CAPÍTULO

HTTP

JPSantos
2023

5. HTTP – HiperText Transfer Protocol

O protocolo *HTTP* é a pedra angular da WEB. Este protocolo, juntamente com a linguagem *HTML*, foi proposto por Tim Berners-Lee enquanto investigava no *CERN*. De facto Tim Berners propôs um sistema de gestão de informação baseado em Hipertexto. Na altura o termo *World Wide Web -WWW* ainda não existia, mas o sistema proposto por Tim Berners já definia:

1. Uma linguagem para formatar documentos de texto (*HTML – HyperText Markup Language*). Esta linguagem e a sua sintaxe são apresentadas no capítulo seguinte.
2. Um protocolo para transferir esses documentos entre equipamentos (*HTTP – Hyper Text Transfer Prototocol*).
3. Um esquema de endereços para identificar e aceder a esses recursos na rede (*URL- Uniform resource locator*).

Convém relembrar que quando Tim Berner propôs o protocolo HTTP já existia a Internet, já existiam servidores e clientes *FTP – File Transfer Protocol* e por isso já era possível transferir documentos entre equipamentos, mas não era fácil localizá-los, nem existia uma forma fácil de um desses documentos fazer uma referência a outro documento (*Hiperligação*). Além disso a forma como cada documento estava formatado e guardado em disco variava de processador de texto para processador de texto.

5.1. Conceitos sobre HTTP

O protocolo *HTTP* define um conjunto de interações entre as aplicações clientes (*Browsers WEB*) e as aplicações servidoras (*Servidores WEB*) e define a sintaxe de um conjunto de mensagens que todos os clientes e servidores *HTTP* reconhecem e sabem processar.

5.1.1. Localização dos recursos na Internet

Tim Berners propôs uma forma de identificar e localizar um documento, um recurso, na Internet e chamou-lhe *URL – Uniform Resource Locator*, mais tarde passou a chamar-se *URI - Uniform Resource Identifier*.

De uma forma simples e incompleta, podemos dizer que o URL não é mais que um texto, formado pelo nome do computador de destino e pela localização do documento no disco duro do equipamento de destino.

O URL tem a estrutura seguinte:

Protocolo://computador:[tcpport number]/localização do documento dentro do computador de destino
[? Paramentos do pedido] [# ancora]

Alguns exemplos de URLs

http://www.ua.pt/index.html
ftp://...../.....

5.1.2. Interações HTTP

A interação entre programas *HTTP* é do tipo "*cliente – servidor*". Isto significa que um programa (o *browser web*) toma a iniciativa de enviar uma mensagem *HTTP* com um pedido, ao passo que o outro programa "limita-se" a processar e a responder a esse pedido com uma mensagem *HTTP* de resposta (*servidor web*).

1. Pedido de ligação

No entanto, antes de ser possível fazer pedidos HTTP, ou seja, antes de enviar mensagens HTTP, a aplicação cliente tem de estabelecer uma ligação TCP com a aplicação remota.
Na figura seguinte, através de um browser Web, o utilizador pretende aceder a um documento remoto com o seguinte URL:
http://localhost/index.html

Através do URL, o browser sabe que deve usar o protocolo http e tem de estabelecer uma ligação TCP com o computador "localhost"

2. Envio de uma mensagem HTTP para o servidor e a mensagem de resposta

Depois da ligação TCP estar estabelecida, o browser pede ao servidor remoto o documento índex.html, enviando para isso uma mensagem HTTP do tipo GET:

GET /index.html HTTP/1.1
.....
....

Quando o *servidor WEB* consegue processar a *mensagem HTTP* descrita na figura, gera uma mensagem de resposta começada por "*HTTP/1.1 200 OK*", acede ao documento pedido, copia o seu conteúdo e acrescenta esse conteúdo à mensagem de resposta.

