PROTOTIPO DE HERRAMIENTA DIDÁCTICA PARA MONITOREO DE VARIABLES FÍSICAS

ANA DANIELA CAICEDO SALAZAR

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELETRÓNICA
SAN JUAN DE PASTO
2016

PROTOTIPO DE HERRAMIENTA DIDÁCTICA PARA MONITOREO DE VARIABLES FÍSICAS

ANA DANIELA CAICEDO SALAZAR

Trabajo de grado para optar por al título de Ingeniería Electrónica

Director:
JUAN CARLOS CASTILLO, MSc.

Asesor:
JESUS INSUASTI, MSc.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELETRÓNICA
SAN JUAN DE PASTO
2016

NOTA DE RESPONSABILIDAD

“La Universidad de Nariño no se hace responsable por las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor.”

Acuerdo 1. Artículo 324. Octubre 11 de 1966, emanado del honorable Consejo Directivo de la Universidad de Nariño.

Nota de Aceptación:

_____
_____
_____
_____
_____
_____
_____

Firma del Presidente del Jurado

Firma del Jurado

Firma del Jurado

San Juan de Pasto, Febrero de 2016.

AGRADECIMIENTOS

Agradezco a:

Agradezco a Dios por su fidelidad y apoyo para cumplir mis metas, a mis padres y amigos incondicionales que me apoyaron en este proceso, a mi director y especialmente a mi asesor por su empeño, paciencia, tiempo, dedicación, apoyo incondicional y colaboración en esta investigación, también agradezco al grupo de investigación GALERAS.NET por brindarme colaboración y un espacio agradable para el desarrollo de mi proyecto, además manifiesto un sincero agradecimiento al ingeniero Rolando Barahona por su colaboración y por brindarme un espacio en su clase para que los estudiantes pudieran interactuar con lo que realice en este proyecto.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

En este proyecto, se presenta la construcción de un prototipo de herramienta didáctica, basada en una aplicación móvil que recibe, interpreta los datos de variables físicas como temperatura, humedad y luminosidad, obtenidas por medio de una red de sensores y con los cuales se realizó actividades didácticas inmersas en la misma teniendo en cuenta un enfoque didáctico en el área de instrumentación electrónica. Para esto se dividió el trabajo en 3 partes: el diseño e implementación de *hardware* que corresponde al de una red de sensores, el diseño e implementación de *software* que corresponde al desarrollo de una app móvil para dispositivos móviles con sistema operativo *Android* y como tercera parte el diseño de un sistema de comunicaciones basado en tecnología inalámbrica *WiFi*. Esto con el fin de interconectar los componentes de la herramienta como *hardware* y *software*; Así entonces con la realización de cada una de las partes se construyó la herramienta didáctica para monitoreo de variables físicas a manera de prototipo.

Además, se realizó una experiencia con los estudiantes de sexto semestre del Programa de Ingeniería electrónica, con el fin de evaluar el impacto de la herramienta como apoyo al proceso de aprendizaje en el área instrumentación electrónica, mediante la aplicación de una encuesta y posteriormente el análisis de los resultados.

ABSTRACT

In this project, the construction of a prototype as a teaching tool based in a mobile application is presented, this prototype receives and interprets the data of physical variables such as temperature, humidity and light intensity which were obtained through a network of sensors, where activities were performed within the same didactic approach in the area of electronic instrumentation teaching. This work consists in three parts: design and implementation of hardware which corresponds to a network of sensors; software design and implementation on the development of a mobile app for mobile devices with Android operating system and the design of the communication system based on WiFi in order to connect the components of the tool as hardware and software; So then with the completion of each of the parties an educational tool for monitoring of physical variables as a prototype was built.

In addition, it was performed an experience with the students of the sixth semester of the Electronic Engineering Program, in order to evaluate the impact of the tool to support the learning process in the electronic instrumentation, area by applying a survey and subsequent analysis of the results.

INTRODUCCIÓN

Debido a la necesidad de obtener mejores resultados por parte de los estudiantes, a la hora de experimentar y realizar prácticas de laboratorio para determinados procesos, es importante realizar proyectos de investigación con fines pedagógicos [11][18], para así aprender acerca de la ciencia y la tecnología de una manera interactiva, con el fin de conocer los fundamentos de dispositivos electrónicos que desempeñan un papel importante en el campo de la industria, medicina, telecomunicaciones, control, e instrumentación electrónica, donde su uso es cada día más común para el monitoreo y control de variables de un proceso.

En la actualidad el uso de dispositivos móviles constituye una realidad que ha logrado que cada vez su utilización sea cada vez más común, ya que en ellos se encuentra gran variedad de aplicaciones para diferentes usos. Estas aplicaciones se diseñan para dispositivos móviles con diferentes sistemas operativos como para *IOS*, *Windows Phone* y el más utilizado *Android* [12][13], este último ha ido aumentando cada vez más su complejidad, además está basado en el núcleo de Linux[12][13] y todas sus aplicaciones se escriben en lenguaje Java, disponiendo de una máquina virtual específica llamada *Dalvik* [12][13].

Las aplicaciones móviles se realizan para diferentes uso como por ejemplo para realizar monitoreo de diferentes variables en varios campos y para esto se necesita captar datos provenientes del mundo real por lo cual, además del *software* de la aplicación, se necesita de un *hardware* con dispositivos que capten magnitudes físicas, estos dispositivos se denominan sensores con los cuales se construyen redes que facilitan la obtención de información de diferentes lugares, a estas redes se las conoce como redes de sensores inalámbricas WSN[15] y utilizan principalmente la tecnología *Zigbee* [14][15] y el estándar IEEE 802.15.4[14] que es la base de esta tecnología, además es el más utilizado debido a su alcance, bajo coste, bajo consumo de potencia, y también porque permite la configuración de la red en varias topologías como estrella, árbol, malla entre otras. Esta tecnología para su implementación utiliza módulos de RF XBEE [6].

Además, las topologías de red [34] para la configuración de las WSN (*Wireless Sensor Networks*) [15], permiten interconectar un gran número de nodos sensores que además de tener la capacidad de sensar, tienen la capacidad de almacenar, procesar y transmitir datos de forma inalámbrica.

Adicional a esto para el procesamiento de la señal es necesario utilizar dispositivos que tengan ADC (*Analog Digita Conversor*) como por ejemplo los pics y arduinos [16] los cuales son de gran ayuda para la lectura de señales analógicas provenientes de sensores, y además son fáciles de configurar.

Por otro lado, la tecnología *WiFi* proporciona una solución efectiva y con un amplio alcance para un gran número de aplicaciones. Y dentro de estas es importante tener en cuenta estándares y protocolos de comunicación.

1. MARCO TEORICO

1.1 SENSOR

Un sensor es un dispositivo que permite captar magnitudes físicas, medirlas y de igual forma interactuar con el entorno.

1.1.1 Tipos. Existe gran cantidad de sensores los cuales captan distintas magnitudes físicas; según el aporte de energía los sensores pueden clasificarse en moduladores o generadores. Los sensores moduladores son aquellos en que la energía de la señal de salida es proveniente de una fuente de energía auxiliar; en cambio en los sensores generadores, la energía de la señal de salida es adquirida por la señal de entrada. [14] [15]

Según la señal de salida los sensores se dividen en analógicos y digitales. Los primeros presentan una señal de salida variable y continua en el tiempo y los digitales proporcionan una señal de salida que varía en forma discreta, por lo que no requieren de un conversor análogo digital.

Según su tipo de relación entrada salida se dividen en: primer orden, segundo y de orden superior, donde su clasificación dependerá del número de elementos almancenadores de energía independientemente del sensor, los cuales afectan la exactitud y velocidad de respuesta del sensor.

1.2 RED DE SENSORES [14][15]:

Una red de sensores es un sistema constituido por nodos, dentro de los cuales se encuentran los *routers,* dispositivos finales y un nodo principal llamado coordinador. Cada nodo de la red posee uno o varios dispositivos llamados sensores por lo cual su nombre.

1.2.1 Elementos de una red de sensores: [15][33][30]

1.2.1.1 Nodo. Son dispositivos autónomos compuestos por un microcontrolador o microprocesador de baja potencia, una fuente de energía, un sistema de comunicación y uno o varios sensores. [30]

1.2.1.2 Nodo coordinador de la red. Es un nodo el cual sirve como eje central en la recepción de datos, provenientes de los nodos sensores y sirve de vínculo entre el dispositivo al cual se va a enviar estos datos y los nodos de toda la red, razón por la cual es el dispositivo más importante de la red. Además cuenta con capacidad de almacenamiento de datos suficientes y es alimentado por baterías recargables para su funcionamiento autónomo.

1.2.1.3 Nodo router. Es el nodo que se encarga de interconectar de forma inalámbrica los dispositivos que están separados en la red, éste a su vez también puede adquirir datos por medio de sensores. [30]

1.2.1.4 Dispositivo final. Es el nodo que adquiere datos de diferentes variables, dependiendo del tipo de sensor y envían la información directamente al coordinador o a un nodo *router* y luego al coordinador. [30]

1.2.2 Topologías de red. Son las diferentes estructuras de intercomunicación en que se pueden organizar las redes de transmición de datos entre dispositivos. [34]

1.2.2.1 Topología en estrella. En esta topología cada nodo se conecta directamente a un concentrador central, es decir todos los datos pasan a través del concentrador antes de alcanzar su destino. Además esta topología es de bajo costo, crea una mayor facilidad de supervisión y control de información, ya que el concentrador central se encarga de gestionar la redistribución de la información a los demás nodos sin embargo puede presentar inconvenientes a razón de un fallo en el mismo y si ésto sucede se bloqueará la comunicación. [34] [37]

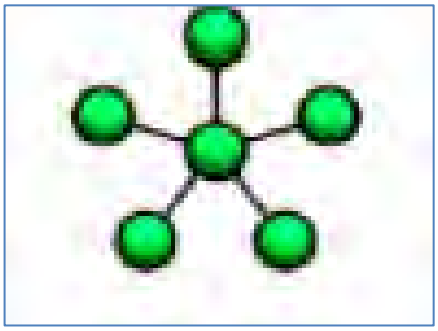


Figura 1. Topología en estrella. Tomado de [34]

1.2.2.2 Topología en árbol. Esta topología se define como un conjunto de nodos configurados en forma de árbol, la conexión en árbol es similar a una serie de

redes en estrella interconectadas, las cuales se conectan a un nodo troncal desde el que se ramifican los demás nodos. [34]

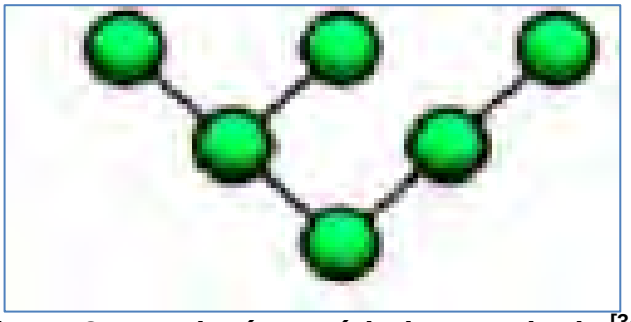


Figura 2. Topología en árbol. Tomado de [34]

1.2.2.3 Topología en malla. La topología en malla se emplea en dos distribuciones de conexión: malla completa y malla parcial. Esta topología presenta la ventaja de que la red puede funcionar, incluso cuando un nodo desaparece o la conexión falla, ya que el resto de los nodos omiten el paso y envían la información por otro camino, razón por la cual esta topología, es muy confiable. Su utilización depende del grado de seguridad que requiera la aplicación. [34]:

Malla completa: Consiste en un enlace directo entre todos los pares de nodos de la red, es una topología de red costosa pero muy confiable. Se usa principalmente para aplicaciones militares.

Malla parcial: en esta topología algunos nodos están organizados en una malla completa, mientras otros se conecta solamente a uno o dos nodos de la red. Esta topología es menos costosa que la malla completa pero tiene la desventaja de que, no es muy confiable.

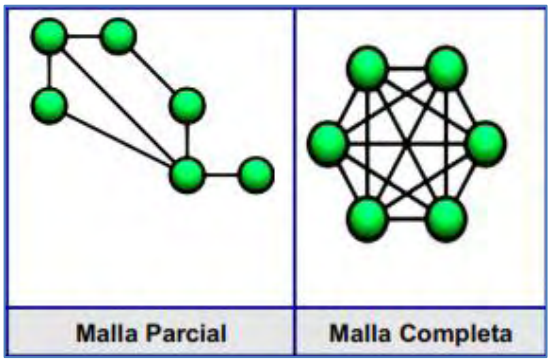


Figura 3. Malla parcial y completa. Tomado de [34]

1.3 TECNOLOGÍAS DE COMUNICACIÓN INALÁMBRICA

1.3.1 Zigbee. Esta tecnología define la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica, para proporcionar una solución para la transmisión de datos, permitiendo arquitecturas de redes inalambricas seguras y confiables. [5][24][14]

Además permite la implementación de una amplia gama de redes inalambricas de bajo costo y bajo consumo de energía. De esta manera proporciona durabilidad y seguridad a nivel de enlace y red para el desarrollo de diferentes aplicaciones que el usuario requiera.

Por otro lado, *ZigBee* soporta múltiples topologías de red como: punto a punto, malla o punto a multipunto, puede contener hasta 65000 nodos por red y opera en las bandas libres de ISM (*Industrial Scientific & Medical*) de 2.4 [15]

Teniendo en cuenta los aspectos de la capa física y MAC de la norma IEE 802.15.4, la tecnología *Zigbee* define las capas de red, aplicación y seguridad, razón por la cual ésta tecnología está basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal WPAN (*wireless personal area network)* [23].

Del mismo modo, *Zigbee* es un protocolo inalámbrico abierto y definido por *Zigbee Alliance*; este protocolo presenta importantes características como velocidad de transferencia, facilidad de implementación de topologías de red, robustez y confiabilidad, por lo cual se puede utilizar en aplicaciones de bajo niveles de transferencia de datos y bajo consumo energético, un ejemplo de esto es el monitoreo de ambientes naturales y control de procesos industriales, en los cuales no se requiere una gran capacidad en la transmisión de datos en cuanto a velocidad y cantidad de datos[38].

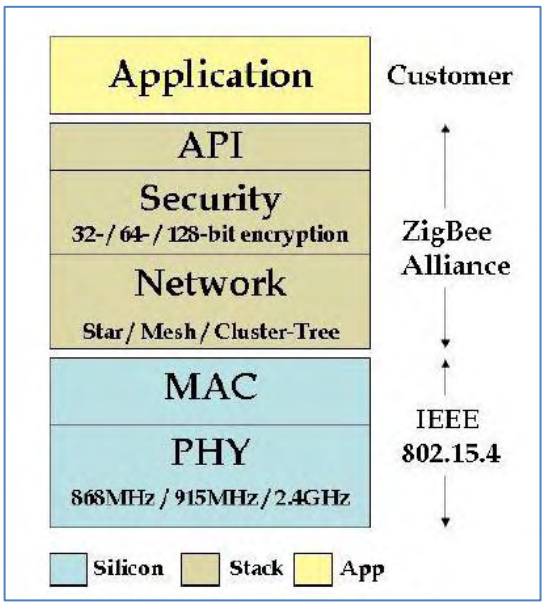


Figura 4. Arquitectura de la tecnología *ZigBee.* Tomado de [24]

En la figura anterior se observa que *Zigbee* trabaja sobre el estándar *IEEE 802.15.4* del mismo organismo, la cual maneja el nivel físico del radio y la capa de medios de accesos, además provee el soporte en el manejo de la seguridad y configuraciones de red y un grupo de interfaces al módulo de radio con la finalidad de facilitar su uso a la aplicación. Es decir, *Zigbee* se encarga del *software* e *IEEE 802.15.4* maneja el *hardware.*

1.3.1.1 Estandar IEEE 802.15.4. En una red de área personal PAN (*Personal Area Network*) [23] el estándar IEE 802.15.4 define la capa física y subcapa de control de Acceso al medio, utilizando como técnica de control de acceso al medio CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance)* con el fin de realizar un acceso múltiple con detección de portadora y evitar colisiones. [14][15]

Ahora bien, el estándar IEEE 802.15.4 es un protocolo de comunicación desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), el cual sirve de apoyo para la transmisión de datos, permitiendo el desarrollo de redes inalámbricas de bajo costo y fácil implementación.

En la siguiente tabla se muestra algunas propiedades del estándar IEE 802.15.4

Tabla 1. Propiedades del estándar IEE 802.15.4

PROPIEDAD	RANGO
Rango de Transmisión de datos	868 MHz: 20 Kb/s 915MHz:40Kb/s;2.4Ghz:250Kb/s
Alcance	10-20m
Canales	868/915MHz:11 Canales 2.4GHZ: 16 Canales
Bandas de Frecuencia	Dos PHY 868/915 Y 2.4Ghz
Direccionamiento	Cortos de 8 Bits o 64 bits IEEE
Canal de Acceso	CSMA-CA
Temperatura	El rango de temperatura industrial:40°+85°C

1.3.1.2 Módulos XBEE. La tecnología *ZigBee* utiliza dispositivos *Xbee* los cuales son módulos de radio frecuencia, que fueron diseñados para ofrecer una solución para la conexión de redes inalámbricas, para aplicaciones de comunicaciones de datos. [6][25]

Además, estos módulos trabajan bajo el estándar IEEE 802.15.4 y en el mercado se encuentran gran variedad de módulos *xbee* los cuales se definen a continuación:

- **XBee Serie 1**:

Estos módulos generalmente son utilizados para comunicación punto a punto y no requieren una configuración muy compleja.

- **XBee Znet 2.5:**

Estos módulos requieren mayor configuración que los de la serie 1 para obtener un buen desempeño, dependiendo del *firmware* con el que se los configure pueden funcionar en modo transparente o trabajar con comandos API, de esta manera se pueden estructurar redes en malla.

Es importante tener en cuenta que estos módulos no son compatibles con la Serie 1, razón por la cual están siendo reemplazados por los módulos ZB en su mayoría compatibles.

- **Módulos ZB** :

Estos módulos poseen el mismo *hardware* de los módulos *Xbee Znet* 2.5 con la diferencia de que se les adaptó un nuevo *firmware*. También pueden funcionar en una red de malla. Generalmente estos módulos son llamados módulos de la Serie 2. [6]

- **Módulos Xbee 900MHz:**

Estos módulos pueden ejecutar 2 tipos diferentes de *firmware*: *DigiMesh* y punto a multipunto y su frecuencia de operación es de 900MHZ.

- **Módulos XSC**:

Básicamente son los módulos *xbee* 900Mhz, con la diferencia de que mientras los módulos *xbee* 900mhz tienen una velocidad de datos de 156KBps, los módulos XSC es sólo tienen una velocidad de 10 Kbps.

- **XSC S3B**:

Esta es una versión mejorada de los módulos XSC ya que tiene mayor potencia de transmisión seleccionable de 250 mw, y cuentan con mayor rendimiento que los módulos XSC [6].

- **XBEE Regular**:

Bajo costo, consumen menos energía y tiene menor alcance.

- **XBEE Pro**:

Mayor costo, consumen más energía y tienen mayor alcance.

Los módulos *xbee* utilizan 4 tipos de antenas Antena *Wire*, PCB, con conector U.FL y conector RPSMA [6].

1.3.1.2.1 Modos de operación. Los modos de operación del módulo *Xbee* son los siguientes: [37] [33][29]

- **Modo recepción/transmisión**

El modo recepción ocurre cuando llega al módulo algún paquete RF por la antena, y el modo transmisión cuando se manda información serial al buffer del pin 3 (DIN), que luego será transmitida. [29]

- **Modo de bajo consumo (*sleep mode*).**

En este modo el módulo se encuentra en un estado de bajo consumo cuando no está en uso. La configuración de éste modo se realiza utilizando comandos, mediante el pin 9 del módulo, el cual es el pin de hibernación llamado *Sleep_RQ*, así cuando este se encuentra en estado alto, el módulo termina cualquier transmisión o recepción y entra en modo reposo y finalmente en modo sueño lo que significa que en este estado el módulo no responderá a comandos entrantes de ningún tipo; sin embargo cuando cambie el estado lógico de *Sleep_RQ* el módulo saldrá del estado de sueño y podrá volver a enviar y recibir datos.

- **Modo de comando.**

Este modo consiste en configurar parámetros del módulo mediante comandos AT. Para poder ingresar los comandos AT es necesario utilizar programas como *Hyperterminal* de *Windows*, *X-CTU* o algún microcontrolador que maneje *UART* (*Universal Asynchronous Receiver Transmitte*) y tenga los comandos guardados en memoria o los adquiera de alguna otra forma.[38]

En caso de tener inconvenientes a la hora de ingresar comandos es importante tener en cuenta que el módulo *Xbee* viene por defecto con una velocidad de 9600bps, ya que estos pueden presentarse por la diferencia de velocidades entre el módulo y la interfaz que se comunica vía serial.

- **Modo transparente [37].**

El módulo se encuentra en este modo cuando lo que ingresa por el pin 3 (DIN), se guarda en buffer de entrada y luego se transmite; de igual manera aquello que ingresa como paquete RF se guarda en buffer de salida y se envía por el pin 2 (DOUT) inmediatamente. Este modo está destinado generalmente para comunicación punto a punto, donde no es necesario ningún tipo de control.

Es importante resaltar que este modo transparente viene por defecto en los módulos *Xbee*.

- **Modo de operación API [29].**

Este modo permite el uso de tramas o *frames* con cabeceras que aseguran la entrega de datos al estilo TCP (*Transmission Control Protocol*). Cuando el módulo opera en este modo, tiene la función de empaquetar en tramas toda la información que entra y sale, también se encarga de definir operaciones y eventos del módulo dependiendo del tipo de trama que transmita. Además es muy completo ya que permite operaciones como transmitir información a múltiples destinatarios, recibir

estado de éxito o fallo de cada paquete RF transmitido e Identificar la dirección de origen de cada paquete recibido.

