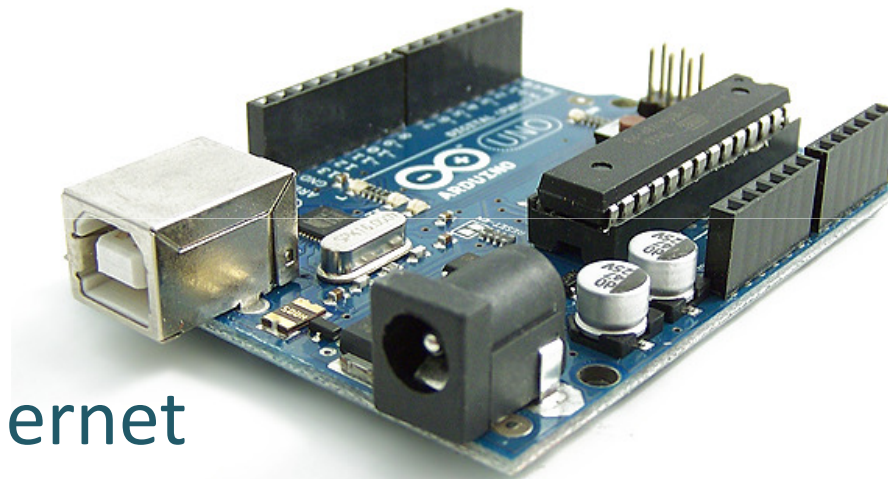




Aplicaciones Prácticas con Sistemas Arduino



Introducción a Ethernet

CEP SEVILLA
IES Los Viveros
Curso 2011/2012
Ref: 11412FP23

Luis Modesto González
José Pujol Pérez
Coordinador:
Leopoldo Acal Rodríguez



UDP



- Modelo Cliente - Servidor
- Para la comunicación UDP, necesitamos:
 - Dirección MAC
 - Dirección IP del origen y del destino
 - Puerto de envío

Entorno ordenador

- Disponemos de:
 - MAC
 - IP origen
- Necesitamos asignar
 - IP destino
 - Puerto



Entorno Arduino

- Disponemos de:
 - Nada
- Necesitamos asignar
 - MAC
 - IP origen (la nuestra)
 - IP destino
 - Puerto



Programación Visual C



- Clase UdpClient
- No es un componente visual
- Documentación en <http://msdn.microsoft.com/es-es/library/9wkb9k12.aspx>
- Métodos usados:
 - **BeginReceive**: Inicia al “escuchador” Udp
 - **EndReceive**: Finaliza la recepción de datos y recupera los datos leídos
 - **Send**: Permite enviar un mensaje Udp.

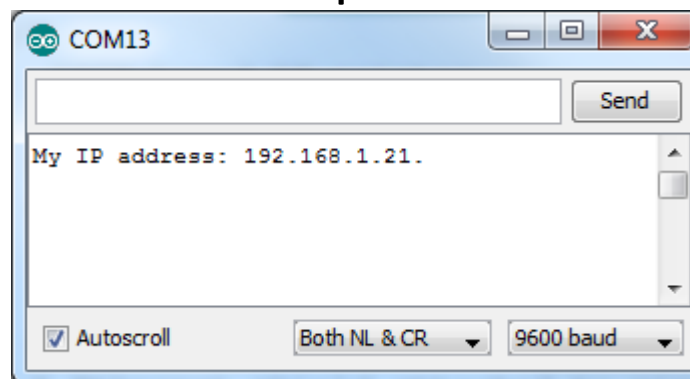




EJEMPLO 1: Encendido del led



- Necesitamos conocer la configuración IP de nuestra red, para configurar arduino.
- Cargamos el ejemplo Ethernet->dhcpPrinter
- Ponemos como dirección MAC la que viene en la pegatina de nuestra Ethernet Shield.
- Conectamos el cable de red y abrimos el monitor serie
- Si todo ha ido bien, obtendremos algo así.
- No podemos tener direcciones repetidas





EJEMPLO 1: Visual C



- Creamos el formulario con los nombres a los textBox en rojo
- Para encender el Led, enviaremos un 50 y para apagarlo un 51
- No podremos usar el led del Pin 13, porque ese pin lo usa el ethernet shield. Colocaremos un led entre los pines 2 y 3.



