Department of Engineering Science

Using PHP to Plot
PART II

Updated: 11/1/18

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.

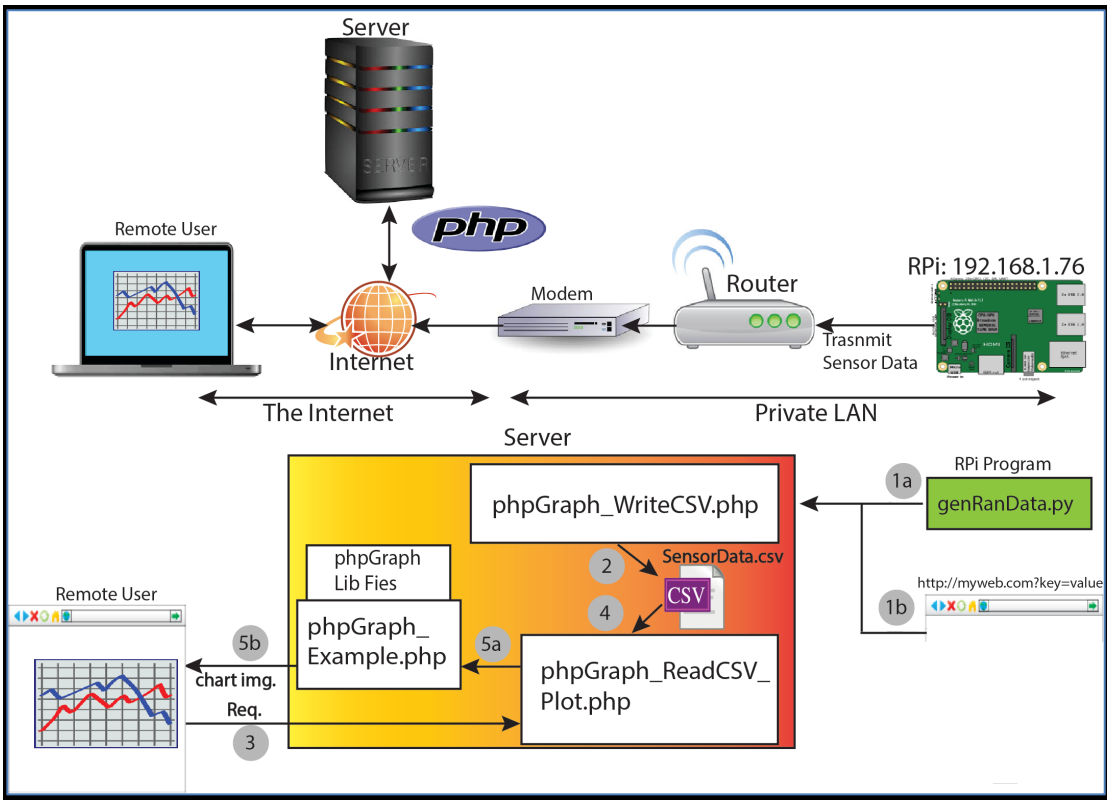

Figure 1: System interface.

Important note: In order to continue you must have completed part I first.

A) Transmitting Data from the RPi.

In order to transmit sensor data from the Pi to the server we need to write a simple python program to read the GPIO ports and send the data. Refer to Figure 1 **(1b)**. In this case, for simplicity, we generate *random()* function to simulate sensor data. Let us examine the code listing in Figure 2:

(1) Include networking and random libraries.
(2) Use the look if you need to make repeated requests Make sure the `time.sleep (200)` command is set.
(3) Create some random sensor values.
(4) Prepare a URL request. Note that the request can be a POST or GET.
(5) Make the request.

```python
#!/usr/bin/env python
# Generating Random Numbers (PYTHON 2)

import random                                          1
import requests, urllib

# while True: # You can make a loop here               2

# define three fields - create random numbers 1-100
temRand=random.randrange(1, 100, 2)
humRand=random.randrange(1, 100, 2)                    3
ligRand=random.randrange(1, 100, 2)

parameters = urllib.urlencode({'sen_id':101,'temp':temRand,'humid':humRand,
    'light':ligRand,'status':"OFF"})
                                                       4
# Construct a request to POST or GET
req = requests.post(
    'http://aitislab.com/rrr/xxx/yyy/phpGraph_WriteInCSV.php',
    params=parameters)
# Making the request                                   5
req.url

print "This is what you are posting: ", parameters
#time.sleep(200)  # Must be at least 15 seconds
```

Figure 2: Details of genRanData.py to simulate sensor data.

