ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERÍA ELECTROMECÁNICA

ESPECIALIDAD ELECTRÓNICA

FPGA WEB

Autor: Rita de Miguel Fernández
Directores: Sadot Alexandres Fernández
José Daniel Muñoz Frías

Madrid
Julio 2016

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Rita de Miguel Fernández DECLARA ser el titular de los derechos de propiedad intelectual de la obra: FPGA WEB, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.
d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción

de derechos derivada de las obras objeto de la cesión.

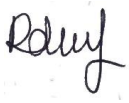
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 18 de Julio de 2016

ACEPTA

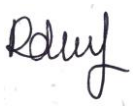


Fdo. Rita de Miguel Fernández

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título FPGA WEB en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2015/2016 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Rita de Miguel Fernández Fecha: 18/07/2016



Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Sadot Alexandres Fernández Fecha: 18/07/2016

Fdo.: José Daniel Muñoz Frías Fecha: 18/07/2016

Vº Bº del Coordinador de Proyectos

Fdo.: Álvaro Sánchez Miralles Fecha: 19/07/2016

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERÍA ELECTROMECÁNICA

ESPECIALIDAD ELECTRÓNICA

FPGA WEB

Autor: Rita de Miguel Fernández
Directores: Sadot Alexandres Fernández
José Daniel Muñoz Frías

Madrid
Julio 2016

FPGA WEB

Autor: de Miguel Fernández, Rita

Directores: Alexandres Fernández, Sadot

Muñoz Frías, José Daniel

RESUMEN

INTRODUCCIÓN

Hoy en día, en prácticamente todos los hogares se dispone de al menos un ordenador, así como dispositivos electrónicos móviles como smartphones y tablets. Mediante este proyecto se pretende hacer uso de estas tecnologías para poder tener acceso a determinado material de los laboratorios de ICAI mediante internet, para que así los alumnos puedan realizar prácticas, o simplemente investigar por su cuenta, de una forma sencilla y sin necesidad de disponer directamente del material hardware necesario. Esta idea se corresponde con el concepto de laboratorio remoto.

Un laboratorio remoto es una herramienta de software y hardware que permite a los estudiantes acceder de forma remota a equipos reales situados en la universidad. Los estudiantes pueden utilizar este material como si estuvieran en una sesión de laboratorio rutinaria.

OBJETIVOS

El objetivo principal del proyecto es la creación de un laboratorio remoto funcional para su instalación en la universidad. Se centrará en la implementación en una placa FPGA Altera DE1 de un sistema que permita su manejo de forma remota mediante un ordenador o dispositivo móvil. Este sistema proporciona a los alumnos la capacidad de controlar el equipo real, localizado en el centro, de manera que pueden programarlo y manipularlo con sus propios archivos sin necesidad de encontrarse en la universidad.

METODOLOGÍA

Como se ha mencionado en el apartado anterior, el dispositivo elegido para su control por internet es la FPGA Altera DE1 [1], escogido por ser una herramienta que se utiliza en algunas asignaturas impartidas en la universidad. Se ha utilizado el software CAD del mismo desarrollador que la placa, Quartus II [2], para la carga de proyectos en la misma.

Los principales aspectos a controlar son los interruptores y pulsadores de la placa, la descarga de archivos y la visión de la placa por internet, además de la creación de una página web que permita la ejecución de todos estos pasos. Para todo ello se ha utilizado como herramienta principal Xampp [3], una distribución de Apache que permite el uso de los lenguajes HTML, PHP y JavaScript entre otros.

En primer lugar se ha procedido a realizar el control de los interruptores y los pulsadores, es decir, las entradas manipulables de la FPGA. Para ello se ha utilizado un Arduino Uno [4]. Las conexiones de los elementos a manejar de la FPGA se han conectado físicamente a pines del Arduino (Figura 1), por los que se envía la información necesaria para configurar estos dispositivos como encendidos o apagados mediante la comunicación por el puerto serie.

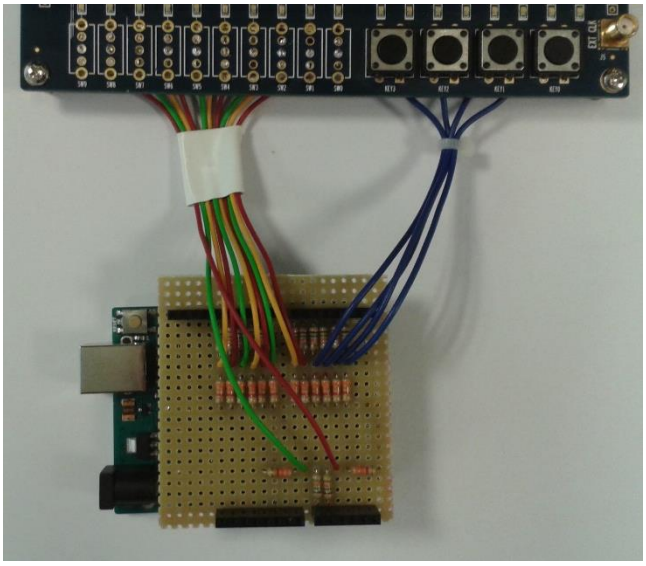


Figura 1. Conexión interruptores y pulsadores

Utilizando el programa Xampp se han construido los botones necesarios para que aparezcan en pantalla y, según cual se presione, se envía un carácter diferente por el puerto serie. Mediante el compilador de Arduino se ha establecido una relación entre cada uno de los posibles caracteres y una acción en alguno de los pines de esta placa.

A continuación se ha desarrollado el módulo de carga de archivos, todo ello a través de Xampp. El sistema creado permite seleccionar el archivo deseado (para la correcta descarga del proyecto es necesario subir el archivo *.sof del mismo, que se crea automáticamente tras una compilación del proyecto en Quartus II) y subirlo al servidor, almacenándolo en una carpeta determinada. Tras esto, otro archivo con extensión *.cdf que se encuentra en la misma carpeta es el que hace posible la descarga del proyecto seleccionado en la FPGA.

Para la visión de la placa se ha utilizado el programa Yawcam [5], que permite la visualización a tiempo real de la grabación de una cámara. Se ha diseñado una estructura ligera (Figura 2) para la distribución de todos los elementos de manera que permanezcan fijos.

Figura 2. Estructura del montaje

Otro componente importante de la página es el temporizador. Existe un temporizador que, después de cierto tiempo de inactividad en la ventana en la que se puede interactuar con la placa, redirige al usuario automáticamente a la página de inicio, permitiendo así el acceso de otras personas.

Finalmente, se han unido todos los módulos anteriores creando una interfaz simple para que el usuario pueda manejar el sistema de forma sencilla. La estructura principal del sitio web es la reflejada en la Figura 3.

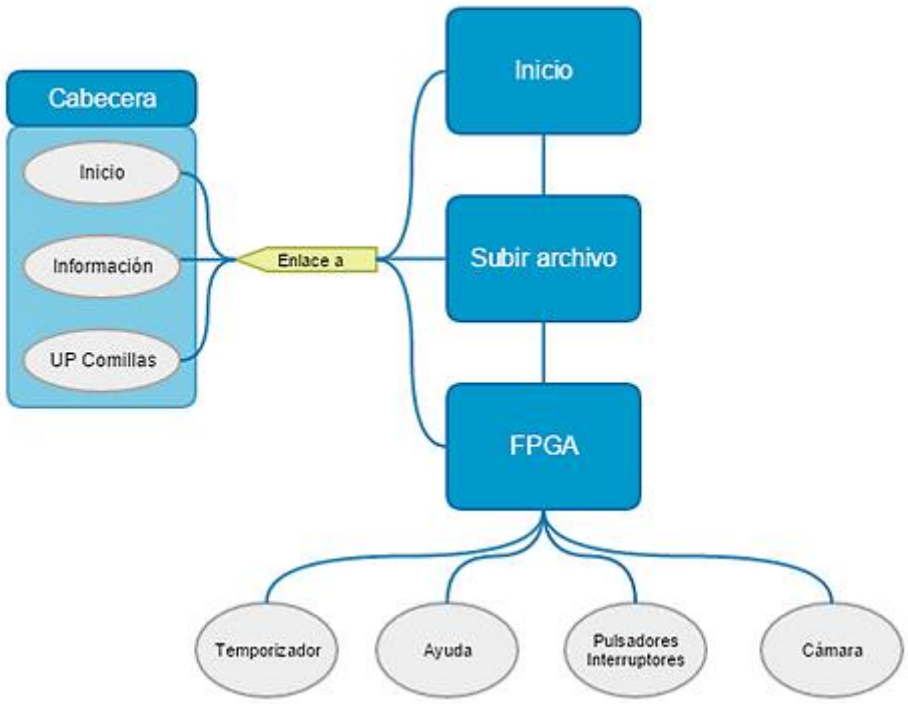


Figura 3. Diagrama de bloques del sitio web

RESULTADOS Y CONCLUSIONES

El resultado final del proyecto es una página web en la que se puede cargar un programa en una placa FPGA, interactuar con ella y ver qué le sucede en tiempo real. La estructura de la misma se puede observar en la Figura 3: a partir de una página de inicio, se accede a la ventana de subida de archivos para finalmente poder manipular la placa de manera remota. La página cuenta con una pantalla de información y ayuda, así como enlaces a la página de inicio y a la página de la Universidad Pontificia de Comillas. La página principal del sitio web, en la que se puede manipular la placa, es la mostrada en la Figura 4.

A pesar de que la página es completamente funcional, cabe destacar que admite la implantación de una gran cantidad de mejoras, como pueden ser crear un control de usuarios o añadir nuevos dispositivos para controlarlos de forma remota. Estos aspectos serán considerados por futuros alumnos que continúen el proyecto.

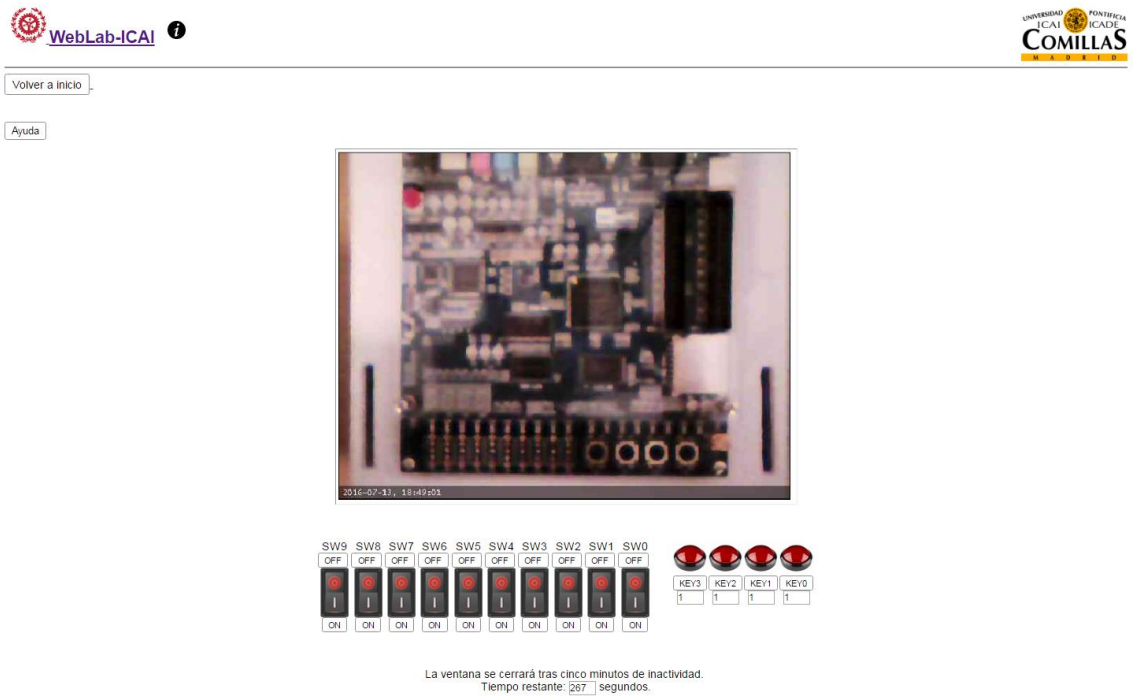


Figura 4. Página web

REFERENCIAS

[1] Altera DE1 [Online] http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=83

[2] Quartus II [Online] http://dl.altera.com/13.0sp1/?edition=web

[3] Xampp [Online] https://www.apachefriends.org/es/index.html

[4] Arduino Uno [Online] https://www.arduino.cc/en/Main/ArduinoBoardUno

[5] Yawcam [Online] http://www.yawcam.com/

FPGA WEB

Author: de Miguel Fernández, Rita

Directors: Alexandres Fernández, Sadot

Muñoz Frías, José Daniel

ABSTRACT

INTRODUCTION

Nowadays, there is a computer in nearly every home, such as mobile electronic devices like smartphones or tablets. This project aims to use these technologies to provide access to determined material of ICAI's laboratories through the internet, in order to allow the students to perform their experiments or investigate easily on their own. This project eliminates the necessity of possessing the hardware devices in the same room. This concept correlates to a remote laboratory.

A remote laboratory consists in a software and hardware tool that allows the students to remotely access to real equipment located at the University. Students can use this material as if they were in a routine laboratory session.

OBJECTIVES

The main objective of the project is the creation of a functional remote laboratory for its installation at the University. It will be focused on the implementation in a FPGA Altera DE1 board of a system that allows its management remotely using a computer or a mobile device. This system provides students the ability of controlling real equipment, located in the building, so they can program it and handle it with their own files without having to be in the University.

METHODOLOGY

As it was mentioned in the section above, the chosen device to control by internet is the FPGA Altera DE1 [1]. It was selected because this instrument is used in some subjects taught at the University. It was chosen the programming tool Quartus II [2] to upload files on the FPGA, since it belongs to the same developer than the board.

The main points to control are the switches and push buttons of the FPGA, the programming of files in the board and the visualization of the board in the web page, in addition to the creation of a web page that allows the execution of all these steps. It has been used as the primary tool a program named Xampp [3], an Apache distribution that allows the use of HTML, PHP and JavaScript languages among others.

First of all the control of the switches and the pushbuttons, i.e. the manipulated entries of the FPGA, needed to be done. To accomplish this point an Arduino Uno [4] was used. Physical connections were made between the elements to handle in the FPGA and the Arduino pins (Figure 1). The needed information to configure the devices as turned on or turned off is sent from the Arduino pins to the buttons using serial port communication.

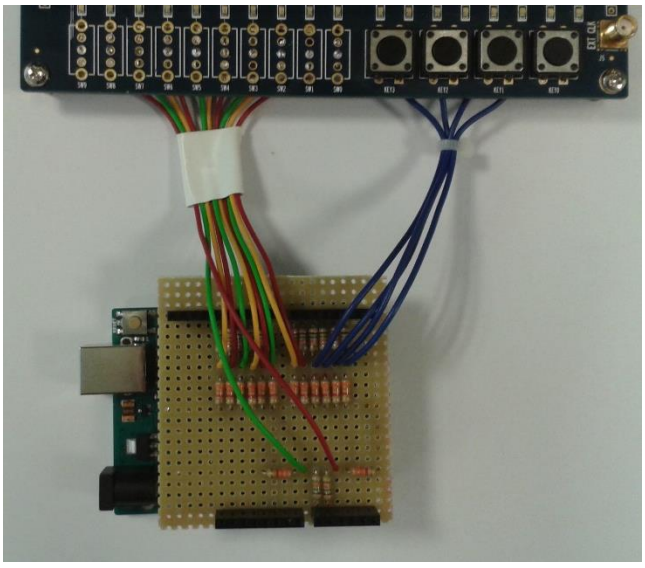


Figure 1. Switches and pushbuttons connection

There have been built the necessary buttons in the screen to control the physical ones, and according to which one is pressed, a different character is sent by the serial port. Using the Arduino compiler a relationship between each of the possible characters and an action on the desired pin has been established.

