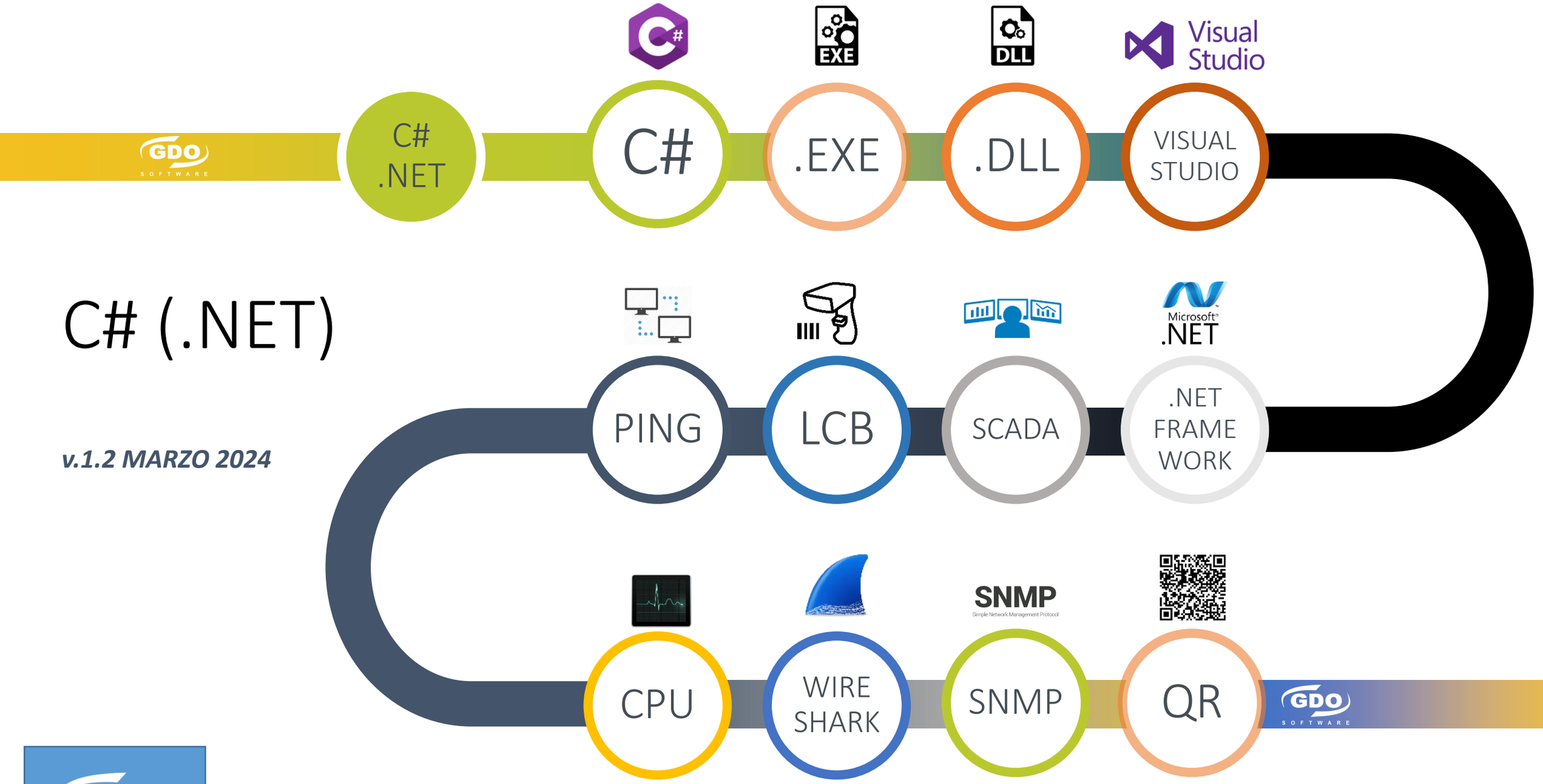
CAPÍTULO 3. C# (.NET)

v.1.2 MARZO 2024

Ricardo Moraleda Gareta

[Director departamento de software de GDO Software]



C# (.NET)

v.1.2 MARZO 2024



C# (.NET)


C# (C sharp)

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

https://docs.microsoft.com/es-es/dotnet/csharp/

En este tutorial no voy a enseñar a programar C#, simplemente ejemplos orientados al control industrial o funcionalidades típicas en este campo, tanto .EXE como .DLL (librería de clases o control de usuario)

Control industrial

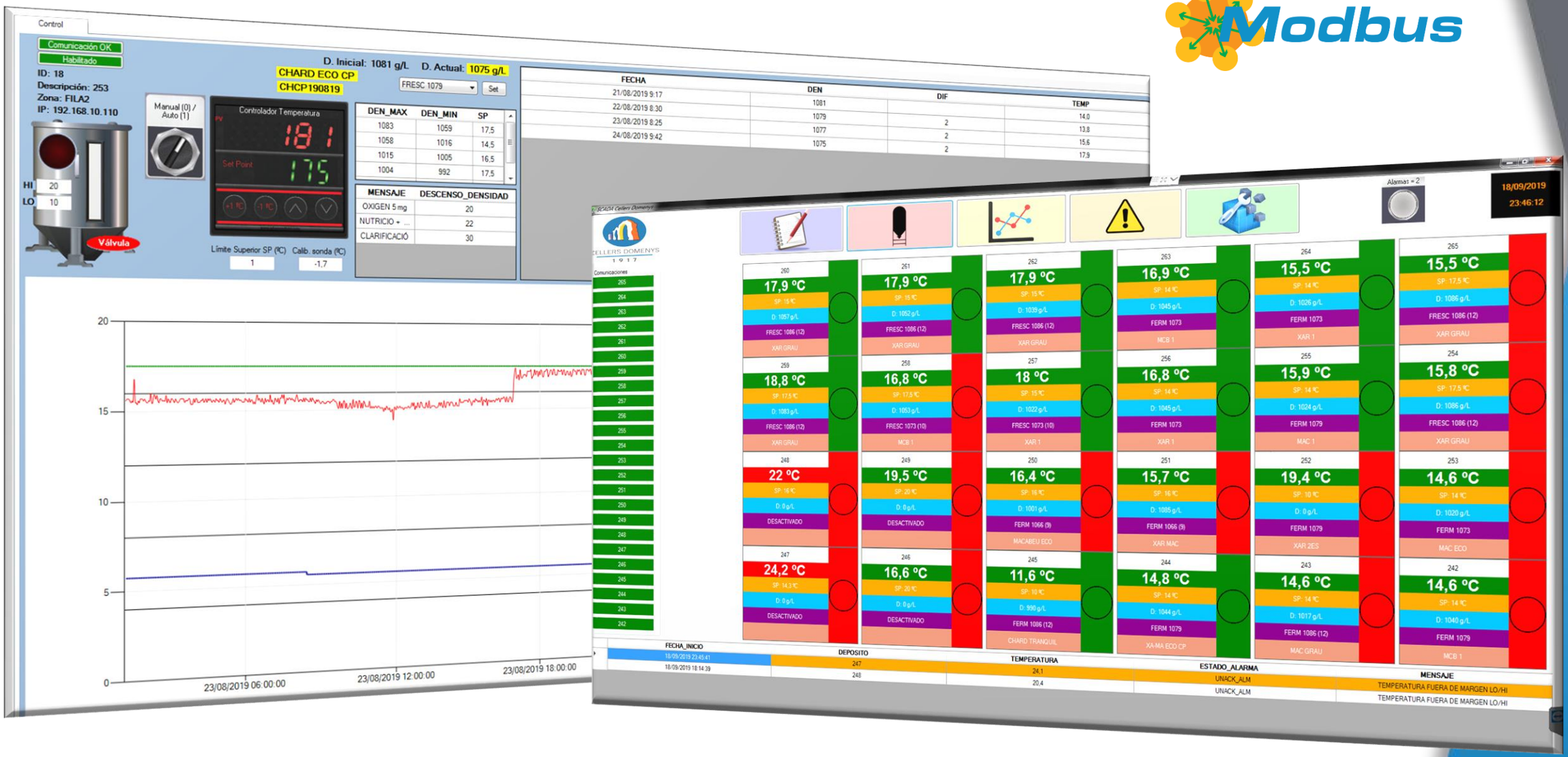
1. SCADA de control de sonda temperatura y válvula frío de depósitos de vino con tasks concurrentes y mediante el protocolo Modbus TCP.
2. Driver lectura código de barras mediante lector USB Datalogic.
3. Ping – ConsoleApplication
4. Monitor CPU del PC y registro en CSV
5. Driver lectura protocolo SNMP

