

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

PIDS: Sistema de información visual para pasajeros de Trenes Argentinos

Autor:

Ing. Carlos German Carreño Romano

Director:

Dr. Ing. Pablo Martín Gomez (UBA)

Jurados:

Nombre del jurado 1 (pertenencia)

Nombre del jurado 2 (pertenencia)

Nombre del jurado 3 (pertenencia)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,
entre marzo de 2020 y diciembre de 2023.*

Resumen

En este trabajo se aborda el desarrollo de un sistema de información visual para pasajeros para la empresa Trenes Argentinos. Las formaciones ferroviarias tienen carteles de matriz led dentro de los coches por donde se comunican mensajes tales como la próxima estación. Estos carteles forman parte de una red de comunicaciones que interconecta distintos sistemas de control dentro del tren. El sistema desarrollado en este trabajo puede recibir datos de la red de comunicaciones y controlar carteles de matriz led compatibles con los existentes. El sistema sigue una arquitectura orientada a eventos, haciendo uso de técnicas de concurrencia y patrones de software tales como objeto activo, máquinas de estado y gestión de memoria dinámica, sobre un sistema operativo de tiempo real. Se han realizado ensayos y mediciones en trenes operativos, que han proporcionado tramas de datos reales que fueron analizadas para usarse como entradas del sistema. En la región del AMBA, los trenes operan aproximadamente con 500 carteles del mismo modelo de forma simultánea. Estos suelen presentar fallas, y su reemplazo se ve dificultado por problemas de importación. Con este trabajo se pretende contribuir a la sustitución de las placas de control y al mantenimiento de las unidades, con el fin de extender la vida útil de los trenes.

Agradecimientos

En primer lugar, quiero agradecer al Dr. Ing. Pablo Gomez por su inagotable paciencia y excelente predisposición a lo largo de todo el tiempo que llevó concluir este trabajo. También quiero agradecer al Dr. Ing. Ariel Lutenberg por su empuje, su espíritu motivador y su actitud de concertación. Me gustaría destacar su liderazgo en la generación de propuestas concretas de vinculación entre la universidad y la industria a través de proyectos de desarrollo de tecnología. Quiero reconocer el valioso aporte de ambos y del cuerpo docente que han volcado a lo largo de los años un enorme y minucioso trabajo en el desarrollo de los contenidos del programa de la Carrera de Especialización en Sistemas Embebidos. Ha sido altamente satisfactorio completar el posgrado.

Asimismo, me gustaría agradecer al personal altamente calificado de la Gerencia de Material Rodante Eléctrico de la compañía Trenes Argentinos Operaciones (SOFSE), el Ing. Sergio Dieleke, y a todos los colaboradores que nos han brindado atención, seguridad y tiempo para realizar mediciones en las formaciones ferroviarias. Quiero hacer una mención especial de reconocimiento a Bruno Pilato por su colaboración desde los talleres de Castelar para realizar pruebas en conjunto durante la etapa de confinamiento debido a la pandemia.

Por último, también agradezco a Magui, que me brindó soporte en todo momento; a mi madre y a mi padre por enseñarme a valorar tanto la formación académica como la experiencia técnica con igual importancia; y a mis tutores y colegas de la empresa con quienes comparto el día a día e intercambio ideas sumamente útiles para pensar fuera de la caja.

Índice general

Resumen	I
1. Introducción general	1
1.1. Descripción del trabajo	1
1.2. Objetivos y alcance	2
1.3. Introducción a los sistemas de información visual	4
1.4. Estado del arte	5
2. Introducción específica	11
2.1. Red de comunicación del tren TCN	11
2.2. PIDS: Sistema de información visual para pasajeros	15
2.3. Carteles y controladoras de matrices led	15
3. Diseño e implementación	19
3.1. Requerimientos	19
3.2. Casos de uso	21
3.3. Arquitectura	22
3.3.1. Contexto	22
3.3.2. Diseño	24
3.4. Implementación	25
3.4.1. Organización del código fuente	25
3.4.2. Uso de recursos en RTOS	26
3.4.3. Patrones de software	30
Máquinas de estado	30
Objeto activo	32
3.4.4. Componentes del sistema	33
UART	34
Display led	36
4. Ensayos y resultados	41
4.1. Mediciones	41
4.2. Análisis de tramas	41
4.3. Pruebas en maqueta	41
Placa de control	41
4.4. Integración con red PIDS	41
4.5. Pruebas de campo	41
5. Conclusiones	45
5.1. Resultados obtenidos	46
5.2. Prospectivas	47
5.3. Bibliografía	48
Bibliografía	49

Capítulo 1

Introducción general

Este capítulo introduce al lector a la motivación original del trabajo realizado. Se explica el marco de investigación del que forma parte este proyecto, se presentan los sistemas de información visual y también el estado del arte en controladores de carteles led para estos sistemas.

1.1. Descripción del trabajo

En este trabajo se desarrolla el sistema de control para carteles de matriz led del sistema de información visual para pasajeros (PIDS) de Trenes Argentinos. Las formaciones de Trenes Argentinos cuentan con carteles de matriz led en sus coches, en el frente y en el contrafrente del tren. Todos estos carteles se interconectan a través de una red de comunicaciones propia del PIDS, por la que también se transmiten distintos tipos de datos, como datos de mapas led, mensajes de audio, información para los carteles led e incluso videos de cámaras de seguridad. La red PIDS convive y se interconecta con la red de comunicaciones del tren (TCN), que es una red estándar. Además, en los buses de datos de la red TCN, se transmiten datos de sensores de velocidad, de frenado y eventos que indican apertura o cierre de puertas, entre otros.

Los carteles led del sistema PIDS suelen presentar fallas a lo largo de su ciclo de vida, lo que implica tareas de mantenimiento, reparación o reposición. Si bien existen muchos tipos de carteles led disponibles comercialmente, la integración al sistema de comunicaciones del tren es propietaria del fabricante de trenes. Para el caso de Trenes Argentinos, el proveedor está radicado en China, lo que hace muy costoso y lento el proceso de reposición o mantenimiento de equipamiento. Por esta razón, el desarrollo local de tecnología para sistemas PIDS es estratégico, ya que no solo fomenta la industria local, sino que también extiende la vida útil de los trenes.

El eje de este trabajo es el desarrollo de un sistema a medida para Trenes Argentinos. La necesidad que prima es generar y brindar al personal de operaciones la información necesaria para construir y mantener los sistemas PIDS. Como resultado, este trabajo también tiene impacto directo en el pasajero, ya que contribuye a mejorar la calidad del servicio.

1.2. Objetivos y alcance

El contexto de este trabajo se encuentra enmarcado en un Proyecto de Desarrollo Estratégico (PDE) de la Secretaría de Ciencia y Técnica de la Universidad de Buenos Aires (UBACyT). El PDE se titula PDE_15_2020 - "Sistema de monitoreo y gestión de la red TCN en formaciones ferroviarias"^[1]. Las partes que se involucran y forman parte del equipo de trabajo en este proyecto son el Grupo de Investigación en Calidad y Seguridad de las Aplicaciones Ferroviarias (GICSA-FE), creado en 2017 en el marco del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) de la República Argentina, y la Operadora Ferroviaria Sociedad del Estado (SOFSE), también conocida como Trenes Argentinos Operaciones. El proyecto está orientado a cubrir necesidades tecnológicas concretas del sistema ferroviario argentino. Este tipo de proyectos son instrumentos de promoción científico-tecnológica que revalorizan e incrementan el aporte de la Universidad al desarrollo socioproyectivo.

El objetivo principal de este trabajo es diseñar e implementar un sistema de información visual para pasajeros a bordo del tren. El sistema de información visual para pasajeros existente consta de una parte manual y una automática. Cuando el conductor del tren asume su puesto para prestar servicio (o toma cabina), programa en una pantalla cuáles van a ser las estaciones cabecera. Los nombres de estas estaciones cabecera se visualizan en las marquesinas del frente y contrafrente del tren, como puede verse en la figura 1.1.



FIGURA 1.1. Foto de una formación operativa de Trenes Argentinos. Se observa el cartel de matriz led frontal que indica el destino Tigre.

En el interior de los coches, también hay carteles led. Estas marquesinas muestran mensajes a los pasajeros, como el nombre de la próxima estación o la estación arribada (por ejemplo, "Próxima estación Belgrano" o "Estás en estación Belgrano"). La información se presenta de manera automática basándose en variables de sistema que monitorean el detenimiento del tren, su velocidad y la apertura o cierre

de puertas. Toda esta información, junto con otros datos de supervisión y control, se transmite a través de una red de comunicación interna del tren que se denomina TCN (*Train Communication Network*) de acuerdo a las especificaciones definidas en el estándar[2]. Este estándar define para la red TCN dos buses jerárquicos donde se conectan los subsistemas electrónicos: el WTB (*Wire Train Bus*), que se utiliza para supervisar cambios topográficos en el tren y se conecta entre los vagones, y el MVB (*Multi-Vehicle Bus*) [3][4], donde se conectan los sensores y actuadores de cada coche, como los sistemas de frenos, los controles de las puertas, los medidores de velocidad, el sistema de información, entre otros. Ambos buses utilizan interfaces eléctricas que se basan en redes RS485.

El sistema propuesto en este trabajo tiene como objetivo captar los mensajes de información al pasajero que viajan por la red existente y presentarlos en un display led. El sistema se compone principalmente de cuatro partes:

- Display led.
- Placa de control.
- Cableado de interconexión.
- Firmware del sistema embebido

El diagrama del prototipo se presenta en la figura 1.2. El display led matricial representa los carteles de los coches del tren. La placa de control se debe poder conectar a la entrada con al bus de la red RS485 que corresponda y a la salida con un display led matricial.

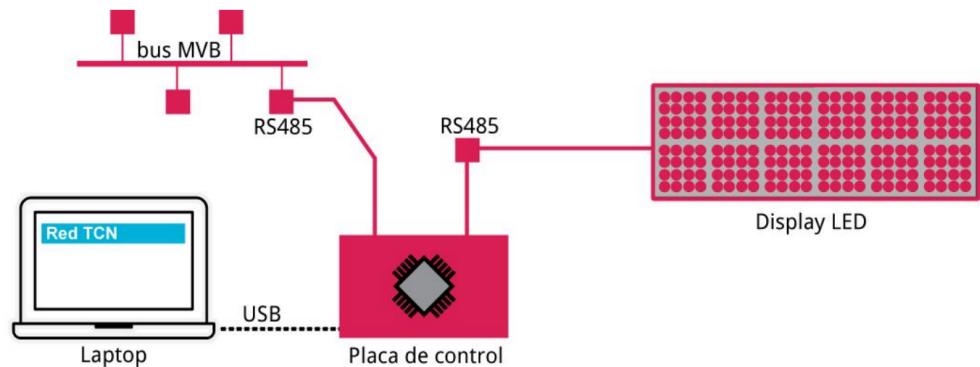


FIGURA 1.2. Diagrama de bloques del sistema embebido propuesto basado en la plataforma EDU-CIAA.

La placa de control está basada en la plataforma EDU-CIAA [5] o en alguna de las plataformas desarrolladas por el CONICET-GICSAFE. La conexión entre el display y la placa así como de la placa con la red TCN deberá ser compatible con el estándar RS-485, definido como capa física de la red TCN. El firmware a desarrollar se carga a la placa de control utilizando el puerto USB de una laptop. Este firmware es el responsable de leer los mensajes del sistema de información al pasajero y presentarlos en el display.

Las cualidades que debe satisfacer este proyecto son:

- Compatibilidad: debe ser compatible con el sistema PIDS existente.

- Practicidad: debe ser de fácil uso para el personal de Trenes Argentinos Operaciones.

Este proyecto permitirá implementar las funciones de visualización del sistema de información al pasajero sin depender del equipamiento existente. El sistema actual es un equipamiento integrado y propietario, y este trabajo busca separar algunas de sus funciones, específicamente las relacionadas con la visualización de información para pasajeros, y presentarlas en un display led genérico. Por otro lado, permitirá reponer los carteles que en la actualidad quedan fuera de servicio por fallas o pérdida del material original y no pueden ser reparados. De esta manera, el valor principal que aporta este trabajo es contribuir con la sustitución de repuestos faltantes por medio de desarrollo y reducir la dependencia tecnológica de la empresa con los fabricantes. Este proyecto tiene impacto directo en las formaciones ferroviarias existentes, que brindan servicio a los pasajeros todos los días.

1.3. Introducción a los sistemas de información visual

Los sistemas de información visual para pasajeros están presentes en diversas industrias y aplicaciones. Se encargan de proveer información a pasajeros en movimiento y tienen un rol fundamental en la industria del transporte.

Las personas se trasladan por tierra o aire usando automóviles, ómnibus, subtes, trenes o aviones, entre otros. Los sistemas de información visual presentan necesidades y soluciones distintas en cada caso. Por ejemplo, en autopistas, se comunican accidentes u obras viales en ejecución usando carteles gigantes con información en tiempo real. En aeropuertos, los pasajeros aéreos acceden a información sobre llegada, el estado o la salida de vuelos. A los pasajeros de ómnibus, les interesa conocer los tiempos de espera y las líneas en operación al llegar a una estación. Los pasajeros de trenes utilizan estos sistemas para conocer el destino o la próxima estación cuando están viajando. Estos carteles pueden estar ubicados al aire libre o dentro de un recinto, pero en general, requieren estar sincronizados con los vehículos en movimiento.

Los sistemas de información visual para pasajeros, constan principalmente de tres componentes: un sistema que genera datos, una red de transmisión y un sistema de pantallas. Las especificaciones de cada sistema varían según el ámbito de aplicación. Típicamente, en los trenes se requiere comunicación en tiempo real, lo que conlleva la adopción de protocolos de datos de tiempo real (RTP). En aplicaciones ferroviarias, también es fundamental garantizar la integridad, disponibilidad y confiabilidad de los datos. Además, existen otros requerimientos de carácter operativo, como el mantenimiento y la facilidad de instalación. Estos últimos aspectos son esenciales en la operación de una formación ferroviaria y tienen impacto directo en el ciclo de vida de un tren.

Los sistemas PIDS instalados en los trenes se interconectan con una red TCN. Ésta última, sigue un estándar que define tanto las interfaces eléctricas como los protocolos de comunicación. En la red TCN, se conectan dispositivos para el sensado

y control de frenos, de puertas, de monitoreo, entre otros, usando una arquitectura jerárquica de buses de datos. La red TCN representa un estándar robusto, maduro, probado y con gran adopción internacional. Sin embargo, los sistemas PIDS se presentan sin la necesidad de ser compatibles con los estándares de TCN, al menos hasta la revisión del año 2005. Existen diversas soluciones comerciales de sistemas PIDS, para aplicaciones de entretenimiento por ejemplo, pero se requiere de un trabajo de integración adicional para que funcionen en un tren.

En este trabajo se introduce una breve descripción de las redes TCN y su evolución en el tiempo. Para el caso de las formaciones de Trenes Argentinos, que forman el marco de este trabajo, se presenta también el detalle de interconexión TCN-PIDS, el desarrollo de un sistema de control para los carteles led del sistema PIDS y los resultados de las pruebas de campo realizadas en conjunto con la empresa Trenes Argentinos Operaciones (SOFSE). Se ha organizado esta memoria buscando acercar al lector primero los conceptos principales de la aplicación y luego el detalle técnico del diseño del sistema embebido propuesto.

