



FACULTAD DE INGENIERÍA

DEPARTAMENTO DE ELECTRÓNICA



66.99 Trabajo Profesional

Tema: “Sistema Domótico de uso general”

Martín Miodosky (81130); e-mail: mmiodosky@gmail.com

Agustín Bertamoni (81041); e-mail: agustinbertamoni@gmail.com

Profesor: Ing. Eduardo Ioli

Fecha: 25/08/2008

1. Índice

1. Índice	3
2. Resumen	7
3. Introducción	7
3.1 Determinación de la Necesidad	7
3.2. Historia, Antecedentes	7
3.3. Definiciones	8
3.4 Justificación del Proyecto	8
4. Objetivo	9
4.1 Finalidad del proyecto	9
4.2 Elementos a desarrollar	9
4.3. Problema a resolver	10
5. Definición del Producto	11
5.1 Requerimientos del usuario	11
5.2 Productos Competitivos	12
5.2.1 Elecei	12
5.2.2. HAI Omni Pro II	13
5.3 Matriz de planeamiento:	14
5.3 Requerimientos Técnicos	15
5.4 Casa de Calidad	16
5.4.1 Conclusiones	17
5.4.2 Aspectos a mejorar	18
5.5 Definición final del producto	19
5.5.1 Software de control y programación	21
5.5.1.1 Interfaz de usuario	21
5.5.2 Módulo controlador	22
6. Análisis de Factibilidad	25
6.1. Elección de una Solución	25
6.1.1 Red de Datos	25
6.1.1.1 RS-232	25
6.1.1.2 USB	25
6.1.1.3 Ethernet (IEEE 802.3)	25
6.1.1.4 Wi-Fi (IEEE 802.11b)	26
6.1.2 Dispositivos Controlados	26
6.1.2.1 Protocolo X.10	26
6.1.2.2 Dispositivos acoplados por Relay	29
6.1.2 Sensor de Temperatura	29
6.1.3 Conversor Analógico Digital	30
6.1.4 Microcontrolador	30
6.1.5 Software	31
6.1.5.1 Sistema operativo	31
6.1.5.2 Web Server	32
6.2 Normas técnicas utilizadas	34
6.2.1 IEEE 802.11b (Wi-Fi)	34
6.2.2 X.10	35
6.2.3 RS-232	37
6.2.4 SPI	38
6.3 Factibilidad de tiempos:	39
6.3.1 Tareas	40

6.3.2 Dependencia de las tareas	41
6.3.3 Estimación de tiempos medios por tarea	43
6.3.5 <i>Simulación de Montecarlo</i>	48
6.3.6 Comparación de resultados:	52
6.3.7 <i>Asignación de recursos</i>	52
6.3.8 <i>Diagrama de Gant</i>	53
6.6 Descripción del plan Comercial.....	57
6.6.1 Posicionamiento Competitivo del Producto.....	57
6.6.2 Precio.....	57
6.6.3 <i>Distribución y Venta</i>	58
6.7 Comunicación.	58
6.7.1 Medios utilizados.	58
6.7.2 Presupuesto asignado.	58
6.8 Descripción del plan de operación.....	59
6.8.1 <i>Descripción de Gastos Operativos</i>	59
6.8.2 <i>Parámetros de Calidad del Servicio</i>	59
6.9 Descripción plan de negocios.	60
6.10 Factibilidad Legal y responsabilidad civil	62
6.10.1 <i>Estudio de Patentes:</i>	62
6.10.3 <i>Responsabilidad Civil:</i>	63
7. <i>Ingeniería de Detalle</i>	63
7.1 Diseño de Hardware	63
7.1.1 <i>Diagrama de bloques</i>	63
7.1.2 <i>Fuente de Alimentación</i>	65
7.1.3 <i>Convertor de nivel RS-232</i>	66
7.1.4: <i>Sensor de temperatura con Convertor Analógico Digital</i>	68
7.1.5: <i>Buffers de Salida con banco de Relays</i>	69
7.1.6: <i>Módulo RCM4400W</i>	71
7.2 Diseño de Firmware.....	73
7.2.1 <i>Flujo de Comunicaciones</i>	73
Definición del modelo cliente servidor.....	73
7.2.1.2 <i>Comunicación Servidor-Módulos Controladores</i>	74
7.2.1.3 <i>Comunicación Módulo Controlador-Controlador X.10</i>	75
7.2.2 <i>Diagramas de Flujo de Firmware</i>	78
7.2.2.1 Main.h	79
7.2.2.2 Funciones de Socket.h	80
7.2.2.3 Funciones de X10-CM11.h	81
7.3 Diseño de Software: Interfaz Web	82
8. <i>Diseño y construcción del prototipo</i>	99
8.1. Módulo Controlador	99
8.1.1 Diseño placa base	99
8.1.2 Listado de componentes de la placa base.....	103
8.1.3 Consumo.....	103
8.1.4 Especificaciones de Montaje.....	103
8.2. Servidor	105
8.3. Red de datos.....	105
8.4. Costos.....	108
8.4.1 Costo componentes módulo controlador	108
8.4.2 Costo placa y soldadura módulo controlador	109
8.5 Disposición final del equipo	110

8.6. Consumo	111
8.7 Plan de prueba de Hardware y software	111
9. Estudios de Confiabilidad	113
9.1 Cálculo de confiabilidad de Hardware	113
9.1.1 Placa base	113
9.1.1.1 Resistores	113
9.1.1.2 Capacitores	114
9.1.1.3 Inductores	115
9.1.1.5 Diodos	115
9.1.1.5 Circuitos Integrados	115
9.1.1.5.1 LM2575	116
9.1.1.5.2 LM1117	116
9.1.1.5.3 TLC 548	116
9.1.1.5.4 MAX3222	117
9.1.1.5.5 74LS244	117
9.1.1.5.6 LM35	117
9.1.1.5 Relays	118
9.1.1.6 Conectores	119
9.1.1.7 Soldaduras	119
9.1.1.8 Tasa de fallas total Placa Base	119
9.1.2 Modulo RCM4400	120
9.1.2.1 Resistores	120
9.1.2.2 Capacitores	120
9.1.2.3 Circuitos Integrados	121
9.1.2.3.1 Microprocesador Rabbit4000	121
9.1.2.3.2 FPGA Xilinx xc35250e	121
9.1.2.3.3 Transmisor uw2453	121
9.1.2.3.4 Memoria Flash	122
9.1.2.3.5 Memoria RAM	122
9.1.2.3 Capacitores	122
9.1.2.4 Conectores	122
9.1.2.5 Tasa de fallas total del módulo RCM4400	123
9.1.2.6 Fiabilidad total y tiempo medio entre fallas (TMEF)	123
9.2 FMEA (análisis de efectos y modo de fallas)	124
9.2.1 Análisis de Fiabilidad de hardware	124
9.3 Estudios de confiabilidad de Software	126
9.3.1: Confiabilidad de Firmware de Controlador	126
9.3.2: Confiabilidad de Software Web Server	129
9.4 Resolución de Problemas	130
9.5 Ensayos de aceptación	131
9.5.1 Calidad del sistema completo	131
9.5.2 Verificación de la calidad	132
9.6 Politicas de Mantenimiento	132
9.6.1 Mantenibilidad	132
9.6.2 Serviabilidad	133
9.6.3 Contratos de mantenimiento	133
10. Conclusiones	134
10.1. Excelencias	134
10.2. Futuros Diseños	134
10.2.1 Interfaz Web	135

10.2.2 Funcionamiento asincrónico de tareas programadas	135
<i>Apéndice A: Código fuente</i>	137
A.1 Firmware	137
A.1.1 <i>Main.c</i>	137
A.1.2 <i>Socket.h</i>	140
A.1.3 <i>X10-CM11.h</i>	146
A.2: Código interfaz web	152
A.2.1 <i>Cgis:</i>	152
A.2.1.1 <i>setcontrollers.pl</i>	152
A.2.1.2 <i>setaddr.pl</i>	154
A.2.1.3 <i>deviceconf.pl</i>	156
A.2.1.4 <i>progevent.pl</i>	160
A.2.1.5 <i>simulpres.pl</i>	164
A.2.1.6 <i>monitor.pl</i>	166
A.2.2 <i>Html Templates:</i>	173
A.2.2.1 <i>header.tpl</i>	173
A.2.2.2 <i>footer.tpl</i>	173
A.2.2.3 <i>error.tpl</i>	173
A.2.2.4 <i>setcontrollers.tpl</i>	174
A.2.2.5 <i>setaddr.tpl</i>	174
A.2.2.6 <i>deviceconf.tpl</i>	175
A.2.2.6 <i>progevent.tpl</i>	177
A.2.2.7 <i>simulpres.tmp</i>	179
A.2.2.8 <i>monitor.tmp</i>	180
A.2.2.9 <i>xlabel.tmp</i>	181
<i>Apéndice C. Bibliografía</i>	183

2. Resumen

El presente proyecto consiste en el desarrollo de un sistema domótico, es decir, de automatización de vivienda o edificio de uso general.

La solución que se plantea busca no solo poner la tecnología en función del confort del habitante de una residencia, sino también facilitar y hacer más eficientes las tareas del personal de asistencia.

Además de los servicios convencionales se pretende que el usuario pueda controlar distintos dispositivos a través de una interfaz Web utilizando una PDA o cualquier PC, como también programar actividades sincrónicas y asincrónicas sobre los dispositivos.

Tales dispositivos pueden ser luces de la habitación, ventiladores, bombas de agua y filtros para piscinas.

Todo esto presupone el concepto de vivienda inteligente, lo cual en muchos casos sugiere una nueva vivienda. Lo que se plantea en este proyecto es la aplicación de este concepto de vivienda inteligente con la innovación de hacer en un ambiente diseñado sin tener en cuenta la instalación de un sistema domótico, es decir en una vivienda convencional ya construida.

Lo novedoso e innovador de este proyecto es que a diferencia de otros sistemas de domótica aquí se combinan varias tecnologías de acceso y control como Wi-Fi y X10 junto con dispositivos tipo PDA con el fin maximizar la relación costo beneficio así como también minimizar las reformas en instalaciones eléctricas ya existentes.

3. Introducción

3.1 Determinación de la Necesidad

Las necesidades de la vida cotidiana hacen que ciertas tareas las cuales antes requerían la atención humana ahora se realicen en forma automática o programada. Teniendo en cuenta que los habitantes de una vivienda pasan muchas de sus horas fuera de ella, surge la necesidad de tener el control de ciertos dispositivos en su ausencia, así como también simular la presencia de los habitantes del hogar.

3.2. Historia, Antecedentes

En la actualidad Domótica agrupa una serie de sistemas tecnológicos que aportan diferentes servicios al hogar, estos servicios pueden ser entre otros, seguridad, bienestar, comunicaciones, gestión energética, etc.

Hace tres décadas los equipos domésticos que se desarrollaban eran totalmente independientes, es decir, que funcionan de forma autónoma, sin necesidad de comunicarse con dispositivos de control remoto ni mucho menos con otros dispositivos del hogar.

Unos años mas tarde los equipos domésticos podían comunicarse entre si y hasta con un controlador pero aún la Domótica estaba relegada a un mercado muy reducido.

Hace poco más de una década desde la explosión de Internet los modelos tecnológicos relacionados a éste han progresado y forman parte del futuro de la domótica lanzando la misma hacia un mercado más masivo y dispuesto a probar nuevas tecnologías en busca de mejor calidad de vida.

Para poder cumplir con los requerimientos actuales de domótica se utilizan diferentes protocolos de acceso y control combinados con dispositivos de última tecnología para lograr sistemas atractivos, confiables y fáciles de utilizar.

3.3. Definiciones

3.4 Justificación del Proyecto

El mercado de domótica tiene una trayectoria a nivel mundial de no mas de 20 años y en la Argentina es un mercado el cual es apenas insipiente, habiendo solo algunas empresas que comercializan productos y servicios de este tipo. Estos sistemas son considerados como novedad existiendo un mercado potencial muy grande. En la mayoría de los casos los sistemas domóticos son diseñados en conjunto con la propiedad que se va a automatizar, de esta forma se planifica el cableado necesario para enviar las señales de control hacia los dispositivos a automatizar. Sin embargo, la mayoría de los sistemas no presentan alternativas para automatizar residencias u oficinas previamente diseñadas en las cuales no existe cableado de datos en las cercanías de los dispositivos a automatizar, haciendo muy difícil y costosa económicamente la implementación.

El presente trabajo propone un sistema de control distribuido con interfaz de programación y monitoreo que puede ser accedida desde cualquier explorador de Internet con conexión a la red de datos local o inclusive desde cualquier parte del mundo a través de Internet. Esto combinado con facilidad de utilización y programación. Además intenta ser innovador incluyendo distintas tecnologías como es la utilización de Wi-Fi (norma IEEE 802.11b), protocolo X.10 y software de uso libre como es el sistema operativo Linux (normas GNU).

4. Objetivo

4.1 Finalidad del proyecto

La finalidad del proyecto es el desarrollo de un sistema de control distribuido para la automatización de:

- Hogares
- Hoteles
- Locales Comerciales
- Oficinas

Además de automatización se desarrollan conceptos de seguridad mediante la simulación de presencia y de ahorro energético mediante la programación de tareas y el uso racional de los recursos.

Está dirigido a usuarios particulares, técnicos, ingenieros y arquitectos y diseñadores que quieran aplicar los conceptos de la automatización y el ahorro de energía para ofrecerlo y comercializarlo con sus clientes.

4.2 Elementos a desarrollar

El desarrollo del presente trabajo práctico consiste tanto en el diseño y desarrollo del Hardware y del Software de un prototipo de un sistema de control distribuido para automatización aplicado a la domótica. Para ello se desarrollaron los siguientes elementos:

- **Red de Datos:** Red necesaria para comunicar a los usuarios con el servidor que contiene el Software de Control y Programación comunicar al último con los Módulos Controladores.
- **Módulo Controlador:** Contiene el microcontrolador que es el encargado de procesar la información para operar las entradas y salidas. Este recibe la información de un Software de Control y Programación.
- **Software de Control y Programación:** Es la interfaz con el usuario a través una interfaz gráfica, amigable y fácil de utilizar. Mantiene la comunicación con el módulo controlador. Almacena las tareas programadas y controla los valores instantáneos para aplicar métodos de corrección y/o emitir alarmas.

- **Red de Aplicación:** Red eléctrica utilizada para transmitir las señales y para conectar los dispositivos a controlar. Se busca modificar en lo menor posible esta red para facilitar la implementación del sistema.

4.3. Problema a resolver

En el desarrollo del presente trabajo práctico se presume la resolución de los siguientes problemas:

- Elección del producto a desarrollar y solución para el mismo (comunicación, protocolos, etc).
- Selección de componentes con disponibilidad en el mercado local
- Diseño del circuito del prototipo
- Fabricación del prototipo
- Elección del sistema operativo y del lenguaje de programación del Software de Control y programación
- Programación del Firmware del Microcontrolador
- Desarrollo del Software de Control y Programación
- Implementación de la Red de Datos

5. Definición del Producto

5.1 Requerimientos del usuario

Para el diseño y desarrollo de nuestro producto nos colocamos del lado del cliente y listamos los siguientes requerimientos en base a la competencia y al mercado actual.

- 5.1.1 Confiable:** Todo equipo electrónico debe ser confiable aunque esta característica es de vital importancia en todo sistema de control. Por confiabilidad se entiende la correcta ejecución del control para el que ha sido programado a lo largo del tiempo.
- 5.1.2 Fácil de utilizar:** El producto debe poder ser utilizado por cualquier miembro de la familia como un electrodoméstico mas, por lo que las interfaces deben ser simples y amigables, en cuanto a la instalación la misma debe estar perfectamente documentada cubriendo todos los casos de aplicación del producto.
- 5.1.3 Económico:** Este es un requerimiento del cliente hacia cualquier producto en el mercado y es de importancia a la hora de competir con otros productos. En un sistema de control no obstante un precio económico no debe ser logrado a costa de otros factores de mayor importancia vital como por ejemplo la confiabilidad.
- 5.1.4 Fácil de instalar:** El sistema debe ser fácil de instalar, permitiendo montarse en una edificación existente y antigua que no haya sido diseñada para un sistema inteligente, evitando de esta forma trabajos de cableado adicional.
- 5.1.5 Velocidad de respuesta:** Un sistema de control tendrá asociada una velocidad de respuesta que es el tiempo entre que dada una condición de trigger se acciona la salida que especifica la lógica. La velocidad de respuesta estará dada por los retardos de comunicación y procesamiento.
- 5.1.6 Escalable:** Debe brindar la posibilidad de incrementar en forma sencilla la cantidad de módulos a controlar y tener posibilidad de poder agregar mas

módulos de control en caso que se supere la cantidad de dispositivos que puede controlar cada módulo.

- 5.1.7 Robusto:** Los sistemas de control deben ser robustos desde el punto de vista del hardware tanto como del software. Un hardware robusto implica un circuito confiable con baja tasa de fallas instalado en un gabinete resistente. Un software de control robusto por su parte implica que el controlador ejecutará el código de forma confiable y segura, esto es, ejecutará el control tal cual se programó de manera constante.

5.2 Productos Competitivos

5.2.1 Elecei



Figura 5.2.2: Gráfico panel de control Elecei

Es un sistema de domótica el cual es realizado completamente en Argentina. Sus principales características son:

- **Conectividad:** Utiliza bus de datos, que requiere cableado estructurado con cables UTP. Debido a esta razón es que el sistema Elecei se ajusta mejor a nuevas instalaciones en donde se diseña la cañería para el bus de datos que utiliza cables UTP. Utilizar cableado lo hace confiable aunque mas complicado de instalar. El sistema de bus de datos es de protocolo propietario.
- **Control:** Se realiza a través de una unidad “touch screen” la cual es provista por Elecei y se programa en forma personalizada al cliente, de forma tal que en la pantalla se puede observar el plano de la casa con los distintos dispositivos a utilizar. También se controla a través de pulsadores, de una PC o PDA si se tiene instalado un sistema Wi-Fi. Utiliza timers para tareas programadas.

- Dirección de control: Es unidireccional, es decir que el sistema no indica el estado de un dispositivo, simplemente envía la orden, si el dispositivo se encuentra apagado lo enciende y si se encuentra encendido lo apaga. Se deben cambiar las llaves de luz instaladas por pulsadores. La empresa provee los pulsadores con los distintos motivos para no alterar la estética del lugar.
- Seguridad de acceso: Se tiene la posibilidad de ingresar claves para controlar dispositivos de uso restringido a ciertos usuarios.

5.2.2. HAI Omni Pro II



Figura 5.2.2: Kit sistem HAI Omni Pro II

Sistema de control distribuido con gran éxito en Estados Unidos y Europa. Consta de una unidad central y múltiples unidades remotas. Las principales características son:

- Unidad central posee un controlador X.10 que lo hace compatible con productos X.10 de otros vendedores.
- Unidad central posee conectividad Ethernet para acceder desde la red local o desde Internet. Utiliza software propietario para conectarse y permite hasta 3 conexiones simultáneas.
- Se pueden agregar módulos externos (requieren cableado) de:
 - Salidas controladas a Relay.
 - Módulos con Interfaz Touch Screen
 - Termostatos para control de temperatura

Esta hace de un sistema altamente escalable, pero que requiere de cableado hacia la unidad central dificultando así su instalación aunque en menor medida que el sistema Elecei.

5.3 Matriz de planeamiento:

Su objetivo es cuantificar los requerimientos del usuario propuestos en base a su importancia y prioridad. A cada una de las propiedades mencionadas se le asigna un número acorde a su importancia con la siguiente escala:

5: Muy importante

1: Sin importancia

Requerimientos	Importancia				
Confiable	5	4	3	2	1
Fácil de utilizar	5	4	3	2	1
Económico	5	4	3	2	1
Fácil de instalar	5	4	3	2	1
Velocidad de respuesta	5	4	3	2	1
Escalable	5	4	3	2	1
Robusto	5	4	3	2	1

Tabla 5.3.1: Requerimientos del Usuario

Las columnas de la siguiente tabla evalúan el grado de satisfacción que aprecian los clientes de nuestro producto y de los de la competencia para cada uno de los requerimientos.

Requerimiento	Producto	Nivel				
Confiable	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1
Fácil de utilizar	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1
Económico	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1
Fácil de instalar	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1
Velocidad de respuesta	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1

Escalable	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1
Robusto	Nuestro Producto	5	4	3	2	1
	Elecei	5	4	3	2	1
	HAI Omni Pro II	5	4	3	2	1

Tabla 5.3.2: Comparativa del grado de satisfacción del nuestro producto respecto de la competencia

5.3 Requerimientos Técnicos

En el diseño del producto existe una serie de requerimientos técnicos propuestos según nuestra percepción de los requerimientos del usuario.

- 5.3.1 Control de dispositivos eléctricos vía X.10:** El encendido y apagado de dispositivos eléctricos se realizará a través del protocolo X.10.
- 5.3.2 Interconexión mediante RS-232:** El modulo principal se conectará con el controlador X10 mediante el uso de este protocolo.
- 5.3.3 Acceso mediante 802.11:** El acceso para el usuario final a la interfaz de control se realizara mediante Wi-Fi.
- 5.3.4 Programación vía Web:** El control de los dispositivos y el seteo de actividades programadas se realizará a través de una interfaz Web.
- 5.3.5 Cumplimiento de Estándares de seguridad:** Los dispositivos conectados a la red eléctrica cumplen los estándares de seguridad de acuerdo a la normativa vigente, así como también el acceso a la red inalámbrica se realiza en forma cifrada protegiendo los datos y accesibilidad solo a personas autorizadas.
- 5.3.6 Alcance de señales:** Determinación del alcance que tendrán las señales emitidas por el sistema y la distancia máxima desde un controlador al dispositivo controlado.
- 5.3.7 Inmunidad a Interferencias:** Determinación del nivel de interferencia producido y aceptado para el ambiente sobre el cual el sistema podrá ser instalado y se garantiza el funcionamiento normal.
- 5.3.8 Rango y precisión de sensores:** Determinación del rango y precisión de sensores que permiten el funcionamiento del sistema sin pasar por alto la detección correcta de señales de interés.

5.4 Casa de Calidad

Con los requerimientos del usuario, nuestros requerimientos técnicos y la tabla de satisfacción de dichos requerimientos de nuestro producto y los que compiten con el mismo construimos la Casa de Calidad.

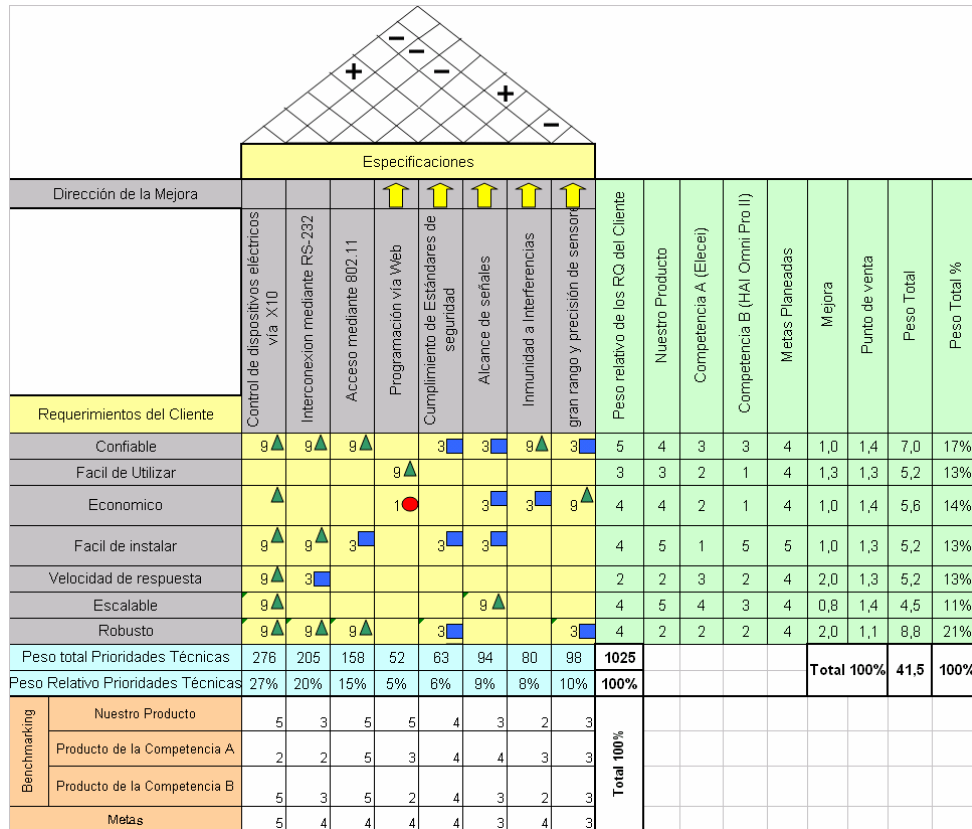


Figura 5.4.1: Casa de la calidad

Puntuación Peso relativo de los RQ del Cliente	
1	Baja importancia
5	Alta importancia

Mejoras = Planeado / Nuestro Producto

Peso Total = Pesro Relat.Cliente x Mejora x Pto.de Venta
--

Peso Total % = Peso Total / Suma Pesos Totales
--

Puntuación Mapeo RQ del Cleinte en Especificaciones	
1	Baja inter-relación
3	Media inter-relación
3	Alta inter-relación

Peso total Prioridades Técnicas = $\sum II$ Pesos Totales x Puntuación Mapeo de RQ en Espec.
--



Figura 5.4.1 Referencias de la casa de la calidad

La columna de metas planeadas establece el valor esperado al que se pretende llegar al mejorar el producto, la columna de mejora se calculará como el cociente entre el valor de las metas planeadas y la apreciación de nuestro producto.

El valor de los puntos de venta se utiliza para agregar peso o ponderar aquellos requerimientos del cliente que ayudan a promocionar el producto en el mercado. Este valor estará comprendido entre 1 y 1,5.

Por último el peso total se calcula como el producto entre el peso relativo de cada requerimiento, el factor de mejora y el valor de los puntos de venta.

El techo de la casa de calidad es una matriz de forma triangular que extiende la matriz de requerimientos del diseñador e indica la relación mutua de estos requerimientos.

En el caso de que sean independientes no habrá relación alguna mientras que en el caso de haber relación se distinguen dos casos. Puede ocurrir que dos requerimientos se opongan, esto es, que la mejora

de uno tienda a degradar a otro lo cual se indica con un signo menos (-). Si por el contrario el desarrollo de un requerimiento resulta en una mejora de otro, esto se indica mediante un signo más (+).

5.4.1 Conclusiones

Benchmark

Aquí se evalúan en un gráfico los pesos relativos de las especificaciones técnicas respecto a los requerimientos del cliente y se hace la comparativa entre nuestro producto y la competencia. En el caso que una especificación incluya datos numéricos se incluye el valor del parámetro para compararlos directamente, en caso contrario se compara el grado en que la especificación cumple el requerimiento.

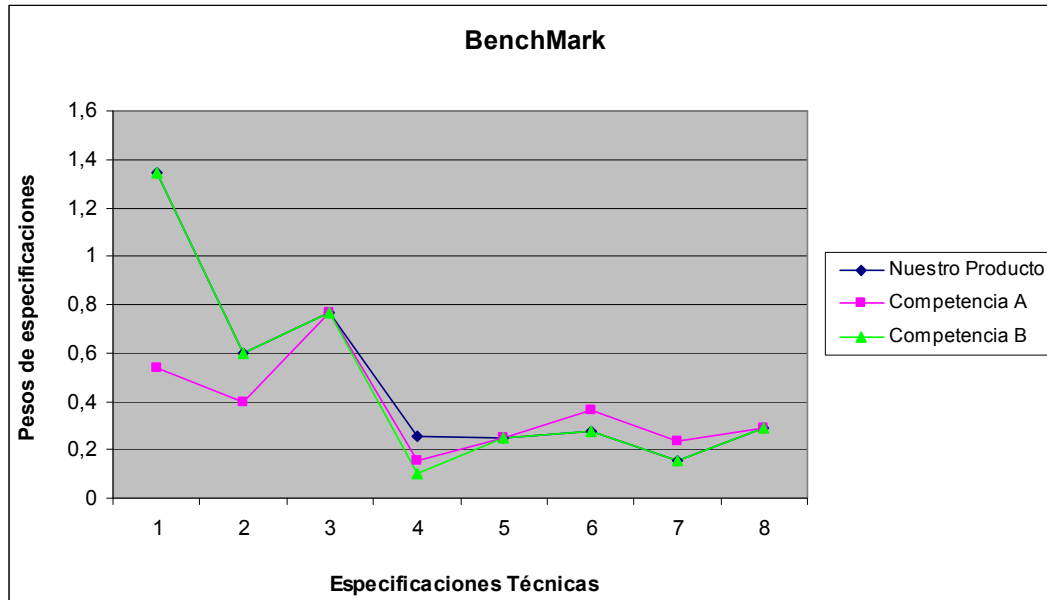


Figura 5.4.1.1: Kit sistem HAI Omni Pro II

5.4.2 Aspectos a mejorar

Teniendo en cuenta los resultados del Benchmark se pueden hacer inferencias sobre que parámetros se pueden mejorar y en cuales se puede ceder o dejar como están.

Los parámetros a mejorar son:

- **Alcance de Señales:** Se deberá garantizar el alcance de las señales Wi-Fi y X.10.
 1. Wi-Fi se especifica un alcance de 35 mts en ambientes cerrados. Utilizando antenas con ganancia mayor que la antena omnidireccional se puede solucionar el inconveniente.
 2. En el caso de X.10 se deberá asegurar que desde un controlador hasta su dispositivo controlado la señal llegará sin degradación. Para ello en lugar de tener un único controlador X.10, se dispondrá de controladores X.10 en cada receptor Wi-Fi, de forma tal que la distancia del controlador al dispositivo controlado se mantenga pequeña asegurando la no atenuación de la señal.

- **Inmunidad a interferencias:** Se deberá asegurar que ante interferencias normales causadas por dispositivos que inserten señales en el medio (línea eléctrica o radiofrecuencia) no afectará el servicio del sistema.

5.5 Definición final del producto

Basados en los resultados obtenidos de los requerimientos del usuario y del benchmark, se realiza una especificación técnica del producto a desarrollar. Para una mejor comprensión del sistema, se realiza un diagrama de bloques en donde se tienen las distintas partes del sistema. Cabe destacar que el diagrama de bloques mostrado es con un módulo controlador, pero es extensible a n módulos.

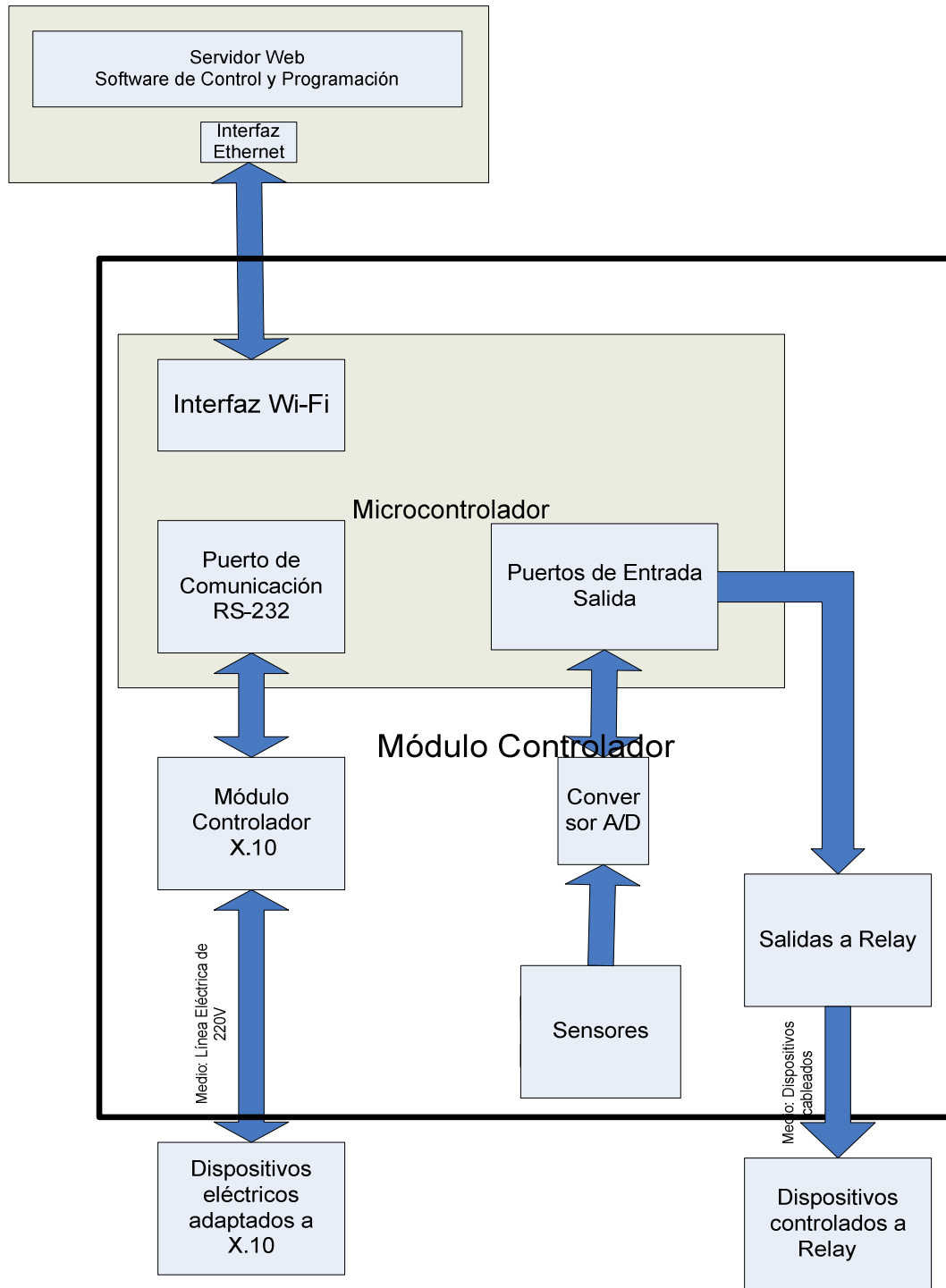


Figura 5.5.1: Diagrama de en bloques del sistema

Como se puede apreciar, el sistema consta de dos bloques bien diferenciados, que son:

1. Software de Control y programación
2. Módulo controlador

A continuación se definen las especificaciones de cada uno de ellos

5.5.1 Software de control y programación

El sistema tendrá un software de control que será el encargado de interconectarse con los distintos módulos de control y recibir la información por parte de ellos para que se visualice en una interfaz online. Además deberá permitir el acceso al sistema desde Internet, es decir desde sitios remotos. Deberá poder ser instalado en un servidor con requerimientos de procesamiento estándar del mercado actual, sin la necesidad de utilizar hardware específico de una marca. El software deberá cumplir con las características que se detallan en los puntos siguientes.

5.5.1.1 Interfaz de usuario

El usuario se conectará al sistema a través de una interfaz Web la cual deberá ser personalizada para cada cliente según los dispositivos que desee controlar y cumplirá con siguientes requisitos de accesibilidad y seguridad:

Accesibilidad y seguridad

- 1) El ingreso será utilizando un explorador de Internet tradicional y se accederá a través de la dirección IP del Server, en el caso que en la red exista un servidor de DNS se podrá acceder por nombre.
- 2) El ingreso al sistema deberá ser con verificación de usuario y contraseña utilizando autenticación por sesiones, expirando la sesión luego de un máximo de 15 minutos de inactividad, garantizando de esta forma la seguridad del sitio. Se deberán definir perfiles de usuarios que le asignarán privilegios, que consistirán tanto en la configuración del sistema como el acceso a prestaciones del sistema solo a personas autorizadas. Esto es útil para que el sistema pueda ser usado por niños y que los mismos no puedan accionar artefactos que no correspondan ser activados por ellos.
- 3) Deberá quedar un registro diario (log) de cada ingreso que realice cada usuario y de las actividades y/o modificaciones que realice en el sistema para poder realizar una auditoría en caso de falla. El log serán archivos de texto de un tamaño máximo de 1 Mbyte. Se crearán un logs diarios con el formato log_ddmmaaaa_# siendo N° el número de log comenzado por el 1. En el caso que se supere el tamaño prefijado de 1 Mbyte se creará un nuevo archivo con el # en 2 y así sucesivamente. Dentro del registro se irán colocando en orden cronológico las acciones que se realicen indicando hora de cada acción y/o acceso. Se podrá descargar desde un link en la web.

Requerimientos propios del entorno grafico:

- 1) Tendrá una sección en donde se puedan programar tareas para que se realicen en forma automática. Las mismas se definirán en la interfaz Web general y luego de salvarla se enviará un comando al módulo controlador para que el mismo la ejecute en el momento indicado. La tarea se mantendrá visible hasta que la misma haya sido ejecutada y en ese momento el módulo controlador enviará una confirmación para que sea borrada del menú. Un ejemplo de tarea programada es: “Encender el lavarropas a las 5:00 AM el día 20/10/2007”.
- 2) Poseerá un sector adicional siempre visible en donde se indican los parámetros de las habitaciones en el lugar donde se miden los mismos.
- 3) Tendrá un servicio de sincronización de hora que cumplirá el protocolo NTP (Network Time Protocol), de forma tal que se asegure que las tareas serán realizadas en el momento esperado.

5.5.2 Módulo controlador

El módulo controlador es la pieza fundamental del proyecto. Es el equipo que recibe y ejecuta los comandos para controlar artefactos eléctricos y electrónicos. Posee los sensores que miden los parámetros como por ejemplo la temperatura para ejecutar acciones correctivas de dichos parámetros. Se instalará uno por ambiente pero en el caso de existir ambientes en donde no se tenga que colocar sensores se podrá compartir el módulo para ahorrar costos. Deberá reunir las siguientes características:

- 1) Será un sistema basado en un microcontrolador con interfaz Wi-Fi. Deberá tener puertos de entrada y salida suficientes para recibir las señales de los sensores, utilizándose conversores A/D externos o incluidos en el microcontrolador. Deberá poder implementar el stack de protocolos TCP/IP. El direccionamiento IP del controlador deberá ser fijo para evitar problemas relativos al cambio de la misma. La interfaz Wi-Fi deberá estar homologada ante la CNC o en su defecto deberá cumplir las condiciones para someterla a homologación ante el mismo organismo.
- 2) La comunicación con el servidor que corre el Software de Control y Programación se realizará a través del protocolo TCP/IP. El usuario activará los comandos ingresando por Web al servidor que corre el mencionado programa y el mismo iniciará un socket TCP/IP por lo que el módulo controlador cumplirá la función de servidor escuchando en un puerto efímero por encima del 1024.
- 3) 4. El encendido de los artefactos eléctricos se realizará utilizando el protocolo de transmisión por línea eléctrica X.10. Para ello se utilizará un módulo controlador del mencionado protocolo con

interfaz RS-232 para conectarlo al microcontrolador. Los dispositivos controlados tendrán un receptor que entenderá el protocolo y energizará al artefacto (también tiene la propiedad de regular tensión eléctrica). El módulo y los receptores X.10 deberán cumplir con los estándares eléctricos de CE, teniendo el sello correspondiente. Además, se deberá prever salidas para dispositivos directamente conectados utilizando salidas acopladas por Relay.

- 4) Sensor de temperatura: Deberá entregar un valor en ° Celsius. Debido a que no se controla ningún proceso crítico, se aceptará una tolerancia alta, de hasta 1° Celsius de error. Se conectará al microcontrolador a través de un conversor analógico digital (A/D), pudiendo ser el bus paralelo o serie, dependiendo de la cantidad de entrada/salidas que tenga el último.
- 5) El tamaño del módulo completo incluyendo el dispositivo X.10 no podrá superar las medidas 20 x 20 x 10 expresadas en centímetros.
- 6) La alimentación del sistema será realizada con una fuente que cumpla las normas UL e IRAM de seguridad eléctrica.
- 7) Los módulos X.10 utilizados deberán tener los sellos UL e IRAM, en caso que se tenga que importar el producto será necesario el sello CE.
- 8) Alcance de señales: Se deberá garantizar el alcance de las señales Wi-Fi y X.10.
 1. Wi-Fi: Se utilizarán antenas con ganancia mayor a la unidad en caso que sea necesario así también como repetidores para extender la red a través de los distintos sitios del predio (jardín, distintas plantas, etc). Se debe respetar la especificación establecida por la CNC en su resolución 302-98 en que establece:
 - Potencia Máxima 1 W con la siguiente limitación:
Si la ganancia de antena supera los 6 dBi, debe reducirse 1 dB la potencia máxima del transmisor por cada 3 dB que dicha ganancia supere los 6 dBi.
 2. En el caso de X.10 se deberá asegurar que desde un controlador hasta su dispositivo controlado la señal llegue sin degradación.
- 9) Inmunidad a interferencias: Se deberá asegurar que ante interferencias normales causadas por dispositivos que inserten señales en el medio (línea eléctrica o radiofrecuencia) el servicio del sistema no se verá afectado. Para ello se deberán realizar pruebas de conexión mediante la utilización de del comando ping

tomándose como válido una recepción del 99% de los mensajes enviados sobre una serie de mínimo 1000 mensajes, asegurando de esta forma una alta confiabilidad del sistema. A su vez, se deberá asegurar que el sistema no interferirá sobre el funcionamiento normal de otros equipos. Esto último deberá ser validado por la CNC para la homologación del dispositivo Wi-Fi.

6. Análisis de Factibilidad

6.1. Elección de una Solución

6.1.1 Red de Datos

6.1.1.1 RS-232

RS-232 es una alternativa de fácil implementación disponible prácticamente en todos los microcontroladores del mercado y de bajo costo. Sin embargo presenta las siguientes desventajas:

- La comunicación es punto a punto (se requiere una computadora por controlador)
- Baja tasa de transferencia (hasta 115kbps)
- Corto alcance (segmentos de hasta 15m)

Las razones mencionadas, hacen que no sea posible utilizar RS-232, particularmente por la razón que se pretende un sistema de múltiple acceso simultáneo siendo esto imposibilitado por la naturaleza de comunicación punto a punto.

6.1.1.2 USB

Esta interfaz brinda la misma funcionalidad que RS-232 con mayores tasas de transferencia (de hasta 500Mbps). No obstante el alcance del segmento es reducido y la comunicación es punto a punto. La implementación de esta interfaz es más costosa que RS-232 y es más demandante para el controlador en cuanto a procesamiento que RS-232. Por las mismas razones anteriormente mencionadas es que USB no es aplicable.

6.1.1.3 Ethernet (IEEE 802.3)

Es una tendencia actual con origen en los años 70 y estableciéndose como estándar en los años 80. Desde entonces evolucionó en gran medida cambiando el medio de acceso de cable coaxial par trenzado. También cambió la tasa de transferencia siendo el estándar mas común actual de 10/100 Mbps (existiendo actualmente redes de 1 Gbps y 10Gbps). El uso de una red IP basada en Ethernet es muy común basado en las siguientes razones:

- Amplia extensión de este tipo de redes
- Disponibilidad de conectividad Ethernet en todas las PCs actuales
- Posibilidad de integración a redes LAN ampliamente difundidas
- Posibilidad de conectividad a través de la Internet

- Bajo costo de los equipos Ethernet
- Posibilidad de acceder al controlador desde cualquier PC conectada al mismo segmento de red que el controlador, desde otras redes o incluso desde Internet.
- Permite acceder y configurar más de un controlador desde múltiples PC.
- Alta tasa de transferencia (10Mbps)
- Largo alcance (segmentos de hasta 100m entre repetidores)

A pesar de las grandes ventajas mencionadas, Ethernet necesita un cable de comunicación, lo que dificulta su implementación respecto de sistemas inalámbricos.

6.1.1.4 Wi-Fi (IEEE 802.11b)

Es un sistema ampliamente utilizado en la actualidad y desde algunos años. 802.11 tiene su origen en los años 90 y el estándar 802.11b fue creado en 1999, pero su masiva utilización se produjo algunos años después. 802.11b tiene las mismas características descritas por Ethernet a diferencia que el medio utilizado es el espectro radioeléctrico en la banda de frecuencia de 2.4 Ghz denominada ISM, industrial, científica y médica (por sus siglas en inglés (Industrial, Scientific and Medical), reduciendo de esta forma el problema de la instalación. Las redes Wi-Fi y Ethernet normalmente conviven y se comunican unas a otras, siendo normal que el punto de acceso a la red (Access Point) esté conectado al Switch de datos logrando de esta forma la conectividad. En pequeñas instalaciones se tiene dispositivos que integran el Switch de Datos, el Enrutador para la conexión a Internet y el punto de acceso para los dispositivos Wi-Fi. El estándar sigue actualizándose existiendo actualmente la versión 802.11g (54 Mbps) y proyectándose la versión 802.11n (248Mbps) siendo todas compatibles con 802.11b.

Por las grandes ventajas de Wi-Fi se elegirá este tipo de conectividad para el controlador a pesar que esto trae aparejado mayor tiempo de desarrollo de hardware, y firmware otros sistemas más simples como por ejemplo conexión serie por RS232. Además, existen menor cantidad de microcontroladores que soporten esta característica teniendo menos opciones en la elección.

Como una ventaja adicional Wi-Fi brinda la solución más factible en cuanto a escalabilidad. La implementación de una red de aplicación permite que varias computadoras tengan acceso al sistema en forma simultánea.

6.1.2 Dispositivos Controlados

6.1.2.1 Protocolo X.10

Dado que uno de los requisitos mas importantes registrado entre los usuarios es una instalación sencilla, es que se busca minimizar los requerimientos de cableado.

Es por este motivo que el protocolo X.10 es una alternativa muy conveniente a ese fin, dado que transmite las señales de control por la red eléctrica haciendo que la instalación no necesite cableado adicional para transmitir señales de datos o modificar el cableado para intercalar algún dispositivo de control como Relays, Contactores o Triacs. Por tal motivo se utilizarán dispositivos X.10. Para controlar los dispositivos, se tienen módulos controladores, que emiten las señales y dispositivos controlados, que se intercalan con los dispositivos a controlar (sin alterar la instalación) los cuales reciben las señales y ejecutan las acciones correspondientes. Existen gran variedad de módulos controladores y especialmente de dispositivos controlados. Se presentan en esta sección los que resultaron más relevantes para el presente proyecto. Sin embargo, cualquier otro dispositivo X.10 no mencionado será 100% compatible con el sistema.

