



PROYECTO FIN DE CARRERA

“Control de PLCs Siemens S7-1200 mediante el protocolo MODBUS a través del programa LABVIEW para realización de prácticas de comunicación industrial”

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber MarínIturrarte
Tutor: Ignacio Del Villar Fernández



MEMORIA

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber Marín Iturrarte
Tutor: Ignacio Del Villar Fernández

	<u>PÁGINAS</u>
<u>1. Memoria</u>	3
<u>1.1 Índice paginado</u>	3
<u>1.2 Objeto del Proyecto</u>	4
<u>1.3 Antecedentes</u>	4
<u>1.4 Datos de partida</u>	4
<u>1.5 Características más importantes del proyecto</u>	5
<u>1.6 Posibles Soluciones</u>	5
<u>1.7 Solución Adoptada</u>	18
<u>1.8 Descripción de lo proyectado</u>	18
<u>2. Cálculos</u>	80
<u>2.1 Cálculos Divisor de Tensión</u>	81
<u>2.2 Cálculos Resistencias del Transistor</u>	81
<u>2.3 Cálculos Velocidad Motor</u>	82
<u>3. Presupuesto</u>	85
<u>3.1 Materiales de Laboratorio</u>	85
<u>3.2 Software</u>	86
<u>3.3 Mano de Obra</u>	86
<u>3.4 Presupuesto Total</u>	89
<u>4. Anexo Memoria</u>	88
<u>4.1 Direccionamiento de variables en el PLC</u>	1
<u>4.2 Hoja Característica del Transistor</u>	4
<u>4.3 Labview</u>	6
<u>4.4 Guión de Prácticas</u>	9
<u>5. Bibliografía</u>	

1. Memoria

<u>1.1 Índice Paginado</u>	<u>PÁGINAS</u>
<u>1.2 Objeto del Proyecto</u>	4
<u>1.3 Antecedentes</u>	4
<u>1.4 Datos de partida</u>	4
<u>1.5 Características más importantes del proyecto</u>	5
<u>1.6 Posibles Soluciones</u>	5
<u>1.7 Solución Adoptada</u>	18
<u>1.8 Descripción de lo proyectado</u>	18
<u>1.8.1 Comunicación</u>	19
<u>1.8.2 Tipo de Comunicación Modbus</u>	20
<u>1.8.2.1 Modbus RTU y Modbus ASCII</u>	21
<u>1.8.2.2 Modbus TCP</u>	26
<u>1.8.2.3 Modbus+</u>	28
<u>1.8.2.4 Resumen Modbus RTU,ASCII,TCP,+</u>	29
<u>1.8.3 Conexión PC con PLC, Modbus aplicado a Labview</u>	31
<u>1.8.3.1 Creación Maestro (Servidor)</u>	31
<u>1.8.3.2 Creación Esclavo (Cliente)</u>	41
<u>1.8.3.3 Acceder a las variables de un PLC</u>	43
<u>1.8.3.4 Acceder a variables de varios PLC's</u>	46
<u>1.8.4 Conexión placa PICDEM Mechatronic Demonstration Board</u>	53
<u>1.8.4.1 Alimentación de la placa</u>	53
<u>1.8.4.2 Control motor mediante un encoder</u>	63
<u>1.8.4.3 Circuito Amplificador</u>	64
<u>1.8.4.4 Leer Variable Encoder</u>	66
<u>1.8.4.5 Variación de la velocidad del motor</u>	71
<u>1.8.5 Monitorización de Datos</u>	72
<u>1.8.6 Conclusiones</u>	79

1.2 Objeto del proyecto

El objeto del presente proyecto consiste en realizar una comunicación MODBUS entre un PC y el PLC Siemens S-7 1200. Para ello se utilizarán como herramientas el software TIA para programación de PLCs de la familia de PLCs 1200 de Siemens y el software Labview. Una vez cumplido este punto, el siguiente objetivo consiste en controlar un motor que viene incluido en la tarjeta PICDEM™ Mechatronics Demonstration Board.

1.3 Antecedentes

El proyecto se va a desarrollar en el Laboratorio de Electrónica de Potencia. El laboratorio consta de varios ordenadores y diversos PLC'-s . La comunicación entre los ordenadores y los PLC'-s se realiza a través del propio software TIA del PLC Siemens y la red Ethernet del laboratorio. Estos elementos han sido utilizados en asignaturas impartidas en varias titulaciones de la Universidad Pública de Navarra tales como Instrumentación de I.T.I. Eléctrico, o Instrumentación y Sensores del Máster de Mecánica Aplicada y Computacional.

1.4 Datos de partida

La Universidad proporciona un ordenador con las correspondientes licencias de uso para los distintos softwares y varios PLCs para poder llevar a cabo las pruebas necesarias. Existe una red Ethernet. También se proporciona la tarjeta PICDEM™ Mechatronics Demonstration Board.



Figura 1.PLC S7-1200

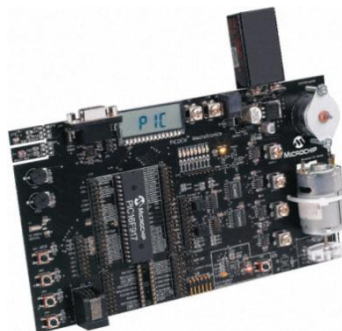


Figura 2.PICDEM Mechatronics Demonstration Board

1.5 Características más importantes del proyecto

- Comunicación MODBUS entre PC y un PLC
- Control de Variables del PLC mediante la interfaz de Labview
- Control de la tarjeta PICDEM Mechatronics Demonstration Borad desde el PLC
- Monitorización mediante Labview de la posición y velocidad de un motor de la tarjeta PICDEM.
- Realizar guión de prácticas para la asignatura de Comunicación Industrial.

1.6 Posibles soluciones

Existen distintas soluciones para la comunicación de un PC con un PLC. Antes de ver las soluciones conviene explicar dos modelos. El primero es el modelo OSI, en el cual se basan la mayoría de los sistemas de comunicación entre dispositivos, y el segundo es el modelo TCP/IP, que es el estándar más empleado en la actualidad, el cual es el fundamento de la red Internet

- Modelo OSI: (Open Systems Interconnection), se desarrolló por la Organización Internacional de Estandarización ISO (International Organization for Standarization) como una arquitectura para comunicaciones entre computadores, con el objetivo de ser el marco de referencia en el desarrollo de protocolos estándares. El modelo OSI consta de siete capas:
 1. Aplicación
 2. Presentación
 3. Sesión
 4. Transporte
 5. Red
 6. Enlace de Datos
 7. Física

-Capa de Aplicación: Proporciona el acceso al entorno OSI para los usuarios, también proporciona servicios de información distribuida.

-Capa de Presentación: Proporciona a los procesos de aplicación independencia respecto a las diferencias en la representación de los datos (syntax).

-Capa de Sesión: Proporciona el control de la comunicación entre las aplicaciones; establece, gestiona y cierra las conexiones (sesiones) entre las aplicaciones cooperadoras.

-Capa de Transporte: Proporciona seguridad, transferencia transparente de datos entre los puntos finales; proporciona además procedimientos de recuperación de errores y control de flujo origen-destino.

-Capa de Red: Proporciona independencia a los niveles superiores respecto a las técnicas de conmutación y de transmisión utilizadas para conectar los sistemas; es responsable del establecimiento, mantenimiento y cierre de las conexiones.

-Capa de Enlace de Datos: Proporciona un servicio de transferencia de datos seguro a través del enlace físico; envía bloques de datos (tramas) llevando a cabo la sincronización, el control de errores y de flujo necesarios.

-Capa de Física: Se encarga de la transmisión de cadenas de bits no estructurados sobre el medio físico; está relacionada con las características mecánicas, eléctricas, funcionales y de procedimiento para acceder al medio físico.



Figura 3. Capas OSI

Una vez explicado el modelo OSI se pasará a describir el modelo TCP/IP, en el cual se aprecian similitudes con respecto al modelo OSI.

- **Modelo TCP/IP:** Este modelo está basado en un modelo de referencia de cinco niveles. Todos los protocolos que pertenecen al conjunto de los protocolos TCP/IP se encuentran en los tres niveles superiores de este modelo. El nivel del modelo TCP/IP corresponde a uno o más niveles del modelo de referencia de conexión de sistemas abiertos (OSI) de siete niveles.
1. **Nivel de Aplicación:** Proporciona la comunicación entre procesos o aplicaciones de computadores separados. En este nivel se definen los protocolos de aplicación TCP/IP y como se conectan los programas de host a los servicios de nivel de transporte para utilizarlos en la red. Protocolos: HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP, X Windows y otros protocolos de aplicación.
 2. **Nivel de Transporte:** Proporciona un servicio de transferencia de datos extremo-a-extremo. Esta capa puede incluir mecanismos de seguridad. Oculta los detalles de la red, o redes subyacentes, a la capa de aplicación. Este nivel permite administrar las sesiones de comunicación entre equipos host. Define el nivel de servicio y estado de la conexión utilizada por el transportador de datos. Protocolos: TCP, UDP, RTP.
 3. **Nivel de Internet (Red):** Esta capa está relacionada con el encaminamiento de los datos del computador origen al destino a través de una o más redes conectadas por dispositivos de encaminamiento. Este nivel se encarga de empaquetar los datos en datagrama IP, que contienen información de las direcciones de origen y destino utilizadas para reenviar los diagramas entre hosts y a través de redes. Realiza el enrutamiento de datagramas IP. Protocolos: IP, ICMP, ARP, RARP.
 4. **Nivel de acceso a la red (Enlace de Datos):** Esta capa está relacionada con la interfaz lógica entre un sistema final y una subred.
 5. **Nivel Físico:** En este nivel se especifica información detallada de cómo se envían físicamente los datos a través de la red, que incluye como se realiza la señalización eléctrica de los bits mediante los dispositivos de hardware que conectan directamente con un medio de red, como un cable coaxial, un cable de fibra óptica o un cable de cobre de par trenzado. Protocolo: Ethernet, Token Ring, FDDI, X.25, Frame Relay, RS-232, V.35

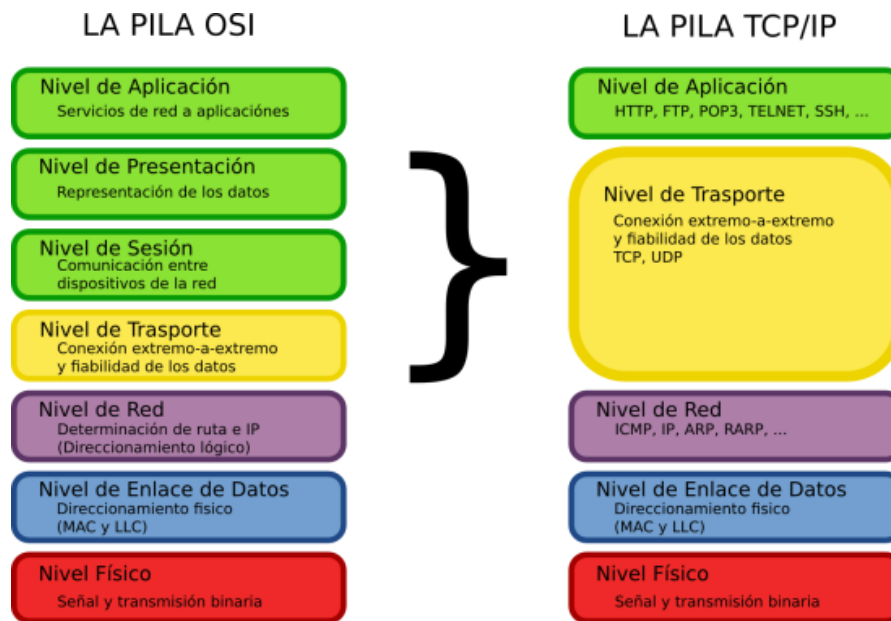


Figura 4. Pila OSI-Pila TCP/IP

Los diseñadores de OSI consideraron que este modelo y los protocolos asociados llegarían a dominar las comunicaciones entre computadores, reemplazando eventualmente las implementaciones particulares de protocolos, así como a modelos rivales tales como TCP/IP. Sin embargo, esto no ha sido así. Aunque se han desarrollado muchos protocolos de utilidad dentro del contexto de OSI, el modelo de las siete capas en su conjunto no ha prosperado. Por el contrario la arquitectura TCP/IP se ha impuesto como dominante. Ya vistos los modelos OSI y TCP/IP, ahora se pasará a describir las posibles soluciones para la comunicación entre PLCs y PCs.

- Comunicación con un cable serie: Para una comunicación con puerto serie el protocolo que se utiliza es el RS-232. El protocolo RS-232 es una norma o estándar mundial que rige los parámetros de uno de los modos de comunicación serial. Por medio de este protocolo se estandarizan las velocidades de transferencia de datos, la forma de control que utiliza dicha transferencia, los niveles de voltajes utilizados, el tipo de cable permitido, las distancias entre equipos, los conectores, etc.

Además de las líneas de transmisión (Tx) y recepción (Rx), las comunicaciones seriales poseen otras líneas de control de flujo (*Hands-hake*), donde su uso es opcional dependiendo del dispositivo a conectar.

A nivel de software, la configuración principal que se debe dar a una conexión a través de puertos seriales RS-232 es básicamente la selección de la velocidad en baudios (1200, 2400, 4800, etc.), la verificación de datos o paridad (paridad par o paridad impar o sin paridad), los bits de parada luego de cada dato (1 ó 2), y la cantidad de bits por dato (7 ó 8), que se utiliza para cada símbolo o carácter enviado.

La Norma RS-232 fue definida para conectar un ordenador a un modem. Además de transmitirse los datos de una forma serie asíncrona son necesarias una serie de señales adicionales, que se definen en la norma. Las tensiones empleadas están comprendidas entre +15/-15 voltios.

Existe una variante de RS-232 denominada bus de transmisión RS-485 (también conocido como EIA-485). Está definido como un sistema en bus de transmisión multipunto diferencial. La diferencia principal que existe entre el bus RS-232 y RS-485 es que con el primer bus la comunicación es entre dos equipos, mientras que en el segundo la comunicación puede ser hasta como 31 dispositivos. La distancia que separa los dispositivos es mayor para el puerto RS-485. Se puede decir que este puerto es ideal para transmitir a altas velocidades sobre largas distancias y reduciendo los ruidos que aparecen en la línea de transmisión.

El medio físico de transmisión es un par entrelazado que admite hasta 32 estaciones en 1 solo hilo, con una longitud máxima de 1.200 metros operando entre 300 y 19.200 bps y la comunicación half-duplex (semiduplex). La transmisión diferencial permite múltiples drivers dando la posibilidad de una configuración multipunto. Al tratarse de un estándar bastante abierto permite muchas y muy diferentes configuraciones y utilizaciones.

- **OPC Server:** **OPC** (*OLE for Process Control*) es un estándar de comunicación en el campo de control y supervisión de procesos industriales, basado en una tecnología Microsoft, que ofrece un protocolo común para comunicación. Permite que componentes software individuales interaccionen y compartan datos. La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. El servidor OPC es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor. Es una solución abierta y flexible al clásico problema de los drivers propietarios. Prácticamente todos los mayores fabricantes de sistemas de control, instrumentación y de procesos han incluido OPC en sus productos.

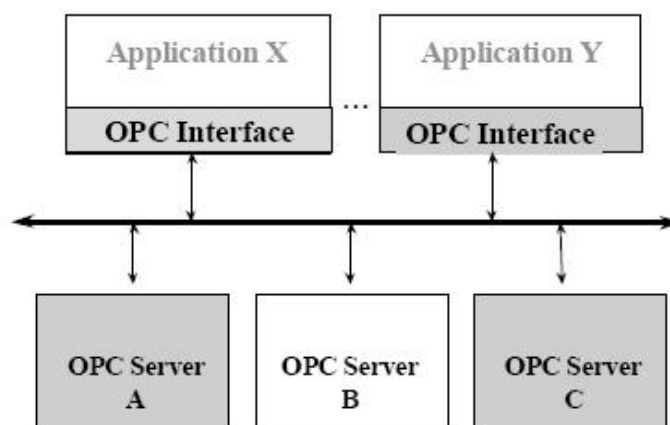


Figura 5. Gráfico OPC

A continuación se pueden ver varios ejemplos para esclarecer lo anteriormente explicado.

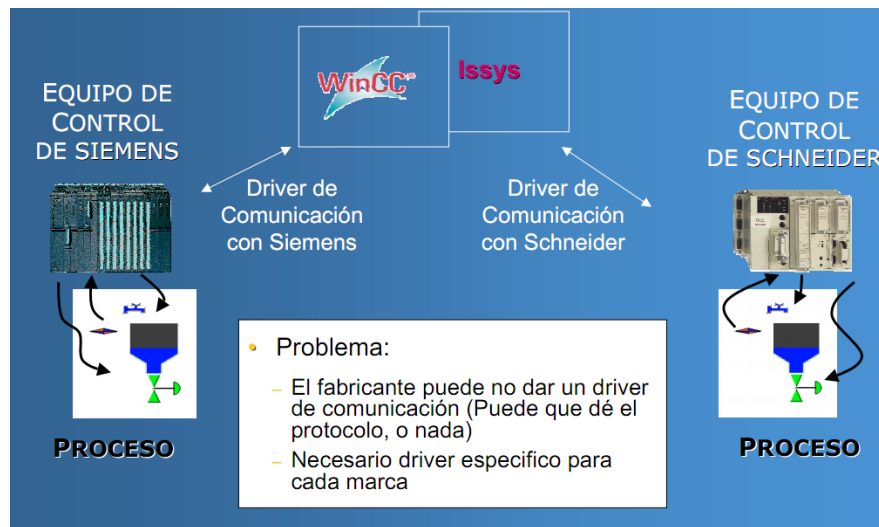


Figura 6. Equipos de distintas marcas sin comunicación

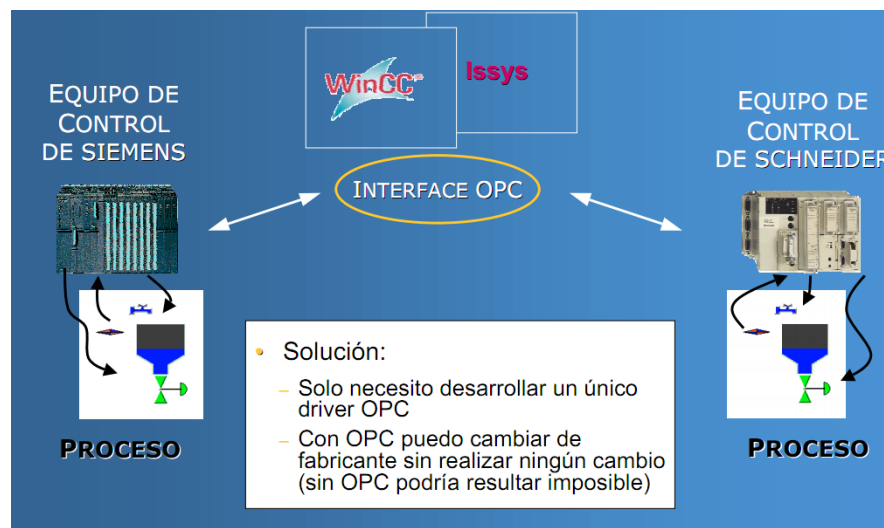


Figura 7. Equipos de distintas marcas se comunican a través de un servidor OPC

Un **servidor OPC** es una aplicación de software (driver) que cumple con una o más especificaciones definidas por la OPC Foundation. El Servidor OPC hace de interfaz comunicando por un lado con una o más fuentes de datos utilizando sus protocolo nativos (típicamente PLCs, DCSs, básculas, Módulos I/O, controladores, etc.) y por el otro lado con Clientes OPC (típicamente SCADAs, HMIs, generadores de informes, generadores de gráficos, aplicaciones de cálculos, etc.). En una arquitectura Cliente OPC/ Servidor OPC, el Servidor OPC es el esclavo mientras que el Cliente OPC es el maestro. Las comunicaciones entre el Cliente OPC y el Servidor OPC son bidireccionales, lo que significa que los Clientes pueden leer y escribir en los dispositivos a través del Servidor OPC.

Existen cuatro tipos de servidores OPC definidos por la OPC Foundation:

1. Servidor OPC DA
2. Servidor OPC HDA
3. Servidor OPC A&E
4. Servidor OPC UA

Los tres primeros servidores son denominados como Servidores OPC Clásicos, para diferenciarlos de OPC UA que se convertirá en la base de las futuras arquitecturas de OPC. Los servidores OPC se componen por una estructura básica que consta de tres partes:

1. Comunicaciones Cliente OPC/Servidor OPC(Servidor OPC DA, Servidor OPC HDA, Servidor OPC A&E): Los Servidores OPC clásicos utilizan la infraestructura COM/DCOM de Microsoft Windows para el intercambio de datos. Debido a esto los Servidores OPC deben instalarse en aquellos PCs que trabajen con el Sistema Operativo de Microsoft Windows. Un Servidor OPC está diseñado para soportar comunicaciones con varios Clientes OPC simultáneamente.
2. Servidor OPC - Traducción de Datos/Mapping: La principal función de un Servidor OPC es traducir datos nativos de la fuente de datos a un formato OPC que sea compatible con una o más especificaciones OPC mencionadas anteriormente (ejemplo: OPC DA para datos en tiempo real). Las especificaciones de la OPC Foundation solo definen la porción OPC de las comunicaciones del Servidor OPC, por lo que la eficiencia y calidad de traducción del protocolo nativo a OPC y de OPC al protocolo nativo, dependen únicamente de la implementación de la persona que desarrolla el Servidor OPC.
3. Servidor OPC –Comunicación Fuente de Datos: Los Servidores OPC se comunican nativamente con las fuentes de datos, como por ejemplo dispositivos, controladores y aplicaciones. Las especificaciones de la OPC Foundation no señalan cómo el Servidor OPC se debe comunicar con la fuente de datos, debido a que existe una gran variedad de fuentes de datos disponibles en el mercado. Cada PLC, DCS, controlador, etc. tiene su propio protocolo de comunicación o API, que a su vez permiten la utilización cualquier cantidad de conexiones físicas (serial RS485 o RS232, Ethernet, wireless, redes propietarias, etc.).

Dos ejemplos comunes de comunicación entre los Servidores OPC con la Fuente de Datos son:

- A través de una interfaz de programación de aplicaciones (API) para un driver personalizado escrito específicamente para la Fuente de Datos.
- A través de un protocolo que puede o no ser propietario, o basado en un estándar abierto (por ejemplo utilizando el protocolo Modbus).



Figura 8. Estructura básica de un OPC Server

- Buses de campo para redes industriales

Se define un bus de campo como un sistema de comunicación para intercambiar datos entre sistemas de automatización y dispositivos de campo en tiempo real basado en el modelo OSI.

Los buses de campo se utilizan principalmente como un sistema de comunicación entre los sistemas de automatización y los dispositivos de campo. El objetivo es reemplazar los sistemas de control centralizados por redes de control distribuido mediante el cual permita mejorar la calidad del producto, reducir los costos y mejorar la eficiencia.

Con los años los buses de campo han ido evolucionando, prueba de ello es que hace más de 50 años la instrumentación de procesos estaba basada en el estándar de señalización neumática 3-15 psi (libras por pulgada cuadrada). 20 años después el estándar pasó a ser del tipo de señales analógicas, donde el control se realizaba a través del bucle de corriente 4-20mA, y en la década de 1980 empezaron las comunicaciones digitales.

- Finales de los 70 Modbus de Modicon
- 1982 → Se inicia Grupo de Trabajo FIP
- 1983 → P-Net (Dinamarca)
- 1984 → CAN
- 1985 → Grupo Profibus
- 1985 → Inicio de trabajos de Normalización Internacional: ISA SP50, IEC, TC65/SC65C
- 1994 → Fieldbus Foundation

Debido a esta evolución en los buses de campo las instalaciones automatizadas también evolucionaron, descentralización de dispositivos inteligentes, aparición e integración de las nuevas tecnologías como los sistemas SCADA, supresión del cableado de entradas/salidas, acceso a los datos y desaparición de las interfaces de entrada/salida. Gracias al uso de los buses de campo se han conseguido mejoras como la reducción de costes, mantenimiento de la red, flexibilidad, simplificación, comunicación bidireccional y servicios de administración.

Debido a la falta de estándares, diferentes compañías han desarrollado diferentes soluciones, cada una de ellas con diferentes prestaciones y campos de aplicación. En una primera clasificación tenemos los siguientes grupos:

▪ **Buses de Alta Velocidad y Baja Funcionalidad**

Están diseñados para integrar dispositivos simples como finales de carrera, fotocélulas, relés y actuadores simples, funcionando en aplicaciones de tiempo real, y agrupados en una pequeña zona de la planta, típicamente una máquina. Básicamente comprenden las capas física y de enlace del modelo OSI, es decir, señales físicas y patrones de bits de las tramas. Algunos ejemplos son:

- CAN: Diseñado originalmente para su aplicación en vehículos
- SDS: Bus para la integración de sensores y actuadores, basado en CAN
- ASI: Bus serie para la integración de sensores y actuadores

▪ **Buses de Alta Velocidad y Funcionalidad Media**

Se basan en el diseño de una capa de enlace para el envío eficiente de bloques de datos de tamaño medio. Estos mensajes permiten que el dispositivo tenga mayor funcionalidad de modo que permite incluir aspectos como la configuración, calibración o programación del dispositivo. Son buses capaces de controlar dispositivos de campo complejos, de forma eficiente y a bajo costo. Normalmente incluyen la especificación completa de la capa de aplicación, lo que significa que se dispone de funciones utilizables desde programas basados en PCs para acceder, cambiar y controlar los diversos dispositivos que constituyen el sistema. Algunos incluyen funciones estándar para distintos tipos de dispositivos (perfiles) que facilitan la interoperabilidad de dispositivos de distintos fabricantes. Algunos ejemplos son:

- DeviceNet: Desarrollado por Allen-Bradley, utiliza como base el bus CAN, e incorpora una capa de aplicación orientada a objetos.
- LONWorks: Red desarrollada por Echelon
- BitBus: Red desarrollada por INTEL
- DIN MessBus: Estándar alemán de bus de instrumentación, basado en comunicación RS-232
- InterBus-S: Bus de campo alemán de uso común en aplicaciones medias

▪ Buses de Altas Prestaciones

Son capaces de soportar comunicaciones en todos los niveles de la producción CIM (Manufactura Integrada por Computadora). Aunque se basan en buses de alta velocidad, algunos presentan problemas debido a la sobrecarga necesaria para alcanzar las características funcionales y de seguridad que se les exigen. La capa de aplicación tiene un gran número de servicios a la capa de usuario, habitualmente un subconjunto del estándar MMS (Manufacturing Message Specification). Entre sus características:

- Redes multi-maestro con redundancia
- Comunicación maestro-esclavo según el esquema pregunta-respuesta
- Recuperación de datos desde el esclavo con un límite máximo de tiempo
- Capacidad de direccionamiento unicast, multicast y broadcast
- Petición de servicios a los esclavos basada en eventos
- Comunicación de variables y bloques de datos orientada a objetos
- Descarga y ejecución remota de programas
- Altos niveles de seguridad de la red, procedimientos de autenticación
- Conjunto completo de funciones de administración de la red

EJEMPLOS:

- Profibus
- FIP
- Fieldbus Foundation

▪ Buses para áreas de seguridad intrínseca

Incluyen modificaciones en la capa física para cumplir con los requisitos específicos de seguridad intrínseca en ambientes con atmósferas explosivas. La seguridad intrínseca es un tipo de protección por la que el componente en cuestión no tiene posibilidad de provocar una explosión en la atmósfera circundante. Un circuito eléctrico o una parte de un circuito tienen seguridad intrínseca, cuando alguna chispa o efecto térmico en este circuito producidos en las condiciones de prueba establecidas por un estándar (dentro del cual figuran las condiciones de operación normal y de fallo específicas) no puede ocasionar una ignición. Algunos ejemplos son HART, Profibus PA o WorldFIP.

A continuación se describirán los buses de campo más populares dentro del mercado internacional, algunos de los cuales ya han sido citados en las anteriores clasificaciones que se han presentado.

1. PROFIBUS

- Profibus DP (Decentralized Periphery). Orientado a sensores/actuadores enlazados a procesadores (PLCS) o terminales.
- Profibus PA (Process Automation). Para control de proceso y cumpliendo normas especiales de seguridad para la industria química (IEC 1 1 15 8-2, seguridad intrínseca).
- Profibus FMS (Fieldbus Message Specification). Para comunicación entre células de proceso o equipos de automatización. La evolución de Profibus hacia la utilización de protocolos TCP/IP para enlace al nivel de proceso hace que este perfil esté perdiendo importancia.

2. DECIVENET:

Bus basado en CAN. Su capa física y capa de enlace se basan en ISO 11898, y en la especificación de Bosh 2.0. DeviceNet define una de las más sofisticadas capas de aplicaciones industriales sobre bus CAN.

3. FOUNDATION FIELDBUS:

Un bus orientado sobre todo a la interconexión de dispositivos en industrias de proceso continuo. Su desarrollo ha sido apoyado por importantes fabricantes de instrumentación (Fisher-Rosemount, Foxboro,...)

4. INTERBUS:

Protocolo propietario, inicialmente, de la empresa Phoenix Contact GmbH, aunque posteriormente ha sido abierta su especificación. Normalizado bajo DIN 19258, norma europea EN 50 254. Fue introducido en el año 1984. Utiliza una topología en anillo y comunicación mediante un registro de desplazamiento en cada nodo. Se pueden enlazar buses periféricos al principal. Capa física basada en RS-485

5. FIP- WorldFIP:

Desarrollado en Francia a finales de los ochenta y normalizado por EN 50170, que también cubre Profibus. Sus capas física y de aplicación son análogas a las de Foundation Fieldbus H1 y Profibus PA. La división Norteamérica de WorldFIP se unió a mediados de los noventa a la Fieldbus Foundation en el esfuerzo por la normalización de un bus industrial común. Utiliza un modelo productor-consumidor con gestión de variables cíclicas, eventos y mensajes genéricos

6. LONWORKS:

La empresa Echelon, localizada en California, fue fundada en 1988. Comercializa el bus de campo LonWorks basado en el protocolo LonTalk y soportado sobre el NeuronChip. Alrededor de estas marcas ha construido toda una estructura de productos y servicios, hábilmente comercializados, dirigidos al mercado del control distribuido en domótica, edificios inteligentes, control industrial etc El protocolo LonTalk cubre todas las capas OSI. El protocolo se soporta en hardware y firmware sobre el NeuronChip.

7. SDS:

SDS ("Smart Distributed System") es, junto con DeviceNet y CANOpen, uno de los buses de campo basados en CAN más extendidos. Fue desarrollado por el fabricante de sensores industriales Honeywell en 1989. Se ha utilizado sobre todo en aplicaciones de sistemas de almacenamiento, empaquetado y clasificación automática. Se define una capa física que incluye alimentación de dispositivos en las conexiones. La capa de aplicación define autodiagnóstico de nodos, comunicación por eventos y prioridades de alta velocidad.

8. CANOpen:

Bus de campo basado en CAN. Fue el resultado de un proyecto de investigación financiado por la Comunidad Europea y se está extendiendo de forma importante entre fabricantes de maquinaria e integradores de célula de proceso. Está soportado por la organización CiA (CAN In Automation), organización de fabricantes y usuarios de CAN que también apoya DeviceNet, SDS etc.

9. MODBUS:

En su definición inicial Modbus era una especificación de tramas, mensajes y funciones utilizada para la comunicación con los PLCs Modicon. Modbus puede implementarse sobre cualquier línea de comunicación serie y permite la comunicación por medio de tramas binarias o ASCII con un proceso interrogación-respuesta simple. Debido a que fue incluido en los PLCs de la prestigiosa firma Modicon en 1979, ha resultado un estándar de facto para el enlace serie entre dispositivos industriales. Modbus Plus define un completo bus de campo basado en técnica de paso de testigo. Se utiliza como soporte físico el par-trenzado o fibra óptica. En la actualidad Modbus es soportado por el grupo de automatización Schneider (Telemecanique, Modicon,...).

A continuación se va a tratar con los buses de campo que son compatibles para Ethernet.

Ethernet: es un estándar de redes de área local para computadores con acceso al medio por contienda CSMA/CD. CSMA/CD (Acceso Múltiple por Detección de Portadora con Detección de Colisiones), es una técnica usada en redes Ethernet para mejorar sus prestaciones. El nombre viene del concepto físico de *ether*. Ethernet define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI. Las tecnologías Ethernet que existen se diferencian en estos conceptos:

- Velocidad de transmisión: Es la velocidad a la que transmite la tecnología.
- Tipo de cable: Es la tecnología del nivel físico que usa la tecnología.
- Longitud máxima: Es la distancia máxima que puede haber entre dos nodos adyacentes (sin estaciones repetidoras).
- Topología: Determina la forma física de la red. Bus si se usan conectores T (hoy sólo usados con las tecnologías más antiguas) y estrella si se usan hubs (estrella de difusión) o switches (estrella conmutada).

A continuación se especifican los anteriores conceptos en las tecnologías más importantes.

Tipo	Velocidad de transmisión	Estándar IEE	Tipo de cable	Tipo PHY	Distancia
Ethernet	10Mbps	802.3	Cobre(UTP) Fibra(MMF)	10 Base-T 10 Base-FL	100m 2Km
Fast Ethernet	100Mbps	802.3u	Cobre(UTP) Fibra(SMF/MMF)	100Base-TX 100Base-FX	100m 60Km/2Km
Ethernet Gigabit	1000Mbps	802.3ab/z	Cobre(UTP) Fibra(MMF) Fibra(SMF) Fibra(SMF)	1000Base-CX 1000Base-SX 1000Base-LX 1000Base-LH	100m 550m 10Km 70Km
Ethernet de 10 Gigabit	10000Mbps	802.3ae	Fibra(LAN-PHY) SDH(WAN-PHY) DWDM Cobre(InfiniBand) Cobre (UTP)	10GBase-R 10GBase-W 10GBase-LX4 10GBase-CX 10GBase-T	10-40Km 10-40Km 0.3-10Km 15-20Km 20-100m

En la actualidad se vive una auténtica revolución en cuanto a su desplazamiento hacia las redes industriales. Es indudable esa penetración. Diversos buses de campo establecidos como Profibus, Modbus etc. han adoptado Ethernet como la red apropiada para los niveles superiores.

Profibus → Profinet

Modbus → Modbus TCP/IP. Se utiliza el protocolo Modbus RTU con una interfaz TCP que funciona en Ethernet. TCP se refiere al Protocolo de Control de Transmisión e IP se refiere al Protocolo de Internet. Utiliza Ethernet para soportar datos de la estructura de mensajes Modbus entre dispositivos compatibles.

1.7 Solución adoptada

De todos los protocolos de bus de campo vistos se ha optado por Modbus, ya que presenta una serie de ventajas:

- Es un protocolo sencillo y que requiere poco trabajo de desarrollo
- Público
- Transmisión de datos sin restricciones

Dentro de Modbus, la versión que se ha escogido es Modbus TCP que es compatible con redes de tipo Ethernet, con lo que se puede integrar la red Modbus con una red de área local como las empleadas en oficinas y domicilios y se permite a la vez el acceso a la red global de Internet

1.8 Descripción de lo proyectado

El proyecto queda enmarcado en el ámbito de las de Comunicaciones Industriales. Los sistemas de comunicación industrial son mecanismos de intercambio de datos distribuidos en una organización industrial. El objetivo principal de las comunicaciones industriales consiste en el intercambio de información (de control) entre dispositivos remotos. El intercambio de datos puede ser:

- Intercambio de datos on-line y, en los niveles inferiores de la pirámide (sensores, actuadores, máquinas, células de fabricación, etc.), se exige el requisito de tiempo real.
- Intercambio de datos eficientes y de bajo coste temporal y económico. Datos en tiempo no real, como puede ser un correo electrónico.

