DISEÑO DE EQUIPOS ELECTRONICOS

Proyecto:

lamp*MATRIX*

Slavkin, Pablo (37267)

Prefacio agradecimientos

Gracias a Marcelo por haberme dedicado horas de discusión telefónica para ayudarme a decidir las ideas. A Eduardo por haberme recuperado el disco rígido de una perdida masiva de información, a Martín por a mis padres Zulema y Gregorio por haberme comprado el primer cartel, y darme soporte anímico y económico durante toda la carrera.

Se lo dedico con amor a Julieta.

Indice

1. Introducción

1.1. Historia, Antecedentes.

Desde la invención de la lampara de Edison hasta las nuevas pantallas de plasmas y leds de alto rendimiento el hombre ha gastado esfuerzos en tratar de lograr la transmisión de imágenes en gran escala con alta definición; y lo han logrado fehacientemente. Ya se ven en los estadios de fútbol, centros de capitales importantes, torres, cines, etc..

Hace ya unas décadas se podían ver las primeras pantallas gigantes basadas en lamparas incandescentes, y eran muy útiles para tanteadores de estadios deportivos, grandes indicadores, etc., así como también los indicadores mecánicos.

Recién con la llegada de los led a bajo precio se comenzaron a ver los carteles de pequeña y mediana escala para interiores como para indicadores de turnos en locales comerciales, hospitales, bancos, etc., carteles de publicidad, hora y temperatura, cotización de divisas, etc., que debido a su diminuto tamaño y que no disipan mucho calor, son muy propicios para este tipo de aplicaciones.

Luego en estos últimos años se comienzan a ver implementaciones de pantallas realmente gigantes a todo color para interior-exterior y con muy buena definición.

Con el crecimiento de esta tecnología tan prometedora la utilización de lamparillas de filamento parece totalmente obsoleta, con lo cual , este proyecto parecería no tener cabida hoy en día. Sin embargo se demostrará que resuelve sectores de mercado en donde la última tecnología no es necesaria y menos aún sus costos asociados

1.2. Definiciones. Glosario de términos.

- Pixel: En ocasiones se hará referencia a pixel de una imagen. Esto deberá interpretarse como la mínima unidad lumínica controlable dentro de la pantalla o imagen.
- Celda: La pantalla se dividirá en celdas inteligentes de 8x8 lamparas.
- SR: Shift Register.
- uP Microprocesador
- cps Cuadros por segundo

1.3. Justificación del proyecto.

Hoy en día las empresas invierten fuertes cantidades de dinero en publicidad de todo tipo. Las más comunes y masivas como le televisión, radio, diario, y las alternativas como ser carteles en las calles, autopistas, sponsors en ciertas actividades deportivas, eventos, etc. En esta última clase entra la cartelería dinámica que a diferencia de las estáticas permite transmitir información actualizada al instante y cambiarla cuantas veces sea necesaria a muy bajo costo. Esto permite tener gran cantidad de clientes publicando en el mismo espacio. Un ejemplo actual es el sistema “info-trans” en los colectivos, también los carteles indicadores del subte, muestran publicidades esporádicamente, etc.

Son contados los casos de empresas exitosas que no apelan a la publicidad para lograr sus resultados. Los llamados “creativos” agotan sus ideas en pos de ser mas originales y conseguir una penetración al mercado de la marca para la que trabajan. Por ello cualquier medio que les resulte atractivo, intentarán utilizarlo.

Es por ello que se espera el éxito de este tipo de herramientas ya que permitiría una difusión masiva, dinámica y versátil, y pondría al alcance de empresas con bajo presupuesto un producto de similares prestaciones que el equivalente de led’s. Asimismo que permitiría que una empresa propietaria de la pantalla alquile el espacio a terceros cobrando por el tiempo de aire.

La principal propuesta de este proyecto es entonces presentar un equipo que permita la transmisión de publicidad, noticias, efectos, información, etc. pero para ambientes exteriores y en grandes dimensiones y distancias de lectura.

2. Objetivos. (Propuesta técnica)

2.1. Finalidad del proyecto (a quien ayuda, para que sirve)

Diseñado para interior/exterior se adapta a aplicaciones de tipo tableros deportivos, estadios, publicitarios dinámicos, publicidad comercial dinámica, informantes, medidores de velocidad. en

autódromos, etc.; es decir todas las aplicaciones en las que se requiera de gran intensidad de pixel, grandes distancias al observador , utilización esporádica, pocos cuadros por seg.. y en donde el consumo no es un factor limitante.

Se destacan como potenciales clientes los siguientes:

- Empresas encargadas de brindar espacios publicitarios a terceras, adquiriendo estas pantallas y haciéndose cargo del mantenimiento y gestión, cobran a terceros por el tiempo de aire brindado.
- Empresas que requieran disponer de un espacio propio, adquiriendo la pantalla, se hacen cargo de la gestión y mantenimiento para su uso propio.
- Estadios, y grandes espacios de concentración de gente que requieran de una herramienta para transmitir información, como tanteadores, publicidad, etc.
- Carteles comerciales de pequeñas pymes en centros de comercio, marcando un estilo original de indicar la presencia de dicha sede.
- Carteles de mensajería en rutas y autopistas para indicar desvíos, estado del transito, etc. ya que el tamaño es ideal para esta aplicación.
- Programas de televisión que requieren de un escenario dinámico y original con grandes dimensiones.
- Grandes recitales que requieran de efectos modernos y con grandes potencias para impresionar al publico.
- Discotecas que usan este tipo de pantalla como uno mas de sus efectos lumínicos logrando un ambiente moderno, atractivo y original
- Indicadores de velocidad en autódromos que requieren de gran tamaño para la visualización a gran distancia.

2.2. Planteamiento del problema a resolver

Los problemas a resolver son los siguientes:

- Manejar el consumo asociado a pantallas grandes que llega a ser de centenares de amperes
- Resolver la disipación de calor asociada a semejante cantidad de lamparas que tienen un rendimiento lumínico bajo,
- Acomodar mecánicamente las lamparas dentro de una celda, de manera que cuando se coloquen varias celdas contiguas la distribución sea homogénea
- Determinar la duración máxima de los cuadros para proveer de la escala de grises requerida
- Determinar la manera en que la información de la imagen se informe a las celdas. Para eso habrá que definir si las celdas son inteligentes, distribuido o son totalmente discretas y el procesamiento es centralizado.
- Proveer un soporte para diferentes tamaños y formas de pantalla para acomodar las celdas de la manera mas flexible posible, dejando la mayor cantidad de grados de libertad de colocación posible, ampliando así el número de aplicaciones.
- Definir la lampara a utilizar.
- Definir el sistema de conexionado de cada celda para una rápida extracción y movimiento.
- Tener en cuenta la acción del sol sobre el frente de la celda ya que debería verse bien aun en ambientes exteriores con luz solar.
- Se deberá resolver el soft de manera que se adapte a todas las aplicaciones a las que se apunta.
- Resolver como implementar las escalas de grises.
- Elegir el método de transferencia de información a las tasas requeridas.
- Evitar las EMI, ya que con los valores de corriente tan altos, si no se maneja correctamente el encendido de las lamparas podría llegar ha ser muy considerable.
- Determinar la estrategia para el recambio de las lamparas.
- Determinar que tipo de protocolos se usará para poner en línea la pantalla a través de internet.

3. Definición de Producto

3.1. Requerimientos.

3.1.1. Construcción de la casa de calidad. (Análisis de valor y competitividad)

Como usuarios de este producto se pueden dividir en varios sectores, pero se concentrará la atención principalmente en los siguientes:

1. Usuarios de la pantalla para objetivos de publicidad dinámica
2. Usuarios de la pantalla para la visualización de información a distancia con poca animación (tanteadores deportivos, etc.)

Se plantea esta segmentación debido a que los requerimientos de ambos pueden ser sutilmente diferentes en algunos aspectos.

Sin embargo existen requerimientos comunes a ambos que son:

- Fácil mantenimiento
- Largo alcance de visión
- Resistente a la lluvia.
- Visible a plena luz.
- Alta potencia lumínica
- Bajo consumo

Los requerimientos específicos del segmento uno son:

- A todo color.
- Soportar un mínimo de 5 cuadros por segundo.
- Manejo individual de cada pixel.
- Acceso vía ethernet mediante algún protocolo estándar.
- Precio moderado.
- Posibilidad de generar las animaciones desde cualquier soft gráfico.

Los requerimientos del segmento dos son:

- Posibilidad de cambiar la forma de la pantalla.
- Comunicación sencilla mediante un soft dedicado para la aplicación.
- Facilidad para transmitir mensajes instantáneos de texto.
- Set de efectos pregrabados básicos.
- Bajo precio.

Si bien estos requerimientos se traducen en especificaciones técnicas del producto, se encuentra que varios requerimientos se contraponen, por ejemplo “Alta potencia lumínica Vs. Bajo consumo Vs. Alto alcance de visión” o “A todo color Vs. Precio moderado”, por lo menos con la tecnología que se encuentra al alcance en estos momentos.

Con lo cual una vez decidida la tecnología a usar (lamparas de filamento) se trataran de ponderar con mayor ímpetu aquellas a las cuales esta tecnología es mas bondadosa
Por ejemplo se ponderará fuertemente la “Alta potencia lumínica Vs, Bajo consumo” etc.

Veamos una rápida y aproximada comparación entre la tecnología de leds y la de filamento para entender los ordenes que se manejan. Lo que se muestra a continuación es una comparación por m2 considerando un componente pegado al otro sin dejar insterticios y usando leds de ultima generación, con precios a consumidor final por unidad para ambas tecnologías:

	Lumenes/W	Consumo individual en W	Precio	Cantidad/m2	Potencia consumida/ m2	Precio de componentes	Lumenes/m2
Led	50	0.5	3	100000	50000	300000	2500000
Lampara de filamento	15	25	1	800	20000	800	300000

Se puede apreciar que si bien los leds tienen claras ventajas con respecto a su potencia lumínica, el precio es muy superior. Con lo cual se puede hacer una nueva comparación igualando las potencias lumínicas por m2, de forma de dispersar los leds para usar menos por m2:

	Lumenes/W	Consumo individual en W	Precio	Cantidad/m2	Potencia consumida/ m2	Precio de componentes	Lumenes/m2
Led	50	0.5	3	12000	6000	36000	300000
Lampara de filamento	15	25	1	800	20000	800	300000

Aun en este caso el precio de los led es casi de 50 veces superior a la de las lamparas y además requieren de grandes fuentes de alimentación para abastecer las grandes corrientes en baja tensión. Gasto que en las lamparas de 220v no se requieren

3.2. Especificaciones funcionales y de diseño:

Funcionalmente el equipo constará de las siguientes propiedades:

- Alimentación de la red de 220V.
- Pixeles bien definidos.
- Homogeneidad de la pantalla evitando los efectos de las uniones de las celdas.
- Escalas de grises
- Diseño plano y de mediano espesor.
- Adecuada ventilación para disipar la temperatura de las lamparas.
- Conexión con el exterior mediante una red LA o RS232
- Software dedicado a la aplicación y posibilidad de importar animaciones de mapa de bits de cualquier software gráfico.
- Montaje versátil de las celdas para lograr las dimensiones finales que se deseen en la forma elegida.
- Soporte de lamparas de hasta 40W en los colores y formas que se deseen.
- Un ángulo de visión cercano a los 180 grados en todas las direcciones
- Buena intensidad en ambientes soleados.
- Baja emisión de EMI.
- Encendido suave de las lámparas para alargar su vida útil.

3.2.1. Especificaciones del hard

- La pantalla se alimentará de la línea de red monofásica.
- Temperatura de funcionamiento desde los 0 grados hasta los 70 grados.
- Consumo máximo igual al producto de la potencia de la lámpara por el número de lámparas.
- El número máximo de lámparas prendidas promedio deberá ser menor a ¼ del total.
- Potencia máxima de cada lámpara 40W.
- Máxima cantidad de lámparas por pantalla 6400

3.2.2. Especificación del soft (interfaces y protocolos)

- Comunicación con el exterior mediante RS232 o LA dependiendo de la aplicación.
- Las animaciones deberán cumplir con un protocolo propietario documentado

4. Análisis de Factibilidad

4.1. Factibilidad tecnológica

Se comprobó que una lámpara de filamento estándar tarda entre 2 a 10 semiciclos ciclos de red en encender y entre 5 y 30 en apagarse completamente dependiendo de la marca, potencia, modelo, etc. si se trabaja con una lámpara de 3 ciclos de encendido y 6 de apagado, se tiene que la máxima cantidad de cuadros por segundo a mostrar sin distorsión será de 1 cuadro cada 90mseg = 11 cuadros por segundo.

Se demostró que el ser humano percibe el movimiento a partir de los 12 cps, con lo cual esto no sería un impedimento, aunque la no simetría del encendido y apagado provocaría un efecto diferente que deberá evaluarse, pero hay aplicaciones en donde 5 cps es más que suficiente y se sabe que esta cifra será claramente alcanzable con esta tecnología.

Por otro lado, debido al bajo rendimiento de esta tecnología, la temperatura de cientos de lámparas encendidas podría ser insoportable hasta para la más robusta estructura, con lo cuál deberá haber una manera de limitarla. Para ello existen varios métodos:

Limitar el número máximo de lámparas encendidas promedio

Utilización esporádica de la pantalla.

Escala de grises
Las primeras dos opciones son las triviales, pero la tercera opción brinda la posibilidad no solo de limitar la temperatura de trabajo, sino también bajar el consumo, aumentar la vida útil de la lámpara, y permitir un efecto adicional más que importante.
El problema es que para obtener una escala de grises nítida en una lámpara, debido a la "rápida respuesta" de encendido de esta, y partiendo de la red eléctrica estándar, habrá que bajar la tensión promedio de la misma sin que el ripple de temperatura del filamento se haga notar. Si le sumamos la necesidad de encender las lámparas suavemente a partir del cruce por cero de la red, para evitar "golpes" de tensión que fatiguen el filamento y disminuyan la vida útil, se tienen 2 posibilidades:

- Encendido y apagado en ciclos completos
- Encendido y apagado dentro de un semiciclo.

Se comprobó que la primera opción hace notar la variación de intensidad y el efecto no es agradable, con lo cual, con estas restricciones la única alternativa será encender y apagar la lampara dentro de cada semiciclo.

Con estas hipótesis como ciertas y suponiendo un promedio de 5 niveles de grises se propone un cálculo de la tasa de información requerida para alimentar la pantalla:

64bits/cuadro * 5grises * 1/10mseg * 100celdas = 3.2Mbit/seg

Aunque la "verdadera" información es de

64bit/cuadro * 5grises * 10cps * 100= 320Kbit/seg

Este factor de incremento de 10 veces la tasa requerida se debe al problema de la escala de grises. Si en vez de esta esquema centralizado de 100 celdas de 8x8 se plantea la posibilidad de 100 celdas de 8x8 inteligentes que cada una resuelve la escala de grises y desde afuera se informa solo los 320K se propone un calculo de los requerimientos de un uP central y de un uP de celda para llevar a cabo esta operación:

Si bien 320K es la entonces la información pura que se deberá transmitir, el microprocesador central debe seleccionar la celda antes de transmitir la información respectiva y se requieren de una identificación de 100 celdas alcanzaría con 7 bits para la identificación, pero como se usa un protocolo de 8 bits se desperdiciará uno; con lo cual por cada 64bits/cuadro * 5grises se 8 bits extras.

Si se supone una transmisión asincrónica del tipo RS232 de 11 bits, 8 de datos ,uno de start, uno de stop y otro para identificar address de datos, el rendimiento será de

8/11=0.72

Además se requerirán de algunos comandos para sincronismo, funciones de configuración que podrían necesitarse durante la transmisión mas un posible tiempo de retardo entre bytes o entre tramas de bytes dado por el tiempo necesario para procesar; estos dos factores se los resume en un rendimiento de 0.9

Finalmente ponderando la información por los rendimientos de los protocolos se obtiene:

((64bit/cuadro * 5grises + 8)* 10cps * 100) / (0.72 * 0.9) = 506Kbps

Este resultado es un flujo mayor que el que soportan la mayoría de las UART estándar que alcanzan los 256Kbps, pero si en promedio no cambia la información de las 100 celdas 10cps, solo se deberá actualizar aquellas celdas que así lo requieran, disminuyendo notablemente la tasa hasta alcanzar la posible.

Si bien esta es la información que debe transmitir el procesador central a las celdas, las celdas deben plasmar esa información a las lamparas mediante una interfaz paralelo-serie que deberá implementar los niveles de grises.

Con se dijo anteriormente para evitar el parpadeo en los niveles de grises a la vez que se contempla el encendido suave de la lámpara, los tiempos se deberán ajustar dentro de un semiciclo de red, dando un flujo de 64bits * 5 cada 10ms = 32Kbps. Si se considera aceptable una aproximación de 2000 instrucciones para resolver la tarea se requerirá de una capacidad de 2000ins/10mseg/5=1MIPS que esta dentro de las velocidades normales de procesamiento para este segmento de microcontroladores hoy en día.

4.1.1. Propuestas de alternativas de diseño.

Existen varias maneras de resolver el problema de acuerdo a las variables que se crean mas importantes en el diseño.

Por ejemplo para el mismo caso analizado en el apartado anterior, 6400 pixeles, se podría realizar una matriz organizada en filas y columnas y hacer el encendido barriendo de a filas o columnas o bien individualmente. Esta solución tiene la ventaja de que se requieren solo 160 transistores para toda la pantalla contra 6400 requeridos en la solución del apartado anterior. Pero sin embargo presenta otros problemas que son:

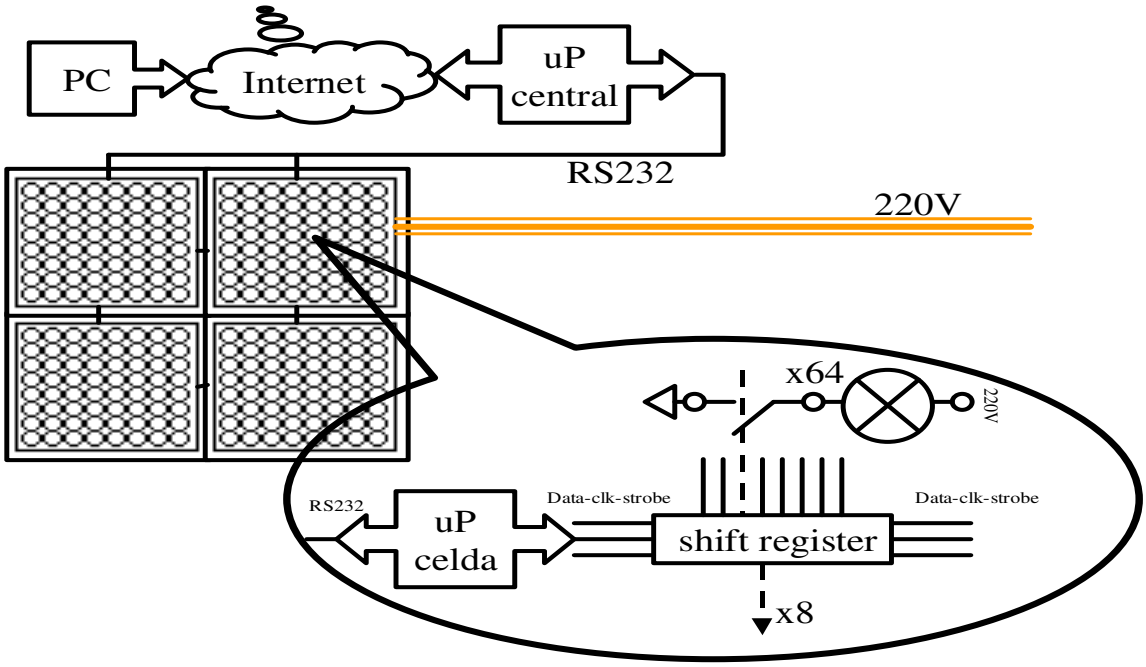
Si se realiza un barrido de las lámparas, cada lámpara podrá encenderse solo durante una fracción de tiempo, con lo cual la energía promedio entregada no será la máxima posible si se la encendiese durante todo el tiempo, esto se notará en que la lámpara nunca alcanzará el brillo para la cual fue diseñada, a la vez que cada lámpara se encenderá en un instante diferente y sería imposible el encendido suave de todas las lámparas acortando sensiblemente su vida útil. Además el EMI será también considerable.

Sin embargo algunos de estos problemas se podrían resolver si se elevara, rectificara y filtrara la tensión lo suficiente como para que la energía promedio alcance la nominal, pero esta solución agrega hardware para aumentar la tensión, a la vez que somete a la lámpara a un brusco encendido acortando más aún su vida útil.

Otra manera de resolverlo sería cambiar de arquitectura distribuida como en el apartado anterior a centralizada. Este método tiene la ventaja de que utilizar solo un procesador para todo el sistema y no requiere de complejos sistemas de sincronización, etc. pero si requiere de un elevado flujo de información que llevaría a la necesidad de un dispositivo muy veloz como un DSP o FPGA para alcanzar dicho requerimiento, aunque una vez logrado estaría acotada su escalabilidad y el canal físico de comunicación debería ser lo suficientemente robusto para evitar errores a tan alta velocidad. La velocidad a la que ser hace referencia seria de 6400 bits cada 10ms/16 = 10Mbps crudos de información, que luego debería afectarse por los rendimientos involucrados en los protocolos elegidos.

4.1.2. Elección de una solución

La solución elegida consiste en el siguiente esquema:



En primer lugar la PC será utilizada como servidor FTP y contendrá los archivos de la animaciones previamente compilados. Luego el uP central los descargará a su memoria en forma ordenada y según tiempos predefinidos.

Este archivo se enviará mediante un protocolo RS232 con capacidad para distinguir entre address y datos (9 bits). Estos datos serán recibidos y ejecutados por cada uP de cada celda.

Para la ejecución, cada celda contará con una etapa de potencia formada por 8 shift register con memoria, conectados a un puerto de 8bits del uP. Todos los shift register tendrán dos conexiones en común que son el pin de Clk y el de Strobe que a su vez se conectan a 2 pines de i/o del uP. Con esta interfaz el uP deberá shiftear los 64 bits en 8 pasos con el pin de strobe desactivado y una vez finalizado activará el strobe para trasladar la información almacenada en los flip-flops del SR a los transistores que finalmente decidirán el estado de la lámpara. Con lo cual habrá 64 transistores, 8 SR, un uP y 64 lámparas por celda.

Esta configuración aunque masiva permite el encendido individual de cada lámpara a su máxima potencia y permitiendo un encendido suave de la misma. Además como se usan transistores bipolares para el encendido de las lámparas se requiere de CC para su correcto funcionamiento, con lo cual la tensión de red se rectificará completamente antes de alimentar las lámparas. Estas placas de potencia tienen cada una un detector de cruce por cero que ingresa a uno de los pines del uP para sincronizarse con la red y así permitir el encendido suave de la lámpara.

Como se explicó en el apartado 4.1 el uP central no puede transmitir el flujo de información requerida debido a la interfaz física elegida; es por ello que el programa que convierte los paquetes de bitmap's en información lista para transmitir a las celdas será lo suficientemente inteligente para poder comprimir las imágenes de manera que el flujo de información se puede ver reducida como mínimo a un promedio de 256Kbps que es la máxima tasa de transferencia posible y será el limite de este equipo con esta interface.

4.2. Factibilidad de tiempos. Planificación

4.2.1. Planificación PERT

Detalle de las tareas: para realizar una celda:

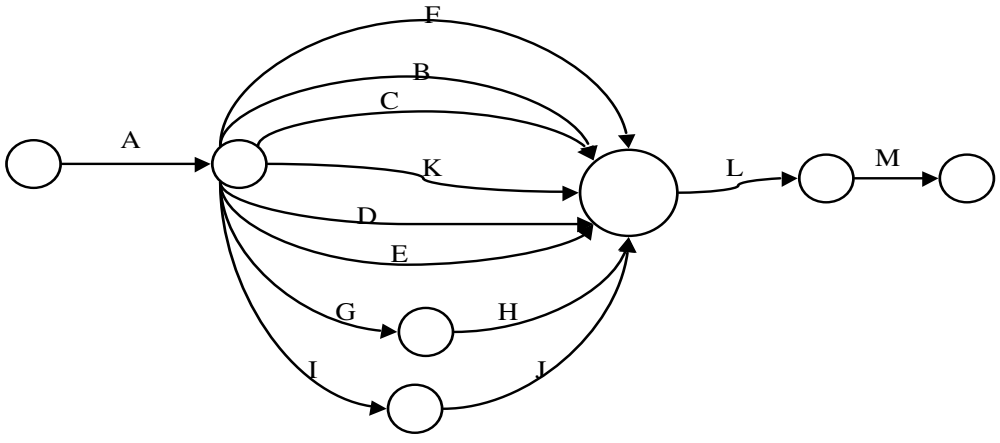
A. Planeamiento: En esta etapa se definen las especificaciones del producto. Se estudia la factibilidad del proyecto, se realiza un diagrama en bloques general del equipo de donde se desprenden las tareas a realizar en el proyecto. A cada tarea se le asigna un tiempo estimado de realización. Se realiza la planificación del trabajo con el método PERT, calculando la duración de los caminos críticos. Se hace un análisis estadístico de los tiempos y una simulación de los mismos, para confirmar los caminos críticos. Se determina el programa de trabajo utilizando el gráfico de GANTT. Se realiza un análisis de costos y de la factibilidad económica del proyecto
B. Programación del uP central: Este es el soft que se encargará de transmitir el archivo de la animación a todas las celdas esclavas. Además deberá conectarse a la PC mediante un protocolo cliente de FTP, para descargar las nuevas animaciones que vayan surgiendo de forma ordenada.
C. Programación del uP de celda: Este soft es el crítico en tiempo y será el encargado de recibir los datos enviados por el uP central y plasmarlo en la placa de potencia asociada, en sincronía con la red eléctrica para lograr un encendido suave de cada lámpara.
D. Fabricación de la celda: Las 64 lamparas de cada celda deberán ser montados en un gabinete de manera ordenada e individualizando cada lámpara para conseguir un "pixel" bien definido.
E. Diseño y armado placa potencia: Se deberá diseñar y fabricar la placa que contiene los 8 shift-registers y los transistores de potencia mas el detector de cruce por cero.
F. Diseño y armado de la placa del uP de celda: Es la placa que contiene el uP, una fuente de alimentación para el mismo y todo lo necesario para el funcionamiento del uP.
G. Elección y compra de la lámpara: Se harán pruebas sobre las diferentes marcas de lámparas disponibles en el mercado. Averiguar costos, características de encendido etc.
H. Validación del equipo. Se verifica el correcto funcionamiento del equipo y se validan las especificaciones.
I. Documentación: Se realiza el informe del proyecto conteniendo: el planeamiento, el diagrama en bloques general, el diagrama circuital completo, la lista de componentes usados, dibujo de los circuitos impresos, estimación de tiempo entre fallas y los planos mecánicos del equipo.