- **Modo Idle.**

El módulo se encuentra en modo idle cuando no se encuentra en ninguno de los estados anteriormente mencionados.

1.3.1.2.2 Configuración [37]. Para el funcionamiento de los módulos xbee, se requiere de una configuración mediante el *software* X-CTU, [47] con ayuda de una placa denominada *Xbee Explorer,* [48] para así poder emplearlos en cualquier aplicación.

- **XCTU**[47]:

XCTU es un *software* que está basado en *Windows* y fué proporcionado por la empresa de *digi,* para interactuar con módulos *xbee* a través de diversas herramientas que permiten configurar el *firmware* de los mismos y también probarlos.

- **Xbee Explorer**[44]:

Es una placa que sirve para comunicar el pc con el módulo *Xbee*, tiene un chip FTDI que hace de puente entre el USB del PC y la UART del microcontrolador. Además funciona con todos los módulos *Xbee*, también posee Leds de estado para encendido de la tarjeta y estado de la comunicación serial. (Tx/Rx).

Figura 5. Placa *Xbee* Explorer. Tomado de [44]

1.3.2 WIFI (*Wireless Fidelity).* La tecnología *WiFi* una red de comunicaciones de datos que se encuentra estandarizado por la IEEE (*Institute of Electrical and Electronics Engineers*), además está basada en el protocolo internacional 802.11x que implementa los niveles inferiores del modelo OSI, es decir el nivel físico y el de enlace, sobre un canal inalámbrico. [41][28]

En las redes *WiFi* se puede utilizar dos tipos de topologías [27] para la configuración de la red como:

Sin infraestructura: este tipo de topologías no necesitan un sistema fijo que interconecte algunos elementos de la arquitectura, es decir no necesita de un punto de acceso, razón por la cual para comunicarse entre sí tienen que utilizar el mismo canal radio y configurar un identificador específico de *WiFi* en modo ad hoc.

Red en modo infraestructura: en este tipo topología se utiliza uno o más puntos de acceso, permitiendo gestionar y transportar cada paquete de información a su destino, mejorando así la velocidad [41]

Las bandas de frecuencia utilizadas por las redes inalámbricas y especialmente por los equipos *WiFi* en su mayoría son las de 2,4 y 5 GHz.

El alcance de la red *WiFi* dependerá de:

- La potencia del punto de acceso.
- La potencia del dispositivo por el cual se realiza la conexión.
- Los obstáculos que la señal tenga que atravesar.

Ahora bien, con respecto a la confidencialidad la tecnología *wiFi* utiliza las siguientes sistemas de seguridad: [40] [26]

WEP (*Wired Equivalent Privacy*): Es un tipo de encriptación que soporta la tecnología *WiFi;* su codificación puede ir de 64 bits hasta 128 bits, utilizando código ASCII o Hexadecimal. Este sistema de seguridad es vulnerable, ya que es sencillo obtener la manera cómo han sido cifrados los datos, la clave está fija y es la misma para todos los usuarios de una red. Por esta razón no es recomendable para aplicaciones que requieran mayor seguridad.

WPA (*WiFi Protected Access*): es un Sistema de cifrado su codificación se basa en el cambio periódico de claves de acceso y fue creado para eliminar las principales debilidades de seguridad de las redes WEP, sin embargo las claves de cifrado son estáticas, cosa que lo hace todavía vulnerable. [26]

WPA2 *(WiFi Protected Access2)*: Utiliza un nuevo sistema de cifrado, denominado AES (*Advanced Encryption Standard*), considerado el más seguro conocido actualmente en cualquier tipo de red. Además, en este sistema la clave cambia en cada sesión y es diferente para cada usuario. Sin embargo tiene como desventaja que no se puede conseguir la primera clave, y esto genera problemas al momento de utilizarla para la autentificación inicial, y para conseguir todas las claves subsiguientes.

También la tecnología *WiFi* presenta los siguientes mecanismos de autenticación

• **Autenticación abierta:** Es el mecanismo de autenticación por defecto que permite que cualquier dispositivo pueda obtener acceso a la red y los datos se transmiten sin ningún tipo de cifrado. **[26]**

• **Autenticación de clavo compartida:** este tipo de autenticación utiliza la clave WEP de la red para autenticar al cliente. Es decir, se envía un texto desde el punto de acceso y que posteriormente el cliente cifra con la clave de red y lo devuelve al punto de acceso.

• **Autenticación por dirección MAC *(Media Access Control*)**: Es un mecanismo de autenticación, basado en listas de control de acceso que contienen las direcciones físicas de los equipos, es decir la dirección MAC de cada equipo así, cada punto de acceso establece las direcciones que son válidas por autenticar un cliente en su red. **[41]**

1.3.2.1 Estándar IEE802.11. La especificación IEEE 802.11 (*ISO*/*IEC* 8802-11) es un estándar internacional que define las características de una red de área local inalámbrica (WLAN). **[41] [27]**

El estándar 802.11 establece los niveles inferiores del modelo OSI *(Open System Interconnection*), para las conexiones inalámbricas que utilizan ondas electromagnéticas. A continuación se muestra las características de los estándares que pertenecen al protocolo 802.11x. (Ver tabla 2)

Tabla 2. Familia 802.11

	802.11	802.11g	802.11a	802.11n
Tasa máxima de datos (Mbps)	11	54	54	300
Tasa real de datos (Mbps)	5	20	22	146
N° de canales disponibles	3	3	12	3 en 2.4GHZ 5 en 12GHZ
Probabilidad de interferencia	Alta	Alta	Baja	Baja
Comportamiento en entornos difíciles	Pobre	Medio	Bueno	Muy Bueno
Compatibilidad	802.11b	802.11b/n	802.11a	802.11a/b/g/n
Frecuencias(GHz)	2.4	2.4	5	2.4 y 5
Seguridad	WEP/WPA/WPA2	WEP/WPA/WPA2	WEP/WPA/WPA2	WEP/WPA/WPA2

Por otro lado, las tecnológicas de la familia 802.11 Independientemente de la banda de frecuencia en que trabajan tienen algunas limitaciones en cuanto a alcance, anchura de banda, calidad de servicio y seguridad.

Los estándares del protocolo 802.11 proveen actualmente seguridad mediante dos atributos: la confidencialidad y la autentificación.

1.3.3 Protocolo http (*Hipertext Transfer Protocol).* Es un protocolo cliente-servidor que articula los intercambios de información entre los clientes web y los servidores http. Además está soportado sobre los servicios de conexión TCP/IP. Se basa en operaciones de solicitud/respuesta; es decir un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud, entonces el servidor responde con un mensaje similar que contiene el estado de la operación y su posible resultado.[36]

Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor y ésta incluye los siguientes parámetros:

- **Una línea de solicitud**: especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizado

- **Los campos del encabezado de solicitud**: son líneas opcionales que permiten aportar información adicional sobre la solicitud y/o el cliente, cada una

está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos y el valor del encabezado.

- **El cuerpo de la solicitud**: son líneas opcionales que deben estar separadas de las líneas precedentes por una línea en blanco.

Por otro lado, una respuesta HTTP es un conjunto de líneas que el servidor envía al navegador. Esta línea incluye los siguientes parámetros:

- **Una línea de estado**: especifica la versión del protocolo utilizado y el estado de la solicitud en proceso mediante un texto explicativo y un código. La línea está compuesta por tres elementos que deben estar separados por un espacio que son: la versión del protocolo utilizado, el código de estado, el significado del código.

- **Los campos del encabezado de respuesta**: son líneas opcionales que permiten aportar información adicional sobre la respuesta y/o el servidor.

- **El cuerpo de la respuesta**: contiene el documento solicitado.

Adicionalmente, para enviar la petición al servidor http se utilizan los siguientes comandos:

GET: con este comando se recoge cualquier tipo de información del servidor, siempre que se pulsa sobre un enlace o se teclea directamente a una URL (*Uniform Resource Locator)*. Como resultado el servidor http envía el documento correspondiente a la URL seleccionada.

HEAD: se utiliza para solicitar la información sobre un objeto es decir características como tamaño, tipo, fecha de modificación.

POST: sirve para enviar información al servidor.

Es importante resaltar que el protocolo http normalmente está configurado para utilizar el puerto 80, ya que la mayoría de *firmware* no bloquean y que un servidor web opera mediante el protocolo HTTP y las peticiones al servidor suelen realizarse mediante HTTP utilizando el método de petición GET, en el que el recurso se solicita a través de la URL al servidor Web. Es decir cuando escribimos una dirección web, el navegador establece una conexión con el servidor web a través del protocolo http.

1.4 SISTEMA OPERATIVO ANDROID [10][12] [13]

Android es un sistema operativo basado en *Linux,* fue pensado para dispositivos móviles; incluye tanto un sistema operativo como *middleware* y diversas aplicaciones [39] de usuario; creado inicialmente por *Android inc* y luego adquirido por *Google*. Todas las aplicaciones para *Android* se programan en lenguaje *Java* y son ejecutadas en una máquina virtual especialmente diseñada para esta plataforma llamada *Dalvik* y es un sistema operativo de código abierto.

Además está formado por alrededor de 12 millones de líneas de código, de estas 2.8 de líneas son de lenguaje c, 2.1 millones de líneas son de java y 1.75 millones de líneas de C++ y 3 millones de líneas son de XML [12]

La arquitectura de *Android* está formada por:

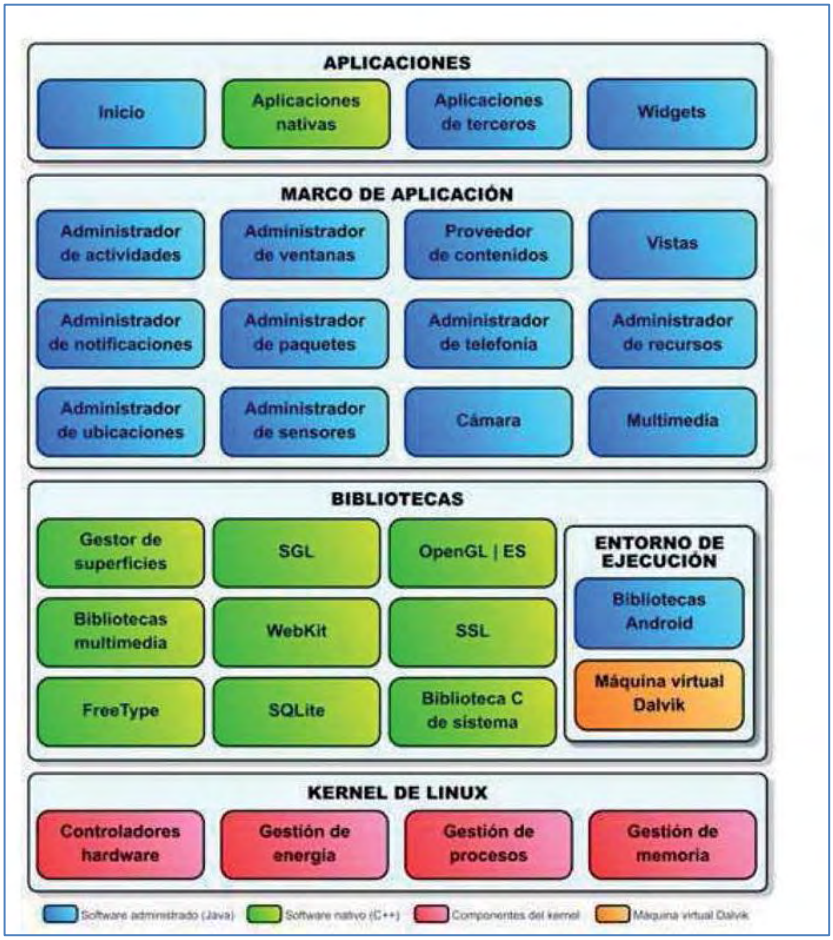


Figura 6. Arquitectura del sistema Operativo *Android.* Tomado de [10]

En *Android* cada aplicación se ejecuta en su propio proceso. La mayoría de las medidas de seguridad entre el sistema y las aplicaciones deriva de los estándares de *Linux* 2.6, cuyo *kernel* constituye el núcleo principal de *Android,*

Adicional a esto, *Android* presenta características como: es adaptable a muchas pantallas y resoluciones, utiliza *SQLite* para el almacenamiento de datos y navegador *web* basado en *WebKit* incluido, soporte de *Java* y muchos formatos multimedia, soporte de HTML, HTML5, *Adobe Flash Player*, etc. También Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del *software.*

Por otro lado, ninguna aplicación tiene permiso para realizar ninguna operación para la ejecución de otras aplicaciones o del sistema como tal; esto es operaciones como de lectura o escritura de ficheros privados del usuario, acceso de red, entre otras ya que no están permitidas.

Además, en *Android* es obligatorio que cada aplicación esté firmada digitalmente mediante un certificado, cuya clave privada sea la del desarrollador de dicha aplicación.

El SDK (*Software Development Kit*) de *Android* incluye un completo emulador que permite probar y depurar eficientemente las aplicaciones que se van desarrollando [10]

1.5 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

1.5.1 Metodologías tradicionales*.* Son aquellas en donde se realiza una intensa etapa de análisis y diseño de la construcción del sistema, uno ejemplo de estas son la metodología RUP y cascada.

1.5.2 Metodologías ágiles. Son aquellas en las que un proceso es ágil, es decir el desarrollo de *software* es incremental, cooperativo y adaptable. Algunas de estas son: XP (*Xtreme Programming***),** *Scrum, Crystal*, *Feature Driven Development*. [19]

2. DISEÑO E IMPLEMENTACIÓN DE HADWARE

En este trabajo se presenta el diseño e implementación de un prototipo de herramienta didáctica para monitoreo de temperatura, humedad y luminosidad, como herramienta de apoyo al proceso de aprendizaje en los estudiantes del programa de Ingeniería Electrónica; de tal manera que se llevó a cabo el cumplimiento de los objetivos propuestos y se dividió el trabajo en 3 partes: diseño e implementación de *hardware,* diseño e implementación de *software* y la conexión entre estas dos partes por medio de un sistema de comunicación inalámbrico basado en tecnología *WiFi.*

2.1 DISEÑO DE HADWARE

2.1.1 Diseño de una red de sensores. Para el diseño de la red de sensores, se realizó una investigación de la estructurada de una red y que componentes debe tener.

Ahora bien, en primer lugar se investigó sobre los tipos de sensores que existen en el mercado; se encontraron gran variedad de sensores pero para este caso se utilizaron los siguientes:

Para medir temperatura se estudiaron los sensores: LM335, DS18S20, DS1722S, LM35-DZ, LM61BIM3 de los cuales se seleccionó el sensor LM35-DZ [2] de *Texas.*

Para medir la humedad relativa se estudiaron los sensores: HIH-4030-002, HS1101 DHT22, DHT11 de los cuales seleccionó el sensor HIH-4030 [1]

Para medir la luminosidad se estudiaron los sensores como: detectores de luminosidad, fotodiodos, LDRs y se optó por utilizar LDR [3].

Ahora bien, para realizar la comunicación inalámbrica para la red se investigó y por las numerosas ventajas que presenta y la mayoría de aplicaciones en las que se han utilizado se seleccionó los módulos *xbee* de *digi*, entonces se estudiaron gran variedad de módulos *xbee* [6] existentes en el mercado como **XBee Serie 1**, **XBee Znet 2.5, xbee serie2, Xbee 900MHz, xbee XSC, xbee S3B** ,**XBEE PRO**.

Teniendo en cuenta lo estudiado anteriormente, se realizó una selección de los componentes que hacen parte de la red y, para esto se tuvo en cuenta el costo, confiabilidad en la medición, fácil utilización y disponibilidad en el mercado. A continuación se presenta las características de todos los componentes utilizados para el diseño e implementación de la red:

SENSOR DE TEMPERATURA LM35 [2]

LM35 es un sensor de precisión, que permite medir la temperatura, opera en un rango de -55 a +150ºC, su salida de voltaje es linealmente proporcional a la de temperatura dando como resultado un voltaje de 10mv/ºC y opera a un voltaje de 4 a 30v.

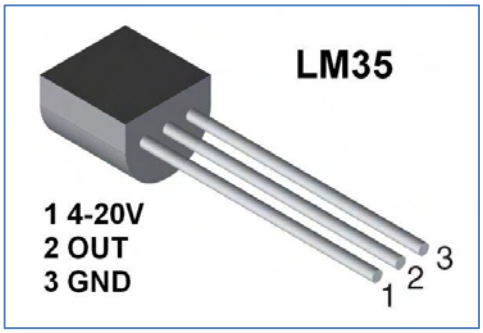


Figura 7. Sensor de Temperatura LM35. Tomado de [2]

SENSOR DE HUMEDAD HIH-4030[1]

HIH-4030 es un sensor que permite medir la humedad relativa (% HR). El voltaje de alimentación debe estar dentro 4-5.8VDC, y óptimamente a 5V. El sensor normalmente sólo consume alrededor de 200µA; La señal de salida es un voltaje proporcional al porcentaje de humedad del área donde se está censando.

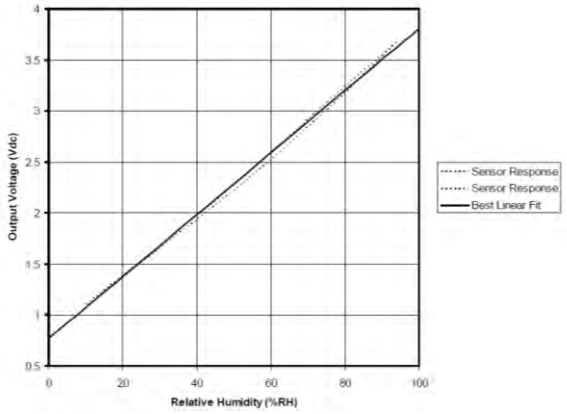


Figura 8. Sensor de humedad. Tomado de [1]

SENSOR DE LUMINOSIDAD

LDR (*Light Dependent Resistor)* [3]

Un sensor LDR es una resistencia de material sensible a la luz de semiconductores, tales como sulfuro de cadmio. La resistencia cae al aumentar la intensidad de la luz. Las aplicaciones incluyen la detección de humo, el control automático de la iluminación, el conteo de lote y sistemas de alarma antirrobo.

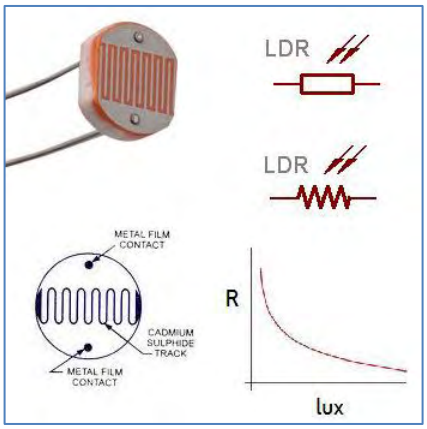


Figura 9. Sensor de luminosidad. Tomado de [3]

Después de haber realizado la selección de los sensores, se hizó el estudio de dispositivos de la empresa de *microchip*, para realizar el procesamiento de las señales provenientes de los sensores para lo cual se estudió gran variedad de microcontroladores de varias características y pines, luego de esto se seleccionó el PIC 16F876A [8] el cual tiene las siguientes tiene 28 pines, tres puertos I / O, catorce interrupciones y cinco canales de entrada A / D.

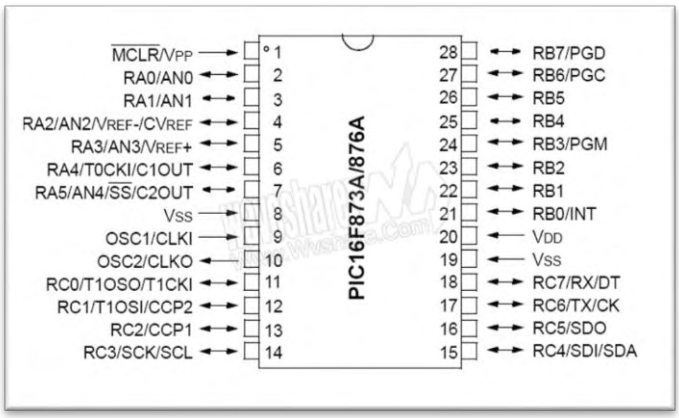


Figura 10. Pic 16F876A. Tomado de [8]

En vista de que, para el desarrollo de este proyecto no se necesitó un alcance en la comunicación inalámbrica muy grande, se seleccionó el siguiente módulo de la serie 2 de *digi.*

XB24-Z7PIT-004[50]:

Es un módulo de comunicación *ZigBee* de 2.4Ghz, 250 Kbps y cumple con el estándar IEEE 802.15.4, con un alcance de 40m en interiores y 120m en exteriores (con línea de vista), además tiene antena PCB integrada, sensibilidad de -96dBm y potencia de transmisión: 1,25mW (+1 dBm).

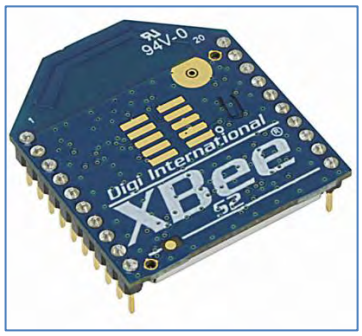


Figura 11. Módulo xbee XB24-Z7PIT-004 SERIE2. Tomado de [6]

Después de la selección de los componentes que van a componer la red se realizó el estudio de las siguientes topologías**:** estrella, árbol y malla. De las cuales se seleccionó la topología en estrella, [34] puesto que ofrece muchas ventajas y por su bajo costo de implementación.

Finalmente teniendo todos los componentes listos, se definió la cantidad de nodos para la red y se los configuró en topología estrella.

Figura 12. Diseño de la red

Teniendo en cuenta el costo de los componentes se decidió hacer la red con 3 nodos sensores y un nodo coordinador.

