Embedded HTTP Server

An embedded web server resides on a hardware product, such as a printer, rather than software loaded on a network server. Can be Arduino or a BC like Raspberry Pi.

An embedded HTTP server is a component of a software system that implements the HTTP protocol. Examples of usage within an application might be:

- To provide a thin client interface for a traditional application.
- To provide indexing, reporting, and debugging tools during the development stage.
- To implement a protocol for the distribution and acquisition of information to be displayed in the regular interface — possibly a web service, and possibly using XML as the data format.
- To develop a web application

There are a few advantages to using HTTP to perform the above:

- HTTP is a well-studied cross-platform protocol and there are mature implementations freely available.
- HTTP is seldom blocked by firewalls and intranet routers.
- HTTP clients (e.g. web browsers) are readily available with all modern computers.

There is a growing tendency of using embedded HTTP servers in applications that parallels the rising trends of home-networking and ubiquitous computing.

Protocol for Embedded and SBC Device Communication

REST (Representational State Transfer) was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. REST is an architectural style for designing distributed systems. It is not a standard but a set of constraints, such as being stateless, having a client/server relationship, and a uniform interface. REST is not strictly related to HTTP, but it is most commonly associated with it.

Principles of REST

- Resources expose easily understood directory structure URIs.
- Representations transfer JSON or XML to represent data objects and attributes.
- Messages use HTTP methods explicitly (for example, GET, POST, PUT, and DELETE).
- Stateless interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.

HTTP methods

Use HTTP methods to map CRUD (create, retrieve, update, delete) operations to HTTP requests.

WebSocket was initially created in order to provide browser a way to establish a full-duplex communication channel. It's the TCP socket of Web, as ordinary browser can not open a native TCP socket. WebSocket can be used for connectivity of embedded devices as well, but is it optimal for that purpose?

MQTT. The acronym stands for Message Queue Telemetry Transport, a protocol initially invented by IBM. The aim of the protocol can be expressed as follows

What makes MQTT more suitable than WebSocket or HTTP REST API in context of Embedded Systems?

- MQTT provides publish/subscribe mechanism already in protocol level
- MQTT provides Quality of Service policy
- MQTT introduces minimal overhead in communication
- MQTT is designed for narrowband communication channel and constrained devices

ESP8266 WiFi client and Posting HTML to your router

```
String request = client.readStringUntil('\r');
 Serial.println(request);
 client.flush();
 // Match the request
 int value = LOW;
          if (request.indexOf("/LED=ON") != -1)
          {
                    digitalWrite(ledPin, HIGH);
                    value = HIGH;
          }
          if (request.indexOf("/LED=OFF") != -1)
          {
                    digitalWrite(ledPin, LOW);
                    value = LOW;
          }
```

```
client.println("<a  href=\"/LED=ON\"\"><button>Turn On </button></a>");

client.println("<a  href=\"/LED=OFF\"\"><button>Turn Off </button></a><br />");
```

Input to: http://www.embedded-iot.net/dht11/dataDisplayer.html

URL tests from Url bar at: http://www.embedded-iot.net/dht11/dataDisplayer.html

Useful protocol testing Tools: cURL

cURL (/kɜːl/ or /kəːl/[4]) is a computer software project providing a library and command-line tool for transferring data using various protocols. The cURL project produces two products, libcurl and cURL. It was first released in 1997. The name originally stood for "see URL".

Sample commands to retrieve.

