Tesis de Maestría

Metodología para la operación de un PLC mediante una PDA vía ZigBee

Por

JUAN GONZALO ZULUGA BOTERO

Maestria en Automatización y Control Industrial
Facultad de Ingenierías
Instituto Tecnológico Metropolitano
Medellín, Colombia
2014

Metodología para la operación de un PLC mediante una PDA vía ZigBee

JUAN GONZALO ZULUAGA BOTERO

Tesis de grado para optar el titulo de:
Magister en Automatización y Control Industrial

Director:
PhD. Jorge Herrera Cuartas

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
2014

A ese árbol mágico donde se fomenta la unión, a ese tesoro grande que se alegra por los demás y comparte cada momento vivido, a mi flor que amo y fecunda esas semillas de alegría que crecen cuya educación son todo un reto, cuyo conjunto de elementos permiten lograr la perseverancia.

A la Divinidad que ayuda a entender que la vida está conformada con fines parciales para luego llegar a un fin infinito, que junto con el conocimiento hace que pueda comprender que el pensamiento va más allá de las palabras y la valoración más allá del pensamiento y que las dificultades no se ven sino hasta cuando se vivencian.

Índice general

Lista de Figuras

Índice de Tablas

Agradecimientos

Primero que todo a Dios en su trinidad con el Espíritu Santo y sus acompañantes, por iluminarme y darme las esperanzas sin desfallecer ante situaciones de desesperación.

A mi director Jorge Herrera Cuartas, por el apoyo, colaboración, empeño, dedicación y lo más importante, por creer en mí que sin él no hubiera sido posible llevar esto a un feliz término. Me enseño a pensar con espíritu de investigación además de las orientaciones en el pensamiento y atacar los problemas con ángulos diferentes. Gracias por la motivación brindada y la estructuración de todo lo planteado en artículos e informes.

A mi esposa y mis 2 adoradas hijas por la paciencia que me han tenido al compartir poco tiempo con ellas y en especial a mi esposa que aunque sabía a grandes rasgos como se presentaban las cosas, también creyó en mí, apoyándome para tener el éxito esperado.

A mi familia en general, por apoyarme con sus oraciones que siempre han querido mi éxito y me han apoyado al respecto, creyendo en mi e impulsarme a realizar estos estudios.

A los compañeros del laboratorio de PLC por su apoyo y paciencia en todo lo relacionado con préstamo de equipos Yuly Castro y Ruben Fonnegra, además de toda la amabilidad brindada en las situaciones presentadas. Al ITM y su grupo de Investigación en Automática, Electrónica y Ciencias Computacionales por el préstamo de equipos y la orientación en el manejo de estos y su apoyo en el proceso que se llevó a cabo.

En general a todos como: compañeros de estudio Delio Aristizabal, Leonardo Serna, compañeros grupos de investigación, investigadores, asesores y todas aquellas personas que se me escapan que aunque sé que fueron muchas, lastimosamente se pasan por alto pero sé que pusieron su granito de arena para llevar a un feliz término este proyecto.

Resumen

En este trabajo se presentan los pasos realizados para lograr la estructura metodológica para el diagnóstico de fallas de un PLC (*Programmable Logic Controller*) que se conecta a una red Zigbee mediante una PDA (*Personal Digital Assistant*) monitoreando el sistema. Para su implementación se realizan un conjunto de comunicaciones establecidas por fases, empalmando cada una de las tecnologías utilizadas para finalmente integrarlas en una sola estructura realizando pruebas de diagnóstico y monitoreo del sistema.

Para lograr el objetivo del proyecto, se inicia con la comunicación entre los dispositivos móviles y el protocolo IEEE 802.15.4 Zigbee, con tramas de datos básicas a través de la red de telefonía móvil celular GSM (Global System for Mobile), mediante mensajería corta recibida por el puerto serial del Zigbee remoto, utilizando microcontroladores para el control de los módulos que conforman el sistema de comunicación. Con esto se establece la comunicación entre el dispositivo móvil y los módulos Zigbee.

La implementación de la comunicación a través de las tramas Profibus, permiten detectar fallas típicas de los PLC, utilizando un microcontrolador como un esclavo mediante el acople de un Gateway, garantizando una comunicación confiable y una red de apertura a cualquier medio. Para su implementación se realiza un estudio de las tramas utilizadas entre un PLC maestro y un PLC esclavo para luego crear un programa en el microcontrolador que por análisis de tramas y respuestas específicas establece una interconexión en tiempo real.

Al integrar las interfaces de comunicación, se logra el diagnostico de fallas mediante mensajes recibidos en una PDA, obteniendo mejores tiempos de reacción y permitiendo optimizar el funcionamiento de la red industrial, ya que mediante la detección del error de manera precisa, concreta e inmediata, evita la inspección de errores, mejorando la productividad y eficiencia de una empresa con control industrial.

Palabras clave Zigbee; PLC;PDA; Profibus; Comunicaciones en control industrial.

Abstract

This article show the steps achieved for methodology structure of PLC (Programmable Logic Controller) fault diagnosis in Zigbee network via PDA (Personal Digital Assistant) to system monitor. For implementing a set of communications is performed by sections, join each technologies used for next integrate them into a single structure to check the system with diagnosis and monitoring.

For archieve the aim, this proyect starts with the communication between mobile device and IEEE 802.15.4 Zigbee protocol, with basic data frames through the cellular mobile telephone network GSM (Global System Mobile), by means of short messages by remote ZigBee serial port using microcontroller for managing the communication system modules. This communication is established between the mobile device via ZigBee.

The implementation of communication through Profibus frames, allowing to detect PLC failures typical. The microcontroller works like a slave by means of a Gateway, ensuring reliable communication and oppening a network to any means. In order to implementation it, a study of frames between a master PLC and slave PLC was realized, creating a microcontroller software that by means of frames analysis and specifics answers establishes a interconnection in real time.

By integrating communication interfaces, fault diagnosis is accomplished by messages received on a PDA, get better reaction times and optimezes the operation of the industrial network, because the error detection is precise and concrete, avoiding the error inspection, improving the productivity and efficiency of a company with industrial control.

Keywords Zigbee; PLC; PDA; Profibus; Communications in industrial control

INTRODUCCIÓN

El control industrial se ha integrado con los sistemas de comunicaciones y la informática industrial (Figueiredo y Costa, 2007), permitiendo movilidad permanente mediante conexión a Internet y redes móviles. Mediante esta red se realiza la gestión, el monitoreo y el control de un conjunto de PLCs o también conocidos como Controladores Lógicos Programables. De esta manera se puede implementar un sistema SCADA (Supervisory Control And Data Adquisition) para el intercambio de variables y señales de control, permitiendo una administración remota de la planta entera. A modo de ejemplo, en (P.-H. Wu, Lin, Kuo, y Wu, 2006) se realiza el control de un PLC mediante SMS (Short Messenger Service), donde se transmiten solo estados a las salidas digitales del PLC y este informa sus estados al celular, mediante un microcontrolador.

Desde hace varios años, se han aplicado estrategias de control en lazo cerrado sobre redes de comunicaciones (Chan y Ozguner, 1994), igualmente se han desarrollado investigaciones como teleoperaciones de sistemas mediante una red de computadores en ambientes dinámicos (Kikuchi, Takeo, y Kosuge, 1998). En (R.-C. Wu, Wu, Teng, y Huang, 2007), se implementó un control remoto inalámbrico de un PLC conectado a una red Ethernet a través de una PDA, el cual se caracteriza por el manejo y monitoreo de un servomotor que está conectado a un PLC y este a una red con un servidor inalámbrico.

Por otro lado, en (Pengfei y Jiakun, 2009), realizaron una conexión entre un PLC y un computador via ZigBee, la cual se logra mediante el protocolo RS485 de manera unidireccional y se visualizan los datos en el programa LabView. Igualmente se han utilizado conexiones mediante bluetooth y la red celular para realizar procesos de facturación de abonados eléctricos (Salazar Jairo, 2008) y monitoreo de sistemas de conducción de aguas vía GSM en redes de Empresas Publicas de Medellín.

Siguiendo esta misma línea, (Xu, Xu, y Xu, 2012), hicieron control bidireccional entre un PLC y un microcontrolador ARM via ZigBee. En este hay monitoreo, almacenamiento de parámetros y análisis de los datos recibidos con capacidad para detener el PLC si estos no cumplen un estándar predeterminado. Recientemente, se tienen investigaciones con sensores ZigBee y redes de sensores con estos dispositivos para el tratamiento en hogares (Y. Wu y Shao, 2012), sin embargo, hay vacíos en las comunicaciones de estos con los PLC, puesto que no utilizan la red Profibus

y tampoco definen un perfil PLC-ZigBee el cual esta compuesto por un conjunto de mensajes o comandos para aplicaciones industriales (*ZigBee Alliance*, s.f.) y diagnóstico a través de las PDAs.

Cuando se integra el control industrial con las comunicaciones, se evalúa la confiabilidad y se diagnostica el funcionamiento de los sistemas (Gaeta, Bobbio, Franceschinis, y Portinale, 2001). El mantenimiento y monitoreo en red, según se establece en (Palluat, Racoceanu, y Zerhouni, 2006) es una parte fundamental en el desarrollo de los proceso productivos, pues estos han permitido detectar fallas a tiempo y errores críticos que pueden evitar daños mayores.

En este trabajo, se propone una metodología para implementar la comunicación entre un PLC y una PDA usando una red ZigBee. De esta manera, se obtiene una monitorización y control del proceso. Aprovechando las potentes características de las PDA y su facilidad de programación se obtiene un incremento en la eficiencia y el desempeño de la planta, como también un diagnóstico oportuno de los errores y fallas básicas que puede presentar un PLC. Teniendo en cuenta que, las conexiones inalámbricas en la industria se han incrementado considerablemente y que el protocolo ZigBee cumple con los requerimientos en un ambiente industrial (Park, Ku, Lee, y Moon, 2007), se puede considerar factible su implementación en la industria colombiana.

Para el desarrollo planteado, se realiza una división de 3 fases de operación. La primera es la conectividad entre la PDA y el sistema ZigBee, una segunda fase es la comunicación entre el PLC y ZigBee para lo cual se utiliza un microcontrolador que establece el puente con la red Profibus del PLC. Finalmente, la tercera fase consiste en la integración de las implementaciones realizadas en las dos fases anteriores.

Justificación

Debido a que los sistemas de comunicación Profibus son muy utilizados a nivel industrial por los PLCs y estos se convierten en una herramienta fundamental para la conexión de sensores, actuadores, otros PLCs, además de otros módulos (Guerrero, Yuste, y Martínez, 2012); es importante conocer este protocolo e implementarlo en sistemas embebidos, estableciendo este como un camino de comunicación.

Siguiendo esta misma línea y mediante un estudio previo de las empresas HACEB, PILSEN y SOFASA, se encontró, que estas tienen su control industrial mediante arreglos de 15 o más PLCs, pero solo 1 o 2 de ellos tiene conexión directa en red con un servidor remoto y con un aplicativo SCADA y ninguna empresa lo tiene mediante PDA; lo que quiere decir que es un campo abierto de investigación en la empresa colombiana y sobre todo en el área del diagnóstico de fallas.

Según estadísticas, en las empresas colombianas el 23 % del presupuesto es invertido en mantenimiento correctivo en vez de preventivo (Nieto, 2009), debido a que no hay una buena verificación de los sistemas de control industrial, que permitan la monitorización y el diagnóstico en tiempo real de los procesos. En la década

de los 70 en Colombia, el área que generaba mayor empleo en las empresas manufactureras y de producción era la de mantenimiento, pero por acondicionamientos justos del sistema hoy en día no se necesita tanto personal como anteriormente (Nieto, 2009).

Continuando con aspectos de la industria colombiana, se tiene, que la Licorera de Caldas, pierde entre 60 y 80 millones de pesos en utilidades netas, al tener parada una línea de producción en una semana para realizarle mantenimiento correctivo y preventivo a los dispositivos electrónicos y mecánicos del sistema de control (González Daniela, 2009).

Teniendo en cuenta que en la integración del control industrial con las comunicaciones, se evalúa la confiabilidad y se diagnostica el funcionamiento de los sistemas (Gaeta y cols., 2001). El mantenimiento y monitoreo en red, según se establece en (Palluat y cols., 2006) es una parte fundamental en el desarrollo de los proceso productivos, pues estos han permitido detectar fallas a tiempo y errores críticos que evitan daños mayores, por lo que es necesario mejorar los procesos de mantenimiento y control en las empresas Colombianas, sobre todo cuando este es posible realizarlo en tiempo real.

Por lo anterior, al desarrollar una estructura metodológica para el diagnóstico de fallas de un PLC y establecer una comunicación Profibus con este, se puede determinar un modelo de comunicación efectiva que informe sobre el diagnóstico de fallas comunes de un PLC y que al ser procesadas por un microcontrolador, este pueda categorizar el tipo de falla.

Planteamiento del Problema

En consecuencia de que se presentan pérdidas por mantenimiento en la industria colombiana (González Daniela, 2009) y considerando que no se tienen concretamente metodologías que relacionen una PDA con el diagnóstico básico de las fallas internas de los PLC vía ZigBee; la metodología propuesta en este trabajo da una proyección del control industrial remoto. Es necesario implementar nuevas aplicaciones, que permitan mejorar la movilidad en la ingeniería de procesos al igual que en los sistemas de mantenimiento y control en las empresas colombianas, permitiendo soporte en tiempo real.

De esta manera se puede aumentar la eficiencia y desempeño de la industria solucionando el problema actual del porcentaje de dinero que invierten las empresas en mantenimiento; gracias a las alertas indicadas en los dispositivos móviles de las fallas de los PLCs. Evitando además la incompatibilidad de los medios de transmisión o protocolos utilizados por un fabricante especifico de PLC, siendo esto adaptable a cualquier tipo de proceso, haciendo de este trabajo una tecnología transversal.

En síntesis se busca resolver los problemas que se tienen en cuanto al diagnóstico de fallas de los PLC en estandarización con el protocolo ZigBee, buscando una movilidad mediante PDAs. Algunos problemas presentados en una planta no

son reportados en tiempo real al Jefe de mantenimiento, de modo que se pueden presentar retardos en las soluciones y/o en el mantenimiento del proceso productivo. Por esta razón este proyecto contribuye a realizar mantenimientos preventivos con alertas indicadas en los dispositivos móviles inmediatas que sean compatibles con diferentes PLCs.

Hipótesis

Mediante la implementación de técnicas que permiten la conexión directa entre el PLC y el ZigBee, con un protocolo definido para dicho perfil, se establecen las pruebas que definen el diagnóstico de fallas del PLC y que estas sean reenviadas a la PDA, estableciendo una metodología de conexión que permita realizar mantenimiento preventivo y/o correctivo. Al realizar este diagnóstico se verificará el comportamiento del PLC ante eventos fortuitos determinando las posibles fallas más comunes de estos y se evaluará la eficiencia del perfil de comunicaciones entre PLC y la PDA vía Zig Bee.

Objetivos

General

- Desarrollar una estructura metodológica para el diagnóstico de fallas de un PLC que se conecta mediante una red Zig Bee a una PDA logrando intervenir mediante control y monitoreo del sistema.

Específicos

- Caracterizar la comunicación de un PLC con una PDA via ZigBee definiendo un perfil, orientado al control operativo.
- Determinar un modelo de comunicación efectiva que informe sobre el diagnóstico de fallas comunes de un PLC a la PDA en el monitoreo y control.

Glosario y Acrónimos

- API: (*Application Programming Interface*). Comando de aplicación que permiten programar y configurar la interfaz de comunicación ZigBee.
- APS: (*Application support Sublayer*). Es una capa de ZigBee utilizada para el manejo y entendimiento de aplicaciones, la cual brinda servicios a nivel de red.
- AT: (*Attention command*). Comando de control de interfaces seriales precedido por sus letra AT para el envío de parámetros de control.
- Broadcast: (*Emisión, Difusión*). Trama de envío o envío de información simultáneamente a varios dispositivos en un mismo instante de tiempo como la Televisión o Radiodifusión.
- Cluster: (*Grupo, Raiz*). Conjunto de mensajes de aplicación definidos dentro de un perfil para especificar función, acción o servicio.
- EN: (*European standards*). Estándares europeos para productos y servicios dados por el comité de estandarización europea.
- FB: (*Function Block*). Bloque de funciones programado en el PLC con memoria
- FBD: (*Function Block Diagram*). Lenguaje de programación gráfico del PLC, basado en funciones incorporadas en bloques y unidos en diagramas.
- FC: (*Function Code*) Función programada en el PLC sin memoria.
- FDL: (*Fieldbus Data Link*). Capa de Profibus en el nivel de enlace para comunicación de dispositivos industriales.
- FMA: (*Fieldbus Management Application*). Capa de Profibus en el nivel de aplicación manejando protocolos y funciones específicas.
- Full-Duplex: Transmisión y recepción simultanea entre 2 dispositivos.
- GAP: (*Groups, Algorithms, Programming*). Sección de manejo de información en el paso de testigo de Profibus, encargada de trabajar parámetros de algoritmos, programación y grupos.
- GAPL: (*Group Algorithm Programming List*). Es la lista de maestros en la red que se encuentran agrupados para programarlos en el algoritmo de paso de testigo.
- Gateway: Equipo que permite enlazar 2 protocolos de comunicación diferentes con diferentes niveles.

- GSM: (*Globlal System Mobile* o *Group Special Mobile*). Estándar para las comunicaciones móviles digitales, iniciando con la tecnología celular de segunda generación, permitiendo mensajería y datos de manera segura.
- HSDPA: (*High Speed Downlink for Packet Access*). Acceso para Descarga de Paquetes con Alta Velocidad vía radio usando la red de telefonía móvil celular conocida como la evolución de UMTS es decir 3.5G.
- IEC: (*International Electrotechnical Commission*). Organización de normalización en los campos eléctricos, electrónicos y tecnologías relacionadas, fundada desde 1906. Adicionalmente certifica equipos con estándares internacionales.
- IEEE: (*Institute of Electrical and Electronics Engineers*). Instituto de Ingenieros Eléctricos y Electrónicos.
- IL: (*Instruction List*). Es un lenguaje de programación del PLC en ensamblador.
- LD: (*Ladder Diagram*).Lenguaje de programación gráfico del PLC, basado en lógica de contactos.
- LSAP: (*Link Service Access Point*). Es el servicio de enlace de las estaciones que se encuentran en el punto de acceso del anillo Profibus.
- LTE: (*Long Term Evolution*). Es la evolución de la telefonía móvil celular dada por un periodo largo de tiempo o a largo plazo, conocida como red de cuarta generación (4G).
- NRZ: (*Non Return to Zero*). Código de línea binario usado en telecomunicaciones donde el pulso en la mitad del ciclo no retorna a 0.
- OSI: (*Open System Interconnection*). Modelo de redes de computadoras descriptivo para su conformación en capas organizado por la ISO en 1977.
- Payload: (Carga Útil). Datos reales que se transfieren sobre un medio. Determina el rendimiento de la transmisión.
- PDA:(*Personal Digital Assistant*). Asistente Digital Personal
- PLC: (*Program Logic Controller*). Controlador Lógico Programable.
- PLL: (*Phase Lock Loop*). Sistema de control de lazo cerrado que toma la señal de oscilación generada y la compara constantemente con la señal de salida verificando su correcto sincronismo.
- PG: (*Plus Grade*). Portátil industrial de Alto grado de rendimiento para conexión de PLCs.

- Profibus: (*Process Field Bus*). Estándar de comunicación para buses de campo en los procesos de automatización industrial promovido en 1989.
- Profibus FMS: (*Profibus Fieldbus Message Specification*). Subtipo del Profibus que funciona mediante el envío de mensajes.
- Profibus DP: (*Profibus Distributed Peripheral*). Subtipo de Profibus utilizado en aplicaciones comunes a nivel industrial para interconectar los dispositivos típicos.
- Profibus PA: (*Profibus Process Automation*). Subtipo de Profibus utilizado para aplicaciones industriales de automatización que impliquen riesgo, confiabilidad y respaldo.
- Router: (*Enrutador-Rúter*). Dispositivo encargado de encaminar paquetes brindando conectividad entre los elementos de la red, aceptando conexión de nuevos dispositivos.
- SCADA: (*Supervisory Control And Data Adquisition*). Sistema de Supervisión, Control y Adquisición de datos.
- SFC: (*Sequential Function Chart*). Lenguaje de programación gráfico del PLC basado en ejecuciones secuenciales y estados, donde en cada estado se ejecuta una función o conjunto de funciones.
- SMS: (*Short Messenger Service*). Servicio de la telefonía móvil, que permiten enviar mensajes de texto cortos con una capacidad de 160 caracteres sobre la red GSM, inventado desde 1985.
- ST: (*Structured Text*). Lenguaje de programación de medio nivel similar al Pascal en su estructura.
- Standby o Stand-by: (*del español espera, disponible*). Consumo en espera o modo de espera de diferentes dispositivos electrónicos o buses de comunicación, preparados para realizar alguna función específica o con disponibilidad para realizar una actividad.
- UART: (*Universal Asynchronous Receiver Transmitter*). Parte de hardware de un dispositivo encargada de realizar la comunicación serial universal asíncrona para transmisión y recepción de información.
- UMTS: (*Universal Movil Telecomunication System*). Sistema de telecomunicación Universal Móvil aplicado a la telefonía celular de tercera generación (3G).
- Unicast: (*Tx única*). Transmisión de un dato, señal o paquete de información a un único receptor.

- USART: (*Universal Synchronous/Asynchronous Receiver/Transmitter*). Moderno componente de hardware agregados a los ICs (Integrated Circuits) que permite también comunicación Sincrónica serial adicional a la UART para transmisión y recepción de información.
- USB: (*Universal Serial Bus*). Estándar de comunicación serial que define cables, conectores, velocidades, protocolos que permite conectar periféricos y proveer alimentación a través del puerto.
- ZDO: (*Zigbee Device Objects*). Es una capa de punto final soportada en ZigBee que permite manejar clusters sobre un perfil.
- ZigBee: Especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica. Este protocolo es seguro, de bajo consumo, gran alcance y baja velocidad de transmisión.

Organización de la Tesis

Este trabajo está dividido en 4 capítulos. Se inicia con un marco teórico y luego se explica paso a paso el desarrollo y configuración de cada una de las fases. En el Capítulo 1 se dá el Marco Teórico de cada una de las tecnologías usadas y los equipos implementados para el desarrollo de la investigación, en este capítulo se dan a entender los protocolos de comunicación que validan la metodología propuesta. En el Capítulo 2 de este manuscrito se presenta la solución de funcionamiento en la fase 1 donde se conecta la PDA a la red ZigBee. Adicionalmente se describen los dispositivos utilizados con su respectiva configuración, programación, conexión e implementación para el mismo, mostrando las respectivas pruebas realizadas para la verificación de la comunicación. En el Capítulo 3 se muestra la conformación de la fase 2, que corresponde a la conexión del PLC con el microcontrolador vía Profibus, explicando la programación, conexión y hardware del Gateway propuesto con sus respectivas pruebas. Finalmente en el Capítulo 4 se explica la fase 3 con sus características de conexión y el respectivo diseño de experimento con los resultados que validan la implementación de la metodología utilizada.

Capítulo 1

Funcionamiento de los Sistemas Independientes

La aplicación de manera integral como se plantea en (Chan y Ozguner, 1994) en los sistemas de control, busca tener nuevas técnicas de comunicación, logrando visualización en tiempo real de las variables en las plantas industriales. Cuando se establecen estos tipos de conexiones es necesario conocer y entender el funcionamiento de cada uno de los elementos que intervienen en la comunicación industrial estableciendo los parámetros de cada uno de estos y sus principales características.

En el desarrollo de este trabajo se establece la interconexión entre el PLC y la PDA vía ZigBee, por lo que es fundamental conocer y enmarcar el trabajo en 4 aspectos fundamentales como son:

i El PLC: en este punto se describe el PLC con su funcionamiento, la ejecución del programa, los bloques funcionales y la programación.

ii Protocolo de Comunicación Profibus-DP: en esta segunda parte se describe la capa física, la capa de Enlace, los tipos de mensajes, los estados con sus respectivas tramas, los procesos de negociación con tiempos de operación y adicionalmente se dá una breve descripción de la Capa de Aplicación.

iii Comunicación ZigBee: como tercer punto se establecen las condiciones de esta comunicación, los tipos de elementos utilizados, los parámetros de conexión, el enrutamiento y los elementos que la conforman.

iv Modulador GSM: Finalmente se muestran los parámetros y factores que se deben tener en cuenta con respecto a la comunicación de este modulador con el microcontrolador y el estándar como tal.

1.1. Controladores Lógicos Programables

El PLC es un dispositivo que puede controlar una maquina o proceso a partir de un conjunto de filas compuesto por terminales de entrada y terminales de salida, que con instrucciones lógicas, variables en memoria, temporizadores, contadores y bloques de operaciones almacenados en la memoria y ubicados con una secuencia adecuada, ejecutan lo deseado (Hyde, Cuspinera, y Regué, 1997).

En la actualidad el uso de los PLCs se está generalizando en la industria, sin embargo también se encuentran presentes en sectores como la domótica, gracias a que tiene la posibilidad de procesar múltiples variables. Adicionalmente, posee registros de estado especializados que permiten informar posibles fallas o bucles en los que puede entrar el PLC en un instante determinado (García, Castillo, y García, 2009).

Los PLCs poseen diferentes medios de comunicación y transmisión de datos. Así mismo se presentan varios tipos de redes de conexión según la compañía que los produce. Algunos sólo tienen un puerto de programación, otros utilizan este mismo puerto para la comunicación, funcionando mediante USB (Universal Serial Bus) y otros manejan comunicación independiente mediante conexión RS485. En este punto, cabe anotar que se han creado diferentes protocolos de comunicación basados en la conexión RS485, como los son: Modbus, Profibus, Fieldbus, entre otros. Existen protocolos de comunicación que funcionan en sistemas de fibra óptica o par trenzado con el estándar Ethernet, además de algunos otros sistemas independientes como Profinet, CANbus, Hart (*High way Addressable Remote Transducter*), DeviceNet, AS-i.

El PLC al ser un sistema de control secuencial, cambia sus salidas, dependiendo del estado de las entradas y de un programa grabado en la memoria del mismo, que se ejecuta cíclicamente (Ramos, 1998). Desde este programa se pueden realizar llamados a otros bloques funcionales, entre ellos los encargados de tomar las fallas que presenta un PLC ante diferentes eventos, y con este se puede diseñar una aplicación que involucre las variables de función encargadas de la detección de fallas.

El proceso típico de ejecución de un programa en un PLC con la capacidad de detectar fallas se visualiza en la Figura 1.1.

Los bloques funcionales del PLC inician con el bloque principal, donde se ejecuta secuencialmente el programa almacenado. Sin embargo existen otros bloques que pueden trabajar dentro del proceso, como bloques cíclicos o bloques de interrupciones, otros encargados de las fallas que es en lo que se concentra este trabajo, además de los bloques de ejecución simultanea que se ejecutan detrás del bloque principal. Finalmente están los bloques de reinicio del sistema y los de errores del mismo. Estos tipos de bloques con cada una de las funciones vienen especificados según el fabricante, que aunque este trabajo es transversal para cualquier PLC con comunicación Profibus, las pruebas respectivas y los resultados obtenidos se

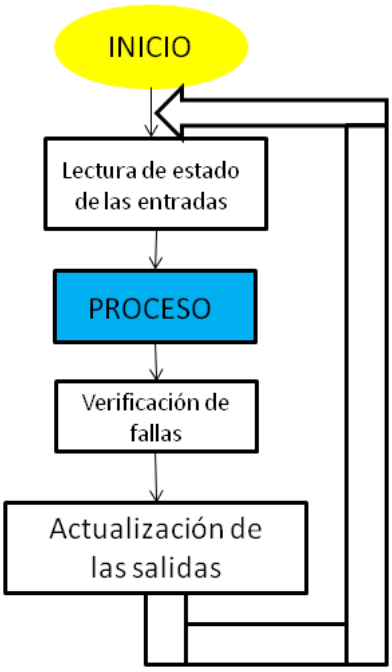


Figura 1.1: Descripción cíclica del flujo de un programa del PLC con detección de fallas.

Fuente: Tomado de (Ramos, 1998)

dieron para un PLC Siemens S7-300, donde sus principales bloques funcionales se pueden apreciar en la Tabla 1.1 (*Siemens Global Website*, s.f.).

La mayoría de los PLC son modulares, es decir, se le pueden insertan bloques funcionales adicionales enlazados por algún protocolo de comunicación. Estos módulos pueden aumentar sus entradas o salidas analógicas o digitales, interfaces de comunicación, entre otras. Cada PLC viene diseñado para trabajar en ambientes industriales, con blindajes especiales que evitan ser afectados por perturbaciones eléctricas o transitorios (Ramos, 1998).

Los PLCs permiten realizar la programación a partir de diferentes lenguajes, siguiendo el sistema normalizado IEC 1131-3, algunos de ellos basados en texto y otros gráficos. Los lenguajes literales están formados por una lista de instrucciones acompañadas de números y símbolos, apareciendo 2 tipos como son el IL (*Instruction List*) el cual es un lenguaje ensamblador y el ST (*Structered Text*) o lenguaje estructurado el cual es similar al Pascal. En cuanto a los lenguajes gráficos existen 3 tipos que son: LD(Ladder Diagram) el cual es un lenguaje con esquema de contactos, FBD(Function Block Diagram) un método gráfico basado en diagramas de funciones y el SFC (*Sequential function Chart*) que es un diagrama funcional de secuencias siendo el más utilizado el GRAFCET (Grafo de Control Etapa-Transición) (Pérez, Acevedo, y Silva, 2006).

BLOQUE	FUNCIÓN
OB10-OB17	Interrupción diaria por tiempo
OB20-OB23	Interrupción por retardo de tiempo
OB30-OB38	Interrupción cíclica
OB40-OB47	Interrupción por hardware
OB60	Interrupción multicomputacional
OB80	Falla en ciclo de tiempo
OB82	Falla I/O en módulo 1
OB83	Falla I/O en módulo 2
OB84	Falla en CPU
OB85	Falla por no carga en bloque OB
OB86	Falla por pérdida de rack
OB87	Falla en la comunicación
OB90	Ciclo atrás del OB1
OB100	Reinicio completado
OB102	Reinicio en frío
OB121	Error en Programa
OB122	Error accediendo al módulo

Tabla 1.1: Bloques Funcionales OB Siemens S7-300.
Fuente:Elaboración Propia con base a (*SIMATIC Programming with STEP 7 V3.0*, 2004)

Con todos los lenguajes de programación que existen bajo la norma IEC 1131-3, se facilita la labor del programador, el cual puede escoger el que más se adapte a su lógica, por esta razón no importa el lenguaje de programación que se elija, la comunicación y detección de fallas se puede realizar de igual manera en forma transversal con el PLC. El programa realizado en el lenguaje seleccionado se ejecuta en el PLC de manera cíclica con un tiempo de escaneo, como el diagrama visto anteriormente en la Figura 1.1, donde el PLC realiza una verificación de las entradas ejecutando secuencialmente el programa y finalizando con la actualización de las salidas físicas y lógicas (Ramos, 1998).

Como la detección de fallas del PLC se implementa con los bloques funcionales dados en la tabla 1.1, la actualización de las salidas no se dará hasta que termine el flujo del programa, es decir, si el tiempo de escaneo del programa es bajo, es porque el programa es corto y la velocidad de respuesta será alta, teniendo una detección de fallas óptima; sin embargo el tiempo de ejecución del programa es lo suficientemente rápido por lo que será despreciable tanto para la comunicación Profibus como para la detección de fallas.

El éxito del PLC se ha debido a que se ha convertido en el elemento central en los procesos de automatización industrial, de modo que el paso de la lógica cableada a la lógica programada, ha hecho de este dispositivo un elemento de fácil implementación y aplicación en los procesos. Su flexibilidad en la programación le

ha dado el carácter de universalidad permitiendo realizar funciones nuevas que no era posible implementarlas con la lógica tradicional. (Pérez y cols., 2006)

La versatilidad, regulación de magnitudes, manejo de variables tanta análogas como digitales, su eficiencia en los procesos ha hecho de este dispositivo un elemento fundamental en la pequeñas y grandes empresas (Pérez y cols., 2006). Al ser un elemento abierto en la interconexión y comunicación con otros dispositivos del ámbito industrial, se pueden formar redes industriales con diferentes protocolos de comunicación como el Profibus el cual se describe a continuación.

1.2. Protocolo de Comunicación Profibus

El Profibus es un estándar de comunicación industrial definido por la norma EN50170 e IEC61158, que permite la comunicación con dispositivos de campo, mediante el cual se puede realizar gestión, monitoreo y control de los dispositivos de un sistema de producción.

Este protocolo es uno de los más implementados a nivel mundial, desarrollado sobre la base del modelo OSI (*Open System Interconnection*) el cual toma tres capas de este modelo, que son la Capa Física, Capa de Enlace y Capa de Aplicación (Ver Figura 1.2.). A su vez se encuentra ramificado en tres perfiles que son Profibus FMS (*Fieldbus Message Specification*), Profibus DP (*Distributed Peripheral*) y Profibus PA (*Process Automation*). La capa de enlace llamada FDL (*Fieldbus Data Link*) esta empalmada con la capa física y estas dos con la de aplicación enlazadas con FMA (*Fieldbus Management*) compuesta por 3 subcapas 1, 2 y 7.

1.2.1. Capa Física

La capa física utiliza el estándar RS485 definido por la EIA (*Electronics Industries Associate*) con norma IEC-1158-2, cuyo estándar permite conectar 32 estaciones por segmento, pero si se realiza una distribución con varios segmentos permite hasta 126 estaciones. Su transmisión es en serie, comunicando a grandes distancias con un volumen de información medio, tiene conexión con la capa de enlace FDL y a su vez estas con la de aplicación FMA(con sus capas 1 y 2) (ver Figura 1.3.). La comunicación utilizada por el bus es asíncrona semidoble, con voltaje diferencial entre los 2 hilos de cobre utilizando el medio de transmisión eléctrico, donde sí la diferencia de los 2 hilos dá un voltaje negativo indica que la transmisión es un 0 y si la diferencia entre los 2 hilos da un voltaje positivo se considera un 1, es decir, se usa un voltaje alto y un voltaje bajo para representar cada bit. La codificación es NRZ(*Non Return to Zero*) con un bit de inicio, 8 bits de datos, paridad par y un bit de parada (Guerrero y cols., 2012).

En esta transmisión de 11 bits se tiene en cuenta que si una estación no está transmitiendo, esta se debe poner en alta impedancia y si en la línea Profibus no hay transmisión esta se debe ver cómo un 1 lógico en espera de inicio de una transmisión. Dentro de Profibus está el perfil Profibus PA que establece una corriente

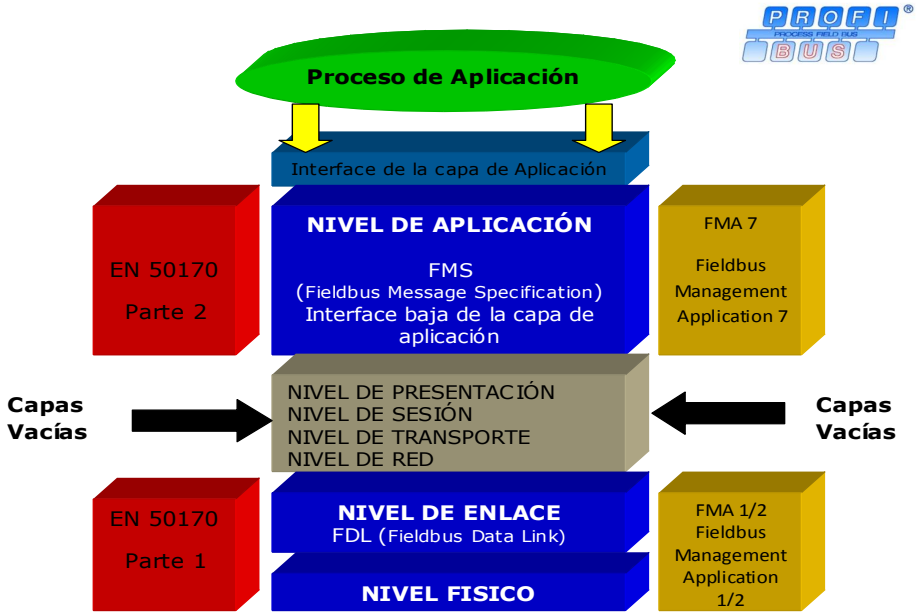


Figura 1.2: Descripción capas del Protocolo Profibus.
Fuente: Tomado de (Profibus Specification, 1998).

de línea promedio de 10mA y cambios de 1mA a 19mA según el dato transmitido. Adicionalmente la codificación Profibus PA es tipo Manchester (*Profibus Specification Normative Parts of PROFIBUS-FMS, -DP,-PA accordin to the European Standard EN 50 170*, 1998).

Las velocidades en Profibus-DP son configurables desde 9600kbps hasta 12Mbps, que según sea la velocidad de comunicación será su alcance variando desde 1200 metros a bajas velocidades hasta 100 metros a altas velocidades como se observa en la Tabla 1.2. La velocidad de comunicación típica es de 1,5Mbps, sin embargo para las aplicaciones utilizadas se llevó a la velocidad mínima. El tiempo máximo para que llegue una información desde un dispositivo de campo a un PLC, no debe ser mayor a los 10ms (Guerrero y cols., 2012).

Vel Tx (Kbps)	9.6	19.2	93.75	187.5	500	1500	2000
Distancia	1200m	1200m	1200m	1000m	400m	200m	100m

Tabla 1.2: Velocidades de Transmisión con sus respectivas distancias.
Fuente: Tomado de (Profibus Specification, 1998).

En Profibus, el método utilizado para el control de la red es mediante el modo maestro-esclavo, permitiendo conectar hasta 126 dispositivos como esclavos dentro de la red. Los maestros son dispositivos activos inteligentes que controlan la red e interrogan a los esclavos que son dispositivos pasivos. Los esclavos pueden ser

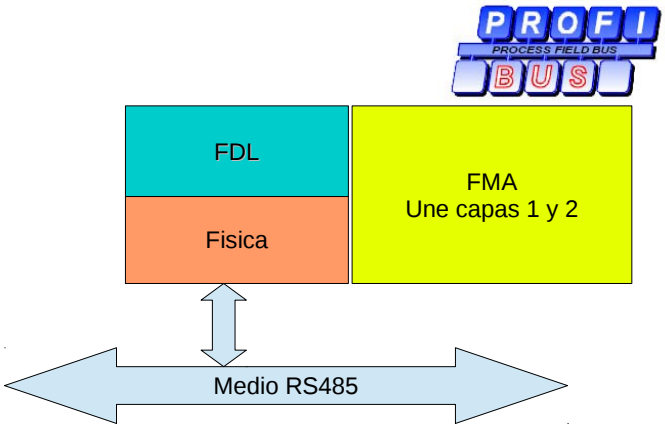


Figura 1.3: Descripción capa física y de enlace del Protocolo Profibus.
Fuente: Tomado de (Profibus Specification, 1998).

dispositivos inteligentes pero nunca podrán iniciar la comunicación.

1.2.2. Capa de Enlace

La capa de enlace en Profibus-DP se basa en el paso de tramas con telegramas mediante el paso de testigo (*token passing*), donde la estación maestra que tiene el testigo, es la que realiza la transmisión. Si la línea de transmisión esta libre su estado es "1" lógico. El testigo se pasa en orden ascendente de direcciones Profibus en comunicación convencional, pero si se va a cerrar el ciclo, el testigo se pasa de la estación con la dirección más alta Profibus a la estación con la dirección más baja. En el proceso de inicialización, cada estación maestra reconoce su predecesora y sucesora. Las estaciones esclavas cuando realizan el proceso de transmisión utilizan el método de sondeo o *polling* (cíclico) después de alguna solicitud de una estación maestra o cuando la línea esté libre.

La transmisión maneja 4 tipos de tramas organizadas por octetos, donde cada octeto está dado por el estándar USART. Cada trama puede llevar de 3 a 246 bytes de información, sin tener en cuenta el encabezado (SD *Start Delimiter*), chequeo de secuencia (FCS *Frame Check Sequence*) y fin de trama (ED *End Delimiter*). El octeto de inicio de trama define el tipo de trama que puede ser:

SD1=10h Trama de longitud fija sin datos

SD2=68h Trama de longitud variable con datos. Hasta 246 Bytes de datos que con las direcciones, longitudes, chequeos, fin de trama entre otros da un total de 255 Bytes por trama.

SD3=A2h Trama de longitud fija con datos

SD4=DCh Trama de Paso de Testigo. Es transmitida constantemente por el maestro, y no se modifica sí solo existe un maestro en la red. Es de 3 Octetos

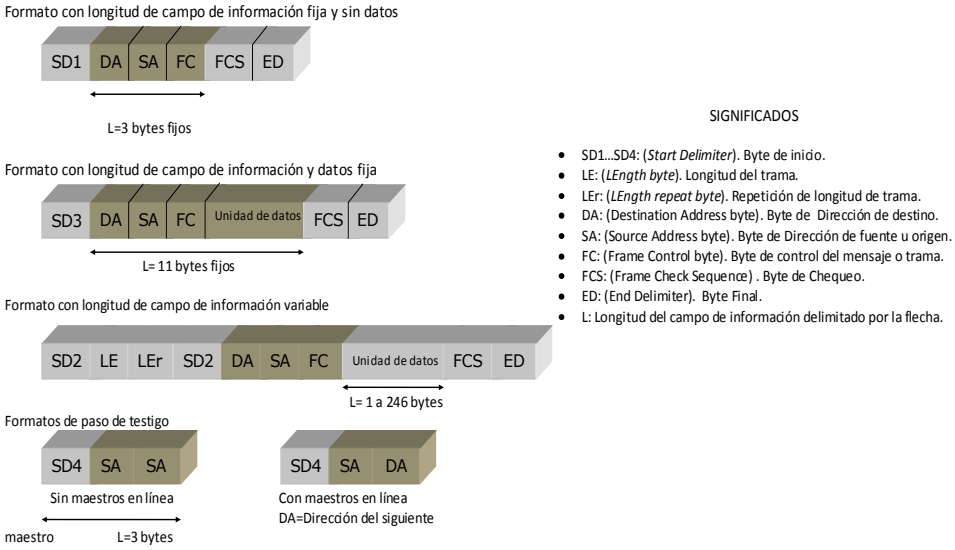


Figura 1.4: Tramas Protocolo Profibus-DP.
Fuente: Tomado de (Profibus Specification, 1998).

iniciando con SD y manda la dirección maestro 2 veces seguidas SD, SA, SA (*Source Address*) buscando estaciones sucesoras y si no hay respuesta asumirá esta que es la única. Una vez encuentra otra estación maestra queda convertida en SD, SA, DA (*Destination Address*) lo que se modifica en la lista de estaciones activas (LAS) que luego son llevadas al GAPL (Lista de Grupo, Algoritmo en Programación). Cada una de las otras tramas es precedida por la trama de testigo.

La comunicación Profibus-DP utiliza diferentes tipos de servicios de transferencia de datos acíclicos, los cuales pueden ser SDN (*Send Data with No Acknowledge*) que son mensajes de difusión del maestro a todos los esclavos, SDA (*Send Data with Acknowledge*) mensajes punto a punto para mandar datos o mensajes de control e información del maestro a esclavo, RDR (*Request Data with Reply*) mensaje punto a punto para solicitar datos de los esclavos y SRD (*Send a Request Data*) mensajes punto a punto para enviar y recibir datos desde el esclavo. Estos servicios están definidos por el byte FC (*Function Code*) de la trama Profibus, donde ante tramas de acción, existirán tramas de reconocimiento o respuesta. Los tipos de tramas y bytes se pueden visualizar en la Figura 1.4.

Como se puede ver en la Figura 1.4 los 3 primeros tipos de tramas tienen en común los bytes SD, el byte de dirección de destino DA, el de dirección de fuente SA y el código de la función de la trama FC, además del Chequeo de Secuencia de Trama FCS y del delimitador final ED. Sin embargo la trama 4 es diferente al igual que una trama de respuesta o identificación de datos recibidos correctamente, la cual es de un solo byte (E5h), el cual se conoce como SC (*Sequence Correct*), indicando que es reconocida la trama o trama recibida correctamente por parte del esclavo. En las tramas adicionalmente se chequea el sincronismo y hay veri-

ficación de desplazamiento o deslizamiento mediante *Hamming Distance* (HD4) (*Profibus Specification Normative Parts of PROFIBUS-FMS, -DP,-PA accordin to the European Standard EN 50 170*, 1998).

Existen 2 tipos de mensajes cíclicos que son el CRDR (*Cyclic Request Data with Reply*) y CSRD (*Cyclic Send and Request Data*) que se utilizan como telegramas especiales de difusión que contienen la petición a todos los esclavos, permitiendo monitorear la red y verificar cual dispositivo responde cuando es direccionado.