Visual Studio Microsoft .NET

Para ello utilizaré el IDE Visual Studio 2015 y el framework .NET 4.5.2

https://visualstudio.microsoft.com/es/free-developer-offers/
(versión Community es gratis)

SCADA (Modbus TCP)

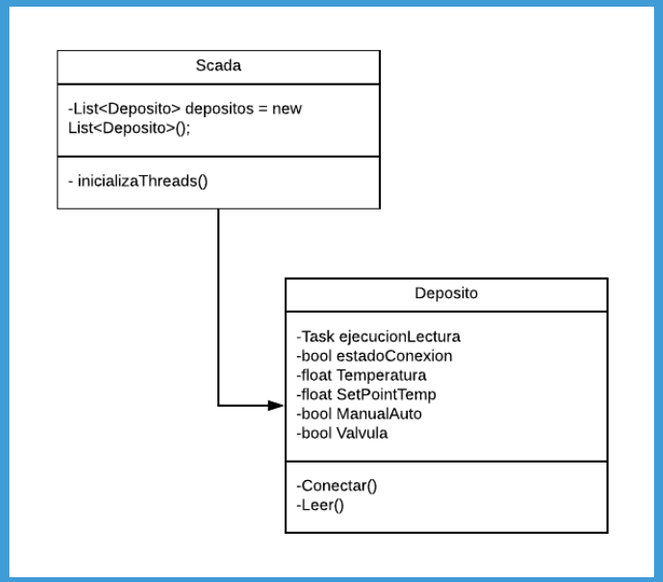


C# (.NET)

SCADA control depósitos vino

Aplicación con WindowsForm (.EXE) y base de datos SQL. En esta parte nos centraremos en C# de manera resumida y esquemática.

La clase **Scada** es estática y contiene una List de tipo Deposito previamente inicializada de BBDD.



Al llamar al método **inicializaThreads()** se generan tantas Tasks como depósitos haya y las inicializa.

La tarea o task hace referencia al método **Leer()** de la clase **Deposito**.

```
public static void inicializaThreads()
{
    try
    {

        foreach (Deposito deposito in Scada.depositos)
        {

            if (deposito.habilitado)
            {
                deposito.ejecucionLectura = new Task(() => deposito.Leer());
                deposito.ejecucionLectura.Start();
                Thread.Sleep(1009);
            }

        }

    }
    catch (Exception e)
    {
        Scada.Log.Error("Error al inicializar threads. " + e.ToString());
    }
}
```

C# (.NET)

El método **Leer()** de **Deposito** contiene un bucle infinito o while(true) donde llama a un método de lectura de datos **readMultipleWords()** por ModBus TCP pasando la conexión o socket, la posición a leer y la cantidad de datos.

```
public void Leer()
{
    try
    {
        while (true)
        {
            int[] datosLeidos = new int[12];//temperatura(2), setpoint temp (2), manualauto (1), abrircerrar valvula (1), cpu (1), ram (1), tempspup (2), tempspdown (2)
            int[] PV = new int[2];
            int[] SP = new int[2];
            int[] SPUPLEVEL = new int[2];
            int[] SPDOWNLEVEL = new int[2];

            if (estadoConexion)
            {
                //lee datos del iot por modbus tcp
                datosLeidos = Scada.factoriaModbus.readMultipleWords(this.serverSocket, 1, Convert.ToInt32(this.wordTempPV) - 1, datosLeidos.Length);
            }
            else
            {
                Thread.Sleep(Convert.ToInt32(this.scan) * 2);
                Conectar();
            }

            Thread.Sleep(Convert.ToInt32(this.scan));
         }

    }
    catch (Exception e)
    {
        Scada.Log.Error("[ID: " + this.descripcion + "] " + e.ToString());
    }
}
```

Esta función lee 12 posiciones de memoria del servidor modbus tcp de una tacada por cada ciclo de scan configurado – Sleep(scan).

En primera instancia, si está desconectado, llama a **Conectar()** que pondrá estadoConexion a true si lo está.

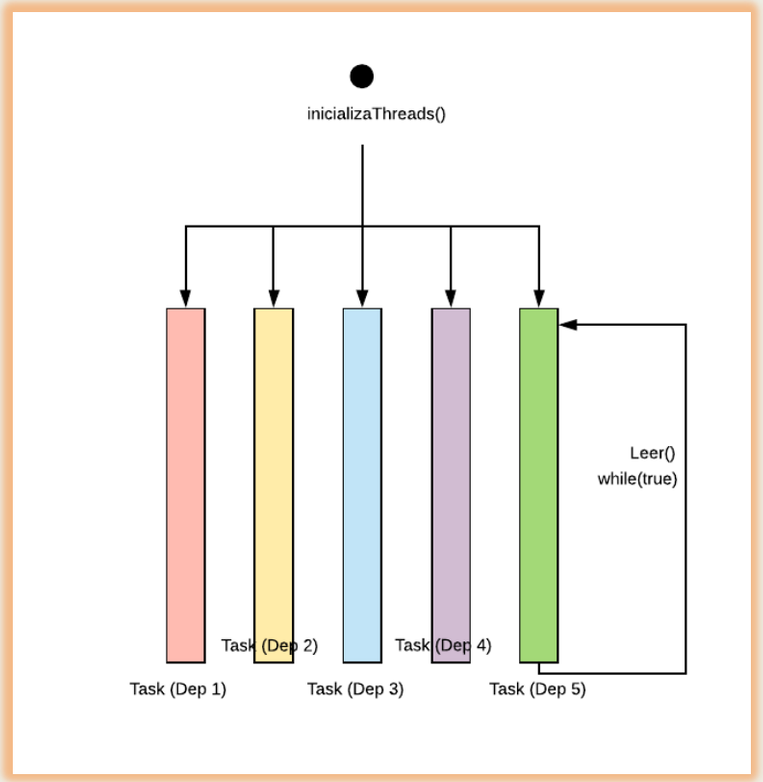
Esto ya es infinito hasta que pase un fallo capturado por la sentencia catch en la cual si es por fallo de comunicación se implementaría una reconexión, etc.

