Department of Engineering Science

Using PHP to Plot PART I

Updated: 11/6/18

A. Objectives

- Learn about Dynamic URL Request
- Learn about cURL and HTTP Request Methods
- How to access and FTP server automatically
- How to use sshpass and scp
- Understanding the differences between ssh and FTP servers
- How to send sensor data from RPi to a remote server and saving it in a CSV file
- How to use PHP to parse the URL with parameter
- How to use PHP to read a CSV file on the server
- How to plot the sensor data from a CSV file using phpGraph

B. Time of Completion

This laboratory activity is designed for students with very little knowledge of protocols and it is estimated to take about 3-4 hours to complete. You can use your RPi to communicate with the server

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. Access to a server with PHP and FTP

D. Pre-Lab

1. Review ftp and ssh
2. Review cULR and its applications: https://curl.haxx.se/docs/httpscripting.html
3. Read about HTTP Protocol: see (https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview)
4. Read about URL Query String (https://en.wikipedia.org/wiki/Query_string)

This is a TWO-part lab! Make sure you complete PART I before you start Part II

E. Lab

In this lab we become familiar with how to send data to a server from our RPi and then plot the data on the server so remote users can view the plot. This is a relatively complex process and we must make sure we understand each intermediate step. Figure 1**,** below, shows the basic physical and logical interfaces between the RPi, server, and remote user. We divide the design process into multiple steps, marked as **(1)-(5)**:

- In Part I of this lab we focus on steps (**1b), (2), (3), (4) and (5a)** – no RPi needed.
- In Part II of this lab we focus on steps (**1a), (2), (3), (4) and (5b)** – must use RPi.

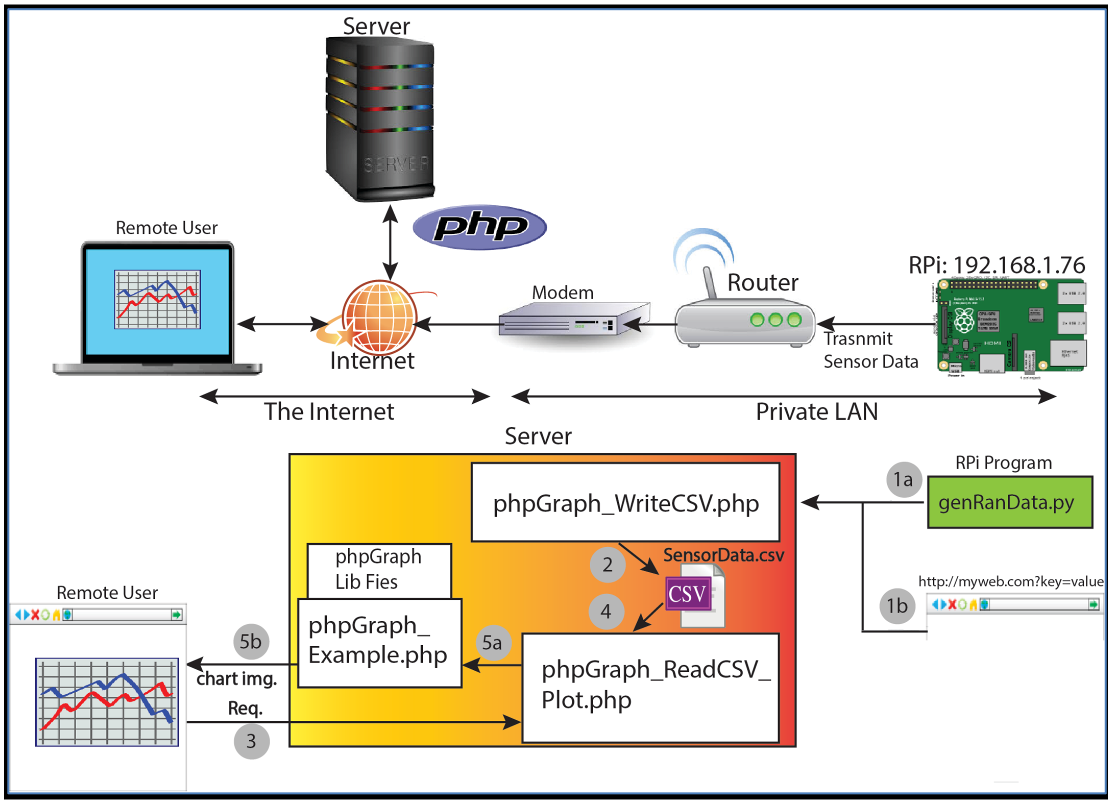


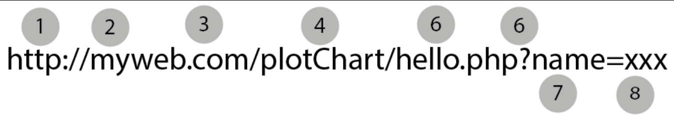
Figure 1: System interface.