Then the file upload module had been developed, all through Xampp. The created system allows users to select the desired file (for a correct compilation and programming of the project is necessary to upload the *.sof file, which is automatically created after a project compilation in Quartus II) and upload it to the server, storing it in a specific folder. After this, another file with extension *.cdf, which is located in the same folder, makes possible the programming of the selected project in the FPGA.

To obtain the visualization of the board a program called Yawcam [5] was used. This program allows real time display of the record of a camera. A lightweight structure (Figure 2) has been designed for the distribution of all the elements so they remain fixed.

Figure 2. Structure

Another important component of the web page page is the timer. There is a timer that, after a certain time of inactivity on the window in which you can interact with the board, it automatically redirects the user to the home page, allowing the access to other people.

Finally, all the previous modules were joined creating a simple interface so the user can manage the system in a simple way. The main structure of the web site is reflected in the Figure 3.

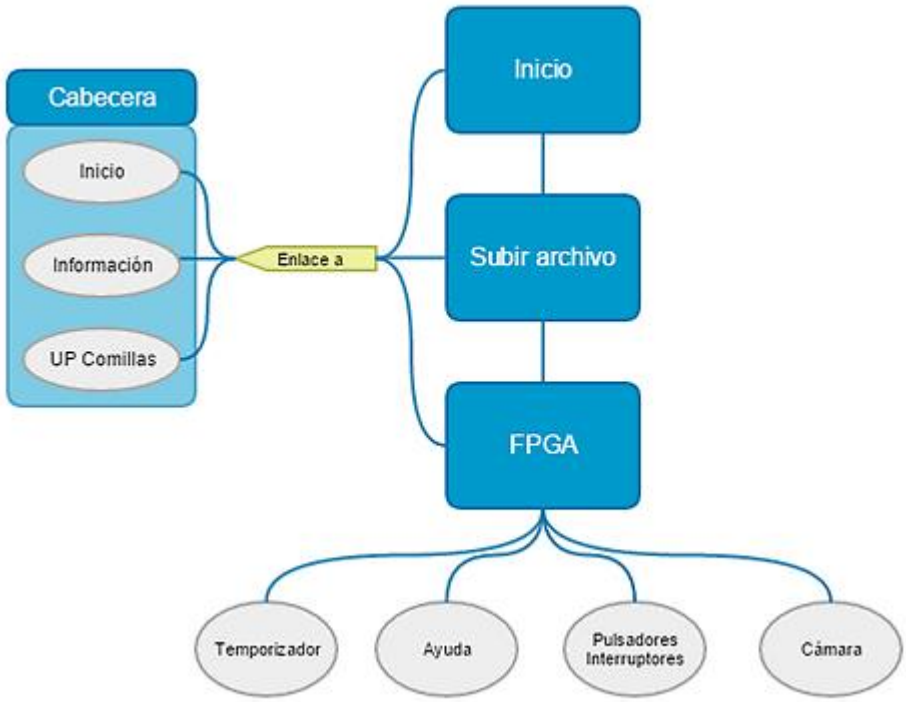


Figure 3. Block diagram of the web site

RESULTS AND CONCLUSIONS

The final result of the project is a web site in which a user can load a program on an FPGA board, interact with it and see what happens in it in real time. The structure of the page can be observed in the Figure 3: beginning in a home page, the user can access to a window where he can upload the desired file to, finally, be able to handle the board in a remote way. The page has a screen of information and support, as well as links to the home page and the web page of the Universidad Pontificia de Comillas. The principal page of the web site, the one in which you can manipulate the board, is shown in the Figure 4.

While the page is fully-functional, is noteworthy that it supports the implementation of a large number of improvements, such as creating a user control or adding new devices to control them remotely. These aspects will be considered by prospective students that will continue the project.

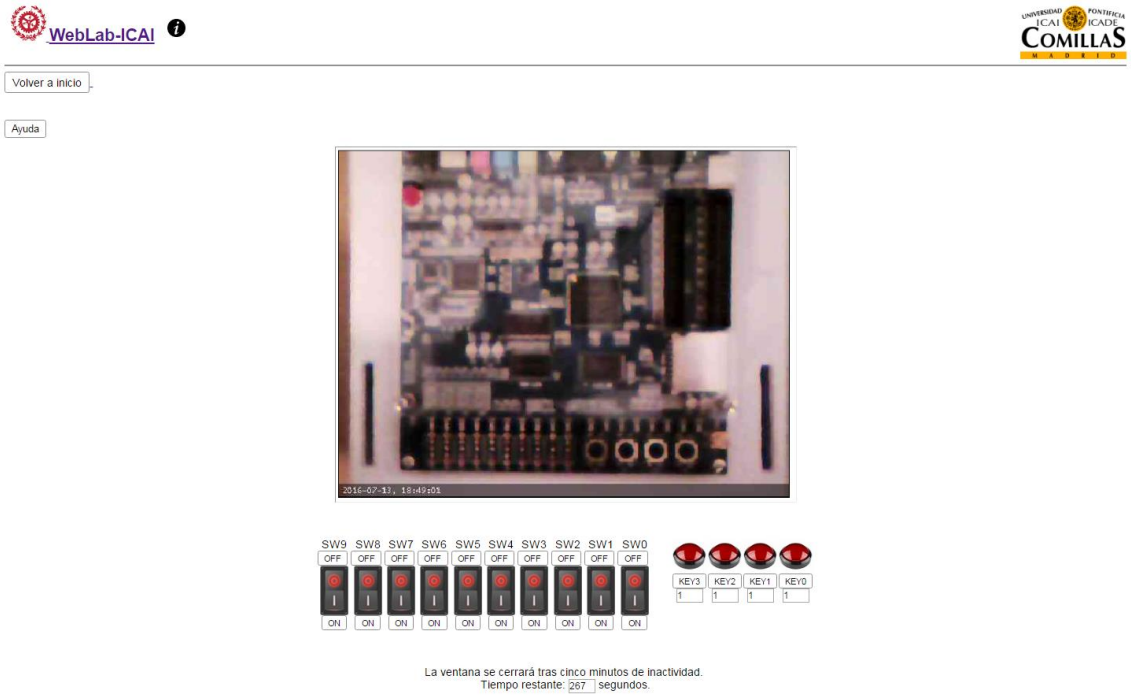


Figure 4. Web page

REFERENCES

[1] Altera DE1 [Online] http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=83

[2] Quartus II [Online] http://dl.altera.com/13.0sp1/?edition=web

[3] Xampp [Online] https://www.apachefriends.org/es/index.html

[4] Arduino Uno [Online] https://www.arduino.cc/en/Main/ArduinoBoardUno

[5] Yawcam [Online] http://www.yawcam.com/

Agradecimientos

A mis directores Sadot y Daniel, por su ayuda y dedicación durante el desarrollo del proyecto.

A David, por compartir una parte de sus conocimientos conmigo.

A José y Antonio, por sus arreglos y su paciencia.

A mis compañeros del día a día, por su compañía, ayuda y paciencia durante este largo camino.

A Javier, por su apoyo y su enorme paciencia conmigo. Gracias por aguantarme en esos momentos que ni yo misma lo consigo.

A mi familia, porque sin su apoyo nada de esto habría sido posible.

Índice de la memoria

Índice de figuras

Índice de tablas

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

El acelerado avance de la tecnología está provocando un cambio substancial en la forma de vida de las personas, haciendo posible realizar muchas tareas de un modo más sencillo y accesible. Uno de los campos en los que esta evolución ha tenido un gran impacto es la docencia, ya que la tecnología ofrece en este ámbito una enorme cantidad de posibilidades tanto para la enseñanza como para el aprendizaje.

Un laboratorio remoto es una herramienta de software y hardware que permite a los estudiantes acceder de forma remota a equipos reales situados en la universidad. Los estudiantes pueden utilizar este material como si estuvieran en una sesión de laboratorio rutinaria.

Hoy en día, prácticamente todos los hogares cuentan con al menos un ordenador, así como con dispositivos móviles inteligentes (smartphones, tablets…). Mediante este proyecto se pretende hacer uso de estas tecnologías para poder tener acceso a determinado material de los laboratorios físicos de ICAI mediante internet, y así poder realizar prácticas o ensayos de una forma sencilla sin necesidad de disponer de los elementos precisos en todo momento.

1.1 ESTUDIO DE LOS TRABAJOS EXISTENTES / TECNOLOGÍAS EXISTENTES

Actualmente existe un laboratorio remoto creado en la Universidad de Deusto bajo el nombre de WebLab-DEUSTO [1] para incentivar el aprendizaje experimental. Para este fin, se ofrecen diversas posibilidades a través de Internet con diferentes equipos en los que se puede trabajar libremente:

- **WebLab-DEUSTO-CPLD** (Figura 1): permite controlar un dispositivo programable CPLD (Complex Programmable Logic Device). Por problemas de seguridad, no se pueden utilizar archivos propios en este conjunto, sino que hay instalado un programa de demostración para utilizar esta parte del laboratorio. Mediante una cámara se pueden observar los resultados en la placa, y se puede interactuar con el dispositivo.

Figura 1. WebLab-DEUSTO-CPLD

- **WebLab-DEUSTO-PIC**: permite programar y controlar un dispositivo programable microcontrolador PIC (Peripheral Interface Controller).
- **WebLab-DEUSTO-VISIR** (Figura 2): permite construir y analizar circuitos analógicos básicos. Los estudiantes pueden crear circuitos utilizando la interfaz, así como analizar los resultados mediante diferentes aparatos como un multímetro.

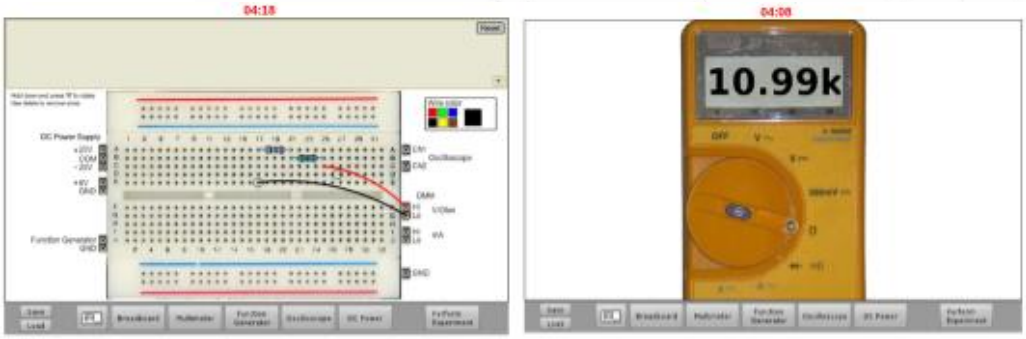


Figura 2. WebLab-DEUSTO-VISIR

- **WebLab-DEUSTO-LXI**: permite construir y analizar circuitos analógicos bajo el estándar LXI.
- **SecondLab**: permite controlar y programar un microrrobot a través del entorno SecondLife.
- **WebLab-DEUSTO-FPGA** (Figura 3): permite practicar de forma remota con un dispositivo programable FPGA (Field Programmable Gate Array). Mediante la herramienta software Xilinx, se puede escribir en el módulo como se haría normalmente. Para ver los resultados, hay una cámara web enfocando a la placa. Además, se puede interactuar con la misma, es decir, la interfaz permite que el usuario maneje los interruptores y pulsadores del dispositivo de forma remota.

Figura 3. WebLab-DEUSTO-FPGA

Este proyecto se centrará precisamente en este último apartado, es decir, en la programación y control remoto de un dispositivo FPGA.

1.2 MOTIVACIÓN DEL PROYECTO

A día de hoy, en ICAI se necesitan unos horarios inflexibles, así como una organización del espacio físico disponible, como son los laboratorios y las aulas. En época de más trabajos y exámenes, a veces no es posible una buena organización respecto a este último tema, el espacio, ya que los laboratorios se llenan de alumnos que necesitan estas zonas para poder cumplir con sus agendas.

Un laboratorio remoto en la Universidad puede ser parte de la solución a este problema al que nos enfrentamos.

1.3 OBJETIVOS

El objetivo principal de este proyecto es implementar un sistema en una placa FPGA (dispuesta especialmente para este fin en el laboratorio) que permita manejarla desde cualquier dispositivo con acceso a internet, tanto en ordenadores como dispositivos móviles.

Este reciente concepto de laboratorio remoto permite que los alumnos sean capaces de controlar un equipo hardware situado en un laboratorio del centro, lo puedan programar y controlar sus entradas para obtener unas salidas, todo ello sin desplazarse a la universidad.

Además, el desarrollo del mismo conlleva el cumplimiento de los siguientes subobjetivos:

- Como el equipo instalado estará encendido 24 horas al día, 365 días al año, se incrementa el rendimiento de los dispositivos utilizados para crear el laboratorio remoto.
- Mejor organización de los laboratorios: mediante la implementación de esta idea, es suficiente con que esté disponible su acceso mediante Internet, no es preciso tener abiertos los laboratorios físicos de la universidad todo el tiempo.
- Con WebLabs, se podrán organizar mejor los horarios de las clases, así como las propias agendas de alumnos y profesores.
- Fomentación del autoaprendizaje.

1.4 METODOLOGÍA / SOLUCIÓN DESARROLLADA

Para lograr los objetivos propuestos es necesario seguir una serie de actividades descritas a continuación.

- **Actividad 1**. Estudio y evaluación del proyecto, que consiste en analizar la viabilidad del mismo y el estado del arte de los sistemas similares que tienen objetivos acordes a los buscados en este proyecto. Se estudian en este punto, tanto los aspectos hardware como software, los económicos, académicos y de funcionalidad.
- **Actividad 2**. Elección y especificación de los sistemas. Esta actividad conlleva la definición a nivel de diagramas de bloques, conexiones e interfaces de los sistemas que se integran más adelante. Evaluación de plataformas web.
- **Actividad 3**. Diseño y adaptación hardware. Preparación de los sistemas hardware, tanto ordenador, conexión, cámara y plataforma programable a usar.
- **Actividad 4**. Diseño del módulo servidor. Desarrollo de los módulos software de adquisición, presentación de la parte correspondiente a la conexión local y servidor web. Se analiza en este punto la funcionalidad de la herramienta del fabricante Altera (Quartus II-Web Edition [2]) de forma remota y local.
- **Actividad 5**. Diseño del módulo cliente. Facilidad y acceso a la herramienta de laboratorio a través del servidor. De la misma forma se diseña la mejor posibilidad de interacción con el hardware de forma remota.
- **Actividad 6**. Integración y prueba de módulos.
- **Actividad 7**. Desarrollo de la memoria del proyecto.

1.5 HERRAMIENTAS EMPLEADAS

En esta sección se procederá a detallar los materiales elegidos tanto a nivel de hardware como de software, así como la función que desempeñará cada parte en el proyecto.

1.5.1 HERRAMIENTAS HARDWARE

1.5.1.1 Placa FPGA

La herramienta elegida para manipular de manera remota es la placa Cyclone II de Altera en una placa de desarrollo DE1 de Terasic [3] (Figura 4). Se ha escogido esta FPGA por ser la utilizada en las prácticas de laboratorio de varias asignaturas impartidas en ICAI. En la Figura 5 se muestra el diagrama de bloques de la plataforma.

