Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Ciencias de la Computación

Interacciones Proxémicas para el Internet de las Cosas

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de
Maestro en Ciencias

Presenta:

Luis Carlos Cárdenas Pérez

Ensenada, Baja California, México
2017

Tesis defendida por

Luis Carlos Cárdenas Pérez

y aprobada por el siguiente Comité

Dr. José Antonio García Macías
Director de tesis

Miembros del comité

Dr. Jesús Favela Vara

Dr. Luis Adrián Castro Quiroa

Dr. Gabriel Alejandro Galaviz Mosqueda

MC. Christian Paúl García Martínez

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Luis Carlos Cárdenas Pérez** como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Interacciones Proxémicas para el Internet de las Cosas

Resumen aprobado por:

_____
Dr. José Antonio García Macías
Director de tesis

El uso de las dimensiones proxémicas en entornos de Internet de las Cosas pueden ayudar a medir las interacciones entre personas y objetos, coadyuvando en la construcción de sistemas proactivos, conscientes de contexto y centrado en los humanos. Si bien se ha explorado el uso de dimensiones proxémicas en el ámbito del cómputo ubicuo, esto no ha permeado hacia plataformas de Internet de las Cosas. En este trabajo de tesis, se presenta ProximiThings Server, un sistema que permite el uso de interacciones proxémicas entre entidades en entornos de Internet de las Cosas. Se muestran los beneficios de incorporar dichas interacciones, mediante el desarrollo de escenarios realistas. También se dan detalles sobre la arquitectura de ProximiThings Server y cómo los desarrolladores pueden incorporarla en sus desarrollos.

Palabras clave: Internet de las Cosas, dimensiones proxémicas, sistema centrado en los humanos

Abstract of the thesis presented **by Luis Carlos Cárdenas Pérez** as a partial requirement to obtain the Master of Science degree in Computer Science.

Proxemic Interactions for Internet of Things

Abstract approved by:

_____

PhD. José Antonio García Macías
Thesis Director

Proxemics dimensions can help mediate interactions between people and objects in Internet of Things environments, contributing to the construction of proactive, context-aware and human centered systems. The use of proxemics dimensions has been explored in the realm of pervasive computing, however, this has not really permeated to Internet of Things platforms. In this work we introduce ProximiThings Server, a system to enable proxemics interaction between entities in Internet of Things environments. This work shows the benefits of having such interactions through realistic scenarios. Details about the architecture of ProximiThings Server are also presented as well as how developers can include them on their projects.

Keywords: Internet of Things, proxemics, human-centric system

Dedicatoria

*A mi esposa, **Vianey**, por su apoyo y amor en esta nueva etapa que comenzamos juntos*

*A mi **mamá Tere**†, por hacer de mí el hombre de bien que siempre quiso que fuera*

Agradecimientos

Al Dr. Antonio García Macías por su apoyo y conocimientos otorgados para el desarrollo de este trabajo de tesis

A mi esposa, Vianey, por su apoyo en esta etapa que vivimos juntos

A los miembros de mi comité de tesis, por su buena disposición para formar parte de este trabajo y su retroalimentación en cada avance

A los doctores del Departamento de Ciencias de la Computación que me dieron clases, por sus conocimientos e inspiración para el desarrollo de este proyecto

A Alberto, Alejandra, Gloria Ivonne, Arturo y Rewer por su apoyo en el desarrollo de los dispositivos y la incorporación de otros dispositivos para el entorno de pruebas

A Christian García y al equipo de Ubilogix por su apoyo y por proporcionar el espacio y las facilidades para realizar las pruebas de ProximiThings Server

Al INFOTEC y al equipo del FIWARE Lab México a través del Laboratorio Nacional del Internet del Futuro por su ayuda en la utilización de la nube de FIWARE Lab para pruebas y desarrollo.

Al Departamento de Ciencias de la Computación de CICESE, por los recursos proporcionados para el desarrollo de este trabajo de investigación y el apoyo económico para la conclusión del mismo.

Al proyecto SmartSDK por el apoyo económico para asistir a la capacitación para la plataforma FIWARE

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico proporcionado para mis estudios de maestría.

Tabla de contenido

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

En este capítulo se describe la motivación y el planteamiento del problema que inspiró el desarrollo de este trabajo de tesis, así como las preguntas de investigación y los objetivos que lo guiaron. Por último, se describe la organización del resto del documento y los capítulos en los que se divide.

1.1 Motivación

Recientemente, hemos visto un aumento exponencial en la cantidad de dispositivos conectados a Internet, una de las razones es por la adopción del Internet de las Cosas. De acuerdo con un estudio de la consultora Gartner, en 2020 van a existir alrededor de 26 mil millones de dispositivos conectados a Internet (Gartner, 2013). Este aumento exponencial se debe a que cada vez habrá más aparatos electrónicos conectados a Internet para complementar sus funcionalidades o darles un nivel de "inteligencia" según se requiera.

Todo esto es debido a que desde 1991, en el cómputo ubicuo se ha trabajado en diversos retos y ha hecho realidad la visión de investigadores como (Weiser, 1991). Con las nuevas tecnologías que aparecieron a partir de estos retos surgió el Internet de las Cosas en 1999. Con el Internet de las Cosas podemos integrar los objetos físicos al mundo virtual. Desde entonces, el cómputo ubicuo y el Internet de las Cosas han trabajado en ciertos problemas similares, pero de forma separada. (R. Ebling, 2016) destaca la intersección de problemas en ambas áreas, por lo que propone la fusión del cómputo ubicuo, el Internet de las Cosas y sus comunidades, para enfrentar estos problemas similares.

En (Satyanarayanan, 2001) se muestra la visión y los nuevos retos en la investigación del cómputo ubicuo y sistemas distribuidos. A 15 años de la publicación de ese artículo, Ebling publicó en la revista "*Pervasive Computing*" una revisión del artículo de Satyanarayanan en el cual se destaca el progreso realizado desde su publicación y los retos que todavía faltan por superar.

De acuerdo con el artículo de Ebling (Ebling, 2016), uno de los temas más explorados en estos 15 años es la conciencia de contexto (*context awareness*), la cual es una característica de los sistemas de cómputo ubicuo (*pervasive & ubiquitous computing*) que les permite adaptarse al contexto o situación actual de un usuario. Por el contrario, se menciona que uno de los temas que poco se ha trabajado en el cómputo ubicuo es el monitoreo de las intenciones de usuario (*user intent*), el cual permite automatizar algunos

aspectos en beneficio de las necesidades del usuario en un entorno inteligente. Aquí es donde radica la principal motivación de realizar este trabajo de investigación, monitorizar las intenciones de usuario en un entorno de Internet de las Cosas que permita proporcionar beneficios a las personas según la intención de uso que tengan con respecto a los dispositivos electrónicos.

1.2 Planteamiento del problema

El Internet de las Cosas es un campo de investigación relativamente nuevo, antecedido por otros campos como el cómputo ubicuo (*pervasive and ubiquitous computing*), pero con objetivos y problemas a resolver en común. El Internet de las Cosas nace de la necesidad de interconectar objetos en una red global de forma heterogénea, identificarlos y monitorizar la actividad que se realizan con estos objetos.

Una de las tendencias del Internet de las Cosas es hacer que las personas utilicen sus dispositivos de forma más natural, y que estos dispositivos sigan un diseño más centrado en las personas y sus necesidades. De acuerdo con (Wilson, 2014), para que el Internet de las Cosas sea más amigable para los humanos, necesita tener tres características: que los objetos se comuniquen entre ellos, que sean atractivos a la vista y que vayan más allá de solo controlarlos remotamente. Un ejemplo de dispositivos más amigables con las personas son los teléfonos inteligentes.

Las plataformas de Internet de las Cosas permiten comunicar dispositivos entre sí utilizando métodos y protocolos orientados a Internet. Esta comunicación ha permitido recolectar datos de sensores en los dispositivos y guardarlos en una base de datos en la nube, la cual permite el análisis de una gran cantidad de datos (*big data*[1]) y obtener cierto conocimiento de esta información. Por otra parte, esta comunicación también ha permitido que las personas puedan interactuar y controlar sus dispositivos de forma remota con controles especializados o a través de un teléfono inteligente.

Sin embargo, esta forma de interacción no está cerca de ser una forma de interacción natural para las personas, pues necesitamos el uso del teléfono inteligente y su interfaz, para comunicar una interacción con los dispositivos. Ejemplos de interacción natural entre las personas y una computadora es el rastreo de los movimientos del cuerpo y el habla.

[1] Big data: http://www.webopedia.com/TERM/B/big_data.html

De acuerdo con Edward Hall, la proxémica es un área que estudia las formas en las que las personas utilizan la distancia interpersonal para comunicarse con otras personas (Hall, 1966). Según (Greenberg, Marquardt, Ballendat, Diaz-Marino, y Wang, 2011), la proxémica puede ayudar a incrementar las posibilidades de interacción entre los dispositivos u objetos y las personas en un entorno.

En los últimos años, en el Laboratorio de Cómputo Móvil y Ubicuo del Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE) se ha trabajado en una plataforma que permita el procesamiento de interacción proxémica en un entorno de Internet de las Cosas. Así nació la plataforma OpenThings por (Calderon, Delgadillo, y Garcia-Macias, 2016a; Delgadillo, 2015). Sin embargo, OpenThings tiene la limitante de que solo procesa dos de las cinco dimensiones proxémicas utilizando razonadores semánticos. De acuerdo con (Calderon, Delgadillo, y Garcia-Macias, 2016b), muestran que quieren explorar con otras formas de procesar las dimensiones proxémicas que sean más flexibles para los desarrolladores, así como hacer que el sistema sea más modular y que tenga una forma de procesar acciones basado en un modelo de condiciones.

Es por eso que es necesario desarrollar una plataforma de Internet de las Cosas que permita el procesamiento de las 5 dimensiones proxémicas para mejorar la inferencia de las interacciones proxémicas. Además, que permita a los desarrolladores de tecnologías de Internet de las Cosas, incorporar las capacidades de interacción proxémica en sus entornos y comunicar acciones a otros dispositivos o servicios en Internet.

1.3 Propuesta

Para este trabajo de tesis se propone: diseñar y desarrollar una plataforma que permita el procesamiento de la interacción proxémica entre personas y objetos en un entorno de objetos inteligentes, y así apoyar en la monitorización de las intenciones de uso de los objetos para disparar acciones de forma proactiva en un entorno de Internet de las Cosas para beneficio de las personas.

1.4 Preguntas de investigación

De acuerdo a la motivación y el planteamiento del problema, se proponen las siguientes preguntas de investigación a responder durante el desarrollo de esta investigación:

- ¿Cuáles son los requerimientos de hardware que deben tener los dispositivos de Internet de las Cosas para medir las dimensiones proxémicas de cada entidad?
- ¿Cómo pueden las interacciones proxémicas contribuir a un Internet de las Cosas más centrada en los humanos?
- ¿Cómo se puede implementar un modelo de acciones y condiciones para disparar funcionalidades en objetos de un entorno de Internet de las Cosas?

1.5 Objetivo general

El objetivo principal de esta investigación es mostrar la utilidad de la aplicación de las dimensiones proxémicas para el monitoreo de las intenciones de usuario en un entorno de Internet de las Cosas.

1.6 Objetivos específicos

Con base en el objetivo principal, se plantean los siguientes objetivos específicos:

- O1: Desarrollar una plataforma para el procesamiento de la Interacción proxémica en un entorno de Internet de las Cosas.
- O2: Identificar los requerimientos y limitantes de hardware en los dispositivos de Internet de las Cosas para la medición de las dimensiones proxémicas.
- O3: Probar en escenarios realistas la utilidad de las dimensiones proxémicas para inferir intensiones de usuario.

1.7 Contribución

La principal contribución de esta investigación está en el desarrollo de una plataforma de Internet de las Cosas que permita realizar el procesamiento de las dimensiones proxémicas para conocer la interacción

proxémica entre los objetos y las personas. Con esto, se facilita el monitoreo de las intenciones de uso de los objetos para así proporcionar beneficios de forma proactiva a las personas. Esta contribución es tanto para los desarrolladores de plataformas, como para los usuarios finales de los entornos de Internet de las Cosas.

Adicionalmente, este trabajo de investigación presenta otras contribuciones con respecto a la literatura publicada a la fecha de publicación de este trabajo. Uno de ellos es el uso de sensores electrónicos como otra forma de medición de las dimensiones proxémicas. Además, este trabajo de investigación está atacando uno de los temas en los que poco se ha trabajado: la monitorización de las intenciones de uso en un entorno de Internet de las Cosas.

Por último, este trabajo contribuirá a extender las funcionalidades de la plataforma FIWARE[2], permitiendo que los desarrolladores de entornos de Internet de las Cosas puedan incorporar capacidades de interacción proxémica en sus entornos. El código de este proyecto será de acceso libre y cualquier persona lo puede descargar[3] y utilizar.

