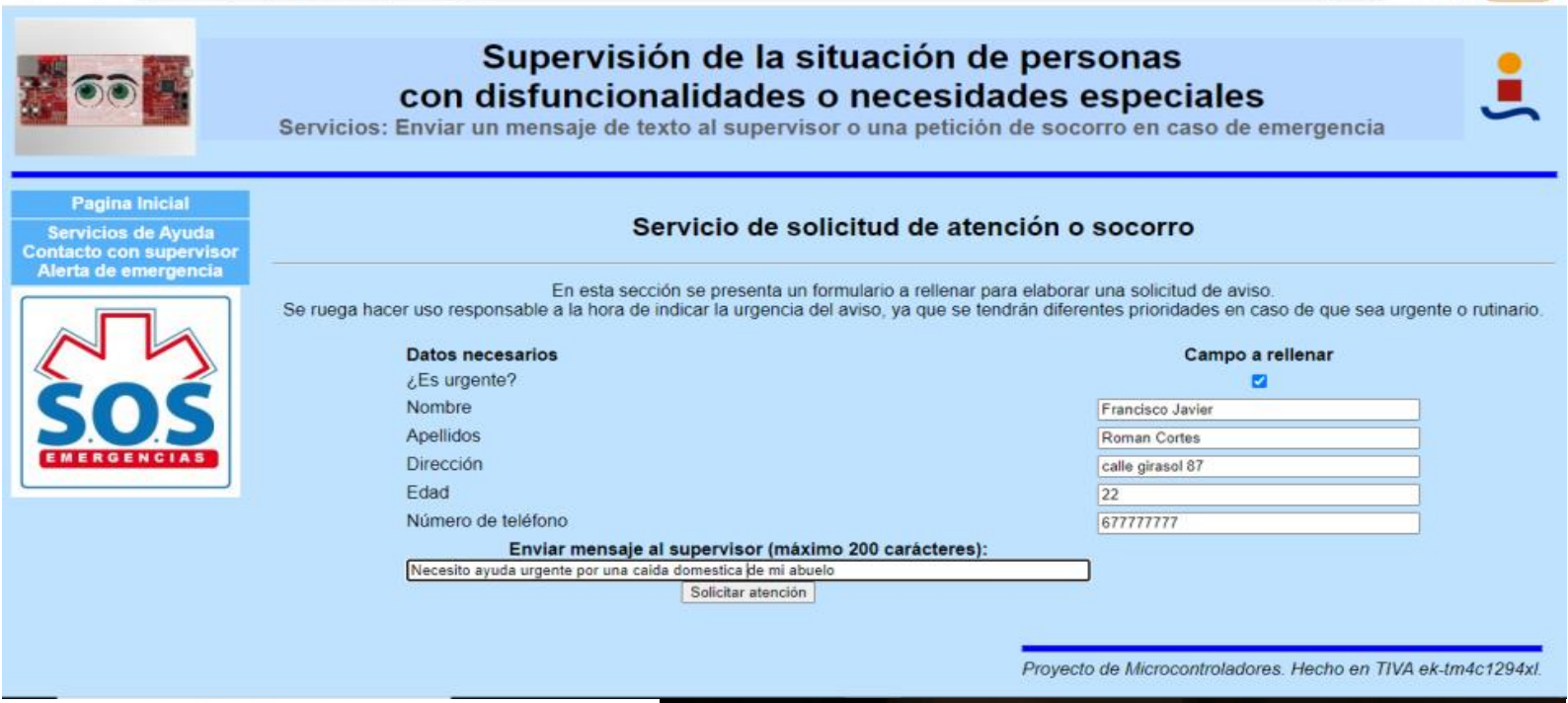
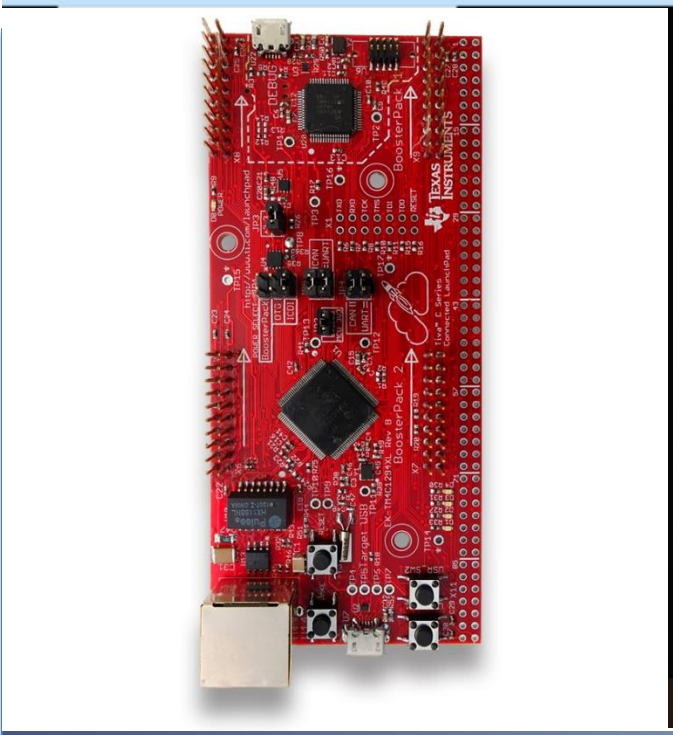



# PROYECTO DE MICROCONTROLADORES:

## SISTEMA DE SUPERVISIÓN PARA PERSONAS CON DIVERSIDAD FUNCIONAL

FRANCISCO JAVIER ROMÁN CORTÉS
JUAN DE DIOS HERRERA HURTADO

# ÍNDICE

# 1.- Idea del proyecto

## 1.1.- Idea general del proyecto

El proyecto que se plantea consiste en un sistema de supervisión de personas con necesidades especiales tales como personas con diversidad funcional (discapacidad o minusvalía) o personas mayores entre otros, cuyo día a día puede ser complicado sin la ayuda de terceros.

Si por ejemplo una persona mayor necesita comprar medicinas, pero no puede valerse por ella misma para hacerlo y no dispone de ninguna persona que pueda realizar dicha tarea por ella, es aquí cuando entra en juego las posibilidades que ofrece nuestro sistema, de manera que un posible supervisor recibiera un aviso y acudiese a realizar dicha tarea.

Aunque el enfoque general está orientado hacia la prestación de servicios para las personas con diversidad funcional, que no pueden valerse por sí mismas en ciertas tareas o incluso pueden verse desvalidas en situaciones de emergencia, también se podrían extender las posibilidades del sistema planteado a la atención de emergencias médicas graves donde se requiere que el aviso llegue rápido y sea atendido lo más pronto posible.

La idea general y en su versión más potente, es la de montar un sistema que tenga soporte en un servidor y desde una aplicación móvil o directamente desde la web, dando libertad al usuario para usar la que mejor le venga en ese momento, se tuviese la posibilidad de recibir avisos de cualquier persona que se conecte remotamente a la página (mediante web o app) e irlos almacenando en la aplicación que ejecuta el microcontrolador de manera ordenada y clasificada para su correcta supervisión.

Esto sería algo bastante potente ya que permitiría centralizar la llegada de avisos en un sólo lugar (una posible centralita), desde donde se realizaría toda la gestión y supervisión de los diferentes avisos recibidos según prioridad de la urgencia de cada uno.

A cargo de este sistema podría estar ya sea una organización pública o una empresa privada a la que contratamos sus servicios.

### 1.1.1.- Ventajas:

La ventaja que ofrece este servicio frente al típicamente convencional mediante llamada telefónica se puede ver ilustrada mediante el siguiente ejemplo: Supongamos el contexto de una emergencia médica en la que se llama al hospital, con nuestro sistema (suponiendo que está constantemente supervisado), el aviso llegaría a la centralita de manera inmediata y la persona que se está encargando de gestionar los avisos podría actuar rápidamente, mientras que por su parte, mediante una llamada telefónica no es extraño (al menos en los tiempos actuales de pandemia, donde los centros sanitarios reciben multitud de llamadas) que las líneas se encuentren ocupadas o que nos salte un contestador, quedando la situación de emergencia sin ser atendida.

Además, nuestro sistema permite al usuario indicar si la solicitud es urgente o no, permitiendo que desde la centralita se dé prioridad a los mensajes urgentes para poder actuar de manera más efectiva. Esto también implica que se asume un uso responsable por parte de los usuarios a la hora de indicar si el aviso es urgente o no, para que realmente sólo tengamos clasificados como urgentes los avisos o situaciones que lo son.

Podemos entender como avisos no urgentes aquellos correspondientes a:

- Necesidad de realizar compra de elementos de primera necesidad (comida, medicación…), pero no es posible desplazamiento autónomo de la persona.
- Tareas domésticas.
- Tareas de cuidado personal (aseo, movilidad…).

Por su parte se pueden definir algunos ejemplos que se esperan como avisos urgentes:

- Cualquier situación en que peligre la integridad física de la persona.
- Carencia de algún medicamento fundamental.
- etc.

### 1.1.2.- Funcionamiento planteado para dicha idea general:

Diferenciamos dos estructuras fundamentales que actúan conjuntamente en el sistema:

***- La página web (hosteada por el micro en nuestro desarrollo pero idealmente residiría en un servidor para poder ser accesible remotamente) que presta los servicios de poder solicitar atención (sea urgente o no):***

- La persona o alguien cercano a la persona (si la propia persona es incapaz de formalizar el aviso) que necesita ayuda envía el aviso mediante la app móvil o desde la página web. La interfaz (de la página web) ofrece en primer lugar una portada con diferentes imágenes, las cuales incluyen enlaces de interés. A la izquierda, en color azul, tenemos un menú donde podemos movernos entre dos pestañas:
    - La página principal (con las imágenes y enlaces).
    - La pestaña donde se puede formalizar el aviso y enviarlo.
- En la ventana donde se ofrecen los servicios de formalización del aviso aparecen los siguientes campos de información a rellenar:
    - Tipo de mensaje → se indica si se trata de una petición urgente o no. Esto permitirá atender los avisos dando prioridad a las peticiones urgentes.
    - Nombre
    - Apellidos
    - Dirección → este será el dato más importante que permitirá a la organización que se encarga del sistema acudir al lugar donde se encuentra la persona que necesita ayuda. Idealmente, en la idea general se daría la opción de adjuntar la ubicación mediante Google Maps por si la persona no se encuentra en una población, por ejemplo, en una montaña o lugar difícil de identificar.
    - Edad
    - Teléfono → en caso de que la organización necesite contactar con la persona que solicita la ayuda.
    - Mensaje → permite al usuario explicar con detalle su necesidad de manera que la organización pueda actuar de manera más efectiva sabiendo a qué situación deben responder.

***- La aplicación que recibe los avisos o solicitudes de ayuda (basada en una especie de registro donde los avisos se encolan clasificándose según su urgencia y por orden de llegada):***

En la idea general, lo ideal para este sistema planteado sería alguna especie de aplicación complementaria ya sea en ordenador o en sistemas móviles (Android o iOS) que se comunique con la página web y permita un acceso cómodo a los diferentes registros que almacenan los avisos o solicitudes (por ejemplo, desplazándose entre páginas de aviso, de manera táctil, etc.).

- El aviso llega a la centralita donde se activa una pequeña alarma que avisa a la persona encargada de gestionar el aviso. Esta consultará los datos de la petición y enviará la ayuda.
- Sean atendidos o no, los avisos se irán almacenando en su registro correspondiente por orden de llegada y siendo clasificados en función de su urgencia.
- Una vez se considera adecuado por parte del supervisor, encargado de gestionar los avisos, se enviará ayuda o asistencia a la ubicación indicada con el fin de satisfacer la petición solicitada.
- De esta manera, una vez se indica en esta aplicación que la ayuda ha sido enviada (entendiendo que realmente se ha enviado) el registro que contenía el aviso se vacía, y en caso de que hubiese avisos posteriores en tiempo (más recientes), se desplazarían todos un lugar hacia arriba en dicho registro, ocupando el aviso posterior al atendido el lugar que ha dejado vacante el que ha sido atendido.

## 2.- Justificación del uso del microcontrolador TM4C1294

Vemos como la idea propuesta es tecnológicamente exigente, es decir, posiblemente extendería las posibilidades de este microcontrolador.

Sin embargo, como explicaremos en el siguiente apartado, sí que permite realizar un prototipado prácticamente similar en cuanto a funcionamiento a la idea propuesta, con la salvedad de que le faltaría el escalado en cuanto a conexionado remoto principalmente (nuestro prototipo permite básicamente conexión a la web hosteada por el micro via Ethernet, es decir, la web sólo es accesible e interacciona con el micro desde el PC conectado al micro via Ethernet. Por tanto, no se podría en dicho prototipo acceder a la web e interactuar con el sistema de manera remota, desde otro PC ajeno por ejemplo).
El servicio de atención planteado, considerando ya el prototipo realizado con sus limitaciones mencionadas, requiere ciertas características técnicas necesarias como:

- Tener suficiente capacidad para almacenar gran densidad de datos, en nuestro caso, se debe poder almacenar hasta 40 estructuras (separadas en dos arrays de structs de dimensión [20] para almacenar tanto avisos urgentes como no urgentes) compuestas por 8 campos formados por cadenas de caracteres de tamaño 50 caracteres al menos. Cada una de estas estructuras contiene la información de un aviso.

  Adicionalmente, el micro debe poder almacenar las imágenes y demás elementos que aparecerán en la página web.
  Estos dos principales requisitos exigen un mínimo de capacidad de almacenaje que nuestro microcontrolador cumple con solvencia.

  Toda la información relacionada con la página web se almacena en la memoria flash del microcontrolador, con lo cual tener una cantidad suficiente de memoria flash disponible es una especificación fundamental a cumplir por el microcontrolador para la aplicación desarrollada, ya que si recordamos el microcontrolador de la asignatura anterior, MSP430 con 16kB de memoria flash disponible, estaría muy lejos de poder soportar esta aplicación simplemente sólo por esta característica.

A continuación, mostraremos el uso de memoria de nuestro programa desarrollado:

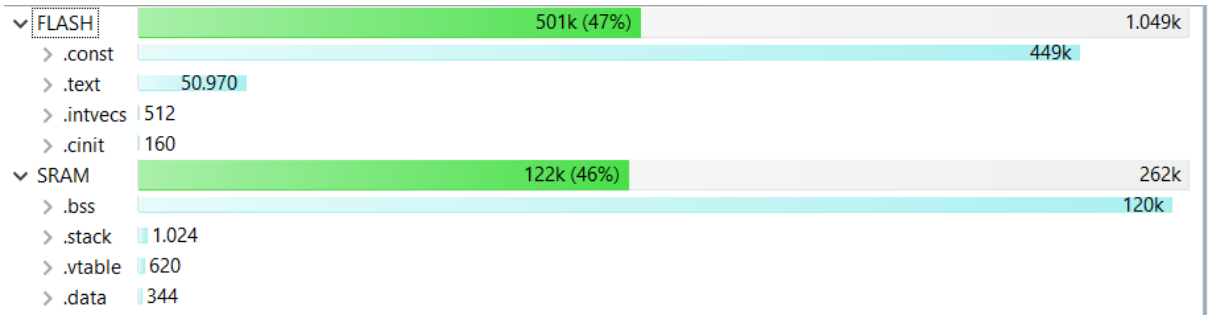



***Figura 1:*** *Consumo de memoria del programa*

Como podemos observar, a pesar de tener que almacenar gran cantidad de datos, sobra una cantidad de memoria considerable.

Comentar que, para la idea del proyecto, la cantidad de avisos que se deberían poder almacenar sería mucho mayor, hablaríamos de cientos de avisos tanto urgentes como no urgentes. En nuestro caso, se ha decidido para el prototipo realizado limitar a 20 avisos urgentes y a 20 avisos no urgentes, aunque se podría ampliar sin problema a unos 50 de cada tipo vista la cantidad de memoria disponible.

- Alta velocidad de cómputo para poder procesar los avisos entrantes de manera rápida (ejecución rápida de interrupciones de Ethernet entre otras tareas que requieren cierta potencia) con el fin de poder actuar frente a una petición lo antes posible y evitar un colapso del sistema en el supuesto de varias fuentes de llegada de avisos.

- Este desarrollo necesita además que el microcontrolador deba estar manejando a la vez timers, interrupciones de los timers y las que lleguen de la página, el controlador Ethernet, la pantalla junto con gran cantidad de datos y animaciones que mostrar, el módulo PWM para mover el servomotor, funciones propias de la librería “string.h” para poder manejar con relativa comodidad las distintas cadenas que componen las estructuras de los avisos o grandes bucles en los que se está trabajando con cadenas de gran longitud. Por todo ello, la funcionalidad propuesta demanda que el microcontrolador sea potente, lo cual encaja en el marco de características que ofrece el TM4C1294.

  Los periféricos utilizados en el programa son:
  - Ethernet
  - Dos timers
  - Módulo PWM

- Puerto SPI para la comunicación con la pantalla
- UART para mostrar los mensajes hasta que recibimos la dirección IP procedente del protocolo DHCP así como su uso a la hora de realizar un correcto debug del programa implementado.

Además, se cuenta con la pantalla FT800 que es la herramienta a través de la cual el usuario (supervisor) interactúa con el programa y obtiene la información procedente de la persona que solicita atención (mediante la página web).
Debido a que es un recurso muy versátil del que disponemos, se ha desarrollado la interfaz de la aplicación para este prototipo sobre esta pantalla táctil.
En ella, se ha tratado de plantear una aplicación con una interfaz que persigue el objetivo de ser “amigable” y cómoda para el usuario, con contrastes de colores adecuados y reparto equilibrado de los elementos informativos (diferentes mensajes, tablas de registro, animaciones…).
De esta manera, aunque se trate de un prototipo que no alcanza las pretensiones de la idea general propuesta inicialmente, podría ser un claro ejemplo de proceso de prototipado de una idea o producto competitivo el cual termina siendo desarrollado en un proceso más extenso en tiempo y con un equipo y recursos más amplios para elaborar la aplicación definida inicialmente en el apartado de “idea general del proyecto”.

# 3.- Desarrollo del proyecto

## 3.1.- Solución de la idea propuesta

Nuestra solución del sistema planteado consiste en una página web hosteada desde el microcontrolador que se comunica con el microcontrolador a través de Ethernet. Para ello, se ha hecho uso del periférico Ethernet basándonos en el ejemplo disponible en: C:\ti\TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c1294xl\enet_io

El microcontrolador usa el protocolo DHCP (Dynamic Host Configuration Protocol) para obtener la dirección IP de la página web. El microcontrolador obtiene la información de la página a través de un formulario típico en HTML el cual usa peticiones HTTP mediante el método GET (el método GET envía la información en la propia URL). Ahora se muestra un trozo del código donde aparece esto en el archivo “solicitud_cgi.ssi” de la carpeta “fs”:

```
<form method="get" action="recoger_datos.cgi" name="recoger_datos">
```

Del formulario se obtienen los datos usando el método GET. Al pulsar el botón para enviar los datos, tenemos la acción “recoger_datos.cgi” que hará que el manejador procese los datos del formulario, es decir, este manejador saltará y procesará los datos entrantes cada vez que se envíe un formulario desde la página.

La principal limitación de esta solución, y por lo cual se dice que es un prototipo de la idea general planteada anteriormente, es que no se puede acceder a la página web de manera remota, ya que el microcontrolador está conectado por Ethernet al ordenador que carga el programa en el microcontrolador.

Para la realización del programa, hemos partido como ya se ha comentado del ejemplo “enet_io”, de TivaWare. Partiendo de la comprensión del ejemplo y las librerías que usa, hemos hecho uso y adaptado en cierta medida algunas funcionalidades relacionadas con Ethernet (quedándonos con lo fundamental para satisfacer las necesidades de nuestra aplicación) de manera que se lograse el prototipo del sistema planteado respecto a su parte de comunicación página web - aplicación, a la vez que se han desarrollado todas las funcionalidades relacionadas con la aplicación que registra los avisos (almacenamiento, procesamiento y actuación).

Cabe destacar que en cuanto a la página web, se han debido hacer modificaciones notorias sobre la plantilla existente con el fin de que la página tuviese una apariencia adaptada a nuestra idea a la vez que nos permitiese obtener toda la información detallada que queríamos mostrar en el programa para cada aviso.

### 3.2.- Explicación de la solución realizada

Aunque ya ha sido mencionado, es importante recalcar que en este apartado el prototipo/solución desarrollado presenta ciertas carencias respecto de la idea general presentada con todo su potencial (posibilidad de conexión remota respecto del sistema propuesto para interaccionar con él sin necesidad de la conexión Ethernet, por ejemplo de manera inalámbrica).
Estas carencias son debidas tanto a limitaciones tanto en infraestructuras (microcontrolador, servidores web, etc.), como en cantidad de tiempo y recursos necesarios para llegar a implementar la idea presentada en todo su esplendor.
Como también se comentaba en el primer apartado de esta memoria, el sistema planteado presenta dos bloques funcionales separables pero cuyo funcionamiento es conjunto.

#### 3.2.1.- Página Web:

Se ha desarrollado una página web simple mediante lenguaje HTML, de manera que permita realizar las funciones ya descritas y necesarias para interactuar con la aplicación cuya interfaz recae sobre la pantalla FT800, siendo que en la página web será donde se conforman los avisos con su información correspondiente y se realizará el envío de dichos avisos.

A la hora de construir esta página web, ya teníamos ciertos conocimientos de la asignatura de 3º “Proyectos Integrados” donde ambos nos dedicamos al diseño de una página web para el proyecto al que nos dedicamos con lo cual medianamente recordábamos cómo montar una pequeña interfaz web con una apariencia agradable dentro de las posibilidades del espacio que ofrece la Flash del microcontrolador.

La página web consta de una **página inicial** con una serie de imágenes representativas de la temática de atención, ayuda y diversidad funcional, siendo que algunas de ellas contienen enlaces de interés, por ejemplo, a la página de Texas Instruments, a la página de la ETSI o bien a un blog que recoge las diferentes asociaciones que ofrecen apoyo a personas con diversidad funcional y enlace a sus diferentes páginas web.

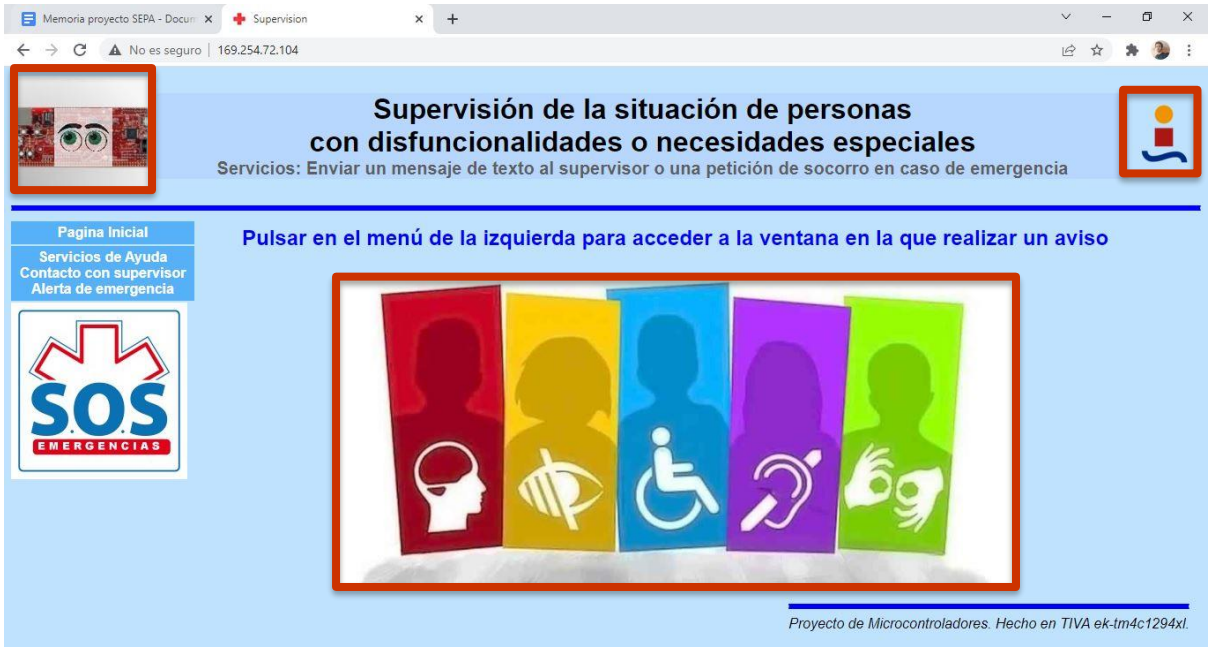



***Figura 2:*** *Página inicial de la web diseñada, con sus diferentes elementos dispuestos a lo largo de la pantalla*

Sobre esta imagen podemos ver la disposición de elementos, donde se ha prestado atención a los diferentes detalles de su apariencia. Tanto el favicon (que es el icono de la pestaña superior), como las diferentes imágenes, tratan de representar la temática particular del proyecto, las personas con diversidad funcional y cómo podemos mejorar la vida de dichas personas.
Si vemos las diferentes imágenes:

- En la esquina superior izquierda el logo del microcontrolador con un par de ojos emula a cómo el microcontrolador “presta atención” a estas personas y si se clica en él, nos redirige a la página de Texas Instruments, diseñador del microcontrolador en el que se sustenta la aplicación.

- En la esquina superior derecha, el logo de la ETSI, representativo de la escuela que nos aporta los conocimientos necesarios para realizar este tipo de proyectos, y si se clica también nos rediriga a la página de la Escuela.

- En la parte central, se observa una imagen representativa del colectivo con personas con diversidad funcional. Esta imagen nos redirige a un blog de “SunriseMedical” que recoge las asociaciones más importantes en este campo en España y presenta cada una de ellas dando enlaces a las mismas.

- La imagen de SOS es simplemente decorativa y no redirige a ninguna otra página.

