



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

TESIS PARA OBTENER EL GRADO DE:
LICENCIADO EN TECNOLOGÍA

TÍTULO:

IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE DISPOSITIVOS
ELECTRÓNICOS DEL AULA DEL FUTURO, CENTRADO EN EL USUARIO, A TRAVÉS
DE UNA INTERFAZ WEB

PRESENTA
ANDRIC VALDEZ VALENZUELA

DIRECTOR DE TESIS
DR. FERNANDO GAMBOA RODRÍGUEZ

NOVIEMBRE 2016

JURIQUILLA, QUERÉTARO

AGRADECIMIENTOS

Quiero agradecer especialmente a mis padres Eric Ismael y María del Rosario por su gran apoyo, paciencia y motivación para poder realizar este trabajo de tesis, siempre estuvieron ahí, alentándome para terminarla. Gracias a ustedes y a mis hermanos soy lo que soy ahora. Muchas gracias, los quiero!

Gracias a mis amigos de la licenciatura Hugo, Toño, Búho, Cris, Irving, Itzel, Bofos, Lalo, Chucho, Eric, Eleazar y más, por los grandes momentos de diversión y de trabajo que pasamos juntos.

Quiero agradecer a Daniela Diez por siempre estar ahí dándome apoyo y ánimos a lo largo de esta tesis, y también por los grandes momentos que hemos pasado juntos. Te quiero!

A mis sinodales y profesores Rafael Chávez, Domingo Rangel y Beatriz Millán, por que cada uno me dio asignaturas de las que más me gustaron y más aprendí, y que marcaron el rumbo de mi área de interés y trabajo. Muchas Gracias!

A mi tutor Fernando Gamboa y mi sinodal Gustavo de la Cruz, en conjunto con todos los investigadores del CCADET Libi, Ricardo, Clarita y más, muchas gracias por sus consejos , apoyo y asesoramiento a lo largo de este trabajo.

Índice General

1. INTRODUCCIÓN	5
1.1 Descripción del problema	
1.2 Hipótesis	
1.3 Objetivo general	
1.4 Objetivos particulares	
2. DISEÑO CENTRADO EN EL USUARIO	10
2.1 El Diseño Centrado en el Usuario	
2.2 Análisis de la tarea	
2.3 Pruebas de usabilidad	
3. TECNOLOGÍAS WEB	19
3.1 Redes	
3.2 Modelo de referencia TCP/IP	
3.3 Protocolos de transporte	
3.4 Dirección de red y dirección física	
3.5 Arquitectura cliente-servidor	
3.6 Protocolos de aplicación	
3.7 Tecnologías Web del lado del cliente	
3.8 Tecnologías Web del lado del servidor	
3.9 Sistemas embebidos	
4. DISEÑO DEL SISTEMA DE CONTROL CENTRADO EN EL PROFESOR ..	49
4.1 Análisis de la tarea del profesor	
4.2 Requerimientos del sistema	
4.3 Diseño de la solución	

5. DESARROLLO DE LA SOLUCIÓN	56
5.1 Descripción general del funcionamiento del sistema	
5.2 Desarrollo de Front-end	
5.2.1 Componentes de la interfaz Web	
5.2.2 Encendido y Apagado de las ZIT	
5.2.3 Proyección de archivos en ZIT	
5.2.4 Programación de ZIT	
5.2.5 Configuraciones en el sistema	
5.3 Desarrollo de Back-end	
5.3.1 Control de computadoras en LAN	
5.3.2 Control de proyectores en LAN	
6. PRUEBAS CON USUARIOS Y ANÁLISIS DE RESULTADOS	90
CONCLUSIONES	99
REFERENCIAS	101
ANEXOS	104
Algoritmo implementado para el control de computadoras en LAN	
Algoritmo implementado para el control de proyectores en LAN	
Documentos para pruebas con usuarios	

CAPÍTULO 1

INTRODUCCIÓN

En las últimas décadas hemos sido testigos de cambios vertiginosos e innovadores en el desarrollo de las tecnologías de la información y comunicación (TIC). No obstante, si valoramos la forma en que dichos desarrollos tecnológicos han impactado la educación, en el sentido de hasta dónde están propiciando un cambio profundo en los paradigmas educativos (la forma en que se aprende y se enseña), los cambios parecen ser más bien modestos [1].

Es en este contexto que surge el proyecto educativo “el Aula del Futuro” en el Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET) de la UNAM, en el que se pretende integrar tecnologías que permitan modernizar y hacer más flexibles los espacios de aprendizaje-enseñanza. Para lograr esto, dicho proyecto propone rediseñar los espacios de aprendizaje bajo un enfoque distinto al tradicional; esto es, introducir tecnologías bajo un profundo estudio del usuario y sus tareas dentro de un aula educativa, de modo que permita anteponer las necesidades y objetivos del usuario a los aspectos técnicos que éstas puedan suponer.

Bajo este enfoque, la literatura especializada identifica tres roles principales a analizar: el profesor, el alumno y la institución; cada uno de estos tendrá nuevas tareas y responsabilidades que antes ninguno tenía que asumir [2, 3-5].

Por otro lado, este nuevo enfoque implica pasar de un aprendizaje centrado en la enseñanza (como es actualmente) a un aprendizaje centrado en el alumno, donde éste tendrá la libertad de dirigir su propio aprendizaje. El rol del profesor en este contexto será de orientar y asesorar a los alumnos en el proceso de aprendizaje que ellos mismo irán construyendo [2].

En este proyecto de tesis nos enfocaremos principalmente en el rol del profesor y sus tareas en este nuevo paradigma, con el objetivo de que se obtenga un máximo beneficio de las soluciones tecnológicas existentes en el Aula del Futuro, lo que a la postre se traducirá en más tiempo de asesoramiento y orientación con los alumnos y menos tiempo con aspectos técnicos en el manejo y uso de tecnologías.

1. Introducción

1.1 Descripción del problema

Actualmente el Aula del Futuro (AF) cuenta con una gran variedad de dispositivos electrónicos, como los son proyectores, computadoras, dispositivos de audio y video (micrófonos, cámaras, etc), pantallas táctiles, televisores, etc. El diagrama de la *figura 1.1* corresponde a un plano o vista aérea de lo que actualmente es el AF, en el que se muestran los dispositivos electrónicos antes mencionados, así como su distribución en el aula.

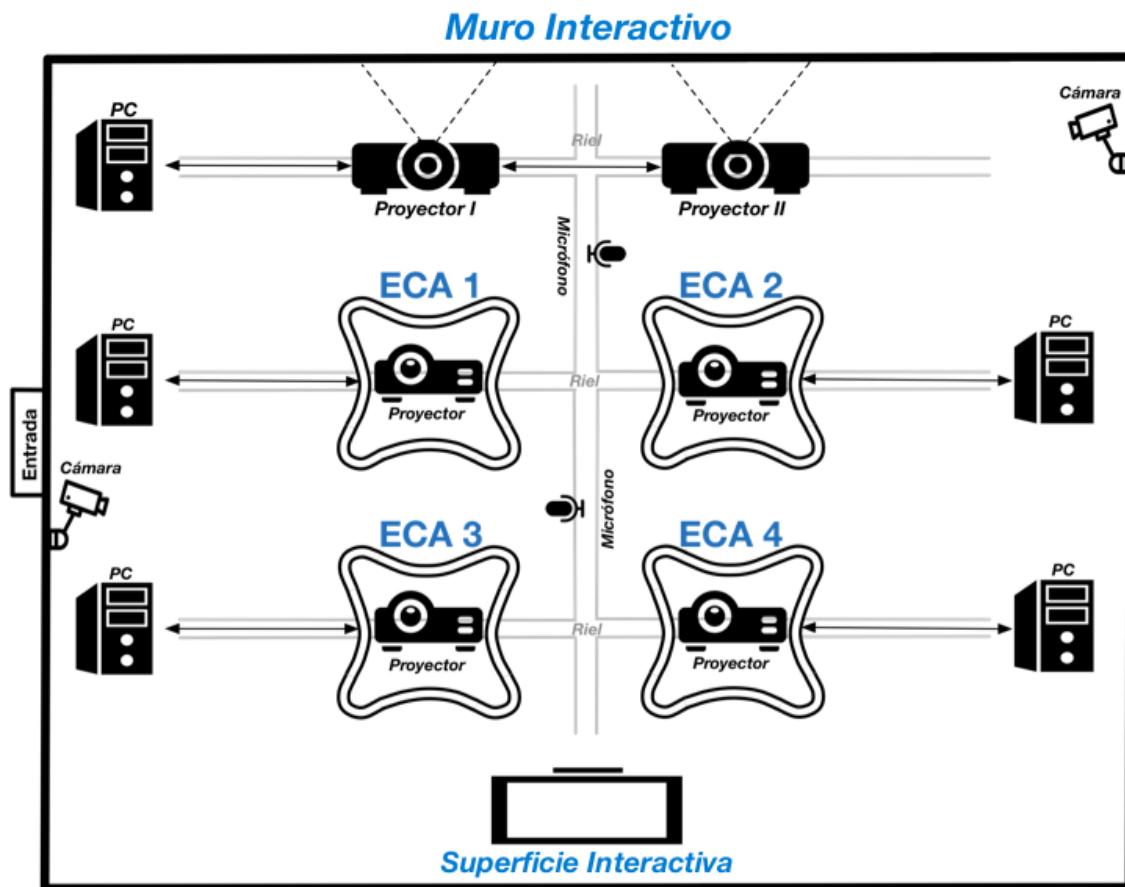


Figura 1.1. Plano aéreo del Aula del Futuro.

En las *figuras 1.1 y 1.2* se muestran las principales tecnologías con las que cuenta actualmente el AF: el muro interactivo, cuatro escritorios colaborativos aumentados (ECA) y una superficie interactiva. Cada una de estas tecnologías define una zona interactiva de trabajo (ZIT), con un espacio de proyección asociado. Estas zonas son:

1. Introducción

- **El Muro Interactivo:** Una de las paredes del aula, de tres metros de alto y nueve metros de ancho, se utiliza para proyectar una imagen a partir de dos proyectores de tiro ultra corto conectados a una computadora.
- **Escritorio Colaborativo Aumentado (ECA):** Uno de los proyectos insignia del Aula, se basa en la proyección de un espacio virtual de trabajo sobre un escritorio. Esta proyección se hace desde el techo hacia el escritorio (mesa) y requiere de un proyector y de una computadora de escritorio para su operación. Actualmente se cuenta con cuatro espacios de este tipo.
- **Superficie interactiva:** Mesa interactiva que permite hasta doce puntos de toque y posibilita a múltiples usuarios intervenir en un problema.

El techo del aula contiene rieles (tipo canaleta) utilizados principalmente para el montaje de proyectores, *access point* (para conexión a una red), micrófonos, cámaras y tomas de corriente. Además, estos rieles se utilizan como vía y repositorio de los cables usados para la alimentación eléctrica de los dispositivos, así como cables para el intercambio de información con otros dispositivos.

Para hacer uso de las tecnologías presentes en el AF, el profesor debe llevar a cabo tareas que se dividen en muchas operaciones puntuales. Por ejemplo, cuando el profesor requiera hacer uso del Muro Interactivo en unas de sus clases, debe realizar la tarea de encendido tanto de la computadora como de los dos proyectores; posteriormente, cargar el material preparado a la computadora y desplegarlo/proyectarlo (aplicación, presentación, video, etc.). Una vez finalizada su cátedra, debe apagar tanto la computadora como los dos proyectores.

Se puede pensar que llevar a cabo las tareas requeridas para el uso de una sola ZIT, como el Muro Interactivo descrito arriba, no represente una tarea compleja o tediosa para el profesor. El problema se presenta cuando se requiere el uso de dos o más ZIT dispersos en el aula. Esto es, el uso de los cuatro ECA, del Muro Interactivo, de los dispositivos de audio, de video, además de los nuevos desarrollos que constantemente se están integrando al AF, hace evidente que conlleva a una sofisticación del espacio cuyo manejo puede, eventualmente, distraer al profesor, alejándolo de sus objetivos originales.

1. Introducción

Es por ello que surge la idea de integrar una solución que ponga a disposición del profesor una manera sencilla de controlar e interactuar con estos dispositivos o tecnologías sin preocuparse por cómo funcionan, o del procedimiento para activarlos o desactivarlos, y que se enfoque en sus objetivos y desarrollo de sus tareas.

Como solución a esta problemática, en este trabajo de tesis se propone desarrollar un sistema de control de dispositivos electrónicos bajo un enfoque de diseño centrado en el usuario.

Esto es, realizar un análisis de la tarea del usuario con el fin de determinar todas las tareas o actividades que tiene que llevar a cabo el profesor para alcanzar sus objetivos o metas dentro del aula, prestando especial atención en aquellas tareas asociadas al uso de elementos tecnológicos y que pudieran representar un problema de acceso y manejo para el profesor. Posteriormente, con base al análisis de su tarea, se propone diseñar, desarrollar e implementar un sistema de control que permita al profesor, desde una interfaz Web, acceder de una forma rápida y sencilla a los elementos tecnológicos del AF.

Por último, con el propósito de probar el correcto funcionamiento del sistema y que además cumpla con los objetivos propuestos, se harán pruebas de usabilidad (evaluaciones) con usuarios reales (profesores) dentro del AF.

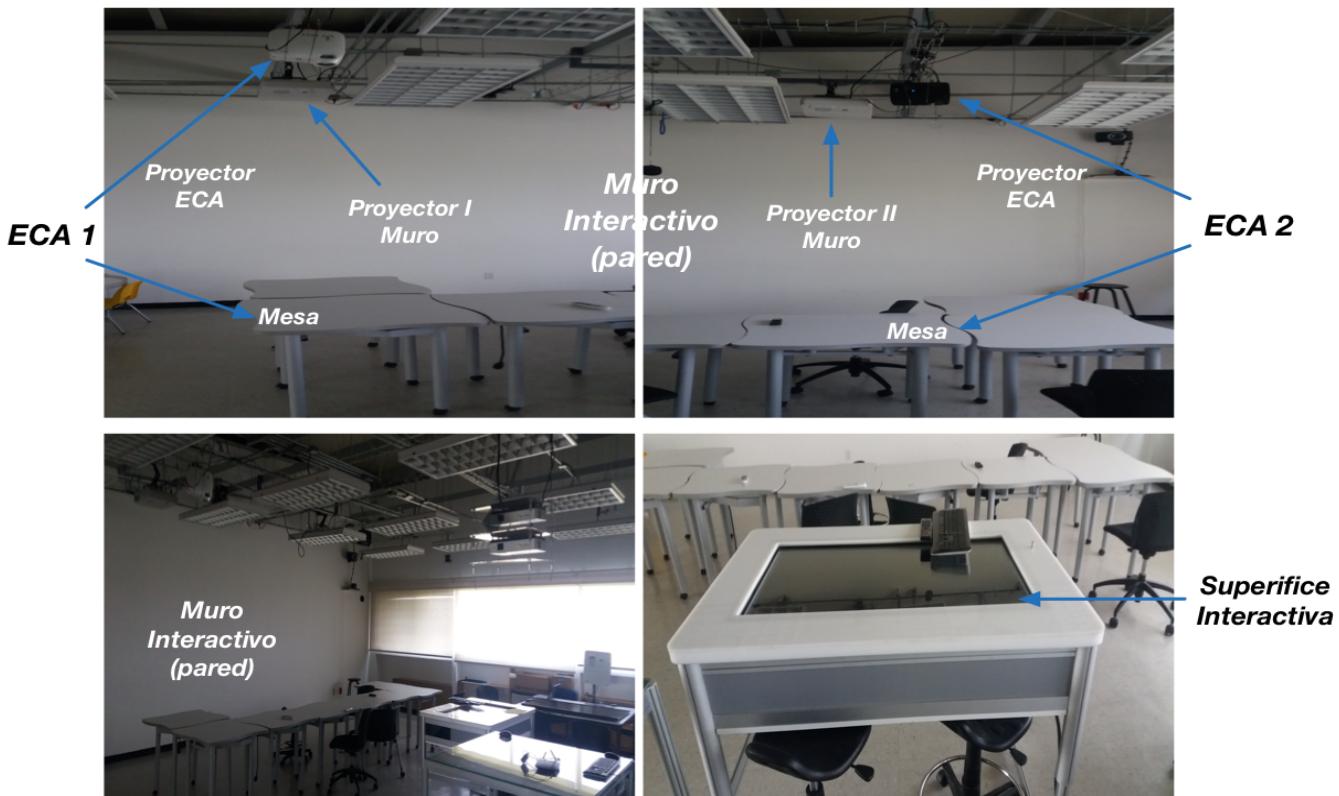


Figura 1.2. Imágenes del AF donde se muestran las ZIT: Muro Interactivo, ECA's y Superficie Interactiva.

1. Introducción

1.2 Hipótesis

El desarrollo de un sistema que controle el encendido y apagado de todos los elementos tecnológicos asociados a una tarea o necesidad específica del profesor, le permitirá concentrarse en sus objetivos y, por esa vía, hacer un mejor uso del Aula del Futuro.

1.3 Objetivo general

Implementar un sistema de control de dispositivos electrónicos del Aula del Futuro que permita al usuario (profesor), con base en sus necesidades y tareas dentro del aula, una forma rápida y sencilla de acceder a ellos.

1.4 Objetivos particulares

- Determinar los requerimientos de control dentro del aula a partir del análisis del usuario.
- Diseñar una solución que responda a las necesidades del usuario a través del análisis de las tecnologías disponibles.
- Desarrollar e implementar un sistema de control de dispositivos electrónicos del aula desde una interfaz Web.
- Verificación del funcionamiento correcto del sistema de control.
- Realizar pruebas con el usuario final.

CAPÍTULO 2

DISEÑO CENTRADO EN EL USUARIO

El Diseño Centrado en el Usuario (DCU) se puede definir como un enfoque de diseño cuyo proceso está dirigido por información sobre las personas que van hacer uso de un producto. Es decir, el proceso de diseño y desarrollo del producto se basa en las necesidades e intereses del usuario, con el objetivo de hacer productos que le sean usables y comprensibles [6]. Donald Norman, ingeniero y diseñador, introduce en su libro *The Design of EveryDay* el término DCU y describe que en todo proceso de diseño existen tres modelos mentales¹ que deben ser comprendidos [7]:

- **Modelo del diseño:** también llamado modelo conceptual, es la conceptualización o idea que el diseñador tiene en mente acerca del funcionamiento u operación de un sistema o dispositivo (ej. una computadora, una silla, una interfaz Web, etc.).
- **Modelo del usuario:** el modelo mental que se genera el usuario acerca de cómo funciona el artefacto que está utilizando, creado a partir de su interacción con el sistema.
- **Imagen del sistema:** es el resultado de la estructura física construida a partir del modelo del diseño (*figura 2.1*).

Idealmente, el **modelo del usuario** y el **modelo del diseño** son equivalentes, en el sentido que el diseñador ha logrado captar a la perfección las necesidades del usuario y la manera como comprende la tarea a ejecutar. Bajo este supuesto, la imagen mental que genera el usuario del sistema también corresponde a la perfección con la configuración real del dispositivo. Infortunadamente, este caso nunca se alcanza debido a las diferencias que existen entre la manera en que el diseñador comprende el problema a resolver, y la visión que de éste tiene el usuario.

¹ Un modelo mental se define como un proceso del pensamiento de las personas acerca de cómo funciona algo en el mundo real [64].

2. Diseño Centrado en el Usuario

El usuario y el diseñador sólo se comunican a través del artefacto creado (su apariencia física, su operación, la forma en que responde, etc.). Dado que de manera inevitable existen diferencias entre el modelo del diseñador y el del usuario, el usuario termina con un modelo mental más o menos erróneo (dependiendo de cuán grandes son las diferencias) y por lo tanto hará un uso incorrecto o limitado del sistema o dispositivo.

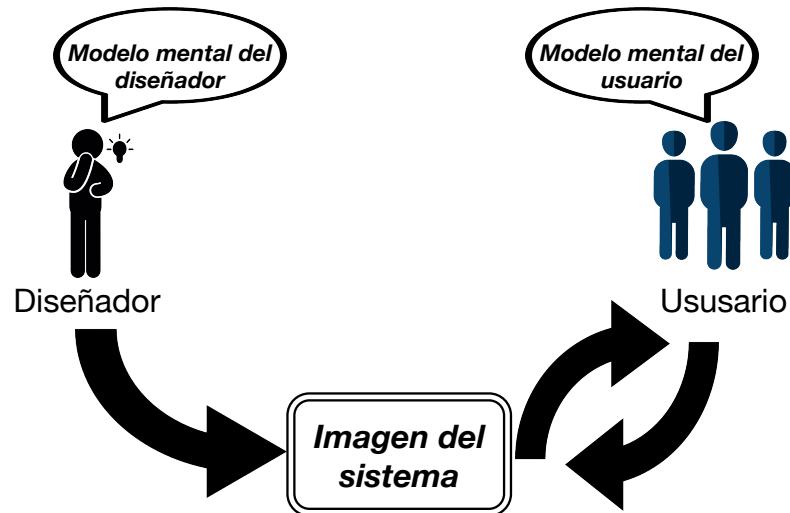


Figura 2.1. Modelos conceptuales involucrados en un proceso de diseño.

El DCU se interesa en desarrollar los modelos y las metodologías que ayuden al diseñador a construir un modelo conceptual o de diseño que sea apropiado y comprensible para el usuario, capturando así las partes importantes de la operación del sistema y haciéndolas evidentes y fácilmente comprensibles para el usuario.

En efecto, el DCU busca asegurar que el usuario comprenda qué hacer y qué está pasando en todo momento, prescindiendo tanto como sea posible de instrucciones o etiquetas, entrenamientos complejos o manuales sin fin. Si en algún momento la explicación sobre cómo funciona un dispositivo conduce al usuario a pensar “¿cómo voy a recordar eso?”, es un indicativo de que el diseño ha fallado [7].

2. Diseño Centrado en el Usuario

2.1 El Diseño Centrado en el Usuario

De modo a desarrollar un producto usable, comprensible y que satisfaga las necesidades del usuario, el DCU se debe pensar como un proceso cíclico y de fases. Este proceso contempla cuatro fases principales más un ciclo de iteración [8]. En la *figura 2.2* se muestra un diagrama de este proceso.

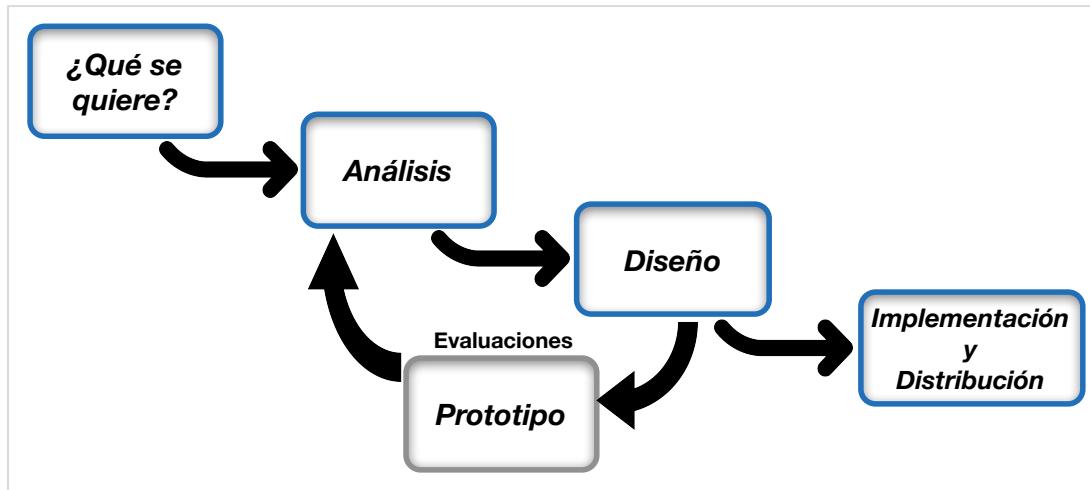


Figura 2.2. Fases en el proceso del DCU.

A continuación se explican cada una de las fases (de Izq. a Der.):

- **El usuario y su contexto**

En esta fase se identifica a los usuarios a los que se dirige el producto, para qué lo usarán y en qué condiciones. Con base en ello, se busca establecer lo más precisamente posible, qué necesita y en qué contexto.

- **Análisis**

En esta fase se hace un estudio del usuario de modo de identificar sus objetivos y metas dentro del área en el que se desenvuelven. Un método ampliamente utilizado es el llamado *análisis de la tarea* el cual describe cómo una actividad debe ser desarrollada con el fin de alcanzar una meta de usuario. Este método se describe en secciones posteriores del capítulo.

2. Diseño Centrado en el Usuario

- **Diseño**

Es difícil obtener el diseño correcto durante la primera aproximación a la solución. Por ello, el DCU propone generar y evaluar diferentes prototipos, para analizar sus ventajas y problemas, y hacer mejoras correspondientes. La evaluación de los prototipos puede ir desde el análisis a bocetos hechos en papel, a versiones tempranas del sistema, para ir hasta versiones más o menos acabadas del mismo.

Todas ellas deben, preferentemente, ser probadas con usuarios reales. Sin embargo, dado que a veces resulta costo en tiempo y dinero realizar evaluaciones con usuarios reales, una técnica auxiliar es la evaluación heurística, la cual consiste en que un grupo de expertos inspeccionan y analizan el diseño en busca de potenciales problemas de usabilidad. Algunos tipos de pruebas de usabilidad para evaluación de prototipos de diseño se describen en secciones posteriores del capítulo.

- **Implementación y distribución**

La última fase corresponde a la implementación y distribución, la cual se logra una vez que se ha determinado que el diseño evaluado satisfizo los requisitos o necesidades iniciales.

2.2 Análisis de la tarea

Esta etapa corresponde a la fase 2 de un DCU y se define como un proceso de análisis de las actividades que realizan usuarios específicos con el fin de alcanzar sus metas dentro de su entorno [8, 9]. Con este análisis es posible que el diseñador capture cuáles son las posibles intenciones y objetivos de los usuarios y así desarrollar soluciones de diseño que respondan a sus necesidades, y consecuentemente, realizar productos usables, comprensibles y que le sirvan de apoyo en el desarrollo de sus tareas.

Una tarea se define como la actividad que tiene que ser realizada con el propósito de alcanzar una meta. Una meta es una modificación deseada del estado de un sistema que el usuario espera obtener. Por lo tanto una tarea está asociada a una meta, y cada meta tiene asociada una o múltiples tareas.

2. Diseño Centrado en el Usuario

El análisis de la tarea, además, permite determinar y entender los siguientes cuestionamientos [10]:

- Cuáles son las metas de los usuarios. Qué están tratando de lograr.
- Qué hacen los usuarios actualmente para alcanzar sus metas.
- Cómo los usuarios son influenciados por su entorno físico.
- Cómo influye su experiencia previa en el desarrollo de sus tareas.
- Cómo es el flujo de trabajo que siguen para llevar acabo sus tareas.

Existen diferentes técnicas o métodos para realizar el análisis de la tarea del usuario. Entre las dos más relevantes se encuentran:

- **Familia GOMS**

Se basa en el mecanismo de razonamiento humano para la resolución de problemas y realiza la formalización de aquellas actividades (físicas y mentales) que intervienen en esa labor. Para cada tarea se describe el objetivo a satisfacer (Goal), el conjunto de operaciones (Operations) que el sistema pone a disposición del usuario para la interacción, los métodos disponibles para llevar a cabo esas operaciones (Methods) y por último, un conjunto de reglas de selección (Selection) para determinar la alternativa más conveniente en cada caso (descritas mediante estructuras de control if–then). Cada tarea se podría descomponer en otras tareas primitivas formando un árbol jerárquico [11].

- **Análisis Jerárquico de Tareas**

Mejor conocido en la literatura por sus siglas en inglés HTA (Hierarchical Task Analysis), este análisis se centra en descomponer las tareas del usuario en sus sub-tareas siguiendo un orden jerárquico, así como describir en qué orden y en qué condiciones se realizan éstas. La jerarquía de tareas puede ser representado en forma de diagrama así como textualmente [8].

La descomposición de tareas puede llevarse a cabo mediante las siguientes etapas [10]:

- ✓ Identificar la tarea (meta) que debe analizarse.
- ✓ Descomponer cada tarea en entre 3 y 8 sub-tareas. Estas deben especificarse en términos de objetivos y, entre ellos, debe cubrir toda el área de interés.
- ✓ Dibujar un diagrama de tareas en capas para cada sub-tarea.

2. Diseño Centrado en el Usuario

- ✓ Decidir hasta qué nivel de detalle se va a descomponer una sub-tarea de modo que se pueda asegurar consistencia en todos los ámbitos.
- ✓ Presentar el análisis a alguien que no ha participado en la descomposición, pero que conoce las tareas lo suficientemente bien como para comprobar la consistencia.

Como modo de ejemplo supongamos que nuestra meta o tarea principal es la de cocinar arroz. A continuación se presenta en forma textual un análisis jerárquico de las tareas (HTA) que se deben llevar a cabo con el fin de alcanzar esta meta:

0. Cocinar Arroz (SEC). Cumplir esta meta implica llevar a cabo las siguientes cinco tareas siguiendo un orden secuencial.
 1. Calentar arroz en agua (SEC). Las siguientes tres tareas se deben realizar de forma secuencial para alcanzar esta tarea.
 - 1.1 Hervir agua
 - 1.2 Agregar el agua hirviendo al arroz
 - 1.3 Tirar el agua después de 10 minutos
 2. Freír arroz (PAR). Las siguientes cuatro tareas se deben realizar de forma paralela para alcanzar esta tarea.
 - 2.1 Calentar cazuella
 - 2.2 Calentar aceite
 - 2.3 Agregar arroz
 - 2.4 Esperar hasta que se dore
 3. Agregar agua al arroz frito (SEC). Las siguientes dos tareas se deben realizar de forma secuencial para alcanzar esta tarea.
 - 3.1 Hervir agua
 - 3.2 Agregar el agua hirviendo al arroz frito
 4. Evaporar agua al arroz (SEC). Las siguientes dos tareas se deben realizar de forma secuencial para alcanzar esta tarea.
 - 4.1 Mantener en el fuego el arroz
 - 4.2 Esperar a que se evapore el agua
 5. Revolver al arroz (SEC). Las siguientes dos tareas se deben realizar de forma secuencial para alcanzar esta tarea.

2. Diseño Centrado en el Usuario

5.1 Tomar una cuchara

5.2 Con cuidado invertir el arroz desde la orillas

Dado que esta última técnica (HTA) es la más sencilla y la más utilizada para realizar un análisis de la tarea del usuario, será la que se implemente en este trabajo de tesis.

Para el caso que nos ocupa, el análisis de la tarea se hará para un profesor que imparte clases en el Aula del Futuro. Con ello se pretende determinar las tareas que realiza actualmente que impliquen el uso de tecnología y así definir los requerimientos de control de dispositivos del aula.

2.3 Pruebas de usabilidad

El término usabilidad lo podemos definir como un atributo de calidad de un producto o servicio y se refiere a qué tan fácil puede ser utilizado por las personas con el fin de alcanzar sus objetivos [12].

Lo que hace a un producto o servicio usable es la ausencia de frustración cuando se usa. Esto es, que el usuario puede hacer lo que quiere de la forma en que espera hacerlo (de acuerdo a su expectativa), sin obstáculo alguno, dudas o preguntas.

Un producto se considera verdaderamente usable cuando cumple con ciertos requerimientos, los cuales se mencionan a continuación [12]:

- **Utilidad.** Se refiere al grado en que un producto permite a un usuario alcanzar sus metas, y es una valoración de qué tan dispuesto esté el usuario de usarlo.
Si un sistema es fácil de usar, fácil de aprender, o incluso satisface al usuario, pero si no permite alcanzar las metas de usuarios específicos, no será un sistema que se use.
- **Eficiencia.** Es la rapidez con la que las metas del usuarios pueden ser alcanzadas con exactitud.
- **Efectividad.** Se refiere al grado en que el producto se comporta de la manera en que los usuarios esperan y la facilidad con la que los usuarios pueden utilizarlo con el fin de hacer lo que se proponen.
- **Fácil de aprender.** Es una parte de la eficacia y tiene que ver con la capacidad del usuario para operar el sistema a un cierto nivel de competencia definido después de una cierta

2. Diseño Centrado en el Usuario

cantidad preestablecida y período de formación (que puede ser muy poco tiempo). También puede referirse a la capacidad de los usuarios poco frecuentes de volver a utilizar el sistema de forma correcta después de períodos de inactividad.

- **Satisfacción.** Se refiere a la percepción del usuario, sentimientos, y opiniones del producto, usualmente tomadas de preguntas orales o escritas. Es más probable que los usuarios acepten productos que resuelvan sus necesidades y les provea satisfacción, que otros que no lo hagan.
- **Accesibilidad** En un amplio sentido, se refiere a acceder a los productos necesarios de modo de lograr una meta y qué tan disponibles están para el usuario.

Las pruebas de usabilidad corresponden a la fase 3 del proceso del DCU y se llevan acabo una vez que se presente una solución de diseño y se realice un prototipo del producto o servicio. Estas pruebas se realizan con usuarios representativos con el fin de evaluar la usabilidad del prototipo. Usualmente, durante una prueba, los participantes intentarán completar ciertas tareas (dirigidas por un moderador) mientras son grabadas todas sus acciones mediante dispositivos de audio y video, y a la vez, un observador escucha y toma notas.

