UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO

COLEGIO DE CIENCIA Y TECNOLOGÍA

Diseño e implementación de una red inalámbrica de sensores

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

P R E S E N T A :

YOLANDA LILIBETH RIVERA BIBIANO

DIRECTOR DE TESIS

M. en I. OSCAR RENÉ VALDEZ CASILLAS

Ciudad de México, noviembre de 2017.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

"Cada día ve a la humanidad un poco más victoriosa en la lucha con el tiempo y el espacio."

Guglielmo Marconi

Agradecimientos

Mi total agradecimiento a la Universidad Autónoma de la Ciudad de México y todas las personas que me dieron su apoyo para que este proyecto sea posible: a mis padres, al profesor Oscar Valdez, mis lectores, mi amiga Melissa y a mi Daniel.

Dedicatoria

Dedico este trabajo a mi hermano Manuel.

Objetivo General

Diseñar e implementar una red inalámbrica de sensores con base en el estándar IEEE 802.15.4 bajo la topología estrella e incluir su interfaz de software para la lectura remota de los datos obtenidos con los sensores.

Objetivos específicos

- Implementar una red inalámbrica en topología estrella con cuatro nodos transmisores a los que se les denomina Nodo Sensor Básico (BSN) y un nodo receptor llamada Nodo Agregador (AN) usando el estándar IEEE 802.15.4. Los BSN leen datos de los sensores y los transmiten inalámbricamente al nodo AN que es capaz de recibir los datos e identificar al nodo BSN que lo transmitió.

- Configurar el nodo AN para que guarde los datos recibidos en una base de datos MySQL siempre que le sea solicitado por un usuario mediante una interfaz de software desarrollada en Java.

Resumen

Este trabajo tiene como objeto diseñar e implementar una Red Inalámbrica de Sensores con base en el estándar IEEE 802.15.4 bajo la topología estrella y su interfaz de software para tomar de muestras de forma remota de los datos obtenidos de los sensores. La Red Inalámbrica tiene cuatro Nodos BSN (Basic Sensor Node) y uno receptor configurado como coordinador en la capa de Acceso al Medio para resolver el problema de colisiones de los datos que se transmiten por los BSN.

En cuanto a la interfaz de usuario, éste fue diseñado con diagramas UML a partir de los requerimientos establecidos para este, para realizar el programa también se configuró un servidor Apache y un servidor de base de datos MySQL. El Nodo Agregador se comunica con la base de datos haciendo solicitudes al servidor Apache como cliente por el puerto 8080 y de este modo se ejecutan los *scripts* en PHP que insertan las muestras solicitadas por el usuario.

Introducción

Las llamadas Redes de Sensores Inalámbricos son redes constituidas por nodos que interactúan entre sí transmitiendo datos de los sensores, por lo que también se le denominan Redes Inalámbricas de Sensores, cada nodo tiene un microcontrolador, alimentación, comunicación inalámbrica, sensores y memoria.

En cuanto al módulo de comunicación inalámbrica se utiliza módulos XBee Serie Uno los cuales trabajan con el protocolo IEEE 802.15.4 para la capa física y MAC y ZigBee para las capas posteriores, los XBee tienen la ventaja de que son tanto de bajo costo como de bajo consumo de energía, pero también son de baja tasa de transmisión.

Para este proyecto se utiliza la Tarjeta de Adquisición de Datos Arduino Uno, la cual tiene un microcontrolador Atmega 328, una memoria *Flash* de 32 *kilobytes*, una *EEPROM* de un *kilobyte*, una *SRAM* de 2 *kilobytes* y se puede alimentar con una pila de 9 [V]. Arduino Uno tiene un solo puerto serial así que para el nodo receptor se utilizan dos de los cuatro puertos seriales de la Tarjeta de Adquisición de Datos Arduino Mega 2560 necesarios para comunicarse con el XBee y una interfaz de usuario desarrollada en Java.

El objetivo de este trabajo es explicar el proceso de diseño e implementación de una Red Inalámbrica de Sensores topología estrella con base en el estándar IEEE 802.15.4 y ZigBee e incluir un programa de software para hacer una lectura remota de los datos obtenidos de los sensores al que se le ha llamado Sistema de Acceso a la Red Inalámbrica de Sensores (SARIS).

La finalidad de hacer una Red Inalámbrica de Sensores es que cualquier usuario pueda tomar muestras de la red sin que éste tenga que tener conocimientos de electrónica, de redes o de programación, es decir, que sea transparente para el usuario. Otro punto es que por la forma en que se diseña y desarrolla la red se pueda reconstruir para las distintas aplicaciones de la Redes Inalámbricas de Sensores basadas en ZigBee.

Este Trabajo Recepcional de divide en cuatro capítulos: 1) Marco teórico, 2) Diseño e implementación de una Red Inalámbrica de Sensores basada en ZigBee, 3) elaboración del Sistema de Acceso a la Red Inalámbrica de Sensores y 4) Pruebas de la Red Inalámbrica de Sensores y del programa de software SARIS.

En el primer capítulo se definen los parámetros principales con los que se describe la Red Inalámbrica de Sensores, se describe el algoritmo CSMA/CA ranurado y se explican los elementos básicos de los diagramas UML que se utilizan para el diseño del programa SARIS, los elementos de hardware que se van a utilizar para la implementación, así como el paradigma cliente – servidor.

En el segundo capítulo se expone el diseño e implementación de RIS comenzando por exponer las especificaciones de la red a configurar, las etapas por las que se pasan al realizarla. La configuración de los módulos XBee como transmisores y un módulo de comunicación XBee como receptor, también se explica cómo se configura el protocolo CSMA/CA ranurado en modo *Sleep* para evitar colisiones en la transmisión de datos.

El tercer capítulo describe los requisitos del programa SARIS que se definieron para realizarlo, con esto se deduce las funciones principales que debe cubrir el programa autenticación, administración de usuarios y lectura de muestras en los nodos, para finalmente llegar a los diagramas UML que definen las clases, paquetes e interacciones entre clases necesarias para codificarlas.

El cuarto capítulo tiene como objeto describir los resultados obtenidos a lo largo de las pruebas que se realizaron con la implementación del sistema completo por lo que se incorporaron elementos que inicialmente no fueron considerados debido a los resultados obtenidos, los cuales son descritos en este capítulo.

Contenido

Índice de figuras

1. Marco teórico

1.1. Definición de una Red Inalámbrica de Sensores

Una Red Inalámbrica de Sensores (RIS), o WSN por sus siglas en inglés (*Wireless Sensor Network*), consiste en nodos interconectados entre sí para transmitir datos. Los nodos están conformados por sensores, un controlador o procesador, memoria, alimentación y un módulo de comunicación inalámbrica (Figura 1.1).

Los sensores son los dispositivos que obtienen los datos que se van a transmitir a lo largo de la red, se define a "*los sensores como dispositivos que generan voltaje con base a una propiedad mecánica o química, es decir, que muestrea alguna variable física y entrega una señal de voltaje proporcional*" (Bell, 2013). Sin embargo, un sensor por sí solo no es capaz de transmitir los datos que obtiene, para ello se requiere del resto de los elementos del nodo.

El controlador es el que interpreta los datos que se toman con el sensor y gestiona su almacenamiento; la memoria es la unidad de almacenamiento de los datos. Por su parte, la alimentación o la fuente de alimentación es la que proporciona la energía eléctrica de la red en términos de voltaje y de corriente eléctrica, siendo a través de baterías; finalmente, el módulo de comunicación inalámbrica es un dispositivo de comunicación que permite hacer la transmisión de datos por radiofrecuencia.

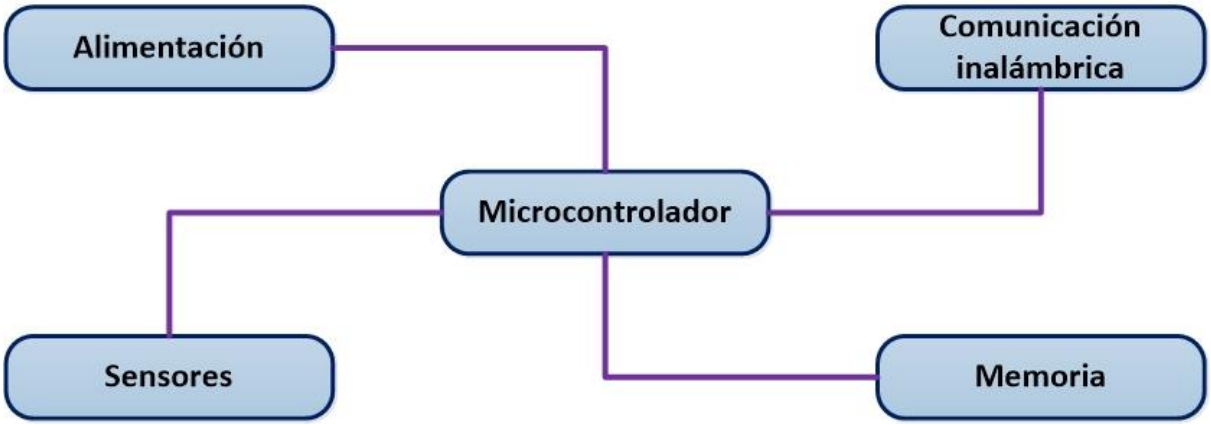

Figura 1.1. Arquitectura de un nodo inalámbrico. Recuperado de *Redes inalámbricas de sensores: teoría y aplicación práctica*, Fernández Martínez, R., et al. (2009).

En una RIS no todos los nodos tienen sensores, existen otros que cumplen con funciones diferentes y de acuerdo a ésta Bell (2013) los clasifica como: 1) el nodo sensor básico (*Basic Sensor Nodes* - BSN) o nodos finales, 2) el de datos (*Data Nodes* - DN) o enrutador también conocido como nodo *router*, y 3) el Nodo Agregador (*Aggregator Nodes* – AN) también conocido como el coordinador.

Los nodos BSN son los que tienen uno o más sensores, estos no procesan la información, únicamente toman los datos con los sensores y los transmiten por radiofrecuencia al siguiente nivel de la red que es el nodo DN. En el caso de los nodos DN su función es identificar los nodos BSN que pertenecen a su red, recibir los datos que cada uno de ellos transmiten y reenviarlos al siguiente nodo. En caso de existir un nodo DN en la red, este puede ser considerado como un router, ya que puede contener una tabla de direcciones que pertenezcan a la red.

El nodo DN puede ser considerado como router, para esto debe tener algoritmos que decidan la ruta por las que va a viajar la información en términos de nodos, de no ser así es más bien un nodo repetidor, lo que significa que sólo tiene que recibir y transmitir los datos, por lo que le da mayor cobertura o alcance a la RIS. Finalmente el nodo AN, tiene como función colectar los datos que recibe el nodo DN y enviarlos a un servidor de base de datos (Bell, 2013).

La finalidad de una RIS o WSN es lograr gestionar una gran cantidad de datos, lo que implica procesarlos y usarlos para tener soluciones (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007). Para lograr que estos datos se conviertan en información se recolectan, integran y organizan de manera sistemática.

Como todo sistema electrónico, las RIS tienen limitaciones y éstas varían de acuerdo a los dispositivos que conforman la red. Elegir los elementos que van a constituir una red inalámbrica de sensores depende de la aplicación, el presupuesto disponible y los requerimientos de la red, en los cuales hay que considerar que tecnologías se van a utilizar en función a la velocidad de procesamiento, tasa de transmisión, alcance, tiempo de vida de la batería, entre otras consideraciones.

1.2. Parámetros para describir la RIS

Los parámetros que permiten describir cualquier red son: la escala, el medio de transmisión, el control de acceso al medio y la topología. En una RIS, la simple elección del módulo de comunicación inalámbrica define la mayoría de los parámetros excepto la topología, esto se debe a que los dispositivos electrónicos que realizan la comunicación inalámbrica ya están configurados con un estándar de comunicación específico.

La escala es un parámetro que permite describir las redes acorde al alcance o distancia de las mismas, por ejemplo, en este texto se describe una RIS con una escala de red del tipo WPAN (*Wireless Personal Area Network*) o red inalámbrica de área personal, la cual se define en el estándar IEEE 802.15, que en términos generales, es de corto alcance, esto es de 10 metros más o menos. Los módulos de comunicación de la RIS son dispositivos XBee de la Serie Uno Pro los cuales están configurados bajo el estándar IEEE 802.15.4 y ZigBee.

El medio de transmisión se refiere a los medios guiados (cableados) y no guiados (inalámbricos). Para este parámetro se debe definir la frecuencia de portadora, el modo de división de canales, la modulación y la tasa de transmisión. Lo anterior ya está definido en el estándar IEEE 802.15.4 en el que se basan los dispositivos XBee.

El control de acceso al medio es el parámetro de red que se define el modo de asignación de canal a los nodos que desean transmitir datos. El protocolo más utilizado es CSMA (*Carrier Sense Multiple Access*), pero también existen otros protocolos como ALOHA que es el que antecede a CSMA; el estándar IEEE 802.15.4 tiene definido varios protocolos de control de acceso al medio que se puede utilizar siguiendo dicho estándar.

Por su parte, la topología de una red es la forma en que los nodos transmiten los datos a través de la RIS. El término topología se refiere a la ruta por la cual llegan los distintos nodos de una red. Para la RIS que se implementa en este trabajo se utiliza la topología estrella como se puede ver en la (Figura 1.2).

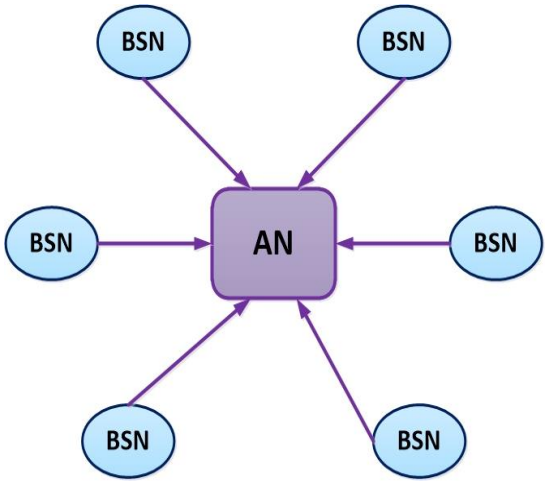

Figura 1.2. Topología estrella con los nodos BSN transmitiendo datos al nodo AN. Esta es la RIS propuesta.

La topología punto a punto es la base para formar cualquier otra topología, pues consiste de sólo dos nodos que se comunican entre sí de tres posibles formas: *simplex*, *half duplex* y *full duplex*. La comunicación *simplex* es unidireccional y consiste en que uno de los nodos únicamente transmite y el otro sólo recibe datos; la comunicación *duplex* es bidireccional, esto significa que los nodos pueden asumir el papel de transmisor o receptor según sea necesario, sin embargo, en la comunicación *half duplex* la transmisión ocurre en una sola dirección, por el contrario, en la comunicación *full duplex*, la transmisión se da simultáneamente en ambas direcciones como se observa en la Figura 1.3.

Por otro lado, en la topología estrella que se propone hay varios nodos BSN que transmiten datos a un nodo central AN (Figura 1.2). Esta topología es simple en el sentido de que no es necesario que se implemente un algoritmo que decida la ruta por la que tienen que pasar los datos para que lleguen a su destino, la desventaja es que si hay una falla en el nodo AN entonces no se puede leer ningún dato que sea transmitido por los nodos BSN.

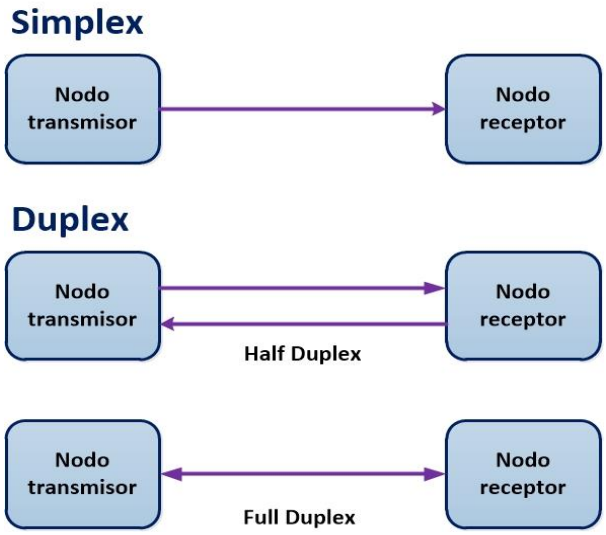

Figura 1.3. Modos de transmisión en la comunicación punto a punto.

1.3. El estándar IEEE 802.15.4 y ZigBee

Entre las tecnologías de comunicación que ejemplifican la WPAN destacan Bluetooth y ZigBee, entre otras. En términos de alcance, la Tabla 1.1 muestra un rango similar de alcance en ambas tecnologías aunque esto varía de acuerdo a los dispositivos que se utilicen; la tabla también muestra que la tasa de transmisión de Bluetooth es mayor con respecto a ZigBee, esto implica que en el primero se puede transmitir imágenes y hasta videos, mientras que en el segundo esto no es posible.

Otra característica importante es que los dispositivos ZigBee consumen menos energía que los dispositivos Bluetooth, para lograrlo se sacrifica la tasa de transmisión de datos (Lee, Su, & Shen, 2016). La potencia de transmisión es considerablemente menor en ZigBee (ver Tabla 1.1), la potencia mínima de transmisión es de 3.62 [W] y la potencia máxima es de 1 [kW].

Estándar	Bluetooth	ZigBee
IEEE	802.15.1	802.15.4
Banda de frecuencia	2.4 GHz	2.4 GHz, 868 MHz, 915 MHz
Tasa de transmisión	1 - 2 Mbps	250 kbps
Alcance	10 – 100 m	100 m
Potencia de transmisión	0 – 20 dBm	-25 – 0 dBm
Ancho de canal	1 MHz	22 MHz

Tabla 1.1. Tabla de comparación de las tecnologías de comunicación inalámbrica. Recuperado de: **http://bibing.us.es/proyectos/abreproy/11761/**. Camargo Olivare, J. L., (2009).

El estándar IEEE 802.15.4 y ZigBee están pensados bajo un modelo de capas como referencia al modelo de interconexión de sistemas abiertos OSI (*Open Systems Interconnection*), el cual tiene siete capas: la física, enlace de datos, red, transporte, sesión, presentación y aplicación. En el modelo OSI la capa física define las especificaciones de *hardware* y los parámetros de transmisión de datos por el canal, pero en la capa de enlace de datos (Tanenbaum & Wetherall, 2012) se define que la tarea es transformar un medio de transmisión en una línea libre de errores. Por su parte la capa de red determina la ruta de los datos hasta que éstos lleguen a su destino, el cual no se aborda a detalle en este trabajo debido a la topología de la red que se implementa.

Con referencia al modelo OSI, el estándar IEEE 802.15.4 define la capa física y la capa MAC que para OSI es parte de la capa de enlace de datos. En cuanto al estándar ZigBee la capa de red es equivalente a la del modelo OSI y tiene la misma función aunque como observa en la Figura 1.4 se indica como NWK por el término en inglés *network*, sin embargo, el resto de las capas del modelo de referencia no están presentes en ZigBee, sólo se trata de una sola capa de aplicación que a su vez se divide en subcapas.

1.3.1. Capas del estándar IEEE 802.15.4

El estándar IEEE 802.15.4 define la capa física (PHY) y la capa MAC (MAC), con la finalidad de contar con las ventajas de una fácil instalación, una confiable transferencia de datos, un rango de operación corto, una vida larga de la batería, mantenimiento sencillo y flexibilidad del protocolo (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

La PHY trabaja en tres bandas de frecuencia: 1) la de 2.45 GHz que tiene 16 canales donde se ocupa la modulación **OQPSK**[1], utiliza 4 bits por símbolo y el periodo del símbolo es de 16 microsegundos lo que establece una tasa de transmisión de 250 kbps; 2) la banda de 868 MHz que tiene un solo canal y 3) la de 915 MHz con 10 canales, éstas dos últimas bandas de frecuencia utilizan la modulación **BPSK**[2].

La capa MAC se refiere al control de acceso al medio, Medium Access Control por sus siglas en inglés, por lo que establece el modo de distribución del canal de transmisión a cada nodo para que pueda transmitir datos. El objetivo de la MAC es evitar una colisión, que tiene lugar cuando dos o más nodos quieren transmitir al mismo tiempo en el mismo canal; para lograrlo, esta capa define dos tipos de nodo: el nodo RFD (*Reduced Function Devices*) y el nodo FFD (*Full Function Devices*).

Los nodos RFD son dispositivos de función reducida que sólo pueden interactuar con un FFD, así que sólo pueden actuar como nodos BSN por lo que están equipados con sensores. Los nodos FFD tienen un conjunto de funciones que les permite actuar como nodos DN o AN, los FFD pueden comunicarse con los nodos RFD y otros FFD.

[1] **OQPSK**. Corresponde a las siglas en inglés de *Offset Quadrature Phase-Shift Keying*, el cual es una variante de QPSK; se trata de una modulación de usando 4 diferentes valores de fase para transmitir. Mayor detalle: http://www.ni.com/tutorial/5487/en/

[2] **BPSK**. Técnica de modulación que usa 2 diferentes valores de fase para transmitir datos que corresponde a las siglas en inglés *Binary Phase-Shift Keying*.

Visto desde la perspectiva de la capa MAC la topología estrella tiene un nodo FFD que recibe datos de los nodos RFD, estos nodos son análogos a los nodos AN y BSN que se observan en la Figura 1.2. En la topología estrella se adopta el modelo de red maestro – esclavo, donde el maestro es el nodo FFD que es considerado el coordinador de toda la red.

La figura de coordinador es una función definida en la capa MAC para los nodos FFD que les permite identificar el rango de direcciones que pertenecen a su red, sin embargo, sólo un nodo FFD puede ser coordinador, es decir, suponiendo que se tiene una red compuesta únicamente de nodos FFD, sólo uno puede fungir el papel de coordinador (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

1.3.1.1. Acceso al canal de transmisión en la capa MAC

La capa MAC envía y recibe datos en forma de tramas, de esta forma logra realizar la asignación del canal de transmisión a cada nodo de la red. En términos muy generales, una trama es un conjunto de bits que se generan de manera cíclica y organizada en cada nodo, una porción de la trama está definida para asignar los datos que el nodo quiere transmitir a lo que se le denomina carga útil (*payLoad*).

Las tramas que se generan en la capa MAC pueden tener una cabecera de símbolos llamado *beacon*, de ser así se trata de una supertrama que tiene una porción activa y otra inactiva, donde la primera está dividida en 16 partes iguales llamadas *slots* o ranuras como se muestra en la Figura 1.5. El *beacon* indica el inicio y fin de una supertrama lo que define el tamaño de la misma en símbolos a la que se le llama intervalo de *beacon* BI (*Beacon Interval*) y a la parte activa se le llama intervalo de duración SD (*Superframe Duration*).

$$\boldsymbol{BI(BO) = DBS \cdot 2^{BO}[Sym]} \qquad \textbf{Ecuación 1.1.}$$

$$\boldsymbol{SD(SO) = DBS \cdot 2^{SO}[Sym]} \qquad \textbf{Ecuación 1.2.}$$

$$\boldsymbol{0 \leq SO \leq BO \leq 14} \qquad \textbf{Ecuación 1.3.}$$

El intervalo BI (Ecuación 1.1) está en función del orden de beacon BO (*Beacon Order*) mientras que el periodo SD (Ecuación 1.2) está en función del orden de la supertrama SO (*Superframe Order*). Ambos intervalos se calculan por el tamaño base de la supertrama DBS (*Duration Base Superframe*) que se toma como valor constante equivalente a 960 símbolos si $SO = 0$.

Los parámetros *BO* y *SO* pertenecen al conjunto de los números enteros y deben cumplir con la condición de la Ecuación 1.3. Pero, si $SO = BO$ entonces se trata de una trama sin beacon y los intervalos BI y SD son equivalentes, esto significa que la trama sólo tiene la parte activa; si $SO < BO$ se trata de una trama con beacon o supertrama por lo que los intervalos BI y SD son diferentes donde necesariamente *BI* es mayor que *SD* ya que *BI* representa el tamaño de toda la trama.

Para conocer el tamaño de una ranura en símbolos basta con dividir SD entre 16 ya que la parte activa tiene siempre 16 ranuras. Si lo que se desea es conocer el periodo de la supertrama $T_{F_{BI}}$ se puede utilizar la Ecuación 1.4 donde el valor constante de 16 microsegundos corresponde al periodo de símbolo, de manera análoga se puede calcular el periodo de la parte activa T_{SD} sólo que en lugar de sustituir el valor BI se sustituye el valor SD.

$$T_{F_{BI}} = 16x10^{-6} \cdot BI \qquad \textbf{Ecuación 1.4.}$$

De manera detallada se puede ver la distribución de la supertrama en la Figura 1.5, los slots de la parte activa están asignados a la CAP (*Contention Access Period*) y al CFP (*Contention Free Period*), el tiempo en el que los nodos de la red compiten por el acceso al medio corresponde al periodo CAP mientras que el periodo donde uno de los nodos ha tomado el canal es el periodo CFP, y el tiempo que el nodo RFD transmite por el canal es el periodo GTS (*Guaranteed Time Slot*) este tiempo es asignado por el nodo FFD (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

En el periodo CAP se ejecuta un algoritmo que permite realizar la asignación de canal a cada nodo que desea transmitir, en este trabajo se utiliza el algoritmo CSMA/CA ranurado (Figura 1.5). En términos generales CSMA/CA (*Carrier Sense Multiple Access with Collision Avoindance*) es un algoritmo que se utiliza para redes inalámbricas con módulos de comunicación basados en WiFi.

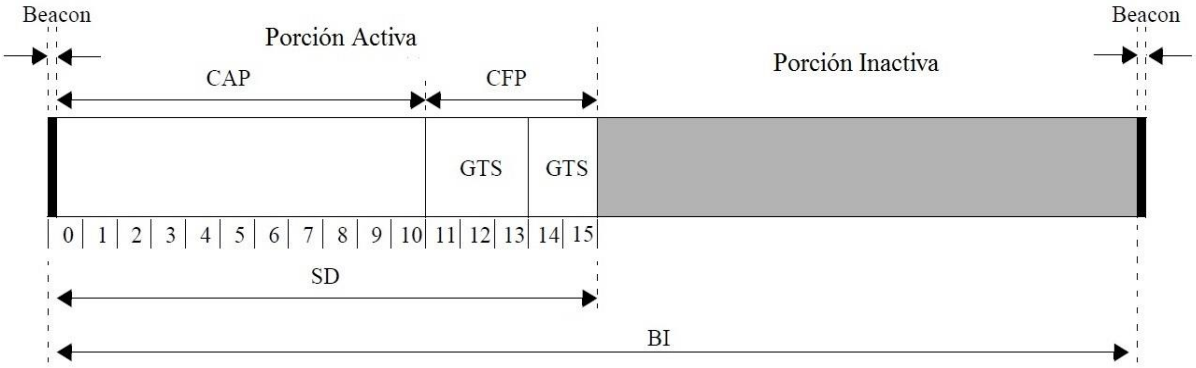

Figura 1.5. Estructura de la supertrama de acuerdo al estándar IEEE 802.15.4 (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

En el algoritmo CSMA/CA el canal debe ser utilizado por un nodo cuando esté disponible, de lo contrario se debe enviar un mensaje de canal ocupado (mensaje *Jamming*). Debido al mensaje *Jamming* se detiene el envío y se genera un tiempo de regreso llamado periodo de retirada (*backoff* por el término en inglés) $T_{backoff}$, pasado ese periodo se verifica nuevamente si el canal está disponible.

El algoritmo CSMA/CA ranurado (Figura 1.5) es una variante de CSMA en el que se tiene que considerar si se utiliza el modo *sleep* o no. El modo *sleep* es una forma en la que puede operar un nodo, se trata de programarlo para que esté inactivo durante un tiempo fijo de manera cíclica, en el tiempo restante del ciclo el nodo se activa para intentar transmitir datos.

Previo al modo *sleep* el algoritmo genera dos parámetros el número de periodos de retirada NB (*Number of Backoffs*) y *la* ventana de contención CW (*Contention Window*). El número NB es el número de veces que el nodo intenta tomar el canal para transmitir datos y este no debe ser mayor a 5 debido a que este es el tope permitido por el *macMaxCsma* que es el máximo valor de NB antes de que el algoritmo realice una asignación de canal, así que el intervalo de *macMaxCsma* es: [0,5] (IEEE Computer Society Sponsored by the LAN/MAN Standards Commitee, 2015).

El ancho de la ventana de contención CW es el valor que define el número de periodos $T_{backoff}$ que se necesita para limpiar la actividad del canal antes de iniciar una transmisión. Inicialmente el valor CW equivale al periodo de la ventana de contención inicial CW_o que es igual a 2; este valor disminuye en una unidad en caso de que el canal esté libre y sólo si CW es igual a cero el nodo puede tomar el canal de lo contrario reinicia el ciclo con CW igual a uno.

$$\boldsymbol{T_{backoff} = 2^{BE} - 1} \qquad \textbf{Ecuación 1.5.}$$

El exponente de retirada *BE* (*Backoff Exponent*) es el valor que define el periodo $T_{backoff}$, de acuerdo al diagrama de flujo que se ve en la Figura 1.5. El exponente *BE* depende de un rango de valores denominado *macMinBe* (Ecuación 1.6) que va de cero al valor *macMaxBe* que se encuentra definido por el rango que se muestra en la Ecuación 1.7.

$$\boldsymbol{macMinBe} \in [\mathbf{0}, \boldsymbol{macMaxBe}] \qquad \textbf{Ecuación 1.6.}$$

$$\boldsymbol{macMaxBe} \in [\mathbf{3}, \mathbf{8}] \qquad \textbf{Ecuación 1.7.}$$

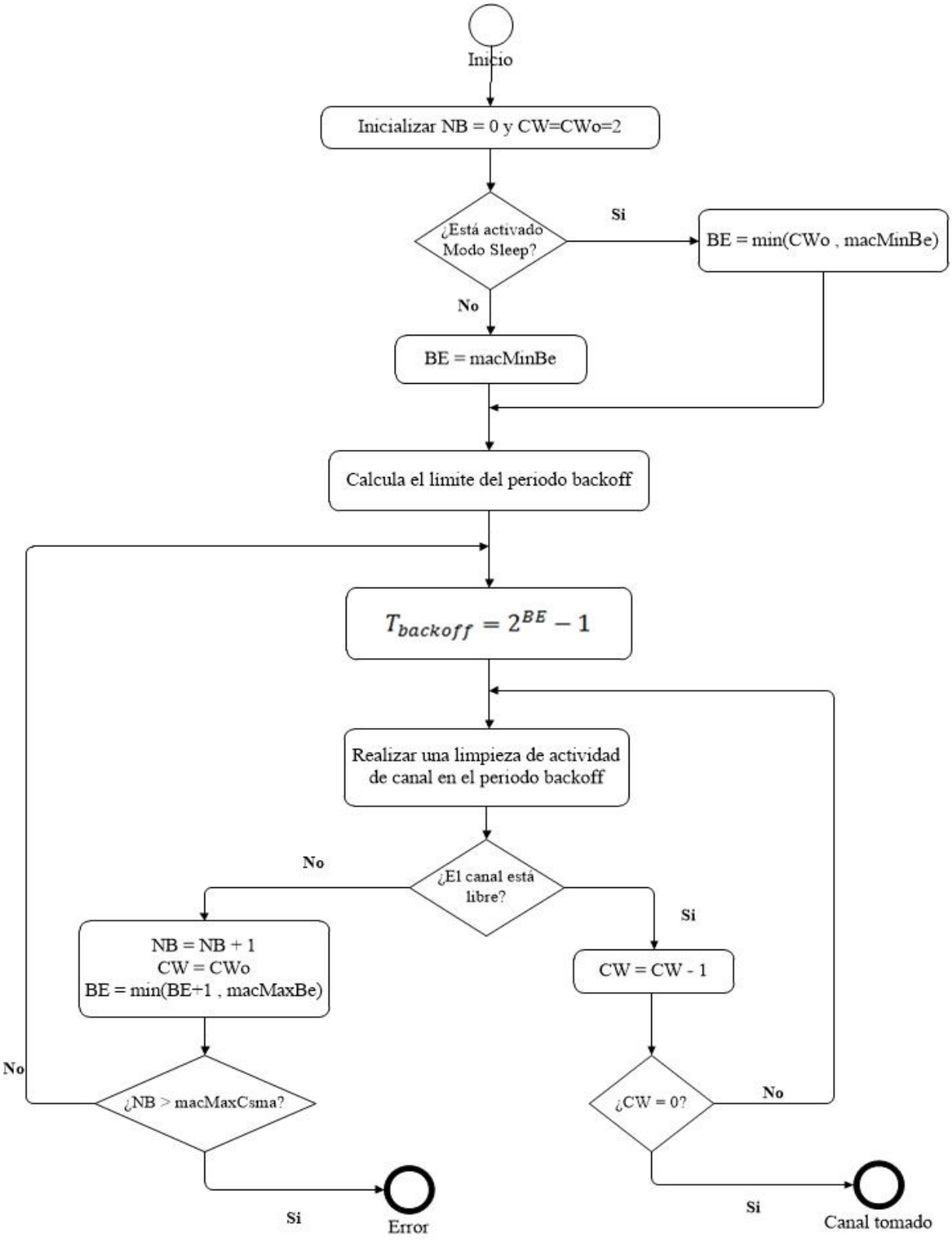

Figura 1.6. Algoritmo CSMA/CA ranurado tomado del estándar IEEE 802.15.4.

El valor de *BE* es distinto si el nodo RFD está configurado en modo *sleep* o no. Si el modo *sleep* no está activado, *BE* es equivalente al valor de *macMinBe*, si por el contrario, el modo *sleep* está activo el exponente *BE* toma el valor mínimo entre *CWo* a *macMinBe* y este valor se incrementa en uno en caso de que el canal esté ocupado teniendo como límite el valor *macMaxBe*.

Una vez calculado el exponente *backoff* se estima el límite del periodo de retirada que es 2^{BE}, posteriormente se calcula el periodo $T_{backoff}$, lo siguiente es limpiar la actividad del canal en el periodo *backoff*, cuando se activa esta etapa el nodo FFD descarta tramas recibidas en ese tiempo, a esta etapa se le denomina *Clear Channel Assessment* (CCA) que como se puede observar en la Figura 1.5, se repite dos veces.

1.3.2. Capas del estándar ZigBee

El estándar ZigBee conjunta la pila de protocolos de las capas más altas que son NWK y la capa de aplicación APL. La NWK se encarga de organizar y generar rutas en una red multipunto, y la capa APL se encarga de tomar y ordenar los elementos que hacen posible que se pueda desarrollar un programa de software que permita visualizar los datos de la red.

En ZigBee se consideran los nodos Coordinador, Router y Dispositivo Final y estos son análogos a los nodos BSN, DN y AN respectivamente. En relación a la capa MAC de IEEE 802.15.4, el Dispositivo Final ZigBee corresponde a los nodos RFD o FFD, el nodo *Router* ZigBee que tiene capacidad de ruteo corresponde a un FFD, y finalmente, el nodo Coordinador ZigBee es un administrador FFD que controla la red.

1.3.2.1. La capa NWK

Para formar una red inalámbrica basado en ZigBee, independientemente de su topología, es indispensable que cada nodo que pertenece a la red tenga indicados la dirección de nodo, el PAN ID[3] y el número de canal, éstos se indican en los dispositivos utilizando numeración hexadecimal. La dirección de nodo es una dirección única que permite distinguir el nodo en la red, ésta puede ser de 16 o 64 bits.

El PAN ID, es un identificador único de una red WPAN, con este número el nodo Coordinador distingue qué nodo pertenece a su red. Por último, el número de canal, con relación a los 16 canales que indica la capa PHY, la capa NWK debe tener identificada que canal se utiliza la transmisión de datos.

Al formarse una red con topología estrella, un nodo BSN o Dispositivo Final que desea unirse a un nodo AN o Coordinador debe primero hacer una solicitud de asociación a la red, esto lo hace mediante la capa MAC; si el nodo BSN es aceptado entonces el nodo AN envía un mensaje de aceptación también mediante la MAC, estableciendo una relación padre e hijo (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

[3] **PAN ID**. Es el término con el que se le denomina el número identificador de la red inalámbrica PAN o WPAN.

Lo anterior establece la base para formar redes con topologías más complejas como malla y árbol, pero para ello es necesario explotar otras funciones importantes de la NWK como (Faludi, 2011):

i. **El ruteo**. Se trata de un término que proviene de la palabra en inglés routing, el ruteo consiste en generar tablas de rutas o enrutamiento en las que se definen el cómo un nodo puede transmitir datos a través de una serie de nodos hasta llegar a su destino final.
ii. **Creación de redes Ad Hoc**. Se trata de un proceso automático en el que se crean distintas redes con los nodos disponibles, sin ninguna intervención humana.
iii. **Autoreparación de red**. Es un proceso automático en el que en caso de cualquier falla se repara la red reconfigurándola, por ejemplo, en el caso de faltar un nodo en la red, ésta se reconfigura con los nodos que aún están disponibles.

1.3.2.2. La capa APL

La capa de aplicación se encuentra dividida en tres subcapas: la infraestructura de aplicación AF (*Application Framework*), el objeto de dispositivo ZigBee ZDO (*ZigBee Device Object*) y la subcapa de aplicación APS (*Application Sublayer*), que trabajan conjuntamente para formar una aplicación ZigBee en la que se definen los formatos de los mensajes y los protocolos para que las subcapas de APL interactúen entre sí (ver Figura 1.4).

La AF está formada por objetos de aplicación APO (*Application Objects*) que son elementos de software que controlan una unidad de hardware que puede ser un sensor o un **actuador**[4] disponible en el nodo con el que se comunica la APO. El diseño e implementación de las APOs permite que distintos fabricantes construyan y vendan dispositivos ZigBee, cada APO provee funcionalidades que permitan colocar o recuperar valores en los atributos del nodo ZigBee.

El ZDO es un objeto que permite la comunicación y administración de una red WPAN desde la AF, otro punto es que se pueden agregar elementos definidos por ZigBee para proveer seguridad en la red. La APS permite la transferencia de datos para la APO y el ZDO que conjuntamente forman la aplicación ZigBee.

[4] **Actuador**. Consiste en un dispositivo que tiene como objeto proporcionar una fuerza para mover ("actuar") sobre otro dispositivo. Un actuador dentro de la RIS es un dispositivo al que se le envían datos desde el nodo AN, al contrario del sensor del que se reciben datos.

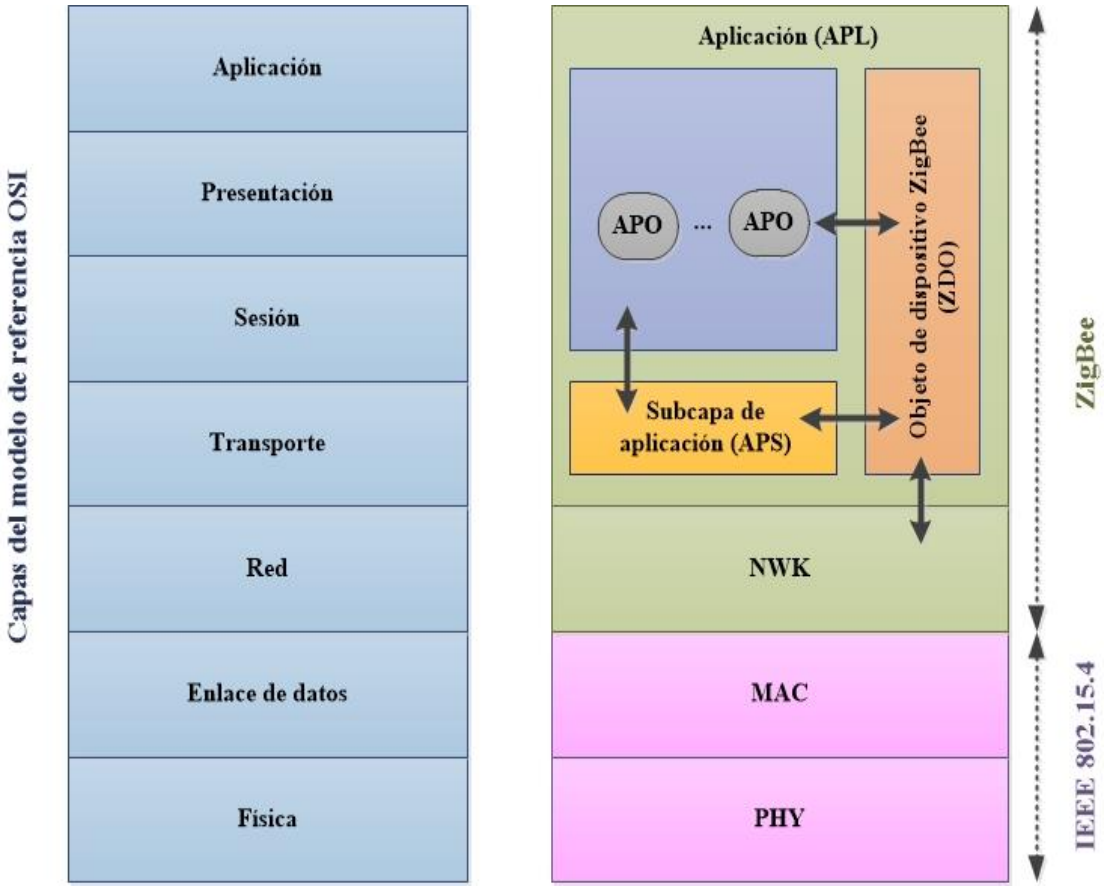

Figura 1.4. Las capas de los estándares IEEE 802.15.4 y ZigBee con el modelo OSI recuperado de: (Baronti, Pillai, Chook, Chessa, Gotta, & Hu, 2007).

