UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

Carrera de Ingeniería en Electrónica, Automatización y Control

“DISEÑO E IMPLEMENTACIÓN DE UN BANCO DE PRUEBAS Y ANÁLISIS DE PROTOCOLOS DE RED IEC 60870-5-104 Y MODBUS TCP DENTRO DE REDES DE DISTRIBUCIÓN ELÉCTRICA UTILIZANDO TARJETAS DE DESARROLLO”

AUTOR: GARCÍA CATOTA JONATHAN MARCELO

DIRECTOR: ING. TIPÁN CONDOLO EDGAR FERNANDO

AGENDA

Introducción

1. **Antecedentes**
2. **Justificación e Importancia**
3. **Objetivos**

Protocolos

1. **Modbus**
2. **IEC60870**

Diseño e Implementación

1. **Maestro**
2. **Esclavo**

Resultados

Conclusiones y Recomendaciones

INTRODUCCIÓN

1. Antecedentes

En cuanto a la infraestructura de control en las subestaciones eléctricas, se utiliza un Sistema Integrado de Control Distribuido (SICD), que facilita el control, la supervisión y la protección de la subestación y sus líneas de entrada y salida (Manzano, 2022). Este sistema distribuido se basa en la interconexión de diversos sistemas en red mediante Dispositivos Electrónicos Inteligentes (IED), lo que permite la integración de funciones como el control, la protección y la medición, así como la ampliación de funciones de operación como la autosupervisión, el análisis de señales y fallas, el almacenamiento de datos y eventos, entre otros (Manzano, 2022).

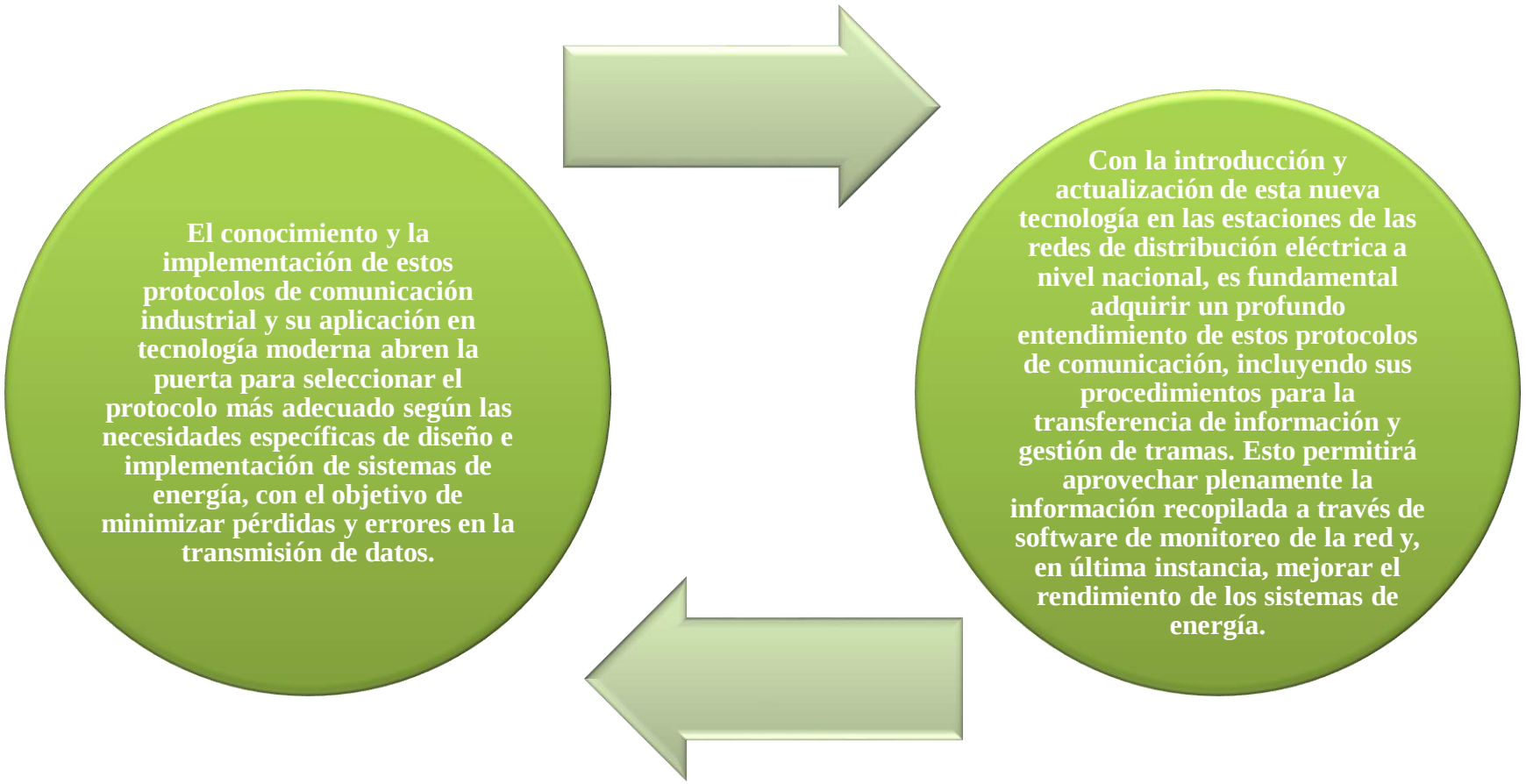
A pesar de que Modbus es implementado por la mayoría de los fabricantes, presenta limitaciones en cuanto a las funciones que puede soportar, especialmente en operaciones de supervisión y control de instalaciones de gestión de energía eléctrica, debido a su incapacidad para informar automáticamente cambios en el estado de las variables de operación (Feria et al., 2015). En respuesta a esto, se desarrolló el Modbus TCP, que permite la comunicación de equipos industriales y dispositivos físicos a través de una red Ethernet.

Además, la especificación IEC 60870-5-104 se creó utilizando estándares establecidos, clases de datos preestablecidos, procedimientos de transferencia de datos y técnicas de administración de tramas previamente definidas en las normas IEC 60870-10, IEC 60870-5-102 y IEC 60870-5-103 (International Electrotechnical Commission, 2023). Esta norma brinda un conjunto integral de funcionalidades y procedimientos que facultan la notificación de eventos relacionados con protección y control.

Los protocolos mencionados anteriormente están orientados hacia medios de comunicación más versátiles, como TCP/IP sobre Ethernet (Gallardo, 2022). Por lo tanto, es cada vez más esencial que todos los componentes de una red de distribución eléctrica estén interconectados. Es aconsejable, por lo tanto, llevar a cabo un prototipo con el fin de realizar pruebas y análisis de los protocolos de red IEC 60870-5-104 y Modbus TCP dentro de redes de distribución eléctrica, utilizando tarjetas de desarrollo para familiarizarse con estos protocolos.

INTRODUCCIÓN

2. Justificación e Importancia



INTRODUCCIÓN

3. Objetivos

Objetivo General

- Diseñar e implementar un banco de pruebas y análisis de protocolos de red IEC 60870-5-104 y Modbus TCP dentro de redes de distribución eléctrica utilizando tarjetas de desarrollo para su estudio y comprensión.