El objetivo de estas pruebas es el de identificar cualquier problema de usabilidad, colectar datos cualitativos y cuantitativos y lograr la satisfacción de los usuarios con el producto.

Los beneficios de estas pruebas permiten al diseñador y al equipo de desarrollo identificar problemas y corregirlos antes de que el producto sea desarrollado en su totalidad, traduciéndose en reducciones de tiempo y costos.

Con el fin de realizar pruebas de usabilidad efectivas y que provean una buena retroalimentación, es recomendable tener en cuenta los siguientes puntos [13]:

- ✓ Ver si los usuarios son capaces de completar correctamente tareas específicas.
- ✓ Identificar cuánto tiempo le toma completar dichas tareas.
- ✓ Determinar qué tan satisfechos están los usuarios con el producto o sistema.
- ✓ Identificar cambios que se requieran realizar con el fin de mejorar el rendimiento y satisfacción de los usuarios.

2. Diseño Centrado en el Usuario

Por otro lado, los documentos que se presentan a continuación sirven de apoyo en la preparación, ejecución y evaluación de las pruebas de usabilidad con usuarios.

- **Cuestionario de entrada.** Es usado para determinar la pericia (o grado de conocimiento de un tema) del usuario. En este cuestionario se realizan preguntas concretas con el fin de obtener un perfil de cada uno de los participantes.
- **Protocolo de bienvenida.** Documento que el moderador lee al usuario al inicio de la prueba indicándole el propósito de la reunión. El cual es evaluar el prototipo de un sistema mediante el estudio de sus acciones y comentarios, y con base a ellos, realizar mejoras en él.
- **Hipótesis de la tarea.** Describe las actividades o tareas que el usuario va a realizar en las pruebas (por pantalla) y lo que se espera que este sea capaz de hacer en cada una. Así como el tiempo estimado en el que el usuario debe completar cada tarea.
- **Actividades de la evaluación.** Este documento se crea a partir de la hipótesis de la tarea y contiene cada una de las instrucciones y tareas que el moderador aplicará en las pruebas con usuarios.
- **Cuestionario de salida (de usabilidad).** Este documento (utilizado en esta tesis) se conoce como Escala de la Usabilidad del Sistema (SUS, por sus siglas en inglés) y es usado para medir la usabilidad de un producto o servicio. Consiste en un cuestionario de 10 preguntas (respondidas por los usuarios) con cinco opciones de respuesta cada una, que van desde frecuentemente de acuerdo a frecuentemente en desacuerdo. Este cuestionario se ha convertido en un estándar en la industria, permitiendo la evaluación de una amplia variedad de productos o servicios, incluidos sitios Web, aplicaciones móviles, etc [14].

Para el desarrollo de esta tesis, una vez realizada una versión o prototipo funcional del sistema de control de dispositivos, se pasará a la parte de pruebas con usuarios reales, en este caso con los profesores que actualmente imparten clases en el Aula del Futuro. Ellos tendrán que completar tareas específicas con el propósito de observar su interacción con el sistema. Con ello se busca evaluar al sistema desarrollado, en el sentido de si verdaderamente representa una herramienta útil y de apoyo en las tareas que realiza dentro del aula que impliquen el uso de tecnología.

CAPÍTULO 3

TECNOLOGÍAS WEB

En el presente capítulo se dan a conocer las tecnologías involucradas en el desarrollo del sistema de control de dispositivos, así como una breve explicación de sus características, funcionalidades y conceptos importantes en relación a éstas. Además, se menciona cómo serán implementadas o utilizadas con el fin de solucionar algunas tareas dentro del sistema.

3.1 REDES

Una red la podemos definir como un conjunto de sistemas interconectados entre sí y que son capaces de intercambiar información y/o recursos entre ellos. Cada uno de estos sistemas se les conoce como nodo de red, el cual se puede clasificar en: nodo de conmutación y nodo terminal. Un nodo de conmutación es aquel que se utiliza para comunicar diferentes redes y sirven como una vía o camino para la transmisión de información entre nodos terminales (también conocido como *Router*). Por otro lado un nodo terminal o simplemente nodo es el que genera y emite datos (emisor) y el que los recibe como destino final (receptor). A los nodos también se les conoce en la literatura como *host*, máquina o equipo, y que generalmente son computadoras, pero puede ser cualquier dispositivo que forme parte de la red (por ejemplo *Hub*, *Switch*, *Router*, *Access Point*, microcontroladores, etc) [15].

Para llevar a cabo el intercambio de información entre dos nodos es necesario establecer un medio de transmisión, lo cuales pueden ser físico o no físico.

Los medios de transmisión físicos, comúnmente llamados alámbricos, utilizan algún tipo de cable para transmisión de datos o señales, por ejemplo el cable RJ45 o cable de red/Ethernet es el medio físico más usado para conectar un *host* a una red física; otro ejemplo es el cable RS232 utilizado para el intercambio de datos de forma serial entre un DTE (Equipo Terminal de Datos) y un DCE (Equipo de Comunicación de Datos).

Por otro lado están los medios de transmisión no físicos, también llamados inalámbricos y son aquellos en los que las señales se propagan por el vacío o el aire; debido a esto es necesario el uso de antenas para la transmisión y recepción de las señales. Algunos ejemplos de este medio de

3. Tecnologías Web

transmisión son el Bluetooth, Infrarrojo y Wi-fi. Este último es el medio más utilizado actualmente para conectar cualquier dispositivo (móviles, laptop, tabletas, impresores, etc) a una red de área local y poder comunicarse con otros dispositivos conectados a esa misma red o a redes externas.

Por otra parte, a una red, generalmente, la podemos clasificar de acuerdo a su tecnología de transmisión y escala o extensión geográfica [16].

Tecnología de transmisión

Esta tecnología describe la forma en que se transfieren los bits de datos (paquetes de información) en una red. Hablando en sentido general, existen dos tipos de tecnología de transmisión que se emplean ampliamente hoy en día: los enlaces de difusión y los enlaces punto a punto.

- **Enlaces de difusión**

Una red de difusión es aquella en la que todas las máquinas en la red comparten el canal de comunicación; los paquetes que envía una máquina son recibidos por todos los demás. Existe un campo de dirección en cada paquete enviado por la red que especifica a quién va dirigido éste (campo de dirección establecido por la capa de red en el modelo TCP/IP); cuando una máquina recibe el paquete, verifica el campo de dirección, si la verificación es correcta, esto es, si la dirección del paquete es igual a la dirección de *host*, esta lo procesa, si no, simplemente lo ignora. Por otro lado, estos enlaces de difusión brindan la posibilidad de envío de paquetes a todos los destinos (*host*) en la red para que los reciban y procesen, esto es posible mediante un código especial en el campo de dirección del paquete comúnmente llamado dirección de *broadcasting* (*figura 3.1 (c)*). Algunos sistemas de difusión también soportan la transmisión a un subconjunto de máquinas o *host* (*figura 3.1 (b)*), lo cual se conoce como multidifusión (*multicasting*).

- **Enlaces punto a punto**

Este tipo de enlace conecta pares individuales de máquinas; que al contrario al enlace de difusión solo existe un canal de transmisión que comunica únicamente dos nodos

3. Tecnologías Web

o máquinas (*figura 3.1 (a)*). A la transmisión punto a punto en donde solo hay un emisor y un receptor se le conoce como unidifusión (*unicasting*).

En la *figura 3.1* se muestra una representación de enlaces de transmisión antes mencionados.

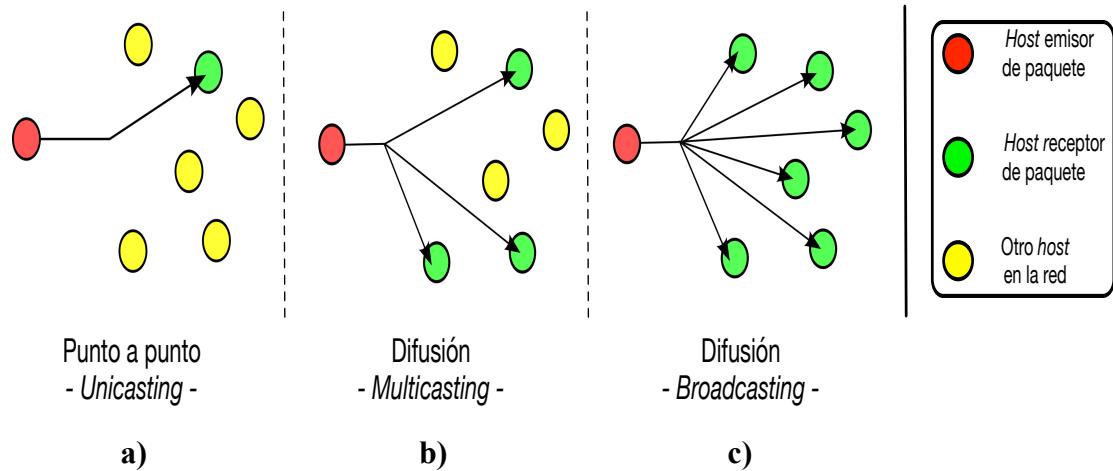


Figura 3.1. Representación esquemática para enlaces de transmisión de difusión y punto a punto.

Para el desarrollo del sistema de control de dispositivos, se hace uso de ambas tecnologías de transmisión. Para el encendido/apagado de computadoras y proyectores a través de una red local se implementa una comunicación de difusión tipo *broadcasting* y una comunicación punto a punto (*unicasting*), respectivamente.

Escala o extensión geográfica

Ahora se mencionan cada una de las clasificaciones de redes según su escala o extensión geográfica.

- **Redes de Área Personal**, generalmente llamadas **PAN** (Personal Area Network) permiten la comunicación local entre máquinas dentro de un rango de un metro cuadrado. Un ejemplo de esta red de corto alcance el **Bluetooth**, mediante el cual es posible conectar

3. Tecnologías Web

varios periféricos (teclado, mouse, impresora) a una PC de forma inalámbrica o conectar unos audífonos a un teléfono móvil sin usar cables.

- **Redes de Área Local**, generalmente llamadas **LAN** (Local Area Network), son redes que permiten la comunicación de máquinas dentro del rango de los 10 m hasta 1 km y operan dentro de un cuarto, edificio o una fábrica. Estas redes se utilizan ampliamente para acceder a recursos compartidos tales como impresoras, archivos digitales o servidores.
- **Redes de Área Metropolitana**, o **MAN** (Metropolitan Area Network) permiten la comunicación dentro de un rango de 10 km. Un ejemplo de este tipo de redes es el de televisión por cable disponibles en muchas ciudades.
- **Redes de Área Amplia**, o **WAN** (Wide Area Network) permiten la comunicación dentro de un rango desde los 100 hasta los 1000 km. Estas redes abarcan una extensa área geográfica como un país o continente. Un ejemplo de uso de este tipo de redes en una empresa con sucursales en distintas ciudades, por lo que es necesario que los recursos (acceso a bases de datos, documentos digitales, etc) de cualquier sucursal, sea accesible para todas las demás.
- **Internet**, es una colección de redes interconectadas y abarca el rango de los 10,000 km. Esta red de redes nos permite comunicar una máquina desde una red, por ejemplo LAN, con otra máquina en otra red distante LAN, situada en cualquier parte de mundo.

En la *figura 3.2* se muestra las características para las clasificaciones de redes antes mencionadas.

Para este proyecto de tesis, todas las comunicaciones entre las partes que componen al sistema desarrollado (como computadoras, proyectores, sistemas embebidos, etc.) se lleva acabo dentro de una red de área local (LAN) localizada en el Aula del Futuro.

3. Tecnologías Web

Distancia entre procesadores	Procesadores ubicados en el (la) mismo(a)	Ejemplo
1 m	Metro cuadrado	Red de área personal
10 m	Cuarto	
100 m	Edificio	Red de área local
1 km	Campus	
10 km	Ciudad	Red de área metropolitana
100 km	País	
1000 km	Continente	Red de área amplia
10 000 km	Planeta	Internet

Figura 3.2. Clasificación de redes según su escala. La primer columna llamada “Distancia entre procesadores” representa el tamaño de la red en metros o kilómetros. En la segunda columna se muestran un ejemplos geográficos de acuerdo al tamaño de la red.

3.2 Modelo de referencia TCP/IP

Este modelo de referencia es el más utilizado hoy en día para el intercambio de información entre dos máquinas enlazadas a una red, sin importar el lugar geográfico donde residen y sin necesidad de conocer la forma de trabajar de la máquina subyacente o la tecnología de red de cada sistema. La arquitectura TCP/IP fue creada por los estadounidenses Robert Elliot Kahn y Vinton Cerf a principios de los años 70, pero fue hasta los primeros años de los años 80 que se aplicó este protocolo a la red de investigación llamada ARPANET (Advanced Research Projects Agency Network) con la finalidad de brindar un servicio más seguro y robusto. Esta red era patrocinada por el Departamento de Defensa de Estados Unidos (DoD) y utilizada principalmente para la comunicación entre instituciones académicas y estatales [15-19].

Este modelo, de modo de reducir su complejidad de diseño, se organiza como una pila de capas o niveles donde cada una está construida a partir de la que está abajo. El propósito de cada capa es ofrecer servicios y soporte a las capas superiores, es decir, cada capa es un tipo de máquina virtual (también conocido como interfaz entre capa) que ofrece ciertos servicios a la capa que está encima de ella. Cada una de las capas incluye una gran variedad de protocolos diferentes enfocados a distintos tipos de aplicaciones y servicios; estos protocolos establecen una comunicación virtual

3. Tecnologías Web

entre capas de cada máquina en el proceso de intercambio de información en una red. A continuación se mencionan las capas que conforman este modelo, así como una breve descripción de su funcionalidad y los protocolos más comunes y usados en cada una de ellas.

- **Capa de Aplicación.** En esta capa se encuentran los protocolos de alto nivel que proporcionan servicios específicos a los usuarios de la red. Ejemplo de protocolo: HTTP, FTP, DNS, WOL.
- **Capa de Transporte.** Esta capa es la encargada de realizar y mantener una comunicación entre equipos finales. Ejemplo de protocolo: TCP, UDP.
- **Capa de Internet o de Red.** Esta capa es la encargada de la entrega y encaminamiento de paquetes entre máquinas situadas en redes distintas. Ejemplo de protocolo: IP, ARP, ICMP.
- **Capa de Enlace.** La capa más baja del modelo se encarga de la transmisión de los datos entre las máquinas, ya sea por un medio físico o no físico. Ejemplo de protocolo: Ethernet, Wi-fi.

La base para todos los protocolos y que dan el nombre al modelo TCP/IP son: el Transfer Control Protocol (TCP) y el Internet Protocol (IP).

La *figura 3.3* es un ejemplo en el que se muestra de forma muy general el proceso de comunicación entre un host emisor y un host receptor haciendo uso del modelo de referencia TCP/IP. Cuando se envía un mensaje o información desde una interfaz o aplicación de usuario (host emisor), cada una de las capas del modelo va agregando información específica a tal mensaje de forma de garantizar que este se envíe por la red de forma correcta al host destino. Esta información que se agrega al mensaje se conoce como encabezado (*header*) de capa. Estos encabezados contienen principalmente la dirección de red, dirección física y el puerto del host destino, así como información de ruteo del paquete y sumas de comprobación (*checksum*) de datos, etc. Cuando se han agregado todos los encabezados necesarios al mensaje, el cual se conoce como *frame*, está listo para enviarse por la red de forma serial por un medio de transmisión, por ejemplo cable RJ45. Una vez llegué el *frame* al host destino, este ahora hará el proceso inverso quitando cada cabecera del paquete de modo de obtener el mensaje tal y como fue enviado por el host emisor.

3. Tecnologías Web

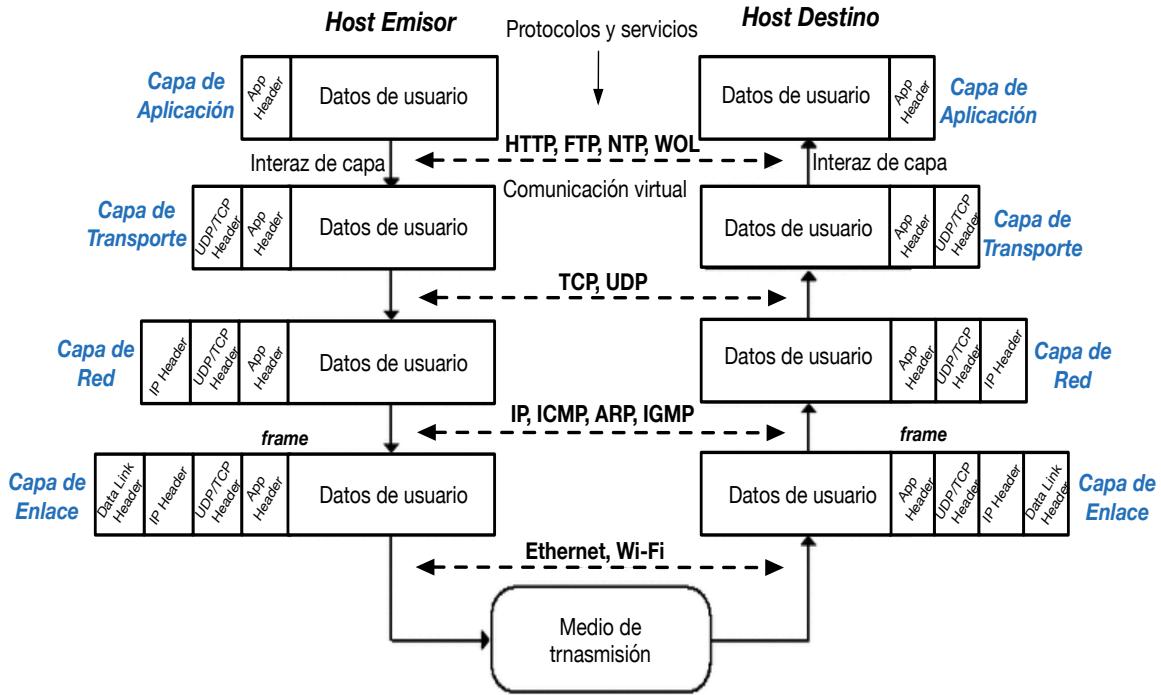


Figura 3.3. Representación del proceso de comunicación en la red en el modelo de referencia TCP/IP.

3.3 Protocolos de Transporte

Se dará una breve descripción del funcionamiento de los protocolos TCP y UDP de la capa de transporte [18].

Protocolo TCP (*Transfer Control Protocol*)

Es un protocolo orientado a conexión, es decir, permite que la comunicación (flujo de bytes) entre dos host en una red se realice generalmente sin errores. Este protocolo es fiable debido a que descompone la información en pequeños paquetes los cuales envía uno a uno hacia el host destino a través de la red; donde el host destino al recibir el paquete, envía una señal de recepción (*Acknowledge*) al host origen o emisor; por otro lado, si el host origen nunca recibe la señal de recepción de algún paquete enviado (después de un intervalo de tiempo específico), el protocolo se encarga de la retransmisión de dicho paquete, garantizando el envío de todos ellos. Por ejemplo

3. Tecnologías Web

este tipo de comunicación se usa generalmente cuando se requiere la transmisión de archivos de texto, ya que es de suma importancia que no se pierdan datos (partes del texto) en el proceso de comunicación. Los protocolos HTTP y FTP de la capa de aplicación son algunos ejemplos que utilizan este tipo de comunicación para garantizar la transferencia de datos.

Protocolo UDP (*User Datagram Protocol*)

Es un protocolo orientado a no conexión y permite el envío de datagramas (paquete de datos o *frames Ethernet*) de forma libre entre dos *host* enlazados a una red, pero no otorga garantía de la entrega de ellos al host destino ni tampoco da informes sobre errores en la transmisión, por lo que no es fiable. Este tipo de comunicación es ampliamente usado para la transmisión de audio y video en tiempo real (*streaming*). Los protocolos DHCP, DNS, WOL de la capa de aplicación del modelo TCP/IP son algunos ejemplos que usan este protocolo de transporte.

3.4 Dirección de Red y Dirección Física

Cada dispositivo enlazado a una red posee tanto una dirección red (también llama dirección IP) y una dirección física (también llamada dirección MAC). A continuación se describe cada una de estas direcciones [15, 17].

Dirección de Red (IP)

Una dirección IP es un número único que identifica a cada nodo en la red y es utilizada por la capa de red del modelo TCP/IP. Esta dirección es un número binario de 32 bits o 4 bytes, lo que equivale a un número en decimal entre 0 y 4294967296. Dado que es una cifra muy grande, para un manejo más fácil de éste, estas direcciones se representan mediante cuatro números enteros de ocho bits o un byte, separados entre ellos por un punto y es conocido en la literatura como dirección IP en formato punto. Por ejemplo, si un host o un nodo en la red contiene la dirección IP: 216.58.210.163, su equivalente en sistema decimal es 3627733667 y en sistema binario es 11011000.0011010.11010010.10100011.

Como vemos en el ejemplo anterior, una representación de la dirección IP en formato punto resulta más fácil de manejar que una representación en sistema binario o decimal.

3. Tecnologías Web

Se dice que una dirección IP es dinámica cuando su numeración cambia cada cierto intervalo de tiempo; el protocolo llamado DHCP (del inglés *Dynamic Host Configuration Protocol*) se encarga de gestionar la asignación de direcciones IP otorgando una única dirección a cada *host* en la red. Por otro lado, se dice que una dirección IP es fija o pública cuando su numeración no cambia con el tiempo y siempre se tiene la misma dirección asignada a un *host*.

Para los humanos resultaría tedioso o complicado aprenderse la dirección IP de servidores (*host*), por lo que se hace uso del sistema de nombres de dominio (del inglés *Domain Name System* o *DNS*). Este sistema asocia o mapea una dirección IP de un servidor a un nombre de dominio. Un ejemplo se muestra en la *figura 3.4*, donde se visualiza la siguiente dirección IP: 216.58.210.163 (usada en el ejemplo de arriba), la cual tiene asociado el nombre de dominio: www.google.es, y con la cual se accede al sitio o buscador Web de *Google*.

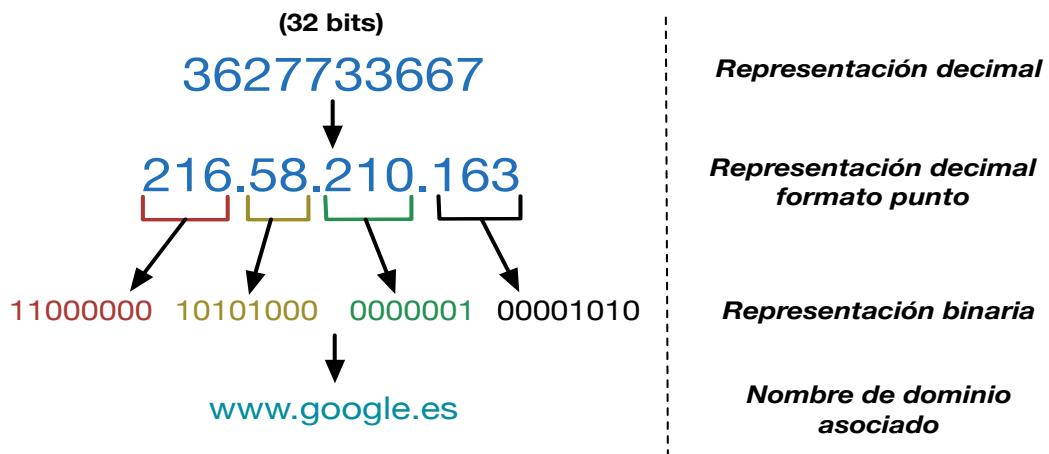


Figura 3.4. Representación de una dirección IP o de red en sus distintos formatos.

3. Tecnologías Web

Dirección Física (MAC)

Una **dirección física**, también conocida como dirección de Hardware o de Control de Acceso al Medio (del inglés *Media Access Control o MAC*), son valores de 48 bits que identifican de manera numérica a cada *host* en la red y es utilizada en la capa de enlace del modelo TCP/IP para acceder a los dispositivos que forman parte de una red. Esta dirección se escribe en un formato de 12 números hexadecimales agrupados en pares y separados por dos puntos (“ : ”), donde cada par forma un byte (se usa este formato para que sea mas legible y de fácil uso, al igual como se mencionó con las direcciones IP).

Este número es asignado y escrito en la tarjeta de red de cada dispositivo en conjunto por el Instituto de Ingenieros Eléctricos Electrónicos (*Institute of Electrical and Electronics Engineers* o IEEE) y el fabricante, por lo que es único y no es posible alterarlo. Los tres bytes más significativos son asignados por la IEEE (a esta parte de la dirección se le denomina identificador único organizacional) y los tres bytes menos significativos por el fabricante. Un ejemplo de dirección física con sus características antes mencionadas se muestra en la *figura 3.5*.

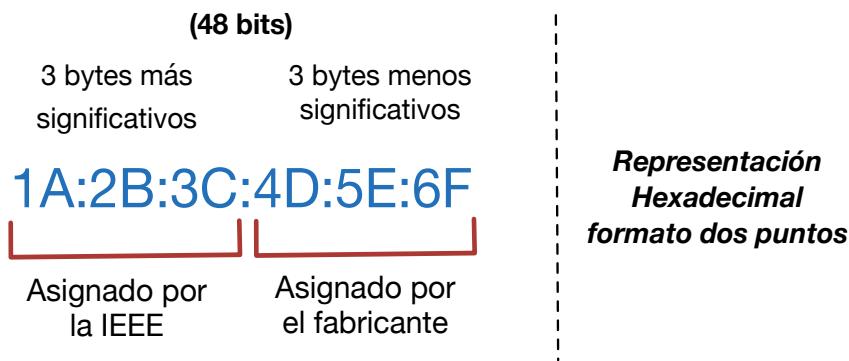


Figura 3.5. Representación de una dirección física o MAC en formato dos puntos y en hexadecimal.

Para el sistema de control desarrollado, las direcciones MAC son usadas, principalmente, para realizar el encendido de las computadoras del aula, mientras que las direcciones IP se utilizan para la identificación y comunicación entre nodos dentro de una red local o LAN.

3. Tecnologías Web

3.5 Arquitectura cliente-servidor

Esta arquitectura es usada para describir un modelo computacional para el desarrollo de sistemas computarizados y está basado en la distribución de funciones entre dos tipos de entidades autónomas e independientes: cliente y servidor; donde el servidor ofrece un servicio y un cliente solicita un servicio al servidor [20, 21].

A continuación se da una explicación de las partes que componen a esta arquitectura.

Cliente

Un cliente se caracteriza por mandar mensajes a un servidor, solicitando un servicio o que realice una tarea en particular. El cliente representa la interfaz de usuario de una aplicación, permitiendo interactuar con servidores. Por otro lado, el cliente es el responsable de validar datos ingresados por el usuario, enviar peticiones a un servidor y mostrar o ejecutar operaciones lógicas a los resultados recibidos. Toda la parte lógica o tareas relacionadas con el cliente se le conoce en la literatura como *Front-end*.

Existen dos modalidades de cliente: cliente ligero y cliente pesado. El primero de ellos es aquel donde la lógica de la aplicación y administración de datos se lleva acabo en el servidor (llamado servidor-pesado) y el cliente solo se encarga de la parte de presentación de datos. Por otro lado, un cliente pesado es aquel en el que el cliente participa en la gestión de datos y es responsable de cierta lógica de la aplicación y solo algunas tareas básicas se llevan a cabo en el servidor (servidor ligero). Para el desarrollo de esta aplicación Web se implementará una distribución de responsabilidad compartida entre ambos, esto es, distribuir la lógica de la aplicación entre el cliente y el servidor (cliente pesado – servidor pesado).

Servidor

Un servidor es aquel el que recibe las peticiones de los clientes. Su objetivo es resolver una tarea a partir de la petición hecha por el cliente y retornar el resultado a éste. Si el servidor de aplicaciones resuelve la petición por sí solo, se dice que es una arquitectura cliente-servidor de dos niveles. Por otra parte las peticiones hechas por el cliente no siempre tienen que ser resueltas por el servidor que las recibe, éste puede estar conectado o tener comunicación con otros tipos

3. Tecnologías Web

de servidores, por lo que puede tomar el rol de cliente y realizar peticiones a estos, delegándoles partes de las tareas o la tarea completa. Tales servidores pueden ser de bases de datos, de archivos, de impresión, de correo, webs o hasta sistemas embebidos basado en microcontroladores. A esta modalidad se le conoce como arquitectura cliente-servidor de tres niveles o más, dependiendo de cuántos servidores estén envueltos en la resolución de la petición del cliente. No importa cuántos servidores estén envueltos en el proceso, al final, la petición hecha por el cliente (usuario) será resuelta.

En la *figura 3.6* se representa mediante un diagrama la arquitectura cliente-servidor antes mencionada.

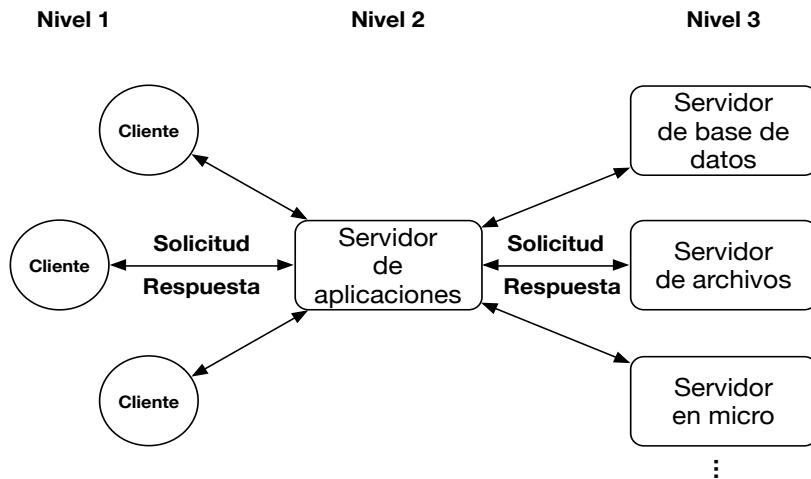


Figura 3.6. Representación de una arquitectura cliente-servidor de tres niveles.

El sistema de control desarrollado en esta tesis trabaja bajo una arquitectura cliente-servidor de 3 niveles. En el nivel 1 se encuentran los clientes (profesores) encargados del envío de peticiones al servidor. En el nivel 2 se encuentra el servidor de aplicaciones encargado de atender las peticiones de los clientes, y que a su vez puede tomar rol de cliente para enviar peticiones a otros servidores que se encuentran en el nivel 3, con el fin de delegar tareas a estos y poder resolver la solicitud inicial del cliente.

Para fines prácticos y de diferenciación, el servidor de aplicaciones que se encuentra en el nivel 2 tendrá el nombre Servidor Maestro, mientras que los que se encuentran en el nivel 3 tendrán el nombre de Servidores Esclavos.

3. Tecnologías Web

3.6 Protocolos de Aplicación

Existen una gran variedad de protocolos utilizados por la capa de aplicación del modelo TCP/IP y que son ampliamente usados para comunicarse con servidores en la red con el fin de solicitar un recurso a este, por ejemplo un archivo, una página web, una aplicación, correos, etc. En este apartado se mencionará el funcionamiento y características de los protocolos de aplicación utilizados en este trabajo de tesis: HTTP y WoL.

Protocolo HTTP

El protocolo de transferencia de hipertexto (del inglés **HyperText Transfer Protocol**) es un protocolo de comunicaciones utilizado para el intercambio de información a través de la Web. Este protocolo trabaja bajo una arquitectura cliente-servidor como la antes mencionada [16, 22]. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud, el servidor responde con un mensaje similar, el cual contiene el estado de la operación (*figura 3.7*).

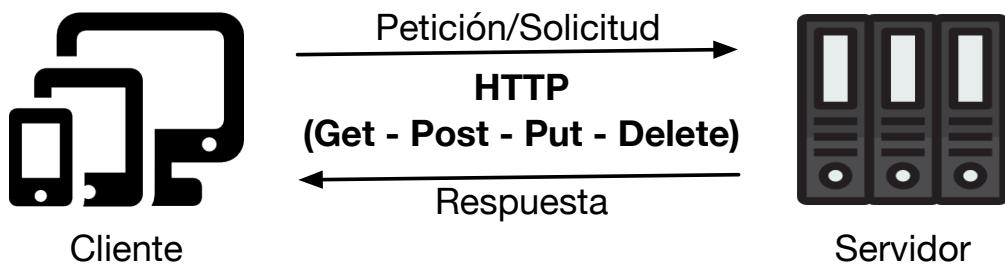


Figura 3.7. Protocolo de aplicación HTTP y sus principales métodos: Get, Put, Post, Delete.

Los mensajes de solicitud de un recurso por parte del cliente son interceptados o localizados en el servidor por un localizador uniforme de recursos o URL (*Uniform Resource Identifier*). En la *figura 3.8* se muestra un ejemplo de una URL HTTP con sus componentes principales. Para este ejemplo se ingresó al navegador *Google Chrome* y se realizó una petición HTTPS a un servidor de *Google* para la búsqueda de documentos o sitios Web relacionados con la palabra “URL”.