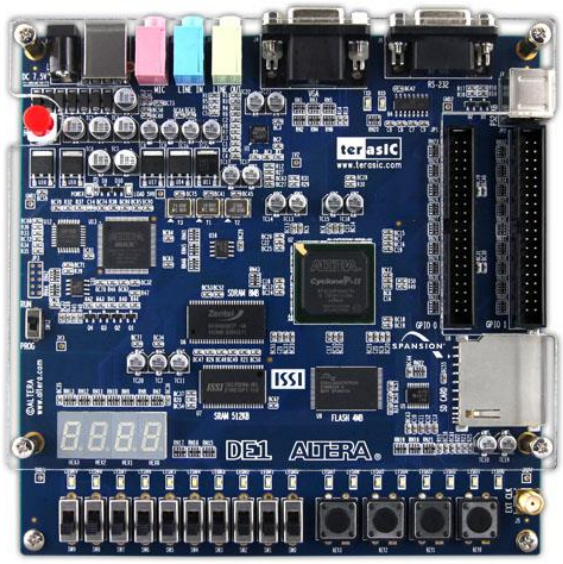


Figura 4. Plataforma Terasic DE1

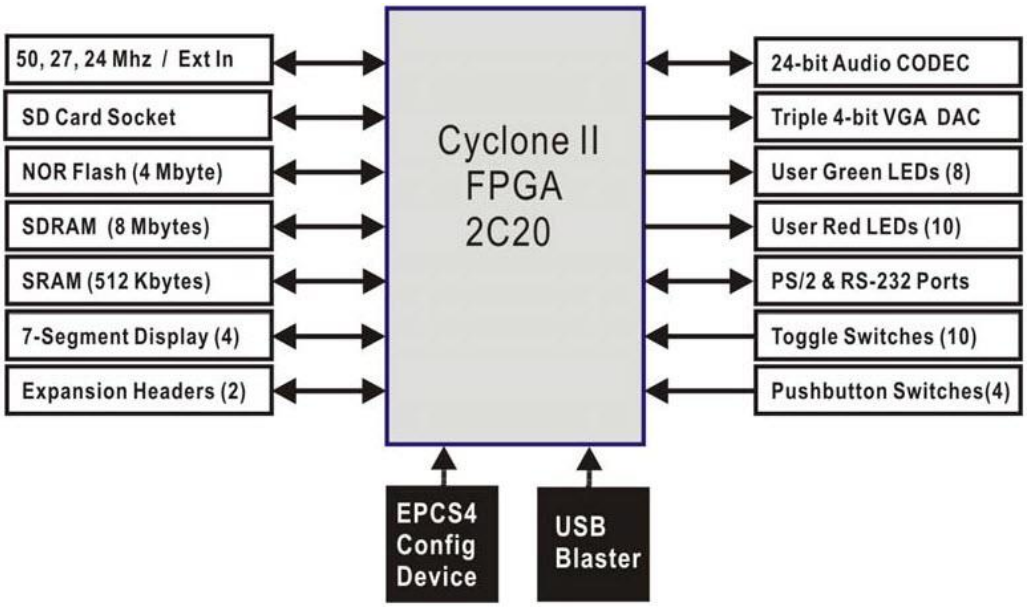


Figura 5. Diagrama de bloques Altera Cyclone II

1.5.1.2 Placa Arduino Uno

Se utiliza una placa Arduino Uno [4] (Figura 6) para realizar el control de los interruptores y pulsadores.

Figura 6. Arduino Uno

1.5.1.3 Cámara web

La cámara web utilizada es una V7 Vantage WebCam 300 CS0300-1E [5] (Figura 7). Su calidad es sencilla y tiene conexión tipo USB y 16 Megapixels. Su papel en el proyecto es mostrar por pantalla a tiempo real las modificaciones realizadas en la placa FPGA causadas por el programa cargado y la manipulación de los

interruptores y pulsadores. A pesar de la baja calidad de la imagen ofrecida por la cámara, ésta es suficiente para observar los cambios mencionados anteriormente.

Figura 7. V7 Vantage WebCam

1.5.1.4 Ordenador

Para el desarrollo del proyecto y su implementación se ha utilizado un ordenador estándar con sistema operativo Windows y navegador Google Chrome.

1.5.2 HERRAMIENTAS SOFTWARE

1.5.2.1 Arduino

Del mismo modo que la placa Arduino Uno, se utiliza el entorno de desarrollo de Arduino para el control de los botones de la FPGA.

1.5.2.2 Quartus II

Para realizar la carga de un programa en la FPGA Altera DE1, se utiliza el compilador del mismo desarrollador (Terasic). Del mismo modo que de forma manual, para cargar un proyecto de forma remota también es necesario el Quartus II.

1.5.2.3 Xampp

Xampp [6] es una aplicación de Apache gratuita. Se utiliza para la creación de la página web en su totalidad.

Capítulo 2 MONTAJE A NIVEL DE HARDWARE

En este capítulo se explicará el montaje llevado a cabo para el correcto funcionamiento del proyecto.

En primer lugar, se ha de tener en cuenta que es necesario el acceso remoto a los interruptores y los pulsadores de la FPGA. Se pretende abordar este tema soldando un cable a cada componente para conectarlo al Arduino, y así ser capaces de enviar desde el ordenador información binaria para simular que dichos componentes están en cada uno de sus respectivos modos de funcionamiento. Para ello, se consulta el manual de usuario de la placa [7], con el objetivo de analizar las conexiones internas del conjunto para comprobar si estas piezas se pueden manipular directamente.

En la Figura 8 se puede observar que en los pulsadores no hay ningún problema en soldar un cable en el mismo punto de conexión, puesto que su estado natural es en abierto.

Por el contrario, en el caso de los interruptores no se puede acceder a ese punto directamente, ya que la conexión entre la FPGA y el dispositivo es siempre una toma a tierra o a la alimentación de la placa (3.3 V), por lo que si se intentara introducir un '0' lógico (0 V) mientras el interruptor está en su posición de encendido (3.3 V), o viceversa, se provocaría un cortocircuito. Como el acceso no es directo, se ha optado por eliminar los interruptores de la plataforma, (Figura 9) y de esta manera poder conectar los cables del mismo modo que en los pulsadores. Se ha tenido en cuenta que la FPGA se alimenta a una tensión de 3.3 V, mientras que las salidas del Arduino dan una tensión de 5 V, por lo que para evitar problemas de conexionado se ha colocado un divisor de tensión para cada componente entre ambos puntos de conexión, haciendo un total de 14 divisores (Figura 10).

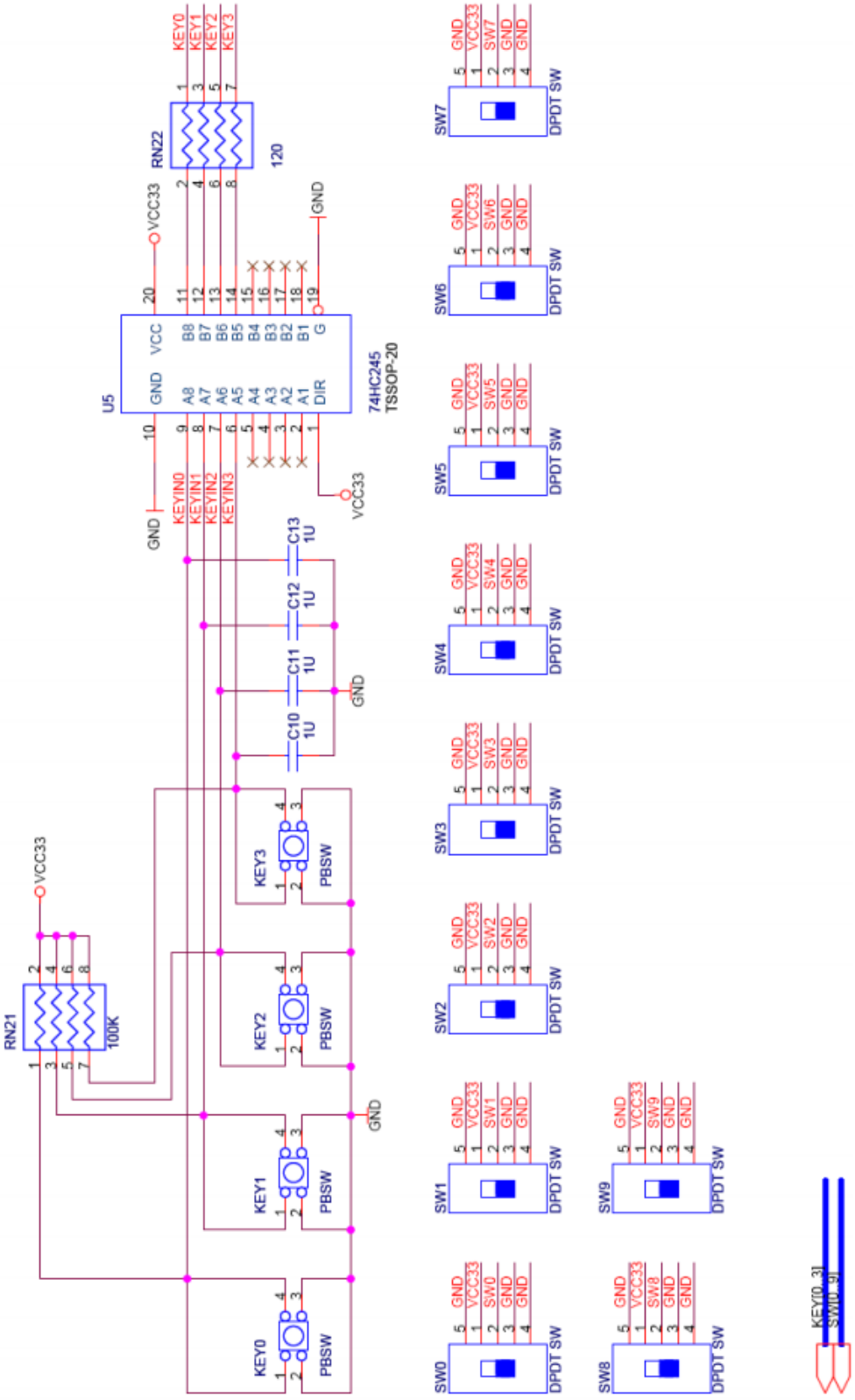


Figura 8. Conexión interna interruptores y pulsadores

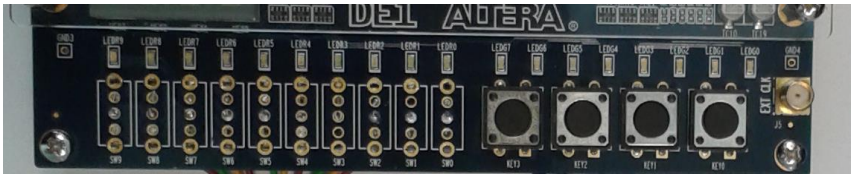


Figura 9. Eliminación de interruptores

En la Figura 10, el terminal denominado "ARDUINO" correspondería con las salidas digitales 2-13, así como las analógicas A0 y A1 (no se pueden utilizar las salidas digitales 0 y 1 porque corresponden a la UART del Arduino, y como el puerto serie está abierto para poder comunicar el sitio web con la placa esos pines se configuran automáticamente). El terminal "FPGA" corresponde con los puertos de entrada de los interruptores (SW0-SW9) y de los pulsadores (KEY0-KEY3).

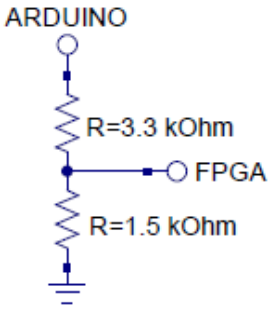


Figura 10. Divisor de tensión

Para la primera parte del desarrollo y las pruebas del conjunto se ha utilizado la placa de Arduino junto con una protoboard como las que se usan para la realización de prácticas en algunas asignaturas, siendo el de la Figura 11 el montaje provisional obtenido.

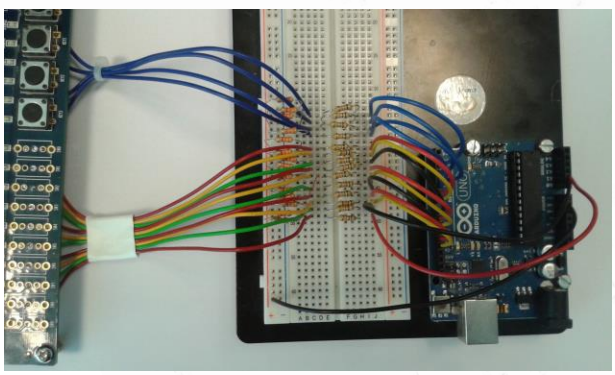


Figura 11. Montaje provisional

Tras verificar el correcto funcionamiento de todas las nuevas uniones y la parte de software relacionada, se procede a ligar permanentemente los componentes del circuito en una placa de soldadura, y a sustituir el Arduino de pruebas por otro sin protoboard, obteniendo el resultado mostrado en la Figura 12.

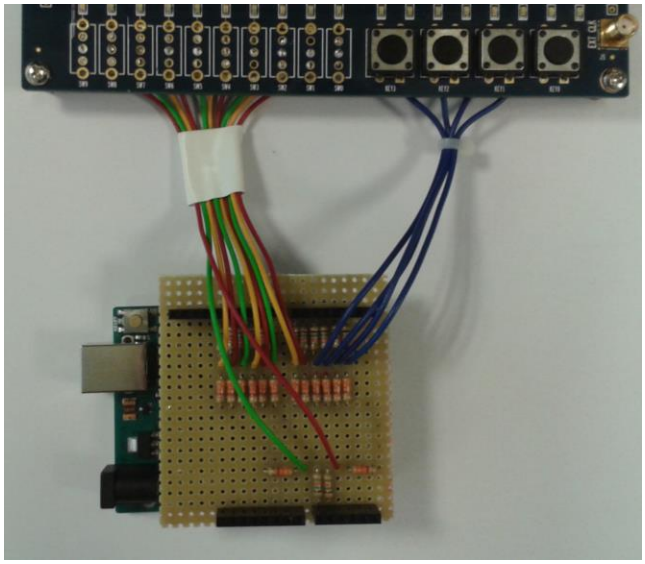


Figura 12. Montaje definitivo

Finalmente, para colocar la cámara de manera que se pueda ver la FPGA por Internet, se ha creado una estructura de cartón pluma (Figura 13), ocultando debajo el Arduino por cuestiones estéticas, ya que no es necesario el acceso al mismo más que por la conexión USB.

Figura 13. Estructura del montaje

Capítulo 3 DESARROLLO A NIVEL DE SOFTWARE

En este apartado se explicará cómo se ha realizado el código de control para cada una de las partes principales del proyecto, así como su integración en un solo sitio web.

Para la creación del sitio web, se han probado diversos programas. En primer lugar se ha intentado hacer mediante Visual Studio [8], pero se ha descartado su uso por la complejidad del programa y por el desconocimiento del mismo. A continuación, se ha procedido a utilizar Joomla [9], pero también se ha rechazado por incompatibilidad del programa con algunos hitos del proyecto (imposibilidad de comunicación con el puerto serie de Arduino y con información del propio servidor). Finalmente, se ha tomado la decisión de trabajar mediante Xampp, una distribución gratuita de Apache que permite la creación de una página web programando directamente en HTML [10], PHP [11] y JavaScript (los lenguajes utilizados durante el desarrollo de la página).

Se explicará el proceso llevado a cabo utilizando pequeñas partes del código; en caso de querer conocer con detalle el mismo, se puede acceder al código fuente definitivo de todo el proyecto, anexado en la Parte IV del documento.

3.1 CONTROL DE LOS INTERRUPTORES Y PULSADORES

Para conseguir en correcto control de los interruptores y pulsadores, se ha utilizado una placa Arduino Uno, como se ha comentado anteriormente, mediante las conexiones explicadas en el apartado anterior. En cuanto al desarrollo software, hay que tener en cuenta dos partes: por un lado, un código integrado en

el Arduino para que se comunique con la FPGA, y por otro, un código encargado de comunicar la interfaz web con el Arduino.

En la primera prueba, se ha decidido que los interruptores y pulsadores actuasen de la misma manera (las posiciones de ON y de OFF se accionan de la misma forma: al enviar un carácter determinado por el puerto serie, si el dispositivo está en nivel alto pasa a nivel bajo, y viceversa), para modificar su funcionamiento progresivamente hasta alcanzar el deseado.