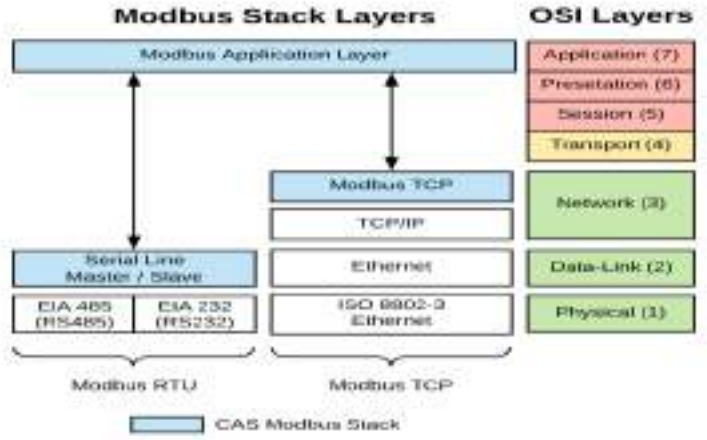
Objetivos específicos

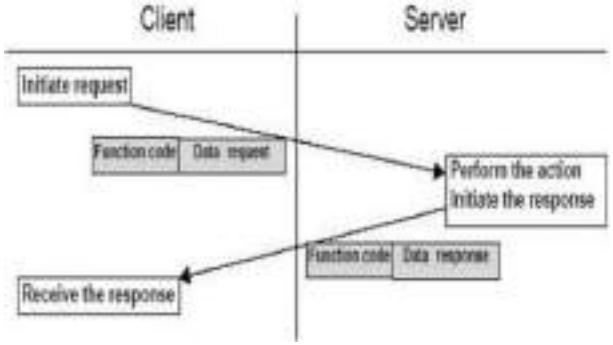
- Establecer la comunicación de los dispositivos desarrollados a partir de tarjetas de desarrollo utilizando los protocolos de red IEC 60870-5-104 o Modbus TCP.
- Investigar los protocolos de red IEC 60870-5-104 y Modbus TCP, desde su estructura, trama y hasta su comunicación a partir de comunicación maestro esclavo.
- Transmitir datos de señales analógicas y digitales entre los dispositivos para simular la comunicación y funcionamiento de una red de distribución eléctrica.

PROTOCOLOS

1. Modbus

Modbus es un protocolo de mensajería de capa de aplicación (capa 7 de OSI) que proporciona comunicación cliente/servidor entre dispositivos conectados a diferentes tipos de buses de redes. El protocolo MODBUS implementa una arquitectura cliente/servidor y opera esencialmente en un modo de 'solicitud/respuesta', independientemente del control de acceso a los medios utilizado en la capa 2.





Este modelo cliente/servidor se basa en cuatro tipos de mensajes, a saber:

- **Solicitudes MODBUS, los mensajes enviados en la red por los clientes para iniciar transacciones**
- **Confirmaciones MODBUS, los mensajes de respuesta recibidos en el lado del cliente**
- **Indicaciones MODBUS, los mensajes de solicitud recibidos en el lado del servidor**
- **Respuestas MODBUS, los mensajes de respuesta enviados por los servidores.**

El servidor (en el dispositivo esclavo) luego realiza la acción requerida e inicia una respuesta

PROTOCOLOS

1. Modbus - Estructura Modbus

Campo de dirección, que consta de un solo byte de información. En las tramas de solicitud, este byte identifica el controlador al que se dirige la solicitud.

Campo de función, también consta de un solo byte de información. En una solicitud, este byte identifica la función que debe realizar el esclavo de destino.

El campo de datos, varía en longitud según la función especificada en el campo de función. En una solicitud de host, este campo contiene información que el PLC puede necesitar para completar la función solicitada. En una respuesta de PLC, este campo contiene cualquier dato solicitado por ese host.

El campo de verificación de errores, asegura que los dispositivos no reaccionen a los mensajes que pueden haberse dañado durante la transmisión.

Campo de dirección	Campo de función	Campo de datos	Campo de comprobación de errores
1 byte	1 byte	Variable	2 bytes

PROTOCOLOS

1. Modbus - Direcciones Modicon y Códigos de Función

Códigos de función, cada trama de solicitud contiene un código de función que define la acción esperada para el controlador de destino. El significado de los campos de datos de solicitud depende del código de función especificado.

Tipo de datos	Direcciones absolutas	Direcciones relativas	Códigos de Función	Descripción
Bobinas	00001–09999	0–9998	01	Leer estado de la bobina
Bobinas	00001–09999	0–9998	05	Forzar bobina simple
Bobinas	00001–09999	0–9998	15	Forzar múltiples bobinas
Entradas discretas	10001–19999	0–9998	02	Leer estado de entrada
Registros de entrada	30001–39999	0–9998	04	Leer registros de entrada
Registros de espera	40001–49999	0–9998	03	Leer registro de espera
Registros de retención	40001–49999	0–9998	06	Registro único preestablecido
Registros de retención	40001–49999	0–9998	16	Registros múltiples preestablecidos
-	-	-	07	Estado de excepción de lectura
-	-	-	08	Prueba de diagnóstico de bucle invertido

PROTOCOLOS

1. Modbus - Códigos de Función Principales

Leer estado de bobina o salida digital (código de función 01)

Esta función permite que el host obtenga el estado ON/OFF de una o más bobinas lógicas en el dispositivo de destino. Ejemplo: el host solicita el estado de las bobinas 000A (00011 decimal) y 000B (00012 decimal). La respuesta del dispositivo objetivo indica que ambas bobinas están encendidas.

Request Message

Address	Function Code	Initial Coil Offset		Number of Points		CRC
		Hi	Lo	Hi	Lo	
01	01	00	0A	00	02	9D C9

Response Frame

Address	Function Code	Byte Count	Coil Data	CRC
01	01	01	03	11 89

Request Message

Address	Function Code	Initial Coil Offset		Number of Points		CRC
		Hi	Lo	Hi	Lo	
01	02	00	00	00	02	F9 CB

Response Frame

Address	Function Code	Byte Count	Input Data	CRC
01	02	01	02	20 49

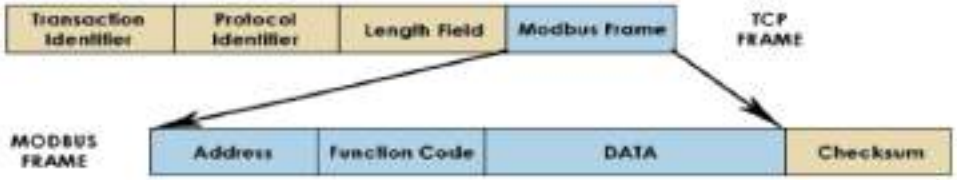
Leer estado de entrada digital (código de función 02). Esta función permite que el host lea una o más entradas discretas en el dispositivo de destino. En el ejemplo, el host solicita el estado de las entradas discretas relativas 0001 y 0002 hexadecimales, es decir, decimal las absolutas 10000 y 10001. La respuesta del dispositivo de destino indica que la entrada discreta 10000 está APAGADA y 10001 están ENCENDIDA.

PROTOCOLOS

1. Modbus – Modbus TCP

Para la comunicación utiliza el puerto 502 y su estructura muestra de forma genérica la trama del protocolo
ID: Dirección del esclavo, número enteros normalmente desde 1, hasta 255.
FUNCIÓN: Tipo de solicitud que se le realiza al esclavo, codificada de manera numérica, normalmente números que inician en 1,2,3,4, etc.
DATO: Este campo se emplea para enviar información complementaria a la solicitud realizada al esclavo, o se emplea para responder a la solicitud.

Para el maestro el campo dato está integrado por dos subcampos, la dirección y la longitud.