1.8 Metodología

De acuerdo con los objetivos de esta investigación, la metodología a seguir es la siguiente:

1. **Revisión y análisis del estado del arte sobre Internet de las Cosas:** En esta etapa, se tiene definido hacer una revisión exhaustiva de la literatura en el campo del Internet de las Cosas, con la finalidad de conocer los avances científicos hasta este momento, tomar aspectos que sean útiles para la investigación y resolver algunas preguntas acerca del campo de investigación.
2. **Análisis de requerimientos técnicos sobre dispositivos del Internet de las Cosas:** En esta etapa, se realiza un análisis de requerimientos para que los dispositivos puedan medir las dimensiones proxémicas. Asimismo, se revisa como incorporar otros dispositivos de Internet de las Cosas en el entorno y que sirvan para los escenarios de prueba que se van a realizar en las siguientes etapas.
3. **Desarrollo e implementación de una plataforma para el Internet de las Cosas:** En este punto, se va a desarrollar una plataforma de Internet de las Cosas utilizando como base la plataforma FIWARE para procesar la interacción proxémica.

[2] Fiware: https://www.fiware.org/about-us/
[3] https://github.com/faxterol/ProximiThings-Server

4. **Implementación y pruebas de concepto del entorno de Internet de las Cosas:** Por último, en esta etapa se va a unir el trabajo realizado en la etapa dos y tres. Es decir, unir el software y el hardware desarrollado para integrar el entorno del Internet de las Cosas. Esto permitirá afinar detalles respecto al funcionamiento que se tenía pensado en etapas anteriores y observar las limitaciones para el trabajo futuro. Además de mostrar la utilidad de las dimensiones proxémicas en el monitoreo de las intenciones de usuario.

1.9 Organización de la tesis

Para que el lector pueda conocer a detalle este trabajo de tesis, el resto del documento se organiza en los siguientes capítulos:

Capítulo 2. Marco teórico: En este capítulo se describe el estado del arte sobre Internet de las Cosas y su relación con las dimensiones proxémicas. Se describe también como se están utilizando las dimensiones proxémicas para apoyar la interacción humano-computadora en el cómputo ubicuo. Además, se explica cómo está evolucionando el Internet de las Cosas para apoyar a las necesidades de las personas en un nuevo paradigma llamado “Internet de las Personas”.

Capítulo 3. Diseño y desarrollo de ProximiThings Server: En este capítulo se describen los requerimientos de hardware y software necesarios para el desarrollo de una plataforma que procese las dimensiones proxémicas. También se describe a detalle la arquitectura de ProximiThings Server y ejemplos de uso.

Capítulo 4. Implementación y pruebas de concepto de ProximiThings Server: En este capítulo se describe como fue implementado y evaluado ProximiThings Server, así como la descripción de los escenarios y los resultados obtenidos en la prueba de concepto. De la misma manera, se describe como un grupo de desarrolladores lo usaron para un entorno de Internet de las Cosas.

Capítulo 5. Conclusiones y trabajo futuro. En este capítulo se muestran las conclusiones obtenidas en este trabajo de investigación. También se muestran sus aportaciones y limitaciones. Adicionalmente se muestran las formas en que este trabajo investigación puede aportar a otros trabajos en el futuro.

Capítulo 2. Marco Teórico

Para facilitar la comprensión de este trabajo de tesis, es necesario mostrar la definición de los conceptos más importantes para entender el contexto en el que se abordó.

2.1 Internet de las Cosas

Como su nombre lo dice, el Internet de las Cosas (conocido como *Internet of Things* o *IoT* por sus siglas en inglés) es un concepto que engloba el uso de tecnologías para proporcionarle a los objetos capacidades de cómputo, almacenamiento de información y conectividad. Por objetos nos referimos a cualquier cosa en nuestro entorno: el reloj de pulso, la taza de café, los electrodomésticos, la televisión, las macetas del jardín, etc. Cualquier objeto electrónico o no electrónico puede ser parte del Internet de las Cosas. Sin embargo, las capacidades de cómputo en los objetos pueden ser limitadas, debido a sus características en tamaño, peso y almacenamiento de energía.

Aun así, con el rápido avance tecnológico, podemos utilizar chips más pequeños con mayor capacidad de cómputo. Esto nos ayuda a que los objetos también sean inteligentes. Algunos objetos del Internet de las Cosas no pueden estar conectados a la corriente eléctrica ni a un cable de conexión a Internet, por lo que utilizan tecnologías de comunicación inalámbrica y baterías. Otros objetos con mayores dimensiones, pueden estar conectados a la corriente eléctrica todo el tiempo, tener mayores capacidades de cómputo y monitorear con sensores algunos aspectos según su funcionalidad.

Weiser, del laboratorio “Palo Alto Research Center” (PARC) de Xerox propuso que la tecnología debe estar incluida en las cosas que utilizamos todos los días, de forma imperceptible para las personas (Weiser, 1991). Para lograr esto, se tenían que desarrollar diferentes tipos de sensores y actuadores que pudieran ser pequeños, eficientes y baratos.

De acuerdo con (Li, Xu, y Zhao, 2010), el Internet de las Cosas es el resultado de la convergencia de tres visiones diferentes: Internet, objetos y semántica (ver **Figura 1**) las cuales se explican a continuación.

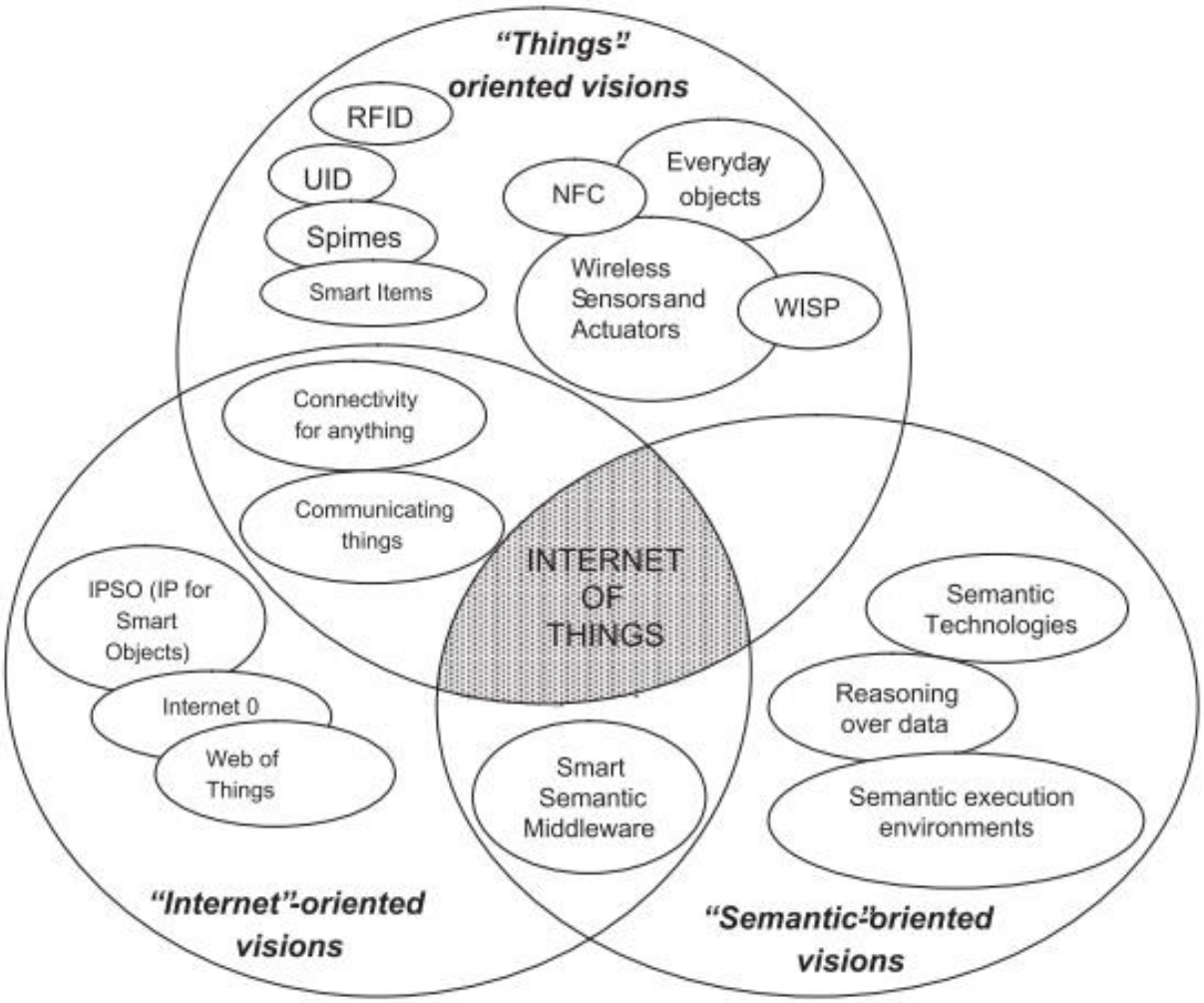


Figura 1 Las tres visiones que hacen intersección para inspirar el IoT (Li, Xu, y Zhao, 2015)

2.1.1. Visión orientada al Internet

En esta visión se encuentra el desarrollo de métodos, protocolos y tecnologías de conexión que permite el intercambio de información entre diferentes dispositivos. Este intercambio se realiza por medio de servicios, implementando protocolos como HTTP y representando la información con RESTful (conocido también solo como REST). Esta visión permite que los objetos inteligentes consuman y proporcionen servicios y para lograrlo se requiere de tecnologías como el Protocolo IP para proporcionarle a los objetos una dirección, además de protocolos en capa física como 6LoWPAN o CoAP. En el **Apéndice 1** se muestran otras tecnologías para la conexión de dispositivos de IoT.

Los servicios proporcionados por el Internet de las Cosas pueden ser consumidos desde cualquier lugar del mundo con conexión a Internet, los dispositivos pueden tener un nombre y una dirección lógica.

2.1.2. Visión orientada a los objetos

En esta visión se encuentran los objetos de la vida diaria, sean o no objetos electrónicos y pueden ser para uso específico como un medidor de voltaje o para uso general como una televisión. A través de nuestro sentido de la vista, los humanos podemos identificar a los objetos, sus propiedades y características. En el cómputo, es necesario tener métodos o tecnologías que nos permitan identificar digitalmente a los objetos.

En 1999, Kevin Ashton propone el uso de R.F.I.D. (*Radio Frecuency Identification*) para identificar a los objetos en una cadena de producción, además de proponer el término "*Internet of Things*" (Ashton, 2009). Posteriormente, aparecen tecnologías como N.F.C. (*Near Field Communication*) o W.S.A.N. (*Wireless Sensor and Actuator Network*) (Li et al., 2010), esto nos permite "borrar la línea entre el mundo real con el mundo digital" (Presser y Gluhak, 2009).

Si unimos la visión de los objetos con el Internet, podemos tener una red de objetos interconectados, con identificación y dirección única que implementan protocolos estándares para el consumo y producción de servicios de sensado.

2.1.3. Visión orientada a la semántica

En la visión de la semántica, tenemos tecnologías basadas en ontologías y razonadores semánticos para procesar el contexto de un entorno. En este sentido, RDF (*Resource Description Framework*) es un formato de archivos que permite representar información semántica en grafos. Con base en esta representación, los procesadores semánticos pueden obtener información entrelazada a través de una consulta, al estilo de una consulta (*query* en inglés) a una base de datos. Para realizar esta consulta, tenemos una tecnología llamada SparQL[4] (Gil, Ferrández, Mora-Mora, y Peral, 2016).

En esta misma visión podemos incluir la Inteligencia Artificial y el Big Data, ya que el Internet de las Cosas genera enormes cantidades de información que un agente inteligente puede identificar y generar beneficios a las personas y a los mismos dispositivos.

[4] SparQL 1.1: https://www.w3.org/TR/sparql11-overview/

2.2 El papel de los sensores en el Internet de las Cosas.

Los sensores son dispositivos que miden aspectos físicos y químicos del ambiente. En los últimos años se han incluido sensores en los dispositivos para mejorar sus funcionalidades. El ejemplo más claro es el teléfono inteligente o *smartphone*. Pero más allá de sensar aspectos físicos o químicos del ambiente a través del teléfono, los sensores del teléfono inteligente han permitido monitorizar las actividades de las personas. Con esto, los proveedores de servicios tecnológicos, fabricantes y desarrolladores pueden mejorar los servicios en sus plataformas, por lo que las personas se convirtieron en el principal objetivo de sensado y principal fuente de datos.