3. Tecnologías Web



Figura 3.8. Composición de un Localizador uniforme de recursos (URL)

Haciendo un análisis de los componentes (de izq. a der.) de la URL de la *figura 3.8*, tenemos:

- **Identificación del protocolo.** Es el protocolo de aplicación utilizado para envío de peticiones al servidor. En este caso *Google* utiliza HTTPS para búsquedas en la Web (que es el protocolo HTTP con comunicación segura), pero también las URL son utilizadas por protocolos como FTP, FILE, HTTP, entre otros. Los caracteres “`://`” son necesarios para el uso de ciertos protocolos, en el caso de los protocolos HTTP o HTTPS sí son requeridos.
- **Nombre (Domino) del servidor.** Este componente incluye el nombre del servidor (*host*) al que va dirigida la petición HTTP. Los dos puntos “`:`” que vienen enseguida, son requeridos para especificar el número de puerto.
- **Puerto.** Es el punto final de comunicación en el que el servidor acepta conexiones con los clientes. En este caso, el servidor acepta conexiones de los clientes en el puerto 443 (coloquialmente se dice que el servidor escucha peticiones de los clientes por el puerto 443). En algunos casos no es necesario especificar el número de puerto en el que el servidor acepta conexiones (en este caso se incluyó el puerto en la URL solo para fines demostrativos), dado que hay puertos llamados como “bien conocidos” para protocolos estandarizados como HTTPS, que por defecto utiliza el puerto 433; o por ejemplo, el protocolo HTTP tiene asignado el puerto 80. La IANA (*Internet Assigned Numbers Authority*) es la autoridad responsable de asignar y mantener números de puertos para uso específico [23].
- **Ruta.** Especifica una ruta de acceso dentro del servidor, la cual dirige a un código a ejecutar de modo de resolver la petición hecha por el cliente y retornar una respuesta a este.

3. Tecnologías Web

- **Parámetros de consulta.** Este parámetro corresponde a una cadena de consulta y contiene datos para enviar al servidor. Esta cadena de consulta es precedida por el carácter “?”. El formato de esta cadena está dado por un conjunto de pares *nombre:valor*. En este ejemplo, los parámetros de consulta enviados al servidor de *Google* son “q=url”, los cuales le indican que el cliente requiere información acerca de la palabra “url”; posteriormente el motor de búsqueda de Google hace una recopilación de todos los documentos/sitios Web disponibles en Internet relacionados con esta palabra y eventualmente retorna al cliente el resultado de la búsqueda (enlaces o hipervínculos a estos sitios/documentos Web).

Protocolo Wake On LAN (*WoL*)

Es una tecnología que permite el encendido de computadoras enlazadas a una red de área local (*LAN*). El Controlador de Interfaz Ethernet (Network Interface Controller - NIC) de la computadora que se requiere encender está a la espera (o “escuchando”) de un patrón específico de datos, que una vez detecta el patrón correcto, es capaz de comenzar el encendido del sistema. A este patrón de datos se le conoce como paquete mágico, el cual es un *frame Ethernet* que contiene información específica para cada computadora en particular y está conformado por 16 repeticiones de la dirección física (*MAC*) de la computadora destino más un conjunto de seis bytes de valor 255 en decimal (FF en hexadecimal) necesarios para sincronización y permitir una detección más simple del paquete o patrón de datos [24, 25, 26].

Como modo de ejemplo supongamos que en una habitación existen tres computadoras de escritorio y se desea encender remotamente la computadora cuya dirección física es: 11:22:33:44:55:66.

El contenido del paquete mágico para encender dicha computadora se muestra en la *figura 3.9*.



Figura 3.9. Estructura de un paquete mágico. Se muestran los seis bytes FFh seguido de 16 repeticiones de la dirección física.

3. Tecnologías Web

Al enviar este paquete por la red local, es necesario realizar una transmisión de difusión por *broadcasting*; esto debido a que la computadora que se requiere que procese el paquete mágico se encuentra apagada, por lo que no cuenta con una dirección IP, y dado a que esta dirección es necesaria para identificación de host o nodos en redes, no es posible establecer una comunicación con ella.

El protocolo de transporte utilizado para implementar esta tecnología es UDP, debido a que es orientado a no conexión y no se necesita establecer una comunicación entre emisor y receptor, por lo que permite el envío de datagramas o paquetes de forma libre a todas las computadoras (*host*) enlazadas a una red local.

En la *figura 3.10* se muestra un diagrama en el que se representa el ejemplo mencionado anteriormente. Mediante una aplicación móvil (o en cualquier otro dispositivo) que integre la tecnología *WOL*, se envían paquetes mágicos por la red de área local con el fin de encender la computadora requerida (PC2 con dirección física: 11:22:33:44:55:66).

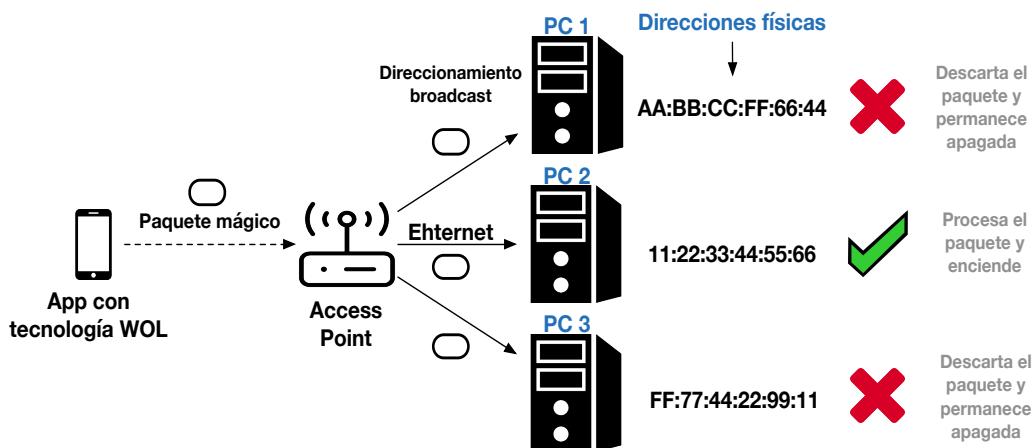


Figura 3.10. Diagrama en el que se representa el funcionamiento de la tecnología WOL utilizada para el encendido de computadoras en redes.

Debido a que el sistema desarrollado en esta tesis adopta una arquitectura cliente-servidor, en las siguientes secciones se mencionan las tecnologías Web empleadas en ambas partes, comenzando por las tecnologías del lado del cliente (*Front-end*) y posteriormente las del lado del servidor (*Back-end*).

3. Tecnologías Web

3.7 Tecnologías Web del lado del cliente

En este apartado se mencionan las tecnologías y los lenguajes de programación utilizados para el desarrollo de la interfaz Web.

Lenguaje de Marcado de Hipertexto (HTML)

Hypertext Markup Language es un tipo especial de lenguaje de computadora que se conoce como lenguaje marcado y está diseñado para especificar tanto el contenido como la estructura de las páginas o aplicaciones Web. La versión actual *HTML5* contiene nuevos elementos que se pueden utilizar en aplicaciones Web que hacen uso de Internet, así como nuevas *API* que pueden ser usadas con JavaScript e implementarlas nativamente en la aplicación. Este lenguaje es ampliamente utilizado hoy en día para el desarrollo de aplicaciones Web multiplataforma [30, 31].

Hojas de Estilo en Cascada (CSS)

Cascade Style Sheets es un lenguaje de hojas de estilo usado para describir la presentación de un documento (fuentes, tamaño, colores, posicionamiento de los objetos en el documento) escrito en un lenguaje marcado (*HTML* o *XML*). Esta tecnología permite agregar un estilo a las aplicaciones *Web* portables en forma independiente a su contenido o estructura. En este proyecto de tesis se hará uso de este lenguaje en conjunto con un marco de trabajo llamado JQuery Mobile, el cual se describe en breve [32].

JavaScript

Es un lenguaje interpretado de alto nivel orientado a objetos; además es de lenguaje pequeño y de peso ligero. Es usado ampliamente para crear el código del lado del cliente en las aplicaciones Web ya que principalmente les otorga un enfoque dinámico; esto es, permite crear aplicaciones o páginas que pueden cambiar su contenido (en un instante) en respuesta a ciertos eventos, por ejemplo hacer clic en un botón, al pasar el cursor por un objeto/elemento de la página, al terminar la ejecución de cierta función, etc [33, 34].

3. Tecnologías Web

JQuery

Es la librería más usada y popular de JavaScript creada por Jhon Resig en el 2006 con el propósito de simplificar el código HTML del lado del cliente. Esta librería facilita la manipulación de los objetos o elementos de una página Web y permite interactuar con los servidores de una forma portable a través de varios navegadores Web [35, 36].

JQuery Mobile

Es un *framework* (marco de trabajo) para la creación de aplicaciones Web móviles. Bajo el enfoque “Write less, do more” (“escribe menos, haz más”), este *framework* permite el uso de HTML5 y CSS3 para el diseño de aplicaciones Web dándoles un estilo atractivo sin necesidad de escribir muchas líneas código. Además, funciona en los más populares *smartphones*, tabletas y plataformas de escritorio; por lo que es independiente de la marca del dispositivo o sistema operativo [37, 38].

Por otra parte, haciendo uso de la librería de *JQuery* es posible realizar de una manera muy sencilla y reducida las siguientes acciones que son ampliamente usadas en el entorno Web, que con el uso de JavaScript puro es un poco más difícil y tedioso de hacer e implican más líneas de código.

- **Manipulación de HTML/DOM.** Es posible acceder y manipular los elementos de una página *Web* de una forma rápida y sencilla.
- **Manipulación de CSS.** Es posible modificar la clase o propiedades de los elementos de una página *Web*, por ejemplo su color, tamaño, composición, etc.
- **Transferencia de datos con AJAX.** *Asynchronous JavaScript and XML* es una tecnología que permite el intercambio de datos o información con un servidor (generalmente haciendo uso del protocolo HTTP o HTTPS) y actualizar partes de una página *Web* sin necesidad de refrescar la página completa. Esta tecnología permite crear aplicaciones Web rápidas y dinámicas y es utilizada en aplicaciones como Google, Gmail, Facebook y Youtube [39].

3. Tecnologías Web

- **Manejador de eventos.** Es posible detectar y procesar ciertas acciones hechas por el usuario en una aplicación Web; como hacer clic en un botón o en una liga, pasar el cursor por ciertas áreas de la interfaz, cambio de tamaño de la pantalla del dispositivo del usuario, entre otros.

Java Script Object Notation (JSON)

Es un formato de peso ligero para el intercambio de datos. JSON utiliza sintaxis de JavaScript, pero su formato es de solo texto, como lo es XML (Extensible Markup Language). Además, es un formato de texto compatible con cualquier lenguaje de programación y tiene la ventaja de ser de fácil lectura y escritura para los humanos y de fácil procesamiento para las máquinas. El formato JSON es idéntico al código para crear los objetos en lenguaje JavaScript, por lo que es sencillo de analizar y de convertirlo a objetos nativos de JavaScript mediante el uso de funciones estandarizadas. El formato de un objeto JSON está dado por un conjunto de pares *Nombre-Valor*, donde cada par se separa con comas y cada objeto va entre corchetes [40, 41]. En la *figura 3.11* se muestra un ejemplo de la estructura de un JSON.

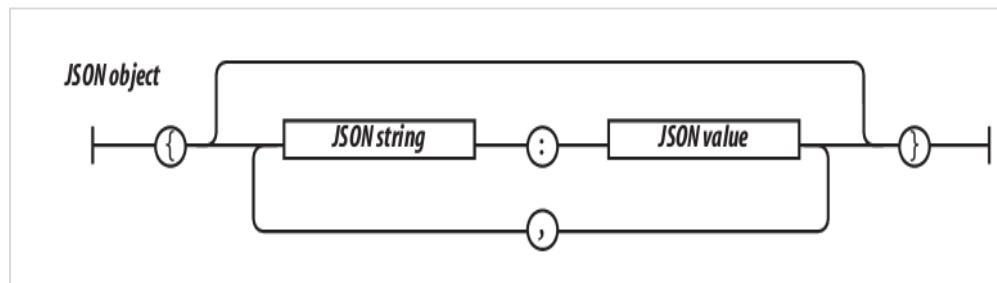


Figura 3.11. Representación de un objeto JSON. El cual se compone de pares *Nombre-Valor*, donde cada par se separa con comas y cada objeto va entre corchetes.

3.8 Tecnologías Web del lado del servidor

La programación del servidor maestro y la mayoría de los servidores esclavos se realiza mediante la plataforma de Node.js en conjunto con un marco de trabajo llamado *Express*. Uno de los servidores esclavos dentro del sistema de control corresponde a un sistema embebido, cuya programación se realiza en lenguaje C y se verá en la siguiente sección del capítulo.

A continuación se mencionan las características principales de cada una de estas tecnologías.

3. Tecnologías Web

Node js

Es un entorno de ejecución para JavaScript, fue desarrollado por Ryan Dahl en el año 2009 y está construido con el motor de JavaScript V8 de Google Chrome. Node.js es de código abierto, multiplataforma (operable en diversos sistemas operativos tales como OS X, Windows y Linux) y es usado para el desarrollo de aplicaciones del lado del servidor [42, 43, 46].

Algunas de las características más importantes de esta plataforma son:

- Utiliza un modelo orientado a eventos, esto es, el flujo del programa es manejado por eventos específicos, como abrir y cerrar conexiones con clientes, terminar de ejecutar una función, entre otros.
- A diferencia de otras plataformas de programación de servidores como lo es Apache, la cual crea un hilo o proceso de comunicación para cada cliente, Node.js crea solo un hilo de ejecución para atender a todas las peticiones de los clientes, reduciendo así el consumo de memoria RAM y procesamiento en el servidor y por ende permite más conexiones de clientes en un solo servidor.
- Utiliza una programación asíncrona, esto es, los eventos son ejecutados de forma independiente al flujo del programa, por lo que en lugar de esperar a que un evento ocurra, el programa pasa un evento a una cola de manejador de eventos continuando con la ejecución del programa principal así como de operaciones de E/S continuamente y sin bloqueos; que una vez que el evento está listo, el programa retorna a éste por medio de las llamadas funciones *callback*. Esta función pasa como un argumento en un evento e indican si este se ejecutó adecuadamente, si hubo algún tipo de error, entre otros.

Por otro lado, Node.js provee una amplia librería de varios módulos de JavaScript que simplifican el desarrollo de aplicaciones Web. Estos módulos son anexados fácilmente a un entorno de trabajo por medio de un ecosistema de paquetería para JavaScript llamado *npm* (del inglés *Node Package Manager*), la cual contiene la más grande colección de librerías de código abierto en el mundo y es soportada y usada por una comunidad muy grande de desarrolladores.

3. Tecnologías Web

Express

Node.js provee una API (*Application Programming Interface*) que permite acceder a su núcleo haciendo uso de ciertos módulos preinstalados en un proyecto, por ejemplo, un módulo ampliamente utilizado es el de HTTP, el cual permite crear un servidor Web que reciba peticiones HTTP hechas por clientes en un puerto específico del servidor. Sin embargo, una desventaja de usar esta API de Node.js es que es de muy bajo nivel y se tienen que escribir y rescribir muchas funciones en una aplicación Web. Para dar solución a esto y hacer que el desarrollo Web con Node.js sea más fácil, nace el marco de trabajo *Express* creado y liberado en 2009, siendo actualmente el *framework* más popular y usado para la programación de aplicaciones Web con Node.js.

Express proporciona al desarrollador un acceso completo al núcleo de la API de Node.js, por lo que todo lo que se pueda hacer con Node.js también se puede hacer con *Express* de una forma más sencilla [44, 45].

Por otro lado, *Express* es considerado un módulo de Node.js como cualquier otro, por lo que puede ser instalado haciendo uso de la paquetería *npm*. Una vez instalado, para hacer uso de él se introduce la siguiente línea de código en el fichero principal del proyecto (y en general para introducir cualquier otro módulo se usa la función *require()*):

```
var express = require('express');
```

Express consta solo de tres componentes principales para la programación de aplicaciones Web, los cuales se describen a continuación:

- **El objeto aplicación.**

Este objeto es una instancia de *Express* y es representada convencionalmente con la variable de nombre *app*. Este contendrá todas las funcionalidad de Express; además se proveen una gran cantidad de métodos y propiedades disponibles para operar sobre este.

Para crear una instancia de *Express* se introduce la siguiente línea de código:

```
var app = express();
```

3. Tecnologías Web

- **El objeto petición.**

Este objeto es creado en automático cuando un cliente realiza una petición HTTP a la app de *Express*. Este objeto es convencionalmente representado por la variable de nombre *req* y están disponibles una gran cantidad de métodos y propiedades para operar sobre este. Un método ampliamente usado sobre este objeto es *query* el cual permite extraer datos o parámetros de una URL enviada por un cliente mediante una petición HTTP.

- **El objeto respuesta.**

Este objeto es creado en automático con el objeto petición y su nombre convencional está representado por la variable de nombre *res*. Al igual que el objeto *req*, hay muchos métodos y propiedades disponibles para operar sobre el objeto *res*. Unos de los más usados son el método *sendfile* y *send*, los cuales envían datos o archivos al cliente (página web, json, xml, etc) almacenados en el servidor.

Por otra parte, para poner en marcha un servidor HTTP usando Node se hace uso del módulo llamado *http*. El siguiente código en JavaScript es un ejemplo en el que se ejecuta un servidor en el puerto 3000 haciendo uso de los módulos *http* y *Express* [45].

```
1 // Incluir el módulo de HTTP
2 var http = require('http');
3 // Incluir el módulo de Express
4 var express = require('express');
5 // Crear una instancia de Express
6 var app = express();
7 // Empezar la aplicación. Pasamos el objeto app a la API de http
8 http.createServer(app).listen(3000, function() {
9   console.log("App Express en el Puerto 3000");
10 });
11 // Ruta para retornar la página de inicio al cliente
12 app.get('/', function(req, res) {
13   res.send("Hola Mundo!");
14});
```

Después de correr este código, para ver su funcionamiento e interactuar con esta aplicación Web, desde un navegador se ingresa la siguiente dirección: <http://localhost:3000/>, mostrando la página de inicio de la app, que en este caso solo muestra el mensaje “Hola Mundo! ” en el navegador.

3. Tecnologías Web

3.9 Sistemas embebidos

Un sistema embebido (SE) se puede definir como un sistema que realiza una tarea en específico y que tiene una microcomputadora embebida dentro; contrario a lo que ocurre con los ordenadores de propósito general (como las PC) los cuales están diseñados para cubrir un amplio rango de necesidades. Un SE se compone principalmente de un microcontrolador y un software que se ejecuta sobre este. En el microcontrolador se incorporan principalmente el procesador, memoria RAM, memoria ROM y puertos o interfaces de entrada/salida como los seriales, analógicos, control de movimiento, de propósito general, etc. La gran mayoría de estos sistemas no cuentan con un teclado, pantalla gráfica, almacenamiento secundario, ni tampoco con un sistema operativo, por lo que todo el software debe ser desarrollado [47, 48].

Actualmente los SE a pesar de no ser muy nombrados, brindan soluciones a cualquier proceso industrial, comercial, escolar, etc. Son un componente ubicuo en nuestra vida diaria encontrándolos dentro de nuestro hogar, carro, juguetes, celulares, hornos de microondas, refrigeradores; en realidad, es difícil encontrar algún dispositivo cuyo funcionamiento no esté basado en algún SE.

Por otra parte, aquellos SE que cuenten con un microcontrolador en comunicación con un controlador de red, es un tema en crecimiento dentro de la industria y las aplicaciones de consumo general, ya que permiten incrementar el alcance de comunicación e interacción con otros dispositivos dentro de una red. Algunas de las aplicaciones en donde se implementa este tipo de sistemas son para el control de hardware desde una página Web, monitoreo remoto de sensores, control de acceso, sistemas con requerimiento de ejecución en tiempo real, entre otras [15].

Uno de los llamados servidores esclavos dentro del sistema de control desarrollado en esta tesis es un SE de *Texas Instruments* (TI) el cual contiene un microcontrolador TIVA TM4C1294NCPDT con procesador ARM CortexTM-M. Este representa una nueva clase de microcontroladores mucho más poderoso que lo que había hace diez años y de los mejores hoy en día en el mercado de microcontroladores debido a su bajo costo y consumo de energía. Además, este SE proporciona varios periféricos de conectividad punto a punto como UART, I²C, USB y provee periféricos de comunicación para redes, integrando un puerto Ethernet junto con una interfaz PHY y una pila TCP/IP de peso ligero (*lwIP*), lo que permite su programación como un servidor HTTP, cliente o ambas [49, 56].

3. Tecnologías Web

A continuación se muestran las características más importantes que presenta este microcontrolador en cuanto a su rendimiento y las interfaces de comunicación. En la figura 3.12 se muestra una imagen de este sistema embebido [50].

De rendimiento:

- Procesador core ARM® Cortex™ – M4F (bus de datos de 32 bits)
- Hasta 120-MHz (150 DMIPS)
- Oscilador PLL
- 1024 KB de memoria Flash
- 256 KB de SRAM
- 6 KB de memoria EEPROM
- Direct Access Memory (DMA)
- In-Circuit Debug Interface (ICDI) para programación y depuración

Interfaces de comunicación:

- Ocho manejadores UART
- Diez módulos I²C
- Dos módulos CAN 2.0
- Ocho salidas PWM
- Dos temporizadores *Watchdog*
- Tres comparadores analógicos
- Un Ethernet MAC 10/100 con PHY
- USB 2.0 con conector A/B
- Más de 80 Pines de propósito general
- 106 ISR manejadas por NVIC

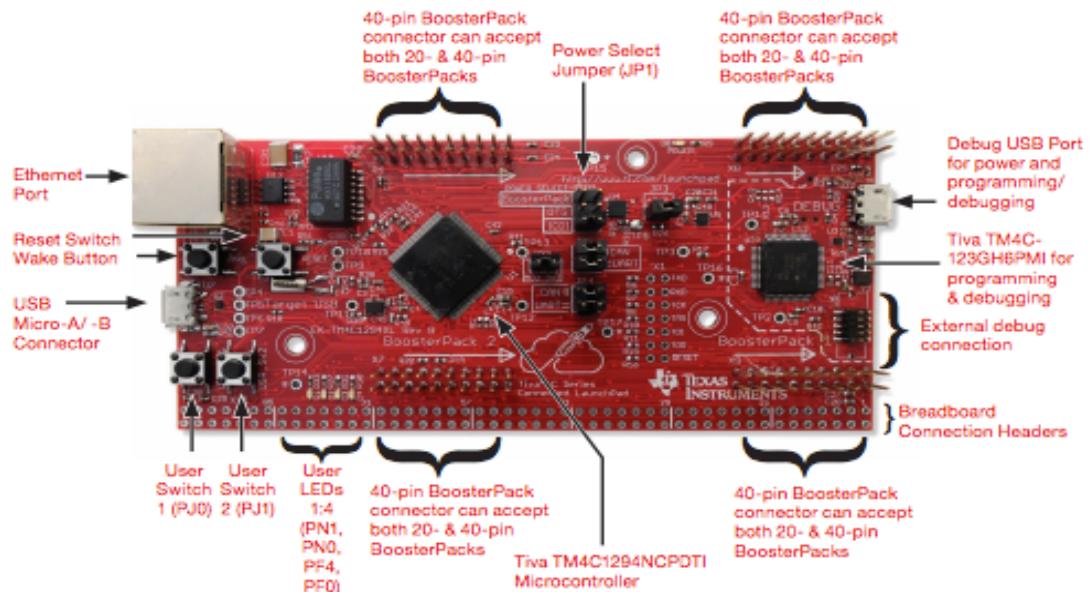


Figura 3.12. Sistema embebido con microcontrolador TIVA TM4C1294NCPDT de Texas Instruments y algunas de sus interfaces de comunicación [51].

3. Tecnologías Web

Para el caso que nos ocupa, el SE cumplirá la tarea de control de proyectores dentro del Aula del Futuro. Actualmente la gran mayoría de los proyectores en el mercado contiene un puerto serial macho DB9 para su control desde una PC o algún otro dispositivo que integre periféricos de comunicación serial como los microcontroladores.

En el manual técnico de cada proyector el fabricante otorga comandos en hexadecimal y especificaciones para su control; la disponibilidad de comandos varía de la marca y modelo del proyector, pero la gran mayoría soporta los comandos para el encendido, apagado, control de puertos de entrada/salida (como el de audio, video, USB, etc.) y ejecutar ciertas funciones como el enfoque de imagen, ingreso al menú de configuraciones, estado de la lámpara, entre otros.

El SE Tiva TM4C integra hasta ocho manejadores UART (Universal Asynchronous Receiver/Transmitter) para la comunicación serial, permitiendo controlar hasta ocho proyectores en conjunto con el protocolo RS232. Además, al conectar al SE a una red local es posible que otro servidor dentro de la red, como el Servidor Maestro, se comunique con este dando la orden o delegando la tarea de control de proyectores. En el capítulo 6 “*Desarrollo de la solución*” se muestra detalladamente el proceso de control de proyectores desde la aplicación Web.

Ahora se pasará a mencionar las herramientas que van a permitir hacer uso del SE como controlador de proyectores. Se comenzará por presentar el entorno de desarrollo utilizado para la programación del microcontrolador; seguido del uso del SE como servidor HTTP, lo cual permitirá establecer una comunicación: *Servidor Maestro – SE*; y por último, el protocolo serial RS232 y el módulo UART usados para establecer una comunicación: *SE – Proyector*.

- **Code Composer Studio (CCS)**

Es un IDE (Integrated Development Environment) ampliamente utilizado en la industria debido a que soporta una amplia gamma de procesadores para su programación, contiene varias características del *framework Eclipse* y otorga una gran cantidad de herramientas para la depuración de programas para aplicaciones embebidas. Además contiene un compilador C/C++, un editor de código fuente, acceso a registros de memoria, etc. [52].

Este IDE será el utilizado en este proyecto de tesis y se encuentra disponible para cualquier plataforma (Linux, Windows, OS X) de forma gratuita para fines académicos; para fines industriales o comerciales es necesario adquirir una licencia.

3. Tecnologías Web

Otro entorno de desarrollo muy utilizado para la programación de sistemas embebidos de Arduino o *Texas Instruments* es *Energia*. Esta plataforma es de código abierto, portable, multiplataforma y contiene una API que permite la programación de SE de forma muy sencilla. Pero por otro lado, presenta muchas limitaciones en cuanto a la disponibilidad de ciertas herramientas de desarrollo, como acceso a registros y depuración profunda. [53].

- **Sistema Embebido como servidor HTTP**

Con el fin de facilitar la programación del SE como servidor HTTP, se encuentran disponibles un conjunto de librerías y API's, de las cuales, la más usada por desarrolladores y que además provee una gran variedad de funciones de acceso a periféricos, es la llamada *TivaWareTM Peripheral Driver Library* y [54, 56].

A continuación se mencionan las configuraciones principales (a nivel código) para inicializar al servidor. Se toma como base uno de los ejemplos disponibles para el SE, llamado *enet_io.c* [55]. Las configuraciones a relazar son:

1. Inicializar reloj del sistema y puerto Ethernet.
2. Inicializar la pila de peso ligero lw-TCP/IP.
3. Inicializar el servidor HTTP.

A continuación se explican cada uno de estos pasos.

1. Inicializar reloj del sistema y puerto Ethernet

De acuerdo al manual técnico (hoja de datos) del SE, para hacer uso de su puerto Ethernet es necesario que se utilice una frecuencia de oscilación de 120 MHz, por lo que el oscilador principal del sistema se inicializa a esta frecuencia mediante las funciones *SysCtlMOSCConfigSet()* y *MAP_SysCtlClockFreqSet()*. Seguido se inicializa el puerto *Ethernet* mediante la función llamada *PinoutSet()*. Esta recibe dos valores booleanos como parámetros, el primer de ellos es para inicializar el puerto Ethernet y el segundo para inicializar el puerto USB.

```
1 // Inicializar oscilador a 120 MHz requerido por la interfaz PHY
2 SysCtlMOSCConfigSet(SYSCTL_MOSC_HIGHFREQ);
3
4 g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
```

3. Tecnologías Web

```
5           SYSCTL_OSC_MAIN |  
6           SYSCTL_USE_PLL |  
7           SYSCTL_CFG_VCO_480), 120000000);  
8  
9   PinoutSet(true, false); // Inicializar puerto Ethernet
```

2. Inicializar la pila de peso ligero lwIP

Para inicializar la pila TCP/IP se hace uso de la función *lwIPInit()*, la cual recibe como parámetros (de izq. a der.) la frecuencia de reloj del sistema (*g_ui32SysClock*), la dirección física o MAC de la tarjeta de red (*pui8MACArray*) y los siguientes campos son para la configuración de red, que este caso se indica que se requiere el uso de una dirección IP estática de valor *192.168.11.166*, una máscara de subred y una puerta de enlace de valor *255.255.255.0*

```
1 lwIPInit(g_ui32SysClock, pui8MACArray,  
2           (192u << 24) | (168u << 16) | (11 << 8) | 166, //Dir. IP  
3           (255u << 24) | (255u << 16) | (255 << 8) | 0, //Máscara subred  
4           (255u << 24) | (255u << 16) | (255 << 8) | 0, //Gateway  
5           IPADDR_USE_STATIC);
```

3. Inicializar el servidor HTTP

Con la función *httpd_init()* se inicializa el servidor HTTP. El puerto donde este acepta conexiones de los clientes es en el 80. Todos los recursos o rutas (URL) son interceptadas y localizadas en el fichero de nombre *io_fs.c* que se encuentra dentro del proyecto.

```
1 httpd_init(); //Inilicializar servidor HTTP
```

- **Módulo UART y el Protocolo RS232**

La comunicación serial implica el envío de un bit a la vez de forma secuencial desde un emisor hacia un receptor, de tal manera que un conjunto de datos sean enviados al cabo de un tiempo determinado. El módulo UART (Universal Asynchronous Receive/Transmit) es un componente de hardware que soporta este tipo de comunicación y se compone principalmente de

3. Tecnologías Web

un registro de datos, un registro de corrimiento, registros para el control de interrupciones, *buffers* tipo FIFO (First In First Out) y un generador de *baud rate* [48, 57, 59].

Cuando una comunicación serial se realiza de forma asincrónica, como en el caso del módulo UART, la transferencia de datos entre dos entidades (emisor y receptor) que se comunican se puede comenzar o detener en cualquier momento; esto debido a que ambas contienen su propia señal de reloj independiente pero operando a la misma frecuencia.

En la *figura 3.13* se muestra lo que se conoce como un *frame* o paquete de datos, que es la unidad completa más pequeña en una transmisión serial.

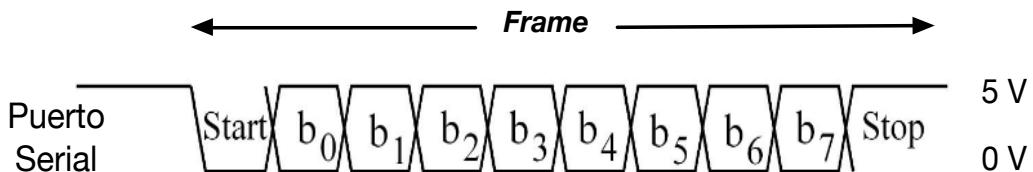


Figura 3.13. Composición de un *Frame* serial el cual se compone de un bit de *start*, 8 bits de datos y 1 bit de *stop*. Donde b₀ es el bit menos significativo y b₇ el más significativo.

Cada *frame* se compone por un bit de comienzo (*start*) que se representa con un 0 lógico (0 volts), 8 bits de datos y un bit de parada (*stop*) que se representa con un 1 lógico (5 Volts). La transmisión de un *frame* comienza cuando el receptor detecta el bit de *start* y termina con la detección del bit de *stop*. En cada UART del SE Tiva TM4C se puede configurar la composición de los *frames* pudiendo seleccionar entre 5 a 8 bits de datos, 1 o 2 bits de parada o agregar un bit de paridad.

Este tipo de comunicación serial se utiliza ampliamente para transmitir datos en código ASCII (American Standard Code for Information Interchange); este código se encarga de representar con números ciertos caracteres; por ejemplo el carácter ‘a’ lo representa o codifica con el número 97 decimal o 61 en hexadecimal.

Se conoce como *baudrate* (velocidad o tasa de transferencia) como el número total de bits enviados por segundo. Por ejemplo, si una transmisión serial se realiza a una tasa de 19200 bps, entonces el tiempo por cada bit es de $1/19200 = 0.5 \mu\text{s}$; y el tiempo por cada *frame* es de $10/19200 = 520 \mu\text{s}$.

3. Tecnologías Web

Por otro lado, el protocolo RS232 (Recommended Standard 232) también conocido como EIA232, es una interfaz que designa un conjunto de normas para el intercambio de información entre una DTE (Data Terminal Equipment) como la terminal o computadora y un DCE (Data Communication Equipment) como un modem o una impresora. Este protocolo serial fue introducido en el año de 1960 por la EIA (Electronic Industries Association) siendo uno de los más antiguos pero que se utiliza ampliamente hoy en día debido a su fácil implementación y bajo costo. Las especificaciones eléctricas en este protocolo indican que trabaja con lógica negativa, indicando que un 0 lógico (también llamado “espacio”) se representa con un voltaje de entre +5 y +15 volts, y un 1 lógico (también llamado “marca”) lo representa con un voltaje entre -5 y -15, y donde la región entre -5 y +5 se considera indefinida.

