

**UNIVERSIDAD TÉCNICA FEDERICO SANTA
MARÍA**

**DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
VALPARAÍSO - CHILE**



Integración de un dispositivo QDC a una Red de Instrumentación Industrial

Rodrigo Alexander Díaz Silva

**TRABAJO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
ELECTRÓNICO**

Mención Electrónica Industrial, Submención Gestión

PROFESOR GUÍA: Manuel Olivares

PROFESOR CORREFERENTE: Cesar Silva

SEPTIEMBRE - 2016

*Quiero agradecer a mis padres,
familiares ,amigos quienes me han
apoyado de forma incondicional en este largo viaje.
Mención especial a mi pareja que ha sido
uno de los pilares en el ultimo tiempo.*

Resumen

En el siguiente trabajo se realiza el desarrollo de un equipo capaz de integrar a la red de instrumentación industrial Profibus DP, un dispositivo QDC.

Para esto se utilizan microcontroladores, ASIC, Single Board Computer (SBC), entre otros. Se establecen limitaciones y mejoras para este desarrollo.

Los dispositivos QDC son utilizados en aplicaciones médicas, en la industria minera y física nuclear, entre otras. En estas aplicaciones son necesarias una gran cantidad de datos en cortos lapsos de tiempo, debido a la naturaleza que estos procesos presentan.

El desarrollo presentado en este documento puede ser utilizado, realizando cambios mínimos, con cualquier otro tipo de dispositivo; por ejemplo, reemplazando el QDC por algún otro ADC, sensor de temperatura, humedad, o para la activación de salidas y/o visualización de datos.

Keywords: SBC, Microcontrolador, Atmega 1280, ASIC, Profibus, Proficonn, Galileo, Caen Digitizer, Red de Instrumentación Industrial, PLC.

Abstract

The following document inspects the development of a device capable of integrating an Industrial Instrumentation Network Profibus DP, a QDC device. To achieve this, microcontrollers, ASIC, Single Board Computer (SBC), among others, are used. Limitations and improvements are established for this development.

The QDC devices are used in medical applications, mining industry and nuclear physics, among others. In this applications are necessary a big amount of data in a short period of time, due to the nature of the process.

The device developed present in this job can be used, with minimal changes, with any other device. For example, to replace the QDC for some other ADC, temperature sensor, humidity sensor, or for output activation and/or data visualization.

Keywords: SBC, Microcontroller, Atmega 1280, ASIC, Profibus, Proficonn, Galileo, Caen Digitizer, Industrial Instrumentation Network, PLC.

Índice general

Índice de cuadros

Índice de figuras

Glosario

<i>QDC</i>	—	Charge Digital Converter.
<i>ELO307</i>	—	Proyecto de Titulación para Ingeniero Civil Electrónico .
<i>SO</i>	—	Sistema Operativo .
<i>MV</i>	—	Máquina Virtual .
<i>SSH</i>	—	Secure Shell .
<i>SoC</i>	—	System on a Chip.
<i>SBC</i>	—	Single Board Computer.
<i>ASIC</i>	—	Application Specific Integrated Circuit.
<i>RAM</i>	—	Random Access Memory.
<i>USB</i>	—	Universal Serial Bus.
<i>SD</i>	—	Secure Digital.
<i>SPI</i>	—	Serial Peripheral Interface.

•

Capítulo 1

Introducción y Objetivos

1.1 Descripción del Problema a Resolver

En el laboratorio SiLab de la Universidad Técnica Federico Santa María se están diseñando equipos basados en fenómenos físicos como el de espectroscopia, a través de los cuales se analiza la interacción de partículas radioactivas con los elementos químicos. Primero se realizan experimentos en los cuales se utilizan QDC's, los cuales son conectados a unas computadoras para la adquisición y procesamiento de los datos. Algo de esto se puede ver en los trabajos de Rene Ríos[?] y Lautaro Narváez[?]. En dichos trabajos se utilizan computadores de escritorio con los cuales se realiza la obtención, vía USB, y procesamiento de los datos obtenidos por el QDC, para su posterior cálculo y análisis. En el trabajo de Rene Ríos [?] se utilizan los QDC's del fabricante CAEN [?] para la obtención de la densidad de cobre, con el fin de desarrollar equipos para aplicaciones mineras. Estos datos deben ser transmitidos a las redes de instrumentación presentes. En la industria existen redes de instrumentación análogas, digitales y análogo/digi-

tales; este trabajo se concentra en las redes digitales, específicamente en la red Profibus DP. El trabajo de Diego Valencia [?] trata de cómo realizar un dispositivo que se puede conectar a la red de instrumentación industrial Profibus-DP y realizar el intercambio de datos.

En relación con el trabajo de Diego Valencia, este desarrollo presenta similitudes en cuanto a la realización de un dispositivo que permite la comunicación con la red de instrumentación Profibus DP, así como en la utilización de un microcontrolador para esta integración. A diferencia del trabajo de Diego que utiliza un microcontrolador, ASIC y electrónica adicional para la adaptación de las señales para su funcionamiento en la red Profibus DP, el dispositivo desarrollado en el presente trabajo utiliza un microcontrolador y un ASIC el cual es el encargado de la adaptación de las señales para la conexión con la red Profibus DP. Para construir un equipo sin utilizar un computador de escritorio, se pueden analizar las computadoras de menor tamaño tales como los llamados Fit PC o también las SBC (Single Board Computer), como por ejemplo la Raspberry PI, Cubie-Board, Galileo, etc. Los SBC cuentan con los mismos puertos de entrada que una computadora común, pero además presentan otras particularidades que serán revisadas más adelante.

El objetivo de este trabajo es presentar una solución para la integración de las mediciones hechas por el par QDC-SBC a una red de instrumentación. En la figura (??) se puede ver un esquema de la solución a implementar.

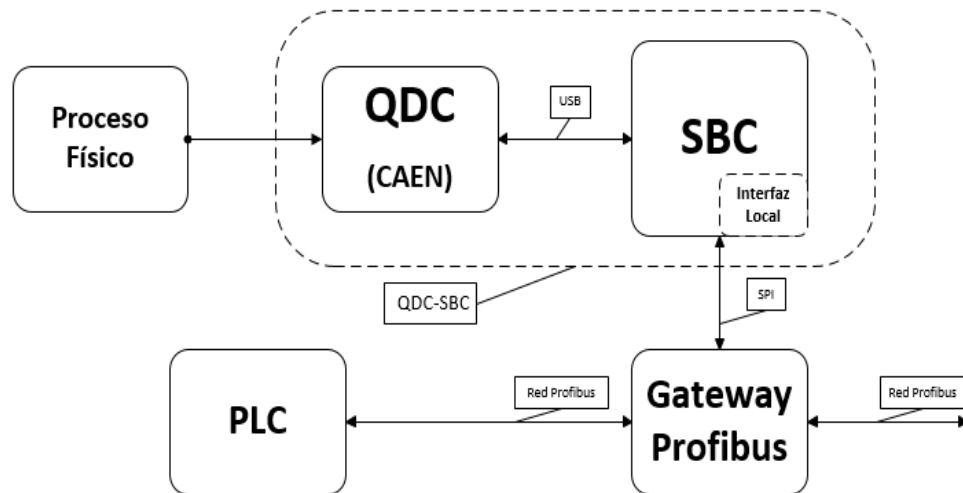


Figura 1.1: Modelo Solución

Los dispositivos que se utilizan son QDC, SBC, microcontrolador y ASIC, los cuales fueron investigados y seleccionados en los trabajos "Alternativas de Solución" y "Solución Seleccionada"[?] para la asignatura ELO-307. Estos elementos son los que cumplen de mejor manera los criterios de accesibilidad, costos, tiempos de desarrollo y módulos de comunicación.

En los siguientes capítulos, se hace una breve descripción de las problemáticas solucionadas que contribuyen a la solución propuesta.

1.2 Requerimientos Funcionales

En este trabajo se presenta una solución para la integración del par QDC-SBC en una red de instrumentación industrial. Para realizar esto se deben cumplir los siguientes requisitos:

- Comunicar el dispositivo QDC a una SBC para la adquisición y procesamiento de datos.
- Comunicar dichos datos a una red de instrumentación industrial.
- Permitir la configuración junto con la adquisición y procesamiento de datos.
- Permitir al usuario establecer localmente los parámetros para el esclavo Profibus, mediante una interfaz apropiada.

1.3 Resultados Esperados

Este trabajo pretende integrar un transductor QDC que convierte la carga eléctrica en una palabra digital de 14 bits, utilizado en diferentes sistemas de medición, como por ejemplo espectroscopia de rayos gamma, a una red de instrumentación industrial Profibus DP, con el fin de facilitar su integración a sistemas SCADA-HMI industriales. Los principales objetivos son:

- Integrar un dispositivo QDC (CAEN) a un SBC.
- Comunicar el par QDC-SBC con una red de instrumentación industrial Profibus DP mediante un microcontrolador y un dispositivo ASIC esclavo.

- Desarrollar una aplicación embebida en el computador de bajo costo para configurar remotamente los parámetros de comunicación del esclavo en la red de instrumentación, mediante TCP/IP.
- Desarrollar una interfaz de usuario en el computador de bajo costo que permita configurar localmente los parámetros de comunicación del esclavo en la red de instrumentación.

Capítulo 2

Red Profibus DP

El protocolo de comunicación industrial Profibus es un sistema de comunicación digital abierto, y su uso se ha masificado principalmente como bus de campo en la industria. Esto se debe a que el protocolo cuenta con características que permiten cubrir aplicaciones rápidas, con tiempos de reacción cortos así como también complejos sistemas de comunicación. Este protocolo está cubierto por el estándar internacional IEC 61158, que agrupa a varios buses de campo, siendo el tipo 3 el correspondiente a Profibus. Dicho estándar está pensado para ser utilizado junto con el estándar IEC 61784, el cual especifica las propiedades del bus de campo para Profibus.

Actualmente el uso de Profibus en la industria es muy importante y su incorporación hacia Ethernet a través de Profinet, hace pensar que seguirá vigente.

2.1 Características Técnicas

La transmisión de datos puede ser realizada por distintas tecnologías, las cuales están resumidas en el cuadro (??).

Cuadro 2.1: Tecnología de Transmisión.

RS485	Velocidades hasta 12Mbps, sobre un par trenzado blindado.
RS485-IS	Diseñado para su utilización en lugares potencialmente explosivos (IS: intrinsically safe), funcionando con baja corriente y bajos voltajes. Trabaja sobre dos pares trenzados.
MBP	Codificación Manchester con poder en el bus. Trabaja sobre un par trenzado en que van datos y poder. Velocidades de hasta 31,5 kbps y existe también en version IS.
Fibra Óptica	Para utilizar donde haya mucho ruido electromagnético o se requiera alcanzar mayores distancias.

Para acceder al medio se distinguen dos tipos de dispositivos, Maestros y Esclavos. Los dispositivos maestros pueden acceder activamente tomando el control del medio, por turnos regidos a través de un anillo lógico mediado por un token. En cambio los dispositivos esclavos sólo pueden acceder al medio en respuesta a alguna petición de un Maestro. El sistema y los dispositivos son mostrados en la figura (??).

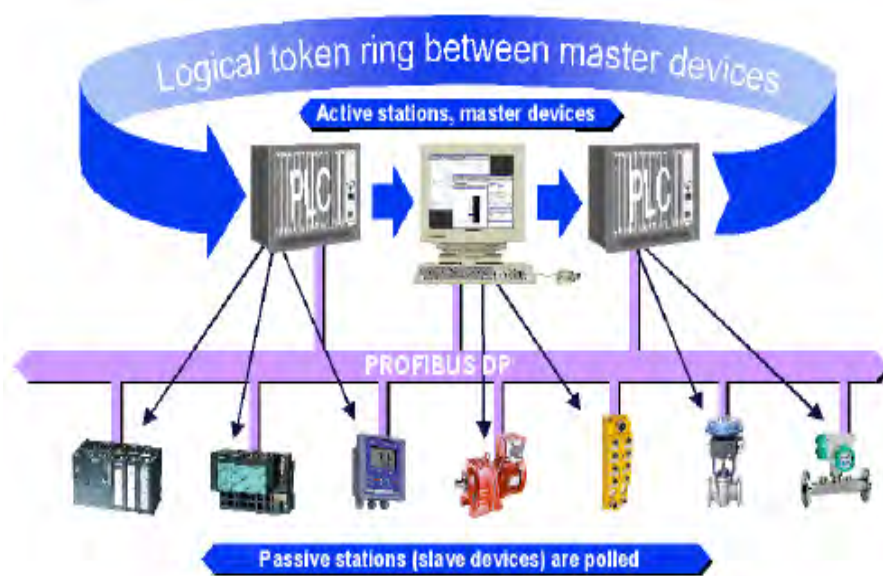


Figura 2.1: Topología de la Red

En la figura anterior se muestra la topología de la red identificando las estaciones Maestras y Esclavas en Profibus DP. A lo largo de su desarrollo han aparecido mejoras en el protocolo, generando nuevas versiones. A continuación se muestran las principales características de cada versión DP, junto con un cuadro explicativo mostrado en la figura (??).

FMS: Se agrega por razones históricas ya que estaba diseñado para la comunicación entre un PLC con otros dispositivos mayores como PCL's o PC's.

DP-V0: Provee las funciones básicas de la comunicación cíclica, incluyendo los servicios de diagnóstico de dispositivo, módulo y específicos del canal.

DP-V1: Contiene extensiones orientadas hacia las necesidades de la automatización de procesos, especialmente al acceso de información acíclica, paralela a la comunicación cíclica. Esta vía de información acíclica permite entre otras, la

anexión de alarmas.

