FACOLTÀ DI INGEGNERIA

UNIVERSITÀ POLITECNICA DELLE MARCHE

Trabajo Fin de Grado

Study and Design of a UDP/IP Protocol Infrastructure for Optical Underwater Communications

Para acceder al Titulo de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Simon Antony Still Puebla

Julio 2018

RESUMEN

Este trabajo pretende comprobar si es posible transmitir datos a través del agua entre un submarino y una base en la superficie. Dado que las ondas electromagnéticas se atenúan considerablemente a través del agua, se debe buscar una solución a este problema. Se pretende estudiar si se pueden transmitir los datos utilizando ondas luminosas gracias a la ayuda de un diodo LED emisor y un fotodiodo receptor. Para ello, en primer lugar se creará un programa basado en el protocolo UDP/IP que permita transmitir los datos; en segundo lugar se adaptarán unas placas LX200V20 para poder conectar dichos componentes; y en último lugar se estudiará cómo se transmiten los datos a través del canal para posteriormente elegir los componentes a utilizar.

ABSTRACT

This study aims to verify if transmitting data under water between a submarine and a base on the surface is possible. As electromagnetic waves lose power quickly travelling through water, a solution must be found. The aim is to study the possibility of transmitting data using luminous waves thanks to an LED diode emitter and a photodiode receiver. To achieve this, in first place a program based on the UDP/IP protocol will be created that allows data to be transmitted; then two LX200V20 boards will be adapted to connect these components; and the signals that travel through the connection will be studied to choose the correct components.

ÍNDICE

INTRODUCCIÓN.

En términos generales, la transmisión inalámbrica de datos se ejecuta utilizando ondas electromagnéticas pertenecientes a la zona del espectro que incluye la radiofrecuencia. En cambio, el ecosistema submarino representa una excepción debido a la fuerte atenuación que sufren las ondas electromagnéticas bajo el agua. La exploración submarina está creciendo a regiones mayores y más profundas gracias al amplio desarrollo en vehículos submarinos autónomos (AUVs). Pero, dichos vehículos tienen la necesidad de transmitir y recibir información con una estación de control, ya sea en tierra o en un barco. El uso de un sistema de comunicación alámbrica es posible pero la comunicación inalámbrica resulta considerablemente más práctica.

Como el uso de ondas electromagnéticas en radio frecuencia no se puede desarrollar, la primera solución utilizada se basa en una comunicación mediante la modulación de ondas acústicas. Por desgracia, esta tecnología presenta una limitación considerable. Las ondas acústicas presentan una fuerte absorción a altas frecuencias por parte del agua durante la transmisión. Esta característica produce una limitación en banda a unos pocos kHz y, por tanto, en el bit-rate de la transmisión. Los módems de altas prestaciones pueden alcanzar tasas hasta los 35 kbit/s, inferiores a los deseados idóneamente.

Un área de desarrollo donde los AUVs tienen un papel fundamental se encuentra en el registro de imagen y video de alta resolución del fondo marino. Para este tipo de datos se requiere una transmisión con una alta tasa de bit y baja latencia. Por tanto, las Comunicaciones Inalámbricas Ópticas Submarinas (Underwater Optical Wireless Communications, UOWC), representan una alternativa válida gracias a su combinación de alta velocidad y transmisión inalámbrica.

Esto ha sido posible gracias al alto desarrollo en paralelo de diodos emisores de luz (LEDs) para propósitos lumínicos. Dichos componentes han mejorado en precio, intensidad lumínica y banda de modulación, permitiendo su uso para transmisión a alta velocidad en ambientes submarinos.

En el ecosistema submarino, existe una ventana en la región visible que es aprovechada por las UOWC, permitiendo a esta tecnología ofrecer tasas de bit mucho mayores a las ofrecidas por módems acústicos pero a menor distancia.

Según la velocidad de transmisión y la atenuación del agua, esta tecnología puede alcanzar tasas de Mbit/s hasta decenas de metros de profundidad.

Por ejemplo, el Woods Hole Oceanographic Institute (WHOI), es pionero en comunicaciones ópticas submarinas basadas en LEDs. Su módem fue capaz de transmitir datos a tasas comprendidas entre 5 y 20 Mbit/s a profundidades superiores a los 100 metros. Dichas características son un gran avance en el ámbito de las comunicaciones submarinas. Especialmente óptimo para trabajos de vigilancia en puertos militares, donde se trabaja en condiciones de baja profundidad y alta turbidez sin que el submarino vuelva a la superficie durante largos periodos de tiempo para no interferir en las operaciones habituales del puerto.

En este documento, se pretende demostrar cómo se ha diseñado la infraestructura para una comunicación submarina que se implementará en un submarino. Dicha infraestructura se basará en la teoría de la comunicación mediante LEDs como se ha explicado previamente, y en el uso de una PowerLine Module LX200V20.

Este módulo es una placa capaz de modificar una señal que viaja por un cable de tipo Ethernet para que se pueda transmitir por un cable convencional de cobre y viceversa. Esto permite que se pueda pasar la señal en transmisión a través de un LED para ser recibido por un fotodiodo conectado a otra placa que vuelve a convertir la señal para ser transmitida por el cable Ethernet.

CAPÍTULO 1. PROTOCOLO UDP/IP.

1.1 INTRODUCCIÓN AL PROTOCOLO UDP/IP

Un protocolo establece una serie de reglas que utilizan dos computadoras para comunicar entre sí. En concreto, el protocolo de Internet o IP es uno de los protocolos más importantes al permitir el desarrollo y transporte de datagramas, paquetes de datos en los cuales se divide el conjunto de datos que se desean transmitir. Esta partición de los datos permite procesar los datagramas de manera independiente al definir su representación, ruta y envío.

Existen varias tipologías de protocolos simples como pueden ser el TCP (Protocolo de Control de Transmisión), UDP (User Datagram Protocol) o ARP (Address Resolution Protocol). Estos se diferencian principalmente en la metodología de reconocer los datagramas por parte del destino y hacer saber al emisor de su llegada. En este documento se realizará un estudio del protocolo UDP para posteriormente desarrollar un código para la transmisión de datos a través de un cable ethernet.

El protocolo UDP se limita a recoger el mensaje y enviar el paquete por la red sin necesidad de establecer una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Para garantizar la llegada, el protocolo exige a la máquina de destino del paquete que envíe un mensaje. Si dicho mensaje no llega pasado un tiempo establecido, la máquina de destino envía el mensaje de nuevo. Esto puede originar la duplicación y/o desordenación de los datagramas a su destino.

Por ello se clasifica como un protocolo de tipo best-effort (máximo esfuerzo), porque hace lo que puede para transmitir los datagramas hacia la aplicación pero no puede garantizar que la aplicación los reciba.

Por tanto, es un protocolo sencillo no orientado a la conexión al no establecer una conexión segura y fiable. Muchas aplicaciones cliente/servidor que funcionan en base a una solicitud y una respuesta recurren al protocolo UDP cuando no es necesario establecer y luego liberar una conexión. La mayoría de las aplicaciones claves de Internet utilizan el protocolo UDP como el Protocolo de Administración de Red, el Protocolo de Información de Enrutamiento o el Sistema de Nombres de Dominio. En este último por ejemplo, es preferible utilizar UDP a TCP debido a que las consultas deben ser rápidas y solo necesitan una solicitud.

En cambio, el protocolo TCP es orientado a la conexión al permitir que las dos máquinas que están comunicadas controlen el estado de la transmisión. Este control se realiza mediante el uso de los llamados ACKs (acknowledgement), un mensaje que el destino de la comunicación envía al origen de esta para confirmar la recepción de un mensaje. Estos mensajes son utilizados con el fin de establecer una conexión estable y fija para poder realizar la transmisión. Una comparación esquemática entre los dos protocolos se muestra en la figura 1.

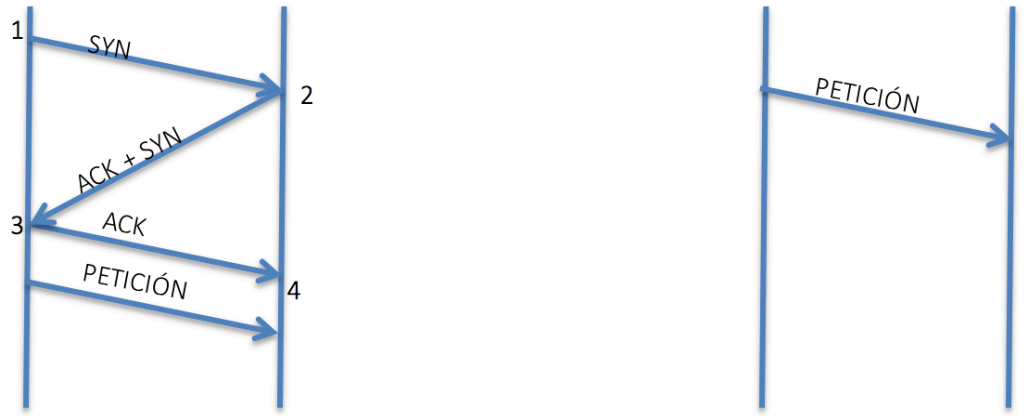

Fig. 1. Comparación entre el protocolo TCP (esquema izquierdo) y el protocolo UDP (esquema derecho).

Aunque el protocolo UDP no establece una conexión antes de comenzar a transmitir datos, es necesario identificar el punto de origen y de destino de la transmisión. Aparte de dichos puntos de origen y destino, es necesario establecer el camino a través del cual se llevará a cabo la transmisión. Esto viene definido por los puertos, un campo que tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535 distintos puertos.

En cuanto al reconocimiento de los puertos de origen y destino, se utilizan las direcciones IP. Una dirección IP es un número que identifica a una interfaz en red de un dispositivo que utilice el protocolo IP. Es decir, una dirección que identifica un elemento de conexión de un dispositivo.

