Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Máster Universitario en Dirección e Ingeniería de Sitios Web

RAMA INVESTIGACIÓN

Tecnologías Web y Web Semántica en el Internet de las Cosas

Trabajo Fin de Máster

Presentado por: Castro Cuasapaz, Sandra Elizabeth

Director/a: Boubeta Puig, Juan

Ciudad: Quito - Ecuador
Fecha: Septiembre 18 de 2018

Contenido

Índice de Figuras

Índice de Tablas

Resumen

El Internet de las Cosas (*Internet of Things*, IoT) es una tecnología que se le ha dado mayor énfasis actualmente, ya que permite tener procesos automatizados a través de la comunicación entre objetos interconectados. El introducir objetos inteligentes en la Web requiere que la tecnología y protocolos existentes para el envío de datos, se adapten a este proceso de comunicación y se mejore a través de aspectos de la web semántica. La Web actual soporta tecnologías y estándares abiertos que permiten controlar los dispositivos y el intercambio de datos entre ellos. Sin embargo, en el desarrollo de software para IoT se requiere conocer la tecnología más adecuada, los estándares, herramientas y ontologías de la web semántica que puedan emplearse para mejorar la interoperabilidad de objetos en el IoT. Para dar solución a estas cuestiones, en este Trabajo Fin de Máster (TFM) se realiza un estudio y descripción de las tecnologías Web para el intercambio de datos en el IoT. Entre estas tecnologías se encuentran: MQTT (*Message Queue Telemetry Transport*), CoAP (*Constrained Application Protocol*), WAMP (*Web Application Messaging Protocol*, WAMP), WebSockets, AMPQ (*Advanced Message Queuing Protocol*) y REST (*Representational State Transfer*). En particular, se elabora un análisis comparativo en base a los aspectos técnicos de las tecnologías descritas para determinar la solución más conveniente en aplicaciones IoT. Además, se realiza un análisis de ontologías y tecnologías semánticas que pueden intervenir en el procesamiento de IoT a través de la revisión de metodologías y estándares de W3C. Finalmente se presentan aplicaciones existentes que hayan implementado estas técnicas y se especifican sus ventajas.

Palabras Clave: web semántica, protocolos web, internet de las cosas.

Abstract

Internet of Things (IoT) is one of the technological trends which allows us to have automate processes through the communication between interconnected objects. The introduction of intelligent objects in the Web requires to adapt the technology and the existing protocols for sending data to this communication process. Moreover, this procedure could be improved through aspects of the semantic web. The actual Web supports open technologies and standards for the exchange of information and control of devices. However, in order to develop software for IoT, it is necessary to know the most appropriate technology, standards, tools and ontologies of the semantic web that could be applied to improve the interoperability of objects in the IoT. To solve these questions, a study and description of Web technologies for the exchange of data in the IoT would be carried out. Among these technologies are: MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), WAMP (Web Application Messaging Protocol, WAMP), WebSockets, AMPQ (Advanced Message Queuing Protocol) y REST (Representational State Transfer). In particular, a comparative analysis will be done based on the technical aspects of the described technologies to determine the most convenient solution in IoT applications. Then, an analysis of ontologies and semantic technologies for IoT processing will be executed through the revision of methodologies and W3C standards. Finally, existing applications that have implemented these techniques are presented in this master´s degree project, specifying their advantages.

Keywords: semantic web, web protocols, internet of things.

1. Introducción

En este capítulo se describe cuál es la motivación para la realización de este TFM y se explica también la estructura del documento contenida en siete capítulos.

1.1. Motivación

El Internet de las Cosas, IoT por sus siglas en inglés según lo define (Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M., 2013) es una red de elementos físicos con identificadores únicos y la capacidad de transferir datos a la red sin requerir de la intervención humana. Esta red convergente de objetos y personas conectados para enviar y recibir datos, requiere una participación importante de las tecnologías web que permitan a través de una conexión remota acceder a servicios y aplicaciones para monitorizar y controlar los dispositivos conectados al IoT. De esta manera, en la capa aplicación de la arquitectura tecnológica del IoT existen protocolos y tecnologías que son necesarios para manejar la comunicación entre objetos. Por lo que se requiere un estudio de estos protocolos, que con parámetros establecidos de acuerdo a un cierto tipo de aplicación IoT, permitan escoger la mejor tecnología para su desarrollo.

Según (Castells, 2003) la "Web Semántica" se define como la introducción de descripciones explicitas del significado y de la estructura de los contenidos y servicios disponibles en la web. Así pues, la Web Semántica corresponde a una ampliación de la web actual, en la cual los datos se describen y se relacionan de tal manera que establecen por si solos un contexto o significado permitiendo que las aplicaciones se pueden integrar y puedan entender la información fácilmente. Al integrar esta tecnología al IoT, a través de herramientas y ontologías que hagan uso de vocabularios estándares permitirá que un mayor número de aplicaciones puedan entender los contextos y dominios gracias a esquemas definidos. Es por ello la importancia de conocer qué aspectos de la tecnología semántica pueden utilizarse en aplicaciones o servicios en el IoT para que mejore la comunicación entre objetos.

En el desarrollo de software para IoT se requiere conocer la tecnología más adecuada para crear aplicaciones interactivas, así como los estándares, herramientas y ontologías de la web semántica que puedan aplicarse para la mejorar la comunicación de la Web con IoT. De esta manera, tanto los protocolos para el envío de datos como aspectos de la web semántica son temas que se estudiarán y se analizarán para conocer el impacto del uso de estas tecnologías en el Internet de las Cosas. Se destacan casos de éxito en los que se resaltará las ventajas de su implementación. Se emplean técnicas de investigación

cualitativas para poder analizar los resultados y plantear recomendaciones y "buenas prácticas" del uso de estas tecnologías.

1.2. Estructura del documento

La presente investigación llevada a cabo en este Trabajo Fin de Máster (TFM) está estructurado de la siguiente manera:

El capítulo 1 corresponde a la Introducción, en él se resume de forma clara cada parte del trabajo destacando el problema que se va a tratar, el planteamiento de la solución y se describe el contenido de los capítulos.

En el capítulo 2 se encuentra el estado del arte, un estudio minucioso que contempla el marco teórico del conocimiento existente en el problema planteado.

El capítulo 3 corresponde al planteamiento de objetivos y metodología de trabajo, se describe el objetivo general y específicos detallando lo que se desea conseguir y la metodología a seguir para la elaboración del documento.

En el capítulo 4, se describe las técnicas de investigación empleadas, detallando los pasos realizados y las herramientas empleadas que permitan analizar cada aspecto de los protocolos para el envío de datos y de la web semántica.

En el capítulo 5 se realiza análisis comparativo de tecnologías y protocolos utilizados para el envío de datos en aplicaciones IoT argumentando varios parámetros que deben ser consideradas en este campo. Como también se demuestra la creación de aspectos semánticos en un ambiente IoT, para esto se realiza un prototipo que capture datos y estos sean visualizados en la Web, utilizando una plataforma IoT en la nube.

El capítulo 6 corresponde a la discusión, tomando en cuenta los resultados obtenidos en el capítulo anterior, se plantea recomendaciones y buenas prácticas para optar por una tecnología adecuada para el desarrollo de estas aplicaciones.

Finalmente, en el capítulo 7 se detallan las conclusiones de la investigación y las líneas de trabajo futuro.

2. Estado del arte

En este capítulo se describe el marco teórico correspondiente a aspectos del IoT, entre ellos el hardware, software y los protocolos de comunicación necesarios para adquirir el conocimiento de generación de un ambiente IoT. Incluyendo a este paradigma de IoT se presentan las tecnologías y aspectos de la Web Semántica que se pueden incluir en un ambiente IoT para mejorar su interoperabilidad.

2.1. Internet de las Cosas

El IoT es una definición que se le ha dado a la interconexión de objetos o cosas de nuestra vida diaria con el internet (Arshdeep Bahga, 2014), por ende es una tecnología que toma cada vez más relevancia en todos los campos. El IoT interrelaciona objetos con identificadores para transferir datos a través del internet lo que conlleva a algunos desafíos de desarrollo, programación como también comunicación y tecnologías web.

Antes de exponer aspectos de la IoT, se requiere definir dos términos que suelen usarse indistintamente Internet y la web. El concepto de Internet corresponde a la capa física o la red compuesta por equipos de interconexión. El objetivo de esta red de redes es transportar información desde un origen a un destino utilizando tecnologías y protocolos para evitar retrasos, pérdida de información y problemas de seguridad. por otro lado, la Web corresponde a la capa aplicación de interconexión de documentos accesibles por el Internet. El objetivo principal de la Web es proporcionar el acceso a la información a través de Internet (Evans, 2011).

Pero más allá del enfoque de Internet, el concepto de Web de las Cosas (*Web of Things*, WoT) reúne los recursos que intervienen en los objetos, los datos y las personas en la Web, considera la forma en que se intercambian los datos. Lo que implica gran cantidad de oportunidades y desafíos en el desarrollo de aplicaciones que mejoren la calidad de vida de las personas y el trabajo en la industria (Gustafson, 2014).

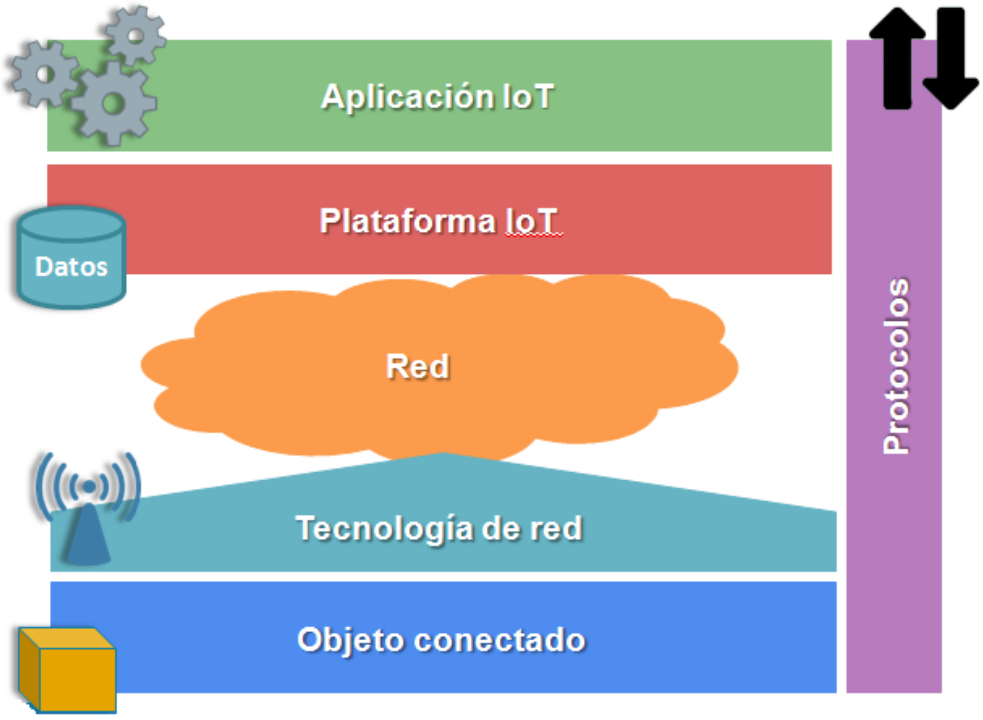
El hecho de conectar objetos a Internet, ha abierto la posibilidad de contar con el diseño de infinitas aplicaciones relacionadas con el uso de sensores, automatización de procesos y comunicación entre máquinas que faciliten las actividades diarias. Las soluciones a través del IoT pueden enfocarse en una amplia variedad de campos: hogar, logística, procesos de negocio, alimentación, gestión de energía, de agricultura, salud, ciudades, etc.

2.1.1. Arquitectura de IoT

La arquitectura de sistemas de IoT se puede dividir en cuatro capas: capa de detección de objetos, la capa de intercambio de datos, capa de integración de la información, y la capa de servicios de aplicaciones (Hua-Dong, 2011).

En la figura 2-1 se describe las capas que corresponden a los pilares sobre los que se asienta la IoT. La IoT cuenta con la capa de detección que permite la interconexión de objetos, interviniendo sensores y objetos físicos. Los conjuntos de sensores conectados se comunican entre sí, a través de la capa intercambio de datos, en la que intervienen los protocolos de comunicación y las tecnologías de red. Para los procesos de integración de información interviene la plataforma IoT que permite el tratamiento inteligente de datos. Debido al crecimiento de las nuevas tecnologías en la nube, se ofrecen numerosas aplicaciones de software en el campo de la IoT orientadas a la industria, negocios, comercio, consumidores, ciencia, salud, entre otras que corresponde a la capa de servicio de aplicación.

Figura 2-1. Arquitectura de IoT



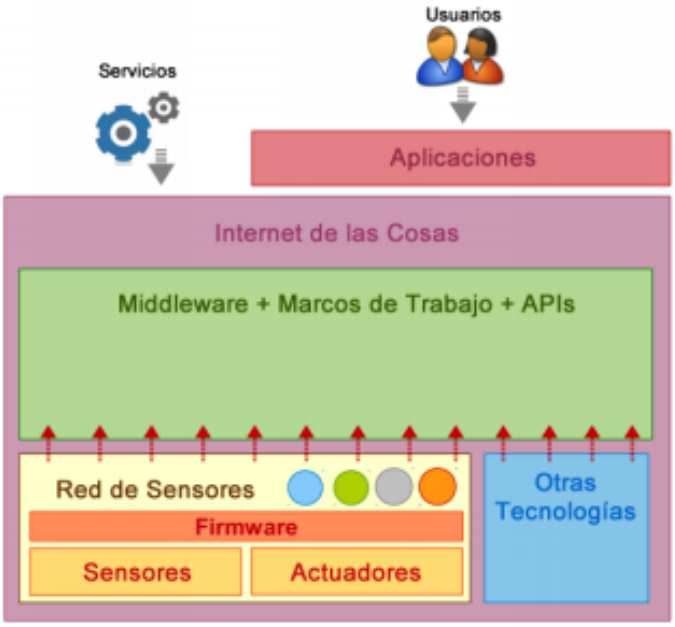
Fuente: Propia

Middleware para IoT

Un middleware es un software distribuido, que en el modelo de capas mostrado en la figura 2-2 se sitúa entre las capas inferiores que corresponde a la capa física de los objetos y las

capas de aplicaciones, su objetivo es lograr comunicación e interacción de todos los objetos de IoT (Perera, 2014).

Figura 2-2. Middleware para IoT



Fuente: (Perera, 2014)

2.1.2. Objetos Inteligentes

Los objetos inteligentes son objetos físicos que contienen una parte electrónica, que le permiten tener capacidad de comunicación ya sea con otros objetos o con un servidor centralizado a través de wireless, Bluetooth, IRDA, etc. Estos objetos también cuentan con la capacidad de almacenamiento y procesamiento, ya sea a través de la ejecución de algún software o a través de un sistema embebido. Ej.: computadoras, teléfonos inteligentes, Raspberry Pi, etc.

Objetos asistidos

Son aquellos dispositivos que no poseen todos los componentes para un servicio por lo tanto requieren de un componente adicional como los microprocesadores Raspberry PI, para permitir programarlo y complementarlo. Ej.: Lectores RFID.

Objetos simples

Estos pueden ser cualquier "cosa", no poseen capacidades de cómputo ni procesamiento, pero si contiene un almacenamiento menor. Ej.: Etiquetas RFID, NFC, códigos de barra o QR. Para que interactúe estos dispositivos con otros objetos requiere asociarse a una representación virtual. Así se observa en la figura 2-3.

Figura 2-3. Código de barras con URL acortada.

Fuente: (Rodríguez, 2015)

2.1.3. Identificación de objetos en la IoT

La base de comunicación para muchos objetos es la identificación por radiofrecuencia, entre las más aplicadas están: Identificación por radio frecuencia (RFID, Radio Frecuency Identification) (ISO, 2018) y comunicación de campo cercano (NFC, Near Field Communication) (ISO/IEC, 2013).

RFID (Radio Frecuencia)

Es la tecnología más utilizada, estas etiquetas permiten almacenar y recuperar datos. Está compuesta por una antena, un transductor de radio y material de encapsulado, existen etiquetas de lectura y de lectura/escritura. Los lectores RFID en cambio captan las señales de las etiquetas.

NFC (Comunicación de Campo Cercano)

Es similar al RFID, pero no son tecnologías compatibles, permite trabajar a cortas distancias. Los lectores NFC permiten leer y escribir etiquetas NFC. Actualmente está equipado en muchos teléfonos móviles. Se emplean para sistemas de pago, control de acceso, para puntos de información, etc.

2.1.4. Hardware para IoT

Corresponde a las plataformas para el diseño de objetos inteligentes específicos, que pueden ser elaboradas a través de la electrónica o por sistemas específicos como: Arduino[1], NodeMCU[2], Raspberry Pi[3], BeagleBone[4], Libelium Waspmote[5] y otros, que permiten construir un objeto inteligente.

Sistemas embebidos

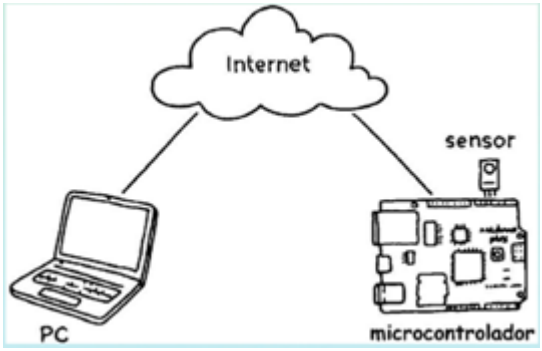
Un sistema embebido "*es una combinación de hardware y software computacional diseñado para enfocarse en una realización de una función específica*." (Valencia).

[1] https://www.arduino.cc/
[2] http://nodemcu.com/index_en.html
[3] https://www.raspberrypi.org/
[4] https://beagleboard.org/bone
[5] http://www.libelium.com/products/waspmote/

Microcontroladores

El elemento clave de internet de las cosas es el microcontrolador, que cuenta con la potencia necesaria de procesamiento para conectarse a internet y representa la interfaz entre el mundo físico y el Internet, así se observa en la figura 2-4.

Figura 2-4. Microcontrolador en el IoT



Fuente: (Pérez, F. A. F., & Guerra, J. L. G. , 2017)

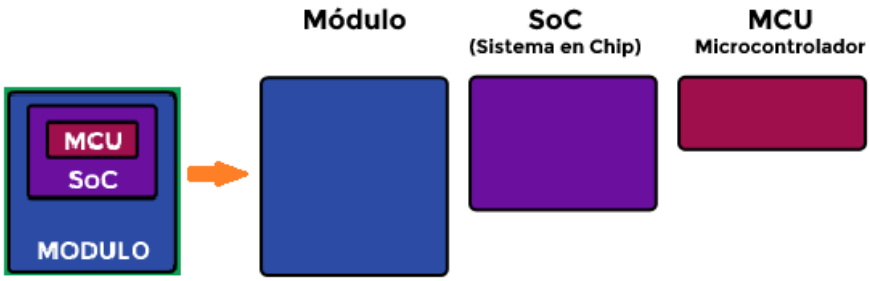
Tradicionalmente, para la programación de estos dispositivos era necesario contar con plataformas y herramientas específicas realizadas por electrónicos, sin embargo, actualmente existen microcontroladores que permiten emplear plataformas software modernas propietarias y no propietarias. Esto permite que, por ejemplo en .NET, se pueda utilizar el mismo lenguaje de programación (C#) y el mismo entorno de desarrollo (Visual Studio) en la creación de programas para pequeños dispositivos incrustados, smartphones, PC, servidores empresariales e incluso servicios en la nube. (C. Pfister, 2011.)

NodeMCU

Es una placa o kit de desarrollo que permite crear proyectos con dispositivos conectados al IoT. Es una plataforma abierta a nivel de hardware y software que tiene incorporado un chip ESP8266 que contiene un microcontrolador.

En la figura 2-5 se observa un esquema de una placa NodeMCU, en donde la parte central es el MCU (Microcontroller Unit) al cual se lo programa a través del kit de desarrollo.