1.4. Estado del arte

En esta breve sección, se resumen algunas características y aspectos comunes de los sistemas PIDS, tanto para sistemas ferroviarios como para sistemas de transporte integrados. En lugar de ser un estudio sistemático, la intención es guiar al lector en las consideraciones que fueron tenidas en cuenta en este trabajo. En primer lugar, se describe el rol que juegan estos sistemas y una noción de su mercado, mencionando aquellos proveedores que se consideraron relevantes por claridad en la información, marca global y diseño conceptual de la solución. Luego, se describen algunas soluciones comerciales innovadoras, y finalmente se presenta en tablas algunos aspectos técnicos comunes en distintas soluciones. Adicionalmente, se mencionan algunos trabajos académicos relevados. Se han elegido como dimensiones de análisis las funcionalidades y servicios que debe ofrecer un sistema PIDS, las características principales de la oferta de carteles electrónicos, y por último las características técnicas de las unidades de control.

El cliente de mayor impacto de los servicios que provee un sistema PIDS es la red de transporte (trenes, subtes, metros, ómnibus) de una gran ciudad, debido a su masividad. En términos generales, se observa que las empresas que proveen sistemas PIDS a las redes metropolitanas de transporte de las grandes ciudades lo hacen bajo formatos distintos. Algunas empresas instalan televisores o pantallas de video, otras carteles led, otras incluyen carteles impresos con algún elemento indicador tipo led, o bien leds en forma de flecha mezclándose con la señalización para indicar nombres de estaciones, pantallas led para desplegar publicidad entre mensajes, etcétera. En algunos países, se han realizado esfuerzos durante la última década para que los sistemas PIDS faciliten el acceso a la información del transporte para personas con discapacidades, movilidad reducida y de edad avanzada. Actualmente, los sistemas PIDS se diseñan teniendo en cuenta al pasajero en el centro de todo, buscando ofrecer servicios de información que mejoren la experiencia de viaje.

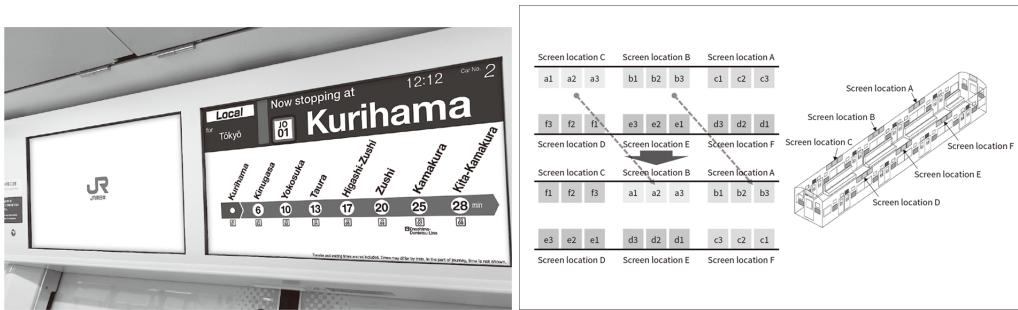


FIGURA 1.3. Solución de carteles para sistemas PIDS de Hitachi.
Consultado en [6]

Hitachi ofrece una solución para publicidad que consiste en tres pantallas en *array*, que se sincronizan para formar una sola y transmitir video con conectividad WiMAX. Cada uno de estos arreglos se posicionan arriba de las ventanas en ambos lados de los coches, alcanzando el despliegue de hasta dieciocho pantallas sincronizadas por coche, como se puede ver en la figura 1.3. De esta manera, logran transmitir varios mensajes distintos en simultáneo a los pasajeros sin que tengan que moverse de su asiento.

Por otro lado, Toshiba ofrece una solución que permite transmitir publicidad e información al pasajero en una misma pantalla LCD en simultáneo. La solución está centrada en la pantalla como dispositivo central, ofreciendo pantallas de 32 y 42, de 1920 x 540 píxeles, full color de hasta 16,7 millones de colores, con amplio ángulo de visión y de gran luminancia[7]. En la mayoría de los casos, las soluciones ofrecidas buscan cubrir tanto la demanda de un sistema PIDS como la oferta de publicidad de cara al pasajero, como es habitual en las estaciones y formaciones ferroviarias.



FIGURA 1.4. Solución de displays LCD para sistemas PIDS de Toshiba. Consultado en [7]

El grupo austriaco Trapeze [8] distingue cuatro tecnologías principales en sistemas PIDS: Led, LCD, canales móviles o apps, y e-ink que es una tecnología de LCD monocromo relativamente nueva. Al seleccionar carteles, es importante considerar varios factores, como los ángulos de visión, las condiciones del ambiente donde van instalados (por ejemplo, si estarán a la intemperie o requerirán visibilidad con la luz del sol), el tamaño o resolución de los caracteres en pantalla, la selección de colores y su relación con la capacidad estadística de visión de los pasajeros, el diseño mecánico, el acceso a controles para personas con movilidad reducida, la fuente de alimentación eléctrica y la capacidad de realizar actualizaciones de sistema de forma remota.



FIGURA 1.5. Sistema PIDS del proveedor austriaco Trapeze. Consultado en [8]

Además, se sugiere la importancia de la precisión en la información que ofrece como servicio el sistema PIDS. Si un pasajero recibe el número de andén incorrecto al llegar a la estación muy probablemente perderá el tren, resultando en una mala experiencia de viaje. La capacidad de interconectar el sistema PIDS con otros canales de información, especialmente en puntos nodales de transporte, también ofrece mayores beneficios al usuario final. Si un pasajero puede anticiparse y ver el tiempo estimado entre una línea de omnibus o de tren antes de llegar a la estación donde hace combinación, entonces puede tomar una mejor elección basada en datos ofrecidos por el sistema PIDS. Estos y otros aspectos de sistema centrados en el usuario se resumen en la tabla 1.1.

TABLA 1.1. Principales aspectos y servicios asociados a sistemas PIDS¹

Conectividad	RS-485, serial, USB. Ethernet, fibra óptica, coaxil. Wi-Fi, WiMax, GPS. 2G / 3G / 4G / 5G.
Interconexión	App del tren, horarios programados. Ómnibus, información multinodal. Portales de noticias, publicidad. Canal de información estatal.
Accesibilidad	Información por audio, ángulos de visión de los carteles. Facilidades para personas en sillas de ruedas. Facilidades para personas de edad avanzada. Correcto y cuidado sistema de señalización.
Información	Estimación de tiempos y aviso de cortes en tiempo real. Correcto trackeo de vehículos y conexiones. Mensajes de alerta o precauciones, números de emergencia.
Mantenibilidad	Fácil instalación, bajo costo de reposición. Consumo eléctrico. Actualizaciones de software.

Desde el punto de vista centrado en los carteles, se consideran varias especificaciones importantes de los sistemas PIDS, como las dimensiones del cartel, la densidad de píxeles por unidad de área, la cantidad de colores o leds por píxel,

¹Fuente: elaboración propia del autor.

los niveles de intensidad lumínica, el brillo y contraste, la potencia eléctrica. Entre estas especificaciones típicas de los carteles de los sistemas PIDS, el ángulo de visión es una de las variables más consideradas ya que en sistemas PIDS implican el alcance a mayor cantidad de pasajeros de la información en pantalla. En la tabla 1.2 se presenta un resumen de estas características. Las fuentes consultadas para la elaboración de esta tabla son diversos portales internacionales de distribución de componentes electrónicos.

TABLA 1.2. Principales características de displays para sistemas PIDS²

Display	led matricial	led RGB	TFT LCD	LCD RGB
Colores	monocromo bicolor tricolor multicolor (<10 colores)	desde 256 hasta 16,7 M (típicamente)	hasta 16,7 M	16.7M (típicamente) 1,000 M
Ángulo de visión	110°	160°	120°-140°	178°
Intensidad cd/m²	450	1500-2000	350	900
Densidad de píxeles	3,9 k 27,7 k 110 k *	P16: 3,9 k P12: 6,9 k P10: 10 k P8: 15,6 k P6: 27,7 k P5: 40 k P4: 62,5 k P3: 111 k P2.5: 160 k P2: 250 k *	29 M/m ² Pixel size 179 x 192 um	4,26 M/m ² Pixel size 484 x 484 um 29 M/m ² Pixel size 179 x 192 um
Potencia	10 W	15-316 W	20 W	25 W

Cada cartel requiere de un controlador como interfaz para procesar información y la codifica según la lógica que requiera el tipo de cartel. Los controladores de los carteles de matriz led suelen basarse en circuitos digitales, en microcontroladores de 8, 16 o 32 bits o en FPGA. Las tasas de transmisión de datos requieren señales de clock que pueden variar desde algunos kHz hasta cientos de MHz. Los tamaños del buffer de memoria depende de la cantidad de píxeles que tenga la pantalla. Las interfaces físicas pueden ser periféricos de un microcontrolador, un pin de propósito general o bien puertos USB o HDMI. Los carteles LCD en muchos casos requieren de la transmisión de señales de video. Esto implica mayores costos de implementación que la alternativa led, pero también mayor versatilidad en la programación de contenidos. En la tabla 1.3 se resumen algunas características principales de los requerimientos de los controladores.

²Fuente: elaboración propia del autor.

³Fuente: elaboración propia del autor.

TABLA 1.3. Principales características de controladores de uso general para aplicaciones PIDS³

Unidad de procesamiento	MCU 8/16/32 bits	FPGA / ASIC / DSP	CPU / DSP
Clock	1-200 MHz	10-250 MHz	1-3 GHz
Memory buffer	1 kB	1-512 MB	1-10 GB
Conectividad	UART (1-4) USB (1-2) RS485 GPIO (1-20) Ethernet	Pmod I/O pins (20-800) Ethernet USB	USB VGA HDMI DVI display Port PCI / PCI-E Ethernet
Programación	C / C++/ Assembly	VHDL, Verilog, XML	C, C++, Java, Python, XML

Una vez instalados, los sistemas PIDS suelen requerir mantenimiento. Muchas veces hay fallas de hardware, como por ejemplo leds que dejan de funcionar, una fuente de alimentación o una memoria que se debe reemplazar. Otras veces se requieren cambios en el software, por ejemplo actualizar el contenido de un mensaje o bien cambiarlo. Los atributos de mantenibilidad, versatilidad, modularidad y confiabilidad en la implementación pueden tener un impacto económico relevante en la operación de un servicio de transporte. Para líneas de trenes que cuentan con muchas formaciones ferroviarias operando en simultáneo, las tareas de actualización pueden ser muy intensivas en términos de horas de trabajo y requerir también capacitaciones técnicas periódicas al personal de mantenimiento. Incluso no todos los dispositivos pueden recibir actualizaciones en producción, esto es, en la locación física donde funcionan. En muchos casos es necesario desinstalarlos, llevarlos a un centro técnico y actualizarlos fuera de operación, lo que requiere de ventanas de mantenimiento y de tiempos reducidos para realizar tareas que pueden ser susceptibles a errores. Otra forma es enviar un técnico al sitio que pueda conectar algún periférico y actualizar manualmente cada dispositivo.

De los trabajos académicos relevados se mencionan aquellos con propuestas del sistema de control que representan distintas tecnologías. En [9] se utiliza el chip AT89C52 para enviar caracteres chinos sobre matrices de 32 x 192 leds de un solo color; en [10] se implementa una pantalla led RGB de 320 x 240 píxeles que rota 360°, permitiendo visualizar imágenes en color por persistencia de visión; en [11] se desarrollan algoritmos sobre FPGA usando búferes de datos para controlar una pantalla led de 160 x 32 píxeles alcanzando 32.768 colores; en [12] se presenta el control de un micro display de transistores de película delgada (TFT) usando modulación por ancho de pulso (PWM) alcanzando 256 niveles de color a una frecuencia de refresco de 60 Hz, basado también en FPGA; en [13] se presenta el control de píxeles virtuales para matrices led multicolor usando flip-flops tipo D.

En el diseño e implementación del presente trabajo, los carteles son de matriz led de un solo color y de distintas dimensiones (8 x 64, 32 x 64, 32 x 128). El control de los carteles tiene como factor común el uso del conjunto de chips digitales

74HC138, 74HC595 y 74HC245. La topología permite interconectar paneles en serie para construir carteles led de distinto tamaño usando la misma lógica de control.

Capítulo 2

Introducción específica

En este capítulo se introduce la terminología específica de la red de comunicaciones del tren (TCN) y del sistema de información visual al pasajero (PIDS). La arquitectura del sistema y sus módulos principales son presentados con una descripción general. Además, se introduce el sistema de carteles led de Trenes Argentinos, que son los que brindan información al pasajero en las formaciones ferroviarias operativas de la empresa.

2.1. Red de comunicación del tren TCN

La red de comunicaciones del tren TCN presenta una arquitectura de buses jerárquicos de dos niveles, y está especificada en el estándar IEC-61375-1 [2] de la *International Electrotechnical Commission* (IEC). En la figura 2.1 se presenta el esquema de conexión de los buses de datos de la TCN, el *Wire Train Bus* (WTB) que se encarga de las comunicaciones entre coches a través de nodos con redundancia física, y el *Multi Vehicle Bus* (MVB), al que se conectan los dispositivos de cada coche. Cada bus de datos tiene su especificación de detalle en las normas de la IEC [3] y [4].

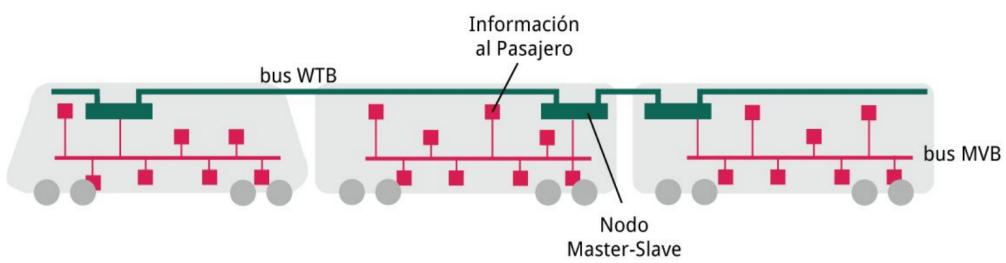


FIGURA 2.1. Diagrama del tren y de los buses WTB/MVB de la red TCN.

En la figura 2.2 se presenta la topología de la red TCN para los trenes de SOFSE. Observar que, leyendo la figura desde arriba hacia abajo, el esquema sugiere la ubicación en los coches de los distintos módulos. En los trenes de SOFSE, existen armarios con racks en los extremos de los coches, donde se alojan los dispositivos de la red.

