



## Features of Raspberry Pi 3 Model B

### A. Objectives

1. Learn about the difference between web framework and web servers
2. Turn your Pi into a Webserver supporting PHP
3. Setting up your ftp server
4. Port Forwarding concept
5. Learn how to play audio files using Pi 3
6. Connect your Pi to a camera

### B. Time of Completion

This laboratory activity is designed for students with some knowledge of Raspberry Pi and it is estimated to take about 4-5 hours to complete.

### C. Requirements

1. A Raspberry Pi 3 Model 3
2. 32 GByte MicroSD card → Give your MicroSD card to the lab instructor for a copy of Ubuntu.
3. USB adaptor to power up the Pi
4. Web camera
5. Speaker with audio jack

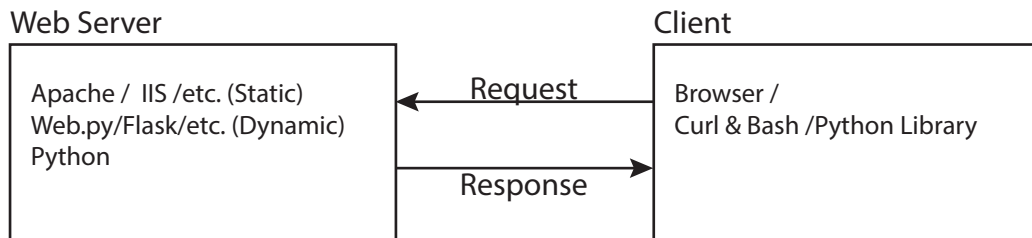
### D. Pre-Lab

Lear about Web and FTP servers and PHP.

## E. Lab

### 1. Understanding Web-Interaction

The figure below shows the interaction between a web server and a client. As shown in the figure below, in such client-server model different software packages can be utilized to enable http requests and responses both in the server's and client's side.



These servers generally require HTML codes to create web pages. Web pages are typically static. Therefore, in order to create Dynamic web pages, it is often required to utilize other coding languages such as PHP on the server side. An alternative to utilizing  
On the client machine, we typically use a browser to access the web server. It is also possible to use Linux commands such as `curl` to send an HTTP request to the server and receive its response. The received response can be parsed using a scripting language. Alternatively, it is possible to use scripting languages, such as Python, to create HTTP requests on the user side. In the case of Python, standard library modules can be utilized to define classes, which implement the client/server sides of the HTTP and HTTPS protocols. Such library modules include `httplib`, `requests`, etc.

#### 1.1. Web Servers Vs. Web Frameworks

A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. The term can refer to the entire system, or specifically to the software that accepts and supervises the HTTP requests<sup>1</sup>. Apache and Internet Information Services (IIS, formerly Internet Information Server) are two commonly used web servers. Other web servers include NGINX and GWS (Google Web Server).

Apache originally referred to the web server, which is now officially called Apache HTTPD and is a HTTP(S) web server that serves pages in response to requests from HTTP(S) clients. Note that Apache has other extensions including Apache Tomcat which is a Java application server that is intended to execute Java code, usually in response to requests from a HTTP(S) client.

NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

Originally, NGINX was designed to handle hundreds of thousands of concurrent connections. Today it is used to power more than 50% of the busiest sites on the web, such as such as Dropbox, Netflix, and Zynga. More than 333 million websites worldwide, including the majority of the 100,000 busiest websites, rely on NGINX Plus and NGINX to deliver their content quickly, reliably, and securely. Google Web Server (GWS) is a custom Linux-based Web server that Google uses for its online services.

An alternative to utilizing Apache or IIS, is to create a web server a web framework (WF) or web application framework (WAF)<sup>2</sup>. A WAF is a software framework that is designed to support the development of web applications including web services (services offered by an electronic device to another electronic device, communicating with each other via the World Wide Web.), web resources (documents, images, URLs, etc.), and web APIs (programmatic interface consisting of one or more publicly exposed endpoints to a defined request–response message system).

<sup>1</sup> [https://en.wikipedia.org/wiki/Web\\_service](https://en.wikipedia.org/wiki/Web_service)

<sup>2</sup> [https://en.wikipedia.org/wiki/Web\\_framework](https://en.wikipedia.org/wiki/Web_framework)

Web frameworks provide a standard way to build and deploy web applications. Web frameworks aim to automate the overhead associated with common activities performed in web development. For example, many web frameworks provide libraries for database access, templating frameworks, and session management, and they often promote code reuse. Although they often target development of dynamic web sites, they are also applicable to static websites.