DP-V2 Agrega más funcionalidades y opciones particularmente útiles en el control de motores. Las extensiones más importantes son el modo Esclavo Isócrono y la comunicación Esclavo-Esclavo a través de sistemas de Broadcast.

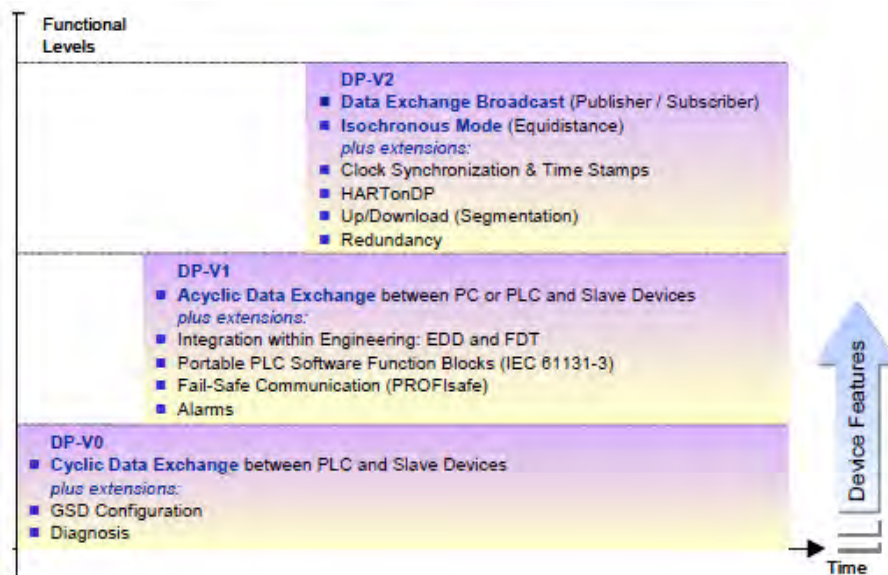


Figura 2.2: Funciones de versiones DP

2.2 Modelo OSI/ISO

Según se menciona anteriormente, Profibus está definido por el estándar IEC 61158, en el que el tipo 3 corresponde a Profibus. Este estándar viene separado en seis secciones, las cuales definen las distintas capas del sistema (excepto la primera que solo da un reporte técnico de las utilidades de las demás partes). La tabla (??) muestra la correspondencia del modelo OSI con el estándar, en la cual

se puede ver que las capas del modelo OSI desde Red hasta presentación son cubiertas en parte por la capa de enlace y en parte por la capa de aplicación del IEC 61158. Además la figura (??) muestra cómo se integra el estándar (capas de la 2 a la 6) más perfiles a la red Profibus.

Cuadro 2.2: Correspondencia entre Profibus y OSI.

Modelo OSI	Descripción	IEC 61158
7 Aplicación	Interfaz con Aplicación	5 y 6.
6 Presentación	Representación de información	
5 Sesión	Sincronización de los procesos comunicativos	
4 Transporte	Manejo de errores, separación en paquetes	-
3 Red	Establecer conexiones, evitar congestión	
2 Enlace	Descripción del Control de Acceso al Medio	3 y 4
1 Física	Definición del medio de transmisión de datos	2

La figura (??) muestra las distintas capas en el protocolo.

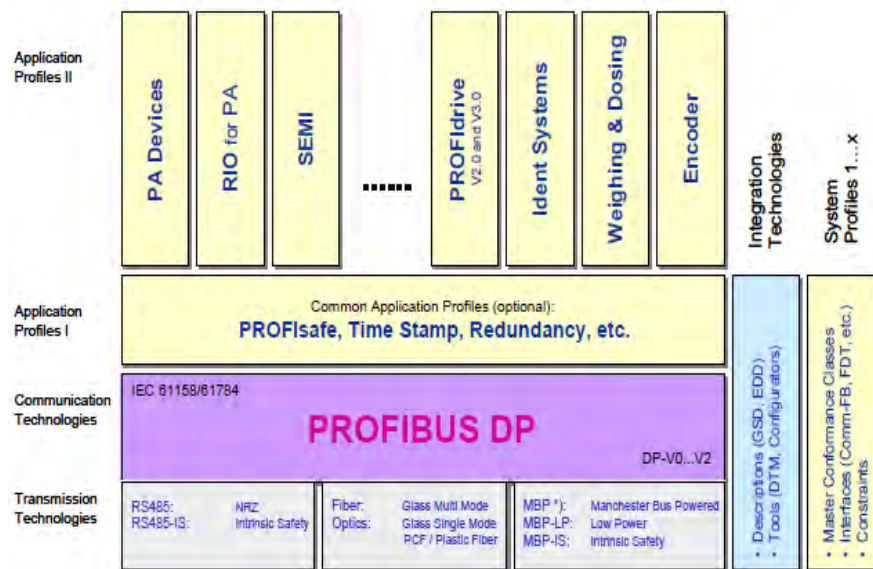


Figura 2.3: Estructura Profibus

2.3 Archivo GSD

Como parte del estándar se encuentra el archivo GSD [?], el cual detalla gran parte de las funcionalidades del Esclavo (o Maestro). A pesar que el archivo se denomina GSD, en realidad es solo uno de los casos, ya que el nombre del archivo es GS, siendo la ultima letra el idioma en que esta hecho. Así es como la D es para Default (por defecto en inglés), por ejemplo también hay archivos en alemán (German) e inglés (English).

El formato de este archivo es ASCII sin codificación especial y con la posibilidad de de agregar comentarios.

La información contenida en este archivo es la siguiente:

Capa Física

- Velocidades soportadas por la estación.
- Autodetección de velocidad soportada.

Capa Enlace

- MaxTSDR a cada velocidad (Máximo tiempo que un esclavo demora en responder).
- Hardware y Software release, Vendor, etc.

Capa de Aplicación

- Soporte de Freeze (establece entradas en cache para los esclavos con dirección de grupo específica), Sync (establece salidas en cache para los esclavos con dirección de grupo específica), Fail-Safe (modo a prueba de fallos).
- Largos máximo de información de diagnóstico.
- Estructuras de información.
- Largos de entradas y salidas, por módulos si corresponde.
- Extensiones DP-V1: Soporte para alarmas, modo isócrono, etc.
- Resolución y funcionamiento del Watch Dog (1 o 10 ms).
- Soporte de Funciones Editor/Subscriber

Este archivo tiene información no específica del protocolo y del equipo, como las imágenes de los iconos a mostrar y el significado de la información de diagnóstico y el máximo largo de la información de parametrización. Existen varias versiones de este archivo, las que incluyen mayor o menor información respecto del desempeño temporal de esclavo.

Capítulo 3

Arquitectura Propuesta

La arquitectura propuesta para realizar la integración del QDC a la red Profibus DP es la mostrada en la figura (??

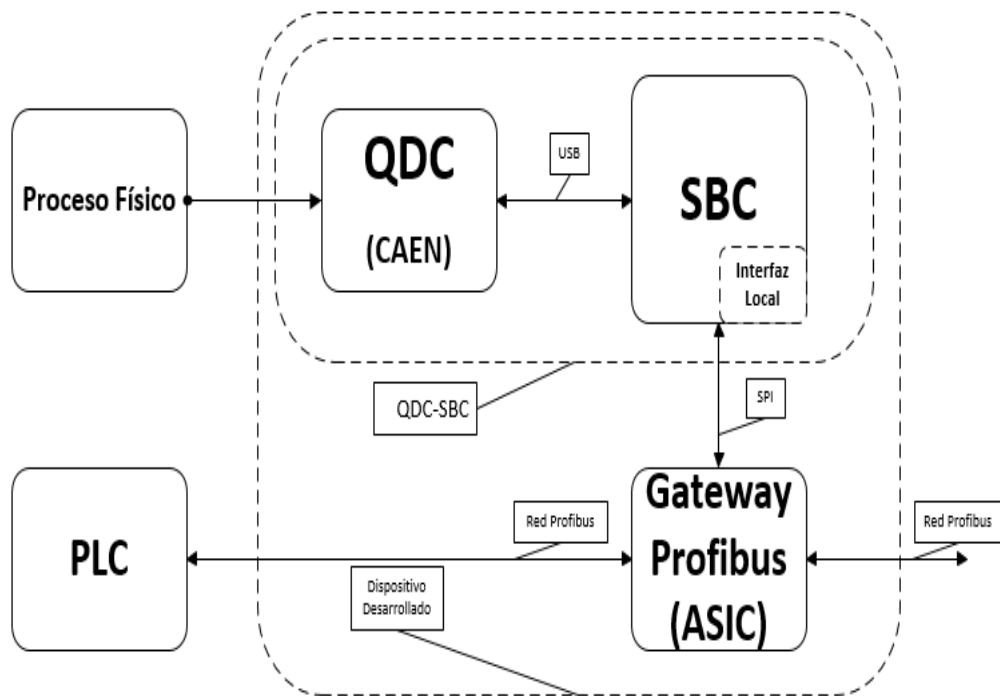


Figura 3.1: Modelo Solución

En las siguientes secciones se revisa cada uno de los módulos identificados en la figura (??).

3.1 Gateway Profibus

El Gateway Profibus es el dispositivo que permite enviar datos a la red de instrumentación, independiente desde donde provengan éstos, siempre y cuando puedan ser leídos por el Gateway. Existen trabajos como el de Diego Valencia, "Desarrollo de un Dispositivo Esclavo Profibus-DP en base al Estándar IEC61158"[?], en el cual se diseña y construye un dispositivo similar. En este trabajo se utiliza

un dispositivo basado en el ASIC VPC3+S[?] de Profichip, el cual se encarga de implementar la capa física y de enlace para el protocolo Profibus DP. La capa de Aplicación es realizada por un microcontrolador Atmega 1280. La figura (??) muestra la arquitectura seleccionada para este Gateway.

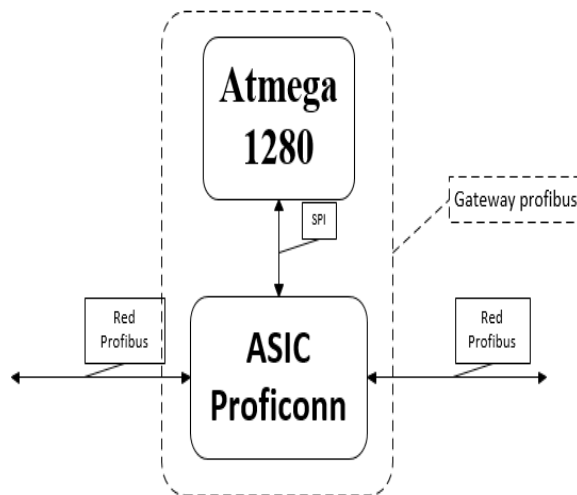


Figura 3.2: Gateway Profibus

A continuación se revisan cada uno de los dispositivos involucrados.

3.1.1 Proficonn de Profichip

El módulo Proficonn de Profichip es un dispositivo Esclavo basado en el ASIC VPC3+S, el cual es un chip de comunicación con la red Profibus DP en todas sus versiones. Dicho ASIC maneja los mensajes y la identificación de dirección, los datos de la secuencia de seguridad y el procesamiento para el protocolo Profibus

DP. Además este chip soporta todas las versiones para Profibus DP (V0,V1,V2). La velocidad máxima de comunicación es de 12[Mbit/s], la cual abarca todas las velocidades para el protocolo. En la figura (??) se muestra un diagrama con los componentes del ASIC.

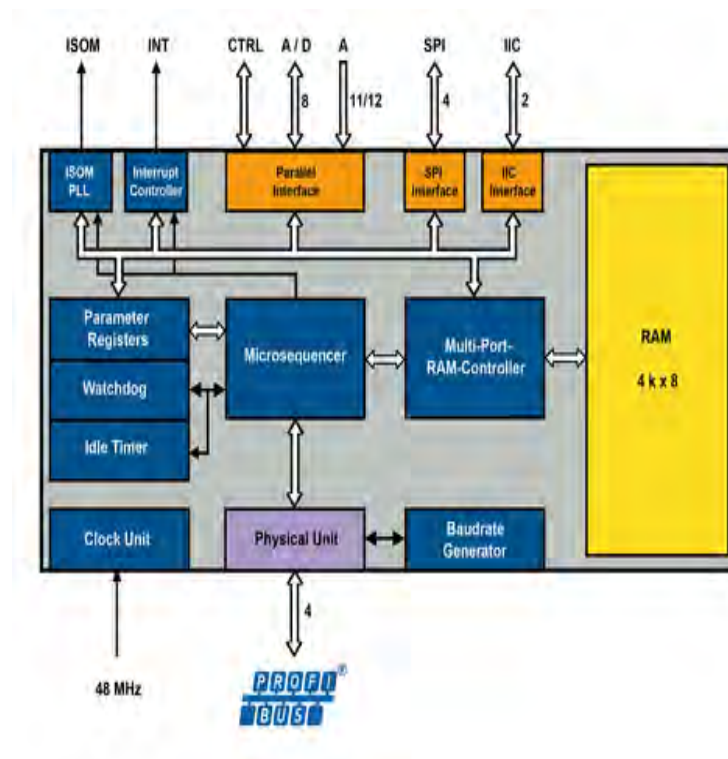


Figura 3.3: ASIC VPC3+S

Cuadro 3.1: Función de Bloques del ASIC.