A) Dynamic URL

A dynamic URL (Uniform Resource Locator) is the address of a Web page with content that depends on variable parameters. A dynamic URL can often be recognized by the presence of certain characters or character strings that appear in the URL (visible in the address bar of your browser). For example, when you request www.myweb.com/plotChart/hello.php?name=xxx, Figure 1 **(1a)**, depending on the value of xxx, the content of the requested web page changes.

In the previous labs, we learned that using PHP scripting language it is possible to read the URL parameters and change the content of the web page accordingly. Note that the parameters may be already present in the URL itself (for example, when you do a search on Google page, a URL with the appropriate parameters is automatically generated[1]) or they may be the result of user input. In either case, the parameters in URL appear after "?" in form of *key-value (or Name/Value) pair*. Consider the dynamic URL shown in Table 1:

[1] Try this: https://airnow.gov/index.cfm?action=airnow.local_city&cityid=320



1. Protocol 2. Domain Name 3. Top-level Domain 4. Directory Path	5. Page / File Name 6. Starting notation for Query Parameter 7. Key 8. Value for the Key

Table 1. URL Dissection.

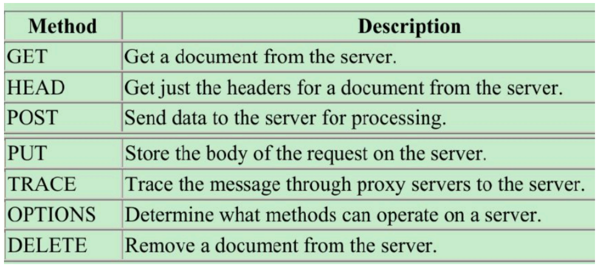
A. Answer the following questions:

1- Use the following URL in your browser and explain what is happening. What happens if you change the values? What happens if you change the keys?
http://aitislab.com/phpf/PHP_Files/plotChart/phpGraph_WriteInCSV.php?status=OFF&humid=69&sen_id=101&temp=7&light=49

2- Carefully examine the following URLs. What is the difference between the responses you receive?

- .http://aitislab.com/phpf/PHP_Files/plotChart/phpGraph_WriteInCSV.php?status=OFF&humid=69&sen_id=101+2&temp=7&light=49
- http://aitislab.com/phpf/PHP_Files/plotChart/phpGraph_WriteInCSV.php?status=OFF&humid=69&sen_id="101 2 43"&temp=7&light=49

B) HTTP Request Methods

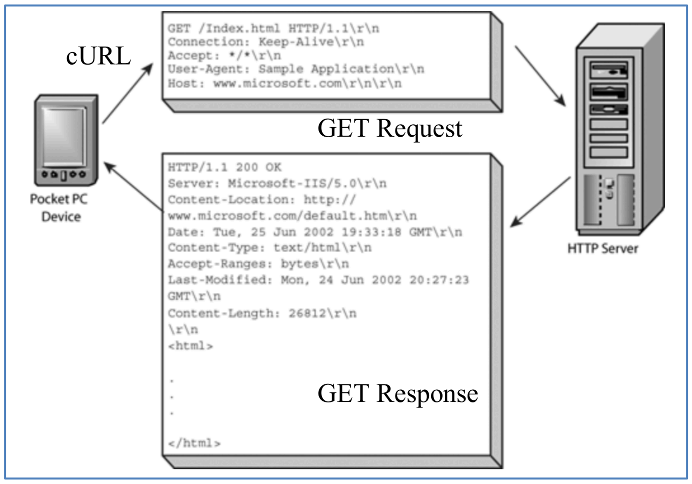


Method	Description
GET	Get a document from the server.
HEAD	Get just the headers for a document from the server.
POST	Send data to the server for processing.
PUT	Store the body of the request on the server.
TRACE	Trace the message through proxy servers to the server.
OPTIONS	Determine what methods can operate on a server.
DELETE	Remove a document from the server.