1.2 CÓDIGO DEL SERVIDOR UDP/IP PARA LA TRANSMISIÓN DE DATOS.

En esta parte del documento, se pretende mostrar cómo se ha desarrollado un código fuente en el lenguaje de programación de tipo C para permitir la transmisión de datos entre dos computadoras a través de un cable Ethernet siguiendo las reglas del protocolo UDP/IP.

El protocolo UDP/IP es unidireccional, es decir, solo podemos transmitir información en un sentido. Por tanto, para poder realizar una conversación entre dos computadoras como requiere este trabajo, se deben desarrollar cuatro programas diferentes. Dos corresponderán a servidores y dos a clientes y al final se probará con un cliente y un servidor en cada computadora. Se comprueba el correcto funcionamiento de los programas de esta manera porque solo podemos escribir con el programa del cliente y recibir en el del servidor. De ahí la necesidad de tener un servidor en cada computadora que reciba los mensajes de los clientes de la otra computadora respectivamente.

Para desarrollar este código en Windows, se recurrirá al uso de programación mediante sockets. Estas son unas librerías que proporciona el fabricante para su uso, como se muestra en la figura 2 que representa el principio del programa realizado para el servidor. Se habla de servidor porque se crearán cuatro códigos distintos, un servidor y un cliente para cada computadora. En esta parte en concreto, se explicará el desarrollo del código para el servidor en la comunicación.

```
3     // SERVER 1
4
5     #include "stdafx.h"
6
7     #include <iostream>
8     #include <WS2tcpip.h>
9
10
11    #pragma comment (lib, "ws2_32.lib")
```

Fig. 2. Inicio del programa del servidor de la computadora 1; utilización de las librerías específicas para sockets de Windows.

Un socket es un mecanismo que permite la comunicación entre procesos, en este caso en dos computadoras distintas que pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada. Es decir, es una conexión ficticia que permite transmitir información entre dos programas. Por tanto, permite implementar una arquitectura cliente/servidor, el cual se desea obtener en este trabajo. La comunicación debe ser iniciada por uno de los programas que se denomina cliente, mientras que el segundo se denomina servidor porque espera a que el cliente inicie la comunicación.

Existen muchas variantes de sockets, pero solo interesa estudiar dos tipos de sockets de Internet: sockets de datagramas y sockets de flujo. La diferencia entre los dos se encuentra en el protocolo de transporte utilizado para la transmisión de datos. Para este trabajo, se utilizará un socket de tipo datagrama al ser el correspondiente al protocolo UDP.

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. Por tanto, a la hora de desarrollar el código, se debe tener en cuenta las direcciones IP de las dos computadoras, que se basará en un protocolo UDP y el puerto que se desea utilizar a la hora de definir el socket.

Los distintos pasos que se deben seguir para la creación de un socket y que, por tanto, se utilizará para el desarrollo del código se muestra a continuación:

- Inicialización: creación de un socket para la comunicación.
- Enlace: asignar una dirección IP al socket.

- Escucha: anunciar la voluntad de aceptar conexiones.
- Conexión: activamente intentar establecer una conexión.
- Envío: enviar datos a través del canal de transmisión.
- Recepción: recibir datos del canal de transmisión.
- Cierre: liberar la conexión.

Por tanto, una vez se han incluido las librerías correspondientes al código desarrollado en este trabajo como se muestra en la figura 2, el siguiente paso consiste en la inicialización del socket. Esta parte consiste en utilizar la función *WSAStartup*, que genera el canal para la comunicación.

```
15  void main()
16  {
17
18      //Startup Winsock
19      WSADATA data;
20      WORD version = MAKEWORD(2, 2);
21      int wsOk = WSAStartup(version, &data);
22      if (wsOk != 0)
23      {
24          cout << "Can't start Winsock!" << wsOk;
25          return;
26      }
```

Fig. 3. Inicialización de un socket para la comunicación.

Una vez se ha creado la comunicación, el siguiente paso consiste en enlazar dicho canal a los dispositivos de origen y destino. Para ello, como se ha descrito en la introducción al protocolo UDP/IP, se utilizarán las direcciones IP correspondientes a los puertos de entrada del cable Ethernet en cada computadora. Dicha dirección se puede establecer manualmente en la computadora en el panel de redes Ethernet.

En la figura 4, en primer lugar se define el tipo de socket que se desea generar. Al ser una comunicación que seguirá las normas del protocolo UDP, bastará con establecer un socket del tipo *SOCK_DGRAM*. En el caso de que se estuviera utilizando el protocolo TCP, sería de la tipología *SOCK_STREAM*. *SOCK_STREAM* es un protocolo basado en la conexión, la conexión se establece y los dos integrantes tienen una conversación hasta que uno de los integrantes termina con la conexión. Mientras que *SOCK_DGRAM* es un protocolo basado en datagramas, se envía un datagrama y se recibe una respuesta que finaliza la conexión.

En segundo lugar, se crea una variable que representa la dirección del servidor (*serverHint)* de tipo *sockaddr_in*, para posteriormente establecer la dirección IP del servidor. En este caso se ha establecido como *ADDR_ANY* debido a que se puede permitir que el cliente se conecte a cualquier dirección IP.

Por último, se debe asignar un puerto al canal de comunicación. En este caso se ha elegido el canal 2001 para la conexión entre el servidor 1 y el cliente 2.

```
29      // Bind socket to ip address and port
30      SOCKET in = socket(AF_INET, SOCK_DGRAM, 0);
31      sockaddr_in serverHint;
32      memset((char*)&serverHint, 0, sizeof(serverHint));
33      serverHint.sin_addr.S_un.S_addr = ADDR_ANY; //Esto da una dirección cualquiera, nosc
34
35      //serverHint.sin_addr.S_un.S_addr = inet_pton(AF_INET,"192.168.254.3",&serverHint);
36
37      serverHint.sin_family = AF_INET;
38      //serverHint.sin_port = htons(54000); //Convierte de little a bid endian, aquí meter
39      serverHint.sin_port = htons(2001);
40
41
42      //Binding part
43      if (bind(in, (sockaddr*)&serverHint, sizeof(serverHint)) == SOCKET_ERROR)
44      {
45          cout << "Can't bind socket! " << WSAGetLastError() << endl;
46          return;
47      }
```

Fig. 4. Enlace de la comunicación asignando dirección IP al destino.

Antes de desarrollar la parte correspondiente a la captación de datos, se crea una variable que contendrá la dirección IP del cliente de la misma manera que se hizo para el servidor en la figura 4. Además, como se muestra en la figura 5, se crea un buffer de almacenamiento donde guardar el mensaje recibido desde el cliente.

```
49      //Create another hint structure for the client that connects
50      sockaddr_in client;
51      int clientLength = sizeof(client);
52      ZeroMemory(&client, clientLength);
53
54      //Create space for the buffer where the message gets recieved into
55      char buf[1024];
```

Fig. 5. Creación de dirección para el cliente y buffer para almacenar los datos.

Una vez que ya se han creado todas las variables necesarias, se realiza la parte del código capaz de captar y almacenar el mensaje recibido por parte del cliente. Toda esta parte del código se encuentra dentro de un ciclo que, de manera simbólica, está preguntando constantemente si ha recibido algún mensaje.

Para detectar que se le ha enviado un mensaje, utiliza la función *recvfrom*, que recibe el mensaje desde un socket basado en mensajes. La función devolverá el mensaje almacenado en el buffer, además de su longitud y la dirección IP origen del mensaje. El mensaje completo se leerá en una sola operación pero, si el mensaje es demasiado largo para el buffer creado los bytes excedentes se perderán.

En la figura 6, se muestra cómo se ha utilizado esta función en este trabajo para que el servidor reciba el mensaje enviado por el cliente. El mensaje recibido queda almacenado en la variable *buf* y la dirección IP del origen del mensaje, en este caso del cliente, en la variable *client*. Para desarrollar el código de este trabajo no es necesario el campo de la longitud del mensaje.

```
57      // Enter a loop
58      while (true)
59      {
60          ZeroMemory(buf, 1024);
61
62          //wait for message
63          int bytesIn = recvfrom(in, buf, 1024, 0, (sockaddr*)&client, &clientLength);
64          if (bytesIn == SOCKET_ERROR)
65          {
66              cout << "Error recieving from client " << WSAGetLastError() << endl;
67              continue;
68          }
69
70          //Display message and client info
71          char clientIp[256];     //Create another buffer
72          ZeroMemory(clientIp, 256);
```

Fig. 6. Escucha a la espera de recibir datos y preparación del mensaje de salida.

Como ya se ha almacenado tanto el mensaje recibido como la dirección IP del origen de la comunicación, el siguiente paso consiste en simplemente mostrar dicho mensaje por pantalla. En este caso, además, se requiere mostrar la dirección IP del origen para comprobar la correcta conexión del canal de comunicación.

Para ello, como se muestra en la figura 6, se ha creado una variable nueva donde se almacenará dicho valor. Pero, dicha dirección IP no queda almacenada en el formato estándar sino como un puntero a memoria. Para realizar la conversión, se utiliza la función *inet_ntop*, como se muestra en la figura 7, que convierte dicho puntero en una variable de tipo *string* en el formato mayormente conocido.

Por último, se escribe la línea de código con la función de mostrar el mensaje por pantalla mostrando el mensaje recibido y la dirección IP, como se muestra al final de la figura 7.

```
74          inet_ntop(AF_INET, &client.sin_addr, clientIp, 256);
75          //In this last instruction, we get information from the recvfrom instruction, wh
76          //client information is put into (soxkaddr*)&client. Now we have the IP of the c
77          //&client.sin_addr
78
79          cout << "Message received from " << clientIp << " : " << buf << endl;
80
81      }
```

Fig. 7. Representación del mensaje recibido por pantalla.

Una vez que se haya finalizado con la transmisión de datos, lo ideal es cerrar el socket utilizando las funciones mostradas en la figura 8.

```
84      //Close socket
85      closesocket(in);
86
87      //Shutdown winsock
88      WSACleanup();
89
90
91  }
92
```

Fig. 8. Cierre del socket para liberar la conexión.