6.1.2.1.1: Módulo Controlador

Los controladores de las distintas marcas vienen preparados para trabajar con un software estandarizado de X.10 denominado Active Home, el cual presenta una interfaz gráfica en donde se agregan dispositivos al sistema y ante cada evento envía una señal por el puerto serie RS-232 que el controlador la convierte en una señal a través de la línea eléctrica. En este proyecto se prescindirá del software Active Home, debido a que el controlador se conectará al puerto RS-232 del microcontrolador quien le proporcionará las órdenes para transformarlas en señales X.10. Debido a que las normas eléctricas en la Argentina son similares a las europeas (220 v, 50 Hz.), es que los productos tendrán su origen proveniente desde Europa, y el uso de producto de otro origen estará restringido a que cumplan con las condiciones locales. Dos modelos de controladores son:

1. XTP040201
2. X-10CM11

Ambos controladores tienen características similares:

- Alimentación: 200-240 V AC.
- Señal X-10: 120 KHz \pm 2 KHz / 2,5V pp.
- Pilas de respaldo tipo: 2 x AAA 1,5 V. Se utilizan para el respaldo de información almacenada en caso de caída de corte de suministro de energía eléctrica.
- Homologación Marca CE , UL
- Conexión a puerto serie en PC mediante: cable RJ9– DB9

Se elige utilizar el X-10CM11 debido a que existe representante local en la Argentina, pero debido a la compatibilidad entre ambos se podrá cambiar en caso de ser necesario o conveniente económicamente.

6.1.2.1.2: Dispositivos Controlados

6.1.2.1.2.1: Módulo de casquillo para encendido de lámparas

1. X-10PSM04

- Responde a comandos: todas las luces apagadas/encendidas, luz encendida, luz apagada
- Programación a distancia sin direccionamiento por ruedillas disminuyendo el tamaño

2. LM 15: Características similares al X-10PSM04. No se tiene disponibilidad en el país.

Por la disponibilidad en el mercado local se eligió el X-10PSM04.

6.1.2.1.2.2: Micro tecla, utilizado para el encendido de lámparas

1. XTP130404

- Responde a comandos: todas las luces apagadas/encendidas, luz encendida, luz apagada
- Activación Manual
- Controla 1 aparato de hasta 10 A

2. X-10AR2223E

- Responde a comandos: control On/Off/Dim y todas las luces apagadas/encendidas
- 500W de potencia
- Activación manual
- Controla hasta 2 aparatos con dos teclas y comandos en forma independiente

Se eligió la línea X-10AR2223E debido a que con un solo aparato se pueden conectar hasta 2 luces mientras que con el XTP130404 se puede controlar solo una (no se encontró en la línea XTP un producto de 2 teclas). Además existe representante en la Argentina de X-10AR2223E

6.1.2.1.2.3: Módulo de artefacto eléctrico externo

1. XTP130402

- Salida On/Off.
- Responde a señales X10: control On/Off/Dim y todas las luces apagadas/encendidas
- Potencia: 3500W
- T/Carga: Artefacto

2. X-tendAR2027A

- Salida On/Off.
- Responde a señales X10: control On/Off y todas las luces apagadas/encendidas
- Disponible con ficha argentina de tres terminales planos
- Potencia: 2400W
- T/Carga: Artefacto

Se eligió la línea X-tendAR2027A debido a que viene en forma nativa con el Terminal argentina, posibilita función dimmer y además por la facilidad de tener el distribuidor local. En caso de conectar artefactos de potencias entre 2400W y 3500w se podrá utilizar el producto XTP130402 sin mayores inconvenientes.

Existe además una amplia gama de dispositivos X.10 de distintas potencias, controladores de persianas y motores etc. Se eligieron en este caso solo los estrictamente necesarios para el funcionamiento del sistema básico, como es la activación luces y artefactos eléctricos. Cabe destacar que al ser X.10 un Standard, se podrán cambiar los componentes de acuerdo a la conveniencia del momento (disponibilidad de importador, costo, etc.) sin alterar el funcionamiento del sistema.

6.1.2.2 Dispositivos acoplados por Relay

Como se mencionó en la sección anterior, el protocolo X.10 brinda la posibilidad de controlar dispositivos remotos sin la necesidad de alterar el cableado eléctrico. Sin embargo, bajo ciertas circunstancias puede resultar ventajoso por controlar dispositivos mediante la utilización de Relay. Para tal motivo se utilizaron Relays de bobina de 5V con corriente de hasta 5 A.

6.1.2 Sensor de Temperatura

Los sensores de temperatura operan en forma similar a un diodo zener, con temperatura de ruptura proporcional a un valor de tensión sobre °K. Dos sensores que se consiguen en el mercado son:

1. LM 35:

- 0,5° de precisión a 25°C
- Escala de -55° a 150°
- Factor de escala de 10mV/°K

2. LM335:

- 1° de precisión
- Escala de -40° a 100°
- Factor de escala de 10mV/°K

Se optó por el LM 35 debido a su mejor precisión, sin que esto implique una diferencia de precios considerable.

6.1.3 Conversor Analógico Digital

La temperatura y otras variables ambientales medidas en general son valores analógicos por naturaleza que para ser medido deben ser digitalizados. Para ello se utilizará un conversor analógico digital colocado a la salida del sensor. Se tienen distintas opciones, entre las que se pueden destacar los conversores serie y paralelo. Los conversores en serie envían la medición en una cadena de bits mientras que la versión en paralelo entrega la lectura en un bus de datos. Se eligieron tres integrados para el análisis:

1. ADC 0801

- Interfaz paralelo
- bit de resolución
- Alimentación simple de 5 v

2. ADC0831

- Interfaz serie por SPI
- bit de resolución
- Alimentación simple de 5 v

3. TLC548

- Interfaz serie por SPI
- bit de resolución
- Alimentación simple con rango de 3 a 6 v

Se decidió utilizar el TLC548 por tener interfaz serie, que reduce la cantidad de puertos del microcontrolador a utilizar y además simplifica el circuito de conexión del mismo. Además, tiene un mayor rango de alimentación siendo compatible con dispositivos alimentados a 3,3v que son ampliamente utilizados actualmente en el mercado.

6.1.4 Microcontrolador

Es el encargado de controlar los sensores y actuadores para controlar el sistema. Tiene cargado el firmware para llevar a cabo las acciones y ejecutar acciones programadas. Tiene que cumplir con las siguientes características:

- Interfaz Wi-Fi (con capacidad de implementar el stack TCP/IP)
- Interfaz serie RS-232 para conectar el módulo X.10
- Amplia disponibilidad de puertos de entrada/salida para conectar conversores A/D y salidas conectadas a Relay
- Disponibilidad de librerías de software TCP/IP, SPI, etc.
- Memoria flash de al menos 512 Kb para almacenar el firmware.

Teniendo en cuenta estos requisitos se proponen dos microcontroladores embebidos con la interfaz Wi-Fi, de manera de no tener que comprar un módulo aparte. Los mismos son:

1. DigiConnect Wi-EM:

- Basado en procesador ARM 7 de 32 bit a 55 MHz
 - 4 MB de Memoria Flash y 8 MB de memoria RAM
 - 2 puertos RS-232 y puertos de entrada salida
 - Kit de desarrollo disponible a uS\$400 con variedad de librerías de software.
- Valor módulo: uS\$ 250.

2. Rabbit RCM4400W RabbitCore:

- Procesador Rabbit de 8 bit a 58 MHz
- 512 KB de memoria flash y 512 KB de memoria RAM
- 4 Puertos RS 232 y 35 puertos configurables de salida
- Reloj de tiempo Real y numerosos times
- Kit de desarrollo disponible a uS\$150 con gran variedad de librerías de software. Valor del módulo uS\$80

Basado principalmente en el carácter económico se decide utilizar el RCM4400W RabbitCore, el cual cumple ampliamente con los requisitos solicitados

6.1.5 Software

Para el diseño de nuestro proyecto realizamos un estudio de las alternativas disponibles para la elección del software utilizado para implementar la interfaz grafica del sistema.

Basándonos en la idea de una interfaz capaz de ser gestionada desde cualquier dispositivo móvil o fijo independientemente de su marca tecnología o modelo optamos por una interfaz Web, luego de esta decisión definimos todos los componentes de software según:

6.1.5.1 Sistema operativo

1. Windows:

En un comienzo desarrollar nuestra plataforma de software sobre un entorno de Windows parecía en principio una alternativa fácil desde el punto de vista de desarrollo pero analizando encontramos las siguientes desventajas:

- Precio elevado de las licencias, en algunos casos superiores al costo total del hardware de la PC.
- Muy vulnerable a ataques de denegación de servicio desde Internet (nuestra interfaz de gestión puede ser accedida desde cualquier parte del mundo por Internet)
- Gestión remota poco cómoda, muy insegura y lenta en caso de tener que brindar soporte remoto a un cliente post venta.

Las razones mencionadas sumadas a la poca confianza que nos genera este sistema operativo hacen que optemos por no utilizarlo.

2. Linux:

Optamos por utilizar Linux como entorno de desarrollo por encontrarnos muy familiarizados con este tipo de sistemas operativos y respaldándonos en las siguientes características:

- Es software de libre distribución, lo que quiere decir que no hay que pagar licencias.
- Es un sistema operativo muy fiable ya que hereda la robustez de UNIX.
- Posee sólido manejo de protocolos de red y aplicaciones relacionadas.
- Documentación actualizada y muy precisa orientada al desarrollo.
- Es un sistema muy seguro, ya que al estar publicado su código fuente miles de personas velan por tu seguridad.
- Cuenta con el soporte de muchas grandes empresas como IBM, Siemens, Sun, Motorola etc.

6.1.5.2 Web Server

Habiendo elegido Linux como entorno de desarrollo para nuestra interfaz Web ahora debemos elegir la aplicación que mediante el protocolo HTTP maneje las peticiones de los navegadores alojados en los dispositivos de nuestros clientes, entregándoles los componentes de las páginas Webs que componen nuestra interfaz de gestión.

Limitando los posibles web servers a los que pueden trabajar sobre un entorno de Linux encontramos entre otros:

- Apache
- Cherokee
- Thttpd
- Lighttpd
- Boa
- Webfs

En este caso optamos por la primera opción por ser el servidor web mas usado en Internet, en el encontramos entre otras las siguientes ventajas:

- Es software libre, publicado bajo la licencia GPL (General Public License).

- Es el servidor más usado de Internet, con una utilización del 65% aproximadamente.
- La primera aparición de Apache fue en Abril de 1995, lo que asegura su buena documentación al momento del desarrollo.
- Las principales metas de su diseño son: velocidad, simplicidad, multiplataforma y facilidad del desarrollo distribuido.
- Código escrito en C, lo que garantiza su estabilidad y rendimiento.

6.2 Normas técnicas utilizadas

6.2.1 IEEE 802.11b (Wi-Fi)

El protocolo IEEE 802.11 o Wi-Fi es un estándar de protocolo de comunicaciones del IEEE que define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en una WLAN. En general, los protocolos de la rama 802.x definen la tecnología de redes de área local.

La familia 802.11 actualmente incluye seis técnicas de transmisión por modulación que utilizan todos los mismos protocolos. El estándar original de este protocolo data de 1997, era el IEEE 802.11, tenía velocidades de 1 hasta 2 Mbps y trabajaba en la banda de frecuencia de 2,4 GHz.

El término IEEE 802.11 se utiliza también para referirse a este protocolo al que ahora se conoce como "802.11 legacy." La siguiente modificación apareció en 1999 y es designada como IEEE 802.11b, esta especificación tenía velocidades de 5 hasta 11 Mbps, también trabajaba en la frecuencia de 2,4 GHz. También se realizó una especificación sobre una frecuencia de 5 GHz que alcanzaba los 54 Mbps, era la 802.11a y resultaba incompatible con los productos de la b y por motivos técnicos casi no se desarrollaron productos. Posteriormente se incorporó un estándar a esa velocidad y compatible con el b que recibiría el nombre de 802.11g. La versión final del estándar se publicó en Junio de 2007 y recoge las modificaciones más importantes sobre la definición original; incluye: 802.11a,b,d,e,g,h,i,j

En la actualidad la mayoría de productos son de la especificación b y de la g. El siguiente paso se dará con la norma 802.11n que sube el límite teórico hasta los 600 Mbps. Actualmente ya existen varios productos que cumplen un primer borrador del estándar N con un máximo de 300 Mbps (80-100 estables).

La seguridad forma parte del protocolo desde el principio y fue mejorada en la revisión 802.11i. Otros estándares de esta familia (c-f, h-j, n) son mejoras de servicio y extensiones o correcciones a especificaciones anteriores. El primer estándar de esta familia que tuvo una amplia aceptación fue el 802.11b. En 2005, la mayoría de los productos que se comercializan siguen el estándar 802.11g con compatibilidad hacia el 802.11b.

Los estándares 802.11b y 802.11g utilizan bandas de 2,4 GHz que no necesitan de permisos para su uso. El estándar 802.11a utiliza la banda de 5 GHz. El estándar 802.11n hará uso de ambas bandas, 2,4 GHz y 5 GHz. Las redes que trabajan bajo los estándares 802.11b y 802.11g pueden sufrir interferencias por parte de hornos microondas, teléfonos inalámbricos y otros equipos que utilicen la misma banda de 2,4 GHz.

Conceptos Generales

- Estaciones: computadores o dispositivos con interfaz inalámbrica.
- Medio: se pueden definir dos, la radiofrecuencia y los infrarrojos.

Punto de acceso (AP): tiene las funciones de un puente (conecta dos redes con niveles de enlace parecidos o distintos), y realiza por tanto las conversiones de trama pertinente.

- Sistema de distribución: importantes ya que proporcionan movilidad entre AP, para tramas entre distintos puntos de acceso o con los terminales, ayudan ya que es el mecánico que controla donde esta la estación para enviarle las tramas.
- Conjunto de servicio básico (BSS): grupo de estaciones que se intercomunican entre ellas. Se define dos tipos:
 1. Independientes: cuando las estaciones, se intercomunican directamente.
 2. Infraestructura: cuando se comunican todas a través de un punto de acceso.
- Conjunto de servicio Extendido (ESS): es la unión de varios BSS.
- Área de Servicio Básico (BSA): es la zona donde se comunican las estaciones de una misma BSS, se definen dependiendo del medio.
- Movilidad: este es un concepto importante en las redes 802.11, ya que lo que indica es la capacidad de cambiar la ubicación de los terminales, variando la BSS. La transición será correcta si se realiza dentro del mismo ESS en otro caso no se podrá realizar.
- Límites de la red: los límites de las redes 802.11 son difusos ya que pueden solaparse diferentes BSS.

802.11b

La revisión 802.11b del estándar original fue ratificada en 1999. 802.11b tiene una velocidad máxima de transmisión de 11 Mbit/s y utiliza el mismo método de acceso CSMA/CA definido en el estándar original. El estándar 802.11b funciona en la banda de 2.4 GHz. Debido al espacio ocupado por la codificación del protocolo CSMA/CA, en la práctica, la velocidad máxima de transmisión con este estándar es de aproximadamente 5.9 Mbit/s sobre TCP y 7.1 Mbit/s sobre UDP.

Aunque también utiliza una técnica de ensanchado de espectro basada en DSSS, en realidad la extensión 802.11b introduce CCK (Complementary Code Keying) para llegar a velocidades de 5,5 y 11 Mbps (tasa física de bit). El estándar también admite el uso de PBCC (Packet Binary Convolutional Coding) como opcional. Los dispositivos 802.11b deben mantener la compatibilidad con el anterior equipamiento DSSS especificado a la norma original IEEE 802.11 con velocidades de bit de 1 y 2 Mbps.

6.2.2 X.10

La tecnología X10 fue desarrollada hace más de 25 años por la empresa escocesa Pico Electronics, desde el inicio fue pensada como un protocolo de comunicaciones para transmitir datos utilizando como medio de transmisión la red eléctrica. El nombre se debe a que fue el décimo proyecto para el cual un grupo de ingenieros trabajaban en común.

En 1978 se presento X10 en Norte América como un sistema de control de dispositivos de enchufe, este sistema controlaba cualquier cosa que se enchufase energizando o desenergizando el mismo.

Unos años mas tarde se presentaron los módulos para interruptores de pared y después de esto muchos fabricantes de electrodomésticos y sistemas de audio presentaron sus nuevas versiones de los productos que tenían en el mercado pero ahora con la capacidad de poder ser controlados por medio de este revolucionario protocolo.

Hoy en día X10 se convirtió en un protocolo de control masivo y existen muchos fabricantes de módulos de control entre los cuales se pueden encontrar:

- Módulos de control de luces.
- Módulos de control de persianas.
- Módulos de control de puertas (apertura cierre y traba).
- Módulos de control de electrovalvulas de riego.
- Módulos para detección de intrusos.
- Módulos controladores de sistemas de calefacción y aire acondicionado centrales.

Descripción de funcionamiento

Los dispositivos X.10 se comunican entre transmisores y receptores enviando y recibiendo señales a través de la línea eléctrica. Estas señales incluyen pequeñas ráfagas de RF que representan la información digital.

Las transmisiones X.10 son sincronizadas al cruce por cero de la señal de AC de la línea eléctrica. El objetivo es transmitir dentro de los 200 microsegundos respecto del cruce. Un "1" binario es representado por una ráfaga de un milisegundo a 120 KHz mientras que un "0" binario es representado por la ausencia de señal de 120 KHz. La ráfaga de un ms debe ser transmitida 3 veces para coincidir con el cruce por cero de las 3 fases en un sistema trifásico. La figura 6.2.2.1 muestra la relación de tiempos entre las ráfagas y el cruce por cero. Un código completo de transmisión comprende 11 ciclos de la línea de transmisión. Los primeros dos ciclos contienen el código de inicio. Los próximos 4 ciclos representan el código de casa mientras que los últimos 5 ciclos representan el código de número o el código de función (encendido, apagado, todos encendidos, etc). El bloque completo de transmisión (inicio, código de casa, función) deben ser transmitidos en grupos de 2 ó 3 ciclos entre grupos de 2 códigos. Ver figura 6.2.2.2.

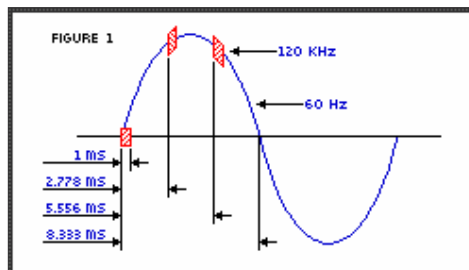


Figura 6.2.2.1: Relación de tiempos entre las ráfagas de 120 Khz

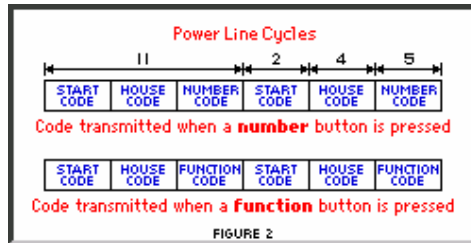


Figura 6.2.2.2: Representación de ciclos de corriente alterna

En la figura 6.2.2.3 se observa la representación de envío de la dirección de dispositivo “A2”, mientras que en la figura 6.2.2.4 se observa el código completo X.10

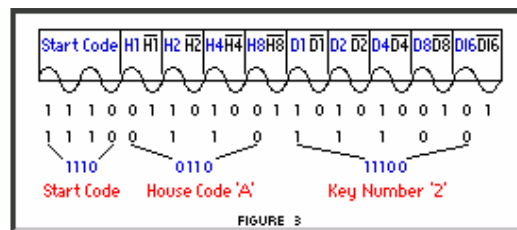


Figura 6.2.2.3: Envío de la dirección “A2”

HOUSE CODES					KEY CODES					
	H1	H2	H4	H8		D1	D2	D4	D8	D16
A	0	1	1	0	1	0	1	1	0	0
B	1	1	1	0	2	1	1	1	0	0
C	0	0	1	0	3	0	0	1	0	0
D	1	0	1	0	4	1	0	1	0	0
E	0	0	0	1	5	0	0	0	1	0
F	1	0	0	1	6	1	0	0	1	0
G	0	1	0	1	7	0	1	0	1	0
H	1	1	0	1	8	1	1	0	1	0
I	0	1	1	1	9	0	1	1	1	0
J	1	1	1	1	10	1	1	1	1	0
K	0	0	1	1	11	0	0	1	1	0
L	1	0	1	1	12	1	0	1	1	0
M	0	0	0	0	13	0	0	0	0	0
N	1	0	0	0	14	1	0	0	0	0
O	0	1	0	0	15	0	1	0	0	0
P	1	1	0	0	16	1	1	0	0	0
All Units Off					0	0	0	0	1	
All Lights On					0	0	0	1	1	
On					0	0	1	0	1	
Off					0	0	1	1	1	
Dim					0	1	0	0	1	
Bright					0	1	0	1	1	
All Lights Off					0	1	1	0	1	
Extended Code					0	1	1	1	1	
Hail Request					1	0	0	0		①
Hail Acknowledge					1	0	0	1	1	
Pre-Set Dim					1	0	1	X		②
Extended Data (analog)					1	1	0	0		③
Status-on					1	1	0	1	1	
Status-off					1	1	1	0	1	
Status Request					1	1	1	1	1	

FIGURE 4

Figura 6.2.2.4: Tabla con los códigos de casa, número de dispositivo y funciones

Fuente:

<http://www.smarthomeusa.com/info/x10theory/#theory>

6.2.3 RS-232

RS-232 (también conocido como Electronic Industries Alliance RS-232C) es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

En particular, existen ocasiones en que interesa conectar otro tipo de equipamientos, como pueden ser computadores. Evidentemente, en el caso de interconexión entre los mismos, se requerirá la conexión de un DTE (Data Terminal Equipment) con otro DTE.

El RS-232 consiste en un conector tipo DB-25 (de 25 pines), aunque es normal encontrar la versión de 9 pines (DE-9), más barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC).

Construcción física

La interfaz RS-232 está diseñada para distancias cortas, de unos 15 metros o menos, y para velocidades de comunicación bajas, de no más de 20 [Kb/s]. A pesar de ello, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de handshaking de la RS-232 se usan para resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

Si un dispositivo de los que están conectados a una interfaz RS-232 procesa los datos a una velocidad menor de la que los recibe deben de conectarse las líneas handshaking que permiten realizar un control de flujo tal que al dispositivo más lento le de tiempo de procesar la información. Las líneas de "hand shaking" que permiten hacer este control de flujo son las líneas RTS y CTS. Los diseñadores del estándar no concibieron estas líneas para que funcionen de este modo, pero dada su utilidad en cada interfaz posterior se incluye este modo de uso

6.2.4 SPI

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interface de periféricos serie o bus SPI es un estándar para controlar casi cualquier electrónica digital que acepte un flujo de bits serie regulado por un reloj

Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

Muchos sistemas digitales tienen periféricos que necesitan existir pero no ser rápidos. La ventaja de un bus serie es que minimiza el número de conductores, pines y el tamaño del circuito integrado. Esto reduce el costo de fabricar, montar y probar la electrónica. Un bus de periféricos serie es la opción más flexible cuando muchos tipos diferentes de periféricos serie están presentes. El hardware consiste en señales de reloj, data in, data out y chip select para cada circuito integrado que tiene que ser controlado. Casi cualquier dispositivo digital puede ser controlado con esta combinación de señales. Los dispositivos se diferencian en un número predecible de formas. Unos leen el dato cuando el reloj sube otros cuando el reloj baja. Algunos lo leen en el flanco de subida del reloj y otros en el flanco de bajada. Escribir es casi siempre en la dirección opuesta de la dirección de movimiento del reloj. Algunos dispositivos tienen dos relojes. Uno para capturar o mostrar los datos y el otro para el dispositivo interno.

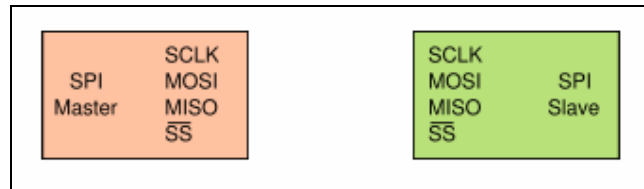


Figura 6.2.4.1: Esquema SPI master/slave.

6.3 Factibilidad de tiempos:

La planificación determina el proceso para llevar a cabo el proyecto del modo más eficiente y efectivo, teniendo como objetivo la finalización del mismo. Dentro de la planificación se pueden marcar algunos hitos importantes:

1. Definir las metas generales
2. Trazar un plan de tareas
3. Desarrollar en detalle los alcances
4. Asignar objetivos para cada actividad
5. Relacionar las actividades mediante una red lógica
6. Establecer la duración y demoras de cada actividad
7. Verificar la consistencia de la red
8. Determinar la necesidad de recursos para cada actividad

La planificación se hace usualmente sobre la base de recursos infinitos, con la finalidad de determinar el menor tiempo en que podría ejecutarse el desarrollo optimizando el uso de dichos recursos. Trabajar con recursos infinitos significa que se puede poner tanta gente a trabajar en el proyecto como sea necesario y posible, y que se cuenta además con todo el soporte que sea requerido. La variante extrema a esta opción sería hacer el desarrollo aprovechando recursos libres (y por ende tardando más tiempo).

Para poder establecer la carga de trabajo de cada etapa del proyecto, el esfuerzo global debe partirse en esfuerzos menores, y estos a su vez nuevamente deben descomponerse, de modo que se va conformando una estructura de partición del trabajo, una WBS (Work Breakdown Structure), hasta un nivel en que pueda determinarse la carga de trabajo y los recursos requeridos.

Una de las herramientas usuales para la planificación es el PERT (*“Program evaluation and review technique”* ó *“Técnica de evaluación, programación y revisión”*). El PERT es una herramienta que está orientada a eventos, y, por trabajar con tiempos aleatorios, es más apropiada para la planificación de proyectos, en los que se supone que al menos una de las tareas no se conoce ni se tiene información del tiempo que puede demorar. Esto lleva a suponer que su duración no puede ser precisada exactamente, pero, como todo proyecto requiere que sea determinado su tiempo de ejecución, es necesario contar con una herramienta que ayude para esa determinación.

6.3.1 Tareas

Para la resolución de este proyecto debemos detallar las tareas a realizar para el desarrollo del primer prototipo completamente funcional. Estas tareas, ordenadas por orden de precedencia, serán luego utilizadas para el desarrollo del PERT. A continuación se enumeran una lista de tareas junto con una breve descripción de cada una:

1. Búsqueda de información: Recopilar la información concerniente al proyecto como ser papers, notas de aplicación, componentes comunes, etc.
2. Análisis de la información: Evaluar la información recolectada y utilizarla en el desarrollo del proyecto.
3. Análisis de mercado: Buscar soluciones comerciales ya implementadas para considerar funcionalidades y costos de la competencia.
4. Propuesta del producto: Definir los lineamientos generales de la solución a la que se quiere llegar.
5. Especificaciones y reglamentaciones: Búsqueda de información sobre normas, estándares, reglamentaciones, patentes, etc.
6. Planteo de las distintas soluciones electrónicas junto con su diagrama de flujo.
7. Diagrama en bloques: Diagrama en bloques de las soluciones planteadas en la tarea anterior.
8. Elección de la solución: Sobre los resultados de la tarea anterior se selecciona la solución más adecuada.
9. Selección de componentes de hardware: Se busca en el mercado local e internacional el microcontrolador, módulos X.10, sensores y demás componentes requeridos.
10. Adquisición de los componentes de hardware: Pedido en el mercado local e internacional de los componentes seleccionados previamente.
11. Diseño del hardware del microcontrolador: Diseño del circuito del microcontrolador con su etapa de conversión de entradas y salidas.
12. Diseño de circuitos periféricos al microcontrolador: Diseño esquemático de interconexión del microcontrolador con sensores, módulos X.10, etc.
13. Diseño de software de bajo Nivel: Diseño de firmware del microcontrolador control para control de dispositivos X.10, lectura de sensores, etc.

14. Elección y descarga de los programas de software libre a utilizar: Elección del software para diseñar página Web, tareas programadas y demás tareas.
15. Desarrollo del software de alto nivel: Desarrollo del software residente en el Servidor: Página Web de acceso al sistema, e interconexión con el modulo principal.
16. Integración de los módulos de software y hardware: Verificación de que el hardware responde a los comandos enviados por el software de alto nivel. Verificación de los valores de los sensores en la página Web de acuerdo a la previa calibración.
17. Diseño del circuito impreso: Se diseñan los circuitos impresos para reemplazar kit de desarrollo y placas experimentales.
18. Armado y prueba del circuito electrónico: Armar y probar el circuito electrónico incluida placa impresa pero sin el montaje en el gabinete definitivo y sin software, solo parámetros físicos (continuidad, impedancias, etc.).
19. Diseño del gabinete: Con el circuito diseñado, y teniendo en cuenta las normas y las especificaciones, se diseña el gabinete.
20. Adquisición de componentes II: Se compran en esta etapa los componentes necesarios para el montaje definitivo.
21. Armado de prototipo: Se arma el prototipo en su totalidad.
22. Ensayo de prototipo: Se prueba el correcto funcionamiento del mismo y el cumplimiento de los valores teóricos.
23. Optimización y corrección de errores: Se optimizan y corrigen errores de software y de hardware.
24. Determinación de especificaciones finales: Según los resultados anteriores se determinan en base a las mediciones, las especificaciones precisas del prototipo.
25. Confección de la documentación: Desde el inicio del proyecto se documenta cada etapa con el mayor detalle posible. En esta etapa se escribe la documentación consistente en manuales, notas técnicas y de aplicación, documentación del proyecto detallada que permita la reproducción y mantenimiento del equipo, además del modo de uso y su correcta instalación.

6.3.2 Dependencia de las tareas

Existe una dependencia entre las tareas ya que excepto por la primera éstas requerirán que se hayan completado otras tareas para que puedan iniciarse. En la siguiente tabla se listan las diferentes tareas y la relación que existe entre ellas.

Nº	Tarea	Tareas ant.	Tareas post.
1	Búsqueda de información	Ninguna	2
2	Análisis de la información	1	3
3	Análisis de mercado	2	4 y 5
4	Propuesta del producto	3	6
5	Especificaciones y reglamentaciones	3	6
6	Estudio de posibles soluciones y diagrama de flujo de cada solución	4 y 5	7
7	Diagrama en bloques	6	8
8	Elección de la solución	7	9 y 14

9	Selección de componentes de hardware.	8	10 y 12
10	Adquisición de los Componentes de hardware.	9	11
11	Diseño del Hardware del microcontrolador	10	13
12	Diseño de circuitos periféricos al microcontrolador	9	13
13	Diseño de software de bajo Nivel	11 y 12	16
14	Elección y descarga de los programas de software libre a utilizar	8	15
15	Desarrollo del software de alto nivel.	14	16
16	Integración de los módulos de software y hardware.	13 y 15	17 y 21
17	Diseño del circuito impreso	16	18 y 19
18	Armado y prueba del circuito electrónico	17	20
19	Diseño del gabinete	17	20
20	Adquisición de componentes II	18 y 19	21
21	Armado de prototipo	20 y 16	22
22	Ensayo de prototipo	21	23
23	Optimización y corrección de errores	22	24
24	Determinación de especificaciones finales	23	25
25	Confección de la documentación	Todas (pero cada una por separado)	Ninguna

Tabla 6.3.1 Listado de tareas y relación de orden entre ellas:

6.3.3 Estimación de tiempos medios por tarea

En principio asignamos un tiempo estimado y más probable para cada tarea según experiencia previa en tareas similares o afines, luego para estimar duración por método de PERT estimaremos más de un tiempo por tarea. La siguiente tabla muestra cada una de las tareas con sus tiempos de ejecución más probable.

Nº	Tarea	Duración estimada (en días de 8 hs.)	11	Diseño del Hardware del microcontrolador	6	22	Ensayo de prototipo	2
1	Búsqueda de información	2	12	Diseño de circuitos periféricos al microcontrolador	5	23	Optimización y corrección de errores	7
2	Análisis de la información	1	13	Diseño de software de bajo Nivel	10	24	Determinación de especificaciones finales	2
3	Análisis de mercado	2	14	Elección y descarga de los programas de software libre a utilizar	2	25	Confección de la documentación	10
4	Propuesta del producto	1	15	Desarrollo del software de alto nivel.	15			
5	Especificaciones y reglamentaciones	2	16	Integración de los módulos de software y hardware.	10			
6	Estudio de posibles soluciones y diagrama de flujo de cada solución	5	17	Diseño del circuito impreso	2			
7	Diagrama en bloques	2	18	Armado y prueba del circuito electrónico	2			
8	Elección de la solución	2	19	Diseño del gabinete	3			
9	Selección de componentes de hardware.	2	20	Adquisición de componentes II	10			
10	Adquisición de los Componentes de hardware.	10	21	Armado de prototipo	3			

Tabla 6.3.2: Estimación de duración de las tareas

6.3.4 Estimación de Pert (TÉCNICA DE EVALUACIÓN, PROGRAMACIÓN Y REVISIÓN)

Para realizar la estimación de tiempos por el método de Pert, los tiempos anteriormente estimado deberán ser ajustados de alguna forma, para esto se definen los siguientes tiempos:

1. Tiempo optimista: es el tiempo en el que se puede realizar una tarea cumpliendo con los requisitos del proyecto.
2. Tiempo pesimista: es la mayor cantidad de tiempo que puede demandar la tarea en las peores condiciones.
3. Tiempo más probable: es el tiempo que llevará cada tarea en condiciones normales.

Considerando que los tiempos tienen una distribución beta normalizada podemos calcular la varianza y la media de la siguiente manera:

$$\bar{t} = \frac{t_o + 4 \cdot t_m + t_p}{6} \quad \sigma_t^2 = \left(\frac{t_p - t_o}{6} \right)^2$$

Donde:

- t_o : Tiempo optimista
 t_m : Tiempo más probable
 t_p : Tiempo pesimista
 \bar{t} : Tiempo esperado

Ahora se procede a determinar todos los valores antes mencionados para las distintas tareas del proyecto. Como se hizo en la sección 2.2, los tiempos serán medidos en días de 8 horas laborales.

Nº	Tarea	t_m	t_o	t_p	\bar{t}	σ_j^2
1	Búsqueda de información	2	1,5	3	2,1	0,1
2	Análisis de la información	1	0,75	1,5	1	0
3	Análisis de mercado	2	1,5	3	2,1	0,1
4	Propuesta del producto	1	0,5	3	1,3	0,2
5	Especificaciones y reglamentaciones	2	1,5	3	2,1	0,1

6	Estudio de posibles soluciones y diagrama de flujo de cada solución	5	3	7	5	0,4
7	Diagrama en bloques	2	1	2,5	1,9	0,1
8	Elección de la solución	2	1	2,5	1,9	0,1
9	Selección de componentes de hardware.	2	1	4	2,2	0,3
10	Adquisición de los Componentes de hardware.	10	4	14	9,7	2,8
11	Diseño del Hardware del microcontrolador	6	4	10	6,3	1
12	Diseño de circuitos periféricos al microcontrolador	5	3	6	4,8	0,3
13	Diseño de software de bajo Nivel	10	7	15	10	1,8
14	Elección y descarga de los programas de software libre a utilizar	2	1	2,5	1,9	0,1
15	Desarrollo del software de alto nivel.	15	9	19	15	2,8
16	Integración de los módulos de software y hardware.	10	6	13	9,8	1,4
17	Diseño del circuito impreso	2	1	4	2,2	0,3
18	Armado y prueba del circuito electrónico	2	1,5	4	2,3	0,2
19	Diseño del gabinete	3	1	5	3	0,4

20	Adquisición de componentes II	5	2	8	5	1
21	Armado de prototipo	3	2	5	3,2	0,3
22	Ensayo de prototipo	2	1	4	2,2	0,3
23	Optimización y corrección de errores	7	4	12	7,3	1,8
24	Determinación de especificaciones finales	4	1	6	3,8	0,7
25	Confección de la documentación	10	4	18	10	5,4

Tabla 6.3.3: Cálculo de tiempos y desvíos para PERT

El valor de tiempo total del proyecto según método PERT no es único, dado el carácter probabilístico que tiene la duración de las tareas, por lo que su determinación dependerá del nivel de riesgo α que se adopte para la estimación. Este factor es la probabilidad de que el proyecto no pueda concretarse en el tiempo que resulta de sumar todos los tiempos efectivos de las tareas del camino crítico. De esta forma, el tiempo total del proyecto según el PERT responde a la fórmula:

$$T_{\text{PROYECTO}} = \sum_j \bar{t}_j + z_{\alpha} \sigma_j$$

Donde \bar{t}_j son los tiempos efectivos involucrados en el camino crítico

correspondiente y z_{α} es el fractil de $(1-\alpha)$, el cual depende del factor de riesgo asumido. De todos los resultados posibles, el mayor será el que determine el camino crítico y por ende la duración estimada del proyecto según método PERT.

Nº Camino	Secuencia de Tareas	$\sum_j \bar{t}_j$	σ_j	T_{PROYECTO} $\alpha=25\%$	T_{PROYECTO} $\alpha=10\%$	T_{PROYECTO} $\alpha=5\%$
1	1,2,3,4,6,7,8,9,10,11,13,16,17,19,20,21,22,23,24,25	90,7	18,06	93,7702	94,6732	97,924
2	1,2,3,5,6,7,8,9,10,11,13,16,21,22,23,24,25	81,28	14,93	83,8181	84,5646	87,252
3	1,2,3,5,6,7,8,9,10,11,13,16,17,18,20,21,22,23,24,25	91,45	19,29	94,7293	95,6938	99,166
4	1,2,3,5,6,7,8,14,15,16,17,18,20,21,22,23,24,25	78,79	17,2	81,714	82,574	85,67
5	1,2,3,5,6,7,8,9,12,13,16,,17,18,20,21,22,23,24,25	79,53	18,99	82,7583	83,7078	87,126

Tabla 6.3.4: Estimación de duración para los distintos caminos utilizando el método PERT

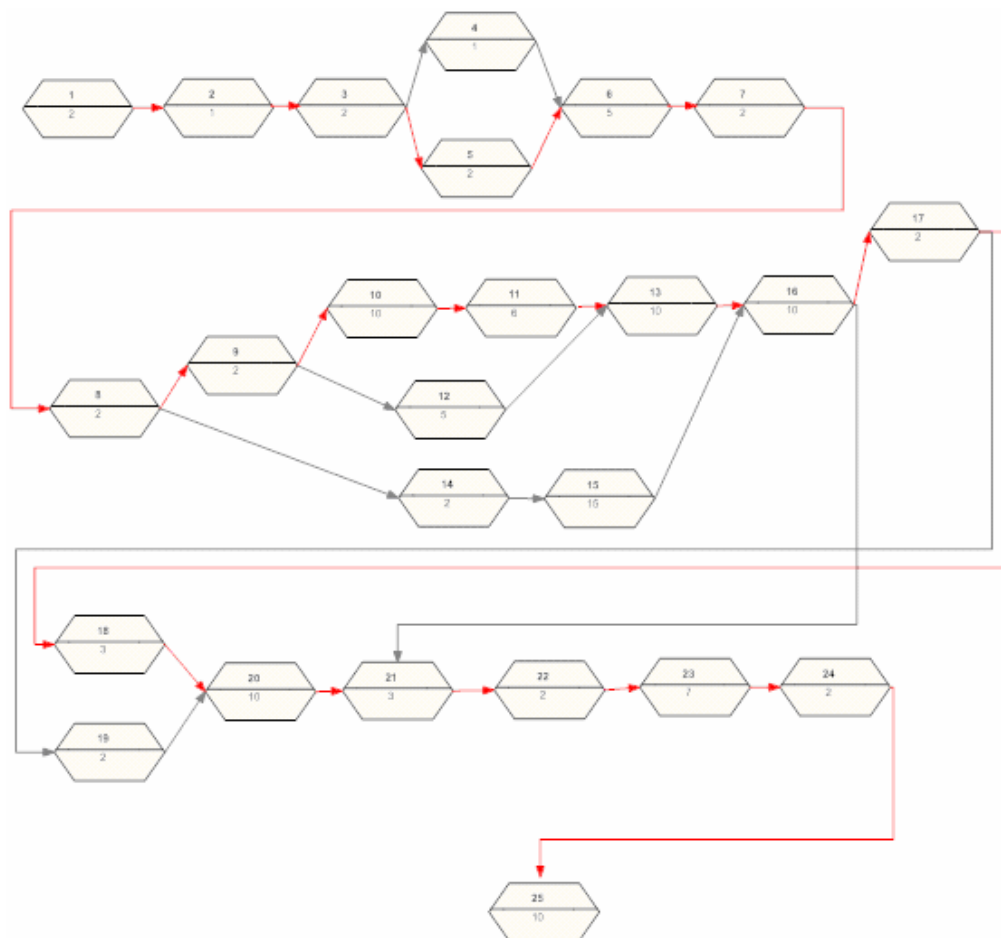


Figura 6.3.1: Diagrama de tarea utilizado en PERT. En rojo se observa el camino crítico

6.3.5 Simulación de Montecarlo

En la simulación de Monte Carlo se estima nuevamente la duración de cada tarea mediante la función Beta Inversa definida entre 0 y 1. Para poder caracterizar a la función Beta Inversa para cada una de las tareas debemos obtener los parámetros α y β . Estos parámetros se calcularon como:

$$\alpha = 1 + 4 \frac{(\bar{t} - t_o)}{(t_p - \bar{t})} \quad \text{y} \quad \beta = 1 + 4 \frac{(t_p - \bar{t})}{(t_p - t_o)}$$

Una vez que tenemos estas formulas calculamos el α y β para cada tarea, obteniéndose los siguientes valores:

Nº	Tarea	t_o	t_M	t_p	α	β
1	Búsqueda de información	1,5	2	3	2,08	0,06
2	Análisis de la información	0,75	1	1,5	1,04	0,02
3	Análisis de mercado	1,5	2	3	2,08	0,06
4	Propuesta del producto	0,5	1	3	1,25	0,17
5	Especificaciones y reglamentaciones	1,5	2	3	2,08	0,06
6	Estudio de posibles soluciones y diagrama de flujo de cada solución	3	5	7	5,00	0,44
7	Diagrama en bloques	1	2	2,5	1,92	0,06
8	Elección de la solución	1	2	2,5	1,92	0,06
9	Selección de componentes de hardware.	1	2	4	2,17	0,25
10	Adquisición de los Componentes de hardware.	4	10	14	9,67	2,78
11	Diseño del Hardware del microcontrolador	4	6	10	6,33	1,00
12	Diseño de circuitos periféricos al microcontrolador	3	5	6	4,83	0,25
13	Diseño de software de bajo Nivel	7	10	15	10,33	1,78
14	Elección y descarga de los programas de software libre a utilizar	1	2	2,5	1,92	0,06
15	Desarrollo del software de alto nivel.	9	15	19	14,67	2,78
16	Integración de los módulos de software y hardware.	6	10	13	9,83	1,36
17	Diseño del circuito impreso	1	2	4	2,17	0,25
18	Armado y prueba del circuito electrónico	1,5	2	4	2,25	0,17
19	Diseño del gabinete	1	3	5	3,00	0,44
20	Adquisición de componentes II	2	5	8	5,00	1,00
21	Armado de prototipo	2	3	5	3,17	0,25
22	Ensayo de prototipo	1	2	4	2,17	0,25
23	Optimización y corrección de errores	4	7	12	7,33	1,78

24	Determinación de especificaciones finales	1	4	6	3,83	0,69
25	Confección de la documentación	4	10	18	10,33	5,44

Tabla 6.3.4: valores de α y β para las distintas tareas

Con los valores de α y β podemos realizar estimaciones estadísticas de la duración de las tareas. Para esto utilizamos la inversa de la función Beta en una hoja de cálculos de Excel como:

$$DISTR.BETA.INV(k; \alpha; \beta; t_o; t_p)$$

Donde:

k : valor aleatorio entre 0 y 1

α y β : los correspondientes para dicha tarea

t_o y t_p : los correspondientes para dicha tarea

Con esta función se obtuvieron 255 posibles tiempos para cada tarea. Estos valores fueron agrupados de a grupos de 22 (un valor por tarea) y se calcularon los tiempos de cada camino posible, para así buscar el camino crítico dentro de ese grupo.

Así es como se determinaron 255 caminos críticos, entre los 6 caminos posibles, y sus tiempos de duración. Los porcentajes de aparición de los caminos posibles como caminos críticos y sus tiempos fueron:

Nº camino	Secuencia de tareas	Porcentaje de aparición	Tiempo medio [días]
1	1,2,3,4,6,7,8,9,10,11,13,16,17,18,20,21,22,23,24,25	5,1%	89,75
2	1,2,3,5,6,7,8,9,10,11,13,16,21,22,23,24,25	0%	-
3	1,2,3,5,6,7,8,9,10,11,13,16,17,19,20,21,22,23,24,25	94,9%	91,4
4	1,2,3,5,6,7,8,14,15,16,17,18,20,21,22,23,24,25	0,00%	-
5	1,2,3,5,6,7,8,9,12,13,16,,17,18,20,21,22,23,24,25	0,00%	-

Tabla 6.3.5: porcentaje de aparición de cada camino como camino crítico según Monte Carlo

Como se puede apreciar en la tabla Nº 6, vemos que el camino Nº 3 representa el camino crítico el 94,9%% de los casos. Además, el camino Nº 3 difiere en solo dos tareas con el camino Nº 3, por lo que podemos inferir que no existen caminos semicríticos.

En la figura Nº 3 podemos observar un histograma con la ocurrencia de los tiempos medios para los caminos críticos así como su porcentaje acumulativo de ocurrencia, lo cual es extremadamente útil a la hora de determinar el tiempo de duración del proyecto para distintos márgenes de confianza.