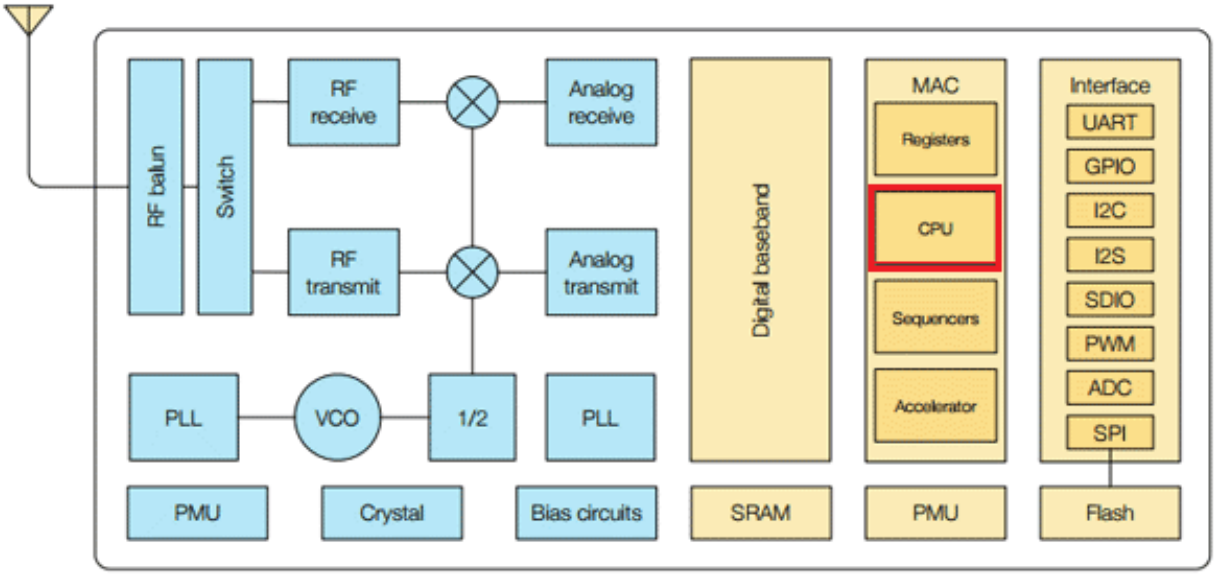
Figura 2-5. Esquema de la placa NodeMCU



Fuente: propia

La característica principal de ESP8266 es que tiene incorporada una placa WiFi que permite crear un ambiente IoT mediante conexión inalámbrica, en la figura 2-6 se puede observar un esquema del microprocesador.

Figura 2-6. Esquema microprocesador ESP8266



Fuente: Datasheet ESP8266

Arduino

Arduino es un dispositivo electrónico abierto, una plataforma con hardware y software libre, al igual de NodeMCU se lo emplea para crear prototipos de forma flexible y fácil. Arduino está compuesto por tres partes:

- Placa hardware: Contiene al circuito impreso con un microcontrolador, pines y puertos necesarios.
- Software de desarrollo: Corresponde a la herramienta para programar las acciones que debe realizar el dispositivo.

- Un lenguaje de programación: Semejante a C++, en las sentencias de control, estructura, operadores, etc.

Raspberry Pi

Es un minicomputador que almacena y procesa la información, trabaja sobre Linux tiene gran variedad de distribuciones y es una plataforma en constante crecimiento por la gran comunidad de usuarios que contribuyen a su desarrollo.

Red de Sensores Inalámbricos (WSN)

Es una red conformada por sensores pequeños que utilizan señales de radio para comunicarse entre ellos. "*Los nodos sensores son básicamente una minicomputadora con funciones básicas, estos tienen una unidad de procesamiento con capacidad de computación restringida, una memoria, un dispositivo de comunicación de radio una fuente de alimentación y uno o más sensores dependiendo del tipo del tipo del nodo sensor*" (Ortiz, 2017).

Los sensores permiten obtener varios datos del entorno midiendo parámetros como: temperatura, sonido, movimiento, posición, luz, ambientales, presión, posición, aspectos médicos, radiación, factor climático y eso hace que sean diversos.

2.1.5. Aplicaciones Software para IoT

En un sistema IoT, muchos de los dispositivos y objetos inteligentes requieren aplicaciones software. Estas aplicaciones definen la lógica del negocio, establecen el comportamiento y la respuesta del objeto o "cosa" del sistema, ya que aquí se define lo que se va a hacer, se podría decir que dotan de cierta inteligencia a los objetos.

Para la implementación de aplicaciones software se requiere de lenguajes de programación, APIs, frameworks y editores que dependen de las características y necesidades del sistema. De esta manera se establecen los siguientes tipos de aplicaciones:

- Aplicaciones software para objetos inteligentes.
- Aplicaciones software centralizadas.

Software para Objetos Inteligentes

Los protagonistas de un sistema IoT son los objetos inteligentes, ya sea diseñados como tal u objetos tradicionales convertidos para aquello, estos contienen cierto sistema embebido con diferentes sistemas operativos como: Android, Windows Embebido, Linux, Duinos-

Arduino, etc. El lenguaje a utilizar dependerá del sistema operativo empleado y de las técnicas de desarrollo que se sigan, en la tabla 2-1 se observa los lenguajes más empleados de acuerdo al sistema operativo del dispositivo.

Tabla 2-1. Software para objetos inteligentes

Software del dispositivo	Lenguaje de programación
Android	Java , C.
Windows Emedded	Java, Python, .Net C#, C, C++
Duinos-Arduino	Lenguaje C
Raspberry PI	Java, Python, C.

Fuente: propia

Entre ellos, Android es uno de los sistemas operativos con mayor crecimiento, ya que comparada con otras soluciones permite un acceso al hardware de manera relativamente sencilla.

Software para Sistemas Centralizados

Los sistemas centralizados son otra parte importante en los sistemas IoT, ya que controlan los dispositivos conectados a él, el trabajo de este punto de control puede estar llevado a cabo por máquinas virtuales, un computador físico o por servicios de cloud computing, que por lo general ejecutan sistemas operativos comunes.

Las aplicaciones que suelen implementarse en estos sistemas son:

- Recolectar datos de objetos inteligentes.
- Recibir/enviar notificaciones ordenes o comandos.
- Procesar información y realizar toma de decisiones.
- Ofrecer servicios a otros objetos.

Para la comunicación, entre los sistemas centralizados y los objetos se debe poseer la misma tecnología de comunicación, algunas tecnologías de corto alcance son: Bluetooth, Wi-Fi, NFC, RFID, etc.

2.1.6. Servicios en la nube para IoT

Se cuenta con tres principales formas de despliegue de Cloud, por un lado, la nube pública la cual se puede acceder desde cualquier lugar con ciertas restricciones, otro modelo es la nube privada (*on premise*) que se implementa de manera local en una empresa para uso exclusivo y la nube híbrida una combinación entre pública y privada. El modelo de

despliegue más viable para IoT sería una nube pública debido a que el IoT está ligado al Internet (Campoverde, A. M., Hernández, D. L., & Mazón, B. E., 2015).

Entre las plataformas públicas más importantes en la nube se tienen:

Google Cloud IoT

La plataforma (Cloud, 2018), brinda servicios IoT en la nube, que permite conectar, administrar y transferir datos de objetos conectados en distintas ubicaciones. Permite escalar los sistemas con respecto a la cantidad de tráfico de datos, visualizar y monitorizar en tiempo real los dispositivos para la toma de decisiones según corresponda. Es de pago, sin embargo, ofrece un período gratuito. Es compatible con los protocolos estándar de comunicación HTTP y MQTT.

AWS IoT

Es una gran plataforma en la nube de Amazon (AWS, 2018), ofrece servicios de recopilación, análisis y gestión de datos y comunicación entre servicios en tiempo real. Es una plataforma de pago que depende de los servicios que se utilicen. Cuenta con servicios para controlar los dispositivos, tratamiento de datos y administración de dispositivos. Admite protocolos MQTT, HTTP y WebSockets.

Azure IoT Suite

Propiedad de Microsoft (Microsoft, 2018), esta plataforma en la nube permite gestionar datos, dispositivos y además ofrece servicios adicionales. Admite protocolos de comunicación MQTT, MQTT sobre Websockets, AMQP, AMQP sobre WebSockets y HTTP.

IBM Watson IoT

Desarrollada por IBM (IBM, 2018), ofrece varios servicios de IoT, clasificados por áreas específicas energía, electrónica, automóviles e industria. Permite administrar los dispositivos, acceder a aplicaciones potentes y analizar datos en tiempo real. Usa el protocolo de comunicación MQTT.

Ubidots

Es una herramienta de análisis y visualización de datos, que permite capturar datos del entorno y convertir esos datos a parámetros que indiquen ejecutar acciones en tiempo real. Su origen fue automatizar remotamente procesos para el cuidado de la salud. Permite visualización interactiva de datos en tiempo real y una personalización de la plataforma con

su código (Ubidots, 2018). Permite acceder a los dispositivos a través de protocolos como HTTP, MQTT.

Xively

Es una plataforma enfocada a IoT que actualmente es parte de la familia Google Cloud Plataform (XIVELY, 2018), permite conectar dispositivos, administrarlos e integrar los datos que se obtienen con otros sistemas. Cuenta con un servicio comercial para empresas que requieran soporte dedicado en su sistema.

ThingSpeak

Es una plataforma abierta para IoT que permite guardar, analizar y visualizar los datos obtenidos por los sensores o aplicaciones (The MathWorks, 2018). Permite conectar objetos, servicios y publicar la información de objetos conectados en canales.

Sofía 2

Es una plataforma open-sorce que ofrece servicios IoT en la nube, es multiplaforma, multilenguaje e independiente de las las comunicaciones (Indra, 2018). Tiene un enfoque semántico con el objetivo de lograr interoperabilidad entre aplicaciones que comparten estos conceptos.

Tal como se menciona en el trabajo de (Chuquimarca Sarango, 2017), en la que realiza un análisis comparativo de plataformas IoT en la nube, cada plataforma tiene sus fortalezas y debilidades enfocadas a alcanzar un mercado específico, por ello la elección de un servicio u otro dependerá de los requisitos propios de cada desarrollador.

2.1.7. Aplicaciones Web para IoT

Son aplicaciones software que se ejecutan a través de un navegador en el internet. Para construir aplicaciones web, en este caso IoT, se requiere de frameworks en tiempo real, ya que ofrecen mejores alternativas en cuanto a peticiones concurrentes en consideración a los de arquitectura bloqueante. Ya que permite servir a varias aplicaciones usando un bucle libre de entrada-salida sin bloquear a los procesos.

A continuación, en la tabla 2-2 se presentan los frameworks en tiempo real más usados, entre ellos están: Node.js, Tornado y Play.

Tabla 2-2. Frameworks en tiempo real

Framework	Modelo de ejecución	Características	Lenguaje de programación soportado	URL
Node.js	No bloqueante	• Capacidad nativa de trabajo con websockets. • Incluye librerías IoT para harware abierto(arduino, Raspberry, Beaglebone, entre otros.)	JavaScript	https://nodejs.org/es/
Tornado	Bloqueante	• Maneja miles de conexiones concurrentes. • Posee herramientas de seguridad y autenticación. • Usa blucle de eventos de un subproceso.	Phyton	http://www.tornadoweb.org/en/stable/
Play	No bloqueante	• Soporta conexiones concurrentes. • Soporte para bases de datos y web sockets. • Es full stack(viene con herramientas incorporadas)	Java o Scala	https://www.playframework.com/

Fuente: propia

2.2. Tecnologías Web para IoT

A continuación se explican los protocolos que han sido utilizados en web y ahora también se utilizan en ambientes IoT.

2.2.1. Protocolo de Internet (*Internet Protocol*, IP)

El uso de protocolo IP es indispensable en la IoT, ya que con la constante evolución de la IoT la interoperabilidad de los sistemas con IP tomará gran importancia. Tecnologías como 6LoWPAN, Ethernet y Wi-Fi requieren en gran medida del protocolo de capa red IPv4 y su nueva versión IPv6.

El uso de las tecnologías y protocolos desarrolladas para la Web para el desarrollo de sistemas IoT es posible, aunque no presentan la misma eficacia que se obtiene con protocolos creados para aquello, sin embargo, se lo puede implementar empleando estándares comunes como HTTP(S) como estándar universal por su simplicidad, flexibilidad, ligereza y escalabilidad. URIs identificadores importantes para el direccionamiento de objetos. Mediante representación de datos como XML (Extensible Markup Languaje) o JSON(JavaScript Object Notation) y REST para que los objetos envían y reciban información y publiquen sus capacidades (ELECTRONICS, 2015).

2.2.2. Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol*, HTTP)

HTTP es un protocolo de comunicación que permite transferencia de información en la Web siguiendo el modelo cliente-servidor. Sin embargo, por motivos de seguridad en sistemas IoT los dispositivos clientes no deben recibir peticiones de conexión sino ellos enviarán las solicitudes al servidor, de esta manera se evitará el acceso a la red de dispositivos de IoT por máquinas externas no permitidas. Este protocolo fue propuesto por Tim Berners-Lee para satisfacer los requerimientos del sistema distribuido de información de la Web. Trabaja sobre TCP/IP, a través de operaciones request/response las mismas que incluyen un objeto o recurso a través de un identificador URL.

2.2.3. WebSockets

WebSockets forma parte de la especificación HTML5, establece un canal sobre un socket TCP y permite una comunicación full dúplex en cualquier aplicación cliente/servidor, ya que está creado para ser implementado en clientes web y servidores web. Es un estándar IETF (RFC 6455). En el lado del cliente, en los navegadores web ya se encuentra implementado WebSockets. En el campo de IoT, la interacción entre HTTP y WebSockets resulta una solución apropiada si los dispositivos soportan las cargas de HTTP.

2.2.4. Protocolo Extensible de Mensajería y Presencia (*Extensible Messaging and Presence Protocol*, XMPP)

XMPP fue diseñado para el intercambio de mensajes, es ampliamente utilizado en Internet y se encuentra estandarizado por el IETF. XMPP es un ejemplo de uso de tecnología web existente en el campo de IoT.

XMPP se ejecuta sobre TCP, trabaja bajo en concepto publicación/suscripción (asíncrona), pero también solicita sistemas de mensajes de respuesta (síncronos). Está diseñado para

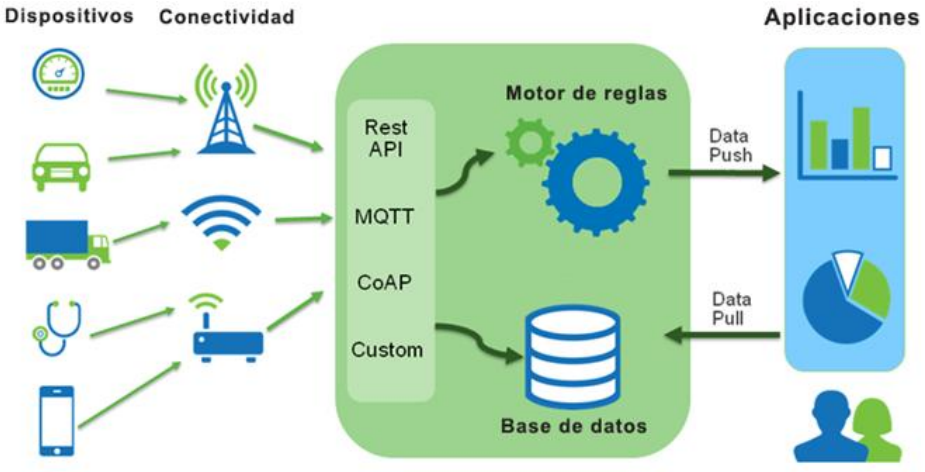
aplicaciones de intercambio de mensajes pequeños y de baja latencia. XMPP tiene seguridad TLS/SSL integrada en el núcleo de la especificación, no proporciona opciones de QoS, sin embargo, hereda la confiabilidad de TCP. Este protocolo utiliza mensajes XML(Lenguaje de marcado extensible) que crea una sobrecarga adicional debido a etiquetas innecesarias y aumento de consumo de energía en el análisis XML, así lo plantea (Vasileios Karagiannis, 2015).

XMPP es un protocolo que por sus ventajas de direccionamiento, seguridad y escalabilidad hace que sea óptimo para aplicaciones IoT de administración de electrodomésticos de consumo a gran escala.

2.3. Tecnologías y Protocolos para IoT

Actualmente, muchos organismos de estandarización trabajan por adaptar las tecnologías actuales del Internet para el IoT, es por esto que tomando en cuenta las limitaciones y características de los dispositivos interconectados se plantean soluciones que a través de protocolos, tecnologías y herramientas se pretende solventar las dificultades y retos del IoT. En la figura 2-7, se observa la participación de los protocolos en la comunicación de dispositivos u objetos para el almacenamiento y demás procesos que permitan visualizarlos.

Figura 2-7. Protocolos de comunicación para IoT



Fuente: propia

Antes de estudiar los protocolos para IoT, es importante describir conceptos que se mencionan a lo largo de este documento.

Modelo publicador/suscriptor

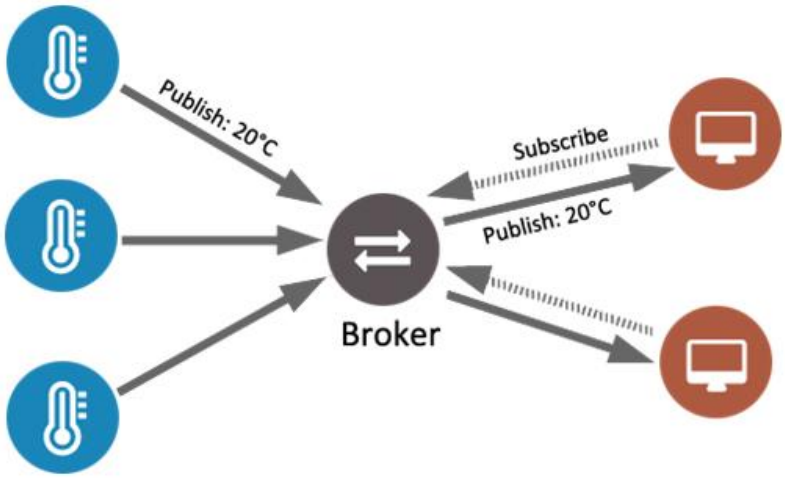
El patrón publicación/suscripción (PUB/SUB) es una alternativa al modelo tradicional cliente-servidor (REQ/RES), en el que se tiene una conexión directa entre el cliente y el servidor.

En este modelo en cambio, se tiene un intermediario entre el que publica y el que se suscribe (hivemq, 2018). Entre el PUB/SUB no conocen la existencia el uno del otro, pero si conocen a un tercer componente llamado broker que se encarga de filtrar los mensajes entrantes y los distribuye.

- Broker: Es el intermediario que comunica al publicador y suscriptor.
- Publicador: Publicará/enviará un mensaje al broker para que lo distribuya.
- Suscriptor: Recibe el mensaje del broker.
- Topics: (temas) Se utilizan para garantizar la entrega de mensajes, se envían únicamente a los suscriptores interesados en él.

Tal como se observa en la figura 2-8, el cliente publica los "topics" a través de mensajes y estos llegarán únicamente a los nodos que se suscribieron a él, este proceso es controlado por el "broker".

Figura 2-8. Modelo Publicación/Suscripción



Fuente: propia

El intermediario "broker", es el corazón del protocolo de publicación/suscripción. Es el principal responsable de recibir todos los mensajes, filtrarlos, decidir quién está interesado en ellos y luego enviar el mensaje a todos los clientes suscritos. Otra responsabilidad es la autenticación y autorización de los clientes.

2.3.1. Protocolo MQTT *(Message Queue Telemetry Transport*)

MQTT fue desarrollado por un trabajo conjunto entre IBM y Cirrus Link, cuando su caso de uso fue crear un protocolo con mínima pérdida de batería y un ancho de banda mínimo conectando oleoductos por satélite (hivemq, 2018). Existen muchas controversias respecto a su significado, por eso se lo denomina únicamente por sus siglas MQTT, no tiene acrónimo.

Luego que IBM utilizara MQTT internamente durante bastante tiempo, la versión 3.1 se lanzó libre de regalías en 2010. MQTT está estandarizado por el Organismo Internacional de Estandarización (OASIS) y actualmente cuenta con su última versión 3.1.1.

Es un protocolo abierto de capa transporte usado para la comunicación máquina-máquina (M2M), y para aplicaciones donde se requiere un espacio de código pequeño y/o ancho de banda de red como en IoT. Los mensajes se transmiten en forma binaria usando el modelo denominado publicador-suscriptor. Es un protocolo muy liviano, que se destaca cuando se transfieren datos a través del cable en comparación con protocolos como HTTP, ya que solo tiene una sobrecarga de paquete mínima. Otro aspecto importante es que MQTT es extremadamente fácil de implementar en el lado del cliente. Esto encaja perfectamente para dispositivos restringidos con recursos limitados (hivemq, 2018).

En cuanto a la seguridad, es importante mencionar que MQTT permite intercambiar los mensajes utilizando diferentes niveles, mediante autenticación de certificados digitales, autenticación de login o a través del transporte SSL/TSL (MS, 2017).