Respuesta

Nombre	Bytes	Función
Identificador de transacciones	0	Para sincronización
Identificador de transacciones	1	
Identificador de protocolo	2	Cero para Modbus/TCP
Identificador de protocolo	3	
Longitud	4	Número de bytes restantes en esta trama
Longitud	5	
Estación	6	Dirección de la estación
Función	7	Código de función
Bytes e datos	8	Bytes de datos
Datos	9...	Datos

Solicitud

Nombre	Bytes	Función
Identificador de transacciones	0	Para sincronización
Identificador de transacciones	1	
Identificador de protocolo	2	Cero para Modbus/TCP
Identificador de protocolo	3	
Longitud	4	Número de bytes restantes en esta trama
Longitud	5	
Estación	6	Dirección de la estación
Función	7	Código de función
Dirección MSB	8	Dirección inicial
Dirección LSB	9	
Cantidad MSB	10	Cantidad
Cantidad LSB	11	

La estructura sigue siendo la misma para el esclavo, pero se produce un cambio en el campo de Dato, añadiendo dos subcampos: el número de bytes suficientes para proporcionar la respuesta y la propia respuesta.

PROTOCOLOS

1. Modbus – Ventajas Modbus TCP

- **La expansión y la gestión son sencillas. El agregar Esclavos a la red Modbus, no implica la utilización de recursos de software ni hardware sofisticados.**
- **No se necesita de softwares o hardware propio de monitoreo, supervisión o configuración. Modbus/TCP se puede emplear en cualquier computadora que utilice o tenga un software compatible con protocolos TCP/IP.**
- **Dispone en el mercado de diferentes equipos de conversión o puentes para Modbus TCP, los mismos que no necesitan configuración.**
- **Su simplicidad y tamaño de trama lo hace muy rápido, por lo que puede alcanzar altas tasas de transmisión, garantizando tiempos de respuesta en milisegundos, lo que se considera tiempo real.**
- **El ingeniero de mantenimiento puede acceder, de manera inmediata, al sistema de control de la planta desde el confort de su hogar.**
- **Permite administrar sistemas distribuidos geográficamente utilizando las tecnologías de Internet e Intranet actuales.**

PROTOCOLOS

2. IEC60870-5-104

La norma IEC 60870-5-104 aplica los conceptos de telecontrol definidos en la norma IEC 60870-5-101 eliminando las cabeceras de transmisión serie y añadiendo cabeceras con información apropiada para la gestión de su envío por canales TCP-IP. Se transporta sobre redes LAN utilizando TCP/IP en las capas de 1 a 4. Su utilización es en telecontrol.

En realidad el protocolo IEC-104 es una extensión del protocolo IEC-101, por tanto los dos se parecen en su estructura, lo que se incorpora a la estructura del IEC-104 es un adicional que es el proceso de usuario.

La capa de aplicación IEC 104 se conserva igual a la de IEC 101, manteniéndose sin cambios la capa de aplicación donde se conserva el ASDU (Unidad de Aplicación de Datos), quien contiene la información de las variables del proceso; en este caso el ASDU también funciona sobre la plataforma TCP/IP, como se había mencionado el TCP está orientado para transmitir datos de una forma segura

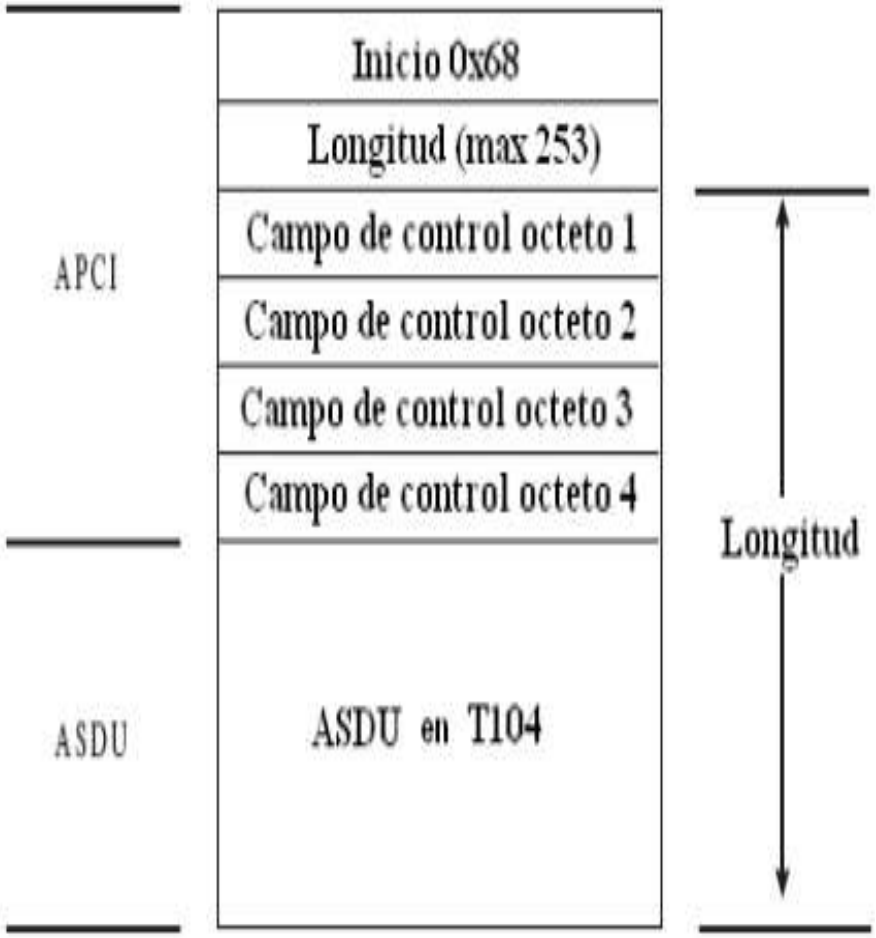
7. Aplicación 6. Presentación 5. Sesión	APDU
4. Transporte	TCP
3. Red	IP
2. Enlace	Transmisión de Ip por ethernet
1. Física	Ethernet

PROTOCOLOS

2. IEC60870-5-104 – Formato de la trama (APCI)

Los dos primeros bits del campo de control del APCI nos permiten diferenciar 3 tipos de tramas en 104:

- **U frame: son tramas de control que determinan si se puede enviar tráfico o no por el canal TCP por medio de mensajes START y STOP y que permiten comprobar el estado del mismo en caso de que no hubiere tráfico con mensajes TEST.**
- **I frame: son tramas que transportan datos de aplicación (ASDUS).**
- **S frame: son tramas de supervisión que indican al extremo opuesto la recepción correcta de una o varias tramas de tipo I pendientes de reconocer.**

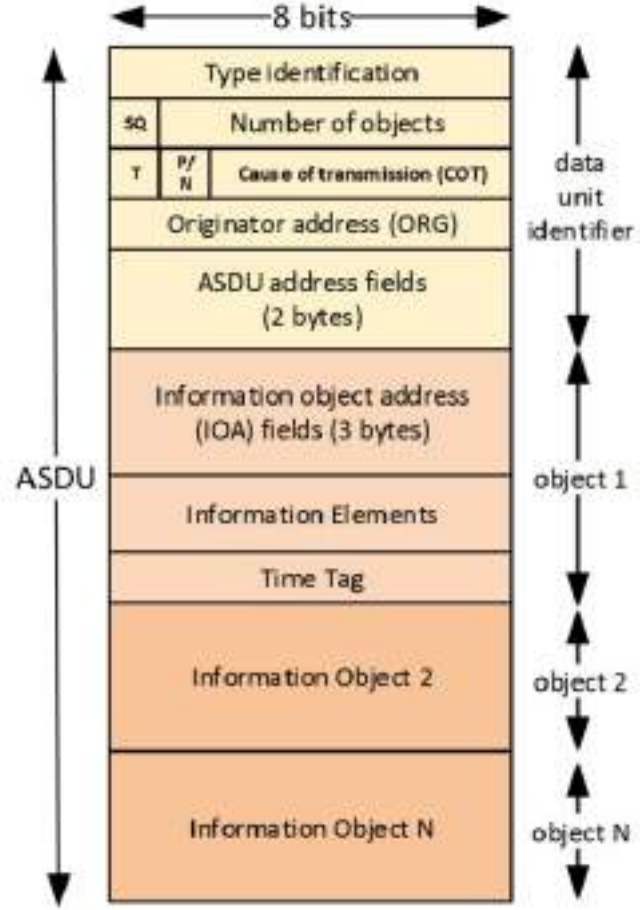


PROTOCOLOS

2. IEC60870-5-104 – Formato de la trama (ASDU)

La estructura de un ASDU, ésta consta de dos secciones principales que son:

- **El identificador de unidad, que define el tipo de dato.**
- **El objeto de información, que es donde se almacena el dato de la variable.**
- **Cada bloque es un byte, es decir el campo causa de transmisión ocupa 2 bytes (en el segundo byte se encuentra la dirección origen), el campo dirección común de la ASDU ocupa 2 bytes y el campo dirección del objeto de información ocupa 3 bytes. Estos 3 bytes se presentan estructurados y se limitan las direcciones a 65535.**



PROTOCOLOS

Tipos de ASDU: Información del proceso en la dirección de monitoreo

N° de tipo	Descripción	Referencia
<30>	Información de un solo punto con etiqueta de tiempo CP56Tiempo2a	M_SP_TB_1
<31>	Información de doble punto con etiqueta de tiempo CP56Tiempo2a	M_DP_TB_1
<32>	Información de posición de paso con etiqueta de tiempo CP56Tiempo2a	M_ST_TB_1
<33>	Cadena de bits de 32 bits con etiqueta de tiempo CP56Tiempo2a	M_BO_TB_1
<34>	Valor medido, valor normalizado con etiqueta de tiempo CP56Time2a	M_ME_TD_1
<35>	Valor medido, valor escalado con el tiempo etiqueta CP56Time2a	M_ME_TE_1
<36>	Valor medido, punto flotante corto número con etiqueta de tiempo CP56Time2a	M_ME_TF_1
<37>	Totales integrados con etiqueta de tiempo CP56Tiempo2a	M_IT_TB_1
<38>	Evento de equipos de protección con etiqueta de tiempo CP56Time2a	M_EP_TD_1
<39>	Eventos de inicio empaquetados de protección equipo con etiqueta de tiempo CP56Time2a	M_EP_TE_1
<40>	Información empaquetada del circuito de salida de equipo de protección con etiqueta de tiempo CP56Tiempo2a	M_EP_TF_1
<41..44>	Reservado para más definiciones compatibles	

Tipo No.	Descripción	Referencia
<0>	No definido	
<1>	Información de punto único	M_SP_NA_1
<2>	Información de punto único con etiqueta de tiempo	M_SP_TA_1
<3>	Información de doble punto	M_DP_NA_1
<4>	Información de punto doble con etiqueta de tiempo	M_DP_TA_1
<5>	Información de posición de paso	M_ST_NA_1
<6>	Información de posición de paso con etiqueta de tiempo	M_ST_TA_1
<7>	Cadena de bits de 32 bits	M_BO_NA_1
<8>	Cadena de bits de 32 bits con etiqueta de tiempo	M_BO_TA_1
<9>	Valor medido, valor normalizado	M_ME_NA_1
<10>	Valor medido, valor normalizado con etiqueta de tiempo	M_ME_TA_1
<11>	Valor medido, valor escalado	M_ME_NB_1
<12>	Valor medido, valor escalado con etiqueta de tiempo	M_ME_TB_1
<13>	Valor medido, número de punto flotante corto	M_ME_NC_1
<14>	Valor medido, número de punto flotante corto con etiqueta de tiempo	M_ME_TC_1
<15>	Totales integrados	M_IT_NA_1
<16>	Totales integrados con etiqueta de tiempo	M_IT_TA_1
<17>	Evento de equipo de protección con etiqueta de tiempo	M_EP_TA_1
<18>	Eventos de inicio empaquetados de equipos de protección con etiqueta de tiempo	M_EP_TB_1
<19>	Información empaquetada del circuito de salida del equipo de protección con etiqueta de tiempo	M_EP_TC_1
<20>	Información empaquetada de un solo punto con detección de cambio de estado	M_PS_NA_1
<21>	Valor medido, valor normalizado sin descriptor de calidad	M_ME_ND_1
<22..29>	Reservado para más definiciones compatibles	

PROTOCOLOS

Tipos de ASDU: Información del proceso en la dirección de control

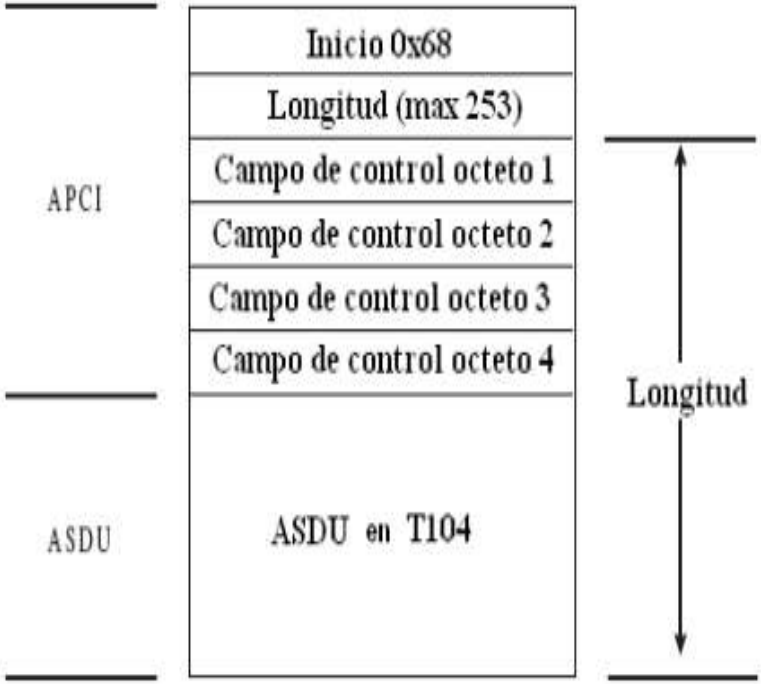
N.° de tipo	Descripción	Referencia
<45>	Comando único	C_SC_NA_1
<46>	Comando doble	C_DC_NA_1
<47>	Mando paso de regulación	C_RC_NA_1
<48>	Comando de consigna, valor normalizado	C_SE_NA_1
<49>	Comando de consigna, valor escalado	C_SE_NB_1
<50>	Comando de punto de ajuste, número de punto flotante corto	C_SE_NC_1
<51>	Cadena de bits de 32 bits	C_BO_NA_1

N.° de tipo	Descripción	Referencia
<100>	Comando de interrogación	C_IC_NA_1
<101>	Comando de interrogación de contador	C_CI_NA_1
<102>	Comando de lectura	C_RD_NA_1
<103>	Comando de sincronización de reloj	C_CS_NA_1
<104>	Comando de prueba	C_TS_NA_1
<105>	Restablecer comando de proceso	C_RP_NA_1
<106>	Comando de adquisición de retardo	C_CD_NA_1
<107..109>	Reservado para más definiciones compatibles	

PROTOCOLOS

2. IEC60870-5-104 – Formato de la trama (APDU)

La capa de aplicación realiza la transmisión de tramas denominadas APDU. APDU significa "unidad de datos del protocolo de aplicación" y es la unidad de información que es entregada desde o hacia la capa de aplicación. El APDU consta de una cabecera que contiene información de control denominada APCI (información de control del protocolo de aplicación), además de la información de los elementos de campo (variables analógicas y digitales), denominada ASDU (unidad de datos de servicios de aplicación).