Bloque	Función
1.ISOM PLL	Provee mecanismos de sincronización de alta precisión, definidos en Profibus DPV2
2.Interrupt Controller	Varios eventos son manejados por esta unidad. Estos eventos son habilitados por las mascara de registro. El VPC3+S tiene una salida común de interrupción.
3.Parallel Interface	Módulo de comunicación paralela con el VPC3+S.
4.SPI Interface	Módulo de comunicación serial SPI.
5.I2C Interface	Módulo de comunicación serial I2C.
6.RAM	Éste opera como puerto de doble RAM y contiene parámetros específicos del protocolo, así también como los buffers de datos.
7.Parameter Register	Almacena parámetros específicos del protocolo.
8.Watchdog Timer	Es el encargado de "descolgar" el ASIC. Lo hace por medio de las velocidades de comunicación, de control o controles del protocolo Profibus DP.
9.Idle timer	Controla directamente los tiempos del bus en la línea serial.
10.clock unit	Proporciona los pulsos de Reloj a los que trabaja el ASIC.
11.Microsecuenser	Controla el proceso completo, en el se encuentra la máquina de estados de Profibus DP Esclavo.
12.Multi-Port-Ram-Controller	Controla la comunicación entre el Microsecuenser y la RAM.
13.Baudrate Generator	Genera las velocidades de comunicación del protocolo.
14.Physical Unit	Conexión con la red Profibus DP.

En la figura (??) se puede identificar cada uno los 14 componentes del ASIC así como también la unidad física, la cual es la que se conecta con una etapa de adecuación de señales para cumplir con las especificaciones de la capa física del protocolo. En las figuras (??), (??) se muestran la implementación modular del Proficonn y su imagen real.

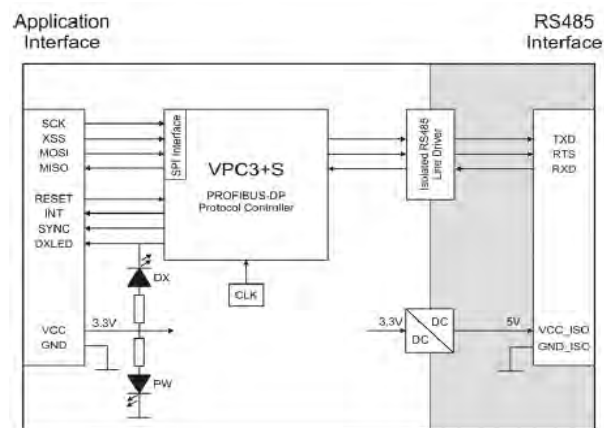


Figura 3.4: Diagrama de Bloques Proficonn DSUB (VPC3+S)



Figura 3.5: Proficonn DSUB (VPC3+S)

Como se menciona anteriormente este dispositivo se ocupa de las capas 1 y 2 del modelo OSI, por lo que se necesita un dispositivo externo que se ocupe de la capa 7, por medio de comunicación serial SPI. Este dispositivo externo sirve además para implementación de los diferentes versiones del protocolo DP (V0, V1, V2), los cuales deben ser configurados mediante los registros del ASIC. Además deben ser manejados los datos de configuración, parametrización y datos, entre otros. Todo esto es realizado por un microcontrolador o dispositivo compatible.

3.1.2 Microcontrolador Atmega 1280

Para complementar lo realizado por el dispositivo Proficonn, se utiliza el microcontrolador Atmega 1280. Éste es el encargado de realizar la capa de aplicación dentro del protocolo. En el cuadro (?? se muestran las principales características de este microcontrolador.

Este microcontrolador se encuentra inserto en una las tarjetas de desarrollo Arduino, específicamente Arduino Mega, la cual le agrega la posibilidad de comunicación vía USB. La figura (?? muestra esta tarjeta de desarrollo.

Cuadro 3.2: Especificaciones Atmega1280.

Atmega 1280	
Procesador	8 bit avr
Flash	128 kbytes
SDRAM	8 kbytes
EEPROM	4096 bytes
Max. Frecuencia de Operación	16 Mhz
GPIO	86 pins
Comunicación	SPI/UART/TWI(I2C)
Canales ADC	16 (10 bits)



Figura 3.6: Arduino Mega

Esta tarjeta puede ser programada de varias formas. Una de ellas es la que se puede realizar mediante la IDE de Arduino, el cual contiene librerías para cada módulo del microcontrolador. Otra de las posibilidades de programación es mediante el software WinAvr [?], el cual instala una serie de librerías (como las de AVR, para el manejo de funciones del microcontrolador) y aplicaciones necesarias para la compilación cruzada y carga de los programas vía USB.

3.2 Aplicación QDC

Esta aplicación QDC es un conjunto formado por electrónica para adecuación de señales (vistas en los trabajos de Rene Ríos[?] y Lautaro Narvaez[?]), las cuales son conectadas a un digitalizador, específicamente CAEN DT-5427[?], el cual se encarga de enviar los datos vía USB. La diferencia con los trabajos mencionados anteriormente, es que en este caso no se utilizan computadores de escritorio sino que un SBC. La figura (??) muestra un esquema de lo descrito.

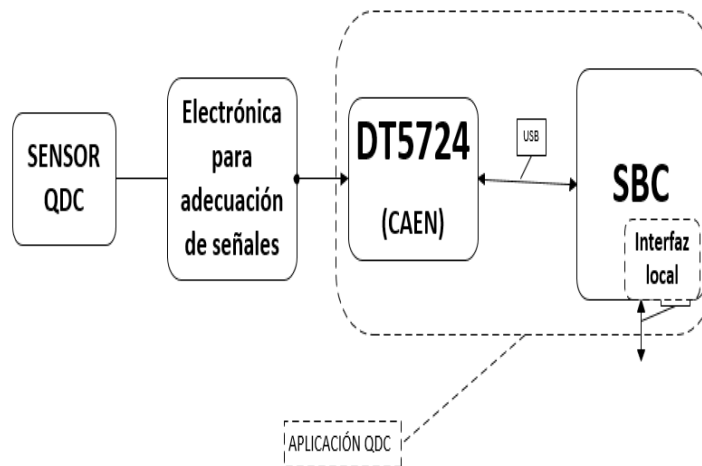


Figura 3.7: Aplicación QDC

Para la conexión del Digitalizador DT5724 y el SBC, es necesario establecer qué dispositivos compatibles existen dadas las restricciones que presenta el DT5724. Estas restricciones son revisadas en secciones posteriores. En relación a la utilización de SBC y su conexión con periféricos existen los trabajos de Juan Nuñez[?] y el de Carlos Valdivieso[?], en los cuales se ve cómo conectar un adquisidor de datos con Arduino y una Raspberry Pi.

Como la electrónica para la adecuación de las señales del sensor QDC ha sido estudiada en los trabajos mencionados anteriormente, no se estudiará en este trabajo.

3.2.1 Digitalizador DT 5724

El Digitalizador utilizado para esta aplicación es el modelo DT 5724 de la empresa CAEN, la cual se especializa en el desarrollo de digitalizadores de alta velocidad y precisión. La tabla (?? muestra las principales características de este dispositivo.

Cuadro 3.3: Especificaciones DT5724.

Canales	4
Velocidad de conversión	100 MS/S
Precisión	14 bits
Comunicación	USB & Optical Link

En las figuras (??) y (??) se muestra este digitalizador, junto con un diagrama de bloques donde se muestran los módulos que lo componen.



Figura 3.8: DT5724.

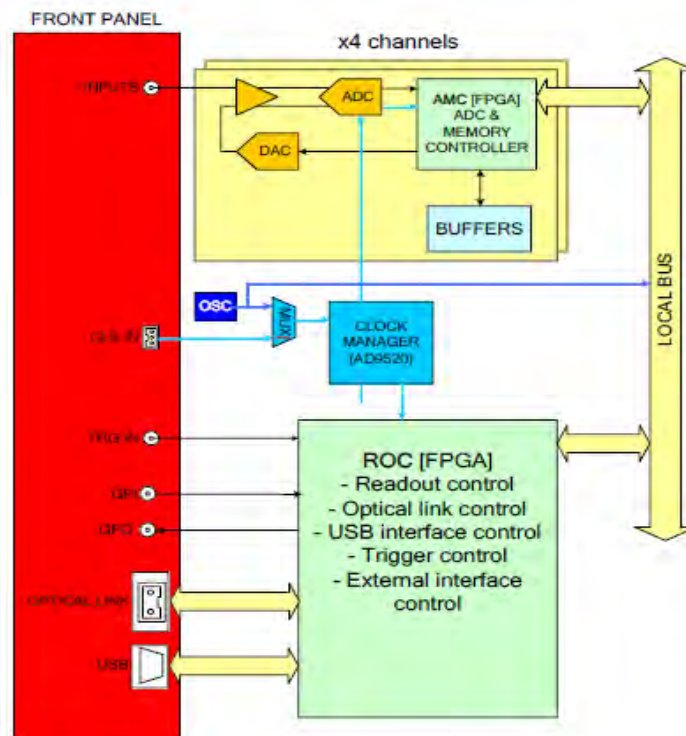


Figura 3.9: Diagrama de Bloques DT5724.

Para establecer la comunicación con este dispositivo es necesario instalar en el computador a utilizar los driver y librerías necesarios, los cuales se encuentran disponibles en la página web del fabricante. Estos archivos no pueden ser cargados en cualquier computador, el cual debe cumplir con arquitectura compatible de x86 y x64 bits para SO Windows y Linux, además de ser necesario Kernel 2.4 a 3.X.

3.2.2 Computador On-Board para Adquisición de Datos

Este dispositivo es un SBC, los cuales son tarjetas de desarrollo basados en SoC (System on a Chip). Los SoC son chips basados en microcontroladores, y cuentan con módulos para el manejo de señales digitales, análogas o mixtas (comunicación Bluetooth, Wifi, Wireless, etc), así como también ciertos módulos de comunicación como UART, SPI, I2C, USB, PCI, entre otros. Además en el mismo chip se pueden encontrar memorias para el intercambio de datos y procesamiento de éstos.

Existen grandes diferencias entre lo que puede hacer un SoC y un microcontrolador, tales como las velocidades de intercambio de datos, la capacidad de almacenamiento y la posibilidad de correr y cargar sistemas operativos (SO), drivers y programas no nativos en el SoC del SBC. Estos SBC se presentan en forma de tarjetas de desarrollo similares a arduinos. En el mercado existe un gran número de SBC, pasando desde las más populares como la Raspberry PI, BeagleBone, CubiBoard, o una de las más recientes como la Galileo[?] de Intel. Cada una de éstas presenta una gran cantidad de módulos de salida para señales análogas, digitales y mixtas, así como también módulos de comunicación para el uso de

periféricos en los computadores como entradas PCI express, USB, Serial RS-232, lectura de tarjetas SD, entre otros. Como se puede suponer, de todas las alternativas que existen, no se puede escoger cualquiera y utilizarla, debido a que difieren tanto en la cantidad de módulos disponibles como en sus arquitecturas, lo que repercute en su SO, por lo que antes se debe analizar cuáles satisfacen los requisitos para la solución. Según lo revisado en .Alternativas de Solución ”[?] se puede establecer que una de las características más importantes para la selección de este dispositivo es la arquitectura y el SO que presente. A continuación en la figura (??) se muestra el SBC seleccionado junto con la tabla (??) que muestra un resumen de este dispositivo.

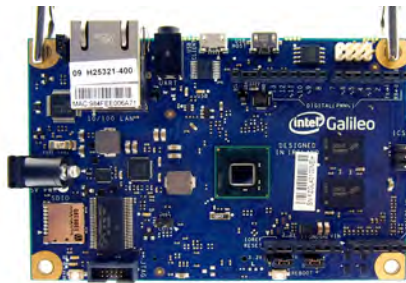


Figura 3.10: Galileo.

Cuadro 3.4: Especificaciones Galileo.

Procesador	Intel® Quark™ SoC X1000 [?]
Arquitectura	32 bits
GPIO	18 pins
Módulos	SPI/UART/USB

3.3 Controlador Lógico Programable (PLC)

Para poder realizar las pruebas de comunicación del dispositivo en la Red Profibus DP se requiere utilizar un maestro de red certificado. En este caso se elige un PLC Siemens S7-300 disponible en el Laboratorio de Control Industrial.

El Controlador Lógico Programable, o PLC por sus sigla en inglés (Programmable Logic Controller), es una máquina programable que cuenta con periféricos para la entrada y salida de señales, así como también módulos de comunicación tanto para la programación como para el intercambio de datos en red. Actualmente está regido por el estándar internacional IEC-61131, el cual pretende regular siete aspectos del PLC y sus periféricos asociados. Estos aspectos son:

- Visión General.
- Hardware.
- Lenguaje de Programación.
- Guías de Usuario.
- Comunicación.
- Control Difuso.
- Guías para el diseño, implementación y uso de lenguajes de programación.

El mismo estándar identifica en la estructura física del PLC cinco bloques funcionales básicos. La figura (??) muestra estos bloques.

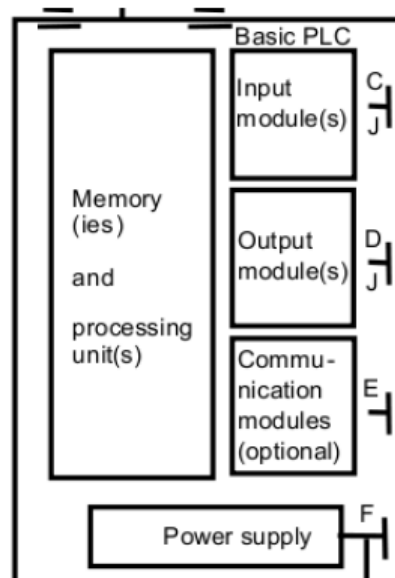


Figura 3.11: Bloques Funcionales.