En el modelo de PUB/SUB manejado por MQTT, el publicador puede definir la Calidad de Servicio (QoS) para la entrega de mensajes en tres niveles:

- **Nivel 0:** "A lo sumo una vez", este nivel de servicio garantiza que un mensaje podría entregarse de acuerdo al mejor esfuerzo, donde la pérdida puede ocurrir.
- **Nivel 1:** "Al menos una vez", donde se garantiza la llegada de los mensajes, pero pueden producirse duplicados.
- **Nivel 2:** "Exactamente una vez", donde se garantiza que el mensaje llegue exactamente una vez. Para esto el publicador deberá guardar el mensaje hasta que el intermediario confirme en envío a través de la red.

A pesar de los niveles de QoS que presta este protocolo, es una tarea compleja llevarlo a un sistema multi-tenant.

2.3.2. Protocolo de Aplicación Restringida (*Constrained Application Protocol*, CoAP)

El protocolo de aplicación restringida es un protocolo de transferencia web, creado por la IETF para usar con nodos y redes restringidas (baja potencia y altos índices de pérdidas). El protocolo está diseñado para aplicaciones máquina-máquina como para aplicaciones de energía inteligente o construcción automatizada.

CoAP sigue el modelo solicitud/respuesta entre puntos finales correspondiente al modelo cliente/servidor, admite el descubrimiento de servicios y recursos (objetos, cosas o personas) a través de identificadores URI. CoAP está diseñado para operar en conjunto con HTTP para la integración con la Web a través de un proxy HTTP/CoAP. Permite realizar operaciones REST, sin dejar a un lado funcionalidades importantes como compatibilidad, multidifusión, muy baja sobrecarga y simplicidad para entornos restringidos (IETF_RFC7252, 2016).

CoAP opera bajo el protocolo de capa transporte UDP, sin embargo, algunas de las características TCP se reproducen allí. Distingue entre menajes que se pueden confirmar, en los mensajes CoAP las solicitudes/respuestas se intercambian de forman asincrónica. Usa codificación binaria para reducir la sobrecarga del protocolo.

2.3.3. Protocolo de Mensajería de Aplicaciones Web (*Web Application Messaging Protocol*, WAMP)

WAMP es un subprotocolo de Websockets, que se ejecuta sobre TCP confiable y no implementa mecanismos de confiabilidad propios. Si es necesario, se pueden asegurar usando Websocket sobre TLS / SSL (Vasileios Karagiannis1, 2015).

WAMP soporta dos modos de comunicación: Llamadas de Procedimientos Remotos (RPC) y publicador/subscriptor (PUB/SUB). Websocket no está diseñado para dispositivos con recursos limitados, sin embargo, está diseñado para la comunicación casi en tiempo real, es seguro y con el uso de WAMP puede proporcionar sistemas de mensajería eficientes.

La comunicación en WAMP se basa en tópicos identificados a través de URIs. WAMP no necesita de intermediarios para la comunicación con clientes web (navegadores), esta característica hace que sea posible la entrega de datos en tiempo "real" desde algún cliente Web (Campoverde, 2015).

2.3.4. Protocolo de Cola de Mensajes Avanzado (*Advanced Message Queuing Protocol*, AMQP)

AMQP es un protocolo de estándar abierto de la capa aplicación, que surgió de la industria financiera, por lo que se caracteriza por su fiabilidad y la interoperabilidad. Puede utilizar diferentes protocolos de transporte, sin embargo, prevalece el uso de TCP, por ser un protocolo confiable.

AMQP proporciona comunicación publicador/suscriptor asincrónica con mensajería. La principal ventaja es su función de almacenamiento y reenvío que garantiza la fiabilidad incluso después de las interrupciones de la red. (Frank T. Johnsen, 2013). Los mensajes AMQP poseen cabecera (propiedades) y el cuerpo (mensaje). Se caracteriza por la flexibilidad de su enrutamiento, transacciones y seguridad.

Esta característica de transaccionalidad es una ventaja de AMQP sobre MQTT y ha sido utilizado en escenarios más amplios y complejos que MQTT, como en Google y la NASA.

2.3.5. Transferencia de Estado Representacional (*Representational State Transfer*, REST)

REST no es realmente un protocolo sino una arquitectura. REST utiliza comandos HTTP de solicitud/respuesta mediante los métodos HTTP: GET, POST, PUT y DELETE para proporcionar un recurso (Fielding, 2000). El tipo de contenido que maneja es XML o JSON (JavaScript Object Notation) y depende del servidor HTTP y su configuración.

REST es compatible con todas las plataformas comerciales de nube M2M. Puede implementarse en teléfonos inteligentes y tabletas fácilmente porque solo requiere una biblioteca HTTP que está disponible para todas las distribuciones de sistemas operativos. Las características de HTTP pueden ser utilizadas en la arquitectura REST, incluido la autenticación y el tipo de contenido de negociación (Bipin Upadhyaya, 2011).

Los servicios RESTful pueden hacer uso de TLS / SSL para seguridad. Sin embargo, hoy la mayoría las plataformas M2M comerciales no son compatibles con las solicitudes HTTPS. Aunque REST se usa ampliamente en plataformas M2M comerciales, es poco probable que se convertirá en un protocolo dominante debido a que no se puede implementar fácilmente y usa HTTP, por lo que no hay compatibilidad con los dispositivos de comunicación restringida.

2.4. Web Semántica

La Web semántica es una extensión de la actual, permite un trabajo cooperativo entre personas y computadores otorgando un significado bien definido a la información. De tal manera que diferentes aplicaciones ubiquen, razonen, reutilicen el contenido en la web a través de datos bien definidos (Tim Berners-Lee, 2001).

El objetivo de la web semántica es que la información pueda ser comprendida por los computadores sin requerir la intervención de una persona, es decir convertir los datos en conocimiento a través de un esquema común que referencie los datos que se encuentran en una página web a metadatos (Tello, Ontologías en la Web Semántica, 2013).

Los metadatos deben determinar tanto un esquema de datos como los axiomas que puedan aplicarse en el dominio de conocimiento correspondiente.

2.4.1. Elementos de la Web Semántica

Tres aspectos importantes que intervienen en la web semántica son:

- Lenguaje de marcado apropiados: lenguajes de etiquetado estructurado y explícito, como XML que permite definir un lenguaje específico para cada tipo de información.
- Metadatos: permiten describir recursos de información para representar, indexar y buscar los documentos de forma precisa.
- Ontologías para describir las relaciones entre recursos y representar el conocimiento de un dominio a través de procedimientos normalizados (Codina, 2013).

Metadatos

Son descripciones estructuradas, basadas en reglas específicas que permiten ubicar, recuperar, relacionar y administrar recursos u objetos de información (Pastor, 2013).

Los metadatos requieren de semántica, sintaxis y unificación bajo una normativa que agrupe la diversidad de plataformas, para esto se crearon los RDF (*Resource Description Framework),* que de manera estándar y común describen los documentos, permitiendo a través de vocabularios RDF representar dichos metadatos.

Existe una organización, DCMI (*Dublin Core Metadata Initiative*) que fomenta el uso de estándares interoperables de metadatos a través del desarrollo de vocabularios especializados, Dublin Core[6] es un modelo de metadatos elaborado.

[6] http://dublincore.org/

Existen organizaciones que trabajan para estandarizar lenguajes, tecnologías y herramientas que permitan a los datos de la web definirlos, enlazarlos y convertirlos en conocimiento y pueda ser reutilizado entre aplicaciones, una de ellas es SematicWeb[7]. Tim Berners-Lee, uno de los inventores de la Web, defiende el desarrollo de la Web con conocimientos (Tello, Ontologías en la Web Semántica).

Ontologías

La ontología es una especificación de conceptos y relaciones de manera explícita, formal y compartida para que sea entendible y utilizable por los computadores, así lo manifiesta (Gruber, 1993).

Según (Gruber, 1993), los componentes que intervienen en las ontologías son:

- Conceptos: definiciones a normalizar.
- Relaciones: interacciones entre conceptos, también se lo denomina taxonomía del dominio. Ej.: trabaja-en.
- Funciones: relaciones en las que se realiza una operación con los elementos de la ontología. Ej.: asignar-fecha.
- Instancias: representa determinado objeto de un concepto.
- Axiomas: son teoremas que deben cumplir los elementos de una ontología.

Estos componentes permitirán a través de ontologías, realizar búsquedas más rápidas y específicas por medio de agentes web.

Los RDF permiten crear una jerarquía de conceptos utilizando las relaciones taxonómicas. Ciertos sitios optan por incluir anotaciones utilizando esquemas RDF, sin embargo, se requiere de un lenguaje de marcado especializado, con capacidad expresiva para representar los conocimientos de las ontologías.

La web semántica plantea una estructura multinivel, partiendo del uso de metadatos para la descripción de recursos, continuando con la definición de reglas de inferencia a través de ontologías.

[7] www.w3.org/standards/semanticweb/

Recursos

Recursos se define a las capacidades de los dispositivos hardware o software, por ejemplo, un dispositivo puede tener sensores de temperatura, humedad, movimiento. Estos tres recursos se pueden agrupar en un solo servicio. Para relacionar características en común directamente evitando redundancia de datos.

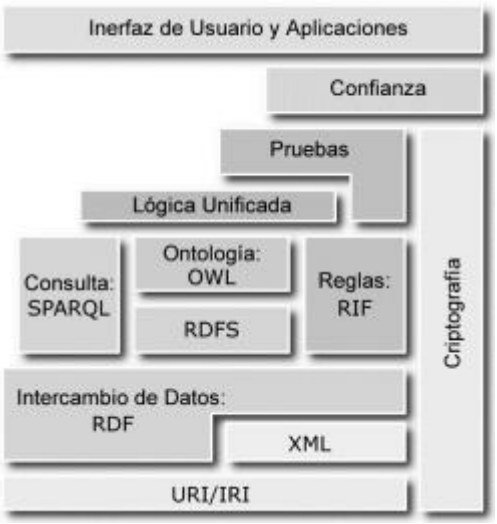
2.4.2. Tecnologías de la web semántica

La Web Semántica es una Web de Datos, datos de fechas, números, propiedades, etc. De esta manera se tendría una importante cantidad de estos en un formato estándar, común, accesibles y manejables. A través de las tecnologías de la web semántica se pretende crear un entorno que permita a las aplicaciones acceder a esos datos y a sus relaciones para realizar inferencias (LinkedData, 2015). Al hablar de inferencias se está hablando de la generación de nuevas relaciones entre los datos en base a un conjunto de reglas y procedimientos automáticos.

Existen también herramientas que permiten configurar puntos finales de consulta para acceder a los datos, para esto la W3C cuenta con tecnologías como: RDF, RDFa, RIF, SPARQL, entre otros).

La W3C propone una pila de Tecnologías para la implementación de la web semántica, así se muestra en la figura 2-9.

Figura 2-9. Pila de Tecnologías Semánticas para IoT.



Fuente: (Rodríguez, 2015)

Como se puede observar, el modelo de capas se inicializa identificando un recurso, a través del URI (Identificador Uniforme de Recursos), que es una cadena de caracteres definido en

RFC (3986). Para definir un concepto de manera estructurada y legible para personas y ordenadores se tiene XML (Lenguaje de Marcado Extensible).

RDF (Marco de Trabajo para Descripción de Recursos) permite relacionar conceptos para crear expresiones, en donde el vocabulario establecido se define en XML. RDFS (Esquema de Marco de Trabajo para Descripción de Recursos) permite definir modelos de conocimiento describiendo clases y recursos para definir ontologías y vocabularios.

OWL (Lenguajes de Ontologías para la Web) permiten al igual que RDFS describir modelos de conocimiento u ontologías, pero con mayores características como cardinalidad, límite de valores, entre otras, evitando ambigüedades y creación de ciclos.

A través de SPARQL (Lenguaje y Protocolo de Consulta RDF) se puede realizar consultas en RDF, RDFS y OWL. El (Lenguaje de Reglas para la Web Semántica) a través de reglas permite crear inferencias más complejas, se define también RIF[8] (Formato de Intercambio de Reglas) para la interoperabilidad entre reglas de diferentes lenguajes y aplicaciones. El uso de la lógica permite revisar la consistencia de información, validar, jerarquizar y clasificar las clases unificando las descripciones ontológicas con las reglas, sin embargo, aún no se estandariza esta capa.

Para brindar seguridad a los datos se establece la capa Criptografía. La capa de Pruebas y Confianza pretende verificar la veracidad de la información que se obtiene, evitando datos incoherentes y validando la calidad de la fuente de información. Finalmente la capa de interfaces de usuario y aplicaciones como motores de búsqueda o agentes de software (Rodríguez, 2015).

XML (Lenguaje de Marcado Extensible, *Extensible Markup Language)*

XML[9] es un formato simple de datos basado en texto para representar información estructurada y compartirla entre programas, entre personas o entre ellos. Se deriva de un formato más antiguo llamado SGML (ISO 8879), pero con adaptaciones para hacerlo más adecuado para el uso en la web. En la figura 2-10 se muestra un ejemplo de un archivo XML para un catálogo de vestidos.

[8] https://www.w3.org/standards/techs/rif#w3c_all

[9] https://www.w3.org/standards/xml/

Figura 2-10. Archivo XML de un catálogo de vestidos

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalogo SYSTEM "/catalogo.dtd">

<catalogo>
  <vestido codigo="vest001">
    <tipo demarca="TH">Vestido de Gala</tipo>
    <modelo>Princesa</modelo>
        <color tonalidad="negro grafito">Negro</color>
        <talla>Large</talla>
    <tela> Crep saten </tela>
    <diseñador fecha="25-02-2015">Ralph Lauren</diseñador>
        <temporada imagen="vestido1"></temporada>
        <precio moneda="dolar">99.90</precio>
  </vestido >
</catalogo>
```

Fuente: propia

XML Schema

Es un lenguaje para expresar restricciones sobre documentos XML, los más utilizados y definidos en la W3C son DTD y W3C XSD. XML Schema[10] proporciona documentación legible para personas y procesable para una máquina.

Bajo la norma Xml Schema de la W3C (*Web Ontology Working Group*), XML es un estándar que permite definir tipo de documentos y una colección de etiquetas para procesarlos y explotarlos automáticamente (Pedraza Jiménez, 2007).

Figura 2-11. Archivo DTD que valida el XML de catálogo de vestidos.

```
<!ELEMENT catalogo (vestido)+>
<!ELEMENT vestido (tipo, modelo, color, tela+, fecha, precio, temporada*)>
<!ELEMENT tipo (#PCDATA)>
    <!ATTLIST tipo demarca IDREF #REQUIRED>
<!ELEMENT modelo (#PCDATA)>
<!ELEMENT color (#PCDATA)>
    <!ATTLIST color tonalidad NMTOKENS #REQUIRED>
<!ELEMENT tela (#PCDATA)>
<!ELEMENT diseñador (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT temporada EMPTY>

<!ATTLIST vestido codigo ID #REQUIRED>
<!ATTLIST diseñador fecha NMTOKEN>
<!ATTLIST precio moneda (euro|dolar) valor CDATA>
<!ATTLIST temporada imagen ENTITY #REQUIRED>
```

[10] https://www.w3.org/standards/xml/schema.html

RDF (*Resource Description Framework,* Marco de Descripción de Recursos)

RDF es un modelo para representación abstracta de información en una relación sujeto-predicado-objeto que se denomina grafo. Para identificar el espacio de nombres de recursos de la información utiliza las URIs (*Uniform Resource Identifier*).

El objetivo de los RDF[11] es garantizar la interoperabilidad entre aplicaciones evitando la intervención de personas y empleando metadatos para describir recursos como sitios web, personas u objetos (Senso, 2003).

La tabla 2-3 muestra las equivalencias de los componentes básicos RDF, aquí se puede observar las relaciones entre ellos mediante un ejemplo.

Tabla 2-3. Componentes básicos RDF

Término Lingüístico	Sujeto	Predicado	Objeto
Término lógico	Recurso	Propiedad	Valor
Ejemplo	Páginas web	Título	Multicines

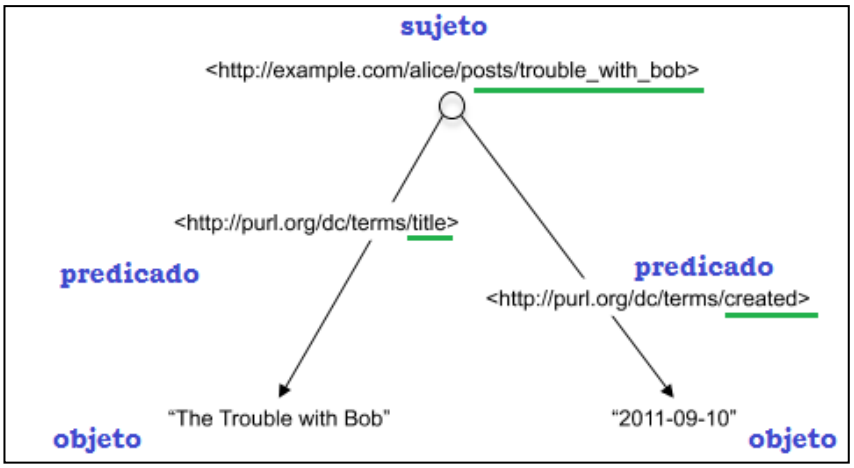
Fuente: propia

Los datos en RDF se pueden escribir en diferentes formatos, sin embargo, W3C recomiendan utilizar RDF-a para la web semántica ya que es como una extensión de documentos XML, HTML y XHTML.

En la figura 2-12 se observa un grafo con dos tripletas RDF, correspondiente a un post de redes sociales, por un lado, describiendo el título del post y por otro lado se describe la fecha de creación del post. RDF emplea vocabularios para proporcionar un lenguaje universal a través de propiedades expresadas como URL. El vocabulario RDF se encuentra definido en W3C, sin embargo, pueden crearse vocabularios personalizados creándolos a partir de instrucciones definidas en el RDF Schema.

[11] https://www.w3.org/standards/techs/rdf#w3c_all

Figura 2-12. Grafo con tripletas RDF



Fuente: http://skos.um.es/TR/rdfa-primer/

RDFa (Marco de descripción de Recursos en Atributos, *Resource Description Framework in Attributes*)

Es un conjunto de extensiones XHTML, HTML o cualquier lenguaje basado en XML que a través de los atributos de los elementos meta y link permite introducir descripción en la información para que sea aprovechada de mejor manera, es decir poniendo semántica a los documentos. Utilizando atributos HTML sencillos, RDFa permite establecer datos entendibles para personas con indicadores entendibles por ordenadores para su interpretación por programas o agentes de búsqueda.

El modelo de datos RDF pretende maximizar la reutilización de vocabularios, en cambio RDFa es una técnica para expresar los datos RDF dentro de documentos HTML para que sean entendibles por los ordenadores.

Figura 2-13. Anotaciones RDFa en un documento HTML

Ejemplo

```
<html>
<head>
  ...
</head>
<body>
  ...
  <h2>The Trouble with Bob</h2>
  <p>Date: 2011-09-10</p>
  ...
</body>
```

Ejemplo

```
<html>
<head>
  ...
</head>
<body>
  ...
  <h2 property="http://purl.org/dc/terms/title">The Trouble with Bob</h2>
  <p>Date: <span property="http://purl.org/dc/terms/created">2011-09-10</span></p>
  ...
</body>
```

Fuente: http://skos.um.es/TR/rdfa-primer/

Como se observa en la figura 2-13, a lado izquierdo se encentra un ejemplo de un documento html con sus etiquetas, su información mostrada en un navegador es entendible para personas, pero no para máquinas, a lado derecho marcado con color rojo se encuentran anotaciones RDFa que expresan los datos de una manera estructurada para que tanto las personas como las máquinas entiendan que esta información corresponde a un título y a una fecha de creación de un recurso.

RDFa utiliza URLs como identificadores para evitar ambigüedades en la terminología, en las URL se incluyen propiedades del recurso, en el por ejemplo son "title" o "created".

RDF Schema

Especifica las instrucciones que se deben seguir para la creación de vocabularios, que entre otros aspectos implica:

- El vocabulario debe tener asignado un identificador URL. Por ejemplo, de URL para un vocabulario de monitoreo de transporte: http://example.com/transporte/vocab#.
- El documento del vocabulario debe contener las clases y propiedades. Por ejemplo, en las clases "vehículo" y "persona", una propiedad podría ser "conducidoPor" que relaciona la persona con el vehículo que conduce.
- Una vez especificada la URL, se debe publicar el documento que contiene al vocabulario. Este procedimiento es parecido al de publicar un sitio web.
- Finalmente, ya se puede utilizar el vocabulario en un documento HTML con métodos de declaración de prefijos.

JSON (*JavaScript Object Notation* , Notación de Objetos de JavaScript)

JSON es un formato de texto ligero, ideal para el intercambio de datos ya que es independiente del lenguaje de programación. Se caracteriza por su facilidad de lectura y escritura, como también su fácil interpretación y generación.

JSON está constituído por dos estructuras:

- Colección de pares de nombre/valor. Ej.:"nombre":valor
- Lista ordenada de valores. Ej.: {"nombre1":valor1, "nombre2":valor2}

Ejemplo: {"LecturaAmbiente":{ "valor_hum":1,"valor_temp":28.6}}

JSON-Schema

El esquema Json es un documento similar a un esquema XML o DTD, que permite establecer tipo de atributos, atributos obligatorios, nulos, etc. en un documento json. Es decir, define los datos requeridos para una aplicación dada y la forma de interactuar con él.

Ejemplo de un esquema Json:

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "LecturaAmbiente Schema",
   "type": "object",
   "required": [
      "LecturaAmbiente"
   ],
   "properties": {
      "LecturaAmbiente": {
         "type": "string",
         "$ref": "#/datos"
      }
   },
   "datos": {
      "description": "Info de temperatura y humedad",
      "type": "object",
      "required": [
         "valor_temp"
      ],
      "properties": {
         "valor_temp": {
            "type": "number"
         }
      },
      "additionalProperties": false
   }
}
```

OWL (W*eb Ontology Language*, Lenguaje de Ontologías Web)

Owl[12] es el lenguaje de marcado estándar de la Web Semántica para expresar ontologías, proporciona un modelo construido sobre RDF y codificado en XML. A través de las ontologías se pretende una comprensión común de conceptos, grupo de conceptos y relaciones entre ellos.

