

## 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 (PIDS) de 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 dentro del tren. Por esta red se reciben datos con información del estado del tren y se actualizan las pantallas.

En la red de Trenes Argentinos de la región AMBA operan más de 30 formaciones. Cada formación puede tener alrededor de 14 carteles dando un total de aproximadamente 420 carteles operativos del mismo modelo. Los carteles suelen presentar fallas y su reemplazo muchas veces se dificulta por importaciones. Los controladores de los carteles se basan en un conjunto de chips 74HC595, 74HC148, 74HC y la placa de comunicación es parte de un sistema propietario del fabricante de trenes. Con este trabajo se pretende hacer aportes para la sustitución de las placas de control y el mantenimiento de las unidades, y de esta manera extender la vida útil de los trenes. El sistema

desarrollado en este trabajo puede recibir tramas de datos de la red de comunicaciones del tren a través de una interfaz UART y controlar carteles de matriz led compatibles con los de las formaciones de Trenes Argentinos. El sistema sigue una arquitectura orientada a eventos. Se desarrollaron bloques funcionales usando técnicas de concurrencia y se usaron patrones de software tales como objeto activo y máquinas de estado. El sistema se ejecuta en un sistema operativo de tiempo real (freeRTOS) sobre una de las plataformas de hardware CIAA. Se han realizado ensayos en formaciones operativas.

Desarrollando piezas de hardware ad-hoc se han obtenido mediciones de tramas de datos de la red de comunicaciones y del módulo de control de las pantallas.

Se presenta un análisis de las tramas y el detalle de implementación para el control de los carteles de matriz led.



## *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. También agradecer a ellos dos y a todo el plantel de profesores 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.

Quiero 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. En especial también quiero agradecer a Bruno Pilato por su colaboración desde los talleres de Castelar para realizar pruebas en conjunto durante la pandemia.

Por último también agradezco a Magui, que me brindó soporte en todo momento; a mi madre y 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

<b>Resumen</b>	<b>I</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Introducción general . . . . .	3
1.2. Objetivos y alcance . . . . .	3
1.3. Estado del arte . . . . .	6
<b>2. Introducción específica</b>	<b>11</b>
2.1. Trenes: Red de comunicación TCN . . . . .	11
2.2. PIDS: Sistema de información visual para pasajeros de trenes . . . . .	12
2.3. Carteles y controladoras de matrices LED . . . . .	13
<b>3. Diseño e implementación</b>	<b>17</b>
3.1. Requerimientos . . . . .	17
3.2. Casos de Uso . . . . .	19
3.3. Arquitectura . . . . .	21
3.3.1. Contexto . . . . .	21
3.3.2. Diseño . . . . .	23
3.4. Implementación . . . . .	24
3.4.1. Organización del código fuente . . . . .	24
3.4.2. Uso de recursos en RTOS . . . . .	25
3.4.3. Patrones de software . . . . .	29
Máquinas de estado . . . . .	29
Objeto Activo . . . . .	31
3.4.4. Componentes del sistema . . . . .	32
UART . . . . .	33
Display LED . . . . .	35
<b>4. Ensayos y resultados</b>	<b>43</b>
4.1. Mediciones . . . . .	43
4.2. Análisis de tramas . . . . .	44
4.3. Pruebas en maqueta . . . . .	45
Placa de control . . . . .	45
4.4. Integración con red PIDS . . . . .	46
4.5. Pruebas de campo . . . . .	47
<b>5. Conclusiones</b>	<b>49</b>
5.1. Resultados obtenidos . . . . .	50
5.2. Prospectivas . . . . .	51
5.3. Bibliografía . . . . .	52
<b>Bibliografía</b>	<b>53</b>



# Capítulo 1

## Introducción

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. En una autopista se comunican accidentes u obras viales en ejecución usando carteles gigantes con información en tiempo real. Los pasajeros aéreos visualizan la información de arribo, estado o despegue de vuelos en un aeropuerto. 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 usan estos sistemas para conocer el destino o la próxima estación cuando están viajando. En algunos casos los carteles están a la intemperie y en otros dentro de un recinto, pero en general requieren estar sincronizados con los vehículos en movimiento.

Los sistemas de información visual para pasajeros (PIDS) tienen principalmente tres componentes: un sistema que genera datos, una red de transmisión y un sistema de pantallas. Dependiendo del dominio de aplicación, las especificaciones de cada sistema tienen distintos requerimientos. 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 importante la integridad, disponibilidad y confiabilidad de los datos. Pero también 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 de comunicaciones (TCN). Esta red TCN sigue un estándar y define tanto interfaces eléctricas como protocolos. A 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.

En el capítulo 1 se introduce al lector a la motivación original del trabajo realizado. Se explica el marco de investigación del que forma parte este proyecto y se presenta el estado del arte en controles de carteles led.

En el capítulo 2 se introduce vocabulario técnico específico. Se presenta una descripción del sistema con el foco en la red de comunicaciones TCN, el sistema PIDS, sus interacciones y componentes.

En el capítulo 3 se abordan cuestiones de diseño de sistema. Se especifican los requerimientos y casos de uso que se plantean en el espacio problema y también las consideraciones del espacio solución. Se detalla la solución en términos de arquitectura, patrones de software, descripción de componentes e implementación. Se incluye también los planos de los circuitos eléctricos del hardware existente que fueron relevados al realizar este trabajo.

En el capítulo 4 se abordan cuestiones relacionadas al entorno real del sistema: visitas técnicas, mediciones realizadas, hardware ad-hoc realizado para las mediciones y un breve análisis de las tramas de datos de la red PIDS existente.

En el capítulo 5 se tratan las conclusiones principales del desarrollo, su potencial fabricación en serie y los pasos a seguir para integrar al resto de ramales ferroviarios. En el apéndice de bibliografía se encontrarán las principales referencias técnicas, científicas e institucionales relevantes para este trabajo.

## 1.1. Introducción general

En este trabajo se desarrolla el sistema de control para carteles de matriz led del sistema de información visual para pasajeros 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 los carteles se interconectan a una red de comunicación del sistema PIDS por la que viajan distintos tipos de datos: por ejemplo datos de mapas led, mensajes de audio, información visual para los carteles, o bien video de cámaras de seguridad. En los buses de datos de la red TCN además se comunican datos de sensores de velocidad, de frenado, eventos que indican apertura o cierre de puertas, por citar algunos ejemplos.

Los carteles led del sistema PIDS presentan fallas a lo largo de su ciclo de vida. Esto 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, haciendo 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 porque además de desarrollar la industria local 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 información necesaria para construir y mantener los sistemas PIDS. Como consecuencia, este trabajo también tiene impacto directo en el pasajero, ya que contribuye a una mejora en la calidad del servicio.

## 1.2. Objetivos y alcance

El marco de este trabajo es 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". 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 (GICSAFE), 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 socioproductivo.

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 tiene una parte manual y una automática. Cuando el conductor del tren toma cabina para brindar servicio, 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. En estas marquesinas se presentan mensajes a los pasajeros como el nombre de la próxima estación, o la estación arribada (“próxima estación Belgrano”, “estás en estación Belgrano”, por ejemplo). Ésta información se presenta automáticamente en base a variables de sistema que monitorean el detenimiento del tren, su velocidad y la apertura o cierre de puertas. Esta y otra información de monitoreo y control viaja por una red de comunicación interna del tren que se denomina TCN (Train Communication Network) de acuerdo al estándar que la define [1]. Este estándar define para la red TCN dos buses jerárquicos donde se conectan los subsistemas electrónicos: el WTB (Wire Train Bus) y el MVB (Multi-Vehicle Bus) [CSN EN 61375-2-1][IEC 61375-3-1:2012]. El primero es el bus de mayor jerarquía que se conecta entre vagones y que se usa para monitorear cambios topográficos del tren. En el segundo se conectan los sensores y actuadores de cada coche como son los frenos, los controles de puertas, los monitores de velocidad, el sistema de información, etcétera. Los dos buses establecen el uso de interfaces eléctricas usando redes RS485.

El sistema propuesto en este trabajo pretende leer 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.