Form1

dirIp puerto

Dir IP Puerto

Encender

Apagar



EJEMPLO 1: Visual C



- Incluimos la librería <windows.h>
- Incluimos la sentencia (sólo si nuestro programa usará threads –hilos) `CheckForIllegalCrossThreadCalls=0;`
- Añadimos los espacios de nombres para poder usar `UdpClient`, justo debajo de las otras declaraciones de nombres

```
using namespace System::Net;  
using namespace System::Net::Sockets;  
using namespace System::Text;  
using namespace System::Threading;
```

```
1 #pragma once  
2 |  
3 |  
4 namespace udp_ip {  
5 |  
6     using namespace System;  
7     using namespace System::ComponentModel;  
8     using namespace System::Collections;  
9     using namespace System::Windows::Forms;  
10    using namespace System::Data;  
11    using namespace System::Drawing;  
12    using namespace System::Net;  
13    using namespace System::Net::Sockets;  
14    using namespace System::Text;  
15    using namespace System::Threading;  
16 }
```





EJEMPLO 1: Visual C



- Función del botón encender

```
/////////////////////////////////encender led //////////////////////////////////  
array<Byte>^ sendBytes = gcnew array<Byte> (10);  
sendBytes[0]=50;  
UdpClient^ cliente = gcnew UdpClient;  
cliente->Send(sendBytes,1,  
              dirIp->Text,Int32::Parse(this->puerto->Text));  
cliente->Close();
```





EJEMPLO 1: Visual C



- Función del botón apagar

```
/////////////////////////////////apagar led //////////////////////////////////  
array<Byte>^ sendBytes = gcnew array<Byte> (10);  
sendBytes[0]=51;  
UdpClient^ cliente = gcnew UdpClient;  
cliente->Send(sendBytes,1,  
              dirIp->Text,Int32::Parse(this->puerto->Text));  
cliente->Close();
```





EJEMPLO 1: Visual C



- Antes de ejecutar, inicializamos los campos IP, y puerto desde el constructor.

```
Form1(void)
{
    InitializeComponent();
    //
    //TODO: agregar código de constructor aquí
    //
    this->dirIp->Text="192.168.1.21";
    this->puerto->Text="1500";
}
```





EJEMPLO 1: Arduino



- Cargamos el programa de arduino.
- Ajustamos dirección Mac, dirección IP y puerto

```
#include <SPI.h>    // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h> // UDP library from: bjoern@cs.stanford.edu 12/30/2008

#define UDP_TX_PACKET_MAX_SIZE 64
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192,168,1,177 };
unsigned int localPort = 1500;      // local port to listen on

// buffers for receiving and sending data
char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //buffer to hold incoming packet,

EthernetUDP Udp;
void setup() {
  pinMode(5,OUTPUT);
  pinMode(7,OUTPUT);
  digitalWrite(5,LOW); //Ponemos el pin 5 a 0 para conectar el led entre el 5 y el 7
  digitalWrite(7,HIGH); //por defecto lo dejo encendido
  // start the Ethernet and UDP:
  Ethernet.begin(mac,ip); //si quitamos el parámetro ip, tomaremos una ip desde el dhcp
  Udp.begin(localPort);
  Serial.begin(9600);
}
```





EJEMPLO 1: Arduino



- Código del bucle principal

```
void loop() {  
    byte dato;  
    int packetSize = Udp.available(); // note that this includes the UDP header  
    Udp.parsePacket(); //Procesamos el paquete  
    //Si se reciben varios paquetes, se acumulan todos en el buffer, por eso, los vamos a  
    separar  
    while (packetSize>8)  
    {  
        Serial.print("Recibido paquete de tamaño ");  
        Serial.println(packetSize);  
        // read the packet into packetBuffer and get the senders IP addr and port number  
        Udp.read(packetBuffer,10);  
        dato=packetBuffer[0];  
        switch(dato)  
        {  
            case 50: digitalWrite(7,HIGH); break;  
            case 51: digitalWrite(7,LOW); break;  
        }  
        Serial.println("Contenido:");  
        Serial.println(dato,DEC);  
        Serial.print("Dir. IP ");  
        Serial.println(Udp.remoteIP());  
        Serial.print("Puerto ");  
        Serial.println(Udp.remotePort());  
        packetSize-=10;//restamos el tamaño de un paquete leído  
    }  
    Udp.flush();//vaciamos cualquier contenido que quede en el buffer UDP  
    delay(10);  
}
```