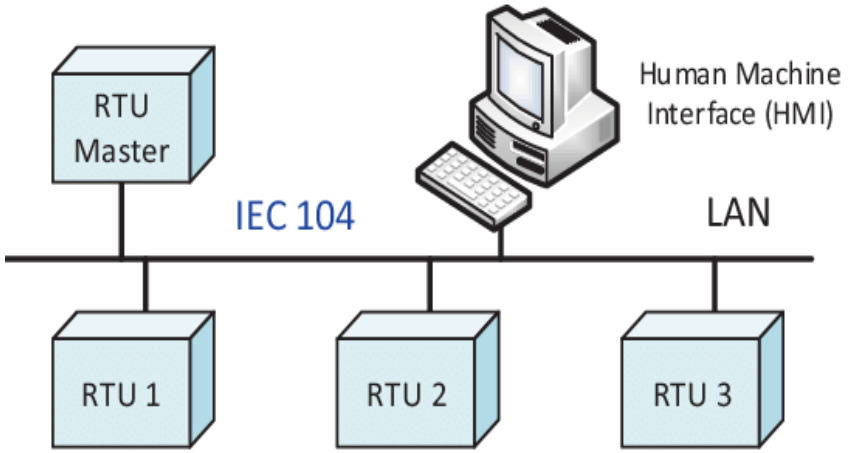


PROTOCOLOS

2. IEC60870-5-104 – Funcionamiento

El funcionamiento del protocolo IEC 104 es el siguiente:

- **La estación de control, comúnmente con rol maestro, es la encargada de iniciar las comunicaciones o enviar comandos a dispositivos o aplicaciones; estos datos transmitidos van en una sola dirección, llamada de control.**
- **Las subestaciones o estaciones controladas serán los dispositivos remotos que transmiten los datos recogidos a las estaciones de control. Esta dirección de envío se conoce por el nombre de supervisión. Al ser dependientes de la estación de control, reciben el nombre de esclavos.**



PROTOCOLOS

2. IEC60870-5-104 – Elementos de Información

En las siguientes definiciones, deben tenerse en cuenta estas reglas de interpretación:

- **Las descripciones de los bits dan el estado lógico para un bit establecido, es decir, bit = 1.**
- **Las posiciones de bits en blanco están reservadas y deben borrarse, es decir, bit = 0.**

Las posiciones de bit se han numerado <0..7>.

Símbolo de tipo general	Descripción
Proceso	
SIQ	Información de punto único con descriptor de calidad
DIQ	Información de doble punto con descriptor de calidad
BSI	Información de estado binario
SCD	Detección de estados y cambios
QDS	Descriptor de calidad
VTI	Valor con indicación de estado transitorio
NVA	Valor normalizado
SVA	Valor escalado
R32-IEEE STD 754	Número de punto flotante corto
BCR	Lectura de contador binario
Protección	
SEP	Evento único de equipos de protección
SPE	Eventos de inicio de equipos de protección
OCI	Información del circuito de salida del equipo de protección
QDP	Descriptor de calidad para eventos de equipos de protección
Comandos	
SCO	Comando único
DCO	Doble mando
RCO	Comando paso de regulación
Tiempo	
CP56Time2a	Tiempo binario de siete octetos
CP24Time 2ª	Tiempo binario de tres octetos
CP16Time2a	Tiempo binario de dos octetos
Calificadores	
QOI	Calificador de interrogación
QCC	Calificador de comando de contrainterrogación
QPM	Calificador de parámetro de valores medidos
QPA	Calificador de activación de parámetros
QRP	Calificador del comando de proceso de descanso
QOC	Calificador de comando
QOS	Calificador del comando de punto de ajuste
Transferencia de archivos	
FRQ	Calificador de archivo listo
SRQ	Calificador de sección lista
SCQ	Calificador de selección y llamada
LSQ	Calificador de última sección o segmento
AFQ	Acuse de recibo calificador de archivo o sección
NOF	Nombre del archivo
NOS	Nombre de la sección
LOF	Longitud del archivo o sección
LOS	Longitud del segmento
CHS	Suma de verificación
SOF	Estado del expediente
Miscelánea	
COI	Causa de inicialización
FBP	Patrón de bits de prueba fijo, dos octetos

DISEÑO

Estación Maestro Modbus TCP

Librería Ethernet: Clases para operar con el módulo Ethernet, incluidas las siguientes:

- ***Ethernet; permite* inicializar la librería E*thernet* y configura la red.**
- ***EthernetServer*; permite recibir y enviar datos a un cliente conectado y este puede ser un navegador web o una aplicación móvil.**
- ***EthernetClient*; es empleada cuando la tarjeta de desarrollo funciona como cliente de otro servidor o cuando un servidor recibe.**

Librería ModbusTCP. La librería Modbus TCP, para el Shield Ethernet, es muy similar a la librería Modbus compatible con el Shield ENC29J60.

Para la comunicación Modbus, como maestro, la librería tiene la instancia ModbusMaster, misma que contiene las siguientes instancias:

- **void frameHreg(word offset, word value), para envío de datos en registros.**
- **void frameCoil(word offset, bool value); para envío de datos en bobinas**

DISEÑO

Estación Maestro IEC60870-5-104

Librería IEC60870-5-104. La librería implementa la comunicación a través del protocolo IEC60870-5-104 utilizando tarjetas de desarrollo Arduino, se puede utilizar múltiples instancias de objetos maestro y esclavo para conectar Arduino con PLC, RTU u otros ARDUINOS que utilicen este protocolo. Esta librería está limitada a los siguientes comandos determinados en la estructura de comunicación del protocolo.

- **M_SP_NA_1; Información de punto único.**
- **M_DP_NA_1; Información de punto doble.**
- **M_ME_NA_1; Valor medido, valor normalizado.**
- **C_SC_NA_1; Comando único.**
- **C_DC_NA_1; Comando doble.**
- **C_IC_NA_1; Comando de interrogación.**
- **M_EI_NA_1; Fin de inicialización.**

DISEÑO

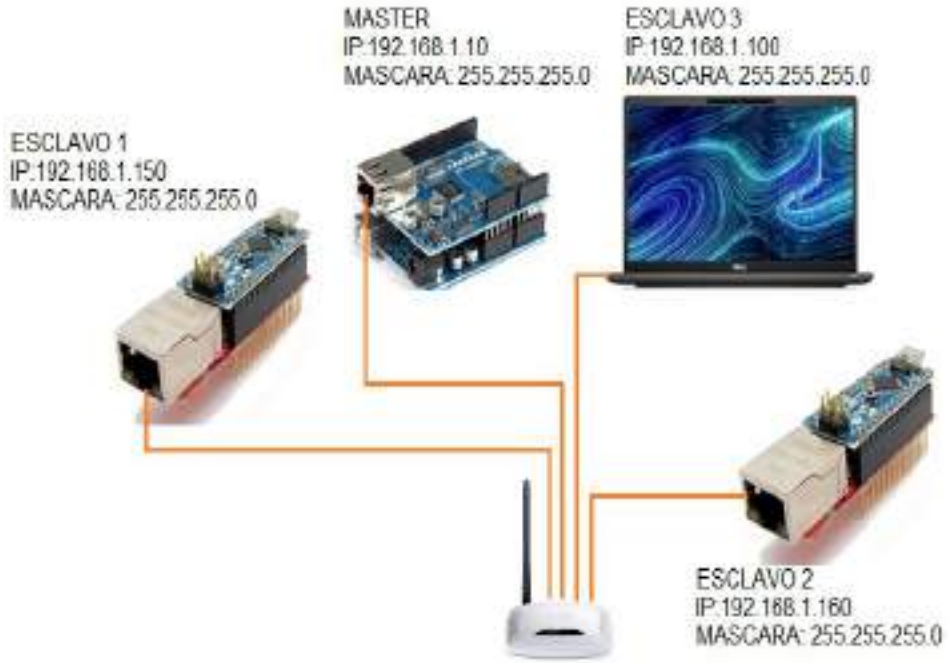
Estaciones Esclavos Modbus e IEC60870-5-104