Los valores que generaron el gráfico de la figura Nº 3 son los de la tabla Nº 7

Bin	Frequency	Cumulative %
70	0	0,00%
71	0	0,00%

72	0	0,00%
73	0	0,00%
74	0	0,00%
75	0	0,00%
76	0	0,00%
77	1	0,39%
78	2	1,18%
79	0	1,18%
80	0	1,18%
81	2	1,96%
82	1	2,35%
83	6	4,71%
84	1	5,10%
85	7	7,84%
86	12	12,55%
87	10	16,47%
88	17	23,14%
89	19	30,59%
90	15	36,47%
91	20	44,31%
92	13	49,41%
93	24	58,82%
94	21	67,06%
95	13	72,16%
96	16	78,43%
97	13	83,53%
98	12	88,24%
99	7	90,98%
100	5	92,94%
101	4	94,51%
102	9	98,04%
103	1	98,43%
104	3	99,61%
105	0	99,61%
106	0	99,61%
107	1	100,00%
108	0	100,00%
109	0	100,00%
110	0	100,00%
111	0	100,00%
112	0	100,00%
113	0	100,00%

114	0	100,00%
115	0	100,00%
116	0	100,00%
117	0	100,00%
118	0	100,00%
119	0	100,00%
120	0	100,00%
More	0	100,00%

Tabla 6.3.6: valores generadores del histograma

De la tabla 6.3.6 podremos definir los tiempos totales del proyecto para distintos márgenes de confianza. De esta manera obtenemos que, según la simulación de Monte Carlo, los tiempos y sus márgenes son:

Margen de confianza	(aprox.) $T_{PROYECTO}$
75%	95 días
90%	101 días
95%	104 días

Tabla 6.3.7: Duración de proyecto de acuerdo a margen de confianza

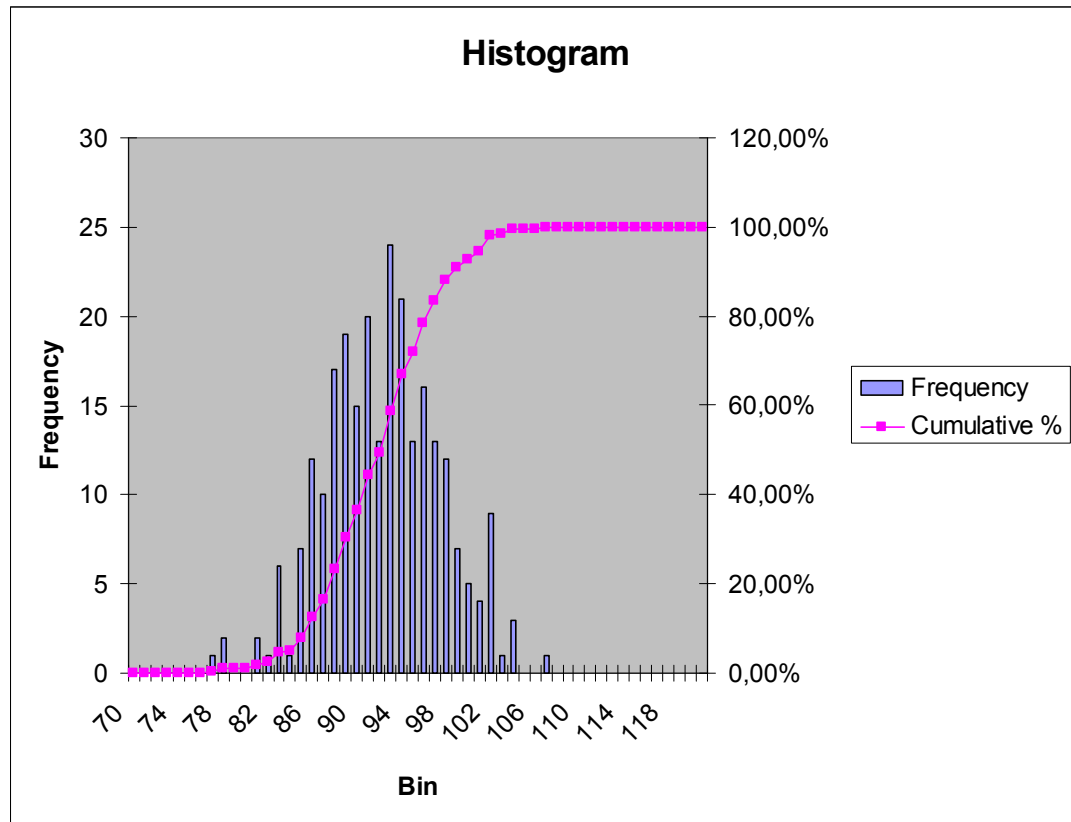


Figura 6.3.2: Histograma con valires acumulados de duración de proyecto

6.3.6 Comparación de resultados:

Los resultados de tiempo total del proyecto estimados según los distintos métodos no son iguales entre sí, como era de esperarse. Comparemos estos valores en una tabla:

Método	Nº camino crítico más probable	T_{PROYECTO}	T_{PROYECTO} $\alpha = 25\%$	T_{PROYECTO} $\alpha = 10\%$	T_{PROYECTO} $\alpha = 5\%$
CPM	3	88 días	-	-	-
PERT	3	89,29 días	94,72 días	95,69 días	99,166 días
Monte Carlo	3	89,54 días	95 días	101 días	104 días

Tabla 6.3.8: comparación de los resultados de CPM, PERT y simulación Monte Carlo

Como puede verse en la tabla N° 9, el método de CPM arrojó un tiempo medio menor a los calculados por PERT y Monte Carlo. Estos dos últimos valores son bastante parecidos entre sí. También podemos ver que para los distintos factores de riesgo (lo cual se puede traducir a niveles de confianza) los valores calculados con PERT y Monte Carlo son relativamente del mismo orden, pero siendo los valores de PERT más cercanos entre sí.

Con esto podemos ver que no existe mucha diferencia en considerar los valores calculados por PERT o por Monte Carlo, ya que la diferencia entre ellos es mínima.

Otra similitud importante entre los distintos métodos es el hecho de que el camino crítico es siempre el mismo, el número 3, conformado por las tareas 1,2,3,5,6,7,8,9,10,11, 13,16,17,19,20,21,22,23,24,25. Finalmente podemos aclarar que el método de CPM sólo arroja un valor ya que se supone que los tiempos son determinísticos, por lo que los conceptos de factor de riesgo y margen de confianza no tienen sentido en este caso.

6.3.7 Asignación de recursos

Antes de poder realizar el diagrama de Gantt, debemos asignar los recursos humanos necesarios para cada tarea. En la tabla N° 10 se detallan las asignaciones para cada tarea del proyecto.

Los recursos asignados corresponden a ingenieros o técnicos electrónicos con distintos perfiles. Estos perfiles podrán ser comercial o bien programador. Para

realizar la asignación se tuvo en cuenta el camino crítico encontrado en los distintos métodos (camino N° 3) para así reforzar los recursos de dicho camino y poder no ser tan estrictos asignando recursos para las otras tareas.

En la tabla 6.3.9 se utilizan las siguientes abreviaciones:

Ing.: ingeniero. E (Electrónico). S (Sistemas)

Tec.: técnico

P.C.: perfil comercial

Prog: programador

N°	Tarea	Recursos asignados
1	Búsqueda de información	2 Ing. (E y S)
2	Análisis de la información	2 Ing. (E y S)
3	Análisis de mercado	1 Ing. (P.C.)
4	Propuesta del producto	2 Ing. (E y S)
5	Especificaciones y reglamentaciones	2 Ing. (P.C y E)
6	Estudio de posibles soluciones y diagrama de flujo de cada solución	2 Ing. (E y S)
7	Diagrama en bloques	2 Ing. (E y S)
8	Elección de la solución	2 Ing. (E y S)
9	Selección de componentes de hardware.	1 Ing. E
10	Adquisición de los Componentes de hardware.	1 Ing. E
11	Diseño del Hardware del microcontrolador	1 Ing. E
12	Diseño de circuitos periféricos al microcontrolador	1 Ing. E
13	Diseño de software de bajo Nivel	2 Ing. E Prog
14	Elección y descarga de los programas de software libre a utilizar	1 Ing. S Prog
15	Desarrollo del software de alto nivel.	1 Tec. Prog y 1 Ing. Prog
16	Integración de los módulos de software y hardware.	2 Ing. (E y S)
17	Diseño del circuito impreso	1 Tec E.
18	Armado y prueba del circuito electrónico	1 Tec E y 1 Ing. E
19	Diseño del gabinete	2 Ing. E
20	Adquisición de componentes II	1 Ing. E
21	Armado de prototipo	1 Ing. E
22	Ensayo de prototipo	2 Ing E.
23	Optimización y corrección de errores	2 Ing. (E y S)
24	Determinación de especificaciones finales	1 Ing
25	Confección de la documentación	2 Ing. (E y S)

Tabla N° 6.3.9: asignación de recursos para las distintas tareas del proyecto

6.3.8 Diagrama de Gant

Luego de haber hecho la asignación de recursos podemos construir el diagrama de Gantt. Para realizar esto se utilizó la herramienta Microsoft Project, dando por resultado el diagrama que se muestra en la siguiente figura:

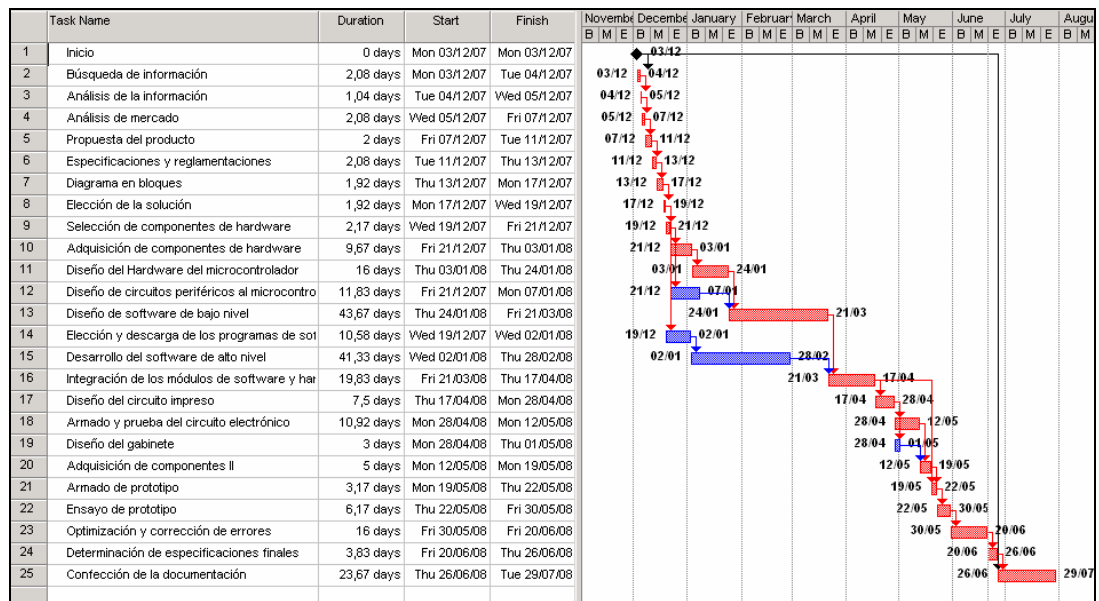


Figura 6.3.4: Diagrama de Gant

6.4 Factibilidad Económica

Todo proyecto que se desarrolle comercialmente debe estar incluido en un plan de negocios y ser factible económicamente. En esta sección se realiza un estudio de mercado el cual representa una parte importante del proyecto.

Nuestro producto está orientado a usuarios con un poder adquisitivo alto que deseen poseer un sistema de domotica en sus hogares nuevos o previamente construidos.

Es necesario tener una idea estimada de la dimensión del mercado antes de analizar su factibilidad económica.

Para este caso de estudio se toman como clientes potenciales a aquellos hogares calificados del tipo CALMAT I, esto es de mayor categoría de construcción y con conexión a Internet. Estos indicadores aseguran que se trata de usuarios con suficiente poder adquisitivo y que además son capaces de entender y operar el software y el sistema. Según el Censo 2001 del INDEC la cantidad de hogares con estas características es de 822.000. Se pretende lograr inicialmente una penetración en este segmento de un pequeño porcentaje. El principal problema para esto es lograr una buena percepción del producto en los usuarios y motivarlos a adquirirlo. Debido al reducido mercado de la automatización de hogares en la Argentina los productos de este tipo son percibidos como novedad lo cual es un punto fuerte para la estrategia de marketing.

6.4.1 Competencia

En secciones anteriores se analizaron productos similares los cuales representan posible competencia. No obstante estos productos no se comercializan masivamente en el mercado argentino.

Nuestro producto apunta a un mercado específico lo que nos permite hacer foco sobre los requerimientos de este pequeño nicho de mercado y poder poner énfasis sobre los mismos sin tener en cuenta la repercusión en el precio final del producto.

6.4.2 Insumos

Nuestro producto depende de solo dos componentes irremplazables en cuanto a marca y modelo, estos son el microcontrolador Rabbit4000 y el controlador X10 Cm11a.

Los demás componentes utilizados se consiguen en el país, y puede variar su marca según disponible de stock.

6.4.3 Puntos de venta

El nivel de customizacion aplicable a nuestro producto post venta sumado a la poca penetración en el mercado argentino de sistemas de domotica son determinantes para el tipo de comercialización a escoger.

En principio adoptamos el sistema de venta directa, de este modo además de vender el producto es posible ofrecer servicios de ingeniería, implementación y soporte post venta del producto.

Este tipo de venta deja el mayor margen de ganancia ya que se minimiza la cantidad de intermediarios y se aprovecha al máximo el know-how de los recursos asignados al producto.

6.5 Estrategia Competitiva y Plan de Acción

Se pretende crear una marca innovadora en el área de electrónica y tecnología domótica.

Esta marca debe ser recordada como parte de una infraestructura de categoría alta dentro de una edificación, además de ser percibida por los clientes como un dominador del mercado local que marca tendencias en ideas y confort. Esto debe alcanzarse en el primer año de vida de la marca y los medios para lograrlo serán una fuerte integración de últimas tecnologías y un fuerte enfoque a productos llamativos.

6.5.1 Enunciado de los objetivos.

Son objetivos primarios a corto plazo:

En principio adoptamos el sistema de venta directa, de este modo además de vender el producto es posible ofrecer servicios de ingeniería, implementación y soporte post venta del producto.

Este tipo de venta deja el mayor margen de ganancia ya que se minimiza la cantidad de intermediarios y se aprovecha al máximo el know-how de los recursos asignados al producto.

Se pretende crear una marca innovadora en el área de electrónica y tecnología domótica.

Esta marca debe ser recordada como parte de una infraestructura de categoría alta dentro de una edificación, además de ser percibida por los clientes como un dominador del mercado local que marca tendencias en ideas y confort. Esto debe alcanzarse en el primer año de vida de la marca y los medios para lograrlo serán una fuerte integración de últimas tecnologías y un fuerte enfoque a productos llamativos.

Son objetivos primarios a corto plazo:

- Obtener un market share del 10% al final del primer año de operación. Alcanzar el (BEP) **break even point** en el mismo lapso.
- Obtener un margen bruto de facturación del 30%.
- Crear una imagen de empresa innovadora de capitales intelectuales locales independientemente del producto introducido al mercado.
- Ser percibidos por los clientes como líder del mercado al cabo de 2 años.

6.5.2 Enunciado de las estrategias.

Para lograr nuestra misión es imprescindible cumplir todos y cada uno de los objetivos propuestos. Para ello llevará a cabo la siguiente estrategia:

- Generar una fuerza de ventas motivada y comprometida con la misión. El grupo de ventas se conformará jerárquicamente por vendedores experimentados, donde se incluirán personas clave de la competencia, y jóvenes que aporten talento y empuje.
- A partir de una campaña publicitaria diagramada por un estudio experimentado aplicando técnicas modernas como publicidad no pautaada, o publicidad no tradicional se impondrá una marca innovadora, nacional y de alta calidad.
- En la estrategia publicitaria se pondrá énfasis siempre en la marca. Se potenciará la característica de industria nacional y de tecnología local. Las pautas serán en medios gráficos masivos y especializados entre los cuales incluiremos periódicos y revistas de construcción. En una primera fase se atacará el mercado de la capital federal y el conurbano. Una vez que el éste mercado muestre signos de éxito mensurados según la aceptación del producto y el número de puntos de venta obtenidos se iniciará la segunda fase. En esta fase se insertará el producto en las principales ciudades del interior del país.

6.6 Descripción del plan Comercial

6.6.1 Posicionamiento Competitivo del Producto.

Teniendo en cuenta las fortalezas y debilidades de nuestro producto frente a la competencia y el poco desarrollo de la domótica en el país lo que se busca es crear un producto que sea percibido como innovador en cuanto marca, tecnología, calidad y prestaciones.

6.6.2 Precio

A partir de la estructura de precios y de la percepción que se piensa obtener del mercado como marca se realizó un análisis de rentabilidad que se muestra en la sección subsiguiente.

Se plantea obtener un margen bruto de ganancia del 30% sobre el producto sin tener en cuenta los servicios de consultaría pre y post venta.

Según ya se vio en el análisis de la competencia, ésta está compuesta primariamente por equipos importados los cuales son costosos por el tipo de cambio vigente en la Argentina. Para diferenciarnos en vez de desarrollar un producto de bajo costo nos concentramos en las funcionalidades y el servicio post venta.

Como resultado del análisis posteriormente detallado se llegó a la conclusión de que el precio del producto deberá ser de \$4800 por unidad sin tener en cuenta gastos de instalación.

6.6.3 Distribución y Venta.

Aquí se presenta la estrategia de venta y distribución, como se menciono anteriormente la modalidad adoptada es del tipo venta directa a través de contacto telefónico y posterior visita de vendedores domiciliarios quienes asesoran al cliente según el proyecto de interés.

Los vendedores luego de concretar la venta realizan la ingeniería de cliente y pasan el proyecto al departamento de implementaciones quienes serán los responsables de la solución hasta el momento que finalice la implementación.

6.7 Comunicación.

En esta sección se describe la estrategia a implementar en lo que se refiere a publicidad y promoción tanto del producto como de la marca a introducir. Se hace mención de los métodos con los cuales se dará a conocer el producto y la marca así también como el monto del presupuesto asignado por el período de cuatro años luego del cual se estima que se completará el ciclo de vida del producto.

6.7.1 Medios utilizados.

Como ya se mencionó anteriormente, se contará con la colaboración de un estudio de publicidad experimentado y moderno que mediante técnicas no convencionales serán los encargados de empujar las ventas y posicionar a la marca.

La estrategia se dividirá en medios masivos gráficos y radiales, y en publicidad estática. Como medios masivos gráficos se pautará en revistas de abonados de televisión por cable y revistas de construcción ya que se consideró que éstos representan el target de mercado al cual se apunta. Con esto se apunta al mercado de usuarios particulares.

6.7.2 Presupuesto asignado.

Para la campaña publicitaria se ha destinado una importante porción de los costos catalogados como fijos de administración. Se ha planificado invertir \$ 5000/mes en pautas publicitarias y promociones.

6.8 Descripción del plan de operación.

Aquí se presenta la estructura organizativa necesaria para el funcionamiento de la empresa y se detallan los gastos operativos mensuales para el primer año de operación.

6.8.1 Descripción de Gastos Operativos.

Al momento de efectuar un análisis de factibilidad económica se debió fijar un monto de costos de operación. Dentro de esta categoría se encuentran los gastos operativos. En esta sección se describe la composición de los gastos operativos mensual durante el primer año de operación.

Empleados:

Función	Cantidad	Sueldo Bruto(\$)	Total(\$)
Jefe de Ingeniería	1	5000	5000
Ingenieros I+D	2	3700	7400
Operarios	2	1700	3400
Vendedores(ing. de Clientes)	2	4000	8000
Total Empleados	7		23800

Tabla 6.8.1: Empleados

Gastos operativos totales tomando como base una producción estimada de 20 unidades mensuales:

Concepto	Monto
Alquiler oficinas + show room	2200
Alquiler fabrica + deposito	1700
Mantenimiento fabrica	1000
Movilidad	1500
Impuestos oficina	650
Impuestos fabrica	750
Insumos/Materia Prima(20/mes)	30000
Sueldos del Personal	23800
Publicidad	5000
Total Costos Operativos Mensuales(\$)	66600

Tabla 6.8.2: Costos Operativos

6.8.2 Parámetros de Calidad del Servicio.

Se han identificado dos parámetros como indicativos de la calidad del servicio de valor agregado prestado junto a la venta de una unidad. A continuación se mencionan.

Garantía: En la primera sección de ese informe se analizaron las características de los productos catalogados como competidores directos de nuestro producto. Allí se encontró que los productos similares ofrecidos por nuestros competidores no cuentan con una garantía seria y transparente, la mayoría ni siquiera ofrece garantía. Por esto se decidió ofrecer una garantía total por un período de 12 meses a partir de la fecha de compra que se resolverá efectuando un recambio de la unidad por una nueva o reacondicionada a nuestro criterio y conveniencia.

Entrega Inmediata: En el mercado actual la demanda de los clientes indica que no se los satisface solamente entregando un producto de calidad. Además de esto, los clientes exigen un servicio de valor agregado a la altura del producto. Para satisfacer esta demanda del mercado y crear la percepción del cliente de que somos una empresa robusta, seria y brindada a la satisfacción del cliente haremos hincapié en la velocidad de entrega de la solución completa dentro de la semana de concretado el pago del mismo.

6.9 Descripción plan de negocios.

A continuación se resumen en tablas y gráficos la evolución comercial del proyecto según el plan de negocios propuesto.

Tabla: proyección unidades vendidas por año de operación.

Año de operación	1	2	3	4	Ganancia \$/un
Cantidad de unidades vendidas	240	380	290	200	1440
Total beneficio por año(\$)	345600	547200	417600	288000	

Tabla 6.9: Plan de Negocios

La ganancia no se mantiene constante a lo largo de la vida útil del producto, según nuestro plan de ventas la ganancia debería variar a lo largo del tiempo de la siguiente manera:

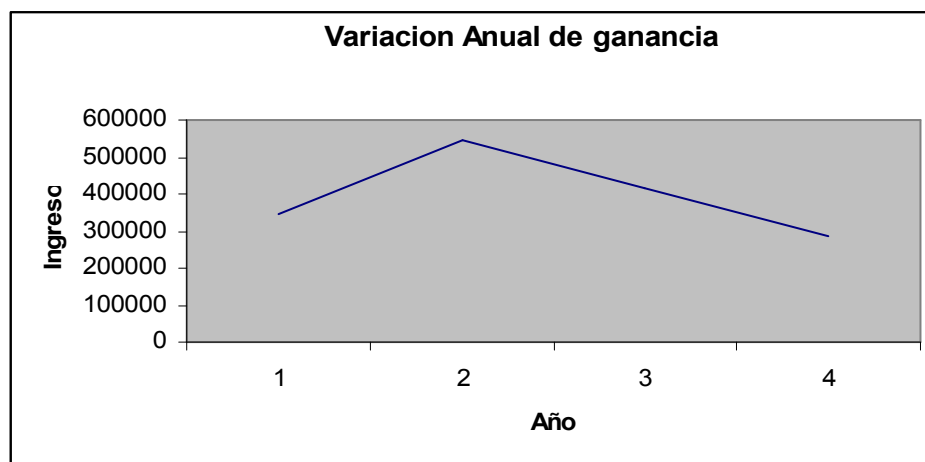


Figura 6.9.1: Variación anual de la ganancia

Los fondos representan una métrica del éxito del producto y la buena gestión económica de la empresa, en la siguiente tabla se detalla la evolución de los fondos a lo largo de la campaña propuesta.

Año de operación	0	1	2	3	4	Precio de venta(\$ miles)
Inversión						
Amortizable(\$ miles)	100					
Operativa(\$ miles)	50					
Total Inversión inicial(\$ miles)	150					
Cantidad de unidades vendidas	Sin Op.	240	380	290	200	4,8
Ingresos por ventas(\$ miles)	Sin Op.	1152	1824	1392	960	
Costos totales(\$ miles)	Sin Op.	-799,2	-799	-480	-300	
Beneficio Bruto(\$ miles)	Sin Op.	352,8	1025	912	660	
Amortización(\$ miles)	Sin Op.	-25	-25	-25	-25	
Utilidad Bruta	Sin Op.	327,8	999,8	887	635	
Flujo de fondos	-150	177,8	1178	2065	2699,6	

Tabla 6.9.1: Evolución de los fondos

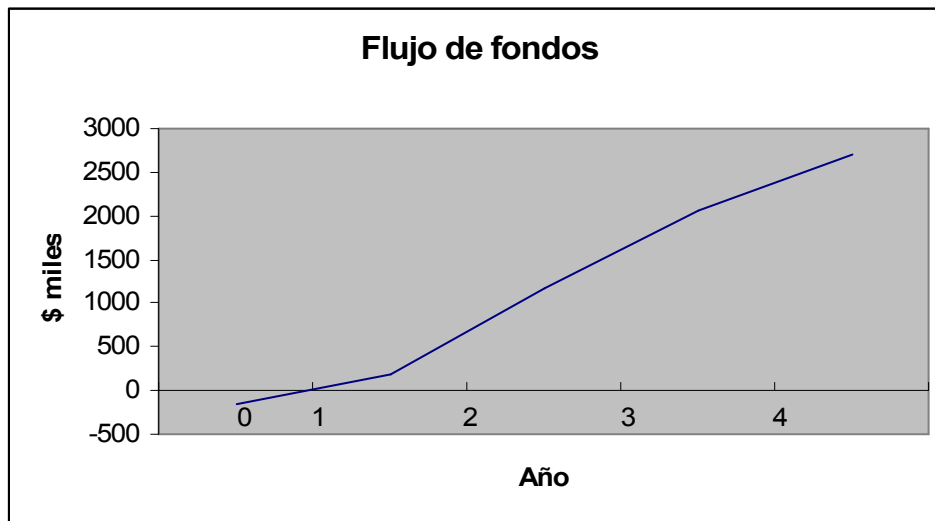


Figura 6.9.1: Evolución de los fondos

Debido a los riesgos comerciales que presupone el desarrollo de un proyecto de electrónica éste debe producir un beneficio mayor a los intereses que se percibirían si se colocara el capital de inversión en un Banco. El beneficio es función de la tasa de interés, tomando esta tasa con un valor de 12% anual calculamos el valor actualizado neto del proyecto y la tasa interna de retorno. Según los valores obtenidos nos aseguramos de que el proyecto es rentable a la tasa de interés propuesta.

Año de operación	0	1	2	3	4	Acumulado(\$ miles)
Flujo de fondos	-150	177.8	1178	2065	2699.6	
Utilidad Bruta	Sin Op	327.8	999.8	887	635	
VAN (tasa 12% anual)						1974.62
TIR						318%

Tabla 6.9.2: Rentabilidad del proyecto

6.10 Factibilidad Legal y responsabilidad civil

6.10.1 Estudio de Patentes:

- Sistema X.10: Es el sistema utilizado para controlar los dispositivos. El sistema es licenciado, pero con la adquisición de los dispositivos controladores y terminales se está adquiriendo la licencia del protocolo.
- Sistemas Domóticos: Existen distintas patentes que engloban todo un sistema de domotica de hogar, incluyendo los protocolos utilizados así como también las interfaces asociadas a cada uno.

6.10.2 Normativa:

- Seguridad eléctrica de X.10: Los productos X.10 vienen con el sello CE, es decir que se encuentran habilitados por la Union Europea debido a que es Europa quien mayormente comercializa dicho productos.
- IEEE 802.11bWi-Fi: Para la realización del proyecto se requiere un punto de acceso norma 802.11 b/g (Access Point) el cual se puede adquirir en el mercado local y se deberá asegurar que el mismo se encuentre autorizado por la CNC. Se deberá también tramitar el permiso para operar en el espacio radioeléctrico a través del Centro de Atención al Usuario del Espectro Radioeléctrico (CAUER) de la CNC (Comisión Nacional de Comunicaciones). También se deberán homologar los terminales que utilicen la norma IEEE 802.11b que sean importados y no hayan sido homologados previamente por la CNC.
- Transmisión de señales a través de la red eléctrica: No existe una normativa en IRAM. Se deberá seguir la normativa europea UNE-EN 50065-1:2002 que detalla: “*Transmisión de señales por la red eléctrica de baja tensión en la banda de frecuencias de 3 kHz a 148,5 kHz. Parte 1: Requisitos generales, bandas de frecuencia y perturbaciones electromagnéticas.*”

6.10.3 Responsabilidad Civil:

Durante el desarrollo de la solución se debe presetar especial atención de cometer errores que puedan volver inseguro el uso del equipo. Para ello se prevén todas las situaciones para no incurrir en errores por omisión y en caso de encontrar alguna situación no prevista que pueda causar un potencial daño se rediseñará la solución, no aceptándose ningún tipo de errores por comisión.

La ley de defensa del consumidor establece que todo producto debe tener un mínimo de 3 meses de garantía. Sin embargo en principio se ofrece una garantía de 12 meses para demostrar la confianza que se tiene sobre el producto y de esta forma favorecer las ventas. Se verificará luego que este plazo sea compatible con el tiempo medio entre fallas (MTBF) calculado en la sección 9.1.2.6.

Se garantiza además que el producto cumple con la función para la cual ha sido diseñado bajo condiciones normales de operación y mantenimiento.

7. Ingeniería de Detalle

En esta sección se realiza el desarrollo de cada bloque que compone el sistema, definiendo a su vez sus bloques internos y las partes que componen a los mismos siguiendo los lineamientos definidos en la sección 6.1.

7.1 Diseño de Hardware

De acuerdo a lo definido en la sección 6.1.5, el sistema está basado en un microcontrolador con interfaz Wi-Fi para conectarse a la red de datos que a su vez se conecta por RS-232 con el módulo controlador X.10. El microcontrolador elegido es un Rabbit 4000 que viene en el embedded kit RCM4400W. A pesar de trabajar con un embedded kit que favorece un rápido desarrollo por la simplicidad del hardware, se debe adaptar las salidas que trabajan a 3,3v los niveles lógicos de los sensores, RS-232, relays etc y proveer además la alimentación correspondiente. Es por esta razón que se define una “Placa Base” que contiene todos los circuitos de alimentación, adaptación de niveles, como así también los periféricos que utiliza el equipo (relays, sensores, etc).

7.1.1 Diagrama de bloques

En la figura 7.1 se observa el diagrama de bloques correspondiente a la placa base. Como se puede observar, se indican las entradas y salidas de cada bloque para mayor claridad. Se pueden diferenciar los siguientes bloques:

- Fuente

- Conversor de Niveles RS-232 para interfaz X.10
- Sensor de Temperatura con conversor A/D
- Salidas contactadas a Relay con buffers de nivel
- Modulo RCM4400W

Para realizar el diseño de cada uno de los bloques que conforman la placa base se tuvieron en cuenta los siguientes aspectos:

- Recomendaciones realizadas por fabricantes
- Disminuir tiempos de desarrollo
- Asegurar cumplimiento de estándares
- Disminuir costos

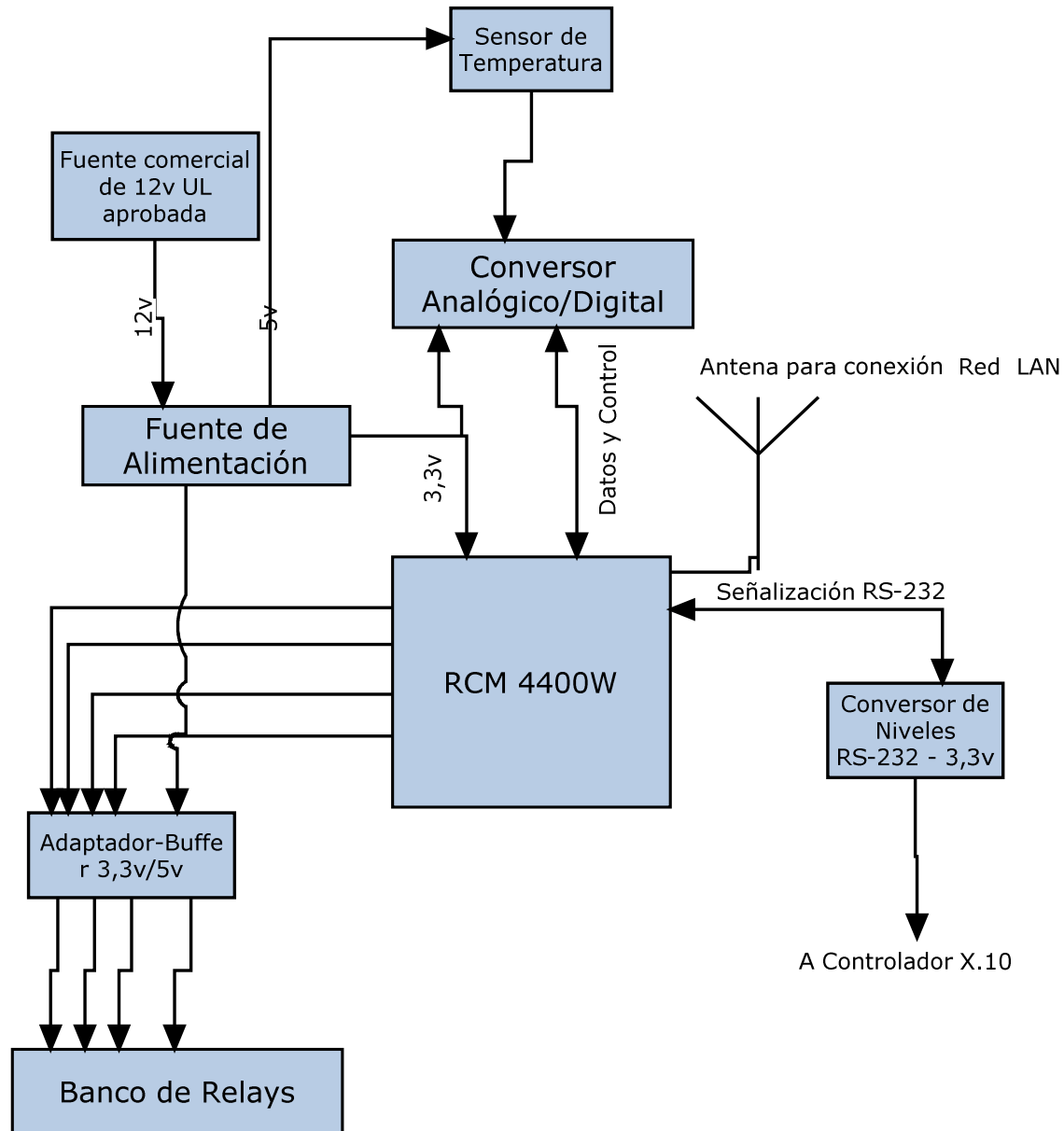


Figura 7.1: Diagrama de Bloques Placa Base

7.1.2 Fuente de Alimentación

Teniendo en cuenta en que el módulo RCM4400W es la pieza fundamental del equipo y a su vez la de mayor valor se decidió diseñar el resto de los componentes en función de la compatibilidad con el microcontrolador. En el caso de la fuente de alimentación, se siguió la recomendación que sigue el fabricante realizando la fuente de alimentación de la misma forma que se desarrolla en el kit de desarrollo del producto.

Bajo esta perspectiva la alimentación consiste en una fuente comercial de conversión de corriente alterna (110v/220v) a 12v y seguido a esto una fuente de conmutación de 5v y una fuente lineal de 3,3v para la alimentación del microcontrolador y conversor analógico digital. Se deberá asegurar que la conversión de corriente alterna a continua sea realizada por una fuente que cumpla con la normativa nacional así también como con las normal UL, siendo estas un estándar de electricidad.

El circuito esquemático correspondiente a la fuente lo expone el fabricante Rabbit en el documento “RCM4400W User Manual” en la página 90, el cual se reprodujo utilizando la herramienta informática OrCad 10.5 en la figura 7.2.

El fabricante utiliza una fuente de switching que se alimenta con los 12 volt de entrada del transformador externo. Se utiliza el diodo Shottky D1 para proteger la fuente y el resto del circuito. Se utiliza la fuente de switching de 5v dado que además del microcontrolador, el fabricante presupone la existencia de otros dispositivos con valores nominales de 5v de alimentación. El microcontrolador puede ser alimentado con una tensión entre 3,0v y 3,6v, por lo que se utiliza un regulador lineal con valor nominal de 3,3v.

La fuente de switching puede entregar un máximo de corriente de 1A. En modo de transmisión/recepción Wi-Fi el microcontrolador consume 450mA mientras que cuando no transmite consume 80mA, con lo que 1A resulta suficiente para alimentar el microcontrolador y los periféricos.

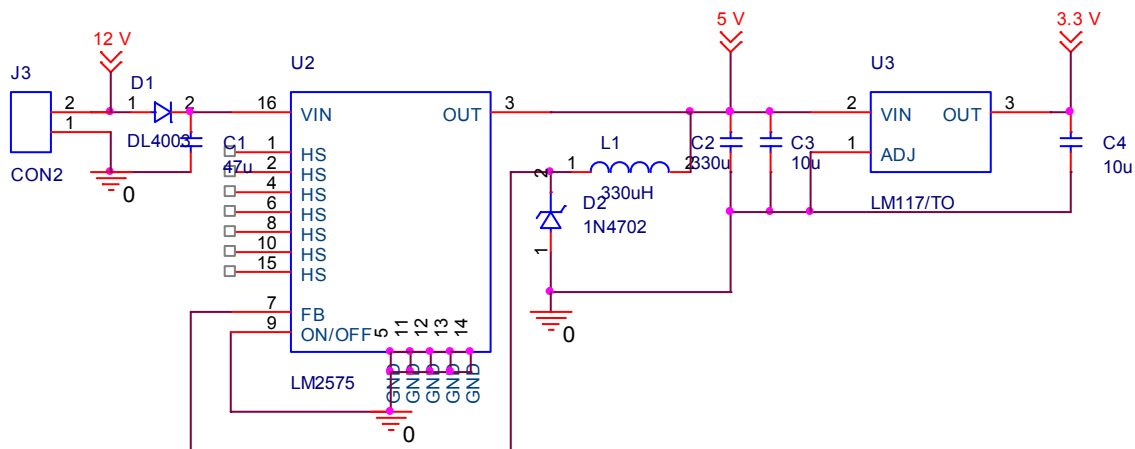


Figura 7.2: circuito esquemático de la fuente de alimentación

Los valores de los capacitores e inductores se tomaron exactamente igual a los sugeridos por el fabricante Rabbit en la placa de desarrollo. Se verificó a su vez que corresponden a los valores nominales utilizados por el fabricante de la fuente de switching. El fabricante sugiere colocar la fuente switching lo mas alejada posible del RCM4400W para minimizar el ruido de RF y utilizar componentes de bajo ruido.

7.1.3 Conversor de nivel RS-232

Debido a que la norma RS-232 utiliza valores de tensión superiores al máximo que puede trabajar el microcontrolador Rabbit4000, es que se debe realizar una

adaptación de niveles. Para llevar a cabo esta tarea se recurre a integrados Standard que realizan la función. Siguiendo la lógica de utilizar las recomendaciones del fabricante, en el kit de desarrollo utiliza el circuito integrado ICL3222 de Intersil. Sin embargo, este integrado no se encuentra disponible en el mercado local y su reemplazo es el MAX3222, cuyo proveedor es Maxim el cual se encuentra disponible. Ambos integrados son 100% compatibles en el pinout y consumen la misma potencia.

De las hojas de datos del proveedor, se extraen los valores de los capacitores, los cuales vienen preestablecidos para el correcto funcionamiento del equipo. En la figura 7.3 se puede observar el pinout con la conexión del integrado al microcontrolador por un lado, y al Terminal RJ-11 con el que se comunica al controlador X-10 (CM11) y a un Terminal DB-9 hembra.

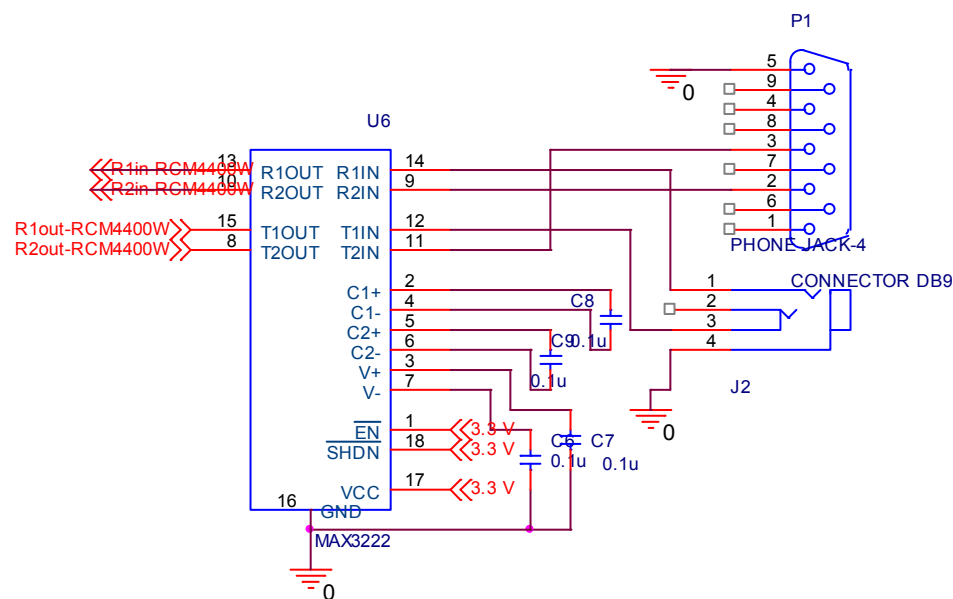


Figura 7.3: Circuito esquemático de conversión de niveles RS-232 a 3,3v

Como se indicó en la sección 6.1.2.1.1, el controlador X.10 modelo CM11 se comunica con el host, en este caso el microcontrolador, utilizando una interfaz RJ-11, utilizada habitualmente en los teléfonos.

Se utilizó el segundo puerto RS-232 para realizar una consola de administración. Esta consola permite realizar la primer configuración del equipo, es decir definir una dirección IP, máscara de red, puerta de enlace, etc. Esto se puede realizar aprovechando que el MAX3222 contiene 2 canales RS-232.

En la tabla 7.1 se observa el pinout del módulo controlador CM11, el cual fué utilizado para el diseño de la placa base:

Signal	RJ11 Connector
SIN	Pin 1
SOUT	Pin 3
GND	Pin 3
RI	Pin 2

Tabla 7.1: Pinout del módulo controlador CM11

En la tabla 7.2 se observa el pinout del protocolo RS-232 cuando se lo utiliza con interfaz DB-9 hembra.

Signal	DB9 Connector
SIN	Pin 2
SOUT	Pin 3
GND	Pin 5

Tabla 7.2: Pinout del conector DB9

En el caso del conector DB9 solo se utilizaron 3 señales, por lo que las otras señales no se conectaron debido a que no fueron utilizadas.

7.1.4: Sensor de temperatura con Conversor Analógico Digital

De acuerdo a lo concluido en la sección 6.1.2 y 6.1.3, se utiliza un sensor de temperatura LM35 y un conversor analógico digital (A/D) TLC548. La figura 7.4 muestra el circuito esquemático del Sensor y el A/D

El sensor de temperatura requiere alimentación de entre 4 y 30 volts y tiene una variación lineal de 10mV/°C a su salida. En este caso se lo alimenta con 5 volt. Por recomendación del fabricante del sensor (Nacional Semiconductor) en su hoja de datos, se agrega a la salida un amortiguador comprendido por un serie R-C con valores de 75Ω (representado por un circuito en paralelo de dos resistencias de 150 Ω) y 150 μF respectivamente. La recomendación es debida a que el fabricante indica que para un mejor desempeño al manejar cargas capacitivas superiores a 50pF se debe utilizar el circuito descrito anteriormente. En este caso, la carga del sensor es el A/D que en su hoja de datos especifica que la máxima capacidad de entrada es de 55pF y la típica es de 7pF. Si bien el valor máximo (peor caso) es ligeramente superior a la recomendación del fabricante del sensor, se comprobó que el desempeño del sistema es mejor si se utiliza la serie R-C a la salida. La mejora consiste en una mayor estabilidad del sensor, que en caso de no existir se manifiesta en variaciones constantes de $\pm 1^\circ \text{C}$. Dado que la aplicación para la cual está destinado el equipo, la variación de 1°C no es crítica, se prefiere la estabilidad que ofrece esta mejora haciendo que el cambio sea suave. Esto se puede apreciar en el siguiente ejemplo: Si se tiene un sensor inestable cerca de una temperatura programada, puede causar un número elevado de encendido/apagado del equipo que corrige la misma, sea el calefactor o el Aire Acondicionado.

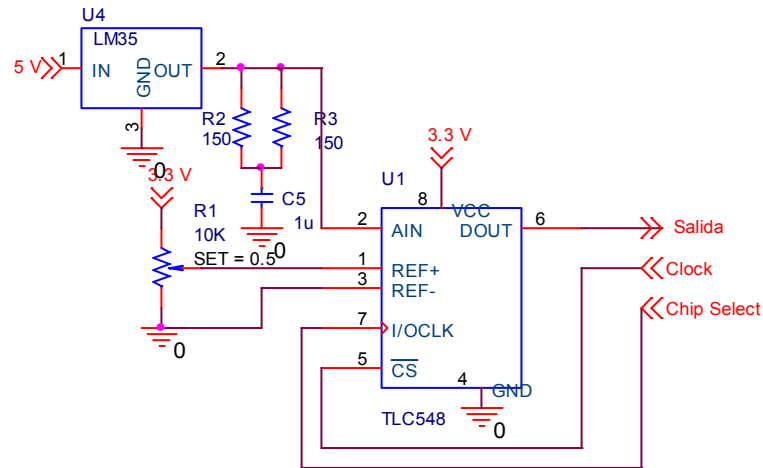


Figura 7.4: Diagrama esquemático del Sensor de temperatura y ADC

El A/D es un circuito con salidas TRI-State, es decir que para que el mismo entre en funcionamiento se debe activar (con un nivel bajo de tensión) la entrada chip select. Dado que es serie utilizando el protocolo SPI requiere un clock que sea sincrónico con el microcontrolador. Es por eso que en el programa que corre el microcontrolador una de sus funciones es generar el la señal de clock para el A/D. Es decir, que el micro le indica al A/D en que momento realizar la conversión y le provee la señalización necesaria para que el mismo le transfiera los datos. Para la parte analógica el A/D tiene dos referencias de tensión Ref+ y Ref-. Cuando el valor de tensión aplicado a la entrada (es decir, el valor resultante del sensor) es igual o superior a Ref+/2, la salida indica el valor 255, equivalente al fondo de escala de los 8 bits del conversor. Cuando el valor es igual o menor a Ref- , la salida es de 0. El valor Ref- se fija en 0 volt correspondiendo a 0°C. Para definir el valor de Ref+ se utilizó un trimmer de 10KΩ ajustándose el valor de salida 2,5 volt. De esta forma, el fondo de escala se obtendrá al 1,25 v, es decir:

$$\frac{Ref+}{2} = 10mv/^{\circ}C \cdot 125^{\circ} = 1,25v$$

Es decir que puede medir hasta una temperatura de 125° C con una resolución de 0,5°C, que resulta apropiado para la aplicación que se pretende. Luego en el programa del microcontrolador se realiza el ajuste de escala (además de valores binarios) para indicar la medición en °C.

7.1.5: Buffers de Salida con banco de Relays

Para la salida se eligieron relays de 5v de corriente continua. Se eligió en particular el modelo TRK2233F 5VCC las siguientes razones:

- Disponibilidad en el Mercado local

- Bajo costo,
- Bobina excitada a 5v, no necesitando fuente de otro valor.
- Pinout estandar
- Soportan corrientes de hasta 1,25A.

Dado que es un componente estandar puede ser reemplazado por un dispositivo similar de igual pinout en caso de ser necesario. Para mayor potencia se deberá colocar un contacto utilizando el Relay como excitador del contactor. Dado que esta tarea es personalizada quedará a cargo del cliente o podrá requerirlo como adicional.

Teniendo en cuenta que las salidas del microcontrolador tienen un bajo nivel de corriente y además son de 3,3v, para excitar los relay se debe poner una interfaz que eleve el nivel de tensión pueda proveer la corriente necesaria para excitar la bobina y originar el campo magnético correspondiente. Para tal motivo se utiliza el circuito integrado 74LS244. Se eligió por las siguientes razones:

- Bajo costo
- Alta disponibilidad en el mercado
- Mas de 10 años en el mercado
- El mínimo valor de entrada que toma como alto es 2v
- Corriente de salida de hasta 15mA

El circuito contiene 8 entradas/salidas de las cuales se utilizaron solo 4. Se debe colocar además un diodo polarizado en inversa. La función del diodo es proteger el circuito de salida del 74LS244 al producirse el corte en la bobina del relay. En la figura 7.5 se puede ver el circuito esquemático de las salidas conectadas a relay. Y1 representa la conexión proveniente del microcontrolador correspondiente a la salida 1. De igual forma ocurre con la salida 2, 3 etc. Además, se puede observar que se incluyeron las borneras correspondientes para las salidas. Las mismas deben ser capaces de manejar corrientes de 1,25A con tensiones de 220v.