Para lo cual en primer lugar se realizó la configuración de los módulos *xbee* con el *software xctu* con la ayuda de la placa *Xbee Explorer* así:

NODO CORDINADOR

El módulo *xbee* se configuró como coordinador AT, la mayoría de parámetros del módulo se dejaron por defecto; solo se configuraron los siguientes:

EL PAN ID: 1

DH (Destination High):0
DL (Destination low):0
Node identifier: 0
Operating channel : E

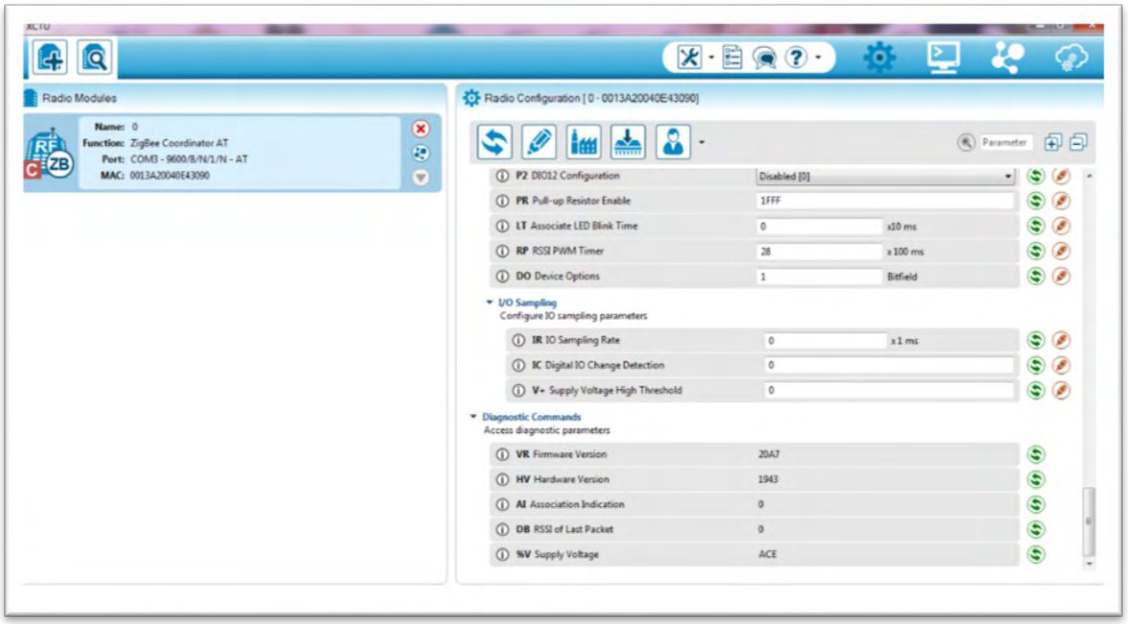


Figura 13 Configuración nodo coordinador

NODO ROUTER1

El módulo *xbee* se configuró como *Zigbee Router* AT, algunos parámetros se dejaron por defecto y los parámetros que se configuraron fueron:

PAN ID: 1
DH: 13A200
DL: 40EA3090
NODE IDENTIFIER: 1
CHANEL VERIFICATION: ENABLED

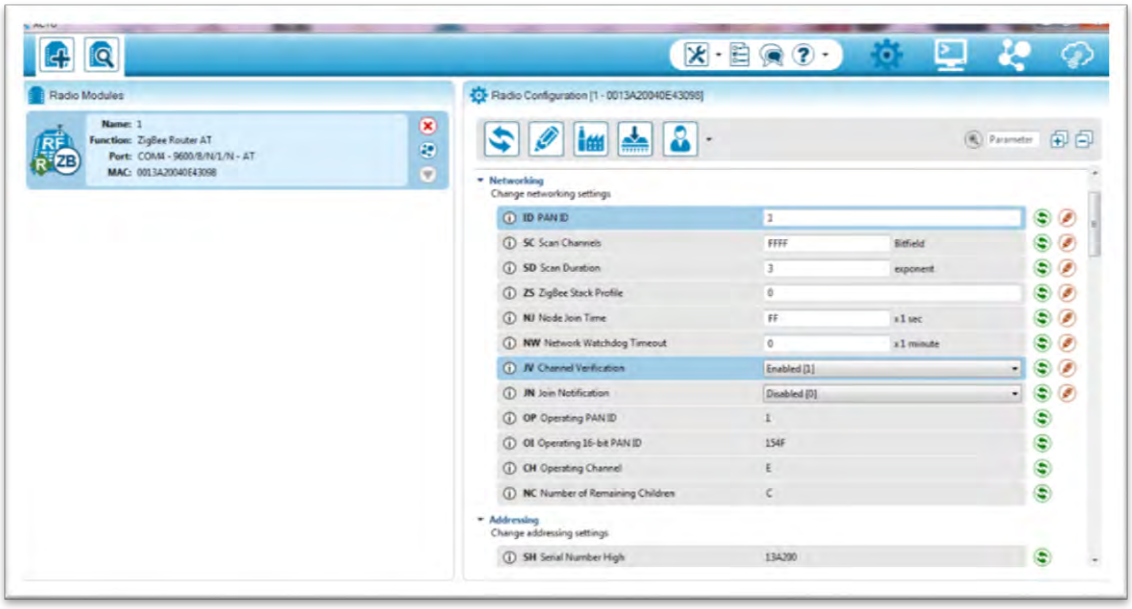


Figura14 Configuración nodo router1

NODO ROUTER2

El módulo *xbee* se configuró como *Zigbee Router* AT, algunos parámetros se dejaron por defecto y los parámetros que se configuraron fueron:

PAN ID: 1
DH: 13A200
DL: 40EA3090
NODE IDENTIFIER: 2
CHANEL VERIFICATION: ENABLED

NODO ROUTER3

El módulo *xbee* se configuró como *Zigbee Router* AT, algunos parámetros se dejaron por defecto y los parámetros que se configuraron fueron:

PAN ID: 1
DH: 13A200
DL: 40EA3090
NODE IDENTIFIER: 3
CHANEL VERIFICATION: ENABLED

Con respecto a lo anterior el PAN ID es el identificador de la red, por lo tanto todos los nodos de la red tienen el mismo PAN ID. DH es la dirección que viene de

fábrica en cada módulo *xbee* en este caso hace referencia al lado superior y DL al lado inferior. Entonces, ya que el coordinador en esta red de sensores solo va a recibir los datos provenientes de los nodos sensores, para éste se configuró el DH y el DL con cero, sin embargo para los nodos 1, 2 y 3 el DH es la dirección que tiene marcada el módulo *xbee* del nodo coordinador en la parte superior y el DL es la dirección del mismo pero que tiene marcada en la parte inferior.

De forma que, teniendo lista la configuración de los módulos se procedió a probarlos en la ventana de simulación del *software xctu,* ésto con el fin de observar si el módulo configurado como coordinador recibe datos provenientes de cualquiera de los nodos configurados como routers.

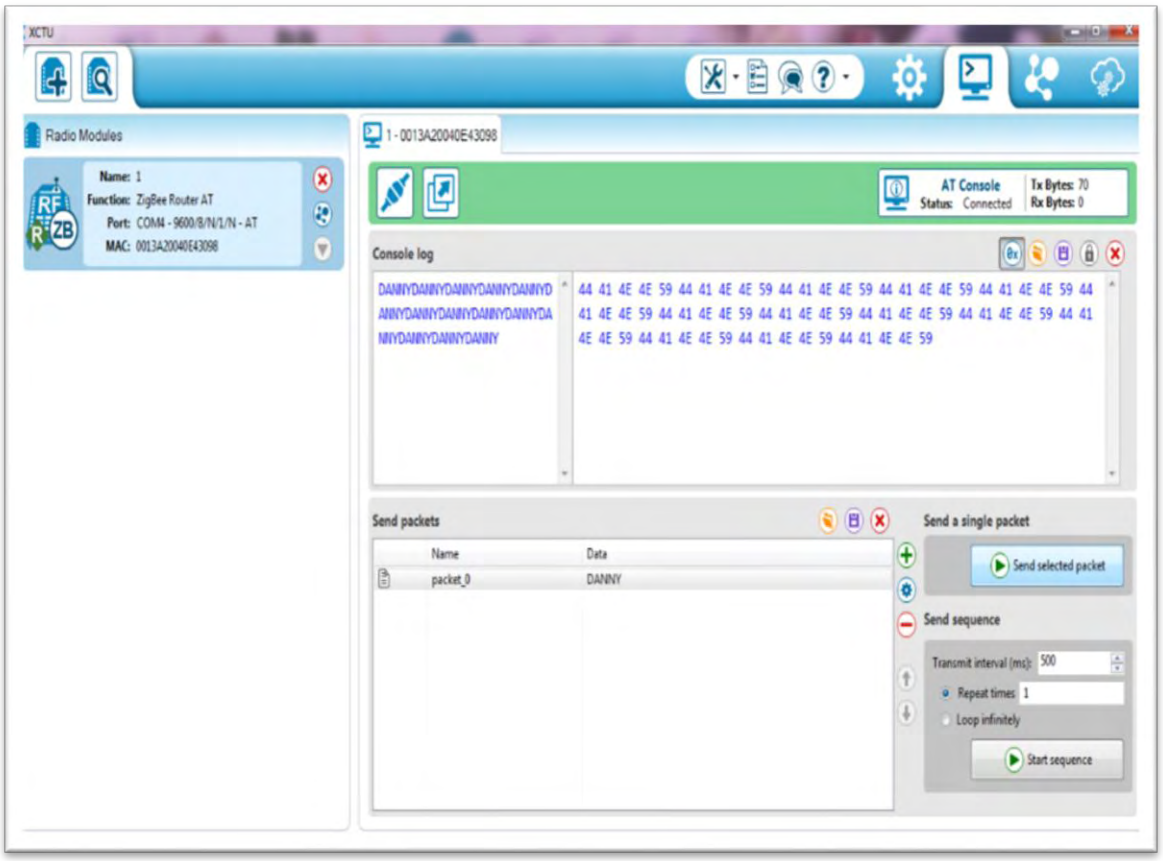


Figura 15. Nodo router enviando datos al nodo coordinador

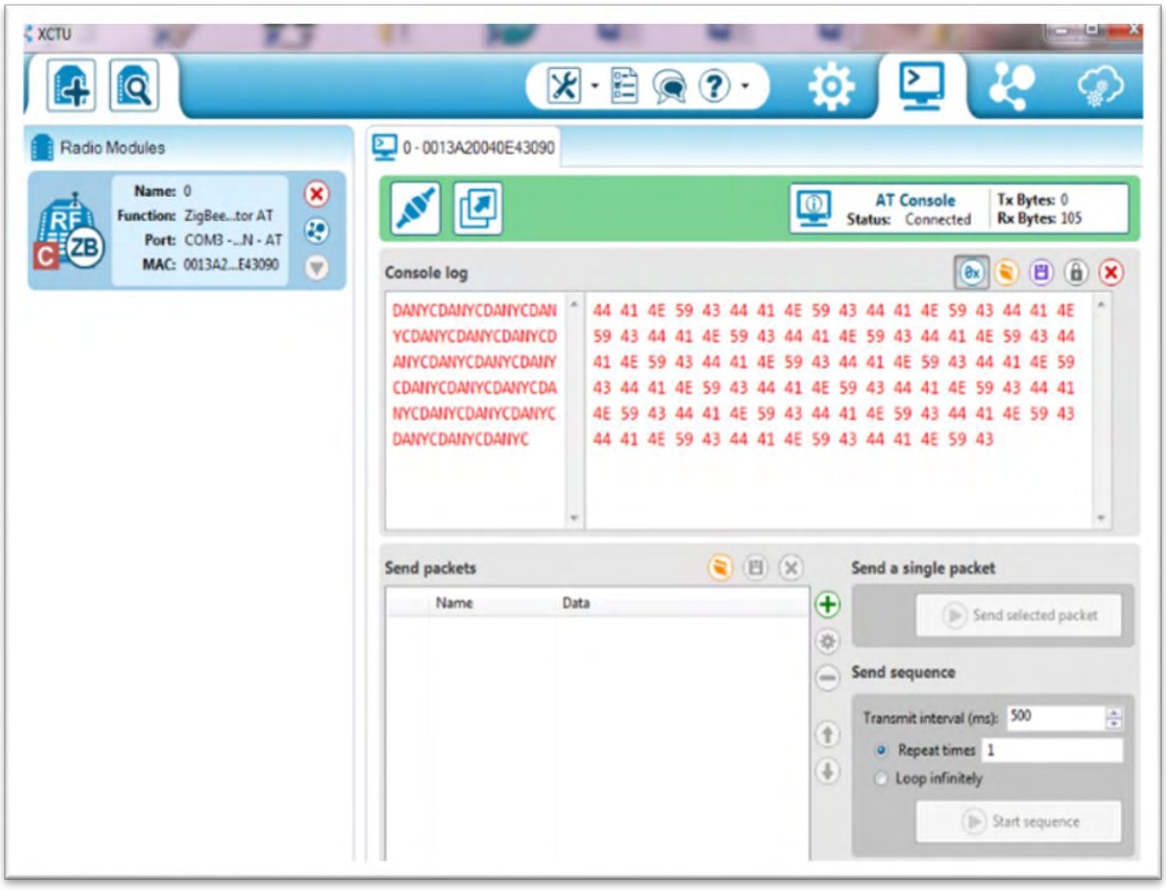


Figura 16. Nodo coordinador recibiendo datos de un nodo router

Ahora bien como se probó los módulos y funcionaron correctamente transmitiendo de forma inalámbrica, se procedió a realizar la programacion del pic 16f876A(ver anexo1).

Luego, se se realizó una simulación en el *software proteus,* en la cual se observó la lectura de las entradas analogicas en un monitor serial.

Debido a que en proteus no se dispone de algunos componentes, se realizó la simulación con potenciometros y se visualizó en un monitor serial:

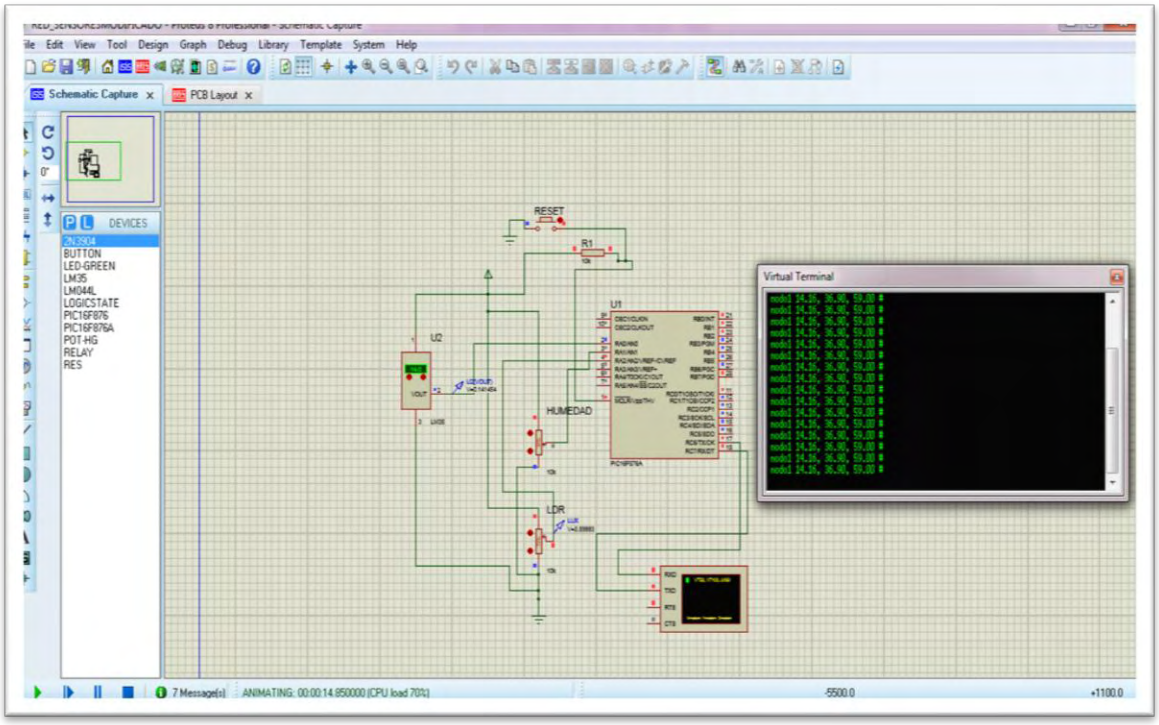


Figura 17. Simulación nodo sensor

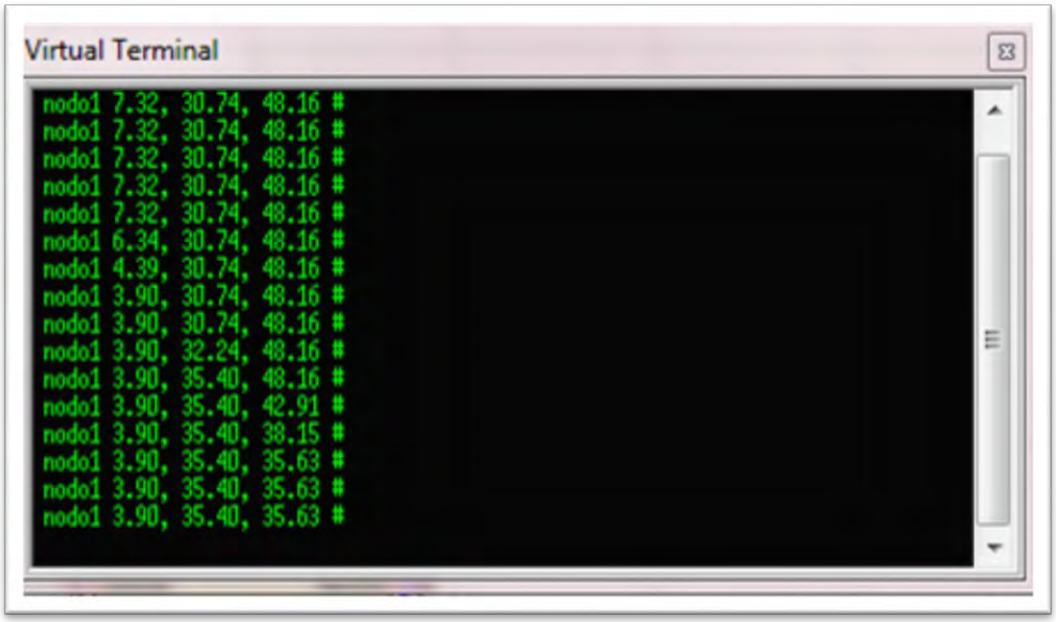


Figura 18. Visualización en monitor serial

En lo anterior se observa los datos para el nodo 1, el primer valor corresponde a la primera variable que es la temperatura en grados centigrados, el segundo a humedad en porcentaje y el tercero a luminosidad en lux.

Puesto que en la simulación funcionó correctamente lo que se programó en el pic, entonces con ayuda del programa *pikit2,* se cargó éste código a cada uno de los pics, configurando unicamente el número del nodo sensor ya que todos los nodos sensores van a medir las mismas variables fisicas pero en diferntes lugares.

2.2 IMPLEMENTACIÓN DE LA RED

Implementación de los nodos sensores: Se implementaron cada uno de los nodos de la red de sensores en protoboar teniendo en cuenta el datashed de cada uno de los elementos de la red.

Por otro lado ya que la mayoria de los componentes funcionan con 5v, a excepción del *xbee*, se alimentó todo el circuito con 5v y para el módulo *xbee* ya que funciona con 3.3v se utilizó bases las siguientes bases reguladoras de voltaje:

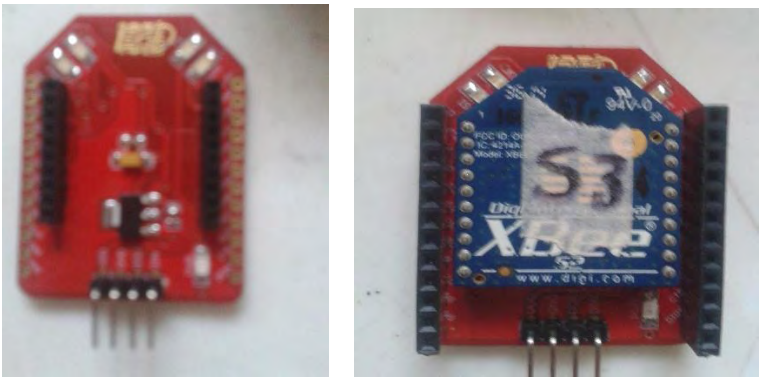
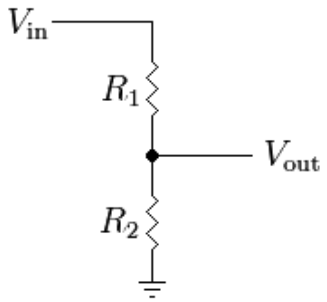


Figura 19. Tarjerta reguladora para xbee. Tomada de [42]

También para la comunicación serial entre el pic 16f876A, se realizó un divisor de voltaje para obtener un voltaje utilizando la siguiente formula para obtener un voltaje de salida de 3.3v esto con el fin de garantizar que el módulo xbee no se dañe.



$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

Donde R1= 1k, R2= 2K y Vin es 5v por lo tanto el voltaje de salida es de 3.3v.

Ahora bien, en la implementación, se conectó los sensores de manera directa a las entradas del conversor del pic. El sensor de temperatura a la entrada 0, el de humedad a la entrada 1 y el de luminosidad a la entrada 2. Los sensores de humedad y temperatura se conectaron de manera directa mientras que el sensor de luminosidad se conectó como se muestra acontinuacion:

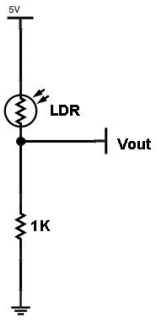


Figura 20. Conexión sensor de luminosidad. Tomada de [3]

Durante las pruebas el montaje en *protoboard*, se lo alimentó con una fuente de 5v pero teniendo en cuenta que no se encuentra disponibles en el mercado baterías de 5v, se decidió alimentar el circuito con una batería de 9v, la cual es comercial y para poderla utilizar realizó una regulación de voltaje con un regulador de 5v de referencia DH7805, el cual proporciona una salida de 5v para así energizar todo el circuito con 5v.