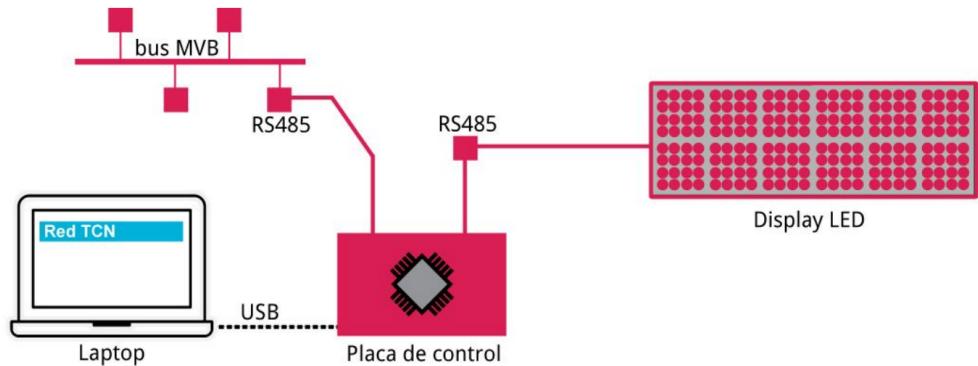


FIGURA 1.2. Diagrama de bloques del sistema embebido propuesto basado en la plataforma EDU-CIAA.

La placa de control está basada en la plataforma EDU-CIAA [2] 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 usando 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 cumplir con los estándares asociados a la red TCN;
- 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 existente es un equipamiento integrado y propietario, y este proyecto busca desacoplar algunas de sus funciones, las que corresponden a 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 proyecto 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 al pasajero todos los días.

### 1.3. 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. Lejos de ser un estudio sistemático, se pretende orientar al lector en las consideraciones que fueron tenidas en cuenta en este trabajo. Primero 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 interesantes y finalmente se presenta usando 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. Lo que se observa en general es que las empresas que proveen sistemas PIDS a las redes metropolitanas de transporte de las grandes ciudades lo hacen bajo formatos distintos. Algunas 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 edades avanzadas. 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.

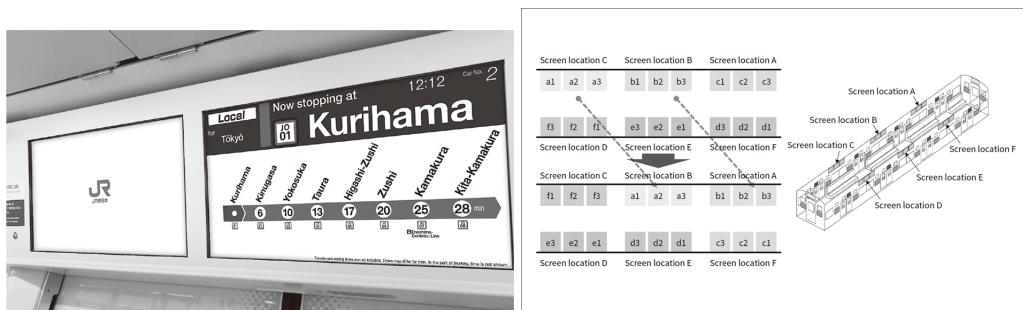


FIGURA 1.3. Solución de carteles para sistemas PIDS de Hitachi.  
Consultado en [3]

Hitachi ofrece una solución para publicidad de tres pantallas en array que se sincronizan para formar una sola y transmitir video con conectividad WiMAX. Cada uno de estos arreglos los 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. Con esto logran transmitir varios mensajes distintos en simultáneo a los pasajeros sin que tengan que moverse de su

asiento.

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" o 42", de 1920 x 540 píxeles, full color de hasta 16.7 millones de colores, con amplio ángulo de visión y de gran luminancia[4]. 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 [4]

El grupo austriaco Trapeze [5] 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. De los factores a tener en cuenta en la selección de carteles se distinguen los ángulos de visión, las condiciones del ambiente donde van instalados, por ejemplo si están a la intemperie o requieren 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 housing mecánico, el acceso a controles para personas con movilidad reducida, la alimentación eléctrica y la capacidad de realizar upgrades de sistema de forma remota.



FIGURA 1.5. Sistema PIDS del proveedor austriaco Trapeze. Consultado en [5]

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 interconexión con otros canales de información sobre todo en puntos nodales de transporte también es favorita. 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

<b>Conectividad</b>	RS485 Ethernet, Fibra Óptica WiFi, WiMax, GPS 2G / 3G / 4G / 5G
<b>Interconexión</b>	App del Tren Información multinodal Ómnibus Tablas de horarios programados Portales de noticias Publicidad Canal de información estatal
<b>Accesibilidad</b>	Información por audio Ángulos de visión de las pantallas Facilidades para personas en sillas de ruedas Facilidades para personas de edad avanzada Correcto y cuidado sistema de señalización
<b>Información</b>	Estimación de tiempos precisa Aviso de cortes en tiempo real Buen trackeo de vehículos Conexiones Mensajes de alerta o precauciones Números de emergencia
<b>Mantenibilidad</b>	Fácil instalación Costo de reposición Consumo eléctrico Upgrades

TABLA 1.1. Principales aspectos y servicios asociados que debe ofrecer un sistema PIDS. Elaboración del autor.

sistema centrados en el usuario se resumen en la tabla 1.1.

Según el punto de vista centrado en los carteles se suele tener en cuenta las dimensiones del cartel, la densidad de píxeles por unidad de área, la cantidad de colores o leds por píxel, los niveles de intensidad lumínica, el brillo y contraste, la potencia eléctrica como 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.

Todo cartel requiere de un controlador como interfaz que procesa 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 están en función de la cantidad de píxeles que

Display	LED matricial	LED RGB	TFT LCD	LCD RGB
<b>Colores</b>	monocromo bicolor tricolor multicolor (<10 colores)	desde 256 hasta 16,7M (típicamente)	hasta 16.7M	16.7M (típicamente)  1,000M
<b>Ángulo de visión</b>	110°	160°	120°-140°	178°
<b>Intensidad</b>	450 cd/m2	1500-2000 cd/m2	350 cd/m2	900 cd/m2
<b>Densidad de píxeles</b>	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/m2 (pixel size 179um x 192um)	4,26 M/m2 (pixel size 484 um x 484 um)  29 M/m2 (pixel size 179um x 192um)
<b>Potencia (consumo promedio)</b>	<10 W	15 W a 250W~316W	20 W	25W

TABLA 1.2. Principales características de displays para sistemas PIDS. Elaboración del autor.

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.

Unidad de procesamiento	MCU 8/16/32 bits	FPGA / ADIC / DSP	CPU / DSP
<b>Clock</b>	1-200 MHz	10-250 MHz	1-3 GHz
<b>Memory buffer</b>	1 KB	1-512 MB	1-10 GB
<b>Conectividad</b>	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
<b>Programación</b>	C / C++ / Assembly	VHDL, Verilog, XML	C, C++, Java, Python, XML

TABLA 1.3. Principales características de controladores de uso general para aplicaciones PIDS. Elaboración del autor.

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. Muchas veces se los necesita desinstalar, llevar a un centro técnico y actualizar 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 [6] se utiliza el chip AT89C52 para enviar caracteres chinos sobre matrices de 32 x 192 leds de un solo color; en [7] se implementa una pantalla led RGB de 320 x 240 píxeles que rota 360° permitiendo visualizar imágenes en color por persistencia de visión; en [8] se desarrollan algoritmos sobre FPGA usando búferes de datos para controlar una pantalla LED de 160 x 32 píxeles alcanzando 32,768 colores; en [9] 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 [10] 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 (8x64, 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 de la red de comunicaciones del tren y del sistema de información visual al pasajero. Se presenta una descripción de la arquitectura del sistema y de sus módulos principales. También se introduce el sistema de carteles led que presentan información al pasajeros, en particular en los coches de las formaciones ferroviarias de Trenes Argentinos.

### 2.1. Trenes: Red de comunicación TCN

La red de comunicaciones del tren (TCN) presenta una arquitectura de buses jerárquicos de dos niveles que se pueden identificar en la figura 1: el bus de datos WTB y el MVB[11]. El WTB se encarga de las comunicaciones entre coches a través de nodos con redundancia física, mientras que al bus MVB se conectan los dispositivos de cada coche. Algunos de estos dispositivos 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. El mapa de recorrido y los carteles LED en conjunto con otros dispositivos como los parlantes y las cámaras de video (CCTV) forman un sistema denominado Sistema de información al pasajero (PIDS).