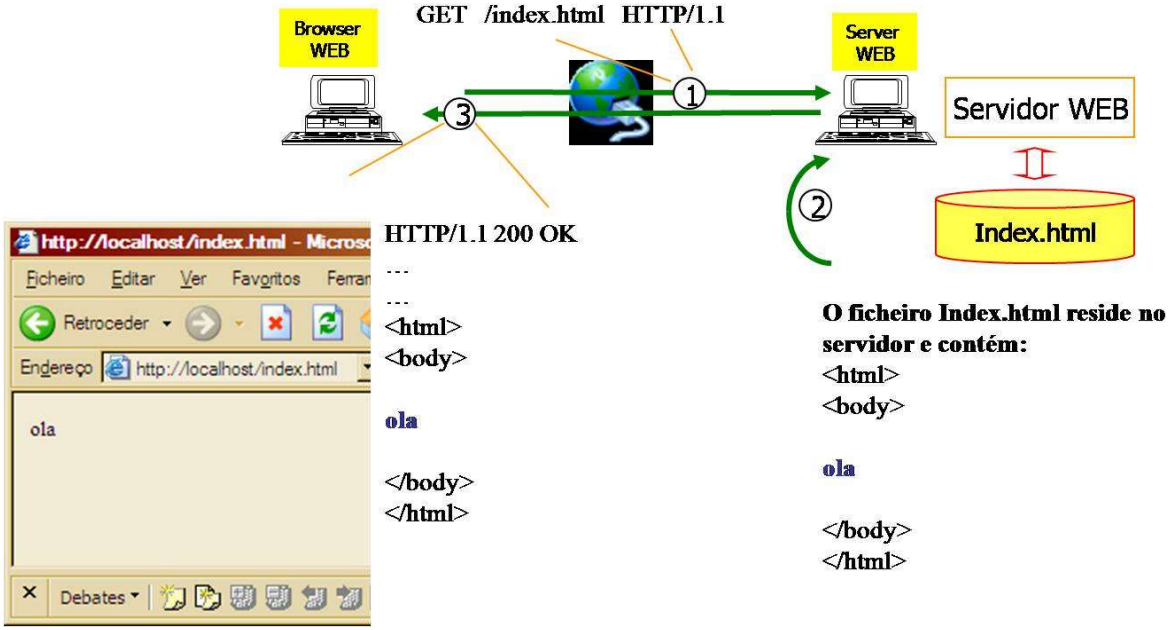


Figura 10.13: Troca de mensagens HTTP

3. Fim de ligação

Depois da troca de mensagens http tanto o browser como o servidor Web podem terminar a ligação TCP.

5.1.3. Mensagens HTTP

Depois da ligação TCP ter sido previamente estabelecida o browser Web pode enviar um de três tipos de pedidos para o servidor: GET, HEAD, e POST.

GET

As mensagens do tipo GET levam o servidor a responder com informações sobre o documento pedido, incluindo o seu conteúdo.

Quando no Browser o utilizador insere o URL:

http://192.18.43.197/index.html?a=1&b=2

O Servidor recebe a mensagem:

GET /index.html?a=1&b=2 HTTP/1.1
Host: 192.168.43.197
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 9; Redmi Note 8T) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Mobile Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,/*;q=0.8,application/signed-exchange;v=b3;q=0.9*
Accept-Encoding: gzip, deflate
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

HEAD

O browser pode pedir ao servidor para lhe enviar apenas informação sobre as características de um documento, por exemplo, pedir-lhe que confirme se esse documento existe no servidor, mas sem pedir o seu conteúdo.

POST

O browser pode enviar um pedido de atualização de um documento no servidor. Neste caso a mensagem enviada pelo browser é do tipo POST e contém também os novos dados que o servidor irá utilizar para atualizar o documento ou executar algumas acções.

POST /path/script.cgi HTTP/1.0
From: frog@jmarshall.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

home=Cosby&favorite+flavor=flies

Respostas do servidor

Consoante o tipo de pedido efetuado pelo browser e dependendo da forma como o servidor foi capaz de reconhecer e executar esse pedido, o servidor Web responde com um de vários códigos possíveis

HTTP/1.1 código de resposta
…..
…..