Durante las pruebas se realizó el proceso de calibración para el sensor de luminosidad LDR.

Ahora bien, las LDRS varían su resistencia en función de la luz que reciben. Entonces, para calibrar el sensor de luminosidad se realizó un montaje para hacer una prueba experimental, para esto se utilizó una lámpara de tungsteno la cual simula luz amarilla, alimentada con un voltaje de 17.2v, un elemento patrón denominado luxómetro de referencia 401025 [46] y la LDR a calibrar así:

Figura 21. Montaje para calibración de la LDR

El montaje anterior, se realizó con el fin de colocar el luxómetro y la LDR a calibrar y en las mismas condiciones. Este proceso consistió en apagar todas las fuentes de luz externas y solo realizar la experiencia con la luz de la lámpara de tungsteno. Así, teniendo listo el montaje se midió y tomó datos del voltaje de salida del divisor realizado en el circuito para la LDR, los valores de resistencia de la LDR y al mismo tiempo los valores en lux dados por el elemento patrón en las mismas condiciones haciendo variar la cantidad de luz emitida por la lámpara.

Figura 22. Toma de Datos

Se tomó varios datos de voltaje, luminosidad en lux y resistencia así:

Tabla 3. Datos Experimentales

V OUT (V)	LUXOMETRO (LUX)	R LDR (KΩ)
2,41	274	3
2,6	192	3,57
2,81	153	4,23
3,11	119	5,4
3,23	103	6
3,36	88	6,76
3,47	77	7,5
3,54	69	8
3,69	57	9,3
3,79	48	10,33
3,86	43	11,17
3,94	38	12,27
4,01	34	13,37
4,07	31	14,4
4,14	27	15,89
4,21	24	17,59
4,30	20	20,3
4,39	16	23,7
4,44	14	26,16
4,50	12	29,7
4,58	10	35,9
4,63	9	41,3

De acuerdo a las pruebas experimentales con los datos tomados se realizó una gráfica de la variación de la resistencia en función de la luminosidad en lux dada por el dispositivo patrón. Así:

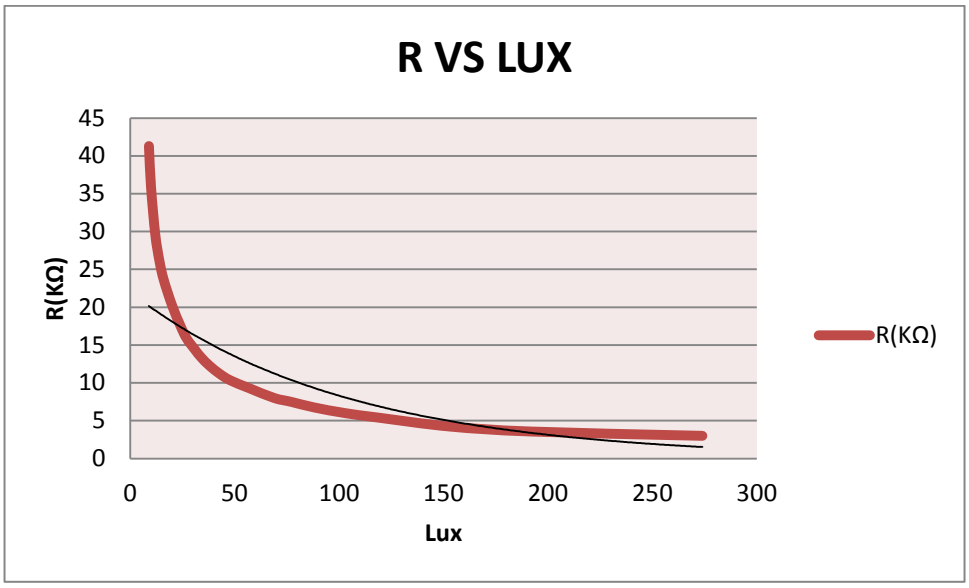


Figura 23. R vs lux

Según lo anterior se observó como la resistencia variaba, según la cantidad de luz emitida por la lámpara del mismo modo se realizó la comparación entre el voltaje de salida del divisor medido y la cantidad de luminosidad en lux dada por el luxómetro así:

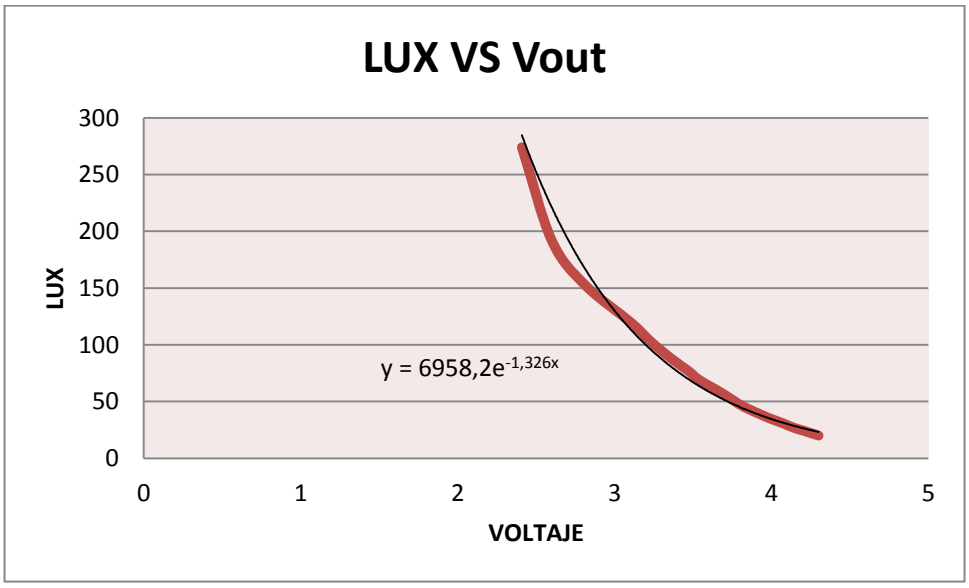


Figura 24. LUX vs Vout

Teniendo en cuenta los resultados de los datos anteriores, se decidió utilizar la fórmula $Y= 6958,2e^{-1.326x}$ para encontrar el valor en lux de las medidas dadas por la LDR, donde Y corresponde al valor en lux y x al valor del voltaje de salida (Vout) del divisor, de tal manera que los datos sean confiables ya que la LDR tiene un comportamiento no lineal. Con esto, se programó la fórmula en el código del PIC, y se probó que los datos obtenidos con la LDR se aproximan en un 98% a los datos obtenidos con el luxómetro en las mismas condiciones.

De acuerdo a este procedimiento se puede concluir que la LDR quedó calibrada y sus medidas son confiables.

Es importante resaltar que el luxómetro que se utilizó, está disponible en los laboratorios de física de la universidad de Nariño y para esto se partió de que este estaba calibrado correctamente, ya que cuando se mide la referencia corresponde a las especificaciones dadas por la hoja de calibración dada por el fabricante en donde dice que el error del instrumento es de + ó - 1 lux.

Para los sensores de humedad y temperatura no se realizó éste proceso, ya que se tomó en cuenta la información del *datasheed* de cada uno, en el que decía que éstos se encontraban pre calibrados.

El procedimiento anterior se realizó para los tres nodos sensores.

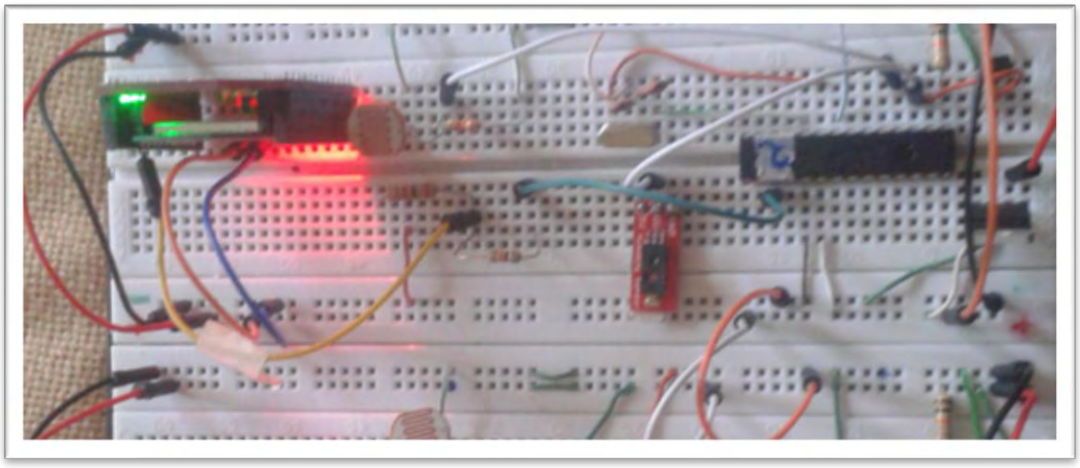


Figura 25 Montaje nodo sensor 1

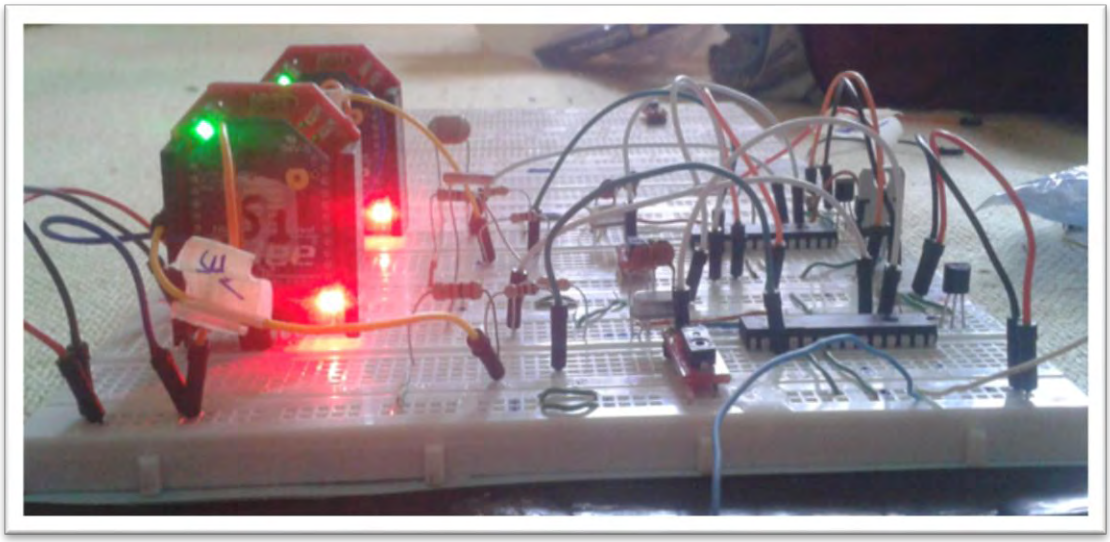


Figura 26. Montaje nodo sensor 1 y 2

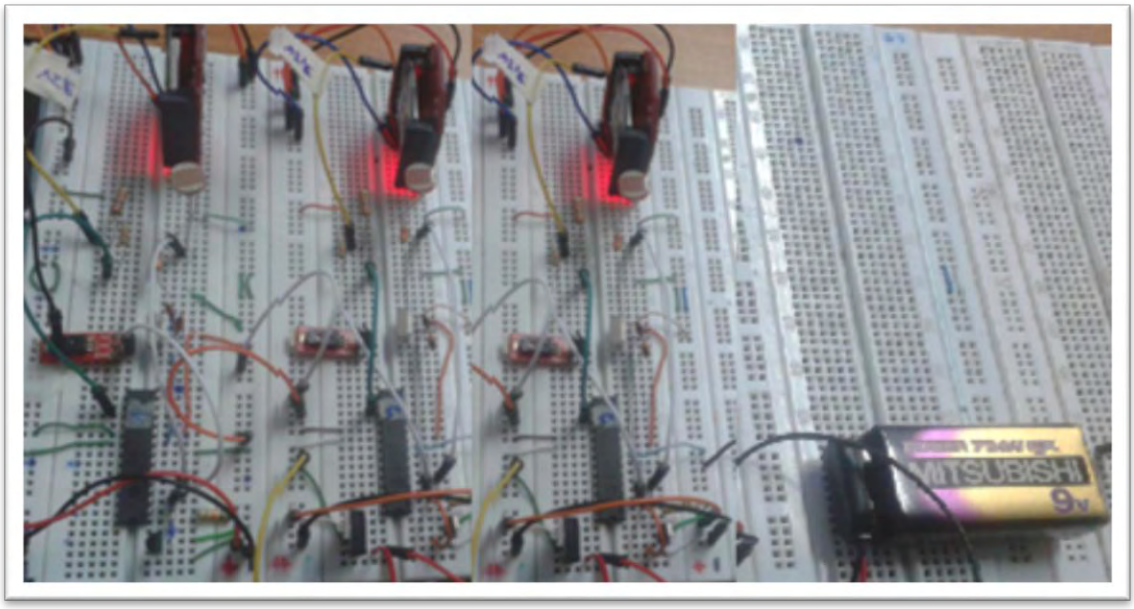


Figura 27. Montaje con 3 nodos sensores y batería de 9v y regulador de 5v

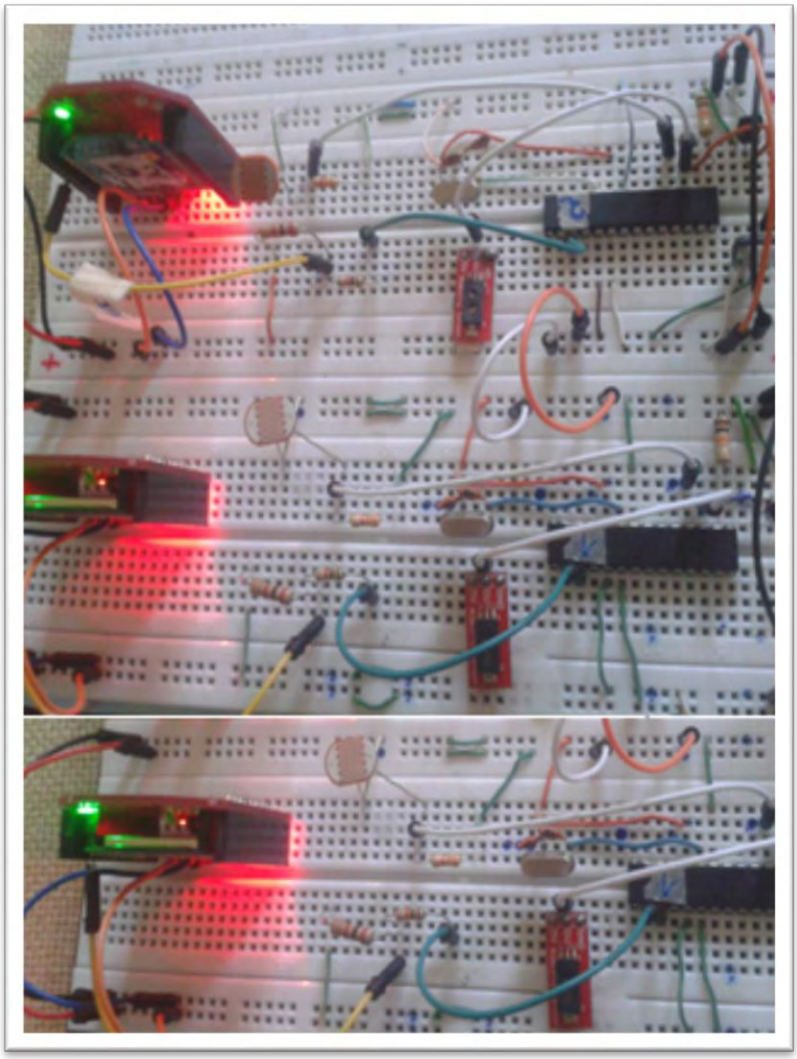


Figura 28. Montaje sensor 1 2 y 3 transmitiendo

En resumen, en la implementación anterior se observa como los sensores de temperatura LM35**,** de humedad HIH4030 y de luminosidad LDR envían la información hacia el PIC 16f876A [8], el cual se encarga de procesarla y enviarla por comunicación serial hacia el módulo *xbee* el cual de forma inalámbrica se encarga de transmitirla hacia el módulo xbee coordinador de la red.

DISEÑO DEL CIRCUITO IMPRESO:

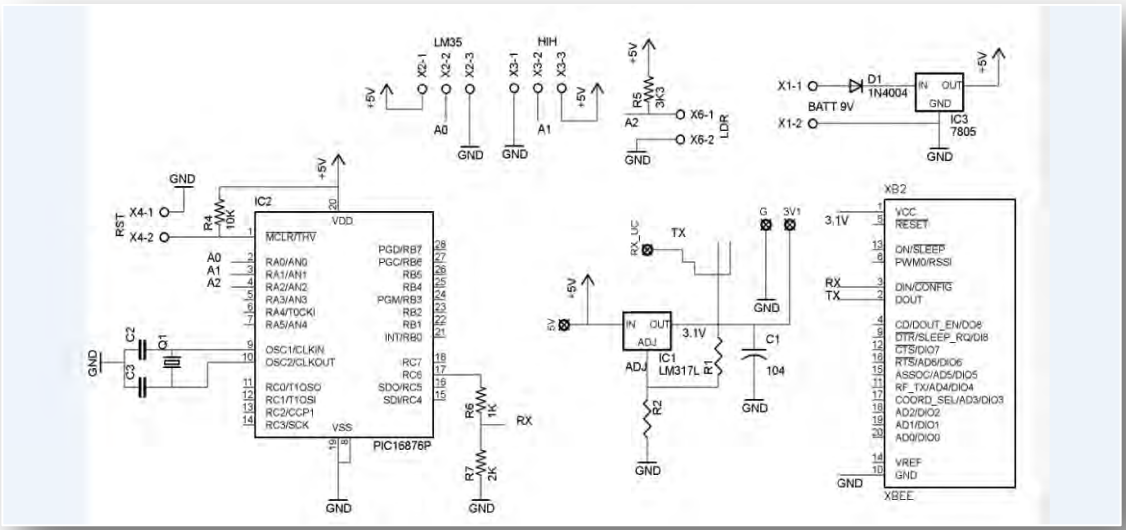


Figura 29. Diseño del nodo sensor

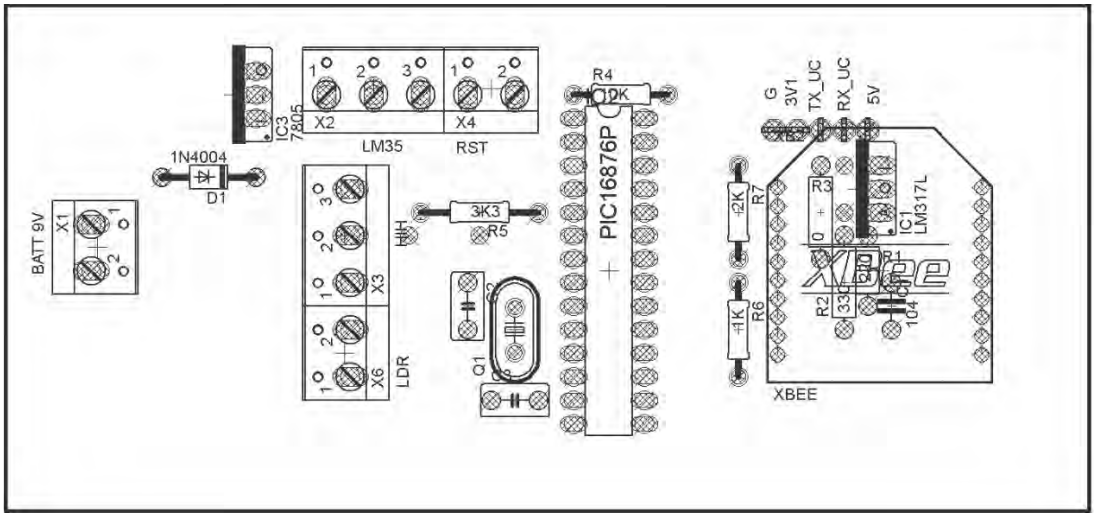


Figura 30. Diseño del circuito impreso del nodo sensor

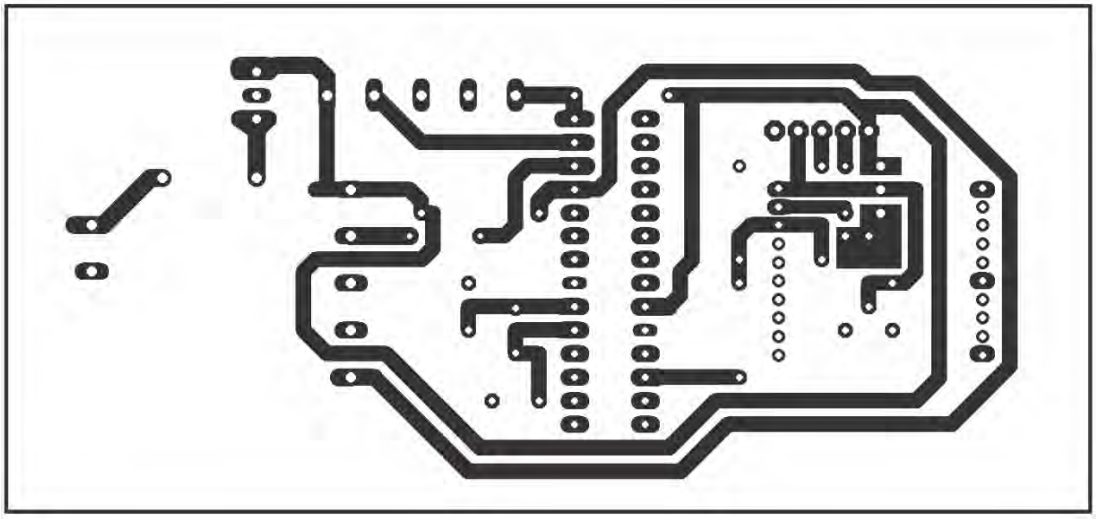


Figura 31. Diseño de las pistas del circuito impreso del nodo sensor

DISEÑO DEL NODO SENSOR TERMINADO:

Para el nodo 1 2 y 3 se realizó el procedimiento anterior y luego se lo implementó asi:

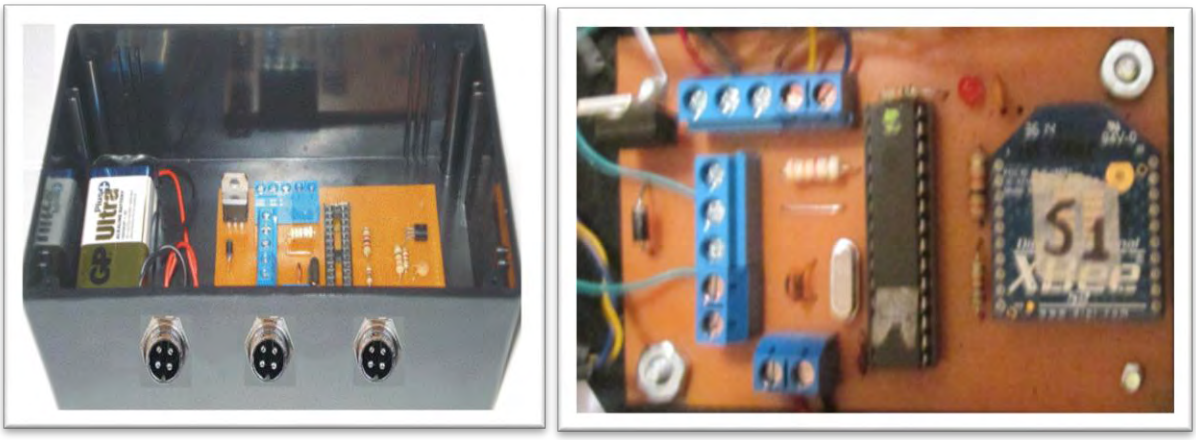


Figura 32. Diseño del circuito impreso terminado del nodo sensor.