Se desarrolla una aplicación en la cual se extrae información de un encoder que controla la velocidad de un motor. La adquisición de datos es en tiempo real, ya que el dato que proporciona el encoder es la velocidad del motor en el momento que se observa dicha variable.

Modbus TCP cumple con los requisitos que se desean para la aplicación sobre la que se acaba de describir.

A continuación se explica en detalle su modo de funcionamiento.

1.8.1 Comunicación:

En primer lugar se pasará a explicar estándar Modbus, para después detallar el subtipo de Modbus empleado.

Modbus: es un protocolo de comunicaciones basado en la arquitectura del tipo maestro/esclavo o cliente/servidor, donde el cliente necesita adquirir datos del servidor o enviar datos al servidor. Modbus es un protocolo de capa de aplicación. Es decir, su objetivo es el de definir reglas que permitan organizar e interpretar datos, con lo que se erige como un sistema que soporta el envío de mensajes, sin que tenga importancia la capa física. Esto le confiere una gran versatilidad, hasta el punto de que se pueden transferir mensajes a través de otras redes mediante técnicas en encapsulación y desencapsulación. En muchos aspectos recuerda a PROFIBUS/PROFINet.

Modbus fue diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar de facto en la industria, goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. Ya se mencionaron anteriormente las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones.

1. es público
2. su implementación es fácil y requiere poco desarrollo
3. maneja bloques de datos sin suponer restricciones

Modbus está basado en la arquitectura maestro/esclavo conectados mediante buses o redes. Con un único maestro cabe la posibilidad de tener varios esclavos bajo sus órdenes, con un máximo de 247 (esto depende del tipo de comunicación Modbus, en Modbus TCP el número es mayor).

En el protocolo Modbus Línea Serie la transmisión de datos siempre es iniciada por el maestro, y los nodos de los esclavos no transmiten datos si el nodo del maestro no envía una orden. El nodo del maestro puede comunicarse de dos formas con el nodo de los esclavos, unicast mode o broadcast mode. A continuación se muestra un ejemplo sencillo de una red Modbus:

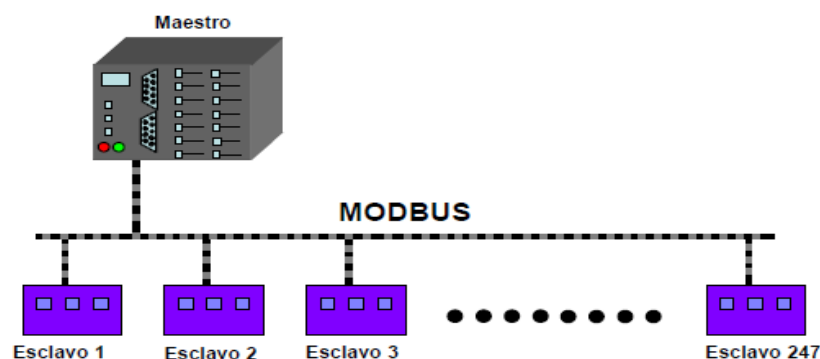


Figura 9.Red Modbus RTU

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA).

1.8.2 Tipo de Comunicación Modbus

Existen distintas versiones del protocolo Modbus:

- puerto serie (RS-232,RS-485)
- Ethernet(Modbus/TCP): es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP.

Dentro de la opción de puerto serie existen dos tipos de representación:

- Modbus RTU es una representación binaria compacta de los datos. Finaliza la trama con un suma de control de redundancia cíclica (CRC).
- Modbus ASCII es una representación legible del protocolo pero menos eficiente. Utiliza una suma de control de redundancia longitudinal (LRC).

Por último existe una versión extendida del protocolo y propiedad de Modicon:

- Modbus Plus (Modbus+ o MB+), dada la naturaleza de la red precisa un coprocesador dedicado para el control de la misma. Con una velocidad de 1 Mbit/s en un par trenzado sus especificaciones son muy semejantes al estándar EIA/RS-485 aunque no guarda compatibilidad con este.

Los protocolos Modbus RTU y Modbus ASCII se implementan a través del puerto serie, a diferencia de Modbus/TCP que se implementa a través del puerto Ethernet.

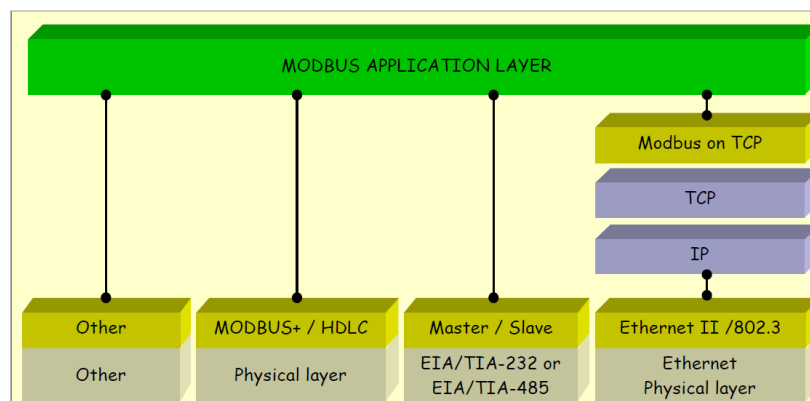


Figura 10. Estructura de las capas de cada tipo de comunicación de Modbus

1.8.2.1 Modbus RTU y Modbus ASCII

Existen dos tipos de transmisión serie: Modbus RTU y Modbus ASCII. Se define el contenido de los campos del mensaje. Se determina como se empaqueta la información en el mensaje, y cómo se decodifica. El modo de transmisión ha de ser el mismo para todos los elementos de la comunicación serie. Todos los dispositivos han de implementar el modo RTU, siendo la transmisión ASCII una opción.

- **Transmisión modo Modbus RTU:** Cuando los dispositivos utilizan este tipo de transmisión, cada 8-bit byte en el mensaje contiene dos 4-bit caracteres hexadecimales. La principal ventaja de utilizar este modo de transmisión es que la gran densidad del carácter permite un mejor rendimiento de velocidad de transmisión que el modo ASCII. Cada mensaje ha de ser transmitido en el mismo flujo de caracteres.

Formato (11 bits) por cada byte en modo RTU:

Sistema de cifrado: 8-bit binarios

Bits por Byte: 1 bit de comienzo

8 bits datos, el bit menos significativo se manda primero

1 bit de paridad de terminación

1 bit de parada

La paridad no es necesaria, pero existen modos que pueden utilizarlo para máxima compatibilidad entre otros productos. Es recomendable utilizar el modo de No paridad. El modo de no usar paridad requiere dos bits de parada.

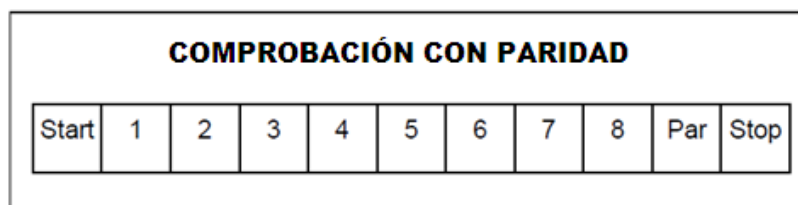


Figura 11. Secuencia de bits modo RTU



Figura 12. Secuencia de bits modo RTU (Sin paridad)

Utiliza la técnica Cyclical Redundancy Check (CRC) para el control de errores. Se explica la técnica más adelante.

Transmisión de los caracteres:

Cada carácter o byte es enviado en el orden de izquierda a derecha, se empieza por el bit de menos significado (LSB) y se acaba por el bit de mayor significado (MSB). Cada palabra transmitida posee 1 start bit, ocho bits de datos, 2 stop bits, sin paridad. En el modo RTU, cada byte de datos es transmitido como siendo una única palabra con su valor directamente en hexadecimal.

Descripción de la Estructura:

La red Modbus-RTU utiliza el sistema maestro-esclavo para el intercambio de mensajes. Permite hasta 247 esclavos, más solamente un maestro. Toda comunicación se inicia con el maestro haciendo una solicitud a un esclavo, y este contesta al maestro que le ha solicitado. En ambos los mensajes (pregunta y respuesta), la estructura utilizada es la misma: Dirección, Código de la Función, Datos y Checksum. Solo el contenido de los datos posee tamaño variable.

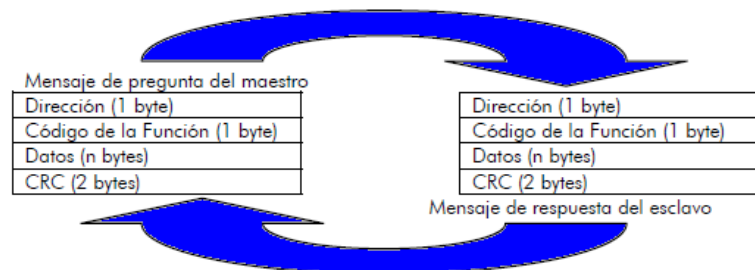


Figura 13. Estructura mensaje

DIRECCIÓN ESCLAVO	FUNCIÓN CÓDIGO	DATOS	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low CRC Hi

Figura 14. Estructura Mensaje RTU

- **Dirección:** El maestro inicia la comunicación enviando un byte con la dirección del esclavo para el cual se destina el mensaje. Al enviar la respuesta, el esclavo también inicia el mensaje con el su propia dirección, por lo que existe la posibilidad que el maestro conozca cual de los esclavos está enviándole la respuesta. El maestro también puede enviar un mensaje destinado a la dirección "0" (cero), lo que significa que el mensaje es destinado a todos los esclavos de la red (broadcast). En este caso, el maestro no recibe respuesta de ninguno de los esclavos.

→ Unicast mode: el maestro direcciona un único esclavo, una vez el esclavo ha recibido y procesado la petición retorna al maestro la respuesta. En este modo la transmisión Modbus consiste en dos mensajes: la petición del maestro y la respuesta del esclavo. Cada esclavo ha de tener una única dirección.

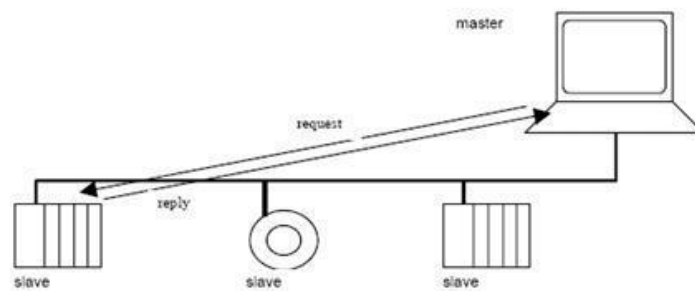


Figura 15.Unicast Mode

→Broadcast mode:el maestro puede enviar peticiones a todos los esclavos que estén conectados. Para este modo es necesario escribir comandos para la petición. Todos los elementos deben aceptar la función de escritura de broadcast. La dirección 0 se reserva para identificar el intercambio de broadcast, ya que los esclavos no emiten mensaje de respuesta y se desconoce si el mensaje ha llegado a su destino.

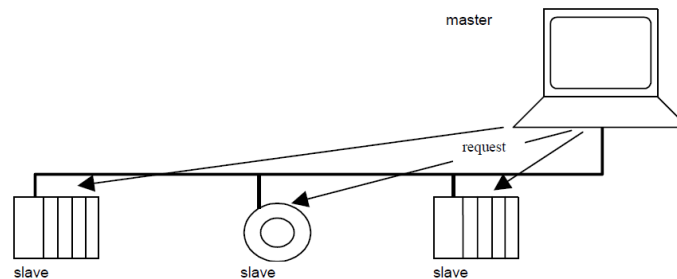


Figura 16.Broadcast Mode

Esto de da para el protocolo Modbus Línea Serie, ya que en TCP esto cambia.

- **Código de la Función:** Este campo también contiene un único byte, donde el maestro especifica el tipo de servicio o función solicitada al esclavo (lectura, escrita, etc.). De acuerdo con el protocolo, cada función es utilizada para acceder un tipo específico de dato.
- **Campo de Datos:** Campo con tamaño variable. El formato y el contenido de este campo dependen de la función utilizada y de los valores transmitidos.
- **CRC:** La última parte del mensaje es el campo para el chequeo de errores de transmisión. El método utilizado es el CRC-16 (Cycling Redundancy Check). Este campo está formado por dos bytes, donde primero es transmitido el byte menos significativo (CRC-), y después el más significativo (CRC+). El cálculo del CRC es iniciado cargándose una variable de 16 bits (referenciado a partir de ahora como variable CRC) con el valor FFFFh. Después se debe ejecutar los pasos de acuerdo con la siguiente rutina:

1. Se somete al primer byte del mensaje (solamente los bits de datos - start bit, paridad y stop bit no son utilizados) a una lógica XOR (O exclusivo) con los 8 bits menos significativos de la variable CRC, retornando el resultado en la propia variable CRC;
2. Entonces, la variable CRC es desplazada una posición a la derecha, en dirección al bit menos significativo, la posición del bit más significativo es rellenada con 0 (cero);
3. Después de este desplazamiento, el bit de flag (bit que fue desplazado para fuera de la variable CRC) se analiza, ocurriendo lo siguiente:
4. Si el valor del bit fuera 0 (cero), nada se ha hecho;
5. Si el valor del bit fuera 1 (uno), el contenido de la variable CRC es sometida a una lógica XOR con un valor constante de A001h y el resultado es regresado a la variable CRC.
6. Se repiten los pasos 2 y 3 hasta que se hayan realizado ocho desplazamientos;
7. Se repiten los pasos de 1 a 4, utilizando el próximo byte del mensaje, hasta que todo el mensaje haya sido procesado. El contenido final de la variable CRC es el valor del campo CRC que es transmitido en el final del mensaje. La parte menos significativa es transmitida primero (CRC-) y en seguida la parte más significativa (CRC+).

→La máxima longitud de estructura para RTU es de 256 bytes

- **Transmisión modo ASCII:** Cuando los dispositivos utilizan este tipo de transmisión, cada 8-bits byte en el mensaje contiene dos caracteres ASCII. Este modo de transmisión se utiliza cuando la comunicación física o las capacidades de los elementos no permiten comunicación mediante el modo RTU. Este modo de transmisión es menos eficiente que el modo RTU, ya que cada byte necesita dos caracteres. Se transmiten dos mensajes de 10 bits, de los cuales 7 bits son de datos lo que son 14 caracteres.

Ejemplo: El byte 0X5B se codifica como dos caracteres:

0x35="5" 0x42="B"

Formato (10 bits) por cada byte en modo ASCII:

Sistema de cifrado: Hexadecimal, ASCII caracteres 0-9,A-F

Bits por Byte: 1 bit de comienzo

7 bits datos, el bit menos significativo se manda primero

1 bit de paridad de terminación

1 bit de parada

La paridad no es necesaria, pero existen modos que pueden utilizarlo para máxima compatibilidad entre otros productos. Es recomendable utilizar el modo de No paridad. El modo de no usar paridad requiere dos bits de parada.

En el campo de error utiliza la técnica Longitudinal Redundancy Check (LRC).

Transmisión los caracteres:

Cada carácter o byte es enviado en el orden de izquierda a derecha, se empieza por el bit de menos significado (LSB) y se acaba por el bit de mayor significado (MSB).

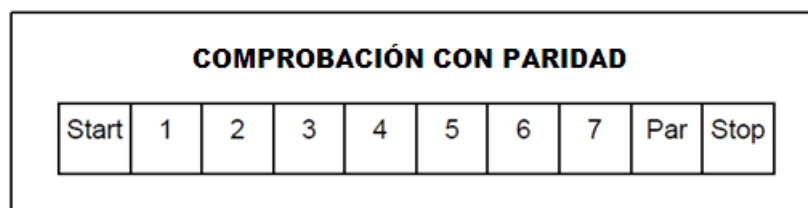


Figura 17. Secuencia bits modo ASCII

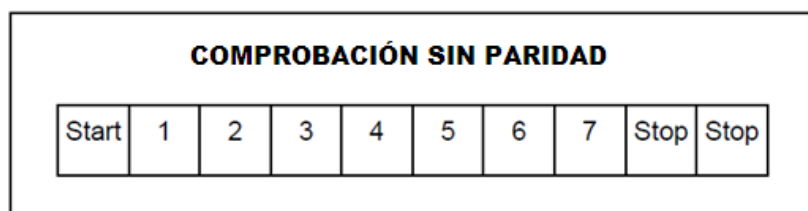


Figura 18. Secuencia bits modo ASCII (Sin paridad)

Descripción de la Estructura:

Una estructura muy común es la que se muestra a continuación:

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

Figura 19. Estructura mensaje ASCII

Cabe destacar que cada byte necesita dos caracteres para ser codificados. Para garantizar la compatibilidad de la aplicación Modbus entre el modo ASCII y el modo RTU, la máxima longitud de estructura de los datos para ASCII es (2x252), que corresponde al doble de la longitud requerida en el modo RTU (252). En consecuencia se puede decir que la máxima longitud de estructura de Modbus ASCII es de 513 caracteres.

1.8.2.2 Modbus TCP

Como se ha citado anteriormente Modbus es un protocolo que permite enviar mensajes sin depender de la capa física. La versión Ethernet es sencilla de obtener. Para ello es necesario el protocolo Ethernet y aplicar la pila TCP/IP, ya que es compatible con redes Ethernet convencionales. La finalidad de utilizar el protocolo TCP/IP es garantizar la comunicación, donde el medio físico de transmisión es el cable de Ethernet, y el TCP/IP es el protocolo con varias capas donde en una capa se hace el control de errores, en otra el enroutado de mensajes, etc . La interpretación de dicho mensaje se gestiona con el nivel de aplicación de Modbus. En consecuencia, TCP/IP y Ethernet forman el soporte para la transmisión de mensajes Modbus entre dispositivos compatibles con Ethernet.

El servicio de mensajes de Modbus proporciona una comunicación entre los dispositivos Cliente/Servidor que estén conectados en una red Ethernet TCP/IP.

El modelo Cliente/Servidor se basa en cuatro tipos de mensajes:

- Modbus Petición, es el mensaje que se envía desde el Cliente a la red para que se inicie la transmisión.
- Modbus Confirmación, es el mensaje de respuesta recibido en el lado del Cliente.
- Modbus Indicación, es el mensaje de petición recibido en el lado del Servidor.
- Modbus Respuesta, es el mensaje de respuesta que envía el Servidor.

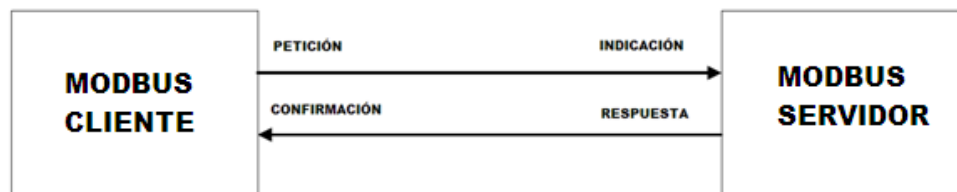


Figura 20. Modelo Cliente/Servidor

El servicio de mensajes de Modbus, modelo Cliente/Servidor, es usado para el intercambio de información en tiempo real. Puede ser utilizado entre:

- Entre aplicaciones de dos dispositivos
- Entre aplicación de un dispositivo y otro dispositivo
- Entre aplicaciones HMI/SCADA y otros dispositivos
- Entre un PC y un dispositivo programado en la línea de servicio

En la práctica, el servicio de mensajes emplea una técnica, la cual consiste en encapsular cada mensaje Modbus en una trama TCP (sin incluir los dos bytes del campo de comprobación de errores de Modbus, ni la dirección del esclavo). El campo de comprobación de errores no se incluye debido a que Modbus TCP/IP contiene su propio sistema frente a errores en la capa de BUSCAR. La dirección del Esclavo se elimina ya que la conexión, la dirección IP de origen y destino y las direcciones MAC de los dispositivos conectados, se definen en los niveles de transporte, de red o de enlace.

Esto supone una gran ventaja, ya que el límite que existía en el modo RTU de un máximo de 247 dispositivos se supera con creces y aumenta la posibilidad de conectar más dispositivos.

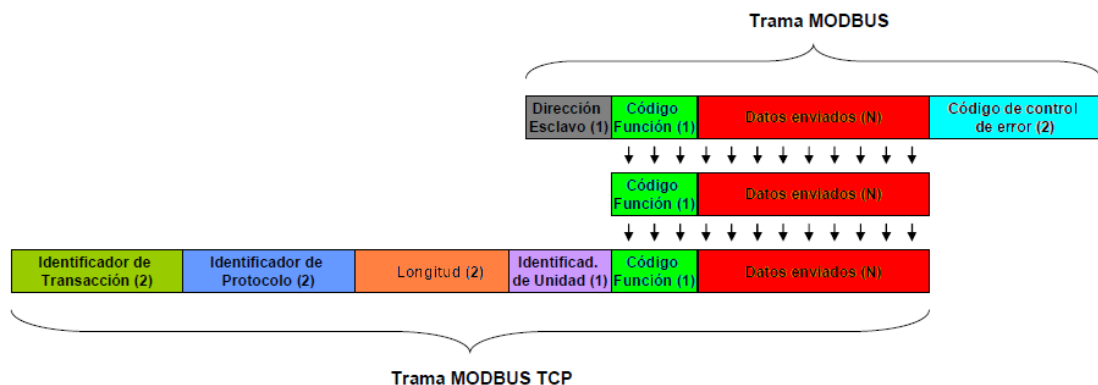


Figura 21. Encapsulado de un Mensaje en una trama TCP

La cabecera consta de cuatro campos:

- Identificador de Transacción, se emplea para distinguir los mensajes consecutivos que se envían través de una conexión TCP, sin que el equipo que envía dichos mensajes obtenga una respuesta de dichos mensajes.
- Identificador de Protocolo, este campo siempre tiene el valor 0. En próximas extensiones del protocolo se podrán observar diferentes variantes.
- Longitud, indica en número de bytes que se obtienen como suma del identificador unidad, el código de función y los datos enviados.
- Identificador de Unidad, se utiliza para identificar un servidor remoto que pertenece a una red que no es TCP/IP. En una red del tipo TCP/IP este campo aparece a 0 o todo unos.

Los paquetes de datos TCP se transmiten a través de un puerto que existe reservado exclusivamente para transacciones Modbus dicho puerto es el 502.

Descripción del Protocolo:

El sistema de comunicación de Modbus TCP/IP incluye distintos dispositivos:

- Dispositivos Cliente y Servidor de Modbus TCP/IP conectados a una red TCP/IP
- La interconexión de dispositivos con un “puente”, router o puertas de enlace para la conexión entre una red TCP/IP y una subred de línea serie, la cual permite conexiones de un Cliente línea Modbus Serie y un dispositivo Servidor final.

Se muestra a continuación una ilustración que representa la arquitectura de una comunicación Modbus:

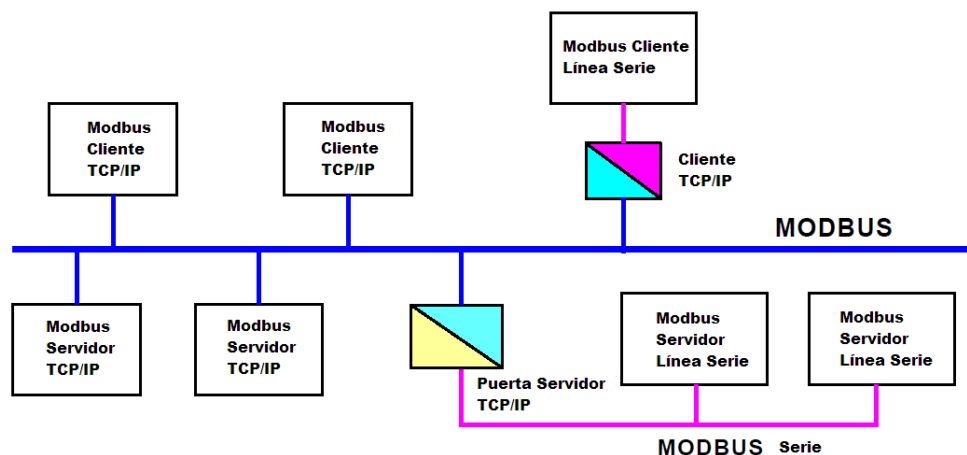


Figura 22.Arquitectura Comunicación Modbus

1.8.2.3 Modbus +

La versión Modbus+ ha sido propiedad del grupo Schneider Electric, el cual ha trabajado en dicha versión. Actualmente la versión Modbus+ está gestionada por el grupo Modbus Ida.

Esta versión de Modbus tiene un sistema más complejo frente a los sistemas de RTU y TCP. Permite la gestión del tráfico con diferentes prioridades, como se gestionan en una red DeviceNet, Profibus o Profinet, lo que supone una gran ventaja. Esta ventaja se debe que el método de control de acceso es diferente al de los sistemas Modbus RTU y Modbus TCP.

La diferencia en el método empleado se basa en técnica empleada, ya que en los sistemas anteriores el maestro realizaba un sondeo de los esclavos. En este caso, la técnica empleada es High-Level Data Link Control (HDLC), que consiste en el paso de testigo en forma de anillo lógico entre estaciones maestro.

Tipos de red:

En esta versión se habla de interconexión de redes, no de una sola red. Se diferencian dos tipos de redes:

- Network, es una red formada por un serie de nodos, en general controladores programables (PLCs) o PCs, en donde los nodos se comunican a través del paso de un testigo.
- DIO(Distributed I/O) Network, es una red que ha sido diseñada para dar servicio a dispositivos de campo de entrada y salida (I/O field devices). La red dispone de un PLC o PC como mínimo, y de uno o varios dispositivos de entrada y salida. Los dispositivos de entrada y salida pueden ser de dos tipos: DIO drop adapter (adaptador para módulos de entrada y salida con uno o varios módulos de salida), o del tipo TIO module (módulo terminal de entradas y salidas).

A continuación se muestra una imagen que representa la interconexión de varias redes Modbus

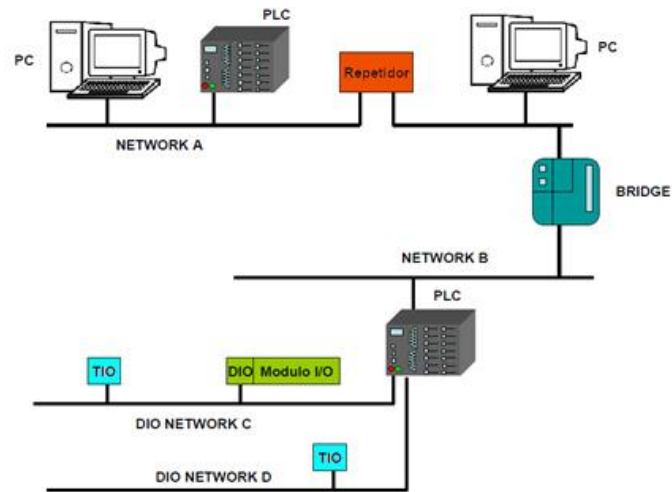


Figura 23. Interconexión Red Modbus

1.8.2.4 Resumen Modbus RTU, ASCII, Modbus+

Características MODBUS RTU y ASCII:

- Se permite la conexión de un máximo de 247 dispositivos
- El tamaño de paquetes a nivel de enlace de datos ha de tener un mínimo de 4 bytes
- Topología en bus sin ramificaciones
- Velocidad de transmisión típicas en Modbus: Modbus RTU y ASCII → 9.6Kbps O 19.2 Kbps
- Los mensajes se direccionan de modo maestro-esclavo

Ventajas:

- No requiere tarjetas de red específicas, lo que hace más sencillo su implementación

Desventajas:

- No permite discriminar el tráfico con diferente prioridad
- Los nodos se limitan a un número máximo de 247
- Al ser un protocolo maestro-esclavo no se puede conocer la existencia de algún fallo en los esclavos. Para ello es necesario un constante sondeo por parte del maestro, que conlleva a una sobreutilización del ancho de banda.
- No existe un mecanismo de seguridad que evite el envío de tramas erróneas o frente ataques externos.

Características MODBUS TCP:

- El número de nodos que se pueden interconectar depende del límite del switch (existen valores actuales que alcanzan 1024 conexiones), como sucede en el resto de redes que se basan en Ethernet como por ejemplo Profinet, Ethernet/IP...
- El tamaño de paquetes a nivel de enlace de datos ha de ser como mínimo de 72 bytes, existiendo la posibilidad de alcanzar los 1526. Existen versiones que pueden aumentar el tamaño de la trama con una extensión, como Gigabit Ethernet.
- La topología es en árbol, que se compone por switches a los que se conectan dispositivos en topología Estrella.
- La velocidad y longitud dependen la una de la otra, por ejemplo, para 10 Mbps la longitud máxima es 2500 metros, y para 100Mbps la longitud es de 400 metros.

Ventajas:

- La velocidad de transmisión es muy alta
- Debido al desarrollo de los switches Ethernet se ha convertido en un protocolo determinístico y versátil. Es capaz de trabajar a distintos niveles jerárquicos de automatización industrial
- Frente a Modbus RTU y Modbus ASCII existe la ventaja que en Modbus TCP no existe límite en el número de nodos que se pueden conectar a la red. Además la gestión de errores en los esclavos se realiza sin la necesidad de realizar un sonde continuamente

Desventajas:

- No permite discriminar tráfico con diferente prioridad
- En aplicaciones donde se han de transmitir pocos datos pero con gran determinación, como aplicaciones de control o de seguridad, este modo es superado por la simplicidad de CAN. Esto se debe a que los paquetes de Ethernet tienen un mínimo de 72 bytes, mientras que CAN tiene un máximo de 14 bytes

Características Modbus+:

- Se trata de una red con dos niveles de determinismo. Un nivel se utiliza para el tráfico de datos entre nodos de una red Modbus en tiempo real. El otro nivel se utiliza para el tráfico de datos entre nodos de redes Modbus interconectadas mediante puentes en tiempo no real.
- Su versatilidad permite establecer relación entre el bus de campo y Ethernet, debido a la posibilidad de moverse entre los niveles de control y de célula.

Ventajas:

- Posee la capacidad de discriminar tráfico de diferente prioridad en tiempo real y en tiempo no real.

1.8.3 Conexión PC con PLC, Modbus aplicado a Labview:

Se desea comunicar un PC con un PLC para compartir variables de estado. Dichas variables pueden ser de distintos tipos. Para llevar a cabo esta práctica primero se va a realizar una demostración en labview con un maestro que simulará el ordenador y un esclavo que hará las veces de PLC.

El primer paso es crear un proyecto que funcione como maestro, donde se reflejen los datos que varían.