Planificación mediante el método de PERT:

Tarea			Toptimista	Tmedio	Tpesimista	Media	Varianza
A	Planeamiento		40	48	50	47	3
	Programación del uP central						
B		Comunicación RS232	40	43	60	45	11
C		Comunicación LAN	100	170	200	163	278
	Programación del uP de celda						
D		Driver de potencia	220	300	320	290	278
E		Comunicación RS232	40	43	60	45	11
F	Fabricación de la celda		200	210	250	215	69
G	Diseno PCB de potencia		75	80	90	81	6
H	Armado placa de potencia		20	22	25	22	1
I	Diseno PCB uP de celda		25	27	32	28	1
J	Armado de la placa del uP de celda		10	11	13	11	0
K	Eleccion de la lampara		70	80	92	80	13
L	Validación del equipo		110	160	180	155	136
M	Documentación		250	300	310	293	100

4.2.2. Simulación de Montecarlo

Para realizar la simulación de Montecarlo primero se realiza el grafo de las tareas con supuestos recursos infinitos como se muestra a continuación:



Luego se realiza la simulación de Montecarlo para 1000 muestras, en donde se resaltan las probabilidades de cada camino de ser crítico y en que tiempo puede concretarse el proyecto para 5 10 y 50% de riesgo:

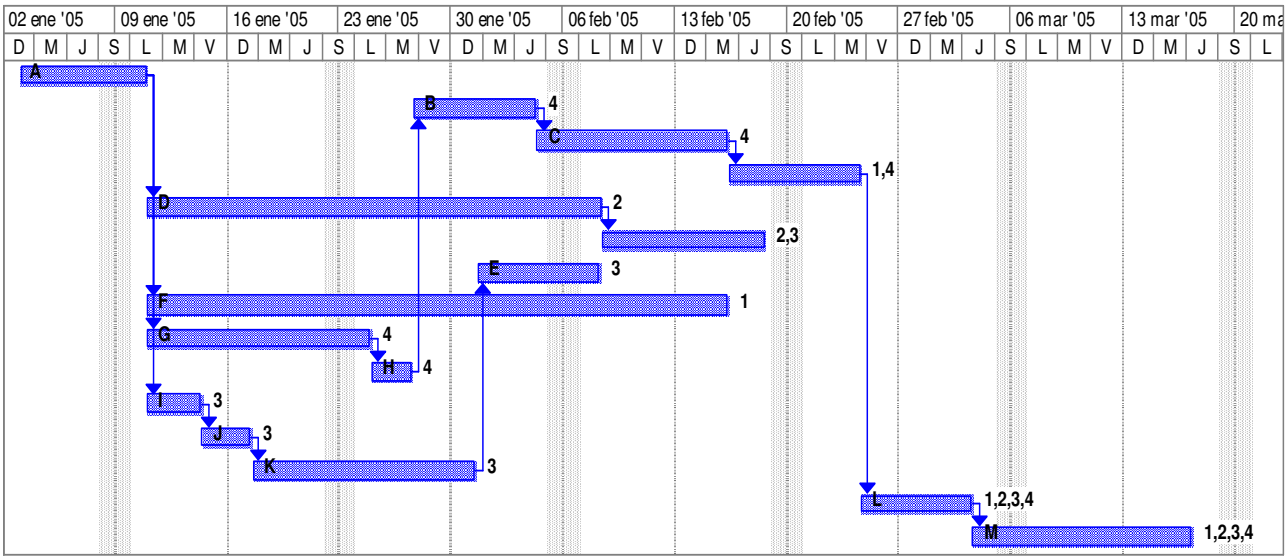
Simulacion de Montecarlo																																
Simulacion	Aleato	Tareas													Duracion de los caminos								Caminos								Duracion	Duracion reordenada
		A	B	C	D	E	F	G	H	I	J	K	L	M	C1	C2	C3	C4	C5	C6	C7	C8	C1	C2	C3	C4	C5	C6	C7	C8		
	α	4.2	1.6	3.8	1.4	1.6	4.4	2.3	2.6	2.1	2.3	2.8	3.9	2.3	AFLM	ABLM	ACLM	AKLM	ADLM	AELM	AGHLM	AIJLM										
	β	1.8	4.4	2.2	4.6	4.4	1.6	3.7	3.4	3.9	3.7	3.2	2.1	3.7																		
1	0.20	45	51	155	180	51	188	83	23	29	12	82	148	283	665	528	631	559	657	528	583	517	1	0	0	0	0	0	0	0	665	769
2	0.55	47	55	175	202	55	202	86	24	30	12	86	162	295	707	559	679	591	707	559	614	547	1	0	0	0	0	0	0	0	707	766
3	0.64	48	56	179	207	56	205	87	24	31	12	87	165	297	716	566	689	598	717	566	621	553	0	0	0	0	1	0	0	0	717	766
4	0.49	47	54	172	199	54	200	86	24	30	12	86	160	293	701	555	672	586	700	555	610	543	1	0	0	0	0	0	0	0	701	765
5	0.27	46	52	160	186	52	192	84	23	29	12	83	152	286	675	535	643	567	669	535	590	525	1	0	0	0	0	0	0	0	675	765
50	0.03	43	47	134	158	47	174	80	22	27	11	78	134	271	622	495	582	526	606	495	550	486	1	0	0	0	0	0	0	0	622	756
100	0.80	49	57	186	215	57	210	88	24	31	13	89	170	302	731	578	707	610	735	578	633	564	0	0	0	0	1	0	0	0	735	749
500	0.54	47	55	174	201	55	202	86	24	30	12	86	162	294	706	559	678	590	705	559	614	546	1	0	0	0	0	0	0	0	706	701
997	0.59	48	55	177	204	55	204	86	24	30	12	87	164	296	711	563	684	594	712	563	618	550	0	0	0	0	1	0	0	0	712	600
998	0.75	48	57	184	212	57	209	88	24	31	13	88	169	300	726	574	701	606	729	574	629	561	0	0	0	0	1	0	0	0	729	586
999	0.66	48	56	179	207	56	206	87	24	31	12	87	166	298	717	567	691	599	719	567	622	554	0	0	0	0	1	0	0	0	719	578
1000	0.24	46	52	158	183	52	190	84	23	29	12	83	150	285	671	532	638	563	664	532	587	521	1	0	0	0	0	0	0	0	671	568
Promedio															695	551	666	582	693	551	606	539	57	0	0	0	43	0	0	0	697	
Porciento de veces que aparece como camino critico																																

Del esquema se puede concluir que habrá que prestar mucha atención a los caminos C1 y C5 para que la duración total del proyecto no se vea perjudicada, ya que son los más probables de convertirse en caminos críticos.

4.2.3. Programación Gannt

Diagrama de Gantt con recursos de 4 personas asignadas de la manera mas conveniente para terminar el trabajo lo antes posible:

Nombre d	Duración	Comienzo	Fin	Predeces	Nombres de los
A	48 horas	lun 03/01/05	lun 10/01/05		
B	43 horas	jue 27/01/05	vie 04/02/05	10	4
C	65 horas	vie 04/02/05	mié 16/02/05	2	4
C1	52.5 horas	mié 16/02/05	jue 24/02/05	3	1,4
D	162 horas	mar 11/01/05	mar 08/02/05	1	2
D1	69 horas	mar 08/02/05	vie 18/02/05	5	2,3
E	43 horas	lun 31/01/05	mar 08/02/05	13	3
F	210 horas	mar 11/01/05	mié 16/02/05	1	1
G	80 horas	mar 11/01/05	lun 24/01/05	1	4
H	22 horas	mar 25/01/05	jue 27/01/05	9	4
I	27 horas	mar 11/01/05	vie 14/01/05	1	3
J	11 horas	vie 14/01/05	lun 17/01/05	11	3
K	80 horas	lun 17/01/05	lun 31/01/05	12	3
L	40 horas	jue 24/02/05	jue 03/03/05	4	1,2,3,4
M	75 horas	jue 03/03/05	jue 17/03/05	14	1,2,3,4



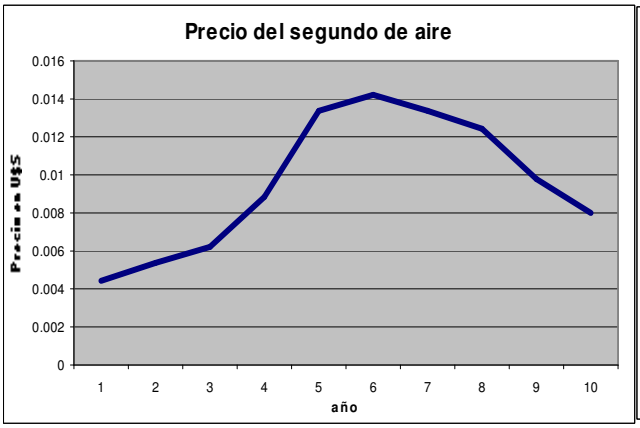
4.3. Factibilidad económica. (Mercado, costos, ciclo de vida, VAN,TIR)

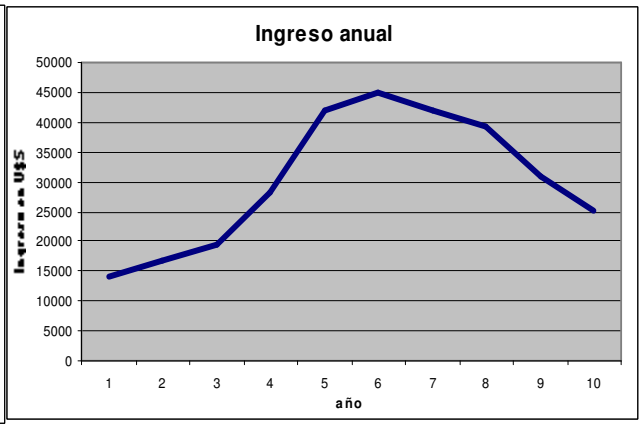
A continuación se realiza el cálculo de los gastos, teniendo en cuenta el costo de una celda y considerando una primera pantalla de 100 celdas (6400 lamp.) usada en modo servicio publicitario. Los precios que se especifican para todas las piezas, son precios de venta al publico con IVA incluído. Luego se considera que el precio puede reducirse a dos terceras partes , ya sea por importación directa o descuento por cantidad.

(Notar que todos los precios están expresados en dólares estadounidenses, debido a que la mayor parte de los componentes se cotiza en esa moneda)

Como este cálculo se específica para uso como servicio publicitario y no para la venta, se estima el ingreso por publicidad de manera creciente durante el ciclo de vida del producto como se muestra en el siguiente gráfico en dólares por segundo en pantalla. Si se considera una pantalla funcionando 8 horas diarias los 365 días del año con un 30% del tiempo en publicidades, el resto del tiempo puede usarse para noticias efectos, etc., la curva se multiplica por el siguiente factor:

3600 seg/h * 8h/día * 365 d/a * 30% pub= 3153600 y se puede apreciar la evolución el gráfico siguiente:





Calculo de la energía eléctrica:

Se considera que en promedio ¼ de la pantalla estará en promedio encendida a una potencia de 2/3 de la potencia máxima de la lámpara, que se elige en 25W, con los mismos parámetros de tiempo se obtiene una potencia promedio instantánea de:

25Watt/lamp*6400lamp*0.5*1/4= 20Kwatt

Con lo cual en 8hs se consumirá 160Kw*h, y mensualmente 4.8Mwatt*h.
Tomando como válido 0.033 dólares el Kwh se tendrá como costo variable energético:
4800Kwh*0.033= 158 U$S mensuales y 1900 U$S anuales, mas un cargo fijo anual esperado de 700U$S, dando como resultado :

2600 U$S

Mantenimiento:
Se propone inicialmente por falta de experiencia que las lamparillas serán reemplazadas en su totalidad al cabo de un año de uso, aunque se espera que este tiempo se pueda duplicar. Con lo cual se tendrá un gasto anual de 0.33/2 * 6400 = 1056 U$S

Luego se plantea un flujo de fondos estimativo para una vida útil de 8 años con valor residual cero:

Calculo del TIR											
	0	1	2	3	4	5	6	7	8	9	10
Publicidad		14016	16819	19622	28032	42048	44851	42048	39245	30835	25229
Energia electrica		-2600	-2600	-2600	-2600	-2600	-2600	-2600	-2600	-2600	-2600
Alquiler del espacio		-1500	-2000	-2500	-2500	-2500	-2500	-2000	-1600	-1000	-1000
Amortizaciones		-500	-500	-500	-500	-500	-500	-500	-500	-500	-500
Mantenimiento		-1056	-1056	-1056	-1056	-1056	-1056	-1056	-1056	-1056	-1056
Utilidad antes Imp.		8360	10663	12966	21376	35392	38195	35892	33489	25679	20073
Utilidad despues Imp.		5434	6931	8428.2	13894	23005	24827	23330	21768	16691	13047
Inversion inicial	-9580.551										
Total	-9580.551	8138	11438	14739	28614	51741	56366	53066	49501	37215	27964
VAN al 10%	$162,472.51										
TIR	128%										
Tasa de corte	20%										

En primera aproximación la TIR del proyecto es superior a la de corte y por lo tanto es rentable.

4.4. Factibilidad legal y responsabilidad civil (regulaciones y licencias)

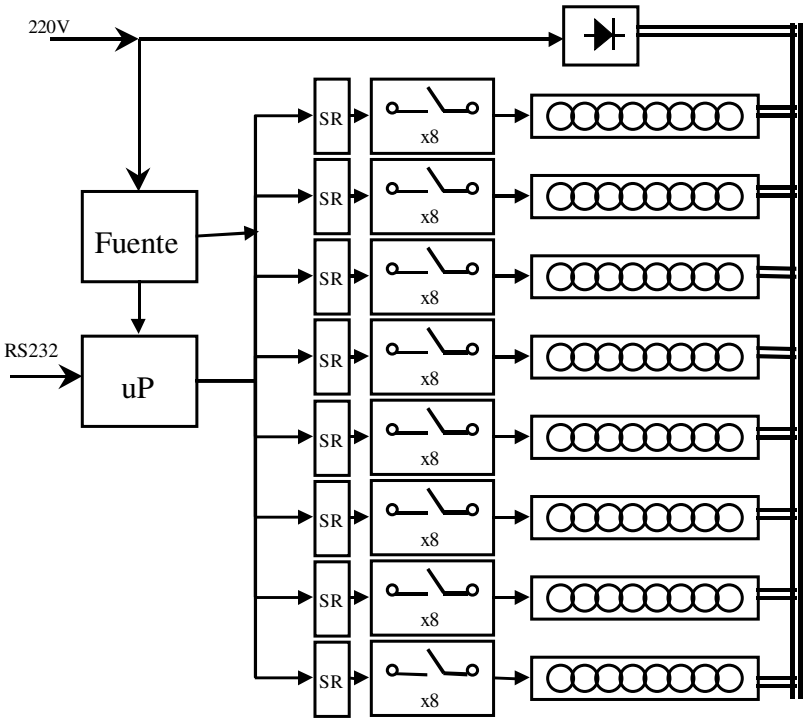
Se detallan las regulaciones del municipio de la ciudad de Concordia, E. Ríos, ya que es el lugar tentativo para la colocación del primer cartel:

5. Ingeniería de detalle

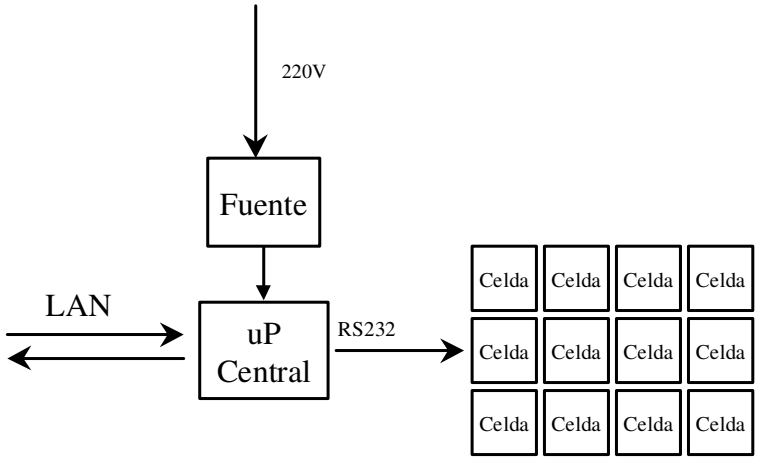
5.1. Hard

5.1.1. Diagrama de bloques

5.1.1.1. Bloques de hard de una celda



5.1.1.2. Bloques de hard del uP central



5.1.2. Descripción detallada de cada bloque

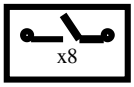
5.1.2.1. Descripción detallada de los bloques que componen una celda.

- 

Este módulo indica el conexionado de las lámparas que responden a cada SR.

- 

Este módulo es el rectificador de onda completa monofásica de la red de 220v, cuya salida alimentará a las 64 lámparas de la celda.

- 

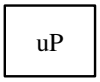
Este módulo indica el conexionado de los transistores usados para comandar las 8 lámparas a su cargo.

- 

Este módulo representa un dispositivo shift register con memoria, que permite recibir la información serial y transferirla en forma paralela en 8bits.

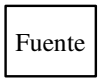
- 

Este módulo es la fuente de alimentación para el uP y la placa de potencia.

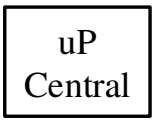
- 

Este módulo representa el microprocesador dedicado a la celda y todo lo necesario para su funcionamiento.

5.1.2.1. Descripción detallada de los bloques que componen el uP central

- 

Esta es la fuente de alimentación necesaria para alimentar el uP central

- 

Este bloque representa el uP central encargado de transmitir la información a las celdas.

5.1.3. Detalles de selección y calculo de los elementos circuitales de cada bloque

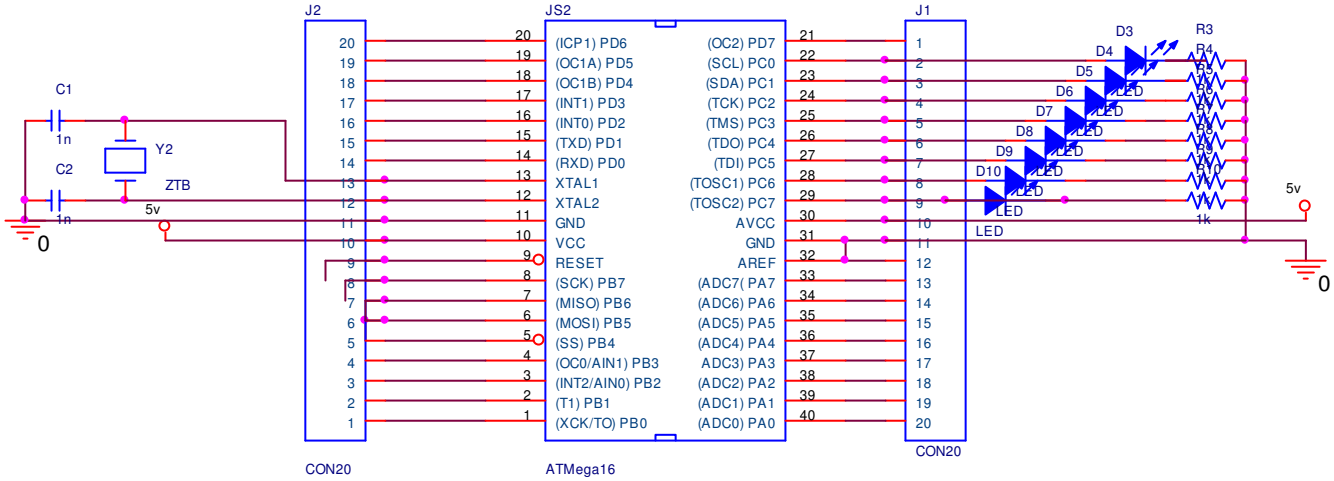
5.1.3.1. Detalles de selección y calculo de los elementos circuitales del bloque de la celda.

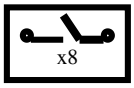
- 

Como se demostró en el apartado 4.1 este microcontrolador deberá ser capaz de procesar un mínimo de 1MIPS considerando que 2000 instrucciones fueran suficiente para el procesamiento exigido. Previendo una necesidad mas exigente en futuras versiones del equipo y considerando una programación poco eficiente en una primera etapa, se elige un uP con superiores prestaciones de velocidad y almacenamiento pero que serán aprovechados para ampliar las prestaciones del equipo. El uP elegido es entonces el AT90Mega16 de la empresa Atmel. Este dispositivo tiene una capacidad de 1k de Ram y 16K de flash con un máximo de 16MIPS y un promedio de 8MIPS aunque debido a su set de instrucciones plano se estima que equivalen a unas 4MIPS reales. Una característica importante de este componente es que posee una USART integrada con un modulo embebido que permite diferenciar tramas de address y de datos, muy útil para comunicación inter-procesadores bajando notablemente la carga del uP en procesarlo por soft.

Cuenta además con varios puertos de 8 bits de propósito general que se usarán para la transmisión de los datos a la placa de potencia. Un conversor A/D de 10bits permitiría en futuras versiones realizar mediciones de temperatura, luz ambiente y consumo, para realimentar el soft y proteger la celda de posibles fallas.

El esquemático utilizado es muy sencillo y se puede apreciar a continuación:



- 

Sin duda este fue el componente que requirió mayor esfuerzo para su correcta selección. Partiendo de la premisa de un dispositivo activo para comandar cada lámpara los requerimientos del mismo son:

Ion>=40w/220v * 1.41=0.25A pico
Vce_off>= 311v pico
Vce_on < 5v
Freq>16/10ms = 1.6Khz
Buenas propiedades para switching
Alto dv/dt.
Muy bajo precio.
Conducción en una o ambas direcciones.

Entre otras. Esta última condición implica que si el dispositivo permite la conducción en un solo sentido obliga a la rectificación de la tensión de red para su correcto uso. Es por eso que la primera búsqueda fue orientada a triacs. Pero el problema de este componente es que debido a propiedades intrínsecas de su construcción no soporta una dv/dt muy alta, con lo cual ante cualquier variación de la tensión en sus bornes mientras se encuentre apagado, provocaría un falso disparo que por su característica de apagado, permanecería encendido hasta el próximo cruce por cero, tiempo suficiente para notar el encendido de la lámpara dando por resultado encendidos aleatorios o en función de los encendidos de lámparas vecinas. Si bien existen métodos para “endurecer” este comportamiento, el precio de los apropiados son demasiado altos para la aplicación.

Descartando los triacs, la segunda opción mas interesante son los Mosfets que tienen muy buena respuesta a condiciones de enendido-apagado, muy baja Rds_on y muy alta dv/dt, y fundamentalmente los requerimientos mínimos de corriente de disparo hacen de este dispositivo ideal para poder dispararlo directamente con las salidas de los SR, evitando así gran cantidad de circuitería extra.

Si bien existen en el mercado local decenas de mosfets que cumplen estos requerimientos, el precio del modelo mas económico era aún tan elevado que influía directamente en la rentabilidad del equipo.

Por último se buscó en el segmento de los bjt’s y se encontró un transistor que encaja en la búsqueda pero debido a que el beta es demasiado bajo, no es posible dispararlo directamente con la salida de los shift-register. Con lo cual habría que usar un transistor de preamplificación para dispararlo. En ese caso se podría usar cualquier transistor corriente en modo seguidor o inversor, pero el problema de estas configuraciones es que la corriente de polarización se extrae de la fuente de alimentación que alimenta al uP y los shift register, y si se considera el peor caso de encender las 64 lámparas, con un beta promedio de 10 y una corriente de saturación de 0.25A la corriente que debería ser capaz de entregar esta fuente sería de:

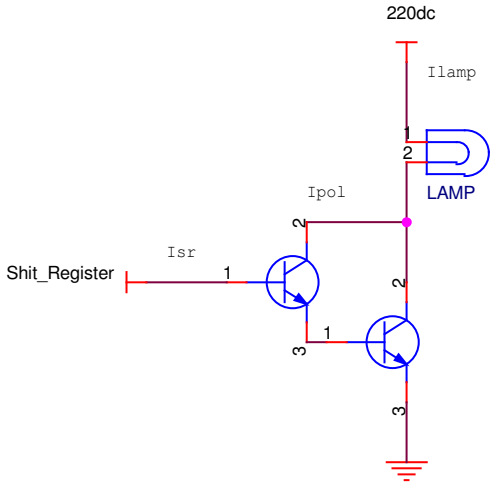
0.25*64/10=1.6A por celda, si se tienen 100 celdas son más de 150A solo para el disparo de los transistores.

Revisando los precios se encontró que el costo de los transistores de potencia elegidos son menores que el los de uso corriente (bc548,337,etc) con lo cual, eligiendo una configuración en Darlington se puede evitar tomar la corriente de polarización de la fuente de baja tensión para tomarla directamente de la red de 220 sin necesidad de circuitería externa.

Por lo tanto fue esta la topología elegida, aunque de encontrar un mejor precio para un mosfet, se cambiaría la elección debido a las ventajas de este ultimo frente a los bjt.