There are many different Web Frameworks, each of which can be based on a different programming (or scripting) languages. The table below shows different examples of Web frameworks.

V · T · E	Web frameworks	[hide]
	Comparison	
<b>C++</b>	CppCMS · Wt	
<b>CLI</b>	ASP.NET (Core · AJAX · Dynamic Data · MVC · Razor · Web Forms) · DNN · BFC · MonoRail · OpenRasta · Umbraco	
<b>ColdFusion</b>	CFWheels · ColdBox Platform · ColdSpring · Fusebox · Mach-II · Model-Glue	
<b>Common Lisp</b>	Caveman2 · CL-HTTP · UnCommon Web · Weblocks	
<b>D</b>	Vibe.d	
<b>Haskell</b>	Happstack · Yesod · Snap	
<b>Java</b>	AppFuse · Flexive · Grails · GWT · ICEfaces · ItsNat · JavaServer Faces · JHipster · Jspix · JWt · OpenXava · Play · Reasonable Server Faces · Remote Application Platform · RIFE · Seam · Sling · Spring · Stripes · Struts · Tapestry · Vaadin · Vert.x · WebWork · Wicket · WaveMaker · ZK	
<b>JavaScript</b>	Ample SDK · Angular/AngularJS · Backbone.js · Chaplin.js · Closure · Dojo Toolkit · Ember.js · Ext JS · jQuery · Meteor · MooTools · Node.js · OpenUI5 · Prototype · React · Rico · script.aculo.us · Sencha Touch · SproutCore · Wakanda	
<b>Perl</b>	Catalyst · Dancer · Mason · Maypole · Mojolicious · <u>WebGUI</u>	
<b>PHP</b>	CakePHP · CodeIgniter · <u>Drupal</u> · Fat-Free · FuelPHP · Flow · Gyroscope · Horde · Kohana · Laravel · Lithium · Midgard · MODX · Nette · Phalcon · Pop PHP · PRADO · Qcodo · Silex · SilverStripe · Symfony · TYPO3 · Xaraya · XOOPS · Yii · Zend Framework	
<b>Python</b>	BlueBream · CherryPy · Django · <u>Flask</u> · Grok · Nevow · Pyjs · Pylons · Pyramid · Quixote · TACTIC · Tornado · TurboGears · web2py · Webware · Zope 2	
<b>Ruby</b>	Camping · Merb · Padrino · Ruby on Rails · Sinatra	
<b>Scala</b>	Lift · Play · Scalatra	
<b>Smalltalk</b>	AIDA/Web · Seaside	
<b>Other languages</b>	Application Express (PL/SQL) · Grails (Groovy) · Kepler (Lua) · OpenACS (Tcl) · <b>Phoenix</b> (Elixir) · SproutCore (JavaScript-Ruby) · Yaws (Erlang)	

Source: [https://en.wikipedia.org/wiki/Web\\_framework](https://en.wikipedia.org/wiki/Web_framework)

In this course we are mainly interested in learning about **Apache web server** as well as **Web.py** and **Flask** web frameworks, both of which are based on Python scripting language.

## 1.2. Creating a Web Server Using Apache

Apache Web Server (or Apache http server) is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers. It was developed and currently maintained by Apache Software Foundation. Apache is an open source software available for free. It is estimated that it runs on over 60 percent of all webservers in the world.

As we mentioned before, a web server, such as Apache, is the software that receives your request to access a web page. It runs a few security checks on your HTTP request and takes you to the web page. Depending on the page you have requested, the page may ask the server to run a few extra modules while generating the document to serve you. Other types of Web servers include *Apache Tomcat* (used mainly for Java Servlets), *IIS* - The standard Microsoft web server for working with Microsoft .NET web apps, and *Nginx*.

*Apache HTTP Server* sometimes also called *Apache httpd*

We often hear people referring to *httpd server*. In reference to Apache, *httpd program* refers to the Apache HyperText Transfer Protocol (HTTP) daemon. Daemon is a program that runs in the background. In this case, httpd is the server program which handles the incoming requests.

