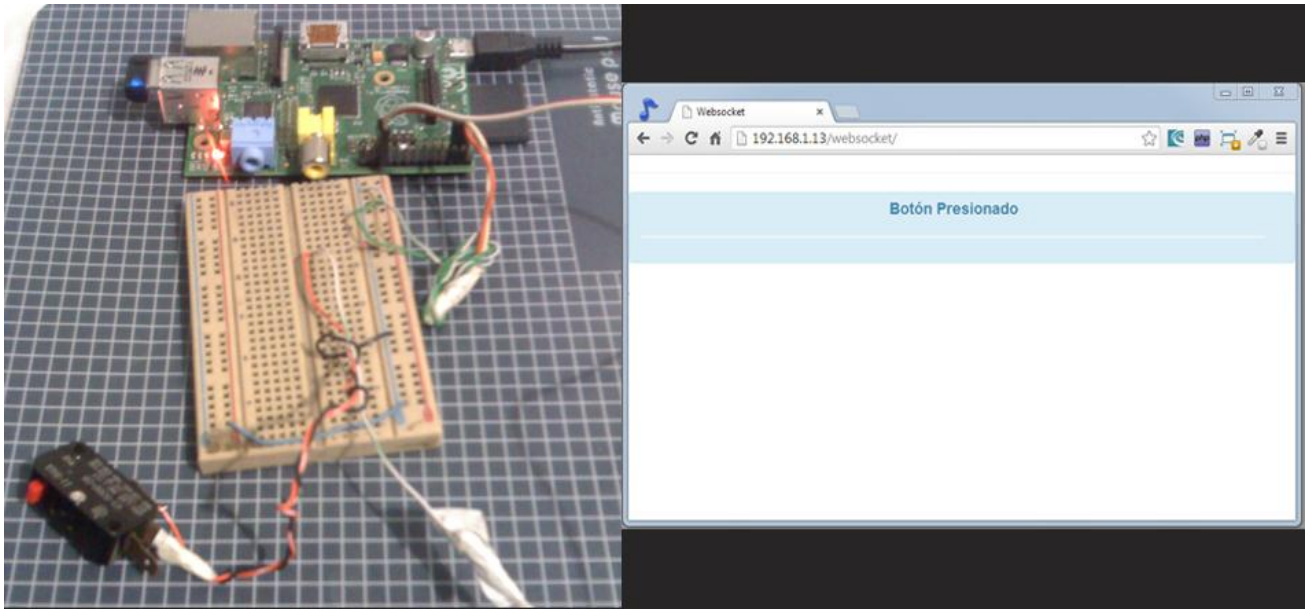
VISUALIZANDO ESTADO DE UN PULSADOR CON WEBSOCKETS - PHP Y RASPBERRY PI

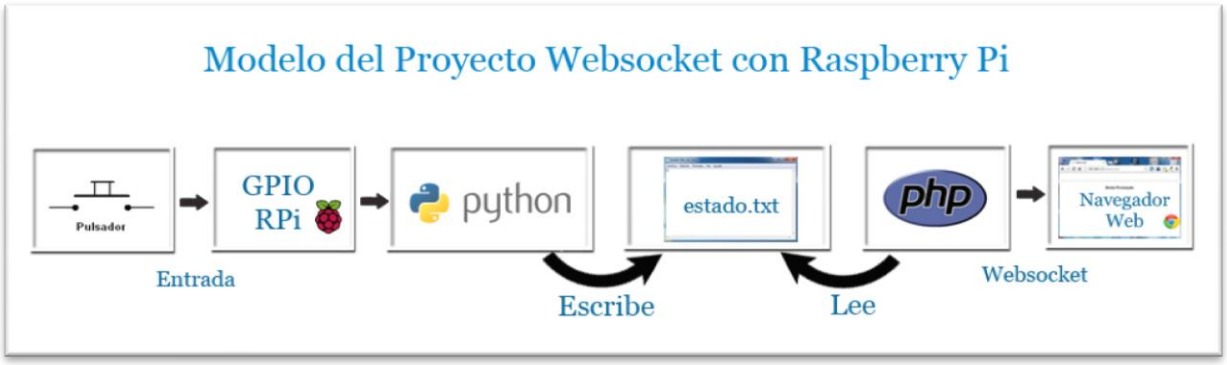


Desarrollado por:
Jefferson Rivera Patiño

_____
@riverajefer
riverajefer.blogspot.com
jeffersonrivera.com

Contenido

1. DESCRIPCIÓN DEL PROYECTO

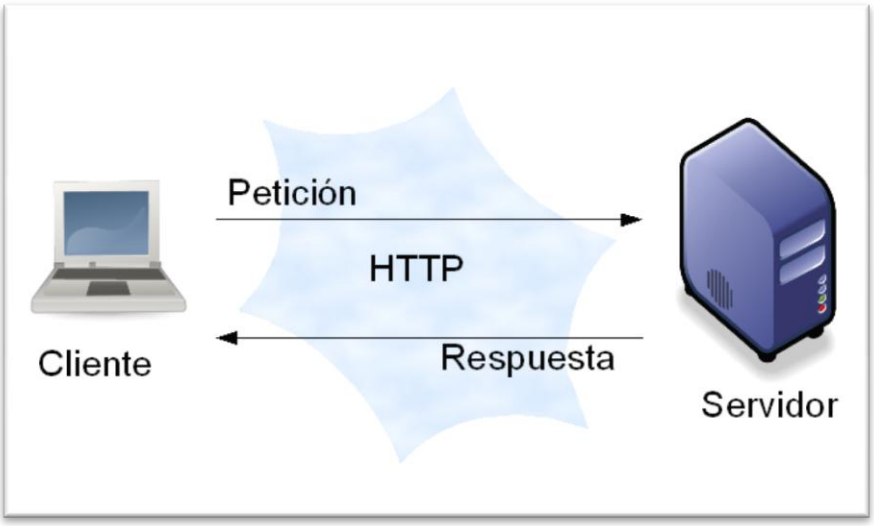


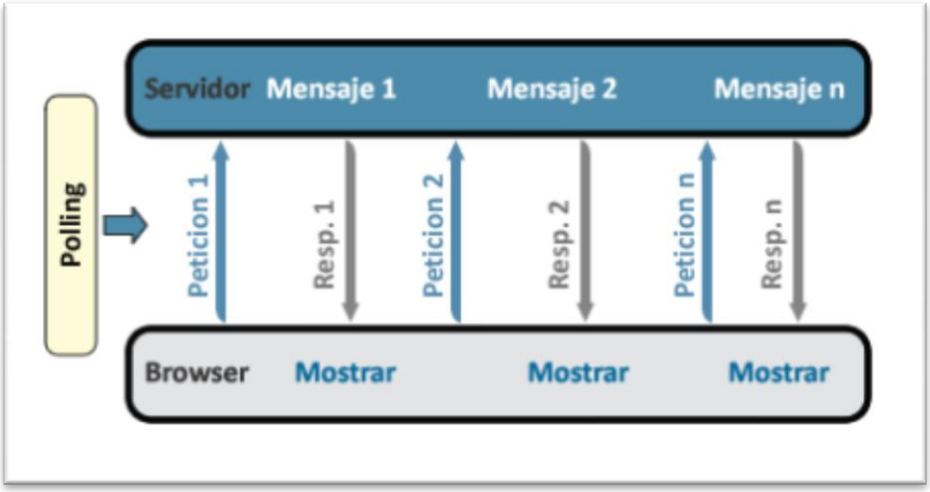
Explicando el flujo del proyecto, de la imagen anterior:

Cuando se oprime el interruptor, se envía un pulso electrónico al GPIO de la Raspberry Pi, está señal de entrada se evalúa con un script de Python y de acuerdo a su estado se escribe sobre el archivo plano *"estado.txt",* por ejemplo '1', y luego con PHP, se lee esté estado y se envía al navegador, para que esté muestre un mensaje, sin necesidad de recargar o actualizar la pagina.

2. TEORÍA CLIENTE SERVIDOR

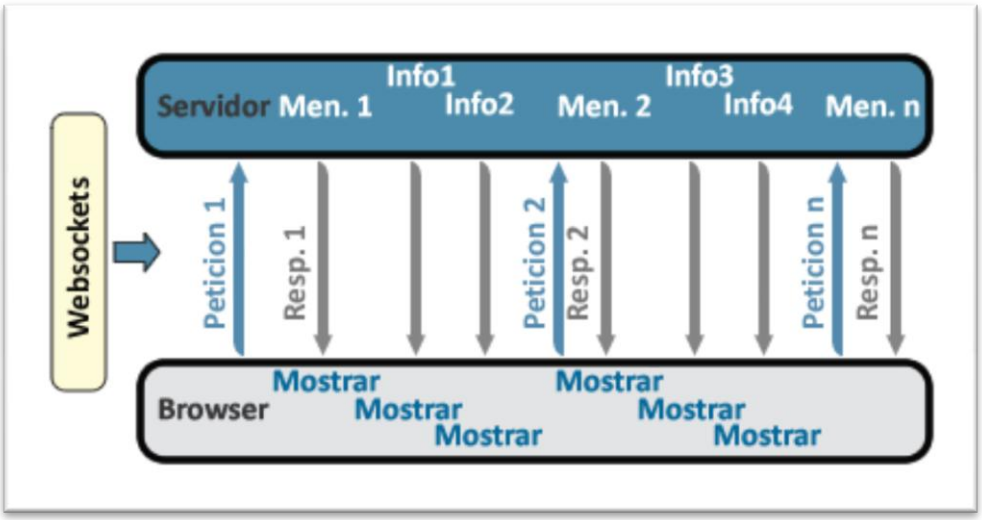
El modelo cliente servidor trabaja por peticiones - respuestas, por ejemplo el cliente o navegador le hace una solicitud o petición al servidor a través del protocolo HTTP, luego el servidor *"despierta"*, procesa la información y la manda al navegador o cliente para que esté la muestre, luego el servidor vuelve y se "*duerme"*, hasta que le envíen otra petición. Este proceso se le llama polling (preguntar y recibir)



