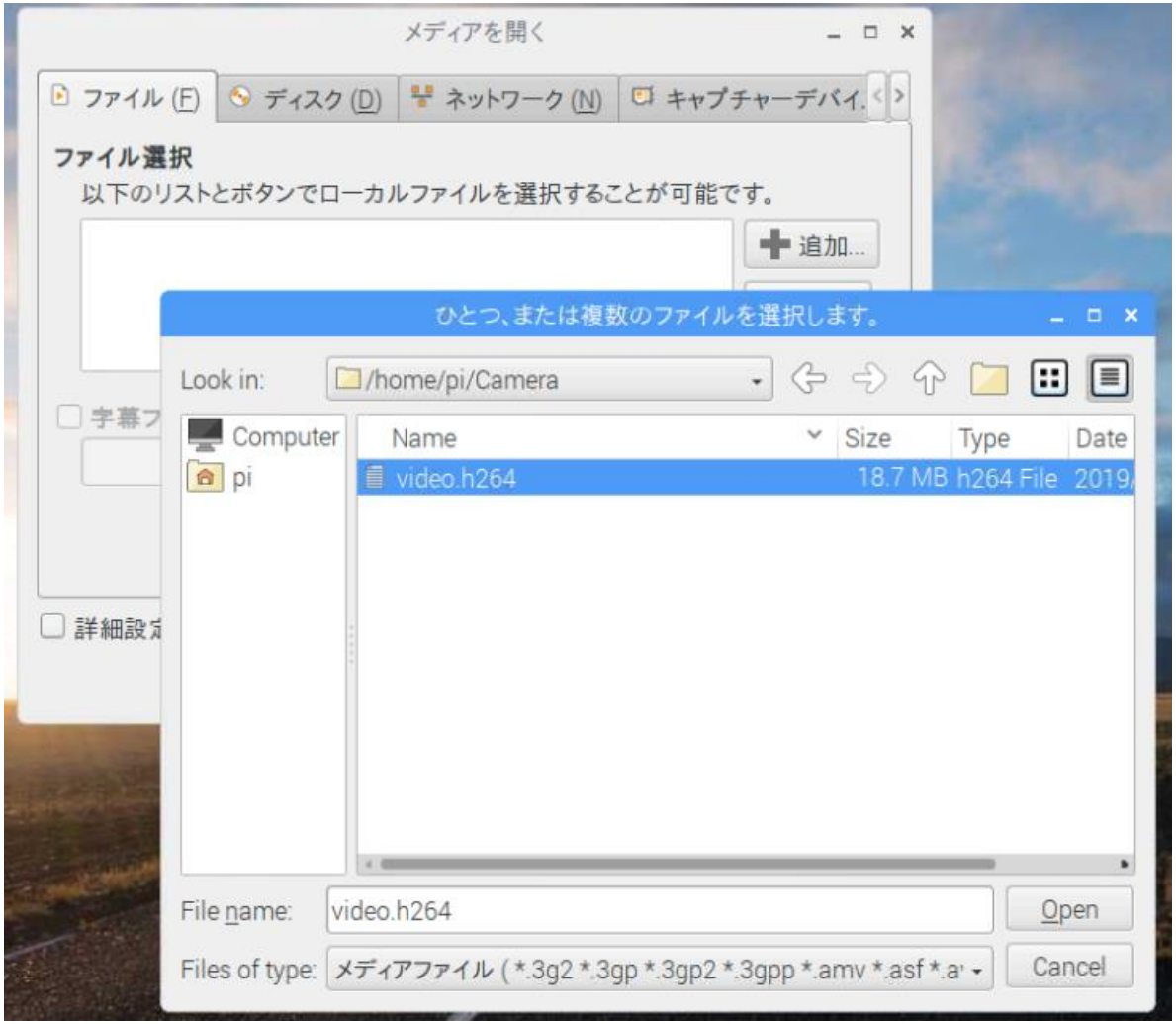
カメラモジュールの接続

「再生」→「再生」



カメラモジュールの接続

「ファイル」 → 「追加」 選択



カメラモジュールの接続

再生をクリック

カメラモジュールの接続

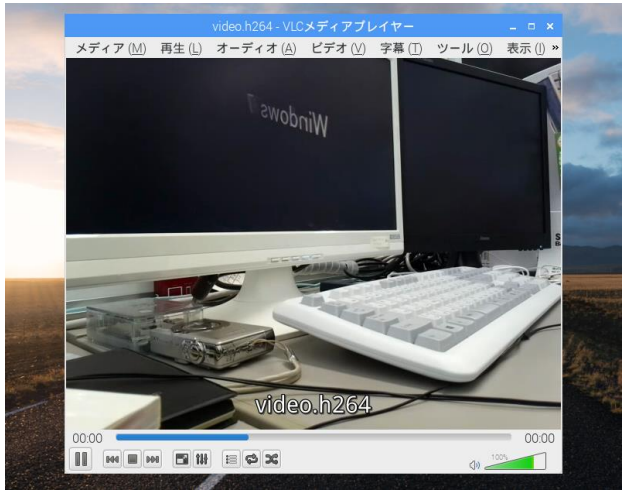
raspividコマンドで解像度640×480で10秒程度、ビットレート10Mbit/s（ネットの10Base程度）の動画ファイルを作成するには.

raspivid に以下のオプションが用意されている。

$ raspivid -o video.h264 -t 10000 -w 640 -h 480 -b 10000000 とタイプインする。H264はmp4形式だが、拡張子をh264にしないと記録されない。

```
pi@raspberrypi:~/Camera $ raspivid -o video.h264 -t 10000 -w 640 -h 480
 -b 10000000
```

