INCORPORACIÓN DE AJAX EN EL SISTEMA EMBEBIDO

AJAX son las siglas de **Asynchronous JavaScript And XML, (Javascript asíncrono y XML)**.

Se trata de una técnica de desarrollo en Internet para crear aplicaciones web asíncronas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible interactuar con el servidor sin necesidad de recargar la página web, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX no es una sola tecnología, ni es un lenguaje de programación. Como se dijo antes, AJAX es un conjunto de técnicas de desarrollo web. El sistema generalmente comprende:

- **HTML/XHTML** para el lenguaje principal y CSS para la presentación.
- **El Modelo de objetos del documento (DOM)** para datos de visualización dinámicos y su interacción.
- **XML** para el intercambio de datos y XSLT para su manipulación. Muchos desarrolladores han comenzado a reemplazarlo por JSON porque es más similar a JavaScript en su forma.
- El objeto **XMLHttpRequest** para la comunicación asíncrona.
- Finalmente, el lenguaje de programación **JavaScript** para unir todas estas tecnologías.

Es necesario algún conocimiento técnico para entenderlo completamente. Sin embargo, el procedimiento general de cómo funciona AJAX es relativamente simple. Revisa la tabla comparativa y el diagrama para clarificar la información.

AJAX - Evento onreadystatechange

Cuando una petición a un servidor se envía, queremos realizar algunas acciones con base a la respuesta.
El evento onreadystatechange se dispara cada vez que cambia el readyState.
La propiedad readyState tiene el estatus de XMLHttpRequest.

Tres importantes propiedades del objeto XMLHttpRequest:

Propiedad	Descripción
onreadystatechange	Almacena una función (o el nombre de una función) que se llamará automáticamente cada vez que cambia la propiedad readyState
readyState	Mantiene el estado de XMLHttpRequest. Cambia de 0 a 4: 0: petición no inicializada 1: conexión establecida 2: petición recibida 3: procesamiento solicitudes 4: solicitud terminada y la respuesta está lista
status	200: "OK" 404: Página no encontrada. 500: error interno del servidor.

En onreadystatechange, se especifica lo que sucederá cuando la respuesta del servidor está lista para ser procesado.

Cuando readyState es 4 y el estado es 200, la respuesta es la funcionalmente operativa.

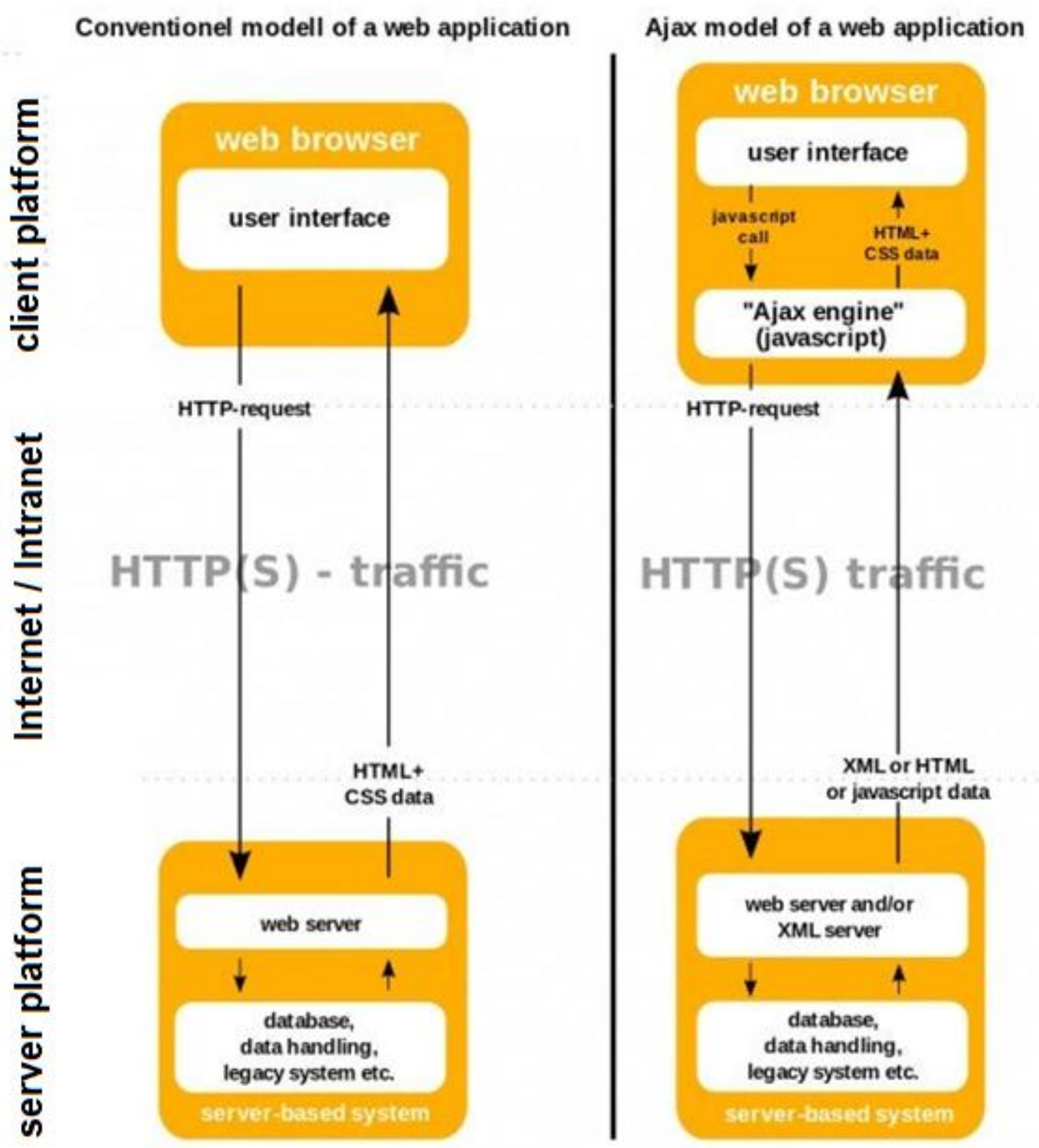
Tabla comparativa:

Modelo convencional	**Modelo AJAX**
1. Se envía una solicitud HTTP desde el navegador web al servidor.	1. El navegador crea una llamada de JavaScript que luego activará XMLHttpRequest.
2. El servidor recibe y, posteriormente, recupera los datos.	2. En segundo plano, el navegador web crea una solicitud HTTP al servidor.
3. El servidor envía los datos solicitados al navegador web.	3. El servidor recibe, recupera y envía los datos al navegador web.
4. El navegador web recibe los datos y vuelve a cargar la página para que aparezcan los datos.	4. El navegador web recibe los datos solicitados que aparecerán directamente en la página. No se necesita recargar.
Durante este proceso, los usuarios no tienen más remedio que esperar hasta que se complete todo el proceso. No solo consume mucho tiempo, sino que también supone una carga innecesaria en el servidor.	Se actualiza solo partes de la página Web del sistema embebido, haciendo más ligera la carga al servidor.

REFERENCIAS:

https://www.um.es/docencia/barzana/DAWEB/Desarrollo-de-aplicaciones-web-AJAX.html
https://developer.mozilla.org/es/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=924:xmlhttprequest-ajax-propiedades-status-onreadystatechange-readystate-responsetext-o-xml-cu01207f&catid=83&Itemid=212
https://www.hostinger.mx/tutoriales/que-es-ajax#Ejemplos_practicos_de_AJAX

Diagrama:


Conventionel modell of a web application
Ajax model of a web application
client platform
Internet / Intranet
server platform
web browser
user interface
HTTP-request
HTTP(S) - traffic
HTML+
CSS data
web server
database,
data handling,
legacy system etc.
server-based system
web browser
user interface
javascript
call
HTML+
CSS data
"Ajax engine"
(javascript)
HTTP-request
HTTP(S) traffic
XML or HTML
or javascript data
web server and/or
XML server
database,
data handling,
legacy system etc.
server-based system

