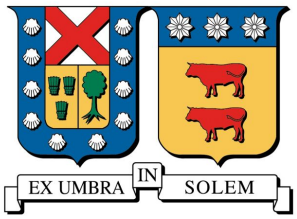
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"DESARROLLO DE UN MÓDULO DE COMUNICACIÓN FIELDBUS HART ESCLAVO PARA SENSORES INDUSTRIALES"

CRISTÓBAL MARCELO RAMÍREZ PÜSCHEL

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO, MENCIÓN COMPUTADORES

PROFESOR GUÍA: AGUSTÍN GONZÁLEZ.

PROFESOR CORREFERENTE: MANUEL OLIVARES.

ENERO - 2016

Resumen

El objetivo de este trabajo es contar con un prototipo de módulo de comunicación HART para sensores industriales. El sensor se conecta por cable USB al módulo de comunicación. Se entrega una biblioteca desarrollada en lenguaje C++ para sistemas Linux.

Para el desarrollo del módulo de comunicación se siguieron las hojas de especificaciones HART. Estas especificaciones indican un modelo OSI conformado por capa física, capa de enlace de datos y capa de aplicación. El protocolo especifica las funciones de cada una de estas capas y nombra servicios para acceder a cada una de ellas. Con este mecanismo es que cada capa puede comunicarse con la capa inferior.

Se utilizó un modem HART comercial para la modulación y demodulación de la señal física sobre el lazo de corriente $4 - 20[mA]$. Se desarrolló un firmware cumpliendo con el diseño de capas especificado por HART. El microcontrolador se conectó por UART al modem HART y por USB al computador con sistema Linux que tiene los datos del sensor.

Se probó el funcionamiento del módulo de comunicación HART con dispositivo HART comercial que actúa como maestro en la red HART y un programa que emplea la biblioteca desarrollada. No se pudo obtener comunicación de extremo a extremo por falla de modem HART. De todas maneras, por el software empleado por el adaptador USB-HART se observa que las demás capas son compatibles entre el dispositivo comercial y el desarrollo presente en este trabajo.

Es necesario reemplazar el modem HART para poder comunicar ambos extremos de la red.

Abstract

The objective of this work is to have a prototype of a HART communication module for industrial sensors. The sensor is connected via USB cable to the communication module. It is provide a library developed in C ++ to be used in Linux systems to communicate to the HART module.

The HART specification were used to develope the communication module. These specifications indicate an OSI model composed only of a physical layer, a data link layer and a application layer. The protocol specifies the functions of each of these layers and the services to access each of them. With this mechanism each layer can communicate with the lower one.

A commercial HART modem was used for the modulation and demodulation of the physical signal over the $4-20[mA]$ current loop. A firmware was developed that implements the layer specified by HART. It was burned to a microcontroller. This microcontroller was connected by UART to the HART modem and via USB to the computer with Linux system that has the sensor data.

The operation of the HART communication module with a commercial HART device that acts as master and a program that uses the developed library was tested. The end-to-end communication from HART modem failed. However, the software used by the comercial HART device shows that the other layers are compatible between the commercial device and the development presented in this work.

It is necessary to replace the HART modem in order to communicate both ends of the network correctly.

Glosario

Fieldbus : Red que comunica un sensor con la red de control.

Field device : Sensor capaz de conectarse en un fieldbus.

multi-drop : Red que soporta varios equipos en mismo bus.

burst-mode : Esclavo HART capaz de enviar datos sin que maestro lo solicite.

OSI : Open System Interconnection o modelo de interconexión de sistemas abiertos.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

Con la aparición de la automatización en la producción industrial surge la necesidad de conectar los sensores y los actuadores mediante una red que permita su comunicación. Es así como surgen los buses de campo y los dispositivos de campo que las conforman. Por esto, surgen en primera instancia redes análogas de comunicación como el lazo de corriente 4-20[mA] o el lazo de 0-10[V]. Luego, se desarrollan los protocolos digitales o buses de campo inteligentes, los que permiten beneficios como conectar más de un dispositivo en el bus de campo, aumento en el ancho de banda para la transmisión de la señal de medición o actuación, y agregar funcionalidades de diagnóstico de estos dispositivos a las redes de automatización.

Los reportes de uso de buses de campo a nivel mundial varían. La cantidad de dispositivos conectados a través de un bus llega a 69 millones [2]. Mientras que la cantidad de dispositivos de campos inteligentes llegan a los 54 millones [2]. Hay aumento año a año en la cantidad de dispositivos de campo compatibles con los buses de campo digitales. Se observa el caso de una empresa, a la cual implementar HART le significo reducción de alrededor de 4 veces en costos de mantención y diagnóstico.

Se desarrolló un prototipo de módulo de comunicación HART con motivo de comunicar el densímetro nuclear desarrollado en los laboratorios SiLab del CCTVAL en la Universidad Técnica Federico Santa María (UTFSM), con una red HART sobre una lazo de 4-20[mA]. El desarrollo involucra la programación de un firmware que cumpla con las especificación de capa de enlace [3] y capa de aplicación [4][5]. Este firmware fue

quemado a un microcontrolador ATMEGA32U4 conectado al modem HART DS8500 por UART y con un computador con sistema Linux por USB. El modem demodula la señal y la transmite a 1200[b/s] por UART al microcontrolador, que ejecuta protocolo HART en su firmware y responde al bus de campo con la última medición actualizada por el sensor o actualizando el valor de la actuación en el programa del sistema Linux.

El prototipo de módulo de comunicación HART comprobó su grado de funcionamiento conectándolo a un dispositivo HART comercial usado para diagnosticar dispositivos de campo y comprobar funcionamiento correcto de implementaciones del protocolo.

Capítulo 2

Automatización de procesos en la industria

Para la realización de automatización de procesos es necesaria la implementación de sensores y actuadores dirigidos a informar y controlar las variables físicas de interés. Sensores y actuadores con funcionamiento remoto a los dispositivos de control fueron llamados dispositivos de campo o field devices en inglés. Tradicionalmente no se les asocia con ninguna capacidad de procesamiento, pero sí de interacción electromecánica y de comunicación. Con el uso de este tipo de dispositivos es que se originan sistemas de comunicación análogos y luego los buses de campo para sustituir estos medios análogos por redes digitales.

Un bus de campo, del inglés fieldbus, es un sistema de transmisión de información que conecta dispositivos de campo con unidades recopiladoras de datos y/o equipos de control. Esto simplifica la instalación y operación de máquinas y equipamientos industriales utilizados en procesos de producción.

Los buses de datos que permiten la integración de equipos de medición y control de proceso reciben la denominación genérica de buses de campo. El objetivo de los buses de campo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control a través del tradicional lazo de corriente de 4-20mA o 0 a 10V DC. Son redes digitales, bidireccionales, multipunto (multi-drop), que conectan dispositivos de campo como PLCs, transductores, actuadores, sensores y equipos de mantención y

diagnóstico.

Varios grupos han intentado generar e imponer una norma que permita la integración de equipos de distintos proveedores. Sin embargo, hasta la fecha no existe un bus de campo universal [9]. En la industria nacional son tres los protocolos principalmente utilizados: HART (Highway Addressable Remote Transducer), Profibus (Process Fieldbus), Fieldbus Foundation.

2.1. Impacto de HART en la industria

El diagnóstico y mantención de las máquinas y dispositivos industriales es fundamental para el funcionamiento de una planta. Permite evitar tiempos muertos y apagos por urgencia. Con buses de campo análogos no se obtienen beneficios para diagnosticar los dispositivos industriales. Por el contrario, con buses de campo digitales tales como HART, se agregan capacidades de encuestar a los dispositivos presentes en el bus de campo desde sistemas centralizados o con dispositivos de mano conectados a ese mismo bus.

Las capacidades de diagnóstico que implementa HART reportan reducción de costos de mantención en grandes empresas del orden de 3 veces (ver figura 2.1). Esto se debe a la capacidad de mantención temprana que ofrece una red HART u otros bus de campo moderno, disminuyendo tiempos muertos de la planta al evitar fallas graves y menores, que paren una cadena de producción.

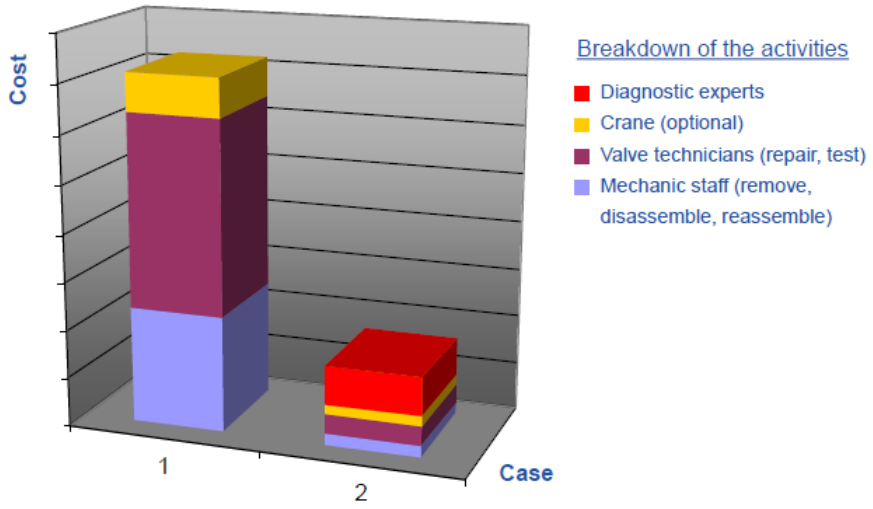


Figura 2.1: Costos de diagnóstico para dos casos de estudio [10].

Los beneficios de implementar una red de control y diagnóstico brindan competitividad a una planta, ya que permite reducir por un lado la cantidad de personal y, por otro, aumentar la eficiencia de los nuevos dispositivos que son capaces de usar estas redes. Estos beneficios permiten satisfacer la demanda de sofisticación de las plantas mediante el aumento del número de sensores que obtengan datos de los procesos de automatizados o diagnostiquen la maquinaria instalada [10].