HTTP is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and a **client-server protocol,** which means requests are initiated by the recipient, usually the Web browser. Using HTTP/1.1 there are different common request methods that can be made to the server. The following figure lists some other HTTP request methods; note that these method names are case sensitive and they must be used in uppercase [2].

Two commonly used methods for a request-response between a client and server are: GET and POST.

- GET - Requests data from a specified resource (server). The URL could itself refer to a web page, an image, or a file. The client issues a GET request to the server and receives the document it asked for. Here is an **example** how you can make request HTTP using GET method:
  `curl http://aitislab.com`
- POST - Submits data to be processed to a specified resource. This way the client sends the data separated from the URL and thus you won't see any of it in the URL address field. **Example:**



```
GET /Index.html HTTP/1.1\r\n
Connection: Keep-Alive\r\n
Accept: */*\r\n
User-Agent: Sample Application\r\n
Host: www.microsoft.com\r\n\r\n
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0\r\n
Content-Location: http://
www.microsoft.com/default.htm\r\n
Date: Tue, 25 Jun 2002 19:33:18 GMT\r\n
Content-Type: text/html\r\n
Accept-Ranges: bytes\r\n
Last-Modified: Mon, 24 Jun 2002 20:27:23
GMT\r\n
Content-Length: 26812\r\n
\r\n
<html>

.
.
.

</html>
```

Figure 2: Request/Response Using GET Method

[2] For more information see https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

`curl --data "status=OFF"` `http://aitislab.com`

- HEAD - Submits data to be processed to a specified resource. This way the client sends the data separated: `curl -X HEAD -i` `http://aitislab.com`

Remember: Curl is a command line tool for doing all sorts of URL manipulations and transfers from your *terminal*. In this lab, we focus on using cURL for making HTTP requests. From a terminal, you can invoke `curl --help` or `curl --manual` to get basic information about it.

B. Answer the following questions:

1. Run the cURL example for POST from your RPi and record the captured *response*. Take a snapshot of the POST response (as shown in Figure 2).
2. Run the cURL example for HEAD from your RPi and record the captured *response*. Take a snapshot of the HEAD response (as shown in Figure 2).

HINT: To complete the above questions you can do the following steps:

- Go to *Introduction to RPi lab*. Follow *Method 5* and from your machine remotely connect to your headless RPi using VNC. Make sure you can see the GUI desktop.
- In the GUI Desktop open a terminal and run the following command: `sudo wireshark-gtk &`. When you use `&` it indicates that you want the process to run in the background. Do `ps -all`. Check out the Process ID (PID) for Wireshark. You can kill the process (exit) by using `kill -9 <PID>`.
- After you invoked the Wireshark application, select the interface and start it. Then, run the cURL command in a terminal. Note that you can always filter HTTP packets.

C) Automate the FTP Uploading

Before you continue, it will be tremendously helpful if you figure out how to automatically upload a file into your Server. Refer to the previous lab and make sure you can write a simple script to perform automatics uploading.

There are other FTP clients that you can download on your machine (Windows, MAC or Linux) that allow to easily perform FTP operation. One such program is called *FileZilla Client* (https://filezilla-project.org/download.php). Download the program and run it. In FileZilla go to File→ Site Manager and set up the ftp site parameters as shown in the figure above. Note that the Port number must be 21.

C. Answer the following questions:

1- Using Wireshark, capture the packets as you attempt to `FTP` into the server. Filter the FTP captured packets. Go to Analyze→Follow→TCP Stream. What is the code when the connection is disconnected? Do you see the password?

2- Is it possible to delete a file on an FTP site? If so, what is the command?

3- Is possible to run `vi` or `nano` on FTP server so we can edit a file remotely?

