Implantación del protocolo CIP sobre una plataforma embebida para EtherNet/IP

Anexos

Diego Andrés Vásquez Escobar

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Febrero de 2013

Implantación del protocolo CIP sobre una plataforma embebida para EtherNet/IP

Anexos

Diego Andrés Vásquez Escobar

Monografía presentada como requisito parcial para optar por el título de Ingeniería en Automática Industrial

Director: Ing. Vladimir Trujillo Arias

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Febrero de 2013

CONTENIDO

LISTA DE FIGURAS

Pág.

ANEXO A: SELECCIÓN Y DESCRIPCIÓN DE LA TARJETA EMBEBIDA PARA EL DESARROLLO DEL PROYECTO.

El objetivo inicial del proyecto, fue encontrar una tarjeta en el mercado que ofreciera una comunicación Ethernet y que soportara como mínimo hasta el nivel de Ethernet TCP/UDP IP, dado que la finalidad es montar la capa superior CIP en dicha tarjeta; para que esté opere bajo una red industrial EtherNet/IP.

De las tarjetas que se encontraron y que cumplían estas características fueron las siguientes:

- PIC 32 Ethernet Starter Kit.
- Arduino Ethernet.
- Netduino Plus.

1.1 Descripción de las características de las tarjetas encontradas:

1.1.1 Tarjeta PIC 32 Ethernet Starter Kit:

La PIC32 Ethernet Starter Kit contiene todo lo necesario para comenzar el desarrollo de Ethernet y aplicaciones USB Host / Device / OTG con la familia de microcontroladores PIC32.
La placa tiene un conector Ethernet RJ45, un programador/depurador embebido, conectores USB de tamaño completo y Micro-B. El kit contiene un cable Ethernet y un cable USB para cada conector.

Se puede bajar desde web de Microchip toda la documentación de diseño, desarrollo de software y ejemplos de código, para el Ethernet starter kit.

La placa también contiene conectores para ampliar el desarrollo con tarjetas accesorias como sean: Expansión de I/O (DM320002), Tarjeta de Expansión Multimedia (DM320005) .

La Ethernet Starter Kit tiene un formato compatible y conector de expansión como otros PIC32 Starter Kits.

Funciona a 80 MHz con flash de 512 K, 128 KB de memoria RAM, 8 ch. DMA + 8 ch. DMA dedicado a Ethernet , USB y Can.

Se programa en el lenguaje Ansi C bajo el entorno MPLAB IDE v8.43 o la versión más reciente en su defecto [17].

Su precio Julio de 2012, 84.25 euros; y no se consigue en el país.

Figura 1: Tarjeta PIC 32 Ethernet starter kit

Fuente: Tomado de [18]. 3 de Diciembre de 2012

1.1.2 Tarjeta Arduino Ethernet:

Permite a las tarjetas Arduino Duemilanove, PRO y MEGA conectarse por Ethernet a diferentes sistemas como el Internet y otros.
La Arduino Ethernet es un microcontrolador basado en el ATmega328. Tiene 14 pines digitales de E/S, 6 entradas analógicas, un oscilador de 16Mhz, un conector RJ45, un conector de alimentación, un conector ICSP y un botón de reset.

Nota: Los Pines 10, 11, 12 y 13 están reservados para la comunicación con el módulo Ethernet, por lo que no pueden ser utilizados para otros usos. Esto reduce el número de pines disponibles a 9, con 4 capaces de dar salida PWM.

La placa puede ser alimentada desde un módulo POE (Power over Ethernet).

La Arduino Ethernet difiere de otras placas en que no posee un chip convertidor USB a Serie (como el ATmega8u2 de la UNO o el FTDI de modelos anteriores, pero posee un chip Wiznet Ethernet, el mismo que emplea el Ethernet shield.

En la placa se encuentra un slot para tarjeta microSD, el cual puede ser usado para almacenar ficheros y enviarlos más tarde por la red. Este slot microSD es accesible a través de la librería SD.

La comunicación con el slot microSD se hace a través del protocolo SPI, por lo que utiliza los pines 11, 12 y 13 al igual que el chip Wiznet, para el pin SS se reserva el pin 4, por lo que tampoco podrá ser utilizado para otros propósitos si le emplea la microSD.

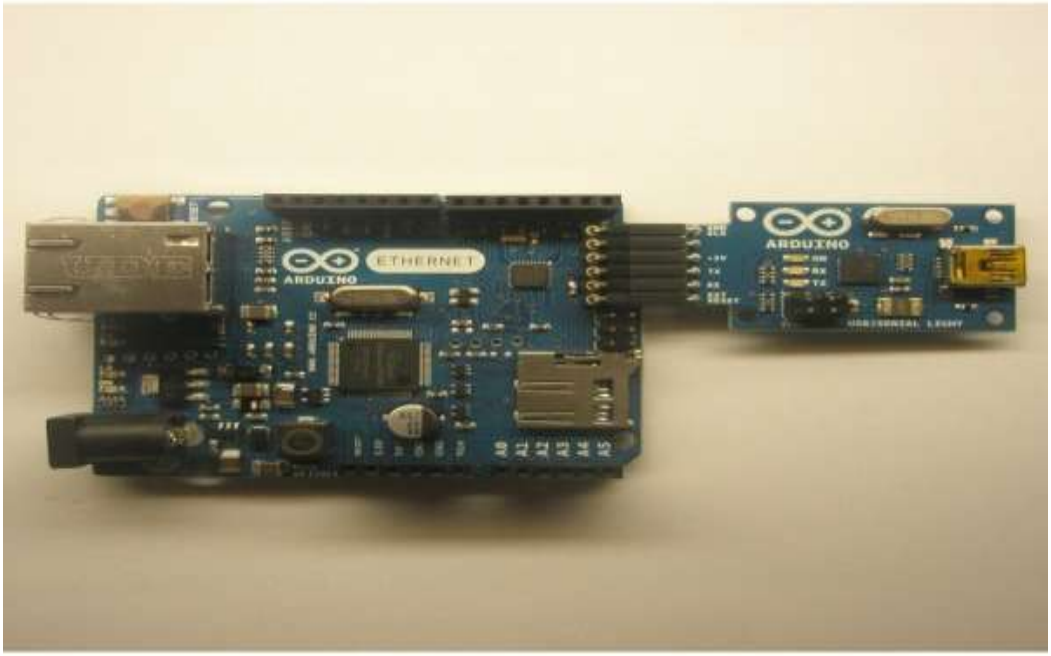
Para programar la Arduino Ethernet se necesita un convertidor USB a Serie externo. Este programador externo provee de comunicación con el ordenador y alimentación eléctrica a través de USB. [19]

Características técnicas:

- Microntrolador Atmega28.
- Voltaje de alimentación: 5 v.
- Alimentación recomendad: 7-12 voltios.
- Voltaje máximo de entrada ro recomendada: 20 voltios.
- Pines digitales I/O 14 (de los cuales 6 dan PWM).
- Pines de entrada analógica: 6.
- Corriente DC por I/O pin 40mA.
- Corriente DC para el pin 3.3 v 50mA.
- Memoria flash 32Kb (Atmega328) 0.5 Kb usado por el bootloader.
- Sram 2Kb (Atmega328).
- EEPROM 1Kb (Atmega328).
- Velocidad de reloj 16 Mhz.

  Precio en el Mercado Colombiano Julio de 2012, $152.000.

Figura 2: Tarjeta Arduino ethernet



Fuente: Tomado de [20],4 de Diciembre de 2012

1.1.3 Tarjeta Netduino Plus:

La tarjeta Netduino es una plataforma abierta basada en Microsoft.NET Micro Framework, programable en Microsoft Visual C Sharp o mejor denotado C#. La

versión Netduino Plus (ND+) cuenta con Ethernet integrado, así como el apoyo de una ranura para microSD en la misma tarjeta.
Cuenta con un potente microcontrolador de 32 bits integrado con un entorno de desarrollo estándar que está disponible gratuitamente a través de Microsoft (Visual Studio 2010).
La plataforma permite una interconexión con switches, sensores, LEDs, dispositivos de serie, y mucho más. La Nd+ combina 20 GPIO con SPI, I2C, UART 2 (1 RTS / CTS), 4 y 6 canales de PWM ADC.

Características técnicas:

- Microcontrolador Atmel de 32 bits.
- Velocidad 48MHz, ARM.
- Memoria de programa: 64 KB (sin emplear networking 128 KB).
- RAM: 28 KB (sin emplear networking 60KB).
- Networking:
  Ethernet: 100 mbps.
  Network stack lwIP [1.]

- Almacenamiento:
  Micro SD de hasta 2 G .
  Autodetección de tarjeta.

Características de la placa:

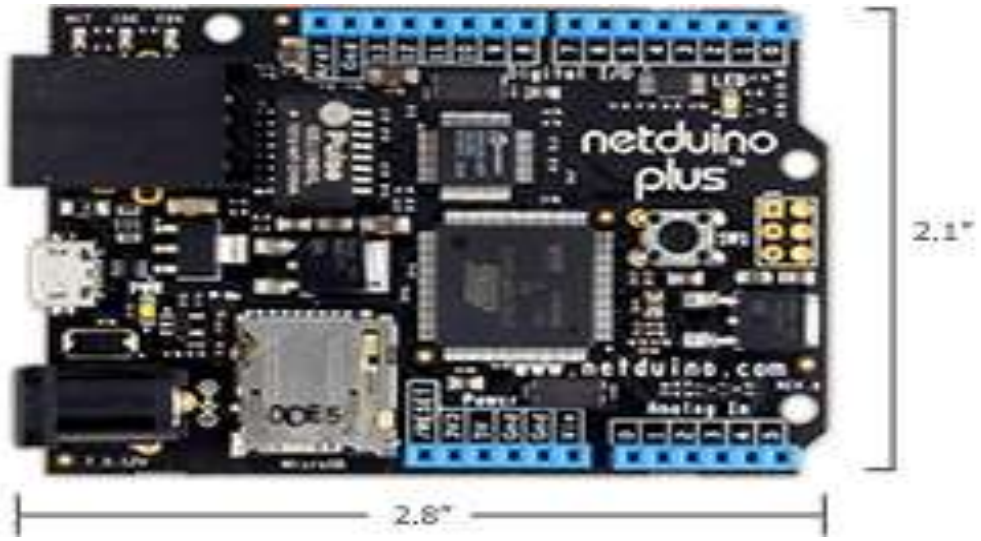
- Todos los 20 pines son digitales o analógicos: GPIO.
- Pines digitales 0-1: UART 1 RX, TX.
- Pines digitales 2-3: UART 2 RX, TX.
- Pines digitales 5-6: PWM, PWM.
- Pines digitales 7-8: UART 2 RTS, CTS.
- Pines digitales 9-10: PWM, PWM.
- Pines digitales 11-13: SPI MOSI, MISO, SPCK.
- Pines analógicos 4-5: I2C SDA, SCL.

La tarjeta Netduino es compatible pin a pin con la tarjeta Arduino [21]

Precio en el Mercado Colombiano Julio de 2012, $180.000.

1 El lwIP es una puesta en práctica ligera del protocolo TCP/IP para sistemas embebidos. Sirve para reducir el uso de recursos y la escala completa TCP.Esto hace lwIP adecuado para su uso en sistemas embebidos con decenas de kilobytes de RAM libre y el espacio para alrededor de 40 kilobytes de código ROM.

Figura 3: Tarjeta Netduino Plus



Fuente: Tomado de [22],4 de Diciembre de 2012

Por el tipo de lenguaje de programación que presenta, el acceso fácil y rápido de códigos y ayudas online que se pueden obtener en los foros de la Netduino [4], las características intrínsecas que ofrece y el carácter innovador que representa el empleo de éste elemento en Colombia, la tarjeta que se escogió fue la Netduino Plus.

1.2 Descripción de los componentes de programación de la Netduino Plus.

1.2.1 C#:

El avance de las herramientas de programación y los dispositivos electrónicos para el consumidor ha creado problemas y nuevos requerimientos. La integración de componentes de software de diversos lenguajes fue difícil y los problemas de instalación eran comunes, ya que las nuevas versiones de los componentes compartidos eran incompatibles con el software anterior. Los desarrolladores también descubrieron que necesitaban aplicaciones basadas en Web, que pudieran acceder y utilizarse a través de Internet. Como resultado de la popularidad de los dispositivos electrónicos móviles, los desarrolladores de software se dieron cuenta que sus clientes ya no estaban restringidos solo a las computadoras de escritorio. Reconocieron que hacía falta software accesible para todos y que estuviera disponible a través de casi cualquier tipo de dispositivo.

Para satisfacer esas necesidades en el año 2000. Microsoft anuncio el lenguaje de programación C#. Este lenguaje, se diseñó en específico para la plataforma .Net

(de la cual se hablara en el siguiente item) como un lenguaje que permitiera a los programadores migrar con facilidad hacia .Net. Tiene sus raíces en C, C++ y Java; adapta las mejores características de cada uno de estos lenguajes y agrega nuevas características propias. C# está orientado a objetos y contiene una poderosa *biblioteca de clases*, que consta de componentes preconstruidos que permiten a los programadores desarrollar aplicaciones con rapidez; C# y Visual Basic comparten la biblioteca de Clases Framework (FCL).

C# es un lenguaje de programación visual, controlado por eventos en el cual se crean programas mediante el uso de un *Entorno de Desarrollo Integrado (IDE)*.

Con un IDE, un programador puede crear, ejecutar, probar y depurar programas en C# de manera conveniente, con lo cual se reduce el tiempo requerido para producir un programa funcional en una fracción del tiempo que se llevaría sin utilizar el IDE.

1.2.2 Microsoft .Net.

En el año 2000 Microsoft anunció su iniciativa .Net; una nueva visión para incluir internet y web en el desarrollo y uso de software.

Un aspecto clave de .Net es su independencia de un lenguaje o plataforma específicos. En vez de estar forzados a utilizar un solo lenguaje de programación, los desarrolladores pueden crear una aplicación .Net en cualquier lenguaje compatible con ésta.

La estrategia .Net extiende el concepto de reutilización de software hasta internet, con lo cual permite que los programadores y las compañías se concentren en sus especialidades sin tener que implementar cada componente de cada aplicación. En vez de ello, las compañías pueden comprar los servicios web y dedicar sus recursos a desarrollar sus propios productos. Por ejemplo, una aplicación individual que utilice los servicios web de varias compañías podrá administrar los pagos de facturas, devoluciones impuestos, préstamos e inversiones.

Un comerciante de aerolínea podrá comprar servicios web para los pagos de tarjeta de crédito en línea, la autenticación de los usuarios, la seguridad de la red y bases de datos de inventarios para crear un sitio web de comercio electrónico.

1.2.3 .Net Framework y Common Language Runtime.