Otra especificación indica que el cable que conecta un DTE y un DCE no debe exceder los 15 metros de longitud. Cuando se requiera una comunicar un DTE con otro DTE, se debe utilizar un cable RS232 de tipo *Null Modem* (no conexión con modem) [58].

En cuanto al tipo de conectores, el cable RS232 soporta principalmente dos tipos, uno de 25 pines llamado DB25 y otro de 9 pines llamado DB9. El más usado de estos dos es el conector DB9 ya que es más barato y presenta un uso más extendido para ciertos periféricos, como el ratón serie de la PC, proyectores, entre otros. En la figura 3.14 se muestra el conector macho DB9 con el nombre cada uno de sus pines.

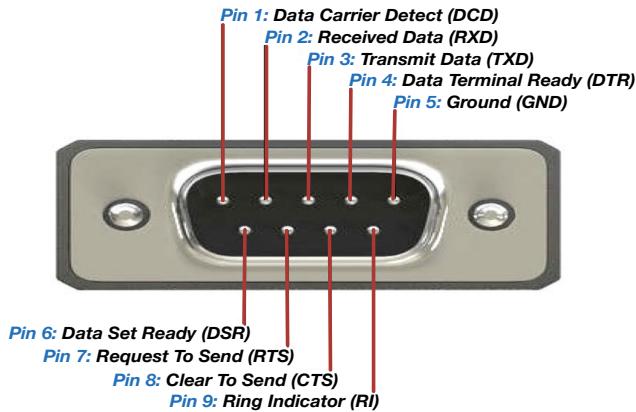


Figura 3.14. Conector macho DB9 con cada uno de sus pines.

El conector DB9 hembra contiene la misma configuración de pines que el conector macho, excepto que el pin 3 lo usa para recibir datos y el pin 2 para transmitir.

3. Tecnologías Web

Para establecer una comunicación en RS232 solo son necesarios tres líneas (pines): el transmisor de datos (TXD), el receptor de datos (RXD) y tierra (GND). Las demás líneas, generalmente, se utilizan para identificar problemas en la comunicación y garantizar el envío de los datos. Por otro parte, la ventaja de usar líneas independientes para recibir y transmitir bits de datos, es que es posible realizar una comunicación *Full Duplex*, esto es, se pueden transmitir datos al mismo tiempo que se reciben otros.

Otro punto importante a considerar al usar el protocolo RS232 en conjunto con los UART del SE, es que este último trabaja con señales lógicas TTL (Transistor-Transistor Logic) y solo opera con voltajes de entre 0 a 5 Volts, mientras que el protocolo RS232 opera con señales de entre -15 a +15 Volts, por lo que es necesario el uso de algún *driver* que permita acondicionar estas señales con el fin de no dañar la circuitería interna del SE. Uno muy utilizado es el circuito integrado (CI) MAX233, el cual consta de 20 pines y se encarga de transformar señales TTL a señales RS232 y viceversa. Además tiene la ventaja sobre otros *drivers* (como el MAX232) de que no es necesario el uso de capacitores externos para su operación [61].

La figura 3.15 es un diagrama en donde se muestra la configuración a implementar cuando se requiere una comunicación serial bajo la norma RS232 entre un microcontrolador que integre un módulo UART y un dispositivo electrónico como proyectores, impresoras, etc.

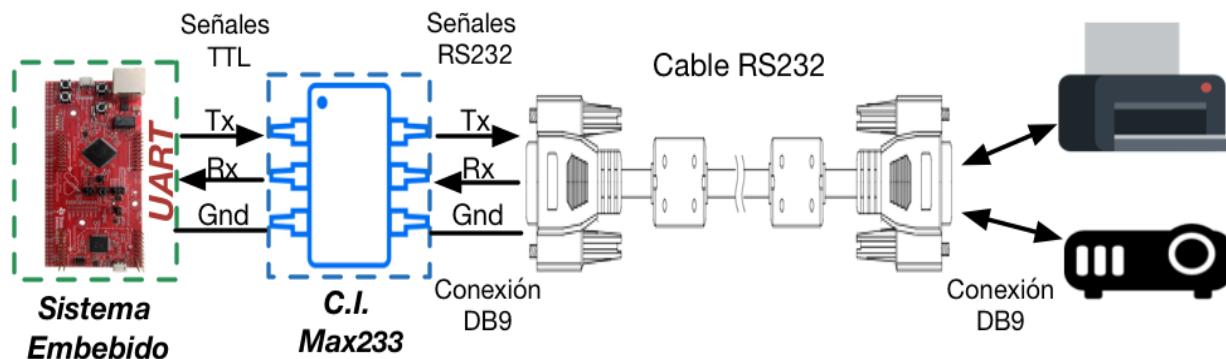


Figura 3.15. Comunicación bajo la norma RS232 entre un microcontrolador con módulo UART y un dispositivo electrónico: proyector, impresora, etc.

CAPÍTULO 4

DISEÑO DEL SISTEMA DE CONTROL CENTRADO EN EL PROFESOR

En el presente capítulo se muestra el diseño del sistema de control centrado en el profesor. Para ello, se comenzará por presentar el análisis de la tarea del profesor haciendo uso del método HTA, prestando especial atención en aquellas actividades que impliquen el uso de elementos tecnológicos. Con base a ello, se van a determinar los requerimientos de control del sistema, esto es, qué dispositivos se van a controlar, qué funcionalidades va a realizar el sistema, etc. Para que finalmente, se presente una solución de diseño del sistema de control que responda a las necesidades del profesor.

4.1 Análisis de la tarea del profesor

Como se mencionó en el capítulo 2, el método HTA (Hierarchical Task Analysis) se centra en descomponer las tareas del usuario en sus sub-tareas, siguiendo un orden jerárquico y describiendo el orden en que se realizan. Esta descomposición o descripción de tareas se puede hacer en forma de diagrama, así como textual.

La *figura 4.1* presenta un diagrama en el que se muestra el análisis de la tarea del profesor haciendo uso del método HTA. El análisis toma como punto de inicio la tarea del profesor “Dar clases”, la cual se puede organizar tomando como base tres momentos: lo que se hace antes, durante y después de clase. Dicho de otra forma, el profesor divide esta tarea en tres sub-tareas fundamentas, que son:

1. Preparar clase
2. Impartir clase
3. Evaluar clase

Todas ellas se deben pensar en un orden secuencial (SEC) de ejecución, esto es, para dar una clase primero se debe preparar, después impartir y por último evaluar.

4. Diseño del sistema de control centrado en el profesor

La primera tarea que corresponde a la preparación de la clase, se puede dividir en las siguientes sub-tareas y pueden ser ejecutadas de forma paralela (PAR):

- 1.1** Agendar clase
- 1.2** Planear clase
- 1.3** Preparar materiales y contenido
- 1.4** Preparar evaluaciones

La segunda tarea que corresponde a impartir una clase, se puede dividir en las siguientes sub-tareas y pueden ser ejecutadas de forma paralela (PAR):

2.1 Dar cátedra. El profesor tiene, generalmente, dos alternativas (ALT) mediante las cuales puede llevarla a cabo (aunque también se pueden llevar a cabo en conjunto). Estas son:

2.1.1 Mediante una computadora (PC). Cuando la cátedra se imparte haciendo uso de una computadora, el profesor debe realizar las siguientes tareas específicas de forma secuencial (SEC):

2.1.1.1 Encender dispositivos (PAR):

2.1.1.1.1 Computadora

2.1.1.1.2 Proyector (es)

2.1.1.2 Transferir archivo (ALT):

2.1.1.2.1 Por internet

2.1.1.2.2 Por dispositivo de almacenamiento externo

2.1.1.3 Ejecutar archivo transferido en la PC (e. g. Aplicación, presentación, video, etc.)

2.1.1.4 Impartir clase

2.1.1.5 Apagar dispositivos (PAR):

2.1.1.5.1 Computadora

2.1.1.5.2 Proyector (es)

2.1.2 Mediante un pizarrón

2.2 Dirigir y controlar las actividades del grupo

4. Diseño del sistema de control centrado en el profesor

2.3 Asignar trabajos y aplicar exámenes

2.4 Responder preguntas y resolver dudas

La tercera y última tarea que corresponde a la evaluación de la clase, se puede dividir en las siguientes sub-tareas y pueden ser ejecutadas de forma paralela (PAR):

3.1 Revisar trabajos y exámenes

3.2 Dar retroalimentación

3.3 Calificar y registrar el aprendizaje

3.4 Calificar su desempeño personal

Por otro lado, tenemos que, como mencionó en la introducción de esta tesis, el Aula del Futuro cuenta con distintas zonas interactivas de trabajo (ZIT), entre las principales se encuentran: un Muro Colaborativo y cuatro Escritorio Colaborativo Aumentado (ECA).

Para hacer uso de cualquier de estas ZIT, según el análisis de la tarea descrito anteriormente, el profesor debe realizar para cada una de ellas de entre 5 a 6 tareas puntuales que impliquen el uso de algún dispositivo electrónico. Estas tareas son:

- 1.** Encender PC.
- 2.** Encender proyector (es).
- 3.** Transferir archivo a la PC.
- 4.** Ejecutar archivo en la PC.

Una vez terminada la clase:

- 5.** Apagar proyector (es).
- 6.** Apagar PC.

En conclusión, cuando el profesor requiera hacer uso de los cuatro ECA y el Mural Interactivo en una de sus cátedras, tendría que llevar a cabo entre 25 y 30 acciones.

4. Diseño del sistema de control centrado en el profesor

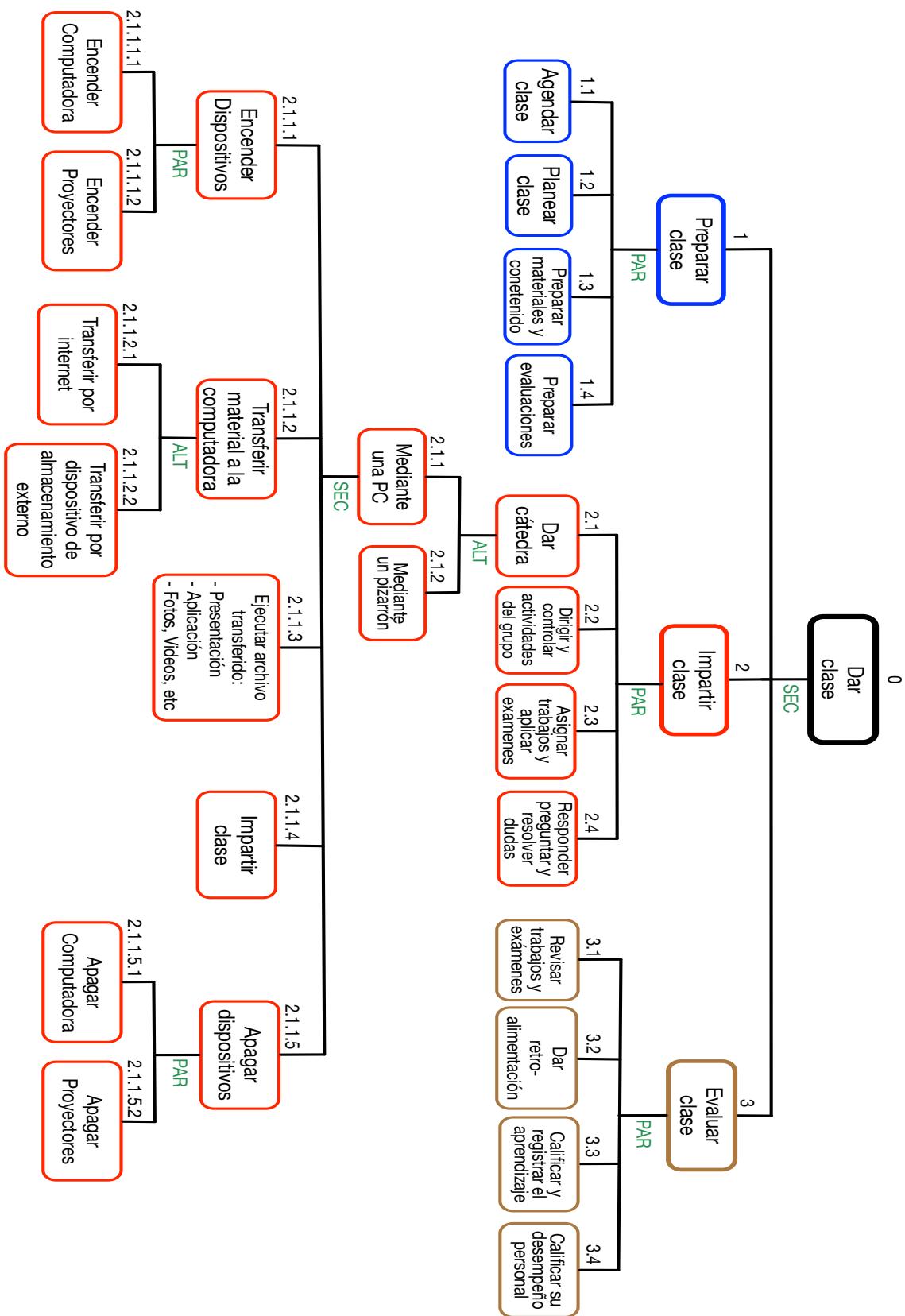


Figura 4.1. Análisis de la tarea del profesor haciendo uso del método HTA.

4. Diseño del sistema de control centrado en el profesor

4.2 Requerimientos del sistema

Una vez hecho el análisis de la tarea del profesor, pasaremos a mencionar los requerimientos del sistema de control, los cuales van a permitir definir las principales funcionalidades que el profesor podrá realizar desde la interfaz Web.

Con el fin de cumplir con los objetivos propuestos en esta tesis, se plantea que los requerimientos de control del sistema a desarrollar sean aquellas tareas o necesidades del profesor asociadas al uso de las zonas interactivas de trabajo o ZIT (Mural y ECA), tal y como se describieron en el análisis de la tarea en la sección anterior. Esto es, el sistema debe permitir:

- Encender los proyectores y computadoras ligadas a cada una de las ZIT
- Subir un archivo y proyectarlo/ejecutarlo en cualquiera de las ZIT
- Apagar los proyectores y computadoras ligadas a cada una de las ZIT

Por otro lado, con el fin de reducir en gran medida el número acciones o tareas (y por lo tanto el tiempo) para el uso o acceso a las ZIT, el sistema debe permitir:

- Encender, al mismo tiempo, todos los proyectores y computadoras ligadas a las ZIT
- Apagar, al mismo tiempo, todos los proyectores y computadoras ligadas a las ZIT
- Programación de sesiones. Esto es, que el profesor pueda encender las ZIT a una hora y fecha indicada, para que así, todas las zonas de trabajo que tiene pensado usar en alguna de sus cátedras esté lista para su uso.

Como últimos requerimientos, que aunque no cubren ninguna necesidad específica del usuario, pueden brindar una gran flexibilidad en el uso de las ZIT. Esto es, el sistema debe permitir:

- Agregar o dar de alta nuevas computadoras y proyectores en el sistema y hacerlo independiente de marcas y modelos. Por un lado, esto va permitir adaptar nuevos dispositivos que lleguen al aula (PC's y proyectores) y hacer uso de ellos en alguna de las ZIT. Por otro lado, debido a que actualmente las ZIT del aula cuentan con proyectores de distintas marcas y modelos, es necesario tener un control remoto para cada uno, con lo que resulta tedioso su control o acceso; para solucionar ello se propone que el sistema de control permita al profesor acceder a ellos desde la interfaz Web de una forma sencilla, sin preocuparse por dónde están estos controles remotos de los proyectores.

4. Diseño del sistema de control centrado en el profesor

- Poder realizar configuraciones en la composición de cada ZIT, esto es, poder seleccionar (desde la interfaz Web) la computadora y proyector con la cual va a operar.

Tener en cuenta todos estos requerimientos va a permitir al profesor reducir el número de tareas para el uso o acceso a las ZIT (Mural y ECA) del aula, lo que también se traduce en una reducción en tiempo y esfuerzo. Esto es, lo que antes podría implicar llevar acabo de entre 25 a 30 tareas para acceder o hacer uso de todas las ZIT, ahora implicaría realizar un par de ellas.

Por otro lado, tampoco tendrá que preocuparse por el procedimiento para acceder a ellas de la forma en que se hace actualmente, esto es, tener a la mano los controles remotos para los distintos proyectores o tener que ir/acceder a cada una de las PC de cada ZIT para encenderla, apagarla o ejecutar una aplicación o archivo.

4.3 Diseño de la solución

En esta sección del capítulo se presenta una solución de diseño para el sistema de control a desarrollar, tomando en cuenta los requerimientos establecidos anteriormente.

Se propuso un diseño del sistema que trabaje bajo una arquitectura cliente-servidor. Esto debido a que es posible la distribución de funciones entre dos tipos de entidades autónomas e independientes: cliente y servidor; donde el servidor ofrece un servicio y un cliente solicita un servicio al servidor (para más información acerca de esta arquitectura ir a la sección 3.5 del capítulo 3).

Para el caso que nos ocupa, lo anterior se puede traducir a que el profesor desde una interfaz Web (cliente) solicita a un servidor el servicio de control o acceso a las distintas ZIT.

La figura 4.2 es un diagrama de bloques en el que se muestra el diseño de solución del sistema de control bajo la arquitectura antes mencionada.

4. Diseño del sistema de control centrado en el profesor

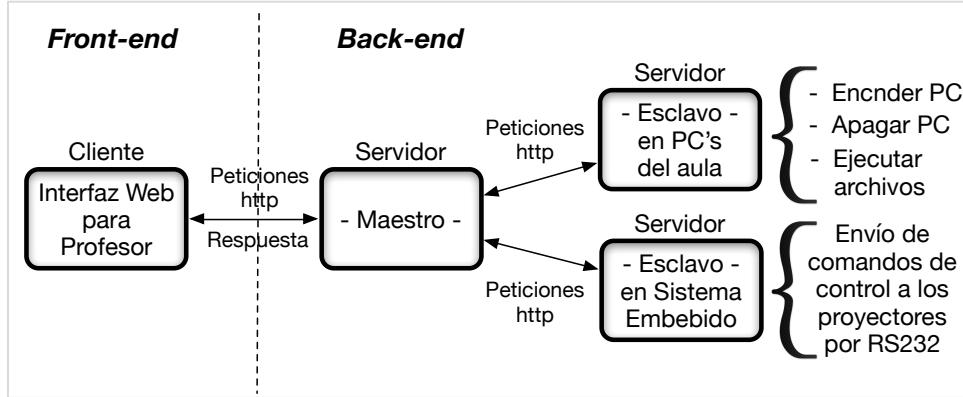


Figura 4.2. Solución de diseño del sistema de control.

Del lado izquierdo de la *figura 5.1* se muestra la parte del cliente, también llamada *Front-end*, la cual corresponde a una interfaz Web con la que el cliente (profesor) envía peticiones o solicitudes a un servidor indicando que se requiere realizar alguna tarea o acción de control en el sistema.

Del lado derecho se muestra la parte del servidor, también llamada *Back-end*, la cual se encarga de recibir las peticiones del cliente y ejecutar rutas de código específicas con el fin de dar respuesta a las peticiones de control del cliente, tales como el encendido y apagado de proyectores y computadoras asociadas a cada una de las ZIT, ejecutar archivos en cada una de las ZIT, etc.

Esta última parte (*back-end*) consta de dos tipos de servidores (http), uno llamado Servidor Maestro y los otros llamados Servidores Esclavos. El primero de ellos se ejecuta en uno de los servidores ubicados dentro del aula y es el encargado de recibir directamente las peticiones del cliente (profesor) y de gestionar el control de los dispositivos dentro del aula. Los otros servidores (esclavos) se ejecutan en las computadoras que conforman a cada una de las ZIT y en un sistema embebido, y los cuales se encargan de recibir órdenes (peticiones http) de control del Servidor Maestro con el fin de que ejecute ciertas tareas y cumplir con la petición inicial del cliente.

Por ejemplo, las tareas a ejecutar por los servidores en las PC's de cada ZIT son tres principalmente: Encender PC ligada a la ZIT, Apagar PC ligada la ZIT y Ejecución de archivo en la PC ligada a la ZIT. Mientras que el servidor esclavo que se ejecuta en el Sistema embebido tiene la principal tarea de envío de comandos de control (por RS232) a todos los proyectores ligados a cada una de las ZIT.

En el siguiente capítulo 5 “Desarrollo de la solución” se realiza el desarrollo e implementación del diseño antes mencionado y se dan más detalles acerca del funcionamiento.

CAPÍTULO 5

DESARROLLO DE LA SOLUCIÓN

En este capítulo se presenta el desarrollo de la solución de diseño descrito en el capítulo anterior. Primero se da una descripción general acerca del funcionamiento del sistema de control dentro del Aula del Futuro, así como las partes que lo componen.

En un segundo momento se explica el funcionamiento del sistema a detalle, dividiendo el contenido en dos secciones, primero se menciona el desarrollo del cliente (*Front-end*), el cual corresponde a la aplicación o interfaz Web desde la cual el usuario (profesor) va interactuar con el sistema. Por último se presenta el desarrollo del lado servidor (*Back-end*), el cual se encarga de enviar los comandos y las informaciones conducentes a los diferentes dispositivos dentro del aula.

5.1 Descripción general del funcionamiento del sistema

La *figura 5.1* es un diagrama que representa una vista aérea del AF y en la que se muestran cómo están distribuidas las cinco ZIT, las partes que componen al sistema de control desarrollado, el modo o medio de comunicación entre estos (en color café), los protocolos utilizados en cada componente (color azul) así como los lenguajes o plataformas de programación utilizados en cada parte del sistema (en color verde).

En la parte inferior izquierda se muestra al profesor, y el cual puede ingresar a la interfaz Web desde cualquier dispositivo inteligente que lleve consigo (como *Smartphone*, tableta, laptop). Desde esta aplicación o interfaz Web, el profesor envía peticiones http a un servidor de nombre Servidor Maestro al cual se le solicita la ejecución de cierta tarea, que como se mencionó anteriormente, puede ser el encendido o apagado de computadoras y proyectores asociados a una ZIT, subir un archivo al servidor maestro y ejecutarlo/proyectarlo en alguna de las cinco ZIT, etc. Por otro lado, el Servidor Maestro puede tomar el rol de cliente y comunicarse con otros servidores presentes dentro del aula llamados Servidores Esclavos (que se ejecutan en cada una de las PC de las ZIT), enviando a estos peticiones http con el propósito de delegar partes de la tareas antes mencionadas y así dar solución a las peticiones del cliente (profesor).

5. Desarrollo de la solución

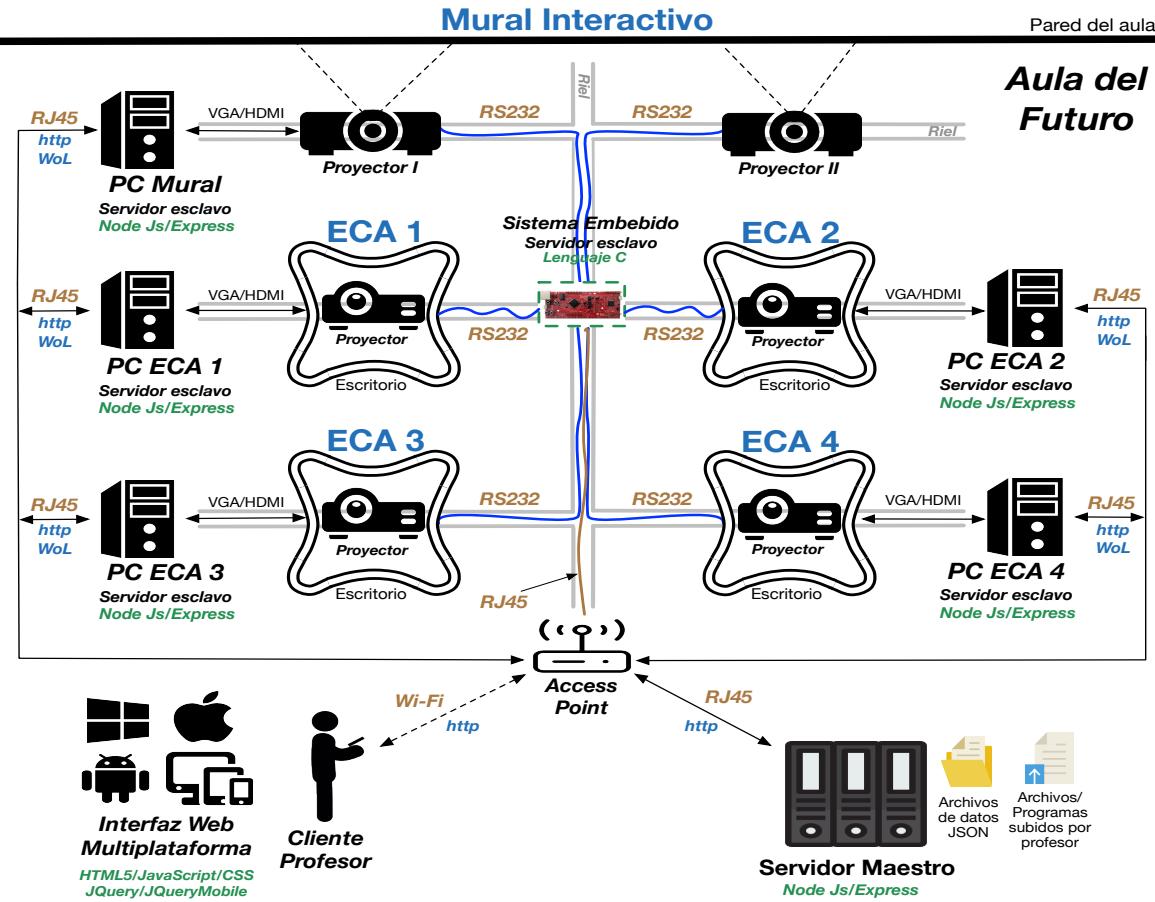


Figura 5.1. Diagrama que representa una vista aérea del Aula del Futuro en la que se muestra la distribución de las zonas interactivas de trabajo pertenecientes al Muro Interactivo y el ECA, así como las partes que componen al sistema de control.

Para la tarea de encendido de una computadora se emplea el protocolo *Wake On LAN* (WoL); para las demás tareas de control relacionadas a ésta se utiliza el protocolo http.

Para el control de los proyectores se hace uso del protocolo RS232, realizando una comunicación punto a punto entre el puerto serial del proyector y un sistema embebido (SE) que integra módulos UART (para más información acerca del SE utilizado ir a la sección 3.9 del capítulo 3).

Este SE se encuentra montado sobre un riel en el techo del aula, con el fin de ubicarlo en un punto medio a todos los proyectores, los cuales también se encuentran montados sobre los rieles. Para la comunicación entre SE y proyectores se utilizan cables RS232 macho-macho tipo *null-modem* y de tres metros de longitud.

5. Desarrollo de la solución

Se optó por hacer uso de este SE debido a que integra varios periféricos e interfaces de comunicación. Entre ellos un puerto *Ethernet* para su conexión a una red de área local; esto permite incrementar el alcance de comunicación e interacción con otros dispositivos dentro de la red, como sensores, actuadores, otros SE, dispositivos de audio o video, servidores, etc. Siendo ideal su uso en un proyecto como lo es “el Aula del Futuro”, en el que uno de sus objetivos es poner a disposición del profesor múltiples sistemas informáticos dentro de un aula educativa.

Para el caso que nos ocupa, el SE se programó con el propósito de utilizarlo como un servidor (esclavo) http y así realizar peticiones a este desde otro servidor (maestro) con el fin de delegar la tarea de control de los proyectores pertenecientes al Mural y ECA’s.

5.2 Desarrollo de Front-end

La interfaz Web se desarrolló de acuerdo a los requerimientos del sistema de control descritos en el capítulo anterior, los cuales a su vez se derivan de las necesidades o tareas del profesor para el uso de una ZIT (mediante el análisis de la tarea).

En esta sección se presentan los componentes de la interfaz Web así como sus principales funcionalidades, las cuales se enlistan a continuación:

- Encendido y apagado de las ZIT
- Proyección de archivo en ZIT
- Programación de ZIT
- Configuraciones en el sistema

En las siguientes secciones se menciona, principalmente, cómo se accede y se llevan a cabo estas funciones en la interfaz Web, así como la forma en que se hacen las peticiones o solicitudes del profesor al Servidor Maestro a nivel código. Esto con el fin de comprender de mejor forma el funcionamiento del sistema y que sirva de apoyo para futuros desarrolladores del proyecto.

La interfaz Web se programó en los lenguajes HTML5, *JavaScript* y CSS, en conjunto con una librería de *JavaScript* llamada *JQuery* y el marco de trabajo *JQuery Mobile*. Se hizo uso del editor de texto *Sublime Text* para escribir todo el código y el navegador *Google Chrome*, junto con las herramientas de desarrollo que ofrece, para realizar pruebas de funcionamiento y depuración de código.

5. Desarrollo de la solución

A lo largo del capítulo se usa la palabra módulo y ZIT de forma indistinta, donde ambos hacen referencia a las zonas de trabajo o tecnologías del Mural Interactivo y/o ECA.

5.2.1 Componentes de la Interfaz Web

En la *figura 5.2* se muestra la página principal de la interfaz Web desarrollada con cada uno de sus componentes. Se explicarán cada uno de estos con el fin de comprender su manejo y funcionamiento.

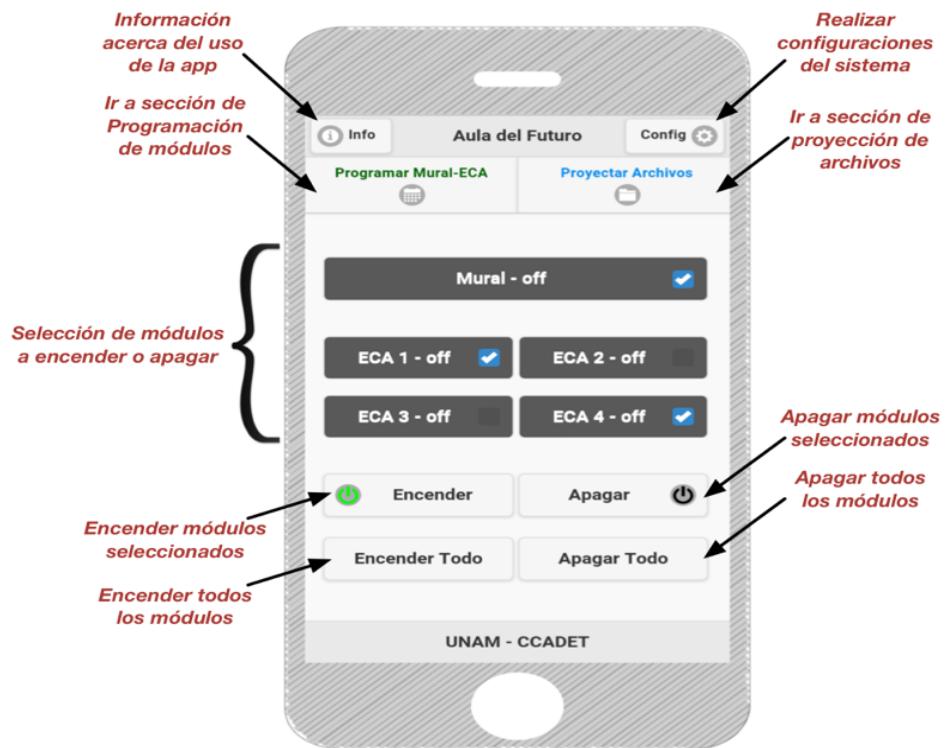


Figura 5.2. Página principal de la interfaz Web.

En la parte superior se encuentran dos botones, el del lado izquierdo llamado “*info*” despliega un panel de información acerca del funcionamiento de la app y describe cómo utilizarla; el del lado derecho llamado “*config*” despliega un panel en el que se muestran tres botones principales llamados: *Mural-ECA*, *Proyectores*, *Computadoras* (*figura 5.3*). Cada uno de ellos direccionan a otra interfaz (pantalla) en la que el usuario puede realizar configuraciones al sistema, tales como agregar nuevos proyectores o computadoras para su control, configuraciones en Mural y ECA, etc. Esta parte de configuraciones se verá con más detalle posteriormente.