VLCにて同様に動画を確認する。



カメラモジュールの接続

VLCはデータを変換する機能をもつ。
「メディア」 → 「変換／保存」 を選択

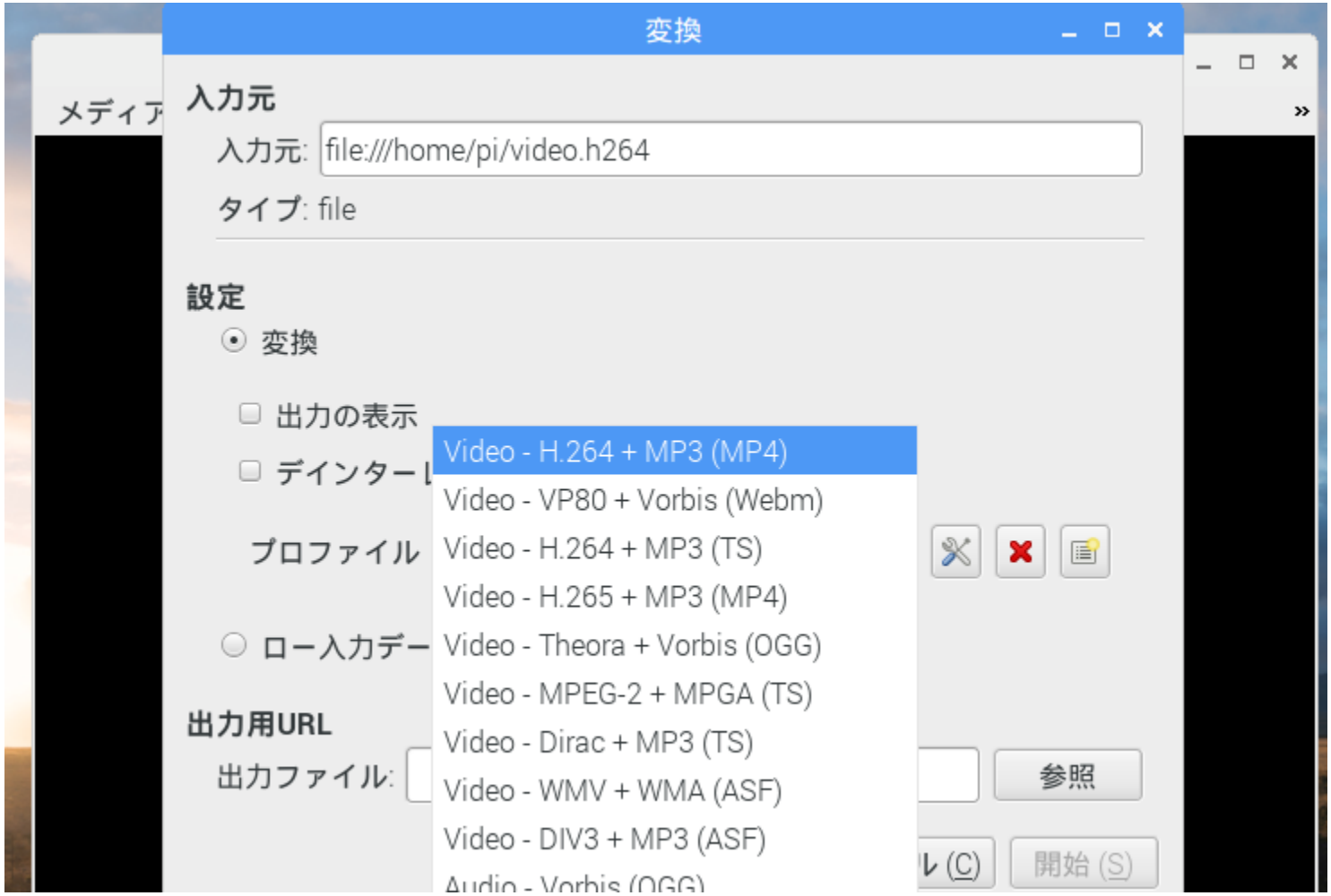
カメラモジュールの接続

「追加」ボタンで対象のファイルを選択する。

「変換／保存」ボタンをクリック

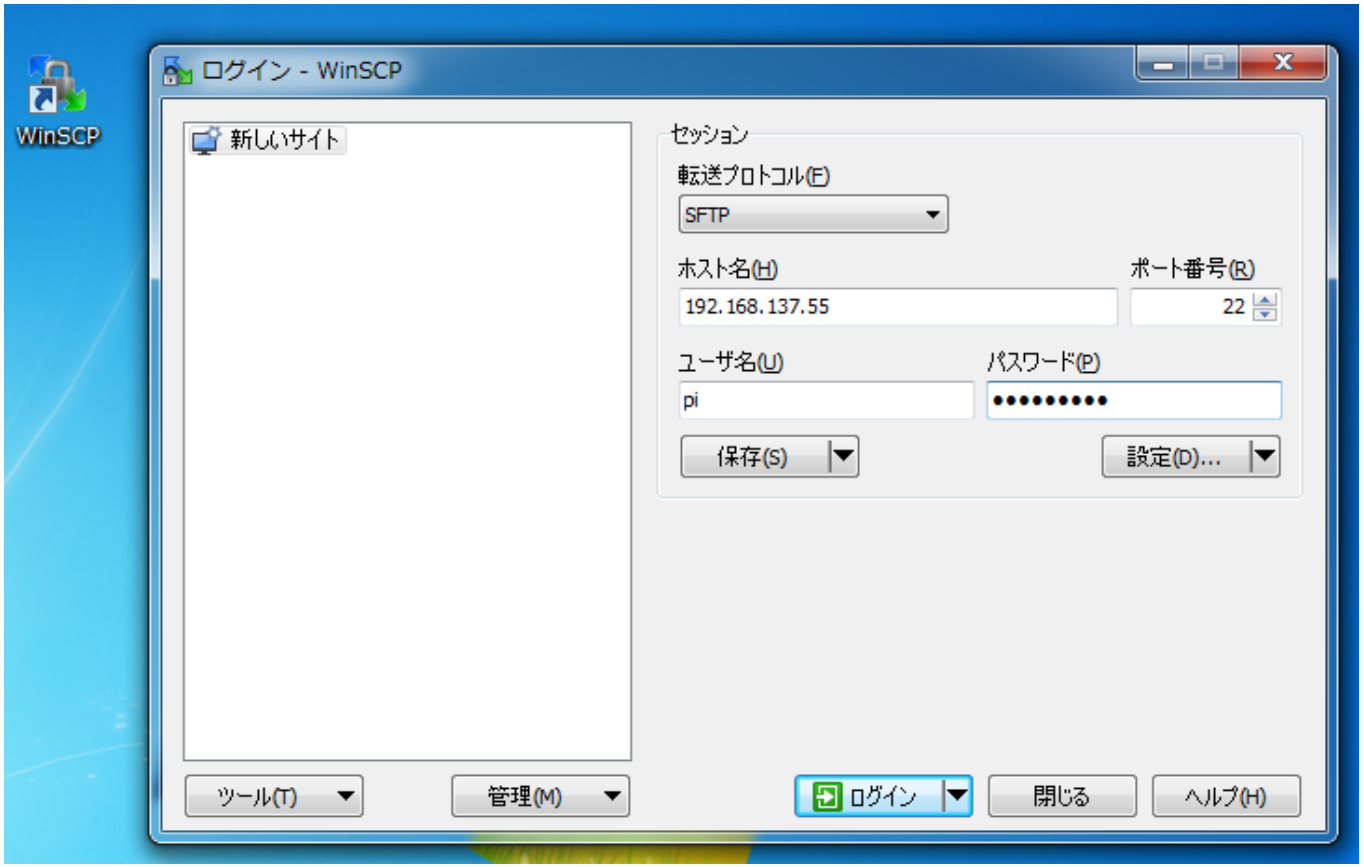
カメラモジュールの接続

