DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

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

AUTORES: ABASOLO ARANDA, SANDY ESTEFANY

CARRERA PAZ Y MIÑO, MICHELLE ALEJANDRA

TEMA: EVALUACIÓN DEL MODELO DE REFERENCIA DE ¨INTERNET OF THINGS¨ (IoT), MEDIANTE LA IMPLANTACIÓN DE ARQUITECTURAS BASADAS EN PLATAFORMAS COMERCIALES, OPEN HARDWARE Y CONECTIVIDAD IPv6

DIRECTOR: ING. GORDILLO, RODOLFO

CODIRECTOR: ING. ROMERO, CARLOS

SANGOLQUÍ, ENERO 2014

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICADO

Ing. Rodolfo Gordillo

Ing. Carlos Romero

CERTIFICAN

Que el trabajo titulado “Evaluación del modelo de referencia de Internet of Things (IoT), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad ipv6”, realizado por Sandy Estefany Abasolo Aranda y Michelle Alejandra Carrera Paz y Miño, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Universidad de las Fuerzas Armadas - ESPE.

Sangolquí, 20 de Diciembre de 2013

Ing. Rodolfo Gordillo	Ing. Carlos Romero
DIRECTOR	CODIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

SANDY ESTEFANY ABASOLO ARANDA

MICHELLE ALEJANDRA CARRERA PAZ Y MIÑO

DECLARAMOS QUE:

El proyecto de grado denominado “Evaluación del modelo de referencia de Internet of Things (IoT), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad ipv6”, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie, de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 20 de Diciembre de 2013

Sandy Estefany Abasolo Aranda

Michelle Alejandra Carrera Paz y Miño

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

SANDY ESTEFANY ABASOLO ARANDA

MICHELLE ALEJANDRA CARRERA PAZ Y MIÑO

Autorizamos a la Universidad de las Fuerzas Armadas - ESPE la publicación, en la biblioteca virtualde la Institución del trabajo “Evaluación del modelo de referencia de Internet of Things (IoT), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad ipv6”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría

Sangolquí, 20 de Diciemebre de 2013

Sandy Estefany Abasolo Aranda

Michelle Alejandra Carrera Paz y Miño

DEDICATORIA

A mis padres y hermana, por ser mis mejores amigos, este logro es suyo también porque sin ustedes no lo hubiera logrado, por ser una familia unida y darnos apoyo unos a otros, por todo el amor que me han dado. A mi papá porque con su esfuerzo día a día es mi mayor ejemplo y con sus concejos siempre me guio para ser una persona de bien. A mi mamá porque me enseñó a ser mujer y me indico como ser responsable y profesional en la vida. A mi hermana por la paciencia y por ser mi confidente. A ustedes les dedico este logro esperando que todos estén orgullosos de mí y dándoles una alegría, gracias por siempre estar a mí lado. Los amo con todo mi corazón.

Sandy Abasolo Aranda.

Dedico este trabajo a mis padres y hermanos que me han dado su apoyo incondicional durante toda mi carrera, su confianza y motivación permanente para cumplir con mis objetivos y metas. A mi familia, amigos y compañeros que siempre estuvieron dispuestos a brindarme toda su ayuda y así lograr culminar un escaño más en mi vida.

Michelle Carrera Paz y Miño

AGRADECIMIENTO

A Dios por guiarme por el buen camino y darme la sabiduría para tomar las mejores decisiones y siempre escuchar todas mis oraciones permitiéndome así culminar esta meta tan importante en mi vida.

A mi Padre y Madre por siempre darme su apoyo en todos los buenos y malos momentos de todo este proceso, por ser mí modelo a seguir y darme todos los concejos necesarios para salir adelante. A mi hermana Leslie por ser mi amiga y siempre estar ahí para escucharme. Porque ustedes son mi complemento y fuerza en mi vida. Los amo mucho.

A una persona muy importante en mi vida Andrés Cepeda por ser mi compañía y siempre querer lo mejor para mí, por estar presente en todo este proceso y darma todo su apoyo.

A mi amiga y compañera Michelle Carrera por siempre ponerle toda la dedicación, por aguantarnos las malas caras y por qué juntas logramos alcanzar esta meta propuesta a pesar de todas las dificultades.

A toda mi familia y amigos por ser los que han confiado en mí, estar a mi lado y tenerme paciencia.

Sandy Abasolo Aranda.

Las cosas buenas de esta vida pasan cuando estamos en compañía de nuestros seres queridos, por eso mi sincero agradecimiento a Dios por todas las bendiciones que he recibido. A mis padres por brindarme la oportunidad de estudiar, a mis hermanos por compartir sus experiencias y conocimientos.

Quiero agradecer a cada uno de los miembros de mi familia, pero en especial a mi tía Martha que siempre representó un apoyo en la culminación de mis metas. A mi compañera de tesis Sandy, por su constancia y empeño para superar todas las dificultades que se presentaron en el desarrollo del proyecto.

A mi Director y Codirector de tesis por su tiempo, paciencia y apoyo para la consecución de este trabajo.

Michelle Carrera Paz y Miño.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

GLOSARIO

6LoWPAN: *IPv6 over Low power Wireless Personal Area Networks*. Es un estándar que posibilita el uso de IPv6 sobre redes basadas en el estándar IEEE 802.15.4.

A

AAA: *Authentication, Authorization, Accounting.* Es un protocolo de seguridad informática que realiza tres funciones: Autenticación, Autorización y Contabilización.

API: *Application Programming Interface.* Es un conjunto de funciones que permite al programador acceder a servicios de una aplicación mediante el uso de un lenguaje de programación.

APS: *Application Support Sublayer.* Es el subnivel de soporte de la capa de aplicación de una red Zigbee.

Arduino: Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo.

C

CASAGRAS: Proyecto que tiene como objetivo proporcionar un marco de estudios que ayuden a la comunidad Europea a definir y adaptar temas relacionados con RFID.

CEDIA: Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado. Integrada por Universidades e Instituciones de Investigación y Desarrollo para estimular el proyecto Redes Avanzadas.

Cloud service: Servicio en la nube. Es cualquier recurso que se proporciona a través de Internet

CoAP: *Constrained Application Protocol*. Protocolo de transferencia web especializado para su uso con nodos y redes WNS.

CoS: *Class of Service*. Es una forma de gestión de tráfico en una red mediante la agrupación de tipos similares de tráfico y tratando cada tipo como una clase con su propio nivel de prioridad de servicio.

CSS: *Cascading Style Sheets.* Lenguaje de hojas de estilos que controla el aspecto de documentos HTML y XHTML.

F

FCAPS: *Fault, Configuration, Accounting, Performance, Security*. Modelo de red de gestión de telecomunicaciones de ISO, para la gestión de redes.

G

GET: Método de una petición HTTP. Devuelve el recurso identificado en la URL pedida.

Grids: Colección de computadoras, interconectadas de forma que los usuarios pueden compartir el acceso a su poder combinado.

H

HomePlug: Estándar de red por línea eléctrica que utiliza cables existentes para crear una red local.

HomePNA: *Home Phoneline Networking Alliance*. Estándar para la interconexión de ordenadores, usando líneas telefónicas existentes y un jack registrado.

HTML: *HyperText Markup Language.* Lenguaje utilizado para crear páginas web.

HTTP: *HyperText Transfer Protocol.* Método de intercambio de información en la www.

I

IaaS: *Infraestructure as a Service*. Modelo de distribución de infraestructura de computación como servicio, mediante una plataforma de virtualización.

ICSP: *In Circuit Serial Programming.*Conector que disponen algunas placas para actualizar o reprogramar el chip sin sacarlo del zócalo donde se encuentre.

IDE: *Integrated Development Environment.* Entorno de programación empaquetado como un programa de aplicación.

IETF: *Internet Engineering Task Force.*Organización que administra tareas de ingeniería de telecomunicaciones, principalmente de Internet.

IPsec: *Internet Protocol Security*. Conjunto de protocolos para asegurar las comunicaciones sobre IP, autenticando y/o cifrando cada paquete.

ISM: *Industrial, Scientific and Medical*. Bandas de radiofrecuencia reservadas para uso no comercial en áreas de trabajo industriales, científicas y médicas.

J

JSON: *JavaScript Object Notation*. Formato ligero de intercambio de datos.

JVM: *Java Virtual Machine*. Máquina virtual ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones de un compilador Java.

L

LibSSH2*:* Es una librería para el cliente implementada en el protocolo SSH2.

LonWorks: *Local Operation Networ*. Es una plataforma para sistemas de redes de control avanzados.

M

M2M**:** *Machine to Machiche*. Se refiere a la comunicación entre dos máquinas remotas.

Mapping**:** Es un objeto físico que puede estar representado por cosas virtuales.

Multihoming**:** Se refiere a un dispositivo u ordenador conectado a más de una red permitiendo aumentar la fiabilidad de una red.

N

NAT: *Network Adress Translation*. Permite intercambiar paquetes entre dos redes que asignan diferentes direcciones.

NFC: *Near Field Communication*. Es una tecnología de comunicación inalámbrica de corto alcance.

NRENs: *National Research and Education Network.* Es un proveedor de servicios de Internet para investigación y educación en un país.

NWK: *Network*. Facilita un interfaz al nivel superior para poder comunicarse con la MAC de 802.15.4.

O

Off-the-shelf*:* Son adaptadores que proporcionan conectividad inalámbrica instantánea a los dispositivos digitales existentes.

Open hardware: Son dispositivos que permiten que susespecificaciones y diagrams de hardware sean públicos.

Open source: Es utilizado para realizar algun cambio en el software de forma pública.

OpenSSH: *Open Secure Shell.* Son aplicaciones que permiten realizar comunicaciones cifradas a través de una red, usando el protocolo SSH.

P

PaaS: *Platform as a Service.* Es un modelo que soportar el ciclo de vida completo de construcción y puesta en marcha de aplicaciones.

PAN*: Personal Area Network*. Una PAN es una red digital orientada a la interconexión de dispositivos dentro de un rango de distancias < 20 metros.

PHP: Es un lenguaje de programación de uso general de código del lado del servidor diseñado para el desarrollo web de contenido dinámico.

POST *:* Metodo de peticiones del protocolo HTTP, envía información desde el cliente hacia el servidor.

Q

QoS*: Quality of Service*. Es el rendimiento promedio de una red, particularmente el rendimiento visto por los usuarios de la red.

R

***RCI**: Remote Command Interface*. Es un protocolo, que utiliza XML y HTTP para intercambiar datos entre el usuario y los dispositivos.

***REST:** Representational State Transfer*.Es una técnica de arquitectura software que se usa para describir una interfaz web como XML y HTTP.

***RFID:** Radio Frequency IDentification*. Es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados tags**.**

S

***SaaS**: Software as a Service*. Modelo de distribución de software para dar soporte y operación que usará el cliente.

***SCADA**: Supervisory Control And Data Acquisition*. Es un software que permite controlar y supervisar procesos industriales a distancia.

***SDK**: Software Development Kit*.Es un conjunto de herramientas de desarrollo de software para crear aplicaciones para un sistema concreto.

***Servlets**:* Los Servlets son módulos escritos en Java que se utilizan un servidor, que puede ser o no ser servidor web.

***SMA**: Sub-Miniature version A*. Es un conector coaxial para RF fabricado como conector en miniatura para la transmisión de señales.

***Smart Home**:* Son sistemas de automatización para el hogar que le permiten tener el control de toda la casa aumentando el ahorro de energía.

***SSH**: Secure SHell*. Es un protocolo que sirve para acceder a máquinas remotas a través de una red.

T

***TCP**: Transmission Control Protocol*. Es un protocolo que garantiza que los datos serán entregados en su destino sin errores.

TinyCore: Es un sistema operativo minimalista sin entorno de escritorio y que ocupa 13 Mb de RAM, el cual es muy estable como servidor.

***TSP**: Team Software Process,* El objetivo del TSP es mejorar los niveles de calidad y productividad de un proyecto de desarrollo de software.

TunnelBroker: Es un servicio que proporciona un túnel de red. Estos túneles pueden proporcionar conectividad encapsulada.

U

***UDP**: User Datagram Protocol,* Es un protocolo del nivel de transporte basado en el intercambio de datagramas.

V

v6udpv4: Es un tipo especial de túnel IPv6 que encapsula los paquetes IPv6 en paquetes IPv4-UDP con el fin de atravesar los NATs.

W

***WSAN**: Wireless sensor and actor networks.* Estas redes se refieren a un grupo de sensores y actores unidos por medio inalámbrico.

X

***XML**: Extensible Markup Language*. Es un lenguaje de marcas utilizado para almacenar datos en forma legible.

RESUMEN

En el presente proyecto se evalúa el modelo de referencia del *Internet of Things*, mediante el diseño e implementación de prototipos de red basados en las plataformas Arduino, Digi e IPv6. Cada prototipo emplea distintas tecnologías dentro de las capas del modelo. En los prototipos Arduino y Digi se realiza el monitoreo de parámetros del ambiente como temperatura y luminosidad y se ejecuta el control de encendido y apagado de dispositivos. Estas tareas se visualizan y gestionan desde una Interfaz web desarrollada y desde una nube de dispositivos propia, en el caso de Digi. Se demuestra y analiza la factibilidad del procesamiento de datos desde la nube. El prototipo basado en IPv6 realiza el monitoreo del estado de dispositivos virtualizados IPv6, desde una interfaz, a la que se puede acceder dentro de una red nativa IPv6, que para este proyecto es la red de CEDIA, o mediante un proveedor de túneles IPv6 sobre IPv4. Como un ejemplo del manejo de dispositivos remotos dentro del modelo de referencia IoT, se presenta una aplicación para el sistema operativo Android que realiza el control de iluminación de un espacio, dentro de una red local. Por último se analiza que los prototipos se acoplen al modelo de referencia y se establecen las diferencias entre ellos.

PalabrasClaves:

Internet of things, Arduino, Digi, IPv6, Android, CEDIA, Monitoreo, Gestión.

ABSTRACT

In this project the reference model of Internet of Things is evaluated through the design and implementation of network prototypes that are based on Arduino, Digi and IPv6 platforms. Each prototype uses different technologies related to the layers of the reference model. The Arduino and Digi prototypes are in charge of the environmental parameters monitoring like temperature and luminosity and the control on and off of devices. These prototypes allow visualizing and managing devices through a proprietary cloud-based infrastructure and on a web application developed. This project demonstrates and analyses the feasibility of cloud data processing. The prototype based on IPv6 performs the status monitoring of virtualized IPv6 devices from an interface, which can be accessed through a native IPv6 network, for this project is CEDIA, or by an IPv6 tunnel provider. As an example of managing remote devices related to the IoT reference model, this project presents an application for the Android operating system that allows controlling space lighting, on a local network. Finally we analyze that prototypes are coupled to the reference model and the differences between them are established.

Key words

Internet of things, Arduino, Digi, IPv6, Android, CEDIA, Monitoring, Managing

CAPÍTULO 1

PRESENTACIÓN DEL PROYECTO

1.1 INTRODUCCIÓN

El mundo se está volviendo cada vez más digitalizado e interconectado. Por el momento, la atención se ha centrado en las interacciones humanas, pero esto está cambiando rápidamente. La comunicación entre dispositivos se encuentra en constante aumento, la cual en sus iniciosestaba basada en sistemas como *SCADA*, *M2M*y actualmente se han desarrollado nuevos conceptos como el *Internet of things*.

IoT (*Internet of things*) representa la evolución del internet que permite relacionar objetos con objetos y objetos con personas, los cuales se pueden conectar, comunicar y controlar a través de la red presentando los datos en tiempo real en una nube de servicios (*Cloud service*), de tal forma que la información que se obtiene se encuentre disponible en cualquier lugar y en todo momento.

En la actualidad IoT es de gran interés e importancia como base tecnológica para la construcción de una nueva arquitectura de comunicaciones.

Se tiene como referenciaIoT-A, es un proyecto Europeo que aborda el concepto del *Internet of things*, propone la creación de un modelo de referencia de arquitectura junto con la definición de un conjunto inicial de elementos fundamentales. Dentro del modelo de referencia IoT se incluyen un modelo de dominio en el que se describen los principales conceptos IoT y la relación entre ellos, un modelo de información encargado del almacenamiento e intercambio de información y un modelo de comunicación que describe como la misma debe ser manejada para cumplir con los requerimientos de IoT.

Una de las principales tecnologías empleadas en los trabajos implementados sobre IoT son las Redes Inalámbricas de Sensores y Actuadores (*WSAN*) las cuales facilitan su desarrollo. Para lo cual se cuenta con la red Zigbee® que ofrece estándares inalámbricos y globales, que conecta una amplia gama de dispositivos para que funcionen en conjunto de forma inteligente, además de haber fortalecido su posición de liderazgo como el estándar global para los sensores inalámbricos como las redes de control y del IoT.

El IoT se encuentra directamente relacionado con el modelo *Cloud Computing*, el cual es un modelo tecnológico quepermite un acceso ubicuo, adaptado y bajo demanda en red. Los consumidores pueden utilizar aplicaciones sin instalarlas y acceder a datos de sensores y dispositivos conectados a la nube de Internet.

Una nueva tendencia para realizar la gestión de dispositivos dentro de IoT, es la basada en esquemas con elementos direccionables, que adoptan el protocolo IP, debido a que permite interactuar miles de dispositivos, los cuales no requieren de intermediarios para acceder a la red y con los que se puede lograr una hiperconectividad, introduciendo así un identificador único para cada objeto.

A través de recursos destinados a proyectos de investigación, la ESPE ha adquirido varios kits de desarrollo en redes de sensores inalámbricos. Particularmente en el año 2010 se adquirió un kit de la marca Xbee ahora Digi que es una plataforma de tipo comercial útil para la implementación de IoT con tecnología Zigbee®, que no ha sido explotada completamente en todas sus potencialidades.Digi proporciona una nube de dispositivos Etherios, que es una plataforma de nube pública para la gestión de dispositivos de red. Proporciona mensajería segura de aplicaciones, almacenamiento de datos y gestión de dispositivos.

En contraparte a las plataformas de desarrollo comerciales, existen alternativas de tipo *open – source* para estos mismos propósitos, una de ellas es la plataforma Arduino. Entre sus beneficios se encuentra su bajo costo, es multiplataforma, su ambiente de programación es simple, claro y tanto su software como su hardware son extensibles.

En este proyecto se evaluará el modelo de referencia de "*Internet of things*" (IoT) que combinen las herramientas y protocolos anteriormente expuestos mediante la implantación de arquitecturas basadas en plataformas comerciales (Digi), *Open Hardware* (Arduino) y conectividad IPv6, para la gestión remota de dispositivos en la nube.

Adicionalmente se pretende demostrar las potencialidades en cuanto a procesamiento de datos desde la nube, lo cual permite realizar un análisis de datos, cuyos resultados son de interés para los usuarios finales que administran operaciones o que toman decisiones en cuanto a cambios de comportamiento e incluso para automatizar parte de sus operaciones.

El proyecto entregará una plataforma base para el desarrollo de varias aplicaciones de IoT en distintos escenarios, entre las cuales se encuentran *Smart Home* (*HAN*) y*Safety/security*.

1.2 ORGANIZACIÓN DEL PROYECTO

El desarrollo del proyecto se presenta en seis capítulos, distribuidos de la siguiente manera:

En el **CAPITULO I** se realiza la presentación del proyecto, el cual consta de una introducción acerca del *Internet of things* mencionando sus antecedentes e

importancia relacionada con el presente trabajo. Adicionalmente se indica el contenido de los capítulos desarrollados.

En el **CAPITULO II** se redacta brevemente el estado del arte y marco conceptual del modelo de referencia del *Internet of things* y de las tecnologías implementadas en este proyecto.

En el **CAPITULO III** se describen los requerimientos del sistema, el diseño y la implementación de los prototipos basados en arquitecturas *open hardware* y comercial. Detallando las etapas de la configuración de dispositivos, programación de la interfaz y el procesamiento de los datos.Adicionalmente se incluye el desarrollo de una aplicación para Android.

En el **CAPITULO IV** se describen los requerimientos del sistema, el diseño y la implementación del prototipo basados en la plataforma de red IPv6. Se detallan la programación de la interfaz de visualización.

En el **CAPITULO V** se detalla la evaluación de las prestaciones de las arquitecturas Digi, Arduino e IPv6 en cuanto a coste, escalabilidad e interoperabilidad en base a pruebas y análisis de resultados del funcionamiento.Se incluye una comparación del modelo de referencia actual de las redes tradicionales con la arquitectura basada en la red nativa IPv6.

En el **CAPITULO VI** se establecen las conclusiones y recomendaciones del proyecto.

CAPÍTULO 2

ESTADO DEL ARTE Y MARCO CONCEPTUAL

2.1 CONCEPTOS BÁSICOS DE IoT.

2.1.1 Concepto de IoT.

En los últimos años, se han dado algunas definiciones que proporcionan una mejor visión de lo que es *Internet of things*, a continuación se citan algunas de ellas.

CASAGRAS define *Internet of things* como: “Una infraestructura de red global, que une los objetos físicos y virtuales a través de la explotación de la captura de datos y capacidades de comunicación. Esta infraestructura incluye desarrollos existentes y en evolución del internet, ofrecerá identificación específica de objetos, sensores y la capacidad de conexión como la base para el desarrollo de servicios cooperativos independientes y aplicaciones. Estos se caracterizan por un alto grado de captura de datos autónoma, la transferencia de eventos, la conectividad de red y la interoperabilidad “,(Furness).

El centro de investigación SAP define el *Internet of things* como “Un mundo en el que los objetos físicos están perfectamente integrados en la red de información, y donde los objetos físicos pueden llegar a ser participantes activos en los procesos de negocio. Los servicios están disponibles para interactuar con estos "objetos inteligentes" a través de Internet, consultar y modificar su estado, y toda la información asociada a ellos, teniendo en cuenta la seguridad y la privacidad. “(Ranz, 2013).

IoT se puede ver como una infraestructura global para la sociedad de la información, que permite servicios avanzados mediante la interconexión de cosas sobre la base de tecnologías de información y comunicación actuales y en evolución (TIC).

IoT hace pleno uso de *“cosas”*, objetos del mundo físico o del mundo de la información, para ofrecer servicios a todo tipo de aplicaciones a través de la explotación de capacidades de identificación, captura, procesamiento e integración a las redes de comunicación.

Las cosas físicas son capaces de ser detectadas, accionadas y conectadas, como por ejemplo robots industriales. Las cosas virtuales son capaces de ser almacenadas y procesadas, como por ejemplo contenidos multimedia y software de aplicación.

2.1.2 Descripción técnica del IoT.

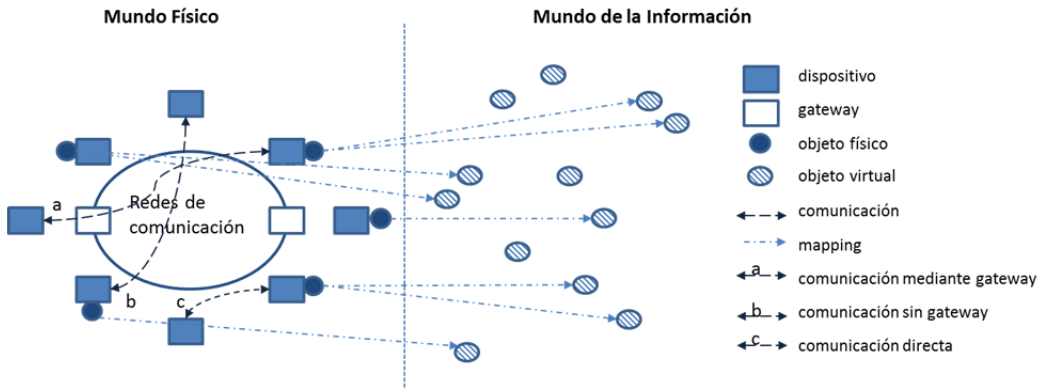

Figura 1. Descripción técnica del IoT.

Fuente:(International Communication Union ITU-T, 2012)

Un objeto físico puede estar representado en el mundo de la información a través de una o más cosas virtuales (mapping).

Un dispositivo tiene capacidades obligatorias de comunicación y capacidades opcionales de detección, captura, almacenamiento y procesamiento de datos. Recolectan diversos tipos de información y la envían a las redes de información y comunicación para su posterior procesamiento. Algunos también ejecutan operaciones.

La comunicación entre dispositivos puede ser: a través de la red de comunicación mediante un gateway (caso a), a través de la red de comunicación sin un gateway (caso b) o directamente, es decir sin necesidad de utilizar la red de comunicación (caso c).

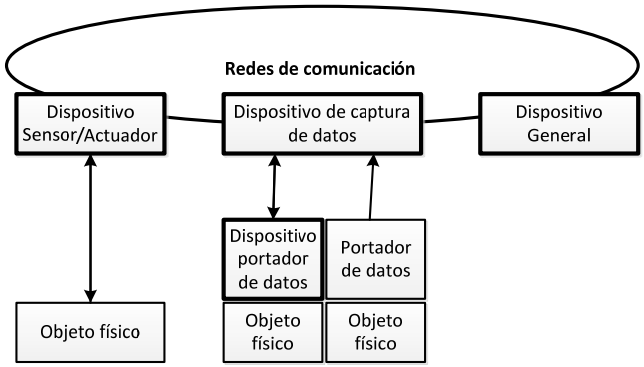

Figura 2. Tipos de dispositivos y su relación con los objetos físicos.

Fuente: (International Communication Union ITU-T, 2012)

Los dispositivos se clasifican en:

- **Dispositivo portador de datos:** Está unido a un objeto físico para conectar indirectamente lo físico con las redes de comunicación.

- **Dispositivos de captura de datos:** Se refiere a un dispositivo de lectura / escritura con la capacidad de interactuar con las cosas físicas.

- **Dispositivos de detección y accionamiento:** Puede detectar o medir la información relacionada con el entorno y convertirlo en señales electrónicas digitales. También puede convertir señales electrónicas digitales en operaciones.

- **Dispositivo general:** Incorpora capacidades de procesamiento y comunicación.

2.1.3 Características fundamentales.

Las características fundamentales de la IoT son las siguientes:

- **Interconexión:** Cualquier cosa puede ser interconectada con la infraestructura global de información y comunicación.

- **Servicios relacionados con objetos:** IoT es capaz de proporcionar servicios tales como protección de la privacidad, dentro de las limitaciones del objeto.

- **Heterogeneidad:** Los dispositivos están basados en diferentes plataformas de hardware y redes, y pueden interactuar con otros dispositivos o plataformas.

- **Los cambios dinámicos:** El estado de los dispositivos cambia de forma dinámica, por ejemplo, dormir y despertar, conectar y / o desconectar, ubicación y velocidad.

- **Gran escala:** El número de dispositivos que necesitan ser gestionados será mucho mayor que los dispositivos conectados a la Internet actual.

2.2 TECNOLOGÍAS DE COMUNICACIÓN.

2.2.1 Modelo de referencia del IoT.

El modelo de referencia del IoT está compuesto de cuatro capas:

- Capa de Aplicación.
- Capa de Servicio.
- Capa de Red.
- Capa del Dispositivo.

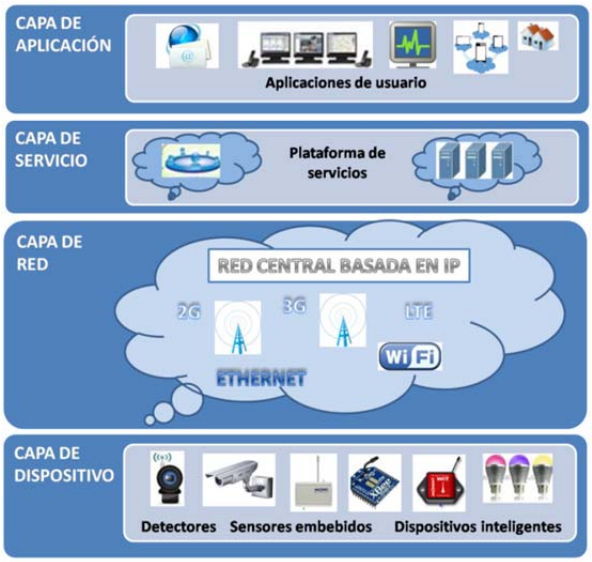

Figura 3. Modelo de Referencia de IoT.

Fuete: (Elaboración Propia)

Capa de Aplicaciones

Es el verdadero sistema de la aplicación, pone en uso la gran cantidad de información creada a partir del IoT y donde se encuentra el mayor potencial.

Capa de Servicio

Definida como computación en la nube, esta capa costa de dos grupos:

- **Función de soporte general:** estas son funciones comunes y pueden ser utilizados por diferentes aplicaciones, tales como procesamiento de datos o almacenamiento de datos. Estas funciones también pueden ser invocadas por las funciones de soporte específicas.

- **Función de soporte específico:** son las funciones especiales que se adaptan a los requisitos de las aplicaciones, con el fin de proporcionar diferentes funciones de apoyo a las diferentes aplicaciones del IoT.

Capa de Red

Es la infraestructura de redes alámbricas e inalámbricas, consta de las dos siguientes tipos de funciones:

- **Función de Red:** proporciona funciones de control, como el acceso y las funciones de control de recursos de transporte, la gestión de la movilidad o la autenticación, autorización y contabilidad (AAA).

- **Función de Transporte:** es responsable de proporcionar conectividad para el transporte del servicio del IoT y la información de datos específicos de la aplicación, así como el transporte de la información de la gestión de control relacionada con IoT.

Capa de Dispositivo

La capa de dispositivo conecta todo con el Internet y es la infraestructura clave para el IoT. Este consta de tecnología inalámbrica y solución/interfaz del sensor para recolectar y transmitir señales análogas/digitales al controlador principal. La tecnología inalámbrica está basada en IEEE 802.15.4 y sus protocolos, tales como ZigBee®, 6LoWPAN y WirelessHART. Utilizando una variedad de I/O y sensores, las señales pueden ser medidas en cualquier lugar por medio de redes inalámbricas, está capa realiza dos funciones:

- Función de dispositivo.
- Función de Gateway.

Las cuatro capas se encuentra en la capacidad de trabajar en función de administrar y dar seguridad, las cuales en conjunto forman el modelo de referencia del *Internet of Things*.

Capacidades de administración

De una manera similar a las redes de comunicación tradicionales, el IoT tiene la capacidad de evitar fallos debido a su configuración, contabilidad, rendimiento y seguridad (FCAPS), es decir, tiene la gestión de fallos, gestión de configuración, gestión contable, gestión del rendimiento y la gestión de la seguridad. Fuente:(International Communication Union ITU-T, 2012).

Función de Seguridad

Aquí se encuentran dos funciones de seguridad:

- Función de seguridad general.
- Función de seguridad específica.

2.2.2 Descripción de los fundamentos teóricos sobre el estándar 802.15.4

En el año 2000 dos grupos especialistas en estándares (ZigBee® y el grupo 15 de trabajo IEEE 802) se unieron para dar a conocer la necesidad de un

nuevo estándar, el cual fue creado con la finalidad de definir los niveles de red básicos para dar un servicio específico de red inalámbrica de área personal (WPAN) centrada en la comunicación entre dispositivos ubicuos con bajo coste y velocidad, que debido al concepto del IoT este estándar es apropiado para implementar en los prototipos.

Para aplicaciones que trabajan con el estándar 802.15.4 es necesario no manejar protocolos muy pesados debido a que esto afectaría al consumo de energía y si quiere una tendencia futura para poder interactuar entre objetos el ahorro de la energía es muy importante. En el siguiente cuadro se indica las propiedades del estándar:

Tabla 1. Características del estándar 802.15.4.

PROPIEDADES	RANGO
Transmisión de Datos	868 MHz: 20kb/s Europeo 915 MHz: 40kb/s Americano 2.4 GHz: 250kb/s. A nivel mundial
Alcance	10-20m
Latencia	< 15ms
Canales	868/915 MHz: 11 canales. 2.4 GHz: 16 canales.
Bandas de Frecuencia	Dos PHY: 868/915 MHz y 2.4 GHz.
Direccionamiento	Cortos de 8 bits o 64 bits IEEE
Canal de acceso	CSMA-CA
Temperatura	-40° a +85° C

Fuente:(El Estandar IEEE 802.15.4)

2.2.3 Arquitectura del Estándar 802.15.4.

Como todos los estándares este también se basa en el modelo de referencia OSI con algunas modificaciones.

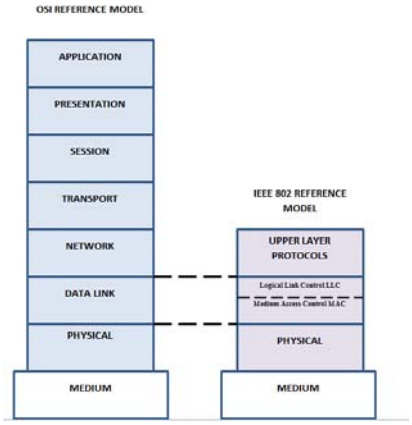

Figura 4.Modelo IEEE 802 frente al modelo OSI.

Fuente: (NewsUniversidad)

Capa Física (PHY):

Es la capa de red más básica, proporcionando únicamente los medios para transmitir bit a bit sobre un enlace de datos conectado a nodos de red. El estándar IEEE 802.15.4 ofrece dos opciones de nivel físico (PHY) que se combinan con el Control de Acceso al Medio (MAC) para permitir un amplio rango de aplicaciones en red.

- La PHY de los 2.4 GHz, específica operación en la banda industrial, médica y científica (ISM): se puede utilizar para conseguir salidas superiores y de poca latencia.

- La PHY de los 868/915 MHz especifica operaciones en Europa y Estados Unidos respectivamente: se utilizan para lograr mayor sensibilidad y mayores áreas de cobertura, con lo que se reduce el número de nodos requeridos para cubrir un área.

Capa de Enlace de Datos (DLL):

Esta capa se divide en dos subcapas la capa de enlace de acceso a medios (MAC) que depende del hardware y de la implementación, y la capa de control de enlaces lógicos (LLC) esta capa es muy común en todas las arquitecturas.

Capa de Red

La capa de red es la responsable por la topología de construcción (entre ellas topología tipo estrella y topología peer to peer) y mantenimiento de la misma, también se encarga de los servicios de mantenimiento y seguridad.

2.3 ESTANDAR ZIGBEE.

El estándar ZigBee® amplía el estándar IEEE 802.15.4 aportando una capa de red (NWK) que gestiona las tareas de enrutado y de mantenimiento de los nodos de la red; y un entorno de aplicación que proporciona una subcapa de aplicación (APS) que establece una interfaz para la capa de red.

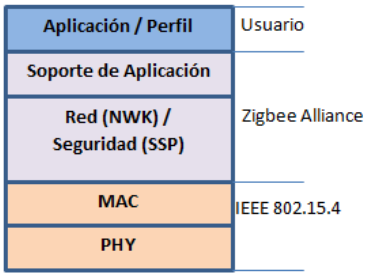

Figura 5. Pila de protocolos para Zigbee®.

Fuente: (Jose Suarez)

2.3.1 Características del estándar Zigbee®.

En el siguiente cuadro se observa características del estándar Zigbee®:

Tabla 2. Característica de la tecnología Zigbee®.

PROPIEDAD	RANGO
Bandas de Transmisión de Datos	868 MHz: 20kb/s Europeo 915 MHz: 40kb/s Americano 2.4 GHz: 250kb/s. A nivel mundial
Velocidad de Transmisión	250 kbps
Cobertura	10 a 75m
Desempeño	A pesar de coexistir en la misma frecuencia con otro tipo de redes como WiFi o Bluetooth su desempeño no se ve afectado, esto debido a su baja tasa de transmisión y, a características propias del estándar IEEE 802.15.4
Capacidad	Redes de gran densidad, aumenta la confiabilidad de la comunicación.
Protocolo de comunicación multi-salto	puede establecer comunicación entre dos nodos aun cuando estos se encuentren fuera del rango de transmisión
Topología	Tipo malla (Mesh)

Fuente:(Carlos, 2007)

2.3.1 Tipos de Dispositivos.

Se encuentran tres tipos diferentes de dispositivos ZigBee®:

- **Coordinador ZigBee® (ZC):** Puede actuar como director de una red en árbol así como servir de enlace a otras redes. Existe exactamente un coordinador por cada red y puede almacenar información sobre la red.

- **Router ZigBee® (ZR):** Además de ofrecer un nivel de aplicación para la ejecución de código de usuario, puede actuar como router interconectando dispositivos separados en la topología de la red.

- **Dispositivo final (ZED):** Posee la funcionalidad necesaria para comunicarse con su nodo coordinador o un routeador, pero no puede transmitir información a otro ZED. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de sus baterías.