Dependiendo de cómo se encuentren distribuidos estos bloques, el PLC se clasifica como Modular o Compacto. El PLC Compacto tiene todos los bloques funcionales en la misma unidad física (caja, gabinete, etc.) y no es posible modificar alguno de ellos. El PLC Modular, por el contrario, tiene sus bloques funcionales separados, lo que permite que se puedan elegir a voluntad la cantidad de entradas, salidas, el bloque CPU, el de comunicación e incluso el de alimentación, según los requerimientos.

Para realizar las pruebas de este trabajo se selecciona el PLC S7-300 con CPU 312-2DP de la marca Siemens, el cual es Modular y cuenta con módulos de comunicación MPI y Profibus DP, además de sus módulos de salida y entrada. En la figura (??) se muestra una imagen de éste.



Figura 3.12: PLC S7-300.

Capítulo 4

Dispositivo Desarrollado

El desarrollo de este dispositivo, ya sea por su naturaleza o por los diversos elementos con los que cuenta, debe ser realizado por etapas. Cada una de las etapas no están estrictamente relacionadas entre sí, aunque la suma de todas conforman la solución final. La figura (??) muestra el modelo de solución separado en las distintas etapas.

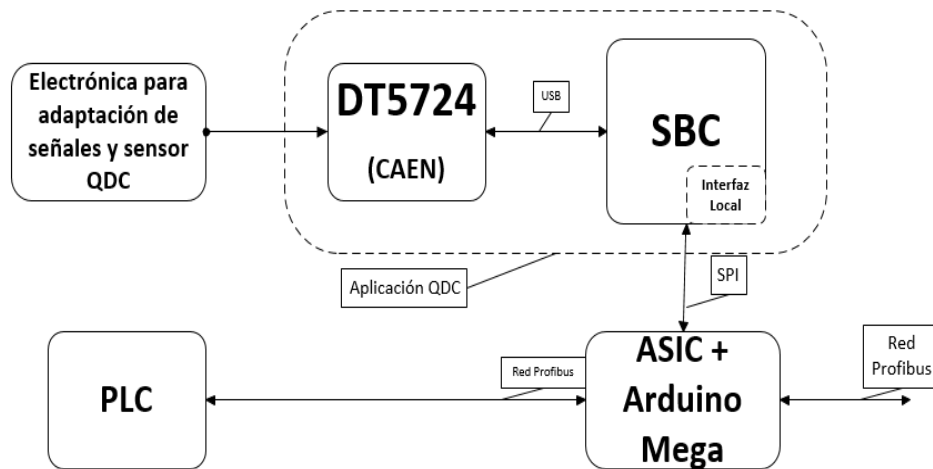


Figura 4.1: Modelo Solución por etapas.

4.1 Comunicación entre DT5724 y SBC

En capítulos anteriores se vieron cuáles son las características principales del DT5724 y SBC a utilizar. Como ya se sabe el fabricante del digitalizador entrega drivers y librerías compilados para computadores con arquitectura de 32 bits y 64 bits. Si bien Intel proporciona un SO llamado Linux Yocto, éste no cuenta con las cabeceras del kernel (al momento de escribir esta tesis), las cuales son necesarias para instalar drivers y librerías de dispositivos externos. Para solucionar este problema se debe realizar una compilación cruzada del kernel (provisto por Intel) en algún SO como Debian[?], lo cual es un proceso largo y complicado. En el foro web de desarrolladores de Intel[?], aparece un método más corto y menos complejo por el cual se puede crear un SO basado en Debian, compatible con el dispositivo Galileo, así como también están las direcciones para los repositorios

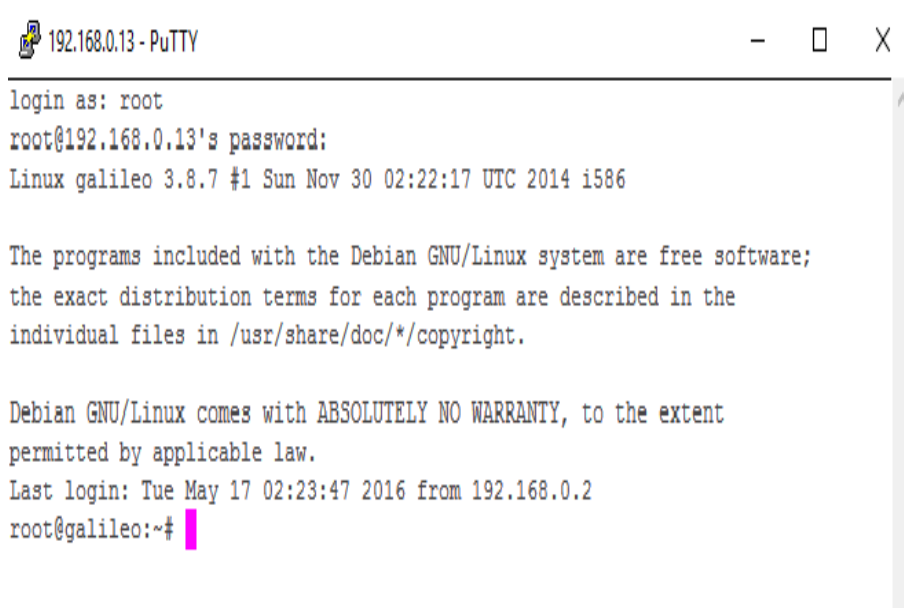
necesarios en la instalación de las cabeceras del kernel.

El método mencionado anteriormente consiste en varias instrucciones a realizar las cuales serán resumidas a continuación (para mayor información dirigirse a [?]).

- Instalar en una MV, con las especificaciones necesarias como arquitectura, memoria RAM y almacenamiento, un SO Debian compatible con arquitectura 32 bits.
- Una vez instalado el SO, entrar en él y descargar la imagen de Linux-Yocto proporcionada por Intel en el link http://downloadmirror.intel.com/23171/eng/LINUX_IMAGE_FOR_SD_Intel_Galileo_v0.7.5.7z y montar la imagen *image – full – clanton – original.ext3* en algún directorio predefinido.
- Se debe preparar el SO Debian instalado creando ficheros destino y copiar ciertos archivos provenientes de la imagen montada en el paso anterior. Los archivos están detallados en el link [?] y en el Anexo A.
- Una vez copiados los archivos y montada la imagen en la SD Card, poner en el Galileo, esperar a que inicie desde la tarjeta SD y acceder al SO. Una vez dentro se deben instalar paquetes para programación y actualizaciones.
- Para acceder al SO se puede hacer mediante el protocolo SSH[?] o con un cable serial para consola (que debe ser construido) para la conversión serial RS232 a USB, según lo mostrado en el Anexo B.
- Para instalar el repositorio donde se encuentran las cabeceras del kernel se debe ir al directorio */etc/apt/source.list.d/* y crear los archivos *galileo.list* y *galileo.list.dpkg – dist* y poner en estos archivos

`"deb http://galileodebian.sourceforge.net/debian galileo main"` y `"deb http://galileo.netweng.com/debian galileo main"`. Una vez creado todo se teclea en el Terminal los siguientes comandos `"apt-get update"` y `"apt-get upgrade"`. Para finalizar se instalan las cabeceras del kernel tecleando `"apt-get install linux-headers"`.

Con el procedimiento anterior ya está preparado el Galileo para la instalación de los drivers y librerías del Digitalizador Caen. La figura (??) muestra el resultado de la preparación del Galileo.



```
192.168.0.13 - PuTTY
login as: root
root@192.168.0.13's password:
Linux galileo 3.8.7 #1 Sun Nov 30 02:22:17 UTC 2014 i586

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 17 02:23:47 2016 from 192.168.0.2
root@galileo:~#
```

Figura 4.2: Galileo con Debian.

De cualquier forma, si algo sale mal en los procedimientos anteriores, en el siguiente link <https://sourceforge.net/p/galileodebian/wiki/Home/> se puede descargar una imagen de Debian para Galileo con los repositorios ya incluidos para la instalación de las cabeceras del kernel. Sólo falta poner

los comandos *"apt –get update"*, *"apt –get upgrade"* y *"apt –get install linux – headers"* en el Terminal de Linux.

Una vez realizada esta operación ya podemos dar inicio a la instalación de los drivers y librerías del DT5724, las cuales se pueden obtener directamente de la página del fabricante las cuales deben ser instaladas de la siguiente forma:

Con los archivos CAENComm, CAENDigitizer, CAENUSBdrvB y CAENVME-Lib descargados, se descomprimen y luego dirigirse a la carpeta de instalación de cada uno. Se instalan dando en la shell *"sh install"*. El orden debe ser el siguiente:

- 1:** CAENUSBdrvB
- 2:** CAENVMELib
- 3:** CAENComm
- 4:** CAENDigitizer

Las figuras (??), (??), (??), (??) muestran cada uno de estos pasos.

```

login as: root
root@192.168.0.13's password:
Linux galileo 3.8.7 #1 Sun Nov 30 02:22:17 UTC 2014 i586

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 17 02:29:10 2016 from 192.168.0.2
root@galileo:~# cd /home/CAEN/
root@galileo:/home/CAEN# cd CAENUSBdrvB-1.4/
root@galileo:/home/CAEN/CAENUSBdrvB-1.4# make
make -C /lib/modules/3.8.7/build M=/home/CAEN/CAENUSBdrvB-1.4 LDDINCDIR=/home/CAEN/CAENUSBdrvB-1.4/./include modules
make[1]: Entering directory `/usr/src/linux-headers-3.8.7'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory `/usr/src/linux-headers-3.8.7'
root@galileo:/home/CAEN/CAENUSBdrvB-1.4# make install
installing CAENUSBdrvB driver.. please wait
installation done
root@galileo:/home/CAEN/CAENUSBdrvB-1.4#

```

Figura 4.3: Instalación Driver USB.

```

installation done
root@galileo:/home/CAEN/CAENUSBdrvB-1.4# cd ../CAENVMELib-2.41/lib/
root@galileo:/home/CAEN/CAENVMELib-2.41/lib# install
install: missing file operand
Try 'install --help' for more information.
root@galileo:/home/CAEN/CAENVMELib-2.41/lib# sh install
root@galileo:/home/CAEN/CAENVMELib-2.41/lib#

```

Figura 4.4: Instalación CAENVMELib.

```

root@galileo:/home/CAEN/CAENVMELib-2.41/lib# cd ../../CAENComm-1.2/lib/
root@galileo:/home/CAEN/CAENComm-1.2/lib# sh install
root@galileo:/home/CAEN/CAENComm-1.2/lib#

```

Figura 4.5: Instalación CAENComm.

```

root@galileo:/home/CAEN/CAENDigitizer_2.6.0# sh install
root@galileo:/home/CAEN/CAENDigitizer_2.6.0#

```

Figura 4.6: Instalación CAENDigitizer.

Ya con los drivers y librerías instaladas podemos revisar la lista de módulos instalados en el SO dando el comando "lsmod" en la shell, el cual muestra todos los módulos disponibles en el kernel. La figura (??) muestra esto.

```

root@galileo:~# lsmod
Module                  Size  Used by
CAENUSBdrvB             12942  0
ip6t                     256918  16
g_serial                24389  2
libcomposite            24102  1 g_serial
pch_udc                 31633  0
udc_core                14227  2 libcomposite,pch_udc
ad7298                  12726  0
industrialio_triggered_buffer 12686  1 ad7298
kfifo_buf              13032  1 industrialio_triggered_buffer
industrialio            39078  3 industrialio_triggered_buffer,ad7298,kfifo_buf
spidev                  13101  0
at24                    12977  0
cy8c9540a               17609  0
spi_pxa2xx              21688  0
gpio_sch                18016  2
intel_qrk_gip           24246  4
uio                     14405  2 intel_qrk_gip,gpio_sch
spi_pxa2xx_pci          13049  1 spi_pxa2xx
evdev                   17469  0
ehci_pci                12656  0
ohci_hcd                26510  0
ehci_hcd                40372  1 ehci_pci
usbcore                 144763  4 ohci_hcd,CAENUSBdrvB,ehci_hcd,ehci_pci
stmmac                  52630  0
usb_common              12482  3 udc_core,libcomposite,usbcore
root@galileo:~#

```

Figura 4.7: Verificación de Módulos.

4.2 Módulo Profibus

Como se ve en capítulos anteriores, la solución para la integración a la red de instrumentación Profibus está compuesta por un Arduino Mega y un dispositivo basado en el ASIC VPC3+S.

Lo primero que se debe hacer es descargar el archivo que contiene los manuales de usuario del ASIC, ejemplos del software para distintas tarjetas de desarro-

llo, junto con otro software disponible para la utilización personalizada del software. Desde el siguiente link [http://www.profiship.com/products/overviewasics/dp-slave-vpc3-c/dp-v0-firmware/?L=5\[?\]](http://www.profiship.com/products/overviewasics/dp-slave-vpc3-c/dp-v0-firmware/?L=5[?]) se pueden descargar los archivos.