A continuación se muestra la topología elegida y sus parámetros:

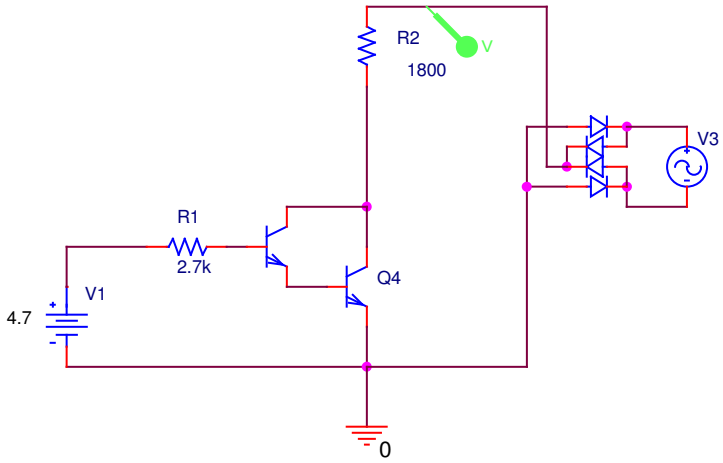
Como la tensión 220dc es en realidad 220Vac rectificada completamente, el valor RMS no se modifica, pero para los cálculos se utilizará el valor pico debido a que es la condición mas exigente

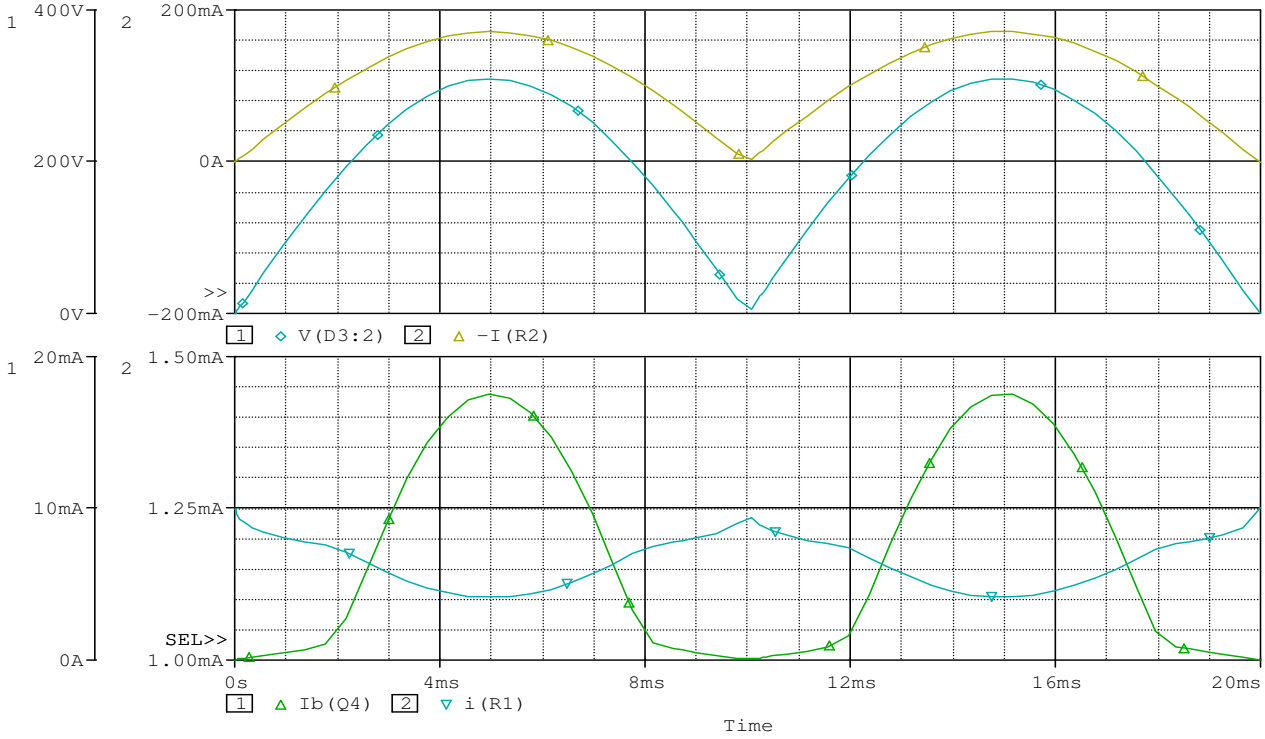


Considerando una lámpara de 40W, Ilamp=(40/220)*1.41=0.25A, si los transistores tienen un beta de 10, Ipol>=0.25/10=25mA. Luego la corriente proveniente de los shift-registers que es la que mas importa debido a que hay que prepara la circuitería para abastecerla deberá ser de Isr>=Ipol/10=Ilamp/beta^2=0.25/100=2.5mA. Bastará entonces que los SR tengan una capacidad de corriente > 2.5mA para utilizarlos directamente mediante una resistencia en el disparo del darlington bajando así en un orden la magnitud de corriente requerida = 2.6mA*64*100=16A.

A continuación se muestra una simulación del circuito en condiciones de trabajo, nominales, es decir a Tambiente, y con el transistor encendido permanentemente.

El circuito usado para la simulación es el mostrado





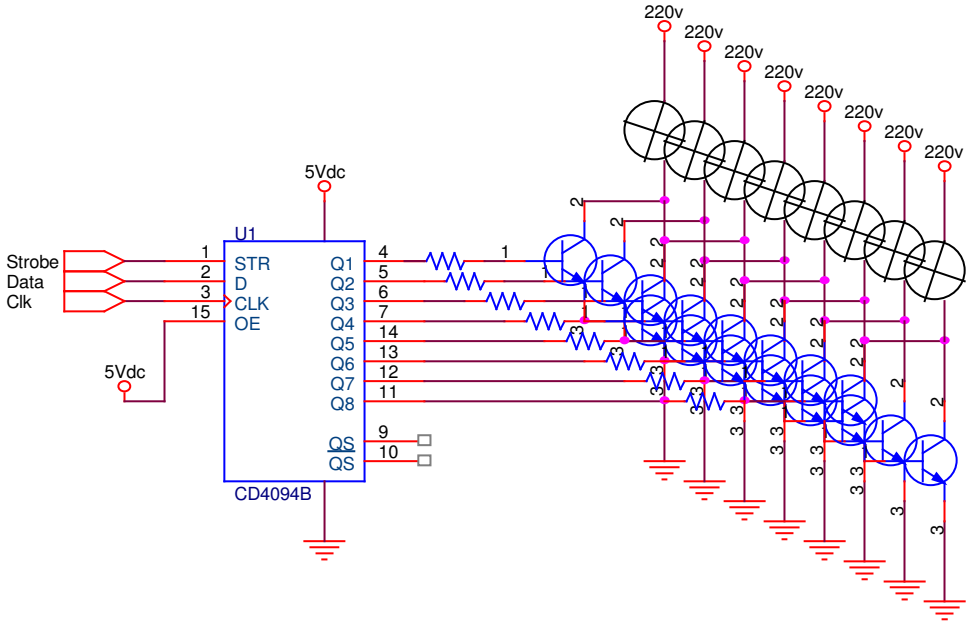
Donde:

- V(D3:2) es la tesón de red rectificada.
- I(R2) es la corriente de colector.
- Ib(Q4) es la corriente de base del transistor de salida
- I(R1) es la corriente de base del primer transistor

En la simulación se puede ver que mientras la corriente de base de Q3, es prácticamente constante e igual a 1.1 mA, la corriente de base de Q4 varia en función de la tensión de línea y se ajusta al valor necesario para autopolarizar a Q4 alcanzando valores de Ib mucho mayores que los soportados por el SR.

- SR

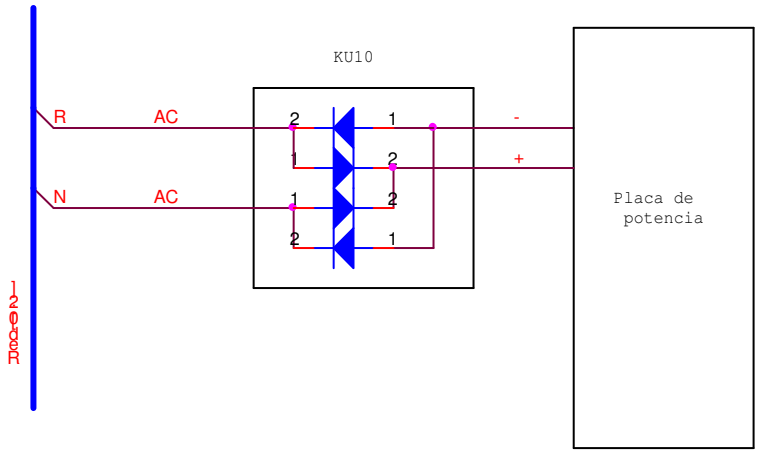
El shift register elegido es el CD4094 que posee todas las características necesarias, entrada serial ,salida paralelo de 8 bits, señal de strobe para transferir los datos almacenados a los pines de salida en el momento adecuado mientras mantiene los anteriores en memoria y salidas tri-state, que no se usa en este caso. Y cuya corriente de salida alcanza para el disparo directo de los transistores. El conexionado simplificado es el siguiente:



La resistencia de polarización del Darlington será de Vhi/2.5mA con Vhi=4.7V se tiene que R=1.8Kohm aproximadamente

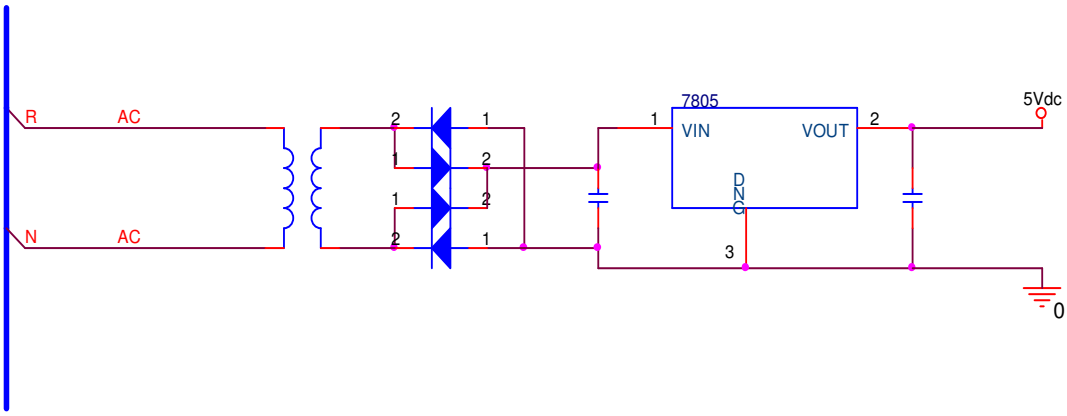
- 

Este rectificador deberá soportar tensiones inversas de mas de 311V y como se coloca uno por celda deberá soportar un corriente máxima de 64*0.25A=16A , Por precio y facilidad de montaje se eligió un puente integrado KU10. Y el conexionado es el siguiente:



- Fuente

Gracias a la disminución de la corriente de polarización proveniente de esta fuente gracias a la configuración Dalrington, este modulo se realiza con un regulador 7805 estándar de 1A de capacidad partiendo de un pequeño transformador de hierro y los puentes y filtros correspondientes, como se muestra:



- 

El conexionado de las lamparas es muy sencillo y se puede apreciar en el esquemático anterior junto con los transistores.

5.1.3.2. Detalles de selección y calculo de los elementos circuitales del bloque del uP central

- uP Central

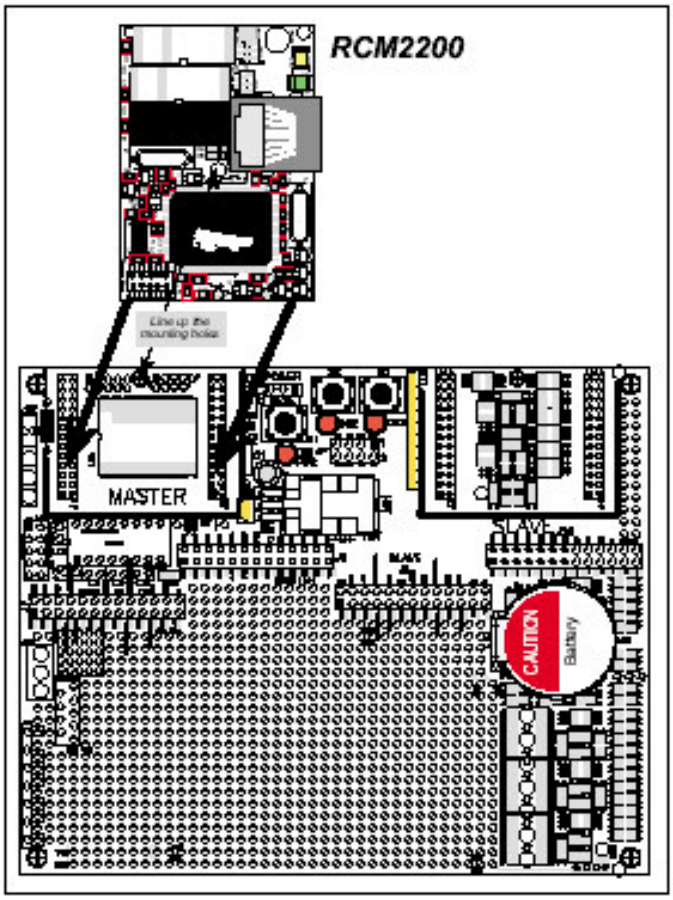
La función de este dispositivo es la de transferir la información de los cuadros a las celdas inteligentes a la vez que obtiene nueva información mediante una conexión LAN.

Se eligió para esta función un microprocesador de la familia Rabbit, exactamente el modulo Rabbit2200 que posee embebido una interfaz física LAN y todo el "open source " para soportar el protocolo IP, además tiene una amplia memoria Ram y varias interfaces RS232. Ya que estos tres requerimientos básicos son cumplidos holgadamente por una PC de uso corriente, no se descarta su uso en ocasiones que así convenga. Por ejemplo se supone la necesidad de almacenar una animación de 24hs con 100celdas a 25 cuadros por segundo en 5 niveles de grises por cuadro, se requerirá de una capacidad de almacenamiento de :

25cps*64bit/cuadro*5niveles*100celdas*3600seg/h=2.8Gbit=360Mb

Si bien esta capacidad puede lograrse integrando un dispositivo de almacenamiento del tipo flash de gran capacidad, como las usadas para los equipos portátiles modernos de fotografía, puede ser conveniente el uso de una PC estándar.

El esquemático de este uP puede verse en la sección de esquemáticos, ya que se adquirió un modulo de desarrollo que ya trae en la placa todo lo necesario para este proyecto.



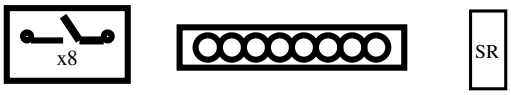
5.1.4. Plan de pruebas de cada modulo

5.1.4.1. Plan de pruebas de los módulos de la celda

Para probar este modulo deberá conectarse la celda a la red y verificar con el osciloscopio que la salida del mismo presenta una señal senoidal rectificada completamente de amplitud 311Vpico y frecuencia 50Hz.

Esta fuente regulada lineal, con lo cual bastara con una medición en la salida y verificar los 5V. Igualmente la placa posee un led indicador de alimentación que servirá para un diagnostico rápido.

La placa del uP posee un led que parpadea continuamente si el programa en el uP esta corriente normalmente, con lo cual este es un método rápido y efectivo para determinar no solo si el uP esta funcionando, sino también si el programa esta cargado y corriendo normalmente.



Estos tres módulos se deberán probar en conjunto. Ni bien la celda recibe alimentación, el software corre un pequeño programa que ayuda al diagnostico. Este pequeño algoritmo realiza el

encendido secuencial de las 64 lamparas para que el técnico determine si alguna de ellas no funciona. A partir de este programa surgen diversas situaciones que responden a determinadas fallas mas probables como ser:

Si hay solo algunas lamparas que no funcionan en ubicaciones aleatorias, habrá que revisar los transistores asociados a estas, el conexionado y las lamparas mismas.
Si falla toda una fila de lamparas, entonces es muy probable que el SR que comanda esta fila, o bien este deteriorado, o no esta recibiendo la información del uP debido a un falso contacto.
Si ninguna de las lamparas enciende, habrá que revisar la alimentación principal de la celda, y la conexión del uP a la placa de potencia en busca de un falso contacto.

uP
Central

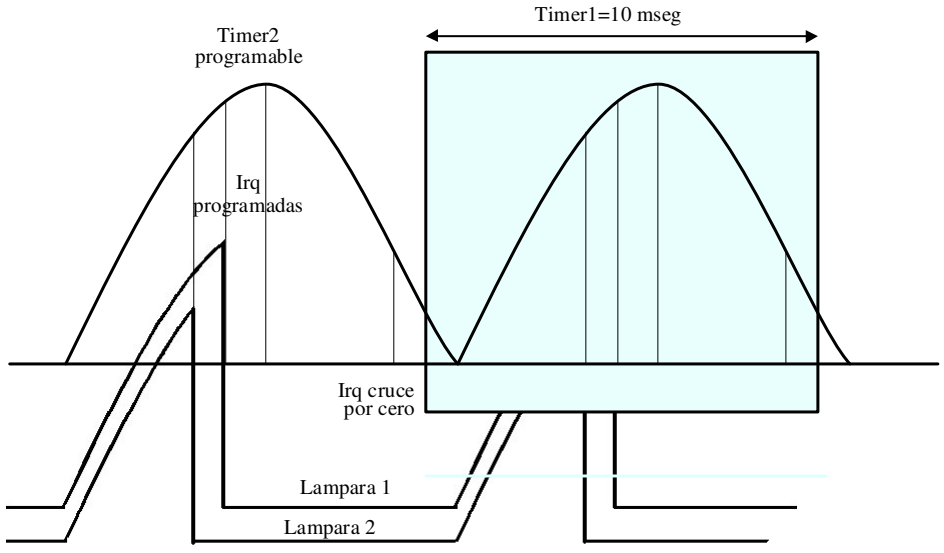
Al igual que el uP de la celda este modulo posee un led que parpadea continuamente indicando el correcto funcionamiento del soft, a la vez que otro led indica la alimentación.
También habrá que conectarlo a la PC mediante el puerto serial y verificar que este modulo esta transmitiendo datos.
Para la prueba de la interfaz LAN habrá que conectarlo a la PC mediante un Patch cruzado y mediante un "ping" verificar que el modulo responde correctamente.

5.2. Soft

5.2.1. Diagrama de estados y flujogramas

5.2.1.1. uP de la celda:

Para una comprensión rápida de las tareas involucradas en el uP de la celda se estudiara el siguiente esquema:



Se utilizara un timer del uP en modo CTC (clear timer on compare) seteando el valor máximo en 10mseg. Luego se usara la señal de cruce por cero como una irq del micro y se usara este disparo para setear el timer de manera sincrónica con la red, como se muestra con un cuadrado sombreado en la figura anterior. Como se puede apreciar el detector de cruce por cero, en realidad no se dispara en los cero volts de la tensión de red, sino que en aproximadamente 30 a 40v debido a que en el tiempo que resta hasta el cruce por cero se deberá enviar la información inicial a mostrar en el próximo cuadro a los SR, de esta manera se logra encender las lamparas justo en cero volt, de manera que los transistores se encienden con corriente cero evitando la disipación, las lamparas no sufren una variación violenta de temperatura de filamento y prolongan su vida útil y además se evita generara EMI, y ruido en la línea.

Luego de enviar el primer cuadro se configura un output compare para que “salte” en el tiempo deseado y envíe el segundo cuadro para apagar algunas lamparas, momento en el cual se vuelve a configurar y así sucesivamente hasta que llega un nuevo salto del timer de 10ms en modo CTC y la cuenta comienza de cero nuevamente. Este método permite hasta 78 saltos dentro del ciclo de 10ms, dando como resultado 78 niveles de grises. Sin embargo serán unos 20 los niveles efectivos a la vista humana y se centraran fundamentalmente en el centro del ciclo, donde debido a la mayor tensión instantánea, los efectos son mas pronunciados. Si bien el encendido se hace en el cruce por cero, no así el apagado debido a que la respuesta de la lampara utilizada es lo suficientemente rápida para que se observe un parpadeo si se utilizan ciclos completos de 10ms para el encendido y ciclos completos para el apagado.

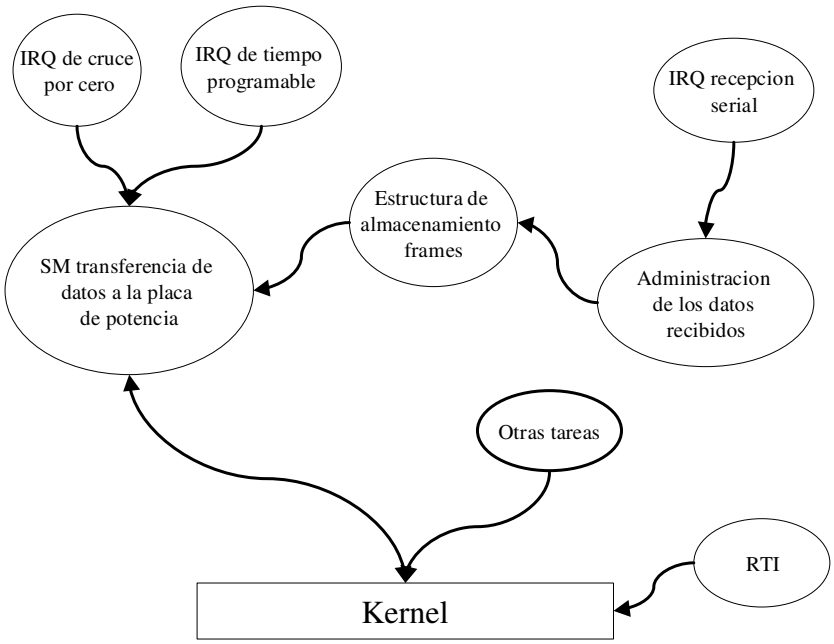
Este esquema hace notar claramente las exigencias de tiempo del uP y porque la elección de un sistema distribuido de celdas inteligentes en lugar de un solo uP centralizado, ya que aunque los cuadros efectivos de información pueden alcanzar los 12cps, los que se debe mandar para lograr una escala de grises nítida es de un máximo de 78/10mseg=7800cps y un promedio de 5/10mseg=500 por celda Luego si multiplicamos por el numero de bits y de celdas se pueden alcanzar valores tan altos como:

500cps*100celdas*64bits=3.2Mbit/seg promedio y aunque no es imposible con la tecnología actual, se prefirió evitar los problemas que presentan las altas velocidades de transmisión a distancias medias con mucha variación de corriente debido al encendido y apagado de cientos de lamparas.

Mientras se realiza esta tarea, además el uP deberá recibir la información de los futuros cuadros a través de la interface RS232.

El software utilizado para el manejo de la celda esta montado sobre un pequeño núcleo a modo de sistema operativo híbrido entre cooperativo puro y preemtivo puro, logrando realizar múltiples tareas cooperativas pero a tiempo real.

Sobre este núcleo se montan todas las maquinas de estados necesarias y tareas como se muestra a continuación:



Una de las piezas fundamentales se este software es la estructura de los frames que se define como se muestra en la siguiente figura:

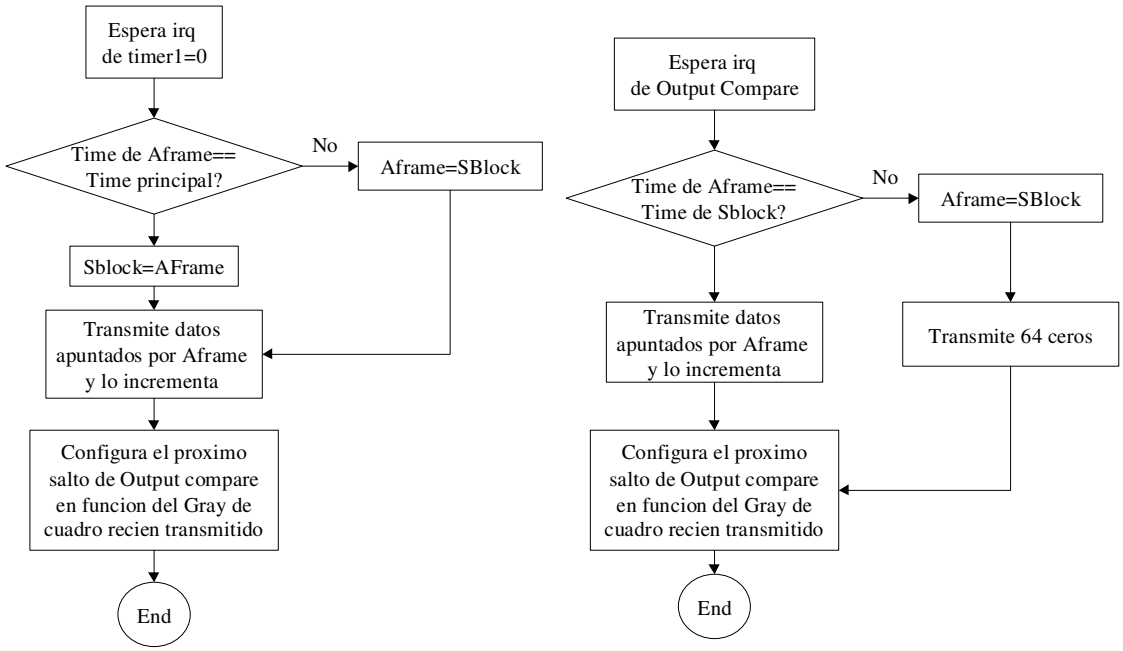
SBlock		Time	Gray	D0	D1	D2	D3	D4	D5	D6	D7	D8	
AFrame		Time	Gray	D0	D1	D2	D3	D4	D5	D6	D7	D8	Bloque1
		Time	Gray	D0	D1	D2	D3	D4	D5	D6	D7	D8	
		Time	Gray	D0	D1	D2	D3	D4	D5	D6	D7	D8	Bloque2
B2write		Time	Gray	D0	D1	D2	D3	D4	D5	D6	D7	D8	
F2Write		Time	Gray	D0	D1	D2							
	MAX												

Se trata de una FIFO circular de estructuras de 11bytes donde Time esta formado por dos bytes, un byte para Gray y ocho para los datos.