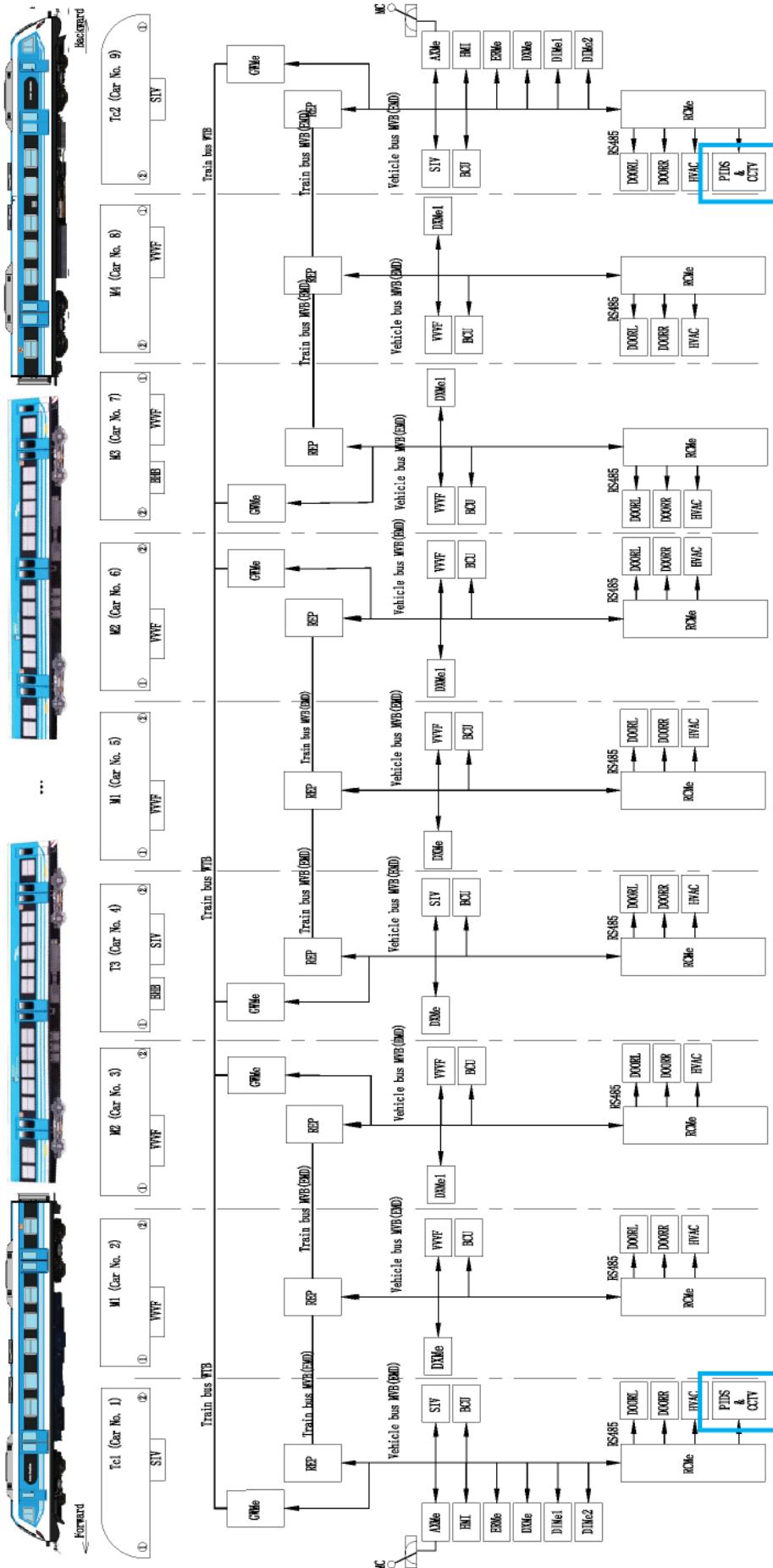


FIGURA 2.2. diagrama de la red TCN de Trenes Argentinos. Plano cortesía SOFSE, editado por el autor.

En la figura 2.2 también se puede observar que varios de los módulos se repiten en cada coche y, resaltado en celeste, se señala la ubicación del sistema PIDS y del circuito cerrado de TV (CCTV) en los coches cabecera. Los buses MVB y WTB también están representados en la topología a lo largo del tren. El nodo *Master-Slave* que interconecta los buses está representado por un bloque denominado GWMe, *Vehicle control module* según la nomenclatura oficial, y funciona de *gateway* entre ambos buses. Siguiendo la cascada de bloques, todos los módulos que le siguen a los bloques REP (*Trunk module*) están conectados al bus MVB. Algunos de los dispositivos y sistemas conectados al MVB que se pueden observar son: el control de puertas (DOORL/R), el aire acondicionado (HVAC), el sistema de tracción (VVVF), el sistema de control de frenos (BCU), entre otros.

En las figuras 2.3 y 2.4 se presentan fotografías de los armarios de los trenes donde se puede ver la disposición de los equipos en los racks de salón. Los módulos negros que se observan en la figura 2.3 se denominan DIMe, DXMe y RCMe, y son parte de la solución TCN del Instituto Zhuzhou, denominada Distribute Train Electric Control System (DTECS) [14].

En la parte inferior del rack, se puede observar un bloque de módulos grises señalados como parte del sistema PIDS. El mapa de recorrido led (LMDU), los carteles de matriz led (IDU), los parlantes (LSP) y las cámaras de video (CCTV), son parte del PIDS, que también se interconectan al bus MVB. Acorde a la topología, el PIDS se conecta a un módulo denominado RCMe, a través de una red RS485. Este último bloque se corresponde con el módulo RCMe, recuadrado en celeste en la figura 2.3. Según lo relevado en las formaciones de Trenes Argentinos, y también en el plano de la red TCN, el PIDS tiene su propia red RS485, aparte de la que corresponde a la interconexión con el MVB. La figura 2.4 es una fotografía en detalle de la unidad de rack donde están instalados los módulos de hardware del PIDS.

Las formaciones existentes de SOFSE fueron adquiridas durante los años 2013-2014. Si bien se observa que las redes RS485 son parte fundamental de la red TCN en estas formaciones, existe una nueva generación de redes TCN basadas en redes *Ethernet* que se comenzó a especificar a partir del año 2008 por la IEC y que se expuso en demostraciones de performance recién a mediados de 2015. En el nuevo estándar de TCN basado en redes *Ethernet* de tiempo real, se definen los análogos a los buses WTB y MVB, los buses ETB (*Ethernet Train Backbone*) y ECN (*Ethernet Consist Network*). Las empresas Siemens, Bombardier, Alstom, Mitsubishi y el Instituto Zhuzhou formaron parte del desarrollo de los nuevos estándares, saliendo al mercado entre los años 2014-2015. En comparación con la versión WTB/MVB, ETB/ECN soporta datos multimedia incluyendo video. Así, en esta nueva generación de TCN *Ethernet*, tanto el CCTV como el PIDS están especificados dentro de las normas IEC62580-2 y IEC62580-4, respectivamente. Sin embargo, el PIDS y el CCTV ya existían en las soluciones de mercado ofrecidas por los fabricantes, como pudimos observar en los coches de SOFSE en los que estos sistemas se basan en redes RS485, aunque no estuvieran especificados por un estándar.

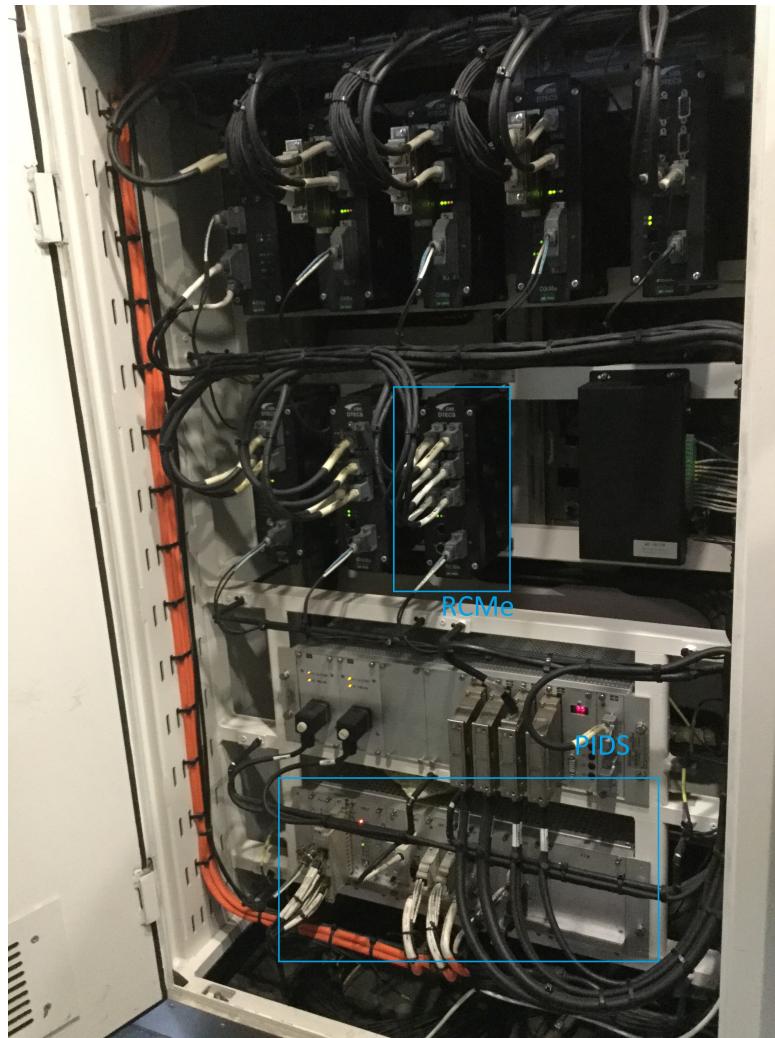


FIGURA 2.3. Fotografía del rack de salón de una formación de Trenes Argentinos.

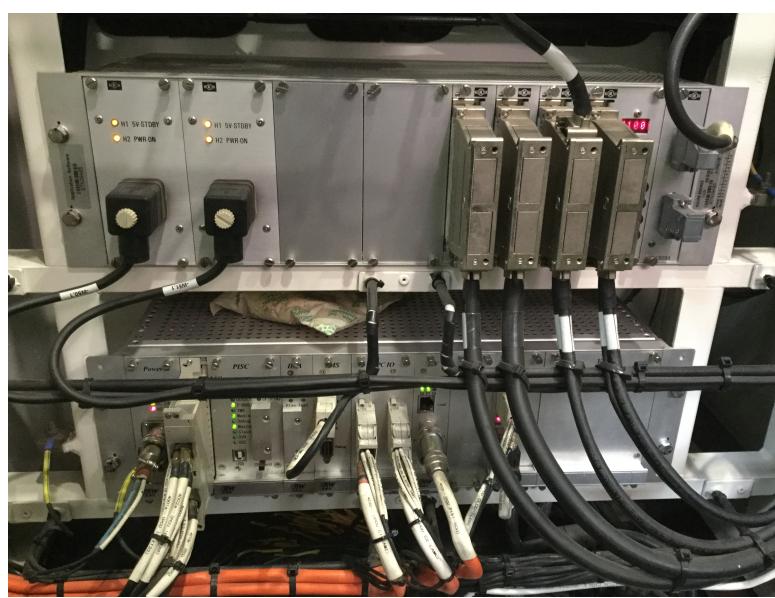


FIGURA 2.4. Fotografía de las unidades de rack del sistema PIDS en el rack de salón.

2.2. PIDS: Sistema de información visual para pasajeros

El PIDS de Trenes Argentinos es una solución propietaria, y no está especificada en ningún estándar. Este sistema integra un circuito cerrado de cámaras de TV (CCTV), un sistema de audio (LSP), mapas de recorrido (LDMU) y carteles de matriz led (IDU). En la figura 2.5 se presenta un diagrama de su arquitectura. Se puede observar un bus de datos ubicado en la parte superior, que recorre todo el largo del plano conectándose con todos los bloques. Este bus está formado por los cables 4001, 4002 y 4003, formando una red RS485 a la que se conectan distintos módulos conectorizados. Existen dos módulos denominados DACU y FDU, encargados del micrófono, del audio, de la comunicación con los sistemas CAM T, CAM C, TCMS, OCC y de la pantalla táctil LCD del conductor (TLCD), a través del módulo PCU. Al bus RS485 también se conecta el módulo SCU, que por su ubicación central y conexionado funciona como un *hub* de dispositivos. Al SCU se conecta otro conjunto de módulos: PECU1 y PECU2 indicando referencia de los laterales del tren, cámaras CAM1 y CAM2, y cuatro arreglos serie de dispositivos entre los que se encuentran los carteles de matriz led, los mapas de recorridos y parlantes. Dos de los arreglos serie integran los bloques IDU (carteles de matriz led) y LDMU (mapas de recorrido led), y los otros dos arreglos serie integran los bloques LSP o parlantes. Se puede observar que existen dos unidades IDU: IDU1 al comenzar y IDU2 al finalizar el arreglo serie. La nomenclatura corresponde con la ubicación de los carteles de matriz led en el salón de cada coche.

Los módulos IDU corresponden a las unidades por las que se transmiten datos a los carteles de matriz led. En el SCU termina un conjunto de cables mediante un conector Harting IO/48 P, como se puede observar en la figura 2.6. Dentro del mismo, el bus RS485 que conecta el SCU con las unidades IDU está compuesto por los cables nomenclados como 4330a (RS485A), 4330b (RS485B) y 4330c (RS485C). Este punto de conexión resulta importante porque, según lo relevado en las formaciones de SOFSE, es el punto de conexión física de los carteles de matriz led de salón.

2.3. Carteles y controladoras de matrices led

En la sección del estado del arte de sistemas de información visual, se han presentado distintos tipos de carteles led y controladores asociados. Los carteles de matriz led de las formaciones de Trenes Argentinos están compuestos de módulos de 8x8 leds, que forman carteles de 48 módulos monocromo (rojo) para los carteles de salón y de 12 módulos bicolor (rojo-verde) de mayor tamaño para los carteles de frente y contrafrente del tren. Las placas de control de estos carteles tienen, por un lado, conexión eléctrica con la red de 110 VDC del tren, y por otro, un circuito de datos de 5 VDC que se comunica con el conjunto de chips digitales de los módulos 8x8. La figura 2.7 es una fotografía de un cartel de matriz led de salón inicializándose para brindar servicio al pasajero.