3.1.1 CÓDIGO DE ARDUINO

En primer lugar, se han definido las variables correspondientes a cada pulsador (*puls0-puls3*) y a cada interruptor (*switch0-switch9*), asignando el valor de los pines digitales del Arduino a cada una de ellas, del 2 al 15 respectivamente (como se ha mencionado anteriormente, no se pueden utilizar los pines 0 y 1, ya que están asignados a la UART y se hará uso de la comunicación serie, por lo que se utilizan los pines analógicos A0 y A1 denominándolos como 14 y 15 respectivamente). También se han definido variables auxiliares, una para cada componente que se quiere controlar (*auxp0-auxp3* para los pulsadores y *auxs0-auxs9* para los interruptores), todas inicializadas con valor cero (estas variables auxiliares servirán para controlar si las salidas asociadas están a nivel alto o bajo), así como una variable llamada *option* para leer la información enviada por el puerto serie. Todas estas variables son definidas como enteras. En la Figura 14 se puede observar una muestra de estas variables.

```
int option;
int puls0 = 2;
int auxp0 = 0;
int switch0 = 6;
int auxs0 = 0;
```

Figura 14. Definición de variables

A continuación, se abre el puerto serie del Arduino a una velocidad de 9600 baudios y se configuran los pines como salidas, además de mandar ceros por los pines asociados a los pulsadores (activos a nivel alto) y unos por los asociados a

los interruptores (activos a nivel bajo). En la Figura 15 se muestra un ejemplo de lo explicado.

```
void setup(){
  Serial.begin(9600);
  pinMode(puls0, OUTPUT);
  digitalWrite(puls0, HIGH);
  pinMode(switch0, OUTPUT);
  digitalWrite(switch0, LOW);
}
```

Figura 15. Setup de variables

En el bucle del código, lo primero que sucede es que el programa consulta si hay algún dato esperando en el puerto serie. Si lo hay, mediante la variable *option* se lee, y si se trata de algún carácter que maneja alguno de los dispositivos a tratar, éste se pondrá a nivel alto o bajo, según corresponda. A continuación, en la Figura 16, se presenta el código correspondiente al accionamiento de uno de los pulsadores y uno de los interruptores.

```
void loop(){
  if (Serial.available()>0){
    option=Serial.read();

    if(option=='q'){ //PULSADOR 0
      if(auxp0==1){
        digitalWrite(puls0, HIGH);
        Serial.println("PULS0 OFF");
        auxp0=0;
      }else if(auxp0==0){
        digitalWrite(puls0, LOW);
        Serial.println("PULS0 ON");
        auxp0=1;
      }
    }
    if(option=='0'){ //INTERRUPTOR 0
      if(auxs0==0){
        digitalWrite(switch0, HIGH);
        Serial.println("SWITCH0 ON");
        auxs0=1;
      }else if(auxs0==1){
        digitalWrite(switch0, LOW);
        Serial.println("SWITCH0 OFF");
        auxs0=0;
      }
    }
  }
```

Figura 16. Comunicación serie Arduino

Como se ha explicado antes, de esta manera tanto los pulsadores como los interruptores cambian su estado al enviar un carácter determinado por el puerto serie, pero el funcionamiento real de estos dispositivos no es así. Para el funcionamiento final de esta parte, se ha decidido que el código relativo a los pulsadores sea el explicado anteriormente (se corregirá para que funcione como

un pulsador real en el código HTML), mientras que se ha modificado la parte relacionada con los interruptores: se han eliminado las variables auxiliares *auxs0-auxs9*, y se ha decidido hacer el envío de datos de una forma distinta; en lugar de cambiar tanto de nivel alto a bajo como de nivel bajo a alto utilizando el mismo carácter, ahora se utilizan dos diferentes. En una ventana gráfica esto se traduce a tener dos botones, uno de ON y uno de OFF, en lugar de sólo uno. Esto se ve reflejado en la Figura 17.

```
if(option=='0'){ //INTERRUPTOR 0 ON
  digitalWrite(switch0, HIGH);
  Serial.println("SWITCH0 ON");
}
if(option=='p'){ //INTERRUPTOR 0 OFF
  digitalWrite(switch0, LOW);
  Serial.println("SWITCH0 OFF");
}
```

Figura 17. Puerto serie interruptores

3.1.2 CÓDIGO HTML

En primer lugar, se ha desarrollado el código correspondiente al primer caso planteado anteriormente, es decir, que un pulsador y un interruptor actúan de la misma manera. Para ello, se definen las variables asignadas a cada dispositivo como variables de entrada de tipo *submit* (una variable de este tipo crea un botón en pantalla que al pulsarlo ejecuta un formulario). En la Figura 18 se muestra un ejemplo de definición de estas variables, y en la Figura 19 el resultado por pantalla.

```
<input type="submit" name="sw9" value="SW9" />
```

Figura 18. Definición de variable

Figura 19. Variables por pantalla

A continuación se configura y se abre el puerto serie, teniendo cuidado de seleccionar el mismo puerto COM al que está conectado el Arduino. Para comprobar que esto se hace correctamente, se hace que se envíe por pantalla un texto de verificación. Finalmente, para detectar la pulsación de uno de los botones, se utiliza la función *isset()*, que devuelve un "1" si la variable a la que se llama existe y no está vacía, es decir, que se ha pulsado un botón. En caso de que esto suceda, el programa envía un carácter por el puerto serie. Este carácter se corresponde con el asignado en el código de Arduino para accionar el dispositivo deseado. Esta parte de la programación se muestra en la Figura 20.

```php
<?php

    exec("mode COM8 BAUD=9600 PARITY=N data=8 stop=1 xon=off");
    $fp = @fopen ("COM8", "w");
    if (!$fp) {
        $status = "No conectado";
    } else {
        $status = "Conectado";
    }
    echo $status;
    if(isset($_POST["puls0"])){
        fwrite($fp,"q");
    }
?>
```

Figura 20. Puerto serie HTML

Con lo abordado hasta ahora, se consigue un funcionamiento básico de los interruptores y los pulsadores, mediante el cual un *click* en el botón correspondiente corresponde a encender o apagar el dispositivo en caso de los interruptores, y a pulsar o soltar en caso de los pulsadores.

A continuación se hará la distinción entre ambos componentes, comenzando por completar el código referente a la parte de los interruptores. Se ha decidido añadir una imagen de un interruptor que cambie entre encendido y apagado cuando se pulse, así como que los botones de encender y apagar sean independientes. El resultado sería el mostrado en la Figura 21 y la Figura 22.

```
<input type="submit" name="sw9off" value="SW9 OFF" /> <br>
<img src="interruptor_off.jpg" width="40px" height="70px" id="fotoi9"/><br>
<input type="submit" name="sw9on" value="SW9 ON" />

<?php
    exec("mode COM8 BAUD=9600 PARITY=N data=8 stop=1 xon=off");
    $fp = @fopen ("COM8", "w");
     if(isset($_POST["sw0on"])){
        fwrite($fp,"0");
        ?>
        <script type="text/javascript">
            document.botones.fotoi0.src="interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw0off"])){
        fwrite($fp,"p");
        ?>
        <script type="text/javascript">
            document.botones.fotoi0.src="interruptor_off.jpg"
        </script>
        <?php
    }
 ?>
```

Figura 21. Código interruptor

Figura 22. Interruptor en pantalla

En la Figura 21, el código de Javascript corresponde con el cambio de imagen de encendido a apagado y viceversa. Al probar el programa, se ha observado que el cambio de imagen no funcionaba de manera correcta. Esto es debido a que los lenguajes utilizados no se ejecutan en bucle (como sucede en el caso de C, VHDL…), sino que se ejecutan una sola vez cada vez que se le ordena (se recarga la página, se pulsa algún botón de reenvío de formulario…), por lo que la imagen del interruptor siempre será la predefinida, en este caso la correspondiente al elemento apagado. Por el mismo motivo, no se puede guardar el estado de la imagen en una variable para recuperarlo en la siguiente ejecución. Para solucionarlo, se ha decidido guardar el estado en un archivo de texto, que se lee al principio de cada ejecución y se actualiza al final. Además, en lugar de predefinir

el estado de la imagen se toma el valor almacenado en el archivo mencionado. El resultado del código es el mostrado en la Figura 23.

```
<?php
//Lectura del archivo de texto
    $f9=fopen("interruptores\int9.txt","r");
    $var9=fread($f9,20);
    fclose($f9);
?>

<input type="submit" name="sw9off" value="SW9 OFF" /> <br>
<img src="<?php echo $var9?>" width="40px" height="70px" id="fotoi9"/><br>
<input type="submit" name="sw9on" value="SW9 ON" />

<?php

    //Configuración y apertura del puerto serie para comunicación con arduino
    exec("mode COM8 BAUD=9600 PARITY=N data=8 stop=1 xon=off");
    $fp = @fopen ("COM8", "w");

    //Envío por puerto serie al pulsar interruptores
    if(isset($_POST["sw9on"])){
        fwrite($fp,"9");
        $var9="interruptor_on.jpg";
        ?>
        <script type="text/javascript">
            document.botones.fotoi9.src="interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw9off"])){
        fwrite($fp,"o");
        $var9="interruptor_off.jpg";
        ?>
        <script type="text/javascript">
            document.botones.fotoi9.src="interruptor_off.jpg"
        </script>
        <?php
    }

    //Almacenamiento de estado de interruptor
     $f9=fopen("interruptores\int9.txt","w+");
    fwrite($f9, $var9);
    fclose($f9);
?>
```

Figura 23. Código interruptor

Para finalizar esta parte del proyecto, falta el código correspondiente a los pulsadores. Se ha decidido configurarlos para que al pulsarlos se activen durante un tiempo, y tras pasar ese tiempo que se desactive automáticamente. También se incluye un módulo de texto (variable de entrada tipo *text*) para poder modificar el periodo que el pulsador permanece encendido. El código de muestra representado en la Figura 24 consiste en lo siguiente: se comprueba si hay algo en el campo de texto mencionado anteriormente. En caso de que no lo haya (que se abra esta

página por primera vez) se guardará un "1" en una variable; en caso contrario se guardará el valor escrito dentro del cuadro de texto. A continuación se definen el botón y el cuadro de texto. En la última parte del código se comprueba si se ha pulsado el botón, tras lo cual, en caso afirmativo, se envía el carácter correspondiente por el puerto serie (pulsar el pulsador), se espera el tiempo indicado, y se vuelve a enviar el carácter (soltar el pulsador). En la Figura 25 se representa el módulo mostrado en la página web.

```
<?php
    if (isset($_POST['temp0'])){
        $tiempo_p0=$_POST['temp0'];
    }else{
        $tiempo_p0="1";
    }
?>

<input type="submit" name="p0" value="KEY0" /><br>
<input type="text" size="1" maxlength="2" value="<?php echo $tiempo_p0?>"
name="temp0"/>

<?php

    if(isset($_POST["p0"])){
        fwrite($fp,"a");
        $segundos_p0=$tiempo_p0;
        while($segundos_p0!=0){
            $segundos_p0-=1;
            sleep(1);
        }
        fwrite($fp,"a");
    }
?>
```

Figura 24. Código pulsador

Figura 25. Pulsador en pantalla

Finalmente, de igual forma que en los interruptores, se añade una imagen que represente a cada pulsador. En este caso no es necesario la creación de un archivo de texto; se predefine la imagen como un pulsador apagado. Al pulsar el botón correspondiente se llama a una función que cambia la imagen a encendido, y como el programa no se recarga hasta después de terminar el tiempo asignado, cuando se vuelve a ejecutar el código es el momento en el que se tiene que

modificar la imagen de nuevo a apagado. En la Figura 26 y Figura 27 se muestran respectivamente el código utilizado y una muestra en la pantalla.

```
<script type="text/javascript">
    function cambiarFotop0(){
        document.pulsadores.fotop0.src="imagenes/pulsador_on.jpg";
    }
</script>

<img src="imagenes/pulsador_off.jpg" width="50px" height="50px"
id="fotop0"/><br>
<input type="submit" class="boton" name="p0" value="KEY0"
onClick="javascript:cambiarFotop0();" /><br>
```

Figura 26. Código pulsador

Figura 27. Pulsador en pantalla

Existe un problema en la manipulación de los pulsadores: al utilizar uno de ellos, no se puede interactuar con otro dispositivo mientras ese está activo; es decir, durante el tiempo de actuación de un pulsador (el tiempo indicado en el cuadro de texto) no se puede encender ningún otro interruptor ni pulsador. Esto es debido a que durante este tiempo el puerto serie de comunicación con el Arduino permanece ocupado por este dispositivo, por lo que no es posible enviar más información. En el caso de los interruptores, es posible activarlos antes de manipular el pulsador en caso de que se quieran mantener encendidos a la vez, pero este error impide la activación de dos pulsadores simultáneamente.

En el Capítulo 3, Capítulo 15 y Capítulo 17 de la Parte IV se muestra el código para los diez interruptores y los cuatro pulsadores, ligeramente modificado al mostrado en este apartado, principalmente por motivos estéticos a la hora de la representación por pantalla.

3.2 CONTROL DE LA CÁMARA

Para poder ver en directo lo que sucede en la FPGA, es necesaria la implantación de una cámara que la grabe. Se ha utilizado una cámara web simple conectada por USB al ordenador. La aplicación Yawcam [12] ofrece la oportunidad de, mediante una cámara conectada al ordenador, obtener una IP para ver el video grabado por la misma a tiempo real (Figura 28).

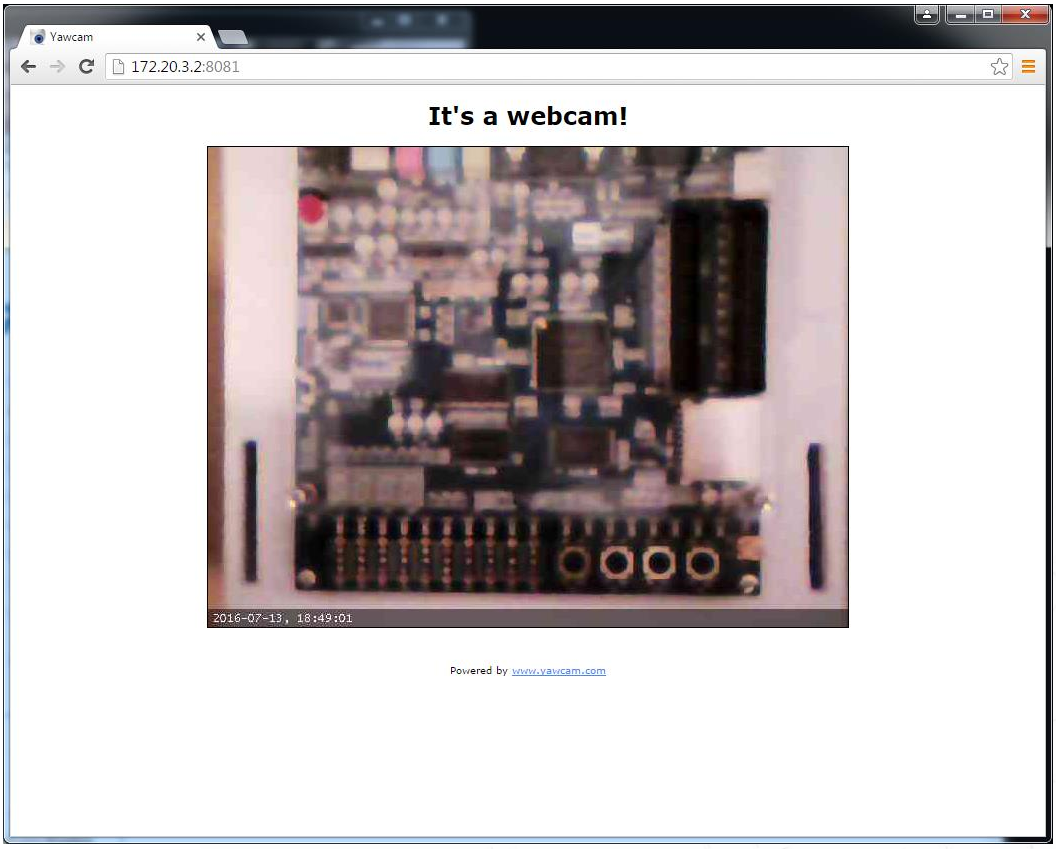


Figura 28. Imagen Yawcam

Para ver esta ventana en la misma página web que el resto de elementos necesarios para la manipulación de la placa, se incrusta creando una URL embebida, es decir, haciendo que una porción de la página muestre información de otra. Esto es posible utilizando la etiqueta *iframe*, que cuyo uso es exactamente el que se necesita, crear un espacio dentro de una página en el que se puede

incrustar otra página. El código puede ser consultado en el Parte IVCapítulo 10 de la Parte IV.