De acuerdo con (Srivastava, Abdelzaher, y Szymanski, 2011), el sensado centrado en los humanos tiene tres categorías en términos de la extensión y el rol de las personas como lo ilustra la **Figura 2**. Las categorías son:

- **Humanos como objetivo de sensado**: Esta categoría es aplicada principalmente en la seguridad. El objetivo de las aplicaciones que caen en esta categoría es sensar las actividades de las personas, comportamientos y patrones.
- **Humanos como operadores de sensores**: En esta categoría se encuentran las aplicaciones que necesitan de las personas para obtener datos, dando un papel más activo a las personas. Un ejemplo son las campañas de sensado participativo en las cuales las personas actúan como un sensor, que con base a su experiencia proporcionan información acerca de un fenómeno.
- **Humanos como fuente de datos:** En esta categoría se encuentran aplicaciones que utilizan la información que las personas generan como: sus movimientos, información histórica e interacciones sociales a través de redes sociales.

Toda la información recolectada por los sensores también es útil para inferir en qué contexto se encuentra una persona. (Abowd et al., 1999) define el contexto como toda la información característica de un entorno o situación en el que un usuario se encuentra. La información de los sensores puede estar etiquetada en un tiempo y espacio definido, lo cual puede ayudar a restringir acciones a ejecutar en el entorno de Internet de las Cosas. Si una computadora o un s*martphone* puede procesar en qué contexto está un usuario, entonces las aplicaciones se enriquecerán, produciendo servicios más útiles.

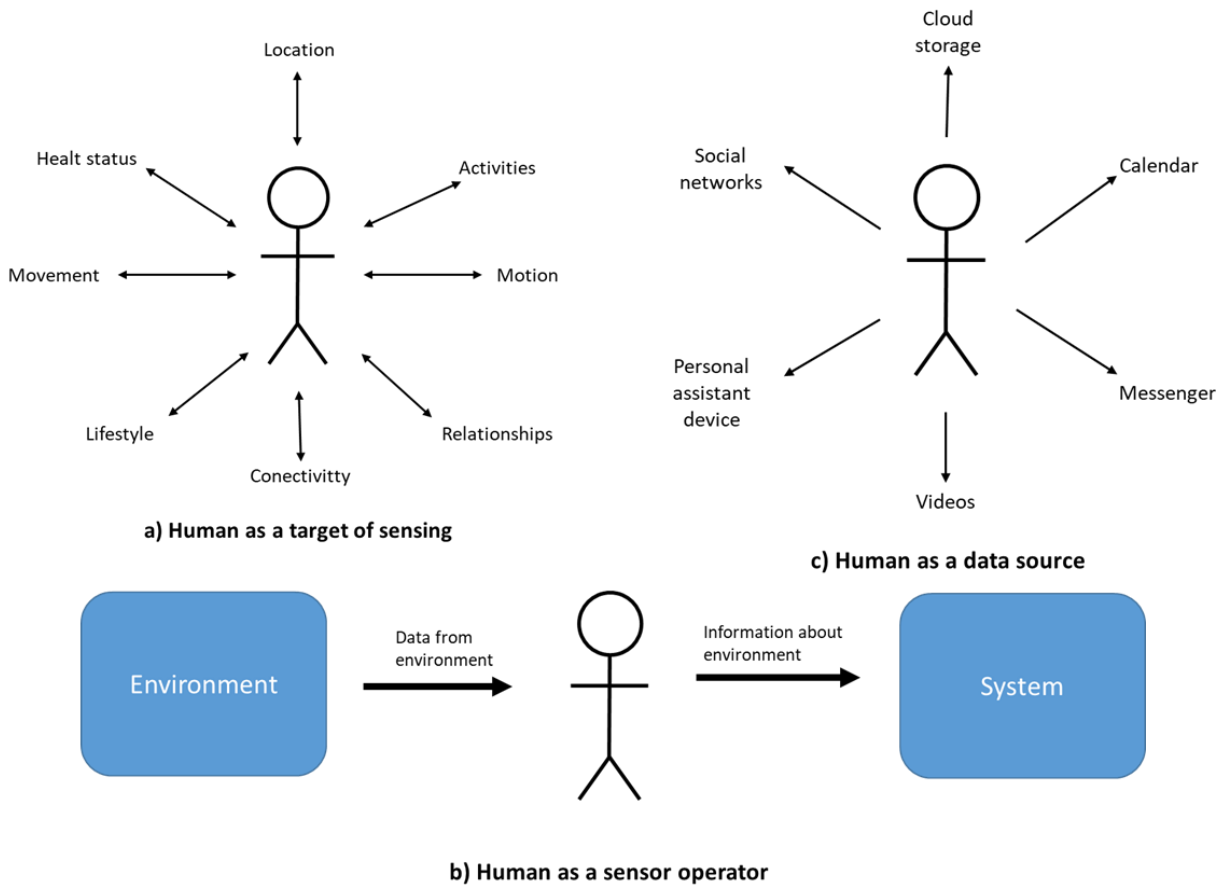


Figura 2 Categorías del sensado centrado en los humanos. a) humano como objetivo de sensado. b) Humano como operador de sensor c) Humano como fuente de datos.

La red de sensores inalámbricos (*Wireless Sensor Network*) es una red en la cual cada nodo que lo conforma contiene sensores que miden aspectos físicos y químicos del ambiente. Una red de sensores tiene la capacidad de comunicar la información de forma inalámbrica. Cada nodo tiene capacidades de almacenamiento y procesamiento limitados.

2.3 Proxémica como una forma de interacción humano-computadora

El uso de dispositivos de entrada y las interfaces de usuario son una forma de interacción humano computadora que utilizan un pequeño subconjunto de formas y habilidades que tenemos los humanos para interactuar con otras entidades. De acuerdo con (Greenberg et al., 2011), el uso de las dimensiones proxémicas tiene un gran potencial de uso como una interfaz de usuario natural. Por ejemplo, (Yang et al.,

2016) propone un método para percibir una intención de uso de un objeto capturando el movimiento de las manos y rastreando movimiento de los ojos de un usuario. De igual manera, (Greenberg y Kuzuoka, 1999) utilizan objetos físicos para comunicar una intención de comunicación con otra persona de forma remota rastreando la posición de una persona con respecto a una computadora para comunicarse.

(Greenberg et al., 2011) ha propuesto utilizar la proxémica para interactuar y cambiar la configuración en dispositivos como pantallas multimedia o videojuegos. Sin embargo, en su trabajo se destaca el uso de la visión por computadora para hacer mediciones de las dimensiones proxémica. Por lo cual, deja abiertas las posibilidades de implementar otras formas de medir las dimensiones proxémicas.

2.3.1 La proxémica en la antropología

La proxémica describe la relación entre una persona y su entorno, e identifica cómo las personas utilizan la distancia para comprender cómo interactúan de acuerdo a su identidad cultural (Hall, 1966). Este término fue acuñado por Edward Hall, quien establece cuatro zonas proxémicas que dependen de la distancia en la que dos personas o entidades están interactuando. Las zonas son: pública, social, personal e íntima (ver **Figura 5**).

Además de la proxémica, Edward Hall propuso una serie de características fijas y semifijas que afectan las interacciones entre las personas o animales como se muestra en la **Figura 3**. Las características físicas marcan un límite entre los espacios que son utilizados para propósitos específicos y donde se realizan actividades específicas. Las características semifijas aplican a entidades cuya posición puede afectar el espacio y hacer que las personas se alejen o se acerquen a dicha entidad.

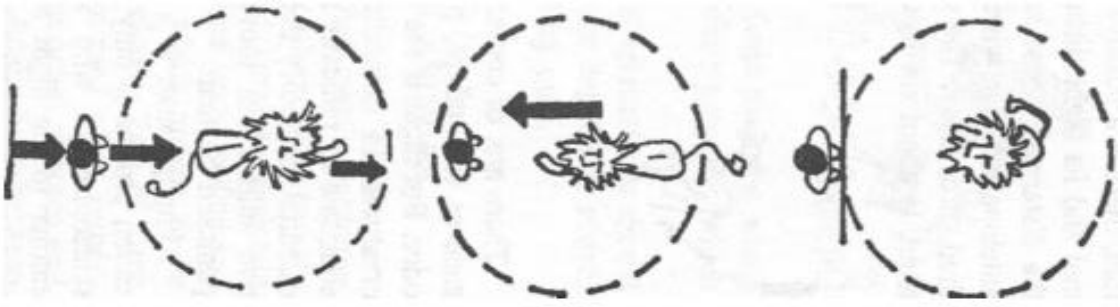


Figura 3 Demostración de cómo puede afectar la distancia en la interacción entre personas y animales (Hall, 1966)

2.3.2 La proxémica en el cómputo ubicuo

Tomando esta idea de la proxémica en la antropología, surge la proxémica en el cómputo ubicuo, especialmente en el área de Interacción Humano-Computadora. De acuerdo con Saul Greenberg, la proxémica puede ayudar a que las personas interactúen con las computadoras de forma natural, incrementando la posibilidad de interacción y conectividad con otros objetos de acuerdo a su proximidad en un entorno específico. Sin embargo, la proxémica de cómputo ubicuo se refiere al uso del espacio de una persona para comunicarse con otros dispositivos, y como una forma de interacción humano-computadora. Las dimensiones proxémicas que define (Greenberg et al., 2011) son las siguientes:

- **Distancia**: El espacio que existe entre dos entidades que pueden ser una persona y un objeto proporcionada en cualquier unidad de medida continua y que puede ser convertida a unidades discretas por rangos de distancia. Una forma de convertirla es usando las zonas proxémicas de Hall o las 4 fases de interacción de (Vogel y Balakrishnan, 2004a).
- **Orientación**: Es el ángulo de una entidad con referencia a otra y que puede ser medida en grados sexagesimales de la misma forma que lo hace una brújula y al igual que la distancia, los grados sexagesimales como unidad continua puede ser convertida a una unidad discreta. Algunos ejemplos de unidades discretas son: “en frente de”, “a un lado de” y “detrás de”.
- **Identidad:** Es la descripción de una entidad dada por su identidad exacta y/o sus atributos. Estos últimos pueden dar una idea acerca de la entidad con la finalidad de poder distinguirlas de otras entidades en un entorno definido.
- **Movimiento:** Es el cambio con respecto al tiempo de la distancia y la orientación de una entidad. El movimiento puede dar una idea acerca de la actividad que está ejecutando una entidad, por ejemplo: moverse en dirección a otra entidad, velocidad de movimiento o si una entidad cambia de lugar.
- **Ubicación:** Es la descripción física de una ubicación con referencia a los objetos o entidades que están en el mismo lugar. Con la localización de los objetos en un mismo lugar, se puede dar una idea del contexto del lugar y las limitaciones que tienen las entidades en la ubicación definida.

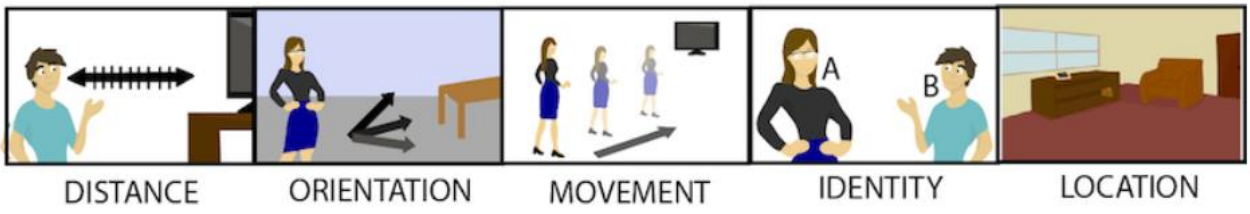


Figura 4 Dimensiones proxémicas en el cómputo ubicuo establecidas por Greenberg (Garcia-Macias y Cardenas, 2017)

Las dimensiones proxémicas (**Figura 4**) en cómputo ubicuo pueden ayudar a que los objetos inteligentes ejecuten ciertas acciones con base en la medición de una o varias dimensiones. Si integramos las dimensiones de la proxémica en el diseño y desarrollo de sistemas interactivos, podemos crear ambientes proactivos que permitan satisfacer e incluso anticiparse a las necesidades de las personas de acuerdo a su comportamiento. Esto ayuda a que un entorno de Internet de las Cosas sea más centrado en los humanos que en los dispositivos en sí.

En el trabajo de (Bâce, Sörös, Staal, y Corbellini, 2017) utilizan el sonido para medir la dimensión proxémica de la distancia. Cuando los relojes inteligentes de dos personas están cerca, intercambian la información de contacto entre ambos. Ambos dispositivos se sincronizan para que uno escuche y el otro emita un sonido. Utilizan sonidos que son inaudibles para los humanos (superior a los 20 KHz), ya que basta con que los dispositivos se escuchen para deducir que están a una distancia corta.

En el trabajo de (Kortuem, Kray, y Gellersen, 2005), desarrollan un sistema que muestra en un grafo, la ubicación de dispositivos y computadoras. Para ello, los autores desarrollaron unos dispositivos que se conectan a la computadora por medio de USB, emiten ultrasonido para reconocer la distancia y orientación de los dispositivos, y se sincronizan a través de una red ad-hoc. Un sistema instalado en cada computadora procesa la información de los sensores y crea el gráfico relacional de cada una de las entidades.