1.4. La plataforma de hardware para configurar los nodos

Para implementar la RIS se utiliza el módulo de comunicación inalámbrica XBee y la Tarjeta de Adquisición de Datos (TAD). Los nodos BSN están formados por un XBee Serie Uno Pro y una TAD que tiene el microcontrolador Atmega 328p, este es Arduino Uno; el nodo AN tiene una TAD Arduino Mega con el microcontrolador Atmega 2560.

Tanto la TAD como el módulo XBee poseen puertos de comunicación serial para comunicarse entre ellos (los pines RX y TX) y una computadora, esto último facilita que sean programados. Para programar un XBee se utiliza la APO de su fabricante llamada X-CTU que es de acceso libre, para programar las TAD se utiliza la interfaz de programación de Arduino.

1.4.1. Tarjeta de Adquisición de Datos

El módulo Arduino Uno puede alimentarse en un rango de 7 a 12 [V], tiene un pin que entrega 5 [V] de salida, otro de 3.3 [V] y tres pines de tierra GND. Tanto Atmega 328p como Atmega 2560 tienen un cristal resonador de 16 [MHz] que determina el tiempo en que se ejecutan los comandos en el controlador. Los módulos de memoria en Atmega 328p son la memoria flash de 32 kB, una EEPROM de 1 kB y una SRAM de 2 kB; considerablemente menores que Atmega 2560 (Tabla 1.2).

Microcontrolador	Atmega 328p	Atmega 2560
Voltaje de entrada recomendable	7 – 12 [V]	7 – 12 [V]
Límite de voltaje de entrada	6 – 20 [V]	6 – 20 [V]
Pines de entrada analógica	6	16
Corriente por pin de 3.3 volts	50 [mA]	50 [mA]
Memoria Flash	32 kB de los cuales 512 bytes se utilizan para el gestor de arranque.	256 kB de los cuales 8 kB se utilizan para el gestor de arranque.
SRAM	2 kB	8 kB
EEPROM	1 kB	4 kB
Velocidad de reloj	16 [MHz]	16 [MHz]

Tabla 1.2. Tabla de comparación entre Arduino Uno y Arduino Mega. Datos recuperados de: **http://www.arduino.org/learning/getting-started**.

Las TAD tienen un convertidor analógico digital y seis entradas analógicas para hacer lectura de los sensores que se le conecten. La función *anologRead*() de Arduino lee los niveles de voltaje que entrega un sensor, sin embargo, los datos se leen de forma digital con palabras de 8 o 10 bits gracias al convertidor analógico digital.

El tamaño de la palabra define el número de muestras donde cada muestra se encuentra en un rango de voltaje de 0 a 5 [V]. Si cada palabra es de 10 bits entonces se tienen 1024 muestras donde la muestra 0 es 0 [V] y la muestra 1023 es 5 [V], si en lugar de 10 bits, cada palabra es de 8 bits el número de muestras se reduce a 256.

1.4.2. Módulo de comunicación XBee

Los XBee son módulos de comunicación que operan bajo el estándar IEEE 802.15.4 y ZigBee, éstos funcionan en la **banda ISM**[5] (2.4 GHz) así que de acuerdo al estándar dispone de 16 canales. Los XBee son manufacturados bajo el estándar registrado **ISO 9001:2000**[6] y optimizado para su uso en cinco países. Los módulos XBee son capaces de tolerar una temperatura mínima de -40 ºC y una máxima de 80 ºC.

[5] **Banda ISM**: El acrónimo ISM (Industrial, Scientific & Medical) hace referencia a la banda de frecuencias del espectro electromagnético definida internacionalmente para su uso no comercial en el área industrial, científica y médica. La banda se encuentra en la frecuencia de 2.4 GHz y se utiliza para redes inalámbricas con base a WiFi, Bluetooth y ZigBee.

[6] **ISO 9001:2000**: Pertenece a la serie de normas ISO 9000 que se enfocan a los Sistemas de Gestión de Calidad, la ISO 9001 es una variante que define los requisitos mínimos para lograr la satisfacción de los parámetros de calidad.

Los módulos XBee de la Serie Uno pueden programarse como nodos coordinador y dispositivos finales y así configurar redes con topología punto a punto y estrella; mientras que los módulos XBee de la Serie Dos pueden ser programados como dispositivos finales, nodos router y nodo coordinador, lo que permite configurar redes con topología árbol y estrella, con esta familia de módulos se tienen más elementos de la capa NWK del estándar ZigBee como el ruteo y la auto reparación de la red que se implemente.

En términos de alcance, los XBee Serie 1 PRO son mejores que los XBee Serie 1 pues tienen un alcance de 100 metros aún con obstáculos (Tabla 1.3), de acuerdo a los manuales, la potencia de transmisión de los PRO y la sensibilidad es mayor, si los comparamos con respecto a los XBee Serie 2 que tienen un alcance de 40 metros con obstáculos y utiliza menos potencia para transmitir, se reduce el consumo de potencia que representa una mejora con respecto a los PRO.

Características	**XBee Serie 1**	**XBee Serie 1 PRO**	**XBee Serie 2**
Alcance con obstáculo	30 [m]	100 [m]	40 [m]
Alcance en línea de vista directa	100 [m]	120 [m]	120 [m]
Potencia de transmisión	0 [dBm]	20 [dBm]	3 [dBm]
Sensibilidad del receptor	-92 [dBm]	-100 [dBm]	-95 [dBm]
Corriente TX	45 [mA]	215 [mA]	40 [mA]
Corriente RX	50 [mA]	55 [mA]	40 [mA]
Corriente en modo sleep	< 10 [µA]	< 10 [µA]	1 [µA]

Tabla 1.3. Comparación de familias XBee. Datos recuperados de: (Digi International, Inc., 2007) y (Digi International, Inc., 2007).

1.5. Recursos para el diseño e implementación de una interfaz de usuario

Uno de los objetivos de este trabajo es implementar una interfaz de usuario diseñada para la lectura de los datos tomados de los sensores colocados en cada nodo de la red, la interfaz debe ser capaz de comunicarse con el nodo AN o nodo coordinador y el servidor de base de datos donde se almacena la información recibida de los sensores.

Para implementar la interfaz se recurre al lenguaje de programación Java y se diseña desde el enfoque de la programación orientada a objetos. Java surge en la década de los 90 con el objetivo de ser una plataforma independiente para programar los aparatos electrodomésticos, sin embargo, debido al crecimiento de internet Java cubrió la creciente demanda de programas portables y compatibles con los navegadores de internet y que se ejecutan sólo si hay un solicitud.

Java es muy recurrente para implementar una interfaz de usuario, usualmente denominada GUI (*Graphic User Interface*). Una GUI es un software ejecutable en un sistema operativo de una computadora, mediante el cual el usuario puede interactuar con el sistema de modo transparente, es decir, de modo simple ya que no es necesario que conozca la programación de los nodos de la red y de la GUI, el usuario simplemente hace lectura de los datos que llegan al nodo AN.

En la programación orientada a objetos, un objeto es una entidad que tiene delimitado el conjunto de tareas que puede realizar y con qué otros objetos pueden interactuar. Esto es un paradigma diferente comparado con la programación estructurada, en donde el algoritmo que resuelve el problema se plantea como una serie de pasos ordenados para los cuales resulta práctico representarlo en un diagrama de flujo.

Al definir un objeto se establece el nombre, la tarea que va a resolver lo que lleva a definir las actividades que resuelven la tarea para la que existe el objeto, entonces, en lugar de diagramas de flujo, en la programación orientada a objetos se utilizan los denominados diagramas UML (*Unified Modeling Lenguage*).

1.5.1. Diagramas UML en el diseño de la GUI

Para diseñar una aplicación de software en Java se requiere en primer lugar realizar un modelo que es un conjunto de imágenes y texto que representan algo para fines de software, la ventaja de hacer un modelo es el ahorro de tiempo en la codificación del software, pues, es más sencillo cambiar un modelo que un código ya implementado.

Los modelos en UML constituyen un lenguaje visual para especificar, construir y documentar cada elemento de un sistema de software. El UML fue adoptado en 1997 por la OMG (*Object Management Group*), desde entonces, se han hecho revisiones y actualizaciones, la versión más reciente es la versión 2.5 emitida en junio de 2015.

En UML los modelos se realizan con base a diagramas que son fáciles de interpretar pues la simbología es simple, existen varios tipos como los diagramas de caso de uso, de actividades, de clases, de interacción que se dividen en diagramas de secuencia y de colaboración, entre otros; para este trabajo se utilizan los de casos de usos, los de actividades y los de secuencias.

1.5.1.1. Diagramas de caso de uso

Los diagramas de caso de uso son un macro registro de lo que se desea estructurar, lo que significa que es un registro “grande” donde se establecen los objetivos principales que se realiza el sistema, se registra lo que el sistema va hacer y lo que no, con ello se especifica el alcance (Kimmel, 2007).Para hacer los diagramas de casos de uso se utilizan tres figuras que representan distintos conceptos de acuerdo a su simbología, estos son el actor, caso de caso de uso y conectores.

El actor representa al objeto, se trata de quien participa en la caja de caso de uso, así que cada actor se asocia con una o más casos de uso y ejecuta las tareas ahí definidas. El caso de uso representa capacidades, es decir, lo que hacer el actor. A toda caja de uso se le debe dar un nombre y una breve descripción de la capacidad que se representa.

Los conectores indican la forma en que están asociados tanto actores como casos de uso, por lo que los conectores pueden tener modificaciones que proporcionan información extra. Así existen los conectores de asociación, de dependencia y de generalización. Los conectores de asociación muestran a los actores y a los casos de uso relacionados entre sí, los conectores de dependencia indican el caso de uso al que depende esto con una flecha discontinua que apunta hacia el caso de uso de la que se depende (Figura 1.7).

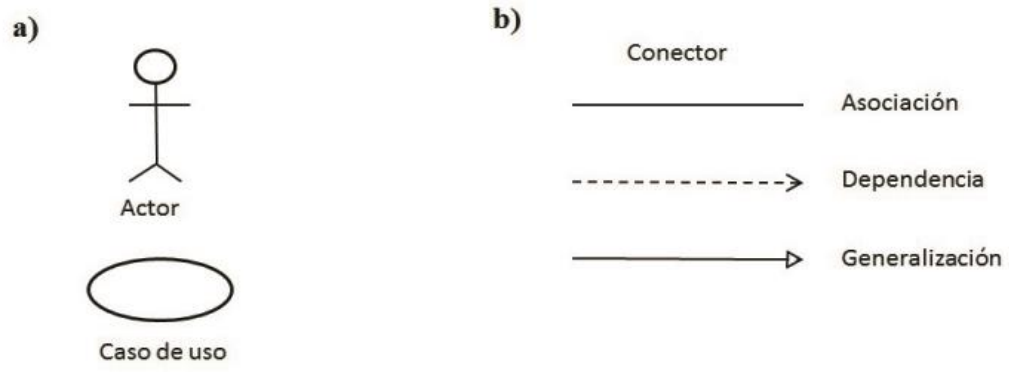

Figura 1.7. Elementos para los diagramas de caso de uso: a) actor, caso de uso y b) los conectores. Recuperado de (Kimmel, 2007).

Hay otros dos tipos de caso de uso que son los de inclusión y los de extensión que utilizan estereotipos que agregan información a la relación entre los elementos de un diagrama (Figura 1.8). Para el esquema de inclusión hay un dependiente que es para reutilizar el caso de uso al que depende, el cual es una entidad completa y distinta. El esquema de extensión se usa para agregar más detalle a una dependencia, lo que significa que no se agregan más capacidades; en una dependencia con relación extendida, la flecha apunta hacia el caso de uso básico y el otro extremo es de extensión.

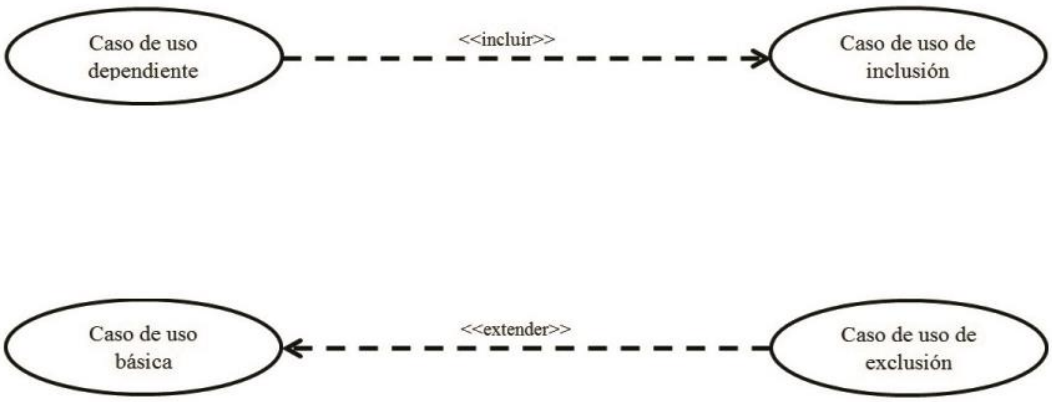

Figura 1.8. Esquema de los casos de uso de inclusión y exclusión.

1.5.1.2. Diagrama de actividades

Los diagramas de actividades son como los diagramas de flujo, pero modelan el comportamiento paralelo, se crean como un conjunto finito de acciones en serie o una combinación de acciones en serie y en paralelo (Kimmel, 2007). Los diagramas de caso de uso poseen un conjunto de símbolos como el nodo inicial que es el punto donde inicia el diagrama de actividades y tiene una línea de transición llamada flujo de control, representada por una flecha.

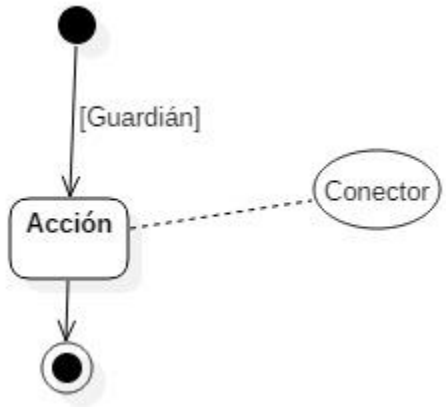

Figura 1.9. Elementos básicos de un diagrama de actividades.

Al flujo de control o el estímulo se le puede agregar también una condición guardián que indica que requiere una prueba antes de que el flujo continúe, esto se implementa como una prueba o condicional. El uso de las condiciones guardianes se muestra como texto entre corchetes, pasada la condición guardián se pasa a la acción; un condicional puede ser previo o posterior a una acción y se indican como notas en los diagramas de actividades.

La acción o el nodo de acción es lo que describe lo que sucede en los diagramas de actividades, éste es de forma más rectangular que los casos de uso. En un diagrama de actividades es muy importante asignar el nombre apropiado a la acción ya que éste define lo que hace y sobre qué actúa. Un conjunto de acciones describen una actividad así que para captarla se requiere completar varias acciones.

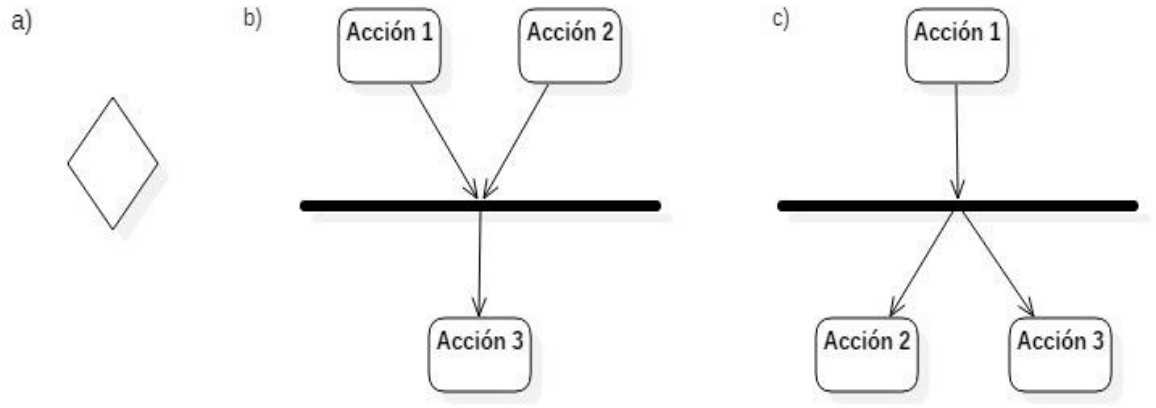

Figura 1.10. Los diagramas de actividades también tienen: a) nodo de decisión, b) y c) nodos de bifurcación. Recuperado de (Kimmel, 2007).

El diamante de decisión es un símbolo similar al diamante de decisión de un diagrama de flujo, pero, en un diagrama de actividades este símbolo representa tanto a un nodo de decisión como un nodo de fusión representando tanto la ramificación como la fusión de condicionales. El nodo de decisión es donde se presentan las opciones posibles dividiendo el diagrama en rutas alternativas y toma un camino de salida; en este nodo se utilizan la condición guardián y funciona bajo la lógica *if - else* y debe ser mutuamente excluyente (lógica XOR).

El nodo fusión tiene varias entradas y una sola salida que se obtiene una vez que todos los flujos han llegado al mismo tiempo, a estos nodos se les puede llamar también nodos de bifurcación ya que las bifurcaciones describen el comportamiento paralelo regresando a un solo flujo; aunque también existe el caso en el que se tiene una entrada y se tienen varias salidas simultáneas como se puede ver en la Figura 1.10.

1.5.1.3. Diagramas de secuencia

Los diagrama que describen el comportamientos de un sistema son los de interacción, entre los que se encuentran los de secuencia y los de colaboración, ambos expresan lo mismo, la diferencia radica en que los primeros "tienen un comportamiento explícito en el tiempo y son lineales" (Kimmel, 2007), mientras que los segundos, el orden en el tiempo está implícito.

Para hacer un diagrama de secuencia se define la línea de vida, que es un rectángulo con una recta vertical que desciende de ese rectángulo. El rectángulo de la línea de vida representa una clase, y la línea que desciende representa el lugar donde se sujetan los mensajes entrantes y salientes. Por otro lado, el símbolo de activación es un rectángulo vertical que reemplaza la línea de vida, representa la duración de un objeto.

Los mensajes son flechas dirigidas que conectan a las líneas de vida, cuando un mensaje puede empezar y finalizar en la misma línea de vida, a esto se le conoce como mensaje anidado y el mensaje de retorno es la respuesta de una línea de vida a otra. Otro tipo de mensajes son el mensaje encontrado y el mensaje perdido que tiene un receptor conocido pero un emisor desconocido.

El mensaje encontrado tiene un receptor conocido, pero un emisor desconocido; el mensaje perdido que tiene un emisor conocido pero el receptor no está especificado. También de acuerdo a la terminación de la flecha el diagrama indica la sincronización del mensaje, si la flecha termina con un triángulo relleno es un mensaje síncrono, si la flecha es de terminación sencilla sin formar un triángulo y sin rellenar es un mensaje asíncrono.

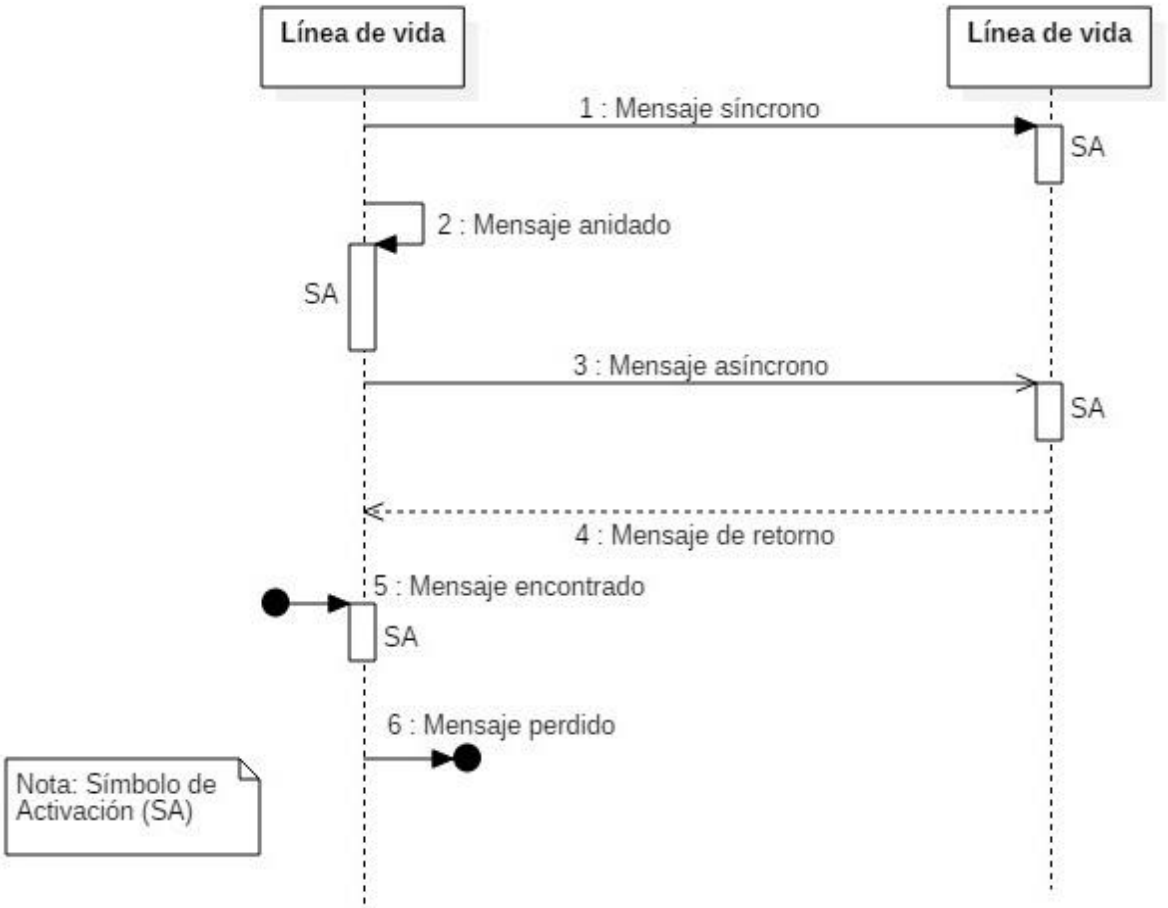

Figura 1.11. Elementos y estructura de un diagrama de secuencia.

1.5.2. Arquitectura cliente – servidor

La arquitectura cliente – servidor es en un paradigma donde el cliente es quien hace peticiones de servicio y el servidor es quien puede proporcionarlo. El servidor está formado por una base de datos donde se almacenan los datos que se reciben de la red, sin embargo, el cliente es quien hace la solicitud de que una muestra de esos datos se guarde en dicha base de datos.

En TCP/IP (*Transmission Control Protocol/Internet Protocol*) las comunicaciones entre dispositivos se rige por el modelo Cliente – Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones (Márquez Avendaño & Zulaica Rugarcía, 2017). Su funcionamiento es simple, se tiene un dispositivo cliente que requiere un servicio y lo solicita al dispositivo servidor, y éste realiza la función para la que está programado.

El cliente tiene un conjunto de procesos que permite al usuario formular los requerimientos y pasarlos al servidor, la forma más común de hacer esto es mediante una interfaz gráfica de usuario. En cuanto el servidor, se encarga de atender a las solicitudes del cliente, de algún recurso administrado por éste. Este trabajo trata de una base de datos, que consiste en un conjunto de datos que tienen elementos en común o que se definen en un mismo contexto que permite dar seguimiento a los datos de los sensores, teniendo así un historial de datos que proporcionan información de humedad del suelo y humedad del aire y temperatura ambiente.

Las funciones que lleva a cabo el cliente se resume en los siguientes puntos:

i. Interactuar con el usuario. Mediante ventanas, formularios y botones permite al usuario interactuar con el servidor mediante la interfaz.
ii. Procesar la lógica de la aplicación y hacer validaciones locales. La lógica de la aplicación en este trabajo se define con los diagramas UML, pues ahí se define que hace el sistema y cómo lo hace. En este punto se incluye las validaciones que son útiles en caso de que el usuario comenta un error, por ejemplo, evitar que ingrese campos de textos vacíos para una búsqueda en una base de datos.
iii. Generar requerimientos de bases de datos. Existe un procedimiento interno que estructura la petición del cliente al servidor de base de datos, éste no es visible al usuario.
iv. Recibir resultados del servidor. Una vez que el servidor ejecuta la petición del cliente, este debe entregar el resultado de esa petición. Por ejemplo un usuario ingresa nuevos datos a una base de datos que se encuentra en el servidor, la interfaz debe tener el procedimiento necesario para recuperar esos datos.
v. Mostrar resultados bajo un formato. Por ejemplo, una tabla de datos que se le denominan vistas en la interfaz de usuario. Esto se hace una vez que se recupera la respuesta del servidor.

Las funciones básicas del servidor se agrupan en:

i. Aceptar los requerimientos de bases de datos que hacen los clientes.
ii. Procesar requerimientos de bases de datos.
iii. Dar formato a los datos para trasmitirlos a los clientes.
iv. Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2. Diseño e implementación de un Red Inalámbrica de Sensores basada en ZigBee

La RIS que se implanta tiene dispositivos que miden la temperatura ambiente, la humedad relativa y la humedad de superficie. La Figura 2.1 muestra la distribución de nodos de la RIS que corresponde a la topología estrella donde hay cuatro nodos BSN y un nodo AN. En cada BSN hay un solo sensor, en BSN1 y BSN2 hay un sensor de humedad del suelo mientras que en los nodos BSN3 y BSN4 hay un sensor de humedad relativa y temperatura ambiente.

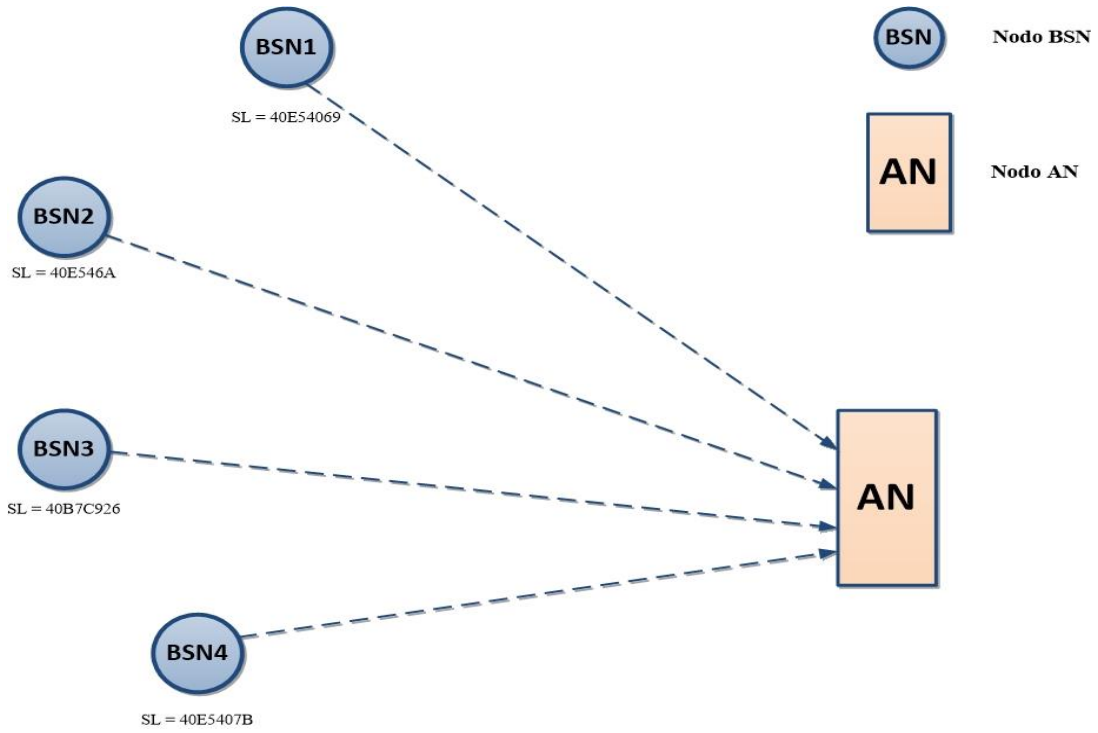

Figura 2.1. Red inalámbrica de sensores topología estrella.

La implementación de la RIS requiere establecer la comunicación de los módulos XBee de los nodos BSN con el módulo del nodo AN, configurar las TAD de los nodos BSN para que tomen muestras de los sensores y los envíen por radiofrecuencia usando los XBee, y finalmente, el receptor o nodo AN debe ser capaz de leer los datos que llegan y quién los envía. Para exponer estos puntos se dividen en cuatro etapas:

i. **Prueba de sensores**. Este punto es básicamente activar la entrada analógica de Arduino para poder leer los datos que los sensores entregan, esto siempre que los sensores los entreguen de modo analógico.

ii. **Comunicación punto a punto con los XBee y Arduino**. Se trata de programar los XBee para que sean capaces de comunicarse y luego utilizar una biblioteca de Arduino denominada XBee.h para poder tomar los datos y enviarlos a la computadora configurada como servidor.

iii. **Configurar la topología estrella**. Una vez establecida la comunicación punto a punto se puede replicar el código para cada nodo BSN que se comunique con AN, pero en este punto, si se unen las partes simplemente replicando el código, el nodo AN recibe datos pero no es capaz de identificar de dónde proviene los datos. Para corregir esto hay que programar la TAD para que reconozca a cada emisor o transmisor de los mensajes.

iv. **Configurar CSMA y el modo Sleep**. Al activar el CSMA/CA ranurado en modo Sleep se reducen las posibles colisiones evitando que todos los nodos transmitan al mismo tiempo.

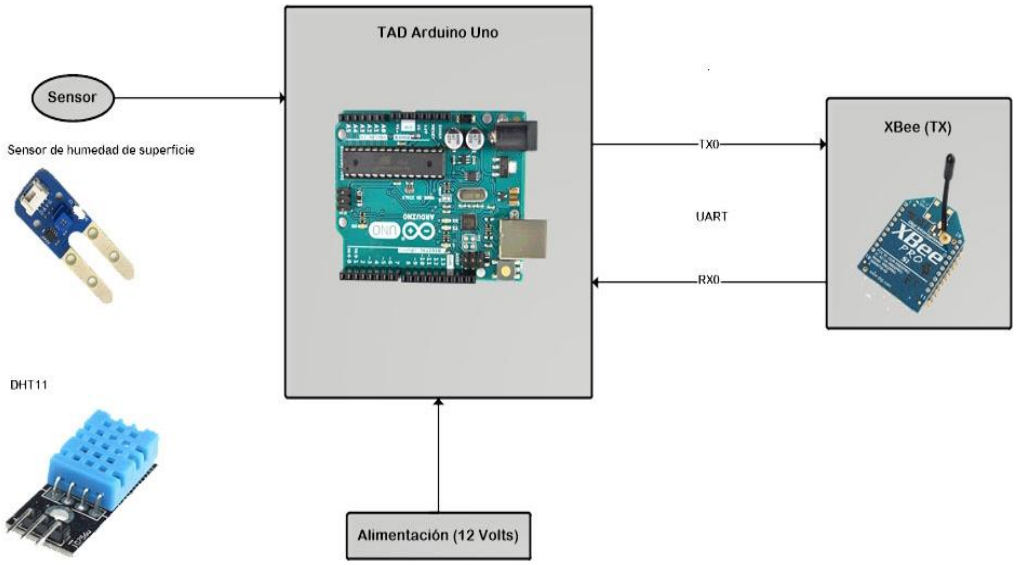

Figura 2.2. Dispositivos que forman la arquitectura de un nodo BSN.

2.1. Sensores en la RIS

Los sensores que se utilizan en la RIS son el de humedad relativa y temperatura, se encuentran en el dispositivo DHT11, y el de humedad de superficies ISTD-027. Para el dispositivo DHT11 simplemente se utiliza una biblioteca (*library* en inglés) ya definida en Arduino, sin embargo, para el sensor ISTD-027 no hay una biblioteca definida así que es necesario hacer un conjunto de pruebas para establecer cómo interpretar los datos que este lee, a este proceso se le llama caracterización del sensor.

2.1.1. Dispositivo DHT11

Para leer los datos del DHT11 en Arduino se utiliza la biblioteca DHT.h, en la cual se define el tipo de DHT y una entrada analógica de Arduino. La biblioteca funciona sólo para los dispositivos DHT11, DHT21 y DHT22, para definirlos en el código se crea una variable dht del tipo DHT con los parámetros pin de entrada (DHTPIN) y el tipo de dispositivo (DHTTYPE); se incluye la función *readHumidity*() y *readTemperature*() que se le aplican a la variable dht para leer la humedad y temperatura del sensor.

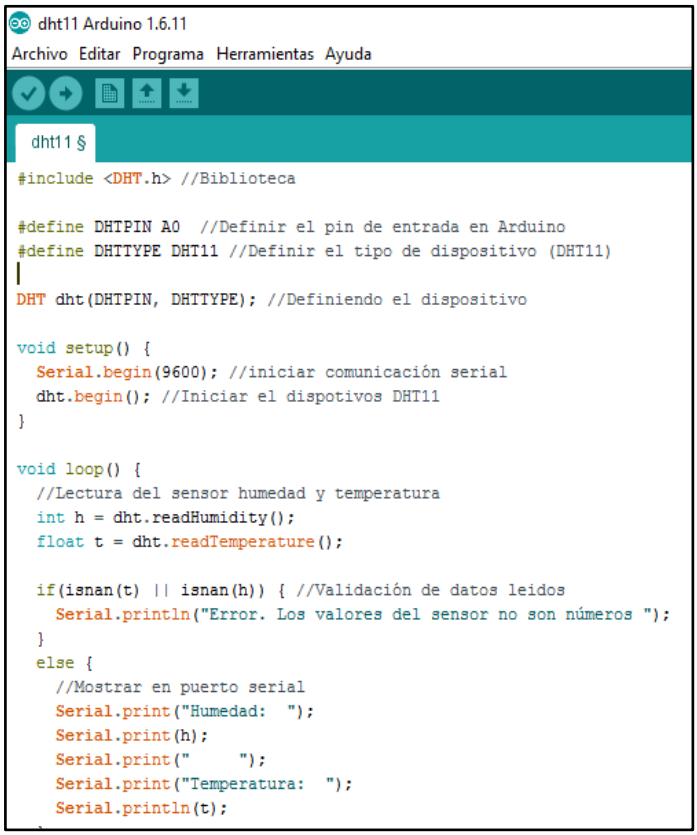

Figura 2.3. Código para tomar datos del dispositivo DHT11.

2.1.2. Sensor ISTD-027

En este caso se utiliza la función *analogRead*() para hacer lectura de una entrada analógica en la TAD de 10 bits por palabra, por lo que el parámetro de entrada de ésta es el pin de Arduino al que se conecta el sensor. Al conectar el sensor se observa que cuando éste entrega 0 [V] la lectura en la TAD es de 1023, pero cuando entrega 5 [V] la lectura es de 0. El cociente entre el rango del voltaje (de 0 a 5 [v]) y el número de muestras (1024) define la resolución de voltaje (ver Ecuación 2.1) y con esto en mente se puede hacer una aproximación matemática de la respuesta del sensor.

$$R_v(v) = \frac{rango\ de\ voltaje}{número\ de\ muestras}[V] \qquad \textbf{Ecuación 2.1}$$

Para realizar una caracterización del sensor es necesario tomar en cuenta el método gravimétrico que determina sólo la cantidad de agua en una muestra de suelo, para el cual se tienen que medir sus masas. Adicionalmente se debe tomar en cuenta los errores de medición y entregar un margen de error del dispositivo que se está caracterizando.

Sin embargo, como lo que se busca es obtener una relación de voltaje de salida con respecto al volumen de agua, se define para este trabajo, hacer una aproximación de la respuesta del sensor usando un vaso con capacidad de 228 [ml] se coloca tierra para maceta seca y con una jeringa de 5 [ml] se agrega agua de manera paulatina de mililitro por mililitro.

La muestra de tierra que se emplea se deja secar al aire libre durante 2 días y una vez que se observa sin humedad, se establece que la tierra está casi "seca" por lo que se considera que tiene un valor cercano al 0% de humedad. Por otro lado, se prueba el sensor en un recipiente que contiene sólo agua al que se ha considerado que tiene el 100% de humedad; estas consideraciones se realizan para definir referencias para las mediciones.

Al tomar una medición con la muestra de tierra que se valora como "seca" la lectura del sensor es de 1023, considerando la Ecuación 2.1, se multiplica por R_v a este valor para obtener el equivalente en términos de voltaje. Por el contrario, cuando el sensor ISTD – 027 entrega un valor digital equivalente a 0 la salida de voltaje es de cero.

En total se tomaron 40 muestras, midiendo la salida de voltaje del sensor de 0 a 41 [ml] de agua después se calcula la media de por cada unidad de volumen de agua en el vaso y con los valores resultantes se realiza la gráfica que se muestra en la Figura 2.4 donde se puede observar que la respuesta del sensor no es lineal.

La Figura 2.4 muestra también que la respuesta no tiende a variar mucho a partir de los 35 [ml] de agua pues entrega un valor aproximado de 1.5 [V], por otro lado, hay muchas variaciones abruptas de 0 hasta los 16 [ml] de agua, donde hasta entonces el valor de voltaje oscila con tendencia a bajar el valor de voltaje, pero justo en los 16 [ml] muestra un pico de casi 4 [V], esto más bien debe ser a un error en la medición más que en el dispositivo electrónico.

Al no existir una especificación, se selecciona el rango del volumen en el que se tomarán mediciones de 10 a 35 [ml] que es donde se aproxima la respuesta del sensor mediante una ecuación lineal, así se logra establecer un límite en el voltaje de salida como se observa en la Tabla 2.1; usando Scilab 6.0 es posible obtener las gráficas del valor obtenido mediante las muestras y valor esperado de acuerdo a la Ecuación 2.2 donde *vol* es el volumen de agua que se introduce al vaso con tierra y *V* es la salida de voltaje del sensor aproximada.

$$\boldsymbol{V(vol) = -0.1vol + 4.5} \qquad \textbf{Ecuación 2.2.}$$

Salida del sensor [V]	Respuesta esperada
V < 1.5	Demasiada humedad, ya no agregar más agua.
1.5 < V < 3.5	Mostrar el porcentaje de humedad leído.
V > 3.5	Demasiado seco, regar urgentemente.

Tabla 2.1. Rango de voltajes de salida de salida del sensor.

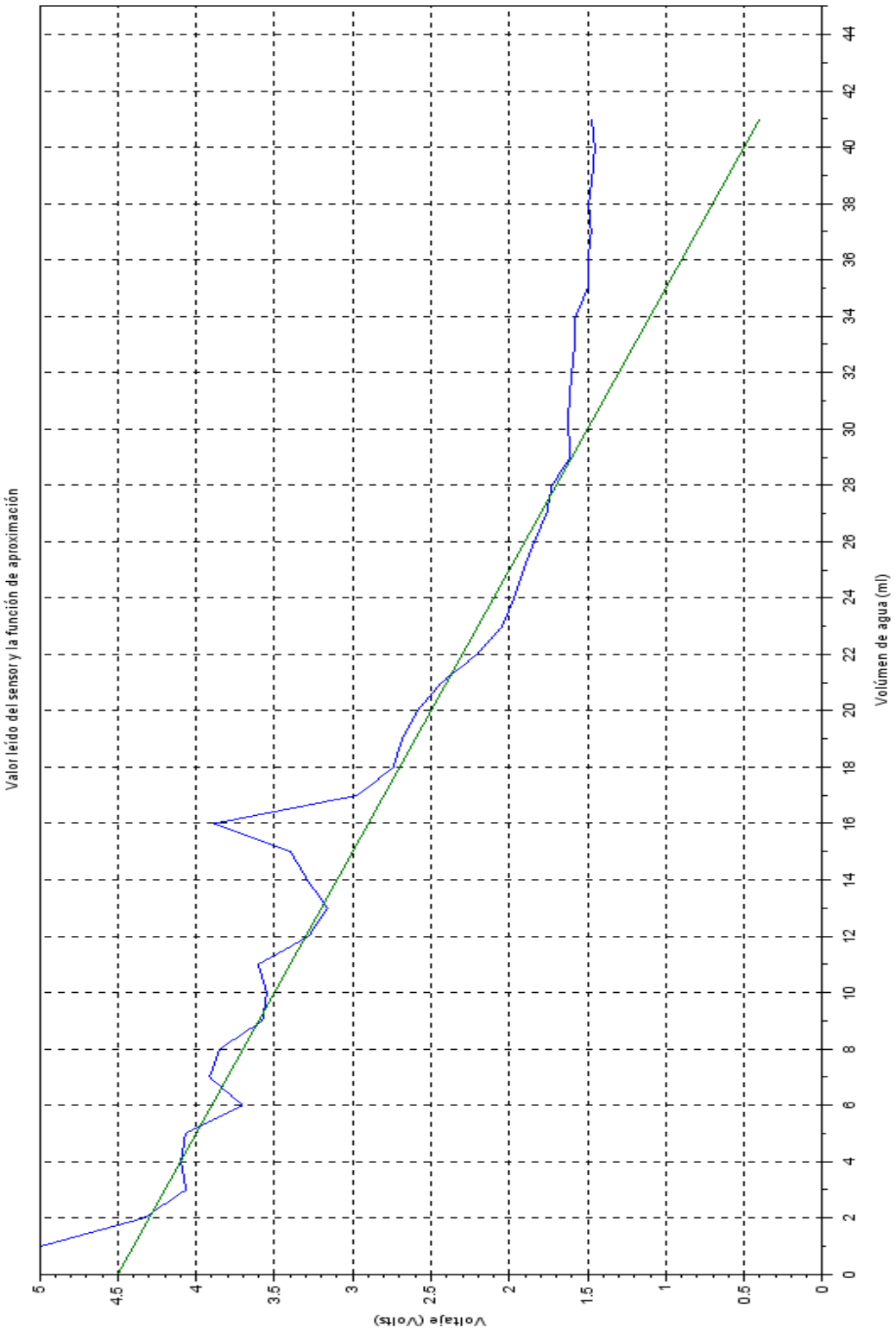

Figura 2.4. Gráfica de voltaje [V] vs volumen de agua [ml] del sensor. La recta es una de aproximación de la respuesta del sensor y la curva es el resultado de las muestras que se obtuvieron con el dispositivo ISTD – 027.

