ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA LIBRERÍA PARA COMUNICACIÓN CON EL ESTÁNDAR ETHERNET ENTRE LA PLATAFORMA ARDUINO Y UNA APLICACIÓN ORIENTADA A IOT UTILIZANDO SOCKETS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

HENRY FERNANDO PROAÑO BENÍTEZ

henry.proano@epn.edu.ec

DIRECTOR: MSc. MOREJÓN TOBAR RAMIRO EDUARDO

ramiro.morejon@epn.edu.ec

Quito, febrero 2020

AVAL

Certifico que el presente trabajo fue desarrollado por Henry Fernando Proaño Benítez, bajo mi supervisión.

MSc. Morejón Tobar Ramiro Eduardo
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Henry Fernando Proaño Benítez, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Henry Fernando Proaño Benítez

DEDICATORIA

Dedico el presente trabajo de titulación a mis padres, por todo el esfuerzo y apoyo que me han brindado durante todo el trayecto, mil gracias.

A mi madre Martha, quien con su amor incondicional siempre ha estado ahí para mi apoyándome y alentándome a seguir a pesar de lo difícil que sea el camino. Gracias por amarme tanto.

A mi padre Henry, quien ha puesto su esfuerzo y confianza en cada una de mis decisiones las cuales han hecho de mi la persona que soy. Gracias por creer en mí.

A mis hermanas Deysi y Karla, con quienes he compartido mis alegrías y tristezas de cada uno de los momentos más fuertes de mi vida.

A mi sobrina Sofia, con su amor y dulzura ha sido una pequeña luz para no rendirse.

A mi amada novia Jeaneth, por llenarme el corazón de alegría y felicidad. Te amo.

A todos mis familiares, amigos y conocidos que de alguna forma han formado parte de este camino en especial a la Hna. Gabriela.

AGRADECIMIENTO

Agradezco a mis padres Henry Proaño y Martha Benítez por todo su esfuerzo, sacrificio y dedicación en educarme y enseñarme los valores de la vida que me han inculcado.

A mis hermanas Deysi y Karla, por ser un apoyo en cada uno de los momentos de mi vida.

A mi sobrina Sofia, quien con su alegría ha hecho que sea llevadera la fatiga.

A mi novia Jeaneth, por acompañarme y apoyarme durante todo el proceso.

A mi director MSc. Ramiro Morejón, por brindarme su apoyo y colaboración durante este proceso que me lleva a culminar una etapa más de vida.

A todos mis familiares, amigos y conocidos que me han acompañado durante este camino y un gracias especial a la Hna. Gabriela.

ÍNDICE DE CONTENIDO

AVAL ……I
DECLARACIÓN DE AUTORÍA..... II
DEDICATORIA..... III
AGRADECIMIENTO..... IV
ÍNDICE DE CONTENIDO..... V
RESUMEN ……VII
ABSTRACT..... VIII
1. INTRODUCCIÓN..... 1
1.1 OBJETIVOS..... 3
1.2 ALCANCE..... 3
1.3 MARCO TEÓRICO..... 4
1.3.1 INTERNET DE LAS COSAS (IOT)..... 4
1.3.2 APLICACIONES DEL INTERNET DE LAS COSAS..... 6
1.3.3 VISUAL STUDIO 2017..... 7
1.3.4 ARDUINO..... 9
1.3.5 ARDUINO UNO..... 11
1.3.6 ETHERNET SHIELD V1..... 13
1.3.7 ESTÁNDAR ETHERNET..... 14
1.3.8 SENSORES ARDUINO..... 15
1.3.9 COMUNICACIÓN DE BUS I2C..... 15
1.3.10 CONEXIÓN Y FUNCIONAMIENTO BUS I2C..... 16
1.3.11 PROTOCOLO Y PUERTOS TCP..... 18
1.3.12 CHIP WIZNET W5100..... 20
2. METODOLOGÍA..... 22
2.1. DESCRIPCIÓN GENERAL..... 22
2.2. ESTRUCTURA..... 23
2.3. IMPLEMENTACIÓN..... 25
2.3.1 NUEVA LIBRERÍA ETHERNET..... 26
2.3.2 DISEÑO DE LA RED LAN: 1 SERVIDOR – 2 CLIENTES..... 32
2.3.2.1 Recursos y descripción de materiales..... 32
2.3.2.2 Diagrama y diseño..... 38
2.3.2.3 Aplicación en Arduino..... 42

2.3.2.4 Aplicación en Visual Studio 55
2.3.3 DISEÑO DE LA RED LAN: 2 SERVIDORES – 2 CLIENTES 62
2.3.3.1 Recursos y descripción de materiales 62
2.3.3.2 Diagrama y diseño 63
2.3.3.3 Aplicación en Arduino 64
2.3.3.4 Aplicación en Visual Studio 68
3. RESULTADOS Y DISCUSIÓN 72
3.1. DISEÑO DE LA RED LAN: 1 SERVIDOR – 2 CLIENTES 72
3.1.1 PRUEBA DE CONECTIVIDAD ETHERNET 76
3.1.1.1 Ping 77
3.1.1.2 Comunicación Serial 77
3.1.1.3 Wireshark 78
3.1.2 PRUEBA DE FUNCIONAMIENTO 79
3.2. DISEÑO DE LA RED LAN: 2 SERVIDORES – 2 CLIENTES 80
3.2.1 PRUEBA DE CONECTIVIDAD ETHERNET 81
3.2.2 PRUEBA DE FUNCIONAMIENTO 82
3.3. REGISTRO DE DATOS 84
4. CONCLUSIONES Y RECOMENDACIONES 86
4.1 CONCLUSIONES 86
4.2 RECOMENDACIONES 88
5. REFERENCIAS BIBLIOGRÁFICAS 89
ANEXOS 94

RESUMEN

Aunque una plataforma de Arduino tiene un gran potencial para el desarrollo, tiene varias limitaciones en la comunicación Ethernet; esta ha sido la causa principal para elegir otro sistema integrado para la comunicación Ethernet para proyectos finales. Es por eso por lo que no hay tantos ejemplos, ni soporte en Ethernet. Por lo general, la placa Arduino y el escudo Ethernet se usan solo como servidor WEB para la mayoría de las aplicaciones.

En este trabajo, para resolver sus limitaciones y dificultades al usar la librería existente para la transferencia de datos, se ha propuesto la creación de una librería Ethernet de Arduino. La librería propuesta utilizará comunicación de socket entre una placa de Arduino y una plataforma de computadora que ejecuta una aplicación. La comunicación se probará con datos provenientes de sensores conectados a la placa como periféricos. La placa Ethernet Shield mantendrá el proceso de comunicaciones Ethernet utilizando sockets en el esquema LAN. Se utilizan varias herramientas de prueba para garantizar y monitorear, tales como: ping, comunicación de consola y Wireshark. Las pruebas a nivel de aplicación se llevan a cabo apoyando la comunicación de extremo a extremo. Los datos recopilados en el extremo Arduino se envían a la aplicación Visual Studio y se muestran de una manera más fácil de usar. Esta es una solución para demostrar el funcionamiento y el comportamiento de la nueva librería en un entorno IOT.

PALABRAS CLAVE: Librería Ethernet, Comunicación Ethernet, Arduino, Ethernet Shield, Protocolo TCP.

ABSTRACT

Even though the Arduino´s one platform has a power full for development, it has several limitations on Ethernet communication; this has being the main cause to choose other embedded system for Ethernet communication for final projects. That is why there are not so many examples, nor support on Ethernet. Usually, the Arduino board and the Ethernet shield are use only as WEB server for most of the applications.

In this work, in order to solve its limitations and difficulties using the existing library for data transfer, it has been proposed the creation of an Arduino´s Ethernet library. The proposed library will use socket communication between an Arduino´s one board and a computer platform running an application. Communication will be tested data coming from sensors attach to the board as peripherals. The Ethernet Shield board will be keep the Ethernet communications process using sockets on LAN schema. Several tool for testing are used to ensure and monitoring, such as: ping, console communication and Wireshark. Tests at application level are conducted by supporting end to end communication. Data collected at the Arduino end are sent to Visual Studio application and display in a more user-friendly way. This being a solution to demonstrate the operation and behavior of the new library in an IOT environment.

KEYWORDS: Ethernet library, Ethernet communication, Arduino, Ethernet Shield, TCP Protocol.

1. INTRODUCCIÓN

En el trabajo cotidiano utilizando sistemas embebidos cada una de las plataformas ofrece características particulares que las hacen útiles para determinado conjunto de tareas, las mismas que están relacionadas con la capacidad de procesamiento, el tamaño de la memoria y en forma particular los periféricos de comunicaciones. Por ejemplo, las plataformas de Arduino están concebidas para la captura de datos a través de los sensores que posteriormente el usuario utilizara de la manera más apropiada según su necesidad. No obstante, la parte de comunicación Ethernet ha tenido un retraso importante.

Por lo dicho anteriormente, los trabajos de titulación desarrollados en la Carrera de Electrónica y Telecomunicaciones [1] [2], que involucran la presencia de sensores y la comunicación en un entorno de red, han debido utilizar en las soluciones propuestas dos módulos de sistemas embebidos interconectados entre sí para conseguir la funcionalidad requerida. Una primera plataforma reducida con herramientas de desarrollo en bajo nivel [3] para realizar la comunicación y captura de datos de los sensores y, por otro lado, una segunda plataforma reducida con herramientas de desarrollo en alto nivel para resolver el acceso a una Red Ethernet y transferir archivos. De la solución descrita arriba, se identifica la dificultad de realizar el acceso a una Red Ethernet con funciones desarrolladas en un lenguaje de bajo nivel, en otras palabras, no se dispone de la librería de soporte para dicha forma de comunicación.

Las limitaciones que se presentan en la plataforma de Arduino para la comunicación Ethernet son varias; por ejemplo, las librerías ofrecidas solo permiten desplegar un servidor a la vez por sesión, también con la librería Ethernet que se hizo el análisis que es la versión 1.1.2 no se puede mantener varias sesiones simultáneas, esto ocurre debido a que el único puerto disponible es el puerto 80 que es del servicio de HTTP. Por esta razón, se ha considerado como una solución utilizar otro sistema embebido mejor desarrollado en la comunicación Ethernet para varios proyectos. Sin embargo, esto no es una solución apropiada, debido a un deficiente uso de memoria y recursos.

Otras dificultades encontradas es la utilización del Arduino como servidor dentro de una comunicación Ethernet, en lugar de utilizarlo como cliente, y el uso de una página web para colgar los datos en la red que posteriormente el programador los utilizará, este el uso más conocido debido a su facilidad. Este funcionamiento hace que no sea muy amigable el desarrollo de la comunicación Ethernet en Arduino por lo que el proceso de comunicación del Arduino solo ha sido concebido de trabajar como servidor.

Constituye un error de concepción pensar que atender los dispositivos remotos a través de un interface hombre máquina (Web) corresponde a la comunicación de los parámetros capturados. Un dispositivo IOT, que se presenta a través de un servicio basado en el protocolo HTTP y permite que cualquier usuario remoto haciendo uso de un navegador de Internet pueda acceder al dispositivo IOT e interactuar con página web embebida en él, no constituye una aplicación de transferencia de datos.

La seguridad se ve comprometida seriamente más allá de mecanismos de autenticación que pudieren implantarse, debido a que la página está disponible y utiliza un número de puerto bien conocido. Debido a esto, se propone como alternativa la comunicación entre el dispositivo IOT y un aplicativo de gestión a través del protocolo TCP con la utilización de sockets para garantizar la privacidad de la comunicación. Las rutinas que permiten esta funcionalidad en la plataforma Arduino y su módulo Ethernet Shield no han tenido amplia difusión. El desarrollo de una librería que contenga la rutinas y funciones que permitan la comunicación entre el dispositivo IOT realizado con la plataforma Arduino y una aplicación de propósito específico desarrollada en Visual Studio [4].

La plataforma Arduino y el módulo Ethernet Shield permiten el desarrollo de aplicaciones en virtud del manejo de código abierto [5]. Los prototipos y aplicaciones están basados en hardware y software flexibles con el uso de sensores y actuadores que han tenido una amplia difusión en aplicaciones de IOT [6]. Además, las librerías son rutinas cuyo código ha sido realizado por terceros y que son incluidos en la implantación de sketches con funcionalidades específicas liberando al desarrollador de la interacción con un dispositivo de hardware específico. “Esto facilita significativamente el proceso de programación y permite la abstracción de los dispositivos periféricos haciendo que las rutinas se dediquen a la aplicación propiamente dicha [7].”

Los sistemas embebidos con aplicaciones que requieren algún método de comunicación utilizan de preferencia alguna forma de comunicación serial. Sin embargo, las aplicaciones de IOT que se desarrollan en ambientes de área extendida utilizan el estándar Ethernet que es el más adecuado para la intercomunicación entre el dispositivo IOT y de aplicaciones de gestión.

Se propone la creación de una librería Ethernet de Arduino para solventar las limitaciones y dificultades que tiene la librería existente en la transferencia de datos. La nueva librería será evaluada por medio de pruebas de comunicación haciendo uso de sensores sobre los periféricos de Arduino para recolectar los datos y el módulo Ethernet Shield que será el encargado del proceso de envío sobre Ethernet usando sockets. Estos datos llegarán a

una aplicación de Visual Studio que serán visualizados de una manera más amigable al usuario. Siendo esta una solución para demostrar el funcionamiento y comportamiento de la nueva librería en un ambiente IOT.

1.1 OBJETIVOS

El objetivo general de este proyecto de titulación es:

- Desarrollar una librería de la plataforma Arduino con comunicación Ethernet utilizando sockets.

Los objetivos específicos de este Proyecto Integrador son:

Comprender el funcionamiento de Arduino y de Ethernet Shield.

- Determinar los requerimientos y las funciones para realizar la comunicación TCP usando Sockets.
- Diseñar una librería que comunique el Arduino con una aplicación en Visual Studio.
- Diseñar una aplicación en Visual Studio donde se pueda gestionar la información recolectada de los sensores.
- Evaluar y realizar las pruebas de comunicación entre el dispositivo IOT y la aplicación de gestión.

1.2 ALCANCE

Los dispositivos IOT están siendo utilizados de manera creciente para realizar tareas en el hogar y en los lugares de trabajo. Dichas tareas involucran sensores y actuadores que deben responder a sistemas de gestión de propósito específico, así la captura de datos para monitorear un proceso y las acciones requeridas para administrar el sistema remotamente se dificultan si cada dispositivo IOT es accedido mediante su propia página Web. Por ejemplo, para controlar los artefactos del hogar, se requiere que el usuario deba acceder a un sinnúmero de páginas Web por cada dispositivo IOT.

El uso de sockets permite que una aplicación de gestión establezca sesiones de extremo a extremo con varios dispositivos IOT y tráfico en ambos sentidos. El desarrollo de un algoritmo para la plataforma Arduino requiere de un conjunto de funciones y rutinas que

simplifiquen este método de comunicación, las cuales deben estar albergadas dentro de una librería de acceso público.

Los trabajos de titulación desarrollados [1] [2] involucran la presencia de dos módulos de sistemas embebidos interconectados entre sí en aplicaciones orientadas a IOT [8], generando dificultad en su realización y en la optimización de los recursos al utilizar dos plataformas con herramientas y lenguajes de programación diferentes. Se propone realizar el acceso a una Red Ethernet con funciones desarrolladas en un lenguaje de bajo nivel, por lo que se dispone a desarrollar la librería de soporte para dicha forma de comunicación.

El presente proyecto contempla una red de sensores que estarán conectados a la plataforma Arduino mediante una comunicación I2C. Cada dispositivo IOT que está conectado al bus I2C está asociado a un Socket que será previamente seleccionado por el programador. El módulo Ethernet Shield posibilita la transmisión de paquetes TCP con la información o estado de los dispositivos IOT la misma que será gestionada por una aplicación desarrollada en Visual Studio. La aplicación de gestión reside en una estación remota que se conecta vía Ethernet [9].

El alcance del proyecto se limita al diseño de la librería de Arduino para la comunicación Ethernet por medio de sockets utilizando el protocolo TCP. La información que intercambian la aplicación de gestión y los dispositivos IOT será transparente para los mecanismos de comunicación. Se realizará un ejemplo en el que un dispositivo IOT se comunique con la aplicación de manera que ilustre la utilización de las funciones de la librería. La librería y la aplicación de gestión (prototipo) serán desarrolladas en lenguaje C.

1.3 MARCO TEÓRICO

1.3.1 INTERNET DE LAS COSAS (IOT)

“El Internet de las Cosas, por sus siglas en inglés como IOT (Internet Of Things), es el conjunto de dispositivos interconectados entre sí que se encargan de medir, recopilar y enviar datos a un servidor centralizado o la nube [10].” Una vez que estos datos son tratados y se ha extraído la información considerada importante, los dispositivos IOT pueden recibir del servidor o de la nube las instrucciones para realizar una acción específica.

“El Internet de las Cosas (Internet Of Things), ha ido evolucionando con el tiempo siendo una de las tecnologías con mayor crecimiento en los últimos años siendo así utilizada con la conexión de cualquier dispositivo conectado a internet [6].” El propósito de IOT (Internet Of Things) es que el usuario tenga acceso a la información de dicho dispositivo desde cualquier punto del planeta donde esté conectado y pueda manejar esta información de la manera más adecuada.

Esta tecnología apunta a que todo dispositivo se conecte a la red global pero también se puede usar para una Red LAN (Local Area Network). Normalmente, el usuario requiere que su información no salga de su red porque no necesita que conozcan de ella. Existen varios dispositivos para manejar datos de extremo a extremo dentro de una Red LAN, pero este proyecto utilizará dispositivos de código abierto como la plataforma Arduino y el módulo Ethernet Shield.

El IOT es un modelo que abarca varias tecnologías de comunicación que están compuestas de sensores y actuadores para transferir datos mediante la red. “Los dispositivos que están inmersos dentro del IOT son varios como smartphones, televisores, automóviles, relojes, etc. Ya no solo las personas, sino también objetos o cosas cotidianas se conectan a la red para aprovechar sus beneficios [11].”

Un ambiente IOT consiste en dispositivos inteligentes habilitados para la red que utilizan procesadores integrados, sensores y hardware de comunicación para recopilar, enviar y proceder sobre la adquisición de datos. Los dispositivos IOT comparten la información del sensor que recopilan al conectarse a una puerta de enlace IOT u otro dispositivo de borde donde los datos se transfieren a la nube para ser analizados. Estos dispositivos suelen comunicarse con otros dispositivos y actúan sobre la información que obtienen unos de otros. Los dispositivos realizan la mayor parte del trabajo sin intervención humana, aunque las personas pueden interactuar con ellos, por ejemplo, para configurarlos, darles instrucciones o acceder a la información que necesitan [12].

En la Figura 1.1 se indica el ejemplo de un sistema IOT desde la recolección de datos hasta el análisis de datos y la toma de decisiones.

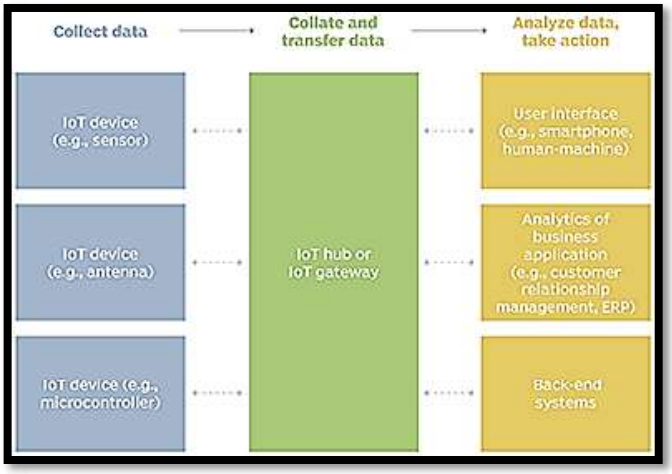

Figura 1.1. Ejemplo de un sistema IOT *[13]*.

"Los protocolos de conectividad, redes y comunicación utilizados con estos dispositivos dependen en gran medida de las aplicaciones específicas de IOT implementadas [12]."

1.3.2 APLICACIONES DEL INTERNET DE LAS COSAS

"Existen numerosas aplicaciones de Internet de las cosas en el mundo real, como la IOT del consumidor, la IOT empresarial y la IOT industrial (IIOT). Las aplicaciones de IOT abarcan numerosos campos, incluyendo automotriz, telecomunicaciones, energía y más [12]."

En la Figura 1.2 se indican diferentes aplicaciones que abarca IOT, algunas de ellas son:

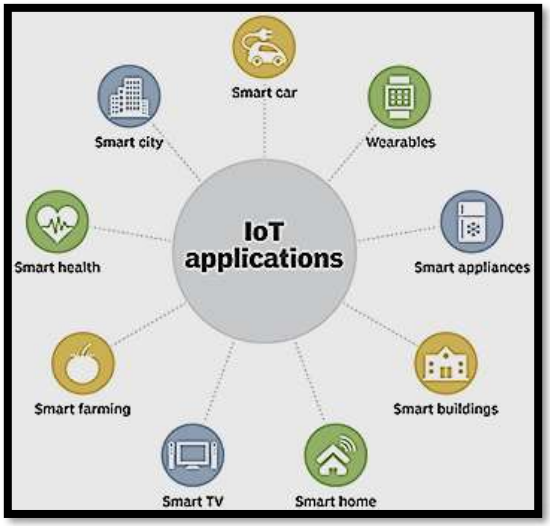

Figura 1.2. Aplicaciones IOT**1.3.2** *[13]*.

- **Ciudades Inteligentes:** integración de los servicios de seguridad, parqueaderos inteligentes, optimización del transporte, tráfico a tiempo real, administración del agua y del sistema eléctrico, etc.
- **Edificios inteligentes:** eficiencia y ahorro energético, control y seguridad, control de acceso a las instalaciones, apagado automático de los equipos cuando no estén en uso, cerraduras inteligentes, etc.
- **Casas inteligentes:** control y gestión de la luz, agua y electrodomésticos, termostatos inteligentes, servicio de salud y educación en el hogar, etc.
- **Automóviles inteligentes:** monitoreo del estado del vehículo, gestión inteligente de la energía, sensores de posición, presencia y proximidad, localización por GPS, control de velocidad del vehículo, etc.
- **Salud:** monitoreo de enfermedades, mejoramiento en la atención, diagnósticos remotos, pulseras y cinturones interactivos, seguimiento de la medicación, monitoreo de dietas, etc.
- **Agricultura y medio ambiente:** medición y control de la contaminación en el ambiente, monitoreo y pronostico del clima, gestión de residuos, detección y control de plagas, indicación del momento óptimo de recolección de los alimentos, etc.

1.3.3 VISUAL STUDIO 2017

En la Figura 1.3 se indica el logotipo de Visual Studio 2017.

Figura 1.3. Logotipo de Visual Studio 2017 *[14]*.

El entorno de desarrollo integrado de Visual Studio es una plataforma que se puede usar para editar, depurar y compilar código, y luego publicar una aplicación. Un entorno de desarrollo integrado (IDE) es un programa de abundantes características que se puede usar para muchos aspectos del desarrollo de software. Además del editor y depurador estándar que proporciona la mayoría de los IDE, Visual Studio incluye compiladores,

herramientas para completar códigos, diseñadores gráficos y muchas más funciones para facilitar el proceso de desarrollo de software [15].

“Visual Studio está disponible para Windows y Mac. Visual Studio para Mac tiene muchas de las mismas características que Visual Studio 2017 de Windows, y está optimizado para el desarrollo de aplicaciones móviles y multiplataforma [15].”

Hay tres ediciones de Visual Studio 2017: Community, Professional y Enterprise.

En la Figura 1.4 se indica el entorno de desarrollo integrado (IDE) de Visual Studio 2017.

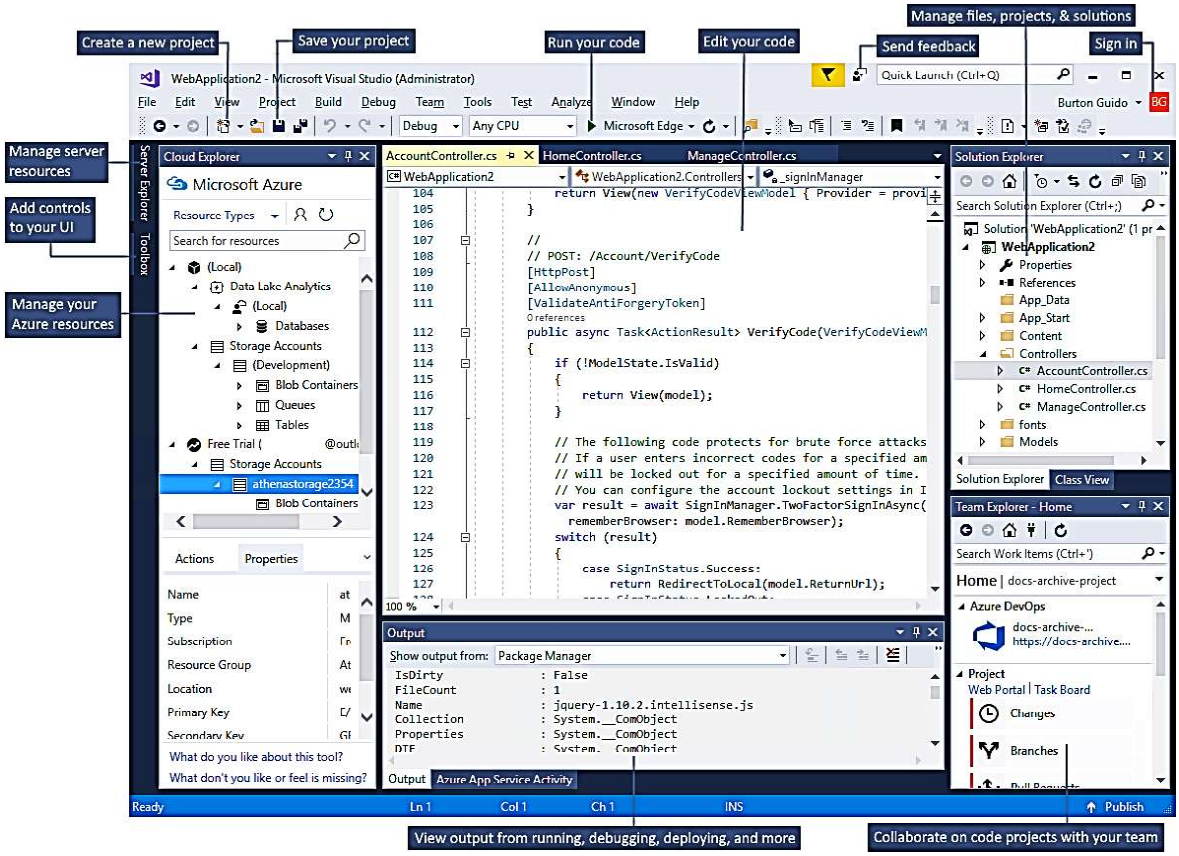

Figura 1.4. Entorno de desarrollo integrado de Visual Studio 2017 [15].

La Figura 1.4 muestra Visual Studio 2017 con un proyecto abierto y varias ventanas de herramientas, entre las más importantes están [15]:

- **El Explorador de soluciones** (arriba a la derecha) le permite ver, navegar y administrar sus archivos de código. El Explorador de soluciones puede ayudar a organizar su código agrupando los archivos en soluciones y proyectos.

- **La ventana del editor** (centro) muestra el contenido del archivo. Aquí es donde puede editar el código o diseñar una interfaz de usuario, como una ventana con botones y cuadros de texto.
- **La ventana de resultados** (en la parte inferior central) es donde Visual Studio envía notificaciones como mensajes de error y de depuración, advertencias del compilador, mensajes de estado de publicación y más. Cada fuente de mensaje tiene su propia pestaña.
- **Team Explorer** (parte inferior derecha) le permite realizar un seguimiento de los elementos de trabajo y compartir el código con otras personas utilizando tecnologías de control de versiones como Git y TFVC (Team Foundation Version Control).

1.3.4 ARDUINO

En la Figura 1.5 se indica el logotipo de Arduino.

Figura 1.5. Logotipo Arduino [16].

Arduino es una plataforma electrónica de código abierto (Open Source) basada en hardware y software libre. Las placas Arduino pueden leer diferentes tipos de entradas a través de sus periféricos y convertirlas en una salida: activar un motor o encender un LED. Puede decirle a su tarjeta qué debe hacer enviando un conjunto de instrucciones al microcontrolador de la tarjeta. Para hacerlo, utiliza el “Arduino Programming Language” (basado en Wiring) y el “Arduino Development Enviroment” (basado en Processing) [17].

El IDE (Integrated Development Environment) de Arduino es un software de código abierto que está escrito en Java y funciona en una variedad de plataformas: Windows, Mac y Linux. El IDE permite escribir código en un entorno con resaltado de sintaxis y otras características que facilitan la codificación para luego cargar fácilmente dicho código en la placa como se aprecia en la Figura 1.6.

“La estructura básica del lenguaje de programación Arduino es bastante simple y se organiza en al menos dos partes o funciones que encierran bloques de declaraciones [17].” Estas dos funciones son:

1. **Setup ():** es la primera función a la que llama el programa por lo que es asignada para inicializar las diferentes librerías o la comunicación serie.
2. **Loop ():** es la segunda función que llama el Sketch siendo esta el núcleo de cualquier aplicación en Arduino porque lleva la mayor carga de trabajo. Esta es una función que ejecuta los códigos que el programador le incluyo de forma continua.