Estados FDL

Una estación maestra tiene 11 estados FDL y sus transiciones, mientras que la esclava maneja 2 estados que son el 0 y el estado 10. Los estados en su respectivo orden empezando de 0 a 10 se presentan en la Tabla 1.3.

0	Offline
1	Listen_Token
2	Active_Idle
3	Claim_Token
4	Use_Token
5	Await_Data_Response
6	Check_Access_Time
7	Pass_Token
8	Check_Token_Pass
9	Await_Status_Response
10	Passive_Idle

Tabla 1.3: Estados de FDL.
Fuente: Elaboracion propia en base a (Profibus Specification, 1998).

El estado **Offline** se dá en errores o reset de la comunicación Profibus, y se mantiene hasta que todos los parámetros han sido inicializados, aquí el PLC no trasmite y después de esto, se pasa a los estados **Passive_Idle** o **Listen_Token** Figura 1.5. En **Passive_Idle** los dispositivos escuchan el estado de la línea, donde el esclavo se queda esperando una trama direccionada o de requerimiento, sin tomar en cuenta el *token* o *broadcast*. Cuando estan en **Listen_Token** el maestro está listo, identifica los otros PLCs maestros, para lo cual se analizan los token y con estos se generan listas de estaciones activas para su direccionamiento o paso de testigo. Luego de 2 rotaciones se mantiene en este estado hasta que es direccionable por un requerimiento de estado FDL (Request_FDL_status) transmitido por el predecesor. Al estado **Listen_Token** también se puede entrar cuando hay error o 2 tramas de token son recibidas sucesivamente con la misma dirección.

La estación que responda "ready to entry token ring", el siguiente token es direccionado a esta estación entrando en el modo **Active_Idle**. El **Active_Idle**

permite escuchar el bus y esperar una trama token hacia él y una vez la recibe pasa al estado **Use_Token** inmediatamente. Si el FDL no observa actividad en el bus, este debe asumir que es necesario restablecer el token por lo que pasa al estado **Claim_Token** para luego de allí ir al **Use_Token**. A el estado **Claim_Token** entra después de haber estado en el **Active_Idle** o **Listen_Token** una vez el tiempo se haya agotado Figura 1.5.

Después de que se transmite la lista de estados se crea el **Await_Status_Response** estado por medio del Request_FDL_status, solicitando la siguiente estación direccionable. Con el estado **Use_Token** se pueden enviar los mensajes de alta o baja prioridad y una vez se envía pasa al estado **Await_Data_Response** pero si hay disponibilidad de tiempo regresará al **Use_Token**.

Se puede entrar al estado **Check_Access_Time** si después del **Use_Token** no hay mensajes de alta prioridad o después de procesar mensajes generales. El mensaje o estado **Await_Data_Response** entra después de enviar una trama de acción, esperando un reconocimiento o trama de respuesta. El estado de **Check_Access_Time** chequea el tiempo remanente para transmisión y si hay tiempo disponible pasa al **Use_token**, sino al **Pass_Token**. En este último estado, el FDL pasa el token a la siguiente estación y chequea si esta está trabajando bien pasando al **Listen_Token** y si este no se recibe es un error fatal y FDL para el anillo y entra a **Offline**, pero si se recibe incorrectamente, se entra al estado **Check_Token_Pass** Figura 1.5.

Si se acaba el tiempo para el GAP(Grupos, Algoritmos y Programación) encargado de construir la lista de dispositivos en la red, pero hay tiempo remanente, se trata de grabar una nueva estación en el GAP en la sección de Grupos de dispositivos en la red e incluirla en el token ring. Para esto se entra al modo Request_FDL_Status y entra a **Await_Status_Response** y si el PLC maestro reconoce que quiere ser incluido, el FDL pasa el token a esta estación y el GAP graba esto y si no quiere entrar, esta queda almacenada en la lista de Grupo GAPL, si esta no responde se elimina del GAPL. Si en el mantenimiento del GAP ninguna estación responde, el FDL pasa el token al original y entra al estado **Check_token_pass** y si solo es la estación activa en el bus pasa a **Use_token**.

Check_Token_Pass es el estado en el cual el FDL espera un tiempo para pasar el token y si detecta una trama valida, supone que el token se pasó correctamente. La trama debería ser procesada, como si fuese recibida en estado **Active_Idle**, o si se recibe una trama invalida, es porque otra estación esta activa, entrando al estado **Active_Idle**. Si el FDL no recibe una trama en el tiempo estimado, este reenvía el **Pass_Token**. Los dispositivos entran al estado **Await_Status_Response** desde el **Pass_Token** después de que un sucesor es desconocido, para lo cual el FDL debe esperar un tiempo para el reconocimiento de trama. Si ninguna trama es recibida o se recibe una trama corrupta, se debe retransmitir el **Pass_Token** a él mismo o al sucesor. Si FDL recibe alguna otra trama contraria a la de reconocimiento, esto indica que múltiples token existen, por lo que se debe pasar al estado **Active_Idle**. Después de que el ciclo de men-

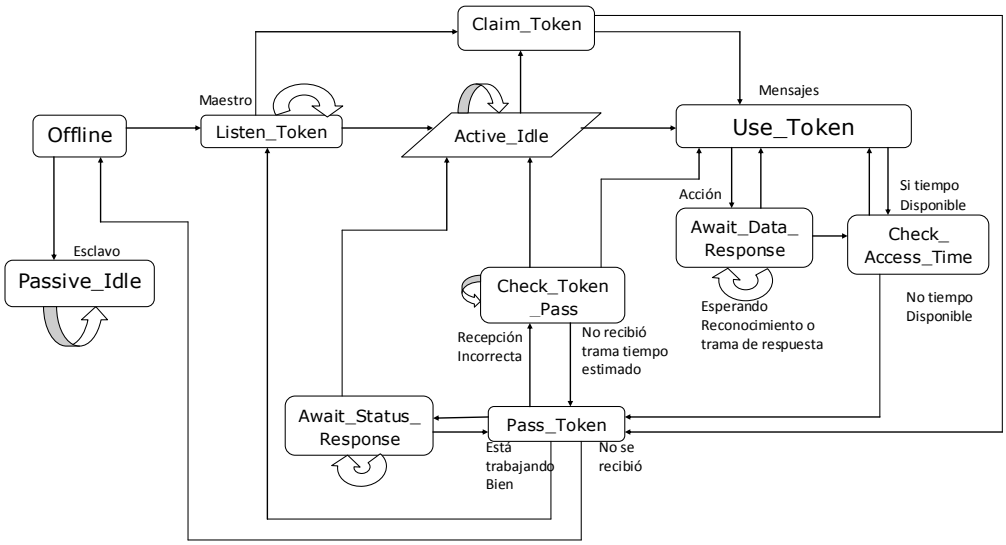


Figura 1.5: Funcionamiento paso de estados FDL.
Tomado de (Profibus Specification, 1998).

sajes a finalizado se hace un chequeo del GAP. Todo el proceso realizado en la capa de enlace con estos estados se ve con más claridad en la Figura 1.5.

Distribución de Bits y estados en la Trama

Los datos que se envían en la trama tienen una distribución bit a bit, que establecen aspectos de direccionamiento, funciones, tipo de servicios, direcciones de extensión entre otras. En el área de datos luego de definir el tipo de trama, y aparte de los bytes de longitud de la trama para el tipo SD2, siguen los bytes DA y SA. Si el byte DA tiene todos los bits en "1" (FFh) indica que la trama es una trama de broadcast o multicast como servicios de punto de acceso o SDN. Si los bytes DA y SA tienen el bit 7 en "1", es porque en la carga útil de información después del byte FC, hay direcciones de extensión DAE (*Destination Address Extension*) y SAE(*Source Address Extension*) que pueden ser de datos o de segmentos, las cuales se utilizan en servicios SDA y SDN, pero si el bit7=0 no hay direcciones de extensión. En las direcciones DAE y SAE en el bit 6 define el tipo, donde si es "0" describen la dirección de región o segmento, pero si es "1" se describe el LSAP(*Link Service Acces Point*) complementándolo con el bit 7 que describe la dirección de extensión adicional.

El byte FC es uno de los más importantes dentro de la trama Profibus ya que define aspectos como: secuencia de trama, tipo de servicio, prioridad del mensaje. En este byte se define si la trama es de acción, solicitud, envío con solicitud de respuesta o reconocimiento. El Bit 8 es un bit reservado que el receptor no lo

interpreta, el bit 7 indica si la trama es de reconocimiento y respuesta o de envío y requerimiento. El bit 6 llamado FCB (*Frame Count Bit*), previene perdida o multiplicación de mensajes. El FCB cambia (*toogle*) por cada trama de acción y respuesta, confirmando al remoto que el mensaje previo fue recibido correctamente. Si el FCB=0 es porque se envió un mensaje SDN. Acompañado del FCB está el FCV (*Frame Count Verification*) (bit 5) el cual es también "0" en mensajes SDN o igualmente cuando una estación queda marcada como no operacional con el FCB=1 el FCV=0. El FCV es igual a "1" cuando se tiene una respuesta a la trama de acción. El receptor evalúa el FCB cuando el FCV=1 siempre y cuando la dirección (SA) sea del que se le hizo la solicitud. Los bits 4 al 1 manejan una codificación indicando el tipo de trama, servicio y prioridad. Esta se da según la Tabla 1.4 que indica los valores del 0 (0000) al 15 (1111).

Seguidamente se encuentran los bytes de datos o unidad de datos (DU), que es donde están las direcciones de extensión que pueden ser hasta 4 y los datos para las capas FDL y FMA. Como las direcciones de extensión solo se utilizan en el proceso de negociación y establecimiento de la comunicación, es posible cambiar parámetros del PLC a través de estos bytes, e inclusive detectar las fallas del mismo con la unidad de datos (DU)

Luego de la unidad de datos está el byte FCS el cual se forma mediante la suma de los bytes DA, SA, FC y los bytes de datos tomando los primeros 8 bits de esta suma, empezando por los de menor peso. Para esto al igual que todos los bits se utiliza el chequeo HD4 (Hamming Distancia 4) que verifica el desplazamiento de los bits con sincronismo.

Finalmente se encuentra el Byte ED cuyo valor es de 16h el cual representa el fin de la trama Profibus. Tomando una trama Profibus y vista en el osciloscopio se tiene la imagen de la Figura 1.6.

Proceso de Negociación

El proceso de negociación de las tramas Profibus-DP inicia con una trama SD4 verificando si hay más maestros en la comunicación. Si los hay estos deben responder con una trama SD3. Seguidamente envía 2 tramas de *broadcast* SD2 (con DA =FFh) de tipo SDN y una vez las envía, continua mandando tramas SD2 de tipo SRD direccionadas al esclavo Profibus configurado, esperando respuesta del respectivo dispositivo. El esclavo al detectar la trama direccionada hacia él, contesta con una trama SD2 de tipo RDR con respuesta OK, informándole al maestro que lo ha reconocido, luego el maestro envía otras 2 tramas SD2 SRD contestando el esclavo con E5h (OK). Finalmente el maestro envía una última de verificación SD2 tipo SRD y el esclavo contesta RDR, ver Figura 1.7.

Una vez termina el proceso de negociación, se envían tramas continuamente entre el maestro y el esclavo, el maestro envía tramas SD1 o SD2 tipo SRD alternándole el FCB y el esclavo siempre contesta con la trama SD2 de tipo RDR. En

CÓDIGO	FUNCIÓN CON bit 7=1
0,1,2	Reservado
3	Envío de datos con reconocimiento. Baja Prioridad
4	Envío de datos sin reconocimiento. Baja Prioridad
5	Envío de datos con reconocimiento. Alta Prioridad
6	Envío de datos sin reconocimiento. Alta Prioridad
7	Reservado. Requiere diagnóstico de datos
8	Reservado
9	Requerimiento de estado de FDL con respuesta
10,11	Reservado
12	Envío y respuesta de datos con baja prioridad
13	Envío y respuesta de datos con alta prioridad
14	Solicitud identificación con respuesta
15	Solicitud estado LSAP con respuesta. (Códigos 14 y 15 para FMA 1/2
CÓDIGO	**FUNCIÓN CON bit 7=0**
0	Reconocimiento positivo (OK)
1	Reconocimiento negativo. Error usuario FDL / FMA 1/2
2	Reconocimiento negativo. Ningún recurso para enviar datos. No respuesta del FDL data
3	Reconocimiento negativo. No servicio activado
4,5,6,7	Reservado
8	Respuesta FDL/FMA 1/2. Datos de baja prioridad y envío de datos ok
9	Reconocimiento negativo. No respuesta de datos del FDL/FMA y envío de datos ok
10	Respuesta del FDL. Alta prioridad de datos
11	Reservado
12	Respuesta del FDL. Baja prioridad de datos. Ningún recurso para enviar datos (RDL)
13	Respuesta datos FDL. Alta prioridad. Ningún recurso para enviar datos (RDH)
14,15	Reservado

Tabla 1.4: Códigos de función bits 4 al 1 en decimal.
Fuente: Tomado de (Profibus Specification, 1998).

estas tramas en el campo de datos (DU) se encuentran los valores de las variables de datos específicas a leer y escribir desde y hacia los maestros o esclavos.

Entre estas tramas existe siempre una trama inicial SD4 que verifica la presencia de maestros, luego van las SD1 o SD2 del maestro seguida de la respuesta SD2 o E5 del esclavo y finaliza con una trama de conteo dada por el maestro que es del tipo SD1 con servicio SDN.

Las tramas de respuesta E5h del esclavo, también pueden ser SD1 con servicio RDR o del tipo SD2 o SD3. Y los servicios SDA por lo general se mandan con tramas SD3 (*Profibus Specification Normative Parts of PROFIBUS-FMS, -DP,-PA accordin to the European Standard EN 50 170*, 1998). Un proceso típico de

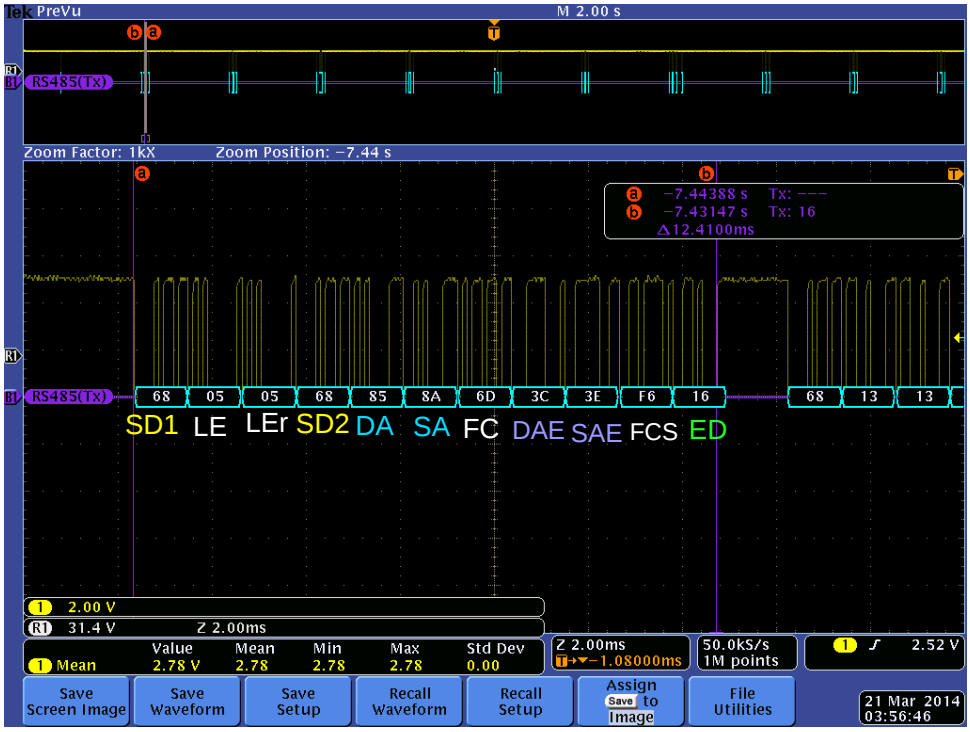


Figura 1.6: Trama Profibus-DP tomada desde osciloscopio con identificación.
Fuente: Elaboración propia

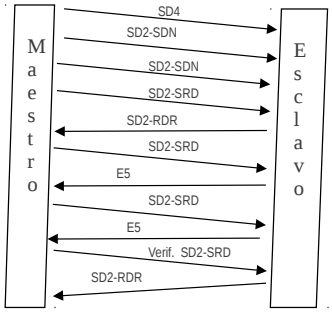


Figura 1.7: Tramas dadas en el Proceso de Negociación.
Fuente: Elaboración propia

negociación con las tramas de bytes dadas podría ser como el que se observa en la Tabla 1.5.

El sistema Profibus-DP tiene la capacidad de detectar errores: como tramas, paridad, desborde de velocidad o errores en el protocolo. Si una trama con servicio

Trama Maestro	Tipo	**Trama Esclavo**	Tipo
68, 05, 05, 68, 85, 8A, 6D, 3C, 3E, F6, 16	SD2 SRD	68, 0B, 0B, 68, 8A, 85, 08, 3E, 3C, 02, 05, 00, FF, 80, D1, E8, 16	SD2 RDR
68, 0F, 0F, 68, 85, 8A, 5D, 3D, 3E, B8, 41, 42, 36, 80, D1, 00, C0, 60, 00, C9, 16	SD2 SRD	E5	RX OK
68, 28, 28, 68, 85, 8A, 7D, 3E, 3E, 04, 00, 00, AD, C4, 04, 00, 00, 8B, 41, 04, 00, 00, 8F, C0, 83, 00, 00, 93, 40, 43, 00, 00, 83, 40, 83, 00, 00, 93, 40, 43, 00, 00, 83, 40, 58, 16	SD2 SRD	E5	RX OK
68, 05, 05, 68, 85, 8A, 5D, 3C, 3E, E6, 16	SD2 SRD	68, 13, 13, 68, 8A, 85, 08, 3E, 3C, 02, 05, 00, FF, 80, D1, 42, 00, 06, 82, 00, 00, 00, 00, B2, 16	SD2 RDR
68, 05, 05, 68, 05, 0A, 7D, 00, 00, 8C, 16	SD2 SRD	68, 05, 05, 68, 0A, 05, 08, 00, 00, 17, 16	SD2 RDR

Tabla 1.5: Proceso de negociación con tramas maestro-esclavo.
Fuente: Elaboración propia según análisis de comunicación

SDA, SRD y RDR no se procesa, esta se reenvía un tiempo después hasta recibir una respuesta válida o confirmación. Si el tiempo de espera por parte del maestro es muy alto, este se reiniciará y deberá volver a empezar el proceso de negociación.

Tiempos de Operación

EL PLC en su comunicación Profibus maneja un conjunto de tiempos que tienen que ver con el token, retardos, tiempos libres, tiempos de trama, tiempos cíclicos, de sincronización, transmisión, tiempos de reacción del sistema entre otros. Algunos de estos tiempos vienen preestablecidos y no son modificables pues dependen del programa o código que se tenga en el PLC, sin embargo hay otros que son modificables.

Por ejemplo, el tiempo de ciclo del token está determinado por el número de estaciones maestras, el tiempo del token (T_{tf}), el retardo de la transmisión (T_{td}) y el tiempo libre (T_{id}). Este no dependerá de los mensajes cíclicos regulares. Para el tiempo de trama del token se toman el número de caracteres, el retardo de la transmisión que depende de la longitud de la línea y el **idle_time** (tiempo de margen inactivo) que depende del retardo en la recepción del token; adicionalmente se debe tener en cuenta el tiempo de sincronización + T_{sm} (tiempo de margen de seguridad).

El tiempo de los ciclos de mensajes (T_{mc}), depende de los tiempos de la trama de acción y la trama de respuesta. Incluyendo transmisión, retardo de transmisión y retardo de respuesta de la estación (T_{sdr}). Los tiempos de reacción del sistema (T_{sr}) equivale al número de mensajes por segundo o el inverso del tiempo de ciclos de mensajes y depende del número de esclavos en el modo de "sondeo puro" (un

maestro), si son varios maestros entonces el $T_{sr} = T_{tr}$ (rotación de tarjeta). Estos tiempos con su dependencias se ven con mayor claridad en la Tabla 1.6.

Tiempos	Depende de
Tiempo de ciclo de token	#Estaciones maestras
	Ttf
	Ttd
	Tid
Tiempo de trama de token	# de Caracteres
	Retardo de la transmisión
Tiempo de inactividad	Retardo en la recepción del token
	Tiempo de sincronización
	Tsm
Tiempo de ciclos de mensajes Tmc	Tiempos de trama de acción
	Trama de respuesta
Tiempos de Reacción del sistema Tsr	# mensajes por segundo
	o inverso del tiempo de ciclos de mensajes
	# de esclavos
	O rotación Ttr con varios maestros

Tabla 1.6: Tiempos con sus dependencias.
Fuente: Elaboración propia basado en (*SIMATIC Programming with STEP 7 V3.0*, 2004).

Al configurar la velocidad de transmisión de las tramas Profibus del PLC algunos tiempos son modificables pero otros no, debido a que unos dependen de otros según se establezcan en la trama, por lo que al modificar algún tiempo, se deben recalcular los tiempos máximos de supervisión y respuesta dependientes los cuales no son modificables. Por ejemplo, el tiempo de iniciación de la trama (T_{slot_init}) trae un valor por defecto pero es modificable, al igual que Max_T_{sdr}, Min_T_{sdr}, T_{set} (tiempo de configuración entre 2 eventos de solicitud y respuesta), T_{qui} (tiempo de espera después de transmitir y de habilitar la recepción) los cuales están relacionados directamente con la velocidad de transmisión escogida. Al cambiar los tiempos modificables hay otros que se cambian automáticamente por defecto en el entorno de programación Step-7 de Siemens como se ve en la Figura 1.8

Otros Parámetros Profibus

Cada sistema PROFIBUS-DP puede contener 3 tipos diferentes de dispositivos:

i DP Maestro Clase 1 (DPM1): Controlador central que intercambia información con las estaciones descentralizadas (DP esclavos) con un ciclo de mensaje específico. Dispositivos típicos son los controladores programables (PLCs), los PCs y los sistemas VME (Versa bus Module Europe) el cual es un bus estándar para comunicar hardware de computadores.

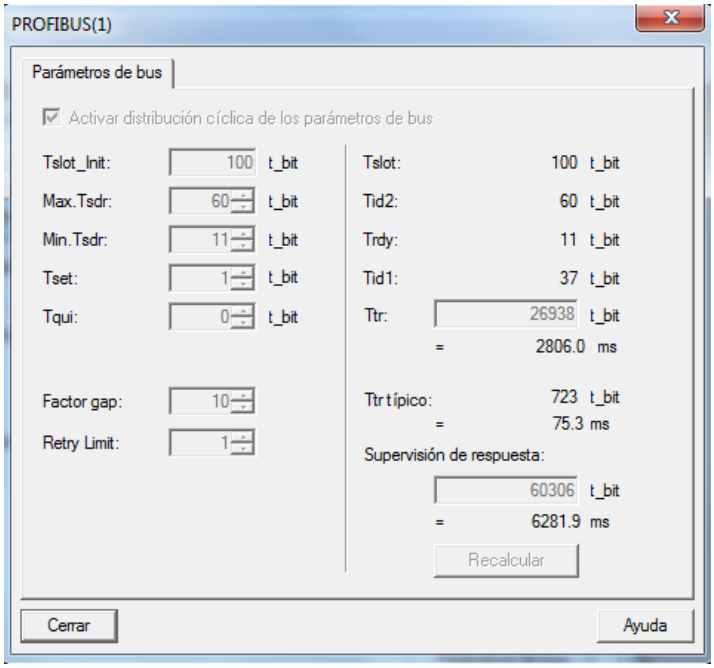


Figura 1.8: Tiempos estándar en una comunicación Profibus a 9600bps en un PLC Siemens.

Fuente:Elaboración propia tomada de Simatic Step 7

ii DP Maestro Clase 2 (DPM2): Son programadores, dispositivos de configuración y operadores. Se usan para la identificación de la configuración del sistema DP o para el funcionamiento y supervisión de operaciones.

iii DP esclavo: Es un dispositivo periférico (entradas/salidas, válvulas, etc.) que recoge información de entrada y/o manda información de salida. Este solo puede mandar datos de reconocimiento.

El DPM1 envía su estado a los DP esclavos mediante un Multicast. En el caso de que se presente un error durante la transferencia de datos, se dará un parámetro de "autoborrado", el cual si pasa a verdadero, el DPM1 pone las salidas de los DP esclavos en espera (seguridad) hasta que se establezca de nuevo las condiciones para una transmisión correcta.

La transmisión de datos de usuario entre DPM1 y DP esclavos se divide en 4 fases

1. Asignación de parámetros o Parametrización (Dada por Software)
2. Configuración
3. Diagnostico (se chequea para luego transmitir) y
4. Transferencia de datos en los cuales puede estar Comandos de Control y Datos del Usuario.

Cada DP esclavo compara su configuración real, con la esperada (la que manda el DPM1). Cuando las dos configuraciones anteriores coinciden, el DP esclavo será incluido en la fase de transferencia. Con esto se garantiza protección contra los errores de parametrización.

Con las funciones de comunicación entre maestros, es posible cambiar parámetros de configuración sobre el bus, habilitar o deshabilitar transferencia de datos y cambiar la operación del DPM1.

Existen intervalos de monitorización que se especifican en la configuración, proporcionando protección. Además solo se pueden modificar los parámetros de los esclavos asignados al maestro respectivo. Los otros maestros solo pueden visualizar las entradas y salidas de los esclavos, pero no las pueden modificar.

La prioridad en una transmisión se puede colocar baja, si los datos de un dispositivo cambian muy poco en relación a la transmisión cíclica, como se da en convertidores de frecuencia u otros esclavos.

LA PRIORIDAD ALTA: son los mensajes que primero procesa un PLC maestro cuando recibe el testigo y luego de esto procesa los mensajes de baja prioridad. En un ciclo un maestro debe ejecutar al menos un ciclo de ALTA prioridad, a modo de ejemplo están: las alarmas, los mensajes temporales críticos, los datos de sincronización y la coordinación. Estos mensajes por norma deben de tener menos de 20 octetos en la unidad de datos.

Los de BAJA PRIORIDAD son datos no urgentes, como: el proceso, los diagnósticos y los datos del programa. Solo pueden ser procesados si T_{rr} (*Time Real Rotation*) $<T_{tr}$ (*Time Token Rotation*)

El sistema PROFIBUS-DP tiene la capacidad de detectar errores, entre los cuales están: errores de tramas, paridad, desborde de velocidad o errores de protocolos de transmisión como fallas en el inicio de trama, en los octetos y finales de trama, al igual que en longitudes y tiempos de respuesta.

Una trama incorrecta no se procesa, por lo que se reenvía después de que ha terminado el tiempo de esta. El iniciador completa la solicitud, después de haber recibido una respuesta válida y si no se da esta, se debe mantener hasta el otro ciclo cuando se haya dado la respuesta de confirmación. Una vez se reciben de manera correcta las respuestas de reconocimiento se utiliza una respuesta de trama con contador de bit. En caso que no se reconozca, esta debe ser marcada como "no operacional ", pero si hay respuesta se marca como "operacional".

Los mensajes transmitidos manejan una secuencia de 4 tipos de modos:

1. *Token handling.*
2. *Acyclic request or send/request operation.*
3. *Cyclic send/request operation, polling.*
4. Registro de estaciones.

Los comandos de control permiten la sincronización de entradas y salidas mediante Modo *Sync*: Sincroniza Salidas y Modo *Freeze*: Sincroniza Entradas. Los comandos de control que manda el maestro a los esclavos quieren decir lo siguiente:

SYNC: Se lee el estado de las salidas y se inmovilizan.

UNSYNC: Se cancela el comando sync para las salidas.

FREEZE: Se lee el estado de las entradas y se inmoviliza.

UNFREEZE: Se cancela el comando Freeze para las entradas.

CLEAR: Se reinician todas las salidas.

Se pueden tener funciones de diagnóstico con 3 niveles jerárquicos de mensajes. Entre estos están los de las estaciones, los cuales indican el estado general de los dispositivos. De igual manera están los de los módulos los cuales se refieren a entradas o salidas. Finalmente el de los canales que refiere a errores en bits individuales de entradas y salidas.

Entre las conexiones acíclicas esta el reconocimiento de alarmas, lecturas de partes del esclavo, entre otras

1.2.3. Capa de Aplicación

La capa FMA tiene 2 modos de operación que son: "Interfaz de usuario" y el modo "Administrador". A pesar de que los servicios son los mismos, el administrador tienen unos parámetros adicionales de control donde puede mandar primitivas sin parámetros definidas por el estándar *Profibus Specification*.

FMA 1/2 "Interfaz de Usuario" describe los servicios de administración de las capas 1 y 2 dadas en el FMA. Los servicios de esta interfaz permiten reinicializar las capas, modificar parámetros de estas, notificar errores, solicitud de identificación, configuración del Enlace del Servicio del Punto de Acceso LSAP local o remoto, estado del FDL, visualizar lista de estaciones funcionales y activación y desactivación del LSAP. En este enlace LSAP se excluye la respuesta de los servicios SRD o CSRD lo cual se realiza con el RSAP (*Reply Service Access Point*). Este servicio posee campos opcionales y campos obligatorios, según se refiera a capas locales o remotas.

Además FMA 1/2 "Manager" funciona de igual manera entre las capas 1 y 2. Posee uno de enlace con la capa física y otro con la capa FDL, los cuales manejan las mismas primitivas que la "Interfaz de Usuario" FMA. El FMA 1/2 -FDL posee los siguientes servicios: reset, set_value, read_value y Fault_FDL. Como no tiene entre las primitivas los valores MSAP (*Manager Service Access Point*) y M_status (estado de la estación maestra) algunas de estas van sin parámetros como el *reset* y el set_value, sin embargo hay otras que tienen primitivas con nombres y parámetros.

Existe el servicio de detección de fallas, el cual permite gestionar ciertos errores indicados por la capa FDL. Los errores que puede trabajar son: dupplicate_address, Fault_transceiver (funciona mal el tx o rx), out_of_ring (sacada del anillo sin pedirlo), GAP_event (cambio del GAP), Time_out (no actividad en el bus), Not_syn (no sincronización en el intervalo). Solo los 2 últimos pueden ser utilizados por las estaciones esclavas o pasivas. En la capa FMA 1/2 "Manager"

también está el enlace FMA 1/2 con la capa física. Mientras que en FMA 1/2 "Usuario" no existe este enlace. En el "Manager", están los servicios de *reset, set value, read value y event.*

1.3. Comunicación Zig Bee

Una de las tecnologías que más se está utilizando actualmente para formar las redes inalámbricas de corto alcance son las dadas por los sistemas ZigBee. La cual está definida en su capa física y de enlace por la Norma IEEE 802.15.4 y respaldada por *ZigBee Alliance*. En esta tecnología también se han definido las capas de aplicación y red con el objetivo de dar mayores usos en diferentes aspectos y perfiles. ZigBee se caracteriza por tener bajo consumo de potencia (menor de 10uA), alta seguridad, con alcances de 30 a 300 metros en aplicaciones básicas con obstáculos y velocidades de transmisión que son del orden de 250kbps, acordes a lo dado en los procesos industriales.

La tecnología ZigBee puede ser implementada en un amplio rango de productos y aplicaciones a nivel comercial, industrial, gobierno, hogar y otros tipos de mercados alrededor del mundo (*ZigBee Alliance*, s.f.). Tales como las implementaciones de control unidireccional entre un PLC y un computador con visualización de variables en Labview vía ZigBee se han llevado a cabo (Pengfei y Jiakun, 2009) con protocolo RS485, al igual que comunicaciones bidireccionales entre PLCs y microcontroladores para control y monitoreo (Xu y cols., 2012).

Como se establece en (Ocampo Daniel, 2010) la especificación de un perfil ZigBee para monitoreo y control de plantas industriales, implica direccionar la plataforma de comunicación a un sistema especifico, desarrollando aplicaciones distribuidas que se puedan desplegar fácilmente del dominio de interés. De esta manera, si en un dispositivo industrial como un PLC, se puede recibir y enviar comandos de un sistema, asignando tareas de control, transferencia de datos y determinar estados críticos dentro de una red con un aplicativo mediante una conexión ZigBee, se está definiendo un perfil de comunicación ZigBee con los PLCs. En otras palabras, un perfil especifica la funcionalidad del dispositivo, que mediante varias descripciones forman una aplicación. Los perfiles pueden ser públicos o privados y definen estándares de comunicaciones, aplicaciones con controladores, entre otros aspectos.

Este estándar define los tres tipos de topologías de red como: bus, estrella y malla, adicionando la red *Ad-oc*. Estas topologías están compuestas por coordinadores, RFD(*Reduced Funtion Device*), enrutadores FFD(*Full Function Device*) y Dispositivos Finales (ED *End Devices*) Figura 1.9, que mediante un perfil definido se logran técnicas de conexión directas con otros dispositivos o estableciendo redes de sensores ZigBee se pueden tratar señales en los hogares (Y. Wu y Shao, 2012).

Con cada módulo ZigBee además de configurarlo como Coordinador, Enrutador (*Router*) o Dispositivo Final, se pueden realizar transmisiones de diferentes funciones mediante comandos API(*Application Programming Interface*) o enviar byte

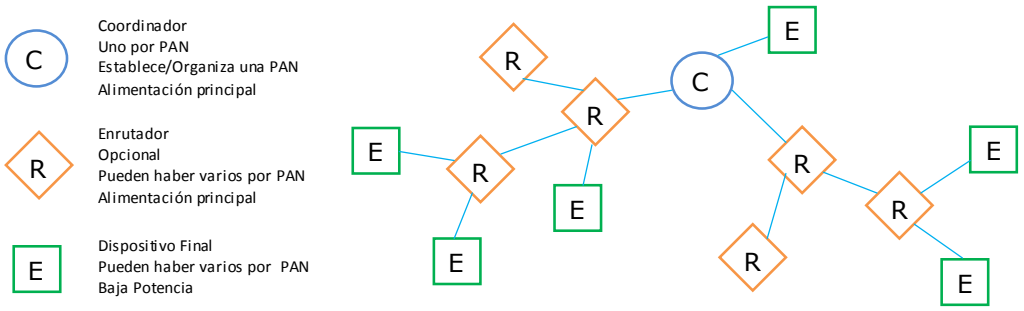


Figura 1.9: Estructura Conformación de una Red ZigBee.
Fuente: Tomado de Manual XB_ZigBee_OEM Digi International, 2010.

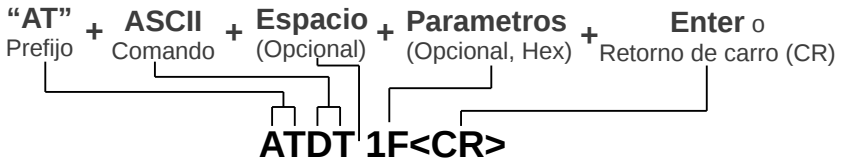


Figura 1.10: Estructura envío comandos AT.
Fuente:Elaboración propia basado en Manual XB_ZigBee_OEM Digi International, 2010.

por byte en el modo *Bypass* o *AT/Transparent* (modo transparente con control por comando AT). Mediante la interfaz API se pueden realizar mayores funciones e inclusive programar remotamente el módulo ZigBee, además de permitir envío a múltiples destinos simultáneamente, identificar direcciones y fallas, entrar remotamente al modo de comandos AT entre otras funciones. En el modo de comandos AT(*Attention command*) el dispositivo trabajará como un simple repetidor, logrando un menor consumo de recursos y envió de información sin procesamiento complejo cuya sintaxis se puede ver en la Figura 1.10. ZigBee además posee unos *buffer* internos para transmisión y recepción que se pueden configurar para evitar desborde con las líneas CTS (*Clear to Send*) y RTS *(Ready to Send*) de una comunicación serial típica completa.

Los coordinadores son los encargados de seleccionar el canal, el perfil, que dispositivos se pueden acoplar y es el único que puede empezar una red y enrutar paquetes. Puede identificarse en la red PAN(*Personal Area Network*) con el conocido PAN-ID de la red. Con un PAN seleccionado puede enviar una señal de *broadcast* y los que contesten quedarán en el PAN *scan* o *active scan*. Luego mantendrá este PAN ID incluso en el ciclo de *reset* y hasta que se salga de la red. Además si hay varios coordinadores en la red se pueden seleccionar estados de inicio por cada uno. Los coordinadores se pueden controlar por un conjunto de comandos con los cuales permiten cambiar parámetros en tiempo real con recono-

Comando	Descripción
ID	Usado para determinar el PAN ID de 64 bits. Si es puesto en 0 (por defecto) un aleatorio PAN ID es seleccionado
SC	Determina la máscara para escanear canales (16Ch) usado por el coordinador cuando forma una red. El coordinador puede realizar un escaneo a todos los canales SC habilitados. Puede realizar un escaneo de PAN ID y formar la red sobre uno de los canales SC
SD	Configura el tiempo de escaneo. Este valor determina que tanto tardará el tiempo de escaneo del coordinador sobre un canal dado
ZS	Configura el perfil de la pila del ZigBee para la red
EE	Habilita o deshabilita la seguridad de la red
NK	Configura el tipo de clave de seguridad para la red. Si se pone en 0 (por defecto), el sistema usa una clave de seguridad aleatoria.
KY	Configura la clave de enlace de confianza para la red. Si es puesta en 0 (por defecto), una aleatoria clave de enlace es usada
EO	Configura la política de seguridad para la red

Tabla 1.7: Tabla de comandos del coordinador
Fuente: Tomado de Manual XB_ZigBee_OEM Digi International, 2010.

cimiento, retardo de cambio o sin reconocimiento ver Tabla 1.7. De igual manera, si después de cambiar los parámetros se desea mantener la configuración después de un *reset*, se debe escribir el dispositivo y si se desea cambiar de coordinador es sino cargar los mismos parámetros en el coordinador de reemplazo.

Los *Routers* o Enrutadores ZigBee inician con un escaneo de canales y cuando encuentran uno disponible, envían una trama que es recibida por un coordinador, el cual determina si acepta o no la conexión. El *router* escanea 9 veces por minuto los primeros 5 minutos y luego 3 veces por minuto. Una vez el *router* se ha conectado, guarda el ID, canal, tipo de seguridad a utilizar y distribución de la red ante eventuales reinicios y deja de realizar la búsqueda. Sin embargo estos parámetros de tiempo de búsqueda son modificables por programación. El ZigBee chequea constantemente la conexión con el coordinador mediante una alarma *Watchdog* (Perro Guardian) configurable de 0 a dias. Igualmente se chequea la potencia dentro de la red, si el *router* detecta disminución de potencia o se cumple el tiempo del *Watchdog* (se reinicia cada vez que se recibe dato del coordinador) se saldrá de la red y buscará una nueva la cual se hará inicialmente con el mismo ID, en caso contrario cambia a otro.

Los dispositivos finales se pueden enlazar con la red ZigBee al igual que entrar en modo de "Dormir", sin embargo, en este modo no se podrán conectar otros dispositivos a este ni enrutar paquetes. Al igual que los *routers* usan un PAN *scan* y si encuentran una ID valida enviará un requerimiento de asociación para conectarse y si la red no está saturada (Max 10 ED por Coordinadores y 12 por

Routers, solo con ED) será aceptado en la red mandando una dirección de 16 bits. Si luego de escanear canal por canal no encuentra *routers* o coordinadores pasará al modo de dormir y solo saldrá de este modo si se dan un conjunto de mensajes por parte de los coordinadores o *routers* para despertarlo. Sin embargo si el dispositivo final está enlazado es posible almacenar la información y direcciones de la red escribiendo el dispositivo, así como también entrar en el modo de bajo consumo mediante un conjunto de mensajes. Si hay problemas de seguridad o potencia por estar fuera del rango, el dispositivo final tiene igualmente la capacidad de buscar una nueva red y enlazarse con esta.

El escaneo para la conexión se establece en la red PAN mediante un enmascaramiento de bits, luego se busca en los canales del 11 al 25 y si se sale empezará desde el canal que iba hasta el más alto. Sin embargo es posible definir inicialmente una red con un filtro PAN ID especificando un número, preconfigurando la seguridad y habilitando la unión de dispositivos coordinador, *router* y dispositivos finales, donde quien se desee enlazar, ya tendrá un mensaje definido dentro de la red y si no hay enlace buscará otra red.

Los sistemas de transmisión regidos por el estándar IEEE 802.15.4 se definen con una implementación en cinco capas que son: Capa Física, Capa MAC (*Media Access Control*), Capa de Red, Capa APS (*Application Support Sublayer*) y la Capa ZDO (*ZigBee Device Objects*). Además de definir las cinco capas, se establecen según la norma cinco modos de operación diferentes que son *Idle Mode* (Modo Inactivo), *Transmit Mode* (Modo de Transmisión), *Receive Mode* (Modo de Recepción), *Command Mode* (Modo de Comandos) y *Sleep Mode* (Modo de descanso) al cual solo pueden entrar los dispositivos finales.

En la capa de red se definen las direcciones y que dispositivos pueden hacer parte de la red, brindando esto seguridad, teniendo direcciones de 16 y 64 bits con 14 canales en ProZigBee(S2) y 16 canales en ZigBee común. Adicionalmente, el dispositivo puede inferir el PAN ID de la red seleccionando a quien conectarse además de que si hay varios dispositivos con 16 bits, migrar a conectividad con 64 bits. O inclusive si hay una dirección repetida en la red el dispositivo tiene la capacidad de cambiar su propia dirección cuando se va a incorporar a esta red. La Capa APS se utiliza en el modelo de pila permitiendo soportar perfiles (comentado anteriormente), *clusters* y puntos finales.

Los *clusters* son mensajes de aplicación definidos dentro de un perfil que sirven para especificar función, acción o servicio. Cada *cluster* tiene 2 bytes de identificación que están en toda la comunicación y tienen asociadas peticiones y respuestas. Los *cluster* se pueden definir en una librería e implementar en varios perfiles y las tramas API pueden ser usadas para enviar o recibir mensajes sobre un *cluster*.

La capa ZDO es la encargada de soportar los perfiles de la capa APS y administrar estos. Dentro de esta capa están los servicios ZDP (*ZigBee Device Profile*), donde se asocia el *cluster* ID a cada dispositivo. Los dispositivos finales permiten correr aplicaciones como comunicación TCP/IP, y empalmar con la capa ZDO conectando hasta 240 dispositivos. Se puede adicionalmente en esta capa adminis-

Comando	Descripción
D5	Habilita el LED asociado a la funcionalidad del dispositivo.
LT	Configura el tiempo de parpadeo del LED asociado, cuando hay enlace. Por defecto es 2 parpadeos por segundo.
NJ	Configura el tiempo en segundos para permitir enlace o para enlazarse con los otros dispositivos de la red. Si NJ=0xFF permite siempre enlazarse.
SM, SP, ST, SN, SO	Parámetros que configuran las características del modo dormir. Solo dispositivo final.

Tabla 1.8: Tabla de comandos para comportamiento de los dispositivos.
Fuente: Tomado de Manual XB_ZigBee_OEM Digi International, 2010.

trar la red en un dispositivo final mediante comando ZDO que se envían en las tramas API y los bytes de la carga en requerimiento se envían de manera inversa.

Los ZigBee poseen 2 direcciones, una de 64 bits que es la MAC(IEEE) del dispositivo y la de 16 bits que es la de la red. Donde la dirección 0x0000 es la del coordinador y este le asigna la dirección a los otros dispositivos que estén en el enlace. El *router* utiliza la dirección de 16 bits para enviar paquetes y tablas de enrutamiento, pero no es confiable para identificar dispositivos pues esta puede cambiar en cualquier momento, mientras que la de 64 bits no cambia. Con el firmware API se pueden enviar datos a cualquier perfil, *cluster* o punto final con una aplicación de capa de direcciones.

Todos los ZigBee poseen una tabla de datos que le permite almacenar simultáneamente hasta 10 direcciones de 64 bits y 16 bits, la cual permite direccionar las transmisiones a 64 o 16 bits. Si no encuentra la dirección de 16 bits seguirá buscando hasta lograr su transmisión. Si se tienen más de 10 dispositivos conectados a un *router* o coordinador se puede almacenar en tramas API los dispositivos, además de dar información de recepción, nodos, indicador de enrutamiento y estado de transmisión. Cada uno de los dispositivos de la red permite configurar su comportamiento dentro de la red con un conjunto de comandos que se pueden visualizar en la Tabla 1.8. De la trama API se puede descargar la dirección de 16 bits a la tabla cuando esté desocupada.

Cada uno de estos dispositivos soporta transmisiones de hasta 84 bytes por trama, donde la carga útil se reduce por los bytes utilizados para seguridad y fuentes de enrutamiento. Sin embargo se pueden hacer paquetes de 255 bytes y aunque se transmite por subpaquetes, la salida no se dará hasta que no sea ensamblada de nuevo. Si alguno de los paquetes falla se deberá realizar una retransmisión.