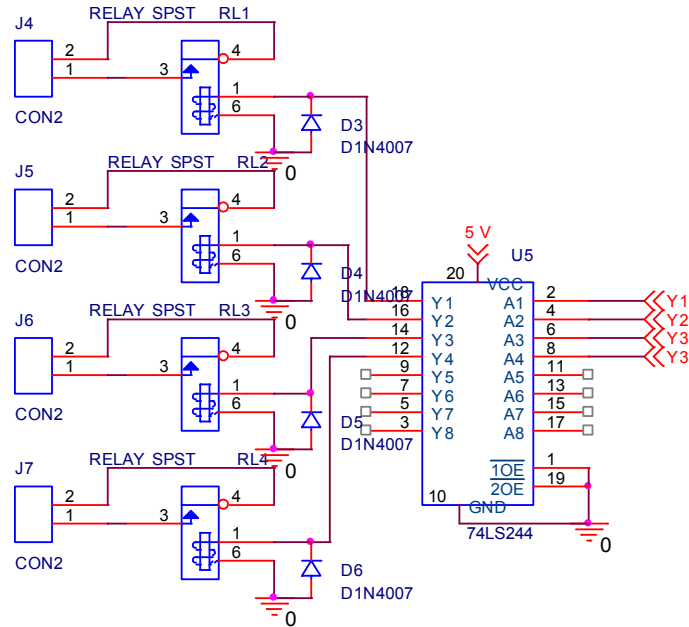


Figura 7.5: Diagrama esquemático de las salidas conectadas a relay

7.1.6: Módulo RCM4400W

De acuerdo a lo establecido en el punto 6.1.4 el microcontrolador a utilizar es el Rabbit4000 contenido el kit RCM4400W. El kit se conecta con la placa base utilizando un conector de 2 filas x 25 pins cada una y una separación entre pines de 1,27mm. El RMC4400W tiene un conector macho por lo que la placa base ser hembra. Como se mencionó en el ítem 7.1.2, se diseñó la fuente en función de los requerimientos del microcontrolador, así como en el punto 7.1.4 se eligió el conversor A/D de forma tal de ser compatible con los niveles lógicos del Rabbit.

En la figura 7.6 se puede observar el diagrama esquemático completo de la placa base.

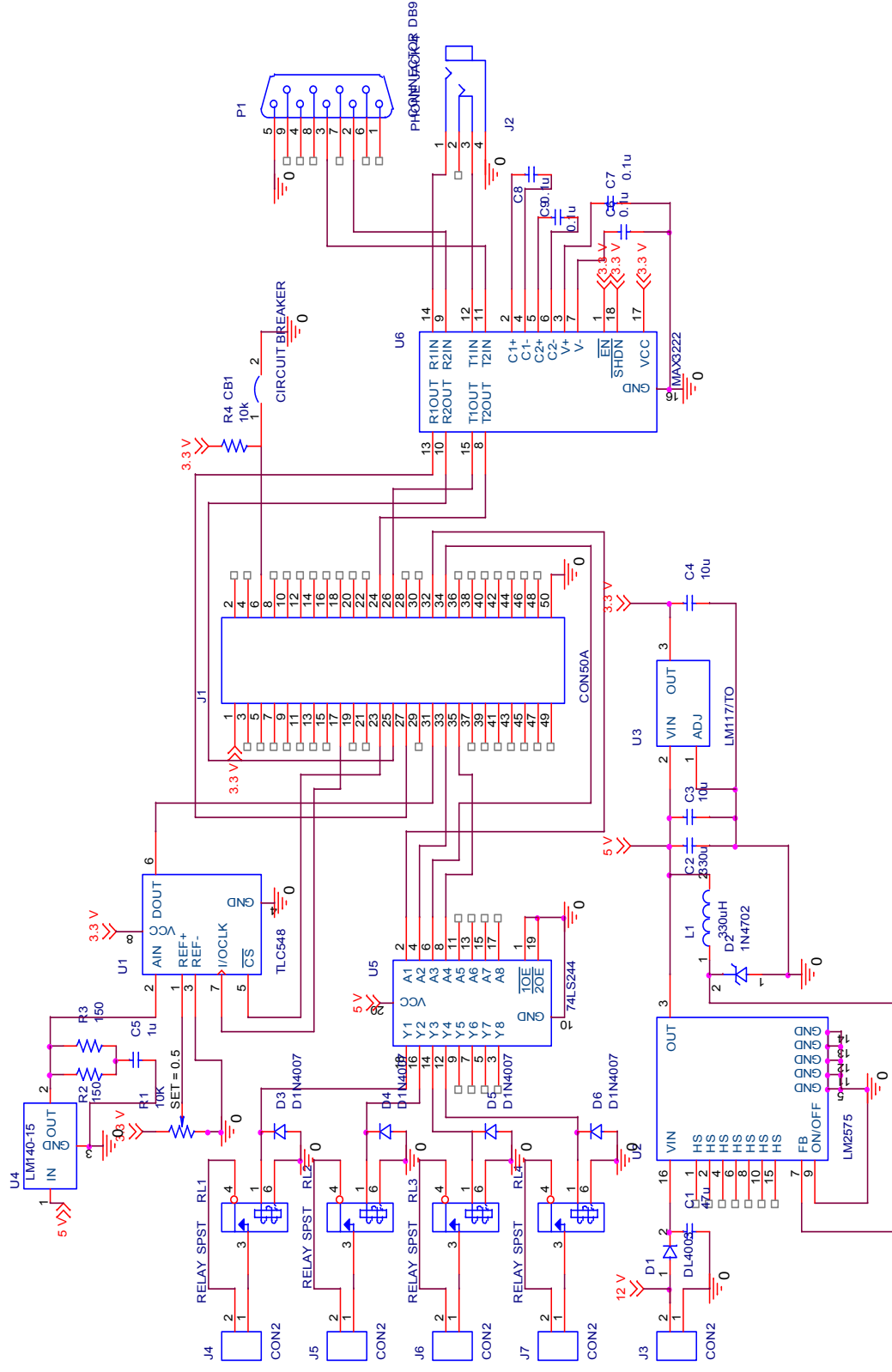


Figura 7.6: Diagrama esquemático completo

7.2 Diseño de Firmware

7.2.1 Flujo de Comunicaciones

Para desarrollar el programa que será compilado en el microcontrolador, antes se deberá definir el esquema de comunicaciones entre los distintos componentes del sistema. Para ello deberá definir el flujo de comunicaciones y la forma en que dialogarán los siguientes componentes:

- Usuarios con el servidor
- Servidor con módulos controladores
- Módulo controlador con dispositivos X.10

7.2.1.1 Comunicación Usuario-Servidor

Definición del modelo cliente servidor

Este modelo se basa en el procesamiento cooperativo de la información, en el cual múltiples clientes, distribuidos en la red, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, se puede definir este modelo como una arquitectura distribuida que permite a los usuarios finales obtener acceso a información o a ejecución de procedimientos remotos en forma transparente. El cliente envía un mensaje solicitando un determinado servicio (request) a un servidor, y este envía uno o varios mensajes con la respuesta (response). En un sistema distribuido cada host puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

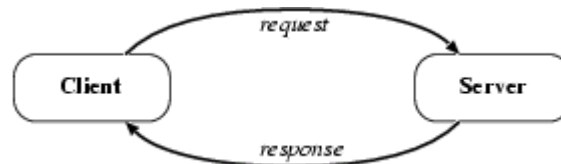


Figura 7.7: Estructura Cliente-Servidor

Tanto los clientes como servidores son entidades independientes que operan conjuntamente a través de una red para realizar una tarea. Pero para hacer la distinción respecto de otras formas de arquitecturas o software distribuidos, se presenta una lista de características que debieran cumplir una arquitectura cliente-servidor:

Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en diferentes máquinas distribuidas a lo largo de la red.

Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.

La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.

Los clientes corresponden a procesos activos en cuanto a que son éstos lo que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.

No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.

Las plataformas de software y hardware entre clientes y servidores son independientes. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente/servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

7.2.1.2 Comunicación Servidor-Módulos Controladores

En este caso se podía haber elegido dos esquemas:

1. El server actuando como cliente de cada RCM4400W, iniciando una comunicación cada vez que se debe ejecutar una acción u obtener un dato del módulo controlador
2. Cada RCM4400W actuando como cliente del Server, iniciando una sesión durante todo el ciclo de encendido y una vez establecida la comunicación el Server indica los comandos a ejecutar

Además de definir los roles de cada equipo, se debe definir en este punto el nivel de transporte a utilizar sobre la capa de red IP. Dado la generalidad y el soporte nativo tanto del sistema operativo como principalmente del RCM4400W se debe elegir entre TCP y UDP. Las características de cada uno de ellos son:

- TCP: Esquema de red con conexión, es decir que se establece un intercambio de mensajes para iniciar una sesión. Este intercambio es denominado “Three way handshaking” dado que se necesitan tres paquetes para establecer una conexión. Esto hace que el establecimiento de la conexión tenga un delay por la mensajería asociada. A su vez, en caso de error requiere retransmisión independientemente de la aplicación, lo que lo hace mas seguro
- UDP: Esquema de red sin conexión, es decir envía el paquete directo a destino sin establecer la conexión. En caso de error de transmisión descarta el paquete. Por

estas razones requiere menos tiempo de establecimiento de la conexión y requiere verificación de errores en la aplicación.

A pesar que el tráfico adicional utilizado para el establecimiento de las sesiones TCP, debido a que la información a transmitir es baja en relación a la información necesaria para realizar el establecimiento de la conexión, se decide utilizar TCP dado que el intercambio de mensajes se realiza en un red local inalámbrica de alta velocidad, por lo que tráfico extra requerido por una sesión TCP resulta muy bajo en relación a la capacidad de transmisión. Esta decisión se toma basándose en que es más sencillo que el control de errores lo realice el protocolo antes que pasarlo a la aplicación, acelerando de esta forma el desarrollo.

Definido TCP como protocolo de transporte sobre IP, resta definir si la sesión la establecerá el RCM4400W hacia el Server o viceversa, definiendo quien actuará de Server y quien de cliente en el modelo cliente-servidor.

Se decidió utilizar al Server como cliente, dado que es más fácil establecer una sesión cada vez que se requiera ejecutar un comando que establecer una sesión durante todo el transcurso de encendido del equipo. En este último caso se debería estar verificando durante todo momento el estado del socket con la consecuente carga del procesador (de ambos equipos) y la dificultad de programación. Entonces, al establecer un socket y cerrarlo por cada solicitud se simplifica el problema a expensas enviar mayor información de la red, algo que ya se mencionó es absolutamente aceptable.

Nota: La nomenclatura de Server se hace a los efectos que hay un software instalado en una computadora que comúnmente se la denomina servidor. Sin embargo, el Server puede ser cliente de otro servidor.

7.2.1.3 Comunicación Módulo Controlador-Controlador X.10

En este caso el DTE (RCM4400W) actúa de cliente y el X10 CM11 (DCE) de servidor, dado que es el Microcontrolador quien inicia la comunicación. Para que el CM11 envíe una señal hacia los dispositivos X.10 se debe establecer un diálogo que consiste en los siguientes pasos

1. El DTE inicia la transmisión indicando que le va a enviar una dirección de dispositivo a controlar. Para ello envía dos bytes. El

primero contiene un encabezado (header) que indica que se envía un dirección. El segundo contiene la dirección para ejecutar el comando

2. El DCE recibe la información y calcula un checksum, que lo envía nuevamente hacia el DTE
3. El DTE verifica el checksum con respecto a los datos enviados. En caso de ser incorrecto transmite nuevamente el header de dirección seguido de la dirección (1). En caso de ser correcto transmite un 0x00 indicando que la transmisión fue correcta.
4. Al recibir el 0x00 el DCE envía un 0x55 indicando que el CM11 se encuentra listo y que recibió bien la dirección
5. Luego inicia el envío de la función. El DTE envía nuevamente 2 bytes. El primero con un header correspondiente a función y el segundo con la housecode seguido de la función (on, off, dimm, etc).
6. El DCE calcula el checksum idem (2)
7. Idem 3
8. Idem 4. Una vez terminada la comunicación, el CM11 envía por la red eléctrica el comando a los dispositivos controlados.

En las figuras 7.10 y 7.11 se pueden observar la secuencia de intercambio de los datos en el tiempo de una forma gráfica. La primera es una transmisión exitosa, mientras que la segunda se detecta un error y solicita retransmisión.

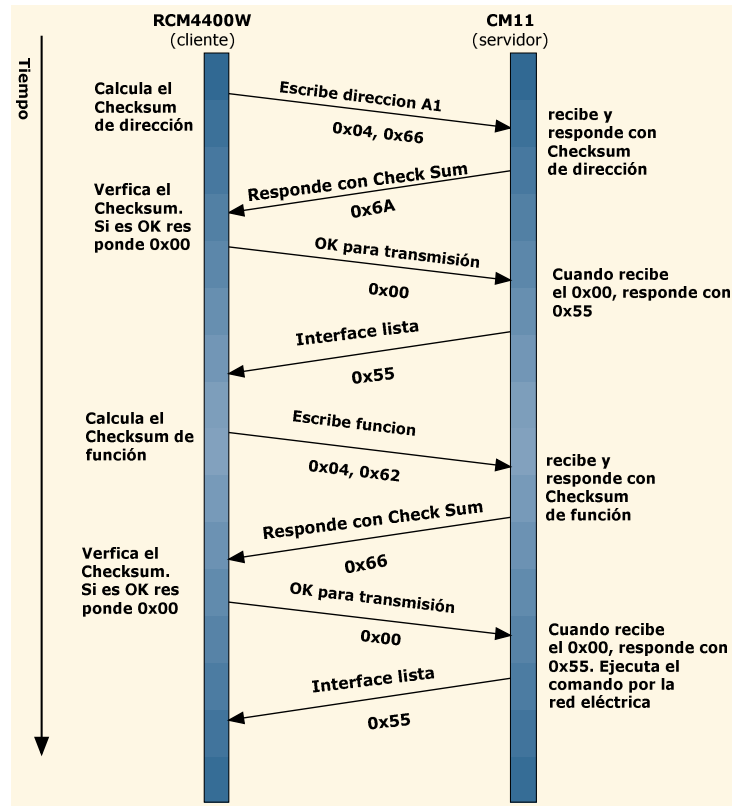


Figura 7.8: Transmisión con entre RCM4400W y X.10 CM11

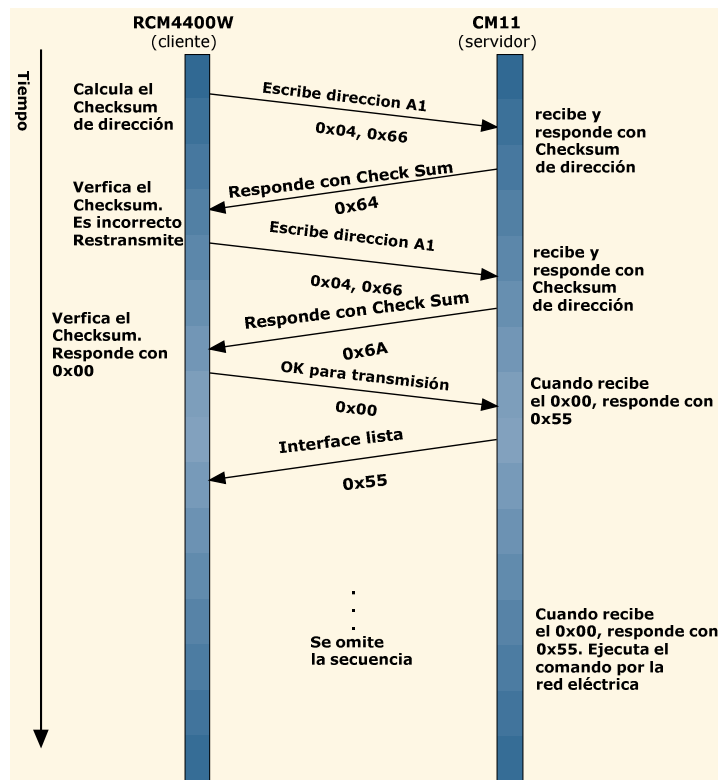


Figura 7.9: Transmisión con error de comunicación

7.2.2 Diagramas de Flujo de Firmware

El programa consta principalmente de 3 módulos:

1. Funcion main: Inicializa las variables, puertos, red. Además se corre el loop principal, que verifica si existe algún socket abierto con datos y verifica si tiene datos en el snack TCP/IP (además de los del socket) para responder ping, arp, etc.
2. Socket.h: Es la librería en donde se define el trabajo del socket y las funciones principales. Invoca una de sus funciones desde el main para verificar el estado del mismo. Lee el contenido del socket y de acuerdo al mismo ejecuta la función correspondiente. Están definidas las funciones para leer temperatura, iniciar acciones con dispositivos directamente conectados, etc. Responde al socket con mensajes de ok, error, not found y time out en un formato similar al utilizado http.
3. X10-CM11.h: Es la librería de funciones X.10. Se invoca desde una de las funciones de Socket.h. La función principal recibe el house code, el device code y la acción. Acomoda la información y realiza el intercambio entre el RCM4400W y el CM11. Verifica el checksum y solicita retransmisión en caso de ser necesario.

Para un mejor entendimiento de los diagramas de flujo, se muestra a continuación la notación empleada:

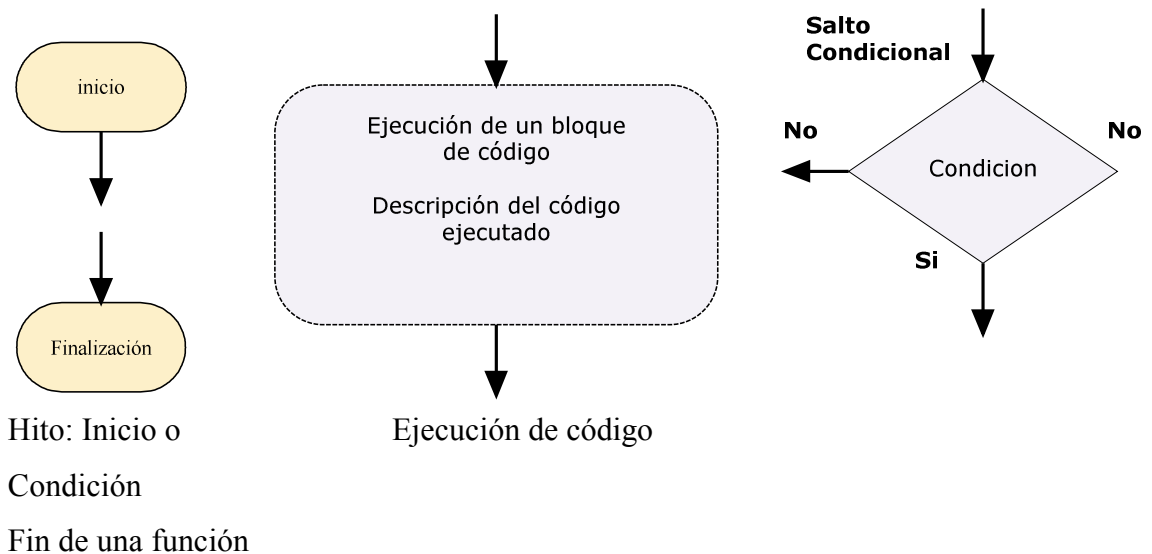


Figura 7.10: Referencias utilizadas en los diagramas de bloques

7.2.2.1 Main.h

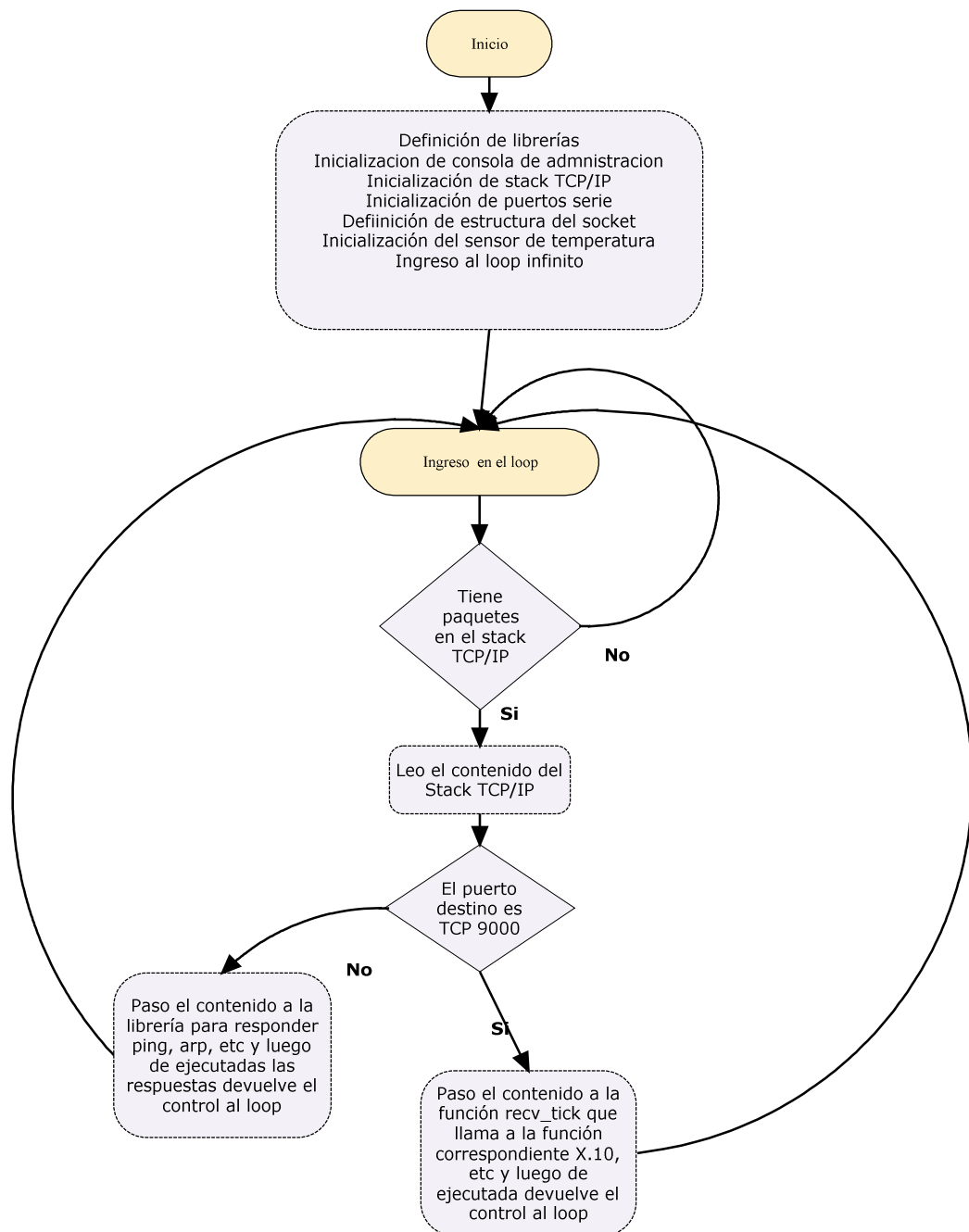


Figura 7.11: Diagrama de flujo de la función main

7.2.2.2 Funciones de Socket.h

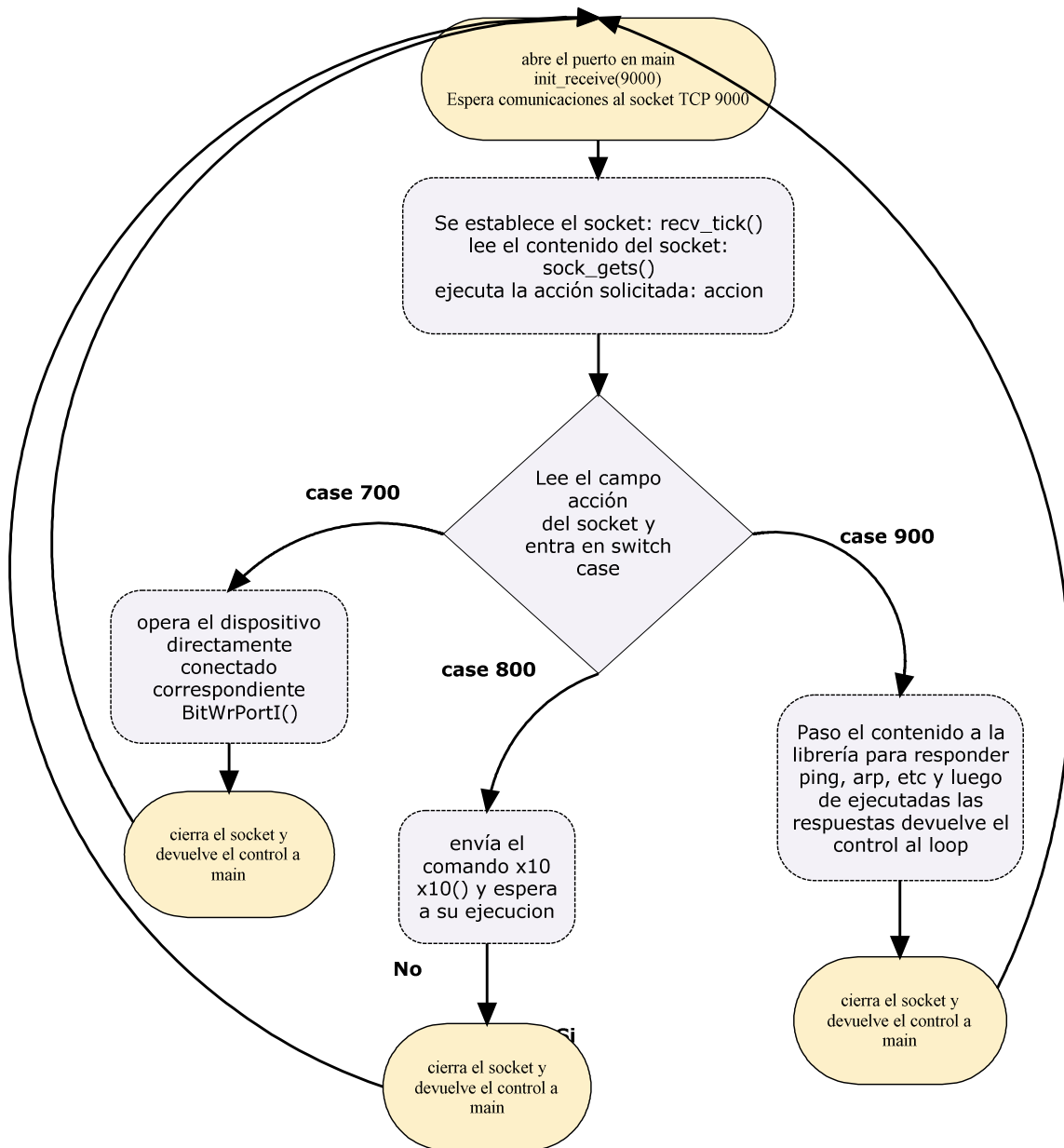


Figura 7.12: Diagrama de flujo de las funciones contenidas en Socket.h

7.2.2.3 Funciones de X10-CM11.h

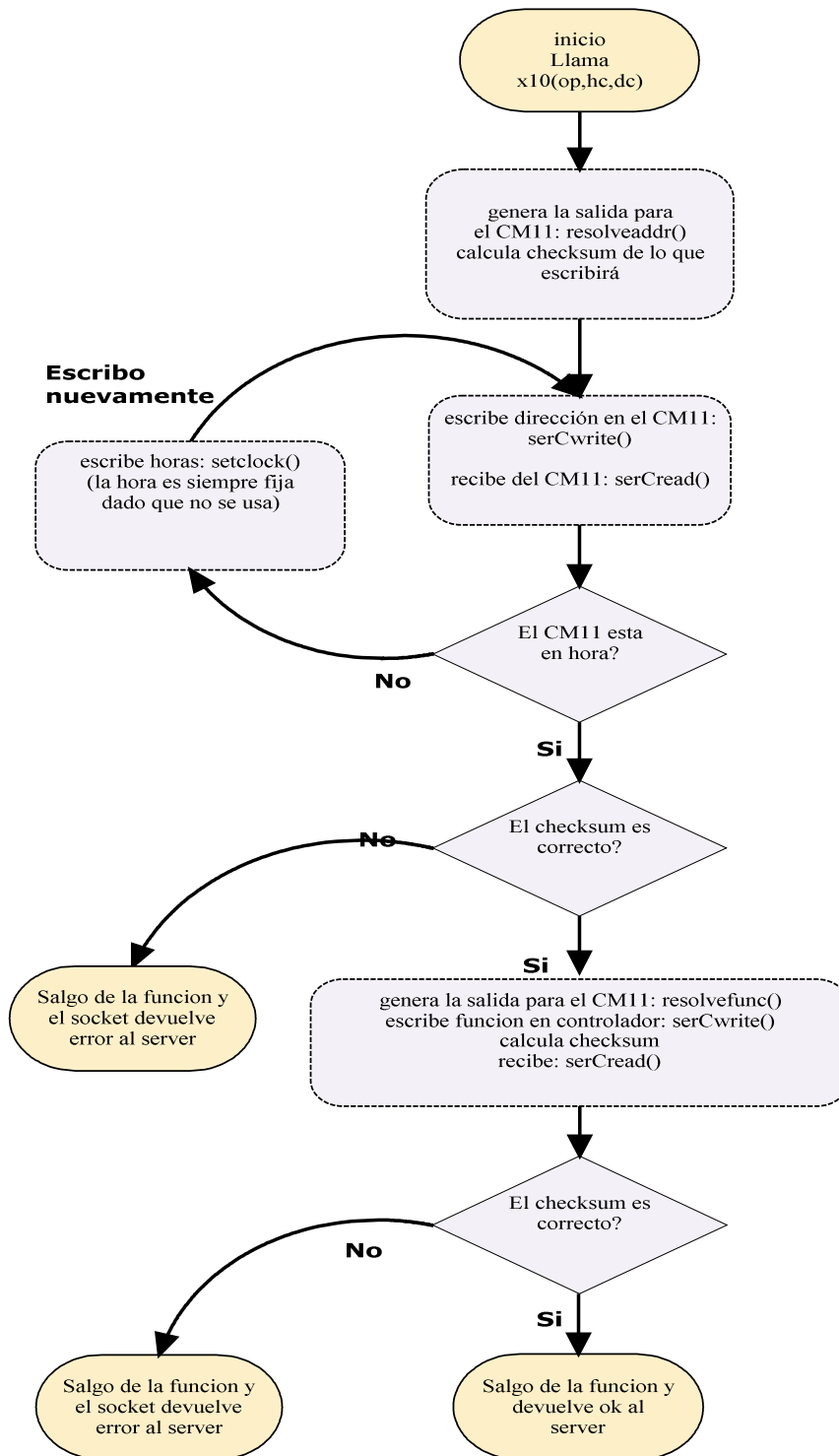


Figura 7.13: Diagrama de flujo de la s funciones contenidas en X10-CM11.h

7.3 Diseño de Software: Interfaz Web

La interfaz web de gestión se diseño con un sistema de validación y autenticación de usuarios para mantener de manera segura la gestión del sistema, su lógica de funcionamiento se muestra en el siguiente esquema:

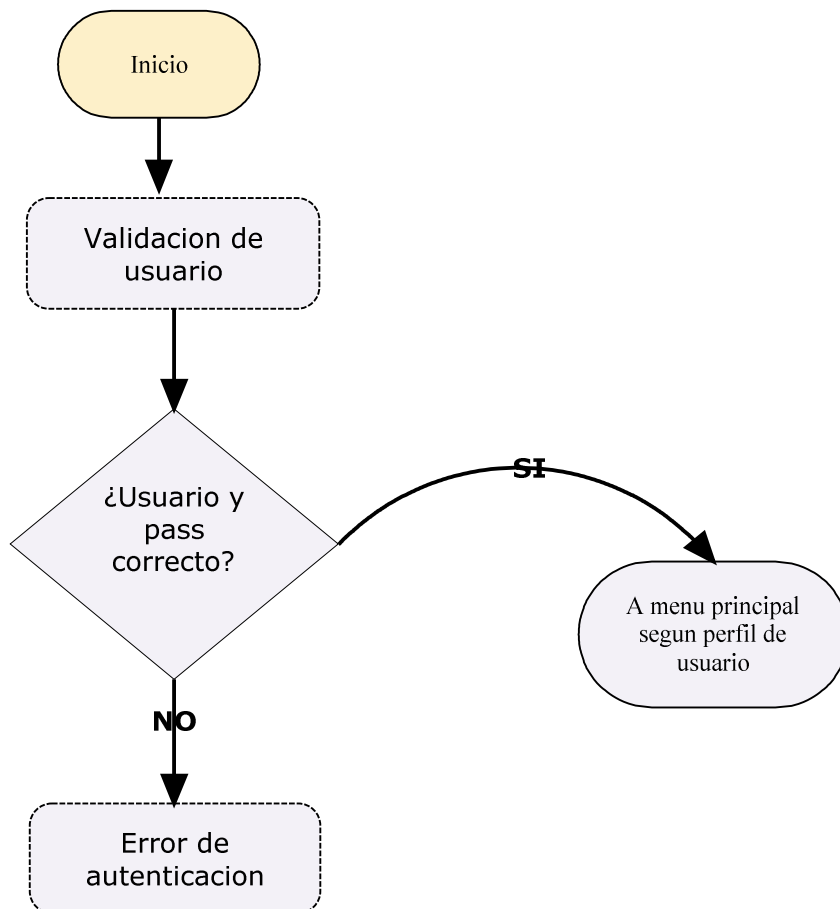


Figura 7.14: Diagrama de flujo autenticación interfaz Web

El modo por el cual se pide al usuario que ingrese sus datos es el estándar de validación de HTTP indicando que se requiere validación para “domotic”:

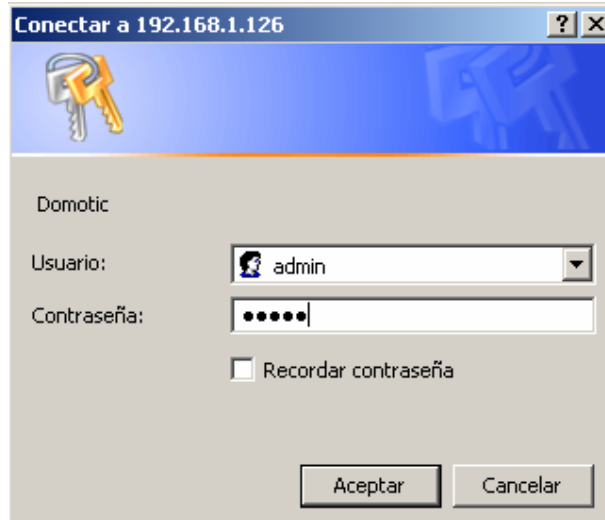


Figura 7.15: Pantalla de validación en página Web

Luego si la validación de usuario se completa de manera exitosa se accede al menú principal el cual cuenta con tres partes dentro de las cuales se encuentra cada una de las aplicaciones según el siguiente esquema:

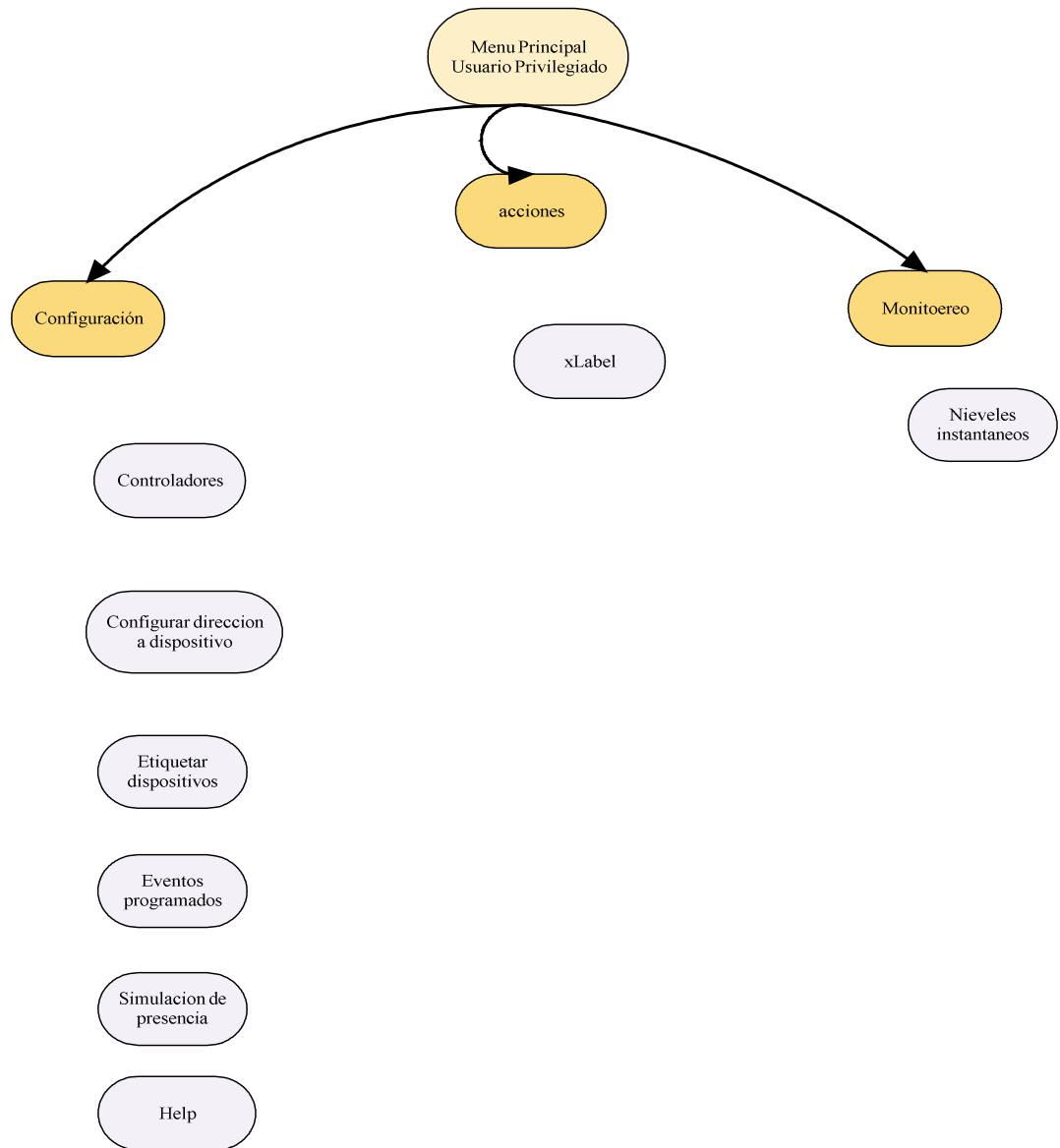


Figura 7.16: Digrama de flujos con las opciones del menú

Accediendo a la interfaz con un explorador convencional de hipertexto el usuario ve lo siguiente:

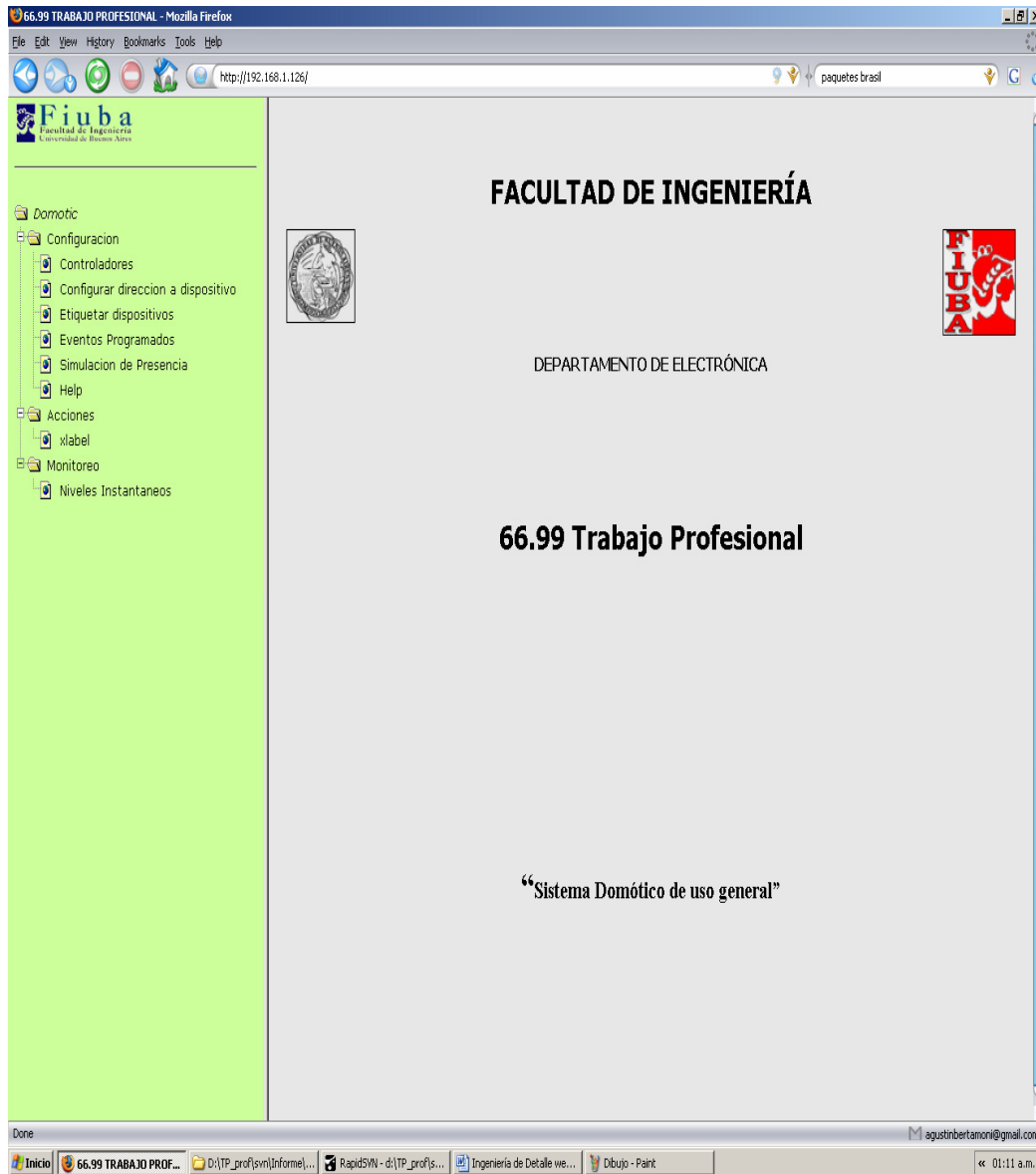


Figura 7.17: Pantalla de Inicio Interfaz Web

Cada uno de las entradas en el índice que se muestra del lado derecho de la pantalla es un link a un programa ejecutado por el web server mostrando en la interfaz el resultado arrojado según las variables con las que se lo alimente al momento de su ejecución.

