

バイナリファイル（写真イメージ）の自動アップロード方法について

2017/12/18

石山隼行

ishiyama@hn.t.u-tokyo.ac.jp

データをサーバに転送するにはGET/POSTメソッドを利用するのが簡単という話を【IoT
サーバで簡易データロガーの作成】

https://docs.google.com/a/hn.t.u-tokyo.ac.jp/document/d/1_8Q7p2QnvL5bXZ_fQ3yeEdbODdRuH3ZMc7CDI_moR0A/edit?usp=sharing

で書きました。

※POSTメソッドを使えば255文字以上のテキストデータをアップすることができます。
getpost.pl をそのまま使えますのでやってみてください。

しかし、画像ファイルなどのように、テキストではないバイナリーデータファイルをアップロードする場合は別のテクニックが必要です。

ここでは、簡単なPerlスクリプト（サーバ側）を利用してデータをサーバに保存する方法と、Pythonを使ってRaspberryPiシリーズから画像をサーバへ自動で転送する方法を紹介します。

ファイルアップロードの仕方

参考：

<http://www.hidekik.com/cookbook/p2h.cgi?id=upload>

□一カル（HTML）

□一カル側からサーバへデータ（ファイル）を送るインターフェースを人間が操作する場合はWebブラウザ経由がもっとも簡単です。

upload_bin.html

```
<form action=".//cgi/upload.cgi" method="post" enctype="multipart/form-data">
<input type=file name="upfile">
<input type=submit name=sub value="Uplaod">
</form>
```

このHTMLファイルをPiサーバのpublic_htmlディレクトリに保存しておきます。

上記のHTMLをブラウザで読み込むと



このようになるはずです。

〔ファイルを選択〕ボタンでファイル選択し、〔Upload〕ボタンで cgi/upload.cgi を呼び出し実行する簡単なHTMLです。

CGI (Perlスクリプト)

HTMLから呼び出されるサーバ側のCGIプログラムは以下のようになります。

upload.cgi

```
#!/usr/bin/perl

use CGI;
use File::Copy;
use File::Basename;

my $q = new CGI;

my $fname = basename($q->param('upfile'));
my $path = '..//files';
my $newfile = "$path/$fname";

my $fh = $q->upload('upfile');
copy ($fh, "$newfile");
undef $q;

print "Location: ..//upload_bin.html\n\n";
```

public_html/cqi ディレクトリに保存します。

細かい解説は <http://www.hidekik.com/cookbook/p2h.cgi?id=upload> を参照。

このCGIは起動されると、`../files` ディレクトリにアップロードされたファイルを保存し、保存した後に `/upload-bin.html` を再度ブラウザに表示する。というものです。

※もちろんCGIプログラムなので、chmod a+x upload.cgi を忘れずに。

アップロードされたファイルが保存されるディレクトリも作成しておきます。

```
[IoT000@iserver:~/public_html]$ mkdir files  
[IoT000@iserver:~/public_html]$ chmod atw files
```

ここで、作ったディレクトリを **chmod a+w** していることに注意。

これは、このディレクトリにすべてのユーザが書き込みを許可した。ということです。データの保存は書き込み権限がないと保存できませんので、ディレクトリ全体を許可した形になります。

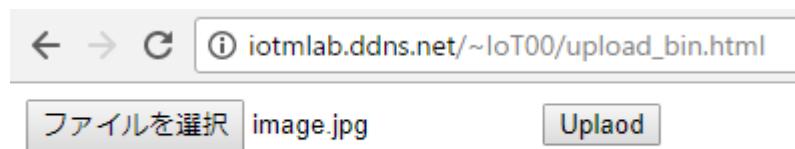
※全体を許可してしまうとセキュリティ上でも大変危険なので、保存用のディレクトリを個別に作成しました。

filesディレクトリの中をls してみます。

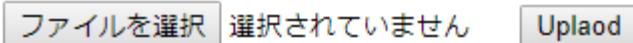
```
IoT00@piserver:~/public_html $ cd files  
IoT00@piserver:~/public_html/files $ ls  
IoT00@piserver:~/public_html/files $
```

当然出来立てなのでなにも入っていません。

ブラウザからバイナリ（画像ファイルなど）をアップロードしてみます。



正しく実行（アップロード）できれば、ブラウザ上は単に同じHTMLを読みだすだけなので、再度



このような表示となります

サーバ側をみてみると

```
IoT00@piserver:~/public_html/files $  
IoT00@piserver:~/public_html/files $ ls  
image.jpg  
IoT00@piserver:~/public_html/files $
```