En la Figura 1.6 se indica una captura de pantalla del IDE de Arduino 1.8.7.

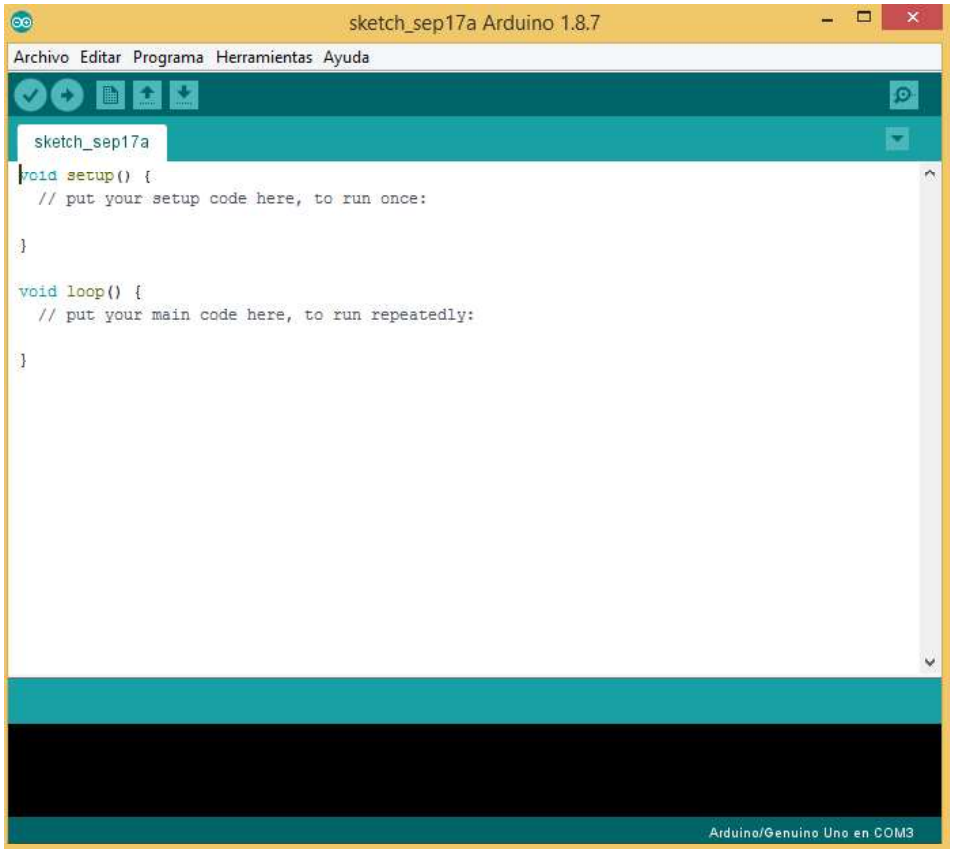

Figura 1.6. Captura de pantalla del IDE de Arduino [5].

El Arduino contiene varias partes e interfaces diferentes juntas en una sola placa, entre las cuales están:

- **Un conector de alimentación**, que proporciona energía tanto al Arduino como un bajo voltaje adecuado que puede alimentar componentes conectados como LEDs y varios sensores o actuadores.

- **Un microcontrolador**, es el chip principal que le permite programar el Arduino para que pueda ejecutar comandos y tomar decisiones. “El chip varía según el tipo de Arduino, pero generalmente son controladores Atmel, generalmente ATmega8, ATmega168, ATmega328, ATmega1280 o ATmega2560. Las diferencias entre estos chips son sutiles, pero la mayor diferencia es la cantidad de memoria integrada [18].”
- **Un conector serie**, que en la mayoría de las tarjetas nuevas se implementa a través de un puerto USB estándar. Este conector le permite comunicarse con la placa desde su computadora, así como cargar nuevas aplicaciones en el dispositivo.
- **Un conjunto de pines E/S (Entrada/Salida)**, que se utilizan para conectar con varios componentes o dispositivos que se puede usar con el Arduino. Existen 2 tipos:
    - **Pines digitales**, que pueden leer y escribir un solo estado, sea alto o bajo, encendido o apagado.
    - **Pines analógicos**, que pueden leer un rango de valores, son útiles para un control más preciso.

Estos pines están dispuestos en un patrón específico, de modo que, si se compra una placa adicional diseñada para encajar en ellos, generalmente llamado "Shield", debería encajar fácilmente según el modelo de Arduino.

1.3.5 ARDUINO UNO

En la Figura 1.7 se indica la imagen de un Arduino UNO.

Figura 1.7. Arduino UNO [19].

Arduino Uno es una placa para microcontroladores basado en el ATmega328P. Tiene 14 pines de entrada / salida digital (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP y un botón de reinicio [20].

En la Tabla 1.1 se indica las especificaciones técnicas del Arduino UNO.

Tabla 1.1. Especificaciones Técnicas Arduino UNO [20].

Especificación Técnica	Valor
Microcontrolador	Atmega328P
Voltaje de Operación	5 V
Voltaje de Entrada (Recomendado)	7 – 12 V
Voltaje de Entrada (Limite)	6 - 20 V
Pines Digitales E/S	14 (de los cuales 6 proporcionan salida PWM)
Pines Digitales PWM E/S	6
Pines de Entrada Analógicos	6
Corriente DC por Pin E/S	20 mA
Corriente DC por Pin 3.3V	50 mA
Memoria Flash	32 KB (Atmega328P) de los cuales 0.5 KB utilizados para el arranque
SRAM	2 KB (Atmega328P)
EEPROM	1 KB (Atmega328P)
Velocidad del Reloj	16 MHz
Longitud	68.6 mm
Ancho	53.4 mm
Peso	25 g

1.3.6 ETHERNET SHIELD V1

En la Figura 1.8 se indica la imagen de la plataforma Ethernet Shield V1.

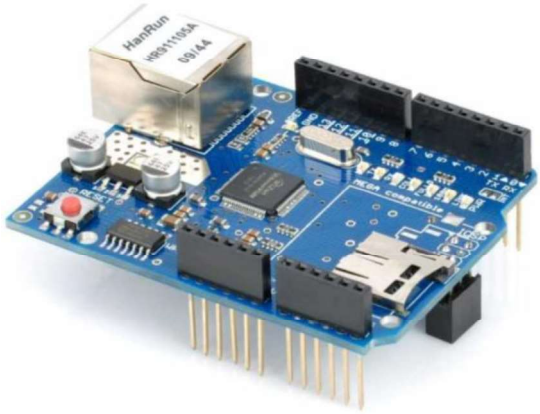

Figura 1.8. Ethernet Shield V1 [21].

El Ethernet Shield V1 permite establecer una conexión de la placa Arduino a Internet a través de una Red Ethernet por medio de un conector RJ45. El corazón del Ethernet Shield V1 es el chip Ethernet Wiznet W5100. El chip Wiznet W5100 proporciona una pila de protocolos como TCP y UDP; este admite hasta cuatro conexiones de socket simultáneas y utiliza la biblioteca de Ethernet para escribir Sketches que permiten el manejo de datos a través de una Red Ethernet [22].

Arduino se comunica tanto con el chip W5100 como con la tarjeta SD utilizando el bus SPI (a través de los pines ICSP). Esto está en los pines digitales 10, 11, 12 y 13 en el Uno y los pines 50, 51, 52 y 53 en el Mega. En ambas tarjetas, el pin 10 se usa para seleccionar el chip W5100 y el pin 4 para la tarjeta SD. Estos pines no se pueden utilizar para entradas y salidas [22].

El Ethernet Shield V1 contiene una serie de LED informativos [23]:

- **PWR:** indica que la placa y el escudo están energizados.
- **LINK:** indica la presencia de un enlace de red y parpadea cuando el Shield transmite o recibe datos.
- **100M:** indica la presencia de una conexión de red de 100 Mb/s (en lugar de 10 Mb/s).
- **FULLD:** indica que la conexión de red es full dúplex.
- **COLL:** parpadea cuando se detectan colisiones en la red.

- **RX:** parpadea cuando el Shield recibe datos.
- **TX:** parpadea cuando el Shield envía datos.

1.3.7 ESTÁNDAR ETHERNET

Ethernet es una tecnología que entrelaza computadoras y dispositivos de un área local (LAN) por medio de diferentes cables conectados entre sí. Esta tecnología crea una red de intercambio de información a través de las tramas de datos.

En la Tabla 1.2 se indica la comparación entre los diferentes estándares de Ethernet.

Tabla 1.2. Tabla comparativa entre diferentes estándares de Ethernet [24].

Estándar de Ethernet	Denominación	Velocidad de Datos	Tecnología de cables	Año de Publicación
802.3	10Base5	10 MB/s	Cable Coaxial	1983
802.3a	10Base2	10 MB/s	Cable Coaxial	1988
802.3i	10Base-T	10 MB/s	Cable de par trenzado	1990
802.3j	10Base-FL	10 MB/s	Cable de fibra óptica	1992
802.3u	100Base-TX 100Base-FX 100Base-SX	100 MB/s	Cable de par trenzado, cable de fibra óptica	1995
802.3z	1000Base-SX 1000Base-LX	1 GB/s	Cable de fibra óptica	1998
802.3ab	1000Base-T	1 GB/s	Cable de par trenzado	1999
802.3ae	10GBase-SR 10GBase-SW 10GBase-LR 10GBase-LW 10GBase-ER 10GBase-EW 10GBase-LX4	10 GB/s	Cable de fibra óptica	2002
802.3an	10GBase-T	10 GB/s	Cable de par trenzado	2006

1.3.8 SENSORES ARDUINO

En la Figura 1.9 se indica la imagen de un sensor ultrasónico a distancia.

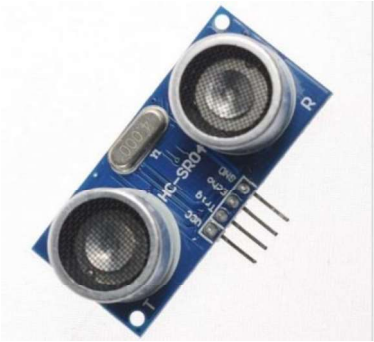

Figura 1.9. Sensor Ultrasónico de Distancia [25].

"Son dispositivos periféricos conectados de forma directa (cable o inalámbricamente) al Arduino desempeñan la función de recolectar datos del exterior. Los sensores son capaces de detectar magnitudes físicas y químicas, llamadas variables de instrumentación, y transformarlas a variables eléctricas [25]." Estas variables pueden ser: temperatura, humedad, presión, altitud, posicionamiento espacial, luminiscencia, distancia, aceleración, etc. Los sensores siempre van conectados a las entradas del Arduino.

Una clasificación de los sensores es según la función de los datos en su salida:

- Analógicos.
- Digitales.
- Comunicación por Bus.

1.3.9 COMUNICACIÓN DE BUS I2C

En la Figura 1.10 se indica el logotipo de la comunicación I2C.

Figura 1.10. Logotipo I2C [26].

“I2C es un protocolo síncrono para comunicación serial utilizando una interfaz de 2 líneas de baja velocidad. Desarrollado originalmente por Phillips a principios de los años 80 para la comunicación interna de los diferentes dispositivos electrónicos de la compañía [27].” I2C es una abreviatura de "Inter-Integrated Circuit". También se llama "IIC" o 'I cuadrado C'.

Originalmente, el bus I2C tenía una velocidad máxima de 100 KHz. La mayoría de las aplicaciones comunes todavía utilizan esta velocidad, ya que es suficiente para transferir datos desde los sensores.

I2C tiene algunos modos de mayor velocidad, pero no todos los dispositivos I2C soportan estos modos:

- **Modo rápido:** tiene una velocidad de reloj máxima de 400 KHz.
- **Modo de alta velocidad:** frecuencia máxima de reloj de 3.4 MHz
- **Modo ultra rápido:** frecuencia de reloj máxima de 5 MHz

En un bus I2C es el maestro el que determina la velocidad del reloj.

1.3.10 CONEXIÓN Y FUNCIONAMIENTO BUS I2C

En la Figura 1.11 se indican las líneas de conexión maestro – esclavo en I2C.

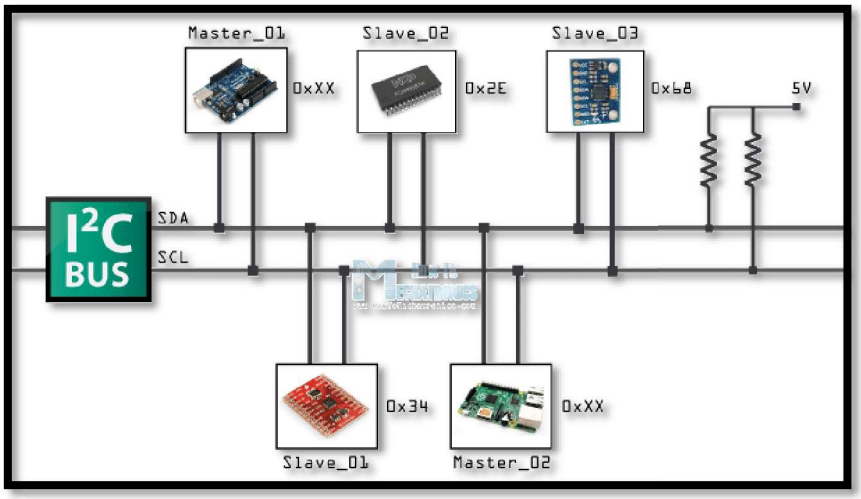

Figura 1.11. Líneas de Conexión Maestro – Esclavo I2C [28].

El bus I2C tiene dos línea de señal, junto con una alimentación y una conexión a tierra. Las dos líneas de señal son las siguientes:

- **SDA:** Esta es la línea de datos bidireccional.
- **SCL:** Esta es la señal del reloj.

Hay dos resistencias pull-up conectadas a cada línea de señal que llevan el bus a la tensión de alimentación cuando está inactivo. Cada dispositivo I2C tiene una etiqueta preestablecida o una dirección de dispositivo única para que el maestro pueda elegir con qué dispositivos se comunicara.

Existen dos tipos de dispositivos que se pueden conectar al bus I2C: maestros y esclavos.

El dispositivo maestro controla el bus y suministra la señal del reloj. Solicita datos de los esclavos individualmente. Puede haber más de un dispositivo maestro en el bus, pero solo uno puede ser el maestro activo en un momento dado. Los dispositivos maestros no tienen una dirección asignada a ellos.

Los dispositivos esclavos tienen una etiqueta, y esta etiqueta debe ser única en el bus. Utilizan un esquema de direccionamiento de 7 bits, por lo que hasta 128 esclavos pueden estar en un bus I2C.

Las conexiones SDA y SCL para I2C son diferentes entre los diferentes modelos Arduino como se muestra en la Tabla 1.3.

En la Tabla 1.3 se indican los pines de conexión I2C en los diferentes modelos de Arduino.

Tabla 1.3. Tabla pines de conexión I2C en los diferentes modelos de Arduino [29].

Placa Arduino o Chip	**SDA**	**SCL**
Uno	A4	A5
Mega	20	21
Nano	A4	A5
Pro Mini	A4	A5
Leonardo	2	3
Due (2 puertos I2C)	20+SDA1	20+SCL1
ATTiny85 & ATTiny45	5	7
Atmega328P	27	28

Para realizar la comunicación sobre la línea SDA, el bus I2C envía o recibe la trama de la siguiente manera: los esclavos dan a conocer su etiqueta al dispositivo maestro conectado en el bus I2C. “El maestro inicia la comunicación indicando la dirección del esclavo con el que desea hablar enviando en los siete primeros bits la etiqueta del esclavo y en el último bit si es una lectura (recibir) o escritura (enviar) [27].” Seguido de un bit de validación que puede ser un bit de retorno, en la siguiente trama vienen los datos enviados o recibidos del dispositivo esclavo seguidos de un bit de retorno como se ve en la Figura 1.12. Después de enviar o recibir todos los datos el maestro termina la comunicación para saltar a otro dispositivo esclavo.

En la Figura 1.12 se indica el funcionamiento del bus I2C.

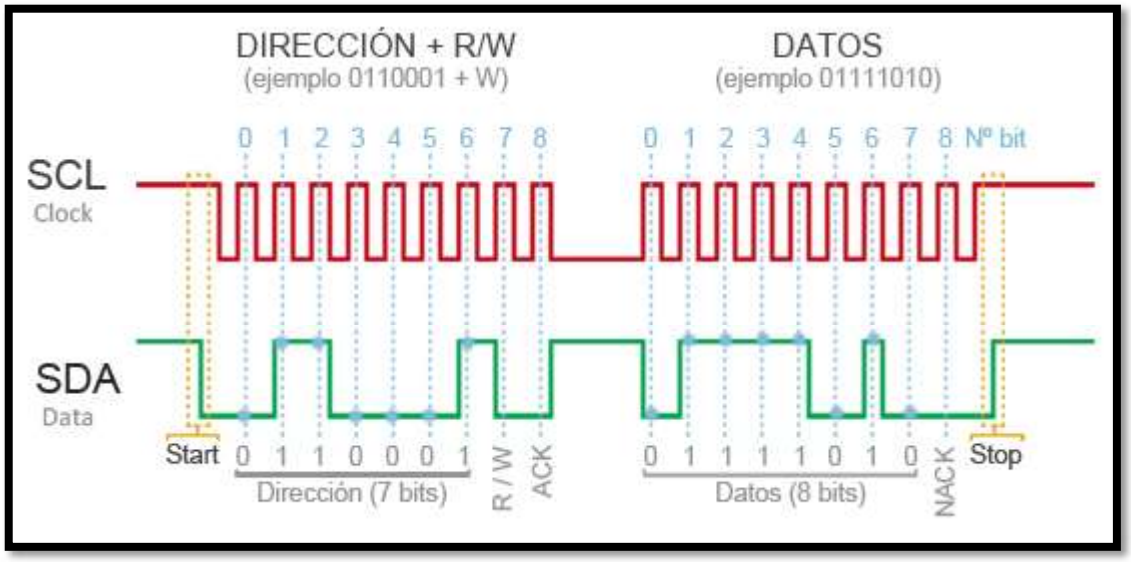

Figura 1.12. Funcionamiento bus I2C [30].

1.3.11 PROTOCOLO Y PUERTOS TCP

“Protocolo es el conjunto de normas y reglas siendo esta una convención que controla o permite la conexión, comunicación y transferencia de datos entre 2 o más dispositivos finales de red [31].“ Existen varios protocolos de capa de aplicación como HTTP, UDP, IP, RTP, TCP entre otros. Todos los protocolos de red son fundamentales para un funcionamiento adecuado de la red, sin embargo, según los requerimientos de la Red LAN que se diseñe se utilizará protocolos específicos.

El Protocolo de Control de Transmisión conocido como protocolo TCP (Transmission Control Protocol) realiza una comunicación orientada a conexión, es decir que mantiene una conexión estable dándole confiabilidad al usuario. Esto lo hace mediante mensajes a 3 vías para establecer o cerrar una conexión TCP como lo explica la Figura 1.13. Lo hace de la siguiente manera [32]:

1. El primer dispositivo pregunta al segundo dispositivo si se quiere sincronizar enviando un mensaje SYN (SYN = "sincronizar").
2. El segundo dispositivo responde al primer dispositivo con un mensaje de confirmación enviando un mensaje ACK (ACK = "asentimiento") Y además envía otro mensaje SYN.
3. El primer dispositivo responde con otro mensaje ACK.

Con este proceso los 2 dispositivos han establecido una conexión. Ahora cada paquete de datos que envíen lo harán de la siguiente manera:

1. El primer dispositivo enviará un paquete de datos y lo etiquetará con un número de secuencia.
2. El segundo dispositivo avisará al primer dispositivo haber recibido el paquete de datos con un ACK.

En el caso que el primer dispositivo no haya recibido un ACK como respuesta de recepción del paquete de datos, este esperará un determinado tiempo antes de reenviar dicho paquete de datos. Cada ACK de respuesta deberá llevar etiquetado el número de secuencia para que el primer dispositivo sepa qué paquetes llegaron correctamente a su destino. Para cerrar la conexión, se procederá de la siguiente manera:

1. El primer dispositivo inicia el cierre de la conexión con el segundo dispositivo enviando un mensaje FIN (FIN = terminar).
2. El segundo dispositivo responde al primer dispositivo con un mensaje de confirmación enviando un mensaje ACK y además envía otro mensaje FIN.
3. El primer dispositivo responde con otro mensaje ACK y la conexión está cerrada.

En la Figura 1.13 se indica el intercambio de mensajes para el inicio y cierre de conexiones en el protocolo TCP.

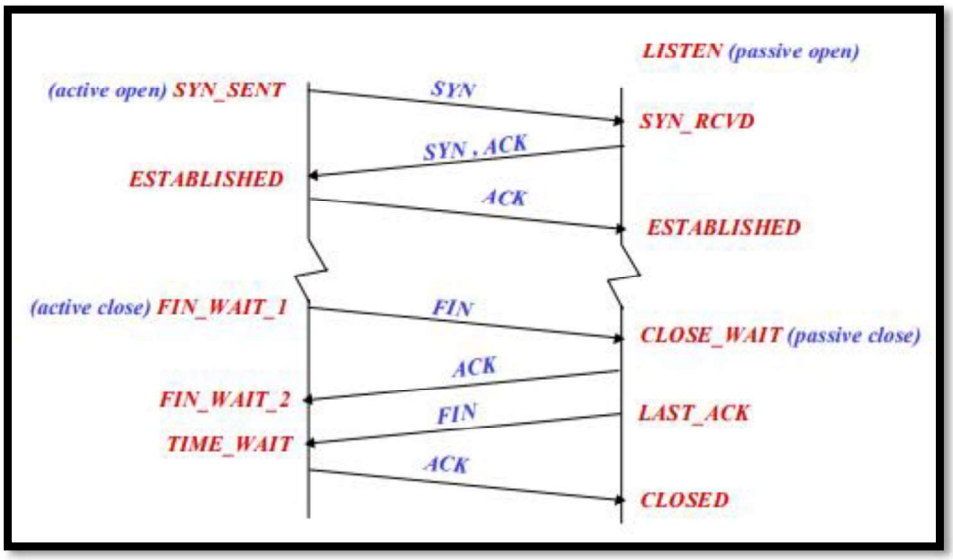

Figura 1.13. Intercambio de mensajes para el inicio y cierre de conexión [33].

Los puertos TCP mejor conocidos como “puertos” sirven para identificar la aplicaciones tanto del receptor como del transmisor. Existen 65536 puertos posibles, los cuales se clasifican en 3 grupos [34]:

1. **Puertos bien conocidos**: van desde el 0 hasta el 1023. Son asignados por la IANA (Internet Assigned Numbers Authority) y son usados normalmente en servicios estandarizados a nivel global. Por ejemplo: SSH (22), Telnet (23), SMTP (25), TFTP (69), HTTP (80) entre otros.
2. **Puertos registrados**: van desde el 1024 hasta el 49151.
3. **Puertos privados**: van desde el 49152 hasta 65535.

1.3.12 CHIP WIZNET W5100

El Chip Wiznet W5100 es un controlador de Ethernet fabricado para aplicaciones y/o sistemas embebidos, este controlador es empleado con un procesador como Arduino para implementar comunicación Ethernet para conectarse a internet o a un red LAN. El chip W5100 está diseñado para facilitar la implementación de conectividad a internet sin necesidad de un sistema operativo, incluyendo una pila de TCP/IP por hardware y buffer

interno de 16Kbytes para Tx/Rx. Esto permite liberar de estas tareas al procesador, siendo una de sus principales ventajas frente a otros controladores de Ethernet como el ENC28J60.

El chip W5100 se conecta mediante SPI, por lo que es muy sencilla la conexión con la mayoría de los procesadores. Algunos módulos incorporan un lector de tarjeta SD, donde se puede guardar ficheros (html, txt, jpg, png) para trabajar cuando actúe como servidor. Maneja velocidades de 10/100 Mbits/s, soportando modos Full-Duplex y Half-Duplex con detección y corrección automática de la polaridad. Cumple con las especificaciones IEEE 802.3 10BASE-T y 802.3u 100BASE-TX. La pila TCP/IP soporta conexiones TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE, y hasta 4 conexiones simultáneas [35].

El W5100 es un chip ampliamente empleado en dispositivos conectados en aplicaciones industriales, domésticas de domótica e IOT. Por ejemplo, entre otros muchos, pantallas y televisores inteligentes, relés activados por internet, impresoras, cámaras IP o dispositivos de almacenamiento en red. Se puede emplear un módulo Ethernet Shield con W5100 para enviar o recibir información desde internet o una red LAN interna para realizar la lectura de un sensor, o para activar, controlar o desactivar cualquier dispositivo.

En la Figura 1.14 se indica el diagrama en bloques del chip W5100.

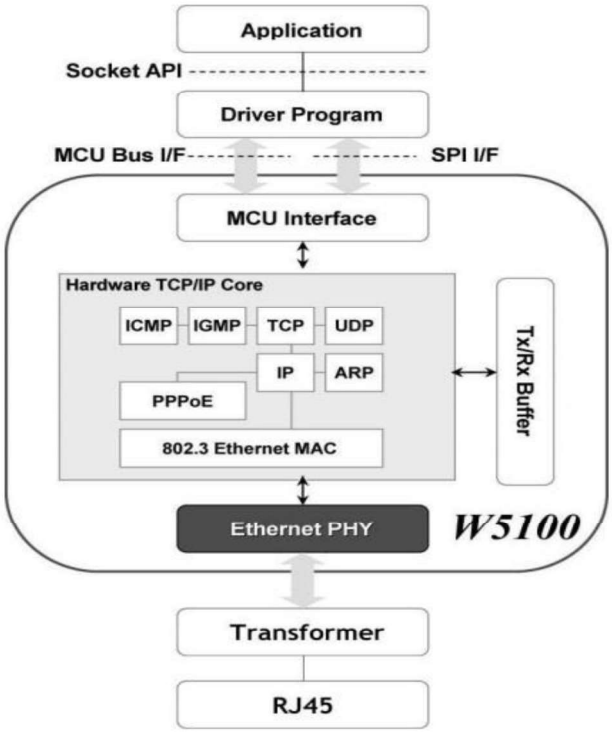

Figura 1.144. Diagrama en bloques del chip W5100 [35].

2. METODOLOGÍA

Este capítulo describe el desarrollo e implementación de una librería para comunicación con el estándar Ethernet entre la plataforma Arduino y una aplicación en Visual Studio orientada a IOT utilizando sockets, lo que constituye el objetivo principal de este proyecto integrador. También se describirá la forma en la cual se envía y recibe la información de los sensores, que serán interpretados y discutidos en el siguiente capítulo.

Se explica brevemente el comportamiento de los diferentes dispositivos a utilizar, sea hardware o software, describiendo el rol que cumplen y la función que realizarán dentro del proyecto. Así como la estructura que deberá tener dicho proyecto para que funcione correctamente. De igual manera, algunos conceptos serán tomados en cuenta para el entendimiento del lector.

Una vez abordado el uso de los dispositivos del proyecto, se pasa a la parte del desarrollo para conseguir la comunicación entre la plataforma Arduino como dispositivo IOT y el Aplicativo en Visual Studio. También, se presentan los pasos o indicadores que determinarán que está funcionando de la manera esperada el proyecto.

Finalmente, se indica la implementación que será llevada a cabo para obtener la comunicación Ethernet entre la plataforma Arduino y la aplicación en Visual Studio dentro de un ambiente IOT.

2.1. DESCRIPCIÓN GENERAL

Este proyecto de titulación contempla el desarrollo de una librería en Arduino que permita transmitir datos desde los sensores a través del Arduino (Cliente) hacia una aplicación en Visual Studio (Servidor) por comunicación Ethernet mediante el módulo Ethernet Shield, siendo el programador quien configure sobre qué socket se envían estos datos y cómo son evaluados posteriormente.

Para un mejor entendimiento será considerado el proyecto por procesos. El primer proceso será la adquisición de los datos desde los sensores hasta el almacenamiento en “la memoria SRAM [36] por medio de variables locales.” El segundo proceso consiste en enviar los datos por vía Ethernet mediante sockets desde el Arduino hasta el aplicativo de Visual Studio. Finalmente, el tercer proceso es la adquisición y presentación de dichos datos sobre una aplicación en Visual Studio.

En el primer proceso, los sensores capturan los datos de presión, temperatura, altitud y posición espacial provenientes del medio donde se encuentren. Estos datos se enviarán por el bus I2C [27] hacia el Arduino que guarda en la memoria SRAM por medio de variables locales. Se debe considerar que la memoria del Arduino UNO está compuesta de la siguiente manera [36]:

- **Flash:** 32 Kbytes (de los cuales 0.5 Kbytes son usados en el arranque)
- **SRAM:** 2 Kbytes
- **EEPROM:** 1 Kbyte

En el segundo proceso, las variables locales serán cargadas en el código de programación del Sketch desarrollado en Arduino como cliente para enviar los datos mediante comunicación Ethernet hacia el aplicativo de Visual Studio como servidor. La nueva librería Ethernet será cargada en el Sketch para que funcionen adecuadamente los sockets durante todo el tiempo que se lleve la transmisión.