「プロファイル」から変換形式を選んで適当なファイル名にして実行する。



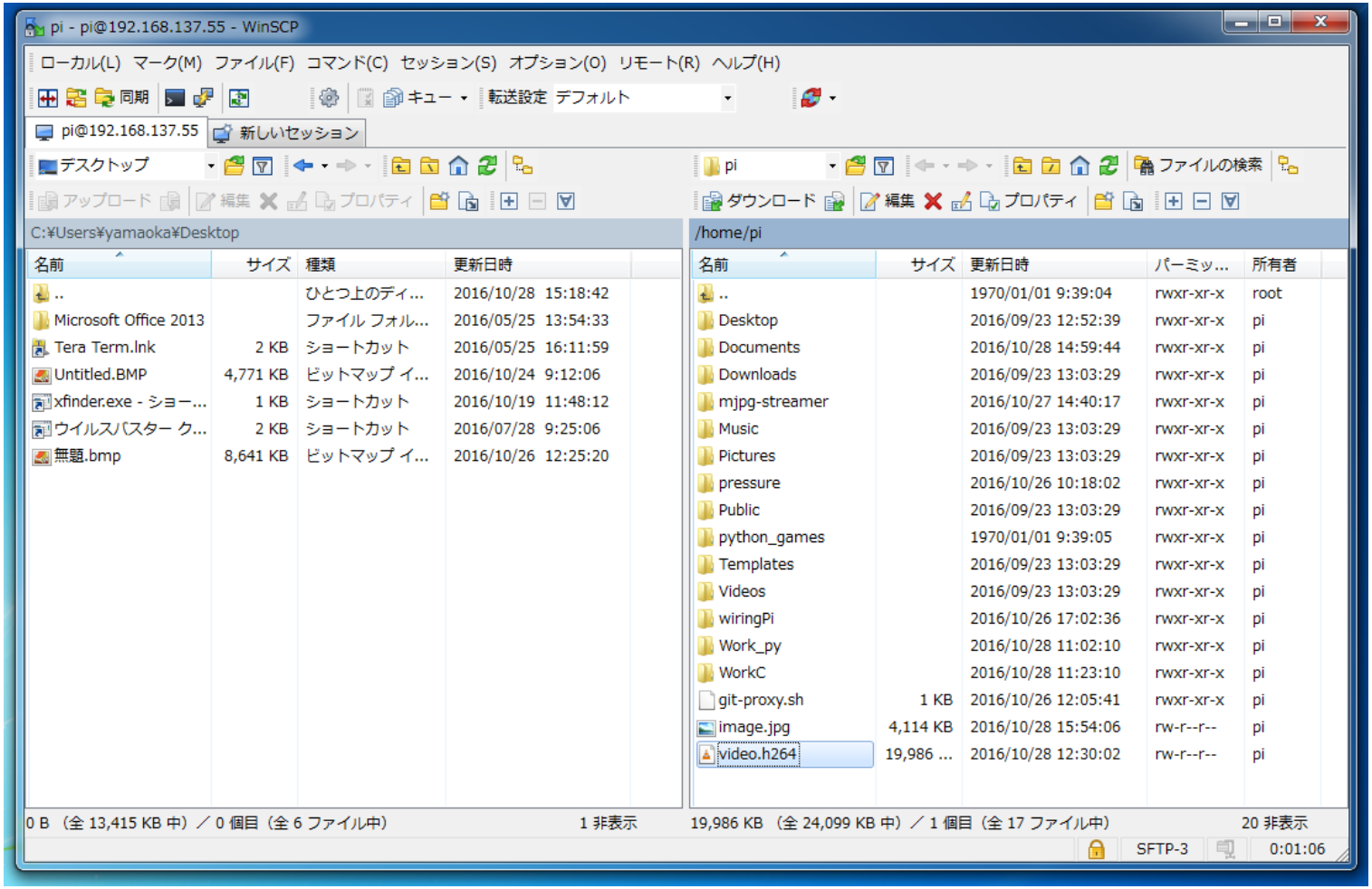
カメラモジュールの接続

VLCデータ変換はVLCプレイヤーがサポートするものしかないので、Windowsマシンの助けを借りてWindowsソフトのXmedia Recodeで変換するのがお勧め。そのためにはWinSCPを使用してRaspiからデータをwindowsマシンの適当なフォルダにコピーする。下記はWinSCPを起動画面。