A continuación se hace una breve descripción de los ítems del menú indicando que programa se ejecuta en cada caso y mostrando la lógica del mismo así como también la pantalla que se le presenta al usuario:

1) Controladores → setcontrollers.pl

Este programa muestra los controladores que se encuentran configurados en caso de que existan, posteriormente se le presenta al usuario un formulario donde debe ingresar la dirección IP del controlador a configurar junto con el nombre que desea asignarle al mismo. Si los datos tienen un formato válido estos son cargados a una base de datos donde se mantiene la información de cada uno de los controladores configurados. La lógica de funcionamiento y lo que el usuario ve en pantalla de este programa se muestran a continuación:

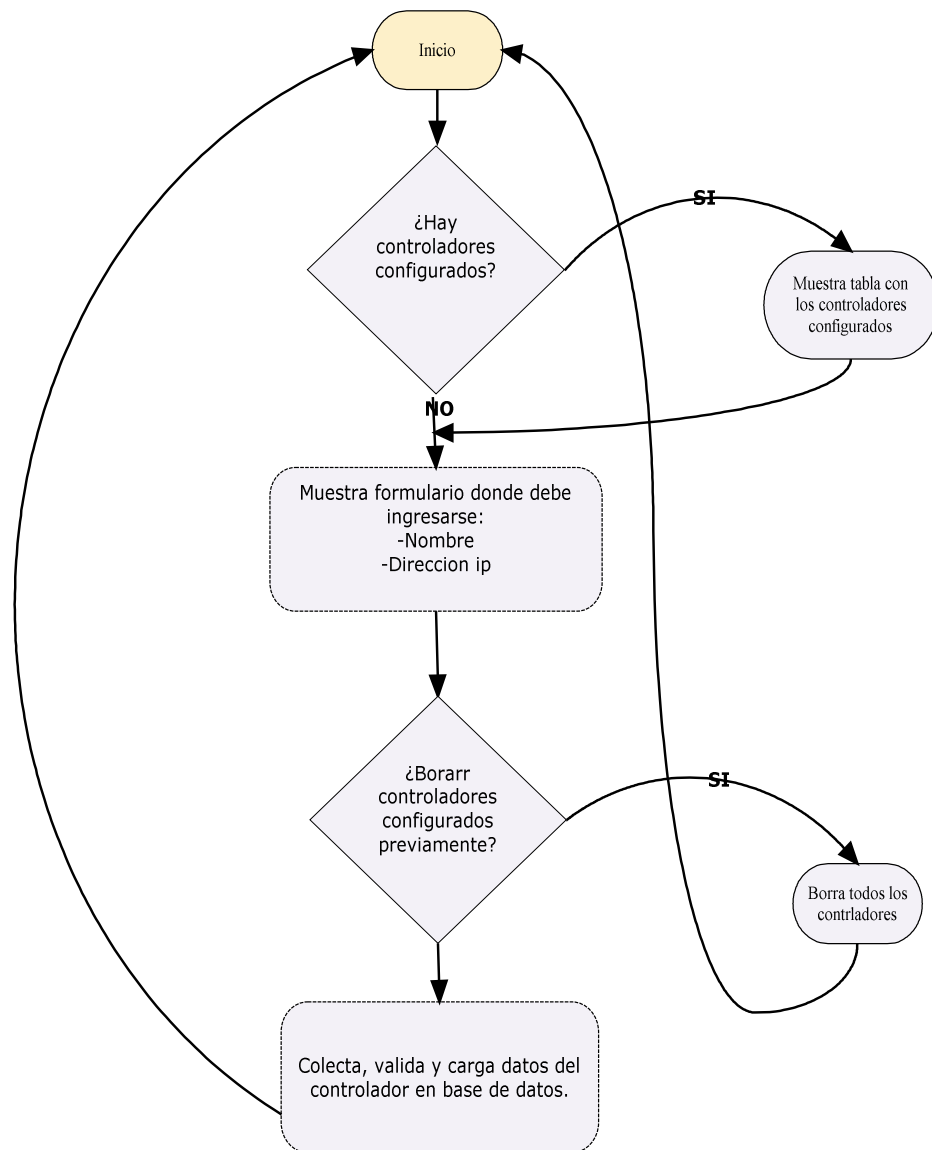


Figura 7.18: Diagrama de flujo programa setcontrollers.pl

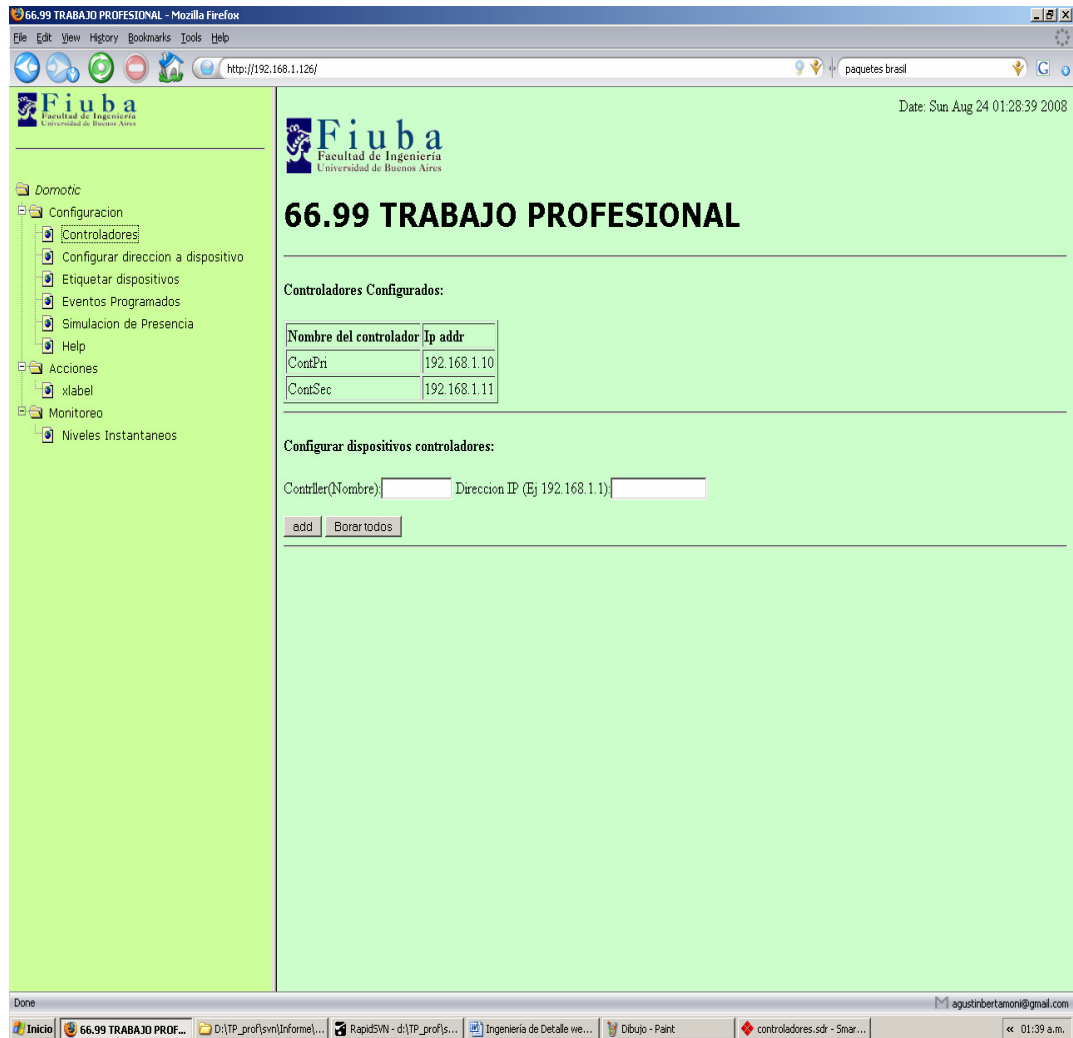


Figura 7.19: Pantalla de configuración de los controladores

2) Configurar direccion a dispositivo → setaddr.pl

Este programa le presenta al usuario un formulario donde debe ingresar el House Code y Device Code que desee asignarle al dispositivo a configurar. Los datos son procesados y enviados al controlador quien posteriormente envía el comando al

dispositivo X10, la lógica de funcionamiento y lo que el usuario ve en pantalla de este programa se muestran a continuación:

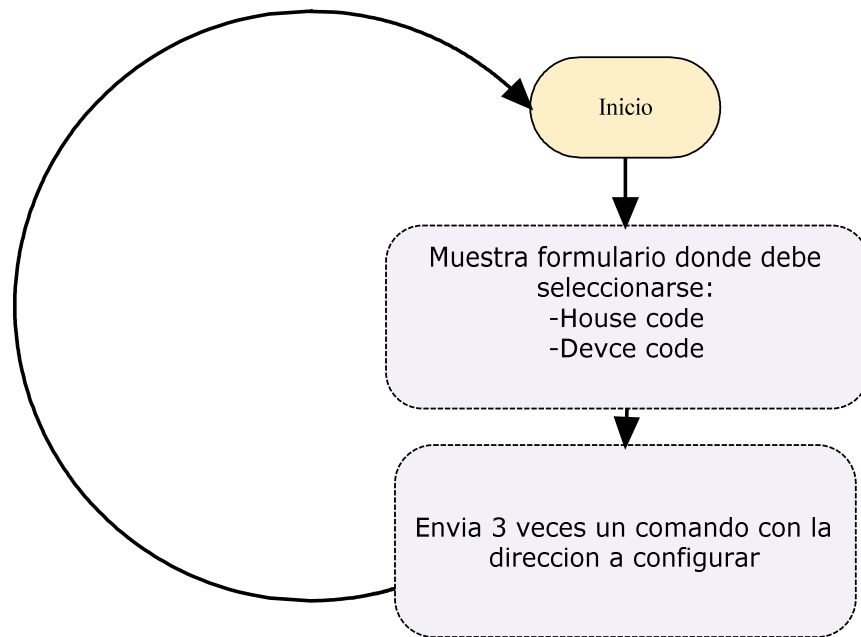


Figura 7.20: Diagrama de flujo programa `setaddress.pl`

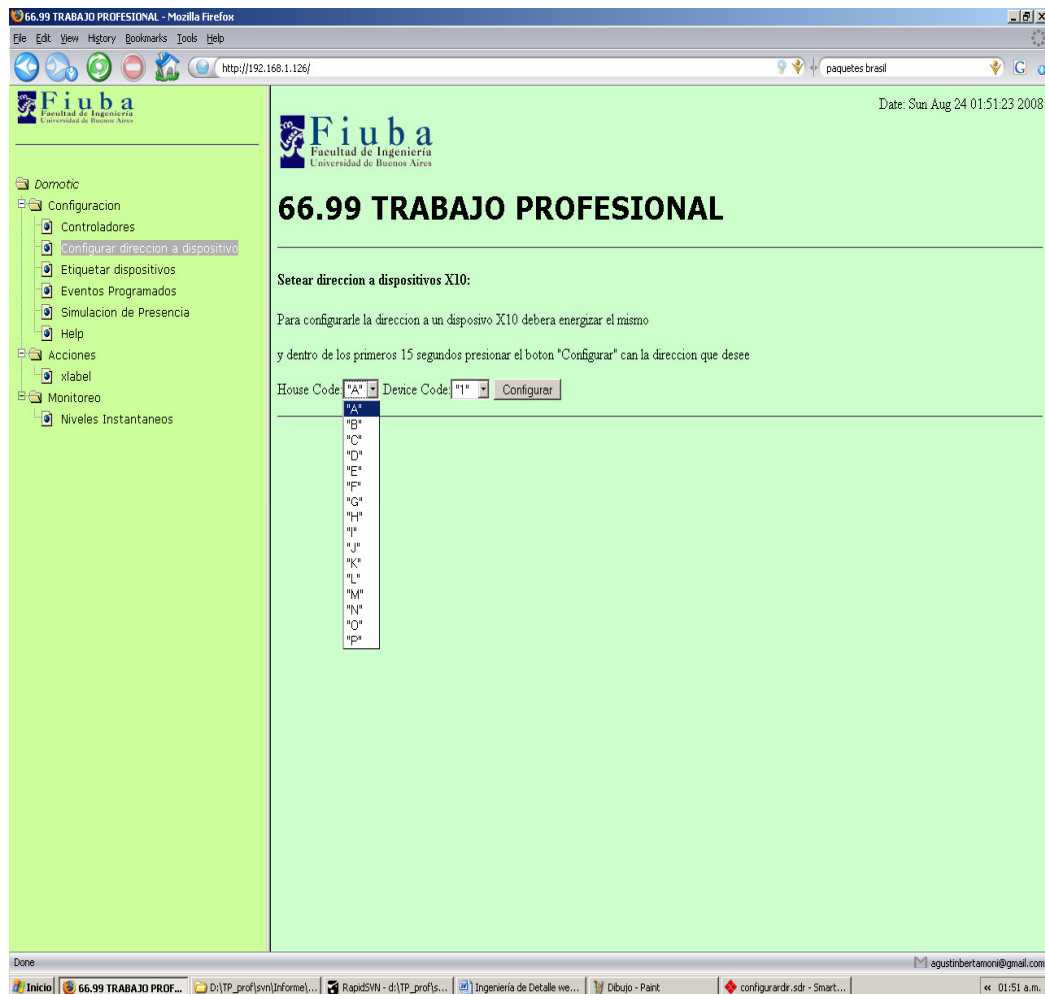


Figura 7.21: Pantalla de configuración de las direcciones X.10

3) Etiquetar a dispositivos → deviceconf.pl

Este programa muestra las etiquetas que se encuentran aplicadas sobre los dispositivos indicando los datos del dispositivo así como también el nombre del controlador por el cual es comandado, posteriormente se le presenta al usuario un formulario donde debe ingresar los datos del dispositivo a configurar junto con el nombre del controlador que al que se desea asignar el mismo. Si los datos tienen un formato válido estos son cargados en una base de datos donde se mantiene la información de cada una de las etiquetas configuradas para luego ser consultados desde los programas que ejecutan tareas sobre estas etiquetas. La lógica de funcionamiento y lo que el usuario ve en pantalla de este programa se muestran a continuación:

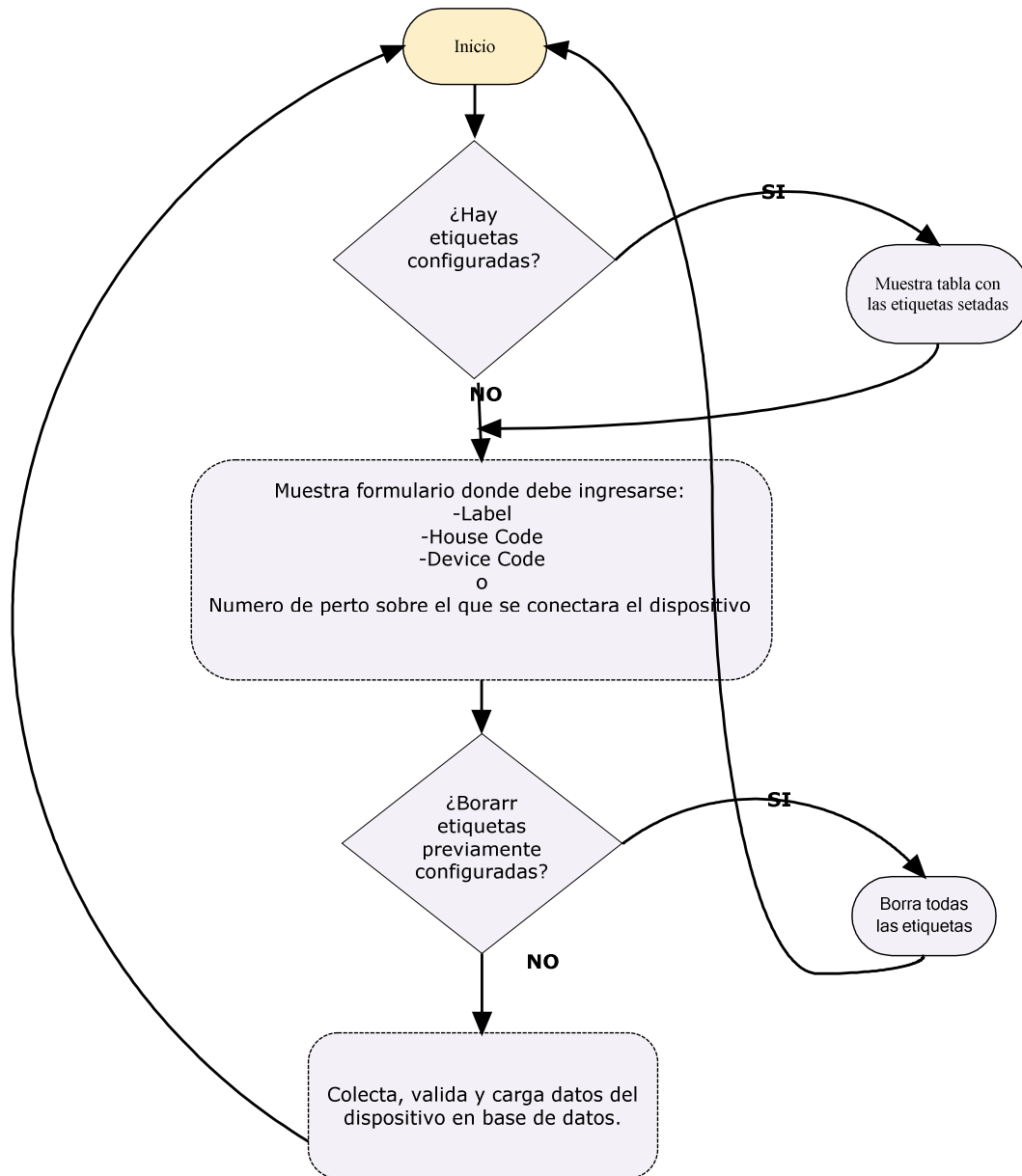


Figura 7.22: Diagrama de flujo programa `deviceconfig.pl`

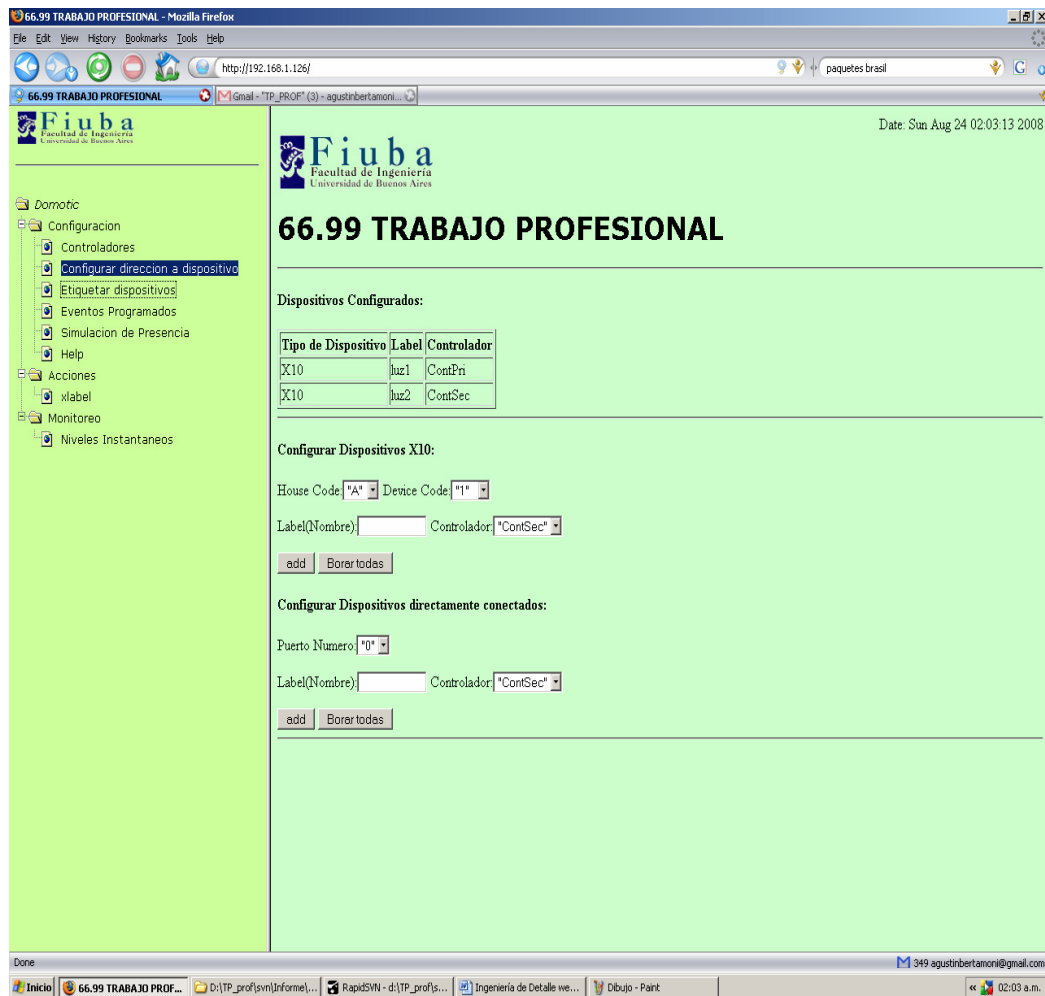


Figura 7.23: Pantalla de configuración de las etiquetas a los dispositivos

4) Eventos programados → progevent.pl

Este programa muestra todos los eventos programados previamente mostrando para cada uno el dispositivo sobre el que se programa el evento así como también la fecha y hora a la que se ejecuta el mismo, posteriormente se le presenta al usuario un formulario donde debe ingresar los datos mencionados en caso de querer agregar un

nuevo evento programado. Si los datos tienen un formato válido estos son cargados en una base de datos donde se mantiene la información de cada uno de los eventos configurados. La lógica de funcionamiento y lo que el usuario ve en pantalla de este programa se muestran a continuación:

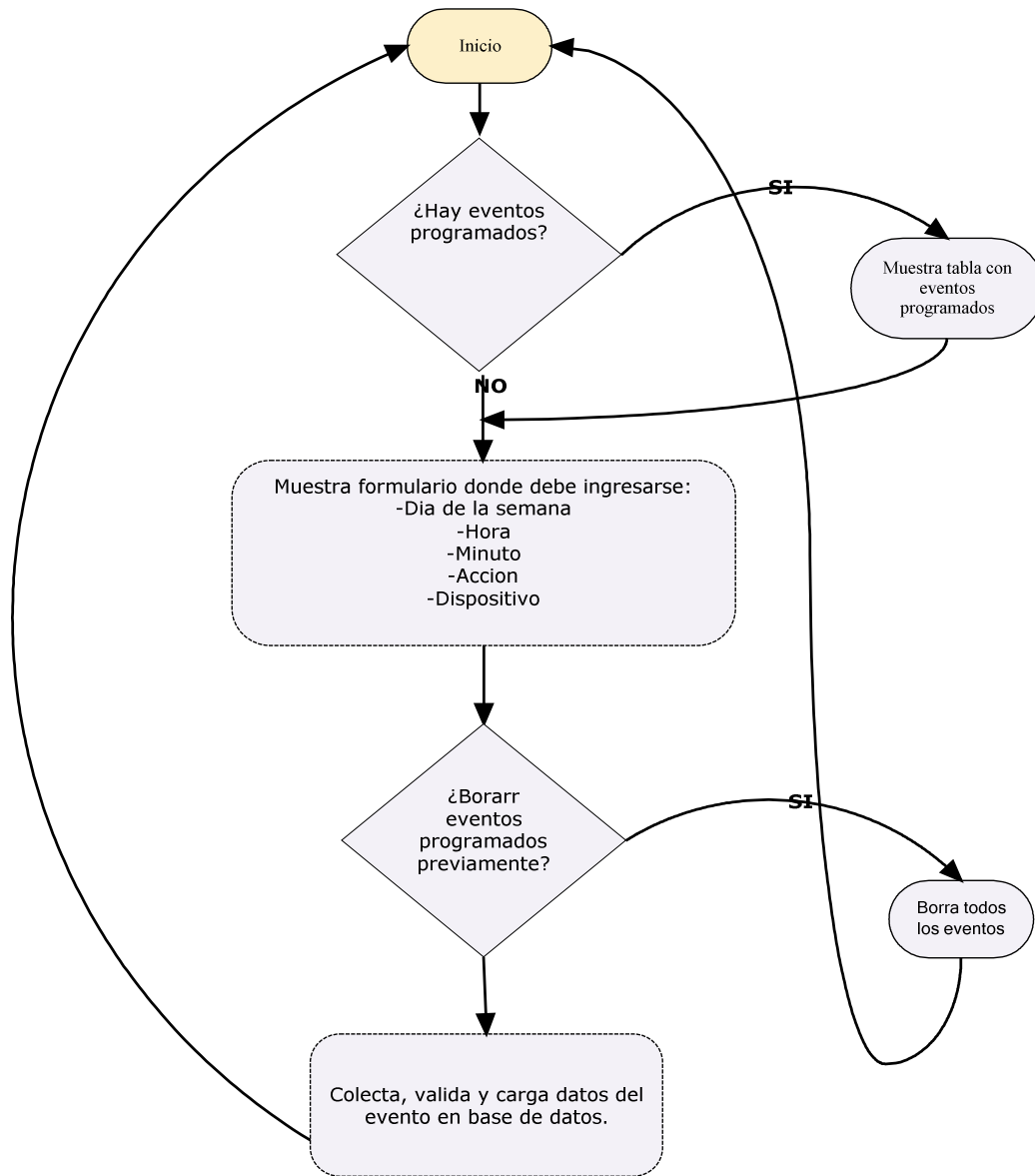


Figura 7.24: Diagrama de flujo programa *progevent.pl*

66.99 TRABAJO PROFESIONAL - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://192.168.1.126/

66.99 TRABAJO PROFESIONAL

Date: Sun Aug 24 14:19:27 2008

Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

66.99 TRABAJO PROFESIONAL

Eventos Programados:

Label	Action	Hora	Minuto	Dia de la semana(0-6)
luz2	On	8	0	5

Configurar tareas programadas:

Dia de la semana: "Domingo" Hora: "0" Minuto: "0"

Acción: "encender" Dispositivo: "luz1"

send Borrar todas

Inicio 66.99 TRABAJO PROF... D:\TP_prof\svn\Informel... D:\TP_prof\svn\web\cgi... Ingeniería de Detale we... EventosProg.sdr - Smart... 02:42 p.m.

Figura 7.25: Pantalla de configuración de las tareas programadas

5) Simulación de presencia → simulpres.pl

Este programa genera eventos programados similares a los anteriormente descritos con la diferencia que estos se ejecutan de manera periódica dentro del intervalo configurado por el usuario. La lógica de funcionamiento y lo que el usuario ve en pantalla de este programa se muestran a continuación:

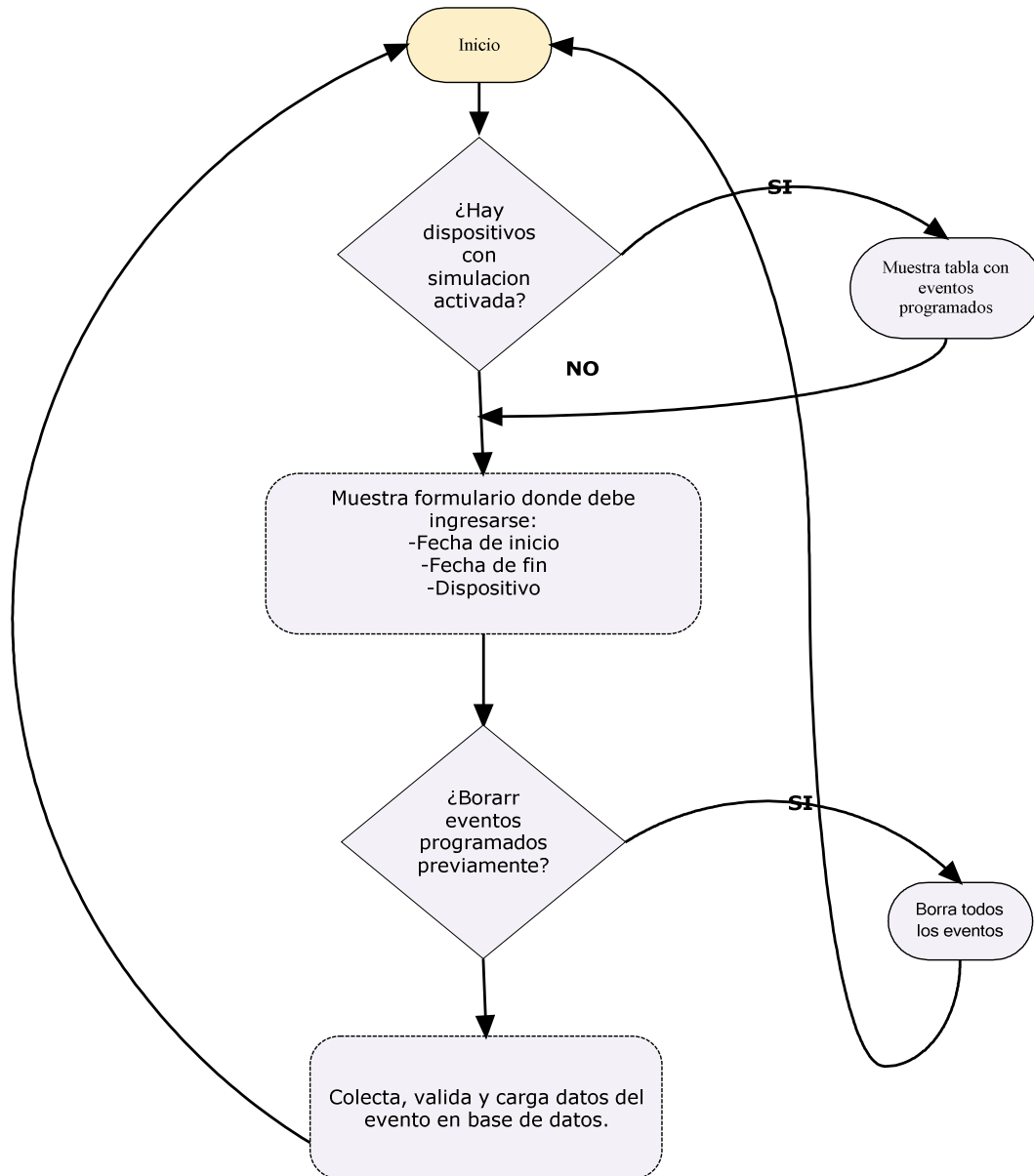


Figura 7.26: Diagrama de flujo programa simulares.pl

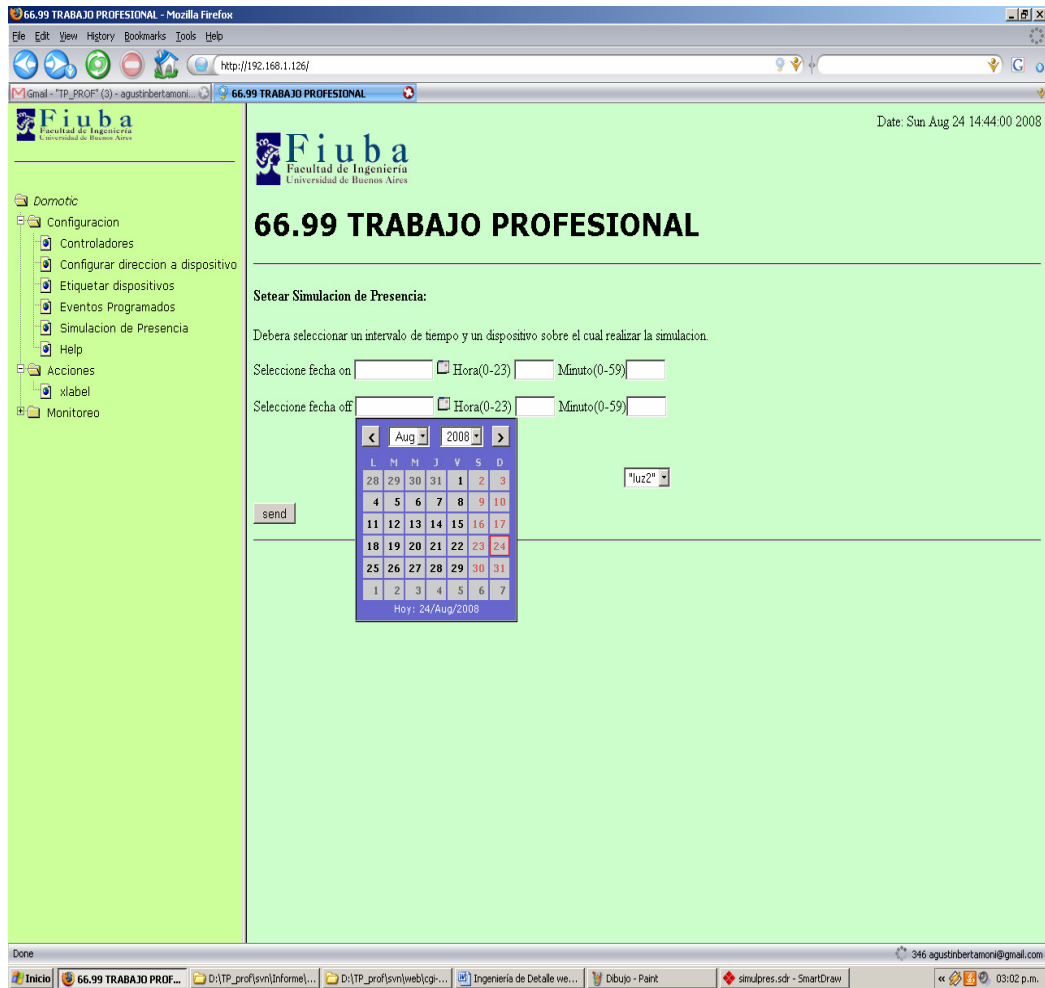


Figura 7.27: Pantalla de configuración de la Simulación de Presencia

6) Xlabel → xlabel.pl

Este programa muestra un dibujo con un formulario embebido donde seleccionando el estado(on/off) y una etiqueta sobre la cual aplicar el comando se puede encender o pagar directamente un dispositivo x10 o cualquier otro con solo seleccionar su etiqueta. La lógica de funcionamiento y la pantalla que se le presenta al usuario se muestra a continuación:

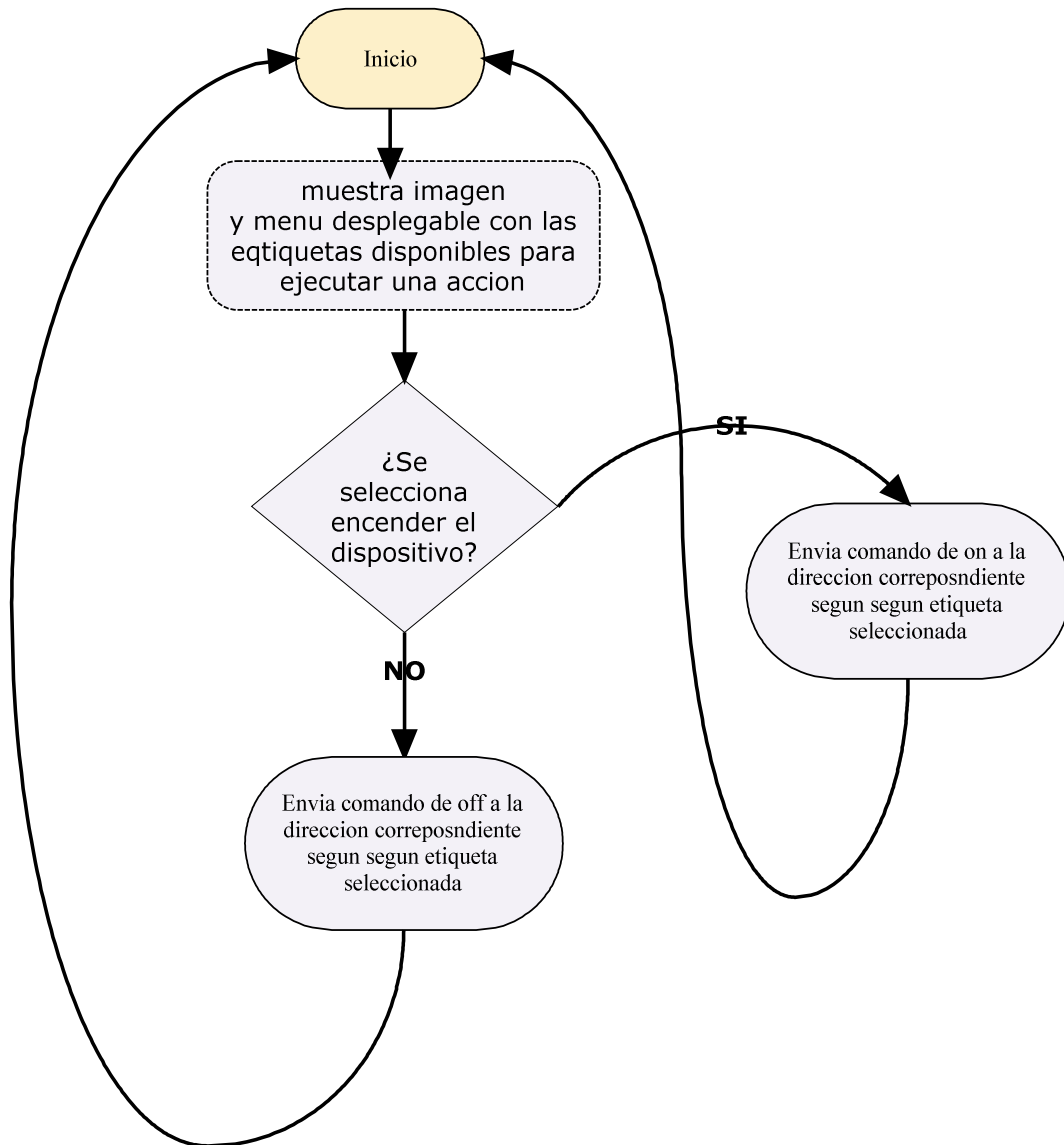


Figura 7.28: Diagrama de flujo programa *xlevel.pl*

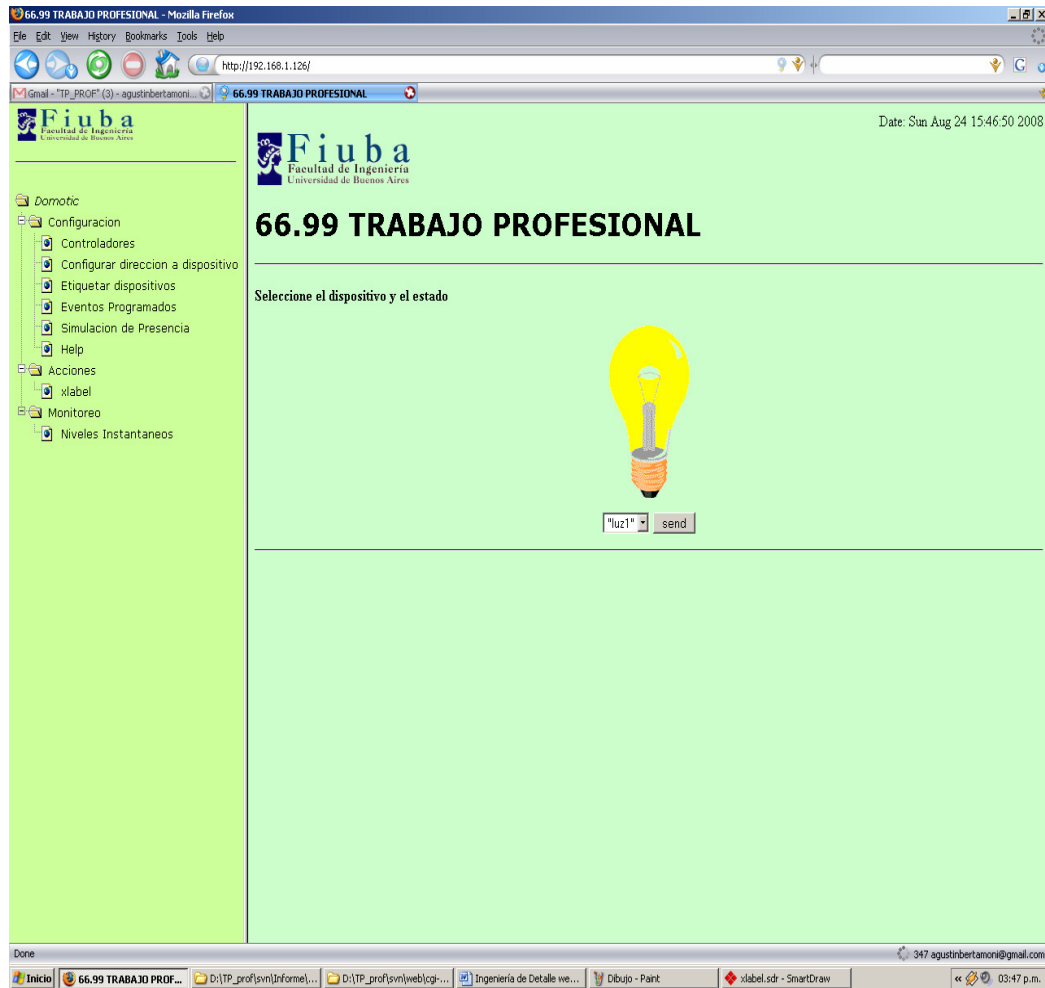


Figura 7.29: Pantalla de ejecución de las tareas instantáneas

7) Niveles instantáneos → monitor.pl

Este programa consulta a cada uno de los controladote por sus sensores activos mostrando el estado de los sensores en pantalla con un periodo de refresco de 5 segundos. La lógica de funcionamiento de este programa se muestra a continuación:

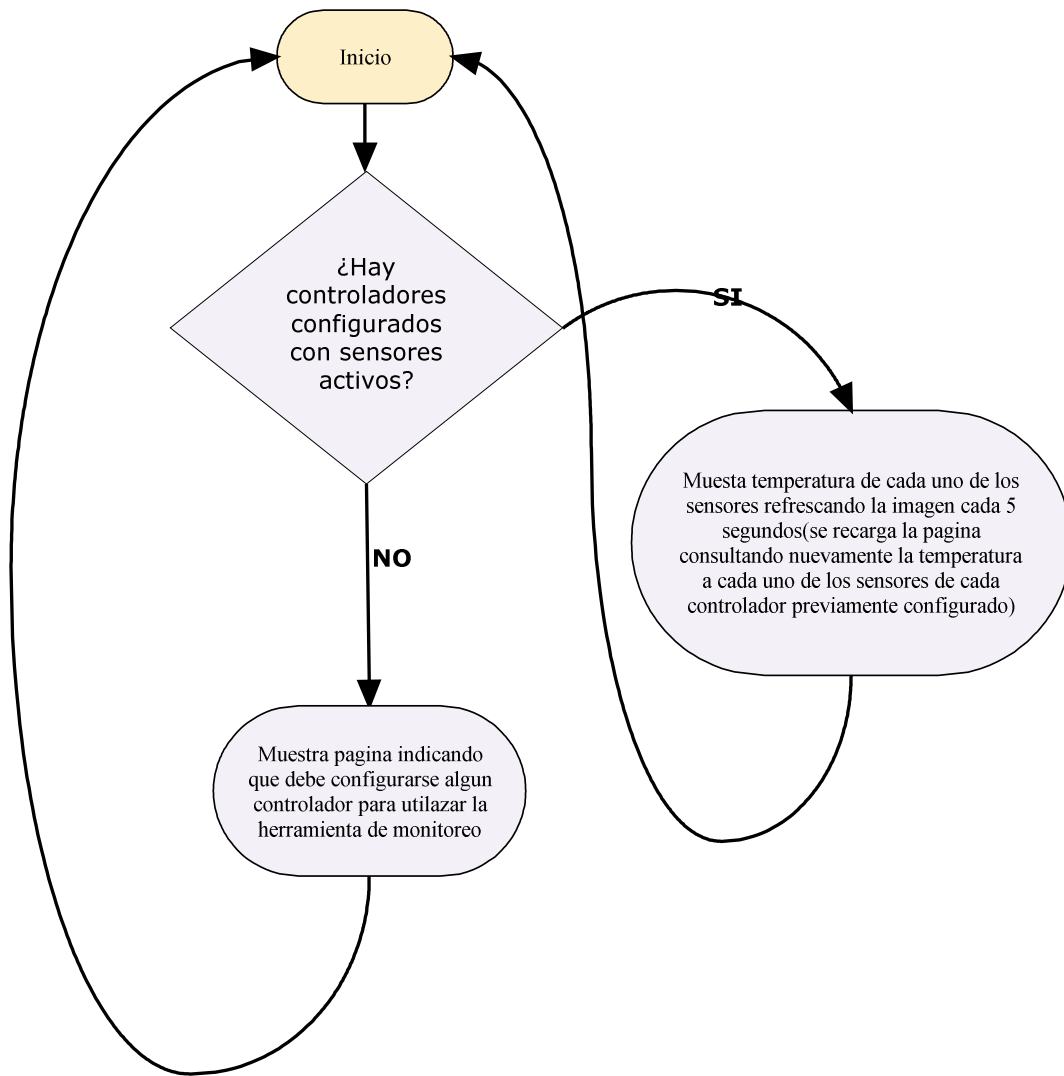


Figura 7.30: Diagrama de flujo programa monitor.pl

8. Diseño y construcción del prototipo

De acuerdo a lo establecido en las secciones precedente el sistema se compone de los siguientes equipos:

- Servidor Central basado en Servidor IBM x86 compatible
- Red de datos compuesta de puntos de acceso para dispositivos inalámbricos, switch de datos, router.
- Módulos controladores

En el caso de los primeros dos, esta sección se limitará a indicar los requerimientos mínimos requeridos para lograr que el sistema funcione correctamente, dado que son equipos de uso masivo que se adquieren con facilidad en el mercado local.

8.1. Módulo Controlador

8.1.1 Diseño placa base

Se desarrollará en esta sección el diseño de la placa base de módulo controlador. De acuerdo a lo desarrollado en las secciones anteriores, esta placa contiene los conectores para vincularla con los distintos componentes del sistema tales como:

- RCM4400W
- controlador X10-CM11
- dispositivos directamente controlados
- Consola de Administración

El circuito impreso fue diseñado basado en el diagrama esquemático de la figura 7.6. Para el desarrollo del mismo se utilizó el software OrCad Layout en la versión 10.5. Una vez realizado el diagrama esquemático se realiza el circuito impreso utilizando el software OrCad Layout 10.5, que toma como entrada el circuito esquemático anteriormente mencionado. Para la realización del circuito impreso se tomaron las siguientes premisas:

- El circuito se realizará en una sola placa de doble faz, que permite tanto una realización industrial en forma seriada del equipo final, como un desarrollo inicial para prototipo en formato manual utilizando la técnica de transferencia de toner sobre pertinax virgen
- La dimensión de la placa no puede exceder los 15 x 15 cm.

- El conector de 50 pin con el RCM4400W debe estar posicionado de forma tal que la antena quede sobre un lateral.
- Los dispositivos directamente controlados deben tener la salida a relay sobre un mismo lateral, distinto al que contiene la antena.
- Sobre un lateral distinto a los anteriores se colocarán los conectores para el controlador X.10-CM11 y la consola serial, siendo estos RJ11 y DB-9 respectivamente y hembras en ambos casos.
- Los sensores deben ser conectado mediante cables con termocontraible y ser ubicados sobre la carcasa, para que la potencia disipada no incida sobre la lectura del mismo.
- Todos los circuitos integrados deben utilizar encapsulado dip con el zócalo correspondiente para un fácil reemplazo en caso de ser necesario.

De esta forma el paquete de diseño arrojó el siguiente circuito. Donde la línea amarilla indica el límite de placa con las medidas correspondientes en centímetros.