5. Desarrollo de la solución

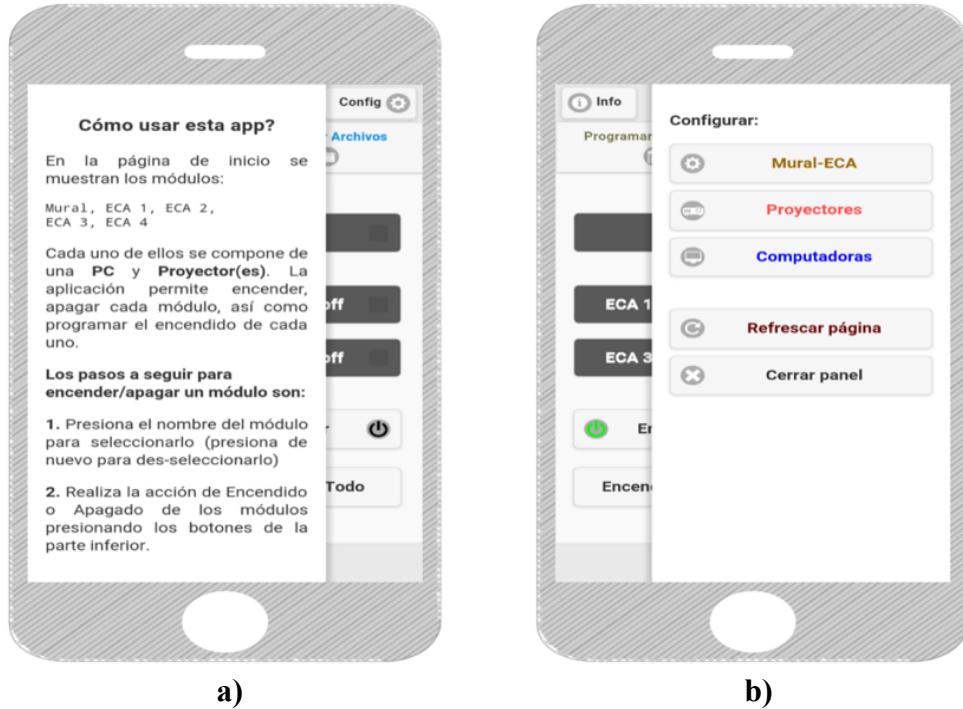


Figura 5.3. a) Panel de información acerca del uso de la app.
b) Panel para realizar configuraciones en el sistema.

En la siguiente sección de la página principal se encuentran dos botones de navegación, uno del lado izquierdo y con letras de color verde llamado “*Programar Mural-ECA*” y el otro del lado derecho y con letras de color azul llamado “*Proyectar Archivos*”. El primero de ellos permite programar el encendido de una ZIT a la hora y fecha indicada por el usuario; mientras que el segundo de ellos permite subir un archivo (foto, video, presentación, etc.) al servidor maestro y proyectarlo/ejecutarlo en cualquiera de las ZIT.

En la siguiente sección de la página principal se encuentran cinco botones de selección en color negro. La ubicación de estos en la interfaz Web es similar a la ubicación de cada módulo o ZIT dentro del aula. El primero y más largo hace referencia al Mural Interactivo y se representa con el nombre de “Mural – off”. Los otros cuatro botones corresponden a los cuatro ECA y se representan con el nombre de “ECA – off”. El color negro de cada botón y la palabra “off” hacen alusión a que ese módulo se encuentra apagado, esto es, tanto la computadora como el proyector usados en él se encuentran apagados.

5. Desarrollo de la solución

En la parte inferior de la app se muestran cuatro botones llamados “*Encender*”, “*Apagar*” y “*Encender Todo*”, “*Apagar Todo*”; los cuales son los encargados del envío de peticiones http al servidor maestro indicándole que se requiere el encendido o apagado de los módulos o ZIT.

5.2.2 Encendido y Apagado de las ZIT

Para realizar el encendido o apagado de un módulo o ZIT basta con dar clic o presionar el botón con el nombre de este para seleccionarlo (una paloma de color azul del lado derecho del nombre de cada módulo indica que este fue seleccionado) y en seguida presionar uno de los botones de la parte inferior “*Encender*” o “*Apagar*”. También es posible encender o apagar todos las ZIT al mismo tiempo sin necesidad de seleccionarlos previamente; esto presionando los botones llamados “*Encender Todo*” y “*Apagar Todo*” de la página principal.

Como modo de ejemplo supongamos que se requiere hacer uso del Mural Interactivo, ECA 1 y ECA 4. El usuario/profesor del aula debe seleccionar estas tres ZIT y presionar el botón “*Encender*”. Este botón cuyo identificador (*id*) dentro del documento es “*turnOnEcaMuralb*” está atado o ligado al evento clic, por lo que cada vez que es presionado se ejecuta la siguiente función.

```
1  $("#turnOnEcaMuralb").on("click",function(e) {  
2  
3      //1. Tomar módulos seleccionados por el usuario  
4      //2. Envío de peticiones HTTP al servidor por cada modulo seleccionado  
5      //3. Tomar respuesta del servidor y notificar al usuario  
6  
7  });
```

En esta función se realizan tres tareas fundamentales: primeramente se toman los módulos seleccionados por el usuario, posteriormente se envía una petición http con AJAX al servidor maestro por cada ZIT seleccionada, donde este servidor comenzará el proceso de comunicación con los servidores esclavos y envío de comandos para encender proyectores y computadoras correspondientes al Mural, ECA1 y ECA4. Finalmente, el servidor maestro retorna el resultado de la operación al usuario, indicándole si fue posible encender las ZIT solicitadas o hubo algún error en el proceso. Si el encendido del Mural Interactivo fue correcto, la interfaz le notifica al usuario cambiando el nombre en el botón a “*Mural – on*” y cambiando su color a verde.

5. Desarrollo de la solución

Este procedimiento es similar para el apagado de ZIT seleccionados así como para el encendido o apagado de todas las ZIT al mismo tiempo.

5.2.3 Proyección de archivos en ZIT

Esta parte de la aplicación permite subir un archivo al servidor maestro y proyectarlo o ejecutarlo en alguno de las ZIT del aula. A esta interfaz se accede con el botón “*Proyectar Archivos*” desde la página de inicio. En la *figura 5.4* se muestra la interfaz correspondiente a la sección de proyección de archivos.



Figura 5.4. Interfaz de usuario para proyección/ejecución de archivos en módulos.

Esta interfaz se compone principalmente de tres secciones, la primera es un contenedor de archivos donde el usuario selecciona el archivo a proyectar. Los nombres de todos los archivos disponibles subidos por el usuario se almacenan en un arreglo de nombre *fileNameArray[]*; los íconos de suma (+) y el bote de basura de la parte superior derecha, son botones que sirven para subir un archivo nuevo al servidor y eliminar un archivo del servidor, respectivamente. La segunda sección es la parte de selección de lugar de proyección, en la cual el usuario elige la ZIT (Mural, ECA 1 2 3 y 4) en donde desea que el archivo seleccionado sea proyectado. La tercera sección corresponde

5. Desarrollo de la solución

al botón de nombre “*Proyectar*” de la parte inferior y es el encargado de enviar la orden de proyección al servidor maestro. Este botón o elemento posee el id “*proyectarFile*” y de igual forma está ligado al evento clic que al presionarlo se ejecuta el siguiente código.

```
1 $("#proyectarFile").click(function() {
2
3     //Tomar archivo y modulo seleccionado por el usuario
4     var currentFile = $('input[name=files]:checked').val();
5     var currentModule = $('input[name=Module]:checked').val();
6     var currentModuleAux = $("#moduleSelected").text();
7     $("#headerDeleteFile").text("Alerta");
8
9     //No ha sido seleccionado un archivo y un lugar de proyección ??
10    if ((currentFile == undefined) || (currentModuleAux == "-")) {
11        $("#conentPopupFiles").text("Debes seleccionar un archivo y un lugar
12                                de proyección!")
13        $("#mustSelectMorF").popup( "open" ); //Abrir cuadro de dialogo
14    }else{
15        //Llamada a función que envía petición HTTP a PC del módulo selecc.
16        startfile(currentFile, currentModule);
17    }
18
19});
```

Las instrucciones de la línea 4 y 5 toman el archivo seleccionado por el usuario y el lugar de proyección, respectivamente. El condicional *if* de la línea 10 revisa si han sido seleccionados los parámetros antes mencionados, de no ser el caso, se notifica al usuario que se deben seleccionar para poder continuar; de haberse seleccionado se llama a la función de nombre *startfile()*, la cual recibe dos parámetros: el archivo seleccionado y el lugar de proyección indicado por el usuario.

```
1 function startfile(file, module){
2
3     //Envío de petición HTTP con AJAX al servidor maestro
4     //En los parámetros de la URL se envían el nombre del archivo
5     //y el lugar de proyección
6     $.ajax({
7         data: {nameFile: file, modulo: module}, //Datos
8         url: "/startFile" //URL a localizar en servidor
9     }).done(function(result){ //Función callback, al terminar petición
10        console.log(result)
11    });
12};
```

5. Desarrollo de la solución

En esta función `startfile()`, de la línea 6 a la 11 se lleva acabo el envío de una petición http con AJAX al servidor maestro con la URL “/startFile” y como datos el nombre del archivo y el módulo especificado. Una vez recibida por el servidor maestro, este se encargará de comunicarse con el servidor esclavo que se ejecuta en la computadora del módulo indicado, enviando a este una petición http indicándole que se requiere la ejecución de cierto archivo.

5.2.4 Programación de ZIT

Esta parte de la aplicación permite programar o agendar el encendido de una ZIT o módulo a la fecha y hora especificada por el usuario/profesor. A esta interfaz se accede con el botón “*Programar Mural-ECA*” desde la página de inicio.

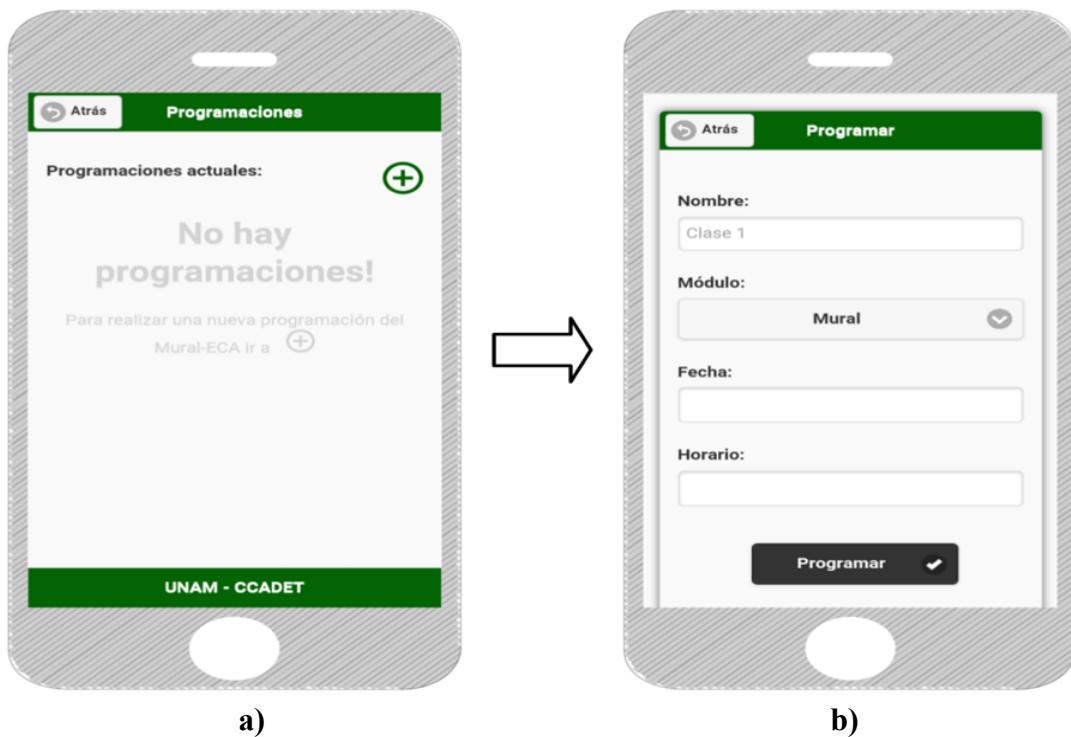


Figura 5.5. a) Programaciones actuales hechas por el usuario. b) Interfaz para realizar la programación de un módulo.

Una vez que se ha ingresado, la primera pantalla que aparece frente el usuario es la que se muestra en la figura 5.5 (a). En esta pantalla el usuario puede observar las programaciones que se hayan hecho para algún módulo o ZIT. Por otro lado, para realizar una nueva programación, se debe

5. Desarrollo de la solución

presionar el signo (+) de color verde de la parte superior derecha, mostrando la pantalla que se muestra en la *figura 5.5 (b)*. En esta última interfaz se requiere que se ingrese un nombre, el módulo, la fecha y la hora para el encendido del módulo; una vez ingresados los datos se debe presionar el botón “*Programar*” de la parte inferior, cuyo *id* es “*programECAMuralSubmit*”.

El siguiente código se ejecuta al detectar el evento clic sobre el elemento.

```
1  $("#programECAMuralSubmit").on("click",function(e) {  
2  
3      //1. Validar datos ingresados por el usuario (Nombre, Hora y fecha)  
4      //2. Envío de petición HTTP al servidor maestro indicando que se requiere  
5          programar un módulo a la hora y fecha indicada. Y guarda datos  
6      //3. Retorno de respuesta del servidor y notificar al usuario si fue  
7          posible programar el modulo. Crear info en programaciones actuales  
8  });
```

Esta función realiza tres tareas fundamentales, la primera es verificar si los datos ingresados por el usuario son válidos, por ejemplo que la hora y fecha ingresada sea posterior al momento de la programación y evitar que ingresen fechas pasadas, o indicar al usuario que ya existe alguna otra programación de ese módulo para la fecha y hora seleccionada, entre otros. La segunda tarea se ejecuta una vez que se hayan validado los datos ingresados y la cual consiste en el envío de una petición http con AJAX al servidor maestro, indicándole que se requiere la programación de un módulo, posteriormente este guardará los datos de la programación en un fichero. La tercer tarea es el retorno de la respuesta e indicar al usuario si fue posible realizar la programación del módulo.

5.2.5 Configuraciones en el sistema

La aplicación Web provee la opción de realizar configuraciones en el sistema: del *Mural-ECA*, de *proyectores* y de *computadoras*. Como se mencionó anteriormente (requerimientos del sistema, capítulo 4) estas configuraciones van a permitir o tienen el objetivo de hacer más flexible el uso de las zonas de trabajo desde la interfaz Web.

Para ingresar a cada una de ellas se debe dar clic en el botón llamado “*config*” ubicado en la parte superior derecha de la página de inicio. El cual despliega un panel con las tres configuraciones antes mencionada (*figura 5.3*).

5. Desarrollo de la solución

Se comenzará por mencionar qué tipo de configuraciones se pueden realizar en la sección de Mural-ECA, posteriormente con proyectores y por último con computadoras.

- **Configuración de Mural-ECA (ZIT)**

En esta sección se muestra al usuario la configuración actual de cada uno de los módulos. Se presenta para cada uno de ellos la computadora, el proyector y el número de puerto serial asignado. En la *figura 5.6* se muestra una tabla con estas configuraciones; por ejemplo, para el módulo/ZIT Mural se observa que tiene asignado una computadora (PC) de nombre *pcMural*, un proyector de la marca *NEC* modelo *UMM330X/UM280X* y el puerto serial número 2.

The screenshot shows a web-based configuration interface titled 'Config. Módulos'. At the top, there is a button labeled 'Atrás' (Back). Below the title, there is a sub-section titled 'Configurar módulos' with a small icon. The main content area is titled 'Configuración actual de cada módulo'. It contains a table with five rows, each representing a module. The columns are labeled 'Módulo', 'PC:', 'Proyector:', and 'Puerto Serial:'. The data is as follows:

Módulo:	PC:	Proyector:	Puerto Serial:
Mural	pcMural	NEC-UM330X/UM280X	2
ECA 1	pcECA1	NEC-LT280/LT380	3
ECA 2	pcECA2	ViewSonic-PJD6251	1
ECA 3	pcECA3	ViewSonic-PJD6251	6
ECA 4	pcECA4	ViewSonic-PJD6251	7

At the bottom of the interface, there is a dark bar with the text 'UNAM - CCADET'.

Figura 5.6. Tabla de configuraciones para cada uno de los módulos/ZIT.

Cada vez que el usuario indique que se requiere encender o apagar el Mural Interactivo o cualquier ECA, el servidor maestro toma la configuración asignada en ese momento para ese módulo del fichero llamado *valueModules.json*, y comienza el proceso para el encendido o apagado de proyectores y computadoras correspondientes. Dicho proceso se verá en la siguiente sección del capítulo en la parte de *Back-end*.

También es posible configurar la composición de cada módulo, esto es, se puede asignar a cada uno de ellos otra computadora, proyector o puerto serial. En la *figura 5.7 (a)* se muestra la interfaz que permite estas configuraciones. Se accede a esta interfaz dando clic o presionando el botón de nombre “*Configurar módulos*” de la parte superior en la interfaz de la *figura 5.6*.

5. Desarrollo de la solución

El primer campo de selección llamado “*Módulo*” se ingresa el módulo/ZIT a configurar, el siguiente campo llamado “*Computadora*” corresponde al nombre de la PC que se quiere asignar a dicho módulo, seguido del campo “*Proyector*” y por último el puerto serial para proyector.

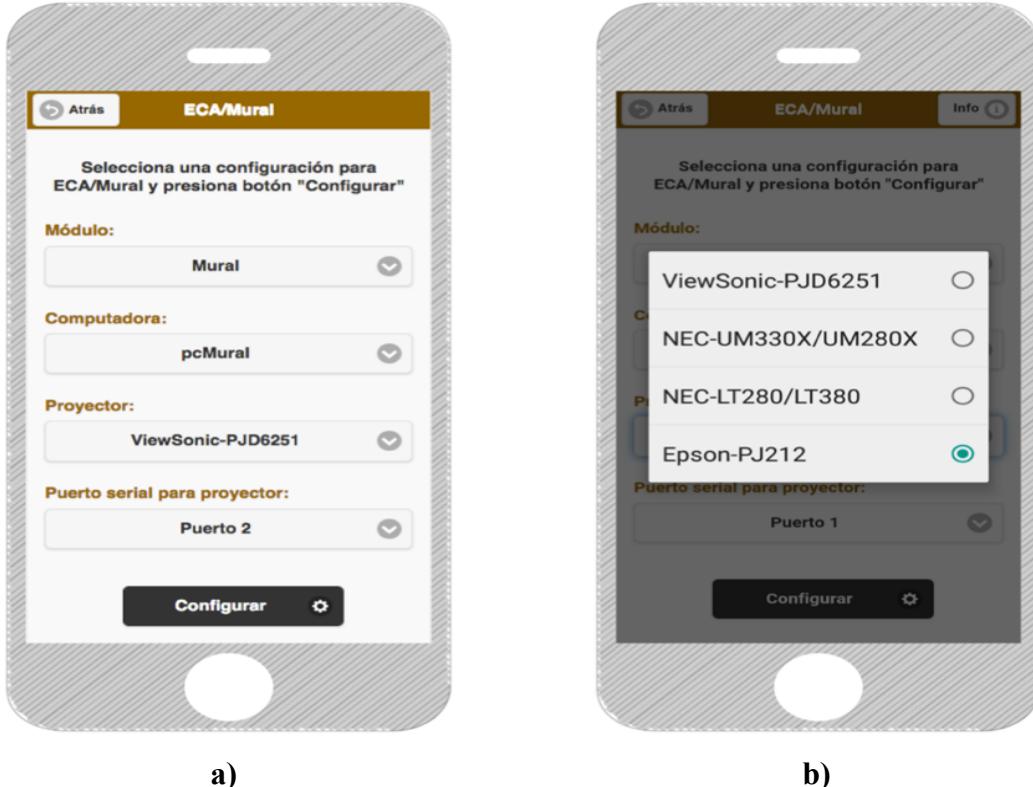


Figura 5.7. a) Interfaz de usuario para realizar las configuraciones de módulos. b) Selección de proyector para asignar a un módulo.

El número de puerto serial va desde el 1 hasta 7. Estos corresponden a conectores DB9 machos que se encuentran en un dispositivo que contiene hasta ocho de estos puertos seriales. Dentro del dispositivo se encuentra el sistema embebido el cual se encarga de generar y enviar los comandos de control a los proyectores del aula haciendo uso de los UART disponibles en este; esta parte se verá más a detalle en la siguiente sección del capítulo.

Por otro lado, en los campos de selección para computadoras y proyectores se muestra al usuario los dispositivos dados de alta y disponibles en el sistema. Por ejemplo, en la figura 5.7 (b) se muestra cada uno de los proyectores con su marca y modelo disponibles para asignar a los

5. Desarrollo de la solución

módulos, siendo dos proyectores de la marca *NEC*, uno *Epson* y uno *ViewSonic*; de igual forma se muestran al usuario las computadoras disponibles.

Una vez seleccionada la configuración deseada para un módulo, se debe dar clic o presionar el botón de color negro de la parte inferior de la pantalla llamado "Configurar", este botón tiene asignado el *id* de nombre "*establecerECAMural*" y está ligado al evento clic. Al presionarlo se ejecuta la siguiente función.

```
1  $("#establecerECAMural").click(function() {
2
3    //1. Tomar configuraciones hechas (modulo, Pc, proyector, Puerto serial)
4    //2. Envío de petición HTTP al servidor maestro indicando que se requiere
5        configurar cierto módulo. Este realiza los cambios correspondiente
6        en el fichero valueModules.json
7    //3. Retorno de respuesta y de fichero valueModules.json modificado al
8        usuario-cliente el cual lo procesa e imprime los cambios hechos en
9        pantalla
10
11});
```

Esta función realiza tres tareas principales; primeramente toma las configuraciones hechas por el usuario para cierto módulo, posteriormente hace una petición http con AJAX al servidor maestro indicándole que se requiere realizar la configuración de cierto módulo. Este toma las configuraciones y hace los cambios correspondientes en el fichero de nombre *valueModules.json*; si fue posible realizar la configuración, el servidor retorna al usuario/cliente dicho fichero, el cual lo procesa e imprime los cambios en la tabla de configuraciones actuales de cada módulo.

- **Configuración de proyectores**

La parte de configuraciones para proyectores se muestra en la *figura 5.8*. En esta parte de la aplicación se muestran los proyectores actuales en el sistema, pudiendo editarlos o eliminarlos y agregar nuevos. En la *figura 5.8 (a)* se muestra la interfaz en donde se enlistan los proyectores actuales del aula. Para observar las características de cada uno de ellos, basta con dar clic o presionar sobre el nombre de este, colapsando una sección de información. En la *figura 5.8 (b)* se muestra la información del proyector marca *NEC* modelo *UM330X/UM280X*, mostrando los comandos en hexadecimal necesarios para encenderlo o apagarlo y la velocidad de transmisión

5. Desarrollo de la solución

para su comunicación por puerto serial; también es posible editar estos parámetros pulsando el botón “*Editar*” en la parte superior o eliminar el proyector de la lista pulsando el botón “*Eliminar*”. Por último, en la *figura 5.8 (c)* se muestra la interfaz de usuario que permite agregar nuevos proyectores al sistema; en los campos de texto se pide ingresar la marca y el modelo, los comandos en hexadecimal para su encendido y apagado, así como la velocidad de transmisión (*baudRate*) que maneja el proyector. Una vez agregado, éste estará disponible para su asignación a alguno de los módulos o ZIT, tal y como se vio en la sección anterior de configuraciones para Mural-ECA.

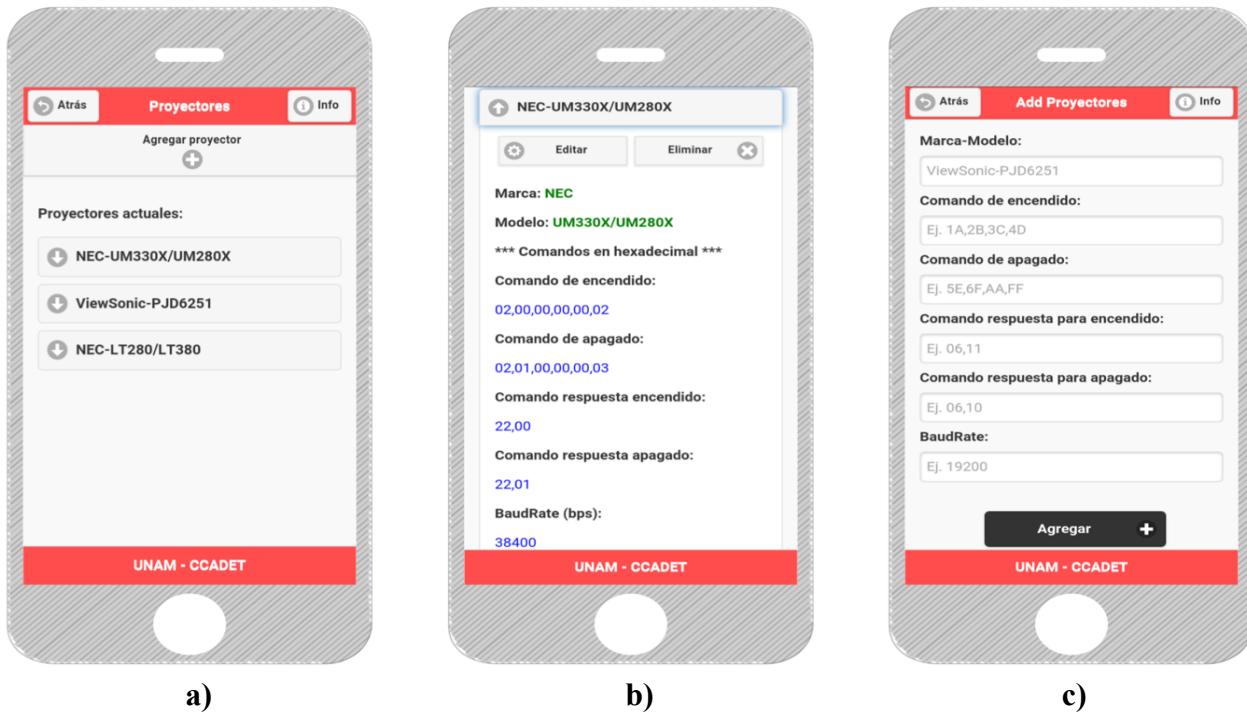


Figura 5.8. Interfaces de usuario para las configuraciones de proyectores en el sistema.

- Interfaz de usuario en la se muestran los proyectores actuales en el sistema.
- Información para cada uno de los proyectores.
- Agregar un nuevo proyector al sistema.

- **Configuración de computadoras**

La parte de configuraciones para computadoras se muestra en la *figura 5.9*. En esta parte de la aplicación se muestran las computadoras actuales en el sistema, pudiendo editarlas o eliminarlas y agregar nuevas. En la *figura 5.9 (a)* se muestra la interfaz en donde se enlistan las computadoras actuales del aula. Para observar las características de cada una de ellos, basta con

5. Desarrollo de la solución

dar clic o presionar sobre el nombre de esta, colapsando una sección de información. En la *figura 5.9 (b)* se muestra la información de la computadora de nombre “*pcMural*” donde se muestra su dirección física (MAC) y su dirección de red (IP). La primera de estas direcciones es usada para realizar el encendido de la PC por la red local y la segunda es usada para identificarla en la red una vez que haya encendido y así realizar operaciones sobre ella.

Es posible configurar el nombre y las direcciones de las computadoras actuales, pulsando el botón “*Editar*” o bien eliminarlas de la lista pulsando el botón “*Eliminar*”, ambas opciones se encuentran en la parte superior. Por último, se muestra la interfaz de usuario de la *figura 5.9 (c)* en la que se agrega una nueva computadora al sistema, introduciendo un nombre (alias), su dirección de red y su dirección física, y a continuación pulsar el botón “*Agregar*” de la parte inferior para guardar los datos en el servidor. Una vez agregada, ésta estará disponible para su asignación a alguno de los módulos o ZIT, tal y como se vio en la sección anterior de configuraciones para Mural-ECA.

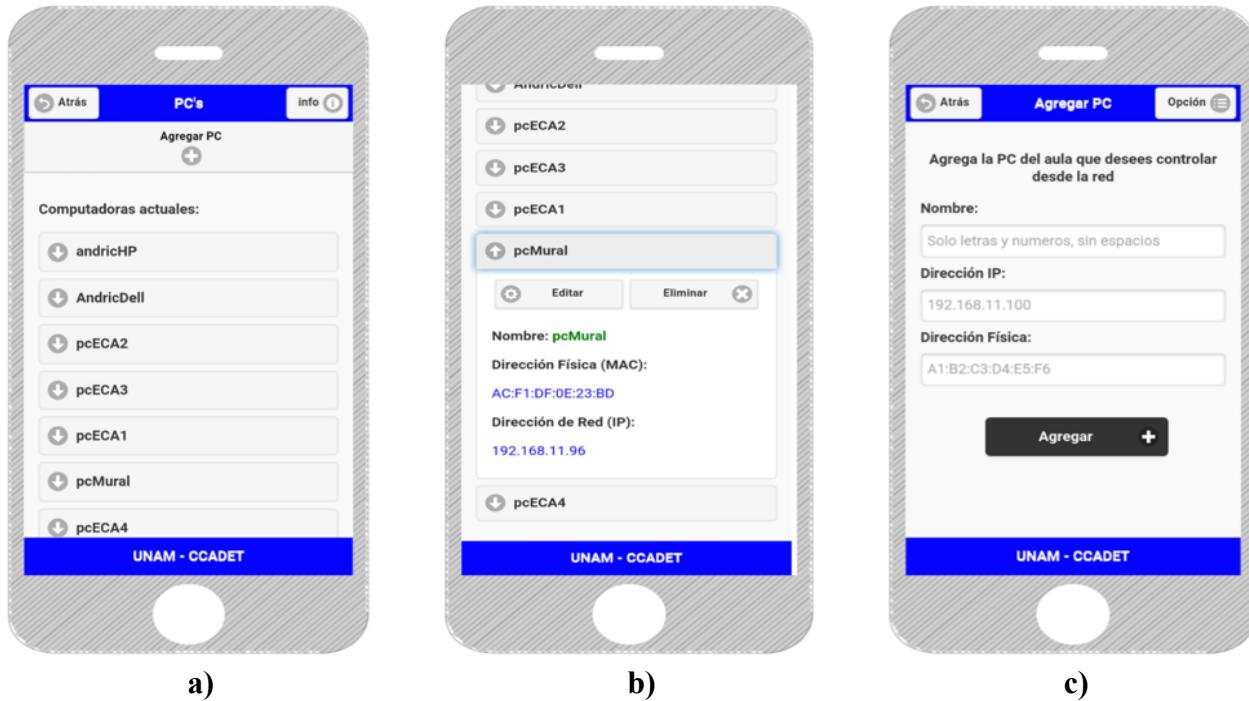


Figura 5.9. Interfaces de usuario para las configuraciones de computadoras en el sistema.

- Interfaz de usuario en la se muestran las PC actuales en el sistema.
- Información para cada una de las computadoras.
- Agregar una nueva PC al sistema.

5. Desarrollo de la solución

5.3 Desarrollo de Back-end

En este apartado del capítulo se presenta el desarrollo de *Back-end*, la cual corresponde a la parte del sistema que se encarga de dar una respuesta a las peticiones o solicitudes hechas por los clientes desde la interfaz Web. El llamado Servidor Maestro es el que se encarga de atender estas peticiones y ejecutar en él rutas que contienen instrucciones de código específico. Una de sus funciones es la de comunicarse con los llamados Servidores Esclavos dentro del aula, con el fin de delegar a estos ciertas tareas que se requieren llevar acabo para dar solución a las solicitudes de los clientes.

La programación del Servidor Maestro y de Servidores Esclavos que se ejecutan en las computadoras se realizó con la plataforma de Node JS en conjunto con el marco de trabajo *Express*. El servidor esclavo que se ejecuta en el sistema embebido está programado en lenguaje C, en el entorno de desarrollo de CCS (*Code Composer Studio*).

Como se vio en la sección anterior, correspondiente a la interfaz Web, el usuario (profesor) puede realizar desde ésta ciertas tareas o acciones, tales como el encendido o apagado de ZIT, programar el encendido de una ZIT, subir archivos al servidor y proyectarlos en una ZIT, configuraciones en el sistema, entre otras. En todas estas acciones está involucrado el uso de proyectores y computadoras, por lo que en esta sección se pretende responder a ciertas preguntas, tales como: ¿Cómo se realiza la tarea de encendido y apagado de computadoras y proyectores de acuerdo a las peticiones del cliente? ¿Cómo se realiza la tarea de proyección de archivos subidos por el usuario (profesor) en alguna de las ZIT? ¿De qué forma y cómo se lleva acabo la comunicación entre el Servidor Maestro y los Servidores Esclavos dentro de la red local? Etc.

Se comenzará por explicar sobre el control de computadoras en LAN y posteriormente sobre el control de proyectores en LAN.

5.3.1 Control de computadoras en LAN

Por control de computadoras se hace referencia a que el usuario puede realizar desde la interfaz Web tres tareas fundamentales sobre esta: encenderla, apagarla y ejecución de archivos.

5. Desarrollo de la solución

A continuación se explica cómo es que se realizan estas tres tareas dentro del sistema. Se comenzará por el encendido de computadoras, posteriormente para el apagado y por último la ejecución de archivos.

- **Encendido de computadoras**

Para el encendido de computadoras en una red de área local se hace uso de la tecnología o protocolo llamado *Wake On Lan* (WOL). Como se mencionó en el capítulo de tecnologías web, WOL consiste en el envío de datagramas (*frames Ethernet*) llamados paquetes mágicos, que contienen un patrón específico de bytes (dirección física o MAC) en el cuerpo del mensaje y que van dirigidos al controlador de red de la computadora destino; que una vez que son correctamente procesados, comienza su proceso de encendido.