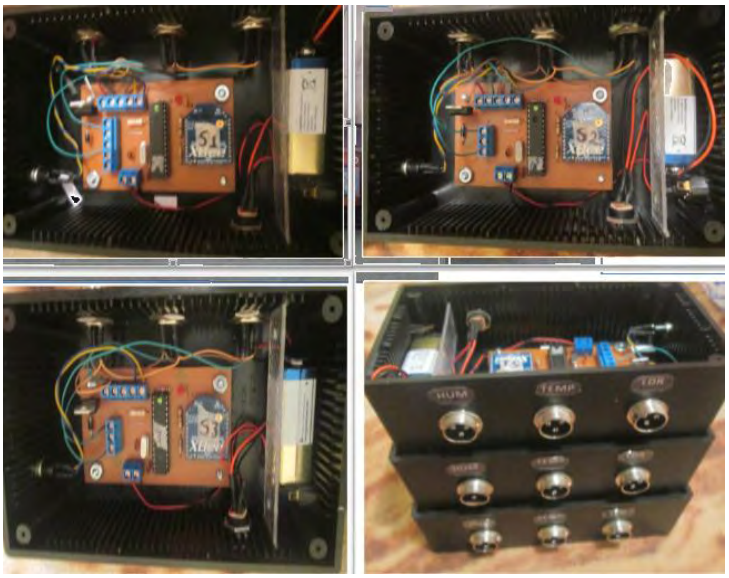


Figura 33. Implementación de los nodos 1, 2, y 3 de la red.

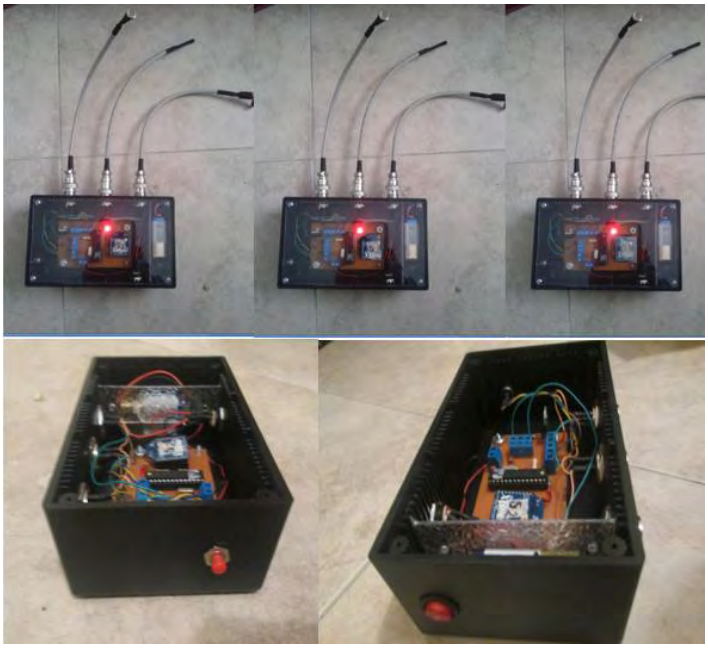


Figura 34. Implementación de los nodos 1, 2, y 3 funcionando con sus respectivos sensores y un botón de rest y otro de encendido y apagado .

3. DISEÑO E IMPLEMENTACIÓN DE SOFTWARE

DISEÑO DE SOFTWARE

Para el diseño de la aplicación móvil se tuvo en cuenta el enfoque didáctico, [18] [11] algunos conceptos de instrumentación electrónica y el programa de estudios [7] de la asignatura como tal, así entonces, se diseñó 3 actividades con interfaces para la aplicación móvil como: parte teorica, monitoreo en tiempo real, ejercicios, junto con el logo de la aplicación el cual es con la imagen del inventor, ingeniero y físico Nikola Tesla[47], por lo cual esta aplicación tiene por nombre NIKOLA.

- **Parte teorica:** en esta parte se encuentran los conceptos básicos de la asignatura de instrumentación electrónica.

- **Monitoreo en tiempo real**: en esta parte se realizó el monitoreo en tiempo real de temperatura, humedad y luminosidad , tambien la gráfica de cada variable física

- **Ejercicios:** en esta parte se realizó una simulación de algunas actividades didáacticas, enfocadas al aprendizaje de algunos conceptos de la asignatura de instrumentación electrónica.

Para la realización de la aplicación móvil para *Android* [10] se utilizó el *software* de *Microsoft Visual Studio* 2015, junto con el lenguaje *C#* (VER ANEXO3).

Se seleccionó la herramienta *Microsoft Visual Studio* 2015, porque permite el uso de múltiples plataformas para la construcción de *software*, su amplia biblioteca de clases y su fácil adaptación de tecnologías.

Se inició con el desarrollo de la aplicación móvil en *Visual Studio* haciendo uso de las bibliotecas *Xamarin,* para la construcción de compilados para sistema operativo *Android*, además se empleó la metodología de desarrollo de *software* XP [17] por su flexibilidad y libertad de interacción entre el constructor y los *stakeholders*.

IMPLEMENTACIÓN DE SOFTWARE

Se generó el APK *(Application Package File)* de la aplicación móvil y se la instaló en varios dispositivos móviles Así:

- **LOGO DE LA APP:**

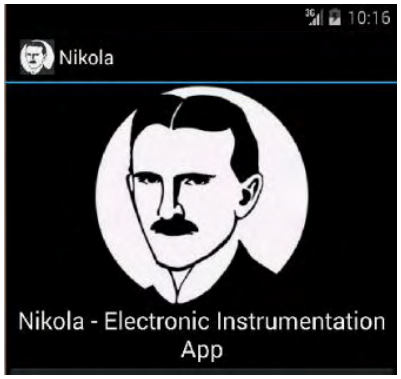


Figura 35. Logo de la aplicación

- **APLICACIÓN INSTALADA EN DISPOSITIVOS MÓVILES**

Figura 36. Aplicación instalada en dispositivos móviles

- **PRESENTACION DE LA APLICACIÓN NIKOLA**

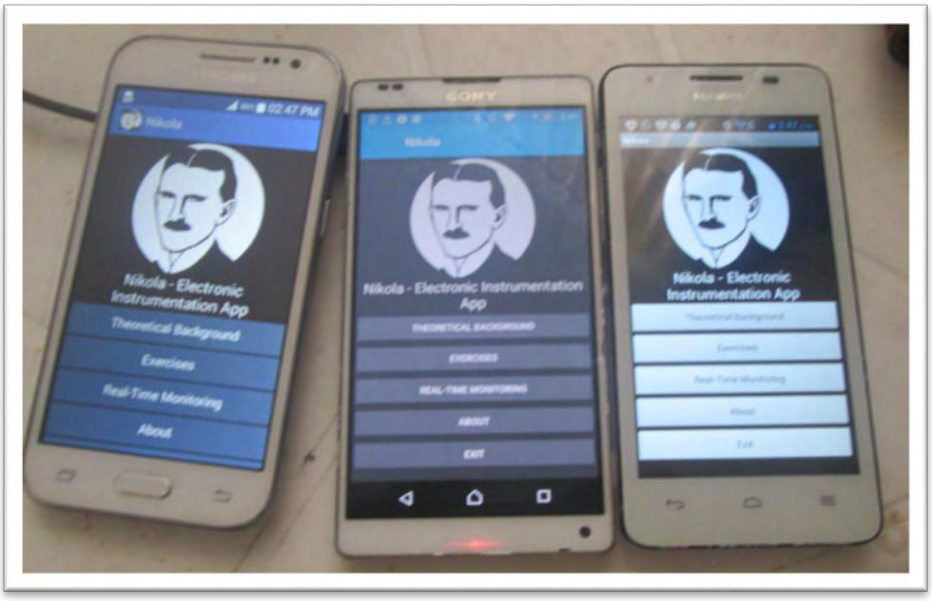


Figura 37. Presentación de la aplicación Nikola

- **PRIMERA PARTE: THEORETICAL BACKGROUND**

Figura 38. Ventana de la primera parte Theoretical Background.

En la figura anterior se observa la primera parte de la aplicación movil, la cual contiene los conceptos básicos de Instrumentación Electrónica y las caracteristicas de los sensores utilizados para realizar el monitoreo.

- **SEGUNDA PARTE: EXERCISES**

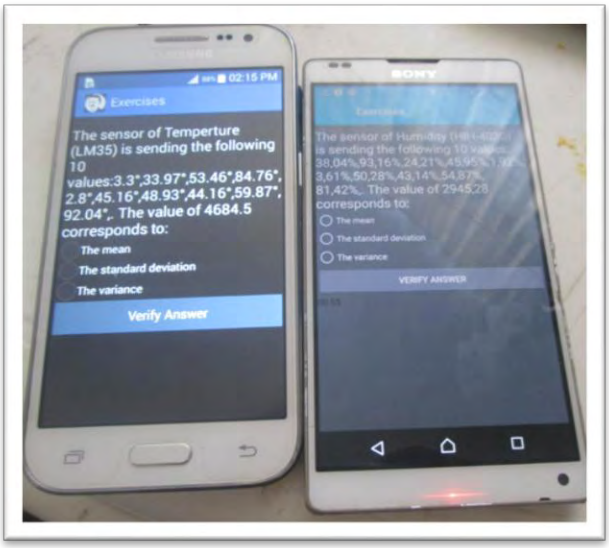


Figura 39. Ventana de la segunda parte Excercises

En la figura anterior se obsserva la segunda parte de la aplicación movil, la cual corresponde a la parte de simulación de ejercicios, como actividad didactica como apoyo al aprendizaje en algunos conceptos de instrumentación electrónica para el análisis estadístico como el cálculo de la media, la desviación estandar y la varianza teniendo en cuenta una muestra de 10 datos.

- **TERCERA PARTE REAL TIME MONITORING**

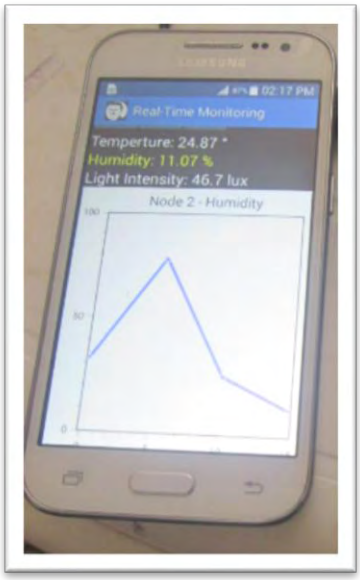


Figura 40. Ventana de la tercera parte Real Time Monitoring

En la figura anterior se obseva el monitoreo en tiempo real de los valores arrojados por la red de sensores, mediante la conexión del *hadware* y *software* terminados. También se puede observar la gráfica de los valores dependiendo de la variable que se requiera graficar ya sea temperatura, humedad o luminosidad. También la aplicación contiene lo siguiente:

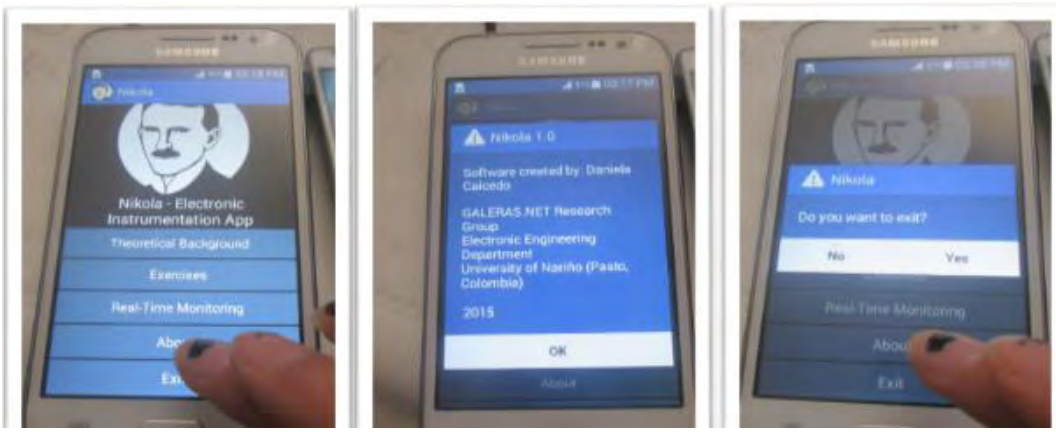


Figura 41. Otras opciones de la aplicación Movil

En la gráfica anterior se muestran mas opciones de la aplicación móvil como los derechos de autor y la opcion de salir.

4. DISEÑO DE UN SISTEMA DE COMUNICACIÓN INALAMBRICA PARA INTERCONECTAR HADWARE Y SOFTWARE

Teniendo lista la red de sensores, se diseñó un sistema que permita la comunicación del *hadware* y *software* por medio de la tecnología de comunicación WiFi , para ésto se estudiaron componentes electrónicos acoplados de 4 formas:

- La red de sensores con el módulo *xbee WiFi S6B* de *digi* ,para ser utilizado como coordinador de la red, realizando una conexión entre la red y los dispositivos móviles ya que maneja el estándar de comunicación 802.11 de manera que recoja los datos y puedan ser transmitidos.

- La red de sensores se comunica con una tarjeta *shield Wireless SD*, sobre la cual está conectado un módulo *wifly RN-XV.* Esto con el fin de recoger datos de la red por medio de *arduino* y comunicar vía *wiFi*.

- El módulo *xbee* coordinador de la red se conecta sobre una tarjeta *xbee shield* [35] y a su vez sobre una tarjera arduino Leonardo [16] junto con el módulo WiFi ESP8266 [31] , permitiendo recoger los datos de la red y que a su vez sean transmitidos vía WiFi

- El módulo *xbee* coordinador de la red se conecta sobre una tarjeta *xbee shield*, la cual a su vez esta sobre una tarjeta arduino *WiFi,* lo cual permite recoger los datos de la red para que pueda ser transmitida vía *wiFi* hacia varios dispositivos.

- De las anteriores opciones se seleccionó la opción 3, es decir el módulo *xbee* coordinador de la red se conectó sobre una tarjeta *xbee shield* [35] y a su vez sobre una tarjera arduino Leonardo [16] junto con el módulo WiFi ESP8266 [31] de tal manera que se recepcionó los datos de la red y se los transmitió vía WiFi. Se realizó esta selección por la facilidad de conseguir los componentes en el mercado y a precio accesible.

Entonces el diseño quedó de la siguiente manera:

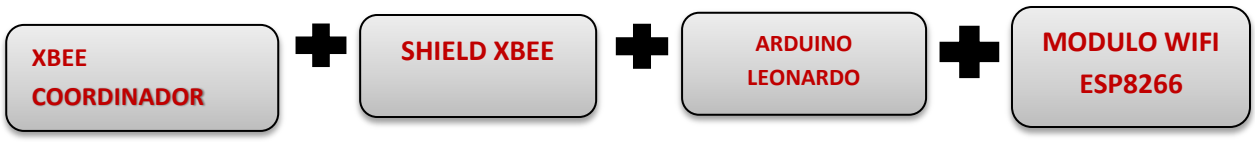


Figura 42. Diagrama de bloques del sistema de comunicación inalámbrica

Para lo anterior, se realizó un programa (ver anexo2) en el *software* de *arduino* para leer los datos del nodo coordinador, para esto se utilizó la placa Arduino Leonardo[16], debido a que posee un microcontrolador ATmega32U4 que permite un diseño mucho más sencillo y económico. Además dispone de USB nativo por hardware y por lo tanto no necesita de ninguna conversión serie-USB. También permite realizar muchas funciones dentro de el para ser utilizada y programada dependiendo de la aplicación que requiera el usuario.

Esta placa tiene 20 pines digitales de entrada/salida, de los cuales 7 pueden ser usadas como salidas PWM y 12 como entradas analógicas.

Es importante resaltar que el puerto de comunicación USB es emulado, por tanto, deja el puerto serial *hardware* libre para la programación; De esta forma ya no ocurren conflictos de programación mientras tenemos periféricos seriales conectados a la placa.

Figura 43. Tarjeta Arduino Leonardo. Tomado de [16]

Luego de que se recepcionó la información de los nodos sensores en el monitor serial de arduino, se realizó la comunicación *WiFi* y se recibió los datos via *WiFi* en dispositivos moviles y se los capturó mediante la aplicación móvil creada, esto se logró gracias a que se utilizó el módulo *WiFi* ESP8266 [31] y a que se realizó un servidor web en *software* de arduino.

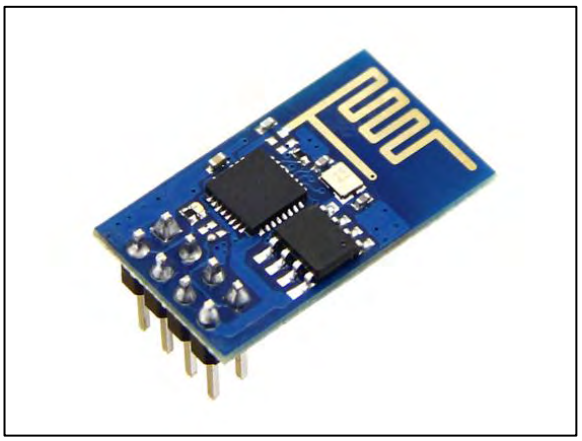


Figura 44. Modulo wifi ESP8266. Tomado de [31]

El módulo ESP8266 permite conectarse via WiFi mediante el protocolo 802.11 b/g/n, mediante la utilización de comandos AT, además funciona a 3.3v y tiene un procesador integrado de 32bits y soporta WPA /WPA2.

Entonces para lograr que los dispositivos se comuniquen de forma inalambrica por medio de WiFi, se diseñó en el *software* de arduino un programa (ver anexo 2), que envía los datos hacia la web, conectandose a un AP (*Accees Point*), de tal manera que la aplicación móvil pueda recepcionar las cadenas de datos y posteriormente extraer los datos de cada una de las variables a monitorear y poder realizar las actividades presentes en la aplicación. Esto gracias a la utilización de comandos AT y de un servidor web montado sobre el arduino.

Además como el arduino funciona a 5v y el módulo *wiFi* a 3.3v se utilizó el conversor SPRK-BOB-12009 [45]; El cual permite bajar la tensión de 5v, dada por el arduino a 3.3v, que es lo que soporta el módulo *wiFi.* y dispone de 2 canales de tensión alta 5v y baja 3.3v.

Figura 45. Conversor lógico.Tomada de [45]

De esta manera se diseñó el sistema de comunicación inalámbrica basado en tecnología wiFi así: El módulo xbee configurado como coordinador AT junto con

una tarjeta Arduino Leonardo, para el procesamiento de datos y con el módulo *WiFi* ESP8266 para enviar la información vía *WiFi.* Es importante resaltar que para la comunicación con el módulo *xbee* desde el arduino, se inicializó el puerto a 9600 baudios y para la comunicación *WiFi* con el módulo *WiFi* esp8266 se lo inicializó a 115200, esto con el fin de evitar que se bloquee el *firmware* de los componentes.