1.3.1. Enrutamiento

El proceso de comunicación regido por la norma 802.15.4 se maneja con mensajes cíclicos para verificar los dispositivos que están en la red y actualizar las tablas de enrutamiento. Cada *router* o coordinador envía un mensaje de estado inicial cada 2 segundos y luego de que se enlaza lo hace de 3 a 4 veces por minuto, siendo un enlace muy confiable y permitiendo restablecer la ruta cuando la señal es débil. La ruta se describe con AODV (*Ad-oc On demand Distance Vector*) que funciona con tablas donde se almacenan los saltos. Con la señal de broadcast inicial se descubren los caminos de la red y luego se selecciona el mejor dejando las otras rutas como alternas. Los paquetes se pueden direccionar con la MAC o con la dirección ID y si estos se reciben correctamente se envía una trama de reconocimiento ACK. Si la señal no se recibe correctamente se puede realizar una retransmisión hasta por 4 veces.

En el caso dado, que muchos dispositivos estén enviando al mismo tiempo una señal a un dispositivo central, el AODV se saturaría, por lo que se utiliza la técnica de "many to one" que establece rutas diversas para cada uno de los vecinos almacenándolas en tablas, determinando el tiempo para el envío de mensajes. La tabla AODV se puede reescribir y permite descubrir rutas de varios dispositivos con el inverso de "many to one". Cuando hay muchas rutas, el dispositivo deberá enviar una trama con las direcciones y saltos grabadas en la ruta además de la dirección del dispositivo al que se le envía la información.

Las rutas almacenadas por los dispositivos remotos, se pueden enviar a los *routers* como envío periódico de paquetes. Una vez recibida la ruta con una trama API, esta se almacena en la aplicación como una trama "API de ruta" la cual se lleva a la tabla interna del *router*. En caso de carecer de la trama "API de ruta", esta se puede construir con las direcciones y los saltos de la red. Los dispositivos que trabajan con la aplicación "Source Route" pueden enviar mensajes "many to one" para refrescar la ruta. Al perder una ruta, los dispositivos pueden enviar una nueva transmisión grabada. ZigBee no solo recibe reconocimiento del vecino, donde si no es reconocida la trama enviada realizará 4 retransmisiones, pero también recibe reconocimiento del destino, si se carece de respuesta, realizará 2 retransmisiones.

El enrutamiento de los paquetes se desarrolla con una codificación y decodificación de la señal con la clave de seguridad del sistema. Donde en cada salto de la ruta se realiza un proceso de decodificación y se vuelve a codificar con su dirección y valor de trama. *Broadcast* soporta 8 bytes más que "unicast" y "source routing" va incluido en el *payload* de la trama reduciendo la cantidad de bits transmitidos por salto (16bits), de igual manera si el cifrado es APS se reduce en 4 bytes.

El rendimiento en el ZigBee varía según la seguridad, saltos, enrutamiento y el número de dispositivos, teniendo en cuenta la velocidad de transmisión. Además según sean los saltos y tipo de trama los tiempos de espera varían, es decir, si es una trama *unicast* utilizada por un *router* es de 1.6 segundos por defecto, esta se puede configurar y variar con tramas APIs según sea el valor de NH (*Number*

Hops) el cual tiene por defecto el valor de 30 y se aplica en la siguiente ecuación:

$$t = 50 \cdot NH + 100 \tag{1.1}$$

Pero si es una transmisión extendida a varios dispositivos se realiza el envío a 10 dispositivos finales, se tiene en cuenta el tiempo que tardarían estos en despertar para un total de 2.8 segundos por defecto, aunque es configurable según sea el parámetro NH y SP (*Sleep Period*) aplicados en la Ecuación 1.2

$$t = (1{,}2 \cdot SP) + (50 \cdot NH) \tag{1.2}$$

El comando NH es un comando AT al igual que el SP.

1.3.2. Seguridad

El ZigBee maneja varias seguridades como cifrado a 128 bits con seguridad AES (*Advanced Encryption Standard*), seguridad de enlaces, tabla de información como soporte para centros o puntos de confianza, mensajes de integridad, mensajes de confianza y autenticaciones. En la parte de seguridad de los enlaces se tienen 2 seguridades, una por saltos y otra por *unicast*.

Con los mensajes se tienen 3 seguridades, una conocida como residencial, la cual se dá cuando se comparte información con los dispositivos de la red, donde dentro de la información compartida, está el tipo de seguridad establecida. La otra es la seguridad estándar donde se tiene en la capa APS una clave de enlace con los dispositivos de la red. Por último, se tiene la seguridad alta, la cual necesita autenticación de los dispositivos y del envío de mensajes para que estos sean aceptados en la red.

En conjunto, las 3 seguridades anteriores se tienen implementadas sobre la capa de red. Adicionalmente se tiene contador de trama, mensaje de integridad de código, cifrado de capa, clave de descarga, APS y centro de confianza, las cuales interactúan con las seguridad residencial, estándar y alta.

1.3.3. Puesta en Marcha de la Red y Diagnostico

Cuando se configura la red de los dispositivos regidos por la norma IEEE 802.15.4, se busca que las rutas de comunicación sean confiables, además de tener opciones de diagnóstico y configuración en red, ya sea con comandos AT o con tramas API, siendo este último más utilizado en conexiones remotas.

Además se pueden establecer los lugares donde es más confiable el enlace lo cual se puede realizar con envío *unicast* de varios paquetes y medir la tasa de recepción o con un *loopback* o realimentación de paquetes. También se puede realizar medida de la calidad de los enlaces mediante la medida del RSSI (*Received Signal Strength Indication*) con el que cuenta el dispositivo, que indica el nivel de potencia recibida

Veces que se presiona	Si está enlazado en la red	Si no está enlazado en la red
1	* Despierta un dispositivo final * Envía un mensaje de difusión con identificación	* Despierta un dispositivo final * Parpadea el pin que indica la causa de error en el enlace mediante un número codificado
2	* Envía una trama de difusión para habilitar el enlace con el coordinador y los dispositivos de la red por un minuto	* N/A
4	* El dispositivo se sale de la PAN * Restaura los parámetros del módulo a los valores por defecto, incluyendo ID y SC * El dispositivo trata de unirse a una red basado en el ID y SC configurado	* Restaura los parámetros del módulo a los valores por defecto, incluyendo ID y SC * El dispositivo trata de unirse a una red basado en el ID y SC configurado

Tabla 1.9: Tabla de comandos para comportamiento de los dispositivos.
Fuente: Tomado de Manual XB_ZigBee_OEM Digi International, 2010.

en -dBm del último salto. Esta potencia puede ser leída por comandos o en el Pin 6 del dispositivo Digi siempre y cuando se haya habilitado el PWM del mismo.

Para descubrir una red, se puede enviar un *broadcast* a todos los dispositivos y estos contestan con la dirección, nodo e información relevante. Aunque el *broadcast* es simultaneo para todos los dispositivos, cada dispositivo espera un tiempo aleatorio para contestar con un máximo de 6 segundos. También es posible descubrir la red con comandos ZDO, lo cual permite tomar ZigBee de otras marcas, leyendo las tablas de los vecinos con los requerimientos de descarga. La otra forma se da cuando un ZigBee se une a la red y envía un anuncio ZDO con el *cluster* el cual incluye direcciones, capacidades, entre otros aspectos importantes.

El dispositivo ZigBee de Digi, si se desea, se puede activar mediante el botón de puesta en marcha que es el pin 20 del integrado. Una vez configurado, cumplirá diferentes funciones según las veces que se presione y si el dispositivo está o no enlazado en la red. Estas funciones se pueden visualizar en la Tabla 1.9. Antes de configurar el botón de puesta en marcha, este se puede simular por software con el comando ATCB, seguido del número de veces que se desee presionar el botón (ATCB2). La trama de identificación de pulso es similar a la de respuesta, lo cual es enviado a la UART como un API de identificación.

Existe un LED (pin 15 del Digi) asociado con la puesta en marcha del dispositivo que por defecto se encuentra habilitado. Cuando el módulo no está asociado este se pone en 1. Pero una vez se asocia, este parpadea un intervalo de tiempo

programado. El LED también trabaja con el botón de enlace, el cual si detecta un error, parpadea un número de veces, que indica el error en el número de parpadeos donde si no hay parpadeos es 0x20 en el registro AI, pero si hay por ejemplo 2 parpadeos, este tomará el valor de 0x22. Al no presentarse ningún error, parpadeará una sola vez y seguirá arriba indicando el correcto funcionamiento del enlace después de transmitir el paquete de identificación *broadcast*.

El dispositivo ZigBee permite configurar sus pines como entradas o salidas análogas-digitales, mediante un número, un comando y según las especificaciones del dispositivo. Sin embargo por las aplicaciones utilizadas en este trabajo, esta opción no se utilizó pero brinda flexibilidad a la hora de usar los dispositivos.

Adicionalmente según todas las características establecidas en este marco teórico, se puede apreciar, que la mayoría de parámetros y configuración del dispositivo para obtener su máximo rendimiento se establece con la capa APS y sus tramas API. Sin embargo, para la aplicación utilizada en este trabajo se estableció en el modo de *bypass* (AT/Transparent mode), por lo que no se profundiza mucho en la comunicación por este medio, sino que se da una mirada general a sus aplicaciones y conexiones posibles.

1.4. Comunicación GSM

La red GSM (*Global System Mobil o Groupe Special Mobile*) es una de la tecnologías que permite la conexión de telefonía móvil celular. Este estándar europeo se estableció por la CEPT en 1982 con el fin de unificar los distintos sistemas móviles digitales y permitir la movilidad entre cualquier país europeo (S.A, 1998). Entre sus principales características, está la tarjeta inteligente, cifrado de las conversaciones, integración de voz y datos, la utilización más eficiente del espectro y menor consumo de energía con terminales más pequeños.

Este sistema maneja 4 bandas (850/900/1800/1900), con 8 canales por portadora (124 por antena), con separación entre portadoras de 200khz, y ancho de banda del canal de radio de 25khz. Su transmisión es TDMA/FDD (*Time Division Multiplex Access/Frecuency Division Duplexing*) con modulación GMSK (*Gaussian Minimunn Shift Keying*) la cual recibe la voz codificada a 13kbps y esta se transmite a la antena con una velocidad binaria de 22.8kbps (Huidobro, 2002).

Esta red está compuesta por un conjunto de BTS (*Base Transceiver Station*) encargadas de crear la red hexagonal de células, las BSC (*Base Station Controler*) que controlan las estaciones base y la MSC (*Mobile Switching Center*) que es donde se realiza todo el proceso de conmutación. Adicionalmente se tienen otros bloques funcionales como el HLR (*Home Location Register*) que almacena los datos estáticos relativos al abonado móvil como el registro de posiciones, VLR(*Visitor Location Register*) que almacena toda la información sobre el abonado móvil que entra en una zona, OMC (*Operation Manager Center*) para realizar las funciones

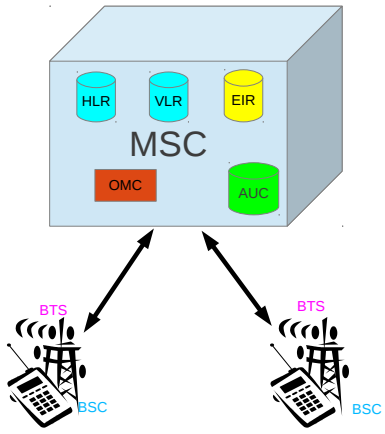


Figura 1.11: Bloques funcionales de la red GSM.
Fuente: Elaboración propia.

de operación y mantenimiento de la red, EIR(*Equipment Identity Register*) que es una base de datos de información sobre el estado de los móviles IMEI (*International Mobile Equipment Identity*) y la AuC (*Autentication Center*) donde se registra la identidad del usuario almacenada en la SIM (*Suscriptor Identification Memory*) y permite la comunicación dentro de la red protegiendo esta de intrusos (Moya, 2004). Esta distribución se puede visualizar en el diagrama de bloques de la Figura 1.11. Cada vez que un dispositivo terminal se ha autenticado en la red, se le asigna un alias temporal que permite actualizar la posición y ubicación del dispositivo.

Una de las principales ventajas de la red GSM es que permite gran capacidad de usuarios debido a la utilización eficiente del espectro y su amplia cobertura, pues se realiza reutilización de frecuencias entre estaciones bases no adyacentes. Además permite *roaming* (seguimiento del móvil según la posición donde se encuentre) internacional. Con GSM se mejora en: la calidad de las señales, confidencialidad en la información y la identidad de los abonados, la seguridad, evitando usos fraudulentos del sistema y se introducen nuevos servicios en datos, mensajes y voz.

Cada canal físico de la red GSM tiene una duración de 577μs en la división TDMA dentro de uno de los enlaces de voz, teniendo en cuenta que se tiene un canal ascendente y otro descendente. Luego al tener 8 canales por portadora, la trama total dura 4.62ms. Dentro de estos canales físicos, están los canales lógicos, los cuales se clasifican según el tipo de información transmitida, como datos de abonado, señalización, control o datos de tráfico.

El canal de tráfico (TCH) contiene la voz codificada o datos transmitidos y recibidos. Este canal es de 22.8kbps, 13kbps se utilizan para la codificación de la voz y el resto como bits de corrección de errores. El canal de control contiene la señalización o datos de sincronización que se dividen en 3 canales. Un canal de

radiodifusión (BCH) el cual transmite información de correcciones de frecuencia de la BTS a los dispositivos móviles. Un segundo canal común de control (CCCH) que transmite punto a punto información de localización del móvil con peticiones de acceso. El último canal de control dedicado (DCCH), sirve para la iniciación de las llamadas y envío de medidas de potencia y señalizaciones espaciales de *handover* (Bates, 2003).

1.4.1. Señalización dentro del sistema de conmutación

La señalización en el sistema de conmutación GSM está basada en las recomendaciones CCITT de canal común Nº7. Sin embargo hay un nuevo esquema con estándares más altos entre la señalización dada de la MSC con los bloques HLR, VLR, AUC, EIR y otras MSC dada en la parte de aplicación de los dispositivos móviles (MAP). Los niveles bajos de señalización son gestionados por la parte de transferencia de mensajes (MTP) dentro del sistema de señalización CCITT Nº7. Entre MAP y MTP se utiliza la parte de aplicación de las capacidades transaccionales (TCAP) y la parte de control de la conexión de señalización (SCCP).

La señalización debe operar internacionalmente entre las redes GSM y los registros de posición. Esta señalización es adicional al tráfico telefónico convencional, puesto que el GSM ha introducido nuevas características como el seguimiento internacional y la autenticación.

Otros procedimientos gestionados por la MAP (*Movil Aplication Part*) son el traspaso de llamada (Cambio de radiocanal), transferencia de información de abonado, actualización de la posición del móvil en los registros de localización, gestión de servicios de abonado y transferencia de datos de seguridad.

La señalización entre el sistema de conmutación y el sistema de estaciones base (Interfaz A), se realiza según la parte de aplicación del sistema de estaciones base (BSSAP) dentro del sistema de señalización CCITT Nº7. Esta parte de aplicación se encarga de la asignación y liberación de los recursos de radio con traspaso de llamada, control de llamada, gestión de movilidad y envío de la tripleta de datos de abonado brindados por la AUC como son RAND (Numero Aleatorio de Acceso a la BTS), SRES (Respuesta con verificación) y KC (Clave de cifrado).

Los niveles bajos de señalización son gestionados por la parte de transferencia de mensajes (MTP) del sistema de señalización por canal común CCITT Nº7. La señalización entre la BTS y la BSC (Interfaz A-bis) es implementada como un esquema especial de señalización por canal común para canales PCM de 64kbps. Finalmente uno de los canales que se usa como canal de señalización es el canal D, el cual transporta información de señalización según el protocolo de acceso de enlace usando LAPD (*Link Access Procedure on D channel*) aplicado para la red GSM.

1.4.2. Datos en GSM

Desde que inició la red GSM, se ha buscado agregarle más prestaciones al sistema, complementando la red GSM como tal. Entre las prestaciones más importantes y que siguen hoy en evolución, son las de transporte de datos, que han pasado por WAP (*Wireless Aplication Protocol*) , luego se evolucionó a GPRS (*General Packet Radio Service*) que permite transportar datos con un factor superior a 10 la velocidad de acceso de GSM. GPRS logró velocidades de transmisión de hasta 115kbps, siendo su directo competidor la tecnología CDMA (*Code Division Multiplex Access*) del estándar americano.

La arquitectura WAP, se basa en el *World Wide Web*, pero adaptada a los dispositivos móviles y las especificaciones de los mismos, donde un terminal móvil solo podía navegar en páginas diseñadas bajo WAP. Con un navegador especifico (que decodifique WAP). Con el navegador se realiza la coordinación de peticiones de información a la red. Luego el servidor procesa la información y la transmite según el derecho de petición de datos. Con WAP es posible eliminar imágenes innecesarias descritas en HTML y lo más importante es que la información se adaptaba a la pequeña pantalla clásica que se tenía a principios del 2000 en los terminales móviles.

Con el objetivo de aumentar la velocidad se evolucionó a la red EDGE (*Enhanced Data rates for GSM Evolution*) que es soportada igualmente por GSM. Luego se ha pasado a lo que hoy conocemos como UMTS (*Universal Mobile Telecomunication System*) que implementa el transporte de datos con WCDMA, seguidamente, HSDPA (*High Speed Downlink Packet Access*) y hasta llegar a las redes actuales 4G LTE (*Long Term Evolution*).

Capítulo 2

Comunicación PDA-ZigBee via GSM

Debido a que las PDA y/o los *SmartPhone* se han convertido en computadores de bolsillo, donde se integra la conectividad como telefonía móvil, Wi-Fi y ZigBee como se desarrolla en esta fase y se expone en este capítulo. Con esta conectividad se busca integrar la red corporativa de una empresa con las PDA, incrementando la eficiencia y desempeño de algunos sistemas industriales, mediante el diagnóstico de fallas en tiempo real (Nieto, 2009) utilizando estos medios de comunicación.

Para aplicar la comunicación PDA-ZigBee es necesario determinar una metodología aplicable que permita realizar la interfaz transversal para los 2 dispositivos, la cual se establece sobre la red GSM permitiendo mayor movilidad a la PDA. Esta comunicación entre los dispositivos móviles y el módulo Zigbee se logra con tramas de datos básicas con el servicio de mensajería corta (SMS, por sus siglas en ingles), a través de la red del Sistema de telefonía Móvil Global (GSM), que es recibida por el ZigBee remoto utilizando un microcontrolador para el control de los módulos, Figura 2.1.

Descripción del Sistema: En esta fase de comunicación se establecen 4 bloques funcionales que son la base para interconectar los dispositivos planteados. Cada uno de estos tiene unas características y modos de configuración diferentes que hacen posible la integración de las diferentes tecnologías utilizadas. Los bloques funcionales que se describen en este capítulo con sus principales parámetros de configuración son: el módulo ZigBee, el módulo GSM, el microcontrolador y la PDA, como se aprecia en la Figura 2.1.

2.1. Descripción Módulo ZigBee Xbee Pro S2B

El módulo ZigBee utilizado para la interconexión con la PDA es el Xbee Pro S2B de Digi International, cuya referencia es XBP24BZ7WITB003 Figura 2.2, el cual se caracteriza por tener un Xmodem que recibe paquetes de 64Kbytes

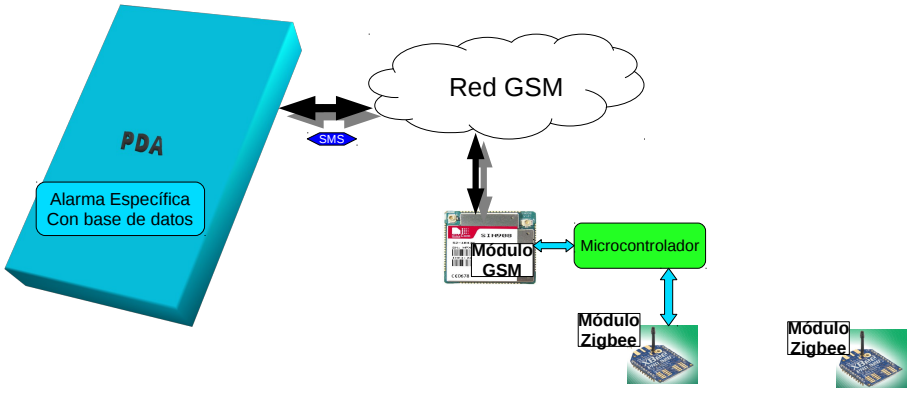


Figura 2.1: Bloques funcionales comunicación Pda-Zigbee via GSM
Fuente: Elaboración propia.

para la programación y conexión con el PC, un microcontrolador interno Freescale MC9S08QE32 de 2Kbytes de memoria RAM, una memoria flash de 32 KBytes y con un oscilador interno que puede operar hasta 50.33MHz. Posee una antena tipo latigo en alambre para trabajar a una frecuencia de 2.4GHz que permite un alcance máximo de alrededor de 1500 metros en espacio abierto y una velocidad de comunicación máxima de 250kbps. Su potencia máxima de transmisión es de 10mW (+10dbm) los cuales son ajustables y un umbral en sensibilidad de recepción de -102dBm.

Se puede configurar por comandos AT y API, posee interfaces CMOS UART, SPI, I2C, PWM, DIO (10 Digital Input Output) y ADC (4 Analog/Digital Converter a 10 bits). Las interfaces seriales son configurables de 1200bps a 1Mbps y funciona en un rango de voltaje de 2.7 voltios a 3.6 voltios, siendo 3.3V el estándar. Tiene un consumo de corriente de 220mA en la transmisión y 62mA en recepción y un nivel de bajo consumo de 4uA a 25°C. Puede trabajar normalmente en un rango de temperatura entre los -40°C y +85°C y una humedad de máximo 95 % no condensable.

La inmunidad de interferencia DSSS (*Direct Sequence Spread Spectrum*) del módulo, permite que este trabaje sobre un entorno industrial, además de tener la característica de reconocer los paquetes enviados y recibidos con chequeo de entrega, que si no se lleva a cabo, será reenviado el respectivo paquete (*XBee® ZB - Digi International*, s.f.).

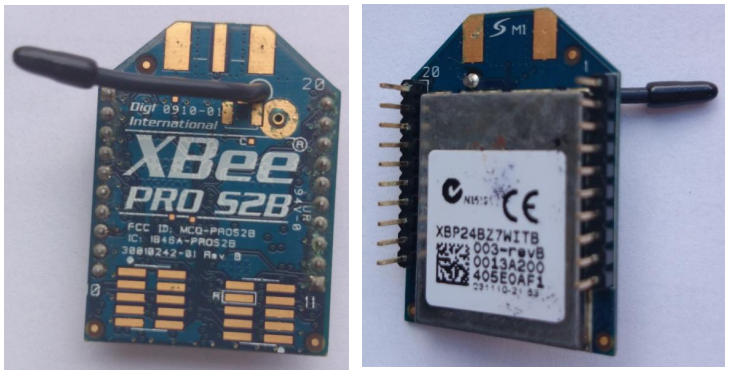


Figura 2.2: ZigBee parte Frontal y trasera.
Fuente: Elaboración propia fotos Zigbee.

2.1.1. Configuración

Para configurar el módulo ZigBee, se utiliza una base que posee el integrado FT232RL del fabricante FTDI y es el encargado de establecer el puente de conexión entre el puerto USB del computador y el modem RS232 UART del módulo Zigbee para su programación. Esta base además brinda la alimentación al módulo que es tomada del puerto USB del computador (5 voltios) y convertida a los 3.3 voltios que necesita el módulo. Tiene indicador de alimentación, transmisión y recepción de señales según sea el tráfico en el módulo Zigbee. También la base, posee un LED indicador de potencia de señal recibida RSSI, tomada del módulo ZigBee, el cual varía su intensidad según sea la fuerza de la señal recibida por el PWM de este, Figura 2.3.

Para la configuración del mismo se utilizó el software X-CTU propietario de Digi International con el cual se programan los dispositivos XBee y se configuran sus parámetros. Para este caso se utilizó la configuración Coordinador-Router. Donde los módulos XBee PRO S2B se programaron como tal, teniendo en cuenta que los primeros números del Firmware indican la operación del ZigBee como tal ver Tabla 2.1

El primer módulo ZigBee cuyo serial es 13A200406644C7 se configura como Coordinador AT al cual se le carga el Firmware 208C con versión de hardware 1E46, el PAN ID por defecto es 123 al igual que el OP PAN ID y un PAN ID de 16 Bits 1CBC. Los valores de escaneo, saltos, direccionamiento, seguridad, potencia, puertos I/O, modos, transmisión, entre otros, se dejan en los valores por defecto para esta aplicación ver apéndice A. La versión mostrada por el X-CTU para este módulo fué BL032-2B0-025_064. Donde 032 indica el tamaño de la memoria flash en kilobytes, 2B0 es el hardware, 025 es la versión y 064 es el tamaño de paquetes preferido por el XMODEM para la actualizaciones. Este módulo no posee dirección de red propia por ser el coordinador, pero si identifica la dirección de destino, la cual corresponde al serial del módulo ZigBee enrutador.

Firmware	Función
20XX	Coordinador AT/Funcionamiento en modo Transparente
21XX	Coordinador con Funcionamiento mediante API
22XX	Enrutador AT/Funcionamiento en modo Transparente
23XX	Enrutador con Funcionamiento mediante API
28XX	Dispositivo Final AT/Funcionamiento en modo Transparente
29XX	Dispositivo Final con funcionamiento mediante API

Tabla 2.1: Firmware ZigBee según sus funciones.
Fuente: Elaboración propia basado en Manual XB_ZigBee_OEM Digi International, 2010.

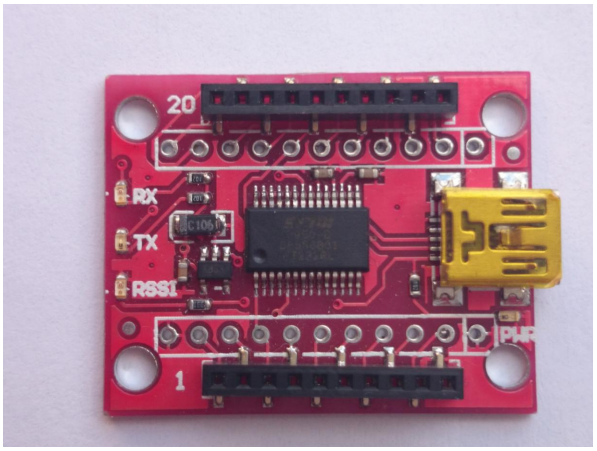


Figura 2.3: Base de programación para ZigBee.
Fuente: Elaboración propia.

El otro módulo ZigBee tiene el serial 13A200405E0AF1 el cual se configura como Enrutador AT con un Firmware 228C. Con PAN ID 0 pero el OP PAN ID si es el 123. El mismo PAN ID de 16 bits, pero con una dirección propia de red D292 e igualmente los otros valores de red, direccionamiento, seguridad, interfaces, comandos AT, entradas y salidas, se dejan con los valores por defecto como se muestra en el apéndice A. La versión de Hardware es 1E42 y la versión de todo el módulo aparece como: BL032-2B0-023_064 donde la única diferencia es en el 023 que indica la versión del mismo.

2.1.2. Conexión

Para conectar los módulos Zigbee con los microcontroladores, se usaron los pines 2 y 3 del módulo, que corresponden al TX y RX del UART interno. Aunque también se puede trabajar las interfaces digitales D7 y D6 correspondientes a los pines 12 y 16 respectivamente, que corresponden a las líneas RTS (*Ready to Send*)

y CTS (*Clear to Send*) de las conexiones RS232 o inclusive configurable para RS485. La programación de la memoria *flash* del módulo ZigBee para conexiones especiales se realiza sobre las direcciones de 0x8400 a 0xF1BC y el *bootloader* va desde 0xF200 hasta 0xFFFF, cuya programación si se desea, se puede realizar con programas compilados en CodeWarrior descargando el archivo .BIN generado.

La conexión inicialmente que se establece con el módulo ZigBee es a 9600 bps, sin control de flujo, bits de datos 8, sin paridad, y un bit de parada, que es lo que viene por defecto por el módulo para su configuración. Una vez se alimenta y conecta con esta configuración base, al darle la tecla "intro" (\r) en la terminal del X-CTU, el módulo Zigbee entra a un menú de selección básica que muestra 5 opciones que son: F (*Update App*), T (*Timeout*), V (*BL Version*), A (*App Version*), R (*Reset*), B (*Bypass Mode*). Con la opción F, se pueden escribir datos o transferir archivos .BIN generado en Code Warrior, pues esta opción corresponde a una actualización o programación del Xbee. La T, muestra y configura el tiempo fuera del dispositivo Zigbee, el cual se puede configurar con los 7 tiempos diferentes que vienen predefinidos con el módulo, los cuales van desde T0 a T6 y se cambian presionando la T. La V muestra la versión BL del dispositivo, comentada anteriormente. La A da la versión del programa cargado con el archivo .BIN, siempre y cuando este se haya configurado para determinada versión, si no es así, o no se ha cargado ningún archivo, muestra versión desconocida. La R, reinicia el dispositivo, pero no es un reinicio de firmware, ni de archivos cargados, sino que es un reinicio de apagar y prender el dispositivo, conservando los valores y datos programados. Finalmente la B es para poner a trabajar el dispositivo en modo transparente de paso de datos, en este modo funcionan los comandos AT y se pueden cambiar o leer los parámetros del dispositivo Xbee.

Luego de configurar ambos módulos (el coordinador y enrutador) en modo de paso (*bypass*) se realizan transmisiones entre estos chequeando la transferencia de caracteres ASCII y datos o mensajes. Adicionalmente se chequean algunos comandos AT, para lo cual, se debe presionar 3 veces la tecla "+" y una vez el módulo conteste con OK, es porque se ha entrado al modo de control de comandos AT. Si está correctamente en este modo al mandarle AT el módulo responde OK. Por ejemplo, si desea saber la temperatura del módulo es ATTP o así cada uno de las funciones que se deseen realizar por control de comandos AT.

Una vez se realizan las pruebas de conexión, se procede a programar el microcontrolador, de tal forma que cumpla con las funciones deseadas para la conexión de estos módulos ZigBee con el módulo GSM.

2.2. Descripción Módulo GSM SIM908

El módulo utilizado para la interconexión entre la PDA y el ZigBee es el SIM908 de SIMCOM, el cual es un dispositivo que integra con alto rendimiento el módulo GSM/GPRS con GPS. Trabaja en las bandas de GSM 850MHz, EGSM 900MHz, DCS1800MHz y PCS1900MHz. Posee un GPRS multi-slot clase 10 que soporta

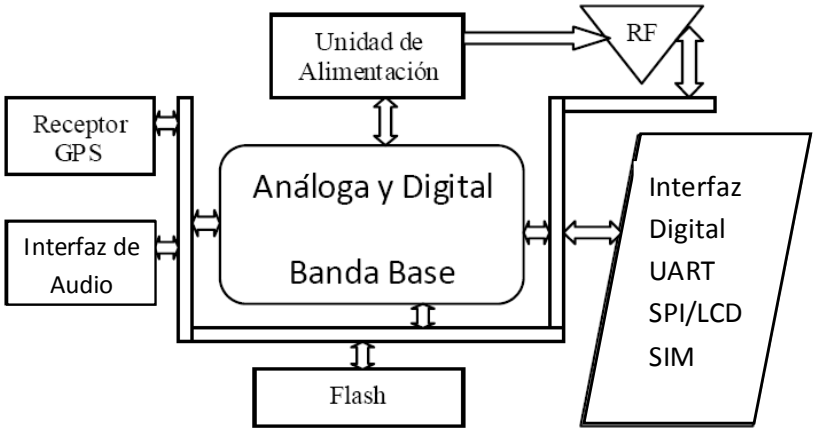


Figura 2.4: Diagrama de bloques módulo SIM908.
Fuente: SIMCOM

diferentes esquemas de codificación y puede integrarse con el protocolo TCP/IP. Posee dos puertos seriales uno para la conexión con la tecnología GSM y otro para el módulo GPS y conectores de interfaz RF para cada tecnología. Además tiene dos canales de audio con entradas y salidas cada uno, interfaz de carga, entre otros periféricos que se muestran de manera resumida en el diagrama de bloques de la Figura 2.4. Para transferir la información entre el microcontrolador y el módulo GSM se utiliza el puerto serial que mediante comandos AT se realiza su configuración, estableciendo la comunicación módulo GSM-terminal celular (PDA) (*SIM908 Hardware Design V1.01*, 2011).

Entre otras características del módulo, están los rangos de temperatura en operación normal que van desde los -30°C a los 80°C, con velocidades de transmisión de GPRS en subida de máximo 42.8 Kbps y de bajada 85.6 Kbps como máximo. Además soporta PBCCH (*Packet Broadcast Control Channel*), transmisión CSD (*Circuit Switched Data*), servicios USSD (*Unstructured Supplementary Services Data*), reloj real (RTC, por sus siglas en ingles) y hasta FAX Grupo 3 Clase 1. El audio es codificado con cancelación de eco, supresión de ruido y tasa multi adaptable (AMR). La operación con la tarjeta SIM es normal, permitiendo almacenar, borrar, editar mensajes y contactos.

El SIM908 también puede trabajar con batería externa aparte de la alimentación con fuente directa, lo que hace que este trabaje con 2 modos de operación adicionales que son el modo de solo carga o modo de carga con operación normal. En el modo de solo carga recibe ciertos comandos AT y no tiene conexión establecida con la red y se dá cuando el módulo se encuentra en estado apagado. El otro modo de operación se establece mientras el dispositivo esta operando normalmente y es conectado un cargador al mismo y la batería está por debajo de los 3.2 voltios. El apagado del dispositivo se puede realizar con comandos AT o

por la tecla de encendido y una vez se apague solo se mantendrá la alimentación del RTC. Este módulo puede trabajar en varios modos como GSM *Idle* (Módulo registrado en la red y listo para comunicar) GSM *Talk* (Comunicación establecida entre 2 usuarios), GPRS *Standby* (Módulo listo para transferencia de datos pero sin envío o transmisión), GPRS *Data* (Con transferencia de datos y uso de la red) y GSM/GPRS *Sleep* donde su consumo es de solo 1mA. Existe un último modo de operación que es el de "Funcionalidad Mínima", en el cual no es accesible la SIM, la parte de RF de GSM no trabaja, pero si se tiene comunicación con el puerto serial, lo que permite cambiar parámetros de configuración del módulo o trabajar con la parte de GPS del mismo (*SIM908 Hardware Design V1.01*, 2011).

Este módulo se implementa en una tarjeta PCB (*Printed Circuit Board*) diseñada para su respectiva conexión, gracias a sus reducidas dimensiones de 30mm x30mm x3.2mm un peso de 5.2 gramos, con sus 80 pines se crea una interfaz flexible que permita conectar no solo el microcontrolador, sino otros dispositivos que permitan la utilización de las principales funciones. Aunque la PCB se generó con este trabajo y se mandó a traer el módulo SIM908 directamente desde China para cumplir las funciones deseadas, el diseño de la misma es flexible, pues no solo sirve para este proyecto de investigación, sino que tiene múltiples aplicaciones y funciones Figura 2.5. Se tienen las principales interfaces, con su respectiva alimentación de 3.2 a 4.8 voltios regulados que permite realizar transmisiones de 2 vatios (Class 4) en las bandas de GSM850 y EGSM900 o a 1 vatio (Class 1) en DCS1800 y PCS1900. Estas bandas de frecuencia se seleccionan automáticamente aunque son modificables si se desea por comandos AT (AT+CBAND).

2.2.1. Conexión Módulo GSM

Aunque el SIM908 es GSM y GPS, de este solo se utiliza la parte de la comunicación GSM para conectarlo con el módulo Zigbee. Para su conexión y configuración se establece el puente a través del puerto serial con comunicación asíncrona mediante control por comandos AT, sin usar las líneas RTS y CTS. De la misma manera que con el módulo ZigBee se trabaja con los valores por defecto, para este caso se utiliza la configuración del SIM908 con sus valores por defecto el cual tiene la característica de auto-detectar la tasa de baudios, luego la conexión se establece con la misma velocidad que con el módulo ZigBee que es de 9600 bps, sin control de flujo, con 8 bits para los datos, sin paridad y un bit de parada.

Una vez configurados los parámetros anteriormente planteados, se procede a chequear la conexión con el mismo, para lo cual se envía el comando AT y la tecla "intro" (\r) y si el módulo está correctamente conectado este contesta con la palabra "OK". Una vez se dá la respuesta de "OK" se configura el formato para presentación de los mensajes SMS, que puede ser modo texto o modo PDU (*Protocol Description Unit*) que es una trama de octetos hexadecimales que representan cada una de las partes del mensaje incluyendo el texto. En esta conexión se utilizó el modo texto, lo cual se realiza enviando "AT+CMGF=1\r" pues si se desea en

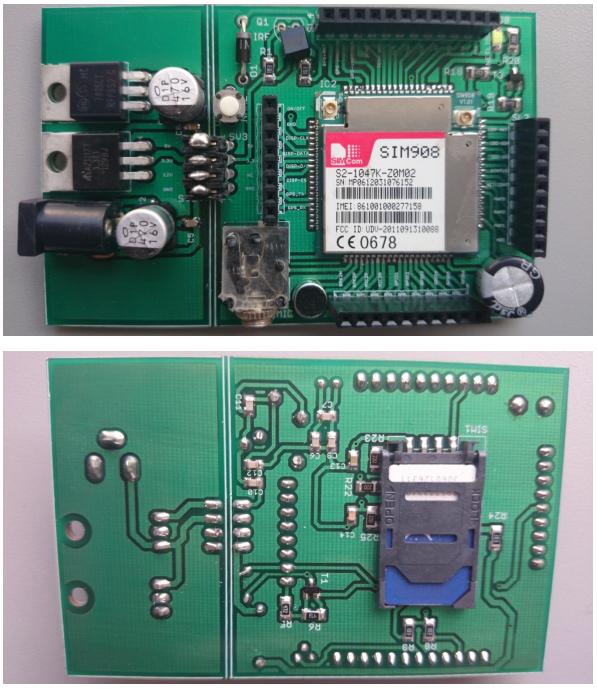


Figura 2.5: Tarjeta PCB del módulo SIM908 diseñada para este proyecto.
Fuente: Elaboración propia.

modo PDU es “AT+CMGF=0\r”.

Una vez se ha configurado el modo de los mensajes de texto (SMS), se procede a borrar el mensaje de texto almacenado en la posición 1 de la tarjeta SIM, de tal forma que cualquier mensaje de texto nuevo que llegue, se recibirá en esta posición. Para el borrado del mensaje de texto en esta posición se envia el comando “AT+CMGD=1\r” donde el número “1” indica la posición del mensaje a borrar. Una vez se tiene limpia la posición 1 de los mensajes SMS, se espera la recepción de un mensaje de texto nuevo, para lo cual se lee constantemente la posición 1 con el comando “AT+CMGR=1\r” y si esta es diferente de “OK” es porque se tiene un nuevo mensaje de texto que se almacenará. Una vez es leído el mensaje, este se borrará y se quedará esperando otra vez un nuevo mensaje de texto SMS.

Igualmente con el módulo GSM se hicieron pruebas de envío de mensajes de texto con el comando “AT+CMGS=“Número de Celular”\r donde seguidamente se escribe el mensaje de texto respectivo por ejemplo “HOLA MUNDO” y para enviarlo es con “Ctrl+Z” cuyo código en hexadecimal es “1A”. También con este módulo se hicieron llamadas con el comando “ATD” seguido del número telefónico y finalizando con “punto y coma” es decir que el comando completo es “ATD317XXXXXXX;” donde las “X” representan cada uno de los dígitos del número telefónico marcado. Luego para colgar el comando es “ATH”. Finalmente se recibieron llamadas y se contestaron estas con el comando “ATA”. Con todas estas

pruebas se verificó el correcto funcionamiento del módem GSM, aunque se pueden aplicar otros comandos AT para realizar otras funciones con el módulo los cuales se ven en la Tabla 2.2.

COMANDO	FUNCIÓN
Comandos Generales	
AT+CGMI	Identificación del Fabricante.
AT+CGSN	Obtener número de serie.
AT+CIMI	Obtener el IMSI.
AT+CPAS	Leer el estado del modem.
Comandos de Servicio de Red	
AT+CSQ	Obtener calidad de la señal.
AT+COPS	Selección de operador.
AT+CREG	Registrarse en una red.
AT+WOPN	Leer nombre del operador.
Comandos de Seguridad	
AT+CPIN	Introducir el pin.
AT+CPINC	Obtener el número de intentos que quedan.
AT+CPWD	Cambiar password.
Comandos para la Agenda de Teléfonos	
AT+CPBR	Leer todas las entradas.
AT+CPBF	Encontrar una entrada.
AT+CPBW	Almacenar una entrada.
AT+CPBS	Buscar una entrada.
Comandos para SMS	
AT+CPMS	Seleccionar lugar de almacenamiento de SMS.
AT+CMGF	Seleccionar formato de los mensajes SMS.
AT+CMGR	Leer un mensaje SMS.
AT+CMGL	Lista los mensajes nuevos almacenados.
AT+CMGS	Envía mensaje SMS.
AT+CMGW	Almacenar mensaje en memoria.
AT+CMSS	Enviar mensaje almacenado.
AT+CSCA	Establecer el centro de mensajes a usar.
AT+WMSC	Modificar el estado de un mensaje.
Comandos para Llamadas	
ATA	Contesta llamada entrante.
ATH	Cuelga llamada en línea.
ATD	Realiza marcación al número telefónico deseado.

Tabla 2.2: Principales comandos AT del módulo GSM.
Fuente: Elaboración propia basado en manual SIMCOM

2.3. Microcontrolador

La integración entre el módulo ZigBee y el módulo GSM se realiza mediante un microcontrolador, el cual establece el puente de conexión entre los dos dispositivos. Para esto se puede usar un microcontrolador con dos puertos seriales, o conexión por USART (*Universal Synchronous Asynchronous Receiver Transmitter*) en conjunto con SPI (*Serial Peripheral Interface*) o I2C (*Inter Integrated Circuit*). Para esta fase se utilizó el microcontrolador PIC18F26k22 el cual se caracteriza por poseer dos puertos seriales o interfaz USART, donde por un puerto se realiza la comunicación con el módulo ZigBee y por el otro con el módulo GSM, sin utilizar las interfaces SPI o I2C por la facilidad de manejo y el conocimiento que se tiene del mismo.

El microcontrolador PIC18F26k22 de 28 pines, se caracteriza por tener 64 KBytes de memoria para programar, 3896 Bytes de RAM, 1024 Bytes de EEPROM, velocidad de reloj máxima de 64MHz, e interna de 16MHz. Posee 2 comparadores análogos, 17 canales ADC de 10 bits y una tasa de muestreo de 100Kbps, 4 Timers de 16 bits y tres de 8 bits preescalables, 3 PWM con 10 bits de resolución de salida, polaridad configurable y con funciones de comparación o captura, 17 canales *Touch* capacitivos, 2 puerto USART, 2 I2C y 2 SPI. Trabaja en un rango de operación de -40°C a 125°C y voltaje de 1.8 a 5.5 voltios.

El encapsulado SPDIP de 28 pines, posee 25 pines configurables como entradas y salidas, además de tener reinicio por bajo nivel de voltaje y temporizador *watch dog*. Posee un procesador RISC con arquitectura optimizada para lenguaje C, instrucciones de hasta 16 bits con 31 niveles de pila y ciclos de instrucciones de 200ns. La tecnología CMOS con *NanoWatt*, es decir bajo consumo alcanza los 20nA y con *timer* habilitado de tan solo 800nA. Entre otras características están las detecciones de errores de oscilación, constatación de reloj, modo de operación de bajo consumo, código de protección en programación y diferentes configuraciones para atención a interrupciones con prioridades. Además este microcontrolador permite configurar 1 pin como salida de máxima corriente de 25mA y 9 *pull-up*, se pueden programar hasta 3 interrupciones externas, asimismo las opciones de auto-apagado y auto-reinicio (*PIC18F26K22 - 8-bit PIC® Microcontrollers*, 2012).

2.3.1. Conexión

La conexión entre los 2 módulos como se comentó anteriormente, se realiza a través de la interfaz USART que está como un módulo interno del microcontrolador PIC18F26k22. La USART soporta conexiones RS232, RS485 y LIN 2.0 (*Local Interconnect Network*), además de tener la capacidad de auto-detectar la tasa de bits y salir del modo de bajo consumo con un bit de inicio en la recepción de datos. Una ventaja de la USART es que funciona bajo interrupciones, lo que permite operar este módulo independiente de la ejecución del programa. Puede trabajar con comunicaciones *Half Duplex* en modo sincrónico o *Full Duplex* en

modo asincrónico, almacenar 2 datos de 8 bits en el *buffer* de entrada por su sistema FIFO (*Firts Input First Output*) y un dato en el *buffer* de salida, longitud variable de 8 a 9 bits en los datos, detección de errores en la velocidad de reloj, modo maestro/esclavo y polaridad programable en la transmisión de los datos.

