

Lab 5

Install a web server (Apache + PHP + MySQL+phpMyAdmin)

Once you've configured, launched and tested your Raspberry Pi, chances are you'll want to use it as a web server.

Why use the Raspberry Pi as a web server?

But why use the Raspberry as a web server, rather than using service providers specializing in web hosting?

First of all, from an economic point of view, it's important to remember that web hosting services aren't free, and that you'll have to pull out your wallet every month/year. Unlike the Raspberry, which only requires a connection.

By choosing the Raspberry, you can modify your services as you wish (e.g. disk size, database hosting, etc.), which is generally not the case with specialized hosting providers, who often sell shared hosting with little configuration capacity.

The components

- Raspberry Pi
- Micro SD card with Raspberry PI OS
- USB Type-C cable

Step 0 : removing all previous installation of Apache, PHP, Maria-DB and phpMyAdmin (if exists)

Since you share machines with your classmates, there's a good chance that the servers listed below have already been installed.

We're going to start by removing the services that are already installed.

To remove apache:

```
pi240@raspberrypi240:~ $ sudo apt purge apache2*
pi240@raspberrypi240:~ $ sudo rm -R /var/www/
```

Purge will remove packets and dependencies and all configuration files will be removed too. The Shell (bash) considers an asterisk "*" to be a wildcard character that can match one or more occurrences of any character, including no character.

To remove php and phpmyadmin:

```
pi240@raspberrypi240:~ $ sudo apt purge php*
```

To remove mySQL:

```
pi240@raspberrypi240:~ $ sudo apt purge mariadb*
```

Step 1: Installing the Apache server

Today, Apache is the most widely used web server, with a market share of around 60%. Apache even has its own license, used by many other projects. Moreover, the massive use of Apache (which has become the standard for web servers), coupled with its great popularity, has led to a tremendous abundance of documentation, courses and other books dealing with its use, from installation to security.

Before installing the server, let's make sure our machine is up to date. To do this, we need to have administrator rights, either by logging on as root, or via the sudo command.

```
pi240@raspberrypi240:~ $ sudo apt update
pi240@raspberrypi240:~ $ sudo apt upgrade
pi240@raspberrypi240:~ $ sudo apt update
```

Once the Raspberry Pi is up to date, we'll install the Apache server.

```
pi240@raspberrypi240:~ $ sudo apt install apache2
```

You will be asked if you accept to install apache2, enter 'Y' to continue.

```
amir — pi240@raspberrypi240: ~ — ssh pi240@raspberrypi240.local — 127x24
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
pi240@raspberrypi240:~ $ sudo apt install apache
apache2                                apache2-dev                          apache2-suexec-custom                apacheds
apache2-bin                          apache2-doc                          apache2-suexec-pristine              apachetop
apache2-data                        apache2-ssl-dev                      apache2-utils
pi240@raspberrypi240:~ $ sudo apt install apache2
apache2                                apache2-data                        apache2-doc                          apache2-suexec-custom                apache2-utils
apache2-bin                          apache2-dev                          apache2-ssl-dev                      apache2-suexec-pristine
pi240@raspberrypi240:~ $ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 2062 kB of archives.
After this operation, 13.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

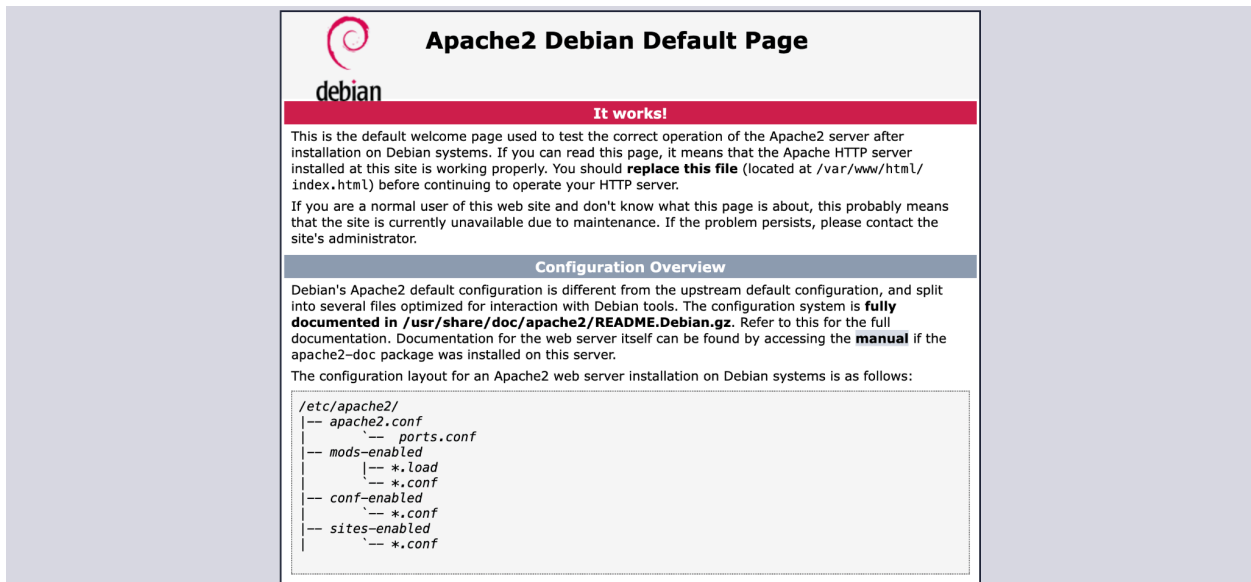
Once installed, we're going to give rights to the apache folder, which will enable you to easily administer the sites. To do this, run the following commands:

```
pi240@raspberrypi240:~ $ sudo chown -R pixyz:www-data
/var/www/html/
pi240@raspberrypi240:~ $ sudo chmod -R 770 /var/www/html/
```

Check that Apache is running

Once the installation is complete, we can test that Apache is running correctly by visiting the Raspberry's address.

To do this, try to access the Raspberry from your favorite browser (using your laptop connected to the same network as your pi) and enter the raspberry pi's hostname, for example:
<http://raspberrypi240.local>



Apache uses `/var/www/html` as the root directory for your site. This means that when you call your Raspberry on port 80 (http), Apache looks for the file in `/var/www/html`.

Right now, you can use your Raspberry to build a pure HTML, CSS and JavaScript site from scratch.

However, you'll probably soon want to enable interaction between the site and the user. For example, allowing the user to register, etc. For this, you'll need PHP.

Step 2: Installing PHP

PHP is an interpreted language. When we talk about PHP, we can mean either the language or the interpreter.

Here, when we talk about installing PHP, we mean that we're going to install the interpreter, in order to use the language.

PHP is one of the most widely used programming languages, and even the most widely used for web programming, with a market share of around 79%.

To install the interpreter, run the following command line:

```
pi240@raspberrypi240:~ $ sudo apt install php php-mbstring
```

After running the command, you'll be asked to accept or refuse to install the necessary packages, you will say 'Y'.

```
amir — pi240@raspberrypi240: ~ — ssh pi240@raspberrypi240.local — 127x24

    </div>
  </div>
  <div class="validator">
  </div>
</body>
</html>

pi240@raspberrypi240:~ $ sudo apt install php php-mbstring
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php8.2 libonig5 php-common php8.2 php8.2-cli php8.2-common php8.2-mbstring php8.2-opcache php8.2-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php8.2 libonig5 php php-common php-mbstring php8.2 php8.2-cli php8.2-common php8.2-mbstring php8.2-opcache
  php8.2-readline
0 upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 4893 kB of archives.
After this operation, 24.6 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Check that PHP is working

To find out whether PHP is working correctly, it's not very complicated, and the method is relatively similar to that used for Apache.