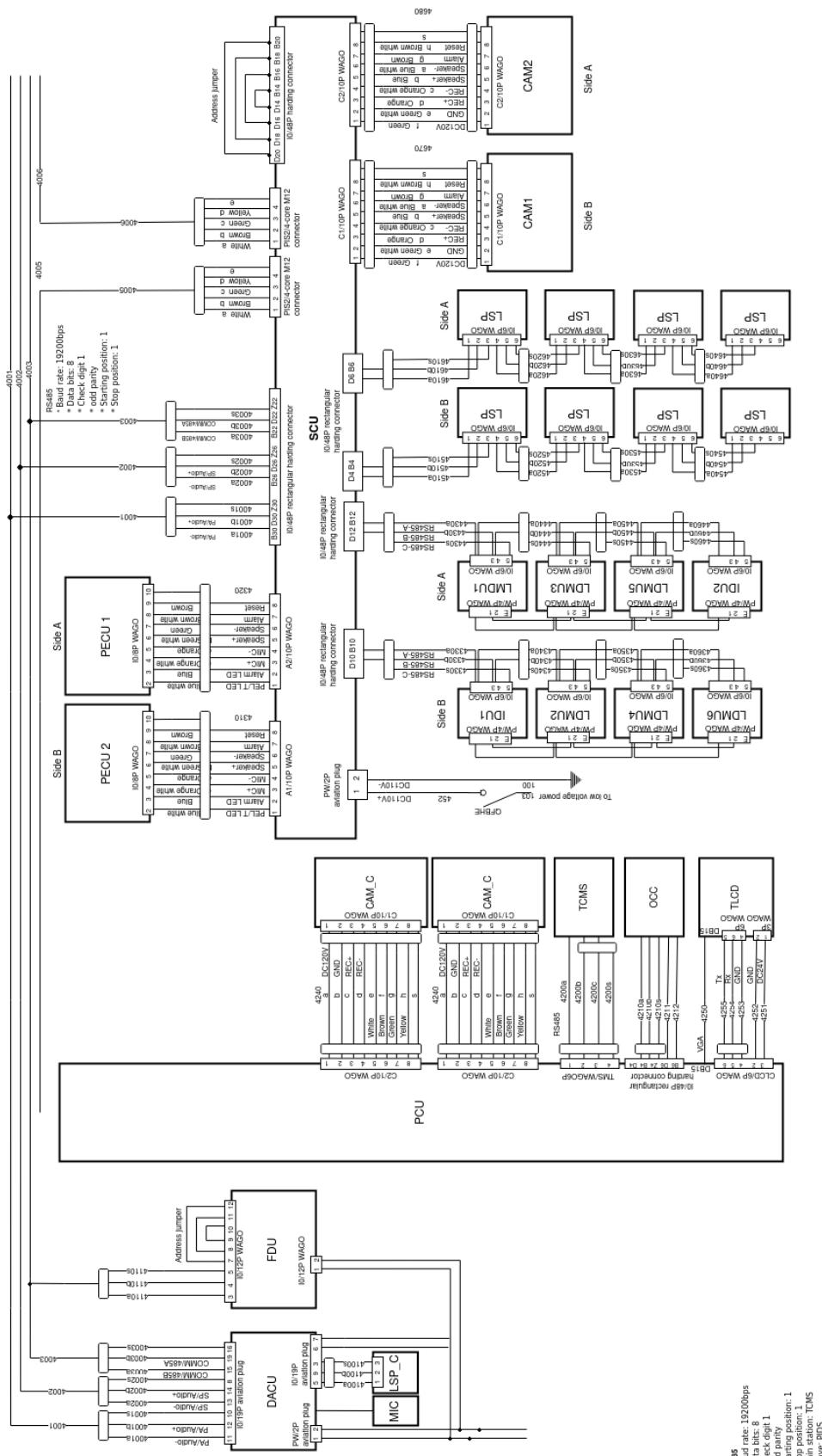


FIGURA 2.5. Diagrama de bloques del sistema PIDS, elaborado por el autor en base al plano de referencia de SOFSE.

RS485
 * Baud rate: 19200bps
 * Data bits: 8
 * Check digit: 1
 * Stop length: 1
 * Start position: 1
 * Main station: TMS
 * Slave: PIDS



FIGURA 2.6. Fotografía del detalle de cableado de la unidad de rack del PIDS que corresponde a los carteles LED de salón.



FIGURA 2.7. Fotografía de un cartel de salón inicializándose bajo una prueba de operación.

Capítulo 3

Diseño e implementación

En este capítulo se abordan cuestiones de diseño, se presentan los requerimientos del sistema y la solución propuesta. Se detalla la solución en términos de arquitectura, patrones de software, descripción de componentes e implementación. En el desarrollo se utiliza la plataforma de hardware EDU-CIAA[15], la API Firmware_v3[16], y freeRTOS[17] como sistema operativo de tiempo real.

3.1. Requerimientos

El objetivo principal de este trabajo es diseñar e implementar un sistema de información visual para pasajeros a bordo del tren. Está dirigido a:

1. Todos los miembros del grupo de trabajo GICSAFE y SOFSE que participan de proyectos orientados a cubrir necesidades tecnológicas del sistema ferroviario argentino.
2. Alumnos y personal académico con intenciones de participar en proyectos de desarrollo aplicados a la industria.
3. Desarrolladores de software y equipamiento para trenes.

A nivel general, los requerimientos del proyecto son los siguientes:

- El sistema debe leer datos de información al pasajero de la red interna de los trenes y presentarlos en un display led. El sistema no se encargará de presentar los mensajes en formato de audio.
- El sistema permitirá implementar las funciones de visualización del sistema de información al pasajero existente. La solución existente es un sistema propietario que integra también un sistema de audio, un CCTV usando cámaras de seguridad, entre otras funcionalidades.
- El sistema que se especifica busca desacoplar funciones del equipamiento propietario para permitir realizar tareas de mantenimiento. Como ejemplo, la reposición de carteles led que en la actualidad quedan fuera de servicio por fallas o pérdida del material original y que no pueden ser reparados.

Estos requerimientos generales se traducen en requerimientos específicos y se dividen en tres grupos que se detallan a continuación: requerimientos funcionales, de integración y de documentación.

Requerimientos funcionales

- El sistema debe controlar arreglos de matrices led de 8x8 (64 leds individuales).
- El sistema debe presentar en el display información dinámicamente.
- El sistema debe poder almacenar una cantidad de información para visualización.
- El sistema debe permitir elegir entre distintos mensajes de visualización.
- El sistema debe permitir cargar los mensajes a visualizar a través de una computadora.
- El sistema debe poder reaccionar a un mensaje que es enviado para visualizar.

Requerimientos de integración con la red TCN

- Las placas de control deben ser compatibles con el sistema PIDS existente.
- Las placas de control deben poder alimentarse con 110 VDC.
- El bus de datos de entrada debe ser una interfaz RS-485.
- El sistema debe interpretar las tramas del PIDS que corresponden a los módulos LDU.
- El sistema debe manejar tramas en ciclos típicos de 16-20 ms.

Requerimientos de documentación

- Se debe generar un documento de casos de prueba.
- Se debe generar una guía de usuario.
- Se debe generar una presentación del sistema.
- Se debe generar un informe final de proyecto.

La tabla 3.1 sintetiza los requerimientos y les asigna un código de referencia para la evaluación de su cumplimiento.

Por último se explicita que para el desarrollo del presente proyecto se asume que:

- No habrá dependencias directas con otros proyectos enmarcados en el mismo PDE[1].
- No habrá dificultad ni excesivas demoras en la compra de los componentes electrónicos o de software necesarios.
- Se contará con recursos y materiales necesarios para validar las pruebas realizadas. Trenes Argentinos dará acceso a una formación ferroviaria con red TCN para realizar pruebas de campo.
- El sistema de información al pasajero se va a instalar en el sistema PIDS existente de las formaciones ferroviarias en operación.
- El sistema de información al pasajero no se va a instalar en redes TCN de tiempo real basadas en *Ethernet* (ETB/ECN).

TABLA 3.1. Tabla de requerimientos funcionales, de integración y de documentación del proyecto.

Código	Descripción
PIDS-REQ-FN-01	Control de módulos de matriz led 8x8.
PIDS-REQ-FN-02	Control de paneles de 2x6 modulos.
PIDS-REQ-FN-03	Control de displays basados en arreglos de 3 paneles.
PIDS-REQ-FN-04	Visualización de mensajes en idioma castellano.
PIDS-REQ-FN-05	Visualización de mensajes dinámicos.
PIDS-REQ-FN-06	Almacenamiento de información de trayecto.
PIDS-REQ-FN-07	Selección de contenidos disponibles.
PIDS-REQ-FN-08	Upstream de mensajes desde una computadora personal.
PIDS-REQ-INT-01	Compatibilidad de sistema con sistema existente.
PIDS-REQ-INT-02	Compatibilidad eléctrica.
PIDS-REQ-INT-03	Compatibilidad de interfaces RS485.
PIDS-REQ-INT-04	Compatibilidad con tramas de datos del módulo LDU.
PIDS-REQ-INT-05	Procesamiento de tramas menor a 16 ms.
PIDS-REQ-DOC-01	Documentación de casos de prueba.
PIDS-REQ-DOC-02	Guía de usuario.
PIDS-REQ-DOC-03	Presentación del sistema.
PIDS-REQ-DOC-04	Informe final de proyecto.

3.2. Casos de uso

Los casos de uso planteados se presentan como respuesta a historias de usuario. Las historias de usuario propuestas en este trabajo son:

- Como usuario del tren quiero ver el nombre de la estación a la que estoy arribando.
- Como conductor del tren quiero elegir el destino y recorrido asociado que se visualizará en los coches.
- Como sistema vinculado quiero transmitir mensajes de asistencia, emergencia e información al pasajero.
- Como componente de sistema quiero recibir e interpretar tramas de la red de datos del sistema PIDS

Estas historias de usuario presentan cuatro tipos distintos de usuarios: pasajeros, conductores, sistemas de información al pasajero, y componentes internos del sistema. Con esta oferta de usuarios de sistema se busca definir funcionalidad y casos de uso. Los principales casos de uso del sistema se presentan en la tabla 3.2.

TABLA 3.2. Tabla de casos de uso.

Código	Descripción
PIDS-UC-01	Visualizar estación.
PIDS-UC-02	Elegir destino.
PIDS-UC-03	Información de asistencia.
PIDS-UC-04	Receptor de tramas.

En el caso de uso UC-1, se involucra al tren como sistema disparador cuando arriba a una estación, y se presenta información visual al pasajero usando los carteles led de salón. En UC-2 se resuelve una acción del conductor al presionar un botón, y también se presenta información visual al pasajero, en este caso las estaciones cabecera del recorrido que se visualizan en los carteles led de frente y contrafrente del tren. El tercer caso de uso, UC-3, presenta información de asistencia cargada previamente, que se dispara por acción de un temporizador mientras el tren está en circulación. Por último, el caso de uso UC-4 involucra al módulo SCU (ver figura 2.5 de la red PIDS) y a un sistema externo, como puede ser la computadora de un operador u otro componente de la red, para decodificar las tramas de datos recibidas desde el SCU.

3.3. Arquitectura

El sistema PIDS de Trenes Argentinos es parte de una solución integral de la red TCN del fabricante de los trenes, la empresa China State Railway Group Company, Ltd. En este capítulo se describen los aspectos más relevantes del sistema PIDS de esa arquitectura y su relación con los componentes del sistema propuesto en este trabajo. El sistema diseñado en este trabajo sigue una arquitectura orientada a eventos. Se desarrollaron e implementaron distintos patrones de software buscando satisfacer propiedades de modularidad, portabilidad y escalabilidad. Algunos de los objetos implementados se describen a nivel de detalle para resaltar criterios y decisiones de diseño relevantes. La relación entre la arquitectura existente del PIDS de trenes y la arquitectura del sistema embebido basado en la plataforma CIAA, busca por un lado satisfacer la compatibilidad eléctrica del hardware, y por otro lado las historias de usuario planteadas en los casos de uso, en la etapa de definición de requerimientos.

En las secciones siguientes se describen los componentes del sistema y sus interacciones.

3.3.1. Contexto

El sistema PIDS forma parte de una solución integral, la red de comunicaciones del tren o red TCN, brinda información a los pasajeros y puede ser operada por el conductor del tren o los operadores. Se representa a nivel sistema con en el diagrama de la figura 3.1.

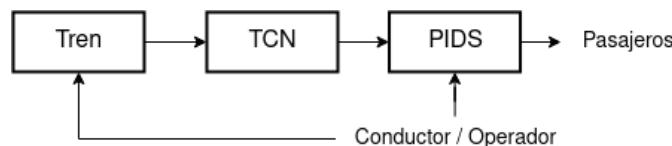


FIGURA 3.1. Diagrama del sistema Tren-TCN-PIDS.

La red TCN define una comunicación estándar usando dos buses jerárquicos llamados WTB (*Wire Train Bus*) y MVB (*Multifunction Vehicle Bus*). El sistema PIDS se interconecta al bus de datos MVB, como se indica en el diagrama de la figura

3.2.

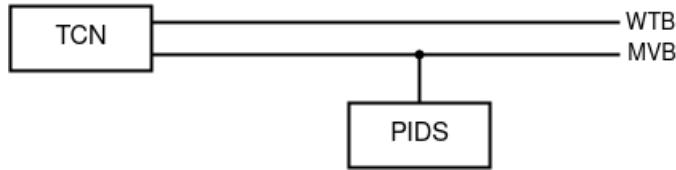


FIGURA 3.2. Diagrama de interconexión TCN-PIDS

El sistema PIDS tiene un bus de comunicación propio a través de una red RS485. Uno de los componentes de esta red es el módulo SCU (descrito en la topología del PIDS del capítulo 2), al cual se conectan distintos dispositivos como los display led, los mapas de recorrido led, las cámaras y parlantes, tal como se indica en la figura 3.3.

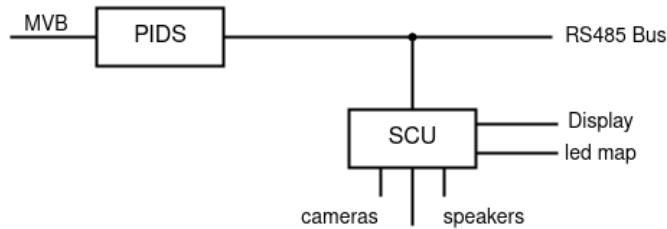


FIGURA 3.3. Diagrama del módulo SCU en la red PIDS.

Al módulo SCU se conectan las unidades IDU, que corresponden a los display led de salón. Cada unidad IDU contiene un controlador (*driver*) y el arreglo de módulos de matriz led que conforman el display. En la figura 3.4 se representan estos bloques funcionales.

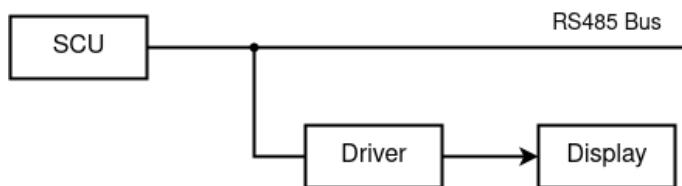


FIGURA 3.4. Diagrama de bloques del sistema SCU, placa de control y carteles led de salón.

El alcance del sistema desarrollado en este trabajo cubre la funcionalidad de este conjunto de bloques *Driver + Display*, que en la nomenclatura del sistema PIDS existente corresponde a los módulos IDU. Existen dos unidades de estos módulos por cada salón o coche. Muchas de las formaciones de SOFSE disponen de siete coches, por lo que se tiene un mínimo de catorce unidades IDU (dos por salón), más dos displays externos adicionales para el frente y contrafrente del tren que indican las estaciones cabecera del recorrido. Esto resulta en un total de al menos diecisésis unidades de control de carteles de matriz led por cada tren. Teniendo en cuenta las formaciones operativas de las líneas Mitre, Sarmiento y Roca del Área Metropolitana de Buenos Aires (AMBA), se puede estimar alrededor de treinta trenes operando en simultáneo, lo que resulta en aproximadamente 500 unidades

de carteles operando en vivo. El impacto que puede tener el aporte de este trabajo estará directamente relacionado con la operación de Trenes Argentinos, y puede contribuir a la extensión de la vida útil de los trenes.

