UTILIZZO DI UNA RASPBERRY COME WEBSERVER

Nel più comune dei casi di utilizzo di una Raspberry come WebServer per la gestione remota di dati e variabili, occorre conoscere il meccanismo di passaggio di informazioni tra il “client”, normalmente un PC o un qualsiasi dispositivo che tramite Browser accede alla Raspberry ed il server su di essa residente.
In primo luogo sulla scheda Raspberry dovrà essere installata la piattaforma L.A.M.P. Acronimo di **L**inux (il sistema operativo) **A**pache (il webserver) **M**ysql (il database) e **P**hp (il liguaggio di scripting interpretato).
Per eseguire questa operazione accedere alla shell dei comandi della Rasberry e digitare I seguenti comandi:

sudo bash	*- accedo come superuser*
apt-get install apache2	*- installazione apache2*
apt-get install mysql-server	*- installazione mysql*
apt-get install phpmyadmin	*- installazione applicazione web per amministrare database mysql*

*(l’utente per phpmyadmin sarà **phpmyadmin** e la password quella scelta durante l’installazione)*

Ovviamente la scheda Raspberry dovrà essere connessa ad internet per poter eseguire l’installazione dei pacchetti richiesti.

Da questo momento in poi digitando nel browser di un qualsiasi pc l’indirizzo IP della scheda Raspberry (supponiamo che l’IP sia 192.168.43.105) si aprira la pagina index.html, presente nel percorso **/var/www/html/index.html**.

Il file index.html iniziale contiene informazioni sul werbserver apache, e pertanto è consigliabile salvarlo con altro nome o non modificarlo.
Nella cartella html, possiamo anche mettere altri file html o php da lanciare sempre dalla barra degli indirizzi del browser specificando dopo l’indirizzo il nome del file, ad esempio **192.68.43.105/prova.html.**

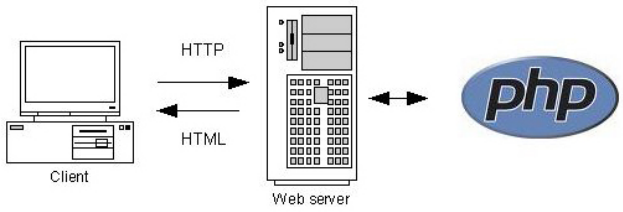
Digitando invece l’indirizzo IP seguito da phpmyadmin, **192.168.43.105/phpmyadmin** , si aprirà l’applicazione web per la gestione del database mysql.

Uno dei requisiti fondamentali per realizzare un webserver è la conoscenza del linguaggio PHP, per fare questo si consiglia di leggere la seguente guida online da dove verranno estrapolate delle parti utilizzate in questo documento.

http://www.html.it/guide/guida-php-di-base/

Metodo GET e POST *(fonte html.it)*

Il client (normalmente un browser) effettua una *richiesta HTTP* al server, il server elabora la richiesta e restituisce una risposta. La risposta può essere una pagina HTML, un'immagine o qualsiasi altro formato.



La specifica HTTP definisce **9 tipi di metodi** alcuni dei quali non sono però usati o supportati da PHP; i più diffusi restano sicuramente **GET** e **POST**. GET è il metodo con cui vengono richieste la maggior parte delle informazioni ad un Web server, tali richieste vengono veicolate tramite query string, cioè la parte di un URL che contiene dei parametri da passare in input ad un'applicazione, ad esempio:

www.miosito.com/pagina-richiesta?id=123&page=3

Il metodo POST, invece, consente di inviare dati ad un server senza mostrarli in query string, è ad esempio il caso dei **form**.

METODO GET

Iniziamo con il metodo GET. Sicuramente è il più semplice e il più immediato. È consigliato soprattutto in quelle richieste in cui è utile salvare nell'URL i parametri richiesti, ad esempio per poter essere cachati.
Un classico caso d'uso è una query di ricerca. Vediamo infatti un esempio di come poter accedere ai parametri in GET di una richiesta HTTP proveniente da un form di ricerca. Avremo bisogno di due file: `form.html` e `search.php`.