En cuanto al menú de la izquierda (en azul, sobre la imagen de SOS), presenta las dos pestañas de las que dispone la página, es decir, podemos viajar de la página inicial a la que presta los servicios y viceversa.
Aunque no es contenido tan propio de la asignatura, es parte del trabajo del proyecto por lo que se anexa el código que define esta página inicial:

1.- Index.htm (página inicial de la web):

Mediante dicho código, sin entrar mucho en su análisis, se construye la página mediante la estructura de , que actúan como contenedores para texto, imágenes, etc. pudiendo colocarse según se defina en el archivo **“styles.css” (2.- Styles.css (define los estilos de los diferentes…),**
asignando atributos a cada uno de los “id” de cada par de <divs>.
Destacar que las imágenes, favicon y demás recursos referidos en el código html deben estar almacenados en la carpeta **“fs”** del workspace.

Por otra parte, antes de seguir con la siguiente pestaña de la página web y la que realmente permite realizar la funcionalidad de envío de mensajes, es importante recalcar cómo se compilan los cambios realizados sobre los archivos que definen la página web para cargarlo correctamente en el microcontrolador (memoria Flash) mediante el archivo **“io_fsdata.h”.**
De esta manera, al modificar estos archivos (index.htm, solicitud_cgi.ssi, styles.css, etc), además de reincluirlos en el workspace manualmente, se debe hacer lo siguiente para modificar el archivo “io_fsdata.h” y reemplazarlo en el workspace también manualmente:

- Se incluyen en la carpeta "fs\" los códigos fuentes que conforman dicha página web.
- Si se modifica alguno de estos archivos (para modificar la web), el archivo de imagen del sistema (iofsdata.h) debe ser recompilado mediante la utilidad disponible "makefsfile" disponible en la carpeta de TivaWare.
- Para usar esta herramienta, tenemos que desplazarnos al directorio (en un CMD o terminal) donde esté alojado el proyecto (con cd \..) y tras ello:
  >> ../../../../tools/bin/makefsfile -i fs -o io_fsdata.h -r -h -q
- Posiblemente se puede modificar este path si se desplaza el archivo "makefsfile" a otro lugar deseado.

Tras esta explicación de cómo se hacían efectivas las modificaciones realizadas sobre los archivos y la construcción de la página web, pasamos a la siguiente pestaña de la página, a la que se accede clicando sobre la opción “Servicios de Ayuda / Contacto con Supervisor / Alerta de Emergencia”:

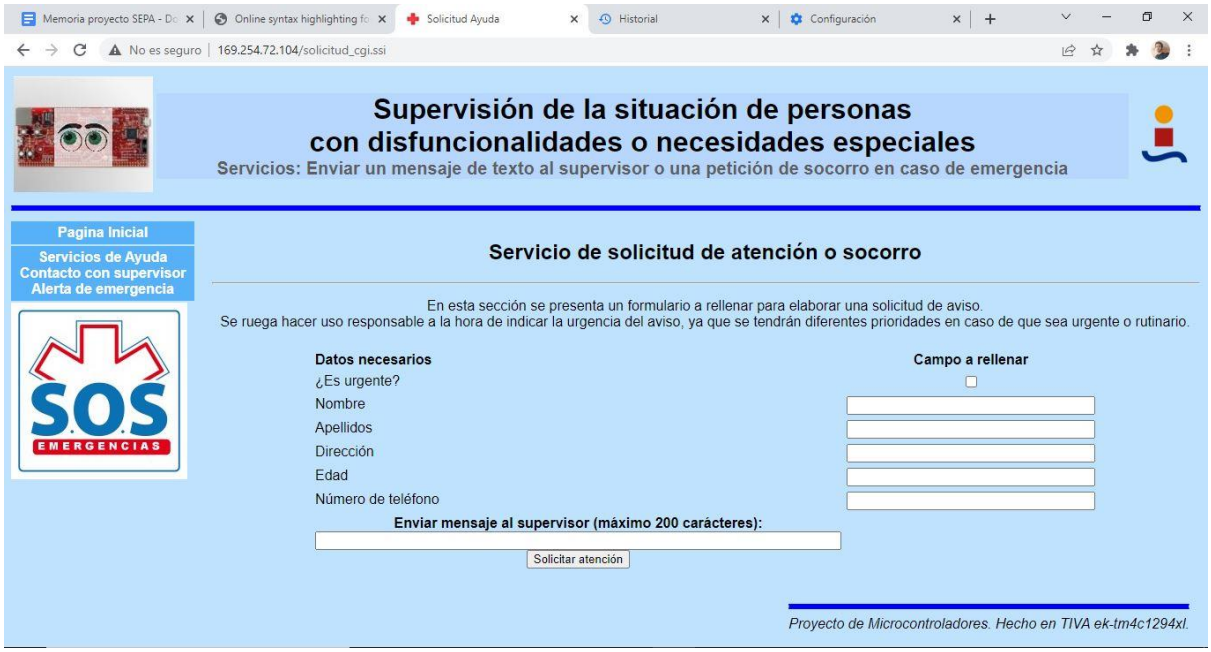



***Figura 3:*** *Pestaña de la web destinada a la formalización del aviso y su envío hacia la aplicación de supervisión*

Vemos en esta figura, la pestaña que cumple la funcionalidad de esta perspectiva del sistema, a través de los diferentes campos a rellenar, donde todos de ellos son necesarios y la página obliga a rellenarlos, a excepción del mensaje inferior el cual sí que es opcional, pero obviamente recomendable. De esta manera una vez se rellenen los campos, ya sea con o sin mensaje, al pulsar el botón “Solicitar atención”, la información llegará al microcontrolador vía Ethernet y será procesada y almacenada ordenadamente según corresponda por orden de llegada y clasificada por su urgencia.
Destacar que, tras enviar el aviso, la página se recarga por propia orden del microcontrolador tras atravesar su handler asociado (HandlerCGI_Receptor_Datos), el cual devuelve “Respuesta_Handler_CGI” que equivale a “/solicitud_cgi.ssi”, que provoca la recarga de dicha página, ya que corresponde al archivo que define esta pestaña en concreto.
La recogida de los datos introducidos aquí se explicará con mayor detalle en el apartado de la aplicación de supervisión.

Finalmente, mostrar cómo responde la página diseñada si no se rellena alguno de los campos obligatorios:

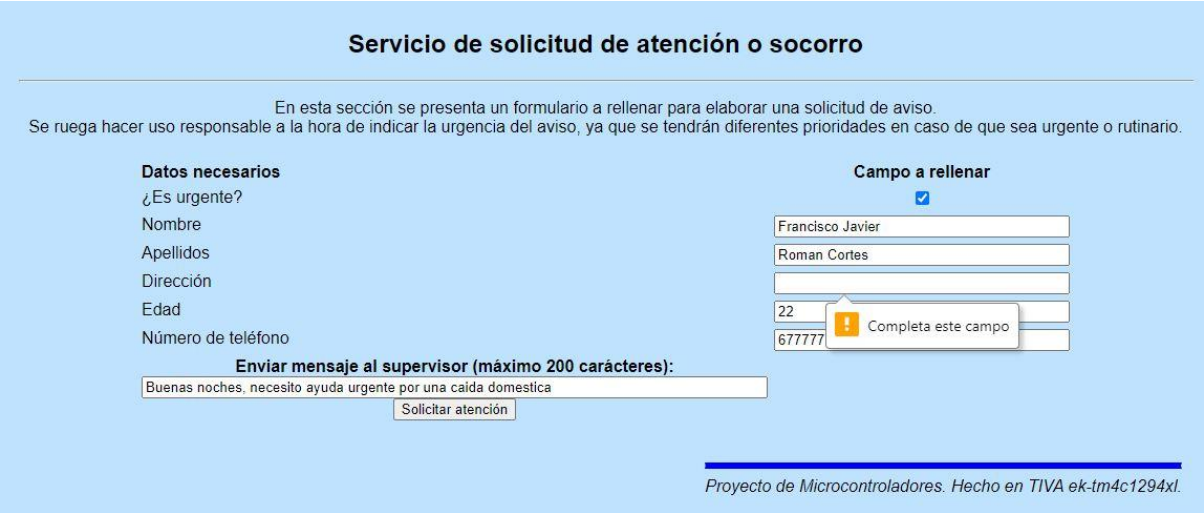



***Figura 4:*** *Respuesta de la página cuando se intenta introducir un aviso con alguno de los campos de datos obligatorios vacío*

En el código de esta pestaña, además de en su apariencia, ha habido que tener en cuenta ciertos aspectos respecto a la funcionalidad del sistema a la hora de poder recoger y procesar adecuadamente los datos en la aplicación de supervisión.
Código anexado: 3.- Solicitud_cgi.ssi (pestaña que permite formalizar y enviar…

Por destacar alguno de los aspectos más importantes respecto de dicho código podemos destacar varias líneas (no son secuenciales, se han extraído del código anexo por ser las más explicativas) de código fundamentales en cuanto a funcionalidad:

```
<form method="get" action="recoger_datos.cgi" name="recoger_datos">
<input name="urgente" value="0" type="checkbox">
<input value="" maxlength="34" size="35" name="nombre" required>
<input value="" maxlength="34" size="35" name="apellidos" required>
<input value="" maxlength="34" size="35" name="direccion" required>
<input value="" maxlength="3" size="35" name="edad" required>
<input value="" maxlength="9" size="35" name="numerotelefono" required>
<input maxlength="200" size="80" name="TextoSupervisor">
<input name="Update" value="Solicitar atención" type="submit">
```

Mediante estas líneas se definen los campos de escritura y sus identificadores, que serán *buscados* posteriormente en el handler mencionado para decodificar la información introducida en cada campo y almacenarla.
Como se mencionó, el formulario usa método GET con la acción “recoger_datos.cgi”. Con esto, se activa el handler para que realice dicha decodificación de la información y la almacene correctamente.

Con esto queda comentada la parte de la elaboración de la página web y unas pinceladas sobre cómo envía la información a la aplicación de supervisión, donde este aspecto se explicará con mayor detalle.

Destacar que, aunque no es común, en ocasiones, si la señal de internet es débil o inestable, el programa puede fallar debido a ese hecho. Por lo que se recomienda que la conexión a internet sea razonablemente estable para usarlo, y si falla inesperadamente, reiniciarlo de nuevo.

### 3.2.2.- Aplicación de recepción y almacenamiento de avisos:

La aplicación realizada trata de emular la interfaz que podría tener una centralita para la gestión de avisos y posteriormente el envío de ayuda. Las principales características que buscábamos durante su desarrollo fueron que fuera práctica y sencilla de utilizar, ya que, el tiempo de respuesta puede ser vital ante ciertos tipos de avisos.

Parece sensato pensar que, para suplir estas dos necesidades, la pantalla FT800 de la que disponíamos, era una buena opción para resolverlo a la hora de conseguir una interfaz cómoda y útil. El programa consiste en diferentes pantallas regidas por una máquina de estados, en las que se muestran los diferentes avisos que llegan, información relativa a los avisos y la opción de actuar en consecuencia.

Los diferentes bloques de los que se compone el programa son:

- Interrupción que actúa cada vez que se rellena y envía un formulario desde la página web. Su prototipo es:

```
static char *HandlerCGI_Receptor_Datos(int32_t iIndex, int32_t i32NumParams, char *pcParam[], char *pcValue[]);
```

  Se encarga de decodificar y procesar la información introducida en la página web y almacenarla adecuadamente para hacer posible el acceso a dicha información a través de la interfaz de la aplicación desarrollada.

- Funciones relacionadas con Ethernet para fines como obtener la dirección IP de la página web, inicializar el servidor o establecer la prioridad entre interrupciones.
- Máquina de estados que rige el funcionamiento del programa y que nos permite movernos a través de las diferentes pantallas de las que dispone el programa.

- Dos timers que provocan interrupción:
    - Timer0 A → configurado periódico y con una cuenta de 1 segundo. Se encarga de habilitar la pulsación de los botones de los estados. Durante la fase de pruebas del programa, nos dimos cuenta de que al pulsar en el botón de algún estado, si en el siguiente estado había un botón en la misma posición, este automáticamente también se pulsaba provocando un comportamiento algo incómodo a la hora de usar la aplicación.

      Para ello, se recurrió a este timer con el fin de que la pulsación de dichos botones solo se habilitase pasados dos segundos desde que entramos en el estado. Al tratarse de un timer periódico, nos ahorrábamos tener que cargar de nuevo la cuenta pero por contra, debíamos ser muy cuidadosos con la habilitación y deshabilitación del timer a la vez que con el reseteo de las variables que se modificaban en la interrupción de este timer.

    - Timer1 A → configurado periódico y con una cuenta de 1 segundo. Este timer se encarga de contar el tiempo de ejecución del programa. Esta información se usará para rellenar el campo “Hora_s” de la estructura que caracteriza a cada aviso, con ello, comparando con la “hora” del sistema que aparece en la pantalla de registro, podemos hacernos una idea rápida de cuánto tiempo ha transcurrido desde el aviso anterior.

- Funciones varias para pintar las diferentes pantallas, emitir sonido o procesar el campo “Mensaje_s” de la estructura del aviso en caso de contener mensaje. Esto último se debe a que el mensaje puede contener hasta 200 caracteres (con la idea tanto de aprovechar el espacio disponible en la pantalla, así como de permitir un espacio suficiente para un mensaje aclaratorio del aviso emitido), si se saca por pantalla el mensaje sin ningún procesamiento previo, gran parte del mensaje se saldría de los límites de la pantalla y por tanto no apareciese una buena parte del mensaje enviado.

Todo aviso se compone de los mismos campos que vienen definidos por la siguiente estructura:

```
// STRUCT:
struct StructResumen {
  char  Tipo_mensaje_s[50];
  char  Hora_s[50];
  char  Nombre_s[50];
  char  Apellidos_s[50];
  char  Direccion_s[50];
  char  Edad_s[50];
  char  Telefono_s[50];
  char  Mensaje_s[200];
};
```

Una vez definida la estructura con los campos de información necesarios para cada aviso, se declaran dos arrays de structs para almacenar los diferentes conjuntos de avisos de los dos tipos que se consideran, urgente y no urgente:

```
// ARRAY OF STRUCTS:
//Array de struct de tamaño 20 para los avisos urgentes
struct StructResumen RegistroUrgente[20];

//Array de struct de tamaño 20 para los avisos no urgentes
struct StructResumen RegistroNoUrgente[20];
```

Para los ejemplos que llevamos a cabo en este prototipo, con una dimensión de cada array de 20 estructuras es suficiente, pero destacar de nuevo que con la memoria disponible en el microcontrolador se podría aumentar esta dimensión a 50 registros, por ejemplo, para cada tipo de aviso sin problemas.

Cada vez que se envía un formulario desde la página web, el manejador de interrupción rellena una estructura de este tipo con la información que le llega, además del tiempo de ejecución que obtiene del propio programa (calculado en la interrupción del Timer 1A).

Pasamos ya a comentar el código haciendo especial hincapié en las partes más importantes:

-Debido a la larga lista de variables, se ha decidido omitir esa parte en esta sección de la memoria. En el propio código se incluyen comentarios que explican la utilidad de cada una.

Adicionalmente, se omitirá cómo se configuran los timers, pantalla, etc…, ya que consideramos que no aporta nada nuevo.

Lo que sí vamos a hacer es comentar en este mismo punto las utilidades de los timers:

- Timer 0 → se encarga de incrementar en una unidad distintas variables que se utilizan para realizar animaciones o activar la pulsación de botones del programa. Además, conmuta entre ‘0’ y ‘1’ el valor de una variable llama “flag_estado_alerta”. Como ya se ha comentado, es un timer periódico con cuenta de 1 segundo. Su código es el siguiente:

```
void IntTimer0(void)
{
  TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

  contador_estado2 = contador_estado2 + 1;
  contador_borrando = contador_borrando + 1;
  contador_estado8 = contador_estado8 + 1;
  activa_estado_9 = activa_estado_9 + 1;
  activa_estado_6 = activa_estado_6 + 1;
  activa_estado_7 = activa_estado_7 + 1;
  activa_estado_1 = activa_estado_1 + 1;

  flag_estado_alerta = !flag_estado_alerta;
}
```

- Timer 1 → como ya se ha comentado, es un timer periódico con cuenta 1 segundo que se encarga de contar el tiempo de ejecución del programa, por ello, empieza a contar desde que se configura al comienzo del main. Su código es el siguiente:

```
void IntTimer1(void)
{
  TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

  cont_circulos = cont_circulos + 1;

  if (cont_circulos == 4) cont_circulos = 0;

  segundos++;
 if(segundos == 60)
 {
    minutos++;
    segundos = 0;
 }
 if(minutos == 60)
 {
    horas++;
    minutos = 0;
 }
 if(horas == 100)
 {
    horas = 0;
 }
```

```
  sprintf(cadena_hora, "%02d:%02d:%02d",horas, minutos, segundos);
}
```

“cadena_hora” será lo que se usa para rellenar el campo “Hora_s” de las estructuras.

- Sí que es mucho más interesante comentar cómo se ha configurado y habilitado el uso del periférico Ethernet, ya que aunque ciertas funciones se extraen de librerías externas, principalmente propiedad de Texas Instruments, sí que hay que tener claro cómo realizar los pasos de configuración y qué funcionalidad tienen ciertas funciones para que la página web que se tiene embebida en la memoria Flash del microcontrolador se comunique adecuadamente con el mismo, con el funcionamiento descrito en el apartado anterior de la memoria.
Si vamos paso a paso:
    - Se preparan unos defines para el reloj del sistema:

```
#define SYSTICKHZ               100
#define SYSTICKMS               (1000 / SYSTICKHZ)
```

    - Se definen las prioridades de interrupción siendo que los 3 bits superiores de estos valores son significativos con los valores más bajos indicando mayor prioridad de interrupción:

```
#define SYSTICK_INT_PRIORITY    0x80
#define ETHERNET_INT_PRIORITY   0xC0
```

    - Se hace referencia a la función que se encarga de inicializar el servidor httpd (HTTP Apache):

```
extern void httpd_init(void);
```

    - Se declara el prototipo del handler CGI que como ya hemos mencionado, al pulsar el botón “Solicitar Atención” (en la página web), recibe, decodifica y almacena la información procedente del formulario rellenado en la página web.

```
static char *HandlerCGI_Receptor_Datos(int32_t iIndex, int32_t i32NumParams, char *pcParam[], char
*pcValue[]);
```

    - Registramos ahora dicho handler mostrado anteriormente como el manejador CGI, informando al servidor httpd de que si se activa la URI: “/recoger_datos.cgi”

      Esto ocurre al rellenar el formulario y pulsar “Solicitar Atención” (en la página web), será dicho manejador “Handler_CGI_Receptor_Datos” el encargado de actuar como se mencionaba.

```
static const tCGI Config_URIs[] =
{
   { "/recoger_datos.cgi", (tCGIHandler)HandlerCGI_Receptor_Datos },
};
```

- A continuación, se presenta un define para enumerar la cantidad de URIS CGI individuales configuradas, en este caso sólo una: “/recoger_datos.cgi”

```
#define Numero_de_URIS_configuradas    (sizeof(Config_URIs) / sizeof(tCGI))
```

- Se define la respuesta a devolver por parte del Handler definido hacia la página web una vez el dicho handler ha terminado de realizar su funcionalidad. Esta respuesta consiste en la URI de la página a cargar y coincide con la propia página que permite rellenar el formulario, con lo cual, el único efecto de esta devolución es la de recargar la página vaciando el formulario, permitiendo emitir nuevos avisos en nuestro prototipo.

```
#define Respuesta_Handler_CGI    "/solicitud_cgi.ssi"
```

- Se define un tiempo de expiración en caso de que DHCP no sea capaz de asignar una IP antes de dicho tiempo:

```
#ifndef DHCP_EXPIRE_TIMER_SECS
#define DHCP_EXPIRE_TIMER_SECS  45
#endif
```

- Se define una variable donde recogeremos la IP actual, donde se aloja la página web (servidor httpd):

```
uint32_t Direccion_IP_actual;
```

- Aunque es ampliamente usada, también se usa para Ethernet, definimos la variable Reloj, donde guardaremos la frecuencia asignada al reloj:

```
uint32_t Reloj;
```

- Se declara el handler para la interrupción de SysTick, este handler llama a la función lwIPTimer con el parámetro “SYSTICKMS”, definido anteriormente, y que corresponde en dicha función al tiempo incremental para las interrupciones periódicas de eventos temporales de la pila TCP/IP de lwIP:

```
void SysTickIntHandler(void)
{
   lwIPTimer(SYSTICKMS);
}
```

- Se define una función que por un lado convierte la dirección IP en una string estándar (mediante la función ‘usprintf’, la cual es práctimente igual a ‘sprintf’) y además cuando la llamemos desde el estado 10 de la FSM de la aplicación, permitirá mostrar por pantalla la dirección IP donde se aloja el servidor httpd

con la página hosteada por el microcontrolador, para que sea fácilmente accesible en las pruebas de este prototipo del sistema:

```
void DisplayIPAddress(uint32_t Addr)
{
   char pcBuf[16];
   // Convertir la dirección IP en una string.
   usprintf(pcBuf, "%d.%d.%d.%d", Addr & 0xff, (Addr >> 8) & 0xff,
         (Addr >> 16) & 0xff, (Addr >> 24) & 0xff);

   // Mostrar la string (dirección IP) por la UART.
   if(estado != 10) UARTprintf(pcBuf);

   //Se muestra por pantalla la dirección IP a introducir en el navegador para usar la Web
   if (estado == 10){
      ComColor(0,0,0);
      ComTXT(HSIZE/2, VSIZE/2-40, 23, OPT_CENTER,"Introduzca la direccion IP en su navegador:");
      ComColor(255,255,255);
      ComTXT(HSIZE/2, VSIZE/2-10, 23, OPT_CENTER, pcBuf);
   }
}
```

- Finalmente, para terminar con los aspectos de Ethernet (fuera del main), se declara una función necesaria para visualizar el proceso mediante el cual la librería lwIP obtiene la dirección IP mediante DHCP:

```
void lwIPHostTimerHandler(void)
{
   uint32_t Nueva_Direccion_IP;

   // Obtener la dirección IP.
   Nueva_Direccion_IP = lwIPLocalIPAddrGet();

   // Comprobar si la dirección IP ha cambiado.
   if(Nueva_Direccion_IP != Direccion_IP_actual)
   {
      // Comprobar si hay una dirección IP asignada.
      if(Nueva_Direccion_IP == 0xffffffff)
      {
         // Indicar que aún no hay enlace de conexión.
         UARTprintf("Esperando enlace\n");
      }
      else if(Nueva_Direccion_IP == 0)
      {
         // No hay dirección IP aún, indicar que DHCP está buscando.
         UARTprintf("Esperando dirección IP\n");
      }
      else
      {
         // Mostrar la nueva dirección IP obtenida.
         UARTprintf("Dirección IP: ");
         DisplayIPAddress(Nueva_Direccion_IP);
         UARTprintf("\n");
         UARTprintf("Abra un buscador e inserte la dirección IP\n");
         direccion_obtenida = 1; // Se activa esta variable para avanzar del modo "Conectando..." de la aplicación.
      }
```

```
        // Guardar la nueva dirección IP.
        Direccion_IP_actual = Nueva_Direccion_IP;
    }

    // Si no hay una nueva dirección IP:
    if((Nueva_Direccion_IP == 0) || (Nueva_Direccion_IP == 0xffffffff))
    {
        // No hacer nada y seguir esperando.
    }
}
```

Para terminar con la parte de configuración de los aspectos de Ethernet, falta mencionar ciertos aspectos que se llevan a cabo **dentro del main()**:

- Se definen variables para configurar la dirección MAC del hardware del controlador Ethernet de la placa:

```
uint32_t User0, User1;
uint8_t pui8MACArray[8];
```

- Se necesita que el oscilador principal esté activo, ya que esto es necesario para el PHY (Physical Layer / Circuito integrado) El parámetro "SYSCTL_MOSC_HIGHFREQ" se utiliza cuando la frecuencia del cristal es >= 10 MHz:

```
SysCtlMOSCConfigSet(SYSCTL_MOSC_HIGHFREQ);
```

- Se configura SysTick para interrupciones periódicas:

```
MAP_SysTickPeriodSet(Reloj / SYSTICKHZ);
MAP_SysTickEnable();
MAP_SysTickIntEnable();
```

- Se configura la dirección MAC hardware para que el controlador Ethernet filtre paquetes entrantes. La dirección MAC se guarda en los registros USER0 y USER1.

```
MAP_FlashUserGet(&User0, &User1);
if((User0 == 0xffffffff) || (User1 == 0xffffffff))
{
    // Hacer saber que no hay dirección MAC:
    UARTprintf("Dirección MAC no programada\n");

    while(1)
    {
    }
}
```

- Se convierte la dirección MAC separada en fragmentos 24/24 procedente de la Non Volatile RAM, en una dirección MAC separada en fragmentos 32/16, que es el formato necesario para programar los registros hardware. Tras ello se programará la dirección MAC en los registros del controlador Ethernet:

```
pui8MACArray[0] = ((User0 >>  0) & 0xff);
pui8MACArray[1] = ((User0 >>  8) & 0xff);
pui8MACArray[2] = ((User0 >> 16) & 0xff);
pui8MACArray[3] = ((User1 >>  0) & 0xff);
pui8MACArray[4] = ((User1 >>  8) & 0xff);
pui8MACArray[5] = ((User1 >> 16) & 0xff);
```

- Se inicializa la librería lwIP, configurándola para que use DHCP:

```
lwIPInit(Reloj, pui8MACArray, 0, 0, 0, IPADDR_USE_DHCP);
```

- Se hace setup del servicio de Locator para el dispositivo (el microcontrolador):

```
LocatorInit();
LocatorMACAddrSet(pui8MACArray);
LocatorAppTitleSet("EK-TM4C1294XL Servicio_Atencion");
```

- Se inicializa el servidor httpd:

```
httpd_init();
```

- Establecer la prioridad de las interrupciones:

```
MAP_IntPrioritySet(INT_EMAC0, ETHERNET_INT_PRIORITY);
MAP_IntPrioritySet(FAULT_SYSTICK, SYSTICK_INT_PRIORITY);
```

- Pasar nuestro CGI handler al servidor httpd con la configuración necesaria:

```
http_set_cgi_handlers(Config_URIs, Numero_de_URIS_configuradas);
```

Con esto finaliza el conjunto de instrucciones necesarias para permitir tanto el hosteo de la página web embebida en el microcontrolador en un propio servidor httpd asociado a la propia dirección IP del microcontrolador, como la configuración correcta de los handlers y procesos de interrupción Ethernet a la hora de realizar los cambios en la página web desarrollada.