3.3 SUBIDA DE ARCHIVOS AL SERVIDOR Y CARGA DEL PROYECTO EN LA FPGA

Para poder programar la FPGA con el proyecto creado por el alumno, es necesario cargar en la placa el archivo *.sof del mismo. Este archivo se crea automáticamente una vez se ha compilado el programa mediante Quartus II, alojándose en la carpeta *output_files* del proyecto. Además, existe un archivo *.cdf, que es el que llama al *.sof para ejecutarlo. Esto se puede hacer mediante la ventana de comandos, enviando el directorio del programador de Quartus II (Quartus PGM), el medio de transmisión de datos (USB-Blaster) y finalmente el directorio del archivo *.cdf. El comando a enviar es el reflejado en la Figura 29.

Figura 29. Comando de programación

Es importante señalar que el archivo *.cdf contiene el nombre del *.sof, por lo que existen dos maneras de conseguir la programación exitosa de un proyecto cualquiera: o bien cambiar el contenido del archivo *.cdf para que cada vez llame a un archivo *.sof diferente, o bien cambiar el nombre del archivo *.sof por el que contiene el *.cdf antes de proceder a la programación. Se ha optado por el segundo método por la simplicidad de este frente al primero.

Para subir el archivo a la página web se crea un formulario específico para este fin mostrado en la Figura 30, en el que a primera vista se incluye una caja de selección de archivos y un botón para subirlo al servidor. Al apretar el botón, lo primero que sucede es que el nombre del archivo seleccionado se modifica y pasa a ser *archivo.sof*, para tratar con el punto explicado en el apartado anterior. En un primer momento se ha intentado que el botón de subida filtre los archivos por tipo, es decir, que sólo permita el paso de archivos con la extensión deseada y

mande un mensaje de error en caso contrario. Para ello es necesario conocer el tipo de archivos al que pertenece la extensión *.sof (*.txt o *.docx son tipo texto, *.png o *.jpg son tipo imagen…), pero ha sido imposible el conocimiento de este dato, por lo que no existe ningún filtro; si se sube un archivo *.jpg, de igual manera pasará a llamarse *archivo.sof* que si se intentara con uno de la extensión correcta. Esto no es un motivo de preocupación puesto que a la hora de programar la placa, el dispositivo programador dará conocimiento de que el archivo subido no es correcto.

Figura 30. Subida de archivos

Tras cambiar el nombre del archivo, el programa comprueba que en el proceso habido hasta el momento no haya sucedido ningún error. En caso de no haber problemas, se comprueba que en la carpeta de destino del servidor no haya ningún archivo con el mismo nombre (esto podría pasar en caso de surgir algún problema en puntos anteriores donde ya se debería de haber borrado el contenido de la carpeta). Si lo hay, se borra el archivo existente y se pide subir el proyecto de nuevo. En caso de que todo vaya bien, el archivo se guarda en la carpeta deseada

y se informa al usuario del éxito del proceso. En la Figura 31 se muestra el caso de un error de subida.

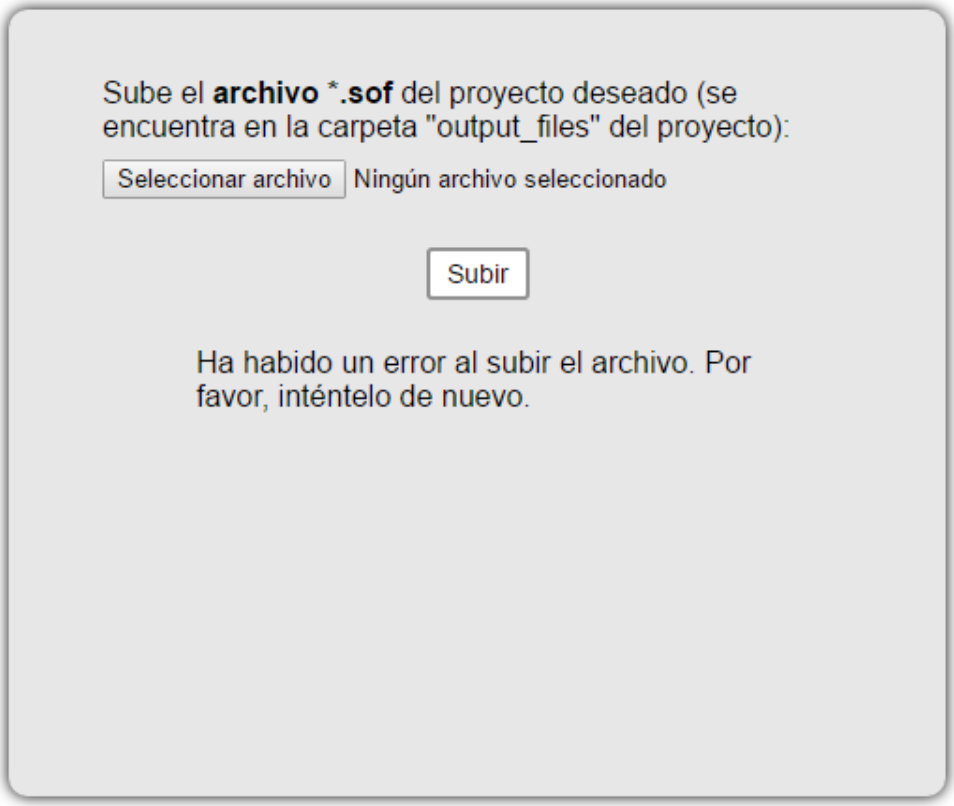


Figura 31. Error de subida

A continuación, el proyecto se programa automáticamente mediante el comando estudiado anteriormente, y se muestra por pantalla la respuesta ofrecida por el Quartus II. En caso de errores en la programación (algún cable mal conectado, el archivo subido es incorrecto o está dañado…), se informa al usuario de que ha habido un error y se muestra también un botón de ayuda, mediante el que se accede a una ventana en la que se detallan los errores más comunes a la hora de programar la placa y cómo actuar ante aquellos de los que puede ser causa el mismo usuario. Finalmente, si no hay problemas y la programación es correcta, se muestra un botón para continuar y poder por fin interactuar con la placa. El código de este módulo se detalla en el Capítulo 5, Capítulo 11 y Capítulo 19 de la Parte IV.

3.4 PLATAFORMA WEB E INTEGRACIÓN

Como se ha mencionado al principio del capítulo, la página web se ha diseñado por completo mediante Xampp, por lo que la unión de todos los módulos creados anteriormente y la adaptación al usuario se ha programado también con esta aplicación de Apache.

Se pretende obtener un sitio web con la estructura de la Figura 32.

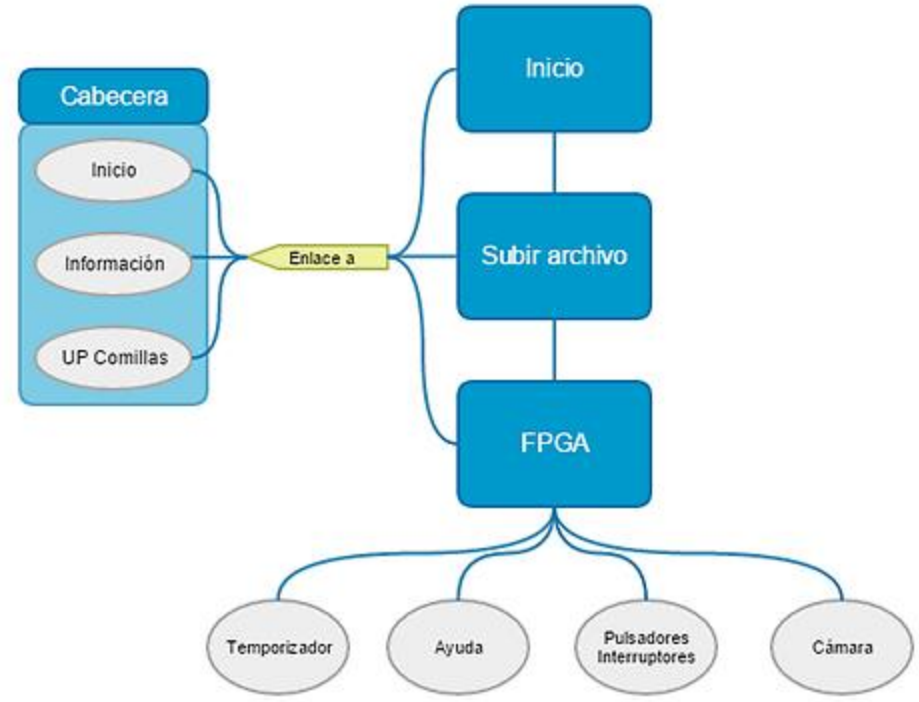


Figura 32. Diagrama de bloques del sitio web

En primer lugar, se procede a la creación de una cabecera que aparezca en las páginas principales. Esta cabecera incluye enlaces a la página de inicio, a una página de información donde se explica qué es WebLab-ICAI y para qué sirve, así como su funcionamiento, y a la página web de la universidad.

A continuación, se programa la página de inicio. En ella se incluye la cabecera, el borrado del archivo *.sof de la carpeta correspondiente y un módulo desde el cual se puede acceder a la subida de un nuevo archivo.

Tras esto, se crea la página de subida de archivos. Esta incluye de nuevo la cabecera, además de un botón adicional de regreso a inicio. También se incrusta el módulo completo de subida de archivos y carga del proyecto en la FPGA, explicado anteriormente. Dentro de este módulo se encuentra la página de ayuda para los principales errores de programación.

Posteriormente, se procede a elaborar la página más importante, en la que se puede interactuar con la placa una vez programada con el proyecto deseado. Esta página incluye la cabecera, los módulos de interruptores y pulsadores, la vista de la cámara, un botón de regreso a inicio, un botón de ayuda y un temporizador. A diferencia de la página de información, que detalla el proceso completo de utilización del producto, este botón de ayuda muestra una captura de la pantalla activa comentando para qué sirve cada módulo incluido en ella (Figura 33). Esta imagen también se encuentra en la página de información de la cabecera.

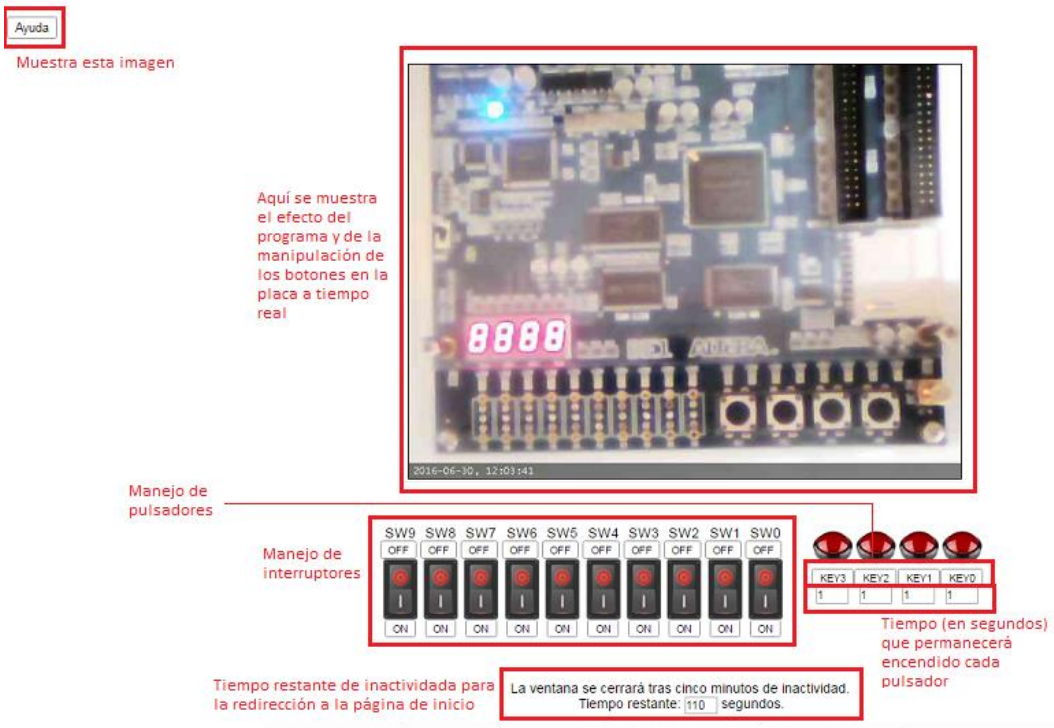


Figura 33. Captura de la pantalla principal

El temporizador (Figura 34) se utiliza para que, tras un tiempo de inactividad de cinco minutos, el programa devuelva al usuario automáticamente a la página de inicio, con el fin de que otro alumno pueda acceder.

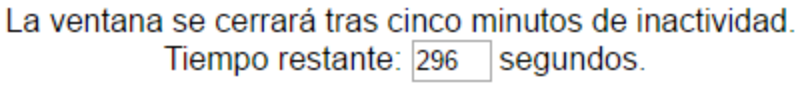


Figura 34. Temporizador

Cabe mencionar que en un principio se deseaba conseguir que no estuviese permitido el acceso de más de un usuario a la vez, puesto que de no ser así es posible que alguien programe la placa mientras otra persona está manipulándola, lo cual no tiene sentido que suceda una vez implantado el sistema en la universidad. No se ha encontrado la manera de conseguir esto, aunque parece una buena forma la creación de un registro de usuarios mediante las claves de alumnos de la universidad, a la que no se ha tenido acceso. De esta manera se podría detectar si alguno de los usuarios registrados está dentro de la página en un momento determinado. Esta es una importante mejora a realizar en un futuro desarrollo de este proyecto.

Capítulo 4 CONCLUSIONES Y FUTUROS DESARROLLOS

Tras la finalización del proyecto se puede concluir con que se han cumplido los objetivos iniciales del proyecto, aunque para la implementación del sistema a nivel de la universidad sería conveniente la realización de algunas mejoras.

En este proyecto se ha construido una página web funcional desde la que se puede programar una FPGA e interactuar con ella a tiempo real. Para ello se ha utilizado un Arduino para controlar los interruptores y los pulsadores de la placa, así como una cámara web para ver lo que sucede mientras se manipulan los dispositivos. Se ha construido una estructura ligera para mantener en una posición fija estos elementos.

Para el desarrollo del proyecto se ha utilizado lenguaje C para el control del Arduino, y HTML, PHP y JavaScript para la creación de la página web. Se ha hecho uso del puerto serie de Arduino para conseguir la manipulación remota de los interruptores y pulsadores.

Como se ha comentado en el apartado 3.1 de la memoria, no se puede accionar un dispositivo si hay algún pulsador encendido. Se ha mencionado que el problema parece ser que recae en que el puerto serie del Arduino está ocupado en el proceso, por lo que no admite más información. Sería conveniente encontrar una solución a este error en futuros desarrollos.

Se ha conseguido realizar un proceso exitoso en la subida y programación de archivos, ofreciendo la información necesaria en caso de que se produzca algún error. No ha sido posible la creación de un filtro que permita solamente el paso de los archivos con la extensión deseada, pero este problema se ha compensado indicando claramente el tipo de archivo a cargar e indicando el fallo a la hora de programar la placa en caso de que se suba un archivo erróneo. También se espera

que en avances futuros se aplique este filtro para evitar la realización del proceso de programación cuando no es realmente necesario.