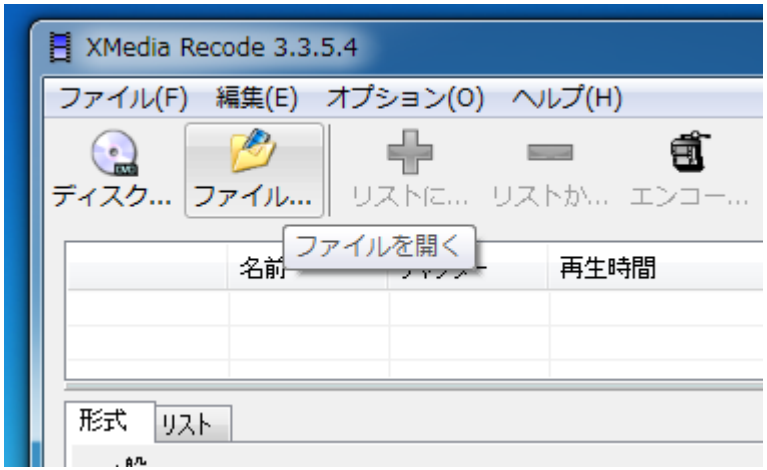
カメラモジュールの接続

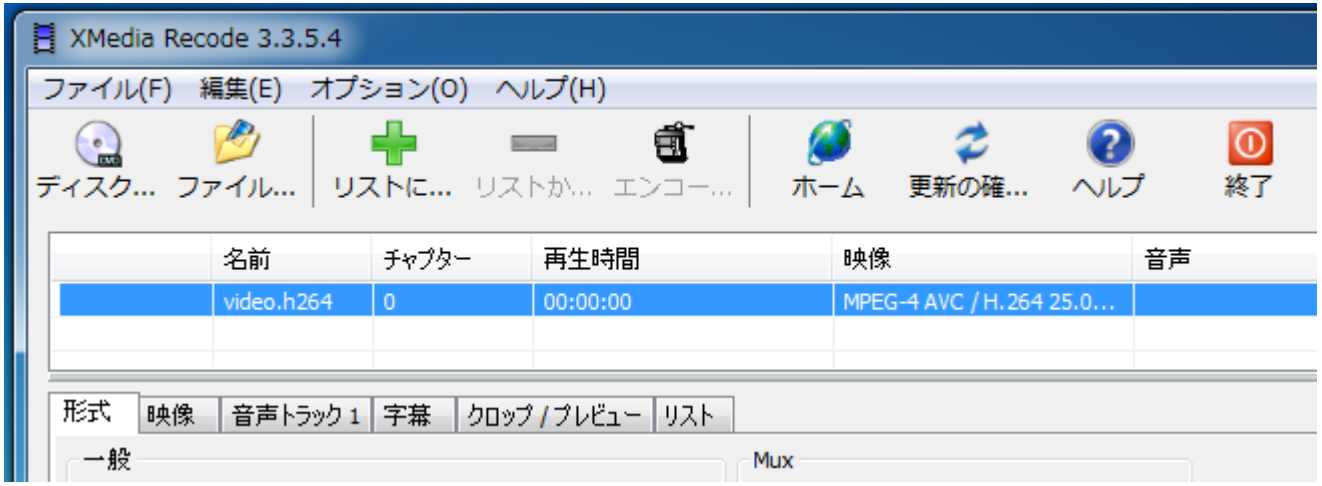
ここではWindowsマシンのディスクトップにコピーする。



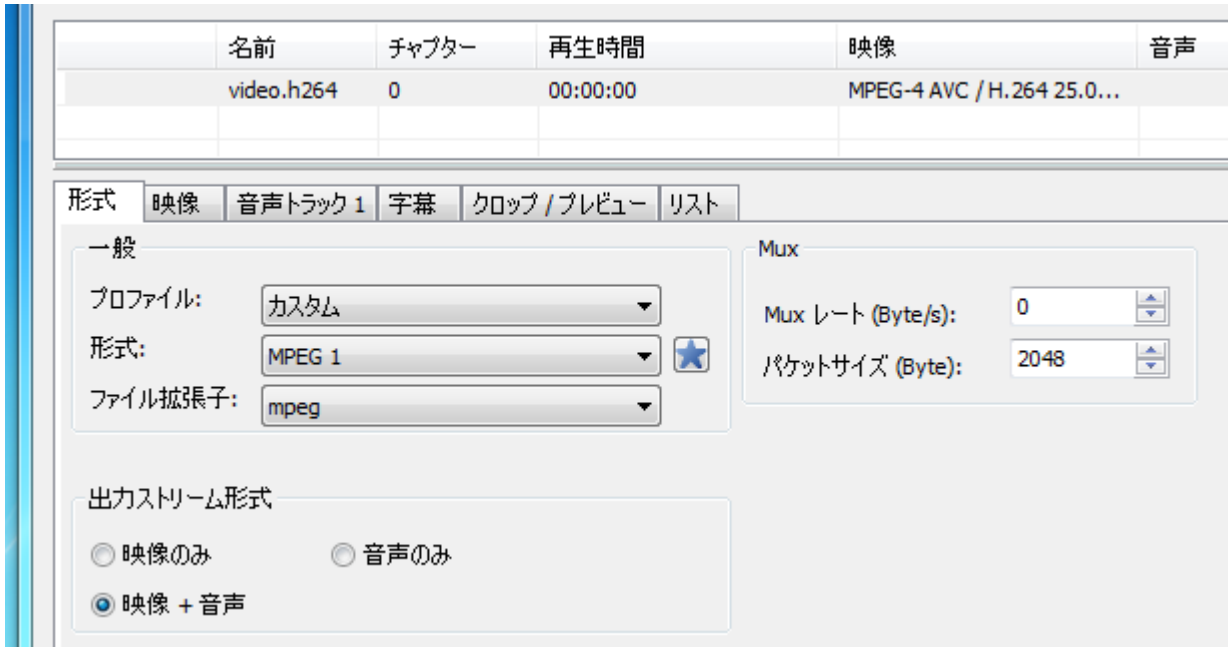
カメラモジュールの接続

Xmedia　Recodeを起動して ファイルを開くでvideo.h264を選択する。

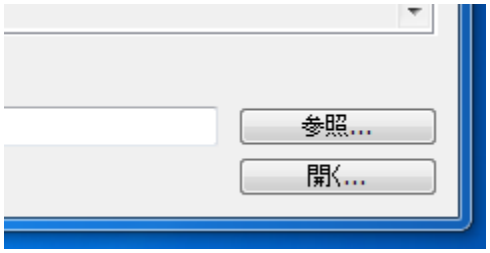




カメラモジュールの接続

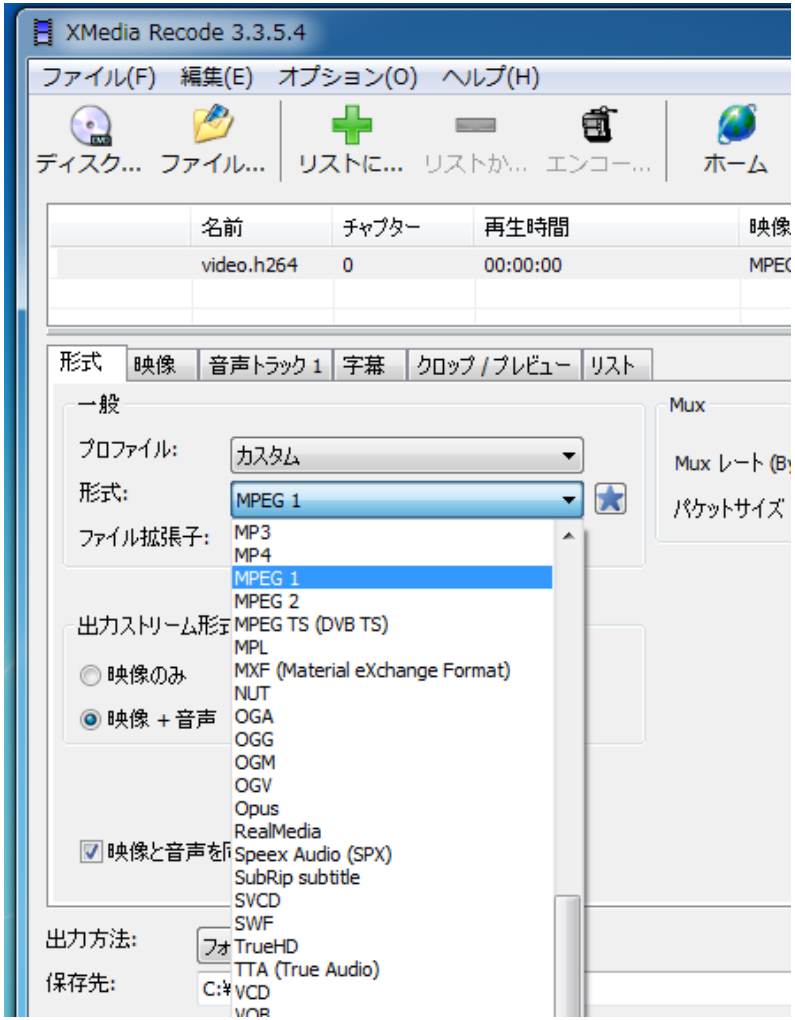


適当なフォルダを指定して「開く」



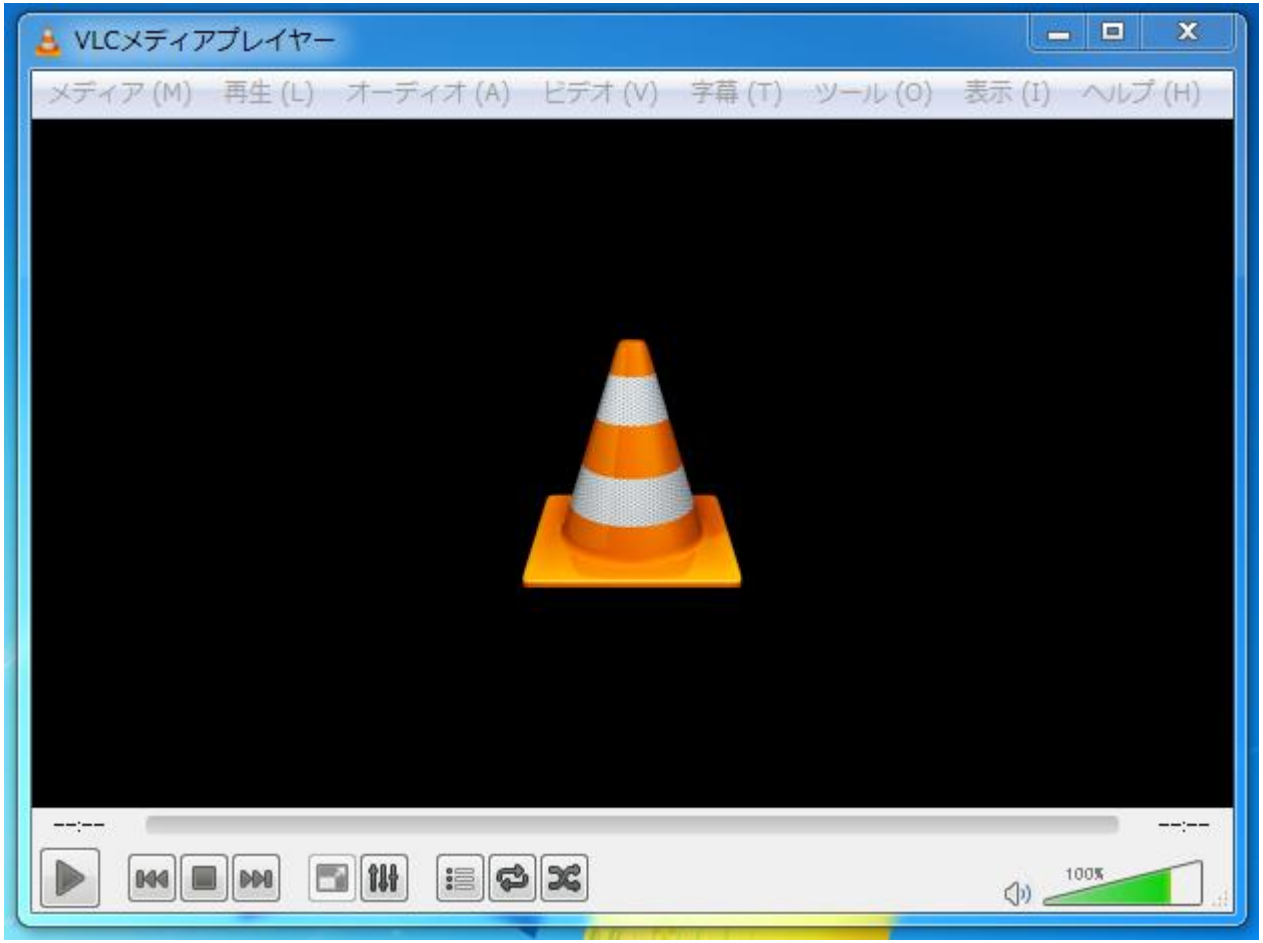
カメラモジュールの接続

形式には多種のデータ形式が用意されている。



カメラモジュールの接続

WinSCPでRaspbrryに書き戻す前にWindowsアプリ側のVLC media playerで確認する。



カメラモジュールのmjpeg-streamer使用
「附１」

カメラモジュールを使ってmjpeg-streamerに使用する。

バージョンを重ねるうちカメラモジュールは接続されると　ディフォルトで /dev/video0 に登録されるようになった。

したがって先の作成した　stream.sh がそのまま使える。

```
pi@raspberrypi:~ $ cd /dev