Al final en el tercer proceso, el aplicativo de Visual Studio receptará los datos provenientes del Arduino a través de un socket de comunicación que el programador previamente configuró. El aplicativo visualizara los datos en una pantalla de presentación para un fácil entendimiento de la información. Además, el aplicativo almacenara lo que se muestra en la pantalla dentro de un archivo para que posteriormente se puedan analizar los datos de la manera que se necesite.

2.2. ESTRUCTURA

Se desarrolla el diseño que contiene los sensores, el Arduino y el aplicativo (computador) de una manera sencilla ya que el corazón de este proyecto es la creación de una nueva librería Arduino. Esta nueva librería mantendrá varias sesiones simultáneas del Arduino como cliente y al aplicativo como servidor, dispondrá de los sensores de forma individual sobre diferentes Sockets para generar varias sesiones mejorando la gestión en cada uno. En el siguiente subcapítulo se describe cada recurso y elemento que conforman el proyecto integrador, así como su funcionamiento.

"La comunicación punto a punto es un método en el que dos dispositivos de comunicación se conectan entre si formando un enlace entre ellos estableciendo una conexión sin tener un alojamiento en un tercer dispositivo [37]." La comunicación punto a

punto se basa en el establecimiento de una conexión entre un transmisor y un receptor donde solo los dos mantienen un cruce de información. Es decir, todo lo que envía el transmisor lo recibe un único receptor. El presente proyecto se realizará mediante el uso de Sockets para llevar una conexión única de la información de cada uno de los sensores siendo el protocolo TCP la mejor opción al utilizar una comunicación punto a punto. Hay que considerar que este tipo de conexión es una comunicación cliente – servidor y se recomienda seguir el diagrama de flujo para el proceso del modo cliente como se muestra en la Figura 2.1 que nos proporciona el datasheet del chip Wiznet W5100 [35] al desarrollar el Sketch para la aplicación en Arduino.

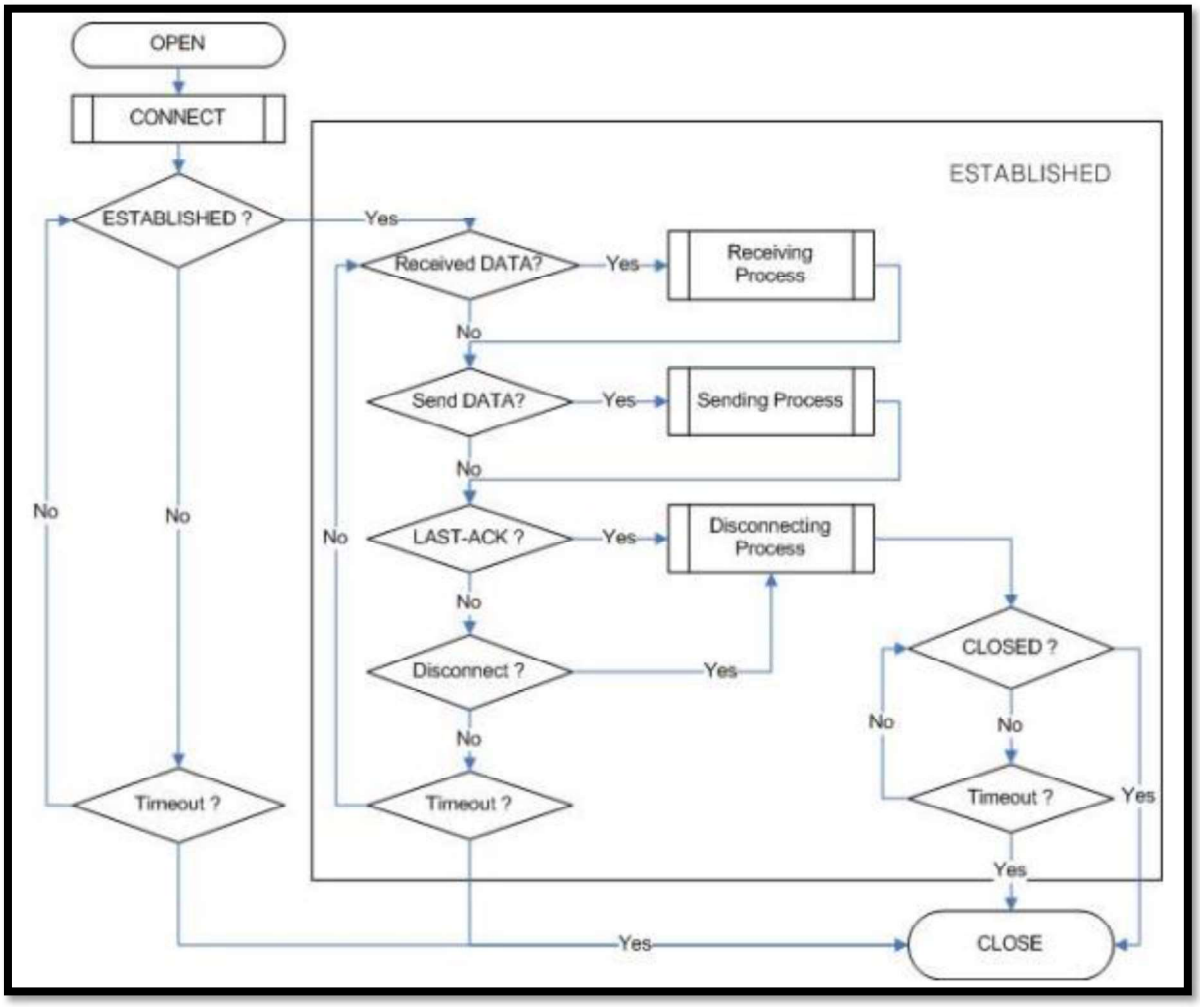

Figura 2.1. Proceso del modo Cliente *[35]*.

Se desarrolla una estructura sencilla y básica como se muestra en la Figura 2.2 para verificar el correcto funcionamiento de la nueva librería Ethernet. Serán capturados datos del ambiente como presión, temperatura, altitud y posición espacial por medio de

sensores, información, que a su vez la enviarán a través de un bus I2C hacia el Arduino. El Arduino encapsulara esta información en una trama Ethernet con la ayuda del módulo Ethernet Shield. El módulo Ethernet Shield enviará las tramas Ethernet por la Red LAN hacia un aplicativo en Visual Studio, de igual manera el aplicativo mantendrá una forma sencilla y básica, en el siguiente subcapítulo se explicarán de manera global los recursos y elementos que conforman la Red LAN de este proyecto integrador.

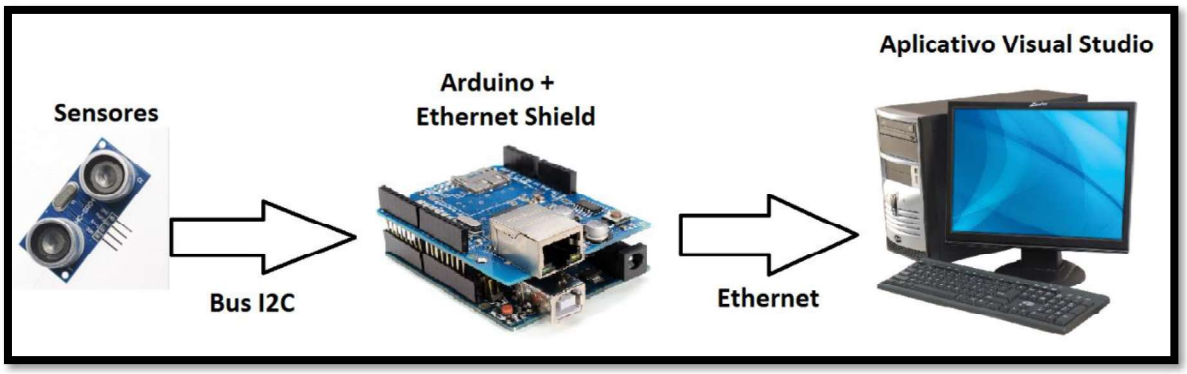

Figura 2.2. Red LAN.

2.3. IMPLEMENTACIÓN

Al tener claro el panorama del proyecto hay que considerar que en Arduino las librerías cumplen una parte primordial en su funcionamiento. El programador debe considerar qué requiere dentro de su proyecto, pues hay una infinidad de librerías a las cuales tiene acceso, sin embargo, no siempre las librerías ofrecidas pueden resolver todos los proyectos presentados debido a las limitaciones que tienen.

Para el presente proyecto se debe considerar cuales son los requerimientos y necesidades que debe abordar:

- Enviar datos del ambiente o medio donde se encuentren a través de sensores al Arduino por medio del Bus I2C.
- Procesar los datos dentro de una trama Ethernet para su posterior envío.
- Lograr una comunicación cliente – servidor entre Arduino y una aplicación en Visual Studio por comunicación Ethernet utilizando Sockets.
- Realizar enlaces multisesión de cada sensor conectado al Arduino a diferentes aplicaciones Visual Studio por comunicación Ethernet utilizando Sockets.

"Enviar datos del ambiente o medio donde se encuentren a través de sensores al Arduino por medio del Bus I2C". Los mismos fabricantes de los sensores crean sus propias librerías para una apropiada utilización de estos y a su vez muchos programadores han creado librerías para aprovechar de mejor manera los sensores según sus requerimientos. Cabe tener en cuenta que los sensores para bus I2C son totalmente diferentes a los sensores estándar, por lo que se debe conseguir sensores y librerías adecuados en el mercado. Dentro de este proyecto se utilizará sensores I2C con sus respectivas librerías para conseguir una correcta comunicación punto a punto.

"Procesar los datos dentro de una trama Ethernet para su posterior envío". La librería Ethernet actual realiza un correcto entramado de la información donde el programador solo se encarga de enviarla a su respectivo destino. Las líneas de programación sobre el envío de tramas no serán alteradas ni modificadas se las mantendrá de la misma manera en la nueva librería ya que en la actualidad funcionan correctamente.

"Lograr una comunicación cliente – servidor entre Arduino y una aplicación en Visual Studio por comunicación Ethernet utilizando Sockets". Un Socket es la asociación de una dirección IP a un puerto. La librería Ethernet actual es capaz de mantener una comunicación cliente – servidor abierta debido a que el único puerto disponible es el puerto 80 que es del servicio de HTTP. Siendo este un limitante para conseguir una comunicación Ethernet en diferentes Sockets según requiera el programador.

"Realizar enlaces multisesión de cada sensor conectado al Arduino a diferentes aplicaciones Visual Studio por comunicación Ethernet utilizando Sockets". La librería Ethernet actual tiene la capacidad de mantener una sesión abierta y para poder usar otra sesión necesita cerrar dicha sesión y abrir una nueva, esto no es una comunicación multisesión. Por otro lado, el módulo Ethernet Shield es capaz de mantener 4 sesiones abiertas simultáneas utilizando Sockets, pero su librería actual tiene limitaciones las cuales deben ser resueltas.

2.3.1 NUEVA LIBRERÍA ETHERNET

La librería Ethernet que se analiza para el desarrollo del presente trabajo de titulación es la versión 1.1.2 y se encontró que la multisesión no se ejecutaba de una manera apropiada. Su multisesión utilizando Sockets permite que para poder utilizar una nueva

conexión se deba cerrar la actual, lo cual es un uso deficiente de las características del módulo Ethernet Shield.

El código de la librería Ethernet maneja una matriz para almacenar los puertos que el programador configura. Los puertos que el programador configura se almacenan en ranuras de la matriz donde el chip Wiznet W5100 hará su búsqueda. El código de la librería Ethernet solo busca según el número de puerto. Entonces, si se configuró el puerto 80 para un primer socket y se configura el mismo puerto 80 para un segundo socket, el chip Wiznet W5100 seguirá leyendo que el puerto 80 solo fue asignado al primer socket descartando todo lo demás.

Este inconveniente no permite desarrollar todo el potencial que tiene Arduino en tecnología Ethernet siendo un limitante para buscar otras opciones al momento de requerir una comunicación Ethernet de los periféricos de Arduino. Se ha resuelto corregir este inconveniente en una nueva librería Ethernet agregando un almacenamiento secundario para la búsqueda de diferentes puertos a la vez de poder identificar los sockets con un número de puerto dado.

Previamente se necesita que este instalado el IDE de Arduino 1.8.7. para poder tener acceso a las librerías de Arduino para su posterior utilización, suponiendo que no se tenga instalado el IDE de Arduino, el proceso es el siguiente:

1. Descargar el IDE de Arduino de [38].
2. Buscar la versión 1.8.7 y escoger la opción *"Windows Installer".*
3. Instalar el IDE de Arduino 1.8.7.

Con el IDE de Arduino 1.8.7. instalado, localizar la carpeta que contiene la librería Ethernet en la dirección *"C:\Program Files (x86)\Arduino\libraries"* o en la ubicación donde se haya instalado el IDE de Arduino y eliminar la carpeta *"Ethernet"* con todo su contenido para instalar la librería Ethernet 1.1.2. lo cual se hará de la siguiente manera:

1. Descargar la librería Ethernet 1.1.2. de [39].
2. Descomprimir el archivo Ethernet-1.1.2.zip en la dirección *"C:\Program Files (x86)\Arduino\libraries"* o en la ubicación donde se encuentran las librerías de Arduino.

En la Figura 2.3 se indica la carpeta que contiene la librería Ethernet 1.1.2.

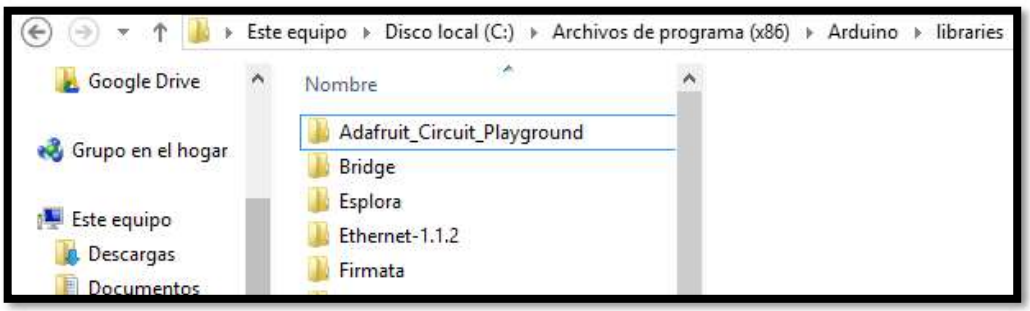

Figura 2.3. Librería Ethernet 1.1.2.

3. Cambiar el nombre de la carpeta *"Ethernet-1.1.2"* a *"Ethernet"*. Esto se hace para que el IDE de Arduino lo detecte como la librería original que fue borrada y no presente problemas.

4. Finalmente, se debe localizar la carpeta que contiene los archivos .cpp y .h de la librería Ethernet en la dirección "C:\Program Files (x86)\Arduino\libraries\Ethernet\src" para realizar las modificaciones de la librería Ethernet actual y crear una nueva librería Ethernet. Para modificar los archivos .cpp y .h están en lenguaje C, se utilizará cualquier compilador que soporte lenguaje C o a su vez utilizando un Bloc de Notas como el Worpad. Si se tiene problemas de restricción al querer modificar los archivos dentro de la carpeta Arduino, una alternativa es copiar los archivos .cpp y .h a otra carpeta para poder modificarlos allí y después volverlos a pegar en la carpeta original. Otra alternativa es cambiar los permisos de la carpeta Arduino para poder modificar los archivos que esta contiene.

En la Figura 2.4 se indican los archivos .cpp y .h de la librería Ethernet 1.1.2.

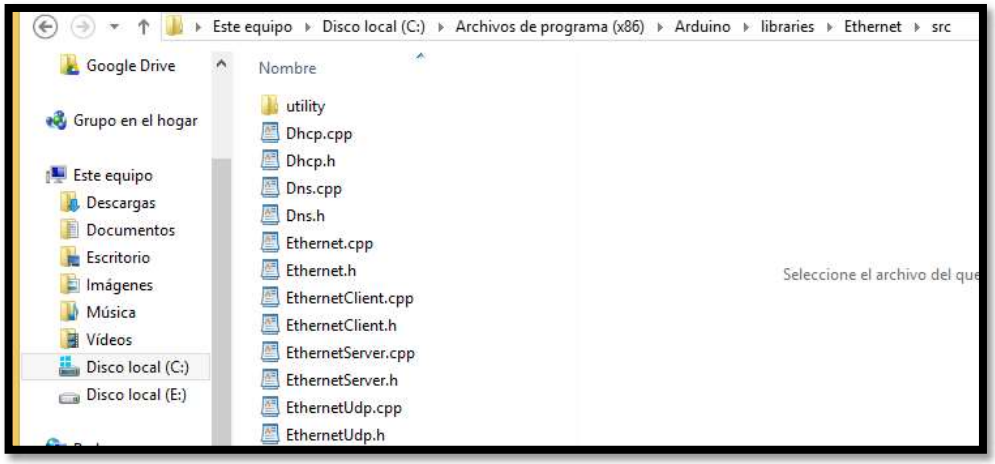

Figura 2.4. Archivos .cpp y .h de la librería Ethernet 1.1.2.

Se modifican los siguientes archivos:

1. **Ethernet.h:** el archivo actual solo busca en el puerto del socket que está escuchando como servidor. Es necesario, agregar una nueva matriz para almacenar el puerto asociado a un socket usado como cliente con la siguiente línea de código:

```
static uint16_t _client_port[MAX_SOCK_NUM]
```

   La instrucción "***static uint16_t"*** define que será usada una variable estática entera de 16 bits sin signo, por otro lado "***_client_port[]"*** es el nombre que se le asigna a la variable estática y "***MAX_SOCK_NUM"*** es una variable del programa que se llama para asignarle el tamaño que tendrá la nueva instrucción. En el Anexo A se presentará el archivo Ethernet.h completo.

2. **Ethernet.cpp:** agregar la declaración de la nueva matriz para el archivo Ethernet.h en el archivo Ethernet.cpp con la siguiente línea de código:

```
uint16_t EthernetClass::_client_port[MAX_SOCK_NUM] = { 0, 0, 0, 0 }
```

   La instrucción "***uint16_t"*** define que será usada una variable entera de 16 bits sin signo, "***EthernetClass::_client_port[]"*** es para asignar el uso de la variable *"_client_port[]"* sobre la estructura *"EthernetClass"*, "***MAX_SOCK_NUM"*** es una variable del programa que se llama para asignarle el tamaño que tendrá la nueva instrucción y "***{0,0,0,0}"*** es agregado para poner en cero todos los valores de la matriz. En el Anexo A se presentará el archivo Ethernet.cpp completo.

3. **EthernetClient.h:** agregar nuevas funciones y declarar una nueva variable local que buscará el puerto y la dirección IP de destino del cliente por medio del uso de un puntero. Las nuevas funciones estarán en la sección privada al final del código.

   En la Figura 2.5 se indica una sección del archivo EthernetClient.h.

```
uint16_t _dstport;
//
uint8_t *getRemoteIP(uint8_t remoteIP[]);
//
uint16_t getRemotePort();
```

Figura 2.5. Sección archivo EthernetClient.h

   Existen varias instrucciones que fueron repetidas anteriormente que solo llevan un cambio de nombre de la variable por lo cual no se explicaran, sin embargo, las nuevas instrucciones en el programa se las explicará. La instrucción "***uint8_t"*** define que será usada una variable entera de 8 bits sin signo, "****getRemoteIP()"*** asigna a la variable *"getRemote()"* como puntero y "***uint8_t remoteIP[]"*** es una variable entera de 8 bits sin signo del programa que se llama para asignarle el tamaño que tendrá la nueva instrucción. En el Anexo A se presentará el archivo EthernetClient.h completo.

4. **EthernetClient.cpp:** cuando el cliente se desconecta del servidor, este reinicia el _server_port a cero. Se debe procurar que el nuevo _client_port este vacío. Se lo realizará a través de la línea de código:

```
EthernetClass::_client_port[_sock] = 0
```

   Además, se debe agregar 2 funciones al final del código para la búsqueda del puerto y la dirección IP de destino del cliente.

   En la Figura 2.6 se indica una sección del archivo EthernetClient.cpp.

```
uint8_t *EthernetClient::getRemoteIP(uint8_t remoteIP[])
//
{
  W5100.readSnDIPR(_sock, remoteIP);
  return remoteIP;
}
```

Figura 2.6. Sección 1 archivo EthernetClient.cpp

La instrucción ***"uint8_t"*** define que será usada una variable entera de 8 bits sin signo, ***"*EthernetClient::getRemoteIP()"*** es para asignar el uso de la variable *"getRemoteIP()"* sobre la estructura *"EthernetClass"* que esta como puntero y ***"uint8_t remoteIP[]"*** es una variable entera de 8 bits sin signo del programa que se llama para asignarle el tamaño que tendrá la instrucción. La instrucción ***"W5100.readSnDIPR()"*** el chip W5100 hace una lectura del registro "SnDIPR" de los valores que tiene en ese momento, y ***"_sock, remoteIP"*** son para el valor del socket (_sock) y la dirección IP del destino (remoteIP) actuales. Finalmente, ***"return remoteIP"*** retorna el valor de "remoteIP" y sale de la función.

En la Figura 2.7 se indica una sección del archivo EthernetClient.cpp.

```
uint16_t EthernetClient::getRemotePort()
//
{
  return W5100.readSnDPORT(_sock);
}
```

Figura 2.7. Sección 2 archivo EthernetClient.cpp

La estructura de esta función es idéntica a la anterior considerando las instrucciones descritas anteriormente. La instrucción ***"return W5100.readSnDPORT()"*** retorna el valor que lee el chip W5100 del registro "SnDPORT" de su valor actual, y ***"_sock"*** es el valor del socket actual y finalmente sale de la función. En el Anexo A se presentará el archivo EthernetClient.cpp completo.

5. **EthernetServer.cpp:** este código debe modificarse para que cuando verifique una conexión, compruebe tanto el puerto existente del servidor (el que ingresa el programador) como el nuevo puerto del cliente (el que se asigna arbitrariamente). Si nunca se ha realizado la conexión, se inicializará el nuevo puerto del cliente

correctamente, en el Anexo A se presentará las modificaciones al archivo EthernetServer.cpp completo.

2.3.2 DISEÑO DE LA RED LAN: 1 SERVIDOR – 2 CLIENTES

Teniendo la nueva librería Ethernet, ahora se debe diseñar la Red LAN que compruebe el funcionamiento de la conexión Ethernet en una comunicación cliente – servidor antes de realizar el diseño final que pone a prueba a la nueva librería Ethernet. El diseño es sencillo y básico, lo suficiente para comprobar la comunicación Ethernet utilizando Sockets.

2.3.2.1 Recursos y descripción de materiales

En esta sección se detallan los recursos para comprobar el funcionamiento de la conexión Ethernet utilizando la nueva librería Ethernet. Los recursos son los siguientes:

- 1 Arduino UNO
- 1 módulo Ethernet Shield
- 1 magnetómetro HMC5883L
- 1 sensor de presión barométrica BMP180
- 1 Protoboard
- Varios cables Jumpers Dupont Macho – Macho
- 1 cable Ethernet
- 1 cable USB tipo B
- 1 cargador USB
- 1 computador

El Arduino UNO y el módulo Ethernet Shield ya fueron descritos en el Capítulo 1. Se describirá brevemente el resto de los recursos que se utilizarán en el presente proyecto.

El magnetómetro HMC5883L tiene 3 materiales magneto resistivos dentro de los cuales están dispuestos en los ejes X, Y y Z que pueden leer los componentes del campo magnético presente, de esta manera, conociendo la dirección del campo magnético de la Tierra podemos calcular la orientación con respecto al norte magnético de la Tierra, esto siempre y cuando el sensor no esté expuesto a algún campo magnético externo o algún objeto metálico que altere el campo magnético de la Tierra [40]. El HMC5883L utiliza la comunicación I2C, de modo que conecte los pines *SCL* a A5 del Arduino y *SDA* a A4 del Arduino como se mostró anteriormente en la Tabla 1.3.

En la Figura 2.8 se indica la imagen del magnetómetro HMC5883L.

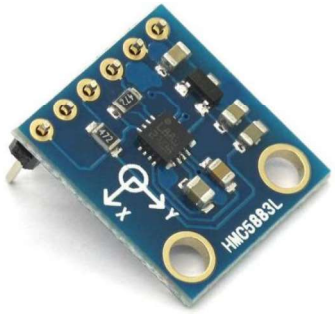

Figura 2.8. Magnetómetro HMC5883L [41].

En la Tabla 2.1 se indican las especificaciones técnicas del magnetómetro HMC5883L.

Tabla 2.1. Especificaciones Técnicas Magnetómetro HMC5883L [40].

Especificación Técnica	Valor
Voltaje de Operación	3.3 V - 5 V
Interfaz	I2C
Rango de medición	±1 – 8 Gauss
Resolución	5 mili-Gauss
Precisión	± 2 °
Velocidad de datos	160 Hz
Ejes Magnéticos	3

El sensor de presión barométrica BMP180 es un sensor piezorresistivo que detecta la presión. Los sensores piezorresistivos están formados por un material semiconductor (generalmente silicio) que cambia la resistencia cuando se aplica una fuerza mecánica como la presión atmosférica. El BMP180 mide tanto la presión como la temperatura, la

cual cambia la densidad de gases como el aire [42]. A temperaturas más altas, el aire no es tan denso y pesado, por lo que aplica menos presión sobre el sensor. Por otra parte, a temperaturas más bajas, el aire es más denso y pesa más, por lo que ejerce más presión sobre el sensor. El sensor utiliza mediciones de temperatura en tiempo real para compensar las lecturas de presión por cambios en la densidad del aire. El BMP180 utiliza la comunicación I2C, de modo que conecte los pines *SCL* a A5 del Arduino y *SDA* a A4 del Arduino como se mostró anteriormente en la Tabla 1.3.

En la Figura 2.9 se indica la imagen del sensor de presión barométrica BMP180.

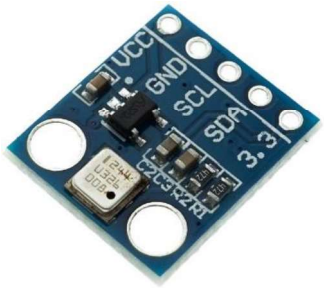

Figura 2.9. Sensor de presión barométrica BMP180 [43].

En la Tabla 2.2 se indican las especificaciones técnicas del sensor de presión barométrica BMP180.

Tabla 2.2. Especificaciones Técnicas Sensor de presión barométrica BMP180 [42].

Especificación Técnica	**Valor**
Voltaje de Operación	3.3 V - 5 V
Interfaz	I2C
Rango de presión	300 – 1100 hPa
Precisión de presión	0.03 hPa
Rango de temperatura	-40 °C a +80 °C
Precisión de temperatura	± 2 °C

“La protoboard es una placa de pruebas para electrónica que contiene numerosos orificios en los que es posible insertar cables y otros elementos electrónicos para montar circuitos provisionales [44].” Existen protoboards de diferente tamaño y fabricante, pero por lo general, todos estos soportan una carga de 3 amperios hasta 5 amperios.

En la Figura 2.10 se indica la imagen de un protoboard.

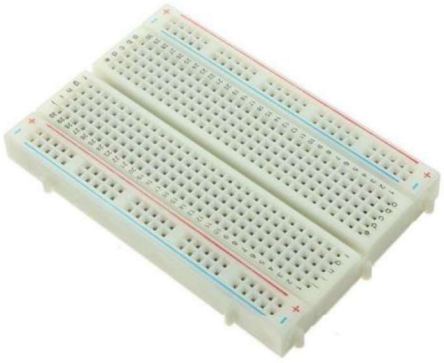

Figura 2.10. Protoboard [45].

“Cable Jumper Dupont también conocido como cable puente usado para interconectar componentes en un protoboard [46].”

En la Figura 2.11 se indica la imagen de cables Jumper Dupont Macho – Macho.

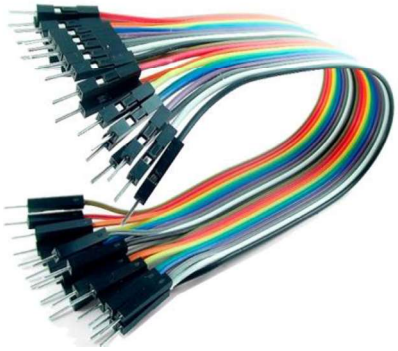

Figura 2.11. Cable Jumper Dupont Macho – Macho [47].

Un cable Ethernet es un cable de par trenzado que une 2 conectores RJ45 que se usa para redes cableadas. Los cables Ethernet conectan dispositivos como computadores, Routers y Switches dentro de una red LAN. Estos cables físicos están limitados por su longitud y durabilidad. Si un cable de red es demasiado largo o de baja calidad, no transmitirá una buena señal de red [48]. Estos límites son una de las razones por las que hay diferentes tipos de cables Ethernet como se detalla en la Tabla 2.3.

En la Figura 2.12 se indica la imagen de un cable Ethernet.

Figura 2.12. Cable Ethernet [49].

Tabla 2.3. Tipos de Cables Ethernet [50].

Categoría	Blindaje	Velocidad máxima de transmisión (a 100 metros)	Ancho de banda máximo
Cat 3	Sin Blindaje	10 Mbps	16 MHz
Cat 5	Sin Blindaje	100 Mbps	100 MHz
Cat 5e	Sin Blindaje	1 Gbps	100 MHz
Cat 6	Con y Sin Blindaje	1 Gbps	250 MHz
Cat 6a	Blindado	10 Gbps	500 MHz
Cat 7	Blindado	10 Gbps	600 MHz
Cat 7a	Blindado	10 Gbps	1 GHz

“El cable USB tipo B es un tipo de conector se ha empleado, principalmente, para conectar a periféricos como impresoras o escáneres. Aunque, en algunos periféricos, este conector se emplea solo para proporcionar alimentación [51].” Este tipo de conectores son compatibles con el estándar USB 3.0. En Arduino, este cable sirve para cargar los programas en su memoria a través del computador y también sirve como alimentación.