A su vez se definen cuatro apuntadores SBlock, de Start of Block; Aframe de Actual Frame, B2Write de Block to Write y F2Write de Frame to Write. La función de cada uno de estos punteros es la siguiente:

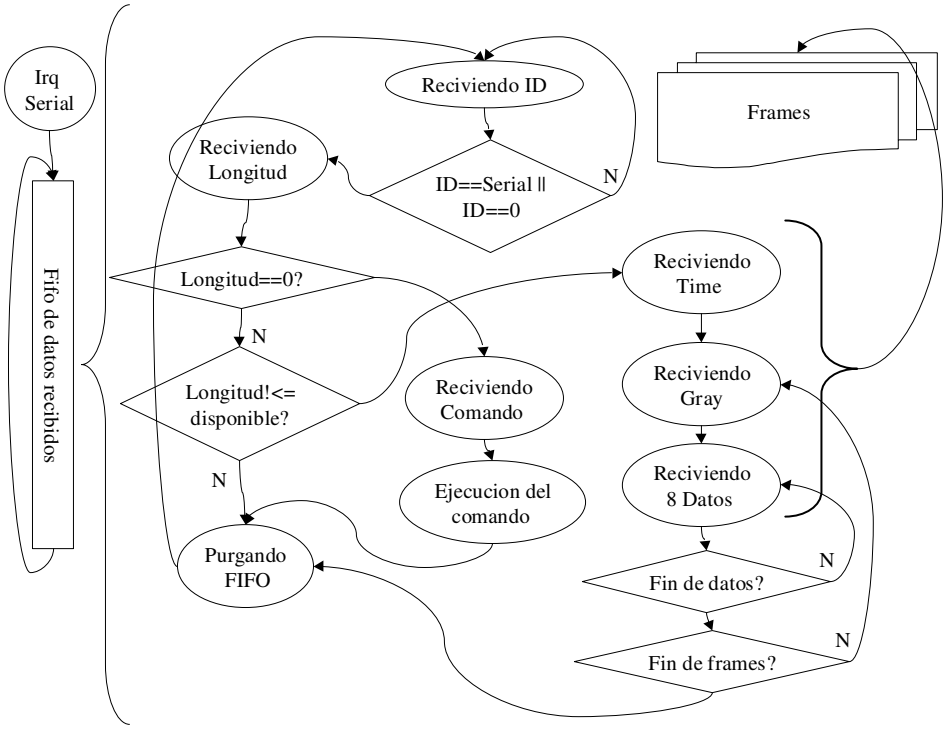
Sblock apunta al ultimo bloque valido de información y que esta actualmente en ejecución y se mantendrá en esa posición hasta que llegue un nuevo bloque. Por su lado Aframe se usa para marcar el turno del próximo frame dentro del bloque. Bwrite se usa para definir el ultimo bloque valido completo y F2Write se usa para marcar el frame que se esta recibiendo en cada momento.

La maquina de estados de transferencia de datos a la placa de potencia se puede resumir con el siguiente diagrama de flujos:



Con la implementación de estos dos esquemas, uno que responde a la interrupción de inicio de cuadro y el otro que responde a las interrupciones de output compare que se generan dentro del cuadro, se logra mantener el ultimo cuadro en pantalla con la escala de grises necesaria, mientras se testea que el próximo cuadro no corresponde al instante actual, y de ser así avanza los punteros para mostrar el nuevo cuadro a la vez que libera espacio en la FIFO circular para el ingreso de los nuevos cuadros.

Por su parte la recepción serial se hace a través de otra maquina de estados que se resume como sigue:



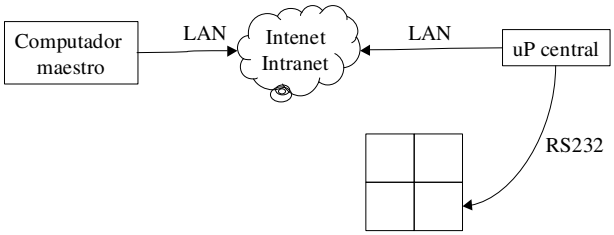
Debido a que el tipo de información a recibir es de ráfagas de pocos datos a muy alta velocidad, este esquema fue pensado para aliviar el tiempo de tareas dentro de la irq para evitar el overrun de los datos. Para ello dentro de la irq solo se realiza la tarea de encolar el dato en la FIFO de eventos y luego el kernel a su debido tiempo lo enviara a la maquina de estados para su procesamiento; además un exceso en el tiempo de la irq serial podría provocar un leve parpadeo en las lamparas debido al retardo ocasionado. No se dedica una FIFO exclusivamente para los datos recibidos por el puerto debido a que esto consumiría algo de RAM, y se prefiere dedicarla lo máximo posible a la estructura de frames, de manera que la FIFO de eventos se usa para este fin mientras que en el momento de procesar el dato se lo ubica de manera definitiva en el cuadro que corresponda. Todo el esquema corre en paralelo con un Time_Out , que ante una eventual corrupción de datos, reiniciará la maquina en un estado definido y purgara los eventos espurios.

Cuando el uP central envía la cantidad de datos tentativa a mandar a la celda, esta responde con la cantidad disponible en ese momento. Si el uP ha pedido una cantidad igual o inferior el pedido es aceptado y el uP central deberá enviar todos los datos prometidos. Si no es así, uP deberá intentarlo mas tarde. En cualquiera de los dos casos, con cada comunicación con la celda el uP toma conocimiento de la disponibilidad de espacio en cada celda y se realimenta con este valor para ajustar los tiempos entre llamadas y evitar demasiadas llamadas negadas por FIFO llena. Este mecanismo optimiza la comunicación con aquellas celdas que si están trabajando y requieren de un mayor flujo de datos a la vez que redice el trafico de datos indeseado.

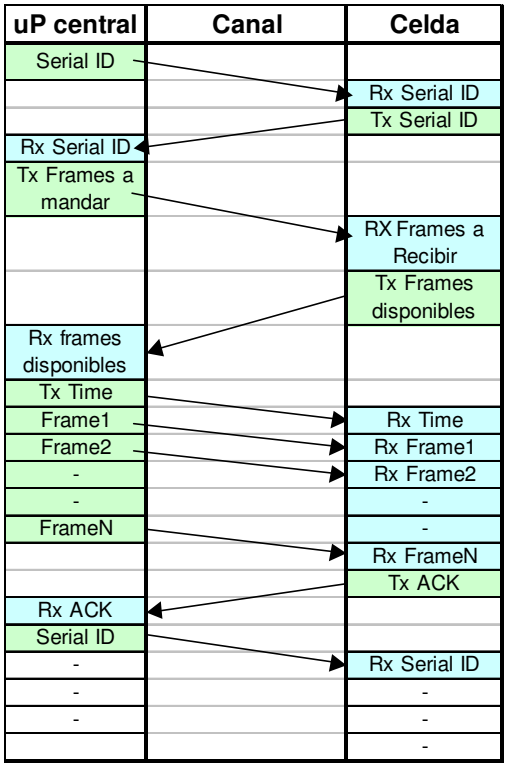
Si la longitud de datos a transmitir por el uP central es cero, entonces se parseará una serie de comandos que permiten por ejemplo la sincronización de los relojes de tiempo, entre otras cosas que se irán implementando mas adelante, como velocidad de ejecución, escalado de grises, etc. Si el ID enviado es cero significa que todos los uP esclavos recibirán los siguientes datos, de manera de que pueden realizarse tareas en paralelo a todas las celdas con mínimos recursos de bus.

5.2.1.2. uP Central

La tarea del uP central es simplemente la de almacenar el paquete de datos a transmitir, comunicarse mediante una red LAN a otro computador donde se almacenan todos las animaciones, y comunicar a todas las celdas esclavas y en forma ordenada la información de la animación en actual ejecución. Gráficamente:



Dependiendo de la aplicación la comunicación con el computador central podrá hacerse en paralelo con la transmisión de datos o separadamente. Ya que la exigencia de la animación podría requerir de todos los recursos del uP. El objetivo principal de este soft será el de mantener a todas y cada una de las FIFO's de las celdas totalmente llenas; con eso se consigue evitar posibles congelamientos de la imagen debido a falta de datos a procesar. El esquema de comunicación diseñado es el siguiente:



Notar que este es un esquema simplificado en donde faltan las situaciones de FIFO llena, comandos, posibles errores, etc. sin embargo este seria el mecanismo normal de comunicación y en base a este se calculara la tasa de información mínima requerida por el uP central como sigue:

Se considerara un promedio de 5 escalas de grises diferentes por cada cuadro a 12 cuadros por segundo y 100 celdas:

uP_ID(1)+Celda_ID(1)+uP_Length(1)+Celda_Length(1)+Time(2)+5*Frames(9)+Ack(1)=50bytes

Estos 50 bytes se deberán enviar 12 veces por segundo dando un total de :

50 * 12 = 600 bytes por seg por celda

600b/seg*celda * 100celdas * 8bit/b = 480Kbit/seg

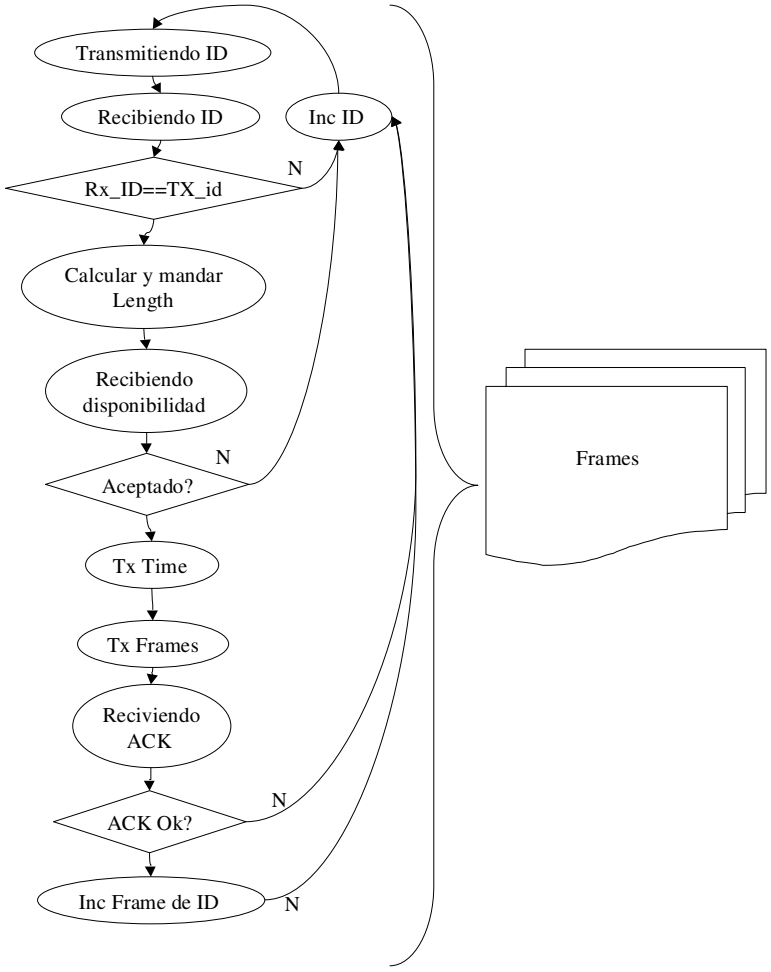
Que comparado con el método centralizado que son :

64bit/celda*5grises*1/10mseg*100celdas=3.2Mbit/seg

Esto arroja una reducción del trafico en 3200Kbit/seg / 480Kbit/seg = 6.6 veces

Pero además, con este sistema distribuido no siempre se necesitaran actualizar las 100 celdas 12 veces por segundo, ya que algunas de ellas podrían permanecer con la misma información incluso por minutos. En promedio se estima que solo un cuarto de las celdas se deberán actualizar 12 cps, con lo cual esta

estimación reduciría el trafico a 480Kbit/seg / 4 = 120 Kbit/seg, que es apenas superior a la tasa máxima alcanzable por la UART del uP central.
Con estos requerimientos en mente se diseño el siguiente esquema de soft para el uP central:



Por simplicidad este es un esquema simplificado en donde se omiten situaciones de error de datos, transmisión de comandos, time_out's por falta de respuesta por parte de la celda, y la realimentación en función de la disponibilidad de espacio en cada celda entre otros detalles.

Notar que este diagrama de estados es único pero no así las variables de estados ya que hay una por cada celda, de manera que cada variable de estado puede estar en un estado diferente en cada instante, con lo cual se logra un único código para las N celdas, que de lo contrario comprometería el espacio de almacenamiento del código en el uP central. Sin embargo como todas las maquinas de estado requieren compartir el uso del puerto serie, la maquina N-1 deberá esperar a que la maquina N finalice el uso del puerto para tomar el control. Pero se mantiene esta filosofía debido a que en una versión futura se podría implementar la transmisión de datos a través de 2 o mas puertos series en forma paralela, logrando aumentar en numero de CPS máxima. Esto es posible con el uP central elegido ya que soporta hasta 3 UART's independientes, así como con cualquier PC de uso corriente.

5.2.2. Análisis de complejidad

5.2.2.1. Complejidad de Hard

5.2.2.1.1. Complejidad de la placa del uP de celda

No se realizan estudios.

5.2.2.1.2. Complejidad de la placa de potencia

No se realizan estudios.

5.2.2.1.3. Complejidad de la placa del uP central

No se realizan estudios.

5.2.2.2. Complejidad de Soft

5.2.2.2.1. Complejidad del soft del uP de celda

Para analizar la complejidad de las funciones involucradas en el proyecto, se utilizara el método de McCabe, este método evalúa la complejidad en base a la estructura del software, y determina la cantidad de decisiones o caminos posibles que tiene el grafo que representa a la función a analizar.

La medición de la complejidad puede ayudar a concentrar esfuerzos durante la depuración sobre aquellos módulos o funciones mas susceptibles de poseer errores, mas complejos.

Como regla general, es deseable que el numero resultante de la complejidad de un modulo se mantenga menor a 10, ya que pasado ese limite, la probabilidad que ese modulo contenga errores se incrementa abruptamente.

Para disminuir la complejidad de una función, es necesario construir otra función que realice parte del proceso de la primera, de manera de simplificarla.

Bajo la hipótesis que la programación es estructurada , para cada función se realiza un grafo, donde el numero de áreas mas uno, es el indicador de la complejidad. Se realiza esto para cada función y aquellas que tengan una complejidad mayor a 10 deberán ser reescritas en partes menos complejas.

A continuación se muestra la tabla de las complejidades de todas las funciones involucradas:

Funcion	Complejidad
Init_Circ_FIFO	1
Write_Circ_FIFO	2
Read_Circ_FIFO	2
Set_Strobe	1
Clr_Strobe	1
Set_Clk	1
Clr_Clk	1
Secuence	2
Secuence_Zero	2
Init_Frame_Counter_And_Zero_Cross_Capture	1
Output_Compare	1
Input_Cap	1
Output_CompareB	1
Init_Frames	1
Init_Frames_State_Machine	1
Inc_Block_Time_Counter	2
Is_Snap_Shot_Time_Block_Time	3
Is_Snap_Shot_Time_Snap_Shot_Block_Time	1
New_Frame_Func	2
Gray_Scale_Func	2
Init_Data_Circ_FIFO	1
Write_Time	1
Write_Gray_Level	1
Write_Data	1
Read_Snap_Shot_Data	1
Inc_Frame2Write	1
Inc_Snap_Shot_Frame	1
Read_Gray_Level	1
Read_Snap_Shot_Time	1
Read_Snap_Shot_Block_Time	1
Is_New_Frame	1
Snap_Shot_Block2Snap_Shot_Frame	1
Snap_Shot_Frame2Snap_Shot_Block	1
Block2Write2Frame2Write	1
Frame2Write2Block2Write	1
Available_Blocks	2
main	1
USART_Init	1
Serial_Service	1
Receive_Central_Time_Hi	1
Receive_Central_Time_Lo	1
Is_New_Central_Time	1
Read_New_Central_Time	1
Answer_ID	1
Check_If_Available	2
Receive_Time_Hi	1
Receive_Time_Lo	1
Receive_Gray_Level	1
Receive8Data	3
Invalid_Data	1
Init_Serial	1
Maximo	3
Promedio	1.3

La tabla muestra que el máximo de complejidad es de 3 pero con un promedio de 1.3; esto quiere decir que la complejidad es muy baja, dado que el mínimo es uno.

5.2.2.2.2. Complejidad del soft del uP central

No se realizan estudios.

5.2.3. Descripción de subrutinas

5.2.3.1. Descripción de las rutinas utilizadas en el uP de celda

La mayoría de las rutinas fueron escritas en C, solo algunas están en ASM, y son solo para la inicialización del stack pointer y el salto inicial al main, ya que no se utiliza las bibliotecas estándar de C ni el cstartup, provisto por el compilador por una cuestión de optimización de Ram. La descripción se divide en módulos en donde se organizan funciones que están relacionadas con alguna tarea:

- Cola circular de eventos

Este modulo se dedica a mantener una cola circular de bytes que será usado por el kernel para levantar los eventos ingresados por las distintas aplicaciones

- Next_Pointer(Actual)

Esta es una macro que recibe la posición actual dentro de la cola y devuelve la próxima posición pudiendo se esta simplemente la siguiente numéricamente o cero si llego al limite de capacidad

- void Init_Circ_FIFO(void)

Inicializa la cola circular colocando los punteros en las posiciones correspondientes

- unsigned char Read_Circ_FIFO(void)

Lee el próximo dato de la cola, y de no haber ninguno devuelve cero.

- void Write_Circ_FIFO(unsigned char Data)

Escribe un nuevo dato a la cola, y de no haber mas capacidad, no lo hace sin devolver ninguna condición de error, es decir que el usuario deberá saber de antemano si queda espacio para evitar un encolamiento falso.

- Data Clock

En este modulo se encuentran las funciones que se comunicación con la placa de potencia para transferir los datos.

- static void Set_Strobe(void)

Esta función simplemente pone en uno el pin del micro que comanda todos los strobes de los shift register's.

- static void Clr_Strobe(void)

Esta función simplemente pone en cero el pin del micro que comanda todos los strobes de los shift register's.

- static void Set_Clk(void)

Esta función simplemente pone en uno el pin del micro que comanda todos los clock's de los shift register's.

- static void Clr_Clk(void)

Esta función simplemente pone en cero el pin del micro que comanda todos los clock's de los shift register's.

- void Secuence(void)

Esta función realiza el shifteado de los 64 bits de información próximos a mandar a los shift registers, que se encuentran en la estructura de frames.

- void Secuence_Zero(void)

Esta función realiza el shifteado de 64 ceros, para apagar a todas las lamparas.

- Modulo Frames

Modulo dedicado a atender las interrupciones de sincronismo con la red y los dos timers usados para las escalas de grises.

- static void Init_Frame_Counter_And_Zero_Cross_Capture(void)

Inicializa todos los registros y variables necesarias para el manejo de los timers y el input capture. Se ejecuta solo una vez al inicio del programa.

- __interrupt void Output_Compare(void)

Interrupción del output compare usada para dar inicio a un nuevo cuadro en forma sincrónica con la red

- __interrupt void Input_Cap(void)

Interrupción de input capture dedicada a sincronizar un timer con la red eléctrica.

- __interrupt void Output_CompareB(void)

Interrupción de output compare2 usada para generar la escala de grises.

- void Init_Frames_State_Machine(void)

Inicializa la maquina de estados de los frames.

- void Inc_Block_Time_Counter(void)

Incrementa el cuadro actual controlando de que no haya una sincronización pendiente, de ser así se asigna el valor de sincronización del master clock.

- unsigned char Is_Snap_Shot_Time_Block_Time(void)

Devuelve uno si el tiempo actual es igual al tiempo del snapshot frame

- unsigned char Is_Snap_Shot_Time_Snap_Shot_Block_Time(void)

Devuelve uno si el tiempo de snapshot es igual al tiempo de bloque.

- void New_Frame_Func(void)

Esta función decide si avanzar al próximo frame o ejecutar el frame actual. Se ejecuta en respuesta al evento generado por la irq de output compara 1.

- void Gray_Scale_Func(void)

Se ejecuta en respuesta al evento de irq de output compare 2 y ejecuta el frame actual

- Modulo frames struct

Este modulo se dedica a mantener una FIFO circular de estructuras de eventos, aportando todas las funciones necesarias para manejarla.

- void Init_Data_Circ_FIFO(void)

Inicializa todos los punteros asociados.

- void Write_Time(unsigned int Time)

Escribe el parámetro en la estructura apuntado por el puntero al próximo frame a escribir

- void Write_Gray_Level(unsigned char Level)

Escribe el parámetro en la estructura apuntada por el próximo frame a escribir

- void Write_Data(unsigned char Index,unsigned char Data)

Escribe el parámetro en la estructura apuntada por el próximo puntero a escribir indexada con el parámetro index.

- unsigned char Read_Snap_Shot_Data(unsigned char Index)

Devuelve el dato de la estructura apuntada por el puntero snapshot en la posición Index.

- void Inc_Frame2Write(void)

Incrementa el puntero a escritura.

- void Inc_Snap_Shot_Frame(void)

Incrementa el puntero a snapshot.

- unsigned char Read_Gray_Level(void)

Devuelve la escala de gris apuntada por snapshot.

- unsigned int Read_Snap_Shot_Time(void)
  Devuelve el tiempo de la estructura apuntada por snapshot.

- unsigned int Read_Snap_Shot_Block_Time(void)
  Devuelve el tiempo de la estructura apuntada por el snapshotblock

- unsigned char Is_New_Frame(void)
  Devuelve uno si el puntero a snapshot es igual al puntero a snapshotblock

- void Snap_Shot_Block2Snap_Shot_Frame(void)
  Asigna a snapshotblock el valor de snapshotframe

- void Snap_Shot_Frame2Snap_Shot_Block(void)
  Asigna el valor de snapshotframe a snapshotblock.

- void Block2Write2Frame2Write(void)
  Asigna el valor de block2write a frame2write.

- void Frame2Write2Block2Write(void)
  Asigna el valor de frame2write a block2write

- unsigned char Available_Blocks(void)
  Devuelve el numero de frames disponibles para escribir.

  - Modulo Serial

Este modulo integra todas las funciones necesarias para el manejo de la UART del uP.

- void USART_Init(void)
  Inicializa la UART seteando los registros correspondientes.

- __interrupt void Serial_Service(void)

Esta interrupción se ejecutara cuando se haya completado la recepción del byte transmitido por el master.

- void Receive_Central_Time_Hi(void)

Guarda temporalmente la parte alta del master time transmitido

- void Receive_Central_Time_Lo(void)

Guarda temporalmente la parte baja del master time transmitido a la vez que prende un flag indicando que esta listo un nuevo tiempo master para sincronismo.

- unsigned char Is_New_Central_Time(void)
  Devuelve uno si el flag de nuevo central time es uno.

- unsigned int Read_New_Central_Time(void)

Devuelve el central time temporalmente guardado.

- void Answer_ID(void)
  Transmite el master el propio ID

- void Check_If_Available(void)
  Controla que haya suficiente espacio para almacenar la información requerida por el master.

- void Receive_Time_Hi(void)
  Recibe del master el byte mas significativo del parámetro tiempo del frame en cuestión

- void Receive_Time_Lo(void)

Recibe del master el byte menos significativo del parámetro tiempo del frame en cuestión

- void Receive_Gray_Level(void)

Recibe el nivel de gris del frame en cuestión.

- void Receive8Data(void)
  Recibe los 8 bytes de datos crudos del frame en cuestión.

- void Invalid_Data(void)
  Envía al master un mensaje de negación debido a algún error.