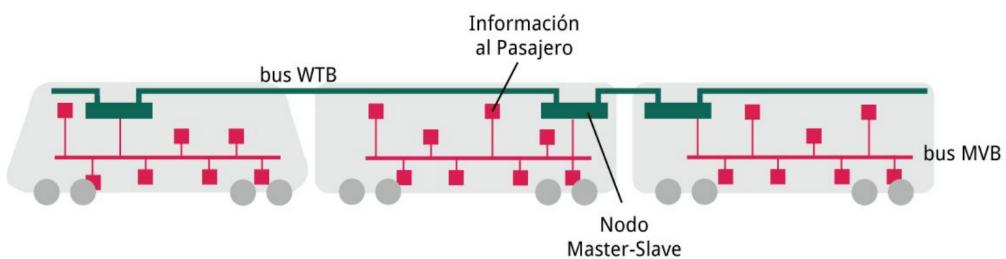


FIGURA 2.1. Diagrama del tren y de los buses WTB/MVB de la red TCN.

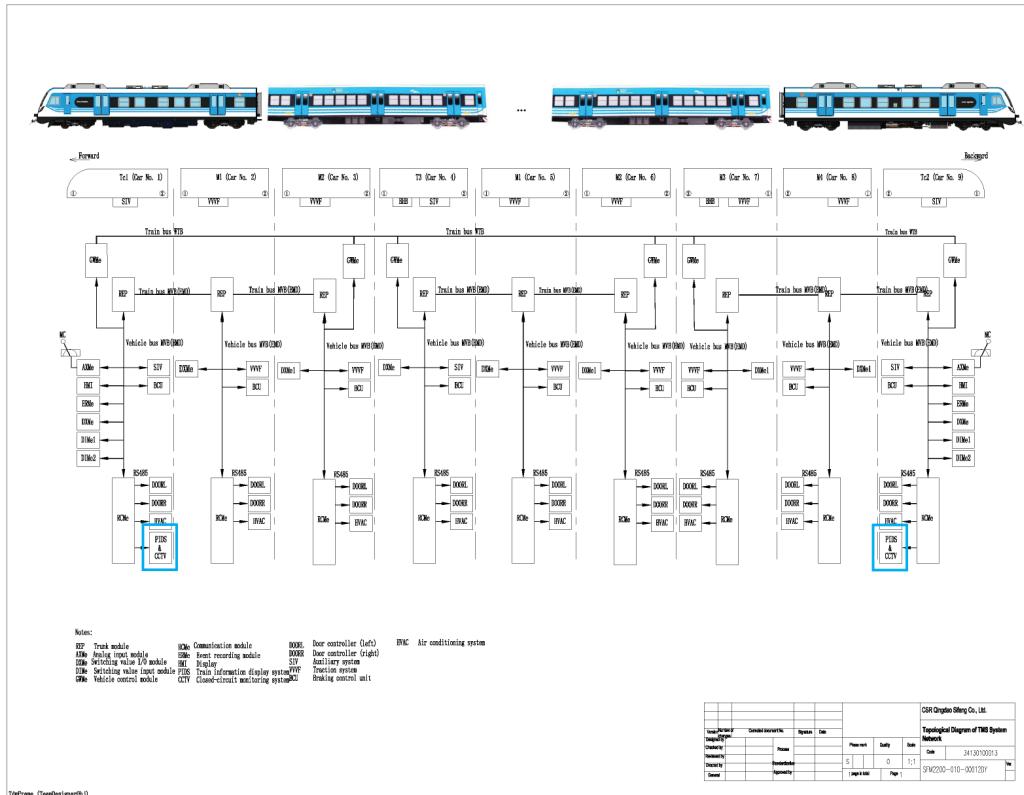


FIGURA 2.2. Diagrama de la red TCN de Trenes Argentinos. Cortesía SOFSE, plano modificado por parte del autor.

## 2.2. PIDS: Sistema de información visual para pasajeros de trenes



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

### 2.3. Carteles y controladoras de matrices LED

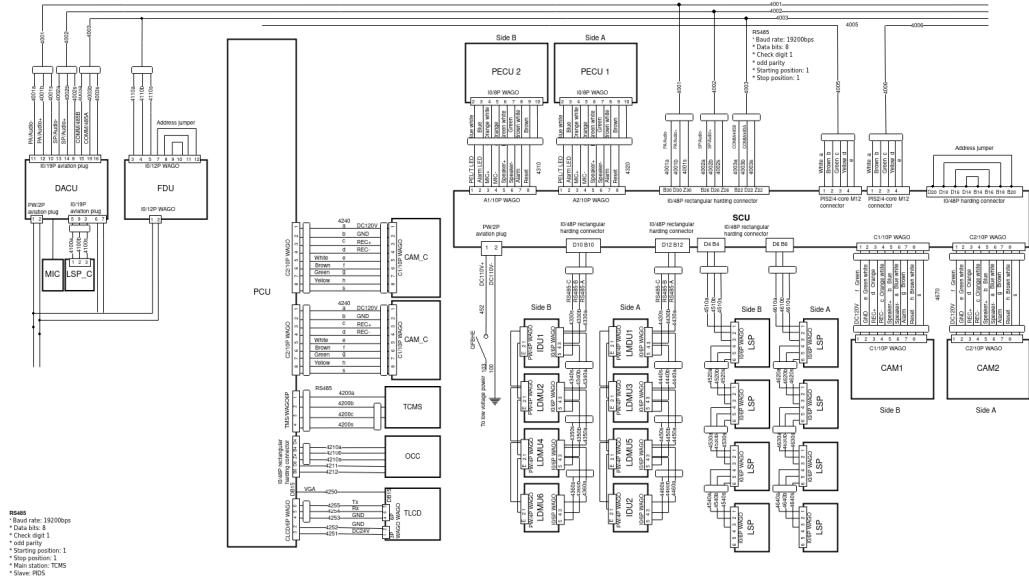


FIGURA 2.4. Diagrama de bloques del sistema PIDS, elaborado en base al plano de referencia de SOFSE.



FIGURA 2.5. Fotografía de las unidades de rack del sistema PIDS en el rack de salón.



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

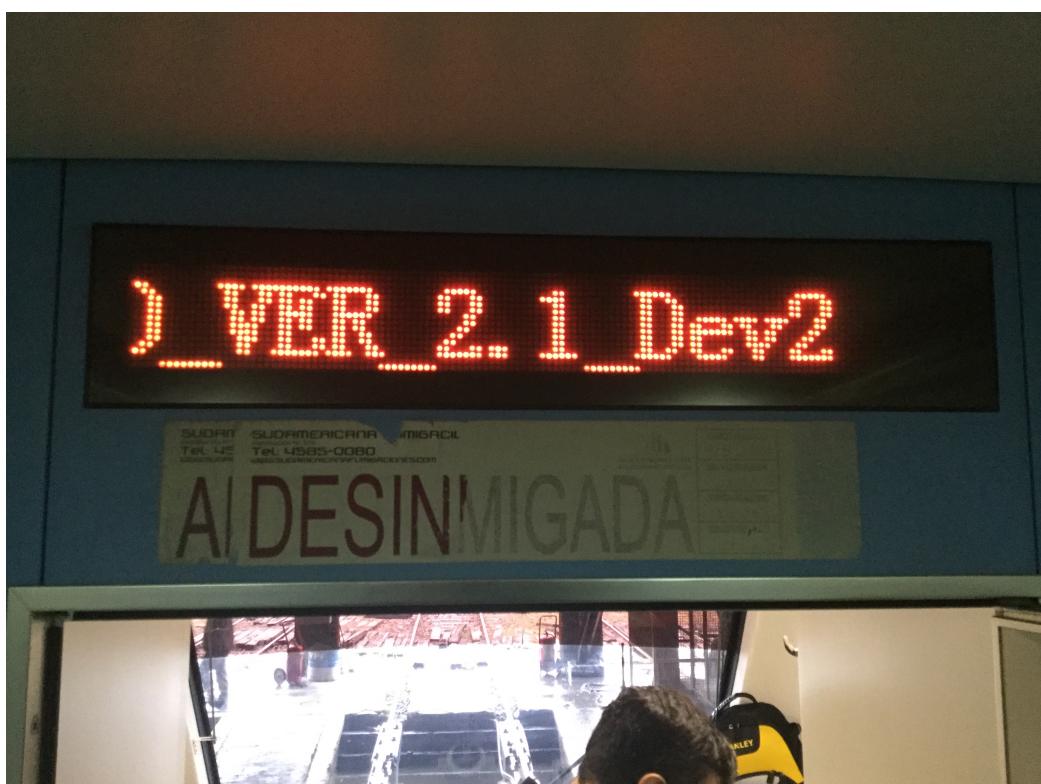


FIGURA 2.7. Fotografía de un cartel de salón inicializándose bajo una prueba de operación.