En la Figura 2.13 se indica la imagen de un cable USB tipo B.

Figura 2.13. Cable USB tipo B [52].

El cargador USB es compatible con la mayoría de los dispositivos que utilizan el estándar USB 3.0. proporcionando la alimentación necesaria para que estos funcionen, el Arduino es compatible y funciona correctamente. En la Tabla 2.4 se muestra las características generales del cargador USB.

En la Figura 2.14 se indica la imagen de un cargador USB.

Figura 2.14. Cargador USB [53].

Tabla 2.4. Características generales cargador USB [54].

Característica General	Valor
Voltaje de Entrada	100 – 240 Vac
Corriente de Salida	2 A
Voltaje de salida	5.3 V

2.3.2.2 Diagrama y diseño

En esta sección, se desarrolla y explica tanto el diagrama, como el diseño electrónico que ayude a comprobar el funcionamiento de la conexión Ethernet en una comunicación cliente – servidor utilizando Sockets.

Para poder diferenciar y apreciar de una manera dinámica el diagrama y el esquema electrónico, se ha designado un color de cable puente para cada conexión, haciendo más práctica su comprensión. En la Tabla 2.5 se muestra la asignación del color al cable puente de una conexión en particular.

Tabla 2.5. Color de cable puente por conexión.

Conexión	Color cable puente
SDA	Azul
SCL	Amarillo
VCC	Rojo
GND	Negro

Primero, se elabora el diagrama electrónico de la Figura 2.15 en base a todos los parámetros y condiciones vistos anteriormente tales como:

1. El pin SDA de cada uno de los sensores se conectará al pin SDA del Arduino, ver Tabla 1.3.
2. El pin SCL de cada uno de los sensores se conectará al pin SCL del Arduino, ver Tabla 1.3.
3. Los sensores deben ser alimentados con 5 V (VCC) y a su vez conectados a tierra (GND).

En la Figura 2.15 se indica la imagen del diagrama electrónico para la red de sensores.

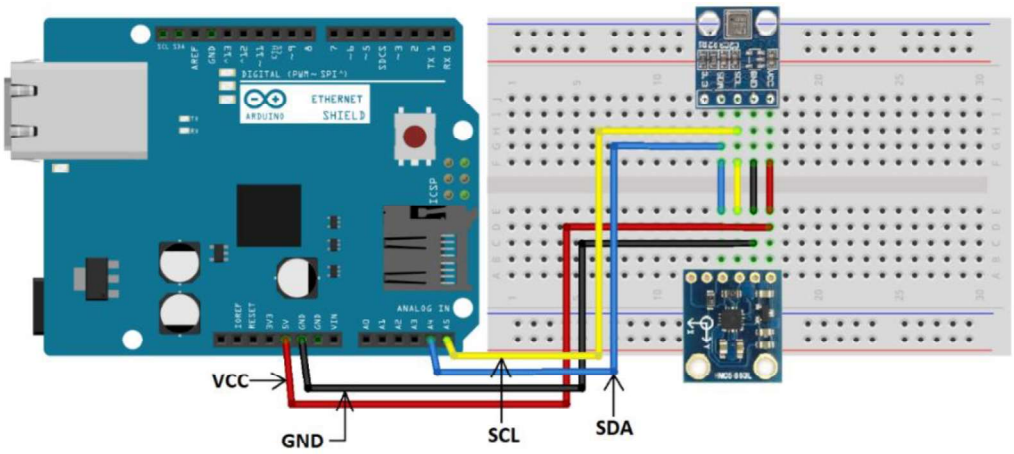

Figura 2.15. Diagrama Electrónico.

Después, se elabora el esquema electrónico como se muestra en la Figura 2.16 en base al diagrama electrónico.

En la Figura 2.16 se indica la imagen del esquema electrónico para la red de sensores.

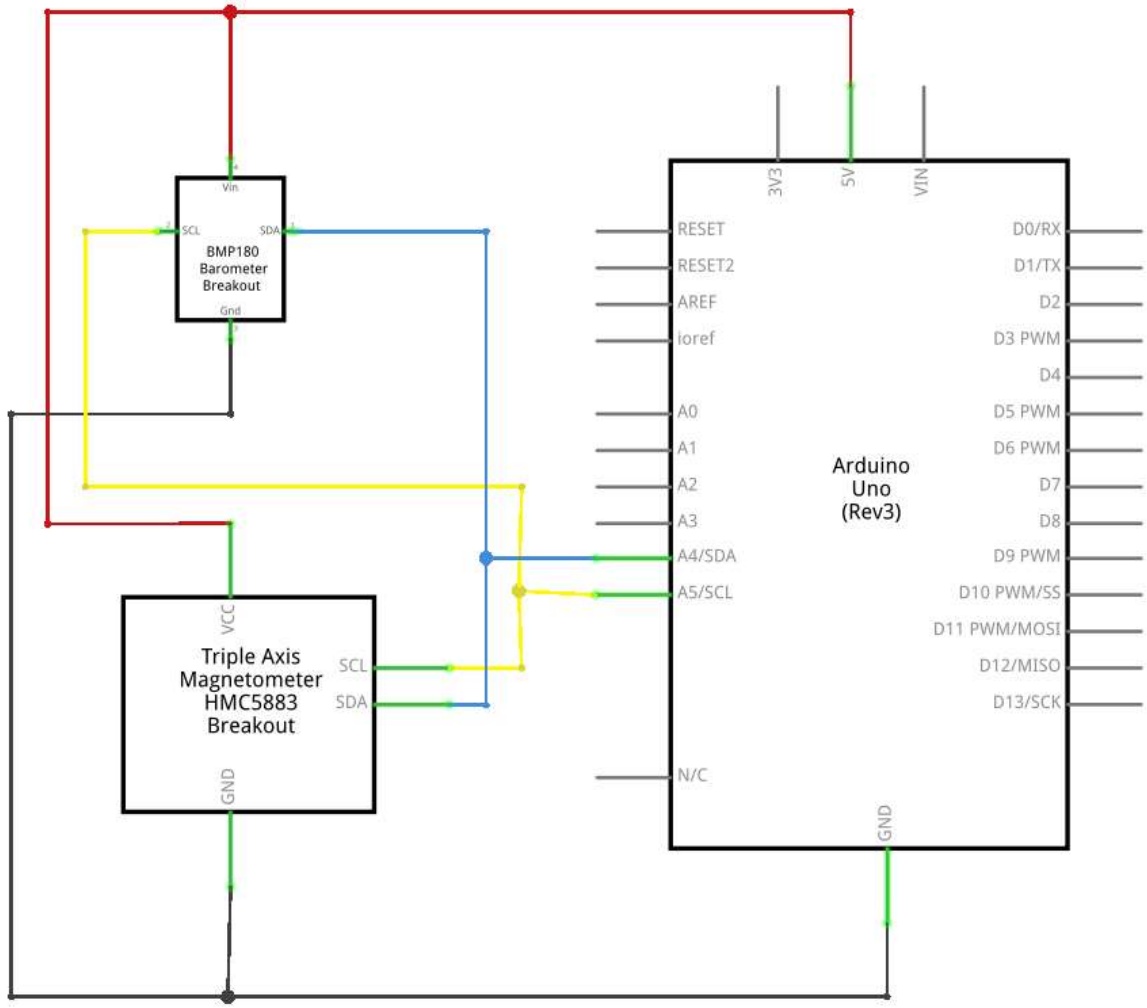

Figura 2.16. Esquema Electrónico.

Teniendo el diagrama y el esquema electrónico se procede al montaje de este. Se recomienda seguir los siguientes pasos:

1. Se acopla el módulo Ethernet Shield a la plataforma Arduino UNO.

En la Figura 2.17 se indica la fotografía del módulo Ethernet Shield acoplado al Arduino UNO.

Figura 2.17. Módulo Ethernet Shield acoplado al Arduino UNO.

2. Se coloca el magnetómetro HMC5883L y el sensor de presión barométrica BMP180 en el protoboard.

En la Figura 2.18 se indica la fotografía de los sensores colocados en el protoboard.

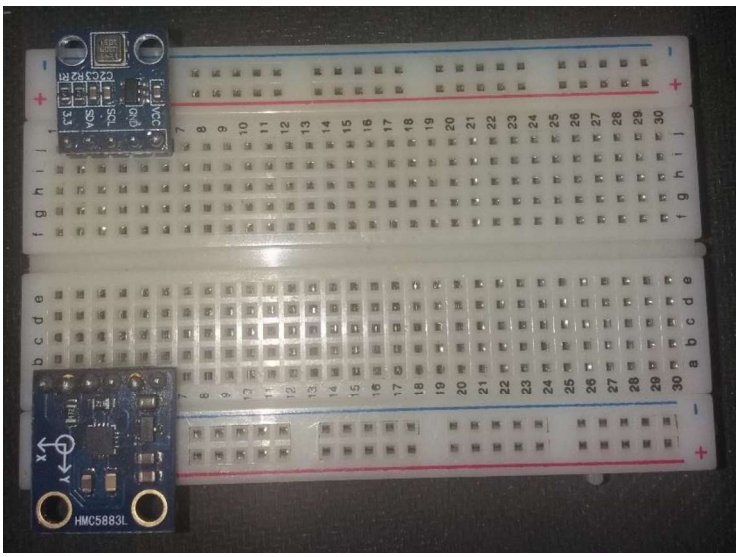

Figura 2.18. Sensores colocados en el Protoboard.

3. Se procede a la conexión circuital para obtener el montaje del diagrama electrónico de la Figura 2.15 de este proyecto.

En la Figura 2.19 se indica la fotografía del montaje del diagrama electrónico.

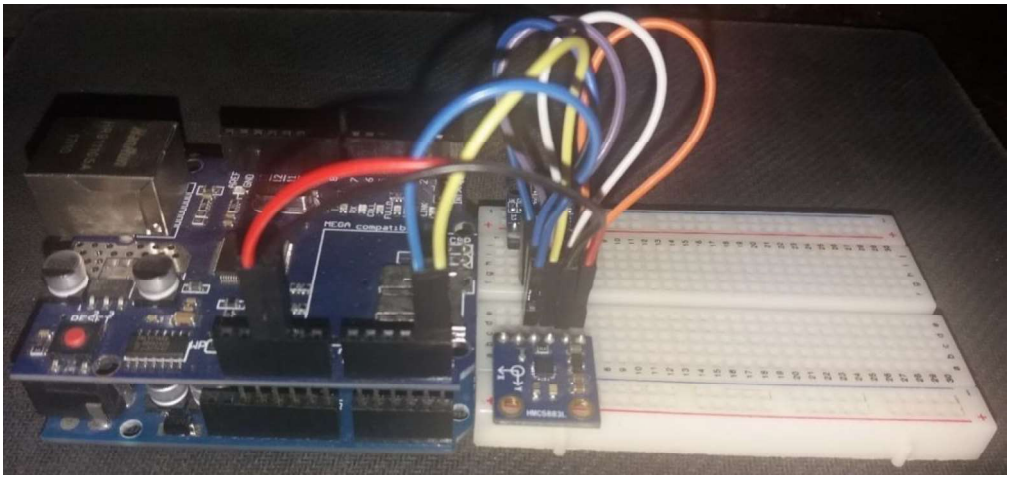

Figura 2.19. Montaje del diagrama electrónico.

El diseño de la Red LAN se muestra en la Figura 2.2 debido a que solo se requiere comprobar el correcto funcionamiento de la conexión Ethernet. El diseño es una conexión de forma directa del computador al Arduino utilizando un cable Ethernet. Para llevar a cabo el diseño se procederá de la siguiente manera:

1. Conectar el cable USB tipo B desde el Arduino al Cargador USB el cual está conectado a un tomacorriente para alimentar los dispositivos electrónicos.
2. Conectar el cable Ethernet del computador al módulo Ethernet Shield.

Con esto, el hardware del diseño está listo, sin embargo, aún resta del software para tener todo el diseño de forma operativa.

2.3.2.3 Aplicación en Arduino

Antes de desarrollar el Sketch para el diseño de la Red LAN, se debe revisar e incluir las librerías del magnetómetro HMC5883L y el sensor de presión barométrica BMP180 para su uso en el IDE de Arduino.

La librería del magnetómetro HMC5883L se puede descargar desde la página del fabricante o a su vez desde la página oficial de GitHub [55] donde varios desarrolladores hacen sus variantes a las librerías según sus requerimientos. Se descarga desde GitHub

la librería del magnetómetro HMC5883L desarrollada por Jeff Rowberg, procediendo de la siguiente manera:

1. Descargar el paquete de librerías desarrollados por Jeff Rowberg de [56].
2. Descomprimir el archivo i2cdevlib-master.zip en el escritorio o en la misma carpeta donde se descargó.

En la Figura 2.20 se indica el paquete de librerías desarrollados por Jeff Rowberg.

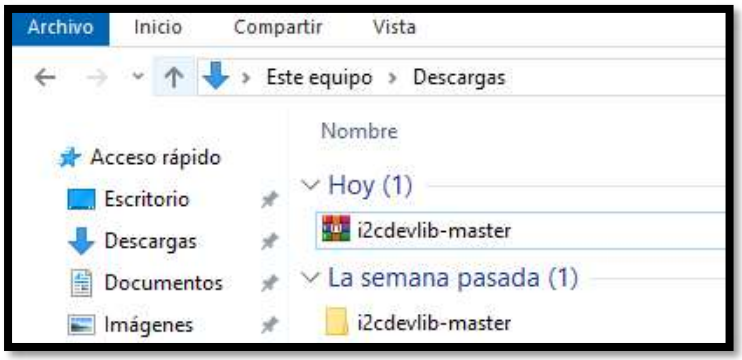

Figura 2.20. Paquete de librerías desarrollados por Jeff Rowberg.

3. Ingresar al IDE de Arduino. Buscar la opción *"Añadir librería .ZIP…"* en la siguiente ruta (Programa – Incluir Librería – Añadir librería .ZIP…) como se muestra en la Figura 2.21.

En la Figura 2.21 se indica la imagen de cómo se añade una nueva librería .ZIP en Arduino.

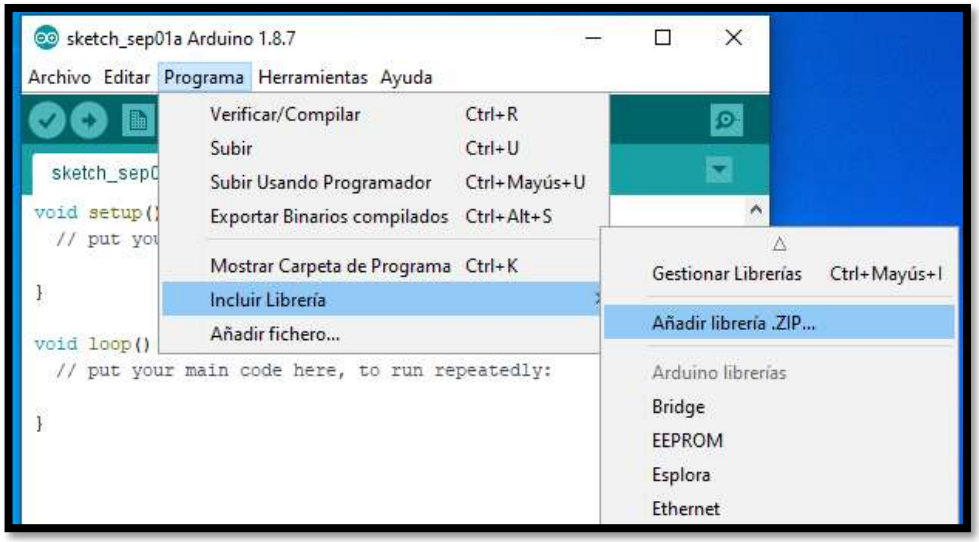

Figura 2.21. Añadir librería .ZIP.

4. Buscar la carpeta *"HMC5883L"* en la siguiente dirección *"C:\Users\User\Downloads\i2cdevlib-master\Arduino"* y seleccionar la carpeta *"HMC5883L"* para después dar clic en *"Abrir"*.

En la Figura 2.22 se indica la imagen de cómo se añade una nueva librería .ZIP en Arduino.

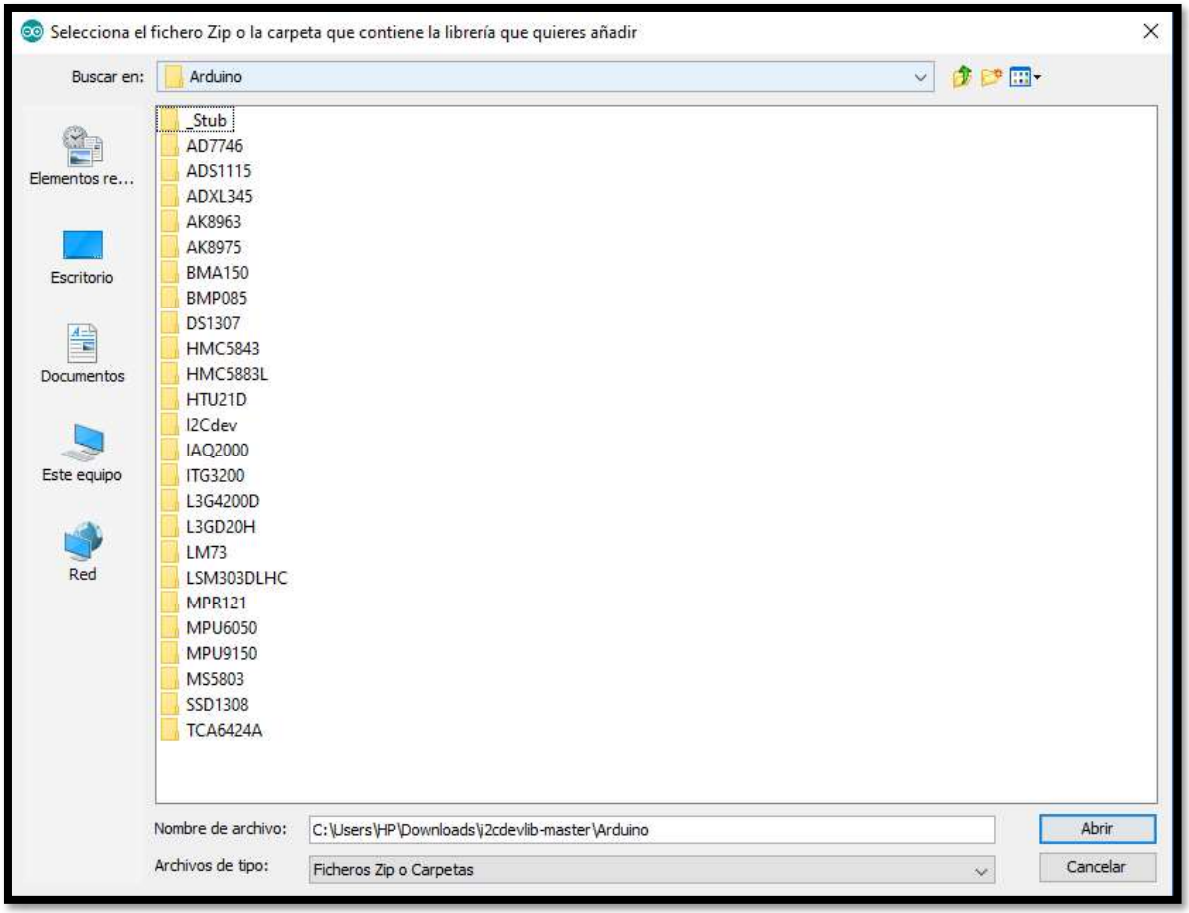

Figura 2.22. Añadir las carpetas "HMC5883L" y "I2Cdev" a las librerías Arduino.

5. Además, la librería del magnetómetro HMC5883L desarrollada por Jeff Rowberg trabaja con una librería adicional para la comunicación I2C que se encuentra en el paquete de librerías descargado anteriormente. Repetir el paso 3.

6. Buscar la carpeta *"I2Cdev"* en la siguiente dirección *"C:\Users\User\Downloads\i2cdevlib-master\Arduino"* y seleccionar la carpeta *"I2Cdev"* para después dar clic en *"Abrir"* como se muestra en la Figura 2.22.

La librería del sensor de presión barométrica BMP180 se puede descargar desde la página del fabricante o, a su vez, desde la página oficial de GitHub [55]. Se descarga desde GitHub la librería del sensor de presión barométrica BMP180 desarrollada por Sparkfun, procediendo de la siguiente manera:

1. Descargar la librería del sensor BMP180 desarrollada por Sparkfun de [57].

2. Descomprimir el archivo BMP180_Breakout-master.zip en el escritorio o en la misma carpeta donde se descargó.

En la Figura 2.23 se indica la librería del sensor BMP180 desarrollada por Sparkfun.

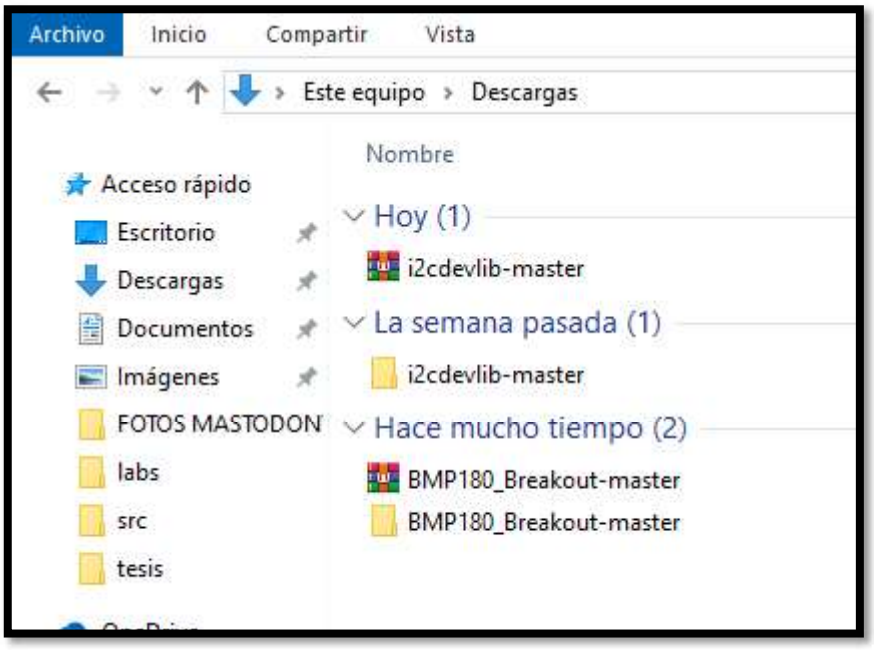

Figura 2.23. Librería del sensor BMP180 desarrollada por Sparkfun.

3. Ingresar al IDE de Arduino. Buscar la opción *"Añadir librería .ZIP…"* en la siguiente ruta (Programa – Incluir Librería – Añadir librería .ZIP…) como se muestra en la Figura 2.21 de la sección anterior.
4. Buscar la carpeta *"Arduino"* en la siguiente dirección *"C:\Users\User\Downloads\BMP180_Breakout-master\Libraries"* y seleccionar la carpeta *"Arduino"* para después dar clic en *"Abrir"*.

En la Figura 2.24 se indica la imagen al añadir la carpeta "Arduino" que contiene la librería del sensor BMP180.

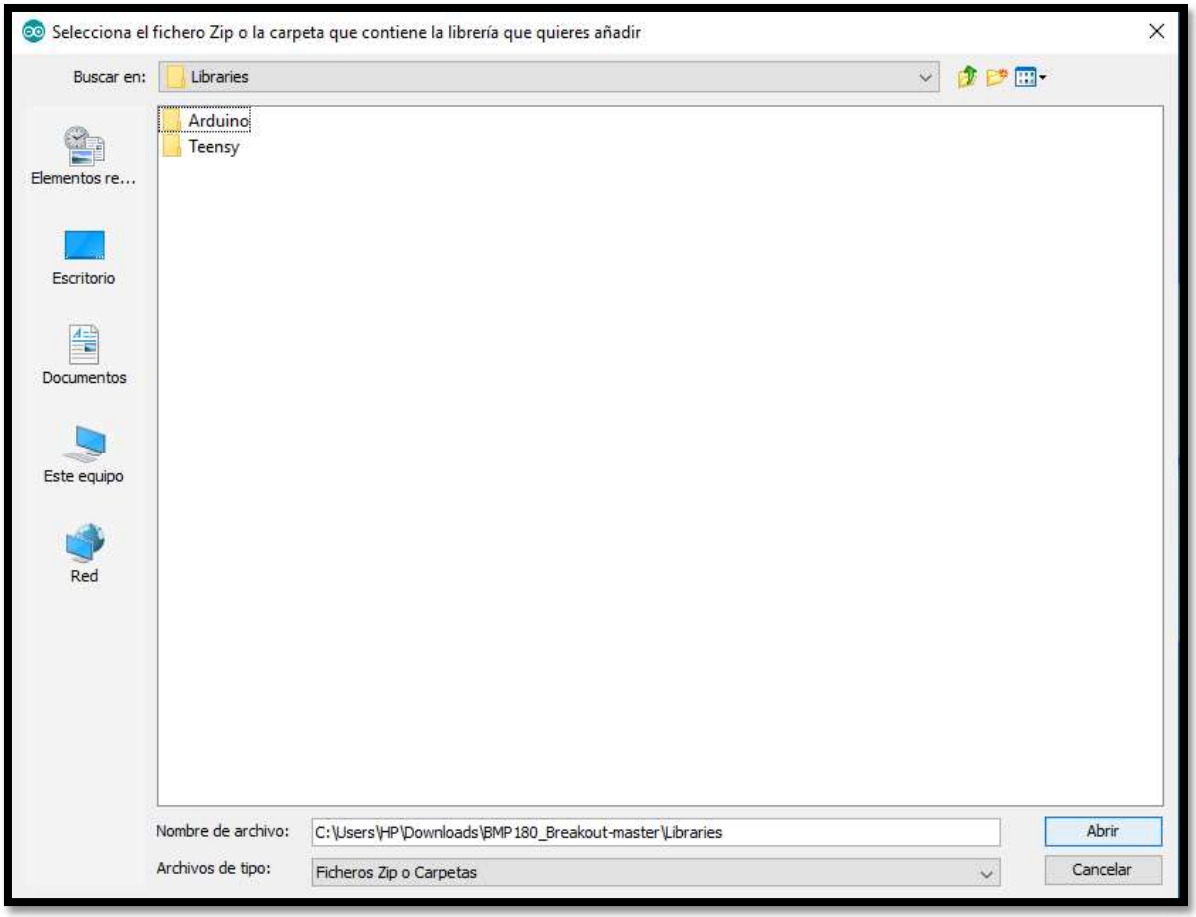

Figura 2.24. Añadir carpeta "Arduino" a las librerías Arduino.

Si se añadieron correctamente ambas librerías aparecerán las carpetas en la dirección *"C:\Users\User\Documents\Arduino\libraries"* como se ve en la Figura 2.25 o también se puede verificar en el IDE de Arduino en la siguiente ruta (Programa – Incluir Librería) se desplaza al final de la ventana y se observara las librerías añadidas que se muestra en la Figura 2.26.

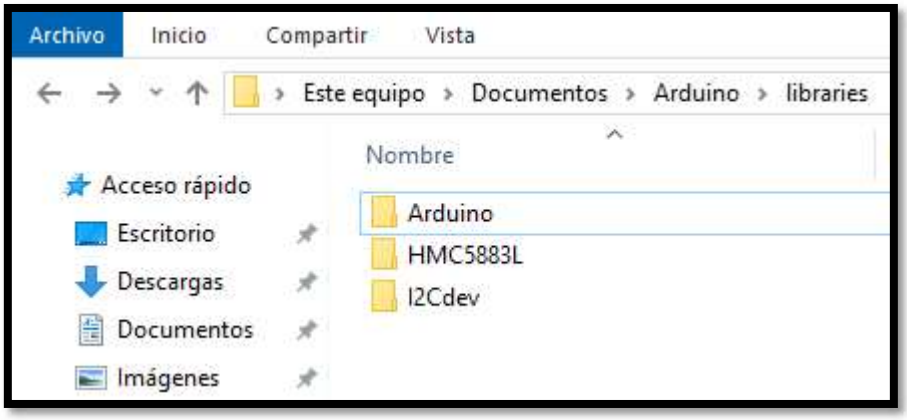

Figura 2.25. Carpetas añadidas a las librerías Arduino.

En la Figura 2.26 se indican las nuevas librerías añadidas en Arduino.