Una vez descargados los archivos se abre el archivo "VPC3+SoftwareDescriptionV600.pdf" [?], el cual muestra una descripción completa del funcionamiento del software, se abre el archivo y se va al capítulo 3 y se modifican los valores necesarios en el archivo platform.h. La figura (??) muestra la configuración para este proyecto.

```
#define TRUE 1
#define FALSE !(TRUE)

#define BOOL unsigned char
#define uint8_t unsigned char
#define uint16_t unsigned int
#define uint32_t unsigned long

#define PTR_ATTR
#define VPC3_PTR PTR_ATTR *
#define VPC3_ADR uint16_t
#define VPC3_UNSIGNED8_PTR uint8_t PTR_ATTR *
#define NULL_PTR (void VPC3_PTR)0

#define MEM_ATTR
#define MEM_PTR MEM_PTR_ATTR *
#define MEM_UNSIGNED8_PTR uint8_t MEM_PTR_ATTR *

#define ROM_CONST__ const

#define _PACKED_

#define LITTLE_ENDIAN 0
#define BIG_ENDIAN 1

#define VPC3_ASIC_ADDRESS ((unsigned char *)0x8000)
```

Figura 4.8: Configuración platform.h

Una vez realizada esta configuración se debe ir al archivo DPCfg.h, donde se establecen los servicios disponibles para el protocolo como alarmas, sincronización, time stamp, entre otros. Estos servicios no están disponibles para el protocolo DP-V0, por lo que no se dejan activados. También se deben configurar la direc-

ción del Esclavo, el número identificador Profibus, así como el valor del registro del Watchdog en el ASIC, en caso que el microcontrolador conectado falle, antes de que el VPC3+ salga del estado de intercambio de datos. La tabla (??) muestra los valores asignados.

Cuadro 4.1: Parámetros generales de Esclavo Profibus.

Párametros generales de esclavo	tipo dato	HEX	Descripción
DP-ADDR	uint8	0x08	Dirección Profibus DP
IDENT-NR	uint16	0xF600	Número Identificador Profibus
USER-WD	uint16	0x01FF	Watchdog

Luego de configurar los parámetros generales del Esclavo se debe realizar la inicialización de los buffers de intercambio, los cuales son utilizados en los diferentes mensajes en la estructura del VPC3+. Esta inicialización consiste en proporcionar el largo de cada buffer. Estas longitudes determinan los buffers establecidos en el ASIC, por lo tanto dependen de la memoria total del ASIC (4kb). La tabla (??) muestra estos parámetros.

Cuadro 4.2: Inicialización de Buffers.

Inicialización de Buffers	tipo dato	HEX
DIN-BUFSIZE	uint8	0x02
DOUT-BUFSIZE	uint8	0x02
PRM-BUFSIZE	uint8	0x0A
DIAG-BUFSIZE	uint8	0x06
CFG-BUFSIZE	uint8	0x01

Se deben establecer también los registros del ASIC, los cuales sirven para habilitar y deshabilitar características del protocolo y del hardware, tales como freeze mode, sincronización, polarización de interrupciones, base de tiempo del usuario,

entre otros. Para más información referirse al capítulo del manual del software[?].

El cuadro (??) muestra los valores configurados para esta aplicación.

Cuadro 4.3: Registros del Hardware.

Inicialización de Registros del Hardware	tipo dato	HEX
INIT-VPC3-MODE-REG-L	uint8	0xC0
INIT-VPC3-MODE-REG-H	uint8	0x23
INIT-VPCE-MODE-REG-2	uint8	0x85
INIT-VPCE-MODE-REG-3	uint8	0x00
INIT-VPCE-MODE-IND-L	uint8	0x02
INIT-VPCE-MODE-IND-H	uint8	0x3D

El valor establecido para los registros de la tabla (??) se determina dependiendo de la versión del protocolo Profibus DP, los servicios habilitados y modos de funcionamiento del ASIC.

Para finalizar la configuración del software se deben definir en el archivo main.c las funciones que aparecen en las figuras (??),(??).

```

void DpAppl_SetResetVPC3Channel1      { void };
void DpAppl_ClrResetVPC3Channel1      { void };
void DpAppl_SetResetVPC3Channel2      { void };
void DpAppl_ClrResetVPC3Channel2      { void };
void DpAppl_EnableInterruptVPC3Channel1 { void };
void DpAppl_DisableInterruptVPC3Channel1 { void };
void DpAppl_EnableInterruptVPC3Channel2 { void };
void DpAppl_DisableInterruptVPC3Channel2 { void };
void DpAppl_EnableInterruptVPC3Sync    { void };
void DpAppl_DisableInterruptVPC3Sync    { void };
void DpAppl_EnableAllInterrupts        { void };
void DpAppl_DisableAllInterrupts        { void };
void Vpc3Wait_1ms                      { void };

```

Figura 4.9: Funciones a definir 1.

```

#ifdef VPC3_SERIAL_MODE

void Vpc3Write ( VPC3_ADR wAddress, uint8_t bData );
uint8_t Vpc3Read ( VPC3_ADR wAddress );
void Vpc3MemSet ( VPC3_ADR wAddress, uint8_t bValue,
uint16_t wLength );
uint8_t Vpc3MemCmp ( VPC3_UNSIGNED8_PTR pToVpc3Memory1,
VPC3_UNSIGNED8_PTR pToVpc3Memory2,
uint16_t wLength );

void CopyToVpc3 ( VPC3_UNSIGNED8_PTR pToVpc3Memory,
MEM_UNSIGNED8_PTR pLocalMemory,
uint16_t wLength );

void CopyFromVpc3 ( MEM_UNSIGNED8_PTR pLocalMemory,
VPC3_UNSIGNED8_PTR pToVpc3Memory,
uint16_t wLength );

#endif//#if VPC3_SERIAL_MODE

```

Figura 4.10: Funciones a definir 2.

Por último hay que configurar la variable `DpApplDefCfg`, que se puede encontrar en el archivo `"DpCfg.c"`, la cual contiene la información sobre cantidad de entradas y salidas del dispositivo, y que debe ser la misma que se encuentre en el archivo `GSD`. Para el presente caso a esta variable se le asigna el número hexadecimal `0x31` (49 decimal), donde el 3 significa que el dispositivo tiene entradas y salidas en byte, y el 1 que son 2 bytes de entrada y 2 bytes de salida.

4.2.1 GSD

Para crear el archivo GSD se utiliza el software GSD EDITOR 5.0, el cual se puede descargar desde el siguiente link http://en.pudn.com/downloads361/sourcecode/editor/detail1569779_en.html[?]. La figura (??) muestra la interfaz que presenta este software.

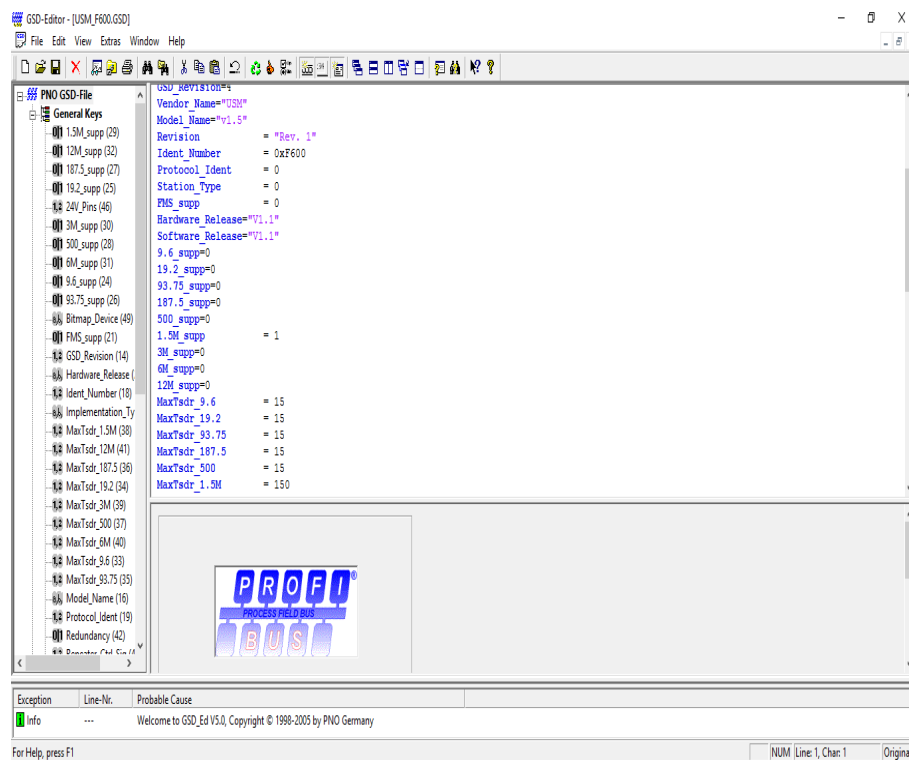


Figura 4.11: GSD Editor 5.0

Los parámetros configurados para el Esclavo son los mostrados en las figuras (??), (??).

```

#Profibus_DP
; Unit-Definition-List:
GSD_Revision=1
Vendor_Name="USM"
Model_Name="v1.8"
Revision                = "Rev. 1"
Ident_Number            = 0xF600
Protocol_Ident          = 0
Station_Type            = 0
FMS_supp                = 0
Hardware_Release="V1.1"
Software_Release="V1.1"
9.6_supp=0
19.2_supp=0
93.75_supp=0
187.5_supp=0
500_supp=0
1.5M_supp               = 1
3M_supp=0
6M_supp=0
12M_supp=0
MaxTsdr_9.6             = 15
MaxTsdr_19.2            = 15
MaxTsdr_93.75           = 15
MaxTsdr_187.5           = 15
MaxTsdr_500             = 15
MaxTsdr_1.5M            = 15
MaxTsdr_3M              = 35
MaxTsdr_6M              = 60
MaxTsdr_12M=95
Redundancy              = 0
Repeater_Ctrl_Sig       = 2
;

```

Figura 4.12: GSD File 1º parte.

```

; Slave-Specification:
24V_Pins           = 0
Implementation_Type="VPC3+ on Proficonn"
;
Bitmap_Device="DPV0_1N"
Freeze_Mode_supp=1
Sync_Mode_supp     = 1
Auto_Baud_supp     = 1
Set_Slave_Add_supp = 0
Min_Slave_Intervall = 6
Modular_Station    = 0
Fail_Safe          = 0
;
Slave_Family=3@Proficonn@DensimetroUSM@Input
Max_Diag_Data_Len=6

Max_User_Prm_Data_Len = 3
Ext_User_Prm_Data_Const(0)= 0x00,0x00,0x00

; Module-Definitions:
;
Module="2 Byte In 2 Byte Out" 0x31;
1
EndModule

```

Figura 4.13: GSD File 2º parte.

Para la selección de los valores más importantes se toman consideraciones relacionadas al protocolo, como la velocidad en que el protocolo se comunica, que en este caso es de 1.5 Mbps. Así también el identificador de la versión del protocolo a utilizar, en este caso la versión 0. El número identificador debe ser el mismo que el establecido en (??). En station type se elige el tipo de dispositivo Profibus DP (esclavo=0, maestro=1). Finalmente, la sección de definiciones del módulo se debe establecer el mismo establecido para la variable DpApplDefCfg, que para nuestro caso es 49 en decimal.

4.3 Adquisición de datos y Profibus

En los capítulos anteriores se detalla cómo es que se logra la comunicación entre el DT5724 y SBC, además de cómo configurar el módulo de los archivos del software para configurar y realizar la comunicación Profibus. Ahora se deben comunicar estos dos procesos para que puedan ser comunicados los valores sensados. Para esto se maneja un programa alojado en el SBC, el cual adquiere datos y los comunica con el módulo Profibus. La comunicación con el módulo Profibus se realiza por medio de la comunicación SPI. La figura (??) muestra el esquema de esta interacción.

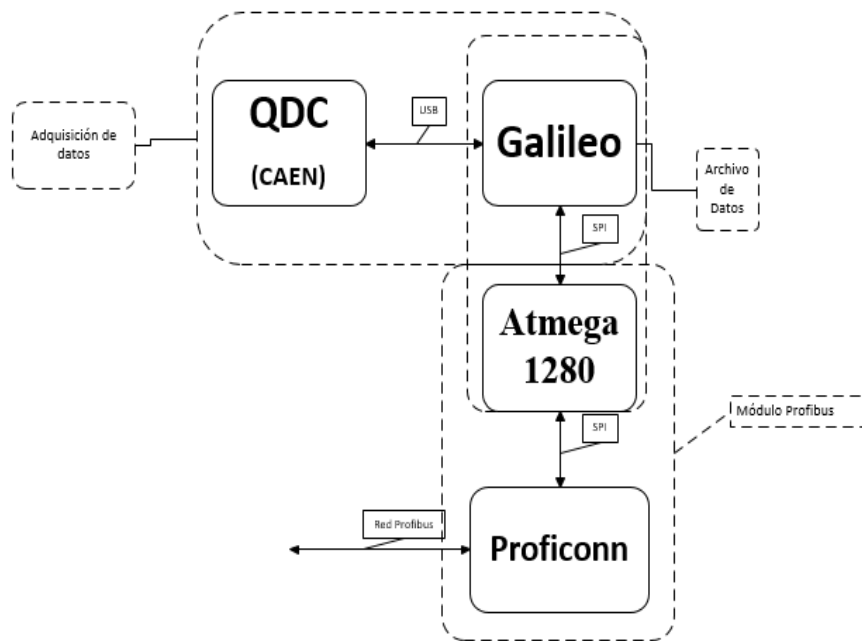


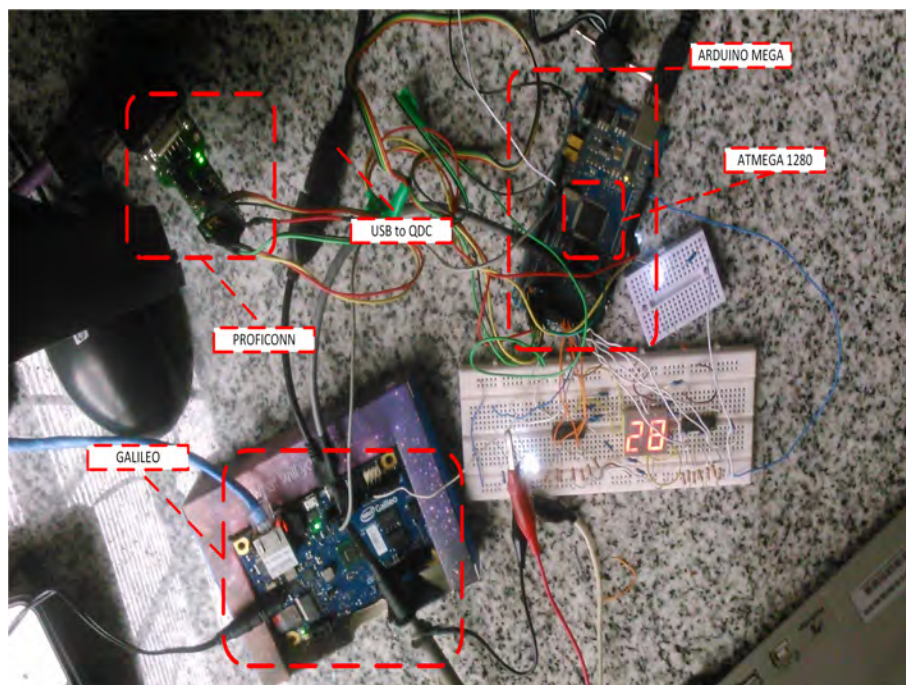
Figura 4.14: Esquema de Interacción.