2.4 PLATAFORMA ARDUINO.

Arduino es una plataforma*open source* compuesta por una placa con un micro controlador Atmegaxx8, el cual permite conectar sensores y actuadores mediante entradas y salidas analógicas y digitales, el micro controlador se programa utilizando un lenguaje propio de arduino basado en C y un entorno de desarrollo integrado gratuito (IDE) nativo creado en Java.

Está pensada para el uso de artistas, diseñadores, aficionados y cualquier interesado en crear objetos o ambientes interactivos desarrollando múltiples diseños autónomos.

La plataforma tiene dos componentes:

- **Hardware:** La plataforma Arduino es una plataforma *open hardware* basada en una sencilla placa con entradas y salidas (E/S), analógicas y digitales. Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta.

- **Software:** Está basado en un Entorno de Desarrollo Integrado (EDI) el cual permite escribir, compilar programas (llamados Sketches) y cargarlos al hardware.

Ventajas de la plataforma Arduino.

- Las placas son más asequibles en comparación con otras alternativas tradicionales (BASIC Stamp).

- Es multiplataforma el software puede trabajar en Windows, Mac OSX y Linux.

- El entorno de programación es fácil de usar tanto para aficionados como para estudiantes, es más fácil que programar en C y en ensamblador.

- El software es ampliable gracias a las librerías y debido a que es de código abierto.

- El hardware también es ampliable debido a que es *open source*.

- Permite crear objetos que interactúen con el entorno, como sensores y actuadores.

2.4.2 Tarjeta Arduino UNO R3.

Lo primero que se necesita es una placa Arduino. Existen varios modelos e incluso se puede construir una placa propia. La placa Arduino es "*open hardware*", lo que quiere decir que su diseño es de libre distribución y utilización.

Figura 6.Tarjeta Arduino UNO Rev 3.

Fuente: (ARDUINO, 2013)

La nueva versión de Arduino UNO: R3. Esta tarjeta incorpora nuevas características como el empleo de un ATmega16U2 en lugar del 8U2. Esto permite tasas de transferencia más altas y brinda más memoria. No se requieren drivers para instalarlo bajo Linux o Mac (La versión de Windows incluye los drivers). Con el 16U2 te permite que el UNO sea reconocido como un teclado, mouse, joystick o cualquier otro periférico USB

Características:

- Micro controlador ATmega328.
- Voltaje de entrada: 7 a 12V.
- 14 pines digitales I/O (6 salidas PWM).
- 6 entradas analógicas.
- 32k de memoria flash.
- Velocidad de reloj 16MHz.

2.4.3 Arduino Shields.

Los Shields o tarjetas de expansión son módulos fabricados por terceros que se pueden apilar encima de la placa Arduino y le proporcionan una funcionalidad determinada, por ejemplo conexiones inalámbricas, control de sensores o motores, lectura y escritura de memorias.

2.4.4 Arduino Xbee Shield

La Xbee shield permite a una placa Arduino comunicarse de forma inalámbrica usando Zigbee®. El módulo puede comunicarse hasta 30 metros en interior y 90 metros al aire libre (en línea de vista). Puede ser usado como reemplazo del puerto serie/usb o en modo de comandos y configurarlo para una variedad de opciones de redes broadcast o malladas.

También provee conectores hembra para usar los pines digitales desde 2 hasta 7 y las entradas analógicas, las cuales están cubiertas por la shield (los pines digitales de 8 a 13 no están cubiertos por la placa, así que pueden usar los conectores de la placa directamente).

Figura 7.MóduloXbee *Shield*.

Fuente:(BRICO GEEK , 2013)

La Arduino Ethernet *Shield* permite a una placa Arduino conectarse a internet. Está basada en el chip Ethernet Wiznet W5100. Este chip provee de una pila de red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de

sockets simultáneas, provee un conectorEthernet estándar RJ45. Usa la librería Ethernet para escribir programas que se conecten a internet usando la shield.

El botón de reset en la *shield* resetea ambos, el W5100 y la placa Arduino, la *shield* contiene un número de LEDs los cuales sirven para información.

Figura 8. Arduino Ethernet *Shield*.

Fuente: (Valenzuela, 2012)

2.5 PLATAFORMA COMERCIAL (DIGI).

2.5.1 Digi Internacional.

Digi International proporciona una gama de productos inalámbricos, una plataforma de *Cloud computing* para dispositivos, y el desarrollo de servicios.

Digi utiliza la Nube de dispositivos Etherios, una plataforma de nube pública como servicio (PaaS), que proporciona una integración de aplicaciones con

redes de dispositivos. Suministra mensajería segura, almacenamiento de datos y la administración de dispositivos.

2.5.2 Digi y Tecnología Zigbee®.

Digi es miembro de la Alianza ZigBee® y ha desarrollado una amplia gama de soluciones de redes basadas en el protocolo ZigBee®. Los productos XBee son una serie de productos modulares de radio frecuencia diseñados para operar dentro del estándar IEEE 802.15.4 en la banda 2.4Ghz ISM.

Características

Su principal objetivo son las aplicaciones que requieren comunicaciones seguras con una baja tasa de transferencia de datos.

Los módulos Xbee proveen 2 formas de comunicación: Transmisión serial transparente (modo AT) y el modo API. En el modo AT, los datos que ingresan se transmiten directamente sin ninguna modificación.En el modo API, los datos se envuelven en una estructura de paquetes que permite direccionamiento. Los módulos pueden ser configurados utilizando el programa X-CTU o desde un microcontrolador.

Existen varios tipos de módulos XBee, se diferencian por el tipo de antena (chip, alambre o conector SMA) y la potencia de transmisión (2mW para 300 pies o 60mW para hasta 1 milla), entre otras. Permiten realizar aplicaciones de redes punto a punto, punto a multipunto y redes tipo malla.

Existen 2 series de estos módulos. La serie 1 y la serie 2 o 2.5. Los módulos de la Serie 1 y la Serie 2 tienen el mismo *pin-out*, sin embargo, no son compatibles entre sí. Para algunas familias, el módulo está disponible en una versión de bajo consumo de energía y en una versión PRO que tiene a menudo un amplificador adicional para un mayor alcance.

Para el desarrollo del presente proyecto se utilizan módulos XBEE series 2, un Gateway ConnectPort X4 y un Wall Router. Los módulos XBEE son usados con adaptadores Xbee Explorer Serial, Xbee Explorer USB y Xbee Adapter DIO.

XBEE series 2

Figura 9. Módulo XBEE series 2 con antena de alambre.

Fuente: (BricoGeek, 2005).

Tabla 3. Características de los módulos XBEE series 2.

Propiedades	XBEE series 2
Alcance máximo	120 metros en exteriores 40 metros en interiores
Potencia de salida	2mW (3dBm)
Sensibilidad	-96 dBm
Alimentación	3.3 VDC
Frecuencia de trabajo	2.4 GHz
Máxima tasa de datos	250 Kbps
Tipo de Antena	de alambre flexible
Modo de trabajo	AT y API

Fuente: (Ingeniería MCI Ltda., 2009)

Descripción de pines.

Tabla 4. Asignación de pines de los módulos XBEE.

Pin #	Nombre	Dirección	Descripción
1	VCC	-	Fuente de alimentación
2	DOUT	Salida	Salida de datos UART
3	DIN/ $\overline{CONFIG}$	Entrada	Entrada de datos UART
4	DIO12	Las dos	I/O Digital 12
5	$\overline{RESET}$	Las dos	Reset del módulo (el pulso debe ser al menos de 200ns)
6	PWM0/ RSSI/ DIO10	Las dos	Salida PWM 0/ RSSI/ IO Digital
7	PWM/ DIO11	Las dos	
8	[reservado]	-	No se conecta
9	$\overline{DTR}$ / SLEEP_RQ/ DIO8	Las dos	Pin de control de Sleep o IO Digital 8
10	GND	-	Tierra
11	DIO4	Las dos	IO Digital 4
12	$\overline{CTS}$ / DIO7	Las dos	Control de flujo despejado para el envío, IO Digital 7
13	ON/ $\overline{SLEEP}$ / DIO9	Salida	Indicador del estado del módulo, IO Digital 9
14	[reservado]	-	No se conecta

Pin #	Nombre	Dirección	Descripción
15	Associate/ DIO5	Las dos	Indicador de asociación, IO Digital 5
16	$\overline{RTS}$ / DIO6	Las dos	Control de flujo pedido de envío, IO Digital 6
17	AD3/ DIO3	Las dos	Entrada analógica 3, IO Digital 3
18	AD2/ DIO2	Las dos	Entrada analógica 2, IO Digital 2
19	AD1/ DIO1	Las dos	Entrada analógica 1, IO Digital 1
20	AD0/ DIO0/ Commissioning Button	Las dos	Entrada analógica 0, IO Digital 0, o Botón de Puesta en marcha.

Fuente:(DigiInternational,Inc., 2007).

XBEE Explorer Serial y USB

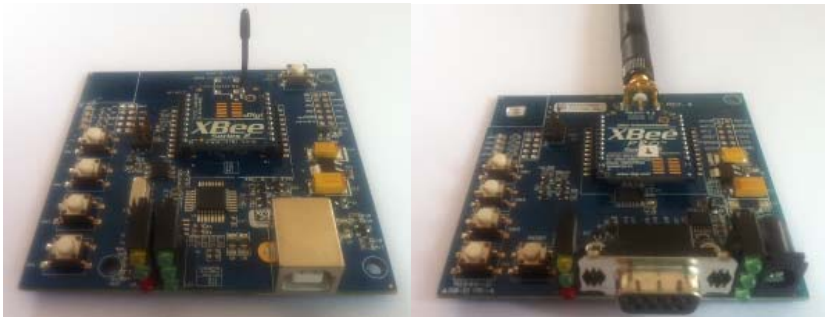

Figura 10. XBEE Explorer USB a la izquierda y XBEE Explorer Serial a la derecha.

Fuente: (Elaboración Propia)

Son interfaces de desarrollo, una de ellas con conexión RS232 y la otra con conexión USB para la línea Xbee. Al conectar el módulo a la tarjeta se puede tener acceso directo a los pines de programación y serial de la unidad Xbee.

ConnectPort X4

Figura 11. Gateway ConnectPort X4.

Fuente:(Zigbee labs, 2012).

El ConnectPort X4 tiene una funcionalidad de puerta de enlace entre las distintas tecnologías de red como Ethernet y XBee.

Características:

- Funciona como coordinador en la red Zigbee®.
- Tiene interfaces Ethernet, serie, USB.
- Protocolos de red: UDP/TCP, DHCP, SNMPv1/v2..
- Soporta el lenguaje de programación Python.
- Posee una interfaz web para configuración, monitoreo y administración de dispositivos Digi.

Adaptador XBEE DIO.

Figura 12. XBEE Adapter DIO.

Fuente:(International., 2013).

El adaptador XBee Digital I / O proporciona conectividad inalámbrica de corto alcance a cualquier dispositivo digital. A diferencia de un módulo inalámbrico incorporado, que requiere la integración de diseño y tiempo de desarrollo, estos adaptadores *off-the-shelf* proporcionan conectividad inalámbrica instantánea a los dispositivos digitales existentes.

Tabla 5. Configuración Pines XBEE DIO.

PIN	FUNCIÓN
1	Estos pines pueden ser configurados como entradas o salidas digitales
2	
3	
4	
5	GND
6	+12VDC 50mA max

Fuente:(International., 2013)

XBEE Wall Router

Figura 13. Xbee Wall Router.

Fuente: (Digi International., 2012).

Es un dispositivo final el cual posee sensores de temperatura y luminosidad embebidos, se conecta al tomacorriente. Se usa en conjunto con otros dispositivos Digi, es conecta a la red mediante la dirección MAC la cual es descubierta por su nodo coordinador enviando datos del ambiente en el que se encuentra.

2.5.3 Software X-CTU.

X-CTU es un programa gratuito desarrollado por Digi para la configuración y manejo de dispositivos XBee. Con este software se pueden configurar los dispositivos con los parámetros de red adecuados y realizar pruebas de cobertura. El programa funciona bajo Windows y permite configurar los

dispositivos de dos formas: mediante la introducción de comandos AT en la pestaña Terminal y una configuración más rápida y sencilla en la pestaña *Modem Configuration*, que enlista los parámetros a configurar.

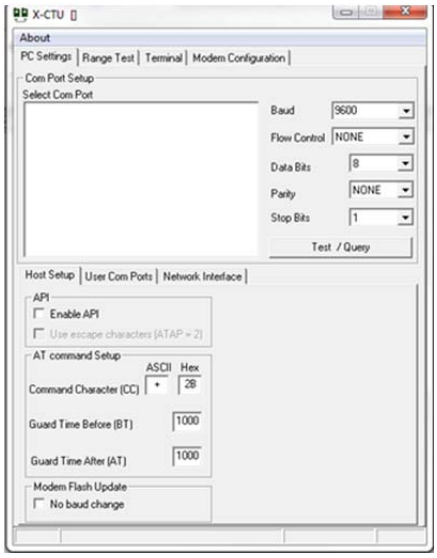

Figura 14. Interfaz software X-CTU.

Fuente: (Elaboración Propia)

2.5.4 Otras Tecnologías.

Todos los sensores de IoT requieren algunos medios para transmitir datos al mundo exterior. Para enlaces de corto alcance o de área local, las tecnologías inalámbricas disponibles son: RFID, NFC, Wi-Fi, Bluetooth, Zigbee®, Z-Wave y Wireless M-Bus. Los enlaces también pueden ser por cable, incluyendo Ethernet, HomePlug, HomePNA, HomeGrid y LonWorks. Para enlaces de larga distancia o de área amplia, existen las redes móviles, mediante GSM, GPRS, 3G, LTE o WiMAX y las conexiones por satélite.

WIFI

Es una tecnología popular que permite a un dispositivo electrónico el intercambio de datos de forma inalámbrica. La Wi-Fi Alliance define Wi-Fi como cualquier producto de red de área local inalámbrica (WLAN) que se basa en los estándares IEEE 802.11.

La visión de IoT requiere conectividad a los pequeños dispositivos como varios tipos de sensores y actuadores, que están obligados a mantener un funcionamiento fiable durante años. El estándar IEEE 802.11 se ha establecido como una de las tecnologías inalámbricas más populares que ofrecen conectividad. Se ha demostrado la viabilidad de la tecnología Wi-Fi de baja potencia para permitir la conectividad IP de los dispositivos alimentados por baterías.

Tabla 6. Tecnologías de comunicación.

	NFC	RFID	Blue tooth	Wi-Fi	ZigBee®	6LoW PAN	WiMAX	2.5-3.5 G
Red	PAN	PAN	PAN	LAN	LAN	LAN	MAN	WAN
Topología	P2P	P2P	Estrella	Estrella	Malla, Estrella, Árbol	Malla, Estrella	Malla	Malla
Potencia	Muy baja	Muy baja	Baja	Baja – Alta	Muy baja	Muy baja	Alta	Alta
Velocidad	400 Kbps	400 Kbps	700 Kbps	11 – 100 Mbps	250 Kbps	250 Kbps	11 – 100 Mbps	1.8 - 7.2 Mbps
Rango	<10 cm	<3 m	<30 m	4 – 20 m	10 – 300 m	800 m (Sub-GHz)	50 km	Red celular
Aplicación	Obtener acceso, compartir, iniciar servicio	Seguimiento de elemen-tos	Intercam-bio de datos	Internet, multime dia	Redes de sensores	Redes de sensores	Internet banda ancha para área metropolitana	Teléfonos celulares, telemetría

Fuente: (Karimi & Atkinson, 2013)

2.6 DESCRIPCIÓN DEL PROTOCOLO IPv6.

2.6.1 Introducción IPv6.

IP versión 6 (IPv6) o IPng (*Next Generation Internet Protocol*)es la nueva versión del protocolo de redes de datos, en los que Internet está basado, diseñado por el IETF como el sucesor de la versión 4 (IPv4). En esta versión se mantuvieron las funciones del IPv4 que son utilizadas, las que no son utilizadas o se usan con poca frecuencia, se quitaron o se hicieron opcionales, agregándose nuevas características.

Los cambios de IPv4 a IPv6 recaen principalmente en las siguientes categorías:

- **Mayor capacidad de direccionamiento:** IPv6 aumenta el tamaño de la dirección IP de 32 bits a 128 bits, para soportar más niveles de jerarquía de direccionamiento, un número mucho mayor de nodos direccionables, y configuración automática simple.

- **Simplificación del formato de cabecera:** Algunos campos de la cabecera IPv4 se han eliminado o hecho opcionales, para reducir el costo de procesamiento de paquetes.

- Seguridad en el núcleo del protocolo (IPsec). El soporte de IPsec es un requerimiento del protocolo IPv6.

- **Capacidad de Etiquetas de flujo:** Se añade una nueva capacidad para asignar etiquetas a los paquetes que pertenecen a tipos de tráfico particulares, "flujos".

- **Autoconfiguración:** la autoconfiguración de direcciones es más simple.

- **Renumeración y *multihoming*:** facilitando el cambio de proveedor de servicios.

- Características de movilidad, la posibilidad de que un nodo mantenga la misma dirección IP, a pesar de su movilidad.

- Calidad de servicio (QoS) y clase de servicio (CoS).

- **Capacidades de Autenticación y privacidad:** Para IPv6 se especifican extensiones para soportar autenticación, integridad de datos y confidencialidad de los datos.

2.6.2 Mecanismos de Transición.

IPv6 es un protocolo nuevo que no es compatible con IPv4, por lo tanto en su diseño de se ha tomado en cuenta un largo período de transición y co-existencia entre ambos. Los mecanismos de transición son un conjunto de mecanismos y de protocolos implementados en *hosts* y *routers* IPv6, que necesitan interoperar con *hosts* IPv4.

Los mecanismos de transición se dividen en 3 grupos y son los siguientes:

Dual Stack

Se trata de mantener simultáneamente en un dispositivo, tanto al *stack* (pila) del protocolo IPv4, como al de IPv6. De esta manera, dependiendo de la pila que tenga implementada el nodo al cual se quiere comunicar, si es IPv4 se utilizará la conectividad IPv4 o si es IPv6 se utilizará la red IPv6.

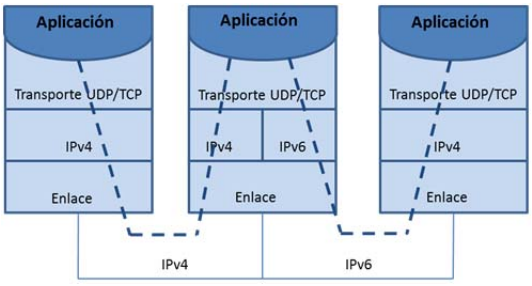

Figura 15. Mecanismo *Dual Stack*.

Fuente: (Elaboración Propia)

Túneles

Este mecanismo permite atravesar redes que no tienen soporte nativo del protocolo que se está utilizando. Los paquetes originales son transportados hasta un punto de la red por medio del protocolo original, posteriormente, un túnel encapsula el contenido de los paquetes IPv6 dentro de paquetes IPv4, para que los mismos puedan viajar por estas redes. Los paquetes son "des encapsulados" al llegar al otro extremo para ser enviados al destino final, un nodo IPv6 o *dual stack*, en forma nativa.

Los tipos de túneles más utilizados son los configurados o manuales y los automáticos.

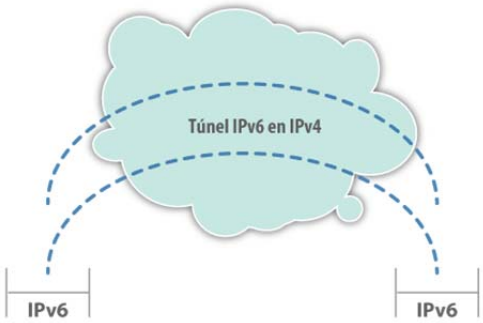

Figura 16. Mecanismo Túneles.

Fuente:(Cicileo).

Traducción

En el mecanismo de traducción, se utiliza algún dispositivo en la red para modificar la cabecera IPv4 a una cabecera IPv6 y viceversa, realizando conversiones de direcciones, de API de programación, o actuando en el intercambio de tráfico TCP a UDP. Permite un enrutamiento transparente de la comunicación entre nodos que sólo posean soporte a una versión del protocolo IP, o que utilizan Doble Pila.

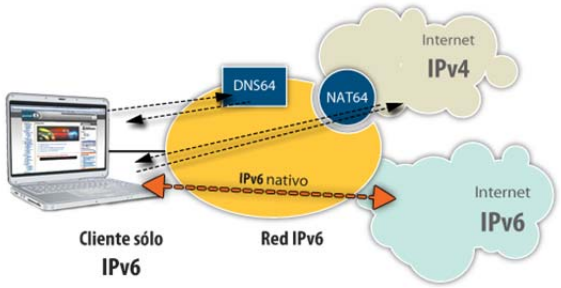

Figura 17. Mecanismo de traducción.

Fuente:(Cicileo).

2.6.3 IoT e IPv6.

La capacidad para identificar de forma única “cosas” es crítico para el éxito de IoT. Esto no sólo permitirá identificar miles de millones de dispositivos, sino también controlar dispositivos remotos a través de Internet. Cada elemento que ya está conectado y los que se van a conectar, deben estar identificados por su

identificación única, ubicación y funcionalidades. El IPv4 actual puede soportar hasta cierto alcance, identificando un grupo de dispositivos sensores geográficamente, pero no individualmente. Los atributos de movilidad de Internet en IPv6 pueden aliviar algunos de los problemas de identificación de dispositivos.

Otro aspecto de IoT es el funcionamiento persistente de la red, para canalizar el tráfico de datos de forma ubicua y sin descanso. Aunque, el protocolo TCP / IP se encarga de este mecanismo mediante el enrutamiento de una manera más fiable y eficiente, desde el origen al destino, IoT se enfrenta a un cuello de botella en la interfaz entre el gateway y los dispositivos de sensores inalámbricos. Además, la escalabilidad de la dirección del dispositivo de la red existente debe ser sostenible.

2.6.3. Consorcio Ecuatoriano para el desarrollo de Internet Avanzado (CEDIA).

El protocolo de nueva generación, IPv6, tuvo un rápido despliegue en el sector académico científico, con el fin de, por un lado la experimentación e investigación y por otro la formación de recursos humanos en el tema. Este sector se ve beneficiado con características disponibles en este protocolo. Son algunos ejemplos:

- La necesidad de contar con direcciones públicamente alcanzables, que permitan la interacción entre pares (aplicaciones "*peer to peer*" como videoconferencia, operación remota de instrumentos, *grids*, etc.)

- Características como *multicast*.

- Disponibilidad de IPSec como parte del *stack*.

Por todas estas razones, en particular en la región de Latinoamérica y Caribe, las NRENs y sus instituciones miembro han estado utilizando el protocolo desde hace años. Cabe destacar la experiencia disponible en Red CLARA, en donde actualmente se encuentra disponible IPv6 en forma nativa.

CEDIA es el Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado y es parte de Red CLARA, lo integran las Universidades e Instituciones de Investigación y Desarrollo de Ecuador, incluida la Universidad de las Fuerzas Armadas ESPE. Fue creada para estimular, promover y coordinar con el Proyecto Redes Avanzadas.

2.7 DESCRIPCIÓN DEL MODELO CLOUD COMPUTING.

2.7.1 Concepto *Cloud Computing.*

El modelo de *Cloud computing* o computación en nube permite el acceso a la información y los recursos informáticos desde cualquier lugar, con una conexión de red disponible. Proporciona un conjunto compartido de recursos, incluido el espacio de almacenamiento de datos, redes, potencia de procesamiento informático y aplicaciones corporativas y de usuario especializadas.

Modelos de servicio

Los modelos de servicio de *Cloud computing* son:

- **Software como servicio:** Un proveedor de SaaS ofrece a los suscriptores acceso a los recursos y aplicaciones, hace innecesario tener una copia física de software para instalar en los dispositivos. Se tiene menos control sobre la nube.

- **Plataforma como servicio:** Un proveedor de PaaS ofrece acceso a los componentes que se requieren para desarrollar y operar aplicaciones a través de Internet.

- **Infraestructura como Servicio:** Un servicio IaaS, se ocupa principalmente de la infraestructura computacional. En IaaS el abonado externaliza completamente los recursos, tales como hardware, software y almacenamiento necesarios.

Figura 18. Modelos de Servicio de *Cloud Computing*.

Fuente: (Elaboración Propia)

Despliegue de servicios en la nube

Los servicios en la nube por lo general se ponen a disposición a través de una nube privada, nube comunitaria, nube pública o nube híbrida.

- **Nube Pública:** Todos los abonados con conexión a internet pueden acceder a la nube. Tanto los datos como los procesos de varios clientes se mezclan en los servidores, sistemas de almacenamiento y otras infraestructuras de la nube.

- **Nube Privada:** La infraestructura de la nube se utiliza exclusivamente para una organización específica, y es administrado por la organización o por un tercero. Posee alta protección de datos y ediciones a nivel de servicio.

- **Nube Comunitaria:** El servicio es compartido por varias organizaciones que tienen requerimientos similares y estará disponible únicamente a esos grupos. La infraestructura puede ser operada por las organizaciones o por un proveedor de servicios cloud.

- **Nube Híbrida:** Una nube híbrida es esencialmente una combinación de al menos dos nubes, donde las nubes son una mezcla de públicas, privadas o comunitarias.

2.7.2 El *Cloud computing* y el IoT.

El *Cloud Computing* ofrece grandes beneficios para las aplicaciones alojadas en la web, que también tienen requerimientos de cómputo y almacenamiento especiales. Además, ofrece maneras fáciles de acceder a ellos a través de redes y permite a los usuarios centrarse en el desarrollo de las aplicaciones. Los usuarios sólo tienen que configurar sus sensores para enviar datos a través de las plataformas IoT.

El *Cloud computing* puede proporcionar la infraestructura virtual que integra dispositivos de control, de almacenamiento, herramientas de análisis, plataformas de visualización y la entrega del cliente. Esta plataforma actúa como un receptor de datos de los sensores ubicuos, para analizar e interpretar los datos, así como proporcionar al usuario una visualización fácil de entender basada en la web, características requeridas para la generación de aplicaciones de *Internet of things*.

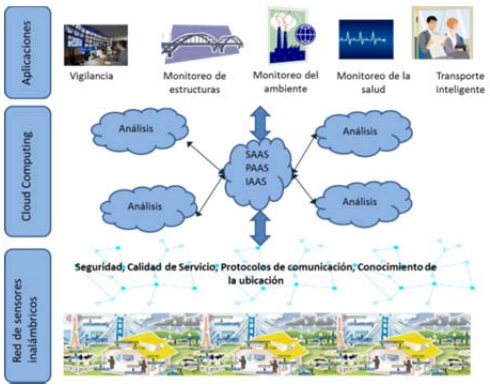

Figura 19. Marco conceptual de IoT con *Cloud Computing* en el centro. Fuente: (Elaboración Propia)

2.7.3 Comunicación con la nube mediante Servicios Web.

Actualmente, la forma más abierta e interoperable para proporcionar acceso a servicios remotos y permitir que las aplicaciones se comuniquen entre sí es utilizar Servicios Web.

Las ventajas de los Servicios Web son:

- Los servicios Web son independientes de la plataforma y del lenguaje, ya que utilizan lenguajes XML estándar.

- La mayoría de Servicios Web utilizan HTTP para transmitir los mensajes.

2.7.4 Servicios de *Cloud Computing* para la gestión de sensores.

Las aplicaciones más populares basadas en la nube, que ofrecen servicios dedicados para la gestión de datos de los sensores en línea son:

- **Cosm:** Es una plataforma de datos en tiempo real para el *Internet of things*, permite a los usuarios crear productos y servicios, almacenar, compartir y descubrir datos de los sensores, la energía, el medio ambiente, objetos y dispositivos en todo el mundo. Se basa en una API abierta y de fácil acceso y tiene un sitio web muy interactivo para la gestión de datos de los sensores.

- **Nimbits:** Es un servicio de procesamiento de datos que se puede utilizar para almacenar y compartir datos de sensores en la nube. Es una plataforma libre y de código abierto para el *Internet of things.* Permite la creación de *Data points* en la nube con los que se pueden realizar

cálculos, generar alertas, transmitir datos a las redes sociales, conectar a diagramas de control de procesos y hojas de cálculo.

- **ThingSpeak:** ThingSpeakesotraaplicación*open source*para el *Internet of things*. Los usuarios pueden crear aplicaciones de registro de sensores, seguimiento de ubicación. Además permite el procesamiento de datos numéricos tales como escalado en el tiempo, promedio, mediana, sumatoria, y redondeo.

- **Etherios Device Cloud:** La plataforma Digi es del tipo Plataforma como servicio (PaaS). La plataforma Digi gestiona la comunicación entre las aplicaciones empresariales y los dispositivos activos remotos, independientemente de su ubicación o de la red. Hace que la conexión de dispositivos remotos sea sencilla, proporcionando todas las herramientas para conectar, administrar, almacenar y transferir información a través de la red.

Lista de plataformas disponibles

Tabla 7. Plataformas *Cloud* para IoT.

Nombre	Características	Free | Open Source	URL
Cosm	Visualizar y almacenar datos de sensores, en línea.	SI | NO	http://www.cosm.com
Nimbits	Registro de datos en la nube.	SI | SI	http://www.nimbits.com
ThingSpeak	Visualizar y almacenar datos de sensores, en línea.	SI | SI	https://www.thingspeak.com
Etherios	Plataforma Device Cloud	SI | NO	http://www.etherios.com
SensorCloud	Visualizar y almacenar datos de sensores, en línea.	SI | NO	http://www.sensorcloud.com
Open.Sen.Se	Plataforma Internet of Everything	SI | NO	http://www.open.sen.se
Exosite	Plataforma para la gestión de dispositivos y datos basada en la nube.	SI | NO	http://www.exosite.com
Manybots	Recolectar y gestionar información de varios dispositivos.	SI | NO	http://www.manybots.com

Fuente:(Doukas, 2012)

2.8 FUNDAMENTOS BÁSICOS DEL PROCESAMIENTO DE SEÑALES.

Todo lo que lleva la información es una señal, es la descripción formal de un fenómeno en evolución en el tiempo o el espacio. Por procesamiento se entiende operar de alguna manera en una señal de forma manual o mecánica para modificar, analizar, manipular esta señal y extraer información útil.

2.8.1 Procesamiento de señales analógicas y digitales.

Las operaciones de procesamiento de señales implicadas en muchas aplicaciones como los sistemas de comunicación, sistemas de control, instrumentación, procesamiento de señales biomédicas, etc, se pueden implementar de dos maneras diferentes:

- método analógico o continuo en el tiempo
- método digital o discreto en el tiempo.

El procesamiento de la señal analógica utiliza elementos de circuitos analógicos, tales como resistencias, condensadores, transistores, diodos, etc. Se basa en la capacidad del sistema analógico para resolver ecuaciones diferenciales que describen un sistema físico. En contraste con el procesamiento de señal digital se basa en cálculos numéricos.

2.8.2 Métodos matemáticos aplicados en el procesamiento de señales.

A continuación las herramientas matemáticas utilizadas en procesamiento de señales.

- Señales y sistemas lineales, y la teoría de la transformada
- Ecuaciones diferenciales

- Espacios vectoriales y álgebra lineal
- Probabilidad y procesos estocásticos
- Teoría de Detección
- Teoría de Estimación
- Optimización
- Métodos Numéricos
- Los métodos iterativos

Magnitudes estadísticas para estimar el valor más representativo

Media: Representa el valor que mejor representa al grupo de medida. Se calcula como,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Ecuación 1. Cálculo de Valor medio.

Mediana: de un conjunto de valores es el valor de la muestra que tiene tantos valores del conjunto por encima como por debajo de ella.

Moda: es el valor de la muestra que se presenta con mayor frecuencia, lo que representa el valor más común en la medida.

Magnitudes estadísticas para estimar la dispersión de valores

La dispersión de los valores de medida respecto del valor medio, se expresa con:

Rango: de un conjunto de datos es la diferencia entre el mayor y el menor de ellos.

Varianza: es una medida de dispersión. La varianza es la media aritmética de los cuadrados de las diferencias de los valores individuales de la media. La elevación al cuadrado asegura que los valores positivos y negativos no se anulan mutuamente.

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2$$

Ecuación 2.Cálculo de la Varianza de una muestra.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu\right)^2$$

Ecuación 3. Cálculo de la Varianza de una población.

Desviación media: es la media de las desviaciones de las diferentes medidas respecto del valor medio.

$$\overline{D} = \frac{1}{N}\sum |d_i| = \frac{\sum |q_i - \overline{q}|}{n}$$

Ecuación 4.Cálculo de la Desviación media.

Desviación estándar experimental: representa la dispersión absoluta de los resultados de una medida obtenida a partir de una serie de mediciones de una misma magnitud.

2.8.3 Procesamiento de datos en IoT.

En el *Internet of things* (IoT), la cantidad de aplicaciones y nodos terminales con conexión de red hacen el procesamiento de los datos extremadamente difícil. Se propone una arquitectura combinada con el *Cloud computing*. Un concepto que hace uso de Codificación y Modulación Adaptable y Computación Granular, que pretende liberar la presión de procesamiento de datos.

Un procesamiento de datos exige un análisis continuo, ya que los orígenes de datos generan datos nuevos sin cesar. Y muchos escenarios tienen que identificar y reaccionar rápidamente frente a unas condiciones que solo se hacen evidentes con el análisis de los datos entrantes, de modo que requieren de un análisis con una latencia baja y con resultados prácticamente instantáneos. Por lo tanto el *Internet of things* realiza un procesamiento controlado por eventos y permite realizar un análisis sofisticado y basado en el tiempo. De hecho, es la tendencia hacia la computación ubicua y

omnipresente, que es la mayor fuerza impulsora hacia análisis de datos grandes.

2.9 SENSORES Y ACTUADORES.

Un sensor es un dispositivo capaz de convertir magnitudes del exterior tanto físicas o químicas (presión, temperatura, velocidad, fuerza, humedad, etc) en variables eléctricas que se pueda cuantificar y manipular. Los sensores se clasifican de dos maneras según el tipo de señal de salida y según la naturaleza del sensor.

Según el tipo de señal de salida:

- **Analógicos:** La salida es un valor de tensión o corriente.
- **Digitales:** La salida únicamente toma dos valores.

Algunos ejemplos de sensores a ser utilizados en este proyecto son:

Sensor de Temperatura LM35

El LM35 es un sensor de temperatura analógico. La salida del sensor es lineal por lo que no se necesita hacer cálculos de conversión.

Características:

- Calibrado en grados Kelvin.
- Rango de trabajo 10 Mv/°K.
- Temperatura de funcionamiento: -40 a 100 ºC

Figura 20. Sensor de Temperatura LM35.

Fuente:(Frikadas con Arduino y Raspberry Pi, 2009).

Sensor de Humedad DHT11

Es un sensor de temperatura y humedad digital.

Características:

- Alimentación: 3 a 5V.
- Rango: 0-100% (±2-5%).
- Capaz de leer temperatura de -40 a 80°C (±0.5°C).
- Sample rate: No más de 0.5 Hz (1 vez cada 2 segundos).

Figura 21.Sensor de Humedad y Temperatura DHT11.

Fuente:(Light In The Box Ltd., 2006).

Fotorresistencia LDR

Una fotorresistencia LDR es sensible a la luz que recibe y ofrece una resistencia mayor o menor en función de la cantidad de luz que recibe.

Características:

- Resistencia (con luz) : ~1 kΩ.
- Resistencia (oscuridad): ~10 kΩ.

Figura 22.Fotorresistencia LDR.

Fuente: (BricoGeek, 2005).

Los actuadores son dispositivos que pueden provocar un efecto sobre un proceso automatizado, son capaces de generar una fuerza a partir de líquidos, energía eléctrica y gaseosa. Los actuadores al ser controlados o regulados emiten una salida necesaria para activar a un elemento final.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DE LAS PLATAFORMAS DE RED TRADICIONALES

3.1 REQUERIMIENTOS DEL SISTEMA

El sistema que se va a implementar, para la evaluación del modelo de referencia del *Internet of things* está basado en las plataformas *Open Hardware (*Arduino*)* y comercial (Digi) y deben cumplir con los siguientes requerimientos:

- **Detección y Recolección de datos:** los tipos de sensores requeridos por el *Internet of things* varían dependiendo de la aplicación. En este proyecto se va a utilizar sensores para el monitoreo de ambientes, que permitan medir parámetros como temperatura, humedad y luminosidad.

- **Capacidad de procesamiento embebido:** se requiere de microcontroladores que permitan el control y procesamiento en tiempo real de los datos obtenidos por los sensores. Para cada plataforma de desarrollo del proyecto se utiliza un controlador.

- **Capacidad de comunicación alámbrica y/o inalámbrica:** la comunicación permite la transferencia de información generada por los sensores y procesada por el microcontrolador. En esta aplicación, la comunicación entre los nodos sensores y los de procesamiento es inalámbrica, y la transmisión de los datos hacia una interfaz de visualización web es Ethernet.