Web servers, such as Apache can offer both Static and Dynamic contents. Static content is published to regular files on your server and handled using the simplest methods available to the web server. The advantages of static content are:

- it is the fastest and most efficient way to deliver content
- it does not require any code to execute or any databases to be accessed, which makes it the most secure way to deliver content
- it uses simple, clean URLs to address the content
- it takes best advantage of web caching systems, which further boosts performance
- it is compatible with every type of webserver technology

Dynamic content is generated for you at the time you request the page. The document you view exists only for you at that moment; if viewed by someone else at the same time, or by you at a slightly different time, you could get something different. Dynamic content is good for:

- pages whose content changes too quickly to easily republish it
- pages that display viewer-specific content (eg. user profiles)
- pages that display content conditionally (ie. member-only pages)

We discuss using **Web.py** and **Flask** web frameworks in the future labs!

## 2. Access your Pi

Before, we start installing Apache on the RPi, let's make sure we can access it remotely first.

Place your 32 Gbyte MicroSD card that you received from your instructor in the appropriate slot on the Pi. Power up the Pi using the Micro-USB adaptor. At this point you should see the RED LED. Connect the Pi to an active LAN line, using the 10/100 Ethernet LAN connector. Note that the GREEN LED on the LAN connector is should be on.

**Using SSH** access your Pi, as described in the previous lab. Let's assume it is `192.168.1.73`.

Assuming your Pi is connected to the same LAN, the IP address of the Pi is going to be within the same sub-domain (in this case `192.168.1.xx`). Run the following commands:

```
ping 192.168.1.73
arp -a
```

Run a few other Linux commands to ensure everything is ok.

### 3. Creating a Webserver

In this section we learn how to setup a Web server. As you know Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages. On its own, Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP. We first have to install Apache2:

First let's make sure we have installed all the updates in the Pi (this may take a few minutes to complete):

```
sudo apt-get update && sudo apt-get upgrade
```

Then, let's install the Apache software:

```
sudo apt-get install apache2
```

Now that we have installed Apache software, we need to start it:

```
systemctl restart apache2
```

Open a web browser and type the IP address of the Pi (e.g., 192.168.1.65). You should see the default page.

It is possible to change the default web page. To do this, go to `/var/www/html` directory and create a duplicate of the `index.html` file:

```
⇒ apache2 -ver
⇒ cd /var/www/html
⇒ sudo cp index.html index.html_back
⇒ sudo rm index.html
```

Using VI create a file called `index.html` and type HELLO WORLD! In there. Open a web browser and type the IP address of the Pi (e.g., 192.168.1.65). You should see your new `index.html` file.

Note that every time we change the `index.html` file, we may have to *restart* the Apache, as mentioned above. We can always STOP the web server, using the following command:

```
systemctl stop apache2
```

At this point, what happens if you try to access the web page?

Who *owns* the `index.html` file? Use `ls -al`

It is possible to change the ownership of `index.html` file to the user, instead of the Pi. This way we do not have to use **su** every time we need to change the `index.html` file. Therefore, use the following command:

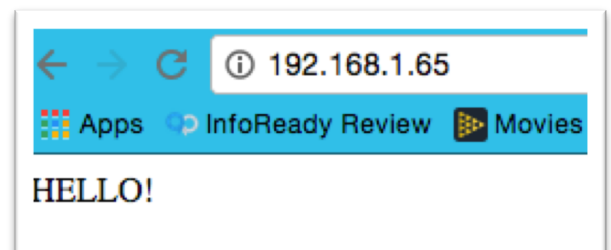
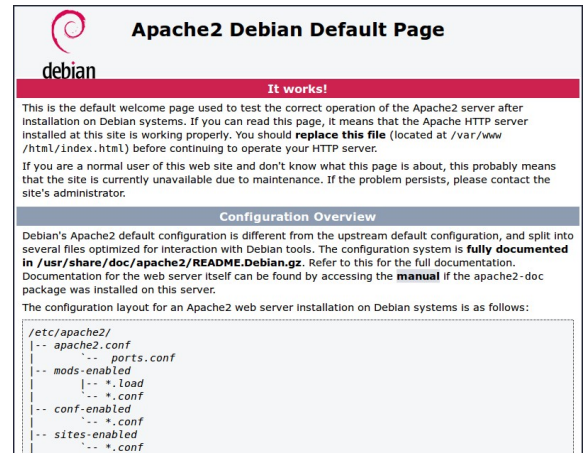
```
sudo chown ssuee: index.html
```


Check the ownership of your index file. Who owns it? Can you edit the file using VI tool without becoming the su?