C# (.NET)

El método **Conectar()** hace un Connect() de un Socket pasando la IP del servidor modbus y el puerto que es el 502.

```
public bool Conectar()
{
    try
    {
        //IP y puerto destino
        IPEndPoint ip = new IPEndPoint(IPAddress.Parse(this.ip), this.puertoTCP);
        this.serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        this.serverSocket.SendTimeout = 5000;
        this.serverSocket.Connect(ip);
        this.estadoConexion = true;
        Scada.Log.Info("[ID: " + this.descripcion + "] Conectado satisfactoriamente a " + this.ip);
    }
    catch (Exception e)
    {
        this.estadoConexion = false;
        Scada.Log.Error("[ID: "+this.descripcion+"] " + e.ToString());
    }
    Scada.Log.Info("[ID: " + this.descripcion + "] Estado de la conexión con el PLC: " + this.estadoConexion);
    return this.estadoConexion;
}
```

En resumen, si tuviésemos 5 depósitos se generarían 5 tareas (hilos) en paralelo ejecutándose en cada una su método Leer() infinitamente.



C# (.NET)

Microsoft® .NET

El método **readMultipleWords()** de Modbus TCP se implementa de la siguiente manera.

```
public int[] readMultipleWords(Socket socket, int unidad, int referencia, int cantidad)
    {
        // Construir la trama Modbus/TCP
        buffer = new byte[12];
        byte[] buffer2 = new byte[cantidad * 2];
        int i = 0;

        for (i = 0; i < 5; i++)
        {
            buffer[i] = (byte)0;
        }

        buffer[5] = (byte)6;
        buffer[6] = (byte)unidad;
        buffer[7] = (byte)3;
        buffer[8] = (byte)(referencia >> 8);
        buffer[9] = (byte)(referencia & 0xFF);
        buffer[10] = (byte)0;
        buffer[11] = (byte)cantidad;

        int sendedDataLength = socket.Send(buffer); //cabecera

        buffer = new byte[137];
        int receivedDataLength = socket.Receive(buffer);

        for (i = 0; i < buffer2.Length; i++)
        {
            buffer2[i] = buffer[i + 9];
        }

        return getWords(buffer2);
    }
```

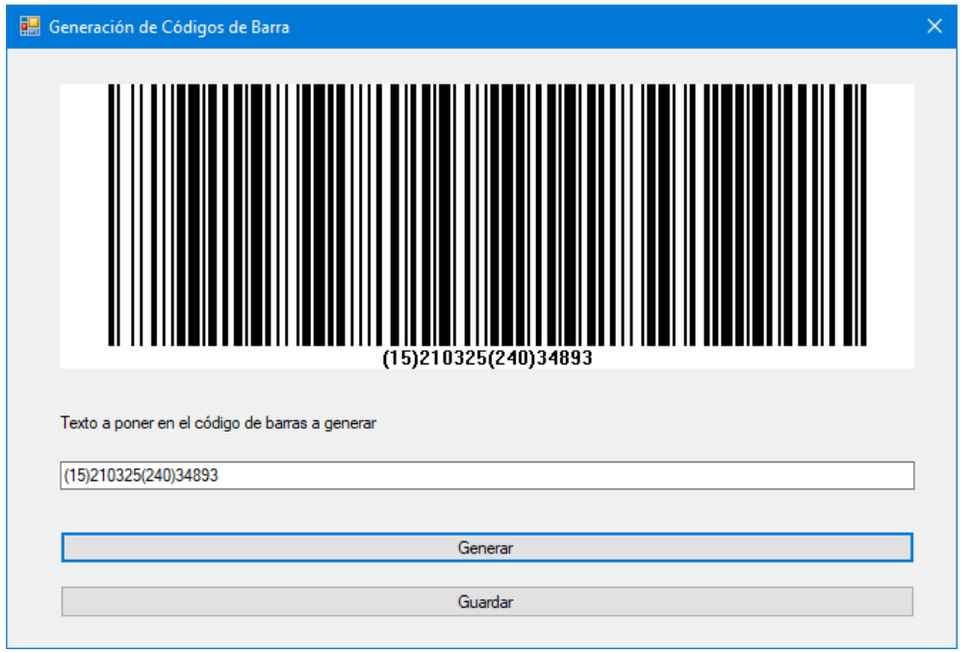
Byte 0	Identificador de transacción. Copiado por el servidor, normalmente 0.
Byte 1	Byte 1 Identificador de transacción. Copiado por el servidor, normalmente 0.
Byte 2	Byte 2 Identificador de protocolo = 0.
Byte 3	Byte 3 Identificador de protocolo = 0.
Byte 4	Byte 4 Campo de longitud (byte alto) = 0. Ya que los mensajes son menores a 256.
Byte 5	Byte 5 Campo de longitud (byte bajo). Número de bytes siguientes.
Byte 6	Byte 6 Identificador de unidad (previamente dirección esclavo.).
Byte 7	Byte 7 Código de función Modbus.
Byte 8 y más	Byte 8 y más Los datos necesarios.

Según protocolo se envía la siguiente trama de petición y se obtiene la respuesta con los 12 registros solicitados desde la posición indicada.

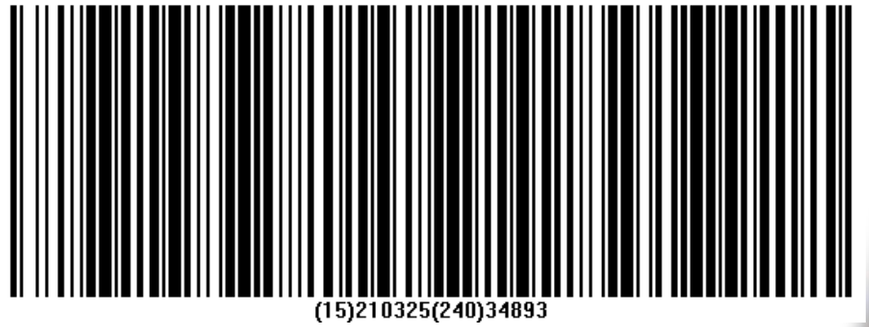
Captura (respuesta) de datos de red con Wireshark .

```
> Frame 337: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 502, Dst Port: 52178, Seq: 34, Ack: 25, Len: 33
v Modbus/TCP
    Transaction Identifier: 0
    Protocol Identifier: 0
    Length: 27
    Unit Identifier: 1
v Modbus
    .000 0011 = Function Code: Read Holding Registers (3)
    [Request Frame: 335]
    [Time from request: 0.000271000 seconds]
    Byte Count: 24
  > Register 99 (UINT16): 0
  v Register 100 (UINT16): 16772
      Register Number: 100
      Register Value (UINT16): 16772
  > Register 101 (UINT16): 0
  > Register 102 (UINT16): 16752
  > Register 103 (UINT16): 0
  > Register 104 (UINT16): 0
  > Register 105 (UINT16): 0
  > Register 106 (UINT16): 0
  > Register 107 (UINT16): 0
  > Register 108 (UINT16): 0
  > Register 109 (UINT16): 0
  > Register 110 (UINT16): 0
```