Analizziamo innanzitutto il codice del file `form.html` che conterrà il form:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <form action="search.php">
      <input type="text" name="author" placeholder="Inserisci autore"
/>
      <input type="submit" value="Cerca" />
    </form>
</body>
</html>
```

La pagina contiene al suo interno due campi:

1. un campo di input di tipo testo chiamato author in cui inserire il nome di un autore da ricercare;
2. un campo submit per effettuare la richiesta.

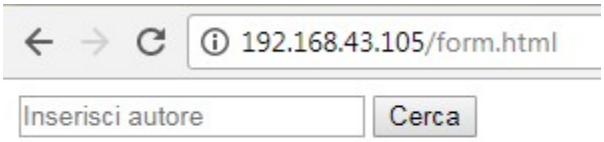
Se clicchiamo sul tasto di submit vediamo che viene effettuata una richiesta al file search.php e che, contestualmente, viene aggiunto un parametro con il valore di ricerca che abbiamo inserito noi. Se usiamo “Pippo” come chiave di ricerca, il browser viene indirizzato alla pagina:

http://localhost/search.php?author=Pippo

Andiamo quindi a creare search.php e vediamo come possiamo recuperare il nome dell'autore ricercato e come fornire un esempio di risposta al client.

```
?php

$author = $_GET['author'];
$author = filter_var($author, FILTER_SANITIZE_STRING);

$authors = [
    'Stephen King' => 'Stephen Edwin King (Portland, 21 settembre 1947) è uno scrittore e
sceneggiatore statunitense',
    'Arthur Conan Doyle' => 'Sir Arthur Ignatius Conan Doyle (Edimburgo, 22 maggio 1859 –
Crowborough, 7 luglio 1930) è stato uno scrittore scozzese',
    'Agatha Christie' => 'Dame Agatha Mary Clarissa Miller, Lady Mallowan, nota come Agatha Christie
(Torquay, 15 settembre 1890[1] – Wallingford, 12 gennaio 1976), è stata una scrittrice britannica.'
];

if (!in_array($author, array_keys($authors))) {
    $error = 'Autore non trovato';
}

$result = $authors[$author];

?>
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <h1>Risultati di ricerca per: <?php echo $author ?></h1>

    <?php if ($error): ?>
        <p style="color: red"><?php echo $error ?></p>
    <?php else: ?>
        <p><?php echo $result ?></p>
    <?php endif ?>
</body>
</html>
```

Possiamo ora creare I due file form.html e search.php all'interno della directory /var/www/html della raspberry e digitare da riga di comando del client remoto 192.168.43.105/form.html.



Come si può vedere nella url in alto troviamo I parametri passati e cioè il nome dell'autore.

Analizziamo il codice nel file search.php. La prima riga di nostro interesse è quella in cui accediamo all'array speciale `$_GET`. Questo array viene automaticamente popolato con i parametri che vengono passati in GET. Nel nostro caso il campo di input in cui abbiamo inserito l'autore si chiama `author`, di conseguenza la chiave dell'array in cui troviamo il valore ricercato ha il medesimo nome.

La riga successiva utilizza la funzione `filter_var()` vista nella lezione precedente per sanitizzare il dato ricevuto e cioè rimuovere i il markup HTML restituendo una stringa depurata dai tags.

Ricordiamoci che è fondamentale **validare tutti i dati che vengono immessi dagli utenti** per ridurre al minimo i problemi di sicurezza.

Il resto del codice è puramente di esempio ed infatti è mancante di un reale sistema di ricerca.

METODO POST

Il metodo POST si differenza da GET in quanto i parametri della richiesta non vengono passati in query string e quindi non possono essere tracciati nemmeno negli access log dei web server. Caso d'uso comune di una richiesta in POST è un form che invia dati personali, come in una registrazione. Vediamo quindi come accedere ai parametri POST con un esempio di registrazione tramite username e password. Anche in questo caso abbiamo bisogno di due file: **`form.html`** e **`register.php`**. Iniziamo con il file contenente il form:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <form action="register.php" method="post">
        <input type="text" name="username" placeholder="Inserisci lo username" /><br>
        <input type="password" name="password" placeholder="Inserisci la password"
/><br>
         <input type="submit" value="Registrati" />
    </form>
</body>
</html>
```

Il codice è simile all'esempio precedente. Le differenze sostanziali sono la presenza di due campi username e password e, soprattutto, l'aggiunta dell'attributo method nel tag form. Quando abbiamo bisogno di effettuare una richiesta POST è necessario specificare il metodo nel form. A questo punto possiamo proporre il codice di register.php:

```
<?php
 $username = $_POST['username'];
$password = $_POST['password'];
 $username = filter_var($username,
FILTER_SANITIZE_STRING);
$password = filter_var($password,
FILTER_SANITIZE_STRING);
if (!$username || !$password) {
    $error = 'Username e password sono obbligatori';
}
 ?>
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <h1>Risultati Registrazione</h1>
     <?php if ($error): ?>
        <p style="color: red"><?php echo $error ?></p>
    <?php else: ?>
        <p>Benvenuto <?php echo $username ?></p>
    <?php endif ?>
</body>
</html>
```

Anche in questo caso creando I due file nella directory **/var/www/html** del raspberry potremo accedere al form digitando 192.168.43.105/form e verificare il funzionamento.