- **Software para automatizar:** el software para el desarrollo de *IoT* permite que varios dispositivos heterogéneos interactúen entre sí y con la infraestructura a su alrededor. La configuración de este proyecto utiliza el lenguaje de programación Python para la plataforma Digi y un lenguaje propio de Arduino basado en C, para dicha plataforma.

- **Procesamiento remoto y acceso a la nube:** se requiere la comunicación entre dispositivos y la nube de internet para ofrecer servicios como visualización, control de dispositivos y procesamiento de datos.(Karimi & Atkinson, 2013).

3.2 DISEÑO E IMPLENTACIÓN DEL PROTOTIPO IOT BASADO EN ARDUINO

3.2.1 Arquitectura de la red

La aplicación del modelo de referencia IoT consiste en la monitorización de dos ambientes y el control de dispositivos, para lo cual se cuenta con cuatro nodos, dos de los cuales recolectan datos de sensores de temperatura, humedad y luminosidad, uno para cada ambiente. Y los otros dos nodos permiten controlar el encendido y apagado de dispositivos. La comunicación entre los nodos es mediante Zigbee®, uno de los nodos realiza las funciones de coordinador y dispositivo final.

El coordinador recolecta los datos de los dos ambientes, realiza un procesamiento local y posteriormente transmite esta información hacia la interfaz web, para su visualización y procesamiento.

La interfaz web utiliza el servicio de alojamiento web, Google App Engine, este servicio permite ejecutar aplicaciones sobre la infraestructura de Google de forma gratuita hasta un límite de 500 Megabytes de almacenamiento permanente. (Google Developers, 2013). En la interfaz se visualizan los datos de los sensores, el tiempo de la muestra, el valor medio y la varianza, dentro de

una tabla y en un gráfico que plasma estos valores. También se puede realizar el encendido o apagado de dispositivos en otra interfaz.

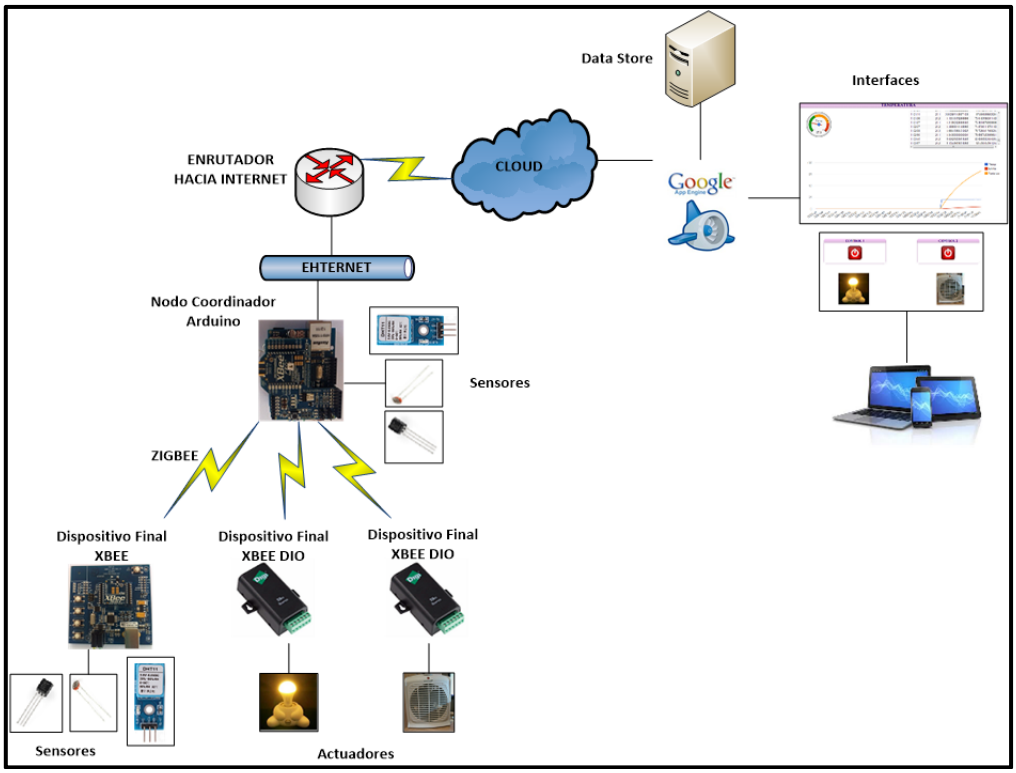

Figura 23. Arquitectura de la red.

Fuente: (Elaboración Propia)

3.2.2 Diseño de la red

La red de sensores para el monitoreo de ambientes y control de dispositivos está compuesta por cuatro nodos, un Coordinador que a la vez realiza las funciones de dispositivo final y tres nodos terminales que se comunican inalámbricamente mediante Zigbee®.

En el prototipo, dos de los dispositivos terminales estarán encargados de medir tres variables (temperatura, humedad y luminosidad), el resto de nodos finales controlan el encendido y apagado de dos dispositivos.

En el presente proyecto la recolección y transmisión de datos desde el nodo terminal hacia el coordinador es automática; en consecuencia, se utiliza gran parte del ancho de banda de Zigbee®, debido a que el tráfico es continuo.

Topología de la red

Debido a las limitaciones en cuanto a acceso y costo de los dispositivos y herramientas necesarias y teniendo en cuenta que el presente proyecto pretende demostrar un prototipo, se optó por el diseño de una red básica con una topología tipo estrella.

Selección de componentes básicos

Para la selección de los componentes se tiene en cuenta las siguientes consideraciones:

- La versión ZigBee® que utilizan.
- Las bandas de frecuencias que soportan.
- El precio del módulo.

- La sensibilidad del receptor.
- La máxima potencia de transmisión.
- El alcance máximo.

En base a estas consideraciones, el coordinador y los dispositivos finales seleccionados corresponden a módulos XBEE Series 2.

El dispositivo final de monitorización utiliza dos entradas analógicas para la lectura de sensores. Los dispositivos finales de control utilizan una salida digital para el control de encendido y apagado.

El coordinador está compuesto por una tarjeta Arduino UNO, la cual contiene un microcontrolador para realizar la adquisición y procesamiento de los datos, tanto de los sensores conectados a sus entradas analógicas y digitales, como de los sensores conectados al dispositivo final y el control de los dispositivos en el resto de nodos finales.

También consta de una Arduino Ethernet*Shield*, que permite la conexión vía Ethernet a un enrutador, que le proporcione acceso a Internet. (Arduino). Para conectarse de forma inalámbrica con los dispositivos finales, requiere de una XBEE*Shield*, que contendrá al módulo XBEE. Las tres tarjetas deben ser compatibles entre sí.

El coordinador utilizará una entrada digital y una analógica para la lectura de sensores.

3.2.3 Diseño de circuitos de acondicionamiento

Sensores

El dispositivo final soporta voltajes de entrada analógicos de máximo 1.2 VDC, mientras que el Coordinador soporta voltajes de máximo 5VDC.

El sensor utilizado para la medición de Temperatura es el LM35, para el cual no fue necesario realizar un circuito de acoplamiento, ya que su rango de voltaje es el requerido tanto para el dispositivo final como para el coordinador.

El sensor utilizado para la medición de Luminosidad es un LDR, el cual varía su resistencia dependiendo de la cantidad de luz, en condiciones de luz tiene aproximadamente una resistencia de 1 kΩ y en obscuridad una resistencia de 10k Ohm aproximadamente. La conexión del sensor es la siguiente:

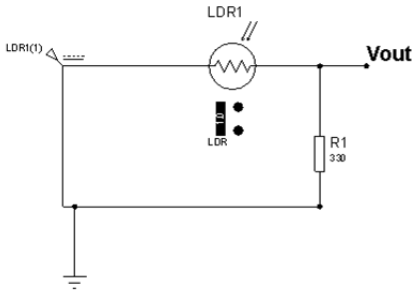

Figura 24. Circuito de acondicionamiento sensor LDR.

Fuente: (Elaboración Propia)

Calcular la resistencia R1, de manera que el voltaje de salida Vout sea menor a 1.2 VDC para evitar daños en las entradas analógicas de los dispositivos.

Para el cálculo se aplica la fórmula de un divisor de voltaje:(Elliott, 2002).

$$Vout = \left(\frac{R1}{Rldr + R1} \right) * Vin$$

$$R1 = \frac{-Rldr * Vout}{Vout - Vin}$$

Ecuación 5. Cálculo de divisor de voltaje.

En donde,

$$Rldr = 1k\Omega$$
$$Vin = 3{,}3VDC$$
$$Vout = 1{,}2VDC$$

Reemplazando en la Ecuación 5,

$$R1 = \frac{-1000 * 3,3}{1,2 - 3,3}$$

$$R1 = 571,42\Omega$$

Se utilizará una resistencia R1 igual a 330 Ω para asegurar que el máximo Vout sea menor a 1.2VDC.

El sensor utilizado para la medición de la humedad y temperatura para el nodo coordinador es un DHT11, el cual es un sensor digital y no necesita de un circuito de acoplamiento debido a que ya viene incorporado en el mismo.

Actuadores

Dos dispositivos finales controlarán el encendido y apagado de dispositivos mediante sus respectivas salidas digitales y actuadores, los cuales requieren de un circuito de acoplamiento. Los actuadores que se utilizarán son relés de 5VDC.

La salida digital del dispositivo final XBEE DIO es en colector abierto y el rango que se obtiene es de 0VDC para el estado de encendido y 4VDC para el estado apagado.(Inc., Digi International, 1996).

La alimentación se obtendrá de una batería de 9VDC. El relé soporta un voltaje de alimentación de 5VDC, por lo tanto se debe utilizar un regulador de voltaje para obtener este valor.

Para la conexión de los circuitos se necesitan los siguientes elementos:

- 2 transistores 2N3904.
- 2 diodos 1N4007.
- 2 resistencias de 10kΩ.
- 2 baterías de 9VDC.
- 2 relés de 5VDC.
- 2 reguladores 7805.
- 2 capacitores 1uF.
- 2 capacitores 10uF.
- 1 calefactor de 120V.
- 1 lámpara 60W.

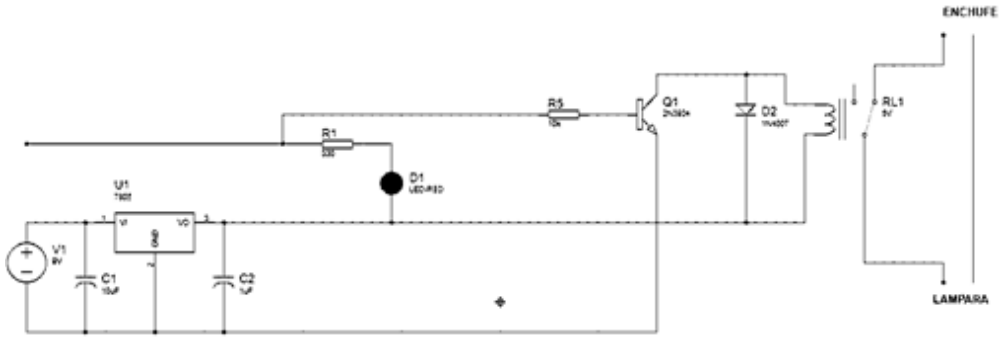

Figura 25. Circuito de acondicionamiento salida digital dispositivo final. Fuente: (Elaboración Propia).

3.2.4 Implementación de la red

Configuración de los nodos

La configuración de los nodos finales y coordinador se realiza mediante el software X-CTU, con ayuda de las interfaces de desarrollo XBEE *Explorer* USB y RS232.

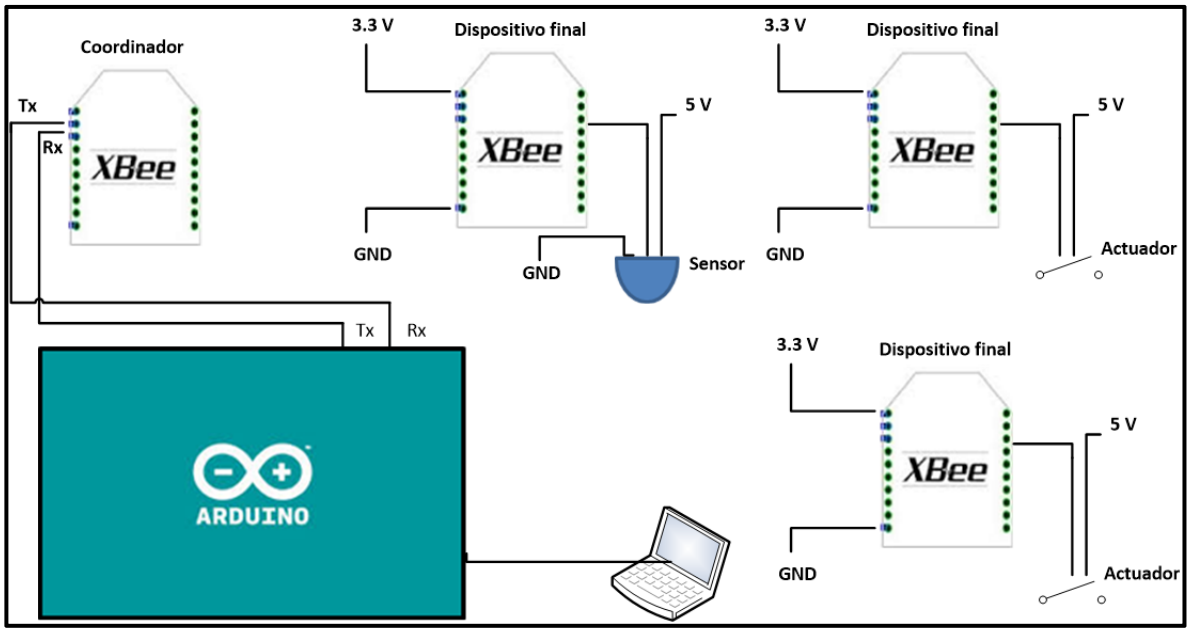

Figura 26. Diagrama de conexión de la red ZigBee®.

Fuente: (Elaboración Propia)

Nodo Coordinador

El nodo coordinador contiene el módulo XBEE, colocado sobre la tarjeta Arduino UNO mediante su XBEE Shield. Los parámetros más importantes configurados en el módulo son:

- *PC Settings*: Activar*ENABLE API.*
- *Function Set*: *ZIGBEE COORDINATOR API.*
- PAN ID: 666.

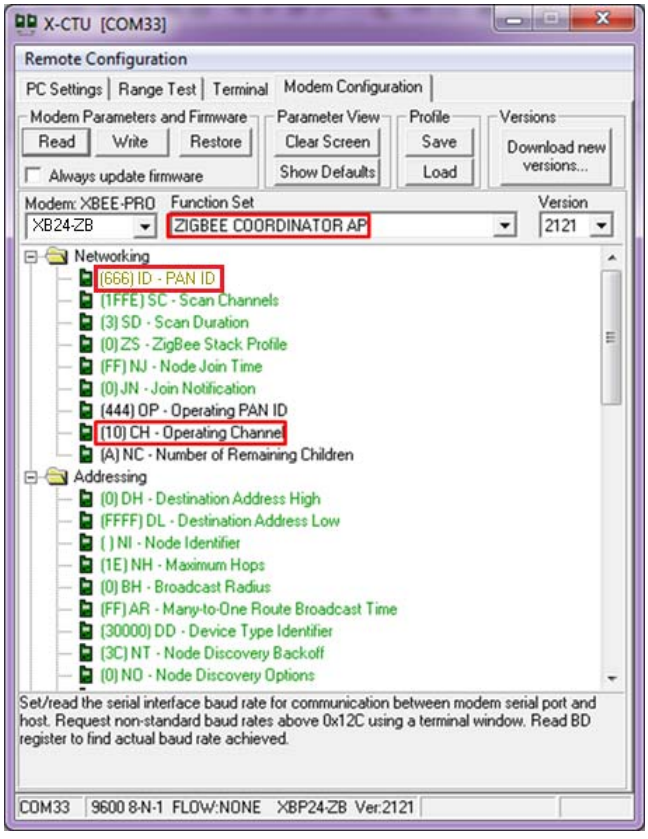

Figura 27.Configuración coordinador en X-CTU.

Fuente: (Elaboración Propia)

Nodos Finales

La configuración del dispositivo final de monitoreo es la siguiente:

- *Function Set:* ZIGBEE *END DEVICE AT.*
- PAN ID: 666 (El mismo valor del coordinador).
- *ChannelVerification*: 1.
- *AD2/DIO2-Configuration*: 2 (Entrada analógica).
- *AD3/DIO3-Configuration*: 2 (Entrada analógica).

La configuración de los dispositivos finales de control es la siguiente:

- *Function Set:* ZIGBEE *END DEVICE AT.*
- PAN ID: 666 (El mismo valor del coordinador).
- *ChannelVerification:* 1.
- *DIO4-Configuration*: 4 (Salida digital en bajo).

Se debe encender primero el nodo coordinador y después sus dispositivos finales para que elijan el mismo canal en el que trabaja el coordinador, para este caso los nodos trabajan en el canal 10.

Los parámetros DH y DL son la dirección de destino, para este caso se ha colocado el valor de 0, con el que se conecta con el coordinador de la red, debido a que no existe otro coordinador en la misma.

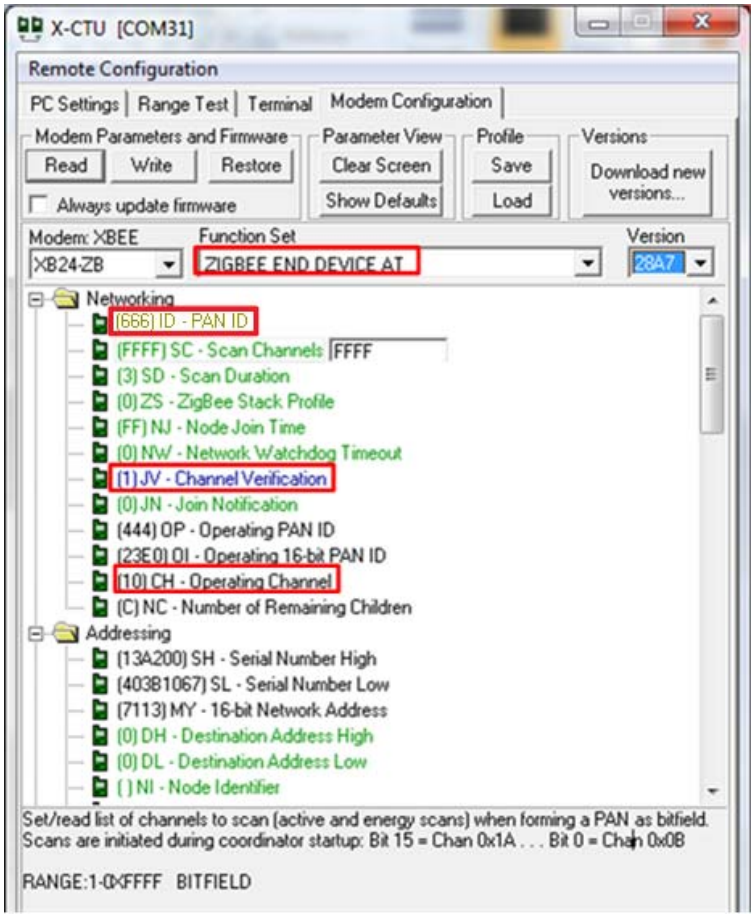

Figura 28.Configuración dispositivos finales en X-CTU.

Fuente: (Elaboración Propia)

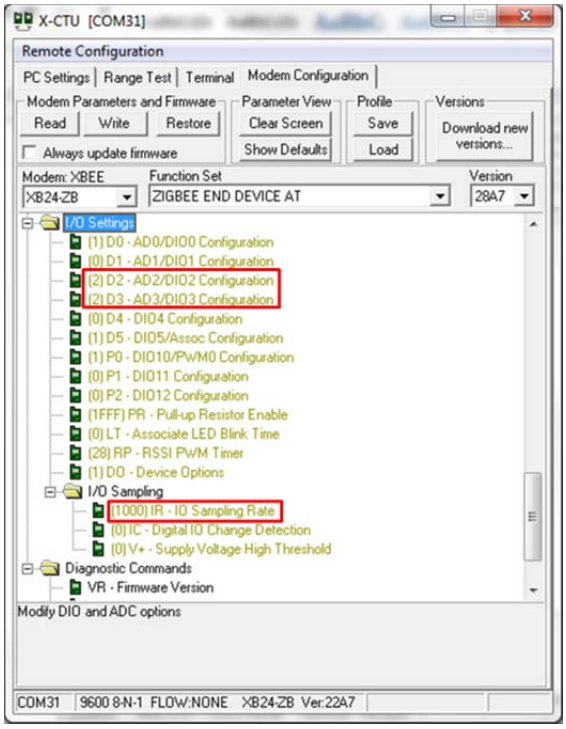

Figura 29.Configuración entradas analógicas dispositivo final en X-CTU. Fuente: (Elaboración Propia)

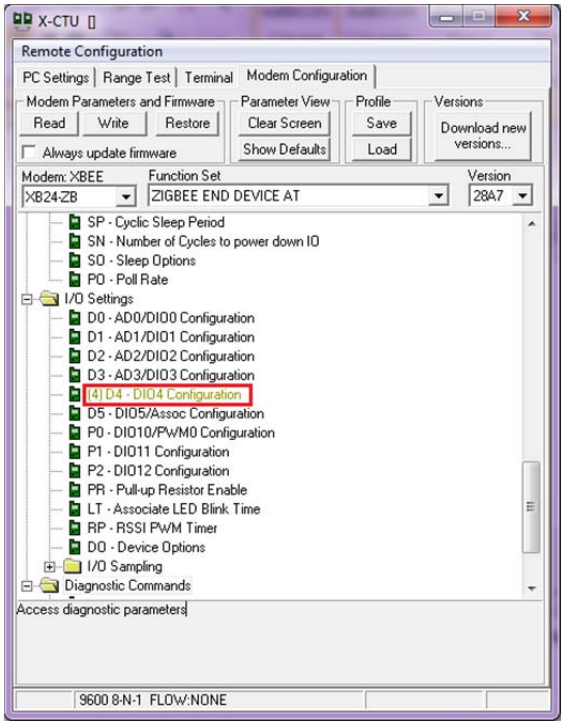

Figura 30. Configuración salida digital dispositivo final en X-CTU.

Fuente: (Elaboración Propia)

Integración del Hardware

- **Montaje del nodo coordinador**

Como ya se mencionó en el diseño de la red, el nodo coordinador está compuesto por una tarjeta Arduino UNO, una Arduino Ethernet *Shield*, una XBEE *Shield* y el módulo XBEE. Las tres tarjetas deben estar apiladas una sobre la otra. La Ethernet *Shield* se coloca directamente sobre la tarjeta Arduino, encajando todos sus pines.

La *XBEE Shield* utiliza los pines ICSP de la tarjeta Arduino para VCC, GND y RESET. Al colocar la Ethernet *Shield*, estos pines son utilizados y ya no están disponibles para conectarse con la XBEE *Shield*. Por lo tanto es necesario utilizar conectores externos para utilizar estos pines. Se sueldan cables en estos pines, en la tarjeta Arduino UNO para poder conectarlos a la XBEE *Shield*, como se muestra en la siguiente figura. Fuente:(Home Automation, 2012).

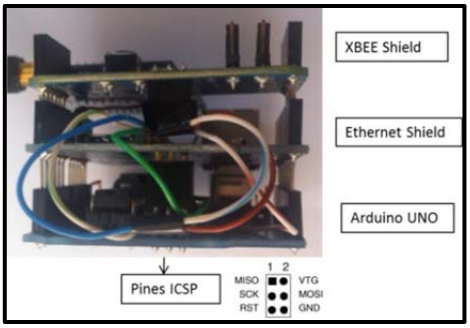

Figura 31. Montaje del nodo coordinador.

Fuente: (Elaboración Propia)

Los pines MISO, MOSI y SCK no son necesarios para la XBEE *Shield*, ya que estos pines son utilizados solo por la Ethernet *Shield.*

El coordinador se conecta con la placa que contiene los sensores y sus circuitos de acondicionamiento.

Figura 32. Integración placa de sensores con el nodo coordinador.

Fuente: (Elaboración Propia)

- **Montaje de los nodos finales de control**

Los nodos finales de control se conectan con la placa que contiene los actuadores y sus circuitos de acondicionamiento.

Figura 33. Integración placa de actuadores con el nodo final.

Fuente: (Elaboración Propia)

Programación Microcontrolador (Arduino)

La programación se la realiza en el entorno de desarrollo Arduino IDE, en el lenguaje de programación C/C++. Se lo puede descargar de la página web de Arduino: http://arduino.cc/es/Main/Software.

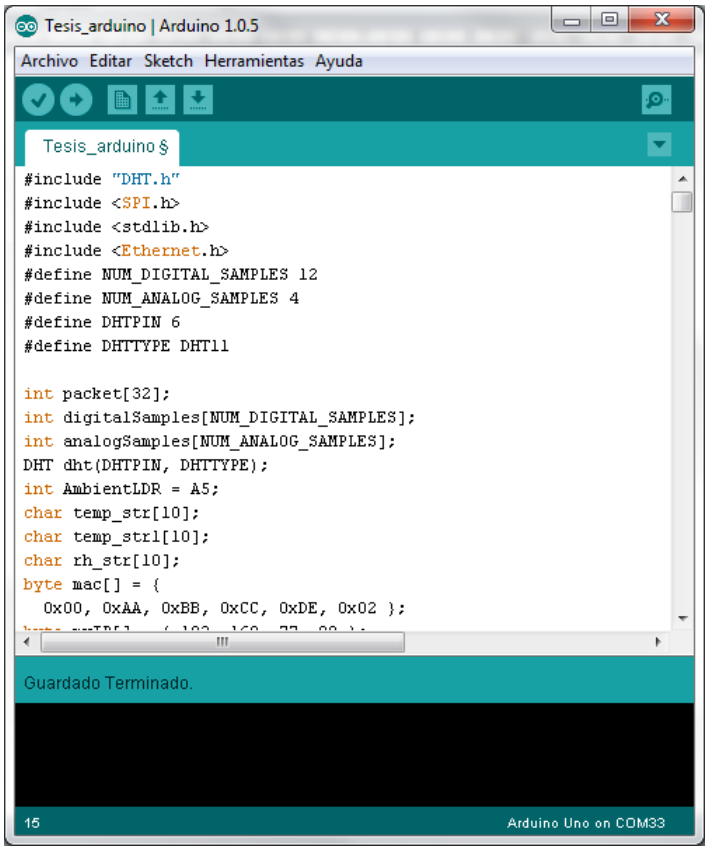

Figura 34. Programa microcontrolador.

Fuente: (Arduino).

El programa desarrollado lee los datos de las entradas analógicas y digitales, tanto del coordinador como del dispositivo final de monitorización, realiza un pre-procesamiento de los valores obtenidos, conecta al coordinador

con las aplicaciones Google App Engine y viceversa enviando y recibiendo datos, permitiendo el control de dispositivos.

Las librerías necesarias para el programa son:

- **SPI.h:** permite la comunicación serial con uno o más dispositivos periféricos. (Arduino). Para este proyecto permite la comunicación entre el computador y la tarjeta Arduino UNO para su programación, el envío y recepción de datos entre el coordinador y el dispositivo final. Debido a que la tarjeta Arduino UNO posee un solo puerto serial se deben utilizar jumpers en la XBEE *Shield*, para la comunicación con el dispositivo final en modo XBEE y para la comunicación con el PC en modo USB.

- **Stdlib.h:** es una biblioteca en la cual se encuentran varias funciones, entre las cuales están: Aritméticas, Números aleatorios y Conversión de cadenas.

- **Ethernet.h:** esta librería permite a la placa Arduino conectarse a Internet ya sea como servidor o como cliente, con conexiones entrantes y salientes. (Arduino).

- **DHT.h:** permite utilizar funciones para la lectura de datos del sensor DHT (sensor de humedad y temperatura).(GitHub, Inc., 2012).

Las funciones realizadas en el programa son las siguientes: (Iowa-Aquaponics).

- **Setup():** Es la función principal de inicialización. Dentro de esta función se inicializa el puerto serial y se establece la velocidad de transmisión en 9600, también inicializa la librería Ethernet y la configuración de red, para lo cual se deben incluir variables que representen la dirección MAC, IP y Gateway de la tarjeta Ethernet *Shield*, declara las salidas digitales de la Arduino y por ultimo inicializa la librería DHT.

- **Loop():** Esta función se ejecuta automáticamente después de Setup() y todo lo contenido dentro de esta función se ejecutará de forma cíclica. En esta función se llaman al resto de funciones del programa. Se realiza el pre-procesamiento de la lectura de los datos, dependiendo de la información enviada por la aplicación, condiciona el estado de las salidas digitales.

- **GetDHT():** Esta función realiza la lectura del sensor DHT, almacenando los valores de humedad y temperatura en variables diferentes y retorna un String que contiene estos valores con sus respectivas etiquetas, para ser enviados en una petición HTTP.

- **GetLDR():** Esta función realiza la lectura del sensor LDR, que se encuentra conectado directamente a la Arduino y retorna un String que contiene el valor con su respectiva etiqueta, para ser enviado en una petición HTTP.

- **GAE1():** Esta función recibe como argumentos el link y el servidor de la aplicación Google App Engine, llama a la función httRequest(), realiza una lectura del cliente Ethernet, mientras este se encuentre disponible, establece la conexión con el cliente y retorna su estado.

- **httpRequest():** Esta función recibe como argumentos el link y el servidor de la aplicación Google App Engine, establece la conexión con el servidor y envía al cliente la petición HTTP.

- **readPacket():** Esta función realiza una lectura del puerto serial que contendrá las tramas enviadas por los dispositivos finales, extrae el valor del pin, que se encuentra en los Bytes 21 y 22 de la

trama (*Analog Sample data*) del dispositivo de monitorización y lo retorna.(Nootropic Design, 2009).

API format for I/O Data Sample RX Indicaton

Byte	Example	Description
0	0x7e	Start byte – Indicates beginning of data frame
1	0x00	Length – Number of bytes (ChecksumByte# – 1 – 2)
2	0x14	
3	0x92	Frame type - 0x92 indicates this will be a data sample
4	0x00	64-bit Source Address (Serial Number)
5	0x13	MSB is byte 4, LSB is byte 11
6	0xA2	
7	0x00	
8	0x40	
9	0x77	
10	0x9C	
11	0x49	
12	0x36	Source Network Address – 16 Bit
13	0x6A	
14	0x01	Receive Opts. 01=Packet Acknowledged. 02=Broadcast packet
15	0x01	Number of sample sets. Always set to 1 due to XBEE limitations
16	0x00	Digital Channel Mask – Indicates which pins are set to DIO
17	0x20	
18	0x01	Analog Channel Mask – Indicates which pins are set to ADC
19	0x00	Digital Sample Data (if any) – Reads the same as Digital Mask
20	0x14	
21	0x04	Analog Sample data (if any)
22	0x25	There will be two bytes here for every pin set for ADC
23	0xF5	Checksum(0xFF - the 8 bit sum of the bytes from byte 3 to this byte)

Figura 35.Descripción formato de trama API.

Fuente: (TunnelsUP, 2012).

- **readByte():** Esta función lee el puerto serial mientras se encuentre disponible.(Nootropic Design, 2009).

- **setRemoteState():** Esta función recibe como argumento el estado de la salida digital del dispositivo de control, envía la trama por el puerto serial para el encendido y apagado de esta salida. La función **setRemoteState1()** realiza el mismo procedimiento para el segundo dispositivo de control.(TunnelsUP, 2012).

Diagrama de flujo del programa de Arduino

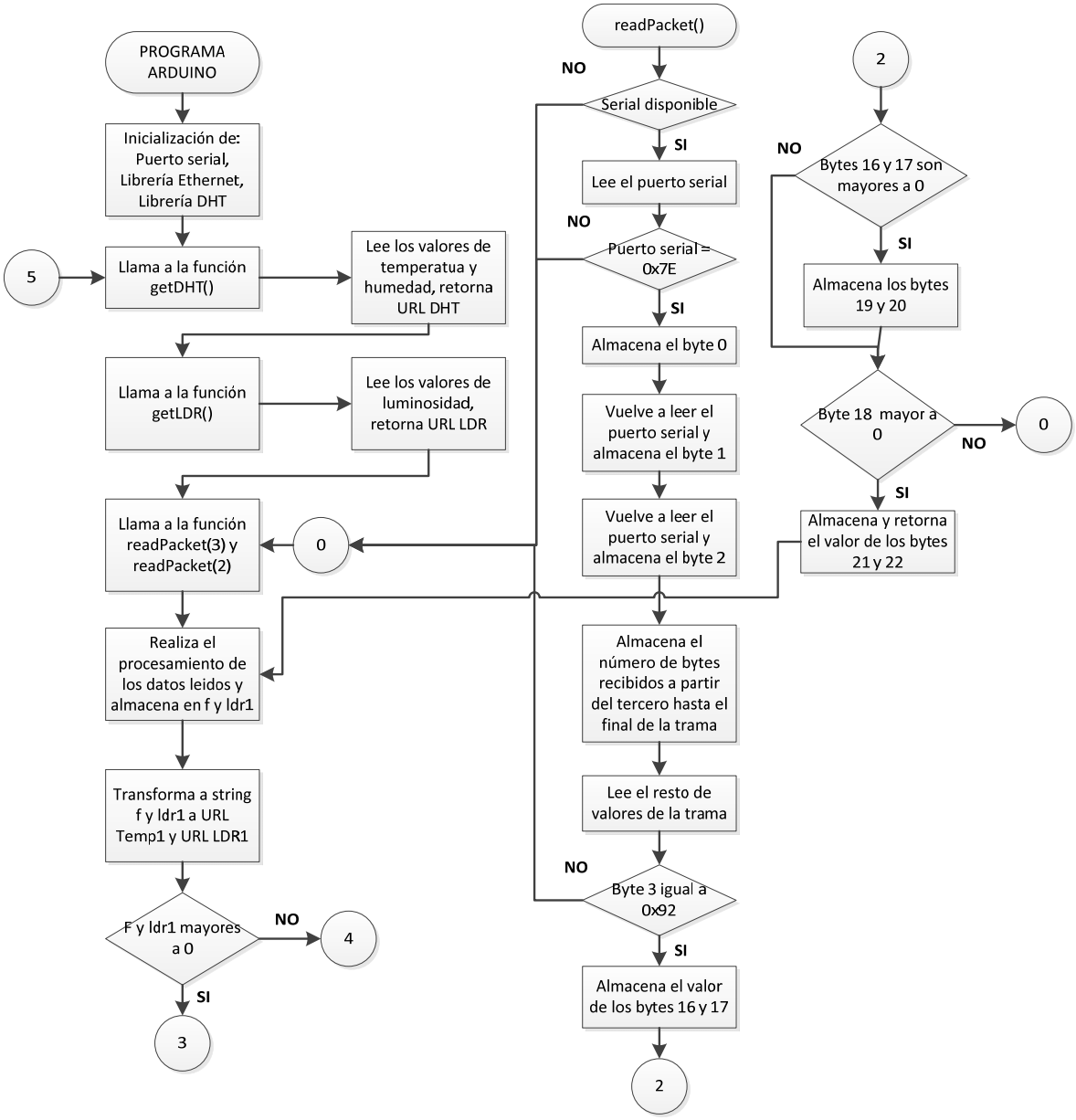

Figura 36. Diagrama de flujo del programa del microcontrolador (primera parte).