El .Net Framework de Microsoft es el corazón de la estrategia .Net. Este marco de trabajo administra y ejecuta aplicaciones y servicios web, contiene una biblioteca de clases (llamada biblioteca de clases de.Net Framework, o FCL), impone la seguridad y proporciona muchas otras herramientas de programación. Los detalles de.Net Framework se encuentran en la *Infraestructura del Lenguaje Común (CLI)*, la cual contiene información acerca del almacenamiento del tipo de datos (es decir, datos que tiene características preferidas como fechas, porcentaje o una cantidad monetaria), los objetos y demás.

El entorno en tiempo de ejecución *Common Language Runtime (CLR)* es otra parte del .Net Framework: ejecuta los programas de .Net. Los programas se compilan en instrucciones específicas para la máquina.

Si existe el .Net Framework (y está instalado) para una plataforma, esa plataforma puede ejecutar cualquier programa .Net. La habilidad de un programa para ejecutarse (sin modificaciones) en varias plataformas se conoce como independencia de la plataforma. De ésta forma, el código inscrito puede usarse en otro tipo de computadoras sin modificación, con lo cual se ahorra tiempo y dinero. Además, el software puede abarcar una audiencia mayor; antes las compañías tenían que decidir si valía la pena el costo de convertir sus programas para usarlos en distintas plataformas (lo que algunas veces se conoce como *portar*).

Cualquier lenguaje .Net puede utilizar la FCL. Esta biblioteca contiene una variedad de componentes reutilizables, con lo cual los programadores se ahorran el problema de crear nuevos componentes. Los lenguajes de programas .Net con mayor renombre son:

- C#.
- Cobol.
- Fortran.
- JAVA.
- JScript.
- Pascal.
- Python.
- Visual Basic.
- Visual C++.[23]

1.3 Experiencia sobre el uso de la tarjeta Netduino Plus:

Para comenzar con el desarrollo con la Nd+, debemos instalar los siguientes componentes que son gratuitos (se recomienda que se instale todo en el mismo orden que aparece a continuación):

1) Microsoft Visual C# Express 2010. Disponible en [24].
2) .Net Micro Framework. Disponible en [25]. (Por cuestiones de accesibilidad a códigos cuando se tenga varios PC trabajando con la misma ND+, se debe también instalar la Microsoft .Net Framework 3.5 SP1).

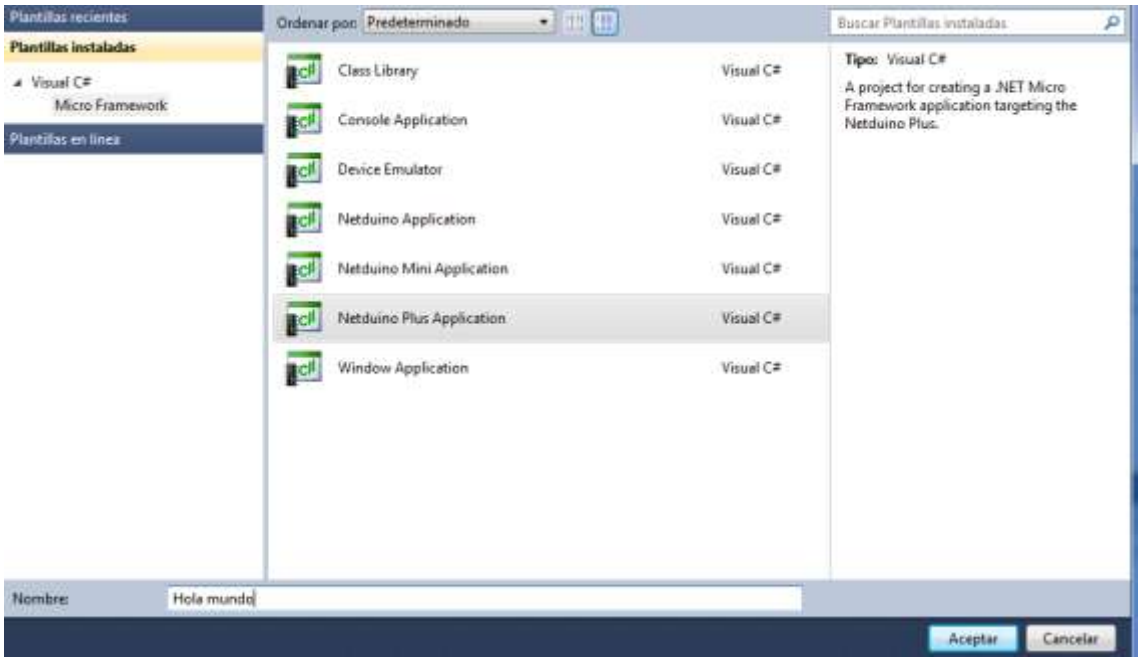
3) Netduino SDK. Esta la versión de 32 bits y la de 64 bits, dependerá de la característica del PC del usuario. Disponible en [26].

Ahora que se tiene todo el software instalado, vamos a probar que todo funciona. Si no aparece estar funcionando se puede descargar los controladores desde el [27].

Para comenzar vamos a mostrar cómo crear un pequeño servidor web, que ejecutaremos en nuestro ND+. El servidor web debe responder ante cada petición HTTP con un "Hola Mundo".

Empezamos creando un nuevo proyecto de tipo ND+ en nuestro Visual Studio.

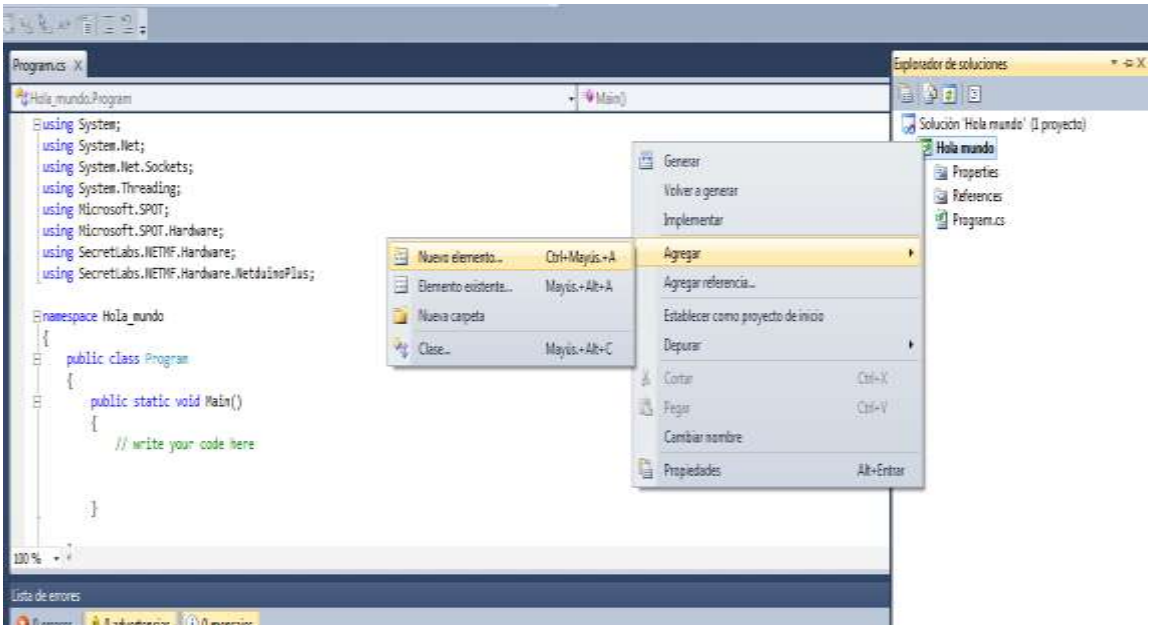
Figura 4: Creación de un proyecto en la Netduino Plus



Fuente propia

Botón derecho sobre el proyecto y se elige Agregar-Clase

Figura 5: Creación de una clase Webserver en el proyecto de la Netduino Plus



Fuente propia

Se le da el nombre WebServery se le da aceptar.

Aquí presentamos una versión muy simple de nuestro servidor web. El servidor se mantiene a la escucha, atendiendo las peticiones, y respondiendo "Hola Mundo" mientras hace parpadear el led de la ND+.

Código Clase Webserver:

```
public class WebServer : IDisposable
{
  private Socket socket = null;
  private OutputPort led = new OutputPort(Pins.ONBOARD_LED, false);
  public WebServer()
 {
      //Inicializamos el socket
      socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
      //Conectamos el socket
      socket.Bind(new IPEndPoint(IPAddress.Any, 80));
     //Iniciamos la escucha
     socket.Listen(10);
     ListenForRequest();
```

```
 }

 public void ListenForRequest()
{
  while (true)
  {
     using (Socket clientSocket = socket.Accept())
     {
       //Aceptamos el cliente
       IPEndPoint clientIP = clientSocket.RemoteEndPoint as IPEndPoint;
       EndPoint clientEndPoint = clientSocket.RemoteEndPoint;
       int bytesReceived = clientSocket.Available;
       if (bytesReceived > 0)
      {
           //Obtenemos la petición
           byte[] buffer = new byte[bytesReceived];
           int byteCount = clientSocket.Receive(buffer, bytesReceived,
SocketFlags.None);
           string request = new string(Encoding.UTF8.GetChars(buffer));
           Debug.Print(request);
           string response = "Hola Mundo";
           string header = "HTTP/1.0 200 OK\r\nContent-Type: text; charset=utf-
8\r\nContent-Length: " + response.Length.ToString() + "\r\nConnection:
close\r\n\r\n";
           clientSocket.Send(Encoding.UTF8.GetBytes(header), header.Length,
SocketFlags.None);
           clientSocket.Send(Encoding.UTF8.GetBytes(response), response.Length,
SocketFlags.None);
           //Parpadeo del led
           led.Write(true);
           Thread.Sleep(150);
           led.Write(false);
      }
    }
  }
}

#region IDisposable Members
~WebServer()
{
     Dispose();
}
public void Dispose()
{
    if (socket != null)
       socket.Close();
}
```

```
#endregion
}
```

El siguiente paso es añadir el siguiente código en el Program.cs para iniciar el servidor web.

```
public static void Main()
{

Microsoft.SPOT.Net.NetworkInformation.NetworkInterface.GetAllNetworkInterfaces
()[0].EnableDhcp(); //despliega la dirección IP del servidor DHCP.
    WebServer webServer = new WebServer();
    webServer.ListenForRequest();
}
```

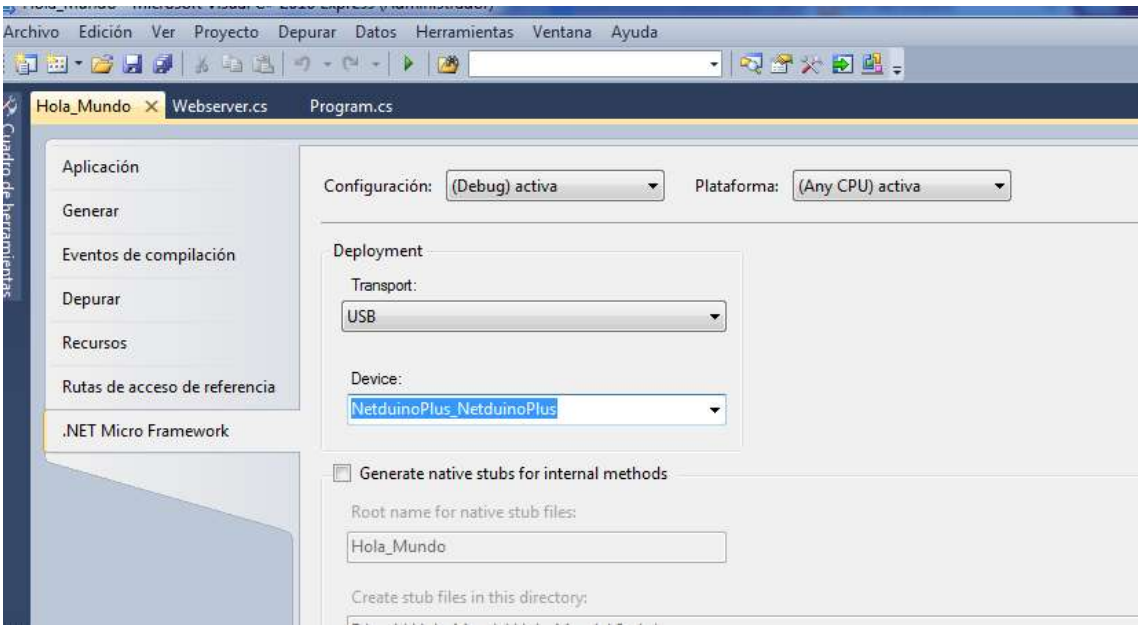
Código obtenido de [28].

A continuación conectaremos la ND+ en el PC (instalar el hardware).

Posteriormente tenemos que desplegar el código en la ND+ haciendo click en las Propiedades del proyecto-Propiedades de Hola mundo.

Una vez cambiadas las propiedades del proyecto, seleccionamos la pestaña ".Net Micro Framework" y cambiamos las siguientes propiedades:

Figura 6: Configuracion de las propiedades de la Netduino Plus.



Fuente propia

Transport: USB
Device: NetduinoPlus_NetduinoPlus

Se conecta la ND+, a un router o a un punto de red. Al igual que la PC o vía wifi.

Al iniciar la depuración del proyecto, el depurador de la ND+ despliega una dirección IP. Finalmente buscamos en el navegador la dirección http y allí podemos encontrar el mensaje “Hola Mundo”.

Ahora ya que te tuvo la primera experiencia con la ND+, lo más importante para seguir con la elaboración del proyecto son los siguientes ítems:

- Aprender a programar en el lenguaje C#, sobre todo el manejo de los sockets que es la base para el funcionamiento de este proyecto.

- Extraer los elementos que proporciona el protocolo CIP, para aplicarlos y garantizar que se está implantando de forma adecuada, para operar en nuestro sistema empotrado.

ANEXO B: EXPERIENCIA EN EL MANEJO DE SOCKETS EN C# Y FORMA DE APLICACIÓN EN LA NETDUINO PLUS.

2.1 ¿Qué es un socket?

Cita de la Wikipedia:

Citar:

***Socket** : Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas, pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.*

El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados.

Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Es decir es un método para que aplicaciones situadas en distintos ordenadores (o no, pueden comunicarse aplicaciones situadas en el mismo ordenador) puedan comunicarse. Para ello, se necesita la IP, un Puerto y un Protocolo.