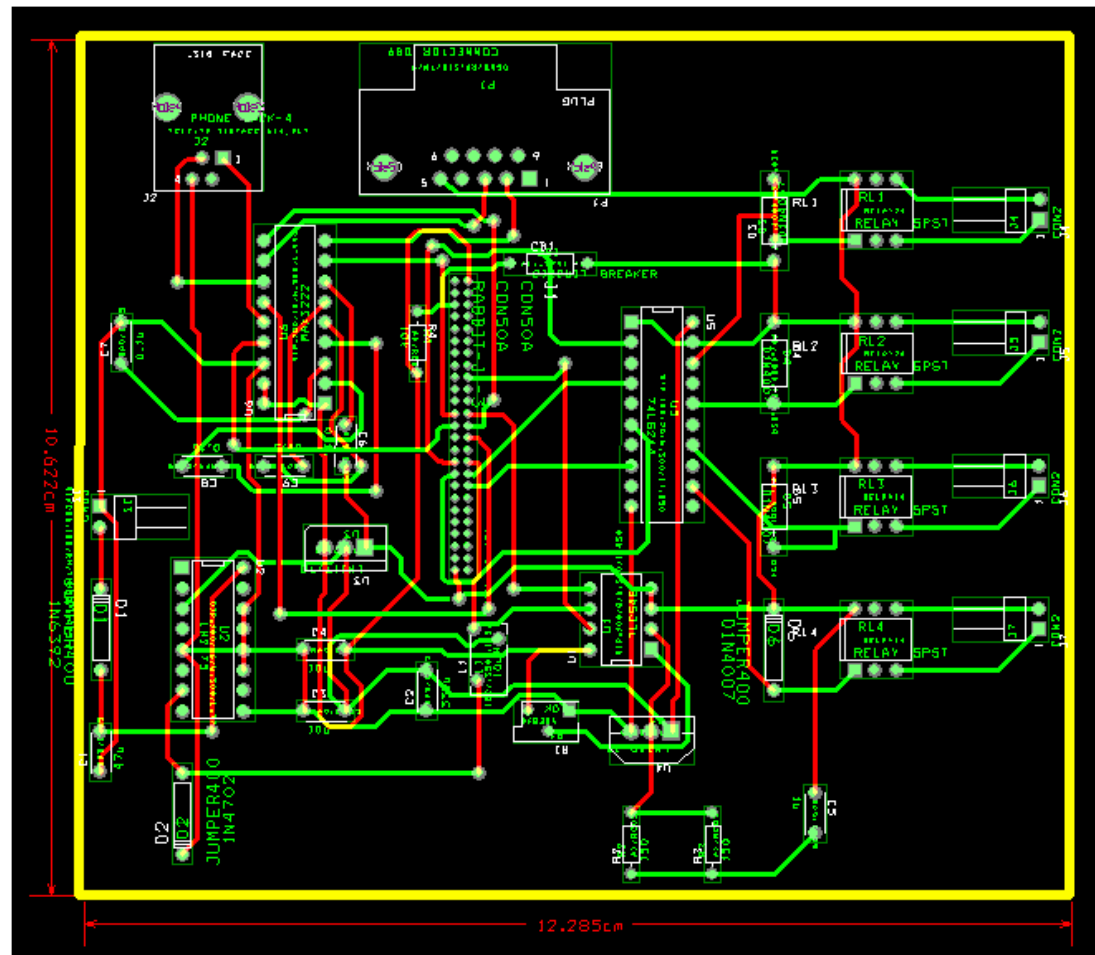
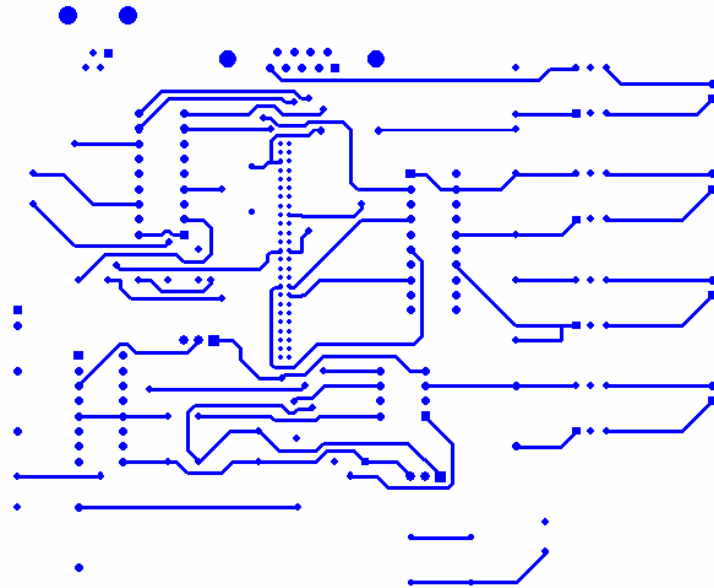
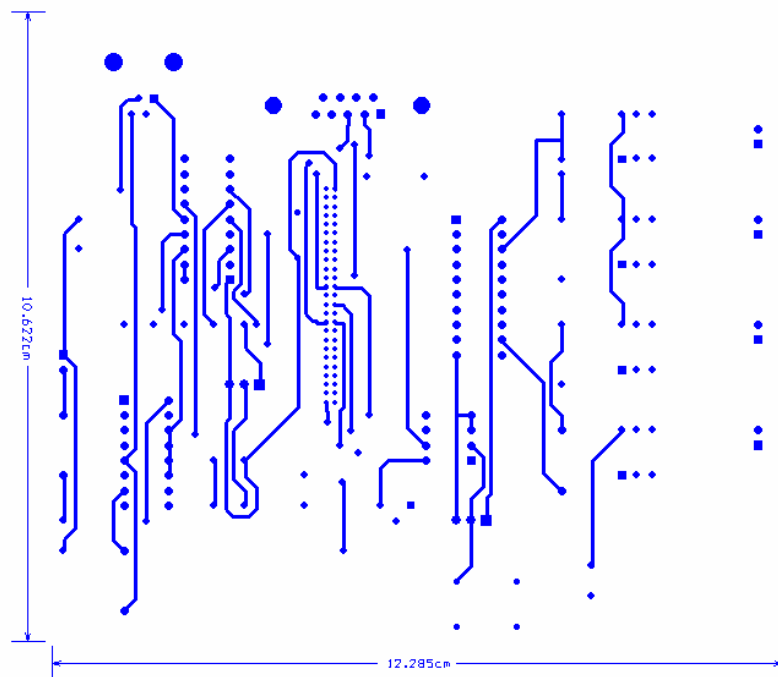


Figura 8.1: Circuito en programa Orcad Layout

Luego, para realizar el circuito impreso por la técnica de transferencia de toner antes mencionada se utiliza. Para ello se necesitan en forma aislada la cara superior y la cara inferior que se muestran en la figura 8.2 y 8.3 respectivamente

*Figura 8.2: Lado superior del circuito**Figura 8.3: Lado inferior del circuito*

Se necesita además para la fabricación de la placa un gráfico con todos los agujeros que deberán realizarse para insertar los componentes, el mismo se observa en la figura 8.4

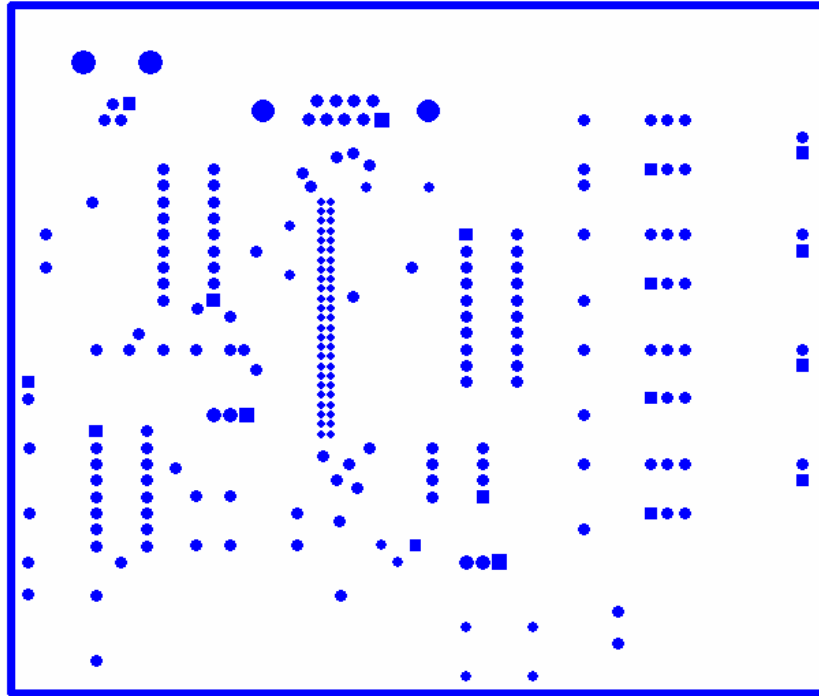


Figura 8.4: Capa de agujereado (drilling side)

Finalmente se realizó un Nuevo gráfico con la vista de los componentes para tener la referencia al momento de insertar los componentes en la placa. Se observa en la figura 8.5.

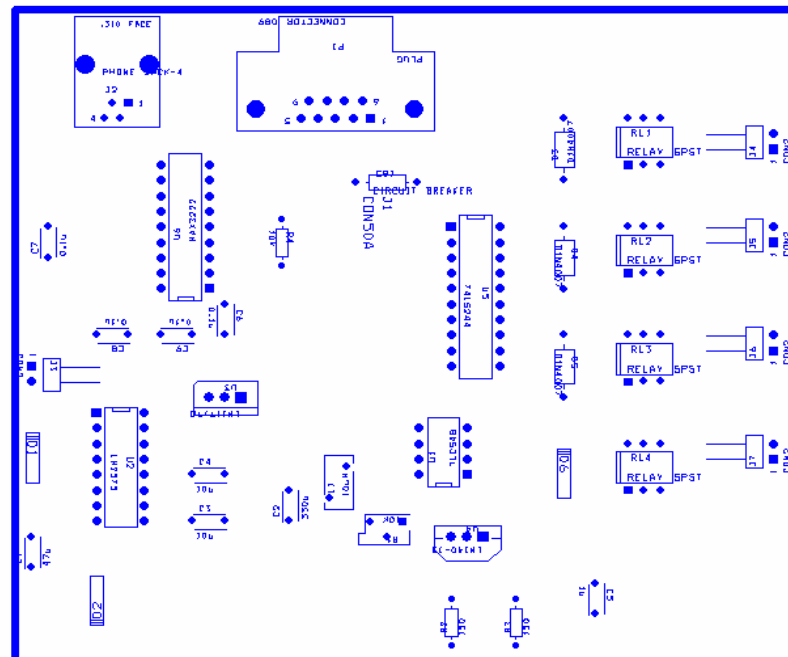


Figura 8.5: Capa de componentes

8.1.2 Listado de componentes de la placa base

Se enumera en esta sección el listado final de componentes utilizados en la placa base.

Módulo	cantidad	Designación	Valor
Fuente	1	C1	47uF
	1	C2	330uF
	2	C3,C4	10uF
	1	L1	330uH
	1	D1	DL4003
	1	D2	1N4702
	1	U2	LM2575
	1	U3	LM1117
	1	J3	Bornera 2 pines
Sensor de Temperatura	1	R1	pot 10K
	2	R2, R3	150
	1	C5	1u
	1	LM35	U4
	1	TLC548	U1
Conversor de Nivel	1	U6	Max3222
	4	C5,C6,C7,C8	0,1uF
	1	P1	Conector DB-9 hembra
	1	J2	Conector RJ11 hembra
Salidas a relay	1	U5	74LS244
	4	D1,D2,D3,D4	1N4007
			Relay TRK2233F
	4	RL1,RL2,RL3,RL3	5VCC
	4	J4,J5,J6,J7	conectores 2 pins

Tabla 8.1: Listado de componentes

Solamente falta agregar al listado el módulo RCM4400W y el módulo controlador, además de los dispositivos controlados X.10 que quedarán a elección del cliente.

8.1.3 Consumo

8.1.4 Especificaciones de Montaje

Diseñada la placa base, para el montaje se deben tener en cuenta los siguientes parámetros:

- El controlador completo se montará sobre una estructura plástica no conductora cuyas dimensiones no podrán superar los 15 x 12 x 7cm, correspondiendo las medidas a largo, ancho y profundidad respectivamente.
- El controlador X.10-CM11 se deberá desmontar de su estructura y ajusta sobre la parte superior de la estructura de montaje. Se deberán utilizar tornillos con aislaciones similares a las utilizadas en la estructura original del controlador
- La estructura deberá contener los orificios para la antena del RCM4400W y para los leds de encendido y conectividad que vienen en el propio módulo Rabbit para indicar los estados correspondientes con facilidad.
- Se utilizará un único conector del tipo australiano para alimentar la estructura. El conector deberá estar certificado bajo las normas IRAM. El mismo será compartido por la fuente y por el controlador X.10.
- La estructura deberá contener los orificios para afirmarla a la pared y ajustarla con tarugos. Sin embargo, deberá tener la suficiente resistencia mecánica para poder afirmarse utilizando como único sustento al conector eléctrico. Bajo condiciones de productividad el equipo deberá estar afirmado, sin embargo para realizar las primeras pruebas es importante que pueda autosujetarse.

En la figura siguiente (8.6) se observa un gráfico de un prototipo que cumple con los requerimientos antes mencionados diseñado por computadora.

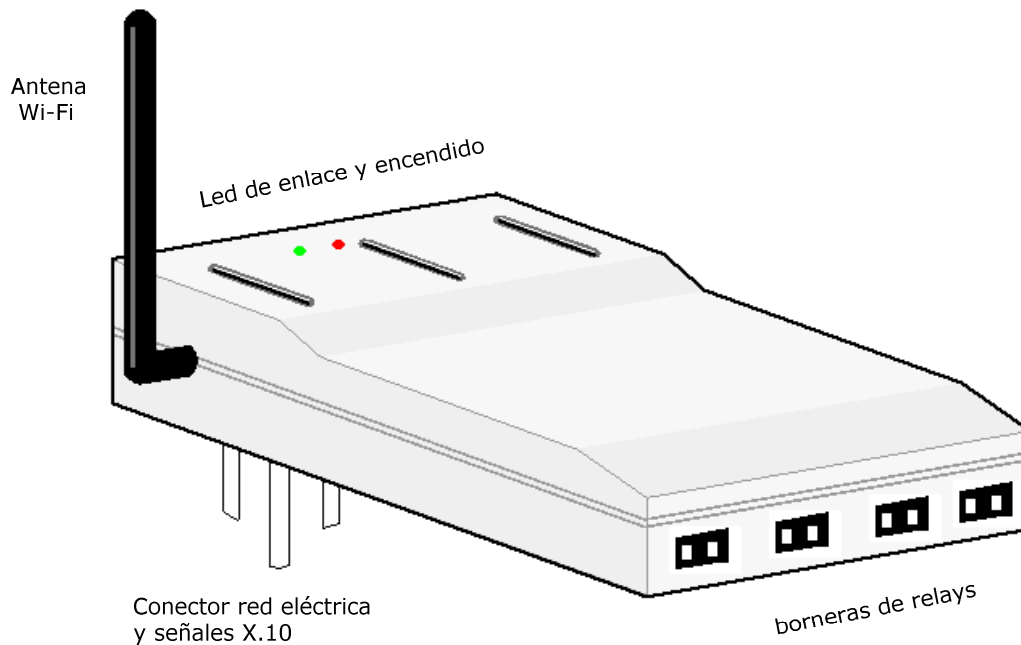


Figura 8.6: Módulo controlador diseñado por computadora

8.2. Servidor

El servidor a utilizar deberá tener la capacidad para albergar un sistema operativo GNU Linux, el cual deberá conectarse con Internet para recibir conexiones entrantes y albergar un servidor web basado en Apache Tomcat. Para ello, se necesita un servidor que cumpla con las siguientes características:

- Arquitectura x86 IBM compatible
- Procesador Pentium III 800 MHz o Superior
- 256 Mb de memoria RAM o superior
- Placa de Red 10/100Mbps
- Fuente de alimentación redundante (opcional)

Se puede observar que los requerimientos del equipo son muy bajos. Esto es porque el sistema operativo no trabajará en modo gráfico, dado que consume mayores recursos y no es necesario, por lo que cualquier computadora del mercado cumplirá sobradamente los requerimientos mínimos. Se recomienda además una fuente redundante dado que el equipo permanecerá encendido ininterrumpidamente y puede causar fallas en el servidor en forma ocasional.

8.3. Red de datos

La red de datos deberá contar con las siguientes características:

- Router con capacidad de firewall y de redireccionar puertos de entrada y capacidad de conectarse con un DNS dinámico para actualizar el registro de dirección IP.
- Switch de datos con un mínimos de 4 puertos
- Un punto de acceso cada 50 mts en exterior y cada 25 mts en interior.

La red de datos se deja a elección del cliente. Se realiza solamente una evaluación para determinar si el sistema funcionará correctamente bajo la red disponible. La evaluación constará de medir si la señal Wi-Fi en los lugares donde se van a instalar los módulos controladores tiene la suficiente potencia para mantener el enlace sin cortes. Es importante mencionar que al igual que en el servidor, los requerimientos de la red los cumplen la mayoría de los routers Wi-Fi de uso doméstico, por lo que no resulta un inconveniente ni difícil de conseguir. Se muestra a continuación dos diagramas de red, uno con una red de uso doméstico y otro con una red de uso profesional, ambos perfectamente compatibles con el sistema.

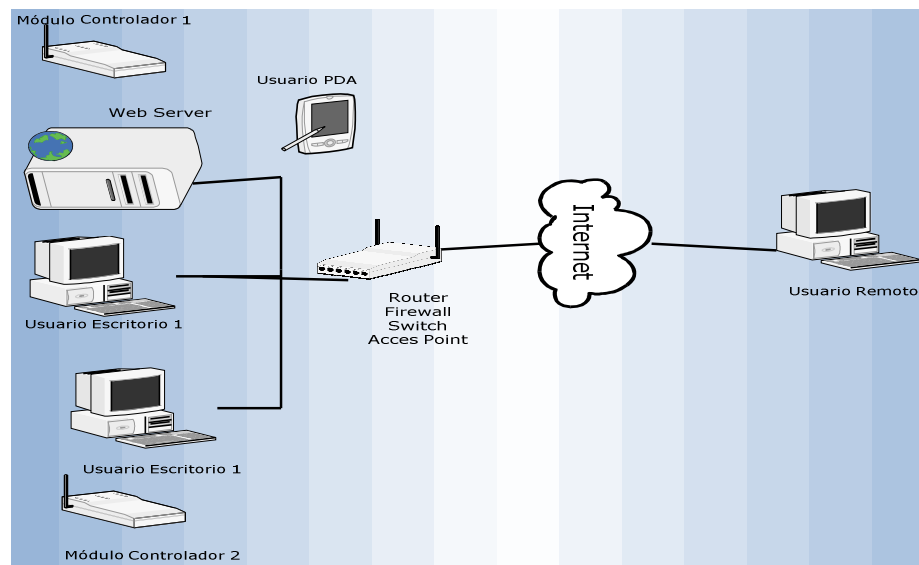


Figura 8.8: Red de uso doméstico

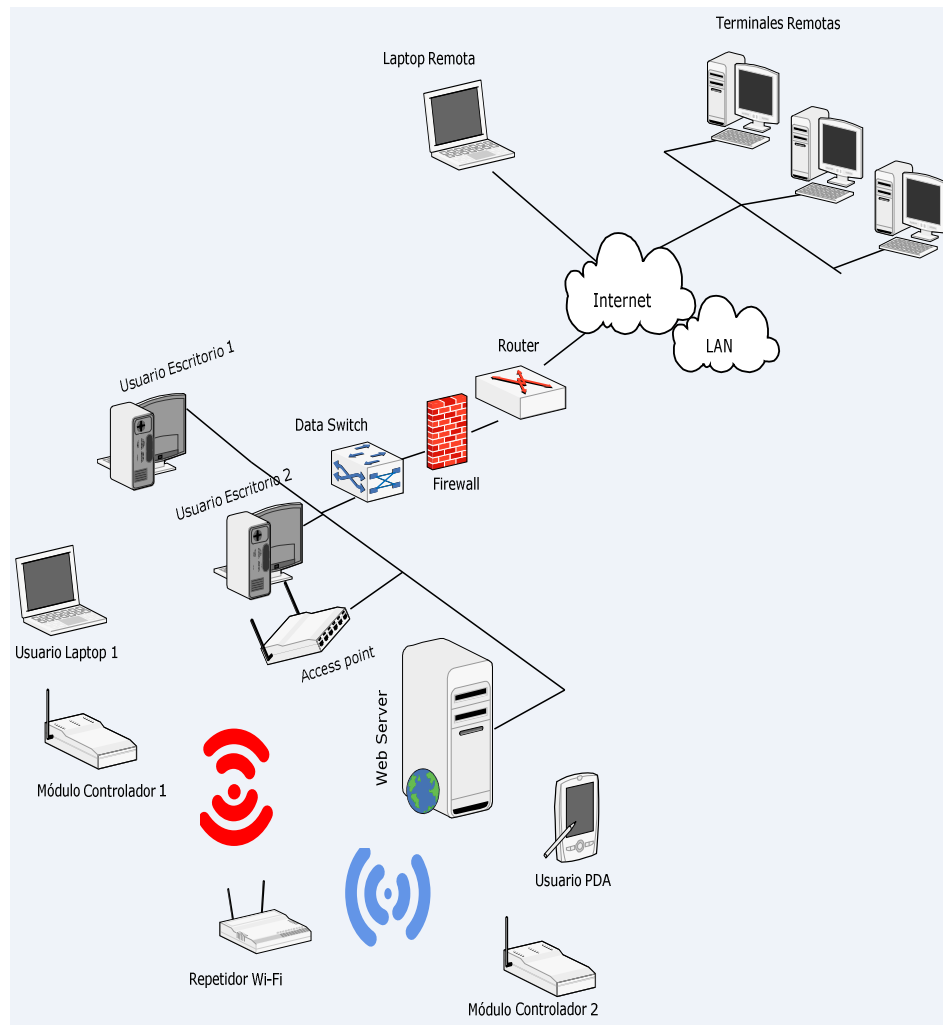


Figura 8.7: Red de uso profesional

Según se puede observar, la diferencia entre ambas redes se encuentra en que la red doméstica tiene todos los servidores de red (router, firewall, acces point, switch) en un mismo dispositivo, denominado normalmente “router wi-fi” mientras que en la red de uso profesional tiene un dispositivo encargado de cada una de las funciones. Además la red de uso profesional tiene repetidores Wi-Fi. Estos repetidores permiten colocar módulos controladores en regiones alejadas del Access Point principal sin la necesidad de cablear un nuevo access point. Además permiten a usuarios conectarse a la red local desde zonas de la residencia alejadas de la región principal (por ejemplo, en un patio) con el objetivo de controlar los dispositivos o inclusive ingresar a Internet. Si bien no se muestran en la red de uso doméstico, los repetidores también pueden ser colocados en ese tipo de red.

La red de uso profesional es mucho más robusta que la red de uso doméstico, dado que permite mayor cantidad de conexiones simultáneas dado que cada dispositivo tiene asignada una tarea específica. Además, se pueden colocar mayor cantidad de políticas de seguridad.

Para el funcionamiento del sistema, el usuario deberá conectarse al Web Server. Desde la red local, esto se puede hacer conectándose a la dirección IP o el nombre del servidor del mismo (en caso de tener un servidor de nombres DNS) desde

el explorador de Internet. Sin embargo, para conectarse en forma remota se deberán cumplir los siguientes dos parámetros:

1. El cliente debe tener una dirección IP pública estática o debe tener un cliente de un servidor de nombres dinámico (DynDNS). Dado que la mayoría de los proveedores de Internet (ISP) cobran un cargo adicional por una dirección IP estática, la segunda opción será la mas sencilla de aplicar. Servidores gratuitos de DynDNS se pueden encontrar en:
 - www.dyndns.org
 - www.no-ip.comentre otros. Para utilizarlos se debe instalar el cliente que provee el proveedor del servicio. La función del DynDNS es cambiar la asignación de dirección IP-Nombre en los DNS principales a medida que el ISP cambia la dirección IP pública asignada.
2. El firewall debe redireccionar el puerto 80 al puerto correspondiente del web Server, de forma tal que el usuario ingrese la dirección IP pública (o nombre) en el explorador ingrese al webserver.

8.4. Costos

En esta sección se analizan los costos finales del desarrollo del sistema. Se analiza solamente el costo de un módulo controlador debido a las siguientes razones:

1. El servidor es provisto por el cliente. Además los costos de los mismos varían notablemente de acuerdo a las características de los mismos. El costo mínimo al momento de la redacción del informe corresponde a US\$200.
2. No se incluye el costo de los dispositivos X.10 controlados, dado que dependerá cuales de ellos el cliente desea instalar. Para tener una aproximación parten desde los US\$ 45 para los módulos de luz empotrados de doble comando, hasta los US\$ 250 para controlar el motor de una persiana eléctrica

Se aclara además, que el sistema puede ser compuesto por mas de un módulo controlador, lo cual será normal en la mayoría de los casos.

8.4.1 Costo componentes módulo controlador

La siguiente tabla enumera los componentes utilizados en el módulo controlador con el costo individual y total de cada uno de ellos.

cantidad	Valor	Costo Unitario (US\$)	Total (US\$)
4	Capacitor 0,1uF	0,033	0,132
1	Capacitor 1uF	0,02	0,02
2	Capacitor 10uF	0,02	0,04
1	Capacitor 47uF	0,025	0,025
1	Capacitor 330uF	0,032	0,032
1	Inductor 330 uH	0,25	0,25
1	Diodo DL4003	0,1	0,1
1	Diodo Zener 1N4702	0,1	0,1
4	Diodo 1N4007	0,05	0,2
2	Resistencia 150	0,01	0,02
1	Preset 10K	0,35	0,35
4	Relay TRK2233F 5VCC	1	4
5	Bornera 2 pines	0,4	2
1	Conector RJ11 Hembra	0,3	0,3
1	Conector DB-9 hembra	0,8	0,8
1	Fuente de switchingLM2575	2,1	2,1
1	Regulador linealLM1117	0,4	0,4
1	Sensor de temperature LM35	1,6	1,6
1	Conversor A/D TLC548	3	3
1	Driver 74LS244	0,2	0,2
1	Conversor de nivel Max3222	4	4
1	Módulo Microcontrolador		
1	WiFi RCM4400W	90	90
1	Controlador X.10 CM11	80	80
			189,669

8.4.2 Costo placa y soldadura módulo controlador

Para una producción en serie del módulo controlador se realizará encargará a proveedores la producción de las placas en forma industrial para disminuir costos. Las características del diseño de las placas impresas será la siguiente:

- Sn/Pb: Estaño Selectivo (Nivelado por aire Caliente)
- PTH: Plated-Through Hole (Hojalillo metalizado)
- Impresión de Componentes para reducir errores humanos de montaje

De esta forma, la tabla 8.3 muestra los costos de producción de las mismas:

Item	Valor
cantidad de circuitos impresos	200
costo de la matriz	60
costo por unidad	10

Tabla 8.3: Costos del circuito impreso

Concluyendo el costo del módulo controlador en aproximadamente US\$ 200 por unidad.

Este costo estipula el no uso de plomo como base para las soldaduras, para contribuir de esta forma a la prevención del medio ambiente. Se toma la decisión de aumentar levemente los costos del circuito en función de eliminar la posibilidad de una prohibición de la comercialización del producto en un futuro, posibilitar la comercialización del producto en la Unión Europea y países que tengan reglamentaciones respecto del contenido de plomo en los equipos electrónicos. Además evita posibles inconvenientes legales y favorecerá su comercialización al ser considerado un producto ecológico.

8.5 Disposición final del equipo

E-Scrap es el término utilizado para denominar a la basura electrónica. Si bien el país no cuenta con una legislación específica indicando el tratamiento de los residuos electrónicos¹, existe la Ley Nacional N° 24.051 de Residuos Peligrosos vigente desde 1992 y reglamentada. Para dar cumplimiento a esa Ley, se tomaron las precauciones determinadas en el punto anterior con el objetivo de anular o minimizar el contenido de sustancias contaminantes. Además, el diseño del producto se orientará a que pueda ser reciclado y separado en distintos componentes una vez que sea obsoleto para su posterior venta por separado de los metales y placas constituyentes como insumos de otros equipos. Para este fin, en Argentina empresas como Silkers S.A. se dedican a la recolección y reciclado de basura electrónica.

¹Fuente: http://www.limpiezaprofesional.net/notas/basura_electronica.pdf

8.6. Consumo

Si bien el equipo se conecta a la red eléctrica, por lo cual no tiene problemas de alimentación, se realiza el cálculo del consumo para ubicarlo entre los requerimientos del sistema. El consumo de la placa base se calcula de acuerdo al consumo de cada uno de sus elementos individuales. El resultado se observa en la tabla 8.4

dispositivo	Cantidad	Unitario (mW)	total (mW)
Sensor de temperatura LM35	1	10	10
Conversor A/D TLC548	1	15	15
Driver 74LS244	1	135	135
Conversor de nivel Max3222	1	0,0055	0,0055
Módulo Microcontrolador WiFi RCM4400W	1	450	450
Relay	4	35	140
			750,0055

Tabla 8.4: Consumo del circuito

Con lo que se puede comprobar que el circuito utilizado para la fuente escogido en 1A fue elegido correctamente. Cabe destacar que los valores de corriente tomados son en el peor caso, es decir con cada uno de los circuitos en modo activo, por lo que el consumo medido bajo circunstancias normales será menor.

8.7 Plan de prueba de Hardware y software

Todo prototipo durante su desarrollo y una vez terminado debe ser sometido a pruebas para evaluar su fiabilidad la cual se verá afectada por las fallas que puedan ocurrir. Las fallas impiden que el equipo realice las funciones para las cuales se diseñó y esto es un hecho indeseable.

Para poder hacer diferentes pruebas mientras desarrollamos el prototipo implementamos un plan de pruebas donde de manera eficiente probamos conjuntamente hardware y software con el objeto de:

- Buscar anomalías en el funcionamiento bajo diferentes condiciones.
- Validar y facilitar el versionado del software para el controlador (firmware).
- Validar interoperatividad de funciones.
- Verificar funcionamiento con más de un controlador comandado desde la misma interfaz de configuración y mantenimiento.
- Forzar funciones en las cuales intervengan componentes con tasas de fallas altas.

- Tratar de validar el tiempo medio entre fallas(deberá ser similar al calculado con los datos suministrados por los fabricantes de cada componente del prototipo)

Según lo calculado anteriormente los componentes con mayor tasa de fallas son los relés. Además su tiempo medio entre fallas decrece notablemente en función del aumento de las solicitudes. Para el caso de fallas por solicitud es posible establecer un plan de pruebas para el módulo de relés donde se obliga a los módulos de E/S a conmutar sus salidas una vez por segundo durante un cierto tiempo. Esta prueba es muy exigente para componentes con movimiento mecánico como los relés y pretende determinar en un tiempo reducido la tasa de fallas.

La fiabilidad de los relés cae rápidamente pasadas las 10 solicitudes por hora, una acción por minuto es el valor máximo que permitimos que el usuario pueda configurar desde la interfaz de configuración, esto indica que en un año dicho componente computa un total de:

$$S(1\text{año})=365 \text{ días} * 24 * 60 \text{ conmutaciones(max)}=525600$$

Teniendo en cuenta que el rele puede conmutar de manera normal cada 1 segundo alcanzando así 3600 conmutaciones por hora, podemos simular un año de operación normal en 146 horas de prueba, con el fin de una prueba integral diseñamos un softw adicional que nos permite además de hacer estas pruebas incluir una serie de funciones que pueden ser testeadas de manera alternada cada cierto numero de conmutaciones de una salida, probando así la totalidad del hardware y software. Mediante este test de regresión se pueden probar entre otros los siguientes componentes del prototipo:

- Interfaces y protocolos de comunicación entre el controlador y el motor de control web.
- Interfaces y protocolos de comunicación entre el controlador y el modulo de control X10.
- Interfaces de salida analógica sobre la que se conectan todo tipo de dispositivos
- Conjunto de protocolos montados sobre interfaz 802.11.
- Métodos y procedimientos involucrados en la ejecución de una tarea programada.
- Prueba con múltiples dispositivos X10 y hasta 5 dispositivos directamente conectados.
- Rutinas de logueo de acceso de usuarios y errores de todas las partes del sistema.
- Procedimientos y rutinas de escape en caso de error o falta de comunicación entre alguno de los componentes del prototipo.

El código fuente utilizado para las pruebas junto con la ayuda e instrucciones se encuentra disponible en el apéndice A bajo el titulo “fulltest.pl”.

9. Estudios de Confiabilidad

La confiabilidad se relaciona con la probabilidad de que el equipo funcione normalmente durante un período de tiempo en un ambiente determinado, ya sea en uso o en almacenamiento.

Otros conceptos relacionados son:

- Vida útil: definida como el tiempo entre su fabricación y cese de utilización.
- Falla: definida como apartamiento de alguna de sus especificaciones que impiden el funcionamiento correcto del equipo.
- Tasa de fallas: definida como la cantidad de fallas en un intervalo de tiempo dado.

9.1 Cálculo de confiabilidad de Hardware

Para conocer la tasa de fallas de un dispositivo electrónico se debe recurrir a algún modelo que permita predecirla. Aquí se recurre a las normas militares los cuales se basan en las siguientes hipótesis:

- El modelo de fallas sigue una distribución exponencial
- La tasa particular de fallas de cada componente es constante

Para estimar la tasa de fallas de cada componente se utilizará el manual HDBK-217 desarrollado por las Fuerzas Armadas de los Estados Unidos.

El hardware corresponde al módulo controlador el cual consiste en la placa base donde se coloca todos los elementos periféricos al módulo embebido RCM4400 y el módulo en sí. Se realizan los estudios de confiabilidad para ambas partes

9.1.1 Placa base

En esta sección se realiza el análisis de confiabilidad de todos los dispositivos contenidos en la placa base del módulo controlador.

9.1.1.1 Resistores

En la placa base se utiliza 4 resistores de 1/8 de Watt de potencia máxima y 5% de tolerancia respecto de su valor nominal. El militar Handbook HDBL-217 indica la siguiente fórmula en la sección 9:

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_R \cdot \pi_E \cdot \pi_Q \cdot \text{fallas}/10^6 \text{ horas}$
Factor de stress	0,7 operando al 90% de su potencia máxima

Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,0017	debido a que son todos menores a 1MΩ Tomando el peor caso
Factor Resistivo	π_R	1	
Factor de Calidad	π_E	15	
Factor Ambiente	π_Q	2	
Tasa de Fallas		0,051	

Tabla 9.1: Fiabilidad de los resistores

9.1.1.2 Capacitores

Se utilizaron 9 capacitores en la placa base: 4 cerámicos de 0,1μF y 5 electrolíticos de 1, 10 (2 de ellos), 47 y 330 μF.

En la tabla 9.2 se calcula la confiabilidad para los capacitores cerámicos y en la 9.3 se calcula la confiabilidad para los capacitores electrolíticos en sus distintas capacidades.

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_{CV} \cdot \pi_E \cdot \pi_Q \cdot \text{fallas}/10^6 \text{ horas}$		
Factor de stress	0,7 operando al 90% del voltaje máximo		
Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,01	
Factor capacitivo cerámico	π_{CV}	1,12	para C = 10 nF
Factor capacitivo cerámico	π_{CV}	1,45	para C = 100 nF
Factor de Calidad	π_Q	10	Tomando el peor caso
Factor Ambiente	π_E	2	
Tasa de Fallas 10 nF		0,224	
Tasa de Fallas 100nF		0,29	

Tabla 9.2: Fiabilidad de los capacitores cerámicos

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_{CV} \cdot \pi_E \cdot \pi_Q \cdot \pi_{SR} \cdot \text{fallas}/10^6 \text{ horas}$		
Factor de stress	0,7 operando al 90% de su potencia máxima		
Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,035	
Factor capacitivo electrolítico	π_{CV}	0,34	C = 1μF
Factor capacitivo electrolítico	π_{CV}	0,5	C = 10μF
Factor capacitivo electrolítico	π_{CV}	0,65	C = 47μF
Factor capacitivo electrolítico	π_{CV}	0,9	C = 330μF
Factor de Calidad	π_Q	2	Tomando el peor caso
Factor Ambiente	π_E	2	
Factor Resistivo	π_{SR}	0,2	valor medio
Tasa de Fallas 1μF		0,00952	
Tasa de Fallas 10μF		0,014	
Tasa de Fallas 47μF		0,0182	

Tasa de Fallas 330 μ F	0,0252
----------------------------	--------

Tabla 9.3: Fiabilidad de los capacitores electrolíticos

9.1.1.3 Inductores

En el circuito se utilizó un único inductor para la fuente de switching. La tasa de fiabilidad del mismo se presenta en la tabla 9.4.

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_Q \cdot \pi_P \cdot \pi_E$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,00071	
Factor conexión/desconexión	π_C	1	valor fijo
Factor de pines activos	π_Q	20	la menor calidad
Factor Ambiente	π_E	1	
Tasa de Fallas		0,0142	

Tabla 9.4: Fiabilidad de inductores

9.1.1.5 Diodos

El circuito contiene diodos conectados en inversa para proteger los circuitos de excitación de los relays y los utilizados por la fuente. La tasa de fallas de los mismos se puede observar en la tabla 9.5.

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_T \cdot \pi_S \cdot \pi_C \cdot \pi_Q \cdot \pi_E$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,0038	Analógico de uso general
Factor de Temperatura	π_T	1,6	T = 50 °C
Factor de Stress	π_S	1	Analógico de uso general
Factor de Calidad	π_Q	8	Encapsulado plástico
Factor de Ambiente	π_E	2	
Tasa de Fallas		0,01216	

Tabla 9.5: Fiabilidad de diodos

9.1.1.5 Circuitos Integrados

La placa base contiene distintos circuitos integrados para realizar las funciones necesarias. Dadas las distintas características de cada uno de ellos las tasas de fiabilidad de ellos serán distintas. Sin embargo para todos se utiliza la siguiente fórmula.

$$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q \text{ fallas/10}^6 \text{ horas}$$

Los circuitos utilizados son:

- LM2575 (fuente de switching)
- LM1117 (fuente lineal de 3,3v)
- TLC548 (convertor analógico digital)
- MAX3222 (convertor de niveles RS-232 – 3,3v)
- 74LS244 (driver para excitación de relays)

- LM35 (sensor de temperatura)

9.1.1.5.1 LM2575

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Indice de complejidad	C ₁	0,01	Dispositivo de menos de 100 transistores Dispositivo con empaquetado DIP 5 pin activos
Falla de empaquetamiento	C ₂	0,0021	
Factor de temperatura	π_T	0,61	12 años de antigüedad
Factor de aprendizaje	π_L	0,031	
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,000319		

Tabla 9.6: Fiabilidad de LM2575

9.1.1.5.2 LM1117

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Indice de complejidad	C ₁	0,01	Análogo de uso general encapsulado plano de 3 pines
Falla de empaquetamiento	C ₂	0,00022	
Factor de temperature	π_T	0,71	10 años de antigüedad
Factor de aprendizaje	π_L	0,063	
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,000475		

Tabla 9.7: Fiabilidad de LM1117

9.1.1.5.3 TLC 548

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/10 ⁶ horas		
Indice de complejidad	Símbolo	Valor	Observaciones
Indice de complejidad	C ₁	0,02	Circuito digital con 101 a 1000 compuertas
Falla de empaquetamiento	C ₂	0,0034	
Factor de temperature	π_T	0,43	25 años de antigüedad
Factor de aprendizaje	π_L	0,00034	
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	5.24E-06		

Tabla 9.8: Fiabilidad de TLC548

9.1.1.5.4 MAX3222

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Índice de complejidad	Símbolo	Valor	Observaciones
Índice de complejidad	C_1	0,02	Circuito digital con 101 a 1000 compuertas
Falla de empaquetamiento	C_2	0,0082	SSOC 18 pines
Factor de temperature	π_T	0,43	
Factor de aprendizaje	π_L	0,25	6 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,00625		

Tabla 9.9: Fiabilidad de TLC548

9.1.1.5.5 74LS244

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Índice de complejidad	Símbolo	Valor	Observaciones
Índice de complejidad	C_1	0,01	Circuito digital con 1 a 100 compuertas
Falla de empaquetamiento	C_2	0,009	DIP 20 pines
Factor de temperature	π_T	0,43	
Factor de aprendizaje	π_L	0,00094	22 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	2,1E-05		

Tabla 9.10: Fiabilidad de 74LS244

9.1.1.5.6 LM35

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Parámetro	Símbolo	Valor	Observaciones
Índice de complejidad	C_1	0,01	Analógico de uso general encapsulado plano de 3 pines
Falla de empaquetamiento	C_2	0,00022	
Factor de temperatura	π_T	0,71	
Factor de aprendizaje	π_L	0,128071	7 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,000966		

Tabla 9.11: Fiabilidad del LM35

9.1.1.5 Relays

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_L \cdot \pi_C \cdot \pi_{CYC} \cdot \pi_F \cdot \pi_Q \cdot \pi_E$ fallas/10 ⁶ horas		
Índice de complejidad	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,0066	relay clase F a 50°C
Factor de Stress	π_L	2,72	Para una carga de 2ª
Forma de Contacto	π_C	1,75	Contactos SPDT
Factor Cíclico	π_{CYC}	1,2	1 ciclo cada 5 minutos
Factor de construcción	π_F	12	Con solenoide menor de 5ª
Factor de Calidad	π_Q	3	No se conoce
Factor Ambiente	π_E	5	Especificación no military
Tasa de Fallas	6,785856		

Tabla 9.11: Fiabilidad de los Relays

Dado que el tiempo medio entre fallas varía drásticamente en función de las solicitaciones por hora, se hace un gráfico que demuestra que a partir de las 20 solicitaciones por hora (según la norma HDBK-217) existe una aceleración en la cantidad de fallas, por lo que se sugiere al usuario no pasar esta cantidad. Dado el uso para el cual está pensado el sistema, tal requerimiento no es necesario.

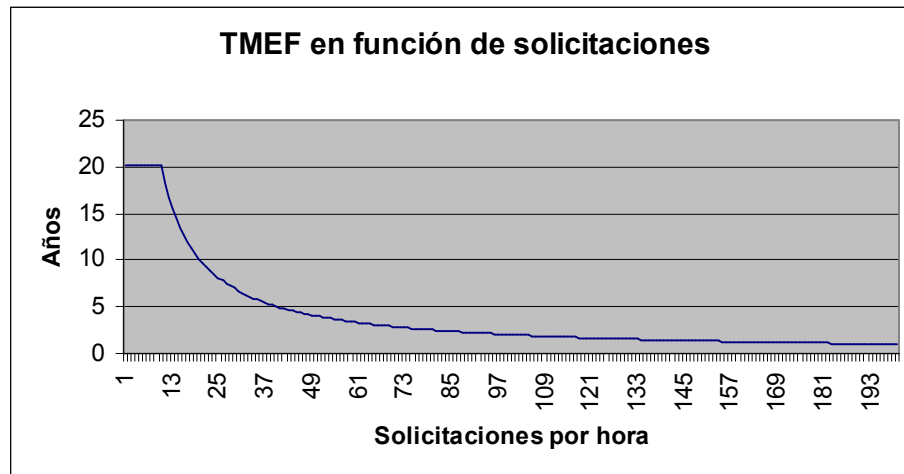


Figura 9.1: Tiempo medio entre fallas en función de las solicitaciones del relay

9.1.1.6 Conectores

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_K \cdot \pi_P \cdot \pi_E$ fallas/ 10^6 horas		
Índice de complejidad	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,0066	Relay clase F a 50°C
Factor de conexión desconexión	π_K	2,72	Para una carga de 2ª
Factor de conexión desconexión	π_K	1	Para cargas menores que 0,5 A
Factor de pines activos	π_P	1,4	2 pines por conector
Factor de pines activos	π_P	9,5	50 pines por conector
Factor ambientes	π_E	2	Calidad no military
Tasa de Fallas 2 pin	0,050266		
Tasa de Fallas 50 pin	0,1254		

Tabla 9.12: Fiabilidad de los Relays

9.1.1.7 Soldaduras

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_P \cdot \pi_E$ fallas/ 10^6 horas		
Índice de complejidad	Símbolo	Valor	Observaciones
Tasa base de fallas	λ_b	0,0026	Soldadura Manual
Factor de calidad	π_K	20	Peor caso
Factor ambientes	π_P	2	
Tasa de Fallas	0,104		

Tabla 9.13: Fiabilidad de los Relays

9.1.1.8 Tasa de fallas total Placa Base

Se calcula la tasa total de fallas para la placa base. Dado que los Relays son el factor que mayor cantidad de fallas aportan, y la gran diferencia en cantidad de fallas que existe de acuerdo a la cantidad de solicitudes por hora, es que se realiza un cálculo para un valor recomendado menor de 10 solicitudes por hora y un valor de 20 solicitudes por hora.

Componente	Cantidad	Tasa de Fallas por componente (fallas/ 10^6 horas)	Total
Resistor	4	0,0017	0,204
Capacitor cerámico 100nF	4	0,29	1,16
Capacitor electrolítico 1µF	1	0,0095	0,0095
Capacitor electrolítico 10µF	2	0,014	0,028
Capacitor electrolítico 47µF	1	0,0182	0,0182
Capacitor electrolítico 330µF	1	0,0252	0,0252

Inductor 10μH	1	0,0142	0,0142
Diodos	6	0,01216	0,07296
LM2575	1	0,000319	0,000319
LM1117	1	0,000475	0,000475
TLC548	1	5,24E-05	0,0000524
MAX3222	1	0,00625	0,00625
74LS244	1	2,10E-05	0,000021
LM35	1	0,000966	0,000966
Relay < 10 sol/hora	4	6,78	27,12
Relay 20 sol/hora	4	11,3	45,2
Conector 2 pin	4	0,05	0,2
Conector 50 pin	1	0,1254	0,1254
total < 10 sol/hora			28,985543
total 20 sol/hora			47,065543

Tabla 9.14: Fiabilidad total. Para un número menor a 10 solicitudes por hora y para un número mayor a 10 solicitudes por hora

9.1.2 Modulo RCM4400

En esta sección se analiza la confiabilidad del módulo Rabbit RCM4400. Como se mencionó anteriormente, el módulo RCM4400 es un módulo embebido que contiene el microprocesador Rabbit4000 y los elementos necesarios para su funcionamiento, tales como la memoria RAM y la memoria Flash y los elementos necesarios para el funcionamiento Wi-Fi, como son el FPGA y el módulo transmisor. Dado que el fabricante no especifica la tasa de fallas del módulo, pero si da un diagrama esquemático del mismo, es que se calcula la tasa de fallas utilizan el military handbook HDBK-217.