OWL tiene más facilidades para expresar el significado y la semántica que XML, RDF y RDF-S, por su capacidad para representar contenido interpretable por máquinas en la Web.

OWL es una revisión del lenguaje de ontología web DAML + OIL que incorpora lecciones aprendidas del diseño y su aplicación de DAML + OIL.

[12] https://www.w3.org/TR/2004/REC-owl-features-20040210/

Existen tres sublenguajes de OWL:

- **OWL Lite**: proporciona una jerarquía de clasificación y restricciones simples. Admite restricciones de cardinalidad, pero solo permite cardinalidad de 0 y 1. Tiene menor complejidad que OWL DL, por lo que proporciona una rápida migración para tesauros y otras taxonomías.
- **OWL DL** (Lógica de descripción): máxima expresividad sin perder la capacidad de respuesta computacional, incluye todas las construcciones del lenguaje, sin embargo, se pueden usar bajo ciertas restricciones.
- **OWL Full:** permite máxima expresividad y libertad sintáctica de RDF, sin garantías computacionales, OWL Full permite aumentar a una ontología el significado del vocabulario predefinido (W3C, 2009).

Existen dos metodologías para el proceso de desarrollo de las tecnologías semánticas:

Procesado manual: Consiste en la descripción de contenidos y ontologías por parte de personas, esto con lleva altos recursos de tiempo y costo, pero obteniendo resultados de gran calidad.

Procesado semiautomático: Mediante el uso de herramientas automáticas que generen descripciones de contenido y ontologías en altos volúmenes, con menor costo y tiempo, pero en contraparte obteniendo resultados que tal vez no satisfagan exigencias mínimas del estándar W3C.

La estructuración de los recursos en la web mediante XML y RDF junto con la aplicación de las especificaciones W3C, es uno de los desafíos que se tienen en la generación de ontologías y obtener un modelo de datos bien definidos en la web. De esta manera existen varias organizaciones dedicadas al estudio y diseño de entornos que a través de tareas automatizadas desarrollan software para agilizar este proceso ontológico, algunas de estas herramientas open-source y comerciales son:

- Hozo Ontology Editor[13]
- Protégé[14]
- Apache Jena[15]
- Top Braid Composer[16]

[13] http://www.hozo.jp/download_en.php

[14] https://protege.stanford.edu/

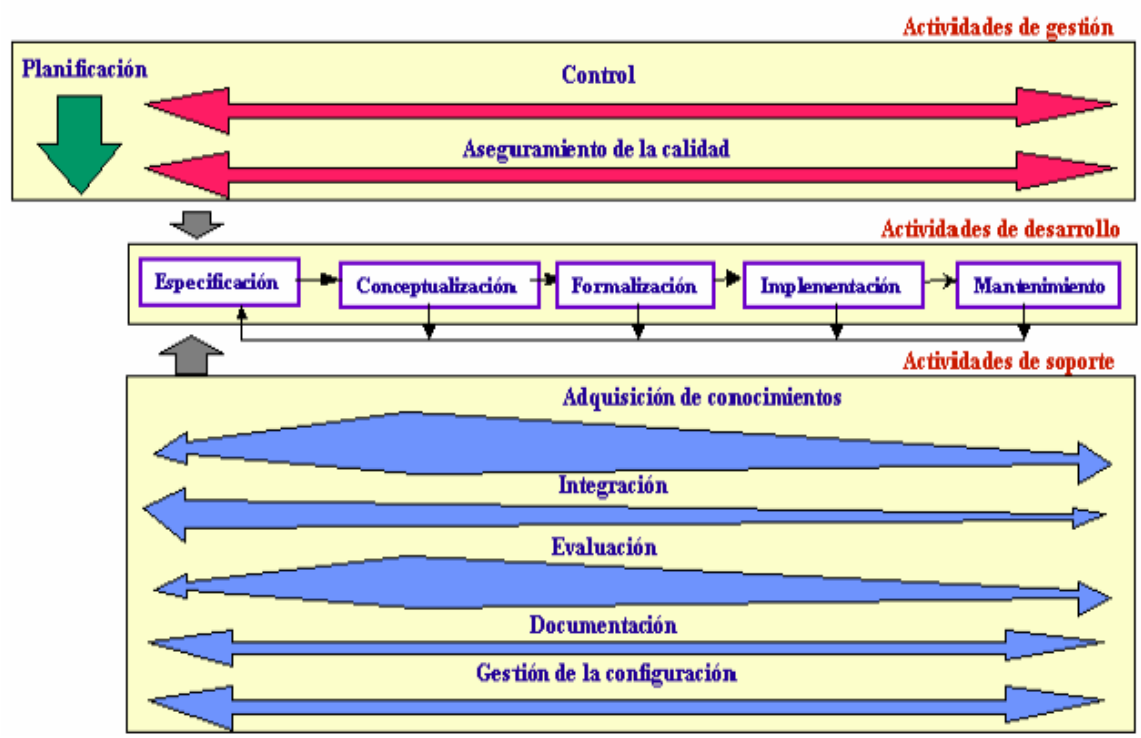
[15] https://jena.apache.org/

- Altova Semantic Works[17]

Existen varias herramientas para la creación de Ontologías: Methontology, Grüninger and Fox's y On-To-Knowledge, sin embargo la más usada es Methontology (Fernández-López, 1997) desarrollada por el grupo de Ingeniería Ontológica de la Universidad Politécnica de Madrid.

Methontology propone por cada prototipo iniciar con actividades de planificación, luego con actividades de administración, especificación y soporte. Apoyado en la adquisición de conocimiento se construye el modelo conceptual y luego las actividades de formalización e implementación.

Figura 2-14. Ciclo de vida de Methontology



Fuente: (Fernández-López, 1997)

Una vez especificado el protocolo el modelo conceptual es construido apoyado por la adquisición de conocimiento a través de ontologías, una vez realizada la conceptualización se desarrollan las actividades de formalización e implementación.

Methontology describe los componentes para el modelado de ontologías, los principales componentes para la contextualización son:

[16]https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/

[17] https://www.altova.com/

- **Conceptos**: Son entidades, objetos, personas.
  Ej.: En un dominio agricultura: plantas, agua, semilla, fertilizantes, clima, etc.
- **Relaciones:** Son la asociación entre conceptos. Ej.:

Concepto	Relación binaria	Concepto
Planta	Necesita de	agua

- **Instancias:** Es el valor asignado a un recurso.

Concepto	Instancia
Planta	manzanilla

- **Constantes**: Son valores que no cambian durante un largo tiempo.
  Ej.: El tiempo de cultivo de patatas es de 6 meses.
- **Atributos:** Describe propiedades.

Concepto	Atributos de instancia
Fertilizante	Peso

Concepto	Atributos de clase
Planta	Tipo: verdura, frutal.

- **Axiomas formales:** Son expresiones lógicas que restringen en una ontología.
  Ej.: Una planta no se le puede dar riego y fertilizar al mismo tiempo.
- **Reglas**
  Si en el ambiente la temperatura es superior a 20° se debe activar el riego.

SPARQL

SPARQL es un lenguaje de consulta para RDF, que está diseñado para cumplir con los requisitos identificados por el grupo de trabajo de acceso a datos RDF. En el contexto de la web semántica, se define "consulta" como tecnologías y protocolos que permiten recuperar información de la web de datos mediante programación. La web de datos o web semántica, típicamente representada por el formato de datos RDF, necesita un lenguaje de consultas como SPARQL que permite recibir resultados utilizando protocolos como HTTP o SOAP.

La mayoría de las formas de consulta SPARQL contienen un conjunto de patrones triples llamados patrón grafo básico.

La consulta consta de dos partes:

- La cláusula *select* identifica las variables del resultado de la consulta.

- La cláusula *where* que contiene un patrón gráfico básico para que coincide con el gráfico de datos. En el siguiente ejemplo consta de un solo patrón triple con la variable "Título".

```
        Datos:
<http://ejemplo.org/libro/libro1> <http://purl.org/dc/elementos/1.1/title> "Especificación SPARQL" .

        Pregunta:
SELECT? title
WHERE
        {
          <http://ejemplo.org/libro/libro1> <http://purl.org/dc/elementos/1.1/title>? title
        }
```

Esta consulta tiene como resultado:

Title
"Especificación SPARQL"

SPARQL se puede utilizar para expresar consultas de diferentes fuentes de datos, ya sea que los datos se almacenan de forma nativa como RDF o a través de middleware. Los resultados de las consultas pueden ser conjuntos de resultados o grafos RDF. Permite extraer información compleja de la web de datos en formatos complejos como pueden ser tablas, estas tablas se pueden incorporar a otra página web que utilice este mismo enfoque. SPARQL tiene herramientas que permiten construir motores de búsqueda que incluyen datos de la web semántica. Al igual que SQL, este lenguaje permite realizar consultas con uniones, intersecciones y otros parámetros.

W3C tiene especificaciones propias para el lenguaje de consultas SPARQL para RDF (W3C R. , 2013). Aquí se describe al lenguaje, terminología, variables, sintaxis, resultados gráficos, entre otros parámetros.

Datos enlazados (*Linked Data*)

Linked Data es el nombre que se le da al modelo con las mejores prácticas para publicar información y datos en la web usando tecnologías semánticas bien definidas. Tim Berners-Lee (2009) con la idea de utilizar aspectos de las tecnologías semánticas en la conexión de información virtual de dispositivos y datos de internet, definió "La Web de Datos".

Existen cuatro reglas que se definen en el *Linked Data*:

1. Usar URI para nombrar los objetos.
2. Usar URIs HTTP para encontrar esos nombres.
3. Usar estándares como SPARQL o RDF para proveer la información.

4. Usar enlaces entre URIs para realizar nuevos descubrimientos.

Siguiendo este modelo, la presencia de Red Abierta de Datos Enlazados (*Open Linked Data*) ha tenido un mayor crecimiento permitiendo el uso público de la información dentro de la red. Esta iniciativa también tiene presencia en Google's Rich Snippet, Facebook's Open Graph o Schema.org (Rodríguez, 2015).

2.4.3. Web semántica de las Cosas

El problema de la heterogeneidad de datos en la interconexión de dispositivos en el IoT obtenidos a través de los sensores, es una de los aspectos que se mejoran mediante descripciones semánticas y modelos de conocimiento. Es a partir de este conocimiento que existen organizaciones dedicadas a aportar tecnologías ontológicas para que sea unificadas y utilizadas en el desarrollo de aplicaciones IoT.

W3C *Semantic Sensor Network Incubator Group*

Es un grupo creado por W3C que tiene como objetivos: desarrollar ontologías para describir sensores y redes de sensores para usarlo en aplicaciones web. Recomendar y estudiar métodos para usar la ontología para habilitar semánticamente las aplicaciones desarrolladas de acuerdo con estándares disponibles y los de habilitación de Web del Sensor (SWE) del OGC (Consorcio Geoespacial Abierto).

La ontología de sensores y redes de sensores SSN, describe un modelo independiente de dominio de extremo a extremo para detectar aplicaciones mediante un sensor centrado como *SensorML*, toma en cuenta subdominios del sensor, como las capacidades de detección que ayudará a caracterizar los datos detectados y a predecir comportamiento en entorno desconocidos.

Sensor ML es un lenguaje de modelo estándar para describir sistemas y procesos de sensores. Proporciona información necesaria para el descubrimiento de los sensores y ubicación de observaciones de sensores.

Red de Sensores Semánticos (*Semantic Sensor Network*, SSN)

Es una ontología desarrollada por W3C, provee de conceptos (dispositivo, plataforma, procesos, mediciones, características, etc.) que describen a los sensores y actuadores. Incluye una ontología ligera y autónoma llamada SOSA (*Sensor Observation Sample Actuator*). SSN y SOSA se pueden aplicar en gran variedad de casos, entre ellos, imágenes satelitales, monitoreos científicos, infraestructuras domésticas, industriales e internet de las cosas. (W3C O. , 2017)

Se cuenta además con otras ontologías desarrolladas para crear modelos de conocimiento en el entorno de IoT:

Ontología SPITFIRE

Extiende la funcionalidad de la ontología SSN incluyendo otros dispositivos que actúan en el IoT, añadiendo conceptos como actuadores, energía, actividad, contexto, entre otros.

Ontologies for the Internet of Things (Hachem, 2011)

Presenta una solución de middleware orientada a servicios empleando tecnologías semánticas para proporcionar interoperabilidad y flexibilidad. Centra su trabajo en el modelado de ontologías basadas en dominio, estimación y dispositivos. Las ontologías describen dispositivos, sus funcionalidades y modelan a fondo el dominio de la física. Ya que considera el dominio de la física como el núcleo de IoT, al permitir la aproximación y la estimación de las funcionalidades proporcionadas habitualmente por las cosas.

A Comprehensive Ontology for Knowledge Representation in the Internet of Things (Wang, 2012) Presenta el diseño de una descripción de la ontología para la representación del conocimiento en el dominio de IoT y discute la manera en que se puede utilizarlas para apoyar tareas como el descubrimiento de servicios, recursos, implementación, pruebas de servicios y composición dinámica.

A Unified Semantic Knowledge Base for IoT (Nambi, 2014)

En este trabajo, se presenta una base de conocimiento semántico unificado para IoT que usa ontologías como bloques de construcción. Presenta ontologías basadas en recursos, servicios, ubicación, información contextual y un conjunto de políticas para ejecutar servicios. La base de conocimientos permite la composición de servicios, el descubrimiento y el modelado para IoT en entornos dinámicos.

2.4.4. Casos de Uso de Tecnologías Semánticas en el IoT

De acuerdo al conocimiento obtenido de la literatura y análisis de las tecnologías y herramientas de la Web Semántica y la manera de cómo utilizar estas tecnologías en el IoT, se presentan las siguientes aplicaciones:

Uso de semántica para mejorar la gestión de la Información en aplicaciones IoT

Los datos generados por los sensores en IoT es información cruda que carece de conocimiento, es por eso que mediante el uso de tecnologías semánticas se puede dotar de conocimiento a la información para que pueda ser mejor utilizada.

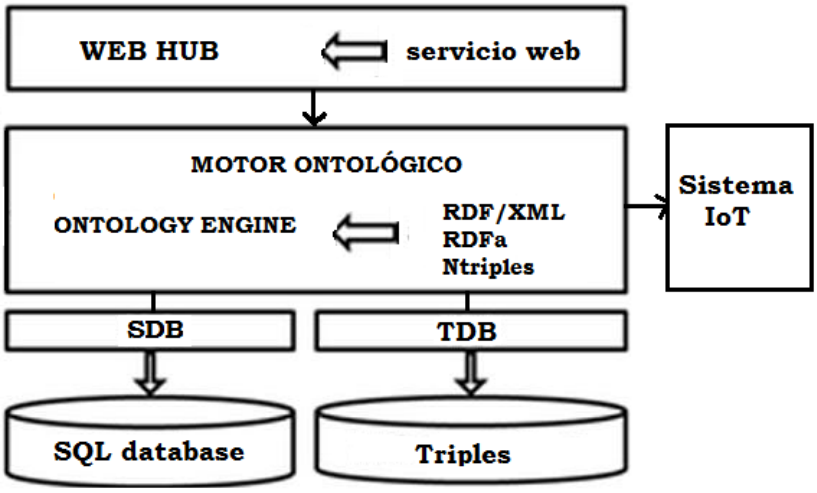
Mediante la información que se obtiene de las aplicaciones IoT, a través de técnicas y herramientas de la web semántica, entre ellas las ontologías, se representa dicha información de tal manera que los usuarios y quienes necesiten monitorizar el sistema tengan una mejor gestión de dicha información.

De esta manera, se crea ontologías para representar la información en un dominio específico, como puede ser agricultura, ambiente, turismo, movilidad, etc. y los usuarios pueden acceder a esa información de una manera semántica a través de un monitoreo ontológico.

En las ontologías se definen parámetros como: clases, relaciones, funciones y constantes para un determinado dominio.

En el siguiente diagrama de contexto se plantea como punto central un motor ontológico para que dote de conocimiento a la información obtenida, está constituido por la Base de Datos que contiene las "triples" que permiten estructurar la información proveniente de la base de datos que almacena la información obtenida del sistema IoT, mediante estructura ontológica orientada a algún dominio empleando modelos en formato RDF/XML, OWL o JSON. La capa Web hub corresponde a un servicio web que son empleados por aplicaciones y para mostrar la información a los usuarios.

Figura 2-15. Diagrama de contexto de la gestión de contenido de IoT



Fuente: propia

Es viable el uso de la semántica específicamente ontologías y vocabularios RDF estándar para el intercambio de información mediante la interoperabilidad de aplicaciones.

Uso de la semántica para desarrollar Método de Indexación en el IoT

El trabajo de (Niño-Zambrano, 2013) presenta una arquitectura para creación de índices semánticos en la IoT, basados en la metodología de indexación semántica web (D. M. Pezo, 2013), utiliza un modelo semántico conceptual basado en cuatro fases: contextualización, creación de base de conocimiento, servicios, construcción de índice semántico y despliegue.

Esta arquitectura se centra en la creación de índices semánticos mediante la capa Motor de Indexación Semántica, aquí se llevan a cabo algunos procesos de la información obtenida de los objetos IoT a través del Middleware de la IoT.

Los procesos llevados a cabo en el Motor de Indexación Semántica son:

- Modelo de consultas: establecer el dominio de los objetos y la forma de manejar los información y objetos para indexar.
- Modelo de conocimiento: creación de ontologías, el modelo conceptual, cómo manejar estas ontologías.
- Algoritmo de similitud: para análisis de representación y concurrencia.
- Modelo de servicios: interfaces de servicios web semánticos de objetos y del índice.

La interfaz de servicios web de middleware permite el acceso a la base de datos de objetos, permitiendo a las aplicaciones acceder a estos datos.

Por otro lado, la capa de servicios de aplicación genera una interfaz de servicios web, permitiendo a los usuarios u otras aplicaciones utilizar el índice semántico, acceder a la información de objetos y realizar consultas empleando motores de búsqueda semánticos.

Uso de la semántica para crear Plataformas Semánticas para el IoT

El aumento de dispositivos en el IoT y la interacción entre ellos, hace que cada vez se tenga gran cantidad de datos, obteniéndose parámetros que indiquen algún aspecto de las actividades diarias a través de sensores. Para que estos datos: temperatura, calor, luz, humedad, entre otros, tengan mejor utilidad deberían estar acompañados de un campo adicional que indique su proveniencia o actividad a la que corresponde generando información útil, a esto se llama "nuevo conocimiento".

Arquitectura

El trabajo desarrollado por (Rodríguez, 2015) plantea una arquitectura de software con modelo distribuido, tomando en cuenta los requerimientos mediante 4 fases: servicios, servicios virtuales, ambientes virtuales y aplicaciones.

- Servicios: Generan la interfaz para que interactúen las clases y relaciones de los objetos. En capsulan los objetos del IoT.
- Servicios Virtuales: Realizan operaciones complementarias de la encapsulación.
- Ambientes Virtuales: Almacenan datos de contexto de los servicios. Encapsulan las ubicaciones físicas del IoT.
- Aplicaciones: Ofrecen servicios a los usuarios: monitoreo, consulta, gestión, etc.

Interfaces de Recursos

Se requiere de interfaces homogéneas para los objetos de IoT y exponer su funcionalidad a través de servicios. Entre los diferentes objetos se tiene: objetos inteligentes, simples o asistidos, sin embargo, a través de varias tecnologías permiten complementar las capacidades para los objetos que no las poseen y puedan acceder a un servicio web.