2.2. Programación de los XBee

Para programar los XBee se utiliza la comunicación serial con la computadora donde debe estar instalado el software X-CTU, que es de distribución libre, en éste se puede configurar la comunicación serial entre el XBee y la computadora. Los comandos AT se utilizan para especificar lo que se desea modificar de XBee, sin embargo, X-CTU ofrece un entorno gráfico para programarlos que es *Modem Configuration*.

Antes de configurar un XBee primero se debe conectar el adaptador explorador XBee que es un dispositivo que comunica el módulo XBee y con la computadora mediante la comunicación serial y después de enviar una trama de prueba para verificar que la comunicación serial esté funcionando correctamente, de ser así se muestra el número serial y el tipo de *modem* de XBee en una ventana emergente.

En caso de no estar funcionando correctamente la comunicación serial no se podrá identificar el módulo XBee y se muestra una ventana con un mensaje de error en el *firmware* de XBee, este caso es un error en XBee y para corregirlo se debe restablecer a sus parámetros iniciales para ello se utiliza el *Xplorer* XBee, realizando los siguientes pasos:

i. Abrir la interfaz X-CTU para leer el módulo de XBee que se desea restablecer.
ii. Verificar que los valores de configuración del puerto serial coincidan con lo descrito en el primer párrafo de esta parte.
iii. Seleccionar el botón *test/query*, si el XBee opera correctamente muestra su número serial de 64 bits, el cual viene de fábrica, de lo contrario, no se podrá leer dicho número serial o no detecta el modem.
iv. Si no se puede leer correctamente el módulo XBee se presiona el botón de reinicio que tiene el *Xplorer* XBee, de no tenerlo, se busca el pin RST (*reset*) y el pin de tierra (GND) y se conecta un interruptor de botón (*push button*) al pin RST en serie con una alimentación de 3.3 [V].
v. Abrir el Administrador de dispositivos, en caso de usar el sistema operativo Windows, en la opción de propiedades, después en configuración del puerto y en opciones avanzadas modificar los bytes de transmisión y recepción de 4096 a 64 bytes, el tiempo de latencia de 16 a 1 [ms], hecho esto se guardan los cambios y se regresa a la interfaz de X-CTU.
vi. Ir a *Modem Configuration* de X-CTU y seleccionar el tipo de modem XBP24 debido a que se trata de un XBee PRO de la serie Uno, activar la opción *always update firmware*.
vii. Desmontar el XBee y seleccionar el botón *Write de Modem Configuration*.
viii. Presionar varias veces el *push button* hasta que X-CTU muestre el mensaje de error de *firmware*, esto pasa porque X-CTU trata de escribir parámetros en el XBee que en realidad no está montado.
ix. Sin cerrar el mensaje de error se monta de nuevo el XBee y nuevamente presiona el botón *Write*.
x. Oprimir el *push button* hasta que X-CTU muestre el mensaje de que se han escrito los parámetros de manera exitosa, de lo contrario repetir el paso viii.
xi. Ya que se escribieron los parámetros se presiona el botón *Restore* que se encuentra en *Modem Configuration* y el modem está listo.

Cuando se trabaja con los módulos XBee se tienden a programar varias veces lo cual puede ocasionar un error en el *firmware* y ya no se puede leer correctamente mediante el puerto serial de XBee; una de las razones para que un *modem* XBee pierda la configuración correcta es seleccionar la opción incorrecta en *Modem Configuration* donde debe coincidir con la versión del módulo, esto es, tratar de escribir los parámetros de un XBee Serie 2 en un XBee Pro Serie Uno.

2.2.1. Entradas analógicas y digitales de un XBee

Al conectar físicamente un XBee se toma en cuenta que este opera con un voltaje máximo de 3.3 [V], que tiene los pines de comunicación serial D_{OUT} y D_{IN}, salida y entrada serial respectivamente como se observa en la Figura 2.5. Arduino tiene salidas de voltaje en DC de 3.3 [V] y de 5 [V] que se utiliza para conectar el módulo XBee y el sensor de DHT11 o el ISTD-027. Es importante verificar que el XBee esté conectado a la salida de 3.3 [V] porque de lo contrario no habrá ni transmisión ni recepción de datos.

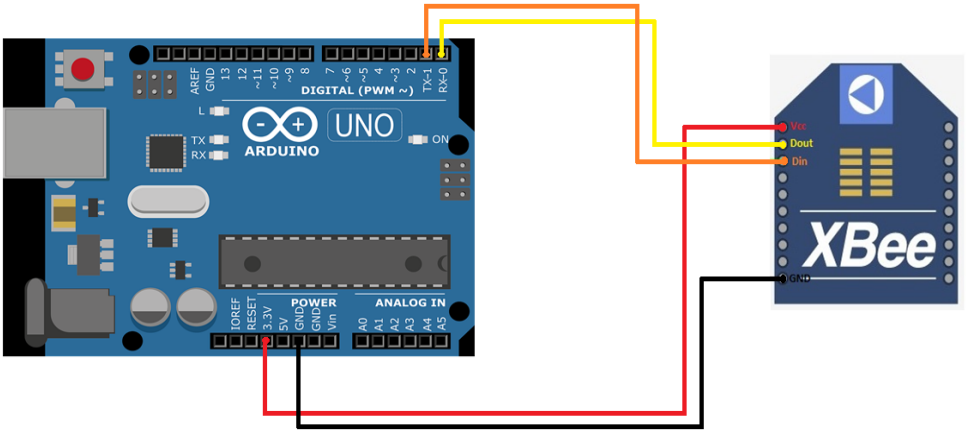

Figura 2.5. Conexión de XBee con Arduino.

Los pines I/O (entrada – salida) o los pines DIO (*Digital Input Output*) se pueden configurar para la transmisión o recepción, ya sea analógica o digital, para ello se consideran dos extremos que crean el 'cable virtual' el que va a transmitir y el que va a recibir. En *Modem Configuration* se selecciona el menú desplegable de cada pin DIO, cada uno tiene opciones específicas de configuración y mediante terminal se utiliza el comando ATD más el número de pin que se va a programar y el valor entero que corresponde a la opción que se desea en esa entrada o salida (ver Tabla 2.2).

Para este trabajo se opta por configurar los pines I/O como entrada y salida digital, debido a que los datos del sensor se leen con la TAD que entrega los valores a XBee de modo digital; si se configuran los XBee para entrada y salida analógica resulta redundante porque se utilizarían dos convertidores analógicos a digital, el de la TAD y el XBee.

A partir de este punto se diseña la RIS a implementar, se considera la Figura 2.1 como base, donde hay una RIS con topología estrella, desde la perspectiva del estándar IEEE 802.15.4, el nodo AN es el nodo coordinador y los cuatro nodos BSN son los nodos dispositivos finales que contienen los dispositivos DHT11 e ISTD – 027. En los XBee se configura el pin DIO0 como salida digital (valor 3 de la Tabla 2.2) para los módulos que están en los nodos BSN y como entrada digital (valor 5) para el nodo AN.

Valor	Pin DIO#	Valores para el comando ATD# y descripción
0	Todos los pines	DISABLE – Desactiva el pin y no se puede transmitir ni recibir por dicho pin.
1	D1, D4 y D8	NA – Significa que el valor no aplica para estos pines.
	D5	ASSOCIATED INDICATOR – Esta es la opción incluida que indica la asociación a la red.
	D6 y D7	CTS FLOW CONTROL
2	D0 – D5	ADC – Activa el convertidor analógico digital y se usa para trabajar con señales analógicas.
	D6 – D8	NA
3	Todos los pines	DI – Configura al pin como salida digital, así que todo dato lo lee y transmite de modo digital.
4	D0 – D7	DO LOW – Salida digital en bajo, significa que recibe una señal digital con referencia en bajo.
	D8	NA
5	D0 – D7	DO HIGH – Salida digital en alto, significa que recibe una señal digital con referencia en alto.
	D8	NA

Tabla 2.2. Valores que se le pueden asignar a los pines I/O, obtenidos de los datos mostrados en la interfaz X-CTU.

En el diseño se considera una forma de transmisión *simplex*. En cuanto a la distribución de nodos, el diseño considera cuatro nodos fijos en donde cada uno tiene preestablecido los sensores. El nodo AN se comunica tanto con los nodos BSN de la RIS como con la computadora que tiene el servidor de base de datos en modo serial, así que para este nodo se utiliza Arduino Mega 2560. En cuanto a los XBee son de la Serie Uno PRO, el direccionamiento de los módulos es de 64 bits, se distingue el nodo dispositivo final y el coordinador, se activa el estándar CSMA/CA ranurado, entradas y salidas digitales.

2.2.2. Direccionamiento de los XBee

Cuando se trabaja con un direccionamiento de 64 bits en los módulos XBee, se considera que la dirección del módulo se forma con un número SH (*Serial Number High*) y un número SL (*Serial Number Low*) de 32 bits cada uno, los cuales ya están definidos por el fabricante; por ejemplo, si SH es igual 13A200 y SL a 40E59069 la dirección completa del módulo es 13A20040E59069.

El direccionamiento de un XBee implica indicar la dirección destino donde se van a transmitir los datos formada por un número DH (*Destination Address High*) y un número DL (*Destination Address Low*). Para que cada nodo BSN transmita al nodo AN se indica la dirección SL del nodo AN, DL es igual al SL del nodo AN respectivamente (Tabla 2.3), como el nodo AN no va a transmitir a ningún otro nodo no es necesario programar una dirección destino. En el caso del número DH siempre equivale a 13A200 ya que todos los módulos tienen este valor SH.

Nodo	SL	DL
BSN1	40E54069	40E84AA6D
BSN2	40E5406A	40E84AA6D
BSN3	40B7C926	40E84AA6D
BSN4	40E5407B	40E84AA6D
AN	40E84AA6D	Nulo

Tabla 2.3. Tabla de valores SL y DL por nodo.

Con esto se logra la comunicación punto a punto de cada nodo BSN al nodo AN, sin embargo, si se encienden todos los nodos de forma simultánea, el nodo AN recibe datos sin identificarlos correctamente de acuerdo al nodo de origen. Para corregir este problema se activa el estándar de control de acceso al medio CSMA/CA ranurado.

2.2.3. CSMA/CA ranurado en los módulos XBee

Para que los módulos XBee operen con el algoritmo CSMA/CA ranurado en modo Sleep es necesario que al nodo AN se configure como coordinador y después se active el CSMA/CA en todos los nodos BSN, y finalmente, configurar el modo Sleep en cada uno de estos. Para realizar cada uno de estos cambios se utilizan comandos AT (*Attention*), aunque el programa X-CTU permite trabajar directamente dichos comandos o utilizar el configurador del modem XBee que es una interfaz gráfica que utiliza estos comandos.

El comando AT configura a un XBee como coordinador o nodo Final es CE (*Coordinator Enable*); si CE es 0 de acuerdo al estándar IEEE 802.15.4 se trata de un nodo RFD, pero si CE es 1 se trata de un nodo FFD. Así que el único nodo que se puede activar como coordinador es el nodo AN de la RIS, este paso es necesario al activar CSMA/CA pues quien asigna el canal es el coordinador.

Para activar el algoritmo CSMA/CA ranurado se utiliza el comando RN (*Random Delay Slots*) y se asigna el valor *macMinBe* a 3, con esta variable se calcula el exponente *BE* (ver Figura 1.5) y posteriormente se hacen las iteraciones de BE para esperar que el canal esté disponible, esto incrementando en 1 el exponente *BE* y hasta el valor *macMaxBe*.

El programa X-CTU tiene establecido el 3 como el valor máximo que se puede asignar a RN, el cual está dentro de los límites que establece la Ecuación 1.7. Pero con las iteraciones *macMaxBe* puede valer hasta 8, sin embargo, el valor *macMaxCsma* limita estas iteraciones hasta 5, así que el rango del exponente *BE* es [2,5] (ver Figura 1.5).

Figura 2.6. Gráfica del Periodo de Operación de un XBee cuando está configurado el modo cíclico de Sleep, realizada de acuerdo al manual de usuario XBee serie 1 Pro.

Para configurar el modo Sleep hay un conjunto de comandos que permiten realizar dicha tarea, entre los que destacan SM (*Sleep Mode*), ST (*Time Before Sleeping*) y SP (*Cyclic Sleep Period*). El comando SM debe ser 1 para activar el modo Sleep y 0 para desactivarlo, sin embargo, si se desea programar el periodo de sueño y de transmisión de cada nodo, SM debe ser equivalente a 4 que representa el modo *sleep* donde los módulos XBee son capaces de activarse y desactivarse cíclicamente.

Para asignar el periodo en el que un módulo XBee se activa o transmite datos, se utiliza el comando ST que es el tiempo de espera antes de entrar al modo *Sleep* y se debe proporcionar ese tiempo en un valor hexadecimal considerado que se multiplica por un milisegundo. El valor mínimo de ST es 0001 y el máximo FFFF que son respectivamente 1 [ms] y 65 [s].

Con el comando SP se asigna el periodo en el que un módulo XBee se encuentra inactivo o en modo *Sleep*, al igual que ST dicho tiempo se debe proporcionar en un valor hexadecimal pero considerando que se multiplica por 10 [ms], el valor mínimo es 20 y el máximo AF0 que son respectivamente 320 [ms] y 28 [s].

Al sumar el periodo ST y SP se obtiene un periodo al que de ahora en adelante se señala como Periodo de Operación (PO), puede tener un valor en el rango de 321 [ms] a 93 [s]. Debido a que la TAD tiene una función *loop* que se ejecuta cíclicamente en periodos de un segundo expresado en milisegundos, se opta por un valor de PO equivalente a un segundo, es decir, los periodos ST y SP al sumarse debe ser igual a un segundo en cada uno de los nodos BSN; por su parte al nodo AN se mantiene activo todo el tiempo que esté conectado a la batería.

Una vez decidido el valor de PO y tomando en cuenta que el tiempo ST es el periodo en el que el nodo despierta, se establece que éste tenga una diferencia de 50 milisegundos entre cada nodo. Al nodo BSN1 se le asigna un periodo ST equivalente de 50 milisegundos, el tiempo restante de 950 milisegundos del periodo total PO corresponde al tiempo SP y para el nodo BSN2 el periodo ST es de 100 milisegundos y así sucesivamente hasta llegar al nodo BSN4 como se muestra en la Tabla 2.4.

Nodo	Periodo (ms)		Valor (Hex)	
	ST	**SP**	**ST**	**SP**
BSN1	50	950	32	5F
BSN2	100	900	64	5A
BSN3	150	850	96	55
BSN4	200	800	C8	50

Tabla 2.4. Periodo de tiempo ST y SP en milisegundos y en hexadecimal asignados a los nodos BSN.

2.3. Programación de las TAD de los nodos

Para implementar los nodos que forman la RIS con topología estrella, se requiere programar las TAD con sus respectivas funcionalidades. Los nodos BSN se programan para que tomen lectura de los sensores, estos datos se asignan a un arreglo en los que el primer elemento del arreglo es una etiqueta que le permite al nodo receptor AN identificar de qué nodo provienen los datos, finalmente los datos se transmiten.

En cuanto el nodo AN, éste realiza tres tareas: la primera es recibir los datos de los nodos BSN e identificar quién envía los datos, la segunda es comunicarse con un servidor de base de datos e insertar datos en éste y finalmente, comunicarse con la interfaz de usuario desarrollada en Java; en esta sección sólo se describe la primera.

Existe una biblioteca de XBee para Arduino llamada XBee.h, con esto lo primero que se logra es la comunicación punto a punto entre un nodo BSN y el nodo AN, los datos recibidos se despliegan en el puerto serial de Arduino, así que en esta etapa la interfaz de X-CTU se utiliza únicamente para verificar que los datos están llegando correctamente.

2.3.1. Comunicación punto a punto

Para lograr la comunicación punto a punto utilizando los dispositivos que se muestran en la Figura 2.2 se debe realizar un código transmisor y uno receptor de modo general, para ambos casos se define la variable xbee que se inicializa con el puerto serial. Con la biblioteca XBee.h se pueden usar funciones ya definidas para transmitir y recibir paquetes de datos.

Para los nodos transmisores BSN se codifica el algoritmo que muestra la Figura 2.7 en los que se cargan a un arreglo los datos que transmiten, éste se denomina *payLoad* o la carga útil ya que tiene el identificador del nodo (d), la humedad (h) y la temperatura (t) estos elementos deben ser del tipo entero ya que el arreglo es del tipo uint8_t ver Figura 2.8. En cuanto a XBee sus datos analógicos se muestran por palabras de 8 bits ya que este módulo transmite y recibe datos por bytes.

Crear el paquete de transmisión tx el cual debe contener el arreglo *payLoad*, la dirección del módulo XBee que va a transmitir datos (addr64) y el tamaño del arreglo de datos en bytes debido a que se debe tener este número de bytes de referencia para el receptor. En la dirección del módulo se puede usar direcciones de 16 bits (*Tx16Request*) o de 64 bits (*Tx64Request*).

Una vez que se tienen todos los elementos del paquete *tx* se pueden enviar al nodo AN y este debe tener una variable que sea capaz de almacenar los datos que recibe del nodo BSN, esta es rx64 del tipo *Rx64Response* que se encuentra definida en la biblioteca XBee.h. El algoritmo del nodo receptor AN se muestra en la Figura 2.9, éste sólo define cómo recibir los datos del transmisor XBee y leerlos en el terminal serial de Arduino.

Para recibir paquetes, la biblioteca XBee.h permite definir el límite máximo de bytes a recibir del nodo BSN, como se puede observar en la Figura 2.10 el límite superior es de 110 bytes (*MAX_FRAME_DATA_SIZE*), en la función *loop*() de Arduino primero se hace lectura de que haya flujo de datos en el módulo XBee receptor y posteriormente se toman los datos del arreglo *payLoad* usando la función *getData*() dentro de un ciclo *for*.

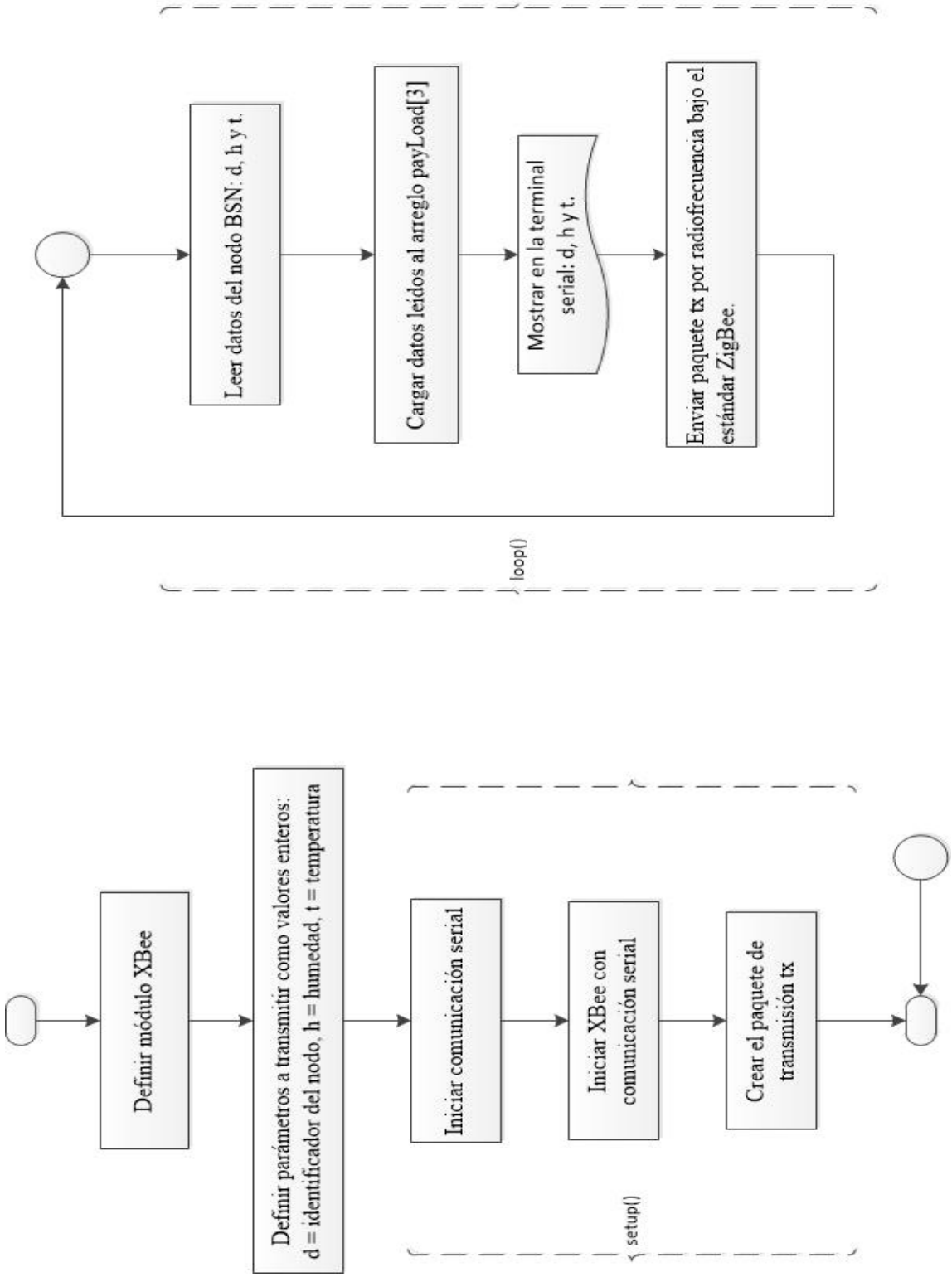

Figura 2.7. Diagrama de flujo del algoritmo codificado en los nodos BSN de acuerdo a los elementos de la biblioteca XBee.h hecha con base a la Guía del desarrollador disponible en: **https://github.com/andrewrapp/xbee-arduino/wiki/Developers-Guide**.

```
#include <XBee.h> //Biblioteca XBee

XBee xbee = XBee(); //Objeto XBee
int d=1; //Identificador del nodo inicializado
int h, t; //Parámetros de lectura
uint8_t payLoad[] = {0,0,0}; //Arreglo para almacenar datos
XBeeAddress64 addr64 = XBeeAddress64(0x0013A200, 0x40720B08); //Dirección de 64 bits del nodo BSN
Tx64Request tx; //Variable que tiene el paquete a transmitir

void setup() {
  Serial.begin(9600); //Iniciar puerto serial
  xbee.begin(Serial); //Inicializar xbee con el serial
  tx = Tx64Request(addr64, payLoad, sizeof(payLoad)); /*Crear paquete de transmisión, este es el que se
  envía por radiofrecuencia*/
}
void loop() {
  int h = analogRead(A0); //Lectura del sensor

  //Cargando valores a elementos del arreglo payLoad
  payLoad[0] = d; //Identificador del nodo
  payLoad[1] = h; //Humedad
  payLoad[2] = t; //Temperatura

  //Enviar por IEEE 802.15.4
  xbee.send(tx);
```

Figura 2.8. Código de los transmisores BSN. Estos varían ligeramente de acuerdo al nodo.

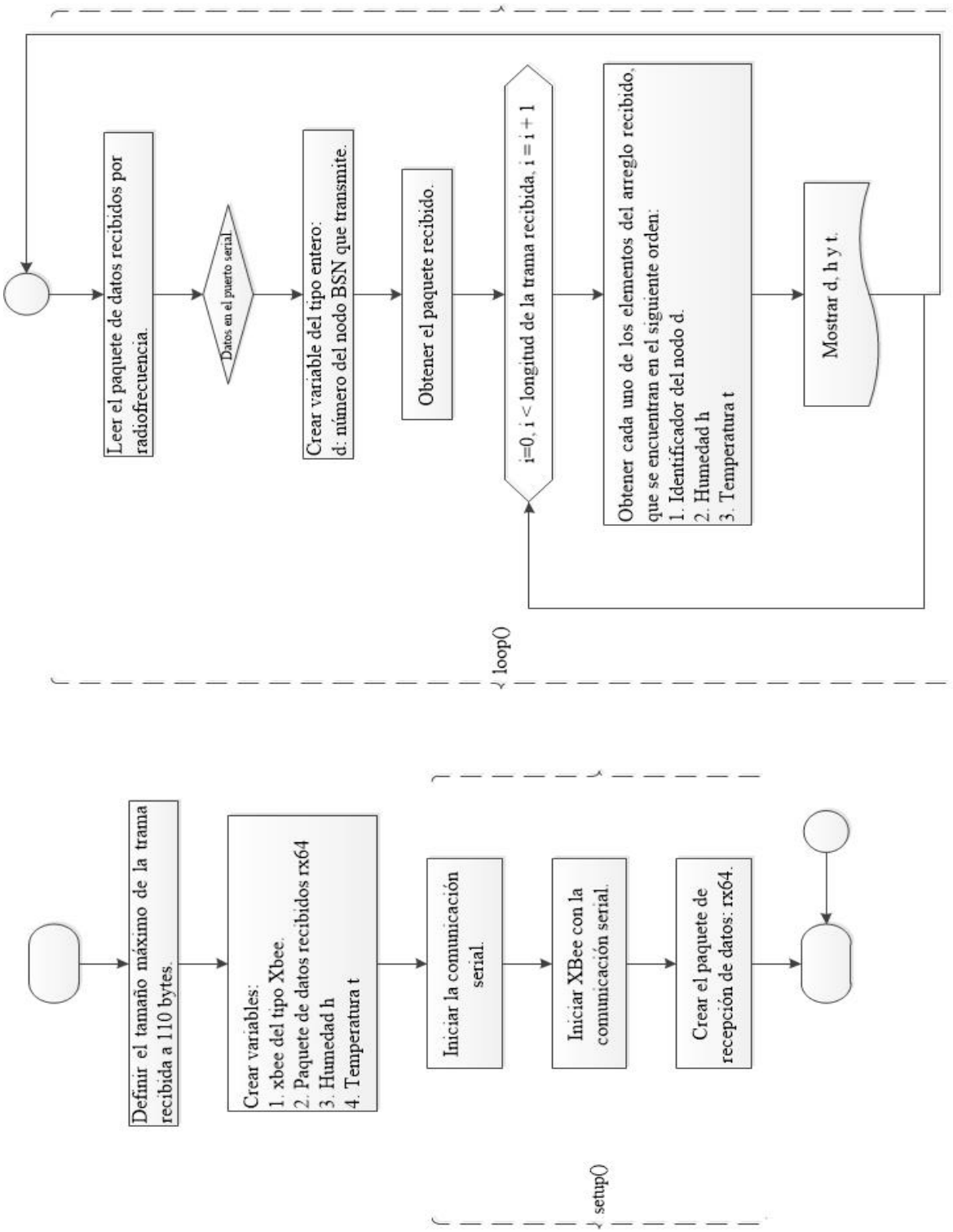

Figura 2.9. Diagrama de flujo del receptor AN. El algoritmo está hecho con base a la Guía del Desarrollador disponible en: **https://github.com/andrewrapp/xbee-arduino/wiki/Developers-Guide**.

receptor §

```
#include <XBee.h>
#define MAX_FRAME_DATA_SIZE 110 //Definiendo límite máximo del paquete recibido

XBee xbee = XBee();
Rx64Response rx64; //Variable que almacena los datos que se reciben del nodo BSN
int d, h, t; //Variables de entrada

void setup(){
  Serial.begin(9600);
  xbee.begin(Serial);
  rx64 = Rx64Response();
}
void loop() {
  xbee.readPacket(100); //Leer el flujo de datos en el módulo XBee

  rx64.getRemoteAddress64(); //Leer la dirección del módulo transmisor de 64 bits


  if(xbee.getResponse().isAvailable()){
    xbee.getResponse().getRx64Response(rx64);
    for(int i=0; i<rx64.getDataLength(); i++){
      d = rx64.getData(0);
      h = rx64.getData(1);
      t =  rx64.getData(2);
      delay(200);
    }
  }
}
```

Figura 2.10. Fragmento del código del receptor AN en Arduino.

2.3.2. Implementación de la RIS con topología estrella

Para forma la red con topología estrella que se muestra en la Figura 2.1, se retoman el código para la comunicación punto a punto (Figura 2.8 y Figura 2.10); las diferencias son: 1) activar el protocolo CSMA/CA ranurado en modo *sleep* y 2) se usa un identificador de nodo que está en el primer elemento del arreglo y con esto se detecta de qué nodo provienen los datos (Figura 2.11).

Para identificar el nodo del que provienen los datos se utiliza un condicional *if*, dado que sólo son cuatro nodos, considerando que los primeros dos nodos BSN1 y BSN2 tienen el sensor ISTD – 027 conectando la condición implica que el identificador del nodo d sea menor o igual a 2, si cumple esta condición se lee la humedad y en cuanto a la temperatura se le asigna el valor de 0 que en este caso representa que no hay lectura de datos en este parámetro. En caso de que la primera condición no se cumpla se establece una segunda condición en el que si el nodo identificador d sea menor o igual a 4, implica que el nodo tiene el dispositivo DHT11 y entonces se reciben datos de humedad relativa y temperatura ambiental.

```
void setDataInput(int d){
  payLoad[0] = d; //Identificador del nodo BSN recibido.

  if(d <= 2){ //Se trata del sensor ISTD - 027
    payLoad[1] = rx64.getData(1); //Humedad del suelo
    payLoad[2] = 0; //No hay lectura de temperatura. Se envía 0.
  }
  else if( d <=4){ //Dispositivo DHT11
      payLoad[1] = rx64.getData(1); //Lectura de humedad relativa
      payLoad[2] = rx64.getData(2); //Lectura de temperatura ambiente
  }
```

Figura 2.11. Función setDataInput() que se agrega al código del nodo AN para que identifique al nodo BSN del que provienen los datos.

Al recibir los datos estos se asignan a un arreglo llamado *payLoad*[] el cual tiene tres elementos donde el primero corresponde al identificador del nodo BSN (d), el segundo elemento almacena la humedad y el tercero la temperatura. Así, lo que se muestra en la terminal del puerto COM de Arduino son grupos de tres números como se observa en la Figura 2.12.

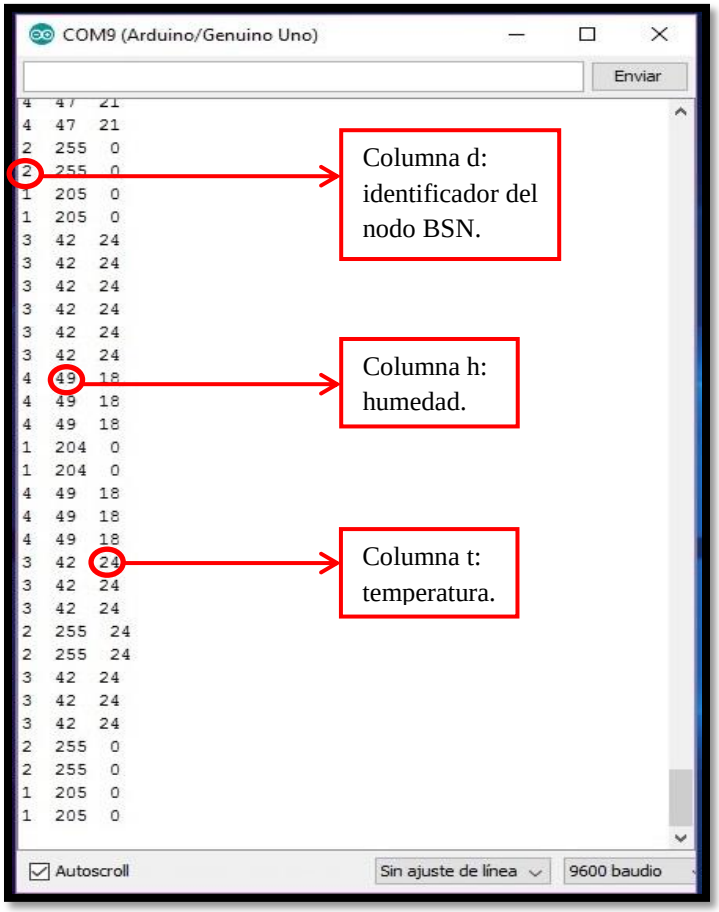

Figura 2.12. Matriz de 3x1 de los datos recibidos en el nodo AN.

Para hacer pruebas se hace un prototipo de red en el que se colocan tres de los cuatro nodos BSN en un macetero de madera y el cuarto nodo BSN queda de forma independiente. En cuanto a las baterías se utilizan unas recargables; el macetero se acondiciona para que los nodos sean alimentados de forma paralela por una batería de 12 volts ver Figura 2.13.

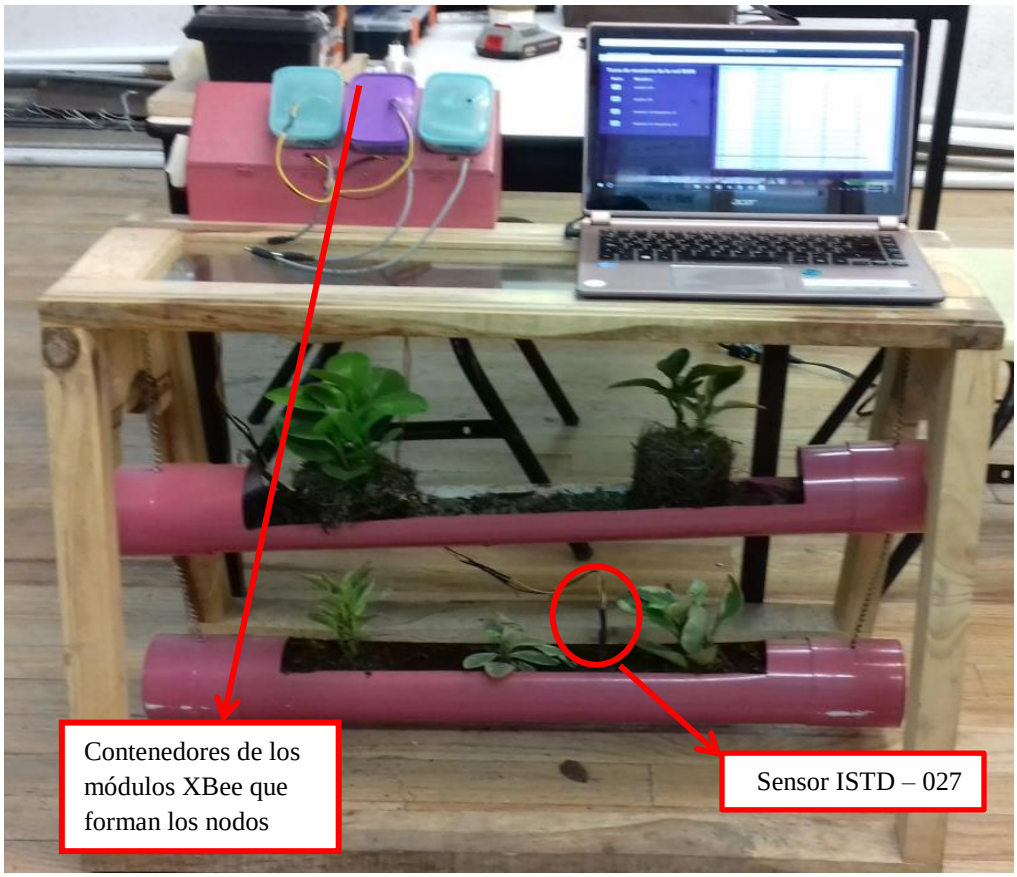

Figura 2.13. Prototipo de red inalámbrica de sensores topología estrella en un macetero de dos niveles, muestras los nodos BSN1, BSN2 y BSN3 fijos en el macetero.

3. Elaboración del Sistema de Acceso a la Red Inalámbrica de Sensores

El Arduino Mega colocado en el nodo AN además de recibir los datos de los nodos BSN tiene que hacer dos cosas: 1) comunicarse mediante un puerto serial con el programa y 2) comunicarse con un servidor de base de datos MySQL como cliente para guardar los datos que lleguen de los nodos BSN. El *software* es una interfaz que está pensada para interactuar con el usuario, a esto también se le denomina GUI por sus siglas en inglés *Graphical User Interface*.

La interfaz de usuario llamado Sistema de Acceso a la Red Inalámbrica de Sensores (SARIS) la cual está desarrollada en Java en la plataforma de NetBeans, los *scripts*[7] del sistema se programan en PHP[8] ya que Arduino puede ejecutarlos. Para lograr el objetivo, se utilizan bibliotecas (*libraries*) que están disponibles en la plataforma GitHub.

El diseño en la implantación del programa tiene las siguientes etapas:

1) **Requerimientos**. Punto en el que se detallan los alcances y limitaciones del software que incluye la forma en la que la interfaz va a interactuar con el usuario y la RIS. Por otro lado, se tiene que definir la base de datos y las tablas donde se va a guardar la información de los usuarios y la lectura de los sensores en los nodos.
2) **Diseño con diagramas UML**. Los diagramas UML permiten establecer un algoritmo bajo el paradigma de la programación orientada a objetos, pues se establece que hace el programa, los actores y se definen sus actividades o tareas.
3) **Instalar el servidor XAMPP**. En este punto se inicia la implantación del programa. El programa XAMPP es una distribución de Apache con licencia GPL (General Public License) que tiene el servidor, PHP y MySQL que es una plataforma para construir bases de datos sobre la que se crean las tablas que se consideraron para almacenar los datos.
4) **Crear los *scripts* PHP**. Para conectarse al servidor de base de datos e insertar las muestras tomadas en el sistema.
5) **Realizar la conexión Cliente – Servidor.** Es necesario para la comunicación entre Arduino, como cliente, y el servidor de base de datos MySQL.
6) **Codificar en Java**. Bajo la plataforma NetBeans considerando las bibliotecas que usará.
7) **Realizar el programa ejecutable**. Esto incluye hacer un archivo jar, un archivo sql, agregar los repositorios y finalmente realizar una versión ejecutable.
8) **Realizar un manual de usuario**. básicamente es un manual de instalación del sistema partiendo del nodo AN.

[7] **Script**. Es un archivo de texto que contiene los códigos de conexión con la base de datos y la inserción de muestras de los nodos de la RIS, éste se interpreta en el servidor Apache línea por línea.
[8] **PHP**. Es la sigla que proviene de su nombre en inglés *Hypertext PreProcessor* que es un lenguaje de código abierto orientado al desarrollo de páginas web.

3.1. Requisitos de la interfaz de usuario

El objetivo de SARIS es que sea una interfaz de usuario donde se puedan tomar muestras de los nodos BSN que se encuentran en la RIS; para poder hacer esta tarea los usuarios deben tener acceso a SARIS mediante el nombre de usuario y la contraseña. En el programa existen dos figuras, el Administrador y el Lector, el primero es un tipo de usuario que puede gestionar otros usuarios y tomar muestras de la red, mientras que el segundo es un tipo de usuario que sólo puede tomar muestras de la red y sólo tendrá acceso si el Administrador lo agrega al sistema.

Para realizar las tareas descritas, el sistema debe contar con un servidor de base de datos en el cual haya tablas específicas para los datos de usuario y las muestras de los nodos. Para que el programa funcione completamente se requiere la conexión de la GUI con el servidor de base de datos y la TAD del nodo AN, finalmente, también es necesario que la TAD se conecte con el servidor de base de datos para que sea quien almacene la información en la base de datos.

Al final se obtiene una Red Inalámbrica de Sensores y una interfaz de usuario hace posible tomar muestras de la lectura de los sensores en cada nodo de la red, esta última tiene una carpeta de instalación la cual tiene todos los archivos necesarios para configurar el nodo AN e instalar SARIS, entre los cuales se incluye un manual de usuario (Apéndice 1) que explica de forma detallada los pasos de configuración e instalación.

El programa SARIS funciona con el nodo AN encendido el cual debe incluir un shield de Ethernet para Arduino, un cable UTP categoría 6e en **par trenzado**[9] conectado a una computadora la cual tenga el programa XAMPP instalado y con un usuario definido para el acceso a la base de datos MySQL; en dicha base de datos deben estar definidas dos tablas, la primera es *userList* que es donde se almacenan los datos de los usuarios que tienen acceso al sistema, la segunda es *dataRX* que es donde se guardan las muestras que se toman de los nodos.

El ejecutable de SARIS deber ser ligero en cuanto al espacio de almacenamiento en la computadora, esto para poder exportar el programa a un **applet**[10] que se ejecute en una página web o a un dispositivo móvil si así se desea. En cuanto al almacenamiento el problema surge con el muestreo y el espacio en memoria de la base de datos, por lo que es beneficioso en este aspecto, delimitar al usuario para que tome la muestra en el momento que éste decida, pues es el usuario quien define horarios y número de muestreos.