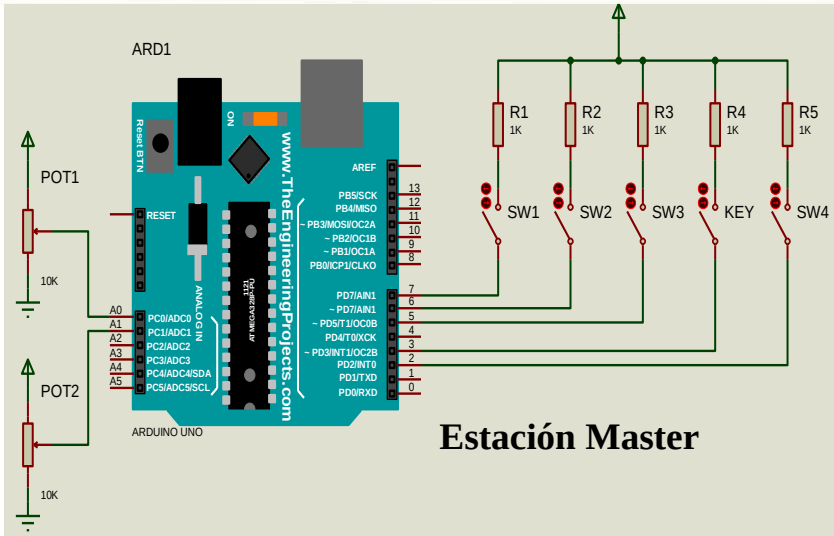
- **Librería EtherCard. Es un controlador para el chip Microchip ENC28J60, compatible con Arduino IDE. En la misma se proporcionan rutinas de alto nivel para permitir una variedad de propósitos, incluida la transferencia simple de datos hasta el manejo HTTP. Esta librería sólo es compatible con el hardware basado en el chip ENC28J60. Además, utiliza la interfaz SPI del microcontrolador y necesita de un pin dedicado para CS, además de los pines SO, SI y SCK de la interfaz SPI.**
- **Librería Modbus-Ethernet para ENC28J60. Esta librería que permite que Arduino se comunique mediante protocolo Modbus, actuando como esclavo utilizando una comunicación TCP/IP para el shield Ethernet ENC28J60. En este caso el encabezado para la utilización de la librería, necesita de la librería Ethercard.h y Modbus.h**

Implementación

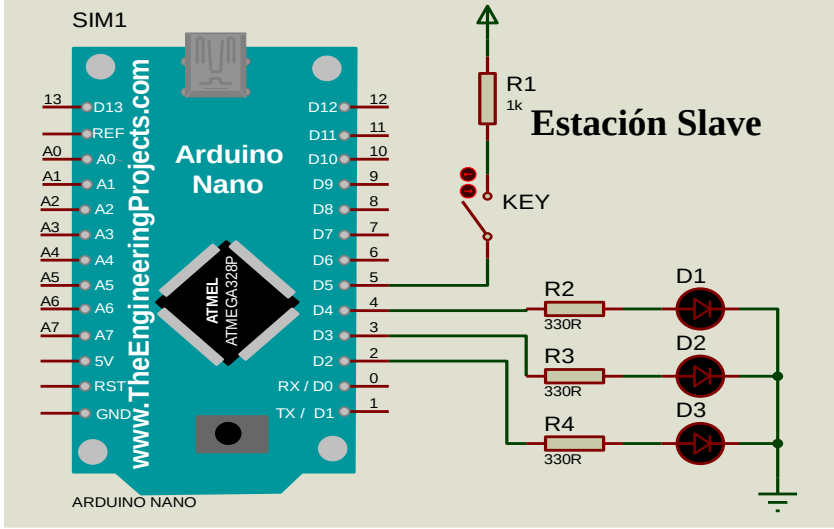
Topología y Conexiones

Topología en estrella





Estación Master



Estación Slave

Implementación

Programación Estación Maestro

```
#include <SPI.h>
#include <Ethernet.h>
#include "Modbus.h"
#include "ModbusIP2.h"
#include "IEC104.h"
#include "IEC104IP.h"


// Direcciones IP de los clientes
byte dst1[] = { 192, 168, 1, 150 }; //ESCLAVO1
byte dst2[] = { 192, 168, 1, 160 }; //ESCLAVO2
byte dst3[] = { 192, 168, 1, 100 }; //ESCLAVO3 (PC)


// direcciones de los interruptores que seran transmitidas
const int SWITCH1_ISTS = 1;
const int SWITCH2_ISTS = 2;

// direcciones de los potenciometros que seran transmitidos
const int POT1_IREG = 1;
const int POT2_IREG = 2;
```

```
// asignación de pines de entrada del arduino
const int switchPin1 = 2;
const int switchPin2 = 5;
const int switchPin3 = 6;
const int switchPin4 = 7;
const int switchPin5 = 3;

// asignación de entradas analógicas del arduino
const int PotPin1 = A0;
const int PotPin2 = A1;

// declaración de objetos de comunicación modbus, uno para cada cliente
ModbusMasterIP mbm1;
ModbusMasterIP mbm2;
ModbusMasterIP mbm3;

IEC104MasterIP iec1;
IEC104MasterIP iec2;
IEC104MasterIP iec3;
```

```
void setup() {
    pinMode(switchPin1, INPUT);
    pinMode(switchPin2, INPUT);
    pinMode(switchPin3, INPUT);
    pinMode(switchPin4, INPUT);
    pinMode(switchPin5, INPUT);
    // Dirección MAC Ethernet del shield
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEE };
    // Dirección IP del shield
    byte ip[] = { 192, 168, 1, 10 };
    //Inicialización de la comunicación Ethernet
    Ethernet.begin(mac, ip);
    // Inicialización de las comunicaciones Modbus con los clientes
    mbm1.config();
    mbm2.config();
    mbm3.config();

    iec1.config();
    iec2.config();
    iec3.config();
}
```

Implementación

Programación Estación Maestro

```
67  void loop() {
68    if(switchPin5){  //MODBUS
69    //Envio de datos al Cliente 1
70      mbm1.sendCoil(dst1, SWITCH1_ISTS, digitalRead(switchPin1));
71      mbm1.sendCoil(dst1, SWITCH2_ISTS, digitalRead(switchPin2));
72      mbm1.sendHreg(dst1, POT1_IREG, analogRead(PotPin1));
73    //Envio de datos al Cliente 2
74      mbm2.sendCoil(dst2, SWITCH1_ISTS, digitalRead(switchPin3));
75      mbm2.sendCoil(dst2, SWITCH2_ISTS, digitalRead(switchPin4));
76      mbm2.sendHreg(dst2, POT1_IREG, analogRead(PotPin2));
77      delay(100);
78    //Envio de datos al Cliente 3
79      mbm3.sendCoil(dst3, SWITCH1_ISTS, digitalRead(switchPin1));
80      mbm3.sendCoil(dst3, SWITCH2_ISTS, digitalRead(switchPin2));
81      mbm3.sendHreg(dst3, POT1_IREG, analogRead(PotPin2));
82      delay(100);
83      }
84    else{//IEC
85        //Envio de datos al Cliente 1
86      iec1.sendCoil(dst1, SWITCH1_ISTS, digitalRead(switchPin1));
87      iec1.sendCoil(dst1, SWITCH2_ISTS, digitalRead(switchPin2));
88      iec1.sendHreg(dst1, POT1_IREG, analogRead(PotPin1));
89    //Envio de datos al Cliente 2
90      iec2.sendCoil(dst2, SWITCH1_ISTS, digitalRead(switchPin3));
91      iec2.sendCoil(dst2, SWITCH2_ISTS, digitalRead(switchPin4));
92      iec2.sendHreg(dst2, POT1_IREG, analogRead(PotPin2));
93      delay(100);
94    //Envio de datos al Cliente 3
95      iec3.sendCoil(dst3, SWITCH1_ISTS, digitalRead(switchPin1));
96      iec3.sendCoil(dst3, SWITCH2_ISTS, digitalRead(switchPin2));
97      iec3.sendHreg(dst3, POT1_IREG, analogRead(PotPin2));
98      delay(100);
99  }
100   }
```