Generación Barcode



CÓDIGO DE BARRAS



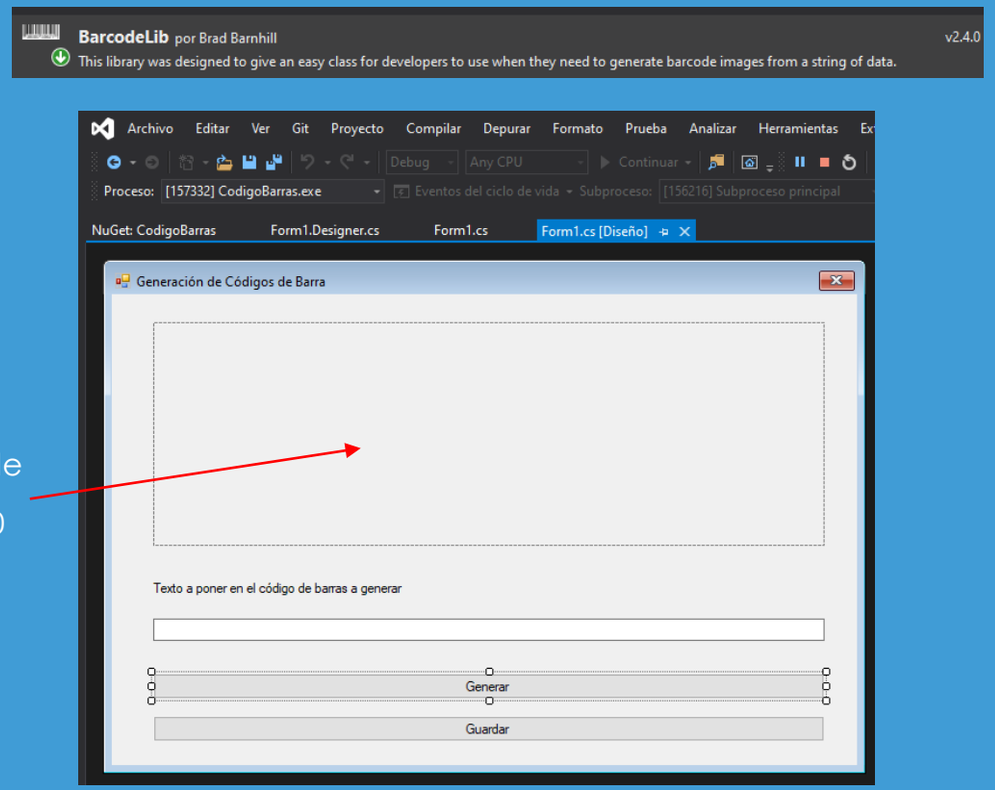
Basado en este enlace: https://www.youtube.com/watch?v=UJm_1BYn9hs

C# (.NET)

Microsoft® .NET

Generación código de barras

Se utiliza la librería BarcodeLib de Brad Barnhill. v2.4



Panel de 600x200

1° Se debe escribir el texto que se desee.

2° Se pulsa el botón Generar para visualizar el código de barras generado.

3° Se pulsa el botón Guardar si se desea guardar en disco en formato PNG.

```
1 referencia
private void Generar_Click(object sender, EventArgs e)
{
    Barcode Codigo = new Barcode();
    Codigo.IncludeLabel = true;
    if (!"".Equals(textBox1.Text))
    {
        panel1.BackgroundImage = Codigo.Encode(BarcodeLib.TYPE.CODE128, textBox1.Text, Color.Black, Color.White, panel1.Width, panel1.Height);
        Guardar.Enabled = true;
    }
}

1 referencia
private void Guardar_Click(object sender, EventArgs e)
{
    Image imgFinal = (Image)panel1.BackgroundImage.Clone();
    SaveFileDialog cajaDialogo = new SaveFileDialog();
    cajaDialogo.AddExtension = true;
    cajaDialogo.Filter = "Image PNG (*.png)|*.png";
    cajaDialogo.ShowDialog();
    if(!string.IsNullOrEmpty(cajaDialogo.FileName))
    {
        imgFinal.Save(cajaDialogo.FileName, ImageFormat.Png);
    }
    imgFinal.Dispose();
}
```

Se genera un código de barras de CODE 128:

https://es.wikipedia.org/wiki/Code_128

https://hmong.es/wiki/Code128

C# (.NET)

Microsoft® .NET

Code 128

Code 128 es un código de barras de alta densidad, usado ampliamente para la logística y paquetería. Puede codificar caracteres alfanuméricos o solo numéricos. Con este código es posible representar todos los caracteres de la tabla ASCII, incluyendo los caracteres de control.

Para comprender cómo se codifica este código, debemos tener en cuenta que cada ASCII se codifica mediante 11 barras.

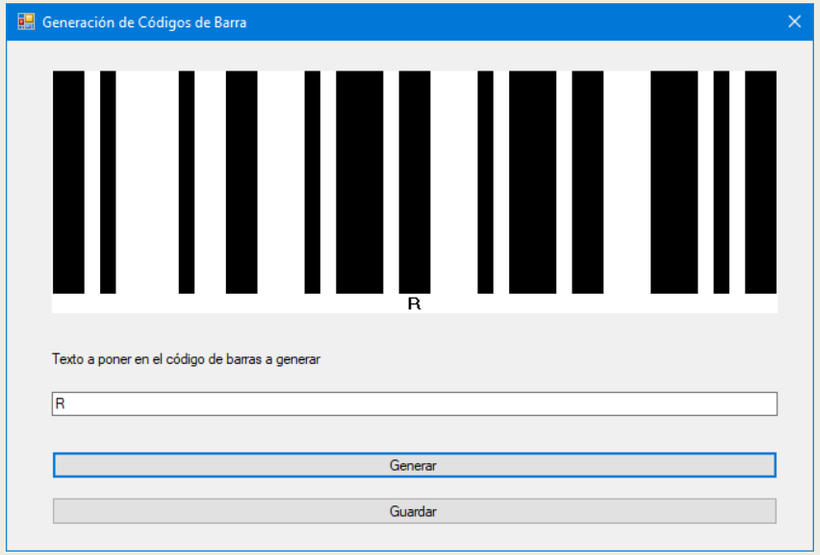
Por ejemplo , el carácter ASCII <espacio> está formado por:

```
* Dos barras negras
* Una barra  blanca
* Dos barras negras
* Dos barras blancas
* Dos barras negras
* Dos barras blancas
* TOTAL= 11 Barras.
```

El código en realidad incluye seis zonas:

```
* A la izquierda, una zona en blanco que debería tener la longitud de dos caracteres.
* El carácter de inicio.
* Un número variable de caracteres ASCII y es lo más útil de este código.
* Un dígito para checkear la integridad de los datos.
* Un carácter de fin o "Stop character"
* A la derecha, una zona en blanco equivalente a dos caracteres.
```