La comunicación es controlada por cinco registros fundamentales para cada USART que son el registro de Estado de la Transmisión y Control (TXSTA el cual lo hay TXSTA1 y TXSTA2), registro de Estado de la Recepción y Control (RCSTA1,RCSTA2), el registro de Control de tasa de bits transmitidos (BAUDCON1, BAUDCON2) y los registro de datos para transmisión (TXREG1, TXREG2) que es el *buffer* del registro no accesible del TSR y para recepción (RCREG1, RCREG2) que es el *buffer* del registro no accesible del RSR. Se utiliza transmisión asíncrona con el estándar de no retorno a cero (NRZ) en 2 niveles de voltaje (0 y 5 voltios), con un bit de inicio, 8 bits de datos, 1 bit de parada, sin bit de paridad a 9600 bps.

La USART del microcontrolador en su modo asíncrono, tiene una estructura interna de conexión de bloques tanto para la parte de transmisión como para la recepción de datos. Esta estructura determina el funcionamiento de cada parte con sus respectivos registros, bits de control, banderas, oscilador con su tasa de transferencia de bits y un conjunto general de bloques que hacen posible la comunicación. Estos bloques funcionales se presentan en la Figura 2.6.

La implementación de la conexión del microcontrolador se lleva a cabo por cada uno de los puertos USART, donde por el puerto 1 se conecta el módulo ZigBee y por el puerto 2 el módulo GSM a cada una de sus respectivas terminales de TX y RX con el microcontrolador. Donde la USART 1 corresponde a los pines 17 y 18 del microcontrolador y la USART 2 a los pines 27 y 28. Adicionalmente se conectan 2 LEDs indicadores del estado de la comunicación en los puertos RB1 y RB2 correspondientes a los pines 22 y 23 respectivamente, donde por programa el LED 1 indica que la comunicación con el módulo ZigBee se ha establecido correctamente y el LED 2 indica que se ha enviado un mensaje de texto al módulo GSM, permitiendo verificar el estado de la comunicación garantizando el funcionamiento de esta. Figura 2.7.

2.3.2. Configuración

Para lograr la configuración ideal del microcontrolador, se utilizaron un conjunto de pruebas físicas y simuladas, donde se verifica la programación y configuración de cada uno de los dispositivos. Además se establecen los tiempos de retardo ideales y se solucionan los problemas presentados ante inconvenientes dados en la comunicación y lectura del mensaje.

Inicialmente el microcontrolador se configura para operar con el oscilador interno a una frecuencia de reloj de 16MHz, deshabilitando las opciones de perro guardián (*watch dog*), reinicio por bajo voltaje (*Low voltage*), al igual que el botón de reinicio (*master clear*). Se habilitan las interrupciones tanto las globales como

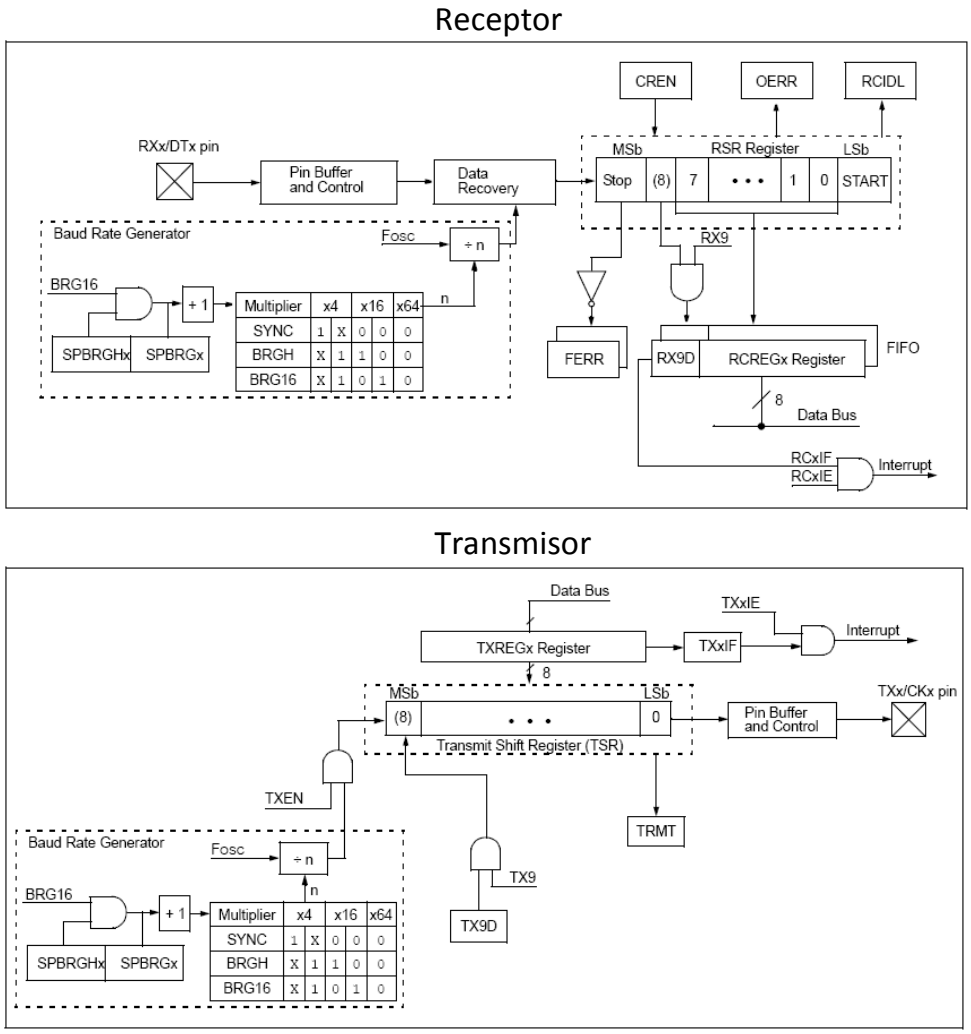


Figura 2.6: Diagramas de Bloques Transmisor y Receptor USART.
Fuente: Tomado de datasheet PIC18F26k22 Microchip®, 2012.

la de recepción de datos seriales, generando funciones para tal fin, una interrupción para la recepción por el USART 1 y otra para el USART2. Estas funciones de recepción de datos almacenan cada carácter (8 bits) recibido del registro RCREG en cada vector con capacidad para 55 bytes que se convierte en el *buffer* de la comunicación. Donde la recepción que viene del ZigBee se toma por el registro RCREG1 y se almacena en RxData1 hasta recibir un "enter" ("0x0D"), lo cual considera que el *buffer* ya tiene la totalidad de los datos recibidos activando una bandera y deshabilitando la recepción de la USART 1. La información que viene del módulo GSM tiene un comportamiento similar donde lo que proviene del RCREG2 se lleva al vector RxData2, el cual incrementa su posición por cada byte recibido hasta un "enter" o "\r".

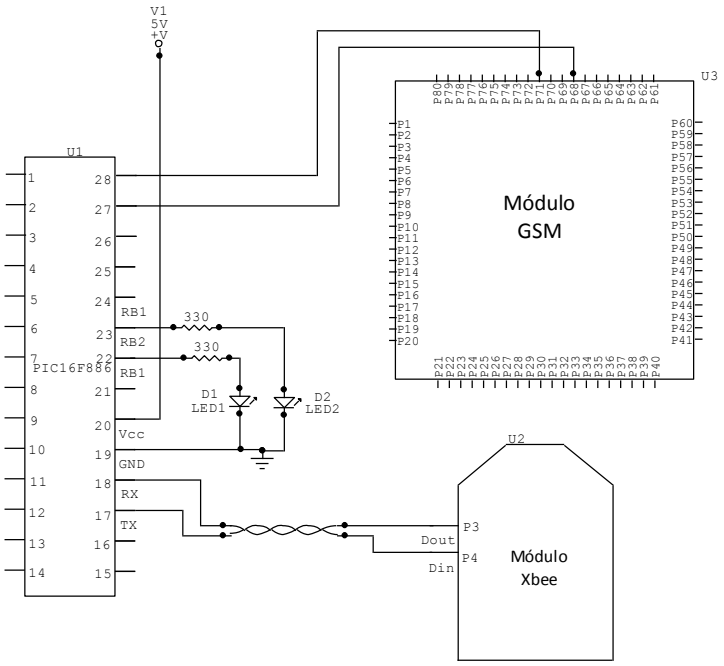


Figura 2.7: Diagrama circuital de conexión módulos GSM-ZigBee con el microcontrolador mediante UART.

Fuente: Elaboración propia.

Igualmente se tienen las funciones de transmisión de cadena de datos, las cuales colocan byte a byte en los registros TXREG1 o TXREG2 los datos a enviar y finaliza cuando no existan mas datos en la cadenas es decir, se tenga un dato vacío ("\0"). Adicionalmente se tiene un contador que determina la posición del dato almacenado en el *buffer* y se incrementa por cada byte nuevo entrante o saliente.

Una vez se tienen creadas las funciones de recepción y transmisión de datos, se configura el microcontrolador, especificando los puertos de entrada y salida de datos, se deshabilitan las entradas análogas junto con las salidas mantenidas (*latch*) se configura la velocidad de comunicación del puerto serial (9600 bps), así como la habilitación del módulo de comunicación USART (SPEN=1) del RCSTA1 y RCSTA2 con sus respectivos bloques de transmisión (TXEN=1 de cada RCSTA) y recepción de datos (CREN=1). Como se tiene una velocidad definida, la comunicación es asíncrona (SYNC=0), para lo cual se define un valor para el SPBRG según tabla del fabricante, que se determina de acuerdo a la velocidad de reloj (16MHz) y los valores de los bits BRGH y BRG16 que se establecen en 1.

Luego de configurar los puertos seriales, se inicia con la comunicación con el módulo Zigbee, donde se manda el comando "intro (enter) " ("\r") para que este responda con el menú de configuración inicial, donde desde el microcontrolador se le envía la letra "B" para que trabaje en modo transparente (*Bypass mode*).

Seguidamente se hace la conexión con el módulo GSM, al cual se le envía el comando "AT\r" y si la respuesta es "OK" se continua con la configuración del mismo, sino se repite el envío del comando "AT" hasta que su respuesta sea valida. Cuando la comunicación se ha establecido correctamente con el módulo GSM, se coloca este en modo texto para los mensajes SMS ("AT+CMGF=1 \r"), se borra el mensaje almacenado en la posición 1 ("AT+CMGD=1\r") y se configura para que todo mensaje nuevo que llegue sea colocado inmediatamente en el puerto ("AT+CNMI=2,2,0,0,0\r").

Finalmente se entra al ciclo infinito con el cual se lee constantemente el valor de la bandera de "recepción completa" en el módulo ZigBee y se verifica que el vector RxData2 no esté vacío, el cual se llena con los mensaje de texto recibidos del módulo GSM. Si se dá la bandera por recepción ZigBee, lo que se hace es mandar un mensaje de texto al celular almacenado con la información recibida del módulo (AT+CMGS="número"), y si hay mensaje en el *buffer* RxData2, lo que se hace es tomar este y reenviarlo al módulo ZigBee cambiando el puerto USART de comunicación. Luego se retorna a la comunicación con el módulo GSM se borra este último mensaje recibido ("AT+CMGD=1\r"), para continuar en este ciclo infinito. Los *buffers* de datos RxData1 y RxData2 son constantemente borrados, para que estos no queden con datos basura ni envíen datos repetidos o ya recibidos, sino que quede concretamente con los datos del mensaje a enviar al módulo ZigBee o con los datos recibidos del otro módulos ZigBee.

Esta configuración del microcontrolador PIC18F26k22, se construye mediante el código en C, desarrollado bajo el software MPLAB X con el compilador XC8 de Microchip (*MPLAB® XC Compilers - Compilers | Microchip Technology Inc.*, 2012). El programa se describe en el diagrama de flujo de la Figura 2.8, el cual utiliza los vectores de recepción y da un descripción del funcionamiento que establece la comunicación serial entre el módulo ZigBee y el módulo GSM.

2.4. Descripción del Asistente Digital Personal

En los últimos años la integración del sector productivo con las comunicaciones, ya tiene aplicaciones a nivel mundial. Las empresas como Kontron, Belsati, Siberline, Ecom, Trimble y Lanpoint han desarrollado y diseñado PDAs exclusivas para procesos industriales teniendo productos como bluetooth industriales o dispositivos de escaneo especiales que han sacado al mercado entre el 2007 y el presente año (*Scanning Devices Inc. | Lexington, MA 02420*, 2009). Igualmente se han utilizado conexiones mediante bluetooth y la red celular para realizar procesos de facturación de abonados eléctricos (Salazar Jairo, 2008) y monitoreo de sistemas de conducción de aguas vía GSM en redes de Empresas Publicas de Medellín.

Una PDA, es la tecnología de un computador al alcance de la mano, puesto que con sus pequeños componentes, permite brindar conexión inalámbrica, instalar software o aplicativos para pruebas, trabajan sobre un sistema operativo, permite

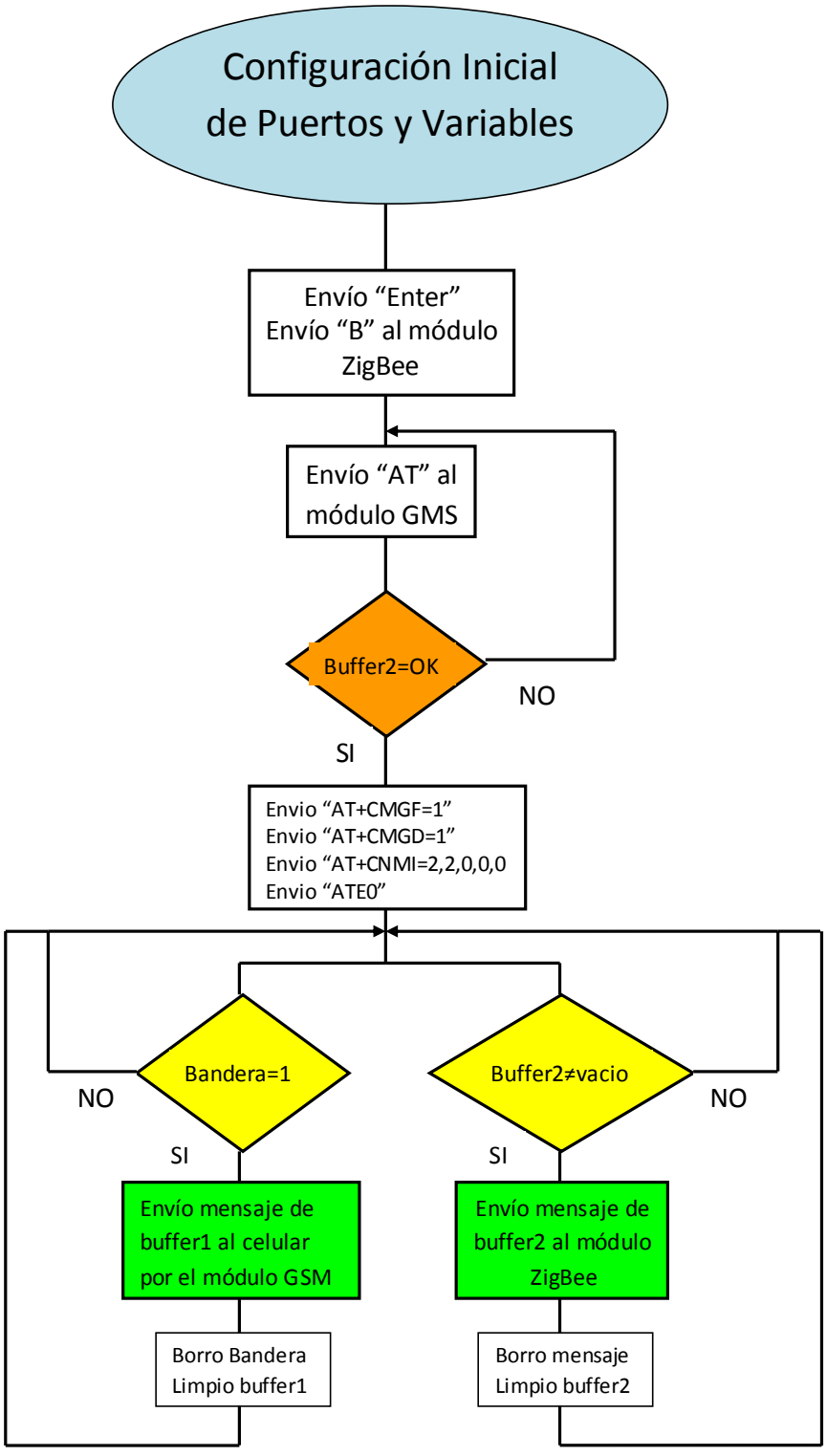


Figura 2.8: Diagramas de Flujo Programa de Configuración del Microcontrolador.
Fuente: Elaboración propia.

programar interfaces, almacenar información, programar fechas, llevar libro de direcciones, en otras palabras es una agenda electrónica (Sp y Williams, 2003).

Estos computadores de bolsillo actuales, poseen una pantalla *touch screen*, lo que permite manipular fácilmente la información de manera táctil y diagnosticar de manera rápida la calidad de información recibida. Además con el teléfono móvil incorporado, se pueden realizar y recibir llamadas permitiendo generar alarmas (Al-Ubaydli y Paton, 2005).

Con la PDA o algún Teléfono Inteligente, es que se plantea la conexión con el PLC a través de ZigBee utilizando la red GSM. Gracias a la evolución de las telecomunicaciones y la integración de estas, se llega al diagnóstico en línea y verificación de funcionamiento del sistema de manera remota en tiempo real. Además se ven involucradas diferentes áreas tanto en el mantenimiento como en las telecomunicaciones y sistemas de monitoreo y control, formando un proceso con aplicaciones a nivel industrial, informático y domótico.

La PDA inicialmente planteada fue un HTC Cruise con sistema operativo Windows Mobile 6.0, el cual con mensajes de texto y alarmas determinadas y si se deseaba un programa en eVB (eMbedded Visual Basic) (R.-C. Wu y cols., 2007) se pueden mandar mensajes especiales a el PLC con los módulos ZigBee. Sin embargo, teniendo en cuenta la evolución de estas y la gran expansión y apertura del sistema operativo Android, se opta por trabajar con equipos de estas características, por su flexibilidad y transversalidad para configurar los eventos y alarmas dados en una línea de producción.

El dispositivo móvil en esta fase, se convierte en el equipo final, el cual puede recibir directamente alarmas de fallas enviadas desde el microcontrolador, siendo un repetidor de la red ZigBee con el módulo GSM. Lo que permite movilidad al ingeniero de planta al realizar monitoreo sin importar el lugar donde se encuentre. La conexión de estas interfaces que se realizan sobre la red GSM, se puede hacer con mensajes SMS como con datos GPRS y abriendo la posibilidad de realizar un aplicativo .apk para el teléfono celular con mayores funciones.

El hardware del Asistente Digital Personal o Teléfono Inteligente no tiene que ser muy poderoso, puesto que la transferencia de datos es mínima, ya que inclusive la máxima velocidad manejada sobre ZigBee es de tan solo 250kbps, además de las mismas configuraciones basadas en la conexión del microcontrolador con cada uno de los módulos. Con esta conexión se hace una integración enfocada a la informática industrial, que realiza teleoperaciones en ambientes dinámicos (Kikuchi y cols., 1998). Adicionalmente uno de los aspectos más importantes para garantizar una buena calidad del servicio en la transmisión de datos de manera inalámbrica, es usando los criterios de QoS (*Quality of Service*) que se implementan con diferentes protocolos, permitiendo flexibilidad de soporte de estos con los dispositivos móviles y soluciones estándares mediante UPnP(*Universal Plug and Play*) (Park y cols., 2007).

Teniendo en cuenta las características de la PDA, se establece la forma como se deben presentar las alarmas de las fallas diagnosticadas en el PLC, tomando

unas alarmas para mantenimiento correctivo y otras para mantenimiento preventivo. Igualmente es posible desde la PDA evaluar la eficiencia del proceso mediante mensajes de diagnóstico en tiempo real enviados desde la PDA, teniendo en cuenta las fallas más comunes que se presentan en los PLCs. Con estos datos, se almacenan los tipos de mensajes a enviar desde el microcontrolador a la PDA, creando mensajes específicos que permiten identificar y procesar las alarmas de una manera correcta. Las transmisiones especificas dadas desde la PDA para control se explicarán con mayor detalle en el capitulo 4, mostrando los mensajes determinados que son enviados por el microcontrolador según sea cada alarma o falla dada, lo que permite definir la metodología de interconexión y diagnóstico de las principales fallas del PLC mediante el Asistente Personal Digital.

2.5. Diseño del Experimento

Para la prueba de esta fase de comunicación, se realiza un conjunto de experimentos que permiten determinar la calidad del proceso de transmisión de mensajes, además de permitir identificar los puntos críticos de funcionamiento y las limitaciones del mismo sistema. Para este diseño se realizan 4 pruebas fundamentales que son:

i) Verificación de Funcionamiento teniendo en cuenta los mensajes enviados y recibidos.

ii) Comprobación de la saturación del *buffer* del microcontrolador ante mensajes largos.

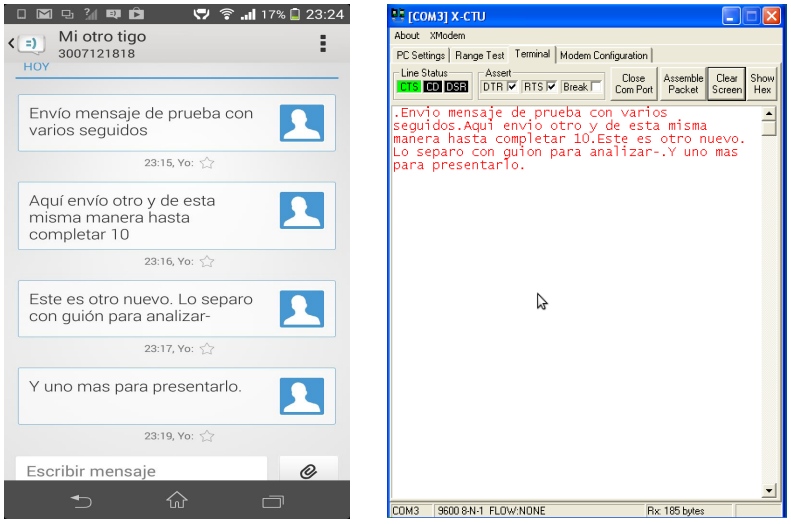
iii) Prueba de simultaneidad y alternancia.

iv) Chequeo de caracteres especiales.

2.5.1. Verificación de Funcionamiento

Para realizar el chequeo del envío de mensajes y mirar el porcentaje de mensajes recibidos, conforme a los enviados se realiza una prueba de envío de mensajes seguidos de manera continua tanto desde la PDA hacia el módulo ZigBee remoto conectado al PC, así como de manera inversa.

Inicialmente se mandaron mensajes desde la PDA o dispositivo móvil al módulo GSM, el cual los recibió correctamente y los reenvió al módulo ZigBee el cual se encontraba conectado a un PC con el cual se verificaba la conexión. Se mandaron cinco mensajes seguidos cortos y todos eran retransmitidos del módulo GSM al ZigBee conectado a este por medio del microcontrolador y de este ZigBee se recibían de manera correcta en el ZigBee remoto conectado al PC como se observa en la Figura 2.9.



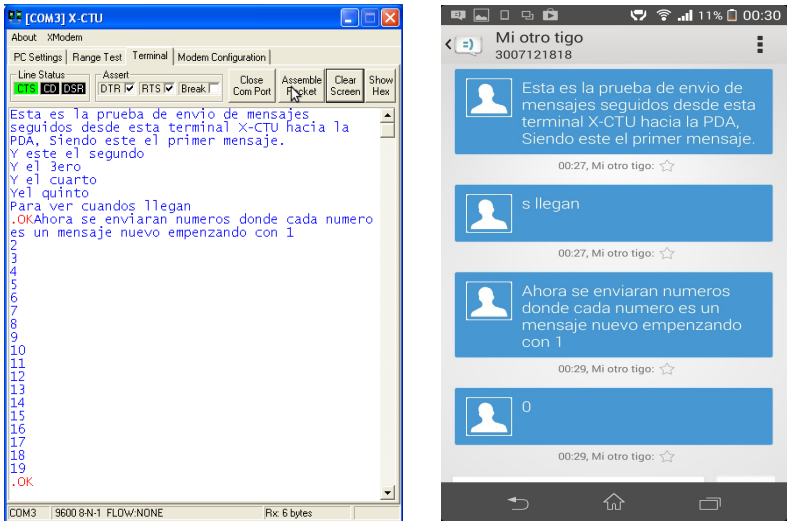
a) Pantalla del Celular b) Pantalla del PC

Figura 2.9: Envío y recepción de mensajes seguidos cortos desde la PDA al terminal XCTU del módulo ZigBee en el PC.
Fuente: Elaboración propia.

También se enviaron mensajes cortos y seguidos desde la terminal del módulo ZigBee a la PDA pasando por el microcontrolador y de este al módulo ZigBee. Ante esta situación existe un retardo que no permite el envío continuo de todos los mensajes del módulo GSM, puesto que el proceso de envío de mensajes evita la retransmisión en ese mismo instante de tiempo de otro mensaje simultaneo. Esto se debe a que en el programa del microcontrolador una vez se tenga el *buffer* lleno, este interrumpe la recepción, envía el número telefónico al cual se le va a enviar el SMS, luego carga el mensaje y luego se le dá enviar, este proceso se visualiza en los mensajes recibidos conforme a los enviados del PC en la Figura 2.10. Sin embargo este retardo no es un inconveniente para el diagnóstico de fallas del PLC, pues el microcontrolador que está conectado al PLC realiza un procesamiento previo de las tramas de detección de errores antes de enviarlas al módulo ZigBee como se verá en el capitulo 4 de este trabajo. El tiempo para enviar 2 mensajes seguidos deberá ser superior a los 3,23 segundos, garantizando de esta manera la correcta recepción de mensajes en la PDA.

2.5.2. Comprobación Buffer

Teniendo en cuenta que el *buffer* donde se almacenan los datos en el microcontrolador, tanto de la comunicación con el módulo GSM como con el módulo ZigBee, tiene capacidad para 255 caracteres, se realizaron pruebas de envío de mensajes superando el estándar de capacidad de caracteres para SMS (160 caracteres como máximo por mensaje), de esta forma el envío se realiza con dos



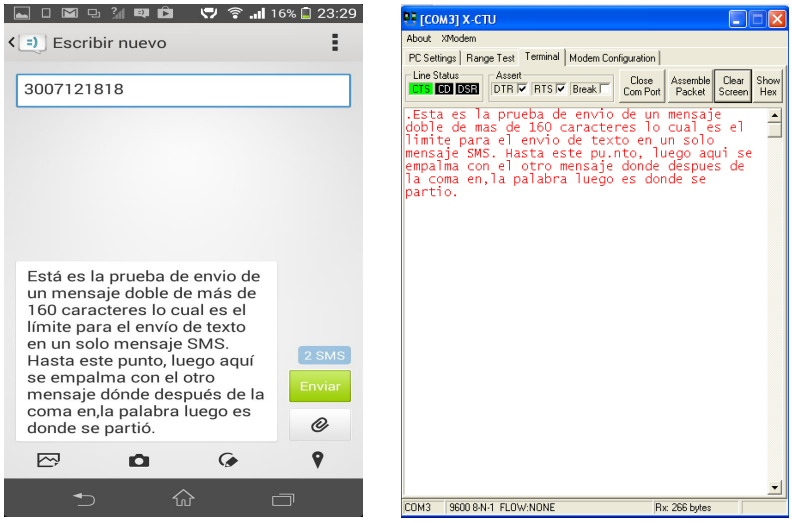
a) Pantalla del PC b) Pantalla del Celular

Figura 2.10: Envío de mensajes seguidos desde el PC a la PDA con perdida de información por tiempo de procesamiento.
Fuente: Elaboración propia.

mensajes seguidos y se verifica los alcances del mismo, teniendo en cuenta que las acciones implementadas entre el PLC y la PDA son para control y diagnóstico de fallas cuyos caracteres manejan información menor a 160.

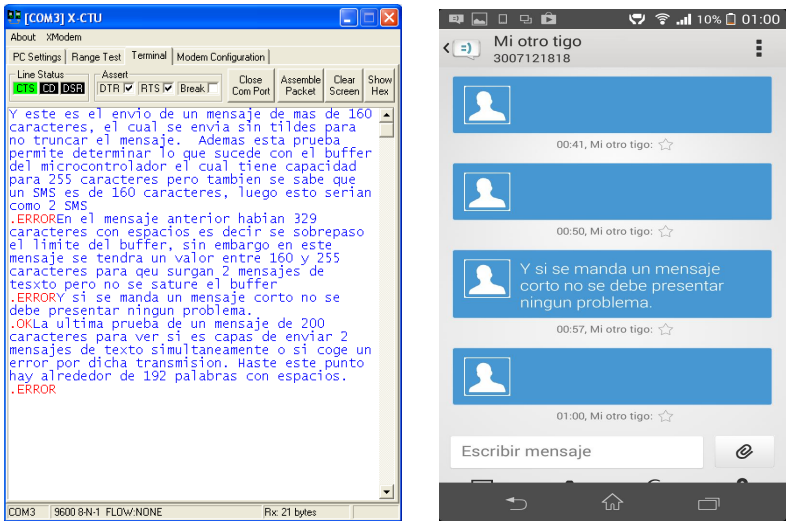
Al realizar el envío de mensajes largos SMS, superando el limite determinado a 160 incluyendo espacios, el mensaje de texto queda convertido en un mensaje doble, es decir, dos SMS, donde al realizar esta prueba ambos mensajes fueron reenviados al módulo ZigBee remoto sin perder información como se aprecia en la Figura 2.11. No se presenta ningún inconveniente, debido a que cuando el dispositivo envía estos, el los separa y llegan en dos instantes de tiempo diferentes al módulo GSM, luego cuando el microcontrolador recibe cada mensaje lo reenvía al módulo ZigBee y borra el *buffer*, cuyo tiempo es suficiente para recibir el otro mensaje correctamente.

Sin embargo, al realizar el proceso inverso desde el módulo ZigBee conectado al PC enviando mensajes a la PDA, se presentó un inconveniente con los mensajes de más de 160 caracteres, que aunque el buffer del microcontrolador tiene capacidad para hasta 255 bytes, el mensaje al llegar al módulo GSM saca error, debido a que cuando se carga el mensaje por comandos AT desde el microcontrolador, el módulo GSM detecta el exceso de caracteres evitando que el mensaje sea enviado. Estas pruebas se pueden ver con más claridad en la Figura 2.12.



a) Pantalla del Celular b) Pantalla del PC

Figura 2.11: Envío y recepción de mensajes dobles desde la PDA al ZigBee remoto.
Fuente: Elaboración propia.



a) Pantalla del PC b) Pantalla del Celular

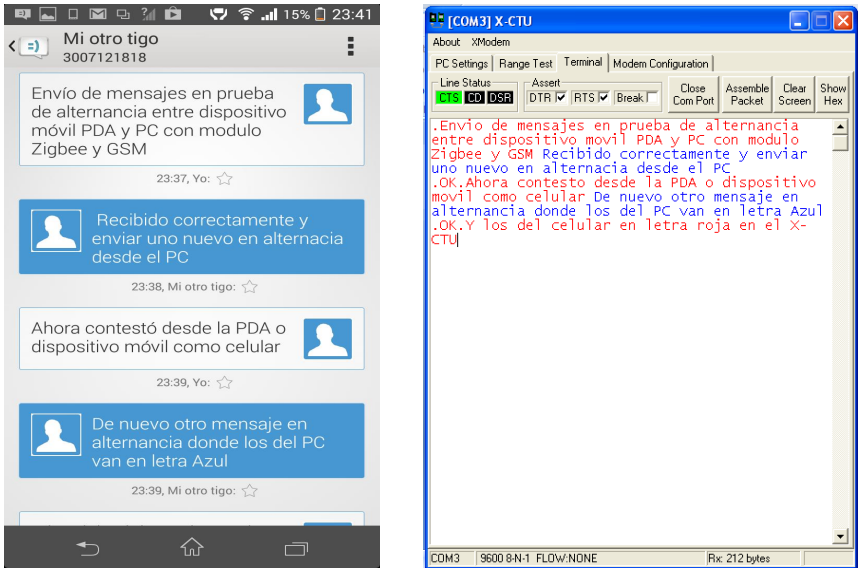
Figura 2.12: Problema de envío de mensajes desde PC a PDA con más de 160 caracteres.

Fuente: Elaboración propia.

2.5.3. Prueba de Simultaneidad y Alternancia

Teniendo en cuenta que el microcontrolador utilizado posee 2 puertos USART y que en un puerto está conectado el módulo GSM que establece la conexión con la PDA y en el otro puerto se encuentra el módulo ZigBee que se conecta de manera remota con el otro módulo ZigBee conectado al PC, se establecen condiciones de simultaneidad y alternancia para probar el comportamiento de manejo de interrupciones e información por parte del microcontrolador ante información recibida por cada puerto.

Para estas pruebas, inicialmente se realiza el envío de a un mensaje alternando entre el dispositivo móvil y el módulo ZigBee, donde se logra el envío de información en ambos sentidos, donde desde la PDA va al módulo GSM y el microcontrolador lo reenvía al ZigBee y cuando se escribe un mensaje en la terminal ZigBee y se dá la tecla de "Entrada" el microcontrolador toma estos caracteres de manera correcta y este los procesa y le indica al módulo GSM que envíe un mensaje de texto con el número telefónico indicado y la información dada. Para este caso se realizó una prueba de cinco mensajes desde cada dispositivo de manera alternada y no se presentó ningún inconveniente, lo cual se visualiza en la Figura 2.13. En la subfigura a) se observa en blanco los mensajes enviados desde la PDA y en azul los enviados desde el PC al módulo ZigBee. En la Figura 2.13 b) en rojo son los mensajes recibidos desde la PDA en el computador y en azul los mensajes enviados desde el PC a la PDA remota.

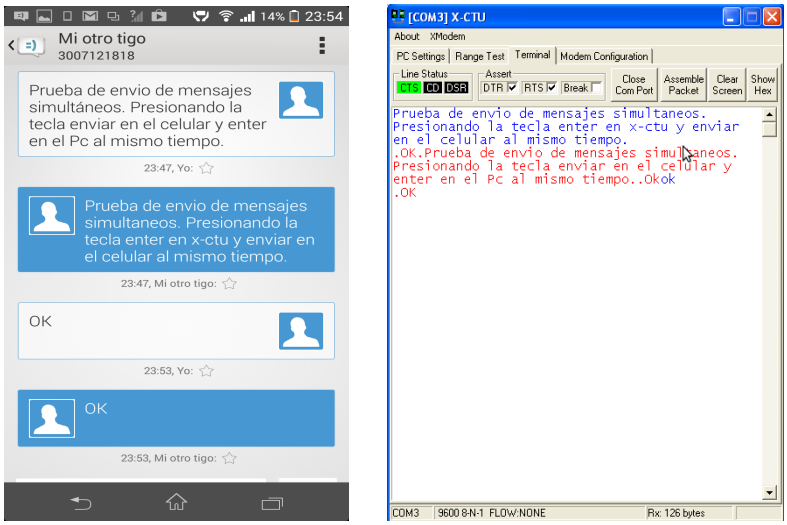


a) Pantalla del Celular b) Pantalla del PC

Figura 2.13: Envío y recepción de mensajes alternados desde la PDA al ZigBee remoto y viceversa.

Fuente: Elaboración propia.

Se realizó también el envío de mensajes simultáneos, presionando el botón "enviar " desde la PDA al mismo tiempo que la tecla "enter" desde la terminal del módulo ZigBee y ambos mensajes llegaron con la información correcta sin problemas de combinación de información (Figura 2.14.), a causa de que la información se almacena en *buffers* de datos diferentes y los puertos del microcontrolador operan de manera independiente, lo que evita interferencias.



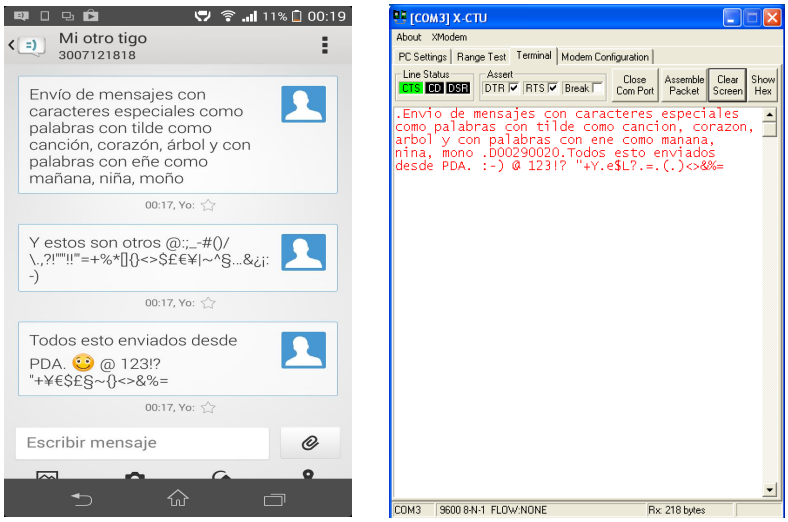
a) Pantalla del Celular b) Pantalla del PC

Figura 2.14: Envío y recepción de mensajes simultáneamente desde el PC y PDA.
Fuente: Elaboración propia.

2.5.4. Caracteres Especiales

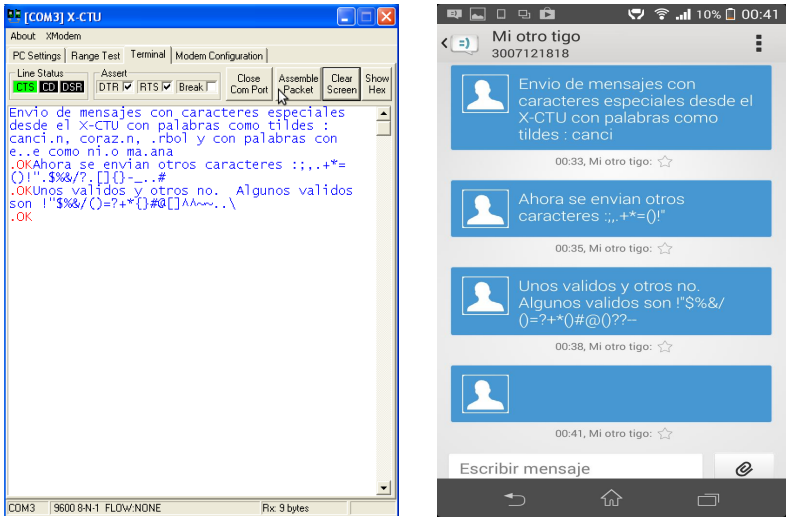
Con el objetivo de estudiar los parámetros que se pueden enviar en este tipo de comunicación y delimitar el uso del mismo ante diferentes entradas, se realizaron pruebas de envío de mensajes con caracteres especiales algunos derivados del español, así como símbolos y nomenclaturas dadas por las interfaces de teclado de los dispositivos.

Los caracteres especiales que se utilizaron para estas pruebas fueron las vocales tildadas y la "ñ", los cuales son enviados desde la PDA al módulo ZigBee, sin embargo cuando llegan a este, estos no son interpretados y coloca la palabra como es pero sin la tilde como se observa en la Figura 2.15. De la misma manera se realizan pruebas desde el módulo ZigBee a la PDA y cuando se envían tildes en el mensaje este queda truncado o cortado en este punto y el mensaje recibido en la PDA llega hasta la letra que tiene el carácter especial ver Figura 2.16.



a) Pantalla del Celular b) Pantalla del PC

Figura 2.15: Envío de mensajes con caracteres especiales PDA a PC.
Fuente: Elaboración propia.



a) Pantalla del PC b) Pantalla del Celular

Figura 2.16: Envío de mensajes con caracteres especiales PC a PDA.
Fuente: Elaboración propia.

2.5.5. Diseño de experimento Completo

El diseño de experimento que une el conjunto de mensajes con las diferentes pruebas realizadas, se aprecia en las Figura 2.17 donde las letras en azul corresponden a los mensajes enviados desde el módulo ZigBee a la PDA, observando los

problemas con las tildes, "ñ" y aparece error con el mensaje largo. Luego aparecen los mensajes en rojo que van desde la PDA al módulo ZigBee apareciendo un punto en la palabra tilde, el cual corresponde al envío de un mensaje doble, que indica la unión de los dos mensajes SMS y en las tildes y "ñ", se entrega el carácter sin esta puntuación. Seguidamente se mandan caracteres extraños los cuales aparecen un poco cambiados desde la PDA y si son símbolos especiales truncan el mensaje. Sin embargo, si se mandan caracteres especiales desde la PDA al módulo ZigBee, este aparece tan extraño como solo números.

En la PDA los mensajes con fondo azul, corresponden a los recibidos desde el módulo ZigBee y los de fondo blanco, son los enviados desde la PDA. En estos mensajes se puede apreciar como se corta el mensaje en la palabra canción, se reciben caracteres especiales sin problema pero en el envío de los mismos aunque salen bien se reciben mal como se aprecia en la Figura 2.18.

Sin embargo después de realizar el respectivo diseño del experimento, se determina que aunque se presentan algunos fallas en el envío de mensajes continuos desde el módulo ZigBee y se presentan inconvenientes tanto en la transmisión como recepción de caracteres especiales, estos no representan problemas de funcionamiento en la interfaz de comunicación, puesto que para el diagnóstico de fallas y los mensajes de control, estos son programados de tal forma que brinden una buena comunicación con el sistema siendo fluida y transparente para el usuario.

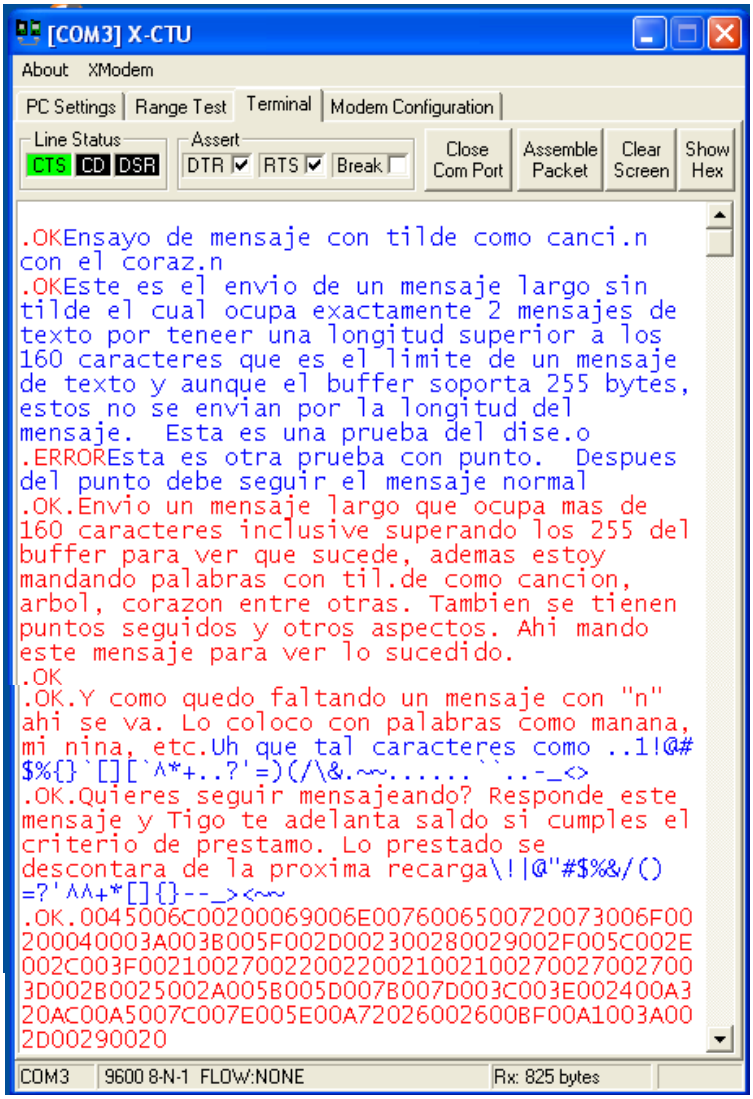


Figura 2.17: Envío y recepción de mensajes desde la terminal XCTU del módulo ZigBee.

Fuente: Elaboración propia.