Implementación

Programación Estaciones Esclavo

```
#include <EtherCard.h>
#include "Modbus.h"
#include "ModbusIP_ENC28J60.h"
#include "IEC104.h"
#include "IEC104IP.h"

const int LAMP1_COIL = 1;
const int LAMP2_COIL = 2;
const int POT_HREG = 1;

// // asignación de entrada y salidas del arduino
const int ledPin1 = 2;
const int ledPin2 = 3;
const int ledPin3 = 4;
const int switchPin1 = 5;
int ledState = LOW;  // ledState used to set the LED
unsigned long currentMillis = 0;
unsigned long previousMillis = 0;  // will store last time LED was updated

// Objeto ModbusIP
ModbusIP mb;

// Objeto IEC104IP
IEC104IP iec;

void setup() {
//dirección MAC
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF,0xFE, 0xED};
//dirección IP
byte ip [] = {192,168,1,168};
mb.config (mac,ip);// se confirma las direcciones
iec.config (mac,ip);
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
pinMode(switchPin1, INPUT);
mb.addCoil(LAMP1_COIL);
mb.addCoil(LAMP2_COIL);
mb.addHreg(POT_HREG);
iec.addCoil(LAMP1_COIL);
iec.addCoil(LAMP2_COIL);
iec.addHreg(POT_HREG);
}
```

Implementación

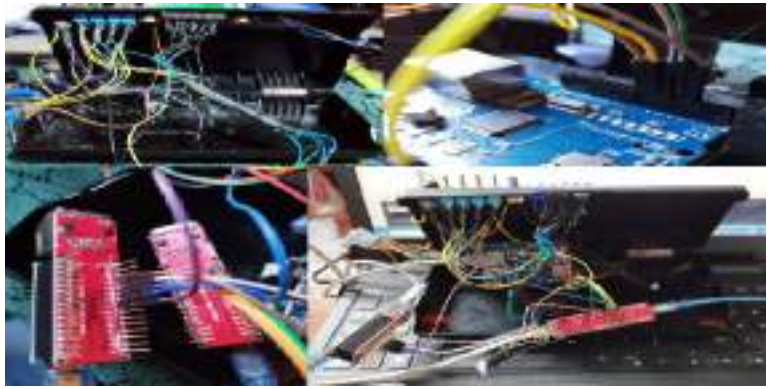
Programación Estaciones Esclavo

```
void loop() {

  if(digitalRead(switchPin1))//MODBUS
  {
    mb.task();// actualiza para que el trabajo se realize con normalidad
    // envia los valores del potenciómetro
    digitalWrite(ledPin1, mb.Coil(LAMP1_COIL));
    digitalWrite(ledPin2, mb.Coil(LAMP2_COIL));
    currentMillis = millis();
    if (currentMillis - previousMillis >= mb.Hreg(POT_HREG)) {
      previousMillis = currentMillis;
      if (ledState == LOW) {
        ledState = HIGH;
      } else {
        ledState = LOW;
      }
      digitalWrite(ledPins, ledState);
    }
  }
  else{IEC

    iec.task();// actualiza para el trabajo se realize con normalidad
    if(iec.Coil(LAMP1_COIL)==1)
    {
      digitalWrite(ledPin1, LOW);
    }
    else
    {
      digitalWrite(ledPin1, HIGH);
    }
    if(iec.Coil(LAMP2_COIL)==1)
    {
      digitalWrite(ledPin2, LOW);
    }
    else
    {
      digitalWrite(ledPin2, HIGH);
    }
    currentMillis = millis();
    if (currentMillis - previousMillis >= iec.Hreg(POT_HREG)) {
      previousMillis = currentMillis;
      if (ledState == HIGH) {
        ledState = LOW;
      } else {
        ledState = HIGH;
      }
      digitalWrite(ledPin3, ledState);
    }
  }
}
```