Existen mecanismos que permiten la comunicación entre los objetos que utilizan el modelo cliente-servidor como las interfaces REST, otros siguen el modelo publicador-suscriptor para recursos que requieren peticiones constantes como el modelo CoAP, AMQP, entre otras que se han descrito en la sección correspondiente a Protocolos para IoT.

Representaciones de los Recursos

Se definen los formatos de los documentos que representan a los recursos, en la web el intercambio de información se lo hace a través de formatos como HTML, XML, JSON o ATOM, sin embargo, para lograr una representación semántica se requiere de RDF y sus formatos de representación, entre ellos: N3, RDF/XML, Turtle, N-Triples, JSON-LD, RDFa. Según Linked Data (Tim Berners Lee, 2009) establece las características que se presentan en la tabla 2-4.

Tabla 2-4. Formatos de representación de datos RDF

Formato	Características
RDF/XML	Formato común. Ofrece mayor interoperabilidad semántica. buena expresividad e interoperabilidad sintáctica.
N3	Se codifica en UTF-8. Su gramática está formada por sentencias que pueden ser directivas o declaraciones. Requiere intervención humana.
Turtle	Es una representación textual de un grafo RDF. Requiere intervención humana.

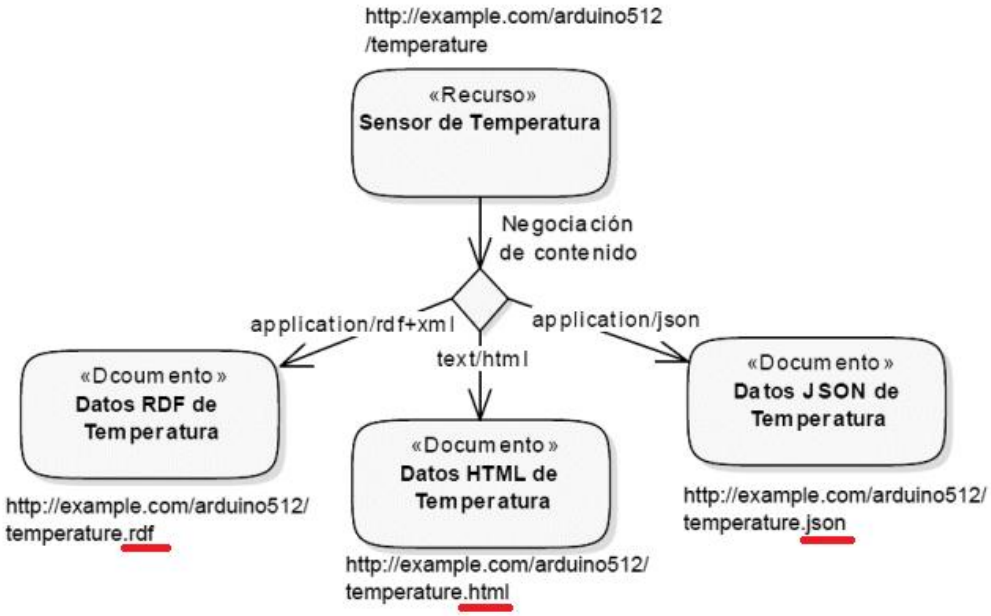
N-Triples	Desarrollado por W3C, formato de texto plano, que pone en código un grafo RDF, es subconjunto de N3.
RDFa	Es una especificación de atributos para expresar datos estructurados en HTML5, XHTML y cualquier aplicación XML.

Fuente: (LinkedData, 2015)

Es decisión de desarrollador elegir la más adecuada para su ambiente.

Según el trabajo realizado por Völkel, 2008, plantea la importancia para publicar contenidos semánticos de objetos mediante URIs, identificando por un lado al recurso y por otro lado al documento que describe al recurso. Plantea agregar la extensión a la dirección del archivo del documento, de esta manera se podrá distinguir semánticamente al dispositivo del documento que lo representa. Con el uso de herramientas ontológicas, el URI se genera de forma automática, seleccionando previamente el lenguaje que se quiere utilizar.

Figura 2-16. Identificación de documentos de recursos mediante URIs.



Fuente: (Rodríguez, 2015)

Los documentos RDF debe contener enlaces a la descripción semántica del servicio e incluir la información semántica de los dispositivos.

Mecanismo de descubrimiento de servicios.

Es necesario un mecanismo de descubrimiento de servicios en la red, anunciando a la red alguna información del servicio como: nombre, puerto y URL del servicio. El grupo *Zero*

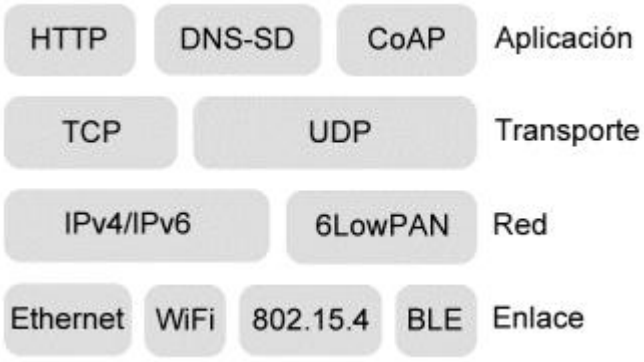
Configuration Networking del IETF plantea mecanismo de descubrimiento basados en el protocolo DNS-SD.

Cada servicio debe poseer una descripción semántica de sí mismo en una representación RDF mediante una dirección bien conocida. La información descrita debe ser mínima incluyendo datos como: nombre, plataforma, recursos y funciones, evitando un procesamiento innecesario en dispositivos de recursos limitados. Se pueden utilizar las ontologías descritas en SSN o SPITFIRE.

Interfaces a la red

Debe existir una conexión transparente entre dispositivos de extremo a extremo, en la figura se presenta la pila de protocolos que podrían utilizarse en un ambiente IoT.

Figura 2-17. Pila de protocolo en ambientes IoT



Fuente: propia

En la capa Enlace, los protocolos ampliamente usados que permiten la conexión con los medios físicos (dispositivos de recursos limitados) son ZigBee, 6LoWPAN/802.15.4, Z-Wave y BLE ya que estos presentan eficiente consumo de energía. De acuerdo a la aplicación y la compatibilidad que tenga con protocolos superiores se debería seleccionar el más idóneo.

Con respecto a la capa Red, es importante que los servicios accedan mediante IPV4/IPV6 a los dispositivos de la red, sin embargo, el más utilizado por los recursos de IoT es 6LoWPAN por lo que existen métodos como tunneling para enlazarlos.

En la capa Transporte predominan los protocolos TCP y UDP, cada uno con sus diferentes características útiles dependiendo de la aplicación y de los protocolos de capa superior que se utilicen, por ejemplo, HTTP trabaja bajo TCP y CoAP bajo UDP.

La capa Aplicación HTTP es el protocolo más común en la web, sin embargo, existen otros protocolos creados para aplicaciones IoT entre ellos están MQTT, CoAP entre otros.

Motores de búsqueda

Mediante el uso de descriptores semánticos se puede mejorar la búsqueda de etiquetas claves en los documentos HTML o XML que describen a los recursos. El uso de ontologías permite realizar inferencias (relaciones) sobre un servicio que puede estar descrito con diferentes términos o lenguajes. De esta manera es importante que los datos estén relacionados mediante ontologías o esquemas RDFS.

Todas las representaciones RDF de los componentes deben estar contenidas en un servicio común, que represente una interfaz de consulta a través de un servicio virtual o servicio de acceso, este trabajo se lo puede realizar a través de SPARQL.

Casos de Estudio de Plataformas Semánticas

Existen varias plataformas que hacen uso de tecnologías semánticas, algunas de ellas son:

Plataforma semántica para el IoT, desarrollada por (Rodríguez, 2015), realiza un estudio en el marco de IoT y tecnologías semánticas, propone una plataforma de IoT en la que con el uso de servicios virtuales almacena descripciones semánticas se sensores y actuadores, lo cual habilita interactividad de dispositivos, a través de un caso de estudio analiza la interoperabilidad entre los dispositivos en un ambiente IoT.

De la misma manera el trabajo de (Pfisterer, 2011) con la Plataforma SPITFIRE en la que emplea el concepto de Web Semántica de las Cosas mediante abstracciones de los objetos y con el uso de los principios de Linked Data. Utiliza reglas semánticas para la interoperabilidad permitiendo que los datos de sensores sean accesibles para las aplicaciones a través de mecanismos automáticos. Demuestra la viabilidad de su proyecto con un caso de uso.

La plataforma de sensores enlazados Sense2Web (Barnaghi, 2010) proporciona interfaces gráficas para el ingreso de datos de IoT utilizando conceptos de ontologías de dominio de sitios. Publica los datos utilizando RDF a través de un punto final SPARQL. Implementa además interfaces RESTful para publicación directa, acceso y consulta de datos vinculados IoT.

Como caso de uso, el proyecto SEMIOTIC da solución al problema de gestionar de forma eficiente la Energía a partir de la integración semántica de datos generados por nodos que contienen sensores en un ambiente IoT. Este proyecto se basa en el procesamiento de datos a través de la representación semántica de los mismos y la relación de estos en la

gestión energética de redes inteligentes distribuidas, integrando tecnologías de IoT, semántica y métodos estadísticos.

Existe una plataforma comercial denominada Sofía2, es única plataforma semántica en el mercado, permite subir información para disposición de sistemas y aplicaciones IoT. Permite realizar un modelado mediante ontologías, aplicar reglas, realiza procesamiento en tiempo real mediante interconexión de dispositivos utilizando diferentes lenguajes y protocolos, facilitando la interoperabilidad entre diferentes nodos.

Dispone de una consola centralizada para gestión, utiliza base de datos relacionales como mongo db, Utiliza protocolos de comunicación ligeros como REST, WebSockets, MQTT, WS, JMS, AMQP.

3. Planteamiento del Proyecto de Investigación

En este capítulo se plantean los objetivos generales y específicos a cumplir en el desarrollo de esta investigación. Además, se describe la metodología a seguir para el cumplimiento de los objetivos propuestos.

3.1. Objetivo General

El objetivo general de este TFM es el estudio comparativo de las diferentes tecnologías y herramientas de la web incorporadas al Internet de las cosas, así como también los estándares para la Web semántica y ontologías. Este estudio servirá de base de conocimiento en el desarrollo de aplicaciones IoT que permitan una comunicación apropiada en los procesos automáticos.

3.2. Objetivos Específicos

A continuación, se describen los objetivos específicos planteados para el desarrollo del TFM:

- Describir los protocolos o tecnologías para transferencia de datos que pueden ser utilizados en una aplicación IoT.
- Determinar y analizar características técnicas de los protocolos de la capa aplicación en un proceso de IoT.
- Proponer buenas prácticas para determinar el mejor protocolo para el desarrollo de aplicaciones IoT.
- Describir los estándares, herramientas y ontologías de la web semántica que puedan ser aplicables al IoT.
- Analizar las tecnologías de la web semántica y plantear el uso en el Internet de las Cosas.
- Implementar un prototipo que capture datos y puedan ser visualizados en la web empleando una plataforma IoT.
- Analizar los datos que intervienen en el ambiente IoT para la creación de ontologías correspondientes a los dispositivos.

3.3. Metodología de trabajo

Para el desarrollo de este Trabajo de Fin de Máster (TFM) se ha optado por una investigación cualitativa, ya que, al realizar un estudio comparativo de las diferentes tecnologías para envío de datos en el Internet de las Cosas, se requiere de un análisis de las características que permitan mejorar la comunicación entre objetos, es decir hablamos de un análisis de texto y otros aspectos que intervienen en este proceso.
De igual manera, para el planteamiento del uso de aspectos de la web semántica en el Internet de las cosas se aplicará procedimientos basados en una investigación cuantitativa y una interpretación sistemática.

A continuación, en la tabla 3-1, se presentan las tareas más relevantes desarrolladas en esta investigación.

Tabla 3-1. Cronograma de actividades

	Abril		Mayo				Junio				Julio			
Descripción	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Estado del arte														
Definición de objetivos específicos														
Definición de Metodología														
Desarrollo de la propuesta														
Recolección de información														
Organización de información														
Procesamiento														
Interpretación y análisis														
Análisis de resultados														
Presentación de resultados y comprobación de hipótesis														
Escritura de la memoria														
Realizar la estructura de capítulos														
Elaboración de Introducción														
Elaboración de capítulos														
Resultados y conclusiones														

Fuente: propia

Estado del arte

Se investigan los trabajos relacionados y se realiza el marco teórico que contribuya al desarrollo de este TFM, enfocándose en protocolos de capa aplicación que permiten el

envío de datos, estándares de la web semántica, ontologías y esquemas aplicables al Internet de las Cosas.

Definición de objetivos y metodología

En este apartado se marcan los objetivos propuestos tanto el general como los específicos que detallen las actividades a realizar para comprobar la hipótesis planteada.

Por otro lado, se plantea la metodología empleada para el desarrollo de esta investigación, el desarrollo temporal y la organización de este documento.

Desarrollo de la propuesta

El desarrollo de esta investigación se divide en dos partes, por un lado, se realiza un análisis de las tecnologías que permiten enviar datos en aplicaciones o servicios para el IoT, entre ellas MQTT, CoAP, WAMP, AMQP, HTTP y REST.

Se analizan atributos y aspectos de cada tecnología, entre ellos: recursos de cómputo, capa transporte que usa, aspectos de Calidad de Servicio (QoS), seguridad, arquitectura, compatibilidad con plataformas IoT.

Por otro lado, se estudia y se analizan aspectos de la web semántica tales como herramientas, estándares y ontologías aplicables al campo de IoT, se analizan estos aspectos empleando un prototipo que capture datos y sean visualizados en una plataforma IoT semántica Cloud y determinar su aplicabilidad en entornos de la vida cotidiana.

Experimentación y validación

A través de técnicas cualitativas se toman en cuenta aspectos, características funcionales, percepciones y relaciones para analizar los protocolos de capa aplicación de la pila IoT.

Con los fundamentos y el conocimiento obtenido del estado del arte con respecto a la Web semántica, se analiza aplicaciones de IoT en las que se han empleado tecnologías, y se implementa un prototipo que capture datos y se define la creación de ontologías en este ambiente.

Análisis de Resultados

Los resultados obtenidos de los estudios realizados nos permiten plantear Recomendaciones y Buenas Prácticas que puedan ser empleadas por los desarrolladores de software para elegir una determinada tecnología de acuerdo a la aplicación que se desee implementar en el ámbito de IoT.

Obteniendo los usos prácticos que se han dado a las tecnologías semánticas y la implementación del prototipo se determinan las ventajas de su adopción en el IoT y se establece el impacto que se tiene al incluir estos aspectos semánticos promoviendo el uso en el ámbito de Internet de las Cosas.

Este análisis en conjunto permite, además, obtener las conclusiones que validen la propuesta de este trabajo.

Escritura de la Memoria

Se elabora la memoria del trabajo, destacando todas las actividades realizadas y explicando todas las fases de la investigación. Finalmente se exponen las conclusiones de trabajo, planteando de forma clara para que sirva de base para los desarrolladores de aplicaciones en el ámbito de Internet de las Cosas y exponiendo los trabajos futuros que nacen de esta investigación.

4. Descripción de los Métodos de Investigación

En este capítulo se describen las técnicas y los métodos utilizados en el proceso de obtención de información, síntesis y análisis de la información relevante que ha servido de base para la obtención de las Recomendaciones y Mejores Prácticas para la elección del protocolo idóneo para IoT, como también los aspectos de la web semántica que puedan aplicarse al IoT.

Este trabajo de investigación consta de dos aspectos, el primero es el estudio comparativo de las tecnologías web y protocolos para IoT que permitan plantear recomendaciones de uso de protocolos adecuados para transferencia de información y el segundo el estudio de aspectos de la web semántica que puedan ser implementadas en IoT para mejorar la interoperabilidad entre dispositivos.

El desarrollo de este trabajo en general se basa en una investigación de tipo cualitativa, ya que se enfoca en el contexto, la definición y significado de aspectos que intervienen en el tema de estudio (Corbin, 2002).

Dentro de este campo de investigación, la teoría documentada es el método que se apega al planteamiento de este trabajo. Su desarrollo se centra en la recolección, análisis de datos y mediante la interpretación se pretende la teorización de algún fenómeno (Mesias, 2010). Las técnicas empleadas son las técnicas documentales, textuales, análisis del contenido y puntos de vista inmersos.

La primera parte de este TFM, sigue un proceso esquematizado en cuatro etapas: descripción, síntesis, análisis y resultados.

Descripción. En esta etapa se recolecta la información más relevante de IoT (hardware, software, plataformas en la nube) , características fundamentales de las tecnologías web y protocolos de la web actual que se usan en IoT y los protocolos específicos que han sido creados para el IoT. Se utiliza ayudas metodológicas como: fuentes bibliográficas, artículos científicos, especificaciones y estándares.

Síntesis. A partir de la información obtenida, se ha sintetizado y conceptualizado la información más relevante para proceder con la siguiente fase.

Análisis. En esta fase se basa en el método "Análisis comparativo constante", ya que pemite categorizar y estimar comparativamente la información recopilada para proporcionar

un amplio sentido de visión. A través de este análisis se establecen los criterios a considerar, destacando funcionalidades de cada protocolo como: características, arquitectura, el protocolo transporte que usa, codificación, Web nativa, si tiene estandarización o recomendación, seguridad, calidad de servicio (QoS) y la compatibilidad con plataformas IoT.

Con esto se obtendrá parámetros que determinen alcanzar una óptima funcionalidad en el sistema interoperable de objetos.

Resultado. El análisis de la información permite realizar un estudio comparativo de los protocolos y determinar recomendaciones de uso de los mismos en aplicaciones con dispositivos con capacidades restringidas, es decir mínimas cantidades de memoria, consumo de batería, redes de bajo ancho de banda y alta latencia. De esta forma, se obtienen resultados que permitan obtener las "buenas prácticas" a seguir en el momento de la selección de un protocolo para IoT, este proceso se lo esquematiza en la figura 4-1.

Figura 4-1. Proceso de desarrollo del trabajo - Protocolos IoT

Fuente: Propia

Por otro lado, continuando con la segunda parte de este TFM, siguiendo el método de teoría fundamentada empleando técnicas y procedimientos documentales y de texto se sigue un proceso de cuatro etapas:

Descripción. Se describen las características de tecnologías, recomendaciones y herramientas de la Web Semántica a través del estado del arte.

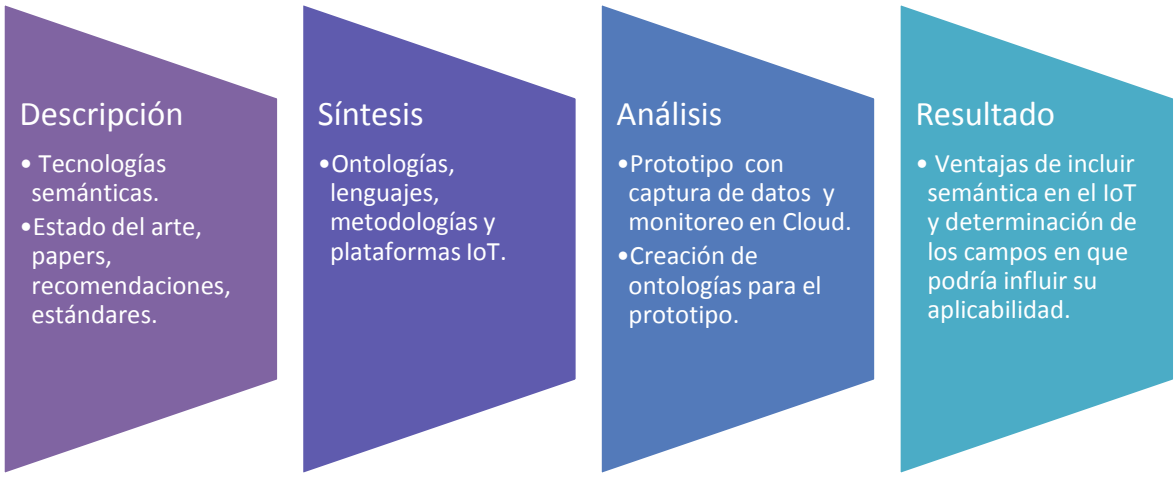
Síntesis. La información y conocimiento obtenido de la Web Semántica que ha sido recopilada, ahora se la categoriza dando prioridad a los conceptos aplicables a IoT. De esta

manera se determinan los aspectos de la Web Semántica que se aplican al IoT, entre ellos Ontologías, Metodologías, Estándares, etc.