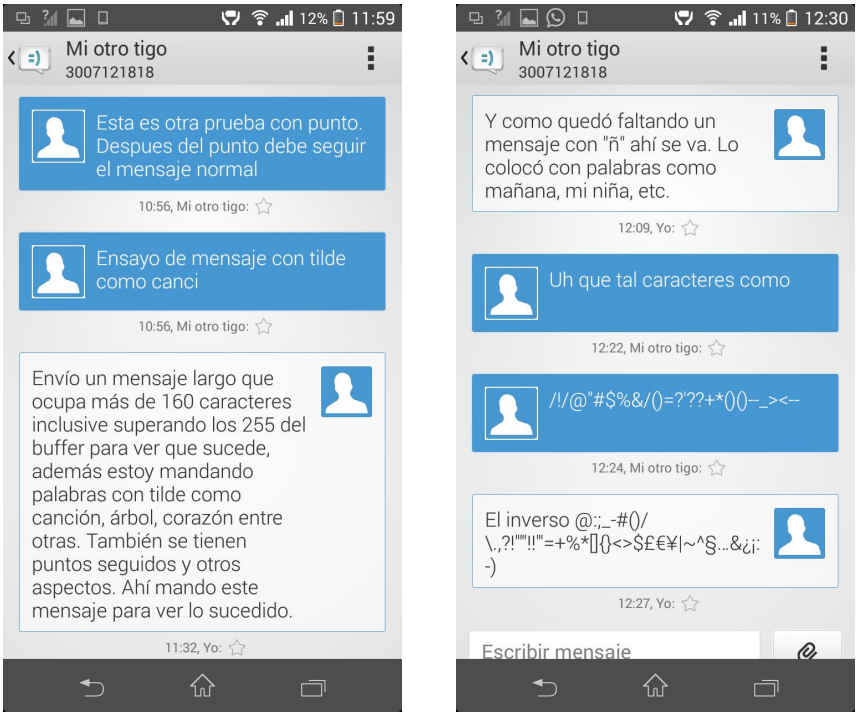


Figura 2.18: Envío y recepción de mensajes PDA.
Fuente: Elaboración propia.

Capítulo 3

Comunicación PLC-Microcontrolador via Profibus

Al definir el protocolo de comunicación que establece el enlace de comunicación entre el PLC y el microcontrolador, se busca tener control y manejo de la información de forma transparente y confiable, por lo que se toma como la mejor opción el protocolo **Profibus**. Con este protocolo se puede realizar un control,visualización y cambio de parámetros, además de la detección de fallas básicas del PLC. El enlace también se puede realizar mediante una red LAN con Ethernet o vía Wi-Fi, sin embargo, con Profibus se sigue el estándar RS485 y no se necesitan módulos adicionales para su respectiva conexión. Figura 3.1.

Al realizar este diagnóstico se verifica el comportamiento del PLC ante eventos fortuitos determinando las posibles fallas más comunes de estos y se evalúa la eficiencia del perfil de comunicaciones entre PLC y el microcontrolador. Para esto se realiza un estudio y análisis de la tramas Profibus que al usar el microcontrolador como esclavo mediante el acople de un Gateway garantiza una comunicación confiable en tiempo real. Con este acople se optimizan los tiempos de reacción y funcionamiento de la red industrial, puesto que la detección de errores es precisa y concreta, mejorando la productividad y eficiencia de una empresa (González Daniela, 2009).

Para que la comunicación Profibus sea efectiva, se debe tener en cuenta el proceso de negociación y los parámetros con los que funciona este protocolo con sus respectivas tramas (Ver Capitulo 1), pues es lo que se procesa en el microcontrolador para que pueda comportarse como un esclavo.

En este capitulo, se plantea la solución de comunicación dada entre el PLC y el microcontrolador a través de la red Profibus, utilizando un circuito que hace las veces de Gateway en la capa física del estándar RS485 en el protocolo Profibus. Con esto se permite desarrollar una estructura metodológica para el diagnóstico de

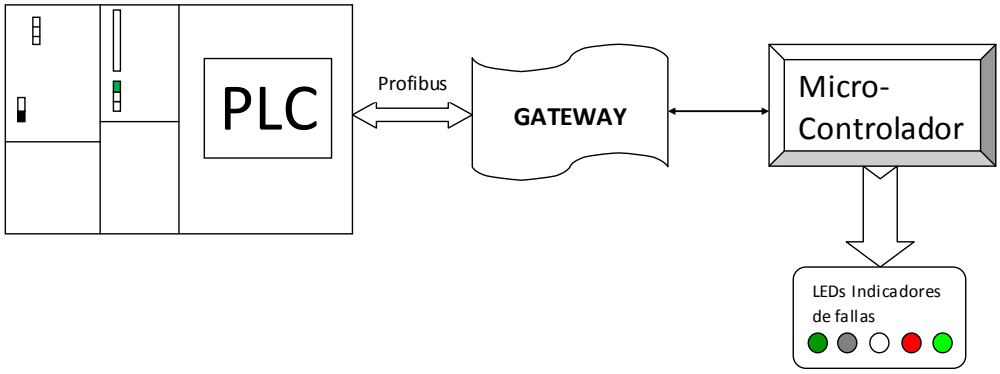


Figura 3.1: Descripción Implementación de la conexión del PLC con el microcontrolador para la detección de fallas
Fuente: Elaboración propia.

fallas de un PLC, que es lo planteado en esta fase 2, buscando cumplir con uno de los objetivos específicos dados. Para abordar de una mejor manera este capitulo, se inicia con las características, configuración y conexión del microcontrolador, luego se plantea el Gateway propuesto (Circuito Análogo con amplificadores operacionales) como solución efectiva de comunicación y finalmente se habla del acople con el PLC para la detección de fallas según las tramas presentadas.

3.1. Microcontrolador

Para esta fase de comunicación, teniendo en cuenta que se necesitan dos vías de comunicación, una para el PLC y otra para el módulo ZigBee, se opta por utilizar también un microcontrolador con 2 puertos USART, permitiendo un manejo más simple de la información y logrando una comunicación confiable sin retardos. El microcontrolador seleccionado debe tener una buena velocidad de procesamiento para garantizar la construcción de las tramas Profibus para que sean correctamente interpretadas por el PLC, puesto que esto se convierte en un punto crítico de comunicación.

El PIC18F26K22 al igual que en el capitulo anterior, se utiliza para realizar esta interfase de la fase 2 que aunque es un circuito totalmente independiente, este microcontrolador se caracteriza por cumplir los parámetros básicos necesarios para permitir una comunicación confiable. Este microcontrolador de 8 bits de Microchip tiene las características de hardware ya explicadas en el capitulo anterior que a diferencia de la otra implementación, este se configura con cristal externo y no con el oscilador interno. El cristal de 16MHz con el PLLx4 activado permite que el microcontrolador tenga una velocidad de procesamiento de 64MHz y ejecución de instrucciones de 16 MIPS, lo que permite el correcto análisis de las tramas Profibus enviadas por el PLC maestro.

3.1.1. Conexión y Configuración

Para la conexión del microcontrolador con el PLC, se utiliza uno de los puertos USART de este microcontrolador, con las características descritas en el capitulo anterior. Se usa conexión RS232 simple con niveles TTL, el cual se acopla al bus Profibus del PLC con el Gateway diseñado, seleccionando una velocidad de transmisión de 9600 bps al igual que el Profibus, con 9 bits de datos, de los cuales 8 corresponden el dato y 1 bit para la paridad y adicionalmente un bit de inicio y un bit de parada. Se deshabilitan los otros osciladores de respaldo, el reinicio por bajo voltaje, así como el *watch dog*, protección por código y el *master clear*. Se habilitan las interrupciones globales, así como las de las USART.

En la conexión del microcontrolador con el PLC, se busca engañar al PLC maestro, haciéndolo creer que el dispositivo embebido es un PLC esclavo, para lo cual se tiene en cuenta todo el estudio del estándar y con este se lleva a cabo un análisis de las tramas de comunicación que utiliza el PLC maestro con el PLC esclavo. Para este análisis se utiliza un osciloscopio Tektronik MS04034B de 350MHz que con la llave de decodificación DPO 4COMP, se logra decodificar las tramas RS485 Profibus correctamente. Igualmente se analizan las tramas Profibus del PLC con otros esclavos como variadores de velocidad, módulos Siemens como el ET200, IE/PB Link, Pantalla TP 177B y DP/PA coupler. Sin embargo sabiendo que a través del PLC Siemens se puede tener mayor control y mejor información por ser este un esclavo inteligente, se opta por simular al microcontrolador como un PLC esclavo y analizar el proceso de negociación y las tramas Profibus entre PLC maestro y PLC esclavo.

Con los tipos de tramas dadas entre el PLC maestro y el PLC esclavo, estas tramas son grabadas en el microcontrolador como vectores, para que de esta manera se pueda realizar el proceso de negociación con el PLC maestro mediante suplantación de identidad manejando la misma información.

Con un correcto acople de impedancias y trabajando los voltajes ideales para este proceso de negociación, el microcontrolador recibe la trama completa de datos enviada por el PLC maestro, teniendo en cuenta que todas las tramas básicas que este envía terminan en 0x16 (16h) y que solo existe una trama de token que es de 3 bytes empezando por 0xDC (DCh) y seguida de la dirección del maestro que cuando hay uno solo, los 2 últimos bytes son iguales. Igualmente se construyen vectores de paridad para cada una de las tramas, garantizando los tiempos de transmisión en las tramas y los bytes (byte a byte), para enviar cada byte de la trama con un valor de paridad que se carga en el bit TX9D, los cuales se concatenan puesto que los microcontroladores PIC no traen por defecto el byte de paridad para definirlo en la comunicación USART.

También se almacena en la memoria del microcontrolador, las tramas de respuesta que debe dar este ante cada trama enviada por el PLC, la cual también debe ser almacenada para su respectiva comparación y dar la respuesta acertada según sea la trama enviada por el dispositivo lógico programable Figura 3.2. El

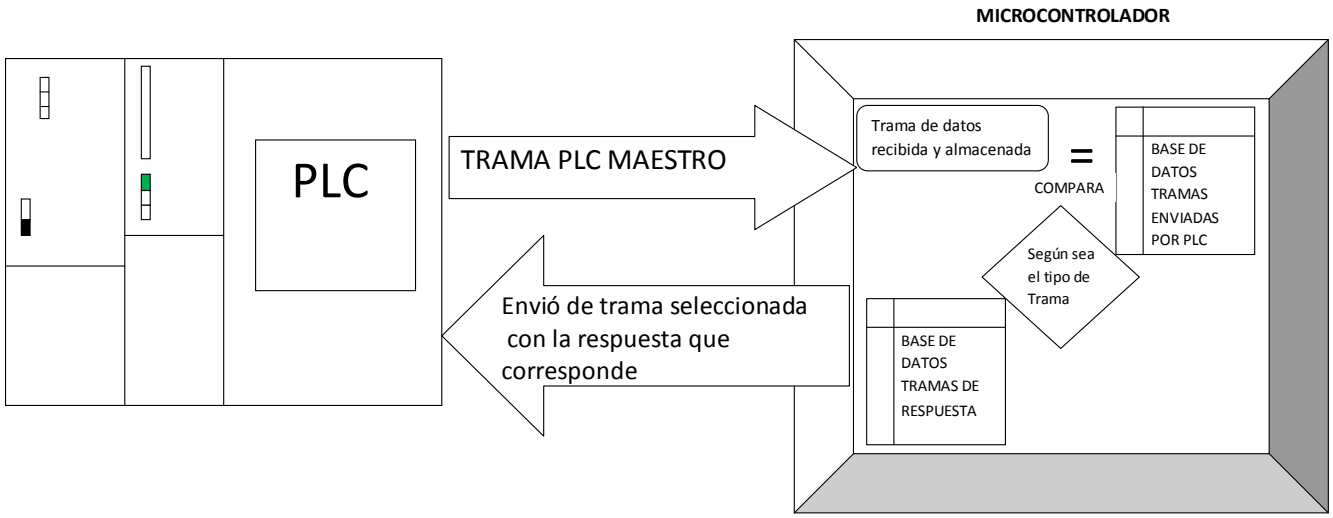


Figura 3.2: Comparación para el proceso de negociación entre el PLC y el microcontrolador.

Fuente: Elaboración propia.

proceso de comparación entre tramas, se realiza mediante la operación lógica binaria AND byte a byte y si hay coincidencia de todos los bytes dentro de una trama, el número de estas coincidirá con la variable que identifica el tipo de trama.

El puerto de comunicación utilizado para la conexión con el PLC es el 1, habilitando el bit RX9 y TX9D para la paridad en recepción y transmisión. Se tiene una declaración especial de variables enteras, vectores tipo char y funciones. Se inicia el programa con la parametrización y configuración básica del microcontrolador, donde se declaran las salidas y entradas, se deshabilitan los puertos de conversión análoga/digital al igual que las salidas aseguradas (Latch), limpieza de vectores buffer, variables de comparación y configuración del puerto serie al igual que del oscilador.

En el ciclo infinito del programa, el microcontrolador se queda esperando la recepción de tramas mediante la rutina de atención por interrupción de recepción. Donde una vez se recibe la trama que envía el PLC, este entra a la interrupción por recepción donde almacena cada byte enviado por el PLC maestro en un buffer de recepción y se detiene cuando dicho buffer recibe un 16h (0x16), debido a que este es el indicador de final de trama Profibus SD1, SD2 y SD3. En esta recepción no se analiza la trama SD4, debido a que esta es de token para recepción con más maestros dentro de la red y debido a que el microcontrolador trabaja como esclavo, no hace falta este análisis. Una vez se recibe la trama completa se activa una bandera y se compara byte a byte con la AND verificando que tipo de trama del proceso de negociación es esta. En la base del programa se chequean los valores de las variables de comparación, los cuales determinan el tipo de trama recibida y

una vez se tiene la trama se da la respuesta con el tipo de trama correspondiente y la paridad de cada byte con el vector de paridades previamente almacenado, de esta manera para cada respuesta guardada como se observa en el diagrama de flujo de la Figura 3.3.

Luego una vez se da la respectiva respuesta se borran las variables de comparación, los buffer, contadores y la bandera en espera de una nueva recepción de otra trama enviada por el PLC maestro. Para evitar interferencias sobre el bus RS485, en el momento que se recibe una trama, se presta atención constantemente a la interrupción de los bytes recibidos hasta el final de la trama, luego se deshabilita momentáneamente el receptor para el análisis de la trama y el envío de la respuesta correcta y una vez se ha enviado esta respuesta, se habilita de nuevo el receptor USART de microcontrolador.

Con esto se logra que el microcontrolador opere como un PLC Siemens esclavo, pues no solo se tiene en cuenta las tramas enviadas según las recibidas, sino también los tiempos de tramas, velocidades de transmisión, intervalos entre tramas e intervalos entre bytes. Además para que la comunicación dada entre el PLC Siemens maestro y un microcontrolador actuando como un PLC Siemens esclavo se necesita realizar un correcto y rápido análisis de las señales, por lo que los 64MHz del reloj en el microcontrolador permiten lograr una óptima velocidad de procesamiento en las tramas Profibus enviadas por el PLC maestro, engañando correctamente a este y haciendo que el microcontrolador se vea como otro PLC.

En la programación de este dispositivo embebido se tiene en cuenta la previa programación del PLC maestro, el cual se configura con una dirección 10 de Profibus y una de 5 para el PLC esclavo, con una velocidad de transmisión de 9600bps en la línea Profibus con altos tiempos de espera y de retransmisión.

Terminado el proceso de negociación, el PLC maestro manda tramas continuas de verificación con el esclavo de que el enlace Profibus sigue activo. En estas tramas de verificación, se transporta la información de los bytes y bits que están en 1 y 0 del proceso de comunicación interno entre maestro y el microcontrolador suplantado como esclavo, los cuales son los fundamentales para diagnosticar el tipo de falla que presenta el PLC maestro.

El proceso de verificación de comunicación Profibus en línea maneja siempre las mismas tramas, con el mismo tamaño (11Bytes), que tienen el tipo de trama, longitud de la trama, la dirección del destino, dirección de fuente, función de la trama, checksum; solo cambiando algunos bytes de la unidad de datos (Bytes 8 y 9), los cuales son chequeados constantemente por el microcontrolador. Si no hay transferencia de información, estos bytes permanecen en 00h y luego ante algún cambio, en el PLC maestro, el microcontrolador detecta este cambio lo que permite indicar el tipo de falla del PLC previamente programada. Siempre que el PLC maestro envié una trama, el microcontrolador verifica la información de los bytes 8 y 9 de la trama y si estos son diferentes de 00h, es porque se ha presentado un tipo de falla en el PLC. Las tramas del maestro y esclavo cíclicas de verificación con bytes 00h se pueden apreciar con claridad en la Figura 3.4.

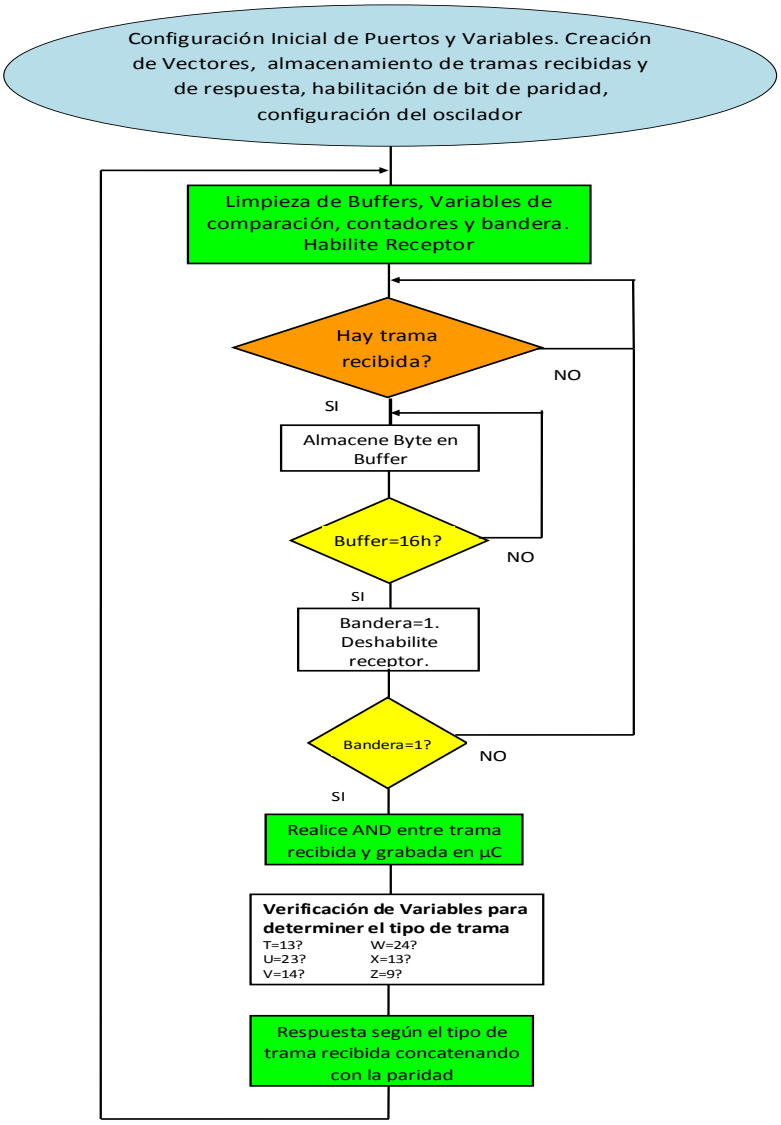


Figura 3.3: Diagrama de flujo del programa en el microcontrolador de respuesta de tramas.

Fuente: Elaboración propia.

Figura 3.4: Tramas Profibus de verificación luego del proceso de negociación.
Fuente: Elaboración propia foto osciloscopio Tektronik MS04034B.

Para transmitir información de control desde el microcontrolador al PLC, esta se introduce en el payload de la trama Profibus de respuesta en el proceso de verificación de conexión, cambiando los mismos bytes 8 y 9 de la trama que envía el esclavo. Donde la acción de control será conforme a la programación previa de los bytes del PLC maestro definidos para tal configuración. En este payload, pueden haber señales de habilitación o deshabilitación de salidas, lectura de variables análogas, información y validación del diagnóstico de fallas o errores del PLC entre otros aspectos de monitoreo y control.

3.2. Gateway Propuesto

El Gateway es el acople que permite realizar una comunicación confiable entre el microcontrolador y el PLC por la red Profibus, permitiendo realizar una apertura de esta red a cualquier medio o dispositivo. Esta interconexión se realiza sobre el bus de comunicación creando un puente entre la interfaz USART del microcontrolador y las líneas A y B del estándar Profibus. Entre las principales características de este Gateway, es que es muy económico, es simple de implementar y lo más importante, es que acopla correctamente los voltajes diferenciales que maneja Profibus en su capa física RS485 con los niveles TTL presentados por

el microcontrolador. Esta unión de las 2 redes con estructuras diferentes permite conexión y desconexión en caliente además de utilizar cualquier tipo de topología en cada red de manera independiente.

Otra característica del Gateway, es que permite ser usado con otros dispositivos de campo, debido a que este tiene un correcto acople de impedancias, permitiendo hacer uso de las redes de sensores (Y. Wu y Shao, 2012) a través de la red Profibus, evitando la conexión directa del sensor con una entrada análoga o digital del PLC. La conexión entre el PLC Siemens usado y el dispositivo embebido es transparente, permitiendo que el microcontrolador funcione como un PLC Siemens esclavo dentro de la red. Este Gateway además de tener su hardware, maneja un pequeño control por software, permitiendo obtener los voltajes diferenciales, activados por un pin del microcontrolador, el cual permite manejar los niveles de *standby* y transmisión de 1 y 0.

3.2.1. Hardware del Gateway

El circuito análogo implementado en el Gateway busca solucionar los problemas presentados con los niveles de voltaje puesto que los PLC Siemens evitan la suplantación de un microcontrolador como un PLC esclavo. Ya que si se conectan 2 PLCs Siemens, uno como maestro y otro como esclavo vía Profibus y se colocan integrados MAX485 en medio de estos, se pueden presentar posibles errores de comunicación. Por esta razón la incompatibilidad entre el mismo fabricante, hace más difícil la implementación.

También este circuito acopla correctamente las impedancias, lo cual es otro factor determinante para la correcta transferencia de información, donde la conexión de un dispositivo embebido con un PLC cambia la impedancia del bus Profibus, lo cual se aprecia como un cambio de voltaje como se vé en la Figura 3.5. A partir de esta verificación experimental, se entiende por que el PLC Siemens no identifica el microcontrolador y detecta este cambio de voltaje como una alteración del bus Profibus. Debido a que un cambio de impedancias implica cambios de voltaje, entonces la solución se dá mediante el aislamiento de impedancias con transistores y amplificadores operacionales, puesto que un transistor y un amplificador operacional permiten realizar un acople de impedancias, haciendo que estos cambios no sean perceptibles para el PLC y no se alteren los voltajes del bus de comunicación.

Lo primero que se realiza en la implementación del Hardware es suprimir el uso del integrado MAX485 en la fase de transmisión de señales desde el microcontrolador hacia el PLC, dado que aunque el estándar Profibus maneja en su capa física la comunicación RS485, las pruebas experimentales determinaron que este presenta inconvenientes para comunicar el PLC con el microcontrolador. El inconveniente se presenta cuando se llevan los niveles lógicos (0 y 1) a voltajes diferenciales dados entre las 2 líneas del bus (Linea A y Linea B del RS485), ya que el bus tiene la particularidad de utilizar valores positivos y negativos, los cuales son diferentes a los tres estados utilizados por el MAX485 ver tablas en figura

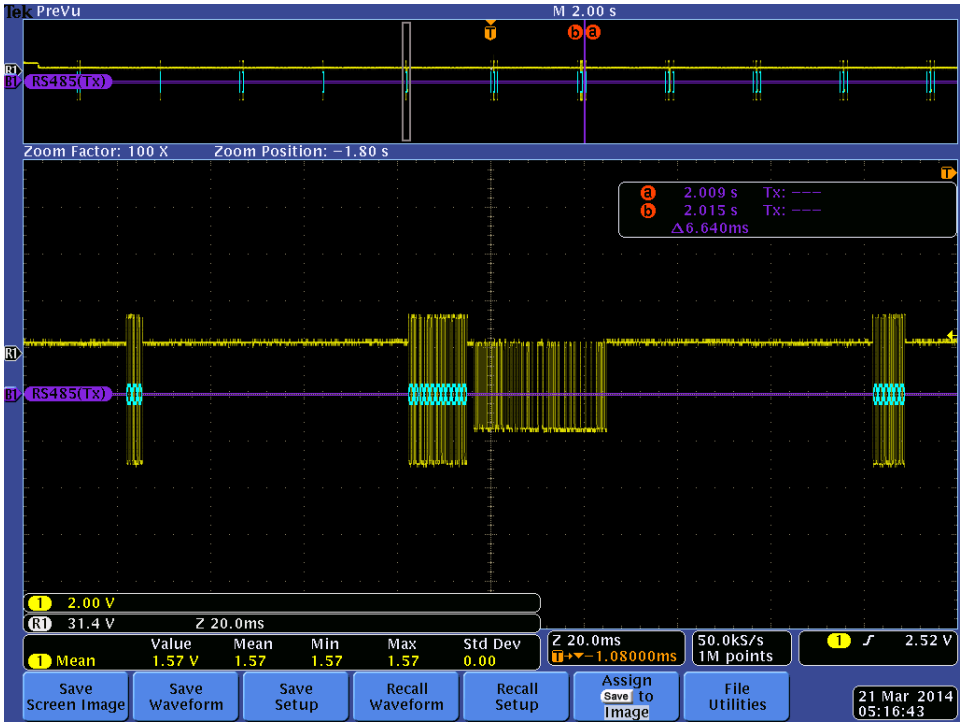


Figura 3.5: Tramas Profibus al conectar microcontrolador.
Fuente: Elaboración propia foto osciloscopio Tektronik MS04034B.

3.6 donde se tiene lo siguiente: "0 1" (Linea A=0 voltios Linea B=5 Voltios), "1 0" (Linea A=5 voltios Linea B=0 Voltios) mostrado en el diagrama del Hardware con MAX484 o DS75176 Figura 3.7 y alta impedancia en ambas líneas, dando como resultado los voltajes mostrados en la Figura 3.8.

Tabla Transmisión

Inputs			Line	Outputs	
$\overline{RE}$	DE	DI	Condition	$\overline{DO}$	DO
X	1	1	No Fault	0	1
X	1	0	No Fault	1	0
X	0	X	X	Z	Z
X	1	X	Fault	Z	Z

Tabla Recepción

Inputs			Outputs
$\overline{RE}$	DE	RI-$\overline{RI}$	RO
0	0	≥ +0.2V	1
0	0	≤ −0.2V	0
0	0	Inputs Open**	1
1	0	X	Z

$\overline{RE}$, DE, DI, RI y $\overline{RI}$
Entradas del MAX485
Z= Alta impedancia
X= Condición no importa
No Fault= no falla
Fault= Falla
Inputs Open= Entada Abierta
Salidas 0 = 0 Voltios 1 = 5 Voltios

Figura 3.6: Tablas niveles y datos MAX485.
Fuente: Tomado Datasheet MAX485-DS75176

Sin embargo en la recepción de señales se tiene un circuito diferente, pues las líneas A y B si se conectan al MAX485, pues la conversión de voltajes diferenciales con valores positivos y negativos a niveles lógicos TTL de 0 y 5 voltios, si son correctamente transformados y entendidos por el microcontrolador. Donde estas líneas del PLC se conectan con el MAX485 utilizando las resistencias de balance

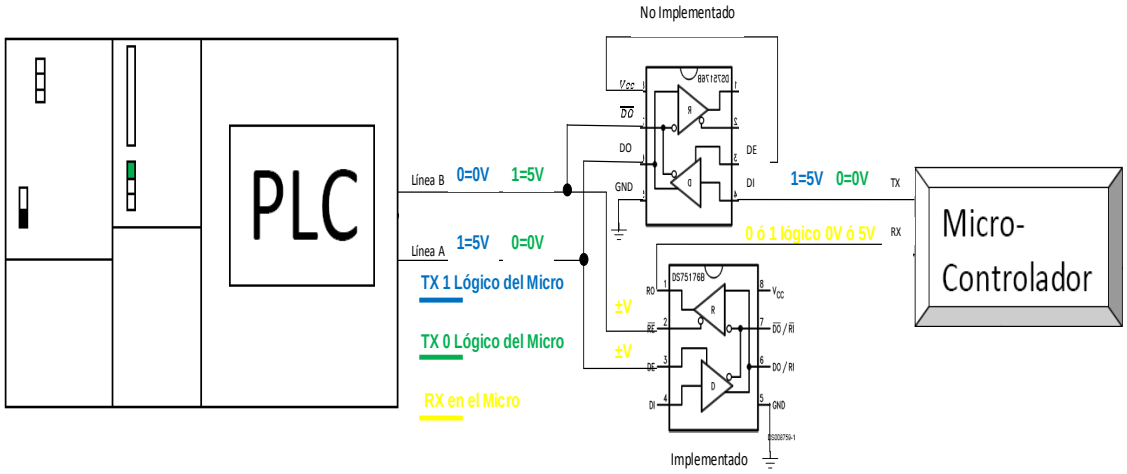


Figura 3.7: Diagrama de conexiones del Hardware con MAX485 con niveles de tensión.

Fuente: Elaboración Propia

Figura 3.8: Voltajes en Profibus con RS485.
Fuente: Elaboración Propia

o fin de línea que aunque el estándar define estas de 220 ohmios (*PROFIBUS Installation Guidelines*, 2009), si se varían de 120 ohmios a 330 ohmios como lo determinan otros fabricantes (*Profibus RS 485-IS User and Installation Guideline*, June, 2003), los voltajes de transmisión en el bus también varían con cambios representativos en las señales como se aprecia en la Tabla 3.1.

R entre líneas A, B	Maestro	Esclavo	Maestro con RS485	Esclavo con RS485
Sin Resitencia	-3.8v, 4.4v	-3.8v, 4.4v	-0.2v, 3.9v	0.2v, 3.9v
120	-1v, 3.8v	-2v, 2.6v	-0.2v, 3.9v	0.2v, 3.4v
220	-1.1v, 4v	-2.4v, 3v	-0.5v, 3.8v	0.3v, 4v
330	-1.1v, 4v	-2,6v, 3.2v	-1v, 3.7v	0.6v, 4.3v
560	-1.2v, 4.1v	-2.8v, 3.5v	-1v, 3.7v	0.8v, 4.3v

Tabla 3.1: Voltajes de datos transmitidos con diferentes resistencias de fin de línea.
Fuente: Elaboración Propia

Para esta conexión solo se utilizó la resistencia de 220 ohmios entre líneas Profibus, sin hacer uso de las resistencias de 390 ohmios a positivo y negativo como lo establece el estándar, pues estas resistencias ponen en conflicto la comunicación ver Figura 3.9.

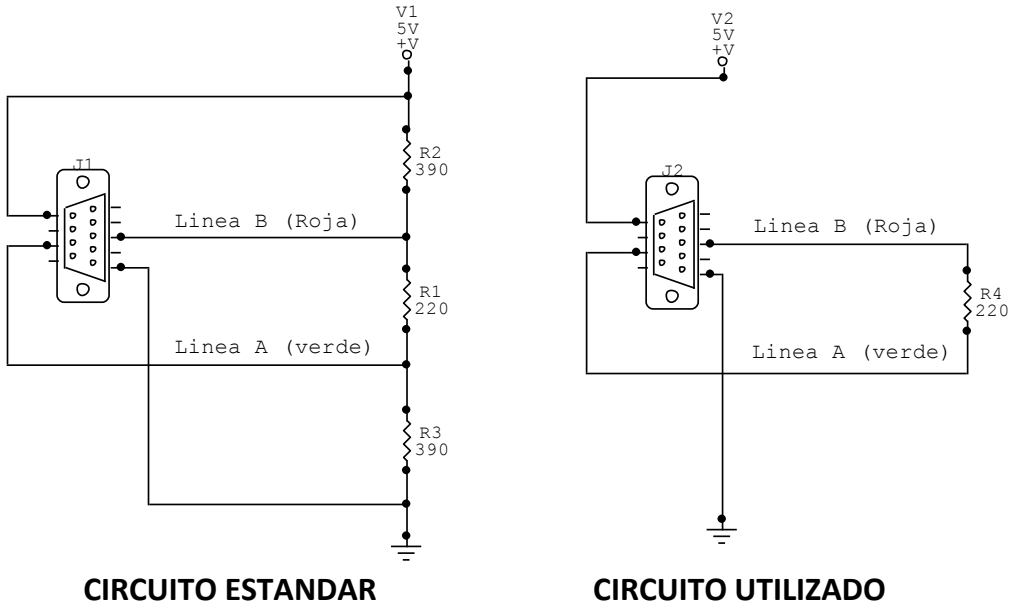


Figura 3.9: Conexión líneas Profibus con Resistencias terminales.
Fuente: Elaboración Propia

Debido a que a la salida de información del pin de transmisión del microcontrolador, no se tienen los voltajes de comunicación Profibus correctos y hay conflicto entre la impedancia de este pin del microcontrolador con el bus de datos del PLC maestro, se tiene que realizar todo un tratamiento de la señal de salida del microcontrolador hacia el PLC Siemens. Para esto se tienen en cuenta los niveles de voltajes del sistema Profibus en el proceso de transmisión, los cuales son voltajes positivos, negativos y voltaje en *standby* o modo de espera, donde este último establece la necesidad de mantener un nivel de voltaje en el bus mientras no haya transmisión y según sea el dato (0 ó 1) cambiar los estados del bus.

En el tratamiento de la señal de salida, se utilizan amplificadores operacionales JFET - TL082CP, los cuales se caracterizan por un consumo de tan solo 1.4mA, con capacidad para trabajar con voltajes diferenciales, protección de corto circuito en la salida, señal de salida estable, baja distorsión, alta impedancia de entrada, alimentación dual, 2 amplificadores operacionales por circuito integrado que soportan alto voltaje por su tecnología JFET y un rango de operación de temperatura de 0ºC a 70ºC (*TL082CP - Texas Instruments® Operational Amplifiers*, Abril, 2013). La señal se toma del pin de TX del microcontrolador y se lleva a un amplificador operacional TL082CP, el cual se configura como seguidor de voltaje, lo que permite acoplar la impedancia de salida del microcontrolador con el Gateway. Seguidamente se pasa esta señal a otro amplificador operacional que se configura como comparador, estableciendo un nivel de referencia y una entrada de voltaje que proviene del microcontrolador después de pasar por el seguidor, estableciendo los niveles de voltaje de transmisión de "1s" y "0s" y luego se conecta a un último amplificador operacional como restador, el cual recibe la señal del puerto de transmisión y controla el nivel de *standby* con otro pin del microcontrolador que establece el inicio de la transmisión. Para lo cual es necesario el nivel de referencia del comparador, con este se implementa todo lo que tiene que ver con el hardware del Gateway relacionado con los amplificadores operacionales.

Finalmente después de estos 3 amplificadores operacionales, se tiene un circuito que acopla las impedancias de los amplificadores operacionales con el bus Profibus, además de brindar los voltajes y corrientes necesarias para el bus, dando una ganancia en cada uno de estos parámetros. Para esta parte se implementa un circuito de amplificación clase B seguidor emisor-complementario, el cual se caracteriza por tener una ganancia unitaria, eficiencia del 78,5 %, ser no lineal y estar compuesto por 2 transistores BJT complementarios, es decir uno tipo NPN y otro PNP, lo que le permite trabajar con señales positivas y negativas con cada transistor y si no hay señal los transistores permanecen inactivos (López, 2002). Para esta implementación se utilizaron los transistores TIP41C y su complementario TIP42C, que son transistores con una potencia media de 65 vatios, soportan una corriente de 6 Amperios de manera continua por el colector y picos de hasta 10 amperios y voltaje colector emisor de 100 voltios como máximo (*TIP4xC - Silicon Power Transistor, Fairchild Semiconductor*, December, 2009). Este amplificador clase B va directamente desde la salida del último amplificador operacional que

está como restador, a las bases de cada uno de los transistores, a través de una resistencia de 220 ohmios como se ve en el circuito de la Figura 3.10. Para terminar, la unión de los 2 emisores de los transistores son los que van a la línea B de la red Profibus.

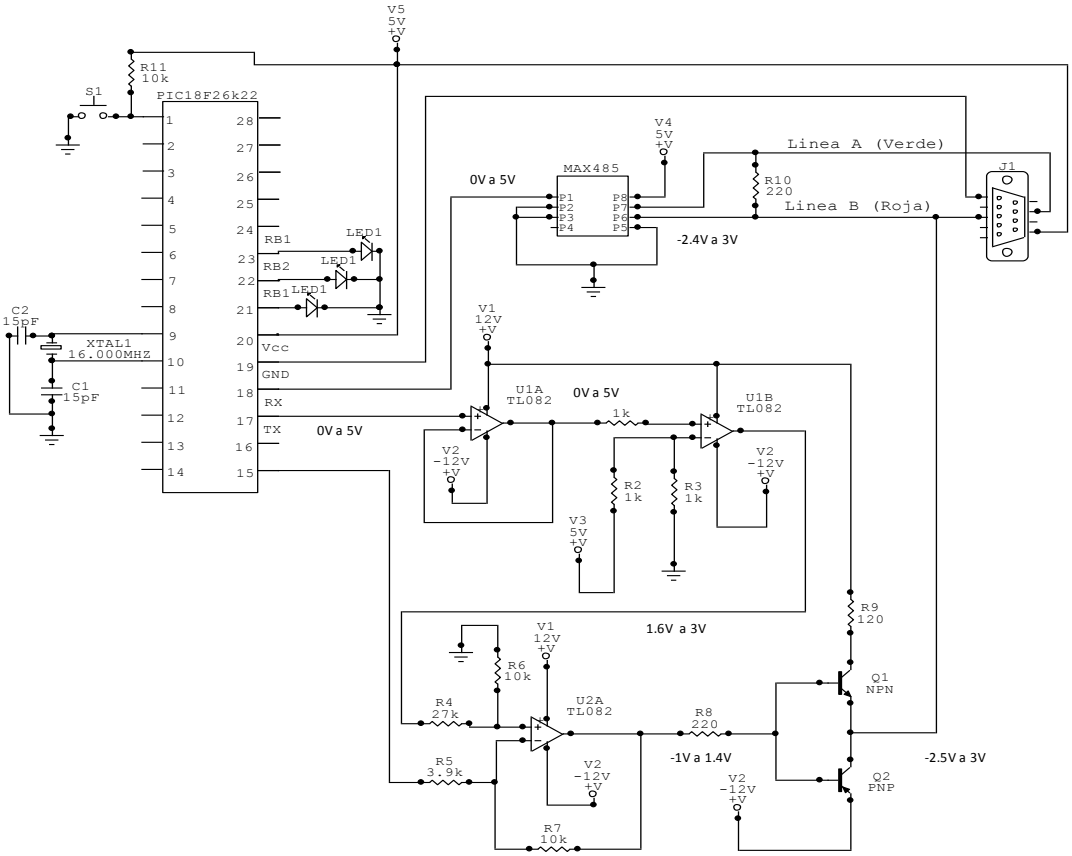


Figura 3.10: Circuito total del Gateway implementado
Fuente: Elaboración Propia

3.2.2. Software del Gateway

Para lograr los cambios de voltaje entre *standby*, voltaje positivo y voltaje negativo en el proceso de transmisión de señales, se implementa la señal adicional ya indicada anteriormente que es la que sale del pin 15 del microcontrolador, que mediante programación se establece por el puerto C en su posición 4 (RC4) del microcontrolador. Este pin se configura como salida y con este se lleva la señal al restador de voltaje, trabajando la señal con lógica inversa, de tal forma que si no hay transmisión, este pin permanecerá en un "1" lógico (5 Voltios), y una

vez se va a empezar la transmisión, este pasa a un "0" lógico y se empieza a realizar la tranmisión continua de "1s" y "0s" a través del pin de transmisión del microcontrolador (RC6 Pin 17) y una vez se termina de realizar la transmisión, se regresa el pin de control a "1 " lógico (Pin 15, RC4). Con estos 5 voltios que se le ponen y se le quitan a la línea de transmisión y que van al restador, se logran establecer los 3 niveles de voltaje en el Gateway, todo esto controlado por software.

Para chequear el correcto funcionamiento de este restador se simuló en Proteus (*Proteus 7.7 SP2. Labcenter Electronics Ltd*, 2009), estableciendo los niveles lógicos de voltaje y mirando si estos cambios lógicos representaban suficientes cambios en la señal de salida de tal forma que fuera similar al cambio diferencial de voltaje dado en una línea Profibus (línea A), estableciendo el nivel de voltaje ideal en el dato enviado ver Figura 3.11.

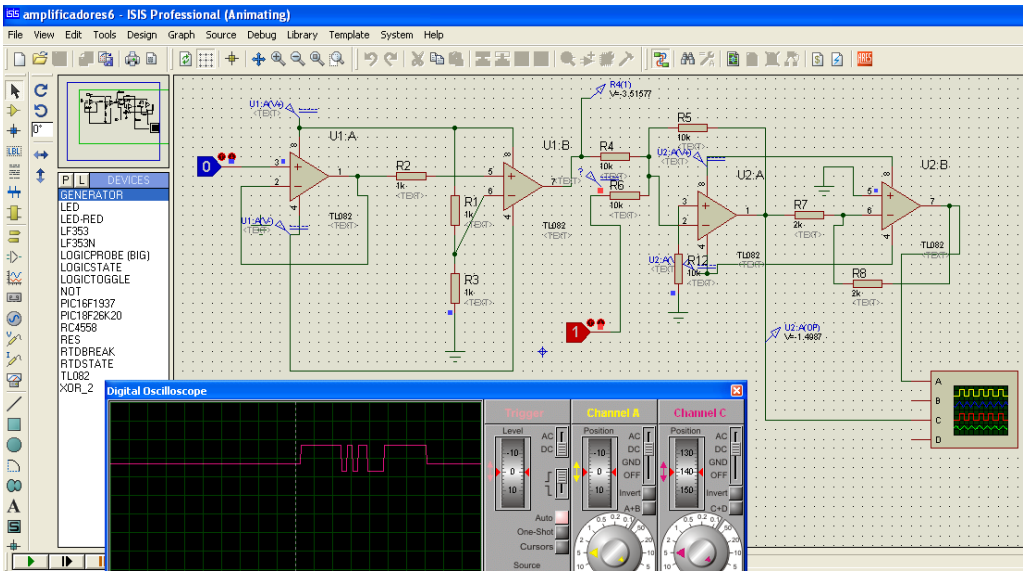


Figura 3.11: Simulación de Software implementado con niveles lógicos.
Fuente: Elaboración Propia

El circuito restador implementado en Hardware con amplificadores operacionales en conjunto con la implementación de software del pin de control del microcontrolador, da el nivel de voltaje negativo necesario para la comunicación Profibus, pues con esto se logra restar 5 Voltios al nivel de referencia de la señal, realizando la transmisión transparente con los niveles mostrados en la Figura 3.12.

Luego de implementar el circuito con su programa de control teniendo un correcto diseño de los circuitos con amplificadores operacionales y calculando las respectivas resistencias acordes para la correcta transmisión, en la conexión del microcontrolador que es la parte encargada del control de software del Gateway con el bus, se presentaron problemas de acoples de impedancias como se comentó

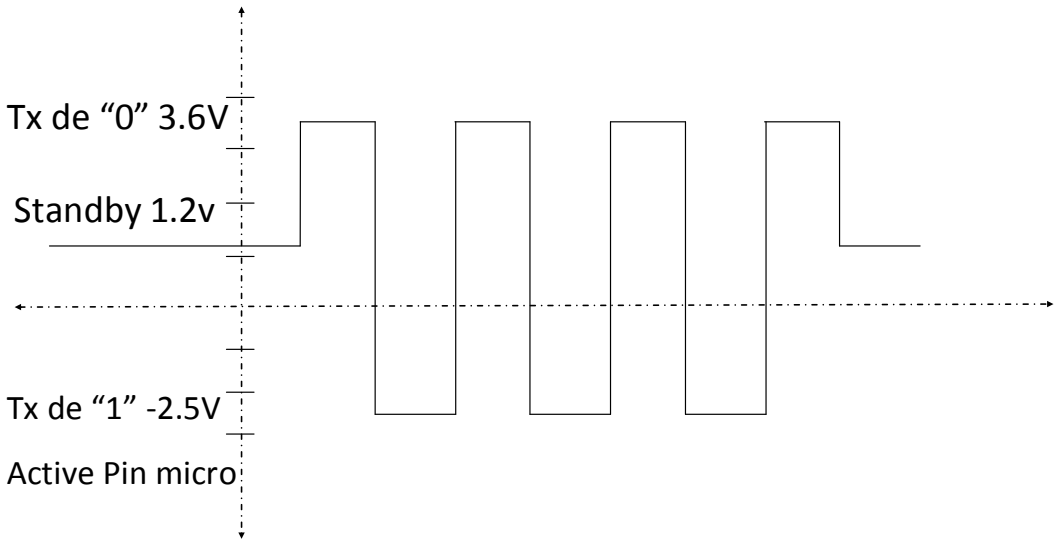


Figura 3.12: Niveles de voltaje del Gateway con el pin de control del microcontrolador sin conexión al bus.

Fuente: Elaboración Propia

anteriormente tanto en la entrada como en la salida de los datos, teniendo caídas de voltaje, en particular con el pin de TX del microcontrolador obteniendo señales fuera de lo esperado como se vé en la Figura 3.13.