このようにバイナリファイルが保存されているはずです。

※HTMLで画像表示などはとても簡単にできるので割愛します。各自やってみてください。

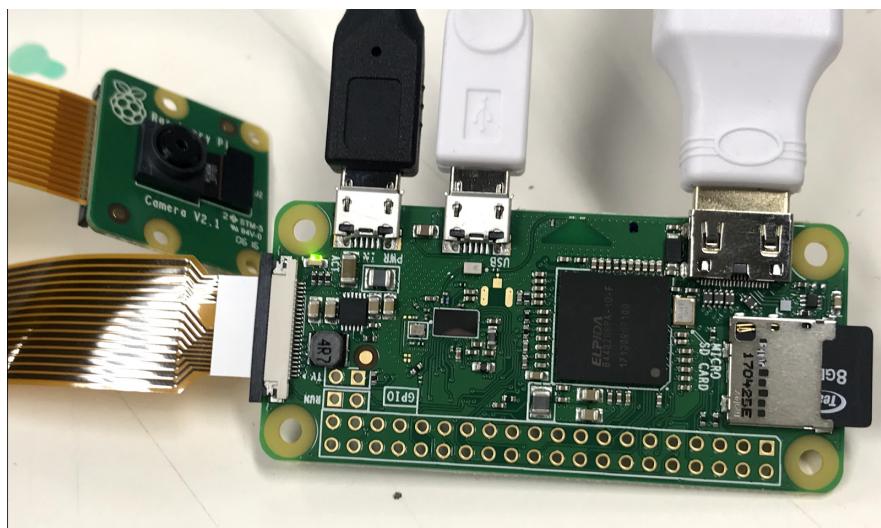
自動アップロード

人間がWebブラウザを触れない状態で自動でアップロードする仕組みを、RaspberryPi + Pythonで作ってみました。

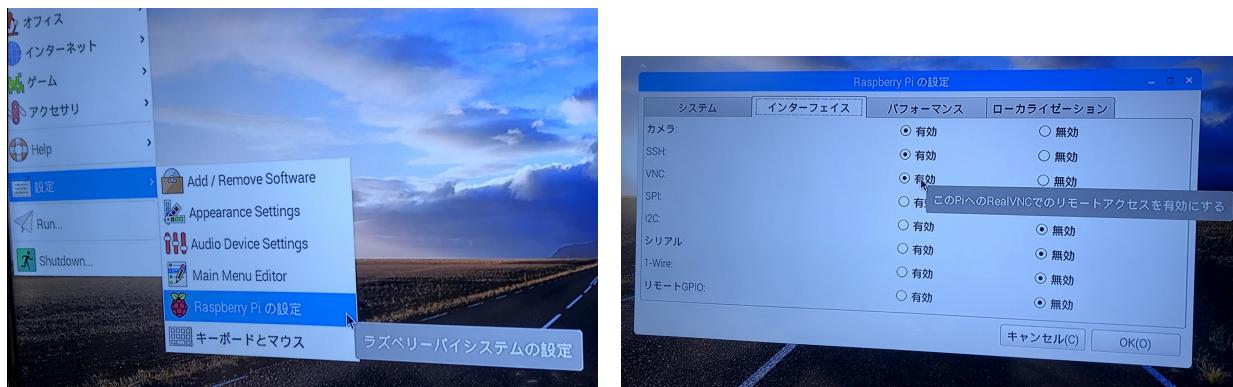
今回は requestsモジュールを利用します。

[Python] requestsモジュールを使って楽々HTTP(s)通信を行う
<http://www.yoheim.net/blog.php?q=2017080> 参考

まず、ローカル側のRaspberryPiを用意。
小型のRaspberryPi Zero W+ カメラをセットアップ。



通常のセットアップに、カメラ、SSH、(今回は使いませんが) VNCをconfigでONにしておきます。



※WiFi設定やapt get update & upgrade は割愛。

requestsモジュールのインストール

コンソール（ターミナル）を立ち上げ、

sudo pip install requests

してインストールします。

続いて python を立ち上げ、
サンプルを参考に手打ちでテスト

```
>>> import requests
>>> url='http://iotmlab.ddns.net/~IoT00/cgi/upload.cgi'
>>> files = {'upfile': ('image.jpg', open('image.jpg', 'rb'))}
>>> r = requests.post(url, files=files)
>>> r.text
u'<form action="./cgi/upload.cgi" method="post" enctype="multipart/form-data">\n<input type=file name="upfile">\n<input type=submit name=sub value="Upload">\n</form>\n\t \n'
>>> █ ファイルを選択して下さい。 | Upload
```