9.1.2.1 Resistores

El cálculo de los resistors es igual al desarrollado en la tabla 9.1, obteniéndose una tasa de fallas de $\lambda = 0,051 / 10^6$ horas

9.1.2.2 Capacitores

Al igual que en el caso de los resistores, el cálculo de los capacitores es el mismo que el realizado en la tabla 9.2 y 9.3 para cerámicos y electrolíticos respectivamente. Se utilizaron capacidades de 10 nF, 100 nF, 1μF, 10 μF con los siguientes valores como resultado:

$$\lambda_{10\text{nF}} = 0,224 / 10^6 \text{ horas}$$

$$\lambda_{100\text{nF}} = 0,29 / 10^6 \text{ horas} \quad \text{recalcular}$$

$$\lambda_{1\mu\text{F}} = 0,00952 / 10^6 \text{ horas}$$

$$\lambda_{10\mu\text{F}} = 0,014 / 10^6 \text{ horas}$$

9.1.2.3 Circuitos Integrados

9.1.2.3.1 Microprocesador Rabbit4000

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Parámetro	Símbolo	Valor	Observaciones
Indice de complejidad	C_1	0,08	Dispositivo CMOS de mas de 3000 trans
Falla de empaquetamiento	C_2	0,0043	Dispositivo SSOC 128 pins
Factor de temperatura	π_T	0,61	
Factor de aprendizaje	π_L	5,2	4 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,29848		

Tabla 9.15: Fiabilidad del microprocesador R4000

9.1.2.3.2 FPGA XLinux xc35250e

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Parámetro	Símbolo	Valor	Observaciones
Indice de complejidad	C_1	0,08	Dispositivo CMOS de mas de 3000 trans
Falla de empaquetamiento	C_2	0,0047	Dispositivo SSOC 144 pins
Factor de temperature	π_T	0,61	
Factor de aprendizaje	π_L	5,2	4 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,30264		

Tabla 9.16: Fiabilidad del FPGA XLinux xc35250e

9.1.2.3.3 Transmisor uw2453

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/ 10^6 horas		
Parámetro	Símbolo	Valor	Observaciones
Indice de complejidad	C_1	0,02	Dispositivo CMOS de mas de 101 trans
Falla de empaquetamiento	C_2	0,003	Dispositivo SSOC 48 pins
Factor de temperatura	π_T	0,61	
Factor de aprendizaje	π_L	5,2	4 años de antigüedad
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,09464		

Tabla 9.17: Fiabilidad del uw2453s

9.1.2.3.4 Memoria Flash

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Índice de complejidad	C ₁	0,042	Memoria de mas de 256k Dispositivo SSOC 32 pins
Falla de empaquetamiento	C ₂	0,0027	
Factor de temperatura	π_T	0,61	4 años de antigüedad
Factor de aprendizaje	π_L	5,2	
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,161304		

Tabla 9.18: Fiabilidad de la memoria Flash

9.1.2.3.5 Memoria RAM

Fórmula de cálculo	$\lambda_p = (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_L \cdot \pi_Q$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Índice de complejidad	C ₁	0,042	Memoria de mas de 256k Dispositivo SSOC 32 pins
Falla de empaquetamiento	C ₂	0,0027	
Factor de temperatura	π_T	0,61	4 años de antigüedad
Factor de aprendizaje	π_L	5,2	
Factor Ambiente	π_E	2	
Factor de Calidad	π_Q	1	
Tasa de Fallas	0,161304		

Tabla 9.19: Fiabilidad de la memoria RAM

9.1.2.3 Capacitores

Fórmula de cálculo	$\lambda_p = \lambda_b \cdot \pi_C \cdot \pi_E$ fallas/10 ⁶ horas		
Parámetro	Símbolo	Valor	Observaciones
Tasa base de fallas a 25 MHZ	λ_b	0,027	
Factor Capacitivo	π_C	2,1	
Factor de Calidad	π_E	3	
Tasa de Fallas	0,1701		

Tabla 9.20: Fiabilidad del cristal

9.1.2.4 Conectores

Al igual que con la placa base, se utilizó un conector de 50 pines con una tasa de fallas de $\lambda = 0,1254 / 10^6$ horas

9.1.2.5 Tasa de fallas total del módulo RCM4400

Componente	Tasa de Fallas por componente (fallas/10 ⁶ horas)		
	Cantidad		total
Resistor	20	0,0017	0,034
Capacitor cerámico 10nF	12	0,224	2,688
Capacitor cerámico 100nF	5	0,29	1,45
Capacitor electrolítico 1µF	10	0,0095	0,095
Capacitor electrolítico 10µF	4	0,014	0,056
R4000	1	0,29848	0,29848
xc35250e	1	0,000475	0,30254
uw2453	1	9,46E-02	0,09464
RAM	1	1,61E-01	0,161304
Flash	1	1,61E-01	1,61E-01
29.49 MHz crystal	1	0,00625	0,00625
Conector 50 pin	1	0,1254	0,1254
total			5,472918

Tabla 9.21: Fiabilidad total

9.1.2.6 Fiabilidad total y tiempo medio entre fallas (TMEF)

Obtenidos las confiabilidades para los dos módulos de hardware basados en los componentes de cada uno de ellos, podemos obtener la confiabilidad total como la suma de las confiabilidades. Dado que se separó en dos debido a variación de la confiabilidad en función de la cantidad de solicitaciones de los Relays, lo mismo se realizará con la confiabilidad total resultando:

$$\lambda_{o, 20 \text{ solicitaciones/h}} = 52,53 / 10^6 \text{ horas}$$

$$\lambda_{o, < 10 \text{ solicitaciones/h}} = 34,45 / 10^6 \text{ horas}$$

El tiempo medio entre fallas se calcula como la inversa de este valor y se calcula en años de la siguiente forma:

$$TMEF_{20 \text{ solicitaciones/h}} = \frac{1}{\lambda_o} = \frac{1}{52,53 \text{ fallas} / 10^6 \text{ horas}} \approx 19033 \text{ horas} \approx 2,17 \text{ años}$$

$$TMEF_{\leq 10 \text{ solicitaciones/h}} = \frac{1}{\lambda_o} = \frac{1}{34,45 \text{ fallas} / 10^6 \text{ horas}} \approx 29020 \text{ horas} \approx 3,31 \text{ años}$$

9.2 FMEA (análisis de efectos y modo de fallas)

9.2.1 Análisis de Fiabilidad de hardware

En esta sección se presenta un estudio de confiabilidad de los componentes de hardware involucrados en el sistema. Para el estudio de confiabilidad se clasifican las fallas según su severidad. Se consideraron las siguientes severidades:

- Baja: El sistema continua funcionando pero algunas de sus funciones estarán limitadas, sin alterar el funcionamiento global del equipo
- Media: El equipo continua funcionando con algunas de sus funciones principales limitadas o funcionando en forma intermitente.
- Alta: El equipo pierde la finalidad para la cual fue diseñado.

Dado que el equipo es diseñado para el control automático, una falla puede afectar el control de un dispositivo sin afectar el funcionamiento de otro. Por tal motivo resulta de gran interés realizar un análisis de cada una de las fallas. Para tal motivo se realiza una tabla indicando el resultado de la falla de cada uno de los componentes involucrados en el sistema.

Módulo	Designación	Descripción	Tipo	Efecto	Severidad
Fuente	C1	Capacitor filtro de entrada	apertura	degradación de la función de la fuente	baja
			corto	la fuente deja de funcionar	alta
	C2	Capacitor filtrado de fuente de switching	apertura	alimentación con ruido	baja
			corto	la fuente deja de funcionar	Alta
	C3	Capacitor filtrado de ruido	apertura	degradación de la función de la fuente	Baja
			corto	la fuente deja de funcionar	Alta
	C4	Capacitor filtrado de ruido	apertura	degradación de la función de la fuente	Baja
			corto	la fuente deja de funcionar	Alta
	L1	Inductor de filtrado en fuente de switching	apertura	degradación de la función de la fuente	Baja
			corto	degradación de la función de la fuente	Baja
	D1	Diodo de protección reversión de polaridad	apertura	la fuente deja de funcionar	Alta
			corto	la fuente queda desprotegida	Baja
	D2	regula la tensión de salida	apertura	pierde la referencia de tensión	Alta
			corto	pierde la referencia de tensión	Alta
	LM2575	fuentes de switching 5v	falla	Pérdida de	Alta

				alimentacion 5 v	
	LM1117	regulador lineal 3,3v	falla	Perdida de alimentacion 3,3v	Alta
Sensor de Temperatura	R1	Preset para regulación del ADC	apertura	perdida de referencia de temperatura	Baja
			corto	cortocircuito en el equipo	Alta
	R2	Resistencia de bloque amortiguador de entrada	apertura	alta variacion en medicion temp	Baja
			corto	alta variacion en medicion temp	Baja
	R3	Resistencia de bloque amortiguador de entrada	apertura	alta variacion en medicion temp	Baja
			corto	alta variacion en medicion temp	Baja
	C5	Capacitor de bloque amortiguador de entrada	apertura	fallo en la medicion de temperatura	Baja
			corto	alta variacion en medicion temp	Baja
	LM35	sensor de temperatura	falla	fallo en la medicion de temperatura	Baja
Convertor de Nivel	TLC548	conversor analógico digital	falla	fallo en la medicion de temperatura	Baja
	Max3222	adaptador de niveles RS-232	falla	Incomunicacion con controlador X.10	Alta
	C6	capacitor filtro alimentacion	corto	ruido en la comunicacion	Media
			apertura	ruido en la comunicacion	media
	C7	capacitor filtro alimentacion	corto	ruido en la comunicacion	Media
			apertura	ruido en la comunicacion	Media
	C8	Filtro canal 2 (programación serie inicial)	corto	falla en canal 2	Baja
			apertura	ruido en el canal 2	Baja
	C9	Filtro canal 1 (comunicación X.10)	corto	falla en canal 1	Alta
Salidas a relay			apertura	ruido en el canal 1	Media
	74LS244	Driver para salida a relay	falla	falla disp directamente controlados	Alta
	D3	diodo protección salida 3	corto	falla disp 3	Media
			apertura	falla proteccion salida disp 3	Baja
	D4	diodo protección salida 2	corto	falla disp 2	Media
			apertura	falla proteccion salida disp 2	Baja

	D5	diodo protección salida 1	corto	falla disp 1	Media
			apertura	falla proteccion salida disp 1	Baja
	D6	diodo protección salida 0	corto	falla disp 0	Media
			apertura	falla proteccion salida disp 0	Baja
	RL3	relay de salida	falla	falla canal de salida 3	Baja
	RL2	relay de salida	falla	falla canal de salida 2	Baja
	RL1	relay de salida	falla	falla canal de salida 1	Baja
	RL0	relay de salida	falla	falla canal de salida 0	Baja
RCM4400W		Módulo microcontrolador	falla	falla del nucleo del sistema	Alta

Tabla 9.22: FMEA

9.3 Estudios de confiabilidad de Software

Se utilizará el modelo de Jelinski-Moranda para estimar la confiabilidad del software tanto del microcontrolador como el de la PC.

El modelo supone que las fallas tienen una distribución exponencial. Siendo i el número de falla, entonces las fallas se distribuyen a lo largo del tiempo de desarrollo/debuggeo según la siguiente fórmula,

$$F_i(t_i) = 1 - e^{-\lambda_i t_i}$$

Donde

$$\lambda_i = (N - i + 1)\Phi$$

Donde N es el número inicial de fallas, Φ es la contribución de cada falla a la falla de todo el sistema. El modelo asume que todas las fallas tienen la misma tasa. El modelo estima mediante el método de máxima verosimilitud los parámetros N y Φ .

Para la estimación se utilizó el software llamado SMERFS, el cual se puede descargar en forma gratuita de: <http://www.slingcode.com/smerfs/downloads/>. Al software se le deben ingresar las fallas encontradas en el software y el tiempo en el que fueron detectadas y solucionadas.

9.3.1: Confiabilidad de Firmware de Controlador

En la tabla 9.23 se listan las fallas de firmware del microcontrolador encontradas durante el debug del programa expresadas en semanas.

semana	Error
1	error de conectividad driver loctl para Wi-fi
1	falla en autenticación wep Wi-Fi
1	error de conectividad a través de gateway
1	error de escritura en memoria flash
1	Error de inicialización de variables dinámicas
2	falla recepción RS-232 controlador X.10
2	falla liberación buffer RS-232
2	falla en calculo de checksum X.10
2	falla en la lectura de RS-232
2	falla en ejecución de interrupcion por timeout de X.10
2	Error en la interrupción
3	falla de interrupción SPI
3	falla de lectura SPI
3	error en la apertura de socket TCP
3	falla en la escritura de socket TCP
3	falla en timeout de respuesta X.10 a Microcontrolador
4	Falla intermitente en la apertura del socket hacia el web server
4	falla en la respuesta por socket al Web Server
4	falla en timeout del socket
4	falla en el seteo de dirección IP por consola
4	falla en la consola por RS-232
4	falla de carga en memoria variables en backup

Tabla 9.23: Listado de errores que encontraron en el firmware

Ingresando los datos en el mencionado software, se estiman los siguientes parámetros

Parámetro	Valor	Unidad
constante de proporcionalidad (Φ)	0,028	
Tiempo medio hasta la próxima falla	6,311	semanas
Cantidad Total de Fallas	27,6	
Faltas restantes	5,6	

Tabla 9.24: Estimaciones del modelo de Jelinsky-Moranda para firmware

Con estos datos, el software indica también la función de aparición de las fallas que se representa en forma gráfica mediante la figura 9.2.

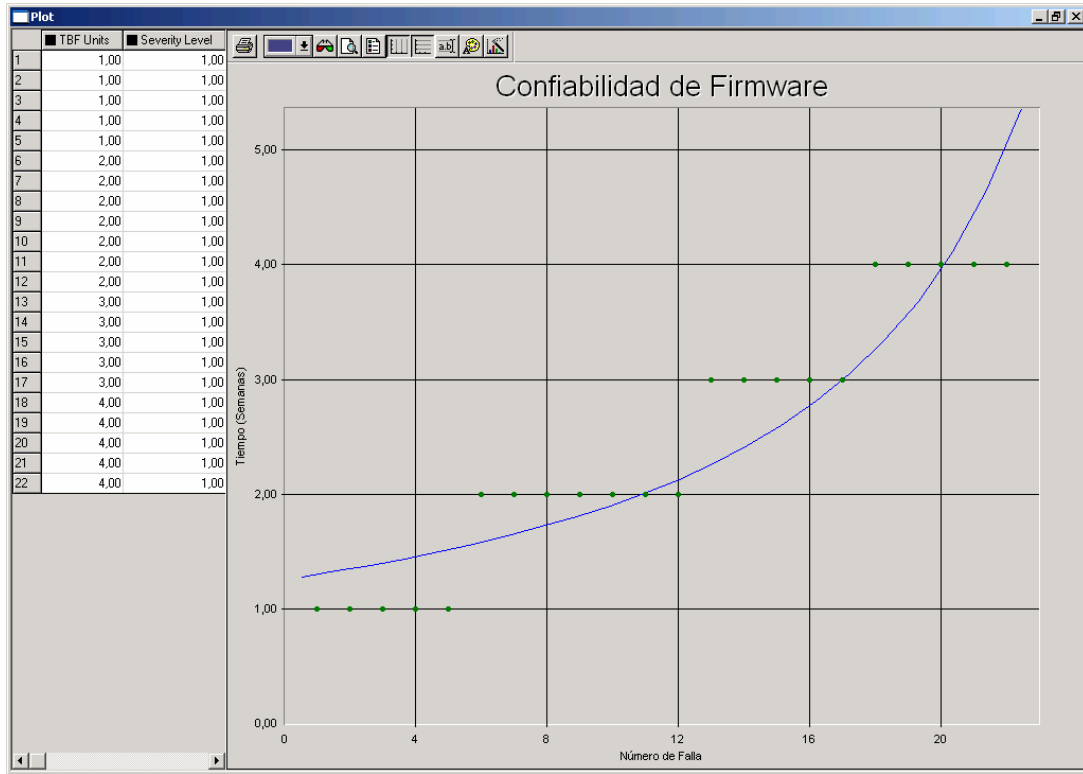


Figura 9.2: Fiabilidad del firmware

9.3.2: Confiabilidad de Software Web Server

En forma análoga a lo desarrollado para el firmware del controlador, se realiza el análisis de confiabilidad para el software del Web-Server. A continuación se muestra la tabla correspondiente.

semana	error/falla
1	error instalacion modulos perl
1	error de compatibilidad de versiones entre modulos y core de perl
1	error de autenticacion de usuarios
1	error de escritura base de datos de configuracion
2	error configuracion virtual host web server
2	falla reutilizacion de port tcp en socket
2	falla en calculo direcciones X10
2	error http method(post cgi con formato no valido)
	falla discriminacion de usuarios para diferentes
3	opciones de configuracion
3	error almacenamiento de fecha en evento programado
3	falla en el cierre de socket TCP
3	error almacenamiento de hashes de configuracion
4	error dentro de menus desplegados de fechas
4	falla en timeout del socket
5	error configuracion de nuevo controlador
	falla envio de comandos por etiqueta cuando la
5	direccion ip fue configurada en un nuevo controlador
	falla seteo de evento programado(opcion "todos los
5	dias")
5	falla limite cantidad de caracteres de una etiqueta

Tabla 9.25: Estimaciones del modelo de Jelinsky-Moranda para Software

Los valores estimados son:

Parámetro	Valor	Unidad
constante de proporcionalidad (Φ)	0,034	
Tiempo medio hasta la próxima falla	9,389	semanas
Cantidad Total de Fallas	21,09	
Faltas restantes	3,093	

Tabla 9.25: Estimaciones del modelo de Jelinsky-Moranda para Software

El siguiente gráfico muestra la función de distribución de aparición de fallas en el tiempo (semanas).

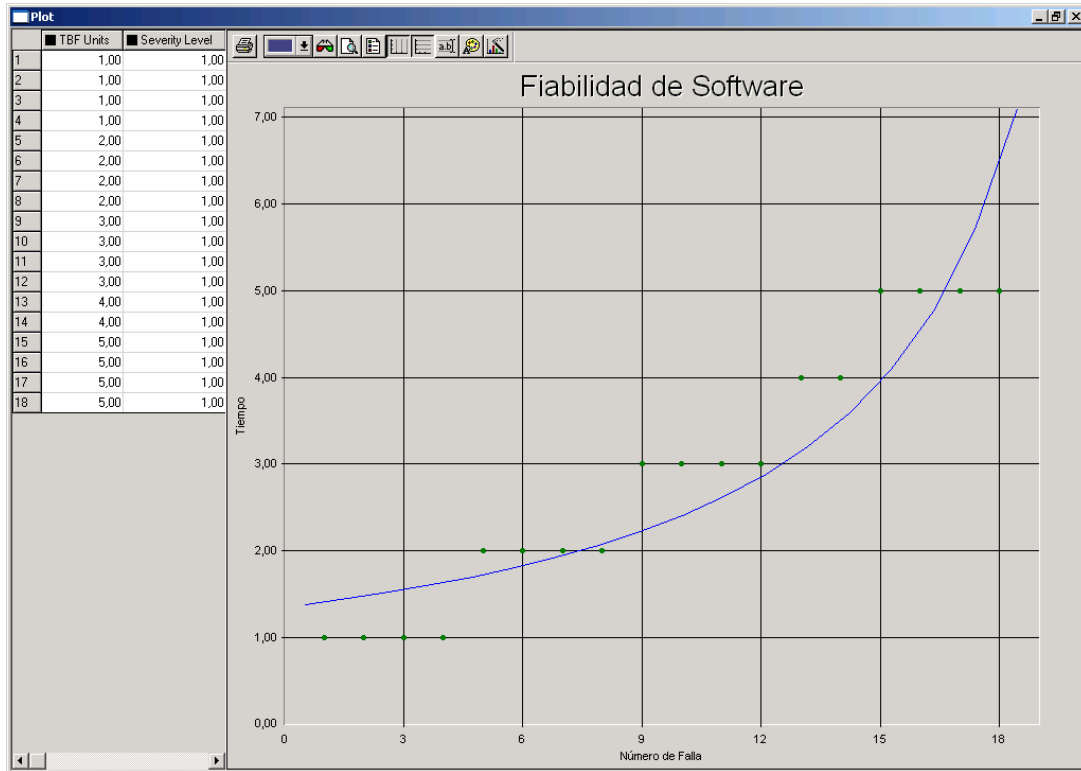


Figura 9.3: Fiabilidad del Software

9.4 Resolución de Problemas

Todo equipo debe contener un instructivo para realizar el diagnóstico y la posible resolución de los problemas que puede presentar el equipo. En esta configuración se pueden presentar problemas en las siguientes interfaces:

- Interfaz Web usuario servidor
- Comunicación socket TCP/IP servidor microcontrolador
- Comunicación RS-232 microcontrolador-controlador X.10

Para estos efectos se realiza la siguiente tabla que resume los posibles errores que pueden aparecer en cada una de las interfaces y la forma de diagnosticarlos.

Problema	Causa	Solución
No se puede ingresar a la página web desde el explorador de internet	El usuario no está conectado a la red	Verificar que el usuario se encuentra conectado a la red. Ejecutar Ping desde el usuario al router
	El servidor se encuentra apagado o desconectado de la red	verificar que el servidor esté encendido y conectado. Ping desde el servidor al router

Los comandos responden con error 404	El comando ejecutado no es válido	Verificar definición del comando/dirección en dispositivos controlados
Los comandos responden con error 500	Timeout comunicación con controlador X.10	Verificar que el controlador X.10 esté conectado a al módulo controlador
La página web no muestra resultado de la operación	Timeout server-controlador	Verificar la conexión a la red de ambos equipos. Reiniciar el que corresponda en caso que no haya respuesta

Tabla 9.27: Resolución de problemas

9.5 Ensayos de aceptación

9.5.1 Calidad del sistema completo

La calidad del sistema completo, compuesto por el controlador junto con la interfaz de configuración y control, estará determinada por la medida en que el producto se mantenga dentro de las especificaciones de fabricación y satisfaga los requerimientos del usuario durante su tiempo de vida útil. La etapa de diseño, desarrollo y manufactura resultan en apartamientos de la especificación original que luego son acentuados por las etapas de instalación, uso y mantenimiento. Existen diferentes tipos de falla que ocasionan un apartamiento respecto de las especificaciones originales. Según su severidad las fallas pueden clasificarse en:

- Severas o críticas: no permiten el normal uso del equipo.
- Perceptibles: permiten el uso del equipo con funcionalidad restringida.
- Leves: permiten el uso del equipo pero con degradación de requerimientos.

Según su tipo las fallas pueden clasificarse en:

- Mecánicas: están dadas por conectores mal instalados o contruidos u otro defecto de fabricación. Son de fácil detección.
Ej: Falla en el encastre del conector serial que vincula el controlador CM11a con el resto del circuito.
- Electrónicas y/o de software: fallas que resultan en la degradación de una señal, nivel de tensión y/o corriente provocando o inhibiendo algún estado particular. Si esta degradación permite el funcionamiento del equipo resultan de ser de difícil diagnostico y reparación.
Ej: Presencia de excesivo ruido de línea sobre la instalación eléctrica del domicilio de instalación lo que puede causar fallas intermitentes sobre todo el sistema.

9.5.2 Verificación de la calidad

El proceso de control de calidad exige de la prueba de un número de equipos dado por lote para determinar la cantidad de equipos defectuosos en la población de producidos. Se definen dos variables en este proceso,

- AQL (*Acceptance Quality Level*): es el nivel de equipos defectuosos aceptable en un cierto lote número que se indica en porcentaje (%).
- LTPD (*Lot Tolerante Percentage Level*): nivel de equipos defectuosos por lote que resulta inaceptable. También se indica en porcentaje (%).

El procedimiento a seguir es el siguiente: se debe medir la cantidad de equipos defectuosos en un lote y si el nivel está por debajo del AQL entonces el lote se acepta, si en cambio este resulta por encima del LTPD entonces deberá ser descartada la totalidad del lote. Si el nivel de defectuosos resultara en un valor intermedio entre AQL y LTPD entonces se deberán descartar los equipos ya probados y efectuar una nueva medición.

9.6 Políticas de Mantenimiento

9.6.1 Mantenibilidad

El sistema se compone de distintos componentes interconectados entre sí. Tal como se detalla en 9.4, el sistema cuenta con herramientas de diagnóstico que permiten determinar con precisión el componente que produce la falla. Esto permite cambiar los componentes que pueden originar la falla con facilidad. Además, considerando la implementación electrónica definida en 8, en donde todos los integrados los integrados son montados sobre zócalos y los distintos componentes se interconectan utilizando conectores estandar, el reemplazo de un componente puede realizarse en forma muy rápida en sitio.

Por tal motivo se debe contar con los repuestos para cada uno de los componentes (resistencias, capacitares, integrados, relays, etc) para que el técnico de campo disponga en caso de ser necesario.

Además, también se deberá tener unidades completas de respaldo para reemplazar una unidad completa en caso de que el técnico de sitio no pueda reparar la unidad en el tiempo establecido para la prestación. En ese caso se utilizará una unidad del lote denominado “reparados” y la unidad reemplazada se llevará a laboratorio para resolver el problema e incorporarla al lote de reparados para ser utilizada en caso de falla de otro equipo.

Los módulos que se deben tener de repuesto en el lote de reparados son:

- Placa Base
- RCM4400
- Controlador X.10 CM11
- Dispositivos x.10 controlados

El reemplazo de los componentes permite:

- La operación de mantenimiento sea realizada en un mínimo tiempo
- El mantenimiento puede ser realizado por personal con mínimo entrenamiento

9.6.2 Serviciabilidad

La serviciabilidad es una medida que indica la capacidad de un equipo para ser reparado. Las consideraciones de diseño basadas en la serviceabilidad tenidas en cuenta son:

- Diseño modularizado
- Conectores flexibles y estandar
- Fácil acceso para el reemplazo de partes

9.6.3 Contratos de mantenimiento

Concluido el periodo de garantía el cliente tiene la opción de contratar un servicio de soporte basado en distintas variantes. En todos los casos el cliente deberá proveer un acceso remoto al servidor basado en el protocolo SSH (secure shell) del snack TCP/IP vía Internet. Para ello el cliente deberá redireccionar los puertos correspondientes en el router y contar con una dirección IP fija o registrarse en un servidor dinámico de nombres (DynDNS). Se definen las siguientes clases de contrato

7x24: Se garantiza que luego del llamado telefónico realizado por el usuario, personal capacitado se comunicará dentro de las 2hs para realizar el soporte correspondiente. El usuario puede llamar las 24 horas del día. En una primera etapa se realiza un diagnóstico remoto del equipo para determinar la causa del problema y en caso de ser necesario un técnico concurre al sitio para realizar una reparación del hardware o reemplazar una parte. El contrato de mantenimiento no incluye la reprogramación del equipo ni la adición de parte, las cuales deben ser solicitadas como un nuevo producto o servicio. Este tipo de contrato incluye el reemplazo sin costo adicional de los componentes. En caso de presentarse la falla fuera de los

horarios laborales de semana (L a V de 9:00 a 18:00), se cuenta con un sistema de radiollamada de guardia. La guardia realizará únicamente tareas de mantenimiento en caso de fallas críticas y no trabajará en fallas menores, donde se deberá esperar al próximo día hábil.

5x8: El servicio es de similares características al de 7x24 con la diferencia de que no se cuenta con el servicio de guardia.

Sin contrato: En este caso no se garantiza un tiempo de respuesta al usuario. Para realizarse el servicio el cliente deberá aceptar una cotización de servicios y acceder a un pago basado en la cantidad de horas que demore la resolución del problema.

En todos los casos el soporte de partes se ve limitado por la disponibilidad de repuestos en el distrito donde el sistema se encuentra instalado. A tales efectos el contrato será analizado particularmente en dichos casos.

10. Conclusiones

En la presente sección se analizan los resultados obtenidos de las evaluaciones del primer prototipo y se proponen mejoras para futuros diseños o siguientes versiones.

10.1. Excelencias

El sistema domótico alcanzó todos los objetivos planteados en el anteproyecto y esperables del primer prototipo. Se demostró además que es posible automatizar los dispositivos que actualmente se operan en forma manual realizándolo con el menor impacto posible sobre una instalación existente, sin necesitar un rediseño por cableado adicional, brindando al usuario la posibilidad de decidir que dispositivo quiere controlar mediante cableado y cuales mediante X.10 de forma tal de no afectar la estética del hogar o residencia.

Se comprobó que la comunicación Wi-Fi logró ser confiable haciendo que no se presenten microcortes entre el módulo controlador y el servidor.

El prototipo fue probado utilizando un dispositivo X.10 de casquillo, funcionando sin ningún tipo de inconvenientes y con luces como dispositivos directamente conectados. Se comprobó además que el circuito es seguro y confiable.

10.2. Futuros Diseños

Para nuevos diseños y segundas versiones del producto se ponen en consideración las siguientes mejoras para hacer el producto aún mas competitivo.

10.2.1 Interfaz Web

La interfaz web utilizada provee todas las utilidades necesarias para agregar dispositivos e ir personalizando el equipamiento a controlar, realizar las tareas programadas, etc. Sin embargo, el entorno gráfico no es del todo intuitivo y el usuario necesita conocer el nombre con el que se ha definido el dispositivo a controlar para operarlo. Con pocos dispositivos, el problema es menor, pero con una gran cantidad de dispositivos controlados se pueden cometer errores y activar dispositivos por equivocación.

Se propone entonces como mejora el desarrollo de una interfaz web con entorno gráfico de forma tal que el usuario pueda identificar rápidamente el dispositivo sobre el plano de la vivienda y clickear en el para operarlo. Dado que cada hogar tiene un plano distinto, se deberá también diseñar un programador del entorno Web de forma que el usuario pueda importar el plano de su casa y agregar los dispositivos que se hayan instalado en forma dinámica, es decir, que si el usuario agrega o elimina dispositivos el plano se pueda actualizar.

Esta solución demandará grandes esfuerzos y tiempos de desarrollo a realizar por profesionales Analistas en Sistemas o Ingenieros Informáticos o en Sistemas.

10.2.2 Funcionamiento asincrónico de tareas programadas

Las tareas programadas se agendan en el crontab de Linux y son ejecutadas desde el servidor al módulo controlador en el momento en que debe realizarse la tarea. Si bien se comprobó que las redes Wi-Fi tienen alta disponibilidad y que la probabilidad de falla del servidor es baja dada a la estabilidad del Linux, siempre existirá el mínimo riesgo de que al momento de ejecutar una tarea, la misma no sea ejecutada por problemas de red o del servidor.

Se propone entonces que las tareas programadas se agenden en el propio módulo controlador y que el Servidor simplemente consulte al módulo 1 minuto después si ejecutó la tarea, para tomar reconocer el suceso o disparar una alarma en caso contrario. De esta forma el módulo controlador ejecutará la tarea en forma autónoma sin necesidad de conectarse al módulo controlador para realizarla solucionando el eventual problema mencionado.

Para lograr esto se deben realizar los siguientes cambios:

- Habilitar en el microcontrolador el reloj de tiempo real (RTC) y sincronizarlo periódicamente con un servidor NTP habilitado en el servidor.
- Guardar las tareas programadas en la memoria flash del equipo y generar el código que cause una interrupción cuando se deba ejecutar la tarea. Para ello se deberá crear un comando que escriba en la flash. Generar comando que verifique si se ejecutó la tarea en forma satisfactoria.

- Agregar una pila al módulo controlador para que mantenga activo al RTC
- Cambiar el código en el servidor de forma que cuando se active una tarea programada el Server envíe el comando al microcontrolador para escribir en la flash la tarea programada y agendar en el crontab que chequee la ejecución de la tarea 1 minuto después y reejecutarla en caso fallido.
- Configurar en el servidor el servicio de NTP

La realización de estas tareas hacen que el sistema sea menos dependiente del servidor, pero requieren un gran esfuerzo de programación y desarrollo (especialmente las primeras 3 modificaciones). Se deberá evaluar la conveniencia de realizarlo con el lanzamiento de la segunda generación.

Apéndice A: Código fuente

A.1 Firmware

A.1.1 Main.c

```
#class auto //asigna en forma automática variables dinámicas y estáticas
#define TCPCONFIG 1 //configuración TCP en 1: No DHCP
#define SPI_SER_A //Define el puerto A para SPI
//definiciones para la console de administracion
#define DINBUFSIZE 255 //255 Bytes de buffer de entrada serie la
console
#define DOUTBUFSIZE 255 //255 Bytes de buffer de salida zconsole
#define MAX_TCP_SOCKET_BUFFERS 3
#define NUM_CONSOLES 2 //2 consolas, serial y TCP/IP
#define CON_HELP_VERSION
#define CON_VERSION_MESSAGE "TCP/IP User Block Console Version 1.1\r\n"
#define CON_INIT_MESSAGE CON_VERSION_MESSAGE
#define CON_BACKUP_USER_BLOCK

//importa los archivos para las ayuda de la consola
#import "D:\Trabajo Profesional\helpconsole\help.txt" help_txt
#import "D:\Trabajo Profesional\helpconsole\help_help.txt" help_help_txt
#import "D:\Trabajo Profesional\helpconsole\help_echo.txt" help_echo_txt
#import "D:\Trabajo Profesional\helpconsole\help_set.txt" help_set_txt
#import "D:\Trabajo Profesional\helpconsole\help_set_param.txt"
help_set_param_txt
#import "D:\Trabajo Profesional\helpconsole\help_show.txt" help_show_txt
#import "D:\Trabajo Profesional\helpconsole\help_set_mail.txt" help_set_mail_txt
#import "D:\Trabajo Profesional\helpconsole\help_set_mail_server.txt"
help_set_mail_server_txt
#import "D:\Trabajo Profesional\helpconsole\help_set_mail_from.txt"
help_set_mail_from_txt
#import "D:\Trabajo Profesional\helpconsole\help_mail.txt" help_mail_txt
#import "D:\Trabajo Profesional\helpconsole\help_add_nameserver.txt"
help_add_nameserver_txt

#define SPI_CLK_DIVISOR 100
#memmap xmem //utiliza memoria extendida

//librerias utilizadas
#use RCM44xxW.LIB //libreria correspondiente al modulo
#use "custnet-final.h" //configuracion inicial de red
#use "dcrtcp.lib" //funciones de red
#use "X10-CM11.h" //funciones para controlador X.10 CM11
#use "socket-final.h" //implementacion del socket TCP/IP
```



```
#use "smtp.lib" //protocolo de mail utilizado por la consola
#use "zconsole-final.lib" //funciones de consola de administracion

//definicion de parámetros y funciones personalizados de la consola

int hello_world(ConsoleState* state);
int my_show(ConsoleState* state);

const ConsoleIO console_io[] =
{
    CONSOLE_IO_SERD(57600),
    CONSOLE_IO_TELNET(23)
};

const ConsoleCommand console_commands[] =
{
    { "HELLO WORLD", hello_world, 0 },
    { "ECHO", con_echo, help_echo_txt },
    { "HELP", con_help, help_help_txt },
    { "", NULL, help_txt },
    { "SET", NULL, help_set_txt },
    { "SET PARAM", con_set_param, help_set_param_txt },
    { "SET IP", con_set_ip, help_set_txt },
    { "SET NETMASK", con_set_netmask, help_set_txt },
    { "SET GATEWAY", con_set_gateway, help_set_txt },
    { "SET NAMESERVER", con_set_nameserver, help_set_txt },
    { "ADD NAMESERVER", con_add_nameserver, help_add_nameserver_txt },
    { "SHOW", con_show_multi, help_show_txt },
    { "SET MAIL", NULL, help_set_mail_txt },
    { "SET MAIL SERVER", con_set_mail_server, help_set_mail_server_txt },
    { "SET MAIL FROM", con_set_mail_from, help_set_mail_from_txt },
    { "MAIL", con_mail, help_mail_txt },
    { "SET PINGCONFIG", con_set_icmp_config, help_set_txt },
    { "SET PINGCONFIG RESET", con_set_icmp_config_reset, help_set_txt },
    { "SET DHCP", con_set_dhcp, help_set_txt },
    { "SET DEBUG", con_set_tcpip_debug, 0 }
};

const ConsoleError console_errors[] = {
    CON_STANDARD_ERRORS
};

const ConsoleBackup console_backup[] =
{
    CONSOLE_BASIC_BACKUP,
    CONSOLE_TCP_MULTI_BACKUP,
    CONSOLE_SMTP_BACKUP
};
```

```

int hello_world(ConsoleState* state)
{
    state->conio->puts("Hello, World!\r\n");
    return 1;
}

//fin de parámetros personalizados de consola

#use "spi.lib" //utiliza libreria SPI
#define MAX_BUFLLEN    11 //longitud del buffer TCP
#define CINBUFSIZE 3    //longitud buffer serie para X.10-CM11
#define COUTBUFSIZE 3    //longitud buffer serie para X.10-CM11

//definicion de estructura del socket de recepcion
typedef struct {
    tcp_Socket sock;
    tcp_Socket *s;

    int lport;
    int state;

    char buf[MAX_BUFLLEN];
} sock_recv;
sock_recv tr_state;
sock_recv * const state = &tr_state;

void main()
{
    //definicion de variables utilizadas
    int sec;
    char temp[4];
    sec = 0;
    brdInit(); //inicializacion de la placa
    BitWrPortI(PBDR, &PBDRShadow, 1, 6);

    // inicializacion de la interfaz de red
    sock_init_or_exit(1);
    serCopen(4800); //abre el puerto y vacia los buffers
    serCwrFlush();
    serCrdFlush();
    //Abre el puerto 9000
    if(init_recv(9000)) { //En caso de error devuelve 1
        printf("Error en init_recv - no puede escuchar en el puerto\n");
        exit(0);
    }

    if (console_init() != 0) //en caso de fallar la consola serie
    {
        printf("No se pudo cargar la configuracion de consola\n");
    }
}

```

```

        con_backup(); //guarda en backup la informacion de la consola
    }

    adc(temp); //inicializa la medicion de temperatura
/*  loop principal. Se incluyen todos los coestados que participan en la ejecucion del
programa en
formato multitarea */
while(1) {
    costate {
        recv_tick(); //verifica el estado del socket
    }

    costate {
        console_tick(); //verifica el estado de la consola
        tcp_tick(NULL); //verifica el estado del stack TCP/IP
    }
}
}

```

A.1.2 Socket.h

```

/** Beginheader */
#define MAX_REPLY      10 //maximo 10 intentos de respuesta en el socket
#define IDLE_TMOUT 3000UL //3 segundos espera respuesta, sino Timeout

/* Los datos se reciben como un string. Luego la funcion decofunc llena la estructura
recv_data separando cada campo y
* lo coloca en el subcampo correspondiente */
typedef struct {
    int operation; //operacion a realizar
    char hc; //house code
    char dc; //devcode
    char value; //x10 function, temperature value, etc
} recv_data;

#define STATE_INIT 0 //escuchando
#define STATE_STEADY 1 //socket establecido
/** endheader */

/** Beginheader init_recv */
int init_recv(int port); //declaración de funcion de recepción
/** endheader */

int init_recv(int port) //definicion: Abre el puerto TCP
{
    int n;
    state->s = &state->sock; //verifica el estado del socket
}

```

```

state->lport = port; //puerto local. Se llama desde el main con el puerto 9000

tcp_listen(state->s, port, 0, 0, NULL, 0); //escucha en le puerto
state->state = STATE_INIT; //inidica que esta en estado inicial
sock_mode(state->s, TCP_MODE_ASCII); //indica que los datos serán pasados
en formato ascii
return 0; //devuelve sin error
}

//funcion ejecutada en el loop principal
/** Beginheader recv_tick */
void recv_tick(void);
/** endheader */

void recv_tick(void)
{
    auto int retval, length, n, p;
    char ready[MAX_BUFLLEN];
    strcpy(ready,"ready\n"); //inicializa el string
    tcp_tick(state->s); //verifica el contenido del socket
    length = 0;
    switch(state->state) {
    case STATE_INIT: //El socket no está establecido. Estado inicial cuando
bootea
        if(sock_established(state->s)) { //si el socket s está activo
            printf("Connection Established.\n"); //detectó conexión
            sock_fastwrite(state->s, ready, 6); //responde con ready
            state->state = STATE_STEADY; //cambia el estado a activo
        }
        break;

    case STATE_STEADY:
        if(!sock_established(state->s)) { //si el socket no está activo
            /* connection died; reset */
            printf("Connection lost.\n\n"); //perdió la conexión
            init_recv(state->lport); //vuelve a abrir el puerto
            break;
        }

        if(sock_bytesready(state->s) != -1) { //si hay bytes
            length = sock_gets(state->s, state->buf,
MAX_BUFLLEN);

            //lee lenght bytes del socket con datos en buf
            if(length > 0) {
                accion(state->buf); //ejecuta accion
pasando el contenido de buf
            }

            break;

```

```

        default:
            //nunca deberia llegar aca
            exit(-1); // reinicia el controlador
        }

    }
}

/** Beginheader accion */
void accion (char *data); //declaracion
/** endheader */

void accion (char *data) //definicion
/* funcion que recibe un string como argumento y ejecuta una accion. Lee
temperatura, ejecuta comando X10,
* activa dispositivo directamente conectado, etc. Los codigos de devolucion se
realizaron de forma analoga
* a los codigos de HTTP */
{
    char house, device, funcion;
    unsigned long t;
    int n, p, led; //return X10, socket fastwrite, and leds
    char not_found[MAX_REPLY], ok[MAX_REPLY],
    temperature[MAX_REPLY], timeout[MAX_REPLY]; //devoluciones al socket
    recv_data sock_data;
    strcpy(ok, "200\n"); //OK al socket
    strcpy(not_found, "404\n"); //comando no valido
    strcpy(timeout, "500\n"); //time out
    decofunc(data, &sock_data); //vuelca el contenido del socket a la estructura
data
    led=0;
    n=0;
    switch(sock_data.operation){
        case 700: //Dispositivos directamente conectados

            if (sock_data.dc == 2) //enciendo. Pongo un 0 porque hay
inversion de bit
            {
                BitWrPortI(PEDR, &PEDRShadow, 0, sock_data.hc);
//hc contiene el N° de bit
                led = 1;
            }
            if (sock_data.dc == 3) //apago. Pongo un 1 porque hay
inversion de bit
            {
                BitWrPortI(PEDR, &PEDRShadow, 1, sock_data.hc);
//escribe 1, apaga
                led = 1;
            }
        }
    }
}

```

```

        if (led ==1) p = sock_fastwrite(state->s, ok, 4); //si ejecuto
responde OK al server
        else p = sock_fastwrite(state->s, timeout, 4); //si no ejecuto
responde timeout al server
        break;

case 800: //Funciones X10

        while (n<1) { //ingresa a un loop infinito que verifica 2
coestados.
                costate {
                        t = MS_TIMER; //inicializa un timer
preveniend un timeout con el controlador
                        //X10
                        //ejecuta la funcion X
                        wfd n = x10(sock_data.hc, sock_data.dc,
sock_data.value);

                        //responde 1 si se ejecuta correctamente
                }

                costate {

                        if (MS_TIMER > t + IDLE_TMOUT)

//si se superan los 3 segundos

                        {
                                t = MS_TIMER;
                                n = 2; //pone la variable de salida
en 2

                                printf("Timed out!\n");
                                break;
                        }

                }

        }

        if (n ==1) p = sock_fastwrite(state->s, ok, 4); //se ejecuto
correctamente, escribo OK al socket
        if (n ==2) p = sock_fastwrite(state->s, timeout, 4); //se ejecutó
incorrectamente, escribo timeout
        serCwrFlush(); //limpio el puerto serie
        serCrdFlush();

        break; //800

case 850:
        //programo x10. Idem anterior pero lo hace 3 veces, lo necesario para
que los dispositivos se programen

```

```

        while (n<1) {
            costate {
                t = MS_TIMER;
                wfd n = x10(sock_data.hc, sock_data.dc,
sock_data.value);
            }
        }
        n=0;
        while (n<1) {
            costate {
                t = MS_TIMER;
                wfd n = x10(sock_data.hc, sock_data.dc,
sock_data.value);
            }
        }
        n=0;
        while (n<1) {
            costate {
                t = MS_TIMER;
                wfd n = x10(sock_data.hc, sock_data.dc,
sock_data.value);
            }
        }
        if (n==1) p = sock_fastwrite(state->s, ok, 4);
        break;
        case 900: //lee la temperatura
            adc(temperature); //funcion que lee la temp del sensor y la
escribe en el string temperature
            p = sock_fastwrite(state->s, temperature, 3); //escribo el valor
de temperatura en el socket
            break; //900

        default:
            p = sock_fastwrite(state->s, not_found, 4);
            break; //default
    }
}

/**/ Beginheader decofunc */
void decofunc(char *data, recv_data *sock_data); //declaracion
/**/ endheader */

void decofunc(char *data, recv_data *sock_data) //definicion
/* Recibe el string data proveniente del socket y llena la estructura de datos recv_data.
* Los campos de la estructura a llenar son:
* operation: operacion a realizar; 700 dispositivos directamente conctados, 800
dispositivos X10, etc
* hc: House Code. En que controlador está el dispositivo

```

```

* dc: Device Code, para identificar el dispositivo X10
* value: valor, para temperatura
* */
{
    int i;
    char temp_op[5], temp_hc[4], temp_dc[4], temp_value[4];
    printf("el socket contiene: %s\n", data);
    //leo el buffer del socket
    for (i=0;i<3;i++){ temp_op[i] = data[i]; temp_op[3] = '\0'; } //operation viene
en los primeros 3 caracteres
    for (i=3;i<5;i++){ temp_hc[i-3] = data[i]; temp_hc[2] = '\0'; } //hc viene en el 4
y 5
    for (i=5;i<7;i++) {temp_dc[i-5] = data[i]; temp_dc[2] = '\0'; } //dc viene en el 6
y 7
    for (i=7;i<9;i++) {temp_value[i-7] = data[i]; temp_value[2] = '\0'; } //value
viene en los ultimos 2
    //completo la estructura
    printf("temp_op: %s, temp_hc: %s, temp_dc: %s, value: %s\n", temp_op,
temp_hc, temp_dc, temp_value);
    //printf("temp_op: %s, temp_hc: %s\n",temp_op, temp_hc);
    sock_data->operation = atoi(temp_op); //transforma el string en un integer. Se
necesitan los valores en integer
    sock_data->hc = atoi(temp_hc);
    sock_data->dc = atoi(temp_dc);
    //printf("temp_hc en int: %d\n", sock_data->hc);
    sock_data->value = atoi(temp_value);
}

/**/ Beginheader adc */
char adc (char *temp); //declaracion
/**/ endheader */

char adc (char *temp) {
/* Esta funcion lee los datos del ADC. Provee la señal de clock al mismo y multiplica
por el coeficiente 0,392
* para obetener la temperatura en °C*/
char adc_reading;
int i, tempint;
float coef, tempfloat, tempread;
//mantiene el CS en bajo durante el tiempo de conversion
    for(i = 0;i < 10;i++)
    {
        BitWrPortI(PBDR, &PBDRShadow, 0, 7); // selecciona el ADC.
Tiene el CS en bajo
    }

    SPIRead(&adc_reading, 1); //Lee por primera vez para incializar
    adc_reading = 0;

```



```

        *temp = 0;
        SPIRead(&adc_reading, 1); //Lee 1 byte del puerto SPI y lo asigna a la
variable adc_reading
        BitWrPortI(PBDR, &PBDRShadow, 1, 7); // deshabilita el dispositivo
        coef = 0.3922;
        tempread = (float)adc_reading; //transformo la lectura en float para multiplicar
        tempfloat = tempread * coef;
        tempint = (int)tempfloat; //luego de multiplicado, vuelvo a convertir a entero
        printf("ADC es: %f\n", tempfloat);
        itoa(tempint, temp); //paso la variable de float a int
        temp[2]='\n';
    }

```

A.1.3 X10-CM11.h

```

/**/ Beginheader */

//define IDLE_TMOUT 5000UL
//Definicion de House Codes
#define XA 0x6
#define XB 0xE
#define XC 0x2
#define XD 0xA
#define XE 0x1
#define XF 0x9
#define XG 0x5
#define XH 0x8
#define XI 0x7
#define XJ 0xF
#define XK 0x3
#define XL 0xB
#define XM 0x0
#define XN 0x8
#define XO 0x4
#define XP 0xC

//Definición de Devices Codes

#define X1 0x6
#define X2 0xE
#define X3 0x2
#define X4 0xA
#define X5 0x1
#define X6 0x9
#define X7 0x5
#define X8 0x8
#define X9 0x7

```

```
#define X10 0xF
#define X11 0x3
#define X12 0xB
#define X13 0x0
#define X14 0x8
#define X15 0x4
#define X16 0xC
```

```
//Definición de Funciones
```

```
#define ON 0x2
#define OFF 0x3
```

```
//Definicion de headers
```

```
#define headeraddress 0x4
#define headerfunction 0x6
```

```
//estructura de mensaje de 2 bytes X10. Normalmente el primer byte es header y el
segundo es un direccion o comando
```

```
typedef struct {
    char fb; //primer byte
    char sb; //segundo byte
} twobstruct;
```

```
//estructura de mensaje de 1 byte. Se usa para aknowledge
```

```
typedef struct {
    char fb;
} onebstruct;
```

```
typedef struct {
    char cmd;      // 0x9b
    char seconds;
    char minutes;  // 0-119
    char hours;    // 0-11 (hours/2)
    char yearday;  // really 9 bits
    char daymask;  // really 7 bits
    char house;    // 0:timer purge, 1:monitor clear, 3:battery clear
} x10_setclock;
```

```
/** endheader */
```

```
/* START LIBRARY DESCRIPTION
```

```
*****
```

```
x10.h
```

```
Copyright (c) abertamoni at fi dot uba dot ar
2007/01/24 mmiodosky at fi dot uba dot ar
```

```
DESCRIPTION:
```

```
Funciones X.10 para controlador CM11
```

SUPPORT LIB'S:

END DESCRIPTION

```

*****/

```

```

/**/ Beginheader resolveaddr */

```

```

char resolveaddr (char housecode, char device);//declaracion

```

```

/**/ endheader */

```

```

char resolveaddr (char housecode, char device)