2.2. Presencia de HART en la industria

Son variadas las fuentes que reportan la presencia de dispositivos de campo en la industria y el porcentaje de participación de los distintos protocolos usados para buses de campo inteligentes. La referencia principal son los estudios realizados por ARC Advisory Group [11] y son los que se revisan a continuación.

En la figura 2.3 se observa que el bus de campo más usado es HART, seguido por Fieldbus (Fieldbus Foundation y PROFIBUS) y luego varios sistemas propietarios distintos. Sin embargo, quedan cerca de un 36 % de instrumentos no incluidos a buses de campo inteligentes. Esto implica que los sistemas no están siendo monitoreados desde un sistema centralizado, lo que lleva a un aumento de gastos de mantención debido a varios factores como se ve en la sección 2.1. Esto da lugar a que HART y otros buses de campo sigan creciendo como se observa en figura 2.2. Esto se complementa con la petición

de incluir comunicación HART, junto con Profibus, para el proyecto de desarrollo del densímetro nuclear del SiLab, solicitud hecha desde Codelco para incorporar el prototipo de este sensor en sus redes de comunicación HART y Profibus.

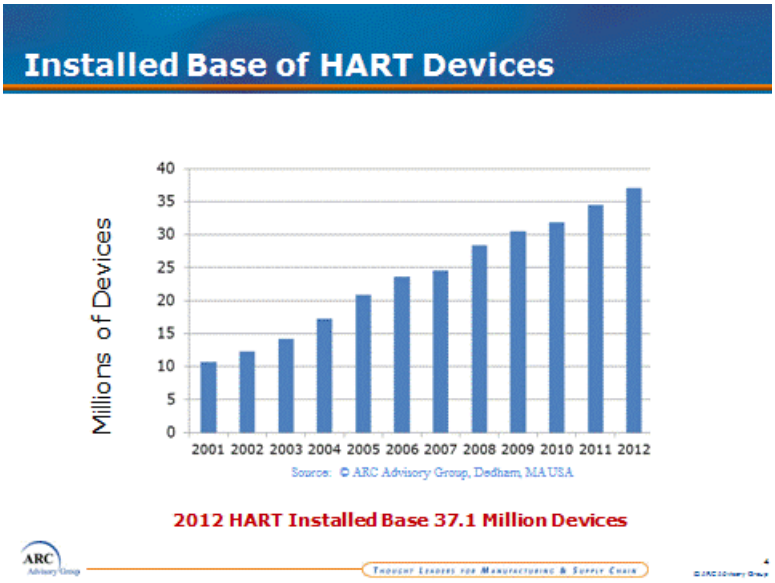


Figura 2.2: Dispositivos capaces de comunicación HART [12].

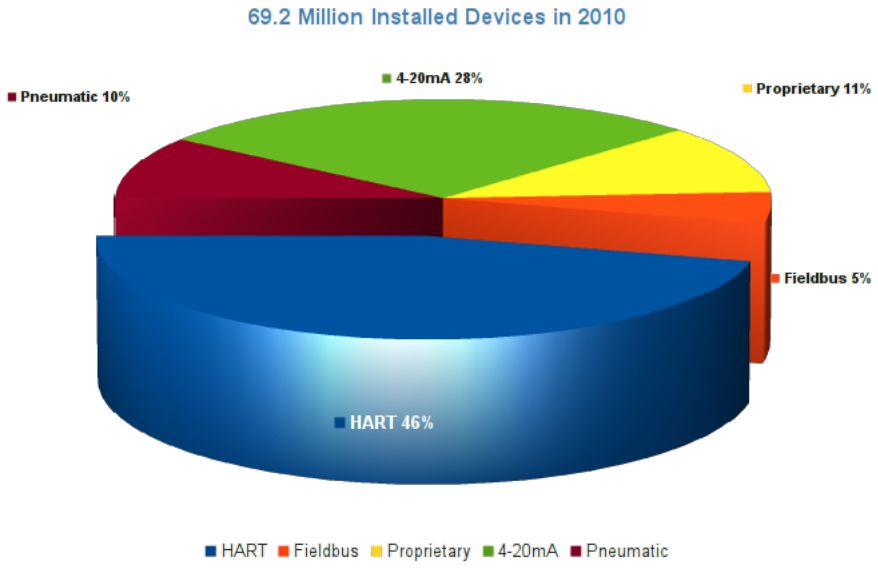


Figura 2.3: Participación de dispositivos HART en el mercado [12].

2.3. Soluciones protocolo HART en el mercado

La presencia de HART en el mercado presenta una diferencia entre su capa física FSK y las capas de enlace de datos y aplicación.

El integrado DS8500 es la solución estándar para la modulación/demodulación FSK. Necesita de la conexión de sólo componentes pasivos para interactuar con el bus de campo HART. Existen varias soluciones que hacen uso del integrado DS8500 e incluyen la conexión recomendada en su hoja de datos. En este trabajo se utilizó el componente DS8500 Evaluation Kit Board, por MAXIM Integrated Circuits [14].

La implementación de la capa de enlace de datos y aplicación se encontró dos alternativas en el mercado, ambas ofreciendo un producto y servicio similar. Se cotizó el Stack del esclavo del protocolo HART en las empresas ExalonDelft[12] y SoftDEL[13].

ExalonDelft ofreció el stack del protocolo por €8000. Además, por €950 se consigue el modem HART. SoftDEL vende el Stack del esclavo a USD$7500. Ambos productos entregan el código fuente y permiten usarlo en la cantidad de dispositivos que sea necesario mientras sean del mismo fabricante que compró la licencia.

Ambos Protocol Stacks tienen la ventaja de un costo variable cero, pero cuentan con un precio inicial que es de carácter restrictivo para las condiciones en las que se inició el trabajo presente. Debido a este motivo se desarrolló el módulo de comunicación HART presente en este trabajo desarrollando capa de enlace de datos y aplicación, pero usando un modem HART disponible en el mercado.

Capítulo 3

Especificaciones de protocolo HART

El protocolo de comunicación HART está basado en el estándar de comunicación Bell 202. Fue diseñado para ser implementado sobre la infraestructura de comunicación análoga de lazo de corriente 4-20[mA]. Es compatible con este bus de campo análogo, ya que puede montarse sobre un valor análogo continuo o de muy baja frecuencia. Hace uso de comunicación Maestro/Esclavo junto con la posibilidad de que un esclavo se configure en modo ráfaga o burst-mode, enviando mensajes periódicamente, sin que el maestro lo encueste. Soporta dos maestros en la misma red, siendo el maestro secundario útil para dispositivos portátiles diagnosticadores de los instrumentos instalados en la red. Permite una comunicación de varios dispositivos esclavos en un mismo bus en configuración multi-drop, pudiendo sólo uno estar en modo ráfaga. Implementa las capas física, enlace de datos y de aplicación del modelo de capas OSI (ver figura 3.1).

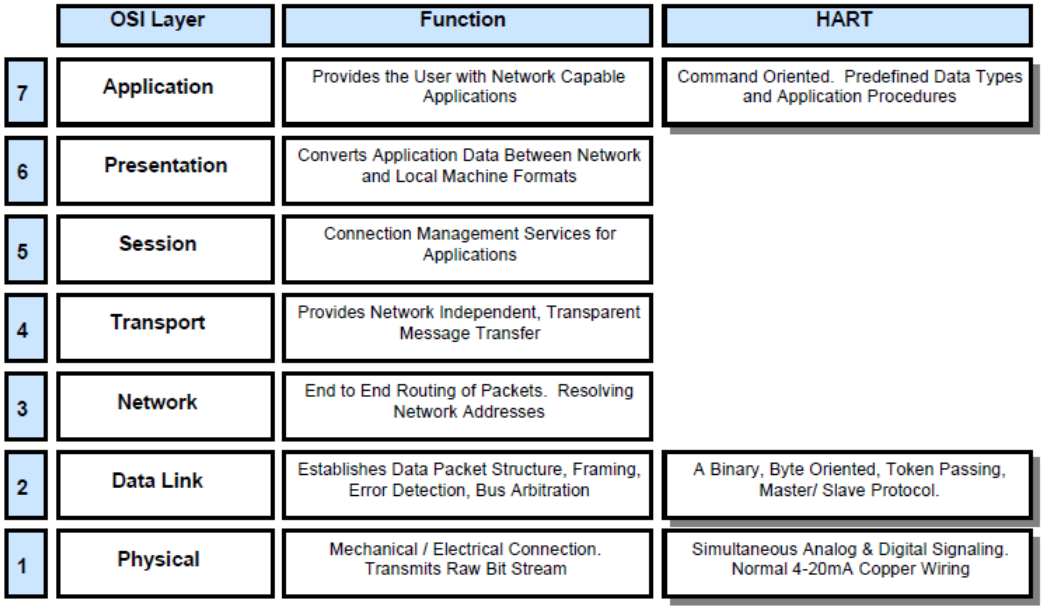


Figura 3.1: Modelo OSI e implementación HART [3].

A continuación se presenta una lista de características de HART con modulación FSK mencionadas y otras adicionales

- Implementado sobre y compatible con lazo de corriente 4-20[mA]
- Transmisión half-duplex de 1200[bits/s]
- Comunicación maestro/esclavo con soporte de un esclavo en modo ráfaga
- Varios dispositivos en mismo bus (multi-drop)
- Implementa 'token-passing' implícito mediante tiempos de espera

HART provee de un ambiente de comunicación industrial con tiempo de respuesta adecuado para configuración de parámetros, diagnóstico centralizado de los dispositivos de campo y la interrogación de variables de control con periodos de actualización superior a 500[ms].