1.8.3.1 Creación del maestro (servidor):

Se comienza por abrir el programa Labview, el cual presenta el siguiente aspecto:



Figura 24. Página Inicial de Labview

A continuación se crea el nuevo proyecto:

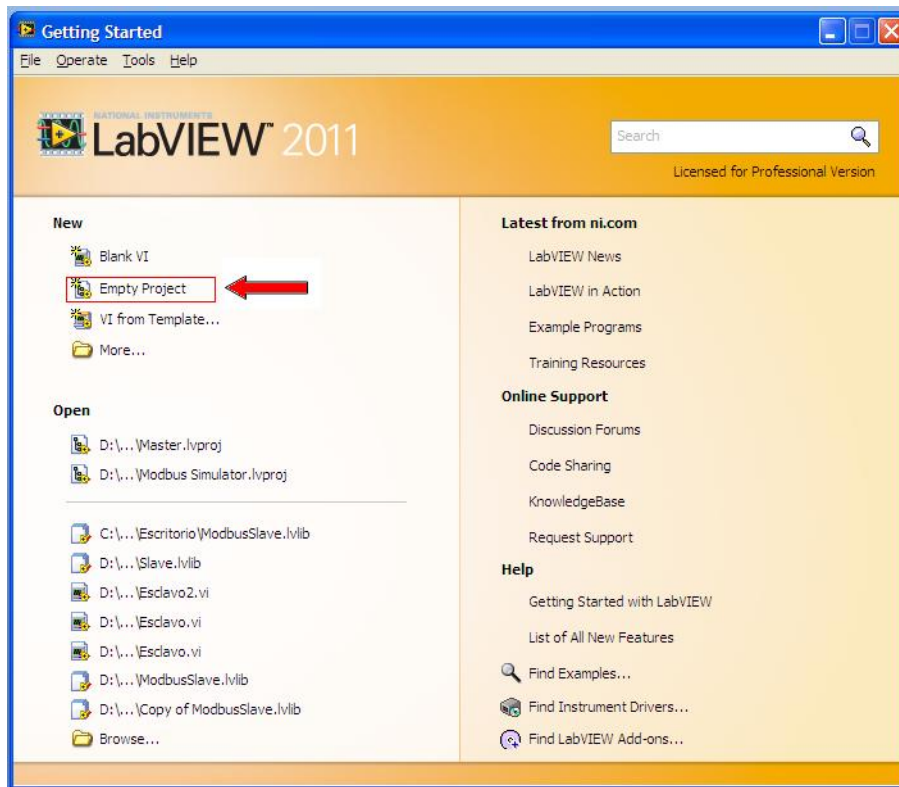


Figura 25. Creación del Proyecto

El proyecto aparece con el siguiente formato:

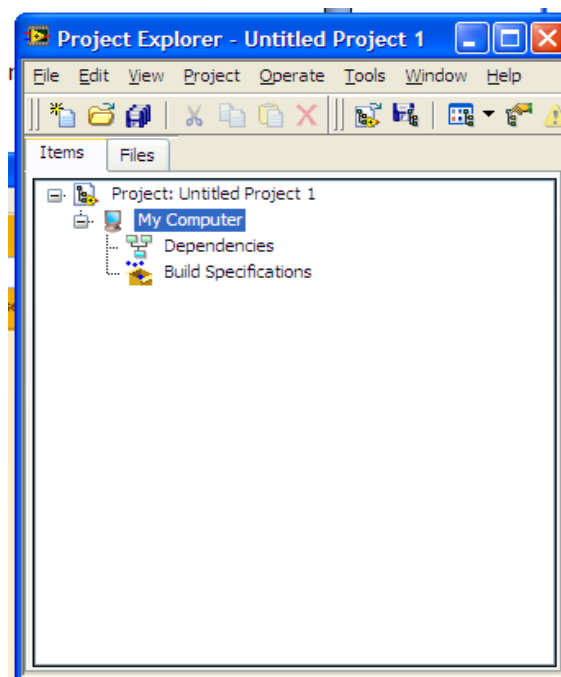


Figura 26. Página Inicial del Proyecto

Se guarda el proyecto con el nombre de Master o Maestro, como el usuario desee en el momento de la creación, ya que el nombre solo va a ser una forma de identificar el proyecto:

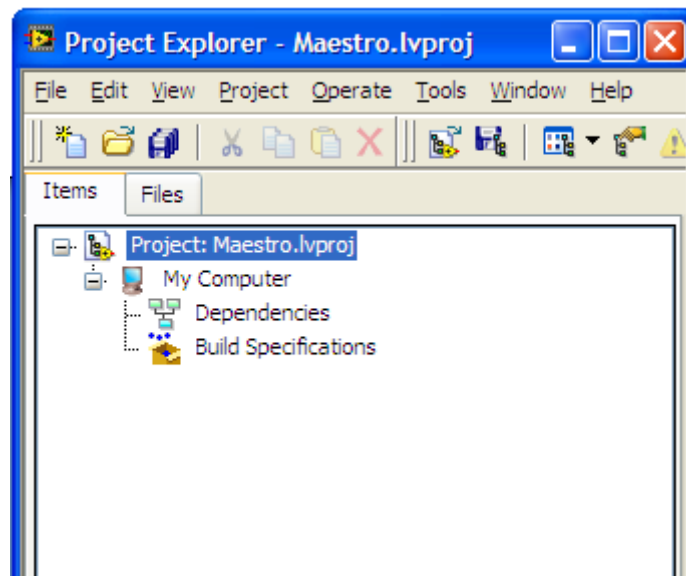


Figura 27. Guardar el proyecto

El siguiente paso consiste en añadir una librería al proyecto, la cual va a contener las variables que se desean compartir:

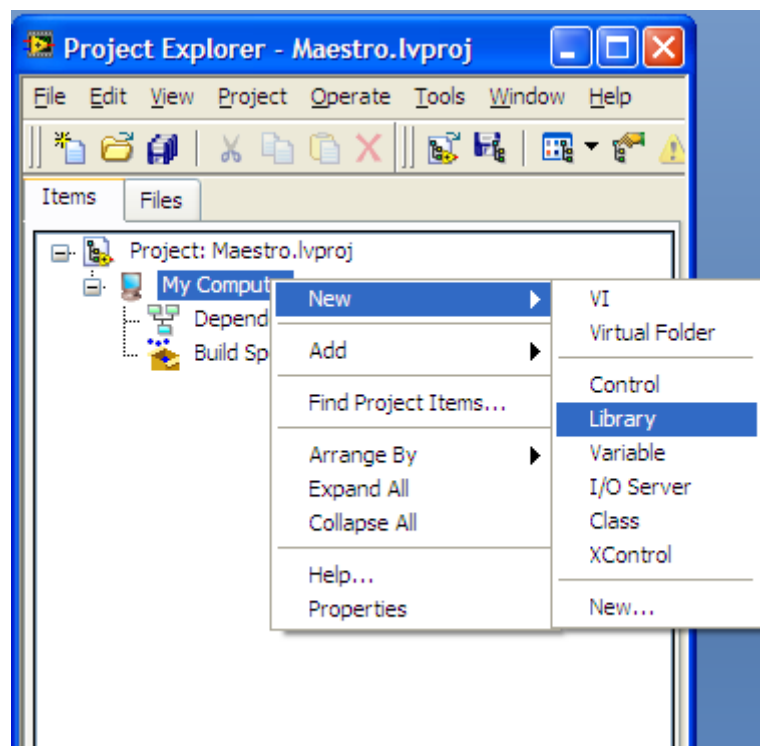


Figura 28. Añadir librería al proyecto

Se guarda la librería con el nombre que se desee, en este caso es recomendable utilizar el nombre del proyecto para así no confundir las librerías con las de otros proyectos:

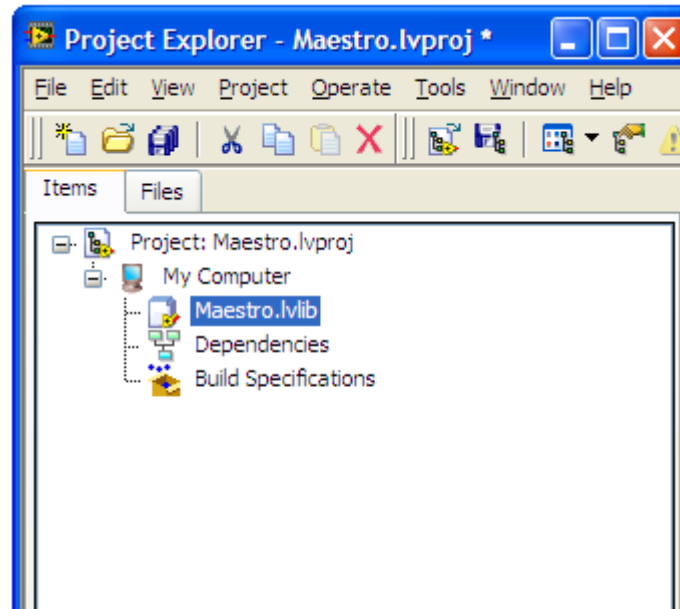


Figura 29. Guardar librería

El siguiente paso es añadir a la librería un I/O Server, el cual va a permitir que las variables del proyecto sean compartidas y accesibles:

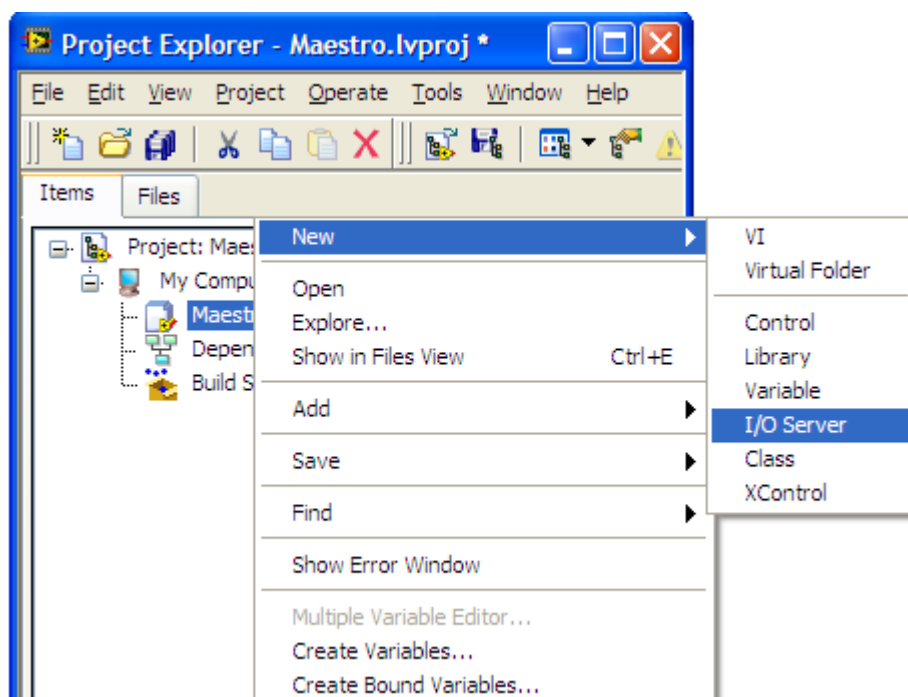


Figura 30. Añadir I/O Server a la librería

Aparece el siguiente cuadro, en el cual se permite la elección del tipo de comunicación que se va a utilizar. En este caso es comunicación Modbus, por lo que se escogerá la opción de Modbus, no Modbus Slave, ya que se está creando el maestro:

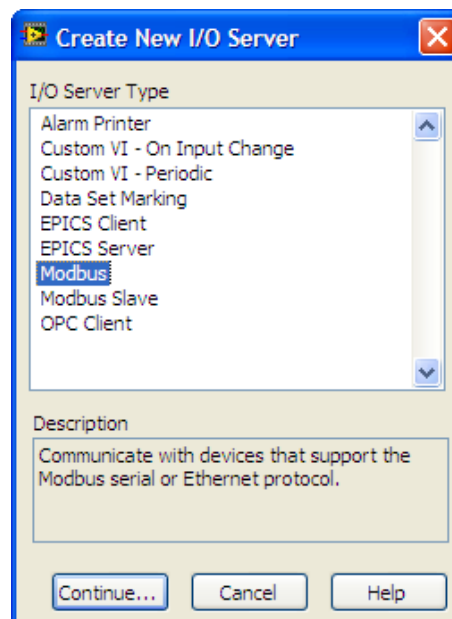


Figura 31. Elección del tipo de comunicación

Se dispone de las versiones de Modbus Serial y Ethernet. De ambas se escogerá Ethernet, porque en realidad es Modbus TCP. Es decir, se trata simplemente de una cuestión de notación. Hay sitios donde se llama a Modbus TCP como Modbus Ethernet.

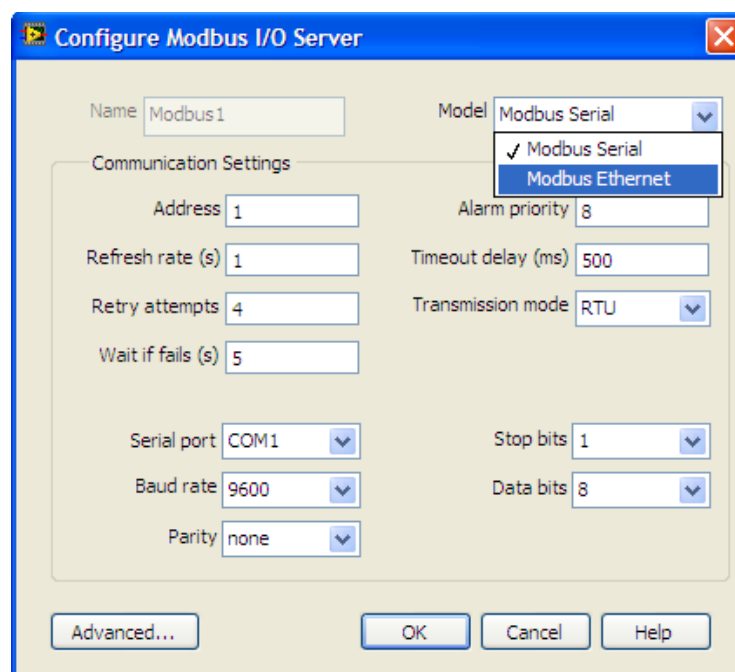


Figura 32. Elección modo Ethernet

Una vez se escoge la opción de Ethernet es necesario indicar la dirección IP. En este caso se utiliza un IP en concreto que es la del propio ordenador, pero se puede utilizar la IP del dispositivo deseado, como puede ser un PLC:

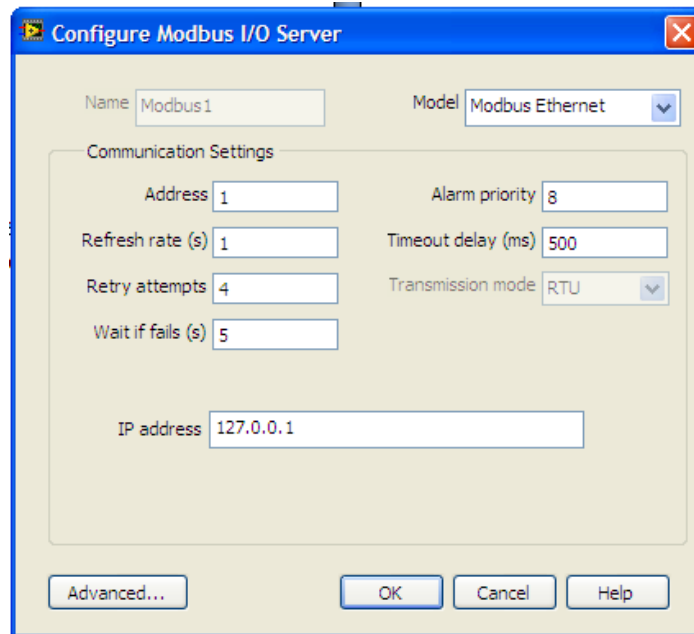


Figura 33. Selección dirección IP

Se observa que en la librería se ha creado un nuevo elemento:

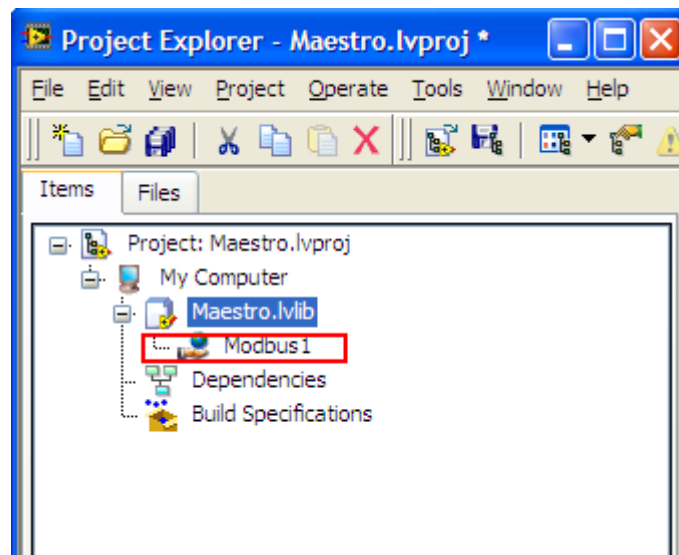


Figura 34. Creación del I/O Server

Ahora se deben crear las variables que se consideren necesarias para llevar a cabo el proyecto. En este caso se crearan varias de prueba, unas de tipo booleano y otras de tipo entero. Para ello es necesario seguir varios pasos:

Primero se crean las variables:

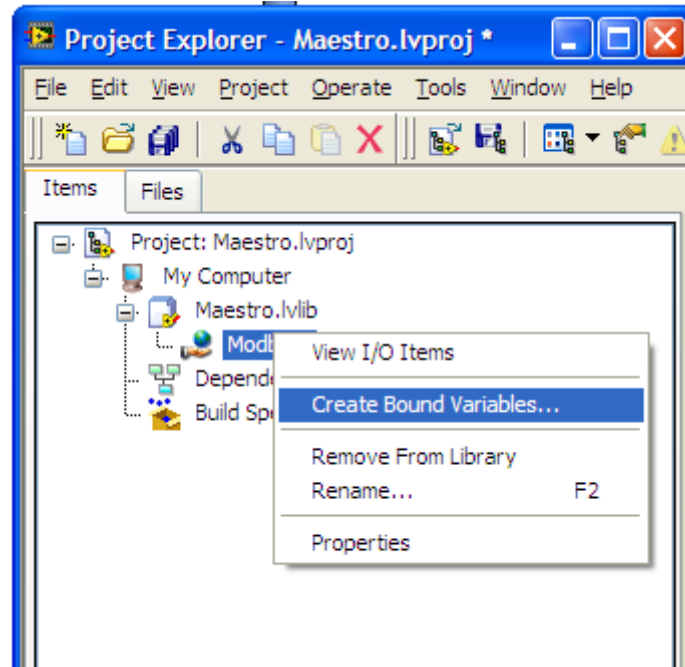


Figura 35. Creación de Variables

En segundo lugar se elige el rango y tipo de variables:

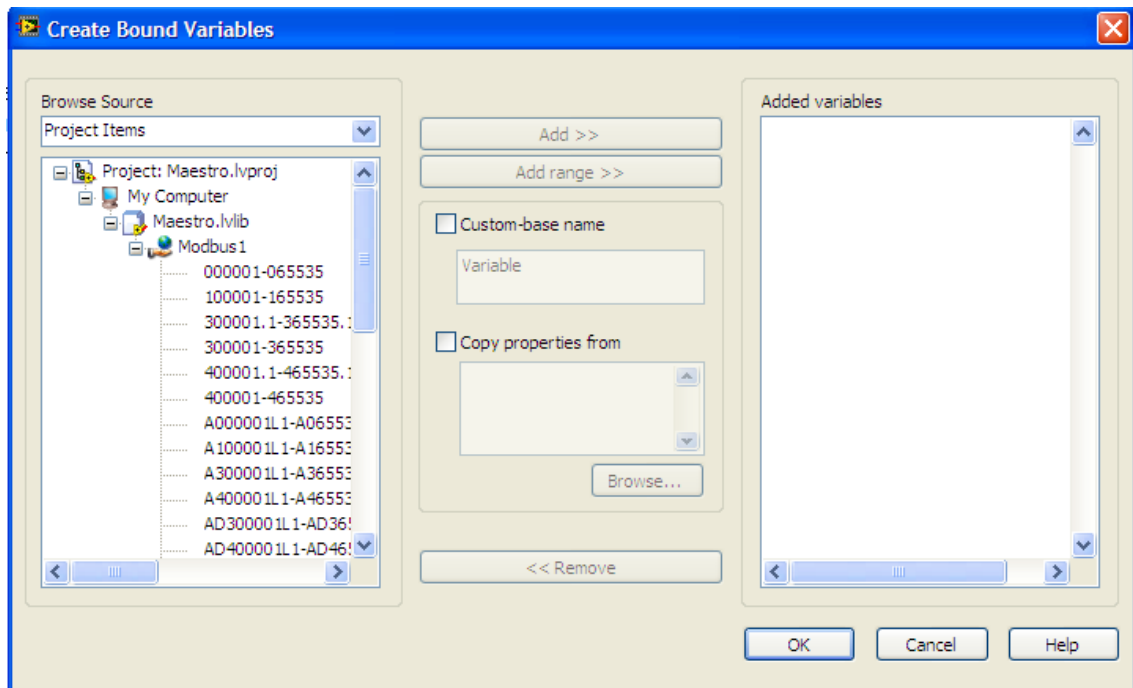


Figura 36. Selección del rango de variables

Se ha de tener especial cuidado a la hora de elegir las variables, ya que hay variables que son de lectura y escritura y otras solo de lectura.

A modo de ejemplo, en primer lugar se eligen variables de tipo booleano que son de escritura y lectura. Estas van de la dirección 000001 a la 065535. También se escogerán variables de tipo entero (de lectura y escritura), que son las que van de la dirección 400001 a la 46535:

Se seleccionan diez variables de tipo booleano y se añaden:

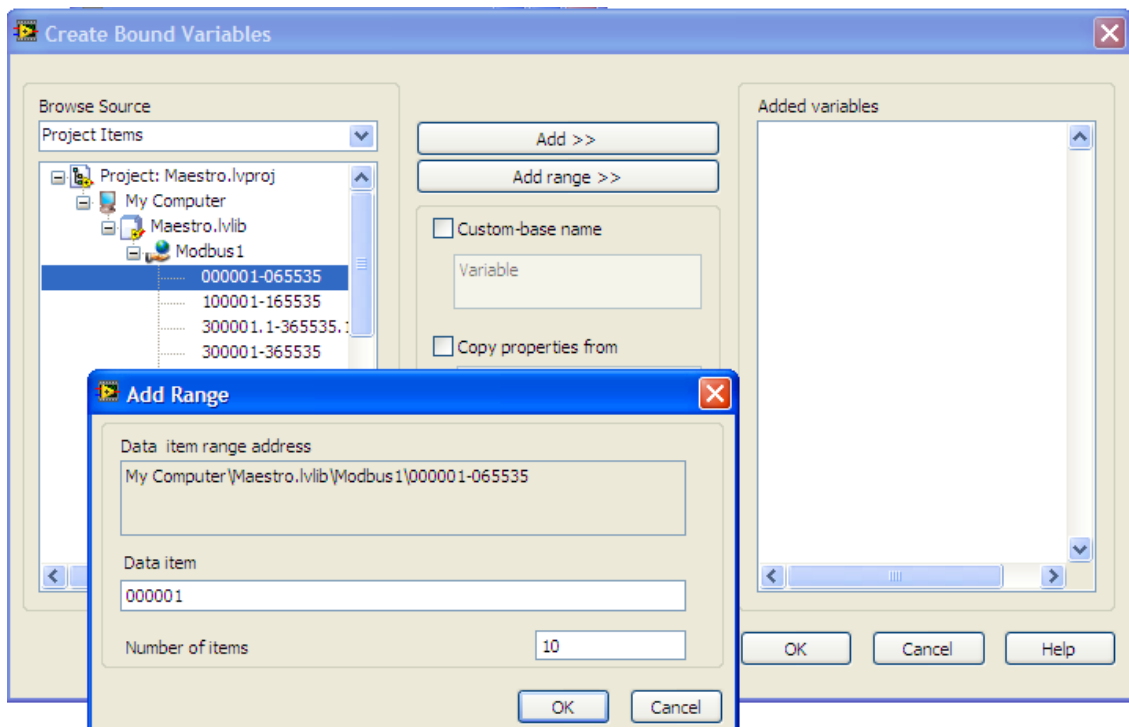


Figura 37. Creación del nº de variables deseadas

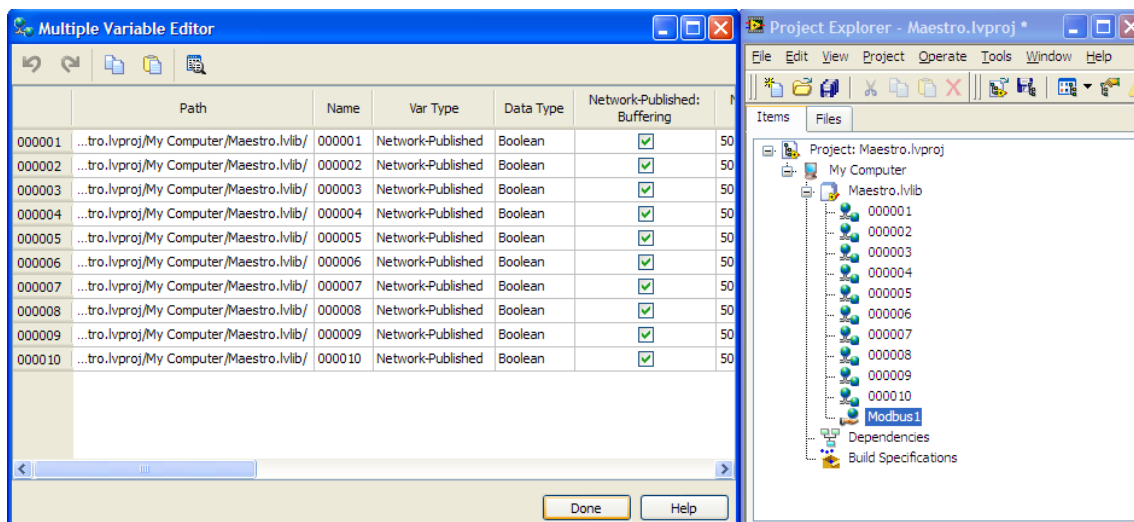


Figura 38. Variables dentro del proyecto

Se observa que las variables se han adjuntado al proyecto. Se repite el mismo procedimiento para las variables de tipo entero:

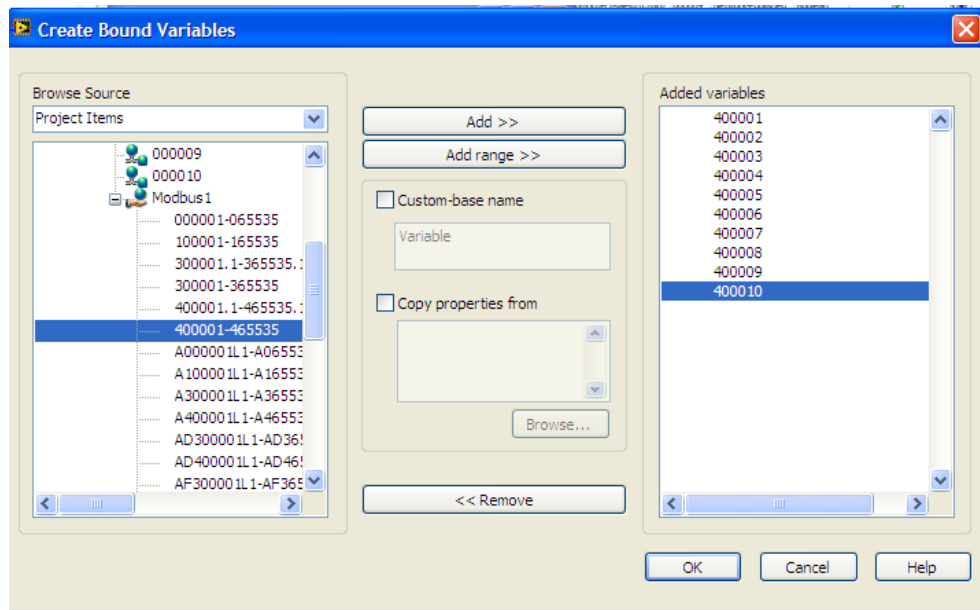


Figura 39. Variables tipo entero

Se observa que las variables han sido añadidas al proyecto:

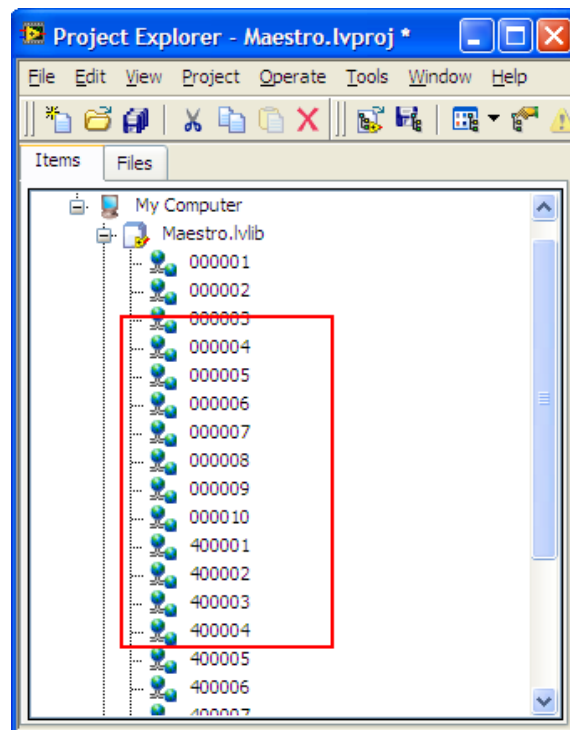


Figura 40. Variables añadidas

Es necesario crear un panel frontal para poder introducir variables y hacerlas funcionar. Para ello se hace una explicación breve de cómo funciona labview.

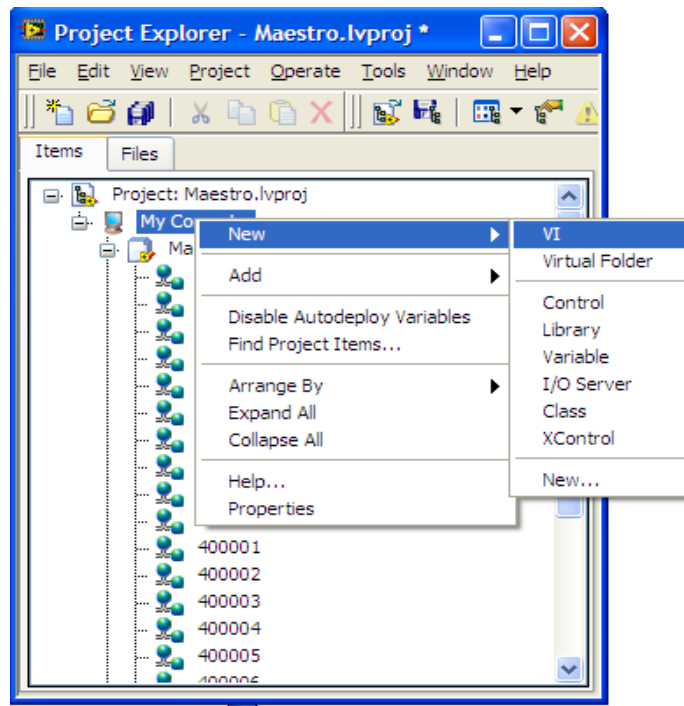


Figura 41. Creación del panel frontal

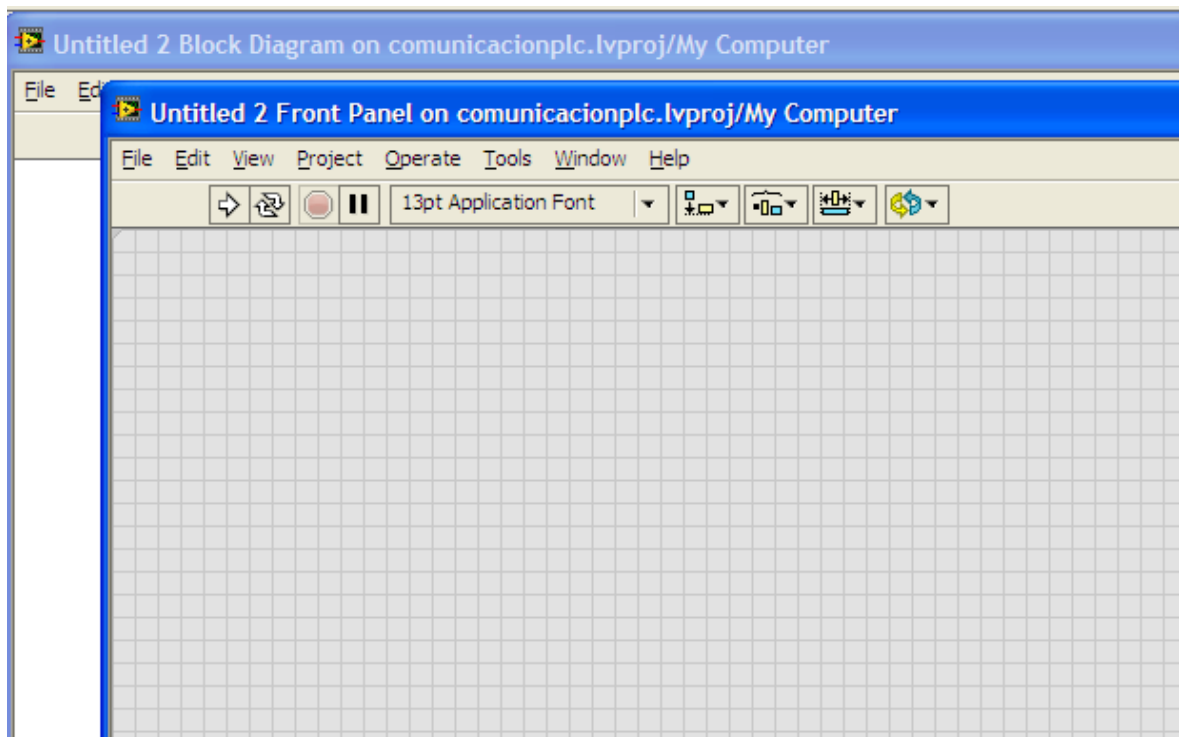


Figura 42. Panel Frontal creado

1.8.3.2 Crear Esclavo(Cliente):

El servidor en este caso es el PLC, donde labview accederá a leer las variables. Para ello es necesario indicar al propio PLC que va a trabajar como servidor. Esto se realiza cargando al PLC un programa con la función servidor como se muestra a continuación:

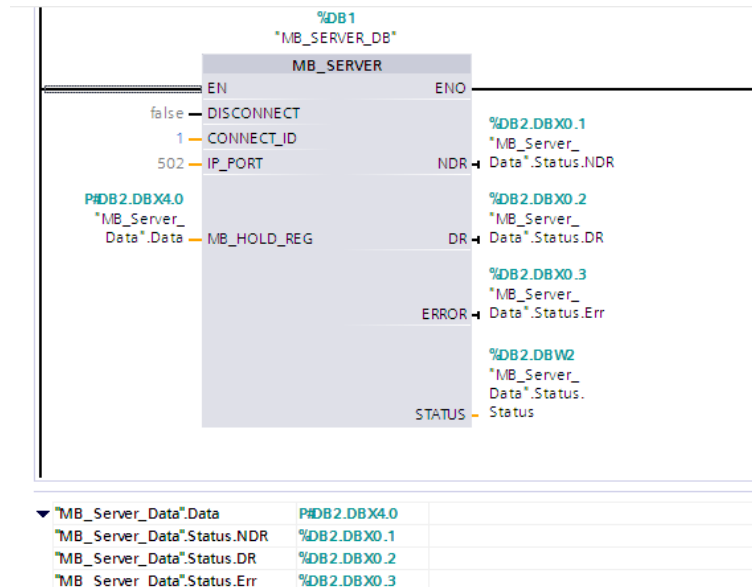


Figura 43. Bloque Servidor

El bloque "MB_SERVER" permite la comunicación como servidor Modbus TCP a través de la conexión PROFINET de la CPU S7-1200. Para utilizar esta instrucción no se requiere ningún módulo de hardware adicional. La instrucción "MB_SERVER" permite procesar peticiones de conexión de un cliente Modbus TCP, recibir peticiones de funciones Modbus y enviar mensajes de respuesta.

Conexiones múltiples a servidor:

Pueden establecerse conexiones múltiples a servidor. Gracias a ello, una sola CPU puede establecer conexiones con varios clientes Modbus TCP al mismo tiempo. Un servidor Modbus TCP puede admitir varias conexiones TCP (el número máximo de conexiones depende de la CPU utilizada). El total de conexiones de una CPU, incluidos los clientes Modbus TCP y los servidores, no debe exceder el número máximo de conexiones admitido.

Para las conexiones de servidor deben respetarse las siguientes reglas:

- Cada conexión "MB_SERVER" debe utilizar un DB de instancia unívoco.
- Cada conexión "MB_SERVER" debe establecerse con un número unívoco de puerto IP. Se admite una sola conexión para cada puerto.
- Cada conexión "MB_SERVER" debe utilizar una ID de conexión unívoca.

Para cada DB de instancia de la instrucción debe utilizarse la correspondiente ID de conexión. Las ID de conexión y los DB de instancia se agrupan por pares y deben ser unívocos para cada conexión.

- Para cada conexión debe llamarse separadamente la instrucción "MB_SERVER".

La tabla siguiente muestra los parámetros de la instrucción "MB_SERVER":

Parámetro	Declaración	Tipo de datos	Descripción
DISCONNECT	Input	BOOL	La instrucción "MB_SERVER" establece una conexión pasiva con un módulo interlocutor. El servidor reacciona a una petición de conexión TCP de cada dirección IP solicitante. 0: Puede iniciarse la conexión de comunicación pasiva 1: Inicialización del establecimiento de la conexión. Este parámetro permite controlar cuándo se acepta una petición de conexión. Si la entrada esta activada en este parámetro, no se ejecutan otras operaciones.
CONNECT_ID	Input	UNIT	Mediante este parámetro se identifica una conexión en la CPU de modo unívoco. Cada una de las instancias de las instrucciones "MB_CLIENT" y "MB_SERVER" debe utilizar una ID unívoca en el parámetro CONNECT_ID.
IP_PORT	Input	UINT	Valor de arranque=502. El número del puerto IP determina qué puerto IP se vigila para peticiones de conexión del cliente Modbus. Estos números de puerto TCP no deben usarse para la conexión pasiva de la instrucción "MB_SERVER": 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 y 34964.
MB_HOLD_REG	InOut	VARIANT	Puntero al registro de parada Modbus de la instrucción "MB_SERVER": Utilice como registro de parada un bloque de datos global con acceso estándar o un área de memoria (M). El registro de parada contiene los valores a los que está autorizado a acceder un cliente Modbus mediante las funciones Modbus 3 (lectura), 6 (escritura) y 16 (lectura).
NDR	Output	BOOL	"New Data Ready":0: No hay datos nuevos 1: El cliente Modbus ha escrito datos nuevos
DR	Output	BOOL	"Data Read":0: No se han leído datos 1: El cliente Modbus ha leído datos
ERROR	Output	BOOL	Si se produce un error durante una llamada de la instrucción "MB_SERVER", la salida del parámetro ERROR se ajusta a TRUE. La causa detallada del error se muestra en el parámetro STATUS.
STATUS	Output	WORD	Código de error de la instrucción.

Parámetros MB_SERVER

1.8.3.3 Acceder a las variables:

Una vez se ha cargado el programa en el PLC se accede a leer las variables del PLC desde Labview. Este proceso se realiza a través del Distributed Manager. Es necesario tener presente en qué parte de la memoria del PLC se direccionan las variables, ya que existen distintos tipos de variables tanto en formato como en el modo de acceso (escritura o lectura).

En el anexo se adjunta un cuadro en el cual aparece especificado el direccionamiento de las variables. Ha sido necesario investigar a fondo la ayuda del PLC para la zona de memoria, así como la de Labview. Por ello ha habido etapas de la programación que no son todo lo correctas que se desean.

El principal problema que ha surgido ha sido poder identificar ciertas variables del PLC en Labview y viceversa. Debido a que Labview restringe el acceso a ciertos tipos de datos y no permite la escritura en todos ellos.