En el trabajo desarrollado por (Calderon et al., 2016a), utilizan Bluetooth Low Energy para procesar las dimensiones proxémicas de la identidad y la distancia entre dispositivos utilizando el teléfono celular como dispositivo de abstracción de una persona. Además, aprovechan la orientación del magnetómetro del teléfono para procesar la orientación de la persona con respecto a los demás dispositivos. Por las características de Bluetooth, es posible identificar cuanta distancia hay entre un dispositivo y otro a través de las emisiones de *beacons*.

2.3.3 Medidas discretas de las dimensiones proxémicas

La combinación de las dimensiones proxémicas proporciona un marco de trabajo para la interacción humano computadora. Estas dimensiones pueden ser medidas de diversas formas. Sin embargo, estas mediciones suelen ser continuas. Para dar mejor sentido a estas medidas, se utilizan valores discretos según la dimensión proxémica. En el caso de la distancia, (Hall, 1966) propone las siguientes zonas

proxémicas, que son rangos de distancia que hay entre una entidad con respecto a otra como se muestra en la **Figura 5**. Las zonas proxémicas son:

- **Zona pública**: Todas las entidades a más de 12 pies (3.65m) de distancia.
- **Zona social**: Todas las entidades a una distancia de entre 4 y 12 pies (1.21m – 3.65m).
- **Zona personal**: Todas las entidades a una distancia de entre 1.5 y 4 pies (0.45m – 1.21m).
- **Zona íntima**: Todas las entidades a menos de 18 pulgadas (0.45m) de distancia.

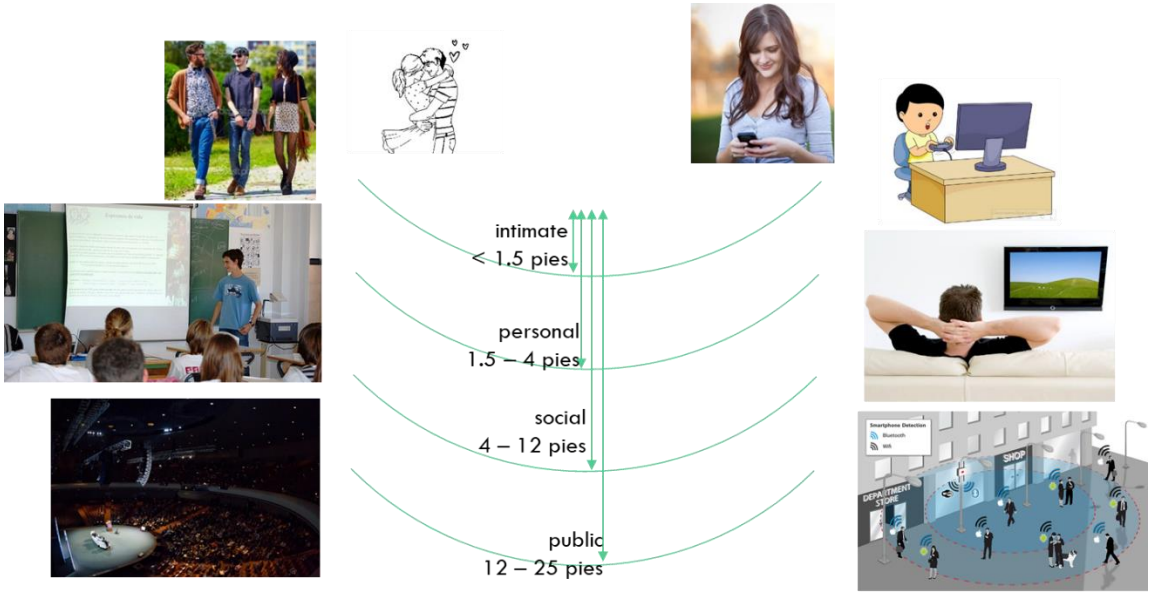


Figura 5 Comparación de las zonas proxémicas entre individuos con el uso de objetos

En cuanto a la dimensión proxémica de la orientación, está la orientación relativa de una entidad con respecto a otra.

- **A la izquierda**: Todas las entidades que se encuentran del lado izquierdo de otra entidad.
- **A la derecha**: Todas las entidades que se encuentran del lado derecho de otra entidad.
- **En frente**: Todas las entidades que se encuentran enfrente de otra entidad.
- **Detrás**: Todas las entidades que se encuentran detrás de otra entidad.

Con respecto a las dimensiones de la ubicación y la identidad, estos no son propiamente medidas continuas, sino que son atributos. En un entorno de IoT que utilice dimensiones proxémicas, no es obligatorio utilizar un identificador propio para un lugar o para una persona u objeto, ya que se pueden utilizar atributos que los describan. Algunos atributos para lugar pueden ser:

- Es para preparar alimentos.
- Tiene una capacidad de 20 personas.

- Hay una cafetera, una mesa y 10 sillas.
- Se encuentra en el piso 2 del edificio de telemática.

Algunos atributos para personas pueden ser:

- Género.
- Edad.
- Rol con respecto a un lugar o espacio de tiempo.
- Estatura.
- Vestimenta.

Para el caso del movimiento, un valor discreto puede ser la variación de la distancia de una entidad con respecto a otra o la variación de distancia entre este objeto con respecto a un obstáculo. Algunos valores discretos que puede adoptar el movimiento pueden ser:

- Quieto.
- En movimiento.
- Caminando.
- Girando hacia la izquierda o derecha.

2.3.4 Fases de Interacción de Vogel y Balakrishnan

Las fases de interacción propuestas por Vogel y Balakrishnan (Vogel y Balakrishnan, 2004b) son un marco de trabajo en el cual se describen los principios de diseño para interactuar con pantallas ambientales. La finalidad de este marco de trabajo es que las pantallas ambientales muestren la información correcta a la persona adecuada. Algunos de estos principios utilizan las mediciones de las dimensiones proxémicas para activar las fases o hacer transiciones entre ellas. Algunas dimensiones proxémicas que utilizan las fases de interacción son: Distancia, Orientación e Identidad.

Las fases de interacción de Vogel y Balakrishnan se muestran en la **Figura 6** y son:

- **Fase ambiental**: En esta fase, una pantalla muestra información que puede ser de interés para la persona en el espacio donde se encuentre. La información puede ser también para atraer a la persona a las otras fases y mostrar información relevante y privada.

- **Fase de interacción implícita**: En esta fase, el usuario tiene su enfoque en trasladarse de un lugar a otro y la pantalla está en su trayectoria, por lo que la pantalla le muestra al usuario información de forma abstracta pero que el usuario entiende.
- **Fase de Interacción sutil:** En esta fase, el usuario tiene un enfoque en la pantalla. El usuario se acerca a la pantalla para obtener información detallada. La información mostrada puede ser particular y personal para el usuario.
- **Fase de interacción personal:** En esta fase, el usuario se encuentra interactuando directamente con la pantalla a una distancia cerrada. El usuario accede a las notificaciones privadas en la pantalla.

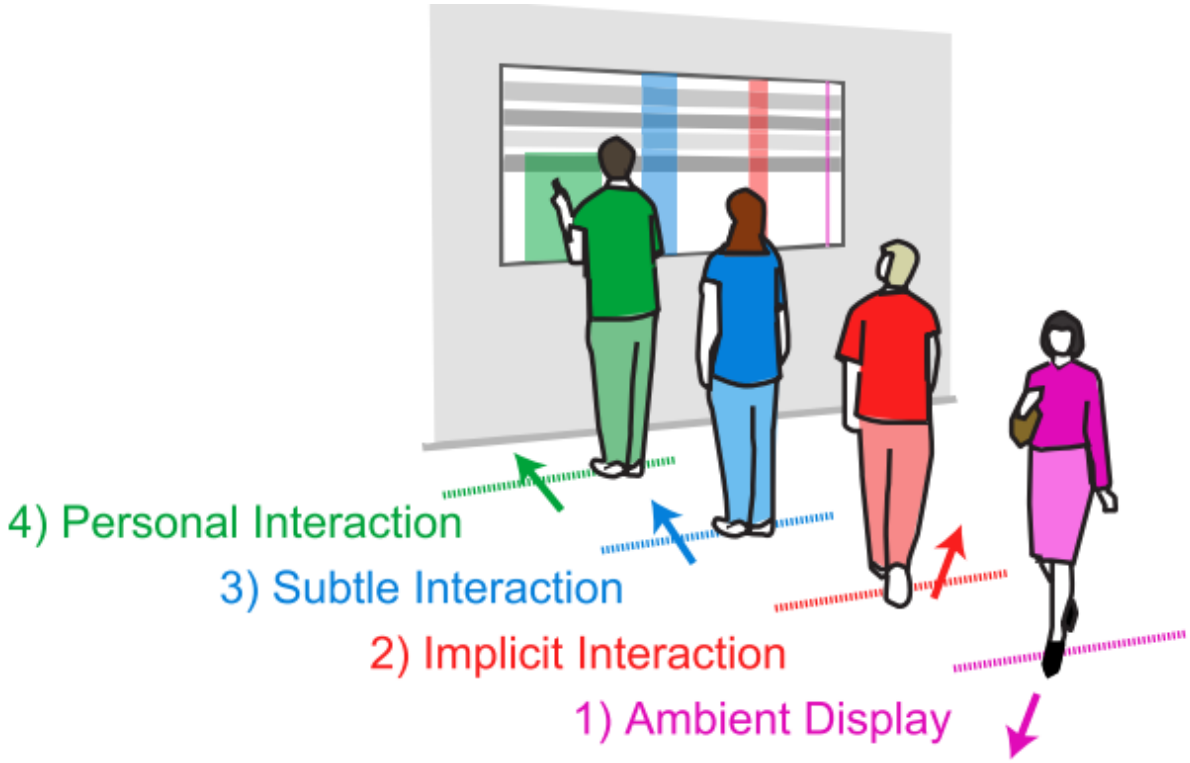


Figura 6 Fases de interacción (Vogel y Balakrishnan, 2004b)

2.4 Intención de Usuario

De acuerdo con el Diccionario de Inglés de Cambridge, una intención (*intent*) es "algo que una persona desea o planea hacer[5]", en este caso la persona es el usuario de un sistema ubicuo. La intención de usuario es una forma intangible de conocimiento porque forma parte de nuestro comportamiento e intuición natural (Yang et al., 2016). La intención de usuario también puede especificar una secuencia de acciones

[5] Intent: http://dictionary.cambridge.org/dictionary/english/intent

basado en el comportamiento actual y pasado de un individuo (Papadopoulou, Abu Shaaban, Gallacher, Taylor, y Williams, 2010). La comunicación forma parte de nuestra naturaleza y una forma de comunicación es el habla, el cual puede ser usado para comunicar las intenciones que un usuario tiene (Sun, Pappu, Chen, y Rudnicky, 2016). Un ejemplo de intención de usuario en el cómputo son los sistemas asistidos por voz como *Siri*, que podemos decirle: "llama a casa" y la intención que el usuario comunicó de forma verbal es realizada por el sistema, realizando todos los pasos que conlleva realizar esa intención.

En la literatura, existen trabajos que han explotado la realización de tareas por medio de la expresión verbal y textual de la intención de usuario en los *Smartphones* (Sun et al., 2016; Zhuang, Mei, Hoi, Xu, y Li, 2011). En el cómputo ubicuo existen trabajos que utilizan gestos físicos para comunicar intensiones de usuario, algunos gestos son el movimiento de las manos y el monitoreo del movimiento de los ojos (Yang et al., 2016), esto utilizando visión por computadora. El proyecto Aura es una propuesta arquitectónica para soportar el procesamiento de contexto e intenciones de usuario con base en el uso de los dispositivos en el entorno (Sousa y Garlan, 2002). De forma comercial en el Internet de las Cosas está por ejemplo Google Home, que es un dispositivo al cual le podemos comunicar intenciones de usuario por voz y que nos regresa un resultado de forma multimedia. De igual manera con Amazon Echo y el sistema de reconocimiento del habla llamado Alexa.

En el trabajo de (Papadopoulou et al., 2010), utilizan la monitorización de las intenciones de usuario y con el historial pueden inferir el comportamiento de un usuario para proporcionar proactividad en un sistema de cómputo ubicuo.

2.5 Internet de las Personas

De acuerdo con (Wilson, 2014), las personas deben ser parte intrínseca de los entornos de Internet de las Cosas. Es importante entender como las personas interactúan con los objetos a su alrededor así entender mejor su experiencia de usuario. Para hacer a las cosas más amigables con las personas, los entornos de IoT deben tener tres puntos importantes:

- **Comunicación entre los objetos que usamos**: Si bien, los objetos actualmente se pueden comunicar entre ellos, pero siempre por orden explicita de un usuario a través de su teléfono inteligente. No existe un control acorde al comportamiento o los movimientos de una persona. Por ejemplo, que el clima de una habitación se empiece a ajustarse minutos antes de que la persona llegue a su casa.