-Intrerrupción del manejador: como ya se ha comentado, se encarga de pasar la información del formulario (de la página web) a un struct del programa. Primeramente, verá si el mensaje es de tipo urgente o no. Una vez hecho y conocido el número de mensajes que ya existen de cada tipo, almacena en una componente del array de mensajes urgentes o no urgentes según corresponda, la información del mensaje en el índice más pequeño del array que esté vacío. A continuación, se muestra el código que implementa lo comentado:

```
static char *HandlerCGI_Receptor_Datos(int32_t iIndex, int32_t i32NumParams, char *pcParam[], char
*pcValue[]){

   long lStringParam;
   urgente = FindCGIParameter("urgente", pcParam, i32NumParams);

   if (urgente == -1) strcpy(cadena_tipo_mensaje, cadena_no_urgente);
   else if (urgente == 0) strcpy(cadena_tipo_mensaje, cadena_urgente);

   lStringParam = FindCGIParameter("nombre", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Nombre, 50);

   lStringParam = FindCGIParameter("apellidos", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Apellidos, 50);

   lStringParam = FindCGIParameter("direccion", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Direccion, 50);

   lStringParam = FindCGIParameter("edad", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Edad, 50);

   lStringParam = FindCGIParameter("numerotelefono", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Telefono, 50);

   lStringParam = FindCGIParameter("TextoSupervisor", pcParam, i32NumParams);
   DecodeFormString(pcValue[lStringParam], Mensaje, 200);

   if (urgente == 0){
      strcpy(RegistroUrgente[mensajes_urgentes].Tipo_mensaje_s, cadena_tipo_mensaje);
      strcpy(RegistroUrgente[mensajes_urgentes].Hora_s, cadena_hora);
      strcpy(RegistroUrgente[mensajes_urgentes].Nombre_s, Nombre);
      strcpy(RegistroUrgente[mensajes_urgentes].Apellidos_s, Apellidos);
      strcpy(RegistroUrgente[mensajes_urgentes].Direccion_s, Direccion);
      strcpy(RegistroUrgente[mensajes_urgentes].Edad_s, Edad);
      strcpy(RegistroUrgente[mensajes_urgentes].Telefono_s, Telefono);
      strcpy(RegistroUrgente[mensajes_urgentes].Mensaje_s, Mensaje);
      mensajes_urgentes = mensajes_urgentes + 1;
   }
   else{

      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Tipo_mensaje_s, cadena_tipo_mensaje);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Hora_s, cadena_hora);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Nombre_s, Nombre);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Apellidos_s, Apellidos);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Direccion_s, Direccion);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Edad_s, Edad);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Telefono_s, Telefono);
      strcpy(RegistroNoUrgente[mensajes_no_urgentes].Mensaje_s, Mensaje);
```

```
        mensajes_no_urgentes = mensajes_no_urgentes + 1;
    }

    cambio_estado = 1;

    return(Respuesta_Handler_CGI);
}
```

Podemos ver además, como este Handler devuelve al servidor httpd una "*Respuesta_Handler_CGI*", que equivale a "*/solicitud_cgi.ssi*" , un archivo que define la pestaña de la web para rellenar el formulario  (URI de la página a cargar).
Esto provoca que la página web se recargue en la pestaña del formulario, con este vacío, tras haber recogido los datos introducidos, quedando lista para volver a ser usada, emitiendo nuevos avisos si se desea.

Como se puede observar, una vez guardado el aviso, se incrementa el número de mensajes de ese tipo y se activa "cambio_estado", variable que provocará un cambio de estado que posteriormente comentaremos.

-La máquina de estados (FSM) del programa consta de 11 estados en la imagen que se adjunta se muestra un esquema global de la FSM:

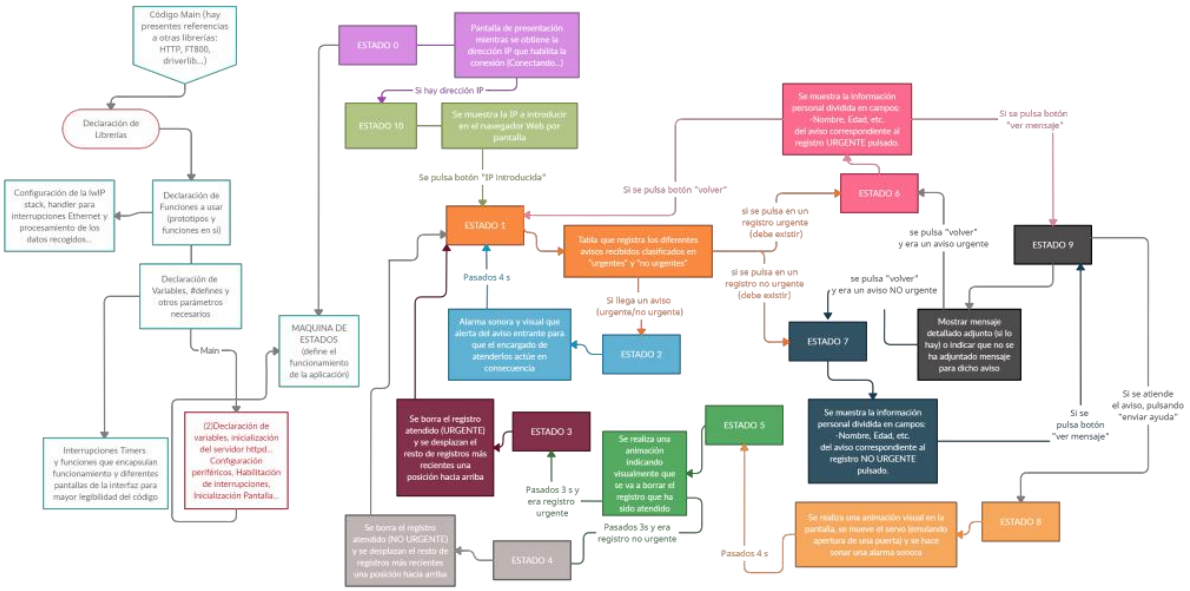



***Figura 5:*** *Esquema global de la máquina de estados y las condiciones que provocan los cambios entre estados*

- Estado 0 → muestra una pantalla de carga en la que se permanece hasta que el protocolo DHCP nos proporciona una dirección IP para nuestra página web. A continuación, se muestra el código que implementa lo comentado:

```
case 0:
        while (direccion_obtenida == 0){
           PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Min_pos);
           Pantalla_inicial();
           Dibuja();
        }
        estado = 10;
        break;
```

  Como podemos observar, la variable “direccion_obtenida” es la que nos bloquea en este estado hasta obtener una dirección. Por ello, es lógico pensar que dicha variable se modifique en la función donde se obtiene la dirección IP.

- Estado 10 → pantalla en la que se muestra por pantalla la dirección obtenida del protocolo DHCP. Junto a esto, se ha colocado un botón para que el usuario haga saber al programa que ha abierto un buscador y ha introducido la dirección que está viendo por pantalla.
  Recalcar de nuevo que esto de proporcionar la IP en la aplicación e introducir la IP en el navegador del PC conectado al microcontrolador por Ethernet es necesario debido a las limitaciones del prototipo realizado, es decir, idealmente, la idea sería que se tuviese la página Web en un servidor de internet normal y el microcontrolador con la aplicación pudiese recibir avisos remotos desde una página web en un servidor remoto.
  Sin embargo, como ya se mencionaba, para este prototipo realizamos las pruebas con la página web abierta en el PC que se conecta al microcontrolador por Ethernet y se interactúa con ambos sistemas de esta manera.

```
case 10:
        Nueva_pantalla(51,222,204);
        DisplayIPAddress(Direccion_IP_actual);
        if(Boton_IP_introducida){
           estado = 1;
        }
        Dibuja();
        break;
```

  Una vez pulsado el botón, pasamos al estado 1, probablemente el más importante del programa.

- Estado 1 → pantalla con una tabla que muestra los 5 avisos más antiguos tanto del tipo urgente como no urgente. En esta tabla adicionalmente a cada aviso, le acompaña la hora (tiempo de ejecución del programa) a la que fue enviado el aviso:

```
case 1:
        TimerEnable(TIMER0_BASE, TIMER_A);
        borrado_urgente = 0;
        borrado_no_urgente = 0;
        activa_estado_6 = 0;
        activa_estado_7 = 0;
        Pantalla_registro();

        Dibuja();

        Reset_Cadenas_Mensaje();

        Lee_pantalla();

        Comprueba_Pulsacion_Registros_Urgentes();

        Comprueba_Pulsacion_Registros_No_Urgentes();

        if(activa_estado_1 == 2)  TimerDisable(TIMER0_BASE, TIMER_A);

        if (cambio_estado == 1){
           contador_estado2 = 0;
           estado = 2;
        }
        break;
```

Con “Reset_Cadenas_Mensaje()” vaciamos seis cadenas de caracteres que serán útiles en el estado 9.

Recapitulando lo que se comentó en el manejador de interrupción, la variable “cambio_estado” provoca que pasemos al estado 2.

La otra forma de cambiar de estado consiste en pulsar sobre alguno de los avisos de la tabla (debe existir aviso donde se pulsa, si no lo hay, no hace nada). En caso de pulsarse sobre un mensaje urgente pasaremos al estado 6, para los no urgentes la transición será hacia el estado 7.

Las funciones “Comprueba_Pulsacion_Registros_Urgentes()” y “Comprueba_Pulsacion_Registros_No_Urgentes()” se encargan de comprobar si se pulsa en alguno de los avisos que se tiene para provocar el cambio al estado 6 y 7 respectivamente. Además, se le da valor a “registro” que será de gran utilidad en estos dos estados y en el 9. A continuación, se muestra el código que implementa lo comentado:

```
void Comprueba_Pulsacion_Registros_Urgentes(void){
```

```
   if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/3+3 && POSY <= VSIZE/2-11 &&
mensajes_urgentes > 0){
      registro = 1;
      IndiceUrgente = 0;
      estado = 6;
   }
   if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/2-11 && POSY <= VSIZE/2+21 &&
mensajes_urgentes > 1){
      registro = 2;
      IndiceUrgente = 1;
      estado = 6;
   }
   if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/2+21 && POSY <= 7*VSIZE/10-1 &&
mensajes_urgentes > 2){
      registro = 3;
      IndiceUrgente = 2;
      estado = 6;
   }
   if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > 7*VSIZE/10-1 && POSY <= 8*VSIZE/10+3
&& mensajes_urgentes > 3){
      registro = 4;
      IndiceUrgente = 3;
      estado = 6;
   }
   if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > 8*VSIZE/10+3 && POSY <= 8*VSIZE/10+36
&& mensajes_urgentes > 4 && activa_estado_1 >= 2){
      registro = 5;
      IndiceUrgente = 4;
      estado = 6;
   }
}
```

También se modifica “IndiceUrgente” para saber a qué componente del array de estructuras de mensajes urgentes se corresponde el mensaje pulsado. Es decir el aviso pulsado se corresponde con RegistroUrgente[IndiceUrgente].

Cabe destacar que en el último *IF* se ha debido introducir una condición adicional con “activa_estado_1” ya que, si se pulsaba el botón de “Volver” que se comentará en los estados 6 y 7, automáticamente el programa interpretaba que se había pulsado esa región de la pantalla lo que nos llevaría al estado 6 de manera indeseada. Es por ello que, “activa_estado_1” se incrementa en una unidad en la interrupción del timer 0. Pasados 2 segundos desde que entramos en este estado, se habilita la pulsación de dicha región.

En cuanto a “Comprueba_Pulsacion_Registros_No_Urgentes()” la idea es exactamente la misma pero adaptada a sus coordenadas y a sus variables análogas. A continuación, se muestra el código:

```
void Comprueba_Pulsacion_Registros_No_Urgentes(void){
   if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/3+3 && POSY <= VSIZE/2-11 &&
mensajes_no_urgentes > 0){
      registro = 1;
```

```
        IndiceNoUrgente = 0;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/2-11 && POSY <= VSIZE/2+21 &&
mensajes_no_urgentes > 1){
        registro = 2;
        IndiceNoUrgente = 1;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/2+21 && POSY <= 7*VSIZE/10-1
&& mensajes_no_urgentes > 2){
        registro = 3;
        IndiceNoUrgente = 2;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > 7*VSIZE/10-1 && POSY <= 8*VSIZE/10+3
&& mensajes_no_urgentes > 3){
        registro = 4;
        IndiceNoUrgente = 3;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > 8*VSIZE/10+3 && POSY <=
8*VSIZE/10+36 && mensajes_no_urgentes > 4){
        registro = 5;
        IndiceNoUrgente = 4;
        estado = 7;
    }
}
```

- Estado 2 → consiste en una pantalla de alerta en la que el fondo de la pantalla alterna entre dos colores llamativos, a la vez que emite sonido para llamar la atención de la persona que se está encargando de gestionar los avisos en la centralita. A continuación, se muestra el código que implementa lo comentado:

```
case 2:
        TimerEnable(TIMER0_BASE, TIMER_A);
        cambio_estado = 0;
        Funcion_nota1();
        if (flag_estado_alerta == 0){
            Pantalla_alerta1();
            Dibuja();
        }
        if (flag_estado_alerta == 1){
            Pantalla_alerta2();
            Dibuja();
        }
        if (contador_estado2 == 4){
            flag_estado_alerta = 0;
            FinNota();
            TimerDisable(TIMER0_BASE, TIMER_A);
            estado = 1;
        }
        break;
```

“flag_estado_alerta” conmuta su valor en la interrupción del timer 0 a la vez que “contador_estado2” se incrementa en una unidad en el mismo timer. Pasados 4 segundos desde que entramos en el estado, volvemos a la pantalla de registro.

“Funcion_nota1()” es una función que se usa para emitir un sonido de *BEEP.* Su código es el siguiente:

```
void Funcion_nota1(){
   VolNota(100);
   TocaNota(S_BEEP, N_SI);
}
```

- Estado 6 → estado en el que se puede observar toda la información del aviso urgente seleccionado a excepción del campo “Mensaje_s” que es el mensaje del aviso. A continuación, se muestra el código que implementa lo comentado:

```
case 6:
        ver_mensaje_urgente = 0;
        volver_mensaje_urgente = 0;
        Pantalla_detalle();
        MuestraDetalles(registro,1);

        if(Boton_volver && activa_estado_6 >= 2){
           activa_estado_1 = 0;
           estado = 1;
        }

        if(Boton_ver_mensaje && activa_estado_6 >= 2){
           borrado_urgente = 1;
           volver_mensaje_urgente = 1;
           ver_mensaje_urgente = 1;
           flag_primera_linea = 0;
           flag_segunda_linea = 0;
           flag_tercera_linea = 0;
           flag_cuarta_linea = 0;
           flag_quinta_linea = 0;
           i = 0;
           k = 0;
           activa_estado_9 = 0;
           terminador = 0;
           estado = 9;
        }

        if(activa_estado_6 == 2) TimerDisable(TIMER0_BASE, TIMER_A);

        Dibuja();
        break;
```

En esta pantalla se da la opción de volver al registro (estado 1) pulsando el botón “Volver” o de ver el contenido del mensaje del aviso (estado 9) en caso de que contenga al pulsar “Ver_mensaje”.

Aquí se vuelve a activar los botones pasados dos segundos desde que entramos en el estado para evitar la problemática anteriormente comentada haciendo uso de “activa_estado_6” que también se incrementa en una unidad en la interrupción del timer 0.

La función “MuestraDetalles()” es la que se encarga de sacar por pantalla los datos correspondientes al aviso que hemos pulsado en el estado 1. Para ello, hace uso de “registro”. A continuación, se muestra el código que implementa lo comentado:

```
void MuestraDetalles(int registro, int urgente)
{
   if(urgente == 1){
      ComColor(0,0,0);
      ComTXT(HSIZE/20, 2*VSIZE/9, 22, OPT_CENTERY, "Tipo de mensaje: ");
      ComTXT(HSIZE/20, 3*VSIZE/9, 22, OPT_CENTERY, "Hora: ");
      ComTXT(HSIZE/20, 4*VSIZE/9, 22, OPT_CENTERY, "Nombre: ");
      ComTXT(HSIZE/20, 5*VSIZE/9, 22, OPT_CENTERY, "Apellidos: ");
      ComTXT(HSIZE/20, 6*VSIZE/9, 22, OPT_CENTERY, "Direccion: ");
      ComTXT(HSIZE/20, 7*VSIZE/9, 22, OPT_CENTERY, "Edad: ");
      ComTXT(HSIZE/2, 7*VSIZE/9, 22, OPT_CENTERY, "Telefono: ");
      ComColor(255,255,255);
      ComTXT(HSIZE/20+135, 2*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-
1].Tipo_mensaje_s);
      ComTXT(HSIZE/20+45, 3*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Hora_s);
      ComTXT(HSIZE/20+67, 4*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Nombre_s);
      ComTXT(HSIZE/20+73, 5*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Apellidos_s);
      ComTXT(HSIZE/20+75, 6*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Direccion_s);
      ComTXT(HSIZE/20+47, 7*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Edad_s);
      ComTXT(HSIZE/2+72, 7*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Telefono_s);
   }
   else{
      ComColor(0,0,0);
      ComTXT(HSIZE/20, 2*VSIZE/9, 22, OPT_CENTERY, "Tipo de mensaje: ");
      ComTXT(HSIZE/20, 3*VSIZE/9, 22, OPT_CENTERY, "Hora: ");
      ComTXT(HSIZE/20, 4*VSIZE/9, 22, OPT_CENTERY, "Nombre: ");
      ComTXT(HSIZE/20, 5*VSIZE/9, 22, OPT_CENTERY, "Apellidos: ");
      ComTXT(HSIZE/20, 6*VSIZE/9, 22, OPT_CENTERY, "Direccion: ");
      ComTXT(HSIZE/20, 7*VSIZE/9, 22, OPT_CENTERY, "Edad: ");
      ComTXT(HSIZE/2, 7*VSIZE/9, 22, OPT_CENTERY, "Telefono: ");
      ComColor(255,255,255);
      ComTXT(HSIZE/20+135, 2*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-
1].Tipo_mensaje_s);
      ComTXT(HSIZE/20+45, 3*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Hora_s);
      ComTXT(HSIZE/20+67, 4*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-
1].Nombre_s);
      ComTXT(HSIZE/20+73, 5*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-
1].Apellidos_s);
      ComTXT(HSIZE/20+75, 6*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-
1].Direccion_s);
      ComTXT(HSIZE/20+47, 7*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Edad_s);
      ComTXT(HSIZE/2+72, 7*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-
1].Telefono_s);
   }
}
```

Además de “registro”, sabiendo que estamos en el estado 6 que se corresponde con los datos de los mensajes urgentes, le damos valor ‘1’ al parámetro “urgente” de la función.

- Estado 7 → la idea es exactamente la misma que el estado 6 adaptando las variables a las de mensajes no urgentes. A continuación, se muestra el código:

```
case 7:
        TimerEnable(TIMER0_BASE, TIMER_A);
        ver_mensaje_no_urgente = 0;
        volver_mensaje_no_urgente = 0;
        Pantalla_detalle();
        MuestraDetalles(registro,0);

        if(Boton_volver && activa_estado_7 >= 2){
           activa_estado_1 = 0;
           estado = 1;
        }

        if(Boton_ver_mensaje && activa_estado_7 >= 2){
           borrado_no_urgente = 1;
           volver_mensaje_no_urgente = 1;
           ver_mensaje_no_urgente = 1;
           flag_primera_linea = 0;
           flag_segunda_linea = 0;
           flag_tercera_linea = 0;
           flag_cuarta_linea = 0;
           flag_quinta_linea = 0;
           i = 0;
           k = 0;
           activa_estado_9 = 0;
           terminador = 0;
           estado = 9;
        }

        if(activa_estado_7 == 2) TimerDisable(TIMER0_BASE, TIMER_A);

        Dibuja();
        break;
```

Como se observa en este caso, el parámetro “urgente” en “MuestraDetalles()” toma valor ‘0’.

- Estado 9 → es el estado en el que se ve el contenido del mensaje del aviso. En caso de que no contenga ningún mensaje, también se indica. El código es el siguiente:

```
case 9:
        TimerEnable(TIMER0_BASE, TIMER_A);
        Pantalla_Mensaje();


        MuestraMensaje(ver_mensaje_urgente, ver_mensaje_no_urgente);

        if(Boton_volver && activa_estado_9 >= 2){
           if (volver_mensaje_urgente == 1){
              estado = 6;
           }
           else if (volver_mensaje_no_urgente == 1) {
              estado = 7;
           }
        }

        if(Boton_enviar_ayuda && activa_estado_9 >= 2){
           contador_estado8 = 0;
           estado = 8;
        }

        if(activa_estado_9 == 2){
           activa_estado_6 = 0;
           activa_estado_7 = 0;
           TimerDisable(TIMER0_BASE, TIMER_A);
        }

        Dibuja();
        break;
```

Con el botón “Volver” vamos de nuevo al estado 6/7 según el tipo de aviso donde se nos mostraban los detalles del mensaje. Para ello se usa la información contenida en “volver_mensaje_urgente” y “volver_mensaje_no_urgente” a los que se le da valor igual a ‘1’ en el estado 6 para el tipo urgente y 7 para el no urgente.

En cuanto al botón “Enviar_ayuda” nos llevaría al estado 8. En este estado al igual que ocurre en el resto de estado con botones, se aplica el mismo mecanismo para no activar los botones hasta que hayamos pasado dos segundos en el estado.

La función clave de este estado es “MuestraMensaje()” a la que se le pasa “ver_mensaje_urgente” a ‘1’ si veníamos del estado 6 o “ver_mensaje_no_urgente” a ‘1’ si veníamos del estado 7. Con la información de estas variables y “registro” se guarda en “cadena_mensaje” el campo “Mensaje_s” correspondiente al aviso que estamos visualizando. El funcionamiento es el siguiente:

```
void MuestraMensaje(int ver_urgente, int ver_no_urgente){

   if(ver_urgente == 1){
      strcpy(cadena_mensaje, RegistroUrgente[registro-1].Mensaje_s);
   }

   if(ver_no_urgente == 1){
      strcpy(cadena_mensaje, RegistroNoUrgente[registro-1].Mensaje_s);
   }

   if ((cadena_mensaje[0]>='a'&& cadena_mensaje[0]<='z') || (cadena_mensaje[0]>='A' &&
cadena_mensaje[0]<='Z')){
      while (i <= 199){
         if (cadena_mensaje[i] == '\0' && terminador == 0) {
            terminador = i;
            break;
         }
         i = i + 1;
      }

      while(k <= terminador){

         /************************************************** RENGLON 1
**********************************************************/
         if(terminador >= 0 && terminador <= 40){
            for (j = 0; j<=terminador; j++){
               cadena_primera_linea[j] = cadena_mensaje[j];
            }
         }
         else{
            if(k >= 40 && cadena_mensaje[k] == ' ' && flag_primera_linea == 0){
               for (j = 0; j<=k; j++){
                  cadena_primera_linea[j] = cadena_mensaje[j];
               }
               caracter = k;
               flag_primera_linea = 1;
            }
         }

         /************************************************** RENGLON 2
**********************************************************/
         if(flag_primera_linea == 1){
            if(terminador > 40 && terminador <= 80){
               for (j = caracter+1; j<=terminador; j++){
                  cadena_segunda_linea[j-caracter-1] = cadena_mensaje[j];
               }
            }
            else{
               if(k >= 80 && cadena_mensaje[k] == ' ' && flag_segunda_linea == 0){
                  for (j = caracter+1; j<=k; j++){
                     cadena_segunda_linea[j-caracter-1] = cadena_mensaje[j];
                  }
                  caracter = k;
                  flag_segunda_linea = 1;
               }
            }
         }
```

```
        /************************************************** RENGLON 3
**********************************************************/
        if(flag_segunda_linea == 1){
          if(terminador > 80 && terminador <= 120){
            for (j = caracter+1; j<=terminador; j++){
              cadena_tercera_linea[j-caracter-1] = cadena_mensaje[j];
            }
          }
          else{
            if(k >= 120 && cadena_mensaje[k] == ' ' && flag_tercera_linea == 0){
              for (j = caracter+1; j<=k; j++){
                cadena_tercera_linea[j-caracter-1] = cadena_mensaje[j];
              }
              caracter = k;
              flag_tercera_linea = 1;
            }
          }
        }

        /************************************************** RENGLON 4
**********************************************************/
        if(flag_tercera_linea == 1){
          if(terminador > 120 && terminador <= 160){
            for (j = caracter+1; j<=terminador; j++){
              cadena_cuarta_linea[j-caracter-1] = cadena_mensaje[j];
            }
          }
          else{
            if(k >= 160 && cadena_mensaje[k] == ' ' && flag_cuarta_linea == 0){
              for (j = caracter+1; j<=k; j++){
                cadena_cuarta_linea[j-caracter-1] = cadena_mensaje[j];
              }
              caracter = k;
              flag_cuarta_linea = 1;
            }
          }
        }

        /************************************************** RENGLON 5
**********************************************************/
        if(flag_cuarta_linea == 1){
          for (j = caracter+1; j<=terminador; j++){
            cadena_quinta_linea[j-caracter-1] = cadena_mensaje[j];
          }
        }
        k = k + 1;
      }
    }

    else {
      ComColor(255, 255, 255);
      ComTXT(HSIZE/2, VSIZE/2, 23, OPT_CENTER, "No hay mensaje adjunto");
    }

    ComColor(255, 255, 255);
    ComTXT(HSIZE/10, VSIZE/5+10, 23, OPT_CENTERY, cadena_primera_linea);
    ComTXT(HSIZE/10, 2*VSIZE/5-10, 23, OPT_CENTERY, cadena_segunda_linea);
    ComTXT(HSIZE/10, 3*VSIZE/5-30, 23, OPT_CENTERY, cadena_tercera_linea);
```

```
    ComTXT(HSIZE/10, 6*VSIZE/10+4, 23, OPT_CENTERY, cadena_cuarta_linea);
    ComTXT(HSIZE/10, 7*VSIZE/10+11, 23, OPT_CENTERY, cadena_quinta_linea);
}
```

Lo primero que hacemos es determinar en qué índice de la cadena está el terminador y lo guardamos en “terminador”.