El acceso al Distributed Manager se realiza desde la ventana del proyecto en la opción TOOLS como se muestra a continuación:

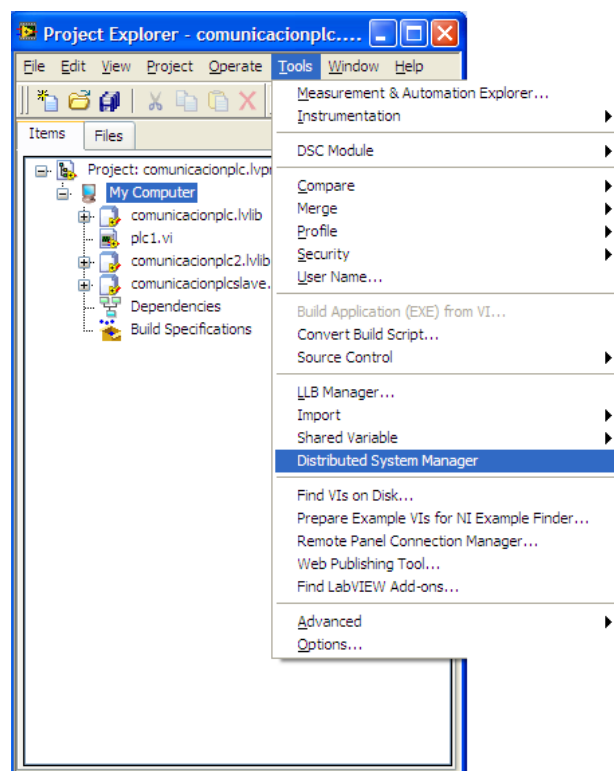


Figura 44. Acceso a Distributed Manager

Para asegurarse que la comunicación es correcta existe una variable que indica false si la comunicación está bien o true si en la comunicación existe algún fallo:

Modbus1	Variable	Estado	Acción
000001	000001	false	Read/Write
000001-065535	000001-065535		Read/Write
000002	000002	false	Read/Write
000003	000003	false	Read/Write
000004	000004	true	Read/Write
000005	000005	true	Read/Write
100001	100001	false	Read
100001-165535	100001-165535		Read
100002	100002	false	Read
100003	100003	false	Read
100004	100004	false	Read
100005	100005	false	Read
300001-365535	300001-365535		Read
300001.1-365535.16	300001.1-365535.16		Read
400001-465535	400001-465535		Read/Write
400001.1-465535.16	400001.1-465535.16		Read
A000001L1-A065535L1	A000001L1-A065535L1		Read/Write
A100001L1-A165535L1	A100001L1-A165535L1		Read
A300001L1-A365535L1	A300001L1-A365535L1		Read
A400001L1-A465535L1	A400001L1-A465535L1		Read/Write
AD300001L1-AD365534L1	AD300001L1-AD365534L1		Read
AD400001L1-AD465534L1	AD400001L1-AD465534L1		Read/Write
AF300001L1-AF365534L1	AF300001L1-AF365534L1		Read
AF400001L1-AF465534L1	AF400001L1-AF465534L1		Read/Write
AS300001L1-AS365535L1	AS300001L1-AS365535L1		Read
AS400001L1-AS465535L1	AS400001L1-AS465535L1		Read/Write
ASD300001L1-ASD365534L1	ASD300001L1-ASD365534L1		Read
ASD400001L1-ASD465534L1	ASD400001L1-ASD465534L1		Read/Write
CommFail	CommFail	false	Read
D300001-D365534	D300001-D365534		Read
D400001-D465534	D400001-D465534		Read/Write
F300001-F365534	F300001-F365534		Read
F400001-F465534	F400001-F465534		Read/Write

Figura 45. Comunicación Correcta

El principal problema surgido ha sido averiguar que el PLC requería un programa concreto para poder comunicarse con el PC. Debido a estos incidentes hubo que ponerse en contacto con el servicio técnico de Labview, donde tuvieron que controlar el propio PC desde la sede situada en Madrid. Tras realizar la prueba sin éxito, un técnico de Labview accedió a venir al laboratorio para realizar las comprobaciones necesarias hasta dar con los fallos y finalmente resolver los problemas y llevar a cabo la comunicación entre ambas máquinas. Una vez se comprueba que la comunicación es correcta, se procede a la lectura de variables. Se realiza un pequeño ejemplo.

Si la entrada del PLC IO.0 (100001) está conectada, las salidas Q0.3 (000004) y Q0.4 (000005) se encuentran desactivadas. Si la entrada IO.0 está desactivada, las salidas Q0.3 y Q0.4 se activan. Se carga el programa siguiente al PLC y se observa lo que ocurre:

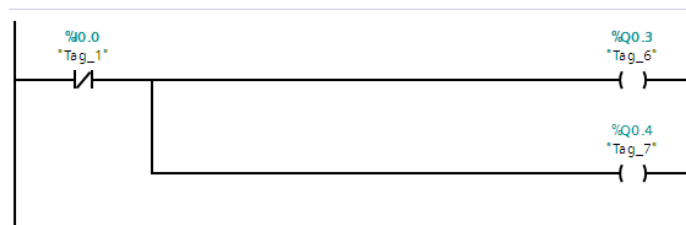


Figura 46. Bloque de programa del PLC



Figura 47. Entrada I0.0 activada, salidas Q0.3 y Q0.4 desactivadas

Address	Value	Access
000001	false	Read/Write
000002	false	Read/Write
000003	false	Read/Write
000004	false	Read/Write
000005	false	Read/Write
100001	true	Read
100002	false	Read
100003	false	Read
100004	false	Read
100005	false	Read

Figura 48. Entrada activada



Figura 49. Entrada I0.0 desactivada, salidas Q0.3 y Q0.4 activadas

Address	Value	Access
000001	false	Read/Write
000002	false	Read/Write
000003	false	Read/Write
000004	true	Read/Write
000005	true	Read/Write
100001	false	Read
100002	false	Read
100003	false	Read
100004	false	Read
100005	false	Read

Figura 50. Entrada desactivada

1.8.3.4 Acceder a variables de varios PLC,s:

Una vez superado el paso de poder acceder al valor de las variables del PLC desde lavbiew, el siguiente paso es poder controlar dos PLC´s a la vez que estén en red. Para ello dentro del mismo proyecto se crea otro servidor I/O Server con la dirección del nuevo PLC. Es necesario que en el nuevo PLC se cargue también el programa de servidor que se ha cargado en el anterior PLC. Aprovechando se carga también el mismo programa que en el PLC anterior, es decir, si la entrada I0.0 (100001) está desactivada la salida Q0.3 (000004) estará activada y viceversa. El proyecto adquiere la siguiente forma:

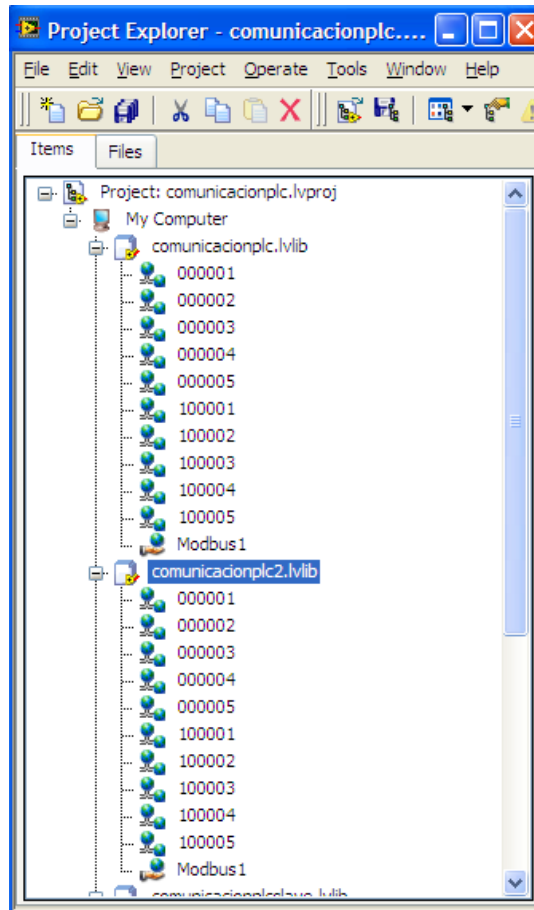


Figura 51. Dos PLC´s dentro de un proyecto

Una vez cargado el programa se observan las variables en el Distributed Manager:

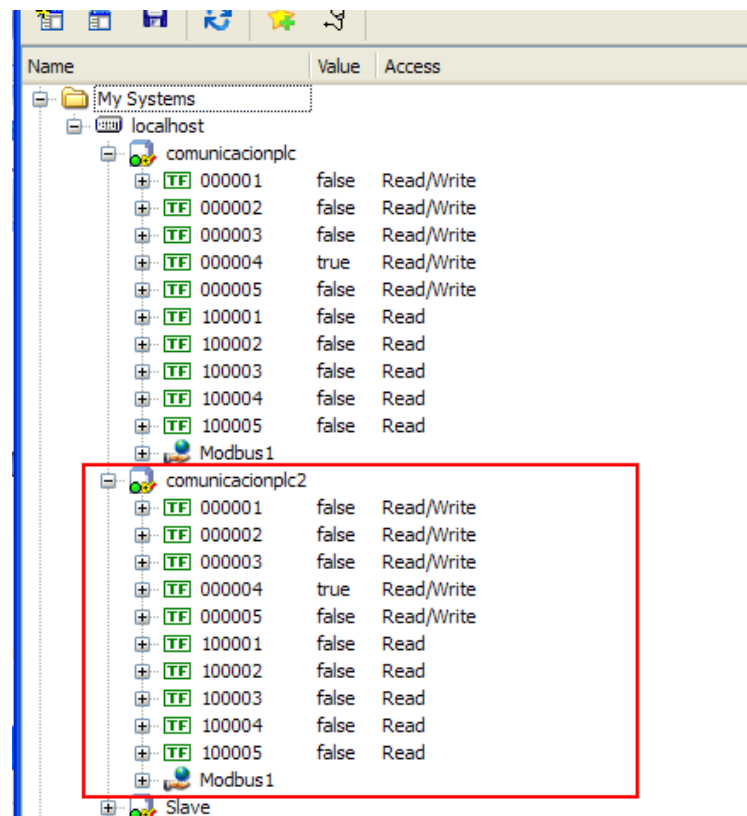


Figura 52. Variables PLC 2

Se observa que la entrada está desactivada y la salida activa. Lo mismo para el PLC1. A continuación se activa la entrada del PLC2 I0.0 (100001) con lo que la salida Q0.3 (000004) se desactivará. El PLC1 se mantiene igual:

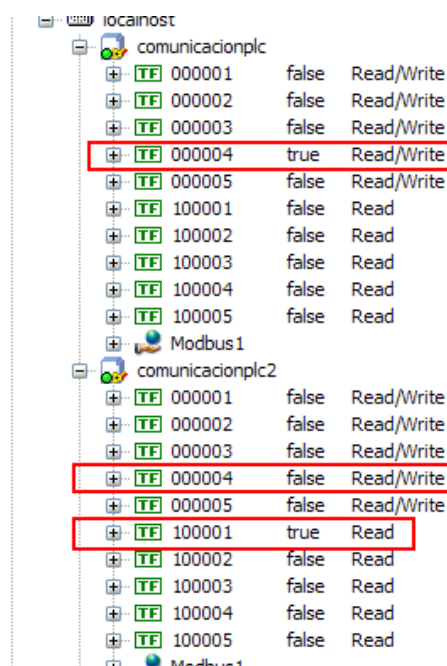


Figura 53. Entrada PLC2 activada, salida desactivada

Para observar que varían ambos, ahora se mantiene el PLC2 como está y se activa la entrada I0.0 (100001) del PLC1, por lo que la salida Q0.3 (000004) se desactivará:

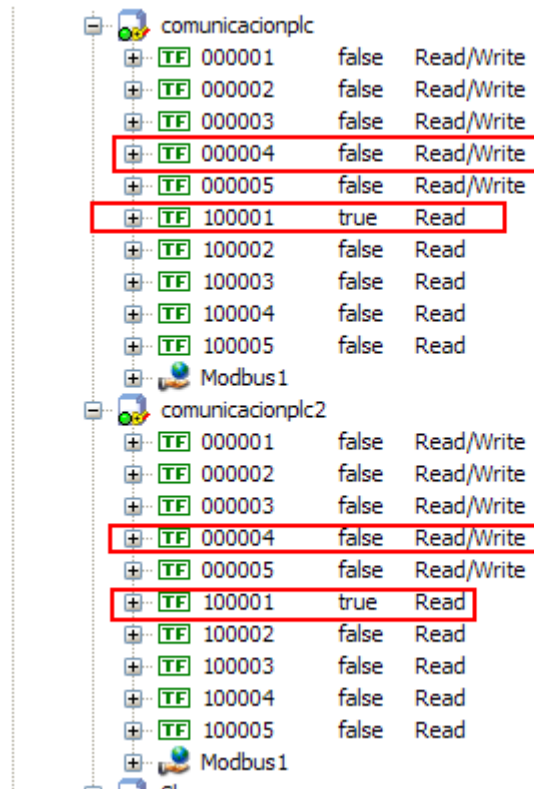


Figura 54. Entrada PLC1 activada, salida desactivada

El siguiente paso consiste en que al activar la entrada I0.0 (100001) en el PLC, esto se refleje en labview con un led encendido. Se muestra la situación inicial:

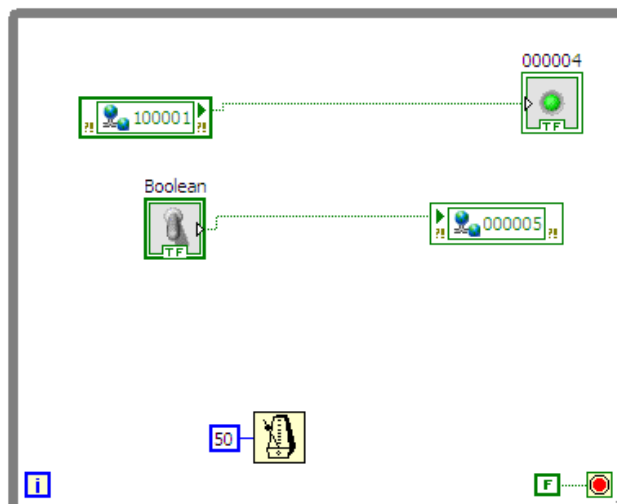


Figura 55. Panel Frontal de Labview donde se programa

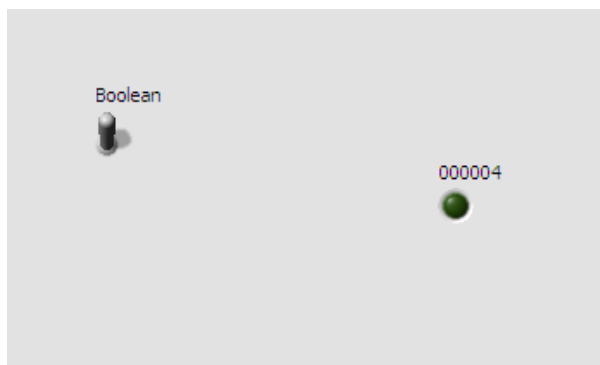


Figura 56. Panel Frontal de Labview donde se observan los cambios

TF	000003	false	Read/Write
TF	000004	true	Read/Write
TF	000005	false	Read/Write
TF	100001	false	Read
TF	100002	false	Read
TF	100003	false	Read
TF	100004	false	Read
TF	100005	false	Read
	Modbus 1		

Figura 57. Distributed Manager donde queda reflejada la situación inicial

Ahora se procede a alimentar la entrada del PLC IO.0 (100001) con un conductor a 24V:

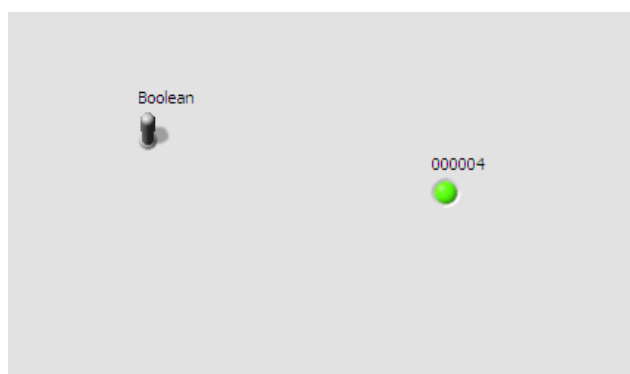


Figura 58. Led activado debido a la alimentación de la entrada del PLC

TF	000002	false	Read/Write
TF	000003	false	Read/Write
TF	000004	false	Read/Write
TF	000005	false	Read/Write
TF	100001	true	Read
TF	100002	false	Read
TF	100003	false	Read
TF	100004	false	Read
TF	100005	false	Read
	Modbus 1		
	comunicacione?		

Figura 59. Entrada PLC activada

Por otra parte se ha probado a utilizar un interruptor en labview que active una salida del PLC2, en este caso la salida Q0.4 (000005). Se muestra la situación inicial:



Figura 60. Interruptor Labview conectado a la salida del PLC2

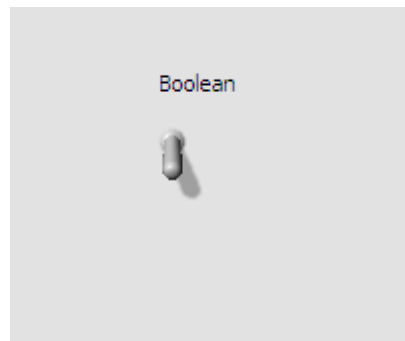


Figura 61. Interruptor desactivado

TF	Address	State	Access
TF	000001	false	Read/Write
TF	000002	false	Read/Write
TF	000003	false	Read/Write
TF	000004	false	Read/Write
TF	000005	false	Read/Write
TF	100001	true	Read
TF	100002	false	Read
TF	100003	false	Read
TF	100004	false	Read
TF	100005	false	Read

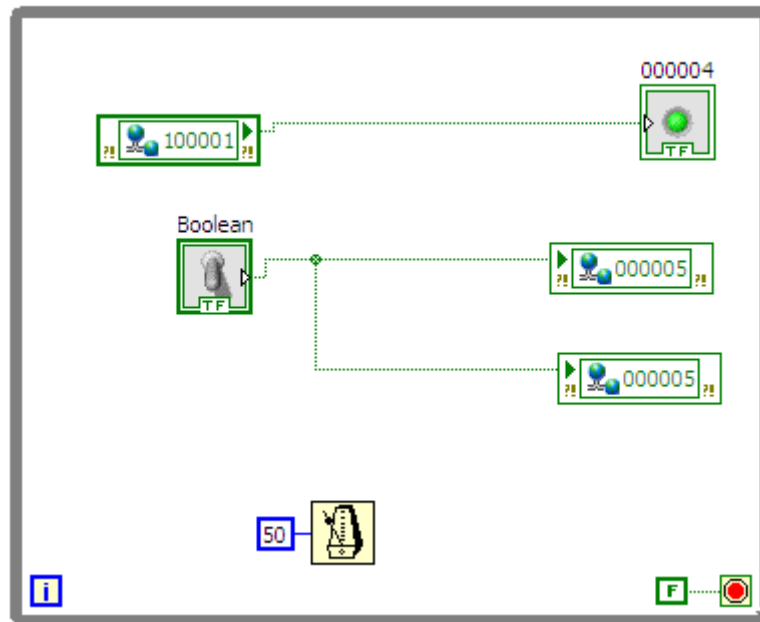
Figura 62. Estado Inicial

Se activa el interruptor para observar el cambio de estado de la salida Q0.4 (000005):

TF	Address	State	Access
TF	000001	false	Read/Write
TF	000002	false	Read/Write
TF	000003	false	Read/Write
TF	000004	false	Read/Write
TF	000005	true	Read/Write
TF	100001	true	Read
TF	100002	false	Read
TF	100003	false	Read
TF	100004	false	Read
TF	100005	false	Read

Figura 63. Interruptor activado. Salida Q0.4 (000005) activada

Esto ocurre para los dos PLC's. Ahora con el mismo interruptor se activan las salidas Q0.4 (000005) de los dos PLC's a la vez:



System	Address	Value	Access
comunicacionplc	000001	false	Read/Write
	000002	false	Read/Write
	000003	false	Read/Write
	000004	false	Read/Write
	000005	true	Read/Write
	100001	true	Read
	100002	false	Read
	100003	false	Read
	100004	false	Read
	100005	false	Read
comunicacionplc2	000001	false	Read/Write
	000002	false	Read/Write
	000003	false	Read/Write
	000004	false	Read/Write
	000005	true	Read/Write
	100001	true	Read
	100002	false	Read
	100003	false	Read
	100004	false	Read
	100005	false	Read

Figura 64. Salida Q0.4(000005) de PLC1 y PLC2 activada

La siguiente prueba es activar una salida del PLC2 con una entrada del PLC1. Si la entrada I0.0 (100001) del PLC1 está activada, la salida Q0.4 (000005) del PLC2 se activará:

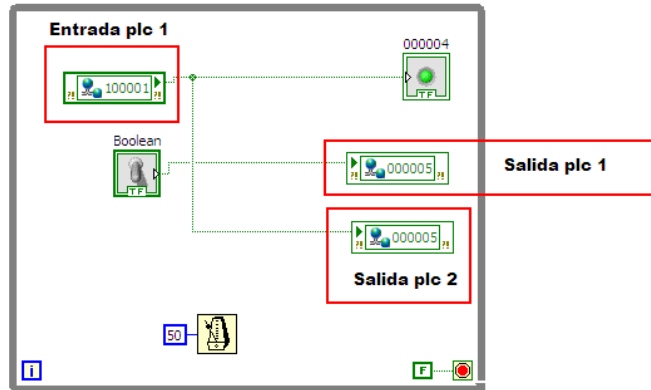


Figura 65. Panel de Bloques de Labview

La entrada del PLC1 está desactivada, con lo que la salida del PLC2 también:

System	Address	Value	Access
comunicacionplc	000001	false	Read/Write
	000002	false	Read/Write
	000003	false	Read/Write
	000004	true	Read/Write
	000005	false	Read/Write
	100001	false	Read
	100002	false	Read
	100003	false	Read
	100004	false	Read
	100005	false	Read
	Modbus 1		
comunicacionplc2	000001	false	Read/Write
	000002	false	Read/Write
	000003	false	Read/Write
	000004	false	Read/Write
	000005	false	Read/Write
	100001	true	Read
	100002	false	Read
	100003	false	Read
	100004	false	Read
	100005	false	Read
	Modbus 1		

Figura 66. Entrada PLC1 desactivada, salida PLC2 desactivada

Se activa la entrada del PLC1, y con ello se activará la salida del PLC2:

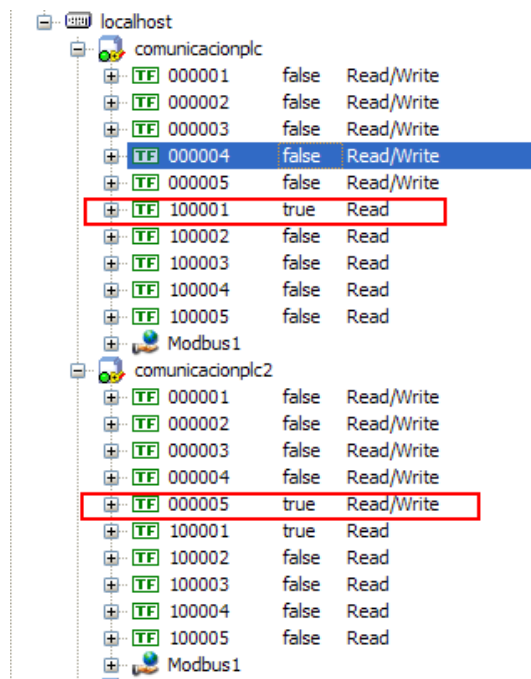


Figura 67. Entrada PLC1 activada, salida PLC2 activada

1.8.4 Conexión placa PICDEM Mechatronics Demonstration Board

Una vez se ha realizado la parte de comunicación del PC con el PLC, se procede al control de un motor situado en la placa de PICDEM. Para ello es necesario seguir ciertos pasos:

1. Alimentación placa
2. Control motor con encoder
3. Circuito amplificador encoder
4. Leer variable encoder
5. Variación de la velocidad del motor

1.8.4.1 Alimentación placa:

La placa de demostración está alimentada con un adaptador de red de 230V/9V. A su vez, mediante un regulador, se convierten los 9V a 5V, que es la tensión con la que trabaja el microcontrolador de la placa. Como se va a controlar solo el motor y el encoder de la tarjeta, se prescinde del microcontrolador y se alimenta el motor exteriormente, es decir, se alimenta el motor con una señal de 5V desde el PLC. Para ello se va a explicar el procedimiento en varios pasos:

1. Configuración CPU
2. Programación bloque señal de impulsos
3. Circuito de adaptación de la señal del PLC a la placa

1. Configuración CPU:

La señal que procede del PLC es una señal de impulsos con forma cuadrada, ya que el motor es de tipo paso a paso. El PLC es capaz de generar una señal cuadrada de 24V, por lo que es necesario realizar un circuito de adaptación para reducir la tensión. Para ello se monta un circuito divisor para que entre los bornes deseados se obtenga 5V. Para que el PLC dé una señal cuadrada se han de realizar cambios en la configuración del dispositivo, y cargar un programa con un bloque que indique que esa señal es cuadrada.

Se procede al cambio de configuración. Primeramente se han de cambiar varias opciones en la configuración de las propiedades de la CPU del PLC:

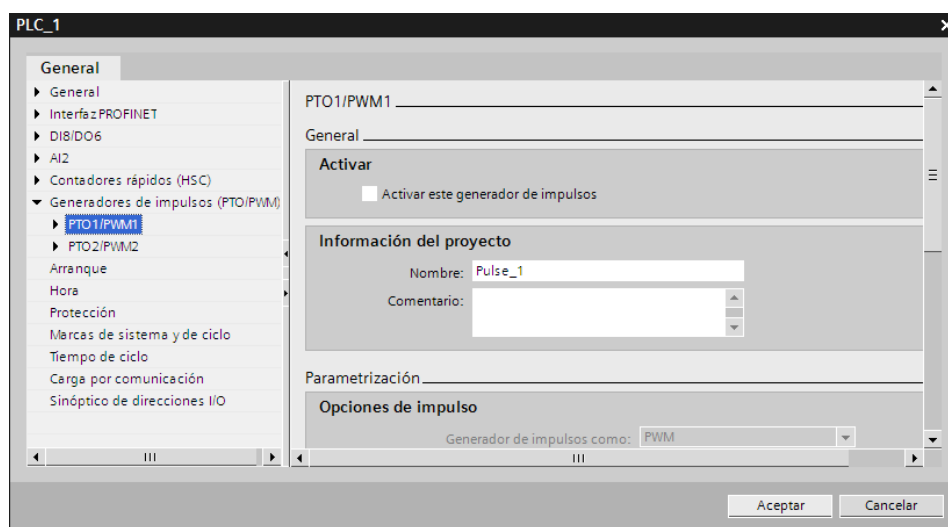


Figura 68. Panel de Configuración de la CPU

Se ha de activar la opción de “Activar generador de Impulso”

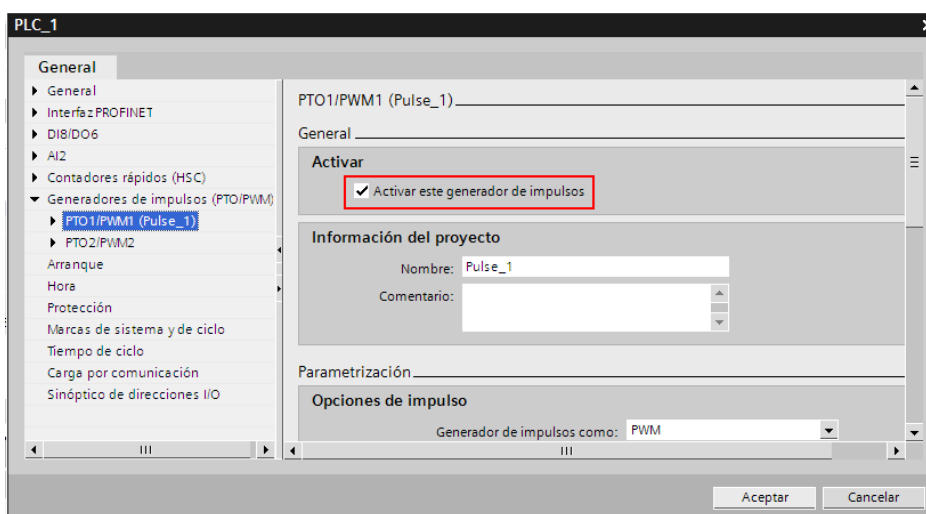


Figura 69. Activar el generador de impulsos

Si se desea se pueden cambiar los datos por defecto que da el PLC, como la duración del impulso, su amplitud...etc

La duración de impulso puede expresarse en centésimos del tiempo de ciclo (0 – 100), milésimos (0 – 1000), diezmilésimos (0 – 10000) o formato analógico S7. La duración de impulso puede variar entre 0 (sin impulso, siempre off) y escala completa (sin impulso, siempre on).

En este caso se los valores requeridos son de unos 40kHz, y el ciclo de trabajo será entre el 0 y el 100%:

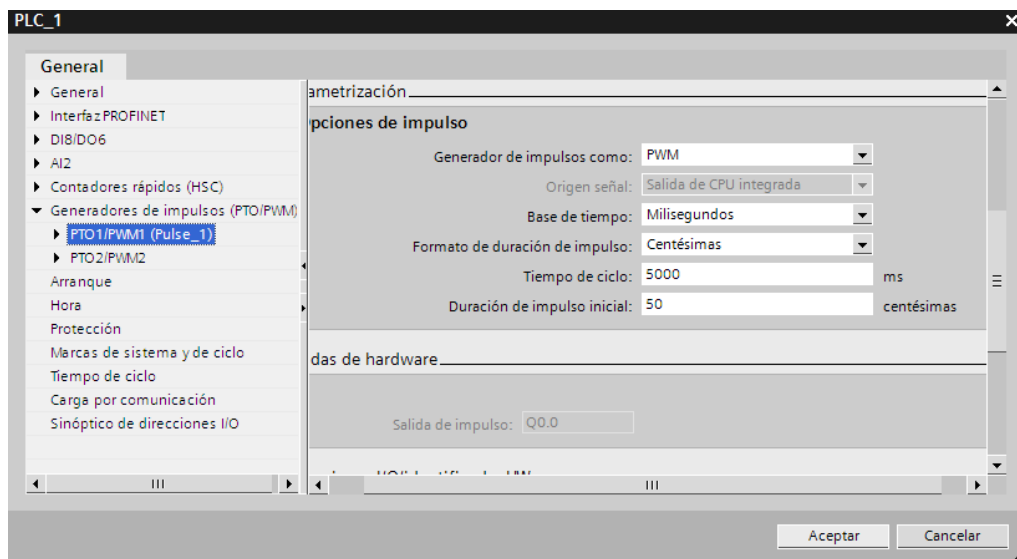


Figura 70. Valores de la señal de impulso

2. Programación bloque señal de impulsos

Una vez se han realizado los cambios se procede a la configuración del bloque que ordena la señal de impulsos. La instrucción CTRL_PWM (Controlar modulación del ancho de pulso) ofrece un tiempo de ciclo fijo con un ciclo de trabajo variable. La salida PWM se ejecuta continuamente tras haberse iniciado a la frecuencia indicada (tiempo de ciclo). La duración de impulso varía según sea necesario para obtener el control deseado. La función es la CTRL_PWM_DB que tiene la siguiente apariencia:

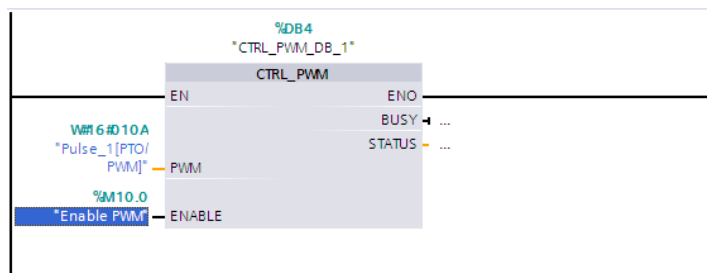


Figura 71. Bloque de Función CTRL_PWM_DB

La instrucción "CTRL_PWM" permite activar o desactivar por software un generador de impulsos soportado por la CPU.

Un generador de impulsos se parametriza exclusivamente en la configuración de dispositivos y no mediante la instrucción "CTRL_PWM", como se ha citado anteriormente. Por ello, sólo es posible modificar los parámetros que afectan a la CPU cuando ésta está en estado operativo STOP. Para poder ejecutar la instrucción correctamente es preciso que el generador de impulsos especificado esté habilitado en la configuración de hardware.

En la entrada PWM sólo es posible indicar variables del tipo de datos HW_PWM. El tipo de datos de hardware HW_PWM tiene una longitud de una WORD (palabra).

El generador de impulsos se activa cuando está activado el bit en la entrada ENABLE de la instrucción. Si ENABLE tiene el valor TRUE, el generador genera impulsos que tienen las propiedades definidas en la configuración de dispositivos. Si se desactiva el bit en la entrada ENABLE o la CPU pasa a STOP, se desactiva el generador de impulsos y ya no se generan impulsos. La instrucción "CTRL_PWM" se ejecuta únicamente si la entrada EN tiene el estado lógico "1".

Las entradas y salidas digitales que se usan para PWM y PTO no se pueden forzar. Las entradas y salidas digitales asignadas mediante la configuración de dispositivos no se pueden controlar con la tabla de forzado ni con la tabla de observación.

La tabla siguiente muestra los parámetros de la instrucción "CTRL_PWM":

Parámetros	Declaración	Tipo de datos	Área de memoria	Descripción
PWM	Input	HW_PWM	I, Q, M, L o constante	Identificación de hardware del generador de impulsos
ENABLE	Input	BOOL	I, Q, M, D, L o constante	El generador de impulsos se activa con ENABLE = TRUE y se desactiva con ENABLE = FALSE.
BUSY	Output	BOOL	I, Q, M, D, L	Estado de ejecución
STATUS	Output	WORD	I, Q, M, D, L	Estado de la instrucción

Figura 72. Parámetros CTRL_PWM

La marca M10.0 es la encargada de habilitar o deshabilitar el bloque de función. Una vez el programa se ha cargado en el PLC, se observa el programa en modo online y se fuerza la entrada “Enable PWM” a 1 para activar la salida de impulsos. El PLC por defecto asigna una salida para dicha señal, que es la Q0.0, sin posibilidad de modificarla.

Descripción	Asignación de salidas predeterminada		
		Impulso	Sentido
PTO 1	Integrada en la CPU	Q0.0	Q0.1
	Signal Board	Q4.0	Q4.1
PWM 1	Integrada en la CPU	Q0.0	--
	Signal Board	Q4.0	--
PTO 2	Integrada en la CPU	Q0.2	Q0.3
	Signal Board	Q4.2	Q4.3
PWM 2	Integrada en la CPU	Q0.2	--
	Signal Board	Q4.2	--

Figura 73. Asignación de salidas en el PLC

La instrucción CTRL_PWM utiliza un bloque de datos (DB) para almacenar la información de parámetros. Cuando se inserta una instrucción CTRL_PWM en el editor de programación, se asigna un DB. El usuario no modifica por separado los parámetros del bloque de datos, sino que la instrucción CTRL_PWM los controla.

A continuación se muestra como la entrada del bloque de función “Enable PWM” está a false, con lo que la señal de impulsos se encuentra desactivada.