5.2.4. Listados comentados del código

5.2.4.1. Listado del código del uP de celda

- Modulo de cola circular de eventos

```
//archivo de implementación de una cola circular
#include <Circ_FIFO.h>
#include <iom16.h>

__no_init static unsigned char Buffer[MAX_CIRC_FIFO_LENGTH+1];          //array usado de buffer
para almacenar los datos...
__no_init static unsigned char Point2Write;                             //puntero para escritura...
__no_init static unsigned char Point2Read;                              //puntero para lectura...

#define Next_Pointer(Actual) ((Actual == MAX_CIRC_FIFO_LENGTH)?0x00:Actual+1)          //macro
que calcula la posición de la próxima posición en la cola.

void Init_Circ_FIFO(void)                       //Inicializa la cola circular...
{
 Point2Read = Point2Write = 0x00;               //se inicializan los punteros...
 return;
}

void Write_Circ_FIFO(unsigned char Data) //escribe un nuevo dato a la cola...
{
 if (Next_Pointer(Point2Write) != Point2Read)          //si el siguiente casillero no esta apuntado por
Point2Read...
   {
    Buffer[Point2Write] = Data;                 //almacena el dato...
    Point2Write = Next_Pointer(Point2Write);    //apunta al siguiente...
   }
 return;
}


unsigned char Read_Circ_FIFO(void)              //lee el próximo dato de la cola...
{
 unsigned char Ans = 0x00;
 if (Point2Read != Point2Write)                 //si los apuntadores son diferentes, es porque hay
algo para leer...
   {
Ans=Buffer[Point2Read];                         //auxiliar para guardar la respuesta, que por default se graba
la data de la posición actual...
    Point2Read = Next_Pointer(Point2Read);//se incrementa el puntero...
   }

return Ans;                                     //notar que si no hay nada para leer, devuelve
siempre el ultimo dato escrito...
}
```

- Modulo de data clock

//archivo dedicado a la transmisión de los datos a la placa de potencia

```
#include <iom16.h>
#include <inavr.h>
#include <data_clk.h>
#include <frames_struct.h>


static __flash unsigned char Info[]={


0,      0,      20,     0,      0,      0,      0,      0,      62,     34,     62, //0
25,     0,      21,     0,      0,      0,      0,      0,      0,      62,     4, //1
50,     0,      22,     0,      0,      0,      0,      0,      38,     42,     54, //2
75,     0,      23,     0,      0,      0,      0,      0,      62,     42,     42, //3
100,    0,24,   0,      0,      0,      0,      0,      62,     8,      14, //4
125,    0,      25,     0,      0,      0,      0,      0,      58,     42,     46, //5
150,    0,      26,     0,      0,      0,      0,      0,      58,     42,     62, //6
175,    0,      27,     0,      0,      0,      0,      0,      62,     2,      6, //7
200,    0,      28,     0,      0,      0,      0,      0,      62,     42,     62, //8
225,    0,      29,     0,      0,      0,      0,      0,      62,     42,     46, //9
250,    0,      28,     0,      0,      0,      0,      0,      62,     42,     62, //8
19,     1,      27,     0,      0,      0,      0,      0,      62,     2,      6, //7
44,     1,      26,     0,      0,      0,      0,      0,      58,     42,     62, //6
69,     1,      25,     0,      0,      0,      0,      0,      58,     42,     46, //5
94,     1,      24,     0,      0,      0,      0,      0,      62,     8,      14, //4
119,    1,      23,     0,      0,      0,      0,      0,      62,     42,     42, //3
144,    1,      22,     0,      0,      0,      0,      0,      38,     42,     54, //2
169,    1,      21,     0,      0,      0,      0,      0,      0,      62,     4, //1
194,    1,      20,     0,      0,      0,      0,      0,      62,     34,     62, //0

};

void Init_Parser(void)
{
 unsigned int i,j;
 for(i=0;i<209;)
  {
   Write_Time(*(unsigned int __flash*) (((unsigned char __flash*)&Info)+i));
   i+=2;
   Write_Gray_Level(Info[i]);
   i++;
   for(j=0;j<8;j++)
    Write_Data(j,Info[i++]);
   Inc_Frame2Write();
  }
 Block2Write2Frame2Write();
 DDRD=(1<<PORTD4)|(1<<PORTD5);
 return;
}

static void Set_Strobe(void)
{
 PORTD|=(1<<PORTD4);                    //enciende el pin dedicado al strobe...
 return;
}
static void Clr_Strobe(void)
```

```
{
 PORTD&=~(1<<PORTD4);                    //Apaga el pin dedicado al Strobe...
 return;
}

static void Set_Clk(void)
{
 PORTD|=(1<<PORTD5);                     //enciende el pin dedicado al clk...
 return;
}
static void Clr_Clk(void)
{
 PORTD&=~(1<<PORTD5);                    //Apaga el pin dedicado al clk...
 return;
}


void Secuence(void)                      //Envía los datos a la placa de potencia...
{
 unsigned char Index=0;                  //define una automática para indexar y contar hasta 8...
 Clr_Strobe();                           //antes que nada apaga el strobe para que no se vea en
pantalla el shifteo de los datos...
 OCR1B=Read_Gray_Level()<<8;                     //setea el salto para el próximo output compara...
 do
  {
   Clr_Clk();                            //apaga el clk...
   PORTC=Read_Snap_Shot_Data(Index++);           //pone el dato siguiente
   Set_Clk();                            //prende el clk y los datos son shifteados...
  }
 while (Index&0x07);                     //hasta que supere 7...
 Set_Strobe();                           //setea strobe y los datos pasan a las lamparas...
 Inc_Snap_Shot_Frame();                  //incrementa el frame actual para que en la próxima entrada
se ejecute el siguiente...
 return;
}


void Secuence_Zero(void)                 //Envía ceros a la placa de potencia...
{
 unsigned char Index=0;                  //define una automática para indexar y contar hasta 8...
 Clr_Strobe();                           //antes que nada apaga el strobe para que no se vea en
pantalla el shifteo de los datos...
 do
  {
   Clr_Clk();                            //apaga el clk...
   PORTC=0;                              //pone cero en el puerto...
   Set_Clk();                            //prende el clk y los datos son shifteados...
   Index++;
  }
 while (Index&0x07);                     //hasta que supere 7...
 Set_Strobe();                           //setea strobe y los datos pasan a las lamparas...
 return;
}
```

- Modulo de frames

```
//archivo dedicado a los frames

#include <iom16.h>
```

```
#include <inavr.h>
#include <frames.h>
#include <circ_fifo.h>
#include <state_machine.h>
#include <loosing_time.h>
#include <serial.h>
#include <frames_struct.h>
#include <Data_Clk.h>


static State __flash  Initializing_Frames[1],
            Free_State1[7],
            Free_State2[4],
            Free_State3[7],
            Free_State4[4],
            Free_State5[1],
            Free_State6[1],
            Free_State7[1];

__no_init State __flash * __tiny Frames_State_Machine;
__no_init static unsigned int Block_Time_Counter;

static void Inc_Block_Time_Counter(void);
static void New_Frame_Func(void);
static void Atomic_Send_Message2Frame_State_Machine(unsigned char Event);




static void Init_Frame_Counter_And_Zero_Cross_Capture(void)
{
 Block_Time_Counter=0xFFFF;              //resetea el contador de frames.

 TCCR1B|=(1<<CS11)|(1<<WGM12) ;                  //setea el output compare en modo CTC y el input
capture en falling...
 OCR1A=20000;                            //este es el valor limite de comparación con TCNT1 cuando
alcance este valor vuelve a cero TCNT1 y empieza de nuevo
 OCR1B=20001;                                    //setea el output compara B mas adelante que el A
para que inicialmente no salte nunca...
 TCNT1=0;                                        //resetea el timer para que empiece desde cero...
 TIFR=0xFF;                                      //elimina cualquier flag de irq pendiente que haya
quedado...
 TIMSK |= (1<<TICIE1)|(1<<OCIE1A)|(1<<OCIE1B);  //enciende la irq de input y output
 return;
}

#pragma vector=TIMER1_COMPA_vect-4
__interrupt void Output_Compare(void)
{
 Inc_Block_Time_Counter();                                  //incrementa el tiempo local...
 Send_Message2Frame_State_Machine(New_Frame_Event); //envía un evento de que hay un nuevo
frame...
 return;
}

#pragma vector=TIMER1_CAPT_vect-4
__interrupt void Input_Cap(void)
{
 TCNT1=500;                              //setea el contador en 500 para sincronizarlo con la red...
 return;
}
```

```
#pragma vector=TIMER1_COMPB_vect-4
__interrupt void Output_CompareB(void)
{
 Send_Message2Frame_State_Machine(Gray_Scale_Event);  //envía un evento de que hay un nuevo nivel de gris...
 return;
 }

//-------------------------------------------------------------
static void Init_Frames(void)          //inicializa los puertos que se usan en esta maquina de estados de propósitos múltiples...
{

 return;
}

void Init_Frames_State_Machine(void)
{
 Frames_State_Machine=Initializing_Frames;          //inicializa la maquina de estados de propósitos múltiples en el estado de inicialización ...
 Init_Frame_Counter_And_Zero_Cross_Capture();   //inicializa las irq's
 return;
}


//-------------------------------------------------------------------

static void Show_Free_State1(void)
{UDR='1';return;}
static void Show_Free_State2(void)
{UDR='2';return;}
static void Show_Free_State3(void)
{UDR='3';return;}
static void Show_Free_State4(void)
{UDR='4';return;}
static void Show_Free_State5(void)
{UDR='5';return;}

static void Show_Frame(void)
{
 UDR=*((unsigned char *)(&Block_Time_Counter)+1);
 return;
}

void Inc_Block_Time_Counter(void)
{
 Block_Time_Counter++;                                  //incrementa el tiempo de bloque...
 if(Is_New_Central_Time())                              //si hay un master time listo...
  Block_Time_Counter=Read_New_Central_Time();//se asigna es tiempo...
 return;
}

unsigned char Is_Snap_Shot_Time_Block_Time(void)          //esta función compara si el próximo frame hay que procesarlo ya, ya se paso , o todavía hay que esperar...
{
 unsigned int Snap_Shot_Time=Read_Snap_Shot_Time();     //por cuestiones de arquitectura se declara una automática y se asigna este valor...
 if(Block_Time_Counter==Snap_Shot_Time)                          //si es el tiempo actual exacto...
  return 1;                                             //devuelve uno...
 if(Block_Time_Counter>Snap_Shot_Time)                          //si el tiempo actual es mayor que el de la foto...
  return (Block_Time_Counter-Snap_Shot_Time)<30000;      //retorna la resta comparado con 30000...
```

```
 return (Snap_Shot_Time-Block_Time_Counter)>30000;       //sino retorna la resta invertida comparada
con 30000...
}

unsigned char Is_Snap_Shot_Time_Snap_Shot_Block_Time(void)
{
 return (Read_Snap_Shot_Block_Time()==Read_Snap_Shot_Time()); //retorna si el tiempo de foto es
igual al tiempo de bloque...
}

void New_Frame_Func(void)
{
 if(Is_New_Frame() && Is_Snap_Shot_Time_Block_Time())            //si hay un nuevo frame y coincide
con el tiempo...
   Snap_Shot_Block2Snap_Shot_Frame();                           //se avanza la foto...
 else
  Snap_Shot_Frame2Snap_Shot_Block();                            //sino se atrasa la foto...
 Secuence();                                                    //se mandan los datos...
 return;
}

void  Gray_Scale_Func(void)
{
 if(Is_New_Frame() && Is_Snap_Shot_Time_Snap_Shot_Block_Time())        //si hay un nuevo frame y
coincide con el tiempo...
   Secuence();                                                  //se mandan los datos...
  else                                                          //sino...
   Secuence_Zero();                                             //se mandan ceros...
 return;
}

//-------------------------------------------------------------

void  Send_Message2Frame_State_Machine(unsigned char Event)
{
Write_Circ_FIFO((unsigned char)&Frames_State_Machine);          //a la maquina de estados
multipropósito...
 Write_Circ_FIFO(Event);                                                //se manda cualquier
evento para que avance,,..
 return;
}

void  Atomic_Send_Message2Frame_State_Machine(unsigned char Event)
{
 __disable_interrupt();
 Send_Message2Frame_State_Machine(Event);
 __enable_interrupt();
 return;
}

void Frames_Rti(void)
{
 Send_Message2Frame_State_Machine(ANY_Event);
 return;
}

//************ MAQUINA DE ESTADOS PARA EVERYTHINGS **************

static State __flash Initializing_Frames[] =
{
 ANY_Event       ,Init_Frames       ,Free_State1,
};
```

```
static State __flash Free_State1[] = //normal mode
{
 New_Frame_Event       ,New_Frame_Func                    ,Free_State1,
 Gray_Scale_Event      ,Gray_Scale_Func   ,Free_State1,
 ANY_Event             ,Rien                              ,Free_State1,
};

static State __flash Free_State2[] =  //FIFO_Empty from Normal
{
 ANY_Event             ,Rien                    ,Free_State2,
};

static State __flash Free_State3[] = //animation
{
 ANY_Event             ,Rien              ,Free_State3,
};

static State __flash Free_State4[] = //fifo empty from animation
{
 ANY_Event             ,Show_Free_State4          ,Free_State4,
};

static State __flash  Free_State5[] = //checking if FIFO empty after end_animation
{
 ANY_Event            ,Show_Free_State5          ,Free_State5,
};

static State __flash Free_State6[] =
{
 ANY_Event       ,Rien                      ,Free_State7,
};

static State __flash Free_State7[] =
{
 ANY_Event       ,Rien                      ,Free_State2,
};
```

- Modulo de la fifo circular de frames

```
//archivo de implementacion de una cola circular
#include <frames_struct.h>
#include <iom16.h>
#include <inavr.h>

typedef struct Frames
{
 unsigned int Time;
 unsigned char Gray_Level;
 unsigned char Data[8];
}Frames_Struct;

__no_init static __near Frames_Struct Frames[MAX_FRAMES];

__no_init static unsigned char Frame2Write;          //puntero para escritura...
__no_init static unsigned char Block2Write;          //puntero para lectura...
__no_init static unsigned char Snap_Shot_Frame;
__no_init static unsigned char Snap_Shot_Block;
```

```
#define Next_Frame(Actual) ((Actual == MAX_FRAMES-1)?0x00:Actual+1)

void Init_Data_Circ_FIFO(void)              //inicializa la cola circular...
{
 Frame2Write = 0x00;                        //se inicializan los punteros...
 Block2Write =0x00;
 Snap_Shot_Frame=0x00;
 Snap_Shot_Block=0x00;
 return;
}

void Write_Time(unsigned int Time)
{
 Frames[Frame2Write].Time=Time;                    //escribe el tiempo en la estructura...
 return;
}

void Write_Gray_Level(unsigned char Level)
{
 Frames[Frame2Write].Gray_Level=Level;  //escribe el nivel de grises en la estructura...
 return;
}

void Write_Data(unsigned char Index,unsigned char Data)
{
 Frames[Frame2Write].Data[Index]=Data;              //escribe el dato en la estructura...
 return;
}

unsigned char Read_Snap_Shot_Data(unsigned char Index)
{
 return Frames[Snap_Shot_Frame].Data[Index];       //lee un dato de la estructura...
}

void Inc_Frame2Write(void)
{
 Frame2Write=Next_Frame(Frame2Write);              //incrementa el puntero a escribir..
 return;
}

void Inc_Snap_Shot_Frame(void)
{
 Snap_Shot_Frame=Next_Frame(Snap_Shot_Frame);           //incrementa el puntero a foto...
 return;
}

unsigned char Read_Gray_Level(void)
{
 return Frames[Snap_Shot_Frame].Gray_Level;       //lee el nivel de gris...
}

unsigned int Read_Snap_Shot_Time(void)
{
 return Frames[Snap_Shot_Frame].Time;              //lee el tiempo de foto...
}
unsigned int Read_Snap_Shot_Block_Time(void)
{
 return Frames[Snap_Shot_Block].Time;              //es el tiempo de bloque...
}

unsigned char Is_New_Frame(void)
```

```
{
 return Snap_Shot_Frame!=Block2Write;          //compara la foto con el bloque...
}

void Snap_Shot_Block2Snap_Shot_Frame(void)
{
 Snap_Shot_Block=Snap_Shot_Frame;              //asigna la foto al bloque...
 return;
}

void Snap_Shot_Frame2Snap_Shot_Block(void)
{
 Snap_Shot_Frame=Snap_Shot_Block;              //asigna el bloque a la foto...
 return;
}

void Block2Write2Frame2Write(void)
{
 Block2Write=Frame2Write;                      //asigna el frame al bloque..
 return;
}

void Frame2Write2Block2Write(void)
{
 Frame2Write=Block2Write;                      //asigna el bloque al frame...
 return;
}

unsigned char Available_Blocks(void)                    //calcula el numero de frames disponibles...
{
 if(Block2Write>=Snap_Shot_Block)                       //si el bloque a escribir es mayor o igual que
el bloque de foto...
  return MAX_FRAMES-1-(Block2Write-Snap_Shot_Block);    //retorna este valor..
 else                                                   //sino
  return (Snap_Shot_Block-Block2Write)-1;               //retorna este otro...
}
```

- Modulo main

```
#include <iom16.h>
#include <inavr.h>
#include <State_Machine.h>
#include <Circ_Fifo.h>
#include <Everythings.h>
#include <rti.h>
#include <frames.h>
#include <data_clk.h>
#include <frames_struct.h>
#include <serial.h>


  void main(void)
{
 Init_Circ_FIFO();                      //inicializa la fifo de eventos...
 Init_Data_Circ_FIFO();                 //inicializa la fifo de estructuras...
 Init_Everythings_State_Machine();      //inicializa las maquinas de estado..
 Init_Frames_State_Machine();
 Init_Serial_State_Machine();
 Init_Parser();
```

```
 Init_Rti();                                    //inicializa la irq periódica...
 for(;;)                                 //loop infinito...
  State_Machine();                              //ejecuta el kernel
}
```

- Modulo Reset

```
NAME reset

      RTMODEL "__64bit_doubles", "disabled"
      RTMODEL "__cpu", "3"
      RTMODEL "__cpu_name", "AT90Mega16"
      RTMODEL "__enhanced_core", "enabled"
      RTMODEL "__has_elpm", "false"
      RTMODEL "__memory_model", "1"
      RTMODEL "__rt_version", "2.30"

      RSEG CSTACK:DATA:NOROOT(0)
      RSEG RSTACK:DATA:NOROOT(0)

      EXTERN main
      FUNCTION main,0a02H

      PUBLIC Reset
      FUNCTION Reset,021203H
      LOCFRAME RSTACK, 2, STACK


      RSEG CODE:CODE:NOROOT(1)

Reset:
      FUNCALL Reset, main
      LOCFRAME RSTACK, 2, STACK
      LDI R28,$D7     ;al stack de variables automáticas le indico que arranque
                  ;dejando espacio para 20 llamadas reentrantes de funciones
                  ;o sea 40 bytes 4FF-28=4D7
      LDI R29,$04     ;notar que el R28 y el R29 juntos forman el Y que se usa
                  ;exclusivamente para puntero de stack a variables auto..
      LDI R16,$01     ;cargo en el 16 el valor 01..
      OUT $3D,R16     ;parte baja del stack de direcciones de retorno
      LDI R16,$05     ;cargo en el 16 el 05.
      OUT $3E,R16     ;parte alta del stack de direcciones de retorno. Notar el
                  ;detalle de que la máxima posición de Ram para el at90m16 es
                  ;4FF pero yo pongo el stack en la $501 como puede ser??
          CALL    main    ;es que esta llamada al main pushea 2 bytes como dirección
                      ;de retorno, pero como yo nunca pienso salir del main, para
                      ;que lasa quiero guardar? con lo cual el micro piensa que las
                      ;guarda, decrementa el stack y queda justo en 4FF para la
                      ;próxima función que entre que seguro que la quiero guardar...
                      ;A ver si el startup  te hace esto man!!!

      END
```

- Modulo serial

```
//archivo dedicado a la recepción de los datos...
#include <iom16.h>
#include <inavr.h>
#include <serial.h>
#include <frames_struct.h>
```

```
#include <state_machine.h>
#include <circ_FIFO.h>
#include <loosing_Time.h>

static State __flash
       Initializing_Serial[1],
            Receiving_ID[4],
            Receiving_Length[4],
            Receiving_Time_Lo[4],
            Receiving_Time_Hi[4],
            Receiving_Gray_Level[4],
            Receiving_Data[4],
            Waiting_End[4],
            Receiving_Command[2],
            Receiving_Central_Time_Lo[1],
            Receiving_Central_Time_Hi[1];

__no_init State __flash * __tiny Serial_State_Machine;          //variable que lleva cuenta del estado de la
maquina de estados de "detodo un poco"...

__no_init static unsigned char Data_Index;
__no_init static unsigned char Frame_Index;
__no_init static unsigned int Time;
__no_init static unsigned int Central_Time;
__no_init static unsigned char New_Central_Time;
__no_init static unsigned char Serial_Time_Out;

void USART_Init(void)
{
 unsigned int baud=51;
 UBRRH = (unsigned char)(baud>>8);
 UBRRL = (unsigned char)baud;

 UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
 UCSRC = (1<<URSEL)| (1<<UCSZ0) | (1<<UCSZ1);
 UCSRA |= (1<<MPCM);

 New_Central_Time=0;
 Atomic_Send_Message2Serial_State_Machine(End_Event);
 return;
}



#pragma vector=0x02C-4
__interrupt void Serial_Service(void)
{
 Send_Message2Serial_State_Machine(UDR);
 return;
}

void Receive_Central_Time_Hi(void)
{
 *((unsigned char*)&Central_Time+1)=Read_Actual_Event();
 return;
}
void Receive_Central_Time_Lo(void)
{
 *((unsigned char*)&Central_Time+0)=Read_Actual_Event();
 New_Central_Time=1;
 UCSRA|=(1<<MPCM);
 Atomic_Send_Message2Serial_State_Machine(End_Event);
```

```
 return;
}



unsigned char Is_New_Central_Time(void)
{
 return New_Central_Time;
}

unsigned int Read_New_Central_Time(void)
{
 New_Central_Time=0;
 return Central_Time;
}


void Answer_ID(void)
{
 UDR=Read_Actual_Event();
 UCSRA &= ~(1<<MPCM);
 return;
}


//---------------------------------------------------------------
void Check_If_Available(void)
{
 Frame_Index=Read_Actual_Event();
 if((UDR=Available_Blocks())<Frame_Index)
  {
   UCSRA |= (1<<MPCM);
   Atomic_Send_Message2Serial_State_Machine(End_Event);
   Serial_State_Machine=Waiting_End;
  }
 return;
}
void Receive_Time_Hi(void)
{
 *((unsigned char*)&Time+1)=Read_Actual_Event();
 return;
}
void Receive_Time_Lo(void)
{
 *((unsigned char*)&Time+0)=Read_Actual_Event();
  Write_Time(Time);
 return;
}
void Receive_Gray_Level(void)
{
 Write_Gray_Level(Read_Actual_Event());
 Data_Index=0;
 return;
}
void Receive8Data(void)
{
 Write_Data(Data_Index,Read_Actual_Event());
 if(++Data_Index>7)
  {
   Inc_Frame2Write();
   if(!--Frame_Index)
    {
```

```
    Block2Write2Frame2Write();
    UCSRAl=(1<<MPCM);
    Serial_State_Machine=Waiting_End;
    Atomic_Send_Message2Serial_State_Machine(End_Event);
    UDR=0xCC;
   }
  else
   {
    Write_Time(Time);
    Serial_State_Machine=Receiving_Gray_Level;
   }
 }
return;
}

void Invalid_Data(void)
{
 UCSRAl=(1<<MPCM);
 Atomic_Send_Message2Serial_State_Machine(End_Event);
 return;
}

//---------------------------------------------------------------
static void Init_Serial(void)            //inicializa los puertos que se usan en esta maquina de estados de
propósitos múltiples...
{
 USART_Init();
 return;
}
void Init_Serial_State_Machine(void)
{
 Serial_State_Machine=Initializing_Serial;  //inicializa la maquina de estados de propósitos múltiples en
el estado de inicialización ...
 Init_Serial();
 return;
}
void  Send_Message2Serial_State_Machine(unsigned char Event)
{
 Write_Circ_FIFO((unsigned char)&Serial_State_Machine);              //a la maquina de estados
multiproposito...
 Write_Circ_FIFO(Event);                                                    //se manda cualquier
evento para que avance,,..
 return;
}

void  Atomic_Send_Message2Serial_State_Machine(unsigned char Event)
{
 __disable_interrupt();
 Send_Message2Serial_State_Machine(Event);
 __enable_interrupt();
 return;
}

void Serial_Rti(void)
{
 Send_Message2Serial_State_Machine(ANY_Event);
 return;
}
void Dec_Serial_Time_Out(void)
{
 if(!--Serial_Time_Out)
  {
```

```
  Invalid_Data();
  Serial_State_Machine=Waiting_End;
 }
return;
}
void Set_Serial_Time_Out(void)
{
Serial_Time_Out=10; //(10=3seg +o-)
return;
}
void Answer_ID_And_Set_Serial_Time_Out(void)
{
Answer_ID();
Set_Serial_Time_Out();
return;
}

//************ MAQUINA DE ESTADOS PARA EVERYTHINGS **************

static State __flash Initializing_Serial[] =
{
ANY_Event      ,Init_Serial         ,Waiting_End,
};
static State __flash Receiving_ID[] =
{
100            ,Answer_ID_And_Set_Serial_Time_Out     ,Receiving_Length,
0              ,Set_Serial_Time_Out                   ,Receiving_Length,
ANY_Event      ,Rien                                  ,Receiving_ID,
};
static State __flash Receiving_Length[] =
{
0              ,Rien                  ,Receiving_Command,
ANY_Event      ,Check_If_Available    ,Receiving_Time_Hi,// || Waiting_End
};
static State __flash  Receiving_Time_Hi[] =
{
ANY_Event      ,Receive_Time_Hi       ,Receiving_Time_Lo,
};
static State __flash Receiving_Time_Lo[] =
{
ANY_Event      ,Receive_Time_Lo       ,Receiving_Gray_Level,
};
static State __flash Receiving_Gray_Level[] =
{
ANY_Event      ,Receive_Gray_Level    ,Receiving_Data,
};
static State __flash Receiving_Data[] =
{
ANY_Event      ,Receive8Data          ,Receiving_Data, // ||Receiving_Gray || Waiting_End
};
static State __flash Waiting_End[] =
{
End_Event      ,Frame2Write2Block2Write      ,Receiving_ID,
ANY_Event      ,Rien                        ,Waiting_End,
};



static State __flash Receiving_Command[] =
{
Central_Time_Event     ,Rien                ,Receiving_Central_Time_Hi,
ANY_Event              ,Invalid_Data        ,Waiting_End,
```

```
};
static State __flash Receiving_Central_Time_Hi[] =
{
 ANY_Event      ,Receive_Central_Time_Hi          ,Receiving_Central_Time_Lo,
};
static State __flash Receiving_Central_Time_Lo[] =
{
 ANY_Event      ,Receive_Central_Time_Lo          ,Waiting_End,
};
```