D) Automate SSH Uploading into RPI

Sometimes we need to transfer files to the RPI using SSH, automatically. One way to perform this is using a utility called `sshpass`. The `sshpass` utility is designed for running `ssh` using the mode referred to as "keyboard-interactive" password authentication. Your machine (Linux or Mac) may not have sshpass utility. In this case, you may need to install it:

```
$ sudo apt-get install sshpass
```

For windows machines, you can always use Putty[3] (`putty -load "host" -l username -pw password`)

From your machine, you can run the following command to copy a file **from** your RPi **into** your machine [4]:

```
sshpass -p student scp -r  ssuee@192.168.1.75:/home/ssuee/x_file_on_RPI
/Users/yourname/Desktop/y_file_on_computer
```

You can also automatically copy a file from your machine into your RPi:

```
sshpass -p student scp -r   /Users/yourname/Desktop/y_file_on_computer
ssuee@192.168.1.75:/home/ssuee/x_file_on_RPI
```

Note that it is possible to create an *alias* (assuming you are using a Linux based machine) using the following command:

```
alias sshlogin='sshpass -p student scp -r
ssuee@192.168.1.75:/home/ssuee/x_file_on_RPI   /Users/yourname/Desktop/y_file_on_computer'
```

In this case `sshlogin` is just a generic name, more like a shortcut name and has no particular significance. Form now on, you can just use `sshlogin` instead or using the long format.

D. Answer the following questions:

1- Using Wireshark, capture the packets as you run `sshpass` command and copy a file from your RPI into your machine. Filter the captured packets using SSH. Answer the following questions:
   - How is SSH application encapsulated?
   - Go to Analyze→Follow→TCP Stream? Can you see the password?

2- Is it possible to delete a file on the remote machine using SSH? If so, what is the command?

3- Is possible to run `vi` or `nano` commands to edit a file using SSH?

4- In general, what are the key differences between FTP and SSH? Which one has more features?

E) Saving the URL Parameters in a CSV File on the Server

In this section, we learn how to receive the URL (query string) and save the URL parameters in a CSV file using a PHP program running on the Server. Refer to Figure 1 **(2).** Using the information you learned in the previous lab, log into the `aitislab.com`. Go to your directory and create a new subdirectory called `plotchart`. Create a PHP file and called it `phpGraph_WriteInCSV.php`.

The complete code for this file is provided in Appendix A. Notice that this code is a combination of HTML and PHP code, yet the file extension is PHP. Using the given PHP code the keys in each incoming request (URL) is parsed and time stamped and saved in a CSV file called `SensorData.csv`.

However, you should first manually generate the CSV file: using Excel, create an empty file called `SensorData.csv` and save it as `MS-DOS Comma Separated (.csv)`. Upload the file in your `plotChart` directory on the server. Make sure you pick the right extension!

[3] Latest version of Putty can be downloaded form here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
[4] For more information about secure copy (scp) see here: https://www.computerhope.com/unix/scp.htm

At this point manually push different data points to your CSV file. You can do this using a browser or CURL command, as described above. Alternatively, you can use the `genRanData.py` (as explain in **Part II**) to automatically upload data points into the CSV file using your computer or RPi.

FTP to the server and make sure your new request has been added to the `SensorData.csv` file. You can view files on the FTP server by using `more` or `less` commands. In order to quit viewing the file just press `q` and you will go back to your FTP prompt.

F) Adding Time to the Received Sensor Data

We usually like to add time to the data points that we receive. One way to do this is to make the RPi to timestamp every set of sensor data that it sends to the server. However, this may require hardware modification and adding a real-time chip. Another cheaper approach is to get the clock from a server:

```
curl -v http://aitislab.com
```
or
```
curl -X HEAD -i  http://aitislab.com
```

Note that in this case the response contains many parameters including the time and date:

```
52197-SALZ20$ curl -v http://aitislab.com
* Rebuilt URL to: http://aitislab.com/
*   Trying 107.180.44.153...
* TCP_NODELAY set
* Connected to aitislab.com (107.180.44.153) port 80 (#0)
> GET / HTTP/1.1
> Host: aitislab.com
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sun, 15 Oct 2017 21:36:21 GMT
```

It is also possible to receive the time from an NTP server [5]. The NTP server is the timing source for your computer; this is how your computer gets the correct timing. One way to get the time from the NTP server is to use the following command from the terminal:
```
ntpq -c 'rv 0 clock' localhost
```

Note that the 'localhost' refers to getting the time from NTP daemon running locally on your machine. If NTP command does not work, install it first : `sudo apt-get install ntp`

It is also possible to write a python code to invoke and NTP request. See the previous lab for more information.