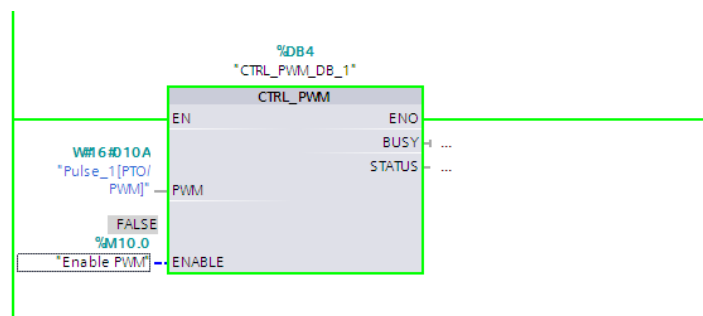


Figura 74. Visión online, señal de impulso desactivada

Se procede a activar la señal de impulsos:

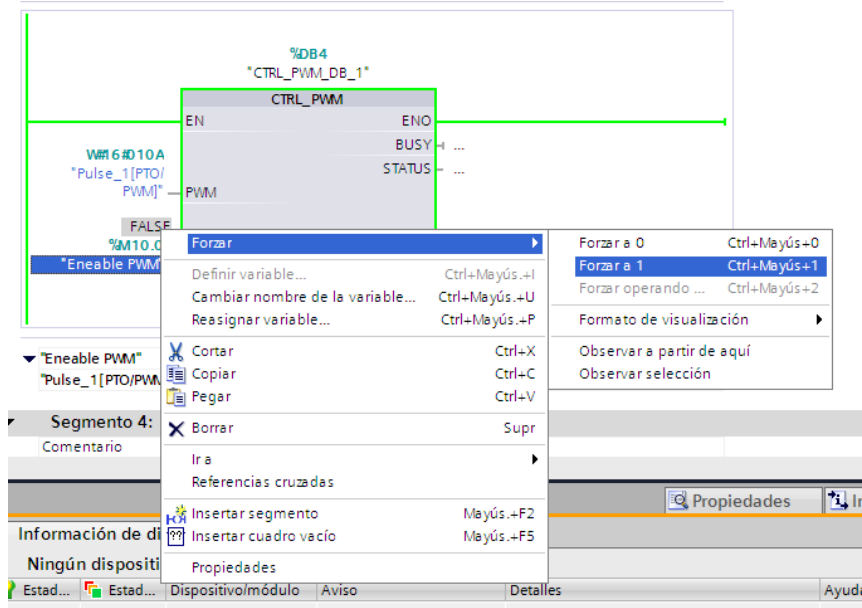


Figura 75. Forzado de la entrada de habilitación

Una vez se ha forzado dicho valor a 1 la señal de impulsos se refleja en el PLC parpadeando la luz de la salida Q0.0. Esta variable no es posible de observar en el Distributed Manager de Labview.

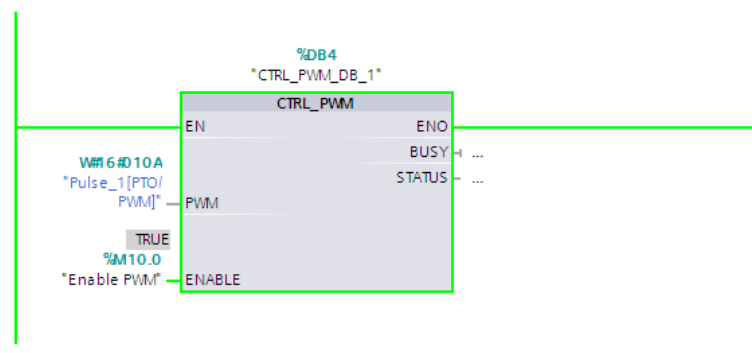


Figura 76. Señal de impulsos activada

Existe la posibilidad de habilitar la señal de impulsos desde labview. Esto se consigue activando una salida desde labview, y que esta salida a su vez active la señal de impulsos:

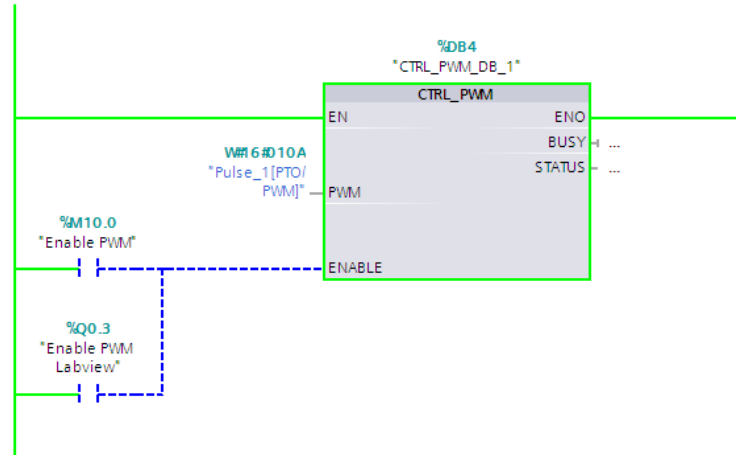


Figura 77. Control señal de impulsos desde Labview

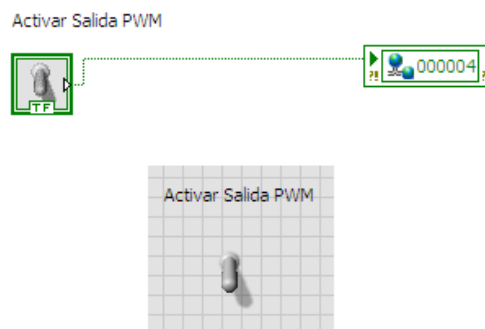


Figura 78. Diagrama de bloques de Labview

comunicacionplc			
TF	000001	false	Read/Write
TF	000002	true	Read/Write
TF	000003	false	Read/Write
TF	000004	false	Read/Write
TF	000005	false	Read/Write
TF	000006	false	Read/Write

Figura 79. Acceso a la variable desde Distributed Manager

Una vez se ha programado correctamente se procede a activar la salida de impulsos:

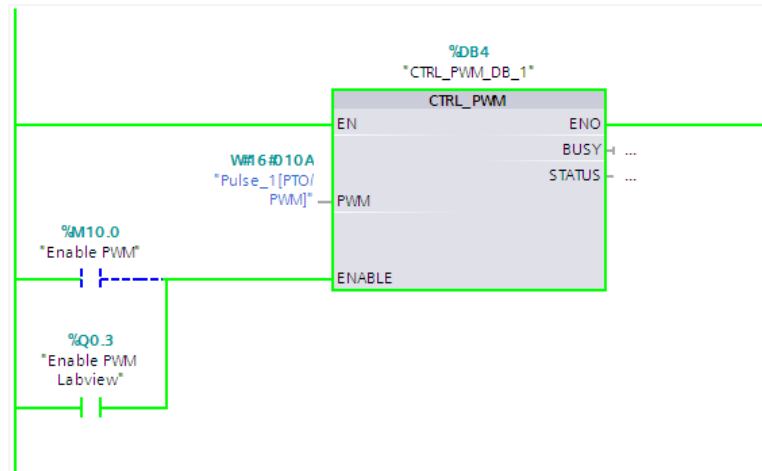


Figura 80. Activar salida de impulsos desde Labview



Figura 81. Pulsador que activa la señal de impulsos desde Labview

comunicacionplc			
TF	000001	false	Read/Write
TF	000002	true	Read/Write
TF	000003	false	Read/Write
TF	000004	true	Read/Write
TF	000005	false	Read/Write
TF	000006	false	Read/Write

Figura 82. Salida activada desde Labview

3. Circuito de adaptación de la señal del PLC a la placa

Una vez se ha conseguido la señal de impulsos en el PLC es necesario realizar cálculos de resistencias para obtener la tensión que se desea para el motor. Se llevan a cabo unos cálculos sencillos, y el divisor queda de la siguiente forma:

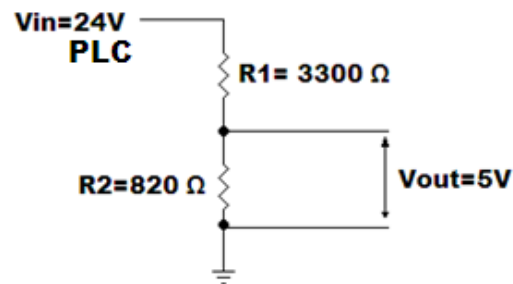


Figura 83.Divisor de tensión

Se muestra a continuación una imagen en la cual aparecen las partes más importantes de la placa, para así poder ubicar los elementos.

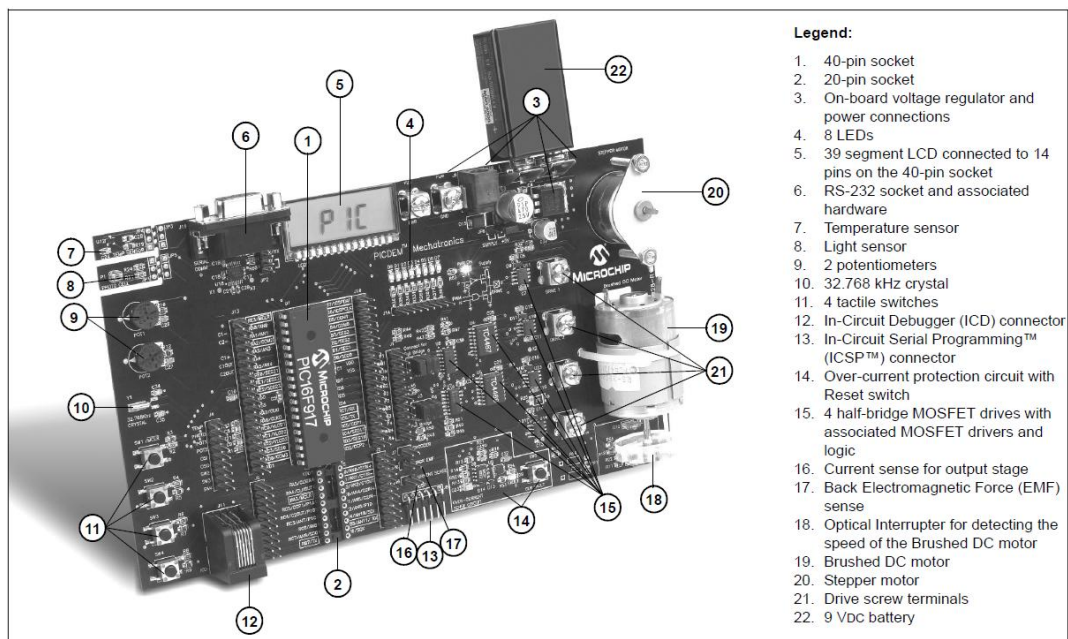


Figura 84.Partes de las que consta la placa

Una vez se ha obtenido la señal de 5V se procede a alimentar el motor. El circuito integrado en la placa que alimenta al motor es el siguiente. En dicho circuito se observa que el motor está conectado mediante el microcontrolador:

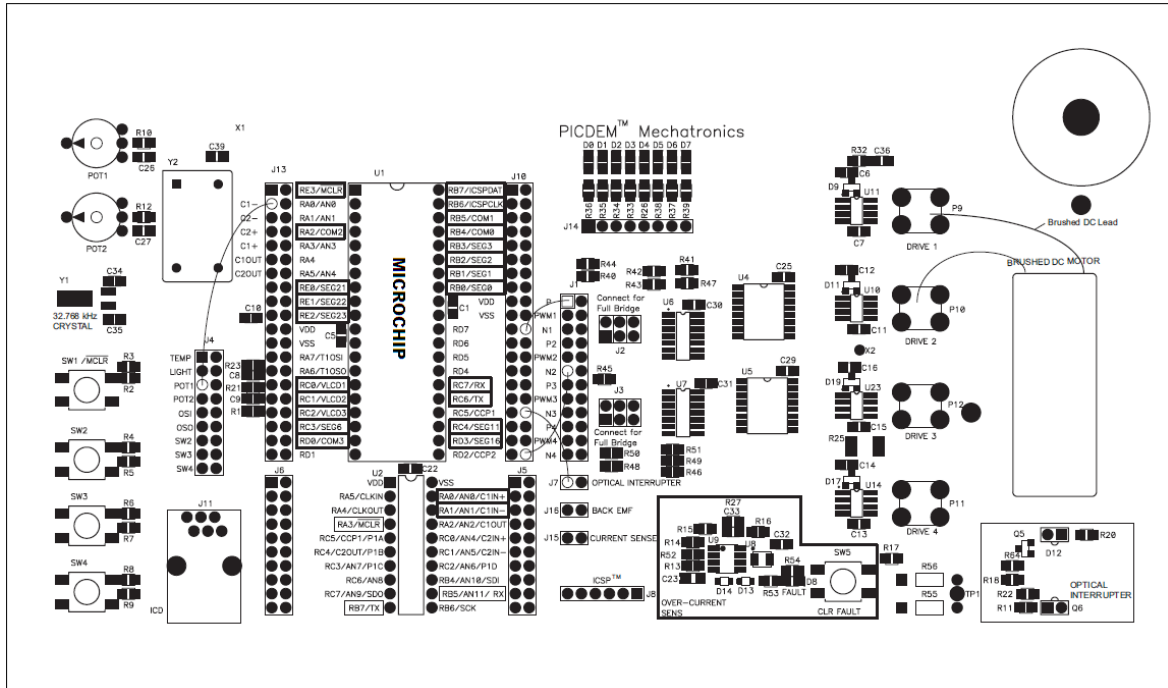


Figura 85. Circuito Placa

El motor se encuentra alimentado por un puente en H:

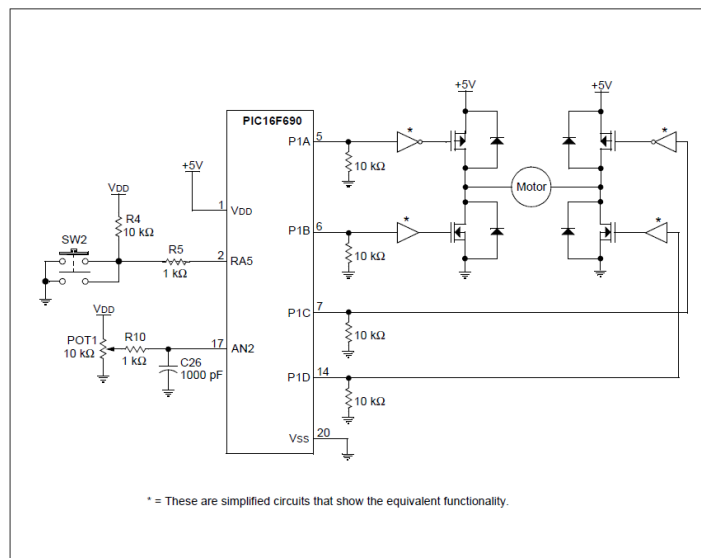


Figura 86. Circuito interno de conexión del motor

Se cambia la alimentación de puente en H, es decir, en vez de ser alimentado por el microchip se alimenta mediante el divisor de tensión que se ha creado:

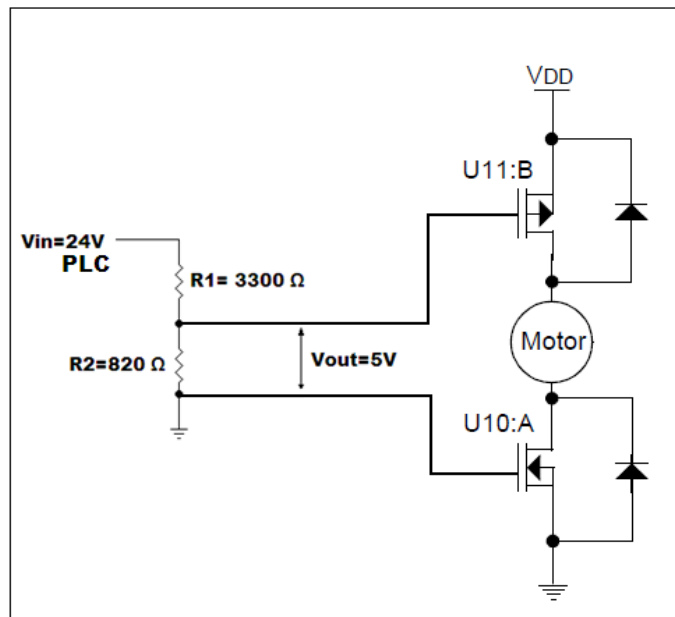


Figura 87. Motor alimentado con el divisor

1.8.4.2 Control motor mediante encoder:

Para monitorizar la velocidad del motor se emplea un encoder óptico que se encuentra integrado en la propia placa. Para ello es necesario conocer la conexión del Interruptor Óptico (J7):

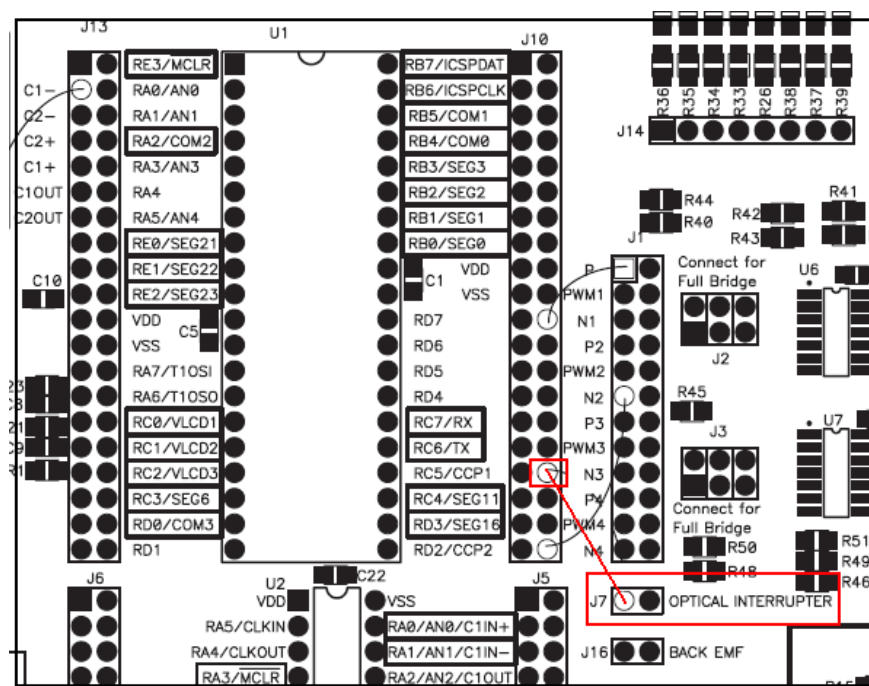


Figura 88. Conexión Interruptor Óptico

1.8.4.3 Circuito amplificador:

La señal que produce el encoder se ha de conectar al PLC para poder obtener los datos necesarios. A su vez dicha señal será accesible desde labview. El problema que se presenta es que la señal que produce el encoder es cuadrada de 5V y las entradas digitales del PLC funcionan a 24V, por lo que es necesario diseñar un circuito amplificador que pase la señal de 5V a 24V.

Este circuito se puede realizar de varias formas, pero para este proyecto se ha pensado en un circuito amplificador con un transistor. El transistor seleccionado se ha de ajustar a las necesidades que presenta el proyecto. Para este proyecto se ha utilizado el transistor BC548B, tipo NPN. El transistor tiene la siguiente forma:

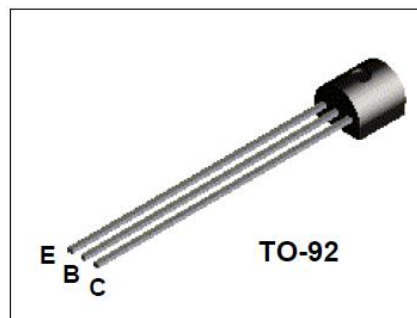


Figura 89. Transistor BC548B, NPN

Se realiza el diseño del circuito con sus correspondientes cálculos. El transistor va a funcionar como un interruptor; es decir, variando entre los estados de corte o de saturación. Cuando el transistor se encuentre en corte, modo OFF, habrá entre los bornes del emisor y la base, $V_{CE}=24V$. Cuando el transistor se encuentre en saturación, modo ON, entre los bornes del emisor y la base, $V_{CE}=0V$

El circuito diseñado tiene la siguiente forma:

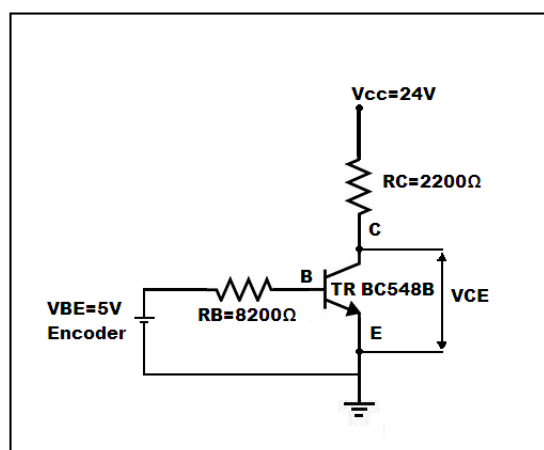


Figura 90. Circuito Amplificador

Tras realizar el montaje se observa que la señal del encoder no es suficiente, por lo que es necesario diseñar un nuevo circuito con un amplificador operacional en modo buffer, o también llamado seguidor de tensión. Para ello se utiliza el amplificador LM741 que se muestra a continuación:

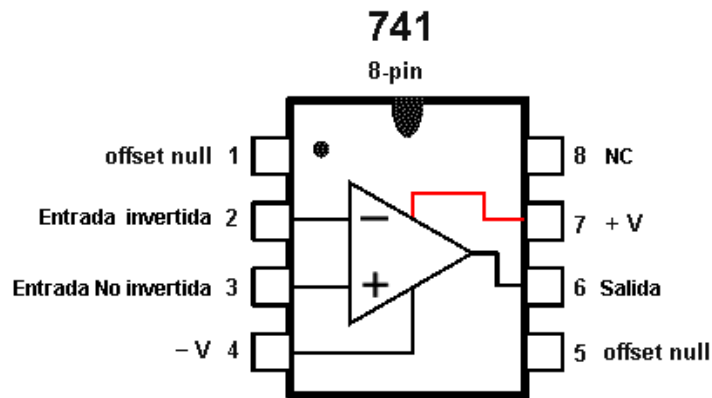


Figura 91. Amplificador Operacional μA 741

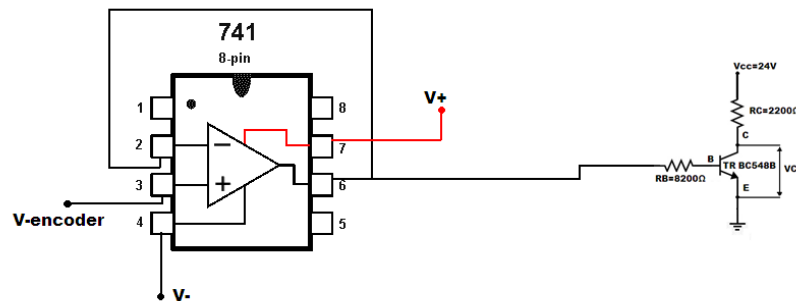


Figura 92. Conexión de circuito seguidor de tensión y circuito amplificador

1.8.4.4 Leer variable encoder:

Una vez se ha amplificado la señal del Interruptor Óptico, se procede a conectar la señal a la entrada del PLC. Al ser una señal cuadrada se puede leer como una señal booleana, y así se puede utilizar las entradas del PLC sin tener que modificar el tipo de variable.

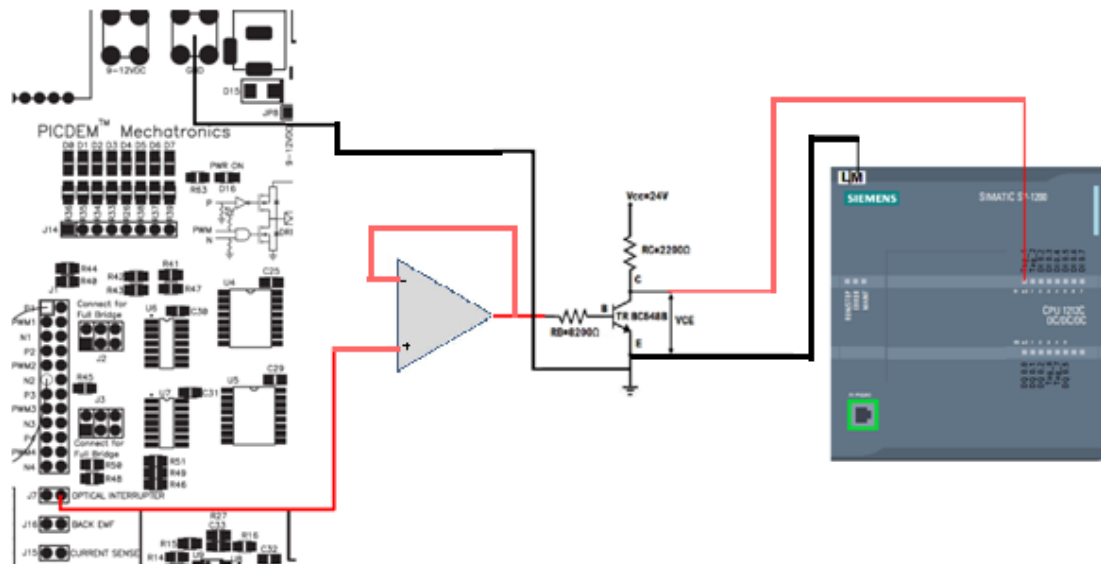


Figura 93. Encoder conectado al PLC

Al introducir una señal cuadrada al motor paso a paso la velocidad ha de variar. Esto se refleja en la señal que el encoder produce, que también es cuadrada debido a que la rueda tiene dos ranuras. Cuando pasa por una ranura llega señal al interruptor óptico. A la entrada del PLC llegará una señal de impulsos, por lo que es necesario crear en el programa del PLC un segmento que cuente el número de impulsos para poder saber a qué velocidad gira el motor.

Es necesario utilizar un contador rápido ya que la señal del encoder es más rápida que la ejecución del OB principal. El contador correspondiente se llama mediante la función CTRL_HSC.

Se utilizan los contadores rápidos (HSC) para el conteo de eventos que se producen con mayor rapidez que la frecuencia de ejecución del OB. La instrucción CTRL_HSC controla el funcionamiento del HSC.

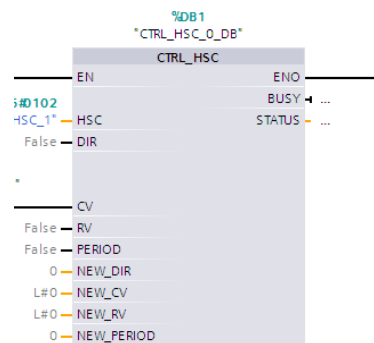


Figura 94. Instrucción Contador Rápido (HSC)

Se han de configurar los parámetros de cada HSC en la configuración de dispositivos de la CPU: modo de contaje, conexiones de E/S, asignación de alarmas y funcionamiento como contador rápido o dispositivo para medir la frecuencia de pulsos.

Algunos de los parámetros del HSC se pueden modificar mediante el programa de usuario para ofrecer un control de programa del proceso de contaje:

- Ajustar el sentido de contaje al valor NEW_DIR
- Ajustar el valor de contaje actual al valor NEW_CV
- Ajustar el valor de referencia al valor NEW_RV
- Ajustar el periodo (para el modo de medición de frecuencia) al valor NEW_PERIOD

Si los siguientes valores booleanos están definidos como 1 cuando se ejecuta la instrucción CTRL_HSC, el valor NEW_xxx correspondiente se carga en el contador. Las peticiones múltiples (varias marcas definidas simultáneamente) se procesan en una sola ejecución de la instrucción CTRL_HSC.

- Al definir DIR = 1, se carga un valor NEW_DIR.
- Al definir CV = 1, se carga un valor NEW_CV.
- Al definir RV = 1, se carga un valor NEW_RV.
- Al definir PERIOD = 1, se carga un valor NEW_PERIOD.

El funcionamiento del contador rápido es el siguiente:

El contador rápido (HSC) realiza el conteo de eventos que se producen con mayor rapidez que la frecuencia de ejecución del OB. Si los eventos que se deben contar se producen con la frecuencia de ejecución del OB o menor que esta, se han de utilizar las instrucciones de conteo CTU, CTD o CTUD. La instrucción CTRL_HSC permite al programa de usuario cambiar algunos de los parámetros del HSC.

Todos los HSCs funcionan de la misma manera en el mismo modo de operación del contador. Hay cuatro tipos básicos de HSC, de los cuales se elige el siguiente:

- Contador de fase simple con control externo del sentido de conteo
- Modos y entradas de contador: La tabla siguiente muestra las entradas utilizadas para las funciones de reloj, control de sentido e inicialización asociadas al HSC. Una misma entrada no se puede utilizar para dos funciones diferentes. Sin embargo, cualquier entrada que no se esté utilizando en el modo actual del HSC se puede usar para otro fin. Por ejemplo, si el HSC1 está en un modo que utiliza entradas integradas, pero que no usa la inicialización externa (I0.3), la entrada I0.3 puede utilizarse para alarmas de flanco o para el HSC2
- Para fase simple: C es la entrada de reloj, [d] es la entrada de sentido opcional y [R] es una entrada de inicialización externa opcional. (La inicialización sólo está disponible para el modo de "conteo").

HSC		Entrada integrada de CPU (0.x)							Entrada de SB (4.x) ³				
		0	1	2	3	4	5	6	7	0	1	2	3
HSC 1 ¹	1 fase	C	[d]		[R]					C	[d]		[R]
	2 fases	CU	CD		[R]					CU	CD		[R]
	Fases AB	A	B		[R]					A	B		[R]
HSC 2 ¹	1 fase		[R]	C	[d]						[R]	C	[d]
	2 fases		[R]	CU	CD						[R]	CU	CD
	Fases AB		[R]	A	B						[R]	A	B
HSC 3	1 fase					C	[d]		[R]				
	2 fases					CU	CD		[R]				
	Fases AB					A	B		[R]				
HSC 4	1 fase						[R]	C	[d]				
	2 fases						[R]	CU	CD				
	Fases AB						[R]	A	B				
HSC 5 ²	1 fase									C	[d]		[R]
	2 fases									CU	CD		[R]
	Fases AB									A	B		[R]
HSC 6 ²	1 fase										[R]	C	[d]
	2 fases										[R]	CU	CD
	Fases AB										[R]	A	B

¹ HSC 1 y HSC 2 se pueden configurar tanto para las entradas integradas como para un SB.

² HSC 5 y HSC 6 sólo están disponibles con un SB. HSC 6 sólo está disponible con un SB de cuatro entradas.

³ Un SB con sólo dos entradas únicamente ofrece las entradas 4.0 y 4.1.

Como en este proyecto el contador se programa como un contador de fase simple, es necesario cablear la entrada I0.1 para indicar que el sentido de conteo es ascendente. Y en la entrada I0.0 se conecta la señal del encoder.

Una vez se ha programado el contador, los datos que se obtienen se almacenan en la variable ID1000, a la cual no se tiene acceso, por lo que se mueven los datos a otra variable para poder acceder a ellos, en este caso MD0.

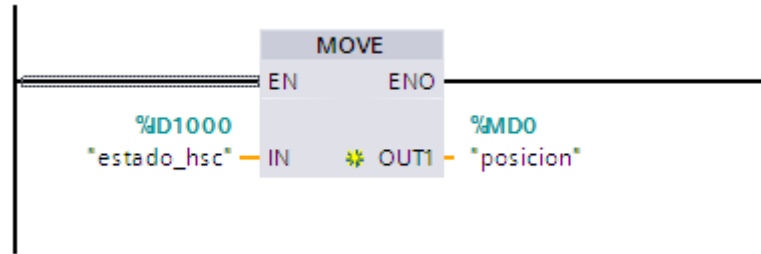


Figura 95.Instrucción MOVE

El dato que se obtiene en esta variable es del tipo DInt, es decir, una variable de 32 bits. Dado que no se puede acceder a ese tipo de datos se utiliza la variable de tipo Int 16 bits. Esto se realiza con una instrucción de conversión:

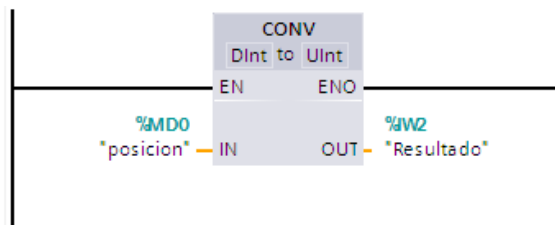


Figura 96.Instrucción Conversión

La variable final se almacena en IW2, a la cual se tiene acceso desde Labview y se puede monitorizar:

+	TF	100005	false	Read
+	I	300001	768	Read
+	I	300002	0	Read
+	I	300003	0	Read
+	I	300004	0	Read
+	I	300005	600	Read
+	I	300006	0	Read
+	I	300010	0	Read

Figura 97.Variable IW2 visualizada desde Labview

El siguiente paso es controlar los impulsos y contabilizarlos. Esto se programa en dos partes, un temporizador que cuente el tiempo que se tarda en contar los impulsos y un comparador que determine el número de impulsos que se van a contar. El programa queda de la siguiente manera:

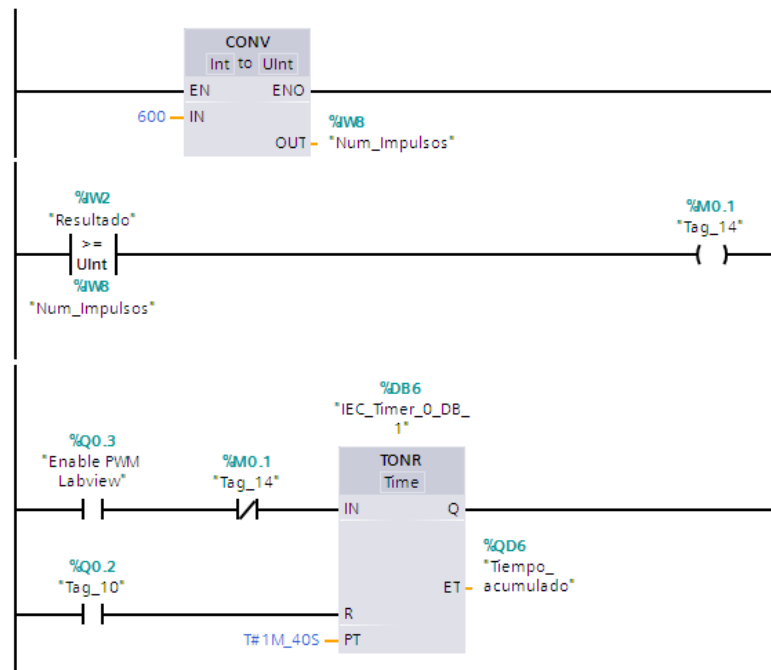


Figura 97. Programa control velocidad del motor

La primera instrucción convierte un número constante, en este caso el número de impulsos que se han de contar, en una variable de tipo Int para poder acceder a ella desde Labview. La variable IW8 es accesible y se puede monitorizar desde labview:

+	I	300001	768	Read
+	I	300002	0	Read
+	I	300003	0	Read
+	I	300004	0	Read
+	I	300005	600	Read
+	I	300006	0	Read
+	I	300010	0	Read

Figura 98. Lectura de la variable IW8 desde Labview

La segunda instrucción compara el número de impulsos que está contando el contador rápido con el número de impulsos que se desean contar. Este comparador activa la marca M0.1 que va a ser una condición para activar el temporizador.