Como se ha mencionado anteriormente, se deben desarrollar dos códigos que funcionan como servidor y dos como cliente, uno para cada computadora. El código para el segundo servidor es igual que el descrito hasta el momento con una diferencia significativa. No se debe modificar la referencia a la dirección IP porque, como se observa en la figura 4, se permite que conecte con cualquier dirección IP.

En cambio, la parte que debe ser modificado es la correspondiente a la declaración del puerto. Para el servidor de la primera computadora se utilizó el puerto 2001. Esto corresponde al canal físico por el cual tiene lugar la comunicación por lo que no se puede utilizar el mismo para el segundo servidor al utilizar un protocolo unidireccional como es el UDP. Por tanto, como se observa en la figura 9, a la hora de crear el enlace de la comunicación, la única línea de código que se debe modificar es la numerada como 34. Se utilizará el puerto número 2002 para realizar la comunicación desde la computadora 1 a la computadora 2, ya que el segundo servidor es el que recibe el mensaje de la computadora 1.

```
28      // Bind socket to ip address and port
29      SOCKET in = socket(AF_INET, SOCK_DGRAM, 0);
30      sockaddr_in serverHint;
31      serverHint.sin_addr.S_un.S_addr = ADDR_ANY; //Esto da una dirección cu
32      //serverHint.sin_addr.S_un.S_addr = inet_pton(AF_INET, "192.168.254.3'
33      serverHint.sin_family = AF_INET;
34      serverHint.sin_port = htons(2002); //Convierte de little a bid endian,
35      //serverHint.sin_port = htons(2002);
```

Fig. 9. Enlace de la comunicación para el servidor de la computadora 2.

Antes de pasar a la siguiente parte del documento, cabe destacar que no se han mencionado determinadas partes del código que se pueden observar en las figuras pertenecientes a este apartado. Dichas partes del programa corresponden a pruebas en puntos clave del código que son capaces de detectar errores y, en caso de que el error se cumpla, escribir por pantalla para avisar al usuario. Algunas funciones concretas devuelven un valor dependiendo del error ocurrido, valor que se mostrará por pantalla para el usuario.

Por ejemplo, en el caso de la parte del código que queda representado en la figura 3, se muestra que en el caso de que la variable *wsOK* sea distinta de 0, se escribe un mensaje de aviso por pantalla. Esta variable es un valor que devuelve la función *WSAStartup* para avisar de que se ha inicializado con éxito el socket. Por tanto, para que el usuario sepa que no se ha inicializado correctamente el socket, se escribe un mensaje por pantalla avisando de que ha ocurrido un error y el tipo de error. Dicho error queda establecido por el valor de la variable *wsOK* que también se escribe por pantalla.

1.3 CÓDIGO DEL CLIENTE UDP/IP PARA LA TRANSMISIÓN DE DATOS.

En esta parte del documento se explicará cómo se ha desarrollado el código para el cliente de nuestra comunicación entre computadoras a través de un cable de tipo Ethernet.

Al igual que para el desarrollo del código para el servidor, se utilizará la teoría de los sockets para crear el canal de transmisión. Por tanto, seguirá los pasos descritos en la parte del código del servidor para la creación de un socket para la transmisión de datos entre dos computadoras. El código es bastante similar al descrito previamente como se muestra a continuación.

Para cualquier código basado en programación con sockets, se debe inicializar el socket. Esto se muestra en la figura 10, realizado de una manera muy similar al código del apartado precedente.

```
16  void main(int argc, char* argv[]) //We can pass in a command line option!!
17  {
18      //Startup Winsock
19      WSADATA data;
20      WORD version = MAKEWORD(2, 2);
21      int wsOk = WSAStartup(version, &data);
22      if (wsOk != 0)
23      {
24          cout << "Can't start Winsock!" << wsOk;
25          return;
26      }
```

Fig. 10. Inicialización de un socket para la comunicación.

El siguiente paso consiste en la creación de una estructura, que se ha denominado *server*, que sirve para establecer los datos necesarios para el direccionamiento. En base a esta estructura se deben definir el puerto y la dirección IP a través de los cuales se define el canal de comunicación. La creación de esta estructura queda representada en la figura 11, además del puerto asignado para el canal de comunicación. En este caso, dicho puerto será el 2002 ya que es el cliente de la segunda computadora que se conectará con el servidor de la primera.

```
28      //Create a hint structure for the server
29      sockaddr_in server;
30      server.sin_family = AF_INET;
31      server.sin_port = htons(2002);  //Here we will later write the corresponding port
32      //server.sin_port = htons(2001);
```

Fig. 11. Creación de una estructura para el direccionamiento.

Una vez que se ha definido el puerto por el cual se establece la conexión, falta establecer la dirección IP. Esta es la primera parte del código que difiere considerablemente del código para el servidor. Para el programa del cliente no se puede asignar cualquier dirección IP como para el caso del servidor. Se debe asignar la dirección IP correspondiente a la primera computadora, ya que solo se pretende enviar datos a esta computadora y, por tanto, a esta dirección IP. En la figura 12 se muestra cómo se ha realizado dicha asignación que corresponde a la dirección 192.168.254.1, dirección IP de la primera computadora.

```
34      //inet_pton(AF_INET, "127.0.0.1", &server.sin_addr);
35      inet_pton(AF_INET, "192.168.254.1", &server.sin_addr);
```

Fig. 12. Asignación a la dirección IP.

El último paso antes de comenzar a transmitir el mensaje pasa por crear la variable que realmente representa el socket. Dicha variable se ha denominado *out*, como se muestra en la figura 13, debido a que a la salida de este programa se enviará un mensaje al servidor correspondiente.

```
38      //Socket creation
39      SOCKET out = socket(AF_INET, SOCK_DGRAM, 0);
```

Fig. 13. Creación del socket.

Llegados a este punto, se puede desarrollar la parte del código correspondiente al envío del mensaje a través del canal de comunicación. En primer lugar, se almacena el mensaje tecleado por el usuario en un buffer de memoria para posteriormente enviar dicho mensaje recurriendo a la función *sendto*. Dicha función permite enviar un mensaje almacenado en un buffer a través de un socket. Para ello se debe indicar el puerto y las dirección IP para la comunicación directamente a través de la estructura *server* creada previamente, como se muestra en la figura 14.

```
41      //Write out to that socket
42      string s(argv[1]);  //argv is our buffer this time
43      int sendOk = sendto(out, s.c_str(), s.size() + 1, 0, (sockaddr*)&server, sizeof(server));
44
45      if (sendOk == SOCKET_ERROR)
46      {
47          cout << "That didn't work! " << WSAGetLastError() << endl;
48      }
```

Fig. 14. Envío del mensaje a través del socket.

Para concluir el código, al igual que en la figura 8, conviene cerrar el socket para dejar libre el canal de comunicación para otras futuras transmisiones.

CAPÍTULO 2. APLICACIÓN DEL PROGRAMA CREADO.

2.1 TRANSMISIÓN POR CABLES ETHERNET.

Una vez que se ha comprobado que el programa, tanto para el servidor como para el cliente, compilan y aparentemente funcionan correctamente, se puede comprobar su funcionamiento de una manera sencilla con un cable Ethernet. Se conectará dicho cable entre las dos computadoras, siendo este el medio físico por el cual tiene lugar la transmisión de datos.

Antes de ejecutar el programa, se deben comprobar las direcciones IP adjudicadas por parte de las computadoras a esta conexión exterior. Como se vio en el apartado referente al desarrollo de un código UDP/IP para la transmisión de datos, el programa define las direcciones IP tanto del servidor como del cliente a través de los cuales se realiza la transmisión de datos. Por tanto, se debe definir manualmente la dirección IP adjudicada al origen y al destino del canal de transmisión.

Ya definidas las direcciones IP de ambas computadoras, se conecta el cable Ethernet y se ejecuta el comando ‘ping’. Dicho comando es una herramienta que permite realizar una verificación de una correcta conexión entre un host local y un equipo remoto en una red. En el caso de este trabajo, dichos terminales corresponden a las dos computadoras. Por tanto, este comando permite comprobar que existe una conexión con las direcciones IP especificadas. Un ejemplo de este comando se muestra en la figura 15.

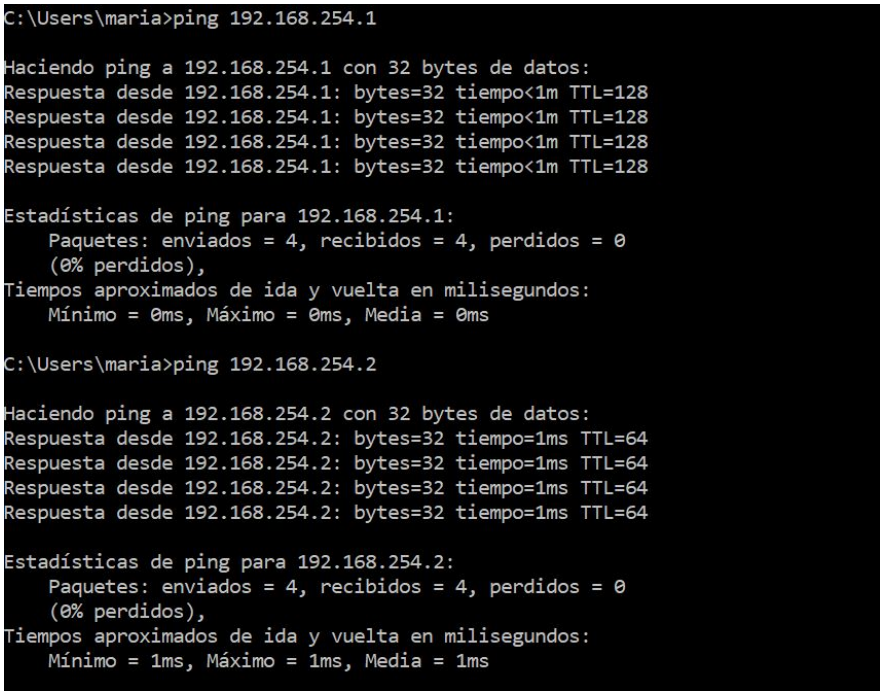

Figura 15. Ejecución del comando ‘ping’.