Código de resposta =
"200" ; OK
"201" ; Created
"202" ; Accepted
"204" ; No Content
"301" ; Moved Permanently
"302" ; Moved Temporarily
"304" ; Not Modified
"400" ; Bad Request
"401" ; Unauthorized
"403" ; Forbidden
"404" ; Not Found
"500" ; Internal Server Error
"501" ; Not Implemented
"502" ; Bad Gateway
"503" ; Service Unavailable

HTTP/1.1 200 OK
Date: Mon, 20 Nov 2017 00:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 20 Jul 2017 19:15:56 GMT
Content-Length: 38
Content-Type: text/html
Connection: Closed
Cache-Control: no-cache

https://pt.wikipedia.org/wiki/Lista_de_campos_de_cabe%C3%A7alho_HTTP

5.2. ESP Servidor WEB – HTTP

Um *ServidorWEB* troca *Mensagens HTTP* com um, ou vários, Browsers, através da Internet. De facto a Internet é usada por várias aplicações para transmitirem as suas mensagens através de todo o mundo. Alguns exemplos são os *ClienteFTP/ServidorFTP*, *Telnet/ServidorTelnet*, e o *Email/Servidor Email*.
Usa-se habitualmente a expressão Internet mas de facto além do *Internet Protocol IP* é também usado, simultaneamente, o *Transmission Control Protocol TCP*.
O computador de destino, onde reside o servidor, ao receber uma dessas mensagens pela Internet, com base no seu "*TCP port number*", entrega-a à aplicação receptora mais adequada, pois só esta a saberá processar correctamente.

De uma forma simplista podemos afirmar que um servidor WEB não é mais que um programa que aceita ligações TCPIP, habitualmente na porta 80, e que responde com o conteúdo dos documentos HTML pedidos pelo Browser. Os pedidos do Browser consistem em mensagens de texto, enviadas por TCPIP para o servidor (ex. GET \index.html). A sintaxe dessas mensagens/pedidos/respostas é definida pelo protocolo HTTP.

Entre outros, os programas *Word*, *Power Point*, e *Excel* permitem gravar documentos num formato *HTML* criando dessa forma páginas WEB. Estes programas permitem também incluir num documento referências a outros documentos ("Links - hiperligações").