Luego recorremos un bucle *while* desde el índice 0 hasta el terminador, es decir la parte de la cadena con contenido. Antes de explicar el resto del algoritmo de la función comentar que, con la pantalla usada, al escribir 40 caracteres en una línea se completa todo el ancho de la pantalla (dejando un cierto espacio de seguridad hasta el borde).

Para cada renglón se aplica lo siguiente:

- ➢ Primero miramos si el terminador se encuentra en esta línea, para ello usamos la información de que el ancho de caracteres establecidos por renglón será de 40 caracteres. Por ejemplo, para el primer renglón el terminador estará en el rango [0,40], para el segundo renglón estará en el rango (40,80] y así sucesivamente.

- ➢ En caso de no cumplirse, pues debemos almacenar unos 40 caracteres aproximadamente y almacenar el índice por el que nos hemos quedado para indicarle al siguiente renglón desde dónde debe empezar a guardar.

  Para ello, cuando superemos un ancho de 40 caracteres en ese renglón buscaremos un espacio y será hasta ahí donde guardemos. El índice del espacio + 1 será donde comience el siguiente renglón a guardar caracteres de la cadena.

Y este sería el algoritmo implementado para sacar el mensaje de un aviso correctamente por pantalla.

- Estado 8 → es el estado de envío de ayuda. Esta pantalla simplemente realiza una animación que emula el movimiento de un coche/ambulancia o cualquier otro tipo de vehículo para prestar ayuda, haciendo ver que la ayuda va en camino. Adicionalmente se mueve el servo como si se abriese una compuerta para dejar salir el vehículo. El código es el siguiente:

```
case 8:
        TimerEnable(TIMER0_BASE, TIMER_A);
        Funcion_nota2();
        if (contador_estado8 == 0){
          PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);
```

```
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 1){
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 2){
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 3){
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 4){
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Min_pos);
            FinNota();
            TimerDisable(TIMER0_BASE, TIMER_A);
            contador_borrando = 0;
            estado = 5;
        }
        break;
```

Al igual que el resto de contadores, “contador_estado8” se incrementa en una unidad en la interrupción del timer 0. Pasados 4 segundos pasamos al estado 5.

“Funcion_nota2()” es una función que se usa para emitir el sonido de un xilófono. Su código es el siguiente:

```
void Funcion_nota2(){
    VolNota(30);
    TocaNota(S_XILO, N_SI);
}
```

- Estado 5 → este estado muestra una pantalla en la que se indica que como ya se ha atendido el aviso, pues se va a borrar del sistema. Destacar que, en esta pantalla solo se realiza la animación no es en este estado donde se borra el aviso sino en el estado 3/4 según si el mensaje era urgente o no respectivamente. El código es el siguiente:

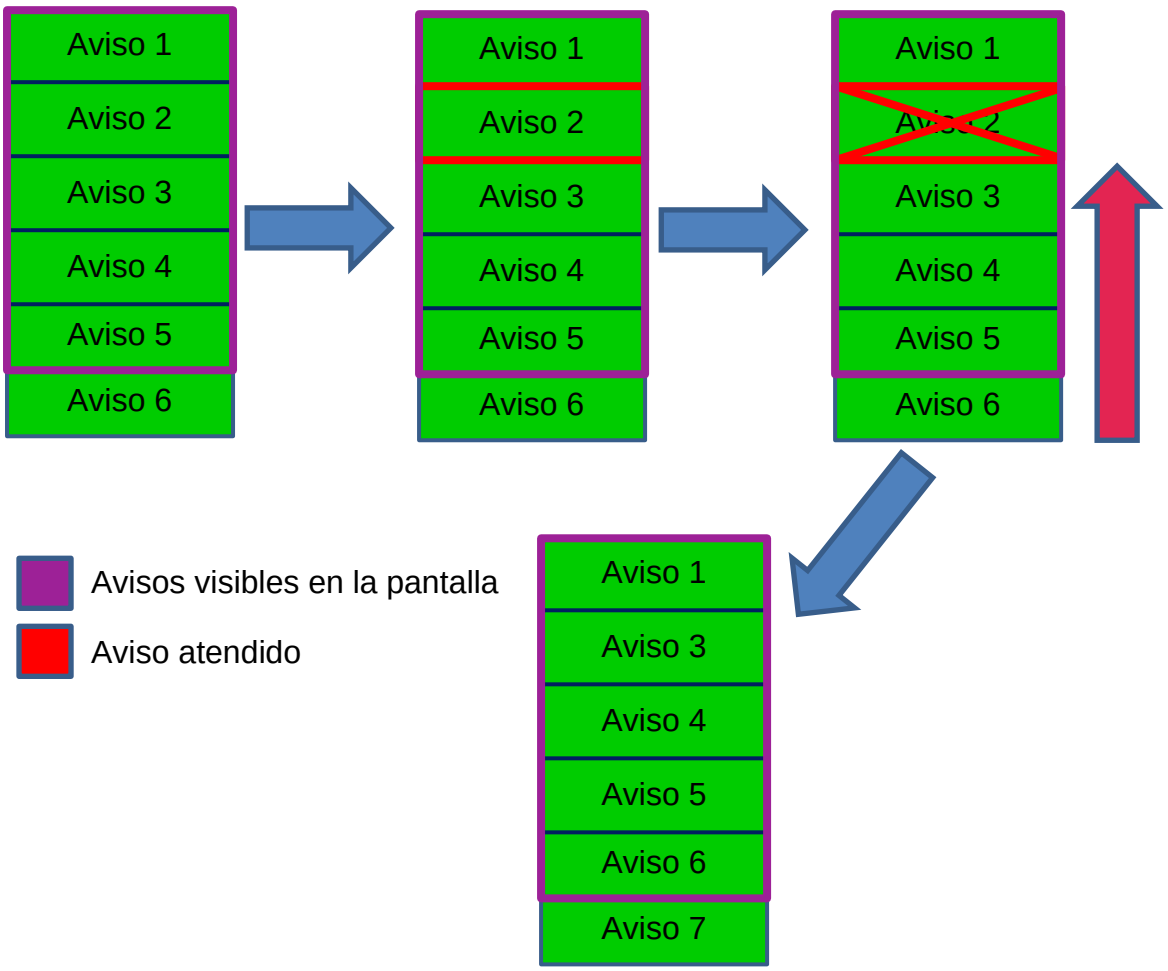
```
case 5:
        TimerEnable(TIMER0_BASE, TIMER_A);
        Dibuja();
        if (contador_borrando == 3){
            TimerDisable(TIMER0_BASE, TIMER_A);
            contador_borrando = 0;
            if(borrado_urgente == 1){
                estado = 3;
            }
            else if(borrado_no_urgente == 1){
                estado = 4;
            }
```

```
    else {}
}
break;
```

En este estado el contador es “contador_borrando” que al igual que el resto de contadores, se incrementa en una unidad en la interrupción del timer 0. Cuando pasan 3 segundos desde que entramos en el estado vemos si el mensaje que se ha atendido era urgente con “borrado_urgente” para pasar al estado 3 o era no urgente con “borrado_no_urgente” para pasar al estado 4.

- Estado 3 → se encarga de borrar el aviso urgente atendido. Para ello lo que hacemos es desplazar todos los avisos posteriores al atendido una posición hacia arriba (todos decrementan su índice en el array de estructuras en una unidad). Tras esto, se decrementa el número de mensajes urgentes almacenados y pasamos a la pantalla de registro (estado 1). A continuación, se ilustra esto:

Esquematización del proceso de borrado de un aviso en el array de estructuras



El código es el siguiente:

```
case 3:
        for(i = IndiceUrgente; i < mensajes_urgentes - 1; i++){
           strcpy(RegistroUrgente[i].Tipo_mensaje_s, RegistroUrgente[i+1].Tipo_mensaje_s);
           strcpy(RegistroUrgente[i].Nombre_s,       RegistroUrgente[i+1].Nombre_s);
           strcpy(RegistroUrgente[i].Apellidos_s,    RegistroUrgente[i+1].Apellidos_s);
           strcpy(RegistroUrgente[i].Direccion_s,    RegistroUrgente[i+1].Direccion_s);
           strcpy(RegistroUrgente[i].Mensaje_s,      RegistroUrgente[i+1].Mensaje_s);
           strcpy(RegistroUrgente[i].Hora_s,         RegistroUrgente[i+1].Hora_s);
           strcpy(RegistroUrgente[i].Edad_s,         RegistroUrgente[i+1].Edad_s);
           strcpy(RegistroUrgente[i].Telefono_s,     RegistroUrgente[i+1].Telefono_s);
        }

        if(mensajes_urgentes <= 5){
           memset(RegistroUrgente[mensajes_urgentes-1].Tipo_mensaje_s, 0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Nombre_s,       0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Apellidos_s,    0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Direccion_s,    0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Mensaje_s,      0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Hora_s,         0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Edad_s,         0, 50);
           memset(RegistroUrgente[mensajes_urgentes-1].Telefono_s,     0, 50);
        }

        mensajes_urgentes = mensajes_urgentes - 1;
        contador_borrando = 0;
        estado = 1;
        break;
```

En caso de que haya 5 o menos mensajes del tipo del aviso atendido, se hace uso de la función memset para lograr que salga vacía la fila del mensaje más reciente (último) desplazado hacia arriba, de no ser así, estaríamos representando en la tabla el mismo aviso dos veces.

- Estado 4 → aplica exactamente la misma idea pero para los avisos de tipo no urgente. El código es el siguiente:

```
case 4:
        for(i = IndiceNoUrgente; i < mensajes_no_urgentes - 1; i++){
           strcpy(RegistroNoUrgente[i].Tipo_mensaje_s, RegistroNoUrgente[i+1].Tipo_mensaje_s);
           strcpy(RegistroNoUrgente[i].Nombre_s,       RegistroNoUrgente[i+1].Nombre_s);
           strcpy(RegistroNoUrgente[i].Apellidos_s,    RegistroNoUrgente[i+1].Apellidos_s);
           strcpy(RegistroNoUrgente[i].Direccion_s,    RegistroNoUrgente[i+1].Direccion_s);
           strcpy(RegistroNoUrgente[i].Mensaje_s,      RegistroNoUrgente[i+1].Mensaje_s);
           strcpy(RegistroNoUrgente[i].Hora_s,         RegistroNoUrgente[i+1].Hora_s);
           strcpy(RegistroNoUrgente[i].Edad_s,         RegistroNoUrgente[i+1].Edad_s);
           strcpy(RegistroNoUrgente[i].Telefono_s,     RegistroNoUrgente[i+1].Telefono_s);
        }

        if(mensajes_no_urgentes <= 5){
           memset(RegistroNoUrgente[mensajes_no_urgentes-1].Tipo_mensaje_s, 0, 50);
           memset(RegistroNoUrgente[mensajes_no_urgentes-1].Nombre_s,       0, 50);
           memset(RegistroNoUrgente[mensajes_no_urgentes-1].Apellidos_s,    0, 50);
           memset(RegistroNoUrgente[mensajes_no_urgentes-1].Direccion_s,    0, 50);
```

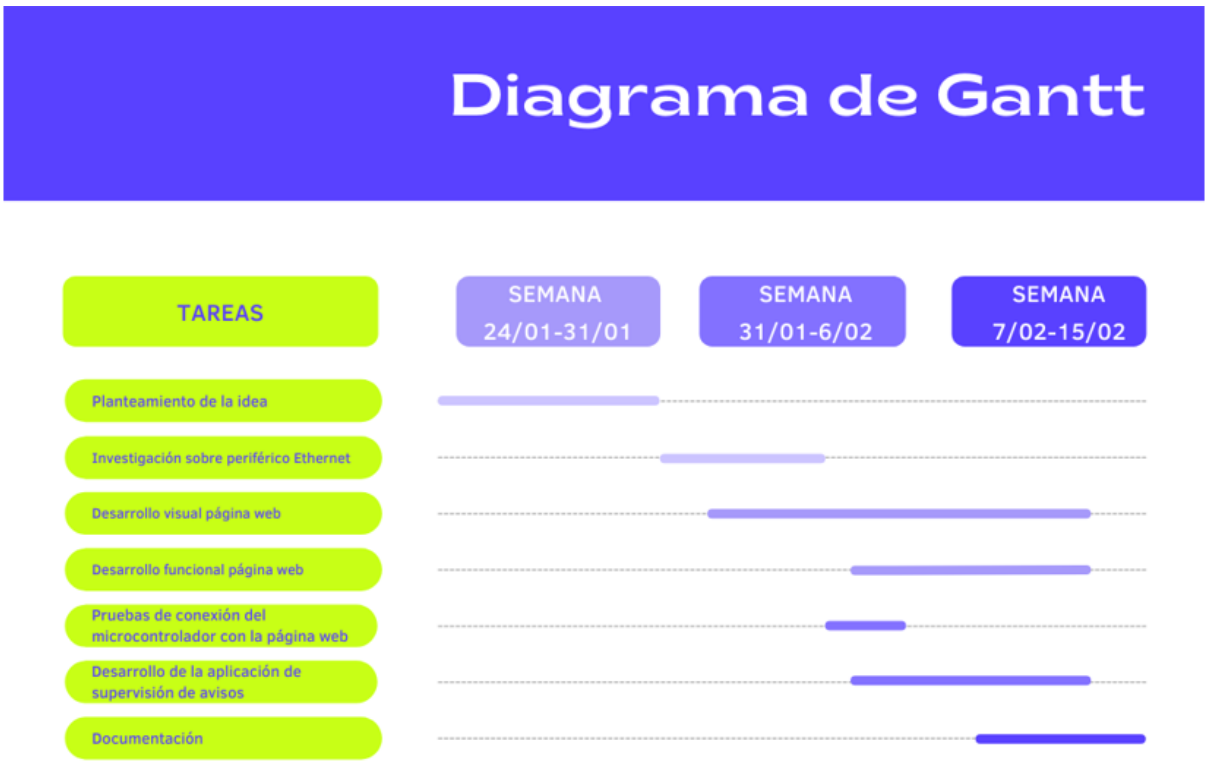
```
    memset(RegistroNoUrgente[mensajes_no_urgentes-1].Mensaje_s,      0, 50);
    memset(RegistroNoUrgente[mensajes_no_urgentes-1].Hora_s,         0, 50);
    memset(RegistroNoUrgente[mensajes_no_urgentes-1].Edad_s,         0, 50);
    memset(RegistroNoUrgente[mensajes_no_urgentes-1].Telefono_s,     0, 50);
}

mensajes_no_urgentes = mensajes_no_urgentes - 1;
contador_borrando = 0;
estado = 1;
break;
```

El resto de las funciones tan solo implementan animaciones, colores de fondo, letras, etc…, por lo que consideramos que no es merecedor de engrosar la memoria al ser algo bastante repetitivo.

### 3.3.- Cronograma de tareas:

Debido a la alta exigencia del conjunto de asignaturas de este cuatrimestre, fue muy complicado comenzar el desarrollo del proyecto previamente a las fechas que se muestran en el siguiente diagrama. Con lo cual para hacer factible el desarrollo del mismo dentro del tiempo disponible, era necesaria una planificación ordenada y clara de las tareas a desarrollar, buscando el paralelismo entre la mayoría de ellas y haciendo un reparto que permitiese aprovechar lo mejor posible tanto los recursos humanos como temporales disponibles.
Para visualizar el reparto temporal planteado para la resolución del proyecto se presenta el siguiente Diagrama de Gantt:



### 3.3.1- Reparto de tareas:

Una vez se tiene planteado la temporización estimada para resolver el proyecto, también es interesante comentar (a día de realización de la documentación con la mayoría de tareas resueltas) cómo se ha llevado a cabo el reparto de las diferentes tareas planteadas:

| **Tarea** | **Miembro asignado** | **Descripción** |
|---|---|---|
| Planteamiento de la idea | Ambos | Búsqueda de una idea adecuada a los objetivos del proyecto |
| Investigación sobre periférico Ethernet | Ambos | Comprensión del funcionamiento del controlador Ethernet y sus posibilidades |
| Desarrollo visual página web | Francisco Javier | Elaboración de la parte visual de la página web para que presente una interfaz agradable al uso |
| Desarrollo funcional página web | Francisco Javier | Desarrollo de la parte que implementa la funcionalidad de la página web en cuanto a intercomunicación con la aplicación de avisos. |
| Pruebas de conexión del microcontrolador con la página web | Juan de Dios | Comprobación de interacción correcta del microcontrolador a la hora de recibir información procedente de la página web |
| Desarrollo de la aplicación de supervisión de avisos | Ambos | Puesta en marcha de la aplicación desarrollada sobre la pantalla FT800 para gestionar y almacenar los avisos entrantes desde la página web |
| Documentación | Ambos | Desarrollo de la memoria y vídeo demostrativo del sistema en funcionamiento |

# 4.- Manual de usuario

En este apartado, ilustraremos las acciones a realizar para hacer uso del trabajo explicado. Dividiremos entre página web (donde se formalizan y envían los avisos) y aplicación de supervisión (donde se reciben y gestionan dichos avisos).

## 4.1.- Página web

En la página web encontramos dos pestañas. Una primera donde se puede hacer click en varias imágenes que nos enlazarán con la página de Texas Instruments, la página de la ETSI o bien a un blog que recoge las diferentes asociaciones que ofrecen apoyo a personas con diversidad funcional y enlace a sus diferentes páginas web. Para ello basta con hacer click sobre las imágenes. El resultado de lo comentado es el siguiente:


***Figura 6:*** *Pantalla inicial de la página web con diferentes enlaces*

Pulsando en la parte izquierda de la página donde pone “Servicios de Ayuda Contacto con supervisor Alerta de emergencia” (remarcado su fondo en azul) nos lleva a la segunda pestaña de la página web.

En esta segunda pestaña, tenemos el formulario para rellenarlo y enviarlo. Mencionar que salvo el campo de “¿Es urgente?” y el mensaje, el resto son obligatorios y no podremos enviar el formulario a menos que todos los demás campos estén rellenados.

En la imagen adjunta se muestran estos resultados:

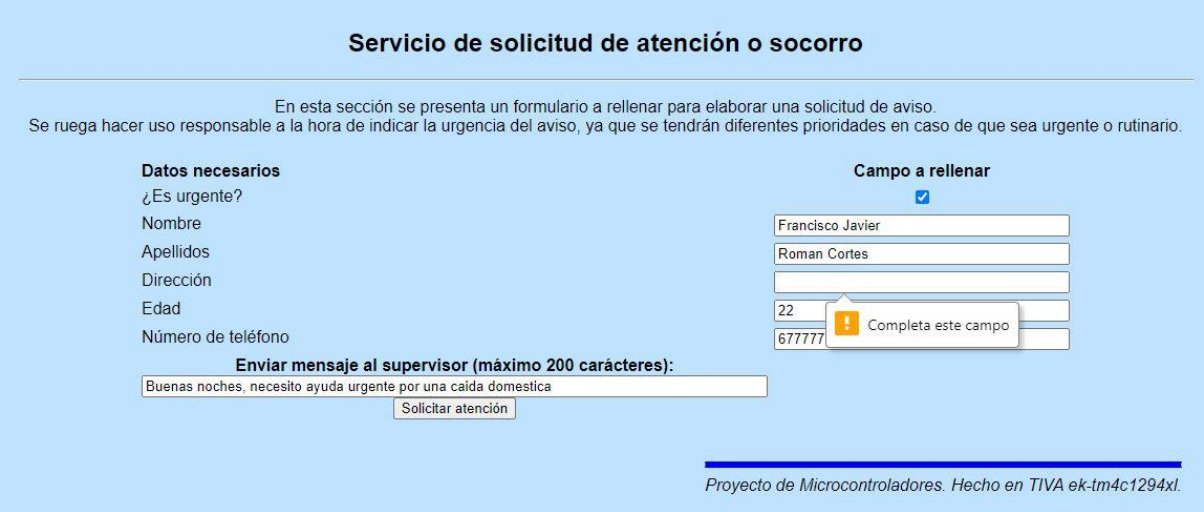



***Figura 7:*** *Ejemplo de uso de la web donde no se rellena uno de los campos obligatorios y se intenta enviar un aviso*

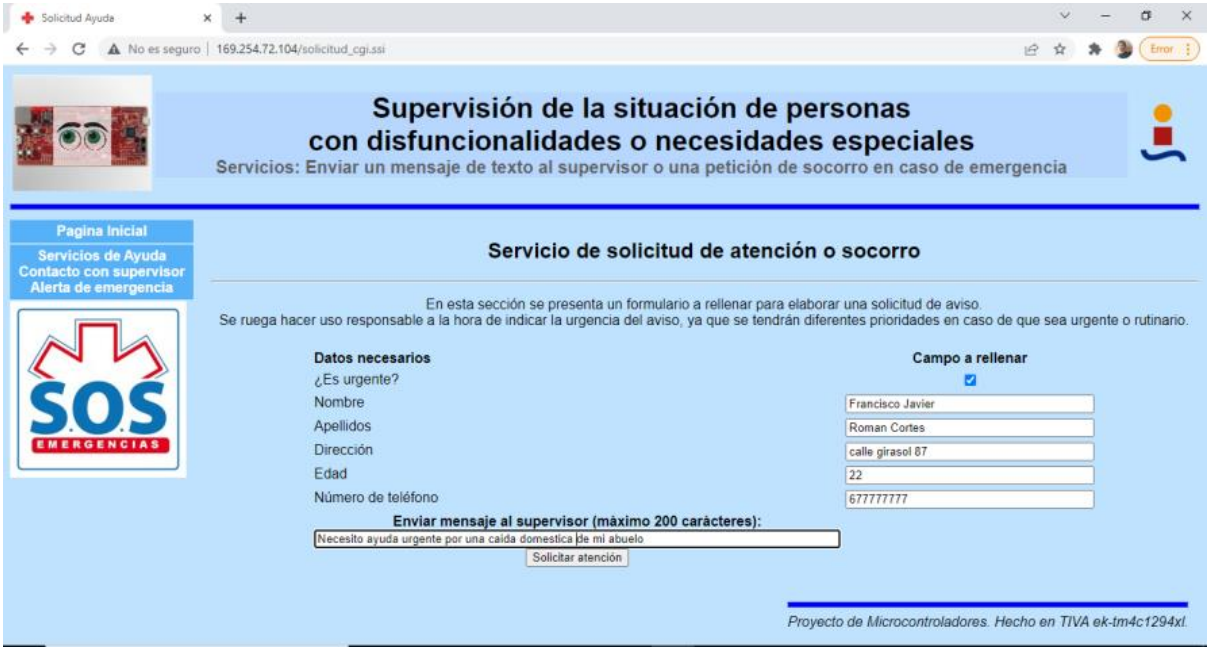



***Figura 8:*** *Pantalla del formulario de la página web*

Una vez rellenados al menos los campos obligatorios, clickamos en “Solicitar atención”. Esto hará que la información se envíe al microcontrolador vía Ethernet.

### 4.2.- Aplicación de recepción y almacenamiento de avisos

En cuanto al programa desarrollado, se mostrarán diferentes pantallas en las que el usuario (entendiendo como usuario la persona encargada de supervisar los avisos entrantes) podrá realizar varias acciones. Antes de comenzar recordar que, la habilitación de todos los botones no se hará efectiva hasta pasados dos segundos desde que entramos en el estado.
De nuevo, dadas las limitaciones del prototipo realizado, es necesario primero que el microcontrolador abra el servidor httpd con la página web diseñada, para que posteriormente haya interacción desde el ordenador conectado vía Ethernet con la aplicación desarrollada, pero sin una “conexión remota”.

Definiremos la funcionalidad y cómo se usan los diferentes menús de esta aplicación:

- Comenzamos con la pantalla inicial donde el usuario no debe hacer nada salvo esperar hasta que el protocolo DHCP nos proporcione una dirección IP para nuestra página web. La pantalla de carga de este estado es la siguiente:


***Figura 9:*** *Pantalla inicial del programa*

- En la siguiente pantalla, el programa nos muestra la dirección IP obtenida y se indica que se debe abrir un buscador e introducirla. Una vez hecho, el usuario deberá pulsar el botón “IP introducida” transicionando al menú con los diferentes registros que almacenarán los avisos entrantes.