Otro punto es que la batería recargable de 12 [V] le da una autonomía al sistema de poco más de 5 [hr], lo que implica que no puede estar encendido un día completo lo cual es conveniente si el usuario va a realizar las lecturas de forma manual. Este sistema tiene la desventaja de que no permite el monitoreo constante en todo momento así que en aplicaciones como un sondeo de cultivos resulta útil.

[9] **Par trenzado**. Es el término que se usa para decir cable UTP que es el cable de red llamado así por las 4 "trenzas" de dos cables que tiene dentro de la cubierta; otro término que significa lo mismo es **par cruzado**, aunque este se utiliza indistintamente a lo largo de este trabajo.
[10] **Applet**. Es un programa hecho en código Java para ejecutarlo en alguna página web.

3.1.1. Definir la base de datos y las tablas

Para disponer de un servidor de base de datos se opta por instalar XAMPP que es una plataforma de prueba y diseño de distribución libre del cual se utiliza Apache, MySQL y PHP, aunque tiene otras herramientas, estas son las que se utilizan para el desarrollo del programa SARIS. La función principal de la base de datos es almacenar los datos de usuarios y las muestras de los nodos, no se considera la integridad referencial de las bases de datos, ni las validaciones de las mismas.

Se proponen dos tablas una llamada userList que se encuentra en la base de datos denominada *userDB* (ver Tabla 3.1), su función es guardar los datos de los usuarios que tienen acceso al sistema; la segunda tabla es *dataRX* que está en la base de datos bancoArduinoPrueba (ver Tabla 3.2) la finalidad de esta tabla es que almacene las muestras de cada nodo BSN que recibe AN.

En la tabla *userList* se definen columnas que almacenen datos del tipo VARCHAR, que en MySQL se entiende que los datos son del tipo carácter, pero la longitud de los datos almacenados varían hasta el límite de caracteres definidos. El identificador *user_id* es del tipo entero (*Int*) y en cada registro abarca una memoria de 3 bytes, a diferencia del identificador del tipo de usuario *usr_tpe_id* que almacena el valor en un byte.

userList	
Columnas de la tabla	**Descripción**
user_id	Es un identificador único de usuario. Es un valor entero.
name	Es el nombre de usuario. Se trata de una variable del tipo VARCHAR con límite de hasta 50 caracteres.
surname	El apellido de usuario. Se trata de una variable del tipo VARCHAR con límite de hasta 50 caracteres.
email	El correo electrónico del usuario. Se trata de una variable del tipo VARCHAR con límite de hasta 60 caracteres.
nickname	Es el sobrenombre del usuario con el que se identifica. Se trata de una variable del tipo VARCHAR con límite de 12 caracteres.
password	Es la contraseña del usuario. Se trata de una variable del tipo VARCHAR con límite de 12 caracteres.
usr_tpe_id	Es el identificador único del tipo de usuario. Es un valor entero.

Tabla 3.1. Columnas de las tablas *UserList*.

dataRX	
Columnas de la tabla	**Descripción**
data_id	Es el identificador único del dato guardado. Es un valor entero.
Event	Es la columna donde se guarda el momento en el que se guardó la muestra, registrando tanto fecha como hora, esta columna evento es la estampa del tiempo por ello es del tipo *TIMESTAMP*.
nodo_id	Es el identificador único de nodo del que se reciben datos, ya que en la base de datos están identificados con valores enteros. Éste está asociado al identificador del nodo que se encuentra en el primer elemento del arreglo *payLoad*.
Humidity	Es el campo que guarda el valor recibido de los nodos BSN sobre la humedad, estos datos se reciben como valores enteros.
Temperature	Es la temperatura ambiente, es el valor que se recibe del dispositivo DHT11, es de del tipo entero, aunque la información ya se encuentra en grados Celsius.

Tabla 3.2. Columnas de la tabla *dataRX*.

3.2. Análisis del problema

Para diseñar la GUI denominada SARIS, se plantea el sistema desde la perspectiva de un usuario, es decir, si fuese un usuario que es lo mínimo que se espera para el sistema SARIS, este punto lleva a no sólo tener una pantalla de toma de muestras tanto para usuarios Lector como Administrador, sino que esto incluye la administración de usuarios.

La administración de usuarios implica guardar, actualizar y eliminar usuarios en la base de datos *userList*:

i. **Guardar datos de un nuevo usuario**. Para guardar un nuevo usuario y darle acceso al sistema de SARIS para que tome lecturas de los nodos BSN, se debe ingresar: nombre (s), apellido (s), correo electrónico, nombre de usuario, contraseña, y tipo de usuario. El tipo de usuario es el Administrador o el Lector, el primero se identifica con el valor entero 1 y el segundo con el 2.

ii. **Mostrar usuarios Lector existentes en una vista**. La vista de usuarios es una tabla con los datos de éstos que se muestra en la pantalla del programa SARIS, este elemento permite que el Administrador vea los usuarios que tienen acceso al sistema y verifique los cambios que se realicen sobre los mismos.

iii. **Actualizar datos de un usuario existente**. Para hacer esta actividad el Administrador debe seleccionar a un usuario para realizar los cambios de datos en un formulario.

iv. **Borrar un usuario existente**. Esta tarea borra definitivamente a un usuario Lector y el evento es irreversible.

Para tomar muestras de los nodos BSN desde el programa SARIS, éste debe dar la posibilidad de hacer una solicitud para guardar una muestra de un nodo específico, para lograrlo la solicitud pasa por la TAD y desde aquí se solicita al servidor Apache que se ejecute el *script* que realiza cambios a la tabla *dataRX* insertando los datos que se recogieron para que posteriormente el servidor los entregue a la interfaz en Java.

En este punto se utiliza el paradigma cliente – servidor, ya que tanto SARIS como la TAD juegan el papel de cliente en el sistema haciendo las solicitudes necesarias al servidor de base de datos MySQL. Al realizar las actividades definidas requiere usar consultas (*queries*) en la base de datos como insertar (*Insert*), actualizar (*Update*) y borrar (*Delete*) los registros de las tablas definidas.

La Figura 3.1 resume de forma general como funciona el programa SARIS, por un lado la TAD del nodo AN recibe datos de los nodos BSN y las solicitudes de la interfaz en Java que se muestra como cliente GUI, y entrega datos para guardarlos mediante una consulta que realiza un *Script* en PHP. También la base de datos en MySQL recibe solicitudes de SARIS y le entrega el resultado de las consultas.

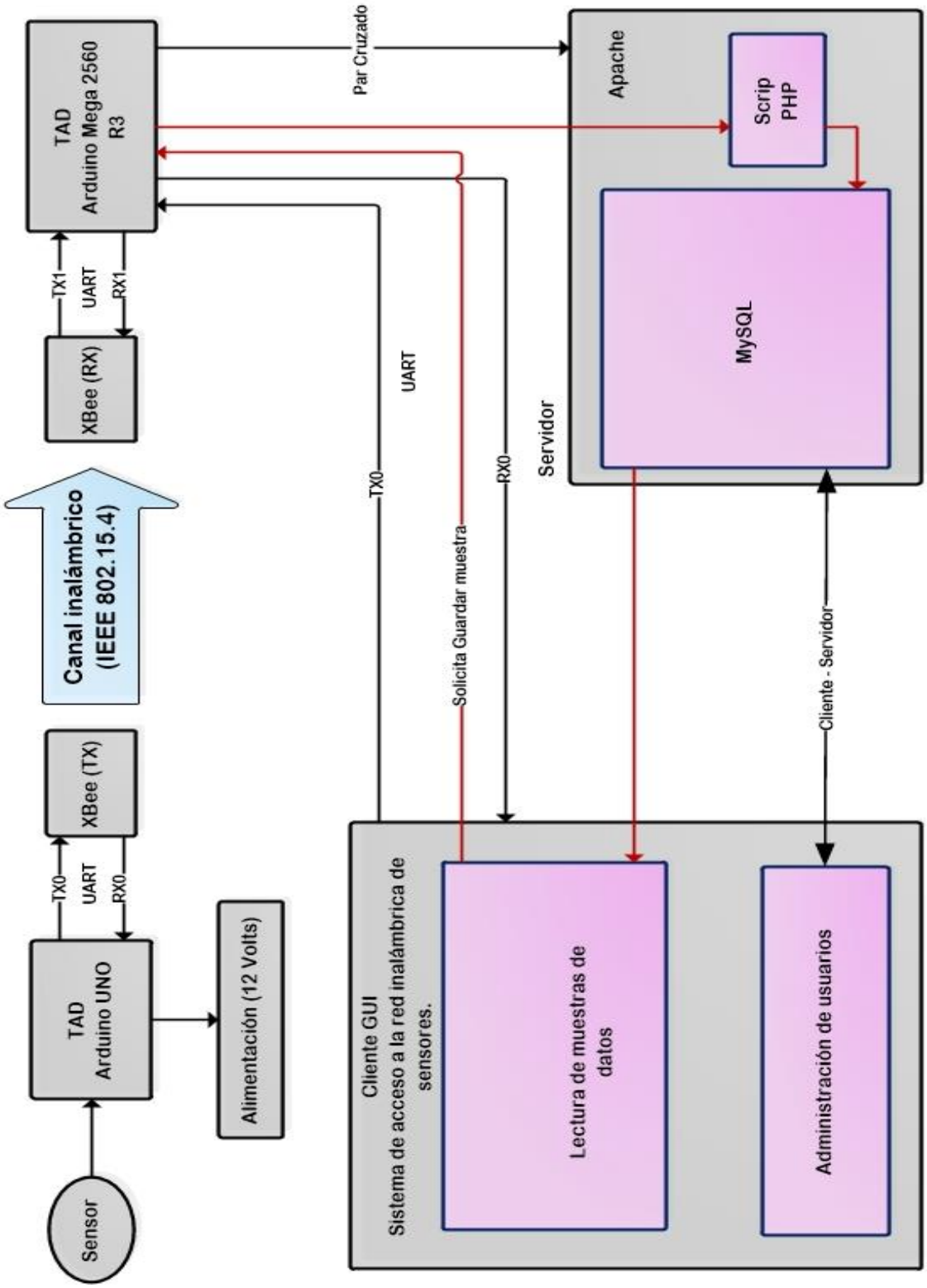

Figura 3.1. Esquema general de SARIS.

3.3. Diseño e implantación de la interfaz de usuario

El programa no incluye la instalación del servidor de base de datos, para ello se utiliza XAMPP que es una distribución de **Apache**[11] con licencia que incluye MySQL y PHP. Hay dos conexiones a realizar, la primera es de la interfaz con el servidor MySQL para lo que se utiliza un biblioteca *mysql-connector-java-5.1.38-bin* y la segunda es conectar a SARIS con Arduino Mega en la cual se utiliza la biblioteca *PanamaHitek_Arduino-2.8*.2.

3.3.1. Diagramas *Unified Modeling Lenguage*

Los diagramas UML van asociados a las tareas que debe cubrir la interfaz, por ejemplo, para realizar una autenticación hay que considerar que en la pantalla de inicio se ingresa el nombre de usuario y la contraseña y después se presiona un botón de ingreso que activa las clases que realizan el proceso de verificar que la contraseña coincida con el usuario para dar acceso al sistema SARIS.

Los diagramas de actividades están principalmente enfocados a tres funcionalidades la primera es la autenticación de usuarios, la segunda es la administración de usuario para el usuario Administrador y finalmente, la tercera es la toma de muestras de los sensores que se encuentran en cada nodo de la red; estas deben verse reflejadas en los diagramas UML.

La plataforma en la que se codifica la interfaz de usuario SARIS es NetBeans 10.0 utilizando el lenguaje Java para construir los objetos definidos en el diseño del software. En este punto comienzan a definirse los paquetes, clases, métodos y variables por medio de los diagramas UML en los que los nombres se asignan en inglés. Así al usuario Administrador en UML y en el código se le denomina *Manager*, mientras que al Lector se le llama *Reader*.

3.3.1.1. Diagramas de caso de uso

Para el proceso de autenticación el primer objetivo a realizar son las validaciones en los campos de texto de la pantalla de inicio (Figura 3.2) el sistema sólo admite números y caracteres del abecedario en mayúscula y minúscula, si se teclea cualquier otro carácter el sistema emite un mensaje de error, tampoco debe permitir campos de texto en blanco. Para corregir un posible error de captura se utiliza el botón limpiar que lo único que hace es borrar todo el texto hasta el momento capturado.

El segundo objetivo es identificar si el usuario puede ingresar al sistema, para ello se cuenta con una base de datos en la que se tiene un registro de los usuarios que pueden ingresar; en este punto se realizan las validaciones del usuario que incluye el tipo de usuario ya sea Administrador o Lector. Esta pantalla se crea con la finalidad de tener la función definida, sin embargo, las contraseñas que se capturan se usan en **texto plano**[12], ya que no se está considerando la parte de seguridad en el desarrollo de SARIS.

[11] **Apache**. Es un servidor web que sirve para utilizar el protocolo HTTP, éste es de código abierto para plataformas Unix, Linux y Windows.
[12] **Texto plano**. En este texto, se refiere a que la contraseña está en texto simple y sin forma alguno, es la contraseña tal cual sin encriptar.

Cada actor en los diagramas de caso de uso equivale a una clase en Java la cual pertenece a un paquete. Los diagramas de la Figura 3.2 y la Figura 3.3 muestran un conjunto de actores que pertenecen al paquete *modelUser* el cual tiene definido al actor *LoginUser*, *Manager* y *Reader*, de los cuales *LoginUser* inicia el proceso de autenticación.

También dentro de *modelUser* se encuentra el paquete *readNetwork* el cual tiene el actor *ReadSensor* que es capaz de conectarse con la TAD, tomar muestras de los sensores que se encuentran en los distintos nodos BSN y mostrar una tabla de lectura. El actor *ReadSensor* interactúa con *Reader* y *Manager* debido a que ambos pueden tomar muestras de la RIS.

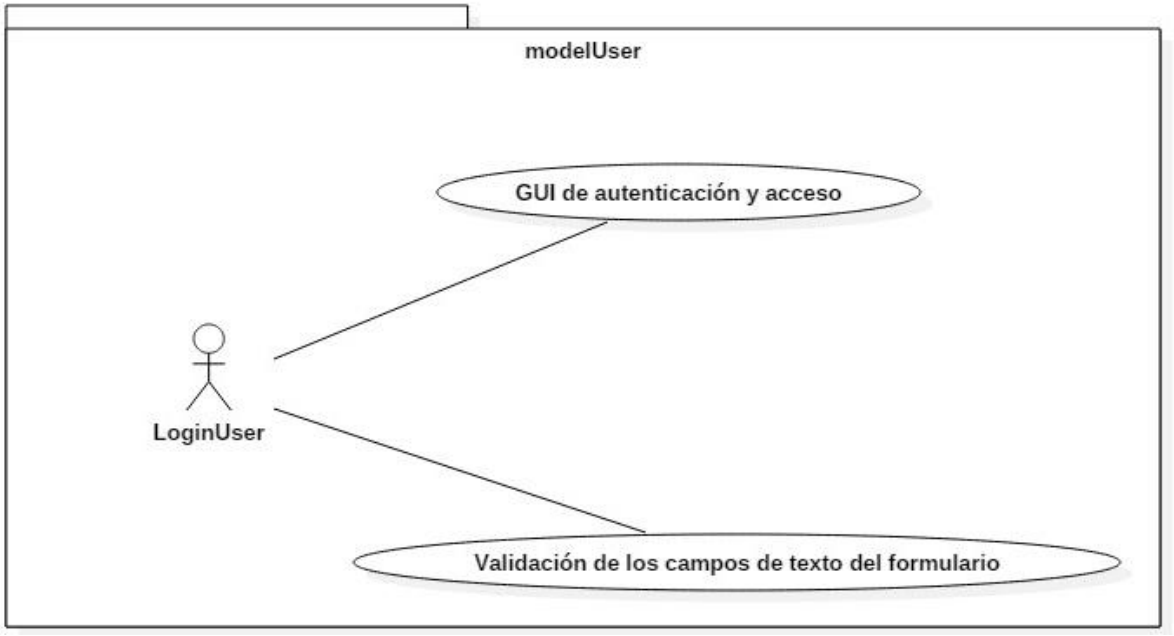

Figura 3.2. Diagrama de caso de uso de la pantalla autenticación.

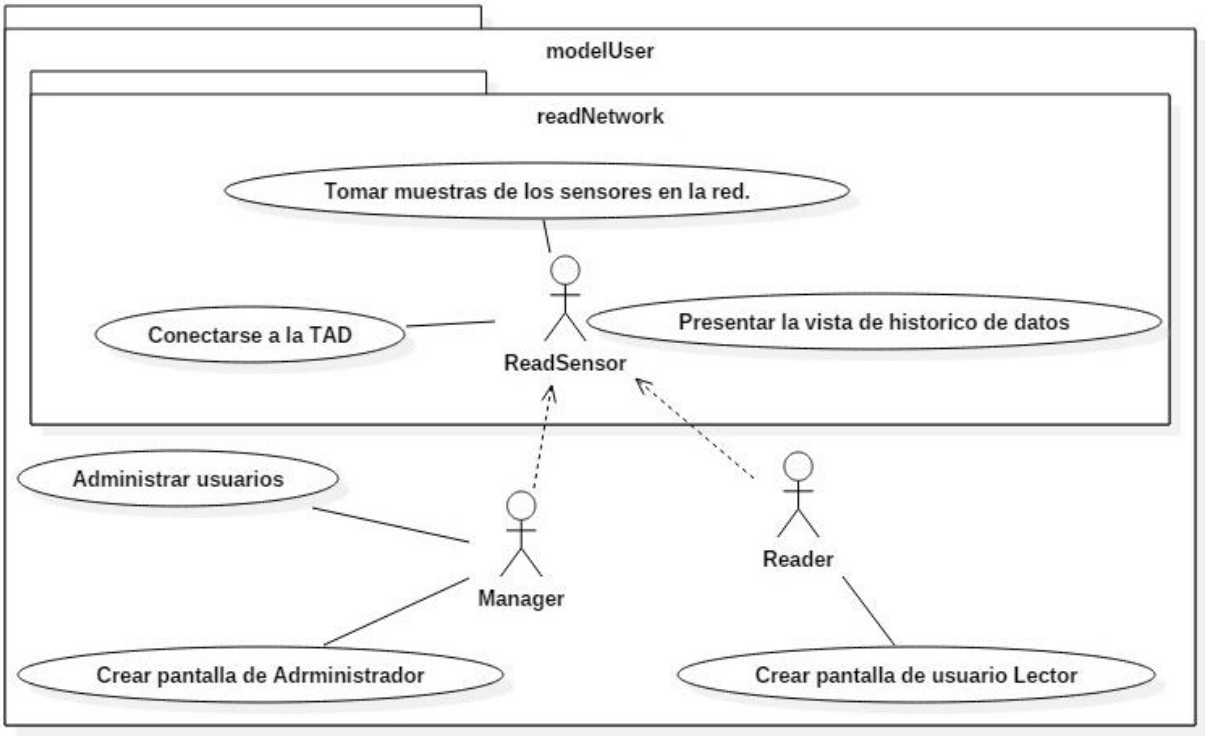

Figura 3.3. Diagrama de caso de uso de las pantallas del sistema SARIS.

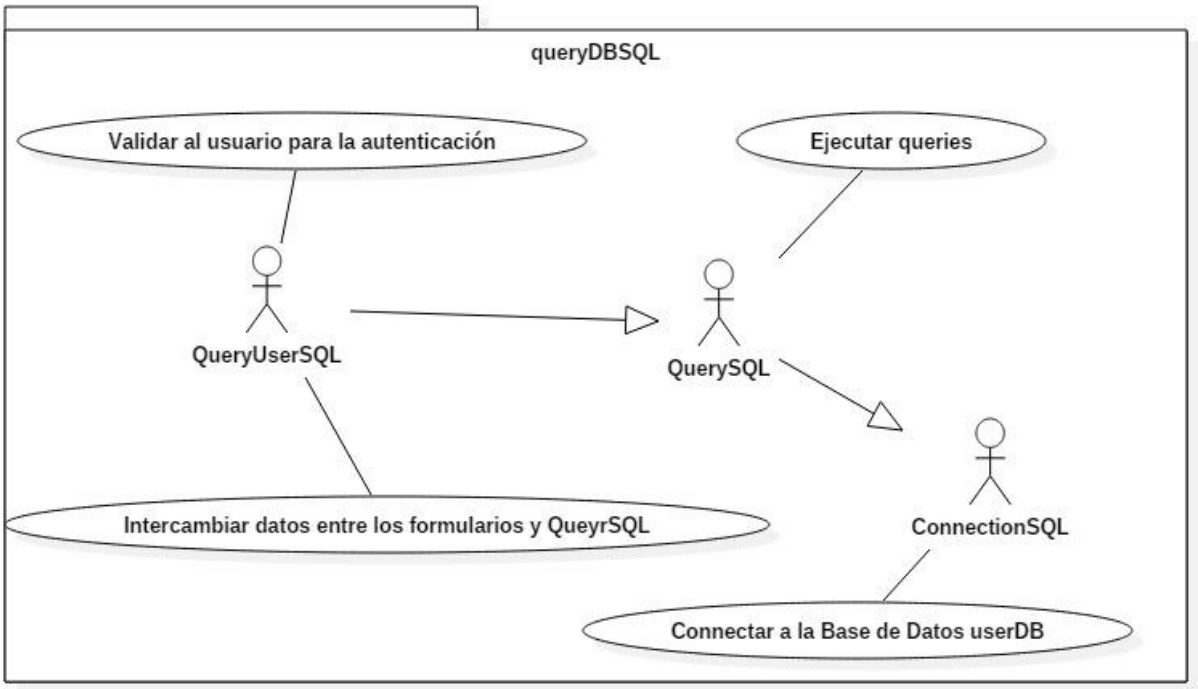

Figura 3.4. Diagrama de caso de uso de los actores que ejecutan las consultas.

En el paquete *queryDBSQL* se encuentran el resto de los actores que hacen posible la autenticación y el insertar, actualizar y eliminar usuarios. El actor *ConnectionSQL* tiene como única función realizar la conexión con la base de datos *userDB*; el actor *LoginUser* y *Manager* son los únicos que deben tener acceso a esta base de datos. Por su parte, el actor *QueryUserSQL* es quién realiza la validación de usuario en el proceso de autenticación e intercambia datos entre los actores del paquete *modelUser* y el actor *QuerySQL* que es quién ejecuta las consultas en la base de datos (ver Figura 3.4).

Estos primeros diagramas UML permiten definir no sólo los actores que serán las clases en el código Java sino la organización de los paquetes como se observa en la Figura 3.5, al proyecto en *NetBeans* con el que se codifica SARIS se le denomina RedWSN15 del cual se desprenden los paquetes dentro de los cuales se encuentran las clases que se definieron por los actores de los diagramas de caso de uso.

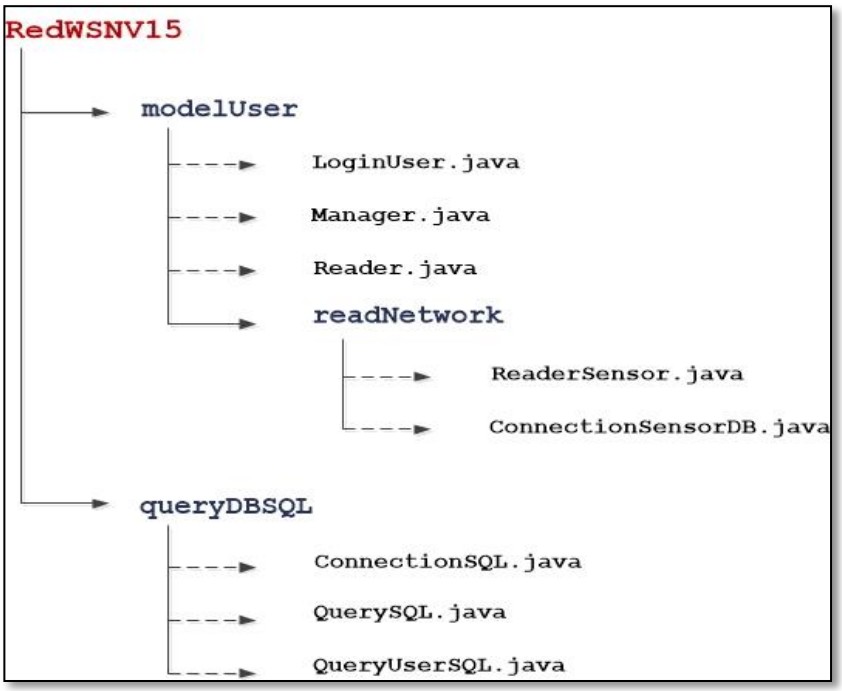

Figura 3.5. Distribución de paquetes y clases en el programa en Java.

3.3.1.2. Diagramas de Actividades

Para la autenticación hay dos puntos de decisión, el primero es que coincida en el registro de la tabla *userList* el nombre de usuario y la contraseña, de no ser así no se puede considerar que el usuario se haya autenticado y se debe mostrar un mensaje de error y la indicación al usuario de lo que debe hacer para corregir el error, en este caso se muestra: “Usuario y/o contraseña no existen”.

El diagrama de actividades de la Figura 3.6 se muestra el algoritmo que resuelve el proceso de autenticación, se puede ver que se hace una validación del tipo de usuario y que el usuario Administrador entra a su sistema de gestión que implica administrar usuarios y tomar muestras, mientras que el usuario Reader sólo puede realizar la segunda. El proceso para las validaciones de los campos de texto en la pantalla de inicio se puede ver en la Figura 3.7.

En pantalla del Administrador definido por Manager se puede solicitar el insertar, actualizar o eliminar usuarios como lo muestran los algoritmos que están en las figuras 3.8, 3.9 y la 3.10. En el paquete *queryDBSQL* también se desglosan las tareas que involucran el recibir y entregar datos asociados a la administración de usuarios, donde las actividades asociados a los casos de uso del actor *QuerySQL* están enfocadas en ejecutar las consultas necesarias para realizar las modificaciones en la base de datos.

Figura 3.6. Diagrama de actividades para la autenticación.

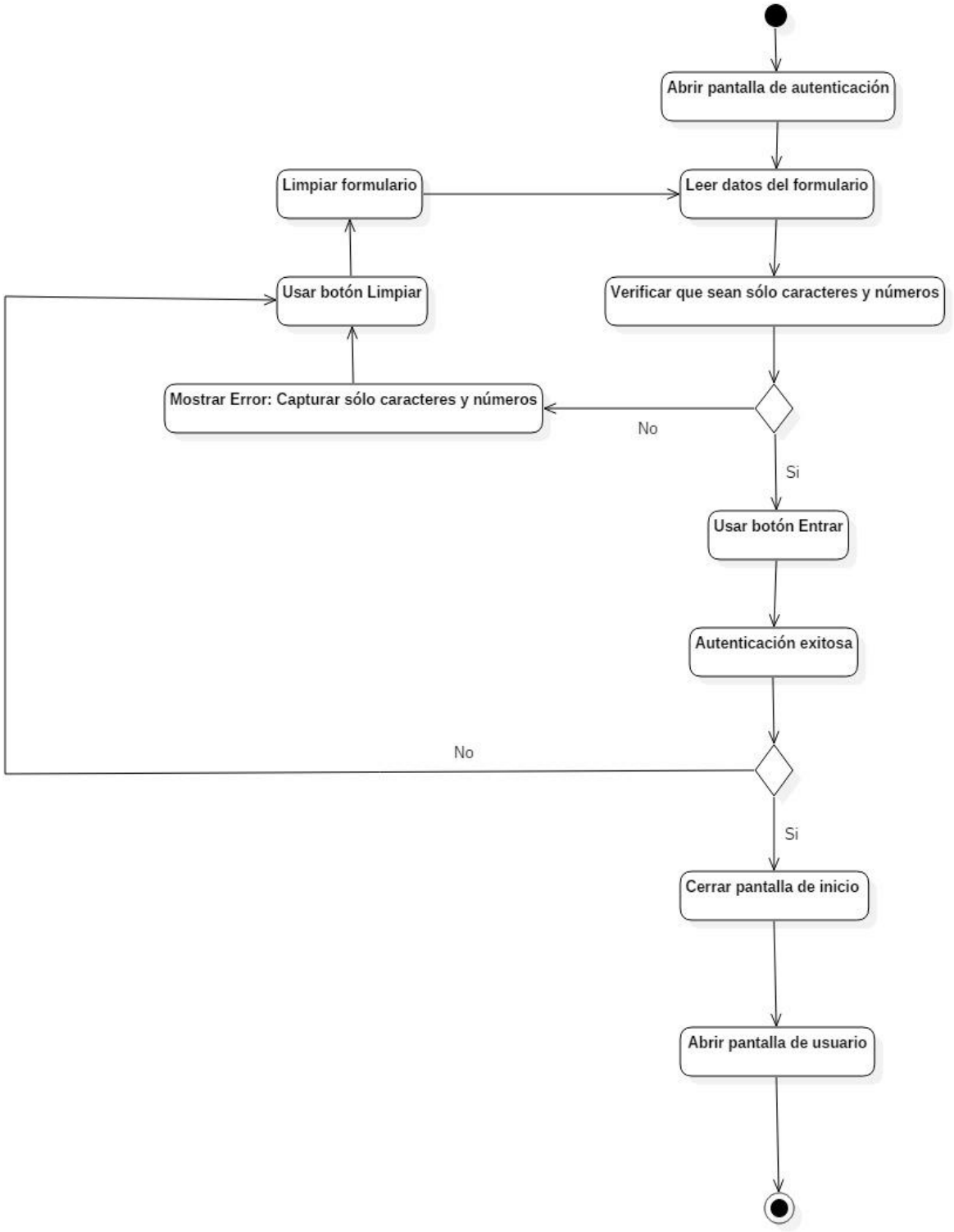

Figura 3.7. Diagrama de actividades para validar los caracteres en los campos de texto del formulario.

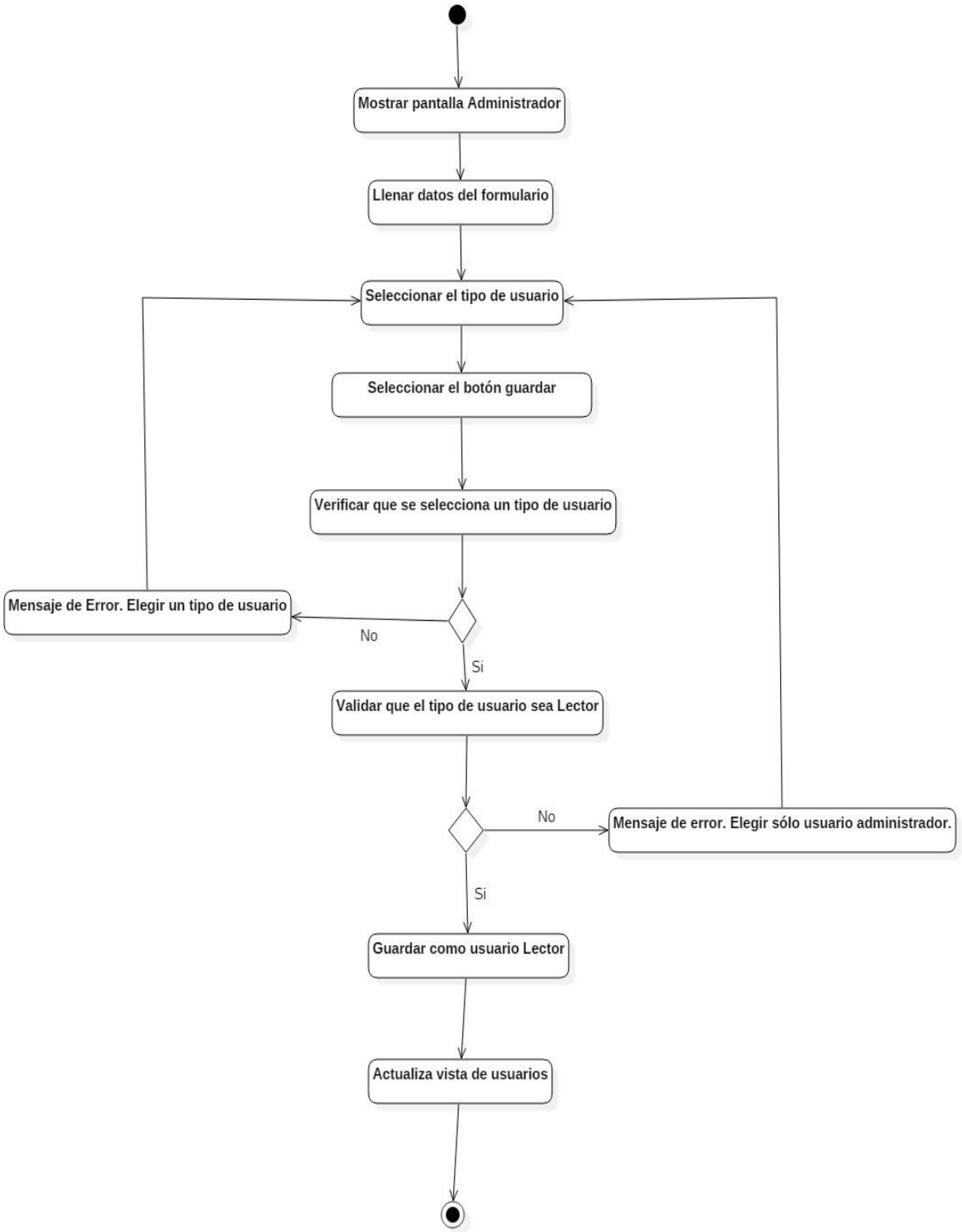

Figura 3.8. Inserción de un nuevo usuario en un registro de la tabla en la base de datos.

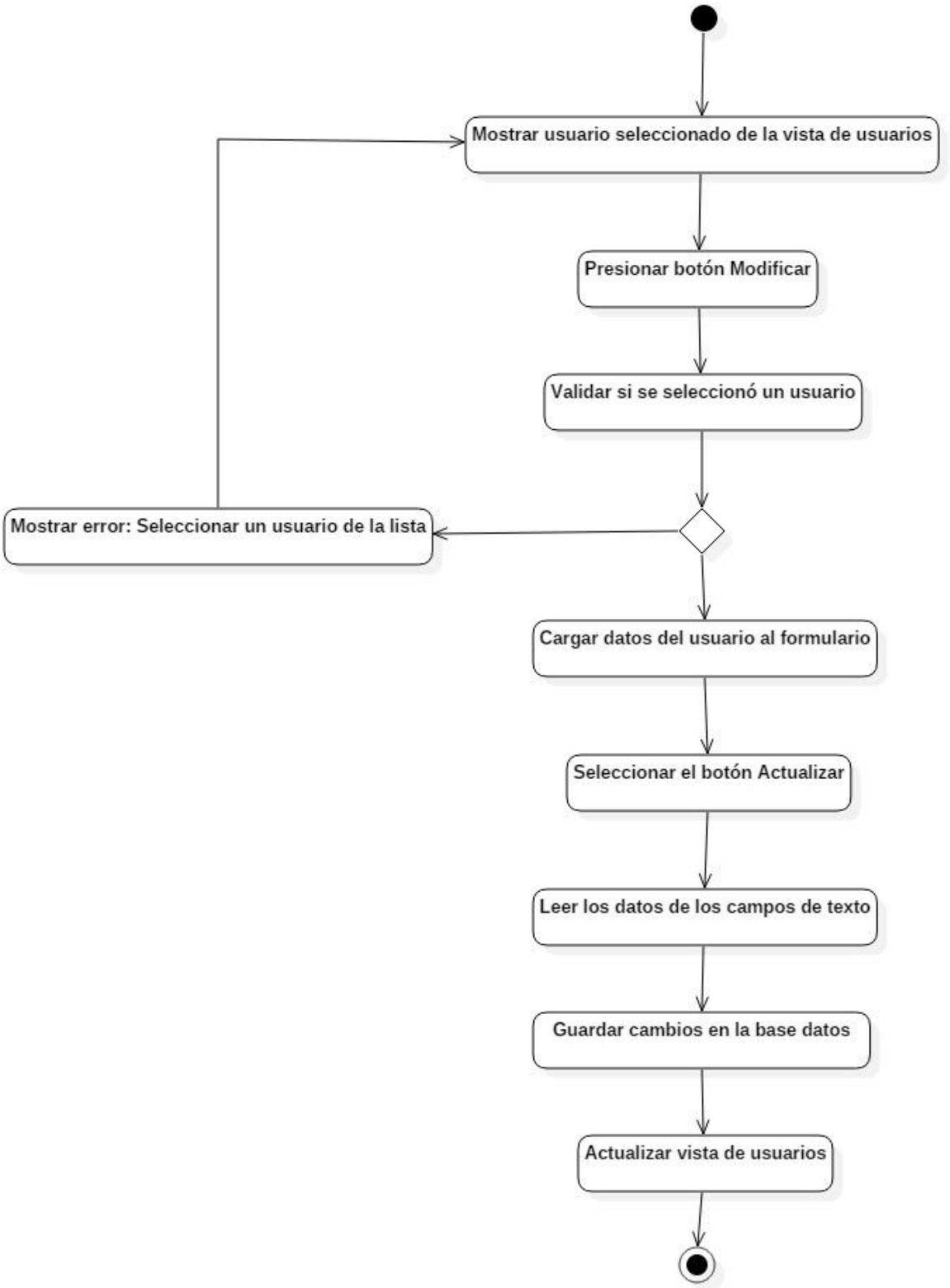

Figura 3.9. Actualización de los datos de un usuario.

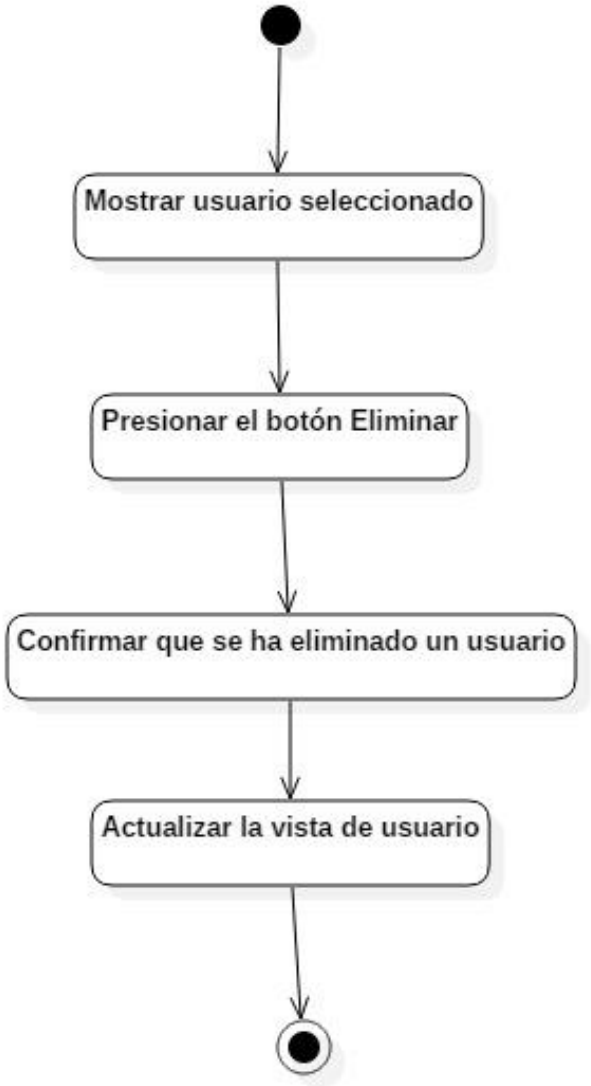

Figura 3.10. Eliminación de un usuario.

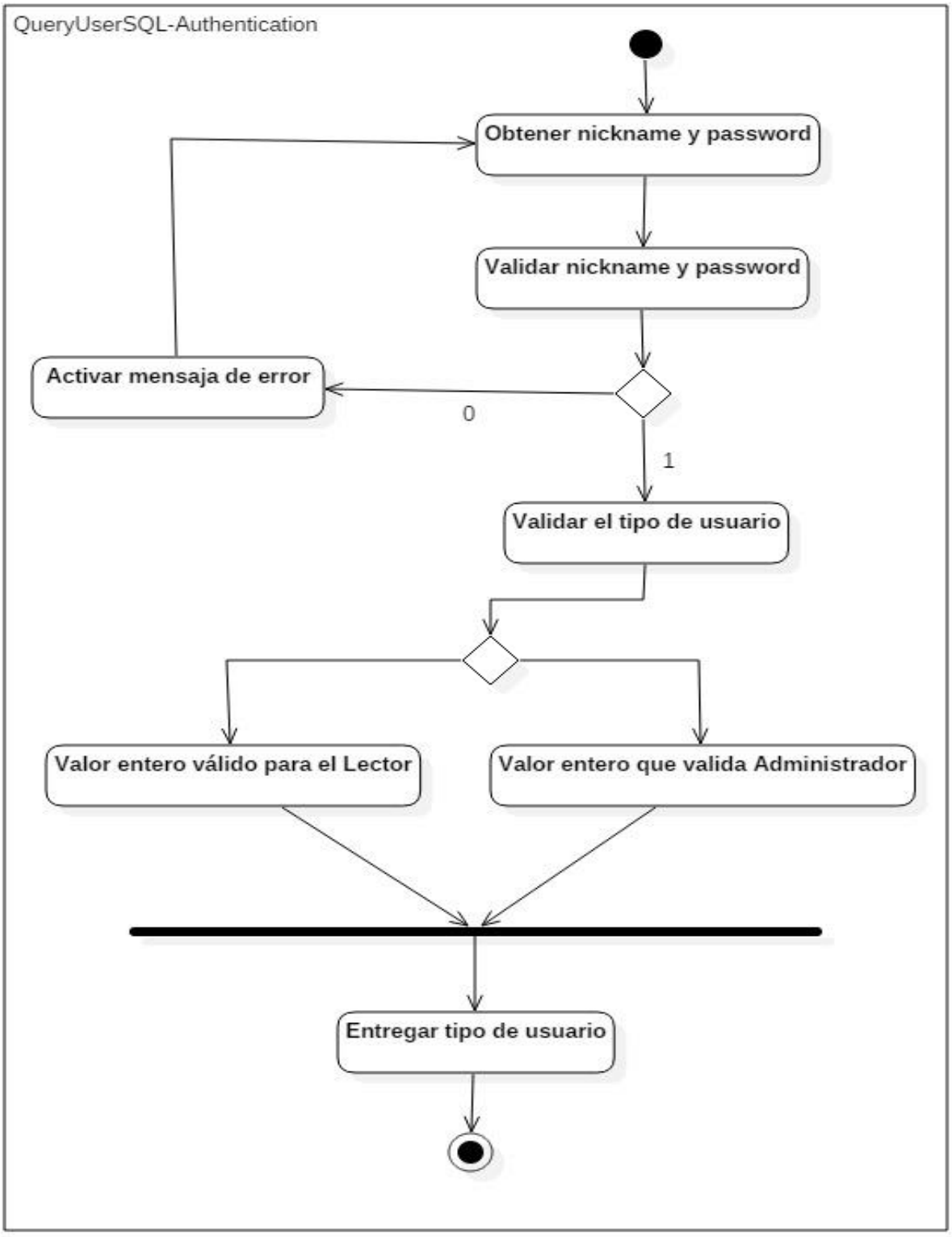

Figura 3.11. Validaciones en el proceso de autenticación.