3.1. Servicios

Al igual que en el modelo OSI, para la comunicación entre cada capa el protocolo HART se definen servicios. Los servicios son los módulos que se encargan de comunicar

dos capas adyacentes en el modelo OSI usado por el protocolo HART. En la capa de enlace se implementan los servicios de la capa física y los servicios de capa de enlace.

En general, cada servicio presenta 4 funciones: request(),confirm(), indicate() y response(). Una petición del servicio se inicia con un request() y se obtiene la respuesta mediante confirm(). Si el servicio requiere de la respuesta de otra capa o dispositivo del bus de campo, este la recibe con indicate() y contesta con response() (ver figura 12). Una petición completa involucra, en orden causal, a: request(), indicate(), response() y confirm().

3.2. Capa física

HART especifica distintas modulaciones y distintos medios de propagación para la señal. Hace uso de modulación FSK, PSK, RS425 y 2.4GHz DSSS O-QPSK para WirelessHART. Típicamente se emplenta modulación FSK y es esta la modulación que fue utilizada en el desarrollo de este trabajo.

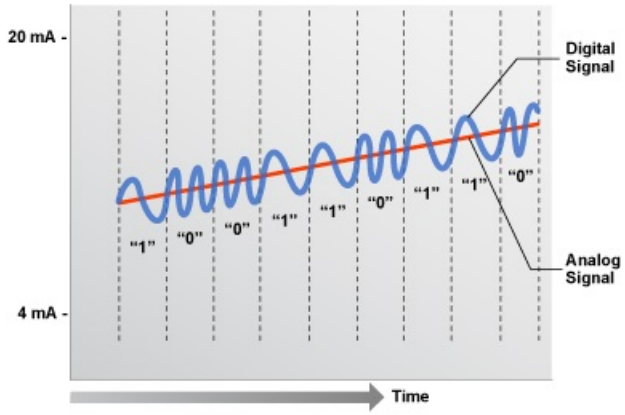


Figura 3.2: FSK superpuesta a señal de corriente [7].

La modulación FSK usada en HART está basada en el estándar de comunicación Bell202 FSK, permitiendo superponer en el bus de campo 4-20[mA], de señal de corriente de baja frecuencia, la señal digital FSK de 1200 [bits/s]. Esta modulación hace uso de dos frecuencias distintas para representar los valores digitales, con amplitud de 0,5[mA], 1,2[KHz] como '1' y 2,2[KHz] como '0' lógicos. Se define un espectro para la señal digital de 0,5[kHz] a 10[KHz]. Además, se especifican -40[dB] por década bajo

los 0,5[KHz] y -20[dB] para frecuencias mayores a 10[KHz] (figura 3.3).

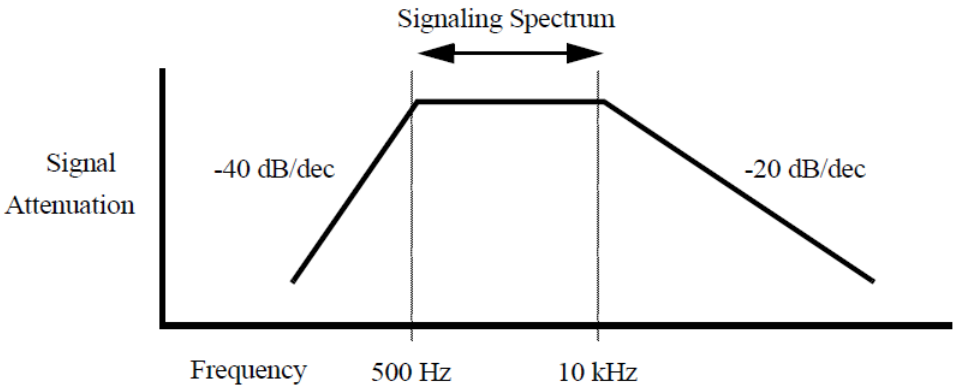


Figura 3.3: Espectro HART FSK [7].

La construcción de un byte se indica en figura 3.4. La transmisión comienza con start bit de valor lógico '0'. Luego se envían los 8 bits de dato del byte que envía la capa de enlace con un bit de paridad. La transmisión termina con un stop bit de valor lógico '1'. El bit de paridad permite la detección de errores y es uno de los dos mecanismos que especifica el protocolo con este fin.

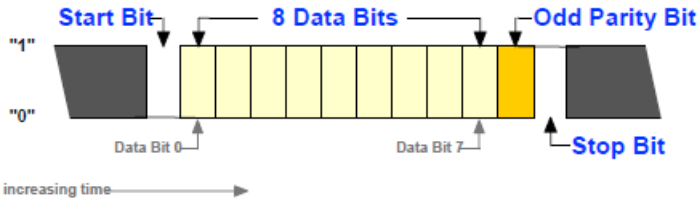


Figura 3.4: Bits que constituyen un carácter [7].

Al tener el protocolo HART una transmisión de 1200[bits/s] y una modulación de desplazamiento de frecuencia es necesario prever, que el valor continuo de la modulación sea igual a 0[mA] para no perturbar la señal del lazo de corriente de 4-20[mA].

Cada bit tiene una duración de 1/1200[s]. El valor lógico '1' tiene un periodo de también 1/1200[s], por lo que completa un ciclo para cada bit. El valor lógico '0' tiene un periodo de 1/2200[s], completando 1,5 ciclos y obteniéndose un valor medio distinto de 0[mA]. Esto se puede corregir al tener una señal de fase continua. Al modular FSK con fase

continua (figura 3.5), el valor medio es corregido por el siguiente ’0’ transmitido, ya que se completa un total de 3 ciclos con valor medio 0[mA] en el lazo de corriente. Este valor no varía por la cantidad de ’1’ transmitidos entre cada ’0’ por su característica de completar un ciclo en cada bit. El uso de fase continua en la modulación impacta, también, en la reducción del uso de ancho de banda, al no presentar cambios de alta frecuencia, pero se necesita un modulador/demodulador más complejo.

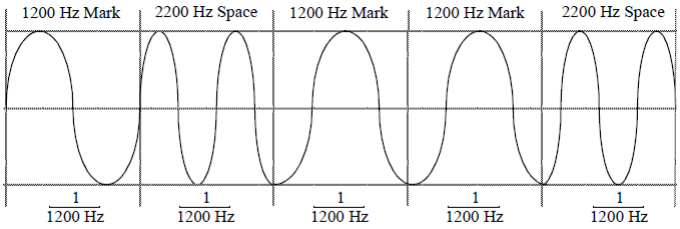


Figura 3.5: Ejemplo de transmisión FSK de fase continua [7].

La demodulación es coherente con la transmisión serial producida en el modem HART. Esto hace que no sea necesario el uso de un buffer en el modem que acumule los valores transmitidos en la red.

3.3. Capa de enlace

La capa de enlace es responsable de una comunicación libre de errores entre los dispositivos HART. Especifica las reglas para permitir comunicación digital sobre una capa física. Hace uso de los servicios de capa física y 2 concepto para implementar modularmente la funcionalidad de esta capa y comunicarse con la capa física y capa de aplicaciones. Estos módulos son los de Control de Enlace Lógico (DLL) y Control de Acceso al Medio (MAC) (figura 3.6).

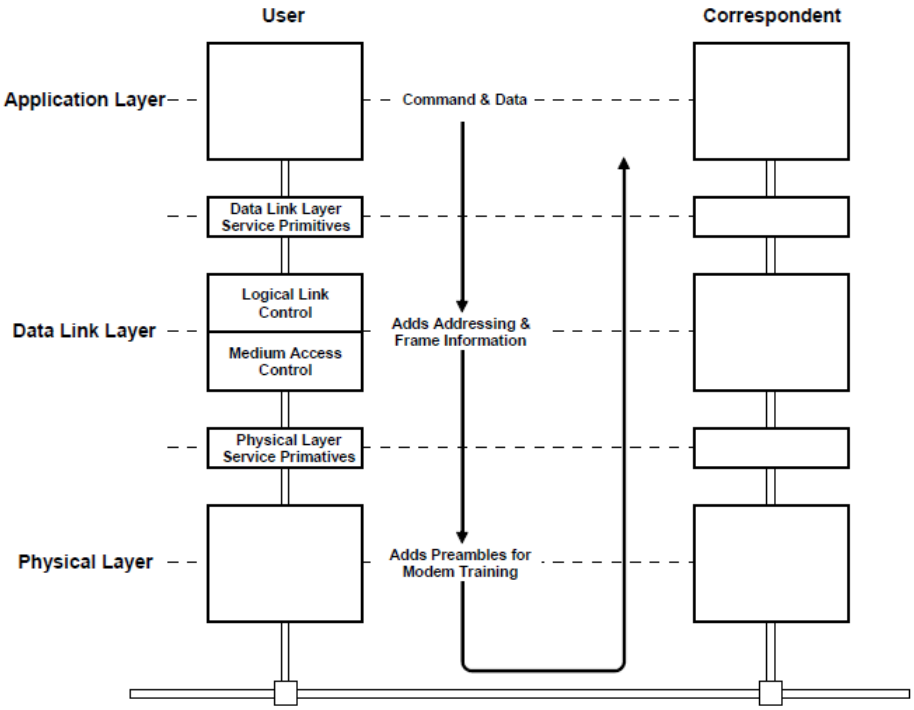


Figura 3.6: Módulos internos de capa de enlace y servicios asociados [3].

La capa de enlace HART hace uso de los servicios de la capa física. Los servicios de capa física permiten implementar la funcionalidad necesaria para comunicarse con un modem HART desde un microcontrolador o unidad computacional con GPIOs y UART. Los servicios son ENABLE, DATA y ERROR (figura 3.7). El servicio ENABLE permite informar si se está comunicando en el bus de campo, solicitar la comunicación y verificar si es posible transmitir. Utilizan las señales de Carrier Detection (CD), Request to Send (RTS), el UART y error en el bit de paridad del UART del microcontrolador para proveer estos servicios. El servicio DATA da la funcionalidad del UART, permitiendo escribir y leer el buffer serial almacenado una vez que ENABLE indique. Por último, el servicio ERROR informa de los errores de paridad en la transmisión de un byte.