3.3.2. Diseño

La propuesta de diseño busca cubrir las funcionalidades del bloque de control del display led. El display led es una unidad que se puede adquirir comercialmente. Sin embargo el driver para la red PIDS es una solución propietaria del fabricante y es la que se busca reemplazar con este desarrollo. En la figura 3.5 se presenta un diagrama de bloques del sistema de control propuesto. Este controlador usa comunicación serie a través de interfaces UART-RS485 y UART-USB. La UART (*Universal Asynchronous Receiver Transmitter*) es un periférico del microcontrolador de la plataforma CIAA. La alimentación de la CIAA difiere de aquella existente en la red RS485 de Trenes Argentinos, por lo que también es necesario un bloque de conversión de tensión DC-DC para garantizar compatibilidad eléctrica. La comunicación con el display se realiza a través de un adaptador eléctrico, que consiste básicamente en un puerto de entrada-salida.

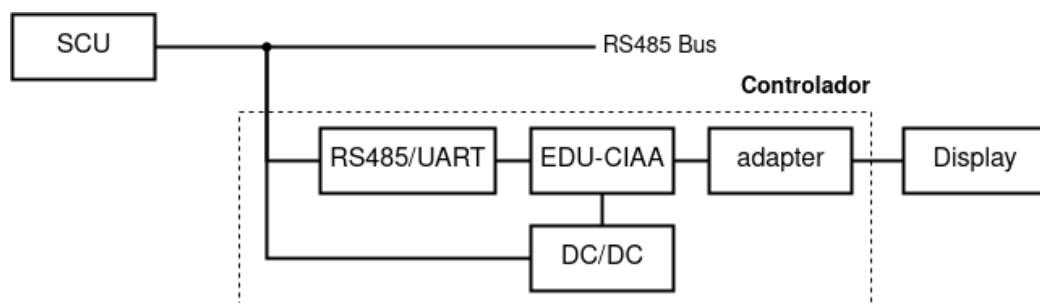


FIGURA 3.5. Diagrama de bloques del controlador propuesto.

A nivel lógico, el sistema que se propone consiste en cuatro objetos activos que interactúan entre sí, tal como se indica en la figura 3.6. El prefijo '`ao:`' señala que el bloque es un objeto activo. El objeto activo `UART` es el encargado de recibir e interpretar las tramas de datos que viajan por la red RS485 del bus de datos del SCU. El objeto activo `Button` es el control manual del operador para accionar el sistema. El objeto activo `PIDS` es una máquina de estados que representa el estado del tren, y contiene los mensajes y nombres de las estaciones. El objeto activo `DisplayLed` es el encargado de codificar los mensajes en el formato correspondiente a la matriz led del display.

Esta vista estructural del sistema plantea la interacción entre componentes e indica su dependencia funcional. El diseño modular permite hacer cambios en los componentes de forma independiente. Por ejemplo, se podrían reemplazar los mensajes al pasajero en el objeto `PIDS` sin afectar el resto del sistema, o si fuera necesario cambiar el hardware de control, se podría reemplazar la lógica de control del display led modificando únicamente el objeto `Displayled`.

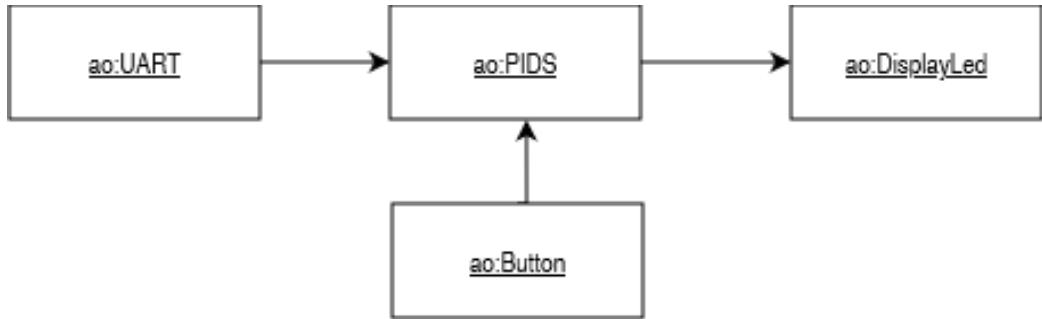


FIGURA 3.6. Vista estructural del sistema propuesto.

3.4. Implementación

En este trabajo se implementa un sistema embebido usando el lenguaje de programación C y el sistema operativo de tiempo real freeRTOS. La plataforma de hardware utilizada es la CIAA-EDU-NXP, que dispone de un microcontrolador LPC4337 de la compañía NXP ([18]), con una arquitectura de 32 bits ARM Cortex-M4/M0. El firmware se desarrolló utilizando la *SAPI* y el *firmwarev3* [16] como capa de abstracción de hardware. Esta API es una interfaz para usar las funciones de la biblioteca CMSIS del fabricante del microcontrolador. Esta API es parte fundamental del proyecto CIAA.

En el diseño del sistema se desarrollaron plantillas para implementar patrones de software como máquinas de estado y objetos activos. La documentación, pruebas unitarias, mantenimiento y escalabilidad son los aspectos de calidad de software que se buscó satisfacer desde el diseño, y que se vieron facilitados con el uso de las plantillas. Los lineamientos y fragmentos de código relevantes de estas plantillas son descriptos en la sección de patrones de software.

La implementación de los objetos activos está descripta en la sección de componentes de sistema. Los atributos principales de la solución, las vistas de interacción entre componentes son parte de la documentación presentada. La implementación del controlador del display led tiene una sección a nivel de detalle para mayor comprensión. El firmware busca ser portable a aquellas versiones de hardware de display de matriz led compatibles con el conjunto de chips 74HC245, 74HC595 y 74HC138, o sistemas digitales equivalentes.

3.4.1. Organización del código fuente

La organización del código fuente que conforma el sistema embebido de este trabajo se detalla en el siguiente árbol de archivos:

```

1
2
3 AppRTOS
4 |-- config.mk
5 |-- inc
6 |   |-- common.h
7 |   |-- displayled.h
8 |   |-- FreeRTOSConfig.h
9 |   |-- ISR_GPIO.h
  
```

```

10 |   |-- ISR_UART.h
11 |   |-- main.h
12 |   |-- modulePanelDisplay.h
13 |   |-- portmap.h
14 |   |-- statemachine_AB.h
15 |   |-- statemachine_button.h
16 |   |-- statemachine_displayled.h
17 |   |-- statemachine_PIDS.h
18 |   |-- statemachine_UART.h
19 |   '-- userTasks.h
20 |-- LICENSE.txt
21 '-- src
22   |-- displayled.c
23   |-- ISR_GPIO.c
24   |-- ISR_UART.c
25   |-- main.c
26   |-- statemachine_AB.c
27   |-- statemachine_button.c
28   |-- statemachine_displayled.c
29   |-- statemachine_PIDS.c
30   |-- statemachine_UART.c
31   '-- userTasks.c

```

Se pueden observar dos directorios principales: *inc* y *src*. En *inc* se incluyen los archivos de encabezados y en *src* los archivos de código ejecutable. Existe un archivo principal o *main* que es el que instancia las secuencias de inicialización, recursos y tareas del sistema operativo. Las tareas, que incluyen a los objetos activos y procesos del sistema, se implementan en los archivos *userTasks*. Luego, para cada implementación de objeto activo existe una máquina de estados asociada en un archivo de encabezados (.h), y un archivo ejecutable (.c) con el prefijo *statemachine*. Finalmente los archivos con el prefijo *ISR* corresponden a las rutinas de interrupción. Como cada máquina de estado utiliza recursos del sistema operativo, se ha creado un archivo *common.h* que declara aquellos recursos como colas, *handlers* y semáforos que se utilizan entre tareas para comunicación interprocesos.

Esta organización permite encapsulamiento y modularidad de los componentes del sistema. De esta manera se facilita el mantenimiento y, con el uso de las mismas reglas de diseño, se facilita la escalabilidad.

3.4.2. Uso de recursos en RTOS

En este desarrollo se utiliza freeRTOS, una versión en C de un *kernel* de sistema operativo de tiempo real. La implementación se basa en la inclusión de los archivos de cabecera *FreeRTOS.h* y *FreeRTOSConfig.h*. Esta implementación ha sido validada en arquitecturas ARM Cortex-M4, en particular en la plataforma EDU-CIAA.

En todos los casos que fue posible, se utilizaron semáforos, colas y *mutex* para organizar y proteger el uso compartido de recursos. El caso de uso típico de *mutex* es la interacción de distintas tareas con la misma interfaz UART-USB para imprimir mensajes por pantalla. En el código fuente de cada objeto implementado se utiliza la siguiente plantilla para proteger el recurso, evitando accesos múltiples

y posibilidad de *deadlock*:

```

1 if (pdTRUE == xSemaphoreTake( xMutexUART, portMAX_DELAY )) {
2     vPrintString("Task AB is running.\r\n");
3     xSemaphoreGive(xMutexUART);
4 }
```

Las colas, *Queue*, se utilizan principalmente para comunicar eventos entre objetos activos. En todos los casos se referian con *handlers* usando el prefijo *queueHandle* como nomenclatura. En el siguiente fragmento de código se muestra un ejemplo, exhibiendo los mecanismos de control de errores utilizados.

```

1 queueHandle_button = xQueueCreate(QUEUE_MAX_LENGTH, sizeof(
    eSystemEvent_button));
2 if (queueHandle_button == NULL){
3     perror("Error creating queue");
4     return 1;
5 }
```

Todas las tareas y recursos del sistema operativo se han protegido contra errores informando al usuario ante fallas en la instanciación, previas al inicio del *scheduler* u orquestador.

```

1 if( xTaskCreate( vTaskReadSerial , "Serial Comm reading task",
2 configMINIMAL_STACK_SIZE*4, NULL, tskIDLE_PRIORITY+2, &
3 xTaskReadSerialHandler)
4 == pdFAIL ) {
5     perror("Error creating task");
}
```

Se utilizan también punteros de tipo *xTaskHandle* para referenciar las tareas del sistema operativo. Este tipo de referencias son de especial utilidad a la hora de eliminar tareas de forma dinámica en tiempo de ejecución. También es posible usar estas referencias para comunicación entre tareas.

El uso de *timers* por software fue el preferido en aquellos casos que su uso facilita el mantenimiento, legibilidad y comprensión de la implementación. La plantilla desarrollada para la creación de *timers* se presenta en el siguiente fragmento de código.

```

1 void timerCallback_displayed(TimerHandle_t xTimerDisplayHandle) {
2
3     if (pdTRUE == xSemaphoreTake( xMutexUART, portMAX_DELAY )) {
4         printf("Timer display led is running.\r\n");
5         xSemaphoreGive(xMutexUART);
6     }
7
8     eSystemEvent_displayed    displayed_timer_event =
9         evDisplayed_timeout;
10
11    if(xQueueSend(queueHandle_displayed , &displayed_timer_event , 0U)!=
12        pdPASS){
```

```

11     }
12   }
13 }
```

perror("Error sending data to the queueHandle_displayed\r\n");

Se procuró tener especial cuidado de superposición o competición del reloj del sistema operativo entre tareas. Distintos componentes pueden tener requisitos temporales que compiten entre si por prioridad del orquestador. Se verá en detalle la implementación del display led como ejemplo que puede competir con el arriba de mensajes por la interfaz UART.

Las rutinas de interrupción por hardware, o *ISR*, se utilizan cuando se requiere respuesta inmediata. Por ejemplo, al accionar manualmente un interruptor o bien al recibir mensajes o señales eléctricas desde el bus de datos del tren. La implementación elegida para el uso de interrupciones se representa con en el diagrama de secuencia de la figura 3.7.

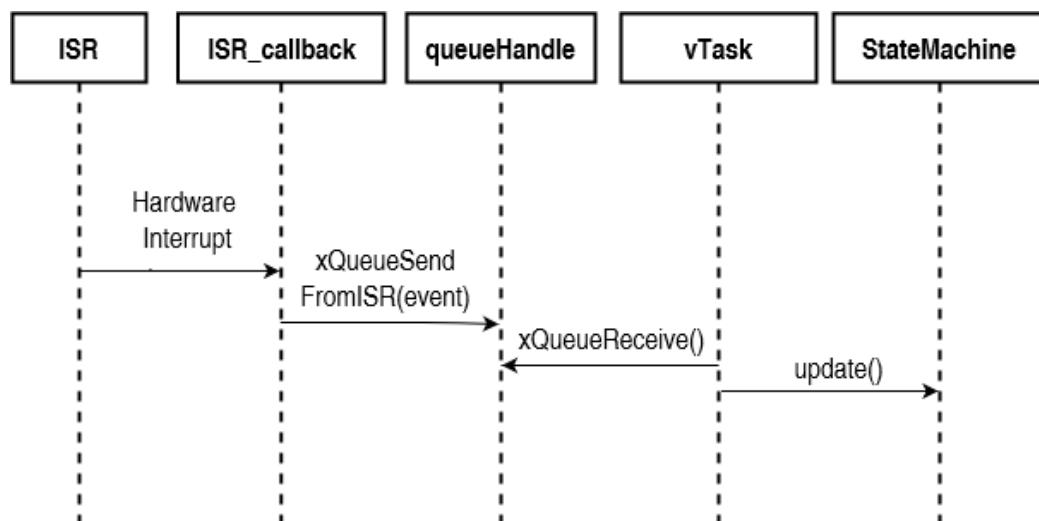


FIGURA 3.7. Diagrama de secuencia que representa la interacción entre componentes del sistema operativo cuando se dispara una interrupción o ISR.

El recurso de hardware que genera interrupciones (ISR) llama a una función *callback* (*ISR callback*), que sólo se encarga de entregar un mensaje de evento al sistema. Esta función no tiene lógica implementada, sólo avisa al sistema que hay una interrupción que atender. El evento se comunica a través de una cola, recurso elegido como interfaz de mensajes de los objetos activos. La cola es atendida por una tarea de sistema (*vTask*) que resuelve el código de ejecución. Esta tarea normalmente implementa un objeto activo que actualiza una máquina de estado.

Si bien puede llamar la atención la complejidad del mecanismo de atención de una interrupción, este patrón basado en objeto activo permite desacoplar la invocación y la ejecución de una máquina de estados. El objeto activo es una técnica de concurrencia muy utilizada por la ventaja de separar en hilos independientes (*threads*) la invocación de funciones (eventos) de las funciones de ejecución. Con

este mecanismo aparece la posibilidad de atender múltiples eventos en simultáneo, sin esperar a que se procese cada evento secuencialmente. Se logra paralelismo, atomicidad en la función que atiende la interrupción, muy baja latencia, y consistencia con en el uso de objetos activos bajo los lineamientos de la arquitectura orientada a eventos.