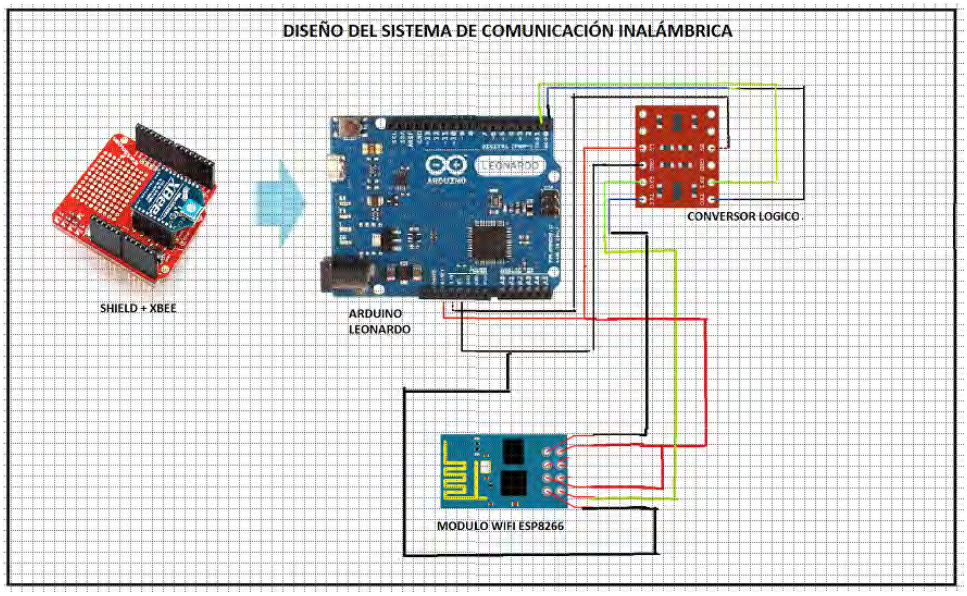


Figura 46. Diseño del sistema de Comunicación inalámbrica

Luego se implementó el sistema de comunicación inalámbrica vía WiFi, de tal forma que todos los componentes del circuito se ajustaron adecuadamente reduciendo espacio.

Figura 47. Implementación del Sistema de comunicación inalámbrica *wiFi*

Luego se subió el programa a la placa arduino y en el monitor serial se obtiene la siguiente información :

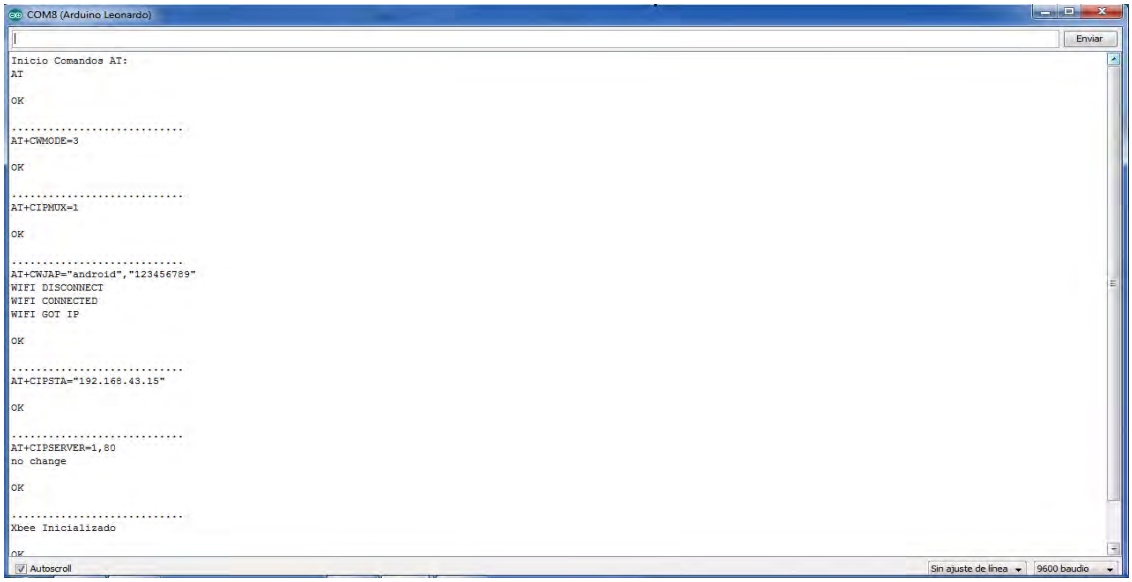


Figura 48. Comunicación WiFi con el modulo ESP8266 mediante comandos AT

En la grafica anterior se observa como el módulo WiFi se conecta a un access point externo mediante comandos AT y esta listo para recibir los datos provenientes del nodo cordinador, el cual es el encargado de recepcionar todos los datos provenientes de los nodo sensores.

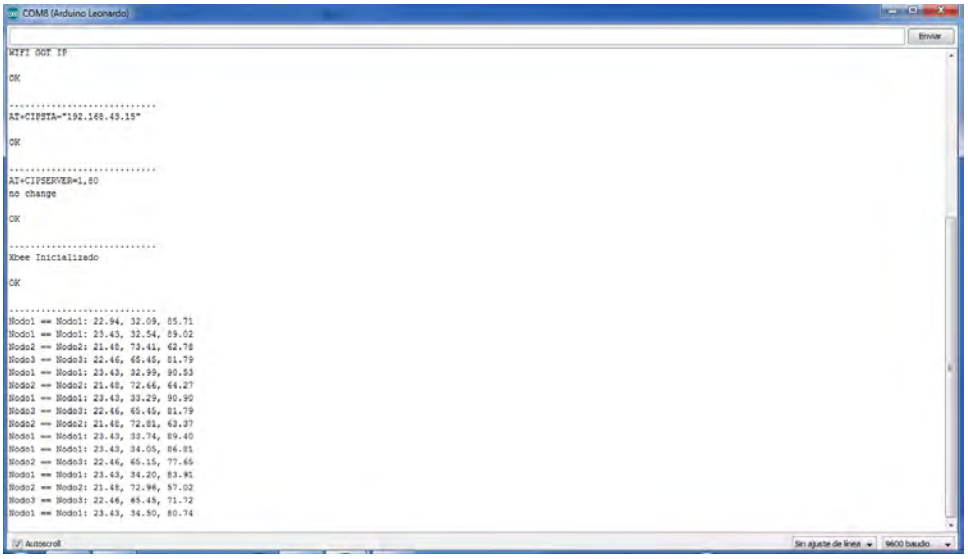


Figura 49 . Recepción de datos de los nodos sensores en el monitor serial de arduino

En la figura anterior se observa que después de conectarse el modulo *WiFi* a un acces point externo, se inicializa el puerto 80 para permitir la comunicación por medio de un servidor web y luego se inicializa el módulo xbee coordinador para la recibir todos los datos provenientes de los nodos sensores.

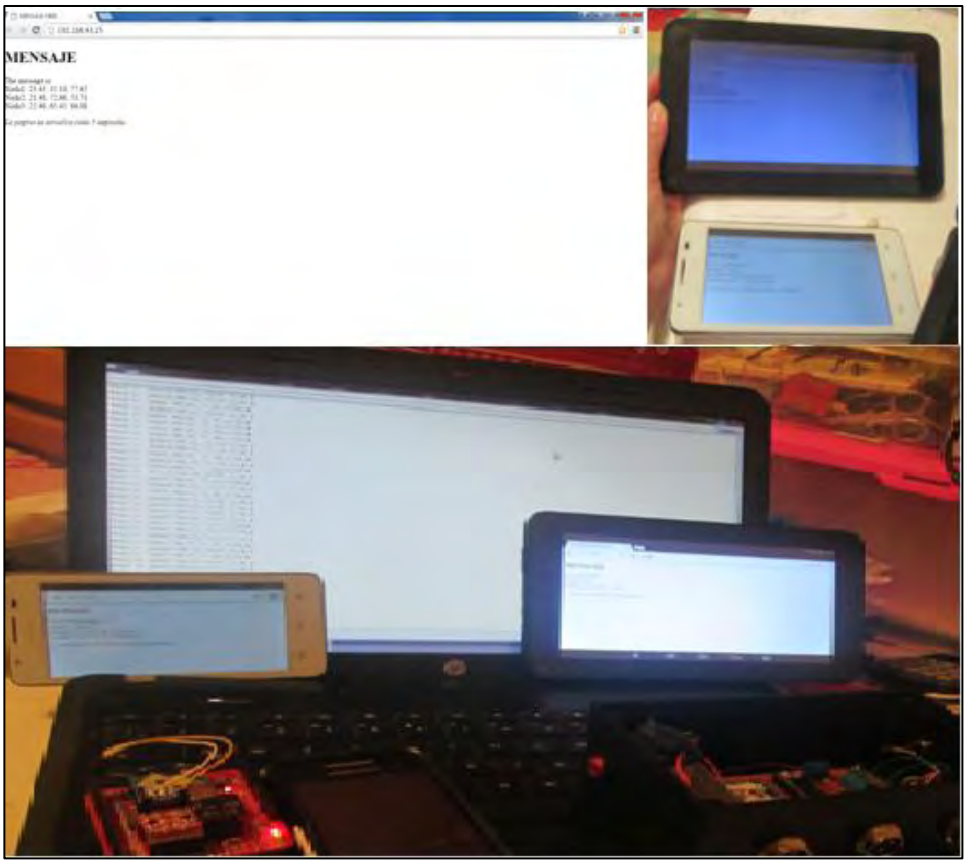


Figura 50. Datos de los nodos 1 2 y 3 en la web

En la grafica anterior se observa el funcionamiento de el servidor web mediante el protocolo http (cliente servidor), es decir se manda la solicitud desde un computador o dispositivos móviles a la web via WiFi y éste responde con la información que se encuentra en el servidor web creado en el *software* arduino(ver anexo 2), perimitiendo observar los datos de los nodos sensores, donde el primer valor corresponde a la temperatura en grados centígrados, el segundo a la humedad en porcentaje y el tercero a la luminosidad en lux; enviados vía WiFi y así se los visualizó en la web desde el computador y desde los dispositivos móviles y luego fueron capturados por la aplicación móvil para realizar el monitoreo en tiempo real.

Finalmente, se conectó todos los componentes de la herramienta permitiendo asi, monitorear los datos y ejecutar todas las actividades creadas en la aplicación móvil.

5. INTERACCIÓN DE LA HERRAMIENTA CON ESTUDIANTES DE SEXTO SEMESTRE

En el semestre B de 2015, se realizó la presentación de la Herramienta Didáctica para monitoreo de variables físicas a los estudiantes de sexto semestre del programa de Ingeniería Electrónica, en la asignatura de Instrumentación gracias a la colaboración del ingeniero Rolando Barahona Cabrera, profesor de ésta asignatura el cual brindó un espacio para la realización de esta experiencia.

Para esto, en primer lugar se presentó a los estudiantes el desarrollo de éste trabajo, luego se colocó en funcionamiento la herramienta didáctica, los estudiantes interactuaron con ella y finalmente respondieron una encuesta para evaluar el impacto de la herramienta como apoyo al proceso de aprendizaje, en lo que se refiere a la asignatura de Instrumentación Electrónica.

Figura 51. Presentación de la herramienta a los estudiantes de sexto semestre del programa de Ingenieria Electrónica

Figura 52. Exposición de el desarrollo y funcionamiento de la herramienta.

Figura 53. Interacción de los estudiantes con la herramienta

Figura 54. Monitoreo e Interacción de los estudiantes con la herramienta

Figura 55. Interacción de los estudiantes con la herramienta

En las figuras anteriores se puede observar como los estudiantes manipularon los componentes de la herramienta didáctica y mediante la aplicación móvil realizaron no solo el monitoreo de temperatura, humedad y luminosidad sino las actividades presentes en la misma enfocadas a la asignatura de instrumentación electrónica sirviendo así como herramienta de apoyo al proceso de aprendizaje de esta

asignatura. Después de la experiencia realizada los estudiantes respondieron una encuesta.

Figura 56. Aplicación de una encuesta a los estudiantes de sexto semestre del programa de Ingeniería Electrónica

Figura 57. Finalización de la experiencia realizada.

Al finalizar ésta experiencia, se agradeció tanto a los estudiantes como al Ingeniero Rolando Barahona profesor de la asignatura, por la colaboración y el espacio prestado para la realización de la misma.

6. ANÁLISIS DEL IMPACTO DE LA HERRAMIENTA EN LOS ESTUDIANTES DE SEXTO SEMESTRE DEL PROGRAMA DE INGENIERÍA ELECTRÓNICA

Para evaluar el impacto de la herramienta como apoyo en el proceso de aprendizaje en la materia de instrumentación Electrónica, se presentó a los estudiantes de sexto semestre del programa de Ingeniería Electrónica, el desarrollo y funcionamiento de la herramienta para que interactúen con ella y se les aplicó una encuesta que arrojaron los siguientes resultados.

El número de estudiantes que matricularon la materia de instrumentación electrónica eran 37, pero el día de la realización de la práctica solo se contó con el número de 33 estudiantes, de los cuales según su género, 8 corresponden al género femenino y 25 al género masculino lo cual en porcentaje corresponde a:

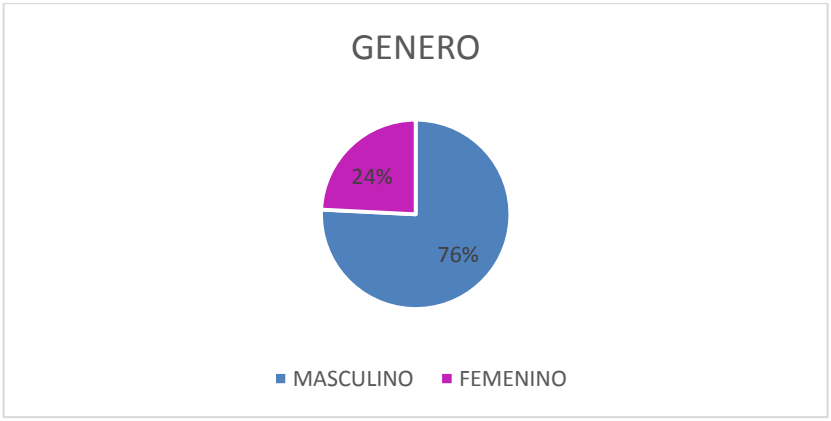


Figura 58. Porcentaje de estudiantes según el género

Ahora bien, analizando los resultados obtenidos de la encuesta realizada se obtuvo:

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta número 1 que se refieren a si con la experiencia realizada, el estudiante afianzó los conceptos básicos de instrumentación electrónica.

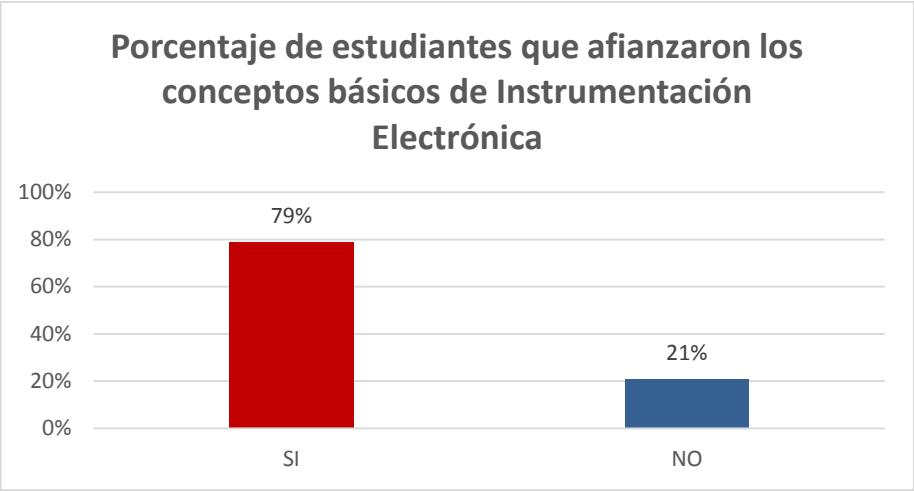


Figura 59. Resultados de las respuestas a la pregunta N° 1

Se observa que la mayoría de los estudiantes respondieron que **SI** afianzaron los conceptos básicos de la materia de Instrumentación Electrónica y una minoría **NO** afianzó los conceptos básicos.

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta número 2 que se refieren a qué tan interesante encontró el estudiante esta herramienta.

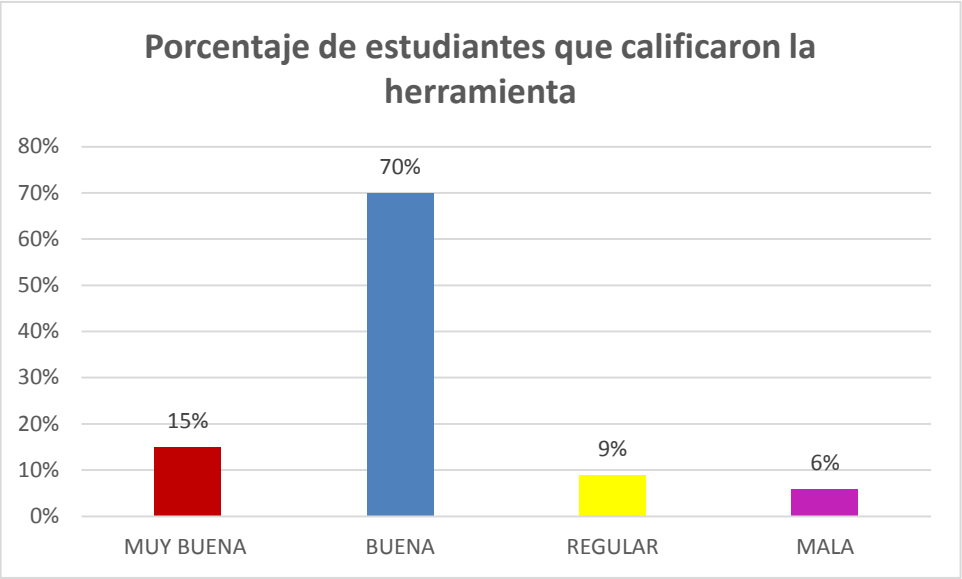


Figura 60. Resultados de las respuestas a la pregunta N° 2

En la gráfica anterior se observa que los estudiantes calificaron la herramienta en primer lugar como buena, en segundo lugar como muy buena, en tercer lugar como regular y en último lugar como mala.

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta número 3 que se refieren a cuál de las expresiones describió mejor la experiencia de los estudiantes con la herramienta:

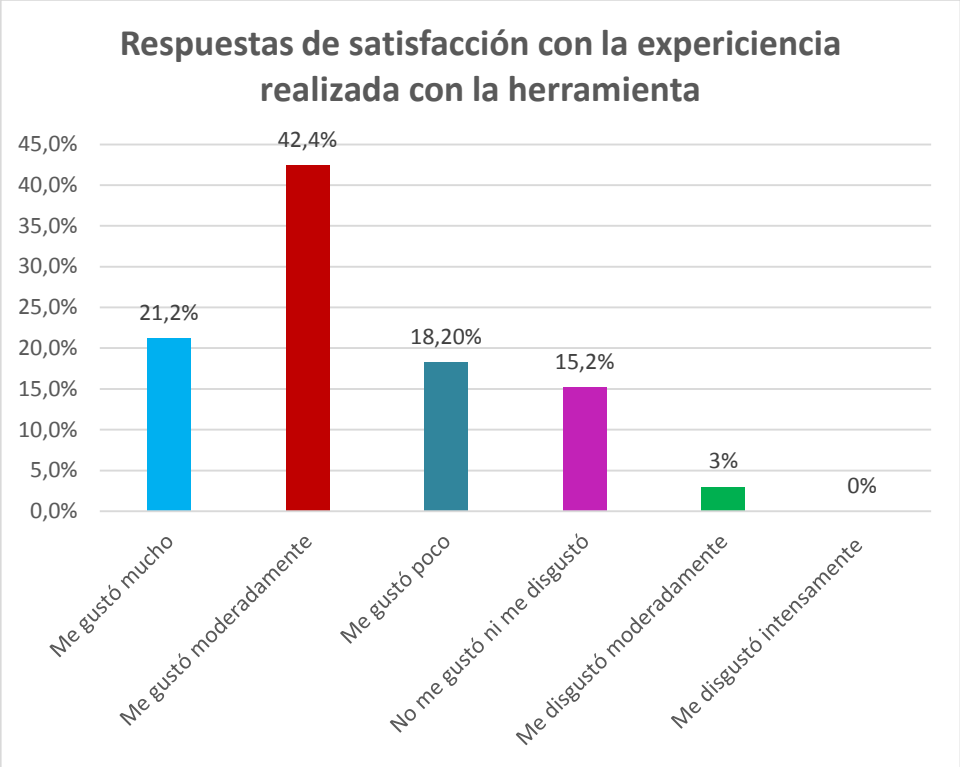


Figura 61. Resultados de las respuestas a la pregunta N° 3

De acuerdo con la información presentada en la gráfica anterior se observa que a la mayoría de los estudiantes les gustó moderadamente, luego con un porcentaje menor les gustó poco, siguiendo con un porcentaje mucho menor de estudiantes que respondieron que les gustó ni les disgustó, luego con un porcentaje de estudiantes que respondieron que les gustó mucho, luego con una minoría de estudiantes que respondieron que les disgustó moderadamente y finalmente con un 0% de estudiantes que marcaron la opción que les disgustó intensamente.

Para la pregunta No.4 que se refiere a lo que los estudiantes aprendieron con la práctica realizada esto fue lo que respondieron en la mayoría de las encuestas:

- Combinando los conocimientos de Ingeniería con los elementos adecuados se

puede hacer buenos proyectos de investigación.

- Funcionamiento de los diferentes sensores y módulos de comunicación inalámbrica y adquisición de datos.
- Control de 3 etapas en un solo sistema
- Buena manera de socializar los datos recopilados mediante un sensor.
- Nuevas herramientas de comunicación y sensores.
- La utilización de los módulos inalámbricos dan muchas posibilidades para diferentes aplicaciones.
- Funcionamiento de tres factores que contiene la herramienta.
- Manipulación de sensores para adquirir variables.
- Importancia de la aplicación móvil en proyectos didácticos en el campo de la Instrumentación.
- Interacción con el mundo real.
- Ideas para realizar nuevos proyectos con comunicación inalámbrica.
- Importancia de la aplicación para sistemas de monitoreo.

Además de esto, se encontró 9 encuestas sin responder a ésta pregunta y 4 encuestas de estudiantes de las cuales 2 con respuesta ´´no aprendí nada´´ y 2 con respuesta ´´aprendí poco´´.