Fuente: (Elaboración Propia)

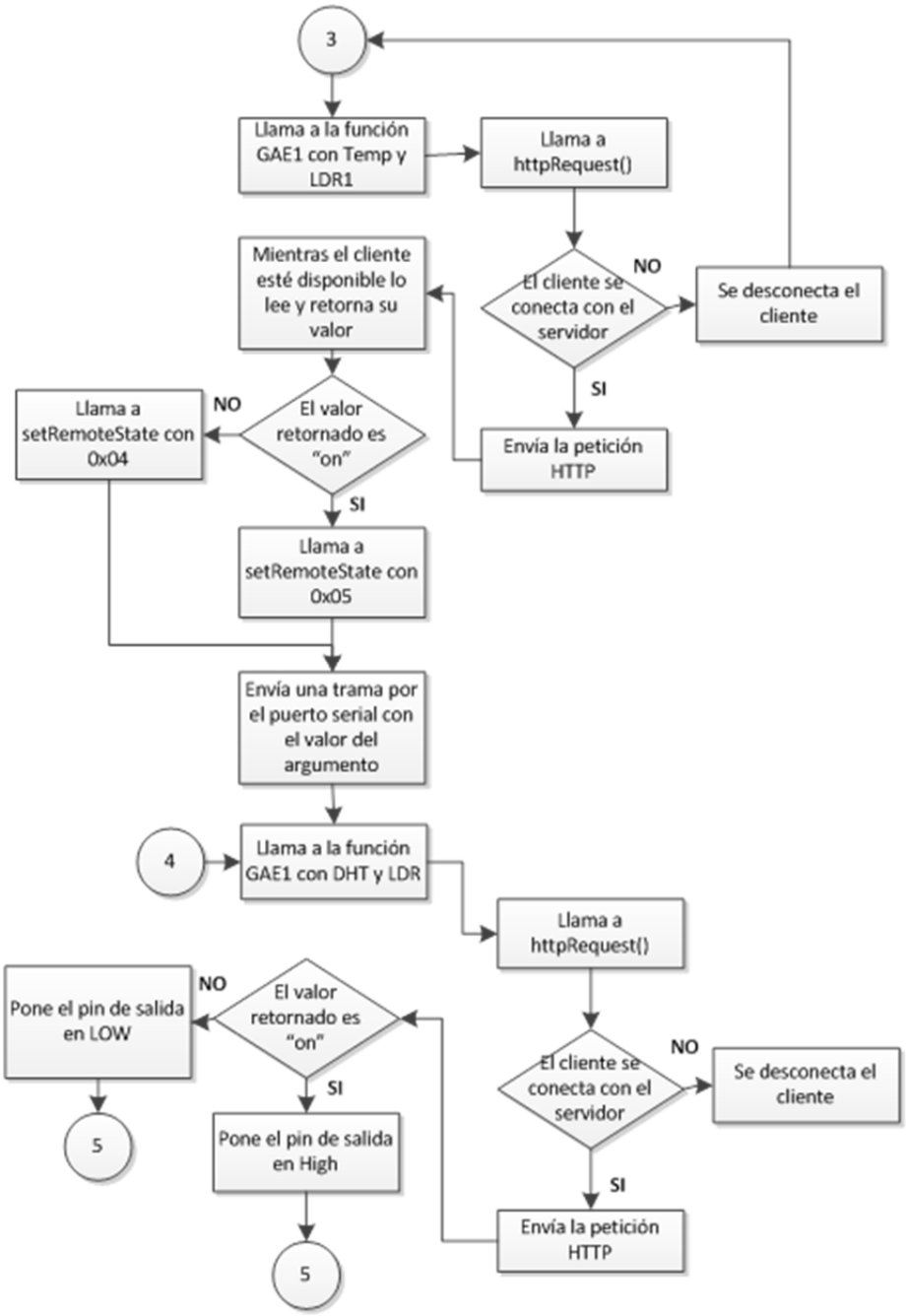

Figura 37. Diagrama de flujo del programa del microcontrolador (segunda parte).

Fuente: (Elaboración Propia)

Descripción de la Interfaz de usuario

Esta plataforma monitorea dos ambientes separados y controla dos dispositivos, por lo cual, se desarrollaron tres interfaces. El ambiente 1 es monitoreado por el coordinador.El ambiente 2 es monitoreado por el dispositivo final.Cada ambiente tiene su interfaz y la tercera es una interfaz de control.

La interfaz del ambiente1 está compuesta por una página principal, que contiene:

- Una barra de menú que contiene tres pestañas: una principal y las dos siguientes permiten abrir la interfaz de monitoreo del ambiente2 y la interfaz de control.

Figura 38. Barra de menú en las interfaces.

Fuente: (Elaboración Propia)

- La pestaña Principal contiene tres secciones, la primera corresponde al monitoreo de la temperatura del ambiente. En esta sección se puede visualizar una tabla con valores de tiempo en que llega la muestra, el valor de la temperatura en grados centígrados, el valor medio de las muestras que se van

presentando y el valor de la varianza de las mismas. También se muestra una gráfica correspondiente a los valores medidos de temperatura, el valor medio y el valor de la varianza. Por último se presenta un medidor con el valor actual de temperatura. El medidor tiene un rango desde 0°C hasta 100°C, este rango se divide en tres partes, el color verde indica que la temperatura ambiente se encuentra entre 0°C y 25°C, el color amarillo indica valores entre 25°C y 50°C, valores que pueden ser soportables y el color rojo indica valores desde 50°C hasta 100°C.

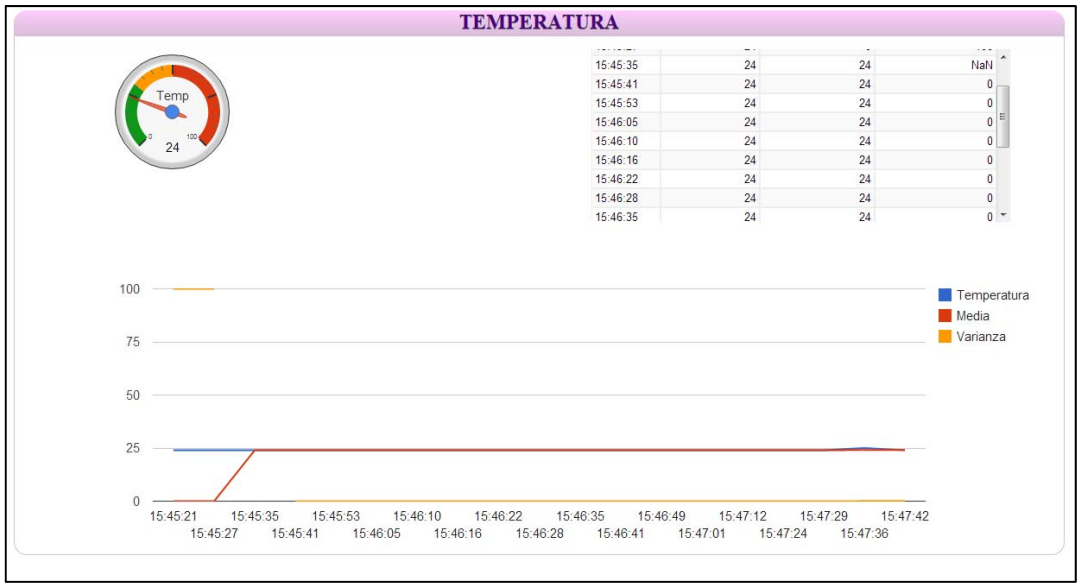

Figura 39. Sección de monitoreo de temperatura del ambiente1.

Fuente: (Elaboración Propia)

- La segunda sección corresponde al monitoreo de la humedad del ambiente. En esta sección se puede visualizar una tabla con valores de tiempo en que llega la muestra, el valor de la humedad

en porcentaje, el valor medio de las muestras que se van presentando y el valor de la varianza de las mismas. También se muestra una gráfica correspondiente a los valores medidos de humedad, el valor medio y el valor de la varianza. Por último se presenta un medidor con el valor actual de humedad. El medidor tiene un rango desde 0% hasta 100%, este rango se divide en tres partes, el color verde indica que la humedad es muy baja y se encuentra entre 0% y 25%, el color amarillo indica valores entre 25% y 80%, valores de humedad normales y el color rojo indica valores desde 80% hasta 100%, que representan una humedad muy elevada cerca del punto de saturación.

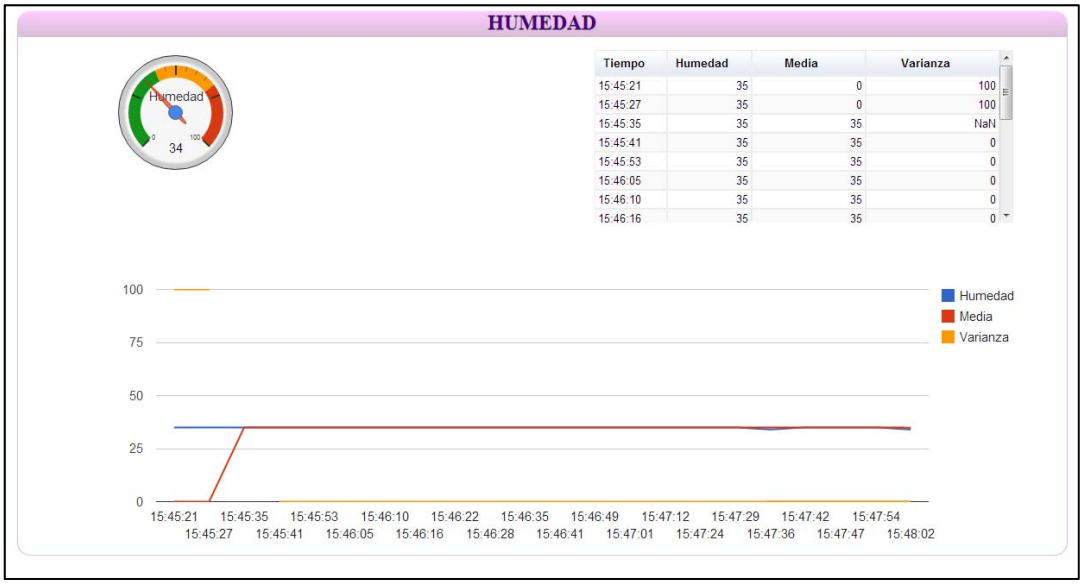

Figura 40. Sección de monitoreo de humedad del ambiente1.

Fuente: (Elaboración Propia)

- La última sección corresponde al monitoreo de la luminosidad del ambiente. En esta sección se puede visualizar una tabla con valores de tiempo en que llega la muestra, el nivel de intensidad de luz del ambiente, el valor medio de las muestras que se van presentando y el valor de la varianza de las mismas. También se muestra una gráfica correspondiente a los valores medidos de luminosidad, el valor medio y el valor de la varianza. Por último se presenta un medidor con el valor actual de luminosidad. El medidor tiene un rango desde 0 hasta 1000, este rango se divide en tres partes, el color verde indica que la luminosidad es muy baja y se encuentra entre 0 y 300, el color amarillo indica valores entre 300 y 600, valores de luminosidad a media tarde y el color rojo indica valores desde 600 hasta 1000, que representan una luminosidad muy elevada, que puede ocurrir en el día.

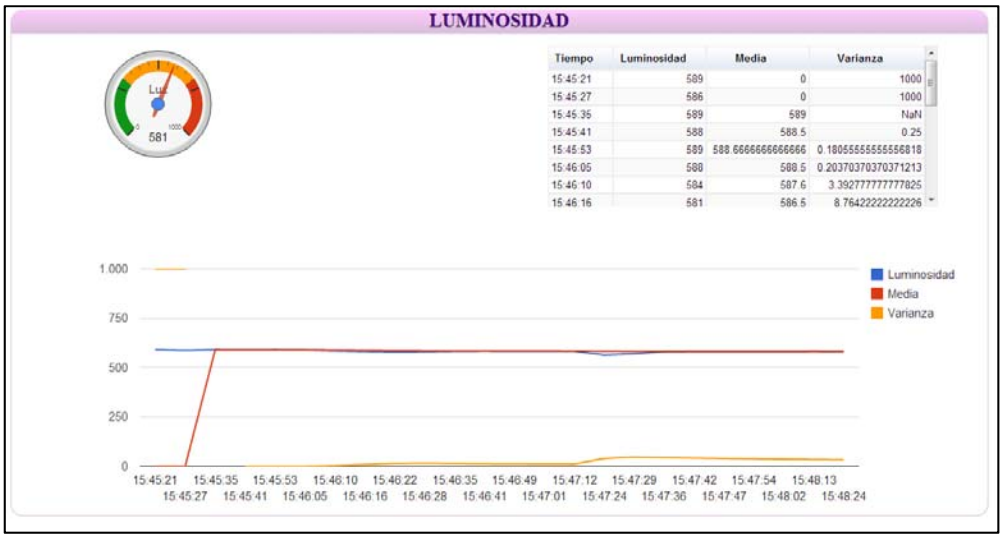

Figura 41. Sección de monitoreo de luminosidad del ambiente1.

Fuente: (Elaboración Propia)

La interfaz del ambiente2 contiene la misma estructura del ambiente1, con la diferencia de que muestra valores de sensores de temperatura y luminosidad.

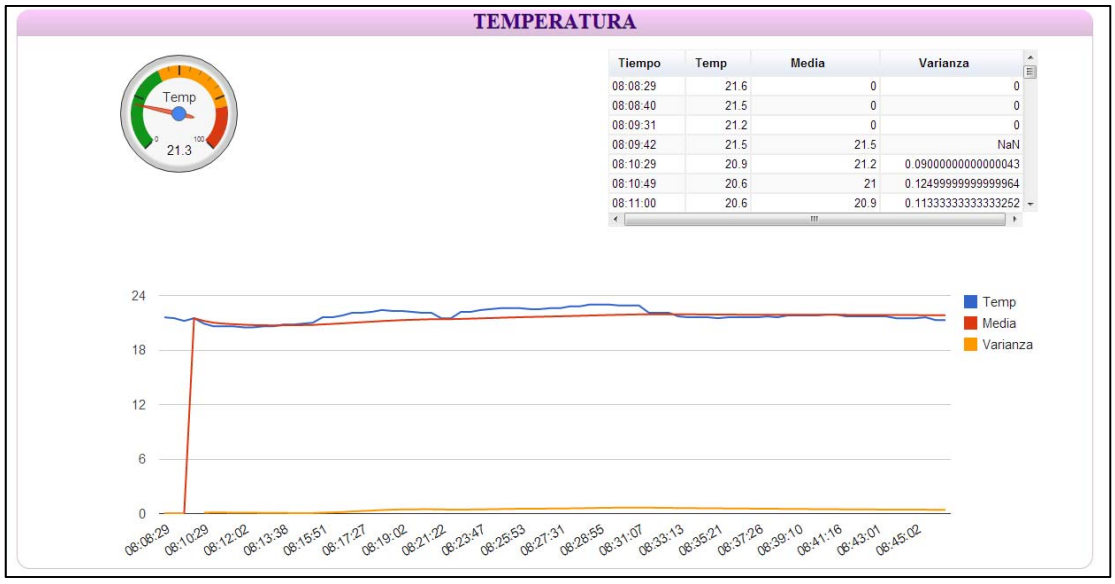

Figura 42. Sección de monitoreo de temperatura del ambiente2.

Fuente: (Elaboración Propia)

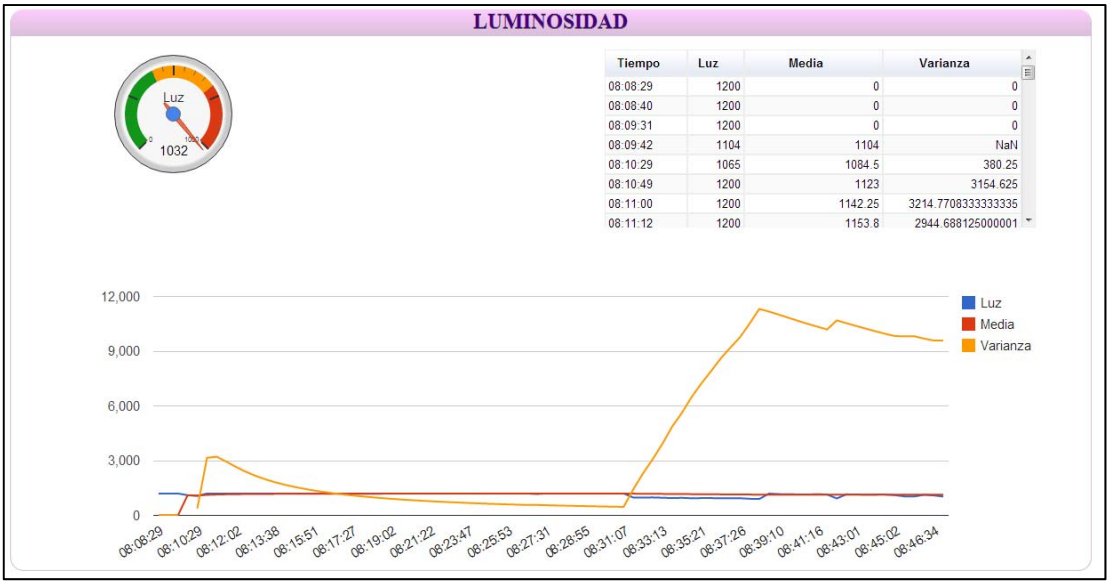

Figura 43. Sección de monitoreo de luminosidad del ambiente2.

Fuente: (Elaboración Propia)

La interfaz de control está compuesta por la barra de menú de las anteriores interfaces de monitorización y control, en el que se encuentran dos botones, que al hacer clic sobre cada uno de ellos cambian su estado de encendido a apagado y viceversa. Los botones se utilizan para encender o apagar una lámpara y un calefactor. El color rojo representa el apagado del dispositivo y el color verde el encendido.

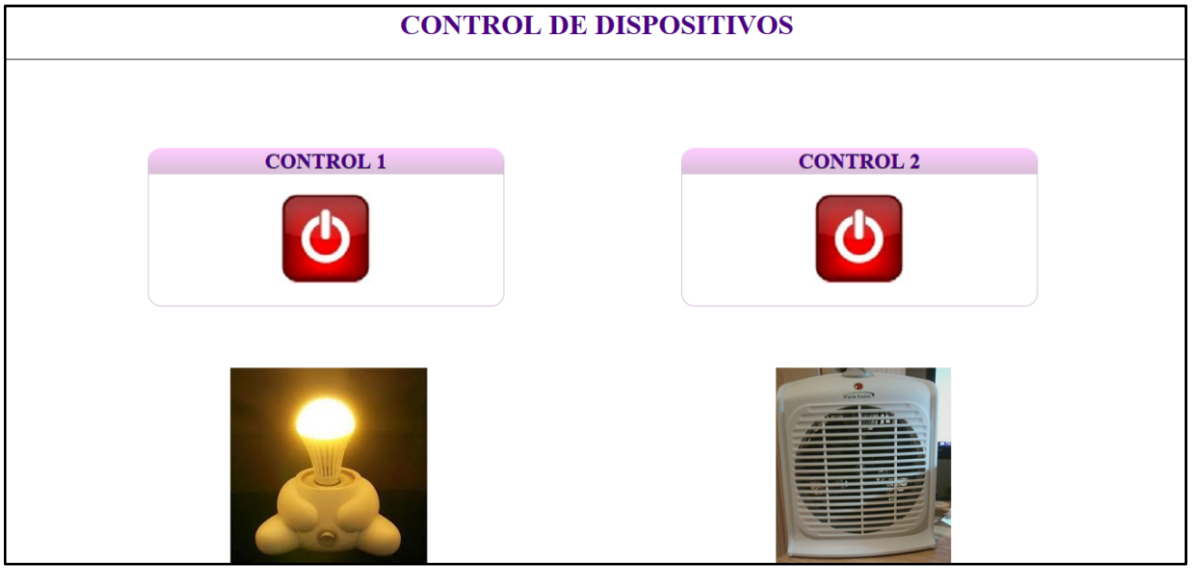

Figura 44. Interfaz de control de dispositivos.

Fuente: (Elaboración Propia)

Para poder acceder a las interfaces se debe colocar las siguientes direcciones:

- www.ambiente11monitoreo.appspot.com,
- www.monitoreoambiente2.appspot.com
- www.controldedispositivos.appspot.com

Para lo cual se debe iniciar sesión en la cuenta de www.appengine.google.com.

Programación de la Interfaz de usuario en Google App Engine

Google App Engine permite ejecutar aplicaciones web en la infraestructura de Google. Con App Engine no se necesita utilizar ningún servidor: solo subir la aplicación para que los usuarios puedan empezar a utilizarla. Todas las aplicaciones pueden utilizar hasta 500 MB de almacenamiento y suficiente CPU y ancho de banda como para permitir un servicio eficaz de alrededor de cinco millones de visitas a la página al mes, sin coste alguno.(Google Developers, 2013).

Es necesario tener una cuenta de correo electrónico en Gmail, para el desarrollo de esta aplicación, se ha creado la cuenta tesissensores@gmail.com. Se debe registrar en www.appengine.google.com con la cuenta de correo de gmail para poder crear las aplicaciones.

Una vez registrado en la página AppEngine hacer clic en *CreateApplication*, el cual direccionará a la siguiente página:

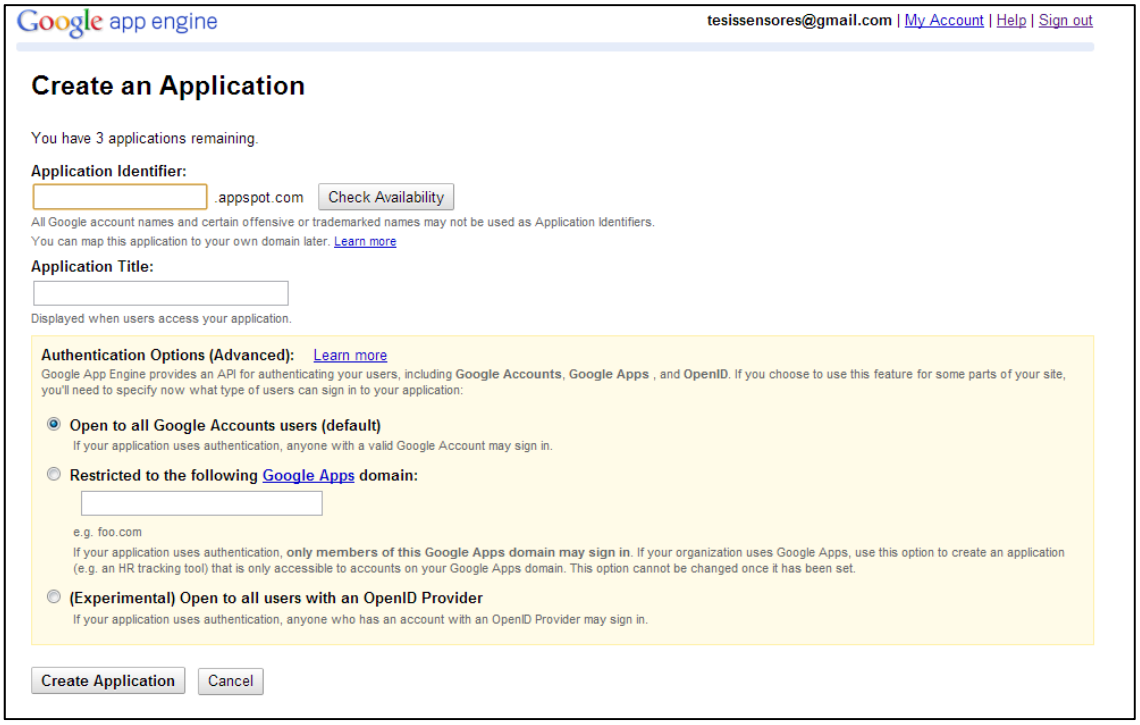

Figura 45. Página para la creación de aplicaciones en Google App Engine.

Fuente: (Google App Engine)**.**

En el campo *ApplicationIdentifier* se debe ingresar un nombre que identificará la aplicación, en caso de que esté disponible en el dominio appspot.com. Por último se hace clic en *CreateApplication.*

Google App Engine admite aplicaciones escritas en varios lenguajes de programación. Gracias al entorno de tiempo de ejecución Java de App Engine, se pueden crear aplicaciones a través de tecnologías Java estándar, que incluyen JVM, servlets Java y el lenguaje de programación Java, o cualquier otro lenguaje que utilice un intérprete o compilador basado en JVM como, por ejemplo, JavaScript o Ruby. App Engine también ofrece un entorno de tiempo de ejecución Python dedicado, que incluye un rápido interprete Python y la

biblioteca estándar Python. Los entornos de tiempo de ejecución Java y Python se generan para garantizar que la aplicación se ejecute de forma rápida, segura y sin interferencias de otras aplicaciones en el sistema. (Google Developers, 2013).

El desarrollo de la interfaz en este proyecto se realiza en el lenguaje de programación Python. El programa está compuesto por los siguientes archivos:(Iowa-Aquaponics).

- Archivos de Python (.py).
- Archivo de configuración (.yaml).
- Archivos JavaScript (.js).
- Archivos HTML (.html).
- Archivos CSS (.css).
- Una carpeta con imágenes.

A continuación se explican brevemente los archivos de la programación de la interfaz.

Archivo de configuración

Las aplicaciones *App Engine* tienen un archivo de configuración llamado app.yaml. Este archivo describe, entre otras cosas, las secuencias de

comandos de controlador que se deben utilizar para cada URL. (Google Developers). Además debe contener el nombre de la aplicación, que para este caso es “ambiente11monitoreo”, “monitoreoambiente2” y “controldedispositivos”.

Archivos de Python

- **adacs.py:** el coordinador de la red envía las peticiones HTTP con los datos de los sensores hacia la URL de este archivo, por lo tanto, este lee y almacena los valores para utilizarlos posteriormente en todo el programa. También lee el estado de los botones de control en la aplicación y lo envía como respuesta a las peticiones enviadas por el coordinador.

- **ambiente.py:** en este archivo se obtienen los datos desde el archivo adacs.py, los coloca en una lista y los transforma a formato json, para después realizar las tablas y los gráficos de los parámetros medidos.

- **herramientas.py:** este archivo permite obtener la fecha y tiempo para cada medición realizada, borra la memoria caché con los datos a las 00:00 horas. Si la aplicación deja de recibir datos

durante 3 minutos, se envía un mensaje de alerta de desconexión a la cuenta de correo gmail.

- **main.py:** al abrir la página web de la aplicación, este archivo carga los archivos html necesarios para la visualización.

Archivos de JavaScript

- **main.js:** este archivo permite cargar la página principal, lee los datos obtenidos por los anteriores archivos explicados, con estos datos, llama a las funciones que realizan las gráficas, tablas y medidores, realiza una actualización de la página, contiene la función de actualización del estado de los botones de control. Por último realiza el procesamiento de los datos de temperatura, humedad y luminosidad para obtener los valores de media y varianza.

- **luz.js, temp.js y humedad.js:** en estos archivos se declaran las funciones que generan y actualizan las gráficas, las tablas y los medidores para cada parámetro.

- **herramientas.js:** en este archivo se declara una función que permite recibir los datos de una URL y actualizar parte de la

página, sin necesidad de refrescar toda la página e interrumpir lo que el usuario se encuentre realizando.

Archivos HTML

- **contenido.html:** este archivo es el contenido de la página principal, coloca la fecha actual y los distintos bloques que tienen las gráficas para cada sensor, como se mencionó en 3.2.3.

- **script.html:** en este archivo se incluyen los códigos de javascript ya explicados y los archivos .css, que dan formato a la página.

- **cabecera.html:** contiene el título de la página y su formato.

- **menu.html:** contiene la etiqueta de la página principal.

- **fin.html:** es el cierre del código html.

Archivos CSS

Estos archivos dan formato a la página web, permitiendo cambiar el color, tipo de letra y tamaño.

Para subir la aplicación a Google App Engine es necesario descargar el SDK de Google App Engine para Python en Windows, el cual permite crear, ejecutar y subir aplicaciones. El SDK puede ser descargado desde el siguiente enlace: https://developers.google.com/appengine/downloads?hl=es

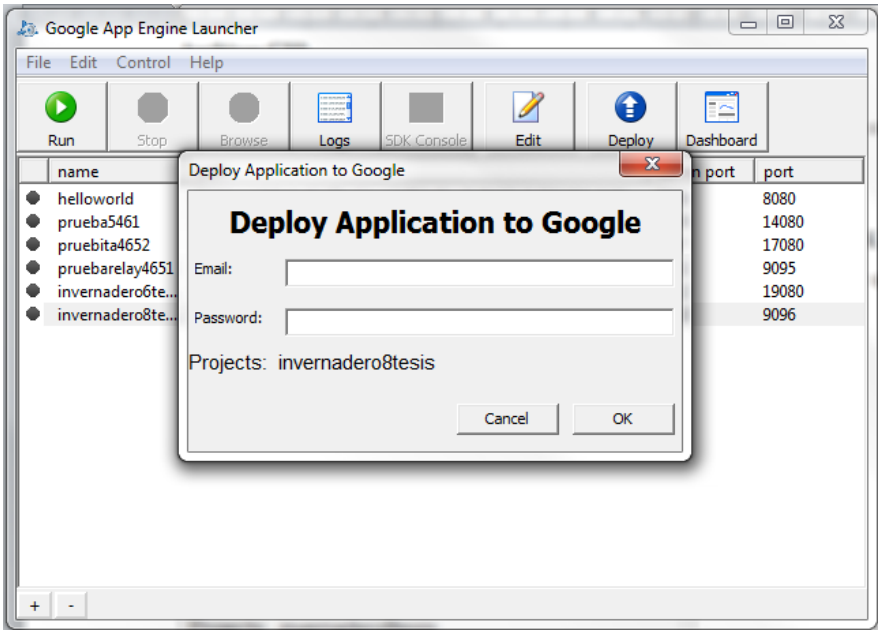

Figura 46. Subir aplicaciones con Google App EngineLauncher.

Fuente: (Elaboración Propia)

La aplicación fue diseñada dentro de la arquitectura Cliente – Servidor REST (*representationalstate transfer*), en donde el cliente inicia peticiones hacia el servidor, el servidor las procesa y retorna respuestas apropiadas. (Sierra & Cubo). En esta aplicación se realizan operaciones GET periódicamente para muestrear el estado de un recurso.

Para la programación, se utiliza el formato JSON (*JavaScript ObjectNotation*) para el intercambio de los datos en forma de listas, debido a

que es capaz de representar la información en un formato ligero, más rápido de transportar y que consume menos ancho de banda. Se emplea en entornos donde el tamaño de flujo de datos entre cliente y servidor es alto, como en este caso.(Benítez, 2010).

Para realizar las gráficas, tablas y medidores se utiliza la herramienta Google Charts que dispone de varios *frameworks* para generar gráficos personalizados, como por ejemplo gráficos de barras, de línea, mapas, diagramas circulares, etc.(Google Developers).

3.2.5 Procesamiento de datos

Como un ejemplo del procesamiento de datos que se puede realizar desde la nube, en la interfaz de usuario del prototipo se presentan los valores de la media y varianza de las variables medidas por los sensores (temperatura, luminosidad y humedad).

La media aritmética indica el valor característico que tienen los datos obtenidos de las variables que se están midiendo. Para su cálculo, se emplea la ecuación descrita en 2.8.2.

Los valores de la media se van presentando en la interfaz de usuario a medida que se recibe una muestra, es decir que se realiza un procesamiento continuo.

Para la obtención de la media, en la programación de la interfaz, se utilizan dos acumuladores, uno de ellos (Acumulador1) almacena los valores de las variables que se van midiendo, mientras que el otro acumulador (Acumulador2) se incrementa de acuerdo al número de muestras que se va recibiendo.

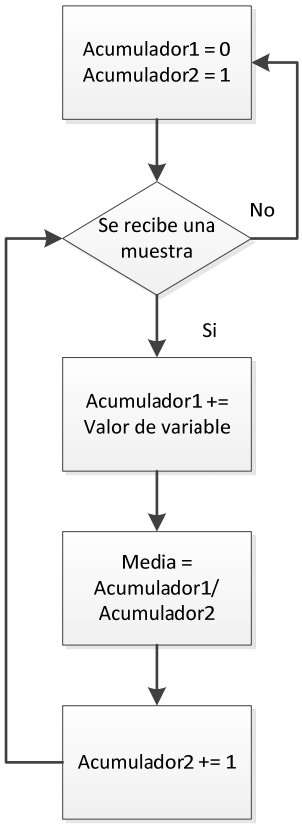

Figura 47. Diagrama de cálculo de la media.

Fuente: (Elaboración Propia)

La varianza indica la variabilidad de los datos medidos por medio de un número. A mayor valor de esta medida mayor variabilidad. En cambio, a menor valor, más homogeneidad. Para su cálculo, se emplea la ecuación descrita en 2.8.2, que corresponde a la varianza muestral, se utiliza esta fórmula debido a que no se conoce el valor total de las mediciones ya que se van presentando en la interfaz de usuario a medida que se recibe una muestra.

Para la obtención de la varianza, en la programación de la interfaz, se utilizan: el valor de la media y tres acumuladores, uno de ellos (Acumulador1) almacena los valores de las variables que se van midiendo, otro acumulador (Acumulador2) almacena los valores de los cuadrados de la diferencia del valor de la muestra menos su media, mientras que el otro acumulador (Acumulador3) se incrementa de acuerdo al número de muestras que se va recibiendo.

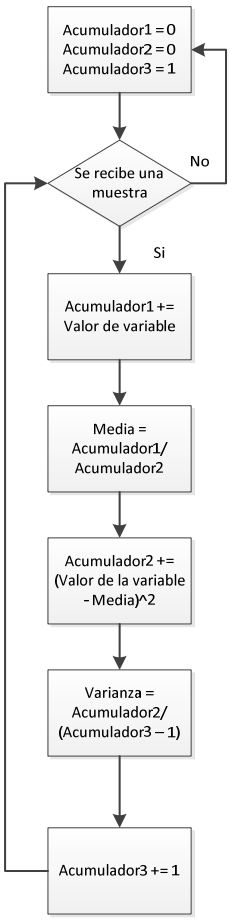

Figura 48. Diagrama de cálculo de la varianza.

Fuente: (Elaboración Propia)

3.3 DISEÑO E IMPLENTACIÓN DEL PROTOTIPO IOT BASADO EN DIGI

3.3.1 Arquitectura de la red

El prototipo de red IoTbasado en la plataforma DIGI consiste en la monitorización de un ambiente y el control de dispositivos. La red cuenta con cuatro nodos, un nodo coordinador, un nodo router de recolección de datos de temperatura y luminosidad. Y dos nodos finalesque se utilizan para el controlde

encendido y apagado de dispositivos. La comunicación entre los nodos es mediante Zigbee®.

El dispositivo coordinador recolecta los datos del ambiente, y los envía directamente a la nube Etherios*Device Cloud*. *Device Cloud* es una nube de dispositivos que ofrece un servicio de infraestructura diseñado para proporcionar la integración de aplicaciones en redes de dispositivos. *Device Cloud* permite integración de software de cliente, aplicaciones web o aplicaciones móviles para nube dispositivos. Con una cuenta gratuita, se pueden conectar hasta 5 dispositivos, cantidad suficiente para el desarrollo del presente prototipo. (Etherios).

La visualización de los datos de los sensores se puede realizar tanto en la interfaz de Etherios*Device Cloud*, que adicionalmente realiza el procesamiento de los mismos, como en una aplicación de cliente alojada en el servicio Google App Engine, que utiliza un servicio web (XML) para la comunicación con el dispositivo coordinador mediante el *Device Cloud*. En la interfaz se visualizan los datos de los sensores, y el tiempo la muestra en un gráfico. También se puede realizar el encendido o apagado de dispositivos.

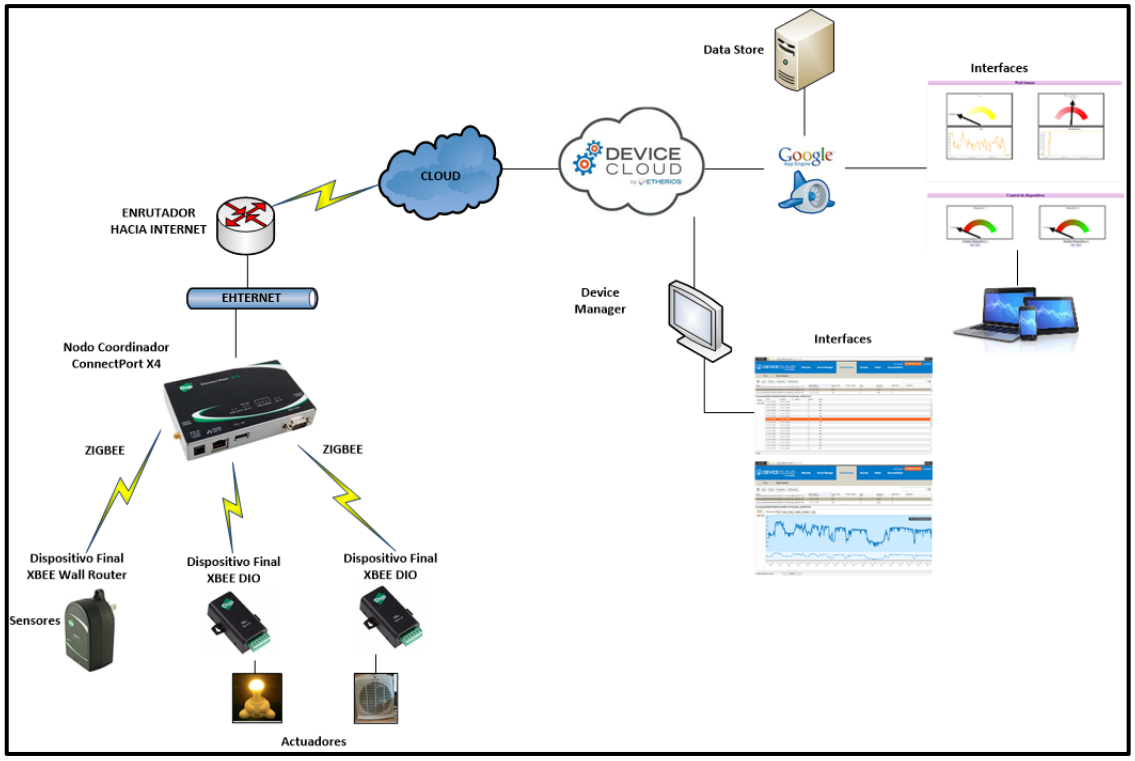

Figura 49. Arquitectura de la red.