5.2.4.2. Listado del código del uP central

```
#class auto
#define BINBUFSIZE  15
#define BOUTBUFSIZE 15

const unsigned char Data[]=
{
3,0,30,0x07,0x05,0x1F,0x00,0x00,0x00,0x00,0x00, //1
3,10,30, 0x00,0x07,0x05,0x1F,0x00,0x00,0x00,0x00, //2
3,20,30, 0x1F,0x00,0x07,0x05,0x1F,0x00,0x00,0x00, //3
3,40,30, 0x05,0x1F,0x00,0x07,0x05,0x1F,0x00,0x00, //4
3,60,30, 0x1F,0x05,0x1F,0x00,0x07,0x05,0x1F,0x00, //5
3,80,30, 0x00,0x1F,0x05,0x1F,0x00,0x07,0x05,0x1f, //6
3,100,30, 0x1F,0x00,0x1F,0x05,0x1F,0x00,0x07,0x05, //7
3,120,30, 0x15,0x1F,0x00,0x1F,0x05,0x1F,0x00,0x07, //8
3,140,30, 0x0A,0x15,0x1F,0x00,0x1F,0x05,0x1F,0x00, //9
3,160,30, 0x00,0x0A,0x15,0x1F,0x00,0x1F,0x05,0x1F, //10
3,180,30, 0x1F,0x00,0x0A,0x15,0x1F,0x00,0x1F,0x05, //11
3,200,30, 0x10,0x1F,0x00,0x0A,0x15,0x1F,0x00,0x1F, //12
3,210,30, 0x10,0x10,0x1F,0x00,0x0A,0x15,0x1F,0x00, //13
3,220,30, 0x00,0x10,0x10,0x1F,0x00,0x0A,0x15,0x1F, //14
3,240,30, 0x1F,0x00,0x10,0x10,0x1F,0x00,0x0A,0x15, //15
4,4,30, 0x11,0x1F,0x00,0x10,0x10,0x1F,0x00,0x0A, //16
4,24,30, 0x1F,0x11,0x1F,0x00,0x10,0x10,0x1F,0x00, //17
4,44,30, 0x80,0x1F,0x11,0x1F,0x00,0x10,0x10,0x1F, //18
4,64,30, 0xE7,0x80,0x1F,0x11,0x1F,0x00,0x10,0x10, //19
4,84,30, 0x7E,0xE7,0x80,0x1F,0x11,0x1F,0x00,0x10, //20
4,104,30, 0x18,0x7E,0xE7,0x80,0x1F,0x11,0x1F,0x00, //21
4,124,30, 0x0C,0x18,0x7E,0xE7,0x80,0x1F,0x11,0x1F, //22
4,144,30, 0x06,0x0C,0x18,0x7E,0xE7,0x80,0x1F,0x11, //23
4,164,30, 0x03,0x06,0x0C,0x18,0x7E,0xE7,0x80,0x1F, //24
4,184,30, 0x00,0x03,0x06,0x0C,0x18,0x7E,0xE7,0x80, //25
4,204,30, 0xC0,0x00,0x03,0x06,0x0C,0x18,0x7E,0xE7, //26
4,224,30, 0x88,0Xc0,0x00,0x03,0x06,0x0C,0x18,0x7E, //27
4,244,30, 0xF8,0x88,0Xc0,0x00,0x03,0x06,0x0C,0x18, //28
5,8,30, 0x08,0xF8,0x88,0Xc0,0x00,0x03,0x06,0x0C, //29
5,28,30, 0x00,0x08,0xF8,0x88,0Xc0,0x00,0x03,0x06, //30
5,48,30, 0xF8,0x00,0x08,0xF8,0x88,0Xc0,0x00,0x03, //31
5,68,30, 0x80,0xF8,0x00,0x08,0xF8,0x88,0Xc0,0x00, //32
5,88,30, 0xF8,0x80,0xF8,0x00,0x08,0xF8,0x88,0Xc0, //33
5,108,30, 0x00,0xF8,0x80,0xF8,0x00,0x08,0xF8,0x88, //34
5,128,30, 0xF8,0x00,0xF8,0x80,0xF8,0x00,0x08,0xF8, //35
5,148,30, 0x80,0xF8,0x00,0xF8,0x80,0xF8,0x00,0x08, //36
5,168,30, 0x80,0x80,0xF8,0x00,0xF8,0x80,0xF8,0x00, //37
5,188,30, 0x01,0x80,0x80,0xF8,0x00,0xF8,0x80,0xF8, //38
```

```
5,208,30, 0xFA,0x01,0x80,0x80,0xF8,0x00,0xF8,0x80, //39
5,228,30, 0x01,0xFA,0x01,0x80,0x80,0xF8,0x00,0xF8, //40
5,248,30, 0x00,0x01,0xFA,0x01,0x80,0x80,0xF8,0x00, //41
6,12,30, 0x00,0x00,0x01,0xFA,0x01,0x80,0x80,0xF8, //42
6,32,30, 0x00,0x00,0x01,0xFA,0x01,0x80,0x80,0xF8, //43
6,52,30, 0x00,0x01,0x02,0xF4,0x02,0x01,0x00,0xF0, //44
6,72,30, 0x00,0x03,0x04,0xE8,0x04,0x03,0x00,0xE0, //45
6,92,30, 0x00,0x07,0x08,0xD1,0x08,0x07,0x00,0xC0, //46
6,112,30, 0x00,0x0E,0x11,0xA2,0x11,0x0E,0x00,0x80, //47
6,132,30, 0x00,0x1C,0x22,0x44,0x22,0x1C,0x00,0x00, //48
6,152,30, 0x00,0x38,0x44,0x88,0x44,0x38,0x00,0x00, //49
6,172,30, 0x00,0x70,0x88,0x10,0x88,0x70,0x00,0x00, //50
6,192,30, 0x00,0xE0,0x10,0x20,0x10,0xE0,0x00,0x00, //51
6,212,30, 0x00,0xC0,0x20,0x40,0x20,0xC0,0x00,0x00, //52
6,232,30, 0x00,0x80,0x40,0x80,0x40,0x80,0x00,0x00, //53
6,252,30, 0x00,0x00,0x80,0x00,0x80,0x00,0x00,0x00, //54
7,16,30, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //55
};



CoData cost1,cost2; // allocate memory for a CoData struct

void main()
{
 int c,d,x;

 c = 1000;
 x = 0;
 serBopen(19200);
 loopinit();
 while (1)
  {
   loophead();
   costate  cost1 always_on
    {
     serBparity(PARAM_2STOP);
     wfd  cof_serBputc(100);
     serBparity(PARAM_EPARITY);
     CoBegin(&cost2);
     do
       wfd d=cof_serBgetc();
     while (d!=100);
     CoReset(&cost2);
     wfd  cof_serBputc(1);
     CoBegin(&cost2);
     wfd d=cof_serBgetc();
     CoReset(&cost2);
     if(d>=1)
      {
       wfd cof_serBwrite(&Data[x],11);
       CoBegin(&cost2);
       wfd d=cof_serBgetc();
       if (d!=0xCC)
        {
         CoReset(&cost2);
         CoReset(&cost1);
         CoBegin(&cost1);
        }
```

```
                    else
                     {
                      x+=11;
                      if(x>603)
                       {
                        wfd  cof_serBputc(0xDD);
                        CoReset(&cost1);
                        CoPause(&cost1);
                        CoReset(&cost2);
                       }
                      else
                       {
                        CoReset(&cost2);
                        CoReset(&cost1);
                        CoBegin(&cost1);
                       }
                     }
                    }
                    else
                     {
                      CoReset(&cost2);
                      CoReset(&cost1);
                      CoBegin(&cost1);
                     }
                   }
                   costate cost2
                   {
                    waitfor (DelayMs(3000));
                    printf("Reset\n");
                            CoReset(&cost1);
                            CoPause(&cost1);
                    waitfor (DelayMs(3000));
                    CoBegin(&cost1);
                    CoReset(&cost2);
                   }
                  }
                 serBclose();              // disables B serial port reading and writing
}
```

5.2.5. Plan de prueba de módulos y de depuración de soft

Pruebas de la interfaz serial:

Es conveniente levantar esta interfaz antes que los demás módulos para poder usarla como comunicación con la PC para debugguear los demás módulos.

Para ello simplemente se inicializan los registros y las interrupciones y se la configura en modo repetidora, de manera de que pueda transmitir lo que acaba de recibir. Una vez que esto funciona, se hace funcionar la maquina de estados que graba la fifo.

Pruebas de la fifo circular de frames:

Para probar este modulo se deberá realizar una rutina simple que escriba un frame en la fifo y otra que lo lea y compare con lo que se escribió. Así repetidamente un alto numero de veces y con distintos ritmos de escritura y extracción de los datos para exponer a la fifo a las situaciones de fifo llena fifo vacía, limite físico del array, etc. Si los datos leídos corresponden con los escritos durante toda la simulación, se habrá probado el modulo.

Prueba del modulo de frames:

Usando el sensor de cruce por cero como entrada al input capture, se utiliza la interfaz serial para transmitir un valor creciente una vez por segundo y se controla que los tiempos sean los correctos.

Luego se transmitirá una vez por segundo datos incrementales a la placa de potencia y se corroborara con el encendido de las lamparas. Finalmente se agregara el apagado e a tiempo controlado

dentro del semiciclo para generar la escala de grises. Una vez resuelto todo esto se procede a cambiar los datos incrementales por los datos levantados de la fifo circular de frames.

6. Construcción del prototipo

6.1. Definición de los módulos

Se definen 2 modulos. El modulo que contiene el uP de celda, y la placa de potencia de la celda.

6.2. Diseño de los circuitos impresos

6.2.1. Diseño del PCB del uP de celda

Se decidió realizar en placa separada todo lo relacionado con el uP de la celda, esto es la fuente de alimentación, las protecciones contra desperfectos, el uP , el oscilador, etc. para poder en un futuro usar un uP para mas de una placa de potencia. Por otro lado se comprobó que si en el momento de encender la placa de potencia, no están debidamente polarizados los transistores por ejemplo en cero , para que la pantalla arranque apagada, se disparan aleatoriamente una gran cantidad de lamparas. Esto produce un pico de corriente muy elevado si se consideran cientos de lampara y puede hacer saltar las protecciones o producir un arco en la llave de encendido que produciría el desgaste prematuro de la misma, con lo cual se hace imposible encender la pantalla de esta manera. Para lograrlo , primero habrá que alimentar el uP de celda hasta que alimente los SR y estos pongan los transistores en una condición estable, preferentemente apagados, y recién luego se puede alimentar a las lamparas que se inicializarian todas apagadas sin peligro alguno. Este detalle es posible realizarlo con dos placas separadas, cada una con su propia red de alimentación individual.

En este PCB se colocan las protecciones de las entradas y salidas del uP con zeners y resistencias. Estas protecciones también se podrían colocar en la placa de potencia, pero se decidió colocarlas acá por la cercanía al pin a proteger y porque como se menciono anteriormente, en un futuro es posible usar un uP para mas de una placa de potencia, y este esquema ahorraría componentes.

Las dimensiones de la placa están conforme a las medidas de la celda, porque los agujeros de sujeción coinciden con los intersticios de las lamparas, ya que son lugares oscuros de la pantalla y no interfieren en la visual.

La conexión con la placa de potencia se hará a través de cable plano y regleta de pines

6.2.2. Diseño del PCB de la placa de potencia

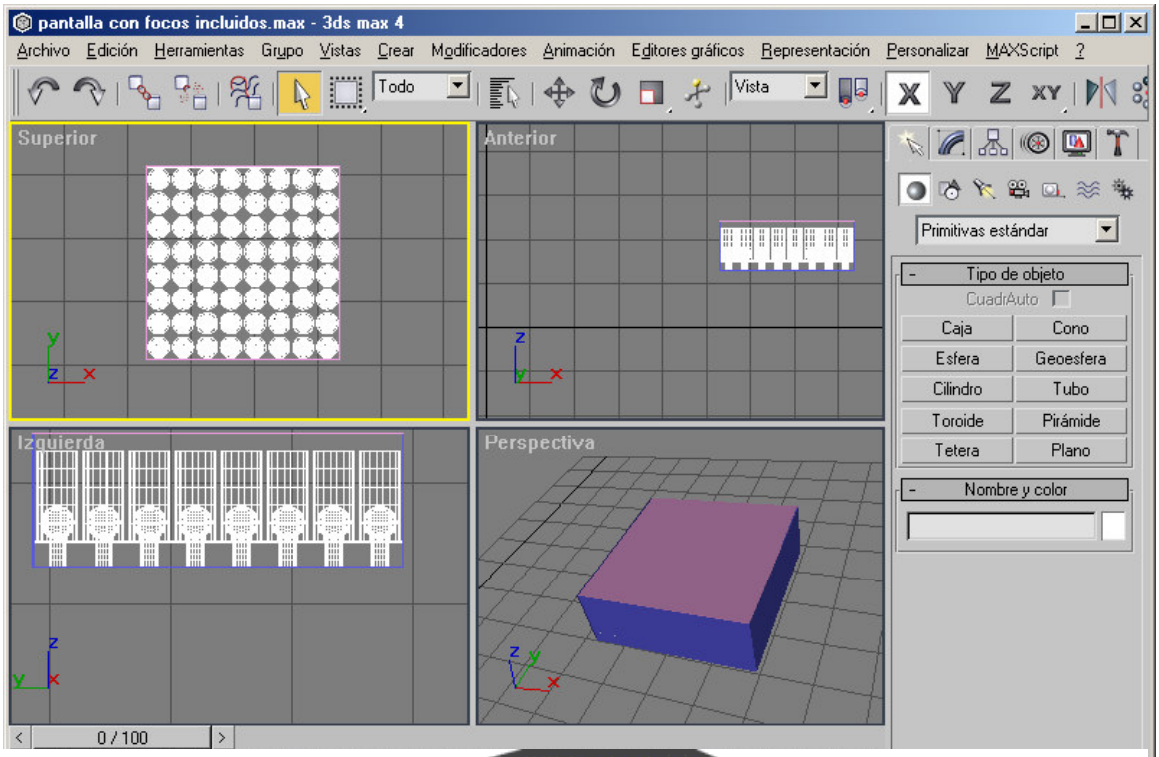
Esta placa fue la que llevo mas tiempo por su densidad de componentes y requerimientos. Contempla un mix entre componentes de SMD y Througth-hole a la vez que intenta hacer una buena distribución de las pistas mas cargadas en el pero caso de encender todas las lamparas.

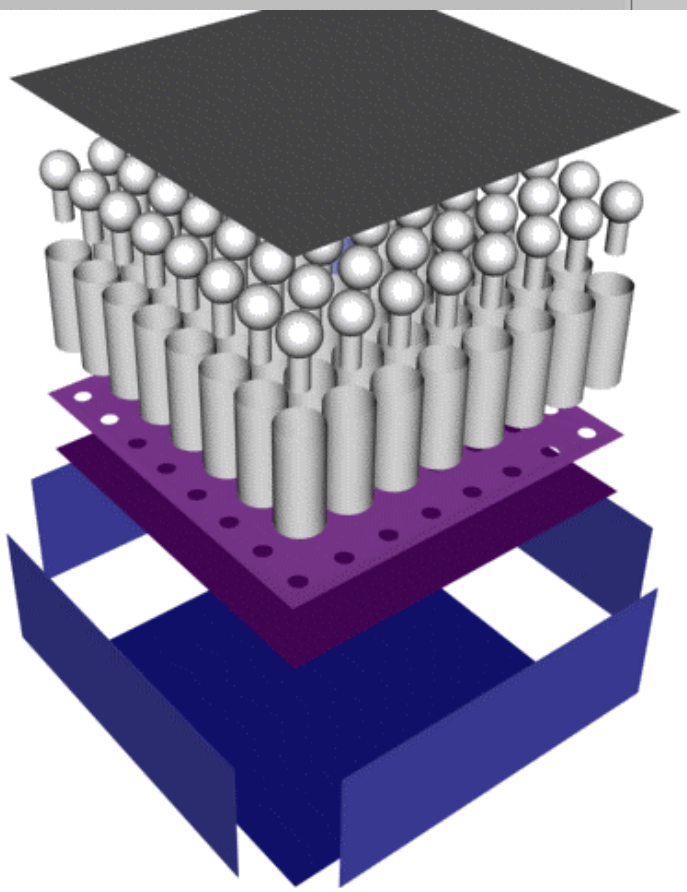
Permite la conexión a 64 lamparas por medio de 2 cables por cada una o por una cable común y uno individual en el caso de que el común sea el mismo chasis. Esto se hace previniendo la situación de chasis aislado de manera de que cada lampara requiera de 2 cables para su funcionamiento.

Las salidas a las lamparas se hace en forma ordenada considerando que la placa se ubicara mecánicamente en el medio de la celda. Contemplando, como en la placa del uP las posiciones de los agujeros de sujeción en función de los intersticios de las lamparas.

6.3. Diseño mecánico

Se presenta un bosquejo del primer prototipo de una celda con sus 64 lamparillas y los separadores individuales :





La plancha perforada es de aluminio y sirve para acomodar las lamparas en forma de grilla a la vez que conecta uno de los terminales de la lampara. En el contorno tendrá pliegues de manera de poder encastrarse con las celdas contiguas.
La aislación de la celda podrá hacerse individualmente o bien en grupos de celdas, dependiendo de la aplicación.

El material del frente de la celda será de policarbonato ya que es un material resistente a la luz solar.

6.4. Detalles de construcción y precauciones especiales de montaje

Debido a que el portalámparas es el mismo chasis, este estará conectado a un nivel de tensión elevado respecto de tierra. Con lo cual requiere de suma atención para lograr una adecuada aislación con el medio exterior y con las placas driver. Notar además que como se usan transistores que permiten el paso de la corriente en un solo sentido, fue necesario rectificar completamente la tensión de red, y debido a ello, el chasis no se puede asignar al neutro de la red, ya que después de rectificar se pierde el sentido de neutro, de manera de que el chasis tendrá potencial con respecto al neutro y al vivo dependiendo de donde se encuentre el semiciclo de la red y habrá que aislarlo completamente del exterior. Sin embargo si las celdas están alimentadas por el mismo polo de red, pueden tener contacto entre si, ya que lo único que se lograría es distribuir la carga a través de todos los puentes de diodos que estén actuando. Si por el contrario se usan distintos polos, se deberán aislar los chasis.

7. Validación del prototipo

7.1. Validación de hard

7.1.1. Plan y protocolos especiales de medición

Se desarrollo un firmware especial para medir la etapa de potencia que consta de 3 funciones. La primera simplemente ejecuta el encendido de todas las lamparas para que se verifique no solo el encendido de las mismas sino la tensión colector emisor de los transistores de salida para asegurarse de que esta dentro del rango esperado.

La segunda función realiza la transmisión secuencial de datos en forma creciente y en lapsos de 10 segundos para verificar que las tensiones de entrada a los shift register's estén dentro de lo esperado y no se supere la máxima corriente de los pines del micro.

Por ultimo la tercera función permite verificar los niveles de tensión de la interfaz RS232 realizando la transmisión de datos constantes.

7.1.2. Medidas

Para la validación se fijaron limites dentro de los cuales se espera que estén los resultados de las mediciones. Estos limites responden a ciertos criterios que se detallan para cada medición.

- Vce_on

  La tensión colector emisor de los transistores en modo encendido debe mantenerse debajo de cierto valor, para evitar la sobredisipación. Con el mínimo teórico de 0.5V aproximadamente, el limite superior dependerá de la carga, para esta medición se uso una lampara de 25W, con lo cual se espera una corriente RMS máxima de 25/220=0.12 A. Luego la potencia máxima

- Ipol

  Es la corriente de polarización del primer transistor del darlington. Deberá ser menor a la máxima soportada por el CD4094 a la vez que suficiente para saturar completamente el segundo transistor del darlington.

  Considerando un Hfe de 20 en las condiciones de trabajo normales a las que estará sometido, y con una Ic = 0.12 *1.42 A máxima, se requerirá de una Ipol > 0.l2 * 1.42 / Hfe^2 = 1.7 mA. Que esta debajo de lo soportado por el SR.

- Máximo Jitter de nivel de gris

  Mientras el uP de celda esta controlando los tiempos para generar la escala de grises, es posible y muy probable que también se estén recibiendo nuevos frames por la UART. Si bien la UART es embebida, una vez completado el byte, el uP deberá almacenarlo para habilitar la UART a recibir el próximo y evitar el overrun. Este tiempo dedicado a almacenar el byte, puede inferir en los tiempos de la escala de grises ya que lo que se busca es una tasa lo mas alta posible en la recepción. El mecanismo implementado para la escala de grises y la UART fueron diseñados para

minimizar el tiempo de trabajo de la IRQ , trasladando este trabajo al main de forma ordenada, de manera de evitar grandes jitters en la escala de grises y así evitar el parpadeo de las lamparas ya que no es agradable visualmente. Con todo esto se espera que el jitter este por debajo del 10% del tiempo mínimo entre niveles de grises. Esto es J < 10ms/78 * 0.1 = 12uSeg

- Ripple de la fuente de 5V.

El componente mas critico en función de la alimentación es el uP que tolera desde 4.75 hasta 5.25. Con lo cual se espera que el ripple en pines de alimentación del uP en condiciones criticas sea menor a la mitad de lo tolerable. Esto es desde 4.88 hasta 5.12 aproximadamente. La condición mas critica es cuando los 64 transistores están conmutando. Con ayuda del firmware especial se obtiene esta medición.

- Corriente AC de la celda.

Debido a que el coeficiente resistivo del filamento de la lampara es positivo; cuando este se encuentre frío presentara una resistencia mucho menor que la que tiene a temperatura nominal. Esto se traduce a una corriente mucho mayor que la nominal durante el encendido de la misma. Habrá que contemplar esta situación para elegir los componentes asociados.

Luego la peor condición a la que se puede ver sujeta la celda será cuando todas las lamparas estando a temperatura ambiente sean encendidas hasta su intensidad nominal en el mismo instante.

La medición se realiza con una sola lampara y se multiplican los efectos.

Si bien esta condición es critica, existe una manera de disminuir sus efectos. Se trata de evitar el salto brusco de temperatura. Como se conoce de antemano la información a transmitir, es posible agregar inteligencia a la animación de manera de prever el instante en que una lampara, que ha estado suficiente tiempo apagada para alcanzar la temperatura ambiente, es encendida, para adelantarse y durante algunos ciclos antes encenderla escalonadamente hasta un nivel de gris que es invisible visualmente, pero que alcanza una temperatura muy superior a la ambiente. Luego cuando sea el momento de llevarla a su estado definitivo , el salto de temperatura será mucho menor. Sumado a esta prevención, el encendido suave por el cruce por cero disminuye significativamente los efectos.

Esta aplicación tiene un efecto lateral en beneficio del sistema que es disminuir el tiempo efectivo de encendido, ya que mientras desde Tambiente a Tfinal la lampara tarda x, con este método los tiempos serán sensiblemente menores, y eso lleva a la posibilidad de visualizar animaciones mas veloces a la vez que teóricamente alarga la vida útil de la lampara ya que deberá soportar menores cambios de temperatura. Claro que lo que no se logra con este método es disminuir el tiempo de apagado que es mucho mayor que el encendido y limitante del tiempo entre cuadros.

Por el contrario esta aplicación infiere en la actividad del bus de comunicaciones incrementando la tasa y además incrementa la corriente promedio y disminuye la eficiencia, ya que una lampara que esta consumiendo energía pero no se visualiza solo esta generando calor.

Esta aplicación será tenida en cuanta solo en futuras revisiones del equipo.

Por el momento llegado el caso de sobrepasar la capacidad de corriente soportada, se agregara un filtro que impida situaciones criticas por cada celda, es decir si llegan frames que requieren el encendido de mas de N lamparas en un mismo instante, se trunca a M, siendo M el máximo admitido. Obviamente habrá que trasladar esta limitación al animador para que realice animaciones menos exigentes o que distribuya los encendidos simultáneos en espacios cortos de tiempos, tal que no se note el efecto de retardo a la vez que protege el hard.

Usando el firmware especialmente diseñado se midieron los valores que figuran en la siguiente tabla:

Parametro	Procedimiento	Minimo	Medido	Maximo
Vce_on	Con la version especial del firmware uno, se mide la tension colector emisor de los transistores de salida	0v	4v	10v
Ipol	Con la version especial del firmware dos, se mide la corriente de salida de los shift register's	1mA	1.5mA	2.5mA
Max. Jitter de nivel de gris	Con la version especial del firmware dos, se mide el maximo jitter de nivel de gris de una de las lamparas	0useg	5useg	12useg
Corriente pico de la lampara con encendido no suave	Con firmware y hardware especial se procede a medir la corriente inicial de la lampara estando a T ambiente desde la cresta de la tension de red	Inominal	4.5*Inominal	2.5*Inominal
Corriente pico de la lampara con encendido suave	Con firmware y hardware especial se procede a medir la corriente inicial de la lampara estando a T ambiente desde el 0V de la tension de red	Inominal	2*Inominal	2.5*Inominal
Ripple de la fuente de 5Vdc	Se mide el ripple en la placa de potencia y en la del uP	0v	10mv	50mv

7.1.3. Evaluación

Los valores medidos encajaron dentro de los márgenes esperados, salvo el encendido de la lampara desde temperatura ambiente desde la cresta de la tensión de red (311V). Esta prueba demostró la ventaja de encender la lampara desde el 0V, que aunque la corriente pico inicial duplica la nominal, es la mitad de la obtenida en el peor caso. Con lo cual se concluye que en ambos casos la lampara se vera sometida en cada encendido a una condición de estrés que acortara su vida útil, sin embargo teniendo en cuenta el procedimiento propuesto para futuras versiones en el apartado 7.1.2, se espera que se reduzca sensiblemente. Este procedimiento no solo desestresaría la lampara sino también los transistores que son sometidos a iguales condiciones.

7.1.4. Resultados

Fueron comentados en el apartado anterior.

7.2. Validación de soft

7.2.1. Validación de soft del uP de celda

Para evaluar la efectividad del soft, se lo somete a los casos mas exigentes.

Se coloca un osciloscopio midiendo los disparos de alguna de las lamparas, y mientras la lampara se setea en un nivel de gris conocido, se envían tramas de nuevos frames a toda velocidad y se mide el jitter del nivele de gris de la lampara, esperando que sea menor que el 10% del tiempo entre niveles.

Luego para evaluar el protocolo serial se esnifea el bus con una PC y se realizan todos los tipos de transferencias. Luego se analizan detenidamente los tiempos y datos enviados y recibidos para corroborar el buen funcionamiento y evitar intersticios temporales entre byte y byte.

7.2.2. Validación de soft del uP central

Mientras el uP central esta enviando los frames a las celdas se insertan datos extras mediante una PC, para llevar al uP central a una situación inesperada, pero posible debido a factores como ruido, mal

funcionamiento de una de las celdas, etc. Luego se deberá verificar que el uP central pueda salir por time_out de esa situación.

8. Estudios de confiabilidad de hard y de soft

8.1. Confiabilidad de Hard

Para calcular la confiabilidad del sistema se utilizo el método de predicción de análisis de la fatiga de los componentes, descripto en norma MIL-HDBK-217F de origen militar.

Para ello, a partir de las características detalladas de los componentes, sus tasas de fallas individuales y las condiciones dadas de funcionamiento determinaremos la tasa de fallas del sistema, tasa que, desde ya, no es extrapolable a ningún otro tipo de situación.

Para determinar la tasa de fallas del sistema se suman las tasas de fallas de cada componente, incluyendo la tasa de fallas del circuito impreso, de interconexiones entre distintos dispositivos y las tasas de los conectores según su tipo.

8.1.1. Confiabilidad de la placa de potencia y uP de celda

Se detalla la lista de componentes asociada a la tasa de fallas propuesta por la HDBK217 en la siguiente tabla.

Calculo de fiablidad de la placa de celda											
Cantidad	Componente	Parametros							Formula	Fallas por unidad millon de	Suma de fallas
		C1	C2	πT	πL	πQ	πE		$(C1*\pi T+C2*\pi E)*\pi Q*\pi L$		
8	CD4094	0.02	0.072	0.6	1	2	2			0.234	1.872
1	ATmega16	0.06	0.19	0.6	1	2	2			0.624	0.624
		λb	πT	πA	πR	πS	πQ	πE	$\lambda b*\pi T*\pi A*\pi R*\pi S*\pi Q*\pi E$		
128	bta42	0.00074	2.5	0.7	0.8	1	5.5	6		0.0329	4.212
		λb	πR	πQ	πE				$\lambda b*\pi R*\pi Q*\pi E$		
64	R0805	0.0015	1	0.3	2					0.0009	0.0576
		λb	πCV	πQ	πE				$\lambda b*\pi CV*\pi Q*\pi E$		
3	Cap. Elec.	0.12	1.3	10	2					3.12	9.36
		λb	πK	πP	πE				$\lambda b*\pi K*\pi P*\pi E$		
64	Conexion lampara-potencia	0.00093	1.5	1	5					0.007	0.4464
1	Conexion uP-potencia	0.00093	1.4	4	5					0.026	0.026
		λb	πQ	πE					$\lambda b*\pi Q*\pi E$		
1	Cristal	0.026	2.1	3						0.1638	0.1638
Total de fallas por millon de horas para una celda											16.762
MTBF en años											6.8104

Notar que no se analizan absolutamente todos los materiales usados sino los mas relevantes ya que debido a que hay repeticiones masivas de unos pocos componentes y conectores, serán los dominantes del total de fallas y no será afectado por componentes aislados, salvo que estos sean extremadamente susceptibles a las fallas, que no es nuestro caso. Con lo cual ya que esto es solo una

aproximación, bastará con aplicar un pequeño factor de ponderación para tener en cuanta los componentes faltante. Por ejemplo en vez de una MTBF de 6.8 años se puede considerar 6 años para ser mas conservador.