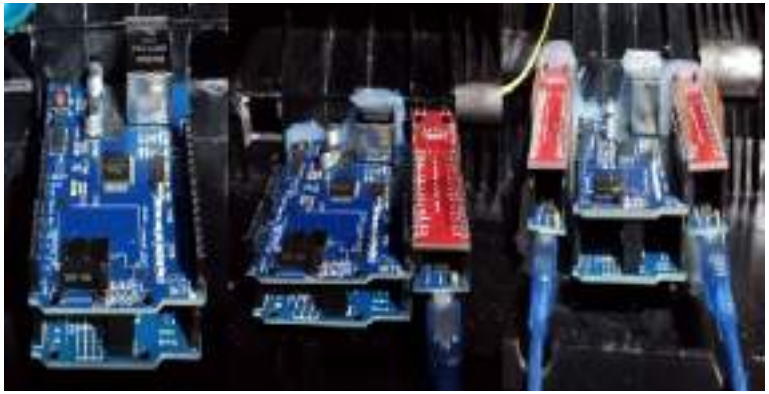
Implementación

Banco de pruebas

Adecuación



Cableado y Conexión



Instalación

Detalles

PRUEBAS

Comunicación Modbus TCP e IEC60870-5-104

Modbus TCP

IEC60870-5-104

PRUEBAS

Comunicación Sistema completo

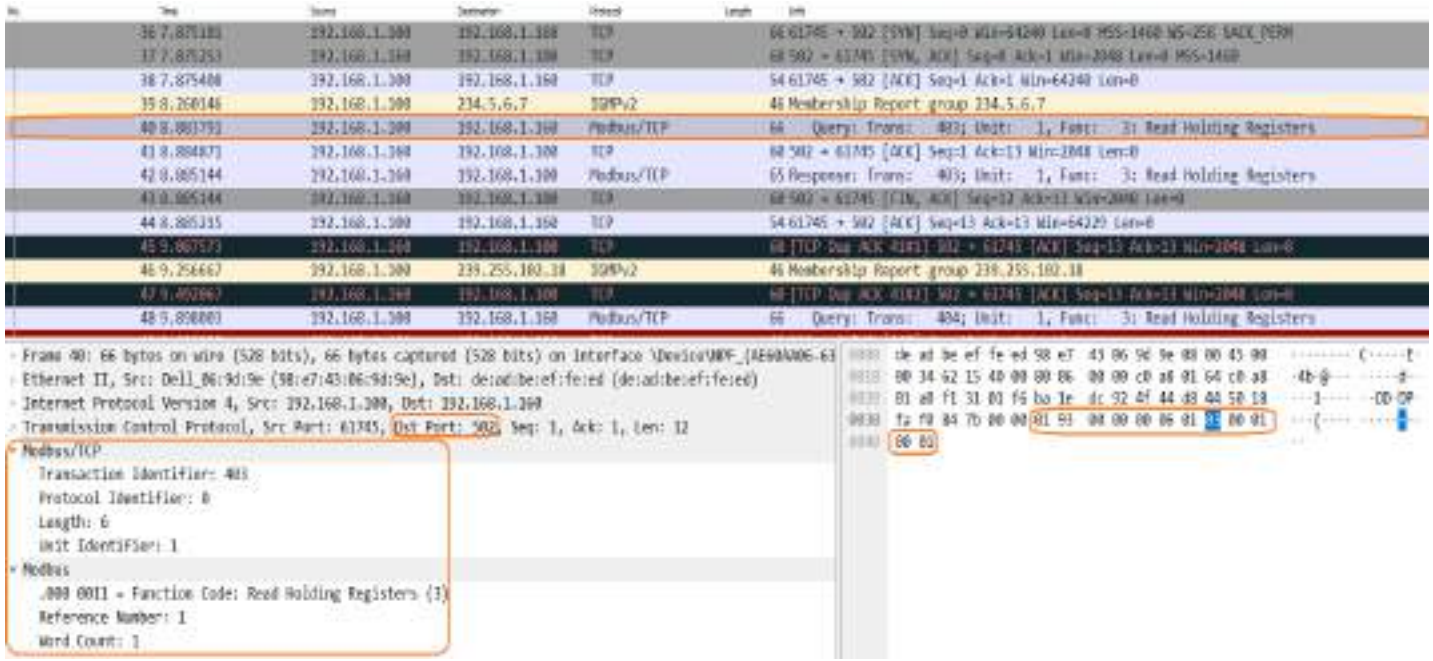
Modbus TCP

IEC60870-5-104

PRUEBAS

Monitoreo de red en Wireshark

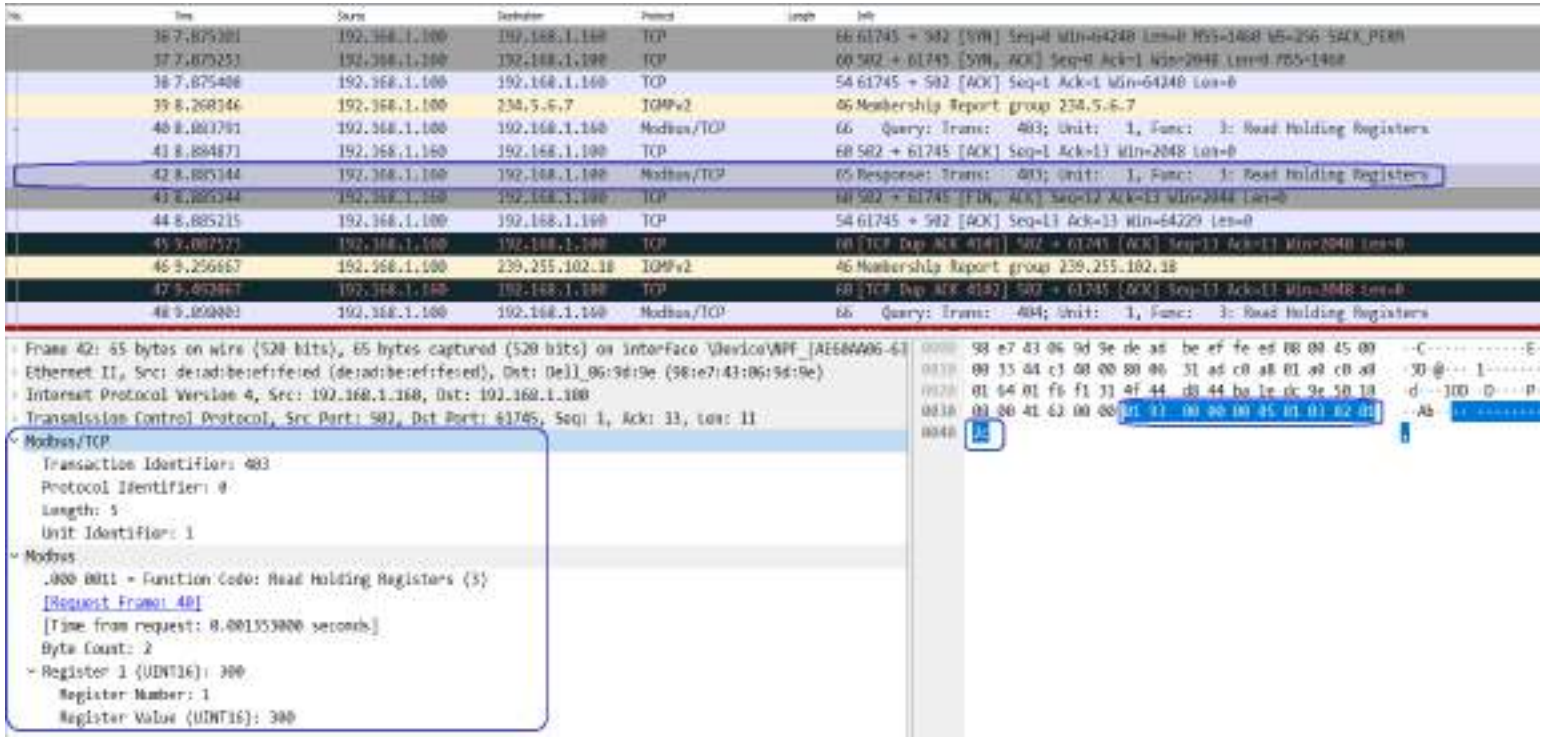
Verificación de tramas esclavo 3 (PC) - esclavo 2



PRUEBAS

Monitoreo de red en Wireshark

Verificación de tramas esclavo 2 – esclavo 3 (PC)



CONCLUSIONES

- **En base a lo desarrollado en el presente proyecto se concluye que el banco de pruebas es 100% funcional ya que cumple con los requisitos propuestos inicialmente permitiendo la comunicación entre las tarjetas de desarrollo por intermedio de los protocolos de red Modbus TCP e IEC60870-5-104.**
- **El diseño del banco de pruebas permite al usuario realizar con facilidad las pruebas de comunicación al manipular los interruptores y potenciómetros que se encargan de enviar los datos para encender, apagar o variar la frecuencia de los diodos led.**
- **En el análisis de las tramas a través del software Wireshark se pudo comprobar de mejor manera la estructura del protocolo de red Modbus TCP a partir de la comunicación entre la estación maestro y la estación esclavo 3 (PC).**
- **Se comprobó la comunicación entre las estaciones maestro y esclavos desconectando los cables de red de cada uno de las estaciones y verificando si la transmisión y recepción de los datos sufrían algún cambio al manipular las entradas del sistema.**
- **Se requiere la utilización de softwares de comunicación para el protocolo Modbus TCP e IEC60870-5-104 para el funcionamiento y verificación de tramas en el PC, utilizando el software Wireshark, en el caso de Modbus TCP se utilizó Modbus Poll, pero para el IEC60870-5-104, no se consiguió un software que permita realizar esta comprobación, por lo que no se puedo verificar la trama, pese a que si se comprobó su funcionamiento en el banco de pruebas.**

RECOMENDACIONES

- **Revisar las conexiones de los cables de red del banco de pruebas antes de comenzar con el funcionamiento, debido a que si uno de estos no está correctamente conectado la comunicación entre sus correspondientes estaciones no se podrá realizar.**
- **Revisar e identificar las direcciones IP de cada una de las estaciones, principalmente de la estación esclavo 3 (PC), para que no existan falencias en la comunicación y poder realizar un mejor análisis de tramas entre las estaciones deseadas.**
- **Para trabajos futuros, realizar la investigación sobre otros protocolos de red que sean compatibles con los shields utilizados en este proyecto y sus respectivas tarjetas de desarrollo, y a su vez consultar la compatibilidad y usabilidad de varios protocolos de red en otras tarjetas de desarrollo.**