Fuente: (Elaboración Propia)

3.3.2 Diseño de la red

La red inalámbrica de sensores Zigbee® está compuesta por cuatro nodos para el monitoreo del ambiente y el control de dispositivos. Se tiene un nodo Coordinador, un nodo router y dos nodos terminales. El nodo router se encarga de la medición de las variables de temperatura y luminosidad del ambiente, mientras que el resto de nodos terminales permiten el control de dispositivos.

La topología de red utilizada para este prototipo es una topología tipo estrella en la que todos los nodos terminales y router se conectan con el dispositivo coordinador.

Selección de componentes básicos

Para la selección de los componentes se tiene en cuenta las consideraciones indicadas en 3.2.2:

El coordinador y los dispositivos finales seleccionados corresponden a módulos XBEE Series 2. El nodo coordinador corresponde al Gateway Connect Port X4. Los dispositivos finalesdecontrolson los adaptadores XBEE DIO y utilizan una salida digital para el control de encendido y apagado. El nodo de monitorización es el dispositivo XBEE Wall Router que incluye los sensores de temperatura y luminosidad.

3.3.3 Diseño de circuitos de acondicionamiento

Sensores

Los sensores de temperatura y luminosidad, que se utilizan para la monitorización del ambiente, se encuentran embebidos en el dispositivo Wall router, por lo tanto no es necesaria la utilización de circuitos de acondicionamiento.

Actuadores

Los circuitos de acoplamiento utilizados por los dispositivos finales para el control de encendido y apagado se han diseñado de igual forma que en el prototipo basado en Arduino, como se explica en 3.2.3.

3.2.4 Implementación de la red

Configuración de los nodos

La configuración de los nodos finales se realiza mediante el software X-CTU, el nodo coordinador se configura mediante su interfaz web y el nodo router no requiere configuración.

Nodo Coordinador

El nodo coordinador es el Gateway Connect Port X4, los parámetros más importantes configurados en su interfaz web son:

- PAN ID: 111

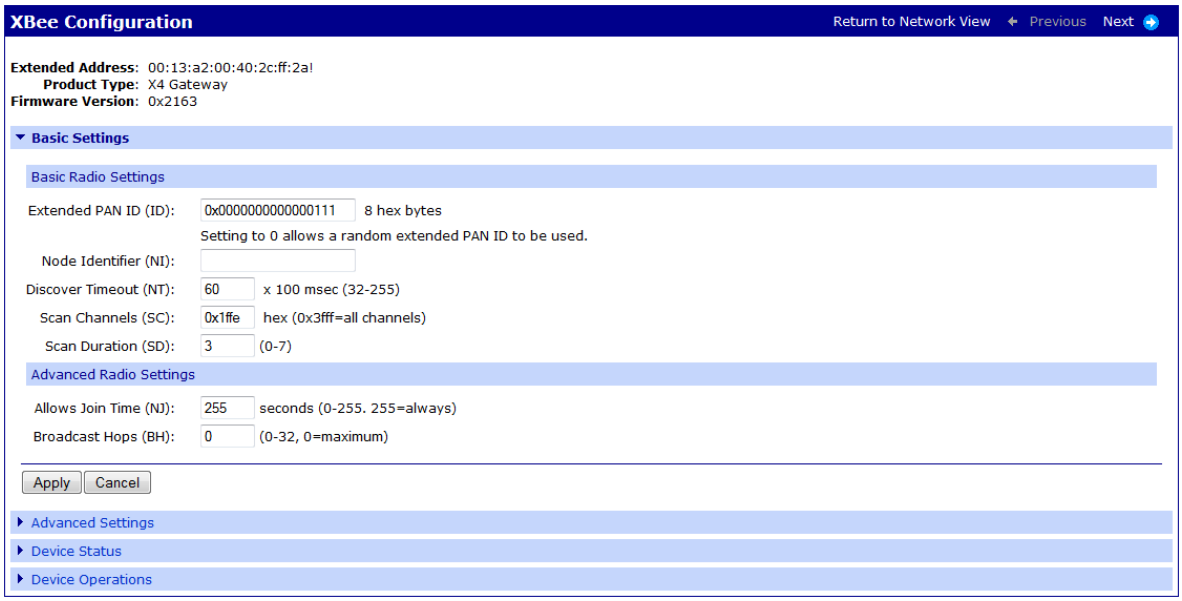

Figura 50. Configuración coordinador en la interfaz web.

Fuente: (Elaboración Propia)

Nodos Finales

La configuración de los dispositivos finales de control es la siguiente:

- *Function Set:* ZIGBEE *END DEVICE AT.*
- PAN ID: 111 (El mismo valor del coordinador).
- *ChannelVerification:* 1.

Se debe encender primero el nodo coordinador y después los dispositivos finales y router para que elijan el mismo canal en el que trabaja el coordinador, para este caso los nodos trabajan en el canal F.

Una vez realizada la configuración, los dispositivos se van a asociar a la red y se puede visualizar en la interfaz web del coordinador todos los nodos que forman parte de la red.

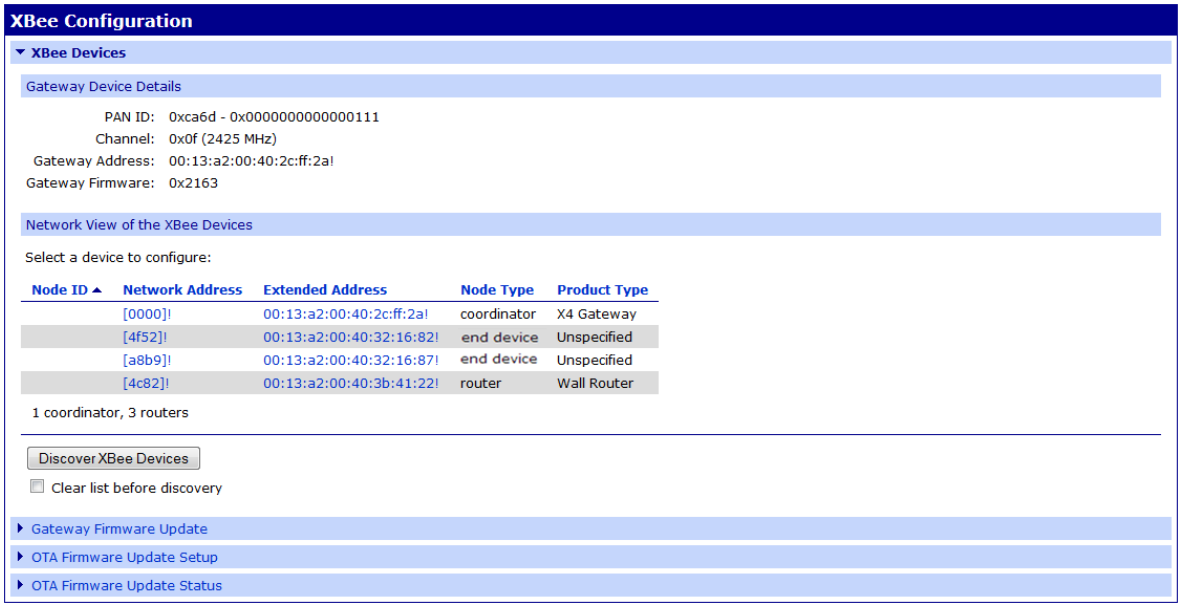

Figura 51. Red de dispositivos XBee.

Fuente: (Elaboración Propia)

Integración del Hardware

El nodo coordinador Connect Port X4 se conecta mediante Ethernet a un dispositivo de red que le provea internet.

El nodo Wall router se conecta al tomacorriente y se espera a que el indicador de asociación se active.

El montaje de los nodos finales de control se realizó de igual manera que la red basada en Arduino, como se explica en 3.2.4.

Conexión del coordinador al *Device Cloud*

Para conectar el nodo coordinador a la nube Etherios*Device Cloud* es necesario crear una cuenta en www.etherios.com. Esta cuenta es gratuita y permite la conexión de 5 dispositivos.

En la ventana de *Device Manager* se añade el Connect Port X4 mediante su dirección MAC. Los dispositivos asociados a este coordinador se los puede observar en la pestaña *XBee Networks.*

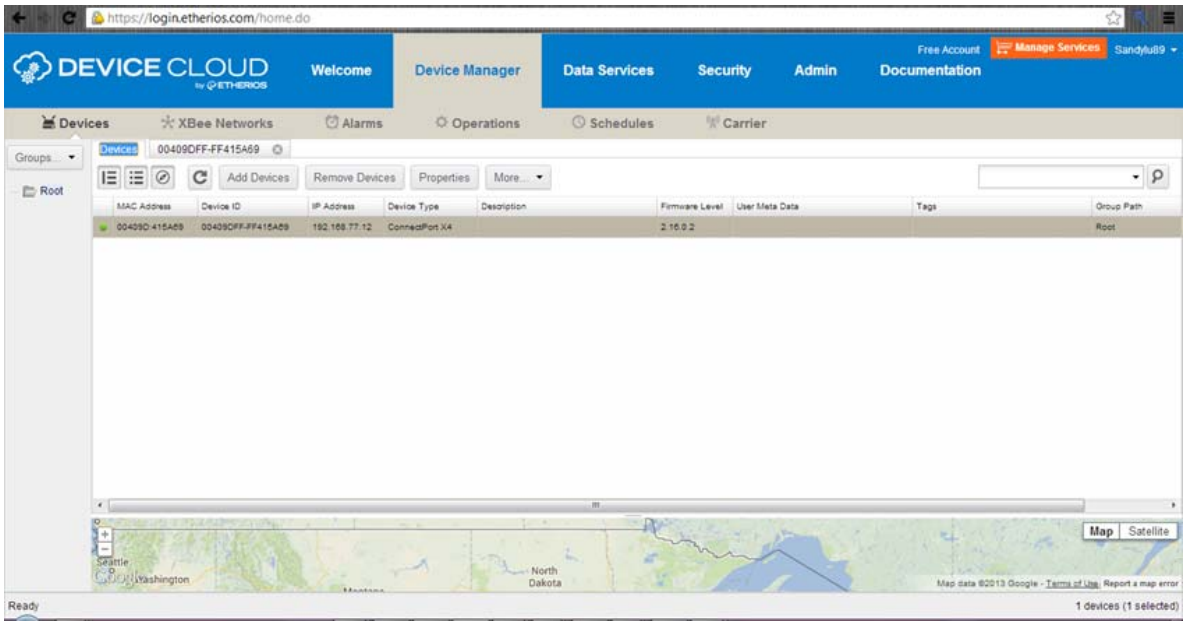

Figura 52. Coordinador en la interfaz *Device Cloud*.

Fuente:(Elaboración Propia)

El programa utilizado para el manejo de los dispositivos conectados al coordinador es DIA, que emplea el lenguaje de programación Python. La herramienta que permite crear la aplicación DIA es Digi ESP forPython, el cual es un IDE que simplifica la programación. Esta herramienta puede ser descargada de la página http://www.digi.com/gatewaydevelopmentkit.

Se crea un nuevo proyecto iDigiDia, que incluye los drivers del XBee Wall Router (dispositivo router) y de los XBee DIO (dispositivos finales). Los dispositivos añadidos deben ser asociados con su respectiva dirección MAC.

El proyecto debe incluir la presentación *RCI Handler*que permite la comunicación entre el Gateway y el *Device Cloud.*

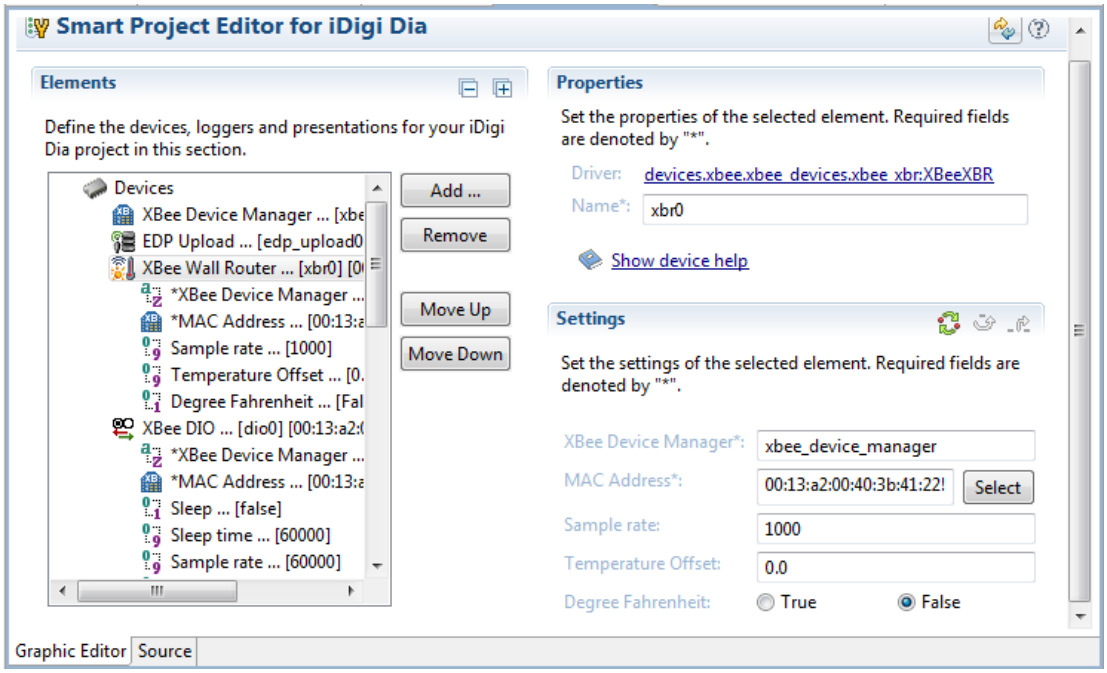

Figura 53. Proyecto iDigiDia.

Fuente: (Elaboración Propia)

Al ejecutar el proyecto, se carga en el Gateway automáticamente y despliega una interfaz que permite la interacción y visualización de los dispositivos asociados a la red.

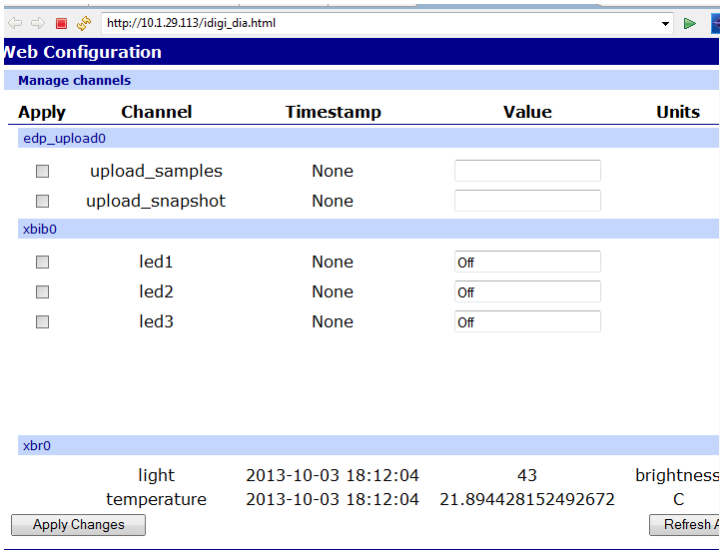

Figura 54. Interfaz web del proyecto iDigiDia.

Fuente: (Elaboración Propia)

Descripción de la Interfaz *Device Cloud*

Los datos generados por los dispositivos conectados al coordinador se pueden visualizar en la pestaña *Data Streams*del Etherios*Device Cloud.*

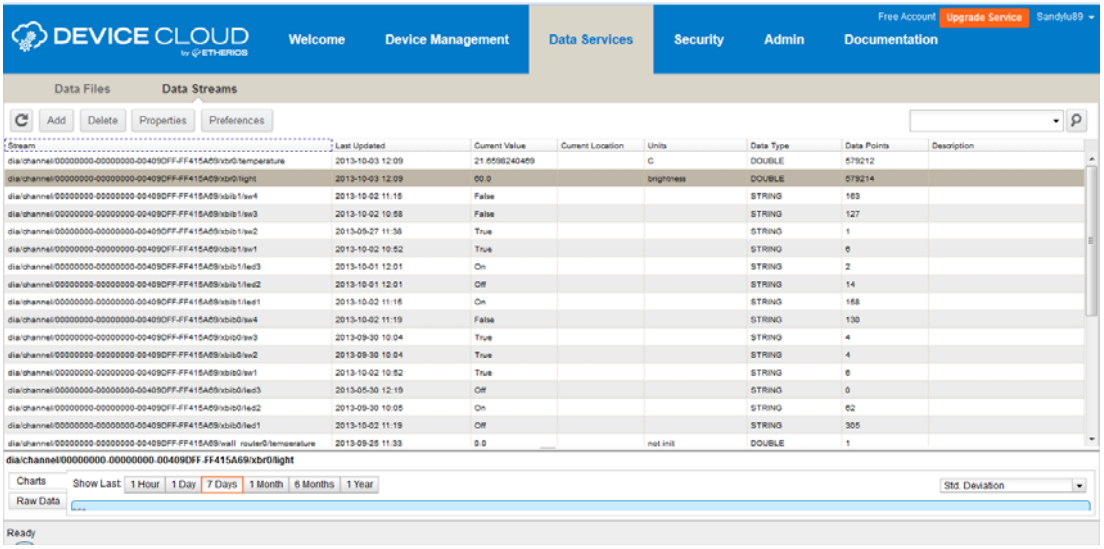

Figura 55. Dispositivos conectados al coordinador con sus valores respectivos.

Fuente: (Elaboración Propia)

El *Device Cloud* realiza un procesamiento de los datos obtenidos y los presenta en gráficas. Los tipos de procesamiento que permite son: Promedio, Sumatoria, Valor mínimo, Valor máximo y Desviación Estándar.

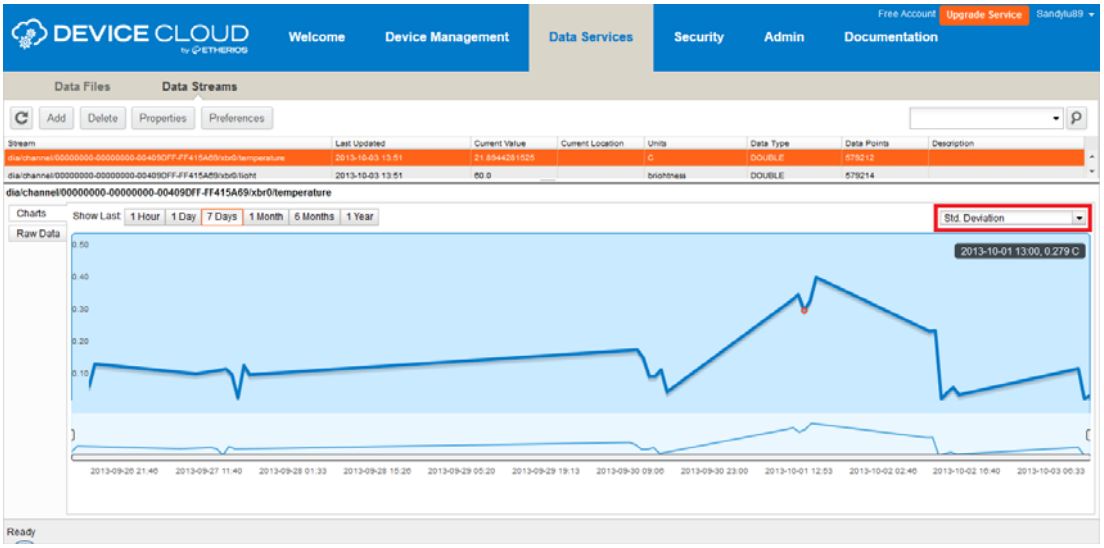

Figura 56. Gráfica de valores de los dispositivos conectados.

Fuente: (Elaboración Propia)

Descripción de la aplicación de cliente

La interfaz alojada en Google App Enginecontiene dos secciones, uno que monitorea la temperatura y luminosidad del ambiente y un segundo bloque que controla dos dispositivos.

En la primera sección se visualizan dos medidores, uno con el valor actual de la temperatura y otro con el valor actual de luminosidad. El rango de los medidores va de 0 a 100. Las unidades son grados centígrados y nivel de luminosidad.

También muestra gráficas correspondientes a los valores medidos de temperatura y luminosidad y el tiempo en el que se adquirieron las muestras.

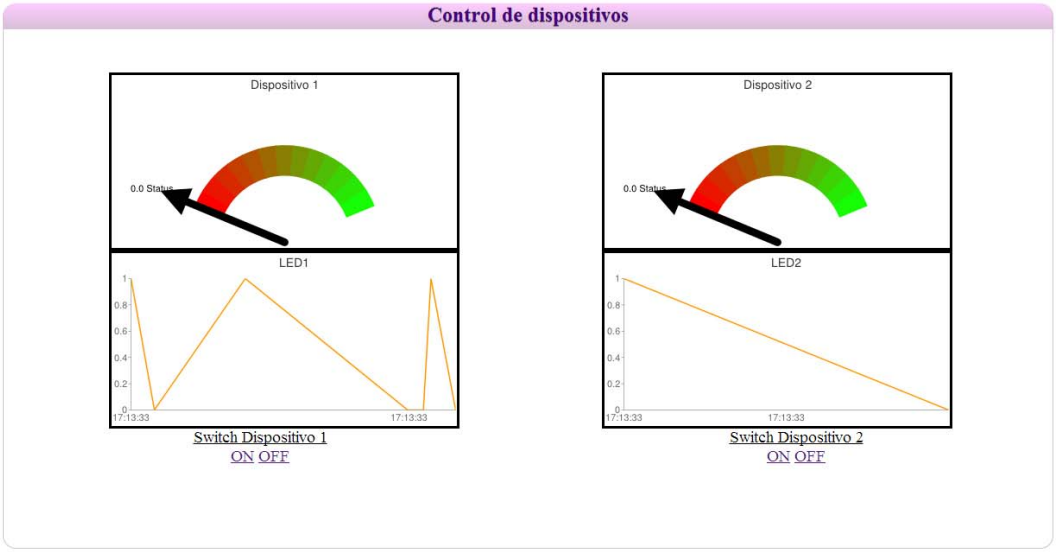

Figura 57. Sección de monitorización en la interfaz web.

Fuente: (Elaboración Propia)

En la segunda sección se visualizan dos medidores con el estado actual de los dispositivos. El valor de 1 representa el estado encendido y el valor 0 representa el estado apagado. También muestra gráficas correspondientes a los valores de los estados.

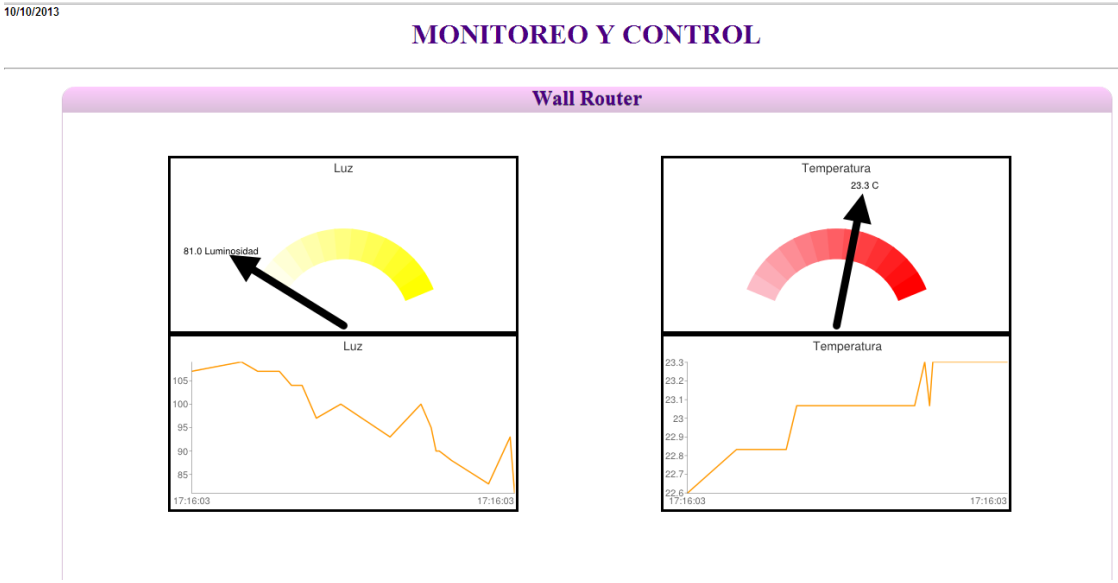

Figura 58. Sección de control en la interfaz web.

Fuente: (Elaboración Propia)

Para poder acceder a la interfaz se debe colocar la siguiente dirección: www.monitoreoycontroldigi.appspot.com.

Programación de la Interfaz de usuario en Google App Engine

El desarrollo de la interfaz para este prototipo se realiza en el lenguaje de programación Python.

La aplicación fue diseñada al igual que en la interfaz de usuario desarrollada para la plataforma Arduino, en donde el cliente inicia peticiones POST hacia el servidor, el servidor las procesa y retorna respuestas apropiadas. Para realizar las gráficas y medidores se utiliza la herramienta Google Charts.(Digi International Inc., 2010).

La aplicación se comunica con Etherios*Device Cloud* para obtener los valores de los dispositivos mediante el método *RCI Request,* que permite controlar y monitorear remotamente los dispositivos conectados a esta nube. Utiliza XML y HTTP para el intercambio de información. Se envíanpeticiones HTTP POST hacia la URL de la nube http://developer.idigi.com/ws/sci junto con el nombre de usuario y la contraseña de la cuenta creada.

En las peticiones RCI se especifica el ID del dispositivo Connect Port X4, los parámetros del dispositivo que con el que se va a interactuar, ya sea el Wall Router o los XBee DIO, la versión del RCI y el target idigi_dia, que es el mismo de la aplicación que se está ejecutando en el Gateway.

3.4 DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID

Combinar el IoT con un sistema operativo abierto como Android, permite potencialmente grandes oportunidades de conectarse a cualquier cosa y en cualquier lugar mediante el internet. Android admite el desarrollo, autonomía y flexibilidad de la aplicación la cual puede estar instalada en cualquier dispositivo

móvil de forma gratuita. De esta manera le permite al usuario una mayor movilidad y un control remoto de sus dispositivos.

3.4.1 Descripción del desarrollo de la aplicación.

El desarrollo de la aplicación tiene por objetivo el control de luminosidad de un entorno, permitiendo regular la cantidad de luz (*dimmer*), el encendido y apagado de luces desde un dispositivo móvil Android, dentro de una red local, mediante una conexión Wifi y tarjetas Arduino.

La aplicación se creó en la plataforma JavaScript Appcelerator, mediante el software Titanium Studio, que es un IDE basado en Eclipse, permite el desarrollo de aplicaciones para los sistemas operativos iOS, Android, BlackBerry, Tizen, Denso, entre otras. (Appcelarator Inc., 2008). Debido a que la aplicación se desarrolló para dispositivos Android, se requiere del SDK de Android.

La plataforma Arduino se encarga de la comunicación entre la aplicación móvil y los dispositivos, proveyendo de una conexión Ethernet hacia la red local, por la cual se envían los datos de peticiones HTTP.

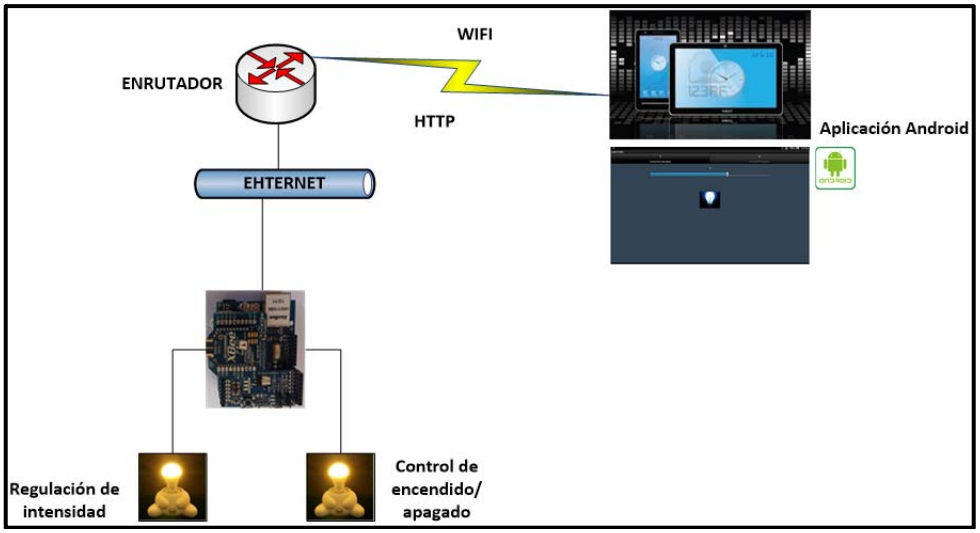

Figura 59. Arquitectura de la red para la aplicación Android.

Fuente: (Elaboración Propia)

3.4.2 Plataforma Arduino

Para la conexión de los dispositivos a la red, se requiere de una tarjeta Arduino UNO que contiene la programación para el control de iluminación, recibiendo las peticiones HTTP desde la aplicación y enviando sus respectivas respuestas. La tarjeta Arduino Ethernet *Shield* le permite obtener una dirección IP dentro de la red local para poder acceder desde el dispositivo móvil.

Programación del microcontroladorArduino

Las librerías utilizadas en el programa son: SPI.h y Ethernet.h, las cuales ya fueron descritas en el **CAPITULO 3** en la sección 3.2.

Las funciones realizadas en el programa son las siguientes:

- **Setup():** En esta función se declaran los puertos necesarios correspondientes a la tarjeta Arduino como salidas. Se inicializa el puerto serial con la velocidad de transmisión en 9600, también inicializa la librería Ethernet y la configuración de red. Se emplea la función propia de Arduino attachInterrupt(), que especifica una función a la cual llamar cuando una interrupción externa ocurre. Esta función es necesaria para la regulación de la intensidad de luz.

- **Loop():** Esta función se ejecuta de forma cíclica. Crea un cliente Ethernet para la lectura de las peticiones HTTP, que, dependiendo de su contenido ejecuta acciones de control. Llama a la función stringToInt().

- **stringToInt():** Es una función que permite convertir una cadena String en un número entero.

Para la regulación de la intensidad de luz se debe considerar lo siguiente:

La potencia de alimentación de una bombilla está directamente relacionada con una onda senoidal, la bombilla se puede regular, permitiendo que fluya sólo una parte de la señal. Por lo tanto, se necesita de un punto de referencia en la señal desde donde calcular cuando la lámpara tiene que estar encendida. El punto de referencia que se utiliza es el cruce por cero.

Así que lo que el programa realiza es la detección del cruce por cero, y luego esperar a una determinada cantidad de tiempo para encender el TRIAC, que es el elemento utilizado en el circuito del *dimmer*.

La frecuencia de corriente alterna en Ecuador es de 60Hz, cada onda senoidal tiene un período de 1/60Hz = 16,66 ms. Como hay 2 picos en una onda, significa que después de cada detección de cero existe un período de 8.33 ms que se puede regular. Como se usa un TRIAC, el programa tiene que esperar al punto cero en la onda y luego esperar una cantidad específica de tiempo dentro de ese período de 8.33 ms para enviar un pulso al TRIAC. (Autodesk Inc., 2013).

En la función **zero_cross_int()**, a la cual el programa salta a después de la interrupción se determina el tiempo que hay que esperar antes de disparar el TRIAC. Para ello, se eligió la cantidad arbitraria de 123 pasos. Eso significa que cada paso es 8.33ms/123 = 67us. El tiempo de regulación total, entonces se calcula a partir de 67x (1 a 123). El número entre 1y 123, determina el nivel de luminosidad, este valor lo envía la aplicación Android.

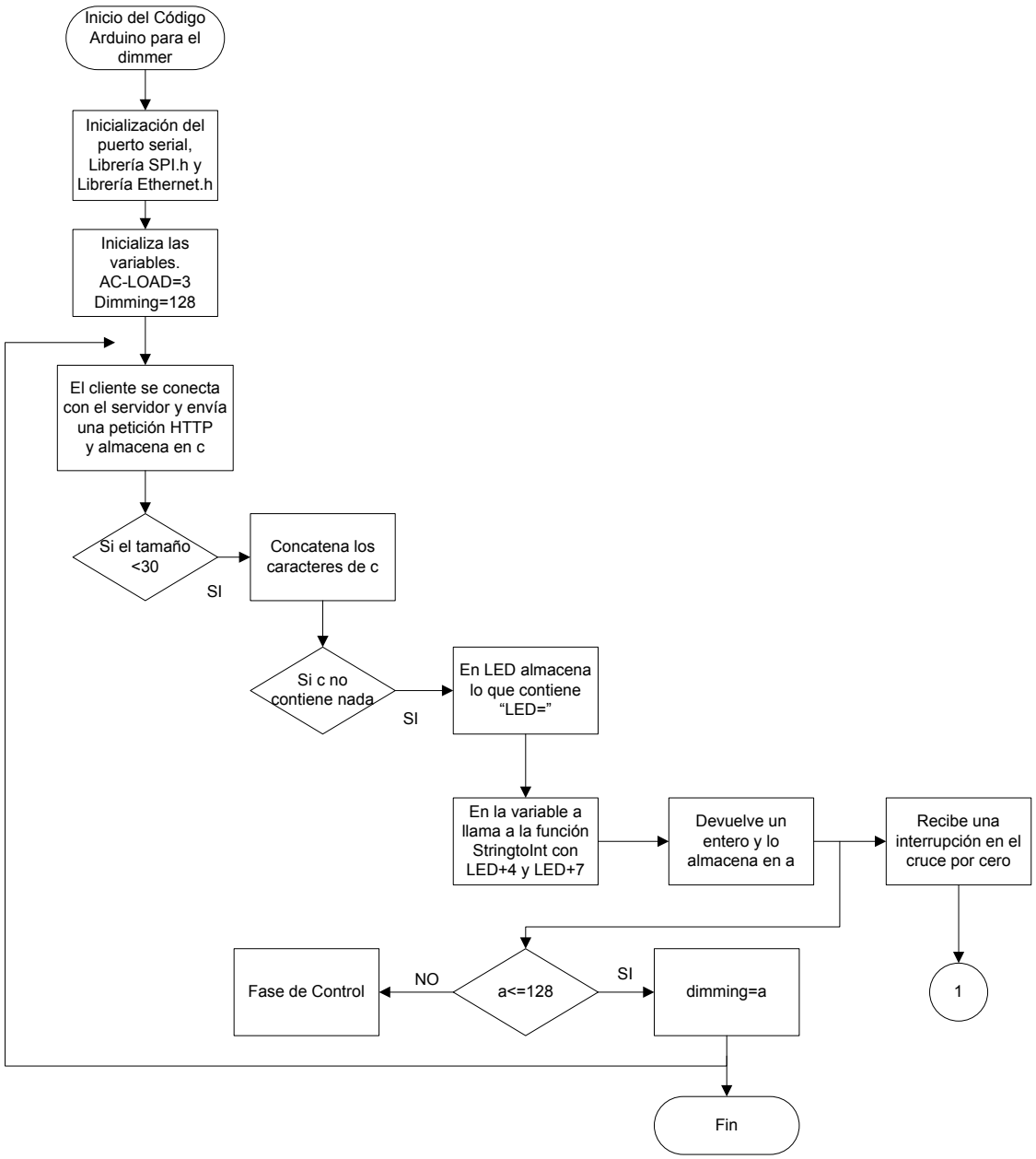

Figura 60. Diagrama de flujo de la regulación de intensidad de luz.

Fuente: (Elaboración Propia)

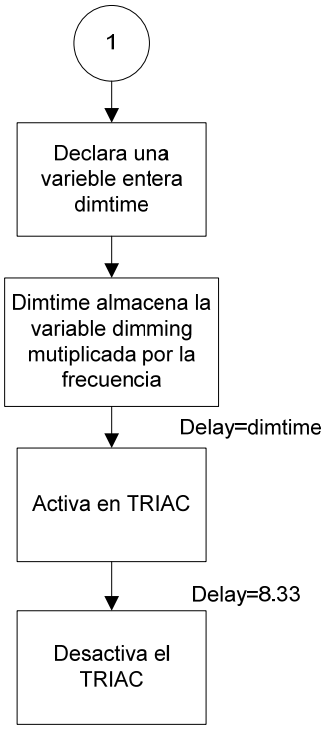

Figura 61. Diagrama de flujo de la función que se ejecuta al producirse una interrupción.