- **Los objetos no necesitan ser llamativos:** Usar los objetos tiene la finalidad de realizar una tarea. Si un objeto llama la atención de un usuario perjudica su atención. La ciencia cognitiva sugiere que para el Internet de las Cosas es necesario diseñar objetos de tal forma que evite minar la atención de los usuarios.
- **No limitarse solo al control remoto:** Hacer que los objetos formen parte de nosotros, como una extensión del cuerpo, como cuando escribimos en el papel, la mano y el lápiz forman inconscientemente un todo.

Los dispositivos electrónicos se están integrando cada vez más en la vida diaria y el dispositivo más común que utilizamos es el teléfono celular o *smartphone. P*ara las aplicaciones, un *smartphone* es la representación de una persona. De todos los dispositivos conectados a Internet, un subconjunto de este universo representa a las personas, pero las personas no son objetos.

En un entorno de Internet de las Cosas centrado en los humanos, la tecnología aprende de la situación de las personas y sus expectativas, permitiendo disminuir la intervención de las personas en el uso de los dispositivos tanto como sea posible. El uso del Internet de las Cosas para el beneficio y para satisfacer las necesidades de las personas es lo que llaman el Internet de las Personas (Miranda et al., 2015a).

Para lograr que los dispositivos se centren más en las necesidades humanas, una opción es enfocarse en la visión de la semántica y de los objetos del Internet de las Cosas. En cuanto a los objetos, es necesario equiparlos de sensores que permitan hacer las mediciones de las dimensiones proxémicas, ya sea construyendo tus propios dispositivos o utilizar dispositivos comerciales. Algunos proporcionan la documentación para que desarrolladores puedan incluirlos en entornos de Internet de las Cosas y otros necesitan de ingeniería inversa para saber cómo utilizarlos. Con los datos de los sensores e inspirados en la visión semántica, se puede procesar los datos de estos para generar diversos beneficios a las personas de forma proactiva.

La proactividad es la capacidad de los sistemas de cómputo ubicuo o de IoT para adaptarse a las necesidades de los usuarios. Esto implica la necesidad de personalizar el comportamiento de un sistema con respecto al contexto de un usuario. La proactividad de los sistemas también permite la anticipación de necesidades y requerimientos de los usuarios.

Con la conversión de las mediciones discretas de las dimensiones proxémicas, un sistema de IoT puede disparar beneficios de forma proactiva a los usuarios. Con la aplicación de la proactividad en un sistema de Internet de las Cosas logramos obtener un entorno de Internet de las Personas según lo propuesto por (Miranda et al., 2015b).

Una propuesta que utiliza la visión semántica del Internet de las Cosas para desarrollar una plataforma de IoT centrado en las personas es la de (Calderon et al., 2016a), en la que utilizan el espacio y el tiempo como variables contextuales para disparar beneficios proactivos en beneficio de las personas en un entorno de IoT.

2.5.1 Proxémica de cómputo ubicuo en el Internet de las Personas

(Greenberg et al., 2011) ha demostrado en sus experimentos que el uso de la proxémica en el cómputo ubicuo puede dirigir la ejecución de diferentes acciones en los dispositivos. Para hacerlo, hay que realizar mediciones en las dimensiones de la proxémica y otorgarles un significado según sus valores discretos o continuos.

Para hacerlo desarrollaron Proximity Toolkit, (Marquardt, Diaz-Marino, Boring, y Greenberg, 2011) una herramienta de trabajo para procesar las dimensiones proxémicas utilizando visión por computadora. Para ello, desarrollaron un servidor que procesa el video obtenido por cámaras. El desarrollador del entorno de IoT modela las características fijas y semi-fijas de un ambiente definido en 3D y define las entidades que interactúan en dicho ambiente. El sistema rastrea el movimiento de las entidades y realiza las mediciones de las dimensiones proxémicas con base al modelo definido. La información de las dimensiones proxémicas de cada entidad está disponible en un API REST.

En uno de los experimentos de Greenberg, se tiene un espacio con una televisión y una persona, como en la **Figura 7**. La televisión muestra diferentes caras de expresión según los valores de las dimensiones de la proxémica. Por ejemplo, pone cara de enojada si la persona se acerca mucho a ella, debido a que está dentro del espacio privado de la televisión. Si la persona se aleja y está dándole la espalda a la televisión, ésta muestra una cara triste debido a que la persona la está ignorando. En estos experimentos se utilizaron algunas dimensiones proxémicas, como la distancia y la orientación, para ejecutar esas acciones.

Figura 7 Experimento de Greenberg utilizando las dimensiones proxémicas (Greenberg et al., 2011)

2.6 Arquitectura de una plataforma de Internet de las Cosas

Un entorno de Internet de las Cosas implica el uso de diversas tecnologías de hardware, software y protocolos de comunicación compuestos en capas. Autores como (Gil et al., 2016), (Gubbi, Buyya, Marusic, y Palaniswami, 2013), (Kosmatos, Tselikas, y Boucouvalas, 2011) o (Botta, de Donato, Persico, y Pescapé, 2016) muestran algunos esquemas y arquitecturas de plataformas de Internet de las Cosas, principalmente utilizando el *cloud computing* como punto de procesamiento de información.

Por su parte, (Gil et al., 2016) hace una compilación de diversos artículos en las que analiza las arquitecturas y servicios de plataformas de Internet de las Cosas. En este artículo se muestra una arquitectura general como el de la **Figura 8** y está compuesta por las siguientes capas:

- **Dispositivos de IoT:** son los dispositivos físicos en el entorno de Internet de las Cosas. Cada dispositivo tiene sus propios sensores y envía la información a un servicio de procesamiento en diversos formatos y múltiples protocolos.
- **Tipos de datos y taxonomías:** En esta capa unifica la estructura de datos y los tipos de datos obtenidos de los dispositivos. Esta capa permite homogenizar las estructuras de datos y protocolos de comunicación de todos los dispositivos a uno solo para entregar estos datos a las demás capas.
- **Pre procesamiento:** Esta capa realiza un procesamiento de la información de tal forma que la información obtenida de los dispositivos pueda ser útil para una aplicación en específico
- **Servicio de minería de datos:** Este servicio puede hacer un minado de datos históricos. Por lo general, este servicio se encuentra en un servidor en la nube para su óptimo rendimiento.
- **Aplicación:** Esta es la capa en la que el usuario final obtiene los datos que necesita y que ya fueron procesados por la plataforma de IoT.

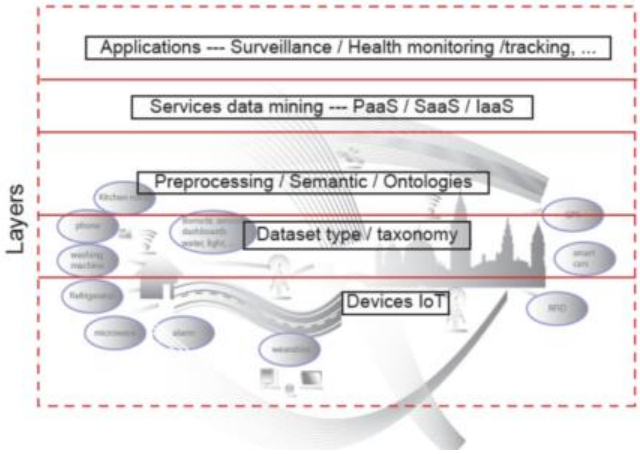


Figura 8 Arquitectura de una plataforma de Internet de las Cosas (Gil et al., 2016)

En la plataforma FIWARE podemos encontrar algunos componentes que permiten realizar las tareas de estas capas. FIWARE es una plataforma de desarrollo para entornos de Internet de las Cosas, *smart cities* y procesamiento de contexto. La plataforma es una iniciativa de la Unión Europea, y México participa a través del programa Horizon 2020[6]. FIWARE está compuesto por componentes de software llamados *Generic Enabler* (GE), que pueden ser utilizados según las necesidades de los desarrolladores. En el grupo de Internet de las Cosas tenemos GEs para administrar dispositivos IoT, agentes para abstraer a estos dispositivos y para comunicarse con ellos en distintos protocolos. El principal GE es el *Orion Context Broker*, que permite el intercambio de información entre otros GEs. También existen otros componentes que se llaman *Domain Specific Enabler* (DSE). Estos componentes procesan información específica como: análisis de imágenes, análisis de audio y video, inteligencia de negocios, salud y energía.

FIWARE y sus componentes son de uso libre y pueden ser utilizados a modo de prueba y con fines de investigación en una nube llamada FIWARE Lab y cumplir ciertos requisitos adicionales. Los Generic Enablers que nos permiten formar esta arquitectura de capas son las siguientes:

- **Dispositivos de IoT:** En cuanto a la administración de los dispositivos de Internet de las Cosas, esta *Backed Device Management IDAS*[7]. Para la conectividad de los dispositivos con el resto de las capas, tenemos el *IoT Agent JSON*[8].

[6] Horizon 2020: https://ec.europa.eu/programmes/horizon2020/en/what-horizon-2020
[7] IDAS: https://catalogue.fiware.org/enablers/backend-device-management-idas
[8] IoT Agent JSON: https://github.com/telefonicaid/iotagent-json

- **Tipos de datos y taxonomías:** Para los tipos de datos y taxonomías, tenemos el Orion Context Broker[9].
- **Pre procesamiento:** Para el procesamiento, el Orion Context Broker también nos puede ayudar, pues podemos enviar desde el OCB a otro software que nos permita procesar la información. De igual manera si nuestra aplicación utiliza multimedia, podemos utilizar Kurento[10] para el pre procesamiento de audio y video.
- **Servicio de minería de datos:** Para esta capa tenemos el componente Cosmos, Cygnus o Comet[11], que nos permiten analizar datos en lote o por stream y conectarlo a Hadoop.
- **Aplicaciones:** Por último, para la capa de aplicaciones se encuentra SpagoBI[12] o Wirecloud[13], para la construcción de interfaces de usuario.

2.7 Resumen

En este capítulo se describe lo que es la interacción proxémica a través de las dimensiones proxémicas en el computo ubicuo, y cómo se puede aplicar en un entorno de Internet de las Cosas para hacerlo más amigable a las personas. Se incluye información sobre el surgimiento del Internet de las Cosas y cómo fue que algunos autores se inspiraron en la antropología para implementar las dimensiones proxémicas, las medidas discretas de estas dimensiones y su combinación.

También se describe la transición del Internet de las Cosas hacia un Internet de las Personas, hacer que las personas sean el centro en un entorno de Internet de las Cosas y no solo objetos de este entorno. Por último, se detalla una arquitectura general para desarrollar una plataforma de Internet de las Cosas, lo cual muestra un panorama sobre cómo se construyó ProximiThings Server.

[9] Orion Context Broker: https://github.com/telefonicaid/fiware-orion
[10] Kurento: https://www.kurento.org/
[11] Cosmos: https://catalogue.fiware.org/enablers/bigdata-analysis-cosmos
[12] SpagoBI: https://www.spagobi.org/
[13] Wirecloud: https://github.com/Wirecloud/wirecloud

Capítulo 3. Diseño de ProximiThings Server

En este capítulo, se presentan detalles de la propuesta de solución al problema que se aborda en este trabajo de investigación, así como el cumplimiento de los objetivos planteados. Con esta propuesta de solución nace ProximiThings Server, una plataforma de IoT para el procesamiento de las interacciones proxémicas. Posteriormente se muestran los requerimientos y los aspectos más importantes de la arquitectura de ProximiThings Server, así como lo necesario para que un desarrollador de software pueda implementar ProximiThings Server en su propio entorno de Internet de las Cosas.

3.1 Requerimientos

3.1.1 Requerimientos no funcionales

Para el desarrollo de esta propuesta se deben tomar en cuenta los requerimientos necesarios para el procesamiento de las dimensiones proxémicas en un entorno de Internet de las Cosas. De acuerdo con (Srivastava et al., 2011) algunos requerimientos no funcionales para hacer un sensado más centrado en los humanos son los siguientes:

- Adquirir información de un objetivo de interés utilizando sensores.
- Procesar la información relevante de los sensores.
- Analizar y combinar la información de múltiples fuentes de datos.
- Tratar a las personas como objetivos del sensado.
- Utilizar cómputo en la nube para que se pueda archivar, analizar y compartir la información con otros sistemas.

Por otro lado, para hacer un entorno de Internet de las Cosas más centrado en los humanos, debemos simplificar el acceso de las personas a estos dispositivos. Actualmente, uno de los accesos más simples que tenemos para un entorno de Internet de las Cosas es a través del *smartphone*. Pero según (Miranda et al., 2015b), es necesario que un entorno de Internet de las Cosas tenga mecanismos propiamente adaptados a la situación en la que una persona se encuentra, y no que la persona se adapte a los mecanismos de un entorno de Internet de las Cosas. Por ello, se plantean los siguientes requerimientos para hacer un sistema de Internet de las Cosas centrado en los humanos:

- Las personas no deben ser tratados como objetos, sino como el centro de atención en un entorno de IoT.
- Es necesario contar con el soporte para una variedad de protocolos de comunicación, para integrar la mayor cantidad de dispositivos.
- Es necesario reducir la complejidad de la interacción del usuario con los dispositivos.
- Debe haber un dispositivo que permita identificar o representar a una persona.
- La infraestructura debe hacer que las aplicaciones estén conectadas a los dispositivos a través del *cloud computing* para permitir el cálculo de la información obtenida de los sensores.