2.2 Protocolos para transportar datos:

Existen muchos protocolos para la comunicación entre aplicaciones. Los protocolos son conjunto de reglas que gobiernan la forma en que deben interactuar dos entidades. Para este caso es importante lo que nos brinda los protocolos de control de transmisión (TCP) y el protocolo de datagramas de usuario (UDP). Las herramientas de red TCP y UDP de .Net se define en el espacio de nombres **System.Net.Sockets.**

El protocolo de transmisión (TCP) es un protocolo de comunicación orientado a la conexión, el cual garantiza que los paquetes enviados llegaran al receptor destinado sin daños y en la secuencia correcta. TCP permite que protocolos como HTTP envíen información a través de una red, de una manera tan simple y confiable como escribir a un archivo en una forma local. Si los paquetes de información no lleguen al recipiente, TCP se asegura que los paquetes se envíen de nuevo. Si los paquetes llegan fuera de orden, TCP los vuelve a ensamblar en el orden correcto, en forma transparente para la aplicación receptora. Si llegan paquetes duplicados, TCP los descarta.

Las aplicaciones que no requieren de la garantía de transmisión confiable de un extremo que ofrece TCP, por lo general utilizan el protocolo de datagramas de usuario (UDP) sin conexión. UDP incurre en la mínima sobrecarga necesaria para la comunicación entre aplicaciones; no garantiza que los paquetes, llamadas datagramas, lleguen a su destino o que lleguen en el orden original.

Existen ciertas ventajas en cuanto al uso de UDP en comparación con TCP. UDP tiene poca sobrecarga, ya que los datagramas no necesitan llevar la información que llevan los paquetes TCP para asegurar la confiabilidad. Además reduce el tráfico de red relativo a TCP, debido a la ausencia de intercambio, retransmisiones, etcétera.

La comunicación no confiable es aceptable en muchas situaciones. En primer lugar, la confiabilidad no es necesaria para algunas aplicaciones, por lo que puede evitarse la sobrecarga impuesta por un protocolo que garantiza la confiabilidad. En segundo lugar, algunas aplicaciones como el audio y video en tiempo real pueden tolerar la pérdida ocasional de datagramas. Esto por lo general produce una pequeña pausa en el audio o video que se está produciendo. Si la misma aplicación se ejecutara sobre TCP, un segmento perdido podría producir una pausa considerable, ya que el protocolo esperaría a que se retransmitiera el

segmento perdido y se entregara en forma correcta antes de continuar. Por último, las aplicaciones que necesitan implementar sus propios mecanismos de confiabilidad distintos a los de TCP pueden crear tales mecanismos sobre UDP.

2.3 Teoría sobre sockets en C#:

Hay unas líneas de código para el manejo de sockets en C#, que son fundamentales:

En primer lugar para indicar que queremos trabajar con sockets, tenemos que agregar las siguientes líneas en el using (los proyectos en Nd+ ya las tiene por defecto).

Código:

```
Using System.Net;

Using System.Net.Sockets;
```

Código:

```
Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
```

Dividámoslo por argumentos:

AddressFamily.InterNetwork: Esto indica que la dirección es para una IP v4 (xxxx.xxxx.xxxx.xxxx).

SocketType.Stream: Es el tipo de socket, hay varios tipos, para más información sobre estos consultar la información que les da el entorno de programación.

ProtocolType.Tcp: Es el protocolo que se usara, también hay varios, para más información sobre estos consultar la información que les da el entorno de programación.

Código:

```
IPEndPoint EP=new IPEndPoint (direccion,puerto);
```

Representa un extremo de red como una dirección IP y un número de puerto.

Código:

```
Socket.Bind(EP);

Socket.Listen(100);
```

Lo que hacemos aquí es proceder a escuchar por el puerto descrito en EP. Luego en la siguiente línea, parámetro Listen establece la longitud máxima de la cola de conexiones pendientes que puede tener el servidor antes de empezar a rechazar conexiones.

La siguiente línea a agregar es la siguiente:

Código:

Socket handler=socket.Accept();

Esta línea indica que el servidor queda esperando una llamada para luego aceptarla.

2.4 Manejo de sockets C# de forma local en PC:

A continuación veremos una comunicación Cliente-Servidor que será una forma de chat de formal local de PC, el cual será primer paso para la ejecución de nuestro proyecto:

En primer lugar abriremos un proyecto de modo consola en el Visual Studio que la denominaremos Clientetcp:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;// Agregamos las librerias para manejo de sockets

namespace Cliente
{
Class Program
    {
Static Socket sck;
Static void Main(string[] args)
        {
sck=new Socket(AddressFamily.InterNetwork,SocketType.StreamProtocolType.Tcp);
//Creamos el socket
IPEndPoint localEndPoint=new IPEndPoint(IPAddress.Parse("127.0.0.1"),44818);//
utilizamos la direccion LocalHost del PC y este numero de puerto es el AF12, que es
el puerto EtherNet/IP que se utilizara en el proyecto.
try
            {
sck.Connect(localEndPoint);// se conecta con el socket
string text = "";
while (text.ToUpper() != "Q")//Con la letra Q finalizaremos la comunicacion
                {
Console.Write("enter text: ");//para que comience el chat
text = Console.ReadLine();// Lee el texto introducido
byte[] data = Encoding.UTF8.GetBytes(text);//Lo transforma a    un string tipo UTF8
var x = sck.Send(data);// Captura el número de bytes de el texto y lo envía.
```

```
Console.Write("Data send : " + x + "\r\n");//Imprime el numero de bytes del texto.
                }
                sck.Close();//Cierra el TCP.
            }
catch
            {
Console.Write("Imposible conectar con el servidor \r\n");
Main(args);
            }
        }
    }
}
```

Ahora el código para el servidor, así que se abre otro proyecto en modo consola en el Visual Studio.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace Server
{
classProgram
    {
staticbyte[] Buffer { get; set; }
staticSocket sck;
staticvoid Main(string[] args)
        {
sck=new Socket(AddressFamily.InterNetwork,SocketType.Stream,ProtocolType.
Tcp);
            sck.Bind(newIPEndPoint(0,44818));//Escuchamos el trafico que viene de
ese puerto, el 0 referencia que es la PC quien lo recibe.
sck.Listen(100);
Socket accepted = sck.Accept();//aceptamos la conexion.

while (true)
            {
                Buffer = new byte[accepted.SendBufferSize];
int bytesRead = accepted.Receive(Buffer);//almcaenamos la informacion recibida en un
buffer
byte[] formatted = new byte[bytesRead];
for (int i = 0; i < bytesRead; i++)
                {
formatted[i] = Buffer[i];
                }
string strData = Encoding.UTF8.GetString(formatted);
Console.Write(strData + "\r\n");//finalmente mostramos el mensaje recibido
if (strData.ToUpper() == "Q") break;//Si se recibe una Q, se termina la conexion.
}
Console.Write("ADIOS");
sck.Close();
accepted.Close();
}
```

```
    }
}
```

Código propio

Y corremos los dos programas.

Figura 7: Chat de PC en forma local

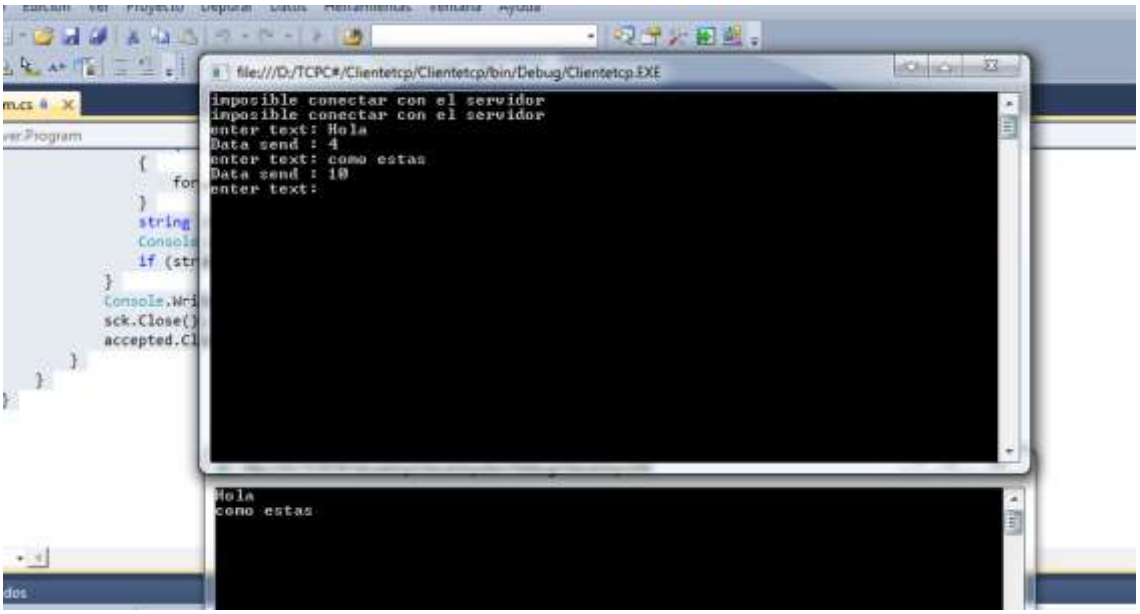


Figura propia

2.5 Manejo de los sockets en la Netduino Plus:

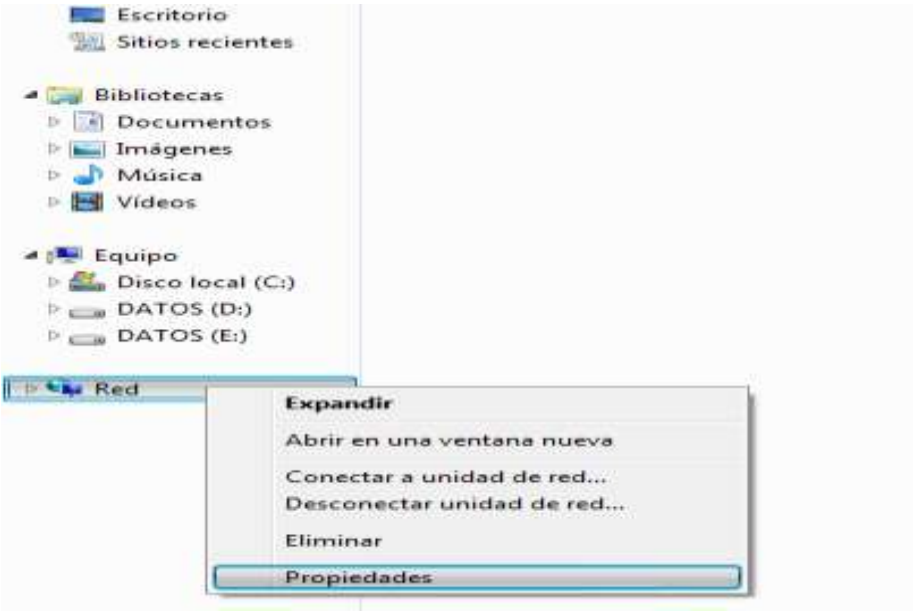
Después de comprender el manejo de los sockets en el PC, el siguiente paso para la consecución del proyecto fue aprender a manejar estos en la ND+.

La primera parte consistía en realizar una conexión física de PC con tarjeta, en el cual se podrá realizar una comunicación punto a punto entre estos dos dispositivos.

Con un cable de red cruzado, se conectaron y se configuraron sus correspondientes direcciones IP.

Para configurar la dirección IP del PC, se va a configuración de red, le damos clic derecho, y se elige propiedades.

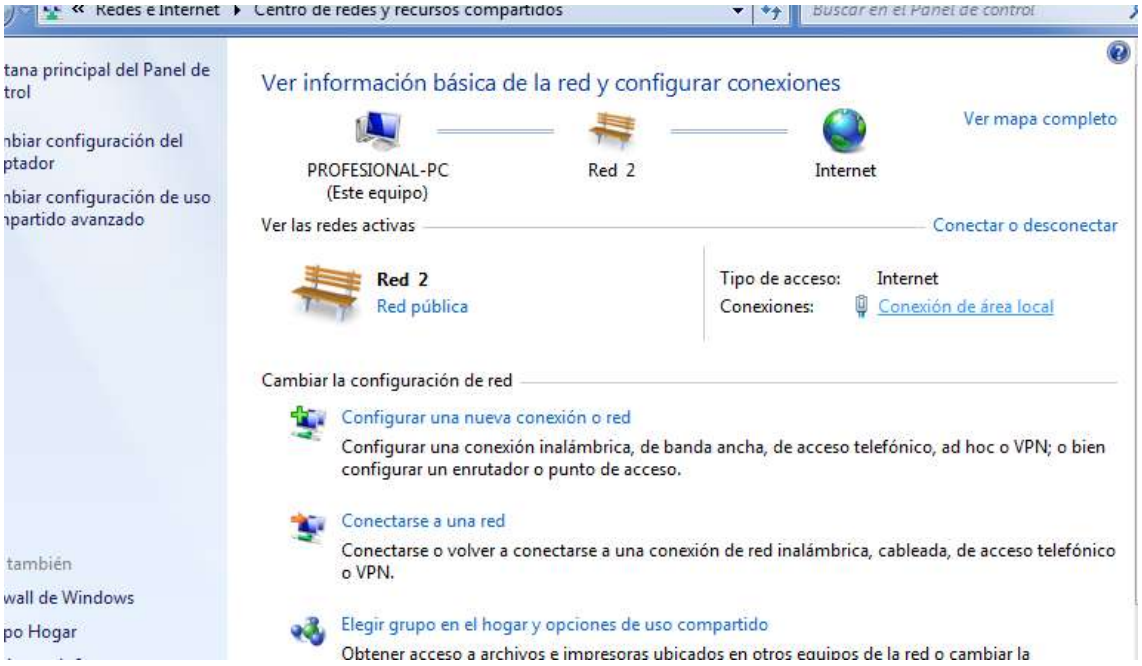
Figura 8: Paso 1, configuración IP del PC.



Fuente propia.

Ahora le damos click en conexión de área local:

Figura 9: Paso 2, configuración IP del PC.



Fuente propia

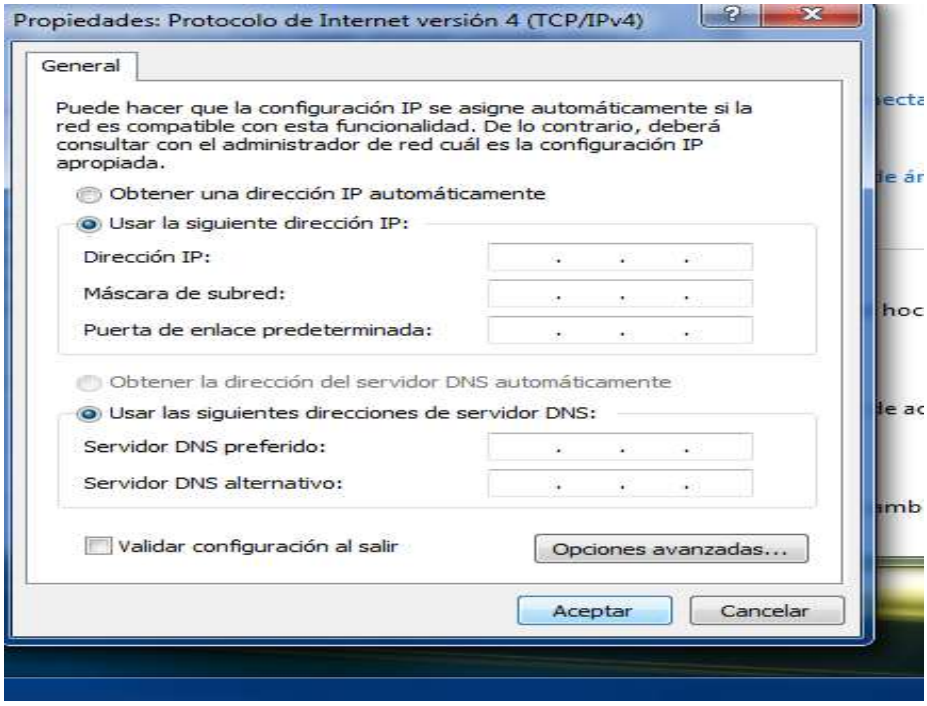
En la ventana que aparecerá le damos propiedades, elegimos la propiedad de protocolo de internet versión 4, y hacemos click en propiedades.