Como se ha visto en el apartado anterior referente al código, las direcciones IP asignadas a cada computadora son 192.168.254.1 y 192.168.254.3. Aunque en la figura 15 se observa que se usó 192.168.254.2 en vez de 192.168.254.3 en las pruebas iniciales. Por tanto, cuando se pruebe el funcionamiento correcto de los códigos, se demuestra que la transmisión se ha realizado por el canal establecido comprobando que el mensaje ha sido recibido desde la dirección IP correspondiente.

La figura 16 corresponde a la ejecución de esta prueba. La computadora de la izquierda corresponde con el cliente que envía los mensajes, mientras que la que se encuentra en la parte derecha de la imagen representa el servidor que recibe los mensajes.

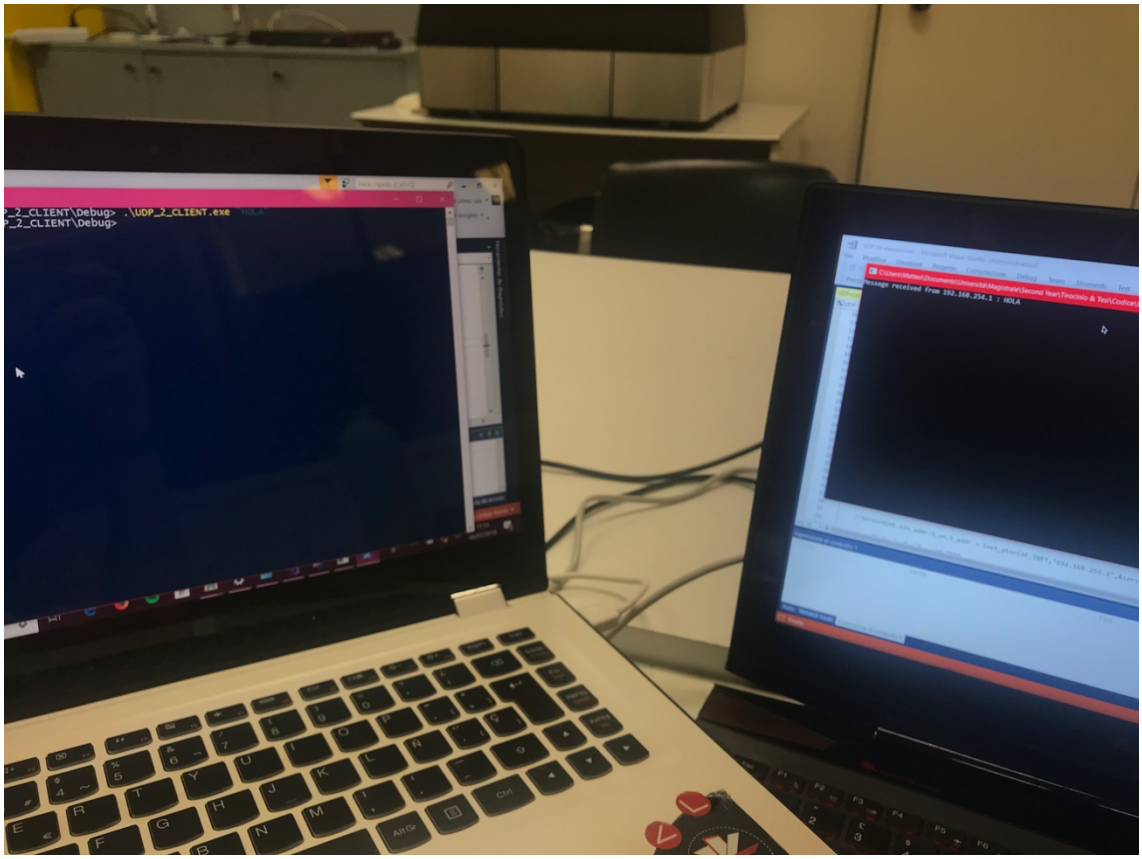

Figura 16. Prueba de los códigos cliente y servidor UDP/IP a través del cable Ethernet.

En la prueba concreta que se muestra en la figura 16, el cliente envía un mensaje con el texto ‘HOLA’ escrito a continuación del ejecutable del programa cliente, aunque no se aprecia claramente en la imagen al ser de un color oscuro. Como era de esperar, la computadora que hace la función de servidor recibe el mensaje escrito.

Además, en el código realizado para el servidor, cuando se recibe un mensaje, este viene acompañado por un mensaje que incluye la dirección IP del emisor del mensaje. En la figura 16 se comprueba como dicho mensaje aparece previo al mensaje enviado y confirma que el mensaje proviene del cliente al venir de la dirección 192.168.254.1.

Por todo ello, se puede confirmar que el código UDP/IP, tanto para el servidor como para el cliente, realizan su función correctamente al conseguir enviar un mensaje entre dos computadoras a través de un cable Ethernet.

2.2 ESTUDIO DE LA PLACA.

Una vez se ha comprobado que la transmisión funciona correctamente a través de los cables Ethernet, se debe estudiar la placa para posteriormente utilizarla en la aplicación requerida en este documento. Dicha placa está formado en líneas generales por tres partes: la parte correspondiente a la alimentación, posible por medio de una fuente o por una entrada USB; los puntos de entrada de los cables Ethernet; y la parte correspondiente a la PowerLine, por donde la señal se transmite a través de cables convencionales.

Esta última parte de la placa se estudiará en mayor profundidad. Para realizar la comunicación submarina mediante un LED, se deben sustituir los dos cables que conectan las placas por un LED y un fotodiodo. Esto no debería modificar el correcto funcionamiento de la transmisión de datos con la variante de que la señal pasa de viajar a través de un cable a mediante luz.

El primer problema que se observa a simple vista consiste en el uso de solamente dos cables entre las dos placas. Para conectar el LED y el fotodiodo se requieren cuatro cables de salida de cada placa. Por tanto, se debe estudiar la placa para conocer cómo convierte los dos cables de transmisión (TX) y los dos de recepción (RX) en uno de transmisión y otro de recepción.

Para ello se estudiará la hoja de características de la placa proporcionada por el fabricante. En la hoja de características se encontraron los circuitos simbólicos de cada una de las partes que forman la placa mencionadas anteriormente. En cuanto a la parte que interesa estudiar para este trabajo, el circuito se representa en la figura 17.

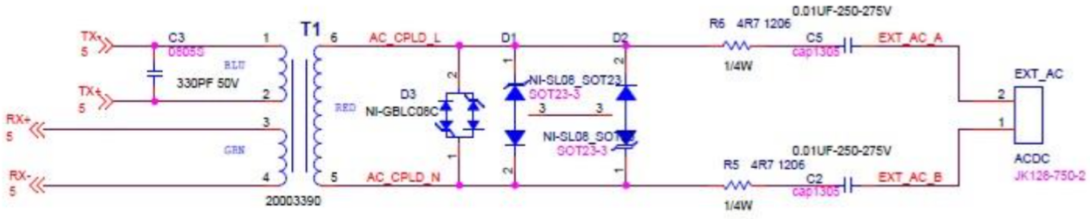

Fig. 17. Circuito TX/RX de la placa.

En la figura 17 se muestra cómo la placa transforma los cuatro cables de transmisión en dos. Para ello recurre a un transformador que permite unificar los cables de transmisión y recepción en dos.

Por tanto, en primer lugar se probará el correcto funcionamiento de la transmisión de datos a través de las placas con dos cables. Esto consiste en conectar dos cables directamente al puerto de salida construido en la placa, que corresponde al bloque en el extremo derecho de la figura 17. Dichos cables irán conectados al puerto de entrada idéntico presente en la segunda placa.

En segundo lugar, se realizará una modificación a las placas para conseguir obtener dos cables TX y dos RX en cada placa. Después de estudiar la placa y su hoja de características correspondiente, se llegó a la conclusión de obtener estos cuatro cables directamente de las cuatro entradas al transformador que se muestran en la figura 17. Dichas entradas aparecen directamente como puertos de entrada/salida en la placa LX200V20, como se muestra en la figura 18.

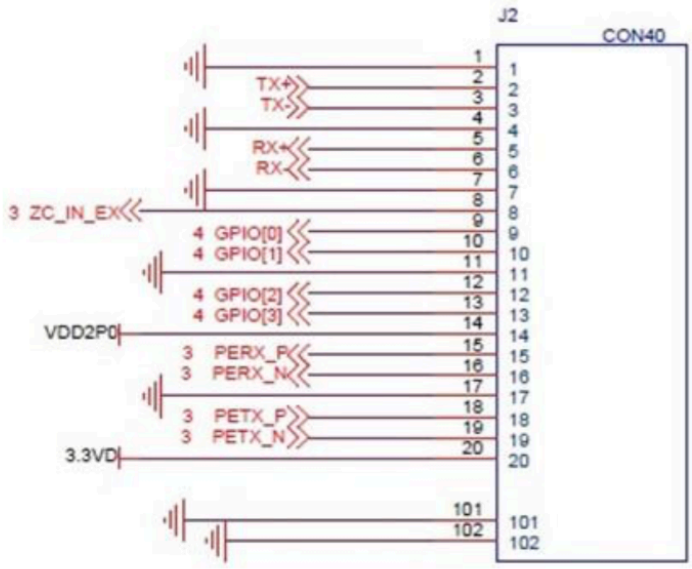

Figura 18. Definición de pines de la placa LX200V20.