```

```

/* resuleve una direccion X10

```

```

* Dado dos string de 1 byte cada uno, coloca en un solo byte el housecode en el
primer nibble y el dicevicode

```

```

* en el segundo nibble. De esta forma se debe enviar el código al controlador X10*/

```

```

{
    return (housecode * 0x10 + device);
}

```

```

/**/ Beginheader resolvehousefunc */

```

```

char resolvehousefunc (char housecode, char func);

```

```

/**/ endheader */

```

```

char resolvehousefunc (char housecode, char func)

```

```

{
    return (housecode * 0x10 + func);
}

```

```

/**/ Beginheader setclock */

```

```

scofunc int setclock(void);

```

```

/**/ endheader */

```

```

scofunc int setclock(void)

```

```

/* Pone la interface en hora. Es requisito para que el CM11 funcione. Dado que no se
utiliza la hora del CM11, se pone

```

```

* siempre una hora fija*/

```

```

{

```

```

    x10_setclock cm11_setclock;
    int b;
    onebstruct clockack;
    cm11_setclock.cmd = 0x9b;
    cm11_setclock.seconds = 0x00;
    cm11_setclock.minutes = 0x00;
    cm11_setclock.hours = 0x00;
    cm11_setclock.yearday = 0x00;
    cm11_setclock.daymask = 0x80;
    cm11_setclock.house = 0x03;
    serCwrite(&cm11_setclock, sizeof(cm11_setclock));
    wfd b = cof_serCread(&clockack, sizeof(clockack), 3000UL);
    printf("cm11 responde luego de poner la hora %x\n",clockack.fb);
}

```

```

serCwrFlush();
serCrdFlush();

return 1;
}

/**/ Beginheader x10cm1lack */
scofunc int x10cm1lack(void);
/**/ endheader */
scofunc int x10cm1lack(void)
/*Escribe a knowladge de X10 */
{
int n;
onebstruct cm1lack, rback;
rback.fb = 0x00; //0x00 indica OK for transmision que envia el DTE
serCwrite(&rback, sizeof(rback)); //escribo el 0x00
wfd n = cof_serCread(&cm1lack, sizeof(cm1lack), 3000UL); //leo el 0x55

if (cm1lack.fb==0x55){ //0x55 indica OK desde el el controlador X10 (DCE)

return 1;
}
else return 0;
}
/**/ Beginheader x10cm1laddr */
scofunc int x10cm1laddr(char house, char device); //declaracion
/**/ endheader */
scofunc int x10cm1laddr (char house, char device)
/* Funcion que hace el intercambio necesario para enviar una direccion en el
controlador X10
* Para realizar una funcion, primero se debe enviar la direccion y luego la funcion
* Recibe un codigo de casa y un código de dispositivo
* Devuelve 1 si se ejecuta correctamente y 0 en caso de error*/
{
int nohora, deviceready, recibidos;
twobstruct address;
onebstruct cm1lack, rback;
char checksum, rbuff;
rback.fb = 0x00;
address.fb = headeraddress; //escribo header de direccion en el primer byte
address.sb = resolveaddr (house, device); //escribe en un solo byte HC y DC
checksum = address.fb + address.sb; //calcula el checksum a recibir del CM11
againaddr: //label al que vuelve en caso de no estar en hora el equipo
serCwrite(&address, sizeof(address)); //escribe la direccion en el puerto serieC
wfd recibidos = cof_serCread(&cm1lack, sizeof(cm1lack), 3000UL); //lee la
respuesta del controlador
rbuff = cm1lack.fb;
switch (rbuff){ //verifica las posibles respuestas

```

```

        case 0xa5: //la interface esta fuera de hora
            nohora=0;
            wfd nohora = setclock(); //pone la interface en hora
            break;

        default:
            if (cm1lack.fb == checksum) //el checksum de address es correcto
            {
                wfd deviceready = x10cm1lack(); //escribe 0x00 y espera
                if (deviceready) return 1; //la interface devolvió un 0x55 al
                else return 0; //la interface devolvió otra cosa o timeout
            }

            else return 0; //el checksum de address no es correcto
            break;
    }

    if (nohora) goto againaddr; //la interface no estaba en hora, luego de sincronizarla se
    vuelve a escribir la dirección
}

/** Beginheader x10cm1function */
scofunc int x10cm1function(char house, char func);
/* Funcion equivalente a x10CM11Addr */
/** endheader */
scofunc int x10cm1function(char house, char func)
{
    int recibidos, nohora, deviceready;
    twobstruct function;
    onebstruct cm1lack, rback;
    char checksum, rbuff;
    function.fb = headerfunction;
    function.sb = resolvehousefunc (house, func);
    checksum = function.fb + function.sb;
    printf("el calculo del checksum function es: %x\n",checksum);
    againfunc:
    serCwrite(&function, sizeof(function));
    wfd recibidos = cof_serCread(&cm1lack, sizeof(cm1lack), 3000UL);
    rbuff = cm1lack.fb;
    switch (rbuff){
        case 0xa5: //la interface esta fuera de hora
            printf("pongo la interface en hora en funcion\n");
            nohora=0;
            wfd nohora = setclock();
            break;

        default:
            printf("la interface esta en hora\n");
            printf("cm1l responde %x\n",cm1lack.fb);

```

```

        if (cm11ack.fb == checksum)
        {
            printf("el cchecksum de direccion es correcto
%x\n",cm11ack.fb);
            deviceready = 0;
            wfd deviceready = x10cm11ack();
            if (deviceready) return 1; //la interface devolvió un
0x55 al 0x00
            else return 0; //la interface devolvió otra cosa o timeout
        }
        else return 0; //el checksum de function no es correcto
        break;
    }
    if (nohora) goto againfunc; //la interface no estaba en hora, luego de sincronizarla se
vuelve a escribir la dirección
}
/**/ Beginheader x10 */
scofunc int x10 (char house, char device, char func); //declaracion
/**/ endheader */

scofunc int x10 (char house, char device, char func)
/* Esta es la funcion que se llama desde el main. Recibe el house code, device code y
la funcion a realizar
* Incluye dentro el seteo de la direccion y la funcion para que sea mas facil
llamarla*/
{
    int addr, funcion;
    addr = 0; funcion = 0;
    wfd addr = x10cm11addr(house, device); //hace el handshake de dirección
    wfd funcion = x10cm11function (house, func); //hace el handshake de funcion
    return 1;
}

```

A.2: Código interfaz web

A.2.1 Cgis:

A.2.1.1 setcontrollers.pl

```
#!/usr/bin/perl -W

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use AsciiDB::TagFile;

#####
#Main                                     #
#####

my $q = new CGI;
my %tiehash;
my $tieObj = tie %tiehash, 'AsciiDB::TagFile',
                DIRECTORY => './tmp',
                SUFIX => '.controller',
                FILEMODE => 0644,
                SCHEMA => { ORDER => ['ip'] };

print CGI::header();
print "<DIV ALIGN=RIGHT>" . "Date: " . scalar(localtime) . "<br>\n" . "<\DIV>";

#miro si es admin sino salgo
if ($ENV{REMOTE_USER} ne "admin"){
    html_error("Para configurar dispositivos debe ser admin");
    exit(0);
}

my $page = $q->param('.page');
if ($page eq 'main_form') { #####Segunda pasada#####
    if ($q->param('send') eq 'add'){
        my($contlabel,$ip) = getformdata();
        $tiehash{"$contlabel"}{'ip'} = "$ip";
        $tieObj->sync();
        main_form(%tiehash);
    }

    elsif ($q->param('Rm') eq 'Borar todos'){
        `rm -rf /var/www/domotic/web/tmp/*.controller`;
```

```

        main_form(\"%tiehash);
    }
}
else { #####Primera pasada#####
    main_form(\"%tiehash);
}

#####
#Subs          #
#####

# Print html form
sub main_form {
    my $key;
    my %hash = shift;

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplform = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/setcontrollers.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );

    $tmplheader->param(
        title => "Dispositivos"
    );

    print $tmplheader->output();

    if (keys %tiehash){#hay dispositivos configurados
        print "<h4>Controladores Configurados:<\h4>";
        print "<table border='1'><tr><td><b>Nombre del
controlador<\b><\td><td><b>Ip addr<\b><\td><\tr>";
        foreach $key (sort keys %tiehash){
            print "<tr><td>$key<\td><td>$tiehash{$key} {'ip'}<\td><\tr>";
        }
        print "<\table><hr>";
    }
    print $tmplform->output();
    print $tmplfooter->output();
    exit(0);
}

# Get data del form
sub getformdata{
    my $contlabel =esc($q->param('controllername'));
    my $ip        =esc($q->param('controllerip'));

```



```

        return($contlabel,$ip);
    }

    # Escapaa caracteres raros
    sub esc {
        my ($ref) = @_ ;
        $ref =~ s/([^\a-zA-Z0-9.])\\$&/g;
        return $ref;
    }

    # Manda pag completa de error
    sub html_error{
        my $tmplerror = HTML::Template->new(
            filename => "/var/www/domotic/web/tmpl/error.tmpl"
        );
        $tmplerror->param(
            title    => "Controlladores",
            msg      => shift
        );
        print $tmplerror->output();
    }

```

A.2.1.2 setaddr.pl

```

#!/usr/bin/perl -W

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use IO::Socket;
#####
#Main                                     #
#####

my $q = new CGI;
print CGI::header();
print "<DIV ALIGN=RIGHT> " . "Date: " . scalar(localtime) . "<br>\n" . "<\DIV>";
#miro si es admin sino lo pateo y salgo
if ($ENV{REMOTE_USER} ne "admin"){
    html_error("Para configurar direccion a dispositivos debe ser admin");
    exit(0);
}
my @formdata;
my $page = $q->param('.page');

```

```

if ($page eq 'main_form') { #####Segunda pasada#####
    if ($q->param('send') eq 'Configurar'){
        @formdata = getformdata();
        my $rbb = ${&connect}; #devuelve el handler y maneja errores en caso de
no poder conectar
        chomp(my $buf= <$rbb>);
        if ($buf eq "ready"){
            print $rbb "850","$formdata[0]","$formdata[1]","02","\n";
            #chomp($buf= <$rbb>);
            html_error("Se ha configurado la nueva direccion al
dispositivo X10");
            exit(0);
        }
        html_error("El Rabbit no parece estar listo..");
        exit(0);
    }
}
else { #####Primera pasada#####
    main_form();
    exit(0);
}

#####
#Subs          #
#####

# Print html form
sub main_form {

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplform = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/setaddr.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );

    $tmplheader->param(
        #diag_date => datetime(time()),
        #username => $ENV{REMOTE_USER}
        title => "Configuracion de direccion a dispositivos"
    );

    print $tmplheader->output();
    print $tmplform->output();
    print $tmplfooter->output();
    return;
}

```

```

# Get data del form
sub getformdata{
    my $housecode =esc($q->param('housecode'));
    my $devicecode =esc($q->param('devicecode'));
    return($housecode, $devicecode);
}

# Escapaa caracteres raros
sub esc {
    my ($ref) = @_;
    $ref =~ s/([a-zA-Z0-9])/\&/g;
    return $ref;
}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(
        title    => "Configuracion de direccion a dispositivo",
        msg      => shift
    );
    print $tmplerror->output();
}

sub connect {
    my ($host, $port) = ("192.168.4.10", "9000");#modificarpara que se configure
    my $handle = IO::Socket::INET->new(Proto => "tcp",
        PeerAddr => $host,
        PeerPort => $port);
    if(defined($handle)){
        $handle->autoflush(1);
        return ($handle);
    }
    else {
        html_error("No se puede conectar al rabbit on $host:$port");
        exit(1);
    }
}

```

A.2.1.3 deviceconf.pl

```

#!/usr/bin/perl -W

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);

```

```

use HTML::Template;
use AsciiDB::TagFile;

#####
#Main                                     #
#####

my $q = new CGI;
my %tiehash;
my %tiehashcont;
my $tieObj = tie %tiehash,'AsciiDB::TagFile',
                DIRECTORY => '../tmp',
                SUFIX => '.label',
                FILEMODE => 0644,
                SCHEMA => { ORDER => ['hc', 'dc', 'cont'] };

my $tieObjcont = tie %tiehashcont,'AsciiDB::TagFile',
                DIRECTORY => '../tmp',
                SUFIX => '.controller',
                FILEMODE => 0644,
                SCHEMA => { ORDER => ['ip'] };

print CGI::header();
print "<DIV ALIGN=RIGHT>" . "Date: " . scalar(localtime) . "<br>\n" . "<\DIV>";

#miro si es admin sino lo pateo y salgo
if ($ENV{REMOTE_USER} ne "admin"){
    html_error("Para configurar dispositivos debe ser admin");
    exit(0);
}

my $page = $q->param('.page');
if ($page eq 'main_form') { #####Segundapasada#####
    if ($q->param('xsend') eq 'add'){
        my($hc, $dc, $lb,$cont) = getformdata('x');
        $tiehash{"$lb"}{'hc'} = "$hc";
        $tiehash{"$lb"}{'dc'} = "$dc";
        $tiehash{"$lb"}{'cont'} = "$cont";
        $tieObj->sync();
        main_form(\%tiehash);
    }
    elsif ($q->param('psend') eq 'add'){
        my($hc, $dc, $lb, $cont) = getformdata('p');
        $tiehash{"$lb"}{'hc'} = "$hc";#guardo variables
        $tiehash{"$lb"}{'dc'} = "$dc";
        $tiehash{"$lb"}{'cont'} = "$cont";
        $tieObj->sync();
        main_form(\%tiehash);
    }

    elsif ($q->param('Rm') eq 'Borar todas'){

```

```

        `rm -rf /var/www/domotic/web/tmp/*.$label`;
        main_form(`%tiehash`);
    }
}
else { #####Primera pasada#####
    main_form(`%tiehash`);
}

#####
#Subs          #
#####

# Print html form
sub main_form {
    my $key;
    my %hash = shift;

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplform = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/deviceconf.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );

    $tmplheader->param(
        #diag_date => datetime(time()),
        #username => $ENV{REMOTE_USER}
        title => "Dispositivos"
    );

    print $tmplheader->output();

    if (keys %tiehash){#si hay dispositivos configurados
        print "<h4>Dispositivos Configurados:</h4>";
        print "<table border='1'><tr><td><b>Tipo de Dispositivo</b></td>
            <td><b>Label</b></td><td><b>Controlador</b></td></tr>";
        foreach $key (sort keys %tiehash){
            my $dt;
            if ($tiehash{$key}{'hc'} eq "port"){
                $dt="Directamente conectado en el puerto
$tiehash{$key}{'dc'}";
            }
            else {
                $dt = "X10";
            }
            print
"<tr><td>$dt</td><td>$key</td><td>$tiehash{$key}{'cont'}</td></tr>";

```

```

    }
    print "<\table><hr>";
}

my @controllers = keys %tiehashcont;
my @loop_data = ();
while(@controllers){
    my %row_data;
    $row_data{controller} = shift @controllers;
    push(@loop_data, \%row_data);
}
$tplform->param(LOOP_ONE => \@loop_data);
print $tplform->output();
print $tplfooter->output();
exit(0);
}

# Get data del form
sub getformdata{
    my $dt = shift;# x o p
    if ($dt eq 'x'){
        my $housecode =esc($q->param('housecode'));
        my $devicecode =esc($q->param('devicecode'));
        my $label =esc($q->param('xlabel'));
        my $controller =esc($q->param('controller'));
        return($housecode, $devicecode, $label, $controller);
    }
    elsif ($dt eq 'p'){
        my $housecode ="port";
        my $devicecode =esc($q->param('portcode'));
        my $label =esc($q->param('plabel'));
        my $controller =esc($q->param('controller'));
        return($housecode, $devicecode, $label, $controller);
    }
}

# Escapea caracteres raros
sub esc {
    my ($ref) = @_;
    $ref =~ s/([^\a-zA-Z0-9])/\\$&/g;
    return $ref;
}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(

```

```

        title    => "Dispositivos",
        msg      => shift
    );
    print $tmplerror->output();
}

```

A.2.1.4 progevent.pl

```
#!/usr/bin/perl -W
```

```

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use AsciiDB::TagFile;
use Data::Dumper;
use Config::Crontab;

```

```
#####
```

```
#Main                                #
```

```
#####
```

```

my $q = new CGI;
my %tiehash;
my %tiehashcron;
my $tieObj = tie %tiehash,'AsciiDB::TagFile',
               DIRECTORY => '../tmp',
               SUFIX => '.label',
               FILEMODE => 0644,
               SCHEMA => { ORDER => ['hc', 'dc', 'cont'] };

```

```

my $tieObjcron = tie %tiehashcron,'AsciiDB::TagFile',
                   DIRECTORY => '../tmp',
                   SUFIX => '.cron',
                   FILEMODE => 0644,
                   SCHEMA => { ORDER => ['m', 'h', 'dow', 'la', 'ac'] };

```

```

print CGI::header();
print "<DIV ALIGN=RIGHT>" . "Date: " . scalar(localtime) . "<br>\n" . "</DIV>";
my $time = time;
my $page = $q->param('.page');
if ($page eq 'main_form') { #####Segunda pasada#####
    if ($q->param('Send') eq 'send') {
        my ($dow,$hour,$minute,$label,$action)= getformdata();
        cronhandle('add',"$minute $hour * * $dow","/var/www/domotic/web/cgi-
bin/cmdxlabel.pl $label $action");
    }
}

```

```

        $tiehashcron{"$time"}{'m'} = "$minute";#guardo variables
        $tiehashcron{"$time"}{'h'} = "$hour";
        $tiehashcron{"$time"}{'dow'} = "$dow";
        $tiehashcron{"$time"}{'la'} = "$label";
        $tiehashcron{"$time"}{'ac'} = "$action";
        $tieObjcron->sync();
        main_form();
        exit(0);
    }
elseif ($q->param('Rm') eq 'Borar todas'){
    `rm -rf /var/www/domotic/web/tmp/*.*cron`;
    main_form();
    exit(0);
}
}
else { #####Primera pasada#####
    main_form(%tiehash);
    exit(0);
}

#####
#Subs      #
#####

# Print html form
sub main_form {

    my $key;
    my %hash = shift;

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplform = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/progevent.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );
    $tmplheader->param(
        title => "Tareas Programadas"
    );

    my @labels = keys %tiehash;
    my @loop_data = ();
    while(@labels){
        my %row_data; # fresh hash
        $row_data{LABEL} = shift @labels;
        push(@loop_data, \%row_data);
    }
}

```



```

    }
    $tmplform->param(LOOP_ONE => \@loop_data);

    print $tmplheader->output();

    if (keys %tiehashcron){#si hay tareas croneadas
        print "<h4>Eventos Programados:</h4>";
        print "<table
border='1'><tr><td><b>Label</b></td><td><b>Action</b></td>
        <td><b>Hora</b></td><td><b>Minuto</b></td><td><b>Dia de la
semana(0-6)</b></td></tr>";
        foreach $key (sort keys %tiehashcron){
            my $cmd;
            if ($tiehashcron{$key}{'ac'} eq "02"){
                $cmd="On";
            }
            else {
                $cmd = "Off";
            }
            print "<tr><td>$tiehashcron{$key}{'la'}</td>
                <td>$cmd</td>
                <td>$tiehashcron{$key}{'h'}</td>
                <td>$tiehashcron{$key}{'m'}</td>
                <td>$tiehashcron{$key}{'dow'}</td>
                </tr>";
        }
        print "</table><hr>";
    }

    print $tmplform->output();
    print $tmplfooter->output();
}

# Get data del form
sub getformdata{
    my $dow    =esc($q->param('dow'));
    my $hour   =esc($q->param('hour'));
    my $minute =esc($q->param('minute'));
    my $label  =esc($q->param('label'));
    my $action =esc($q->param('action'));
    return($dow,$hour,$minute,$label,$action);
}

# Escapea caracteres raros
sub esc {
    my ($ref) = @_;
    $ref =~ s/([a-zA-Z0-9])/\&/g;
    return $ref;
}

```

```

}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(
        title    => "Simulpres",
        msg      => shift
    );
    print $tmplerror->output();
}
#my @caca = cronhandle('del','24 1 * * *','/bin/caca2');
#cron#####
# *min
# *hour
# *dom
# *month
# *dow
#####
#Interfaz para acceder al cron
#paranetros
#    action:add/del/qry
#    datetime
#    command
sub cronhandle{
    my $ct = new Config::Crontab( -owner => 'www-data');
    my ($action,$datetime,$command)= @_ ;
    $ct->read;

    if ($action eq "add"){
        my $ev = new Config::Crontab::Event( -datetime => $datetime,
                                              -command => $command);
        my $bl = new Config::Crontab::Block;
        $bl->last($ev);
        $ct->last($bl);
        $ct->write;
        return 0;
    }

    if ($action eq "del"){
        $ct->remove($ct->select(-datetime => $datetime,
                              -command => $command));
        $ct->write;
        return 0;
    }

    if ($action eq "qry"){
        my @events = ();

```

```

    for my $event ( $ct->select( -type => 'event' ) ) {
        push @events,$event->dump . "\n";
    }
    return @events;
}
}

```

A.2.1.5 *simulpres.pl*

```
#!/usr/bin/perl -W
```

```

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use AsciiDB::TagFile;
use Data::Dumper;

```

```

#####
#Main                                     #
#####
my $q = new CGI;
my %tiehash;
my $tieObj = tie %tiehash,'AsciiDB::TagFile',
                DIRECTORY => './tmp',
                SUFIX => '.label',
                FILEMODE => 0644,
                SCHEMA => { ORDER => ['hc', 'dc'] };
print CGI::header();
print "<DIV ALIGN=RIGHT>" . "Date: " . scalar(localtime) . "<br>\n" . "</DIV>";
my $page = $q->param('.page');
my @formdata;
if ($page eq 'main_form') { #####Segunda pasada#####
    if ($q->param('Send') eq 'send'){
        @formdata = getformdata();
        html_error(Dumper(\@formdata));
        exit(0);
    }
}
else { #####Primera pasada#####
    main_form(\%tiehash);
    exit(0);
}

#####
#Subs                                     #

```

#####

Print html form

sub main_form {

my \$key;

my %hash = shift;

my \$tmplheader = HTML::Template->new(

filename => "/var/www/domotic/web/tmpl/header.tpl"

);

my \$tmplform = HTML::Template->new(

filename => "/var/www/domotic/web/tmpl/simulpres.tpl"

);

my \$tmplfooter = HTML::Template->new(

filename => "/var/www/domotic/web/tmpl/footer.tpl"

);

\$tmplheader->param(

title => "Tareas Programadas"

);

my @labels = keys %tiehash;

#my @labels = qw(I Am Cool);

my @loop_data = ();

while(@labels){

my %row_data; # fresh hash

\$row_data{LABEL} = shift @labels;

push(@loop_data, \%row_data);

}

\$tmplform->param(LOOP_ONE => \@loop_data);

print \$tmplheader->output();

print \$tmplform->output();

print \$tmplfooter->output();

}

Get data del form

sub getformdata{

#on

my \$dateon =esc(\$q->param('dateon'));

my \$horaon =esc(\$q->param('horaon'));

my \$minon =esc(\$q->param('minon'));

#off

my \$dateoff =esc(\$q->param('dateoff'));

my \$horaoff =esc(\$q->param('horaoff'));

my \$minoff =esc(\$q->param('minoff'));

my \$label =esc(\$q->param('label'));

```

    return($dateon,$horaon,$minon,$dateoff,$horaoff,$minoff,$label);
}

# Escapaa caracteres raros
sub esc {
    my ($ref) = @_ ;
    $ref =~ s/([^\a-zA-Z0-9])\\$&/g;
    return $ref;
}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(
        title    => "Simulpres",
        msg      => shift
    );
    print $tmplerror->output();
}

```

A.2.1.6 monitor.pl

```

#!/usr/bin/perl -W

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use IO::Socket;

#####
#Main                                     #
#####

my $q = new CGI;
print CGI::header();

my $cmdx10 = "900\n";
my $rbb = ${&connect}; #devuelve el handler y maneja errores en caso de no poder
conectar
chomp(my $buf= <$rbb>);
if ($buf eq "ready"){
    print $rbb "$cmdx10"; #mando comando a sock
    chomp($buf= <$rbb>);
    main_form($buf);
    exit(0);
}

```

```

    }
    html_error("El Rabbit no parece estar listo..");
    exit(1);

#####
#Subs                                #
#####

# Print html form
sub main_form {

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplmon    = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/monitor.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );
    $tmplheader->param(
        title    => "Monitoreo de niveles instantaneos",
        refresh  => 1
    );

    $tmplmon ->param(
        temp    => shift
    );

    print $tmplheader->output();
    print $tmplmon->output();
    print $tmplfooter->output();
}

# Escapea caracteres raros
sub esc {
    my ($ref) = @_ ;
    $ref =~ s/([a-zA-Z0-9])/\&/g;
    return $ref;
}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(
        title    => "Monitoreo de niveles instantaneos",
        msg      => shift
    );

```

```

    print $tmplerror->output();
}

sub connect {
    my ($host, $port) = ("192.168.4.10", "9000"); #modificarpara que se configure
    my $handle = IO::Socket::INET->new(Proto => "tcp",
                                       PeerAddr => $host,
                                       PeerPort => $port);
    if(defined($handle)){
        $handle->autoflush(1);
        return ($handle);
    }
    else {
        html_error("No se puede conectar al rabbit on $host:$port");
        exit(1);
    }
}

```

cmdxlabel.pl

```

#!/usr/bin/perl -W

use strict;
use AsciiDB::TagFile;
use IO::Socket;

#####
#Main                                     #
#####
my %tiehash;
my %tiehashcont;
my $tieObj = tie %tiehash, 'AsciiDB::TagFile',
               DIRECTORY => '/var/www/domotic/web/tmp',
               SUFIX     => '.label',
               FILEMODE  => 0644,
               SCHEMA    => { ORDER => ['hc', 'dc', 'cont'] };

my $tieObjcont = tie %tiehashcont, 'AsciiDB::TagFile',
               DIRECTORY => './tmp',
               SUFIX     => '.controller',
               FILEMODE  => 0644,
               SCHEMA    => { ORDER => ['ip'] };

my $label = shift;
my $action = shift;

```

```

my ($cmd,$ip) = mkcmd($label,$action); #paso(label,02|03)
my $rbb = ${&connect($ip)}; #devuelve el handler y maneja errores en caso de no
poder conectar
chomp(my $buf= <$rbb>);
if ($buf eq "ready"){
    print "enviando: $cmd";
    print $rbb "$cmd"; #mando comando a sock
    chomp($buf= <$rbb>);
    print "$buf\n";
    exit(0);
}
print "El Rabbit no parece estar listo..";
exit(1);

```

```

#####
#Subs                                     #
#####

```

```

sub mkcmd {
    my $label = shift;
    my $action = shift;
    my $cmd;
    my $ip = $tiehashcont{$tiehash{$label}{'cont'}}{'ip'};
    if ($tiehash{$label}{'hc'} eq "port"){
        $cmd ="700"."$tiehash{$label}{'dc'}"."$action"."n";
    }
    else{
        $cmd ="800"."$tiehash{$label}{'hc'}"."$tiehash{$label}{'dc'}"."$action"."n";
    }
    return($cmd,$ip);
}
sub connect {
    my $host = shift;
    my $port = "9000";
    my $handle = IO::Socket::INET->new(Proto => "tcp",
                                      PeerAddr => $host,
                                      PeerPort => $port)
        ||die "No se puede conectar al rabbit on $host:$port";
    $handle->autoflush(1);
    return ($handle);
}

```

cmdxlabel.pl

```
#!/usr/bin/perl -W
```



```

use strict;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use HTML::Template;
use AsciiDB::TagFile;
use IO::Socket;

#####
#Main                                     #
#####

my $q = new CGI;
my %tiehash;
my %tiehashcont;
my $tieObj = tie %tiehash,'AsciiDB::TagFile',
               DIRECTORY => './tmp',
               SUFIX    => '.label',
               FILEMODE  => 0644,
               SCHEMA    => { ORDER => ['hc', 'dc', 'cont'] };

my $tieObjcont = tie %tiehashcont,'AsciiDB::TagFile',
                   DIRECTORY => './tmp',
                   SUFIX    => '.controller',
                   FILEMODE => 0644,
                   SCHEMA    => { ORDER => ['ip'] };

print CGI::header();
print "<DIV ALIGN=RIGHT>" . "Date: " . scalar(localtime) . "<br>\n" . "</DIV>";
my $page = $q->param('.page');
my @formdata;
if ($page eq 'main_form') { #####Segunda pasada#####
    if ($q->param('Send') eq 'send'){
        @formdata = getformdata();
        my ($cmd,$ip)= mkcmd($formdata[0],$formdata[1]); #paso(label,02|03)
        my $rbb = ${&connect($ip)}; #devuelve el handler y maneja errores
        en caso de no poder conectar
        chomp(my $buf= <$rbb>);
        if ($buf eq "ready"){
            print $rbb "$cmd"; #mando comando a sock
            chomp($buf= <$rbb>);
            main_form("TRUE");
            exit(0);
        }
        html_error("El Rabbit no parece estar listo..");
        exit(0);
    }
}
else { #####Primera pasada#####
    main_form();
    exit(0);
}

```

```
}
```

```
#####
#Subs          #
#####
```

```
# Print html form
```

```
sub main_form {

    my $tmplheader = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/header.tmpl"
    );
    my $tmplform = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/xlabel.tmpl"
    );
    my $tmplfooter = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/footer.tmpl"
    );
    $tmplheader->param(
        title => "Tareas Programadas"
    );

    my @labels = keys %tiehash;
    my @loop_data = ();
    while(@labels){
        my %row_data; # fresh hash
        $row_data{LABEL} = shift @labels;
        push(@loop_data, \%row_data);
    }
    $tmplform->param(RETURN => shift,
        LOOP_ONE => \@loop_data);

    print $tmplheader->output();
    print $tmplform->output();
    print $tmplfooter->output();
}
```

```
# Get data del form
```

```
sub getformdata{
    my $label =esc($q->param('label'));
    my $action =esc($q->param('action'));

    return($label,$action);
}
```

```
# Escapea caracteres raros
```

```
sub esc {
```

```

my ($ref) = @_;
$ref =~ s/([a-zA-Z0-9])\\$&/g;
return $ref;
}

# Manda pag completa de error
sub html_error{
    my $tmplerror = HTML::Template->new(
        filename => "/var/www/domotic/web/tmpl/error.tmpl"
    );
    $tmplerror->param(
        title    => "xlabel",
        msg      => shift
    );
    print $tmplerror->output();
}

sub mkcmd {
    my $label = shift;
    my $action = shift;
    my $cmd;
    my $ip = $tiehashcont{$tiehash{$label}{'cont'}}{'ip'};
    if ($tiehash{$label}{'hc'} eq "port"){
        $cmd = "700"."$tiehash{$label}{'dc'}"."$action"."n";
    }
    else{
        $cmd = "800"."$tiehash{$label}{'hc'}"."$tiehash{$label}{'dc'}"."$action"."n";
    }
    return($cmd, $ip);
}

sub connect {
    my $host = shift;
    my $port = "9000";
    my $handle = IO::Socket::INET->new(Proto => "tcp",
        PeerAddr => $host,
        PeerPort => $port);
    if(defined($handle)){
        $handle->autoflush(1);
        return ($handle);
    }
    else {
        html_error("No se puede conectar al rabbit on $host:$port");
        exit(1);
    }
}

```

A.2.2 Html Templates:

A.2.2.1 header.tpl

```
<HTML>
<HEAD><TITLE><TMPL_VAR NAME="title"></TITLE>
<TMPL_IF NAME="REFRESH">
<META HTTP-EQUIV="REFRESH" CONTENT="2">
<META HTTP-EQUIV="Expires" CONTENT="0">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</TMPL_IF>
</HEAD>
<BODY BGCOLOR="#CCFFCC">
<TD style="padding: 1.5pt; width: 28.68%;" width="28%"> </TD>
<H1><font size=6 face="Verdana, Arial, Helvetica, sans-serif">66.99 TRABAJO
PROFESIONAL</font></H1><HR>
```

A.2.2.2 footer.tpl

```
<HR><BR>
</BODY>
</HTML>
```

A.2.2.3 error.tpl

```
<HTML>
<HEAD><TITLE><TMPL_VAR NAME="title"></TITLE></HEAD>
<BODY BGCOLOR="#CCFFCC">
<TD style="padding: 1.5pt; width: 28.68%;" width="28%"> </TD>
<H1><font size=6 face="Verdana, Arial, Helvetica, sans-serif">66.99 TRABAJO
PROFESIONAL</font></H1><HR><BR>

<TMPL_VAR NAME="msg">

<HR><BR>
</BODY>
</HTML>
```

A.2.2.4 setcontrollers.tmpl

```

<h4>Configurar dispositivos controladores:</h4>
<FORM METHOD=POST ACTION="/cgi-bin/setcontrollers.pl">
<p>Contrller(Nombre):<input type='text' size='10' maxlength='10'
name='controllername' value="">
Direccion IP (Ej 192.168.1.1):<input type='text' size='15' maxlength='15'
name='controllerip' value=""></p>
<INPUT TYPE="SUBMIT" NAME="send" VALUE="add">
<INPUT TYPE="SUBMIT" NAME="Rm" VALUE="Borrar todos">
<INPUT TYPE="HIDDEN" NAME=".page" VALUE="main_form">

```

A.2.2.5 setaddr.tmpl

```

<h4>Setear direccion a dispositivos X10:</h4>
<p>Para configurarle la direccion a un dispositivo X10 debera energizar el mismo </p>
<p>y dentro de los primeros 15 segundos presionar el boton "Configurar" can la
direccion que desee</p>
<FORM METHOD=POST ACTION="/cgi-bin/setaddr.pl">
<p>House Code:<select name="housecode">
  <option value="06" > "A"
  <option value="14" > "B"
  <option value="02" > "C"
  <option value="10" > "D"
  <option value="01" > "E"
  <option value="09" > "F"
  <option value="05" > "G"
  <option value="08" > "H"
  <option value="07" > "I"
  <option value="15" > "J"
  <option value="03" > "K"
  <option value="11" > "L"
  <option value="00" > "M"
  <option value="08" > "N"
  <option value="04" > "O"
  <option value="12" > "P"
</select>

Device Code:<select name="devicecode">
  <option value="06" > "1"
  <option value="14" > "2"
  <option value="02" > "3"
  <option value="10" > "4"
  <option value="01" > "5"
  <option value="09" > "6"
  <option value="05" > "7"

```

```

    <option value="08" > "8"
    <option value="07" > "9"
    <option value="15" > "10"
    <option value="03" > "11"
    <option value="11" > "12"
    <option value="00" > "13"
    <option value="08" > "14"
    <option value="04" > "15"
    <option value="12" > "16"
  </select>
  <INPUT TYPE="SUBMIT" NAME="send" VALUE="Configurar">
  <INPUT TYPE="HIDDEN" NAME=".page" VALUE="main_form">

```

A.2.2.6 deviceconf.tpl

```

<h4>Configurar Dispositivos X10:</h4>
<FORM METHOD=POST ACTION="/cgi-bin/deviceconf.pl">
<p>House Code:<select name="housecode">
    <option value="06" > "A"
    <option value="14" > "B"
    <option value="02" > "C"
    <option value="10" > "D"
    <option value="01" > "E"
    <option value="09" > "F"
    <option value="05" > "G"
    <option value="08" > "H"
    <option value="07" > "I"
    <option value="15" > "J"
    <option value="03" > "K"
    <option value="11" > "L"
    <option value="00" > "M"
    <option value="08" > "N"
    <option value="04" > "O"
    <option value="12" > "P"
  </select>

```

```

Device Code:<select name="devicecode">
    <option value="06" > "1"
    <option value="14" > "2"
    <option value="02" > "3"
    <option value="10" > "4"
    <option value="01" > "5"
    <option value="09" > "6"
    <option value="05" > "7"
    <option value="08" > "8"
    <option value="07" > "9"
    <option value="15" > "10"
    <option value="03" > "11"

```

```

        <option value="11" > "12"
        <option value="00" > "13"
        <option value="08" > "14"
        <option value="04" > "15"
        <option value="12" > "16"
    </select></p>
<p>Label(Nombre):<input type='text' size='10' maxlength='10' name='xlabel'
value="">
<TMPL_IF LOOP_ONE>
    Controlador:<select name="controller">
        </TMPL_IF>
        <TMPL_LOOP LOOP_ONE>
            <option value="<TMPL_VAR NAME="controller">" > "<TMPL_VAR
NAME="controller">"
        </TMPL_LOOP>
        <TMPL_IF LOOP_ONE>
            </select></p>
</TMPL_IF>

<INPUT TYPE="SUBMIT" NAME="xsend" VALUE="add">
<INPUT TYPE="SUBMIT" NAME="Rm" VALUE="Borar todas">

<h4>Configurar Dispositivos directamente conectados:</h4>
<FORM METHOD=POST ACTION="/cgi-bin/deviceconf.pl">
Puerto Numero:<select name="portcode">
    <option value="00" > "0"
    <option value="01" > "1"
    <option value="02" > "2"
    <option value="03" > "3"
    <option value="04" > "4"
    <option value="05" > "5"
</select></p>
<p>Label(Nombre):<input type='text' size='10' maxlength='10' name='plabel'
value="">
<TMPL_IF LOOP_ONE>
    Controlador:<select name="controller">
        </TMPL_IF>
        <TMPL_LOOP LOOP_ONE>
            <option value="<TMPL_VAR NAME="controller">" > "<TMPL_VAR
NAME="controller">"
        </TMPL_LOOP>
        <TMPL_IF LOOP_ONE>
            </select></p>
</TMPL_IF>
<INPUT TYPE="SUBMIT" NAME="psend" VALUE="add">
<INPUT TYPE="SUBMIT" NAME="Rm" VALUE="Borar todas">

<INPUT TYPE="HIDDEN" NAME=".page" VALUE="main_form">

```

A.2.2.6 progevent.tmpl

```
<h4>Configurar tareas programadas:</h4>
<form action="/cgi-bin/progevent.pl" method="post">
<p>Dia de la semana:
    <select name="dow">
        <option value="0" > "Domingo"
        <option value="1" > "Lunes"
        <option value="2" > "Martes"
        <option value="3" > "Miercoles"
        <option value="4" > "Jueves"
        <option value="5" > "Viernes"
        <option value="6" > "Sabado"
        <option value="*" > "TODOS"
    </select>
```

Hora:

```
    <select name="hour">
        <option value="0" > "0"
        <option value="1" > "1"
        <option value="2" > "2"
        <option value="3" > "3"
        <option value="4" > "4"
        <option value="5" > "5"
        <option value="6" > "6"
        <option value="7" > "7"
        <option value="8" > "8"
        <option value="9" > "9"
        <option value="10" > "10"
        <option value="11" > "11"
        <option value="12" > "12"
        <option value="13" > "13"
        <option value="14" > "14"
        <option value="15" > "15"
        <option value="16" > "16"
        <option value="17" > "17"
        <option value="18" > "18"
        <option value="19" > "19"
        <option value="20" > "20"
        <option value="21" > "21"
        <option value="22" > "22"
        <option value="23" > "23"
    </select>
```

Minuto:

```
    <select name="minute">
```


<option value="0" > "0"
<option value="1" > "1"
<option value="2" > "2"
<option value="3" > "3"
<option value="4" > "4"
<option value="5" > "5"
<option value="6" > "6"
<option value="7" > "7"
<option value="8" > "8"
<option value="9" > "9"
 <option value="10" > "10"
<option value="11" > "11"
<option value="12" > "12"
<option value="13" > "13"
<option value="14" > "14"
<option value="15" > "15"
<option value="16" > "16"
<option value="17" > "17"
<option value="18" > "18"
<option value="19" > "19"
 <option value="20" > "20"
<option value="21" > "21"
<option value="22" > "22"
<option value="23" > "23"
<option value="24" > "24"
<option value="25" > "25"
<option value="26" > "26"
<option value="27" > "27"
<option value="28" > "28"
<option value="29" > "29"
 <option value="30" > "30"
<option value="31" > "31"
<option value="32" > "32"
<option value="33" > "33"
<option value="34" > "34"
<option value="35" > "35"
<option value="36" > "36"
<option value="37" > "37"
<option value="38" > "38"
<option value="39" > "39"
 <option value="40" > "40"
<option value="41" > "41"
<option value="42" > "42"
<option value="43" > "43"
<option value="44" > "44"
<option value="45" > "45"
<option value="46" > "46"
<option value="47" > "47"
<option value="48" > "48"
<option value="49" > "49"

```

        <option value="50" > "50"
        <option value="51" > "51"
        <option value="52" > "52"
        <option value="53" > "53"
        <option value="54" > "54"
        <option value="55" > "55"
        <option value="56" > "56"
        <option value="57" > "57"
        <option value="58" > "58"
        <option value="59" > "59"

</select></p>

<p>Accion:
    <select name="action">
        <option value="02" > "encender"
        <option value="03" > "Apagar"
    </select>

<TMPL_IF LOOP_ONE>
    Dispositivo:
    <select name="label">
</TMPL_IF>

    <TMPL_LOOP LOOP_ONE>
        <option value="<TMPL_VAR NAME="label">" > "<TMPL_VAR
NAME="label">"
    </TMPL_LOOP>

<TMPL_IF LOOP_ONE>
    </select></p>
</TMPL_IF>

    <input type="SUBMIT" name="Send" value="send">
    <input TYPE="SUBMIT" NAME="Rm" value="Borar todas">
    <input type="HIDDEN" name=".page" value="main_form">
</form>

```

A.2.2.7 simulpres.tmp

<h4>Setear Simulacion de Presencia:</h4>

<p>Debera seleccionar un intervalo de tiempo y un dispositivo sobre el cual realizar la simulacion.</p>

```
<form action="/cgi-bin/simulpres.pl" method="post">
  <script type='text/JavaScript' src='/html/scw.js'></script>

  <p>Seleccione fecha on
  <input id='dateon'type='text' size='12' name='dateon' value="" />
  <img src='/imagenes/calwidget-icon.gif' title='Toque Aqui' alt='Toque Aqui'
  onclick="scwShow(scwID('dateon'),event);" />
  Hora(0-23) <input type='text'size='4' maxlength='2' name='horaon' value="" />
  Minuto(0-59)<input type='text'size='4' maxlength='2' name='minon' value=""
/>
  </p>

  <p>Seleccione fecha off
  <input id='dateoff'type='text' size='12' name='dateoff' value="" />
  <img src='/imagenes/calwidget-icon.gif' title='Toque Aqui' alt='Toque Aqui'
  onclick="scwShow(scwID('dateoff'),event);" />
  Hora(0-23) <input type='text'size='4' maxlength='2' name='horaoff' value="" />
  Minuto(0-59)<input type='text'size='4' maxlength='2' name='minoff' value=""
/>
  </p><br>

  <TMPL_IF LOOP_ONE>
  <p><div align="center">
  <select name="label">
  </TMPL_IF>

  <TMPL_LOOP LOOP_ONE>
    <option value="<TMPL_VAR NAME="label">" > "<TMPL_VAR
NAME="label">"
  </TMPL_LOOP>

  <TMPL_IF LOOP_ONE>
  </select></div></p>
  </TMPL_IF>

  <input type="SUBMIT" name="Send" value="send">
  <input type="HIDDEN" name=".page" value="main_form">
</form>
```

A.2.2.8 monitor.tmp

```
<TMPL_IF NAME="TEMP">
<H3>Temperatura Actual tomada en el contolador 1: <TMPL_VAR
NAME="TEMP"> &deg C</H3>
</TMPL_IF>
```

A.2.2.9 xlabel.tmp

```

<TMPL_IF NAME="RETURN">
<p>El comando X10 se ejecuto con exito</p>
</TMPL_IF>
<h4>Seleccione el dispositivo y el estado</h4>
<form action="/cgi-bin/xlabel.pl" method="post">
    <script type="text/javascript">

cc=0;
function changeimage() {
if (cc==0) {
    cc=1;
    document.getElementById('myimage').src="/imagenes/bulbon.gif";
    document.getElementById('action').value="02";
    }
else{
    cc=0;
    document.getElementById('myimage').src="/imagenes/bulboff.gif";
    document.getElementById('action').value="03";
    }
}
</script>
<div align="center">
    <TMPL_IF LOOP_ONE>
    <p><select name="label">
    </TMPL_IF>

    <TMPL_LOOP LOOP_ONE>
        <option value="<TMPL_VAR NAME="label">" > "<TMPL_VAR
NAME="label">"
    </TMPL_LOOP>

    <TMPL_IF LOOP_ONE>
    </select>
    </TMPL_IF>
    <input type="SUBMIT" name="Send" value="send"></p>
    <input type="HIDDEN" name=".page" value="main_form">
    <input type="HIDDEN" name="action" id='action' value="03">
</form></div>

```


Apéndice C. Bibliografía

- Introducción al Proyecto de Ingeniería
Ing. Enrique Villamil García, Dr. Miguel J. García Hernández
Diciembre de 2003
- Military Handbook MIL-HDBK-217F Reliability Prediction of Electronic Equipment
United States Department of Defence
Enero de 1990
- Dynamic C User Manual
060313 • 019-0125-E
Z-World Rabbit Semiconductor 2007
- RabbitCore RCM4400W C-Programmable Wi-Fi Core Module User's Manual
019-0160 • 070330-A
Z-World Rabbit Semiconductor 2007
- Dynamic C TCP/IP Users Manual Vol 1
019-0143 • 070720-C
Z-World Rabbit Semiconductor 2007
- Dynamic C TCP/IP Users Manual Vol 2
019-0143 • 070720-C
Z-World Rabbit Semiconductor 2007
- X.10-CM11 Interface Communication Protocol
X10home system's
X10home 2003
- Redes Globales de Información con Internet y TCP/IP, Principios básicos, protocolos y arquitectura, 3ra Edición.
Douglas E. Comer.
Prentice Hall - 1996

Para el diseño de software se utilizó exclusivamente bibliografía online. Se citan los sitios de acuerdo al tema

- Perl Core:
 - Perl cookbook: <http://www.unix.com.ua/oreilly/perl/cookbook/index.htm>
 - Perldoc: <http://perldoc.perl.org/perl.html>
- Perl modules:
 - Cpan: [http://www.is.informatik.uni-
duisburg.de/projects/SFgate/CPAN/](http://www.is.informatik.uni-duisburg.de/projects/SFgate/CPAN/)

- Cgi: <http://search.cpan.org/~lds/CGI.pm-3.40/CGI.pm>
- Html::Template: <http://search.cpan.org/~samtregar/HTML-Template-2.9/Template.pm>
- AsciiDB::TagFile: <http://search.cpan.org/~joserodr/AsciiDB-TagFile-1.06/TagFile.pm>
- Config::Crontab: <http://search.cpan.org/~scottw/Config-Crontab-1.21/Crontab.pm>

Para el cálculo de la confiabilidad de software se utilizo el siguiente software junto con la documetacio del mismo:

- <http://www.slingcode.com/smerfs/>