Fuente: (Elaboración Propia)

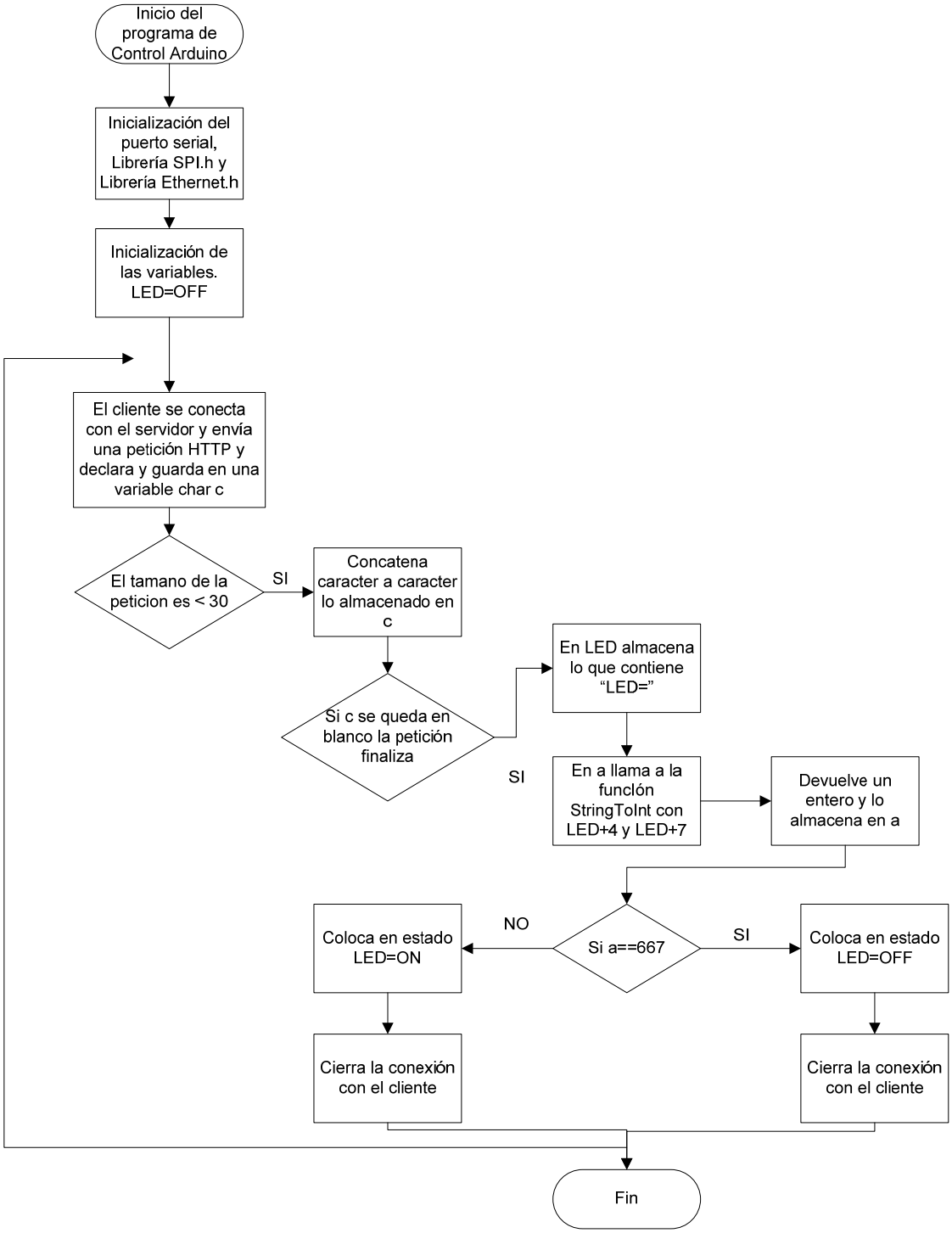

Figura 62. Diagrama de flujo del control de encendido/apagado.

Fuente: (Elaboración Propia)

Circuitos de acondicionamiento

Para la implementación del circuito correspondiente al control de la intensidad de luz se utilizaron los siguientes elementos:

- 1 Resistencia de: 10k, 220 y 470 Ω.
- 2 Resistencias de 30kΩ.
- 1 Opto 4N25.
- 1 Optoacoplador MOC3021.
- 1 Puente rectificador de diodos de 400 V.
- 1 Triac TIC206.

El esquema utilizado, se muestra a continuación. (Autodesk Inc., 2013).

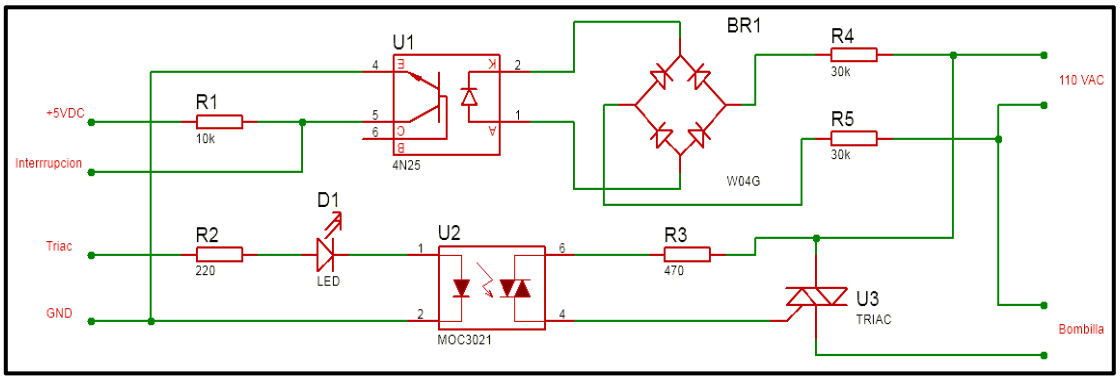

Figura 63. Circuito de acondicionamiento *dimmer*.

Fuente: (Elaboración Propia)

El circuito de acondicionamiento para el control de encendido y apagado de la bombilla es similar al utilizado en las plataformas de red tradicionales implementadas en el **CAPÍTULO 3**, se encuentra detallado en 3.2.3.

3.4.3 Desarrollo de la aplicación Android en Titanium Mobile

La aplicación para dispositivos móviles que realizará el control de iluminación del entorno consta de dos pestañas. La primera pestaña se encarga de la regulación de la intensidad de luz, para lo cual cuenta con un slider que permite variar la intensidad entre los valores de 1 a 123, donde 1 corresponde a la bombilla encendida en su totalidad y 123 a la bombilla apagada.

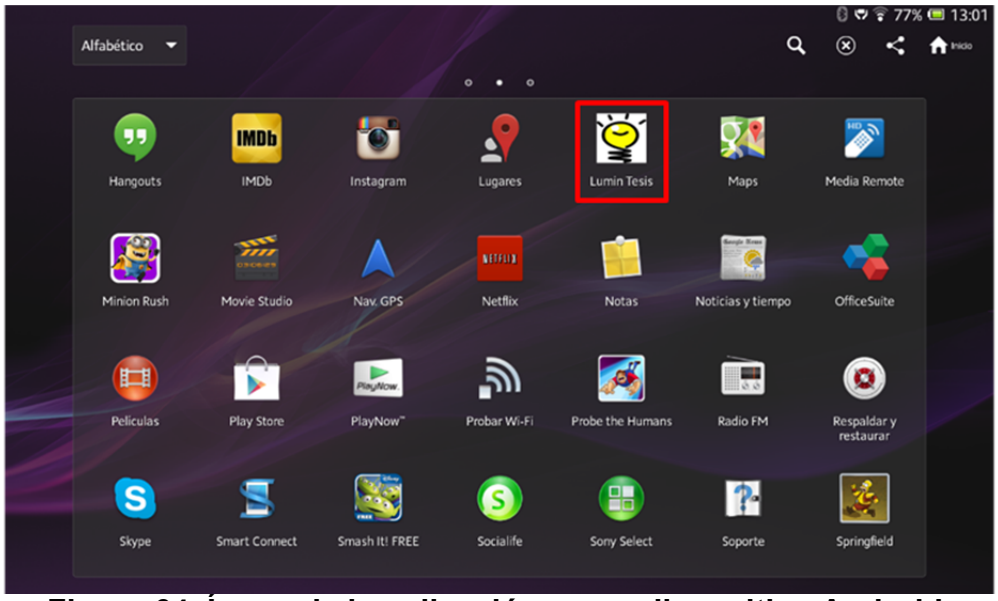

Figura 64. Ícono de la aplicación en un dispositivo Android.

Fuente:(Elaboración Propia)

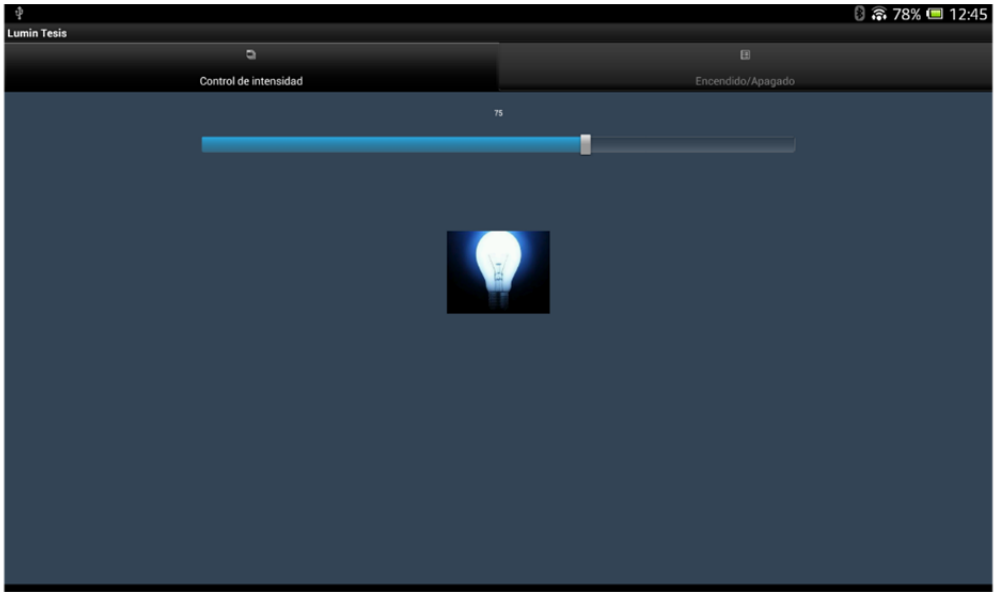

Figura 65. Pestaña de regulación de la intensidad de luz.

Fuente: (Elaboración Propia)

La segunda pestaña realiza el control de encendido y apagado de una bombilla, por lo tanto se utilizan dos botones, al hacer clic en el botón ON se enciende la bombilla y se presenta una imagen que indica este estado y al hacer clic en el botón OFF se apaga la bombilla y se muestra la imagen con el correspondiente estado.

Los eventos utilizados en la aplicación son los siguientes:

- **slider.addEventListener('change', function(e){}:** El evento que produce el movimiento del slider es del tipo ‘change’, al detectar este evento, el programa imprime el valor en el que se encuentra el slider después del cambio y envía este valor mediante una petición HTTP GET hacia la URL correspondiente a la tarjeta Arduino.

- **boton1/2.addEventListener('click', function(){}:** Al hacer clic en uno de los botones se envía este estado mediante una petición HTTP GET hacia la URL de la tarjeta Arduino y se presenta la imagen del estado correspondiente.

Al ejecutar la aplicación en un emulador Android se crea el archivo app.apk que puede ser trasladado a un dispositivo con este sistema operativo para que una vez instalado se pueda hacer uso de la aplicación. Siempre que el dispositivo se encuentre en la misma red de la tarjeta Arduino.

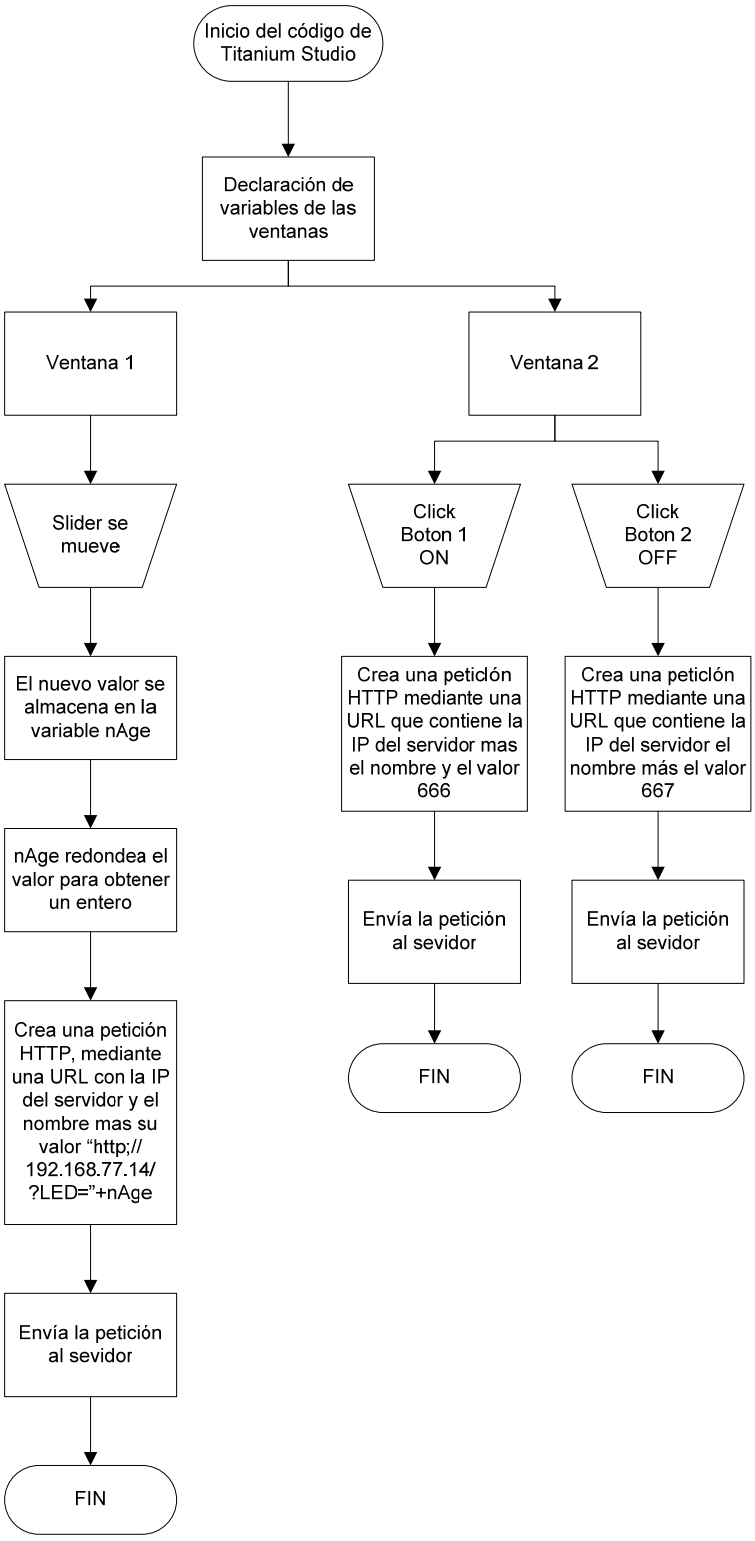

Figura 66. Diagrama de flujo del programa de la aplicación

Android. Fuente:(Elaboración Propia)

CAPÍTULO 4

DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA DE RED IPv6

En el presente capítulo se describen los requerimientos del sistema para la plataforma de red IPv6 de acuerdo al modelo de referencia del IoT. Se muestra el diseño de la red en cuanto a elementos, arquitectura y el desarrollo de la interfaz web de usuario. Se presentan las alternativas de acceso a la interfaz web, ya sea dentro de una red nativa IPv6 o a través de un túnel IPv6 sobre IPv4.

4.1 REQUERIMIENTOS DEL SISTEMA

- **Sistemas inteligentes de detección y recolección de datos:** deben servir de base para las actividades automatizadas tomando información del entorno. Los tipos de sistemas requeridos por el *Internet of things* varían dependiendo de la aplicación. En este proyecto se van a utilizar máquinas virtuales que simulan una red de sensores, los cuales permiten medir algunos parámetros de su propio sistema.

- **Conectividad de extremo a extremo:** La arquitectura IoT con IPv6 debe admitir conexiones de extremo a extremo entre objetos situados en diversas ubicaciones geográficas. Las conexiones pueden ser alámbricas y/o inalámbricas, permitiendo la transferencia de información generada por los sistemas inteligentes. En esta aplicación, la comunicación entre los nodos y el internet IPv6 es Ethernet. La comunicación debe ser continua y fiable en la transmisión de datos.

- **Compatibilidad de Direccionamiento:** La arquitectura IoT debe ser capaz de utilizar las direcciones IPv6 para identificar e interconectar sus diversos componentes.

- **Compatibilidad de Internet:** La arquitectura IoT debe permitir que los componentes ubicados en diferentes partes del mundo puedan interactuar a través de Internet mediante una red nativa IPv6 o por medio de túneles IPv6 sobre IPv4. La arquitectura debe permitir el acceso ubicuo a sus dispositivos y servicios. El sistema debe tener la capacidad para actualizar el software y la realización de otros procedimientos como mantenimiento.

- **El sistema debe proporcionar una API:** Es un software que permite que varios dispositivos heterogéneos interactúen entre sí y con la

infraestructura a su alrededor, realizando la automatización de procesos. La interfaz de esta plataforma utiliza el lenguaje de programación PHP y HTML y se encuentra alojada en un servidor web Apache.(Mandat International, 2012).

4.2 DISEÑO E IMPLEMENTACIÓN DE LA ARQUITECTURA DE LA RED IPv6

4.2.1 Arquitectura de la red

La arquitectura basada en dispositivos IPv6 aplicada al modelo de referencia IoT, consiste en la monitorización del estado de dispositivos, como sensores y actuadoresdentro del modelo de referencia.

Se dispone de tres nodos, que se representan mediantemáquinas virtuales debido a escases de sensores IPv6, de los cuales se obtienen datos de estado de memoria, estado de disco, procesos ejecutados, estado de puertos y se realiza un control de estos dispositivos cerrando procesos activos. Los nodos se conectan a una red IPv6 a través de RedCLARA, para este proyecto se utilizó la red CEDIA, que provee direcciones IPv6 mediante las cuales pueden acceder a los servicios de internet.

El control de los dispositivos y la visualización de los datos se realiza de dos formas: la primera consiste en una interfaz web, que se encuentra alojada en un servidor Apache conectado a la red de CEDIA, que no corresponde a ninguno

de los nodos que son monitorizados, la cual concentra la información de todos los dispositivos finales. En la segunda forma la interfaz web se encuentra en servidores Apache en cada uno de los dispositivos y se accede a ella mediante la dirección IPv6 de cada nodo. En la interfaz se pueden visualizar los datos de los parámetros ya mencionados de los dispositivos IPv6 y un gráfico que plasma estos valores. También se puede realizar el control de los procesos que se encuentran ejecutando los dispositivos.

Se puede acceder al servicio web mediante dos formas:

- El usuario puede estar conectado directamente al internet mediante una dirección IPv6. Es decir la red entre el usuario y el servicio es IPv6 nativa.

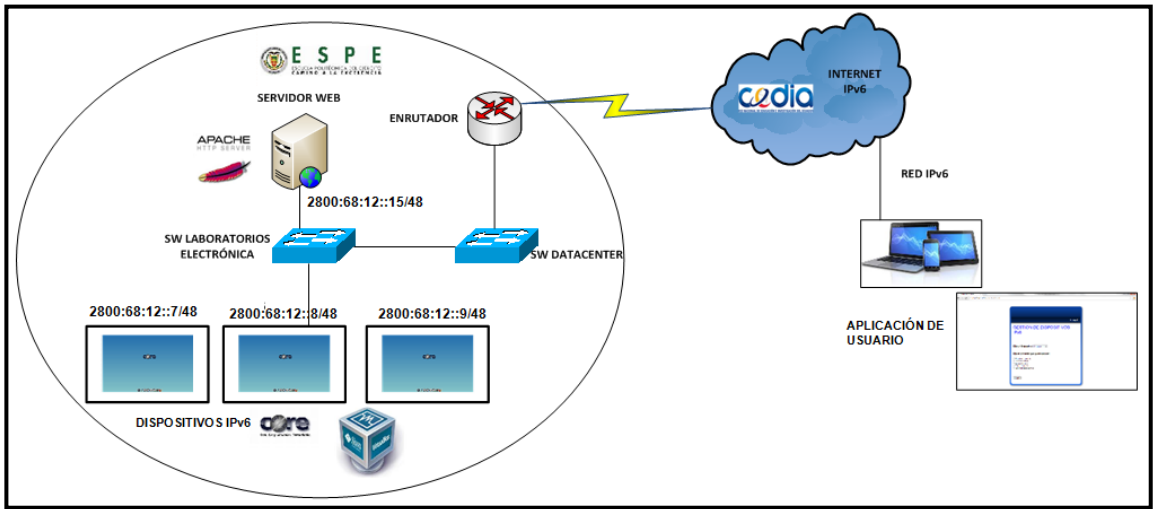

Figura 67.Arquitectura de la red con acceso al servidor externo desde una red IPv6 nativa.

Fuente: (Elaboración Propia)

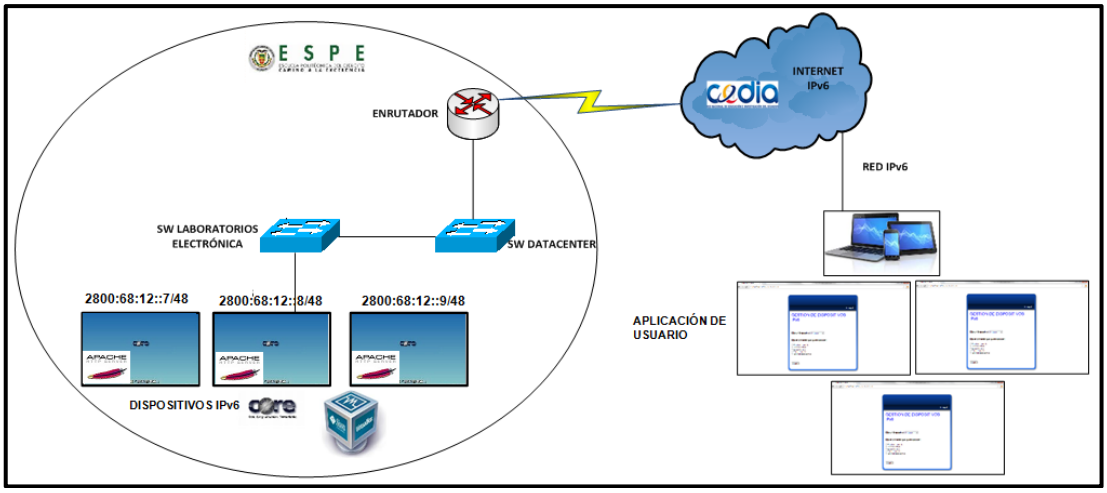

Figura 68.Arquitectura de la red con acceso al servidor del nodo desde una red IPv6 nativa.

Fuente: (Elaboración Propia)

- El usuario puede estar conectado al internet IPv6 mediante un proveedor de túneles de red IPv6, que permiten la conectividad sobre una infraestructura de red IPv4.

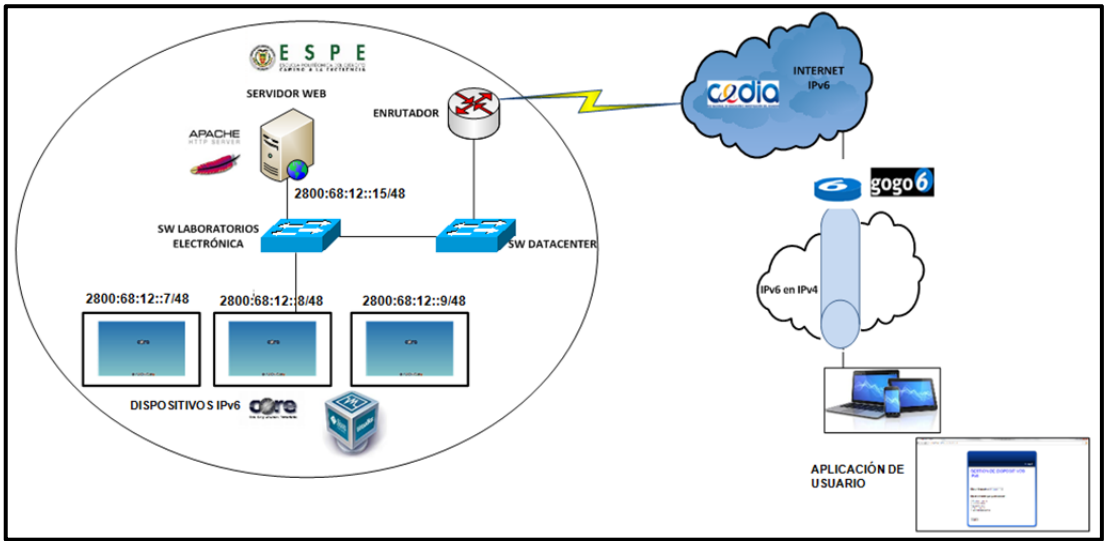

Figura 69. Arquitectura de la red con acceso al servidor externo desde una red con túnel IPv6 sobre IPv4.

Fuente: (Elaboración Propia)

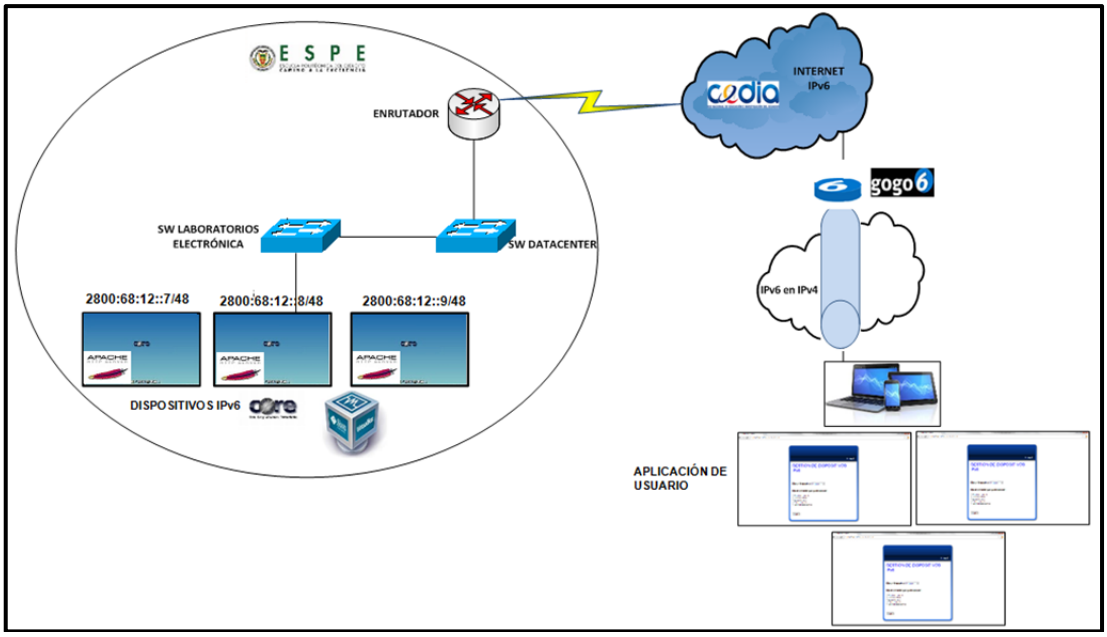

Figura 70. Arquitectura de la red con acceso al servidor del nodo desde una red con túnel IPv6 sobre IPv4.

Fuente: (Elaboración Propia)

4.2.2 Implementación de la red

La red de dispositivos IPv6 está compuesta por tres nodos, que corresponden a máquinas virtuales con el sistema operativo TinyCore, que es una distribución minimalista de Linux, cuyo tamaño es de 10MB, es un software libre, de código abierto, que se caracteriza por ser rápido, potente y flexible. Es posible utilizarlo como software para pequeños dispositivos que se pueden embeber. Tiene soporte IPv6 e incluye un servidor SSH. (The Core Team, 2009).

Configuración IPv6 en TinyCore

Para habilitar el soporte IPv6, es necesario instalar el paquete ipv6-3.8.10-tinycore.tcz, cargar el módulo al kernel y hacerlo persistente.

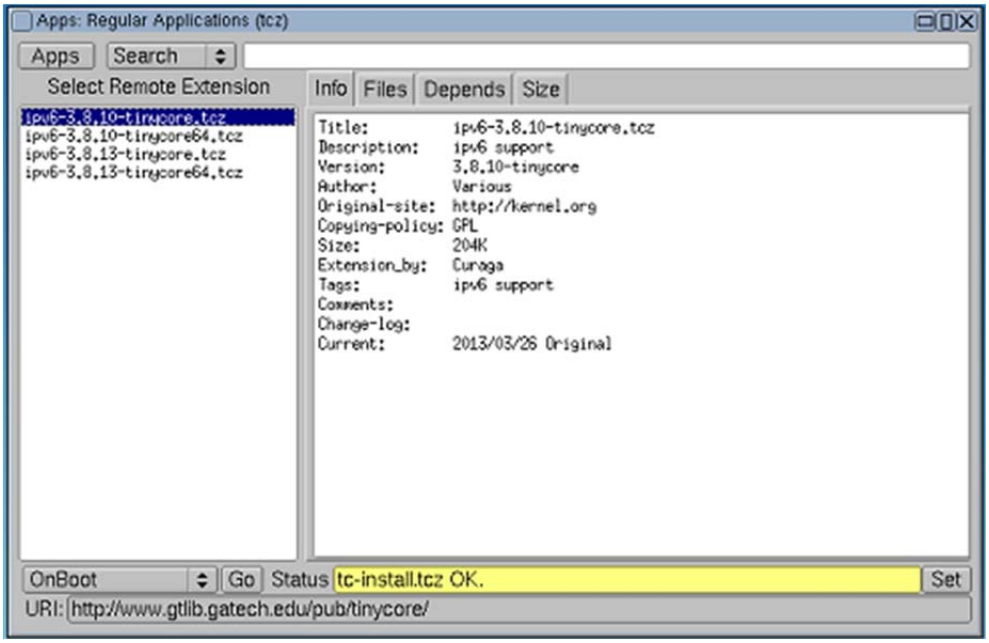

Figura 71. Instalación del paquete IPv6.

Fuente: (Elaboración Propia)

Se debe asignar una dirección IPv6 del rango otorgado por CEDIA a la ESPE, que es 2800:68:12::/48.

Las direcciones asignadas a los dispositivos son las siguientes:

- Dispositivo1: 2800:68:12::7/48.
- Dispositivo2: 2800:68:12::8/48.

- Dispositivo3: 2800:68:12::9/48.

En la consola de comandos se comprueba el funcionamiento de IPv6.

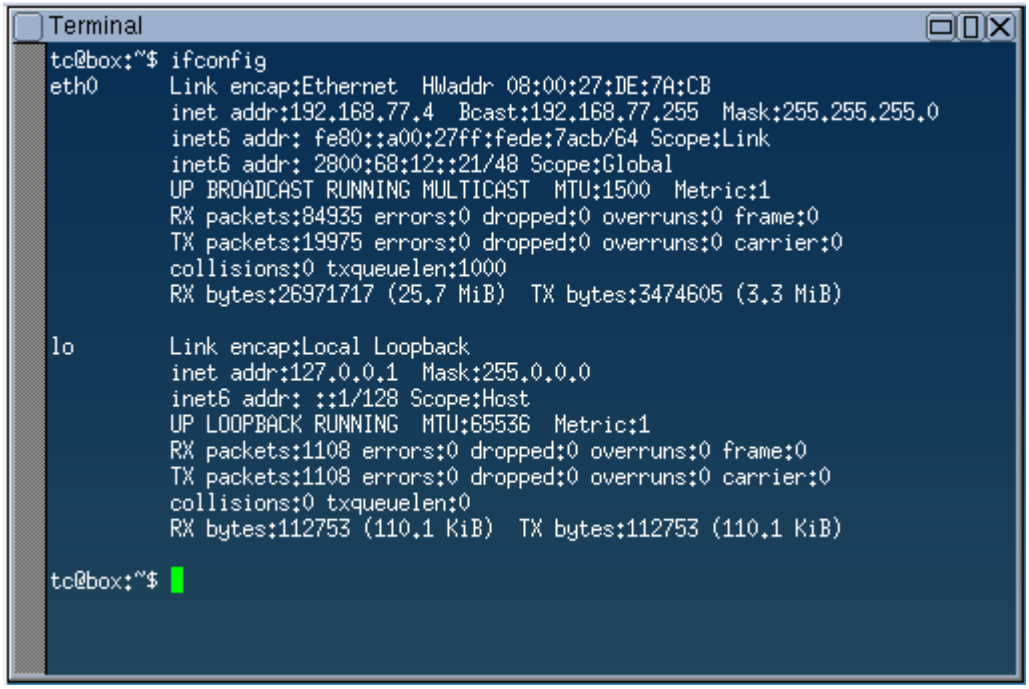

Figura 72. Configuración de red de la máquina virtual TinyCore Linux.

Fuente: (Elaboración Propia)

Servidor SSH en TinyCore

El acceso a las máquinas virtuales TinyCore desde la interfaz web, se lo realiza mediante un servidor SSH, para lo cual es necesario instalar y ejecutar OpenSSH en cada máquina virtual.

Servidor Web en TinyCore

La interfaz web de cada nodo se encuentra alojada en servidores Apache, por lo cual es necesario instalar y ejecutar Apache cada máquina virtual. La interfaz se desarrolla en el lenguaje de programación PHP que se instala junto con Apache. Para la conexión SSH se requiere la instalación de la librería LibSSH2.

El servicio SSH utiliza el puerto TCP 22 y el servicio web utiliza el puerto TCP 80, por lo tanto se debe verificar en la consola de comandos de cada dispositivo que estos puertos se encuentren activos o en estado *LISTEN*.

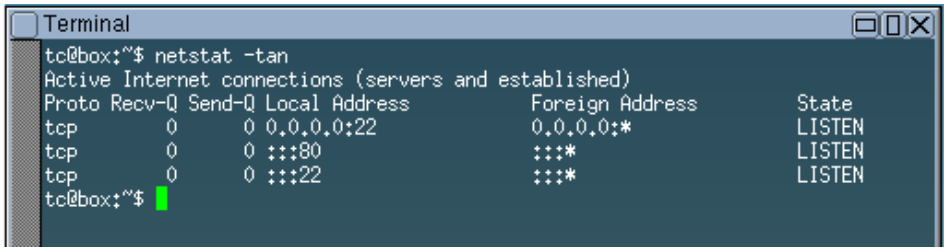

Figura 73. Estado de puertos.

Fuente:(Elaboración Propia).

4.3 DISEÑO DE LA INTERFAZ DE VISUALIZACIÓN

La descripción de la interfaz de usuario que se indica a continuación se aplica tanto en el servidor externo como en los servidores de cada nodo. Para acceder a la interfaz de usuario del servidor externo se debe ingresar la

dirección IPv6 del mismo (2800:68:12::15). Si se desea ingresar directamente al dispositivo es necesario colocar la dirección IPv6 correspondiente.

4.3.1 Descripción de la Interfaz de usuario

En esta plataforma se monitorea el estado de dispositivos IPv6 (máquinas virtuales), por lo cual, se desarrolló una interfaz que obtiene datos de: Espacio de memoria, Procesos activos, Espacio de disco, Estado de puertos y se controla el cierre de procesos. La recolección de datos se realiza mediante la ejecución de comandos consola a la cual se accede mediante SSH.

Los comandos utilizados son:

- Estado de memoria: comando “free”.
- Procesos activos: comando “ps”.
- Espacio de disco: comando “df –h /dev/sda1”.
- Estado de puertos: comando “netstat -tan”.
- Cierre de procesos activos: comando “kill -9”.

En la página principal de la interfaz del servidor externo se puede elegir el dispositivo y el proceso a realizar.