Para la comunicación de los datos hacia el módulo Profibus, se utilizan interrupciones activadas desde el SBC, la cual una vez activada envía los 12 bits de la conversión, los que se guardan en las variables que posteriormente serán guardados en los registros del ASIC el cual se encargará posteriormente de actualizarlos en la red Profibus.

Capítulo 5

Pruebas y Resultados

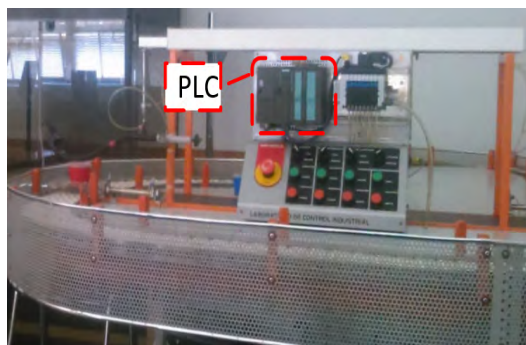
El sistema implementado para realizar las pruebas es el mostrado en la figura(??). En la figura(??a) se puede apreciar la placa de desarrollo Galileo, el ASIC con el Arduino Mega y la protoboard con 2 display de 8 segmentos. Según se aprecia en (??a) se configura una red Profibus según el esquema mostrado en la figura(??). En la figura (??b) y (??c) se aprecia la implementación física de este esquema.



(a) Circuito



(b) Conexión a Red Profibus



(c) Estación de Trabajo

Figura 5.1: Estación de Pruebas

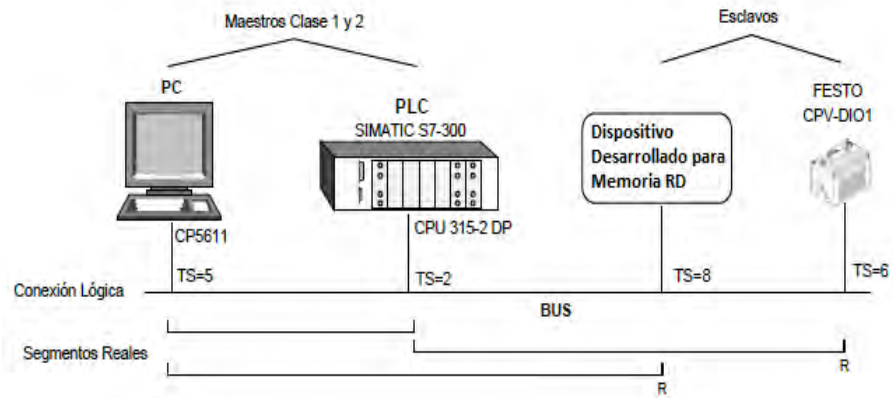


Figura 5.2: Esquema de Interacción en la red.

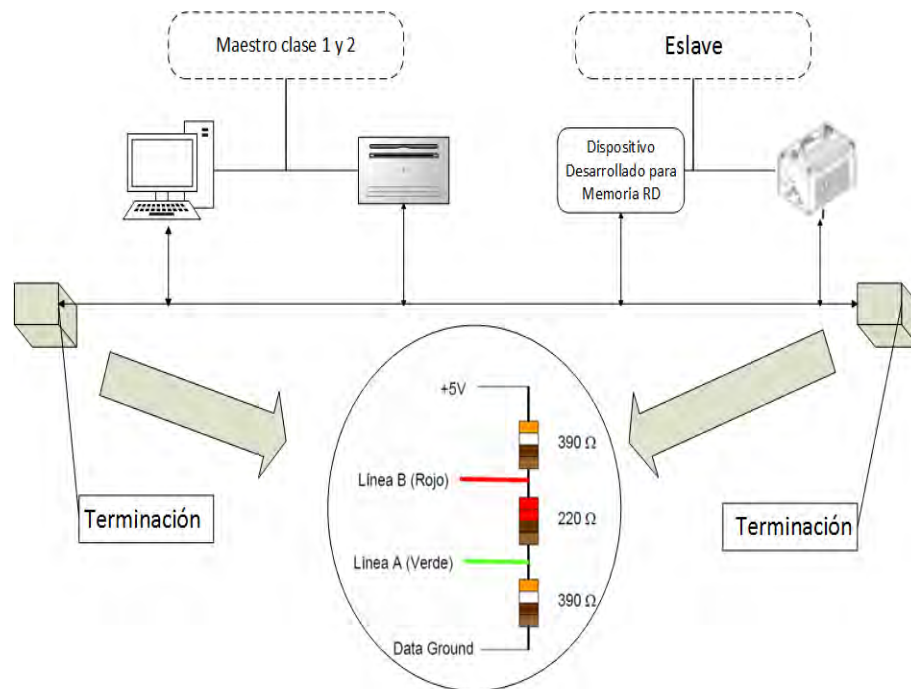


Figura 5.3: Esquema Físico Completo.

La figura (??) muestra las estaciones Maestras y Esclavas conectadas a la red, así como también los terminadores necesarios para la red. Estos terminadores son necesarios para evitar las reflexiones de las señales que viajan por el bus de comunicación y así asegurar comunicación entre dispositivos Profibus.

A continuación se revisan las pruebas y resultados obtenidos para el dispositivo realizado.

5.1 HMI ProTool

Para verificar la correcta integración del sistema desarrollado, y verificar que se envían y reciben datos desde el PLC, se crea una interfaz HMI en una estación OP con opciones de visualización de los datos enviados en un gráfico de tendencias, junto con unos recuadros en los cuales se pueden establecer y leer datos en el dispositivo Esclavo creado. La figura(??) muestra la interfaz realizada.

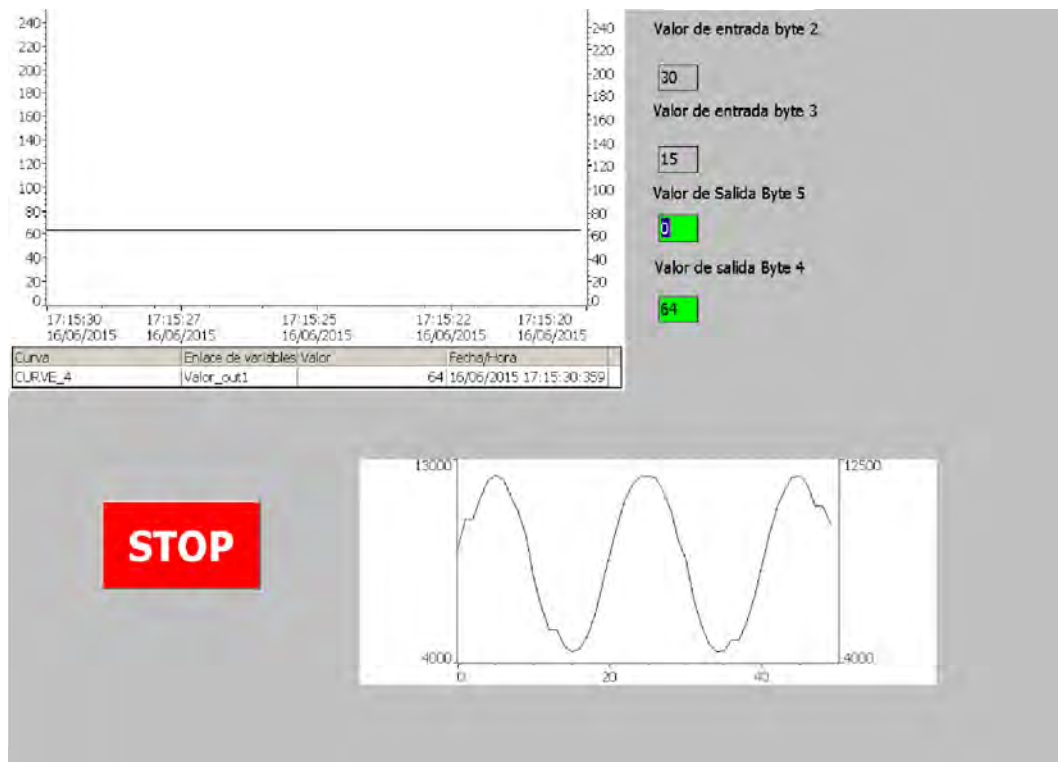


Figura 5.4: Interfaz Hombre Máquina.

En la figura (??) se pueden apreciar dos gráficos de tendencias y cuatro recuadros de lectura y escritura. En la parte superior izquierda se visualiza la señal generada por el PLC, así como también los datos leídos desde el dispositivo Esclavo. En los recuadros de la parte superior derecha, se visualiza BYTE por BYTE el valor tomado tanto en la entrada como en la salida. En el cuadro que se ubica en la parte inferior se visualiza la señal obtenida por el dispositivo QDC y enviado a la red de instrumentación por medio del dispositivo desarrollado.

5.2 Lectura Cíclica de datos

Para la lectura cíclica de datos se utilizan los 2 BYTES de lectura configurados; un BYTE guarda valores de una señal triangular generada desde el ASIC esclavo. Esta señal produce el incremento y el decremento en el brillo de un led, el cual está conectado a unas de las salidas PWM del Arduino que utiliza el dispositivo Esclavo. Para establecer el valor del ciclo de trabajo de la PWM, con una frecuencia portadora de 10[KHz], se utiliza el registro OC0A, el cual corresponde a un BYTE en el Arduino. Para generar la señal triangular se cambia este valor desde 0 a 255, variando así el ciclo de trabajo. El otro BYTE es leído desde el SBC mediante un programa, el cual pide por consola un número de 1 a 255, según se ve en la figura (??), lo que corresponde a un BYTE, para ser enviado al ASIC Esclavo y actualizado en la red. La figura (??) muestra las señales graficadas.

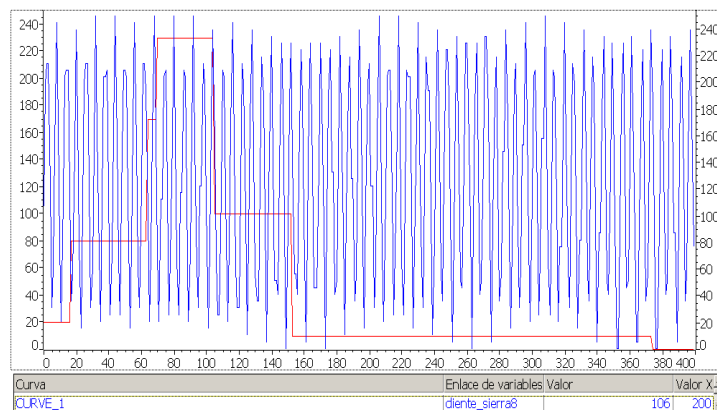


Figura 5.5: Señales de prueba en funcionamiento.

```

root@galileo:/home/spi_master# ./spi_master
echo -n "strong" >/sys/class/gpio/gpio27/driver
echo -n "strong" >/sys/class/gpio/gpio28/driver
spi iniciado
Entrando a WHILE
ingrese un numero de 1 al 255
5
enviando 5
ingrese otro valor: 100
enviando 100
ingrese otro valor: 230
enviando 230
ingrese otro valor: 80
enviando 80
ingrese otro valor: 5
enviando 5
ingrese otro valor:

```

Figura 5.6: Software en SBC.

En la figura (??) se puede distinguir ambas señales. En rojo se ve la señal que es enviada desde el SBC, la cual es arbitraria. En este caso se manda el valor 5, 100, 230, 80 y nuevamente 5. En tanto en azul se puede apreciar la señal triangular que se envía desde el Arduino.

5.3 Escritura Cíclica de datos

Para generar la escritura cíclica de datos se genera una señal triangular en el PLC que va desde 0 a 255, con pasos de 1 en 1, la cual es establecida en un BYTE que es asignado al registro OC0B en el Arduino, el cual establece el ciclo de trabajo para una salida PWM, con frecuencia portadora de 1[KHz] al igual que se explica en la sección ???. La figura (??) muestra la salida graficada en el HMI y la figura (??) muestra los ciclos de trabajo visualizados medidos directamente en el Arduino con un osciloscopio.