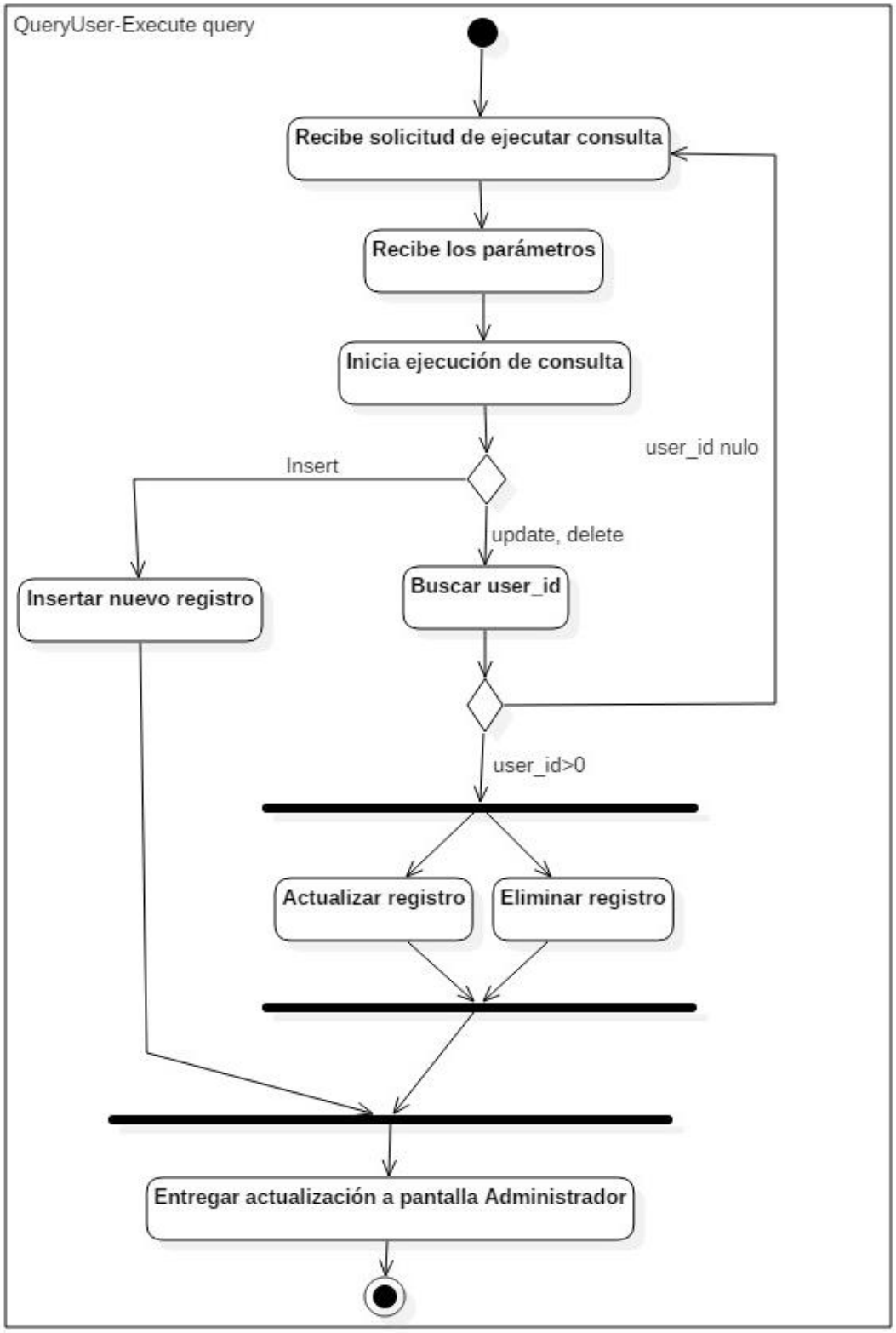

Figura 3.12. Recepción y entrega de datos para ejecutar de consultas en la base de datos.

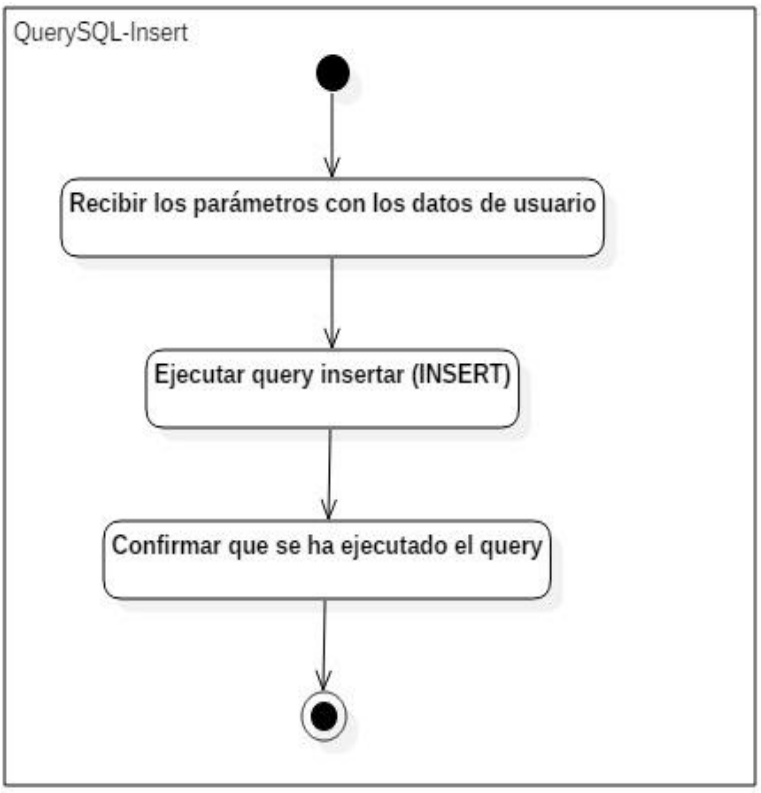

Figura 3.13. Diagrama de actividades para insertar un nuevo usuario.

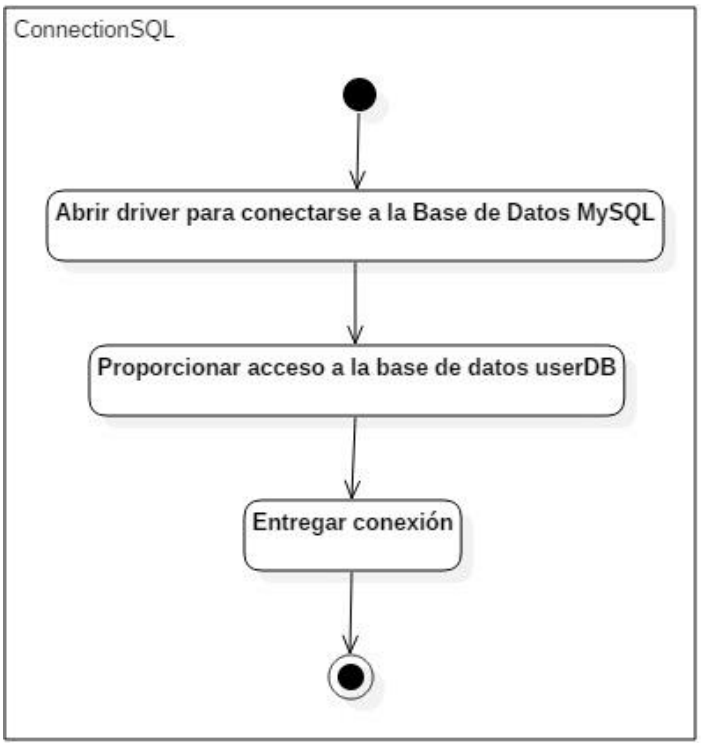

Figura 3.14. Diagrama de actividades para conectar a SARIS con la base de datos userDB.

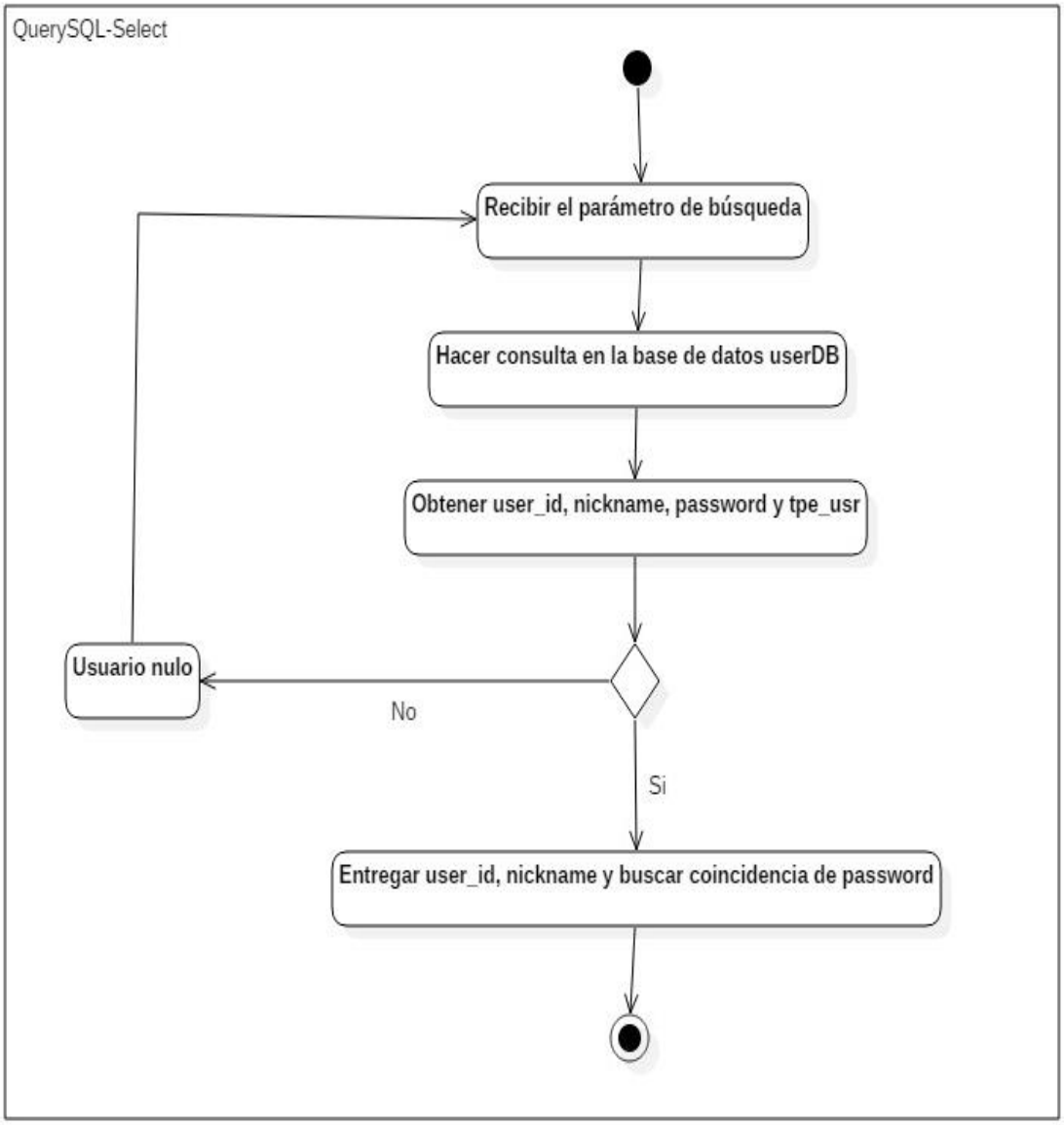

Figura 3.15. Diagrama de actividades para hacer una consulta en la tabla de userList.

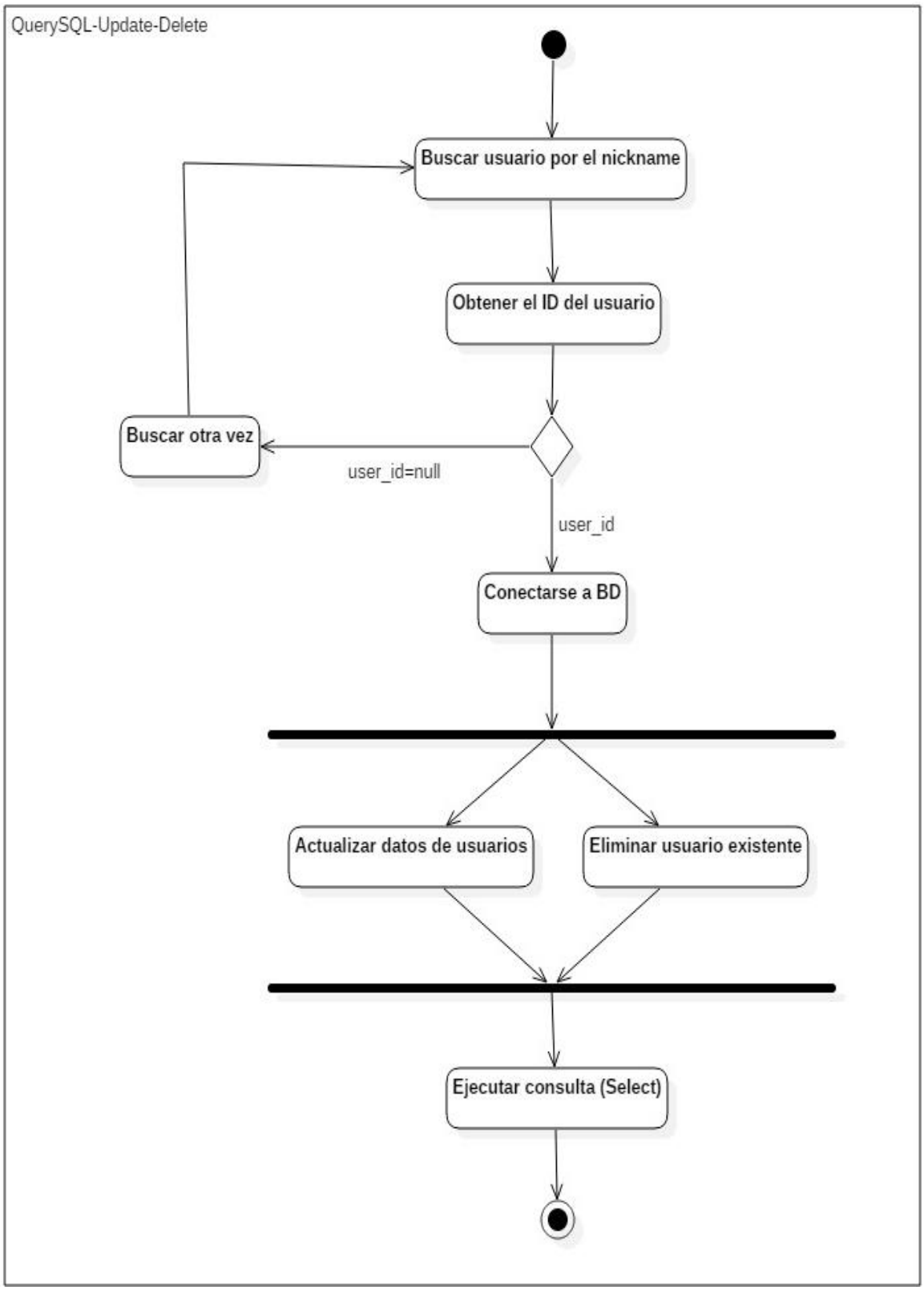

Figura 3.16. Diagrama de actividades para eliminar o actualizar los datos de un usuario.

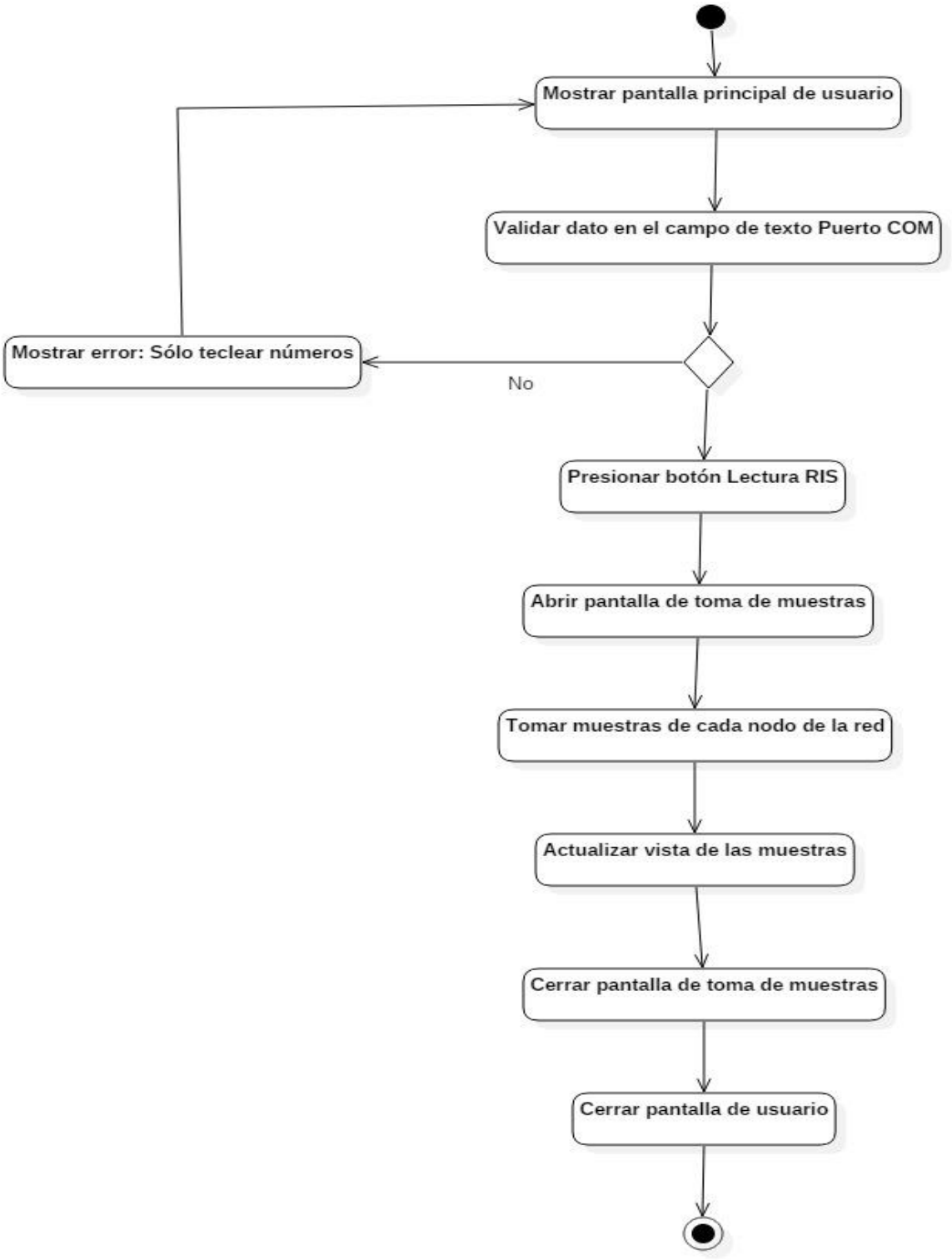

Figura 3.17. Diagrama de actividades para tomar las muestras en los nodos BSN de la RIS.

En la Figura 3.17 se muestra el diagrama de actividades que describe el proceso de tomar muestras de los cuatro nodos BSN de la RIS construida. Para ambos usuarios se define una pestaña denominada Principal con la cual se puede abrir al actor *ReadSensor* que define la clase que realiza las muestras. En el proceso el usuario debe teclear el número del puerto COM al que se conecta el Arduino Mega con la computadora, esto porque este número varía en cada equipo; aquí la única validación que se realiza es si se ha tecleado un número pero no valida si es el puerto correcto.

3.3.1.3. Diagrama de secuencias

El diagrama de secuencias (Figura 3.18) muestra la forma en que interactúan las clases *LoginUser*, *QueryUserSQL*, *QuerySQL*, *ConnectionSQL*. La clase *LoginUser* es la que inicia el proceso de autenticación pues es en esta clase que se tiene la pantalla que muestra al usuario que pide el nombre de usuario (*nickname*) y contraseña de usuario (*password*) se envían a la clase *QueryUserSQL* que ordena los datos recibidos para enviarlos a un método de la clase *QuerySQL* quien se encarga de ejecutar la consulta, para poder hacer esto la clase *ConnectionSQL* es quien se encarga de conectarse a la base de datos ver Figura 3.14.

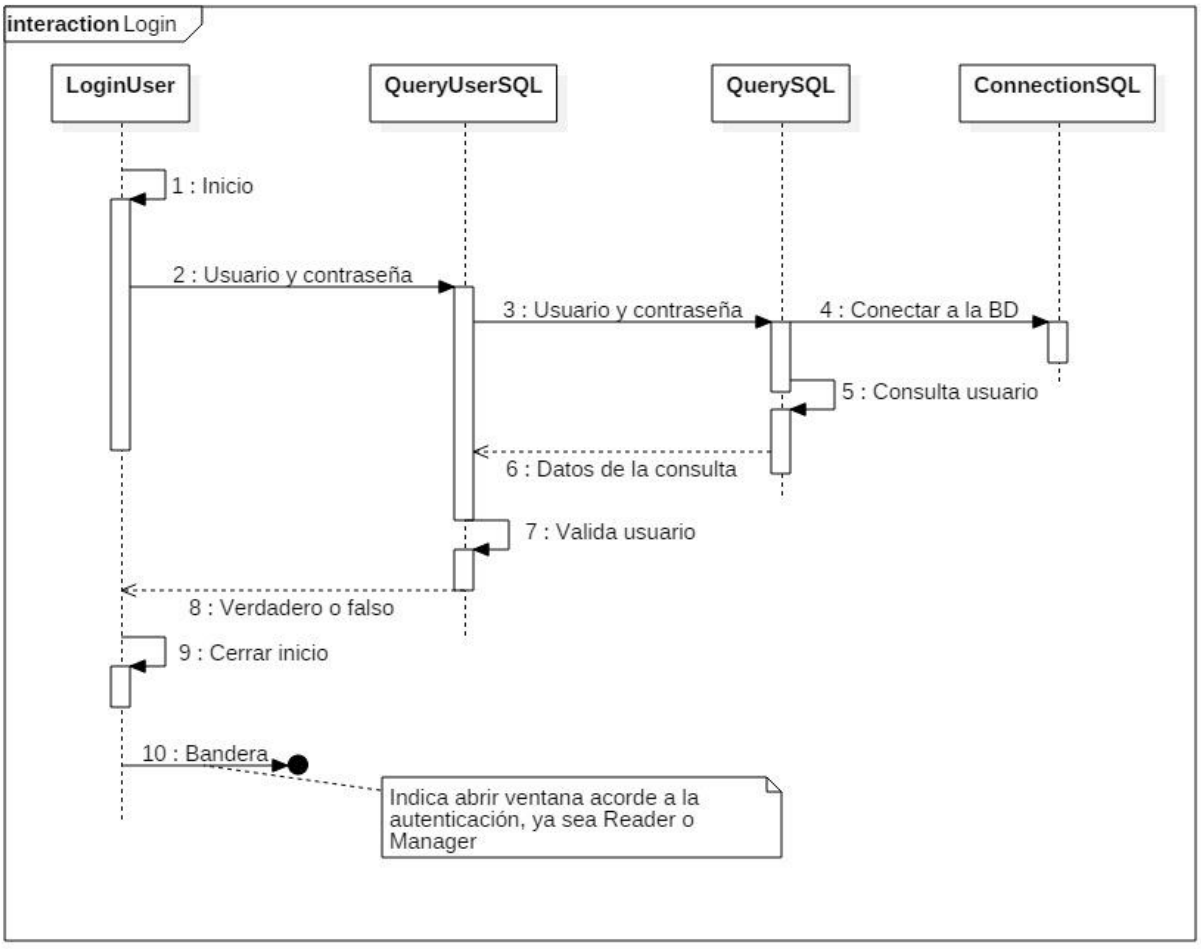

Figura 3.18. Diagrama de secuencia para la autenticación.

3.3.2. Configuración del servidor de base de datos

La función del Administrador incluye la gestión de usuarios, por lo que la tabla userList se crea en una base de datos propia que es llamada *userDB* y para la tabla *dataRX* se crea en una base de datos llamada bancoArduinoPrueba. Para acceder a las bases de datos del servidor se crean dos cuentas de usuario y sus respectivas contraseñas por lo que se utilizan dos cuentas.

El actor *ConnectionSQL* que realiza la conexión es quién tiene la información del usuario y la contraseña correspondiente a *userDB* para poder acceder a la tabla userList, y un segundo usuario con su respectiva contraseña puede ingresar a bancoArduinoPrueba que es la base de datos que utiliza el actor *ReadNetwork* y el *script* PHP que se encuentra en el directorio del servidor Apache.

Para crear las tablas se ejecuta el comando *CREATE TABLE* (crear tabla) y se crean los campos de acuerdo a lo definido en la Tabla 3.1 y Tabla 3.2, se establece que el campo donde se guarda el identificador de usuario (*user_id*) y el identificador de datos (*data_id*) sean incrementados de forma automática al insertar un nuevo registro, esto es el comando *AUTO_INCREMENT* que se observa en la Figura 3.19 al crear la tabla *userList*.

```
CREATE TABLE userList (user_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
                       name VARCHAR(50) NOT NULL, surname VARCHAR(50) NOT NULL,
                       email VARCHAR(60) NOT NULL, nickname VARCHAR(12) NOT NULL,
                       password VARCHAR(12) NOT NULL, usr_tpe_id INT(1));
```

Figura 3.19. Consulta para crear la tabla *UserList*.

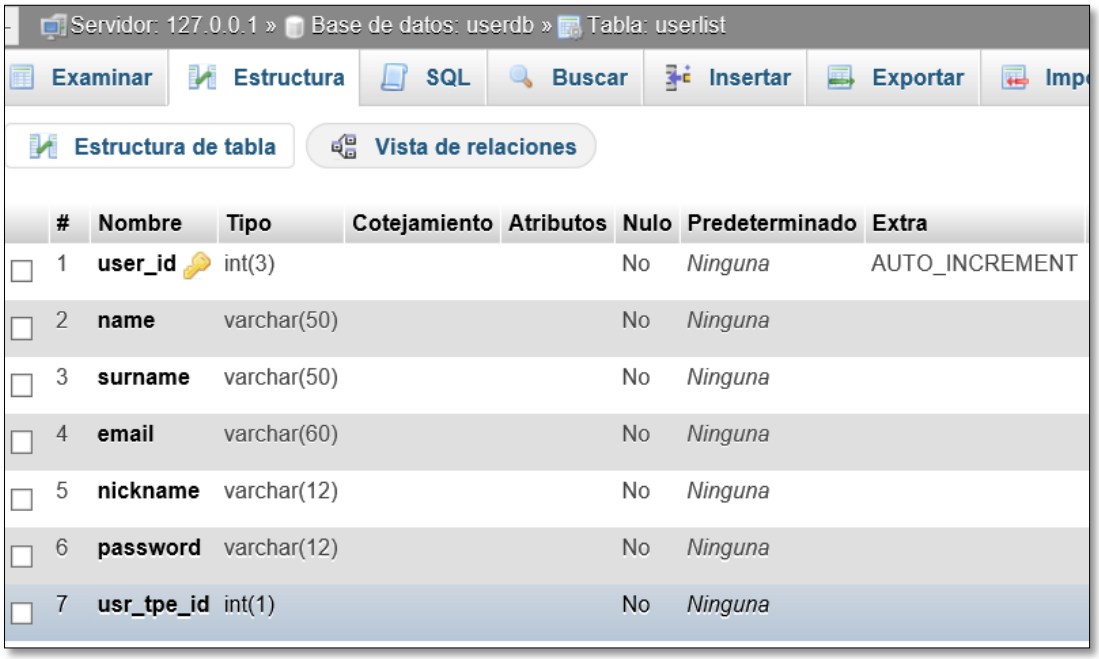

Figura 3.20. Tabla userList en el servidor de base de datos userDB.

3.3.3. Comunicación con la base de datos desde un *script* en PHP

El nodo AN recibe los datos de los nodos BSN constantemente, sin embargo hasta que el usuario solicita guardar una muestra es que se insertan los datos de la muestra del nodo solicitado. Cuando el usuario presiona un botón S1, S2, S3 o S4 lo que el sistema hace es enviar una valor entero 1, 2, 3, y 4 respectivamente, valores que están ligados al identificador del nodo d que se utiliza para construir la RIS, pero en esta etapa se le denomina nodo_id para llevar los registros en la tabla *dataRX*.

Cuando la TAD del nodo AN recibe el valor entero interpreta que debe tomar el valor del identificador del nodo al que se le solicitó, por lo que toma los datos del arreglo, y mediante el puerto 8080 le solicita al servidor que ejecute el *script* que se llama *saveDataRX.php* y se encuentra en el directorio que se muestra en la parte superior izquierda de la Figura 3.21.

El algoritmo que sigue el *script* que se utiliza en esta implantación corresponde al código que se muestra en la Figura 3.21 y se divide en los siguientes pasos:

i. Incluye al archivo “conecta.php” que se encarga de conectarse a la base de datos bancoArduinoPrueba usando los datos de unos de los usuarios creados en el administrador de MySQL.
ii. Se crean las variables que almacenan los datos recibidos por el puerto 8080, este punto es importante porque es la parte que se encarga de recibir los datos de las muestras tomadas en el nodo AN.
iii. Genera la consulta que se encarga de insertar los datos en la tabla dataRX.
iv. Muestra la consulta completa con los datos recibidos.
v. Ejecuta la consulta en MySQL.
vi. Muestra el mensaje “Datos guardados”.

La dirección web que se ve en el código de la Figura 3.21, se encuentra comentado porque con éste se prueba en un navegador de internet para verificar que el *script* funcione, sin embargo, esta misma dirección se replica en la TAD del nodo AN para que concatene la información recibida con la estructura correspondiente como se observa en la función de *writeData*() de la Figura 3.22.

La función *writeData*() recibe los datos del arreglo *payLoad*, si el cliente Arduino se encuentra conectado al servidor mediante el puerto 8080, usando la función *print*() se concatena el directorio donde se encuentra el archivo PHP y los datos del arreglo identificador del nodo, humedad y temperatura.

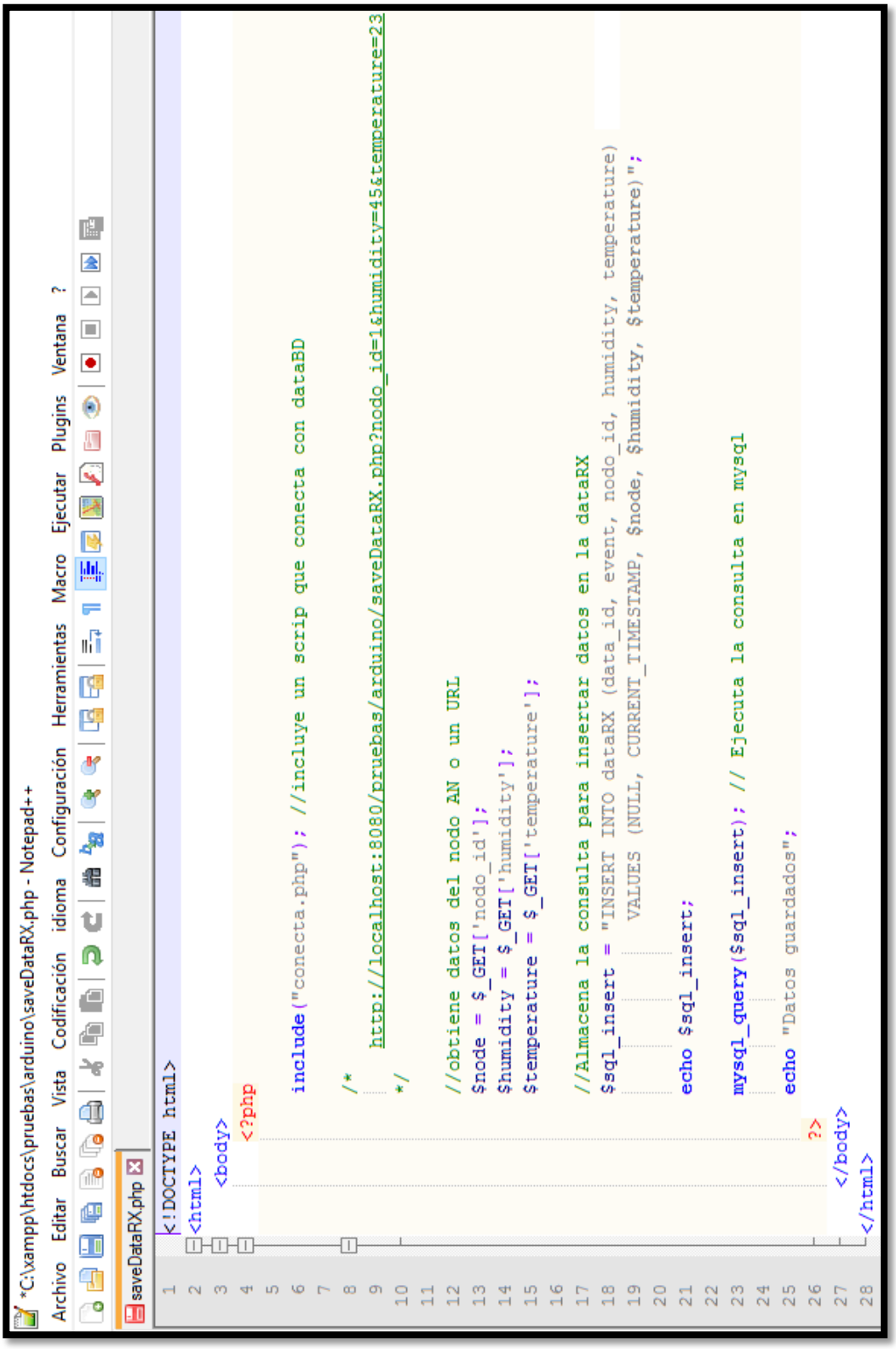

Figura 3.21. Código PHP que inserta los datos de una muestra en la base de datos.

Esto funciona tanto en el navegador como en Arduino Mega porque el servidor Apache es configurado para que las peticiones del cliente se escuchen en el puerto 8080, así que si se utiliza el navegador se tiene que hacer referencia a la máquina local (*localhost*) y el puerto, mientras que la TAD del nodo AN está configurado como un cliente que se comunica al servidor Apache en el que se encuentra la base de datos mediante el mismo puerto.

```
void writeData(int d, int h, int t){
  if(client.connect(server, 8080)){
    client.print("GET /pruebas/arduino/saveDataRX.php?node_id=");
    client.print(d);
    client.print("&humidity=");
    client.print(h);
    client.print("&temperature=");
    client.println(t);
    Serial.println("PHP ejecutado");
  }
}
```

Figura 3.22. Función en Arduino que ejecuta el *script* en PHP para insertar datos en la tabla dataRX.

3.3.4. Nodo AN como Cliente

Conectar la RIS implementada con el programa SARIS basta para la comunicación serial y la biblioteca en Java correspondiente, pero para que el nodo AN pueda realizar solicitudes al servidor de base de datos es necesario configurar la TAD con la computadora como una red LAN bajo TCP/IP. Para ello se usa un módulo de conexión a Ethernet para Arduino con el que es posible asignar la dirección IP, como la dirección MAC del módulo para Arduino.

	Servidor Base de Datos	**Cliente Arduino**
Dirección MAC	Dir. MAC de la computadora	90-A2-DA-0F-62-61
Dirección IP	192.168.0.7	192.168.0.2
Máscara de subred	255.255.255.0	255.255.255.0

Tabla 3.3. Direcciones IP para TCP/IP v4.

Para configurar el nodo AN como cliente se programa la TAD con la biblioteca Ethernet.h con la que se pueden definir la dirección MAC del cliente, la dirección IP del servidor y la dirección IP del cliente, es decir, la IP que se le asigna a Arduino (ver la Figura 3.23). Esta conexión se mantiene constante siempre que el nodo AN se encuentre conectado con la computadora, ya que la conexión se realiza en la función *setup*() de Arduino que se ejecuta una vez y se deja abierta.

Por su parte en la función *loop*() de Arduino se hace la verificación de que el cliente Arduino esté conectado al servidor por el puerto 8080, como se hace en la función *writeData*() la cual se ejecuta aquí. Para crear el cliente *Ethernet* se utiliza la variable *client* del tipo *EthernetClient*, la cual se inicia con la dirección MAC (mac) y la dirección IP de Arduino (ip).

Siguiendo el orden de lo que aparece en el código, primero se conecta con el módulo XBee receptor del nodo AN lo cual significa que ya puede recibir datos de los nodos BSN, después inicia la conexión Ethernet y finalmente si la variable client se ha conectado se muestra en la terminal serial el mensaje "conectado". El mensaje "PHP ejecutado" que se repite varias veces, indica que ya se está ejecutando la función *loop*() de Arduino (Figura 3.24).

```
#include <Ethernet.h>
#include <XBee.h>
#define MAX_FRAME_DATA_SIZE 110

EthernetClient client; //Variables de entrada
XBee xbee = XBee(); //Variables de XBee
Rx64Response rx64;

//Contantes de conexion a la red TCP/IP
byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0x62, 0x61}; //Dir. MAC de Arduino
byte ip[]  = {192, 168, 0, 2}; //IP de Arduino
byte server[] = {192, 168, 0, 7}; //IP de la computadora
uint8_t payLoad[] = {0, 0, 0}; //Arreglo para almacenar datos

void setup() {
    Serial.begin(9600);
    xbee.begin(Serial);
    rx64 = Rx64Response();
    Serial.println("1. Activa la red ZigBee");
    Ethernet.begin(mac, ip);
    Serial.println("2. Conectando Cliente Arduino - Servidor IP ...");
    while(client.connect(server, 8080)){
        Serial.println("3. Conectado");
    }
    delay(300);
}
```

Figura 3.23. Código para programar el nodo AN como cliente.

Figura 3.24. Terminal serial del cliente Arduino mostrando las etapas de conexión y validando que el PHP se ha ejecutado.

4. Pruebas de la RIS y de SARIS

El sistema completo tiene un componente de hardware que es la RIS y un elemento de *software* al que se le denomina SARIS, las pruebas se realizan en el sistema conforme se avanza en la implantación del mismo. Para la implantación de la RIS se hicieron pruebas con los sensores, los módulos de comunicación inalámbrica y las TAD, finalmente se hicieron pruebas con el protocolo CSMA/CA ranurado en modo *sleep*.

En cuanto al programa SARIS el diseño y los requisitos del sistema tienen definido dos tipos de usuarios los cuales tienen delimitados las clases y las tablas a las que puede acceder. Otro punto en el programa SARIS es que se conecta con el nodo AN para pedir los datos que recibe de un nodo BSN en el instante que se le socita el dato.

4.1. Banco de pruebas a la RIS implantada

La RIS se elabora de modo gradual y en cada etapa se realizan pruebas de las cuales se obtienen resultados que permiten seguir construyendo hasta tener la red inalámbrica que se define. Lo primero fue la comunicación punto a punto y la configuración de la red inalámbrica con topología estrella, después se hicieron las pruebas en el nodo AN con respecto a cómo lograr enviar a la base de datos las muestras que llegaban de los nodos BSN, fue necesario probar esto primero antes de implantar el programa SARIS, con esto se definen las siguientes pruebas a la RIS:

i. Prueba del sensor ISTD – 027 y el dispositivo DHT11.
ii. Lograr la comunicación punto a punto con los módulos XBee Serie Uno PRO y Arduino Uno.
iii. Agregar tres nodos transmisores BSN y ver los datos en el puerto serial COM del nodo AN.
iv. Activar el protocolo CSMA/CA ranurado para evitar colisiones y verificar que los datos lleguen íntegros.
v. Ejecutar códigos PHP que inserten datos en la base de datos donde se van a almacenar las muestras.
vi. Configurar el nodo AN como cliente y que se solicite ejecutar el *script* en PHP para que inserte unas muestras predefinidas.
vii. Solicitar que tome muestra por el puerto serial especificando el nodo por un valor entero recibido por la terminal serial.
viii. Agregar la comunicación serial con la interfaz de Java y el nodo AN para enviar los valores enteros que especifican los nodos que se envían.