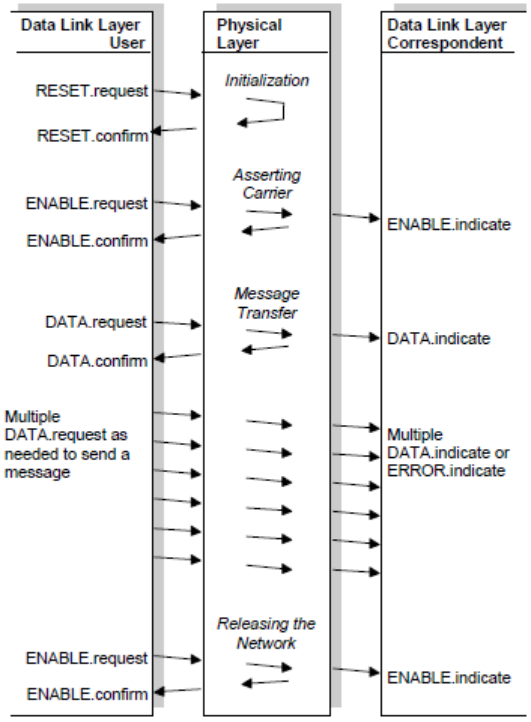


Figura 3.7: Interacción de servicios de capa física [3].

3.3.1. Control de enlace lógico

El Control de Enlace Lógico (LLC) define el empaquetamiento del marco del mensaje resultante de los bytes transmitidos mientras haya portadora. HART denomina al marco del mensaje como unidad de datos del protocolo o Protocol Data Unit (PDU). El PDU está compuesto por el delimitador, la dirección, bytes de expansión opcional, un comando, cantidad de bytes en datos, datos y byte de chequeo (figura 3.8). El delimitador tiene un tamaño fijo y define el tamaño de la cabecera, mientras la cantidad de bytes define el tamaño de los datos en cantidad de bytes. A continuación se examinarán los campos que conforman el PDU. El PDU no incluye los bytes de preámbulo que se reciben antes del start byte.

Delimiter	Address	[Expansion Bytes]	Command	Byte Count	[Data]	Check Byte

Figura 3.8: PDU establecido por el protocolo HART [3].

- Delimitador

  El delimitador es el campo de 1 byte de tamaño que especifica qué tipo de dirección se usa, el tamaño del byte de expansión, la modulación en uso, y el tipo que dispositivo HART se comunica con quien (figura 3.9). El bit 0 especifica si está en uso dirección polling o unique. Luego se usan 2 bits para especificar el largo de los bytes de expansión, siendo su valor la cantidad de bytes. Los bits 3 y 4 definen la modulación en uso, donde FSK tiene valor 0. Los tres bits restantes establecen cual dispositivo se comunica con cual.

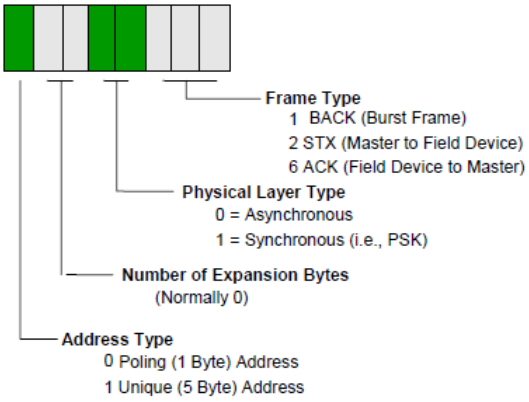


Figura 3.9: Campos definidos por el byte del delimitador [3].

- Dirección

  La dirección puede ser o de tipo polling o unique. Una dirección polling tiene un tamaño de 1 byte (ver figura 3.10). La dirección propia de cada dispositivo sólo puede definirse en los 6 bits de menor significancia (LSB). El bit de mayor significancia especifica con un ’1’ lógico si la comunicación es con el maestro primario y con ’0’ si es con el maestro secundario. El bit 6 indica si se trata de una transmisión en modo ráfaga con un ’1’ o no con un ’0’. Todos los dispositivos manufacturados deben tener la dirección 0 desde fábrica.

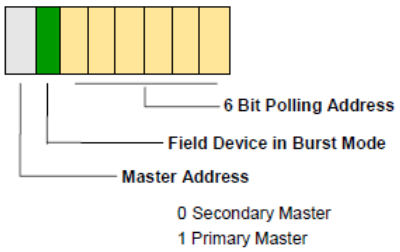


Figura 3.10: Campos de bits del byte de una dirección polling [3].

Una dirección unique usa 5 bytes. El primer byte se configura semejante a una dirección polling, pero con su valor siendo el ID del manufacturador de 6 bits. El segundo byte es el Código de Tipo Dispositivo manejado por el manufacturador. Los 3 últimos bytes son el identificador de Dispositivo Único y su valor debe ser tal para que identifique únicamente al dispositivo esclavo en un mismo bus de campo HART.

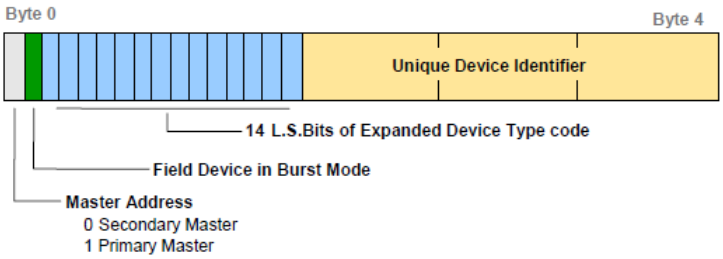


Figura 3.11: Campos de bits de los bytes en una dirección unique [3].

- Bytes de expansión

  Los bytes de expansión son definidos sin utilidad. Tienen el propósito de mantener compatibilidad con futuras versiones HART.

- Comando

  El campo de comando es de 1 byte de tamaño y específica la petición que hace el Maestro, si es un PDU con tipo de marco STX, o a que comando está respondiendo el Esclavo, si es ACK.

- Tamaños de datos y datos

  El campo de tamaño de datos indica la cantidad de bytes del campo de datos, con un tamaño máximo de 256 bytes para datos, al ser un campo de sólo 1 byte. En datos se encuentra la información acorde al comando, especificando cada comando los campos dispuestos en data.

- Bytes de chequeo

  HART implementa detección de error de paridad. Esto se consigue al realiza un 'o' exclusivo sobre los bytes recibidos. Si el resultado es '0' en todos sus bits se tiene un PDU sin errores y el paquete es procesado en la capa de enlace de datos. De esta manera, se implementa una detección de errores de dos dimensiones, calculando la paridad del PDU recibo en capa enlace y la paridad de los bits de cada byte como se observa en la figura 3.12.

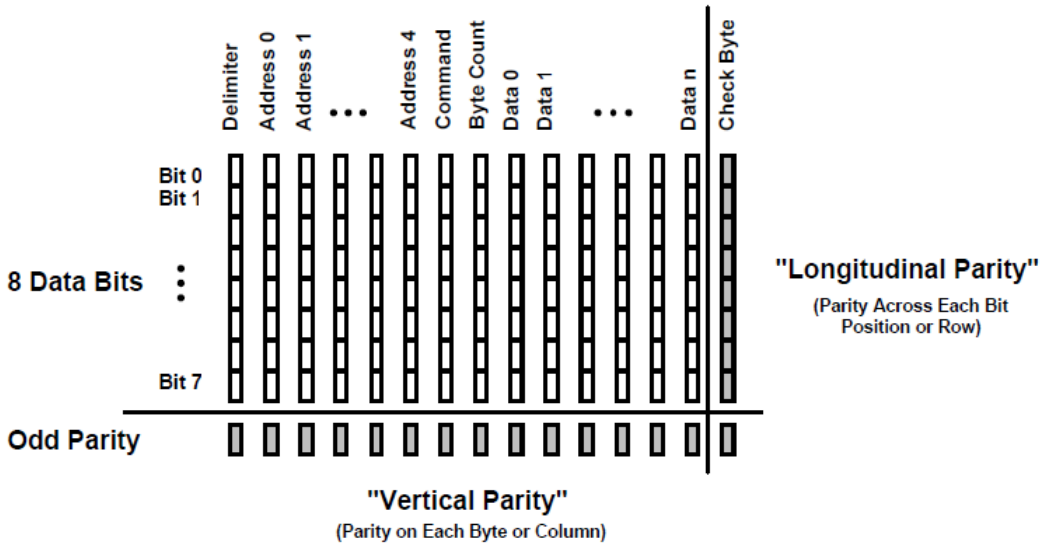


Figura 3.12: Detección de errores en protocolo HART [3].

3.3.2. Control de acceso al medio

El Control de Acceso al Medio (MAC) permite la existencia en un mismo bus de campo de un único maestro primario, un único maestro secundario y un único esclavo

en modo ráfaga. Los dispositivos esclavos son pasivos al no estar en modo ráfaga, y pueden existir varios activos en una misma red. Este acceso está garantizado de forma equitativa. Se implementa mediante el paso implícito de un ?token? o símbolo.

La operación del Control de Acceso al Medio depende de los dispositivos activos del red, el campo de tipo de marco y la dirección del maestro (’0’ para Maestro secundario y ’1’ para Maestro primario), y el momento en que la portadora desaparece. El protocolo define los tipos de marco:

- STX : token pasado a esclavo
- ACK : token pasado a Maestro que no se indica en campo de dirección del esclavo.
- BACK : token pasado a Maestro que no se indica en campo de dirección de esclavo en modo ráfaga.