El temporizador está programado de tal forma que cuando se active la señal de impulsos y la M0.1 no esté activada comience a contar el tiempo que transcurre. En el momento que el número de impulsos que cuenta el contador es igual al número de impulsos que se desean contar, la M0.1 se activa y se desactiva el temporizador. El tiempo acumulado se puede visualizar en la variable QD6. Como ya se ha citado anteriormente, esta variable no es accesible desde Labview, por lo que se repite el proceso de conversión para poder acceder a este valor desde Labview:

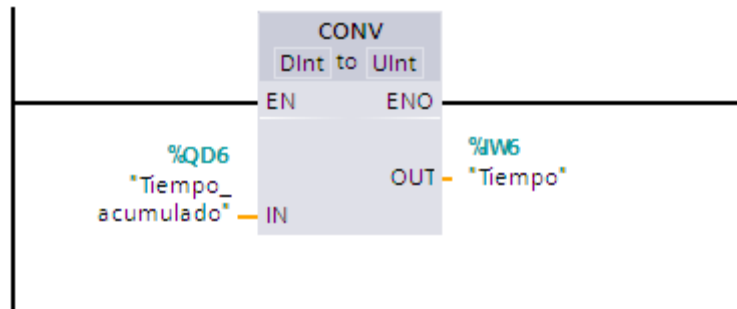


Figura 99. Conversión del dato tiempo acumulado

El tiempo acumulado se almacena en la variable IW6, la cual se monitoriza en Labview como se muestra a continuación:

+	I	100000	raise	Read
+	I	300001	768	Read
+	I	300002	0	Read
+	I	300003	0	Read
+	I	300004	0	Read
+	I	300005	600	Read
+	I	300006	0	Read
+	I	300010	0	Read

Figura 100. Lectura de la variable IW6 desde Labview

1.8.4.5 Variación de la velocidad del motor:

La velocidad del motor depende de la señal con la que se alimente. Para modificar la velocidad basta con variar los parámetros de la señal de impulsos citados en anteriormente en la parte de configuración de la CPU. Los parámetros son tales como la duración del impulso, es decir, el periodo que sigue la señal. La velocidad del motor está relacionada con la frecuencia, por lo que variando la frecuencia varía el periodo.

La aplicación que se va a desarrollar en este proyecto necesita una frecuencia de unos 40kHz, lo que implica que el tiempo de ciclo sea de 25 microsegundos. Variando el ciclo de trabajo se podrá variar la velocidad del motor, la cual depende de la duración del impulso.

Se realizan pruebas para ver que el circuito funciona correctamente y que la velocidad del motor varía según el ciclo de trabajo que se configure. Efectivamente todo funciona como era de esperar. A continuación se procede a la lectura de datos.

1.8.5 Monitorización de datos

Como ya se ha citado anteriormente para la lectura de la señal del encoder es necesario crear un segmento que cuente el número de impulsos que el encoder genera. Para determinar la velocidad del motor es necesario que un temporizador controle el tiempo en que tarda el contador rápido en contar los impulsos que genera el encoder.

Una vez obtenido el dato del temporizador basta con realizar unas operaciones matemáticas en labview que proporcionen el dato final de la velocidad del motor en un display. Las operaciones matemáticas necesarias se explican en el apartado de cálculos, en este apartado se muestra la apariencia del diagrama de bloques del programa en labview:

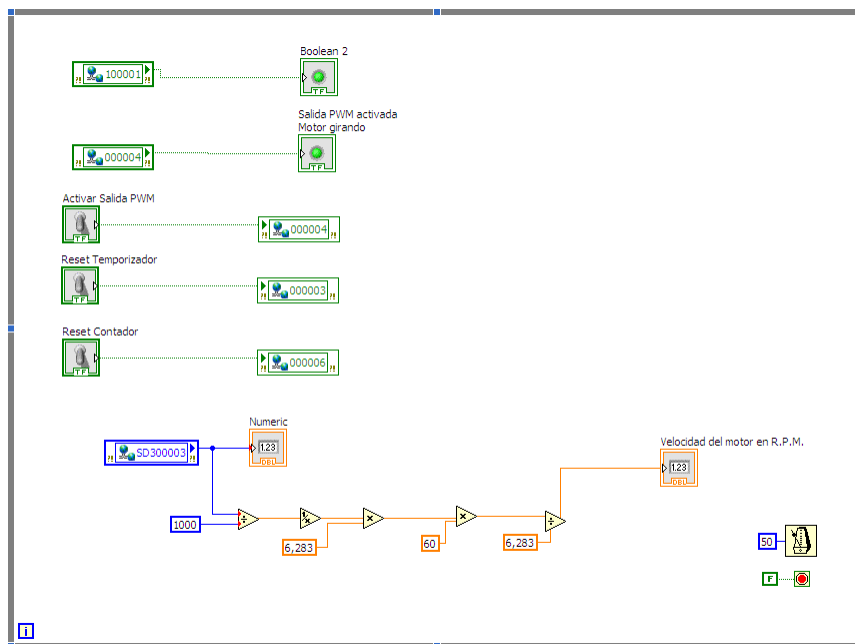


Figura 101. Diagrama de bloques para el control de velocidad del motor

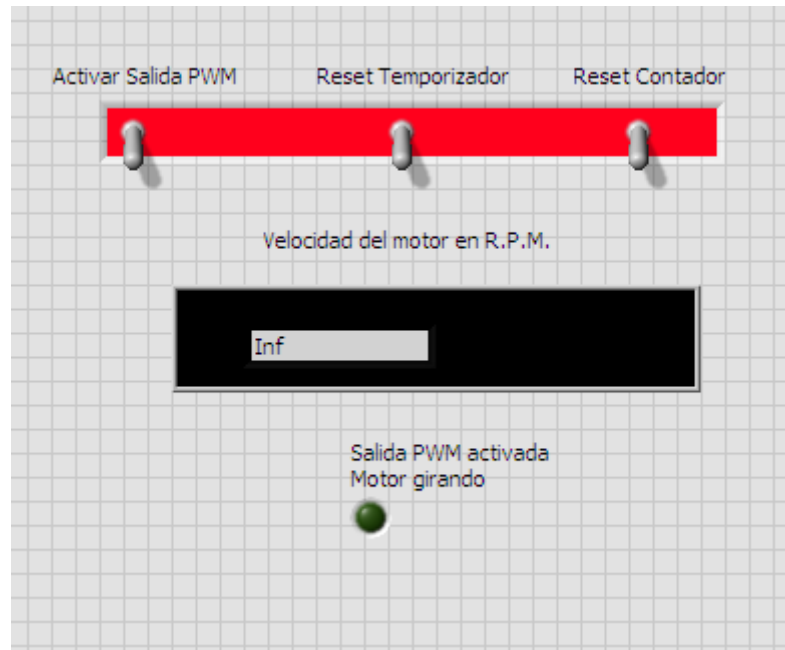


Figura 102. Panel de monitorización de Labview

Para comprobar el funcionamiento, se prueba con la señal de impulsos al 100%:

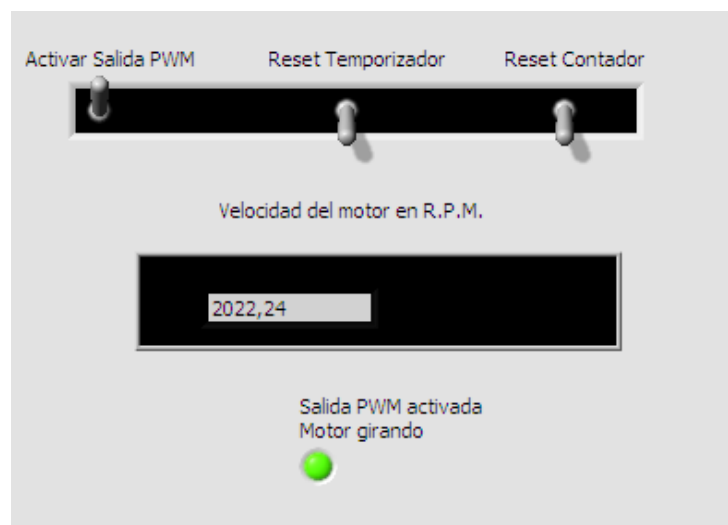


Figura 103. Velocidad del motor con la señal de entrada al 100%

A continuación se cambia el valor de la señal de impulsos al 50%, y se observa el valor obtenido. Es necesario hacer un reset al contador y al temporizador:

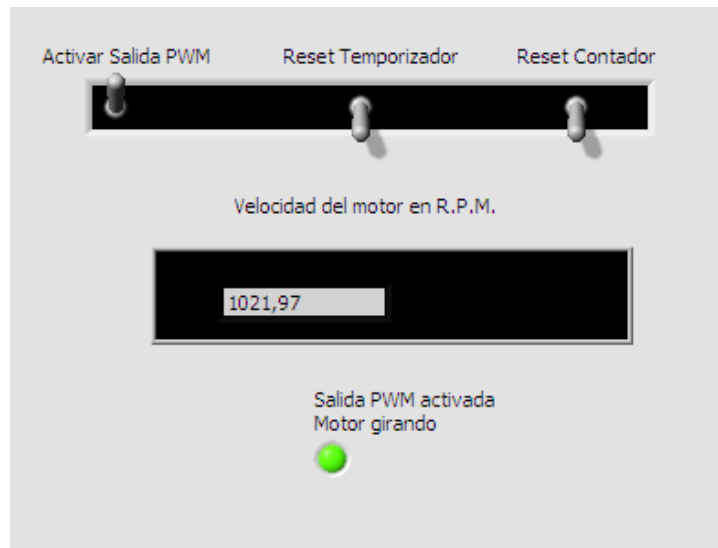


Figura 104.Velocidad del motor con la señal de entrada al 50%

Ahora lo mismo para un 80% de la señal de impulsos:

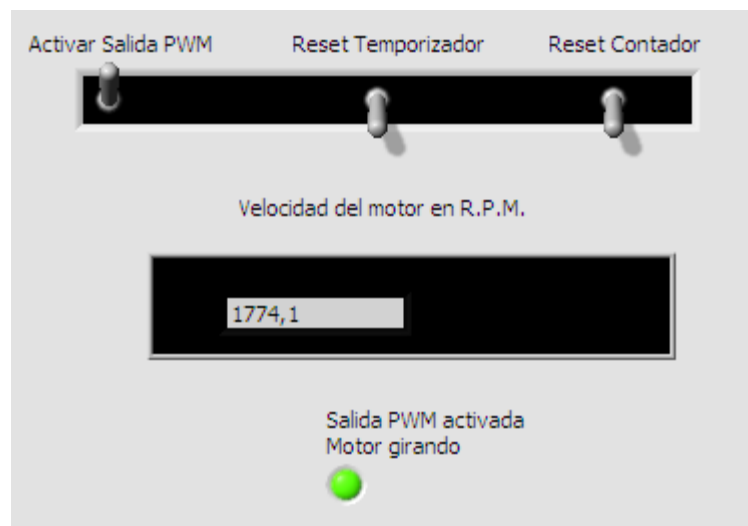


Figura 105.Velocidad del motor con la señal de entrada al 80%

Ahora lo mismo para un 30% de la señal de impulsos:



Figura 106.Velocidad del motor con la señal de entrada al 30%

Para que el motor arranque la señal de impulsos ha de ser como mínimo de un 20%, para así superar el par de arranque, sino no hay fuerza suficiente y el motor no arranca. En la figura siguiente se muestra la velocidad del motor para el 20% de la señal.

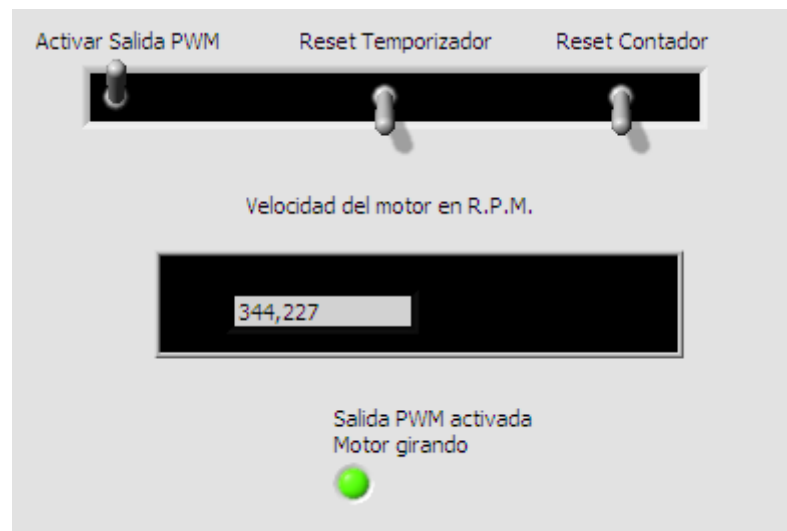


Figura 107.Velocidad del motor con la señal de entrada al 20%

Se ha modificado la programación para que el temporizador y el comparador se reseteen automáticamente, es decir, cuando se inicia el movimiento del motor el temporizador y contador se ponen a cero para empezar de nuevo a contar. A su vez se ha cambiado la forma de activar el motor, se activa con un pulsador y se desactiva con otro pulsador. A continuación se muestra como queda el panel:

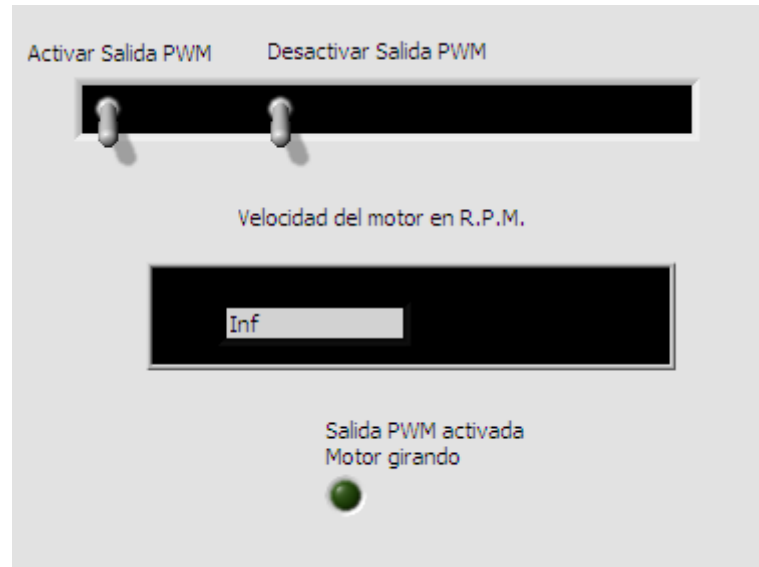


Figura 108. Ilustración del nuevo panel de Control

Como último, se ha conseguido la variación de una variable de tipo WORD. El problema que se ha presentado hasta el momento es que solo se podían acceder a ese tipo de variable en modo lectura, y una vez investigado se ha logrado acceder de modo escritura. Esto implica que se puede indicar al motor el número de impulsos que se desea que gire mediante un potenciómetro. Es decir, una vuelta son dos impulsos, si le indicamos 200 impulsos, el motor girará 100 vueltas.

Para realizar este paso ha sido necesario configurar el bloque MB_SERVER, concretamente el parámetro MB_HOLD_REG. Dicho parámetro se rige por las siguientes pautas:

El parámetro MB_HOLD_REG es un puntero hacia un búfer de datos para almacenar los datos que se han leído o escrito en el servidor Modbus. Puede usarse como búfer de datos un bloque de datos global o un área de memoria (M). Como puntero hacia un búfer del área de memoria (M), utilice el formato ANY según el patrón "P#dirección_del_bit" "Tipo de datos" "Longitud" (ejemplo: P#M1000.0 WORD 500).

En la siguiente tabla se muestran ejemplos de la representación de direcciones Modbus en el registro de parada de las funciones Modbus 3 (leer WORD), 6 (escribir WORD) y 16 (escribir varias WORD). El límite superior del número de direcciones del bloque de datos viene determinado por la memoria de trabajo máxima de la CPU. Si se utiliza un área de memoria, el número máximo de direcciones viene determinado por el tamaño del área de memoria de la CPU.

Direcciones Modbus	Parámetro MB_HOLD_REG: ejemplos		
	P#M100.0 WORD 5	P#DB10.DBx0.0 WORD 5	"Recipe".ingredient
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

El bloque programado en el PLC adquiere la siguiente forma:

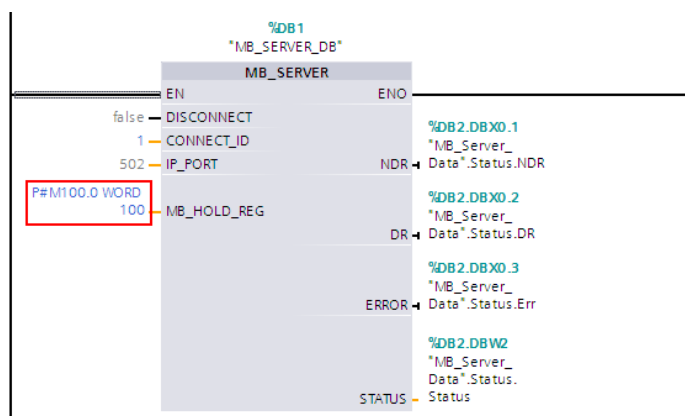


Figura 109. Bloque configurado para la escritura de datos tipo WORD

Se programa un indicador para que el usuario observe las vueltas a las que gira el motor y verifique a su vez lo citado anteriormente. Si el usuario gira el potenciómetro, éste indicará un número, y a su vez el indicador marcará la mitad. Como se puede observar en la figura siguiente, el potenciómetro marca 600 impulsos y el indicador 300 vueltas, por lo que el motor girará 300 vueltas y se detendrá.

El diseño del panel se ha cambiado debido a que se va a realizar la grabación de un vídeo de demostración y se pierde resolución. El panel queda de la siguiente manera:

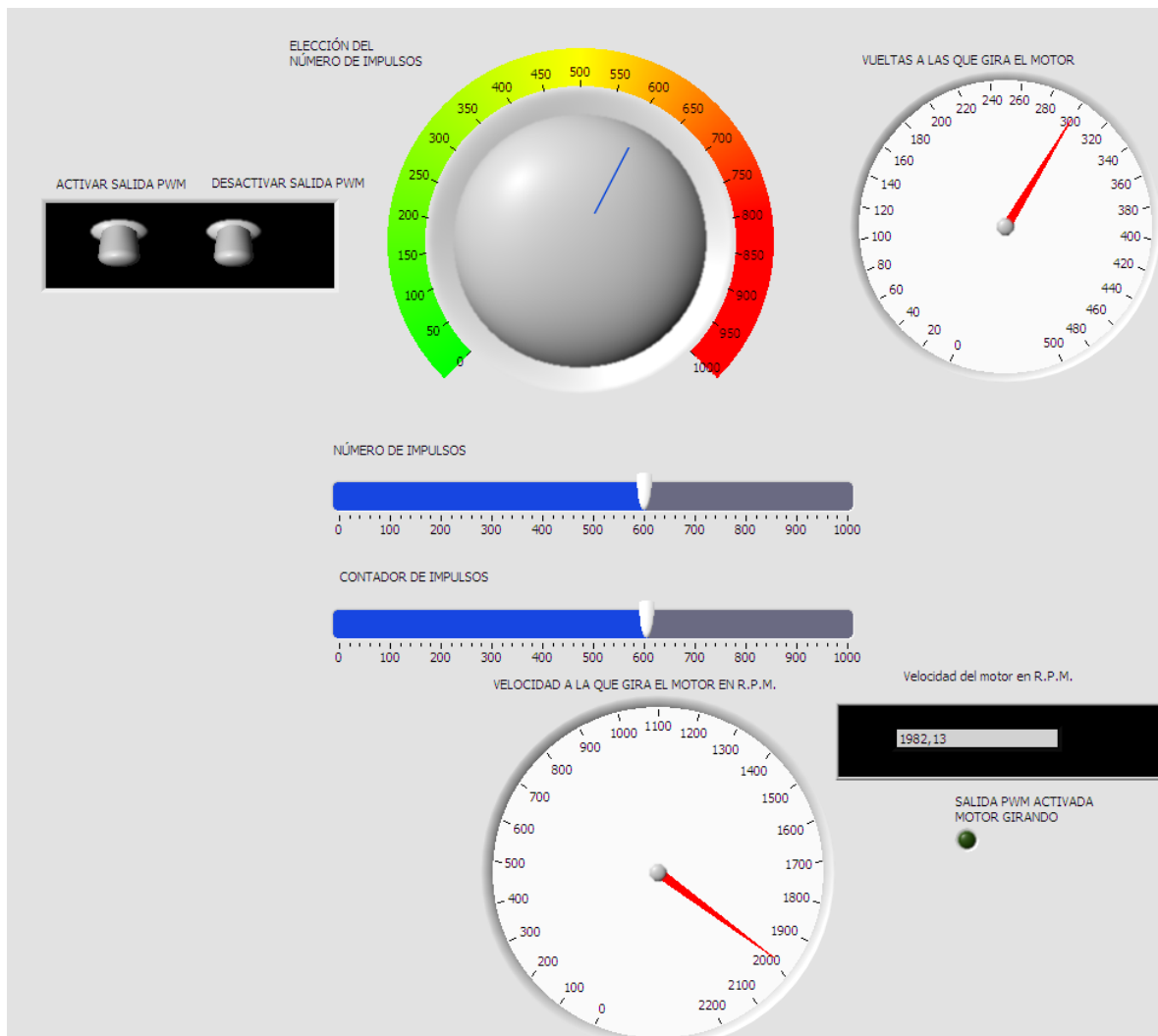


Figura 110. Panel de Control del Motor

1.8.6 Conclusiones

En este proyecto fin de carrera se ha logrado comunicar un PC con un PLC mediante Modbus. Esta parte del proyecto se alargó debido a las complicaciones que se sufrieron a la hora de comunicar una máquina con la otra. Se requerían versiones específicas tanto para el PLC como para el control desde el PLC, que en este caso se realizaba mediante Labview. Para el PLC se requería una versión de firmware 2.1 o superior, para el software de programación del PLC se necesitaba la versión 11 del TIA, y para Labview la versión 2011 o superior.

El siguiente logro obtenido es poder comunicar un PC con varios PLC's y poder administrar el manejo de variables entre ambos PLC's. Es decir, controlar los dos PLC's desde el PC, o controlar un PLC desde el otro PLC. Es una parte interesante debido a que se pueden controlar PLC's que estén en red en cualquier parte de la universidad, siempre y cuando tengan instalados los softwares necesarios y los programas adecuados. Esta parte del proyecto queda abierta para futuras prácticas ya que sólo se han realizado pruebas de pequeña magnitud, como el encendido y apagado de entradas y salidas.

Como ejemplo de aplicación de lo anterior se ha trabajado en el control desde un PC que hace las veces de Scada que controla a un PLC, desde el que se controla un motor de una placa de demostraciones del laboratorio. Para llegar a este punto ha sido necesario idear varios circuitos como ya se han citado anteriormente y hacer que cada parte funcionase en conjunto.

Tras conseguir monitorizar la velocidad del motor y controlar el encendido del mismo desde el PLC, se ha elaborado un guión de prácticas para alguna asignatura en la que se impartan contenidos sobre comunicaciones industriales. El fin de la práctica es también que los futuros alumnos de la universidad mejoren el programa para tener que evitar etapas de la programación.

Por otro lado se ha realizado un intenso trabajo de investigación de las distintas zonas de memoria del PLC y cómo controlarlas desde labview, tarea complicada dada la poca documentación disponible, y que se espera que se profundice en un futuro proyecto.

Finalmente se ha conseguido acceder a las variables de tipo WORD en modo escritura. Esto supone un gran avance por lo citado anteriormente ya que la documentación disponible es escasa y el acceso a la memoria del PLC también. Pero tras varios intentos esto se ha conseguido y prueba de ello es que se puede indicar cuantas vueltas ha de girar el motor mediante un potenciómetro.

Una de las líneas que quedan abiertas para un futuro es encontrar la forma de poder variar la velocidad del motor sin necesidad de hacerlo desde la configuración de la CPU.



CÁLCULOS

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber Marín Iturrarte
Tutor: Ignacio Del Villar Fernández

2. Cálculos

2.1 Cálculos Divisor de Tensión

Tensión de entrada 24V

Tensión de salida 5V

Para realizar los cálculos se toma como base que el divisor va a dar la misma tensión en ambos bornes, es decir, 12V y 12V. Con estos valores bastaría con poner dos resistencias de 2200Ω. Realizando una regla de tres se hallan los valores de resistencia necesarios para la aplicación. En la parte superior del divisor se tendrán 19V () y en la inferior 5V (), los necesarios para alimentar el motor

$$\left. \begin{array}{l} 12V \rightarrow 2200\Omega \\ 19V \rightarrow R_1 \end{array} \right\} R_1 = 3484\Omega$$

$$\left. \begin{array}{l} 12V \rightarrow 2200\Omega \\ 5V \rightarrow R_2 \end{array} \right\} R_2 = 917\Omega$$

Se normalizan los valores de resistencias, por lo que los valores que se han de elegir son los siguientes:

$$R_1 \rightarrow 3300\Omega$$

$$R_2 \rightarrow 820\Omega$$

2.2 Cálculos Resistencias del Transistor

Para que el transistor funcione como interruptor requiere ciertos valores de corriente en el colector y en la base que son los siguientes:

Suponiendo una caída de tensión entre la base y el emisor 0.7V se realizan los cálculos siguientes:

$$I_B = \frac{V_B - V_{BE}}{R_B} \rightarrow R_B = \frac{5 - 0.7}{0.5 \times 10^{-3}} = 8600\Omega$$

$$I_C = \frac{V_{CC} - V_C}{R_C} \rightarrow R_C = \frac{24 - 0}{10 \times 10^{-3}} = 2400\Omega$$

Se normalizan los valores de resistencias, por lo que los valores que se han de elegir son los siguientes:

$$R_B \rightarrow 8200\Omega$$

$$R_C \rightarrow 2200\Omega$$

2.3 Cálculos Velocidad Motor

$$f = \frac{\text{numero de impulsos}}{T} = \text{Hz}$$

$$\omega = 2\pi \times f = \frac{\text{rad}}{\text{s}}$$

$$\omega = \frac{\text{rad}}{\text{s}} \times \frac{60}{2\pi} = \text{r.p.m}$$



PRESUPUESTO

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber Marín Iturrarte
Tutor: Ignacio Del Villar Fernández

3. Presupuesto

Se ha realizado una estimación del presupuesto del presente Proyecto Fin de Carrera como si se tratase de un trabajo de investigación que se realiza dentro de una empresa. Para ello se ha de tener en cuenta varios apartados que se muestran a continuación.

3.1MATERIALES LABORATORIO

	Cantidad	Precio Unitario(€)	I.V.A.(%)	Importe(€)
PLC	1	1999	18	2358,82
PICDEM Mechatronic Demonstration Board	1	119,67	18	141,2106
Transistor	10	0,1175	18	1,3865
Amplificador Operacional	2	0.8	18	1,888
Resistencias	6			
820Ω	1	0,022		
1000Ω	2	0,022		
2200Ω	1	0,022	18	0,15576
3300Ω	1	0,022		
8200Ω	1	0,022		
Placa ARISTON	1	30	18	35,4
Conductores	1 (Rollo100metros)	10,2	18	12,036
TOTAL(€)				2550,89686

3.2 SOFTWARE

	Cantidad	Precio Unidad(€)	Costes Envío(€)	Importe (€)
Software TIA	1	GRATUITO		0
Software Labview	1	1551	17,83	1568,63
TOTAL (€)				1568,63

3.3 MANO DE OBRA

En este apartado del presupuesto se tiene en cuenta el coste del personal encargado de realizar el proyecto.

Se considera el sueldo de dos personas, ya que el proyecto ha sido realizado por un ingeniero y una persona que ha supervisado el proyecto en todo momento. Se estima el sueldo de un futuro ingeniero y un responsable que dedica un 20% de su trabajo en tareas de asesoramiento y revisión. Se tendrá en cuenta los cargos sociales.

3.3.1 SALARIO BASE

	Meses	Sueldo / Mes (€)	Total(€)
Ingeniero Técnico Industrial, especialidad Electricidad	6	1500	9000
Responsable Asesor	6	600	3600
TOTAL(€)			12600

3.3.2 CARGOS SOCIALES

	Porcentaje
Indemnización despido	3%
Seguros de accidente	5%
Subsidio familiar	3%
Subsidio vejez	5%
Abono días festivos	10%
Días de enfermedad	2%
Plus de cargas familiares	3%
Gratificación extraordinaria	10%
Otros conceptos	8%
TOTAL	49%

3.3.3 SALARIOS EFECTIVOS

Para el cálculo de sueldo final se añade al salario base los cargos sociales correspondientes.

	Salario base (€)	Cargas sociales(€)	Salario total(€)
Ingeniero Técnico Industrial, especialidad Electricidad	9000	4410	13410
Responsable asesor	3600	1764	5364
TOTAL(€)			18774 €

3.4 PRESUPUESTO TOTAL

El I.V.A. ha sido añadido en cada apartado, por lo que no es necesario añadirlo aquí. Se ha de tener en cuenta el beneficio industrial en la suma total del presupuesto.

Coste Material Laboratorio	2550,90€
Coste Personal	18774 €
Coste Software	1568,63€
Coste Total Ejecución Material	22893,53€
Beneficio Industrial 10% (E.M.)	2289,35€
COSTE TOTAL	25182,88€

El coste del proyecto asciende a la cantidad de VEINTICINCO MIL CIENTO OCHENTA Y DOS euros (25182,88) y OCHENTA Y OCHO céntimos de euro.



ANEXO MEMORIA

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber Marín Iturrarte
Tutor: Ignacio Del Villar Fernández

4. Anexo Memoria

4.1 Direccionamiento de variables en el PLC

Data Item	Data Type	Modbus		Modbus Slave		Description	Example
		Read	Write	Read	Write		
000001–065535	Boolean value	Yes	Yes	Yes	Yes	Accesses single-bit coils.	000001 = {000001}
100001–165535	Boolean value	Yes	No	Yes	Yes	Accesses single-bit discrete inputs.	100002 = {100002}
300001.1–365535.16	Boolean value	Yes	No	Yes	Yes	Accesses individual bits of input registers and interprets them as logical TRUE or FALSE values. The least significant bit is 1. The most significant bit is 16.	300001.1 = {the first bit of 300001}
300001–365535	16-bit unsigned integer	Yes	No	Yes	Yes	Accesses 16-bit input registers as unsigned integers ranging from 0 to 65,535.	300001 = {300001}
400001.1–465535.16	Boolean value	Yes	Yes	Yes	Yes	Accesses individual bits of holding registers and interprets them as logical TRUE or FALSE values. The least significant bit is 1. The most significant bit is 16.	400002.16 = {the 16th bit of 400002}
400001–465535	16-bit unsigned integer	Yes	Yes	Yes	Yes	Accesses 16-bit holding registers as unsigned integers ranging from 0 to 65,535.	400002 = {400002}
A000001L1–A065535L1	Array of Boolean values	Yes	Yes	Yes	Yes	Accesses arrays of single-bit coils.	A000001L2 = {000001, 000002}
A100001L1–A165535L1	Array of Boolean values	Yes	No	Yes	Yes	Accesses arrays of single-bit discrete inputs.	A100005L3 = {100005–100007}
A300001L1–A365535L1	Array of 16-bit unsigned integers	Yes	No	Yes	Yes	Accesses arrays of 16-bit input registers as arrays of unsigned integers.	A300001L2 = {300001, 300002}
A400001L1–A465535L1	Array of 16-bit unsigned integers	Yes	Yes	Yes	Yes	Accesses arrays of 16-bit holding registers as arrays of unsigned integers.	A400005L3 = {400005–400007}
AD300001L1 – AD365534L1	Array of 32-bit unsigned integers	Yes	No	Yes	Yes	Accesses arrays of 32-bit unsigned integers. Each 32-bit unsigned integer in the array is composed of two adjacent 16-bit input registers.	AD300001L1 = {300001, 300002}
AD400001L1 – AD465534L1	Array of 32-bit unsigned integers	Yes	Yes	Yes	Yes	Accesses arrays of 32-bit unsigned integers. Each 32-bit unsigned integer in the array is composed of two adjacent 16-bit holding registers.	AD400002L3 = {400002–400007}
AF300001L1 – AF365534L1	Array of 32-bit floating-point numbers	Yes	No	Yes	Yes	Accesses arrays of 32-bit floating-point numbers. Each 32-bit floating-point number in the array is composed of two adjacent 16-bit input registers.	AF300001L2 = {300001–300004}

AF400001L1 – AF465534L1	Array of 32-bit floating-point numbers	Yes	Yes	Yes	Yes	Accesses arrays of 32-bit floating-point numbers. Each 32-bit floating-point number in the array is composed of two adjacent 16-bit holding registers.	AF400002L3 = { 400002 – 400007 }
AS300001L1 – AS365535L1	Array of 16-bit signed integers	Yes	No	Yes	Yes	Accesses arrays of 16-bit input registers as arrays of signed integers.	AS300001L1 = { 300001 }
AS400001L1 – AS465535L1	Array of 16-bit signed integers	Yes	Yes	Yes	Yes	Accesses arrays of 16-bit holding registers as arrays of signed integers.	AS400002L3 = { 400002 – 400004 }
ASD300001L1 – ASD365534L1 1	Array of 32-bit signed integers	Yes	No	Yes	Yes	Accesses arrays of 32-bit signed integers. Each 32-bit signed integer in the array is composed of two adjacent 16-bit input registers.	ASD300001L1 = { 300001 , 300002 }
ASD400001L1 – ASD465534L1 1	Array of 32-bit signed integers	Yes	Yes	Yes	Yes	Accesses arrays of 32-bit signed integers. Each 32-bit signed integer in the array is composed of two adjacent 16-bit holding registers.	ASD400002L3 = { 400002 – 400007 }
CommFail	Boolean value	Yes	No	N/A	N/A	Represents a signal the Modbus I/O server generates. The signal is TRUE if the Shared Variable Engine fails to communicate with a Modbus device. Modbus Slave I/O servers do not support this data item.	N/A
D300001 – D365534	32-bit unsigned integer	Yes	No	Yes	Yes	Accesses two adjacent 16-bit input registers as one 32-bit unsigned integer ranging from 0 to 4,294,967,295.	D300001 = { 300001 , 300002 }
D400001 – D465534	32-bit unsigned integer	Yes	Yes	Yes	Yes	Accesses two adjacent 16-bit holding registers as one 32-bit unsigned integer ranging from 0 to 4,294,967,295.	D400002 = { 400002 , 400003 }
F300001 – F365534	32-bit floating-point number	Yes	No	Yes	Yes	Accesses two adjacent 16-bit input registers as one 32-bit floating-point number.	F300001 = { 300001 , 300002 }
F400001 – F465534	32-bit floating-point number	Yes	Yes	Yes	Yes	Accesses two adjacent 16-bit holding registers as one 32-bit floating-point number.	F400002 = { 400002 , 400003 }
OffHook	Boolean value	Yes	Yes	N/A	N/A	Specifies that a Modbus object retain exclusive use of a communication port when the value of OffHook is TRUE. If the value is FALSE, the Modbus object does not retain exclusive use of the communication port. Modbus Slave I/O servers do not support this data item.	N/A
S300001 – S365535	16-bit signed integer	Yes	No	Yes	Yes	Accesses 16-bit input registers as signed integers ranging from –32,768 to 32,767.	S300001 = { 300001 }
S400001 – S465535	16-bit signed integer	Yes	Yes	Yes	Yes	Accesses 16-bit holding registers as signed integers ranging from –32,768 to 32,767.	S400002 = { 400002 }
SD300001 – SD365534	32-bit signed integer	Yes	No	Yes	Yes	Accesses two adjacent 16-bit input registers as one 32-bit signed integer ranging from –2,147,483,648 to 2,147,483,647.	SD300001 = { 300001 , 300002 }