Los puertos 2-3 y 5-6 corresponden a estos puertos de TX y RX, respectivamente. Por tanto, más adelante se explicará cómo se han utilizado estos puertos para obtener los cables necesarios.

2.3 TRANSMISIÓN UTILIZANDO LA PLACA.

Como se ha dicho previamente, el siguiente paso a la hora de conseguir los objetivos establecidos en este trabajo consiste en realizar la transmisión a través de las placas. Para ello, en este caso se necesitarán dos cables Ethernet que conecten ambas computadoras a las palcas, las dos placas y dos cables convencionales por los cuales viajan los datos entre las placas.

No se debería observar ningún cambio a la hora de enviar un mensaje entre una computadora y otra. La única diferencia que podría tener lugar consiste en un retardo temporal con respecto a la transmisión directa a través del cable Ethernet. Pero, este retardo sería tan pequeño que no se apreciaría a simple vista a la hora de realizar el envío.

En la figura 19 se muestra cómo se han conectado las placas a las computadoras. Además de los cables para la transmisión se observan otros dos cables conectados a cada placa. Estos corresponden a las tomas de alimentación de las placas que pueden venir directamente de la misma computadora a través de un cable USB.

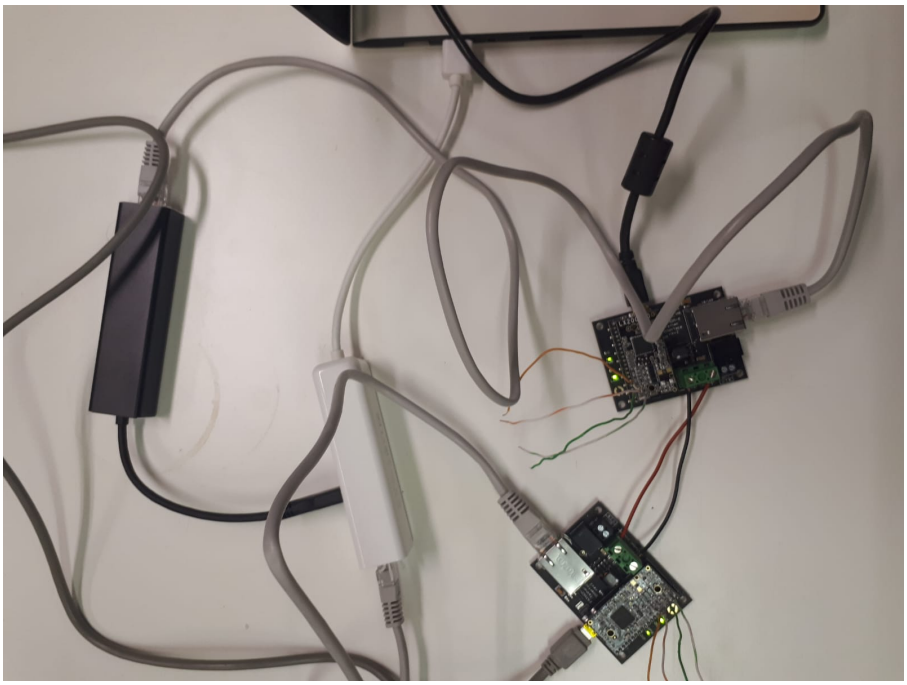

Fig. 19. Configuración convencional de las placas LX200V20.

Al igual que para la prueba con cables Ethernet, se enviaron varios mensajes de una computadora a otra a través del canal de comunicación. Se comprobó que los mensajes fueron recibidos correctamente por parte de la otra computadora y que provenían de la dirección IP establecida, en ambas direcciones.

Por todo ello, se puede concluir que las placas utilizadas funcionan de manera esperada y permiten modificar los datos para poder ser transmitidos a través de cables convencionales sin observar ninguna modificación en la conexión del canal.

2.4 TRANSMISIÓN SEPARANDO TX Y RX.

Llegado a este punto, se ha comprobado que el uso de las placas de manera convencional funciona correctamente. Es decir, montando una configuración donde se conectan los cables Ethernet provenientes de ambas computadoras a las dos placas y conectando dichas placas entre sí con cables convencionales, se consigue transmitir datos a través de las placas.

El siguiente paso consiste en obtener los 4 cables, dos de TX y dos de RX, por parte de cada placa. Para ello, cuando se estudió la placa se decidió obtener dichos cables de los puntos representados en la parte izquierda del circuito de la figura 20.

Estudiando más en profundidad la hoja de características, se observó que dichos cables del circuito representan puertos de entrada/salida de la placa. Por tanto, es posible soldar directamente cables a dichos puertos de la placa al tener contactos metálicos.

En la figura 20 se observa la placa LX200V20 que va montada sobre la placa general con la alimentación, puerto cable Ethernet, etc. El lateral izquierdo de la placa está compuesto por una serie de contactos metálicos que corresponden a todos los puertos de la placa.

Por tanto, observando el esquemático de los puertos de la figura 18 de la hoja de características se soldaron cables en los puertos correspondientes. En la figura 20 se observan los cuatro cables soldados a los dos puertos de TX y los dos puertos de RX.

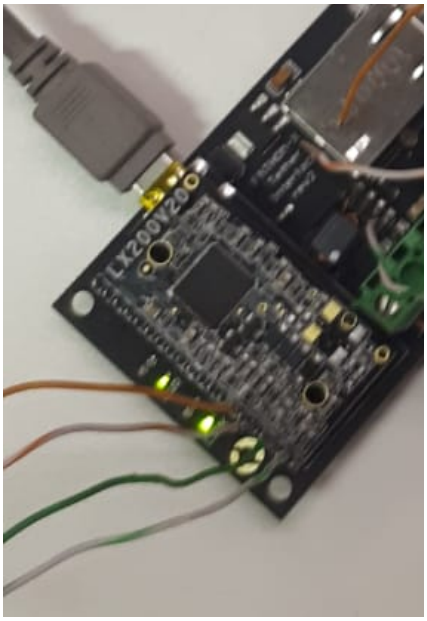

Fig. 20. Conexión de los dos cables TX y los dos RX.

Para alterar de la menor manera posible la configuración inicial sin placas, se extrajeron los cables que se encuentran dentro de los cables Ethernet. Dichos cables presentan colores diversos que representan la utilidad de dicho cable dentro del cable Ethernet general.

En la figura 21 se observa un diagrama con la definición de cada uno de los cables. Con la ayuda de esta tabla, se soldaron los cables al puerto correspondiente de la placa.

Cat5e Wire Diagram for T568B (Straight Through Cable)				
RJ45 Pin #	Wire Color (T568A)	Wire Diagram (T568A)	10Base-T Signal 100Base-TX Signal	1000Base-T Signal
1	White/Orange		Transmit+	BI_DA+
2	Orange		Transmit-	BI_DA-
3	White/Green		Receive+	BI_DB+
4	Blue		Unused	BI_DC+
5	White/Blue		Unused	BI_DC-
6	Green		Receive-	BI_DB-
7	White/Brown		Unused	BI_DD+
8	Brown		Unused	BI_DD-

Fig. 21. Clasificación de cables internos de un cable Ethernet.

Antes de proceder con la última parte de este trabajo, conviene probar que el canal de comunicación funciona igual que en pruebas anteriores. Tanto para las pruebas sin placas como cuando se introdujeron las placas, se estaban creando canales de transmisión de datos probados que deberían funcionar. En cambio, al conectar estos cuatro cables directamente a la placa, se está modificando el funcionamiento habitual de las placas.

Por tanto, es importante probar minuciosamente el funcionamiento del canal de transmisión al estar variando el funcionamiento convencional de las placas. Para esta configuración, se reduce considerablemente el espacio utilizado. Podría no ser necesario la utilización de la placa inferior con los puertos para los cables convencionales al tomarlo directamente de la placa LX200V20. Pero, en este caso, no se decidió prescindir de esta placa al tener la parte referida a la alimentación simplificada gracias al uso de puertos USB y a los puertos de los cables Ethernet que se encuentran conectados directamente a la placa superior.

Para probar la conexión en esta configuración, se conectaron al final de cada uno de los cuatro cables mostrados en la figura 20 conectores macho y hembra de oro para minimizar las perdidas en la conexión. Se podrían haber conectado cables completos de una placa a otra directamente pero no sería práctico para las pruebas futuras que se realizarán. En la figura 22 se observa cómo se han conectado dichos cables extraídos del cable Ethernet.