Sin embargo en este análisis tampoco se contemplan las lamparas de filamento ya que no se esta de acuerdo con lo propuesto por la norma, que como no contempla lamparas de alta tensión (220), si se extrapolara hacia arriba, arroja un valor demasiado alto. No obstante, las lamparas tienen un tratamiento especial de mantenimiento preventivo, considerando validas las 1000horas de vida útil que indican los fabricante. Por lo tanto ponderando este valor, que se interpreta como que a las 1000h la lampara tiene un 90% de probabilidades de fallar, con un factor que depende del uso ¨dinámico¨ a las que se verán sometidos nos dará el tiempo para el recambio de todas las lamparas.

8.1.2. Confiabilidad de la placa del uP Central

Debido a la poca influencia de esta placa frente a la placa de celda no se realiza el análisis.

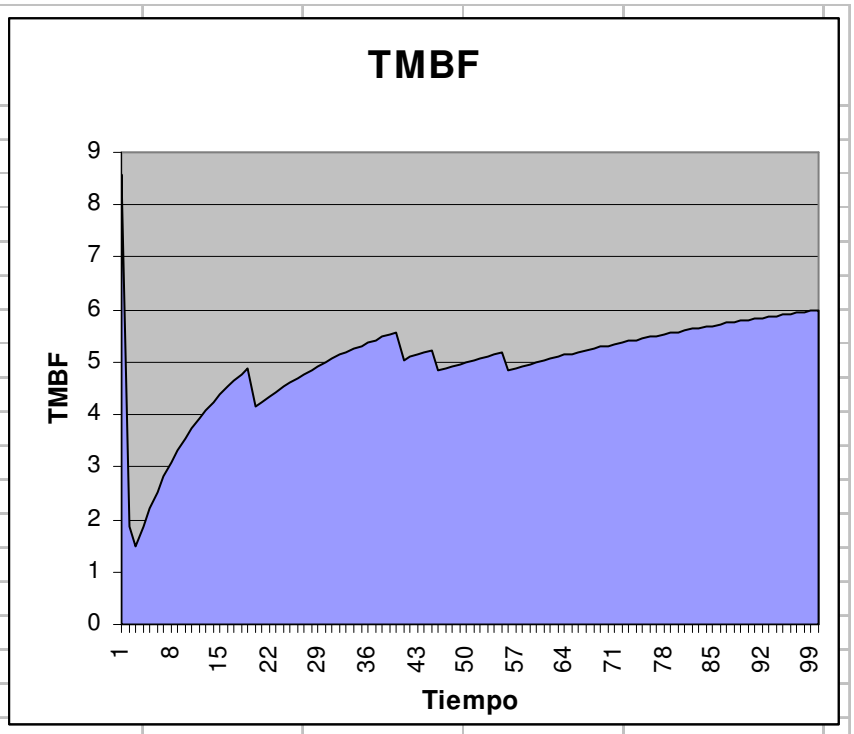
8.2. Confiabilidad de soft

8.2.1. Confiabilidad del soft del uP de celda

Para realizar el calculo de la confiabilidad se uso el modelo de Shooman. Para ello durante el desarrollo del soft se tomaron los tiempos entre errores y se confecciono la siguiente tabla resumida:

Tiempo	Errores	TMBF	R(t)	Hx/rx
1	0	8.6	0.8899	
2	1	1.9	0.3442	2
3	2	1.5	0.1331	1.5
14	2	4.2	0.0369	7
15	2	4.4	0.0328	7.5
16	2	4.5	0.0292	8
20	3	4.1	0.0080	6.666667
41	4	5.1	0.0003	10.25
42	4	5.1	0.0003	10.5
43	4	5.1	0.0002	10.75
44	4	5.2	0.0002	11
45	4	5.2	0.0002	11.25
46	5	4.8	0.0001	9.2
49	5	5.0	0.0001	9.8
50	5	5.0	0.0000	10
51	5	5.0	0.0000	10.2
56	6	4.9	0.0000	9.333333
99	6	6.0	0.0000	16.5
100	6	6.0	0.0000	16.66667

Et	-14
K	-0.83

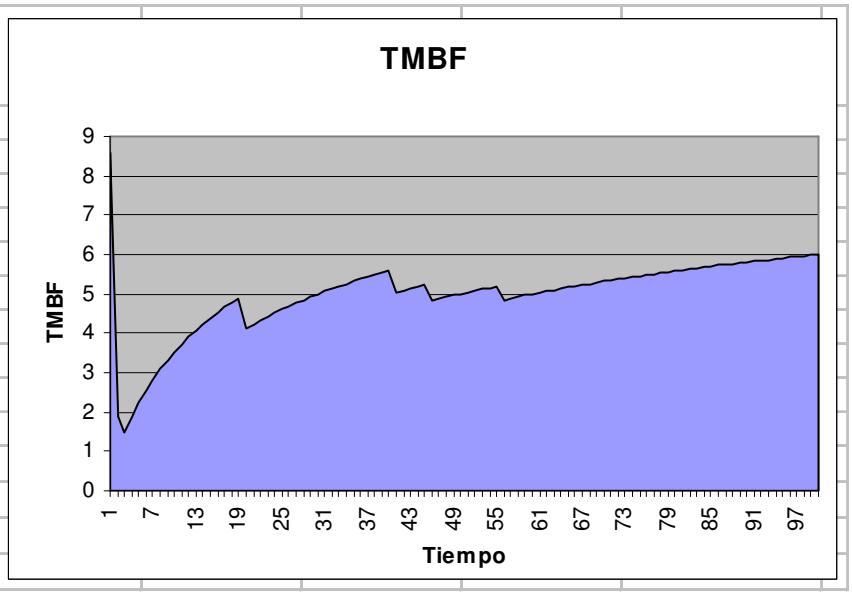


8.2.1. Confiabilidad del soft del uP Central

Usando los mismos modelos explicados en el apartado anterior se obtuvo el siguiente MTBF:

Tiempo	Errores	TMBF	R(t)	Hx/rx
1	0	4.1	0.7839	
2	1	0.6	0.0452	2
3	2	0.5	0.0026	1.5
4	3	0.5	0.0002	1.333333
11	5	0.7	0.0000	2.2
15	6	0.8	0.0000	2.5
26	7	1.1	0.0000	3.714286
50	8	1.5	0.0000	6.25
87	9	1.9	0.0000	9.666667
99	9	2.1	0.0000	11
100	6	2.5	0.0000	16.66667

Et	-9.33
K	-2.61



9. Conclusiones

9.1. Excelencias. Objetivos alcanzados.

Considerando que lo mas importante es satisfacer las necesidades del cliente, el objetivo mas importante logrado es permitir la implementacion de las escalas de grises para dar la capacidad de regular el consumo, limitar la temperatura, posibilidad de transmitir profundidad en la imagen y mayor caridad en la lectura. Piezas fundamentales desde el punto de vista del cliente.

Desde el punto de vista de optimización de recursos se cree que ciertas decisiones no fueron las mas optimas, pero se espera que las siguientes versiones puedan aprovechar la redundancia presente para brindar mas aplicaciones con el mismo costo.

Partiendo de la idea original de alcanzar un mínimo de 5 cuadros por segundo, se logro un rate de 100 cuadros por segundo para cada celda, y disminuyendo de acuerdo a la cantidad de celdas que deban ser actualizadas en cada cuadro. Este parámetro puede parecer sin sentido, pero se demostró que permite realizar efectos visuales imposibles de realizar con un rate de 5 cuadros por seg.

9.2. Fallos. Recomendaciones para futuros diseños

Se perdió mucho tiempo realizando conversiones manuales de bitmaps a código entendible por el sistema, hasta que se realizo un conversor automático. Se recomienda que siempre que haya tareas que involucren un calculo repetitivo que requiera mucho tiempo para la realización manual, se realice un soft adicional para la realización automática siempre que este soft lleve menos tiempo que digamos unas 10 conversiones manuales. Este parámetro dependerá de la aplicación.

Cuando se trabaja con altas tensiones (en este caso tensión de red 220V), se recomienda realizar con mucha atención las interfaces aisladas galvánicamente para la conexión de los dispositivos con la PC en el inicio del desarrollo, para evitar posibles accidentes.

No intente trabajar con componentes de montaje superficial sin las herramientas adecuadas, por muy caras que parezcan.

Habrá que tener en cuanta en futuros diseños el efecto de la corriente pico de la lampara en todos los componentes asociados. Por ejemplo el puente de diodos de entrada, los transistores de potencia, etc., ya que en este caso no se contemplo tan importante situación. Sin embargo experimentalmente se obtuvieron buenos resultados durante los meses que duro el desarrollo y es por eso que se supone que las corrientes están dentro de los márgenes. Pero faltan hacer todas las mediciones analizando detenidamente todos los casos críticos y sus posibles soluciones.

También habría que realizar pruebas relacionadas con incrementos inesperados de la tensión de red, bastante habitual en nuestro país, que por no contar con equipamiento no se realizo. También

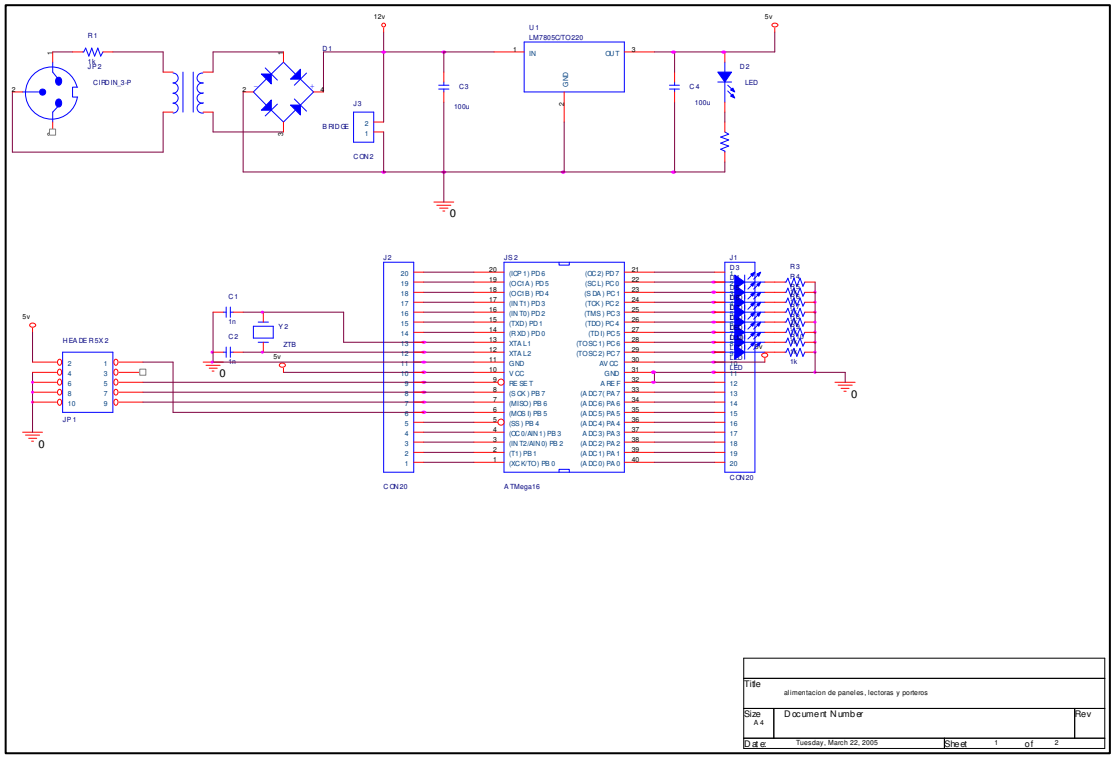
habría que estudiar el comportamiento frente a pequeños picos de alta tensión, fundamentalmente en la tensión de ruptura de los transistores de potencia que son los mas ajustados.

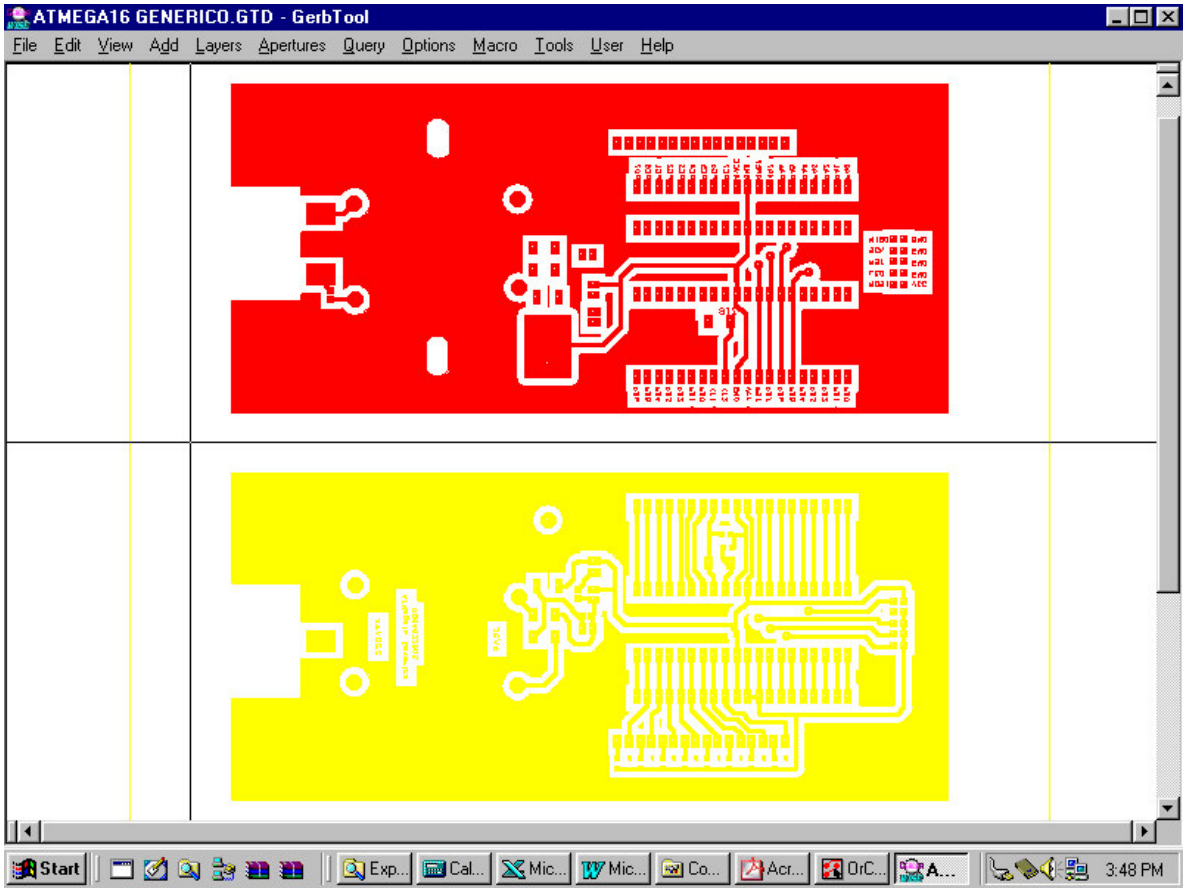
Si bien intuitivamente se realizaron esfuerzos para disminuir el EMI en la red, no se realizaron las debidas pruebas para corroborar dicha mejora, ni tampoco cuanto es el ruido residual y como afecta a los usuarios de la red. Habría que estudiar asimismo la influencia de estas variaciones de corriente de red en los circuitos digitales y uP que se encuentran sumergidos en estos.

10. Anexos: Técnicos. Justificativos. Descriptivos. Documentales

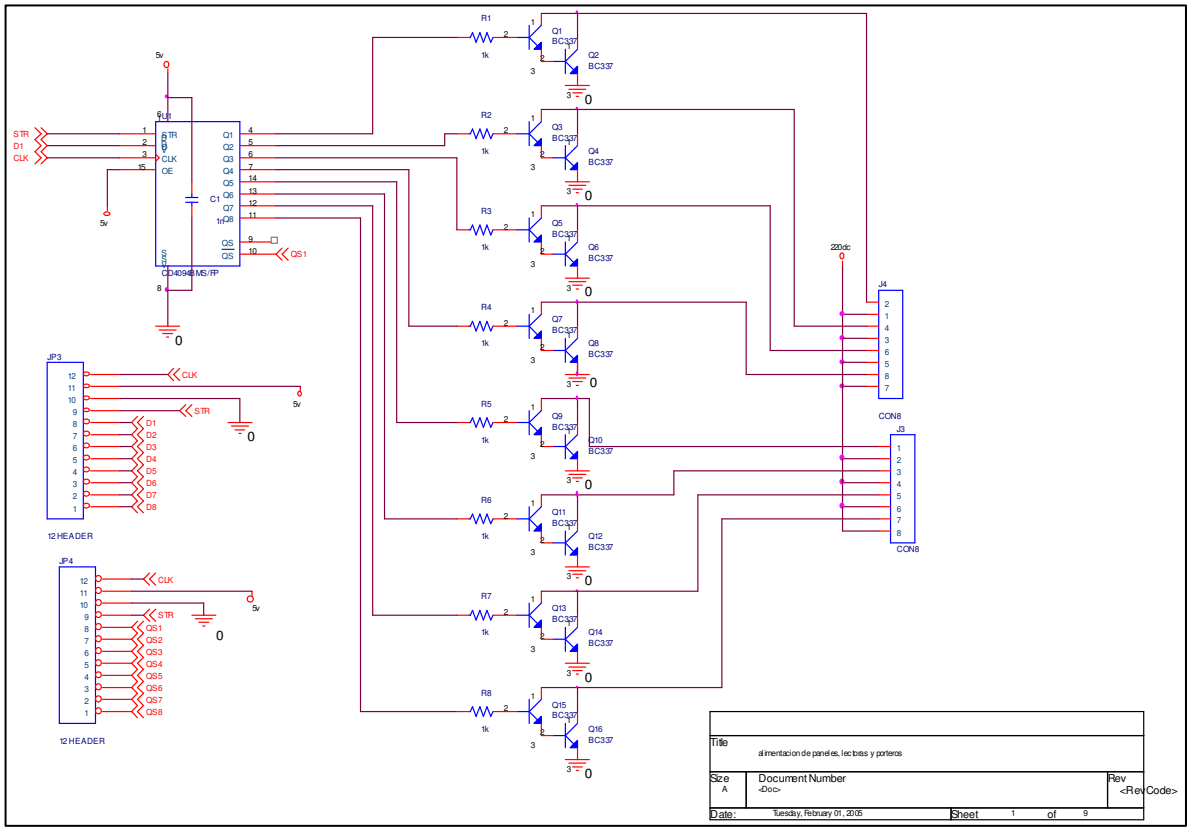
10.1. Planos

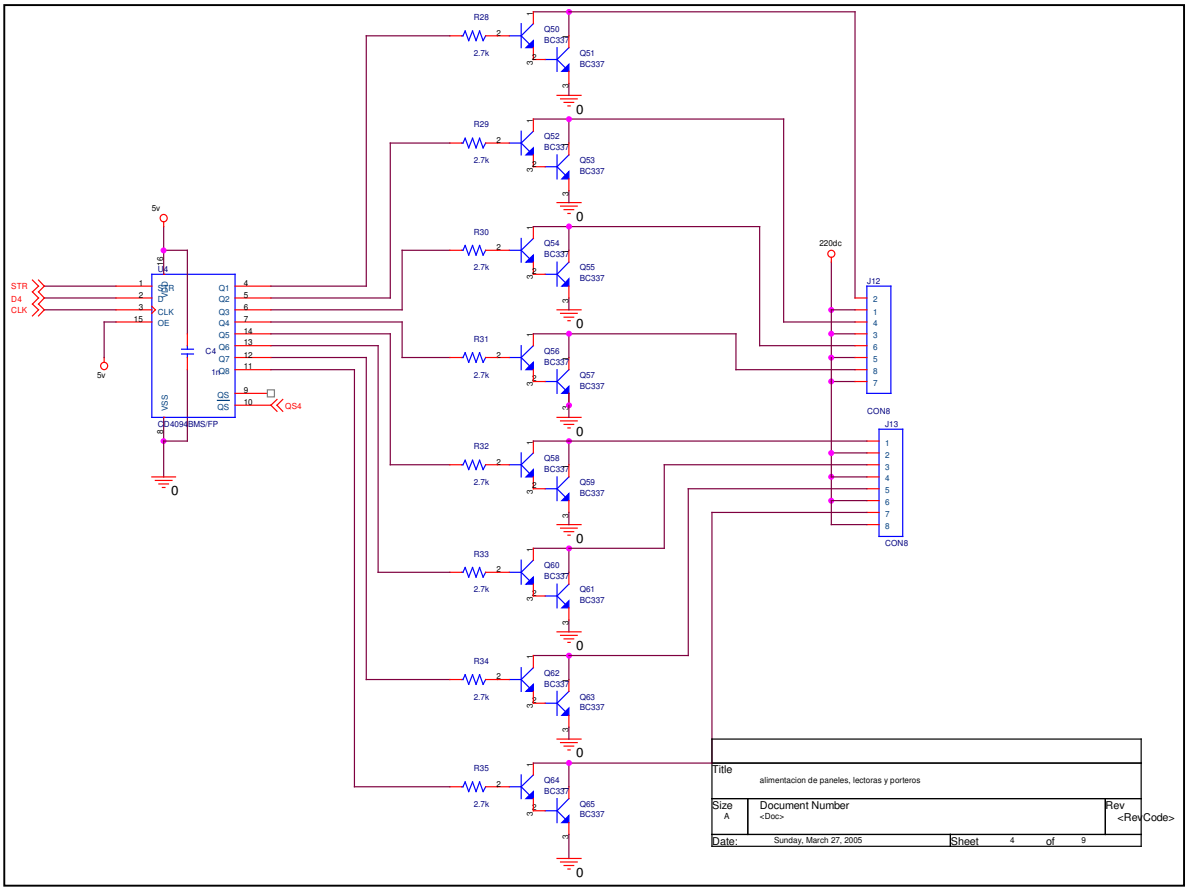
10.1.1. Esquemáticos y PCB de la placa del uP de celda