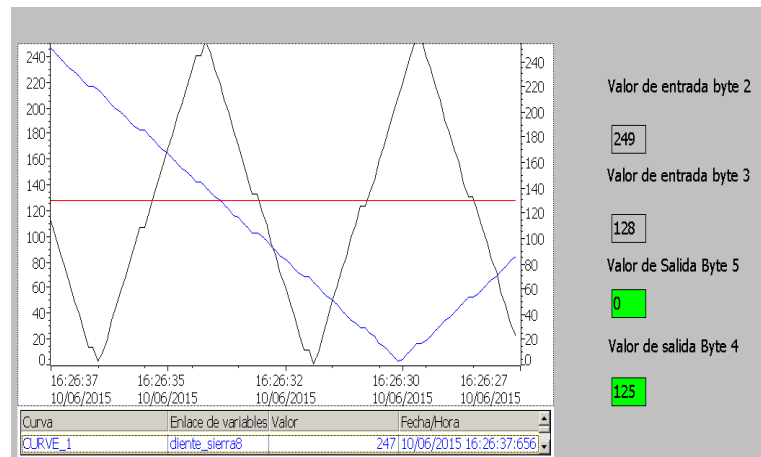
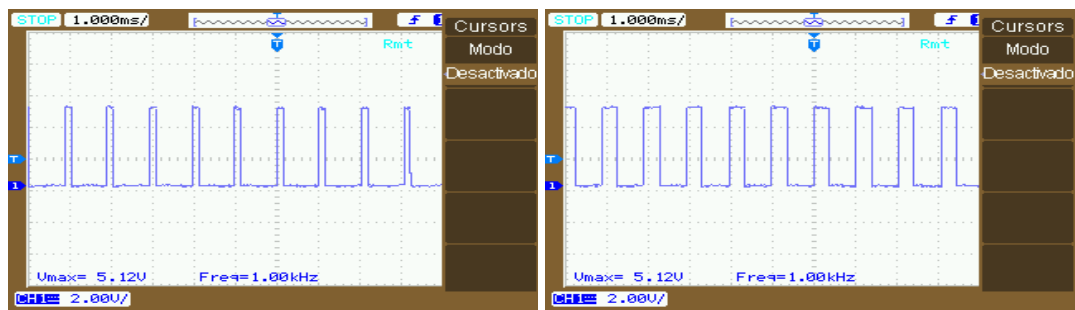
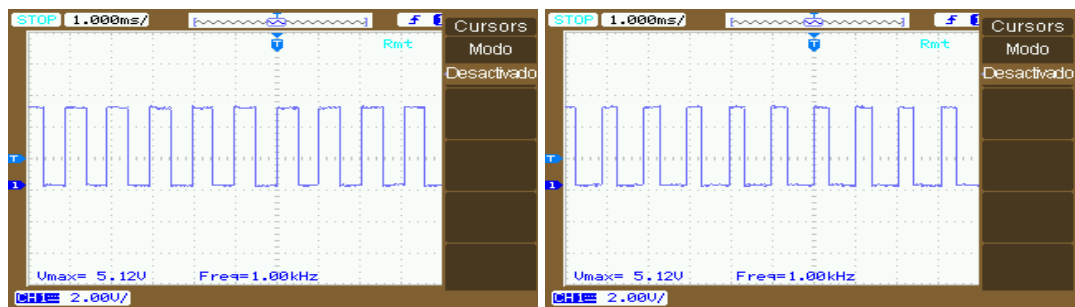


Figura 5.7: Señales en HMI generadas por el PLC.



(a) PWM 20 %

(b) PWM 40 %



(c) PWM 50 %

(d) PWM 40 %

Figura 5.8: Señales PWM generadas por el microcontrolador.

En la figura (??) se pueden visualizar en rojo y azul las señales graficadas en (??) y en negro la señal triangular generada en el PLC. En la figura (??) se nota que el ciclo de trabajo de la PWM, establecida en el Arduino, no presenta grandes variaciones entre un periodo y otro. Para visualizar cambios significativos hay que esperar varios periodos de la PWM, lo que hace pensar que existe una pérdida de datos. Para poder establecer cuál es la frecuencia de actualización del dato enviado desde el PLC al Arduino, se cuenta la cantidad de veces que se repite el mismo ciclo de trabajo, la cual es de 11; con esto se calcula la frecuencia de la portadora, que resulta ser de 90,9[Hz]. Para esto la PWM en el Arduino debe ser generada con un reloj de 2.295[KHz], ya que debe entregar 255 pulsos en 11[ms].

$$FrecuenciaPortadoraNueva = \frac{1}{Repeticiones * Periodo} Hz \quad (5.1)$$

$$FrecuenciaPortadoraNueva = \frac{1}{11 * 0,0001} = 90,9 Hz$$

Con esta frecuencia establecida se mandan ciclos de trabajo de 25 %, 50 % y de 75 % desde el PLC. La nueva frecuencia generadora de la PWM se realiza con un reloj externo, pulsos externos entregados por otro Arduino a 2.295 KHz. El resultado de esta prueba se muestra en la figura (??).

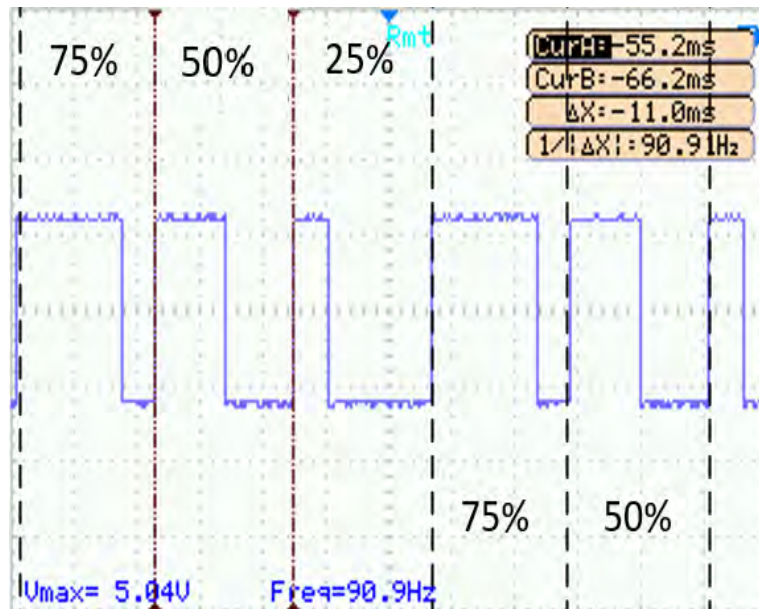


Figura 5.9: Salida PWM con Portadora 90[Hz].

En la figura (??) se pueden identificar claramente los tres ciclos de trabajos generados, pero esto se logra solo durante un tiempo y luego se desfase. Lo anterior ocurre debido a que la comunicación del bus, si bien es cíclica, no es a intervalos regulares de tiempo, lo que produce el desfase cada ciertos intervalos de tiempo.

5.4 Conexión con el QDC

La conexión del QDC se realiza vía USB, como se explica en capítulos anteriores, los drivers y librerías ya están instalados, y para probar la conexión del QDC con el SBC se genera un programa el cual consulta al dispositivo sobre su modelo, firmware, entre otros. Además adquiere datos y los guarda en un archivo. Las figuras (??), (??) muestran esto.

```

root@galileo:/home/caen_test# lsusb
Bus 002 Device 002: ID 21e1:0000
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@galileo:/home/caen_test#

```

(a) lsusb

```

root@galileo:/home/caen_test# ./test1
firmware 04.01 - Build D411
firmware 00.12 - Build D502
model DT5724
root@galileo:/home# cd caen_test/

```

(b) Conexión establecida

Figura 5.10: Conexión QDC-SBC.

#line:event	ch0	ch1	ch2	ch3
0 1	4521	8777	8669	8744
1 2	5728	8778	8669	8742
2 3	5941	8777	8669	8744
3 4	6144	8777	8670	8742
4 5	6394	8778	8669	8745
5 6	6633	8779	8669	8744
6 7	6876	8778	8667	8744
7 8	7127	8776	8670	8745
8 9	7377	8776	8669	8743
9 10	7641	8777	8669	8744
10 11	7910	8777	8669	8743
11 12	8174	8777	8669	8744
12 13	8464	8777	8669	8744
13 14	8747	8779	8666	8743
14 15	9029	8777	8668	8745
15 16	9268	8777	8670	8743
16 17	9534	8778	8668	8745
17 18	9800	8777	8669	8744
18 19	10050	8776	8667	8745
19 20	10324	8778	8669	8742
20 21	10551	8777	8669	8742
21 22	10785	8778	8669	8743
22 23	10991	8777	8669	8743
23 24	11180	8778	8669	8742
24 25	11364	8778	8668	8741
25 26	11559	8774	8668	8743

Figura 5.11: Archivo Generado.

Como se observa en las figuras anteriores, la comunicación entre el dispositivo QDC y el SBC es exitosa.

5.5 Monitoreo del QDC vía Profibus DP

Para realizar el monitoreo del QDC en la red de instrumentación Profibus DP, se genera un programa alojado en el SBC, el cual obtiene la digitalización en un instante para luego enviarla vía SPI hacia el módulo Profibus, el cual se encarga de actualizar los datos en la red. La señal que se digitaliza es un seno de amplitud 1[V] y de frecuencia 1[Hz]. El valor digitalizado es una palabra de 16 bits por lo que se descompone en dos BYTES y se envía. Las figuras (??), (??) muestran lo visto en el HMI, así como también las veces que se actualiza el dato.

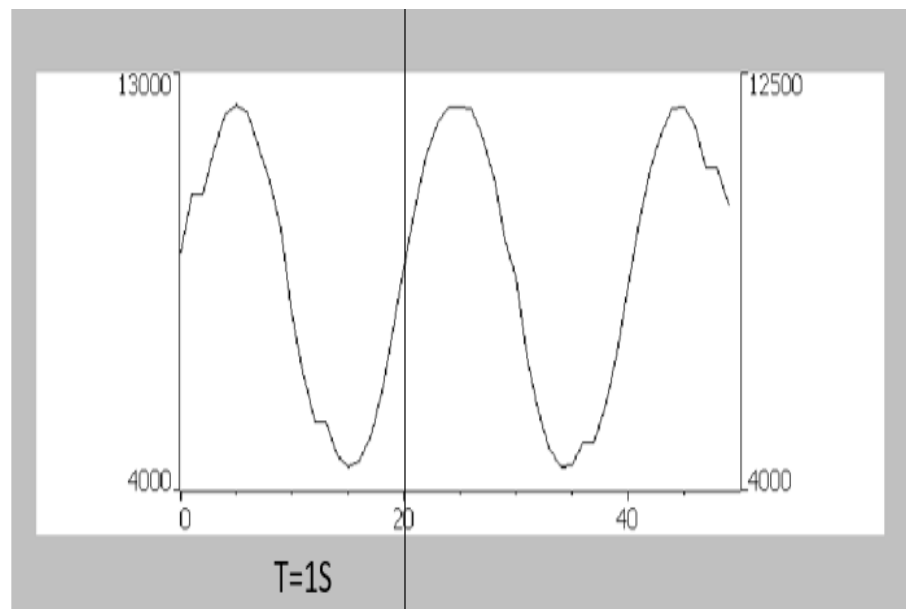


Figura 5.12: Señal QDC en HMI.

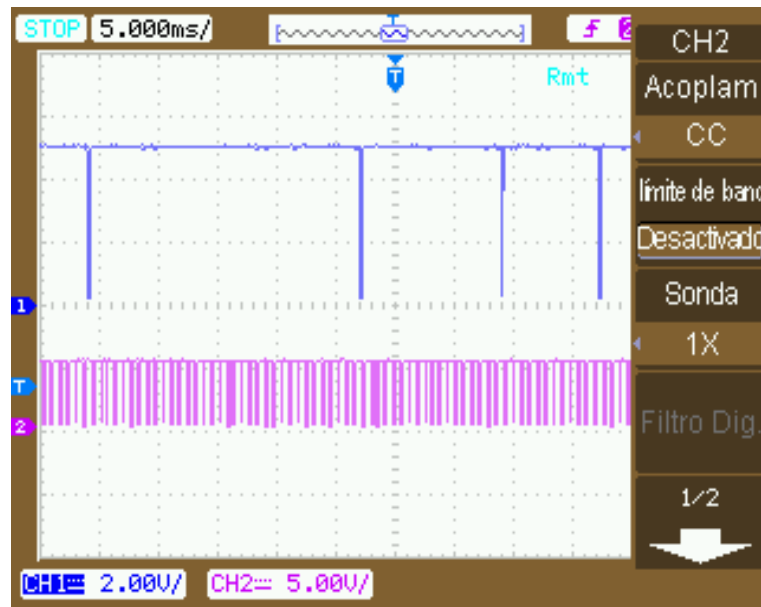


Figura 5.13: Comunicación SBC-Módulo Profibus.

En la figura (??) se visualiza la forma sinusoidal de la señal, con ciertas discontinuidades las cuales pueden ser asociadas a la actualización de los datos en la red, donde se nota la pérdida de datos, lo que concuerda con lo visto en la sección ???. En la figura (??) se ven 2 señales; la azul corresponde a los instantes en los que el módulo Profibus actualiza los datos en la red, lo que corresponde a los instantes en que esta señal es cero. Esta señal es generada por el Arduino. En rosado se puede ver cada cuanto es que el programa actualiza el valor del dato, por lo que queda en evidencia que hay pérdida de datos entre actualizaciones. Esta señal es generada por la línea CS de la comunicación SPI desde el SBC al Arduino. Además esto mismo produce que el módulo de comunicación se cuelgue; lo cual requiere la re-inicialización del protocolo. Para solucionar el problema de actualización se hacen modificaciones en el programa alojado en el SBC, el cual actualiza los datos solo en los instantes en los que es requerido por el Arduino.