First, delete the "index.html" file in the "/var/www/html" directory.

```
pi240@raspberrypi240:~ $ sudo rm /var/www/html/index.html
```

Then create an "index.php" file in this directory, with this command line

```
pi240@raspberrypi240:~ $ echo "<?php phpinfo(); ?>" >
/var/www/html/index.php
```

From this point on, the operation is the same as for the Apache check. You try to access your page, and you should get a result similar to this image.

| PHP Version 8.2.18 | |
|---|--|
|  | |
| System | Linux raspberrypi240 6.6.20+rtpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.20-1+rtpt1 (2024-03-07) aarch64 |
| Build Date | Apr 11 2024 22:07:45 |
| Build System | Linux |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | disabled |
| Configuration File (php.ini) Path | /etc/php/8.2/apache2 |
| Loaded Configuration File | /etc/php/8.2/apache2/php.ini |
| Scan this dir for additional .ini files | /etc/php/8.2/apache2/conf.d |
| Additional .ini files parsed | /etc/php/8.2/apache2/conf.d/10-opcache.ini, /etc/php/8.2/apache2/conf.d/10-pdo.ini, /etc/php/8.2/apache2/conf.d/20-calendar.ini, /etc/php/8.2/apache2/conf.d/20-ctype.ini, /etc/php/8.2/apache2/conf.d/20-exif.ini, /etc/php/8.2/apache2/conf.d/20-ffi.ini, /etc/php/8.2/apache2/conf.d/20-fileinfo.ini, /etc/php/8.2/apache2/conf.d/20-ftp.ini, /etc/php/8.2/apache2/conf.d/20-gettext.ini, /etc/php/8.2/apache2/conf.d/20-iconv.ini, /etc/php/8.2/apache2/conf.d/20-mbstring.ini, /etc/php/8.2/apache2/conf.d/20-phar.ini, /etc/php/8.2/apache2/conf.d/20-posix.ini, /etc/php/8.2/apache2/conf.d/20-readline.ini, /etc/php/8.2/apache2/conf.d/20-shmop.ini, /etc/php/8.2/apache2/conf.d/20-sockets.ini, /etc/php/8.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.2/apache2/conf.d/20-sysvsem.ini, /etc/php/8.2/apache2/conf.d/20-sysvshm.ini, /etc/php/8.2/apache2/conf.d/20-tokenizer.ini |
| PHP API | 20220829 |
| PHP Extension | 20220829 |

Step3: Install the MySQL database management system

MySQL is a powerful, massively used free DBMS (approximately 56% market share of free DBMSs). Once again, MySQL is such an essential part of any development project, whatever the branch, that you absolutely must learn and master it.

To do this, we're going to install mariadb-server and php-mysql (which will act as a link between php and mysql):

```
pi240@raspberrypi240:~ $ sudo apt install mariadb-server php-mysql
```

Check that MySQL is working properly

To check MySQL's operation, this time we'll be using the command line only. To do this, we'll simply connect via the command :

```
pi240@raspberrypi240:~ $ sudo mysql --user=root
```

We're now going to delete the root user and create a new root user, as the default one can only be used by the system administrator account, and is therefore not accessible to the server's PHP scripts.

To do this, once connected to MySQL, run the following commands (replace password with the password of your choice):

```
MariaDB [(none)]> DROP USER 'root'@'localhost';

MariaDB [(none)]> CREATE USER 'root'@'localhost' IDENTIFIED BY
'<your password>';

MariaDB [(none)]>
```

To exit the MySql command line, you can use 'quit'.

So you now have a web server, linked to PHP and MySQL. That's all you need. We can also install phpMyAdmin in order to manage the databases easily.

(Next time you connect, you can use the command `mysql --user=root --password=<yourpassword>`).

Step 4: Install phpMyAdmin

However, you may want a slightly simpler interface for administering your databases than a simple MySQL console. For this, you can install PHPMyAdmin.

PHPMyAdmin is an application developed in PHP to provide a simplified interface for MySQL.

For example, it lets you quickly and easily view the contents of your database, or manipulate it without having to make MySQL queries yourself.