Figura 10: Paso 3, configuración IP del PC.

Fuente propia

Finalmente se desplegara la ventana en donde se podrá configurar la dirección IP, la puerta de enlace, servidor DNS y el servidor DNS 2.

Figura 11: Paso 4, configuración IP del PC.

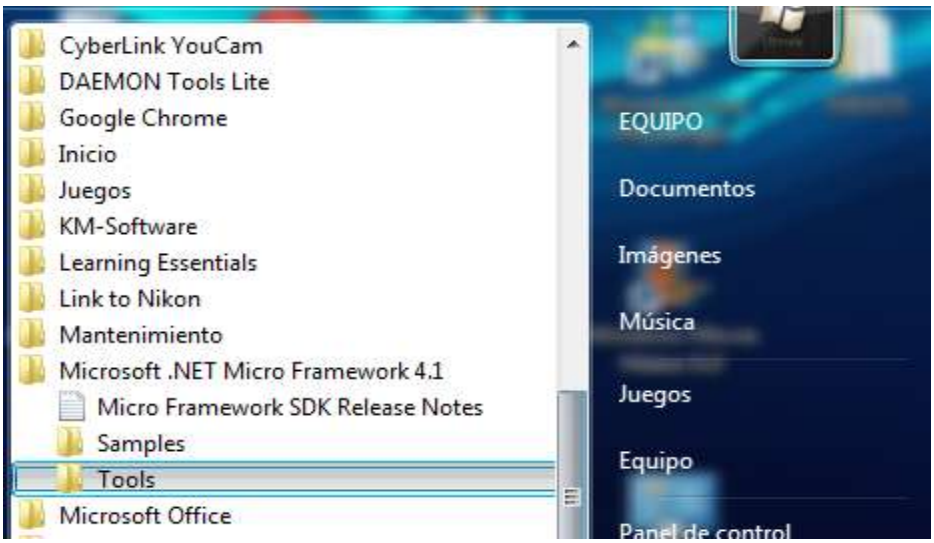


Fuente propia

Elegiremos una dirección IP clase C, “192.168.1.1” y las otras del mismo estilo.

Para configurar la dirección IP de la ND+, se abre la Microsoft .Net Micro Framework y se elige Tools.

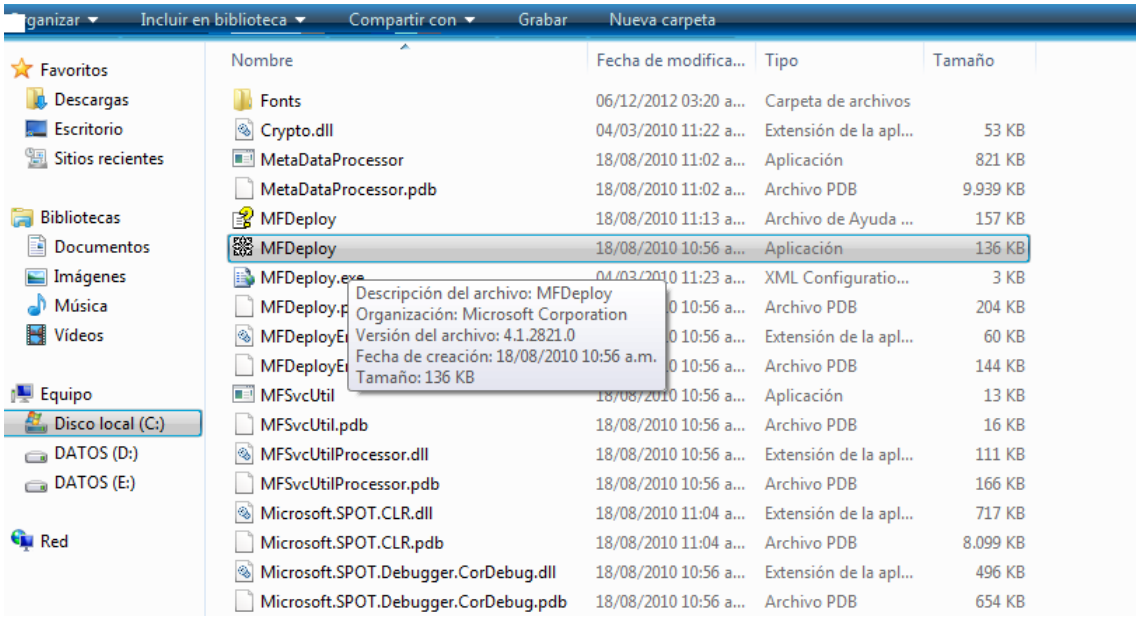
Figura 12: Paso 1, configuración IP de la Netduino plus



Fuente Propia

Una vez abierto elegimos la opción MFDeploy.

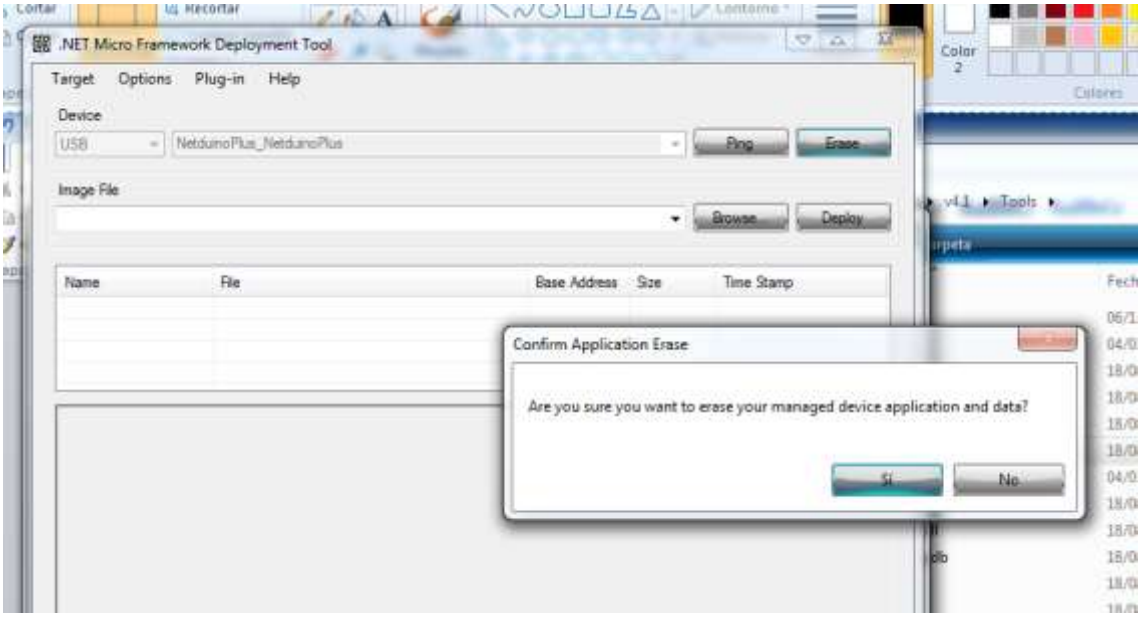
Figura 13: Paso 2, configuración IP de la Netduino plus



Fuente propia.

En la opción Device elegimos USB, y allí aparecerá desplegado la Nd+. Damos Erase y le damos sí.

Figura 14: Paso 3, configuración IP de la Netduino plus

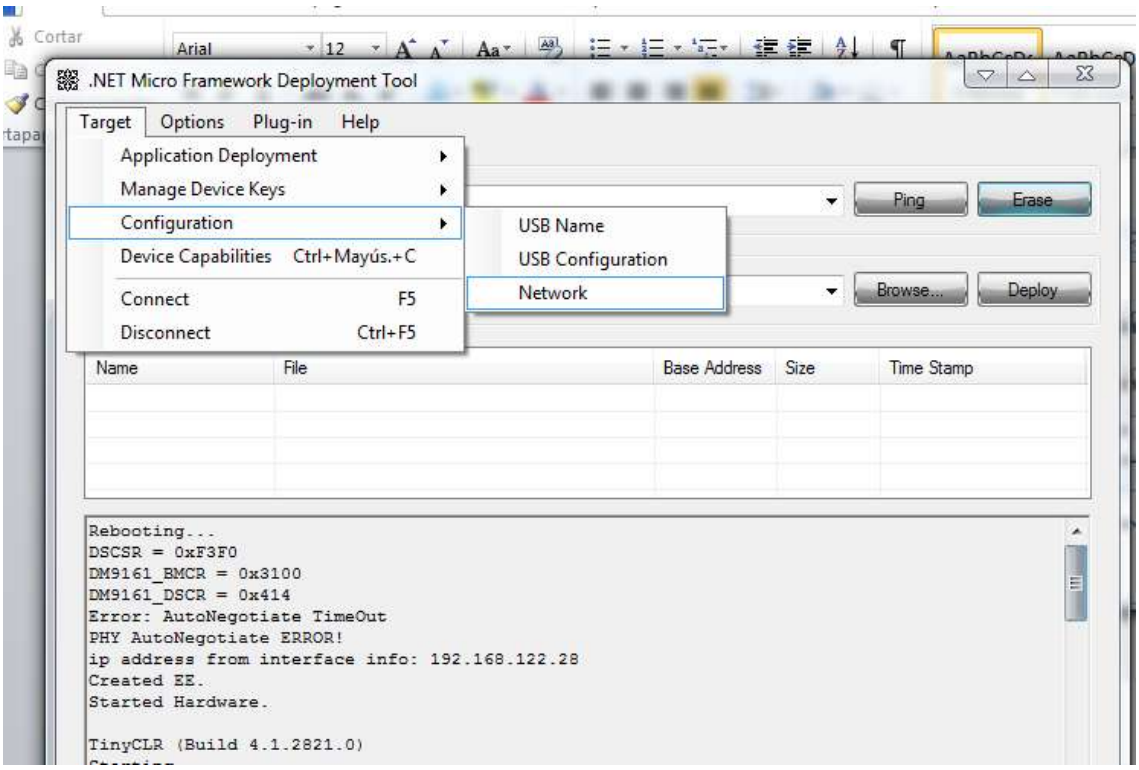


Fuente propia

Así tendremos la ND+ en modo tiny, que es para que ella ya pueda ser configurada.

Ahora se elige la opción Target-Configuration-Network.

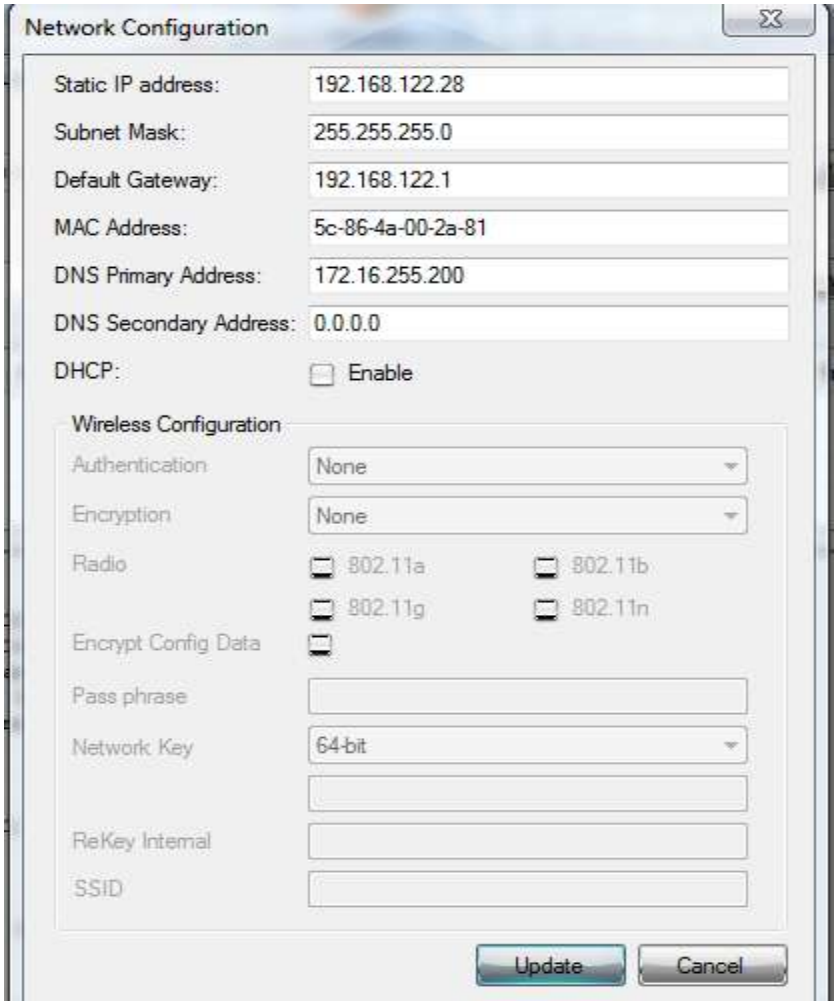
Figura 15: Paso 4, configuración IP de la Netduino plus



Fuente propia

Y ahí se desplegara el menú para que esta se pueda configurar su dirección IP, puerta de enlace, DNS primario y secundario.

Figura 16: Paso 5, configuración IP de la Netduino plus



Fuente propia.

Se configura la IP, de tal manera que este en el rango de la IP del PC, es decir “192.168.1.xxx”, igual que las otras direcciones.

Para resetear la IP de la ND+, simplemente se mantiene presionado el botón que tiene esta y se repite el procedimiento.

Después de hacerles un ping en la ventana de comandos para comprobar, que ambos dispositivos estén conectados (la ND+ se mostró muy intermitente en cuanto a estado de conexión). Se hizo el siguiente código que manda un mensaje desde el PC, y este se mostrara en la ventana de depuración de la ND+.