Para una red HART sin esclavo en modo ráfaga es un dispositivo maestro el que controla el acceso del bus de campo (ver figura 3.13). El maestro envía encuesta a un esclavo y pasa el token a este esclavo con un tipo de marco STX. Luego, el esclavo responde a maestro con su dirección (’0’ o ’1’) en el campo de dirección, pasando de esta forma el token al otro maestro en la red con un tipo de marco ACK. Al tener el token este nuevo maestro puede enviar un comando a algún esclavo. Así, el esclavo que recibe el comando responde al segundo maestro y el token es pasado al primer maestro, obteniéndolo nuevamente. Con este mecanismo se garantiza el acceso tanto al maestro secundario como al maestro secundario.

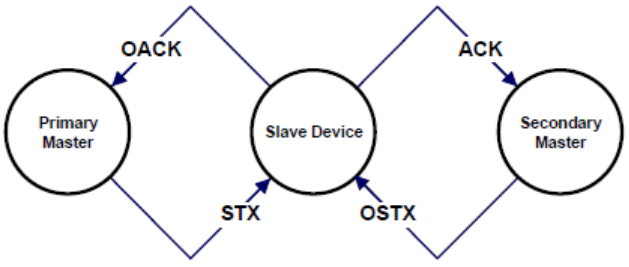


Figura 3.13: Paso de token en red HART sin esclavo en modo ráfaga [3].

En el caso de una red HART con un dispositivo en modo ráfaga el paso de token cambia. En esta configuración es el dispositivo Esclavo en modo ráfaga quien controla el

paso del token (ver figura 3.14). Este Esclavo transmite periódicamente un tipo de marco BACK, con la dirección del Maestro (’0’ para Maestro secundario y ’1’ para Maestro primario) cambiando a cada mensaje. Con este BACK se pasa el token al Maestro que no indica el campo de dirección del Esclavo. El Maestro con el token puede encuestar a un esclavo cualquiera pasándole el token con tipo de marco STX. Luego, el Esclavo con el token responde al Maestro devolviendo el token al dispositivo en modo ráfaga. Esta vez el Esclavo envía un BACK con la dirección del otro Maestro. En caso que el Maestro no transmita, el dispositivo en modo ráfaga recupera el token enviando el mensaje al Maestro distinto del mensaje anterior, pasando el token al Maestro que envío el mensaje anterior.

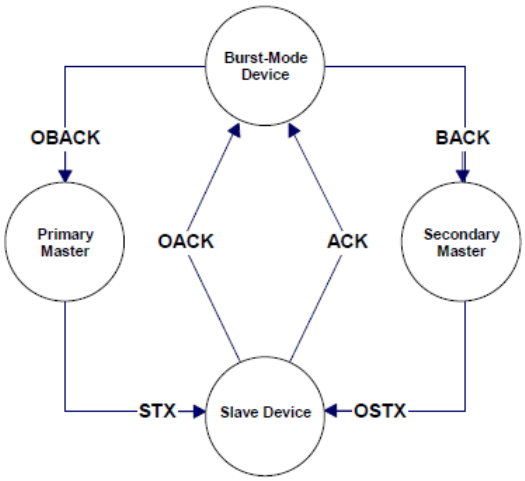


Figura 3.14: Paso de token en red HART con Esclavo en modo ráfaga [3].

El paso implícito del token se implemente mediante tiempos de espera una vez que no se detecta portadora en el bus de campo. Se definen los siguientes tiempos de espera de ingreso a la red para cumplir con la distribución equitativa del token:

- Tiempo de salida de enlace (RT1 primario o secundario) : tiempo de espera del Maestro a la respuesta de un Esclavo o de un Esclavo en modo ráfaga para siguiente transmisión.
- Tiempo de salida de esclavo (STO) : tiempo máximo de inicio de transmisión de un esclavo para responder a un Maestro.
- Tiempo de concesión de enlace (RT2) : tiempo de espera de un Maestro para que el otro Maestro o Esclavo en modo ráfaga acceda a la red.

- Tiempo de mantención (HOLD) : tiempo de retardo de envío de un maestro al haberse terminado RT2.

Los tiempo de mayor a menor son: RT2, RT1 secundario, RT1 primario, STO, HOLD. Un Esclavo en modo ráfaga debe enviar su mensaje en al termino del tiempo RT1. Así garantiza el uso del token al iniciar la transmisión antes de que el tiempo RT2 termine y el Maestro acceda a la red. Los tiempos usados se especifican en figura 3.15.

Parámetro	**Requerimiento**
Preámbulo	5 bytes de 0xFF
Start byte	2 bytes de 0xFF
Formato de carácter	Asíncrono, 11 bits (1 start bit, 8 bits de datos, 1 bit paridad, 1 stop bit)
Time-out esclavo (STO)	28 caracteres de tiempo
HOLD	2 caracteres de tiempo
Tiempo de concesión de enlace (RT2)	8 caracteres de tiempo
Tiempo de salida de enlace (RT1) Primario Secundario	 33 caracteres de tiempo 41 caracteres de tiempo
Error de gap	Menos de 1 carácter de tiempo entre 2 caracteres cualesquiera.

Figura 3.15: Tiempos usados para paso de 'token' implícito [3].

El esclavo debe cumplir con el menor tiempo de respuesta. Por este motivo, basta con responder lo más rápido posible desde el microcontrolador. El tiempo máximo estipulado de retardo en el inicio de la respuesta es de 28 caracteres o 256[ms].

3.4. Capa de aplicación

La capa de aplicación específica los comandos que son usados por el protocolo HART, las unidades de medición estándar y los estados de reporte de error usados en la red. Los dispositivos de campo esclavos deben responder con al menos 2 bytes en el campo de datos. El primer bit debe ser '0' si no hay error y '1' si hay error. En caso de error debe incluirse el código de error que especifica el comando en particular. Cada comando especifica los campos de los que está compuesto, el formato de cada uno y los códigos de errores posibles con los que se debe responder si ocurren.

Son tres clases de comandos los que especifica el protocolo:

- Comandos universales
- Comandos de práctica común
- Comandos específicos a familias de dispositivos

Los comandos de práctica común y los comandos específicos a familias de dispositivos son opcionales. Los comandos universales son obligatorios de implementar, ya que son requeridos para el funcionamiento mínimo de un dispositivo. Permiten la transmisión de los datos de medición de hasta 4 variables de medición, diagnóstico del dispositivo conectado y configuración de parámetros del protocolo HART del dispositivo. Por su parte, los de práctica común son siempre implementados por dispositivos modernos.

En total, se listan 18 comandos universales. Algunos comandos característicos de los comandos universales se describen en figura 3.16.

Comando	**Descripción**
Comando 0 Leer identificador único	Retorna información de identificación del dispositivo de campo. Incluye el tipo de dispositivo, ID del dispositivo (datos necesarios para obtener la dirección unique junto con el ID del manufacturador) y nivel de revisión. Este dispositivo debe retornar para las direcciones polling y unique asociadas al dispositivo.
Comando 1 Leer variable primaria	Retorna el valor de la variable primaria junto con la unidad de medida como Units Code (ver fig. 15).
Comando 6 Escritura de dirección polling	Escribe la dirección polling del dispositivo de campo y configura el modo del lazo de corriente.

Figura 3.16: Descripción de comandos representativos de la lista de comandos universales [8].

La especificación de comandos universales [4] define una lista de formatos posibles para los campos de los comandos. Cada campo tiene un formato que define como transmitir su valor. En figura 3.17 se listan estos formatos y como codificar el valor en ese formato.

Formato	Codificación
bits	cada bit del campo tiene un significado particular
date	3 bytes donde cada uno especifica el día, el mes y el año desde 1900 (máximo 31/12/2155).
enum	entero donde los valores tienen un significado particular. El campo hace referencia a la especificación Common Table [16].
float	número de punto flotante IEEE754. Se transmite el exponente y luego la mantisa.
latin-1	ISO Latin-1 de 8-bits
packed	caracteres alfa-numéricos de 6-bits.
unsigned-nn	entero positivo con nn bits. Se envía de MSB a LSB.

Figura 3.17: Formato usado por los campos de cada comando [8].

Capítulo 4

Desarrollo de módulo de comunicación HART

El módulo de comunicación HART para sensores industriales cumplió con conectar un sistema Linux con un software que almacena información para entregar a una red HART. Se ofrece una biblioteca en el lenguaje de programación C++ para ser usada en un sistema Linux que permite comunicar el software a través de una conexión USB con el firmware en el microcontrolador. Esta biblioteca es un adaptador a la capa de aplicación HART. Separa la interacción propia de la red HART de la del software. Esto permite que el software sólo escriba o lea las variables de control hacia o desde la red, sin conocer el protocolo.

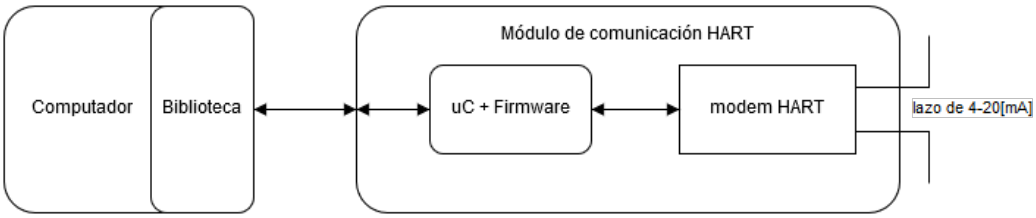


Figura 4.1: Diagrama de bloques de módulo de comunicación HART.

4.1. Diseño de módulo de comunicación

El diseño lógico del módulo de comunicación HART (ver figura 4.1) está compuesto por los siguientes submódulos:

- Modem HART
- Microcontrolador + Firmware
- Biblioteca para Sistema Operativo GNU/Linux

El modem HART realiza la modulación y demodulación de la señal FSK presente en el bus de campo. El microcontrolador se conecta a la placa del modem para recibir las señales de control y el firmware quemado en su memoria se encarga de realizar las tareas HART. El microcontrolador está igualmente conectado por USB a una unidad de computación con sistema operativo GNU/Linux. Fue desarrollada una biblioteca que permite incluir de forma sencilla el manejo de los datos serial del USB al programa del sensor.