FIGURA 2.8. Fotografía del detalle de conexión de la placa de control de los carteles led de salón.

## Capítulo 3

# Diseño e implementación

En este capítulo se presentan los requerimientos del sistema y la solución propuesta. En el desarrollo se utiliza la plataforma de hardware EDU-CIAA[12], la API Firmware\_v3[13], y freeRTOS[14] como sistema operativo de tiempo real.

Siguiendo lineamientos de ingeniería de software, se presenta la solución en el siguiente orden: requerimientos, funcionalidades, casos de uso principales como ejes del espacio problema; arquitectura, patrones de software utilizados, técnicas de concurrencia, diagramas de secuencia, diagramas de interacción entre componentes, organización del firmware y planos de hardware para describir el espacio solución.

### 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 LED 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[15].
- 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.

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

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

- El sistema de información al pasajero no se va a instalar en redes TCN de tiempo real basadas en Ethernet (ETB/ECN).

## 3.2. Casos de Uso

Los casos de uso planteados se presentan como respuesta a historias de usuario. Las historias de usuario principales 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.

El caso de uso UC-1 involucra al tren como sistema disparador cuando arriba a una estación y presenta información visual al pasajero usando los carteles LED

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

TABLA 3.2. Tabla de casos de uso.

<b>UC-1: Visualización del nombre de la estación arribada</b>		
1	Nombre	Visualizar el nombre de la estación arribada.
1.1	Descripción	El sistema genera un mensaje que contiene información para el pasajero.
1.2	Actor principal	Pasajeros
	Disparadores	El evento se inicia cuando el tren arriba a una estación.
2	Flujo de eventos	
2.1	Flujo básico	<p>El tren comienza a frenar hasta llegar a velocidad 0 km/h.</p> <p>El subsistema PIDS recibe del bus MVB las tramas con la variable de control de velocidad.</p> <p>El subsistema PIDS busca el nombre de la estación actual, busca el nombre en la memoria.</p> <p>El subsistema PIDS envía la trama a la Red TCN con destino al subsistema HMI.</p> <p>El subsistema HMI recibe la trama generada por el PIDS y presenta el nombre de la estación en la pantalla.</p>
2.2	Flujo alternativo	<p>El tren se queda detenido en la estación.</p> <p>El subsistema PIDS recibe del bus MVB las tramas con la variable de control de velocidad.</p> <p>El subsistema PIDS compara el estado actual y no detecta cambios.</p> <p>El subsistema PIDS envía una trama a la red TCN interrogando al subsistema HMI.</p> <p>El subsistema HMI recibe el requerimiento y entrega el mensaje que indica que no se ha detectado cambios.</p> <p>El subsistema PIDS recibe el mensaje y no detecta cambios, no envía tramas.</p>
3	Requerimientos especiales	
4	Pre-condiciones	El sistema debe estar en modo ONLINE.
5	Post-condiciones	El sistema pasa al estado detenido.

TABLA 3.3

de salón. El UC-2 resuelve una acción del conductor al presionar un botón, y presenta también 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 previamente cargada que se dispara por acción de un timer mientras el tren está en circulación. Por último, el caso de uso UC-4 involucra al módulo SCU (ver 2.4 de la red PIDS) y a un sistema externo, como puede ser una 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 satisfacer por un lado la compatibilidad eléctrica de hardware, y por otro 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.

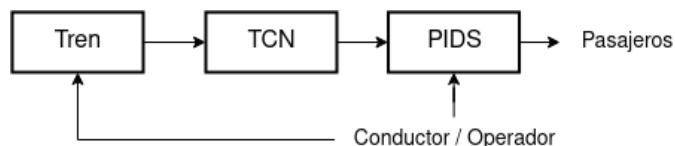


FIGURA 3.1. Diagrama del sistema Tren-TCN-PIDS.

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.

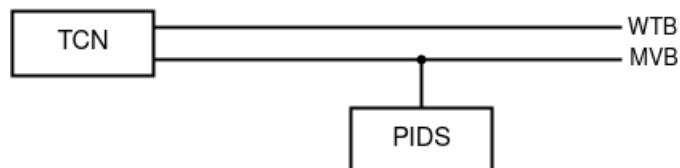


FIGURA 3.2. Diagrama de interconexión TCN-PIDS

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, 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.

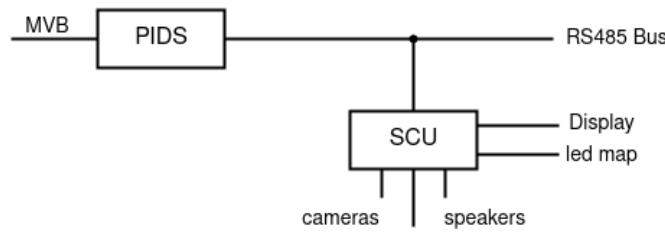


FIGURA 3.3. Diagrama del módulo SCU en la red PIDS.

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

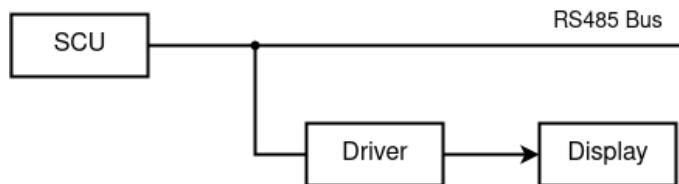


FIGURA 3.4. Diagrama de bloques del sistema SCU, placa de control y carteles LED de salón.

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 dieciséis unidades de control de display 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 displays operando en vivo. El impacto que puede tener el aporte de este trabajo estará directamente relacionado con la operación de Trenes Argentinos y definitivamente 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 es un periférico del microcontrolador de la plataforma CIAA. La alimentación de la CIAA difiere de aquella existente en la red RS485, por lo que también es necesario un bloque de conversión DC-DC para garantizar compatibilidad eléctrica. La comunicación con el display se realiza a través de un adaptador, que consiste básicamente en un puerto de entrada-salida.

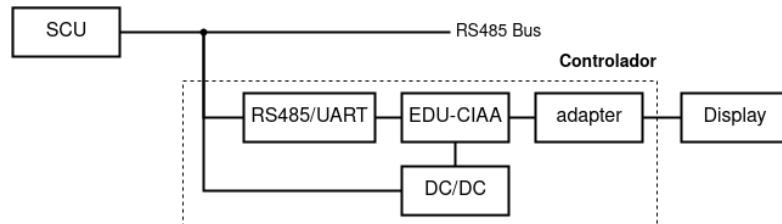


FIGURA 3.5. Diagrama de bloques del controlador propuesto.

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.

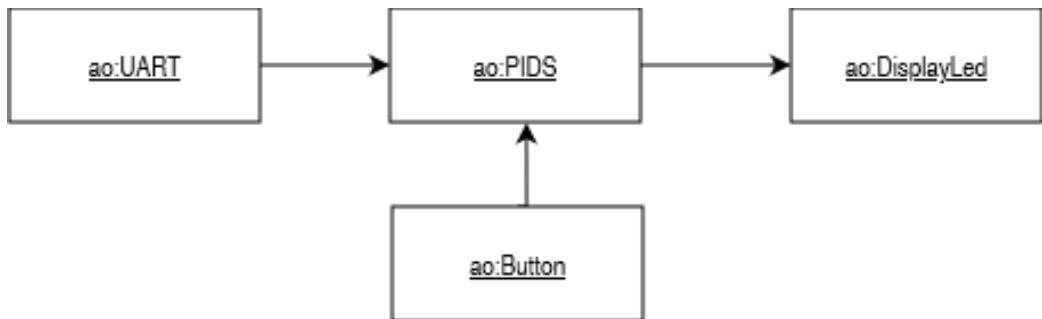


FIGURA 3.6. Vista estructural del sistema propuesto.

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 es el que 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. El prefijo .<sup>a</sup>o:<sup>z</sup> el subrayado denotan objeto activo.

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 bien reemplazar la lógica de control del display led si fuera necesario cambiar el hardware de control modificando únicamente el objeto DisplayLed.