Código PC:

```
sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Parse("192.168.1.1"), 44818);//Que se ubique en la IP de la Nd+, puerto AF12 EtherNet/IP
try
{
sck.Connect(localEndPoint);
}
catch
{
Console.Write("Imposible conectar \r\n");
Main(args);
}

string text = "hola desde PC";
byte[] data = Encoding.UTF8.GetBytes(text);
while (true)
{
sck.Send(data);
System.Threading.Thread.Sleep(1000);
}
```

Código propio

Código ND+:

```
OutputPort led = new OutputPort(Pins.ONBOARD_LED, false);

Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Any, 44818);
sck.Bind(localEndPoint); //Escuche lo que viene en ese Puerto.
sck.Listen(10);
Socket clientSocket;
while (true)
{
clientSocket = sck.Accept();

while (clientSocket != null)
{
try
{
Byte[] buffer = new Byte[1024];

clientSocket.Receive(buffer);
clientSocket.Send(buffer); // send back the data
string request = new string(Encoding.UTF8.GetChars(buffer));
Debug.Print("data recieved: " +request);
}
catch
{
```

```
clientSocket = null;
}
}
```

Código propio

El resultadofuenegativo; así que se concluyó que podría ser por la intermitencia en la conexión de la ND+.

Así que se escogió utilizar el servidor DHCP para que configurara las direcciones IP de ambos dispositivos.

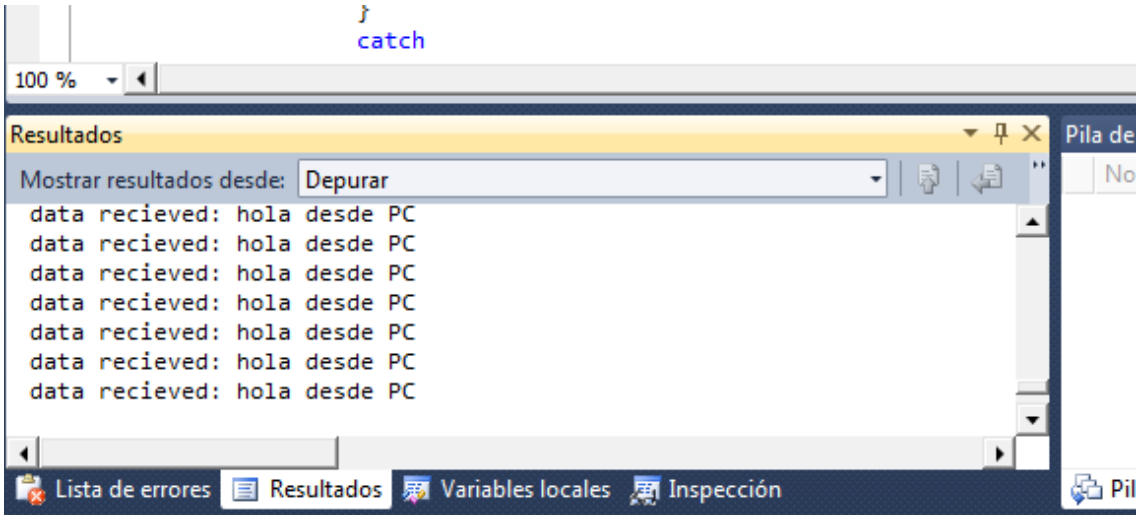
Para saber la dirección IP de la ND+, hay que realizar el siguiente código:

```
Debug.Print(Microsoft.SPOT.Net.NetworkInformation.NetworkInterface.GetAllNetworkInte
rfaces()[0].IPAddress);
```

Utilizando un router inalámbrico, la dirección IP que arrojo fue “192.168.100.1”.

Ejecutando nuevamente el código con dicha dirección IP, finalmente se logró el propósito.

Figura 17: Comunicación PC con la Netduino Plus



Fuente propia

En el momento en que se logra el objetivo de comunicar el PC con la ND+, por el puerto reservado EtherNet/IP, se procede con la selección de los elementos que proporciona el protocolo CIP EtherNet/IP para finalmente codificarlos y garantizar que éste se va a implantar de forma adecuada.

ANEXO C: EXPERIENCIA EN LA IMPLANTACIÓN DEL PROTOCOLO CIP EN EL PC Y LA NETDUINO PLUS.

3.1 Requisitos para la ejecución del proyecto:

Toda trama CIP para EtherNet/IP, se ejecuta por medio de una serie de comandos, que activan todo el manejo de *modelo de objetos* CIP. Es por esto que para este proyecto (según la formulación del protocolo), utilizaremos 3 comandos.

- RegisterSession: Que se utiliza para abrir una conexión, y todo el tráfico de comunicación se enviara por el puerto AF12 hexadecimal, que es el único puerto que garantiza la transmisión EtherNet/IP.

- SendUniData: Ya que como vimos anteriormente la comunicación entreel PC y la ND+, tiene su ruta de conexión establecida.

- UnRegisterSession: Para terminar la conexión.

Generalmente una conexión CIP en EtherNet/IP (implícitamente o explícitamente), se abre mediante una función UCMM.

Esta se lleva a cabo dentro del servicio Forward_Open, que es de tipo opcional lo cual lo hace el objeto Conection Manager.

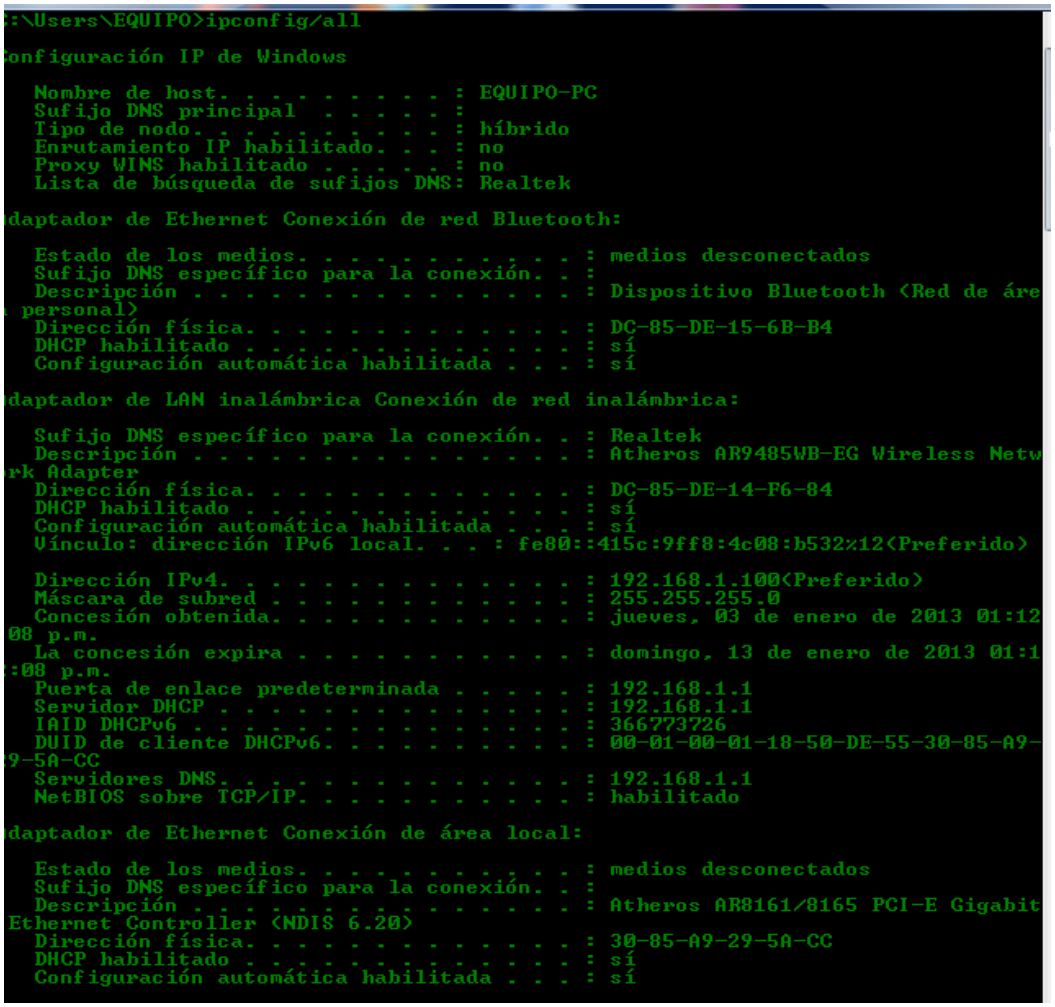
Pero esta función se puede llevar a cabo, siempre y cuando los dispositivos involucrados que están desconectados,actúan sin conexión. Que es contrario como actúa la comunicación entre PC y la ND+. Así que esta función no podrá ser implementada.

Ahora los objetos Identity, Message Router, TCP/IP Interface, y Ethernet Link con los atributos requeridos correspondientes si serán implementados en ambos dispositivos. Además se hará un objeto adicional propio en la ND+, el cual tendrá como objetivo hacer parpadear el led que tiene esta.

Para obtener los valores de atributo, los objetos del PC y la tarjeta. Es necesario ver la configuración IP, de cada uno de estos dispositivos.

Para ver los parámetros IP del PC, vamos a símbolo del sistema o damos CMD en el buscador de programas del PC; y hacemos el siguiente comando ipconfig/all.

Figura 18: Datos de red del PC



Fuente propia

Y así podemos ver los parámetros del PC, que se necesitan anexarlos en los valores de atributo CIP.

Para conocer ahora los de la ND+, hacemos todos los pasos que se muestran de la figura 11 a la 15.

Ahora si los valores de atributo de los dispositivos fueron:

3.1.1Atributos de instancia de la Netduino Plus:

Atributos Objeto Identity:

- Vendor ID: 2000.(Valor definido)
- Device Type: 120 (que está en el rango de direcciones para perfil de dispositivo permitido para una definición específica de fabricante).
- Product Code: 2020. (Valor definido)

- Major Revision:2
- Minor Revision:1(Valores comúnmente usados por dispositivos que operan bajo EtherNet/IP)
- Status: 0.(Ver definición de este atributo)
- Serial Number: ND1 (Valor definido)
- Product Name: Netduino Plus. (Valor definido)

Atributos Objeto TCP/IP Interface:

- Status:1 (Ver definición de este atributo)
- Configuration Capability: 2. (Ver definición de este atributo)
- Configuration Control: 2. (Ver definición de este atributo)
- Physical Link Object: 20f62401. (Ver definición de este atributo)
- IPaddress: 192.168.1.100 (Ver definición de este atributo.)
- Name server: 172.16.255.200 (Ver definición de este atributo)
- Name Server 2: 0.0.0.0 (Ver definición de este atributo)
- Host Name: Netduino (Valor definido).

Atributos EtherNet Link:

- Interface Speed:1.433 Mbbps (Ver definición de este atributo)
- Interface Flags: 0 (Ver definición de este atributo)
- Physical Address:5C-86-4A-00-2A-81. (Ver definición de este atributo)

3.1.2Atributos de instancia del PC:

Atributos Objeto Identity:

- Vendor ID: 2050. (Valor definido)
- Device Type: 110 (que está en el rango de direcciones para perfil de dispositivo permitido para una definición específica de fabricante).
- Product Code: 2070. (Valor definido)
- Major Revision:2
- Minor Revision:1. (Valores comúnmente usados por dispositivos que operan bajo EtherNet/IP)
- Status: 0.
- Serial Number: F45U (Número de serie del PC)
- Product Name: Axus (Nombre de la empresa del PC)

Atributos Objeto TCP/IP Interface:

- Status:1. (Ver definición de este atributo)
- Configuration Capability:2 (Ver definición de este atributo)
- Configuration Control:2 (Ver definición de este atributo)

- Physical Link Object: 21f62401. (Ver definición de este atributo)
- IPaddress: 192.168.1.100 (Ver definición de este atributo)
- Name server: 192.168.1.1 (Ver definición de este atributo)
- Name Server 2: 0.0.0.0 (Ver definición de este atributo)
- Host Name: Equipo-PC(Ver definición de este atributo)

Atributo EtherNet Link:

- Interface Speed:1.433 Mbbps (Ver definición de este atributo)
- Interface Flags: 0 (Ver definición de este atributo)
- Physical Address:DC-85-DE-14-6B-B4 (Ver definición de este atributo)

Ahora ya que tenemos estos valores de atributo, podemos empezar con la programación del software.

3.2 Restricciones para la ejecución del proyecto:

Cuando se ejecuta un proyecto para la ND+, este tiene dos formas de visualización para un usuario:

El primero es que los datos se vean en el depurador del lenguaje C#, y lo segundo es que se puede realizar un código en html para embeber una página Web en este. La cual solo es posible hacerla a través del puerto 80, que es el puerto reservado para este dispositivo. Si por ejemplo en el código del ejemplo de las figuras 4 a 7, se cambia a cualquier otro valor de puerto, este no funcionara.

Además, lo que él va a recibir en el socket son los comandos de instrucción para la formación del código html. Es decir que para el proyecto, no se podrá ejecutar porque lo que se necesita es que reciba datos por el socket (en este caso), emitidos desde el PC.

Por lo que esto va en contra de lo que se pide en una comunicación EtherNet/IP, que es el puerto AF12 hexadecimal o 44818 en decimal.

En una comunicación CIP punto a punto, los dos dispositivos son productores y consumidores. Otra restricción que presenta la ND+, no es capaz de situarse en otra dirección IP diferente a la que se asigna el servidor DHCP. Veamos lo que sucedió con el siguiente código.

Codigo ND+:

```
Public class Program
    {
Public static void Main()
        {
OutputPort led = new OutputPort(Pins.ONBOARD_LED, false);
```

```
Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Parse("192.168.1.100"),44818);
//Direccion IP del PC
sck.Connect(localEndPoint);
try
                {
sck.Connect(localEndPoint);
string Sender_Context = "Conectar";

byte[] data = Encoding.UTF8.GetBytes(Sender_Context);

sck.Send(data);
Debug.Print("Sender_Context enviado");
sck.Close();
                }
catch
                {
Debug.Print("imposible conectar con el PC \r\n");
                }


        }

    }
```

Código propio

Código PC:

```
ClassProgram
{
Static void Main(string[] args)
        {
Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint localEndPoint = newIPEndPoint(IPAddress.Any, 44818);
sck.Bind(localEndPoint);
sck.Listen(100);
Socket clientSocket;
while (true)
            {
clientSocket = sck.Accept();

Byte[] buffer = newByte[100];
clientSocket.Receive(buffer);
string request = newstring(Encoding.UTF8.GetChars(buffer));
////////////////////////////////////
Console.Write(request);

}

}

}
```

Código propio

El cual arrojo un resultado negativo, no conecto ni envío el mensaje.

Por lo que se concluye que desde este dispositivo es imposible conectarse al PC, de esta forma.