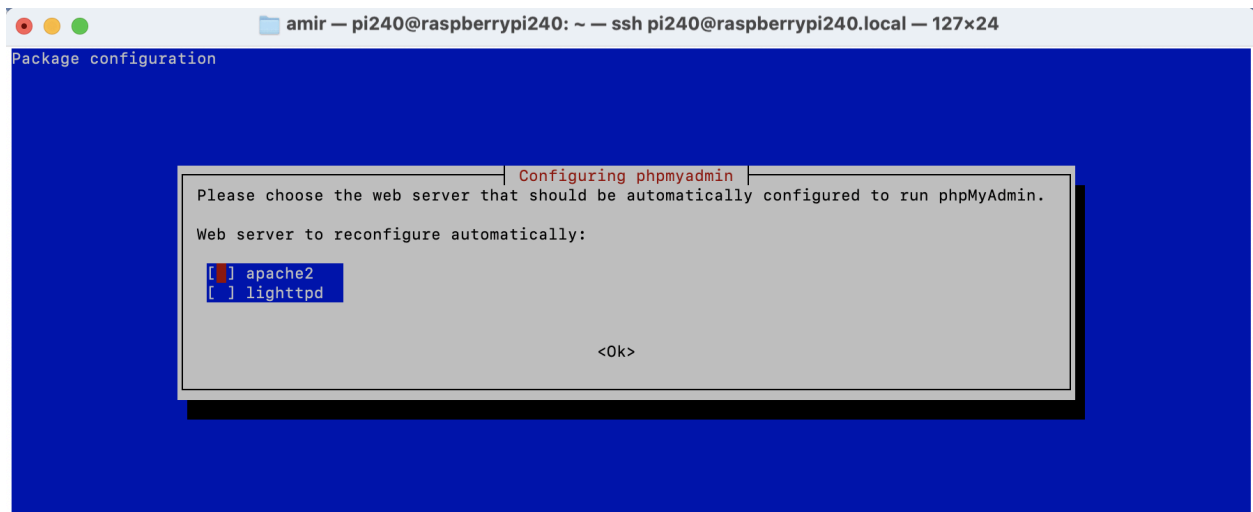
Installing PHPMyAdmin is not at all compulsory. Here, we'll be installing without any special security settings.

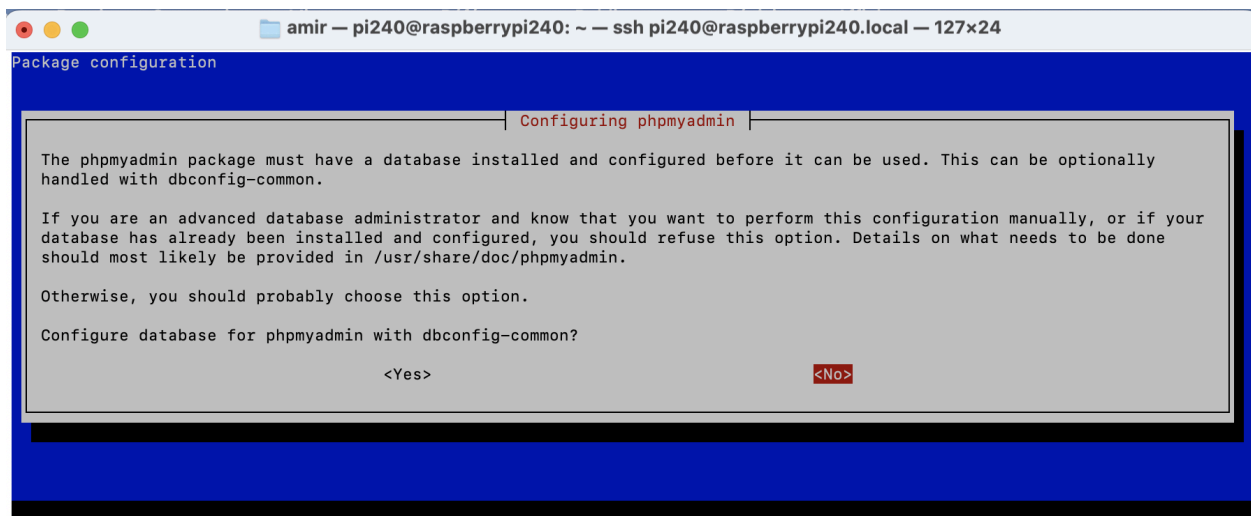
PHPMyAdmin is easily installed via the package manager, using the following command:

```
pi240@raspberrypi240:~ $ sudo apt install phpmyadmin
```

PHPMyAdmin will ask you several questions about its settings.

As we've already configured the database, choose no to the question about using dbconfig-common. Choose to use PHPMyAdmin for an Apache server. For the root password, this is the same as the one you used for MySQL.