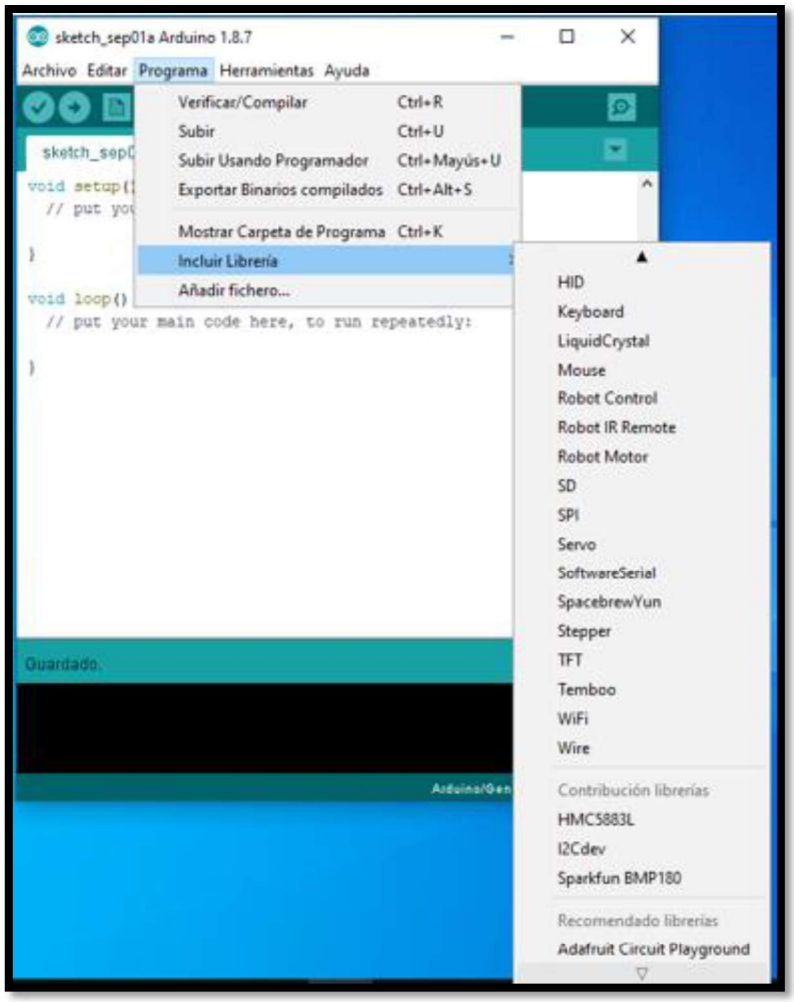

Figura 2.26. Nuevas librerías añadidas a las librerías Arduino.

Al comenzar con el Sketch de Arduino, se deben inicializar las librerías que se van a utilizar. De igual manera, es necesario arrancar la sesión Ethernet antes de establecer una conexión cliente – servidor. Intentarán conectarse tanto el Cliente 1 como el Cliente 2 al Servidor (aplicación en Visual Studio). Si se establece la conexión, el cliente enviará los datos al servidor. En caso contrario, la conexión se dará por terminada. La Figura 2.27 muestra el diagrama a seguir para desarrollar el Sketch de Arduino para la Red LAN.

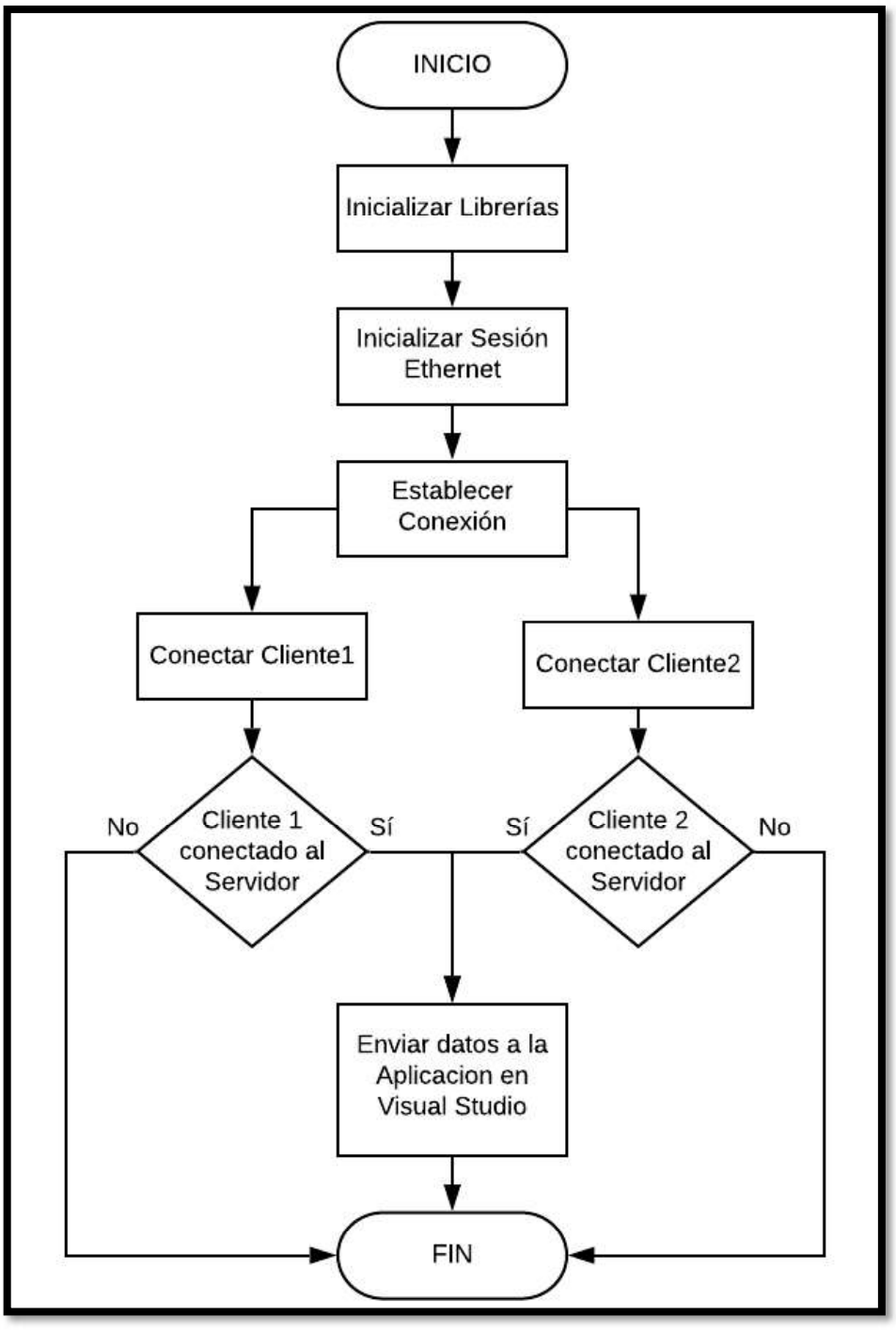

Figura 2.27. Diagrama para el Sketch de Arduino.

Con todas las librerías listas que requiere el diseño y con el diagrama de la Figura 2.27, lo siguiente es desarrollar el código para el Sketch de la Red LAN. En el Anexo B se presentará el Sketch de la aplicación de Arduino completo. A continuación, se describirá de manera global cada parte del Sketch de la aplicación desarrollada en Arduino:

1. Incluir las librerías necesarias para el desarrollo del Sketch, como se muestra en la Figura 2.28.

```
#include <SFE_BMP180.h> //Libreria de presión barométrica
#include <Wire.h>       //Libreria para la comunicación I2C con Arduino
////////////////////////////////////////////////////////////////////////////////////////////////

#include <I2Cdev.h>     //Libreria adicional I2C para el magnetómetro
#include <HMC5883L.h>   //Libreria del magnetómetro
////////////////////////////////////////////////////////////////////////////////////////////////

#include <SPI.h>        //Libreria para la comunicación entre el Arduino y el Ethernet Shield
#include <Ethernet.h>   //Libreria para el uso de Ethernet Shield
```

Figura 2.28. Librerías incluidas en el Sketch Arduino.

2. Declarar los objetos y variables globales tanto para el magnetómetro como para el barómetro:

```
SFE_BMP180 barometro;
```

La instrucción ***"SFE_BMP180"*** sirve para declarar el objeto al sensor bmp180 y ***"barometro"*** es el nombre del objeto.

```
double PresionNivelMar=1013.25
```

La instrucción ***"double"*** es una variable de punto flotante de 32 bits que se aplica cuando se usa números con decimales y ***"PresionNivelMar=1013.25"*** es el nombre de la variable asignada con un valor decimal, esta variable es la presión sobre el nivel del mar medido en milibares.

```
HMC5883L magnetometro
```

La instrucción ***"HMC5883L"*** sirve para declarar el objeto al sensor HMC5883L y ***"magnetometro"*** es el nombre del objeto.

```
int16_t mx, my, mz
```

La instrucción ***"int16_t"*** es una variable entera de 16 bits con signo y ***"mx, my, mz"*** son el nombre de las variables de posición espacial X, Y, Z asignadas.

3. Asignar los parámetros básicos para la comunicación Ethernet como se muestra en la Tabla 2.6. Todos los parámetros son asignados a criterio del programador. Tomar en cuenta que no deben existir dentro de la Red LAN: dirección ip duplicadas y dirección mac duplicadas. Al diseñar, no olvidar que, para transferir datos de una terminal a otra terminal, las direcciones ip de las terminales deben coexistir dentro de la misma Red o Subred.

En la Tabla 2.6 se indican las direcciones ip para 1 servidor – 2 clientes.

Tabla 2.6. Direcciones IP para 1 servidor – 2 clientes.

Dirección IP	Máscara de Subred	Equipo Terminales
192.168.0.215	255.255.255.0	Cliente (Arduino)
192.168.0.221	255.255.255.0	Servidor (Computador)

El programador para asignar las variables lo hace a través de la instrucción ***"byte"*** que almacena un valor entero de 8 bits sin signo. Este valor puede ser escrito en decimal, binario, octal o hexadecimal. Los nombres que sean asignados serán según la necesidad de la aplicación:

- **Mac [ ]:** dirección mac del Arduino, proporcionada según el criterio del programador.
- **Ip [ ]:** dirección ip del Arduino, ver Tabla 2.6.
- **Subnet [ ]:** mascara de red del Arduino, ver Tabla 2.6.
- **Server [ ]:** dirección ip del PC con el aplicativo, ver Tabla 2.6.

En la Figura 2.29 se indican los parámetros de diseño para la comunicación Ethernet.

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xAD };
byte ip[] = { 192, 168, 0, 215 };
byte subnet[] = { 255, 255, 255, 0 };
byte server[] = { 192, 168, 0, 221 };
```

Figura 2.29. Parámetros de diseño para la comunicación Ethernet.

4. Crear los clientes dentro del Arduino para la transmisión Ethernet:

```
EthernetClient cliente1
```

La instrucción ***"EthernetClient"*** crea el espacio para una variable cliente con el nombre que el programador decida para que pueda ser asignada con otra instrucción una dirección IP y un puerto específicos.

5. Dentro de la función *"setup()"* se inicializa la comunicación I2C, el magnetómetro, el barómetro y la comunicación Ethernet. Además, se arranca la conexión serial que servirá al programador como un indicador de control para verificar si se ha establecido una conexión Ethernet:

```
Wire.begin()
```

La instrucción ***"Wire.begin()"*** inicializa la comunicación I2C llamando a la librería "Wire" que es donde se encuentra el programa adicional del magnetómetro para su comunicación I2C.

```
magnetometro.initialize()
```

La instrucción ***"magnetometro.initialize()"*** inicializa la librería del sensor HMC5883L a través del objeto llamado magnetometro.

```
barometro.begin()
```

La instrucción ***"barometro.begin()"*** inicializa la librería del sensor bmp180 a través del objeto llamado barometro.

```
Ethernet.begin(mac, ip, subnet)
```

La instrucción ***"Ethernet.begin()"*** inicializa la conexión Ethernet del Arduino en base a los variables solicitadas ***"mac, ip, subnet"***.

```
Serial.begin(9600)
```

La instrucción ***"Serial.begin()"*** inicializa la comunicación serial del Arduino y ***"9600"*** establece la velocidad de los datos en bits por segundo (baudios), valores normalmente usados son: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200. Hay que considerar que ambos equipos deben tener la misma velocidad para poder realizar una comunicación serial. La comunicación serial dentro del programa principal es irrelevante porque no afecta en nada sobre el objetivo del proyecto, sin embargo, sirve al programador como apoyo para determinar si existió una comunicación Ethernet.

6. Dentro de la función *"loop()",* se crearán subfunciones para tener un mejor control de la aplicación desarrollada en Arduino tanto para corrección de errores o cambios del mismo.

7. Dentro de la subfunción *"Conexion1 ()",* se cargarán y guardarán los componentes del campo magnético del magnetómetro:

```
magnetometro.getHeading(&mx, &my, &mz)
```

La instrucción ***"magnetometro.getHeading()"*** viene diseñada para la librería del magnetómetro que realiza una lectura y almacena las componentes X, Y y Z que detecta el sensor HMC5883L y ***"&mx, &my, &mz"*** son las variables en las cuales se guardaran los datos.

8. Dentro de la subfunción *"Conexion1 ()",* se conectará y enviará datos del cliente1 al servidor por el puerto que el programador defina (en este caso se usará el puerto 11500) obteniendo una comunicación Ethernet utilizando un Socket:

```
cliente1.connect(server, 11500)
```

La instrucción ***"cliente1.connect()"*** conecta la variable "cliente1" a la comunicación Ethernet del Arduino y ***"server, 11500"*** son las variables que la instrucción usará como dirección IP y puerto especificados.

```
cliente1.print(mx)
```

La instrucción ***"cliente1.print()"*** es un comando de Arduino que envía lo que está dentro del paréntesis por vía Ethernet, puede enviar valores de variables o si se requiere enviar texto plano se usan comillas.

9. Dentro de la subfunción *"Desconexion1 ()",* se realizará la desconexión del cliente1 en el caso que no haya sido exitosa una conexión con el servidor:

```
cliente1.connected()
```

La instrucción ***"cliente1.conected()"*** informa que la variable "cliente1" está conectada con 1 lógico y con un 0 lógico en caso contrario.

```
cliente1.stop()
```

La instrucción ***"cliente1.stop()"*** desconecta la variable "cliente1" del servidor al que se encuentra conectada.

10. Dentro de la subfunción *"Conexion2 ()",* se crean las variables locales del barómetro. La variable T es para la "*temperatura*", la variable P es para la "*presión*" y la variable A es para la "*altitud*".

11. Dentro de la subfunción *"Conexion2 ()",* se cargarán y guardarán los componentes de altitud, presión y temperatura del barómetro. Todos los códigos vienen diseñados para la librería del barómetro que realizan una lectura y almacenan componentes de altitud, presión y temperatura. Se explicará a continuación la función de cada código:

```
barometro.startTemperature()
```

La instrucción ***"barometro.startTemperature()"*** inicia la lectura de la temperatura.

```
barometro.getTemperature(T)
```

La instrucción ***"barometro.getTemperature()"*** obtiene un valor de temperatura y lo almacena en la variable ***"T"***.

```
barometro.startPressure(3)
```

La instrucción ***"barometro.startPressure()"*** inicia la lectura de la presión atmosférica. El número ***"3"*** indica la cantidad de muestras que se tomaran.

```
barometro.getPressure(P,T)
```

La instrucción ***"barometro.getPressure()"*** obtiene un valor de la presión y lo almacena en la variable ***"P"***, es necesario incluir el valor de la temperatura ***"T"*** para el cálculo de la presión.

```
barometro.altitude(P,PresionNivelMar)
```

La instrucción ***"barometro.altitude()"*** calcula el valor de la altitud entre dos puntos en mbar, en este caso como indican las variables ***"P,PresionNivelMar"*** desde la presión a nivel del mar hasta donde se tomó el valor de la presión.

12. Dentro de la subfunción *"Conexion2 ()",* se conectará y enviará datos del cliente2 al servidor por el puerto que el programador defina (en este caso se usará el puerto 11501) obteniendo una comunicación Ethernet utilizando un Socket.
13. Dentro de la subfunción *"Desconexion2 ()",* se realizará la desconexión del cliente2 en el caso que no haya sido exitosa una conexión con el servidor.
14. Dentro de la subfunción *"Refrescar ()",* se realizará la desconexión del cliente1 y del cliente2. Esto se lo realiza para refrescar la memoria del Arduino y liberar las sesiones.

2.3.2.4 Aplicación en Visual Studio

Previamente, se necesita instalar Microsoft Visual Studio Community 2017 para poder desarrollar una aplicación en lenguaje C al diseño de la Red LAN, suponiendo que no se tenga instalado Microsoft Visual Studio Community 2017, el proceso es el siguiente:

1. Poseer una cuenta en Microsoft (puede usarse cualquier cuenta de Outlook o Hotmail); en el caso de no tener una cuenta en Microsoft crearse una cuenta.
2. Descargar Microsoft Visual Studio Community 2017 de [58].
3. Seguir todos los pasos de autenticación que determina Microsoft antes de la descarga.
4. Buscar la opción *"Visual Studio Community 2017 (version 15.9)"* y escoger la opción de descarga.
5. Instalar Microsoft Visual Studio Community 2017.

Al iniciar la aplicación en Visual Studio, se deben generar los espacios de nombres que se van a utilizar. De igual manera, es útil considerar las variables globales antes de establecer una conexión cliente – servidor. Intentaran conectarse tanto el Cliente 1 como el Cliente 2 al Servidor (aplicación en Visual Studio). Si se establece la conexión, el servidor recibirá los datos de cada uno de los clientes. En caso contrario, la conexión se

dará por terminada. La Figura 2.30 muestra el diagrama a seguir para desarrollar la aplicación en Visual Studio para la Red LAN.

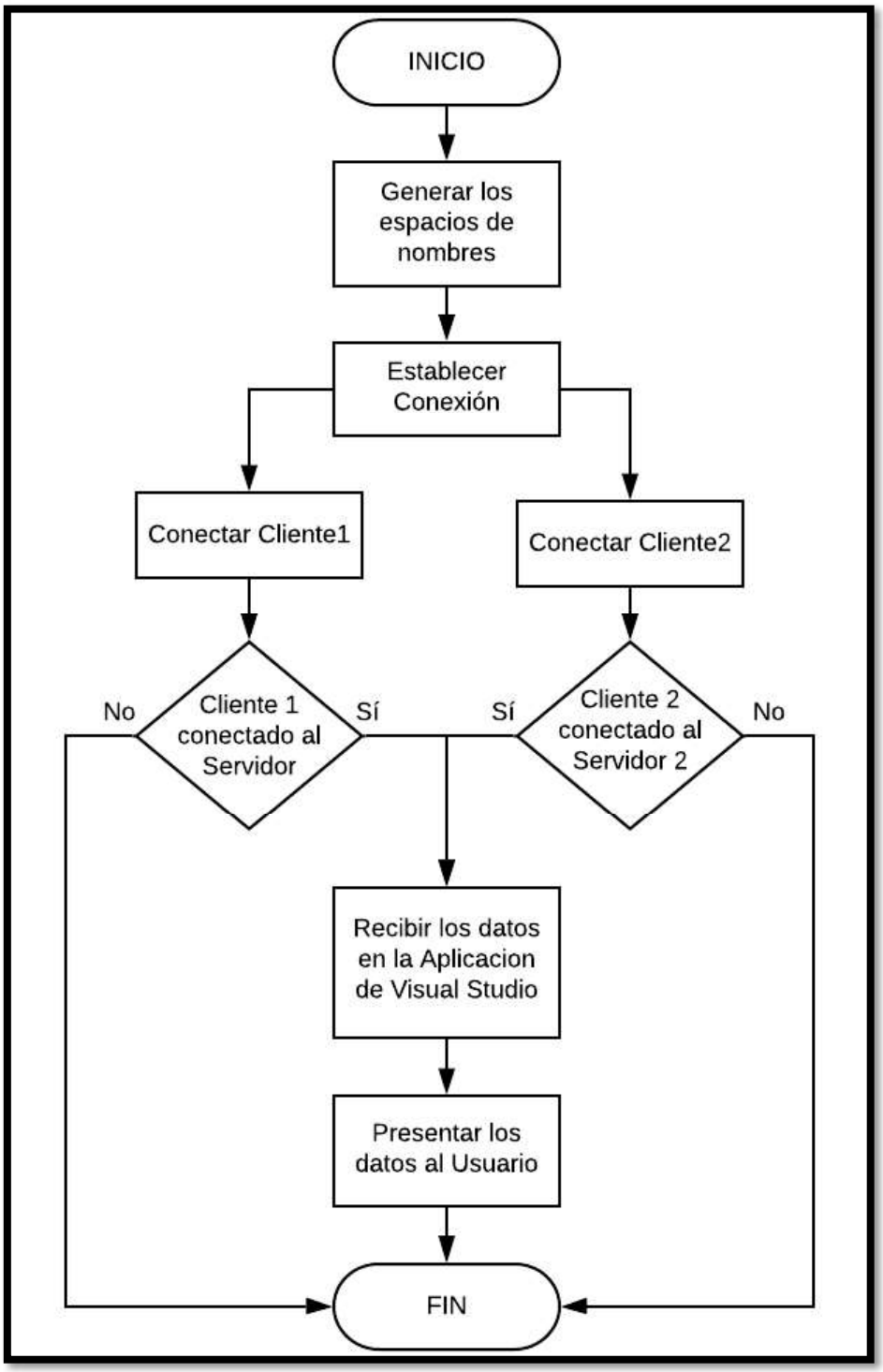

Figura 2.30. Diagrama para la aplicación en Visual Studio.

Antes de comenzar a desarrollar una aplicación en Visual Studio usando el formato de *"Aplicación de Windows Forms"* el programador debe tener claro como desea presentar los datos al usuario, es decir, cómo se va a ver la aplicación en Windows. En el presente diseño se requiere ver los datos del magnetómetro y del barómetro de una manera sencilla y fácil de entender, además de guardar dichos datos en un documento de texto plano en formato TXT, para lo cual se procederá de la siguiente manera:

1. Crear un nuevo proyecto en Visual Studio como aplicación de Windows Forms bajo el concepto de Visual Basic.

En la Figura 2.31 se indica la imagen para crear una nueva aplicación de Windows Form en Visual Studio.

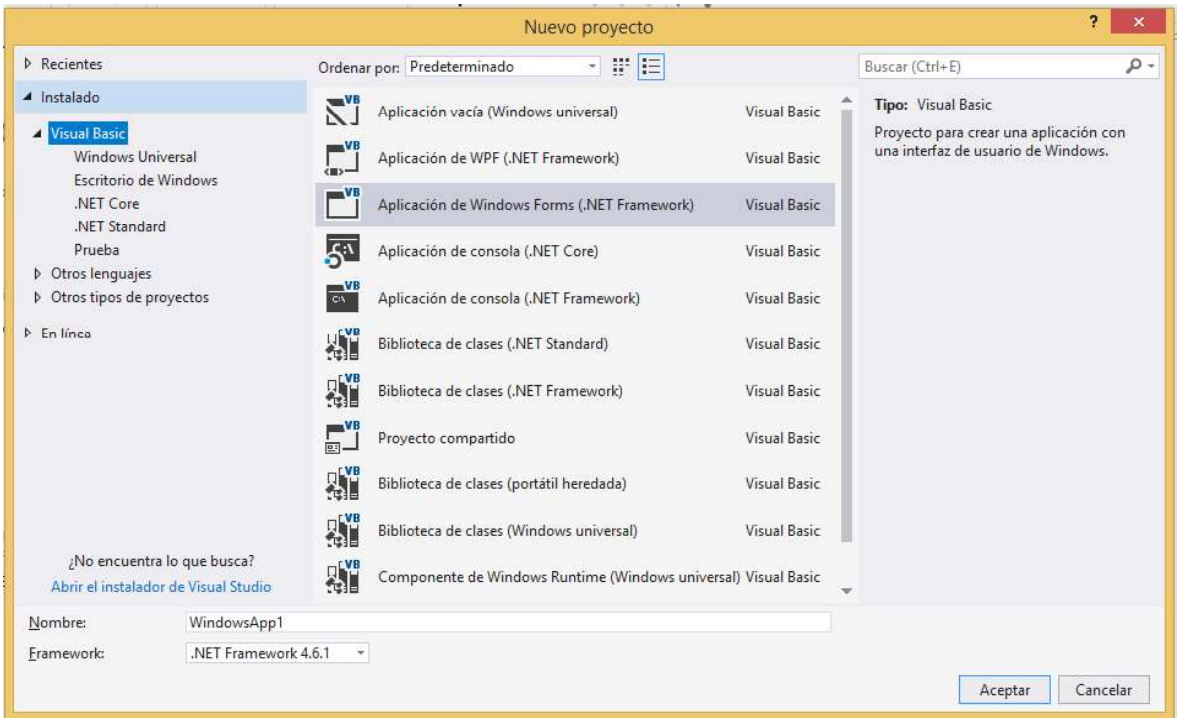

Figura 2.31. Nueva aplicación de Windows Forms.

2. Diseñar la forma de la aplicación que se despliega en Windows. Para el presente proyecto la aplicación tiene la forma que se muestra en la Figura 2.32.

Figura 2.32. Diseño de la aplicación.

Teniendo en cuenta el diagrama de la Figura 2.30 y el diseño de la aplicación de la Figura 2.32, lo siguiente es desarrollar el código para el aplicación en Visual Studio de la Red LAN. En el Anexo B se presenta el código de la aplicación desarrollada en Visual Studio completo. A continuación, se describe de manera global las partes más importantes del código de la aplicación en Visual Studio:

1. Generar los espacios de nombres necesarios para el desarrollo del código de la aplicación, como se muestra en la Figura 2.33.

```
Imports System.Net          ''Espacio de nombre para el uso de protocolos de Red
Imports System.Net.Sockets  ''Espacio de nombre para la implementación de sockets en el acceso de la Red
Imports System.Text         ''Espacio de nombre para la codificación de caracteres ASCII y Unicode
```

Figura 2.33. Espacios de nombres en el código de la aplicación.

2. Declarar las variables globales que se usaran en todo el código de la aplicación:

```
Public Conexion1 As TcpListener
```

La instrucción ***"Public"*** hace que una variable sea global y pueda ser usada en cualquier punto de la aplicación, ***"Conexion1"*** es el nombre otorgado a la variable, ***"As"*** es un conector para especificar qué tipo de dato será la variable y ***"TcpListener"*** es un dato de tipo Object que escucha las conexiones de clientes de red TCP.

```
Public cliente1 As TcpClient
```

Mantiene el mismo formato de la declaración anterior, pero ***"TcpClient"*** es un dato de tipo Object que proporciona conexiones de cliente para servicios de red TCP.

```
Public Const maximo As Integer
```

La instrucción ***"Const"*** hace que una variable sea del tipo constante y a su vez ***"Integer"*** es un dato de tipo entero sin decimal.

3. Declarar las variables privadas: IP del servidor y los puertos que se conectan a cada uno de los clientes:

```
Dim ip As IPAddress = IPAddress.Parse("192.168.0.221")
```

La instrucción ***"Dim"*** declara y asigna un espacio de almacenamiento privado dentro de la clase donde se designa la variable, ***"IPAddress"*** es un dato de tipo Object que proporciona una dirección IP, además la función ***"IPAddress.Parse()"*** convierte una cadena de dirección IP como ***"192.168.0.221"*** a una instancia IPAddress.

4. A cada conexión se asigna o crea un nuevo tcplistener que estará escuchando por cada intento de conexión que se realice. Para que sea una conexión exitosa, el cliente debe estar conectado a la dirección IP del servidor y a su puerto asignado:

```
Conexion1 = New TcpListener(ip, puerto1)
```

La instrucción ***"New"*** crea una nueva instancia de objeto, ***"New TcpListener()"*** inicializa una nueva instancia TcpListener que escucha las conexiones entrantes con dirección IP ***"ip"*** y número de puerto especificado ***"puerto1"*** cuyos valores están definidos en dichas variables.

```
Conexion1.Start()
```

La instrucción ***"Conexion1.Start()"*** empieza a escuchar las solicitudes de conexiones entrantes.

5. Preguntar si existen solicitudes pendientes en Conexion1. Posteriormente se acepta, luego se controla y se mantiene la conexión. La información del cliente1 se guarda en el vector tcpclient1. Se repite el mismo proceso para Conexion2:

```
Conexion1.Pending()
```

La instrucción ***"Conexion1.Pending()"*** determina si existen solicitud de conexiones pendientes.

```
cliente1 = Conexion1.AcceptTcpClient
```

La instrucción ***"Conexion1.AcceptTcpClient"*** acepta una solicitud de conexión pendiente.

```
cliente1.Connected
```

La instrucción ***"cliente1.Connected"*** obtiene un valor que indica si el socket de un TcpClient está conectado.

6. Declarar las variables privadas almacenar la información recibida por conexión Ethernet del cliente1. Se repite el mismo proceso para el cliente2:

```
Dim leer1 As NetworkStream
```

A la variable se le aplica la instrucción ***"NetworkStream"*** que es un dato de tipo Object que proporciona una secuencia de datos subyacentes para el acceso a la red.

```
Dim bytes1() As Byte
```

A la variable se le aplica la instrucción ***"Byte"*** que es un dato de tipo carácter representado por un entero de 8 bits sin signo.

```
Dim cadena1 As String
```

A la variable se le aplica la instrucción ***"String"*** almacenando la información como un texto representado por una secuencia de caracteres.

```
Dim ruta = My.Computer.FileSystem.SpecialDirectories.Desktop & "\magnetometro.txt"
```

La instrucción ***"My.Computer.FileSystem.SpecialDirectories.Desktop"*** describe la ruta hacia el escritorio del computador donde se tendrá acceso y se podrá crear un archivo ***"\magnetometro.txt"*** en el caso que no exista el archivo.

7. Recopilación, almacenamiento, codificación y visualización de los datos recibidos desde el cliente1. Se repite el mismo proceso para el cliente2. Además, se recopilan

los datos codificados en un documento de texto plano en formato TXT para un manejo posterior de la información por parte del usuario.