Can you tell what version of Apache2 you have installed? What is the command you should use to know the version? When was this version built on your Pi?

**PROJECT A:** Read <https://www.w3schools.com/html/>. Make sure you understand the basics of HTML coding. Then correct the given code below to get the following display. Move your HTML code into your website that you just created. This is a great tool to check the errors in your HTML code:

<https://validator.w3.org/nu/#textarea>



<pre>&lt;!DOCTYPE html&gt; &lt;html lang="en"&gt;   &lt;head&gt;     &lt;title&gt;Hot Cocoa ☕/title     &lt;meta charset="utf-8"&gt;     &lt;style&gt;     body {       backgrnd-color: blue;       color: yellow;     }   &lt;/style&gt; &lt;/head&gt; &lt;h1&gt;Hot Cocoa Varieties&lt;/h1&gt; &lt;ul&gt;   &lt;li&gt;Mayan Cocoa   &lt;li&gt;Coconut Chai Cocoa &lt;/ul&gt; &lt;body&gt; &lt;/html&gt;</pre>	
Course code	Output

#### 4. Installing ftp Service

It is highly useful to be able to pass files back and forth between you Pi and your computer. Therefore, let's install an FTP server using vsftpd package. First things first, we need to install the vsftpd package.

```
sudo apt-get install vsftpd
```

By default vsftpd is configured for anonymous access with read-only permissions. We're going to change things so that it requires you to authenticate with a local user. Let's open the configuration file.

```
sudo nano /etc/vsftpd.conf
```

Note that you can also use Vi text editor. We want to change or uncomment the following values.

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
chroot_local_user=YES
user_sub_token=$USER
local_root=/home/$USER/ftp
```

You may want to just add the parameters if you don't find them. Make sure you add them at the end of the file. Save the file and check your edit. Make sure everything is in the `vsftpd.conf` file.

At this point restart the vsftpd service:

```
sudo service vsftpd restart
```

Now we need to create a couple folders in your local user's home folder. The ftp folder will be the root when you connect. The way vsftpd works, the root is not allowed to have write permissions on it, so we'll create a sub-folder inside the root called `files` which our local user will be allowed to write to.

This is where you'd upload/download files from with an FTP client. Note that you may have to use `sudo` option to accomplish the following.

```
mkdir /home/ssuee/ftp
mkdir /home/ssuee/ftp/files
chmod a-w /home/ssuee/ftp
chmod 777 /home/ssuee/ftp/files
```



Now you should be able to connect to your Raspberry Pi from any FTP client and start uploading/downloading files. From your home directory on your computer, type the following (use the IP address of your Pi).

```
ftp ssuee@192.168.1.xx
```

Then use *student* for password. Once you get in, use `cd`, `get`, `put`, commands to transfer files manually.

Upload a file into the `/ftp/files` directory. In your Pi, then move the uploaded file to the `html` directory:

```
sudo cp test_file.sh /var/www/html
```

Go to the `html` directory in your Pi and make sure the `test_file.sh` file is there.

## 5. Installing PHP Server

Let's first see what PHP is! PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML<sup>3</sup>. PHP pages contain HTML with embedded code that does "something" (in this case, output "Hi, I'm a PHP script!"). The PHP code is enclosed in special start and end processing instructions `<?php` and `?>` that allow you to jump into and out of "PHP mode."

In order to be able to take advantage of PHP scripting language we need to install it first. To install PHP run the following

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

Check the PHP version using `php -ver` command. Create new file called `index2.php` and change the content of it with the PHP example above. Go to your browser and type `192.168.1.65/index2.php`. You should see your new `index2.php` file.

Add the following PHP commands in your php file. Explain what they do.

```
echo date('Y-m-d H:i:s');
phpinfo();
```

**PROJECT B:** Carefully, review the tutorial in <https://www.w3schools.com/php/>. Fix the PHP code below such that you exactly get the shown output. Add the file into your webpage and display the output.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
      echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