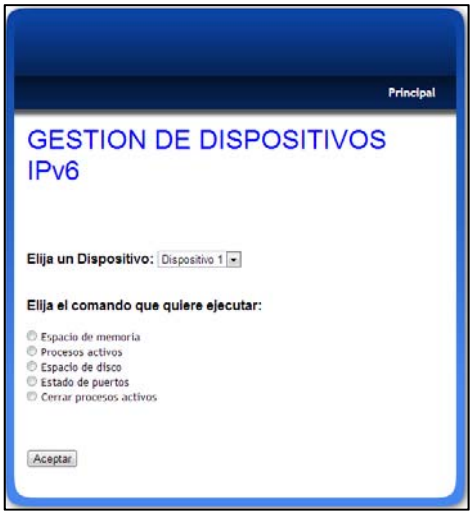

Figura 74. Página principal de la interfaz de usuario del servidor externo.

Fuente: (Elaboración Propia)

En la página principal de la interfaz de los servidores de cada nodo se indica el nodo al que se ha accedido y las opciones a realizar.

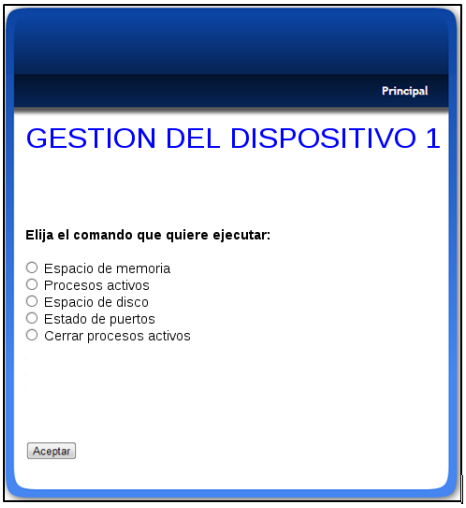

Figura 75. Página principal de la interfaz de usuario del servidor del dispositivo 1.

Fuente: (Elaboración Propia)

Una vez elegidas estas opciones se ejecutará el comando respectivo y se puede visualizar el resultado obtenido.

Figura 76. Resultado de la ejecución del comando "free" del Dispositivo 1.

Fuente: (Elaboración Propia)

Para las opciones de Estado de memoria y Espacio de disco, adicionalmente, en una tercera ventana, se puede visualizar un gráfico de tipo pie (pastel) con los porcentajes de memoria utilizada, libre, compartida y para buffers en el caso de Estado de memoria y los valores de porcentaje utilizado y libre de la partición sda del disco.

Figura 77. Valores de espacio de memoria del Dispositivo 1.

Fuente: (Elaboración Propia)

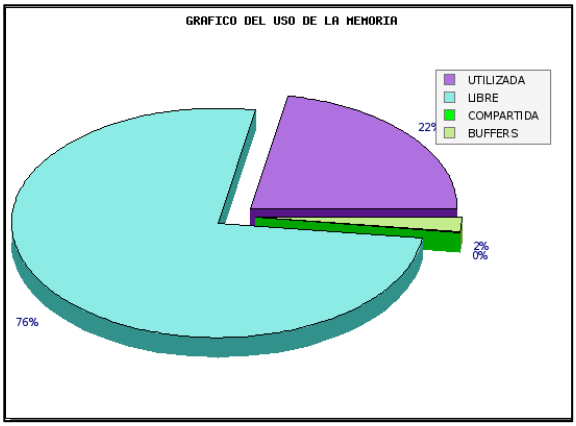

Figura 78. Gráfico del Espacio de Memoria en el Dispositivo 1.

Fuente: (Elaboración Propia)

4.3.2 Programación de la Interfaz de usuario en PHP

La interfaz de usurario desarrollada se aloja en un servidor HTTP apache de código abierto instalado en el sistema operativo Linux. El servidor se encuentra escuchando por el puerto 80 alguna petición proveniente de la red, al recibir

esta petición responde enviando al cliente los archivos que solicita y continúa escuchando.

El desarrollo de la interfaz se lo realiza en el lenguaje de programación PHP, que es un lenguaje de código abierto, adecuado para desarrollo web y que puede ser combinado con HTML. Se ha elegido este lenguaje debido a que puede utilizar la librería libssh2 para el acceso remoto a dispositivos. Libssh2 es una librería utilizada como cliente SSH que implementa el protocolo SSH2. (LibSSH2 Team).

Las principales funciones utilizadas para la conexión SSH desde la interfaz web hacia los dispositivos son:

- **ssh2_connect:** Esta función establece la conexión con un servidor SSH, tiene como argumentos la dirección IP y el puerto correspondiente. En este caso la dirección IP es IPv6 y el puerto es el 22.(The PHP Group, 2001).

- **ssh2_auth_password:** Esta función autentifica sobre SSH utilizando texto plano, tiene como argumentos la conexión obtenida de la función anterior, el nombre de usuario donde se encuentra el servidor y la clave para acceder al mismo.

- **ssh2_exec:** Esta función ejecuta un comando en el servidor y asigna un canal para el mismo. Tiene como argumentos la conexión y el comando a ejecutar.

- **fread:** Esta función lee una cantidad específica de bytes desde un puntero al fichero referenciado por el canal reservado en la función ssh2_exec. Retorna un string con la información obtenida.

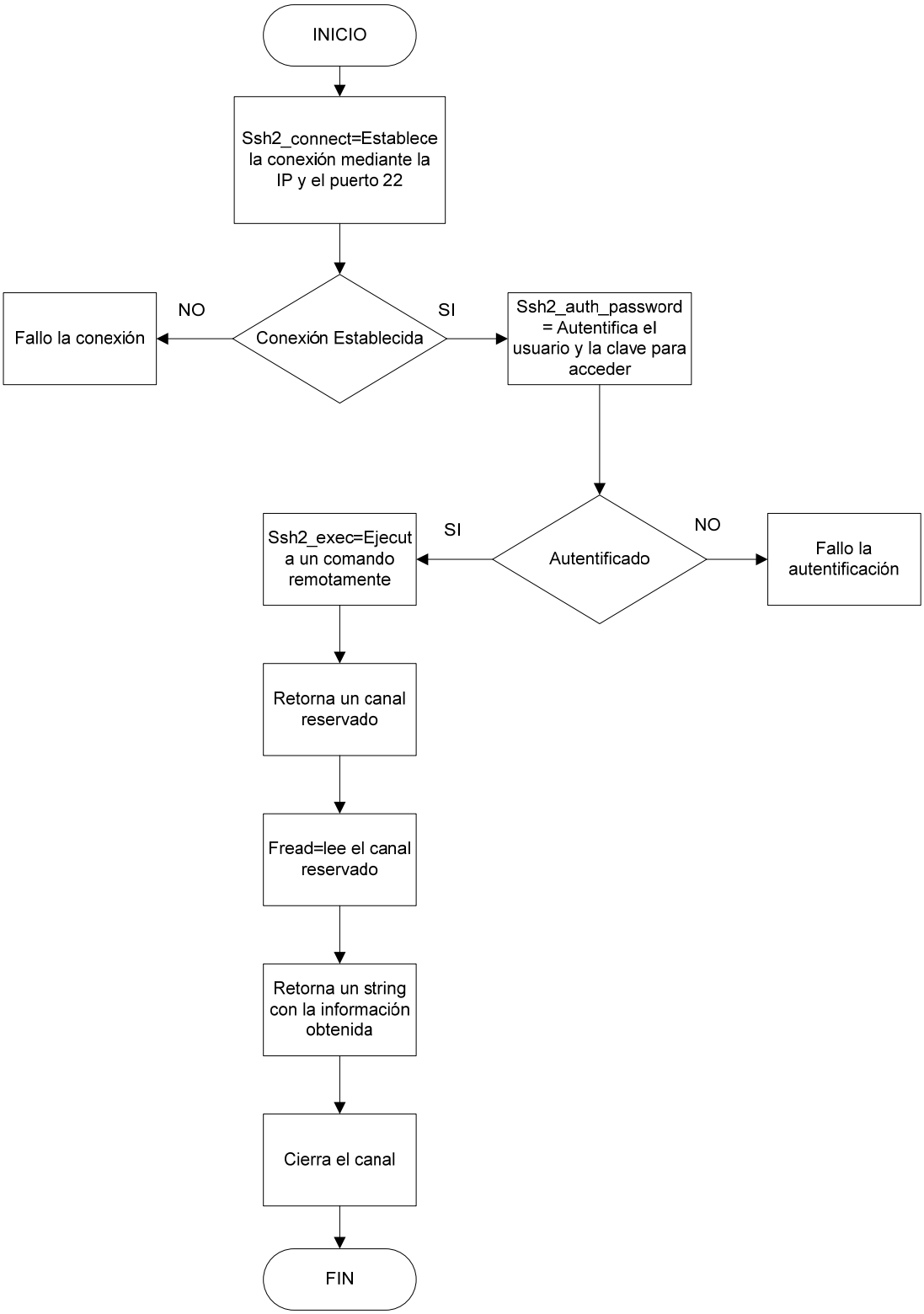

Figura 79. Diagrama de flujo de la conexión remota con SSH.

Fuente: (Elaboración Propia)

Para la realización de los gráficos correspondientes al uso de memoria y espacio de disco se emplea la librería jpgraph para PHP, la misma que sirve para generar imágenes de todo tipo de datos en PHP.

Las clases principales utilizadas para la realización de los gráficos son:(Asial Corporation).

- **PieGraph:** Es una clase que puede generar gráficos tipo pie desde vectores de valores para cada pedazo del pie. Los valores graficados se presentan en porcentajes.

- **PiePlot3D:** Es una clase que permite visualizar el grafico pie en 3D.

4.3.3 Acceso al servidor mediante un túnel IPv6.

El sitio web al que el usuario debe acceder para la gestión de los dispositivos es IPv6, por lo tanto el usuario debe tener en su equipo una dirección IPv6. Una de las formas de obtener esta dirección es mediante un proveedor de servicios de Internet IPv6, como para las instituciones educativas es CEDIA.

La segunda opción es obtener una dirección de un proveedor de tunnelBroker. Los proveedores actuales del servicio de Tunnel Broker se

encuentran en Norte America, Europa y Asia, por lo que los tiempos de retardo de los paquetes hacia estos servidores son altos.

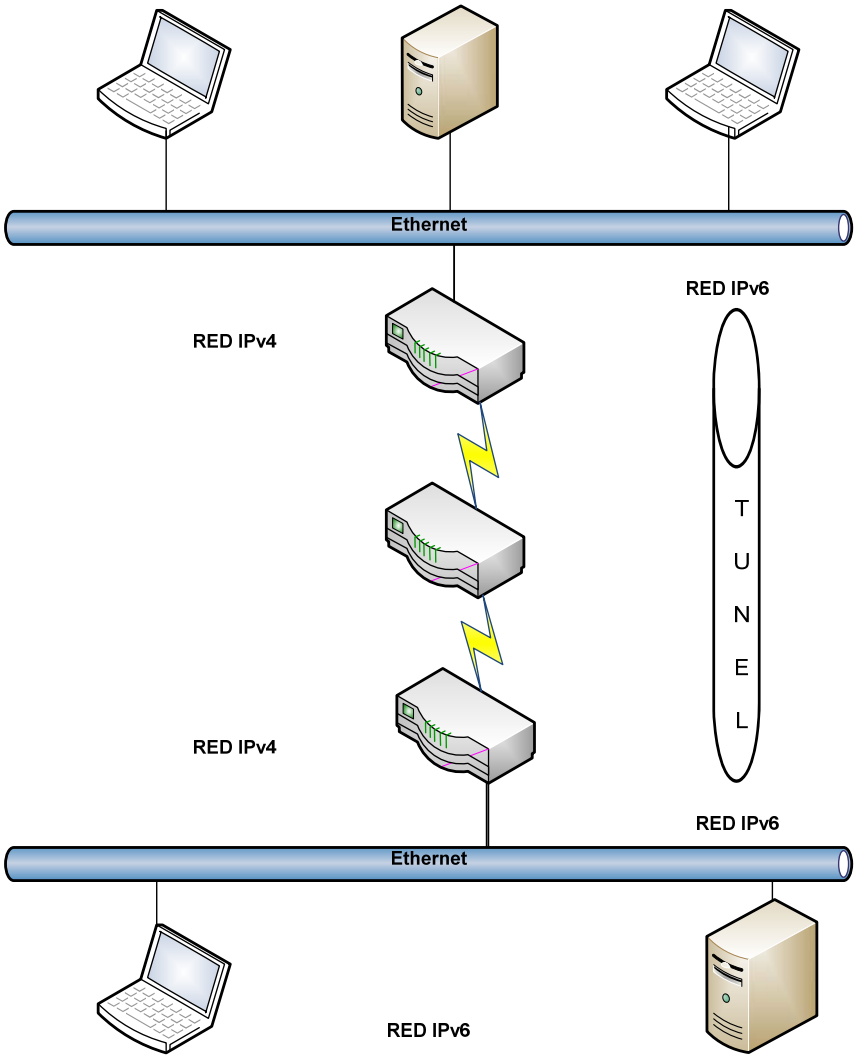

Figura 80. Esquema de un túnel IPv6.

Fuente:(Elaboración Propia)

En este proyecto se utiliza la aplicación gogoCLIENT. Es una aplicación de cliente que establece un túnel con la red gogo6 proporcionándole al usuario una

dirección IPv6 alcanzable desde Internet. GogoCLIENT establece y mantiene el túnel mediante el protocolo propietario TSP.

La aplicación ofrece varios modos de encapsulación de túnel, el utilizado en este proyecto es **v6udpv4** que realiza un encapsulado de IPv6-in-UDP-in-IPv4, la cual está diseñada para trabajar a través de NAT.

GogoCLIENT, se encuentra situado en el equipo local, se conecta con el túnel gogoSERVER y obtiene información relacionada con el protocolo TSP. Tras la recepción de la información para el túnel, el cliente crea un túnel estático en el sistema operativo local.(gogo6 INC., 2002).

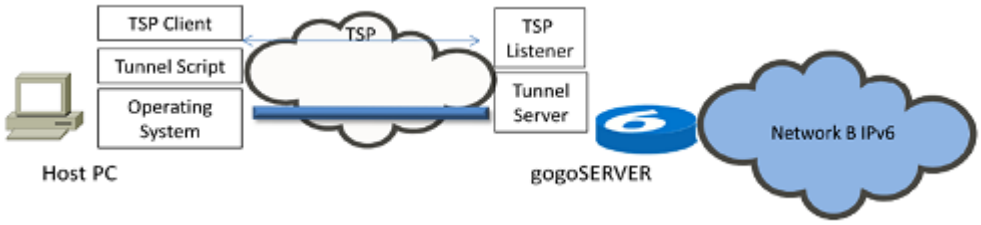

Figura 81. Componentes de gogoClient.

Fuente: (gogo6 INC., 2002).

Para utilizar esta aplicación es necesario registrarse en www.gogo6.com y descargarse la aplicación, hay versiones para Windows y Linux. La instalación es sencilla no requiere configuración.

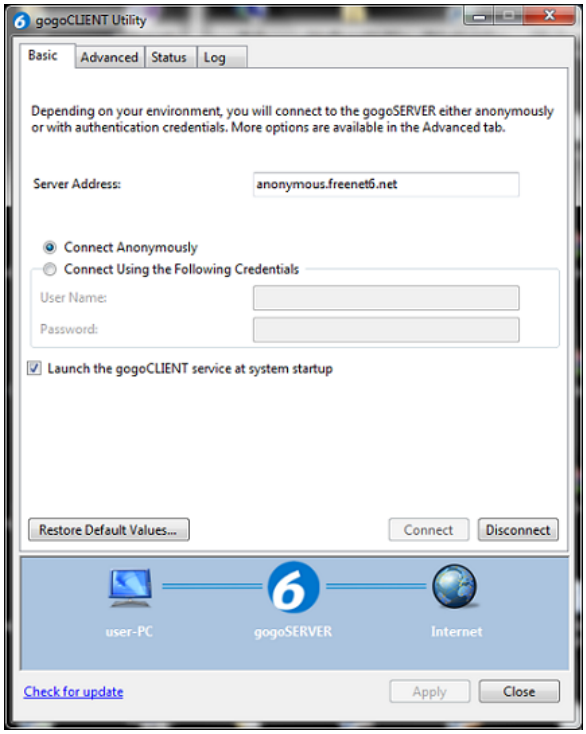

Figura 82. Configuración básica de la aplicación gogoClient.

Fuente: (Elaboración Propia)

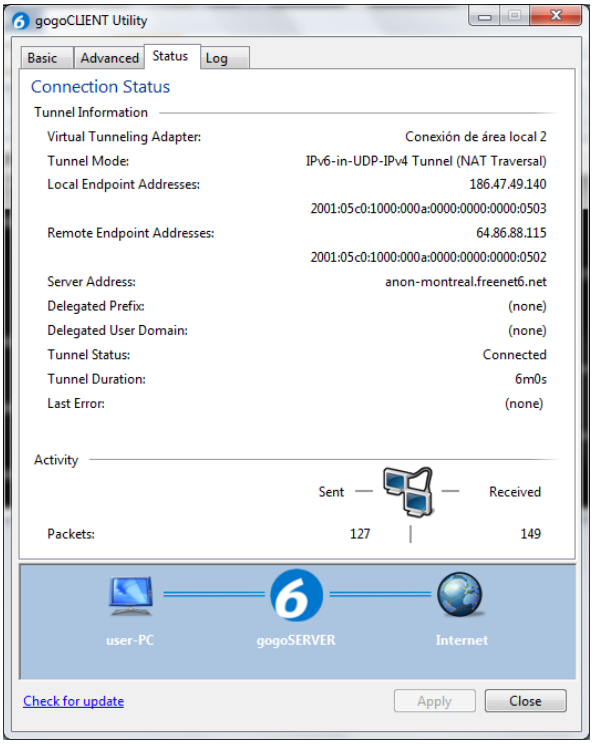

Figura 83. Status de conexión de gogoClient.

Fuente: (Elaboración Propia)

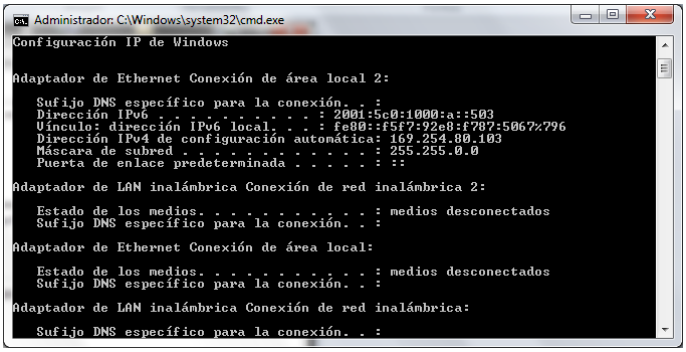

Figura 84. Dirección IPv6 obtenida mediante gogoClient.

Fuente:(Elaboración Propia)

CAPÍTULO 5

PRUEBAS Y ANÁLISIS DE RESULTADOS

5.1. EVALUACIÓN DEL MODELO DE REFERENCIA CON LA PLATAFORMA ARDUINO.

Basado en el modelo de referencia del IoT detallado en el **CAPÍTULO 2**, a continuación se explica cómo la plataforma *Open Hardward*Arduino se acopla al modelo. El modelo de referencia está formado por 4 capas:

5.1.1. Capa de Aplicación y Servicio.

La capa de servicio está basada en una arquitectura *cloudcomputing* en el que consta el software de servicio, la plataforma del servicio y la infraestructura del servicio, por lo que la aplicación se realizó en el dominio de Google App Engine el cual está basado en un lenguaje de programación Python y html, en este se puede visualizar y acceder a los datos ubicuamente desde cualquier dispositivo conectado a internet. Permitiendo visualizar valores de sensores de temperatura, luminosidad, humedad y el estado de dispositivos. La aplicación

permite realizar el procesamiento de los datos obtenidos de los sensores como media y varianza, y el control de encendido y apagado de los dispositivos.

5.1.2. Capa de Red.

La capa de red le permite al dispositivo coordinador acceder a internet asignándole una dirección IPv4. El coordinador de la red recibe los datos y estados que provienen desde los dispositivos de forma inalámbrica yenvía esta información mediante Ethernet hacia el servidor en la nube de Google App Engine.

5.1.3. Capa del Dispositivo.

En la capa del dispositivo se encuentran sensores requeridos por el IoT los cuales sirven para la detección y recolección de datos de temperatura, luminosidad y humedad. Los sensores interactúan de forma inalámbrica con su coordinador en la red Zigbee®, para la transmisión y recepción de todos los datos obtenidos por los mismos.

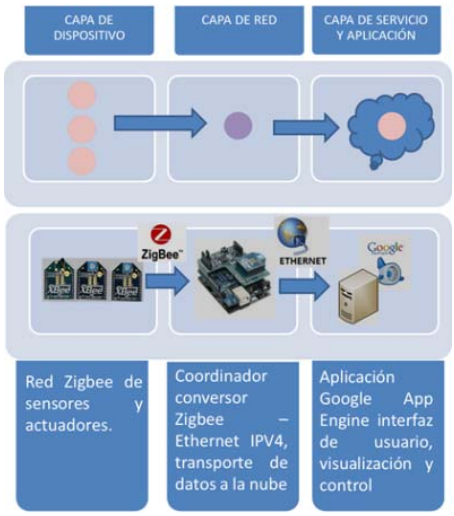

Figura 85. Prototipo basado en la plataforma Arduino dentro del modelo de comunicación del IoT.

Fuente: (Elaboración Propia)

5.1.4. Bloques de construcción del IoT basado en el prototipo Arduino.

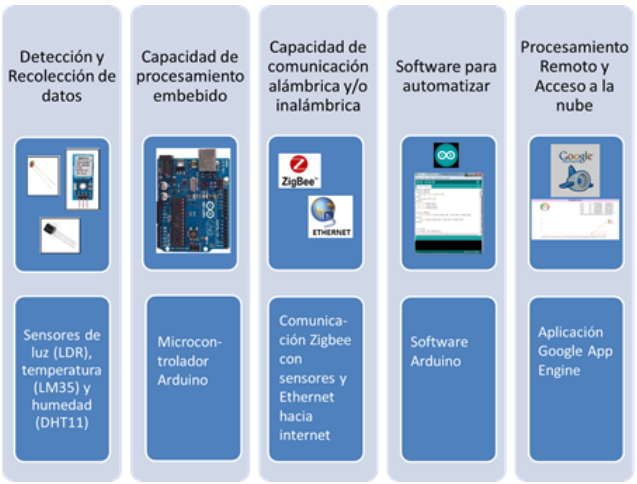

Figura 86. Bloques del IoT.

Fuente: (Elaboración Propia)

5.2. EVALUACÓN DEL MODELO DE REFERENCIA CON LA PLATAFORMA DIGI.

5.2.1. Capa de Aplicación.

La aplicación qué se realizó en la plataforma Digi, al igual que la plataforma Arduino, fue desarrolla en el dominio de Google App Engine en la cual se puede visualizar y acceder a los datos ubicuamente desde cualquier dispositivo conectado a internet. Visualizando valores de sensores de temperatura, luminosidad y el estado de dispositivos obtenidos desde la nube Etherios mediante XML y HTTP. La aplicación permite el monitoreo y control de encendido y apagado de los dispositivos.

5.2.3. Capa de Servicio.

La capa de servicio está basada en una arquitectura *cloudcomputing*. La plataforma comercial Digi tiene su propio entorno de desarrollo en la nube llamado Etherios que es una plataforma de servicio (PaaS), en el cual se puede visualizar todos los dispositivos conectados a la red zigbee® tanto coordinador, routers y dispositivos finales. En esta interfaz se puede realizar la manipulación de los datos permitiendo un procesamiento de los mismos, la visualización de los datos y el control de los dispositivos mediante comandos XML.

5.2.3. Capa de Red.

El coordinador de la red Zigbee® llamado ConnectPort X4 basada en la plataforma comercial Digi recibe los datos y estados que provienen desde los sensores de forma inalámbrica y envía los datos mediante Ethernet hacia el entorno de desarrollo Etherios. El software que permite a los sensores interactuar con su coordinador se llama *DeviceIntegrationApplication* (DIA) basado en el lenguaje de programación de Python.

5.2.4. Capa de Dispositivo.

En la capa de dispositivo se encuentran sensores requeridos por el IoT los cuales sirven para la detección y recolección de datos. La tecnología inalámbrica utilizada en esta capa es Zigbee®.

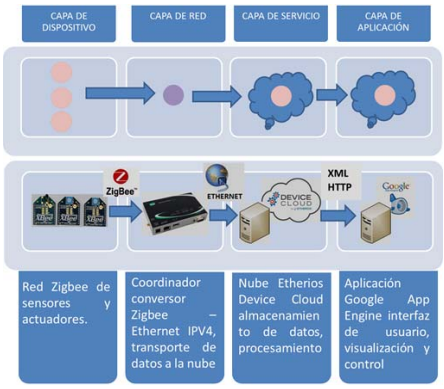

Figura 87. Prototipo basado en la plataforma Digi dentro del modelo de comunicación del IoT

Fuente: (Elaboración Propia).

5.2.5. Bloques de construcción del IoT basado en el prototipo Digi.

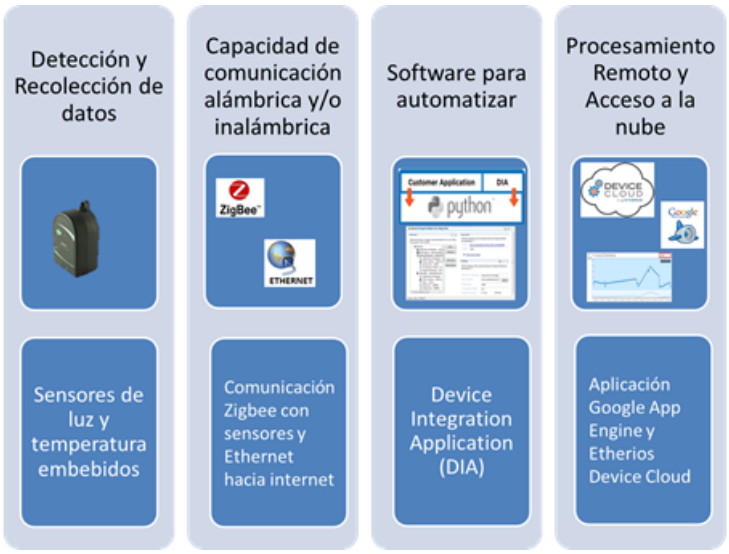

Figura 88. Bloques del IoT.

Fuente: (Elaboración Propia)

Debido a que los proyectos realizados tanto en Arduino como en Digi pueden generarse sin la necesidad de conectar a un ordenador, tomando información del entorno de toda una gama de sensores en tiempo y espacio, afectando el ambiente que lo rodea, se puede decir que los prototipos desarrollados en las plataformas open hardware Arduino y comercial Digi se acoplan al modelo de referencia del IoT.

5.3. EVALUACÓN DEL MODELO DE REFERENCIA CON LA PLATAFORMA IPv6.

5.3.1. Capa de Aplicación y Servicio.

La aplicación se realizó, en un servidor web Apache que permite visualizar y acceder a los datos ubicuamente desde cualquier dispositivo conectado a internet IPv6, mediante SSH. Visualizando el estado de los dispositivos y controlando sus procesos. Debido a que no se logró obtener sensores con IPv6, la aplicación se realizó con el esquema tradicional de servicios REST sobre HTTP/TCP. Sin embargo el nuevo esquema de IoT sobre IPv6 propone la aplicación del nuevo estándar CoAP/UDP el cual es un protocolo de transferencia web, que conserva el modelo REST pero utiliza UDP como capa de transporte. Especializado para el uso con nodos compactos en redes como 6LowPAN.

5.3.1. Capa de Red y Dispositivo.

En la capa de dispositivo se tienen máquinas virtuales con sistema operativo Linux que representan a sensores con soporte IPv6, es el protocolo de red utilizado en esta plataforma, por lo tanto la capa de dispositivo corresponde a una red Ethernet y la capa de red es IPv6. Estos dispositivos se conectan directamente con internet IPv6 sin la necesidad de pasarelas

propietarias ni concentradores y ofrecen servicios básicos como el servicio web. En el nuevo esquema IoT IPv6 los nodos finales corresponden a dispositivos inteligentes que forman redes IEEE 802.15.4 y en la capa de red tienen el estándar 6LowPAN que posibilita el uso de IPv6 sobre estas redes.

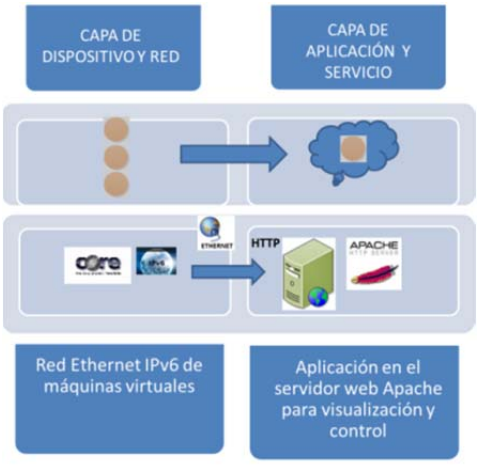

Figura 89. Prototipo basado en la plataforma IPv6 dentro del modelo de comunicación del IoT.

Fuente:(Elaboración Propia)

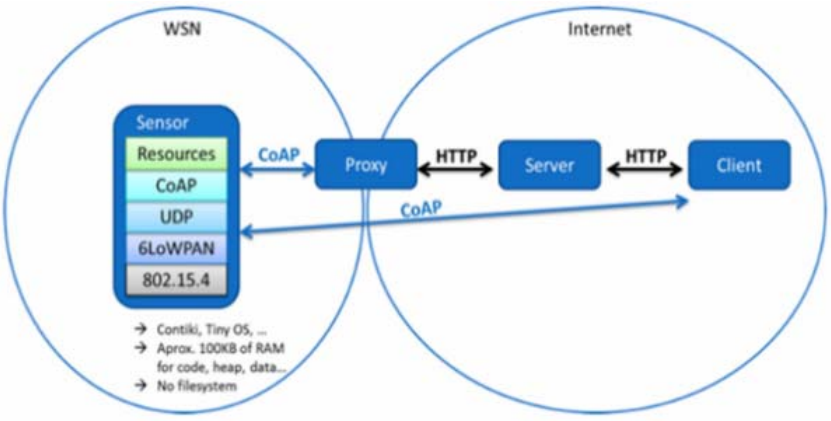

Figura 90. Nuevo esquema de IoT IPv6.

Fuente: (Telefónica, 2013).

5.4. PRUEBA DE LOS PARÁMETROS DE DESEMPEÑO

5.4.1. *Receive Signal Strength Indication* (RSSI).

Este parámetro indica el nivel de potencia de las señales recibidas en redes inalámbricas. El RSSI indica la intensidad de la señal más no la calidad. Su rango se encuentra entre:

- 0 dBm – Señal ideal.
- -40 dBm a -60 dBm Enlace idóneo, tasas de transferencias estables.
- -60 dBm a -70 dBm Enlace bueno.
- -70 dBm a -80 dBm Enlace normal-bajo.
- -80 dBm a -90 dBm Señal mínima aceptable para establecer la conexión.
- -90 dBm a -100 dBm Enlace malo.

La fórmula para encontrar el RSSI es:

$$RSSI = -(10n \log_{10} d + A)$$

Ecuación 6. Fórmula para encontrar el RSSI.

Donde:

n: es igual a la constante de propagación de la señal.

d: distancia entre los nodos.

A: la potencia de la señal recibida a la distancia de 1m.

Se realizaron pruebas mediante el software X-CTU, midiendo la potencia de la señal recibida y el rango del alcance de la señal, verificando cuantos paquetes llegan buenos y cuantos no.

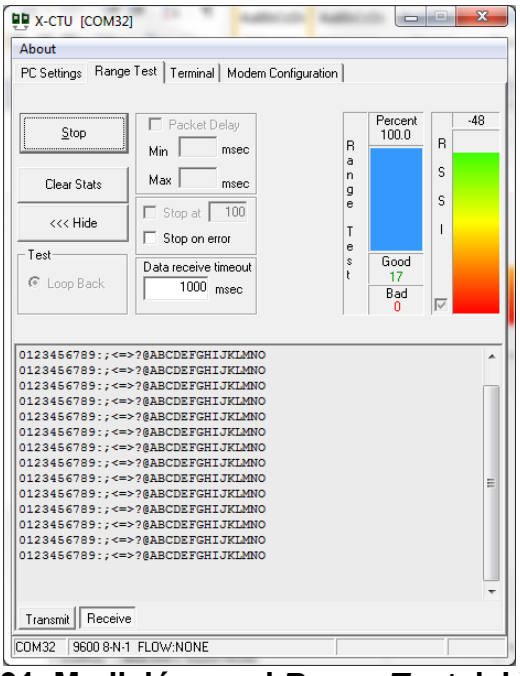

Figura 91. Medición en el *Range Test* del X-CTU.

Fuente: (Elaboración Propia)

Se realizaron pruebas con dos módulos XBee, tanto *indoor* como *outdoor* enviando paquetes de 32 *bytes*.

Pruebas *Indoor*:

Tabla 8. Medición del RSSI en un ambiente *indoor*.

Distancia [m]	RSSI [dBm]	Paquetes Perdidos
0.5	-48	0
2	-51	0
2	-59	0
4	-60	0
6	-62	0
6	-56	0
9	-69	0
9	-70	0
12	-85	2
12	-84	2

Fuente: (Elaboración Propia)

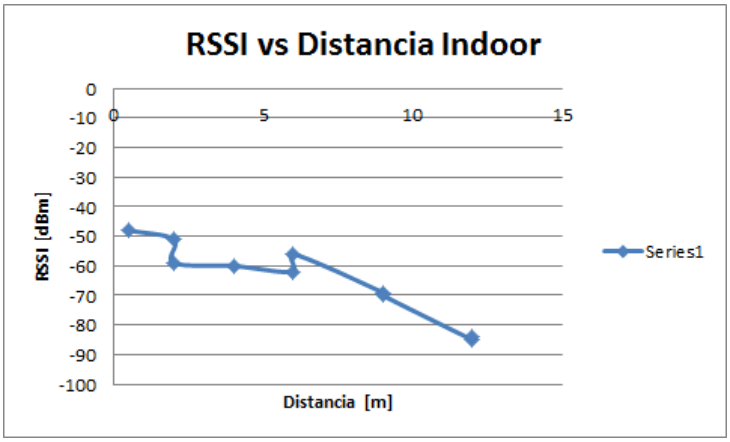

Figura 92. Nivel de potencia de la señal recibida vsla distancia.

Fuente: (Elaboración Propia)

Como se puede observar, a mayor distancia entre nodos, la potencia recibida en los módulos XBee es menor e incrementan el número de paquetes

perdidos. De acuerdo a las especificaciones técnicas de los módulos, tienen un alcance en interiores de 40m. En la práctica de este proyecto se determina que el alcance es mucho menor en interiores y sin línea de vista. La distancia máxima alcanzada, aproximadamente es de 12m.

Pruebas *Outdoor*:

Tabla 9. Medición del RSSI en un ambiente *outdoor*.

Distancia [m]	RSSI [dBm]	Paquetes perdidos
10	-65	0
10	-58	0
10	-68	0
30	-77	0
30	-82	0
30	-84	0
50	-80	7
50	-76	0
50	-71	0
80	-83	0
80	-91	1
80	-87	4

Fuente: (Elaboración Propia)

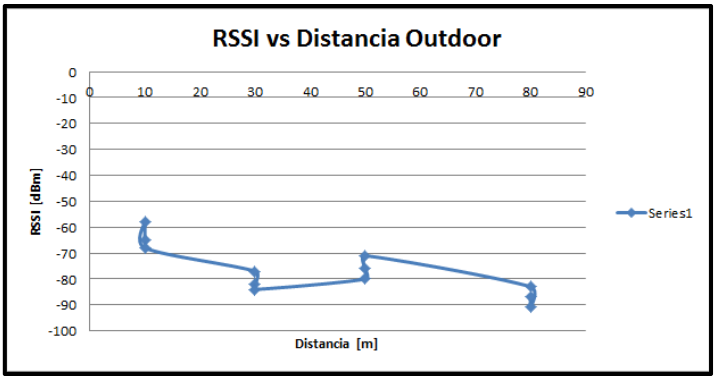

Figura 93. Nivel de potencia de la señal recibida vsla distancia en un ambiente *outdoor*.

Fuente: (Elaboración Propia)

5.4.2. Tiempo de encendido y apagado para el control de los dispositivos en el prototipo Arduino.

Control de Dispositivos:

Tabla 10. Medición del tiempo de respuesta del dispositivo a la menor distancia de su coordinador.