- La pantalla es la siguiente:

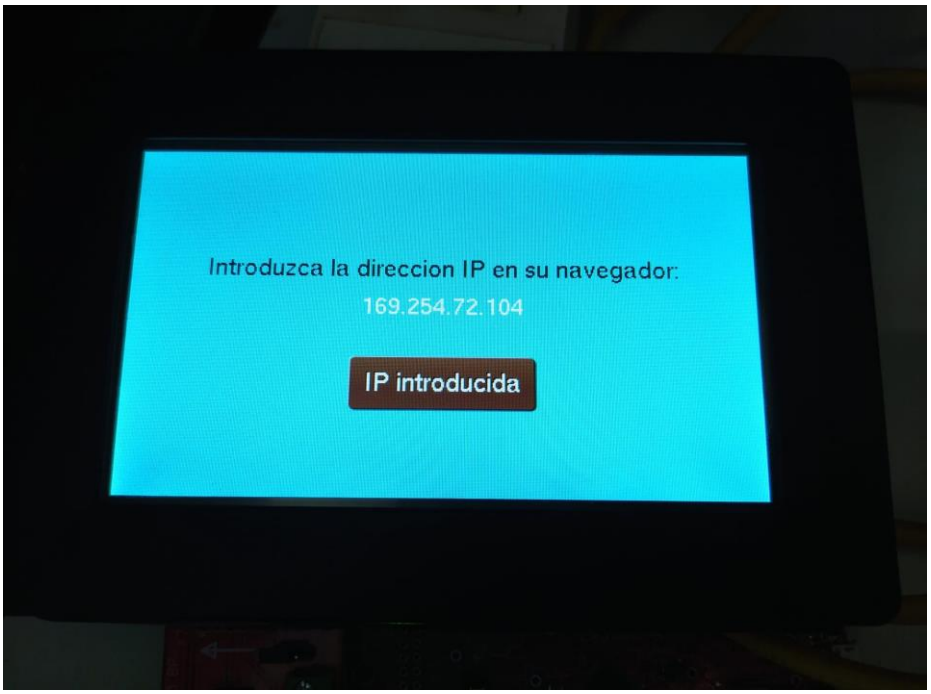



***Figura 10:*** *Pantalla para introducir la dirección IP en el buscador*

- El menú contiene una tabla con los diferentes avisos entrantes clasificados por orden de llegada, es decir, los situados en la parte superior de la tabla, corresponden a los más antiguos. El usuario podrá pulsar en el registro de la tabla correspondiente a un aviso siempre y cuando exista un aviso en la cuadrícula donde pulsa, en caso contrario, el programa no hará nada. La pantalla del registro es la siguiente:

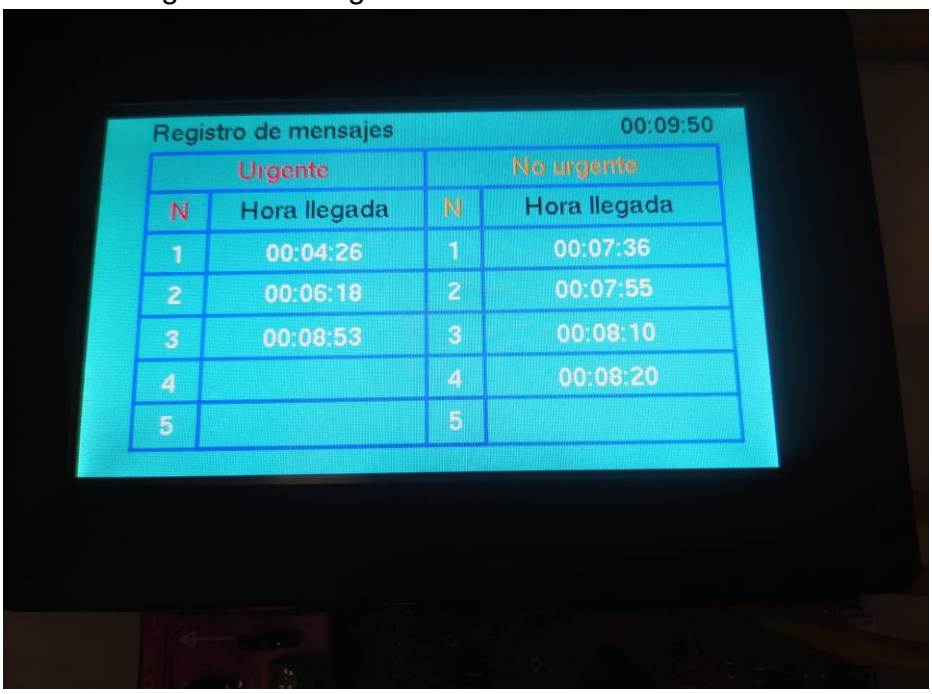



***Figura 11:*** *Pantalla del menú con tabla recopilatoria de los avisos entrantes (I)*


***Figura 12:*** *Pantalla del menú con tabla recopilatoria de los avisos entrantes (II)*

- Si nos llega un aviso estando en el menú, automáticamente pasaremos a la pantalla de alerta donde se nos avisará de ello (si llega mientras se está viendo un aviso, la alerta saltará al volver al menú de registros). En esta pantalla, el usuario tan solo debe esperar 4 segundos hasta que termine la pantalla de alerta de aviso entrante:


***Figura 13:*** *Pantalla de alerta*

- Si pulsamos en un aviso en la pantalla de registro pasaremos a la pantalla donde se muestran los detalles del aviso salvo el mensaje.

  En esta pantalla se da la opción de pulsar “Volver” lo que nos devolverá a la pantalla del menú o pulsar “Ver mensaje” lo que nos conducirá a la pantalla del mensaje. Esta pantalla tiene la siguiente apariencia:

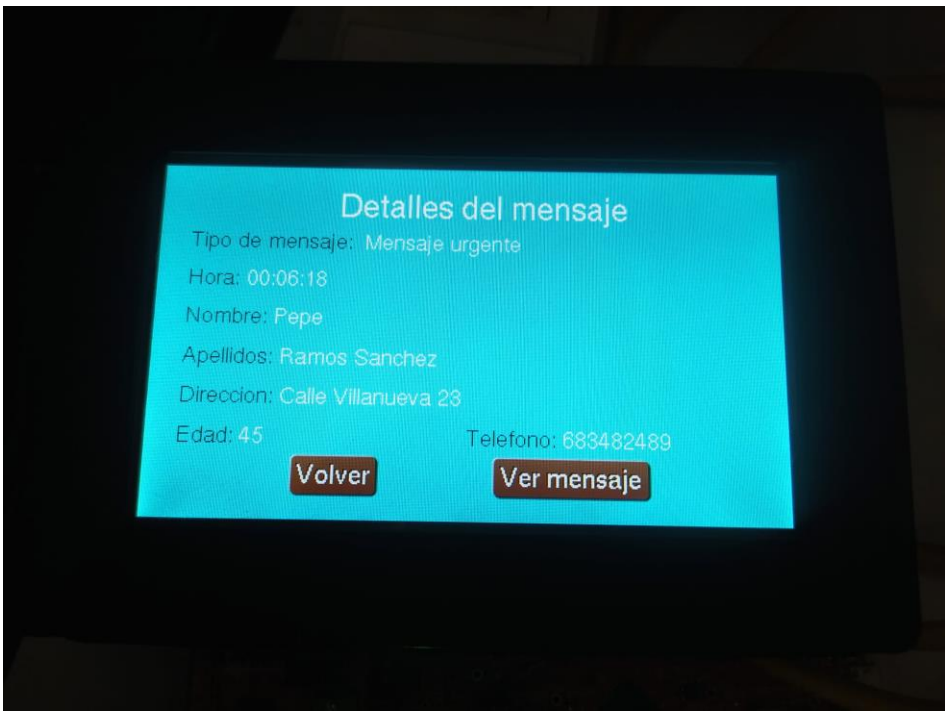



***Figura 14:*** *Pantalla con los detalles del aviso*

Si hemos pulsado para ver el mensaje, veremos el mensaje del aviso, en caso de no contener mensaje, el programa avisa al usuario. Una vez en esta pantalla podemos o bien pulsar “Volver”, lo que nos devolverá a la pantalla con los detalles del aviso o bien pulsar “Enviar ayuda”, lo que nos llevará a la pantalla de ayuda.

La apariencia de esta pantalla es la siguiente:

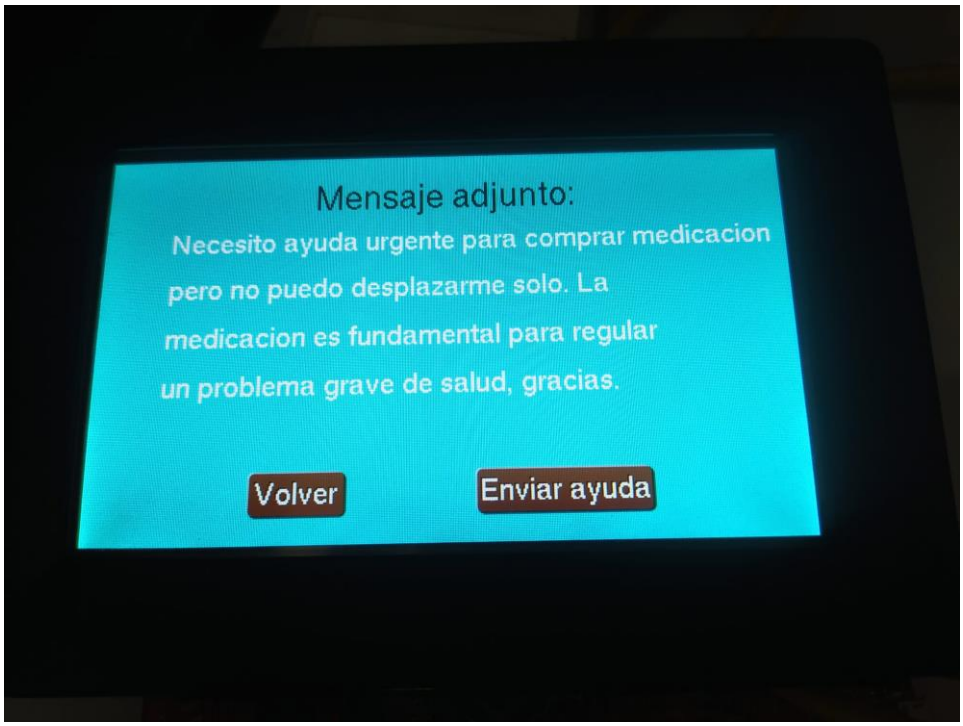



***Figura 15:*** *Pantalla que muestra el mensaje del aviso*

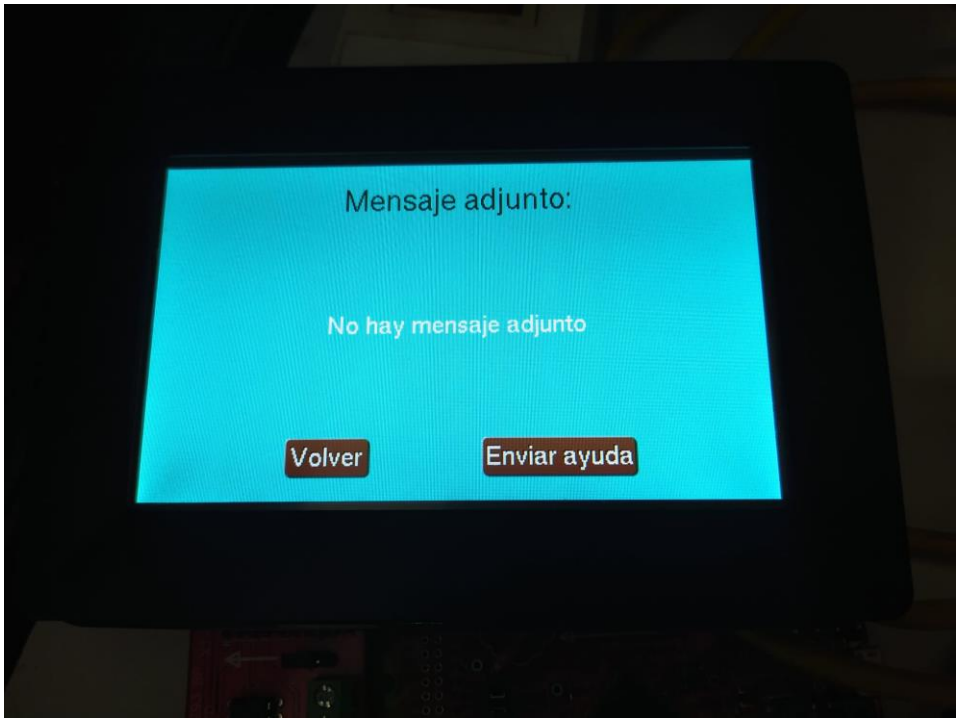



***Figura 16:*** *Pantalla del mensaje indicando que no hay mensaje adjunto con el aviso*

- Si hemos enviado ayuda, la pantalla de envío de ayuda nos mostrará una animación de un cochecito moviéndose, que emula el envío de ayuda para la persona que la ha solicitado. En esta pantalla el usuario solo debe esperar a que pasen 4 segundos. La pantalla es la siguiente:


***Figura 17:*** *Pantalla de envío de ayuda*

- Tras esto, pasamos a la pantalla de borrado donde se muestra animación indicativa de que se está borrando dicho aviso de la “base de datos” del programa, ya que el aviso ha sido atendido. El usuario nuevamente solo debe esperar a que pasen 3 segundos en este caso.

- La pantalla es la siguiente:


***Figura 18:*** *Pantalla de borrado*

- Una vez termine esta pantalla, ya volveríamos al menú de la pantalla registro. Al borrarse dicho registro, todos los que hubiese por debajo del mismo, suben un puesto en la tabla quedando ordenados de nuevo.

Recalcar además que, aunque el tamaño limitado de la pantalla permita mostrar 5 registros de cada tipo, si hay más avisos se encolan correctamente, de manera que al atender (y eliminar) alguno de los 5 registros, los ocultos subirán un puesto de nuevo, siendo que el que fuera el 6º registro (que antes no era visible) ahora aparecería en el 5º lugar de la tabla.

Y con esto habríamos abarcado todas las posibles interacciones que el usuario puede realizar en el programa.

# 5.- Conclusiones

Se llega al punto de sacar conclusiones sobre el proyecto realizado, una vez aquí hay varias preguntas que plantearse con respecto a los objetivos que se planteaban para el mismo:

- ¿Mejora el sistema automático desarrollado la vida de una persona con diversidad funcional? ¿Se podría llevar a la práctica?

Como se ha ido mencionando a lo largo de esta memoria, el prototipo desarrollado requiere de conexión Ethernet con el ordenador para comunicarse e interaccionar, por lo que no habría una supervisión remota “real” en la fase actual del sistema desarrollado.
Sin embargo, si se invirtieran mayor cantidad de recursos y tiempo en este prototipo e idea, podría llegar a desarrollarse una aplicación prácticamente con la misma funcionalidad, pero cuyos protocolos de conexionado con la web sean tales que permitan interacción remota. Consiguiendo esto, sí consideramos que el sistema podría ser muy útil para un cierto sector de la población que experimenta dificultades en su día a día, y podrían tener una supervisión garantizada si fuese necesario, evitando que queden abandonados o desatendidos ante posibles situaciones que supongan cierto riesgo para ellos debido a su condición especial.

Aunque se ha enfocado en la memoria bastante la idea en una supervisión mediante una centralita (empresa privada o pública), también sería totalmente viable desarrollar un sistema embebido que pudiese ser accesible en precio al público general de manera que por poner un ejemplo, una persona de la que dependa otra con diversidad funcional, pueda disponer de dicho sistema y recibir los avisos directamente para conocer las necesidades de dicha persona de manera rápida, variando así el enfoque dado de la centralita que supervisa los avisos de múltiples fuentes.

- ¿Es este microcontrolador una opción adecuada para el desarrollo de sistemas automatizados?

Como mencionamos en algún apartado anterior, el microcontrolador satisface sin problemas los requerimientos de una aplicación relativamente compleja, sin retrasos y sin falta de memoria. Esto último es fundamental a la hora de “embeber” la página web con imágenes y demás elementos relativamente pesados sin que falte espacio para ello. A la hora de, por ejemplo, encapsular este sistema para una posible comercialización (considerando conexión inalámbrica y posibilidad de interacción remota del sistema (no estando presente estas características en nuestro prototipo)), quizás podría llegar a conformar un dispositivo de dimensiones relativamente compactas (pantalla, micro y batería encapsulados en un dispositivo único), pudiendo llegar a ser portable.

Subjetivamente, pensamos que con la velocidad con la que avanza la era digital y la deshumanización de la sociedad en estos tiempos tan frenéticos, es necesario mediante las ventajas que ofrece dicha digitalización, tratar de plantear soluciones que ayuden a no dejar de lado a las personas con diversidad funcional.
Por ello, este proyecto y su enfoque, nos parece una buena elección temática a la hora de dar soluciones a problemas reales haciendo uso de la tecnología y sus posibilidades, para facilitar la vida de dichas personas que, por su condición, tienen un día a día más complejo de sobrellevar.

# 6.- Anexo de programas

## 6.1.- Index.htm (página inicial de la web):

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
  <title>Supervision</title>
  <link rel="stylesheet" type="text/css" href="styles.css"/>
  <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"/>
  <script src="javascript.js" language="JavaScript1.2" charset="utf-8"></script>
  <script src="javascript_load.js" language="JavaScript1.2" charset="utf-8"></script>
 </head>
 <body>
  <div id="heading">
   <table width="100%">
    <tr>
     <td>
      <a id="Logo_pagina" target="_" href="https://www.ti.com/">
       <img src="Logo2.jpg" alt="Supervision con TIVA">
      </a>
     </td>
     <td width="100%">
      <div id="Encabezado_h1">
       Supervisión de la situación de personas <br>con disfuncionalidades o necesidades especiales
      </div>
      <div id="Encabezado_h2">
       Servicios: Enviar un mensaje de texto al supervisor o una petición de socorro en caso de emergencia
      </div>
     </td>
     <td>
      <a id="Logo_ETSI" target="_" href="https://www.etsi.us.es/">
       <img src="LogoEtsi.png" alt="LogoETSI">
      </a>
     </td>
    </tr>
   </table>
   <hr>
  </div>
  <div id="menu">
   <ul>
    <li><a href="index.htm" id="index">Pagina Inicial</a></li>
    <li><a href="solicitud_cgi.ssi" id="solicitud_cgi">Servicios de Ayuda<br>Contacto con supervisor<br>Alerta de
emergencia</a></li>
   </ul>
   <img src="sos.jfif" alt="Imagen miscelanea de SOS">
  </div>
  <div id="content">
  </div>
  <div id="footing">
   <hr>
   Proyecto de Microcontroladores. Hecho en TIVA ek-tm4c1294xl.
  </div>
  <div id="texto">
   <h2>Pulsar en el menú de la izquierda para acceder a la ventana en la que realizar un aviso</h2>
   </div>
   <div id="diversidad_funcional">
    <a id = "diversidad_funcional" href= "https://www.sunrisemedical.es/blog/asociaciones-de-discapacitados">
    <img src="diversidad_funcional.jpg" alt="Imagen diversidad funcional"
    width="765"
    height="350">
    </a>
  </div>
 </body>
</html>
```

## 6.2.- Styles.css (define los estilos de los diferentes elementos de la web):

```
/* Styling for the entire document. */
body
{
    background-color: #bfe3ff;
    color: black;
    font-family: Ariel,Helvetica,sans-serif;
}

/* Styling for the heading div. */
#heading
{
    clear: both;
    margin-bottom: 10px;
}

#Encabezado_h1
{
    background-color: rgba(110, 110, 255, 0.1);
    color: #000000;
    font-size: 2em;
    font-weight: bold;
    text-align: center;
}

#Encabezado_h2
{
    background-color: rgba(110, 110, 255, 0.1);
    color: #605e5e;
    font-size: 1.25em;
    font-weight: bold;
    text-align: center;
}

#heading hr
{
    border-bottom: 3px solid #0000ff;
    border-top: 3px solid #0000ff;
}

#texto
{
    position: absolute;
    top: 160px; left: 270px;
    /*background-color: #a9d9ff;*/
    color:  blue;
}

#diversidad_funcional
{
    position: absolute;
    top: 120px; left: 190px;
}

#heading a img
{
    border: 0;
}

/* Styling for the menu div. */
#menu
{
    float: left;
    margin-bottom: 10px;
    width: 225px;
```

```
}

#menu ul
{
    list-style-type: none;
    margin: 0;
    padding: 0;
}

#menu a:link, #menu a:visited
{
    background-color: #57b1f8;
    color: #ffffff;
    display: block;
    font-weight: bold;
    margin-bottom: 2px;
    margin-top: 2px;
    padding: 4px;
    text-align: center;
    text-decoration: none;
    width: 200px;
}

#menu a:hover, #menu a:active
{
    background-color: #0000ff;
}

/* Styling for the content div. */
#content
{
    margin-bottom: 10px;
    margin-left: 225px;
}

#content_heading
{
    font-size: 1.5em;
    font-weight: bold;
    text-align: center;
}

/* Styling for the footing div. */
#footing
{
    position: absolute;
    top: 600px; left: 890px;
    font-size: 1.0em;
    font-style: italic;
    text-align: right;
}

#footing hr
{
    border-bottom: 3px solid #0000ff;
    border-top: 3px solid #0000ff;
}
```

## 6.3.- Solicitud_cgi.ssi (pestaña que permite formalizar y enviar un aviso):

```
<!DOCTYPE HTML>
<html>
<head>
   <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
   <title>Solicitud Ayuda</title>
   <link rel="stylesheet" type="text/css" href="styles.css" />
   <link rel="shortcut icon" type="image/x-icon" href="favicon.ico" />
   <!--script src="javascript.js" language="JavaScript1.2" charset="utf-8"></script-->
</head>

<body>
   <div id="heading">
    <table width="100%">
     <tr>
      <td>
       <a id="Logo_pagina" target="_" href="https://www.ti.com/">
        <img src="Logo2.jpg" alt="Supervision con TIVA">
       </a>
      </td>
      <td width="100%">
       <div id="Encabezado_h1">
        Supervisión de la situación de personas <br>con disfuncionalidades o necesidades especiales
       </div>
       <div id="Encabezado_h2">
        Servicios: Enviar un mensaje de texto al supervisor o una petición de socorro en caso de emergencia
       </div>
      </td>
      <td>
       <a id="Logo_ETSI" target="_" href="https://www.etsi.us.es/">
        <img src="LogoEtsi.png" alt="LogoETSI">
       </a>
      </td>
     </tr>
    </table>
    <hr>
   </div>
   <div id="menu">
    <ul>
     <li><a href="index.htm" id="index">Pagina Inicial</a></li>
     <li><a href="solicitud_cgi.ssi" id="solicitud_cgi">Servicios de Ayuda<br>Contacto con supervisor<br>Alerta de
emergencia</a></li>
    </ul>
    <img src="sos.jfif" alt="Imagen miscelanea de SOS">
   </div>
   <div id="content">
   <table width="100%">
      <tbody>
         <tr>
            <td align="left" valign="top" width="75%">
               <center>
                  <h2 align="center">Servicio de solicitud de atención o socorro</h2>
               </center>
               <hr size="2" width="100%">
               <ul></ul><center>En esta sección se presenta un formulario a rellenar para elaborar una solicitud de aviso.<br>Se
ruega hacer uso responsable a la hora de indicar la urgencia del aviso, ya que se tendrán diferentes prioridades en caso de
que sea urgente o rutinario.
               <br>
               <br>
               </center>
               <table align="center" border="0" width="80%">
                  <tbody>
                     <tr>
                        <td>
                           <form method="get" action="recoger_datos.cgi" name="recoger_datos">
```

```
                    <table align="center" border="0" cellpadding="2" cellspacing="2" width="60%">
                      <tbody>
                        <tr>
                          <td align="left" valign="top">
                            <b>Datos necesarios</b>
                          </td>
                          <td align="center" valign="top">
                            <b>Campo a rellenar</b>
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">¿Es urgente?</td>
                          <td align="center" valign="top">
                            <input name="urgente" value="0" type="checkbox">
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">Nombre</td>
                          <td align="center" valign="top">
                            <input value="" maxlength="34" size="35" name="nombre" required>
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">Apellidos</td>
                          <td align="center" valign="top">
                            <input value="" maxlength="34" size="35" name="apellidos" required>
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">Dirección</td>
                          <td align="center" valign="top">
                            <input value="" maxlength="34" size="35" name="direccion" required>
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">Edad</td>
                          <td align="center" valign="top">
                            <input value="" maxlength="3" size="35" name="edad" required>
                          </td>
                        </tr>
                        <tr>
                          <td align="left" valign="top">Número de teléfono</td>
                          <td align="center" valign="top">
                            <input value="" maxlength="9" size="35" name="numerotelefono" required>
                          </td>
                        </tr>
                        <tr align="center">
                          <td valign="top">
                            <b>Enviar mensaje al supervisor (máximo 200 carácteres):<br></b>
                            <input maxlength="200" size="80" name="TextoSupervisor">
                            <br>
                            <input name="Update" value="Solicitar atención" type="submit">
                          </td>
                        </tr>
                      </tbody>
                    </table>
                  </form>
                </td>
              </tr>
            </tbody>
          </table>
          <ul></ul>
          <br>
          <br>
        </td>
      </tr>
```

```
      </tbody>
    </table>
    </div>
    <div id="footing">
      <hr>
      Proyecto de Microcontroladores. Hecho en TIVA ek-tm4c1294xl.
    </div>
</body>
</html>
```

## 6.4.- Proyecto.c (código principal que implementa la FSM y ejecuta el servidor de la página web embebida en el microcontrolador):

```
//*****************************************************************************//
//                                                                             //
//       Proyecto de microcontroladores SEPA 4º GIERM. Realizado por:          //
//         - Francisco Javier Román Cortés. (fraromcor3)                       //
//         - Juan de Dios Herrera Hurtado. (juaherhur)                         //
//                                                                             //
//*****************************************************************************//
/*****************************************************************************/
//
//     --> Resumen/Explicación del proyecto planteado:
//
// Aunque puede parecer algo áspero desde fuera, teníamos claro que queríamos
// explorar el periférico Ethernet y sus posibilidades. En primer lugar, comenzamos
// planteando la idea de comunicar 2 micros usando protocolo UDP, de manera que
// un cuidador pudiese tener comunicación directa con una persona con necesidades especiales.
// Sin embargo, la complejidad de establecer un intercambio de información entre ambos micros
// (al menos via Ethernet(UDP)) se hizo notoria y hubo que reenfocar el proyecto.
// En ese punto, vimos como otra opción montar una página web simple que estará embebida
// en el micro de manera que es el que la hostea. Dicha página ofrece una interfaz simple
// mediante la cual una persona con necesidades tiene contacto directo con una "centralita"
// donde se recibirán las notificaciones de avisos, separándolos en "Urgentes" y "No urgentes",
// de manera que se tiene definida una prioridad sobre qué avisos son más importantes.
// Esta persona podrá rellenar un pequeño formulario con sus datos e indicará si el aviso es urgente
// o no, además tiene la posibilidad de enviar un mensaje de texto con ciertos comentarios que vea
// conveniente.
// Para el punto de vista de la centralita, se ha desarrollado lo que se podría denominar como
// una aplicación o interfaz con diferentes menús que permiten gestionar el registro de los diferentes
// avisos recibidos, ya sea viéndolos y volviendo al registro sin actuar sobre dicho aviso, viendo los
// detalles (nombre, edad, dirección...), así como viendo el mensaje adjunto (si lo hubiese).
// En esta aplicación, se emula lo que sería el envío de ayuda a la solicitud entrante, moviendo el servomotr
// (puede emular la apertura de una puerta, por ejemplo), así como haciendo sonar cierto sonido.
// También, cuando se recibe un aviso o solicitud de ayuda, la interfaz muestra una alerta parpadeante visual y
sonora
// que permita reconocer con claridad la llegada de dicho aviso.
// Finalmente, entre otros detalles y animaciones desarrollados, destaca el borrado automático del aviso atendido
una
// vez se ha atendido y enviado ayuda a dicho aviso, reordenando en la pantalla de registros el resto de avisos.
//
//     --> Principios técnicos de funcionamiento:
//
// En términos generales, nos basamos en el controlador Ethernet disponible en
// Microcontrolador TIVA ek-tm4c1294xl y en la pila lwIP TCP/IP.  El protocolo
// que se utiliza para obtener una dirección de Ethernet es DHCP
// (Dynamic Host Configuration Protocol). Si DHCP llega a un timeout (45s) sin
```