Ejemplo con el carácter “R” ASCII 82



Prueba de escanear con el móvil este código:

Generación QR

CÓDIGO QR

PNG

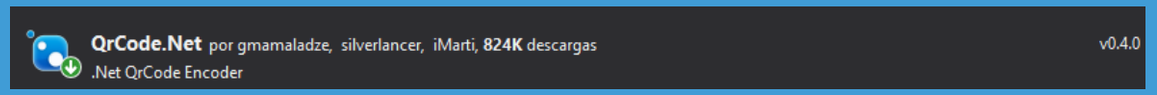
Basado en este enlace: https://www.youtube.com/watch?v=R_EAZ19AkHU



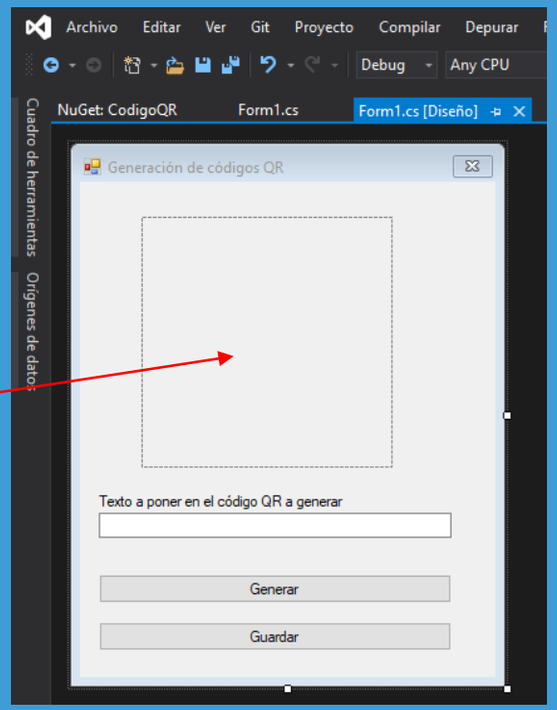
C# (.NET)


Generación código QR

Se utiliza la librería QrCode.Net v0.4.0



PictureBox de 200x200



1° Se debe escribir el texto que se desee.

2° Se pulsa el botón Generar para visualizar el código QR generado.

3° Se pulsa el botón Guardar si se desea guardar en disco en formato PNG.

```
1 referencia
private void Generar_Click(object sender, EventArgs e)
{
    if (!"".Equals(textBox1.Text))
    {
        QrEncoder qrEncoder = new QrEncoder(ErrorCorrectionLevel.H);
        QrCode qrCode = new QrCode();
        qrEncoder.TryEncode(textBox1.Text.Trim(), out qrCode);

        GraphicsRenderer renderer = new GraphicsRenderer(new FixedCodeSize(400, QuietZoneModules.Zero), Brushes.Black, Brushes.White);
        MemoryStream ms = new MemoryStream();
        renderer.WriteToStream(qrCode.Matrix, ImageFormat.Png, ms);
        var imageTemporal = new Bitmap(ms);
        var imagen = new Bitmap(imageTemporal, new Size(new Point(imgQr.Width, imgQr.Height)));
        imgQr.BackgroundImage = imagen;
        Guardar.Enabled = true;
    }
}

1 referencia
private void Guardar_Click(object sender, EventArgs e)
{

    Image imgFinal = (Image)imgQr.BackgroundImage.Clone();
    SaveFileDialog cajaDialogo = new SaveFileDialog();
    cajaDialogo.AddExtension = true;
    cajaDialogo.Filter = "Image PNG (*.png)|*.png";
    cajaDialogo.ShowDialog();
    if (!string.IsNullOrEmpty(cajaDialogo.FileName))
    {
        imgFinal.Save(cajaDialogo.FileName, ImageFormat.Png);
    }
    imgFinal.Dispose();
}
```



C# (.NET)


Code QR (Quick Response)

https://es.wikipedia.org/wiki/C%C3%B3digo_QR

QR es la evolución del código de barras. Es un módulo para almacenar información en una matriz de puntos o en un código de barras **bidimensional**. La matriz se lee en el dispositivo móvil por un lector específico (lector de QR) y de forma inmediata nos lleva a una aplicación en Internet, un mapa de localización, un correo electrónico, una página web o un perfil en una red social.

Presenta tres cuadrados en las esquinas que permiten detectar la posición del código al lector.

El objetivo de los creadores fue que el código permitiera que su contenido se leyera a alta velocidad. Los códigos QR son muy comunes en Japón, donde son el código bidimensional más popular.

Ejemplo con el carácter "R" ASCII 82

Prueba de escanear con el móvil este código:

Driver código barras

BASE CHARGER COM.

USB

READER & DISPLAY

CÓDIGO BARRAS

Para recibir el código de barras por el puerto USB e interpretarlo. Para ser utilizada debe ser importada en un proyecto EXE.



C# (.NET)

Microsoft® .NET


Driver lector código barras

Driver DLL importable por cualquier aplicación para hacer la función de recibir datos por USB (COM).

Al llamar a **Conectar()** se abre el puerto serie y se mantiene a la espera de recibir datos.

```
public bool Conectar(string portName, int baudRate, Parity parity, int dataBits, StopBits stopBits)
{

    try
    {
        _portName = portName;
        _baudRate = baudRate;
        _parity = parity;
        _dataBits = dataBits;
        _stopBits = stopBits;

        _serialPort = new SerialPort(portName, baudRate, parity, dataBits, stopBits);
        _serialPort.Handshake = Handshake.None;
        _serialPort.DataReceived += new SerialDataReceivedEventHandler(sp_DataReceived);
        _serialPort.ReadTimeout = 500;
        _serialPort.WriteTimeout = 500;

        if (!_serialPort.IsOpen)
        {
            _serialPort.Open();
            return true;
        }

        return false;

    }
    catch (Exception)
    {
        return false;
    }

}
```

El evento de datos recibidos se hace a través del delegado **sp_DataReceived()** a través del evento SerialDataReceivedEventHandler(), cuando se llama al método **ReadExisting()** del puerto serie.

```
void sp_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        Thread.Sleep(500);
        string data = _serialPort.ReadExisting();
    }
    catch (Exception)
    { }
}
```

```
public bool Ack(string comando)
{
    try
    {
        _serialPort.WriteLine(comando);
        return true;
    }
    catch (Exception)
    {
        return false;
    }
}
```

Después de leer se puede enviar un **Ack()** programado con un comando que entienda el lector. Puede ser OK, pitido OK, led **verde** o KO, pitido KO, led **rojo**.