En cuanto a los sensores, el único sensor que requirió pruebas es el sensor ISTD – 027 debido a que no existe una biblioteca en la que simplemente se obtenga el porcentaje de humedad como es el caso del dispositivo DHT11. Una vez que se establecieron los extremos de totalmente seco, es decir, el relativo 0% de humedad y totalmente sumergido en agua el relativo 100% de humedad, se modela la respuesta del sensor con la Ecuación 2.2 que expresa el voltaje de salida del sensor con respecto a la cantidad de mililitros de agua que entra.

4.1.1. Comunicación punto a punto

Ya que se tiene listo la lectura de los sensores en las TAD se conecta el módulo XBee a Arduino para que los datos de los sensores sean enviados por radiofrecuencia y recibidos por otro módulo XBee en el cual se espera ver los datos en la terminal serial usando el programa X-CTU. En la Figura 4.1 se observa que se reciben los datos del sensor ISTD – 027 junto con una cadena de caracteres que indica el número del transmisor y lo que se está leyendo.

En principio se utilizaron cadenas de caracteres que indicaban los datos que se estaban recibiendo, como en un inicio la perspectiva de lo que se implementaba era un solo transmisor y un receptor se utilizó TX más un número para indicar que se trata del transmisor y el número consecutivo que le correspondía de acuerdo a la configuración.

Una vez lograda la transmisión y recepción, era importante que el XBee receptor entregue las muestras a la TAD, pues de otro modo no se podría dar la información a la base de datos. En este punto la biblioteca XBee.h resulta útil porque se define un arreglo *payLoad* que lleva los datos que se obtuvieron del sensor se pueden conectar varios sensores a la TAD y los envían por un pin de entrada y salida del módulo de comunicación.

Considerando que el módulo XBee utiliza un pin y que tiene 8 pines que se pueden usar de entrada y salida digital para enviar y recibir datos, resulta una ventaja utilizar una TAD pues incrementa en número de sensores que pueden utilizarse para otras aplicaciones en las que se requieran utilizar más sensores como monitoreo de cultivos. Esto significa que se puede utilizar únicamente los módulos XBee en los nodos BSN pero usar la TAD permite extender la cantidad de sensores que se utilicen en la red e inclusive para enviar datos a los nodos BSN desde la aplicación SARIS para obtener una respuesta del mismo.

Usando el código del receptor en la TAD para leer los datos del módulo XBee, es suficiente utilizar dicha configuración, pero hay una diferencia importante en la respuesta de la terminal serial cuando el módulo XBee se configura para que reciba y transmita datos de forma simultánea, esta es que aparece un conjunto de caracteres fijos además de los datos recibidos. Esto sucede si se desea realizar una comunicación bidireccional o cuando se desea reenviar a un tercer nodo, de modo que el primer nodo que recibe datos es en realidad un repetidor en la red que permite ampliar el alcance.

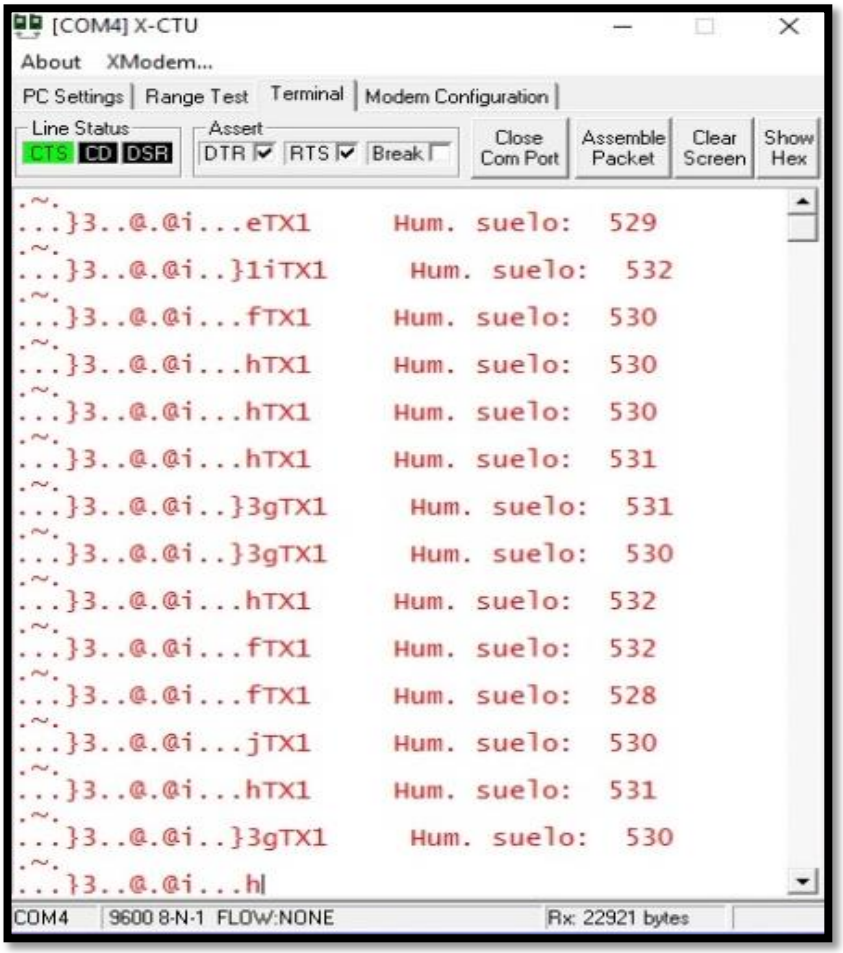

Figura 4.1. Datos recibidos en el XBee receptor por uno nodo transmisor.

4.1.2. Implantación de la topología estrella

En este punto se repite la configuración para agregar más nodos transmisores y se prueba en la terminal serial de Arduino. Con dos nodos transmisores el receptor muestra los datos de que recibe de ambos nodos, pero al incrementar los nodos transmisores a tres sólo se observa lo que uno de los transmisores BSN envía.

Esto sucede cuando se encienden los tres nodos de forma simultánea, pero cuando se enciende uno y un tiempo aleatorio después se enciende otro, la terminal serial muestra distintos nodos, sin embargo, los datos que se leen no corresponden al nodo que los envía, si no que más bien la información se cruza o colisiona.

Para resolver el problema de colisión o de que un solo nodo acapare el canal de transmisión se activa el estándar de acceso al medio CSMA/CA ranurado en modo sleep, así se puede obtener un resultado congruente entre los valores del nodo BSN y lo que se ve en la terminal serial del nodo receptor AN. A pesar de ello, se decide mostrar el arreglo *payLoad* ya que los caracteres que se envían hacen que los datos que se muestren se vean confusos y en realidad no son útiles para el sistema SARIS.

4.1.2.1. Activar el algoritmo CSMA/CA ranurado para evitar colisiones

Una de las pruebas que se hicieron fue retrasar el inicio de la función loop en las TAD de los nodos BSN para desfasar en tiempo el inicio de transmisión de cada uno de los nodos BSN, lo cual no varía el resultado en el receptor AN. Como el tiempo de desfase es en el orden de milisegundos se define que basta con un desfasamiento de 50 [ms] entre los XBee y que el periodo de operación sea de un segundo.

Así que los módulos XBee están activos transmitiendo con una diferencia de 50 [ms], lo cual funciona para que en el nodo AN sea posible tomar muestras correctas de los datos que transmite cada nodo BSN. También al funcionar en modo sleep los XBee consumen menos energía, sin embargo, estos se encuentran conectados con los Arduino que también consumen energía por lo que no varía significativamente la duración de la batería de 12 volts.

Al utilizar CSMA/CA ranurado en modo sleep hay breves periodos de tiempo en los que no se recibe ningún dato lo cual era de esperarse por el tiempo en que cada nodo está inactivo, aunque es diferente para cada uno, la repetición cíclica hace que en algún momento todos los nodos coincidan en estar inactivos.

Hasta a este punto la RIS funciona de acuerdo a las especificaciones, sin embargo, el nodo AN debe de ser capaz de funcionar con el programa en Java llamado SARIS, aunque se tiene el diseño de SARIS el punto importante es que este se comunique con la TAD para guardar muestras en una base de datos, esto define las siguientes pruebas y decisiones.

Si se desea tomar datos de la RIS de forma aleatoria, existe la probabilidad de que el usuario solicite una muestra de los datos recibidos y lo haga justo en el momento en el que todos los nodos están inactivos, lo cual no sucede la mayoría de las veces pero llega suceder como se pudo ver en las pruebas.

Esto representa un problema en un sistema donde el usuario puede tomar muestras en cualquier hora del día y elegir un nodo de forma aleatoria para leer los datos, para lo que se implementa el código que pueda hacer esto enviando un valor entero en la terminal serial, lo cual funciona si se utiliza sólo la interfaz serial de Arduino, pero no con una interfaz de prueba Java.

La razón por la que no funciona con la interfaz de Java, es porque utiliza la comunicación serial para comunicarse con Arduino al igual que el módulo XBee. Cuando se envía un valor entero es se hace de la muestra del nodo pero el XBee deja de comunicarse con Arduino y se tiene que apagar y volver a encender el nodo AN. Todo esto requiere que se utilice una versión de Arduino diferente que cuente con más seriales, por lo que se decide usar Arduino Mega que tiene cuatro.

En este punto una primera alternativa era crear una función dentro del programa en la TAD en el nodo AN, dicha función lo que hace es almacenar en un arreglo individual los datos recibidos donde cada uno es para un BSN diferente de la RIS y se actualizan constantemente, así que cuando se solicitan los datos de un nodo se utilizan estos arreglos en lugar del *payLoad*; también se podría usar un arreglo bidimensional.

Asimismo se hicieron las modificaciones con respecto al arreglo *payLoad* y el módulo XBee del nodo AN se configura sólo para recibir datos, lo cual permite que los valores obtenidos no tengan ningún dato adicional como se observa en la Figura 2.12. También se respeta la propuesta de que datos agregar al arreglo a transmitir y es el mismo orden en el que se trata a lo largo del sistema.

4.1.3. Pruebas con el servidor de base de datos

Aunque el servidor de base de datos es más bien parte del programa SARIS, es necesario ver que el nodo AN sea capaz de enviar los datos a la tabla dataRX, para ello, una vez instalado el servidor XAMPP se realizaron pruebas independientes. El primer punto fue probar un *script* que se conecta al servidor de base de datos (Figura 4.3), con el objeto de reutilizar el *script* que guarda los datos en el servidor, ambos *scripts* muestran un mensaje en el navegador que confirman la tarea que se acaba de realizar, el segundo *script* muestra la consulta utilizada para insertar datos y confirma que los datos se han guardado ver Figura 4.4.

Al probar el *script* hay que activar el servidor Apache y MySQL desde la interfaz de XAMPP y se utiliza un navegador de internet para ejecutar los archivos PHP, dentro del directorio C://xampp/htdocs/pruebas/arduino se guardan todos los archivos HTML y PHP, dentro de este están los archivos PHP que se ejecutan para conectarse a bancoArduinoPrueba e insertar registros en *dataRX*.

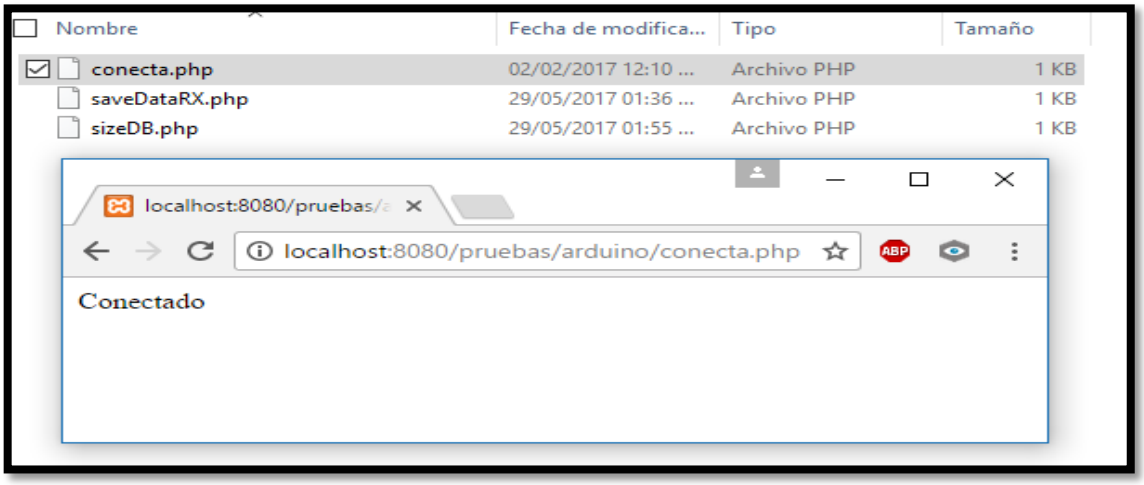

Figura 4.2. Prueba del *script* conecta.php que se encarga de conectarse a bancoArduinoPrueba.

Una vez que se logra la conexión con la base de datos bancoArduinoPrueba el *script* muestra el mensaje "Conectado", mientras que el *script saveDataRX.php* muestra toda la consulta *Insert* en MySQL cuando se ejecuta, los valores *node_id*, *humidity* y *temperature* son los campos creados en la tabla *dataRX* pues son los que se requieren insertar, ya que los primeros dos se generan de forma automática al realizar la consulta.

Figura 4.3. Respuesta del script *saveDataRX.php* al probar en el navegador de internet.

Para la realización de la interfaz SARIS se crea una pantalla en Java de prueba (Figura 4.5) y una tabla llamada *users* dentro de una base de datos de prueba llamada *dataNetwork* (Figura 4.6), para lo que se utiliza la biblioteca *mysql-connector-java-5.1.38-bin* que es la que hace conexión con Java y la base de datos MySQL. En estas pruebas las consultas se realizan desde las clases de Java esta, parte resuelve el conjunto de la interfaz probando lo más básico de la administración de usuarios.

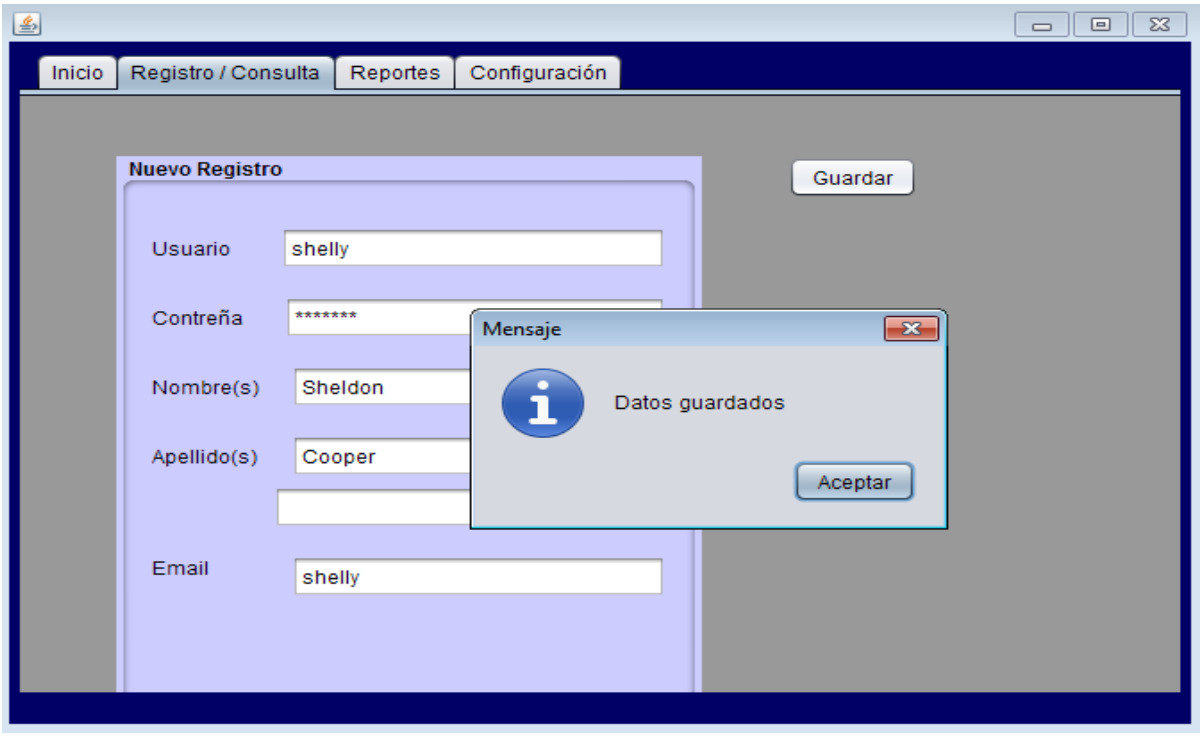

Figura 4.4. Pantalla de prueba de administración de usuarios guardando los datos a la tabla *users* de prueba.

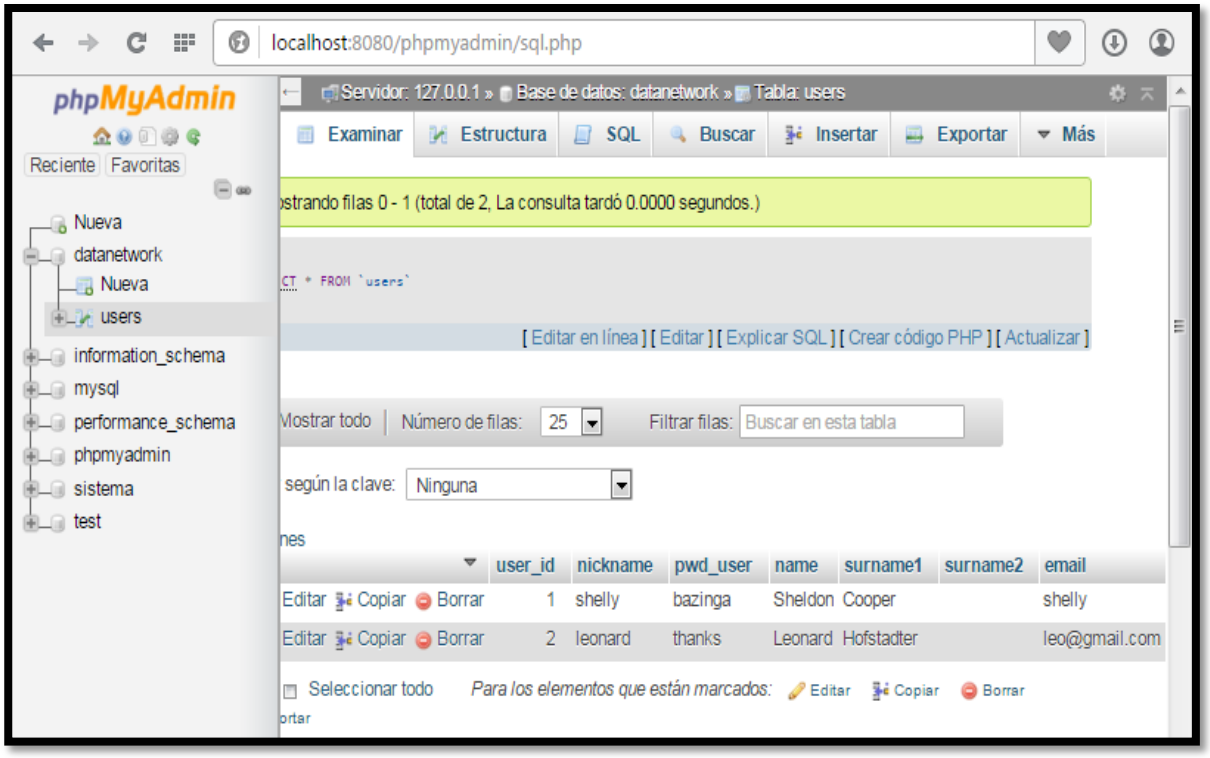

Figura 4.5. Pantalla del administrador de MySQL en el *localhost* que muestra el contenido de la tabla *users*.

4.1.4. Nodo AN como cliente Arduino

Se tiene comprobado que la RIS en topología estrella funciona correctamente, los scripts se pueden ejecutar en el servidor y la biblioteca de conexión de Java a MySQL funciona de forma deseada y permite ejecutar las consultas deseadas a las tablas de la base de datos. Lo siguiente es utilizar el shield de *Ethernet* para poder hacer pruebas con Arduino como cliente, para que este sea capaz de ejecutar el *script*.

Una vez configurada la red LAN con cable UTP en par trenzado se envía una trama con el comando ping y la dirección IP del nodo AN (Figura 4.8), de esta forma se comprueba que el nodo AN esté conectada a la red TCP/IP . Otra forma de ver la conexión de la red es mediante el puerto de serial del nodo AN en el que muestre el mensaje “Conectado” una vez que lo haga con la dirección IP asignada para este en la red (Figura 4.7).

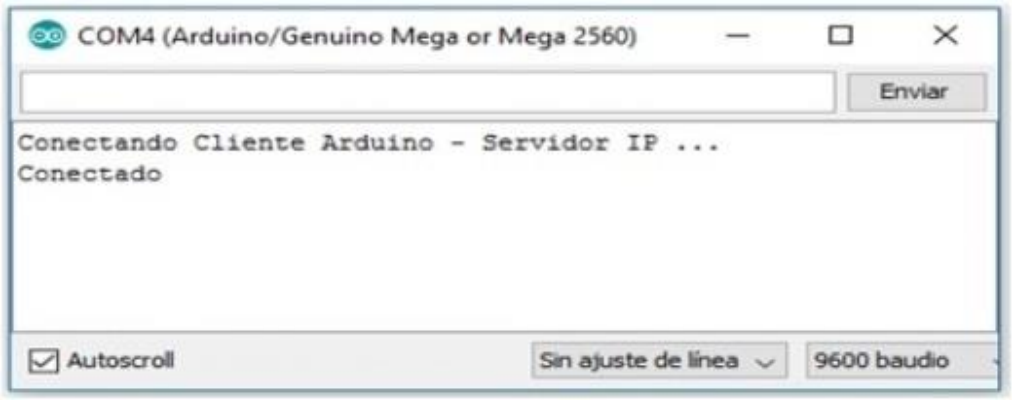

Figura 4.6. Terminal serial de la TAD del nodo AN mostrando que se ha conectado al servidor Apache como cliente.

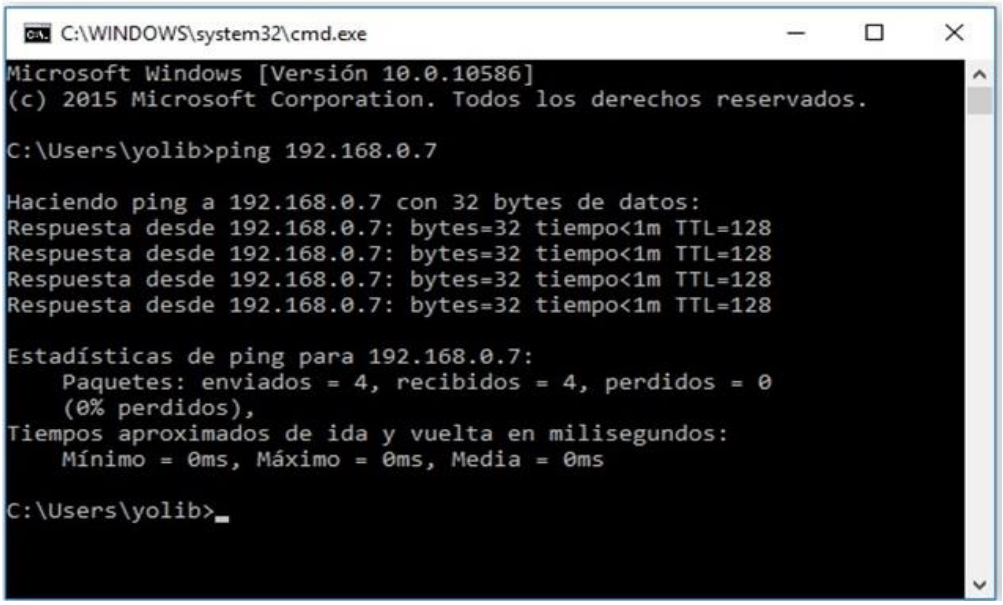

Figura 4.7. Comando TCP/IP ping para comprobar la conectividad con el nodo AN.

Ya conectado el nodo AN a la red TCP/IP, la TAD puede reaccionar de acuerdo a los valores enteros que reciban por la comunicación serial, esto permite que Arduino sea programado para que guarde las muestras cada vez que se reciban datos por la terminal serial, estos datos son valores enteros que hacen referencia a un nodo BSN, es decir, 1 es BSN1, 2 es BSN2, 3 es BSN3 y 4 es BSN4 (Figura 4.9); estos valores enteros son los que también se envían por el programa SARIS.

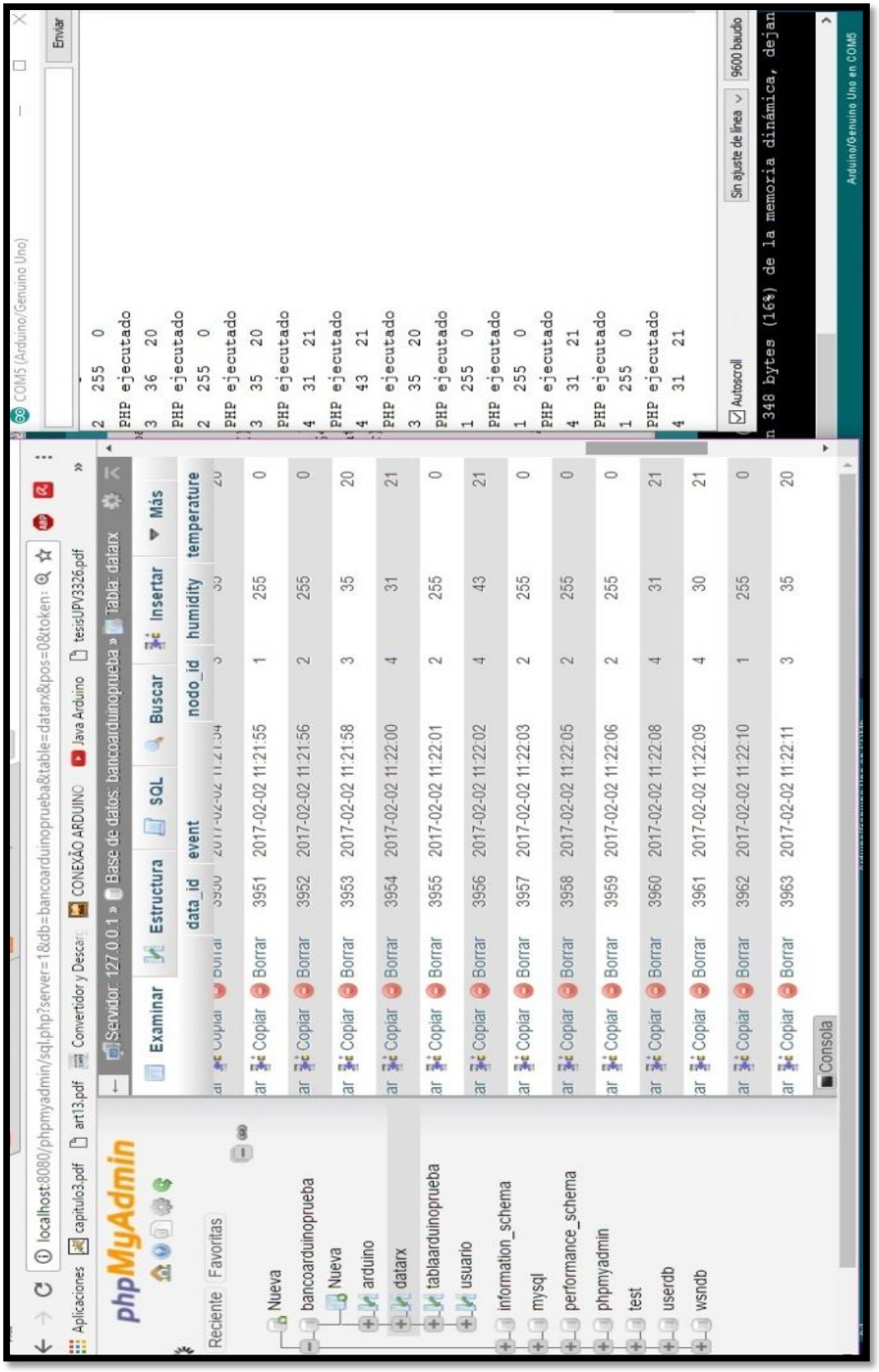

Figura 4.8. Datos insertados a la tabla dataRX desde el nodo AN.

Hasta este punto el sistema hace lo que se había planteado guardar muestras de la RIS en la base de datos, sin embargo, guarda datos incluso si son equivalentes a cero lo cual no tiene mucho sentido, ya que el nodo cero no está definido. Se establece que cuando la lectura del nodo_id sea cero no guarde ningún dato en la tabla y como sigue mostrando el valor cero se interpreta que el nodo no está disponible.

4.2. Banco de pruebas al programa de software SARIS

Al ser SARIS un programa de software se considera que las pruebas son necesarias para probar si cumplió con el objetivo de comunicarse con el nodo AN para tomar los datos que recibe de los nodos BSN, en segundo lugar la administración de usuarios de la red. En la comunicación con la RIS sólo falta probar que la interfaz sea capaz de enviar los valores enteros que requiere para identificar el nodo del que se le solicita una muestra.

En cuanto a la administración de usuarios sólo abarca crear, actualizar y borrar usuarios, sin embargo, el programa SARIS debe solicitar la autenticación y realizarla de acuerdo a las especificaciones establecidas en el diseño. Otro punto, es que lo que se entrega es una red y un programa de software del cual se creó el programa ejecutable para que sea instalado por un usuario por ello la realización de un manual de usuario.

Las pruebas de todo el sistema y sus objetivos generales se enumeran en:

i. **Instalación**. Probar que el sistema SARIS se pueda instalar y ejecutar en las distribuciones de Windows XP a Windows 10, sin embargo, el programa XAMPP también debe ser compatible ya que se instala de forma independiente.
ii. **Autenticación**. El proceso de autenticación debe dar acceso solo al usuario que tecle correctamente su contraseña, no debe validar campos vacíos.
iii. **Administración de usuarios**. Sólo un usuario administrador puede hacer esta tarea.
iv. **Comunicación con el nodo AN**. SARIS debe enviar en el puerto COM del nodo AN valores enteros para indicar al nodo que desea hacer lectura.

4.2.1. Ambiente de pruebas

En el Apéndice 1 de este texto se encuentra el manual de usuario, que indica el proceso de instalación del sistema SARIS en el que se incluye dar de alta la IP tanto de la computadora como del nodo AN, el manual también incluye el uso del sistema SARIS. En cuanto a la autenticación del sistema y el uso en sí se debe mantener activo tanto el servidor Apache como el servidor XAMPP.

- **Plataformas**
  Considerando que para instalar SARIS se tiene que instalar XAMPP, este programa de software pide como requisitos mínimos una memoria RAM de 256 MB y un espacio de instalación de 85 MB. Las computadoras en las que se prueban el sistema tienen alguna distribución de *Windows* y características diferentes:

i. *Windows* 10. Computadora con un sistema operativo Windows 10 de 64 bits, procesador *Core* i3, memoria RAM de 4 GB.
ii. *Windows* 7. Computadora con sistema operativo Windows 7 Pro de 64 bits, procesador *Core* i7, memoria RAM de 4 GB.
iii. *Windows* XP. Es una computadora de 32 bits con un procesador X, memoria RAM de 512 MB.

- **Configuraciones**
  Para probar primero se debe instalar el servidor XAMPP el cual tiene sus limitaciones, según su manual de instalación, el requisito mínimo es de una memoria RAM de 256 MB y un espacio de almacenamiento de 85 MB. El espacio de almacenamiento que puede ocupar SARIS es de 3 MB. La carpeta de instalación incluye archivos sql que tienen la base de datos y una carpeta con archivos PHP que se deben colocar en el directorio del servidor.

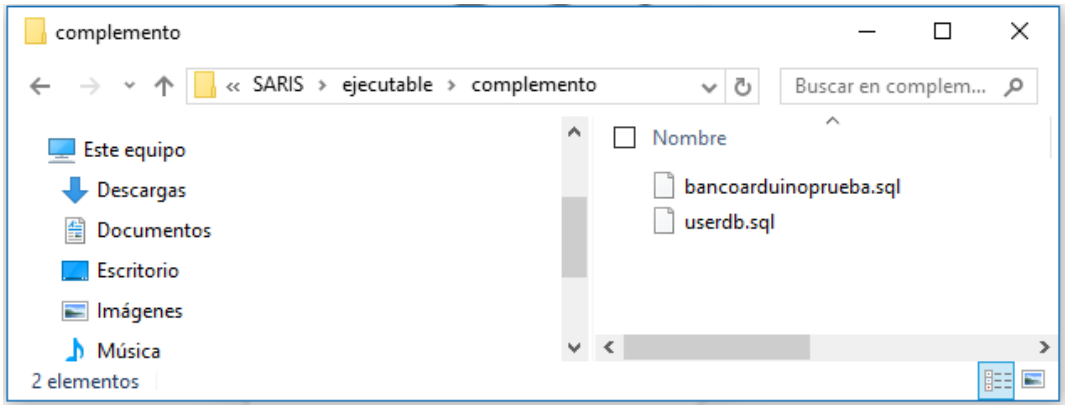

Figura 4.9. Directorio de SARIS donde se encuentran los archivos SQL que permiten importar las bases de datos que se utilizan.

4.2.2. Análisis y estrategias de pruebas

La prueba de SARIS se hace desde el punto de vista de la funcionalidad, es decir, que las pruebas van enfocadas a lo que ve el usuario como resultado. Por ello se establecen unos datos de entrada que prueban el sistema con respecto a su respuesta o los datos de salida que este muestra. De acuerdo a esto y las etapas de las pruebas, los objetivos de éstas son:

i. **Instalación**. Verificar que el sistema SARIS se pueda ejecutar en cualquier distribución de Windows de XP en adelante.
ii. **Autenticación**. Validar que el proceso de autenticación sea correcto y responda según lo establecido.
iii. **Administración de usuario**. Encontrar las incongruencias del sistema en la Administración de usuarios, para verificar cómo repercute en el proceso de autenticación.
iv. **Conexión con el nodo AN**. Validar que la conexión con el nodo AN se realiza y se puede tomar datos de este.

4.2.2.1. Objetivos con condiciones de prueba de la instalación de SARIS

La única condición de prueba es que se disponga de la carpeta de instalación completa. El programa SARIS se pudo instalar y usar una computadora con procesador *Core i7* y sistema operativo *Windows* 7 de 64 bits al igual que en una computadora con Windows 10 y procesador *Core i3*. En un sistema de 32 bits se puede instalar el servidor XAMPP siempre que se cumplan con los requisitos mínimos de *hardware* y el programa SARIS como tal es compatible con sistemas de 32 o 64 bits en las distribuciones de *Windows* desde XP en adelante.

Es importante que al instalar el programa XAMPP se verifique que el equipo no tenga ocupado el puerto 8080 como sucedió en uno de los equipos al que se le instaló SARIS, donde el servidor Apache no pudo configurarse con el puerto 8080. Otro punto es que SARIS requiere atención en la colocación de los archivos SQL y PHP que en ocasiones puede pasar desapercibido por el usuario por lo que se puede decir que no resulta del todo práctico para el mismo.

Por su parte el programa SARIS no tiene problemas de compatibilidad, sin embargo para su instalación y configuración se requiere de archivos SQL y de no instalarse el servidor XAMPP la configuración de SARIS no es posible, como se puede ver en la Figura 4.11 el instalador busca el archivo *mysql.exe* dentro del directorio del servidor XAMPP que debe existir, así, aunque se instale el programa de *software* SARIS, es inútil sin esta configuración.

Figura 4.10. Error al tratar de instalar el programa SARIS en un equipo Windows XP de 32 bits con memoria RAM de 256 MB.

4.2.2.2. Objetivos con condiciones de prueba de la Autenticación

Para probar la autenticación del programa SARIS se requiere que esté instalado y correctamente configurado, se ejecute y, la más importante condición de prueba, es que el servidor Apache y MySQL estén activos, lo cual se hace arrancando el servidor XAMPP y seleccionando el botón iniciar de ambos. Para verificar el proceso de autenticación se deben combinar los distintos posibles errores que puede tener el usuario al momento de capturar su usuario y contraseña; esos distintos escenarios se pueden ver en la Tabla 4.1.

El programa SARIS no sólo identifica que el usuario pertenezca a la base de datos *userDB*, sino que también identifica el tipo de usuario abriendo la pantalla que les corresponde al Lector y al Administrador. Sin embargo, hay otros escenarios, como el hecho de que la contraseña no sea correcta y el sistema pide al usuario que vuelva a capturar sus datos, para ello se requiere presionar el botón Limpiar pues los campos de texto no admiten que se teclee ninguna dentro de este campo que no sea texto y números, debido a las validaciones en el campo de texto.

El mensaje “Llenar los campos correctamente” significa que el usuario no ingresó el usuario y/o la contraseña correcta, esto incluye dejar campos de texto en blanco. Por otro lado el mensaje “Sólo teclear número, letras del abecedario en minúsculas y mayúsculas” significa que el usuario acaba de teclear algo que no es carácter (letra) o número y tiene que corregir el error.

Entrada		Salida	
Usuario	**Contraseña**	**Esperada**	**Obtenida**
Sin texto.	Sin texto.	No da acceso y muestra mensaje de error.	No da acceso y no hay ningún mensaje de error.
Usuario Administrador correcto.	Contraseña incorrecta.	Mostrar mensaje de Error: “Llenar correctamente los campos de texto”.	Mostrar mensaje de Error: “Llenar correctamente los campos de texto”
Usuario Administrador incorrecto.	Contraseña correcta.	Mostrar mensaje de error: “El usuario y/o la contraseña no existe”.	Mostrar mensaje de error: “Llenar correctamente los campos de texto”.
Usuario Administrador correcto.	Contraseña correcta.	Mostrar mensaje: “Bienvenido al sistema Administrador”.	Mostrar mensaje: “Bienvenido al sistema Administrador”.
Usuario Lector correcto.	Contraseña correcta.	Mostrar mensaje: “Bienvenido al sistema Lector”.	Mostrar mensaje: “Bienvenido al sistema Lector”.
Usuario Lector o Administrador correcto.	Sin texto.	Mostrar mensaje de error: “Llenar los campos correctamente”.	Mostrar mensaje de error: “Llenar los campos correctamente”.
Usuario con algún símbolo no válido.	Teclear una contraseña.	Mostrar mensaje de error: “Sólo teclear número, letras del abecedario en minúsculas y mayúsculas”.	Mostrar mensaje de error: “Sólo teclear número, letras del abecedario en minúsculas y mayúsculas”.
Usuario válido.	Contraseña con símbolo no válido.	Mostrar mensaje de error: “Sólo teclear número, letras del abecedario en minúsculas y mayúsculas”.	Mostrar mensaje de error: “Sólo teclear número, letras del abecedario en minúsculas y mayúsculas”.

Tabla 4.1. Escenario de pruebas para la autenticación.

4.2.2.3. Objetivos con condiciones de prueba de la administración de usuarios

La condición de prueba es que se haya iniciado el programa SARIS y el usuario se haya autenticado como el usuario Administrador, pues sólo este puede realizar las modificaciones a la base de datos que tiene la información de los usuarios que acceden al sistema incluyendo él mismo. Para realizar las pruebas se dividen en crear un nuevo usuario, actualizar los datos de usuario y borrar usuarios de la base de datos, quitar el acceso al sistema.

i. Crear un nuevo usuario

Crear un nuevo usuario implica llenar los campos de texto sus datos para insertarlo en la base de datos y darle acceso a SARIS. Debido a que no existe ningún tipo de validación en los campos de texto del formulario se pueden crear usuarios sin ningún tipo de restricción, por ejemplo, con todos los campos nulos, así que esto repercute en la autenticación donde una vez que se agregue un usuario con *nickname* y *password* nulo se puede ingresar al sistema con campos de textos nulos.

También se puede crear un segundo usuario del tipo administrador, sin embargo, en el proceso de autenticación sólo entra como Lector, ya que el sistema está codificado para que identifique al usuario por el identificador de usuario (*user_id*). Este caso se puede tomar como que el Administrador ingresó datos erróneos y este los puede corregir posteriormente realizando una actualización de los datos de usuario.

ii. Actualizar datos de usuario

A excepción del menú desplegable que está para definir el tipo de usuario, ningún campo del formulario tienen validaciones de ningún tipo, así que nuevamente se podrían utilizar datos no válidos para los campos, por ejemplo, ingresar números en los campos de nombre y apellido, ingresar en el campo de correo electrónico un texto sin el formato correspondiente, contraseñas nulas, etcétera (Figura 4.12). En cuanto al tipo de usuario, se hacen las pruebas que se muestra en la Tabla 4.2.

	Entrada	**Salida**	
		Esperada	**Obtenida**
Tipo de usuario	Lector	Datos actualizados	Datos actualizados
	Administrador	No se puede actualizar como usuario administrador.	No se puede modificar al usuario Administrador
Modificar	Hay un usuario seleccionado de la vista.	Envía datos al formulario.	Envía datos al formulario.
	No hay un usuario seleccionado.	Muestra mensaje de error: “No se seleccionó una fila”	Muestra mensaje de error: “No se seleccionó una fila”

Tabla 4.2. Escenarios de prueba para el menú desplegable “Tipo de usuario” y el botón “Modificar”.