The above code will generate a URL request similar to below:
http://aitislab.com/rrr/xxx/yyy/phpGraph_WriteInCSV.php?status=OFF&humid=95&sen_id=101&temp=1&light=15

Note that in this case we are creating a URL with parameter. As we mentioned before, the parameters appear after "?" in form of *key-value (or Name/Value) pair*. For example: `{"temp": "65"}` is a simple example, where the *key* is temp and the *value* is 65. In our case the key-value appears in a more complex way:

```
{"send_id" : "101", "temp" : "65", "humidity" : "6", "light" :
"400"}.
```

We often refer to this data format as JSON format. JSON is a text-based open standard for data exchange. Because JSON format is text only, it is considered to be it is language independent.
You can use the following site to test and ensure your code operates correctly:
http://aitislab.com/phpf/PHP_Files/plotChart/phpGraph_WriteInCSV.php

A. Answer the following questions:

1- In the program listing shown in Figure 2 change `requests.post()` to `requests.get()`. What do you think will happen?

2- Using JSON format, is there a difference between the following data entries? If so, what is the difference?

```
a) {"send_id" : "101", "light" : "400"}
b) {"light" : "400", "send_id" : "101"}.
```

B) Plotting the Data

Data Visualization (plotting the data) is a key component of Internet-of-Things (IoT) or Machine-to-Machine (M2M) technologies. In a Server-Client model, data visualization can be performed on either the server side or the client side. When there are many users accessing the server, apart from using the server's processing power, image based charts can significantly increase the bandwidth consumption of servers. Therefore, in some cases, it may be more efficient and faster to perform data visualization on

the client side. This is done by using client side components such as JavaScript based programing. In such cases, the client browser will fetch the JSON or XML data from the server and render the charts locally (on the client machine). Client side components also have a greater scope of customization. They are often interactive, support animation and can easily blend into the design of your application. If visualization is performed on the server side, usually PHP wrappers are utilized to perform data visualization.

However, before we visualize the data, the data must be stored somewhere. One common approach to sensor data storage is to use a *database*. There are many different packages that can perform databasing function such as MySQL, Mango, etc. Another approach to store the data is storing all the incoming data into a large file, such as a CSV file. Generally speaking, using a database is much more powerful in terms of manipulating data, especially when the data size is very large.

There are a number of issues with storing the data in a CSV file. For example, it is not easy to plot only a part of the data, or delete an entry or replace an entry. However, using a database such operations are very simple and can be accomplished by one or two simple database command.

For this laboratory activity, we use a CSV file to store the data on the server size. Furthermore, we use PHP-base scripts to visualize the data. There are a number of available PHP charting options:

- ChartLogix PHP Graphs (http://www.binpress.com/app/chartlogix-php-graphs/1265)
- pChart (http://www.pchart.net/)is one of the few native PHP charting libraries that is still under active development
- JpGraph (http://jpgraph.net/) is an Object-Oriented Graph creating library for PHP5 (>=5.1) and PHP7.0 The library is completely written in PHP and ready to be used in any PHP scripts (both CGI/APXS/CLI versions of PHP are supported).
- PHP Graph is yet another very simple PHP based charting tool[1]. (http://www.ebrueggeman.com/phpgraphlib/examples)

Regardless of which charting tool we use, the general idea is to utilize some type of pre-existing library. In this case, we create a PHP sensor data array and send the array to the library files in order to generate our chart and visualize the data on the browser.

For the rest of this exercise we focus on *PHP Graph*, a set of libraries to plot the data using PHP. The disadvantage of PHP Graph is that it does not generate very rich interactive graphs such as these: (https://www.highcharts.com/demo).

So, let's start…..

Copy the code below in `phpGraph_Example1.php` file and upload it in your `plotChart` directory on the server, as we discussed in the previous lab (Part 1). Go to class web site and download the zip file for this lab – this is marked as Related Resources for Part 2 [2]. Extract the file. Upload all THREE files into your `plotChart` directory on the server. From a browser set URL to http://aitislab.com/rrr/xxx/yyy/phpGraph_Example1.php. You should see the line chart.

B. Answer the following questions:

1- Examine the chart's characteristics:
2- How many data points do you have in the chart?
3- What happens if you change `PHPGraphLib(650,200)` to `PHPGraphLib(450,100)`?

[1] For more information see https://www.codediesel.com/php/6-excellent-charting-libraries-for-php/
[2] Here is the link: http://web.sonoma.edu/users/f/farahman/sonoma/courses/es465/lab/basiclab/RPi_Released/files/resources/phpgraphlib_v2.31%202.zip

4- What is `setGoalLine(.0025)` for?