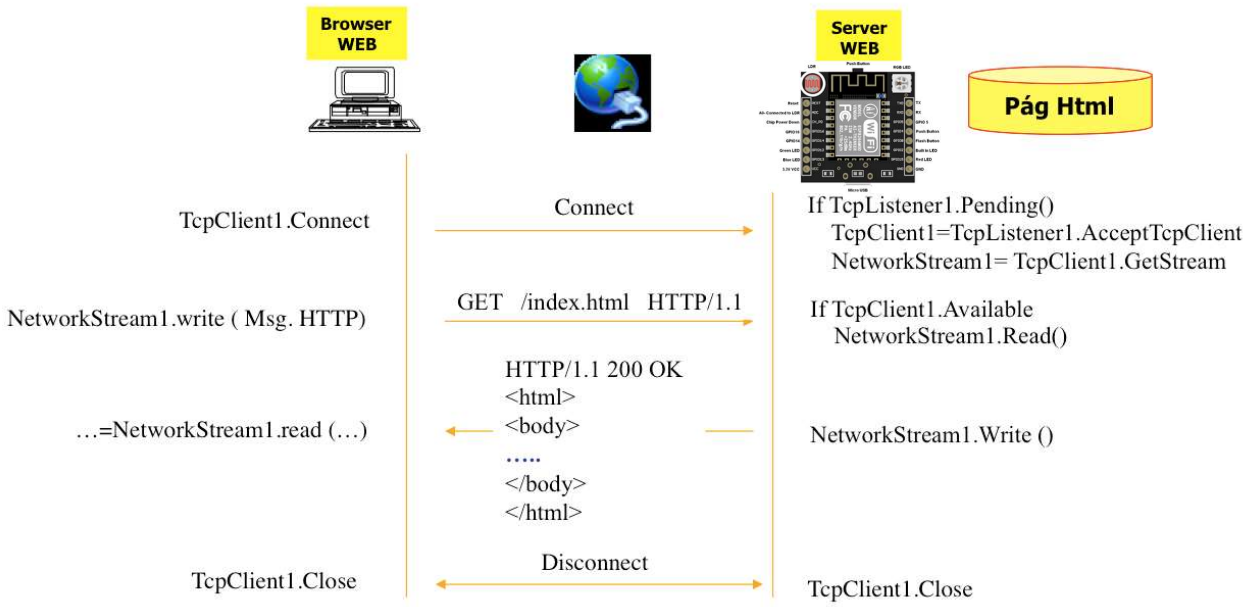


Figura 10.14: Troca de mensagens HTTP usando os objetos TcpCient, TcpListener, e NetworkStream

5.3. Exemplo – ESP Servidor WEB elementar

De facto, a mensagem HTTP é mais complexa. Além do documento HTML, começado por <html> e terminado por </html> , a mensagem HTTP começa pelo seu cabeçalho, por exemplo:

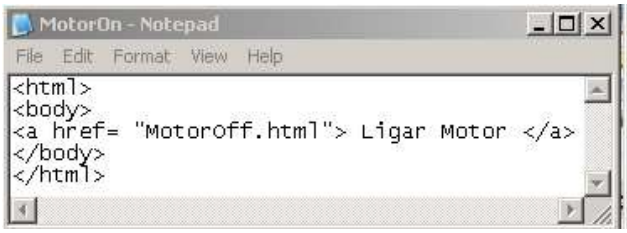
```
HTTP/1.1 200 OK
Date: Mon, 20 Nov 2017 00:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 20 Jul 2017 19:15:56 GMT
Content-Length: 38
Content-Type: text/html
Connection: Closed
<html>
  <body>

  Esta e' uma pagina de teste

  </body>
</html>
```

Relembre o exemplo Exemplo ESP8266 – Wifi 802.11 WebServer

5.3.1. Exemplo – ESP Servidor WEB com duas páginas HTML

Neste exemplo, o servidor WEB, desenvolvido num ESP, deve identificar o nome do documento pedido pelo Browser, abrir o ficheiro em disco e enviar o seu conteúdo para o Browser. O documento MotorOn.html tem uma hiperligação para o documento MotorOff.html e vice-versa. Dessa forma, quando o utilizador, no Browser, "clicar" na hiperligação, pede ao servidor WEB o outro documento HTML.