Este requerimiento se realiza por medio un pin de salida, el cual se deja en bajo cuando se requiera un dato nuevo. Para sincronizar la frecuencia de actualización por el Arduino en la red Profibus, con la frecuencia de actualización desde el SBC y el Arduino, se debe pedir solo una vez por periodo, lo sincronizaría las frecuencias de actualización.

Capítulo 6

Conclusiones

Según lo visto en capítulos anteriores se establece que:

- La comunicación y adquisición de datos del SBC con el QDC es exitosa, ya que una vez instalados los drivers y librerías en el computador SBC, se habilita la comunicación vía USB con el QDC. Esto se ve reflejado en las pruebas de comunicación y adquisición que fueron un éxito, ya que se logra adquirir, almacenar y enviar estos datos.
- La comunicación con el SBC (el pc de desarrollo) mediante TCP/IP es satisfactoria y bastante útil, ya que esto permite la configuración de la adquisición y procesamiento de los datos, pudiendo de esta forma entregar resultados personalizados.
- El módulo de Comunicación Profibus logra enviar y recibir datos por la red de instrumentación, lo que es satisfactorio, aunque se encuentra la limitación de la velocidad de actualización de datos, que según lo experimentado es de 90[Hz]; a pesar de ser variable es la mayor frecuencia posible. Esto se

debe a que el protocolo consta de varias etapas en las cuales no se envían datos de actualización sino propios para el establecimiento de la comunicación para cada esclavo.

- Para enviar los datos desde el QDC a la red de instrumentación se usa el SBC conectado vía SPI con el módulo de Comunicación Profibus. Este conjunto funciona bien dentro de las limitaciones detectadas, puesto que si se quieren enviar datos a una velocidad mayor a 90[Hz], el conjunto presenta problemas de pérdida de datos y de inestabilidad en la comunicación con la red de instrumentación. Esto se debe a las limitaciones propias de la red de instrumentación, así como también con limitaciones que tiene el SBC. Por lo que se determina que este conjunto es apto para procesos en los cuales la actualización de variables sea hasta 90[Hz], siendo la velocidad de la red Profibus de 787,5 Kbps.
- La interfaz realizada para la configuración local del esclavo y la dirección del Esclavo en la red funcionan correctamente, con el inconveniente que al cambiar este valor se deberá volver a cargar un archivo GSD con la nueva dirección, o cambiarlo con el software de programación del dispositivo Maestro.
- El dispositivo desarrollado permite conectar cualquier dispositivo a una red de instrumentación, siendo una buena alternativa para el desarrollo de equipos aptos para estas redes de instrumentación.

6.1 Trabajos Futuros

En el futuro se puede visualizar un gran número de mejoras y desarrollos para este equipo. A continuación se listan algunas:

- La velocidad de actualización se puede mejorar conectando el dispositivo a una red Profibus que esté configurada a una mayor velocidad que en la que se hicieron pruebas y verificar que la velocidad de actualización es mayor, de no ser así ver cómo mejorar este aspecto.
- Implementar ProfiNet, que es una red de instrumentación igual a la red Profibus, pero utilizando la conexión Ethernet.
- Agregar funcionalidades de las otras versiones de Profibus (V1 y V2).
- Agregar una tarjeta WiFi al SBC para la comunicación inalámbrica del dispositivo, lo que conllevaría grandes mejoras y características nuevas a este dispositivo.
- Solo con el módulo de comunicación se pueden desarrollar equipos para fines diferentes al desarrollado en este trabajo, como conexión de diversos sensores (temperatura, presión, velocidad, sensores on/off, PH, etc) y actuadores (drivers, válvulas on/off, indicadores, etc) a la red Profibus.
- Terminar el desarrollo del equipo creando una PCB, para su posterior encapsulamiento.

Anexo A

Creación de imagen Debian Para Galileo

echo"Author : MatthewDavidElgert"

echo"Authoreddate : 1/24/2014"

echo"Modifieddate : 1/24/2014"

echo"Step1.BuildaDebian – 7,3,0 – i386 – netinst.isovirtualmachine."

echo"Note : Virtualboxmachinewith256MBmemory,1CPU,3,8GBHDD(GalileoSDcardfat324GBlimit)"

echo"Note : Addsecound20GBHDDtodumpandbulddiskimage."

echo'Note : Userscreated"root"and"scott"password"tiger"'

echo"Note : SoftwareinstalledSSHserver."

echo"Step2.TestSSHandupdate."

echo"Note : Loginonconsoleasroot."

ifconfig

echo"Note : SSHintoVMusingputtyhttp : //www.putty.org/."

echo"Note : ThisVMusesDHCPtogetIPaddressSSHdefaultport22."

```

apt - getupdate
apt - getupgrade
echo "Step3.Setup ntp to fetch time from internet.(Galileo has a on board clock that reset when power is pulled)
echo "Note : Url reference - https://wiki.debian.org/DateTime"
echo "Note : Get current date time."
date
apt - getinstall ntpdate
ntpdate pool.ntp.org
echo "Step4.Add second 20GB drive, create and mount partition."
echo "Note : List current disks."
fdisk -l
fdisk /dev/sdb
n
p
echo "Note : (pressed enter 3 times)"
w
mkfs.ext3 /dev/sdb
y
cd /
mkdir sdb
mount -t ext3 /dev/sdb/sdb
e2label /dev/sdb/sdb
df -h
cd sdb
ls
echo "Note : Directory lost + found when mounted."

```

```

echo"Step5.DownloadandmountIntelGalileolinuxoriginalimage."
cd/sdb
wgethttp://downloadmirror.intel.com/23171/eng/LINUX_IMAGE_FORS_DIntelGalileo_v0,7,5,7z
echo"Note : Debian default does not have 7zip installed."
apt-getinstallp7zip
p7zip-h
p7zip-dLINUX_IMAGE_FORS_DIntelGalileo_v0,7,5,7z
cd/sdb/LINUX_IMAGE_FORS_DIntelGalileo_v0,7,5
echo"Note : Verify files extracted."
ls
cd/sdb
mv/sdb/LINUX_IMAGE_FORS_DIntelGalileo_v0,7,5/image-full-clanton.ext3/sdb/image-
full
-clanton_original.ext3
mkdir/sdb/image-full-clanton_original
mount-text3/sdb/image-full-clanton_original.ext3/sdb/image-full-clanton_original
cd/sdb/image-full-clanton_original
echo"Note : Verify mount point directory has files"
ls
echo"Step6.ModifyDebian directory structure to match Intel Galileo image."
cd/media
mkdir/media/card
mkdir/media/cf
mkdir/media/hdd
mkdir/media/mmc1
mkdir/media/net

```

```

mkdir/media/ram
mkdir/media/realroot
mkdir/media/union
mkdir/sketch
cp-avr/sdb/image-full-clanton_original/lib/modules/3,8,7-yocto-standard//lib/modules/3,8,7-yocto-standard
cd/lib/modules/3,8,7-yocto-standard
echo"Note : Verifyfiles copied."
ls
cp-avr/sdb/image-full-clanton_original/opt/cln//opt/cln
cd/opt/cln/galileo
echo"Note : Verifyfiles copied."
ls
echo"Note : ***Mayneedtorevisitandverifythisstep***Hadaissuewithfirstbootnotrunningapt-getinstallArduino***dialoutpermissionsdidnottransfer"
echo"Step7.EnableDebianoutputtoaconsole."
nano/etc/inittab
echo"Note : AddthefollowinglinetoEOF"
echo"s0:2345:respawn:/sbin/getty-L115200ttyS1vt102"
echo"Note : Seeprojectfileinittabforfullexample."
echo"Step8.DumpLinuxOSimagetoafile."
cd/sdb
echo"Note : Dumpfullsystemthisstepmightbebetterduringacoldorrecoverybootsofaritworkswithoutany
fdisk-l
echo"Note : Noticetheblocksizeof/dev/sda1inoutputmyvirtualmachineis3766272."
ddif=/dev/sda1of=/sdb/image-full-clanton.ext3bs=3766272conv=sync,noerror

```

```

mkdir/sdb/image - full - clanton
mount - text3/sdb/image - full - clanton.ext3/sdb/image - full - clanton
echo"Note : If disk won't mount had to dump again or re boot. This step may be better while system is cold or in re
cd/sdb/image - full - clanton
echo"Note : Verify file structure."
ls echo"Step 9. Setup disk image."
rm - rf/sdb/image - full - clanton/sys
rm - rf/sdb/image - full - clanton/proc
mkdir/sdb/image - full - clanton/sys
mkdir/sdb/image - full - clanton/proc
cd/sdb
umount - l/sdb/image - full - clanton
mv/sdb/image - full - clanton.ext3/sdb/LINUX_IMAGE_FOR_SDIntelGalileo_v0,7,5/image -
full - clanton.ext3
cd/sdb/LINUX_IMAGE_FOR_SDIntelGalileo_v0,7,5
echo"Note : Verify file copy image - full - clanton.ext3"
ls
echo"Step 10. Setup Galileo MicroSD card - https://communities.intel.com/servlet/JiveServlet/previe
102-1-25429/Galileo_Getting_Started_329685_05.pdf
echo"From a Windows machine after SD card is installed run command prompt as Administrator." echo"Run
echo"DISKPART > list disk"
echo"DISKPART > select vol < a >; (where < a > = the drive letter of the SD card) (example select Disk 3)"
echo"DISKPART > clean"
echo"DISKPART > create part primary"
echo"DISKPART > active"
echo'DISKPART > format quick label = "BOOTME"'

```

```
echo"Note : MaywantafullformattoverifyagoodSDcardthistakesalongtime."
echo'DISKPART > formatlabel = "BOOTME"'
echo'DISKPART > exit'
echo"Step11.Bootnewimage – full – clanton.ext3"
echo"Note : Ifyourebootwillneedtoremount/sdb."
mount – text3/dev/sdb/sdb
echo"Note : UsesftpclientsuchasfreeWinSCPtocopyallfilesfrom/sdb
/LINUX_IMAGE_FOR_SDIIntelGalileo_v0,7,5torootdirectoryofnewSDcard"
```

Anexo B

Cable Serial para consola SBC

Primero se necesitan los siguientes componentes:

- Plug estéreo de 3.5 mm (1/8').
- Conector hembra DE9.
- Cable de diferentes colores.
- Aislador engomado.
- Cautín, estaño, cables desnudos o cortos.

Para hacer el cable debemos conectar los cables del plug estéreo a los siguientes terminales:

- DB9 pin 2 al cable tip.
- DB9 pin 3 al cable middle.
- DB9 pin 3 al cable base.

En las siguientes figuras se ven las figuras del DB9 y el plug estéreo, señalando el nombre de los pins. Además se muestra una fotografía del cable ya terminado.



Figura B.1: Pins DB9 y plug estéreo.



Figura B.2: Pins DB9 y plug estéreo.

Bibliografía

- [1] Rene Ríos, Memoria de Titulación Ingeniería Civil Electrónica, "Desarrollo de un método de densimetría nuclear con aplicaciones en minería", Universidad Técnica Federico Santa María, 2015.
- [2] Lautaro Narvaez, Memoria de Titulación Ingeniería Civil Electrónica, "Diseño y construcción de una gamma-cámara para SPECT", Universidad Técnica Federico Santa María, 2015.
- [3] Diego Valencia, Memoria de Titulación Ingeniería Civil Electrónica, "Desarrollo de un dispositivo esclavo Profibus-DP en base al estándar IEC6115", Universidad Técnica Federico Santa María, 2010.
- [4] <http://www2.elo.utfsm.cl/~elo307/> en S1-2015. Última vez revisado 18 de septiembre 2016.
- [5] <http://www.profiship.com/products/overviewasics/dp-slave-vpc3-s/?L=5>
- [6] <http://winavr.sourceforge.net/>
- [7] <http://www.caen.it>

- [8] Carlos Valdivieso, Revista Tecnológica ESPOL, "Utilización de la Minicomputadora Raspberry Pi para la adquisición y evaluación de datos de consumo de energía eléctrica de equipos de 220 Voltios", Ecuador, 2013.
- [9] Juan Nuñez, Tesis Profesional, "Diseño e integración de un sistema de adquisición de datos mediante el uso de Arduino y Raspberry Pi", Universidad Nacional Autónoma de México, México, 2014.
- [10] <https://www.debian.org/index.es.html>
- [11] <http://www.intel.la/content/www/xl/es/do-it-yourself/galileo-maker-quark-board.html>
- [12] <https://communities.intel.com/thread/48074?tstart=0>
- [13] http://downloadmirror.intel.com/23171/eng/LINUX_IMAGE_FOR_SD_Intel_Galileo_v0.7.5.7z
- [14] <http://www.openssh.com/>
- [15] <http://www.profiship.com/products/overviewasics/dp-slave-vpc3-c/dp-v0-firmware/?L=5>
- [16] VPC3+ Software Description V600.pdf, Profichip.
- [17] http://linksprite.com/wiki/index.php5?title=Touch_LCD_Shield
- [18] Device Description Data Files GSD. SIEMENS, Versión 2.0, Junio 2002.
- [19] http://en.pudn.com/downloads361/sourcecode/editor/detail11569779_en.html