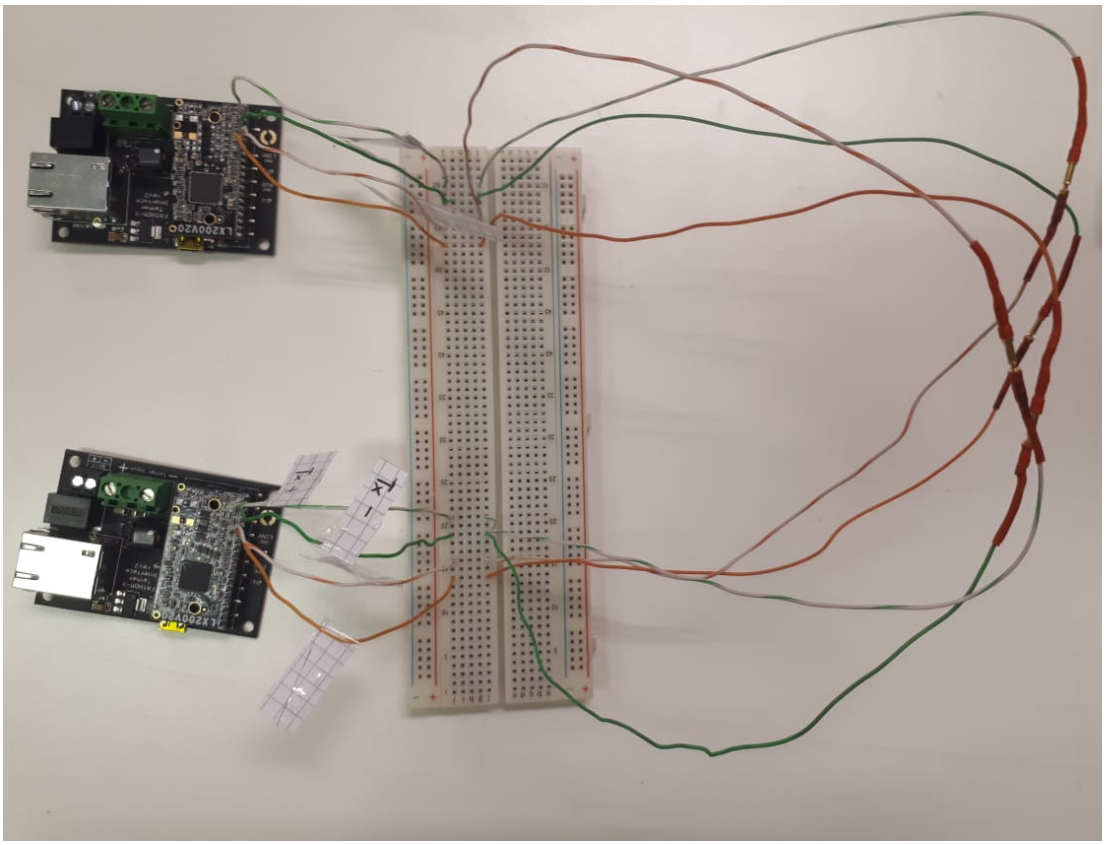

Fig. 22. Conexión de los cuatro cables entre las dos placas.

Una vez montado así el canal de comunicación, se realizaron pruebas como las anteriores para probar que la transmisión tiene lugar correctamente. Al igual que en pruebas anteriores, el mensaje se envió correctamente a través del canal indicando la dirección IP correspondiente a la computadora que envía el mensaje seguido del mensaje escrito.

Por tanto, se puede concluir que la prueba realizada extrayendo los cuatro cables directamente de la placa LX200V20 ha resultado exitosa. Como consecuencia, será posible conectar en un futuro un LED y un fotodiodo en los extremos de los cables para conseguir transmitir el mensaje utilizando una señal luminosa.

CAPÍTULO 3. ESTUDIO DE LA SEÑAL.

3.1 VISUALIZACIÓN DE LA SEÑAL.

Recordando, el objetivo de este trabajo consiste en preparar un canal de comunicación de datos submarino que funcione utilizando la transmisión de datos mediante una señal luminosa a través del agua. Hasta el momento se ha conseguido preparar el canal de transmisión de datos con una configuración que permite la conexión del diodo emisor y el fotodiodo receptor que funciona correctamente. Pero, por ahora los cables por los cuales viaja la señal se conectan directamente entre sí. Esta característica de la configuración actual se aprovechará para el último paso a realizar en este trabajo.

Antes de poder conectar un diodo directamente a los cables del canal, se debe estudiar la señal que viaja por dicho canal. Este paso previo a la utilización de diodos es importante porque en función de las características de la señal se deberá utilizar un diodo y un fotodiodo que se ajusten a dichas características. En caso de no realizar un buen estudio de la señal la transmisión podría no funcionar o incluso se podrían estropear los diodos utilizados.

Para el caso de conectar diodos, la característica más importante a estudiar es la referida a los niveles de tensión que pueda tomar la señal en transmisión. Por tanto, con la ayuda de un osciloscopio digital se pretende observar los niveles de tensión que presenta la señal a través del canal de transmisión cuando se transmiten datos y cuando no.

El osciloscopio digital utilizado es un Virtual Bench de National Instruments. Dicho osciloscopio no presenta una pantalla propia sino que debe ser conectado directamente a una computadora. Este osciloscopio presenta la plataforma informática LabView y tiene instalado un programa que permite observar las señales del osciloscopio en cualquier PC.

Las funciones interesantes para este trabajo que aporta el programa incluyen la capacidad de congelar la señal en un instante temporal para observar con mayor detalle la señal, marcadores para poder observar los niveles de tensión y permite exportar los datos recogidos a lo largo de todo el eje temporal visible a un archivo de tipo Excel con el fin de estudiar con más detalle los datos y poder representar gráficamente dichos datos.

3.2 VISUALIZACIÓN DE LA SEÑAL SIN TRANSMISIÓN.

Para observar la señal se utilizará una de las sondas del osciloscopio. Como se observa en el diagrama de pines de la figura 18, los puertos de las placas se representan como TX^+ y TX^-, para los puertos de transmisión por ejemplo. Se conectaron los cables siguiendo la regla donde el puerto TX^+ de una de las placas debe estar conectado con el puerto RX^+ de la otra placa. Esto se puede comprobar comparando las uniones mostradas en la figura 22 y la tabla de la figura 21.

Es importante diferenciar TX^+ de TX^- debido a que por TX^+ viajará nuestra señal en transmisión, mientras que TX^- se puede tomar como una tierra. En realidad, existen estos dos cables porque la diferencia de voltaje entre ambos debe ser 0 o 1 al transmitir en binario los datos, o acorde a una modulación en amplitud.

Por tanto, a la hora de conectar la sonda a los conectores, esta debe ir conectada al cable que hace de TX^+ del cliente y la tierra al cable TX^- también del cliente. Por esto es importante tener en cuenta qué computadora está funcionando como cliente y cuál como servidor, es decir, ejecutar el programa correspondiente en cada computadora de los desarrollados en la primera parte de este trabajo. En la figura 23 se observa cómo se ha conectado la sonda a la configuración mostrada en la figura 22.

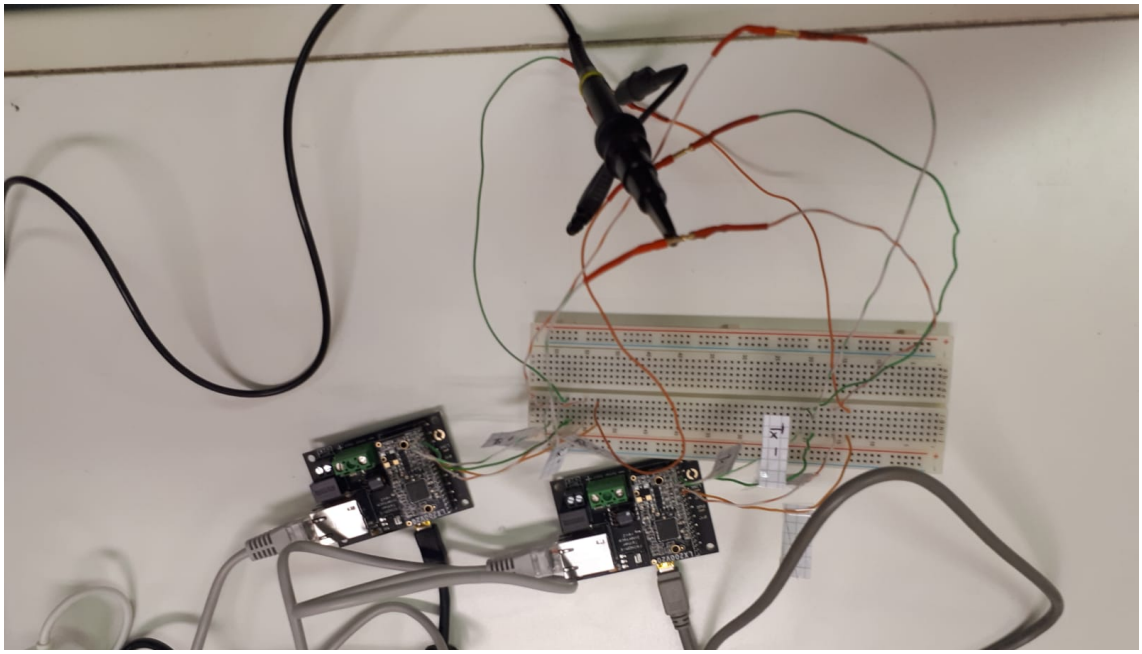

Fig. 23. Conexión de la sonda del osciloscopio.

En la figura 23 queda comprobado lo dicho previamente. La sonda se encuentra conectada al cable de transmisión del cliente TX^+ (blanco y naranja), siendo la placa de la derecha en la figura la conectada a la computadora que hace de cliente; mientras que la tierra de la sonda se encuentra conectada al TX^- de la misma placa, correspondiente al cable naranja liso.

Se ejecutó el programa en el PC y se comenzó a observar la señal a través del canal de comunicación sin enviar ningún mensaje, es decir, cuando no existía transmisión de datos a través del cable. Se esperaba observar una señal plana, pero lo primero que se observó fue algo parecido a lo mostrado en la figura 24.

Fig. 24. Primeras observaciones de la señal sin transmisión.

La mayoría del tiempo la señal era plana como se esperaba al no estar transmitiendo datos por el canal. Pero, cada cierto tiempo aparecía una señal durante un breve periodo de tiempo como se observa en la figura 24. En dicha figura se observa como se ha utilizado la función ‘Stop’ que permite congelar la señal en el tiempo. Esto permite observar la señal con más facilidad ya que sino se encuentra en continuo movimiento.

Una de las ventanas que se observan en la figura 24 corresponden a la señal a lo largo de todo el tiempo que está midiendo el osciloscopio. Esta ventana se puede observar con mayor claridad en la figura 25.

Fig. 25. Gráfica completa del osciloscopio.

La ventana principal de la figura 24 corresponde con la parte central recortada que se observa en la figura 25. Pero, en realidad el osciloscopio mide a lo largo de todo el eje temporal de la figura 25. En esta figura, queda comprobado de manera clara como esta señal breve se repite a lo largo del tiempo y con el mismo espacio temporal entre cada pulso.