PING

www.google.es

```
Símbolo del sistema

C:\Users\r.moraleda>ping www.google.es

Haciendo ping a www.google.es [172.217.17.3] con 32 bytes de datos:
Respuesta desde 172.217.17.3: bytes=32 tiempo=36ms TTL=54
Respuesta desde 172.217.17.3: bytes=32 tiempo=11ms TTL=54
Respuesta desde 172.217.17.3: bytes=32 tiempo=12ms TTL=54
Respuesta desde 172.217.17.3: bytes=32 tiempo=11ms TTL=54

Estadísticas de ping para 172.217.17.3:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 11ms, Máximo = 36ms, Media = 17ms
```

*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
10	2.470686	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=254/65024, ttl=128 (reply in 11)
11	2.482196	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=254/65024, ttl=54 (request in 10)
12	4.485067	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=255/65280, ttl=128 (reply in 13)
13	4.496989	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=255/65280, ttl=54 (request in 12)
16	6.499628	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=256/1, ttl=128 (reply in 17)
17	6.514041	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=256/1, ttl=54 (request in 16)
20	8.516586	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=257/257, ttl=128 (reply in 21)
21	8.529157	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=257/257, ttl=54 (request in 20)
22	10.532151	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=258/513, ttl=128 (reply in 23)
23	10.544698	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=258/513, ttl=54 (request in 22)
33	12.546115	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=259/769, ttl=128 (reply in 34)
34	12.557921	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=259/769, ttl=54 (request in 33)
53	14.560237	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=260/1025, ttl=128 (reply in 54)
54	14.573897	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=260/1025, ttl=54 (request in 53)
63	16.575784	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=261/1281, ttl=128 (reply in 64)
64	16.588277	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=261/1281, ttl=54 (request in 63)
65	18.590663	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=262/1537, ttl=128 (reply in 66)
66	18.603187	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=262/1537, ttl=54 (request in 65)
67	20.605603	192.168.1.151	172.217.17.3	ICMP	74	Echo (ping) request id=0x0001, seq=263/1793, ttl=128 (reply in 68)
68	20.618117	172.217.17.3	192.168.1.151	ICMP	74	Echo (ping) reply id=0x0001, seq=263/1793, ttl=54 (request in 67)

```
> Frame 67: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor_89:cf:79 (c0:b6:f9:89:cf:79), Dst: Sagemcom_23:f7:61 (70:0b:01:23:f7:61)
> Internet Protocol Version 4, Src: 192.168.1.151, Dst: 172.217.17.3
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4c54 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 263 (0x0107)
    Sequence number (LE): 1793 (0x0701)
    [Response frame: 68]
  > Data (32 bytes)
```

Envío de comandos PING

C# (.NET)

Ping

Ejecutable que envía 10 pings a www.google.es y va printando el resultado.

Program.cs

```
ConsoleApplication1                                   ConsoleApplication1.Program
1   using System;
2   using System.Net.NetworkInformation;
3   using System.Threading;
4
5   namespace ConsoleApplication1
6   {
        0 references
7       class Program
8       {
            0 references
9           static void Main(string[] args)
10          {
11              bool resultado;
12
13              for (int i=1; i<=10;i++)
14              {
15                  resultado = ping("www.google.es", 2000);
16                  Console.WriteLine(i.ToString() + "- " + resultado.ToString());
17                  Thread.Sleep(2000);
18              }
19          }
20
            1 reference
21          public static bool ping(string host, int timeout)
22          {
23              Ping pingSender = new Ping();
24
25              PingReply reply = pingSender.Send(host,timeout);
26
27              if (reply.Status == IPStatus.Success)
28              {
29                  return true;
30              }
31              return false;
32          }
33      }
34  }
```

Se utiliza la librería **System.Net.NetworkInformation** creando un objeto **Ping** y llamando a **Send().**

Se pasa el host, en este caso www.google.es y el timeout, en este caso 2 segundos.

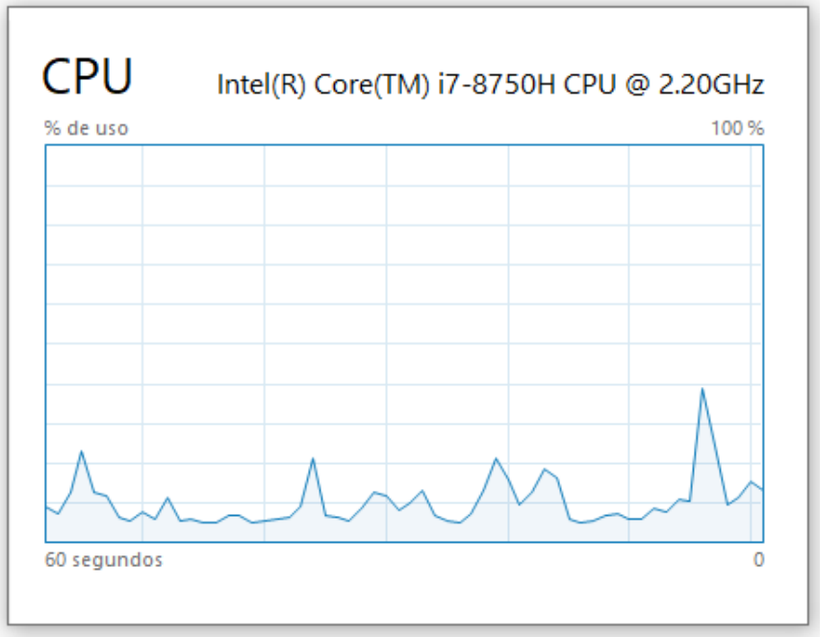
Con Console.WriteLine() se printa por consola el resultado del método **ping()** ejecutado cada 2 segundos.

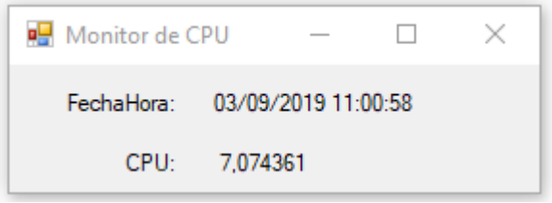
```
C:\WINDOWS\system32\cmd.exe
1- True
2- True
3- True
4- True
5- True
6- True
7- True
8- True
9- True
10- True
Presione una tecla para continuar . . .
```

En la página anterior se muestra una captura con Wireshark de los paquetes ICMP al ejecutar la aplicación. Son 10 requests y 10 replies.

Monitor de CPU

Monitor de CPU del PC





CSV

	A	B
1	03/09/2019 11:00:47	3,734244
2	03/09/2019 11:00:48	3,806889
3	03/09/2019 11:00:49	4,319988
4	03/09/2019 11:00:50	3,662477
5	03/09/2019 11:00:51	6,046159
6	03/09/2019 11:00:52	4,609344
7	03/09/2019 11:00:53	10,97619
8	03/09/2019 11:00:54	20,00967
9	03/09/2019 11:00:55	3,267673
10	03/09/2019 11:00:56	5,195621
11	03/09/2019 11:00:57	8,973086
12	03/09/2019 11:00:58	7,074361
13	03/09/2019 11:00:59	11,46766
14	03/09/2019 11:01:00	7,468812
15	03/09/2019 11:01:01	6,705626
16	03/09/2019 11:01:02	6,129755
17	03/09/2019 11:01:03	4,225739
18	03/09/2019 11:01:04	5,770069

C# (.NET)