3.3 Aprendizaje de proyectos de interfaz gráfica en C#:

Una interfaz gráfica de usuario (GUI) permite a un usuario interactuar con un programa en forma visual; además proporciona a un programa una "apariencia visual" única. Al proporcionar distintas aplicaciones en las que los componentes de la interfaz de usuario sean consistentes e intuitivos, los usuarios pueden conocer con mayor rapidez las aplicaciones y así volverse más productivos.

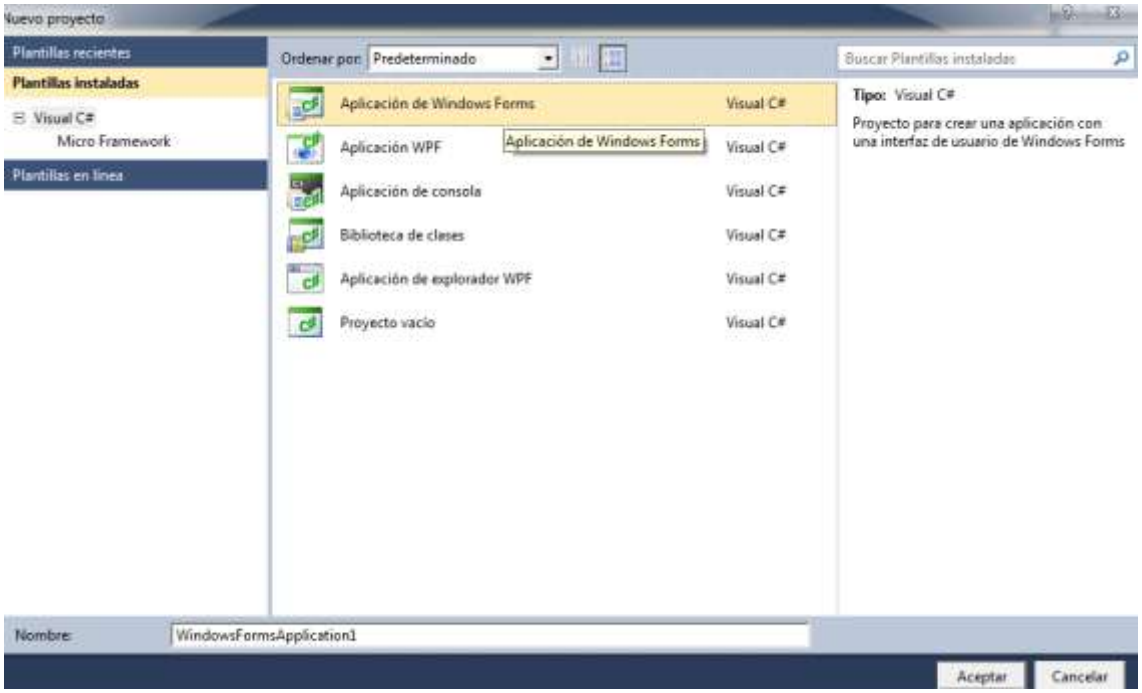
3.3.1 Formularios Windows Forms:

Los formularios Windows Forms se utilizan para crear las GUIs para los programas. Un formulario (Form) es un elemento grafico que aparece en el escritorio de una computadora; puede ser un cuadro de dialogo o una ventana.

Las GUIs se crean a partir de controles de controles de la GUI (conocidos como componentes o widgets [accesorios de ventana]). Los controles de la GUI son objetos que pueden demostrar información en la pantalla o que permita a los usuarios interactuar con una aplicación a través del ratón o del teclado.

Para comenzar un proyecto GUI en C#, abrimos el Microsoft Visual Studio, damos nuevo proyecto, y Aplicación de Windows Forms.

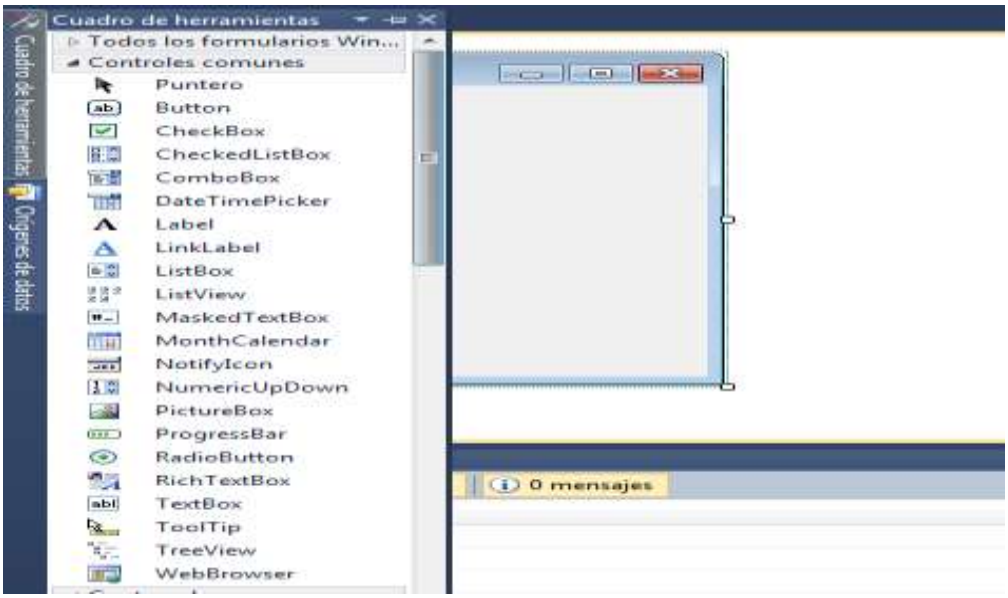
Figura 19: Inicio de un proyecto Windows Forms



Fuente propia

Ahora se podrá visualizar una plantilla en la cual se podrán anexar todos los controles GUI para la ejecución de un proyecto. Si la damos click en el cuadro de herramientas, aparecerán los controles comunes de la GUI.

Figura 20: Controles Gui

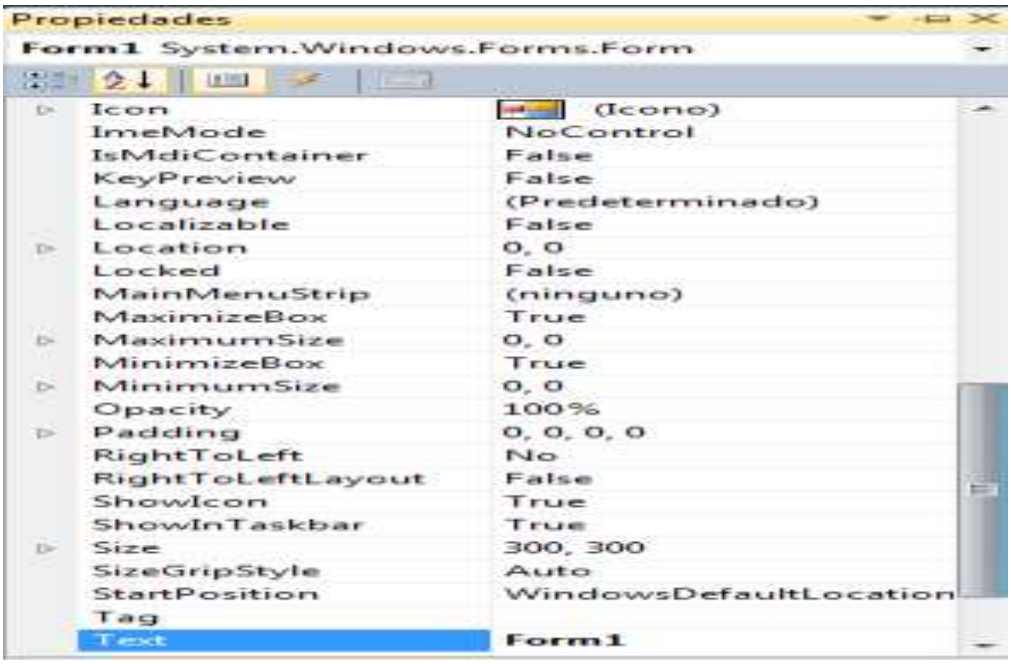


Fuente propia.

Para comenzar a realizar un programa de esta categoría, se va arrastrando con el mouse cada controlador a la plantilla.

En el lado derecho de la pantalla, se desplegara un menú de propiedades en cual se podrá ajustar cambios gráficos de cada controlador.

Figura 21: Propiedades de los controladores GUI



Fuente propia

Hay se podrá cambiar a modo de programador accesorios como color, tipo de letra, tipo de fondo, color de fondo, etc.

Por ejemplo el ítem “(Name)”, se encargara de llevar el nombre con que se denominara en el código fuente del programa, y el ítem “Text” el nombre de visualización del controlador en la plantilla.

Los controladores más básicos que se utilizan en este tipo de proyectos son los siguientes:

3.3.1.1 Label: Muestra imágenes o texto que no pueden editarse

3.3.1.2 TextBox: Permite al usuariointroducir datos mediante el teclado. También puede usarse para mostrar texto que puede o no puede editarse.

3.3.1.3 Panel: Un contenedor en el que se pueden colocarse y organizarse controles.

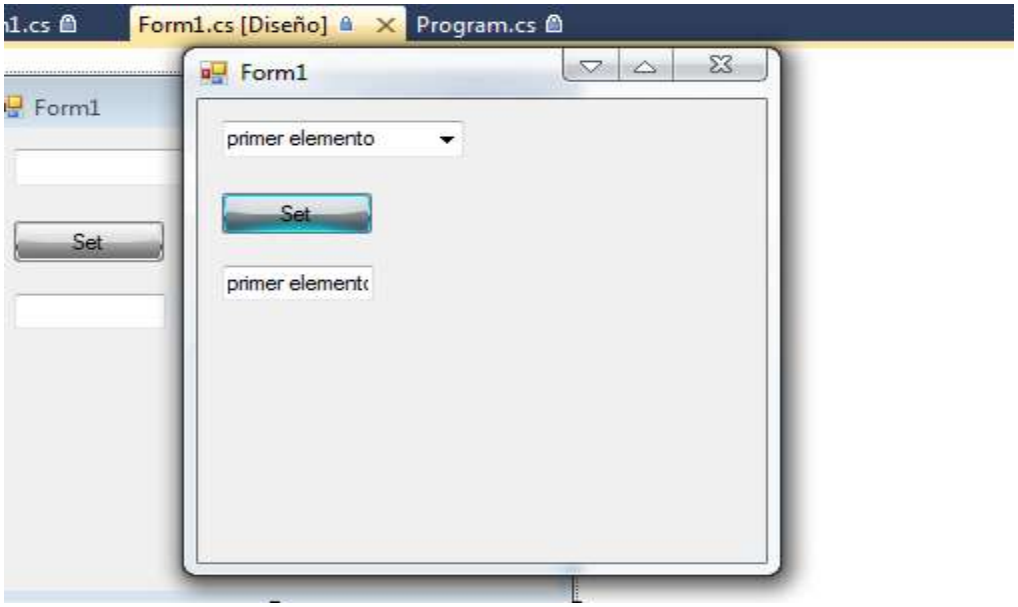
3.3.1.4 Button: Activa un evento cuando se hace click con el ratón.

3.3.1.5 CheckBox: Especifica unaopción que puede seleccionarse (activada) o deseleccionarse (desactivada).

3.3.1.6 ComboBox: Proporciona una lista desplegable de elementos, de los cuales el usuario puede seleccionar uno, haciendo click en un elemento de la lista o escribiendo dentro de un cuadro.

En el siguiente ejemplo, vamos a ver un ejemplo del ComboBox. Hay un button (Set) que adiciona el elemento seleccionado del ComboBox a un TextBox.

Figura 22: Ejemplo ComboBox



Fuente propia

Ahora como cada controlador se le programa un código fuente, este es el código para este programa.

Codigo ComboBox:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Combobox
{
```

```
Public partial class Form1 :Form
{
public Form1()
{
InitializeComponent();
comboBox1.Items.Add("primer elemento");//adicionando elementos al combobox
comboBox1.Items.Add("segundo elemento");

        }

Privatevoid cmdSET_Click(object sender, EventArgs e)//Esta funcion se crea sola,
dando doble click al button en la plantilla
{
if (comboBox1.SelectedIndex == 0)
            {
                txtSet.Text = "primer elemento";
}
if (comboBox1.SelectedIndex == 1)//Se van adicionando al textbox segun el item
correspondiente
            {
                txtSet.Text = "segundo elemento";
            }
        }
    }
}
```

Código propio

3.3.1.7 LixtBox: Proporciona una lista de elementos, de los cuales el usuario puede seleccionar uno, haciendo click en un elemento de la lista. Pueden seleccionarse varios elementos a la vez.

En el siguiente ejemplo, se anexaran números a un ListBox, además hay un opción que permite limpiar la lista de números.

Figura 23: Ejemplo ListBox

Fuente propia

Código ListBox:

```
namespace TutorialListBox1
{
Public partial class Form1 :Form
    {
public Form1()
        {
InitializeComponent();
        }

Private void btnAgregar_Click(object sender, EventArgs e)
{
int Numero;
            Numero = Int32.Parse(txtNumero.Text);//Convierte a INT lo que se agrega
al TextBox
listBox1.Items.Add(Numero);
txtNumero.Clear();
        }

Private void btnLimpiar_Click(object sender, EventArgs e)
{
listBox1.Items.Clear();
        }


        }
    }
```

Código propio

Bueno y de esta manera ya se tienen todos los elementos aprendidos para la codificación del proyecto

ANEXO D: CÓDIGO FUENTE EN C# DEL PROYECTO IMPLANTACIÓN DEL PROTOCOLO CIP SOBRE UNA PLATAFORMA EMBEBIDA PARA ETHERNET/IP.

El código se hizo de dos formas, en forma GUI y en modo consola. En él se pueden activar los atributos CIP del PC y la ND+. Lo sockets con los que se direcciona la ND+ y el PC, se hizo a tal modo que se utilizara los comandos de encapsulación y el path del Message Router.

4.1 Código en Windows Form Productor PC :

4.1.1 Clase Identity del PC:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ProtocoloCIP
{
Static class Identity
    {
Public static int Vendor_ID
        {
get { return 2050; }
        }
Public static int Device_Type
        {
get { return 110; }
        }
Public static int Product_Code
        {
get { return 2070; }
        }
Public static int Major_Revision
        {
get { return 2; }
        }
Public static int Minor_Revision
        {
get { return 1; }
        }
Public static int Status
        {
get { return 0; }
        }
Public static string Serial_Number
        {
get { return"F45U"; }
        }
Public static string Product_Name
{
get { return"Axus"; }
        }
    }
}
```

Código propio

4.1.2 Clase TCP/IP Interface del PC:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ProtocoloCIP
{
Static class TCPInterface
    {
Public static int Status
        {
get { return 1; }
        }