### 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 ([16]), con una arquitectura de 32 bits ARM Cortex-M4/M0. El firmware se desarrolló utilizando también la *SAPI* y el *firmwareV3* [*firmwarev3*] 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. Con las plantillas se facilita la documentación, testing, mantenimiento y escalabilidad. Estos fueron aspectos de calidad de software que se buscó satisfacer desde el diseño. Se describen los lineamientos y fragmentos de código principales de estas plantillas en la sección de patrones de software.

La implementación de los objetos activos del sistema se describe en la sección de componentes de sistema. Se mencionan los atributos principales de la solución y se documentan las vistas de interacción entre componentes. 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 describe en el siguiente árbol de archivos:

```

1 AppRTOS
2 |--- config.mk
3 |--- inc
4 |   |--- common.h
5 |   |--- displayLed.h
6 |   |--- FreeRTOSConfig.h
7 |   |--- ISR_GPIO.h
8 |   |--- ISR_UART.h
9 |   |--- main.h
10 |   |--- modulePanelDisplay.h
11 |   |--- portmap.h
12 |   |--- statemachine_AB.h
13 |   |--- statemachine_button.h
14 |   |--- statemachine_displayLed.h
15 |   |--- statemachine_PIDS.h
16 |   |--- statemachine_UART.h

```

```

17 |   '-- userTasks.h
18 |-- LICENSE.txt
19 '-- src
20   |-- displayLed.c
21   |-- ISR_GPIO.c
22   |-- ISR_UART.c
23   |-- main.c
24   |-- statemachine_AB.c
25   |-- statemachine_button.c
26   |-- statemachine_displayLed.c
27   |-- statemachine_PIDS.c
28   |-- statemachine_UART.c
29   '-- 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 y un archivo ejecutable 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 referencian con handlers usando la nomenclatura con prefijo *queueHandle*. En el siguiente fragmento de código se muestra un ejemplo, exhibiendo los mecanismos de control de errores usados.

```

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

```

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 es 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_displayLed(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_displayLed displayLed_timer_event =
9         evDisplayed_led_timeout;
10
11    if (xQueueSend(queueHandle_displayLed, &displayLed_timer_event, 0U)!=
12        pdPASS){
13        perror("Error sending data to the queueHandle_displayLed\r\n");
14    }
15 }
```

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 scheduler. Se verá en detalle

la implementación del display Led como ejemplo que puede competir con el arribo de mensajes por la interfaz UART.

Las rutinas de interrupción por hardware o *ISR* se utilizan cuando se requiere respuesta inmediata, como 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 en el diagrama de secuencia de la figura 3.7.

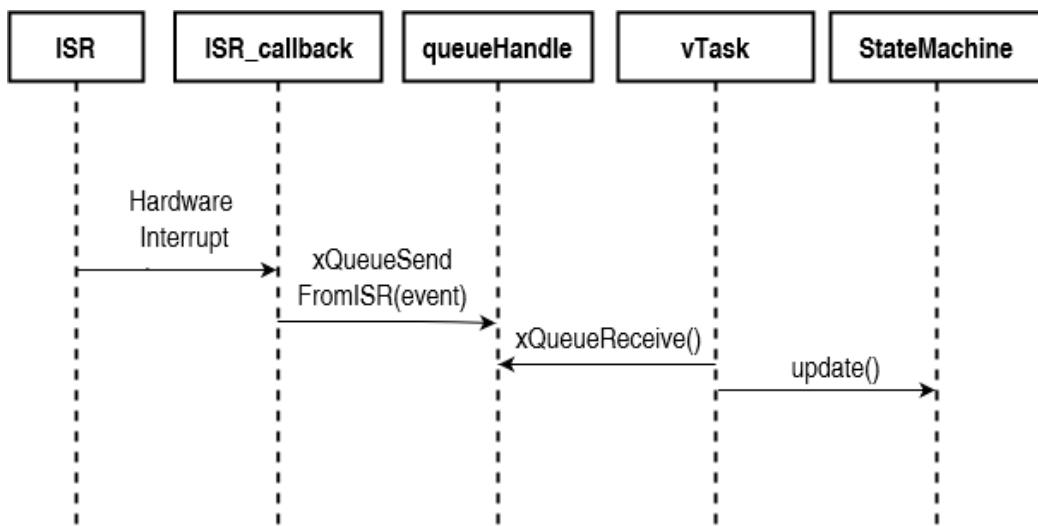


FIGURA 3.7. Diagrama de secuencia que representa la interacción entre componentes del sistema operativo cuando se dispara una interrupción o ISR.

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, 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 estados.

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 en una máquina de estados. El objeto activo es una técnica de concurrencia muy utilizada por la ventaja de separar en hilos independientes la invocación de funciones (eventos) de las funciones de ejecución. Con esto, la posibilidad de atender múltiples eventos en simultáneo, sin esperar a que se procese cada evento secuencialmente. Se logra paralelismo, atomicidad en el callback de la interrupción, muy baja latencia, y se mantiene consistencia en el uso de objetos activos siguiendo 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 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             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, observ and react, 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 el siguiente procedimiento paso a paso:

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
3     evInit ,
4     evTimeout
5
6 } eSystemEvent ;
  
```

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), un handler específico para cada estado.

8. Escribir el código ejecutable de los handlers. Una implementación 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 con dos superciclos anidados de tipo *while(true)*, 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    }

```

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 superciclo 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 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.

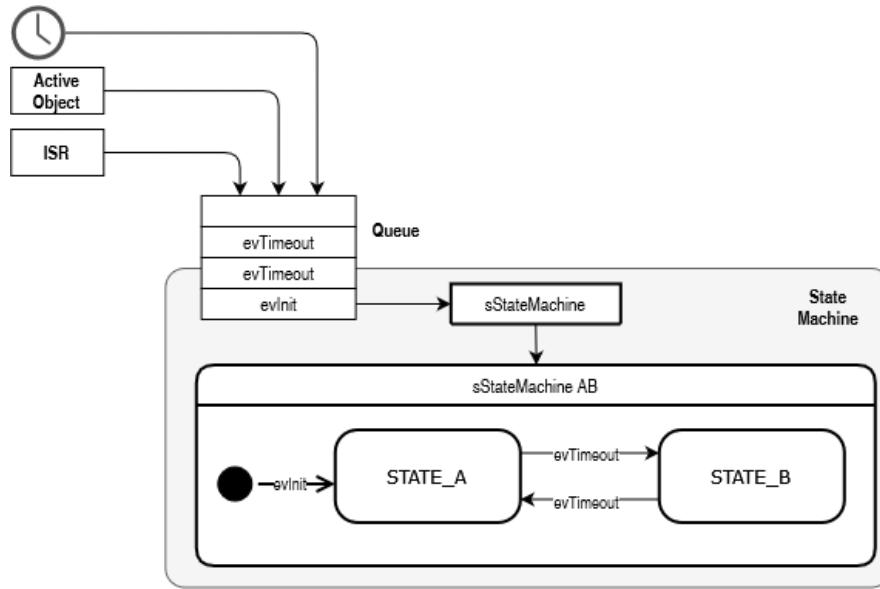


FIGURA 3.9. Diagrama del objeto activo de la máquina de estados AB ejemplo.

En este caso 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 podrían ser generados también por otros objetos activos o por rutinas de interrupción. Los mensajes en cola se reciben en la tarea de sistema que actualiza la máquina de estados.