Con respecto a la pregunta No.5 que se refiere a los cambios o mejoras que según la opinión de los estudiantes, debería incorporar nuestra herramienta se encontraron las siguientes respuestas:

- Incorporar más sensores y etapas de actuadores.
- Una forma de manejo de sensores más sencilla.
- Opción en la app móvil para cambiar el lenguaje.
- Descripción de todo el sistema dentro de la aplicación móvil.
- Modificación de las cajas.
- Utilizar otro tipo de sensores.
- Tiempo de actualización de los datos más rápido.

Además se encontró para esta pregunta 6 encuestas sin responder.

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta No. 6, que se refiere a cuántas herramientas didácticas similares para prácticas de laboratorio los estudiantes han utilizado.

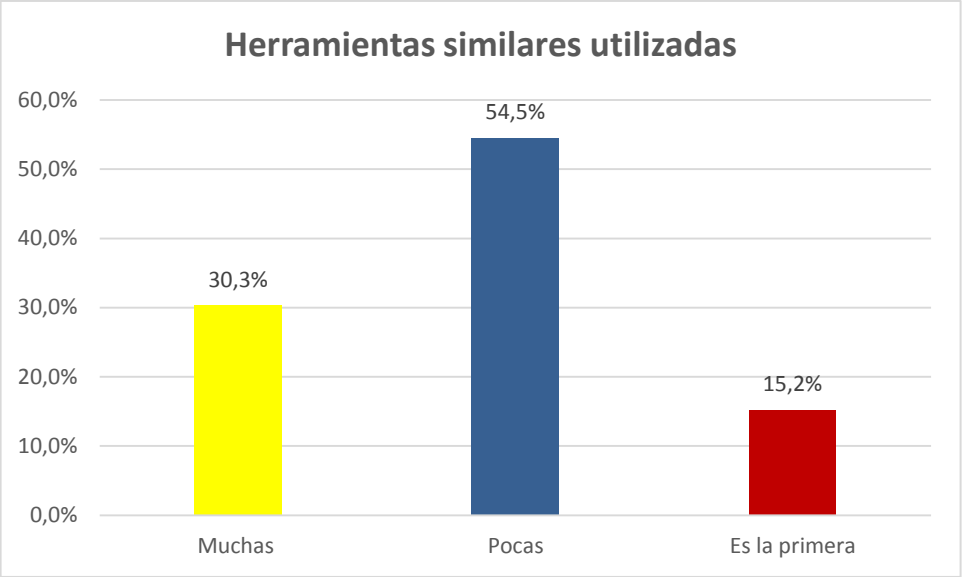


Figura 62. Resultados de las respuestas a la pregunta N° 6

En la gráfica anterior se observa que la mayoría de los estudiantes han utilizado pocas herramientas, siguiendo con un porcentaje de estudiantes que han utilizado muchas herramientas similares y finalmente con un porcentaje menor que respondieron que es la primera herramienta que han utilizado.

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta No.7 que se refiere a la pregunta ¿es más fácil y motivador aprender mediante la utilización de herramientas didácticas, como con la herramienta utilizada? Valorando de 1 a 5 1, en donde la calificación de 5 representa muy fácil y muy motivador, 4 moderadamente fácil y motivador 3, medianamente fácil y motivador, 2 un poco fácil y motivador y la calificación de 1, nada fácil y nada motivador.

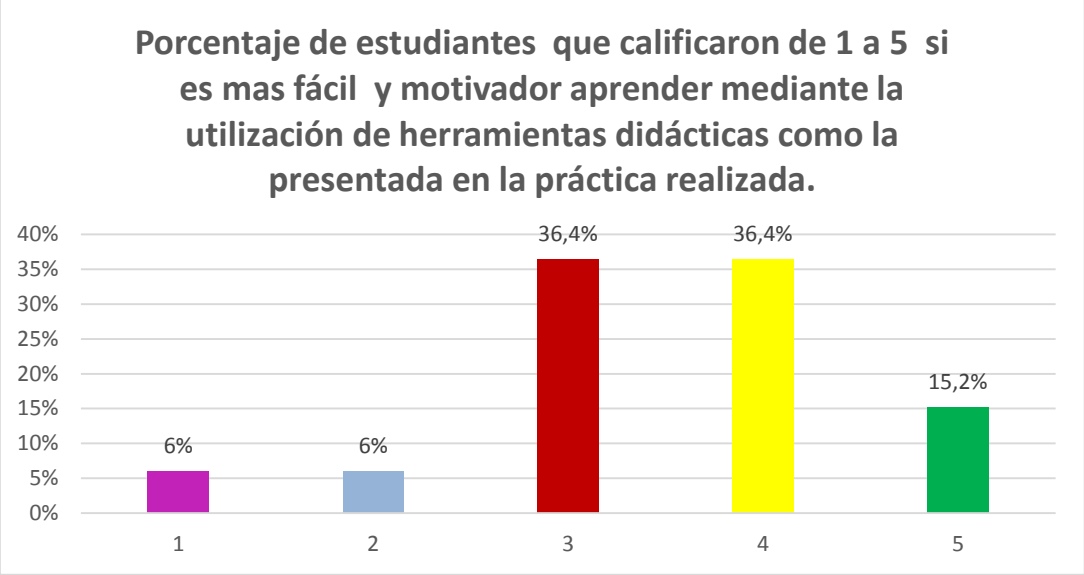


Figura 63. Resultados de las respuestas a la pregunta N° 7

En la gráfica anterior se observa que para la mayoría de estudiantes brindó una calificación de 3 o más, de acuerdo con esto se puede concluir que, la respuesta es positiva ya que se puede inferir que para estos estudiantes se hace más fácil y motivador aprender mediante la utilización de herramientas didácticas como la que se presentó en la práctica realizada. Por otro lado, una minoría dieron la calificación de 1 y 2, entonces para esta minoría se hace menos fácil y motivador aprender mediante la utilización de herramientas didácticas como la que se presentó en la práctica realizada.

La siguiente gráfica muestra las respuestas de los estudiantes a la pregunta número 8 que se refiere a que si cada estudiante consideró importante el uso de diferentes estrategias didácticas para que los estudiantes sean en el aula más creativos y participativos.

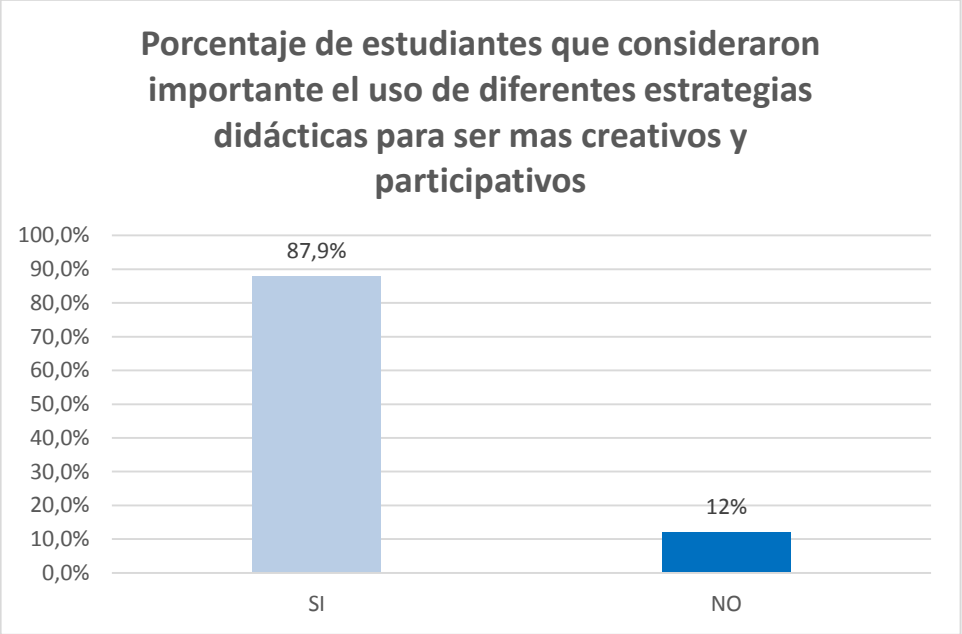


Figura 64. Resultados de las respuestas a la pregunta N° 8

En la gráfica anterior se observa que la mayoría de estudiantes. **SI** consideraron importante el uso de diferentes estrategias didácticas para que un estudiante sea en el aula más creativo y participativo y un porcentaje muy bajo de estudiantes consideraron que **NO** es importante el uso de las mismas.

7. TRABAJOS FUTUROS

Como líneas futuras de trabajo, se proponen los siguientes puntos:

- Agregar más sensores y nodos para monitorear no solo tres variables físicas y con esto realizar con monitoreo más extenso.

- Con base en el diseño de la aplicación móvil, diseñar aplicaciones que sirvan de apoyo no solo a la asignatura de instrumentación electrónica sino para otras áreas como control, comunicaciones entre otras.

- Realizar proyectos de investigación con enfoques didácticos

- Desarrollar aplicaciones móviles en otro lenguaje y con otros sistemas operativos.

8. CONCLUSIONES

La construcción de redes de sensores son muy utilizadas en varios campos para diferentes aplicaciones, ya que permiten el monitoreo de muchas variables existentes y la implementación de un gran número de nodos facilitando así el control de las mismas.

Los sistemas de comunicación inalámbrica con módulos XBEE aportan una solución efectiva para muchas aplicaciones en las que se requiere comunicación a largo alcance. Esto permite las investigaciones e inversiones en nuevas tecnologías que mejoren y faciliten la transmisión de datos

Los proyectos de investigación con enfoque didáctico motivan y sirven de apoyo al aprendizaje en las diferentes áreas, fomentando así la investigación y aplicaciones en un entorno real. Cabe destacar que este proyecto tuvo un enfoque didáctico en torno a la asignatura de instrumentación electrónica.

La integración de varios sistemas en un solo proyecto permite acoplar varios componentes que en conjunto permiten realizar varias actividades en un solo sistema.

Android es un sistema operativo muy avanzado y muy útil, que permite el desarrollo de aplicaciones en una plataforma accesible y proporciona todas las interfaces necesarias, permitiendo el acceso a las funciones del teléfono de una forma muy sencilla en un lenguaje de programación muy completo.

9. RECOMENDACIONES

Se recomienda el uso de sensores digitales para facilitar el diseño e implementación.

Se recomienda el uso de módulos XBEE con mayor alcance para aplicaciones de monitoreo de datos más complejas.

Se recomienda manipular y configurar adecuadamente los módulos XBEE, para evitar que se bloque su firmware.

Se recomienda utilizar en lugar microprocesadores como los PIC, Arduinos nano para reducir el tamaño y realizar más funciones dentro del nodo.

Se recomienda calibrar los sensores en caso de que no vengan pre calibrados de fábrica.

Se recomienda para las asignaturas en general profundizar el conocimiento sobre XBEE, protocolos de comunicación, tecnologías de comunicación inalámbricas, y más niveles de programación.

Se recomienda utilizar con la velocidad de transmisión de datos adecuada en el puerto serial, al momento de trabajar con arduinos y utilizar módulos como *xbee* y *WiFi* ESP8266 ya que si no se utiliza la adecuada se pueden bloquear.

Se recomienda profundizar acerca del sistema operativo *Android* ya que es un sistema operativo nuevo.

REFERENCIAS BIBLIOGRÁFICAS

[1] Honeywell. “HIH_4030/31 series Humidity Sensors”. USA: s.n., 2008. [en línea] [citado 2015-03-01] Disponible en internet: www.honeywell.com/sensing

[2] Texas Instruments. LM35 Precision Centigrade Temperature Sensors. Dallas, Texas: s.n., 2014. [en línea] [citado 2015-03-01] Disponible en internet: www.ti.com

[3] Sunrom Technologies. Ligth Dependent Resistors. India: LDR, 2008. [en línea] [citado 2015-03-01] Disponible en internet: http://www.sunrom.com/p-510.html

[4] Salcedo, E. Tecnologias inalámbricas. 2013. [en línea] [citado 2015-03-01] Disponible en internet: http://bibing.us.es/proyectos/abreproy/70218/descargar_fichero/2. Tecnolog% c3%ADas+InAlambricas.pdf

[5] Salgado, I. «Zigbee y sus aplicaciones». Madrid: s.n., 2012.

[6] Digi Internacional. XBEE/XBEE PRO/RF. USA: Modules, 2010. [en línea] [citado 2015-03-01] Disponible en internet: http://ftp1.digi.com/support/documentation/90000976.pdf

[7] Rodríguez O, D. Instrumentacion electronica SB-2013. Pasto: s.n., 2013.

[8] Microchip. «PIC16F87XA,» EEUU: s.n., 2013.

[9] Canós, P. L. y Penadés, M. C. «Metodologias Agiles,» España: s.n., 2013.

[10] Ávila Mejía, Ó. «ANDROID,» España: s.n., 2013. [en línea] [citado 2015-03-01] Disponible en internet: www.izt.uam.mx/new page/contactos /revista 83/pdfs/android.pdf

[11] Aliane, N. y Bemposta, S. &. «Arquitectura Robótica con Orientación Didáctica para Ingeniería,» Mexico: s.n., 2010. [en línea] [citado 2015-03-01] Disponible en internet: www.fimpes.org.mx/phcadownload/premios/1investigacion2010.pdf

[12] Sistema-Operativo-Android. 20 3. 2013. [en línea] [citado 2015-04-23] Disponible en internet: http://android .blogspot.com/2012/12/sistema-operativo-android.html.

[13] Curso Android. 20 1. 2015. [en línea] [citado 2015-04-02] Disponible en internet: http://www.maestrosdelweb.com/curso-android-construir-lector-de-feeds/.

[14] Navarro Montesinos, J. S. «Red de sensores Autoconfigurables Mediante Tecnologia Zigbee y Arduino con Monitorización por Aplicación por Android,» Cartagena: s.n., 2013.

[15] Agudelo Quiroz, S. A. «Red de Sensores Inalámbricos Utilizando ZIGBEE/802.15.4,» Medellín: s.n., 2013.

[16] Arduino. 19 9 2012. [en línea] [citado 2015-06-02] Disponible en internet: http://arduino.cc/en/Guide/Arduino.

[17] Damaso Larrañaga, A. «Problemas y soluciones en la implementación de extreme programming,» Montevideo: s.n., 2010.

[18] Fernández Reyes, D., Sánchez, F. J. y zquierdo García, A. I. «Estrategias pedagógicas para uso de los dispositivos,». Bogotá: s.n., 2010. [en línea] [citado 2015-03-01] Disponible en internet: www.uam.es/gruposinv/dim/assets/dalia-uned-2014

[19] Canós, J. H., Letelier, P. y Penadés, M. C. «Metodologías Agiles en el Desarrollo de Software,» Valencia: s.n, 2009.

[20] Rúiz Gutiérrez, J. M. «Manual de Programación Arduino,» Bogotá: s.n., 2007. [en línea] [citado 2015-03-01] Disponible en internet: arduinobot.pbworks.com/f/Manual +Programacion+Arduino.pdf

[21] Ruiz Gutiérrez, J. M. «Implementación de Sistemas de Transmisión de Datos y Sensores en Redes Inalámbricas con Xbee Integrado en la "Plataforma Open Hadware" Arduino,» Italia: s.n., 2012.

[22] Riascos Mejia, F. «Protocolo TCP/IP,» México: s.n., 2013.

[23] Brunetti, F., Moreno, J. y Ceres, R. «Redes inalámbricas de área personal al servicio de los discapacitados y personas mayores,» Madrid: s.n., 2012.

[24] Ruiz, P. S. «ZigBee,». 2012. [en línea] [citado 2015-03-01] Disponible en internet: https://docs.zigbee.org/zigbee-docs/dcn/09-4825.pdf

[25] Digi. International, «Manual Xbee ZB,» 2012. [en línea] [citado 2015-03-01] Disponible en internet: www.digi.com/support/productdetail? pid=4549

[26] Jiménez, C. «Curso Wifi,» Medellín: s.n., 2013.

[27] Recalde Baraibar, A. y Rodríguez Alija, M. «Redes Inalambricas/802.11,» México: s.n., 2013.

[28] Rojo, F. «Crea tu propia Red Wifi. Gijon: s.n., 2008. [en línea] [citado 2015-03-01] Disponible en internet: http://www.cabuenes.org/08/documentacion/taller14/CreapropiaredwifiFranciscoRojo.pdf

[29] González Iraceburu, J. y Goicoechea Fernández, J. «Desarrollo e implementación de una red inalámbrica de sensores de temperatura y humedad». Pamplona: s.n., 2014.

[30] Álvarez Erazo, F. X. «Implementación de una red de sensores inalámbricos con tecnología cluster tree en el laboratorio de istrumentacion industrial,» Quito: s.n., 2015.

[31] Instruments, T. «Datasheed ESP8266 Module – Wi-Fi 802.11b/g» EEUU: s.n., 2014.

[32] Aranaz Tudela, J. «Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma android de google,» Madrid: s.n., 2009.

[33] Lopez, D. y Gala Trallero, S. «Red inalámbrica para instrumentación de procesos,» Barcelona: s.n., 2012.

[34] Carrasco Muñoz, A. y Ropero Rodríguez, J. «Topologías de redes,» Sevilla: s.n., 2013.

[35] SparKfun. «XBee Shield Hookup Guide,» [en línea] [citado 2015-07-06] Disponible en internet: https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide.

[36] CCM. «El protocolo HTTP,» 4 8 2015. [en línea] [citado 2015-10-02] Disponible en internet: http://es.ccm.net/contents/264-el-protocolo-http.

[37] Oliva Patiño, D. E. y Verdugo Muñoz, C. E. «Implementación y caracterización de una red de sensores inalámbricos para el control distribuido de iluminación en edificaciones.,» Pasto: s.n., 2014.

[38] Cázarez Ayala, G. López Macías, C. Morales García, J. «Diseño de un prototipo didáctico para la implementación de redes de sensores inalámbricos basados en el protocolo zigbee,» México: 2011.

[39] Maldonado, D. «Aplicación en Android para el monitoreo de Variables en un Motor Eléctrico,» México: s.n., 2013.

[40] Jornadawifi. «Tecnologias Wifi y Wimax» 2014. [en línea] [citado 2015-03-01] Disponible en internet: www.dip-badajoz.es/agenda/tablon/jornada WiFi/doc/tecnologias-wifi-wimax.pdf

[41] Mohammed. El Yaagoubi, [en línea] [citado 2015-03-01] Disponible en internet: vía WiFiWiMax. Leganés: s.n., 2012.

[42] Electrónica, D. «I+D Electrónica -tarjeta reguladora para xbee,» Copyright I+D Electrónica, 2011. [en línea] [citado 2015-06-05] Disponible en internet: http://www.didacticaselectronicas.com/index.php?page=shop.product_details&flypage=flypage. tpl&product_id=2338& 71.

[43] Rabbit, A. Digi International Brand. [en línea] [citado 2015-03-01] Disponible en internet: http://ftp1.digi.com/support/documentation/0220145_a.pdf

[44] Sparfun. Descripción producto Xbee Xplorer Dongle sparkfun. [en línea] [citado 2015-03-01] Disponible en internet: https://www.sparkfun.com/products/8687.

[45] Sparfun. Logic Level Converter [en línea] [citado 2015-06-15] Disponible en internet: https://www.sparkfun.com/products/retired/11978.

[46] EXTECH INSTRUMENTS. Foot Candle/Lux Meter. [en línea] [citado 2015-10-13] Disponible en internet: https://extech.com/instruments/resources/datasheets/401025.pdf.

[47] NIKOLA TESLA. [en línea] [citado 2015-09-13] Disponible en internet: https://es.wikipedia.org/wiki/Nikola_Tesla.