Monitor CPU write to csv

Ejecutable que lee el % de CPU y lo historiza en un CSV

MonitorCPU | MonitorCPU.Form1

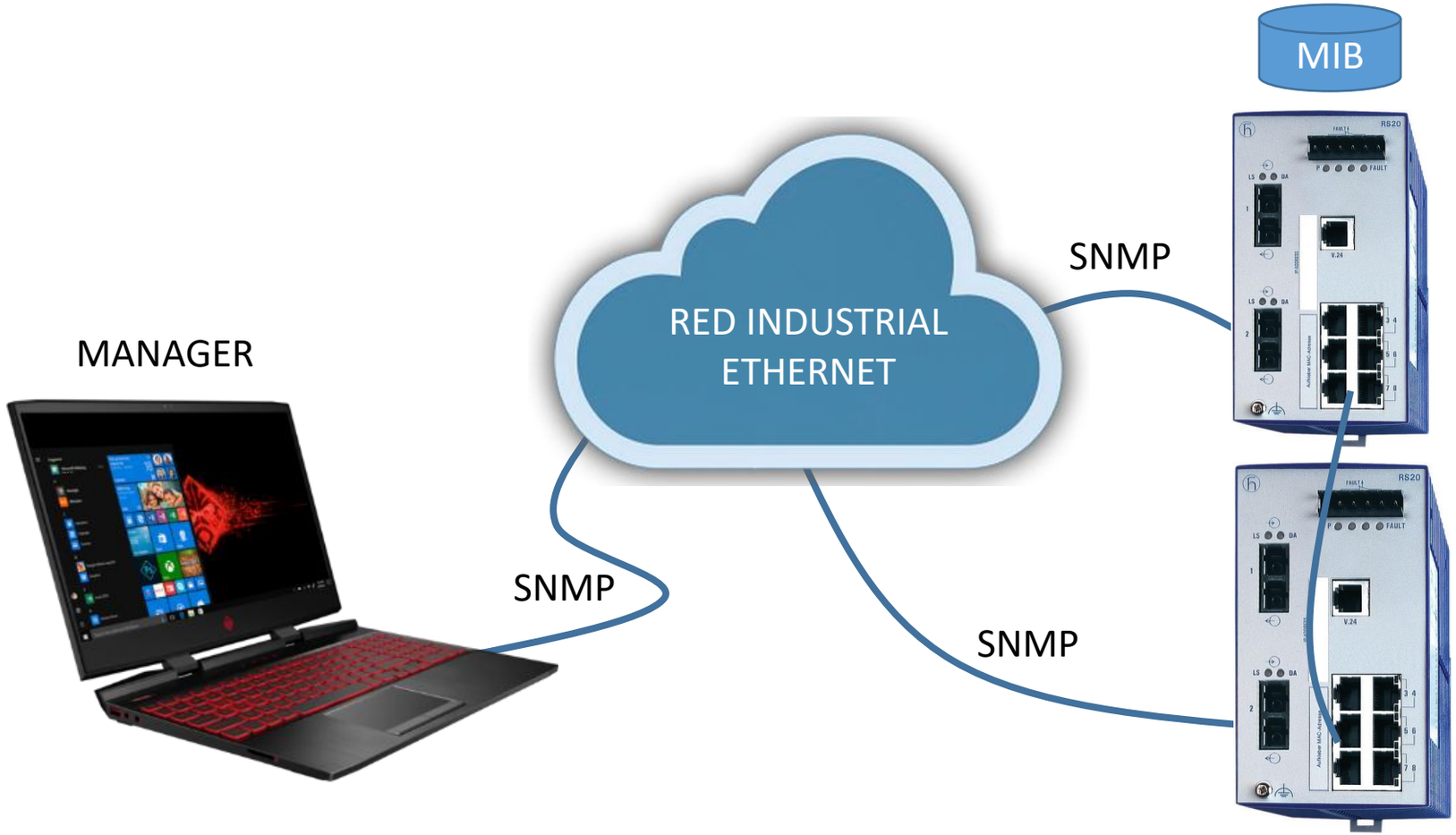
```
using System;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;

namespace MonitorCPU
{
    3 references
    public partial class Form1 : Form
    {

        private PerformanceCounter CPUCounter = new PerformanceCounter("Processor", "% Processor Time", "_Total");

        1 reference
        public Form1()
        {
            InitializeComponent();
            this.timer1.Enabled = true;
            this.timer1.Interval = 1000;
        }

        1 reference
        private void timer1_Tick(object sender, EventArgs e)
        {
            this.label4.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
            this.label1.Text = this.CPUCounter.NextValue().ToString();
            this.WriteFile();
        }

        1 reference
        private void WriteFile()
        {
            using (StreamWriter stream = new FileInfo("cpuLog.csv").AppendText())
            {
                stream.WriteLine(this.label4.Text + ";" + this.label1.Text);
            }
        }
    }
}
```

Se utiliza la librería **System.Diagnostics.PerformanceCounter()** pasando el contador deseado en este caso % Processor Time – Total.

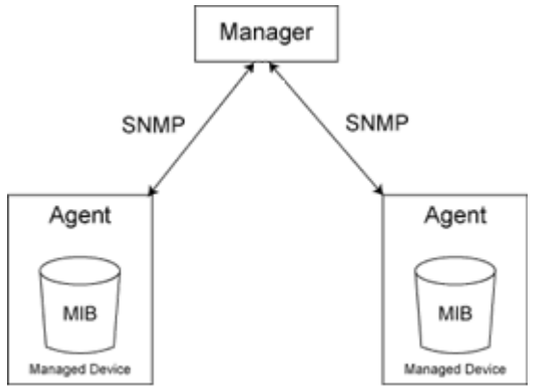
Se crea un thread (hilo) con intervalo 1 segundo donde se recoge el valor con **CPUCounter.NextValue()** y posteriormente se guarda en el fichero csv a través del método **writeFile()**.

Driver SNMP

SNMP
Simple Network Management Protocol



Switches industriales de comunicaciones Ethernet de la marca **HIRSCHMANN** modelo RS20.



Para recibir datos por SNMP de la MIB del dispositivo a conectar.
MIB: Management Information Base

Reference record for OID 1.3.6.1.4.1.248

1 iso › 3 identified-organization, org, iso-identified-organization › 6 dod › 1 internet › 4 private › 1 enterprise, enterprises › 248 hirschmann



C# (.NET)


Driver SNMP

Driver DLL importable por cualquier aplicación para hacer la función de recibir datos por Ethernet a través del protocolo SNMP (UDP).

```
public SNMPv1CommunicationInterface(int version, string hostAddress, string community, int receiveTimeout, int sendTimeout)
{
    try
    {
        this.version = version;
        this.hostAddress = hostAddress;
        this.community = community;
        this.receiveTimeout = receiveTimeout;
        this.sendTimeout = sendTimeout;
        dSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
        dSocket.SendTimeout = this.sendTimeout;
    }
    catch (SocketException)
    {
       throw;
    }
}
```

Es válido para cualquier dispositivo Ethernet que hable SNMP. En la actualidad la mayoría; como impresoras, switches, routers, servidores, etc.