```
leer1 = tcpclientes1(i).GetStream
```

La instrucción ***"tcpclientes1().GetStream"*** genera una recopilación de los datos byte a byte para almacenarla en una variable.

```
leer1.DataAvailable
```

La instrucción ***"leer1.DataAvailable"*** devuelve un valor indicando si hay datos disponibles en la variable.

```
ReDim bytes1(cliente1.ReceiveBufferSize)
```

La instrucción ***"ReDim"*** reasigna un espacio de almacenamiento privado dentro de la clase donde se designa la variable y ***"cliente1.ReceiveBufferSize"*** establece el tamaño según la recepción del buffer.

```
leer1.Read(bytes1, 0, bytes1.Length)
```

La instrucción ***"leer1.Read()"*** lee los datos de la variable leer1, según el tamaño de cada byte ***"bytes1.Length"*** y los almacena en la variable ***"bytes1"***.

```
Encoding.UTF8.GetString(bytes1)
```

La instrucción ***"Encoding.UTF8.GetString()"*** codifica los caracteres en formato UTF-8 que es método de codificación de caracteres de longitud variable (1 a 4 bytes) siendo capaz de representar cualquier carácter Unicode.

```
USUARIOS1.Text = cadena1.ToString
```

La instrucción ***"USUARIOS1.Text"*** establece la construcción de texto asociado al evento "USUARIOS1" y ***"cadena1.ToString"*** devuelve en forma de string los valores codificados almacenados en la variable "cadena1" para poder visualizar en la aplicación diseñada en Visual Studio.

```
My.Computer.FileSystem.WriteAllText(ruta, vbCrLf & USUARIOS1.Text & vbCrLf, True)
```

La instrucción ***"My.Computer.FileSystem.WriteAllText()"*** permite escribir los datos de "USUARIOS1" en el documento de extensión TXT creado anteriormente.

2.3.3 DISEÑO DE LA RED LAN: 2 SERVIDORES – 2 CLIENTES

Este diseño de la Red LAN contempla el objetivo de este proyecto que es conectar mediante la comunicación Ethernet cada periférico (sensor) del Arduino como cliente hacia una aplicación desarrollada en Visual Studio como servidor utilizando Sockets siendo la nueva librería Ethernet la que permite esta comunicación Punto a Punto. Esta nueva librería Ethernet maneja de manera eficiente el Arduino permitiendo utilizar los 4 Sockets de manera simultánea, pero por uso práctico este diseño utilizará 2 Sockets de manera simultánea.

Como este nuevo diseño se basa sobre el *"Diseño de la Red LAN: 1 Servidor – 2 Clientes"* de la sección 2.3.2 se hace énfasis en los cambios a realizar para poder obtener un diseño de 2 Servidores – 2 Clientes.

2.3.3.1 Recursos y descripción de materiales

En esta sección se detalla los recursos para comprobar el funcionamiento de la conexión Ethernet utilizando la nueva librería Ethernet. Los recursos son los siguientes:

- 1 Arduino UNO
- 1 módulo Ethernet Shield
- 1 magnetómetro HMC5883L
- 1 sensor de presión barométrica BMP180
- 1 Protoboard
- Varios cables Jumpers Dupont Macho – Macho
- 1 cable Ethernet
- 1 cable USB tipo B
- 1 cargador USB
- 2 computadores
- 1 router doméstico

2.3.3.2 Diagrama y diseño

Se mantiene el diagrama electrónico de la Figura 2.15, el esquema electrónico de la Figura 2.16, y el montaje del diagrama electrónico de la Figura 2.19 de la sección 2.3.2.2.

El diseño de la Red LAN se muestra en la Figura 2.34 debido a que cada periférico (sensor) debe establecer una conexión con un servidor. El diseño es una conexión desde el Arduino hacia un router doméstico que servirá de switch utilizando un cable Ethernet y desde el router se conectaran 2 computadores sea por cable Ethernet o por medio inalámbrico (WIFI), para poder llevar a cabo el diseño se procederá de la siguiente manera:

1. Conectar el cable USB tipo B desde el Arduino al Cargador USB el cual estará conectado a un tomacorriente para alimentar los dispositivos electrónicos.
2. Conectar el cable Ethernet del módulo Ethernet Shield al router doméstico.
3. Conectar desde el router hacia 2 computadores (PC o Notebook) por medio inalámbrico o a través de un cable Ethernet.

En la Figura 2.34 se indica la imagen de una red LAN compuesta de 2 servidores – 2 clientes.

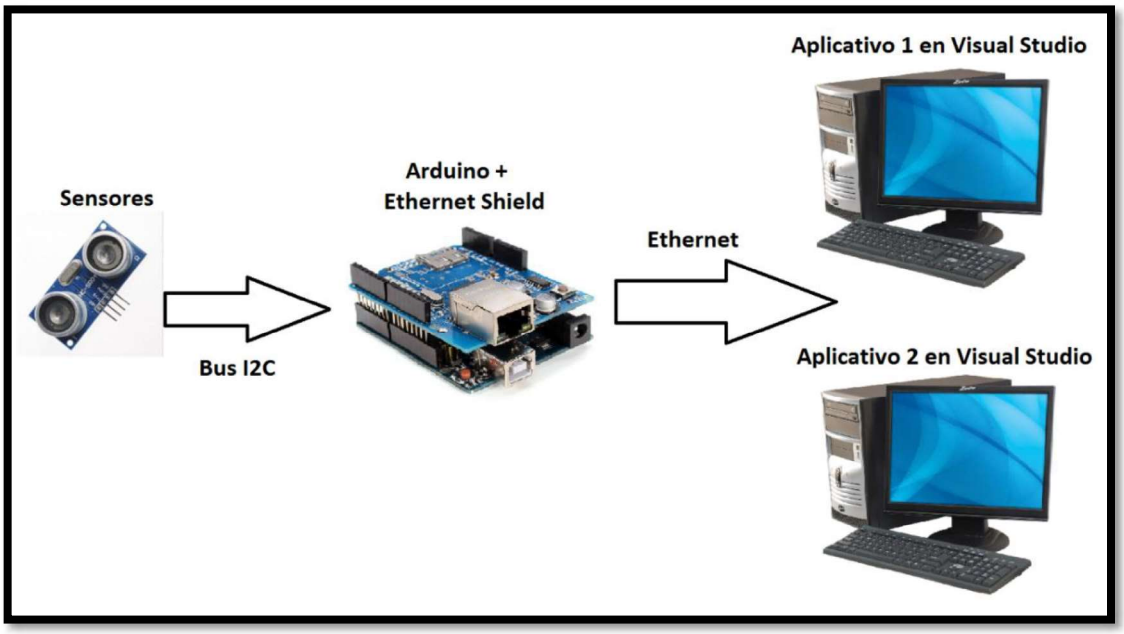

Figura 2.34. Red LAN: 2 servidores – 2 clientes.

De igual manera que en el subcapítulo anterior el diseño de la parte de hardware está listo, solo quedaría pendiente el diseño de la aplicación en Arduino para tener el proyecto completo.

2.3.3.3 Aplicación en Arduino

Se mantiene la librería del magnetómetro HMC5883L desarrollado por Jeff Rowberg y la librería del sensor de presión barométrica BMP180 desarrollado por Sparkfun de la sección 2.3.2.3.

Antes de desarrollar el Sketch de Arduino es importante saber cuál es la puerta de enlace (Gateway) del computador que está conectado al router doméstico. Se siguen los siguientes pasos:

1. Presionar las teclas “Windows” + “R” o abrir el “Ejecutar” de Windows.
2. Digitar “cmd” y dar clic en aceptar.
3. Digitar “ipconfig” y después digitar la tecla “Enter”.

Se desplegará la información básica del adaptador Ethernet que tiene el computador con la información de la puerta de enlace, la cual servirá para diseñar el Sketch Arduino, como se muestra en la Figura 2.35.

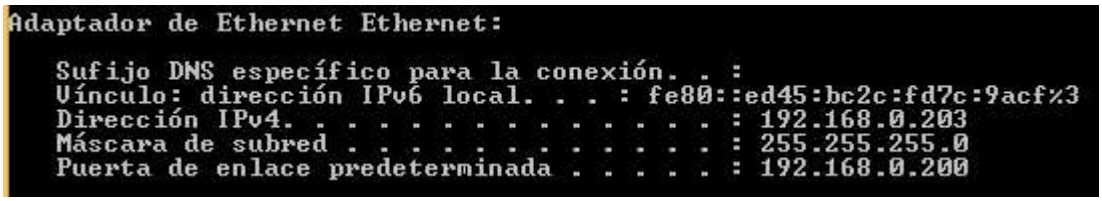

Figura 2.35. Puerta de enlace.

Al iniciar el Sketch de Arduino, se deben inicializar las librerías que se van a utilizar. De igual manera, es necesario inicializar la sesión Ethernet antes de establecer una conexión cliente – servidor. Intentarán conectarse tanto el Cliente 1 al Servidor 1 (computador 1) como el Cliente 2 al Servidor 2 (computador 2). Si se establece la conexión, el cliente enviará los datos al servidor. En caso contrario, la conexión se dará por terminada. La Figura 2.36 muestra el diagrama a seguir para desarrollar el Sketch de Arduino para la Red LAN.

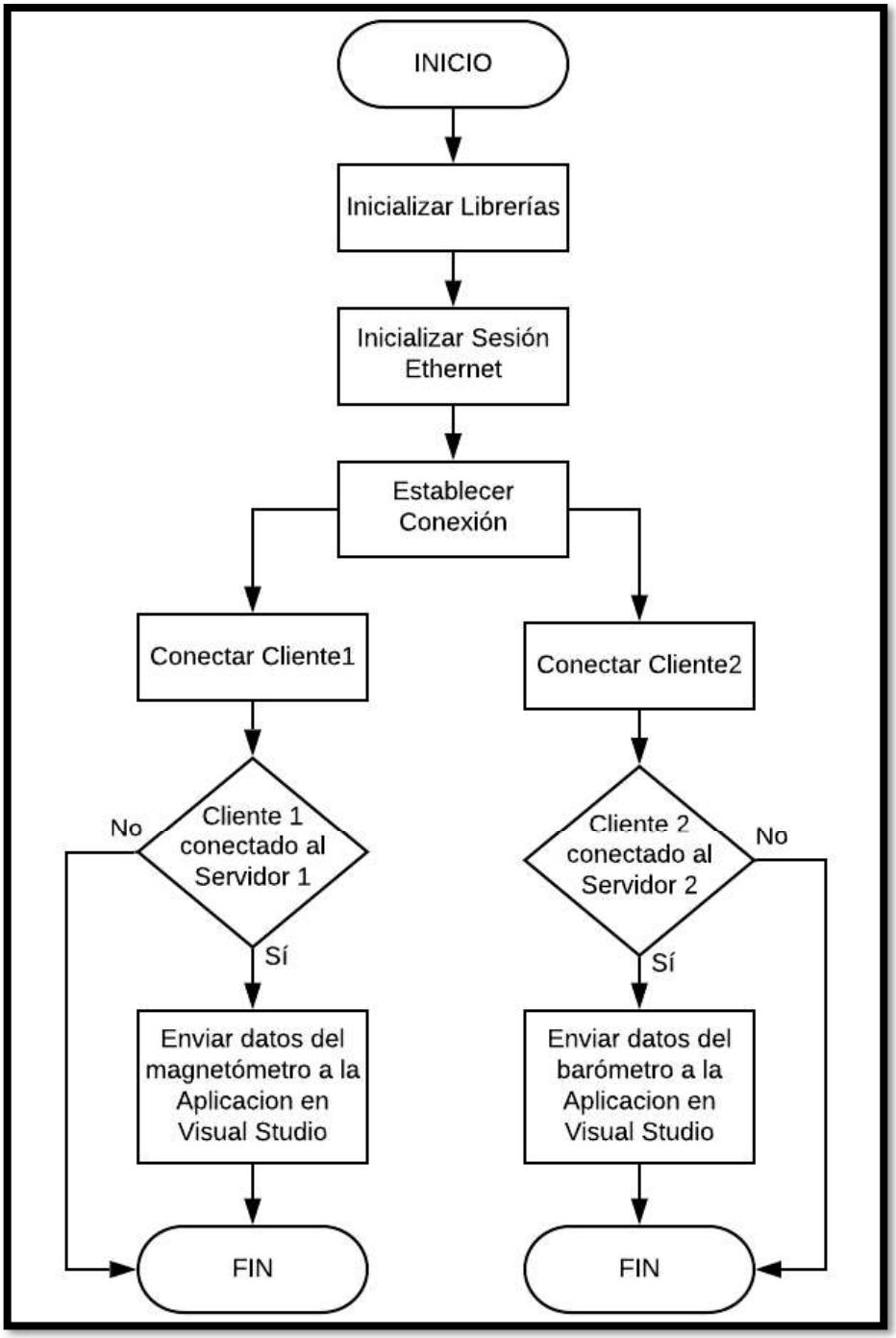

Figura 2.36. Diagrama para el Sketch de Arduino.

Con todas las librerías listas que requiere el diseño y con el diagrama de la Figura 2.36, lo siguiente es desarrollar el código para el Sketch de la Red LAN. En el Anexo C, se presentará el Sketch de la aplicación desarrollada en Arduino completo. El nuevo Sketch de la aplicación en Arduino se basa esencialmente en el Sketch de la sección 2.3.2.3, se describirá de manera global los cambios más significativos:

1. Asignar los parámetros básicos para la comunicación Ethernet como se muestra en la Tabla 2.7. Todos los parámetros son asignados a criterio del programador. Hay que considerar que no deben existir dentro de la Red LAN: dirección ip duplicadas y dirección mac duplicadas. Al diseñar no olvidar que, para transferir datos de una terminal a otra terminal, las direcciones ip de las terminales deben coexistir dentro de la misma Red o Subred. Al tener un router es necesario declarar la puerta de enlace (gateway) para evitar errores posteriores.

En la Tabla 2.7 se indican las direcciones ip para 2 servidores – 2 clientes.

Tabla 2.7. Direcciones IP para 2 servidores – 2 clientes.

Dirección IP	Máscara de Subred	Equipo Terminales
192.168.0.215	255.255.255.0	Cliente (Arduino)
192.168.0.200	255.255.255.0	Puerta de enlace (Gateway)
192.168.0.221	255.255.255.0	Servidor 1 (Computador 1)
192.168.0.222	255.255.255.0	Servidor 2 (Computador 2)

Los nombres que sean asignados son según la necesidad de la aplicación:

- **Mac [ ]:** dirección mac del Arduino, proporcionada según el criterio del programador.
- **Ip [ ]:** dirección ip del Arduino, ver Tabla 2.7.
- **Gateway [ ]:** puerta de enlace, ver Tabla 2.7.
- **Subnet [ ]:** mascara de red del Arduino, ver Tabla 2.7.
- **Server1 [ ]:** dirección ip del PC con el aplicativo 1, ver Tabla 2.7.
- **Server2 [ ]:** dirección ip del PC con el aplicativo 2, ver Tabla 2.7.

En la Figura 2.37 se indican los parámetros para la comunicación Ethernet hacia 2 servidores.

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xAD };
byte ip[] = { 192, 168, 0, 215 };
byte gateway[] = { 192, 168, 0, 200 };
byte subnet[] = { 255, 255, 255, 0 };
byte server1[] = { 192, 168, 0, 221 };
byte server2[] = { 192, 168, 0, 222 };
```

Figura 2.37. Parámetros para la comunicación Ethernet hacia 2 servidores.

2. Dentro de la función *"setup()"* se inicializa la comunicación I2C, el magnetómetro, el barómetro y la comunicación Ethernet. Además, se inicializa la conexión serial que servirá al programador como un indicador de control para verificar si se ha establecido una conexión Ethernet. Al inicializar la comunicación Ethernet, cabe considerar todos los parámetros necesarios (dirección mac, dirección ip, puerta de enlace y mascara de subred) para una correcta conexión Ethernet:

```
Ethernet.begin(mac, ip, gateway, subnet)
```

La instrucción ***"Ethernet.begin()"*** inicializa la conexión Ethernet del Arduino en base a los variables solicitadas ***"mac, ip, gateway, subnet"***.

3. Dentro de la subfunción *"Conexion1 ()",* se conectará y enviará datos del cliente1 al servidor1 por el puerto que el programador defina (en este caso se usará el puerto 11500) obteniendo una comunicación Ethernet utilizando un Socket:

```
cliente1.connect(server1, 11500)
```

La instrucción ***"cliente1.connect()"*** conecta la variable "cliente1" a la comunicación Ethernet del Arduino y ***"server1, 11500"*** son las variables que la instrucción usará como dirección IP y puerto especificados.

4. De la misma manera, en la subfunción *"Conexion2 ()",* se conectará y enviará datos del cliente2 al servidor2 por el puerto que el programador defina (en este caso se usará el mismo puerto 11500) obteniendo una comunicación Ethernet utilizando otro Socket:

```
cliente2.connect(server2, 11500)
```

La instrucción ***"cliente2.connect()"*** conecta la variable "cliente2" a la comunicación Ethernet del Arduino y ***"server2, 11500"*** son las variables que la instrucción usará como dirección IP y puerto especificados.

2.3.3.4 Aplicación en Visual Studio

Al desarrollar la aplicación en Visual Studio, se deben generar los espacios de nombres que se van a utilizar. De igual manera, es útil considerar las variables globales antes de establecer una conexión cliente – servidor. Intenta conectarse el Cliente 1 al Servidor 1 en un computador y de la misma forma el Cliente 2 al Servidor 2 en otro computador. Si se establece la conexión, el servidor recibirá los datos de cada uno de los clientes. En caso contrario, la conexión se dará por terminada. La Figura 2.38 muestra el diagrama a seguir para desarrollar la aplicación en Visual Studio para la Red LAN.

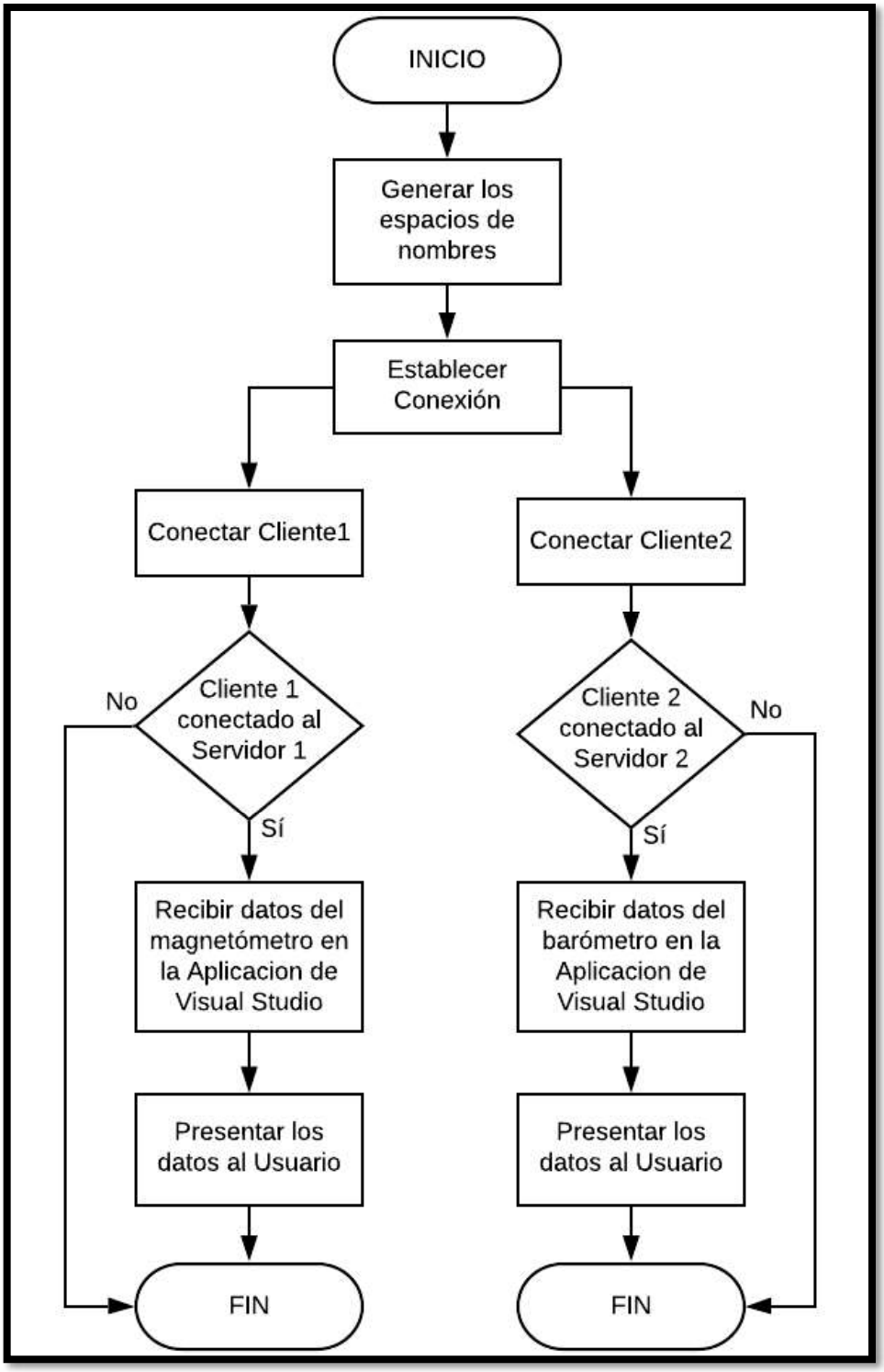

Figura 2.38. Diagrama para la aplicación en Visual Studio.

Antes de comenzar a desarrollar una aplicación en Visual Studio usando el formato de *"Aplicación de Windows Forms"* el programador debe tener claro como desea presentar los datos al usuario, es decir cómo se va a ver la aplicación en Windows. En el presente

diseño se requiere ver los datos del magnetómetro en una aplicación y del barómetro en otra aplicación de una manera sencilla y fácil de entender, además de guardar dichos datos en un documento de texto plano en formato TXT. Se creará un nuevo proyecto en Visual Studio como aplicación de Windows Forms bajo el concepto de Visual Basic de la Figura 2.31 para el magnetómetro y para el barómetro.

Se diseña la forma de la aplicación que se desplegará en Windows para la aplicación del magnetómetro. Para el presente proyecto la aplicación tendrá la forma que se muestra en la Figura 2.39.

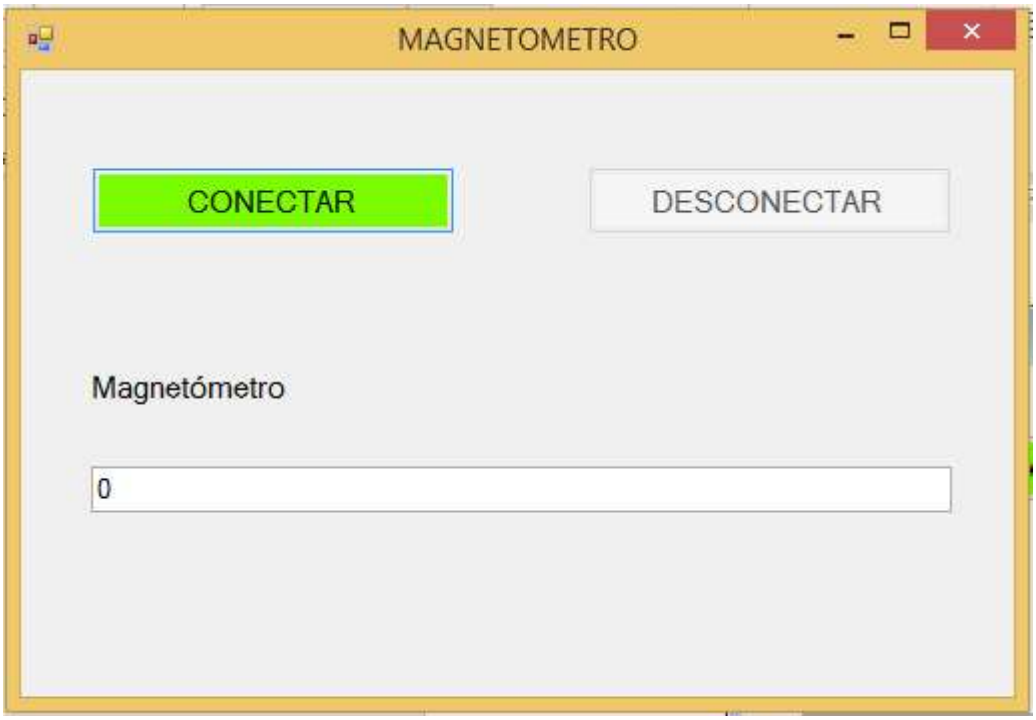

Figura 2.39. Diseño de la aplicación del magnetómetro.

Teniendo en cuenta el diagrama de la Figura 2.38 y el diseño de la aplicación de la Figura 2.39, lo siguiente es desarrollar el código para el aplicación en Visual Studio de la Red LAN. En el Anexo C se presentará el código de la aplicación magnetómetro desarrollada en Visual Studio completo. El nuevo código de la aplicación magnetómetro se basa esencialmente en la aplicación desarrollada en Visual Studio de la sección 2.3.2.4, descartando todas las líneas de código en exceso que no son utilizadas para la aplicación.

Se diseña la forma de la aplicación que se desplegará en Windows para la aplicación del barómetro. Para el presente proyecto la aplicación tendrá la forma que se muestra en la Figura 2.40.

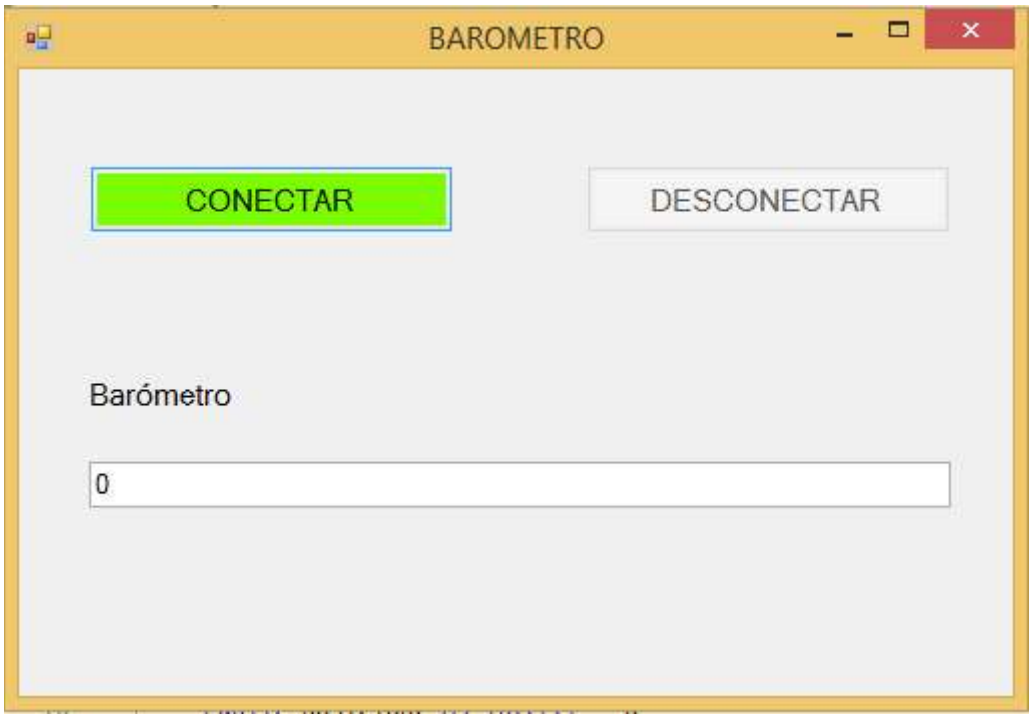

Figura 2.40. Diseño de la aplicación del barómetro.

Teniendo en cuenta el diagrama de la Figura 2.38 y el diseño de la aplicación de la Figura 2.40, lo siguiente es desarrollar el código para la aplicación en Visual Studio de la Red LAN. En el Anexo C se presentará el código de la aplicación barómetro desarrollada en Visual Studio completo. El nuevo código de la aplicación barómetro se basa esencialmente en la aplicación desarrollada en Visual Studio de la sección 2.3.2.4, descartando todas las líneas de código en exceso que no son utilizadas para la aplicación.

El único cambio significativo para tener en cuenta es la dirección IP del servidor 2 que es *"192.168.0.222"* y su puerto es el 11500 que es el mismo para el servidor 1.

En el siguiente capítulo se observa como la información enviada por los periféricos de Arduino se los presenta de una manera más amigable en la aplicación desarrollada en Visual Studio

3. RESULTADOS Y DISCUSIÓN

En este capítulo se analiza la conectividad Ethernet de ambos diseños LAN realizados en el capítulo anterior para comprobar su funcionamiento y posteriormente presentar los resultados. Se efectuan pruebas de conectividad Ethernet a través de 3 mecanismos: usando el comando PING, con líneas de código dentro del Sketch de la aplicación desarrollada en Arduino mostrando mensajes de alerta a través de la comunicación serial de Arduino; y un programa analizador de red (Wireshark). Si la comunicación Ethernet es correcta se visualizarán los resultados en la interfaz gráfica de la aplicación desarrollada en Visual Studio. Además, toda la información presentada en las ventanas de texto se guardará en documentos de texto plano en formato TXT, como registros para que el usuario tenga toda la información recolectada por los sensores para un posterior análisis si así lo desea.