Public static int Configuration_Capability
```

```
        {
get { return 2; }
        }

Public static int Configuration_Control
        {
get { return 2; }
        }
Public static string PhysicalLinkObject
        {
get { return"21F62401"; }
        }

Public static string IPaddress
        {
get { return"192.168.1.100"; }
        }

Public static string NameServer
        {
get { return"192.168.1.1"; }
        }

Public static string NameServer2
        {
get { return"0.0.0.0"; }
}
}
}
```

Código propio

4.1.3 Clase Ethernet Link del PC:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ProtocoloCIP
{
Static class EthernetLink
    {
Public static string Interface_Speed
        {
get { return"1.433 Mbbps"; }
        }

Public static int Interface_Flags
        {
get { return 0; }
        }
Public static string Physical_Address
        {
get { return"DC-85-DE-14-6B-B4"; }
}
```

```
    }
}
```

Código propio

4.1.4 Clase ClientePC :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace ProtocoloCIP
{
Public class ClientePC
    {
Socket SendUniData;
IPEndPoint RegisterSession;

public ClientePC()
       {
            SendUniData = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            RegisterSession = new IPEndPoint(IPAddress.Parse("192.168.1.102"),
44818); ///44818 puerto AF12 que garantiza comunicacion EtherNet/IP
SendUniData.Connect(RegisterSession);
       }

Public void enviar(String Sender_Context)////Solicitud RegisterSesion
       {
try
                {
if (Sender_Context.ToUpper() != "102")
{
byte[] data = Encoding.UTF8.GetBytes(Sender_Context);
SendUniData.Send(data);
                    }
                }
catch
                {
Console.WriteLine("imposible conectar con la netduino");
}
        }

        ~ClientePC()
        {
try
            {
SendUniData.Close();
            }
catch
            {
Console.WriteLine("imposible cerrar con la netduino");
            }

        }
    }
```

}
Código propio

4.1.5 Código Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ProtocoloCIP
{
Public partial class Form1 : Form
    {
public Form1()
        {
InitializeComponent();

comboBox2.Items.Add("01 Hex");
comboBox2.Items.Add("F5 Hex");
comboBox2.Items.Add("F6 Hex");
comboBox2.Items.Add("100 Hex");
//////////////////////////
comboBox3.Items.Add("1");
comboBox3.Items.Add("2");
comboBox3.Items.Add("3");
comboBox3.Items.Add("4");
comboBox3.Items.Add("5");
comboBox3.Items.Add("6");
comboBox3.Items.Add("7");
//////////////////////////
comboBox1.Items.Add("20012401");
comboBox1.Items.Add("20012402");
comboBox1.Items.Add("20012403");
comboBox1.Items.Add("200124043001");
comboBox1.Items.Add("200124043002");
comboBox1.Items.Add("20012405");
comboBox1.Items.Add("20012406");
comboBox1.Items.Add("20012407");

comboBox1.Items.Add("20f52401");
comboBox1.Items.Add("20f52402");
comboBox1.Items.Add("20f52403");
comboBox1.Items.Add("20f52404");
comboBox1.Items.Add("20f524053001");
comboBox1.Items.Add("20f524053002");
comboBox1.Items.Add("20f524053003");
comboBox1.Items.Add("20f52406");

comboBox1.Items.Add("20f62401");
comboBox1.Items.Add("20f62402");
comboBox1.Items.Add("20f62403");

comboBox1.Items.Add("201002401");
```

```
comboBox1.Items.Add("UnRegisterSession");
//////////////////////////////////////////////
comboBox4.Items.Add("21012401");
comboBox4.Items.Add("21012402");
comboBox4.Items.Add("21012403");
comboBox4.Items.Add("210124043001");
comboBox4.Items.Add("210124043002");
comboBox4.Items.Add("21012405");
comboBox4.Items.Add("21012406");
comboBox4.Items.Add("21012407");

comboBox4.Items.Add("21f52401");
comboBox4.Items.Add("21f52402");
comboBox4.Items.Add("21f52403");
comboBox4.Items.Add("21f52404");
comboBox4.Items.Add("21f524053001");
comboBox4.Items.Add("21f524053002");
comboBox4.Items.Add("21f524053003");
comboBox4.Items.Add("21f52406");

comboBox4.Items.Add("21f62401");
comboBox4.Items.Add("21f62402");
comboBox4.Items.Add("21f62403");

comboBox4.Items.Add("UnRegisterSession");
        }


Private void btnAceptar_Click(object sender, EventArgs e)
        {
if(comboBox1.SelectedIndex==0 && comboBox2.SelectedIndex==0 &&
comboBox3.SelectedIndex==0){
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Vendor ID";
Console.WriteLine("Activar atributo Vendor ID de la Netduino");
textBox1.Text = "20012401";


            }
if (comboBox1.SelectedIndex == 1 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Device Type";
Console.WriteLine("Activar atributo Device Type de la Netduino");
textBox1.Text = "20012402";

            }
if (comboBox1.SelectedIndex == 2 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Product Code";
Console.WriteLine("Activar atributo Product Code de la Netduino");
textBox1.Text = "20012403";

            }
```

```
if (comboBox1.SelectedIndex == 3 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Revision-Major revision";
Console.WriteLine("Activar atributo Major revision de la Netduino");
textBox1.Text = "200124043001";

            }
if (comboBox1.SelectedIndex == 4 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Revision-Minor revision";
Console.WriteLine("Activar atributo Minor revision de la Netduino");
textBox1.Text = "200124043002";
            }

if (comboBox1.SelectedIndex == 5 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Status";
Console.WriteLine("Activar atributo Status del objeto identity de la Netduino");
textBox1.Text = "20012405";
            }
if (comboBox1.SelectedIndex == 6 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 5)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Serial Number";
Console.WriteLine("Activar atributo Serial Number de la Netduino");
textBox1.Text = "20012406";
            }
if (comboBox1.SelectedIndex == 7 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 6)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Product name";
Console.WriteLine("Activar atributo Product Name de la Netduino");
textBox1.Text = "20012407";
            }
//////////////////////////////////////////////////////////////////////
if (comboBox1.SelectedIndex == 8 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 0)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Status";
Console.WriteLine("Activar atributo Status del objeto TCPinterface de la Netduino");
InitializeComponent();
            }
if (comboBox1.SelectedIndex == 9 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Configuration Capability";
Console.WriteLine("Activar atributo Configuration Capability de la Netduino");
textBox1.Text = "20f52402";
```

```
            }
if (comboBox1.SelectedIndex == 10 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Configuration Control";
Console.WriteLine("Activar atributo Configuration Control de la Netduino");
textBox1.Text = "20f52403";
            }
if (comboBox1.SelectedIndex == 11 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Physical Link Object";
Console.WriteLine("Activar atributo Physical Link Object de la Netduino");
                textBox1.Text = "20f52404";
            }

if (comboBox1.SelectedIndex == 12 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "IP address";
Console.WriteLine("Activar atributo IP Adress de la Netduino");
textBox1.Text = "20f524053001";
            }
if (comboBox1.SelectedIndex == 13 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Name server";
Console.WriteLine("Activar atributo Name server de la Netduino");
textBox1.Text = "20f524053002";
            }
if (comboBox1.SelectedIndex == 14 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Name server 2";
Console.WriteLine("Activar atributo Name server 2 de la Netduino");
textBox1.Text = "20f524053003";
            }
if (comboBox1.SelectedIndex == 15 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 5)
            {
                txtSet1.Text = "Objeto TCPinterface";
                txtSet.Text = "Host Name";
Console.WriteLine("Activar atributo Host Name de la Netduino");
textBox1.Text = "20f52406";
            }
////////////////////////////////////////////////////////////////
if (comboBox1.SelectedIndex == 16 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 0)
            {
                txtSet1.Text = "Objeto Ethernet Link";
                txtSet.Text = "Interface Speed";
Console.WriteLine("Activar atributo Interface Speed de la Netduino");
textBox1.Text = "20f62401";
```

```
InitializeComponent();
            }
if (comboBox1.SelectedIndex == 17 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto Ethernet Link";
txtSet.Text = "Interface Flags";
Console.WriteLine("Activar atributo Interface Flags de la Netduino");
textBox1.Text = "20f62402";
InitializeComponent();
            }

if (comboBox1.SelectedIndex == 18 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto Ethernet Link";
                txtSet.Text = "Physical address";
Console.WriteLine("Activar atributo Physical Address de la Netduino");
textBox1.Text = "20f62401";
            }

////////////////////////////////////////////////////////////
if (comboBox1.SelectedIndex == 19 && comboBox2.SelectedIndex == 3 &&
comboBox3.SelectedIndex == 0)
            {
                txtSet1.Text = "Objeto Netduino";
                txtSet.Text = "Blinkled";
Console.WriteLine("Activar atributoBlinkled de la Netduino");
textBox1.Text = "201002401";
            }


if (comboBox1.SelectedIndex == 20)
            {
                textBox2.Text = "102";
            }
////////////////////////////////////////////////////////////////////////////////
/

if (comboBox4.SelectedIndex == 0 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 0)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Vendor ID";
list.Items.Add("Atributo Vendor ID del PC");
list.Items.Add(Identity.Vendor_ID);
                textBox1.Text = "21012401";
            }

if (comboBox4.SelectedIndex == 1 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Device Type";
list.Items.Add("Atributo Device Type del PC");
list.Items.Add(Identity.Device_Type);
                textBox1.Text = "21012402";
            }
```

```
if (comboBox4.SelectedIndex == 2 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Product Code";
list.Items.Add("Atributo Product Code del PC");
list.Items.Add(Identity.Product_Code);
                textBox1.Text = "21012403";
            }
if (comboBox4.SelectedIndex == 3 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Major revision";
list.Items.Add("Atributo Major revision del PC");
list.Items.Add(Identity.Major_Revision);
                textBox1.Text = "210124043001";
            }
if (comboBox4.SelectedIndex == 4 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Minor revision";
list.Items.Add("Atributo Minor revision del PC");
list.Items.Add(Identity.Minor_Revision);
                textBox1.Text = "210124043002";
            }
if (comboBox4.SelectedIndex == 5 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Status";
list.Items.Add("Atributo Status del PC");
list.Items.Add(Identity.Status);
                textBox1.Text = "21012405";
            }
if (comboBox4.SelectedIndex == 6 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 5)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Serial Number";
list.Items.Add("Atributo Serial Number del PC");
list.Items.Add(Identity.Serial_Number);
                textBox1.Text = "21012406";
            }
if (comboBox4.SelectedIndex == 7 && comboBox2.SelectedIndex == 0 &&
comboBox3.SelectedIndex == 6)
            {
                txtSet1.Text = "Objeto Identity";
                txtSet.Text = "Product Name";
list.Items.Add("Atributo Product Name del PC");
list.Items.Add(Identity.Product_Name);
                textBox1.Text = "21012407";
            }
//////////////////////
if (comboBox4.SelectedIndex == 8 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 0)
```

```
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Status";
list.Items.Add("Atributo Status del TCPinterface del PC");
list.Items.Add(TCPInterface.Status);
                textBox1.Text = "21f52401";
            }
if (comboBox4.SelectedIndex == 9 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Configuration Capability";
list.Items.Add("Atributo Configuration Capability  del PC");
list.Items.Add(TCPInterface.Configuration_Capability);
                textBox1.Text = "21f52402";
            }
if (comboBox4.SelectedIndex == 10 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Configuration Control";
list.Items.Add("Atributo Configuration Control del PC");
list.Items.Add(TCPInterface.Configuration_Control);
                textBox1.Text = "21f52403";
            }
if (comboBox4.SelectedIndex == 11 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 3)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Physical Link Object";
list.Items.Add("Atributo Physical Link Object del PC");
list.Items.Add(TCPInterface.PhysicalLinkObject);
                textBox1.Text = "21f52403";
            }
if (comboBox4.SelectedIndex == 12 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "IP Address";
list.Items.Add("Atributo IP Address del PC");
list.Items.Add(TCPInterface.IPaddress);
                textBox1.Text = "21f524053001";
            }
if (comboBox4.SelectedIndex == 13 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Name server";
list.Items.Add("Atributo Name server del PC");
list.Items.Add(TCPInterface.NameServer);
                textBox1.Text = "21f524053002";
            }
if (comboBox4.SelectedIndex == 14 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 4)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Name server2";
list.Items.Add("Atributo Name server2 del PC");
```

```
list.Items.Add(TCPInterface.NameServer2);
                textBox1.Text = "21f524053003";
            }
if (comboBox4.SelectedIndex == 15 && comboBox2.SelectedIndex == 1 &&
comboBox3.SelectedIndex == 5)
            {
                txtSet1.Text = "Objeto TCP/IP Interface";
                txtSet.Text = "Host name";
list.Items.Add("Atributo Host Name del PC");
list.Items.Add(TCPInterface.HostName);
                textBox1.Text = "21f52406";
            }
//////////////////////////////////////////
if (comboBox4.SelectedIndex == 16 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 0)
            {
                txtSet1.Text = "Objeto Ethernet Link";
                txtSet.Text = "Interface Speed";
list.Items.Add("Atributo Interface Speed del PC");
list.Items.Add(EthernetLink.Interface_Speed);
                textBox1.Text = "21f62401";
            }
if (comboBox4.SelectedIndex == 17 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 1)
            {
                txtSet1.Text = "Objeto Ethernet Link";
                txtSet.Text = "Interface Flags";
list.Items.Add("Atributo Interface Flags del PC");
list.Items.Add(EthernetLink.Interface_Flags);
                textBox1.Text = "21f62402";
            }
if (comboBox4.SelectedIndex == 18 && comboBox2.SelectedIndex == 2 &&
comboBox3.SelectedIndex == 2)
            {
                txtSet1.Text = "Objeto Ethernet Link";
                txtSet.Text = "Physical Address";
list.Items.Add("Atributo Physical Address del PC");
list.Items.Add(EthernetLink.Physical_Address);
                textBox1.Text = "21f62403";
            }
if (comboBox4.SelectedIndex == 19)
{
textBox2.Text = "102";
            }
        }

Private void button1_Click(object sender, EventArgs e)
        {
ClientePC miCliente = newClientePC();
String dato = textBox1.Text;
miCliente.enviar(dato);
        }

Private void button2_Click(object sender, EventArgs e)
        {
list.Items.Clear();
        }
```

```
Private void btnTerminar_Click(object sender, EventArgs e)
        {
ClientePC miCliente = new ClientePC();
String dato = textBox2.Text;
miCliente.enviar(dato);
        }

        }
    }
```

Código propio

4.2 Código en modo consola:

Nota: Las clases y sus respectivos atributos son los mismos que el de Windows Form