```
<php
$txt = "Learn PHP";
$txt2 = "Engineering @ SSU";
x = 5;
y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 ;
echo "the sum of X and Y is: " . $x + $y)
>
```

### Learn PHP

Study PHP at Engineering @ SSU  
the sum of X and Y is: 9

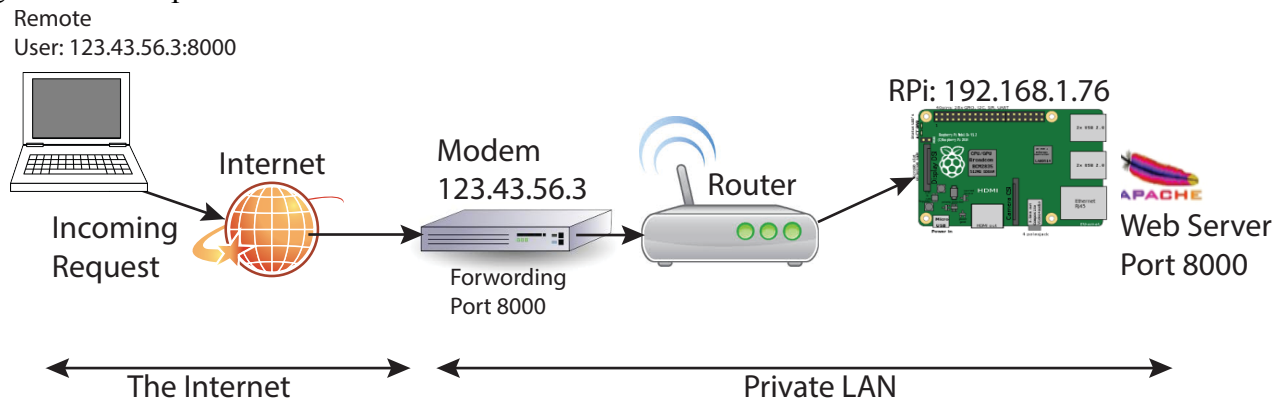
PHP Source code

Display

<sup>3</sup> See this <http://php.net/manual/en/intro-what-is.php> for a good tutorial

## 6. Port Forwarding

Let us learn a bit about port forwarding. In computer networking, port forwarding (PF) or port mapping is an application of network address translation (NAT) that redirects a communication request from one address and port number combination to another while the packets are traversing a network gateway, such as a router or firewall. As the result of PF a using outside the private network, LAN, can access a local server on the LAN. This is particularly desirable when the server does not have a static IP address. The figure below depicts this idea.



The key idea in PF is that the Modem (gateway or router that connects the LAN to the Internet through the ISP) must be configured such that it is aware which devices inside the LAN can be reached using designated port addresses. Every connection request includes a "port", in this case port 8000. Remember, the port is just a number, and it's part of how a computer knows which application to access and what type of packet has been received. IANA (The Internet Assigned Numbers Authority) has specified how different ports are designated to different applications. For example, port 80 is used for HTTP<sup>4</sup>. When port forwarding is enabled, all incoming connections with a matching port number will be forwarded to the internal computer with the specified IP address designated to handle that port (application).

Some routers have UPnP port forwards capability. That is via software the router automatically forwards the traffic to the computer based on the designated port (in the case above packet with port 8000, automatically is forwarded to RPi).

Another simpler approach to achieving port forwarding is through establishing Faux-DMZ in the router. In such cases the router has a feature called DMZ (Demilitarized Zone), which is similar to having a network security configuration. The DMZ on home routers is often referred to as faux-DMZ because it lacks the features of an actual DMZ (faux means fake!). Using the Faux-DMZ feature all incoming connection requests can be sent to a specific machine. This approach may not be sufficient if we have multiple computers that should be accessed. It is important to understand that when we DMZing a computer inside your network (setting it as the DMZ destination) that particular machine loses will be accessible to incoming connections from the Internet, and thus need to be fully secured. In the case of regular port forwarding or UPnP port forwarding, only the port is exposed directly and not the entire machine.

<sup>4</sup> For more information refer to IANA's list of ports: <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=2>



## 7. Play Audio Files

In this section we learn how to enable the analog audio jack and play a WAV audio file. Perform the following steps<sup>5</sup>:

- Make sure you SSH into your Pi.
- In your Raspberry Pi download from WilhelmScream.wav file from <https://ia802706.us.archive.org/20/items/WilhelmScreamSample/WilhelmScream.wav>. This can be done using the following command:  
Curl  
`https://ia802706.us.archive.org/20/items/WilhelmScreamSample/WilhelmScream.wav > wilhelmscream.wav.`
- Verify and make sure you have the file in your directory
- Run the following commands. These commands install ALSA (Advanced Linux Sound Architecture) drivers :
  - `sudo modprobe snd-bcm2835`
  - `sudo apt-get install alsa-utils`
- At this point you have to choose which audio out put to use: 0=Auto, 1=Analog, 2=HDMI. We choose Analog audio jack. Make sure you have your speaker microphone connected to the Pi:
  - `sudo amixer cset numid=3 1`
- Run a test on the speaker. You should hear a rushing (white) noise on your microphone.
  - `sudo speaker-test`
- Now play the WAV file:
  - `sudo aplay wilhelmscream.wav.`

**PROJECT C:** Write a program such that when a switch is pressed it says something or plays an audio. You can create your own audio file here (<http://www.fromtexttospeech.com/>). **Show your movie to the instructor!**

## 8. Connecting the Pi to a camera

In this section we would like to connect the Pi to a camera. For this section I used a typical USB-based PC webcam (Rosewill Webcam <sup>6</sup>). In your case, you may experience poor quality pictures with a USB webcam. Note that some webcams are more reliable than others. If the problem persists, ensure your system is up to date. Also try other webcams, but you'll get the best performance from the Raspberry Picamera module <sup>7</sup>.

In your Pi run the following commands:

```
⇒ cd
⇒ mkdir webcam
⇒ sudo apt-get install fswebcam
⇒ fswebcam image1.jpg
```

if you open the directory you should see an image in image.jpg file. Cool huh?

You can change the size of the image:

```
⇒ fswebcam -r 1280x720 image2.jpg
```

You can also remove the banner:

<sup>5</sup> For more information read Chapter 3 of Raspberry Pi Projects by Norris (eBook)

<sup>6</sup> See this link: <http://www.newegg.com/Product/Product.aspx?Item=N82E16826193057>

<sup>7</sup> See the link for the camera: <http://www.raspberrypi.org/help/camera-module-setup/>

⇒ fswebcam -r 1280x720 --no-banner image3.jpg

Open image1-image3 and compare them.

Let's create a bash file such that it takes a snapshot every 5 seconds and save the image as jpg. The file's name will be the exact date-time (2016-10-02\_221503.jpg). Below is an example file you can modify for proper operation. When you took 10 snapshots look at all of them and observe the difference. The best way to do this is to take snapshots of a working digital or analog clock.

```
#!/bin/bash
for <whatever>
do
  DATE=$(date +"%Y-%m-%d_%H%M%S")
  echo "Welcome $i times"
  sleep <something> # Waits 5 seconds
  fswebcam -r 1280x720 --no-banner <filename>
done
```

Sample script to take snapshots every 5 seconds.


It is also possible to stitch the images together and create a movie! Follow the steps below:

```
⇒ cd webcam
⇒ mkdir temp // make a copy of all the jpg files in the temp
  directory. Do not MOVE them, just copy!
⇒ cd temp
⇒ mogrify -resize 800x800 *.jpg
⇒ convert temp/*.JPG -delay 10 -morph 10 temp/%05d.jpg
⇒ rm 2016* // this will remove all the old jpg files
⇒ convert *.jpg output.mpg
```

**PROJECT D:** At this point you should be able to easily connect your Pi to a switch as described above such that if the switch is pressed a picture is taken! Imagine what you can do with this! Create a movie with at least one snapshot of yourself in it. **Show your movie to the instructor!**

**PROJECT E (Graduate students only – Extra Credit for UG Students interested to learn PHP):** Carefully review lessons 1-12 in this site (<https://www.mikedane.com/web-development/php/> - many thanks to md for providing these great lessons].

Create a HTML form and a PHP code similar to the one shown in the assignment to display the dynamic web page shown in the table. Note that in this case, the user can enter any value and the values will be displayed in the page as the submit button is pressed. Use the provided example code and remove all the errors. **Make sure you can demonstrate it.**

<pre> &lt;form action="mysite.php" method="GET"&gt;   Color: &lt;input type="text" name="color"&gt;   Plural Noun: &lt;input type="text" name="pluralNoun"&gt;   Celebrity: &lt;input type="text" name="celebrity"&gt;   &lt;input type="submit"&gt; &lt;/form&gt;  &lt;?php   \$color = \$_GET["color"];   pluralNoun = _GET[pluralNoun];   \$celebrity = _GET["celebrity"];    echo "Roses are \$color";   echo "\$pluralNoun are blue"   echo "I love \$celebrity"; ?&gt; </pre>	
PHP Code to generate the form – with errors	The form as displayed as a web page

## F. Credits

Special thanks to online resources and all SSU students who assisted putting together this lab.

## E. References