#### 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.

La organización del scheduler del sistema operativo se representa con el diagrama temporal de la figura 4.2. La base de este diagrama es el orden de prioridades

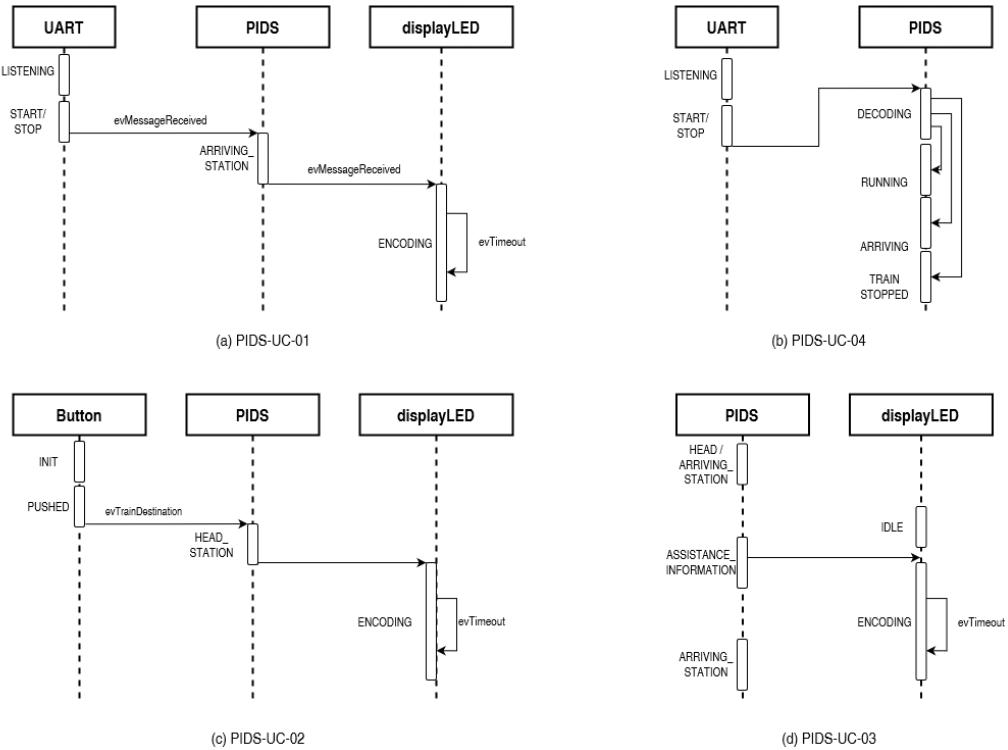


FIGURA 3.10. (a)caso de uso de visualización de estación; (b) caso de uso de receptor de tramas;(c) caso de uso de elección de destino por accionamiento del conductor;(d) caso de uso de visualización de información de asistencia.

de las tareas de objetos activos, los timers y los handlers de interrupciones que manejan los periféricos de hardware.

## UART

En esta sección se describe la implementación del componente objeto activo UART (Universal Asynchronous Receiver Transmitter) del sistema. 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 "echo": 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, se ha observado también que los mensajes del tren pueden tener largo variable. El diseño de este

componente se representa con el diagrama de la figura 3.11.

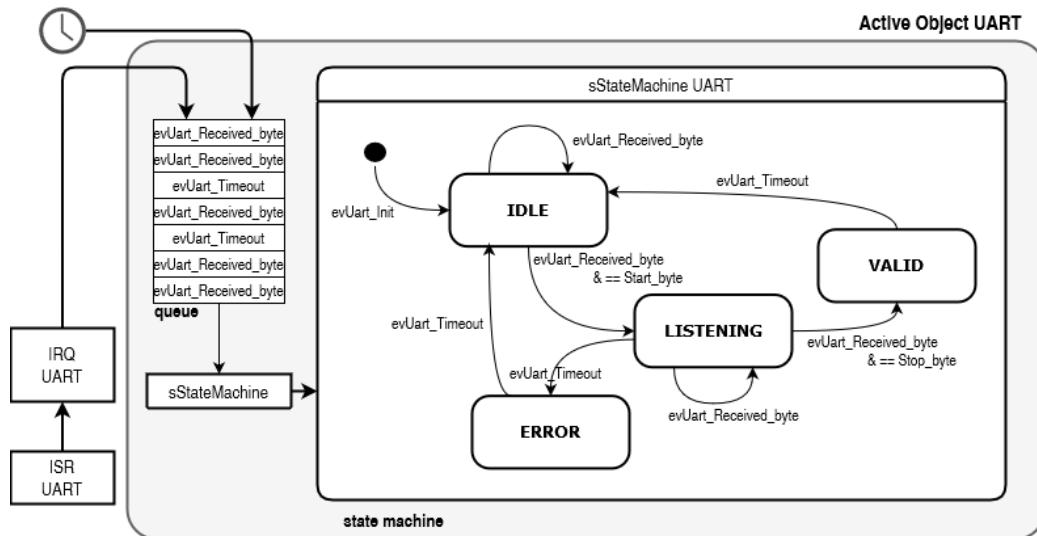


FIGURA 3.11. Diagrama del objeto activo UART implementado.

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 diseñó un componente UART flexible, que permite cambiar los bytes de header y trailer y admite un buffer de largo variable usando memoria dinámica.

### Display LED

El objetivo de esta sección es describir la solución implementada para el manejo de carteles de matriz led, su estructura, su forma de uso y cómo se debe modificar en caso de reutilización en otros sistemas.

Los carteles led de esta se basan en una tecnología de microcontrolador (MCU) y sistema digital. El MCU genera las señales de control y el sistema digital se encarga de codificar los datos del micro en señales eléctricas para prender leds de forma ordenada. Los display se componen de módulos de matriz led de 8x8. Estos módulos se direccionan por filas y columnas para visualizar información. Los arreglos de estos módulos forman paneles, y estos últimos pueden ser direccionados por un grupo de señales de control. En la figura 3.12 se presenta el diagrama de bloques del sistema de control de los carteles de matriz led.

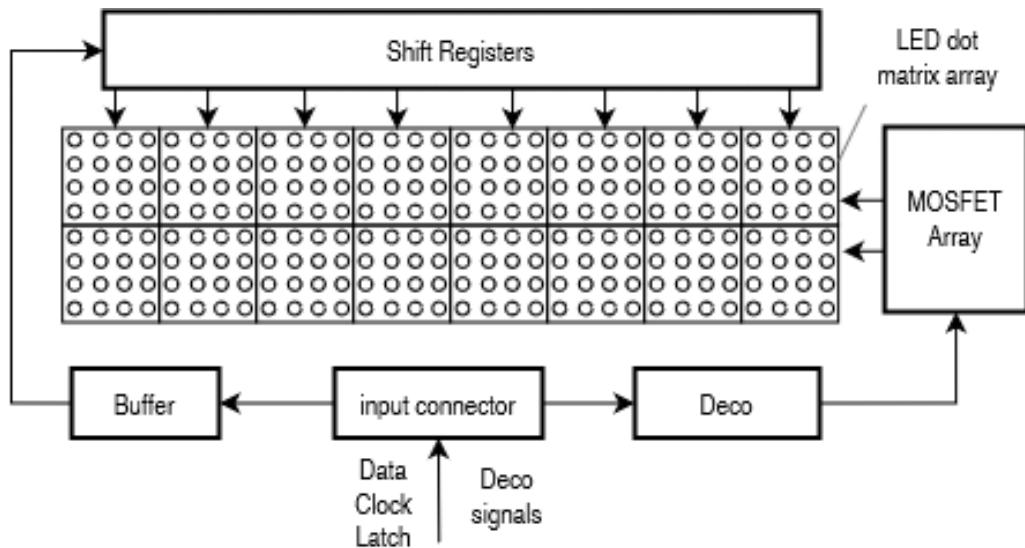


FIGURA 3.12. Diagrama de bloques del controlador de los carteles de matriz LED utilizados en esta implementación.

Las señales de control (data, clock, latch, deco) se generan en el MCU y se transmiten al circuito eléctrico del display led a través de un conector (CONN). Estas señales se direccionan eléctricamente usando buffers. Luego, hacia la izquierda, se transmiten las señales data, clock y latch, que envían los datos del mensaje codificado a un arreglo de Shift Registers. Estos últimos son los encargados de transmitir bit a bit los datos para las columnas de cada módulo que forman una fila completa del cartel. Una vez enviados todos los datos de una fila, se envía un pulso como señal de latch y se energiza la fila completa. Se repite la operación fila a fila escaneando todo el cartel a través de las señales del Deco, energizando cada fila a través de un arreglo de transistores MOSFET que mantienen la corriente necesaria para encender los leds. Este encendido por fila sucede durante un período de tiempo tal que toda la pantalla se pueda energizar en el orden de 20 a 50 veces por segundo para formar una imagen continua vista por el ojo humano.

En el circuito esquemático de la figura 3.14 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.

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 3.15.