ANEXOS

ANEXO A. CÓDIGO PIC

```
#include <16f876a.h>
#DEVICE ADC=10
#FUSES NOWDT
#FUSES HS
#FUSES NOPUT
#FUSES NOPROTECT
#FUSES NODEBUG
#FUSES NOBROWNOUT
#FUSES NOLVP
#FUSES NOCPD
#FUSES NOWRT

#USE DELAY(CLOCK=20000000)
#USE RS232(BAUD=9600,BITS=8,PARITY=N,XMIT=PIN_C6,RCV=PIN_C7)
#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#byte porta = 5
#byte portb = 7
#byte portc = 7
#include <stdio.h>
#include <math.h>
void main()
{
   //VARIABLES
   long A;
```

```
long B;
long C;
float RH;
float TEMP;
float Vout;
float HUM;
float LUX;
float D;

SETUP_ADC_PORTS(ALL_ANALOG);
SETUP_ADC(ADC_CLOCK_DIV_32);
SETUP_SPI(SPI_SS_DISABLED);
SETUP_TIMER_0(RTCC_INTERNAL|RTCC_DIV_1);
SETUP_TIMER_1(T1_DISABLED);
SETUP_TIMER_2(T2_DISABLED,0,1);
SETUP_CCP1(CCP_OFF);
SETUP_COMPARATOR(NC_NC_NC_NC);
SETUP_VREF(FALSE);

set_tris_a(0X3F); //configuración de puertos
set_tris_b(0x00); //configuración de puertos
set_tris_c(0X00);
while (true)
{
set_adc_channel(0);//selecciona la entrada analogica A_0   TEMPERATURA
   delay_ms(90);//retardo de 20 us
   A=read_adc();//lee el canal seleccionado
```

```
      TEMP=A*(5.0*100.0)/1024.0; //conversion a voltaje  temperatura


      set_adc_channel(1);//selecciona la entrada analogica A_1  HUMEDAD

      delay_ms(90);

      B=read_adc();//lee el canal seleccionado

      HUM=(5.0*B/1024.0); //conversion a voltaje humedad

      RH=30.77*HUM-21.54;


      set_adc_channel(2);//selecciona la entrada analogica A_2
      LUMINOSIDAD

      delay_ms(90);

      C=read_adc();//lee el canal seleccionado

      Vout=(5.0*C/1024.0);  //conversion a voltaje

      D=exp(-1.326*Vout);

      LUX=6958.2*D; //formula para encontrar la luminosidad en lux.

printf("xNodo1: %f, %f, %f\r",TEMP,RH,LUX);//imprime en RS232 el valor
VARIABLES DE TEMPERATURA ªC, HUMEDAD % Y LUMINOSIDAD EN LUX

         delay_ms(1000);


   }

}
```

ANEXO B. CÓDIGO ARDUINO

```
#include <SoftwareSerial.h>
const int RX1 = 10;
const int TX1 = 11;
SoftwareSerial xbee(RX1,TX1);

#define esp8266 Serial1

const int refresh = 5;

String comandos[]= {
  "AT",
  "AT+CWMODE=3",
  "AT+CIPMUX=1",
  "AT+CWJAP=\"android\",\"123456789\"",
  "AT+CIPSTA=\"192.168.43.15\"",
  "AT+CIPSERVER=1,80",
  "END"
};

String mns = "",
       mns2mns = "No hay mensaje",
       mnsxbee1 = "Nodo1: Vacio",
       mnsxbee2 = "Nodo2: Vacio",
       mnsxbee3 = "Nodo3: Vacio";

boolean bandera = true,
        xbee1 = false,
        xbee2 = false,
        xbee3 = false;

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Inicio Comandos AT: ");
  pinMode(RX1, INPUT);
  pinMode(TX1, OUTPUT);
  esp8266.begin(115200);

  lag(500);
  Serial.flush();
  esp8266.flush();
  lag(500);

  enviarComandos();

  xbee.begin(9600);
  xbee.flush();
  Serial.println("Xbee Inicializado\r\n");
  Serial.println("OK\r\n\r\n.........................");
}
```

```
void loop() {
  while (esp8266.available()){
    wifi:
    if(esp8266.find("+IPD,")){
      mns2mns = "<br />" + mnsxbee1 + ";" + mnsxbee2 + ";" + mnsxbee3;
      Serial.println("Enviada Web Request");
      lag(50);
      webserver(String(refresh),String (mns2mns));
    }
  }
  esp8266.flush();

  while(xbee.available()){
    if (esp8266.available())
      goto wifi;
    if(bandera){
      bandera = !bandera;
      mns2mns = "";
    }
    mns = GetLineXbee();
    mns2mns += mns;
  }
  if(xbee1 == true){
    mnsxbee1 = mns2mns;
    Serial.print("Nodo1 == ");
    Serial.println(mnsxbee1);
    xbee1 = false;
  }
  if(xbee2 == true){
    mnsxbee2 = mns2mns;
    Serial.print("Nodo2 == ");
    Serial.println(mnsxbee2);
    xbee2 = false;
  }
  if(xbee3 == true){
    mnsxbee3 = mns2mns;
    Serial.print("Nodo3 == ");
    Serial.println(mnsxbee3);
    xbee3 = false;
  }
  bandera = true;
}

void enviarComandos(){
  int index = 0;
  String response = "";
  String salir = "";

  while(comandos[index] != "END"){
    response = "";
    if(comandos[index] == "AT+RST")
      salir = "ready";
    else
      salir = "OK";
```

```
    esp8266.println(comandos[index]);
    while(true){
      while(esp8266.available()>0){
        char c = esp8266.read();
        response += c;
        lag(10);
      }
      if (response.endsWith(salir+"\r\n"))
        break;
    }
    response += "\n.........................";
    Serial.println(response);
    index++;
  }
}

String GetLineXbee(){
  String responsexbee = "";

  while(true){
    while(xbee.available()){
      char x = xbee.read();
      if(x=='x'){
       xbee1=true;
       xbee2=false;
       xbee3=false;
       x = xbee.read();
      }
      if(x=='y'){
       xbee1=false;
       xbee2=true;
       xbee3=false;
       x = xbee.read();
      }
      if(x=='z'){
       xbee1=false;
       xbee2=false;
       xbee3=true;
       x = xbee.read();
      }
      responsexbee += x;
    }
  break;
  }
  return(responsexbee);
}


void http(String output){
  esp8266.print("AT+CIPSEND=0,");
  esp8266.println(output.length());
  if (esp8266.find(">")){
    Serial.println(output);
```

```
    esp8266.println(output);
    lag(10);
    while (esp8266.available() > 0 ){
      if (esp8266.find("SEND OK") )
        break;

  }
}

void webserver(String sg, String mnsx){

  http("HTTP/1.1 200 OK");
  http("Content-Type: text/html");
  http("\r\n");
  http("Connection: close");
  http("\r\n");
  http("<!DOCTYPE HTML>");
  http("<html>");
  http("<head><title>MENSAJE XBEE</title>");
  http("<meta http-equiv=\"refresh\" content=\""+sg+"\"></head>");
  http("<body><h1> MENSAJE</h1>");
  http("The message is: ");
  http( String(mnsx));
  http("<br />");
  http("<p><em> La pagina se actualiza cada "+sg+"
segundos.</em></p></body></html>");
  esp8266.println("AT+CIPCLOSE=0");
  lag(1);
}

void lag(const long interval){
  unsigned long currentMillis = millis();
  unsigned long previousMillis = currentMillis;
  while(currentMillis - previousMillis < interval)
    currentMillis = millis();
}
```

ANEXO C. CÓDIGO FUENTE DE LA PANTALLA PRINCIPAL DE LA APLICACIÓN MÓVIL EN VISUAL STUDIO

CODIGO INTERFACES

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android

    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:minWidth="25px"
    android:minHeight="25px">

    <ScrollView
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/MainScroll">

        <LinearLayout

            android:orientation="vertical"
            android:minWidth="25px"
            android:minHeight="25px"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/MainLayout">

            <ImageView

                android:src="@drawable/title"
                android:layout_width="match_parent"
                android:layout_height="202.8dp"
                android:id="@+id/imgTitle"
                android:layout_marginBottom="0.0dp" />

            <TextView

                android:text="Nikola - Electronic Instrumentation App"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/textView1"
                android:gravity="center" />

            <Button

                android:text="Theoretical Background"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/btnTheoreticalBg" />
```

```
<Button
                android:text="Exercises"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/btnExercises" />

            <Button
                android:text="Real-Time Monitoring"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/btnRTMonitoring" />
            <Button
                android:text="About"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/btnAbout" />
            <Button
                android:text="Exit"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/btnExit" />
        </LinearLayout>
    </ScrollView>

                                    </LinearLay
```

PARTE EXERCISES

```
using System;
using Android.App;
using Android.Util;
using Android.Widget;
using System.Text;

namespace Nikola
{
    [Activity(Label = "Exercises", Icon = "@drawable/icon", ConfigurationChanges =
Android.Content.PM.ConfigChanges.Orientation |
Android.Content.PM.ConfigChanges.ScreenLayout)]
    public class ExercisesActivity : Activity
    {
        private int sec, node, sensor, i;
        private const int MAX = 10;
        private Button btnProposeExercise, btnVerifyAnswer;
        private Chronometer clock;
        private TextView lblProblem;
        private RadioGroup radAnswer;
        private Random rnd = null;
        private double mean = 0.0;
        private double averagedeviation;
        private double standarddeviation;
        private double variance;
        private double[] data = new double[MAX];
        private double[] deviation = new double[MAX];
        private int rightanswer;

protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Exercises);
              btnProposeExercise = FindViewById<Button>(Resource.Id.btnProposeExercise);
              btnVerifyAnswer = FindViewById<Button>(Resource.Id.btnVerifyAnswer);
              btnVerifyAnswer.Visibility = Android.Views.ViewStates.Gone;
              lblProblem = FindViewById<TextView>(Resource.Id.lblProblem);
              lblProblem.Visibility = Android.Views.ViewStates.Gone;
              radAnswer = FindViewById<RadioGroup>(Resource.Id.radAnswer);
              radAnswer.Visibility = Android.Views.ViewStates.Gone;
            btnProposeExercise.Click += BtnProposeExercise_Click;
            btnVerifyAnswer.Click += BtnVerifyAnswer_Click;
            clock = FindViewById<Chronometer>(Resource.Id.clockExec);
            clock.ChronometerTick += Clock_ChronometerTick;
            rnd = new Random(DateTime.Now.Second);
        }

private void Clock_ChronometerTick(object sender, EventArgs e)

              {
            sec++;
            StringBuilder txt = new StringBuilder("");
            string d;
            // Borrar esta parte una vez ya este terminado...
            d = Convert.ToString(Math.Round(rnd.NextDouble() * 100, 2));
            data[i] = double.Parse(d);
            i++;
            switch (sensor)
```

```
            {
                case 0: d += "°"; break;
                case 1: d += "%"; break;
                case 2: d += "lux"; break;
            }
            txt.Append(d); txt.Append(", ");
            txt.Remove(txt.Length - 1, 1);
            lblProblem.Text += txt.ToString();
            if (i == 10)
            {
              lblProblem.Text += ". The value of " + this.Calculations() + " corresponds
              to:";
            radAnswer.Visibility = Android.Views.ViewStates.Visible;
              btnVerifyAnswer.Visibility = Android.Views.ViewStates.Visible;
                clock.Stop();
            }
        }

private void BtnProposeExercise_Click(object sender, EventArgs e)
        {
              btnProposeExercise.Visibility = Android.Views.ViewStates.Gone;
            lblProblem.Visibility = Android.Views.ViewStates.Visible;
            node = rnd.Next(0, 2);
            sensor = rnd.Next(0, 2);
            switch (sensor)
            {
                    case 0: lblProblem.Text = "The sensor of Temperture (LM35) of the
                    node " + node.ToString() + " is sending the following 10 values:";
                    break;
                    case 1: lblProblem.Text = "The sensor of Humidity (HIH-4030) of the
                    node " + node.ToString() + " is sending the following 10 values:";
                    break;
                    case 2: lblProblem.Text = "The sensor of Luminosity (LDR) of the node
                    " + node.ToString() + " is sending the following 10 values:"; break;
            }
            sec = 0;
            i = 0;
            clock.Start();
        }

private string Calculations()
        {
            mean = 0.0;
            averagedeviation = 0.0;
            standarddeviation = 0.0;
            int i = 0;
            string result = "";
            //
            for (i = 0; i < MAX; i++)
            {
                mean += data[i];
            }
            mean /= MAX;
            for (i = 0; i < MAX; i++)
            {
                deviation[i] = data[i] - mean;
                averagedeviation += deviation[i];
                standarddeviation += Math.Pow(deviation[i], 2.0);
            }
```

```
            averagedeviation /= MAX;
            standarddeviation /= (MAX - 1);
            variance = standarddeviation;
            standarddeviation = Math.Sqrt(standarddeviation);
            rightanswer = rnd.Next(0, 2);
            switch (rightanswer)

            {
                     case 0: result = Convert.ToString(Math.Round(mean * 100, 2));
                     break;

                     case 1: result = Convert.ToString(Math.Round(standarddeviation *
                     100, 2)); break;

                     case 2: result = Convert.ToString(Math.Round(variance * 100, 2));
                     break;
            }
            return (result);
        }

private void BtnVerifyAnswer_Click(object sender, EventArgs e)

       {
            int id = radAnswer.CheckedRadioButtonId;
            int r1 = FindViewById<RadioButton>(Resource.Id.radioButton1).Id;
            int r2 = FindViewById<RadioButton>(Resource.Id.radioButton2).Id;
            int r3 = FindViewById<RadioButton>(Resource.Id.radioButton3).Id;

       if (id == r1 && rightanswer == 0) Toast.MakeText(this, "Well done :-)",
       ToastLength.Long).Show();

       else if (id == r2 && rightanswer == 1) Toast.MakeText(this, "Well done :-)",
       ToastLength.Long).Show();

       else if (id == r3 && rightanswer == 2) Toast.MakeText(this, "Well done :-)",
       ToastLength.Long).Show();
            else Toast.MakeText(this, "Wrong :-(", ToastLength.Long).Show();
        }

public override void OnBackPressed()
        {
            clock.Stop();
            this.Finish();
        }

public override void OnConfigurationChanged(Android.Content.Res.Configuration newConfig)

        {
            base.OnConfigurationChanged(newConfig);
            if (newConfig.Orientation != Android.Content.Res.Orientation.Portrait)
            {
                this.RequestedOrientation =
Android.Content.PM.ScreenOrientation.Portrait;
            }
        }
```

PARTE THEORETICAL BACKGROUND

```
using Android.App;
using Android.OS;
using Android.Util;
using Android.Webkit;

namespace Nikola
{
    [Activity(Label = "Theoretical Background", Icon = "@drawable/icon",
ConfigurationChanges = Android.Content.PM.ConfigChanges.Orientation |
Android.Content.ConfigChanges.ScreenLayout)]
    public class TheoreticalBgActivity : Activity
    {
        //----------------------------------------------------------------------
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.TheoreticalBg);
            WebView wv1 = FindViewById<WebView>(Resource.Id.wv1);
            wv1.LoadUrl("file:///android_asset/TB.html");
        }
        //----------------------------------------------------------------------
        public override void OnConfigurationChanged(Android.Content.Res.Configuration
newConfig)
        {
            base.OnConfigurationChanged(newConfig);

            if (newConfig.Orientation != Android.Content.Orientation.Portrait)
            {
                this.RequestedOrientation = Android.Content.ScreenOrientation.Portrait;
            }
        }
        //----------------------------------------------------------------------
    }
}
```

REAL TIME MONITORING

```
using System;
using System.Text;
using Android.App;
using Android.Widget;
using OxyPlot.Xamarin.Android;
using Android.Views;

namespace Nikola
{
    [Activity(Label = "Real-Time Monitoring", Icon = "@drawable/icon",
ConfigurationChanges = Android.Content.PM.ConfigChanges.Orientation |
Android.Content.PM.ConfigChanges.ScreenLayout)]
        private int node, sec, sensor;
        private Chronometer clock = null;
        private Button btnNode1, btnNode2, btnNode3;
        private TextView lblT, lblH, lblLI;
        private FrameLayout fl = null;
        private MyClass myClass = null;
        //-------------------------------------------------------------------------------
-------------------
        private double[] GetRealData(int node)
        {
            double[] mydata = new double[3];
            string[] mystrings = new string[3] { "", "", "" };
            WebClient wc = new WebClient();
            try
            {
                byte[] raw = wc.DownloadData("http://192.168.43.15");
                string html = Encoding.UTF8.GetString(raw);
                //
                int ip = html.IndexOf("Nodo1");
                int fp = html.IndexOf("<p><em>");
                string data = html.Substring(ip, fp - ip);
                string[] sensors = nodes[node - 1].Split(',');
                sensors[0] = sensors[0].Substring(7);
                mydata[0] = Double.Parse(sensors[0]);
                mydata[1] = Double.Parse(sensors[1]);
                mydata[2] = Double.Parse(sensors[2]);
            }
            catch (Exception ex)
            {
                mydata[0] = 0.0;
                mydata[1] = 0.0;
                mydata[2] = 0.0;
                lblLI.Text = ex.Message;
            }
            wc.Dispose();
            return (mydata);
        }
        //-------------------------------------------------------------------------------
-------------------
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.RealTimeMonitoring);
            btnNode1 = FindViewById<Button>(Resource.Id.btnNode1);
```

```
            btnNode2 = FindViewById<Button>(Resource.Id.btnNode2);
            btnNode3 = FindViewById<Button>(Resource.Id.btnNode3);
            lblT = FindViewById<TextView>(Resource.Id.lblT);
            lblH = FindViewById<TextView>(Resource.Id.lblH);
            lblLI = FindViewById<TextView>(Resource.Id.lblLI);
            lblT.Click += LblT_Click;
            lblH.Click += LblH_Click;
            lblLI.Click += LblCO_Click;
            btnNode1.Click += BtnNode1_Click;
            btnNode2.Click += BtnNode2_Click;
            btnNode3.Click += BtnNode3_Click;
            clock.ChronometerTick += Clock_ChronometerTick;
            clock.Start();
            fl = FindViewById<FrameLayout>(Resource.Id.myGraph);
            node = 1;
            sensor = 1;
            sec = 0;
            txt.Clear();
            this.LoadNode();
        }
        //--------------------------------------------------------------------------------
--------------------
        private void GetData()
        {
            string t, h, lux, title;
            //
            double[] mydata = this.GetRealData(node);
            t = mydata[0].ToString();
            h = mydata[1].ToString();
            lux = mydata[2].ToString();
            //.............................................
            lblT.Text = "Temperture: " + t.ToString() + " °";
            lblH.Text = "Humidity: " + h.ToString() + " %";
            lblLI.Text = "Light Intensity: " + lux.ToString() + " lux";
            title = "Node " + node.ToString();
            txt.Append(sec.ToString()); txt.Append(',');
            switch (sensor)
            {
                case 1:
                    title += " - Temperture";
                    txt.Append(t);
                    break;
                case 2:
                    title += " - Humidity";
                    txt.Append(h);
                    break;
                case 3:
                    title += " - Light Intensity";
                    txt.Append(lux);
                    break;
            }
            txt.Append(',');
            myClass = new MyClass(title, txt.ToString(), sensor);
            var plotView = new PlotView(this)
            {
                Model = myClass.MyModel

            fl.RemoveAllViews();
            fl.AddView(plotView,
```

```
                new ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.MatchParent,
                    ViewGroup.LayoutParams.MatchParent));

        }
        //------------------------------------------------------------------------------
--------------------
        private void Clock_ChronometerTick(object sender, EventArgs e)
        {
            sec++;
            if (sec % 5 == 0)
            {
                this.GetData();
            }
        }
        //------------------------------------------------------------------------------
--------------------
        private void LblT_Click(object sender, EventArgs e)
        {
            lblT.SetTextColor(Android.Graphics.Color.Yellow);
            lblH.SetTextColor(Android.Graphics.Color.White);
            lblLI.SetTextColor(Android.Graphics.Color.White);
            sensor = 1;
            sec = 0;
            txt.Clear();
            this.GetData();
        }
        //------------------------------------------------------------------------------
--------------------
        private void LblH_Click(object sender, EventArgs e)
        {
            lblT.SetTextColor(Android.Graphics.Color.White);
            lblH.SetTextColor(Android.Graphics.Color.Yellow);
            lblLI.SetTextColor(Android.Graphics.Color.White);
            sensor = 2;
            sec = 0;
            txt.Clear();
            this.GetData();
        }
        //------------------------------------------------------------------------------
--------------------
        private void LblI_Click(object sender, EventArgs e)
        {
            lblT.SetTextColor(Android.Graphics.Color.White);
            lblH.SetTextColor(Android.Graphics.Color.White);
            lblLI.SetTextColor(Android.Graphics.Color.Yellow);
            sensor = 3;
            sec = 0;
            txt.Clear();
            this.GetData();
        }
        //------------------------------------------------------------------------------
--------------------
        private void BtnNode1_Click(object sender, EventArgs e)
        {
            node = 1;
            sensor = 1;
            sec = 0;
            txt.Clear();
```

```
            this.LoadNode();
        }
        //--------------------------------------------------------------------------------
--------------------
        private void BtnNode2_Click(object sender, EventArgs e)
        {
            node = 2;
            sensor = 1;
            sec = 0;
            txt.Clear();
            this.LoadNode();
        }
        //--------------------------------------------------------------------------------
--------------------
        private void BtnNode3_Click(object sender, EventArgs e)
        {
            node = 3;
            sensor = 1;
            sec = 0;
            txt.Clear();
            this.LoadNode();
        }
        //--------------------------------------------------------------------------------
--------------------
        private void LoadNode()
        {
            switch (node)
            {
                case 1:
                    btnNode1.SetTextColor(Android.Graphics.Color.Black);
                    btnNode2.SetTextColor(Android.Graphics.Color.White);
                    btnNode3.SetTextColor(Android.Graphics.Color.White);
                    break;
                case 2:
                    btnNode1.SetTextColor(Android.Graphics.Color.White);
                    btnNode2.SetTextColor(Android.Graphics.Color.Black);
                    btnNode3.SetTextColor(Android.Graphics.Color.White);
                    break;
                case 3:
                    btnNode1.SetTextColor(Android.Graphics.Color.White);
                    btnNode2.SetTextColor(Android.Graphics.Color.White);
                    btnNode3.SetTextColor(Android.Graphics.Color.Black);
                    break;
            }
            this.GetData();
            lblT.SetTextColor(Android.Graphics.Color.Yellow);
            lblH.SetTextColor(Android.Graphics.Color.White);
            lblLI.SetTextColor(Android.Graphics.Color.White);
        }

        public override void OnConfigurationChanged(Android.Content.Res.Configuration
newConfig)
        {
            base.OnConfigurationChanged(newConfig);

        }
```

GRÁFICAS

```
using OxyPlot;
using OxyPlot.Axes;
using OxyPlot.Series;
namespace Nikola
{
    public class MyClass
    {
        public PlotModel MyModel { get; set; }
        public PlotModel plotModel = new PlotModel();
        public LinearAxis xaxis = new LinearAxis();
        public LinearAxis yaxis = new LinearAxis();
        public MyClass(string title, string points, int sensor)
        {
            plotModel.Title = title;
            xaxis.Position = AxisPosition.Bottom;
            yaxis.Position = AxisPosition.Left;
            yaxis.Minimum = 0;
            if (title.Contains("Light")) yaxis.Maximum = 300;
            else yaxis.Maximum = 100;
            plotModel.Axes.Add(xaxis);
            plotModel.Axes.Add(yaxis);
            {
                MarkerSize = 4,
                MarkerStroke = OxyColors.White,
            };
            var array = points.Split(',');
            for (int i = 0; i < array.Length - 1; i += 2)
            {
                series1.Points.Add(new DataPoint(double.Parse(array[i]),
double.Parse(array[i+1])));
            }
            switch (sensor)
            {
                case 1: series1.Color = OxyColors.Red; break;
                case 2: series1.Color = OxyColors.Blue; break;
                case 3: series1.Color = OxyColors.Green; break;
            }
            plotModel.Series.Add(series1);

    }
```