Antes de explicar la implementación de esta tecnología dentro del sistema, es necesario mencionar y tener en cuenta que hay requerimientos y/o configuraciones a nivel de software y hardware que hay que llevar acabo en la computadora que se desea encender por la red [27, 28].

Por otro lado, existen diversas aplicaciones disponibles para cualquier plataforma que implementan la tecnología *Wake On Lan*. La aplicación que se uso en este proyecto de tesis como modo de prueba fue una aplicación móvil para sistema operativo *Android*. La aplicación se llama *Wakeup on LAN Free* y se puede descargar de forma gratuita desde *PlayStore*.

En la *figura 5.10* se muestra la interfaz de usuario de la aplicación antes mencionada. En el primer campo de la aplicación llamado “*Address*” se requiere ingresar una dirección IP o un nombre de dominio; dado que el envío de paquetes mágicos debe hacer uso de una transmisión de difusión por *broadcasting*, la dirección IP usada debe ser aquella que permita este tipo de transmisión. La secuencia numérica de esta dirección depende de la clase de red y su configuración, en la red donde se hicieron estas pruebas la dirección IP es *192.168.1.255*.

El siguiente campo “*Port*” se introduce el número de puerto al que irá dirigido el paquete, dado que este tiene que ir dirigido a la tarjeta de red de la computadora, el puerto oficial asignado a este módulo es el 9. El campo siguiente “*Mac address*” es la dirección física de la computadora destino. El último campo “*Address for Ping*” se usa para saber si un host está conectado a la red local, para la realización de estas pruebas no será necesario el uso de este campo.

5. Desarrollo de la solución

Como modo de ejemplo, supongamos que se requiere encender una computadora enlazada a una red local alámbrica y cuya dirección física (MAC) es `11:22:33:44:55:66`. El llenado de estos campos en la aplicación móvil se visualiza en la figura 5.10.

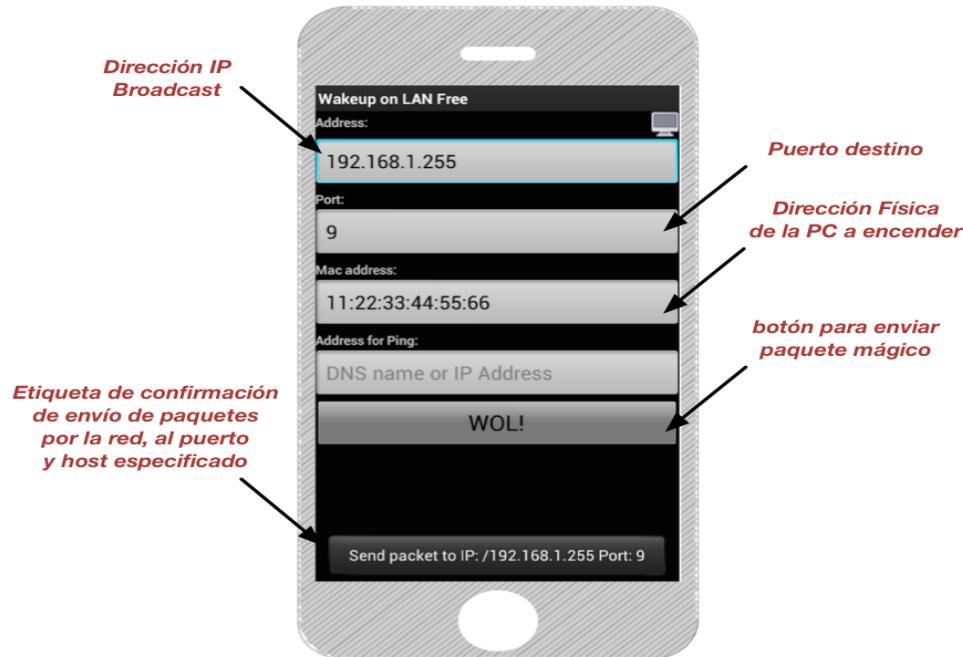


Figura 5.10. Aplicación móvil *Wake on LAN Free* para sistema operativo *Android*.

Una vez enviado el paquete por la red, la tarjeta de red del host destino los procesará y comenzará el encendido de la computadora.

Una desventaja del uso de esta tecnología es que la computadora destino no devuelve algún comando de confirmación (*Acknowledge*) de recibimiento del paquete al host emisor, por lo que resulta complicado saber si este llegó a su destino y logró encender la computadora.

Por otro lado, existen diversas herramientas de software que permiten analizar el tráfico de paquetes en una red local y tener el conocimiento y la certeza de que ciertos paquetes están siendo correctamente enviados desde un host emisor hacia uno destino, y que además, llevan la información correcta a procesar por este. El programa llamado *WireShark* (también conocido como *Ethereal*) es una plataforma utilizada como analizador en tiempo real de paquetes que se transfieren por la red; además es de código abierto, gratuita y ampliamente usado en el ámbito de redes de computadoras [29].

5. Desarrollo de la solución

Para el caso que nos ocupa, ésta fue usada con el fin de reducir la desventaja que presenta el uso de la tecnología WOL y también para el análisis del contenido de paquetes mágicos, para posteriormente implementarlos en la aplicación Web a desarrollar.

En la *figura 5.11* se muestra la interfaz de usuario que presenta *Wireshark*, la cual muestra el tráfico de todos los paquetes que enviados por la red local inalámbrica (*Wi-Fi*). Se encierra en un cuadro de color rojo el paquete mágico que se transmitió por la red desde la aplicación móvil descrita anteriormente.

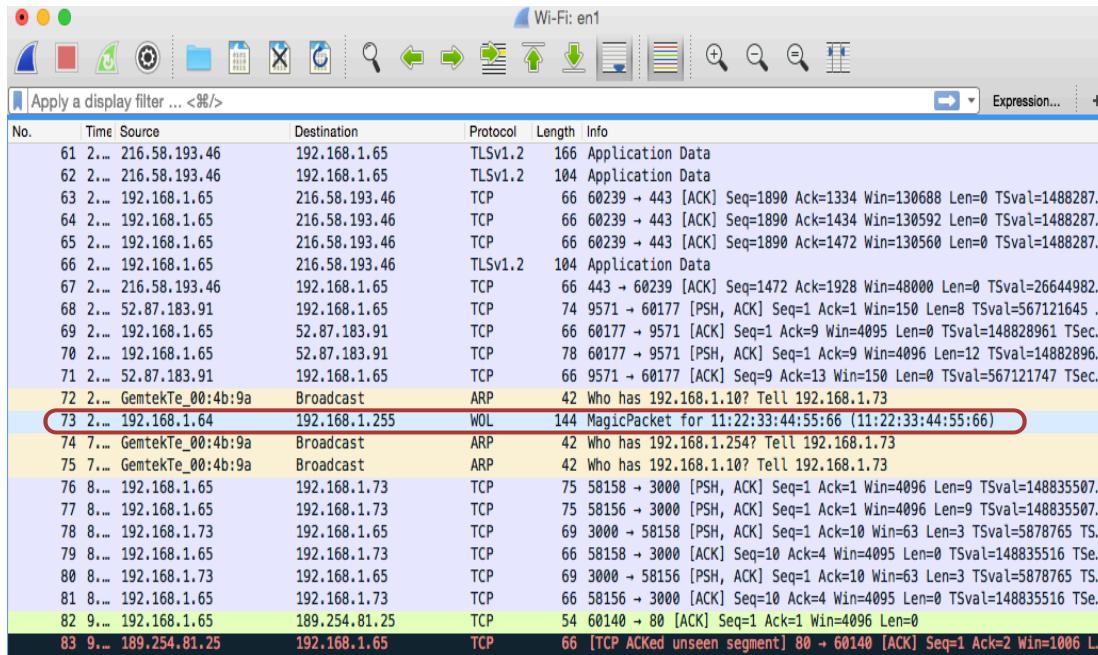


Figura 5.11. Interfaz de usuario de *WireShark* en la que se muestra el tráfico en tiempo real de todos los paquetes detectados en la red local inalámbrica.

Dentro de este cuadro rojo se observa que el protocolo de aplicación es WOL y el protocolo de transporte UDP, los cuales *WireShark* los muestra de un color azul en su fondo. También nos muestra el host fuente (emisor) y el host destino (receptor) y sus respectivas direcciones IP; para el caso de la dirección IP destino se observa que corresponde a la dirección *broadcasting* de la red. Además, se muestra el tamaño del paquete y una parte de información respecto a éste, donde el programa lo detecta como un *MagicPaquet* (paquete mágico) y en seguida muestra la dirección física del host al que va dirigido.

5. Desarrollo de la solución

Para observar y analizar el contenido del paquete a detalle, damos doble clic sobre éste, mostrando la interfaz que aparece en la *figura 5.12*.

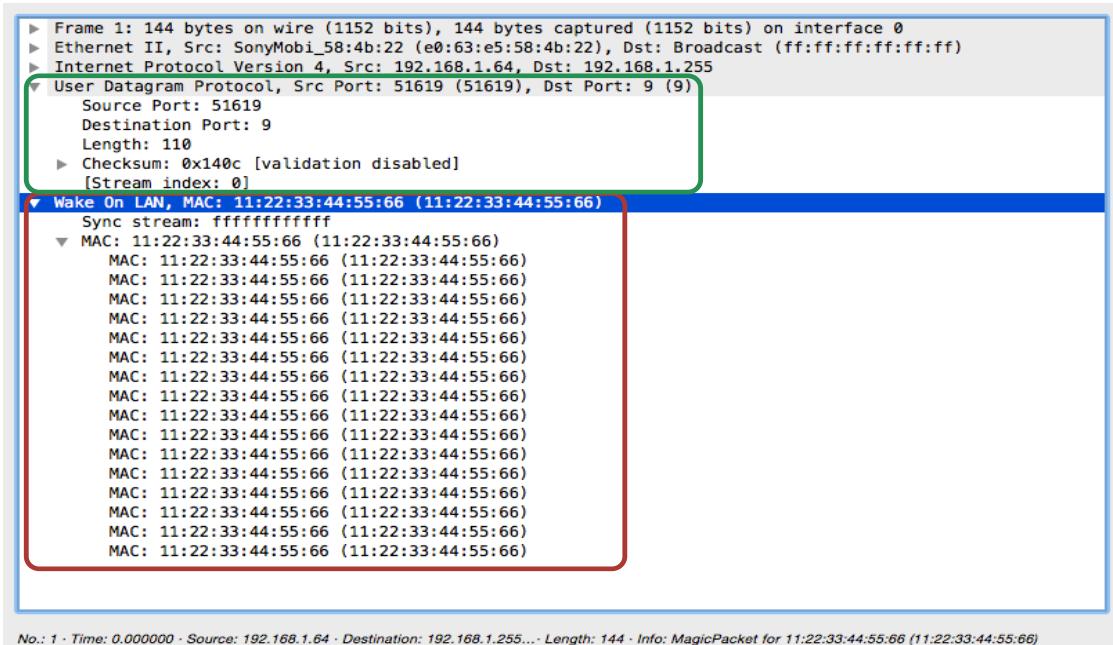


Figura 5.12. Contenido del paquete mágico enviado por la red.

En esta sección se muestra todo el contenido del paquete (*frame Ethernet*) enviado. En los cuadros de color verde y rojo se muestran la información más relevante. El primero de ellos contiene el protocolo de transporte utilizado, puerto fuente, puerto destino y longitud del mensaje. El segundo contiene el nombre de protocolo de aplicación y la secuencia de bytes que componen el paquete mágico, que como se mencionó anteriormente, el patrón comienza con una secuencia de seis bytes de valor *FFh* en hexadecimal (usados para sincronización) y la repetición de 16 veces la dirección física (MAC) del host destino.

Una vez hecho el análisis de paquetes mágicos y haber hecho algunas pruebas de funcionamiento del uso de la tecnología WOL, se pasó a su programación e integración en la aplicación Web.

Como se mencionó anteriormente, el Servidor Maestro se encuentra programado en la plataforma llamada Node JS, la cual contiene una API que provee un módulo para el uso del protocolo UDP, llamado “*dgram*” [62]. Este módulo crea *sockets* de conexión que permiten el envío de datagramas a un puerto y host específico; por lo que este módulo puede ser usado para crear los paquetes

5. Desarrollo de la solución

mágicos correspondientes y encender las computadoras dentro del aula de manera similar a como se aplicó en la parte de pruebas con la aplicación móvil para *Android*.

Como se vio en la sección 5.2 de este capítulo, cada vez que el usuario requiere encender un módulo o ZIT, por ejemplo el Muro interactivo, debe seleccionarlo y presionar el botón “*Encender*”. Al presionarlo se envía una petición http al servidor maestro. La forma de la petición (URL) enviada es: <http://192.168.11.100:3000/turnOnModule?Module=Mural>; donde “http://” indica el protocolo utilizado, “192.168.11.100” es la dirección IP del servidor y el número “:3000” indica el puerto en el que se aceptan conexiones de los clientes; la parte “/turnOnModule” es la ruta de acceso en el servidor y la parte “?Module=Mural” son los datos o parámetros a enviar, que en este caso son para indicar que se requiere el encendido del módulo Mural; ejecutando el siguiente código en el servidor:

```
1  exports.turnOnModule = function(req1, res1) {
2    //Tomar parámetro de la URL (modulo que se require encender)
3    var module = req1.query.Module;
4    //Leer fichero valueModules.json y tomar los valores del módulo
5    fs.readFile(valueModules, 'utf8', function(err,data){
6      var auxData = JSON.parse(data); //Análisis(parseo) de datos de fichero
7      //Loop para encontrar el modulo dentro del fichero
8      for (var j = auxData.Values.length - 1; j >= 0; j--) {
9        if (auxData.Values[j].modulo == module) {
10          var proyector = auxData.Values.proyector; //Tomar proyector
11          var pcName    = auxData.Values.pc;         //Tomar PC
12          var puerto    = auxData.Values.puerto;     //Tomar puerto
13        };
14      };
15
16      //Envío de petición HTTP al Sistema Embebido para dar la orden de
17      //envío de comandos de encendido al proyector por el puerto indicado
18
19      turnOnPc(pcName); //Función para el encendido de la PC atada al módulo
20      req1.end();        //Terminar petición, cierre de conexión
21    }); //Cierre de fichero
21 }; //Cierre de función
```

En la línea 3 se toma el módulo que el usuario requiere encender, después desde la línea 5 a la 15 se toman los valores correspondientes para ese módulo (pc, proyector, puerto) guardados en el fichero de nombre *valuemodules.json*, en la figura 5.13 se muestra una parte de su contenido, mostrando las configuraciones para los módulos Mural y ECA1. En la línea 16, a modo de reducir

5. Desarrollo de la solución

las líneas de código, se expresa textualmente que la siguiente instrucción se envía una petición al sistema embebido indicándole que se requieren enviar los comandos de control para el encendido del proyector por el puerto especificado, esta parte se verá a detalle en el apartado dedicado al control de proyectores en LAN.

```
{  
    "Values": [  
        {  
            "modulo": "Mural",  
            "proyector": "NEC-UM330X/UM280X",  
            "pc": "pcMural",  
            "puerto": "2"  
        },  
        {  
            "modulo": "ECA1",  
            "proyector": "NEC-LT280/LT380",  
            "pc": "pcECA1",  
            "puerto": "3"  
        }  
    ]  
}
```

Figura 5.13. Contenido del fichero *valuemodules.json* en el que se almacenan las configuraciones para cada módulo en un formato *nombre:valor*.

En la línea 19 se hace la llamada a la función *turnOnPc()*, la cual se encarga del encendido de la computadora. Esta contiene las siguientes instrucciones:

```
1 function turnOnPc (pcName) {  
2     //1 Tomar dirección física correspondiente a la PC del fichero PcAulaFuturo  
3     //2 Crear paquete mágico haciendo uso del módulo "dgram"  
4     //3 Envío del paquete por la red local por transmisión broadcasting  
5     //4 Retorno de función  
6 }
```

Esta función recibe como parámetro el nombre de la computadora a encender (*pcName*), después toma su respectiva dirección física almacenada en el fichero de nombre *PcAulaFuturo.json*. Posteriormente crea el paquete mágico (datagrama UDP) a partir de su dirección física y se almacena en un *Buffer*; se crea un *socket* de tipo UDP y se ata a un puerto local aleatorio, después se especifica el puerto y la dirección IP destino y finalmente éste se envía a todos los dispositivos conectados a la red local por transmisión de difusión *broadcasting*. El contenido completo de esta función se muestra en la sección de anexos llamada “*Algoritmo implementado para el control de computadoras en LAN*”.

Por último se retorna de la función y seguido el servidor maestro retorna una respuesta al usuario indicando si el encendió del módulo fue correcto y está en proceso o si hubo algún de tipo error en el proceso.

5. Desarrollo de la solución

- **Apagado de computadoras**

Todas las computadoras dadas de alta en el sistema contienen un “pequeño” servidor al que llamamos servidor esclavo. Se hace referencia a que es pequeño ya que en comparación al que se ejecuta en el servidor maestro, el cual realiza múltiple tareas y ejecuta varias rutas de código, este solo ejecuta tres tareas o rutas: una ruta para su apagado, una para la ejecución de archivos y otra para saber el estado de la PC (encendida o apagada). Este servidor está instalado en cada una de las máquinas y se ejecuta en automático desde la línea de comandos (con archivos tipo *batch*) cada vez que esta enciende. El puerto en donde permite conexiones o peticiones por parte de los clientes es en el 3001; que en este caso el cliente siempre será el servidor maestro solicitando la ejecución de alguna de estas tres tareas.

Cuando el usuario requiere apagar un módulo, por ejemplo el Mural Interactivo que se encendió anteriormente, debe seleccionarlo y presionar el botón “*Apagar*” desde la interfaz Web. De la misma forma que para el proceso de encendido de un módulo se envía una petición http servidor maestro de la forma: <http://192.168.11.100:3000/turnOffModule?Module=Mural>; ahora la ruta a ejecutar en el servidor cambia a “*/turnOffModule*”. El código a implementar es muy similar al que se aplica para el proceso de encendido de un módulo, la diferencia radica en que ahora se le indica al servidor esclavo en el sistema embebido que se requiere el apagado del proyector y por otro lado el llamado a la función de la línea 19, que en este caso se llama a la función de nombre *turnOffPC()*, la cual contiene las siguientes instrucciones:

```
1 function TurnOffPC(pcName) {
2     //1 Tomar dirección IP correspondiente a la PC del fichero PcaulaFuturo
3     //2 Envío de petición HTTP con AJAX a servidor esclavo de la PC a apagar
4     //3 Retorno de función
5 }
```

Esta función recibe como parámetro el nombre la computadora a apagar (pcName), después se toma su respectiva dirección de red o IP almacenada en el fichero de nombre *PcaulaFuturo.json*. Posteriormente, el servidor maestro envía una petición HTTP al servidor esclavo al puerto 3001 y a la ruta de nombre */shutdownMyself*. El contenido completo de esta función se muestra en la sección de anexos llamada “*Algoritmo implementado para el control de una computadora en LAN*”.

5. Desarrollo de la solución

Supongamos que la dirección IP de la computadora asignada al Muro Interactivo es *192.168.11.90*, la petición HTTP enviada por el servidor maestro al servidor esclavo sería de la forma: <http://192.168.11.90:3001/shutdownMyself>; ejecutándose en este la siguiente ruta:

```
1 // Módulo de Node que permite el uso de la línea de comandos del S.O.
2 var cmd = require('node-cmd');
3
4 // Ruta "shutdownMyself"
5 exports.shutdownMyself = function(req, res) {
6
7     //Ejecución del comando shutdown
8     cmd.get('shutdown -s -f -t 0 -m \\\\'+addresses[0]);
9
10    // Retorno de respuesta de petición al servidor maestro
11    res.end("Apagado Ok");
12}
```

En la línea 2 de esta ruta se declara un módulo de Node llamado *node-cmd*, el cual permite hacer uso de la línea de comandos (terminal del sistema) de la computadora y así poder ejecutar en ella un comando llamado *shutdown* [60]. Este permite el apagado, reinicio o hibernación de una máquina desde su línea de comandos haciendo uso de su dirección IP o nombre de *host*. Este comando en específico solo está disponible para máquinas con sistema operativo *Windows*, aunque también existe un comando de apagado equivalente en otros sistemas.

El comando *shutdown* contiene parámetros que permiten especificar el tipo de apagado de la computadora. La configuración usada en este caso se muestra en la *figura 5.14*. De izquierda a derecha, lo primero que se especifica es el nombre del comando “*shutdown*”, después la acción que se requiere ejecute el comando, en este caso ”*s*“ indica el apagado de la máquina, el parámetro ”*f*“ indica un apagado forzado, el parámetro ”*t 0*“ indica un apagado al cabo de cero segundos, el parámetro ”*m*“ indica que a continuación se especifica la máquina destino y por último ”\direcciónIP“ se indica la dirección IP de la computadora destino.

Dicho de otra forma, esta línea indica: ejecuta el comando *shutdown* para el apagado forzado al tiempo cero de la máquina con dirección IP *192.168.11.90*.

5. Desarrollo de la solución

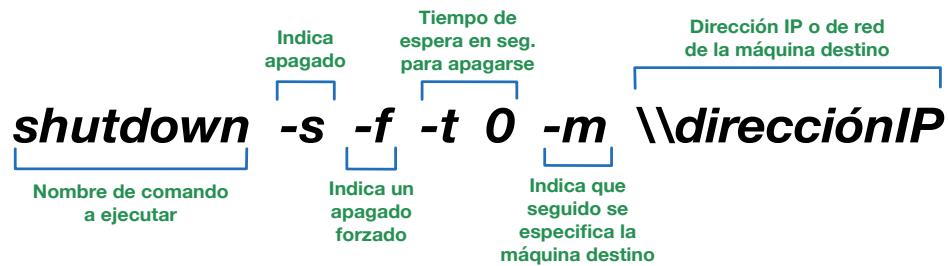


Figura 5.14. Uso del comando *shutdown* con sus parámetros de especificación para indicar el tipo de apagado una computadora.

La ejecución del comando descrito anteriormente se lleva acabo en la línea 8 de la ruta `/shutdown`:

```
8     cmd.get('shutdown -s -f -t 0 -m \\\\'+addresses[0]);
```

Donde `addresses[0]` contiene la dirección IP de la máquina. Una vez ejecutado este comando, la máquina comenzará el proceso para su apagado. Por otro lado, el servidor esclavo retorna al servidor maestro el resultado de la operación, donde este último retorna una respuesta al usuario indicando si el apagado del módulo fue correcto o si hubo algún tipo de error en el proceso.

- **Ejecución/Proyección de archivos**

Como se vio en la sección de desarrollo de Font-end en *Proyección de archivos*, el usuario es capaz desde la interfaz Web de proyectar un archivo en alguna de las ZIT o módulos. Para esto se debe seleccionar alguno de los archivos disponibles, seleccionar el módulo de proyección y presionar el botón “*Proyectar*”.

Como modo de ejemplo, supongamos que el usuario/profesor requiere proyectar una presentación en *PowerPoint* de nombre *InternetDeLasCosas.pptx* en el Muro Interactivo; una vez que ha seleccionado ambos y dé la orden de proyección, el cliente envía al servidor maestro una petición HTTP con AJAX de la forma:

<http://192.168.11.100:3000/startFile?fileName=InternetDeLasCosas.pptx&modulo=Mural>

Esta petición es localizada en el servidor en la siguiente ruta:

5. Desarrollo de la solución

```
1 exports.startFile = function(req1, res1) {  
2   //1. Tomar de la URL el nombre del archivo y el módulo de proyección  
3   //2. Tomar del fichero valueModules la PC atada/ligada al módulo  
4   //3. Tomar del fichero PxAulaFuturo la dirección IP correspondiente a la PC  
5   //4. Llamada a función sendCmdToStartFile(dirIP, fileName);  
6 };
```

En esta ruta se realizan cuatro tareas fundamentales: en la primera se toman de la URL el nombre del archivo y el módulo de proyección, luego del fichero *valueModules.json* se toma la computadora ligada a ese módulo, después se toma del fichero *PxAulaFuturo.json* la dirección IP para esa computadora y por último se llama a la función de nombre *sendCmdToStartFile()*, la cual recibe dos parámetros, la dirección IP (*dirIP*) de la computadora del módulo y el nombre del archivo a proyectar o ejecutar en ella (*fileName*). Siguiendo con el ejemplo, supongamos que la dirección IP de la computadora Mural es *192.168.11.90*, por lo que la llamada a la función sería: *sendCmdToStartFile (192.168.11.90, InternetDeLasCosas.pptx)*. A continuación se muestra el contenido de esta función:

```
1 function sendCmdToStartFile(dirIP, fileName) {  
2   // Indicar opciones de la petición  
3   var options = {  
4     hostname: dirIP,    //dirección IP de la PC  
5     port: 3001,          //Puerto donde recibe conexiones  
6     path: "/start?data=\\""+hostNameServer+"\\"+programToShare+"\\"+fileName  
7   };  
8  
9   ***//Envío de petición al servidor esclavo  
10  
11  req.end(); //Terminar petición  
12 };//Cierre de función
```

En esta función el servidor maestro envía una petición HTTP al servidor esclavo que se ejecuta en la computadora del módulo para dar la orden de ejecución del archivo. De la línea 3 a la 7 se indican las opciones de la petición, esto es, se da la dirección IP del servidor esclavo, el puerto donde recibe conexiones y la ruta de acceso. Esta ruta de la línea 6 contiene el parámetro *data* cuyo valor es la ruta de acceso al directorio o carpeta donde se encuentra almacenado el archivo a ejecutar. Todos los archivos que son subidos por el usuario desde la interfaz Web se almacenan en el servidor maestro en una carpeta compartida en red de nombre *programsToShare*. Esta puede ser

5. Desarrollo de la solución

vista por todos los dispositivos conectados a la misma red a la que esté conectado este servidor y por lo tanto pueden hacer uso de los archivos que se encuentran en ella.

Para el caso del archivo de nombre *InternetDeLasCosas.pptx*, la ruta de acceso a este sería: *hostNameServer*\programsToShare\InternetDeLasCosas.pptx; donde la variable *hostNameServer* contiene el nombre de la PC donde se ejecuta el servidor maestro.

Por otro lado, la ruta “/start” que se localiza en el servidor esclavo contiene las siguientes instrucciones:

```
1 //Modulo de Node para la ejecución de procesos/archivos
2 var exec = require('child_process').execFile;
3 exports.start = function(req, res) {
4     // Tomar de la URL la variable data, la cual contiene la ruta de acceso
5         al archivo dentro de la carpeta programsToShare
6     var file = req.query.data;
7     // Ejecución de un proceso, en este caso del archivo en file
8     execFile(file,function callback(error, stdout, stderr){
9         console.log("stdout") //Proceso terminado
10    });
11    res.end(); //Cierre de respuesta
12});
```

En la línea 6 se toma la ruta de acceso al archivo y se almacena en la variable de nombre *file*, en la línea 8 se hace uso del método *execFile()* el cual permite la ejecución de un archivo haciendo uso del módulo *child_process* de la API de Node. Este método recibe dos parámetros principalmente, el primero de ellos es el archivo o la ruta del archivo a ejecutar y el segundo es una función de *callback*.

Una vez ejecutada esta línea se abrirá el archivo indicado en la computadora indicada por el usuario. Para el ejemplo descrito anteriormente, se ejecuta o proyecta en el Mural Interactivo la presentación de nombre *InternetDeLasCosas.pptx*.

5. Desarrollo de la solución

5.3.2 Control de proyectores en LAN

En esta sección del capítulo se muestra la parte correspondiente al control de proyectores en una red de área local. Antes de pasar al proceso de control de ellos desde la interfaz Web, se mencionan algunos detalles y configuraciones que se requieren tener en cuenta con el fin de comprender mejor su funcionamiento dentro del sistema.

Todos los proyectores del Aula del Futuro y la gran mayoría de los que se encuentran actualmente en el mercado, cuentan con un puerto serial macho DB9 para su control. En el manual técnico de cada proyector el fabricante otorga una lista de comandos y un conjunto de especificaciones de comunicación que permiten su control desde un dispositivo externo. Un ejemplo de ello se muestra en la *figura 5.15*.

6 PC Control Codes and Cable Connection							
PC Control Codes							
Function	Code Data						
POWER ON	02H	00H	00H	00H	00H	02H	
POWER OFF	02H	01H	00H	00H	00H	03H	
INPUT SELECT COMPUTER	02H	03H	00H	00H	02H	01H	01H 09H
INPUT SELECT HDMI1	02H	03H	00H	00H	02H	01H	1AH 22H
INPUT SELECT HDMI2	02H	03H	00H	00H	02H	01H	1BH 23H
INPUT SELECT VIDEO	02H	03H	00H	00H	02H	01H	06H 0EH
INPUT SELECT S-VIDEO	02H	03H	00H	00H	02H	01H	08H 13H
INPUT SELECT VIEWER	02H	03H	00H	00H	02H	01H	1FH 27H
INPUT SELECT NETWORK	02H	03H	00H	00H	02H	01H	20H 28H
INPUT SELECT USB DISPLAY	02H	03H	00H	00H	02H	01H	22H 2AH
PICTURE MUTE ON	02H	10H	00H	00H	00H	12H	
PICTURE MUTE OFF	02H	11H	00H	00H	00H	13H	
SOUND MUTE ON	02H	12H	00H	00H	00H	14H	
SOUND MUTE OFF	02H	13H	00H	00H	00H	15H	

NOTE: Contact your local dealer for a full list of the PC Control Codes if needed.

Cable Connection

Communication Protocol

Baud rate 38400 bps
Data length 8 bits
Parity No parity
Stop bit One bit
X on/off None
Communications procedure Full duplex

Figura 5.15. Lista de comandos de control en hexadecimal y especificaciones de comunicación para proyector de la marca *NEC-UM330X/UM280X* [63].

En la parte superior se muestra una lista de comandos de control en hexadecimal para el proyector de la marca *NEC* modelo *UM330X/UM280X*. Se marca en color verde los comandos para su encendido y en color azul los correspondientes para su apagado. En la parte inferior se definen las especificaciones de comunicación, esto es, se especifica que la velocidad de transferencia (*Baud Rate*) es de 38400 bps, que utiliza ocho bits de datos, no paridad, un bit de parada y soporta una comunicación *full duplex*.

5. Desarrollo de la solución

El sistema embebido (SE) utilizado como servidor esclavo contiene ocho manejadores UART con los cuales es posible establecer una comunicación punto a punto entre estos y los proyectores.

Dentro de una caja de acrílico se montó el SE y se perforó de modo de introducir ocho puertos seriales (conectores macho DB9), un puerto Ethernet (RJ45) y un puerto Micro-USB. En el diagrama de la *figura 5.16* se muestran las dimensiones y el contenido de este dispositivo.

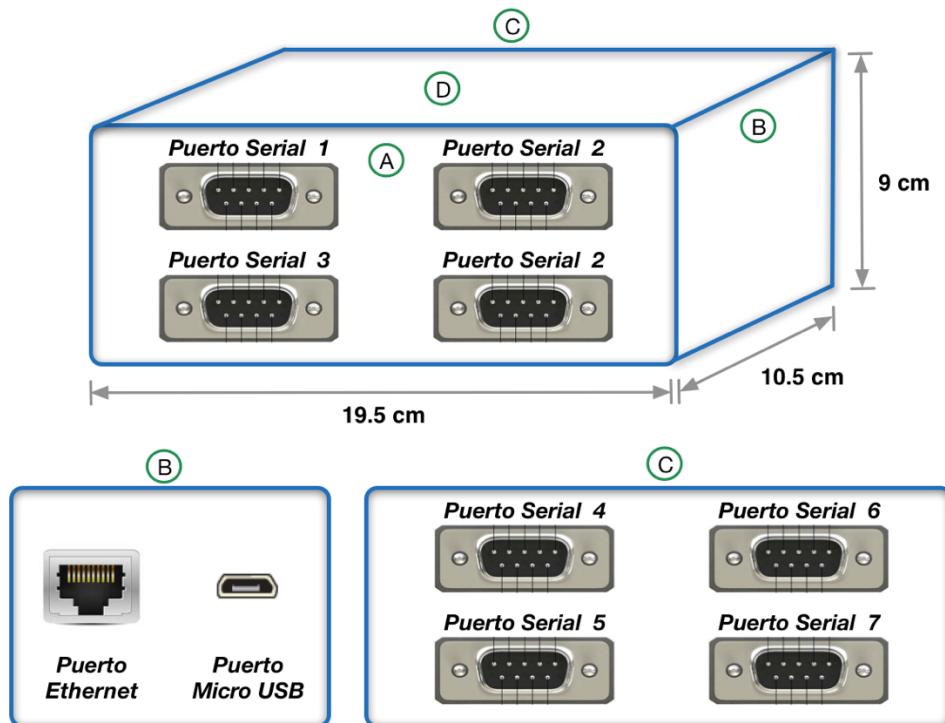


Figura 5.16. Dispositivo que contiene al SE, ocho puertos seriales (DB9), un puerto de red Ethernet (RJ45) y un puerto de Micro-USB.