3.1.2 Requerimientos funcionales

De acuerdo con (Uckelmann, Harrison, y Michahelles, 2011), una de las funcionalidades principales de un entorno de IoT es la integración de la identificación, el sensado y la actuación. Con el sensado se obtiene información que después es procesada para activar un actuador.

Proximity Toolkit tiene un mecanismo para obtener las mediciones de las dimensiones proxémicas de las entidades en un entorno de IoT. Este mecanismo utiliza visión por computadora, teniendo como entrada las imágenes de una cámara y de esas imágenes obtiene los datos en crudo para obtener los valores de las dimensiones proxémicas. Proximity Toolkit también tiene una interfaz gráfica para especificar las entidades en el entorno que captura la cámara. La información de los sensores (Marquardt et al., 2011).

En el trabajo de (Calderon et al., 2016a) se muestran algunos requerimientos funcionales para un sistema de Internet de las Cosas centrado en los humanos. El principal de esos requerimientos es la implementación de un esquema de acciones basado en un modelo de evento-condición-acción.

3.2 Arquitectura básica de un entorno de Internet de las Cosas

Antes de entrar en detalles acerca de la arquitectura de ProximiThings Server, es necesario explicar un poco acerca de la arquitectura básica de un entorno de Internet de las Cosas. Como ya se mencionó anteriormente, un entorno de Internet de Internet de las Cosas tiene tres elementos básicos: Dispositivos, La Nube y Las aplicaciones finales (**Figura 9**).

- **Dispositivos**: Son los objetos físicos encargados de leer y modificar aspectos del ambiente.
- **Nube**: Es la infraestructura de la red en donde se encuentran todas las capacidades de cómputo, procesamiento, almacenamiento y conectividad de los dispositivos con los usuarios o aplicaciones finales.
- **Aplicaciones finales**: Son los programas que muestran los datos a los usuarios finales o que necesitan de los datos del entorno de IoT para aspectos específicos.

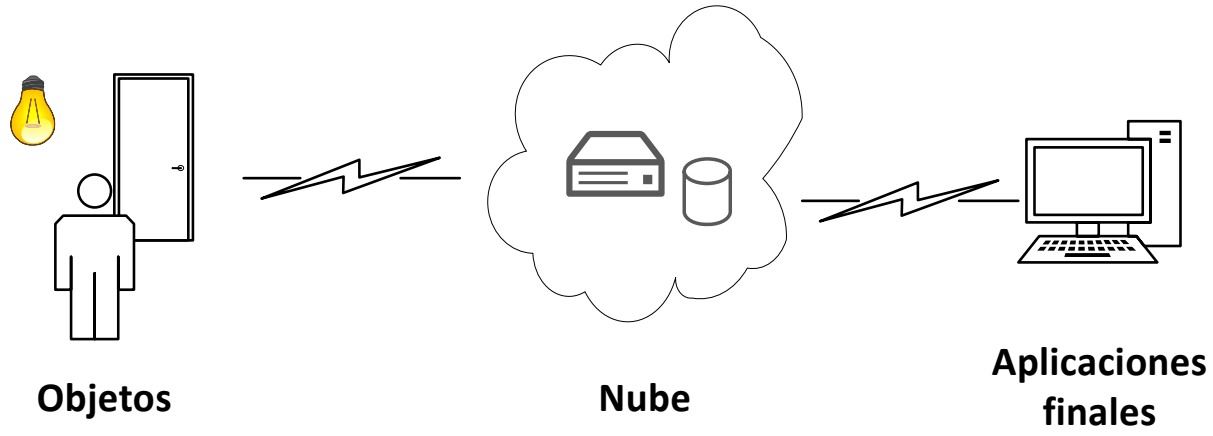


Figura 9 Arquitectura básica de un entorno de IoT

En este trabajo de investigación, se han desarrollado algunos dispositivos en la plataforma Arduino[14], los cuales realizan las mediciones de las dimensiones proxémicas. También se han utilizado algunos otros dispositivos como actuadores para cambiar el ambiente. En la nube está instalado el ProximiThings Server, que es el servicio encargado de procesar las mediciones de las dimensiones proxémicas y activar las funcionalidades de los dispositivos correspondientes.

Además, se cuenta con un API REST para guardar datos de los dispositivos y obtener datos generados por ProximiThings Server. Los datos de las dimensiones proxémicas se muestran en una interfaz web que facilita su visualización.

[14] Arduino: https://www.arduino.cc/

3.3 Medición de las dimensiones proxémicas en los dispositivos

En esta sección, se detallan los sensores que se utilizaron para hacer las mediciones de las dimensiones proxémicas en los dispositivos de Internet de las Cosas.

Distancia

Para medir la distancia, utilizamos sensores de proximidad de ultrasonido como el que se muestra en la **Figura 10**. Los HC-SR04 nos permiten medir la distancia en centímetros desde el dispositivo de IoT al objeto más cercano utilizando una bocina para emitir una onda de ultrasonido y un micrófono para escucharla y así medir el tiempo de vuelo. El tiempo de vuelo es el tiempo que tarda en regresar una emisión de ultrasonido al receptor a partir del momento en que es emitido. Cuando una onda de ultrasonido se topa con un objeto, esta onda rebota y regresa al sensor. Esto lo hace a una frecuencia de 40KHz.

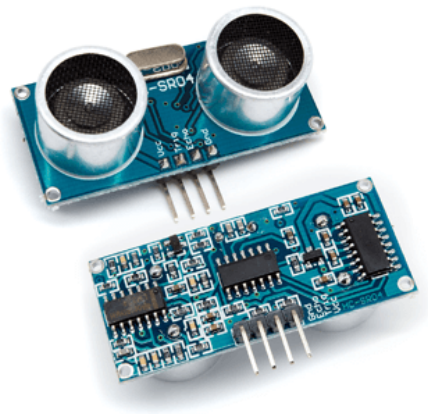


Figura 10 Sensor para medir la distancia HC-SR04

El HC-SR04 proporciona el tiempo que tardó la onda de ultrasonido en rebotar. Por lo que utilizaremos este dato para medir la distancia con la siguiente formula:

$$\mathrm{D} = \frac{x}{148}$$

Donde:

- D: Es la distancia recorrida por la onda de ultrasonido en pulgadas

- x: es el tiempo en microsegundos para que la onda de ultrasonido regresara al sensor a partir de que es emitido. También es la resta del tiempo en que fue recibida la onda de ultrasonido menos el tiempo en que la onda de ultrasonido fue emitida.
- 148: es el tiempo en microsegundos que le toma en recorrer 2 pulgadas de distancia (1 pulgada de ida y otra de regreso) a una onda de ultrasonido.

Orientación

Se utilizó el sensor magnetómetro MPU-9250 (ver **Figura 11**) para medir la orientación tomando como referencia el polo norte magnético. Estos sensores regresan una medida de 0 a 359° grados, donde 0° es el polo norte magnético. Sin embargo, no se quiere medir el ángulo con respecto al norte magnético, sino el ángulo de una entidad con respecto a otra entidad. Posteriormente se explica cómo se convierte la orientación de los dispositivos a una orientación relativa.

Figura 11 Sensor magnetómetro MPU-9250, similar a una brújula

Identificación

Para identificar a las personas se utilizó la emisión de un ID en infrarrojo, esto por las limitaciones para que las personas puedan portar un dispositivo. El ID en infrarrojo estará emitiendo cada segundo, por lo que otros dispositivos deberán captar esta señal en infrarrojo.

Ubicación

Para procesar la ubicación, se tendrán dispositivos en varios lugares. La configuración de este dispositivo almacenará su ubicación, por lo que este dispositivo al recibir la señal de identificación infrarroja de una persona, se inferirá que la persona está en la misma ubicación que el dispositivo.

Movimiento

Para calcular el movimiento, se tomarán los valores de la distancia en todo momento. En ProximiThings Server se va a procesar si una entidad está o no en movimiento.

3.4 ProximiThings Server

ProximiThings Server es un servicio que permite procesar las dimensiones proxémicas de interacción entre dos o más entidades. ProximiThings Server es compatible con FIWARE y se alimenta de la información que proporciona la plataforma FIWARE para el Internet de las Cosas. ProximiThings Server fue desarrollado con NodeJS, utiliza MongoDB como sistema de base de datos y se le asignó el puerto 6253 para el API REST a través de HTTP.

3.4.1 Arquitectura de ProximiThings Server

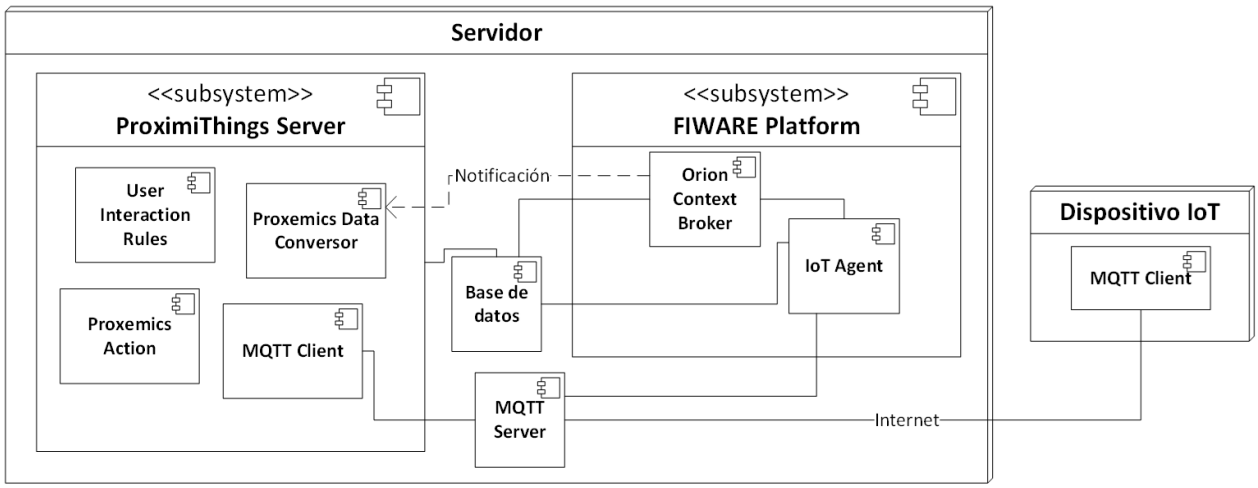


Figura 12 Arquitectura de ProximiThings Server

En la **Figura 12** se muestra la arquitectura completa de ProximiThings. Está compuesto de tres grandes módulos:

- **ProximiThings Server:** Es el servicio desarrollado por este trabajo de tesis que procesa las dimensiones proxémicas de interacción.
- **FIWARE Platform:** Contiene los componentes necesarios para conectarse con los dispositivos de IoT y enviar o recibir información de ellos, así como para enviar la información a ProximiThings Server.
- **IoT Device:** Se especifica que cada dispositivo de IoT tiene un cliente MQTT[15]. MQTT es el protocolo utilizado para enviar y recibir la información de los dispositivos.
- Adicionalmente, necesitamos del MQTT Server para conectar estos tres grandes módulos.

3.4.2 Componentes de FIWARE Platform

La plataforma FIWARE tiene componentes para conectarse con dispositivos de IoT y recibir o enviar información. Estos componentes son desarrollados por la comunidad de software libre. Los componentes de la plataforma FIWARE que utilizamos son los siguientes:

Orion Context Broker

El Orion Context Broker (OCB) es un componente de software que permite el intercambio de información con otros componentes. Es el componente medular de la plataforma FIWARE ya que los demás componentes no pueden funcionar sin el OCB.

OCB tiene algunas características que lo hacen ideal para apoyar el desarrollo de esta propuesta:

- **Abstracción de entidades:** OCB permite abstraer una entidad y asignarle atributos, parámetros o información de contexto.
- **Capas de servicio:** OCB tiene la capacidad de manejar capas de servicio según las necesidades de los desarrolladores. Se pueden tener las mismas entidades y manejar diferentes atributos en cada entidad según el servicio que está manejando la entidad. Los servicios son independientes, por lo que al hacer peticiones al OCB, se debe de especificar la capa de servicio. En la **Figura 13** se muestra un ejemplo de las capas.

[15] MQTT: De las siglas en inglés *Message Queue Telemetry Transport* (Ver definición en el Apéndice 1)

- **Rutas de servicio:** Para un entorno de Internet de las Cosas, las rutas de servicio permiten posicionar entidades en un lugar específico (una habitación) dentro de otro lugar más grande (como un piso). Puede tener tantos niveles como sean necesarios y facilita la consulta de entidades según el lugar o ruta de servicio.
- **Servicio de notificaciones y suscripciones:** Las suscripciones permiten que un componente de software pueda recibir una notificación cada vez que se actualiza uno o más atributos de una o más entidades según se especifique en la suscripción. Las notificaciones son enviadas a través de Webhooks (Notificaciones HTTP).