3.1. DISEÑO DE LA RED LAN: 1 SERVIDOR – 2 CLIENTES

Considerando que se tiene listo el diseño de la Red LAN: 1 Servidor – 2 Clientes del acápite 2.3.2, es necesario configurar la tarjeta de Red del computador previamente para que se puede tener una conexión cliente – servidor exitosa. Siendo el computador usado como servidor, se considera que la dirección IP que debe tener su tarjeta de Red es la que el programador determinó para el diseño en la Tabla 2.6. Se indicará los pasos para la configuración de la tarjeta de Red:

1. Ingresar al “Panel de control” de Windows.
2. Seleccionar “Redes e Internet”.

En la Figura 3.1 se indica la imagen para ingresar al menú “Redes e Internet”.

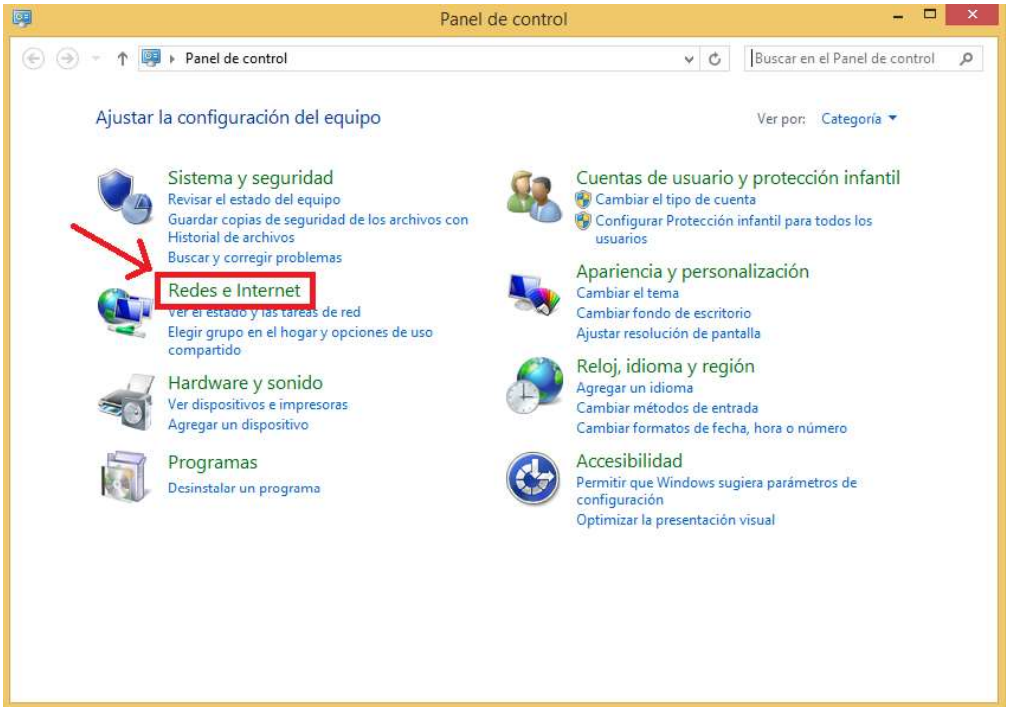

Figura 3.1. Ingresar al Menú “Redes e Internet”.

3. Ingresar en “Centro de Redes y Recursos Compartidos”.

En la Figura 3.2 se indica la imagen para ingresar al submenú “Centro de Redes y Recursos Compartidos”.

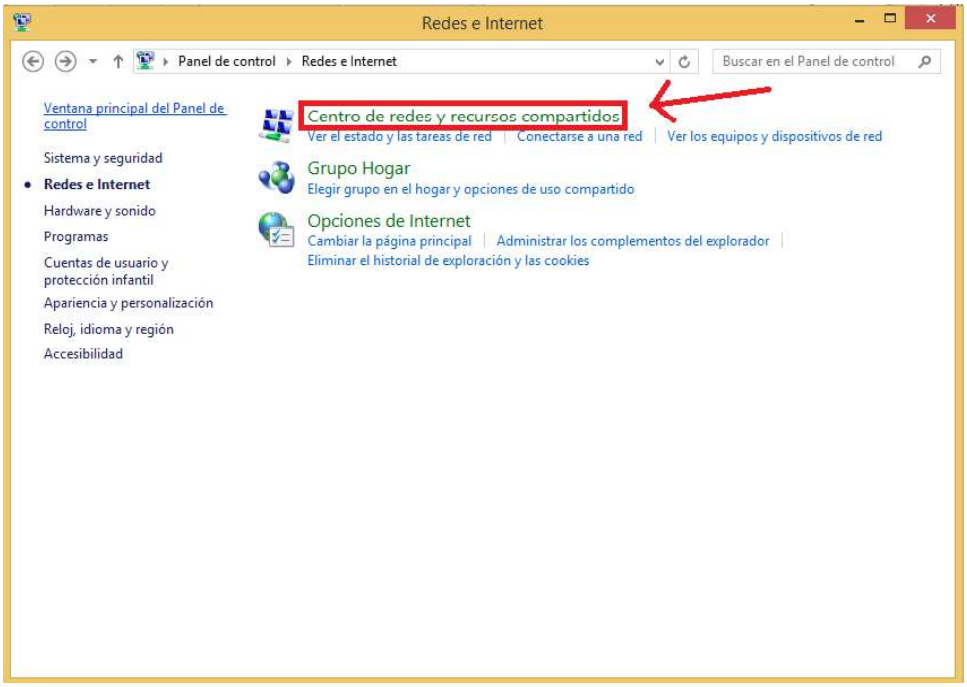

Figura 3.2. Ingresar al Submenú “Centro de Redes y Recursos Compartidos”.

4. Seleccionar la opción “Cambiar configuración del adaptador”.

En la Figura 3.3 se indica la imagen para ingresar a la opción “Cambiar configuración del adaptador”.

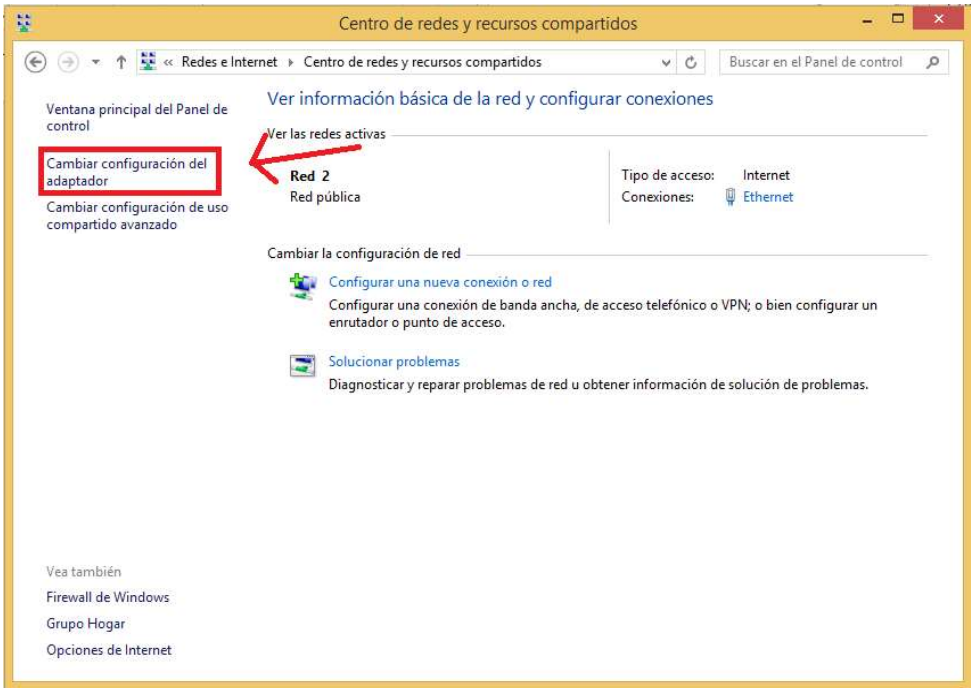

Figura 3.3. Ingresar a la Opción “Cambiar configuración del adaptador”.

5. Buscar y dar clic derecho sobre la tarjeta de Red que utiliza el computador para seleccionar la opción “Propiedades”, como se muestra en la Figura 3.4.

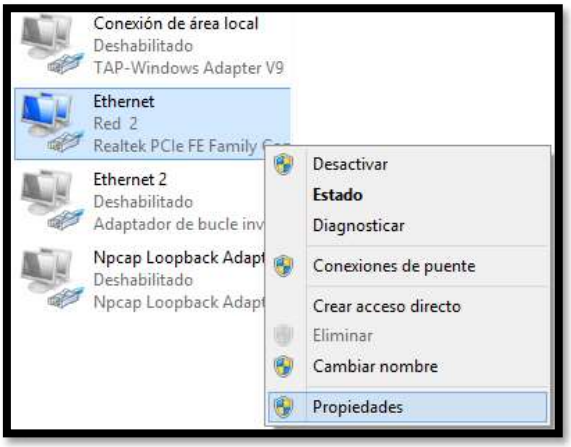

Figura 3.4. Ingresar a las propiedades de la tarjeta de Red.

6. Seleccionar “Protocolo de Internet versión 4 (TCP/IPv4)” y dar clic en el botón “Propiedades” para desplegar la ventana de configuración de la tarjeta de Red.

En la Figura 3.5 se indica la imagen de las propiedades de la tarjeta de Red.

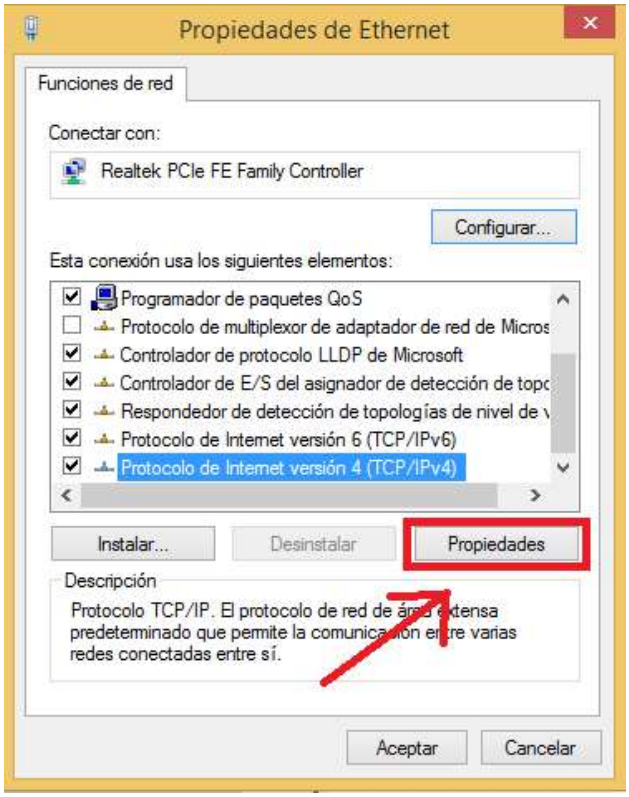

Figura 3.5. Propiedades de la tarjeta de Red.

7. Configurar el servidor con la dirección IP y máscara de subred de la Tabla 2.6.

En la Figura 3.6 se indica la imagen de configuración del servidor.

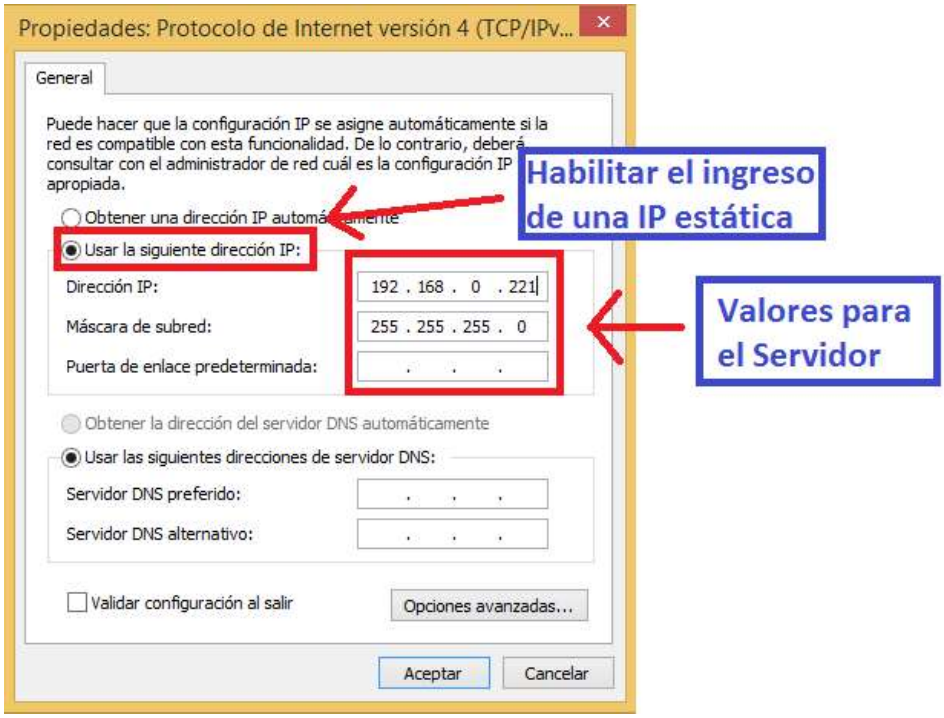

Figura 3.6. Configuración del Servidor.

3.1.1 PRUEBA DE CONECTIVIDAD ETHERNET

Configurada la dirección IP y la máscara de subred de la tarjeta de Red del computador (Servidor) se tiene la Red LAN completa y lista para funcionar. Durante la realización del proyecto, se pueden encontrar errores de tipeo en las instrucciones que errores en conexiones físicas. Se determinará a través de diferentes mecanismos de apoyo si existe una conexión Ethernet exitosa a pesar de que los datos no se lleguen a visualizar en la aplicación desarrollada como servidor creada en Visual Studio. Esto se lleva a cabo para poder discretizar los errores y poder aplicar las soluciones pertinentes al proyecto.

Con lo expresado en el párrafo anterior, hay que conectar el computador con la aplicación de Visual Studio al Arduino por comunicación Ethernet; se abre la aplicación llamada servidor que debe mostrarse como la Figura 2.32 y se da clic en el botón “CONECTAR”. Esto iniciará la comunicación Ethernet entre el Servidor y el Cliente para comprobar si existe una conexión exitosa se procede a verificar en los mecanismos de apoyo.

3.1.1.1 Ping

"La comprobación por el comando Ping sirve para hacer un diagnóstico a la Red desde una dirección IP hacia otra dirección IP verificando la accesibilidad en dicha Red entre ambas direcciones [59]." Si el comando Ping es exitoso, ambos dispositivos (cliente y servidor) están en la misma Red y pueden transferirse información de una terminal a otra. Hay que considerar que tener un Ping exitoso no determina que la aplicación funcione, pero sí asegura que ambos dispositivos están configurados en la misma Red y tienen la capacidad de transferirse datos. Para poder ejecutar el comando Ping es necesario saber la dirección IP de la otra terminal, en este caso del Arduino que se puede encontrar en la Tabla 2.6.

En la Figura 3.7 se indica la imagen de la ejecución de un comando Ping exitoso.

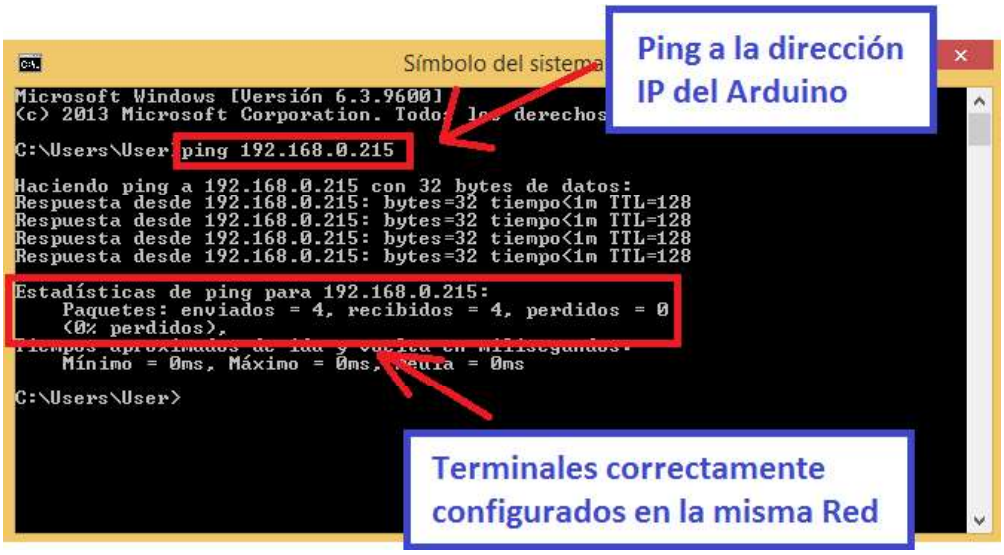

Figura 3.7. Ejecución de Comando Ping exitoso.

3.1.1.2 Comunicación Serial

El uso de este mecanismo es la inclusión de instrucciones de apoyo dentro del Sketch de la aplicación en Arduino que no afectan al programa principal, pero permiten al programador detectar si se obtuvo una conexión Ethernet exitosa o no. Estas instrucciones devolverán un valor por medio del "Monitor Serial" de Arduino al programador. Las cuales dirán si hay o no una conexión entre el cliente y el servidor.

En la Figura 3.8 se indica la imagen del monitor serial Arduino.

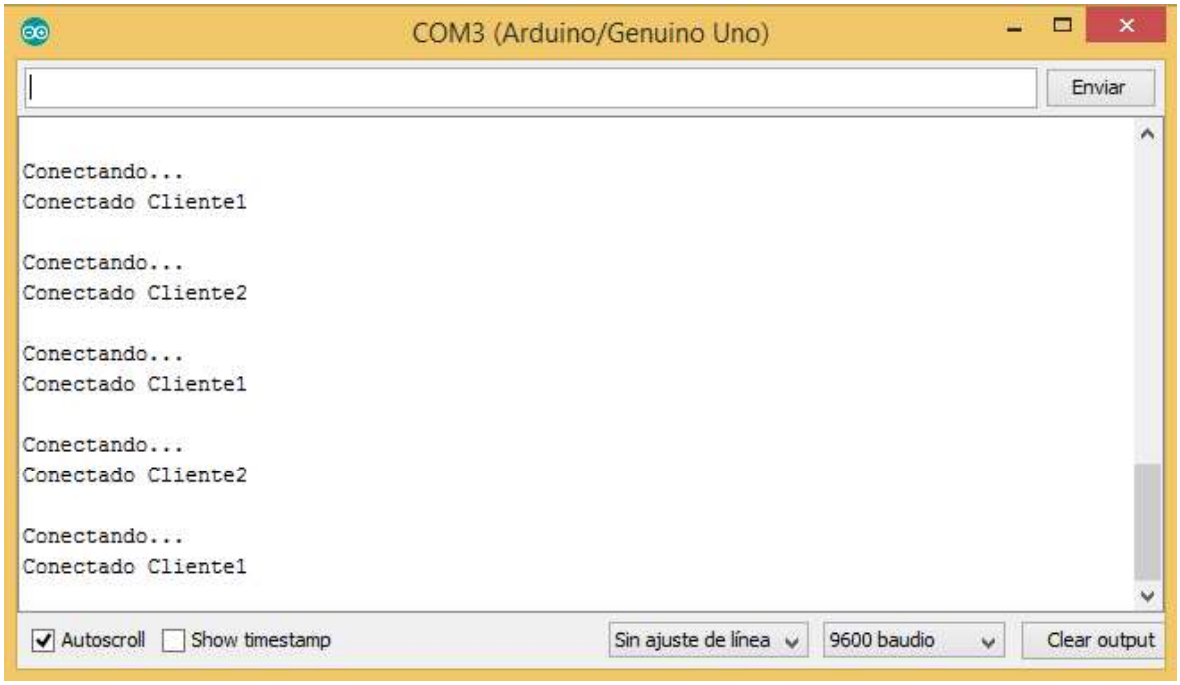

Figura 3.8. Monitor Serial Arduino conexión Ethernet.

Estas instrucciones de apoyo son simples y van intercaladas entre las instrucciones del programa principal del Sketch de Arduino como se muestra en el Anexo B, donde el programador sabe que debe existir una conexión Ethernet exitosa. Estas instrucciones son mensajes de anuncio por comunicación serial como, por ejemplo:

```
Serial.println("Conectado Cliente1");
```

La instrucción ***"Serial.println ()"*** es un comando de Arduino que imprime lo que está dentro del paréntesis con un salto de línea al final, puede imprimir valores de variables o si se requiere imprimir texto plano se usan comillas donde Arduino lo interpretará como formato ASCII a través del puerto serie.

3.1.1.3 Wireshark

"Wireshark es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes, incluye un completo lenguaje para filtrar lo que se quiere ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP [60]." Es una herramienta desarrollada en software libre, muy útil para que el programador puede interpretar cada una de las tramas enviadas y las tramas recibidas. Esta herramienta de apoyo da mucha

información al programador para entender de una mejor manera que sucede en cada momento de la Red.

Después de haber comprobado con los dos mecanismos anteriores que se tiene una conexión Ethernet exitosa lo último que se debe comprobar es que se esté usando el protocolo TCP y el uso de sockets para la comunicación Ethernet. Esto dirá al programador que se consiguió el objetivo planteado del proyecto.

En la Figura 3.9 se indica la imagen un ejemplo de trama en Wireshark.

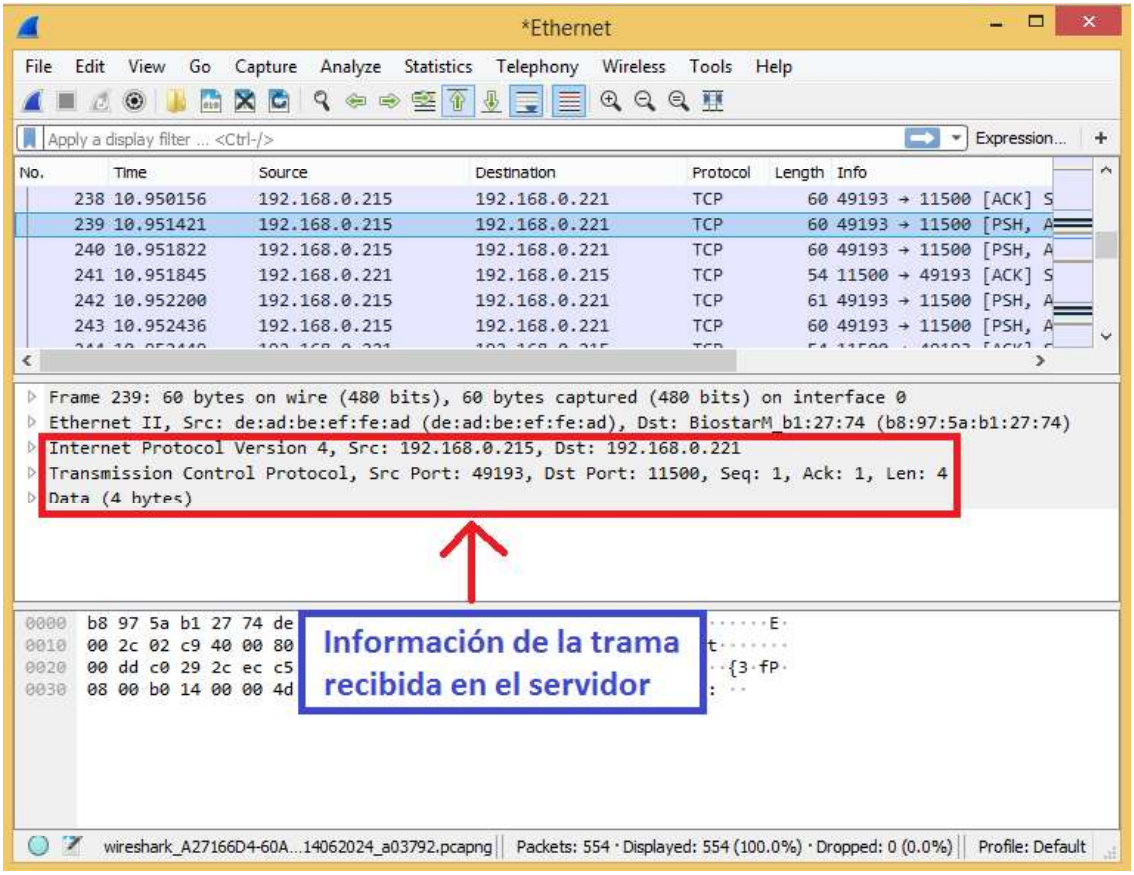

Figura 3.9. Ejemplo de trama en Wireshark.

3.1.2 PRUEBA DE FUNCIONAMIENTO

Las pruebas de funcionamiento de la librería se realizan verificado el funcionamiento de la aplicación desarrollada como servidor. En primer término, se establece la conexión que se comprueba con el uso de los mecanismos descritos en el acápite 3.1.1. La comunicación se inicia seleccionando el botón “CONECTAR” y la ventana de texto del servidor muestra los datos obtenidos del sensor como se observa en la Figura 3.10 cuando se ha obtenido una conexión exitosa. Las ventanas de texto presentan la

información de forma clara y concisa, los datos corresponden a los registrados o leídos en forma directa de los sensores. Los errores en la transmisión son solventados por los mecanismos inherentes al protocolo TCP.

En la Figura 3.10 se indica la imagen de la presentación de la información en la aplicación.

Figura 3.10. Presentación de la información en la Aplicación.

La Figura 3.10 muestra la posición espacial del magnetómetro a través de su ubicación en 3 dimensiones y por otro lado el barómetro muestra la temperatura, la presión y la altitud que tiene en ese momento el sensor.

3.2. DISEÑO DE LA RED LAN: 2 SERVIDORES – 2 CLIENTES

Considerando que se tiene listo el diseño de la Red LAN: 2 Servidores – 2 Clientes del acápite 2.3.3, es necesario configurar previamente la tarjeta de Red de los computadores para que se puede tener una conexión cliente – servidor exitosa. Siendo el computador usado como servidor se considera que la dirección IP, que debe tener su tarjeta de Red, es la que el programador determinó para el diseño como se muestra en la Tabla 2.7. Así, se seguirán los mismos pasos para la configuración de la tarjeta de Red del acápite 3.1 para ambos servidores, es importante no olvidar de configurar la puerta de enlace en

cada una de las tarjetas de Red debido a que en este diseño se utiliza un router domestico como un switch.

3.2.1 PRUEBA DE CONECTIVIDAD ETHERNET

Configurada la dirección IP, la máscara de subred y la puerta de enlace de la tarjeta de Red de los computadores (Servidores) se tiene la Red LAN completa y lista para operar. Durante la realización del proyecto se pueden encontrar errores de tipeo en las instrucciones antes que errores en conexiones físicas. Se determinará a través de diferentes mecanismos de apoyo si existe una conexión Ethernet exitosa a pesar de que los datos no se lleguen a visualizar en las aplicaciones desarrolladas como magnetómetro y barómetro en Visual Studio. Esto se lleva a cabo para poder discretizar los errores y poder aplicar las soluciones necesarias al proyecto.

Con lo expresado en el párrafo anterior, hay que conectar los computadores que contienen las aplicaciones desarrolladas en Visual Studio al Arduino por comunicación Ethernet a través de un router. Para lo cual, se abren las aplicaciones magnetómetro y barómetro en cada computador como se muestra en la Figura 2.39 y la Figura 2.40 respectivamente; y para comenzar la comunicación Ethernet se da clic en el botón “CONECTAR”. Esto iniciará la comunicación Ethernet entre el Servidor y el Cliente que corresponde, para comprobar si existe una conexión exitosa se procede a verificar en los mecanismos de apoyo:

1. **Comando Ping**: Este comando se utiliza de la misma forma descrita en el acápite 3.1.1.1 utilizando la información de la Tabla 2.7 para identificar a los servidores y clientes.
2. **Comunicación Serial**: se procede de la misma manera como lo descrito en el acápite 3.1.1.2 donde el resultado debe ser idéntico, en esta prueba se utiliza la aplicación desarrollada en Arduino que se detalla en el Anexo C.
3. **Wireshark**: de igual manera como se describe en el acápite 3.1.1.3 se verifica que para la comunicación Ethernet el protocolo TCP esté presente y se conecte el cliente al servidor utilizando sockets.

3.2.2 PRUEBA DE FUNCIONAMIENTO

Como se describe en el acápite 3.1.2, las pruebas de funcionamiento de la librería se realizan verificado el funcionamiento en las aplicaciones desarrolladas magnetómetro y barómetro. En primer lugar, se establece la conexión que se comprueba con el uso de los mecanismos descritos en el acápite 3.2.1. La comunicación se inicia al seleccionar el botón “CONECTAR” en las ventanas del magnetómetro y del barómetro mostrando los datos obtenidos por cada sensor como se observa en la Figura 3.11 y la Figura 3.12 cuando se ha obtenido una conexión exitosa. Las ventanas de texto presentan la información de una forma comprensible a usuario, los datos corresponden a los registrados o leídos en forma directa de los sensores. Los errores en la transmisión son solventados por los mecanismos inherentes al protocolo TCP.