Como se puede apreciar, se tienen diferencias de voltajes entre la Figura 3.12 y la Figura 3.13 con una disminución del 70 % en el valor de *standby*, del 55 % en la transmisión del “1” lógico y una perdida del 73 % en la transmisión del “0” lógico (Voltaje Negativo). Sin embargo este problema se logró corregir con el circuito indicado anteriormente, donde los transistores complementarios para la salida de la señal con cierta ganancia de voltaje permitieron el correcto acople con el PLC Siemens. El procesamiento previo por software de la trama a transmitir y verificación de los niveles de voltaje de salida permite el correcto funcionamiento del Gateway diseñado.

Poder cambiar los parámetros de comunicación, las direcciones del dispositivo esclavo, realizar una red de sensores y poder hacer un control distribuido con los PLCs, son características que se pueden implementar y programar con el Gateway desde la parte microcontrolada y administrada por software.

3.3. Programación del PLC

Para detectar las fallas típicas del PLC mediante el acople del Gateway, se debe realizar una programación específica en cada uno de los bloques funcionales

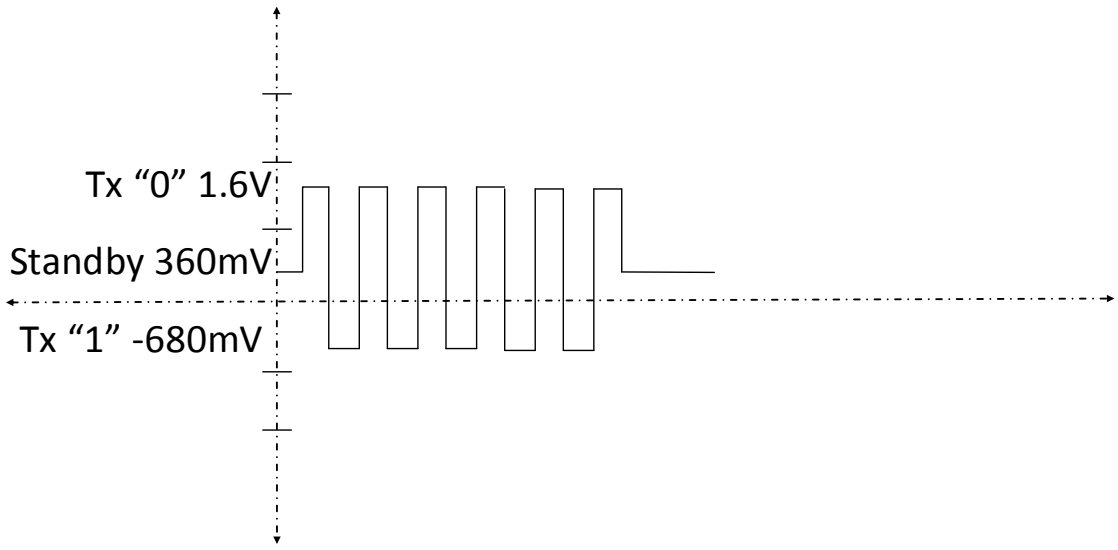


Figura 3.13: Niveles de voltaje del Gateway con el pin de control del microcontrolador con conexión al bus.
Fuente: Elaboración Propia

del PLC. Cada falla específica activa una variable del PLC esclavo, que finalmente será el microcontrolador quien categorice y realice la inspección rápida del PLC, diagnosticando correctamente la falla presentada, sin presentarse ambigüedad en la misma.

Para la comunicación con el microcontrolador, se puede realizar conexión con polling (sondeo) o paso de testigo. Donde en la programación del PLC se opta por utilizar el primer método, puesto que se tiene poco volumen de información. Con este método el maestro interroga de manera constante a las estaciones esclavas y cuando estas tienen información a transmitir, usan la trama de respuesta con los respectivos datos y como Profibus es determinista, el tiempo entre una recepción y transmisión es el mismo entre los maestros y esclavos. Mientras que el Paso de testigo es puesto en la red y rotado ya sea por una lista de direcciones o aleatoriamente lo coge quien desee transmitir y se usa cuando se tiene varios maestros en la red.

3.3.1. Configuración Tiempos

Los parámetros de configuración de la comunicación Profibus estándar viene por defecto a 1.5Mbps, sin embargo este velocidad en conjunto con los tiempos de intervalos entre trama y trama son cambiados para poder realizar un mejor procesamiento de las señales y comunicar de manera óptima el microcontrolador.

Inicialmente se configura el Maestro con una velocidad de comunicación de

9,6kbps con dirección Profibus de Maestro 10 y dirección Profibus de esclavo 5 y los tiempos entre tramas, son configurados con los valores máximos para poder realizar capturas óptimas de datos establecidos como sigue:

T_{slot_Init}: es el tiempo máximo que una estación maestra puede esperar por una respuesta. Valor máximo aceptado

$$T_{slot\_Init} = 16383\ t/bit \tag{3.1}$$

T_{sdr}: *Station Delay for Respond*. Es el tiempo de espera en una respuesta antes de generar una trama de respuesta.

$$MaxT_{sdr} = 1023\ t/bit \tag{3.2}$$

$$MinT_{sdr} = 54\ t/bit \tag{3.3}$$

T_{set}: Setup Time. El tiempo entre un evento (ejemplo, interrupción Syc) y la reacción necesaria.

$$T_{set} = 255\ t/bit \tag{3.4}$$

T_{qui}: *Quiet Time*. Tiempo de falla en el transmisor (tiempo de incertidumbre en la línea de estado) y/o en el tiempo de suicheo del repetidor. El tiempo en que una estación transmisora podría esperar después de que finaliza su trama antes de habilitar que este reciba ver Figura 3.14.

$$T_{qui} = 0\ t/bit \tag{3.5}$$

Donde el t/bit dependerá de la velocidad de transmisión del bus Profibus, determinando el tiempo de transmisión por cada bit. Luego las unidades estarán determinadas por el numero de veces del t/bit.

3.3.2. Configuración Bloques según Tramas de negociación

Cuando se tienen los tiempos configurados, se realiza una comunicación Profibus maestro-esclavo mediante un conector Profibus diseñado para tal fin. Con un conversor RS485 se llevan las tramas capturadas a niveles TTL que son tomadas por el puerto serial del computador en una terminal, donde se analizan cada una de estas tramas Profibus. En el análisis de las tramas, se estudia todo el estándar Profibus y con este se entiende el proceso de negociación dado entre un PLC maestro y un PLC esclavo, teniendo en cuenta el tipo de tramas, tamaño de las tramas, tipo de servicios o mensajes, respuestas, solicitudes, detección de errores y la estructura como tal de la trama byte a byte teniendo en cuenta el chequeo de distancia *Hamming* para el desplazamiento de los bits con sincronismo (Zuluaga Juan Gonzalo, 2014).

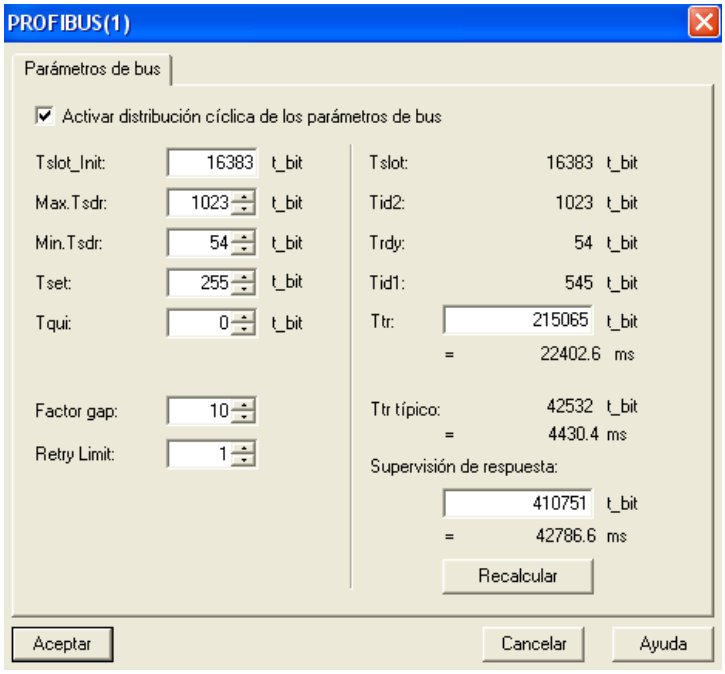


Figura 3.14: Tiempos configurados en la interfaz de programación del PLC Siemens
Fuente: Tomado de Software Simatic Step 7

Cuando el PLC Siemens se enciende, este toma un tiempo para iniciar el proceso de comunicación a través de su red Profibus y solo inicia la transmisión de tramas cuando el PLC maestro pasa de paro (*stop*) a programa corriendo (*run*). Inicialmente el PLC maestro envía la trama de paso de testigo indicando su dirección propia y verificando si hay más maestros en la red (DC 0A 0A), que por tener el maestro la dirección 10, esta se expresa en hexadecimal como 0A. Esta trama la repite pasando a enviar una trama *broadcast* larga seguida de un contador de trama y luego una trama *broadcast* corta con conteo de trama. Las tramas de *broadcast* se usan para verificar la cantidad de maestros que hay en la red y algunos esclavos especiales.

Seguidamente empieza el proceso de negociación donde el Maestro envía una trama típica de reconocimiento 68 05 05 68 85 8A 6D 3C 3E F6 16. Donde el 68 corresponde al SD2, los 05 corresponden a la longitud de trama la cual indica que es de 5 bytes y la repite, seguidamente esta el 68 que es el SD2 y sigue la dirección de destino y dirección de origen (85 8A) que tiene el primer bit en 1, indicando que existirán direcciones de extensión para la transmisión de información. Luego está el código de función que indica que es una trama SRD sin FCB de alta prioridad. Luego se encuentran las direcciones de extensión 3C, 3E que son para control y finaliza con el *cheksum* FCS sumando desde la dirección de destino y finaliza la trama con el 16h.

Ante esta trama el PLC esclavo contesta con una trama de longitud variable en donde sus direcciones tienen extensiones y son inversas a la trama del maestro (8A 85). Su función indica que es una estación esclava, sin FCB y con dato RDR de respuesta de baja prioridad con recepción ok (*Profibus Specification Normative Parts of PROFIBUS-FMS, -DP,-PA accordin to the European Standard EN 50 170*, 1998). Seguidos de las direcciones de extensión, datos, *cheksum* y final de trama; el maestro contesta con un conteo de trama.

Continuando con la negociación vuelve un encabezado de token DC0A0A, seguido de una trama de identificación que el PLC maestro manda al esclavo, el cual si la reconoce manda un E5 (recepción OK) y el maestro realiza un conteo de trama. Luego el maestro para verificar que si es el esclavo correspondiente manda otra trama de reconocimiento alternado el FCB antecedido por DC0A0A con otra información y el esclavo contesta E5 (OK), seguido por un conteo de trama del maestro. Finalmente, el maestro envía otro encabezado DC0A0A con una trama corta de confirmación contestando el esclavo con una trama de longitud variable con datos e información de recepción correcta que es seguida por el conteo de trama del maestro.

Después de realizada la negociación existe una homogeneidad en el flujo de tramas cuya configuración es un encabezado DC0A0A, una trama SD2 con datos para mandar al esclavo, que si no los hay, permanecen los bytes en 00h. El esclavo contesta de la misma manera en todos los casos variando solo la información a enviar al maestro sin variar la longitud de la trama, terminando el maestro con el conteo de trama. Este funcionamiento se puede ver con más claridad en el diagrama de flujo de la Figura 3.15.

Por programa, se configura el PLC Siemens maestro, con 2 bytes de comunicación hacia el PLC esclavo, lo que permite poner en 1 cualquiera de los 16 bits que transmite al esclavo para determinar el tipo de falla. Para esto se activa de a un bit en cada bloque de programa con los 12 tipos de fallas diferentes dadas en los bloques OB80 a OB122 del Siemens S7-300 (*Universal Controller SIMATIC S7-300 - PLCs*, 2011). Para esta programación se seleccionaron el Byte 00 y Byte 01 para la transmisión de fallas. Luego en el OB80 hay una línea de programa activando la bobina Q0.0 (en ingles), de esta manera se indica que la falla es de ciclo y lo que finalmente se manda en la comunicación Profibus es un 01 en el byte 8 de la trama Profibus. De la misma manera se programó el OB82 que indica falla en el módulo 1 de entradas y salidas del PLC, activando la salida Q0.1 del PLC que es la de comunicación con el esclavo, luego en la trama Profibus si esta se activa, se envia el byte 8 con valor 02h y así sucesivamente con cada uno de los bloques de detección de fallas del PLC Siemens maestro. Donde si en la trama hay un 04h es falla de la CPU o si hay un 05h indicaría falla del ciclo de programa y falla de CPU y así completando hasta los 2 bytes que se transmiten al microcontrolador. Si adicionalmente pasan más de 1.7 segundos sin recibir ninguna trama del PLC, esto indica un error en la comunicación Profibus, reportando este tipo de falla el microcontrolador.

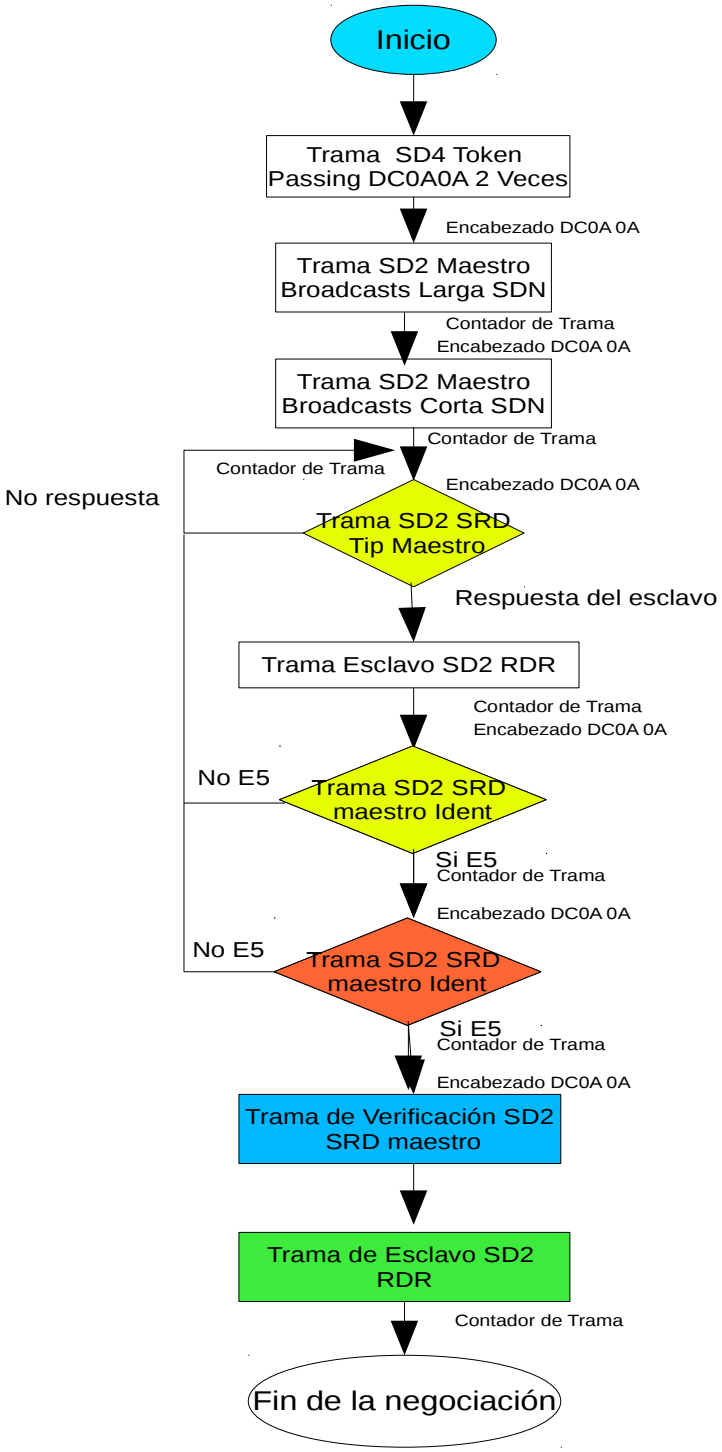


Figura 3.15: Diagrama de flujo Proceso Negociación PLC maestro PLC esclavo.
Fuente: Elaboración Propia

La lectura de las tramas Profibus, permite verificar el estado del PLC actual, lo cual se puede realizar mediante el envío de tramas especiales. Además se puede mejorar el tiempo en las soluciones o mantenimiento que se le puede hacer a un proceso, conociendo el estado de las variables en el PLC. De la misma manera se puede solicitar información adicional previamente programada en el PLC, con el envío de solicitudes especiales dadas en la carga útil de la trama Profibus, mediante configuración de los bytes de comunicación dados entre un PLC Siemens maestro y un PLC Siemens esclavo.

Finalmente, se estable la comunicación con el Gateway diseñado, el cual toma los voltajes diferenciales del bus y los transforma en datos digitales correctamente entendibles por el microcontrolador y de manera inversa, transforma los datos digitales de la UART del microcontrolador y los convierte en voltajes diferenciales con niveles positivos y negativos entendibles por el PLC como se observa en la Figura 3.16. Con esto se logra una comunicación estable, puesto que ante desconexiones y conexiones, ausencias de fluido eléctrico y retorno de este, siempre se logra restablecer la comunicación, sin presentarse error en el bus BF en verde, mientras que bajo otras condiciones, la comunicación con MAX485 o interfaz UART-RS232-RS485-Profibus, siempre presenta error poniendo el indicador del bus BF (Bus Fault) en rojo. Además debido al correcto acople de impedancias, el PLC Siemens maestro siempre ve al dispositivo embebido como un PLC Siemens esclavo, permitiendo transportar información y datos como control de variables, cambio de parámetros básicos, encendido y apagado de entradas o salidas y comunicación general entre estos dispositivos.

Aunque en los resultados obtenidos y medidas de las tramas Profibus se aprecia una pequeña diferencia de voltaje, esta no es significativa y no impide la correcta transmisión y empalme dados entre el PLC Siemens maestro y el microcontrolador.

3.4. Diseño de Experimento

En esta parte se presenta el diseño de experimento de la fase 2, donde se realizan la pruebas de verificación de la interfaz de comunicación dada entre el PLC y el microcontrolador a través del bus estándar Profibus para el PLC Siemens.

Para verificar el funcionamiento del Gateway diseñado y el programa del microcontrolador, se realizaron pruebas de conexión y desconexión del bus, haciendo que el PLC detectara errores si los hubiera y determinando los tiempos de respuesta ante cada tipo de desconexión y restablecimiento de la misma.

De las pruebas realizadas se trabajó en 5 conjuntos de 10 medidas cada uno y a cada conjunto se sacó un promedio, obteniendo 50 medidas en total y con estas 5 resultados promedio finales. Entre las pruebas realizadas se encuentran: la desconexión del cable Profibus, la desconexión de la alimentación del Gateway y finalmente la desconexión de la alimentación del microcontrolador.

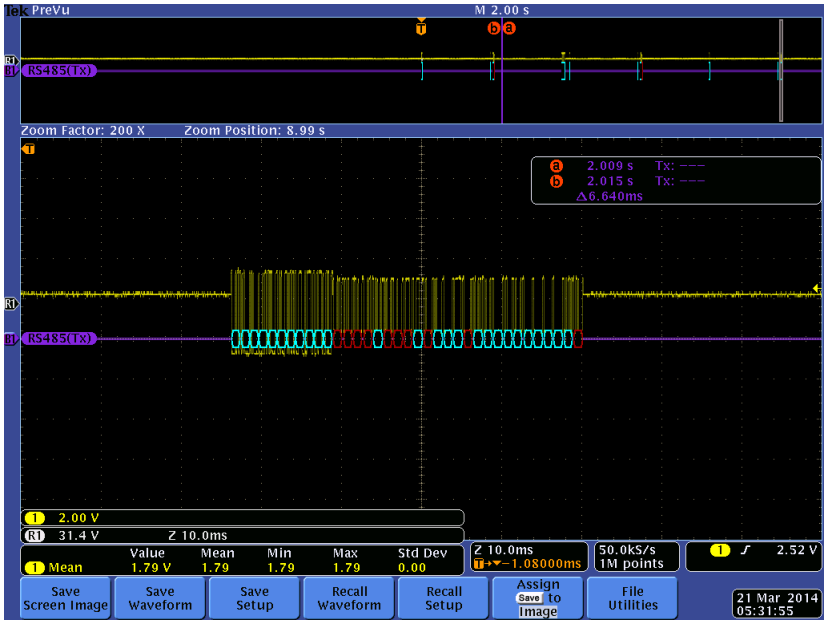


Figura 3.16: Tramas Profibus PLC vs Gateway con óptimo funcionamiento e igual nivel de *standby*.

Fuente: Elaboración Propia

3.4.1. Desconexión y Reestablecimiento con Cable Profibus

En esta desconexión y restablecimiento de la comunicación, se verifica la interfaz completa hasta los cables de comunicación Profibus, estableciendo los tiempos que tarda el PLC en detectar la falla en la comunicación y los tiempos en restablecerse la misma a través del cable Profibus que se encuentra entre el PLC y el Microcontrolador con el Gateway diseñado.

Teniendo en cuenta que una desconexión del cable implica un cambio de impedancia e interrupción de las tramas Profibus enviadas de manera continua entre el PLC y el microcontrolador, los resultados de los datos promedio de las medidas tomadas se muestran en la Tabla 3.2.

3.4.2. Alimentación Gateway

La otra prueba realizada dentro del diseño de experimento, consiste en verificar los voltajes de alimentación dados por el Gateway e identificar la influencia de estos en las tramas Profibus, donde al realizar la desconexión de Alimentación del Gateway, los amplificadores operacionales no trabajan y aunque las tramas Profibus siguen transportándose, el PLC maestro no detecta las tramas enviadas por el microcontrolador, pues estas no alcanzan los niveles de voltaje dados por el estándar, luego se presentan problemas en la comunicación.

Tiempo de Conexión y Enganche	Tiempo de desconexión del Bus (fallo)
10.37 s	4.69 s
9.86 s	5.04 s
10.24 s	4.78 s
11.24 s	4.58 s
10.46 s	4.31 s
Prom. Total 10.434 s	**4.68 s**

Tabla 3.2: Tiempos Conexión y Desconexión del Cable Profibus en Enganche y Fallo.

Fuente: Elaboración Propia

Para esta prueba se suspende la alimentación del Gateway cuyos tiempos de desconexión son los mismos que del bus, ya que el PLC para detectar la desconexión y mostrar el fallo en la comunicación Profibus, lo tiene en cuenta de la misma manera, sin embargo se tomaron los tiempos de encendido en la alimentación del Gateway esperando que enganche el PLC con el microcontrolador. En este enganche se tuvieron tiempos cortos como largos con mínimo de 9 segundos y máximos de hasta 31 segundos. Para estas pruebas se tomaron 50 medidas a las cuales se sacaron promedios de a 10 y se obtuvieron los siguientes resultados de la Tabla 3.3.

Tiempo de Conexión y Enganche con Alimentación Gateway
12.31 s
11.21 s
11.75 s
9.95 s
12.08 s
Promedio Total = 11.46 s

Tabla 3.3: Tiempos Conexión Gateway al alimentarlo.
Fuente: Elaboración Propia

3.4.3. Desconexión y Enganche Alimentación del Microcontrolador

En esta prueba, se toman igualmente tiempos de conexión y desconexión por parte del microcontrolador, donde en el proceso de enganche se debe tener en cuenta el tiempo de reinicio del microcontrolador con toda la configuración de sus parámetros, sin embargo como este es del orden de milisegundos, los tiempos tomados no se ven tan afectados por las medidas.

Cuando se realiza la desconexión de la alimentación del microcontrolador, el consumo de corriente cambia en la alimentación del voltaje negativo del Gateway el cual pasa de 10mA en -V a 120mA en el +V, lo que quiere decir un cambio de impedancia en lo que tiene que ver con bus el cual se encuentra interconectado al puerto USART del microcontrolador, por ende, si este está desactivado, el puerto no presenta impedancia y por esto se dá el cambio en la corriente.

En el proceso de desconexión se tuvieron algunos tiempos ilógicos, en que el PLC no detectara la ausencia de las tramas enviadas por el microcontrolador, esto se debe a que el cambio de impedancia al desconectar el microcontrolador, determinan cambios en los niveles de voltaje, los cuales crean un conflicto en el PLC no detectando error inmediato en la comunicación, sino como un circuito abierto teniendo tiempos de fallo de hasta 1 minuto con 9 segundos.

Los otros tiempos promedios de conexión y desconexión se pueden apreciar con claridad en la Tabla 3.4.

Tiempo Enganche Microcontrolador Conectado	Tiempo perdida de conexión microcontrolador desconectado
11.9 s	4.38 s
9.72 s	4.7 s
11.54 s	4.72 s
11.47 s	4.49 s
12.34 s	4.59 s
Prom. Total= 11.394 s	**4.776 s**

Tabla 3.4: Tiempos Conexión y Desconexión cambiando alimentación el Microcontrolador.

Fuente: Elaboración Propia

Entre las pruebas realizadas con el microcontrolador, fue la de presionar el botón de reinicio (*Master Clear*), el cual quitaba el nivel de voltaje del restador del amplificador operacional, el cual es controlado por un pin del microcontrolador, lo que causaba que la línea del bus, tuviera un voltaje negativo de -8.8 voltios a -9,2 voltios (ver Figura 3.17), haciendo que el PLC anulara el envío de tramas para chequear la conexión y que este abriera el bus, por lo que es necesario reiniciar el PLC para que el bus vuelva a trabajar y el PLC maestro envié las tramas de verificación de conexión y del proceso de negociación.

3.4.4. Otras Pruebas y Resultados de esta fase

Al momento de encender el PLC que se encuentra conectado con el Gateway y este a su vez con el microcontrolador, se tiene que entender que el microcontrolador realiza un inicio demasiado rápido respecto al PLC, pues éste antes de realizar el enlace, realiza un chequeo de sus propios puertos y sistema cargado, además de

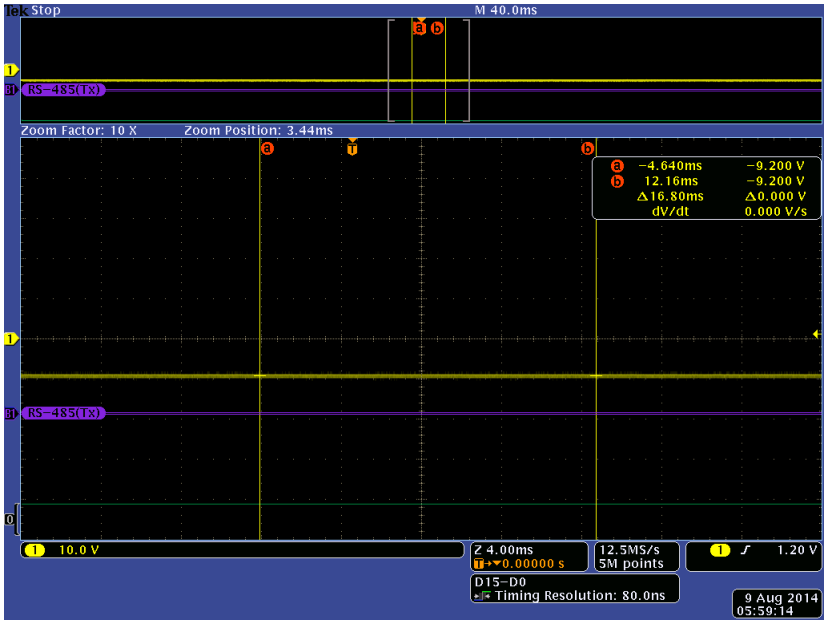


Figura 3.17: Voltaje bus Profibus ante reinicio del microcontrolador.
Fuente: Elaboración Propia foto osciloscopio Tektronik MS04034B.

configurar todo el hardware para su correcta conexión, cuyo proceso para pasar del estado de encendido a su función de "RUN" tarda 45 segundos en enlazarse. Aunque cuando se pasa del estado de "STOP" a correr el programa, solo tarda 13.12 segundos en enlazarse con el microcontrolador.

El PLC maestro también se encendió y apagó varias veces, para verificar el funcionamiento y condiciones de establecimiento del enlace, mediante el Gateway diseñado para la conexión con el microcontrolador, cuya comunicación obtenida fue óptima y no se presentó ningún inconveniente al realizarlo varias veces seguidas.

Algunas de las imágenes obtenidas de la comunicación establecida en esta fase teniendo los niveles de voltaje establecidos, se pueden apreciar en la Figura 3.18, donde se aprecian los voltajes modificados del Gateway de una trama enviada por el microcontrolador, la cual se decodifica con el estándar RS485 correctamente.

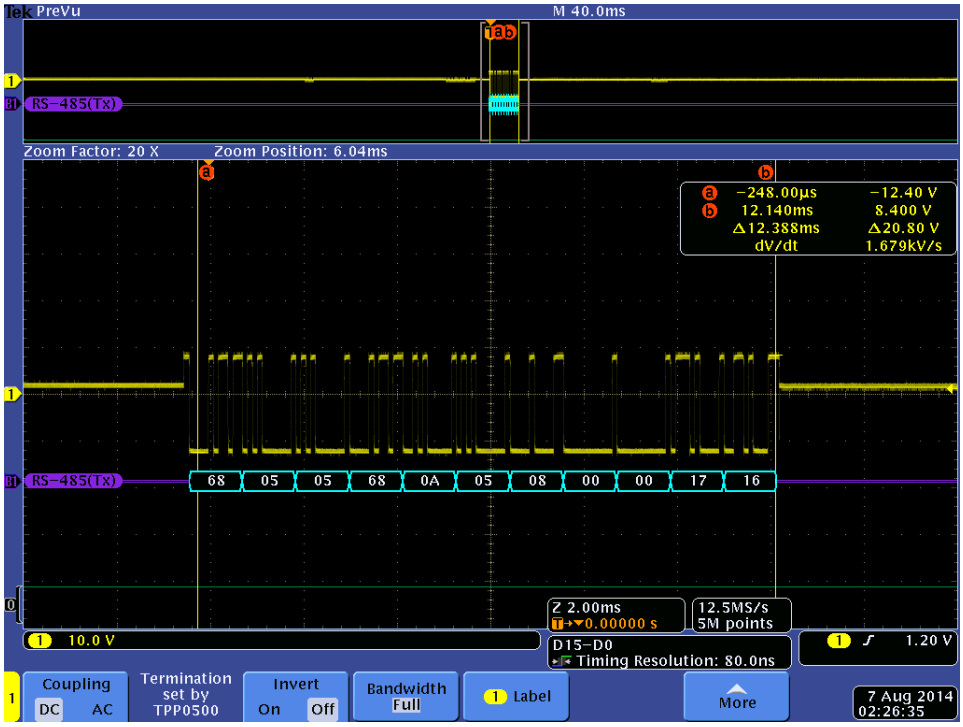


Figura 3.18: Trama enviada del microcontrolador con voltajes modificados por el Gateway.

Fuente: Elaboración Propia foto osciloscopio Tektronik MS04034B.

Capítulo 4

Interconexión PLC-PDA vía ZigBee

La interconectividad entre el PLC y la PDA dentro del proceso industrial, teniendo en cuenta que esta cumple con los requerimientos básicos de conexiones en un ambiente industrial para el control y monitoreo, se logra mediante el enlace de la fase 1 que conecta la PDA con el módulo ZigBee mediante la red GSM explicada en el capitulo 2 y la fase 2 con la que se enlaza el PLC con el microcontrolador utilizando la red Profibus la cual se explica en el capitulo anterior.

Esta conexión se establece mediante ZigBee, utilizando este medio para hacer de los 2 sistemas dados en la fases 1 y 2, uno solo, desarrollando la estructura metodológica para el diagnóstico de fallas de un PLC que se conecta mediante la red ZigBee a una PDA logrando establecer monitoreo y control, siendo este el objetivo planteado en el desarrollo de este proyecto.

La integración del PLC con ZigBee, se realiza mediante la interfaz Profibus establecida con el microcontrolador y gracias a que este procesa la información y posee dos puertos, permite replicar los mensajes recibidos. Luego de establecer esta comunicación se cierra el ciclo con la PDA permitiendo la transferencia de información de manera bidireccional y envío de mensajes simultáneos como se ve en la Figura 4.1.

En este capitulo se muestra el proceso de integración de las 2 fases planteadas, iniciando con la configuración del microcontrolador conectado al PLC por Profibus para la detección de fallas, seguidamente se establece el puente de conexión entre el módulo ZigBee y el puerto 2 del microcontrolador creando los mensajes a enviar en cada falla hacia la PDA, con su respectiva inicialización y finalmente se trabaja en los mensajes realizados desde la PDA dirigidos al PLC, con los cuales se ejercen acciones de control vía Profibus.

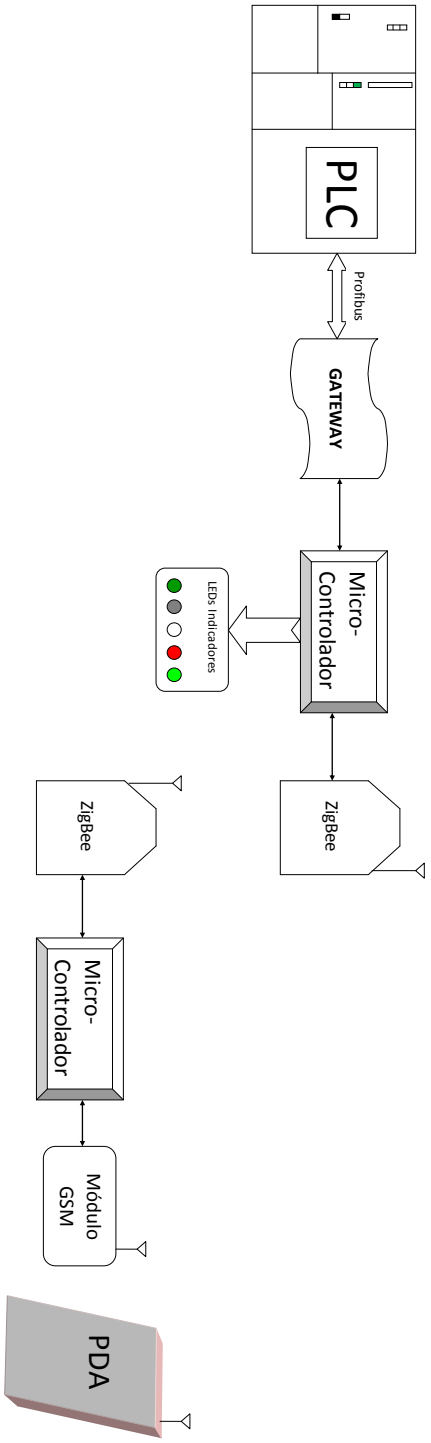


Figura 4.1: Diagrama de Implementación de la conexión del PLC con la PDA via ZigBee.

Fuente: Elaboración propia.

4.1. Configuración Microcontrolador

En la etapa inicial de configuración del microcontrolador, teniendo claro que este tiene un programa almacenado, mediante el cual se establece la comunicación Profibus, se le agrega a este mismo programa la comunicación ZigBee, enlazando el microcontrolador Esclavo con la red ZigBee. Gracias a que el microcontrolador utilizado, maneja 2 puertos seriales, en un puerto, se establece la comunicación Profibus y en el otro la comunicación ZigBee la cual se configura como se muestra a continuación.

4.1.1. Configuración Puerto 2 Microcontrolador

La comunicación ZigBee que se establece con el puerto 2 del microcontrolador, se realiza a una velocidad de 9600 baudios, sin control de paridad, funcionando con interrupción en la recepción y transmisión de cadena de bytes en formato ASCII tipo *string*, mediante espera de que el *buffer* de transmisión TRMT esté desocupado para enviar el siguiente byte. Adicionalmente se configuran los pines de transmisión (TX) y de recepción (RX) en el TRISB del puerto USART2.

Una vez se tiene configurado el USART2, se debe inicializar el módulo ZigBee, el cual se configura de la misma manera como se configuró en la fase 1, mediante un "enter " (\r o 0x0D) y luego sabiendo que se encuentra el ZigBee en el menú, se le envía la letra "B " para que quede trabajando en el modo *Bypass*.

Los datos que provienen del módulo ZigBee, son almacenados en un *buffer* de datos, cuyo *buffer* se compara con una estructura estándar, definida con una sintaxis para la transmisión de datos hacia el PLC por Profibus. Este *buffer* es limpiado en cada ciclo, una vez se procesa la información que se tiene dentro de este.

De la misma manera, si se quieren enviar datos desde el microcontrolador conectado a la red Profibus por el módulo ZigBee, es sino colocar la trama de datos a enviar en la cadena tipo *string* y este los monta en el registro TXREG2 *byte* a *byte* una vez está desocupado el *buffer* de transmisión.

4.2. Transmisión Mensaje GSM al ZigBee conectado a Profibus

Cuando el módulo GSM recibe un mensaje, este es capturado y procesado por el microcontrolador 1, el cual toma la información útil y la reenvía al módulo ZigBee, luego el ZigBee remoto (2) que se encuentra conectado a la red Profibus y al microcontrolador 2, reciben dicho mensaje.

Para poder procesar y analizar correctamente el mensaje, se realizan pruebas estándares del formato que se recibe en las mismas, cuya estructura siempre está precedida por su valor en hexadecimal de 0x0A. Esto se debe tener en cuenta para

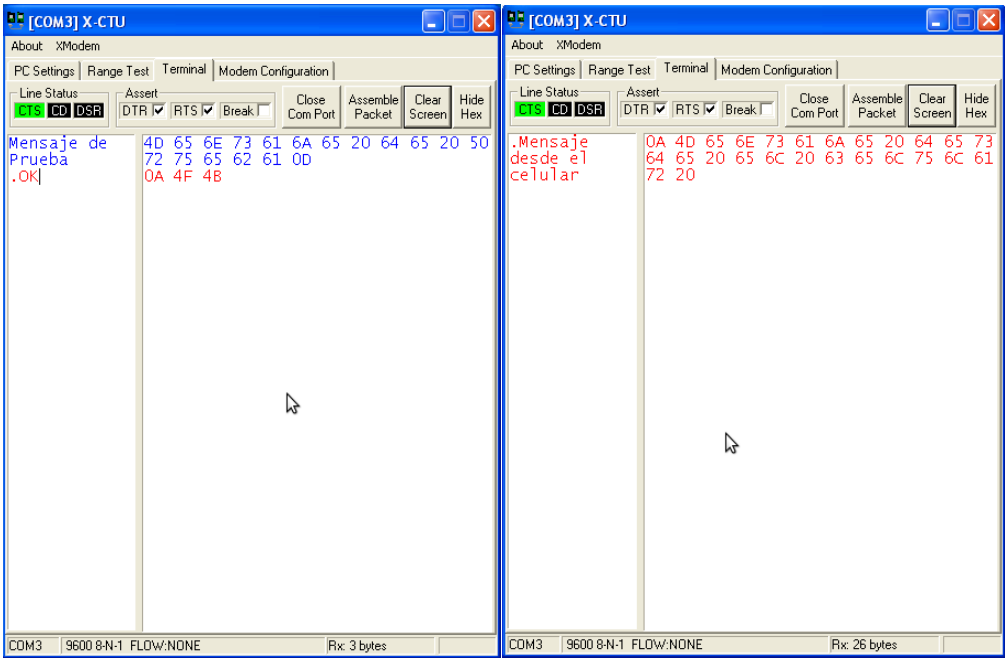


Figura 4.2: Mensajes del módulo GSM donde al inicio siempre maneja el *Byte* 0A
Fuente: Elaboración propia, Imagen interfaz ZigBee del PC.

poder realizar un correcto control con los mensajes recibidos como se aprecia en la Figura 4.2.

Una vez se recibe el mensaje desde el ZigBee 1 conectado al módulo GSM, se realiza una comparación *Byte* a *Byte* mediante la operación XOR, donde si son iguales dará 0 y si son diferentes dará 1. Luego se cuentan la cantidad de bytes iguales y con esto se ejerce la acción de control realizada en el envío de mensajes de control que se explica a continuación.

4.2.1. Envío mensajes de control

Para el envío de mensajes de control desde el celular o PDA hacia el Gateway Profibus, es necesario generar una sintaxis la cual tiene una estructura metodológica que permite cambiar *bytes* completos de alguna variable del PLC como activar o desactivar salidas, cambiar parámetros del PLC o de algún controlador PID si se desea, llevar variables como se haría desde una pantalla, etc.

La estructura de sintaxis garantiza el funcionamiento del Gateway, evitando que cualquier mensaje enviado por un cable operador o de una persona diferente al ingeniero que tiene control de la planta, interfiera en los parámetros de control y aplicación del PLC.

Para esto se envía inicialmente la palabra “Control ” teniendo en cuenta que la “C” está en mayúscula. Seguido de la palabra “Control”, se envía la palabra “byte”

Figura 4.3: Mensaje con Sintaxis enviado desde la PDA.
Fuente: Elaboración propia, imagen de mensajes tomada de la PDA.

con la "b" en minúscula. Luego el número de *byte* a controlar, lo cual dependerá del programa realizado en el PLC donde podrá ser desde el byte 0 al 9 y finalmente el dato en hexadecimal el cual se deberá escribir "x00 " o "x" el valor deseado finalizando con "# ". Luego la estructura total de la sintaxis del mensaje a enviar será: "Controlbyte1x03# " como se vé en la Figura 4.3.

Cuando el mensaje se recibe correctamente mediante la sintaxis dada, el Gateway toma el *byte* de control y se encarga de montarlo sobre la trama Profibus para de esta manera ejercer la acción de control solicitada y una vez se ha mandado esta al PLC, el Gateway devuelve la respuesta de "Control realizado " al módulo ZigBee 1 remoto que está conectado al módulo GSM y este lo retorna a la PDA.

Con el mensaje de control enviado, se puede encender y apagar un *byte* completo donde si se quiere todo encendido es mandar el parámetro "xFF " o todo apagado "x00 ". Por ejemplo mandar "xA5 ", implica mandar un 1 a los bits 7,

5, 2 y 0, teniendo en cuenta que las letras del número en hexadecimal siempre se deberán colocar en mayúscula, ver Tabla 4.1.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	0	1	0	1

Tabla 4.1: Envío *Byte* de control xA5 para Trama Profibus.
Fuente: Elaboración propia.

4.3. Detección de Fallas

Uno de los ejes fundamentales de este trabajo se basa en la detección de fallas que puede presentar un PLC en un proceso industrial, el cual se puede monitorear remotamente mediante una PDA. Para esta parte se busca desarrollar una estructura metodológica que garantice el envío de información al ingeniero de planta, según los problemas que se puedan presentar y quizás detengan el proceso productivo de la empresa.

En esta sección, se explica todo el proceso de detección de fallas, el cual está relacionado con la configuración que se tenga del PLC y con este el envío de tramas Profibus al Gateway esclavo que identifica el tipo de falla presentada.

Para esta aplicación, es necesario definir claramente las fallas a identificar y comprobar que estas son las fundamentales en el proceso productivo. Teniendo en cuenta lo realizado en la fase 1, donde se comunica la PDA con el módulo ZigBee y en la fase 2 donde se establece la comunicación Profibus con el microcontrolador, se busca enlazar estas fases con el objetivo de detectar las fallas concretas de manera correcta y confiable. Este enlace se dá mediante la comunicación inalámbrica a través de los módulos ZigBee conectados a cada uno de los circuitos de las fases correspondientes, logrando en este punto establecer el puente de comunicación.

Además como se comentó anteriormente, el protocolo ZigBee cumple con los requerimientos para el monitoreo de los procesos en un ambiente industrial (Park y cols., 2007). Luego la integración de este con el estándar Profibus para la detección de fallas, busca compatibilidad y viabilidad en su implementación en la industria colombiana, debido a su fácil conectividad y configuración del bus Profibus en el PLC como se describe a continuación.

4.3.1. Configuración PLC

Un programa bien estructurado en un PLC, maneja condiciones de seguridad, como alarmas sonoras, indicadores luminosos, paros de la maquina, llamados a bloques de fallas, detección de subprocesos, entre otros. Estas condiciones son

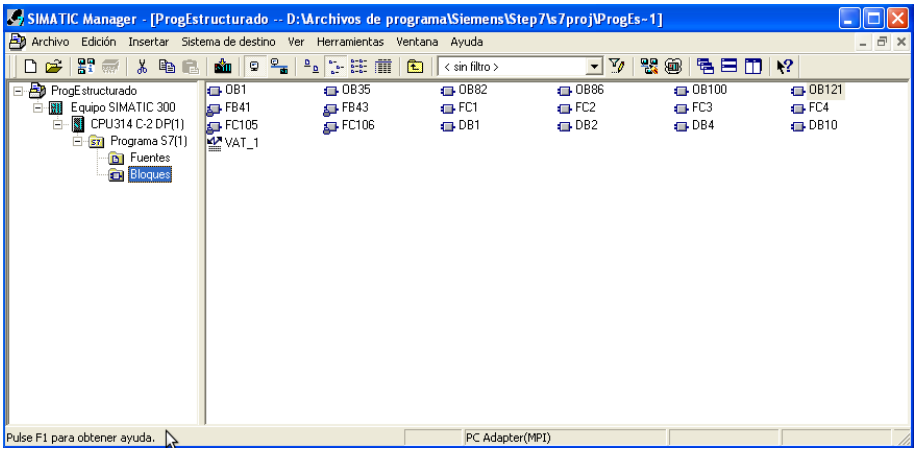


Figura 4.4: Programa estructurado con varios bloques que permite el control de seguridad.