La biblioteca desarrollada es una capa extra de abstracción encargada de separar la implementación HART de la implementación del software en el sistema Linux. Esta capa permite actualizar y obtener el valor de las variables de control que maneja la red. Expone sólo las variables de control en el software del sistema Linux. De esta manera fue posible desacoplar la lógica del sensor de la lógica del protocolo HART. Esto entrega el beneficio que los errores en la red HART son responsabilidad de la biblioteca y módulo de comunicación desarrollado en este trabajo.

4.2. Implementación de módulo de comunicación

Se escogió como lenguaje de programación C++. Este lenguaje permite programar orientado objetos (OOP). La principal ventaja de usar OOP es la encapsulación, que permite diseñar cada capa aisladamente, usando para cada clase las interfaces que especifica el protocolo HART. Se utilizan clases para definir cada capa y sus métodos públicos son los servicios definidos de cada una (ver figura 4.2).

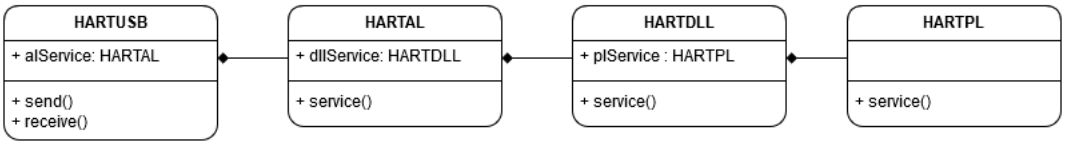


Figura 4.2: Diagrama de clases básico de firmware.

En el diseño Orientado a Objetos se escogió composición de clases por sobre herencia. Esto debido a que al acceder a capa aplicación no se desea necesitar servicios de la capa física o de enlace de datos para que el protocolo funcione correctamente. Esto hace que la relación sea ?tiene un? antes que ?es un?.

Cada clase tiene su cabecera e implementación. Se usó una cabecera para la definición de estructuras, enums y constantes que ayudaron a reducir errores y a la legibilidad del código. Se define una estructura para almacenar los valores de los campos del delimitador, enums como el tipo de modulación y de dirección, si se trata de maestro primario o secundario. Constantes definidas son los tamaños de los campos del PDU, sus valores (preámbulo y start byte son iguales a 0xFF en hexadecimal de C y C++). De esta manera se trabajó con texto a nivel del código y no con números.

Una clase de manejo de errores fue agregada. Define mediante un enum los errores posibles. Almacena en su estado interno el último error producido. Debido a que al primer error encontra es posible responder correctamente en la red, sólo es necesario almacenar el último y ningún otro más. Se implementó un objeto global a las implementaciones de las clases (variable extern) que almacena el error generado. Así se comunica un error a la capa superior.

La clase HARTPL se encarga de obtener el buffer de bytes obtenidos haya presencia de portadora en el lazo de 4-20[mA]. Este buffer es luego verificado por la capa de enlace implementada en HARTDLL. Esta clase verifica el valor de los campos, si el comando es enviado al dispositivo y se hay errores. HARTAL, que implementa la capa de aplicación, obtiene los campos de comando, tamaño de datos y datos desde los servicios de HARTDLL. Sus servicios responden asignando el valor correcto a datos y tamaño de datos. HARTUSB recibe comandos desde el puerto USB. Permite actualizar el valor de las variables de control, ajustar el estado del esclavo HART y usar el servicio LOCAL_ MANAGEMENT de HARTDLL.

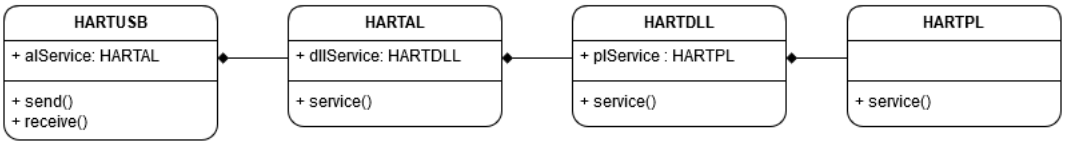


Figura 4.3: Diagrama UML básico de biblioteca.

Se ofrece una cabecera en el lenguaje de programación C++ para incluir el protocolo HART en el código del sensor. Esta biblioteca incluye la posibilidad de interaccionar directamente con la capa de aplicación HART o a través del adaptador que facilita su uso. Se diseñó para un sistema operativo Linux, entorno en el que se trabaja en el proyecto densímetro nuclear.

4.2.1. Implementación capa física

Para cumplir con las especificaciones de la capa física [7] se utilizó el integrado DS8500. Este integrado requiere conexión de componentes pasivos para transformar la señal análoga en digital. Para esto se usó el dispositivo comercial DS8500 KIT proporcionado por Xinzhu Tech [15]. Este componente requiere de la conexión en paralelo a los pines FSK_ IN y FSK_ OUT (figura 4.4), polarización 3,3[V], la conexión de un UART en Tx y Rx, y a 4 pines para las señales de control.

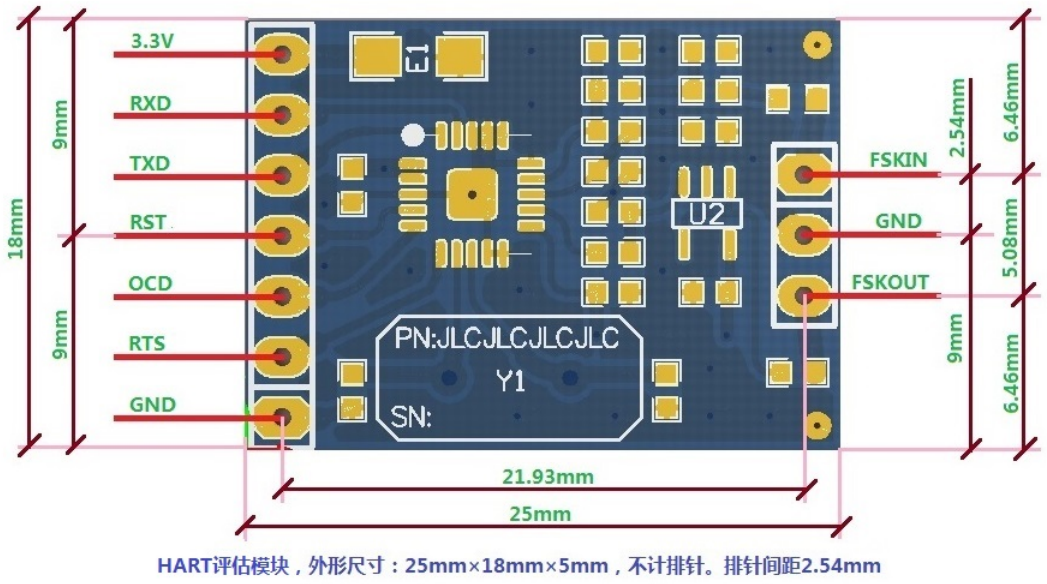


Figura 4.4: Pines disponibles del PCB DS8500-KIT.

En el presente trabajo se obtuvo VDD y GND desde una plataforma de desarrollo Arduino Leonardo, conectando su UART con los pines Tx y Rx de DS8500 KIT y los 4 pines de control de la plataforma fueron conectados a pines GPIO.

La capa física requiere de servicios para que la capa enlace de datos pueda hacer uso de sus funcionalidades. Se implementó la clase HARTProtocolPL (ver fig. 18) con los servicios como métodos públicos. Los servicios se implementaron con las funciones provistas por Arduino para el manejo de UART y GPIOs facilitando la programación del prototipo.

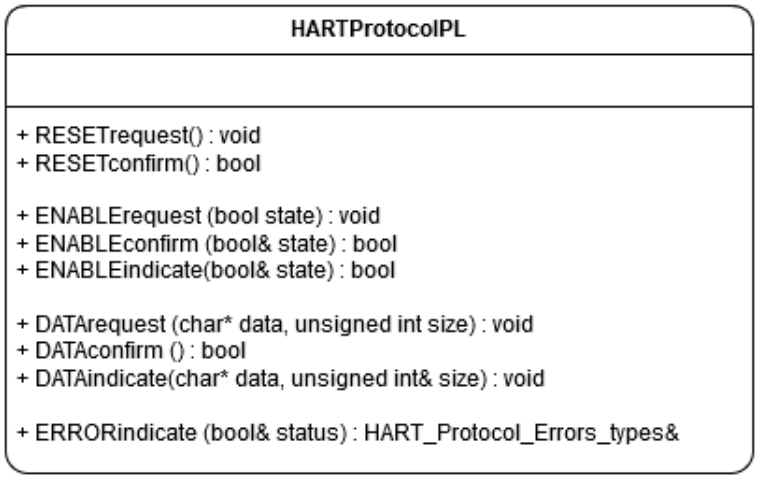


Figura 4.5: Diagrama UML de miembros públicos de clase para capa física.

4.2.2. Implementación capa de enlace de datos

La capa de enlace está conformada por MAC, LLC (capítulo 3.3.) y los servicios de capa de enlace. Los servicios definen los servicios que otorga esta capa a la capa de aplicación (ver figura 4.6). DLL especifica cómo trabajar con la información proporcionada por los campos del PDU. MAC especifica los requisitos de sincronización del sistema.