Distancia [m]	Encendido [seg]	Apagado [seg]
1	3.12	3.91
1	3.41	5.27
1	2.59	3.44
1	3.18	3.34
1	4.92	2.77
1	2.32	4.66
1	5.86	5.37
1	2.27	4.37
1	5.64	3.77
1	5.78	2.59
Promedio	**3.909**	**3.949**

Fuente: (Elaboración Propia)

Tabla 11. Medición del tiempo de respuesta del dispositivo 1 para su mayor alcance.

Distancia [m]	Encendido [seg]	Apagado [seg]
12	4.67	5.1
12	2.97	4.16
12	6.18	3.65
12	2.69	3.91
12	5.64	4.79
12	2.36	12.74
12	7.97	3.16
12	3.39	13.17
12	7.62	3.17
12	2.91	6.57
Promedio	**4.64**	**6.042**

Fuente: (Elaboración Propia)

5.4.3. Tiempo de encendido y apagado para el control de los dispositivos en el prototipo Digi.

Control de Dispositivo:

Tabla 12. Medición del tiempo de respuesta del dispositivo a la menor distancia de su coordinador.

Distancia [m]	Encendido [seg]	Apagado [seg]
1	0.82	0.62
1	0.7	0.75
1	0.59	0.68
1	0.66	0.42
1	0.61	0.64
1	0.71	0.66
1	0.61	0.53
1	0.65	0.57
1	0.57	0.56
1	0.56	0.53
Promedio	**0.648**	**0.596**

Fuente: (Elaboración Propia)

Tabla 13. Medición del tiempo de respuesta del dispositivo a la mayor distancia de su coordinador.

Distancia [m]	Encendido [seg]	Apagado [seg]
20	4.62	3.50
20	4.21	3.47
20	4.12	3.20
20	4.83	4.01
20	3.95	3.11
20	5.41	6.25
20	4.33	4.72
20	4.73	4.15
20	3.78	3.19
20	4.59	3.62
Promedio	**4.45**	**3.92**

Fuente: (Elaboración Propia)

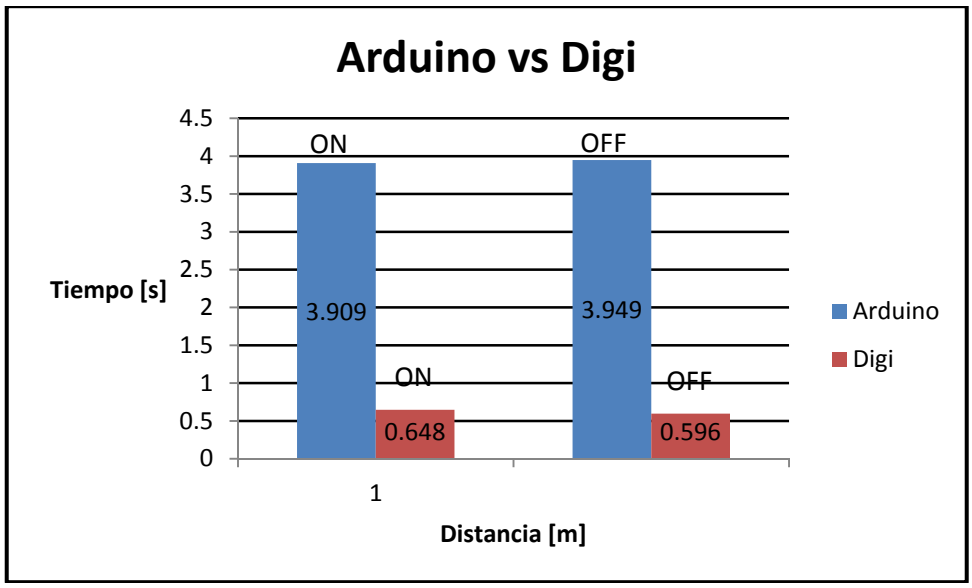

Figura 94. Comparación del promedio del tiempo de respuesta de los dispositivos a la menor distancia de su coordinador.

Fuente: (Elaboración Propia)

5.4.4. Control de dispositivos en productos comerciales.

Figura 95. InterruptorBelkin.

Fuente:(Belkin International, Inc., 2013).

Los interruptores Belkin permiten realizar el control de encendido y apagado de dispositivos en un ambiente wireless, trabaja con cualquier red wi-

fi, su eficacia depende de la velocidad de transmisión de la red. Para manipular los dispositivos es necesario descargar la aplicación llamada WEMO que se puede encontrar tanto para el sistema operativo Android como para IOS.

Para poder visualizar la velocidad de la red se recurrió a la dirección www.speedtest.net, la cual permite realizar un test en la que indica la velocidad de descarga y carga.

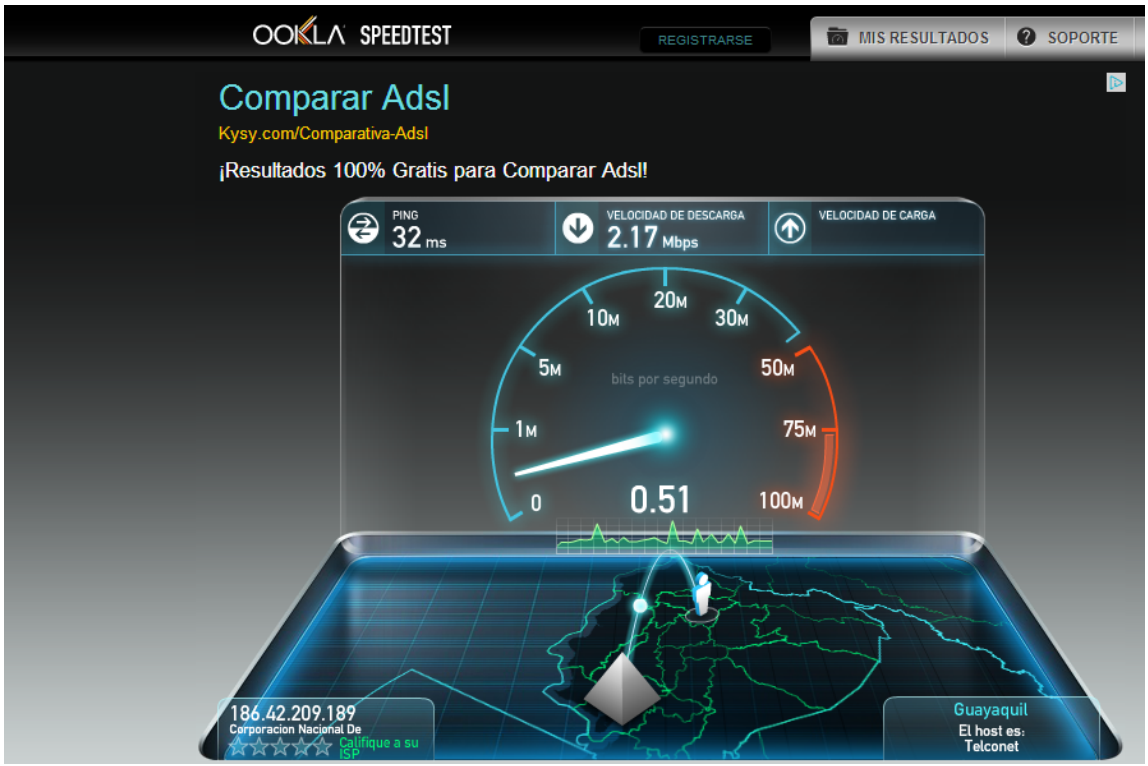

Figura 96.Velocidad de transmisión de la red.

Fuente: (OOKLA SPEEDTEST).

Se realizaron pruebas con una velocidad de:

Figura 97. Valores de carga y descarga de la red.

Fuente:(OOKLA SPEEDTEST).

Prueba Local: Tanto el interruptor como el dispositivo que lo controla se encuentran en la misma red.

Tabla 14. Medición del tiempo de respuesta del encendido y apagado del dispositivo desde una red local.

Red Local	Encendido [seg]	Apagado [seg]
	0.76	1.04
	0.55	0.66
	0.67	0.69
	0.57	0.52
	0.5	0.59
	0.62	0.93
	0.58	0.5
	1.01	0.65
	0.77	0.52
	0.58	0.76
Promedio	**0.661**	**0.686**

Fuente: (Elaboración Propia)

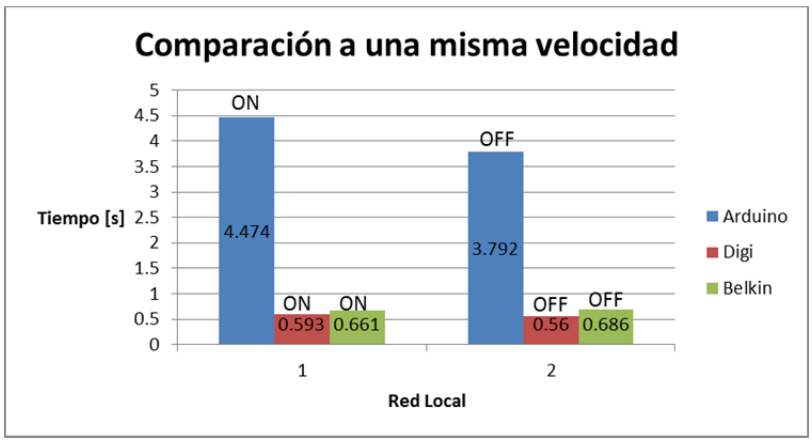

Figura 98. Comparación del tiempo de respuesta entre los prototipos Arduino, Digi y los dispositivos Belkin, en la misma red local.

Fuente: (Elaboración Propia)

Se puede observar que el prototipo Arduino presenta el mayor retardo para el control de dispositivos, debido a que la interacción con los nodos de control es mediante lectura de tramas, su arquitectura fue ensamblada manualmente y la programación es más compleja que los dispositivos comerciales. La eficiencia depende de la velocidad de transmisión de la red para todos los prototipos.

Prueba Externa: El dispositivo que controla al interruptor se encuentra en una red diferente.

Tabla 15. Medición del tiempo de respuesta del encendido y apagado del dispositivo desde red externa.

Red Externa	Encendido [seg]	Apagado [seg]
	12.48	2.7
	2.55	5.84
	3.06	3.22
	2.86	3.93
	7.03	3.03
	6.87	3.17
	6.63	3.66
	9.17	5.02
	2.84	2.26
	2.65	2.53
Promedio	**5.614**	**3.536**

Fuente: (Elaboración Propia)

5.5. COMPARACIÓN ENTRE LOS PROTOTIPOS BASADOS EN DIGI, ARDUINO Y CONECTIVIDAD IPv6.

5.5.1. Comparación de las Características del Modelo de Referencia entre los prototipos.

Tabla 16. Comparación del diseño entre los prototipos.

PARAMETROS	PROTOTIPO DIGI	PROTOTIPO ARDUINO	PROTOTIPO IPv6
Dispositivos Sensores	Sensores Embebidos – mayor precisión. (Wall Router).	Sensores Modulares – menor precisión, requieren circuitos de acondicionamiento. (LDR, LM35, DHT11).	Se utiliza máquinas virtuales como dispositivos finales, los cuales al tener un sistema operativo son más inteligentes y ofrecen mejores características.
Dispositivos Actuadores	Requieren circuito de acondicionamiento (Xbee DIO).	Requieren circuito de acondicionamiento (Xbee DIO).	
Módulos de conectividad inalámbrica.	Módulos Zigbee®. (Xbee series 2).	Módulos Zigbee®. (Xbee series 2).	El módulo de comunicación ya viene integrado en el sensor o actuador y este puede ser alámbrico o inalámbrico.

PARAMETROS	PROTOTIPO DIGI	PROTOTIPO ARDUINO	PROTOTIPO IPv6
Gateway.	Gateway comercial compacto, realiza la conversión de zigbee® a Ethernet. Configuración vía web. (ConnectPort X4).	Gateway modular, requiere microcontrolador, adaptador Ethernet y adaptador zigbee®. Para la configuración requiere del software X-CTU. (Arduino UNO, Arduino Ethernet Shield, ArduinoXbeeShield y Modulo Xbee).	No se requiere de un dispositivo intermedio debido a que los sensores se conectan directamente a la red.
Software de Automatiza-ción.	DeviceIntegration Application (DIA). Permite interactuar de forma sencilla con los sensores. Envía los datos directamente hacia la nube de digi.	Utiliza software propio de Arduino basado en el lenguaje de programación C. Requiere código para lectura de sensores, envío de datos hacia una aplicación de cliente, recepción de datos y control de dispositivos.	Al poseer el sensor un sistema operativo puede realizar el procesamiento local sin necesidad de un software adicional, e incluso puede ofrecer más servicios.
Aplicación y nube de dispositivos.	Digi posee su propia nube de dispositivos ETHERIOS. La aplicación de cliente en Google App Engine que accede a los datos mediante XML y HTTP.	La aplicación de cliente en Google App Engine recibe los datos directamente del gateway.	La aplicación realizada en el prototipo IPv6 se encuentra en un servidor Apache externo y local, para interactuar directamente con los dispositivos finales.

Fuente: (Elaboración Propia)

En función de la tabla 5.10 se puede resumir con respecto al modelo de referencia de IoT de los tres prototipos implementados:

- En los prototipos IoT tradicionales no se puede acceder directamente a los dispositivos finales, sino solo a través de su coordinador, mientras que en el prototipo basado en ipv6 el acceso al dispositivo es directo y mediante su dirección IP, que es su identificador dentro de la red y es único.

- Los dispositivos en el prototipo IPv6 vienen integrados con todas las capas del modelo IoT, incluyendo sensores embebidos, módulo de comunicación, y servicios de usuario. Es decir que son dispositivos inteligentes con mayores capacidades.

- En el prototipo IPv6 no se requiere de un dispositivo intermedio (Gateway) que concentre la información de los dispositivos finales, debido a que los sensores se encuentran conectados directamente a la red de Internet IPv6.

- En el prototipo basado en Digi, al poseer una nube de dispositivos propia, llamada Etherios, reduce la complejidad de la obtención y visualización de los datos de la red de dispositivos y permite una mejor administración de los nodos. El retardo en el acceso a los datos es menor que en una nube pública como la que se utiliza en el prototipo basado en la plataforma Arduino.

- El prototipo basado en IPv6 permite una conectividad end-to-end, a diferencia de los prototipos tradicionales (Digi, Arduino) que utilizan el protocolo IPv4, ya que elimina la barrera de NAT, con las direcciones IP específicas y permanentes en los dispositivos. Añadiendo así la capacidad de movilidad, permitiendo cambiar de puntos de acceso a internet sin perder conexión.

- Una de la más importante diferencias entre las redes tradicionales e IPv6 es la gran dimensión de las tablas de encaminamiento en la red troncal de Internet que utiliza IPv4, que la hace ineficaz y perjudica considerablemente los tiempos de respuesta. En IPv6 el encaminamiento en la red troncal es más eficiente, debido a una jerarquía de direccionamiento basada en la agregación y a que la fragmentación y desfragmentación de los paquetes se realiza extremo a extremo y no en los nodos intermedios la cual mejora su tiempo de respuesta. (Millán, 2006).

- En el acceso a la interfaz web mediante un túnel IPv6 sobre IPv4, el proceso de transmisión de los paquetes inicia en un extremo del túnel, luego son encapsulados y enviados hasta el enrutador del proveedor de túnel, el cual lo desencapsula y por último se inicia el enrutamiento hacia el destino final en la nube de IPv6. Este proceso incrementa el retardo en un 50% de una red nativa.(Ministerio de Ciencia y Tecnología de ESPAÑA, 2004). Por lo tanto una red IPv6 nativa mejora la velocidad de transmisión.

5.5.2. Análisis y Evaluación de los Prototipos para el desarrollo del IoT.

Los aspectos a ser evaluados tendrán una valoración de 5 para el prototipo que mejor cumpla con cada parámetro y de 0 como mínima valoración.

- **Disponibilidad de Información:** Es la base para el desarrollo de los prototipos, tiene que ver con información de configuración, soporte de dispositivos, formas de implementación, características y requerimientos técnicos.

- **Disponibilidad de equipos:** Se refiere a la cantidad de equipos que se pueden encontrar en el mercado.

- **Costos de implementación:** Este aspecto hace referencia al costo de los dispositivos para cada prototipo.

- **Facilidad de implementación:** Se refiere al diseño de cada prototipo, el nivel de complejidad en la construcción de la red así como el desarrollo de las interfaces de usuario.

- **Administración:** Para la evaluación de este aspecto se tomará en cuenta el nivel de accesibilidad a los dispositivos finales.

- **Escalabilidad:** Se evalúa el nivel de escalabilidad horizontal en cada prototipo.

- **Movilidad:** En este aspecto se evalúa la capacidad de los dispositivos de cambiar de puntos de acceso y el nivel de complejidad en realizarlo.

Tabla 17. Valoración de los Prototipos.

PARÁMETRO DE EVALUACIÓN	PROTOTIPO ARDUINO	PROTOTIPO DIGI	PROTOTIPO IPv6
Disponibilidad de Información	5	3	1.5
Disponibilidad de equipos	5	5	1
Costos de implementación	4	2.5	2
Facilidad de implementación	2	5	5
Administración	1	3.5	5
Velocidad de la red	1	4	5
Escalabilidad	3	3.5	5
Movilidad	2	2	5
PROMEDIO	**2.875**	**3.5625**	**3.6875**

Fuente: (Elaboración Propia)

5.5.3. Análisis de Costos.

En este análisis se presenta el costo de los materiales de implementación de los diferentes prototipos realizados dentro del modelo de referencia del IoT. Posteriormente se realiza una comparación entre los costos obtenidos permitiendo observar cuál de las implementaciones es más económica y determinar la importancia de este factor para el desarrollo de proyectos basados en IoT.

Tabla 18. Costos de implementación del prototipo basado en Arduino.

CANTIDAD	DISPOSITIVO	PRECIO	TOTAL
	Red Zigbee®		
1	Arduino UNO REV 3	$25.79	$25.79
1	Ethernet W5100	$23.93	$23.93
1	ArduinoXbeeShield	$27.68	$27.68
2	Módulos Xbee Series 2	$17.00	$34.00
1	Interface Board - XBee, RS232, development	$70.00	$70.00
2	XBee-PRO DigiMesh 2.4, Digital IO adapter	$129.00	$258.00
	Circuitos de Acondicionamiento Sensores y Actuadores		
1	Sensor DHT 11	$15.00	$15.00
1	Sensor LM35	$1.79	$1.79
2	LDR	$1.00	$2.00
6	Resistencias ½ W	$0.02	$0.12
2	L7805	$0.35	$0.70
4	Capacitores	$0.10	$0.40
2	Transistor 2N3904	$0.06	$0.12
2	Diodo 1N4007	$0.08	$0.16
2	Relé 5V	$0.71	$1.42
2	LED	$0.06	$0.12
2	Broche de Batería	$0.20	$0.40
10	Borneras	$0.10	$1.00
2	Baquelita	$1.00	$2.00
2	Baterías 9V	$3.00	$6.00
4	Cajas	$5.16	$20.65
2	Cable Extensión 110V	$1.95	$3.90
	TOTAL		**$495.18**

Fuente: (Elaboración Propia)

Tabla 19. Costos de implementación del prototipo basado en Digi.

CANTIDAD	DISPOSITIVO	PRECIO	TOTAL
	Red Zigbee®		
1	ConnectPort X4	$384.00	$384.00
1	XBee-PRO ZB Wall Router	$69.00	$69.00
2	XBee Digital IO adapter	$129.00	$258.00
	Circuitos de acondicionamiento actuadores		
2	Relé 5V	$0.71	$1.42
2	78LS05	$0.35	$0.70
4	Capacitores	$0.10	$0.40
2	Transistor 2N3904	$0.06	$0.12
2	Diodo 1N4007	$0.08	$0.16
4	Resistencias 1/2W	$0.02	$0.08
2	Led	$0.06	$0.12
2	Cable Extensión 110V	$1.95	$3.90
2	Baterías 9V	$3.00	$6.00
2	Broches de bacteria	$0.15	$0.30
4	Borneras	$0.10	$0.40
2	Baquelita	$1.00	$2.00
2	Cajas	$3.55	$7.10
		TOTAL	733,7

Fuente: (Elaboración Propia)

Debido a que no se logró obtener sensores IPv6 para el desarrollo de la aplicación no se presenta tabla de costos para el prototipo basado en la plataforma IPv6.

5.5.4. Comparativo de CostosArduino vs. Digi.

Se presenta un análisis comparativo de costos entre una red basada en la plataforma Arduino y una basada en la plataforma Digi, compuesta por los siguientes elementos:

- 1 Coordinador.
- 1 Nodo sensor de temperatura y luminosidad.
- 2 Nodos actuadores.

Tabla 20. Comparativo de costos de prototipos Arduino vs. Digi.

Elemento	Plataforma Arduino	Plataforma Digi
Coordinador	$ 99,56	$ 384,00
Nodo sensor	$ 96,17	$ 69,00
Nodos actuadores	$284,02	$ 280,80
TOTAL:	479,75	733,8

Fuente: (Elaboración Propia)

Como se puede observar la implementación mediante la plataforma Arduino al ser Open Hardware es más económica, pero su construcción es más compleja al ser manual y carece de soporte técnico de fabricante.

5.5.5. Análisis Comparativo de Escalabilidad e Interoperabilidad.

Escalabilidad

Uno de los factores más importantes de IoT es la escala de la cantidad de dispositivos, o las cosas, que se conectan a Internet. Las redes y tecnologías actuales están diseñadas para cantidades más pequeñas de dispositivos, por lo tanto, la ampliación de la red y la comunicación para una gran cantidad de cosas debe ser considerada en el despliegue de redes IoT.

El sistema de redes IoT se puede escalar de forma horizontal, que es el proceso de agregar más nodos al sistema, o de forma vertical, en la que se añaden mayor cantidad de recursos a un solo nodo.

Tabla 21. Comparación de escalabilidad entre los prototipos.

PARÁMETRO	PROTOTIPO ARDUINO	PROTOTIPO DIGI	PROTOTIPO IPv6
Escalar horizontalmente	Es posible añadir una mayor cantidad de nodos finales al coordinador existente y en el caso de requerirlo, se pueden agregar más dispositivos coordinadores con sus respectivos nodos finales.	Es posible añadir una cantidad de hasta 25 nodos finales al coordinador existente y en el caso de requerirlo, se pueden agregar más dispositivos coordinadores con sus respectivos nodos finales.	Es posible añadir gran cantidad de dispositivos ya que IPv6 tiene millones de direcciones disponibles.
Escalar verticalmente	Debido a que Arduino es de tipo Open Hardware permite agregar más recursos en cuanto a memoria para la adquisición de datos de los nodos finales.	Debido a que los dispositivos de la marca Digi son de tipo comercial, no es posible la modificación del hardware. Por lo tanto no se pueden incrementar recursos.	En el caso de este proyecto es posible aumentar más recursos debido a que los dispositivos finales son máquinas virtuales. Sin embargo al tener dispositivos embebidos en el nuevo esquema no es posible.
Escalabilidad de la nube.	Al incrementar el número de nodos que se deben monitorizar o controlar desde la nube, se requiere de una mayor cantidad de almacenamiento en la nube, lo que implica un costo adicional. Además, la complejidad de la programación de la interfaz aumenta.	La nube de Etherios permite la conexión de hasta 5 dispositivos de forma gratuita, en el caso de incrementar este número, se debe cancelar un valor. No se requiere ninguna programación para el manejo de los dispositivos.	A diferencia de los anteriores prototipos, el prototipo basado en la plataforma IPv6 es escalable en la nube ya que cada dispositivo tiene su propia dirección IPv6 y puede conectarse directamente con la nube sin la necesidad de dispositivos intermedios.
Escalabilidad en la conexión a Internet	El acceso a internet en estos prototipos se lo realiza mediante el protocolo IPv4 haciendo uso de la traducción de direcciones NAT. Debido a la escasez de direcciones IPv4, estos prototipos no son escalables.		Este prototipo es escalable ya que gran cantidad de dispositivos pueden conectarse a internet por la disponibilidad de direcciones IPv6.

Fuente: (Elaboración Propia)

Interoperabilidad

Es la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información.

La interoperabilidad es uno de los principales aspectos del IoT, la cual debe extenderse por toda la pila. El sistema basado en IoT tiene que realizar lo que diga la norma técnica, que es la responsable de que cada capa sea interoperable. La norma efectúa las acciones que sean precisas para comunicarse con otros elementos mediante mensajes e intercambio de datos.

Es importante una interoperabilidad semántica en la que los dispositivos deben tener la capacidad de entender el significado y el contexto de los datos que comunican. La normalización es la mejor manera de alcanzar este nivel de interoperabilidad semántica.

La capa física de los prototipos basados en Arduino y Digi, corresponde a una red inalámbrica estandarizada bajo IEEE 802.15.4, que establece las especificaciones para baja potencia de redes inalámbricas de área personal. Los dispositivos utilizados en el prototipo Arduino están diseñados para interoperar con otros productos de red Digi, incluyendo gateways y adaptadores.

La herramienta Google App Engine utilizada en la capa de aplicación permite el desarrollo de las interfaces en los dos prototipos, por lo tanto existe interoperabilidad entre estos sistemas.

En el prototipo basado en la plataforma IPv6 es posible lograr la interoperabilidad entre IPv4 e IPv6 en cuanto al acceso a la aplicación gracias a la existencia de los túneles que se pueden crear entre estas redes. Los dispositivos finales como sensores embebidos solo pueden ser interoperables entre sí mientras posean soporte IPv6.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES.

- En general se concluye, en base al proyecto, que es posible la aplicación del modelo de referencia del *Internet of things* en la implementación de todas las plataformas propuestas Arduino, Digi e IPv6, variando el nivel de complejidad en su construcción por sus características de ser *Open Hardware* en el caso de Arduino, comercial para el caso de Digi y la carencia de dispositivos IPv6.

- Se puede evidenciar la gran ventaja que proporciona el manejo de datos en la nube (*cloud*), demostrando que no se requiere de infraestructura propia para el almacenamiento de aplicaciones ya que se puede utilizar la estructura necesaria de un proveedor, como en este proyecto se utilizó la plataforma Google para el prototipo basado en Arduino y Etherios, que es una plataforma de servicios propia de Digi.

- La monitorización de sensores y la gestión de dispositivos desde la nube, no es factible en tiempo real ya que los retardos presentes en la adquisición de datos son aleatorios, por lo tanto a pesar que se puede realizar el procesamiento de información en la nube, no es aconsejable para aplicaciones críticas.

- El modelo de referencia del *Internet of things* se basa en cuatro capas, en la capa de dispositivo la plataforma *Wireless Sensor Network* es la base para su desarrollo, en la capa de red se emplea el protocolo IPv4 e IPv6 con el protocolo TCP como transporte, en la capa de servicio, se proveen opciones de almacenamiento y procesamiento de datos, por último en la capa de aplicación se utiliza el esquema HTTP-REST.

- En base a las pruebas realizadas se concluye que el prototipo basado en la plataforma comercial Digi presenta menores retardos en la gestión de dispositivos debido a que el coordinador de la red Zigbee® tiene mayor alcance y potencia de transmisión, posee una interfaz web propia que realiza un mejor tratamiento de los datos comparada con la plataforma Arduino en la que el coordinador es ensamblado de forma manual y su programación es más compleja.

- Se concluye que la diferencia entre las redes tradicionales IoT (Arduino y Digi) y el prototipo implementado basado en IPv6 es que en esta

plataforma no es necesaria la existencia de un nodo coordinador que concentre la información de los nodos finales, debido a que cada nodo se conecta directamente con su interfaz de gestión mediante una dirección IPv6, eliminando el retardo que posee la red Zigbee® y permitiendo una fácil administración de los dispositivos. Además al poseer mayor cantidad de direcciones IP, el prototipo basado en IPv6 es más escalable que los prototipos tradicionales.

- Al ser Digi un producto comercial, la implementación basada en esta plataforma es más costosa que el prototipo basado en Arduino debido a que es Open Hardware y toda su infraestructura se la construye manualmente. En el mercado se encuentran innumerables dispositivos que aplican el modelo IoT como sensores y actuadores WiFi, bombillas Zigbee®-WiFi, entre otras.

6.2. RECOMENDACIONES.

- Los dispositivos para el desarrollo de prototipos funcionan con baterías, con un tiempo de vida corto, por lo tanto es necesario minimizar el consumo de energía con elementos eficientes y optimizando el sistema con periodos de guarda en que los dispositivos no están activos.

- La seguridad, privacidad y la protección de datos son fundamentales para el desarrollo del Internet of Things, debido a la gran cantidad de dispositivos conectados a la nube, para lo cual se requiere implementar sistemas de protección tanto en la infraestructura física como de red.

- Existen varios modelos de referencia para la implementación de arquitecturas dentro del Internet of things, se recomienda la utilización del modelo de referencia propuesto por la ITU, ya que la información que contiene es más específica.

- En la implementación de este proyecto se pudo verificar que el prototipo basado en dispositivos IPv6 es el recomendable para el IoT, debido a sus características de escalabilidad y administración, sin embargo tiene un limitante en la actualidad, que es la disponibilidad de equipos en el mercado.

BIBLIOGRAFÍA

- *ARDUINO*. (07 de Octubre de 2013). Obtenido de http://www.arduino.cl/
- *BRICO GEEK* . (7 de Octubre de 2013). Obtenido de http://www.bricogeek.com/shop/14-arduino-xbee-shield.html
- Appcelarator Inc. (2008). *Titanium*. Obtenido de http://www.appcelerator.com/
- Arduino. (s.f.). *Arduino Ethernet Shield*. Obtenido de http://arduino.cc/es/Guide/ArduinoEthernetShield
- Arduino. (s.f.). *Librería Ethernet library*. Obtenido de http://arduino.cc/es/Reference/Ethernet
- Arduino. (s.f.). *SPI library*. Obtenido de http://arduino.cc/en/Reference/SPI
- Asial Corporation. (s.f.). Obtenido de JpGraph: http://jpgraph.net/download/manuals/classref/PiePlot.html
- Autodesk Inc. (2013). *Arduino Controlled Light Dimmer*. Obtenido de http://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/
- Belkin International, Inc. (2013). *Wemo Switch*. Obtenido de http://www.belkin.com/us/F7C027-Belkin/p/P-F7C027/
- Benítez, C. (30 de Diciembre de 2010). *Tutorial JSON.* Obtenido de http://www.etnassoft.com/2010/12/30/tutorial-json/
- BricoGeek. (2005). *FOTORESISTENCIA LDR*. Obtenido de http://www.bricogeek.com/shop/373-fotoresistencia-ldr.html
- BricoGeek. (2005). *XBEE 2MW SERIE 2 (ZB) CON ANTENA*. Obtenido de: http://www.bricogeek.com/shop/43-xbee-2mw-serie-25-con-antena.html

- Cicileo, G. (s.f.). *Mecanismos de transición*. Obtenido de http://portalipv6.lacnic.net/mecanismos-de-transicion/
- Digi International Inc. (2010). *Google App Engine Device Cloud Client.* Obtenido de: http://www.digi.com/wiki/developer/index.php/Google_App_Engine_iDigi_Client
- Digi International. (15 de Marzo de 2012). *XBee Wall Router*. Obtenido de http://www.digi.com/wiki/developer/index.php/XBee_Wall_Router
- DigiInternational,Inc. (2007). *XBee™ Series 2 OEM RF Modules*. Obtenido de ftp://ftp1.digi.com/support/documentation/90000866_A.pdf
- Doukas, C. (2012). An Intruction to Cloud Computing . En C. Doukas, *Building Internet of Things with the Arduino.*
- Elliott, R. (18 de Diciembre de 2002). *Voltage Dividers & Attenuators* . Obtenido de http://sound.westhost.com/vda.htm
- Etherios. (s.f.). *Device Cloud.* Obtenido de http://www.etherios.com/pdf/ds_devicecloud.pdf
- Frikadas con Arduino y Raspberry Pi. (2009). *Arduino + LM35*. Obtenido de http://www.sherkhan.net/blogs/frikadas/?p=397
- Furness, A. (s.f.). CASAGRAS.
- GitHub, Inc. (29 de Diciembre de 2012). *DHT-sensor-library*.
- gogo6 INC. (2002). *CLIENT GUIDE .* Obtenido de http://content.gogo6.com/gogo_dc_0010_gogoclient_guide.pdf
- Google App Engine. (s.f.). *Create an Application*. Obtenido de https://appengine.google.com/start/createapp
- Google Developers. (20 de Junio de 2013). *Google App Engine Tutorial*. Recuperado el 12 de Agosto de 2013, de https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=es
- Google Developers. (s.f.). *Go Application Configuration with app.yaml*. Obtenido de https://developers.google.com/appengine/docs/go/config/appconfig

- Google Developers. (s.f.). *Using Google Charts.* Obtenido de https://developers.google.com/chart/interactive/docs/index?hl=es
- Haller, S. (s.f.). SAP AG.
- Home Automation. (3 de Mayo de 2012). *Home-Automation*. Obtenido de http://home-automation-upenn.blogspot.com/2012/04/using-ethernet-shield-and-xbee-shield.html
- Inc., Digi International. (1996). *XBee USB Adapter ZNet 2.5.* Obtenido de http://www.digi.com/support/productdetail?pid=3140&type=documentation
- Ingeniería MCI Ltda. (2009). *Familias XBee Serie 1, XBee Serie 2 y XBee 900*. Obtenido de http://www.xbee.cl/diferencias.html
- International Communication Union ITU-T. (2012). Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Netoworks. *Telecommunication Standarization Sector of ITU*, 22.
- International., D. (3 de Mayo de 2013). *XBee Digital I/O Adapter*. Obtenido de http://www.digi.com/wiki/developer/index.php/XBee_Digital_I/O_Adapter
- Iowa-Aquaponics. (s.f.). *Environment Data Acquisition System*. Obtenido de http://www.iowa-aquaponics.com/arduino/#Environment DAQ
- Karimi, K., & Atkinson, G. (1 de Junio de 2013). *What the Internet of Things (IoT) Needs to Become a Reality.* Recuperado el 7 de Agosto de 2013, de http://www.freescale.com/files/32bit/doc/white_paper/INTOTHNGSWP.pdf
- LibSSH2 Team. (s.f.). *LIBSSH2 the SSH library*. Obtenido de http://www.libssh2.org/
- Light In The Box Ltd. (2006). *6490 DHT11 Temperatura Humedad Sensor Module (Azul)*. Obtenido de http://www.miniinthebox.com/es/6490-dht11-temperatura-humedad-sensor-module-azul_p421815.html
- Mandat International. (30 de Marzo de 2012). *Universal Integration of the Internet of Things through an IPv6-based Service Oriented Architecture enabling heterogeneous components .* Recuperado el 15 de Noviembre

de 2013, de http://www.iot6.eu/images/stories/deliverables/IoT6_D1.1_v1.0.pdf

- Millán, R. (2006). *El Protocolo IPv6 (I)*. Obtenido de http://www.ramonmillan.com/tutoriales/ipv6_parte1.php
- Ministerio de Ciencia y Tecnología de ESPAÑA. (18 de Abril de 2004). *IPv6 Servicio de Información y Soporte*. Obtenido de Despliegue IPv6 en RedIRIS: http://www.6sos.org/pdf/la_transicion_de_las_redes_academicas:rediris_v2.pdf
- Nootropic Design. (1 de Noviembre de 2009). *Wireless Temperature Sensor*. Obtenido de http://nootropicdesign.com/projectlab/2009/11/01/wireless-temperature-sensor/
- Ranz, J. (2013). *The Things´s City*. Obtenido de La IoT según SAP : http://thingscity.com/la-iot-segun-sap/
- Sierra, A., & Cubo, A. (s.f.). *Representational State Transfer (REST).* Obtenido de http://trajano.us.es/~antonio/REST.ppt
- Telefónica. (4 de Abril de 2013). *IPv6: El motor de "La WEB de las Cosas"*. Recuperado el 10 de Noviembre de 2013, de http://blogthinkbig.com/ipv6-motor-internet-de-las-cosas-iot/
- The Core Team. (14 de Febrero de 2009). *Tiny Core Project*. Obtenido de http://www.tinycorelinux.net/intro.html
- The PHP Group. (2001). *Funciones de SSH2*. Obtenido de http://php.net/manual/en/ssh2.installation.php
- TunnelsUP. (30 de Noviembre de 2012). *XBee S2 Quick Reference Guide/Cheat Sheet and Video Tutorials to Getting Started*. Obtenido de http://www.tunnelsup.com/xbee-s2-quick-reference-guide-cheat-sheet/
- Valenzuela, R. (7 de Mayo de 2012). *Shields*. Obtenido de http://roberto-valenzuela.blogspot.com/2012/05/shields.html
- Zigbee labs. (7 de Julio de 2012). *Motes: Tercera parte, sacando la información de la red Zigbee*. Obtenido de http://www.zigbe.net/archivos/387