Fuente: Elaboración propia, imagen tomada de Step 7.

fundamentales en un programa y no se tienen que modificar en el programa para la integración de la detección de fallas con la PDA, gracias a la estructura metodológica diseñada para tal fin.

Un programa como el que se vé en la Figura 4.4 se caracteriza por una construcción modular con un conjunto de funciones que determinan en los subprogramas condiciones de operación, que a su vez se encuentra interconectado con una Función principal (FC3), la cual es la encargada de manejar las condiciones de seguridad del proceso en particular. Luego este sería el único bloque a intervenir.

Sin embargo, se debe aclarar que tampoco es tan simple como un "Plug and Play " del Gateway, pues se deben realizar pequeñas modificaciones del programa, del PLC, realizando siempre un backup del mismo. Donde estas modificaciones van a estar relacionadas con cada una de las alarmas o paros dados por el PLC que se encuentre en el bloque de función en particular, además de los bloques de fallas OBxx (Ver capitulo 1) donde se deben de tener programas que también activen dichas alarmas.

La modificación más simple del programa del PLC, es colocar en paralelo a cada una de las bobinas de condiciones de seguridad, otra bobina que activará la salida correspondiente al byte que comparten en la comunicación Profibus con el Esclavo (ver Figura 4.5). Esta bobina activará un bit y una vez el Gateway lo detecte, enviará un mensaje al ZigBee Remoto, dependiendo del bit activado dentro del byte. Una vez es identificado, este pasará al módulo GSM para llevarlo a la PDA.

Configurados los bits a activar en el esclavo, se debe programar el Gateway o microcontrolador en el hardware, lo cual se establece como una CPU Siemens SIMATIC 300 la cual se le agrega al proyecto y se le carga el Hardware de la CPU

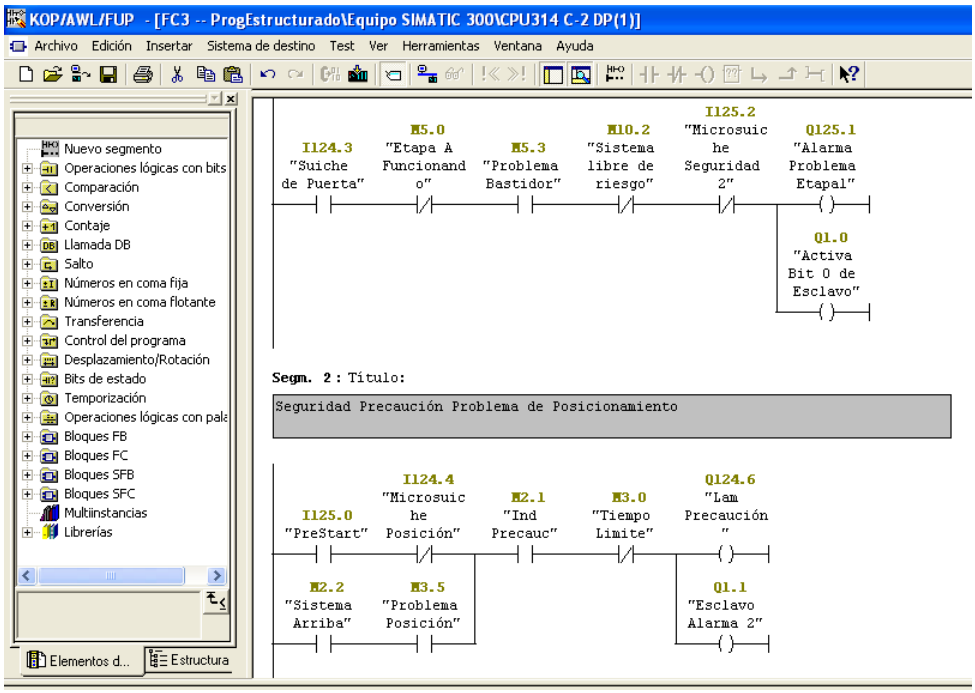


Figura 4.5: Modificación de Programa colocando Bobinas del Esclavo en Paralelo.
Fuente: Elaboración propia.

que para este caso se seleccionó la 314C-2DP, por tener comunicación Profibus directa y sus características óptimas de control.

Al cargar la CPU se establece la comunicación Profibus con la Dirección deseada, que para este caso se utilizó la dirección 5, a una velocidad de 9,6 kbit/s, con las propiedades del bus Profibus, dadas en el ajustes de red con los tiempos personalizados en los parámetros del bus explicados en el capitulo anterior como se vé en la Figura 4.6.

Luego en las propiedades del DP, se debe seleccionar el modo de operación del Gateway, el cual será como esclavo dentro de la red Profibus, recibiendo y enviando bytes de control y fallas desde el PLC maestro el cual tiene el programa original de la empresa. Para poner a trabajar este Gateway conectado al microcontrolador, se debe ir a la pestaña modo de operación y en esta seleccionar Esclavo DP, dejando las direcciones de diagnóstico y direcciones del *slot* por defecto junto con los otros parámetros

En la pestaña de configuración se deben crear los bytes con los cuales se va a establecer la comunicación entre el PLC maestro y el Gateway esclavo. En este punto se pueden crear los bytes que se deseen, sin embargo dependiendo de la cantidad de bytes, será la trama Profibus a transportar. Para esto se inserta el byte con el botón nuevo y allí se selecciona la dirección, si el byte es de entrada o salida y la longitud que se trabaja por defecto de 1 byte. Teniendo claros los

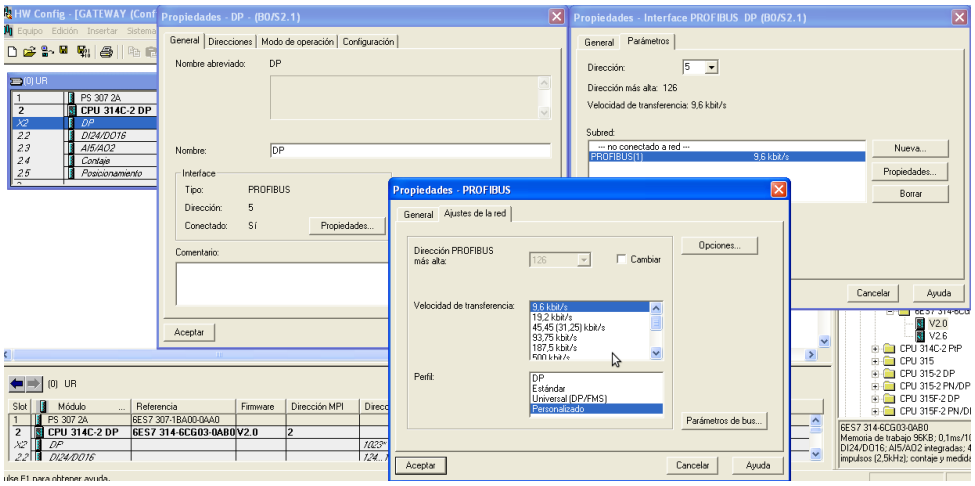


Figura 4.6: Configuración del Hardware del Gateway en programa existente.
Fuente: Elaboración propia, imagen tomada de Simatic Step 7.

bytes que se van a utilizar tanto de entrada como de salida, se puede mandar la información desde el Gateway al PLC con los bytes de salida del esclavo o desde el PLC al Gateway con los bytes de entrada del esclavo. Estas configuraciones del modo de operación y bytes se ven en la Figura 4.7.

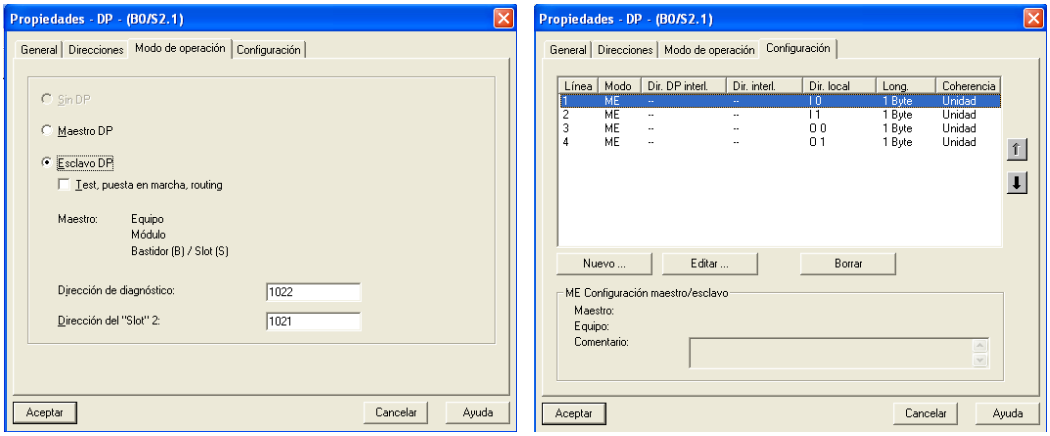


Figura 4.7: Configuración de Bytes y Modo de Operación del Gateway.
Fuente: Elaboración propia.

Finalmente, en el Hardware del PLC maestro, se debe agregar la red Profibus, si esta no existe en el programa original, o si existe es sino conectar a esta red el Gateway previamente configurado. Para agregar la red se da en DP y en propiedades se selecciona la dirección y la red Profibus creada con el Gateway similar a lo presentado en la Figura 4.6 colocando la dirección 10 para el maestro.

En el catalogo de elementos de la derecha, en Profibus DP, se busca Estaciones ya configuradas, y se coge una CPU31x y se arrastra al bus, donde debe aparecer el Gateway ya configurado, donde simplemente se le dá en Acoplar y seguidamente en la pestaña configuración se editan cada una de las direcciones de los bytes, relacionando la entrada de uno con la salida del otro, es decir, entrada de Maestro con salida de Gateway y viceversa, colocando las mismas direcciones para evitar confusiones a la hora de transferir la información (ver Figura 4.8)

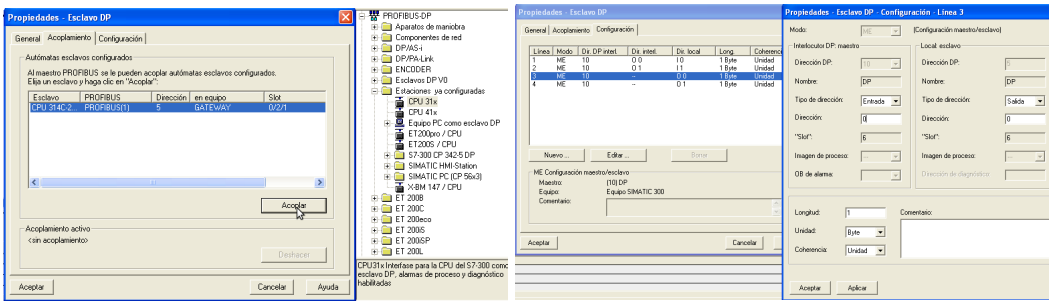


Figura 4.8: Conexión Maestro con Gateway en Profibus con Bytes.
Fuente: Elaboración propia.

Para concluir la configuración del Hardware del Gateway conectado al PLC maestro, se Aceptan todas los parámetros expuestos anteriormente y se guarda y compila la información del software Step 7 de Siemens (*SIMATIC Programming with STEP 7 V3.0*, 2004), quedando la conexión como se muestra en la Figura 4.9.

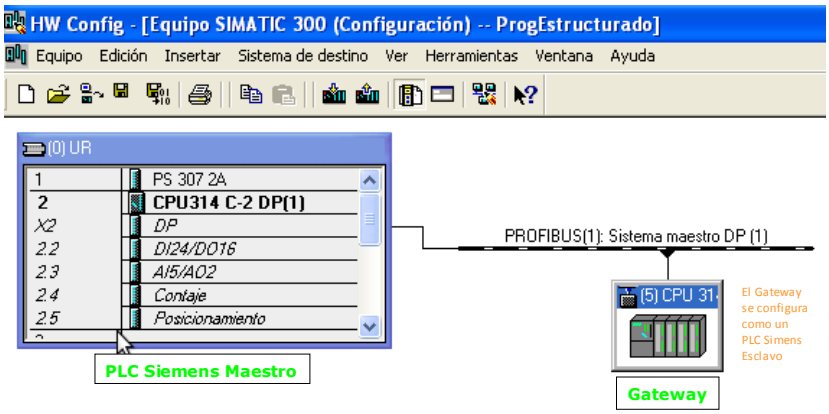


Figura 4.9: Conexión en bus Gateway con PLC maestro.
Fuente: Elaboración propia, imagen interfaz de red en Simatic Step 7.

Una vez realizadas estas configuraciones es posible conectar el Gateway físico al PLC mediante la red Profibus y como ya se tiene configurado el microcontrolador

para tomar o insertar los valores dentro de la trama Profibus, con esto se logra la transferencia de información hacia el módulo ZigBee y con este a la PDA mediante la red GSM y viceversa.

4.4. Inconvenientes Presentados y Soluciones

En esta fase 3 en donde se integraron las 2 fases expuestas en los capítulos anteriores, la unión necesita de toda una estructura de programación, estableciendo el puente de transferencia de información entre cada una de las tecnologías usadas, como son GSM, ZigBee y Profibus. Esta unión lleva consigo unos retos junto con un conjunto de problemas a solucionar, lo cual implica una reconstrucción en la estructura utilizada.

En la fase 1, inicialmente, se trabajó con un microcontrolador de un solo puerto USART y se hacia el suicheo entre el módulo GSM y el módulo ZigBee mediante un Relé, sin embargo teniendo en cuenta que esto generaba interrupciones en la comunicación y no se podían recibir 2 datos al mismo tiempo, se pasó a trabajar con un microcontrolador de 2 USART como se explica en la sección 2.3, el cual permite atender interrupciones independientes y simultáneas.

En la fase 2, uno de los aspectos más relevantes tuvo que ver con los niveles de voltaje dados en la comunicación Profibus, que aunque el estándar establece que solo vasta con tener voltajes diferenciales, para el diseño del Gateway se tuvo que trabajar no solo con el estándar RS485, sino también definir unos niveles de voltajes negativos, además de realizar de manera correcta el control de paridad y tener bien definidos los tiempos de transferencia en la trama Profibus entre tramas y entre bytes.

En el empalme de la fase 3, aunque aparentemente es simple, tiene muchos detalles que no se pueden dejar por fuera para que el funcionamiento del sistema sea lo más óptimo posible. Lo cual se divide en 3 aspectos fundamentales que tienen que ver con el envío de tramas de control desde la PDA al PLC, el otro aspecto está relacionado con el diagnóstico de fallas preciso del PLC enviando la información a la PDA y finalmente el aspecto de retorno de control realizado desde la PDA.

En el primer aspecto, relacionado con el envío de las tramas, de control, se comprobó que mediante un solo dígito como 1, 2, 3, se podían encender y apagar salidas del PLC desde la PDA, sin embargo, al verificar esto y montarlo sobre la trama Profibus, no se tenían condiciones óptimas de funcionamiento, puesto que solo se podían enviar bits y no bytes completos, además de que no existen condiciones de seguridad, y si en un mensaje existía el dígito enviado, este podía cambiar parámetros del PLC. Bajo esta condición, se buscó tener una estructura de sintaxis que lograra brindar seguridad a la transferencia de información dada entre la PDA y el PLC y además permitir el envío de bytes completos que no solo cambien salidas del PLC sino que permitan cargar datos en variables del

mismo como se hace desde una interfaz HMI como Labview o desde cualquier pantalla táctil. Esta estructura como se explicó anteriormente esta compuesta por la palabra control, el byte a controlar y la información del byte.

En el aspecto de diagnóstico de fallas, es necesario leer con detenimiento y continuamente la trama Profibus, que contiene la información enviada desde el PLC Maestro al Gateway Esclavo según como se haya programado. El tipo de falla que se presenta en el PLC debe buscar activar una salida del esclavo y esta activación debe ser analizada con mucha precisión y características de las misma, extrayendo la información de la trama Profibus. Para esto se toma el byte o los bytes de transferencia de información desde el PLC hacia el Gateway y mediante una AND realizada del byte recibido con cada una de las posiciones de los bits de información de falla, se construye un mensaje el cual es enviado a la PDA.

El tercer y ultimo aspecto que garantiza un óptimo monitoreo del sistema, es el retorno o validación de que la información se ha recibido correctamente, donde al mandar un mensaje de control, el sistema responde indicando "Control Realizado" como se vé en la Figura 4.3. El inconveniente surge cuando el Gateway envía dicho mensaje y el módulo GSM manda el SMS, el cual retorna un "OK" precedido de el dato en hexadecimal "0x0A"cuando el mensaje es enviado. Este "OK"y dato "0x0A", es como una información nueva para el Gateway, la cual se debe análizar con detenimiento para ver si corresponde a una trama de Control, sin embargo como esta no es una información valida se debe descartar y por consiguiente mantener los mismos valores de control de bytes que se establecieron en la palabra de control, enviando estos mismos de manera continua al PLC con la trama Profibus estructurada.

4.5. Validación y Resultados

En esta última sección, se muestran todos los resultados y pruebas realizadas, validando de esta forma los objetivos propuestos para este trabajo. De las misma manera se hacen pruebas de diseño de experimento, simulando diferentes aspectos que se pueden presentar en el ámbito industrial que permitan realizar control y monitoreo del sistema desde la PDA mediante la red ZigBee. Además de la estructura metodológica definida para el diagnóstico de fallas que se puedan presentar en un proceso industrial controlado por un PLC.

Inicialmente se realiza el control del byte enviando el dato en hexadecimal de "01" , lo que equivale a activar la entrada del bit 0 del byte de comunicación del Maestro, el cual por programación del PLC, se definió que equivale al Byte 1. Luego la entrada 1.0 se activa y con esta se ejercen acciones de control como activar una salida física del PLC que para la prueba realizada fue Q124.5 como se aprecia en la visualización en tiempo real del programa del PLC. De la misma manera, en el mismo programa del PLC, se transfiere la información del Byte 1 a una marca de memoria y al Byte de salida QB125 utilizando el bloque de transferencia MOVE, como se visualiza en la Figura 4.10, se observa que MB10 queda cargado

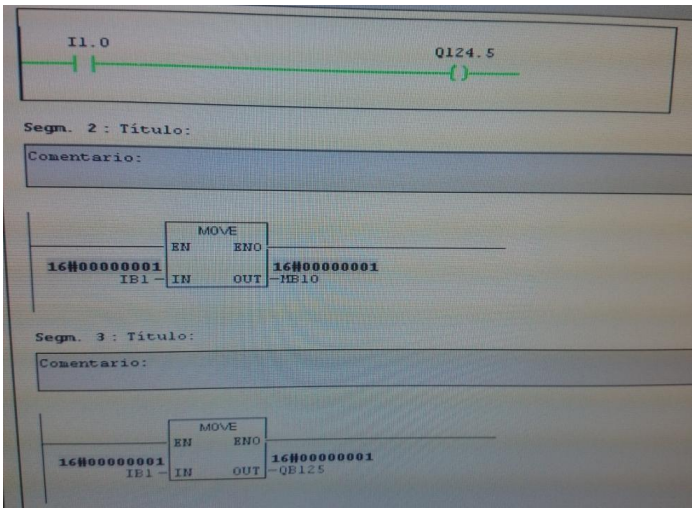


Figura 4.10: Visualización en tiempo real del Software.
Fuente: Elaboración propia, imagen tomada Step 7 PLC *online*.

con el bit 0 en 1 al igual que el byte Q125, el cual activa al bit 0 de esta salida. Estas activaciones de las salidas físicas Q124.5 y Q125.0 del PLC se observan con claridad en la Figura 4.11, donde se pueden observar las características de control que se pueden realizar desde la PDA hacia el PLC.

Otra de las validaciones realizadas, consiste en el envío de bytes completos que permiten inclusive cambiar parámetros con valores deseados transfiriendo un valor en Hexadecimal. Para esto se escribe en la palabra de control con la sintaxis establecida, el valor de "B2 " (Ver Figura 4.12), lo que permite mandar el valor del byte a la memoria MB10 y de la misma manera con el programa estructurado, al byte QB125. Luego teniendo en cuenta la equivalencia de "B2" en hexadecimal dada por la Tabla 4.2, permite poner en 1 y en 0 los bits correspondientes al dato en Hexadecimal de B2 como se observa claramente en las salidas físicas del PLC en el byte correspondiente al 125 de la Figura 4.13.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
1	0	1	1	0	0	1	0	B2

Tabla 4.2: Envío Byte de control xB2 para Trama Profibus.
Fuente: Elaboración propia

Figura 4.11: Salidas Activadas en PLC con dato 0x01.
Fuente: Elaboración propia, foto tomada de PLC con programa.

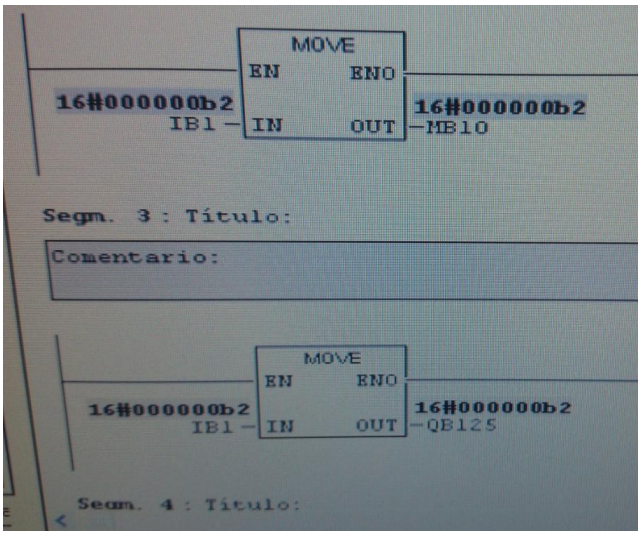


Figura 4.12: Visualización *Online* Software de PLC dato B2.
Fuente: Elaboración propia

Figura 4.13: Salidas activadas del PLC con dato B2.
Fuente: Elaboración propia, foto del PLC en *Run* con programa.

Para el diagnóstico de fallas que pueda presentar el PLC en su programa o alguna falla dada en un proceso industrial, estas se pueden definir con marcas de activación dadas dentro de un bloque funcional con condiciones de seguridad como se comentó anteriormente, o mediante marcas de memoria que se activen en el llamado de cada uno de los bloques OB encargados del diagnóstico de fallas del PLC.

Para la validación de esta parte funcional y verificación del envío de fallas del PLC a la PDA con la red ZigBee, se realizan pruebas de simulación de fallas forzando las marcas con la tabla de variables, las cuales se definieron en el programa como las MB80, es decir el byte 80. Dentro de este byte, se colocaron los primeros 4 bits en la tabla de variables (M80.0, M80.1, M80.2 y M80.3) los cuales se activan en el programa cambiándolos de "0" (false) a "1" (*true*) en el programa de pruebas del PLC. Este cambio, activa una bobina del esclavo con el byte 1 (Q1.0), que para este caso es el Gateway como se muestra en la Figura 4.14.

Esta activación es recibida por el Gateway, quien la lee, la decodifica y se la manda al módulo ZigBee, quien activa el bit de envío de mensajes que luego es recibido en la PDA. Cada falla se encuentra codificada en Hexadecimal, la cual maneja un valor según sea el tipo de falla, la cual se encuentra programada en el microcontrolador del Gateway, como se aprecia en la Tabla 4.3. Además el Gateway tiene la capacidad de enviar hasta las 8 fallas definidas por el byte, donde una activación de 2 o más bits dentro del byte son decodificados para mandar en un solo mensaje la falla o el conjunto de fallas presentadas, es decir, por cada bit o bits de activación se manda un mensaje como tal.

Gracias a que el PLC es el dispositivo de control actualmente más usado en las plantas de procesos industriales (Guerrero y cols., 2012), la integración de este con la tecnología ZigBee, permite una comunicación transversal a cualquier medio

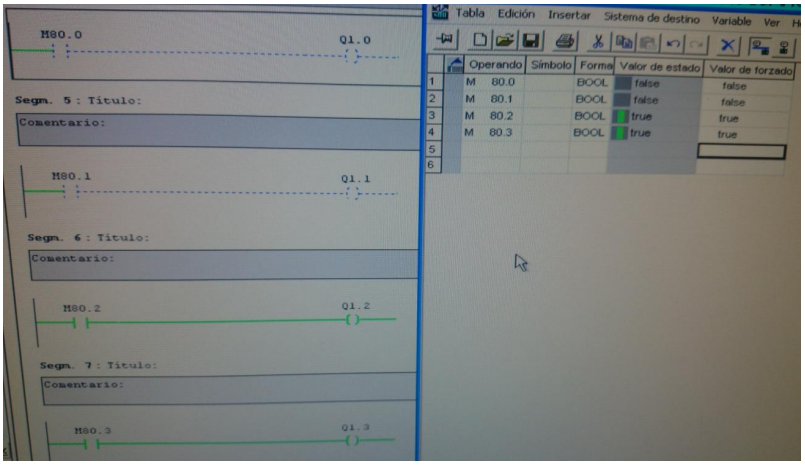


Figura 4.14: Visualización *Online* Tabla de Variables con Valores Forzados.
Fuente: Elaboración propia

Bit Activado en Hexadecimal	Mensaje a Enviar
0x01	Falla Comunicación
0x02	Falla de Ciclo de Tiempo
0x04	Falla módulo 1
0x08	Falla en CPU
0x10	Error en Programa
0x20	Error Accediendo al módulo
0x40	Reinicio en el sistema
0x80	Falla en módulo 2

Tabla 4.3: Detección de Fallas Codificadas.
Fuente: Elaboración propia

que se desee en el ámbito industrial, como sensores, equipos móviles, interfases, actuadores, entre otras. Con ZigBee se tiene una buena inmunidad al ruido en la comunicación inalámbrica, lo cual permite definir un perfil de estandarización con aplicaciones distribuidas siendo compatible para la interconexión de diferentes fabricantes.

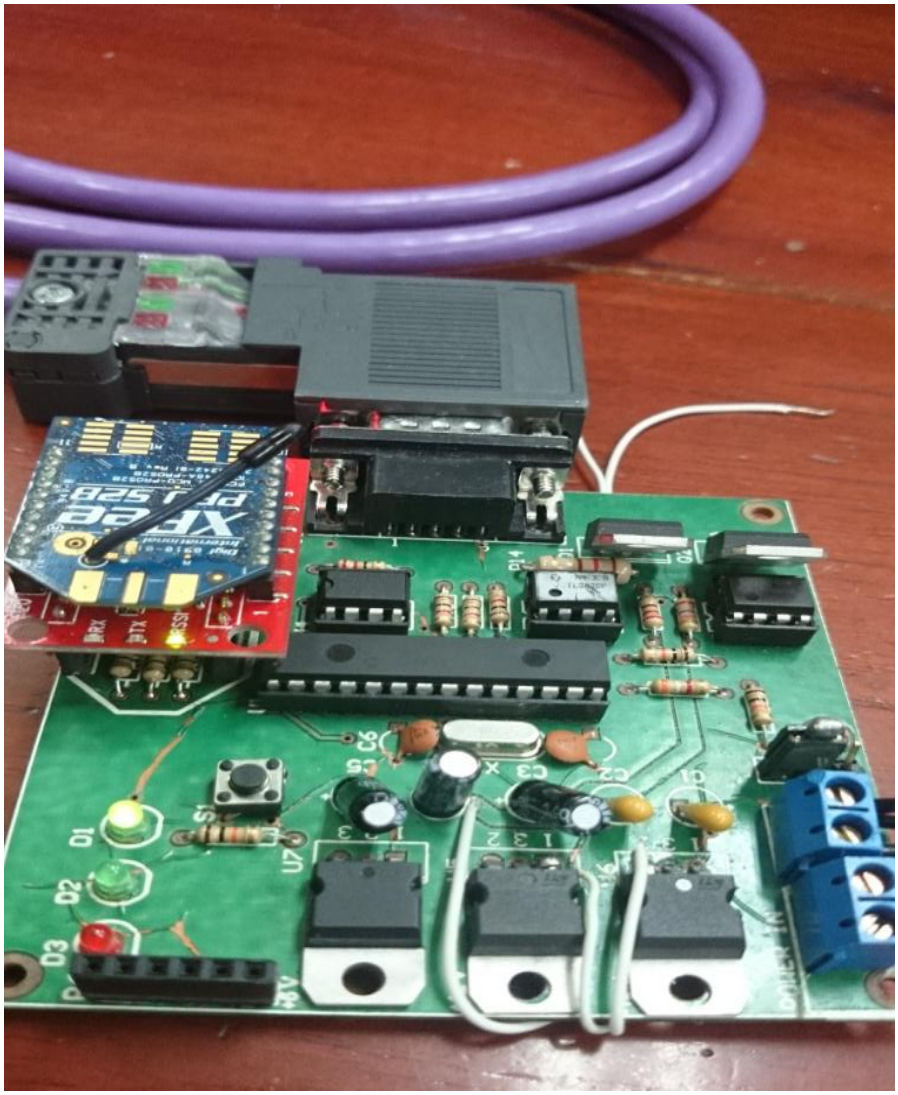


Figura 4.15: Gateway Diseñado Funcionando con módulo ZigBee.
Fuente: Elaboración propia, foto del PCB

Capítulo 5

Conclusiones y Lineas de trabajo futuro

La estructura metodológica empleada para el diagnóstico de fallas de un PLC, aunque básica, permite actuar de manera oportuna ante diferentes eventos presentados en la industria, puesto que los mensajes recibidos en la PDA del ingeniero de planta a través de la red ZigBee, permiten mejorar el rendimiento y nivel productivo de la empresa, definiendo el tipo de mantenimiento a realizar en tiempo real.

En el diseño del Gateway empleado para establecer el puente de comunicación entre el PLC y el módulo ZigBee, se debió de caracterizar correctamente los parámetros y variables del sistema que intervienen en el canal de comunicación Profibus. Para esto, se utilizaron técnicas de muestreó y análisis de tramas de datos dadas entre un PLC Maestro y un Microcontrolador Esclavo, haciendo compatibles las impedancias, niveles de voltaje y tiempos óptimos de respuesta, asegurando la comunicación, permitiendo identificar de manera adecuada, los parámetros que influyen en esta, haciendo operativo el sistema. La suplantación de un PLC con un dispositivo embebido, no es tan simple a la hora de implementarse, pues se necesita de un correcto análisis de las tramas Profibus recibidas y transmitidas.

Una comunicación efectiva dada entre el PLC y la PDA, permite una operación transparente y con cualquier otro tipo de dispositivo que se desee, debido a que con el dispositivo embebido se pueden encontrar un conjunto de posibilidades de operación, monitoreo y control, dependiendo únicamente de las características de programación y configuración realizadas, logrando la transversalidad sobre el protocolo de comunicación Profibus. La detección de fallas comunes, es solo un aspecto básico de lo que se puede tener como monitoreo, sin embargo, se pueden realizar inclusive interfases HMI en las PDA o Tablets con el transporte de datos y variables dadas entre estos 2 sistemas de comunicaciones utilizando las redes de tercera o cuarta generación. Entre las condiciones más relevantes que determinaron el correcto funcionamiento del canal de comunicación establecido con el PLC, indican que aunque Profibus esta dado por el estándar RS485 en su capa física,

las pruebas y características evaluadas demostraron que la implementación con el chip MAX485 no es fiable, presentando errores en el bus.

La validación utilizada que permite evaluar el funcionamiento de la metodología utilizada para el diagnóstico de fallas, control y monitoreo del PLC, detectó los retardos presentados por la red GSM, donde dichos tiempos, están condicionados al tipo de red celular utilizada, con su trafico circundante. Estos aspectos sumados a los tiempos entre tramas Profibus, al tiempo de procesamiento de datos en el Gateway, y la recepción y reenvío de información, afectan el rendimiento de comunicaciones entre la PDA y el PLC.

5.1. Lineas de trabajo futuras

De este trabajo de investigación, surgen muchos aspectos de aplicabilidad y mejoras con respecto a lo planteado en los objetivos, que aunque lo establecido se ha cumplido, se pueden tener muchas más funcionalidades y manejos de otros parámetros o fallas, cambiando lo básico por niveles más complejos.

Aunque algunas de las fallas presentadas por el PLC están dadas por sus bloques funcionales, también se tienen las memorias de registro del sistema, donde se puede hacer un diagnóstico del estado operativo del PLC o de los módulos del mismo así como del hardware de manera profunda. Ingresar remotamente a estos registros desde la PDA, puede ser de gran utilidad a nivel industrial, suprimiendo el uso de la PG o algún otro equipo de diagnóstico de PLCs que tienen alto costo y en algunas ocasiones no justifica su inversión.

Integrar la PDA con las redes de transporte de datos en generaciones superiores como UMTS, HSDPA o LTE, permiten hacer un control más completo y con características de un grado de complejidad mayor, logrando inclusive alcanzar mayores velocidades de respuesta y monitoreo de un conjunto de variables trabajando simultáneamente en un proceso de producción industrial.

Construir un app, que permita a la PDA o dispositivo móvil como Smart Phone, ser una Interfaz HMI, podría ser de gran utilidad para los ingenieros de planta, visualizando en tiempo real las principales variables del proceso e identificar algún problema que se pueda presentar, todo esto integrado con la red de datos 4G u otra utilizada, así como una integración con la red WiFi de una empresa.

Apéndices

Apéndice A

Configuración y Valores Módulos ZigBee

A.1. Parámetros ZigBee del Gateway-Enrutardor AT

En esta sección, se muestran los valores con los que queda configurado el módulo ZigBee que se encuentra en el Gateway, donde algunos se cambiaron parámetros y otros se dejaron por defecto. Inicialmente en la tabla A.1, se muestran cada uno de los menús con los submenús y en la figura A.1 se muestra la misma información, visualizada directamente desde la interfaz o aplicativo XCTU de Digi para configuración y programación de módulos ZigBee.

En esta configuración se trabaja el módulo como enrutador y control por comandos AT con la codificación establecida como se explica en el capitulo 2 sección 2.1.1. Esta parametrización es dada para el módulo ZigBee conectado al Gateway

Menú	Submenú	Valor o Submenú 2	Significado o Valor
Networking			
	ID-PAN ID	0	
	SC-Scan Channels	4000	
	SD-Scan Duration	3	
	ZS-ZigBee Stack Profile	0	
	NJ-Node Join Time	FF	
	NW-Network Watchdog Timeout	0	
	JV- Channel Verification	0	DISABLED
	JN-Join Notification	0	DISABLED
	OP-Operating PAN ID	123	
	OI-Operating 16-bit PAN ID	1CBC	
	CH-Operating Channel	19	
	NC-Number of Remaining Children	C	

Menú	Submenú	Valor o Submenú 2	Significado o Valor
Addressing			
	SH-Serial Number High	13A200	
	SL-Serial Number Low	405E0AF1	
	MY - 16-bit Network Address	D292	
	DH-Destination Address High	0	
	DL-Destination Address Low	0	
	NI-Node Identifier		Set
	NH-Maximum Hops	1E	
	BH-Broadcast Radius	0	
	AR-Many-to-One Route Broadcast Time	FF	
	DD-Device Type Identifier	30000	
	NT-Node Discovery Backoff	3C	
	NO-Node Discovery Options	0	
	NP-Maximum Number of Transmission Bytes	54	
	ZigBee Addressing		
		SE-ZigBee Source Endpoint	E8
		DE-ZigBee Destination Endpoint	E8
		CI-ZigBee Custer ID	11
RF Interfacing			
	PL-Power Level	4	HIGHEST
	PM-Power Mode	1	BOOST MODE ENABLED
	PP-Power at PL4	12	
Security			
	EE-Encryption Enable	0	DISABLED
	EO-Encryption Options	0	
	KY-Encryption Key		Set
Serial Interfacing			
	BD-Baud Rate	3	9600
	NB-Parity	0	NO PARITY
	SB-Stop Bits	0	ONE STOP BIT

Menú	Submenú	Valor o Submenú 2	Significado o Valor
	RO-Packetization Timeout	3	
	D7-DIO7 Configuration	1	CTS-FLOW CONTROL
	D6-DIO6 Configuration	0	DISABLE
AT Command Options			
	CT-AT Command Mode Timeout	64	
	GT-Guad Times	3E8	
	CC-Command Sequence Character	2B	
Sleep Modes			
	SM-Sleep Mode	0	NO SLEEP (ROUTER)
	SN-Number of Cyclic Sleep Periods	1	
	SO-Sleep Options	0	
	SP-Cyclic Sleep Period	20	
	ST-Time Before Sleep	1388	
I/O Settings	D0-AD0/DIO0 Configuration	1	COMMISSIO-NING BUTTON
	D1-AD1/DIO1 Configuration	0	DISABLED
	D2-AD2/DIO2 Configuration	0	DISABLED
	D3-AD3/DIO3 Configuration	0	DISABLED
	D4-DIO4 Configuration	0	DISABLED
	D5-DIO5/Assoc Configuration	1	ASSOCIATED INDICATOR
	P0-DIO10/PWM0 Configuration	1	RSSI-PWM OUTPUT
	P1-DIO11 Configuration	0	DISABLED
	P2-DIO12 Configuration	0	DISABLED
	PR-Pull-up Resistor Enable	1FFF	
	LT-Associate LED Blink Time	0	
	RP-RSSI PWM Timer	28	
	I/O Sampling		
		IR-IO Sampling Rate	0

Menú	Submenú	Valor o Submenú 2	Significado o Valor
		IC-Digital IO Change Detection	0
		V+-Supply Voltage High Threshold	0
Diagnostic Commands			
	VR-Firmware Version	228C	
	HV-Hardware Version	1E42	
	AI-Association Indication	0	
	DB-RSSI of Last Packet	0	
	%V-Supply Voltage	AC0	
	TP-Temperature	27	

Tabla A.1: Menú con Valores de configuración de ZigBee del Gateway.

Fuente: Elaboración propia a partir del software de configuración.

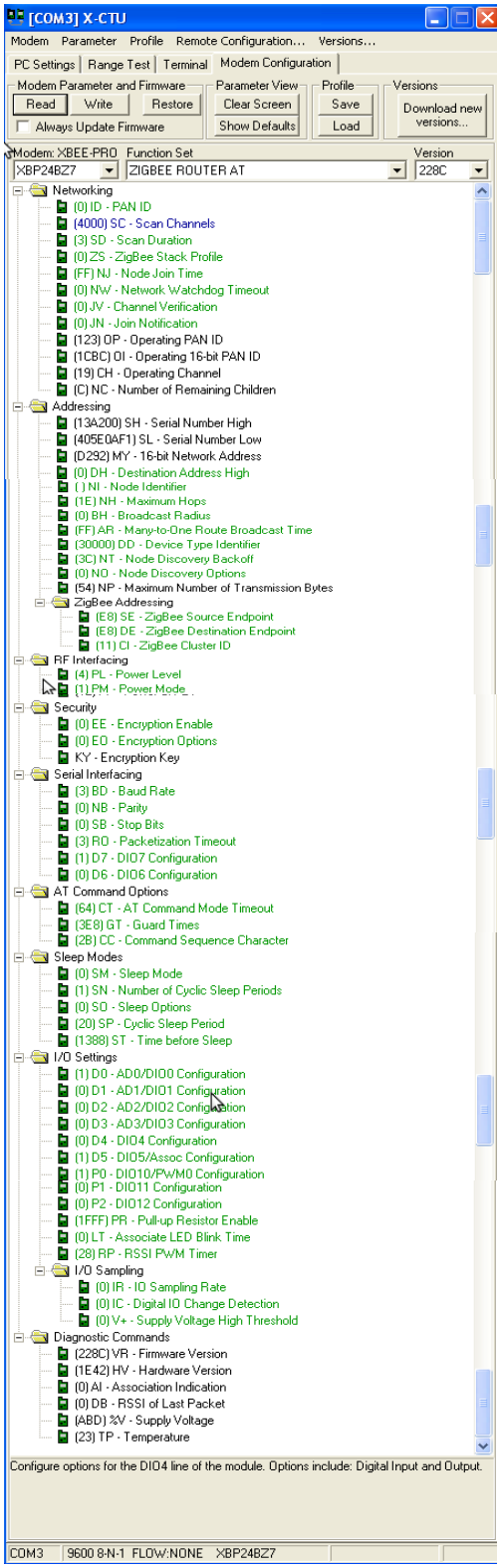


Figura A.1: Configuración del ZigBee del Gateway desde XCTU.
Fuente: Tomado de software de Configuración.

A.2. Parámetros ZigBee conectado al módulo GSM como Coordinador AT

En la tabla A.2, se muestran cada uno de los parámetros de configuración establecidos para el módulo ZigBee conectado al módulo GSM, el cual se encuentra configurado como Coordinador AT, explicado en el capitulo 2 sección 2.1.1, modificando únicamente los parámetros básicos, dejando los otros por defecto. De la misma manera en la figura A.2 se muestra, los parámetros de configuración del módulo ZigBee visto desde la terminal X-CTU propietaria del fabricante de los ZigBee.

Menú	Submenú	Valor o Submenú 2	Significado o Valor
Networking			
	ID-PAN ID	123	
	SC-Scan Channels	4000	
	SD-Scan Duration	3	
	ZS-ZigBee Stack Profile	0	
	NJ-Node Join Time	FF	
	OP-Operating PAN ID	123	
	OI-Operating 16-bit PAN ID	1CBC	
	CH-Operating Channel	19	
	NC-Number of Remaining Children	A	
Addressing			
	SH-Serial Number High	13A200	
	SL-Serial Number Low	406644C7	
	MY - 16-bit Network Address	0	
	DH-Destination Address High	13A200	
	DL-Destination Address Low	405E0AF1	
	NI-Node Identifier		Set
	NH-Maximum Hops	1E	
	BH-Broadcast Radius	0	
	AR-Many-to-One Route Broadcast Time	FF	
	DD-Device Type Identifier	30000	
	NT-Node Discovery Backoff	3C	
	NO-Node Discovery Options	0	
	NP-Maximum Number of Transmission Bytes	54	
	ZigBee Addressing		
		SE-ZigBee Source Endpoint	E8

Menú	Submenú	Valor o Submenú 2	Significado o Valor
		DE-ZigBee Destination Endpoint	E8
		CI-ZigBee Custer ID	11
RF Interfacing			
	PL-Power Level	4	HIGHEST
	PM-Power Mode	1	BOOST MODE ENABLED
	PP-Power at PL4	12	
Security			
	EE-Encryption Enable	0	DISABLED
	EO-Encryption Options	0	
	KY-Encryption Key		Set
	NK-Network Encryption Key		Set
Serial Interfacing			
	BD-Baud Rate	3	9600
	NB-Parity	0	NO PARITY
	SB-Stop Bits	0	ONE STOP BIT
	RO-Packetization Timeout	3	
	D7-DIO7 Configuration	1	CTS-FLOW CONTROL
	D6-DIO6 Configuration	0	DISABLE
AT Command Options			
	CT-AT Command Mode Timeout	64	
	GT-Guad Times	3E8	
	CC-Command Sequence Character	2B	
Sleep Modes			
	SP-Cyclic Sleep Period	20	
	SN-Number of Cyclic Sleep Periods	1	
I/O Settings	D0-AD0/DIO0 Configuration	1	COMMISSIO-NING BUTTON

Menú	Submenú	Valor o Submenú 2	Significado o Valor
	D1-AD1/DIO1 Configuration	0	DISABLED
	D2-AD2/DIO2 Configuration	0	DISABLED
	D3-AD3/DIO3 Configuration	0	DISABLED
	D4-DIO4 Configuration	0	DISABLED
	D5-DIO5/Assoc Configuration	1	ASSOCIATED INDICATOR
	P0-DIO10/PWM0 Configuration	1	RSSI-PWM OUTPUT
	P1-DIO11 Configuration	0	DISABLED
	P2-DIO12 Configuration	0	DISABLED
	PR-Pull-up Resistor Enable	1FFF	
	LT-Associate LED Blink Time	0	
	RP-RSSI PWM Timer	28	
	I/O Sampling		
		IR-IO Sampling Rate	0
		IC-Digital IO Change Detection	0
		V+-Supply Voltage High Threshold	0
Diagnostic Commands			
	VR-Firmware Version	208C	
	HV-Hardware Version	1E46	
	AI-Association Indication	0	
	DB-RSSI of Last Packet	0	
	%V-Supply Voltage	AE4	
	TP-Temperature	21	

Tabla A.2: Menú con Valores de configuración de ZigBee conectado al módulo GSM.
Fuente: Elaboración propia a partir del software de configuración.