You also need to activate the `mysqli` extension if you haven't already done so. To do this, use the commands below.

```
pi240@raspberrypi240:~ $ sudo phpenmod mysqli
pi240@raspberrypi240:~ $ sudo /etc/init.d/apache2 restart
```

Check PHPMyAdmin installation

To check that PHPMyAdmin is working properly, simply try to access it using your favorite browser and the address of your Raspberry, followed by `/phpmyadmin`.

If you get an error, it may be because PHPMyAdmin has been installed in a different folder. In this case, try the command

```
pi240@raspberrypi240:~ $ sudo ln -s /usr/share/phpmyadmin
/var/www/html/phpmyadmin
```


Control LED using a web server

The purpose of this lab is to create a web page to power on/off the LED connected to the raspberry pi.

The components

- Raspberry Pi
- Micro SD card with Raspberry PI OS
- Webserver preinstalled
- USB Type-C cable
- LED
- Resistor
- Jumper wire
- Breadboard

Step 1: Wiring the circuit

Ask your instructor to provide you with a sufficient number of cables, an LED, a resistor and a breadboard. Connect the LED to GPIO number 18 (GPIO.BCM).

The raspberry pi must be configured, connected and the web server must be configured.

Step 2: Write your code

First, we have to import the required librairies

- `os` to manage operating system dependent functionality
- `http.server` to manage HTTP servers.

After that, we have to set up different functions

- `setupGPIO()` to prepare the GPIO 18 (BCM)
- `getTemperature()` to execute the system command in order to obtain the processor's temperature.

Then, we create the http server class and methods to manage the HTTP requests.

The do_GET method contains the html code to be displayed in the browser. This source code contains a for with two submit buttons, respectively to power on and power off the LED.

The following source code is to be copied and pasted into the raspberry pi's root directory (/var/www/html/)

```
#!/usr/bin/env python3

import RPi.GPIO as GPIO
import os
from http.server import BaseHTTPRequestHandler, HTTPServer

host_name = 'raspberrypixyz.local' # update the xyz
host_port = 8000

def setupGPIO():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(18, GPIO.OUT)

def getTemperature():
    temp = os.popen("vcgencmd measure_temp").read()
    return temp

class MyServer(BaseHTTPRequestHandler):

    def do_HEAD(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def _redirect(self, path):
        self.send_response(303)
        self.send_header('Content-type', 'text/html')
        self.send_header('Location', path)
        self.end_headers()

    def do_GET(self):
        html = '''
        <html>
```



```

        <body
        style="width:960px; margin: 20px auto;">
        <h1>Welcome to my Raspberry Pi</h1>
        <p>Current GPU temperature is {}</p>
        <form action="/" method="POST">
            Turn LED :
            <input type="submit" name="submit" value="On">
            <input type="submit" name="submit" value="Off">
        </form>
        </body>
        </html>
'''
temp = getTemperature()
self.do_HEAD()
self.wfile.write(html.format(temp[5:]).encode("utf-8"))

def do_POST(self):

    content_length = int(self.headers['Content-Length'])
    post_data = self.rfile.read(content_length).decode("utf-8")
    post_data = post_data.split("=")[1]

    setupGPIO()

    if post_data == 'On':
        GPIO.output(18, GPIO.HIGH)
    else:
        GPIO.output(18, GPIO.LOW)

    print("LED is {}".format(post_data))
    self._redirect('/') # Redirect back to the root url

# # # # # Main # # # # #

if __name__ == '__main__':
    http_server = HTTPServer((host_name, host_port), MyServer)
    print("Server Starts - %s:%s" % (host_name, host_port))

    try:
        http_server.serve_forever()
    except KeyboardInterrupt:
        http_server.server_close()
```

In order to do that you must update the attribute to the html directory

```
pi240@raspberrypi240:~ $ sudo chmod -R 777 /var/www/html/
```

Step 3: run your code

In order to run your web page from your computer, you should run the following command in the pi:

```
pi240@raspberrypi240:~ $ sudo python your_file.py
```

After launching the python file, open your favorite browser from your computer and enter:

`http://raspberrypixyz.local:8000/`

The webpage will be displayed as the following figure