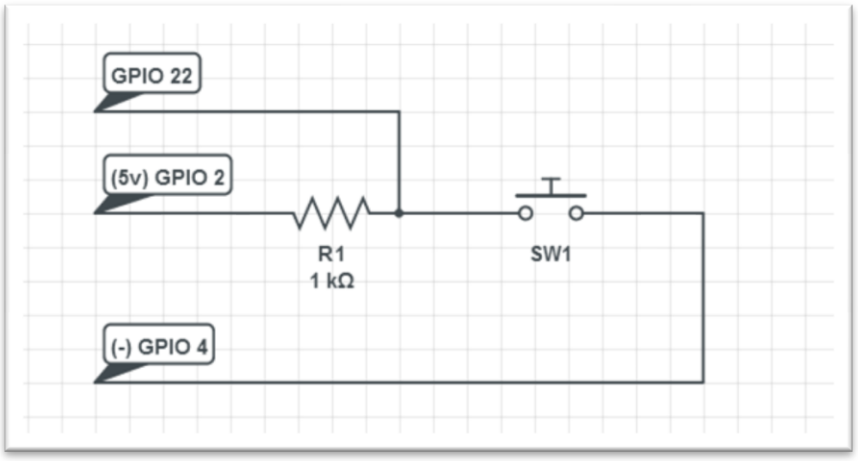


3. TEORÍA WEBSOCKET

En el modelo Websocket, el cliente o navegador le hace una primera solicitud al servidor, y se crea un canal abierto, en donde el servidor le envía mensajes o datos al navegador, en cualquier momento, sin necesidad de que el cliente haga peticiones. Está tecnología es muy importante, para hacer aplicaciones en tiempo real, por ejemplo en facebook y twitter, donde se ven las solicitudes o mensajes en el navegador sin necesitad de estar haciendo peticiones al navegador. Gracias a esta funcionalidad el modelo websocket, permite ahorrar ciclos de CPU, memoria y trafico.

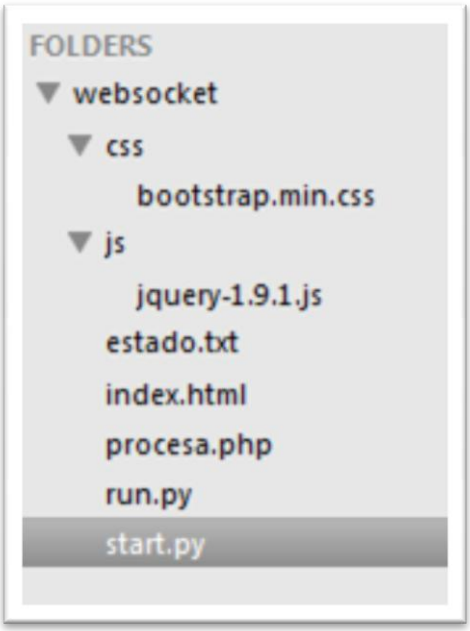


4. EXPLICACIÓN DEL HARDWARE



El diagrama es muy básico, mientras el pulsador está abierto la señal que ingresa al GPIO 22 será positiva 5v, y cuando el pulsador se cierre la señal de entrada será de 0v.

5. EXPLICACIÓN DEL SOFTWARE



La aplicación-software cuenta con las siguientes carpetas y archivos.

- **Css/**-> hojas de estilos bootstrap
- **Js/**->jquery 1.9
- estado.txt -> archivo plano donde se guarda el estado *'1'*
- index.html -> Declaración y conexión con el servidor a través de la función *Comet*
- procesa.php -> Lectura del archive estado.txt y envió de datos al cliente *(index.html)*
- run.py -> lee el pin GPIO 22 de la RPi, hace un while true, y ejecuta start.py

- start.py -> escribe en el archivo de texto plano *estado.txt,* de acuerdo a la entrada del GPIO 22.