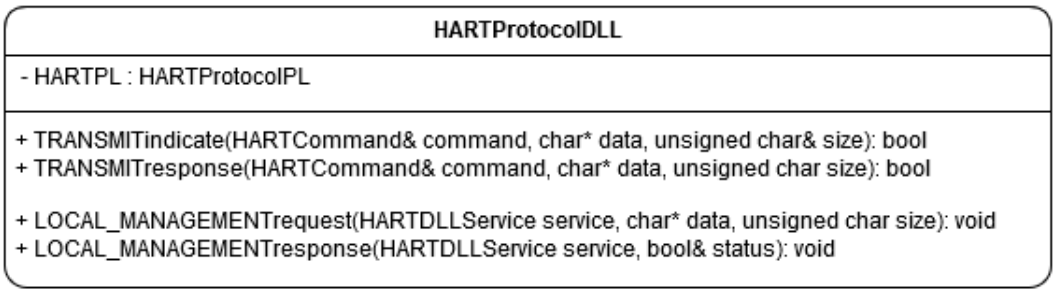


Figura 4.6: Diagrama UML de miembros públicos de clase para capa de enlace de datos.

La implementación de los servicios de enlace de datos necesita de hacer usar de los servicios de la capa física y luego corroborar que los campos tengan los valores adecuados. Para esta evaluación se mantuvo el estado del esclavo en memoria, almacenando el tamaño del preámbulo, valor de modulación FSK, valor de dirección polling, dirección unique y si se encuentra en modo ráfaga o no.

Se corrobora la presencia de error en los campos del PDU uno a la vez y en el orden indicado en figura 4.7. Los errores de capa física, o sea, el error de paridad de byte se manifiesta como un PDU que no contiene todos sus campos completos. Un error no detectado en capa física es detectado con un valor incorrecto del campo checksum en el PDU.

Campos PDU a evaluar	**Acción ante error en el campo**
1. Tamaño y valor de preámbulo	Omitir marco
2. Valor de start byte	Omitir marco
3. Delimitador	Omitir marco
4. Dirección	Omitir marco
5. Bytes de expansión	Omitir marco
6. Byte de comando	Responder con comando leído
7. Byte de tamaño de datos	Omitir marco
8. Datos	Responder indicando error
9. Checksum	Responder indicando error

Figura 4.7: Especificación de respuesta ante error de un campo del PDU [3].

Los servicios de LOCAL_ MANAGEMENT le permiten a capa de aplicación cambiar el estado del esclavo. En figura 4.8 se listan las opciones que especifica HART.

Servicio	Data	Descripción
SET_UNIQUE_ADDRESS	dirección unique	Asigna la dirección unique del dispositivo esclavo.
SET_POLLING_ADDRESS	dirección polling	Asigna la dirección polling del dispositivo esclavo.
SET_PREAMBLE	tamaño de preámbulo de respuesta	Asigna la cantidad de bytes del preámbulo del PDU.

Figura 4.8: Servicios opcionales para cambiar estado de esclavo [3].

4.2.3. Implementación capa de aplicación

La clase HARTProtocolAL (ver figura 4.9) implementó la capa de aplicación en el firmware. Obtiene el comando y los datos a usar del servicio TRANSMITindicate() de la capa de enlace. Usa como interfaz un solo método público que recibe el comando, los datos y el tamaño de los datos. Esta función llama al método privado que responde al comando recibido. Este diseño ofrece una interfaz sencilla, encapsulando la necesidad de llamar la función particular al comando recibido. Esto mejora la legibilidad del código que hace uso de la clase HARTProtocolAL.

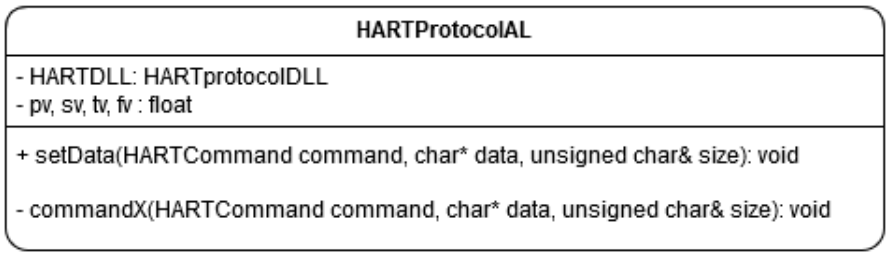


Figura 4.9: Diagrama UML de miembros públicos de clase HARTProtocolAL.

Los 18 comandos universales implementados en HARTProtocolAL permiten al maestro encuestar al esclavo sobre las variables de medición y sobre su estado, además de poder modificarlo. A continuación se listan los primeros 4 comandos universales fundamentales para la comunicación en la red HART y sus requisitos de implementación.

- Comando 0 (leer identificador único)

Necesita acceder a la dirección unique, la revisión del dispositivo, versión del

software y hardware, el número de variables dinámicas usadas (1 por defecto). Los valores de las versiones se implementaron como constantes en la cabecera de definiciones HARTProtocolDeclarations.

- Comando 1 (leer variable principal PV)

  Se define una struct como estructura de dato de las variables dinámicas. Se incluye valor y unidad (valor de unidad depende de Common Tables [16]).

- Comando 2 (leer lazo de corriente y porcentaje de rango)

  Dado un valor máximo y mínimo de corriente del dispositivo esclavo para transmitir medición de PV se define su porcentaje actual. Se agregan como atributos de la clase los valores máximo y mínimos usados por el dispositivo. A partir del valor actual de PV se calcula el porcentaje que le corresponde entre los valores límites definidos.

- Comando 3 (leer variables dinámicas)

  A partir de la cantidad de variables dinámicas establecidas en la configuración se responde con sus valores. Por defecto hay sólo 1 variable dinámica en uso, PV.

Capítulo 5

Pruebas de funcionamiento

Las pruebas realizadas comprueban el funcionamiento de las capas física, enlace de datos y aplicación. Se verifica el funcionamiento separado de cada capa mediante el establecimiento de métodos fijos. De esta manera, al realizarse una nueva revisión del módulo de comunicación HART puede probarse el correcto funcionamiento de cada capa.

Se desarrolló una cabecera para ejecutar sencillas funciones para declarar las pruebas que se desean realizare n las capas de enlace de datos y aplicación. La cabecera permite usar 3 macros. Una macro define el título de la prueba (TESTER_ NAME(text)), mientras que otra recibe la descripción (TESTER_ DESCRIPTION(text)). La tercera TESTER_ TEST(text, function) recibe un texto que nombra la prueba y una función que ejecuta la prueba. Esta función debe retornar un entero y no debe requerir ningún argumento. Si la función retorna 0 significa que fue exitosa. En caso contrario, anota el resultado retornado en pantalla. Esto permitió diseñar pruebas unitarias verificando el funcionamiento del desarrollo del protocolo HART. La cabecera formatea la salida para tener una salida en pantalla ordenada y fácil de observar (ver figura 5.1).

```
TITUTLO
=======

       DESCRIPTION

PRUEBA 1.....................................................[OK]
PRUEBA 2.....................................................[BAD][1]
```

Figura 5.1: Formato de cabecera tester.h para pruebas unitarias.

5.1. Pruebas capa física

Es necesario revisar el funcionamiento del modem DS8500 KIT. Se usó ES232UP USB-HART Modem (figura 5.2) como dispositivo comercial para realizar la modulación de los comandos enviados. Este dispositivo permite 3 modos. En el primer modo actúa como fuente de 24[V] conectando en serie a esta fuente interna una resistencia de 270[Ohm] en paralelo al modem. Otro en el que se conecta a la red a través de la resistencia de 270[Ohm] y el tercero la resistencia debe proveerse externamente. Es el primer modo, que implementa la fuente de 24[V], el que se usa para realizar las pruebas.

Se usó ES232UP USB-HART Modem como maestro primario de la red. Este modem requiere de ser conectado por USB a un computador y usar el software HART Analyzer en un entorno Windows para enviar comandos por medio de la red HART.

Figura 5.2: ES232UP USB-HART Modem usado como maestro primario.

Su modulación fue registrada con osciloscopio comprobando la presencia de portadoras en 1200[Hz] y 2200[Hz] (ver figuras 5.3 y 5.4), valores que especifica HART para los valores binarios ’1’ y ’0’. Para verificar que la modulación sea correcta y envíe los comandos que se eligen en el software es necesario demodular y ver sus valores binarios.

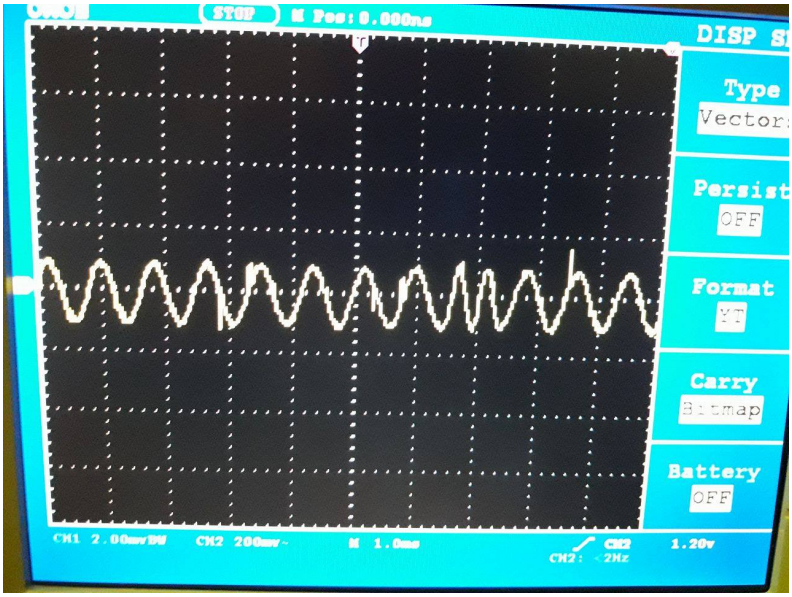


Figura 5.3: Modulación de ES232UP USB-HART Modem mostrada en tiempo.