La vista de usuarios es la que se muestra en la Figura 4.11 que contiene la lista de usuarios que se encuentra en la tabla *userList* mostrando sólo los datos de algunos de los campos de dicha tabla. Esta sirve para que el usuario Administrador tenga una relación de las personas que acceder a la red y desde este punto actualice o elimine a los usuarios Lector.

La pantalla asignada a la administración de usuarios denominada “Registro de usuarios” tiene un formulario para crear y actualizar usuarios; al crear un nuevo usuario se llenan los campos del formulario y se selecciona el botón “Guardar”, al actualizar los datos de un usuario ya existente se selecciona uno de la vista y se selecciona el botón “Modificar” que carga los datos al formulario para realizar la modificaciones, finalmente se selecciona el botón Actualizar y el programa emite una pantalla con el mensaje “Datos actualizados”.

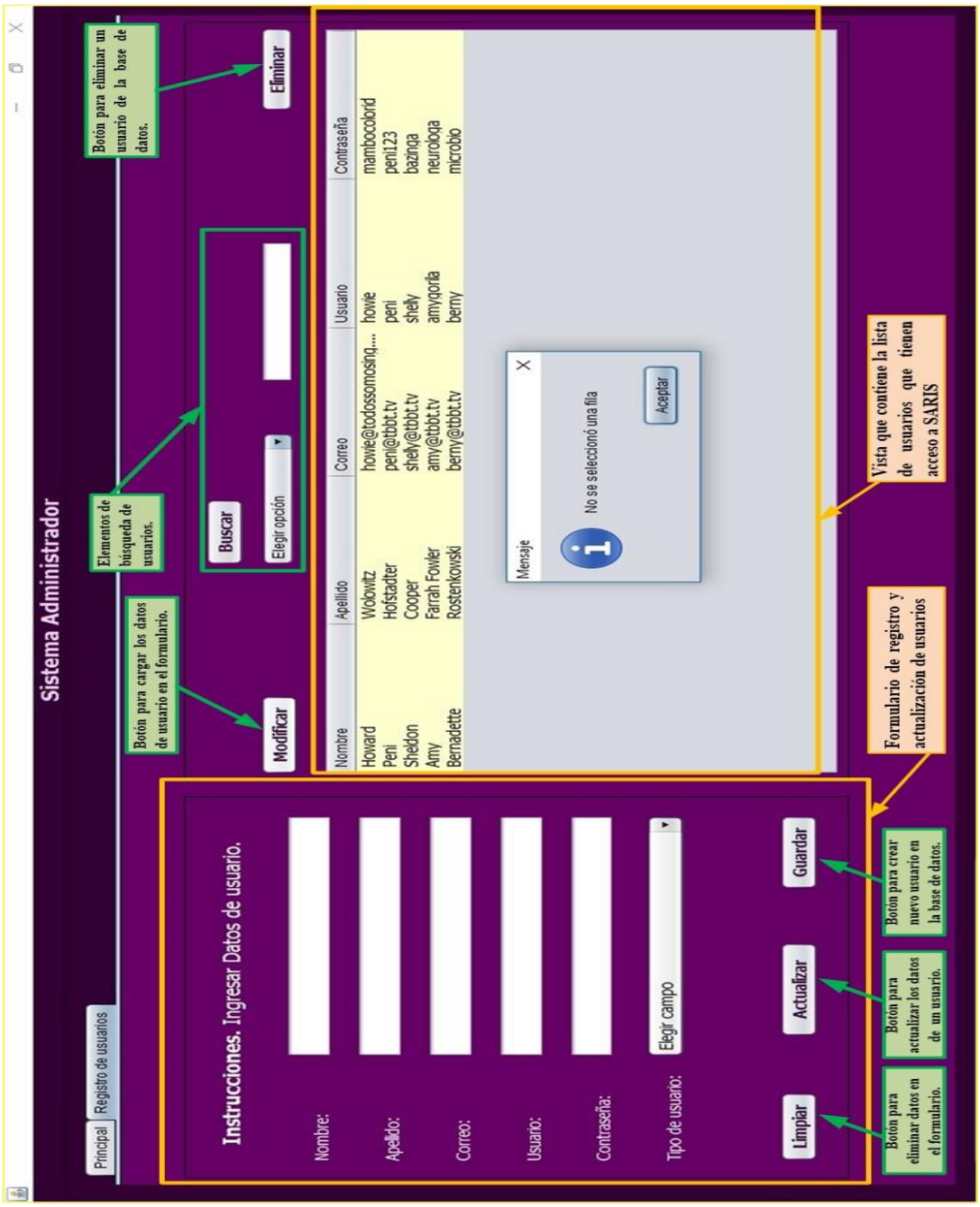

Figura 4.11. Pantalla “Registro de usuarios” mostrando el mensaje de error “No se seleccionó una fila” que indica que no se seleccionó un usuario de la vista.

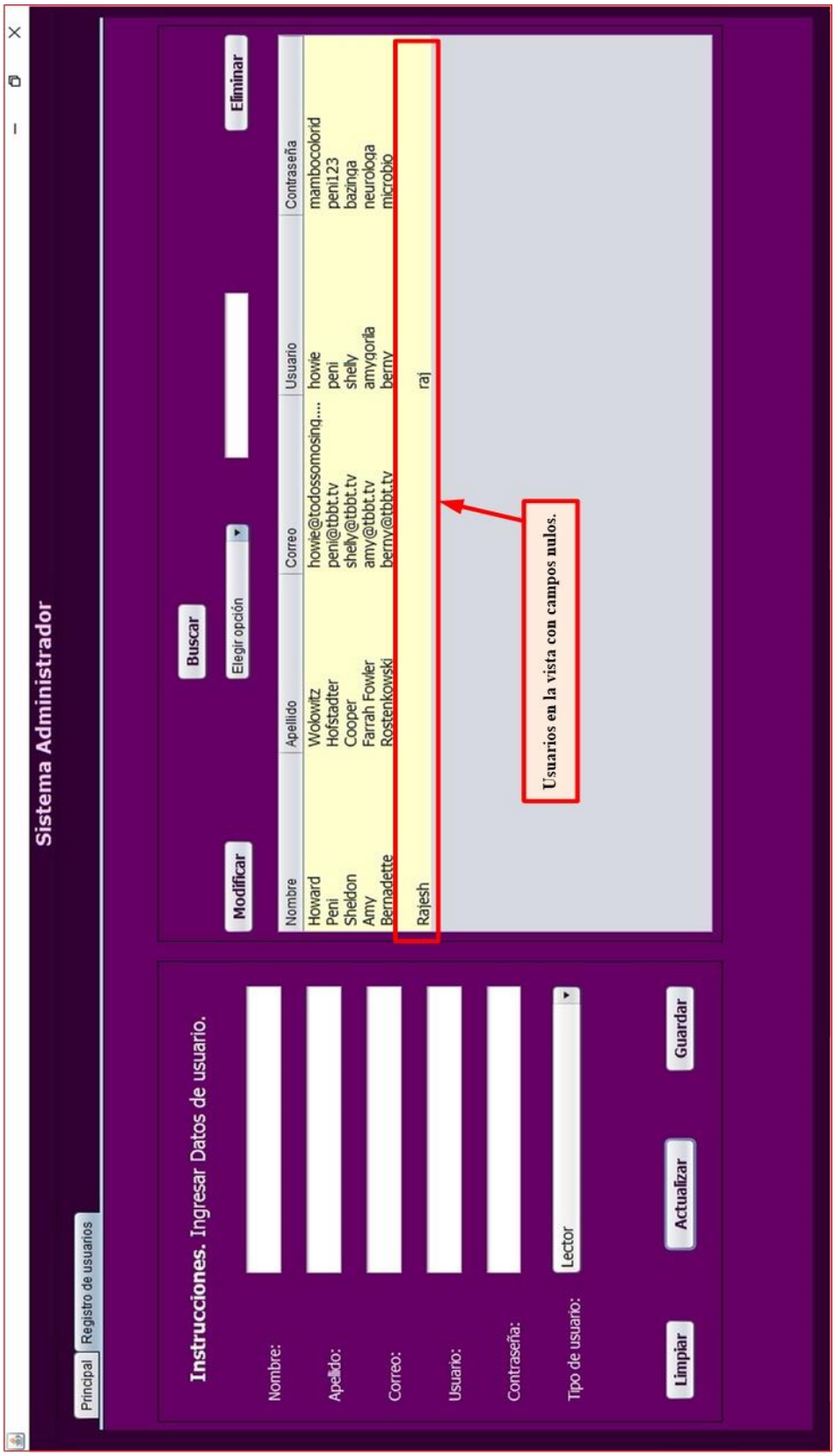

Figura 4.12. Interfaz SARIS con vista de usuarios donde dos de los usuarios tienen campos nulos.

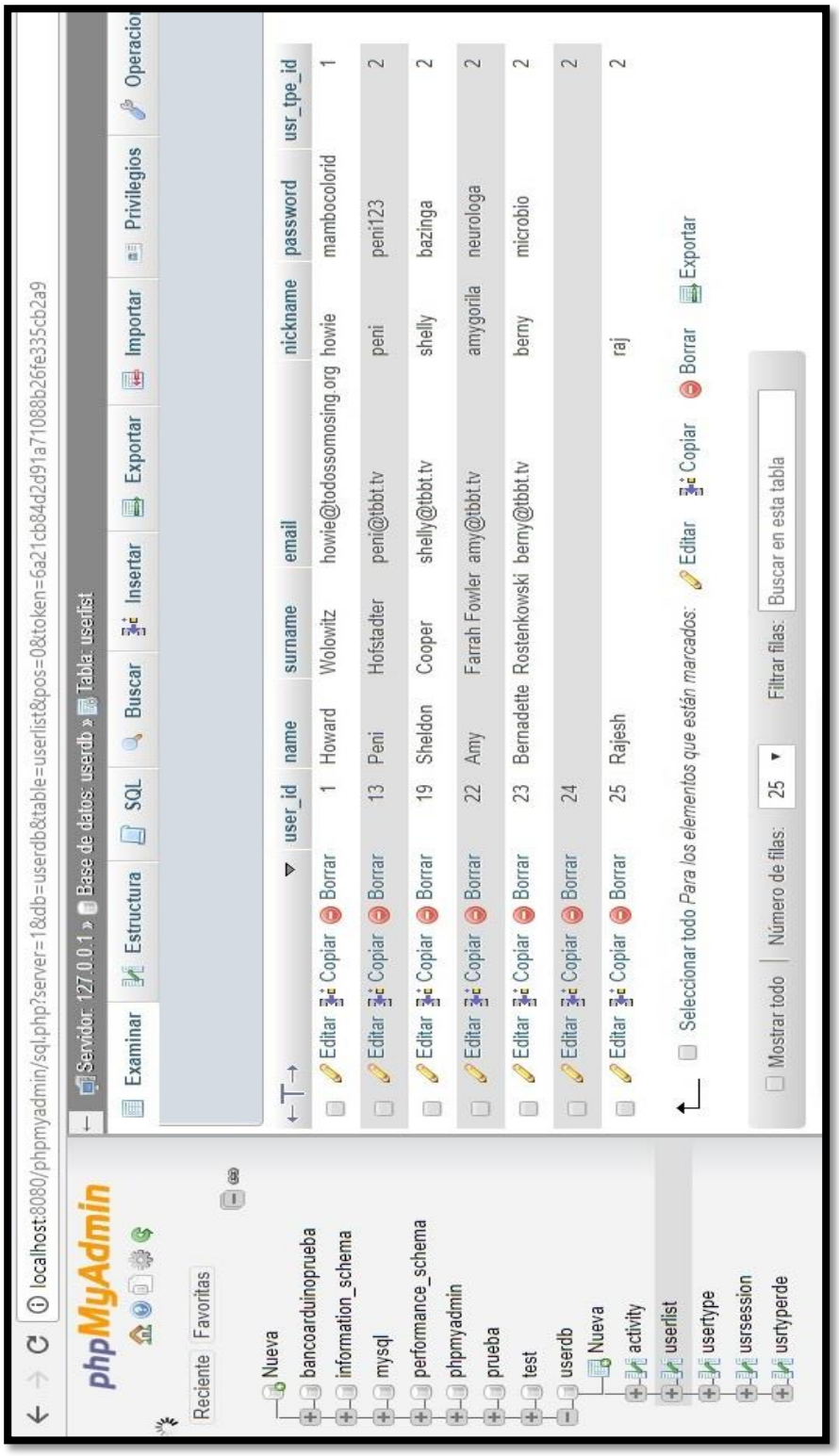

Figura 4.13. Base de datos userDB mostrando los campos de texto nulos que se ingresaron a la tabla userList.

En cuanto al usuario Administrador, el sistema no permite que se le realicen modificaciones desde el programa, al intentarlo, el programa emite una pantalla con el mensaje “No puede guardar otro usuario Administrador” cuando se intenta realizar una actualización a este usuario, después de seleccionar el botón “Aceptar”, aparece el mensaje “Datos Guardados”, sin embargo el sistema no realiza ninguna modificación.

Figura 4.14. Mensaje de error al intentar actualizar los datos del usuario Administrador.

iii. Eliminar un usuario

Cuando el usuario Administrador borra a algún usuario Lector de la vista en la pantalla "Registro de usuarios", el programa SARIS no muestra ningún mensaje de confirmación al borrar un usuario sólo muestra el mensaje "Datos eliminados" que significa que el usuario ya se borró y desaparece de la lista una vez que se selecciona el botón "Aceptar.

Desde la perspectiva del usuario el botón "Eliminar" del programa SARIS debe contener un mensaje de confirmación antes de borrar a un Lector pues éste se puede equivocar seleccionando a un usuario Lector equivocado. Sin embargo, este punto no está en la interfaz porque no fue considerado desde el diseño del programa al realizar los diagramas UML.

	Entrada	**Salida**	
		Esperada	**Obtenida**
Botón Eliminar	Hay un usuario seleccionado de la vista.	Mostrar un mensaje: "¿Está seguro que desea borrar al usuario (nickname de usuario)?". En caso de que el usuario confirme mostrar un mensaje de que se ha borrado el usuario.	Borra el usuario y muestra el mensaje "Datos eliminados". Finalmente lo elimina de la vista de usuarios.
	No hay un usuario seleccionado.	Mostrar mensaje de error: "Seleccione un usuario".	No hay respuesta del sistema.

Tabla 4.3. Escenarios de prueba para el botón Borrar del Registro de usuarios.

En caso de que el usuario Administrador se elimine así mismo por error, es posible ya que no hay ninguna validación de eliminar sólo a los usuarios Lector, éste ya no podrá autenticarse en el programa SARIS, la única forma de recuperar al usuario administrador es ingresándolo directamente en el servidor de base de datos.

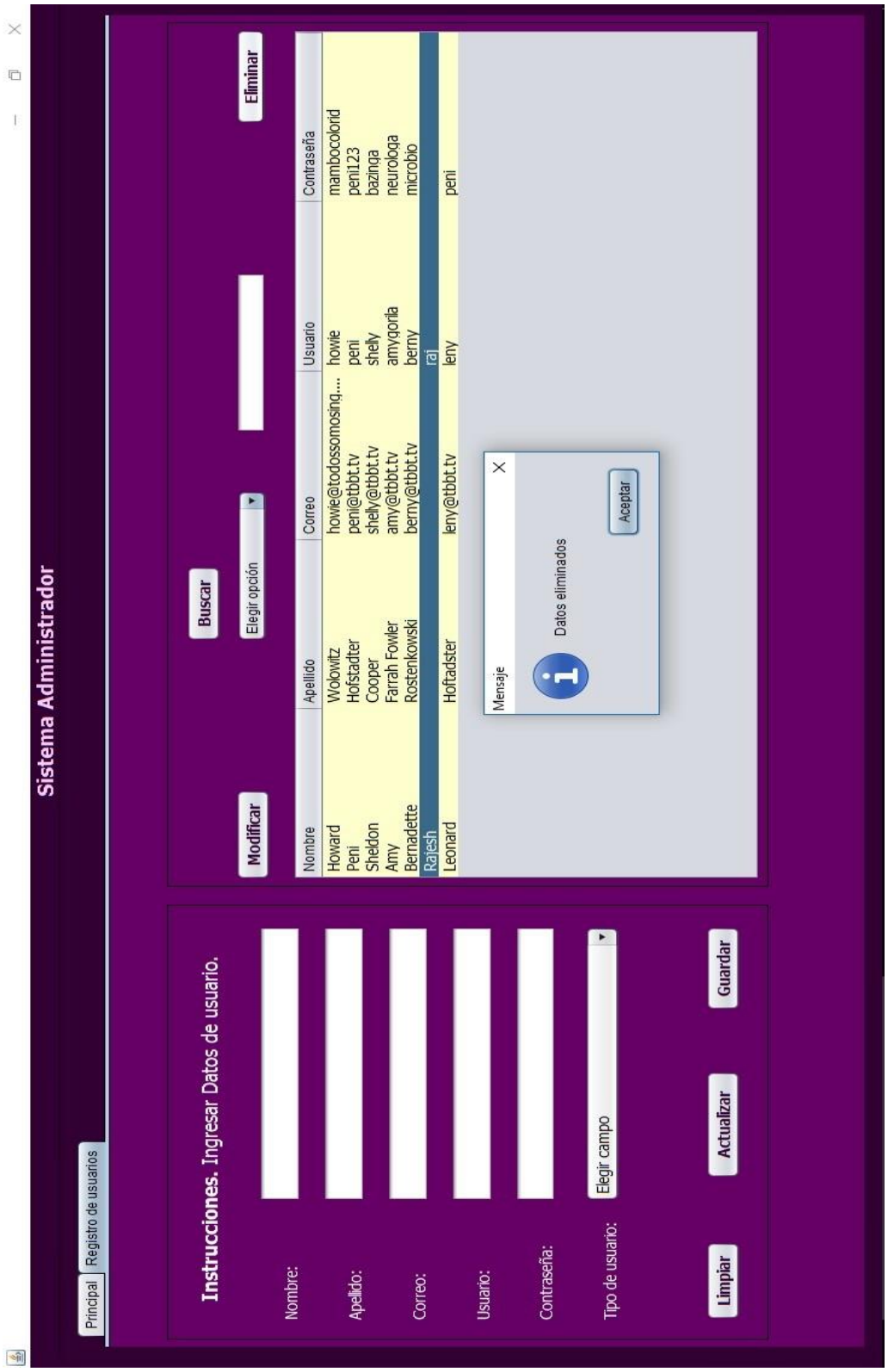

Figura 4.15. Eliminando un usuario de la vista de la interfaz.

4.2.2.4. Objetivos con condiciones de prueba de la toma de muestras en la RIS

El software SARIS tiene una pantalla para tomar muestras de los nodos BSN mediante los botones asignados para cada uno de éstos como se muestra en la Figura 4.16, como sólo está colocado un sensor por nodo para esta implantación en la interfaz se utiliza S1, S2, S3 y S4 para indicar el nodo al que se está conectando, se toma la letra "s" por la palabra "sensor", sin embargo, se trata de los nodos BSN.

El Arduino Mega se comunica tanto con la interfaz de Java como con el módulo XBee y el servidor Apache, por su parte la interfaz también conecta a la base de datos después de que se guardaron las muestras en la tabla dataRX y realiza una consulta en la base de datos para mostrarlos a la vista de datos que se encuentra en la pantalla de lectura de la RIS. Aun así, en la pantalla asignada para tomar muestras se encuentra lo siguiente:

i. **Seleccionar nodo cuando la RIS apenas se ha encendido**. Hay que dar tiempo a la TAD para que se encienda y ejecute la función *setup*(), una vez que entre a la función cíclica *loop*() se pueden tomar muestras de lo contrario el mensaje que se vería en la pantalla de muestras "Conectando al cliente Arduino – servidor IP", lo que significa que apenas se está conectando a la red TCP/IP y no se pueden guardar aún ninguna muestra.

ii. **Seleccionar nodo cuando este no ha sido encendido**. La respuesta del programa es mostrar un cero, ya que desde el inicio la TAD está programado a dar este resultado en caso de que no reciba datos del nodo y es lo que la interfaz SARIS recibe como dato.

iii. **Palabras enviadas por el puerto serial**. La respuesta del programa una vez que se selecciona un nodo es mostrar la palabra "Humedad=" más un numero entero y "Temperatura=" más otro entero. En ocasiones la palabra humedad no aparece completa en la pantalla.

Figura 4.16. Toma de muestras en los nodos de la RIS con la interfaz en Java denominada SARIS.

Conclusiones

En este trabajo se diseñó e implementó una Red Inalámbrica de Sensores con base al estándar IEEE 802.15.4 bajo la topología estrella y su interfaz de software para la lectura remota de los datos obtenidos con los sensores colocados en cada uno de los nodos transmisores BSN, al programa se le denominó Sistema de Acceso a la Red Inalámbrica de Sensores y también incluye una pantalla de administración de usuarios.

En la implantación de la Red Inalámbrica de Sensores el principal reto fue resolver la colisión de los datos que se reciben en el nodo AN, lo cual se resolvió usando el algoritmo CSMA/CA ranurado en modo *Sleep*. Esto no es expresado de forma explícita en el manual de usuario de XBee PRO Serie 1, sin embargo, como está basado en el estándar IEEE 802.15.4 en éste se puede extender la información de lo que se está configurando.

Visto desde el punto de vista de las capas de los estándares, la RIS implementada para este proyecto tiene la capa física y la capa MAC que se resuelve con los módulos XBee y el resto de los dispositivos electrónicos. En cuanto a las capas de ZigBee, la capa de Red fue resuelta con el código de la TAD que identifica el nodo BSN y los datos que transmiten, aunque no se estableció ningún protocolo de ruta de los nodos debido a la topología de la red.

En cuanto a la capa de Aplicación, todos los elementos que forman el programa SARIS constituyen la infraestructura de aplicación (AF) de la cuales sólo la clase *ReadNetwork* es un objeto de aplicación (APO) pues es la única que interactúa con la RIS, aunque de un modo pasivo al realizar sólo lecturas de los datos que llegan de los nodos BSN. El driver que permite comunicarse a la TAD con la interfaz es el objeto de dispositivo ZigBee (ZDO).

Las clases que se utilizan para la autenticación y la administración de usuarios no son APOs, porque ninguna se conecta directamente con la RIS, la única razón por lo que se definieron es porque en el sistema se delega la responsabilidad de gestionar los usuarios al Administrador, mientras que todos los usuarios que tengan acceso pueden tomar muestras.

Dentro de las líneas de trabajo futuro de la RIS y SARIS está agregar más sensores a los nodos e implementar una topología malla en la RIS, configuración vía remota de la RIS a través del programa SARIS. En el primer caso, al tener mayor número y variedad de sensores para tener mayor información de los espacios que se monitorean con la RIS. En este punto, es necesario trabajar con los sensores para caracterizarlos y establecer cómo se van a interpretar sus datos.

En cuanto la implantación de la RIS topología malla se debe utilizar un algoritmo de enrutamiento para que los nodos decidan el camino a seguir para llegar al nodo AN, y que se utilicen sólo microcontroladores en lugar de usar las TADs para reducir costos al aumentar el número de nodos en la red. El configurar vía remota los nodos por la interfaz de Java exige utilizar un tipo de transmisión en la red del tipo bidireccional o duplex, ya que para reprogramar los nodos vía remota en principio se tienen que enviar los **comandos API**[13] a los módulos XBee de los nodos BSN, además de recibir los datos que éstos transmiten al nodo AN.

[13] **Comando API**. Se utiliza en la configuración de los XBee cuando estos trabajan en modo API, lo que significa que permite tramas con cabecera que aseguran la entrega de datos por lo que una aplicación cliente puede interactuar con el módulo de comunicación.

Bibliografía

Baronti, P., Pillai, P., Chook, V. W., Chessa, S., Gotta, A., & Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, 1655–1695.

Bell, C. (2013). *Introduction to Sensor Network.* Nueva York: Technology in action.

Faludi, R. (2011). *Building wireless sensor network. A practical guide to the ZigBee Mesh Networking Protocol.* Estados Unidos de America: O'Reilly Media, Inc.

IEEE Computer Society Sponsored by the LAN/MAN Standards Commitee. (2015). IEEE Standards Association. *IEEE Standard for Low-Rate Wireless Networks*. Nueva York, Estados Unidos de America: Revisión de IEEE Std 802.15.4-2011.

Kimmel, P. (2007). *Manual de UML.* México: McGraw Hill.

Tanenbaum, A. S., & Wetherall, D. J. (2012). *Redes de computadoras.* México: Pearson Educación.

Camargo Olivare, J. L. (2009). *Biblioteca de Ingeniería.* Recuperado el 18 de Abril de 2017, de Universidad de Servilla: http://bibing.us.es/proyectos/abreproy/11761

Digi International, Inc. (31 de Mayo de 2007). *XBee / XBee-PRO OEM RF Modules.* Recuperado el Marzo de 2017, de http://users.ece.utexas.edu/~valvano/EE345L/Labs/Fall2011/XBeeManual.pdf

Digi International, Inc. (1 de Junio de 2007). *XBee Series 2 OEM RF Modules.* Recuperado el Marzo de 2017, de http://www.farnell.com/datasheets/27606.pdf

Márquez Avendaño, B. M., & Zulaica Rugarcía, J. M. (6 de Enero de 2017). *UDLAP Biblioteca.* Recuperado el 11 de Abril de 2017, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/

Lee, J. S., Su, Y. W., & Shen, C. C. (26 de Noviembre de 2016). *Old Dominion University.* Recuperado el 20 de Febrero de 2017, de Department of Computer Science: http://www.cs.odu.edu/~nadeem/classes/cs795-WNS-S13/papers/advance-005.pdf

Apéndice A

Manual de usuario SARIS

El presente manual forma parte de los resultados obtenidos en el diseño e implementación de una Red Inalámbrica de Sensores y su interfaz de usuario SARIS del Trabajo Recepcional para obtener el grado de Licenciada en Ingeniería en Sistemas Electrónicos y de Telecomunicaciones por la Universidad Autónoma de la Ciudad de México.

Introducción

SARIS es el Sistema de Acceso de Redes Inalámbricas de Sensores conformado por un nodo Estación Base (EB) el cuál se comunica con los nodos sensores S1, S2, S3 y S4 (Figura 1). En los nodos S1 y S2 está colocado el dispositivo ISTD – 027 que es un sensor de humedad de superficie; en los nodos S3 y S4 está colocado un dispositivo DHT11 que tiene el sensor de temperatura y humedad del aire.

El presente manual tiene como objetivo describir de manera detallada como conectar el nodo EB a la computadora, cómo instalar SARIS para tomar lectura de los sensores conectados a los nodos S y el cómo usarlo. SARIS guarda la lectura a una base de datos y muestra lo almacenado en un histórico que se muestra en la interfaz.

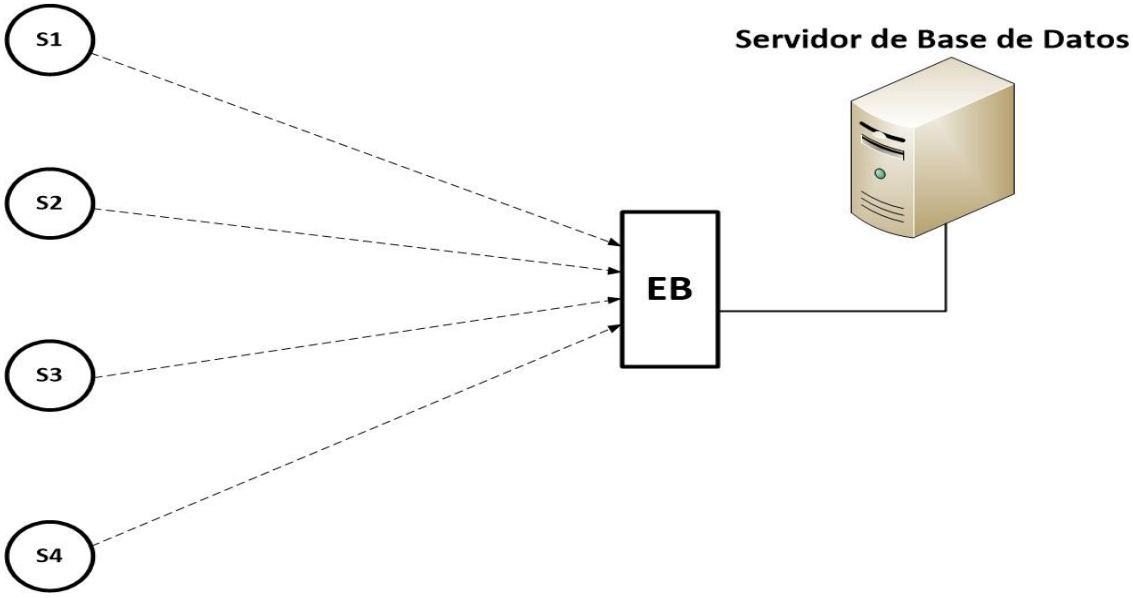

Figura 1. Esquema general de la RIS con topología estrella y un servidor conectado al nodo EB.

1. Instalar el servidor XAMPP – cliente SARIS

SARIS está diseñado para el sistema operativo Windows y es compatible con las distribuciones de Windows 7 a Windows 10 de 64 bits, sin embargo, se debe verificar que la computadora tenga instalada alguna versión de JRE (Java SE Runtime Environment), la cual la mayoría de los equipos Windows más recientes lo tienen. De no tenerlo se puede descargar en el siguiente enlace:

http://www.oracle.com/technetwork/java/javase/downloads/server-jre8-downloads-2133154.html

1.1. Instalar el servidor XAMPP

En primer lugar es necesario configurar la computadora que se conecta con el nodo Estación Base como servidor, para ello se debe utilizar XAMPP que es una aplicación que instala el servidor Apache, MySQL y PHP que son necesarias para SARIS. El ejecutable de instalación de XAMPP se encuentra en la carpeta de instalación **//SARIS/servidor** y el manual de instalación está disponible en:

http://www.mclibre.org/consultar/php/otros/in_php_instalacion.html

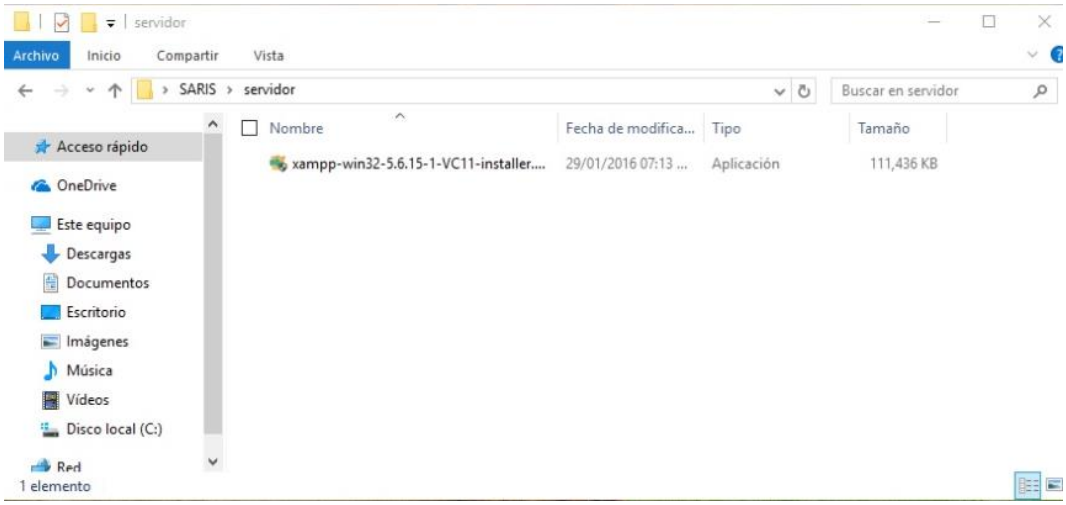

Figura 2. Archivo ejecutable de XAMPP

1.2. Cambiar puerto 80 a puerto 8080 en el servidor Apache

Configurar el puerto DNS del servidor de Apache debe ser 8080 y no 80 ya que este es el que utiliza cualquier navegador con el proveedor de internet y de no modificarse crea conflicto con el servidor Apache y el programa SARIS no podrá funcionar correctamente.

i. Abrir el archivo **httpd.conf** en el botón *Config* que se encuentra en la Fila de Apache.
ii. En las líneas que se señalan en la Figura 3 y la Figura 4 debe escribirse 8080.
iii. Presionar el botón *Config* que se encuentra en la última columna (Figura 3) justo arriba del botón *Netstat*.
iv. Seleccionar *Service* and Port Settings.
v. Escribir 8080 en el campo de texto que se encuentra en *Main Port* o puerto principal (Figura 7).
vi. Hecho lo anterior, iniciar el servidor Apache y MySQL presionando el botón *Start* que cambia la etiqueta a *Stop*, por lo cual, el botón también sirve para parar el servidor.

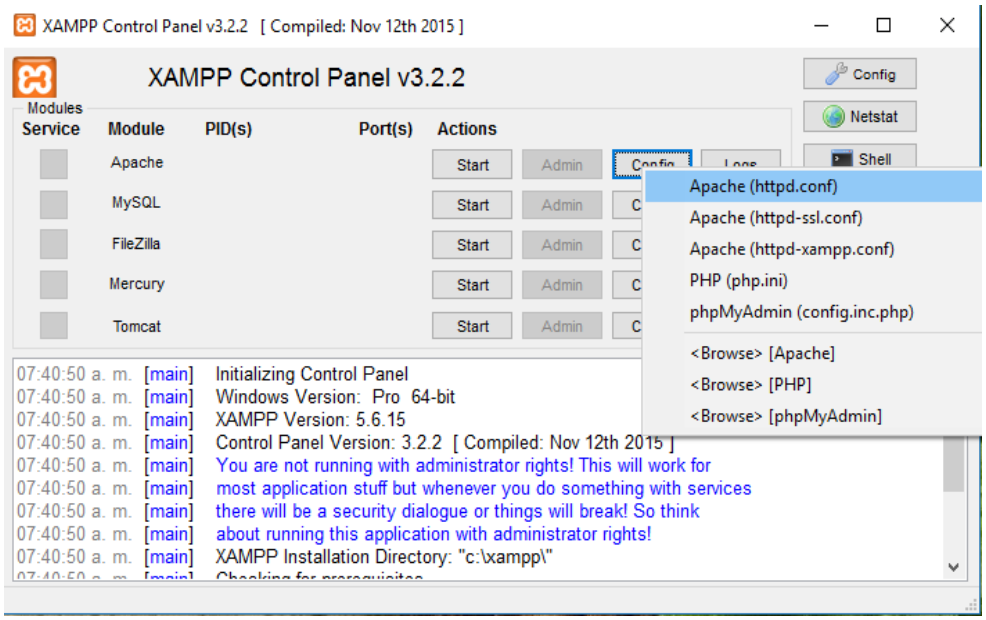

Figura 3. Abriendo el archivo *httpd.conf*.

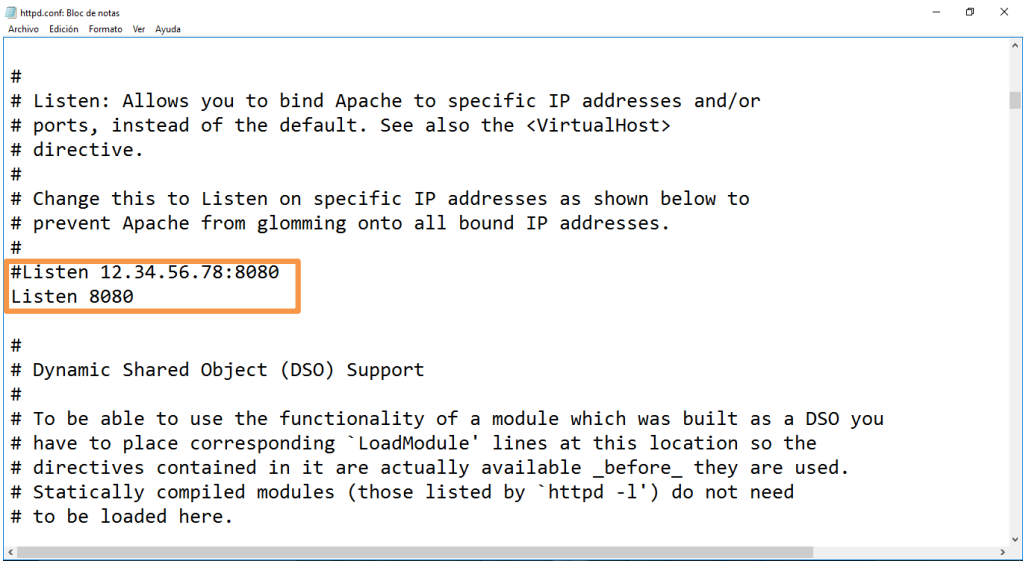

Figura 4. Modificación del archivo *httpd.conf*.

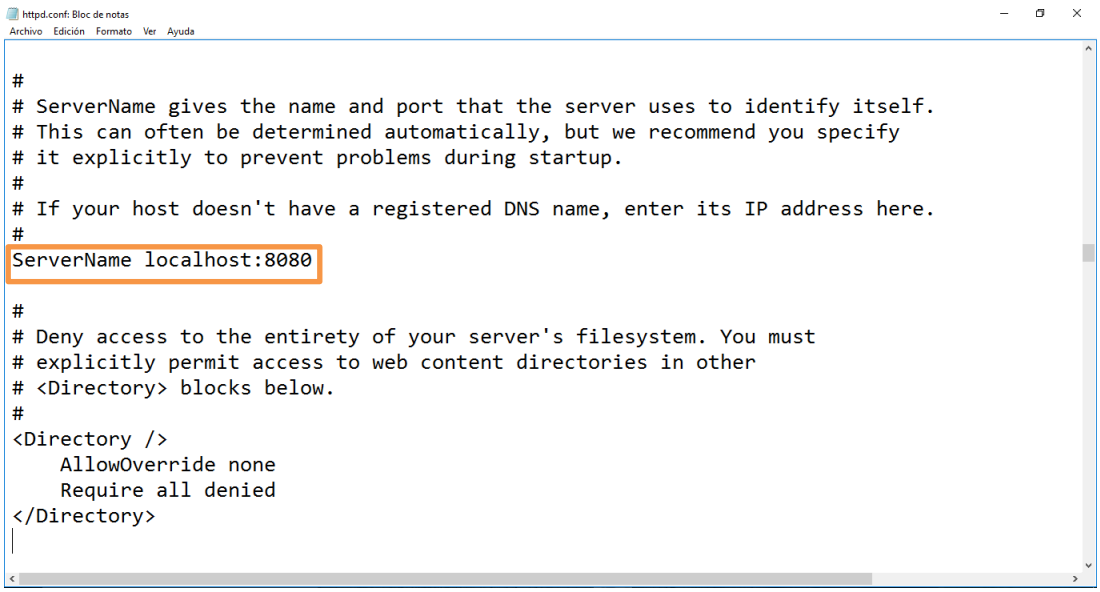

Figura 5. Segunda modificación del archivo *httpd.conf*.

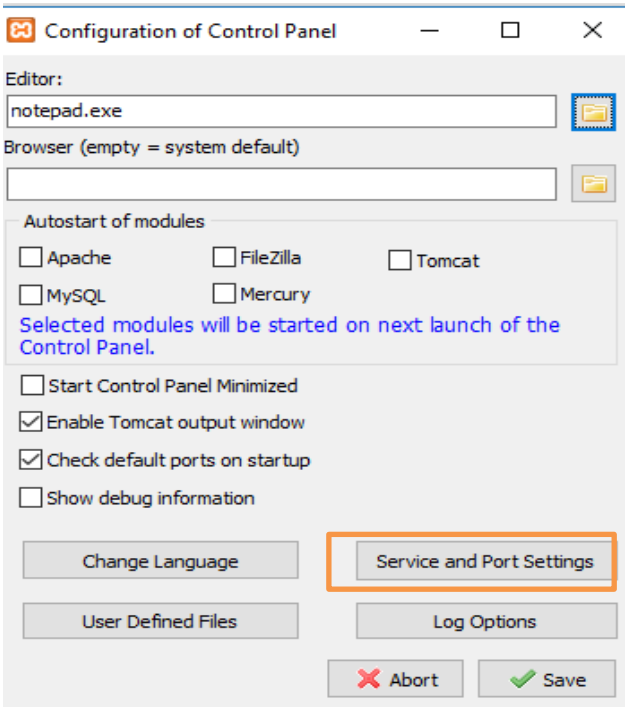

Figura 6. Seleccionar el botón *Service and Port Settings*.

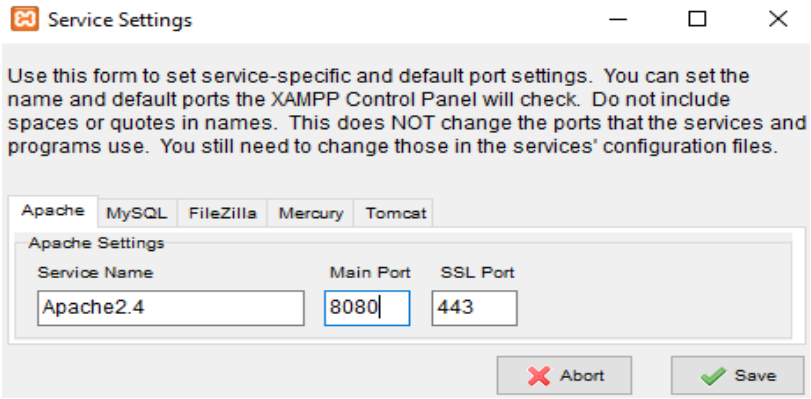

Figura 7. Modificando el puerto para HTTP y SSL.