Por último se menciona el uso de memoria dinámica. El sistema tiene una interfaz de comunicación UART que a priori recibirá mensajes de largo variable y desconocido. Por lo tanto, el uso de *buffers* dinámicos para el almacenamiento y procesamiento de datos fue el mecanismo preferido. Para utilizar memoria dinámica se usan las funciones *memset()*, *pvPortMalloc()*, *memcpy()* y *vPortFree()*, a través de dos instancias de ejecución: una de recepción y almacenamiento de datos, y otra de ejecución y liberación de memoria. En el siguiente fragmento de código se explica este uso de funciones con dos tareas: *reader* y *writer*.

```

1 void vTaskReader(void *parameters) {
2     // Clear whole buffer
3     memset(buf, 0, buf_len);
4     // Loop forever
5     while (true) {
6         // Read from UART
7         uint8_t c_data=0;
8         if (uartReadByte( UART_USB, &c_data ) == true ){
9             // Store to buffer if not over buffer limit
10            if (idx < buf_len - 1){
11                buf[idx] = c_data;
12                idx++;
13            }
14            // Create a message ending string with null
15            if (c_data == '\n') {
16                buf[idx - 1] = '\0';
17                // Allocate memory and copy message.
18                if (msg_flag == 0) {
19                    msg_ptr = (char *)pvPortMalloc(idx * sizeof(char));
20                    if (msg_ptr==NULL){
21                        if (pdTRUE == xSemaphoreTake( xMutexUART,
22                            portMAX_DELAY)){
23                            printf("Buffer out of memory\r\n");
24                            xSemaphoreGive(xMutexUART);
25                        }
26                        // Copy message
27                        memcpy(msg_ptr, buf, idx);
28                        // Notify other task that message is ready
29                        msg_flag = 1;
30                    }
31                    // Reset buffer and index
32                    memset(buf, 0, buf_len);
33                    idx = 0;
34                }
35            }
36        }
37    }
38
39 void vTaskWriter(void *parameters) {
40     if (uart_msg_flag == true) {
41         // Print the message in the buffer
42         if (pdTRUE == xSemaphoreTake( xMutexUART, portMAX_DELAY)){
43             printf("%s\r\n", uart_msg_ptr);
44         }
45     }
46 }
```

```

44     xSemaphoreGive(xMutexUART);
45 }
46 // Free the memory block
47 vPortFree(uart_msg_ptr);
48 uart_msg_ptr = NULL;
49 }
50 }

```

Como se puede observar, cuando se recibe un carácter por la interfaz UART se copia el mismo en el *buffer*, siempre y cuando no esté lleno. Cuando el *buffer* se completa, se pide memoria y se copia el mensaje usando un puntero al bloque de memoria asignado. Luego, la tarea de ejecución lee el mensaje de ese puntero a memoria, o bien lo procesa para invocar otra función.

3.4.3. Patrones de software

En este desarrollo se ha hecho uso extensivo de los patrones máquina de estado, objeto activo, *pipeline*, observar y reaccionar, y superciclo. Se presenta en detalle el formato de plantillas desarrollado en C para freeRTOS para los patrones de máquinas de estado y de objeto activo.

Máquinas de estado

El desarrollo de la arquitectura orientada a eventos propuesta se basa en la interacción de máquinas de estado usando objetos activos. En este trabajo, las máquinas de estado se implementan sistemáticamente en C con un procedimiento paso a paso que se describe a continuación, usando como ejemplo una máquina de dos estados A y B que salta de un estado a otro por un evento temporal (*evTimeout*):

1. Representación de los estados y eventos con diagrama de estados.

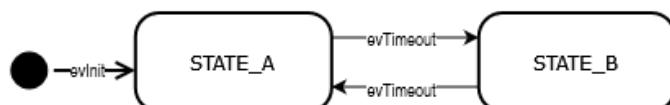


FIGURA 3.8. Diagrama de la máquina de estados AB ejemplo.

2. Definición de los estados, usando tipo enumerativo definido con el prefijo *eSystemState*:

```

1 typedef enum {
2
3     STATE_INIT,
4     STATE_A,
5     STATE_B
6
7 } eSystemState;

```

3. Definición de los eventos. Los eventos representan transiciones entre estados con un tipo enumerativo definido con el prefijo *eSystemEvent*:

```

1 typedef enum{
2   evInit ,
3   evTimeout
4
5 } eSystemEvent;
6 
```

4. Definición de un tipo puntero a función usando el prefijo `*pfEventHandler()` para designar los handlers específicos:

```

1 typedef eSystemState (*pfEventHandler)(void);
```

5. Definición de una estructura para la máquina de estados definiendo un tipo `sStateMachine`. Esta estructura debe incluir una variable estado (`eSystemState`), una variable evento (`eSystemEvent`) y un puntero a función (`pfEventHandler`). El puntero a función será una instancia de `handler` específico que maneje las transiciones entre estados.

```

1 typedef struct {
2
3   eSystemState    fsmState ;
4   eSystemEvent    fsmEvent ;
5   pfEventHandler  fsmHandler ;
6
7 } sStateMachine;
```

6. Definición de los `handlers` a implementar para el manejo de ejecución y transiciones entre estados.

```

1 eSystemState  InitHandler(void);
2 eSystemState  AHandler(void);
3 eSystemState  BHandler(void);
```

7. Instanciación de la máquina de estados como un arreglo de estructuras usando el prefijo `sStateMachine_`. Para el ejemplo de la máquina AB, la instancia de arreglo de estructuras sería la siguiente:

```

1 sStateMachine_AB fsmMachineAB [] =
2 {
3   {STATE_INIT_AB, evInit_AB, InitHandler_AB},
4   {STATE_A, evTimeout, AHandler},
5   {STATE_B, evTimeout, BHandler}
6 };
```

En esta definición habrá un `handler` en el momento de inicialización (`initHandler`) y un `handler` específico para cada estado.

8. Escribir el código ejecutable de los `handlers`. Una implementación a modo de ejemplo se presenta en el siguiente fragmento de código:

```

1 eSystemState InitHandler(void){
2     printf("State Machine Init...\n");
3     return STATE_A;
4 }
5
6 eSystemState AHandler(void){
7     printf("State Machine State A\n");
8     return STATE_B;
9 }
10
11 eSystemState BHandler(void){
12     printf("State Machine State B\n");
13     return STATE_A;
14 }
```

Se debe notar que para este ejemplo cada transición de estados se ejecutará por el evento *evTimeout*. La máquina inicia y se define en el estado STATE_A y luego alterna entre STATE_A y STATE_B por cada evento de timeout.

De esta manera, queda desacoplada la implementación de los *handlers* del resto de la estructura de la máquina de estados, logrando portabilidad, escala y modularidad. Los *handlers* serán funciones que se implementan con el sufijo *Handler()* y que por definición tienen un solo argumento de tipo *void*.

Con esta técnica de desacoplamiento, se implementan dos archivos por cada máquina de estado : uno de encabezados (*stateMachine.h*) con las definiciones, y otro con la implementación (*stateMachine.c*) de los handlers.

Objeto activo

Los objetos activos se implementan en esta aplicación como tareas de freeRTOS, usando dos superciclos anidados de tipo *while(true)*, tal como se muestra en el siguiente fragmento de código:

```

1 void vTaskAB(void *xTimerHandle)
2 {
3     (void)xTimerHandle;
4
5     // Task successful creation message
6     if (pdTRUE == xSemaphoreTake( xMutexUART, portMAX_DELAY )) {
7         printf("Task AB is running.\r\n");
8         xSemaphoreGive(xMutexUART);
9     }
10
11    while(true) {
12
13        // State machine init
14        eSystemEvent_AB newEvent = evInit_AB;
15        eSystemState_AB nextState = STATE_INIT_AB;
16        fsmMachineAB[nextState].fsmEvent = newEvent;
17        nextState = (*fsmMachineAB[nextState].fsmHandler)();
18
19        // Active object
20        while(true) {
21            if( pdPASS == xQueueReceive(queueHandle_AB, &newEvent,
22                portMAX_DELAY)) {
23                fsmMachineAB[nextState].fsmEvent = newEvent;
24                nextState = (*fsmMachineAB[nextState].fsmHandler)();
25            }
26        }
27    }
28}
```

```

25     }
26 }
27 }
```

Notar que el primer ciclo `while()` es el que corresponde al funcionamiento normal de una tarea o proceso de RTOS. En este caso, es el encargado de inicializar la máquina de estados asociada al objeto activo. El ciclo `while()` de la línea 20 se encarga de bloquear la tarea hasta que se reciba un evento por la interfaz (cola de eventos). La interfaz del objeto activo es una cola FIFO (*First In First Out*) con la función de sistema `xQueueReceive()` (línea 21). Si se recibe un evento, entonces se actualiza el estado de la máquina instanciando el handler que corresponda, como se observa en la línea 25. El diagrama de la figura 3.9 representa el objeto activo detallado.

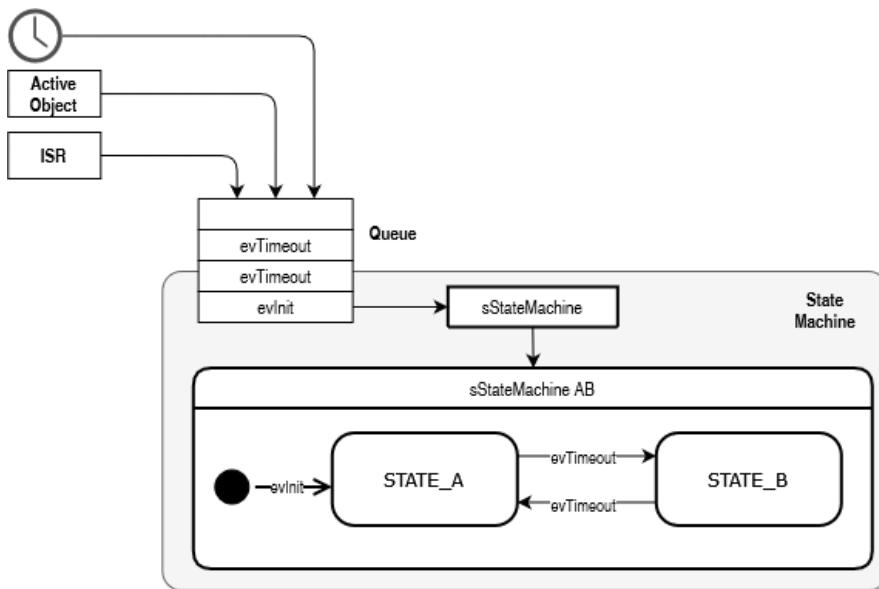


FIGURA 3.9. Diagrama del objeto activo de la máquina de estados AB ejemplo.

Para este ejemplo, la tarea de sistema recibe una referencia a un timer (`xTimerHandle`) que genera eventos de *timeout* y los encola en la interfaz del objeto activo (*Queue*). Con la misma interfaz, los eventos también podrían ser generados por otros objetos activos, o por rutinas de interrupción. Los mensajes en cola los recibe la tarea de sistema que actualiza la máquina de estados (`vTaskAB`).

3.4.4. Componentes del sistema

En esta sección se describen los componentes principales de la solución representada con el diagrama estructural de la figura 3.6. En la figura 3.10 se presentan las interacciones entre componentes en forma de secuencia para los distintos casos de uso. Los cuatro objetos activos utilizados son:

- **UART:** objeto activo que sirve de interfaz de comunicación con la red RS485 del sistema PIDS. Se encarga de recibir los mensajes de la red, identificarlos y enviarlos al objeto PIDS.

- **PIDS:** objeto activo que contiene la lógica de procesamiento de señales del tren, los mensajes que se visualizan en pantalla y los trayectos disponibles.
- **displayLED:** objeto activo responsable de codificar los mensajes que vienen del PIDS para ser visualizados en un display de matriz led.
- **Button:** objeto activo responsable de recibir accionamientos manuales del conductor del tren.

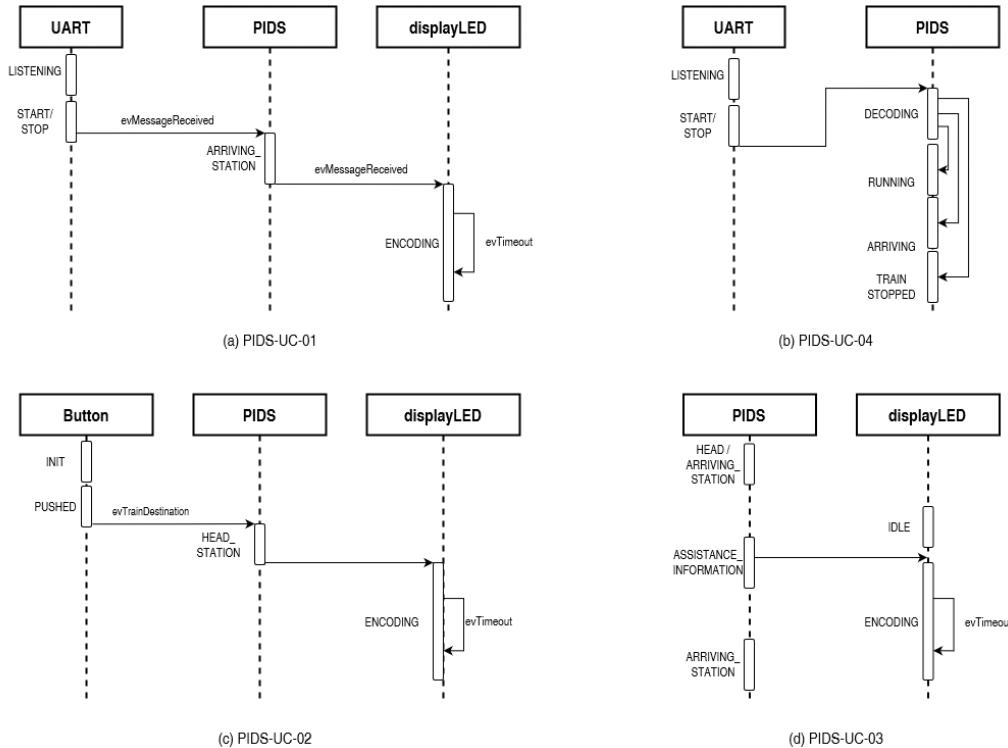


FIGURA 3.10. (a)caso de uso de visualización de estación; (b) caso de uso de receptor de tramas;(c) caso de uso de elección de destino por accionamiento del conductor;(d) caso de uso de visualización de información de asistencia.

UART

Este tipo de interfaz suele ser común en numerosas aplicaciones y abundan ejemplos utilizados para leer y escribir desde y hacia un periférico UART. El caso trivial es una aplicación éco: todo lo que se recibe por la UART se vuelve a escribir y enviar por la UART. Sin embargo, la aplicación específica determina la lógica de procesamiento de mensajes. Por ejemplo, si se recibe un carácter determinado, entonces se activa tal objeto; si se genera un evento en tal otro objeto, entonces se envía un mensaje de aviso.

En el sistema desarrollado, la UART es la interfaz de comunicación con el resto de la red PIDS. Esta debe procesar eventos que indican aceleración y desaceleración del tren. Los eventos se reciben codificados en tramas con formato específico, que a priori carecen de documentación. Como se verá en la sección de ensayos, se ha observado que las tramas normalmente tienen un encabezado (*header*), una carga de datos (*payload*), y un final de trama (*trailer*). Sin embargo, también se ha observado que los mensajes del tren pueden tener largo variable. El diseño de este

componente se representa con el diagrama de la figura 3.11.