Se pueden distinguir tres estados, cada uno con un handler asociado. En el siguiente fragmento de código se presenta la especificación de la máquina de estados del display led.

```

1 sStateMachine_displayLed fsmDisplayLed[] =
2 {
3     {STATE_DISPLAYLED_INIT, evDisplayed_init, displayed_initHandler},
4     {STATE_DISPLAYLED_IDLE, evDisplayed_msg_received,
5         displayed_idleHandler},
6     {STATE_DISPLAYLED_PROCESSING, evDisplayed_timeout,
7         displayed_procHandler},
8     {STATE_DISPLAYLED_ENCODING, evDisplayed_timeout,
9         displayed_dataHandler}
7 };

```

CÓDIGO 3.1. Display Led state machine definition

```

1 eSystemState_displayLed     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    int 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;
30 }
```

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

El estado 'STATE\_DISPLAYLED\_PROCESSING' ejecuta el handler 'displayled\_procHandler()' encargado de recibir mensajes externos en formato string (char\*). Este implementa un pipeline de procesamiento para transformar strings en matrices de enteros de 8 bits.

```

1 eSystemState_displayLed    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<displayled_size; i++){
13        data_8b = displayed_buffer[i];
14
15        for(int j=0; j<8; j++){
16            // displayed_data
17            value = (((data_8b << j) & 0x80) == 0) ? 1 : 0;
18            printf("%d", value);
19            gpioWrite(displayled_panel_1, value);
20            gpioWrite(displayled_panel_2, value);
21
22            // displayed_clock
23            gpioWrite(displayled_clk, ON);
24            gpioWrite(displayled_clk, OFF);
25        }
26
27        if (i%DISPLAYLED_COLS==0){
28
29            // displayed_latch
30            printf("\r\n");
31            gpioWrite(displayled_latch, ON);
32            gpioWrite(displayled_latch, OFF);
33
34            // displayed_row_scanning
35            displayed_deco_cnt++;
36            displayed_deco_cnt%DISPLAYLED_ROWS;
37            if ((displayed_deco_cnt%1)==0){
38                gpioToggle(displayled_deco_A0); }
39            if ((displayed_deco_cnt%2)==0){
40                gpioToggle(displayled_deco_A1); }
41            if ((displayed_deco_cnt%4)==0){
42                gpioToggle(displayled_deco_A2); }
43            if ((displayed_deco_cnt%8)==0){
44                gpioToggle(displayled_deco_A3); }
45
46        }
47    }
48
49 }
```

```
51     return STATE_DISPLAYED_ENCODING;  
52 }
```

CÓDIGO 3.3. Código fuente del handler de encoding del display  
Led.

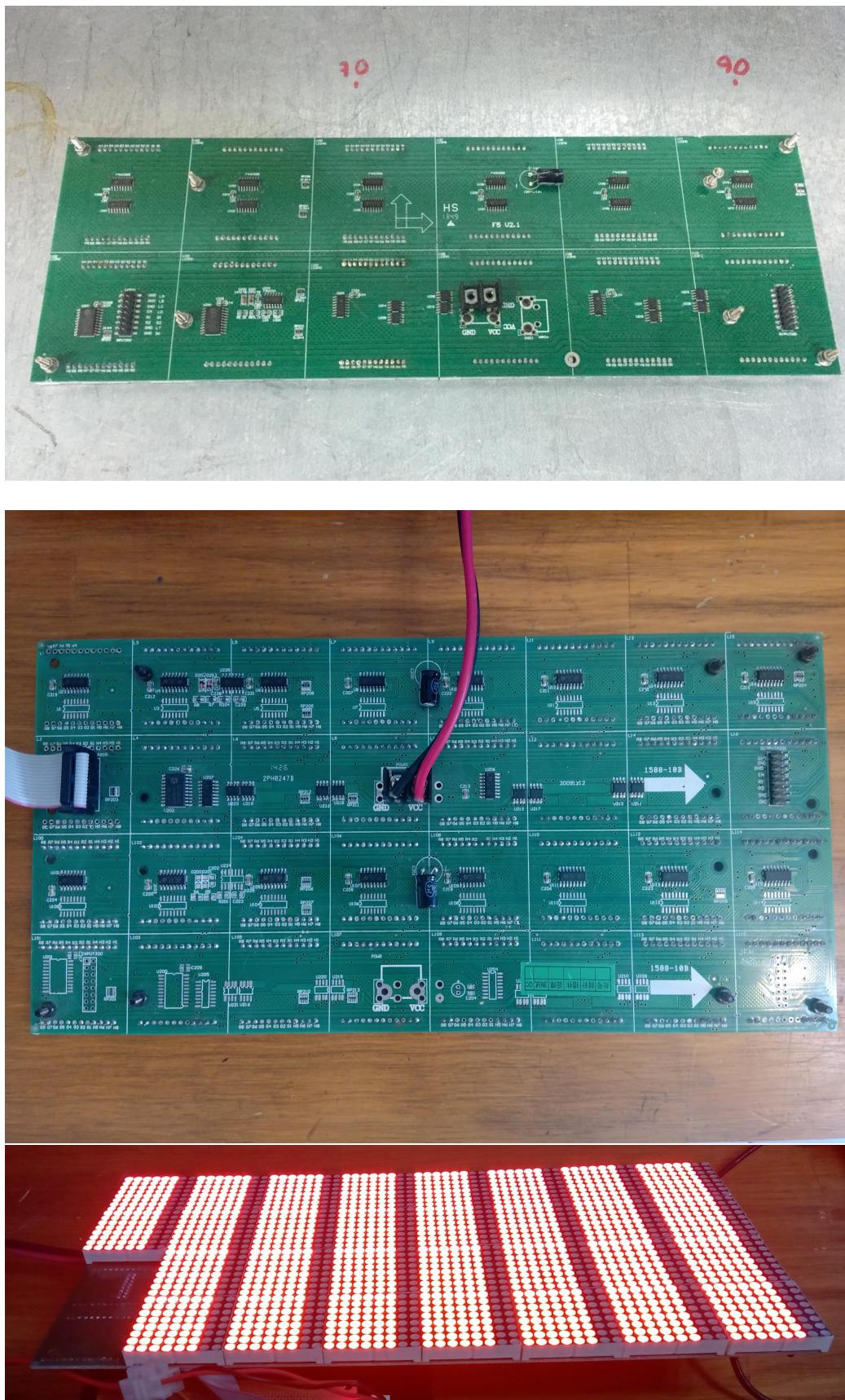
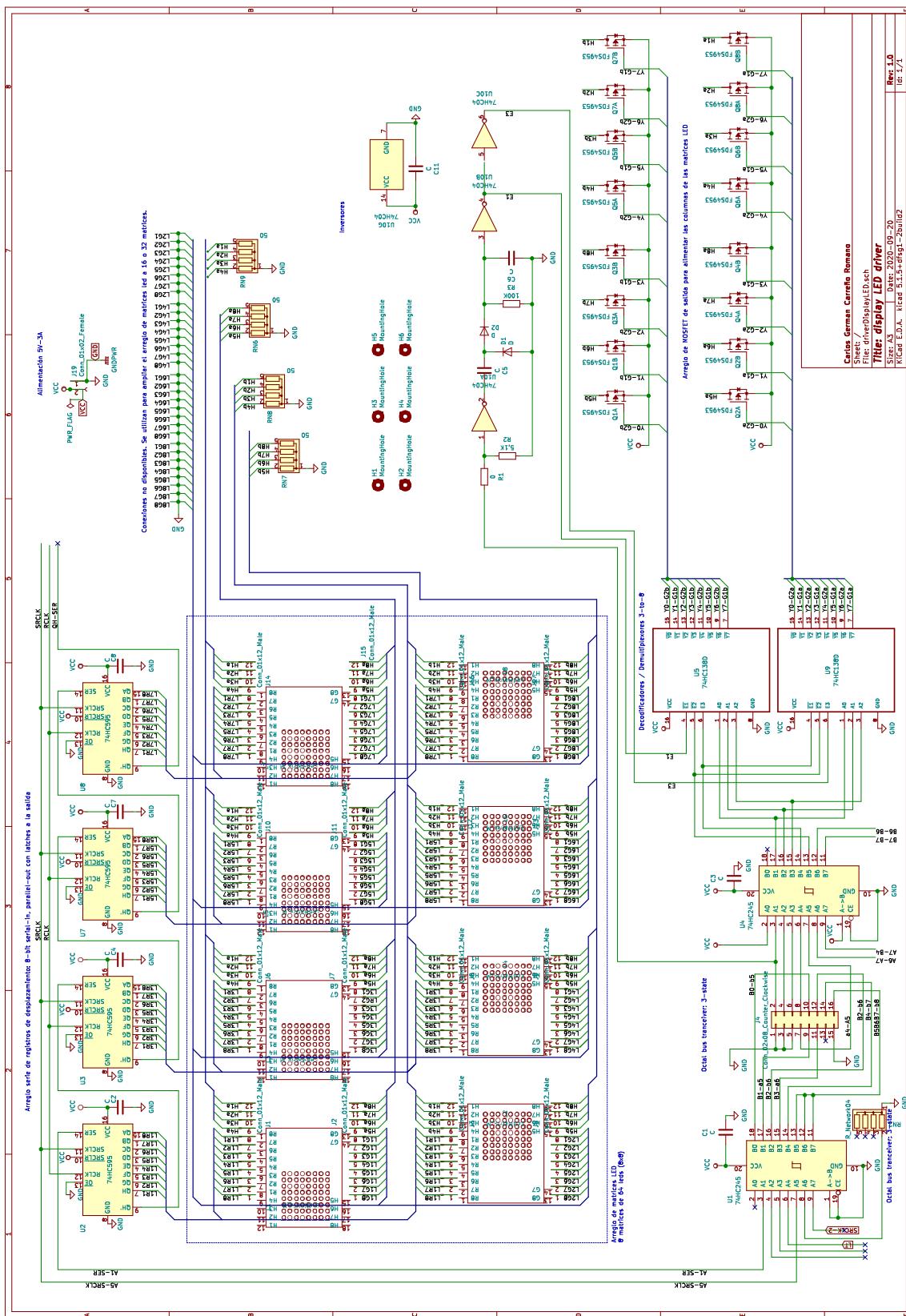