SD400001– SD465534	32-bit signed integer	Yes	Yes	Yes	Yes	Accesses two adjacent 16-bit holding registers as one 32- bit signed integer ranging from -2,147,483,648 to 2,147,483,647.	SD400002 = {400002, 400003}
UpdateNow	Boolean value	No	Yes	N/A	N/A	Specifies that the Modbus I/O server refresh the Modbus device once if the value of UpdateNow changes from FALSE to TRUE. Modbus Slave I/O servers do not support this data item.	N/A
UpdateRate	64-bit floating- point number	Yes	Yes	N/A	N/A	Specifies how often the Modbus I/O server refreshes a Modbus device, in seconds. You can specify a non- integer value for this data item. If the value of this data item is zero, the Modbus I/O server does not refresh the device. Modbus Slave I/O servers do not support this data item.	N/A
Updating	Boolean value	Yes	No	Yes	No	Represents a signal the Modbus or Modbus Slave I/O server generates. The signal is TRUE while the Modbus I/O server polls a Modbus device or the Modbus Slave I/O server is being updated.	N/A

Figura 1. Dirección de Variables

4.2 Hoja características transistor

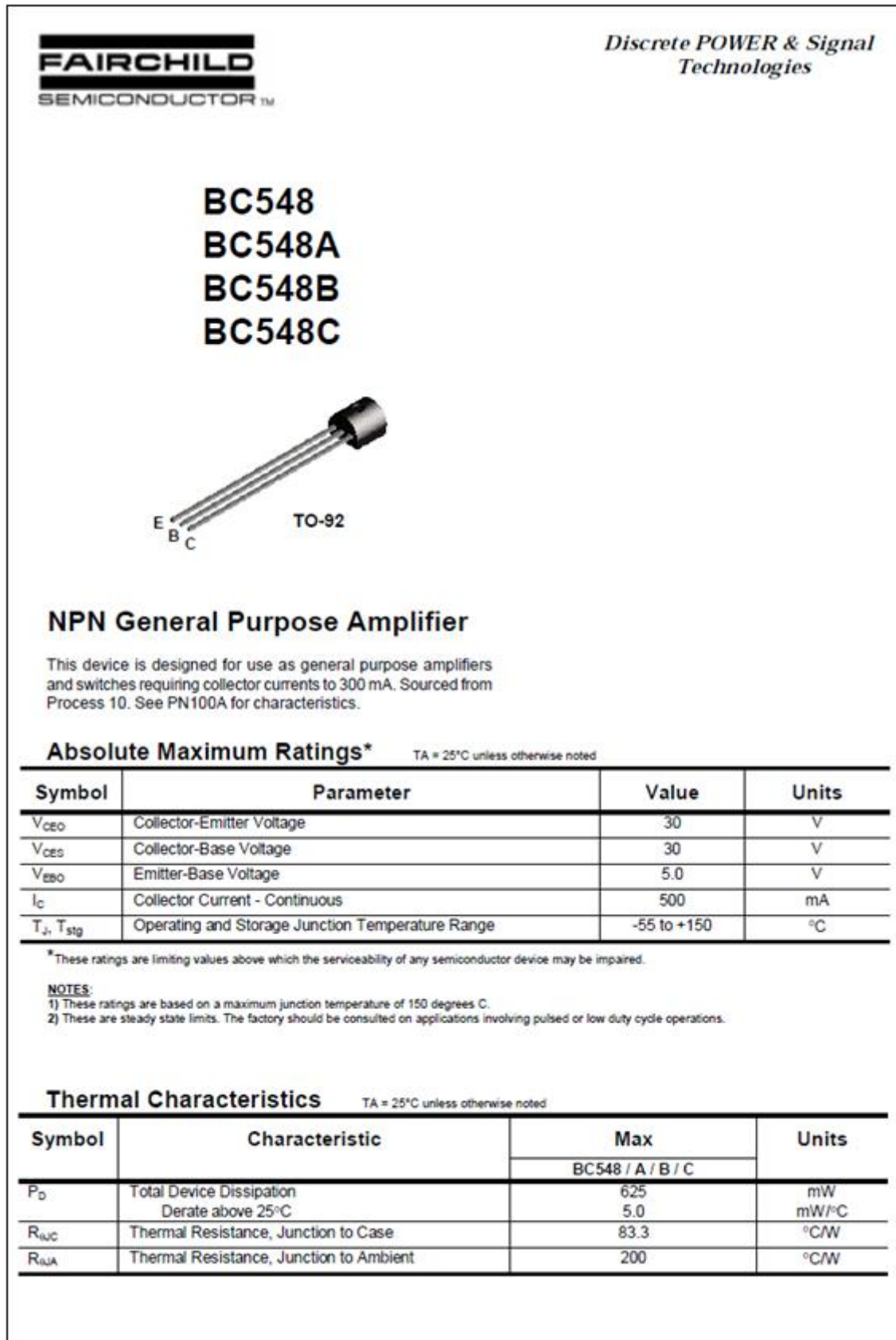


Figura 2. Hoja de Características del Transistor Utilizado

NPN General Purpose Amplifier (continued)					
Electrical Characteristics		TA = 25°C unless otherwise noted			
Symbol	Parameter	Test Conditions	Min	Max	Units
OFF CHARACTERISTICS					
V _{(BR)CEO}	Collector-Emitter Breakdown Voltage	I _C = 10 mA, I _B = 0	30		V
V _{(BR)CBO}	Collector-Base Breakdown Voltage	I _C = 10 μA, I _E = 0	30		V
V _{(BR)CES}	Collector-Base Breakdown Voltage	I _C = 10 μA, I _E = 0	30		V
V _{(BR)EBO}	Emitter-Base Breakdown Voltage	I _E = 10 μA, I _C = 0	5.0		V
I _{CBO}	Collector Cutoff Current	V _{CE} = 30 V, I _E = 0 V _{CB} = 30 V, I _E = 0, T _A = +150 °C		15 5.0	nA μA
ON CHARACTERISTICS					
h _{FE}	DC Current Gain	V _{CE} = 5.0 V, I _C = 2.0 mA	548 548A 548B 548C	110 110 200 420	800 220 450 800
V _{CE(sat)}	Collector-Emitter Saturation Voltage	I _C = 10 mA, I _B = 0.5 mA I _C = 100 mA, I _B = 5.0 mA			0.25 0.60
V _{BE(on)}	Base-Emitter On Voltage	V _{CE} = 5.0 V, I _C = 2.0 mA V _{CE} = 5.0 V, I _C = 10 mA	0.58	0.70 0.77	V V
SMALL SIGNAL CHARACTERISTICS					
h _{fe}	Small-Signal Current Gain	I _C = 2.0 mA, V _{CE} = 5.0 V, f = 1.0 kHz	125	900	
NF	Noise Figure	V _{CE} = 5.0 V, I _C = 200 μA, R _S = 2.0 kΩ, f = 1.0 kHz, B _W = 200 Hz		10	dB

Figura 2.Hoja de Características del Transistor Utilizado

4.3 LABVIEW

LabView es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.

Este programa fue creado por National Instruments (1976) para funcionar sobre máquinas MAC, y salió al mercado por primera vez en 1986. Actualmente está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. La última versión es la 2011.

Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica (Instrumentación electrónica) sino también a su programación embebida. Un lema tradicional de LabVIEW es: "La potencia está en el Software", que con la aparición de los sistemas multinúcleo se ha hecho aún más potente. Entre sus objetivos están: el de reducir el tiempo de desarrollo de aplicaciones de todo tipo (no sólo en ámbitos de Pruebas, Control y Diseño), y el de permitir la entrada a la informática a profesionales de cualquier otro campo. LabVIEW consigue combinarse con todo tipo de software y hardware, tanto del propio fabricante - tarjetas de adquisición de datos, PAC, Visión, instrumentos y otro Hardware- como de otros fabricantes.

Principales usos:

Este programa es usado principalmente por ingenieros y científicos para tareas como:

- Adquisición de datos y análisis matemático
- Comunicación y control de instrumentos de cualquier fabricante
- Automatización industrial y programación de PACs (Controlador de Automatización Programable)
 - Diseño de controladores: simulación, prototipaje rápido, hardware-en-el-ciclo (HIL) y validación
 - Diseño embebido de micros y chips
 - Control y supervisión de procesos
 - Visión artificial y control de movimiento
 - Robótica
 - Domótica y redes de sensores inalámbricos

Principales características

Su principal característica es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación, que pueden hacer (programas) relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. Otra característica es la rapidez con la que se pueden hacer programas con LabVIEW y cualquier programador, por experimentado que sea, puede beneficiarse de él. Los programas en LabVIEW son llamados instrumentos virtuales (VIs). Para los amantes de lo complejo, con LabVIEW pueden crearse programas de miles de VIs (equivalente a millones de páginas de código texto) como pueden ser aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, proyectos para combinar nuevos VIs con VIs ya creados, etc.

Programar en LabVIEW

Como ya se ha mencionado anteriormente Labview es una herramienta gráfica de programación, esto significa que los programas no se escriben, sino que se dibujan, facilitando su comprensión. Al tener ya prediseñados una gran cantidad de bloques, se le facilita al usuario la creación del proyecto, con lo cual en vez de estar una gran cantidad de tiempo en programar un dispositivo/bloque, se le permite invertir mucho menos tiempo y dedicarse un poco más en la interfaz gráfica y la interacción con el usuario final. Cada VI consta de dos partes diferenciadas:

- **Panel Frontal:** El Panel Frontal es la interfaz con el usuario, se utiliza para interactuar con el usuario cuando el programa se está ejecutando. Los usuarios podrán observar los datos del programa actualizados en tiempo real (la velocidad de un motor, la temperatura de un horno, etc). En esta interfaz se definen los controles (se usan como entradas, pueden ser botones, marcadores etc...) e indicadores (se usan como salidas, pueden ser gráficas....).

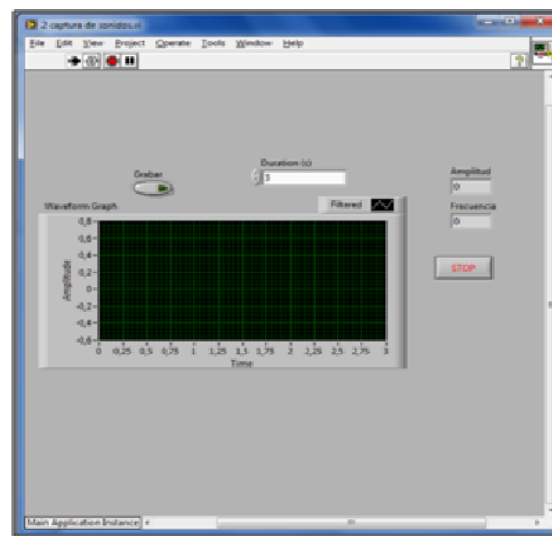


Figura 3. Panel Frontal

- **Diagrama de Bloques:** es el programa propiamente dicho, donde se define su funcionalidad. Aquí se colocan íconos que realizan una determinada función y se interconectan constituyendo el código que controla el programa. Suele haber una tercera parte icono/conector que son los medios utilizados para conectar un VI con otros VIs.

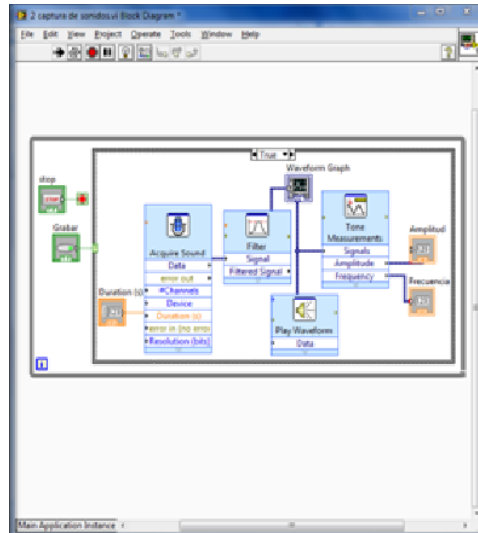


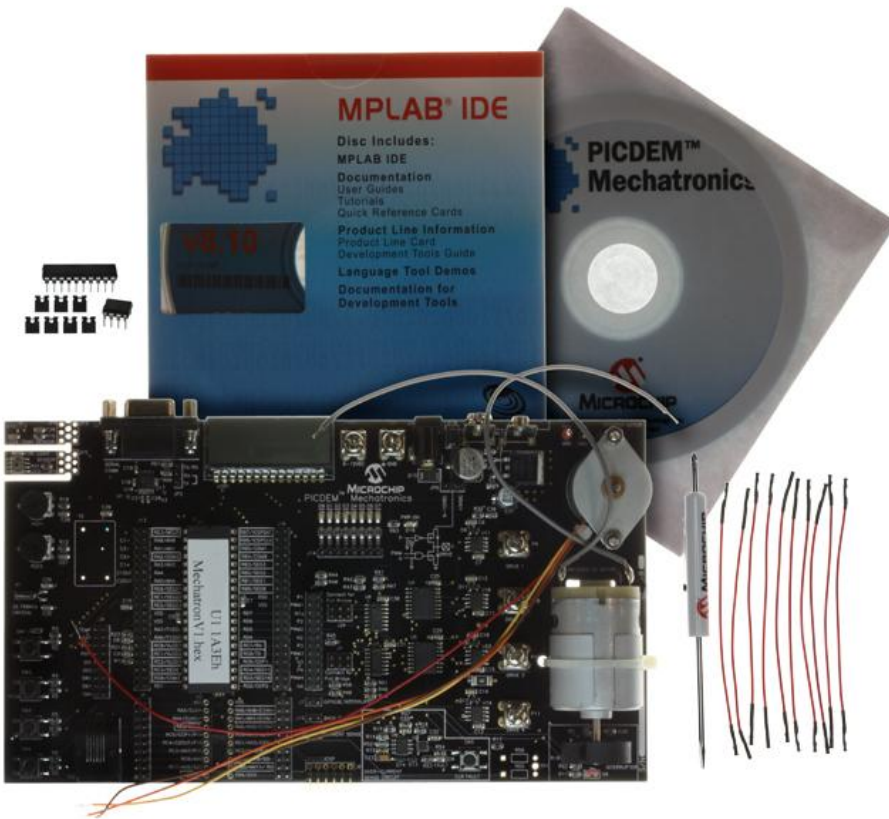
Figura 114. Diagrama de Bloques

En el panel frontal se encuentran todo tipo de controles o indicadores, donde cada uno de estos elementos tiene asignado en el diagrama de bloques una terminal, es decir el usuario podrá diseñar un proyecto en el panel frontal con controles e indicadores, donde estos elementos serán las entradas y salidas que interactuarán con la terminal del VI. Se puede observar que en el diagrama de bloques todos los valores de los controles e indicadores van fluyendo entre ellos cuando se está ejecutando un programa.

4.4 Guión de Prácticas

[Año]

[Escribir el título del documento]



amarin

UPNA

ÍNDICE

Sección 1. Comunicación PLC-Labview

- Configuración de la comunicación Modbus en Labview.....pág. 4-7
- Creación de variables en Labview.....pág. 8-10
- Configuración de la comunicación Modbus en el PLC.....pág.11
- Conexión Modbus entre Labview y el PLC.....pág.12
- Ejemplos.....pág.13-15
- Ejercicio propuesto.....pág.16

Sección 2. Configuración de la placa del motor

- Adaptación de la señal de entrada del PLC a la placa.....pág.16-21
- Configuración de la placa para el funcionamiento del motor.....pág.21-22
- Adaptación de la señal de salida de la placa al PLC.....pág.23-24
- Ejercicio propuesto.....pág.25

Sección 3. Control de la velocidad del motor.....pág.25

- Aplicar una señal PWM desde el PLC a la placa, controlando la señal desde Labview
- Variar la velocidad del motor desde Labview
- Programar en Labview para obtener el dato de la velocidad del motor

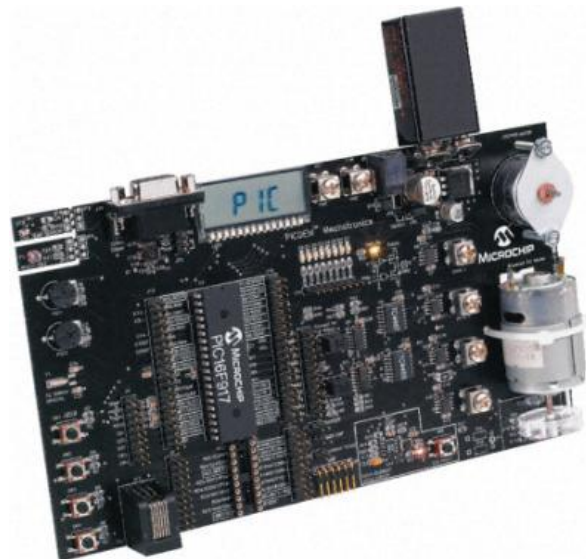
La práctica que se va a realizar consiste en comunicar un PLC S7-1200 con el software Labview mediante el protocolo comunicación Modbus. Una vez realizada esta parte, la práctica se centra en controlar la velocidad de un motor.

La práctica consta de tres partes:

- Comunicación PLC-Labview
- Configuración de la placa del motor
- Control de la velocidad del motor



PLC S7-1200



PICDEM Mechatronics Demonstration Board

Sección 1.Comunicación PLC-Labview

- **Configuración de la comunicación Modbus en Labview**

El primer paso consiste en configurar Labview para realizar una comunicación mediante Modbus TCP.

Se comienza por abrir el programa Labview, el cual presenta el siguiente aspecto:



Figura1.Página Inicial de Labview

A continuación se crea el nuevo proyecto:



Figura2.Creación del Proyecto

El proyecto aparece con el siguiente formato:

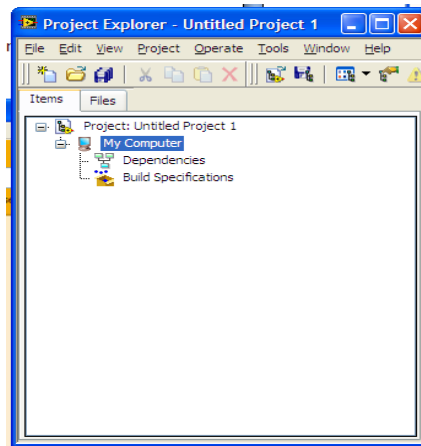


Figura3. Página Inicial del Proyecto

Se guarda el proyecto con el nombre de Master o Maestro, como el usuario desee en el momento de la creación, ya que el nombre solo va a ser una forma de identificar el proyecto:

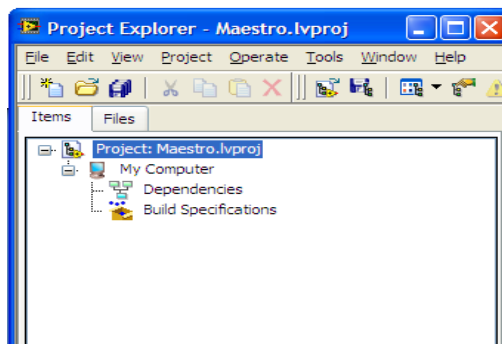


Figura4. Guardar el proyecto

El siguiente paso consiste en añadir una librería al proyecto, la cual va a contener las variables que se desean compartir:

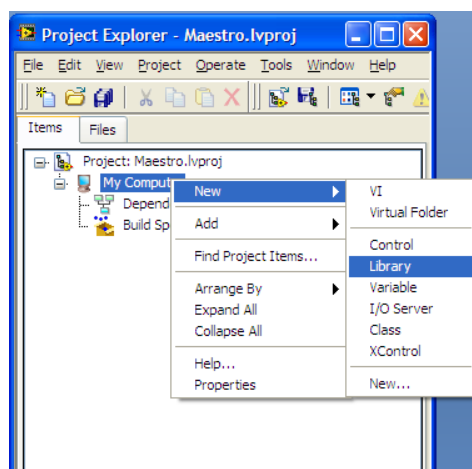


Figura5. Añadir librería al proyecto

Se guarda la librería con el nombre que se desee, en este caso es recomendable utilizar el nombre del proyecto para así no confundir las librerías con las de otros proyectos:

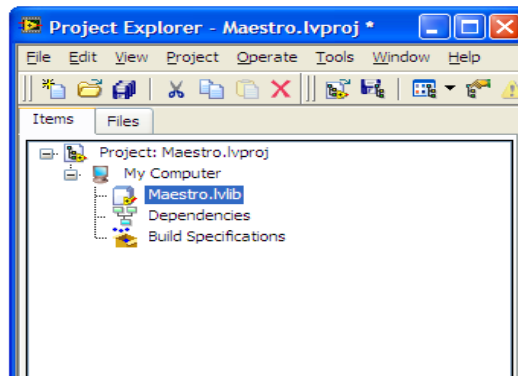


Figura6.Guardar librería

El siguiente paso es añadir a la librería un I/O Server, el cual va a permitir que las variables del proyecto sean compartidas y accesibles:



Figura7.Añadir I/O Server a la librería

Aparece el siguiente cuadro, en el cual se permite la elección del tipo de comunicación que se va a utilizar. En este caso es comunicación Modbus, por lo que se escogerá la opción de Modbus, no Modbus Slave, ya que se está creando el maestro:

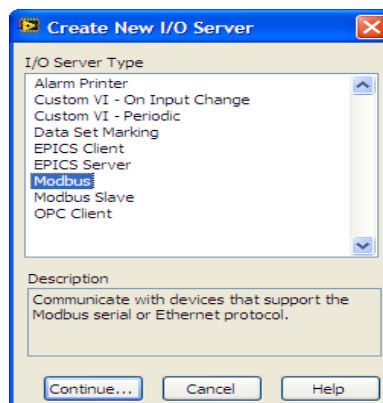


Figura8.Elección del tipo de comunicación

Se dispone de las versiones de Modbus Serial y Ethernet. De ambas se escogerá Ethernet, porque en realidad es Modbus TCP. Es decir, se trata simplemente de una cuestión de notación. Hay sitios donde se llama a Modbus TCP como Modbus Ethernet.

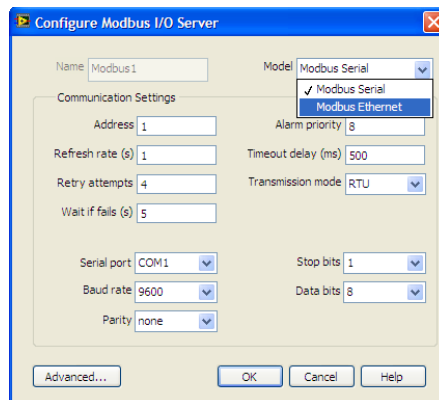


Figura9.Elección modo Ethernet

Una vez se escoge la opción de Ethernet es necesario indicar la dirección IP. En este caso se utiliza un IP en concreto que es la del propio ordenador, pero se puede utilizar la IP del dispositivo deseado, como puede ser un PLC:

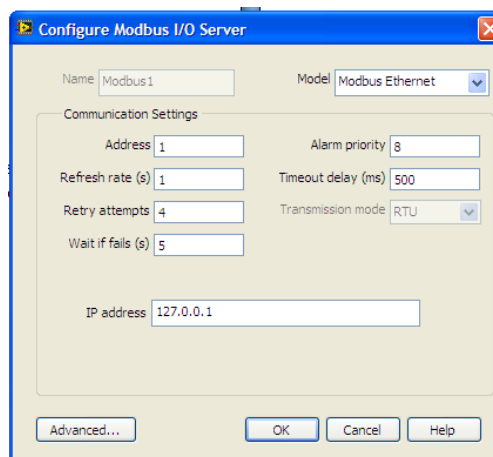


Figura10.Selección dirección IP

Se observa que en la librería se ha creado un nuevo elemento:

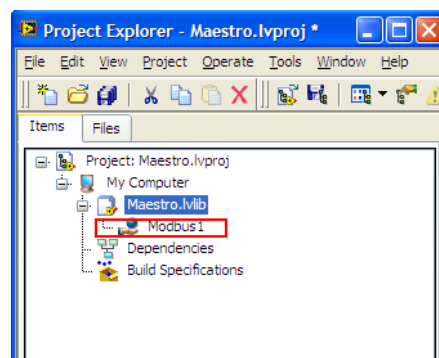


Figura11.Creación del I/O Server

- **Creación de variables en Labview**

El siguiente paso es crear las variables que se consideren necesarias para llevar a cabo el proyecto. En este caso se crearan varias de prueba, unas de tipo booleano y otras de tipo entero. Para ello es necesario seguir varios pasos:

Primero se crean las variables:

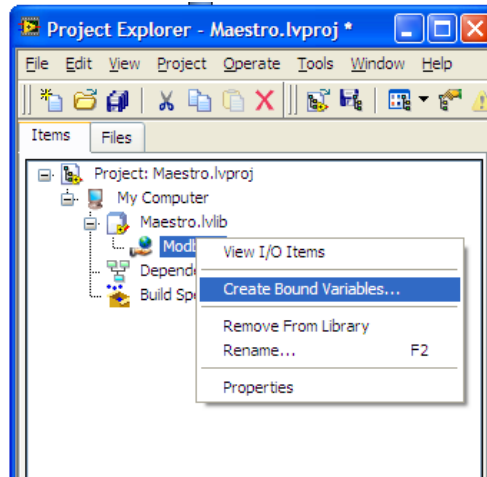


Figura12.Creación de Variables

En segundo lugar se elige el rango y tipo de variables:

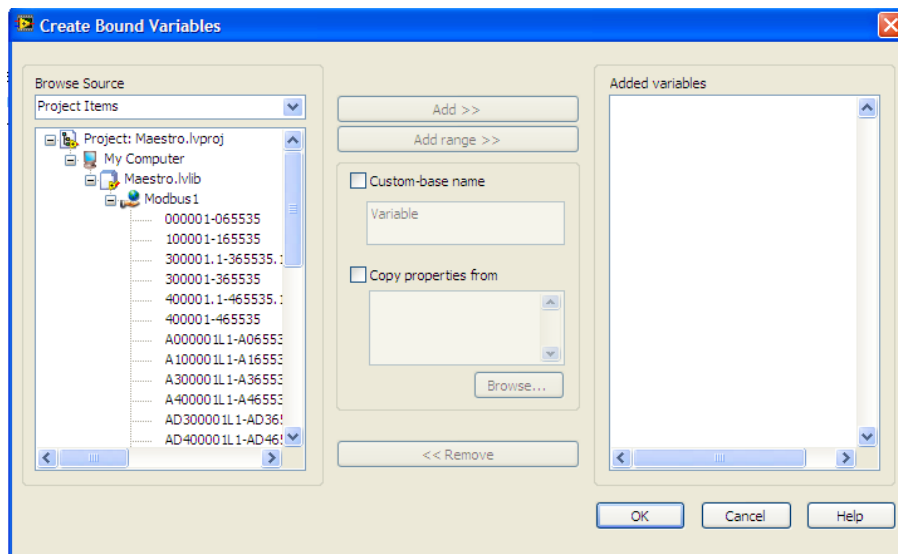


Figura13.Selección del rango de variables

Se ha de tener especial cuidado a la hora de elegir las variables, ya que hay variables que son de lectura y escritura y otras solo de lectura.

A modo de ejemplo, en primer lugar se eligen variables de tipo booleano que son de escritura y lectura. Estas van de la dirección 000001 a la 065535. También se escogerán variables de tipo entero, que son las que van de la dirección 400001 a la 46535, las cuales también son de escritura y lectura:

Se seleccionan diez variables de tipo booleano y se añaden:

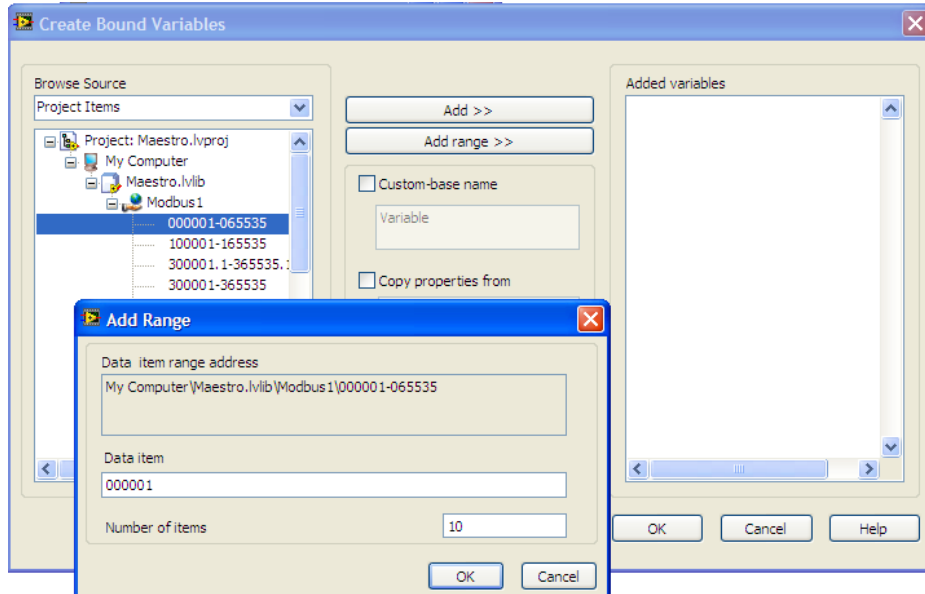


Figura14. Creación del nº de variables deseadas

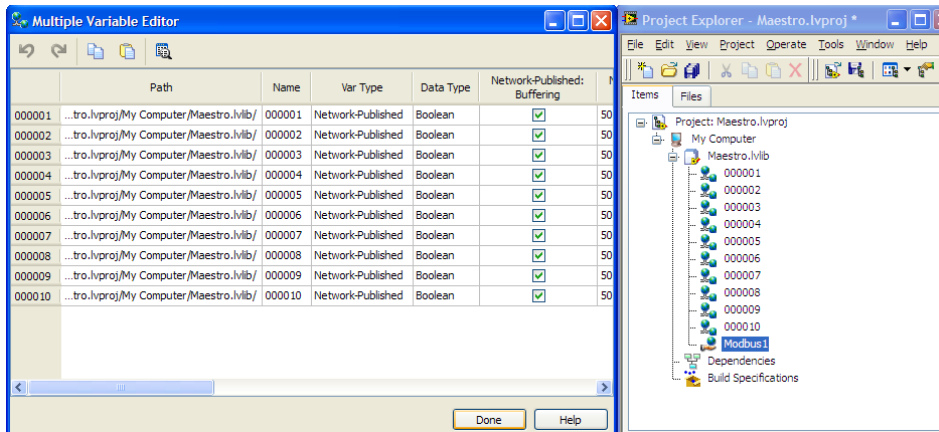


Figura15. Variables dentro del proyecto

Se observa que las variables se han adjuntado al proyecto. Se repite el mismo procedimiento para las variables de tipo entero:

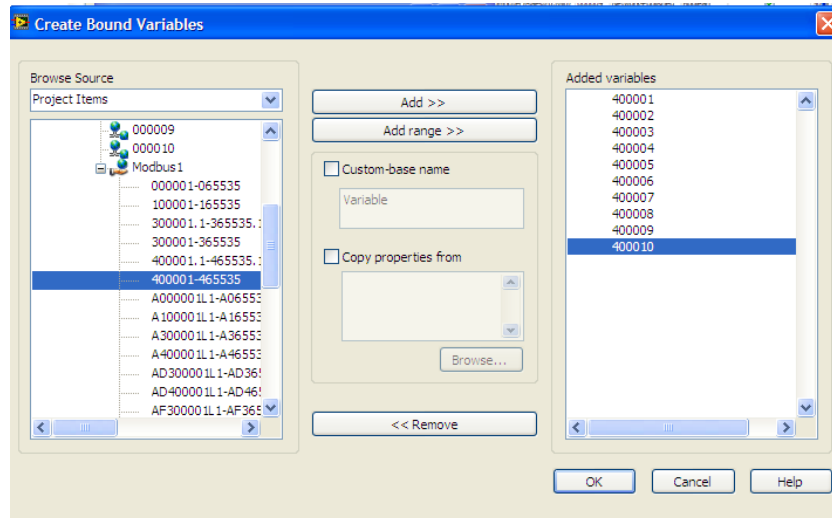


Figura16. Variables tipo entero

Se observa que las variables han sido añadidas al proyecto:

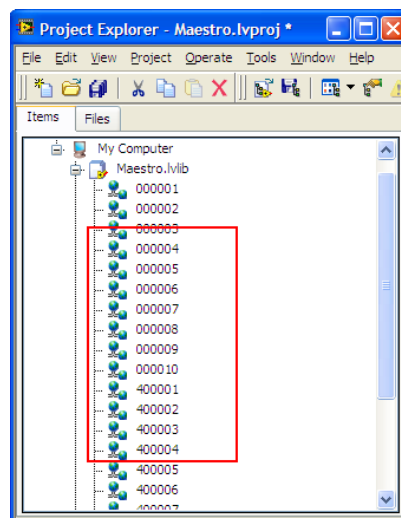


Figura17. Variables añadidas

- **Configuración de la comunicación Modbus en el PLC**

El servidor en este caso es el PLC, donde labview accederá a leer las variables. Para ello es necesario indicar al propio PLC que va a trabajar como servidor. Esto se realiza cargando al PLC un programa con la función servidor como se muestra a continuación:

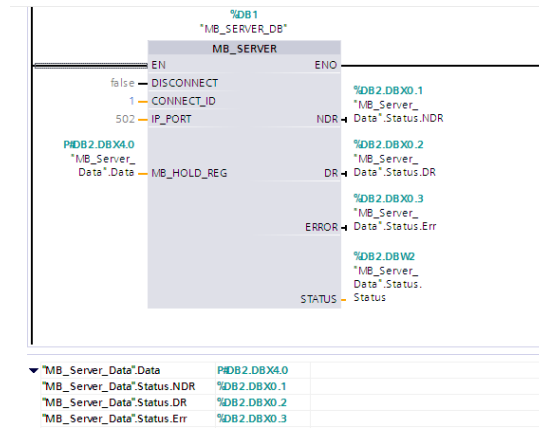


Figura18.Bloque Servidor del PLC

La instrucción "MB_SERVER" permite la comunicación como servidor Modbus TCP a través de la conexión PROFINET de la CPU S7-1200. Para utilizar esta instrucción no se requiere ningún módulo de hardware adicional. La instrucción "MB_SERVER" permite procesar peticiones de conexión de un cliente Modbus TCP, recibir peticiones de funciones Modbus y enviar mensajes de respuesta.

Para llevar a cabo la programación del bloque el alumno ha de consultar la ayuda. Se adjunta también un programa en el cual el bloque ya está programado para facilitar al alumno la realización de la práctica.

- **Conexión Modbus entre Labview y el PLC**

Una vez se ha cargado el programa en el PLC se accede a leer las variables del PLC desde labview. Este proceso se realiza a través del Distributed Manager. Es necesario tener presente en que parte de la memoria del PLC se direccionan las variables, ya que existen distintos tipos de variables tanto en formato como en el modo de acceso (escritura o lectura).