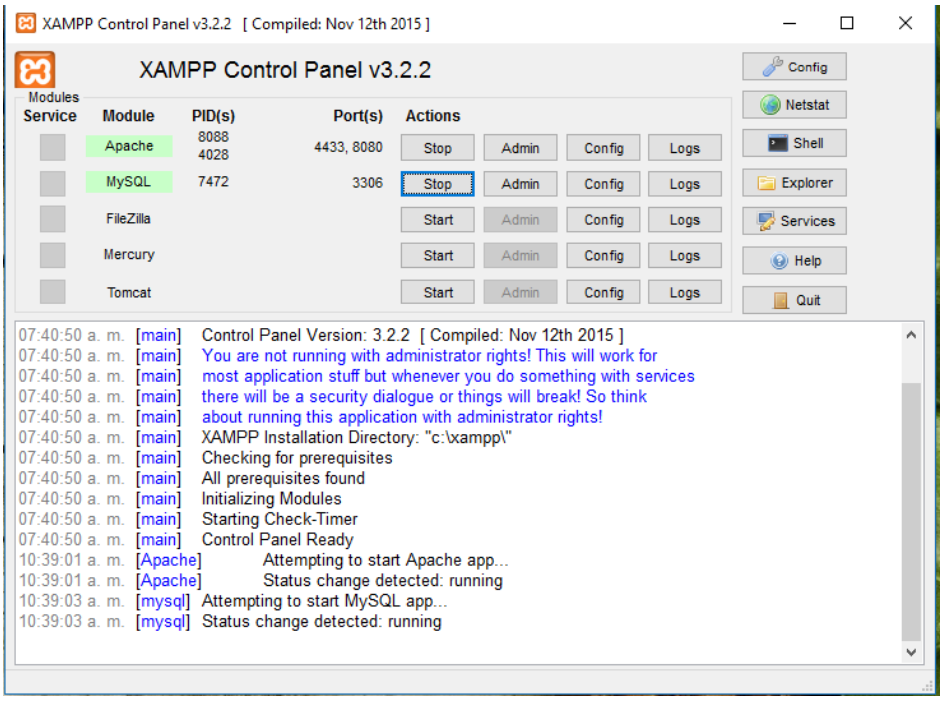

Figura 8. Pantalla de administración de XAMPP.

Al instalar el servidor XAMPP se obtiene un directorio de servidor **C:\xampp** en el cual se encuentran carpetas importantes en para la configuración de SARIS.

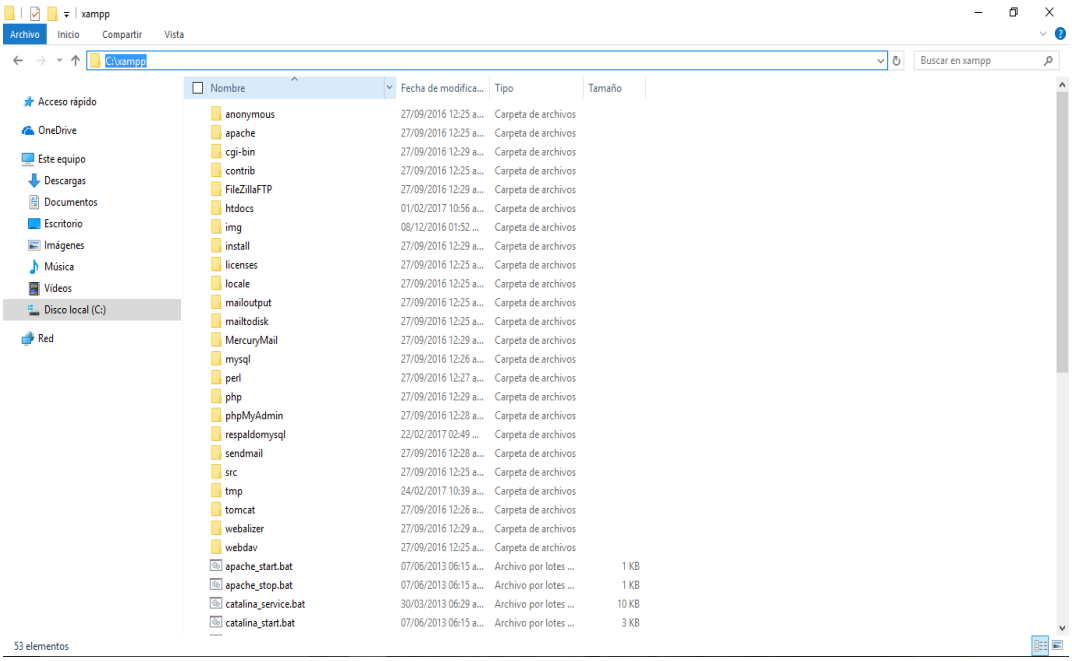

Figura 9. Directorio de la plataforma XAMPP instalada.

1.3. Crear usuario Arduino en el servidor de base de datos

Para poder ejecutar la aplicación SARIS es necesario crear un usuario en el servidor.

i. En el Panel de Control de XAMMP seleccionar el botón administrar que se encuentra en la fila de MySQL , al hacerlo se abre el navegador con la dirección:
http://localhost:8080/phpmyadmin/

ii. Seleccionar el enlace Cuenta Usuarios.

iii. Seleccionar el enlace Crear un Nuevo Usuario.

iv. Llenar los campos de texto en el cuadro Información de cuenta.

Nombre de usuario: Escriba un usuario. Nombre de host: Seleccionar local en el menú desplegable. Contraseña: Escriba una contraseña. Volver a escribir: Repita la contraseña.

v. Seleccionar todos los privilegios globales.

1.4. Instalar SARIS

La carpeta de SARIS tiene todos los archivos necesarios para su instalación de los cuales no se debe tocar la carpeta JAR que se muestra en la Figura 10.

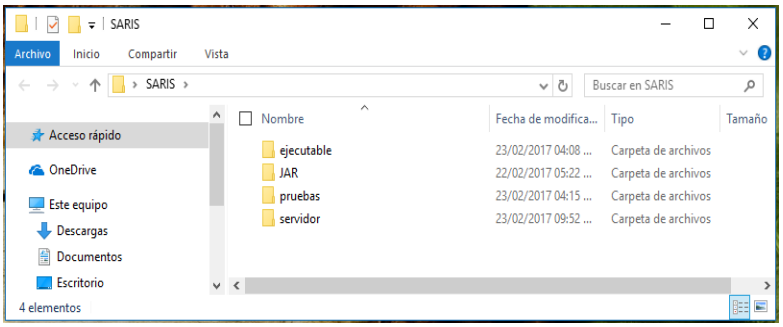

Figura 10. Carpeta de instalación de SARIS.

i. Copiar los archivos SQL **bancoarduinoprueba.sql** y **userDB.sql** al directorio XAMPP que se indica.

De: **\SARIS\ejecutable\complemento**
A: **C:\xampp\mysql\bin**

ii. Copiar toda la carpeta pruebas del directorio **C:\xampp\htdocs**.

iii. Ejecutar como administrador el archivo SARISetup.exe que se encuentra en el directorio **\SARIS\ejecutable\Output**.

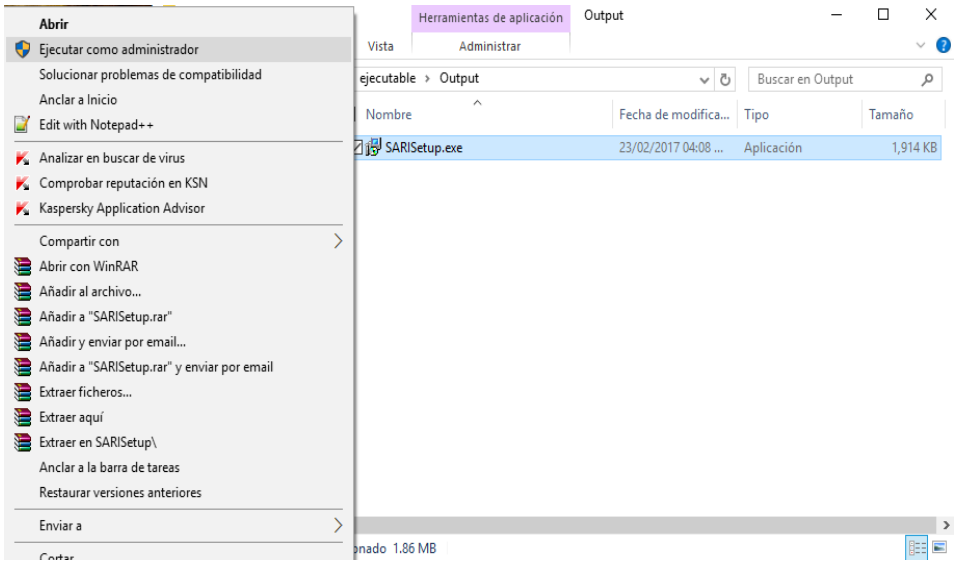

Figura 11. Archivo ejecutable de SARIS.

iv. La primera pantalla de instalación indica el directorio en donde se va a instalar SARIS, dar *click* en siguiente.

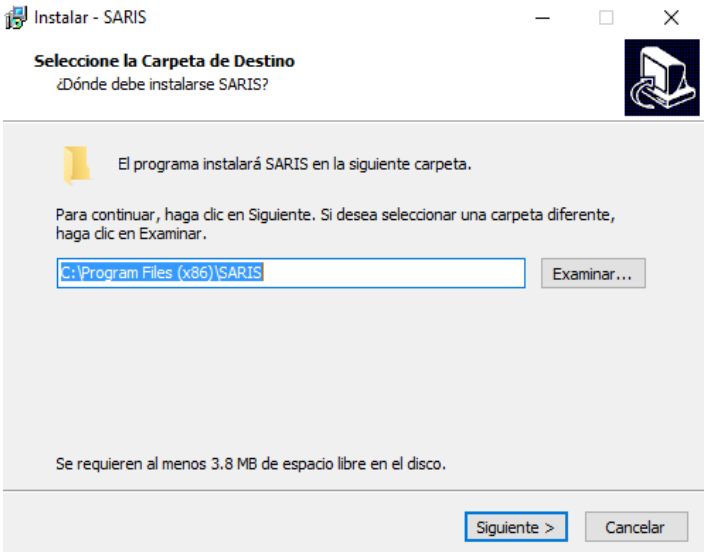

Figura 12. Primera pantalla de instalación.

v. La siguiente pantalla de instalación se debe indicar la carpeta del Menú de Inicio del sistema, por defecto es SARIS.

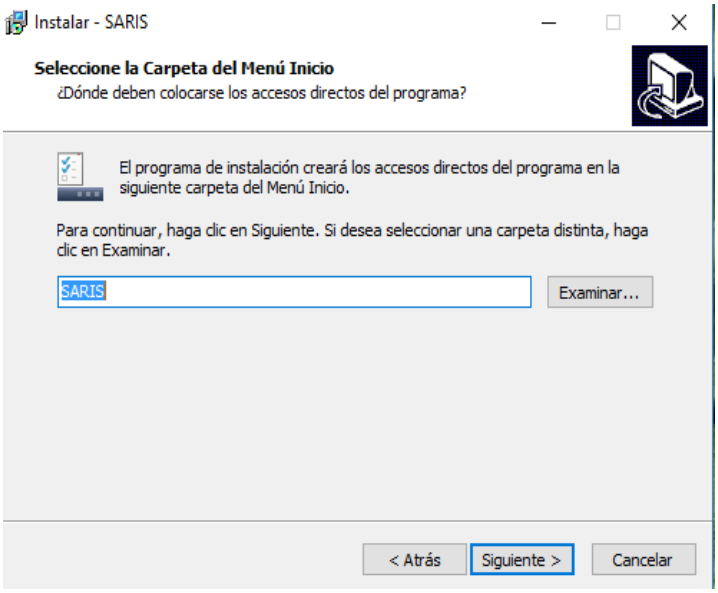

Figura 13. Seleccionando la carpeta donde se va a instalar el programa.

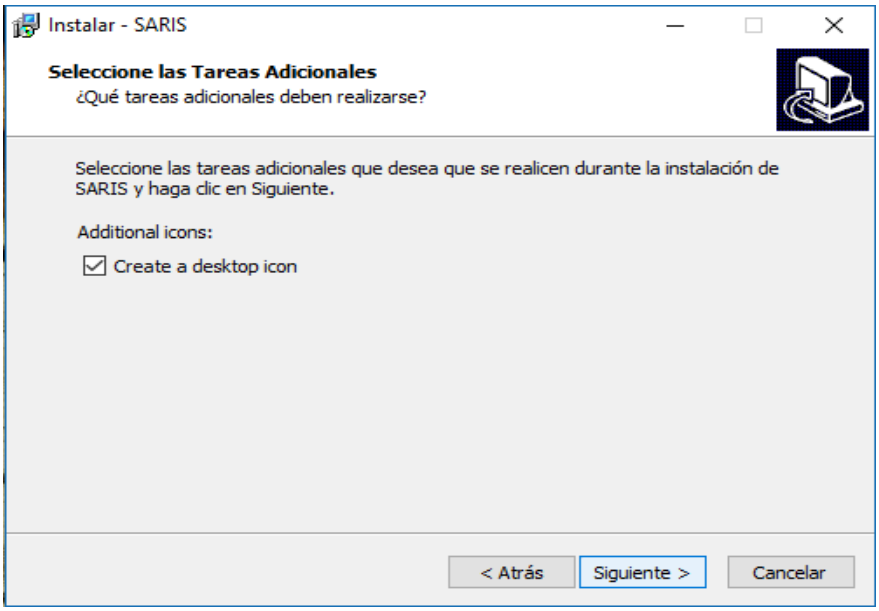

Figura 14. Además de la instalación se crea un acceso directo en el escritorio.

vi. En este punto está todo listo para instalarse, sólo hay que seleccionar el botón Instalar.

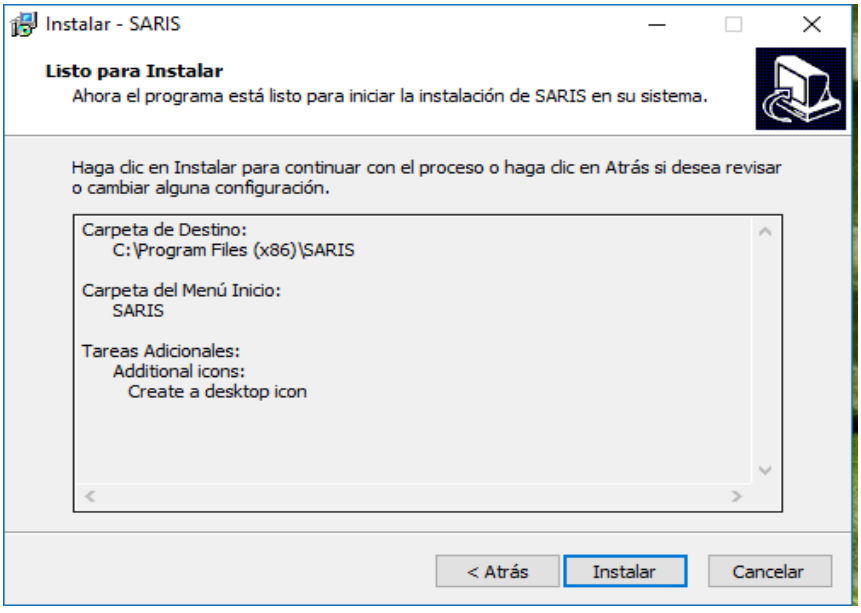

Figura 15. Listo para instalar el programa SARIS.

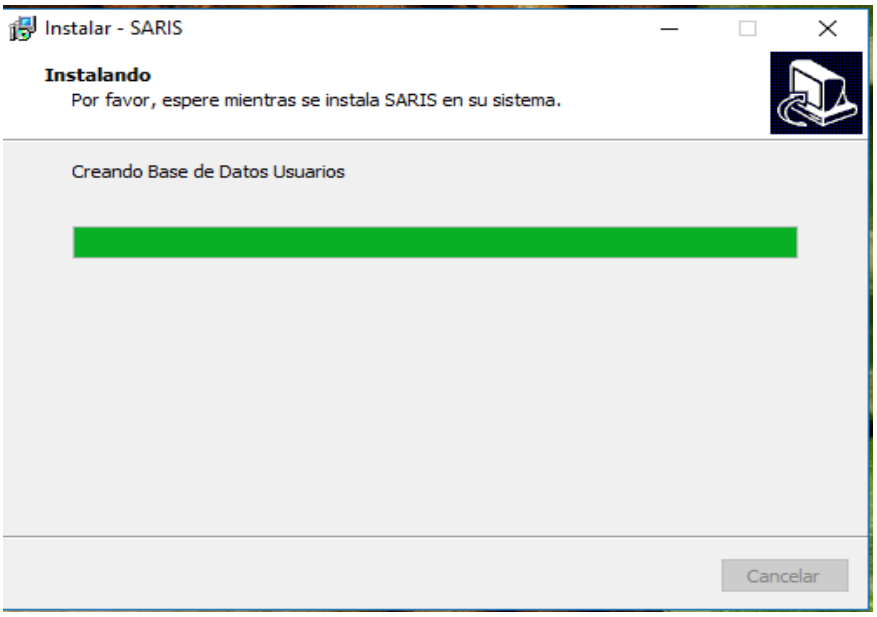

Figura 16. Instalación de SARIS

Figura 17. Instalación finalizada.

Figura 18. Acceso directo de SARIS (mariposa naranja).

2. Conexión del nodo EB con la computadora

Una vez instalado el programa SARIS se puede proceder a conectar el nodo EB a la computadora instalada como servidor. En este punto se debe realizar la conexión como se indica en la figura 21.

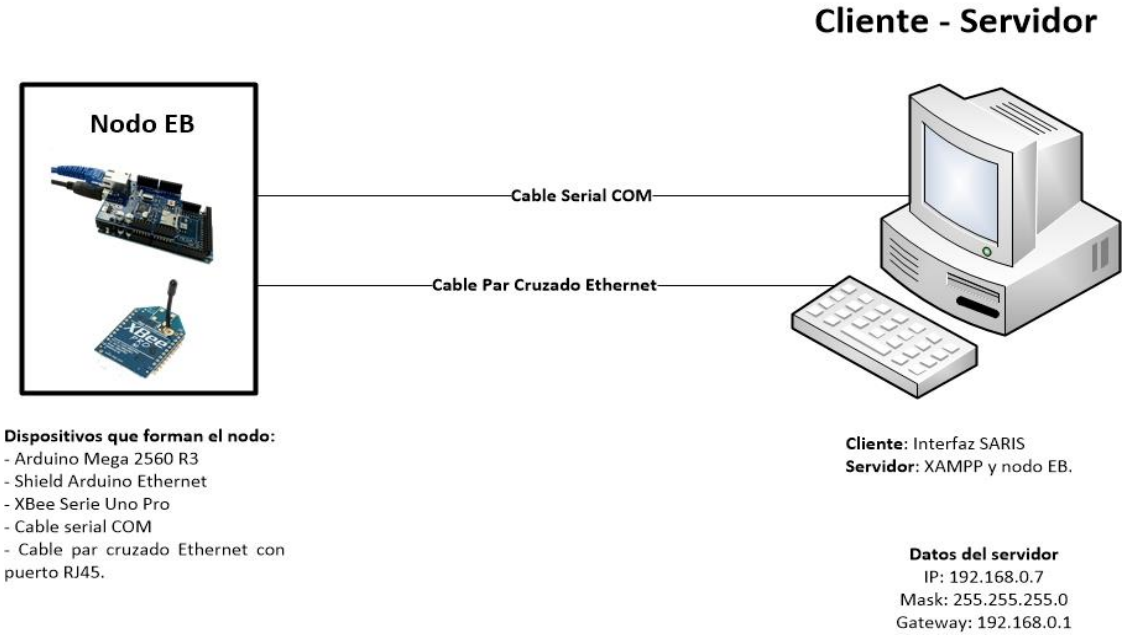

Figura 19. Esquema del nodo Estación Base.

2.1. Asignar dirección IP a la computadora

i. Acceder a la siguiente ruta: **\Panel de control\Redes e Internet\Conexiones de red**.

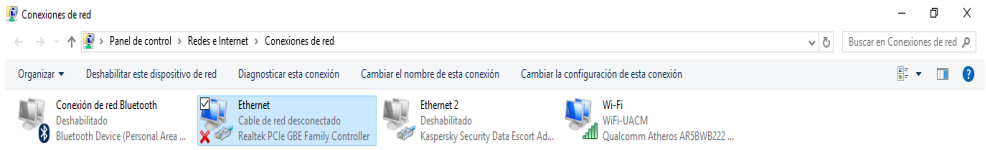

Figura 20. Configuración de la conexión de red entre la computadora y el nodo EB.

i. Dar doble click a la opción Ethernet y debe aparecer la pantalla que aparece en la Figura 23, donde se debe seleccionar la opción: Protocolo de Internet versión 4 (TCP/IPv6) y dar click al botón que se activa Propiedades.

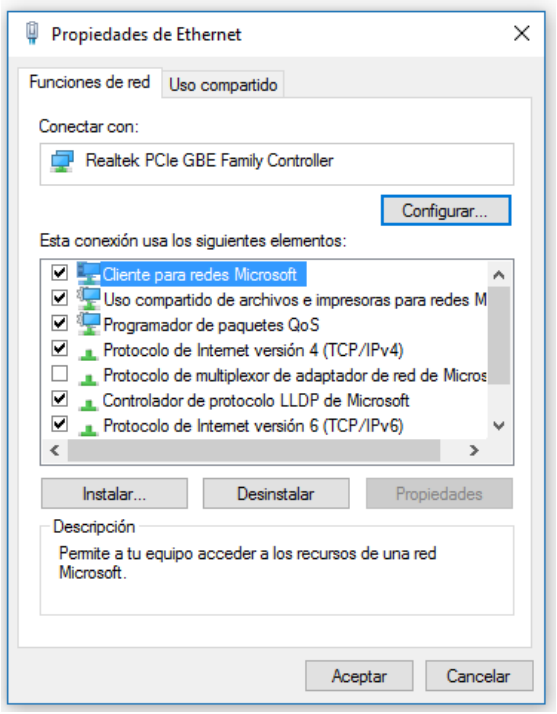

Figura 21. Seleccionar TCP/IPv4.

ii. Asignar las direcciones indicadas en la Figura 24 y dar click en el botón Aceptar. Estás direcciones corresponden a las de la computadora que se configura como servidor.

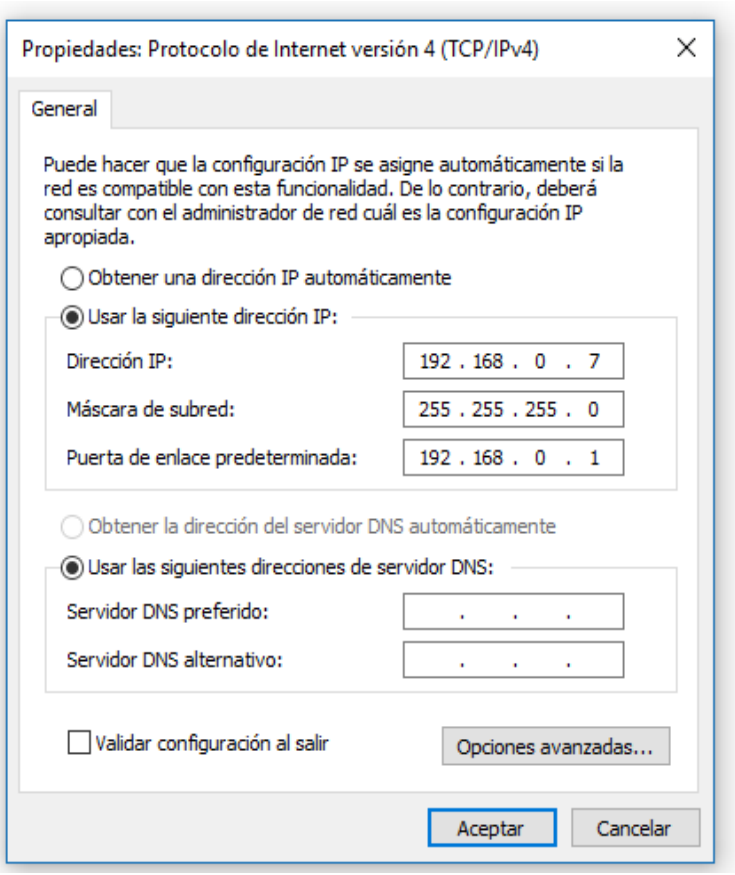

Figura 22. Asignación de direcciones para la red Arduino Mega – Computadora.

iii. Comprobar que Arduino Mega y la computadora, esto se hace ejecutando el Símbolo del Sistema.

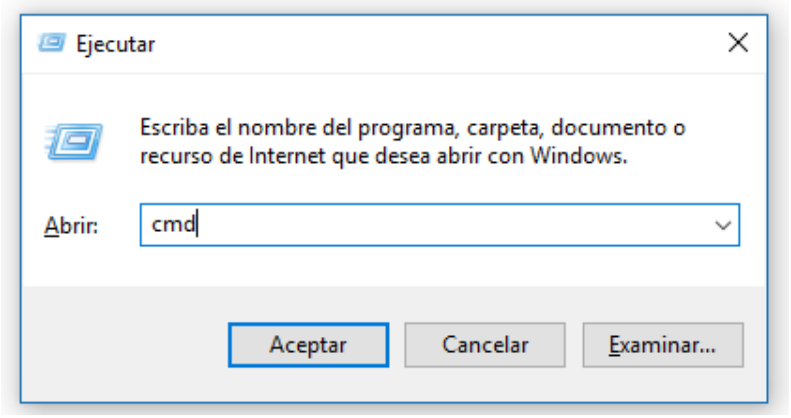

Figura 23. Abriendo el símbolo del sistema.

iv. Ejecutar el comando ping más la dirección IP de Arduino que es 192.168.0.2, como se observa en el ejemplo de la Figura 26 que realiza el ping con otra dirección IP. Si la tarjeta

Arduino Mega está en red se debe tener la misma respuesta que en la Figura 26, es decir, indicar 4 paquetes enviados y 4 paquetes recibidos.

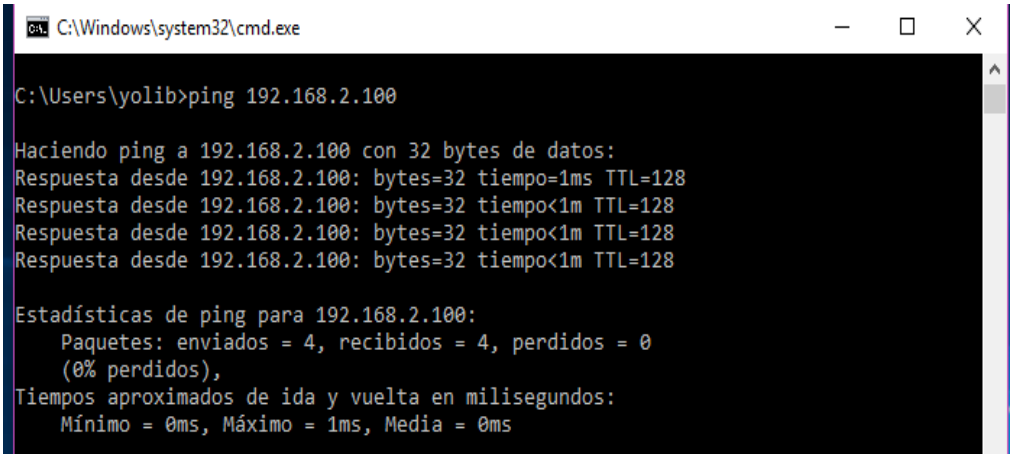

Figura 24. Comando ping en el Símbolo de Sistema.

2.2. Conectar XBee con la tarjeta Arduino

El módulo XBee permite que se tomen los datos de la Red Inalámbrica de Sensores, es por ello que se debe conectar una vez que se haya montado el Shield de Ethernet en Arduino Mega, se puede conectar el módulo XBee. La alimentación se puede tomar del Shield de Ethernet (V_{CC} = 3 volts) y las pines transmisión y recepción de datos entre XBee y Arduino Mega se hacen mediante el puerto serial uno (RX1 y TX1).

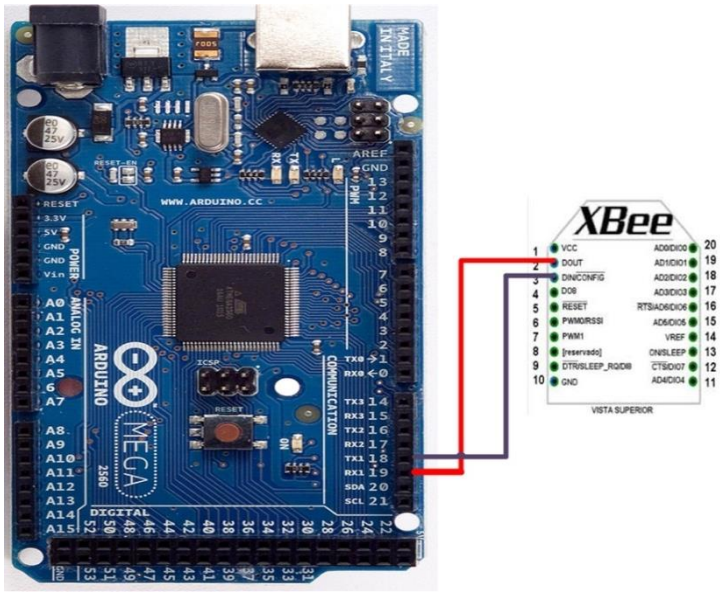

Figura 25. Esquema de conexión del puerto serial uno con el módulo XBee.

¡Hecho todo lo anterior el sistema SARIS está listo para utilizarse!

3. Interfaz SARIS

El sistema SARIS define dos tipos de usuarios uno es el usuario *Administrador* y el segundo el usuario *Lector*, ambos se identifican por el sistema una vez que el usuario se autentifica, por lo que se abren pantallas diferentes para cada usuario.

Figura 26. Pantalla de Autenticación.

El proceso de autenticación es cuando un usuario escribe en los campos de texto el nombre de usuario y la contraseña (Figura 28). Para ingresar a SARIS se debe seleccionar el botón Entrar una vez que se haya teclado los datos correspondientes, de haber cometido un error el sistema lo mostrará y se deben teclear los datos de nuevo, como en esta primera pantalla sólo se admiten los números, letras del abecedario en minúsculas y mayúsculas se tiene que seleccionar el botón Limpiar cuando se cometa algún error de captura.

3.1. Usuario Administrador

El sistema admite sólo un usuario administrador y no más, este usuario tiene definida las tareas: *administrar usuarios* y *tomar muestras* de la Red Inalámbrica de Sensores.

i. La administración de usuarios se hace en la pantalla Registro de usuarios, esta tarea consiste en:

- Poder ingresar nuevos usuarios.
- Actualizar los datos de los usuarios ya registrados-
- Borrar usuarios existentes.
- Buscar usuarios ya registrados.

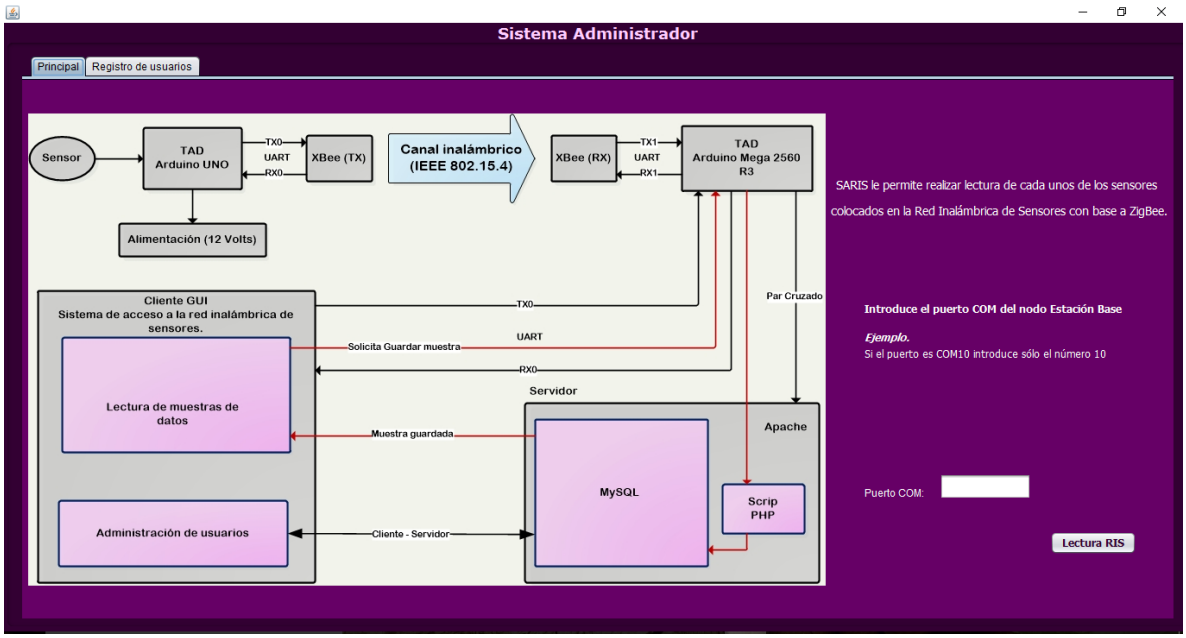

Figura 27. Interfaz del usuario Administrador.

ii. Para guardar un usuario basta con llenar los campos de texto que se encuentran en el recuadro de la izquierda y seleccionar Lector como tipo de usuario en el menú desplegable y finalmente dar click en el botón Guardar.

iii. Para actualizar un usuario ya existente seleccionar un usuario de la tabla de usuarios que se encuentra en el recuadro de la derecha, seleccionar el botón Modificar.

Al seleccionar el botón Modificar los datos de usuario aparecen en los campos de textos del recuadro de la izquierda, donde se puede modificar cualquiera de ellos excepto el Tipo de Usuario que siempre debe seleccionarse Lector. Una vez capturados los campos que se desean modificar dar click en el botón Actualizar, hecho esto el sistema avisa que los datos han sido actualizados.

iv. Borrar usuarios existentes es una tarea que se debe tomar con cautela ya que el sistema no envía ningún mensaje que solicite una confirmación de esta tarea por parte del usuario. Además nunca se debe borrar el usuario Administrador. Para borrar un usuario basta con seleccionar un usuario de la tabla, asegurarse que sea el usuario que desea eliminar y finalmente seleccionar el botón Eliminar.

v. Buscar usuarios existentes en la base de datos, para ello debajo del botón Buscar hay un menú desplegable que permite hacer la búsqueda de usuarios ya sea por: **nombre**, **apellido**, **correo** y **usuario**, se debe seleccionar uno y luego escribir el dato en el campo de texto. Finalmente debe dar click en el botón Buscar.

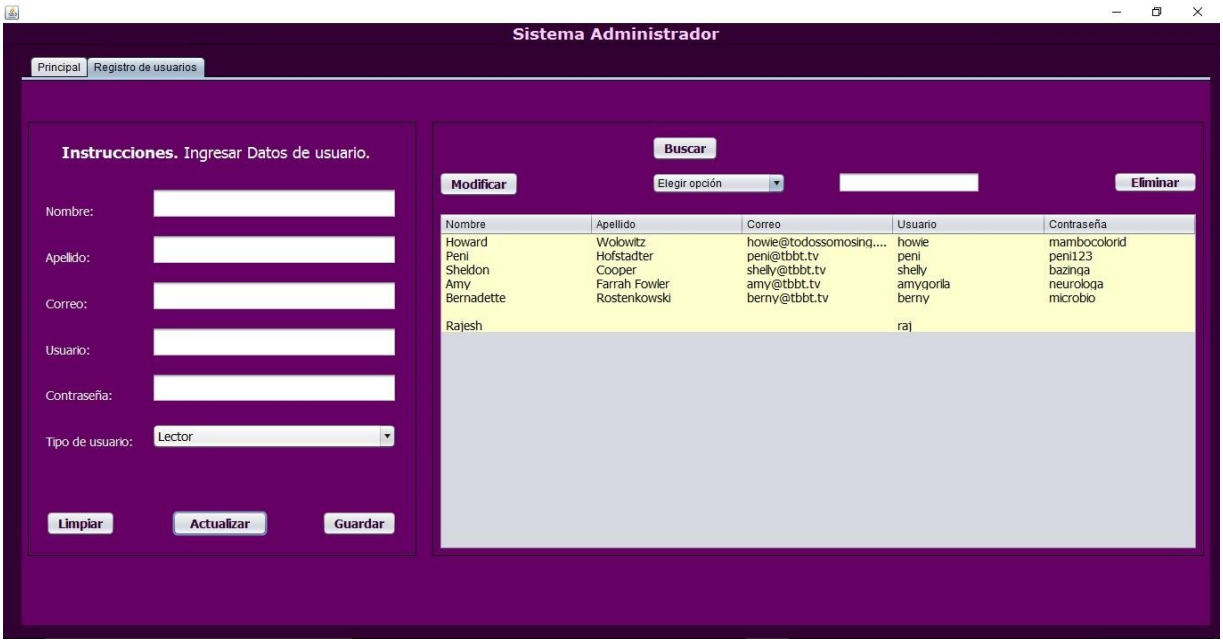

Figura 28. Pantalla Registro de usuario.

3.2. Usuario Lector. Tomar muestras de la RIS

Tanto el usuario Lector como el usuario Administrador pueden realizar la tarea de tomar muestras de la Red Inalámbrica de Sensores (RIS). Los botones S1, S2, S3 y S4 corresponde a los nodos sensor que se muestran en la figura 1, al seleccionarlos, se guardan las lecturas de los sensores en los la base de datos que almacena esta información.

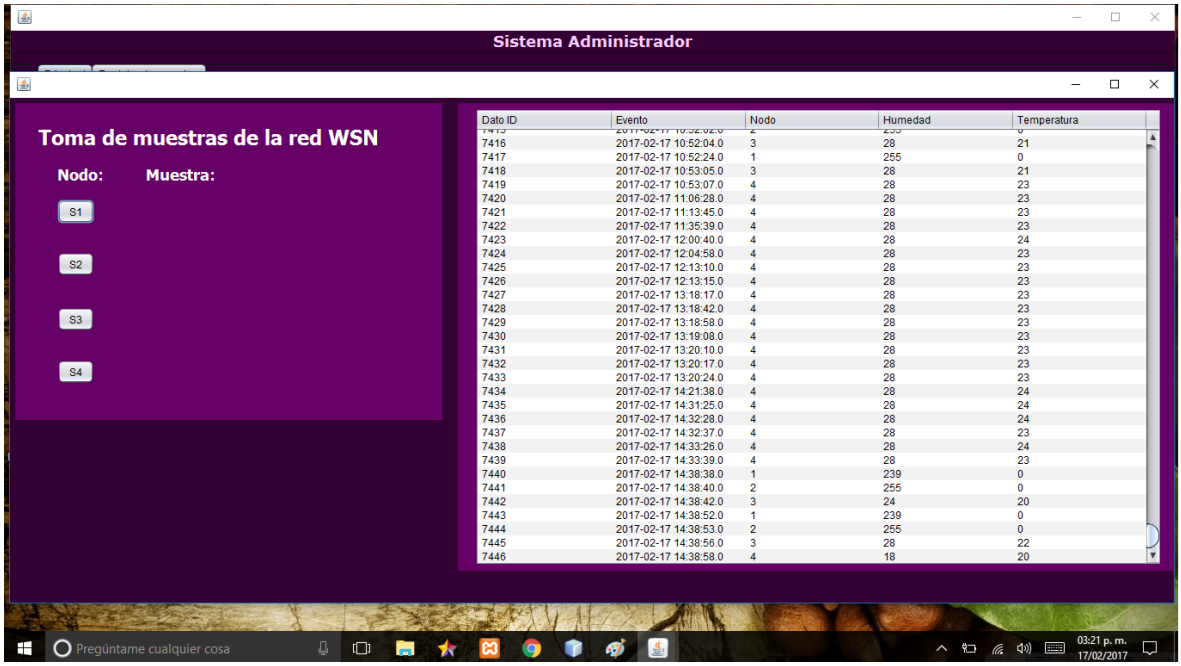

Figura 29. Pantalla que permite al usuario hacer las muestras de los sensores colocados en los nodos S. En esta pantalla se puede ver que aún no se toma ninguna muestra.

Apéndice B

Desglose de la cotización realizada el 6 de Abril de 2017.

Cantidad	Descripción	Precio Unitario	Subtotal
1	Arduino Mega 2560 Rev3	$ 734.44	$ 734.44
1	Shield Ethernet compatible con Arduino	$ 199.14	$ 199.14
1	10 cables jumper macho - macho	$ 81.03	$ 81.03
1	10 cables jumper macho - hembra	$ 87.93	$ 87.93
3	Shield Xbee Arduino	$ 170.21	$ 510.64
4	Arduino Uno Rev3	$ 340.24	$ 1,360.95
2	Sensor de humedad/temperatura DHT11	$ 31.03	$ 62.07
2	Sensor de humedad del suelo ISDT-027	$ 196.16	$ 392.32
3	Modulo adaptador para radios Xbee	$ 36.68	$ 110.04
5	Xbee-PRO Serie 1, 100 mW, Wire 250 kbps XBP24	$ 661.19	$ 3,305.95
		Subtotal sin IVA	$ 6,844.51
		IVA (16%)	$ 1,095.12
		TOTAL	$ 7,939.63