FIGURA 3.13. Fotografías de placas de control de los carteles de matriz LED: (a) placa de 2x6 módulos; (b) placa de 4x8 módulos; (c) vista posterior de la placa de 4x8.



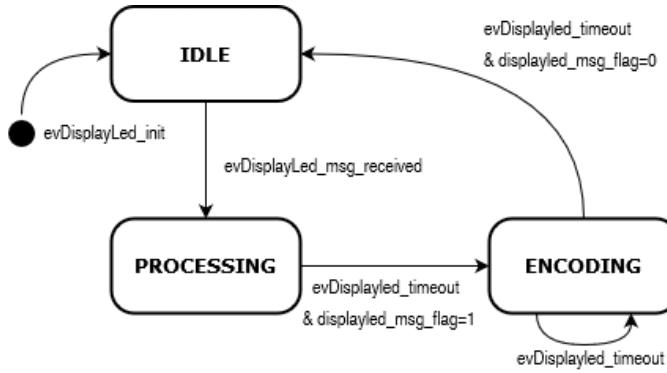


FIGURA 3.15. Diagrama de estados para la máquina de estados del display Led.

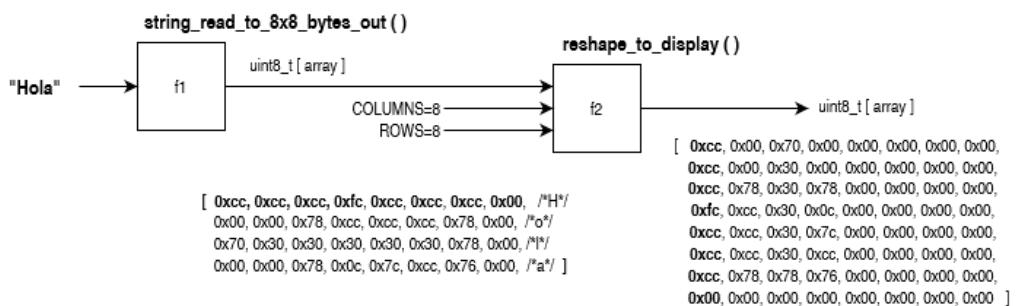


FIGURA 3.16. Lógica de procesamiento de datos para visualizar en el display.



## 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

## Placa de control

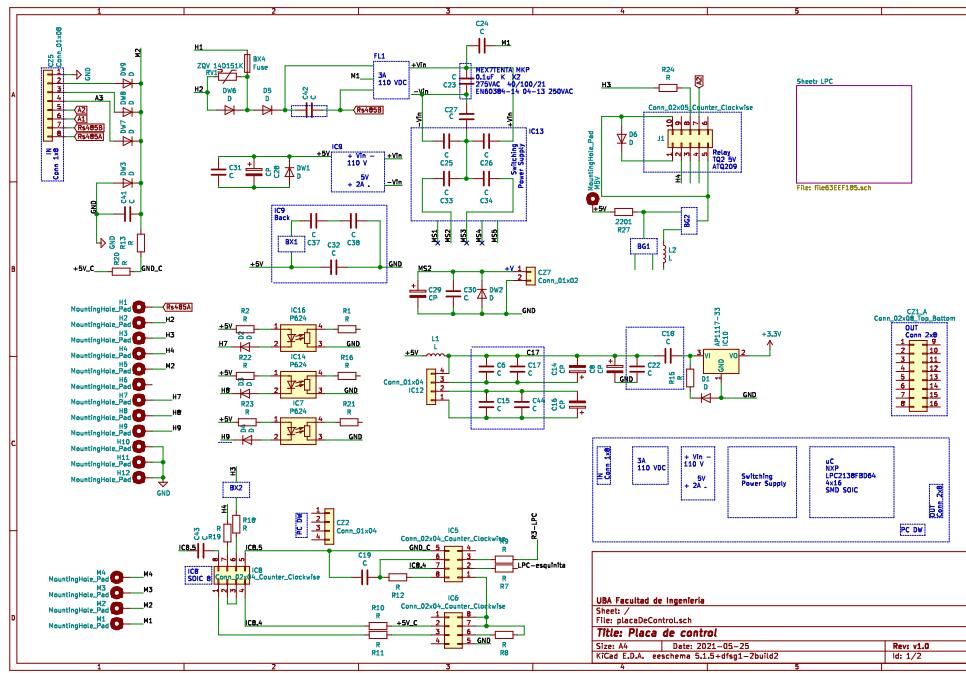


FIGURA 4.1. Circuito esquemático de la placa de control de los carteles LED de salón.

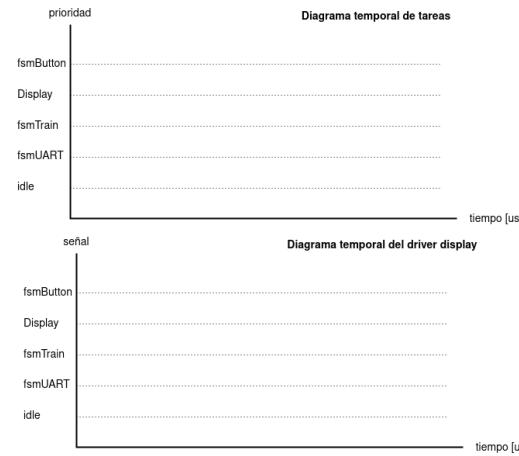


FIGURA 4.2

#### **4.4. Integración con red PIDS**

## **4.5. Pruebas de campo**



## 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] .
- [2] .
- [3] .
- [4] .
- [5] .
- [6] 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.
- [7] 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.
- [8] W Kurdthongmee. *Design and Implementation of an FPGA-based Multiple-color LED Display*. 2004.
- [9] 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.
- [10] 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.
- [11] .
- [12] Proyecto CIAA. *Computadora Industrial Abierta Argentina*. Visitado el 2016-06-25. 2014. URL: <http://proyecto-ciaa.com.ar/devwiki/doku.php?id=start>.
- [13] Eric Pernia. *firmware v3*. Visitado el 2023-04-07. 2021. URL: [https://github.com/epernia/firmware\\_v3](https://github.com/epernia/firmware_v3).
- [14] Richard Barry. *Free Real Time Operative System*. Visitado el 2023-04-07. 2016. URL: [https://freertos.org/Documentation/RTOS\\_book.html](https://freertos.org/Documentation/RTOS_book.html).
- [15] Pablo Gomez. *PDE 15 2020 SISTEMA DE MONITOREO Y GESTIÓN DE LA RED TCN EN FORMACIONES FERROVIARIAS*. Visitado el 2023-04-07. 2021. URL: [https://github.com/epernia/firmware\\_v3](https://github.com/epernia/firmware_v3).
- [16] 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>.