Como no se esperaba observar esta señal periódica, el siguiente paso consiste en estudiar dicha señal y conocer su utilidad. El primer paso que se realizó fue observar dicha señal en el dominio de la frecuencia. La misma aplicación proporcionada por el osciloscopio permite observar sobre la misma ventana la transformada de Fourier de la señal. Dicha ventana queda reflejada en la figura 26.

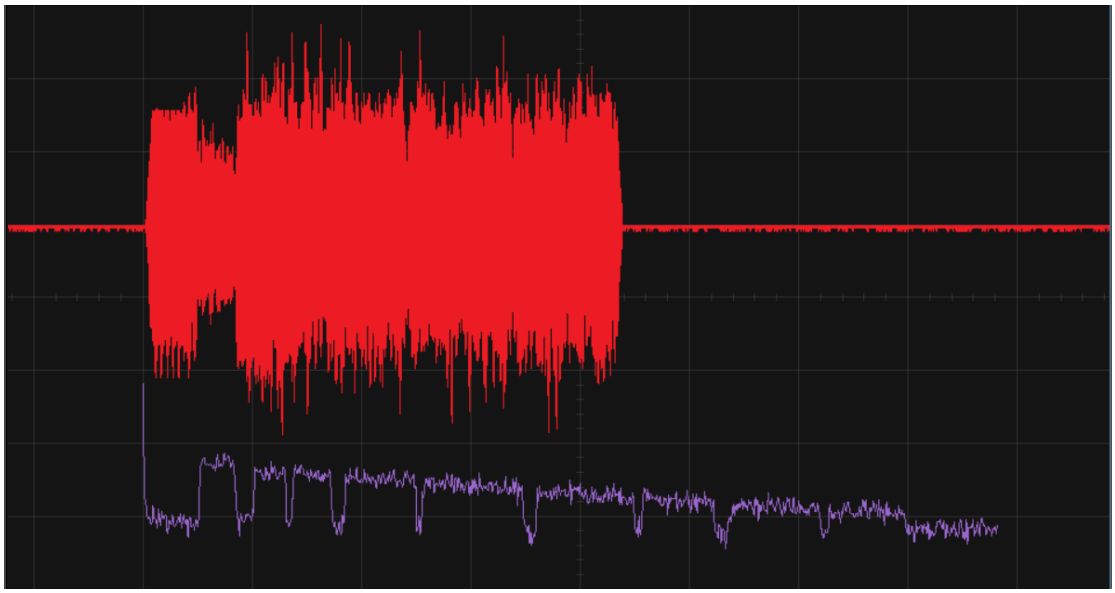

Fig. 26. Señal sin transmisión en el dominio de la frecuencia.

Observando la transformada de la figura 26, representada en la parte inferior en color violeta, no se obtuvo una idea clara de lo que podía representar esta señal. Esto se debe a que no es una representación en frecuencia que represente algo en concreto en el dominio del tiempo. Se observa como existen varias bandas de frecuencia cuya amplitud disminuye al aumentar la frecuencia.

Por tanto, se tuvo que cambiar el enfoque a la hora de estudiar la señal. El siguiente paso realizado se observa en la figura 27, donde se decidió hacer un 'zoom' de la señal reduciendo considerablemente la escala temporal. En la gráfica de la figura 27, cada espaciado del eje temporal corresponde a 50 ns.

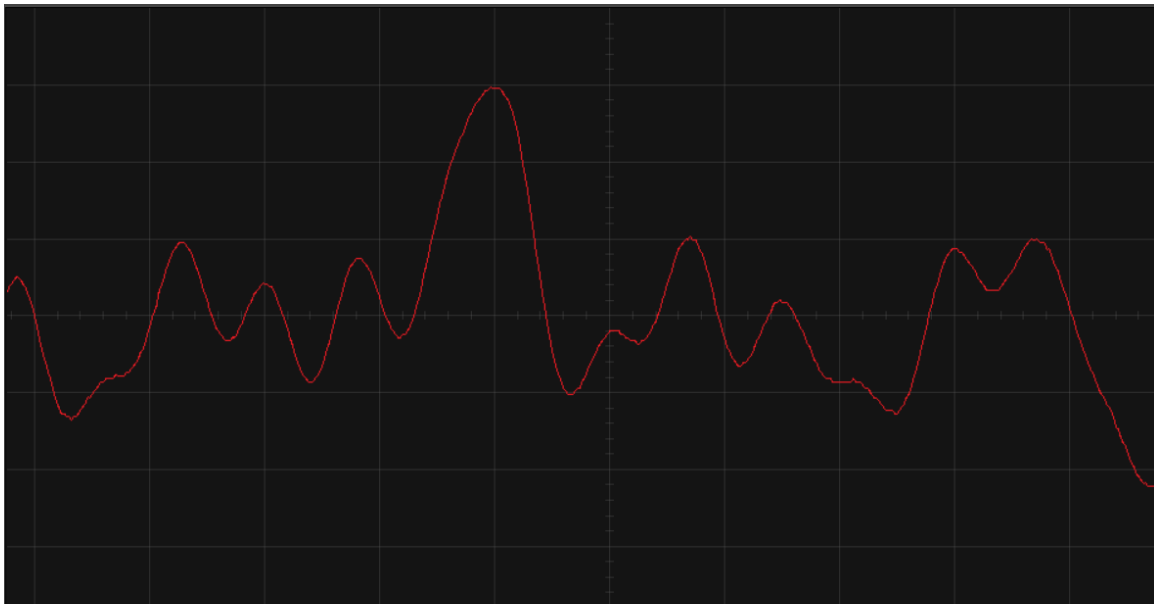

Fig. 27. Gráfico ampliado de la señal sin transmisión.

Estudiando con detalle el gráfico de la figura 27, no se observa un patrón fijo en la señal. Pero, después de realizar varias pruebas y observar diferentes zonas de la señal, se llegó a una conclusión. La figura 27 muestra una de las zonas de la señal donde se muestra de forma mas clara la suposición.

Gracias a esta señal, se ha llegado a la conclusión de que la transmisión a través del canal tiene lugar a través de una modulación de tipo PAM5. Es decir, una modulación en amplitud a 5 niveles de tensión. Dicha modulación concuerda con lo esperado ya que la configuración utilizada con el uso de las placas y el cable Ethernet es de tipo 10BASE-T. En el esquema de la figura 28 se observa como dicho estándar corresponde a una modulación PAM5.

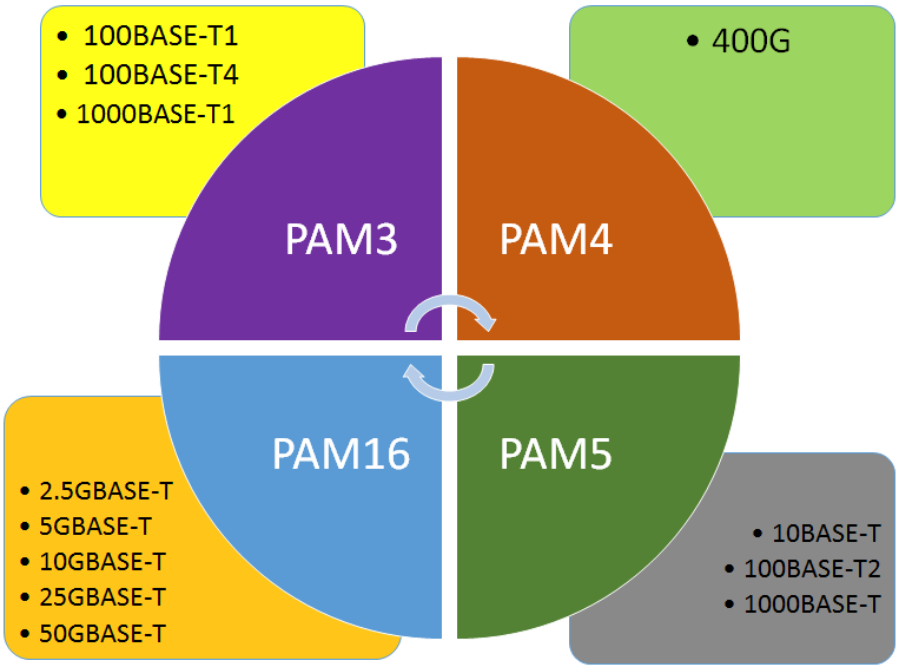

Fig. 28. Esquema de modulaciones.

Volviendo a la figura 27, se puede comprobar que la señal utiliza esta modulación a 5 niveles. En la parte central se observa el pico de máxima tensión, mientras que el mínimo se aprecia en el extremo derecho del gráfico. Estos dos máximos corresponden a los dos niveles extremos de la modulación. Los demás picos corresponden a los dos niveles intermedios y el quinto se corresponde con el nivel de 0 voltios.

Pero, solo con la gráfica no se conocen los niveles de tensión a los cuales corresponden cada nivel de la modulación. Para ello, la aplicación proporcionada por el osciloscopio digital permite mostrar datos numéricos en referencia a tensión pico a pico, máximo, mínimo, media, etc. Dicha ventana queda reflejada en la figura 29.

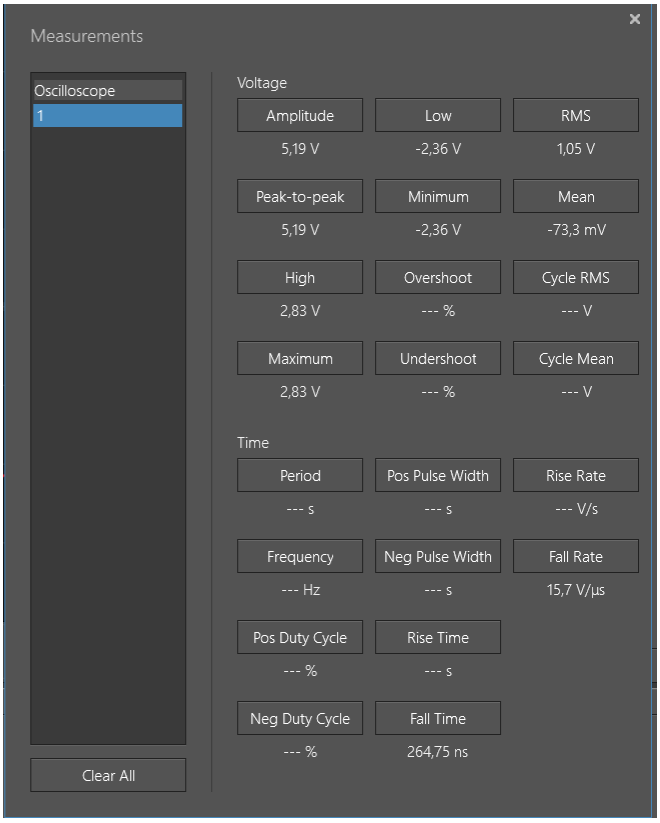

Fig. 29. Datos numéricos de la señal sin transmisión.