パラメータを'upfile'にしていることに注意（受取側のCGIスクリプトに合わせてあります）

r.text にレスポンスの内容が入っています。（この場合cgiの返答がhtml形式ではいっています）

サーバ内の保存先ディレクトリ (files)を確認すると

```
IoT00@piserver:~/public_html/files $ ls
image.jpg
```

無事データが保存されています。

先程のテストをプログラム化します。

img_up_test.py

```
#!/usr/bin/python2.7
import requests
url='http://iotmlab.ddns.net/~IoT00/cgi/upload.cgi'
files = {'upfile': ('image.jpg', open('image.jpg', 'rb'))}
r = requests.post(url, files=files)
print r.text
```

※最後の行、python2.7系は print r.textですが、python3系では print(r.text) としないとエラーになります。

※もちろん実行ファイルは chmod a+xを忘れずに。

実行結果

```
[pi@raspberrypi:~ $ ls
Documents img_up_test.py ダウンロード デスクトップ ビデオ 画像
image.jpg python_games テンプレート ドキュメント 音楽 公開
[pi@raspberrypi:~ $ cat img_up_test.py
#!/usr/bin/python2.7
import requests
url='http://iotmlab.ddns.net/~IoT00/cgi/upload.cgi'
files = {'upfile': ('image.jpg', open('image.jpg', 'rb'))}
r = requests.post(url, files=files)
print r.text

[pi@raspberrypi:~ $ ./img_up_test.py
<form action="./cgi/upload.cgi" method="post" enctype="multipart/form-data">
<input type=file name="upfile">
<input type=submit name=sub value="Uplaod">
</form>

[pi@raspberrypi:~ $ ]
```

※image.jpgがあるディレクトリで実行すること。

参考

<http://danglingfarpointer.hatenablog.com/entry/2016/01/27/221213>

参考：Piカメラでのフォトキャプチャー

これは非常に簡単にできます。

[Python picamera – Raspberry Pi Documentation](#)

<https://www.raspberrypi.org/documentation/usage/camera/python/README.md>

を参考に

まず、

```
[pi@raspberrypi:~ $
[pi@raspberrypi:~ $ sudo apt-get install python-picamera
パッケージリストを読み込んでいます... 完了
```

でパッケージのインストールを行い、

サンプルのとおり、Python上から

```
[>>> import picamera
[>>> camera = picamera.PiCamera()
[>>> camera.capture('image.jpg')
```

わずかこの3行でカメラからフォトデータをキャプチャできます。

組み合わせて動作

カメラでキャプチャし、サーバへアップロードするまでを自動化（組み合わせただけ）してみました。

ソースコード（カメラ側のRaspberryPi Zeroにおきます）

auto_post.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#初期設定
import picamera
import requests

#カメラからキャプチャー
camera = picamera.PiCamera()
camera.capture('image.jpg')

print 'カメラキャプチャ'

#キャプチャしたイメージをサーバーへpost
url='http://iotmlab.ddns.net/~IoT00/cgi/upload.cgi'
files = {'upfile': ('image.jpg', open('image.jpg', 'rb'))}
r = requests.post(url, files=files)