INDICACIONES: Compilar el siguiente código mediante la extensión Platformio, posteriormente grabar en el hardware:

```
String header = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";
// Inicia la cadena cruda
String CadenaCruda = R"=====(
<!DOCTYPE html>
<html>
 <head>
  <meta name='viewport' content='width=device-width, initial-scale=1.0'/>
  <meta charset='utf-8'>
  <style>
    body {font-size:100%;}
    #main {display: table; margin: auto;  padding: 0 10px 0 10px; }
    h2 {text-align:center; }
    p { text-align:center; }
  </style>
   <script>
    function actualizaContador()
    // al terminar su conteo descendente
    // el timer ejecuta obtenerCuenta
    {
       ajaxLoad('obtenerCuenta');
    }
   // Aquí está la parte de Ajax:
     var SolicitudAjax = null;
    if (window.XMLHttpRequest)  { SolicitudAjax =new XMLHttpRequest(); } /* PASO 1: Obtener la instancia del objeto XMLHttpRequest  */
    /* El objeto XMLHttpRequest Es el que permite la comunicación asíncrona (en segundo plano)
     con el servidor. */
    else                        { SolicitudAjax =new ActiveXObject("Microsoft.XMLHTTP"); }

   function ajaxLoad(ajaxURL)
    {
      if(!SolicitudAjax) { alert('AJAX is not supported.'); return; }

     SolicitudAjax.open('GET',ajaxURL,true); //  PASO  2:  Realiza la petición HTTP
      SolicitudAjax.onreadystatechange = function() // PASO 3: Prepara la función de respuesta
      {
        if(SolicitudAjax.readyState == 4 && SolicitudAjax.status==200) //PASO 4: Ejecuta la respuesta
        {
          var ResultadoDeAjax = SolicitudAjax.responseText;
          document.getElementById('cont_ParteDePagina').innerHTML = ResultadoDeAjax;
          //hace referencia a la parte que se va a actualizar
        }
      }
      SolicitudAjax.send();
    }  // PASO 5: Se realiza la actualización parcial en la página Web
 setInterval(actualizaContador, 200); //Ejecuta por siempre en un intervalo de 200 ms
```

```
    // Cuando el timer inicia, es llamado actualizaContador
    // después de 200 ms (porque el timer tiene un contador descendente)
    // actualizaContador llama ajaxLoad()y es cuando se realiza el
    // trabajo de desplegar el valor actualizado del contador

  </script>
  <title>Auto refresh parcial con AJAX</title>
 </head>

 <body>
   <div id='main'>
     <h2>Ejemplo autoactualizable usando AJAX (Asynchronous JavaScript And XML)</h2>
     <!--establece un elemento id indicando que es el contador
         Como parte de la página Web a autoactualizar:  -->
       <p id='cont_ParteDePagina'>cont = 0</p>
     <!-- cont inicia en cero -->
     </div>
   </div>
 </body>
</html>
)====="; // Se cierra cadena cruda


#include <ESP8266WiFi.h>

const char* ssid = "MEGACABLE-F72429F";
const char* password = "ltw23445TtMd";

WiFiServer server(80);

String request = "";
unsigned int cont = 0;

void setup()
{
    Serial.begin(115200);
    Serial.println();
    Serial.println("Puerto serie a velocidad 115200");
    Serial.println();

    // Conectar a la red WiFi
    Serial.print(F("Conectando a "));  Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(500);
    }
```

```
    Serial.println("");
    Serial.println(F("[CONNECTED]"));
    Serial.print("[IP ");
    Serial.print(WiFi.localIP());
    Serial.println("]");

    // start a server
    server.begin();
    Serial.println("Servidor iniciado");
 }

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 void loop()
{
    // Verifica si un cliente se ha conectado
    WiFiClient client = server.available();
    if (!client)  {  return;  }

    // Lee la primera linea de la petición
    request = client.readStringUntil('\r');
    // El servidor está constantemente verificando por si hay una
    // solicitud, y cuando obtiene una, verifica si la petición
    // incluye la cadena "obtenerCuenta"
    if ( request.indexOf("obtenerCuenta") > 0 ) // si la petición
    // contiene la cadena "obtenerCuenta":
   {
        cont ++; // se incrementa contador
        client.print( header );  // se despliega en página Web
        client.print("<html>");
        client.print( "<h2> contador </h2>" );
        client.print( "<h2>" );
        client.print(  cont );
        client.print( "</h2>" );
        client.print("</html>");
        Serial.print("cont="); //Se despliega en monitor
        Serial.println(cont);
     }
      else
     // de lo contrario que la petición NO CONTIENE obtenerCuenta
     //enviamos la página Web y reseteamos cont a cero
     {
        client.flush();
        client.print( header );
        client.print( CadenaCruda );
        cont = 0;
     }
     delay(5);
 }
```

PRUEBA: Al ejecutarse se deberá observar lo siguiente: Observar que ya no se está actualizando la página web entera en el navegador sino solo la parte de ella referente al valor del contador.