EJEMPLO 2: Leer entrada digital Interface



- Similar al anterior, pero ahora leeremos el valor de un entrada digital.
- Usaremos un protocolo pregunta-respuesta, en la que el PC solicita el estado del puerto y arduino contesta
- Usaremos un timer para fijar la frecuencia con que realizamos las lecturas.



Form1

dirIp puerto

Dir IP 192.168.1.21 Puerto 1500

Encender

Apagar

Iniciar servidor

entrada1

■ Entrada Digital



EJEMPLO 2: Visual C Variables globales



- Nos basamos en el ejemplo anterior, para añadir funcionalidades.
- Declaramos una variable global de tipo UdpClient
- Lo hacemos al principio del código

```
/// <summary>  
/// Variable del diseñador requerida.  
private :UdpClient ^listener;  
/// </summary>
```





EJEMPLO 2: Visual C Iniciar servidor



- Botón Iniciar servidor
 - Inicia al listener Udp en el puerto seleccionado.
 - Asigna una función al evento de recibir datos
 - Inicia el timer que controla la petición de datos

```
//////////////////// boton iniciar servidor////////////////////  
listener=gcnew UdpClient (Int32::Parse(this->puerto->Text));  
Listener->BeginReceive(gcnew  
AsyncCallback(this,&Form1::recibirUdp),listener);  
timer1->Enabled=true;
```

```
//////////////////// Tratamiento del evento de recibir datos //////////////////////  
private: System::Void recibirUdp(IAsyncResult^ result) {  
    array<Byte>^ receivedBytes = gcnew array<Byte> (256);  
    UdpClient^ receptor = (UdpClient^ )result->AsyncState;  
    IPEndPoint^ RemoteIpEndPoint = gcnew IPEndPoint(0,Int32::Parse(this->puerto->Text) );  
    receivedBytes=receptor->EndReceive(result,RemoteIpEndPoint);//leemos los datos  
    listener->BeginReceive(gcnew AsyncCallback(this,&Form1::recibirUdp),listener); //reassignamos  
    //Tratamiento de los datos recibidos  
}
```





EJEMPLO 2: Visual C Timer



- Asignamos el evento del timer y lo ajustamos a 500ms
- El timer enviará un paquete con el código 52, para solicitar la lectura del puerto 2

```
////////// Tratamiento del tick del timer//////////  
array<Byte>^ sendBytes = gcnew array<Byte> (10);  
sendBytes[0]=52;  
UdpClient^ cliente = gcnew UdpClient;  
cliente->Send(sendBytes,1,dirIp->Text,Int32::Parse(this->puerto->Text));  
cliente->Close();
```





EJEMPLO 2: Visual C

Tratamiento datos recibidos



- Arduino devolverá 2 bytes, código de función y estado
- En nuestro caso, sólo hay un código de función que es el 52
- Si el estado es 0, el color será negro, y si es 1 será verde lima
- Esto lo colocaremos dentro de la función recibirUdp

```
////////Tratamiento de los datos recibidos
switch (receivedBytes[0]){
    case 52: if(receivedBytes[1]==0)
        this->entrada1->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
        else this->entrada1->BackColor = System::Drawing::Color::Lime;
        break;
    }
}
```





EJEMPLO 2: Visual C



- Al iniciar el servidor, debemos ver el puerto a la escucha, ejecutando `netstat -a`, en una ventana de `msdos`

```
C:\>netstat -a
Conexiones activas

Proto  Dirección local
UDP    0.0.0.0:161
UDP    0.0.0.0:500
UDP    0.0.0.0:1500
UDP    0.0.0.0:1947
```