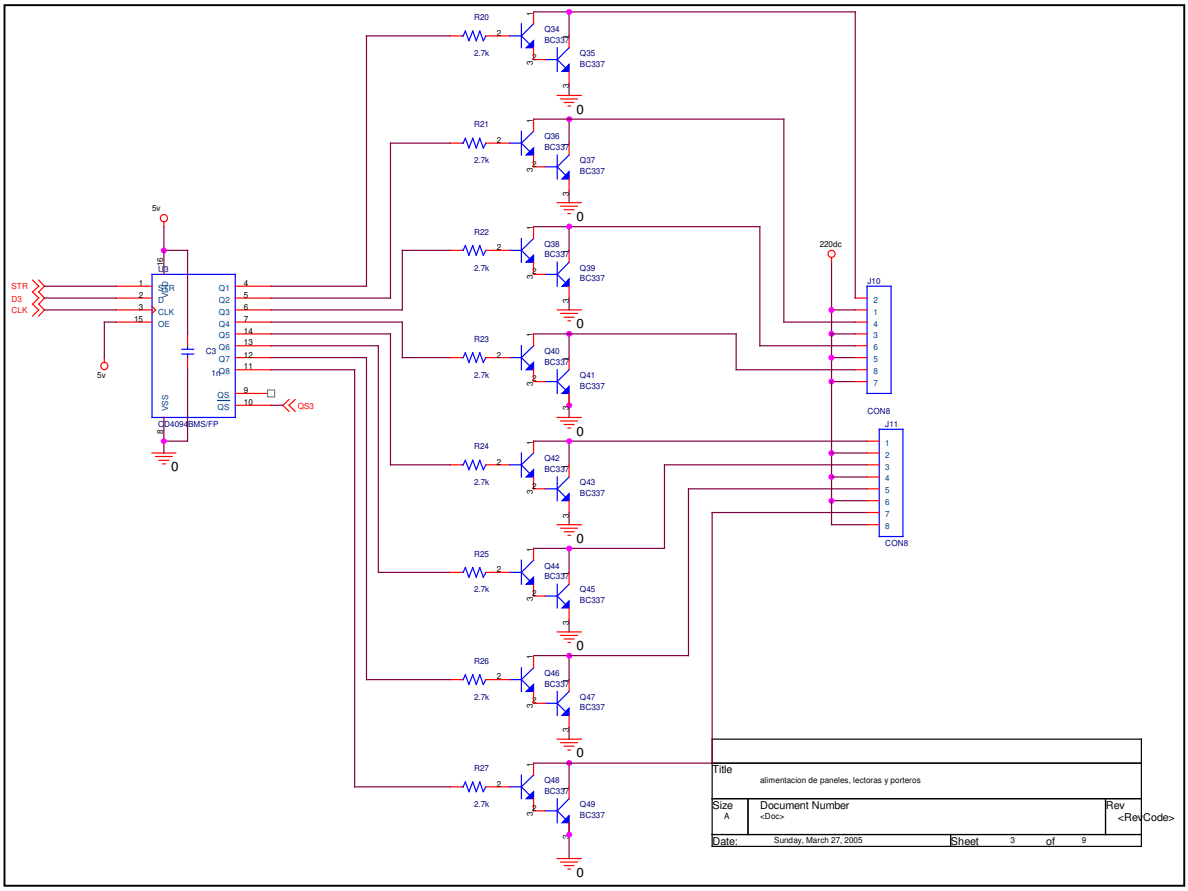


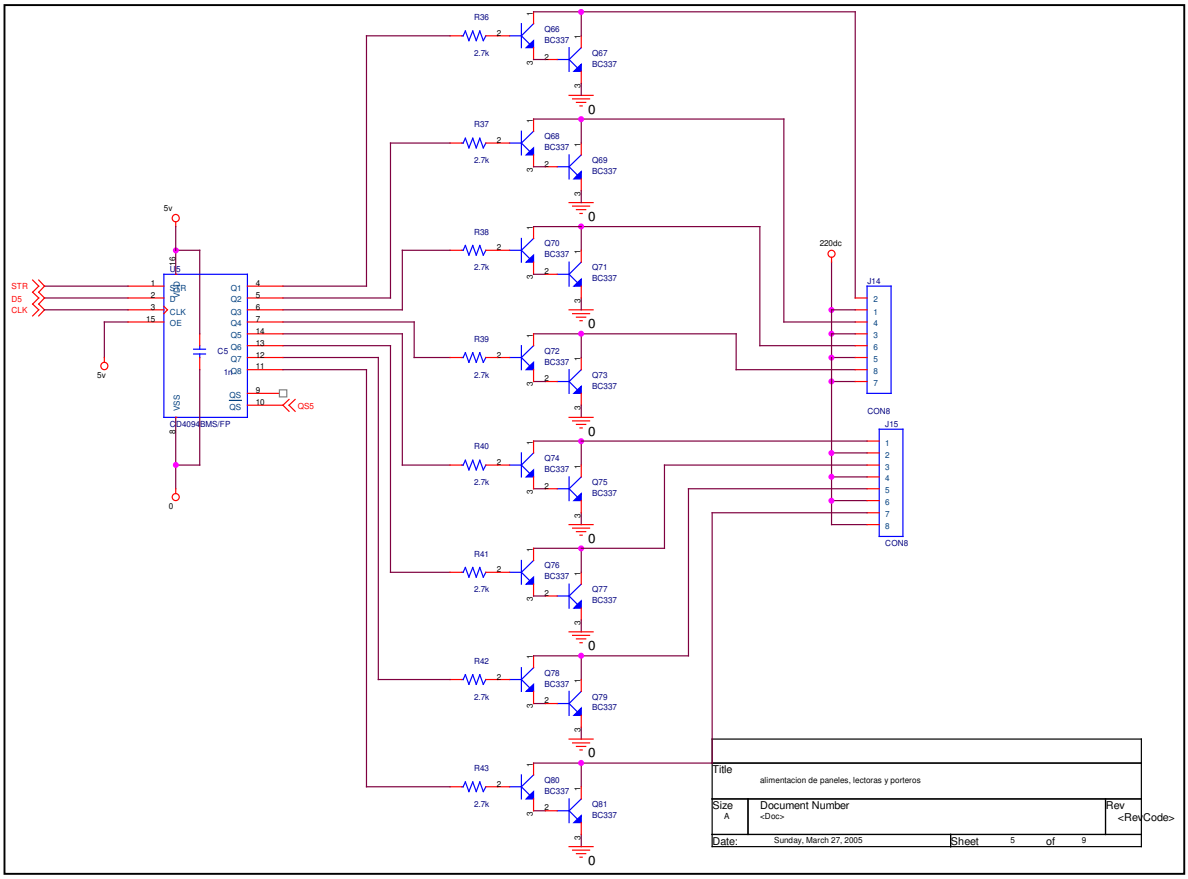


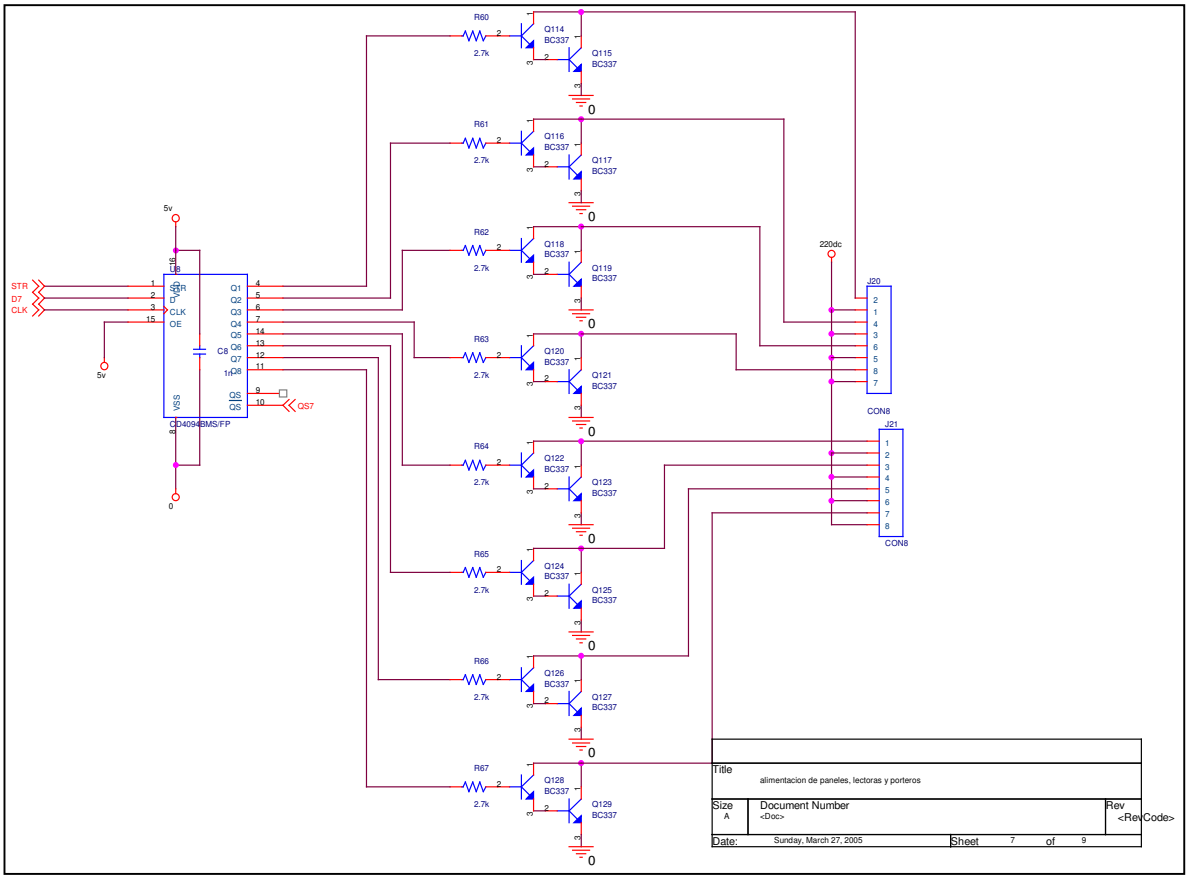
10.1.2. Esquemáticos y PCB de la placa de potencia de la celda

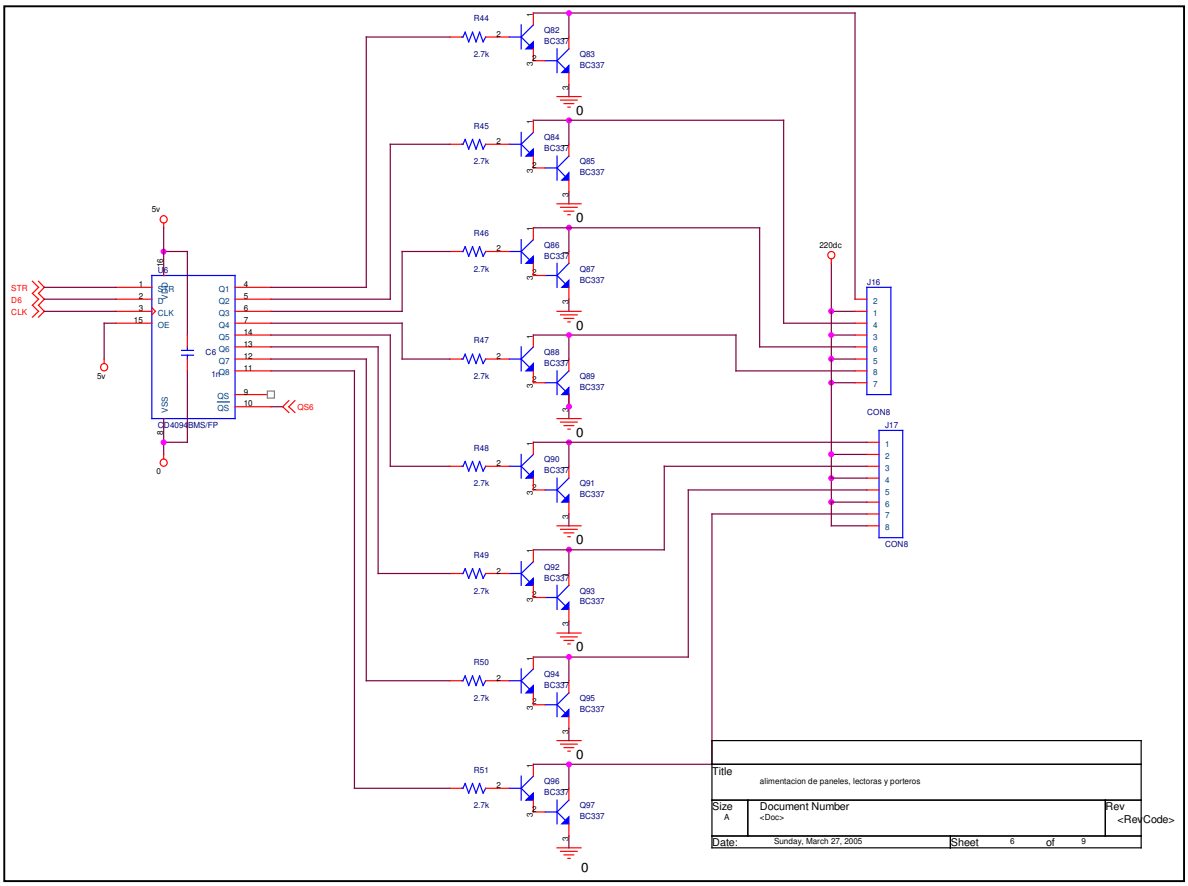


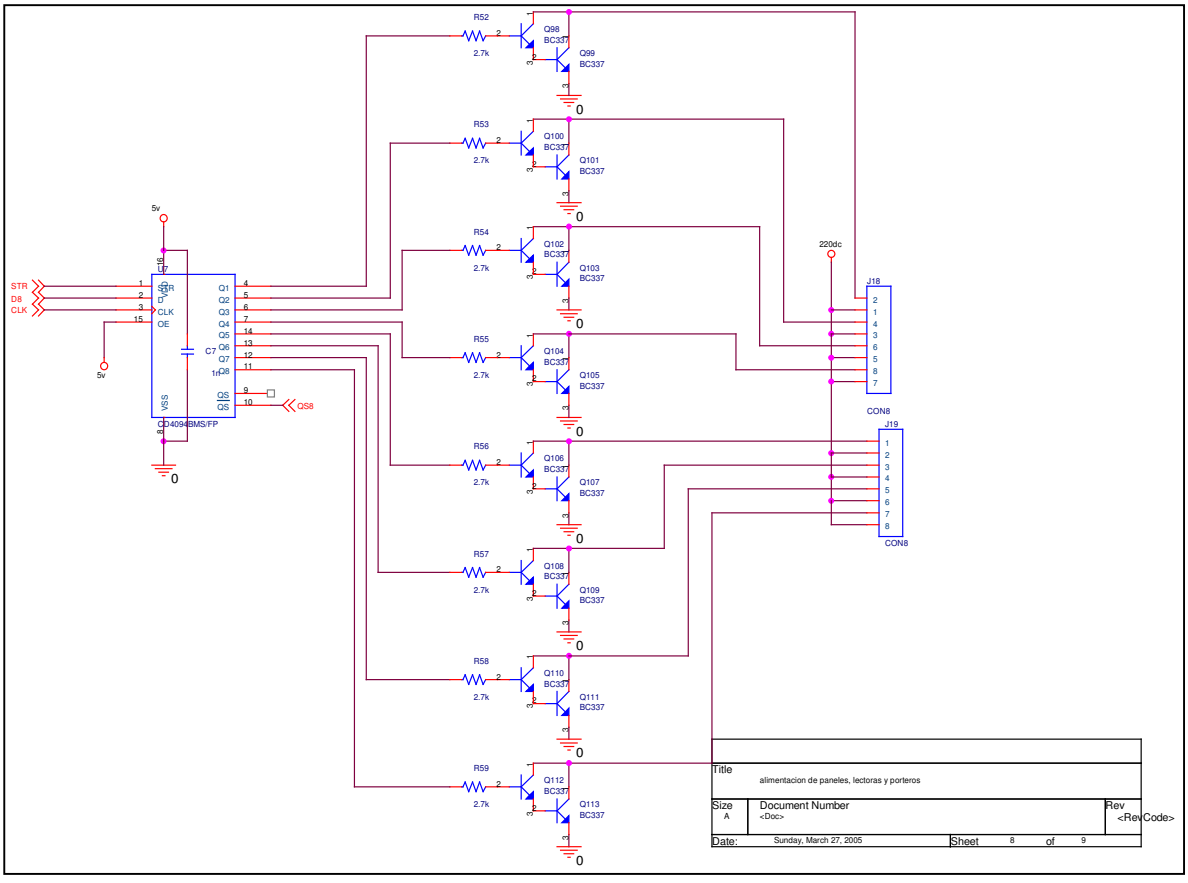

R28
Q50 BC337
Q51 BC337
2.7k
R29
Q52 BC337
Q53 BC337
R30
Q54 BC337
Q55 BC337
R31
Q56 BC337
Q57 BC337
R32
Q58 BC337
Q59 BC337
R33
Q60 BC337
Q61 BC337
R34
Q62 BC337
Q63 BC337
R35
Q64 BC337
Q65 BC337
5v
STR
D4
CLK
OE
C4
1uF
CD4094BMS/FP
QS4
220dc
J12
CON8
J13
Title
alimentacion de paneles, lectoras y porteros
Size
A
Document Number
<Doc>
Rev
<RevCode>
Date:
Sunday, March 27, 2005
Sheet
4
of
9

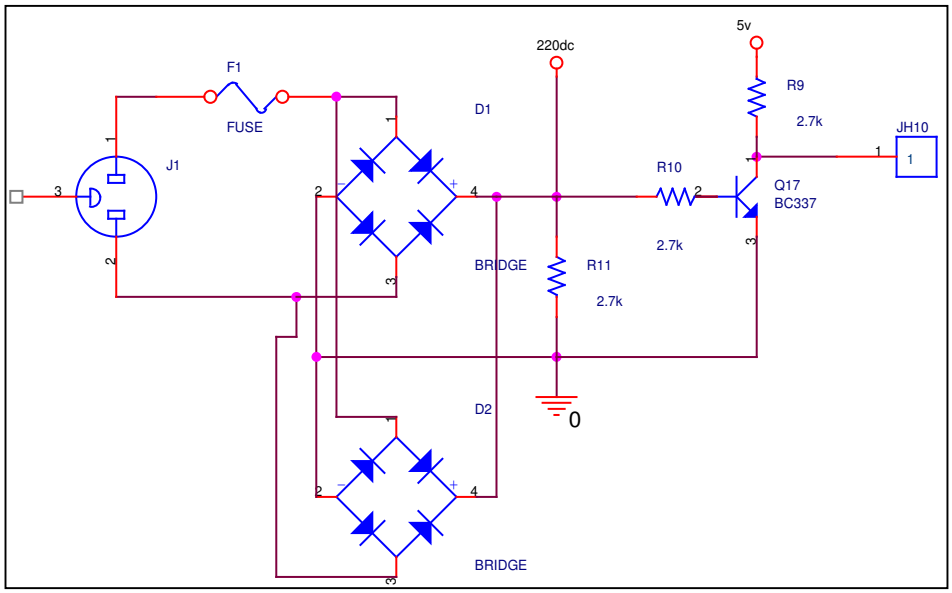

R20
Q34 BC337
Q35 BC337
2.7k
R21
Q36 BC337
Q37 BC337
R22
Q38 BC337
Q39 BC337
R23
Q40 BC337
Q41 BC337
R24
Q42 BC337
Q43 BC337
R25
Q44 BC337
Q45 BC337
R26
Q46 BC337
Q47 BC337
R27
Q48 BC337
Q49 BC337
5v
STR
D3
CLK
OE
C3
1uF
CD4094BMS/FP
QS3
220dc
J10
CON8
J11
Title
alimentacion de paneles, lectoras y porteros
Size
A
Document Number
<Doc>
Rev
<RevCode>
Date:
Sunday, March 27, 2005
Sheet
3
of
9

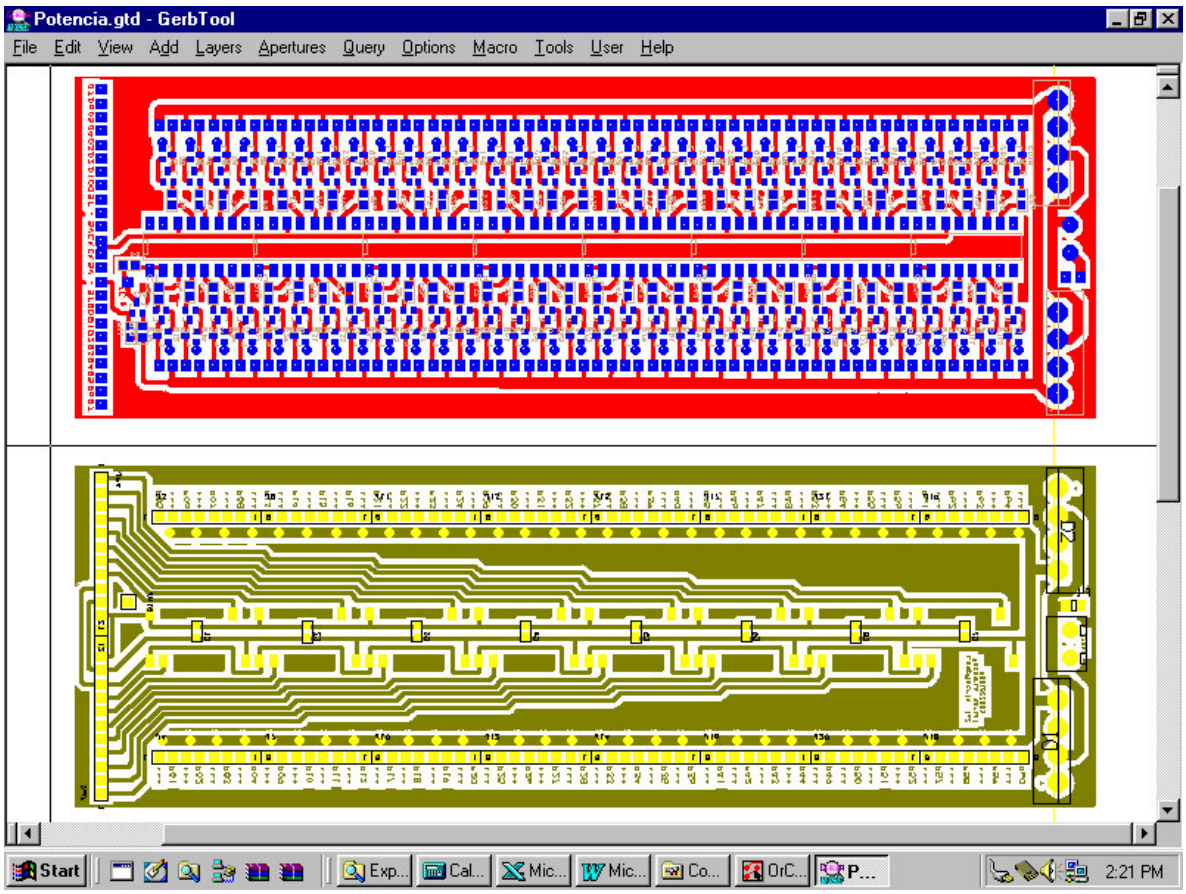

R36
Q66 BC337
Q67 BC337
R37
Q68 BC337
Q69 BC337
R38
Q70 BC337
Q71 BC337
R39
Q72 BC337
Q73 BC337
R40
Q74 BC337
Q75 BC337
R41
Q76 BC337
Q77 BC337
R42
Q78 BC337
Q79 BC337
R43
Q80 BC337
Q81 BC337
2.7k
5v
STR
D5
CLK
OE
C5
1nF
CD4094BMS/FP
QS5
220dc
J14
CON8
J15
Title
alimentacion de paneles, lectoras y porteros
Size A
Document Number
<Doc>
Rev
<RevCode>
Date: Sunday, March 27, 2005
Sheet 5 of 9


R60
Q114 BC337
Q115 BC337
R61
Q116 BC337
Q117 BC337
R62
Q118 BC337
Q119 BC337
R63
Q120 BC337
Q121 BC337
R64
Q122 BC337
Q123 BC337
R65
Q124 BC337
Q125 BC337
R66
Q126 BC337
Q127 BC337
R67
Q128 BC337
Q129 BC337
2.7k
5v
STR
D7
CLK
OE
C8
1nF
CD4094BMS/FP
QS7
220dc
J20
CON8
J21
Title
alimentacion de paneles, lectoras y porteros
Size A
Document Number
<Doc>
Rev
<RevCode>
Date: Sunday, March 27, 2005
Sheet 7 of 9


R44
Q82
BC337
Q83
BC337
R45
Q84
BC337
Q85
BC337
R46
Q86
BC337
Q87
BC337
R47
Q88
BC337
Q89
BC337
R48
Q90
BC337
Q91
BC337
R49
Q92
BC337
Q93
BC337
R50
Q94
BC337
Q95
BC337
R51
Q96
BC337
Q97
BC337
2.7k
5v
STR
D6
CLK
OE
C6
1n
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
QS
QS
QS6
CD4094BMS/FP
220dc
J16
CON8
J17
CON8
0
Title
alimentacion de paneles, lectoras y porteros
Size
A
Document Number
<Doc>
Rev
<RevCode>
Date:
Sunday, March 27, 2005
Sheet
6
of
9


R52
Q98
BC337
Q99
BC337
R53
Q100
BC337
Q101
BC337
R54
Q102
BC337
Q103
BC337
R55
Q104
BC337
Q105
BC337
R56
Q106
BC337
Q107
BC337
R57
Q108
BC337
Q109
BC337
R58
Q110
BC337
Q111
BC337
R59
Q112
BC337
Q113
BC337
2.7k
5v
STR
D8
CLK
OE
C7
1n
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
QS
QS
QS8
CD4094BMS/FP
220dc
J18
CON8
J19
CON8
0
Title
alimentacion de paneles, lectoras y porteros
Size
A
Document Number
<Doc>
Rev
<RevCode>
Date:
Sunday, March 27, 2005
Sheet
8
of
9





10.1.3. Esquemáticos de la placa del uP central

Se muestran al final del informe junto con las hojas de datos de los componentes.

10.2. Esquemas

No se detallan.

10.3. Listado de partes

10.3.1. Listado de partes de la placa de potencia

Bill Of Materials March 22,2005 14:10:33 Page1

Item	Quantity	Reference	Part	Package	Obs
1	8	C1,C2,C3,C4,C5,C6,C7,C8	capcitor	0.1uX16V 0805	
2	2	D1	BRIDGE	KBU10M	
3	1	F1	FUSE	10A	2cm
4	2	JP4,JP3	12 HEADER	100 mils	
5	16	J3,J4,J7,J8,J10,J11,J12, J13,J14,J15,J16,J17,J18, J19,J20,J21	CON8	100mils	
6	129	Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8, Q9,Q10,Q11,Q12,Q13,Q14, Q15,Q16,Q17,Q18,Q19,Q20, Q21,Q22,Q23,Q24,Q25,Q26, Q27,Q28,Q29,Q30,Q31,Q32, Q33,Q34,Q35,Q36,Q37,Q38, Q39,Q40,Q41,Q42,Q43,Q44, Q45,Q46,Q47,Q48,Q49,Q50, Q51,Q52,Q53,Q54,Q55,Q56, Q57,Q58,Q59,Q60,Q61,Q62, Q63,Q64,Q65,Q66,Q67,Q68, Q69,Q70,Q71,Q72,Q73,Q74, Q75,Q76,Q77,Q78,Q79,Q80, Q81,Q82,Q83,Q84,Q85,Q86, Q87,Q88,Q89,Q90,Q91,Q92, Q93,Q94,Q95,Q96,Q97,Q98, Q99,Q100,Q101,Q102,Q103, Q104,Q105,Q106,Q107,Q108, Q109,Q110,Q111,Q112,Q113, Q114,Q115,Q116,Q117,Q118, Q119,Q120,Q121,Q122,Q123, Q124,Q125,Q126,Q127,Q128, Q129		BTA42 SOT23	
7	67	R1,R2,R3,R4,R5,R6,R7,R8, R9,R10,R11,R12,R13,R14, R15,R16,R17,R18,R19,R20, R21,R22,R23,R24,R25,R26, R27,R28,R29,R30,R31,R32, R33,R34,R35,R36,R37,R38, R39,R40,R41,R42,R43,R44, R45,R46,R47,R48,R49,R50, R51,R52,R53,R54,R55,R56, R57,R58,R59,R60,R61,R62, R63,R64,R65,R66,R67		2.7K	0805
8	8	U1,U2,U3,U4,U5,U6,U7,U8		CD4094B SOIC16	

10.3.2. Listado de partes de la placa del uP de celda

Bill Of Materials March 22,2005 13:53:28 Page1

Item	Quantity	Reference	Part	Obs.

1	2	C1,C2	0.1uX16V 0805
2	2	C4,C3	100ux16V electrolitico
3	1	D1	BRIDGE KBU4M
4	1	JP1	HEADER 5X2 100mils
5	1	JP2	CIRDIN_3-P 100mils
6	1	JS2	ATMega16-16AI
7	2	J1,J2	CON20 100mils
8	1	U1	LM7805C/TO220
9	1	Y2	Cristal 16Mhz

10.3.3. Listado de partes de la placa del uP central

Este dispositivo es un único modulo preparado para el desarrollo que se identifica como RCM2200 y ya contiene todo el hardware necesario para esta aplicación.

10.4. Códigos de soft

Se detallaron en los puntos 5.2.4.1 y 5.2.4.2

10.5. Experiencias accesorias

No se detallan.

10.6. Hojas de datos de componentes

Se detallan a continuación la lista de archivos pdf's de los componentes mas importantes. Los mismos se encuentran impresos al final del informe

Atemga16.pdf
cd4094.pdf
mpsa42.pdf
rcm2200.pdf

10.7. Hojas de aplicación, etc.

Notas de ejemplos sobre el uso de TCP/IP se pueden encontrar en la pagina de rabbit, www.rabbitsemiconductor.com/documentation . Alli se puede extraer numerosos ejemplo sobre este protocolo y su utilizacion en esta plataforma.

11. Bibliografía

11.1. Libros.- Autor. Título. Editorial. Fecha

Philip T. Krein, "Elements of power electronics", Oxford University Press, 1998
Sklar, "Digital Comunications. Fundamentals and applications", Prentice Hall, 2001
Brian W. Kernigham, Dennis M Ritchie, "El lenguaje de programación C", Prentice Hall, 1991
Williams Stallings, "Counicaciones y redes de computadores",Prentice Hall, 2000

11.2. Hojas de datos de algunos de los componentes utilizados

Fin.