En la cara **A** del dispositivo se muestran cuatro de los conectores seriales de número 1, 2, 2, 3. Mientras que en la cara opuesta **C** se muestran los conectores de número 4, 5, 6, 7. El medio de transmisión a utilizar para la comunicación entre estos puertos y los proyectores es un cable RS232 hembra-hembra de modem nulo (*Null-Modem*) y de longitud de tres metros cada uno.

Se presentan dos puertos seriales número 2 de modo de transmitir la misma señal y poder controlar dos proyectores de la misma marca al mismo tiempo. Tal y como lo requiere para el Mural Interactivo del aula, ya que actualmente se compone de dos proyectores de la marca *NEC*.

5. Desarrollo de la solución

En la cara **B** del dispositivo se encuentran los puertos Micro-USB y Ethernet. El primero de ellos es usado para alimentar y programar al sistema embebido y el segundo para enlazarlo a la red local y poder usarlo como un servidor HTTP esclavo.

La parte correspondiente a la cara **D** del dispositivo se muestra en el diagrama de la *figura 5.17*.

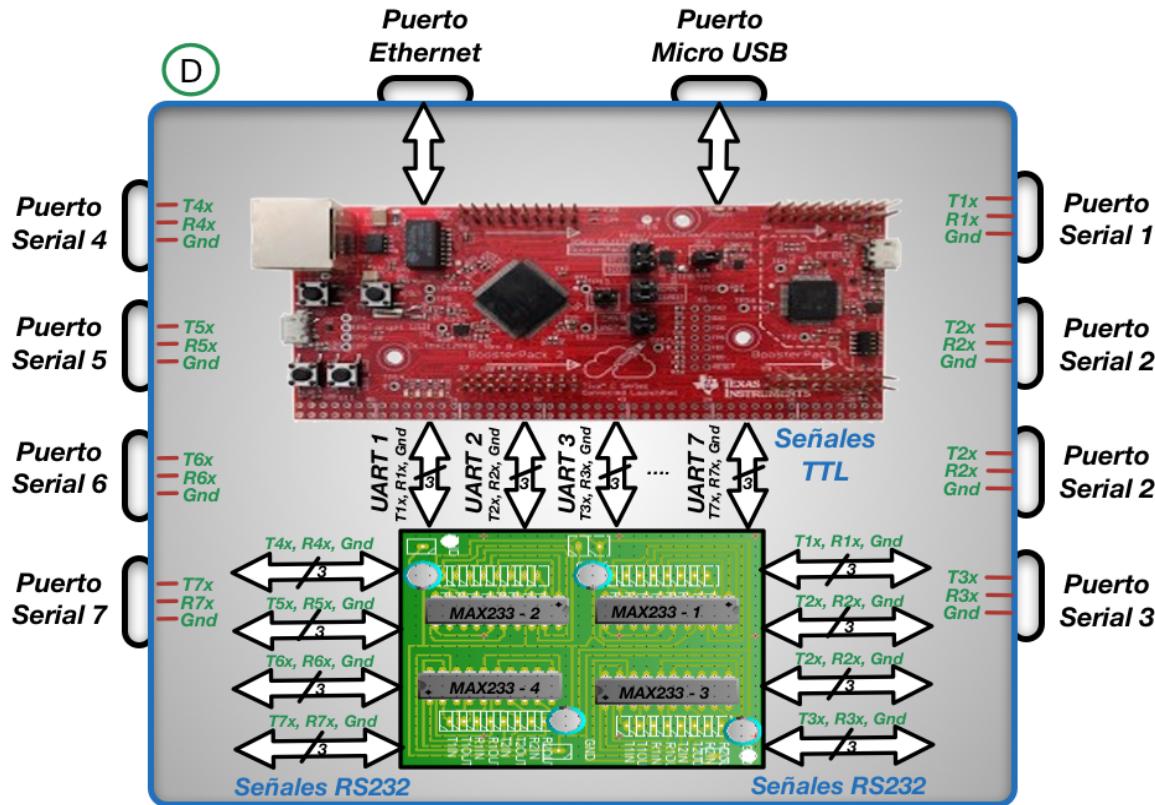


Figura 5.17. Vista aérea del dispositivo en el que muestra su contenido: un sistema embebido, una placa con cuatro C.I. MAX233 y las conexiones entre estos y puertos seriales.

Esta representa una vista aérea en la que se muestra el contenido del dispositivo. En él se encuentra el sistema embebido y una placa con cuatro circuitos integrados MAX233 usada para la conversión de señales TTL a RS232 y viceversa. Tanto el SE como esta placa de C.I. contienen pines o *headers* macho, por lo que las conexiones entre estos se realizó mediante conectores o cables *jumpers* hembra-hembra. En la parte interna de cada puerto serial se soldaron tres cables, que corresponde al transmisor (Tx), al receptor (Rx) y a tierra (GND), en el otro extremo del cable el conector es de tipo jumper hembra, para su conexión con la placa de los C.I.

5. Desarrollo de la solución

En la *figura 5.18* se muestra la placa con los cuatro *drivers* MAX233. Para su construcción, en un principio se realizó la simulación del circuito con ayuda del programa *LiveWire* con el fin de comprobar su correcto funcionamiento. Posteriormente, en el programa *PCB Wizard* se realizó el diseño del circuito impreso. Finalmente, se pasó a la etapa de construcción, en la que se realizó el planchado, perforación y soldado de componentes en una placa fenólica de cobre.

Cada MAX233 consta de 20 pines y permiten, para cierta configuración de conexión, que para dos señales de entrada TTL (*T1 IN* y *T2 IN*) las salidas las convierte a RS232 (*R1 OUT* y *R2 OUT*) y para dos entradas RS232 (*R1 IN* y *R2 IN*) las salidas las convierte en TTL (*T1 OUT* y *T2 OUT*). Por lo que por cada dos manejadores UART del SE, se utiliza uno de estos circuitos lógicos. En la hoja de datos de este C.I. se muestran las conexiones y configuraciones correspondientes. [61].

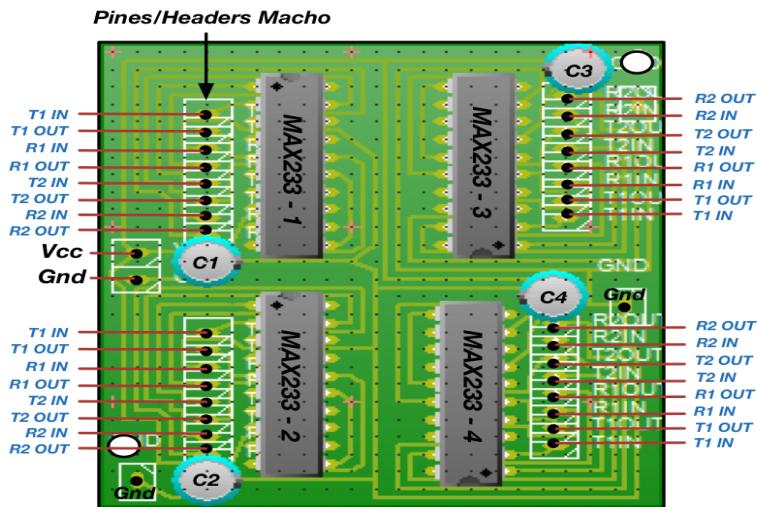


Figura 5.18. Placa con cuatro C.I. MAX233 con ocho pines de conexión por cada circuito.

Pasando a la parte de su control desde la interfaz Web, cada vez que el usuario (profesor) requiere encender o apagar un módulo, éste envía una petición HTTP al servidor maestro que es localizada en la ruta de nombre “/turnOnModule” o “/turnOffModule”, respectivamente. En esta ruta, como se vio en la sección para el control de computadoras, se toman los valores correspondientes para cada módulo del fichero *valueModules.json*, esto es, se toma el proyector, puerto serial y la computadora. Posteriormente, del fichero *projectorAulaFuturo.json* se toman los comandos de control para el proyector, en la *figura 5.19* se muestra una parte del contenido de este fichero. En este se incluyen los comandos para el encendido (*onCmd*), apagado (*offCmd*), la velocidad de

5. Desarrollo de la solución

transmisión (*baudRate*), así como los comandos que retorna el proyector en respuesta a los comandos de encendido y apagado (*onAckCmd* y *offAckCmd*, respectivamente) para los proyectores de la marca *ViewSonic* y *NEC*.



```
{  
    "PROJECTORS": [  
        {  
            "projectorName": "ViewSonic-PJD6251",  
            "onCmd": "BE,EF,10,05,00,C6,FF,11,11,01,00,01,00",  
            "onAckCmd": "06",  
            "offCmd": "BE,EF,03,06,00,DC,DB,69,00,00,00,00,00",  
            "offAckCmd": "06",  
            "baudRate": "19200"  
        },  
        {  
            "projectorName": "NEC-UM330X/UM280X",  
            "onCmd": "02,00,00,00,00,02",  
            "onAckCmd": "22,00",  
            "offCmd": "02,01,00,00,00,03",  
            "offAckCmd": "22,01",  
            "baudRate": "38400"  
        }  
    ]  
}
```

Figura 5.19. Contenido del fichero *projectorAulaFuturo.json* en el que se almacenan los comandos de control para cada proyector.

Poniendo como ejemplo que se requiere el encendido del módulo o ZIT Mural Interactivo, el cual tiene asignado la computadora de nombre *pcMural*, un proyector de la marca *NEC-UM330X/UM280X* y el puerto serial número 2. Una vez tomados los valores y comandos correspondientes, el servidor maestro comienza el proceso de comunicación enviando una petición HTTP al servidor esclavo en el sistema embebido de la forma:

http://192.168.11.166/puerto2?cmd=020000000002&baudRate=38400+&Ack=2200*

En donde la dirección IP *192.168.11.166* correspondiente a la del sistema embebido, se programó de tal forma que ésta fuera estática o fija y así contenga siempre el mismo valor.

Las rutas de acceso a este servidor esclavo están determinadas por el número de puerto serial por el cual se requiere el envío de comandos, por lo que contiene siete rutas que van desde “/puerto1” hasta “/puerto7”. Para el caso del ejemplo la ruta a ejecutar es: “/puerto2”.

Los parámetros a enviar en la URL siempre son tres: uno es “*cmd*” el cual contiene los comandos a enviar por el puerto serial del SE hacia el proyector, el otro es “*baudRate*” que es la velocidad de transmisión que maneja y el último es “*Ack*” que es la respuesta de retorno del proyector al comando enviado, indicando si este fue correctamente procesado y está en ejecución.

A continuación se muestra la ruta correspondiente a “/puerto2” en el servidor esclavo del SE:

5. Desarrollo de la solución

```
1 else if(ustrncmp(pcName, "/puerto2?", 8) == 0){  
2  
3 //1. Habilitar interrupción de UART2  
4 //2. Llamada a función getParamURL() para tomar los parámetros de la URL  
5 //3. Configurar modulo UART2 mediante la función UARTConfigSetExpClk()  
6 //4. Inicializar FIFO mediante función initFifo() y la variable Flag a cero  
7 //5. Envío de comandos de control al proyector con función UARTCharPut()  
8 //6. Delay a la espera de que ocurra la ISR de UART Rx (Ack de proyector)  
9 //7. Llamada a función checkProjectorResponse() para checar el tipo de respuesta  
10 //8. Deshabilitar interrupción de UART2  
11 //9. Retorno de respuesta al servidor maestro  
12  
13 }
```

En cada una de las siete rutas del SE se ejecutan en general nueve tareas fundamentales: la primera de ellas es habilitar la interrupción del UART correspondiente a ese puerto, en este caso es el 2. La segunda tarea corresponde a la llamada a función de nombre *getParamURL()* en la cual se extraen los parámetros *cmd*, *baudRate* y *Ack* de la URL. En la tercera se establece la configuración del módulo UART mediante la función *UARTConfigSetExpClk()*, en esta función se indica el número de UART a configurar, se pasa la frecuencia de reloj del sistema y se indica la velocidad de transmisión (*baudRate*), ocho bits de datos, un bit de parada y no paridad. En la cuarta tarea se inicializa un *Buffer* (arreglo) tipo FIFO mediante la llamada a función de nombre *initFifo()* y la variable global de nombre *Flag* se establece a cero, ambas variables son usadas en la rutina de interrupción y sirven para la sincronización de hilos; en el arreglo FIFO se van almacenar los comandos de respuesta del proyector y la variable *Flag* cambia su valor a uno para indicar que se ingresó a la interrupción del UART. En la quinta tarea se lleva acabo el envío de comandos al proyector por el puerto especificado mediante la función *UARTCharput()*; para el caso del ejemplo mencionado, se transmitirán por el UART2 (T2x) los comandos en serie: 02, 00, 00, 00, 00 02, por los pines número 7 del puerto A (PA7) y el número 5 del puerto D (PD5), los cuales están conectados a la placa de C.I. MAX233 para la conversión de señales TTL a RS232 y posteriormente transmitir por ambos puertos seriales (número 2) estos comandos a los proyectores. En la sexta tarea se realiza un pequeño *delay* (del orden de milisegundos) con el fin de esperar la respuesta del proyector a los comandos antes enviados, de recibirse se ejecuta o dispara la interrupción del UART, en la rutina *UARTIntHandler()*; en ella se toman los comandos recibidos y se almacenan en el FIFO. En la séptima tarea se hace la llamada a la función de nombre

5. Desarrollo de la solución

checkProjectorResponse(), en la cual se verifica que la respuesta del proyector coincida (que contenga el mismo valor) con el parámetro *Ack* tomado inicialmente de la URL, de ser el caso se indica que el comando inicialmente enviado fue correctamente procesado por el proyector, de no coincidir se indica que hubo algún error en el proceso. En la octava tarea se deshabilita la interrupción del UART, esto con el fin de que solo se utilice en el contexto de transmisión y recepción de comandos entre el S.E. y los proyectores y así evitar entradas erróneas por otros contextos. En la novena y última tarea se retorna una respuesta al servidor maestro, indicándole el resultado de la operación. Si el retorno de respuesta fue correcto, esto es, si se logró encender o apagar al proyector, la tarea siguiente en el servidor maestro consiste en el encendido o apagado de la computadora del módulo o ZIT correspondiente; de lo contrario, ya no se ejecuta la orden de encendido de la computadora. Ambas posibles respuestas, se retornan eventualmente al usuario indicándole si la tarea que solicitó en un inicio se ejecutó correctamente o si hubo algún tipo de error en el proceso.

El contenido completo de esta función se muestra en la sección de anexos llamada “*Algoritmo implementado para el control de proyectores en LAN*”.

CAPÍTULO 6

PRUEBAS CON USUARIOS Y ANÁLISIS DE RESULTADOS

Para identificar posibles problemas de usabilidad así como probar la funcionalidad del sistema de control desarrollado, se realizaron evaluaciones con usuarios dentro del Aula del Futuro (AF). Estas evaluaciones se aplicaron a un total de cinco personas, de las cuales tres fueron con profesores del AF y las otras dos fueron con profesores externos.

Todas pruebas se realizaron desde la interfaz o aplicación Web desarrollada, que para este caso, se corrió/ejecutó desde el navegador Web *Google Chrome* en un dispositivo móvil (*Smartphone* marca Samsung). En cada sesión se realizó una grabación de audio y video (hecha desde el mismo *Smartphone*) para captar todas las interacciones entre el usuario y el sistema. Esto con el fin de analizar con detalle esta interacción y así detectar problemas de usabilidad con mayor facilidad.

Las pruebas se realizaron en dos sesiones (una por día), en la primera se realizó la prueba con dos profesores y en la segunda con tres. Al inicio de cada sesión, el moderador daba a conocer al usuario el propósito de la reunión al leerle el **protocolo de bienvenida** (figura 6.1). Posteriormente, se pasaba a la parte de pruebas, en donde el moderador daba instrucciones (con el documento llamado **actividades de la evaluación**) a cada profesor con el fin de que completaran las tareas o actividades dentro del sistema. Por último, con el fin de evaluar la usabilidad del sistema, a cada usuario se le aplicó un **cuestionario de salida** (usabilidad).

En la parte de anexos “*Documentos para pruebas con usuarios*” se muestran todos los documentos elaborados y utilizados para las pruebas con usuarios.



Figura 6.1. Moderador aplicando pruebas de usabilidad a profesores dentro del Aula del Futuro.

6. Pruebas con usuarios y análisis de resultados

Por otro lado, las tareas o actividades que realizaron los profesores en estas evaluaciones corresponden un total de cinco, las cuales fueron seleccionadas con el fin de probar las grandes funcionalidades del sistema. Estas son:

- I.** Descripción de la página principal.
- II.** Encender ZIT: Mural Interactivo y ECA 1.
- III.** Subir un archivo y proyectarlo en una ZIT: Mural Interactivo.
- IV.** Apagar ZIT: Mural Interactivo y ECA 1.
- V.** Programar el encendido de una ZIT (Mural Interactivo) a una hora y fecha indicada.

A continuación se describen en qué consistieron cada una de estas tareas así como un análisis de los resultados obtenidos, mencionando observaciones y recomendaciones, con el fin de realizar mejoras en el sistema.

I. Descripción de la página principal

Esta primera prueba consistió en que el profesor diera una descripción de cada uno de los elementos que percibe en la página principal y que indicara para qué cree que sirven (*figura 6.2*).

Observaciones:

Para las descripciones de los profesores que imparten clases en el AF se observó que, sin necesidad de dar algún tipo de explicación previa acerca del funcionamiento del Mural Interactivo o ECA, pudieron entender, en general, qué tipo de funciones les brindaba el sistema de control. Por otro lado, para los otros dos profesores externos al AF, sí fue necesario una explicación previa de cómo funcionaban estas dos tecnologías o ZIT.

En general, todos los profesores lograron comprender que desde la interfaz Web podían controlar o acceder a las distintas ZIT, que podían subir y proyectar archivos así como programar el encendido de alguna de ellas. Además, pudieron relacionar que la ubicación de cada botón de la interfaz correspondía a la ubicación espacial de cada ZIT en el aula.

Por último, se observó que no supieron explicar qué tipo de configuraciones son las que podrían realizar desde la interfaz Web con el botón “Config” que aparece en la parte

6. Pruebas con usuarios y análisis de resultados

superior derecha. Ni tampoco qué tipo de información se mostraba o desplegaba al presionar el botón “Info” de la parte superior izquierda.

Recomendaciones:

En la gran mayoría de la pruebas, los profesores no percibían que los botones en negro (en el centro de la página de inicio) eran de tipo selectivo. Por lo que se recomienda cambiar la tonalidad del botón o bien implementar otra forma de selección.

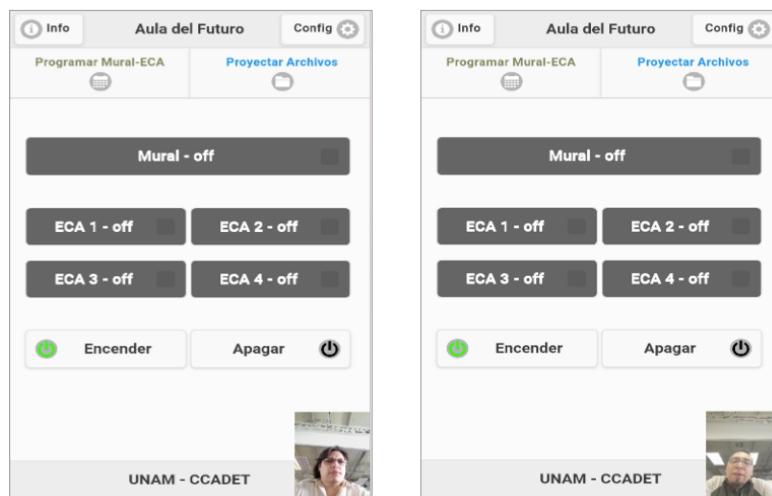


Figura 6.2. Profesor describiendo los elementos que percibe en la página de inicio.

II. Encender Mural Interactivo y ECA 1

La segunda tarea consintió en que el profesor realizara el encendido de dos ZIT: Mural Interactivo y ECA 1 (*figura 6.3*).

Observaciones:

Todos los profesores fueron capaces de completar satisfactoriamente esta tarea en el tiempo estimado. Por un lado pudieron darse cuenta que después de seleccionar el Mural y ECA1 y presionar el botón “Encender”, se mostraba un ícono de “cargando” en cada uno de los botones de las ZIT seleccionadas, por lo que concluyeron que el encendido de las dos ZIT (proyectores y PC de cada una) se encontraba en proceso. También pudieron darse cuenta de ello al ver desde su lugar (desde donde relazaron las pruebas) que las zonas de

6. Pruebas con usuarios y análisis de resultados

trabajo comenzaban a proyectar una imagen. Por otro lado, una vez que se encendieron las dos ZIT percibieron el cambio de color de negro (*off*) a verde (*on*) en los botones de selección.

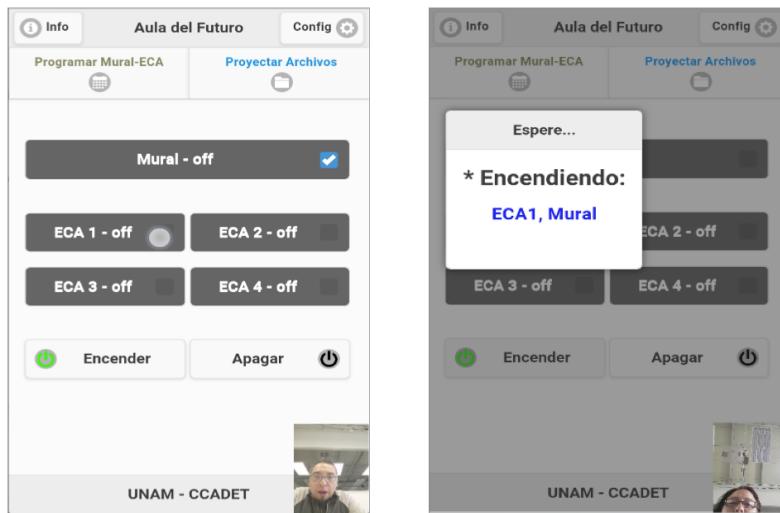


Figura 6.3. Profesores realizando la tarea de encendido del Mural y ECA 1.

III. Subir un archivo y proyectarlo en el Mural Interactivo

La tercera tarea consistió en que el profesor se dirigiera a la interfaz (pantalla) de proyección de archivos, posteriormente se le pidió que subiera un archivo (una presentación en *PowerPoint*) al servidor y que lo proyectara/ejecutara en el Mural Interactivo (*figura 6.4*). Por último debía eliminar el archivo subido anteriormente.

Observaciones:

Esta tarea pudo ser completada por todos los profesores en el tiempo estimado (a excepción de uno de ellos que tardó más de lo supuesto). Todos lograron dirigirse a la sección (pantalla) de proyección de archivos sin dificultades y después subir un archivo al servidor de forma correcta. En la siguiente parte de la tarea, la cual consistió en la proyección del archivo recién subido en el Mural Interactivo, se observaron dificultades para llevarla a cabo. Esto debido a que daban la orden de proyección en el Mural (presionando el botón “Proyectar” de la parte inferior) sin haber seleccionado el archivo,

6. Pruebas con usuarios y análisis de resultados

pero la interfaz les respondía con un cuadro de dialogo indicándoles que era necesario la selección de un archivo y una ZIT para poder proyectarlo; después de ello, todos los profesores pudieron completar esta parte de la tarea correctamente.

Para la última parte de esta tarea, la cual consistió en eliminar el archivo subido anteriormente, todos pudieron realizarla correctamente.

Recomendaciones:

A la hora de seleccionar una ZIT para la proyección del archivo, todos los profesores comentaron que esperaban que se mostraran como opciones de selección solo aquellas zonas que estuvieran encendidas (actualmente se muestra como opción de selección todas las ZIT, aunque se encuentre inactiva/apagada), por lo que se recomienda mostrar como opción solo las ZIT que estén encendidas y listas para proyectar o ejecutar archivos. Por otra parte, se recomienda que cuando un archivo sea subido al servidor, este se encuentre seleccionado en automático en el contenedor de archivos y esté listo para proyectarse en alguna ZIT.

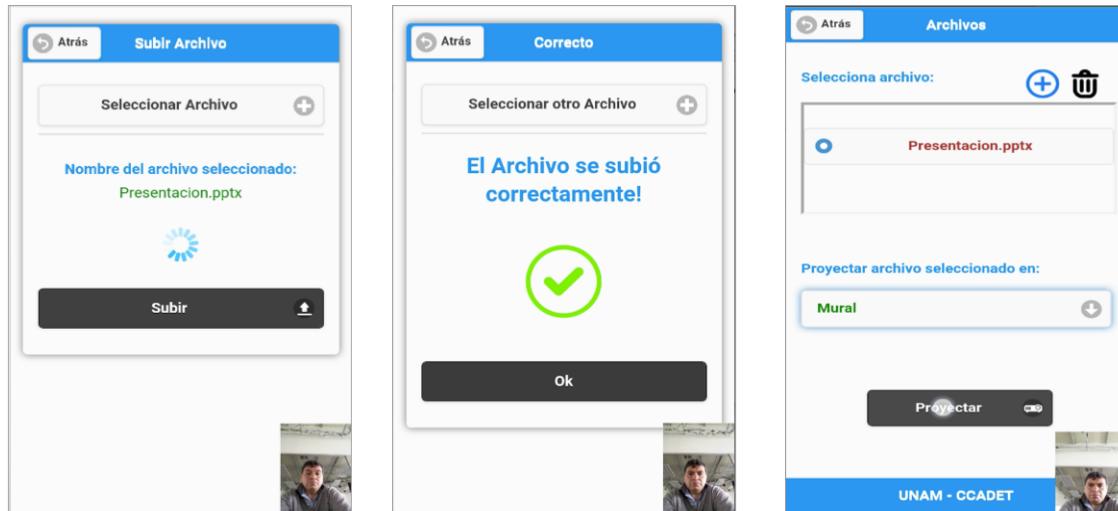


Figura 6.4. Profesor realizando la tarea de subir un archivo al servidor (llamado *presentación.pptx*) y proyectarlo en el Mural Interactivo.

6. Pruebas con usuarios y análisis de resultados

IV. Apagar Mural Interactivo y ECA 1

La cuarta tarea consistió en que el profesor realizara el apagado de las ZIT correspondientes al Mural Interactivo y ECA 1 (*figura 6.5*), las cuales había encendido anteriormente en la tarea 2.

Observaciones:

Todos los profesores fueron capaces de completar satisfactoriamente esta tarea en el tiempo estimado. Primero, lograron dirigirse desde la interfaz de proyección de archivos a la página de inicio. Una vez ahí, seleccionaron el Mural y ECA1 y presionaron el botón “Apagar” de la parte inferior, esto dio como resultado el apagado de ambas zonas y donde el profesor percibió los cambios de color de los botones de verde (*on*) a negro (*off*) en la interfaz. También pudieron darse cuenta de ello al ver desde su lugar que las zonas de trabajo dejaban de proyectar una imagen.

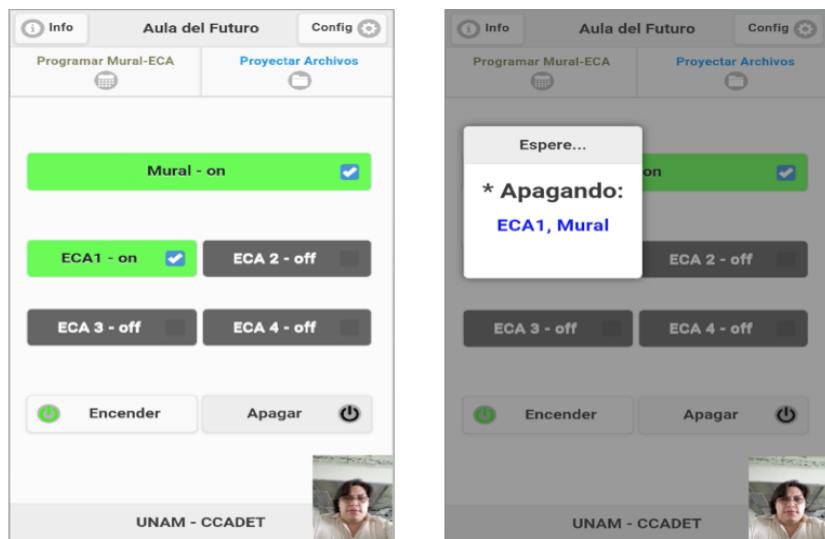


Figura 6.5. Profesor realizando la tarea de apagado del Mural y ECA 1.

V. Programar el encendido del Mural Interactivo a la hora y fecha indicada

La quinta y última tarea consistió en que el usuario debía realizar la programación del encendido de una ZIT (Mural Interactivo) a una hora y fecha indicada (*figura 6.6*).

6. Pruebas con usuarios y análisis de resultados

Observaciones:

Todos los profesores fueron capaces de completar esta tarea de forma correcta, sin embargo, dos de los cinco lo hicieron fuera del tiempo estimado.

Primero, todos lograron dirigirse sin problemas desde la interfaz de inicio hasta la interfaz de programación de una ZIT (Mural-ECA). Una vez ahí se les indicó que debían realizar una nueva programación, en la que ingresarán un Nombre, el Módulo o ZIT a programar, una Fecha y un Horario. Para fines prácticos y que pudieran observar el funcionamiento de esta parte del sistema, se le pidió al usuario que ingresará los siguientes datos:

- **Nombre:** *Clase08*
- **Módulo o ZIT:** *Mural Interactivo*
- **Fecha:** *La del día de las pruebas*
- **Horario:** *Dos minutos más de la hora actual* (a la del día de las pruebas)

Una vez ingresados estos datos, los usuarios dieron la orden de programación al presionar el botón de la parte inferior de nombre “Programar”, observando que aparecía un cuadro de diálogo de confirmación en el que se indicaba si deseaba continuar con la programación de la ZIT a la hora y fecha indicada. Una vez confirmado, observaron que la ZIT quedaba programada al mostrar en la interfaz otro cuadro de dialogo que indicaba el éxito de la programación y que en la interfaz correspondiente a “Programaciones actuales”, mostraba la programación que acaba de realizar. Al cabo de dos minutos, percibieron que la ZIT (Mural Interactivo) comenzaba su encendido, tal y como se había indicado anteriormente en la programación.

Recomendaciones:

Uno de los profesores intentó editar la programación de la ZIT que realizó en la pruebas. Actualmente solo se pueden cancelar programaciones, por lo que se recomienda que también el profesor pudiese editarlas, cambiando algún de los parámetros: Nombre, ZIT, Fecha, Hora. Por otro lado, a pesar que ninguno de los profesores hizo la observación, se recomienda que la programación de las ZIT se pudiera realizar en conjunto y no de forma individual como es actualmente, esto es, poder realizar la programación del encendido de distintas ZIT en una sola tarea.

6. Pruebas con usuarios y análisis de resultados

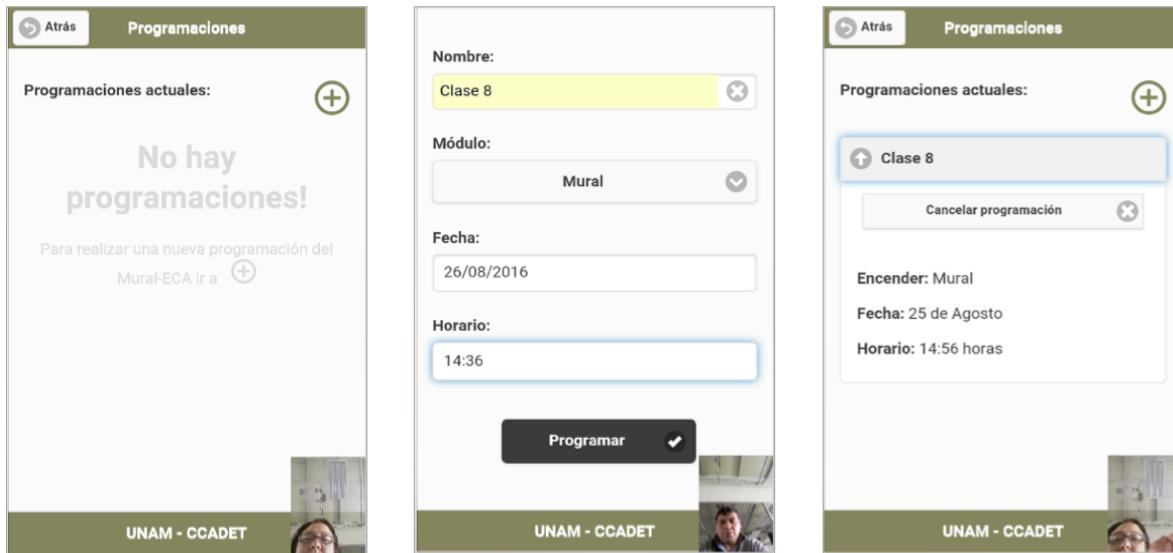


Figura 6.6. profesor realizando la tarea de programación del encendido del Mural Interactivo, a una hora y fecha indicada.