```
// lograr obtener una dirección IP, se elige una IP estática usando AutoIP.
// La dirección obtenida, se muestra por la UART, de manera que se puede acceder
// a la página web confeccionada, hosteada por el microcontrolador a través de
// cualquier navegador.
// Respecto a esto último, la página web sólo es accesible desde el pc conectado
// al microcontrolador via Ethernet, lo cual limita las posibilidades del
// desarrollo de la aplicación realizada a una especie de prototipo. Es decir,
// podría ser un ejemplo de cómo prototipar un producto que podría escalar a
// una versión real y mucho más potente.
// A donde se quiere llegar con esto es que, si se dispusiese de más tiempo,
// recursos y conocimientos sobre protocolos de red (principalmente hosteo de webs
// embebidas en un microcontrolador de manera que sea accesible de manera remota),
// no sería inviable escalar este proyecto/prototipo a una fase en la cual, la aplicación
// quedase hosteada en una página web habitual embebida en este microcontrolador y accesible
// de manera remota desde cualquier terminal (PC, móvil, etc.), haciéndolo un sistema mucho más práctico.
//
//      --> Conceptos acerca de cómo se obtiene la información procedente de la página web:
//
// Una vez entramos en la página web, nos aparece la página inicial, con una presentación de la página.
// Para poder conformar y enviar un aviso, debemos ir al enlace situado en la parte izquierda:
// "Servicios de Ayuda//Contacto con supervisor//Alerta de emergencia".
// En este enlace, aparece un formulario, en el cual se solicitan los datos necesarios para poder enviar
// la ayuda, así como la posibilidad de indicar si es urgente o no y de adjuntar un mensaje explicativo si se
considera necesario.
//
// Para recoger la información procedente de este formulario, se usan formularios HTML estándar que pasan
parámetros
// a un handler (o manejador de interrupción) CGI (Common Gateway Interface) que se ejecuta en la placa.
// En el programa, se registra el handler, de manera que cuando se pulse el botón de "Solicitar Atención",
// dicho formulario, definido por: <form method="get" action="recoger_datos.cgi" name="recoger_datos">,
ejecuta la acción
// "recoger_datos.cgi", que activa el handler donde se decodifica la información (se explicará con más detalle en
el handler).
// Para decodificar dicha información se parsea la página HTML con los datos incluidos y se decodifica el
mensaje introducido
// asociado a cada "id" de cada "text box": <input value="" maxlength="34" size="35" name="nombre" required>,
donde el "nombre", sería
// la "id" a buscar al parsear, y "value" el valor de la string recibida en dicho campo.
// Con esto, podemos recoger dicha información y pasar al problema de su procesamiento y almacenamiento de
manera ordenada.
//
// Destacar también que, al enviar el formulario, la página web se refresca, quedando preparada para el envío de
un nuevo aviso.
//
//      --> Elaboración de la página web: (HTML básico, modificación de estilos...)
//
// Se incluyen en la carpeta "fs\" los códigos fuentes que conforman dicha página web.
// Si se modifica alguno de estos archivos (para modificar la web), el archivo de imagen del sistema (iofsdata.h)
// debe ser recompilado mediante la utilidad disponible "makefsfile" disponible en la carpeta de TivaWare.
// Para usar esta herramienta, tenemos que desplazarnos al directorio donde esté alojado el proyecto (con cd \..)
// y tras ello: >> ../../../../tools/bin/makefsfile -i fs -o io_fsdata.h -r -h -q
// Posiblemente se puede modificar este path si se desplaza el archivo "makefsfile" a otro lugar deseado.
//
// Referencias sobre la pila lwIP(Lightweight IP):
// http://savannah.nongnu.org/projects/lwip/
// https://git.savannah.nongnu.org/cgit/lwip.git
//
```

```
//*****************************************************************************************************************
*****************//
////////////////////////////////////////////////////// DECLARACI?N DE LIBRERIAS ////////////////////////////////////////////////////////
#include <stdbool.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <FT800_TIVA.h>
#include "driverlib2.h"
#include "utils/locator.h"
#include "utils/lwiplib.h"
#include "utils/uartstdio.h"
#include "utils/ustdlib.h"
#include "httpserver_raw/httpd.h"
#include "drivers/pinout.h"
#include "io.h"
#include "cgifuncs.h"

//*****************************************************************************************************************
*****************//

#define dword long
#define byte char

// Defines para los botones presentes en la interfaz de la aplicación desarrollada:
#define Boton_volver Boton(HSIZE/2-130, 7*VSIZE/9+14, 63, 28, 23, "Volver")
#define Boton_ver_mensaje Boton(HSIZE/2+20, 7*VSIZE/9+14, 115, 28, 23, "Ver mensaje")
#define Boton_enviar_ayuda Boton(HSIZE/2+20, 7*VSIZE/9+14, 115, 28, 23, "Enviar ayuda")
#define Boton_IP_introducida Boton(HSIZE/2-70, VSIZE/2+30, 140, 40, 23, "IP introducida")

//
================================================================================
==========================================================
// Declaración/Inicialización de variables para pantalla FT800
//
================================================================================
==========================================================

char chipid = 0;                    // Holds value of Chip ID read from the FT800

unsigned long cmdBufferRd = 0x00000000;       // Store the value read from the REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;       // Store the value read from the REG_CMD_WRITE register
unsigned int t=0;

unsigned long POSX, POSY, BufferXY;
unsigned int CMD_Offset = 0;
int32_t REG_TT[6];
const int32_t REG_CAL[6]={22073,59,-1044578,400,-16911,16162095};
const int32_t REG_CAL5[6]={32146, -1428, -331110, -40, -18930, 18321010};   // Calibraciones de las pantallas
(se ha realizado en la de 5")

//==============================================================================
==========================================================
// Lista de variables:
// CHAR:
char cadena_primera_linea[50];  //Al mostrar el mensaje de un aviso (si lo hay), se han creado cinco strings
char cadena_segunda_linea[50];  //que permiten separar todo el mensaje recibido de manera que se pueda
char cadena_tercera_linea[50];  //mostrar correctamente por la pantalla sin que el mensaje se salga
```

```
char cadena_cuarta_linea[50];   //de los límites de la pantalla, que es lo que ocurriría si no se separase
char cadena_quinta_linea[50];   //el mensaje en trozos
char cadena_mensaje[200];       //En esta string se guarda el mensaje del aviso que, posteriormente se dividirá en las cinco cadenas previamente comentadas
char cadena_urgente[] = "Mensaje urgente";      //Según si el aviso es urgente o no, el manejador de
char cadena_no_urgente[] = "Mensaje rutinario"; //interrupción da un valor u otro a "cadena_tipo_mensaje"
char cadena_tipo_mensaje[50];                   //Toma el valor de alguna de las dos cadenas predefinidas según si el mensaje es urgente o no, todo ello se realiza en el manejador de interrupción

// VOLATILE CHAR (para string modificada en interrupciones):
volatile char cadena_hora[50];  //Cuando llega un nuevo aviso, el manejador de interrupción rellena el campo "Hora_s" de la estructura del aviso con esta cadena (se basa en el tiempo de ejecución del programa que está controlado por el TIMER1 A)
volatile char Nombre[50];
volatile char Apellidos[50];    //Cadenas en las que se guardan los valores del aviso que luego se usarán para rellenar los diferentes campos de la
volatile char Direccion[50];    //de la estructura que caracteriza al aviso (faltaría por caracterizar el campo "Tipo_mensaje_s" que se rellena con
volatile char Edad[50];         //"cadena_tipo__mensaje")
volatile char Telefono[50];
volatile char Mensaje[200];

// VOLATILE INT (para enteros modificados en interrupciones (CGI handler y timer)):
volatile int Max_pos = 4200;        //Posición izquierda del servomotor
volatile int Med_pos = 2750;        //Posición central del servomotor
volatile int Min_pos = 1300;        //Posición derecha del servomotor
volatile int flag_estado_alerta = 0;//Variable que permite realizar el cambio de color del fondo de la pantalla en el estado de alerta que avisa de que ha llegado un nuevo aviso
volatile int contador_estado8 = 0;  //Contadores que permiten realizar animaciones en el estado 2 (pantalla de alerta de aviso entrante),
volatile int contador_estado2 = 0;  //en el estado 8 (pantalla de envío de ayuda)
volatile int contador_borrando = 0; //y en el estado 5 (pantalla de borrado)
volatile int horas=0, minutos=0, segundos=0; //Variables que se actualizan en el TIMER1 A para llevar el tiempo de ejecución
volatile int estado = 0;            //Variable fundamental que rige la máquina de estados (FSM)
volatile int direccion_obtenida = 0;//Esta variable hace que permanezcamos en el estado 0 (pantalla de carga inicial) hasta que el protocolo DHCP nos proporciona una dirección IP para nuestra página web
volatile int cambio_estado = 0;     //Cuando llega un aviso, el manejador de interrupción activa esta variable que hará que en la máquina de estados pasemos a la pantalla que alerta de que hay un nuevo aviso (para que salte la pantalla de alerta debemos de estar en la pantalla del registro de mensajes)
volatile int cont_circulos = 0;     //Variable para realizar la animación de los "..." en la pantalla de carga inicial. La secuencia que realiza es "", ".", "..", "..." y así cíclicamente
volatile int mensajes_urgentes = 0; //Almacena la cantidad de avisos urgentes que tenemos almacenados
volatile int mensajes_no_urgentes = 0; //Almacena la cantidad de avisos no urgentes que tenemos almacenados
volatile int activa_estado_1 = 0;   //Estas variables se utilizan para habilitar la pulsación de los botones y del quinto mensaje
volatile int activa_estado_6 = 0;   //que aparece por pantalla en la columna de los mensajes urgentes. Esto se hace porque al
volatile int activa_estado_7 = 0;   //tener botones colocados en la misma posición en diferentes estados, al pulsarlos, inmediatamente
volatile int activa_estado_9 = 0;   //se activaba también el botón del estado al que se pasaba que se encontraba en coordenadas similares al botón pulsado
volatile int urgente = 0;           //Toma valor '-1' si no se marca la casilla de aviso urgente y '0' si se marca

// INT:
int registro = 0;                   //En la pantalla de registro se muestran 5 avisos de cada tipo, según qué aviso se pulse para ver, esta variable toma valor en el rango [1,5], siendo 1 el aviso de más arriba (el más antiguo) y 5 el aviso de la parte inferior de la pantalla (no importa si el aviso es urgente o no, se usa para ambos)
```

```
int volver_mensaje_no_urgente = 0;  //En el estado 9 mostramos el mensaje del aviso en caso de tener, si volvemos para ver los detalles del aviso (nombre, apellidos...),
int volver_mensaje_urgente = 0;     //necesitamos saber si el aviso del que se deben mostrar los detalles era urgente (estado 6) o no (estado 7). Todo esto es posible conociendo también el valor de "registro"
int ver_mensaje_urgente = 0;        //Estas dos próximas variables son análogas pero al pasar del estado 6/7 donde se ven los detalles del aviso,
int ver_mensaje_no_urgente = 0;     //al estado 9, donde debemos saber qué mensaje debemos mostrar si urgente o no. Nuevamente nos apoyamos también en la variable "registro"
int borrado_urgente = 0;            //Misma idea que los dos pares de variables que preceden a estas dos, al salir del estado 5 (animación de que estamos borrando el aviso una vez se envía la ayuda), debemos saber si pasar al
int borrado_no_urgente = 0;         // estado 3 (urgente) o 4 (no urgente), que es donde realmente se borra el registro y se desplazan todos los avisos que llegaron después al que se borra una posición hacia arriba en el array de structs
int PeriodoPWM;                     //Periodo con el que se configura el PWM para mover el servomotor
int IndiceUrgente = 0;              //Se les da valor a la vez que "registro", es decir, al pulsar para ver
int IndiceNoUrgente = 0;            //los detalles de un aviso. Toman valor igual a registro - 1
int terminador = 0;                 //Esta variable nos permite saber en qué posición se encuntra el terminador del mensaje del aviso, pudiendo tomar valores en el rango [0,199]
int flag_primera_linea = 0;
int flag_segunda_linea = 0;         //Variables para realizar el guardado del mensaje del aviso en diferentes zonas de la pantalla de manera secuencial
int flag_tercera_linea = 0;         //Por ejemplo para el caso de la segunda línea, no se le dará valor hasta que se haya terminado de manipular
int flag_cuarta_linea = 0;          //y rellenar la primera línea. Así con el resto de líneas hasta llegar a la quinta que debe esperar a la cuarta línea
int flag_quinta_linea = 0;
int i = 0;
int j = 0;                          //Índices para recorrer bucles for para manipular los arrays de structs y las líneas con el mensaje del aviso
int k = 0;
int caracter = 0;                   //Al rellenar la segunda línea del mensaje por ejemplo, debemos saber hasta qué índice del mensaje se ha guardado en la primera línea, esta variable nos lo indica

// STRUCT:
struct StructResumen {
  char  Tipo_mensaje_s[50];
  char  Hora_s[50];
  char  Nombre_s[50];
  char  Apellidos_s[50];          //Struct que caracteriza cada aviso que llega de la página web (8 campos).
  char  Direccion_s[50];          //Se rellena en el manejador que salta cada vez que se envía una aviso desde la página
  char  Edad_s[50];
  char  Telefono_s[50];
  char  Mensaje_s[200];
};

// ARRAY OF STRUCTS:
//Array de struct de tamaño 20 para los avisos urgentes
struct StructResumen RegistroUrgente[20];

//Array de struct de tamaño 20 para los avisos no urgentes
struct StructResumen RegistroNoUrgente[20];

//////////////////////////////////////// CONFIGURACI�"N DE LA lwIP STACK (Ethernet) ////////////////////////////////////////

// Defines para hacer setup del reloj del sistema
#define SYSTICKHZ               100
```

```
#define SYSTICKMS               (1000 / SYSTICKHZ)

// Definiciones de las prioridades de interrupción. Los 3 bits superiores de estos valores son significativos con los valores
// más bajos indicando mayor prioridad de interrupción
#define SYSTICK_INT_PRIORITY    0x80
#define ETHERNET_INT_PRIORITY   0xC0

// Referencia a la función que inicializa el servidor "httpd" (HTTP Apache)
extern void httpd_init(void);

// Prototipo para la función handler CGI (Common Gateway Interface).
static char *HandlerCGI_Receptor_Datos(int32_t iIndex, int32_t i32NumParams, char *pcParam[], char *pcValue[]);

// Se registra la función handler anterior como el manejador CGI de manera que informa
// al servidor HTTPD de que cierta URI: "/recoger_datos.cgi" debe ser tratada a través del handler
// registrado si dicha URI se "activa". Es decir, al pulsar el botón en la web de "Solicitar Atención",
// tenemos un "action": "recoger_datos.cgi" que hará que el handler procese los datos de interés del
// formulario diseñado en dicha página web.
static const tCGI Config_URIs[] =
{
   { "/recoger_datos.cgi", (tCGIHandler)HandlerCGI_Receptor_Datos },
};

// Numero de URIS CGI individuales configuradas para este sistema.
#define Numero_de_URIS_configuradas    (sizeof(Config_URIs) / sizeof(tCGI))

// Una vez se completa la acción del handler CGI, se devuelve al navegador el siguiente archivo (URI de la página a cargar).
// Esto provoca que la página web se recargue en la pestaña del formulario, con este vacío, tras haber recogido
// los datos introducidos, quedando lista para volver a ser usada.
#define Respuesta_Handler_CGI    "/solicitud_cgi.ssi"

// Timeout para la solicitud de dirección mediante DHCP (en segundos).
#ifndef DHCP_EXPIRE_TIMER_SECS
#define DHCP_EXPIRE_TIMER_SECS  45
#endif

// Variable para recoger la dirección IP actual.
uint32_t Direccion_IP_actual;

// Variable para la frecuencia del reloj del sistema:
uint32_t Reloj;

// Rutina de error que es llamada si se encuentra un error durante el debug
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

// Handler CGI que es llamado cada vez que el navegador web solicita "recoger_datos.cgi" (con método GET)
static char *HandlerCGI_Receptor_Datos(int32_t iIndex, int32_t i32NumParams, char *pcParam[], char *pcValue[]){
```

```
    long lStringParam;  //Variable para guardar el indice si la cadena se encuentra (referida a su parámetro ("nombre","apellidos")), valdrá -1 si no se encuentra (no ocurre, la página obliga a rellenar todos los datos)
    urgente = FindCGIParameter("urgente", pcParam, i32NumParams);   // busca el parámetro "urgente" para ver si se le ha pasado desde el formulario de la web y guarda su índice,
                                                    // es decir, la variable urgente valdrá 0 (siempre es el primer parámetro del formulario) si se ha marcado la casilla de ¿es urgente?
                                                    // o -1 si no se ha marcado y por tanto ese parámetro no ha sido pasado a este handler.

    if (urgente == -1) strcpy(cadena_tipo_mensaje, cadena_no_urgente);  // Sabiendo si es urgente o no, copiamos un mensaje predefinido en "cadena_tipo_mensaje", que identifica si el mensaje es urgente o no
    else if (urgente == 0) strcpy(cadena_tipo_mensaje, cadena_urgente);

    // A partir de aquí, nos interesa recoger los mensajes introducidos por el usuario en los diferentes campos, para ello,
    // indicamos que se deben buscar los parámetros asociados a cada campo de texto, y al encontrarlos (Todos los datos son obligatorios
    // de rellenar (hemos forzado que la página web obligue a rellenarlos)), se decodifica el mensaje incluido en cada campo de texto que
    // se ha rellenado, guardándolo en diferentes variables de interés.
    // Por ejemplo para el nombre, tenemos en el archivo "solicitud_cgi.ssi" lo siguiente:
    // <input value="" maxlength="34" size="35" name="nombre" required>
    // de manera que al introducir texto (limitado a 34 carácteres), su campo "value" recogerá ese texto
    // y con la función "DecodeFormString" extraemos una string con dicha información, que almacenamos.
    lStringParam = FindCGIParameter("nombre", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Nombre, 50);

    lStringParam = FindCGIParameter("apellidos", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Apellidos, 50);

    lStringParam = FindCGIParameter("direccion", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Direccion, 50);

    lStringParam = FindCGIParameter("edad", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Edad, 50);

    lStringParam = FindCGIParameter("numerotelefono", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Telefono, 50);

    lStringParam = FindCGIParameter("TextoSupervisor", pcParam, i32NumParams);
    DecodeFormString(pcValue[lStringParam], Mensaje, 200);

    if (urgente == 0){      // Si el mensaje es urgente:
        // Copiamos los datos decodificados en strings en nuestra componente, indexada por el número de avisos
        // urgentes recibidos, del array de structs correspondiente a Registros Urgentes.
        strcpy(RegistroUrgente[mensajes_urgentes].Tipo_mensaje_s, cadena_tipo_mensaje);
        strcpy(RegistroUrgente[mensajes_urgentes].Hora_s, cadena_hora);
        strcpy(RegistroUrgente[mensajes_urgentes].Nombre_s, Nombre);
        strcpy(RegistroUrgente[mensajes_urgentes].Apellidos_s, Apellidos);
        strcpy(RegistroUrgente[mensajes_urgentes].Direccion_s, Direccion);
        strcpy(RegistroUrgente[mensajes_urgentes].Edad_s, Edad);
        strcpy(RegistroUrgente[mensajes_urgentes].Telefono_s, Telefono);
        strcpy(RegistroUrgente[mensajes_urgentes].Mensaje_s, Mensaje);

        mensajes_urgentes = mensajes_urgentes + 1;  //Se incrementa el número de mensajes urgentes recibidos en 1.
    }
    else{               // Si el mensaje no es urgente:
```

```
        // Copiamos los datos decodificados en strings en nuestra componente, indexada por el número de avisos
        // no urgentes recibidos, del array de structs correspondiente a Registros No Urgentes.
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Tipo_mensaje_s, cadena_tipo_mensaje);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Hora_s, cadena_hora);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Nombre_s, Nombre);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Apellidos_s, Apellidos);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Direccion_s, Direccion);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Edad_s, Edad);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Telefono_s, Telefono);
        strcpy(RegistroNoUrgente[mensajes_no_urgentes].Mensaje_s, Mensaje);

        mensajes_no_urgentes = mensajes_no_urgentes + 1; //Se incrementa el número de mensajes no urgentes
recibidos en 1.
    }

    cambio_estado = 1;  //Flag para entrar en el estado de alerta en la aplicación cada vez que recibimos un aviso
nuevo.

    return(Respuesta_Handler_CGI);  //Se devuelve la respuesta al servidor ("/solicitud_cgi.ssi"), recargando la
página
}

// Handler para la interrupción de SysTick.
void SysTickIntHandler(void)
{
    // Se llama al timer handler de lwIP:
    lwIPTimer(SYSTICKMS);
}


// Mostrar la dirección IP.
void DisplayIPAddress(uint32_t Addr)
{
    char pcBuf[16];
    // Convertir la dirección IP en una string.
    usprintf(pcBuf, "%d.%d.%d.%d", Addr & 0xff, (Addr >> 8) & 0xff,
            (Addr >> 16) & 0xff, (Addr >> 24) & 0xff);

    // Mostrar la string (dirección IP) por la UART.
    if(estado != 10) UARTprintf(pcBuf);

    //Se muestra por pantalla la dirección IP a introducir en el navegador para usar la Web
    if (estado == 10){
        ComColor(0,0,0);
        ComTXT(HSIZE/2, VSIZE/2-40, 23, OPT_CENTER,"Introduzca la direccion IP en su navegador:");
        ComColor(255,255,255);
        ComTXT(HSIZE/2, VSIZE/2-10, 23, OPT_CENTER, pcBuf);
    }

}

// Función requerida por la libería lwIP para dar soporte a funciones de temporización (Proceso de obtener la IP):
void lwIPHostTimerHandler(void)
{
    uint32_t Nueva_Direccion_IP;

    // Obtener la dirección IP.
    Nueva_Direccion_IP = lwIPLocalIPAddrGet();
```

```
    // Comprobar si la dirección IP ha cambiado.
    if(Nueva_Direccion_IP != Direccion_IP_actual)
    {
        // Comprobar si hay una dirección IP asignada.
        if(Nueva_Direccion_IP == 0xffffffff)
        {
            // Indicar que aún no hay enlace de conexión.
            UARTprintf("Esperando enlace\n");
        }
        else if(Nueva_Direccion_IP == 0)
        {
            // No hay dirección IP aún, indicar que DHCP está buscando.
            UARTprintf("Esperando dirección IP\n");
        }
        else
        {
            // Mostrar la nueva dirección IP obtenida.
            UARTprintf("Dirección IP: ");
            DisplayIPAddress(Nueva_Direccion_IP);
            UARTprintf("\n");
            UARTprintf("Abra un buscador e inserte la dirección IP\n");
            direccion_obtenida = 1; // Se activa esta variable para avanzar del modo "Conectando..." de la aplicación.
        }

        // Guardar la nueva dirección IP.
        Direccion_IP_actual = Nueva_Direccion_IP;
    }

    // Si no hay una nueva dirección IP:
    if((Nueva_Direccion_IP == 0) || (Nueva_Direccion_IP == 0xffffffff))
    {
        // No hacer nada y seguir esperando.
    }
}

//Prototipos de funciones:
void Funcion_nota1(void);
void Funcion_nota2(void);
void IntTimer0(void);
void IntTimer1(void);
void Pantalla_inicial(void);
void Pantalla_registro(void);
void Pantalla_alerta1(void);
void Pantalla_alerta2(void);
void Pantalla_borrado(void);
void Pantalla_detalle(void);
void Pantalla_Mensaje(void);
void ComProgress(int16_t x,int16_t y,int16_t w,int16_t h,uint16_t options,uint16_t val,uint16_t range);
int map(int valor, int entradaMin, int entradaMax, int salidaMin, int salidaMax);
void MuestraDetalles(int registro, int urgente);
void MuestraMensaje(int urgente, int no_urgente);
void Pantalla_ayuda(void);
void Reset_Cadenas_Mensaje(void);
void Comprueba_Pulsacion_Registros_Urgentes(void);
void Comprueba_Pulsacion_Registros_No_Urgentes(void);

int main(void)
{
```

```
    uint32_t User0, User1;      // Variables para configurar la dirección MAC del hardware del controlador Ethernet de la placa
    uint8_t pui8MACArray[8];

    // Se requiere que el oscilador principal esté activo, ya que esto es necesario para el PHY (Physical Layer / Circuito integrado)
    // El parámetro "SYSCTL_MOSC_HIGHFREQ" se utiliza cuando la frecuencia del cristal es >= 10 MHz.
    SysCtlMOSCConfigSet(SYSCTL_MOSC_HIGHFREQ);

    // Se configura el reloj del sistema para que use PLL a 120MHz
    Reloj = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    ////////////////////////////////////////////////////HABILITAMOS PUERTOS////////////////////////////////////////////////////////
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);    //Para la UART
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);    //Para el PWM que controla el servomotor
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);     //Módulo PWM 0