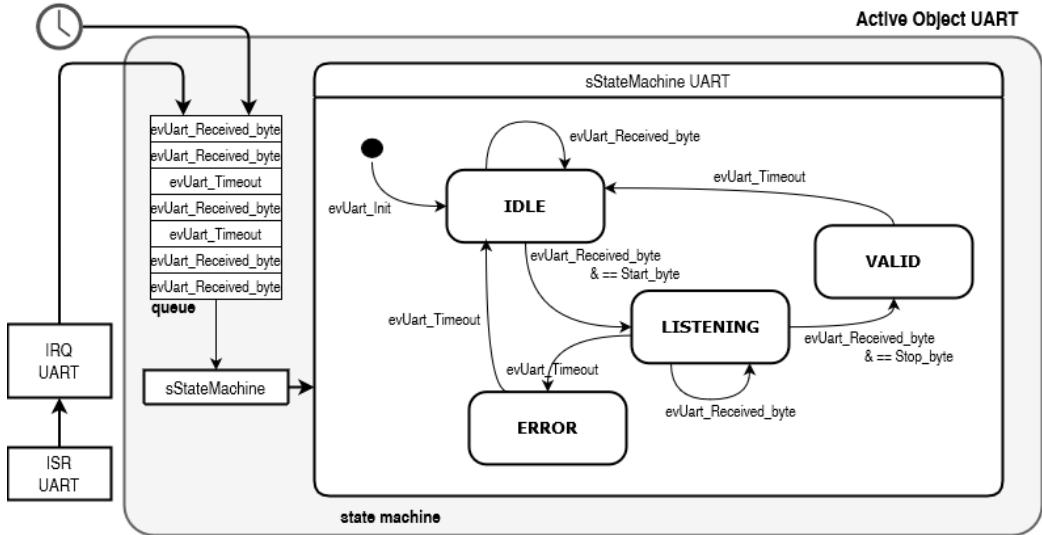


FIGURA 3.11. Diagrama del objeto activo UART implementado.

El periférico UART es un componente de bajo nivel del microcontrolador que admite el control por interrupciones (ISR). Cada byte entrante genera una interrupción y de inmediato el *handler* *IRQ_UART* envía un mensaje de actualización al objeto activo. Los eventos admitidos son:

- *evUart_Received_byte* para la recepción de un byte de datos;
- *evUart_Timeout*, generado por un timer específico de control.

La máquina de estados admite cuatro estados distintos:

- *IDLE*: estado inicial y de reposo.
- *LISTENING*: estado generado al recibir el Header o inicio de trama.
- *VALID*: estado generado al completar un mensaje con el Trailer o final de trama.
- *ERROR*: estado alcanzado una vez generado un timeout en el estado listening.

El estado *LISTENING* es el *core* de la máquina de estados. El *handler* de ejecución permite guardar en memoria dinámica el contenido de un mensaje de largo variable una vez que se recibe un byte de inicio de trama. El mensaje se completa cuando se recibe un byte de final de trama, que genera mediante *timeout* el evento de transición al estado *Valid*. La lógica que se plantea con este diseño es que, una vez validada una trama de datos, se genere un evento de actualización hacia el componente externo PIDS. De esta manera, se desarrolló un componente UART con flexibilidad, que permite cambiar los bytes de *header* y *trailer* y admite un *buffer* de largo variable usando memoria dinámica.

Display led

En esta sección se describe la solución implementada para controlar carteles de matriz led. La estructura de control, el conjunto de chips compatibles, el uso y posible reutilización en otros sistemas se presenta con diagramas y detalles de código fuente.

Los carteles led utilizados en este trabajo se basan en una tecnología de microcontrolador (MCU) y sistema digital. El MCU genera los mensajes a visualizar en formato de datos binarios, y el sistema digital recibe los datos binarios y señales de control para prender los leds del cartel de forma ordenada. Los carteles se componen de módulos matriciales de 8x8 leds, es decir, 8 filas de 8 leds por fila cada módulo. Los arreglos de módulos forman paneles, y los arreglos de paneles forman carteles. Los módulos pueden ser controlados a través de pulsos de tensión sincronizados entre filas y columnas. Sin embargo, si se quisiera controlar varios módulos de matriz led en simultáneo, la cantidad de señales a priori podría ser proporcional al número total de leds. El sistema digital expone una solución para que los paneles, como arreglos de módulos, puedan ser direccionados por un grupo de tres o cuatro señales de control usando técnicas de multiplexación y registros de desplazamiento, o *Shift Registers*, como se describe a continuación.

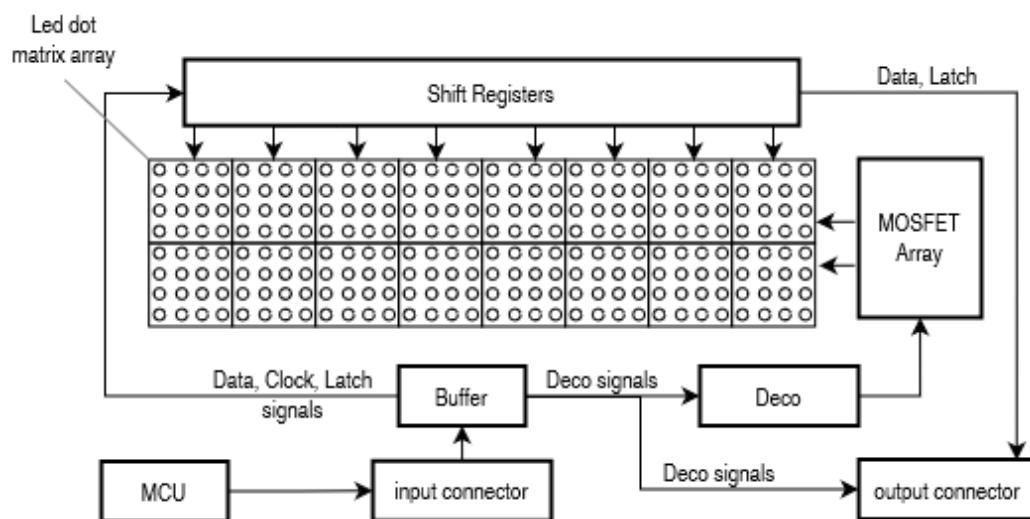


FIGURA 3.12. Diagrama de bloques del controlador de los carteles de matriz led utilizados en esta implementación.

En la figura 3.12 se presenta un diagrama de bloques del sistema digital de control para carteles de matriz led basados en el *chipset* 74HC245, 74HC595 y 74HC138. Se pueden distinguir los siguientes bloques:

- *MCU*: se encarga de generar y transmitir las señales del sistema a través del conector de entrada del cartel de matriz led.
- *input connector*: conector de pines en el cartel para señales de entrada (*Data, Clock, Latch*) provenientes del MCU.
- *output connector*: conector de pines en el cartel para señales de salida, para interconexión en serie con otro cartel.

- *Buffer*: adaptador de nivel de la señal de tensión y derivador a izquierda o derecha de la placa física.
- *Shift Registers*: registros de desplazamiento para enviar datos binarios a las columnas de los paneles.
- *Deco*: doble decodificador 3x8 para habilitar secuencialmente las filas de los paneles.
- *MOSFET Array*: circuito de corriente para energizar las filas de los paneles.
- *Led dot matrix array*: el cartel de matriz led propiamente dicho.

El funcionamiento del circuito es el siguiente. Los mensajes y señales de control (*data*, *clock*, *latch*, *deco*) se generan en el MCU y se transmiten al cartel a través de un conector de entrada (*input connector*). Las señales de control se direccionan a izquierda y derecha a través de *buffers* de la serie 74HC245D, que además entregan un nivel lógico de salida consistente, por ejemplo de 5 Volt. Las señales que van para el lado izquierdo contienen los datos binarios a visualizar en el cartel. A través de los registros de desplazamiento, los datos binarios (*data*) se cargan bit a bit sincronizados con cada ciclo de reloj (*clock*), hasta cargar los leds de una fila completa del cartel. Luego se envía un pulso de *latch* para descargar la fila y habilitar los registros de desplazamiento para los datos de la siguiente fila. Por el lado derecho del *buffer*, un par de decodificadores 74HC138 permiten encender hasta 16 salidas secuencialmente. En cada ejecución de la señal *deco* se energiza una fila usando transistores (*MOSFET Array*), que entregan la corriente necesaria para que todos los leds de cada fila puedan brillar con intensidad. La secuencia coordinada de enviar los datos a una fila, energizarla y luego hacer lo mismo con la fila siguiente, se repite hasta completar todas las filas del cartel en ciclos de 20 ms. De esta manera, se transmite un mensaje completo al cartel aproximadamente 50 veces por segundo, formando una imagen continua vista por el ojo humano.

Para armar carteles más grandes, se suele enchufar el conector de salida de un cartel al conector de entrada de otro igual, logrando conexiones en serie entre paneles. La lógica de codificación de mensajes es la misma, logrando que un sólo panel se conecte al MCU y el resto siga una conexión en cascada, usando el mismo grupo de señales generadas.

Los datos que se generan en el MCU para visualizar mensajes responden a un procesamiento ordenado de información. En la implementación desarrollada, se debe transformar el mensaje a visualizar como texto plano, y luego codificarlo en una matriz de unos y ceros que tenga las dimensiones del cartel de matriz led. A modo de ejemplo, si se codificara un carácter por módulo, se podría enviar mensajes de hasta 16 caracteres con un arreglo de 16 módulos 8x8. Cada carácter requiere de un mapa de bits que permita asociarlo con una matriz de datos binarios de 8x8. En la figura 3.13 se muestra un ejemplo con la letra 'H' codificada por la función f1, y se presenta esquemáticamente la implementación desarrollada. Se han diseñado dos funciones f1 y f2 usando el patrón *pipeline* (f1 y f2) para preprocesar mensajes de texto en datos binarios compatibles con el formato de carteles de matriz led. La función f1 se llama "*string_read_to_8x8_bytes_out()*", se encarga de recibir un mensaje en texto plano, mapearlo a un diccionario de caracteres y matrices, y entregar un arreglo de números que corresponden al mapa

binario de 8x8 de cada carácter. Así, por cada carácter que recibe f1 se generan 8 números binarios. Luego, la función f2 llamada `reshape_to_display()` recibe como argumentos las dimensiones del cartel y reordena el arreglo de números binarios acorde al formato del cartel. Esta última función básicamente hace la función de matriz transpuesta y considera tres casos posibles: que el mensaje a visualizar sea más corto que el cartel, que sea más largo, o que sea igual de largo. En los casos distintos, rellena con ceros la matriz transpuesta.

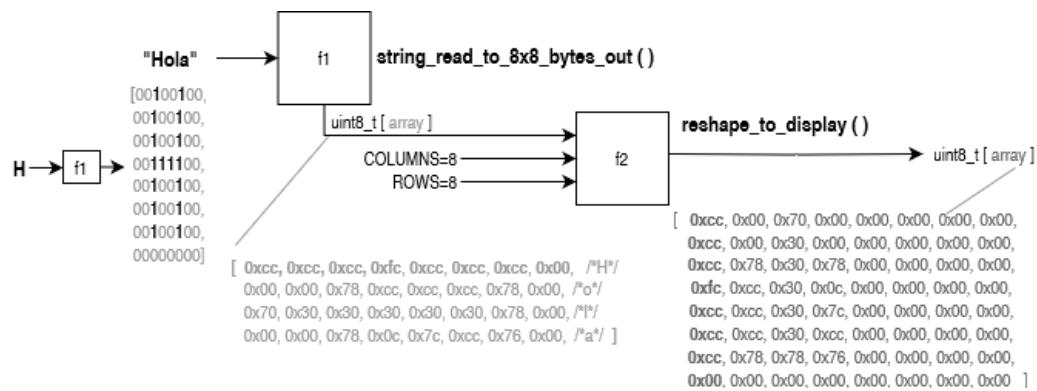


FIGURA 3.13. Procesamiento de datos para codificar mensajes.

La pieza de software desarrollada para el control del display led es un objeto activo consistente con el resto del sistema. El diagrama de la máquina de estados asociada se presenta en la figura 3.14. Se pueden distinguir tres estados, cada uno con un handler asociado. El estado *IDLE* es el estado inicial del sistema y es también el estado de reposo cuando no hay información para visualizar en el cartel. Ante un evento de mensaje recibido (`evDisplayLed_msg_received`), hay una transición al estado *PROCESSING* y se ejecuta el *pipeline* descrito para transformar texto plano en matrices de unos y ceros. Finalizado el procesamiento, se entrega al sistema digital para energizar el cartel y visualizar la información dentro del estado *ENCODING*.

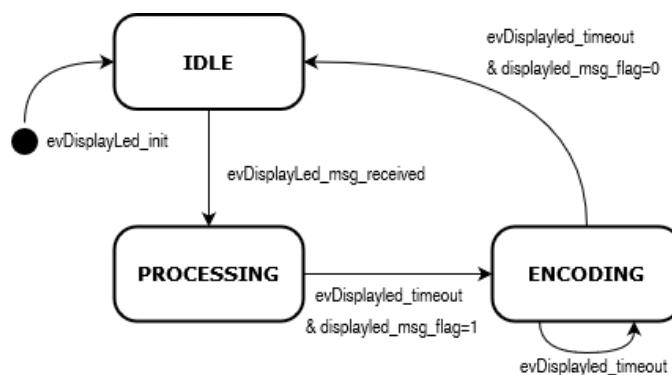


FIGURA 3.14. Diagrama de estados para la máquina de estados del display led.

En los siguientes fragmentos de código se detallan los *handlers* desarrollados para la máquina de estados presentada. El handler `displayed_procHandler()` implementa el pipeline de las funciones f1 y f2. Se puede observar que luego del mapeo de caracteres a arreglos de bytes en la línea 9, el reordenamiento de los bytes implica

pasar de una matriz de dimensiones $n \times m$, con ' n ' la cantidad de filas por carácter y ' m ' el largo del mensaje, a una matriz $p \times q$, con ' p ' la cantidad de filas del display led y ' q ' la cantidad de columnas. El ciclo de la línea 19 inicializa en cero la nueva matriz y en la línea 22 se completa el pipeline con la función `f2`.

```

1 eSystemState_displayed      displayed_procHandler(void) {
2
3     char *str1=messages[displayed_msg_idx];
4
5     uint8_t str1_len=strlen(str1);
6     uint8_t buffer_size=str1_len*CHAR_LENGTH;
7     uint8_t buffer[buffer_size];
8
9     string_read_to_8x8_bytes_out(str1,str1_len,buffer);
10
11    int n=CHAR_LENGTH;
12    int m=str1_len;
13    int p=DISPLAYLED_ROWS;
14    int q=DISPLAYLED_COLS;
15
16    displayed_size = p*q;
17
18    uint8_t B[displayed_size];
19    for (int i=0; i<displayed_size; i++)
20        B[i]=0;
21
22    reshape_to_display(buffer, displayed_buffer, buffer_size,
23                      displayed_size);
24
25    displayed_msg_idx++;
26    displayed_msg_idx%MESSAGES_TOTAL_NUMBER;
27    displayed_msg_flag=0;
28
29    return STATE_DISPLAYLED_ENCODING;
}

```

CÓDIGO 3.1. Código fuente del handler de procesamiento del display led.