Análisis. En esta fase, con el conocimiento obtenido en las etapas anteriores, se determina aspectos de software y hardware para desarrollar un prototipo IoT con el uso de sensores y plataformas electrónicas abiertas. Se analiza su funcionalidad en una plataforma en la Nube. Se crean las ontologías correspondientes a los dispositivos del prototipo, plasmando en un ambiente de prueba la creación de las mismas.

Resultados: Toda la literatura obtenida en el desarrollo de este trabajo permite plantear la importancia del uso de aspectos de la semántica en ambientes IoT y plantear los casos de uso de integración de IoT con aspectos semánticos.

Figura 4-2. Proceso de desarrollo del trabajo - Tecnologías Semánticas en el IoT



Fuente: propia

Tal como lo manifiesta (Corbin, 2002), para construir una teoría bien desarrollada, integrada y exhaustiva, un investigador debería elegir el método más conveniente, teniendo en mente que siempre es necesaria una interacción de métodos que lo complementen.

Tomando en cuenta el enfoque teórico que se le ha dado a esta investigación, durante este proceso se ha integrado aspectos cualitativos con ideas propias, que permitan usar la recolección de datos y el análisis para plasmar los conceptos que construyan la teoría. De esta manera, empleando las relaciones y conceptos obtenidos del análisis cualitativo, se puede usar esta información para establecer la manera de conseguir más información para el avance de la investigación.

Tal como lo manifiesta (Corbin, 2002) es importante señalar que independientemente del método que se utilice, la fuerza motriz siempre deberá ser la teoría en su constante evolución.

5. Análisis de Resultados

En este capítulo se plantean las características más relevantes de cada protocolo de comunicación en el IoT analizando los parámetros que influyen en el proceso de obtención de datos. Por otra parte, se diseña un prototipo de prueba con dos sensores que capturen datos y se visualicen a través de una plataforma IoT en la Nube, en el que se planteará las ontologías correspondientes a los recursos que se estén empleando.

5.1. Análisis de Protocolos para IoT

Una vez que se ha examinado los protocolos de Internet y las nuevas tecnologías que apoyan las iniciativas de IoT (véase Capítulo 2.2), analizamos sus características determinando aspectos importantes para el IoT.

Es importante señalar que no se puede determinar cuál es el mejor protocolo para el envío de mensajes, ya que el ambiente de IoT es tan diverso que dependiendo al campo en que se vaya a utilizar el protocolo, resultará beneficioso en unos casos y en otros no.

Para elegir un protocolo de comunicación, se debe tomar en cuenta ciertas características que influyen en el rendimiento de cada dispositivo conectado al IoT, aspectos como: duración de batería, memoria, ancho de banda, entre otros. Los protocolos creados específicamente para este tipo de aplicaciones son MQTT, AMQP, XMPP, CoAP y WAMP. Sin embargo, también prevalece el uso de la tecnología nativa de la Web como el protocolo WebSocktes, HTTP para ambientes por ejemplo en los que se requiera transmitir grandes cantidades de información.

En la tabla 5-1 se realiza un análisis comparativo de las tecnologías y protocolos para IoT, tomando en cuenta algunas características funcionales de los protocolos entre ellas: la principal característica, la arquitectura cliente/servidor o publicación/suscripción, el protocolo transporte que utiliza, el formato de los datos, web nativa, el organismo que lo estandariza, opciones de seguridad, calidad de servicio y las plataformas comerciales o abiertas que lo utilizan.

Tabla 5-1. Comparativa Tecnologías y Protocolos para IoT

	Características	**Arquitectura**	**Protocolo Transporte**	**Codificación**	**Web nativa**	**Estandarización/ recomendación**	**Seguridad**	**QoS**	**Compatibilidad con plataformas IoT**
MQTT	Ligero, simple y reducido consumo de energía.	PUB/SUB	TCP	Binaria, Json personalizado.	No	OASIS (estándar abierto)	SSL/TLS	Si	Sofía2, Microsoft Azure IoT, Fiware, IBM Watson, Artik, Ubidots, Google Cloud IoT, AWS IoT.
AMQP	Fiabilidad Interoperabilidad Transaccionalidad.	PUB/SUB	TCP	Binaria	No	OASIS (estándar abierto)	SSL/TLS, SASL	---	Microsoft Azure IoT, Sofía2.
XMPP	Envía mensajes en tiempo real.	PUB/SUB	TCP	XML	Si	IETF (estándar abierto)	Cifrado, autenticación, autorización.	---	XMPP-IoT
CoAP	Arq REST, acerca el modelo HTTP a dispositivos restringidos.	Cli/Serv	UDP	Binaria, Json	Si	IETF (estándar abierto)	DTLS Datagram Transport Layer Security	Si (Ack)	Fiware, Artik.
HTTP	Es muy acccesible. No define presentación de información.	Cli/Serv	TCP	Json, XML	Si	IETF (estándar abierto)	SSL	Si (Ack)	Sofía2, Fiware, Microsoft Azure, Ubidots, Google Cloud IoT, AWS IoT.
Websockets	Comunicación bidireccional independientemente de la plataforma.	Cli/Serv	TCP	Binaria, texto, Json.	Si	IETF (estándar abierto)	TSL/SSL	----	Sofía2, Azure IoT, AWS IoT, Artik.
REST	Estilo de arquitectura de software, ampliamente utilizado para aplicaciones web.	REQ/RES	TCP	XML, Json	Si	Recomendación no oficial. Está basado en estándares HTTP, URL, tipos MIME.	TSL/SSL	---	Sofía2, Microsoft Azure IoT, Artik.

5.2. Análisis de Web semántica en IoT

Para realizar el análisis de la integración de aspectos de la Web Semántica en el IoT, se plantea un escenario de prueba, con el objetivo de tomar datos del entorno empleando sensores de humedad, temperatura y luminosidad.

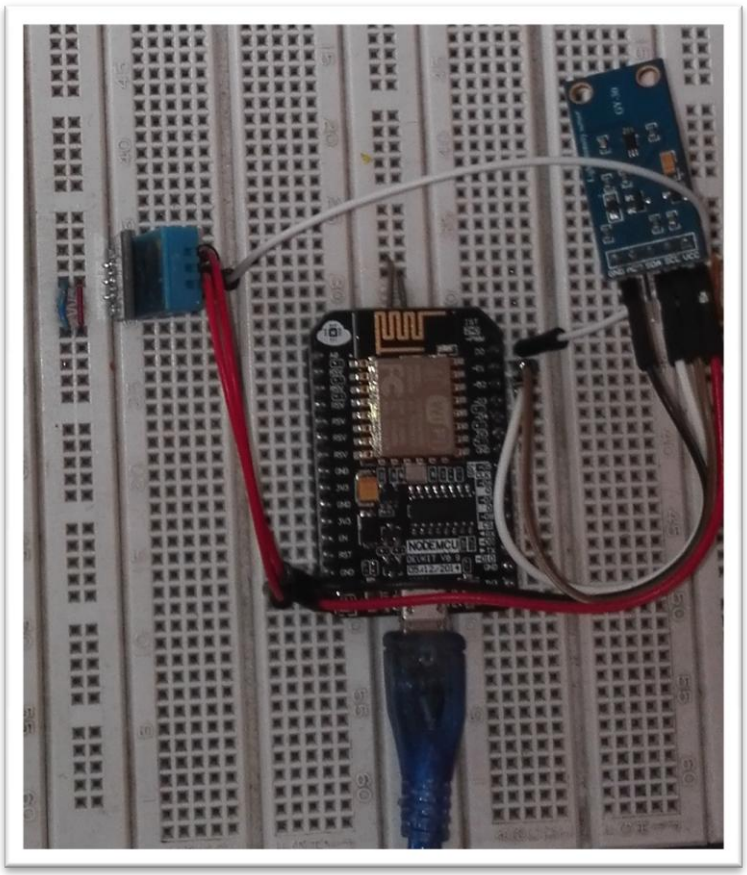
5.2.1. Análisis de Monitoreo en Plataforma IoT

En este caso se utiliza la Plataforma Ubidots para poder monitorizar y controlar el ambiente descrito. Mediante reglas se programa alertas que permitan una mejor gestión de los datos.

Esquema del circuito

En la figura 5-1, se observa los elementos empleados en el prototipo.

Figura 5-1. Circuito del prototipo

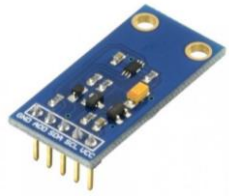


Fuente: Propia

Componentes hardware utilizados

A continuación en la tabla 5-2 se describe los elementos que intervienen en el prototipo.

Tabla 5-2. Descripción de componentes hardware

Elemento	Nombre	Descripción
	Placa de desarrollo NodeMCU	• Conversor Serie-USB para poder programar y alimentar a través del USB. • Pines de alimentación para sensores. • LEDs para indicar estado. • Botón de reset. • Contiene un Chip ESP8266.
	Chip ESP 8266	• Módulo WiFi integrado 2.4 GHz. • Contiene un microprocesador (Tensilica L106) de 32-bit. • RAM de 50 kB • 1 pin analógico A0 • 13 pines digitales numerados del D0 al D12. • 4 pines de 3 -5V. • 1 pin de 5V. • 4 pines de tierra GND.
	Sensor GY-30	• Sensor digital de intensidad de luz del ambiente. • Rango de medición 0-65535 Lx. • Con interfaz IC2.
	Sensor DTH-11	• Sensor digital de temperatura y humedad. • Rango de medición de temperatura 0 a 50°C. • Rango de medición de humedad 20% a 90% RH.

Entre los componentes hardware el principal protagonista es la placa NodeMCU con un chip ESP 8266, contiene un módulo wifi integrado que permitirá establecer una conexión inalámbrica con la plataforma en la nube. La programación es semejante a la de Arduino que es el más popular del mercado por su amplia comunidad y entorno de desarrollo propio, incluso se ha utilizado su IDE de desarrollo. En cuanto a costos, el de ESP8266 es mucho menor que Arduino, estas características han hecho elegir este dispositivo.

Componentes Software utilizados:

- IDE desarrollo Arduino
- Plataforma IoT Ubidots

El IDE de desarrollo Arduino tiene total compatibilidad con la placa NodeMCU y ha permitido el desarrollo de la aplicación.

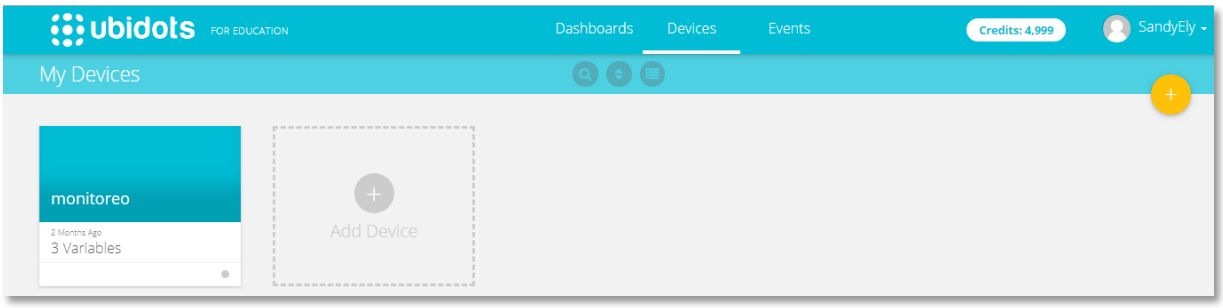
Se ha elegido la plataforma Ubidots, cuenta con dos ambientes de desarrollo: educación e industrial. En este caso se emplea la consola para educación, ideal para fines de investigación académica, que permite registrar los dispositivos, visualizar los datos y generar eventos ante alguna condición establecida. Permite realizar experimentación con prototipos de una manera muy amigable, pudiendo publicar datos sobre una API REST y una autenticación mediante tokens. Muchas de estas características técnicas y prestaciones son ofrecidas también por otras plataformas, que son de pago y cuentan con cortas versiones de prueba, sin embargo con Ubidots se puede crear aplicaciones pilotos con las funcionalidades concisas de manera muy rápida y fácil que lo hizo sea elegible para esta demostración.

Configuración Plataforma IoT Ubidots

Para trabajar en esta demostración, se ha elegido la Plataforma Ubidots para educación, por lo que para su uso únicamente requiere un login con email y contraseña, sin registrar datos bancarios que impliquen algún cobro.

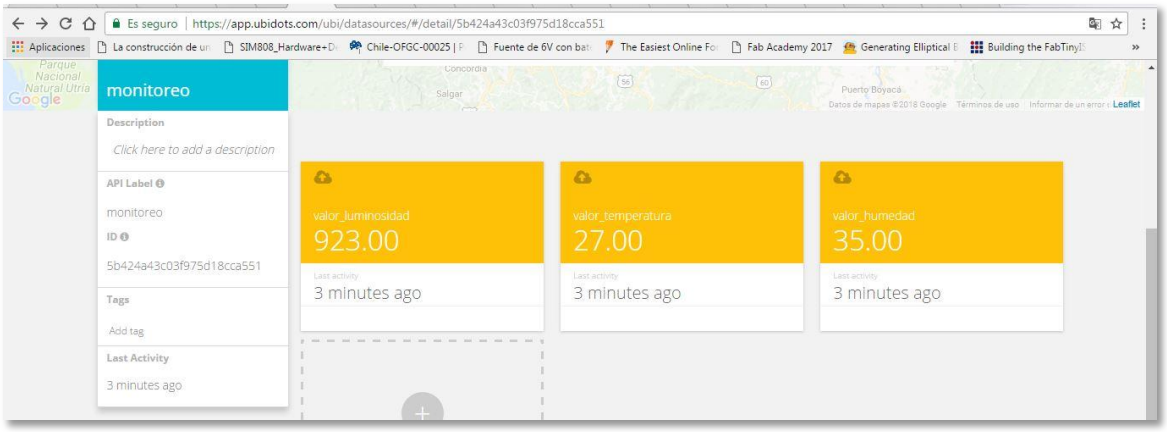
Una vez que contamos con la parte hardware del prototipo, ahora procedemos con la parte software. Inicialmente, en la pestaña "Devices" se ha identificado al nodo de sensores, en este caso asignando el nombre: "monitoreo", junto con la creación de este dominio, la plataforma genera un Id o token para identificarlo al momento de la programación, así se observa en la figura 5-2.

Figura 5-2. Creación del nodo



Al dominio monitorizado se agregan los tres recursos o sensores con los que se va trabajar para obtener los datos, en este caso se ha asignado los nombres: *valor_temperatura, valor_luminosidad y valor_humedad*, ver figura 5-3.

Figura 5-3. Creación de recursos en Ubidot



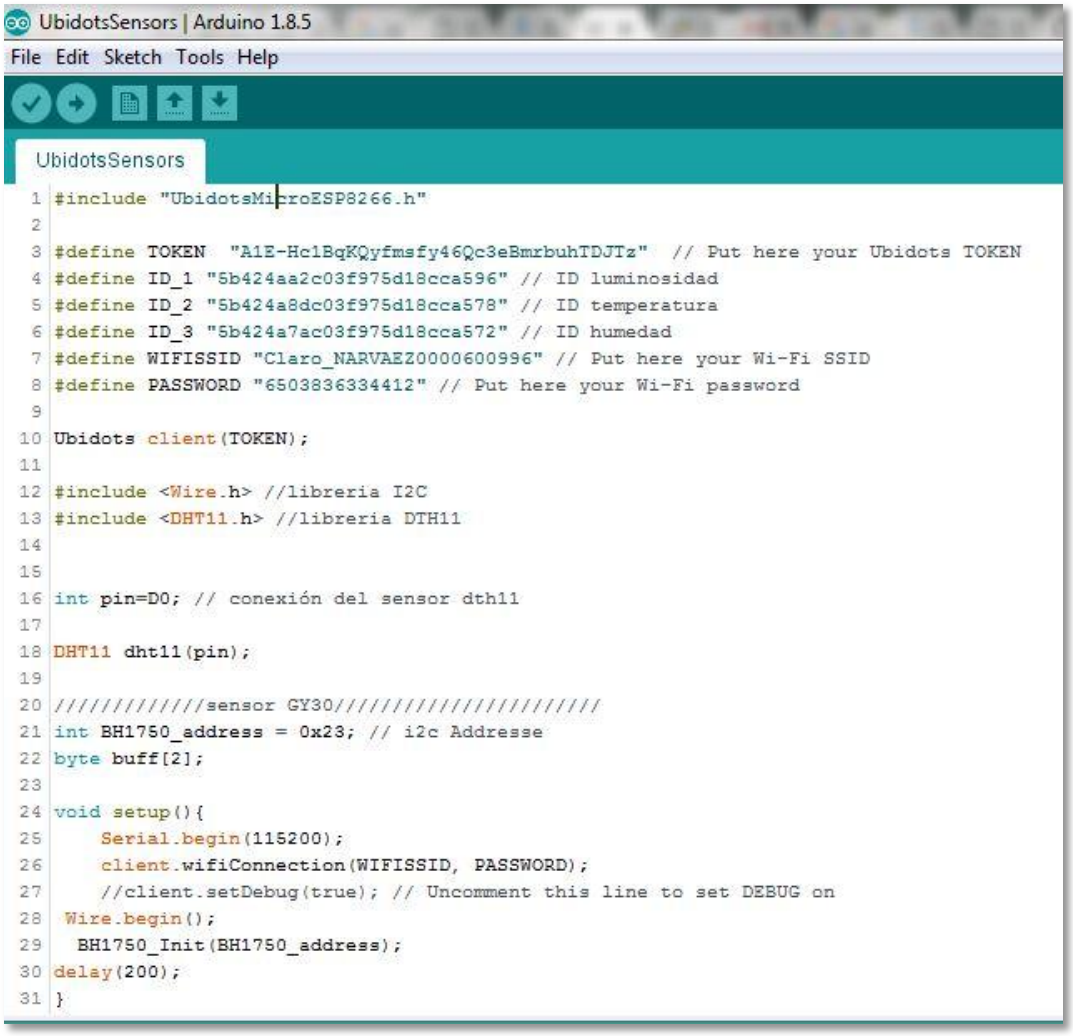
Ubidots crea los Ids para cada sensor. Estos identificadores son necesarios para la programación de la placa.

Programación del microprocesador

Se utiliza el IDE de desarrollo Arduino para programar al nodo implementado por el NodeMCU y los sensores de temperatura, humedad y luminosidad.

Para que el IDE reconozca al CHIP, fue necesario agregar la librería "*UbidostMicroESP8266.h"* que corresponde al microprocesador ESP8266. Se ha definido algunas variables, entre ellas el "*Token*" del nodo y los identificadores asignados a los sensores por la Plataforma IoT Ubidots. Para habilitar la conexión con la plataforma Ubidots se requiere el *SSID* de la conexión inalámbrica que en ese momento se está usando y su *password*, así se muestra en la figura 5-4.

Figura 5-4. Programación del MCU ESP8266



Código fuente del microprocesador:

```
#include "UbidotsMicroESP8266.h"

#define TOKEN  "A1E-Hc1BqKQyfmsfy46Qc3eBmrbuhTDJTz"  // Ubidots TOKEN
#define ID_1 "5b424aa2c03f975d18cca596" // ID luminosidad
#define ID_2 "5b424a8dc03f975d18cca578" // ID temperatura
#define ID_3 "5b424a7ac03f975d18cca572" // ID humedad
#define WIFISSID "Claro_NARVAEZ0000600996" // Wi-Fi SSID
#define PASSWORD "6503836334412" // Wi-Fi password

Ubidots client(TOKEN);

#include <Wire.h> //libreria I2C
#include <DHT11.h> //libreria DTH11


int pin=D0; // conexión del sensor dth11

DHT11 dht11(pin);

//////////////sensor GY30//////////////////////
```

```
int BH1750_address = 0x23; // i2c Addresse
byte buff[2];

void setup(){
    Serial.begin(115200);
    client.wifiConnection(WIFISSID, PASSWORD);
    //client.setDebug(true); // Uncomment this line to set DEBUG on
 Wire.begin();
  BH1750_Init(BH1750_address);
delay(200);
}

void loop(){

//////////sensor gy-30///////////////////////

 float valf=0;

  if(BH1750_Read(BH1750_address)==2){

    valf=((buff[0]<<8)|buff[1])/1.2;

    if(valf<0)Serial.print("> 65535");
    else client.add(ID_1, (int)valf,DEC);
  }

////////////sensorDTH11/////////////////////////
  int err;
  float temp, humi;
  if((err=dht11.read(humi, temp))==0)
  {
     client.add(ID_3, humi);
      client.add(ID_2, temp);
  }
////////////////////////////////////////////////

/////////////////Retardo de envio de datos//////////
  delay(DHT11_RETRY_DELAY); //delay for reread
    client.sendAll(false);
}

///////////Metodos correspondientes al protocolo I2C////////
void BH1750_Init(int address){

  Wire.beginTransmission(address);
  Wire.write(0x10); // 1 [lux] aufloesung
  Wire.endTransmission();
}

byte BH1750_Read(int address){

  byte i=0;
  Wire.beginTransmission(address);
  Wire.requestFrom(address, 2);
  while(Wire.available()){
    buff[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();
  return i;
}
```

Una vez depurado el código, se observa en la figura 5-5 la estructura de datos que se está enviando, se envían en formato Json la cadena de datos conformada por: el Id del recurso y el valor capturado en ese momento, para los tres sensores.