MotorOn.html

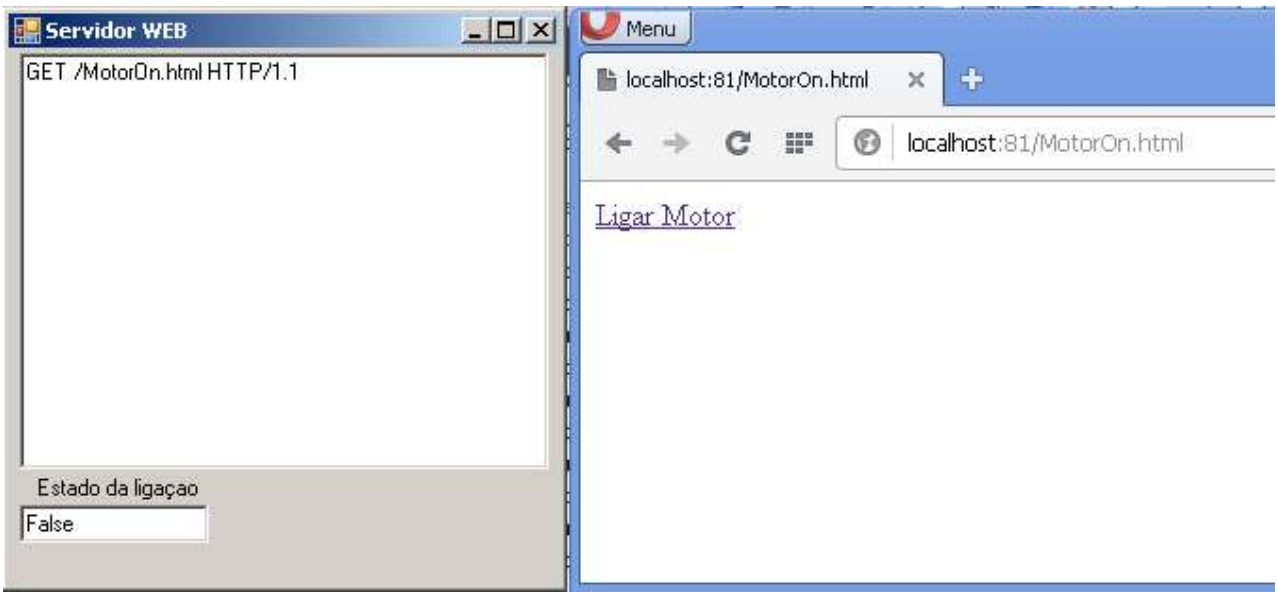


MotorOff.html

A partir do Browser, deve pedir ao servidorWEB, os documentos:
http://localhost/MotorOn.html
http://localhost/MotorOff.html

O Browser estabelece uma ligação TCPIP com o ServidorWeb e envia-lhe a mensagem HTTP com o nome do documento HTML pretendido:
GET /MotorOn.html ... ou GET /MotorOff.html ...

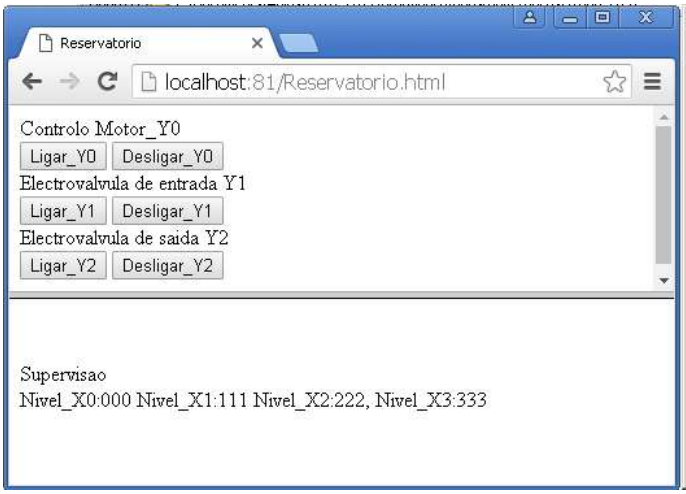
Como a mensagem HTTP contém o nome do documento HTML pedido pelo utilizador/Browser, o ServidorWEB lê o ficheiro pedido e envia o seu conteúdo para o Browser.



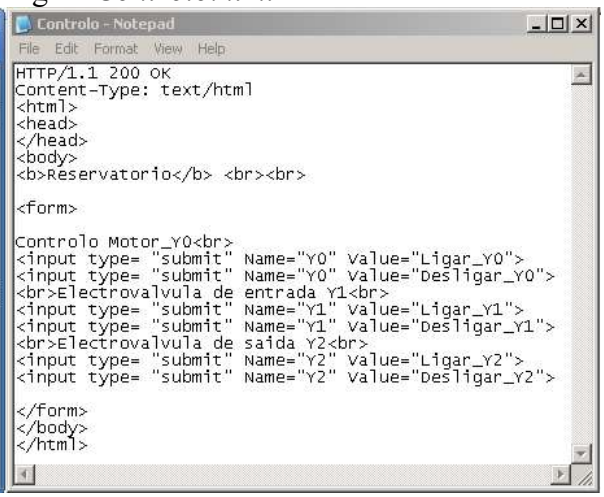
5.3.2. Exemplo – ESP Servidor WEB com três páginas HTML

Neste exemplo, o ServidorWEB, desenvolvido em ESP, deve enviar três documentos HTML para o Browser.
A figura seguinte apresenta os três documentos HTML e o Browser.
Quando o Browser pedir o documento "Reservatorio.html", mostra duas "frames". A "frame" de cima apresenta o documento "Controlo.html", com os botões para ligar ou desligar Y0, Y1 e Y2. A "frame" de baixo apresenta o documento "Supervisao.html", com o estado dos NívelX0, X1, X2 e X3. A página "Reservatorio.html" define a organização da página que o utilizador verá no Browser, em duas "Frames", e que documento HTML aparece em cada "Frame".