El acceso al Distributed Manager se realiza desde la venta del proyecto en la opción TOOLS como se muestra a continuación:

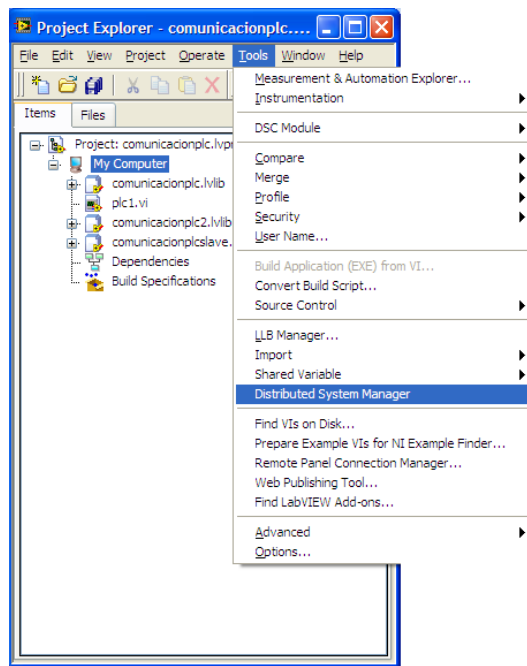


Figura19. Acceso a Distributed Manager

Para asegurarse que la comunicación es correcta existe una variable que indica false si la comunicación está bien o true si en la comunicación existe algún fallo:

A40000 1L 1-A465535L 1	Read/Write
AD30000 1L 1-AD365534L 1	Read
AD40000 1L 1-AD465534L 1	Read/Write
AF30000 1L 1-AF365534L 1	Read
AF40000 1L 1-AF465534L 1	Read/Write
AS30000 1L 1-AS365535L 1	Read
AS40000 1L 1-AS465535L 1	Read/Write
ASD30000 1L 1-ASD365534L 1	Read
ASD40000 1L 1-ASD465534L 1	Read/Write
TF CommFail false	Read
D30000 1-D365534	Read
D40000 1-D465534	Read/Write
F30000 1-F365534	Read
F40000 1-F465534	Read/Write

Figura20. Comunicación Correcta

- **Ejemplos**

Como parte final de la comunicación se realizan dos ejemplos:

Ejemplo1: Monitorización de variables del PLC con el Distributed Manager

Si la entrada del PLC I0.0 (100001) está conectada las salidas Q0.3 (000004) y Q0.4 (000005) se encuentran desactivadas. Si la entrada I0.0 está desactivada las salidas Q0.3 y Q0.4 se activan. Se carga el programa siguiente al PLC y se observa lo que ocurre:

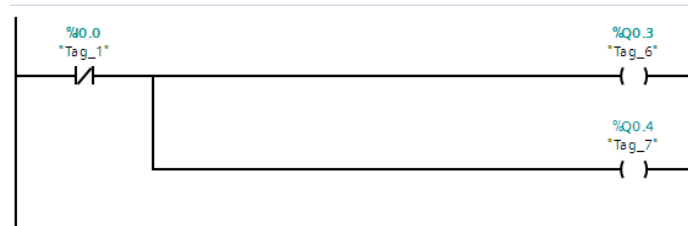


Figura21.Bloque de programa del PLC

Variable	Value	Access
TF 000001	false	Read/Write
TF 000002	false	Read/Write
TF 000003	false	Read/Write
TF 000004	false	Read/Write
TF 000005	false	Read/Write
TF 100001	true	Read
TF 100002	false	Read
TF 100003	false	Read
TF 100004	false	Read
TF 100005	false	Read

Figura22.Entrada I0.0 activada, salidas Q0.3 y Q0.4 desactivadas

Se alimenta la entrada I0.0 del PLC:

Variable	Value	Access
TF 000001	false	Read/Write
TF 000002	false	Read/Write
TF 000003	false	Read/Write
TF 000004	true	Read/Write
TF 000005	true	Read/Write
TF 100001	false	Read
TF 100002	false	Read
TF 100003	false	Read
TF 100004	false	Read
TF 100005	false	Read

Figura23.Entrada I0.0 desactivada, salidas Q0.3 y Q0.4 activadas

Ejemplo 2: Control de variables de PLC mediante Labview:

Monitorización:

El siguiente paso consiste en que al activar la entrada I0.0 (100001) en el PLC y que esto se refleje en labview con un led encendido.

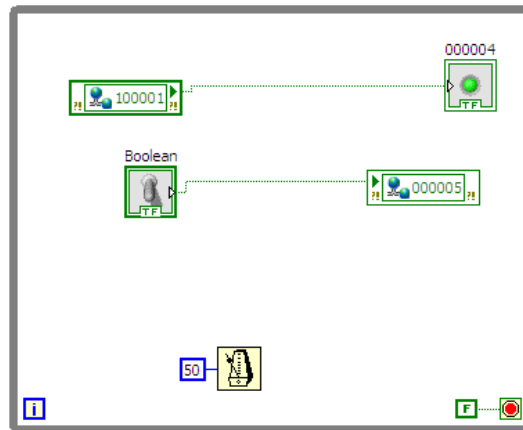


Figura24. Panel Frontal de Labview donde se programa

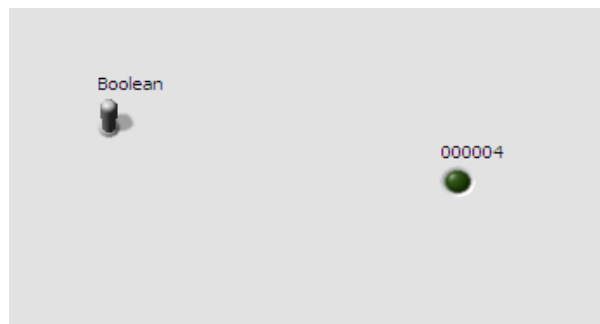


Figura25. Panel Frontal de Labview donde se observan los cambios

TF	000003	false	Read/Write
TF	000004	true	Read/Write
TF	000005	false	Read/Write
TF	100001	false	Read
TF	100002	false	Read
TF	100003	false	Read
TF	100004	false	Read
TF	100005	false	Read
	Modbus 1		

Figura26. Distributed Manager donde queda reflejada la situación inicial

Ahora se procede a alimentar la entrada del PLC IO.0 (100001) con un conductor a 24V:

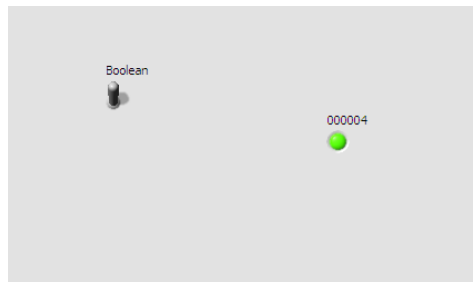


Figura26.Led activado debido a la alimentación de la entrada del PLC

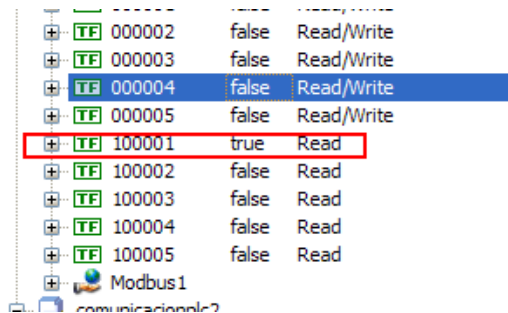


Figura27.Entrada PLC activada

Control de variables:

Por otra parte se ha probado a utilizar un interruptor en labview que active una salida del PLC, en este caso la salida Q0.4 (000005). Se muestra la situación inicial:



Figura28.Interruptor Labview conectado a la salida del PLC

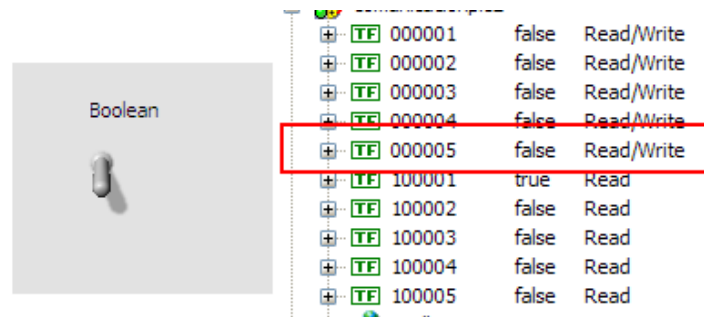


Figura29.Interruptor desactivado.Estado Inicial

Se activa el interruptor para observar el cambio de estado de la salida Q0.4 (000005):

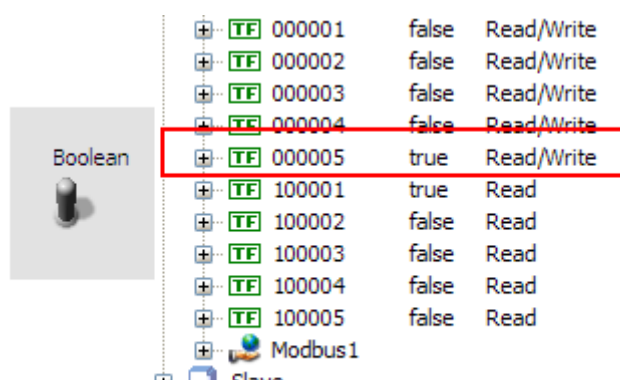


Figura30.Interruptor activado. Salida Q0.4 (000005) activada

- **Ejercicio**

Diseña un programa en labview que haga que, cuando se pulse un botón, active todas las salidas del PLC de tal forma que parpadeen con un periodo de 1 segundo. Para ello debes programar el PLC de tal forma que al activarse una variable controlada por labview se activen las salidas. Añade en labview controles para monitorizar las salidas del PLC

Sección 2. Configuración de la placa del motor

- **Adaptación de la señal de entrada del PLC a la placa**

La placa de demostración está alimentada con un adaptador de red de 230V/9V. A su vez, mediante un regulador, se convierten los 9V a 5V, que es la tensión con la que trabaja el microcontrolador de la placa. Como se va a controlar solo el motor y el encoder de la tarjeta, se prescinde del microcontrolador y se alimenta el motor exteriormente, es decir, se alimenta el motor con una señal de 5V desde el PLC.

La señal que procede del PLC es una señal de impulsos con forma cuadrada, ya que el motor es de tipo paso a paso. El PLC es capaz de generar una señal cuadrada de 24V, por lo que es necesario realizar un circuito de adaptación para reducir la tensión. Para ello se monta un circuito divisor para que entre los bornes deseados se obtenga 5V. Para que el PLC dé una señal cuadrada se han de realizar cambios en la configuración del dispositivo, y cargar un programa con un bloque que indique que esa señal es cuadrada.

Se procede al cambio de configuración. Primeramente se han de cambiar varias opciones en la configuración de las propiedades de la CPU del PLC:

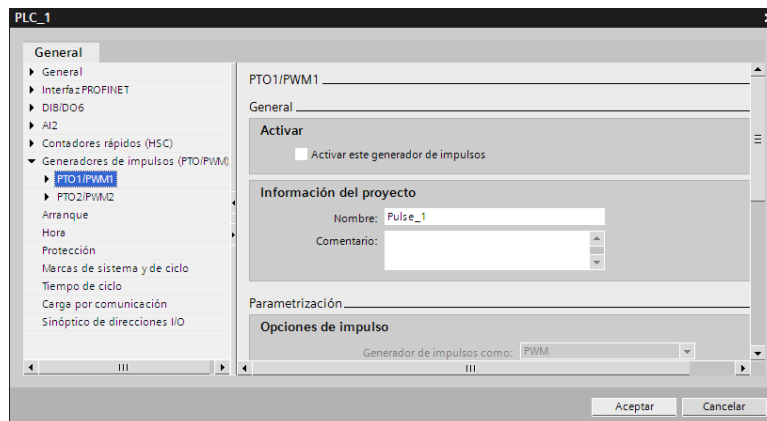


Figura31. Panel de Configuración de la CPU

Se ha de activar la opción de “Activar generador de Impulso”

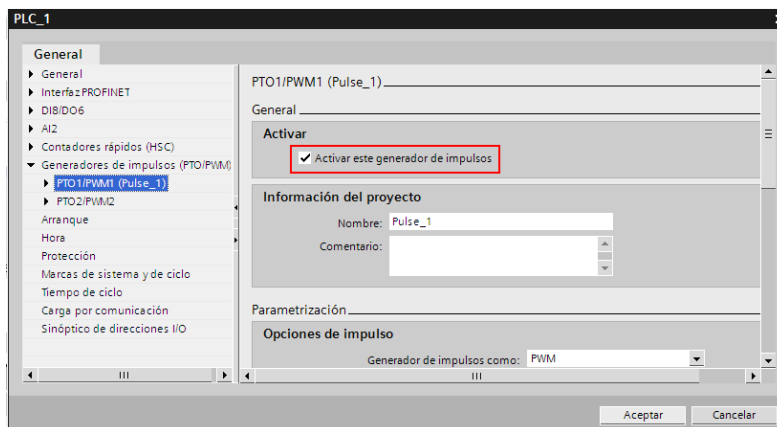


Figura32. Activar el generador de impulsos

Si se desea se pueden cambiar los datos por defecto que da el PLC, como la duración del impulso, su amplitud...etc, como se muestra en la figura 33.

La duración de impulso puede expresarse en centésimos del tiempo de ciclo (0 – 100), milésimos (0 – 1000), diezmilésimos (0 – 10000) o formato analógico S7. La duración de impulso puede variar entre 0 (sin impulso, siempre off) y escala completa (sin impulso, siempre on).

En este caso se los valores requeridos son de unos 40kHz, y el ciclo de trabajo será entre el 0 y el 100%:

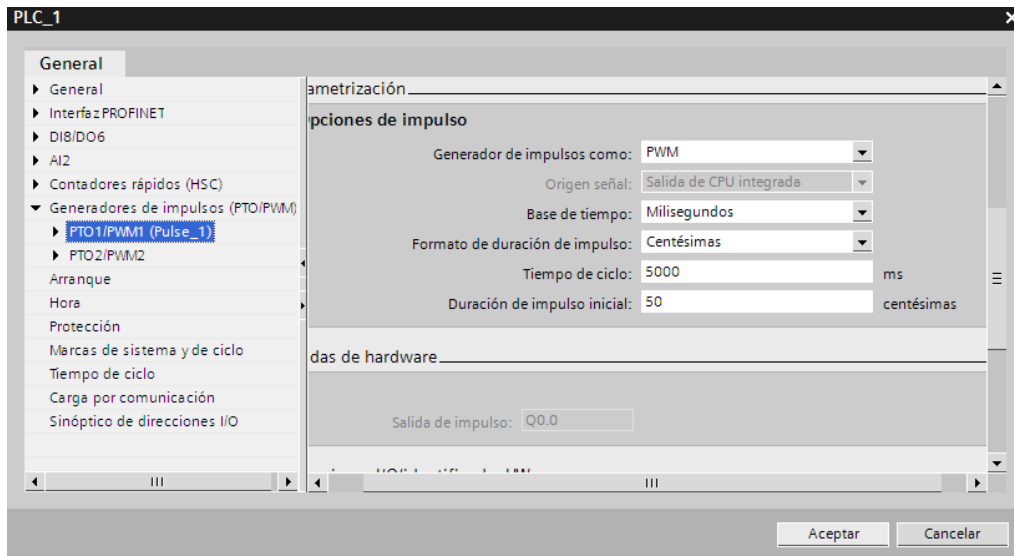


Figura33. Valores de la señal de impulso

Una vez se han realizado los cambios se procede a la configuración del bloque que ordena la señal de impulsos. La instrucción CTRL_PWM (Controlar modulación del ancho de pulso) ofrece un tiempo de ciclo fijo con un ciclo de trabajo variable. La salida PWM se ejecuta continuamente tras haberse iniciado a la frecuencia indicada (tiempo de ciclo). La duración de impulso varía según sea necesario para obtener el control deseado. La función es la CTRL_PWM_DB que tiene la siguiente apariencia:

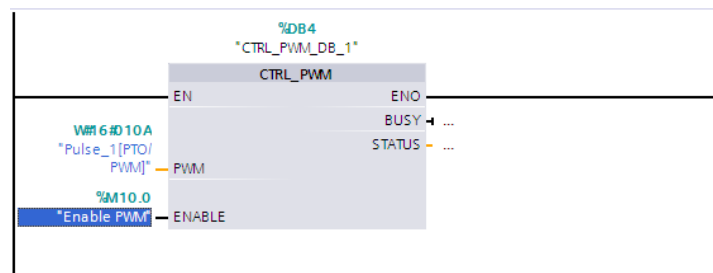


Figura34. Bloque de Función CTRL_PWM_DB

A continuación se muestra cómo la entrada del bloque de función “Enable PWM” está a false, con lo que la señal de impulsos se encuentra desactivada.

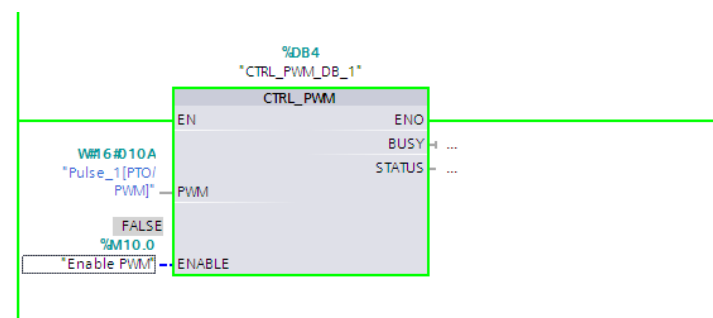


Figura35. Visión online, señal de impulso desactivada

Se procede a activar la señal de impulsos:

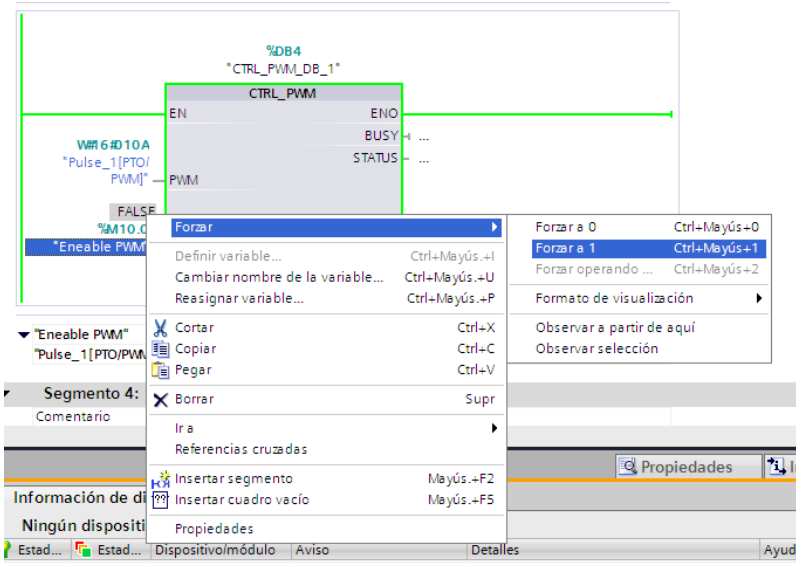


Figura36. Forzado de la entrada de habilitación

Una vez se ha forzado dicho valor a 1 la señal de impulsos se refleja en el PLC parpadeando la luz de la salida Q0.0. Esta variable no es posible de observar en el Distributed Manager de Labview.

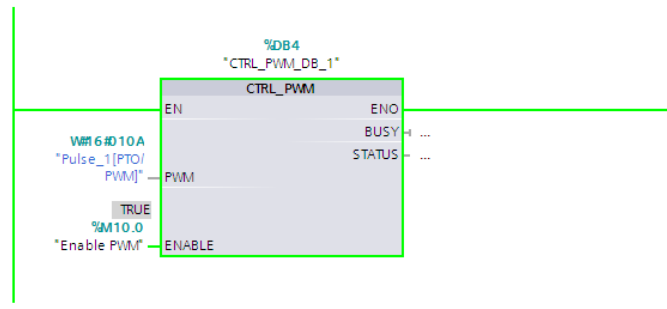


Figura37. Señal de impulsos activada

Existe la posibilidad de habilitar la señal de impulsos desde labview. Esto se consigue activando una salida desde labview, y que esta salida a su vez active la señal de impulsos:

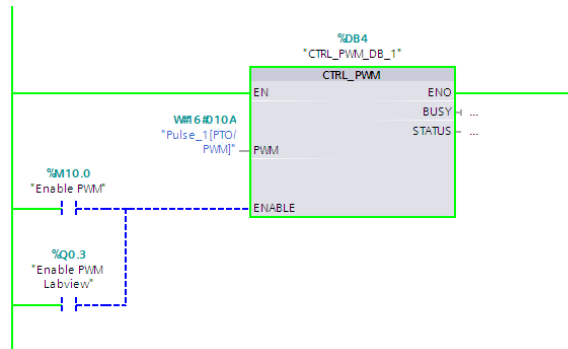


Figura38. Control señal de impulsos desde Labview

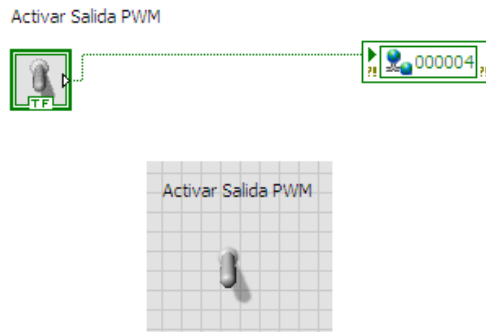


Figura39. Diagrama de bloques de Labview

comunicacionplc				
+	TF	000001	false	Read/Write
+	TF	000002	true	Read/Write
+	TF	000003	false	Read/Write
+	TF	000004	false	Read/Write
+	TF	000005	false	Read/Write
+	TF	000006	false	Read/Write

Figura40. Acceso a la variable desde Distributed Manager

Una vez se ha programado correctamente se procede a activar la salida de impulsos:

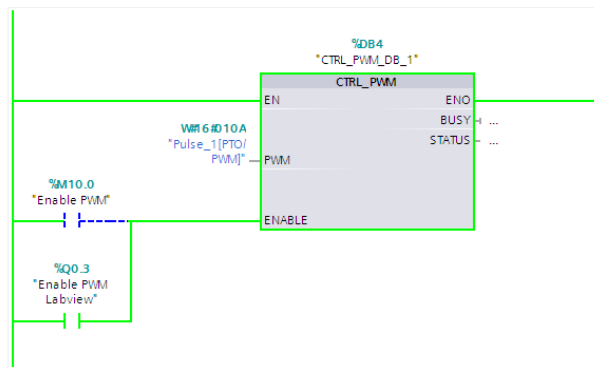


Figura41. Activar salida de impulsos desde Labview

comunicacionplc				
+	TF	000001	false	Read/Write
+	TF	000002	true	Read/Write
+	TF	000003	false	Read/Write
+	TF	000004	true	Read/Write
+	TF	000005	false	Read/Write
+	TF	000006	false	Read/Write

Figura42. Pulsador que activa la señal de impulsos desde Labview. Monitorización de la variable desde Labview

Una vez se ha conseguido la señal de impulsos en el PLC es necesario realizar cálculos de resistencias para obtener la tensión que se desea para el motor. Se llevan a cabo unos cálculos sencillos, y el divisor queda de la siguiente forma:

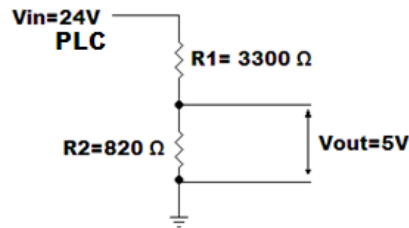


Figura43.Divisor de tensión

- **Configuración de la placa para el funcionamiento del motor**

Una vez se ha obtenido la señal de 5V se procede a alimentar el motor. El circuito integrado en la placa que alimenta al motor es el siguiente. En dicho circuito se observa que el motor está conectado mediante el microcontrolador:

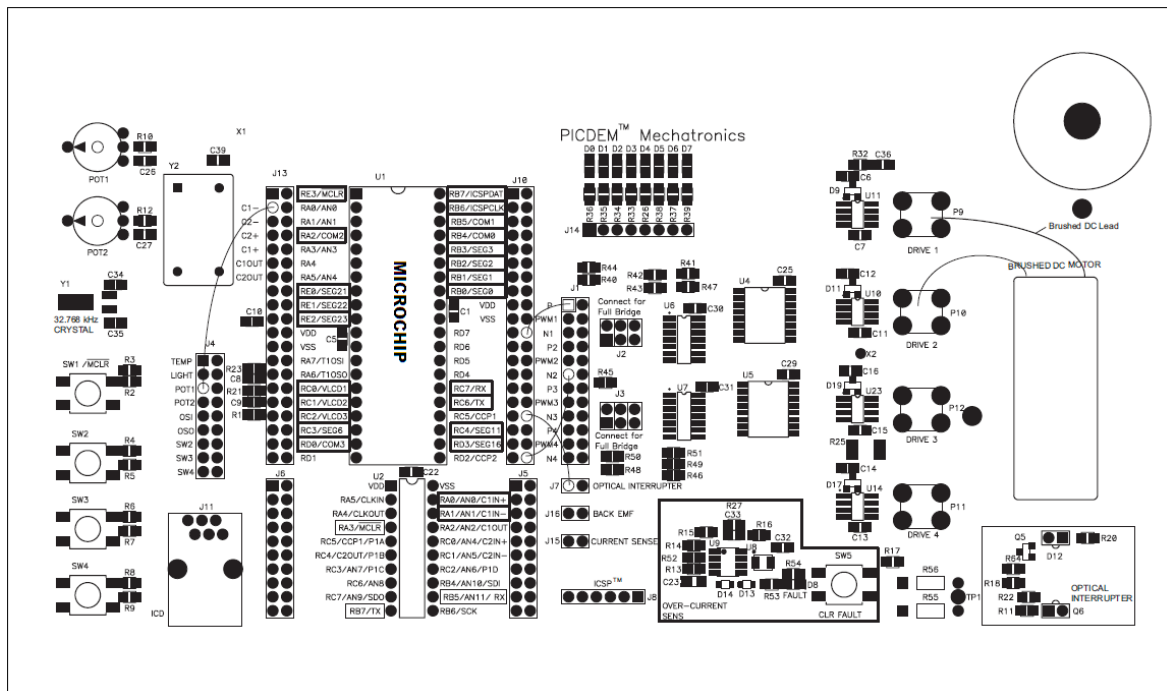


Figura44.Circuito Placa

El motor se encuentra alimentado por un puente en H:

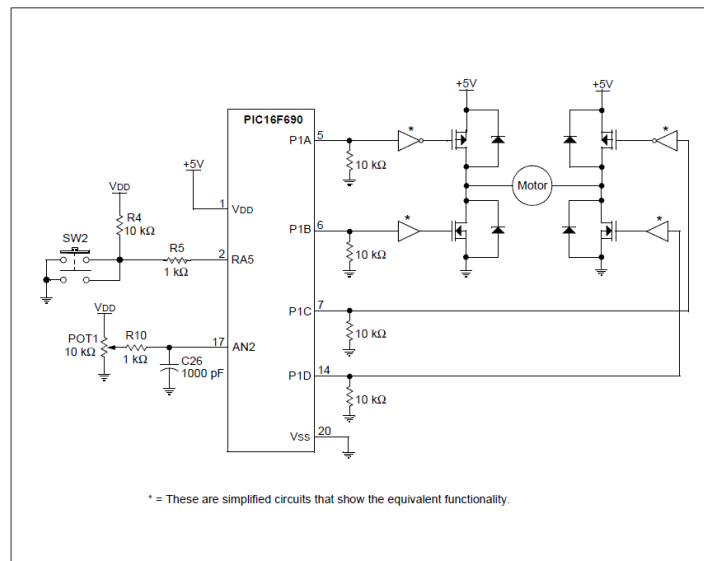


Figura45.Circuito interno de conexión del motor

Se cambia la alimentación de puente en H, es decir, en vez de ser alimentado por el microchip se alimenta mediante el divisor de tensión que se ha creado:

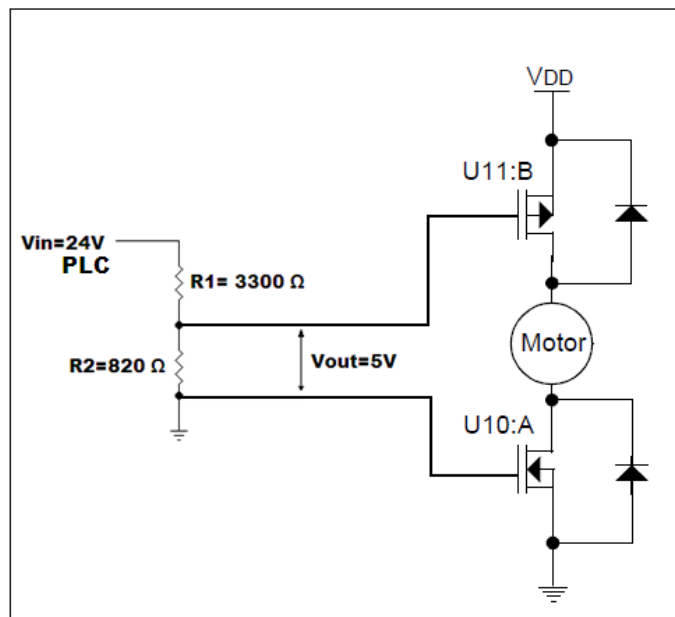


Figura46.Motor alimentado con el divisor

- **Adaptación de la señal de salida de la placa al PLC**

La señal que produce el encoder se ha de conectar al PLC para poder obtener los datos necesarios. A su vez dicha señal será accesible desde labview. El problema que se presenta es que la señal que produce el encoder es cuadrada de 5V y las entradas digitales del PLC funcionan a 24V, por lo que es necesario diseñar un circuito amplificador que pase la señal de 5V a 24V.

Este circuito se puede realizar de varias formas, para realizar la práctica se ha pensado en un circuito amplificador con un transistor. El transistor seleccionado se ha de ajustar a las necesidades que presenta el proyecto. Para este proyecto se ha utilizado el transistor BC548B, tipo NPN. El transistor tiene la siguiente forma:

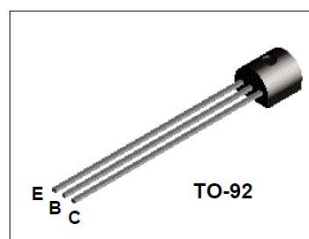


Figura47. Transistor BC548B, NPN

Se realiza el diseño del circuito con sus correspondientes cálculos. El transistor va a funcionar como un interruptor; es decir, variando entre los estados de corte o de saturación. Cuando el transistor se encuentre en corte, modo OFF, habrá entre los bornes del emisor y la base, $V_{CE}=24V$. Cuando el transistor se encuentre en saturación, modo ON, entre los bornes del emisor y la base, $V_{CE}=0V$

El circuito diseñado tiene la siguiente forma:

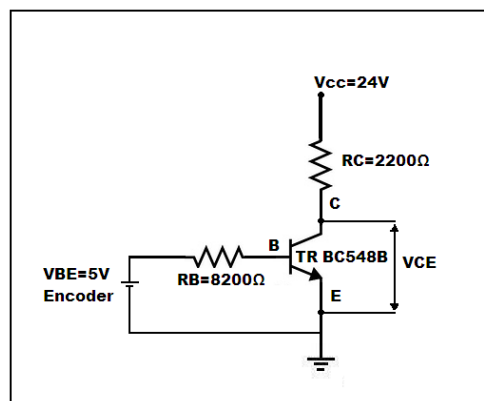


Figura48. Circuito Amplificador

Es necesario diseñar un nuevo circuito con un amplificador operacional en modo buffer, o también llamado seguidor de tensión. Para ello se utiliza el amplificador LM741 que se muestra a continuación:

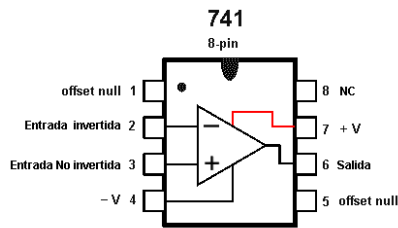


Figura49. Amplificador Operacional μ A 741

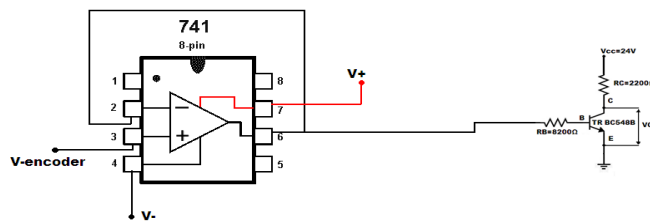


Figura50. Conexión de circuito seguidor de tensión y circuito amplificador

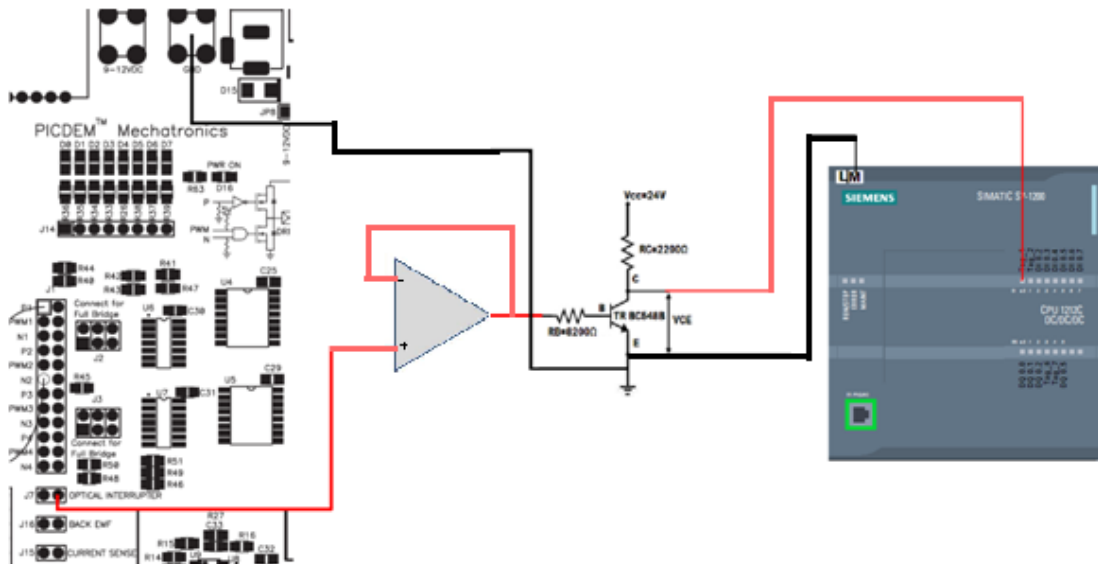


Figura51. Circuito adaptador de la señal de la placa al PLC

- **Ejercicio propuesto**

Una vez hecho todo el montaje comprobar con el PLC que se puede controlar la velocidad del motor y monitorizar con un osciloscopio la salida del encoder de la tarjeta y la señal que llega al PLC. ¿Qué diferencias hay entre ambas señales? ¿Por qué?

Sección 3. Control de la velocidad del motor

La última parte de la práctica consiste en monitorizar la velocidad de del motor desde Labview y controlar el avance del motor por vueltas. Se añaden varios pasos a seguir para que el alumno realice esta parte de la práctica:

- Aplicar una señal PWM desde el PLC a la placa, controlando la señal desde Labview
- Programar en Labview para obtener el dato de la velocidad del motor
- Variar la velocidad del motor desde Labview

Nota1: Para acceder a la variable donde se observa la velocidad es necesario acceder al cuadro donde se especifican las direcciones de las variables para conexionar el PLC con Labview. Se adjunta un documento PDF con dicho cuadro.

Nota2: Para modificar la variable que controla el número de vueltas a las que ha de girar el motor se adjunta otro documento PDF con las indicaciones necesarias.



BIBLIOGRAFÍA

Departamento de Ingeniería Eléctrica y Electrónica

Alumno: Aitziber Marín Iturrarte
Tutor: Ignacio Del Villar Fernández

5. Bibliografía

Web:

www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf

www.ni.com/swf/demos/us/labview/dscmodbus/

www.youtube.com/watch?v=wd49IsIBfqE

www.youtube.com/watch?v=SmMmnc0YTrE

<http://zone.ni.com/devzone/cda/tut/p/id/10000#toc7>

www.automation.siemens.com

www.uhu.es/raul.jimenez/SEA/ana_guia.pdf

www.icmm.csic.es/fis/gente/josemaria_albella/electronica/9%20Circuitos%20Amplificadores.pdf

www.microchip.com

Libros:

[1] W.Stallings, “Comunicaciones y Redes de Computadores” (América), (6ª Edición)

[2] Adel S. Sedra y Kenneth C. Smith, “Circuitos Microelectrónicos”, Mc Graw Hill, (5ª Edición), 2006