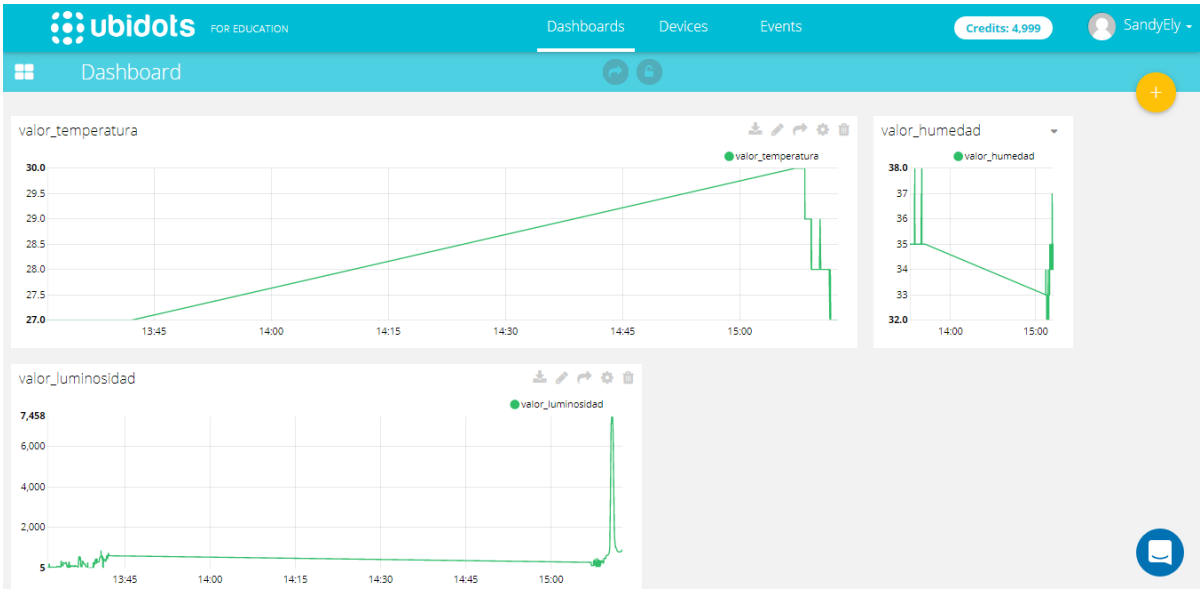
Figura 5-5. Depuración del programa

```
osting your variables:
OST /api/v1.6/collections/values/?force=true HTTP/1.1
ost: things.ubidots.com
ser-Agent:UbidotsESP8266/1.3
-Auth-Token: A1E-Hc1BqKQyfmsfy46Qc3eBmrbuhTDJTz
onnection: close
ontent-Type: application/json
ontent-Length: 178
{"variable": "5b424aa2c03f975d18cca596", "value":455.0000}, {"variable": "5b424a7ac03f975d18cca572"
 "value":35.0000}, {"variable": "5b424a8dc03f975d18cca578", "value":27.0000}]
TTP/1.1 200 OK
erver: nginx
ate: Sun, 08 Jul 2018 18:00:31 GMT
ontent-Type: application/json
ransfer-Encoding: chunked
onnection: close
ary: Accept-Encoding
ary: Cookie
llow: POST, OPTIONS

2
{"status_code": 201}, {"status_code": 201}, {"status_code": 201}]
```

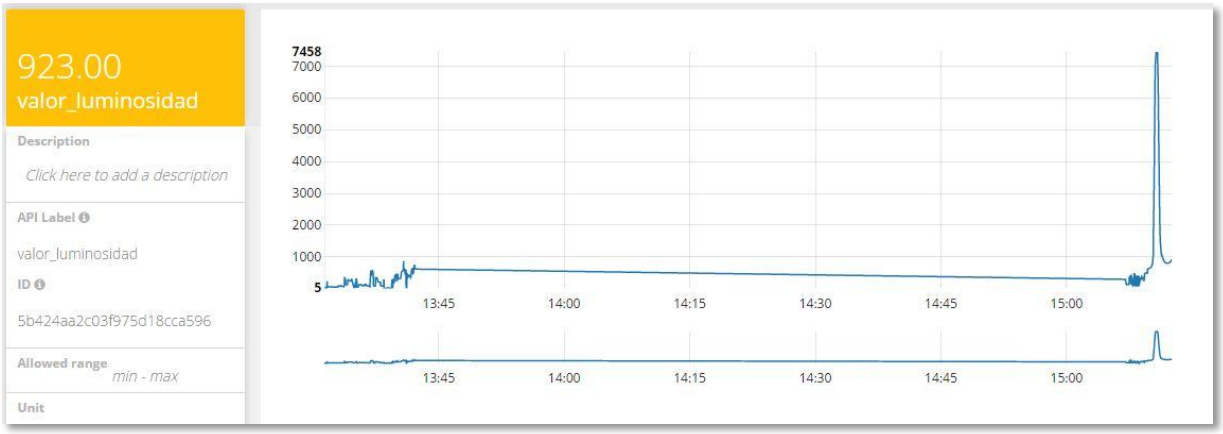
Una vez obtenidos los datos, la plataforma permite generar gráficos para visualizarlos, en la figura 5-6 se tiene un esquema de los tres sensores: temperatura, humedad y luminosidad.

Figura 5-6. Visualización de datos de sensores en la plataforma Ubidots



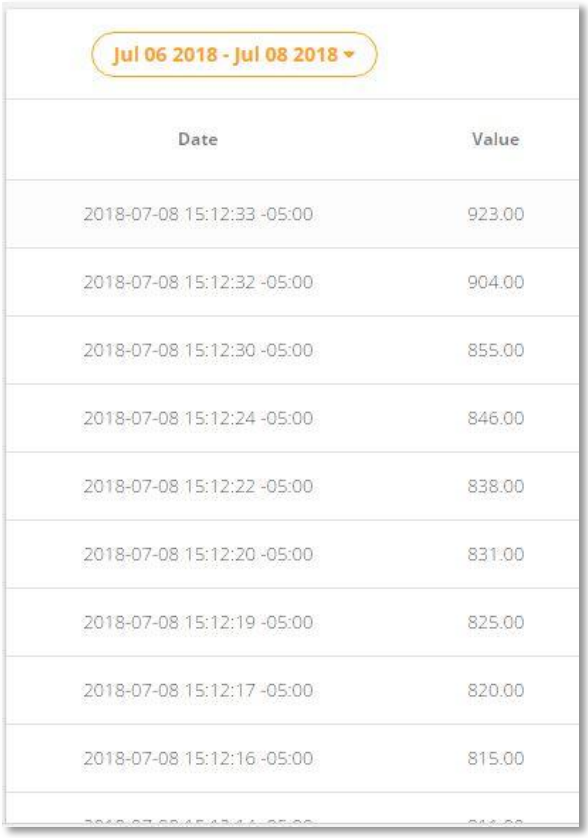
En la figura 5-7 se muestran los datos de luminosidad generados por el sensor.

Figura 5-7. Visualización de datos de un sensor en Ubidots



Además de visualización, se generan reportes y también alarmas a través de reglas que envían notificaciones como mensajes de texto o correo electrónico. En la figura 5-8 se observa los datos generados por el sensor a través de un reporte.

Figura 5-8. Reporte de datos de un sensor.



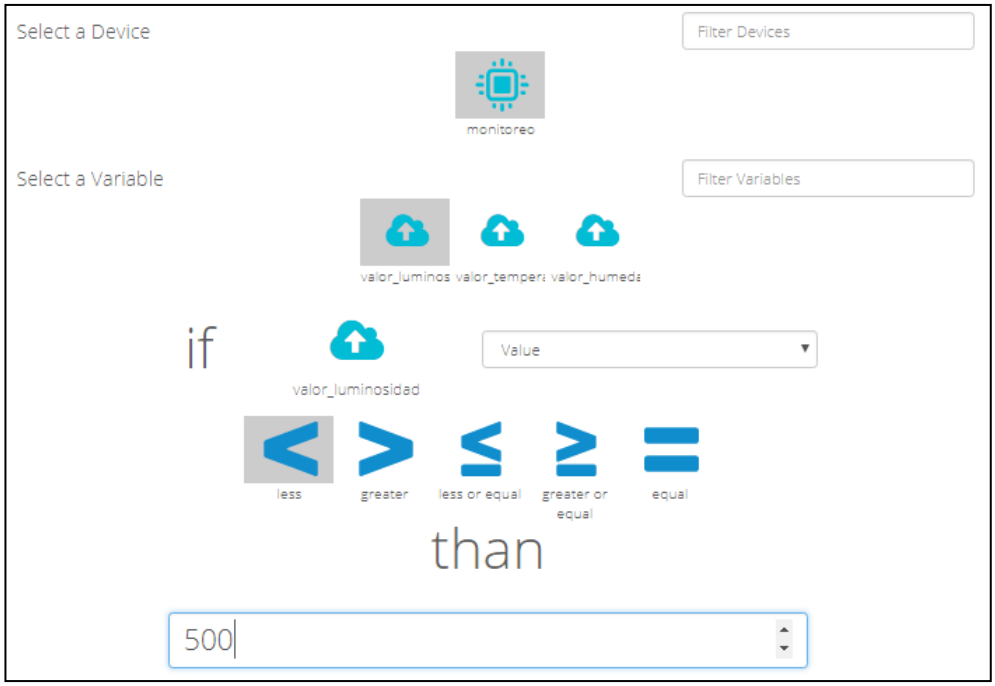
Configuración de eventos

Para contar con mayor control de los dispositivos, se configuran los eventos que permiten enviar mensajes a las personas encargadas informando algún comportamiento inusual de los dispositivos, como por ejemplo un rango muy alto de valores captados por el sensor que permitan activar un plan de emergencia.

La consola "Events" permite configurar estas alarmas, en este ejemplo se configura una alarma en la que si el valor de lumnosidad es menor a 500 Lux. se envíe por correo electrónico una notificación indicando el suceso.

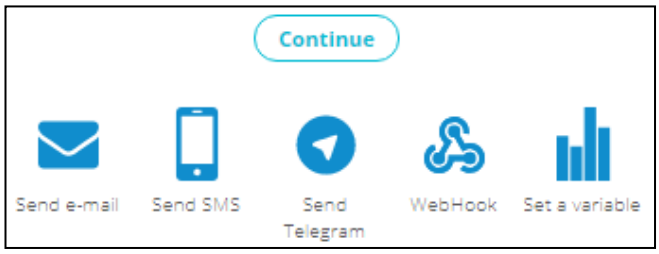
Inicialmente se selecciona el nodo, en este caso corresponde a *monitoreo*, se selecciona la variable *valor_luminosidad*, el condicionante que corresponde a menor que "<", se establece el valor, observe la figura 5-9 .

Figura 5-9. Configuración de alarma



A continuación, se elige el medio o tipo de mensaje que se va a enviar, para el ejemplo: *send e-mail,* observe la figura 5-10.

Figura 5-10. Selección de medio de envío de alerta.

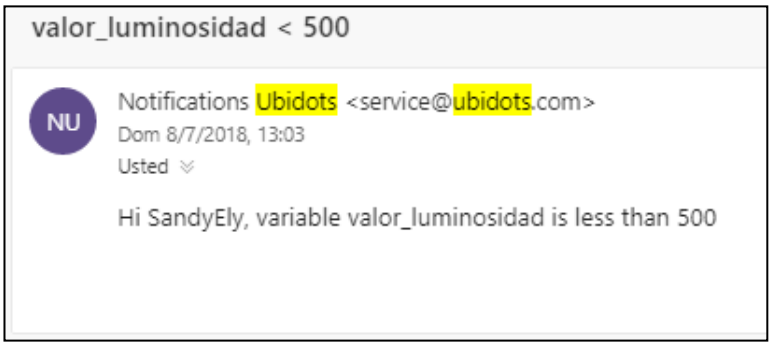


Se configura los parámetros del correo, entre ellos, e-mail, título y mensaje, observe la figura 5-11.

Figura 5-11. Configuración de parámetro de e-mail.

Evitando que el sensor capte lumninosidad del ambiente tapándolo con la palma de la mano, se logra que el sensor capte un valor inferior de luminosidad, por lo que la plataforma al recibir este valor genera la alarma, indicando por correo electrónico este suceso, ver figura 5-12.

Figura 5-12. Alarma enviada por la plataforma.



Es muy valioso el aporte de las Plataformas en la nube para IoT, ya que a través de una interfaz amigable permiten configurar y enlazar dispositivos para tener un control y monitorización de un ambiente a través del Internet.

Existen varias soluciones en cloud, su elección dependerá de las caraceterísticas que cada desarrollador esté buscando para determinada aplicación. Para ampliar el tema, el trabajo de (Sarango, 2017) realiza una comparativa de varias soluciones IoT en la nube.

Análisis de aspectos semánticos

A continuación, se realiza una descripción sobre la creación de ontologías en el prototipo IoT desarrollado, aclarando de esta manera los conceptos semánticos presentados teóricamente y la relación con los dispositivos empleados.

Modelado de datos

El modelado de entidades que representan a los dispositivos, se lo realiza a través de ontologías, las funcionalidades de un ambiente IoT giran en torno a los datos generados por los sensores, dispositivos o aplicaciones, estos datos pueden ser de medidas, comandos o propiedades, que pueden ser enviadas/consultadas o generar alertas para que sean controladas.

Se requiere modelar cada tipo de datos utilizados por el dispositivo, en este caso el prototipo cuenta con dos sensores:

- Temperatura y humedad
- Luminosidad

Cada tipo de datos correspondiente a estos dispositivos, se debe modelar mediante una ontología en la que se define semánticamente el tipo de datos, sus atributos y restricciones para no admitir otras propiedades para este tipo de datos.

Ontología "SensorAmbiente"

Para el sensor DTH-11 de temperatura y humedad se crean las siguientes propiedades con la restriccion de no admitir ninguna propiedad adicional.

Propiedades	Tipo de dato
valor_hum	number
valor_temp	number
Registro	timestamp

La ontología "SensorAmbiente" se representa con el siguiente esquema JSON "SensorAmbiente Schema".

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "SensorAmbiente Schema",
    "type": "object",
```

```
"required": [
    "EmptyBase"
],
"properties": {
    "EmptyBase": {
        "type": "string",
        "$ref": "#/datos"
    }
},
"datos": {
    "description": "datos de temperatura y humedad",
    "type": "object",
    "required": [
        "valor_hum",
        "valor_temp",
        "registro"
    ],
    "properties": {
        "valor_hum": {
            "type": "number",
            "description": "humedad"
        },
        "valor_temp": {
            "type": "number",
            "description": "temperatura"
        },
        "registro": {
            "type": "object",
            "required": [
                "$date"
            ],
            "properties": {
                "$date": {
                    "type": "string",
                    "format": "date-time"
                }
            },
```

```
                    "additionalProperties": false,
                    "description": "tiempo "
                }
            },
            "additionalProperties": false
        }
}
```

Instancia JSON de la Ontología:

{"Sensor Ambiente":{ "valor_hum":28.6,"valor_temp":28.6,"registro":{"$date": "2018-01-30T17:14:00Z"}}}

Ontología "SensorLumi"

Para el sensor GY-30 de luminosidad se crea la ontología con las siguientes propiedades y no admitirá datos con propiedades adicionales a las establecidas.

Propiedades	Tipo de dato
valor_lum	number
Registro	timestamp

La ontología para este sensor se representa con el siguiente esquema JSON:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "SensorLumi Schema",
    "type": "object",
    "required": [
        "EmptyBase"
    ],
    "properties": {
        "EmptyBase": {
            "type": "string",
            "$ref": "#/datos"
        }
    },
    "datos": {
        "description": "Valor luminosidad",
```

```
            "type": "object",
            "required": [
                "valor_lum",
                "registro"
            ],
            "properties": {
                "valor_lum": {
                    "type": "number",
                    "description": "luminosidad"
                },
                "registro": {
                    "type": "object",
                    "required": [
                        "$date"
                    ],
                    "properties": {
                        "$date": {
                            "type": "string",
                            "format": "date-time"
                        }
                    },
                    "additionalProperties": false,
                    "description": "fecha"
                }
            },
            "additionalProperties": false
        }
}
```

Instancia JSON de la Ontología:

{"SensorLumi":{ "valor_lum":28.6,"registro":{"$date": "2018-01-30T17:14:00Z"}}}

Es así como de esta manera se pueden crear ontologías, a traves de clases que representen un dominio determinado, definiendo los atributos requeridos para una aplicación dada, este caso se lo realiza para los sensores de temperatura-humedad y luminosidad.

Las ventajas de incluir aspectos semánticos son:

- Describir ontologías basadas en formatos de texto que sean apropiados para dispositivos IoT incluyendo aquellos con capacidades restringidas.
- Dotar de conocimiento a los datos y aprovechar esta semántica para mejorar los procesos de consultas y procesos analíticos de datos.
- Representar las entidades de un sistema a través de descripciones semánticas de clases llamadas ontologías.
- Reutilización de ontologías a través del uso de modelos creados para diversos dominios.
- Lograr la interoperabilidad entre diferentes aplicaciones que comparten conceptos semánticos.

6. Discusión

En este capítulo se plantean las recomendaciones y buenas prácticas para la elección del protocolo más idóneo para la comunicación en un ambiente IoT. Además, se determinan los resultados del uso de las dos plataformas en la nube y se plantean casos prácticos en los que, el uso de tecnologías semánticas influye en ambientes generados por el IoT.

6.1. Tecnologías y Protocolos web para IoT

Una vez realizado el análisis de cada uno de los protocolos, a continuación, se propone algunas Recomendaciones o "Mejores Prácticas" que servirán para ser consideradas en la elección de determinado protocolo, de acuerdo a circunstancias específicas. Estas recomendaciones han sido planteadas tomando en cuenta casos de uso de estos protocolos como también aspectos definidos en las especificaciones de los estándares correspondientes a cada protocolo.

Las buenas prácticas recomendadas han sido agrupadas tomando en cuenta los siguientes aspectos: uso de tecnologías web existentes, calidad de servicio, escalabilidad, seguridad, transmisión, monitorización y control, consumo de batería, aplicaciones en "tiempo real" e integración con HTTP.

Mejor Práctica Tecnologías Web existentes 1. Los dispositivos de IoT tienen una limitación con respecto al tamaño de la carga, usar el protocolo HTTP para el envío de mensajes resulta ser un protocolo pesado (mensajes extensos) si se tiene un sistema con numerosos dispositivos conectados, en donde enviar o aceptar comandos se lo puede realizar en forma eficaz con un protocolo liviano.

Mejor Práctica Tecnologías Web existentes 2.- Usar protocolos de solicitud/respuesta se genera sobrecarga adicional que afecta el uso de batería, debido a las actualizaciones y sondeos que en ocasiones resultan inútiles en aplicaciones IoT, en contraparte a los protocolos publicador/suscriptor los cuales han sido creados para aquello.

Mejor Práctica Tecnologías Web existentes 3. Si se cuenta con un entorno web, el uso del protocolo CoAP será relativamente fácil, ya que es la versión liviana de REST y hereda el conocimiento definido del protocolo HTTP.

Mejor Práctica Tecnologías Web existentes 4. En redes cerradas que no admiten protocolos que no sean HTTP, para evitar conflictos con los puertos se recomienda usar MQTT sobre Websockets, AMQP sobre Websockets o el propio HTTP.

Mejor Práctica Calidad de Servicio (QoS) 1. Si bien MQTT ofrece tres niveles de QoS, se recomienda aplicar el Nivel 0 para aplicaciones que manejan datos de sensores ambientales ya que las publicaciones se realizan en intervalos pequeño de tiempo y la pérdida de un mensaje no tendría mayor relevancia.

Mejor Práctica Calidad de Servicio (QoS) 2. En el Protocolo MQTT el nivel de QoS 2, se recomienda utilizarlo en sistemas de facturación, ya que no se admiten duplicados o pérdidas de mensajes en este tipo de aplicaciones. Sin embargo, se debe tener presente que esta funcionalidad implica un mayor control entre publicador y el intermediario que llevará a que el proceso sea más lento.