En primer lugar, los datos demuestran que la señal tiene una componente continua nula ya que la media se encuentra en torno a los 0 voltios. Por otro lado, el valor máximo se encuentra cercano a los 3 voltios y el mínimo algo más alejado pero también cercano a los -3 voltios.

Se conoce que cada división de la gráfica de la figura 27 corresponde a 1 voltio. Por tanto, se concluye que la señal se transmite a través del cable Ethernet utilizando una modulación a 5 niveles de amplitud: +3, +1, 0, -1 y -3 voltios.

Una vez finalizada esta parte del análisis, cabe destacar uno de los resultados obtenidos durante las distintas pruebas realizadas hasta el momento. Este resultado se muestra en la figura 30.

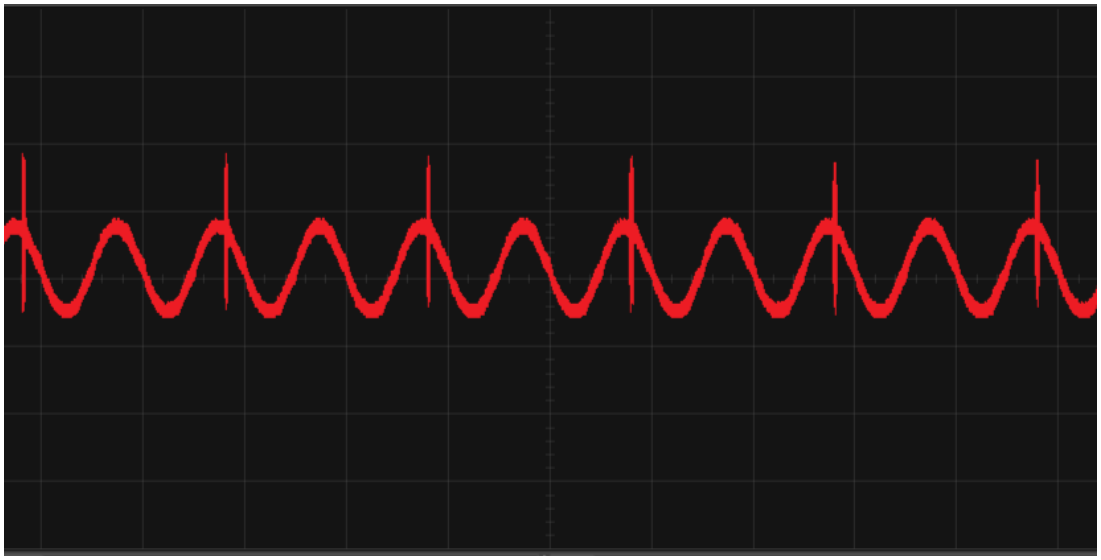

Fig. 30. Señal sin transmisión con la sonda no conectada a tierra.

En la figura 30 se observa un cambio considerable en la señal cuando no existe transmisión de datos a través del canal. Sigue apareciendo la señal periódica cada cierto tiempo que corresponde a los picos de la figura 30. Pero, en este caso no se obtuvo una onda plana cuando no existe esa señal, sino que se observa claramente una onda de tipo sinusoidal.

Una vez se observó este resultado diferente a todos los anteriores a la hora de realizar las medidas, se comprobó que la configuración se encontraba igual que en pruebas anteriores. Como era de esperar, se encontró un cambio considerable en la forma de tomar las medidas. La masa de la sonda se encontraba desconectada del cable de referencia correspondiente al TX^-.

Al no tener conectada la masa de la sonda al cable de TX^-, se pierde la señal de referencia. Esto provoca que la señal pasa de ser plana a tener un valor, ya que la señal de referencia existe precisamente para provocar una señal nula cuando no existe transmisión. Pero, este fallo ha permitido observar esta señal sinusoidal que corresponde a la portante de la señal. Es decir, la señal a través de la cual se transmitirán los datos en pruebas posteriores.

Llegado a este punto, se ha estudiado la señal observada cuando no se transmiten datos entre las dos computadoras pero no se ha resuelto la duda inicial. Se recuerda que todas estas pruebas se han realizado sin transmitir datos y, por tanto, se esperaba observar simplemente una onda plana.

Se ha estudiado el protocolo y cómo se transmitan datos a través de cables Ethernet utilizando el protocolo UDP/IP llegando a la conclusión de que esta señal periódica corresponde a una señal de verificación automática. Es decir, cuando no se están transmitiendo datos, el protocolo envía una señal muy breve a través del canal para avisar al servidor de que el cliente sigue conectado aunque no transmita nada.

3.3 VISUALIZACIÓN DE LA SEÑAL DURANTE LA TRANSMISIÓN.

El siguiente paso a realizar consiste en medir la señal al igual que cuando no se transmitían datos. Por tanto, se procederá a observar los resultados mostrados por el osciloscopio cuando se envía un mensaje entre las dos computadoras.

En las primeras pruebas realizadas, fue difícil obtener un resultado que aportase información. Esto fue debido a que, aunque se envíe un mensaje muy largo entre las dos computadoras, la velocidad de transmisión es tan alta que no es posible apreciarlo a simple vista.

En este momento, es donde se ha recurrido a la función de exportación de datos a Excel por parte del programa, mencionado anteriormente. Se probó a utilizar esta función, pero se comprobó que toma los datos a lo largo del tiempo que se observa en la ventana superior del programa mostrada en la figura 25. Se esperaba que guardase los datos anteriores durante un cierto tiempo y poder observar la señal transmitida que corresponde al mensaje pero al tomar solo los datos de esta ventana sigue siendo demasiado rápido el envío del mensaje.

Como el problema se basa en que el mensaje es tan corto que la señal es muy breve y no se llega a apreciar, se pensó en variar el código original del protocolo para poder alargar el mensaje. Se metió dentro de un ciclo casi infinito la función de enviar el mensaje, provocando que el mensaje se envíe repetidamente muchas veces. La idea es que en vez de enviar un mensaje muy largo, enviamos un mensaje repetidamente muchas veces alargando el mensaje.

Pero, al igual que en las pruebas anteriores con un solo mensaje, solo se observaba la señal de verificación mostrada en figuras anteriores. Por tanto, se debe realizar otro tipo de pruebas para observar la señal en transmisión.

Aunque no se ha podido observar la señal en transmisión, con la señal de verificación se han obtenido suficientes datos para el propósito de estas pruebas. Se recuerda que todas estas pruebas se realizan con el fin de obtener las características de la señal para poder montar el sistema de transmisión mediante luz adecuado para funcionar con estas características.

Se puede concluir que la tensión no sobrepasa los 5 voltios en ningún momento de dicha señal. Por tanto, el siguiente paso que no se expone en este documento, consiste en elegir un diodo emisor de luz y un fotodiodo que funcione correctamente con estas características de entrada y salida, y que se adapte a las necesidades para poder realizar a cabo la transmisión de luz a través del agua.

CONCLUSIÓN.

Se puede concluir que se han alcanzado los objetivos establecidos en este trabajo, como se puede comprobar a lo largo de este documento. Se ha conseguido realizar una serie de programas escritos en lenguaje C++ para crear un protocolo UDP/IP de tipo cliente/servidor, como se muestra en el primer capítulo; se han utilizado estos códigos para transmitir mensajes entre dos computadoras a través de un cable Ethernet y utilizando dos placas LX200V20, como queda reflejado en el capítulo 2; y se ha estudiado la señal que viaja por el canal de transmisión para poder colocar un diodo LED y un fotodiodo para poder transmitir los datos utilizando luz, como se muestra en el tercer capítulo.

Por todo ello, se cree que es posible transmitir datos desde un submarino hasta un punto en la superficie utilizando ondas luminosas en vez de electromagnéticas.

BIBLIOGRAFÍA.

REFERENCIAS BIBLIOGRÁFICAS:

- http://neo.lcc.uma.es/evirtual/cdd/tutorial/transporte/udp.html
- https://es.slideshare.net/EquipoSCADA/unidad-vi-tema-8-scada
- https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario
- https://www.xataka.com/basics/que-es-una-direccion-ip-y-como-puedes-saber-la-tuya
- https://research.ncl.ac.uk/game/mastersdegree/workshops/networkintroduction/intro.pdf
- https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf
- https://en.wikipedia.org/wiki/Network_socket
- http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html
- http://pubs.opengroup.org/onlinepubs/009604499/functions/recvfrom.html
- https://www.bluerobotics.com/store/electronics/fathom-x-r1/
- http://www.manhattan-products.com/usb-20-fast-ethernet-adapter
- http://docs.bluerobotics.com/fathom-x/
- http://docs.bluerobotics.com/fathom-x/#specification-table
- http://docs.bluerobotics.com/fathom-x/#function-diagram
- https://github.com/bluerobotics/fathom-x/
- https://www.incentre.net/tech-support/other-support/ethernet-cable-color-coding-diagram/
- https://en.wikipedia.org/wiki/Carrier-sense_multiple_access_with_collision_detection
- http://www.huntervalleyhotels.co/ethernet-color-code-cat5-wiring.html
-