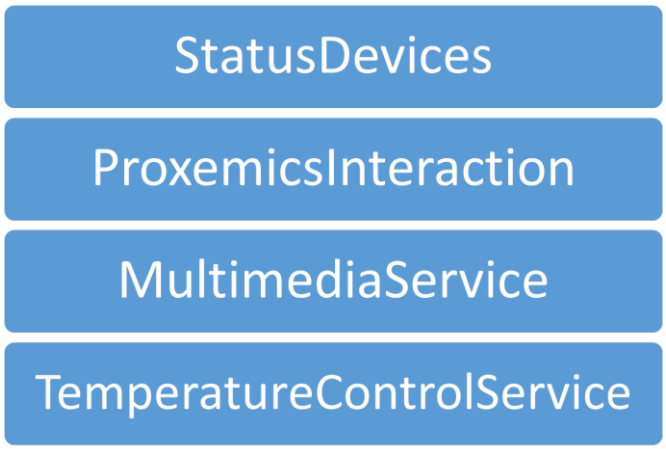


Figura 13 Ejemplo de capas de servicio en OCB para manejar la información según el tipo de servicio

IoT Agent MQTT-JSON

El agente de Internet de las Cosas (*IoT Agent*) es un componente que forma parte de la plataforma FIWARE que permite establecer una comunicación entre el OCB y un dispositivo conectado a un entorno de IoT para después, mostrar la información a un cliente, como se muestra en la **Figura 14**. Este agente de IoT se comunica con los dispositivos utilizando el protocolo MQTT. Para poder transferir los datos al OCB por medio del IoT Agent, es necesario enviarle los datos al IoT Agent en formato JSON. Además, es necesario hacer un mapeo de parámetros, lo que implica que a los parámetros o atributos que tengamos en el OCB, le asignemos otro nombre en el IoT Agent pero más corto. Esto se debe hacer por las características del protocolo MQTT.

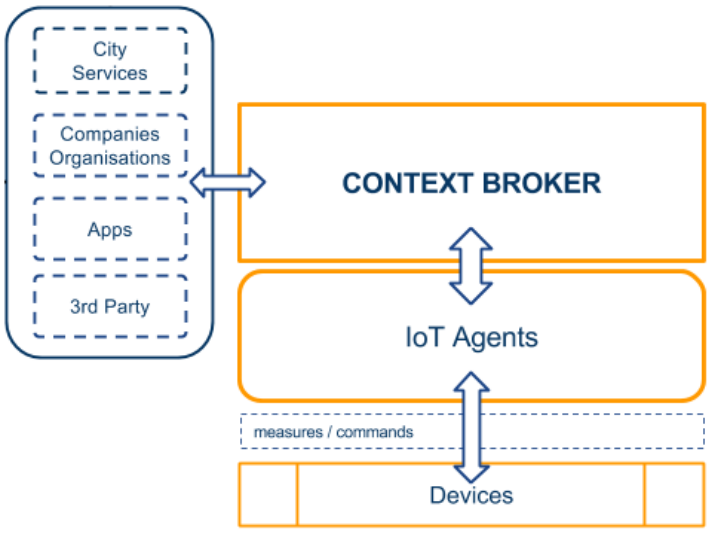


Figura 14 Flujo de información entre OCB y IoT Agent.

3.4.3 Componentes de software complementarios

Estos componentes son necesarios para el funcionamiento de los componentes de la plataforma FIWARE.

MongoDB[16]

Como almacén de datos se debe utilizar MongoDB, el cual es un sistema de gestión de bases de datos de tipo no-SQL. Este gestor es utilizado por el Orion Context Broker y el IoT Agent MQTT-JSON como almacén de datos, por lo tanto, se vio como una buena alternativa a utilizar para el desarrollo de ProximiThings Server debido a sus características. Estas características se muestran en el Apéndice 2.

Mosquitto[17]

Mosquitto es un es un servidor MQTT que permite el envío y recepción de mensajes en este protocolo. Mosquitto es un complemento al IoT Agent MQTT-JSON debido a que recibe directamente los mensajes MQTT de los dispositivos y se los transfiere al IoT Agent, el cual hace las validaciones y el mapeo correspondiente.

[16] MongoDB, documentación y descarga: https://www.mongodb.com/es
[17] Mosquitto: https://mosquitto.org/

3.4.4 Componentes de ProximiThings Server

ProximiThings Server está formado por tres componentes o módulos: P*roxemics Data Conversor, User Interaction Rules* y *Proxemics Action*. Adicionalmente se incluyó un módulo cliente MQTT para el envío de mensajes en el módulo *Proxemics Action*. ProximiThings Server tiene un API RESTful como interfaz para establecer las reglas de interacción de *User Interaction Rules* y las acciones de *Proxemics Action*. También permite a los desarrolladores utilizar las dimensiones proxémicas para sus propios sistemas de IoT compatibles con FIWARE.

Proxemics Data Conversor

Este módulo del ProximiThings Server permite que los datos de las dimensiones proxémicas sean convertidas de unidades continuas a discretas. Además, es el modulo que está directamente conectado al Orion Context Broker para recibir las actualizaciones en las mediciones de las dimensiones proxémicas.

User Interaction Rules

Este módulo permite que un usuario pueda establecer reglas de interacción con base en las mediciones de las dimensiones proxémicas obtenidas del *Proxemics Data Conversor.* A partir de las mediciones se pueden disparar acciones o comandos definidos en el módulo Proxemics Action. Cada comando puede tener múltiples reglas de interacción. Una regla de interacción es un subconjunto de posibles valores que puede tener una o más dimensiones proxémicas. Por ejemplo, si quisiéramos mostrar información reservada a una persona (p.e. Luis Carlos) en una pantalla, podemos establecer reglas de interacción como los de la **Tabla 1**.

Tabla 1 Reglas de interacción entre dos entidades

Entidad 1		**Entidad 2**	
Identidad	Display 1	**Identidad**	Luis Carlos
Orientación	Frente a: Luis Carlos	**Orientación**	Frente a: Display 1
Movimiento	- (indistinto)	**Movimiento**	- (indistinto)
Distancia	Intima	**Distancia**	Intima
Ubicación	Pasillo 1	**Ubicación**	Pasillo 1
Fase de Interacción	Interacción directa	**Fase de Interacción**	Interacción Directa
Comando	Mostrar mensajes a Luis Carlos en Display 1		

Proxemics Action

Cuando una regla de interacción proxémica definida por el usuario se cumple, se ejecuta un comando almacenado en este módulo. Un comando puede ser de dos tipos:

1) Un mensaje a transmitir por medio de MQTT a los dispositivos de Internet de las Cosas.
2) Un *callback* HTTP (*webhook*) con el comando y los valores de las dimensiones proxémicas de las entidades.

El mensaje MQTT permite indicarle a un dispositivo una función a ejecutar. Cuando un comando es de tipo *callback* HTTP, *Proxemics Action* hace una llamada HTTP a la URL indicada, y a través del método POST, envía los datos de la interacción proxémica realizada. El *callback* HTTP permite incorporar ProximiThings Server a otros servicios en la nube como por ejemplo IFTTT, Facebook, Twitter, Dropbox, etc.

API REST

A través del API RESTful, los desarrolladores podrán incorporar en sus sistemas de IoT, la capacidad para obtener y procesar información de interacción proxémica de ProximiThings Server. Este API tiene soporte de las operaciones CRUD (*Create, Retrive, Update, Delete*) para las reglas de interacción (User Interaction Rules), las acciones a ejecutar (Proxemics Action) y consultar la información de cada uno de los dispositivos del entorno de IoT.

Este API RESTful proporciona las respuestas en formato JSON. Los desarrolladores también pueden crear sus propias interfaces para mostrar información relevante según las necesidades de sus sistemas. Los métodos soportados por el API REST se muestran en la

Tabla *2*:

Tabla 2 Recursos disponibles en el API REST de ProximiThings Server

Método	PATH	Descripción
GET	/v1/rules	Muestra las reglas de interacción proxémica que están siendo monitoreadas actualmente para la ejecución de comandos.
POST	/v1/rules	Crea una nueva regla de interacción proxémica.
POST	/v1/rules/{RuleID}	Actualiza una regla de interacción proxémica.
DELETE	/v1/rules/{RuleID}	Elimina una regla de interacción proxémica
GET	/v1/actions	Muestra las acciones y comandos a ejecutar.
POST	/v1/actions	Crea una nueva acción
POST	/v1/actions/{ActionID}	Actualiza una acción o comando a ejecutar en un dispositivo de IoT.
DELETE	/v1/actions/{ActionID}	Elimina una acción o comando a ejecutar en un dispositivo de IoT.
GET	/v1/entities	Enlista las entidades de las que el sistema procesa las interacciones proxémicas
POST	/v1/entities	Da de alta un nuevo dispositivo en ProximiThings Server y en Orion Context Broker
POST	/v1/entities/{EntityID}	Actualiza la información proxémica de una entidad proporcionando su ID.
DELETE	/v1/entities/{EntityID}	Elimina la entidad con el ID proporcionado
GET	/v1/proxemics-history	Enlista el historial de interacción proxémica. Agregando las variables GET fecha y EntityID, podemos filtrarlos por fecha y/o entidad.

3.5 Procesamiento de las dimensiones proxémicas

De los dispositivos obtenemos las mediciones de las dimensiones proxémicas en unidades continuas. Estas medidas son convertidas a unidades discretas como en la **¡Error! La autoreferencia al marcador no es válida.**.

Tabla 3 Conversión de unidades continuas a discretas de las dimensiones proxémicas

Dimensión proxémica	Unidades continuas	Unidades discretas
Distancia	Centímetros	Zonas proxémicas: • Íntima • Personal • Social • Pública
Orientación	Grados sexagesimales	Orientación • Frente • Lado izquierdo • Lado derecho • Detrás
Movimiento	Centímetros	• En Movimiento • Pasivo
Distancia y Orientación	Centímetros y grados sexagesimales	Fases de interacción • Interacción directa • Interacción Sutil • Interacción Indirecta • Interacción ambiental

3.5.1 Procesamiento de la distancia

Para el caso específico de la distancia, las medidas discretas a utilizar son las zonas proxémicas descritas en capítulos anteriores. Para eso, se utiliza la distancia para establecer la zona proxémica en la que una persona esta con respecto a un dispositivo utilizando los rangos de la **Tabla 4**.

Tabla 4 Límites de las zonas proxémicas en centímetros

Límite inferior	Límite superior	Zona proxémica
0 cm	45 cm	Íntima
46 cm	121 cm	Personal
122 cm	365 cm	Social
366 cm	762 cm o más	Pública

3.5.2 Procesamiento de la orientación

Tenemos dos dispositivos A y B para un objeto y una persona respectivamente. Se quiere obtener el ángulo relativo γ entre estos dos dispositivos utilizando la siguiente fórmula de la suma de vectores unitarios:

$$G_x = \alpha_x + \beta_x$$
$$G_y = \alpha_y + \beta_y$$
$$\gamma = \tan^{-1}\frac{G_y}{G_x}$$

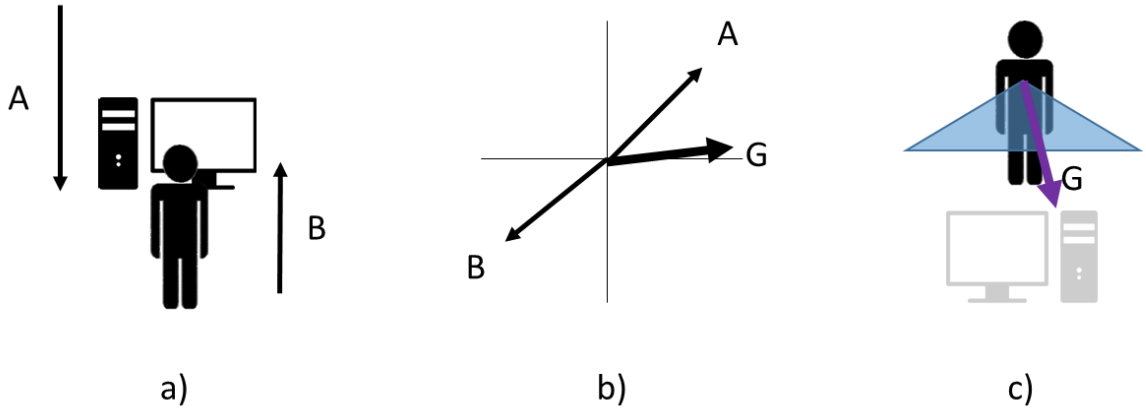


Figura 15 Inferencia de la orientación relativa. a) Entidades de frente b) Vectores de las entidades vistas en un plano c) Vector resultante de la suma de los vectores unitarios de cada entidad

En la **Figura 15** se muestra como se realiza el cálculo de vectores unitarios según la orientación de cada entidad. Para saber qué posición tiene una entidad con respecto a la otra, podemos establecer rangos del ángulo del vector resultante γ como en la **Tabla 5**.