pi@raspberrypi:/dev $ ls video*
video0  video10  video11  video12
pi@raspberrypi:/dev $
```

ここでは複数のカメラを配置した場合を紹介する。

カメラモジュールのmjpeg-streamer使用
「附１」

ちなみにカメラモジュールデバイス情報は

$ v4l2-ctl –list-device で確認できる。

```
pi@raspberrypi:~ $ v4l2-ctl --list-device
bcm2835-codec (platform:bcm2835-codec):

mmal service 16.1 (platform:bcm2835-v4l2):
        /dev/video0
```

カメラモジュールのmjpeg-streamer使用
「附１」

また、カメラモジュールの動画フォーマットは

$ v4l2-ctl –list-formats で確認できる。

```
pi@raspberrypi:~ $ v4l2-ctl --list-formats
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
```

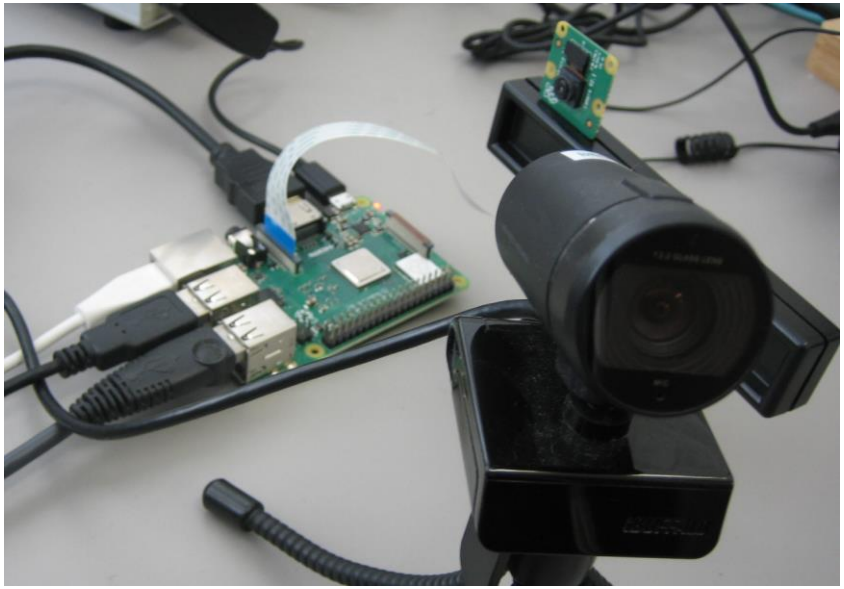
Index 1 および 5 にそれぞれ YUYV，MJPGフォーマットが確認できる。

```
        Index       : 1
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV 4:2:2
```

```
        Index       : 5
        Type        : Video Capture
        Pixel Format: 'MJPG' (compressed)
        Name        : Motion-JPEG
```

カメラモジュールのmjpeg-streamer使用
「附１」

Webカメラを接続追加する。



```
pi@raspberrypi:/dev $ ls video*
video0  video1  video10  video11  video12  video2
pi@raspberrypi:/dev $
```

新たに　video1 と　video2が現れるが、video1が追加されたカメラになる。

カメラモジュールのmjpeg-streamer使用
「附１」

さらにWebカメラを接続追加する。
このとき、カメラも電力を使用するので電源容量には余裕のあるものを使用すること。
省電力を考慮してギリギリの電源供給の場合、カメラを認識しない場合がある。

```
pi@raspberrypi:/dev $ ls video*
video0  video1  video10  video11  video12  video2  video3  video4
pi@raspberrypi:/dev $
```