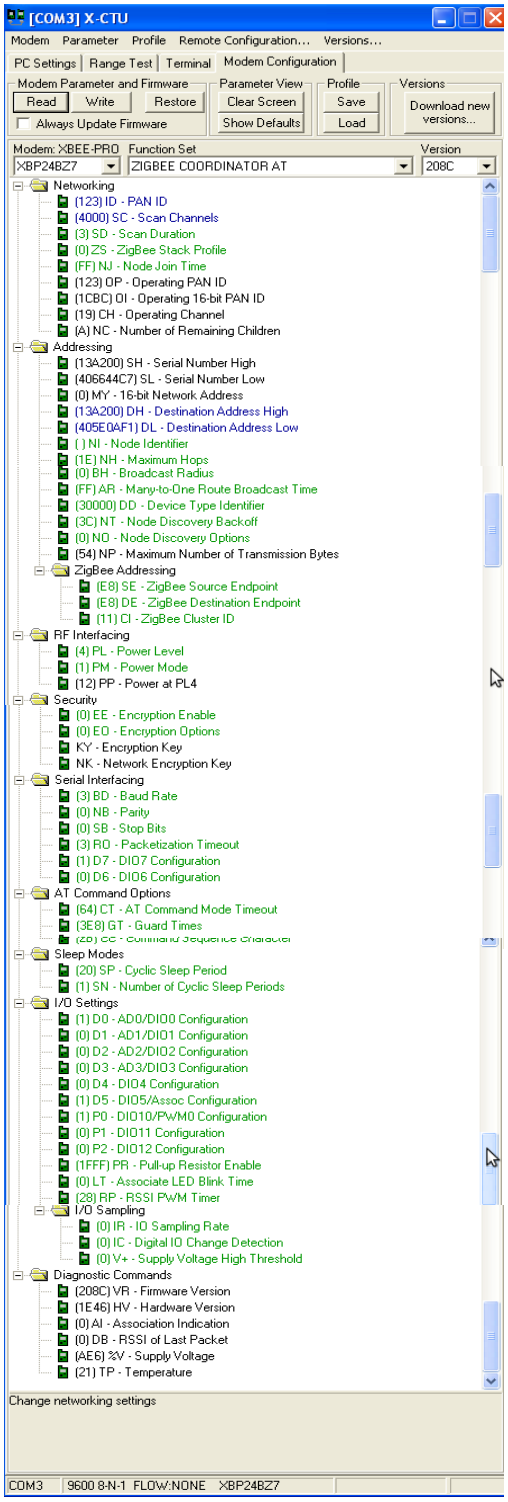


Figura A.2: Configuración del ZigBee conectado al módulo GSM visto desde XCTU.

Fuente: Tomado de software de Configuración.

A.2.1. Opciones de Configuración de algunos parámetros del módulo ZigBee

El módulo ZigBee, en algunos de sus parámetros de configuración, tienen varias opciones, indicadas por dígitos, donde cada dígito establece las opciones que permite el parámetro, los cuales se encuentran en el menú de *RF Interfacing* lo cual se observa en la tabla A.3.

Submenú	Opciones	Función del Parámetro
PL-Power Level	0	LOWEST
	1	LOW
	2	MEDIUM
	3	HIGH
	4	HIGHEST
PM-Power Mode	0	BOOST MODE DISABLED
	1	BOOST MODE ENABLED

Tabla A.3: Opciones de Configuración con diferentes parámetros en el menú de *RF Intefacing*.

Fuente: Elaboración propia a partir del software de configuración.

En el menú de seguridad solo se puede cambiar la configuración en el sub-menú de *Encryption Enable* (EE) donde si está en 0 es Deshabilitado (*DISABLED*) y si está en 1 es Habilitado (*ENABLED*). Mientras que en *Serial Interfacing* si existen varias opciones las cuales se tienen en la tabla A.4.

Finalmente las otras opciones de configuración del módulo ZigBee cambiando los parámetros por dígitos, se dá en el menú I/O *Settings*, donde se configuran cada uno de los pines del módulos ZigBee donde según sean las características de estos se pueden configurar más o menos parámetros y pines que tienen la misma función su configuración es la misma lo cual se muestra en la tabla A.5.

Submenú	Opciones	Función del Parámetro
BD-Baud Rate	0	1200
	1	2400
	2	4800
	3	9600
	4	19200
	5	38400
	6	57600
	7	115200
NB-Parity	0	NO PARITY
	1	EVEN PARITY
	2	ODD PARITY
	3	MARK PARITY
SB-Stop Bits	0	ONE STOP BIT
	1	TWO STOP BIT
D7-DIO7 Configuration	0	DISABLE
	1	CTS FLOW CONTROL
	2	NA
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH
	6	RS-485 ENABLE LOW
	7	RS-485 ENABLE HIGH
D6-DIO6 Configuration	0	DISABLE
	1	RTS FLOW CONTROL
	2	NA
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH

Tabla A.4: Opciones de Configuración con diferentes parámetros en el menú de *Serial Intefacing*.

Fuente: Elaboración propia a partir del software de configuración.

Submenú	Opciones	Función del Parámetro
D0-AD0 DIO0 Configuration	0	DISABLED
	1	COMMISSIONING BUTTON
	2	ADC
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH
D1-AD1 DIO1 Configuration D2-AD2 DIO2 Configuration D3-AD3 DIO3 Configuration	0	DISABLED
	1	NA
	2	ADC
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH
D4-DIO4 Configuration P1-DIO11 Configuration P2-DIO12 Configuration	0	DISABLED
	1	NA
	2	NA
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH
D5-DIO5/Assoc Configuration	0	DISABLED
	1	ASSOCIATED INDICATOR
	2	NA
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH
P0-DIO10/PWM0 Configuration	0	DISABLED
	1	RSSI PWM OUTPUT
	2	NA
	3	DIGITAL INPUT
	4	DIGITAL OUT, LOW
	5	DIGITAL OUT, HIGH

Tabla A.5: Opciones de Configuración de los Pines del ZigBee con diferentes parámetros según dígito.

Fuente: Elaboración propia a partir del software de configuración.

Apéndice B

Código Comunicador GSM-ZigBee

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <p18f26k22.h>
#include <string.h>
#include <delays.h>
#include <usart.h>


    // CONFIG1L
#pragma config FOSC = INTIO67 // Oscillator Selection bits (Internal oscillator block)
#pragma config PLLCFG = ON // 4X PLL Enable (Oscillator multiplied by 4)
#pragma config PRICLKEN = OFF // Primary clock enable bit (Primary clock can be disabled by software)
#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

    // CONFIG2L
#pragma config PWRTEN = OFF // Power-up Timer Enable bit (Power up timer disabled)
#pragma config BOREN = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)
#pragma config BORV = 190 // Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)

    // CONFIG2H
#pragma config WDTEN = OFF // Watchdog Timer Enable bits (Watch dog timer is always disabled. SWDTEN has
no effect.)
#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

    // CONFIG3H
#pragma config CCP2MX = PORTC1 // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<5:0>pins are configured as digital I/O on Reset)
#pragma config CCP3MX = PORTC6 // P3A/CCP3 Mux bit (P3A/CCP3 input/output is mulitplexed with RC6)
#pragma config HFOFST = OFF // HFINTOSC Fast Start-up (HFINTOSC output and ready status are not delayed
by the oscillator stable status)
#pragma config T3CMX = PORTC0 // Timer3 Clock input mux bit (T3CKI is on RC0)
#pragma config P2BMX = PORTC0 // ECCP2 B output mux bit (P2B is on RC0)
#pragma config MCLRE = INTMCLR // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

    // CONFIG4L
#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)
#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addres-
sing mode disabled (Legacy mode))

    // CONFIG5L
#pragma config CP0 = OFF // Code Protection Block 0 (Block 0 (000800-003FFFh) not code-protected)
#pragma config CP1 = OFF // Code Protection Block 1 (Block 1 (004000-007FFFh) not code-protected)
#pragma config CP2 = OFF // Code Protection Block 2 (Block 2 (008000-00BFFFh) not code-protected)
#pragma config CP3 = OFF // Code Protection Block 3 (Block 3 (00C000-00FFFFh) not code-protected)

    // CONFIG5H
#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)
```

```
// CONFIG6L
#pragma config WRT0 = OFF // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)
#pragma config WRT1 = OFF // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)
#pragma config WRT2 = OFF // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)
#pragma config WRT3 = OFF // Write Protection Block 3 (Block 3 (00C000-00FFFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh)
not write-protected)
#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF // Table Read Protection Block 0 (Block 0 (000800-003FFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR1 = OFF // Table Read Protection Block 1 (Block 1 (004000-007FFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR2 = OFF // Table Read Protection Block 2 (Block 2 (008000-00BFFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR3 = OFF // Table Read Protection Block 3 (Block 3 (00C000-00FFFFh) not protected from table
reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not protected
from table reads executed in other blocks)

//—————————————————————————————————————————————-

#define _XTAL_FREQ 16000000
#define enablePeripInt PEIE = 1 //habilita las interrupciones de los perifericos
#define enableIntRecUsart RC1IE = 1
#define enableIntRecUsart2 RC2IE = 1
#define enableIntRxUsart GIE=1

void configUSART(void);
void enviostring1(const char *);
void enviostring2(const char *);

char RxData1[ 55 ] = '0' ;
char RxData2[ 55 ] = '0' ;
char TramaControl1[9]=0x43,0x6F,0x6E,0x74,0x72,0x6F,0x6C,0x31,0x0D;
char usartSt = 0x00; //registro de estado del modulo usart
char conta = 0x00; //puntero para el buffer de recepcion
char conta2= 0x00;
int i=0;
int j=0;
int bandera;
int ciclo;

void interrupt isrInt( void )
{
   if( RC1IF==1)
   {
      if( RCREG1 != 0x0D)
      {
       RxData1[ conta ] = RCREG1;
       conta++;
      }
      else
      {
       RCSTA1bits.CREN = 0;
       bandera=1;
      }
   }
   if( RC2IF==1 )
   {
      if( RCREG2 == '\r' )
      {
       RxData2[ conta2 ] = '\0'; //final de los datos recibidos
       conta2 = 0; //incializa el puntero del arreglo
      }
      else
      {
       RxData2[ conta2 ] = RCREG2; //carga los datos en el buffer
       conta2++; //incrementa la posicion para llenar el buffer
          if( conta2 == 54 )
          {
           RxData2[ conta2 ] = '\n';
           conta2++;
```

```
        }
     }
    }
}

void configUSART()
{
    RCSTA1bits.SPEN = 1; //habilito el modulo de comunicacion
    RCSTA1bits.CREN = 1; //habilitamos la recepcion por el modulo usart
    TXSTA1 = 0x24; //habilito el transmisor con BRGH=1
    BAUDCON1bits.BRG16=1;
    SPBRG1 = 0B10100000; //velocidad 9600 baudios con reloj de 64MHz BRG16=1 y BRGH=1
    SPBRGH1=0B00000001;
    RC1IE = 1; //habilito la interrupcion por recepcion del serial
    TXSTA1bits.TXEN=1;
    TXSTA1bits.SYNC=0;

    RCSTA2bits.SPEN = 1; //habilito el modulo de comunicacion
    RCSTA2bits.CREN = 1; //habilitamos la recepcion por el modulo usart
    TXSTA2 = 0x24; //habilito el transmisor con BRGH=1
    BAUDCON2bits.BRG16=1;
    SPBRG2 = 0B10100000; //velocidad 9600 baudios con reloj de 64MHz BRG16=1 y BRGH=1 SPBRGH2= 0B00000001;
    RC2IE = 1; //habilito la interrupcion por recepcion del serial
    TXSTA2bits.TXEN=1;
    TXSTA2bits.SYNC=0;
}

void enviostring1( const char *ptrsString )
{
    while( *ptrsString != '\0' )
    {
     TXREG1 = *ptrsString; //carga el dato a enviar al puerto
     ptrsString++; //incremento del puntero
     while( !TXSTA1bits.TRMT );
    }
}
void enviostring2( const char *ptrsString )
{
    while( *ptrsString != '\0' )
    {
     TXREG2 = *ptrsString; //carga el dato a enviar al puerto
     ptrsString++; //incremento del puntero
     while( !TXSTA2bits.TRMT );
    }
}

void main(void)
{
    OSCCON=0B01110010; //Bit 2 del OSCCON indica estabilidad o no del HFINTOSC
    OSCTUNE=0B11011111;
    OSCCON2=0B10000000;
    bandera=0;

    ADON=0;
    ADCON1=0x0F;
    Delay10KTCYx(255);
    Delay10KTCYx(255);
    TRISC=0B11001111;
    TRISB=0B10000000;
    ANSELA=0xE0;
    ANSELB=0x00;
    ANSELC=0x00;
    TRISA=0xFF;
    LATA=0x00;
    LATB=0x00;
    LATC=0x00;

    configUSART();
    PEIE = 1; //habilita las interrupciones de los perifericos
    ei();
    conta=0;
    conta2=0;

    Delay10KTCYx(255);
    enviostring1("\r");
    Delay10KTCYx(255);
    Delay10KTCYx(255);
    Delay10KTCYx(255);
    enviostring1("B \r");
    Delay10KTCYx(255);
    PORTBbits.RB1=1;

    for (i=0;i<=55;i+=1)
    {
```

```
 RxData1[ i ] = '\0';
 RxData2[ i ] = '\0';
}

while (ciclo==0)
{
 conta=0;
 conta2=0;
 enviostring2("AT\r");
 Delay10KTCYx(255);
 Delay10KTCYx(255);
 for (i=0;i<=10;i++)
 {
     if(RxData2[i]=='O')
     {
      if(RxData2[i+1]=='K')
      {
       Delay10KTCYx(255);
       PORTBbits.RB1= PORTBbits.RB1;
       enviostring2("AT+CMGF=1\r");
       Delay10KTCYx(255);
       enviostring2("AT+CMGD=1\r");
       Delay10KTCYx(255);
       enviostring2("AT+CNMI=2,2,0,0,0\r");
       Delay10KTCYx(255);
       enviostring2("ATE0\r");
       Delay10KTCYx(255);
       ciclo=1;
      }
     }
 }
}

while (1)
{
 if (bandera==1)
 {
     Delay10KTCYx(255);
     enviostring2("AT+CMGS=\"3007842390\"\r");
     Delay10KTCYx(255);
     enviostring2("Alarma1PLC \r");
     Delay10KTCYx(255);
     enviostring2(RxData1);
     Delay10KTCYx(255);
     TXREG2=0x1A;
     Delay10KTCYx(255);
     PORTBbits.RB2= PORTBbits.RB2;
     j=0;
     i=0;
     RCSTA1bits.CREN = 1;
     bandera=0;
     conta=0;
     for (i=0;i<=55;i+=1)
     {
      RxData1[ i ] = '\0';
     }
 }
 if (RxData2[conta2]!='\0')
 {
     if(RxData2[conta2]!='O')
     {
      if (RxData2[conta2+1]!='K')
      {
       Delay10KTCYx(255);
       enviostring1(RxData2);
       Delay10KTCYx(255);
       PORTBbits.RB1= PORTBbits.RB1;
       Delay10KTCYx(255);
       enviostring2("AT+CMGD=1\r");
       Delay10KTCYx(255);
      }
     }
 }
 else
 {
     if(RxData2[conta2+1]=='K')
     {
      enviostring2("AT\r");
     }
 }
}
}
```

Apéndice C

Código Gateway PLC-ZigBee via Profibus

```
#include <stdio.h>
#include <stdlib.h>
#include <p18f26k22.h>
#include <string.h>
#include <xc.h>
#include <delays.h>
#include <usart.h>


// CONFIG1H
#pragma config FOSC = HSHP // Oscillator Selection bits (Internal oscillator block)
#pragma config PLLCFG = ON // 4X PLL Enable (Oscillator multiplied by 4)
#pragma config PRICLKEN = OFF // Primary clock enable bit (Primary clock can be disabled by software)
#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

    // CONFIG2L
#pragma config PWRTEN = OFF // Power-up Timer Enable bit (Power up timer disabled)
#pragma config BOREN = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)
#pragma config BORV = 190 // Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)

    // CONFIG2H
#pragma config WDTEN = OFF // Watchdog Timer Enable bits (Watch dog timer is always disabled. SWDTEN has
no effect.)
#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

    // CONFIG3H
#pragma config CCP2MX = PORTC1 // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<5:0>pins are configured as digital I/O on Reset)
#pragma config CCP3MX = PORTC6 // P3A/CCP3 Mux bit (P3A/CCP3 input/output is mulitplexed with RC6)
#pragma config HFOFST = OFF // HFINTOSC Fast Start-up (HFINTOSC output and ready status are not delayed
by the oscillator stable status)
#pragma config T3CMX = PORTC0 // Timer3 Clock input mux bit (T3CKI is on RC0)
#pragma config P2BMX = PORTC0 // ECCP2 B output mux bit (P2B is on RC0)
#pragma config MCLRE = EXTMCLR // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

    // CONFIG4L
#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)
#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addres-
sing mode disabled (Legacy mode))

    // CONFIG5L
#pragma config CP0 = OFF // Code Protection Block 0 (Block 0 (000800-003FFFh) not code-protected)
#pragma config CP1 = OFF // Code Protection Block 1 (Block 1 (004000-007FFFh) not code-protected)
#pragma config CP2 = OFF // Code Protection Block 2 (Block 2 (008000-00BFFFh) not code-protected)
#pragma config CP3 = OFF // Code Protection Block 3 (Block 3 (00C000-00FFFFh) not code-protected)

    // CONFIG5H
#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)
```

```
// CONFIG6L
#pragma config WRT0 = OFF // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)
#pragma config WRT1 = OFF // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)
#pragma config WRT2 = OFF // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)
#pragma config WRT3 = OFF // Write Protection Block 3 (Block 3 (00C000-00FFFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh)
not write-protected)
#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF // Table Read Protection Block 0 (Block 0 (000800-003FFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR1 = OFF // Table Read Protection Block 1 (Block 1 (004000-007FFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR2 = OFF // Table Read Protection Block 2 (Block 2 (008000-00BFFFh) not protected from table
reads executed in other blocks)
#pragma config EBTR3 = OFF // Table Read Protection Block 3 (Block 3 (00C000-00FFFFh) not protected from table
reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not protected
from table reads executed in other blocks)

//————————————————————————————————————————-

#define _XTAL_FREQ 64000000
#define enablePeripInt PEIE = 1 //habilita las interrupciones de los perifericos
#define enableIntRecUsart RC1IE = 1
#define enableIntRecUsart2 RC2IE = 1
#define enableIntRxUsart GIE=1

void configUSART(void);
void funparidad(void);
void enviostring2(const char *);
void enviofalla();

char RxData[ 55 ] = { '0' };
char RxData2[ 55 ] = { '0' };
char buftx[255]={'0'};
char buf1[55] = { '0' };
char buf2[55] = { '0' };
char buf3[55] = { '0' };
char buf4[55] = { '0' };
char buf5[55] = { '0' };
char buf6[55] = { '0' };

char Trama1MaesTip[15]={0xDC,0x0A,0x0A,0x68,0x05,0x05,0x68,0x85,0x8A,0x6D,0x3C,0x3E,0xF6,0x16, 0x0D};
char TramaRx1TipEscla[18]={0x68,0x0B,0x0B,0x68,0x8A,0x85,0x08,0x3E,0x3C,0x02,0x05,0x00,0xFF,0x80,
0xD1,0xE8,0x16,0x0D};
char Trama2MaesTipConEnca[25]={0xDC,0x0A,0x0A,0x68,0x0F,0x0F,0x68,0x85,0x8A,0x5D,0x3D,0x3E,0xB8,
0x41,0x42,0x36,0x80,0xD1,0x00,0xC0,0x60,0x00,0xC9,0x16,0x0D};
char TramaRx2OK[2]={0xE5, 0x0D};
char Trama3MaesTip[50]={0xDC,0x0A,0x0A,0x68,0x28,0x28,0x68,0x85,0x8A,0x7D,0x3E,0x3E,0x04,0x00,0x00,
0xAD,0xC4,0x04,0x00,0x00,0x8B,0x41,0x04,0x00,0x00,0x8F,0xC0,0x83,0x00,0x00,0x93,0x40,0x43,0x00,0x00,0x83,
0x40,0x83,0x00,0x00,0x93,0x40,0x43,0x00,0x00,0x83,0x40,0x58,0x16,0x0D};
char Trama4MaesTip[15]={0xDC,0x0A,0x0A,0x68,0x05,0x05,0x68,0x85,0x8A,0x5D,0x3C,0x3E,0xE6,0x16,0x0D};
char TramaRx3TipEscla[26]={0x68,0x13,0x13,0x68,0x8A,0x85,0x08,0x3E,0x3C,0x02,0x05,0x00,0xFF,0x80,
0xD1,0x42,0x00,0x06,0x82,0x00,0x00,0x00,0x00,0xB2,0x16,0x0D};
char Trama5MaesTip[15]={0xDC,0x0A,0x0A,0x68,0x05,0x05,0x68,0x05,0x0A,0x7D,0x00,0x00,0x8C,0x16,0x0D};
char TramaRx4TipEscla[12]={0x68,0x05,0x05,0x68,0x0A,0x05,0x08,0x00,0x00,0x17,0x16,0x0D};
char Trama6MaesTip[15]={0xDC,0x0A,0x0A,0x68,0x05,0x05,0x68,0x05,0x0A,0x5D,0x00,0x00,0x6C,0x16,0x0D};

char usartSt = 0x00; //registro de estado del modulo usart
char conta = 0x00; //puntero para el buffer de recepcion
char conta2= 0x00;
char plcdato;
int i=0;
int j=7,k=8,l=0;
int t=0,u=0,w=0,x=0,y=0,z=0,p=2,q=2;
int paridad;
int par;
int bandera;
char otracosa;
int ciclocontrol;
```

```
void interrupt isrInt( void )
{
    if( RC1IF==1)
    {
        if( RCREG1 != 0x16)
        {
            RxData[ conta ] = RCREG1; //carga los datos en el buffer
            buf1[conta]=RxData[conta] ^Trama1MaesTip[conta];
            buf2[conta]=RxData[conta] ^Trama2MaesTipConEnca[conta];
            buf3[conta]=RxData[conta] ^Trama3MaesTip[conta];
            buf4[conta]=RxData[conta] ^Trama4MaesTip[conta];
            buf5[conta]=RxData[conta] ^Trama5MaesTip[conta];
            buf6[conta]=RxData[conta] ^Trama6MaesTip[conta];
            conta++;
            bandera=0;
        }
        else
        {
            RCSTA1bits.CREN = 0;
            for(i=conta;i>=0;i–)
            {
                if(buf1[i]==0)
                    t=t+1;
                if(buf2[i]==0)
                    u=u+1;
                if(buf3[i]==0)
                    w=w+1;
                if(buf4[i]==0)
                    x=x+1;
                if(buf5[i]==0)
                    y=y+1;
                if(buf6[i]==0)
                    z=z+1;
            }
            bandera=1;
            if(w>24)
            {
                t=0;
                y=0;
                u=0;
                v=0;
                s=0;
                x=0;
                z=0;
            }
        }
    }
    if( RC2IF==1 )
    {
        if( RCREG2 == '\r' )
        {
            RxData2[ conta2 ] = '\0'; //final de los datos recibidos
            conta2 = 0; //incializa el puntero del arreglo
        }
        else
        {
            RxData2[ conta2 ] = RCREG2; //carga los datos en el buffer
            if (RCREG2=='1')
            {
                enviostring2(Çontrol 1 realizado ñ);
                ciclocontrol=1;
            }
            conta2++; //incrementa la posicion para llenar el buffer
            if( conta2 == 54 )
            {
                RxData2[ conta2 ] = '\n';
                conta2++;
            }
        }
    }
}

void configUSART()
{
    RCSTA1bits.SPEN = 1; //habilito el modulo de comunicacion
    RCSTA1bits.CREN = 1; //habilitamos la recepcion por el modulo usart
    TXSTA1 = 0x64; //habilito el transmisor CON BRGH=1
    BAUDCON1bits.BRG16=1;
    SPBRG1 = 0B10000010; //velocidad 9600 baudios con reloj de 64MHz BRG16=1 y BRGH=1
    SPBRGH1=0B00000110;
    RC1IE = 1; //habilito la interrupcion por recepcion del serial
    TXSTA1bits.TXEN=1;
    TXSTA1bits.SYNC=0;
    RCSTA1bits.RX9=1;
```

```
    RCSTA2bits.SPEN = 1; //habilito el modulo de comunicacion
    RCSTA2bits.CREN = 1; //habilitamos la recepcion por el modulo usart
    TXSTA2 = 0x24;
    BAUDCON2bits.BRG16=1;
    SPBRG2 = 0B10000010; //velocidad 9600 baudios con reloj de 64MHz BRG16=1 y BRGH=1
    SPBRGH2= 0B00000110;
    RC2IE = 1; //habilito la interrupcion por recepcion del serial
    TXSTA2bits.TXEN=1;
    TXSTA2bits.SYNC=0;
}

void enviostring2( const char *ptrsString )
{
    while( *ptrsString != '\0' )
    {
        TXREG2 = *ptrsString; //carga el dato a enviar por el puerto
        ptrsString++; //incremento del puntero
        while( !TXSTA2bits.TRMT );
    }
}

void funparidad()
{
    paridad=0;
    par=0x01;
    j=0;
    while (j<=7)
    {
        if ((otracosa & par)==par)
        {
            paridad=paridad+1;
        }
        par=par<<1;
        j=j+1;
    }
    if (paridad==0)
        TXSTA1bits.TX9D=0;
    if (paridad==1)
        TXSTA1bits.TX9D=1;
    if (paridad==2)
        TXSTA1bits.TX9D=0;
    if (paridad==3)
        TXSTA1bits.TX9D=1;
    if (paridad==4)
        TXSTA1bits.TX9D=0;
    if (paridad==5)
        TXSTA1bits.TX9D=1;
    if (paridad==6)
        TXSTA1bits.TX9D=0;
    if (paridad==7)
        TXSTA1bits.TX9D=1;
    if (paridad==8)
        TXSTA1bits.TX9D=0;
}

void enviofalla()
{
    checkfalla=0x01;
    for(j=0;j<=7;j++)
    {
    and2=RxData[11] & checkfalla;
        if(and2==0x01)
            enviostring2("Falla Comunicacion \n");
        if(and2==0x02)
            enviostring2("Falla ciclo tiempo \n");
        if(and2==0x04)
            enviostring2("Falla modulo 1 \n");
        if(and2==0x08)
            enviostring2("Falla en CPU \n");
        if(and2==0x10)
            enviostring2(.Error en programa \n");
        if(and2==0x20)
            enviostring2(.Error accediendo al modulo \n");

    checkfalla=checkfalla << 1;
    }
    enviostring2("\r");
}


void main(void)
{
    OSCCON=0B01111000; //Bit 2 del OSCCON indica estabilidad o no del HFINTOSC
```

```
OSCTUNE=0B11011111;
OSCCON2=0B00000000;

ADON=0;
ADCON1=0x0F;
Delay10KTCYx(255);
Delay10KTCYx(255);
TRISC=0B11001111;
TRISB=0B10000000;
ANSELA=0xE0;
ANSELB=0x00;
ANSELC=0x00;
TRISA=0xFF;
LATA=0x00;
LATB=0x00;
LATC=0x00;

PORTB=0x00;
configUSART();
PEIE = 1; //habilita las interrupciones de los perifericos
ei();
conta=0;

for (i=0;i<=55;i+=1)
{
    RxData[ i ] = '0';
    RxData2[i]='0';
    buf1[i]='0';
    buf2[i]='0';
    buf3[i]='0';
    buf4[i]='0';
    buf5[i]='0';
    buf6[i]='0';
}
conta2=0;
Delay10KTCYx(255);
enviostring2("\r");
Delay10KTCYx(255);
Delay10KTCYx(255);
Delay10KTCYx(255);
enviostring2("B \r");
Delay10KTCYx(255);

conta=0;
PORTBbits.RB3=1;
PORTCbits.RC4=1;

while (1)
{
    if(PORTAbits.RA4)
    {
        PORTCbits.RC4=0;
        PORTCbits.RC4=1;
        PORTBbits.RB1= PORTBbits.RB1;
    }
    PORTBbits.RB3= PORTBbits.RB3;
    if (bandera==1)
    {
        if(t==13)
        {
            Delay10KTCYx(9);
            PORTCbits.RC4=0;
            if (vectorparidad[l]==0x01)
                TXSTA1bits.TX9D=1;
            else
                TXSTA1bits.TX9D=0;
            TXREG1 = TramaRx1TipEscla[l],TXSTA1bits.TX9D;
            if (vectorparidad[l+1]==0x01)
                TXSTA1bits.TX9D=1;
            else
                TXSTA1bits.TX9D=0;
            l++;
                while (TramaRx1TipEscla[l] != 0x0D)
                {
                    while(TX1IF==0)
                    {
                        TXREG1 = TramaRx1TipEscla[l],TXSTA1bits.TX9D;
                        if (vectorparidad[l]==0x01)
                            TXSTA1bits.TX9D=1;
                        else
                            TXSTA1bits.TX9D=0;
                        l++;
                    }
                }
```

```
    PORTBbits.RB0= PORTBbits.RB0;
    Delay1KTCYx(37);
    PORTCbits.RC4=1;
    conta=0;
    t=0;
    Delay10KTCYx(9);
}
if(u==23)
{
    Delay10KTCYx(9);
    PORTCbits.RC4=0;
    Delay1KTCYx(1);
    TXSTA1bits.TX9D=1;
    TXREG1 = 0xE5,TXSTA1bits.TX9D;
    PORTBbits.RB0= PORTBbits.RB0;
    Delay1KTCYx(37);
    PORTCbits.RC4=1;
    conta=0;
    u=0;
    Delay10KTCYx(9);
}
if(w>24)
{
    Delay10KTCYx(9);
    PORTCbits.RC4=0;
    Delay1KTCYx(1);
    TXSTA1bits.TX9D=1;
    TXREG1 = 0xE5,TXSTA1bits.TX9D;
    PORTBbits.RB0= PORTBbits.RB0;
    Delay1KTCYx(1);
    PORTCbits.RC4=1;
    conta=0;
    w=0;
    Delay10KTCYx(9);
}
if(x==13)
{
    Delay10KTCYx(9);
    PORTCbits.RC4=0;
    Delay1KTCYx(1);
    if (vectorparidad3[l]==0x01)
        TXSTA1bits.TX9D=1;
    else
        TXSTA1bits.TX9D=0;
    TXREG1 = TramaRx3TipEscla2[l],TXSTA1bits.TX9D;
    if (vectorparidad3[l+1]==0x01)
        TXSTA1bits.TX9D=1;
    else
        TXSTA1bits.TX9D=0;
    l++;
        while (TramaRx3TipEscla2[l] != 0x0D)
        {
            while(TX1IF==0)
            {
                TXREG1 = TramaRx3TipEscla2[l],TXSTA1bits.TX9D;
                if (vectorparidad3[l]==0x01)
                    TXSTA1bits.TX9D=1;
                else
                    TXSTA1bits.TX9D=0;
                l++;
            }
        }
    PORTBbits.RB0= PORTBbits.RB0;
    Delay1KTCYx(37);
    PORTCbits.RC4=1;
    conta=0;
    x=0;
    Delay10KTCYx(9);
}
if(y==13)
{
    if(ciclocontrol==1)
    {
        TramaRx4TipEscla[8]=0x01;
        TramaRx4TipEscla[9]=0x18;
        vectorparidad4[8]=0x01;
        vectorparidad4[9]=0x00;
        Delay10KTCYx(9);
        PORTCbits.RC4=0;
        Delay1KTCYx(1);
        if (vectorparidad4[l]==0x01)
            TXSTA1bits.TX9D=1;
        else
            TXSTA1bits.TX9D=0;
```

```
            TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
            if (vectorparidad4[l+1]==0x01)
                TXSTA1bits.TX9D=1;
            else
                TXSTA1bits.TX9D=0;
            l++;
                while (TramaRx4TipEscla[l] != 0x0D)
                {
                    while(TX1IF==0)
                    {
                        TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
                        if (vectorparidad4[l]==0x01)
                            TXSTA1bits.TX9D=1;
                        else
                            TXSTA1bits.TX9D=0;
                        l++;
                    }
                }
            PORTBbits.RB0= PORTBbits.RB0;
            Delay1KTCYx(37);
            PORTCbits.RC4=1;
            conta=0;
            y=0;
            Delay10KTCYx(9);
        }
        else
        {
            TramaRx4TipEscla[8]=0x00;
            TramaRx4TipEscla[9]=0x17;
            vectorparidad4[8]=0x00;
            vectorparidad4[9]=0x00;
            Delay10KTCYx(9);
            PORTCbits.RC4=0;
            Delay1KTCYx(1);
            if (vectorparidad4[l]==0x01)
                TXSTA1bits.TX9D=1;
            else
                TXSTA1bits.TX9D=0;
            TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
            if (vectorparidad4[l+1]==0x01)
                TXSTA1bits.TX9D=1;
            else
                TXSTA1bits.TX9D=0;
            l++;
                while (TramaRx4TipEscla[l] != 0x0D)
                {
                    while(TX1IF==0)
                    {
                        TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
                        if (vectorparidad4[l]==0x01)
                            TXSTA1bits.TX9D=1;
                        else
                            TXSTA1bits.TX9D=0;
                        l++;
                    }
                }
            PORTBbits.RB0= PORTBbits.RB0;
            Delay1KTCYx(37);
            PORTCbits.RC4=1;
            conta=0;
            y=0;
            Delay10KTCYx(9);
        }
    }
    if(z==13)
    {
        Delay10KTCYx(9);
        PORTCbits.RC4=0;
        Delay1KTCYx(1);
        if (vectorparidad4[l]==0x01)
            TXSTA1bits.TX9D=1;
        else
            TXSTA1bits.TX9D=0;
        TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
        if (vectorparidad4[l+1]==0x01)
            TXSTA1bits.TX9D=1;
        else
            TXSTA1bits.TX9D=0;
        l++;
            while (TramaRx4TipEscla[l] != 0x0D)
            {
                while(TX1IF==0)
                {
                    TXREG1 = TramaRx4TipEscla[l],TXSTA1bits.TX9D;
```

```
                    if (vectorparidad4[l]==0x01)
                        TXSTA1bits.TX9D=1;
                    else
                        TXSTA1bits.TX9D=0;
                    l++;
                }
            }
        PORTBbits.RB0= PORTBbits.RB0;
        Delay1KTCYx(37);
        PORTCbits.RC4=1;
        conta=0;
        z=0;
        Delay10KTCYx(9);
    }

    conta=0;
    t=0;
    u=0;
    w=0;
    x=0;
    y=0;
    z=0;
    l=0;
    for (i=0;i<=55;i+=1)
    {
        RxData[i]='0';
        buf1[i]='0';
        buf2[i]='0';
        buf3[i]='0';
        buf4[i]='0';
        buf5[i]='0';
        buf6[i]='0';
    }
    i=0;
    bandera=0;
    RCSTA1bits.CREN = 1;
    enviofalla();
  }
 }
}
```

Bibliografía

Al-Ubaydli, M., y Paton, C. (2005, noviembre). The doctor's PDA and smartphone handbook personal digital assistant. *Journal of the Royal Society of Medicine*, *98*(11), 494–495. Disponible en http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1275996/ 56

Bates, R. J. (2003). *Comunicaciones inalámbricas de banda ancha.* McGraw-Hill Profesional. 39

Chan, H., y Ozguner, U. (1994, junio). Closed-loop control of systems over a communication network with queues. En *American control conference, 1994* (Vol. 1, pp. 811–815 vol.1). 1, 10

Figueiredo, J., y Costa, J. da. (2007, octubre). A concept for an operational management system for industrial purposes. En *IEEE international symposium on intelligent signal processing, 2007. WISP 2007* (pp. 1–6). 1

Gaeta, R., Bobbio, A., Franceschinis, G., y Portinale, L. (2001). Dependability assessment of an industrial programmable logic controller via parametric fault-tree and high level petri net. En *9th international workshop on petri nets and performance models, 2001. proceedings* (pp. 29–38). 2, 3

García, J. C. M., Ma Pilar, Castillo, J. C. M., y García, M. P. G. (2009). *Automatismos industriales.* Editex. 11

González Daniela, V. S. (2009). *Trabajo de investigacion de mercados ilc.* Universidad de Manizalez, Scribd. Disponible en http://es.scribd.com/doc/15855540/Trabajo-de-Investigacion-de-Mercados-2 3, 67

Guerrero, V., Yuste, R. L., y Martínez, L. (2012). *Comunicaciones industriales.* Alfaomega. 2, 14, 15, 107

Huidobro, J. M. (2002). *Todo sobre comunicaciones.* Paraninfo / Thomson Learning. 37

Hyde, J., Cuspinera, A., y Regué, J. (1997). *Control electroneumático y electrónico.* Marcombo. 11

Kikuchi, J., Takeo, K., y Kosuge, K. (1998). Teleoperation system via computer network for dynamic environment. En *ICRA* (pp. 3534–3539). IEEE Computer Society. Disponible en http://dblp.uni-trier.de/db/conf/icra/icra1998-4.html#KikuchiTK98 1, 56

López, J. d. D. S. (2002). *Dispositivos electrónicos de potencia.* UABC. 78

Moya, J. M. H. (2004). *Manual de telecomunicaciones.* Alfaomega. 38

MPLAB® XC compilers - compilers | microchip technology inc. (2012). Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, Arizona, USA. Disponible en http://www.microchip.com/pagehandler/en_us/devtools/mplabxc/ 54

Nieto, J. M. (2009). *Presentacion javier mejía nieto congreso ingeniería mecánica eafit l?* [Business]. Disponible en http://www.slideshare.net/mejianietojavier/presentacion-javier-meja-nieto-congreso-ingeniera-mecnica-eafit-la-eficacia 2, 3, 41

Ocampo Daniel, S. S. (2010). Seguridad del hogar mediante integración de una red de área personal zigbee con la red de telefonía móvil celular gsm. En *Segundo congreso virtual, microcontroladores y sus aplicaciones.* Universidad Tecnológica de Paraná. Argentina. 29

Palluat, N., Racoceanu, D., y Zerhouni, N. (2006, agosto). A neuro-fuzzy monitoring system application to flexible production systems. *Comput. Ind.*, *57*(6), 528?538. Disponible en http://dx.doi.org/10.1016/j.compind.2006.02.013 2, 3

Park, D.-H., Ku, T.-Y., Lee, K., y Moon, K.-D. (2007, junio). Design and implementation of QoS guaranteed bridge system for high speed PLC and UWB. En *IEEE international symposium on consumer electronics, 2007. ISCE 2007* (pp. 1–6). 2, 56, 98

Pengfei, L., y Jiakun, L. (2009, diciembre). Application of communication and remote control in PLC based on ZigBee. En *International conference on computational intelligence and security, 2009. CIS '09* (Vol. 2, pp. 533–536). 1, 29

PIC18f26k22 - 8-bit PIC® microcontrollers. (2012). Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, Arizona, USA. Disponible en http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC18F26K22 50

Profibus installation guidelines. (2009). PROFIBUS Competency Centre Automation Systems Centre Manchester Metropolitan University Revision 7.2. Disponible en http://www.sci-eng.mmu.ac.uk/ascent/literature/documents/installationguide.pdf 77

Profibus rs 485-is user and installation guideline. (June, 2003). PROFIBUS Nutzerorganisation e.V.Haid-und-Neu-Str. 7 D-76131 Karlsruhe. Disponible en http://www.procentec.com/downloads/others/RS485-IS_User_Inst_2262_V11_jun03.pdf 77

Profibus specification normative parts of profibus-fms, -dp,-pa accordin to the european standard en 50 170 [Entry1998Topic]. (1998). Volume 2. Disponible en https://www.kuebler.com/PDFs/Feldbus_Multiturn/specification_DP.pdf 15, 18, 22, 85

Proteus 7.7 sp2. labcenter electronics ltd. (2009). Build 9089 - Advanced Simulation, 53-55 Main Street Grassington North Yorkshire BD23 5AA England. Disponible en http://www.labcenter.com 80

Pérez, E. M., Acevedo, J. M., y Silva, A. Q. J., Celso Fernández. (2006). *Autómatas*

programables. entorno y aplicaciones. Thomson Editores. 12, 14

Ramos, G. (1998). *Los plc.* Electronica y Computadores. 11, 12, 13

S.A, M. (1998). *Telecomunicaciones móviles.* Marcombo. 37

Salazar Jairo, L. R. (2008). Lectura y monitoreo remoto de contadores de electricidad via bluetooth. En *Sitec.* Universidad Tecnológica de Pereira. 1, 54

Scanning devices inc. | lexington, MA 02420. (2009). Industrial I/O server. Disponible en http://www.scanningdevices.com/ 54

Siemens global website [Entry2013Topic]. (s.f.). Disponible en http://www.siemens.com/answers 12

Sim908 hardware design v1.01. (2011). A company of SIM Tech, SIMCom Wireless Solutions Ltd. Disponible en http://wm.sim.com/upfile/201264161410f.pdf 46, 47

Simatic programming with step 7 v3.0. (2004). Information and Download Center - Industry Automation and Drive Technologies (en) - Siemens. Disponible en http://http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objID=10805383&subtype=133300 13, 25, 102

Sp, D. W., y Williams, D. H. (2003). *PDA robotics: Using your personal digital assistant to control your robot.* McGraw Hill Professional. 56

Tip4xc - silicon power transistor, fairchild semiconductor. (December, 2009). Fairchild Semiconductor Corporation 82 Running Hill Road South Portland, ME 04106 U.S.A. Disponible en http://www.fairchildsemi.com/ds/TI/TIP42.pdf 78

Tl082cp - texas instruments® operational amplifiers. (Abril, 2013). Executive Offices 12500 TI Boulevard Dallas, Texas 75243 USA. Disponible en http://www.ti.com/lit/ds/symlink/tl082-n.pdf 78

Universal controller SIMATIC s7-300 - PLCs [WCMS3Portfolio]. (2011). - Siemens, CPU 31xC and CPU 31x: Installation Operating Instructions, 03/2011, A5E00105492-12 Copyright © Siemens AG Technical data subject to change Industry Sector Postfach 48 48 90026 NÜRNBERG GERMANY. Disponible en http://w3.siemens.com/mcms/programmable-logic-controller/en/simatic-s7-controller/s7-300/Pages/Default.aspx 85

Wu, P.-H., Lin, C.-C., Kuo, C.-H., y Wu, P.-L. (2006, julio). Design and implementation of the PLC control lab using GSM system. En *Sixth international conference on advanced learning technologies, 2006* (pp. 393–395). 1

Wu, R.-C., Wu, H.-C., Teng, J.-H., y Huang, C.-C. (2007, octubre). The realization of PLC wireless remote graphic control by PDA. En *TENCON 2007 - 2007 IEEE region 10 conference* (pp. 1–4). 1, 56

Wu, Y., y Shao, P. (2012, enero). Middleware-based distributed data acquisition and control in smart home networks. En F. L. Gaol y Q. V. Nguyen (Eds.), *Proceedings of the 2011 2nd international congress on computer applications and computational science* (pp. 219–226). Springer Berlin

Heidelberg. Disponible en http://link.springer.com/chapter/10.1007/978-3-642-28308-6_30 1, 29, 74

XBee® ZB - digi international. (s.f.). World Headquarters 11001 Bren Road East Minnetonka, MN 55343. Disponible en http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#specs 42

Xu, M., Xu, P., y Xu, J. (2012, enero). Design of industrial PLC based on ZigBee protocol. En M. Zhao y J. Sha (Eds.), *Communications and information processing* (pp. 54–61). Springer Berlin Heidelberg. Disponible en http://link.springer.com/chapter/10.1007/978-3-642-31968-6_7 1, 29

Zigbee alliance. (s.f.). ZigBee Alliance Interest Group, USA. Disponible en http://www.zigbee.org 2, 29

Zuluaga Juan Gonzalo, S. S. I., Herrera Jorge A. (2014, mayo). Interconectividad del bus profibus de un controlador logico programable para detección de fallas. *Tercer Congreso Virtual, Microcontroladores y sus Aplicaciones*, *3*(COM102). Disponible en 190.114.222.38/file.php/2/Ponencias/03-COM-CON/COM102.pdf?forcedownload=1 83

Publicaciones que ha generado este trabajo

Zuluaga J.G., Herrera J.A., Serna S.I. *Interconectividad del Bus Profibus de un Controlador Lógico Programable para detección de fallas.* Tercer Congreso Virtual, Microcontroladores y sus Aplicaciones. Universidad Tecnológica Nacional Facultad Regional Paraná, Entre Rios. Argentina, Mayo del 2014. url: www.microvirtual.org. COM102

Zuluaga J.G., Herrera J.A., Serna S.I. *Gateway for Profibus Communication between a PLC and Embedded System.* STSIVA 2014. XIX Simposio Internacional de Tratamiento de Señales, Imágenes y Visión Artificial. Armenia, Q., Colombia. 2014. ID173.

Zuluaga J.G., Herrera J.A., Serna S.I. *Interfaz ZigBee-PDA para aplicaciones en los procesos industriales.* Revista Ingenio Magno. Universidad Santo Tomás, Seccional Tunja, Colombia. ISSN: 2145-9282.