También es un tema importante a tratar el registro de usuarios. No ha sido posible su implantación por no tener acceso a las claves académicas de la universidad, pero con este sistema se daría solución al mayor problema del proyecto, que es que pueda entrar más de un usuario a la vez.

Finalmente, se concluye con que el laboratorio web creado es completamente funcional, a pesar de tener algunos errores en su programación, que se espera que en la continuación del proyecto de futuros alumnos se arreglen y se incluyan diferentes elementos con los que trabajar de forma remota.

Capítulo 5 REFERENCIAS

[1] WebLab-DEUSTO [Online] http://weblab.deusto.es/website/

[2] Quartus II [Online] http://dl.altera.com/13.0sp1/?edition=web

[3] Altera DE1 [Online] http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=83

[4] Arduino Uno [Online] https://www.arduino.cc/en/Main/ArduinoBoardUno

[5] Webcam [Online] http://www.v7-world.com/v7_uk/products/connect/webcam/vantage-webcam.html

[6] Xampp [Online] https://www.apachefriends.org/es/index.html

[7] Manual de usuario Altera DE1 [Online] ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE1/DE1_User_Manual.pdf

[8] Visual Studio [Online] https://www.visualstudio.com/

[9] Joomla [Online] https://www.joomla.org/

[10] A. Martínez Echevarría, «Manual Práctico de HTML», *Universidad Politécnica de Madrid*, 1995

[11] M. Maraboli Rosselott, «Manual de Programación en PHP», Universidad Técnica Federico Santa María, 2003

[12] Yawcam [Online] http://www.yawcam.com/

Parte II PRESUPUESTO

Capítulo 1 MEDICIONES

En este apartado se listan todos los aspectos del proyecto que implican una aportación económica, así como la cantidad de la que se ha requerido para la consecución de los objetivos.

1.1 MATERIAL

En la Tabla 1 se incluyen todos los costes relacionados con la compra de utensilios necesarios para realizar el proyecto.

Material	*Unidades*
Ordenador	1
Placa FPGA DE1	1
Placa Arduino Uno	1
Cable USB tipo AB	2
Cámara web	1

Tabla 1. Mediciones. Material

1.2 INGENIERÍA

A continuación, en la Tabla 2, se presentan las tareas de ingeniería realizadas y las horas necesarias para su consecución.

Tarea	*Horas*
Búsqueda de información	60
Montaje hardware	30
Control de la cámara	20
Control de los interruptores	80
Diseño de la interfaz	100
Pruebas e integración	100
Escritura de la memoria	50

Tabla 2. Mediciones. Ingeniería

Capítulo 2 PRECIOS UNITARIOS

En este apartado se listan los precios unitarios del material necesario para la realización del proyecto así como del precio por hora para cada tipo de tarea realizada.

2.1 MATERIAL

Material	*Precio (€)*
Ordenador	700
Placa FPGA DE1	127
Placa Arduino Uno	20
Cable USB tipo AB	4
Cámara web	10.54

Tabla 3. Precios unitarios. Material

2.2 INGENIERÍA

Tarea	*Precio (€)*
Búsqueda de información	40
Montaje hardware	50
Control de la cámara	50
Control de los interruptores	50
Diseño de la interfaz	50
Pruebas e integración	50
Escritura de la memoria	40

Tabla 4. Precios unitarios. Ingeniería

Capítulo 3 PRESUPUESTOS PARCIALES

A continuación se multiplican las mediciones por el precio unitario de cada uno de los elementos para obtener la el presupuesto de cada uno de las secciones utilizadas en los apartados anteriores.

3.1 MATERIAL

Material	*Unidades*	*Precio (€)*	**Total (€)**
Ordenador	1	700	700
Placa FPGA DE1	1	127	127
Placa Arduino Uno	1	20	20
Cable USB tipo AB	2	4	8
Cámara web	1	10.54	10.54
Total			**865.54**

Tabla 5. Presupuestos parciales. Material

3.2 INGENIERÍA

Tarea	*Horas*	*Precio (€)*	**Total (€)**
Búsqueda de información	60	40	2400
Montaje hardware	30	50	1500
Control de la cámara	20	50	1000
Control de los interruptores	80	50	4000
Diseño de la interfaz	100	50	5000
Pruebas e integración	100	50	5000
Escritura de la memoria	50	40	2000
Total			**20900**

Tabla 6. Presupuestos parciales. Ingeniería

Capítulo 4 PRESUPUESTO GENERAL

A continuación se presenta el presupuesto general como la suma de todas sumas parciales obtenidas en el apartado anterior. Además, se incluirán los gastos generales, que abarcarán todo lo que no se ha contenido en ninguna sección pero también debe ser tenido en cuenta (luz, mantenimientos, material de oficina, etc.). También se incluirá el beneficio industrial, así como el IVA.

Apartado	*Suma parcial (€)*
Materiales	865.54
Ingeniería	20900
Suma	21765.54
Gastos generales (13%)	2829.52
Beneficio industrial (6%)	1305.93
Subtotal	25900.99
IVA (21%)	5439.21
Total	31340.20

Tabla 7. Presupuesto general

Parte III MANUAL DE USUARIO

¿Qué es WebLab-ICAI?

WebLab-ICAI es un proyecto basado en el laboratorio web creado en la Universidad de Deusto, WebLab-Deusto. Este proyecto consiste en un sitio web que permite interactuar con elementos físicos, habilitados especialmente para ello, que están localizados dentro del edificio de ICAI, y permite hacerlo de una manera sencilla. Hasta el momento solamente se dispone de una FPGA en este laboratorio, pero en un futuro se pretenden aumentar los servicios ofrecidos por el mismo.

Mediante la FPGA es posible que los alumnos de la universidad puedan comprobar el funcionamiento de sus programas de VHDL sin necesidad de acudir a los laboratorios de ICAI, sino simplemente subiéndolos a Internet. Además, el alumno es capaz de interactuar con la placa, mediante botones en la pantalla y una cámara para observar lo que sucede en el laboratorio a tiempo real.

¿Cómo funciona?

En la página de inicio aparecerá el cuadro de la Figura 35.

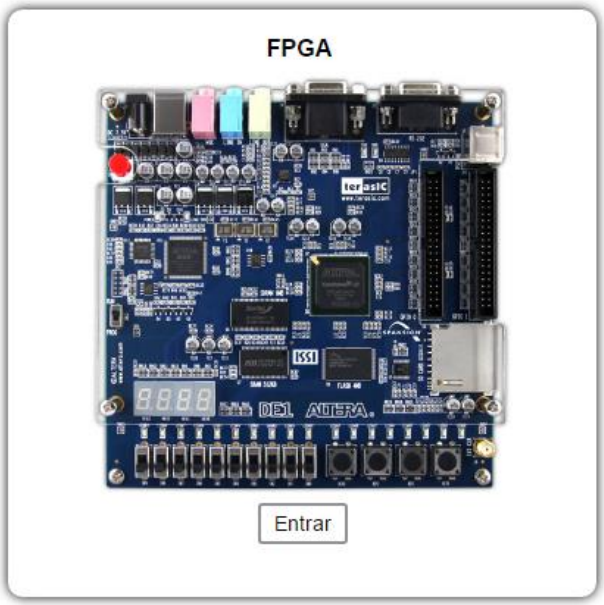


Figura 35. Cuadro de inicio

Al pulsar el botón "Entrar" se cargará la página en la que se debe de subir el archivo *.sof del proyecto (Figura 36).

Figura 36. Subida de archivos

Al subir el archivo pueden surgir diversos errores, bien sean al intentar guardar el archivo en el servidor o al cargarlo en la placa. En el primer caso (Figura 37), bastará con elegir de nuevo el archivo y volver a subirlo.

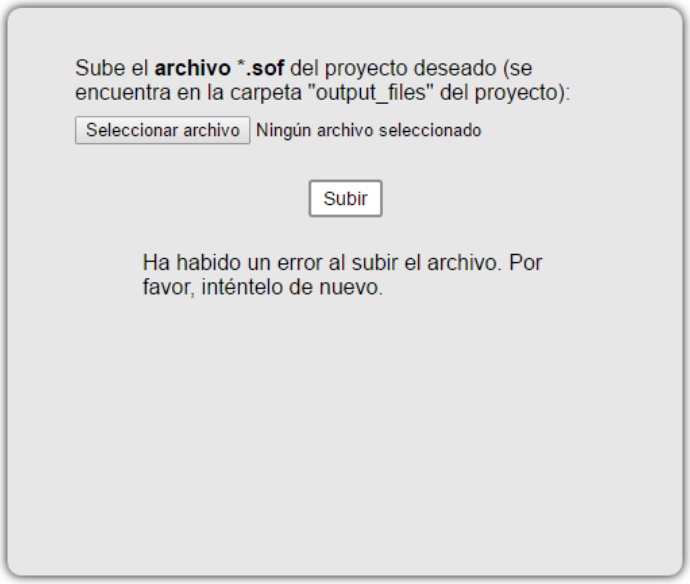


Figura 37. Error al guardar el archivo

En caso de que el error aparezca al cargar el archivo (Figura 38), es conveniente apretar el botón de ayuda y comprobar los fallos. Si el error se debe al archivo que se ha subido, puede volver a intentarlo.

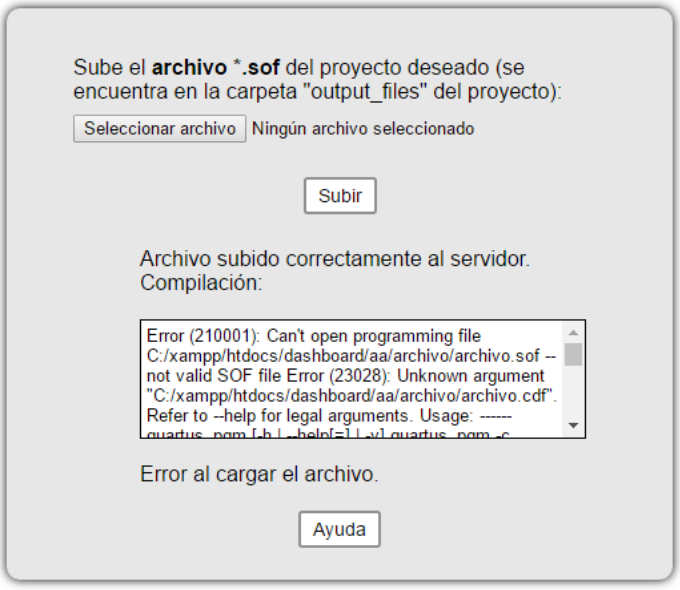


Figura 38. Error al cargar el archivo

Si no hay ningún error en el proceso, aparecerá el botón "Continuar". Al pulsarlo, se accederá a la ventana donde se puede interactuar con la placa, mostrada en la Figura 39. En ella se encuentran los siguientes elementos:

- Una ventana de vídeo, en la que se puede observar lo que sucede en la placa a tiempo real.
- 10 interruptores, mediante los cuales se pueden manipular los interruptores reales de la FPGA pulsando los botones de ON y OFF correspondientes.
- 4 pulsadores, mediante los cuales se pueden manipular los pulsadores reales de la FPGA pulsando los botones correspondientes. Por defecto se mantendrán pulsados durante un segundo.
- 4 cuadros de texto, mediante los cuales se pueden manipular los tiempos, en segundos, que permanecen activados los pulsadores.
- Un temporizador que indica el tiempo restante de inactividad para devolver al usuario a la página de inicio.

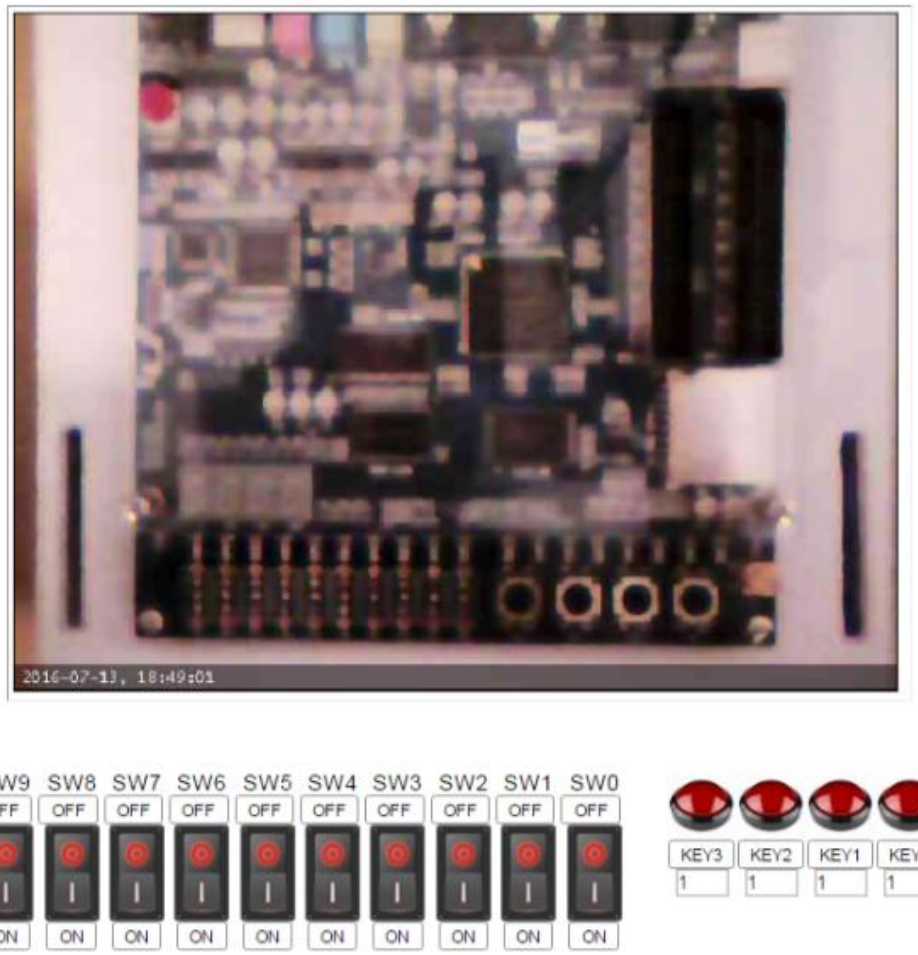


Figura 39. Página FPGA

Parte IV CÓDIGO FUENTE

Capítulo 1 ARCHIVO.CDF

```
/* Quartus II 64-Bit Version 13.0.1 Build 232 06/12/2013 Service Pack 1
SJ Web Edition */
JedecChain;
	FileRevision(JESD32A);
	DefaultMfr(6E);

	P ActionCode(Cfg)
		Device PartName(EP2C20F484)
Path("C:/xampp/htdocs/dashboard/aa/archivo/") File("archivo.sof")
MfrSpec(OpMask(1));

ChainEnd;

AlteraBegin;
	ChainType(JTAG);
AlteraEnd;
```