Mejor Práctica Escalabilidad 1. Si se busca escalabilidad, utilizar un protocolo con arquitectura publicación/suscripción proporciona mejoras en cuanto al enfoque cliente/servidor, debido a que las operaciones en el intermediario pueden ser paralelizadas y procesadas por eventos. Y para miles de conexiones se recomienda utilizar nodos intermediarios agrupados para distribuir la carga en más servidores individuales con balanceadores de carga.

Mejor Práctica Escalabilidad 2. En un protocolo publicador/suscriptor, el broker es el eje central por donde cada mensaje debe pasar. Por lo tanto, es importante que sea altamente escalable, integrable en sistemas back-end, fácil de monitorear y resistente a fallas.

Mejor Práctica Seguridad 1. Con respecto a la seguridad es importante tomar en cuenta que, si bien CoAP está diseñado para implementaciones con restricciones, al añadir al protocolo una capa de seguridad DTLS hace que incremente el consumo de las capacidades de recursos del IoT, en este caso dependiendo la aplicación se debe dar prioridad a uno de los dos aspectos.

Mejor Práctica Seguridad 2. MQTT originalmente es diseñado para ambientes con recursos restringidos con sistema de seguridad integrado en aplicaciones de telemetría, por lo que puede ser empleado eficientemente en redes de sensores minimizando el máximo overhead.

En un escenario con puerta de enlace (gateway) entre los dispositivos y la plataforma IoT, el uso de MQTT o HTTPS no es óptimo cuando se utiliza una misma conexión TLS para conectar varios dispositivos IoT.

Mejor Práctica Transmisión 1. Es importante tomar en cuenta que, si bien el protocolo CoAP usa UDP, posee algunas funciones de TCP, las solicitudes/respuestas se

intercambian de forma asincrónica, distingue mensajes que requieren confirmación o no y reduce la sobrecarga del protocolo ya que los métodos, encabezado y códigos de estado siguen una codificación binaria.

Mejor Práctica Transmisión 2. Si no se requiere un sistema con las características de TCP y puede funcionar con las mínimas capacidades de UDP, CoAP es un protocolo que ayuda a reducir la carga total, por eso se lo llama protocolo de Internet ligero.

Mejor Práctica Monitorización y Control 1. El protocolo MQTT se orienta a grandes redes de pequeños dispositivos que requieren el control del lado de servidor en Internet, por lo tanto, no es ideal para la transmisión de datos de dispositivo a dispositivo, ni multidifusión de datos a muchos receptores.

Mejor Práctica Consumo de Batería 1. WAMP posee métodos de comunicación que se adaptan al IoT, sin embargo, su inconveniente es su adaptación a ambientes con dispositivos de recursos limitados, en especial al consumo de batería.

Mejor Práctica Consumo de Batería 2.- CoAP que corresponde a la versión liviana de REST, a pesar de que tiene las mismas desventajas que la arquitectura solicitud/respuesta, se ejecuta a través de UDP por lo que se lo puede utilizar para dispositivos con capacidades restringidas (limitadas cantidades de memoria, alta latencia y redes con ancho de banda bajo).

Mejor Práctica Tiempo Real 1. WAMP es capaz de comunicarse en forma nativa con el navegador web, por ende, resulta efectivo utilizarlo en aplicaciones casi en tiempo real.

Mejor Práctica Tiempo Real 2. AMQP y MQTT permiten inserciones de servidor de mensajes hacia los dispositivos de manera inmediata, por lo tanto, estos protocolos son elegibles cuando el parámetro importante es la latencia de entrega.

Mejor Práctica Tamaño de Carga 1. MQTT y AMQP son protocolos binarios que producen cargas más compactas que HTTP.

Mejor Práctica Transacciones completas 1. AMQP no es un protocolo ligero ya que fue creado para sistemas TIC, de esta manera es un sólido modelo para comunicaciones que soporta transacciones completas y se agrupa con otros protocolos IoT para trabajar en este ambiente garantizando este tipo de transacciones de información.

Mejor Práctica Integración con HTTP 1. Existen herramientas que permiten incluir información MQTT en HTTP cuando las aplicaciones finales no soportan este protocolo.

Existen soluciones que integran CoAP desde y hacia HTTP, mejorando de esta manera el problema de interoperabilidad con software y hardware específicos, así como también el uso de ancho de banda por parte de HTTP.

El uso de HTTP es importante para transacciones con grandes cantidades de información, sin embargo, no se recomienda su uso para información de video de alta velocidad.

Para el caso de implementación de plataformas para obtención de datos de un ambiente IoT dependerá de las necesidades de la plataforma o del sistema elegir el protocolo más idóneo. En cuanto a las plataformas comerciales y abiertas para IoT de acuerdo a la investigación realizada podemos observar en la figura que el protocolo más utilizado en estos ambientes es MQTT.

6.2. Web Semántica para IoT

Las tecnologías relacionadas con la Web Semántica al aplicarlas en la representación semántica de los datos generados por nodos del IoT trae consigo grandes beneficios: en lo que corresponde a la gestión de los datos, mejora el rendimiento de consultas de datos interrelacionados, controlando su exactitud, utilizando de mejor manera las ontologías definidas para optimizar la gestión en varios dominios.
A través de ontologías, que es la base de las tecnologías semánticas, se puede mejorar la interoperabilidad en grandes volúmenes de datos, de esta manera se puede aplicar en los siguientes ámbitos:

Entre estos ámbitos, destacan las ciudades inteligentes (*smart cities*): con una visión global de indicadores obtenidos a través de sensores que permiten gestionar y conocer datos de componentes relacionados para un dominio y proceder con medidas de prevención temprana, por ejemplo, sensores de movimiento y de temperatura que indiquen el caudal de los ríos y de la temperatura del ambiente para prevenir tragedias ocasionadas por desastres en los naturales. Permitiendo de esta manera realizar análisis predictivos de riesgos y protocolos dinámicos de actuación, reaccionando a cualquier emergencia en tiempo real.

En cuanto a turismo, mediante la interoperabilidad de datos en un dominio formado por las clases Turismo y Automotor se puede realizar consultas de información turística en tiempo

real y personalizada, obtener la mejor oferta turística entre otros servicios para la ciudad y el comercio.

En aspectos de seguridad, el dominio determinado por clases como Seguridad y Banca a través de ontologías y sus relaciones podría permitir una detección de fraude en el sistema.

Realizando un control en la operabilidad de datos provenientes de una clase Energía y otra Salud, se puede relacionar estos conceptos para gestionar de mejor manera el consumo eficiente de energía y la optimización de infraestructuras energéticas a través de la gestión domótica inteligente.

En la salud, a través de ontologías definidas en el ámbito de Seguridad y Salud se puede contar con un sistema de seguridad personalizado, a través de teleasistencia y dispositivos *wereables* para una persona, previniendo y controlando su salud.

En la movilidad, a través del conocimiento semántico generado en la relación de clases como Vehículos, Autoridades y Señalización se tiene una integración con infraestructuras de movilidad urbana permitiendo realizar análisis de patrones de actividad a partir de datos generados en el ámbito de movilidad.

7. Conclusiones y Trabajo Futuro

En este capítulo se plantean las conclusiones de este TFM como también queda abierta la brecha de actividades que podrían realizarse como un trabajo futuro relacionado a este tema de investigación.

7.1. Conclusiones

La gran variedad de dispositivos conectados a IoT con diferentes capacidades de comunicación, almacenamiento, recursos y las limitaciones que estos presentan, hace que existan tecnologías, protocolos y herramientas que mejoren la interoperabilidad y comunicación de dispositivos heterogéneos facilitando la implementación de estos ambientes.

La adaptación de la web actual al IoT abre la brecha para la creación de tecnologías específicas para este fin, de esta manera en este TFM se han analizado los protocolos Web existentes y los que han sido creados para IoT. ¿Cuál es el mejor?, no se puede confirmar, cada uno tiene sus pros y contras. Todos tienen su lugar en el IoT, su uso es relativo, es importante que se elija tomando en cuenta el que mejor se acople a sus requerimientos y la compatibilidad con el hardware y software con tales protocolos. Esto evitará problemas de incompatibilidad y asegurará el éxito de las aplicaciones IoT.

Así pues, en este TFM también se han analizado características técnicas de los protocolos de transporte físico para IoT y se ha propuesto una guía de buenas prácticas que permite elegir el protocolo de transporte físico más idóneo para la comunicación en un ambiente IoT.

Las recomendaciones planteadas sirven de guía para los desarrolladores de aplicaciones IoT que bajo características específicas como: calidad de servicio, escalabilidad, seguridad, consumo de batería, retardo de transmisión e integración con tecnologías web existentes como HTTP califican la idoneidad de un determinado protocolo de transporte.

Por otro lado, la web semántica es un paradigma relativamente nuevo en el IoT, su aplicabilidad aún se sigue definiendo, es por ello que hasta la entrega de este TFM, en el mercado existe una sólida plataforma semántica para IoT en Cloud, que inició como un proyecto de investigación, es la plataforma Sofía2.

Mejorar la interoperabilidad de dispositivos heterogéneos es el gran reto a resolver a través de la anotación semántica, esta descripción de la información permite definir el comportamiento de los objetos en la red.

En este trabajo de investigación, se han descrito y analizado tecnologías de la web semántica entre ellos estándares y herramientas aplicables al IoT, tanto en los casos de uso como en plataformas semánticas analizadas utilizan ontologías como la base para dotar de conocimiento a los datos obtenidos por sensores.

A través de las ontologías se puede describir semánticamente las entidades de un sistema IoT representando el mundo físico en un mundo digital, de esta manera modelando ontologías se puede simular sensores definiendo las propiedades para representar los datos a enviar/recibir y las restricciones que podrían admitirse para un tipo de datos.

Para la definición de ontologías existen varios formatos como JSON, XML, RFD los cuales se validan a través de sus correspondientes esquemas, estos formatos dotan de conocimiento a los datos facilitando su manipulación a las aplicaciones.

Se implementó un prototipo IoT con tres sensores, luminosidad, temperatura y humedad en los cuales se ha hecho una demostración de cómo modelar las ontologías correspondientes al tipo de datos que se está enviando, se definió las ontologías con formato JSON y su correspondiente esquema.

El principal propósito de aplicar tecnologías semánticas en el ámbito IoT es lograr la interoperabilidad entre diferentes aplicaciones que comparten conceptos semánticos a través de ontologías definidas.

Existen ontologías en dominios definidos, como en el ámbito SmartCity, Smart Energy, Smart Health, Security, que permiten ser reutilizadas por varios dispositivos como sensores o aplicaciones que manejen grandes cantidades de datos en común, mejorando los procesos de búsqueda de información, consultas o procesos analíticos con los datos enviados o recibidos por diferentes plataformas.

7.2. Trabajo Futuro

El IoT es un tema muy profundo a tratar, su aplicación a practicamente todos los campos, su influencia en actividades económicas y en el día a dia de las personas, hace que su adopción sea frecuente y su mercado cada vez más amplio.

Esta implantación tecnológica hace que requiera una constante evolución, abriendo nuevas posibilidades de investigación en su entorno, de esta manera nació la idea de este TFM en la investigación de la integración de tecnologías semánticas en el IoT.

Por un lado, el análisis de los protocolos y las tecnologías Web existentes, hizo que se plantee Buenas Prácticas que permitan determinar la más idónea para determinada aplicación IoT, sin embargo este análisis se lo realizó mediante una investigación cualitativa, quedando como futura línea de trabajo realizar una comparación cuantitativa de los protocolos para el transporte IoT obtenidos a través de la implementación de estos en un mismo ambiente.

Por el lado semántico, su repercusión en la integración con el IoT encierra más retos que la Web Semántica actual ya que se debe incorporar las mismas tecnologías con las limitaciones propias del IoT, haciendo que se requieran sólidos conocimientos de diseño e implementación que permitan armonizar estas dos tecnologías entre sí. Si bien, la redacción de este trabajo deja claro el enriquecimiento que obtienen los datos mediante tecnologías semánticas, se requiere de metodologías que guien el proceso de creación y reutilización de ontologías en varios ambientes, descriptores de sensores mediante modelos de datos alineados y compatibles.

Referencias

Arshdeep Bahga, V. M. (2014). *Internet of Things:A Hands-on Approach.* Georgia: A Hands on Approach.

AWS. (2018). *AWS IoT*. Obtenido en Junio de 2018, de https://aws.amazon.com/es/iot.

Barnaghi, P. P. (2010). Publishing linked sensor data. *Workshop on Semantic Sensor Networks (SSN)* , vol.668.

Bipin Upadhyaya, Y. Z. (2011). Migration of SOAP based services to RESTful services. *13th IEEE International Symposium on Web*, 105-114.

C. Pfister, S. O. (2011.). Getting Started with the Internet of Things. *O´Reilly Media Inc.*

Campoverde, A. M., Hernández, D. L., & Mazón, B. E. (2015). Cloud computing con herramientas open-source para Internet de las cosas.,173-182.

Castells, P. (2003). *La web semántica: Sistemas interactivos y colaborativos en la web.* Madrid: Univ de Castilla La Mancha.

Chuquimarca Sarango, E. F. (2017). *Evaluación de servicios en la nube para el desarrollo de aplicaciones IoT sobre redes inalámbricas de sensores.* Valencia: Universidad Politécnica de Valencia.

Cloud, G. (2018). *Google Cloud IoT*, Obtenido en Junio de 2018, de https://cloud.google.com/solutions/iot.

Codina, L. (2013). La web semántica: una visión crítica, El Profesional de la información, vol.12.

Corbin, A. S. (2002). *Bases de la investigación cualitativa, Técnicas y Procedimientos para desarrollar la teoría fundamentada.* (I. 958-655-624-7) Antioquia: Editorial Universidad de Antioquia.

D. M. Pezo, D. J.-Z. (2013). Procedimeinto para la construcción de Indices Semánticos basados en Ontologías de Dominio Específico. Vol.9 *Entramado* , 27-30.

ELECTRONICS, A. (2015). *ARROWS ELECTRONICS*. Obtenido en en Julio de 2018 de https://www.arrow.com/es-mx/research-and-events/articles/protocols-for-the-internet-of-things.

Evans, D. (2011). Internet de las cosas. Cómo la próxima evolución de Internet lo cambia todo. *Cisco Internet Bussiness Solutions Group-IBSG* , vol. 11, 4-11.

Fernández-López, M.-P. A. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. En S. University (Ed.), *AAAI-97 Spring Symposium Series.* Estados Unidos.

Fielding, R. T. (2000). *Software, Architectural Styles and the Design of Network-based.* University of California, Irvine.

Frank T. Johnsen, T. H. (2013). Evaluation of Transport Protocols for Web Services, Military Communications. 1-6.

Gruber, T. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *Technical Report KSL- Knowledge Systems Laboratory, Stanford Universit.* , 93-104.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Towards the Implementation of IoT for Environmental Condition Monitoring in Homes. *Future generation computer systems*, vol. 29(7), 1645-1660.

Gustafson, S. &. Sheth, A. (2014). Web of Things. *Computing Now* , vol. 7.

Hachem, S. T. (2011). Ontologies for the internet of things. *In Proceedings of the 8th Middleware Doctoral Symposium.* ACM.

hivemq. (2018). *MQTTessentials*. Obtenido en Julio de 2018, de https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe.

Hua-Dong, M. (2011). Internet of things: Objectives and scientific challenges. *Journal of computer science and technology* , 919-924.

IBM. (2018). *IBM Watson IoT*. Obtenido el 28 en Junio del 2018, de The Internet of Things becomes the Internet that thinks with Watson IoT: https://www.ibm.com/internet-of-things.

IETF_RFC7252. (2016). *IETF ORG*. Obtenido el 12 de Junio de 2018, de https://datatracker.ietf.org/doc/rfc7252.

Indra. (2018). *Minsait IoT Sofia2*. Obtenido el 28 de Junio de 2018, de https://sofia2.com/es.

ISO. (2018). *ISO/TS 10891:2009*. Obtenido el 10 de Julio de 2018, de https://www.iso.org/standard/46282.html.

ISO/IEC. (2013). *ISO/IEC 18092:2004 Information technology -- Telecommunications and information exchange between systems - Near Field Communication - Interface and Protocol (NFCIP-1)*. Obtenido el 23 de Junio de 2018, de https://www.iso.org/standard/38578.html.

LinkedData. (2015). *W3C ORG.* Recuperado el 11 de Junio de 2018, de https://www.w3.org/standards/semanticweb/data.

Lozano, T. A. (2011). Ontologías en la Web Semántica. *Jornadas de Ingeniería Web* vol. 1. España: Universidad de Extremadura.

Mesias, O. (2010). Metodología Cualitativa. *Seminario de Tesis.* Univerdidad Central de Venezuela.

Microsoft. (07 de 2018). *Azure IoT Hub*. Obtenido el 6 de Junio de 2018, de https://azure.microsoft.com/en-us/services/iot-hub.

MQTT, O. (2018). *mqtt*. Obtenido el 9 de Junio de 2018, de http://mqtt.org/documentation.

MS, G. (2017). Obtenido el 12 de Julio de 2018, de https://geeks.ms/jorge/2017/07/11/messaging-con-amqp-mqtt-y-stomp.

Nambi, S. A. (2014). A unified semantic knowledge base for IoT. *2014 IEEE World Forum on Internet of Things (WF-IoT),* 575-580. Seoul: IEEE.

Niño-Zambrano, M. A.-Z.-G.-F. (2013). Método de Indexación Semántica en Internet de las Cosas. *In Workshop on Semantic Web and Linked Open Data-Mexican International Coference on Computer Science. México.* Morelia: Michoacan.

Node.js, A. (2015). Obtenido el 15 de Junio de 2018, de https://nodejs.org/en/about.

ORG, W. (2018). *W3C Standards*. Recuperado el 27 de Mayo de 2018, de https://www.w3.org/standards/techs/rif#w3c_all.

Pastor, J. A. (2013). *Tecnologías de la web semántica* (Primera ed.).Editorial UOC.

Pedraza Jiménez, R. C. (2007). Web semántica y Ontologías el el procesamiento de Información. *El profesional de la Información* , 569-578.

Perera, C. Z. (2014). Context Aware Computing for The Internet of Things. *IEEE , doi:10.1109/SURV.2013.042313.00197* vol.16(1), 414-454.

Pérez, F. A. F., & Guerra, J. L. G. . (2017). Internet de las Cosas. *Perspectiv@ s* , 45-49.

Pfisterer, D. R. (2011). SPITFIRE: toward a semantic web of things. *Communications Magazine IEEE* (doi:10.1109/MCOM.2011.6069708), 40-48.

Rodríguez, S. D. (2015). Plataforma Semántica para Internet de las Cosas. *CICESE* .

Sarango, E. F. (2017). *Evaluación de servicios en la nube para el desarrollo de aplicaciones IoT sobre redes inalámbricas de sensores.* Universidad Politécnica de Valencia, Informática de Sistemas y Computadores. http://hdl.handle.net/10251/89152.

Senso, J. A. (2003). "Herramientas para trabajar con rdf". *El profesional de la Información.* , 132-139.

Tello, A. L. (2013). Ontologías en la Web Semántica. *Jornadas de Ingeniería Web*. Escuela Politécnica.

The MathWorks, I. (2018). *ThingSpeak*. Obtenido el 15 de Junio de 2018, de https://thingspeak.com.

Tim Berners-Lee, J. H. (2001). Semantic Web. *Scientific American* .

Ubidots. (2018). *Ubidots*. Obtenido el 24 de Junio de 2018, de https://ubidots.com.

Valencia, D. E. *Desarrollo de sistema embebido en tiempo real.* Trabajo de grado, Bogotá, Colombia: Universidad Nacional de Colombia.

Vasileios Karagiannis, P. C. (2015). A Survey on Application Layer Protocols for the Internet of Things. *Centre Tecnologic de Telecomunicacions de Catalunya* .

W3C. (2009). *OWL Web Ontology Language*. Obtenido el 20 de Junio de 2018, de https://www.w3.org/TR/owl-features.

W3C, O. (2015). *Linked Data.* Obtenido el 13 de Junio de 2018, de https://www.w3.org/standards/semanticweb/data.

W3C, O. (2017). *W3C RECOMENDATIONS*. Obtenido el 27 de Mayo de 2018, de https://www.w3.org/TR/vocab-ssn.

W3C, R. (Marzo de 2013). *SPARQL Query Language for RDF.* Obtenido el 11 de Junio de 2018, de https://www.w3.org/TR/rdf-sparql-query.

WAMP. (2015). *Transportes alternativos para WAMP*. Recuperado el 3 de Junio de 2018, de https://tools.ietf.org/html/draft-oberstet-hybi-tavendo-wamp-02#section-5.3.

Wang, W. D. (2012). A Comprehensive Ontology for Knowledge Representation in the Internet of Things. *In Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference*, 1793-1798.

XIVELY. (2018). *Xiveli*. Obtenido el 22 e Junio de 2018, de https://xively.com.