ドライバは　video3 と　video4が追加される。
ペアで追加される理由は不明。
注意:
どのカメラがどのドライバに割り当てられるか不明である。
今回のように順番に接続した場合は識別できるが、カメラ接続状態から立ち上げた場合の割振りは不明である。
複数カメラを使用する場合、別途特定ルーチン、あるいは設定が必要となるだろう。

カメラモジュールのmjpeg-streamer使用
「附１」

./stream.sh は　３台のカメラのための./stream.shを用意する。

```
pi@raspberrypi:~ $ cd mjpg-streamer/
pi@raspberrypi:~/mjpg-streamer $ cp stream.sh stream0.sh
pi@raspberrypi:~/mjpg-streamer $ cp stream.sh stream1.sh
pi@raspberrypi:~/mjpg-streamer $ cp stream.sh stream3.sh
pi@raspberrypi:~/mjpg-streamer $
```

カメラモジュールのmjpeg-streamer使用
「附１」

./stream0.shの編集。

```
pi@raspberrypi:~/mjpg-streamer $ nano stream0.sh
```

```
#! /bin/sh

PORT="8080"
SIZE="640x480"
F_RATE="60"
MJPG_STREAMER="/home/pi/mjpg-streamer"
sudo "$MJPG_STREAMER/mjpg_streamer" \
-i "$MJPG_STREAMER/input_uvc.so -f $F_RATE -r $SIZE -d /dev/video0 -y -n" \
-o "$MJPG_STREAMER/output_http.so -w $MJPG_STREAMER/www -p $PORT"
```

動画サイズを640x480。 デバイスは /dev/video0。動画フォーマットは
YUYV として “-y”。

カメラモジュールのmjpeg-streamer使用
「附１」

./stream1.shの編集。

```
pi@raspberrypi:~/mjpg-streamer $ nano stream1.sh
```

```
#! /bin/sh

PORT="8082"
SIZE="640x480"
F_RATE="60"
MJPG_STREAMER="/home/pi/mjpg-streamer"
sudo "$MJPG_STREAMER/mjpg_streamer" \
-i "$MJPG_STREAMER/input_uvc.so -f $F_RATE -r $SIZE -d /dev/video1 -n" \
-o "$MJPG_STREAMER/output_http.so -w $MJPG_STREAMER/www -p $PORT"
```

動画サイズを640x480。デバイスは /dev/video1。使用するBuffaroカメラは
MJPEGフォームしかないので“-y”をはずす。

ポートNoは、競合を防ぐため “8082”

カメラモジュールのmjpeg-streamer使用
「附１」

./stream3.shの編集。

```
pi@raspberrypi:~/mjpg-streamer $ nano stream3.sh
```

```
#! /bin/sh

PORT="8084"
SIZE="640x480"
F_RATE="60"
MJPG_STREAMER="/home/pi/mjpg-streamer"
sudo "$MJPG_STREAMER/mjpg_streamer" \
-i "$MJPG_STREAMER/input_uvc.so -f $F_RATE -r $SIZE -d /dev/video3 -n" \
-o "$MJPG_STREAMER/output_http.so -w $MJPG_STREAMER/www -p $PORT"
```

動画サイズを640x480。デバイスは /dev/video3。使用するカメラはMJPEG、YUYVフォームともにあるが “-y” を入れ YUYV とする。

ポートNoは、競合を防ぐため “8084”

カメラモジュールのmjpeg-streamer使用
「附1」

./stream0.sh を起動

```
pi@raspberrypi:~/mjpg-streamer $ ./stream0.sh
MJPG Streamer Version: svn rev: 3:172M
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: 60
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: /home/pi/mjpg-streamer/www/
 o: HTTP TCP port.....: 8080
 o: username:password.: disabled
 o: commands.........: enabled
```

カメラモジュールのmjpeg-streamer使用
「附１」

./stream1.sh を起動

```
pi@raspberrypi:~/mjpg-streamer $ ./stream1.sh
MJPG Streamer Version: svn rev: 3:172M
 i: Using V4L2 device.: /dev/video1
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: 60
 i: Format............: MJPEG
 o: www-folder-path...: /home/pi/mjpg-streamer/www/
 o: HTTP TCP port.....: 8082
 o: username:password.: disabled
 o: commands..........: enabled
```