    ////////////////////////////////////////////////////CONFIGURACI?N DE LA UART////////////////////////////////////////////////////////
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);    //Se habilita la UART0
    GPIOPinConfigure(GPIO_PA0_U0RX);                //Pin A0 para recibir
    GPIOPinConfigure(GPIO_PA1_U0TX);                //Pin A1 para enviar
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);  //Pin A0 y A1 para el uso de la UART
    UARTStdioConfig(0, 115200, Reloj);  //A 115200 baudios

    ////////////////////////////////////////////////CONFIGURACI?N DEL PERIFÉRICO PWM////////////////////////////////////////////////////
    PWMClockSet(PWM0_BASE,PWM_SYSCLK_DIV_64);    //Al PWM le llega un reloj de 1.875MHz
    GPIOPinConfigure(GPIO_PG0_M0PWM4);          //Configurar el pin a PWM (módulo 0 - generador PWM 2)
    GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_0); //Pin PG0 para PWM
    // Configurar el PWM 0, contador descendente y sin sincronización (actualización automática)
    PWMGenConfigure(PWM0_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC); //Se usa el generador PWM 2
    PeriodoPWM=37499;                           // 50Hz a 1.875MHz
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, PeriodoPWM); //frec:50Hz
    PWMOutputState(PWM0_BASE, PWM_OUT_4_BIT , true);   //Habilita la salida 4, ya que se usará M0PWM4
    PWMGenEnable(PWM0_BASE, PWM_GEN_2);         //Habilita el generador PWM 2

    ////////////////////////////////// CONFIGURACI?N DE LA PANTALLA ////////////////////////////////////////////////////////
    HAL_Init_SPI(1, Reloj);  //Boosterpack a usar, Velocidad del MC
    Inicia_pantalla();       //Arranca la pantalla
    SysCtlDelay(Reloj/3);    //Pequeño delay para asegurarnos de que la pantalla está lista

    // Escritura en registros de los valores de calibración de la pantalla de 5.0"
    #ifdef VM800B50
       for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL5[i]);
    #endif

    ////////////////////////////////////////TIMER 0 PARA CONTAR SEGUNDOS (SE USA PARA LAS ANIMACIONES)////////////////////////////////////
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);        //Se usa el timer 0
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM); //T0 a 120MHz
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);      //T0 periódico y 32bits
    TimerLoadSet(TIMER0_BASE, TIMER_A, Reloj-1); //Se carga 1 segundo en la cuenta
    TimerIntRegister(TIMER0_BASE,TIMER_A,IntTimer0);      //"IntTimer0" nombre de la rutina de interrupción del timer 0
    IntEnable(INT_TIMER0A);                      //Habilita la interrupción del timer
```

```
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);      //El timeout (fin de la cuenta) provoca
interrupción en el timer 0

    ////////////////////////////////////////TIMER 1 PARA LA HORA (TIEMPO DE EJECUCI◆"N DEL
PROGRAMA)////////////////////////////////////////
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);         //Se usa el timer 1
    TimerClockSourceSet(TIMER1_BASE, TIMER_CLOCK_SYSTEM); //T1 a 120MHz
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);      //T1 periódico y 32bits
    TimerLoadSet(TIMER1_BASE, TIMER_A, Reloj-1); //Se carga 1 segundo en la cuenta
    TimerIntRegister(TIMER1_BASE,TIMER_A,IntTimer1);      //"IntTimer1" nombre de la rutina de interrupción del
timer 1
    IntEnable(INT_TIMER1A);                          //Habilita la interrupción del timer
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);      //El timeout (fin de la cuenta) provoca
interrupción en el timer 1
    TimerEnable(TIMER1_BASE, TIMER_A);                   //Habilita el timer 1 (empieza a contar)
    IntMasterEnable();                                //Habilita globalmente las interrupciones

    ////////////////////////////////////////////////////////////////////////////////////////////////////////////

    // Se configura SysTick para interrupciones periódicas
    MAP_SysTickPeriodSet(Reloj / SYSTICKHZ);
    MAP_SysTickEnable();
    MAP_SysTickIntEnable();

    // Configurar la dirección MAC hardware para que el controlador Ethernet filtre paquetes entrantes.
    // La dirección MAC se guarda en los registros USER0 y USER1.
    MAP_FlashUserGet(&User0, &User1);
    if((User0 == 0xffffffff) || (User1 == 0xffffffff))
    {
        // Hacer saber que no hay dirección MAC:
        UARTprintf("Dirección MAC no programada\n");

        while(1)
        {
        }
    }

    // Informar de que estamos buscando la IP (por UART):
    UARTprintf("Esperando IP\n");

    // Se convierte la dirección MAC separada en fragmentos 24/24 procedente de la Non Volatile RAM, en una
    // dirección MAC separada en fragmentos 32/16, que es el formato necesario para programar los registros
hardware.
    // Tras ello se programará la dirección MAC en los registros del controlador Ethernet.
    pui8MACArray[0] = ((User0 >>  0) & 0xff);
    pui8MACArray[1] = ((User0 >>  8) & 0xff);
    pui8MACArray[2] = ((User0 >> 16) & 0xff);
    pui8MACArray[3] = ((User1 >>  0) & 0xff);
    pui8MACArray[4] = ((User1 >>  8) & 0xff);
    pui8MACArray[5] = ((User1 >> 16) & 0xff);

    // Inicializar la librería lwIP, usando DHCP.
    lwIPInit(Reloj, pui8MACArray, 0, 0, 0, IPADDR_USE_DHCP);

    // Hacer Setup del servicio de Locator para el dispositivo (el microcontrolador)
    LocatorInit();
    LocatorMACAddrSet(pui8MACArray);
    LocatorAppTitleSet("EK-TM4C1294XL Servicio_Atencion");
```

```
    // Inicializar el servidor httpd (HTTP Apache)
    httpd_init();

    // Setear la prioridad de interrupciones:
    MAP_IntPrioritySet(INT_EMAC0, ETHERNET_INT_PRIORITY);
    MAP_IntPrioritySet(FAULT_SYSTICK, SYSTICK_INT_PRIORITY);

    // Pasar nuestro CGI handler al servidor HTTP con la configuración necesaria:
    http_set_cgi_handlers(Config_URIs, Numero_de_URIS_configuradas);

    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    // Bucle infinito, donde se ejecuta la FSM de la aplicación desarrollada, la interacción con la página web
    // se realiza mediante el handler asociado a la interrupcióne correspondiente.

    while(1){
        switch (estado){
        /************************************************************ ESTADO 0
*************************************************************/
        case 0: // Pantalla de inicio del programa donde permanecemos hasta que el protocolo DHCP nos
proporciona la dirección IP de la página web
            while (direccion_obtenida == 0){
                PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Min_pos);  //Posición derecha del servomotor
                Pantalla_inicial();                         //Definimos lo que aparecerá por pantalla
                Dibuja();                                 //Lo dibujamos
            }
            estado = 10;                                    //Una vez tenemos la dirección IP, pasamos al estado 10, donde
se muestra la IP por pantalla
            break;
        /************************************************************ ESTADO 1
*************************************************************/
        case 1: // Pantalla de registro con los avisos
            TimerEnable(TIMER0_BASE, TIMER_A);  //Se habilita el timer que habilitará la pulsación del aviso 5
urgente
            borrado_urgente = 0;
            borrado_no_urgente = 0;            //Reseteo de ciertas variables
            activa_estado_6 = 0;
            activa_estado_7 = 0;
            Pantalla_registro();               //Definimos lo que aparecerá por pantalla

            Dibuja();                         //Lo dibujamos

            Reset_Cadenas_Mensaje();    //Vaciamos las cadenas que se rellenan para mostrar por pantalla el
mensaje del aviso en caso de que tenga

            Lee_pantalla();            //Leemos si se ha pulsado la pantalla y guardamos dónde se ha pulsado en las
variables "POSX" y "POSY"

            // 5 campos de pulsaciones para ver la información de cada aviso (PARA URGENTE). Se diseña para
que no permita pulsar en cierto campo
            // si no hay mensaje recibido en dicho campo. Según el aviso pulsado, se le da valor a "IndiceUrgente", a
"registro" y pasamos al estado 6
            Comprueba_Pulsacion_Registros_Urgentes();

            // 5 campos de pulsaciones para ver la información de cada aviso (PARA NO URGENTE). Se diseña
para que no permita pulsar en cierto campo
            // si no hay mensaje recibido en dicho campo. Según el aviso pulsado, se le da valor a
"IndiceNoUrgente", a "registro" y pasamos al estado 7
```

```
        Comprueba_Pulsacion_Registros_No_Urgentes();

        if(activa_estado_1 == 2)  TimerDisable(TIMER0_BASE, TIMER_A);  //Pasados dos segundos,
habilitamos la pulsación del aviso 5 del tipo Urgente

        if (cambio_estado == 1){
            contador_estado2 = 0;   //Si nos llega un nuevo aviso procedente de la página web, pasamos al estado
2
            estado = 2;
        }
        break;

    /************************************************************ ESTADO 2
************************************************************/
    case 2: // Pantalla de alerta de aviso entrante
        TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitamos el timer 0 para realizar las animaciones de este
estado
        cambio_estado = 0;                //Reseteamos
        Funcion_nota1();                  //Se llama a la primera función que hará que la pantalla emita sonido
        if (flag_estado_alerta == 0){
            Pantalla_alerta1();
            Dibuja();                     //Se alterna entre dos colores de fondo de pantalla mostrando lo mismo por
pantalla
        }
        if (flag_estado_alerta == 1){
            Pantalla_alerta2();
            Dibuja();
        }
        if (contador_estado2 == 4){        //Cuando hemos pasado 4 segundos en este estado, volvemos al
estado 1 (pantalla de registro)
            flag_estado_alerta = 0;        //Reseteamos "flag_estado_alerta"
            FinNota();                    //Dejamos de emitir sonido
            TimerDisable(TIMER0_BASE, TIMER_A); //Se deshabilita el timer
            estado = 1;
        }
        break;

    /************************************************************ ESTADO 3
************************************************************/
    case 3: // Estado para borrar un aviso urgente cuando se pulsa en el estado 9 el botón "Enviar ayuda".
        // Adicionalmente se desplazan todos los avisos posteriores (avisos que llegaron luego al que se borra)
una posición hacia arriba en el array de structs de avisos urgentes
        for(i = IndiceUrgente; i < mensajes_urgentes - 1; i++){
            strcpy(RegistroUrgente[i].Tipo_mensaje_s, RegistroUrgente[i+1].Tipo_mensaje_s);
            strcpy(RegistroUrgente[i].Nombre_s,       RegistroUrgente[i+1].Nombre_s);
            strcpy(RegistroUrgente[i].Apellidos_s,    RegistroUrgente[i+1].Apellidos_s);
            strcpy(RegistroUrgente[i].Direccion_s,    RegistroUrgente[i+1].Direccion_s);
            strcpy(RegistroUrgente[i].Mensaje_s,      RegistroUrgente[i+1].Mensaje_s);
            strcpy(RegistroUrgente[i].Hora_s,         RegistroUrgente[i+1].Hora_s);
            strcpy(RegistroUrgente[i].Edad_s,         RegistroUrgente[i+1].Edad_s);
            strcpy(RegistroUrgente[i].Telefono_s,     RegistroUrgente[i+1].Telefono_s);
        }

        // Si el número de avisos urgentes es inferior a 5 (no hay encolados) y se borra uno, borramos el último
para que aparezca en blanco en el registro
        if(mensajes_urgentes <= 5){
            memset(RegistroUrgente[mensajes_urgentes-1].Tipo_mensaje_s, 0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Nombre_s,       0, 50);
```

```
            memset(RegistroUrgente[mensajes_urgentes-1].Apellidos_s,    0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Direccion_s,    0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Mensaje_s,      0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Hora_s,         0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Edad_s,         0, 50);
            memset(RegistroUrgente[mensajes_urgentes-1].Telefono_s,     0, 50);
        }

        mensajes_urgentes = mensajes_urgentes - 1;  //Decrementamos el número de mensajes urgentes
        contador_borrando = 0;                  //Se resetea "contador_borrando" que permite realizar la animación de la pantalla del estado 5
        estado = 1;                         //Se vuelve al registro
        break;

    /********************************************************** ESTADO 4 **********************************************************/
    case 4: // Estado para borrar un aviso no urgente cuando se pulsa en el estado 9 el botón "Enviar ayuda".
        // Adicionalmente se desplazan todos los avisos posteriores (avisos que llegaron luego al que se borra) una posición hacia arriba en el array de structs de avisos no urgentes
        for(i = IndiceNoUrgente; i < mensajes_no_urgentes - 1; i++){
            strcpy(RegistroNoUrgente[i].Tipo_mensaje_s, RegistroNoUrgente[i+1].Tipo_mensaje_s);
            strcpy(RegistroNoUrgente[i].Nombre_s,       RegistroNoUrgente[i+1].Nombre_s);
            strcpy(RegistroNoUrgente[i].Apellidos_s,    RegistroNoUrgente[i+1].Apellidos_s);
            strcpy(RegistroNoUrgente[i].Direccion_s,    RegistroNoUrgente[i+1].Direccion_s);
            strcpy(RegistroNoUrgente[i].Mensaje_s,      RegistroNoUrgente[i+1].Mensaje_s);
            strcpy(RegistroNoUrgente[i].Hora_s,         RegistroNoUrgente[i+1].Hora_s);
            strcpy(RegistroNoUrgente[i].Edad_s,         RegistroNoUrgente[i+1].Edad_s);
            strcpy(RegistroNoUrgente[i].Telefono_s,     RegistroNoUrgente[i+1].Telefono_s);
        }

        // Si el número de avisos urgentes es inferior a 5 (no hay encolados) y se borra uno, borramos el último para que aparezca en blanco en el registro
        if(mensajes_no_urgentes <= 5){
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Tipo_mensaje_s, 0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Nombre_s,       0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Apellidos_s,    0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Direccion_s,    0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Mensaje_s,      0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Hora_s,         0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Edad_s,         0, 50);
            memset(RegistroNoUrgente[mensajes_no_urgentes-1].Telefono_s,     0, 50);
        }

        mensajes_no_urgentes = mensajes_no_urgentes - 1;    //Decrementamos el número de mensajes no urgentes
        contador_borrando = 0;                      //Se resetea "contador_borrando" que permite realizar la animación de la pantalla del estado 5
        estado = 1;                             //Se vuelve al registro
        break;

    /********************************************************** ESTADO 5 **********************************************************/
    case 5: // Pantalla animada que indica que se está borrando el aviso para el que se acaba de enviar la ayuda
        TimerEnable(TIMER0_BASE, TIMER_A);      //Habilitamos el timer 0 para realizar las animaciones de este estado
        Pantalla_borrado();                 //Definimos lo que aparecerá por pantalla
        Dibuja();                       //Lo dibujamos
```

```
        if (contador_borrando == 3){          //Pasados 3 segundos desde que entramos en este estado, vamos al
estado 3 si el aviso que hemos atendido era urgente, o al estado 4 si era no urgente
            TimerDisable(TIMER0_BASE, TIMER_A); //Deshabilitamos el timer
            contador_borrando = 0;            //Reseteamos "contador_borrando"
            if(borrado_urgente == 1){
               estado = 3;
            }
            else if(borrado_no_urgente == 1){
               estado = 4;
            }
            else {}
        }
        break;

    /*********************************************************** ESTADO 6
************************************************************/
    case 6: // Estado en el que se muestran los detalles de un aviso urgente pulsado en el registro (nombre,
apellidos, edad...)
        TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitamos el timer 0 para habilitar la pulsación de los
botones de esta pantalla pasados dos segundos
        ver_mensaje_urgente = 0;          //Reseteo de variables
        volver_mensaje_urgente = 0;
        Pantalla_detalle();               //Definimos lo que aparecerá por pantalla
        MuestraDetalles(registro,1);      //Llamamos a la función "MuestraDetalles" que define la pantalla que
contiene toda la información del aviso a excepción del mensaje

        if(Boton_volver && activa_estado_6 >= 2){   //Si pulsamos el botón "Volver", nos devuelve al registro
            activa_estado_1 = 0;                //Reseteamos "activa_estado_1"
            estado = 1;                         //Volvemos a la pantalla de registros
        }

        if(Boton_ver_mensaje && activa_estado_6 >= 2){
            borrado_urgente = 1;
            volver_mensaje_urgente = 1;
            ver_mensaje_urgente = 1;
            flag_primera_linea = 0;             //Activamos y reseteamos variables que serán de utilidad en el
estado
            flag_segunda_linea = 0;             //en el que mostramos el mensaje del aviso urgente en caso de
tener
            flag_tercera_linea = 0;
            flag_cuarta_linea = 0;
            flag_quinta_linea = 0;
            i = 0;
            k = 0;
            activa_estado_9 = 0;
            terminador = 0;
            estado = 9;                         //Si pulsamos el botón "Ver mensaje" pasamos al estado 9
        }

        if(activa_estado_6 == 2) TimerDisable(TIMER0_BASE, TIMER_A);    //Pasados dos segundos en este
estado desactivamos el timer

        Dibuja();                           //Dibujamos la pantalla
        break;

    /*********************************************************** ESTADO 7
************************************************************/
```

```
        case 7: // Estado en el que se muestran los detalles de un aviso no urgente pulsado en el registro (nombre, apellidos, edad...)
            TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitamos el timer 0 para habilitar la pulsación de los botones de esta pantalla pasados dos segundos
            ver_mensaje_no_urgente = 0;         //Reseteo de variables
            volver_mensaje_no_urgente = 0;
            Pantalla_detalle();                 //Definimos lo que aparecerá por pantalla
            MuestraDetalles(registro,0);        //Llamamos a la función "MuestraDetalles" que define la pantalla que contiene toda la información del aviso a excepción del mensaje

            if(Boton_volver && activa_estado_7 >= 2){   //Si pulsamos el botón "Volver", nos devuelve al registro
                activa_estado_1 = 0;                    //Reseteamos "activa_estado_1"
                estado = 1;                             //Volvemos a la pantalla de registros
            }

            if(Boton_ver_mensaje && activa_estado_7 >= 2){
                borrado_no_urgente = 1;
                volver_mensaje_no_urgente = 1;
                ver_mensaje_no_urgente = 1;
                flag_primera_linea = 0;                 //Activamos y reseteamos variables que serán de utilidad en el estado
                flag_segunda_linea = 0;                 //en el que mostramos el mensaje del aviso no urgente en caso de tener
                flag_tercera_linea = 0;
                flag_cuarta_linea = 0;
                flag_quinta_linea = 0;
                i = 0;
                k = 0;
                activa_estado_9 = 0;
                terminador = 0;
                estado = 9;                             //Si pulsamos el botón "Ver mensaje" pasamos al estado 9
            }

            if(activa_estado_7 == 2) TimerDisable(TIMER0_BASE, TIMER_A);    //Pasados dos segundos en este estado desactivamos el timer

            Dibuja();                               //Dibujamos la pantalla
            break;

        /************************************************************ ESTADO 8 ************************************************************/
        case 8: // Este estado emula el envío de ayuda al lugar que indica el aviso atendido. Puede ser por ejemplo la apertura de una compuerta que permite salir a la ambulancia ante un aviso de una persona a la que le ha dado un infarto
            TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitamos el timer 0 para realizar las animaciones de este estado
            Funcion_nota2();                    //Se llama a la segunda función que hará que la pantalla emita sonido
            if (contador_estado8 == 0){         //Al entrar en este estado movemos el servomotor a la posición izquierda
                PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);
                Pantalla_ayuda();               //Definimos lo que aparecerá por pantalla (animación de un cochecito moviéndose)
                Dibuja();                       //Dibujamos la pantalla
            }
            if (contador_estado8 == 1){
                Pantalla_ayuda();               //A cada segundo que pasa, el cochecito avanza hacia la derecha de la pantalla
                Dibuja();
```

```
        }
        if (contador_estado8 == 2){
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 3){
            Pantalla_ayuda();
            Dibuja();
        }
        if (contador_estado8 == 4){                     //Pasados 4 segundos
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Min_pos);    //Llevamos el servomotor a su posición
de reposo
            FinNota();                                  //La pantalla deja de emitir sonido
            TimerDisable(TIMER0_BASE, TIMER_A);             //Deshabilitamos el timer 0
            contador_borrando = 0;                      //Reseteamos "contador_borrando"
            estado = 5;                                 //Pasamos al estado donde se anima el borrado del aviso
atendido
        }
        break;

    /************************************************************ ESTADO 9
************************************************************/
    case 9: //Estado en el que se muestra el mensaje del aviso en caso de contener
        TimerEnable(TIMER0_BASE, TIMER_A);          //Habilitamos el timer 0 para habilitar la pulsación de los
botones de esta pantalla pasados dos segundos
        Pantalla_Mensaje();                     //Definimos lo que aparecerá por pantalla

        //Llamamos a la función "MuestraMensaje", esta define el valor de las 5 cadenas que imprimen
        //el mensaje ("cadena_primera_linea, "cadena_segunda_linea"...) y las saca por la pantalla
        MuestraMensaje(ver_mensaje_urgente, ver_mensaje_no_urgente);

        if(Boton_volver && activa_estado_9 >= 2){       //Al pulsar el botón "Volver", según qué tipo de aviso
estábamos viendo
            if (volver_mensaje_urgente == 1){           //volvemos al estado 6 (caso urgente) o al estado 7 (caso no
urgente)
                estado = 6;
            }
            else if (volver_mensaje_no_urgente == 1) {
                estado = 7;
            }
        }

        if(Boton_enviar_ayuda && activa_estado_9 >= 2){ //Si pulsamos el botón "Enviar ayuda" pasamos al
estado 8 de envío de ayuda
            contador_estado8 = 0;                   //Reseteamos "contador_estado8"
            estado = 8;
        }

        if(activa_estado_9 == 2){                   //Pasados 2 segundos
            activa_estado_6 = 0;                    //Reseteamos variables
            activa_estado_7 = 0;
            TimerDisable(TIMER0_BASE, TIMER_A);         //Desactivamos el timer 0
        }

        Dibuja();                               //Dibujamos la pantalla
        break;
    /************************************************************ ESTADO 10
************************************************************/
```

```
        case 10:    // Estado que muestra la dirección IP a introducir en el navegador para que no sea necesario a
acudir a la UART para verla
            Nueva_pantalla(51,222,204);
            DisplayIPAddress(Direccion_IP_actual);  // Se pinta la dirección IP en la pantalla
            if(Boton_IP_introducida){              // Una vez introducida la IP en el navegador, se pulsa este botón para
hacer uso de la aplicación de manera habitual.
                estado = 1;
            }
            Dibuja();
            break;
        }
    }
}


////////////////////////////////////////////////////////// FUNCIONES ////////////////////////////////////////////////////////////

// Función para emitir sonido en la pantalla de alerta (estado 2)
void Funcion_nota1(){
    VolNota(100);
    TocaNota(S_BEEP, N_SI);
}

// Función para emitir sonido en la pantalla de envío de ayuda (estado 8)
void Funcion_nota2(){
    VolNota(30);
    TocaNota(S_XILO, N_SI);
}

void IntTimer0(void) //Cuenta segundos al entrar en los estados 1, 2, 5, 6, 7, 8 y 9
{
  TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT); // Borra flag

  contador_estado2 = contador_estado2 + 1;
  contador_borrando = contador_borrando + 1;
  contador_estado8 = contador_estado8 + 1;
  activa_estado_9 = activa_estado_9 + 1;      //Se incrementan las variables necesarias en una unidad
  activa_estado_6 = activa_estado_6 + 1;
  activa_estado_7 = activa_estado_7 + 1;
  activa_estado_1 = activa_estado_1 + 1;

  flag_estado_alerta = !flag_estado_alerta;    //Se conmuta el valor de "flag_estado_alerta" entre '0' y '1' para la
pantalla de alerta (estado 2)
}

void IntTimer1(void) //Cuenta el tiempo de ejecución del programa desde su arranque
{
  TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // Borra flag

  cont_circulos = cont_circulos + 1;            // Variable que anima los puntitos de la pantalla inicial (estado 0)

  if (cont_circulos == 4) cont_circulos = 0;     // Si toma valor 4, se resetea

  segundos++;                                 // Incrementa variables de segundos en 1
 if(segundos == 60)                            // Si se llega a 60 segundos:
 {
    minutos++;                               // Se incrementa minutos en 1
    segundos = 0;                             // Se resetea segundos a 0
```

```
  }
  if(minutos == 60)                       // Si se llega a 60 minutos:
  {
     horas++;                            // Se incrementa horas en 1
     minutos = 0;                         // Se resetea minutos a 0
  }
  if(horas == 100)                        // Si horas llega a 100 (parece sensato contar más de 24h por si el
dispositivo se deja encendido)
  {
     horas = 0;                          // Se resetea horas a 0
  }

  sprintf(cadena_hora, "%02d:%02d:%02d",horas, minutos, segundos); // Se define la cadena que se usará para
rellenar el campo "Hora_s" conforme llegan los avisos
}

// Función que define la pantalla inicial (estado 0)
void Pantalla_inicial(void)
{
    Nueva_pantalla(31,58,208);
    ComColor(255,255,255);
    ComTXT(HSIZE/2, 2, 24, OPT_CENTERX,"Atencion de personas");
    ComTXT(HSIZE/2, 27, 24, OPT_CENTERX,"con necesidades especiales");

    ComCirculo(HSIZE/2, 144, 67);

    ComColor(255, 0, 0);
    ComRect(HSIZE/2-50, 129, HSIZE/2+50, 161, true);
    ComRect(HSIZE/2-15, 94, HSIZE/2+15, 196, true);

    ComColor(0, 0, 0);
    ComTXT(HSIZE/2-38, VSIZE/2+10, 24, OPT_CENTER,"F");
    ComTXT(HSIZE/2+38, VSIZE/2+10, 24, OPT_CENTER,"J");
    ComTXT(HSIZE/2, 105, 24, OPT_CENTER,"J");
    ComTXT(HSIZE/2, 181, 24, OPT_CENTER,"D");