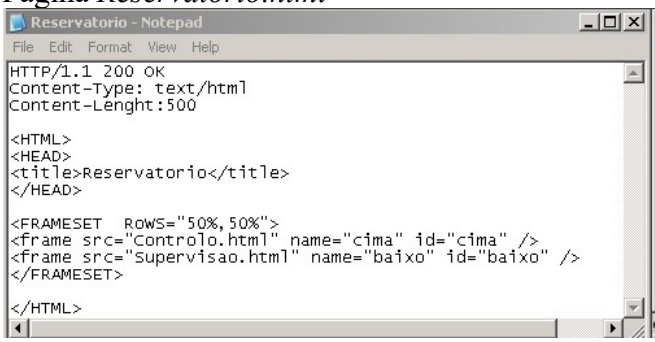
Browser



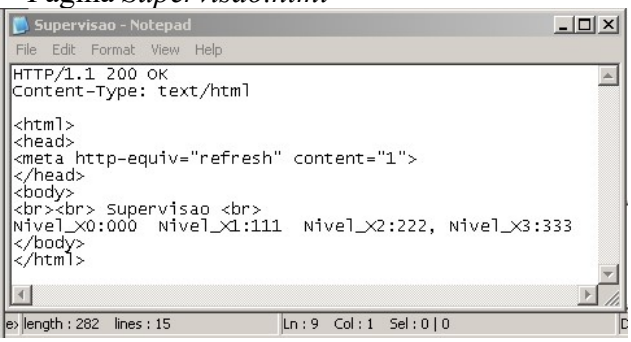
Página *Controlo.html*



Página *Reservatorio.html*



Página *Supervisao.html*

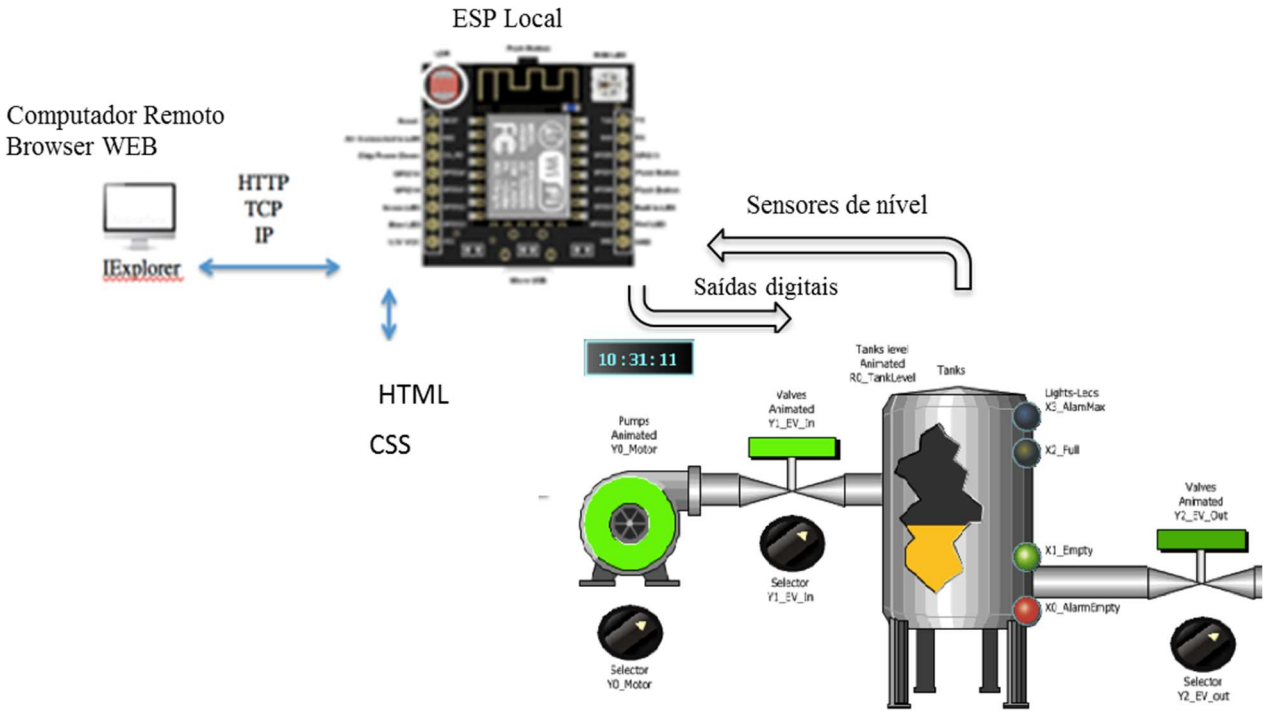


Altere o documento Reservatorio.html para usar DIV e CSS.
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_float

Trab nº 3 Módulo IOT, Servidor WEB/HTTP para controlo e supervisão do Reservatório

Este trabalho tem por objectivo facilitar a compreensão do protocolo HTTP, lecionado nas aulas teóricas. Para isso, pretende-se desenvolver com um ESP um servidor WEB/HTTP elementar que possa responder a pedidos de um browser WEB.

Neste trabalho (uma semana), pretende-se que um utilizador em qualquer ponto do mundo possa usar um **Browser WEB** para comunicar com um **ESP local**, e dessa forma controlar e monitorizar remotamente um processo industrial (ex. reservatório de água).



O **ESP local**, tem o **servidorWEB** desenvolvido neste trabalho. O ServidorWEB desenvolvido, aceita os pedidos de ligação TCPIP e os pedidos de documentos HTML (GET ...) efectuados pelo browser WEB remoto, de acordo com o protocolo HTTP.
Escreva o texto HTML e CSS no código/programa do ESP local. Quando o ESP/ServidorWEB receber pedidos de documentos HTML, lê o seu conteúdo, responde ao BrowserWEB remoto, e ativa as suas saídas digitais conforme as ordens recebidas para controlar o reservatório.

Três saídas digitais do ESP:
Com três saídas digitais do ESP à sua escolha, pretende-se controlar um motor, uma electroválvula que permita a entrada de água no Reservatório, e uma electroválvula que permita a saída de água do reservatório (por gravidade).

Y0_Motor : corresponde a uma saída digital do ESP, usada para ligar o motor que bombeia a água para o reservatório.

Y1_EV_In: corresponde a uma saída digital do ESP, usada para ativar a electroválvula de entrada de água no reservatório.

Y2_EV_Out: corresponde a uma saída digital do ESP, usada para ativar a electroválvula de saída de água do reservatório.