Una vez aplicadas todas las actividades o tareas a los profesores, se les pidió que contestaran el cuestionario de salida o usabilidad. Como se mencionó anteriormente, este cuestionario llamado SUS (System Usability Scale) permite medir la usabilidad de un sistema o producto y consta de 10 preguntas con cinco opciones de respuesta cada una, que van desde frecuentemente de acuerdo a frecuentemente en desacuerdo (para ver cuestionario ir a la parte de Anexos) [14].

A continuación se muestran los puntos obtenidos (en escala del 0-100) para cada usuario en esta encuesta:

- Usuario 1: **90**
- Usuario 2: **78**
- Usuario 3: **80**
- Usuario 4: **96**
- Usuario 5: **66**

Calculando el promedio de todos estos puntajes, tenemos como resultado una media de **82**. Para darle una interpretación a este número, la SUS provee una escala de puntuación, la cual se observa en la *figura 6.7*. En ella se muestra un rango de aceptación, donde en No Aceptable se encuentran los puntajes desde 0 hasta 50 y Aceptable desde 70 a 100. El otro rango corresponde a un adjetivo

6. Pruebas con usuarios y análisis de resultados

de clasificación que hace referencia a que la usabilidad del sistema puede ser la Peor Imaginable (de 0 a 25), Pobre (de 25 a 38), *Ok* (de 38 a 52), Buena (de 52 a 73), Aceptable (de 73 a 85) y la Mejor imaginable (de 85 a 100). Por último muestra una escala de calificaciones que va desde la A a la F, donde una A se obtiene con un puntaje de 90 a 100, una B con 80 a 90, una C de 70 a 80, una D de 60 a 70 y una F de 0 a 60.

Para el caso del sistema evaluado, el promedio obtenido (82) recae sobre el rango de aceptable, con una clasificación de entre bueno y excelente y una calificación de B.

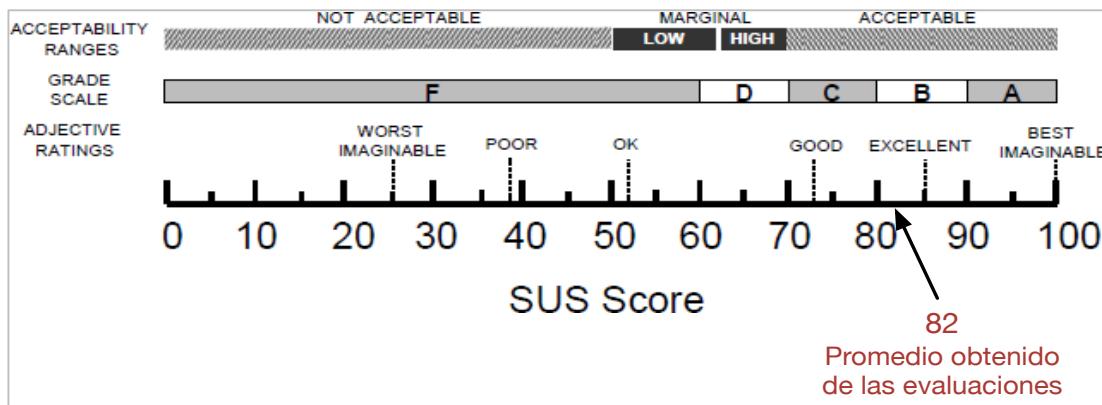


Figura 6.7. Escala de puntuación SUS usada para evaluar la usabilidad de un producto o sistema [14].

CONCLUSIONES

En el capítulo 2, correspondiente al Diseño Centrado en el Usuario (DCU), se mencionó que existen un conjunto de requerimientos para considerar a un producto verdaderamente usable, esto es, que el producto o sistema tiene que ser para el usuario útil, eficiente, efectivo, fácil de aprender, satisfactorio y accesible.

Con base a los resultados obtenidos del cuestionario de salida (usabilidad) hecho al final de las pruebas por los usuarios y a las observaciones hechas de la interacción del usuario con el sistema desarrollado, se puede concluir que el sistema fue:

- ✓ **Útil.** Debido a que se pudieron alcanzar las metas específicas del usuario para el uso de las zonas interactivas de trabajo (ZIT) del Mural y ECA's. Esto es, el profesor pudo encender, apagar, programar y ejecutar archivos en estas zonas, tal y como lo hacia antes, pero de una forma distinta, que le permitía, de manera considerable, reducir el número de tareas para el uso o acceso a ellas.
- ✓ **Eficiente.** Debido a que el usuario pudo alcanzar sus metas para el uso de las ZIT en un tiempo mucho menor en comparación a como se realizaban anteriormente. Esto es, antes tenía que ir individualmente a cada una de las ZIT para encender los dispositivos correspondientes <>PC y proyector (es)<>, luego cargar archivo y proyectarlo/ejecutarlo en las PC, y al final de su clase, apagar los dispositivos. Con el sistema desarrollado, desde la interfaz o aplicación Web desde su escritorio o zona de trabajo puede realizar ese conjunto de tareas de una forma mucho más sencilla y rápida, sin preocuparse por dónde se encuentran los controles remotos para los proyectores o dónde se encuentran las PC para cada ZIT, etc.
- ✓ **Efectivo.** Dada la descripción de la página principal de la interfaz Web al inicio de cada prueba y la forma en que realizaron cada una de sus tareas, nos percatamos que, en general, el modelo mental creado por cada profesor acerca de la imagen del sistema fue aceptable. Esto es, pudo describir de manera correcta qué funciones o tareas podía realizar desde la interfaz Web y qué esperar de cada una. Después, al realizar sus pruebas, percibieron que el sistema se comportaba, en la mayoría de los casos, de la forma en que ellos esperaban.
- ✓ **Fácil de aprender.** Para saber si el sistema de control resultó fácil de aprender por los profesores, es necesario realizar pruebas o hacer uso del sistema después de un periodo de

Conclusiones

inactividad y observar si fue posible realizar las tareas de forma correcta y sin instrucciones. Por otro lado, una de las preguntas del cuestionario de salida fue: *¿Cree que la gente pueda aprender a usar el sistema rápidamente?* A lo que la mayoría de los profesores respondió positivamente (con “Frecuentemente de acuerdo”).

- ✓ **Satisfactorio.** A lo que se observó en las pruebas con los usuarios y después de haber analizado las videogramaciones hechas de cada uno, se puede decir que estuvieron, en general, satisfechos con el sistema, esto es, sus opiniones acerca de cómo se sintieron haciendo uso del sistema fueron positivas. Aunque sí realizaron muchas observaciones y recomendaciones acerca de cómo pensaban y/o preferirían que funcionara cierta parte del sistema.
- ✓ **Accesible.** Todas las tareas que debe llevar a cabo el profesor para alcanzar su meta de acceso o uso de alguna de las ZIT, se llevan a cabo desde una interfaz Web, la cual se ejecuta desde cualquier dispositivo inteligente que el profesor lleve consigo, ya sea un teléfono inteligente, tableta, laptop, etc. Por lo que esto lo convierte en un sistema muy accesible.

Aunque en estas primeras evaluaciones del sistema con usuarios se obtuvo una calificación promedio de usabilidad entre buena y excelente de acuerdo a la escala SUS, aún se pueden hacer muchas mejoras en el sistema con el fin de satisfacer más las necesidades del usuario y subir más en la escala. Esto es, se debe seguir con el proceso de iteración del DCU, realizando mejoras en el sistema y probarlas con usuarios, hasta alcanzar un punto óptimo.

Por otro lado, podemos concluir que se logró el objetivo de que el sistema de control de dispositivos representara una herramienta útil para el profesor en el uso de las zonas interactivas de trabajo, debido a que pudo acceder a ellas de una forma muy fácil y sencilla. Todo ello se traduce en que ahora el profesor puede hacer un mejor uso del Aula del Futuro y emplear más tiempo en sus actividades con los alumnos y menos tiempo en el manejo y acceso a las zonas de trabajo del Mural o ECA.

REFERENCIAS

- [1] Gamboa, F. (2007). Diseño, uso y evaluación de tecnología de cómputo ubicuo en el aula universitaria, bajo un enfoque centrado en el usuario y su tarea. 1-5. México.
- [2] López-Rayón Parra, A.E. (2001), Ambientes Innovadores de Aprendizaje, in XVII Simposio, S.M.d.C.e. Educación, Editor. 2001, SOMECE: Guanajuato, México.
- [3] Garzón Gaitán, C.A. (1996) Diseño de Ambientes de Aprendizaje Significativo. Revista de Educación en Tecnología. 1(1).
- [4] Hannafin, M., S. Land, and K. Oliver (2000), Entornos de aprendizaje abiertos: Fundamentos, métodos y modelos., in Diseño de la instrucción. Teorías y modelos., C. Reigeluth, Editor. Aula XXI Santillana: Madrid. 125-152.
- [5] Jiménez Ugalde, A.I. (2002), Creación de Ambientes de Aprendizaje, in Licenciatura en Intervención Educativa. 2002, Universidad Pedagógica Nacional: México D.F.
- [6] Hassan-Montero, Y.; Ortega-Santamaría, S. (2009). Informe APEI sobre Usabilidad. Gijón: Asociación Profesional de Especialistas en Información, 2009, 73pp.
- [7] Norman, D. A. (1988). The design of everyday things. New York: Doubleday. 13-17
- [8] Dix, A., Finlay, J., Abow, G. D., Beale, R. (2004). Human Computer Interaction. England: Pearson Prince Hall.
- [9] Paterno, F. (2010). Task Models in Interactive Software Systems. IEEE.
- [10] <https://www.usability.gov/how-to-and-tools/methods/task-analysis.html> (2006)
- [11] Gutiérrez F.L., Gea, M. (2002). El diseño. España: Universidad de Granada.
- [12] Rubin, J., Chisnel D.L. (2008). Handbook of Usability Testing. USA: Wiley.
- [13] <https://www.usability.gov/how-to-and-tools/methods/usability-testing.html> (2006)
- [14] Brooke, J. (2013). SUS: A Retrospective. United Kingdom. Recuperado de <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>
- [15] Departamento de Electrónica FI UNAM. (2014). Conceptos básicos de Redes y TCP/IP. México.
- [16] Tanenbaum, A.S., Wetherall D.J. (2012). Redes de computadoras. USA: Pearson.
- [17] http://w3schools.sinsixx.com/tcpip/tcpip_intro.asp.htm (2009)
- [18] UDP and TCP (2013): Comparison of Transport Protocols. Recuperado de: <https://www.youtube.com/watch?v=Vdc8TCESIg8>.

Referencias

- [19] Dunkels, A. (2010). Interconecting Smart Object with IP. USA: Morgan Kaufmann.
- [20] Sublash, C.Y., Sanjay, K.S.. (2009). An Introduction to Client/Server Computer. Nueva Dheli: New Age International.
- [21] Lehmann, H. (2004). Client/Server Based Statistical Computing. Germany.
- [22] Deitel, P., Deitel, H., Deitel A. (2014). Internet & World Wide Web. México: Pearson.
- [23] <https://www.iana.org/>
- [24] AMD. (1995). Magic Packet Technology. USA. Recuperado de:
<http://support.amd.com/TechDocs/20213.pdf>
- [25] Abilene, P., Alfandika, A.C. (2014). Implementing Wake-on-LAN in Institutional Networks
- [26] <http://www.computerhope.com/jargon/w/wol.htm> (1998)
- [27] <http://ccm.net/faq/9200-shutdown-or-wake-up-a-pc-on-a-lan> (2013)
- [28] <http://lordandhooks.com/index.php/blog/the-ultimate-wake-on-lan-guide/> (2011)
- [29] <https://www.wireshark.org/> (1998)
- [30] Crowther, R., Lennon, J., Blue, A., Wanish, G. (2014). HTML5 In Action. USA: Manning
- [31] <http://www.w3schools.com/html/default.asp> (2016)
- [32] <http://www.w3schools.com/css/default.asp> (2016)
- [33] Sheley Powers. (2009). Learning Java Script. USA: O'Reilly
- [34] <http://www.w3schools.com/js/default.asp> (2016)
- [35] Bibeault, B., Katz, Y. (2008). JQuery In Action. USA: Manning
- [36] <http://www.w3schools.com/jquery/default.asp> (2016)
- [37] <https://jquerymobile.com/> (2016)
- [38] <http://www.w3schools.com/jquerymobile/default.asp> (2016)
- [39] http://www.w3schools.com/xml/ajax_intro.asp (2016)
- [40] <http://www.json.org/> (2015)
- [41] http://www.w3schools.com/js/js_json_intro.asp (2016)
- [42] Sevilleja, C., Lloyd, H. (2015). MEAN Machine. USA: Leanpub.
- [43] <https://nodejs.org/> (2016)
- [44] <http://expressjs.com/> (2016)
- [45] Yappa, H. (2013). Express Web Application Development, United Kingdom: Packt
- [46] http://www.tutorialspoint.com/nodejs/nodejs_scaling_application.htm (2016)
- [47] Galeano, G. (2009). Programación de Sistemas Embebidos en C. España: Alfaomega

Referencias

- [48] J. W. Valvano, (2012), Embedded Systems: Introduction to ARM CortexTM-M Microcontrollers, USA.
- [49] <http://www.ti.com/lit/ug/spmu365c/spmu365c.pdf> (2014)
- [50] Texas Instruments, (2014), <http://www.ti.com/product/tm4c1294ncpdt> (2014)
- [51] <http://www.ti.com/lit/ml/spmz858/spmz858.pdf> (2014)
- [52] <http://www.ti.com/tool/cestudio> (2014)
- [53] <http://energia.nu/> (2012)
- [54] <http://www.ti.com/lit/ug/spmu298d/spmu298d.pdf> (2014)
- [55] https://github.com/yuvadm/tiva-c/blob/master/boards/dk-tm4c129x/enet_io/enet_io.c (2014)
- [56] <http://www.ti.com/lit/ds/spms433b/spms433b.pdf> (2014)
- [57] <https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.03x+1T2016> (2012)
- [58] <http://www.computerhope.com/jargon/n/nulmodca.htm> (2000)
- [59] <https://www.freebsd.org/doc/en/articles/serial-uart/article.html> (1996)
- [60] <http://www.computerhope.com/shutdown.htm> (1998)
- [61] <http://ecee.colorado.edu/~mcclurel/max232ds.pdf> (1997)
- [62] <https://nodejs.org/api/dgram.html> (2010)
- [63] http://www.support.nec-display.com/dl_service/data/manual/english/UM330X_manual_ENG.pdf (2012)
- [64] <https://www.nngroup.com/articles/mental-models> (2010)

ANEXOS

- Algoritmo implementado para el control de una computadora en LAN

Encendido

```
1 function turnOnPc(pcName) {
2     // Declaración de variables
3     var macAddr;
4     var control = false;
5
6     // Tomar dir. MAC correspondiente guardada en fichero "PcAulaFuturo"
7     fs.readFile(PcAulaFuturo, 'utf8', function(err,data){
8         var auxA = JSON.parse(data);
9         for (var i = auxA.PC.length - 1; i >= 0; i--) {
10             if(auxA.PC[i].pcName == pcName){
11                 macAddr = auxA.PC[i].macAddr;
12                 control = true;
13                 break;
14             };
15         };
16
17         // Si existe la PC en la lista, continuar con el proceso de encendido
18         if (control){
19             var client      = dgram.createSocket('udp4'); //Crear socket UDP
20             var PORT        = 9;                      // Puerto destino (NIC)
21             var HOST        = ipbroadCast;           // IP broadcast (192.168.11.255)
22             var splitmacAddr = macAddr.split(':');
23             var HexN        = [];
24
25             // Convertir la Dir. Física de la PC a formato hexadecimal
26             for (var i = splitmacAddr.length-1; i >= 0; i--) {
27                 HexN[i] = parseInt(splitmacAddr[i], 16);
28             };
29
30             // Objeto "Buffer" que contiene los datos para el paquete mágico
31             var MagicPacket = new Buffer([
32                 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
33                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
34                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
35                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
36                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
37                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
38                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
39                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
40                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
41                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
42                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
43                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
44                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
45                 HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],
```

Anexos

```
46      HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],  
47      HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5],  
48      HexN[0],HexN[1],HexN[2],HexN[3],HexN[4],HexN[5] );  
49  
50      // Atar socket a un puerto y habilitar la transmisión broadcasting  
51      client.bind( function() { client.setBroadcast(true) } );  
52  
53      // Envío de paquete mágico por la red, al puerto y host especificado  
54      client.send(MagicPacket, 0, MagicPacket.length, PORT, HOST,  
55  
56      // Una vez enviado el paquete, ejecuta función callback  
57      function(err, bytes) {  
58  
59          // Hubo error en el envío?  
60          if (err){  
61              statusPC = "0";  
62              throw err  
63              client.close(); //Cerrar conexión  
64          };  
65  
66          console.log('Envío de paquetes UDP a ' + HOST +':'+ PORT);  
67          statusPC = "1";  
68          console.log("encendiendo "+pcName);  
69          client.close(); //Cerrar conexión  
70      });  
71  
72      // Si no existe PC en la lista, retorno de función  
73  }else{  
74      statusPC = "2";  
75      console.log("Error: no se encontró PC en la lista")  
76  };  
77  
78  }); // Cerrar fichero  
79 };//Cierre de función
```

Apagado

(En Servidor Maestro)

```
1 function TurnOffPC(pcName) {  
2     // Declaración de variables  
3     var ipAddr;  
4     var control = false;  
5     // Tomar dirección IP correspondiente guardada en fichero "PcAulaFuturo"  
6     fs.readFile(PcAulaFuturo, 'utf8', function(err,data){  
7         var auxA = JSON.parse(data);  
8         for (var i = auxA.PC.length - 1; i >= 0; i--) {  
9             if(auxA.PC[i].pcName == pcName){  
10                 ipAddr = auxA.PC[i].IP;  
11                 control = true;  
12                 break;  
13             };  
14         };  
15     };
```

Anexos

```
16 // Si existe la PC en la lista, continuar con el proceso de apagado
17 if (control){
18     // Opciones de petición HTTP
19     var options = {
20         hostname: ipAddr, // Dirección IP del host destino
21         port: 3001, // Puerto en el que escucha peticiones
22         method: 'GET', // Método de la petición
23         path: "/shutdownMyself" // Ruta (URL)
24     };
25     // Envío de petición HTTP a servidor esclavo
26     var req = http.request(options, function(res) {
27         res.setEncoding('utf8');
28         res.on('data', function(result) {
29             console.log(result);
30         })
31     });
32     // Hubo error en el envío de la petición ??
33     req.on('error', function(e) {
34         console.log('ERROR: ' + e.message);
35     });
36     req.end(); // Terminar petición
37
38     // Indicar al usuario que la PC no se encuentra en la lista
39 }else{
40     console.log("Error: no se encontró PC en la lista")
41 }
42}); //fin fichero
43}; //fin función
```

Apagado

(En Servidor Esclavo)

```
1 // Módulo que permite el uso de la línea de comandos (Terminal)
2 var cmd = require('node-cmd');
3
4 // Ruta "shutdownMyself"
5 router.get('/shutdownMyself', function(req, res, next) {
6
7     // Uso de comando "shutdown". Donde los parámetros:
8     -s → Indica que se requiere el apagado
9     -f → Indica el cierre forzado de la PC
10    -t 0 → Indica el tiempo de espera (en seg) para el apagado de la PC
11    -m \IPaddress or HostName → Indica a la PC que se requiere apagar
12        mediante su dirección IP o nombre del host (en este caso se usa
13        su misma dirección IP para apagarse así misma)
14    cmd.get('shutdown -s -f -t 0 -m \\\\'+addresses[0]);
15
16    // Retorno de respuesta de petición al servidor maestro
17    res.end("Apagado Ok");
18});
```

Anexos

- Proyección de archivos
(En Servidor Maestro)

```
1 function sendCmdToSlaveFile(dirIP, fileName) {
2   // Indicar opciones de la petición
3   var options = {
4     hostname: dirIP,    //dirección IP de la PC
5     port: 3001,          //Puerto donde recibe conexiones
6     path: "/start?data=\\""+hostNameServer+"\\"programToShare3\\"+fileName
7   };
8   //Envío de petición al servidor esclavo
9   var req = http.request(options,
10     function(res) {      // Función callback
11       res.setEncoding('utf8');
12       //Mensaje de servidor esclavo
13       res.on('data', function(result) {
14         console.log(result);
15       });
16       //Hubo error en el envío de la petición ??
17       req.on('error', function(e) {
18         console.log('ERROR: ' + e.message);
19       });
20     }); //Cierre de función callback
21   req.end(); //Terminar petición
22 }; //Cierre de función
```

(En Servidor Esclavo)

```
1 //Modulo de Node para la ejecución de procesos/archivos
2 var exec = require('child_process').execFile;
3 exports.start = function(req, res) {
4   // Tomar de la URL la variable data, la cual contiene la ruta de acceso
5   // al archivo dentro de la carpeta programsToShare
6   var file = req.query.data;
7   // Ejecución de un proceso, en este caso del archivo file
8   execFile(file,function callback(error, stdout, stderr){
9     console.log("stdout") //Proceso terminado
10   });
11   res.end(); //Cierre de respuesta
12 });
```

Anexos

- **Algoritmo implementado para el control de proyectores en LAN**

(En Servidor Esclavo del sistema embebido)

```
1 if(ustrncmp(pcName, "/puerto1?", 8) == 0){  
2  
3     // Habilitamos interrupción de UART Rx  
4     UARTIntEnable(UART1_BASE, UART_INT_RX | UART_INT_RT);  
5     // Deshabilitamos WatchDog, para evitar resets en este contexto  
6     WatchdogResetDisable(WATCHDOG0_BASE);  
7  
8     // Declaración de variables  
9     unsigned long baudRate; //Velocidad de transmisión del proyector  
10    static char TypeResponse[2]; //Tipo de respuesta (Error(0,2), Success(1))  
11    uint32_t ui32Loop, i, j;  
12  
13    // Obtener baudrate, llamada a función getParamURL() pasando como param.  
14    .      un apuntador que apunta al carácter que está después del signo ?  
14    baudRate = getParamURL((char*) pcName + 12);  
15  
16    // Configuración UART1: 8 bits de datos, 1 bit start y 1 bit stop, etc  
17    UARTConfigSetExpClk(UART1_BASE, g_ui32SysClock, baudRate,  
18                (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |  
19                UART_CONFIG_PAR_NONE));  
20    // Variables usadas para sincronización de hilos (ISR y Main)  
21    Flag = 0; // Bandera usada para indicar que se ingreó a la ISR  
22    initFifo(); // Buffer tipo FIFO para almacenar respuesta del proyector  
23  
24    // Limpiar FIFO Receive para quitar basura y almacenar respuesta de proj  
25    while(UARTCharsAvail(UART1_BASE))  
26        j = UARTCharGetNonBlocking(UART1_BASE);  
27  
28    // Envío de comandos de control al proyector, tomados de la URL  
29    for(i=0; i<nnumberCmd; i++)  
30        UARTCharPut(UART1_BASE, onCmdHex[i]);  
31  
32    // Delay para esperar a que ocurra la ISR de UART Rx (Ack de proyector)  
33    for(ui32Loop = 0; ui32Loop < delayToWaitResopnseProj; ui32Loop++){  
34        if (Flag == 1)  
35            break;  
36    }  
37    // Limpiar Flag para posteriores entradas a UART Rx  
38    Flag = 0;  
     // Capturar el tipo de respuesta enviada por el proyector
```

Anexos

```
39     TypeResponse[0] = checkProjectorResponse();
40
41     // Imprimir tipo de respuesta (por UART0 Tx; solo para fines de prueba)
42     if (TypeResponse[0] == '0')
43         // Error: los comandos de respuesta de proyector no corresponden con
44         .           los comandos dados por el usuario
45         UARTCharPut(UART0_BASE,0X50);
46     else if (TypeResponse[0] == '1')
47         // Success: El comando fue correctamente enviado y ejecutado por el
48         .           proyector (Encendiéndose o Apagándose)
49         UARTCharPut(UART0_BASE,0X51);
50     else if (TypeResponse[0] == '2')
51         // Error: No respuesta de proyector, no hay conexión Proyector-uC
52         UARTCharPut(UART0_BASE,0X52);
53
54
55     // Retorno al cliente (servidor maestro) de tipo de respuesta
56     psFile -> data = TypeResponse;
57     psFile -> len = strlen(TypeResponse);
58     psFile -> index = psFile->len;
59     psFile -> pextension = NULL;
60
61     // Desabilitamos interrupción para que solo se utilice
62     // en el contexto de Envío de Comandos y Respuesta del Proyector
63     // y evitar entradas no deseadas por otros contextos
64     UARTIntDisable(UART1_BASE, UART_INT_RX);
65     WatchdogResetEnable(WATCHDOG0_BASE);
66     return(psFile);
67 }
```

Documentos para pruebas con usuarios

- **Protocolo de Bienvenida**

Buenos días, gracias por permitirnos distraerle unos momentos de sus actividades, mi nombre es _____ y estaré con usted en esta sesión.

Sistema de control de dispositivos electrónicos del AF

Fecha de 1^a. Evaluación: 23/Agosto/16

Permítame explicarle por qué estamos aquí.

Estamos probando el prototipo de un sistema de control que le permitirá realizar de una manera más rápida y sencilla el uso de las zonas interactivas de trabajo del Muro Interactivo y Escritorio Colaborativo Aumentado (ECA) que se encuentran en el Aula del Futuro.

Mediante el estudio de sus acciones y comentarios podremos identificar las posibles mejoras al sistema. Por ello le pedimos que actúe con naturalidad, pero sobre todo, sus opiniones las exprese en voz alta, ya que este ejercicio tiene como propósito evaluar al sistema. Es importante recordarle que estamos evaluando al sistema y no a usted.

Debido a que es un sistema que está en proceso de evaluación, eventualmente podría presentar variantes imprevistas, por lo que le agradeceremos considerarlo.

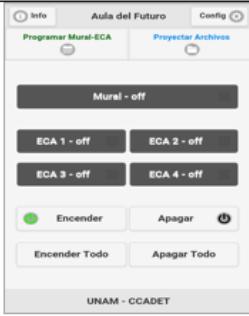
Se le presentará un prototipo y se le pedirá que realice algunas tareas típicas para las cuales está diseñado el sistema, la sesión consistirá en que usted las efectúe y describa en voz alta sus acciones, así como cualquier opinión que tenga, ya que esto será de gran utilidad para mejorar el sistema.

Una vez comenzada la sesión, siéntase en total libertad de hacer cualquier pregunta aunque no podré contestar algunas de ellas, ya que el objetivo es simular la situación real en la cual operará el sistema de manera autónoma.

¿Tiene alguna duda?

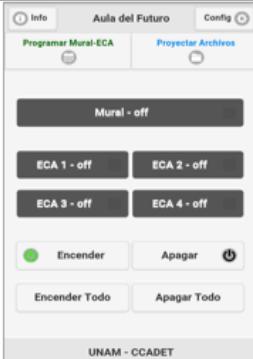
Anexos

- Hipótesis de la tarea

Pantalla	Hipótesis	Tarea	Desarrollo de tarea (Instrucciones)	Tiempo estimado
Menú principal/Página de Inicio				
	<p>1) El usuario es capaz de reconocer las acciones que puede realizar en la pantalla</p> <p>2) El usuario es capaz de encender el Mural Interactivo y ECA 1.</p>	Reconocer el menú principal y encender el Mural Interactivo y ECA1.	<p>1) Apuntando con el ratón, describa en voz alta los elementos que ve en la pantalla e indique para qué cree que sirven.</p> <p>2) Por favor encienda el Mural Interactivo y el Escritorio Colaborativo Aumentado 1 (ECA1).</p>	1' 30"
Menú principal (Mural y ECA1 se encendieron exitosamente)				
	<p>1) El usuario es capaz de reconocer los elementos</p> <p>2) El usuario es capaz de comenzar el proceso para subir un archivo.</p>	<p>1) Reconocer elementos</p> <p>2) subir un archivo y proyectarlo/ejecutarlo en el Mural</p>	<p>1. ¿Funcionó como esperaba?</p> <p>2. Apuntando con el ratón, describa en voz alta los cambios que percibe en la pantalla e indique para qué sirven.</p> <p>3. Por favor suba el archivo "XXX" y proyectelo en el Mural Interactivo.</p>	1'20"
Gestión de Archivos (Proyección de archivos)				
	<p>1) El usuario es capaz de reconocer los elementos</p> <p>2) El usuario es capaz de eliminar un archivo</p>	Reconocer elementos y eliminar el archivo subido anteriormente	<p>1. ¿Funcionó como esperaba?</p> <p>2. Por favor elimina el archivo "XXX" que subió anteriormente</p>	20"
Gestión de Archivos (Eliminar archivo)				
	<p>1) El usuario es capaz de reconocer los elementos</p> <p>2) El usuario es capaz de regresar al menú principal</p>	Regresar al menú principal	<p>1. ¿Funcionó como esperaba?</p> <p>2. Regresa al menú principal.</p>	15"

Anexos

Menú principal (Mural y ECA1 apagados exitosamente)

	1) El usuario es capaz de apagar el Mural I. y ECA1.	Apagar el Mural Interactivo y ECA1.	1. ¿Funcionó como esperaba? 2. Ahora vamos a apagar el Mural Interactivo y ECA1 3. Por favor apáguelos.	35"
---	--	-------------------------------------	---	-----

Programar Mural Interactivo

	1) El usuario es capaz de reconocer los elementos 2) El usuario es capaz de realizar la programación de una Zona Interactiva de Trabajo (Mural)	Programar el encendido del Mural interactivo	1. ¿Funcionó como esperaba? 2. Ahora vamos a programar el encendido del Mural interactivo. 3. Por favor programe el Mural interactivo con los datos de la hoja X 4. ¿Funcionó como esperaba? 5. Regrese al menú principal.	45"
--	--	--	--	-----

Anexos

- **Actividades de la evaluación**

Usuario No.:_____

Hora de inicio:_____

Sistema para control de dispositivos
electrónicos

Fecha de 1^a. Evaluación: 23/Agosto/16

Instrucciones para el moderador

1. Ir tachando las actividades realizadas con el fin de no perderse durante la prueba. NO se realizará ninguna actividad si antes no ha concluido la anterior (a menos que se le instruya lo contrario).
2. En caso de que el usuario cometa algún error preguntar: *¿El sistema le informó con claridad qué fue lo que pasó?*

Menú principal

- 1) ____ Esta es la página principal.
- 2) ____ Apuntando con el ratón, describa en voz alta los elementos que ve en la pantalla e indique para que cree que sirven.
- 3) ____) Por favor encienda el Mural Interactivo y el Escritorio Colaborativo Aumentado 1 (ECA1).

Menú principal (Mural y ECA1 encendido exitosamente)

- 1) ____ ¿Funcionó como esperaba?
- 2) ____ Apuntando con el ratón, describa en voz alta los cambios que percibe en la pantalla e indique para qué sirven.
- 3) ____ Ahora vamos a subir un archivo y proyectarlo en el Mural interactivo.

Gestión de Archivos (Proyectar archivo)

- 1.____ Por favor vaya a la sección de Proyectar Archivos
- 2.____ Suba el archivo con el nombre que aparece en la ficha 1 y proyéctelo en el Mural Interactivo.
- 3.____ ¿Funcionó como esperaba?

Gestión de Archivos (eliminar archivo)

- 1.____ Por favor elimina el archivo subido anteriormente
- 2.____ ¿Funcionó como esperaba?
- 5.____ Regresa al menú principal.

Anexos

Menú principal (Apagar mural y ECA1)

1. _____ Ahora vamos a apagar el Mural Interactivo y ECA1
2. _____ Por favor apague el Mural Interactivo y el 1 (ECA1).
3. _____ ¿Funcionó como esperaba?

Programar encendido Mural Interactivo

1. _____ Ahora vamos a programar el encendido del Mural interactivo.
2. _____ Por favor vaya a la sección de programación Mural-ECA
3. _____ Por favor realice la programación del Mural usando los datos que aparecen en la ficha 2 (Esperar 2 min...)
4. _____ ¿Funcionó como esperaba?
5. _____ Regrese al menú principal.

Gracias por participar en la evaluación.

Anotar hora en que termine la evaluación _____

Notas del moderador: _____

Anexos

- **Cuestionario de salida (Usabilidad)**

Sistema para control de dispositivos
electrónicos

Fecha de 1^a. Evaluación: 23/Agosto/16

Lea cuidadosamente las siguientes afirmaciones y marque con una x que tan de acuerdo o desacuerdo está con ellas, considerando que 5 es equivalente a fuertemente desacuerdo y 1 a fuertemente de acuerdo. Si no está seguro (a) de que contestar marque 3.

	Fuertemente en desacuerdo					Fuertemente de acuerdo				
	5	4	3	2	1					
1.	Creo que me gustaría usar el sistema frecuentemente.									
2.	Encuentro el sistema innecesariamente complejo.									
3.	Pensé que el sistema era fácil de usar.									
4.	Creo que necesitaré la ayuda de un técnico para usar el sistema.									
5.	Encontré que las distintas funciones del sistema estaban bien integrada.									
6.	Pienso que había mucha inconsistencia en el sistema.									
7.	Me imagino que la gente aprenderá a usar el sistema bastante rápido.									
8.	Encuentro el sistema engorroso de usar.									
9.	Me sentí muy seguro usando el sistema.									
10.	Necesito aprender muchas cosas antes de poder utilizar el sistema.									