    ComColor(255, 255, 255);
    ComTXT(HSIZE/2-25, 225, 24, OPT_CENTERX,"Conectando");

    if (cont_circulos == 1) ComCirculo(HSIZE/2+50, 244, 5); //Según el valor de "cont_círculos", los puntitos
aparecen y desaparecen
    else if (cont_circulos == 2){
       ComCirculo(HSIZE/2+50, 244, 5);
       ComCirculo(HSIZE/2+65, 244, 5);
    }
    else if (cont_circulos == 3){
       ComCirculo(HSIZE/2+50, 244, 5);
       ComCirculo(HSIZE/2+65, 244, 5);
       ComCirculo(HSIZE/2+80, 244, 5);
    }
    else {}
}

// Función que define la pantalla del registro (estado 1) con los 5 primeros avisos de cada tipo, tanto urgentes
como no urgentes
void Pantalla_registro(void)
{
    Nueva_pantalla(51,222,204);
```

```
    ComColor(0,0,0);
    ComTXT(7*HSIZE/8, VSIZE/20, 23, OPT_CENTER, cadena_hora);
    ComTXT(HSIZE/13, VSIZE/20, 23, OPT_CENTERY, "Registro de mensajes");

    ComColor(255,0,0);
    ComTXT(HSIZE/4+15, VSIZE/6, 23, OPT_CENTER, "Urgente");
    ComTXT(HSIZE/6-20, VSIZE/4+9, 23, OPT_CENTER, "N");
    ComColor(59,32,103);
    ComTXT(3*HSIZE/9-1, VSIZE/4+9, 23, OPT_CENTER, "Hora llegada");

    ComColor(255,255,255);
    ComTXT(HSIZE/6-20, 2*VSIZE/5+2, 23, OPT_CENTER, "1"); //(59,98)
    ComTXT(3*HSIZE/9-1, 2*VSIZE/5+2, 23, OPT_CENTER, RegistroUrgente[0].Hora_s);
    ComTXT(HSIZE/6-20, VSIZE/2+5, 23, OPT_CENTER, "2");
    ComTXT(3*HSIZE/9-1, VSIZE/2+5, 23, OPT_CENTER, RegistroUrgente[1].Hora_s);
    ComTXT(HSIZE/6-20, 4*VSIZE/6-7, 23, OPT_CENTER, "3");
    ComTXT(3*HSIZE/9-1, 4*VSIZE/6-7, 23, OPT_CENTER, RegistroUrgente[2].Hora_s);
    ComTXT(HSIZE/6-20, 3*VSIZE/4+2, 23, OPT_CENTER, "4");
    ComTXT(3*HSIZE/9-1, 3*VSIZE/4+2, 23, OPT_CENTER, RegistroUrgente[3].Hora_s);
    ComTXT(HSIZE/6-20, 3*VSIZE/4+32, 23, OPT_CENTER, "5");
    ComTXT(3*HSIZE/9-1, 3*VSIZE/4+32, 23, OPT_CENTER, RegistroUrgente[4].Hora_s);

    ComColor(255,197,0);
    ComTXT(3*HSIZE/4-55, VSIZE/6, 23, OPT_CENTERY, "No urgente");
    ComTXT(HSIZE/2+20, VSIZE/4+9, 23, OPT_CENTER, "N");
    ComColor(59,32,103);
    ComTXT(7*HSIZE/9-6, VSIZE/4+9, 23, OPT_CENTER, "Hora llegada");

    ComColor(255,255,255);
    ComTXT(HSIZE/2+20, 2*VSIZE/5+2, 23, OPT_CENTER, "1");
    ComTXT(7*HSIZE/9-6, 2*VSIZE/5+2, 23, OPT_CENTER, RegistroNoUrgente[0].Hora_s);
    ComTXT(HSIZE/2+20, VSIZE/2+5, 23, OPT_CENTER, "2");
    ComTXT(7*HSIZE/9-6, VSIZE/2+5, 23, OPT_CENTER, RegistroNoUrgente[1].Hora_s);
    ComTXT(HSIZE/2+20, 4*VSIZE/6-7, 23, OPT_CENTER, "3");
    ComTXT(7*HSIZE/9-6, 4*VSIZE/6-7, 23, OPT_CENTER, RegistroNoUrgente[2].Hora_s);
    ComTXT(HSIZE/2+20, 3*VSIZE/4+2, 23, OPT_CENTER, "4");
    ComTXT(7*HSIZE/9-6, 3*VSIZE/4+2, 23, OPT_CENTER, RegistroNoUrgente[3].Hora_s);
    ComTXT(HSIZE/2+20, 3*VSIZE/4+32, 23, OPT_CENTER, "5");
    ComTXT(7*HSIZE/9-6, 3*VSIZE/4+32, 23, OPT_CENTER, RegistroNoUrgente[4].Hora_s);

    ComColor(0,0,255);
    ComRect(HSIZE/13, VSIZE/9, HSIZE-23, 8*VSIZE/10+36, false);
    ComLine(HSIZE/2, VSIZE/9, HSIZE/2, 8*VSIZE/10+36, 1);
    ComRect(HSIZE/13, VSIZE/5+7, HSIZE-23, VSIZE/3+3, false);
    ComRect(HSIZE/13, VSIZE/2-11, HSIZE-23, VSIZE/2+21, false);
    ComRect(HSIZE/13, 7*VSIZE/10-1, HSIZE-23, 8*VSIZE/10+3, false);
    ComRect(HSIZE/6+2, VSIZE/5+7, 4*HSIZE/7+7, 8*VSIZE/10+36, false);
    ComRect(HSIZE/6+2, 8*VSIZE/10+3, 4*HSIZE/7+7, 8*VSIZE/10+36, false);
}

// Función que define la primera pantalla del estado de alerta (estado 2)
void Pantalla_alerta1(void)
{
    Nueva_pantalla(255,0,255);
    ComColor(0,0,0);
    ComTXT(HSIZE/2, 25, 25, OPT_CENTERX,"!SOLICITUD DE");
    ComTXT(HSIZE/2, 60, 25, OPT_CENTERX,"AYUDA");
    ComTXT(HSIZE/2, 95, 25, OPT_CENTERX,"ENTRANTE!");
```

```
    ComColor(255,255,255);
    ComCirculo(HSIZE/2-25, 170, 10);
    ComRect(HSIZE/2-30, 190, HSIZE/2-20, 230, 1);
    ComRect(HSIZE/2+10, 160, HSIZE/2+20, 200, 1);
    ComCirculo(HSIZE/2+15, 220, 10);
}

// Función que define la segunda pantalla del estado de alerta (estado 2), tan solo varía el color del fondo con respecto a la primera pantalla
void Pantalla_alerta2(void)
{
    Nueva_pantalla(57,208,31);
    ComColor(0,0,0);
    ComTXT(HSIZE/2, 25, 25, OPT_CENTERX,"!SOLICITUD DE");
    ComTXT(HSIZE/2, 60, 25, OPT_CENTERX,"AYUDA");
    ComTXT(HSIZE/2, 95, 25, OPT_CENTERX,"ENTRANTE!");
    ComColor(255,255,255);
    ComCirculo(HSIZE/2-25, 170, 10);
    ComRect(HSIZE/2-30, 190, HSIZE/2-20, 230, 1);
    ComRect(HSIZE/2+10, 160, HSIZE/2+20, 200, 1);
    ComCirculo(HSIZE/2+15, 220, 10);
}

// Función que define la pantalla del estado de borrado (estado 5)
void Pantalla_borrado(void)
{
    Nueva_pantalla(31,58,208);
    ComColor(255, 255, 255);
    ComTXT(HSIZE/2-30, 205, 24, OPT_CENTERX,"Borrando registro");

    ComCirculo(HSIZE/2+80, 224, 5);
    ComCirculo(HSIZE/2+95, 224, 5);
    ComCirculo(HSIZE/2+110, 224, 5);

    //Barra de progreso de borrando. Se rellena (1/3) cada segundo
    ComProgress(HSIZE/6,VSIZE/2,300,30,OPT_FLAT,map(contador_borrando, 0, 3, 0, 100), 100);
}

// Función que define el color del fondo y el título en los estados de los detalles del mensaje (estado 6/7)
void Pantalla_detalle(void)
{
    Nueva_pantalla(51,222,204);
    ComColor(255, 255, 255);
    ComTXT(HSIZE/2+10, VSIZE/9, 24, OPT_CENTER,"Detalles del mensaje");
}

// Función que define el color del fondo y el título en el estado en el que se muestra el mensaje del aviso consultado (estado 9)
void Pantalla_Mensaje(void)
{
    Nueva_pantalla(51,222,204);
    ComColor(0, 0, 0);
    ComTXT(HSIZE/2, VSIZE/9, 24, OPT_CENTER,"Mensaje adjunto:");
}

// Función que define la barra de progreso del estado de borrado (estado 5)
void ComProgress(int16_t x,int16_t y,int16_t w,int16_t h,uint16_t options,uint16_t val,uint16_t range)
{
```

```
    EscribeRam32(CMD_PROGRESS);
    EscribeRam16(x);
    EscribeRam16(y);
    EscribeRam16(w);          //8 parametros de 16bits, 16 bytes: BIEN
    EscribeRam16(h);          //si fuesen 7 parametros de 16 bits, hay que mandar un ultimo EscribeRam16 con ceros.
    EscribeRam16(options);    //En este caso sí hay que mandarle los ceros al final
    EscribeRam16(val);
    EscribeRam16(range);
    EscribeRam16(0);
}

// Función de mapeo
int map(int valor, int entradaMin, int entradaMax, int salidaMin, int salidaMax)   //Por las características de la progress bar de "borrando..." se puede mapear con enteros
{
    return ((((valor-entradaMin)*(salidaMax-salidaMin))/(entradaMax-entradaMin))+salidaMin);
}

// Función que muestra todos los campos de un aviso a excepción del mensaje
void MuestraDetalles(int registro, int urgente)
{
    // Conociendo el valor de "urgente" y "registro" se sabe de qué aviso de los 10 posibles a consultar debemos imprimir su información
    if(urgente == 1){
        ComColor(0,0,0);
        ComTXT(HSIZE/20, 2*VSIZE/9, 22, OPT_CENTERY, "Tipo de mensaje: ");
        ComTXT(HSIZE/20, 3*VSIZE/9, 22, OPT_CENTERY, "Hora: ");
        ComTXT(HSIZE/20, 4*VSIZE/9, 22, OPT_CENTERY, "Nombre: ");
        ComTXT(HSIZE/20, 5*VSIZE/9, 22, OPT_CENTERY, "Apellidos: ");
        ComTXT(HSIZE/20, 6*VSIZE/9, 22, OPT_CENTERY, "Direccion: ");
        ComTXT(HSIZE/20, 7*VSIZE/9, 22, OPT_CENTERY, "Edad: ");
        ComTXT(HSIZE/2, 7*VSIZE/9, 22, OPT_CENTERY, "Telefono: ");
        ComColor(255,255,255);
        ComTXT(HSIZE/20+135, 2*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Tipo_mensaje_s);
        ComTXT(HSIZE/20+45, 3*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Hora_s);
        ComTXT(HSIZE/20+67, 4*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Nombre_s);
        ComTXT(HSIZE/20+73, 5*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Apellidos_s);
        ComTXT(HSIZE/20+75, 6*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Direccion_s);
        ComTXT(HSIZE/20+47, 7*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Edad_s);
        ComTXT(HSIZE/2+72, 7*VSIZE/9, 22, OPT_CENTERY,RegistroUrgente[registro-1].Telefono_s);
    }
    else{
        ComColor(0,0,0);
        ComTXT(HSIZE/20, 2*VSIZE/9, 22, OPT_CENTERY, "Tipo de mensaje: ");
        ComTXT(HSIZE/20, 3*VSIZE/9, 22, OPT_CENTERY, "Hora: ");
        ComTXT(HSIZE/20, 4*VSIZE/9, 22, OPT_CENTERY, "Nombre: ");
        ComTXT(HSIZE/20, 5*VSIZE/9, 22, OPT_CENTERY, "Apellidos: ");
        ComTXT(HSIZE/20, 6*VSIZE/9, 22, OPT_CENTERY, "Direccion: ");
        ComTXT(HSIZE/20, 7*VSIZE/9, 22, OPT_CENTERY, "Edad: ");
        ComTXT(HSIZE/2, 7*VSIZE/9, 22, OPT_CENTERY, "Telefono: ");
        ComColor(255,255,255);
        ComTXT(HSIZE/20+135, 2*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Tipo_mensaje_s);
        ComTXT(HSIZE/20+45, 3*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Hora_s);
        ComTXT(HSIZE/20+67, 4*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Nombre_s);
        ComTXT(HSIZE/20+73, 5*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Apellidos_s);
        ComTXT(HSIZE/20+75, 6*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Direccion_s);
        ComTXT(HSIZE/20+47, 7*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Edad_s);
```

```
        ComTXT(HSIZE/2+72, 7*VSIZE/9, 22, OPT_CENTERY,RegistroNoUrgente[registro-1].Telefono_s);
    }
}

// Función que se encarga de separar el mensaje del aviso de tamaño 200, en 5 cadenas para se pueda
visualizar de
// manera correcta por la pantalla sin que haya texto que sobrepase los límites de la pantalla en el estado 9
void MuestraMensaje(int ver_urgente, int ver_no_urgente){

    //Sabiendo el valor de "registro" y sabiendo si venimos del estado 6 (aviso urgente) o
    //del 7 (aviso no urgente) guardamos en "cadena_mensaje" el mensaje correspondiente
    if(ver_urgente == 1){
        strcpy(cadena_mensaje, RegistroUrgente[registro-1].Mensaje_s);
    }

    if(ver_no_urgente == 1){
        strcpy(cadena_mensaje, RegistroNoUrgente[registro-1].Mensaje_s);
    }

    //Si el primer carácter del mensaje es una letra, significa que no está vacío el mensaje
    if ((cadena_mensaje[0]>='a'&& cadena_mensaje[0]<='z') || (cadena_mensaje[0]>='A' &&
cadena_mensaje[0]<='Z')){
        //Búsqueda de la longitud de la cadena del mensaje envíado (se busca el terminador)
        while (i <= 199){
            if (cadena_mensaje[i] == '\0' && terminador == 0) {
                terminador = i;     //Se guarda el valor del termindor del mensaje en "terminador"
                break;
            }
            i = i + 1;
        }

        //Bucle desde el comienzo de la cadena hasta el terminador
        while(k <= terminador){

            /************************************************** RENGL?"N 1
************************************************************/
            //Se separa el primer renglón del mensaje entrante que cabe en la pantalla (en torno al
            //carácter 40 se llega al lateral derecho de la pantalla con un cierto margen de seguridad)
            if(terminador >= 0 && terminador <= 40){            //Si el mensaje cabe en la primera línea
                for (j = 0; j<=terminador; j++){
                    cadena_primera_linea[j] = cadena_mensaje[j]; //Se guarda el mensaje en "cadena_primera_linea"
                }
            }
            else{                                       //Si el mensaje no cabe en la primera línea
                if(k >= 40 && cadena_mensaje[k] == ' ' && flag_primera_linea == 0){
                    for (j = 0; j<=k; j++){
                        cadena_primera_linea[j] = cadena_mensaje[j]; //Se guarda el mensaje en "cadena_primera_linea"
hasta el primer
                    }                                   //espacio encontrado del carácter número 40 en adelante
                    caracter = k;               //Almacenamos en "caracter" el índice desde el que debemos empezar a
guardar en la siguiente línea
                    flag_primera_linea = 1;    //Activamos "flag_primera_linea" para realizar el almacenamiento del
mensaje de manera secuencial
                }
            }

            /************************************************** RENGL?"N 2
************************************************************/
```

```
        //Se separa el segundo renglón del mensaje entrante que cabe en la pantalla (en torno al
        //carácter 80 se llega al lateral derecho de la pantalla con un cierto margen de seguridad)
        if(flag_primera_linea == 1){    //Esperamos a que se hayan realizado las operaciones necesarias para
rellenar la primera línea
            if(terminador > 40 && terminador <= 80){        //Si el mensaje termina en la segunda línea
                for (j = caracter+1; j<=terminador; j++){
                    cadena_segunda_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_segunda_linea"
                }
            }
            else{                                   //Si el mensaje no termina en la segunda línea
                if(k >= 80 && cadena_mensaje[k] == ' ' && flag_segunda_linea == 0){
                    for (j = caracter+1; j<=k; j++){
                        cadena_segunda_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_segunda_linea" hasta el primer
                    }                                       //espacio encontrado del carácter número 80 en adelante
                    caracter = k;           //Almacenamos en "caracter" el índice desde el que debemos empezar a
guardar en la siguiente línea
                    flag_segunda_linea = 1; //Activamos "flag_segunda_linea" para realizar el almacenamiento del
mensaje de manera secuencial
                }
            }
        }

        /************************************************* RENGL?"N 3
************************************************************/
        //Se separa el tercer renglón del mensaje entrante que cabe en la pantalla (en torno al
        //carácter 120 se llega al lateral derecho de la pantalla con un cierto margen de seguridad)
        if(flag_segunda_linea == 1){    //Esperamos a que se hayan realizado las operaciones necesarias para
rellenar la segunda línea
            if(terminador > 80 && terminador <= 120){       //Si el mensaje termina en la tercera línea
                for (j = caracter+1; j<=terminador; j++){
                    cadena_tercera_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_tercera_linea"
                }
            }
            else{                                   //Si el mensaje no termina en la tercera línea
                if(k >= 120 && cadena_mensaje[k] == ' ' && flag_tercera_linea == 0){
                    for (j = caracter+1; j<=k; j++){
                        cadena_tercera_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_tercera_linea" hasta el primer
                    }                                       //espacio encontrado del carácter número 120 en adelante
                    caracter = k;           //Almacenamos en "caracter" el índice desde el que debemos empezar a
guardar en la siguiente línea
                    flag_tercera_linea = 1; //Activamos "flag_tercera_linea" para realizar el almacenamiento del
mensaje de manera secuencial
                }
            }
        }

        /************************************************* RENGL?"N 4
************************************************************/
        //Se separa el cuarto renglón del mensaje entrante que cabe en la pantalla (en torno al
        //carácter 160 se llega al lateral derecho de la pantalla con un cierto margen de seguridad)
        if(flag_tercera_linea == 1){    //Esperamos a que se hayan realizado las operaciones necesarias para
rellenar la tercera línea
            if(terminador > 120 && terminador <= 160){      //Si el mensaje termina en la cuarta línea
                for (j = caracter+1; j<=terminador; j++){
```

```
                    cadena_cuarta_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_cuarta_linea"
                }
            }
            else{                                   //Si el mensaje no termina en la cuarta línea
                if(k >= 160 && cadena_mensaje[k] == ' ' && flag_cuarta_linea == 0){
                    for (j = caracter+1; j<=k; j++){
                        cadena_cuarta_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_cuarta_linea" hasta el primer
                    }                                       //espacio encontrado del carácter número 120 en adelante
                    caracter = k;          //Almacenamos en "caracter" el índice desde el que debemos empezar a
guardar en la siguiente línea
                    flag_cuarta_linea = 1;  //Activamos "flag_cuarta_linea" para realizar el almacenamiento del
mensaje de manera secuencial
                }
            }
        }

        /************************************************** RENGL?N 5
**********************************************************/
        //Se separa el quinto renglón del mensaje entrante.
        //Aquí ya debemos guardar desde donde se quedó la cuarta línea hasta el terminador
        if(flag_cuarta_linea == 1){     //Esperamos a que se hayan realizado las operaciones necesarias para
rellenar la cuarta línea
            for (j = caracter+1; j<=terminador; j++){
                cadena_quinta_linea[j-caracter-1] = cadena_mensaje[j]; //Se guarda el mensaje en
"cadena_quinta_linea" hasta el final
            }
        }
        k = k + 1;
    }
  }

  // Si el aviso no contiene mensaje
  else {
    ComColor(255, 255, 255);
    ComTXT(HSIZE/2, VSIZE/2, 23, OPT_CENTER, "No hay mensaje adjunto");
  }

  ComColor(255, 255, 255);
  ComTXT(HSIZE/10, VSIZE/5+10, 23, OPT_CENTERY, cadena_primera_linea);
  ComTXT(HSIZE/10, 2*VSIZE/5-10, 23, OPT_CENTERY, cadena_segunda_linea);
  ComTXT(HSIZE/10, 3*VSIZE/5-30, 23, OPT_CENTERY, cadena_tercera_linea);
  ComTXT(HSIZE/10, 6*VSIZE/10+4, 23, OPT_CENTERY, cadena_cuarta_linea);
  ComTXT(HSIZE/10, 7*VSIZE/10+11, 23, OPT_CENTERY, cadena_quinta_linea);
}

// Función que define la pantalla del estado de envío de ayuda (estado 8)
void Pantalla_ayuda(void){
  Nueva_pantalla(51,222,204);
  ComColor(255, 255, 255);
  ComTXT(HSIZE/2-30, VSIZE/2-20, 24, OPT_CENTER, "Enviando ayuda");

  ComCirculo(HSIZE/2+67, VSIZE/2-13, 5);
  ComCirculo(HSIZE/2+82, VSIZE/2-13, 5);
  ComCirculo(HSIZE/2+97, VSIZE/2-13, 5);

  //Según el valor de "contador_estado8", se anima el movimiento del cochecito por la pantalla
```

```
    ComColor(209,170,0);
    ComRect(contador_estado8*HSIZE/7+HSIZE/10-6, VSIZE/2+20, contador_estado8*HSIZE/7+3.25*HSIZE/10,
2.25*VSIZE/3+10, true);
    ComColor(0,0,0);
    ComCirculo(contador_estado8*HSIZE/7+HSIZE/10+12, 2.25*VSIZE/3+13, 15);
    ComCirculo(contador_estado8*HSIZE/7+3.25*HSIZE/10-18, 2.25*VSIZE/3+13, 15);
}

// Función que permite vaciar las cadenas que muestran el mensaje adjunto, si lo hubiese
void Reset_Cadenas_Mensaje(void){
    memset(cadena_primera_linea, 0, 50);
    memset(cadena_segunda_linea, 0, 50);
    memset(cadena_tercera_linea, 0, 50);    //Vaciamos las cadenas que se rellenan para mostrar por pantalla el
mensaje del aviso en caso de que tenga
    memset(cadena_cuarta_linea, 0, 50);
    memset(cadena_quinta_linea, 0, 50);
    memset(cadena_mensaje, 0, 50);
}

// Función que permite comprobar si se ha pulsado alguno de los registros Urgentes, y cuál de ellos ha sido el
que se ha pulsado
void Comprueba_Pulsacion_Registros_Urgentes(void){
    if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/3+3 && POSY <= VSIZE/2-11 &&
mensajes_urgentes > 0){
        registro = 1;
        IndiceUrgente = 0;
        estado = 6;
    }
    if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/2-11 && POSY <= VSIZE/2+21 &&
mensajes_urgentes > 1){
        registro = 2;
        IndiceUrgente = 1;
        estado = 6;
    }
    if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > VSIZE/2+21 && POSY <= 7*VSIZE/10-1 &&
mensajes_urgentes > 2){
        registro = 3;
        IndiceUrgente = 2;
        estado = 6;
    }
    if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > 7*VSIZE/10-1 && POSY <= 8*VSIZE/10+3 &&
mensajes_urgentes > 3){
        registro = 4;
        IndiceUrgente = 3;
        estado = 6;
    }
    if(POSX > HSIZE/13 && POSX <= HSIZE/2 && POSY > 8*VSIZE/10+3 && POSY <= 8*VSIZE/10+36 &&
mensajes_urgentes > 4 && activa_estado_1 >= 2){
        registro = 5;
        IndiceUrgente = 4;
        estado = 6;
    }
}

// Función que permite comprobar si se ha pulsado alguno de los registros NO Urgentes, y cuál de ellos ha sido
el que se ha pulsado
void Comprueba_Pulsacion_Registros_No_Urgentes(void){
```

```
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/3+3 && POSY <= VSIZE/2-11 &&
mensajes_no_urgentes > 0){
        registro = 1;
        IndiceNoUrgente = 0;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/2-11 && POSY <= VSIZE/2+21 &&
mensajes_no_urgentes > 1){
        registro = 2;
        IndiceNoUrgente = 1;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > VSIZE/2+21 && POSY <= 7*VSIZE/10-1 &&
mensajes_no_urgentes > 2){
        registro = 3;
        IndiceNoUrgente = 2;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > 7*VSIZE/10-1 && POSY <= 8*VSIZE/10+3 &&
mensajes_no_urgentes > 3){
        registro = 4;
        IndiceNoUrgente = 3;
        estado = 7;
    }
    if(POSX >= HSIZE/2 && POSX <= HSIZE-23 && POSY > 8*VSIZE/10+3 && POSY <= 8*VSIZE/10+36 &&
mensajes_no_urgentes > 4){
        registro = 5;
        IndiceNoUrgente = 4;
        estado = 7;
    }
}
```

## 6.5.- Referencias a librerías:

Como se vio en el apartado 3, para conseguir configurar el controlador Ethernet, así como para lograr cargar una página web embebida en el microcontrolador en un servidor que él mismo hostea, se puede intuir que es necesario hacer uso de múltiples librerías.
Podemos referenciar algunas de ellas, quedándonos con las más importantes, aunque todas son necesarias, mencionar también que algunas de ellas han sido ligeramente modificadas para nuestra aplicación desarrollada.

- Librería de lwIP destinada a la gestión de un servidor httpd, presente en:
  C:\ti\TivaWare_C_Series-2.2.0.295\third_party\lwip-1.4.1\apps\httpserver_raw

- Librería de la pantalla FT800, donde ha residido el peso del desarrollo de la interfaz de la aplicación de supervisión.

- “lwipopts.h” que es un archivo que se encarga de la configuración de lwIP.

- “cgifuncs” que es una librería con una serie de funciones útiles para interaccionar con la página web y la información que envía la misma.

- “iofsdata.h” que contiene codificados los contenidos de la página web, de manera que serán escritos en la memoria flash del microcontrolador.

- “io_fs.c” que procesa el contenido del archivo “iofsdata.h”.

Entre otras, como “driverlib2.h” o “string.h” por ejemplo.