Capítulo 2 ARDUINO.INO

```
int option;
int puls0 = 2;
int puls1 = 3;
int puls2 = 4;
int puls3 = 5;
int auxp0 = 0;
int auxp1 = 0;
int auxp2 = 0;
int auxp3 = 0;
int switch0 = 6;
int switch1 = 7;
int switch2 = 8;
int switch3 = 9;
int switch4 = 10;
int switch5 = 11;
int switch6 = 12;
int switch7 = 13;
int switch8 = 14; //A0
int switch9 = 15; //A1

void setup(){
  Serial.begin(9600);
  pinMode(puls0, OUTPUT);
  pinMode(puls1, OUTPUT);
  pinMode(puls2, OUTPUT);
  pinMode(puls3, OUTPUT);
  digitalWrite(puls0, HIGH);
  digitalWrite(puls1, HIGH);
  digitalWrite(puls2, HIGH);
  digitalWrite(puls3, HIGH);
```

```
  pinMode(switch0, OUTPUT);
  pinMode(switch1, OUTPUT);
  pinMode(switch2, OUTPUT);
  pinMode(switch3, OUTPUT);
  pinMode(switch4, OUTPUT);
  pinMode(switch5, OUTPUT);
  pinMode(switch6, OUTPUT);
  pinMode(switch7, OUTPUT);
  pinMode(switch8, OUTPUT);
  pinMode(switch9, OUTPUT);
  digitalWrite(switch0, LOW);
  digitalWrite(switch1, LOW);
  digitalWrite(switch2, LOW);
  digitalWrite(switch3, LOW);
  digitalWrite(switch4, LOW);
  digitalWrite(switch5, LOW);
  digitalWrite(switch6, LOW);
  digitalWrite(switch7, LOW);
  digitalWrite(switch8, LOW);
  digitalWrite(switch9, LOW);
}

void loop(){

  if (Serial.available()>0){

    option=Serial.read();

    if(option=='a') {
      if(auxp0==1){
        digitalWrite(puls0, HIGH);
        Serial.println("PULS0 OFF");
        auxp0=0;
      }else if(auxp0==0){
        digitalWrite(puls0, LOW);
```

```
    Serial.println("PULS0 ON");
    auxp0=1;
  }
}

if(option=='s') {
  if(auxp1==1){
    digitalWrite(puls1, HIGH);
    Serial.println("PULS1 OFF");
    auxp1=0;
  }else if(auxp1==0){
    digitalWrite(puls1, LOW);
    Serial.println("PULS1 ON");
    auxp1=1;
  }
}

if(option=='d') {
  if(auxp2==1){
    digitalWrite(puls2, HIGH);
    Serial.println("PULS2 OFF");
    auxp2=0;
  }else if(auxp2==0){
    digitalWrite(puls2, LOW);
    Serial.println("PULS2 ON");
    auxp2=1;
  }
}

if(option=='f') {
  if(auxp3==1){
    digitalWrite(puls3, HIGH);
    Serial.println("PULS3 OFF");
    auxp3=0;
  }else if(auxp3==0){
```

```
    digitalWrite(puls3, LOW);
    Serial.println("PULS3 ON");
    auxp3=1;
  }
}

if(option=='0'){ //INTERRUPTOR 0 ON
  digitalWrite(switch0, HIGH);
  Serial.println("SWITCH0 ON");
}
if(option=='p'){ //INTERRUPTOR 0 OFF
  digitalWrite(switch0, LOW);
  Serial.println("SWITCH0 OFF");
}

if(option=='1'){
  digitalWrite(switch1, HIGH);
  Serial.println("SWITCH1 ON");
}
if(option=='q'){
  digitalWrite(switch1, LOW);
  Serial.println("SWITCH1 OFF");
}

if(option=='2'){
  digitalWrite(switch2, HIGH);
  Serial.println("SWITCH2 ON");
}
if(option=='w'){
  digitalWrite(switch2, LOW);
  Serial.println("SWITCH2 OFF");
}

if(option=='3'){
  digitalWrite(switch3, HIGH);
```

```
  Serial.println("SWITCH3 ON");
}
if(option=='e'){
  digitalWrite(switch3, LOW);
  Serial.println("SWITCH3 OFF");
}

if(option=='4'){
  digitalWrite(switch4, HIGH);
  Serial.println("SWITCH4 ON");
}
if(option=='r'){
  digitalWrite(switch4, LOW);
  Serial.println("SWITCH4 OFF");
}

if(option=='5'){
  digitalWrite(switch5, HIGH);
  Serial.println("SWITCH5 ON");
}if(option=='t'){
  digitalWrite(switch5, LOW);
  Serial.println("SWITCH5 OFF");
}

if(option=='6'){
  digitalWrite(switch6, HIGH);
  Serial.println("SWITCH6 ON");
}
if(option=='y'){
  digitalWrite(switch6, LOW);
  Serial.println("SWITCH6 OFF");
}

if(option=='7'){
  digitalWrite(switch7, HIGH);
```

```
      Serial.println("SWITCH7 ON");
    }if(option=='u'){
      digitalWrite(switch7, LOW);
      Serial.println("SWITCH7 OFF");
    }

    if(option=='8'){
      digitalWrite(switch8, HIGH);
      Serial.println("SWITCH8 ON");
    }
    if(option=='i'){
      digitalWrite(switch8, LOW);
      Serial.println("SWITCH8 OFF");
    }

    if(option=='9'){
      digitalWrite(switch9, HIGH);
      Serial.println("SWITCH9 ON");
    }
    if(option=='o'){
      digitalWrite(switch9, LOW);
      Serial.println("SWITCH9 OFF");
    }
  }
}
```

Capítulo 3 ARDUINO.PHP

```
<html>
<head>

<style type="text/css">
.botones{
    background:#FFFFFF;
    width:700px;
    height:180px;
    margin-right:auto;
    margin-left:auto;
    margin: 50px auto 0;
}

.boton{
    background-color: white;
    color: black;
    border: 2px solid #919191;
    padding: 3px 8px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 12px;
    border-radius: 3px;
}

.boton:hover{
    background-color: #919191;
    color: white;
}
```

```
</style>
</head>

<body>
<div class="botones">
<?php

    //Configuración y apertura del puerto serie para comunicación con
arduino
    exec("mode COM8 BAUD=9600 PARITY=N data=8 stop=1 xon=off");
    $fp = @fopen ("COM8", "w");

    include "interruptores.php";
    include "pulsadores.php";

?>
</div>
</body>
</html>
```

Capítulo 4 AYUDA.PHP

```
<html>
<head>
<style type="text/css">

.boton_medio{
    background-color: white;
    color: black;
    border: 2px solid #919191;
    padding: 4px 9px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 14px;
    border-radius: 3px;
}

.boton_medio:hover{
    background-color: #919191;
    color: white;
}

</style>

<script type="text/javascript">
    function abreventana(){
        window.open("ayuda fpga.php")
    }
</script>
</head>
```

```
<body>
<br>
<input onClick="javascript:abreventana()" style="float:left"
class="boton_medio" type="button" name="boton" value="Ayuda" />
<br><br>
</body>
</html>
```

Capítulo 5 AYUDA ERRORES.PHP

```
<html>
<body>
<font face="Arial">
<p>Compruebe si aparece alguno de los siguientes errores en el formulario de compilación: <br>
</p>
<p><li><b>Error (213013): Programming hardware cable not detected.</b> El cable conectado a la placa se ha desconectado.<br>
</p>
<p><li><b>Error (210001): Can't open programming file.</b> El archivo subido no es correcto. Compruebe que el archivo elegido es el *.sof del proyecto (localizado en la carpeta <i>output_files</i>).<br>
</p>
<p><li><b>Error (209040): Can't access JTAG chain.</b> La placa está apagada en ese momento.<br>
</p>
<p><li><b>Error (209012): Operation failed.</b> Es posible que la placa esté apagada.<br>
</p>
<p><li><b>Error (210009): Device is not supported.</b> Es posible que el archivo subido esté dañado. Por favor, vuelva a intentarlo para comprobarlo.<br>
</p>
<p><li><b>Error (23028): Unknown argument.</b> Es posible que el archivo subido esté dañado. Por favor, vuelva a intentarlo para comprobarlo.
</p></font>
</body>
</html>
```

Capítulo 6 AYUDA FPGA.PHP

```
<html>
<head>
<style type="text/css">
    .foto{
        border: solid 1px #000000;
    }
</style>
</head>

<body>
<img src="imagenes/pantalla fpga.png" class="foto">
</body>
</html>
```

Capítulo 7 BORRAR ARCHIVO.PHP

```
<?php
array_map('unlink', glob("C:/xampp/htdocs/dashboard/aa/archivo/*.sof"));
?>
```

Capítulo 8 CABECERA.PHP

```
<html>
<head>
<style type="text/css">
.titulo{
    background:#FFFFFF;
    width:220px;
    height:50px;
    margin: 0px auto 0;
    margin-left:0px;
}

.info{
    background:#FFFFFF;
    width:25px;
    height:25px;
    margin: 0px auto 0;
    margin-left:10px;
}

.comillas{
    background:#FFFFFF;
    width:150px;
    height:75px;
    margin: 0px auto 0;
    margin-right:0px;

}

</style>
</head>
```

```
<body>
<center><div class="titulo" style="float:left">
<a href="http://172.20.3.13/dashboard/aa/inicio.php" target="_self">
<img src="imagenes/icai.png" width="50px" height="50px"/>
<font face="Arial" size="5"><b>WebLab-ICAI</b></font>
</a>
</div></center>

<div class="info" style="float:left">
<br>
<a href="http://172.20.3.13/dashboard/aa/info.php" target="_self">
<img src="imagenes/info.jpg" width="25px" height="25px"/>
</a>
</div>

<div class="comillas">
<a href="http://www.comillas.edu/es" target="_self"><img
style="float:right" src="imagenes/comillas.png" width="150px"
height="75px"/></a>
</div>

<hr color="#919191" size="2"/>
</body>
</html>
```

Capítulo 9 CAJA FPGA.PHP

```
<html>
<head>
<style type="text/css">

.caja{
    background:#FFFFFF;
    box-shadow:0px 0px 10px black;
    width:400px;
    height:400px;
    margin-right:auto;
    margin-left:auto;
    margin: 50px auto 0;
    border-radius:10px;
}

.boton_medio{
    background-color: white;
    color: black;
    border: 2px solid #919191;
    padding: 4px 9px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 14px;
    border-radius: 3px;
}

.boton_medio:hover{
    background-color: #919191;
    color: white;
```

```
}


</style>
</head>


<body>
<div class="caja">
<form enctype="multipart/form-data">
<center>
<br><font face="Arial"><b>FPGA</b></font> <br>
<img src="imagenes/fpga.jpg" width="300px" height="300px"><br>
<a href="http://172.20.3.13/dashboard/aa/subir.php" target="_self">
<input type="button" class="boton_medio" name="boton" value="Entrar" />
</a>
</center>
</form>
</div>
</body>
</html>
```

Capítulo 10 CAMARA.PHP

```
<html>
<body>
    <center><iframe onload="iFrameHeight()"
        id="camara"
        src="http://172.20.3.2:8081/"
        width="650"
        height="500"
        scrolling="no"
        frameborder="1"
        class="wrapper">
        Sin marcos
    </iframe></center>
</body>
</html>
```

Capítulo 11 ERRORES.PHP

```
<html>
<head>
<script type="text/javascript">
    function abreventana(){
        window.open("ayuda errores.php")
    }
</script>
</head>

<body>
<br>
<center><input onClick="javascript:abreventana()" class="boton_medio"
type="button" name="boton" value="Ayuda" /></center>
</body>
</html>
```

Capítulo 12 FPGA.PHP

```
<html>
<body>

<?php
    $fc=fopen("archivo.txt","w+");
    fwrite($fc, "abierto");
    fclose($fc);
    include "cabecera.php";
    include "volver.php";
    include "ayuda.php";
    include "camara.php";
    include "arduino.php";
    include "temporizador.php";
?>

</body>
</html>
```

Capítulo 13 INFO.PHP

```
<html>
<head>
<style type="text/css">
    .foto{
        border: solid 1px #000000;
    }
</style>
</head>

<body>
<?php
    include "cabecera.php";
?>
<br>
<font face="Arial" size=5><b>¿Qué es WebLab-ICAI?</b></font>
<font face="Arial">
<p align="justify">
WebLab-ICAI es un proyecto basado en el laboratorio web creado en la
Universidad de Deusto,
<a href="weblab.deusto.es">WebLab-Deusto.</a>
Este proyecto consiste en un sitio web que permite interactuar con
elementos físicos, habilitados especialmente para ello, que están
localizados dentro del edificio de ICAI, y permite hacerlo de una manera
sencilla. Hasta el momento solamente se dispone de una FPGA en este
laboratorio, pero en un futuro se pretenden aumentar los servicios
ofrecidos por el mismo.
</p>
<p align="justify">
Mediante la FPGA es posible que los alumnos de la universidad puedan
comprobar el funcionamiento de sus programas de VHDL sin necesidad de
acudir a los laboratorios de ICAI, sino simplemente subiéndolos a
Internet. Además, el alumno es capaz de interactuar con la placa,
```

```
mediante botones en la pantalla y una cámara para observar lo que sucede
en el laboratorio a tiempo real.
</p>
</font>
<br>

<font face="Arial" size=5><b>¿Cómo funciona?</b></font>
<br><br>
<font face="Arial">
<center>
En la página de inicio aparecerá el siguiente cuadro:
<br>
<img src="imagenes/caja fpga.png">
<br><br>
Al pulsar el botón "Entrar" se cargará la pagina en la que se debe de
subir el archivo *.sof del proyecto.
<br>
<img src="imagenes/subir archivo.png">
<br><br>
Al subir el archivo pueden surgir diversos errores, bien sean al intentar
guardar el archivo en el servidor o al cargarlo en la placa. En el primer
caso, bastará con elegir de nuevo el archivo y volver a subirlo.
<br>
<img src="imagenes/error subir.png">
<br><br>
En caso de que el error aparezca al cargar el archivo, es conveniente
apretar el botón de ayuda y comprobar los fallos. Si el error se debe al
archivo que se ha subido, puede volver a intentarlo.
<br>
<img src="imagenes/error cargar.png">
<br><br>
Si no hay ningún error en el proceso, aparecerá el botón "Continuar". Al
pulsarlo, se accederá a la ventana donde se puede interactuar con la
placa.
<br><br>
<img src="imagenes/pantalla fpga.png" class="foto">
```

```
<br>
</center>
</font>
</body>
</html>
```

Capítulo 14 INICIO.PHP

```
<html>
<body>
<?php
    include "cabecera.php";
    include "borrar archivo.php";
    include "caja fpga.php";
?>
</body>
</html>
```