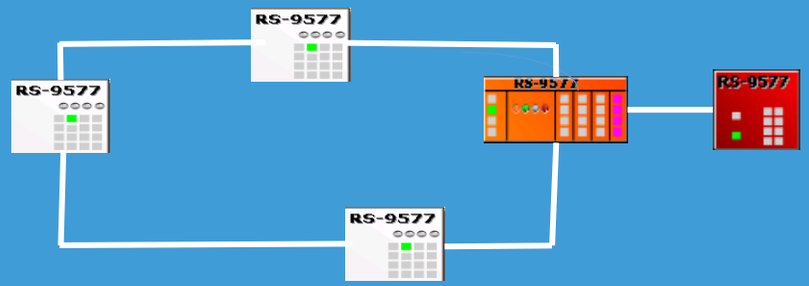
Al llamar a getMIBEntry() pasando una OID se conecta al dispositivo, nos devuelve un objeto con su valor numérico o alfanumérico y se desconecta.

```
public MibEntry getMIBEntry (string oid)
{
    SNMPv1CommunicationInterface snmp = null;
    MibEntry entrada = new MibEntry();

    try
    {
        snmp = new SNMPv1CommunicationInterface(version, host, community, receiveTimeout, sendTimeout);
        snmp.setSocketTimeout(timeout);
        SNMPVarBindList newVars = snmp.getMIBEntry(oid);
        SNMPSequence pair = (SNMPSequence)(newVars.getSNMPObjectAt(0));
        SNMPObjectIdentifier snmpOID = (SNMPObjectIdentifier)pair.getSNMPObjectAt(0);
        SNMPObject snmpValue = pair.getSNMPObjectAt(1);
        snmp.closeConnection();

        if (snmpValue is SNMPInteger)
        {
            BigInteger binteger = (BigInteger)snmpValue.getValue();
            entrada.resultado = MibEntry.MIBENTRY_OK;
            entrada.oid = snmpOID.getDigits();
            entrada.valorAlfanumerico = "" + binteger.valor;
            entrada.valorNumerico = binteger.valor;
            return entrada;
        }
        else if (snmpValue is SNMPOctetString)
        {
            entrada.resultado = MibEntry.MIBENTRY_OK;
            entrada.oid = snmpOID.getDigits();
            entrada.valorAlfanumerico = snmpValue.toString();
            entrada.valorNumerico = 0;
            return entrada;
        }
    }
    catch (SNMPException e)
    {
        entrada.resultado = MibEntry.MIBENTRY_ERROR;
        entrada.codigoExcepcion = e.errorStatus;
        entrada.mensajeExcepcion = e.message;
    }

    try
    {
        snmp.closeConnection();
    }
    catch (Exception)
    {
    }
    return entrada;
}
```

C# (.NET)

Driver SNMP. Utilización

He utilizado este driver para importarlo en un SCADA (System Platform / Intouch de Wonderware) para obtener los datos de la red de switches a monitorizar.



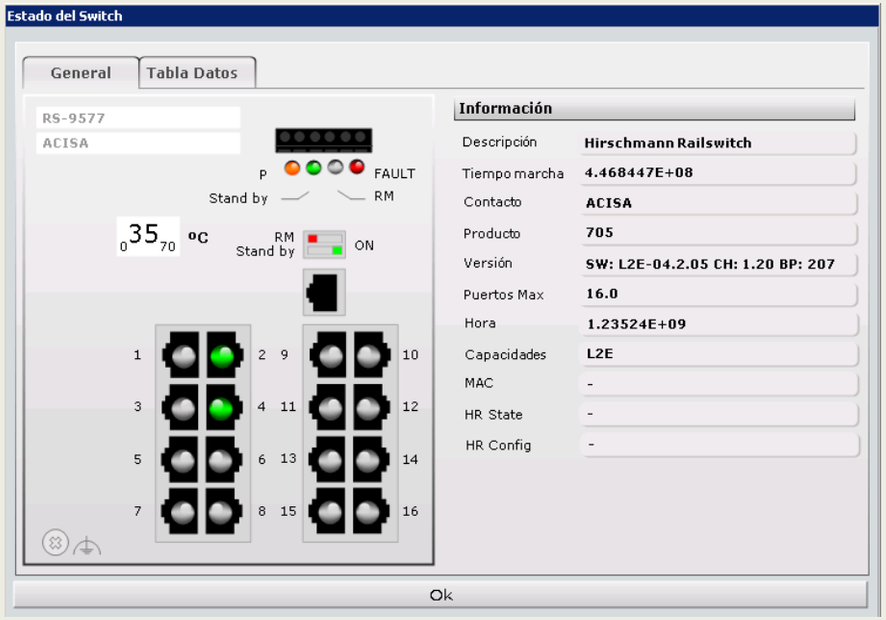
Utilización. Instanciar la DLL y llamar a getMIBEntry() pasando el OID.

```
Me.ifNumber = libreriaSNMP.getMIBEntry("1.3.6.1.2.1.2.1.0").valorNumerico - 1;
Me.sysDescr = libreriaSNMP.getMIBEntry("1.3.6.1.2.1.1.1.0").valorAlfanumerico;
Me.sysUpTime = libreriaSNMP.getMIBEntry("1.3.6.1.2.1.1.3.0").valorNumerico;
Me.sysContact = libreriaSNMP.getMIBEntry("1.3.6.1.2.1.1.4.0").valorAlfanumerico;
Me.sysName =  libreriaSNMP.getMIBEntry("1.3.6.1.2.1.1.5.0").valorAlfanumerico;
Me.sysLocation = libreriaSNMP.getMIBEntry("1.3.6.1.2.1.1.6.0").valorAlfanumerico;
Me.hmSysProduct = libreriaSNMP.getMIBEntry("1.3.6.1.4.1.248.14.1.1.1.0").valorAlfanumerico;
Me.hmSysVersion = libreriaSNMP.getMIBEntry("1.3.6.1.4.1.248.14.1.1.2.0").valorAlfanumerico;
Me.hmSysModulePortCapacity = libreriaSNMP.getMIBEntry("1.3.6.1.4.1.248.14.1.1.8.0").valorAlfanumerico;
Me.hmSystemTime = libreriaSNMP.getMIBEntry("1.3.6.1.4.1.248.14.1.1.30.0").valorNumerico;
Me.hmSysSoftwareCapability = libreriaSNMP.getMIBEntry("1.3.6.1.4.1.248.14.1.1.34.0").valorAlfanumerico;
```

```
    mibEntries = libreriaSNMP.getMIBEntriesNumber("1.3.6.1.2.1.2.2.1.8", Me.ifNumber);
    for i = 1 to Me.ifNumber step 1
       Me.ifOperStatus[i] = mibEntries.entradas[i].valorNumerico;
    next;
```

Al clicar en el elemento abre una ventana detalle con la información. Cada parámetro tiene una OID asociada por el fabricante.

- Temperatura, puertos conectados, información configurada del switch, alarmas, estados, bytes TX, RX, errores, etc.



C# (.NET)
v.1.2 MARZO 2024

https://www.linkedin.com/in/ricardo-moraleda-gareta-9421099

https://www.linkedin.com/company/gdo-electric1996/

RICARDO MORALEDA GARETA

GDO SOFTWARE