Index.html, parte 1,

index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Websocket</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
    <script type="text/javascript" src="js/jquery-1.9.1.js"></script>
  </head>
  <body>
  <hr>
  <div align="center" id="content" class="alert alert-info">
  </div>
```

index.html. Parte 1

Index.html, parte 2 el código base fue tomado de [1]

index.html

```
  <script type="text/javascript">
  // comet implementation
  var Comet = function (data_url) {
    this.timestamp = 0;
    this.url = data_url;
    this.noerror = true;

    this.connect = function() {
      var self = this;

      $.ajax({
        type : 'get',
        url : this.url,
        dataType : 'json',
        data : {'timestamp' : self.timestamp},
        success : function(response) {
          self.timestamp = response.timestamp;
          self.handleResponse(response);
          self.noerror = true;
        },
        complete : function(response) {
          // send a new ajax request when this request is finished
          if (!self.noerror) {
            // if a connection problem occurs, try to reconnect each 1 second
            setTimeout(function(){ comet.connect(); }, 1000);
          }else {
            // persistent connection
            self.connect();
          }

          self.noerror = false;
        }
      });
    }
    this.disconnect = function() {}

    this.handleResponse = function(response) {
      if(response.msg == '1'){
       console.log('Boton Presionado:'+response.msg);
       $('#content').append('<h4>Bot&oacute;n Presionado <hr></h4>');
      }
    }

  }
  var comet = new Comet('./procesa.php');
  comet.connect();
  </script>
```

index.html. Parte 2

[1] http://webscepter.com/simple-comet-implementation-using-php-and-jquery/

Procesa.php

start.py

```php
<?php
$filename  = dirname(__FILE__).'/estado.txt';
// infinite loop until the data file is not modified
$lastmodif    = isset($_GET['timestamp']) ? $_GET['timestamp'] : 0;
$currentmodif = filemtime($filename);
while ($currentmodif <= $lastmodif) // check if the data file has been modifi
{
  usleep(10000); // sleep 10ms to unload the CPU
  clearstatcache();
  $currentmodif = filemtime($filename);
}

// return a json array
$response = array();
$response['msg']       = file_get_contents($filename);
$response['timestamp'] = $currentmodif;
echo json_encode($response);
flush(); //Vaciar el búfer de salida

/* filename: backend.php */
```

Procesa.php

Start.py, el código base fue tomado de[2]

start.py

```python
import time
import RPi.GPIO as GPIO
GPIO.setup(22,GPIO.IN)
prev_input = 0

while True:
  input = GPIO.input(22)
  if ((not prev_input) and input):
    print("Button pressed")
    archivo = open('estado.txt','w')
    archivo.write('1')
  archivo.close()
  prev_input = input
  time.sleep(0.05)
```

Run.py

[2] http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/

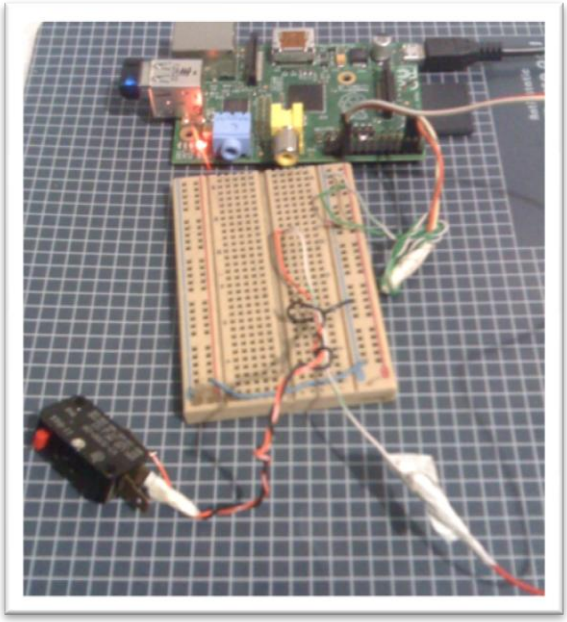
start.py

```python
import RPi.GPIO as GPIO
import time
import os

#adjust for where your switch is connected
buttonPin = 22
GPIO.setup(buttonPin,GPIO.IN)

while True:
  #assuming the script to call is long enough we can ignore bouncing
  if (GPIO.input(buttonPin)):
    #this is the script that will be called (as root)
    os.system("python /var/www/websocket/start.py")

```

Run.py

6. PRUEBA DE FUNCIONAMIENTO

Ver Video de Youtube



7. BIBLIOGRAFÍA

http://es.wikipedia.org/wiki/WebSocket

http://es.wikipedia.org/wiki/Tecnolog%C3%ADa_Push

http://es.wikipedia.org/wiki/Cliente-servidor

http://es.wikipedia.org/wiki/Comet

http://socketo.me/

http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/

http://grupo-701-marce-nestore.wikispaces.com/Cliente-Servidor

ANEXOS

Código fuente de la aplicación:

http://jeffersonrivera.com/pi/websocket.zip