En la Figura 3.11 se indica la imagen de la presentación de la información en la aplicación magnetómetro.

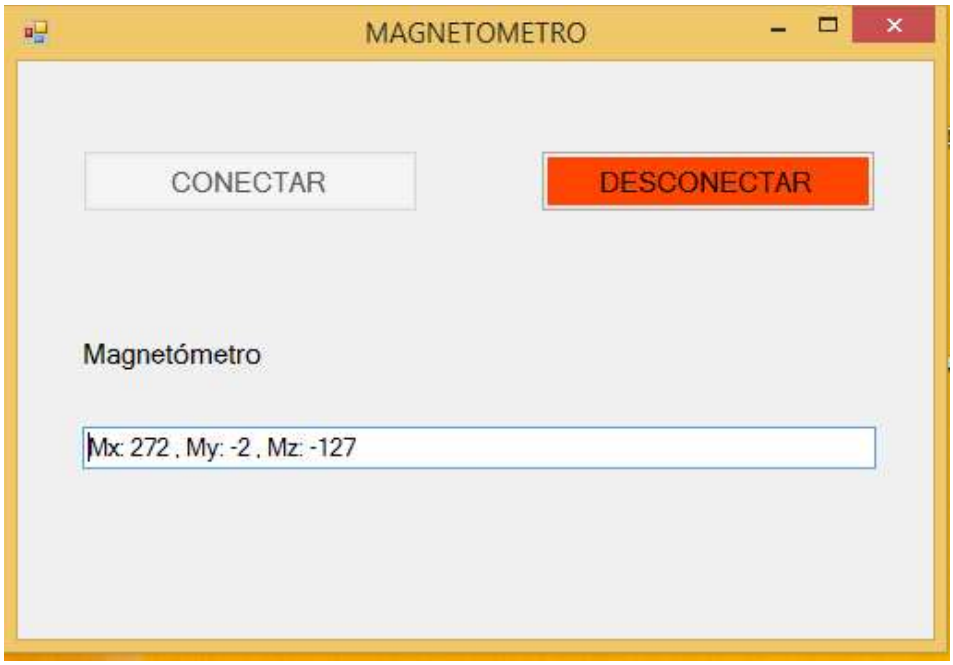

Figura 3.11. Presentación de la información en la Aplicación Magnetómetro.

En la Figura 3.12 se indica la imagen de la presentación de la información en la aplicación barómetro.

Figura 3.12. Presentación de la información en la Aplicación Barómetro.

Como adicional, se incluye en el Anexo D la prueba de funcionamiento del encendido de un LED para comprobar la recepción de datos en la plataforma Arduino, por lo cual se observa la capacidad de la librería para manejar una red de sensores y actuadores de manera simultánea.

3.3. REGISTRO DE DATOS

Las diferentes aplicaciones desarrolladas en Visual Studio de este proyecto tienen la capacidad de almacenar grandes cantidades de información recolectada por cada sensor en un documento de texto plano en formato TXT. Estos registros o históricos son útiles para generar una base de datos y posteriormente analizar en conjunto la información que se obtuvo durante un cierto periodo de tiempo, como por ejemplo ¿Por qué existió un descenso brusco de la temperatura? Los registros también ayudan al usuario a no estar presente durante la recolección de datos debido a que toda esta información queda almacenada de manera automática.

En la Figura 3.13 se indica la imagen del registro de datos del magnetómetro.

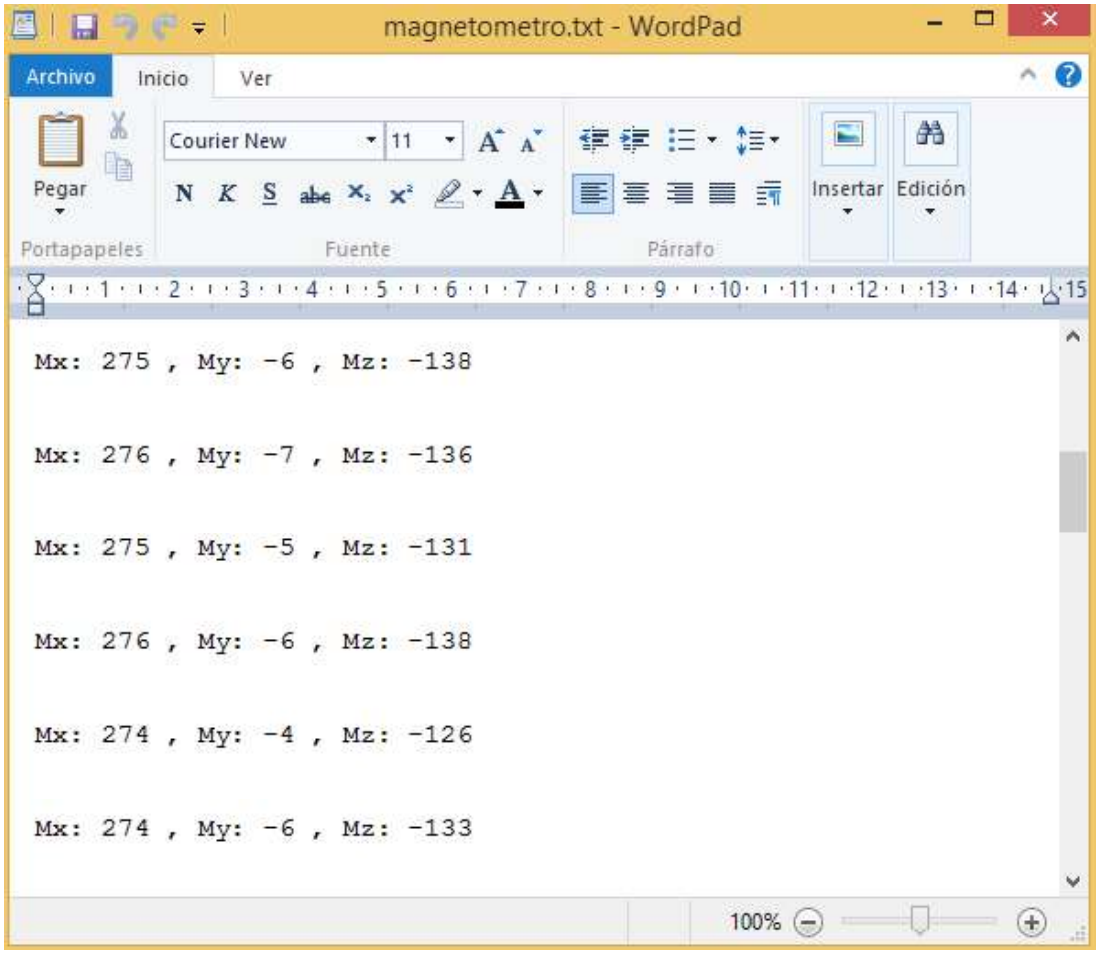

Figura 3.13. Registro de datos del Magnetómetro.

En la Figura 3.14 se indica la imagen del registro de datos del barómetro.

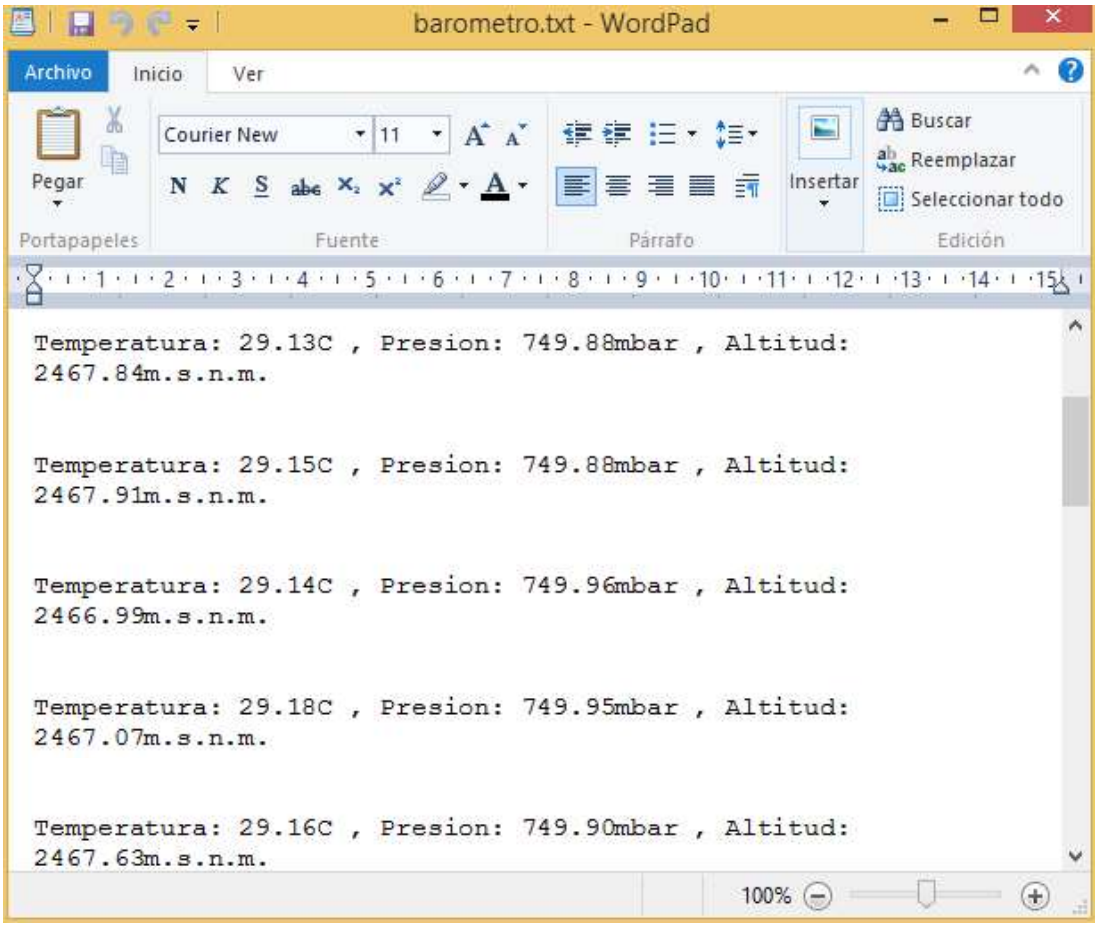

Figura 3.14. Registro de datos del Barómetro.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El propósito de este trabajo de titulación es el desarrollo de una librería Ethernet que permita comunicar los periféricos de la plataforma Arduino hacia una aplicación orientada a IOT haciendo uso de sockets para que los datos no puedan ser accedidos por un tercero de una manera muy sencilla en el caso de utilizar un puerto conocido como es el puerto 80 que es del servicio de HTTP.

- El chip Wiznet W5100 tiene la capacidad de usar cuatro conexiones Ethernet simultáneas sin embargo la librería Ethernet 1.1.2. está diseñada para solventar una conexión a la vez. Esto lo hace leyendo únicamente el puerto y no el socket, es decir si se tiene dos sockets diferentes que usan el mismo puerto, ira al primer espacio de memoria donde esta guardado el primer socket. Este inconveniente es resuelto en el presente proyecto incluyendo matrices de almacenamiento donde se lee y se compara no solo el puerto sino el socket al cual se dirige cada conexión Ethernet para así ocupar más de una conexión de las cuatro posibles que ofrece el chip Wiznet W5100.

- La librería Ethernet 1.1.2 busca al socket por el puerto asignado dentro del registro de memoria del chip Wiznet W5100 cuando la plataforma Arduino es usada como servidor siendo necesario, agregar una nueva matriz para almacenar el puerto asociado a un socket cuando se usa como cliente.

- Para poder solucionar el error de la búsqueda de los sockets se agregan variables que alberguen y comparen si el puerto está asociado a la dirección IP, si no es el socket que se busca se salta al siguiente registro de sockets de la memoria del chip Wiznet W5100.

- Usar el protocolo TCP implica usar una conexión orientada a conexión, debido a que TCP es un protocolo estable y confiable. Incluyendo el uso de un socket por cada periférico conectado al Arduino, el programador está diseñando una comunicación punto a punto entre un dispositivo cliente y un servidor.

- La plataforma Arduino puede ser usada como servidor o como cliente, sin embargo, el Arduino al estar dedicado exclusivamente a la recolección y al envío de información la manera más apropiada de usar dicha plataforma es como cliente.

- El uso de sesiones simultaneas consumen mucha memoria y la capacidad de procesamiento se ve afectada por lo cual se ve que es necesario cerrar dichas sesiones para no saturar al programa antes de capturar la siguiente información de los sensores.
- Es necesario tener varios mecanismos de apoyo en diferentes niveles del diseño para verificar si existe o no una correcta comunicación Ethernet, debido a que puede haber errores tanto de hardware como de software y es útil aislar los problemas para poder dar una solución enfocada según sea el caso.
- El uso del comando ping ayuda a comprobar que es posible la transferencia de datos entre el servidor y el cliente, si el comando ping es fallido determina la existencia de un error a nivel de hardware o que ambos dispositivos están en diferentes reden LAN.
- El incluir instrucciones de apoyo en el sketch de Arduino solo como una forma de verificar la conexión Ethernet es más habitual de lo que parece, la inclusión o no de estas instrucciones no altera el programa más bien sirven de ayuda al programador a determinar en cada punto del programa si su ejecución fue correcta. Estas instrucciones usan la comunicación serial de Arduino para informar si es correcta la ejecución del programa en cierto punto de este y verificar que se está transmitiendo las tramas Ethernet entre el servidor y el cliente.
- El uso de un analizador de red como Wireshark es fundamental a la hora de comprobar que todo lo descrito en este trabajo de titulación está siendo ejecutado. Wireshark proporciona todos los resultados de trama en trama para que el programador verifique en cada uno de sus campos si se está aplicando los protocolos, los sockets y direcciones IP de forma correcta. Así como, la desconexión entre el servidor y el cliente en el caso de existir.
- El uso de sensores implica que la plataforma Arduino debe enviar los datos hacia el aplicativo de Visual Studio y por otro lado el uso de actuadores es la recepción de datos en la plataforma Arduino.
- El encendido y apagado de un LED en la plataforma Arduino demuestra la capacidad del mismo y de la librería Ethernet en poder no solo enviar datos sino también recibirlos, para posteriormente manejar el o los actuadores que el usuario requiera.

4.2 RECOMENDACIONES

- Para optimizar este trabajo de titulación sería prudente diseñarlo para una comunicación Ethernet Full Dúplex dándole versatilidad y un mejor uso como dispositivo IOT.

- El uso del protocolo TCP y el uso de sockets en el diseño otorgan una seguridad mínima a los datos, es recomendable mejorar los mecanismo de seguridad.

- Revisar las actualizaciones de las diferentes librerías en Arduino antes de elaborar un proyecto porque constantemente van cambiando, ya que detrás de dicha plataforma existe un grupo de desarrolladores trabajando en la mejora de Arduino y todos sus componentes.

- Usar la librería Ethernet 2.0.0 (creada el 26 de Julio de 2018) que apareció después de la aprobación del plan de titulación en lugar de la nueva librería Ethernet que se propone en este trabajo de titulación que fue basado en la librería Ethernet 1.1.2 (creada el 30 de Agosto de 2016). La librería Ethernet 2.0.0. viene con más mejoras de las desarrolladas en este trabajo como, por ejemplo, mejora el uso de los espacios de memoria y los tiempos de búsqueda en las diferentes matrices, optimizando un mejor desempeño.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. Peña y G. Suquillo, «Estudio del modelo de referencia del internet de las cosas (IOT), con la implementación de un prototipo domótico,» Marzo 2016. [En línea]. Available: https://bibdigital.epn.edu.ec/bitstream/15000/15096/1/CD-6908.pdf. [Último acceso: 2019 Junio 26].

[2] G. Toazo y M. Vega, «Prototipo de una gaveta experimental basada en IOT (internet de las cosas),» Julio 2017. [En línea]. Available: https://bibdigital.epn.edu.ec/bitstream/15000/17519/1/CD-8024.pdf. [Último acceso: 26 Junio 2019].

[3] G. Tojeiro Calaza, Taller de Arduino Un Enfoque Práctico Para Principiantes, Marcombo S.A., 2014.

[4] F. J. Ceballos Sierra, Visual Basic Interfaces Graficas y Aplicaciones para Internet con WPF, WCF y Silverlight, Alfaomega Grupo Editor S.A., 2013.

[5] J. Blum, Arduino a Fondo, Grupo Anaya S.A., 2014.

[6] D. Evans, «Internet of Things La próxima evolución de Internet lo está cambiando todo,» Abril 2011. [En línea]. Available: https://www.cisco.com/c/dam/global/es_es/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf. [Último acceso: 21 Junio 2018].

[7] O. Torrente Artero, El Mundo Genuino Arduino Curso Práctico de Formación, Alfaomega Grupo Editor S.A., 2016.

[8] «Que es IOT,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/2017/03/29/que-es-iot/. [Último acceso: 21 Junio 2020].

[9] M. Halvorson, Visual Basic 2008 Paso a Paso, Grupo Anaya S.A., 2009.

[10] CepymeNews, «Caracteristicas y usos del internet de las cosas,» 04 Julio 2018. [En línea]. Available: https://cepymenews.es/caracteristicas-usos-internet-cosas. [Último acceso: 10 Junio 2019].

[11] M. Alcaraz, «Internet de las Cosas,» Octubre 2014. [En línea]. Available: http://jeuazarru.com/wp-content/uploads/2014/10/Internet-of-Things.pdf. [Último acceso: 10 Junio 2019].

[12] S. J. Brau, «De qué manera Internet of Things está cambiando nuestra forma de trabajar y vivir,» 2019. [En línea]. Available: http://sebastianbrau.com/de-que-manera-internet-things-esta-cambiando-nuestra-forma-de-trabajar-y-vivir/. [Último acceso: 26 Junio 2019].

[13] TechTarget, «internet of things (IoT),» 13 Marzo 2019. [En línea]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Último

acceso: 10 Junio 2019].

[14] Wikimedia Commons, «Visual Studio 2017 logo,» 06 Julio 2017. [En línea]. Available: https://commons.wikimedia.org/wiki/File:Visual_Studio_2017_logo_and_wordmark.svg. [Último acceso: 10 Junio 2019].

[15] Microsoft, «Le damos la bienvenida al IDE de Visual Studio | Visual Basic,» 14 Noviembre 2018. [En línea]. Available: https://docs.microsoft.com/es-es/visualstudio/get-started/visual-basic/visual-studio-ide?view=vs-2017. [Último acceso: 26 Junio 2019].

[16] Wikipedia, «Arduino,» 10 Junio 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Arduino. [Último acceso: 10 Junio 2019].

[17] R. Enriquez Herrador, «Guía de usuario de Arduino,» 13 Noviembre 2009. [En línea]. Available: http://electroship.com/documentos/Arduino_user_manual_es.pdf. [Último acceso: 11 Junio 2019].

[18] Red Hat, «What is an Arduino?,» 2019. [En línea]. Available: https://opensource.com/resources/what-arduino. [Último acceso: 11 Junio 2019].

[19] Reichelt Elektronik, «Arduino UNO DIP,» 01 Junio 2019. [En línea]. Available: https://www.reichelt.com/de/en/arduino-uno-rev-3-dil-version-atmega328-usb-arduino-uno-dip-p154902.html. [Último acceso: 11 Junio 2019].

[20] Arduino, «Arduino UNO Rev3,» 2019. [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 11 Junio 2019].

[21] JSumo, «Arduino Ethernet Shield,» 2019. [En línea]. Available: https://www.jsumo.com/arduino-ethernet-shield. [Último acceso: 11 Junio 2019].

[22] AprendiendoArduino, «Ethernet Shield,» 04 Julio 2016. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/07/04/ethernet-shield/. [Último acceso: 11 Junio 2019].

[23] Arduino, «Arduino Ethernet Shield Without PoE Module,» 2019. [En línea]. Available: https://store.arduino.cc/usa/arduino-ethernet-shield-without-poe-module. [Último acceso: 11 Junio 2019].

[24] IONOS, «¿Qué es Ethernet (IEEE 802.3)?,» 15 Agosto 2018. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/ethernet-ieee-8023/. [Último acceso: 11 Junio 2019].

[25] AprendiendoArduino, «Sensores y Actuadores,» 18 Diciembre 2016. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/. [Último acceso: 12 Junio 2019].

[26] Digital Micro Devices, «I2C,» 2019. [En línea]. Available: https://d3.xlrs.eu/conectividad/i2c/. [Último acceso: 17 Junio 2019].

[27] A. M. Fernández, «El Bus I2C,» Abril 2004. [En línea]. Available: http://www.uco.es/~el1mofer/Docs/IntPerif/Bus%20I2C.pdf. [Último acceso: 17 Junio 2019].

[28] HowToMechatronics.com, «How I2C Communication Works and How To Use It with Arduino,» 2019. [En línea]. Available: https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/. [Último acceso: 17 Junio 2019].

[29] DroneBot Workshop, «I2C Communications Part 1 – Arduino to Arduino,» 2019. [En línea]. Available: https://dronebotworkshop.com/i2c-arduino-arduino/. [Último acceso: 17 Junio 2019].

[30] L. LLamas, «El bus I2C en Arduino,» 18 Mayo 2016. [En línea]. Available: https://www.luisllamas.es/arduino-i2c/. [Último acceso: 17 Junio 2019].

[31] EcuRed, «Protocolos de Red,» 29 Enero 2012. [En línea]. Available: https://www.ecured.cu/Protocolos_de_red. [Último acceso: 20 Agosto 2019].

[32] IETF.org, «RFC793,» Septiembre 1981. [En línea]. Available: https://tools.ietf.org/html/rfc793. [Último acceso: 20 Agosto 2019].

[33] A. Calveras, «El protocolo TCP,» Noviembre 1999. [En línea]. Available: https://www.tdx.cat/bitstream/handle/10803/7040/04AMCA04de15.pdf;sequence=4. [Último acceso: 20 Agosto 2019].

[34] IETF.org, «RFC6335,» Agosto 2011. [En línea]. Available: https://tools.ietf.org/html/rfc6335#page-11. [Último acceso: 20 Agosto 2019].

[35] WIZnet Co., «W5100 Datasheet,» 2011. [En línea]. Available: https://www.wiznet.io/wp-content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.7.pdf. [Último acceso: 30 Julio 2019].

[36] AprendiendoArduino, «Memoria Flash, SRAM y EEPROM,» 21 Junio 2017. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2017/06/21/memoria-flash-sram-y-eeprom-3/. [Último acceso: 04 Julio 2019].

[37] Diffzi, «Point-to-point Connection vs. Multipoint Connection,» 2019. [En línea]. Available: https://diffzi.com/point-to-point-connection-vs-multipoint-connection/. [Último acceso: 01 Agosto 2019].

[38] Arduino, «Download Arduino,» 2019. [En línea]. Available: https://www.arduino.cc/en/Main/OldSoftwareReleases#previous. [Último acceso: 05 Agosto 2019].

[39] GitHub, Inc., «Ethernet-1.1.2.,» 30 Agosto 2016. [En línea]. Available: http://downloads.arduino.cc/libraries/github.com/arduino-libraries/Ethernet-1.1.2.zip. [Último acceso: 05 Agosto 2019].

[40] Alldatasheet.com, «HMC5883L Datasheet (PDF) - Honeywell Solid State Electronics Center,» Octubre 2010. [En línea]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/428790/HONEYWELL/HMC5883L.html. [Último acceso: 01 Agosto 2019].

[41] RAM Electronics, «Compass Module (3 Axis Magnetometer) – HMC5883L,» 2019. [En línea]. Available: https://ram-e-shop.com/product/kit-hmc5883l-compass/. [Último acceso: 01 Agosto 2019].

[42] Alldatasheet.com, «BMP180 Datasheet (PDF) - Bosch Sensortec GmbH,» [En línea]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132068/BOSCH/BMP180.html. [Último acceso: 01 Agosto 2019].

[43] Alibaba, «GY-68 300-1100hPa BMP180 Sensor de temperatura Módulo Sensor de presión atmosférica,» 2019. [En línea]. Available: https://spanish.alibaba.com/product-detail/gy-68-300-1100hpa-bmp180-temperature-sensor-module-atmospheric-pressure-sensor-62202756289.html. [Último acceso: 01 Agosto 2019].

[44] V. Ferrer, «Que es una Protoboard o Breadboard,» 06 Mayo 2019. [En línea]. Available: https://vicentferrer.com/protoboard-breadboard/. [Último acceso: 20 Agosto 2019].

[45] IBEROBOTICS, «Placa Prototipos Protoboard Breadboard 400 puntos,» 2019. [En línea]. Available: https://www.iberobotics.com/producto/placa-prototipos-protoboard-breadboard-400-puntos/. [Último acceso: 20 Agosto 2019].

[46] Wikipedia, «Cable Puente,» 08 Junio 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Cable_puente. [Último acceso: 20 Agosto 2019].

[47] Natytec, «40 Jumper Cables Dupont Macho Macho,» 2018. [En línea]. Available: https://natytec.com.mx/CNC/40-jumper-cables-dupont-macho-macho/. [Último acceso: 20 Agosto 2019].

[48] B. Mitchell, «Ethernet Cables and How They Work,» 29 Agosto 2019. [En línea]. Available: https://www.lifewire.com/what-is-an-ethernet-cable-817548. [Último acceso: 30 Agosto 2019].

[49] DHgate.com, «RJ45 CAT5 Ethernet Cable macho a macho,» 2019. [En línea]. Available: https://es.dhgate.com/product/wholesale-20cm-rj45-cat5-ethernet-cable-male/406852617.html. [Último acceso: 20 Agosto 2019].

[50] M. Peña, «Te explicamos cómo elegir el mejor cable de Ethernet,» 04 Abril 2019. [En línea]. Available: https://es.digitaltrends.com/computadoras/como-elegir-el-mejor-cable-de-ethernet/. [Último acceso: 30 Agosto 2019].

[51] J. D. de Usera, «Conectores USB: tipos y para qué sirve cada uno,» 30 Junio 2018. [En línea]. Available: https://hardzone.es/2018/06/30/conectores-usb-tipos/. [Último acceso: 30 Agosto 2019].

[52] MechatronicStore, «Cable USB tipo B para Arduino 30cm,» 2019. [En línea]. Available: https://www.mechatronicstore.cl/adaptadores-y-shields/cable-usb-tipo-b-para-arduino-

30cm/. [Último acceso: 30 Agosto 2019].

[53] www.quetalcompra.com, «Genérico Cargador USB de carga rápida 2A,» 2018. [En línea]. Available: https://www.quetalcompra.com/Gen%C3%A9rico-Cargador-USB-de-carga-r%C3%A1pida-2A-p10534.html. [Último acceso: 30 Agosto 2019].

[54] SAMSUNG, «Cargador USB 3.0,» 2019. [En línea]. Available: https://www.samsung.com/es/mobile-accessories/travel-adapter-micro-usb-3-0-ta10/EP-TA10EWEQGWW/. [Último acceso: 30 Agosto 2019].

[55] GitHub, Inc., «Built for developers,» 2019. [En línea]. Available: https://github.com/. [Último acceso: 01 Septiembre 2019].

[56] GitHub, Inc., «i2cdevlib-master,» 2019. [En línea]. Available: https://github.com/jrowberg/i2cdevlib/archive/master.zip. [Último acceso: 01 Septiembre 2019].

[57] GitHub, Inc., «BMP180_Breakout-master,» 2019. [En línea]. Available: https://github.com/sparkfun/BMP180_Breakout/archive/master.zip. [Último acceso: 01 Septiembre 2019].

[58] Microsoft, «Visual Studio 2017,» 2019. [En línea]. Available: https://my.visualstudio.com/Downloads?q=Visual%20Studio%202017. [Último acceso: 10 Septiembre 2019].

[59] Zator Systems, «Apéndice C. Packet Internet Groper,» 2016. [En línea]. Available: https://www.zator.com/Internet/X_Ap_C.htm. [Último acceso: 2019 Septiembre 30].

[60] Wireshark.org, «Wireshark,» 2019. [En línea]. Available: https://www.wireshark.org/. [Último acceso: 01 Octubre 2019].

ANEXOS

La librería Ethernet de este proyecto de titulación está basada en la librería Ethernet 1.1.2.

ANEXO A. Archivos .cpp y .h de la librería Ethernet

1. Archivo Ethernet.h
2. Archivo Ethernet.cpp
3. Archivo EthernetClient.h
4. Archivo EthernetClient.cpp
5. Archivo EthernetServer.cpp

ANEXO B. Diseño de la Red LAN: 1 Servidor – 2 Clientes

1. Sketch de la aplicación en Arduino
2. Código de la aplicación en Visual Studio

ANEXO C. Diseño de la Red LAN: 2 Servidores – 2 Clientes

1. Sketch de la aplicación en Arduino
2. Código de la aplicación en Visual Studio (Magnetómetro)
3. Código de la aplicación en Visual Studio (Barómetro)

ANEXO D. Diseño de la Red LAN con el encendido de 1 LED

1. Sketch de la aplicación en Arduino
2. Código de la aplicación en Visual Studio (LED)

Nota: debido a la extensión, los anexos se han incluido en el disco compacto adjunto a este documento.

ORDEN DE EMPASTADO