WEB SERVER + PYTHON

Come visto nei vari tutorial, la raspberry è una scheda che oltre ad essere dotata delle normali periferiche di un PC, ha una serie di terminali digitali (GPIO, General Purpose Input ed Outout) per interfacciarsi con altri dispositivi. Molte delle applicazioni che si possono realizzare richiedono pertanto l'esecuzione di programmi scritti in C o in Python da file PHP.

Un esempio potrebbe essere il classico diodo LED acceso o spento tramite interfaccia web.

Nella maniera più semplice si potrebbero creare due file scritti in python per accendere o spegnere un led come di seguito.

accendi.py

```
#importiamo la libreria a GPIO
import RPi.GPIO as GPIO

#Stabiliamo il tipo di numerazione dei pin
#in questo caso il sistema BCM
GPIO.setmode(GPIO. BCM)

#Configurare il pin GPIO 4 come uscita
GPIO.setup(4, GPIO. OUT)
#Si accende il led
GPIO.output(4, GPIO. HIGH)
#si libera il GPIO
GPIO.cleanup()
```

spegni.py

```
#importiamo la libreria a GPIO
import RPi.GPIO as GPIO

#Stabiliamo il tipo di numerazione dei pin
#in questo caso il sistema BCM
GPIO.setmode(GPIO. BCM)

#Configurare il pin GPIO 4 come uscita
GPIO.setup(4, GPIO. OUT)
#Disattivare il led
GPIO.output(4, GPIO. LOW)
#si libera il GPIO
GPIO.cleanup()
```

Per lanciare i due programmi scritti in python dal file php, si utilizza il comand **shell_exec.**

Il comando consente di avviare un qualsiasi comando come se stessimo scrivendo sulla shell dei comandi, per poter funzionare il web server deve avere i giusti permessi.

L'utente di default del webserver in Linux si chiama **www-data**, esso dovrà appartenere agli utenti che possono agire con privilegi di amministrazione.

Il file sudoers presente nella cartella **etc**, contiene i nomi degli utenti che possono diventare temporaneamente amministratori (root) pertanto digitando il comando **sudo visudo** si apre il file **sudoers** e si deve aggiungere la seguente riga: **www-data ALL=NOPASSWD: ALL**

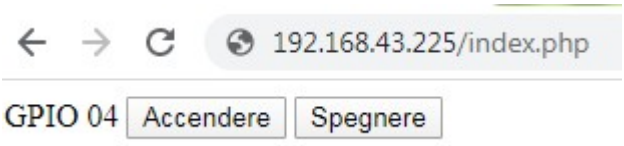
Dopo aver salvato il comando shell_exec potrà sicuramente essere utilizzato in un qualsiasi file php.

Ad esempio combinando i due programmi con il seguente file php, avremo la possibilità di accendere e spegnere il led collegato alla GPIO4, mediante una pagina web accessibile digitando sul browser: ***indirizzo_ip/index.php***

File **index.php**, da inserire nella directory **/var/www/html**

```
<html>
 <head>
        <!--index.php-->
 </head>
 <body>
  <!--GPIO4-->
  <form action="" method="post">
   GPIO 04 <input type="submit" name="accendi4" value="Accendere">
   <input type="submit" name="spegni4" value="Spegnere">
 <br></br>
 </body>
</html>
<?php

// Funzioni PHP del pin GPIO 4
  if ($_POST[accendi4]) {
   $a- exec("sudo python /var/www/accendi.py");
   echo $a;
  }
  if ($_POST[spegni4]) {
   $a- exec("sudo python /var/www/spegni.py");
   echo $a;
  }
// Fine funzioni del pin GPIO 4
?>
```

Digitando sul browser l'indirizzo ed il nome del file come detto prima avremo il seguente risultato:



ESEMPIO APPLICATIVO

Tramite il php possiamo anche passare dei valori al programma scritto in python, come nel seguente esempio. In questo caso ci sono 3 file, il file **html**, presente sul client ed i file **php** e **python**, presenti sul server nella cartella **\var\www\html.**

File prova.html

Il file contiene la pagina in html con un form per scrivere un testo ed inviarlo al server con il metodo GET con il pulsante INVIA, il file come detto prima, è residente sul PC client connesso alla stessa rete del server.