5- What happens if `setLine(true)` is false?

```
<?php
include('phpgraphlib.php');
$graph = new PHPGraphLib(650,200);
$data = array("1" => .0032, "2" => .0028, "3" => .0021, "4" => .0033,
"5" => .0034, "6" => .0031, "7" => .0036, "8" => .0027, "9" => .0024,
"10" => .0021, "11" => .0026, "12" => .0024, "13" => .0036,
"14" => .0028, "15" => .0025);
$graph->addData($data);
$graph->setTitle('PPM Per Container');
$graph->setBars(false);
$graph->setLine(true);
$graph->setDataPoints(true);
$graph->setDataPointColor('maroon');
$graph->setDataValues(true);
$graph->setDataValueColor('maroon');
$graph->setGoalLine(.0025);
$graph->setGoalLineColor('red');
$graph->createGraph();
?>
```

`phpGraph_Example1.php`. Simple PHP code to plot a line.

It is possible to read the data from a CSV file instead of directly inserting it into the PHP file. Review the `phpGraph_Example1.php` and notice how it is different with the previous example. The key difference is having:

```
$data = unserialize(urldecode(stripslashes($_GET['mydata'])));
```

In this case the data that is being plotted is being received by a variable called `mydata`. The method, in which the file receives the data is via an HTTP request.

```
<?php

include('phpgraphlib.php');
$graph = new PHPGraphLib(500,350);

// Read from a file
$data = unserialize(urldecode(stripslashes($_GET['mydata'])));

$graph->addData($data);
$graph->setTitle('Random Data From RPi');
$graph->setGradient('red', 'maroon');

// make it line or bar
$graph->setBars(false);
$graph->setLine(true);
$graph->setDataPoints(true);

// Draw a Goal Line
$graph->setGoalLine(.0025);
$graph->setGoalLineColor('red');

$graph->createGraph();
?>
```

`phpGraph_Example2.php`. Simple PHP code to read from a file.

Therefore, in order to provide the data we need to change the HTML portion of the `phpGraph_ReadCSV.php` program listing as described in the previous lab.

```
<head>
<title>Data Visualization from the CSV file</title>
</head>
<body>
<h3>This is where I want to display my graph</h3>
<img src="plot_example.php?mydata=<?php echo urlencode(serialize($arr)); ?>" />
</html>
```

Modified `phpGraph_ReadCSV.php` program listing. Let's call this new file `phpGraph_ReadCSV_AndPlot.php` This code must be uploaded into the server in your `plotChart` directory.

From a browser set URL to http://aitislab.com/rrr/xxx/yyy/phpGraph_ReadCSV_AndPlot.php. You should see the line chart of all values in your CSV file. Refer to Figure 1 **(5)**.

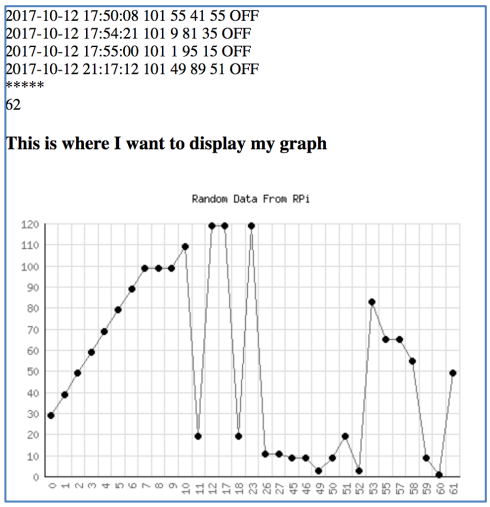

Figure 3: Data chart generate.

C) PROJECT: Plot Your Own Data

You are now ready to create your own sensor data set and plot them using PHP. Your design must do the following:

1. As you press a switch on the RPi the following key-value appears must be generate and send to the server and recorded in a file called `SensorData.csv`:

```
{"sen_id" : "x", "temp" : "y", "humidity" :
"z", "light" : "w"}.
```

Note that the `sen_id` must be your RPi's MAC address. We are assuming x,y,w,z are random integer values between 1-100.

2. On the server create a PHP file called `phpGraph_ReadCSV_AndPlot.php` such that when it is called, at least two separate plots are generated.
   a. One plot shows the temperature and the other one how show the light sensor data.
   b. Your x-axis can be {0,1,2,..}, similar to Figure 3.
3. Calculate the overall average of light and temperature and display the values under each chart.

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.

You must also demonstrate Part C.1-C.3 (50 points). You should show two separate plots. Every time a switch is pressed a new value should be plotted on each graph.

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.