print 'サーバPOST終了'
```

実行結果

```
[pi@raspberrypi:~]$ ./auto_post.py
カメラキャプチャ
サーバPOST終了
```

実行したカメラPi側の画像

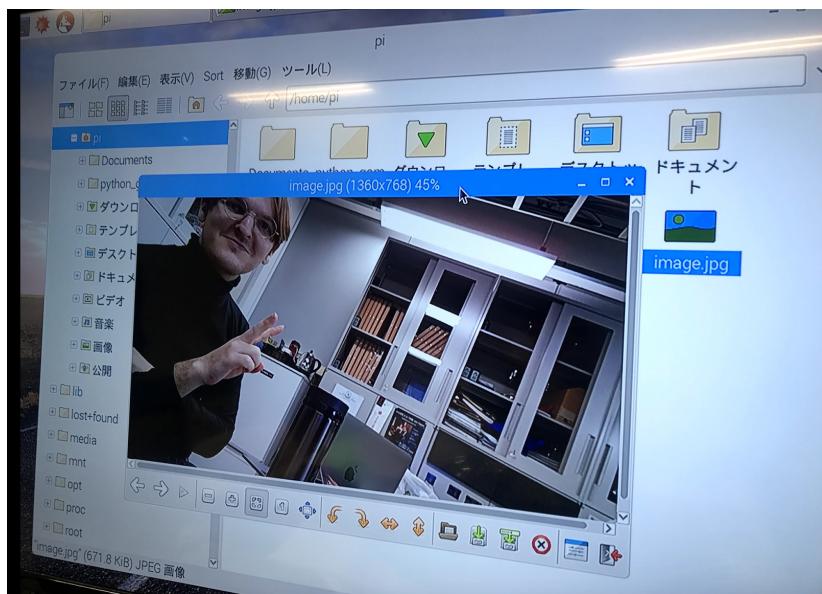
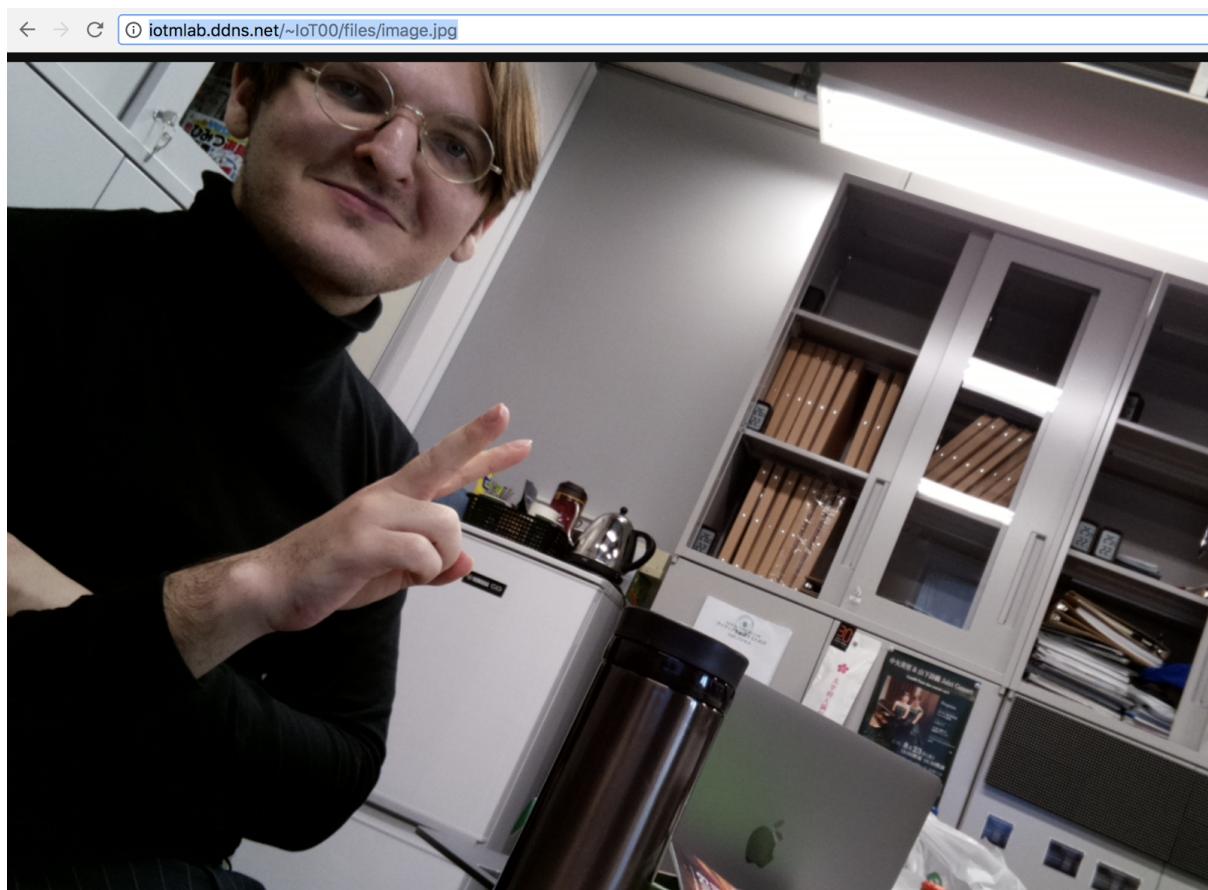


image.jpgに撮影した写真が保存されます。

サーバ側は、public_html/files/image.jpg にデータが保存されるので
<http://iotmlab.ddns.net/~IoT00/files/image.jpg>を見れば写真が見れるはずです。

ブラウザで見てみると正常に保存されていることがわかります。



※実際に動かしてみると、どうやらカメラの初期設定に時間が数秒かかるようです。

プログラムは初期設定だけ先にしておいて、ボタン等のトリガーでキャプチャするように改造したほうがいいでしょう。

また、現状はimage.jpgを常に書き換えてしまっているので、そこも、たとえばファイル名を日時（分秒）にするなど細工をすると良いと思います。

※これらはすべてPiサーバと、手元のRaspberryPi ZEROで作ったサンプルです。Arduinoでも同様のことができるはずですが未検証です。

—以上—