File index.php

Il file contiene il programma in php che riceve il testo inviato con il metodo GET, e lancia con il comando sheel_exec la stringa per avviare il programma in python, come se stessimo digitando nella CLI. Nella stringa viene passato il testo ricevuto tramite la variabile “valore”.

File scritta.py

Il file contiene il programma che andrà in esecuzione ed a cui verrà passato il testo inizialmente digitato nel form in html.

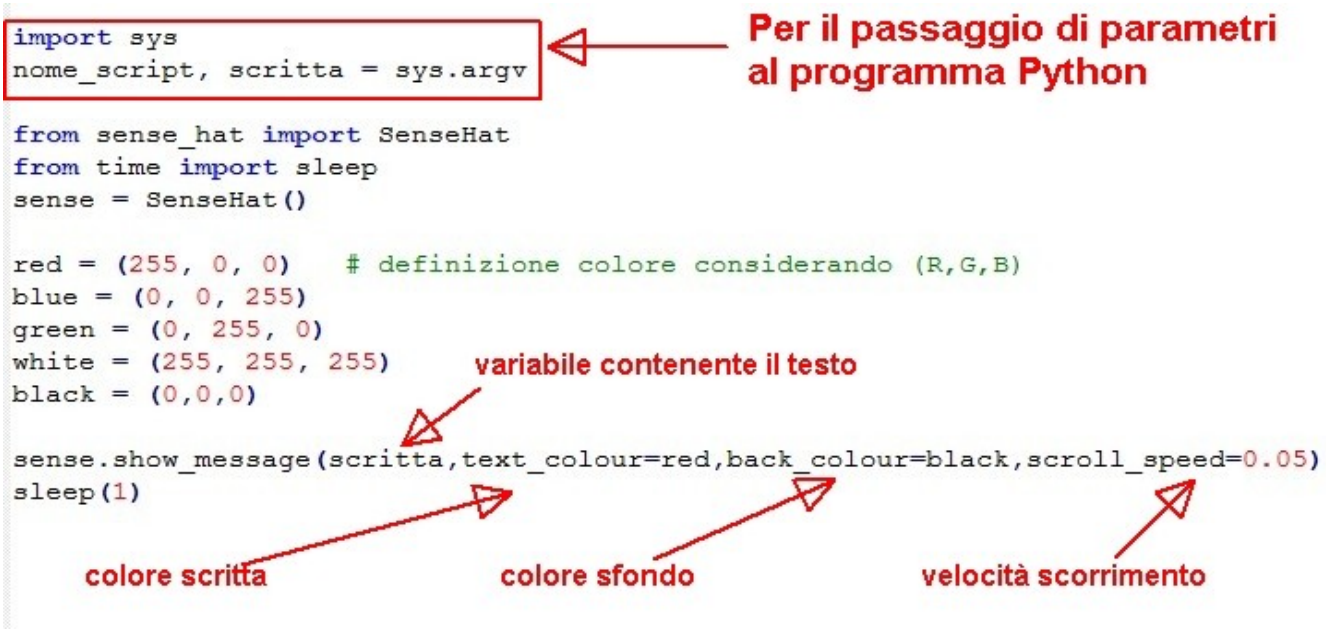
Il passaggio di valori ad un programma in python può avvenire importando il modulo sys, tramite import sys.

Questo modulo contiene una serie di funzioni e parametri utili per interagire con il sistema operativo in uso.

Tra le possibilità offerte c'è quella di importare dei parametri da riga di comando, come nel nostro caso.

Infatti il comando eseguito nel file php, contiene il parametro contenuto nella variabile $valore, come visto sopra.

Nel nostro caso il programma visualizza la scritta inviata sul display della scheda sense_hat, e per questo si dovrà importare ed utilizzare il relativo modulo con le sue funzioni, come descritto nella dispensa sulla stessa scheda.



Possiamo provare l'esempio sopra ricopiando i 3 file, ricordandoci di inserire il corretto indirizzo IP della scheda Raspberry nel file html.

Questo esempio può dare l'idea delle potenzialità offerte dalla scheda che se utilizzata come server, può consentire lo sviluppo di svariate applicazioni per il controllo in remoto. Ricordiamo inoltre la possibilità di organizzare e gestire anche dei database più o meno complessi, parte questa che richiedere la conoscenza di mysql e phpmyadmin, si consiglia per questo di utilizzare l'ottimo supporto offerto dal sito: https://www.w3schools.com