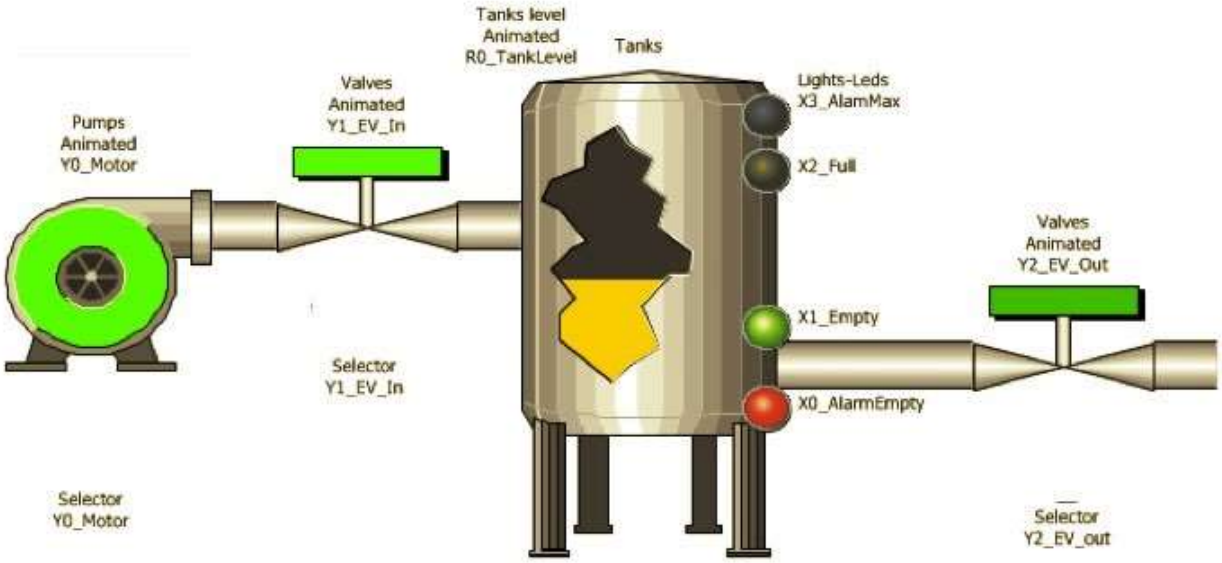
Duas entradas digitais do ESP,
Correspondentes aos 2 sensores de nível de água no reservatório, escolha 2 entradas digitais do ESP. Quando detetam água, os sensores de nível ficam ativos e as entradas digitais do ESP correspondentes também:

X1_Empty: quando o nível de água no reservatório passar a baixo do 20% este sensor deixa de estar ativo.

X2_Full: quando o nível de água passar acima de 90% este sensor fica ativo.

Uma entrada analógica:
R0_TankLevel: corresponde ao sensor de nível analógico. Este sensor gera uma tensão de 0 a 1V em função do nível de água. A entrada analógica do ESP recebe este sinal.

1Volt - corresponde ao reservatório 100% cheio
0 Volt - corresponde ao reservatório vazio 0%



O **computador remoto**, possui apenas um **Browser WEB** para **visualizar o estado dos sensores de nível** do reservatório e o **estado das saídas**: o estado do motor e das electroválvulas de entrada e de saída.
Conforme a figura seguinte ilustra, no computador remoto, o Browser WEB deve permitir ao utilizador **controlar o estado do motor e das electroválvulas (saídas Y0_Motor, Y1_EV_In, Y2_EV_Out); monitorizar o estado dos sensores de nível (entradas X0,X1,X2,X3); e monitorizar o nível de água**.

Quando o utilizador no browser premir um dos botões, por exemplo o botão "Ligar_Y0", é enviado para o servidor WEB a mensagem HTTP:

"**GET /Controlo.html?Y0=Ligar_Y0**"

Nessa altura, no ESP local deve ativar a saída digital "Y0_Motor".

A frame da esquerda, apresenta a referência à imagem:
<img src="http://www.ua.pt/images/40anos.png" width=300 height=150>

Conhecimentos a adquirir
Introdução ao protocolo HTTP e ao funcionamento dos Servidores WEB.
Familiarização com páginas HTML e CSS
Programação ESP

- Sequência de interações Cliente WEB – servidor WEB
- Sintaxe dos pedidos HTTP e respectivas respostas.
- Estrutura e sintaxe dos documentos HTML

Importante:
- O trabalho será avaliado por questionário individual, na semana seguinte à entrega do mesmo.
- Deve submeter no Moodle o programa desenvolvido sob pena da nota obtida no questionário não ser considerada.

Bibliografia

• [Rosen 2003] **Web Application Architecture, Principles, Protocols and Practices**, Leon Shklar, Richard Rosen, John Wiley & Sons, 2003
Chapter 3. Birth of the World Wide Web: HTTP
Chapter 4. Web Servers
Chapter 6. HTML and Its Roots

Norma HTTP
https://www.ietf.org/rfc/rfc2616.txt
http://www.w3.org/Protocols/rfc2616/rfc2616.html

http-protocol.wmv (25 min)
https://www.youtube.com/watch?v=WGJrLqtX7As

Introdução ao HTML
http://www.w3schools.com
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_float