To retrieve the embedded-iot.net homepage, type:

curl www.embedded-iot.net

capture the code to output file:

curl -o exawww.example.commple.html

for our Post example

curl -v -d "Humidity=66&Temperature_Cel=77&Temperature_Fehr=88&HeatIndex_Cel=99&HeatIndex_Fehr=00" http://www.embedded-iot.net/dht11/dataCollector.php

---

Postman Chrome Plug-in a graphic URL API tester: Demo

Chrome extension **Postman** REST Client, an HTTP client for testing web services. Build, test, and document your **APIs** faster. ... The idea for **Postman** arose while the founders were working together, and were frustrated with the existing tools for testing **APIs**.

can post data to : http://embedded-iot.net/dht11/dataCollector.php?Humidity=58

Link to my program:

https://www.hackster.io/detox/send-esp8266-data-to-your-webpage-no-at-commands-7ebfec

```
    String data =
    "Humidity="
    + (String) Humidity
    + "&Temperature_Cel=" +(String)
    Temperature_Cel
    + "&Temperature_Fehr=" +(String)
    Temperature_Fehr
    + "&HeatIndex_Cel=" +(String)
    HeatIndex_Cel
    + "&HeatIndex_Fehr=" +(String)
    HeatIndex_Fehr;

    //client.println("POST
    /dht11/dataCollector.php HTTP/1.1");
    client.println("POST
    /YOUR_SUBDOMAIN/YOUR_PHP_PAGE.php
    HTTP/1.1");
    //client.print("Host: embedded-
    iot.net\n");
    client.print("Host:
    YOUR_TOPLEVEL_DOMAIN_HERE\n");
    client.println("User-Agent:
    ESP8266/1.0");
    client.println("Connection:
    close");
    client.println("Content-Type: application/x-
    www-form-urlencoded");
    client.print("Content-
    Length: ");
    client.print(data.length());
    client.print("\n\n");
    client.print(data);
client.stop();
```

```
<?php

date_default_timezone_set("America/Los_Angeles");
$TimeStamp = "The current time is " . date("h:i:sa");
file_put_contents('dataDisplayer.html', $TimeStamp, FILE_APPEND);


  if( $_REQUEST["Humidity"] ||  $_REQUEST["Temperature_Cel"] ||
$_REQUEST["Temperature_Fehr"]
      ||  $_REQUEST["HeatIndex_Cel"] ||  $_REQUEST["HeatIndex_Fehr"] )
  {
     echo " The Humidity is: ". $_REQUEST['Humidity']. "%<br />";
     echo " The Temperature is: ". $_REQUEST['Temperature_Cel']. " Celcius<br />";
     echo " The Temperature is: ". $_REQUEST['Temperature_Fehr']. " Fehrenheit<br />";
     echo " The Heat Index is: ". $_REQUEST['HeatIndex_Cel']. " Heat Index Celcius<br />";
     echo " The Heat Index is: ". $_REQUEST['HeatIndex_Fehr']. " Heat Index Fehrenheit<br />";
  }


$var1 = $_REQUEST['Humidity'];
$var2 = $_REQUEST['Temperature_Cel'];
$var3 = $_REQUEST['Temperature_Fehr'];
$var4 = $_REQUEST['HeatIndex_Cel'];
$var5 = $_REQUEST['HeatIndex_Fehr'];

$WriteMyRequest=
"<p> The Humidity is : "       . $var1 .  "% </p>".
"<p>And the Temperature is : " . $var2 .  " Celcius </p>".
"<p>And the Temperature is : " . $var3 .  " Fehreinheit</p>".
"<p>And The Heat Index is : "  . $var4 .  " Heat Index Celcius </p>".
"<p>And The Heat Index is : "  . $var5 .  " Heat Index Fehrenheit </p><br/>";
file_put_contents('dataDisplayer.html', $WriteMyRequest, FILE_APPEND);

?>
```

Input to: http://www.embedded-iot.net/dht11/dataDisplayer.html

URL tests from Url bar at: http://www.embedded-iot.net/dht11/dataCollector.php

Hosted Web code from Authors:

http://devices.webofthings.io/pi/actuators/display/

embedded WebCode

http://www.embedded-iot.net/dht11/WoT_IoT.php
Go to: http://www.embedded-iot.net/dht11/WoT_SimpleSend.html

Now

receive picture at:

http://devices.webofthings.io/camera/sensors/picture/

or embed:

http://www.embedded-iot.net/dht11/WoT_SensorValue.html

Embedded graph with Google API:

www.embedded-iot.net/dht11/WoT_Graph1.html