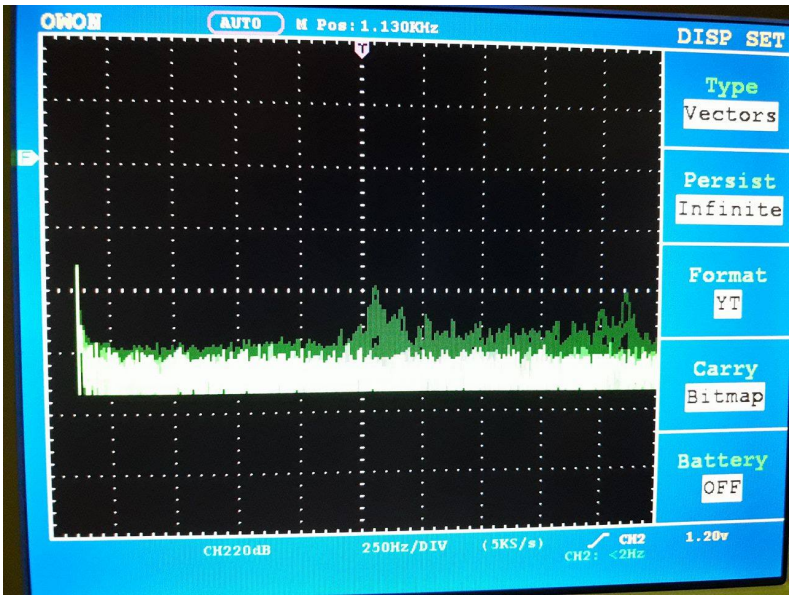


Figura 5.4: Modulación de ES232UP USB-HART Modem mostrada en frecuencia.

Se probó el modem DS8500-KIT conectándolo a ES232UP USB-HART Modem en modo de 24[V] con resistencia interna. Se usaron 3 resistencias de 100[Ohm] y 2[W] para DS8500-KIT. El pin de detección de portadora OC variaría de ’0’ a ’1’ indicando la presencia de modulación HART en la red. Como resultado se obtuvo que en ningún experimento hubo detección de portadora. Observando la red mediante osciloscopio se

notó que al conectar el modem DS8500-KIT la portadora no se encontraba de ES232UP USB-HART Modem. Esto indica que el dispositivo esta defectuoso.

5.2. Pruebas capa de enlace de datos

La capa de enlace de datos fue probada en un sistema Linux. Esto se realizó implementando una clase con la misma interfaz que la clase HARTProtocolPL, pero transmitiendo los datos por la entrada y salida estándar (stdin y stdout). Se creó un archivo con marcos anotados en caracteres ASCII representando valores hexadecimales (ver figura 5.5). Se usa una tubería anónima para redirigir la salida estándar del comando cat hecho sobre archivo de marcos a la entrada estándar del programa de prueba, que modifico la capa física.

Se implementaron los siguientes errores para corroborar su correcta detección (ver figura 5.5)

- Transferencia correcta de maestro primario a esclavo de comando 0 (línea 1)
- Error en valor de 5 byte de preámbulo (línea 3)
- Error en valor de start byte (línea 5)
- Modulación PSK en delimitador (línea 7)
- 2 byte de expansión indicados en delimitador (línea 9)
- Dirección polling equivocada (línea 11)
- Dirección unique equivocada (línea 13)
- Incongruencia en el uso de dirección polling en delimitador y comando distinto de comando 0 (línea 15)
- Declaración de 2 bytes de datos, pero no se envía ninguno (línea 17)
- Error en checksum (línea 19)
- Demaciados bytes en datos, produciendo error de checksum. Para corroborar que no hay problema de override de buffer. (línea 21)

```
1   FFFFFFFFFFFF0280000082
2
3   FFFFFFFF0FFF0280000082
4
5   FFFFFFFFFF0F0280000082
6
7   FFFFFFFFFFFF0A80000082
8
9   FFFFFFFFFFFF2280000082
10
11  FFFFFFFFFFFF0281000082
12
13  FFFFFFFFFFFF828100000000000082
14
15  FFFFFFFFFFFF0280010083
16
17  FFFFFFFFFFFF828000000000000200
18
19  FFFFFFFFFFFF8280000000000002FFFF01
20
21  FFFFFFFFFFFF8280000000000001FFFF03
```

Figura 5.5: Marcos usados para probar implementación DLL.

Mediante la cabecera tester.h y las pruebas unitarias se obtuvo una validación de la implementación de la capa de enlace de datos de HART (ver figura 5.6).

```
DLL :: Transmit Indicate.....................................[OK]
DLL :: Transmit (preamble) bad preamble size.................[OK]
DLL :: Transmit (start byte) no start byte...................[OK]
DLL :: Transmit (delim) wrong modulation -> Not FSK..........[OK]
DLL :: Transmit (delim) not 0 expansion bytes................[OK]
DLL :: Transmit (adrress) bad polling address................[OK]
DLL :: Transmit (address) bad unique address.................[OK]
DLL :: Transmit (delim&command) polling address command 1...[OK]
DLL :: Transmit (data) not enough data.......................[OK]
DLL :: Transmit (checksum) bad checksum calculation..........[OK]
DLL :: Transmit (checksum) too much data.....................[OK]
```

Figura 5.6: Resultado de pruebas unitarias.

5.3. Pruebas módulo de comunicación integrado

Las pruebas para el módulo de comunicación se deben realizar con un dispositivo maestro HART comercial. Esto comprueba que el desarrollo realizado funciona de acuerdo a las especificaciones HART y es compatible con dispositivos utilizados en redes HART instaladas en la industria.

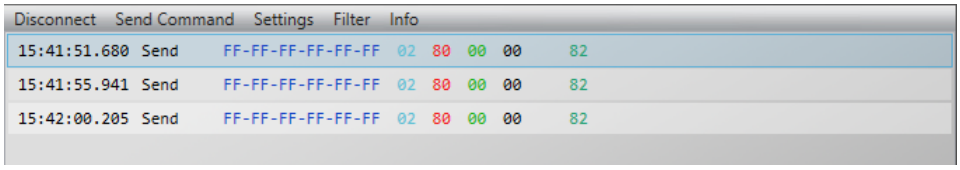


Figura 5.7: Reporte de bytes enviados por HART Analyzer.

A pesar del mal funcionamiento de la capa física fue posible aislar los errores del

submódulo del modem (ver figura 4.1) y a la subcapa MAC de la capa de enlace de datos. Esto debido a que los marcos enviados por el software HART Analyzer reporta los mismos valores esperado por LLC de la capa de enlace (ver figura 5.5 y 5.7). El marco usado por el software es el mismo que se usó como primera prueba de las pruebas de capa de enlace. Esto permite prever un correcto funcionamiento de las capas de enlace de datos y aplicación.

Capítulo 6

Conclusiones

El diseño modular del protocolo HART permitio aislar los errores durante el desarrollo y corregirlos sin afectar el resto del trabajo. Además, permitió corroborar el funcionamiento de las capas implementadas aún cuando la comunicación de extremo a extremo no fue posible.

El uso de pruebas unitarias facilita la tarea de realizar modificaciones. Esto debido a que es sencillo verificar que los cambios hecho hayan o no tenido impacto en el resto del código al fallar pruebas que no deberían verse afectadas.

Será necesario adquirir un nuevo modem HART para el módulo de comunicación esclavo. Con este nuevo dispositivo quedará por probar su funcionamiento y los tiempos de respuesta máximos en una comunicación de extremo a extremo.

Aún con una comunicación fallida es esperable que una vez se logren las funciones de capa física y MAC el módulo esclavo responda correctamente a las peticiones del maestro, quedando agregar más comandos a la lista de los ya implementados.

El desarrolló del presente trabajo permitirá disponer de un módulo de comunicación esclavo HART de fácil configuración y uso, sin tener que modificar el código fuente. Esta última característica es una ventaja importante frente a los producto ofrecidos por el mercado del Stack HART.

CAPÍTULO 6

[1] http://www.automation.com/automation news/industry/arc says hart is the leader in communications

[2] http://www.profibus.com/technology/profibus/overview/

[3] Token Passing Data Link Layer Specification, HCF_ SPEC-81, http://en.hartcomm.org/hcf/org_ mbr/documents/documents_ spec_ list.html

[4] Universal Command Specification, HCF_ SPEC 127, http://en.hartcomm.org/hcf/org_ mbr/documents/documents_ spec_ list.html

[5] Common Practice Command Specification, HCF_ SPEC 151, http://en.hartcomm.org/hcf/org_ mbr/documents/documents_ spec_ list.html

[6] HART FieldComm Group, http://en.hartcomm.org/hcp/tech/aboutprotocol/aboutprotocol_ how.html

[7] FSK Physical Layer Specification, HCF Spec 54, http://en.hartcomm.org/hcf/org_ mbr/documents/documents_ spec_ list.html

[8] Universal Command Specification, HCF Spec 127, http://en.hartcomm.org/hcf/org_ mbr/documents/documents_ spec_ list.html

[9] Asociación de la Industria Electrica Electrónica (AIE), http://www.aie.cl/files/file/comites/ca/articulos/ag 06.pdf

[10] Reliability Driven Asset Management using HART, http://www.arcweb.com/events/arc-industry forum orlando/arcindustryforumorlando2013presentations/Reliability % 20Driven % 20Asset % 20Management % 20Using % 20HART.pdf

[11] ARC Advisory Group, https://www.arcweb.com/about

[12] ExalonDelft, http://www.exalondelft.nl/en/products HART.shtml

[13] SoftDEL, http://www.softdel.com/hart/

[14] Maxim Integrated, https://www.maximintegrated.com/en.html

[15] Xinzhu Tech en Aliexpress, https://es.aliexpress.com/store/704907

[16] Common Tables Specification, HCF_ SPEC 183, http://en.hartcomm.org/hcf/org_ mbr/documents/ documents_ spec_ list.html

CAPÍTULO

Anexo 1 : Especificaciones HART

Anexo 2 : DS8500 Evaluation Kit

Anexo 3 : ES232UP USB-HART Modem

Anexo 4 : Código proyecto Arduino Leonardo

Anexo 5 : Código proyecto PC