In this lab, we like to send the sensor data to the server and let the server timestamp the data. Thus, we use a PHP code on the server side. The following PHP code, as shown in Appendix A, creates a local timestamp in this format: `Y-m-d H:i:s` ; something like this: *`2017-10-12 17:54:21`*

[5] Read about NTP servers here https://en.wikipedia.org/wiki/Network_Time_Protocol - we will discuss them later.

```
<?php
date_default_timezone_set("America/Los_Angeles");
      $today = date("Y-m-d H:i:s"); echo $today;
      ?>
```

E. Answer the following questions:

1- In the PHP code above, can you change the format to `d-m-Y H:i:s`? How will you change the code if you line in NY City?

2- Open Wireshark and enter `ntp` in the filter. Wait for a while. Do you see any NTP packets? If so, which timing source is your machine trying to synchronize with?

3- While running the Wireshark and filtering for NTP packets, execute the following command:

```
ntpq -c 'rv 0 clock' 193.79.237.14
```

Answer the following:

3.a - How is the NTP application encapsulated (in terms of layers)?

3.b - What is the Port address for NTP application?

3.c - Is it possible to know the name of the machine with IP address `193.79.237.14?` How?

E. Submissions

Submit your answers to all the questions. Please make sure you include the questions and properly number them (e.g., A.1 or C.3). All snapshots much have full explanation. Do not include any figures or snapshots without description. Answers without questions or snapshots without description do not receive credit.

F. Credits

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

G. References

[1] none.

Appendix A

```
<html>
<body>
<!-- ********* Create current time *********  -->
<!-- prints something like: 2001-03-10 17:16:18 (the MySQL DATETIME format)
 Mountain Standard Time (MST) Time Zone        -->
date_default_timezone_set("America/Los_Angeles");
Current Time: <?php date_default_timezone_set("America/Los_Angeles");
                           $today = date("Y-m-d H:i:s"); echo $today;
                           ?> <br>

<!-- ********* Parse URL *********  -->
<!--  Format $key=$_GET["key"] -->
<br>  Sensor Data Received: <br>
       Sensor ID: <?php $sen_id=$_GET["sen_id"];echo $sen_id ?><br>
       Temperature  (C): <?php $temp=$_GET["temp"]; echo $temp ?><br>
       Humidity (%): <?php $humid=$_GET["humid"]; echo $humid?><br>
       Light (Lux): <?php $light=$_GET["light"]; echo $light?><br>
       Status (On/Off):  <?php $stat=$_GET["status"]; echo $stat?><br>

<br> Sensor Data is Recorded in SensorData.csv File on the Server.....<br>
<!-- ********* Save in CSV FIle *********  -->
<?php
$list = array // creating an array
(
"$today, $sen_id, $temp, $humid, $light, $stat"
);
$file = fopen("SensorData.csv","a"); // keep appending to an existing file
foreach ($list as $line)
  {
  fputcsv($file,explode(',',$line)); // separate by , - each entry will be in " "
  }
fclose($file);
?>
</body>
</html>
```

`phpGraph_WriteInCSV.php` program listing. This code must be uploaded into the server in your `plotChart` directory.

Appendix B

```
<?php
        $row = 1;
        if (($handle = fopen("SensorData.csv", "r")) !== FALSE) {
               while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
                       $num = count($data);
                       //echo "<p> $num fields in line $row: <br /></p>\n";
                       $lastEntry= $row++;                   // this represents each data entry
                       for ($c=0; $c < $num; $c++) {
                               echo $data[$c]; // each c value represents one field of data
                                                             // in this case we have the following:
                                                             // time, id, temp, humid, light, status
                               $myArray[$row-2][$c] = $data[$c]; // this contains the sensor data
                       }
                       echo "<br />\n";
               }
               fclose($handle);
        }

        // Let's parse the time & temp data;
                 echo "***** <br />\n";
                 $arr = array();
                       $myData = array();
                       $mynumber = array();
                 $sensor_ID = $myArray[0][1];
                 echo $lastEntry;
                 for ($col = 0; $col < $lastEntry; $col++) {
                       $myTime = $myArray[$col][0]; // copy time
                       $myData[] = $myArray[$col][2]; // copy temperature
                       $mynumber[] = $col;
                 }
            $arr = array_combine($mynumber, $myData);
?>
<html>
        <head>
               <title>Data Visualization from the CSV file</title>
        </head>
        <body>
               <h3> List is completed! </h3>
        </body>
</html>
```

`phpGraph_ReadCSV.php` program listing. This code must be uploaded into the server in your `plotChart` directory.