Capítulo 15 INTERRUPTORES.PHP

```
<html>
<body>

<?php
    //se almacena en las variables el contenido de los archivos
correspondientes, que indican el estado anterior de las imágenes que
representan a los interruptores

    $f9=fopen("interruptores\int9.txt","r");
    $var9=fread($f9,20);
    fclose($f9);

    $f8=fopen("interruptores\int8.txt","r");
    $var8=fread($f8,20);
    fclose($f8);

    $f7=fopen("interruptores\int7.txt","r");
    $var7=fread($f7,20);
    fclose($f7);

    $f6=fopen("interruptores\int6.txt","r");
    $var6=fread($f6,20);
    fclose($f6);

    $f5=fopen("interruptores\int5.txt","r");
    $var5=fread($f5,20);
    fclose($f5);

    $f4=fopen("interruptores\int4.txt","r");
    $var4=fread($f4,20);
```

```
    fclose($f4);

    $f3=fopen("interruptores\int3.txt","r");
    $var3=fread($f3,20);
    fclose($f3);

    $f2=fopen("interruptores\int2.txt","r");
    $var2=fread($f2,20);
    fclose($f2);

    $f1=fopen("interruptores\int1.txt","r");
    $var1=fread($f1,20);
    fclose($f1);

    $f0=fopen("interruptores\int0.txt","r");
    $var0=fread($f0,20);
    fclose($f0);
?>

<center>
<form method="post" id="interruptores" name="interruptores"
action="<?=$_SERVER['PHP_SELF']?>">

<!--Divisiones para cada grupo de elementos que representan los
interruptores-->

<!--SWITCH 9-->
<div id="int9" style="float:left">
<font face="Arial">SW9</font><br>
<input type="submit" class="boton" name="sw9off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var9?>" width="40px" height="70px"
id="fotoi9"/><br>
<input type="submit" class="boton" name="sw9on" value="ON" />
</div>

<!--SWITCH 8-->
```

```
<div id="int8" style="float:left">
<font face="Arial">SW8</font><br>
<input type="submit" class="boton" name="sw8off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var8?>" width="40px" height="70px"
id="fotoi8"/><br>
<input type="submit" class="boton" name="sw8on" value="ON" />
</div>

<!--SWITCH 7-->
<div id="int7" style="float:left">
<font face="Arial">SW7</font><br>
<input type="submit" class="boton" name="sw7off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var7?>" width="40px" height="70px"
id="fotoi7"/><br>
<input type="submit" class="boton" name="sw7on" value="ON" />
</div>

<!--SWITCH 6-->
<div id="int6" style="float:left">
<font face="Arial">SW6</font><br>
<input type="submit" class="boton" name="sw6off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var6?>" width="40px" height="70px"
id="fotoi6"/><br>
<input type="submit" class="boton" name="sw6on" value="ON" />
</div>

<!--SWITCH 5-->
<div id="int5" style="float:left">
<font face="Arial">SW5</font><br>
<input type="submit" class="boton" name="sw5off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var5?>" width="40px" height="70px"
id="fotoi5"/><br>
<input type="submit" class="boton" name="sw5on" value="ON" />
</div>

<!--SWITCH 4-->
```

```
<div id="int4" style="float:left">
<font face="Arial">SW4</font><br>
<input type="submit" class="boton" name="sw4off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var4?>" width="40px" height="70px"
id="fotoi4"/><br>
<input type="submit" class="boton" name="sw4on" value="ON" />
</div>

<!--SWITCH 3-->
<div id="int3" style="float:left">
<font face="Arial">SW3</font><br>
<input type="submit" class="boton" name="sw3off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var3?>" width="40px" height="70px"
id="fotoi3"/><br>
<input type="submit" class="boton" name="sw3on" value="ON" />
</div>

<!--SWITCH 2-->
<div id="int2" style="float:left">
<font face="Arial">SW2</font><br>
<input type="submit" class="boton" name="sw2off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var2?>" width="40px" height="70px"
id="fotoi2"/><br>
<input type="submit" class="boton" name="sw2on" value="ON" />
</div>

<!--SWITCH 1-->
<div id="int1" style="float:left">
<font face="Arial">SW1</font><br>
<input type="submit" class="boton" name="sw1off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var1?>" width="40px" height="70px"
id="fotoi1"/><br>
<input type="submit" class="boton" name="sw1on" value="ON" />
</div>

<!--SWITCH 0-->
```

```
<div id="int0" style="float:left">
<font face="Arial">SW0</font><br>
<input type="submit" class="boton" name="sw0off" value="OFF" /> <br>
<img src="imagenes/<?php echo $var0?>" width="40px" height="70px"
id="fotoi0"/><br>
<input type="submit" class="boton" name="sw0on" value="ON" />
</div>
</form>
</center>

<?php

    //Envío por puerto serie al pulsar interruptores+actualización de
imágenes

    //SWITCH 9
    if(isset($_POST["sw9on"])){
        fwrite($fp,"9");
        $var9="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi9.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw9off"])){
        fwrite($fp,"o");
        $var9="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi9.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }
```

```
    //SWITCH 8
    if(isset($_POST["sw8on"])){
        fwrite($fp,"8");
        $var8="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi8.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw8off"])){
        fwrite($fp,"i");
        $var8="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi8.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 7
    if(isset($_POST["sw7on"])){
        fwrite($fp,"7");
        $var7="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi7.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw7off"])){
        fwrite($fp,"u");
```

```
        $var7="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi7.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 6
    if(isset($_POST["sw6on"])){
        fwrite($fp,"6");
        $var6="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi6.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw6off"])){
        fwrite($fp,"y");
        $var6="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi6.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 5
    if(isset($_POST["sw5on"])){
        fwrite($fp,"5");
        $var5="interruptor_on.jpg";
        ?>
```

```
        <script type="text/javascript">

document.interruptores.fotoi5.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw5off"])){
        fwrite($fp,"t");
        $var5="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi5.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 4
    if(isset($_POST["sw4on"])){
        fwrite($fp,"4");
        $var4="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi4.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw4off"])){
        fwrite($fp,"r");
        $var4="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi4.src="imagenes/interruptor_off.jpg"
        </script>
```

```
        <?php
    }

    //SWITCH 3
    if(isset($_POST["sw3on"])){
        fwrite($fp,"3");
        $var3="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi3.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw3off"])){
        fwrite($fp,"e");
        $var3="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi3.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 2
    if(isset($_POST["sw2on"])){
        fwrite($fp,"2");
        $var2="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi2.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
```

```
    if(isset($_POST["sw2off"])){
        fwrite($fp,"w");
        $var2="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi2.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 1
    if(isset($_POST["sw1on"])){
        fwrite($fp,"1");
        $var1="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi1.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw1off"])){
        fwrite($fp,"q");
        $var1="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi1.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //SWITCH 0
    if(isset($_POST["sw0on"])){
        fwrite($fp,"0");
```

```
        $var0="interruptor_on.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi0.src="imagenes/interruptor_on.jpg"
        </script>
        <?php
    }
    if(isset($_POST["sw0off"])){
        fwrite($fp,"p");
        $var0="interruptor_off.jpg";
        ?>
        <script type="text/javascript">

document.interruptores.fotoi0.src="imagenes/interruptor_off.jpg"
        </script>
        <?php
    }

    //Se guarda el estado de las imágenes de los interruptores para poder
leerlos en la siguiente ejecución

    $f9=fopen("interruptores\int9.txt","w+");
    fwrite($f9, $var9);
    fclose($f9);

    $f8=fopen("interruptores\int8.txt","w+");
    fwrite($f8, $var8);
    fclose($f8);

    $f7=fopen("interruptores\int7.txt","w+");
    fwrite($f7, $var7);
    fclose($f7);

    $f6=fopen("interruptores\int6.txt","w+");
    fwrite($f6, $var6);
```

```
    fclose($f6);

    $f5=fopen("interruptores\int5.txt","w+");
    fwrite($f5, $var5);
    fclose($f5);

    $f4=fopen("interruptores\int4.txt","w+");
    fwrite($f4, $var4);
    fclose($f4);

    $f3=fopen("interruptores\int3.txt","w+");
    fwrite($f3, $var3);
    fclose($f3);

    $f2=fopen("interruptores\int2.txt","w+");
    fwrite($f2, $var2);
    fclose($f2);

    $f1=fopen("interruptores\int1.txt","w+");
    fwrite($f1, $var1);
    fclose($f1);

    $f0=fopen("interruptores\int0.txt","w+");
    fwrite($f0, $var0);
    fclose($f0);
?>
</body>
</html>
```

Capítulo 16 PROGRAMAR.PHP

```
<html>
<head>
<style type="text/css">
    .scroll{
        border : solid 1px #000000;
        background : #FFFFFF;
        color : #000000;
        padding : 4px;
        width : 325px;
        height : 80px;
        overflow : auto;
    }
</style>
</head>

<body>

<!--Scroll donde se muestra la salida de la programación-->
<div class="scroll">
<font size="2">
<?php

    // Programación de la placa
    $ultima_linea = system('C:/altera/13.0sp1/quartus/bin/quartus_pgm -c
USB-Blaster C:/xampp/htdocs/dashboard/aa/archivo/archivo.cdf', $retval);

    echo '
    </pre>
    <hr/>Ultima linea de la salida: ' . $ultima_linea . '
    <hr/>Valor de retorno: ' . $retval;
```

```
    echo "<hr/>";
?>
</font>
</div>

<?php
    if(isset($retval)){
        if ($retval==0){
            include 'redireccion.php';
        }else{
        echo "</br>Error al cargar el archivo.</br>";
        include "errores.php";
        include "borrar archivo.php";
        }
    }
?>
</body>
</html>
```

Capítulo 17 PULSADORES.PHP

```
<html>
<head>

<!--Funciones en JS para cambiar las imágenes de los pulsadores-->
<script type="text/javascript">
    function cambiarFotop0(){
        document.pulsadores.fotop0.src="imagenes/pulsador_on.jpg";
    }
    function cambiarFotop1(){
        document.pulsadores.fotop1.src="imagenes/pulsador_on.jpg";
    }
    function cambiarFotop2(){
        document.pulsadores.fotop2.src="imagenes/pulsador_on.jpg";
    }
    function cambiarFotop3(){
        document.pulsadores.fotop3.src="imagenes/pulsador_on.jpg";
    }
</script>
</head>

<body>
<center>
<form method="post" id="pulsadores" name="pulsadores">

<?php
    //Asignación de tiempos de pulsación

    if (isset($_POST['temp0'])){
        $tiempo_p0=$_POST['temp0'];
    }else{
```

```
        $tiempo_p0="1";
    }

    if (isset($_POST['temp1'])){
        $tiempo_p1=$_POST['temp1'];
    }else{
        $tiempo_p1="1";
    }

    if (isset($_POST['temp2'])){
        $tiempo_p2=$_POST['temp2'];
    }else{
        $tiempo_p2="1";
    }

    if (isset($_POST['temp3'])){
        $tiempo_p3=$_POST['temp3'];
    }else{
        $tiempo_p3="1";
    }

?>

<!--Divisiones que agrupan los componentes que representan a cada
pulsador-->

<!--PULS0-->
<div id "puls0" style="float:right">
<img src="imagenes/pulsador_off.jpg" width="50px" height="50px"
id="fotop0"/><br>
<input type="submit" class="boton" name="p0" value="KEY0"
onClick="javascript:cambiarFotop0();" /><br>
<input type="text" size="1" maxlength="2" value="<?php echo $tiempo_p0?>"
name="temp0"/>
</div>
```

```
<!--PULS1-->
<div id="puls1" style="float:right">
<img src="imagenes/pulsador_off.jpg" width="50px" height="50px"
id="fotop1"/><br>
<input type="submit" class="boton" name="p1" value="KEY1"
onClick="javascript:cambiarFotop1();" /><br>
<input type="text" size="1" maxlength="2" value="<?php echo $tiempo_p1?>"
name="temp1"/>
</div>

<!--PULS2-->
<div id="puls2" style="float:right">
<img src="imagenes/pulsador_off.jpg" width="50px" height="50px"
id="fotop2"/><br>
<input type="submit" class="boton" name="p2" value="KEY2"
onClick="javascript:cambiarFotop2();" /><br>
<input type="text" size="1" maxlength="2" value="<?php echo $tiempo_p2?>"
name="temp2"/>
</div>

<!--PULS3-->
<div id="puls3" style="float:right">
<img src="imagenes/pulsador_off.jpg" width="50px" height="50px"
id="fotop3"/><br>
<input type="submit" class="boton" name="p3" value="KEY3"
onClick="javascript:cambiarFotop3();" /><br>
<input type="text" size="1" maxlength="2" value="<?php echo $tiempo_p3?>"
name="temp3"/>
</div>

</form>
</center>

<?php
    //Escritura por el puerto serie y temporizadores

    //Pulsador 0
```

```
if(isset($_POST["p0"])){
    fwrite($fp,"a");
    $segundos_p0=$tiempo_p0;
    while($segundos_p0!=0){
        $segundos_p0-=1;
        sleep(1);
    }
    fwrite($fp,"a");
}

//Pulsador 1
if(isset($_POST["p1"])){
    fwrite($fp,"s");
    $segundos_p1=$tiempo_p1;
    while($segundos_p1!=0){
        $segundos_p1-=1;
        sleep(1);
    }
    fwrite($fp,"s");
}

//Pulsador 2
if(isset($_POST["p2"])){
    fwrite($fp,"d");
    $segundos_p2=$tiempo_p2;
    while($segundos_p2!=0){
        $segundos_p2-=1;
        sleep(1);
    }
    fwrite($fp,"d");
}

//Pulsador 3
if(isset($_POST["p3"])){
    fwrite($fp,"f");
```

```
        $segundos_p3=$tiempo_p3;
        while($segundos_p3!=0){
            $segundos_p3-=1;
            sleep(1);
        }
        fwrite($fp,"f");
    }
?>
</body>
</html>
```

Capítulo 18 REDIRECCION.PHP

```
<html>
<body>
<br>
<center><a href="http://172.20.3.13/dashboard/aa/fpga.php"
target="_self"> <input type="button" name="boton" value="Continuar" />
</a></center>
</body>
</html>
```

Capítulo 19 SUBIR ARCHIVO.PHP

```
<html>
<head>
<style type="text/css">
.caja{
    background:#e7e7e7;
    box-shadow:0px 0px 10px black;
    width:500px;
    height:430px;
    margin-right:auto;
    margin-left:auto;
    border-radius:10px;
}

form{
    margin: 40px auto 0;
    width: 400px;
}

label{
    display: block;
}

input[type="file"]{
    display: block;
    margin: 8px 0;
}

div.resultado{
    margin: 25px auto 0;
    width: 300px;
```

```
}

</style>
</head>

<body>
<div class="caja">
<form action="" method="post" enctype="multipart/form-data">
<br><br><font face="Arial">Sube el <b>archivo *.sof</b> del proyecto
deseado (se encuentra en la carpeta "output_files" del proyecto):
<input type="file" name="archivo" id="archivo" /> <br>
<center><input type="submit" class="boton_medio" name="boton"
value="Subir" /></center>
</form>
</div>

<div class="resultado">

<?php
    if(isset($_POST['boton'])){

        // Se cambia el nombre del archivo
        $nombre_archivo = "archivo.sof";

        // Se comprueba si ha habido error
        if ($_FILES["archivo"]["error"] > 0) {
            echo "Ha habido un error al subir el archivo. Por favor,
inténtelo de nuevo.";
        } else {
            // Se comprueba si el archivo ya existe
            if (file_exists("C:/xampp/htdocs/dashboard/aa/archivo/" .
$nombre_archivo)) {
                echo "Ha habido un error al subir el archivo. Por favor,
inténtelo de nuevo.";
                include "borrar archivo.php";
            } else {
```

```
                // Se sube el archivo a la carpeta
                move_uploaded_file($_FILES["archivo"]["tmp_name"],
                "C:/xampp/htdocs/dashboard/aa/archivo/" .
$nombre_archivo);
                echo "Archivo subido correctamente al servidor.
Compilación:<br/><br/>";
                include 'programar.php';
            }
        }
    }
?>
</div>
</font>
</body>
</html>
```

Capítulo 20 SUBIR.PHP

```
<html>
<head>
<style type="text/css">
.boton_medio{
    background-color: white;
    color: black;
    border: 2px solid #919191;
    padding: 4px 9px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 14px;
    border-radius: 3px;
}

.boton_medio:hover{
    background-color: #919191;
    color: white;
}
</style>
</head>

<body>
<?php
    include "cabecera.php";
    include "volver.php";
    include "subir archivo.php";
?>
</body>
</html>
```

Capítulo 21 TEMPORIZADOR.PHP

```
<html>
<body>
<form name="redirect" id="redirect">
<center>
<font face="Arial">La ventana se cerrará tras cinco minutos de
inactividad. <br />Tiempo restante: </font>
<form>
<font face="Arial"><input type="text" size="1" name="redirect2" /></font>
</form>
<font face="Arial">segundos.</font>
</center>
<script type="text/javascript">
var targetURL="http://172.20.3.13/dashboard/aa/inicio.php"
var countdownfrom=300
var currentsecond=document.redirect.redirect2.value=countdownfrom+1
function countredirect(){
    if (currentsecond!=1){
        currentsecond-=1
        document.redirect.redirect2.value=currentsecond
    }else{
        window.location=targetURL
        return
    }
    setTimeout("countredirect()",1000)
}
countredirect()
</script>
</form>
</body>
</html>
```

Capítulo 22 VOLVER.PHP

```
<html>
<head>
<style type="text/css">
    .boton_grande{
    background-color: white;
    color: black;
    border: 2px solid #919191;
    padding: 5px 10px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    border-radius: 3px;
}

.boton_grande:hover{
    background-color: #919191;
    color: white;
}
</style>
</head>

<body>
<a href="http://172.20.3.13/dashboard/aa/inicio.php" target="_self">
<input type="button" class="boton_grande" name="boton" value="Volver a
inicio" /> </a>
<br><br>
</body>
</html>
```