EJEMPLO 2: Arduino



- El código será similar al ejemplo anterior

```
#include <SPI.h>    // needed for Arduino versions later than 018
#include <Ethernet.h>
#include <EthernetUdp.h> // UDP library from: bjoern@cs.stanford.edu 12/30/2008
#define UDP_TX_PACKET_MAX_SIZE 64 // Enter a MAC address and IP address for your control
// The IP address will be dependent on your local network:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192,168,1,177 };
unsigned int localPort = 1500; // local port to listen on
char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //buffer to hold incoming packet,
byte ReplyBuffer[]="datos recibidos en arduino";//buffer de respuesta
EthernetUDP Udp;

void setup() {
  pinMode(2, INPUT);
  pinMode(5, OUTPUT);
  pinMode(7, OUTPUT);
  digitalWrite(5, LOW); //Ponemos el pin 5 a 0 para conectar el led entre el 5 y el 7
  digitalWrite(7, HIGH); //por defecto lo dejo encendido
  digitalWrite(2, HIGH); //activamos el pull-up del pin1
  // start the Ethernet and UDP:
  Ethernet.begin(mac, ip); //si quitamos ip, tomaremos una ip desde el dhcp
  Udp.begin(localPort);
  Serial.begin(9600);
}
```





EJEMPLO 2: Arduino



- El loop también será parecido

```
void loop() {  
  byte dato;  
  int packetSize = Udp.available(); // note that this includes the UDP header  
  Udp.parsePacket(); //Procesamos el paquete  
  //Si se reciben varios paquetes, se acumulan en el buffer, por eso, los vamos a separar  
  while (packetSize>8) {  
    Serial.print("Recibido paquete de tamaño ");  
    Serial.println(packetSize); // read the packet into packetBuffer  
    Udp.read(packetBuffer,10);  
    dato=packetBuffer[0];  
    switch(dato) {  
      case 50: digitalWrite(7,HIGH); break;  
      case 51: digitalWrite(7,LOW); break;  
      case 52: ReplyBuffer[0]=52;  
               ReplyBuffer[1]=digitalRead(2); //leo el pin 2  
               Udp.beginPacket(Udp.remoteIP(), localPort);  
               Udp.write( ReplyBuffer,2 );  
               Udp.endPacket(); break; }  
    Serial.println("Contenido:"); Serial.println(dato,DEC);  
    Serial.print("Dir. IP "); Serial.println(Udp.remoteIP());  
    Serial.print("Puerto "); Serial.println(Udp.remotePort());  
    packetSize-=10; //restamos el tamaño de un paquete leído  
  }  
  Udp.flush(); //vaciamos cualquier contenido que quede en el buffer UDP  
  delay(10); }
```





EJEMPLO 3: Descubrir arduinos



- Vamos a añadir al ejemplo anterior, un botón que permita descubrir arduinos en nuestra red
- Para ello, implementaremos el comando 49, que lo dirigiremos a la dirección IP 255.255.255.255, que corresponde al broadcast.
- En arduino, implementaremos una respuesta a este comando que identifica a nuestro arduino con su IP y nuestro nombre



Form1

Dir IP 192.168.1.177 Puerto 1500

Encender Descubrir

Apagar

Iniciar servidor

☐ Entrada Digital



EJEMPLO 3: Visual C Botón descubrir



- Envía un mensaje al broadcast con el código 49

```
/////////////////////////////////Descubrir arduinos //////////////////////////////////  
array<Byte>^ sendBytes = gcnew array<Byte> (10);  
sendBytes[0]=49;  
UdpClient^ cliente = gcnew UdpClient;  
cliente->Send(sendBytes,1,"255.255.255.255",Int32::Parse(this->puerto->Text)  
cliente->Close();
```





EJEMPLO 3: Visual C Tratamiento datos recibidos



- Arduino devolverá un mensaje con código 48.
- Mostramos en el textbox la cadena que contenga.

case 48:

```
textBox1->Text+=cadena[1]+"-"+cadena[2]+ "\r\n";  
break;
```