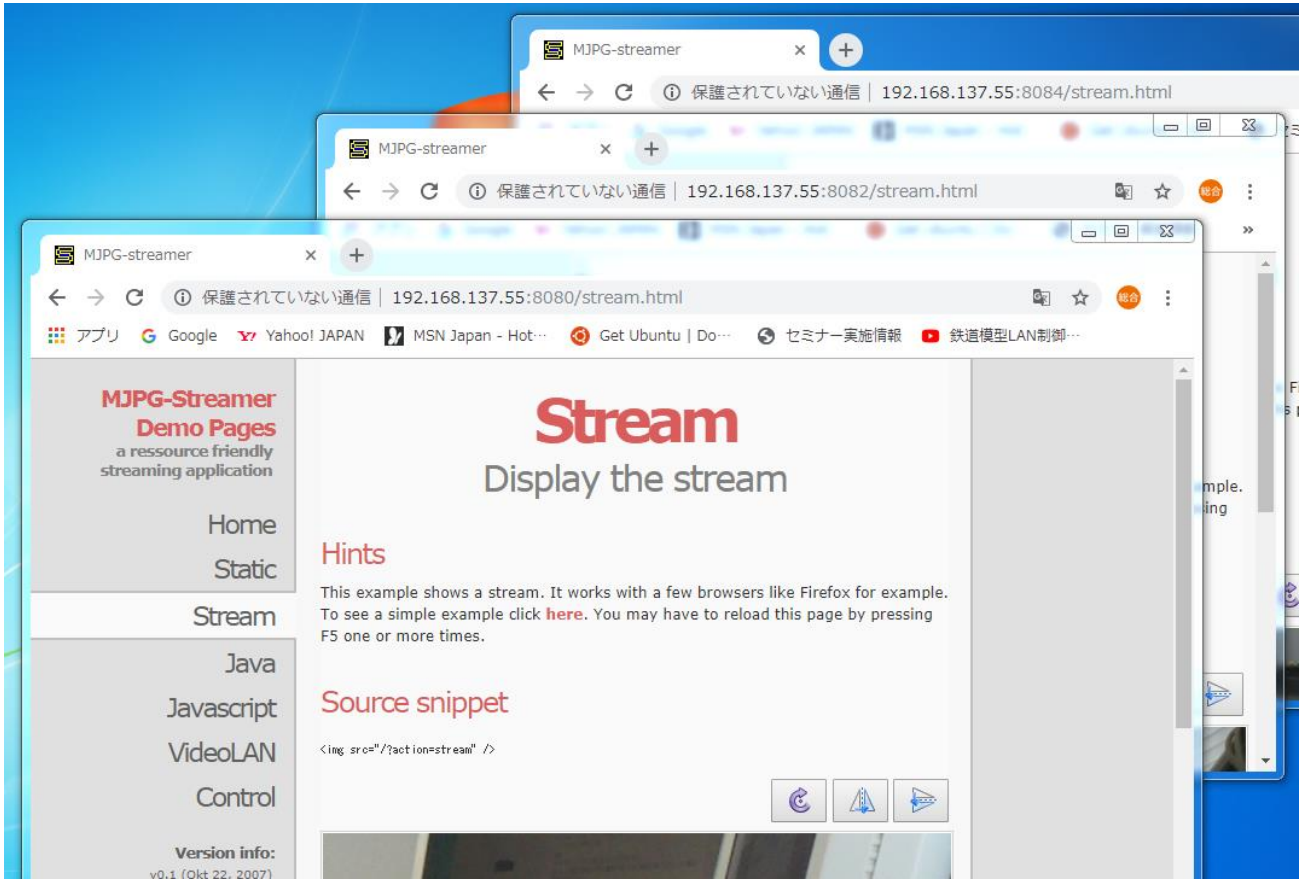
カメラモジュールのmjpeg-streamer使用
「附１」

./stream3.sh を起動

```
pi@raspberrypi:~/mjpg-streamer $ ./stream3.sh
MJPG Streamer Version: svn rev: 3:172M
 i: Using V4L2 device.: /dev/video3
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: 60
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: /home/pi/mjpg-streamer/www/
 o: HTTP TCP port.....: 8084
 o: username:password.: disabled
 o: commands..........: enabled
```

カメラモジュールのmjpeg-streamer使用
「附１」

３箇所のmjpeg-stream による stream動画が確認できる。



課題：連続した5秒動画を個別の10ファイルで保存する。

シェルスクリプトを使い、以下のサンプルファイルを作成せよ

（条件）

・640x480解像度で5秒動画。

・動画のファイル名は識別できるファイル名とすること。

・Recordフォルダ内に当日のフォルダ名がわかるように保存すること。

・systemd.serviceを使い電源投入時に自動起動させる。

課題：連続した5秒動画を個別に10個保存する。

Recordフォルダを作成後、record.shスクリプトファイルを作成する。

```
pi@raspberrypi:~/Camera $ mkdir Record
pi@raspberrypi:~/Camera $ nano record.sh
```

```
#! /bin/sh

today=$(date "+%y%m%d")
dirname=/home/pi/Camera/Record/record$today

if [ ! -d "$dirname" ]; then
        mkdir $dirname
fi

for i in `seq 0 9`
do
        file=file$today$i
        raspivid -o $dirname/$file.h264 -t 5000 -w 640 -h 480
        echo i=$i
done
```

課題：連続した5秒動画を個別の10ファイルで保存する。

パーミションを変更する。

```
pi@raspberrypi:~/Camera $ chmod 755 record.sh
```

実行すると

./Recordフォルダにrecord181101フォルダが作成されてフォルダ内に10個のファイルが保存される。

再度実行時には上書きされる。

```
pi@raspberrypi:~/Camera $ ./record.sh
```

```
pi@raspberrypi:~/Camera $ ls ./Record/
record181101
pi@raspberrypi:~/Camera $ ls ./Record/record181101/
file1811010.h264  file1811013.h264  file1811016.h264  file1811019.h264
file1811011.h264  file1811014.h264  file1811017.h264
file1811012.h264  file1811015.h264  file1811018.h264
pi@raspberrypi:~/Camera $
```

課題：連続した5秒動画を個別の10ファイルで保存する。

記録時刻は

$ ls -l にて確認可能

```
pi@raspberrypi:~/Camera $ ls -l ./Record/record181101/
合計 3104
-rw-r--r-- 1 pi pi 348441 11月  1 16:14 file1811010.h264
-rw-r--r-- 1 pi pi 322404 11月  1 16:15 file1811011.h264
-rw-r--r-- 1 pi pi 312015 11月  1 16:15 file1811012.h264
-rw-r--r-- 1 pi pi 321380 11月  1 16:15 file1811013.h264
-rw-r--r-- 1 pi pi 337066 11月  1 16:15 file1811014.h264
-rw-r--r-- 1 pi pi 322362 11月  1 16:15 file1811015.h264
-rw-r--r-- 1 pi pi 307182 11月  1 16:15 file1811016.h264
-rw-r--r-- 1 pi pi 309491 11月  1 16:15 file1811017.h264
-rw-r--r-- 1 pi pi 297509 11月  1 16:15 file1811018.h264
-rw-r--r-- 1 pi pi 279788 11月  1 16:15 file1811019.h264
pi@raspberrypi:~/Camera $
```