4.2.1 Código Program.cs PC:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace ClientePCCIP
{
ClassProgram
    {
Static void Main(string[] args)
        {
Socket SendUniData = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint RegisterSession = new IPEndPoint(IPAddress.Parse("192.168.1.102"),
44818); ///44818 puerto AF12 que garantiza comunicacion EtherNet/IP
try{
SendUniData.Connect(RegisterSession);

string Sender_Context = "";


while (Sender_Context != "102")
                {
Console.Write("\r\n Inserte el Packet Epath del Message Router: \r\n ");
                    Sender_Context = Console.ReadLine();


byte[] data = Encoding.UTF8.GetBytes(Sender_Context);

if (Sender_Context == "20012401")
                        {
SendUniData.Send(data);
Console.Write("activar atributo vendor_ID \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
```

```
Main(args);
                        }
if (Sender_Context == "20012402")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Device Type \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20012403")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Product Code \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }
if (Sender_Context == "200124043001")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Major revision\r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "200124043002")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Minor revision \r\n" + "tipo ID= A1");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n");
Main(args);

                        }
if (Sender_Context == "20012405")
                        {
SendUniData.Send(data);
Console.Write("activar atributo status del objeto identity\r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20012406")
                        {
SendUniData.Send(data);
Console.Write("activar atributo serial number \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }
if (Sender_Context == "20012407")
                        {
SendUniData.Send(data);
Console.Write("activar atributo product name \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
//////////////////////////////////////////////////////////////////////////////
if (Sender_Context == "20f52401")
                        {
SendUniData.Send(data);
```

```
Console.Write("activar atributo Status del TCPinterface \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f52402")
                        {
SendUniData.Send(data);
Console.Write("activar atributo configuration Capability \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }

if (Sender_Context == "20f52403")
                        {
SendUniData.Send(data);
Console.Write("activar atributo configuration Control \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f52404")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Physical Link Object \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f524053001")
                        {
SendUniData.Send(data);
Console.Write("activar atributo IP Address \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f524053002")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Name server \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f524053003")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Name server 2 \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
if (Sender_Context == "20f52406")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Host Name \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);
                        }
//////////////////////////////////////////////////////
if (Sender_Context == "20f62401")
                        {
SendUniData.Send(data);
```

```
Console.Write("activar atributo Interface speed \r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }
if (Sender_Context == "20f62402")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Interface Flags\r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }
if (Sender_Context =="20f62403")
                        {
SendUniData.Send(data);
Console.Write("activar atributo Physical address\r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }
if (Sender_Context == "201002401")
                        {
SendUniData.Send(data);
Console.Write("activar Netduino\r\n");
Console.WriteLine("options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
Main(args);

                        }

//////////////////////////////////////////////////////////////////
if (Sender_Context == "21012401")
{
SendUniData.Send(data);
Console.Write("VendorID \r\n"+Identity.Vendor_ID);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21012402")
                        {
SendUniData.Send(data);
Console.Write("Device Type \r\n"+Identity.Device_Type);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21012403")
                        {
SendUniData.Send(data);
Console.Write("Product Code \r\n"+Identity.Product_Code);
Console.WriteLine("0 Realizado\r\n");
Main(args);

                        }
if (Sender_Context == "210124043001")
                        {
SendUniData.Send(data);
Console.Write(" Major revision \r\n"+Identity.Major_Revision);
Console.WriteLine("0 Realizado\r\n");
```

```
Main(args);
                        }
if (Sender_Context == "210124043002")
                        {
SendUniData.Send(data);
Console.Write(" Minor revision \r\n"+Identity.Minor_Revision);
Console.WriteLine("0 Realizado\r\n");
Main(args);

                        }
if (Sender_Context == "21012405")
                        {
SendUniData.Send(data);
Console.Write("status del objeto identity \r\n"+Identity.Status);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21012406")
                        {
SendUniData.Send(data);
Console.Write("serial number \r\n" + Identity.Serial_Number);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21012407")
                        {
SendUniData.Send(data);
Console.Write("Product name \r\n"+Identity.Product_Name);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
//////////////////////////////////////////////////////////////////////////////
if (Sender_Context == "21f52401")
                        {
SendUniData.Send(data);
Console.Write("Status del TCPinterface \r\n"+TCPInterface.Status);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f52402")
                        {
SendUniData.Send(data);
Console.Write("Configuration Capability
\r\n"+TCPInterface.Configuration_Capability);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }

if (Sender_Context == "21f52403")
                        {
SendUniData.Send(data);
Console.Write("Configuration Control \r\n"+TCPInterface.Configuration_Control);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f52404")
                        {
SendUniData.Send(data);
```

```
Console.Write("Physical Link Object \r\n"+TCPInterface.PhysicalLinkObject);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f524053001")
                        {
SendUniData.Send(data);
Console.Write(" IP Address \r\n"+TCPInterface.IPaddress);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f524053002")
                        {
SendUniData.Send(data);
Console.Write(" Name server \r\n"+TCPInterface.NameServer);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f524053003")
                        {
SendUniData.Send(data);
Console.Write("Name server 2 \r\n"+TCPInterface.NameServer2);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
if (Sender_Context == "21f52406")
                        {
SendUniData.Send(data);
Console.Write("Host Name \r\n" + TCPInterface.HostName);
Console.WriteLine("0 Realizado\r\n");
Main(args);
                        }
//////////////////////////////////////////////////////
if (Sender_Context == "21f62401")
                        {
SendUniData.Send(data);
Console.Write(" Interface speed \r\n"+EthernetLink.Interface_Speed);
Console.WriteLine("0 Realizado\r\n");
Main(args);

                        }
if (Sender_Context == "21f62402")
                        {
SendUniData.Send(data);
Console.Write("Interface Flags \r\n"+EthernetLink.Interface_Flags);
Console.WriteLine("0 Realizado\r\n");
Main(args);

                        }
if (Sender_Context == "21f62403")
                        {
SendUniData.Send(data);
Console.Write("Physical address \r\n"+EthernetLink.Physical_Address);
Console.WriteLine("0 Realizado\r\n");
Main(args);

                        }
```

```
if (Sender_Context == "102") {
SendUniData.Send(data);
                        }
                    }

SendUniData.Close();
}
catch
            {
Console.Write("imposible conectar con la netduino");
            }

        }
    }
}
```

4.3 Código Netduino Consumidor:

Nota: Este funcionara tanto para en modo GUI o en el modo consola.

4.3.1 Clase Identity Netduino Plus:

```
using System;
using Microsoft.SPOT;
using System.Text;

namespace NetduinoConsumidor
{
StaticclassIdentity
    {
Publicstaticstring Vendor_ID
        {
get { return"2000"; }
        }
Publicstaticstring Device_Type
        {
get { return"120"; }
        }
Publicstaticstring Product_Code
        {
get { return"2020"; }
        }
Publicstaticstring Major_Revision
        {
get { return"2"; }
        }
Publicstaticstring Minor_Revision
        {
get { return"1"; }
        }
Publicstaticstring Status
        {
get { return"0"; }
        }
Publicstaticstring Serial_Number
        {
get { return"ND1"; }
```

```
        }
Publicstaticstring Product_Name
        {
get { return"Netduino Plus"; }
}

    }
}
```
Código propio

4.3.2 Clase TCP/IP Interface Netduino Plus:

```
using System;
using Microsoft.SPOT;

namespace NetduinoConsumidor
{
StaticclassTCPinterface
    {
Publicstaticstring Status
        {
get { return"1"; }
        }

Publicstaticstring Configuration_Capability
        {
get { return"2"; }
        }

Publicstaticstring Configuration_Control
        {
get { return"2"; }
        }
Publicstaticstring PhysicalLinkObject
        {
get { return"20F62401"; }
        }

Publicstaticstring IPaddress
        {
get { return"192.168.1.100"; }
        }

Public staticstring NameServer
        {
get { return"172.16.255.200"; }
        }

Publicstaticstring NameServer2
        {
get { return"0.0.0.0"; }
        }
    }
}
```
Código propio

4.3.3 Clase Ethernet Link Netduino Plus:

```
using System;
using Microsoft.SPOT;

namespace NetduinoConsumidor
{
StaticclassTCPinterface
    {
Publicstaticstring Status
        {
get { return"1"; }
        }

Publicstaticstring Configuration_Capability
        {
get { return"2"; }
        }

Publicstaticstring Configuration_Control
        {
get { return"2"; }
        }
Publicstaticstring PhysicalLinkObject
        {
get { return"20F62401"; }
        }

Publicstaticstring IPaddress
        {
get { return"192.168.1.100"; }
        }

Publicstaticstring NameServer
        {
get { return"172.16.255.200"; }
        }

Publicstaticstring NameServer2
{
get { return"0.0.0.0"; }
}
    }
}
```

Código propio

4.3.4 Código Program.cs Netduino Plus:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Text;
```

```
namespace NetduinoConsumidor
{
PublicclassProgram
    {
PublicstaticvoidMain()
        {
OutputPort led = newOutputPort(Pins.ONBOARD_LED, false);

Socket SendUniData = newSocket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint RegisterSession = newIPEndPoint(IPAddress.Any, 44818);
SendUniData.Bind(RegisterSession);
SendUniData.Listen(100);
Socket clientSocket;
while (true)
            {
clientSocket = SendUniData.Accept();

Byte[] buffer = newByte[100];
clientSocket.Receive(buffer);
string Sender_Context = newstring(Encoding.UTF8.GetChars(buffer));
//////////////////////////////////////

if (Sender_Context == "20012401")
                        {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Vendor_Id\r\n " + Identity.Vendor_ID);
                    }

if (Sender_Context == "20012402")
{
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo device type\r\n" + Identity.Device_Type);
                    }

if (Sender_Context == "20012403")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Product Code\r\n " + Identity.Product_Code);
                    }
if (Sender_Context == "200124043001")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("Instancia Revision, atributo Major  revision\r\n " +
Identity.Major_Revision);
                    }
if (Sender_Context == "200124043002")
{
Debug.Print("0 Realizado\r\n");
Debug.Print("Instancia Revision, atributo Minor  revision\r\n" +
Identity.Minor_Revision);
                    }
if (Sender_Context == "20012405")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo status\r\n" + Identity.Status);
                    }
```

```
if (Sender_Context == "20012406")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Serial Number\r\n" + Identity.Serial_Number);
                    }
if (Sender_Context == "20012407")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Product Name\r\n" + Identity.Product_Name);
                    }
////////////////////////////////////////////////
if (Sender_Context == "20f52401")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Status\r\n" + TCPinterface.Status);
                    }
if (Sender_Context == "20f52402")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Configuration Capability\r\n" +
TCPinterface.Configuration_Capability);
                    }
if (Sender_Context == "20f52403")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Configuration Control\r\n " +
TCPinterface.Configuration_Control);
                    }
if (Sender_Context == "20f52404")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Physical Link Object\r\n " + TCPinterface.PhysicalLinkObject);
                    }

if (Sender_Context == "20f524053001")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("Instancia Interface Configuration, atributo IP Address\r\n" +
TCPinterface.IPaddress);
                    }

if (Sender_Context == "20f524053002")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("Instancia Interface Configuration, atributo Name Server\r\n" +
TCPinterface.NameServer);
                    }

if (Sender_Context == "20f524053003")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("Instancia Interface Configuration, atributo Name Server 2\r\n" +
TCPinterface.NameServer2);
                    }

if (Sender_Context == "20f52406")
                     {
Debug.Print("0 Realizado\r\n");
```

```
Debug.Print("atributo HostName\r\n" + TCPinterface.HostName);
                    }
////////////////////////////////////////////////
if (Sender_Context == "20f62401")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Interface Speed\r\n" + Ethernet_Link.Interface_Speed);
                    }
if (Sender_Context == "20f62402")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Interface flags\r\n" + Ethernet_Link.Interface_Flags);
                    }
if (Sender_Context == "20f62403")
                    {
Debug.Print("0 Realizado\r\n");
Debug.Print("atributo Physical address\r\n" + Ethernet_Link.Physical_Address);
                    }

if (Sender_Context == "201002401")
                    {
while (true)
                        {
led.Write(true);
Thread.Sleep(1000);
led.Write(false);
Thread.Sleep(1000);
                        }
                    }

////////////////////////////////////////////////////////////////////
if (Sender_Context == "21012401")
                    {

Debug.Print("Se activo atributo Vendor_ID del PC\r\n" + "options 0 \r\n" + "status 0
\r\n" + "tipo ID= A1\r\n");

}
if(Sender_Context == "21012402")
                    {

Debug.Print("Se activo atributo Device Type del PC\r\n" + "options 0 \r\n" + "status
0 \r\n"+"tipo ID= A1\r\n");

                    }
if (Sender_Context == "21012403")
                    {

Debug.Print("Se activo Product Code del PC\r\n" + "options 0 \r\n" + "status 0 \r\n"
+ "tipo ID= A1\r\n");
}
if (Sender_Context == "210124043001")
                    {

Debug.Print("Se activo Major revision del PC\r\n" + "options 0 \r\n" + "status 0
\r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "210124043002")
```

```
                    {

Debug.Print("Se activo Minor revision del PC\r\n" + "options 0 \r\n" + "status 0
\r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21012405")
                    {

Debug.Print("Se activo atributo Status del objeto Identity del PC\r\n" + "options 0
\r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21012406")
                    {

Debug.Print("Se activo atributo Serial Number del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }

if (Sender_Context == "21012407")
                    {

Debug.Print("Se activo atributo Product Name del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }
//////
if (Sender_Context == "21f52401")
                    {

Debug.Print("Se activo atributo Status del objeto TCP/Interface del PC\r\n" +
"options 0 \r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f52402")
                    {

Debug.Print("Se activo atributo Configuration Capability del PC\r\n" + "options 0
\r\n" + "status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f52403")
                    {

Debug.Print("Se activo atributo Configuration Control del PC\r\n" + "options 0 \r\n"
+ "status 0 \r\n" + "tipo ID= A1\r\n");
}
if (Sender_Context == "21f52404")
                    {

Debug.Print("Se activo atributo Physical Link Object del PC\r\n" + "options 0 \r\n"
+ "status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f524053001")
                    {

Debug.Print("Se activo atributo IP Address Object del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f524053002")
                    {
```

```
Debug.Print("Se activo atributo Name server del PC\r\n" + "options 0 \r\n" + "status
0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f524053003")
                    {

Debug.Print("Se activo atributo Name server 2 del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f52406")
                    {

Debug.Print("Se activo atributo Host name  del PC\r\n" + "options 0 \r\n" + "status
0 \r\n" + "tipo ID= A1\r\n");
}
////////////////////
if (Sender_Context == "21f62401")
                    {

Debug.Print("Se activo atributo Interface Speed del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "21f62402")
                    {

Debug.Print("Se activo atributo Interface Flags del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
}
if (Sender_Context == "21f62403")
                    {

Debug.Print("Se activo atributo Physical Address del PC\r\n" + "options 0 \r\n" +
"status 0 \r\n" + "tipo ID= A1\r\n");
                    }
if (Sender_Context == "102") break;//UnregisterSession


                }
SendUniData.Close();
            }


//////////////////////////////////////////////
    }

        }
```