El `handler displayLed_dataHandler()` implementa la lógica del sistema digital asociado al cartel, detallado al principio. Se pueden observar dos ciclos `for` anidados en las líneas 12 y 16, uno para recorrer carácter a carácter el mensaje, y otro para escanear bit a bit cada carácter. La línea 19 implementa una máscara de bit a bit para transmitir el dato al registro de desplazamiento, y las líneas 33-34 generan el pulso de *latch*. La secuencia codificada de las líneas 36 en adelante implementan la habilitación fila a fila usando decodificadores 3 a 8.

```

1 eSystemState_displayed      displayed_dataHandler(void) {
2
3     uint8_t data_8b;
4     bool_t value;
5
6     displayed_timer_cnt--;
7     if (!displayed_timer_cnt){
8         displayed_msg_flag=0;
9         return STATE_DISPLAYLED_IDLE;
10    };
11
12    for (int i=0; i<displayed_size; i++){

```

```

13     data_8b = displayed_buffer[i];
14
15     for (int j=0; j<8; j++){
16
17         // displayed_data
18         value = (((data_8b << j) & 0x80) == 0) ? 1 : 0;
19         printf("%d", value);
20         gpioWrite(displayed_panel_1, value);
21         gpioWrite(displayed_panel_2, value);
22
23         // displayed_clock
24         gpioWrite(displayed_clk, ON);
25         gpioWrite(displayed_clk, OFF);
26     }
27
28     if (i%DISPLAYLED_COLS==0){
29
30         // displayed_latch
31         printf("\r\n");
32         gpioWrite(displayed_latch, ON);
33         gpioWrite(displayed_latch, OFF);
34
35         // displayed_row_scanning
36         displayed_deco_cnt++;
37         displayed_deco_cnt%DISPLAYLED_ROWS;
38         if ((displayed_deco_cnt%1)==0){
39             gpioToggle(displayed_deco_A0); }
40         if ((displayed_deco_cnt%2)==0){
41             gpioToggle(displayed_deco_A1); }
42         if ((displayed_deco_cnt%4)==0){
43             gpioToggle(displayed_deco_A2); }
44         if ((displayed_deco_cnt%8)==0){
45             gpioToggle(displayed_deco_A3); }
46     }
47 }
48
49 return STATE_DISPLAYED_ENCODING;
50 }
51 }
```

CÓDIGO 3.2. Código fuente del handler de encoding del display led.

Con esta implementación de controlador de matriz led, ha sido posible realizar distintas pruebas de funcionamiento que se detallan en el capítulo siguiente.

Capítulo 4

Ensayos y resultados

En este capítulo se detallan los ensayos realizados en las formaciones ferroviarias y en los talleres de Trenes Argentinos. El orden cronológico de los ensayos es distinto al del desarrollo del firmware. En este documento se ha presentado previamente el diseño de la solución para facilitar la comprensión del trabajo realizado. El desarrollo de la solución fue posterior a una serie de mediciones realizadas en los talleres que permitieron identificar parámetros clave del sistema.

En las secciones que siguen se explican las mediciones realizadas en las visitas a los talleres de Victoria y Castelar de Trenes Argentinos Operaciones. Luego se presenta un análisis de datos de las tramas relevadas y también las pruebas de integración propuestas para validar el desarrollo.

4.1. Mediciones

4.2. Análisis de tramas

4.3. Pruebas en maqueta

En el circuito esquemático de la figura ?? se presenta el detalle de conexiones eléctricas entre bloques. Se puede observar que a la salida del conector de datos (CONN 2x8) hay dos buffers de la serie 74HC245D que direccionan las señales eléctricas a izquierda y derecha del arreglo de matrices led. A izquierda viajan las señales SER(data), SRCLK (Clock) y XXX (latch) al arreglo de Shift Registers de la serie 74HC595. Por la derecha se maneja la habilitación secuencial de las filas a través de un arreglo de decodificadores 3x8 de la serie 74HC138. Cada salida de los decodificadores se conecta a un driver de corriente en arreglo de transistores MOSFET FDS4953. Estos decodificadores cableados adecuadamente permiten manejar las 32 señales de un cartel de 4x8 módulos led.

Placa de control

4.4. Integración con red PIDS

4.5. Pruebas de campo

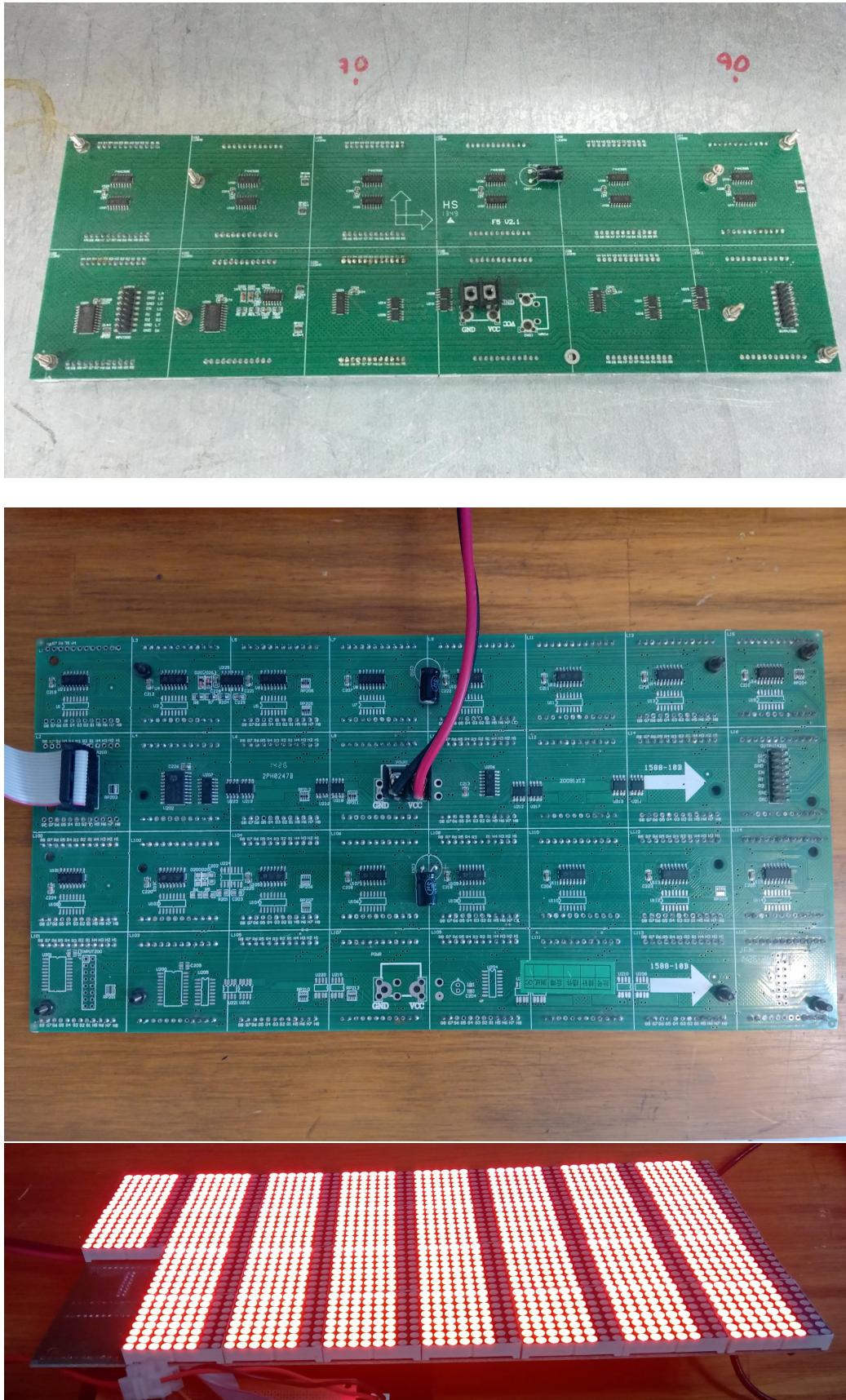


FIGURA 4.1. Fotografías de placas de control de los carteles de matriz led: (a) placa de 2x6 módulos; (b) placa de 4x8 módulos; (c) vista posterior de la placa de 4x8.

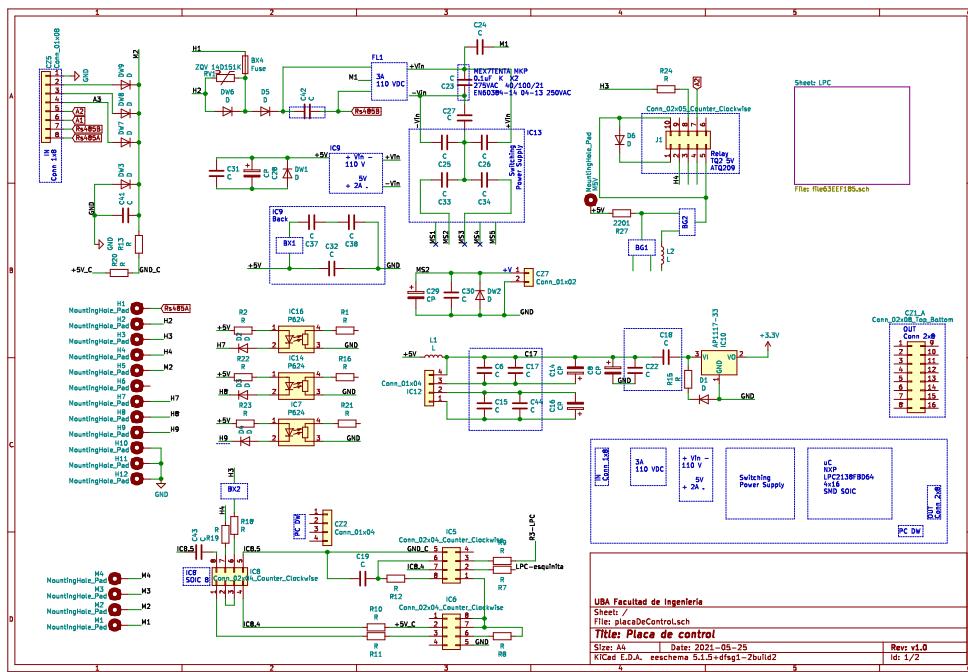


FIGURA 4.2. Circuito esquemático de la placa de control de los carteles LED de salón.

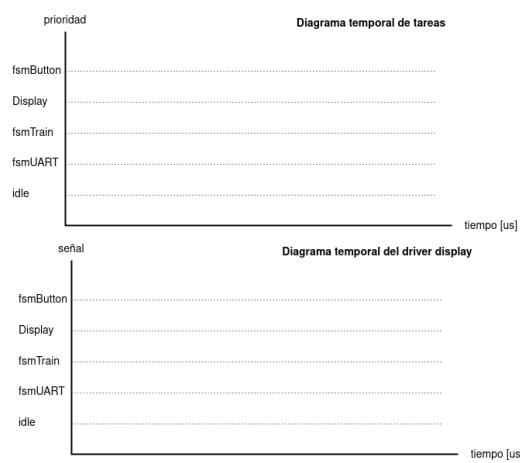


FIGURA 4.3



FIGURA 4.4. Fotografía del detalle de conexión de la placa de control de los carteles led de salón.

Capítulo 5

Conclusiones

En este capítulo se exponen los principales resultados obtenidos de este trabajo. Se presenta también su relación con el contexto y se plantea una serie de trabajos futuros.

5.1. Resultados obtenidos

5.2. Prospectivas

5.3. Bibliografía

Bibliografía

- [1] Pablo Gomez. *PDE 15 2020 SISTEMA DE MONITOREO Y GESTIÓN DE LA RED TCN EN FORMACIONES FERROVIARIAS*. Visitado el 2023-04-07. 2021.
- [2] IEC-61375-1999.
- [3] *Electronic railway equipment - Train communication network (TCN) - Part 2-1: Wire Train Bus (WTB)*. URL: <https://www.en-standard.eu/csn-en-61375-2-1-electronic-railway-equipment-train-communication-network-tcn-part-2-1-wire-train-bus-wtb/>.
- [4] *Electronic railway equipment - Train communication network (TCN) - Part 3-1 : Multifunction Vehicle Bus (MVB) - Matériel électronique ferroviaire*. URL: <https://www.en-standard.eu/iec-61375-3-1-2012-electronic-railway-equipment-train-communication-network-tcn-part-3-1-multifunction-vehicle-bus-mvb/>.
- [5] *Computadora Industrial Argentina Abierta*. URL: <http://www.proyecto-ciaa.com.ar/>.
- [6] *Digital Transformation of On-board Passenger Information Using Smart LCD Display Unit*. URL: <https://www.hitachi.com/>.
- [7] *Passenger Information Display*. URL: <https://www.global.toshiba/ww/products-solutions/railway/rolling-stock/passenger-information-display.html>.
- [8] *Passenger Information Displays – for Every Stage of the Journey*. URL: <https://www.trapezegroup.com.au/blog/passenger-information-displays-stop-app-website/>.
- [9] Yongxian Song et al. «Design of LED Display Control System Based on AT89C52 Single Chip Microcomputer». En: *J. Comput.* 6.4 (2011), págs. 718-724.
- [10] Shih-Min Liu, Ching-Feng Chen y Kuang-Chung Chou. «The design and implementation of a low-cost 360-degree color LED display system». En: *IEEE transactions on consumer electronics* 57.2 (2011), págs. 289-296.
- [11] W Kurdthongmee. *Design and Implementation of an FPGA-based Multiple-color LED Display*. 2004.
- [12] Yi-Zhen Lin et al. «Active-matrix micro-LED display driven by metal oxide TFTs using digital PWM method». En: *IEEE Transactions on Electron Devices* 68.11 (2021), págs. 5656-5661.
- [13] Alfonso Gago, José Fernández y Alfonso G Bohórquez. «Control architecture of a virtual matrix LED display without current drivers». En: *2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE. 2009, págs. 53-56.
- [14] Jianghua Feng et al. «Survey of development and application of train communication network». En: *Proceedings of the 2015 International Conference on Electrical and Information Technologies for Rail Transportation*. Springer. 2016, págs. 843-854.

- [15] Proyecto CIAA. *Computadora Industrial Abierta Argentina*. Visitado el 2016-06-25. 2014. URL:
<http://proyecto-ciaa.com.ar/devwiki/doku.php?id=start>.
- [16] Eric Pernia. *firmware v3*. Visitado el 2023-04-07. 2021. URL:
https://github.com/epernia/firmware_v3.
- [17] Richard Barry. *Free Real Time Operative System*. Visitado el 2023-04-07. 2016. URL: https://freertos.org/Documentation/RTOS_book.html.
- [18] NXP. *LPC4337*. Visitado el 2023-04-30. 2023. URL:
<https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc4300-arm-cortex-m4-m0/32-bit-arm-cortex-m4-m0-mcu-up-to-1-mb-flash-and-136-kb-sram-ethernet-two-high-speed-usb-lcd-emc:LPC4337FET256>.