Tabla 5 Límites de la orientación para inferir la posición u orientación relativa de una entidad con respecto a otra

Límite inferior	Límite superior	Orientación relativa
315° (-45°)	45°	Frente
46°	135°	Derecha
136°	225°	Detrás
226° (-135°)	315° (-45°)	Izquierda

3.5.3 Procesamiento del movimiento

Para el caso del movimiento, se va a utilizar la variación de la distancia para saber si una entidad está en movimiento. Por ello, ProximiThings Server guardará la distancia de la medición anterior y la comparará con una nueva. Si la distancia nueva es igual a la anterior, entonces se infiere que la entidad no se ha movido. Si son diferentes, entonces se infiere que está en movimiento.

3.5.4 Procesamiento de las fases de interacción de Vogel y Balakrishnan

Para el caso de las fases de interacción, es necesario utilizar los valores de la distancia y la orientación ya discretizados. Adicionalmente, se valida que las entidades están en el mismo lugar. Los valores que debe tener la distancia y la orientación para cada fase se muestran en la **Tabla 6.**

Tabla 6 Valores de la distancia y orientación para las fases de interacción

Distancia	Orientación	Fase de interacción
Íntima	Frente	Personal
Personal	Frente	Sutil
Social	Izquierda, Derecha, Frente	Implícita
Pública, *(sin información)*	Detrás, *(sin información)*	Ambiental

Si no se tienen información de la orientación o la distancia, pero se sabe dos entidades que están en el mismo lugar, entonces se infiere que ambas entidades tienen al menos la fase de interacción ambiental.

3.6 Flujo de información

3.6.1 Flujo del dispositivo a ProximiThings Server

El flujo de la información desde los dispositivos hasta ProximiThings Server es como se muestra en la **Figura 16** y las funciones y datos se describen a continuación.

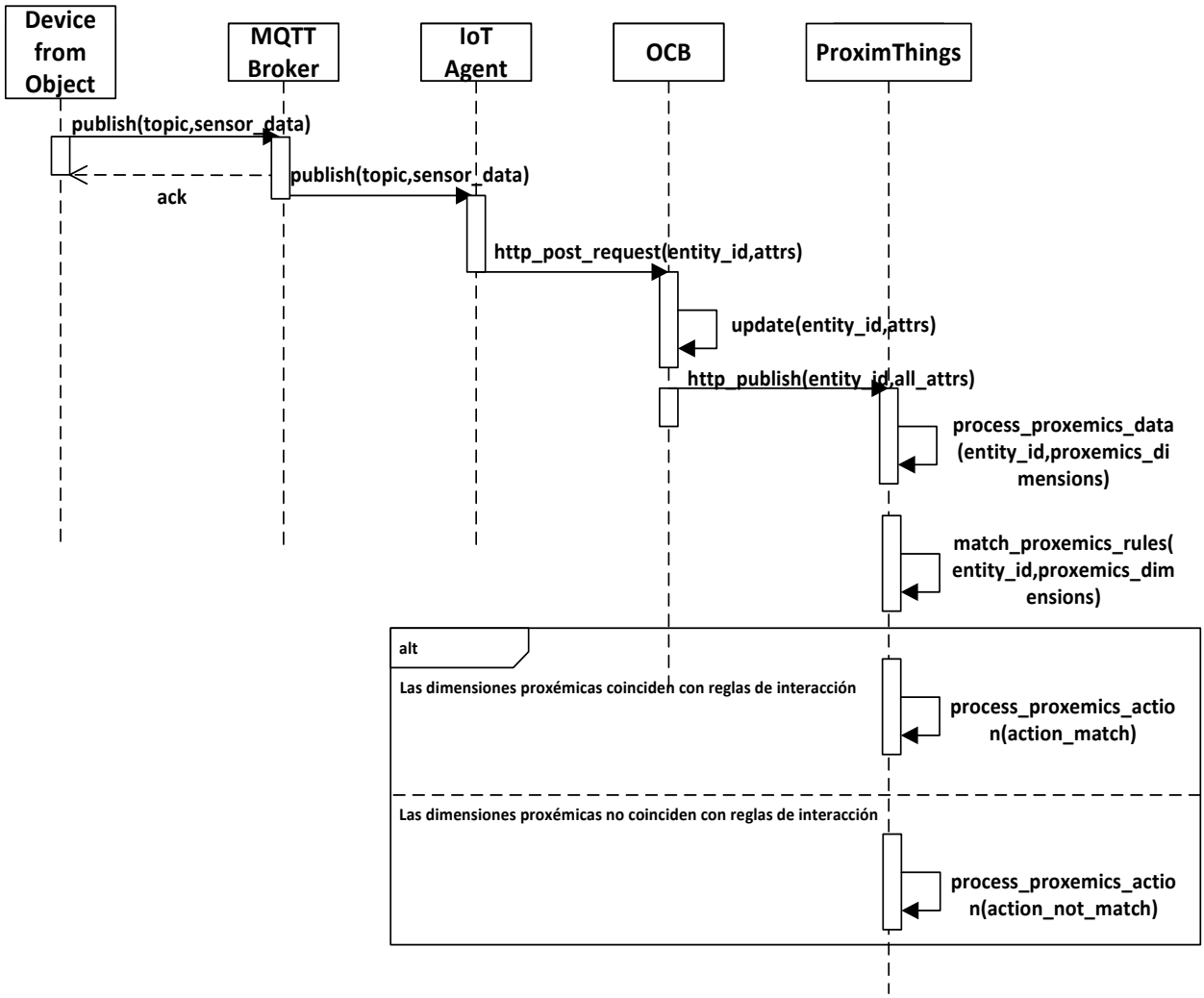


Figura 16 Diagrama de secuencia que representa el flujo de información proxémica desde el dispositivo hasta ProximiThings Server

- **publish(topic,sensor_data) desde device:** En este momento, se hace un publish al MQTT bróker con la información de los datos de sensores de las mediciones de las dimensiones proxémicas. En la publicación se especifica el tópico del bróker MQTT al que se van a publicar los datos y los datos de los sensores en formato JSON. Cada atributo del formato JSON tiene un nombre corto. El topic se debe obtener del IoT Agent.
- **publish(topic,sensor_data) desde MQTT Broker:** El bróker envia los datos del dispositivo al IoT Agent. Esto debido a que el IoT Agent hizo una suscripción a los tópicos de los dispositivos dados de alta.
- **http_post_request(entity_id,attrs) desde IoT Agent:** El IoT Agent envía los datos al Orion Context Broker después de procesarlos. Como el IoT Agent recibe los datos de los sensores con otros nombres de parámetros, estos datos deben ser mapeados a los nombres de los parametros

originales en el Orion Context Broker. Después de mapearlos, se obtiene el ID de la entidad en el Orion Context Broker y el IoT Agent realiza la actualización de los valores en el Orion Context Broker. En esta actualización se incluye un atributo de tipo *timestamp* con la fecha y hora de la última actualización recibida en el IoT Agent de los dispositivos. Para conocer a detalle cómo funciona el IoT Agent, puede consultar la documentación en fiware.org.

- **update(entity_id,attrs) desde OCB:** El Orion Context Broker realiza una actualización de los datos en su base de datos. Cambia el valor de los atributos que recibe en el request.
- **http_publish(entity_id,attrs) desde OCB:** OCB detecta que hubo una actualización en las entidades y envía una notificación a los suscriptores con los nuevos datos. En este caso, la notificación es para ProximiThings Server.
- **process_proxemics_data(entity_id,proxemics_dimensions) desde ProximiThings Server:** En este momento, ProximiThings Server procesa los datos obtenidos del Orion Context Broker para convertirlos en unidades discretas. Si la entidad a actualizar no existe, ProximiThings Server puede crear una nueva entidad y procesar la información proxémica. El desarrollador en el API puede cambiar o agregar datos según sea necesario.
- **match_proxemics_rules(entity_id,proxemics_dimensions) desde ProximiThings Server:** En este momento, ProximiThings Server verifica si las dimensiones proxémicas ya discretizadas coinciden con reglas dadas de alta previamente. Para hacer la petición de estas reglas, solamente se toma en cuenta las reglas donde la entidad está involucrada.
- **process_proxemics_action(action_match) desde ProximiThings Server:** Si las dimensiones proxémicas de la entidad actualizada coinciden con las reglas de interacción que ya han sido dadas de alta, entonces esta función ejecuta la acción definida en las reglas de interacción.
- **process_proxemics_action(action_not_match) desde ProximiThings Server:** Si las dimensiones proxémicas de la entidad actualizada no coinciden con las reglas de interacción que ya han sido dadas de alta, entonces esta función ejecuta la acción definida en las reglas de interacción para cuando las reglas no coincidan.

3.7 Organización del código de ProximiThings Server

ProximiThings Server fue desarrollado utilizando NodeJS como plataforma y el código es completamente abierto, el repositorio se puede encontrar en Github[18].

Como es de código abierto, cualquier desarrollador puede descargarlo, modificarlo e implementarlo en su propia plataforma. El código se compone de las siguientes carpetas como lo muestra la **Figura 17**.

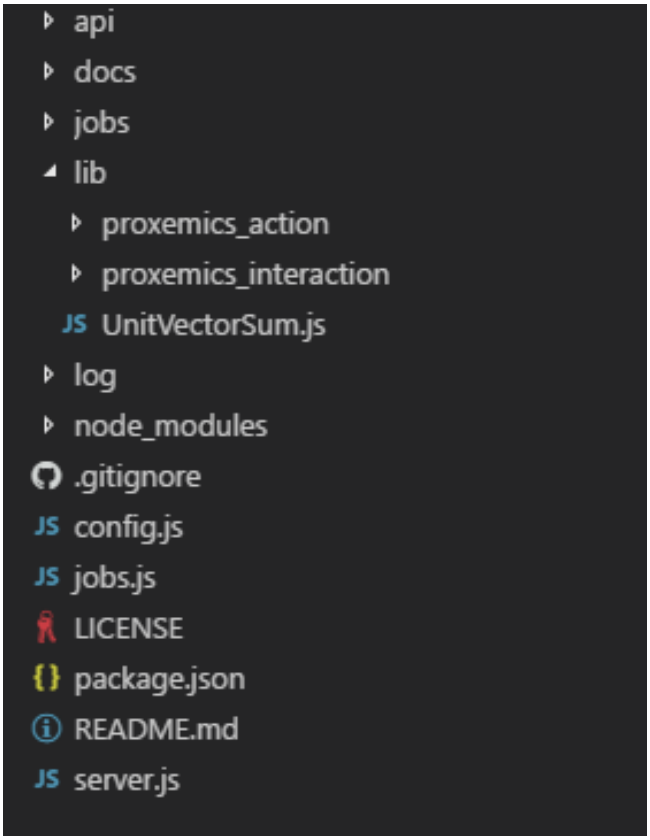


Figura 17 Organización de los archivos de ProximiThings Server

- **api:** En esta carpeta, se encuentra el código respectivo para el funcionamiento del API REST de ProximiThings Server, principalmente la configuración de las rutas HTTP, los controladores y modelos de la base de datos.
- **docs:** Esta carpeta contiene ejemplos de implementación del API REST de ProximiThings Server que también puedes acceder a ellos desde Github Pages[19].

[18] https://github.com/faxterol/ProximiThings-Server
[19] https://faxterol.github.io/ProximiThings-Server/

- **jobs:** En esta carpeta se encuentran las funciones que permiten procesar la información proxémica y las acciones en segundo plano utilizando colas de servicio.
    - **proxemics_action:** Aquí se encuentran las funciones para procesar las acciones a ejecutar cuando las reglas de interacción se cumplen.
    - **proxemics_interaction:** En esta carpeta, se encuentran las funciones para procesar las dimensiones proxémicas. Para definir el orden en que se van a ejecutar las funciones, ver el archivo `config.js`.
- **lib:** Esta carpeta es una biblioteca de funciones para ProximiThings Server. En esta se encuentran las funciones para procesar las dimensiones proxémicas y las acciones.
- **log:** Carpeta que contiene el log de acceso HTTP a ProximiThings Server.
- **node_modules:** Contiene las bibliotecas de NodeJS.
- **config.js:** Es un archivo de configuración de ProximiThings Server en el que se especifica que puerto es en el que va a estar disponible el API REST de ProximiThings Server y datos de conexión a la base de datos en MongoDB, conexión al Orion Context Broker, conexión al bróker MQTT y un arreglo con el orden en que se van a ejecutar las funciones para procesar las dimensiones proxémicas. En este arreglo, cada elemento debe coincidir con el nombre del archivo dentro de `lib/proxemics_interaction`.
- **jobs.js:** En este archivo se inicializa el procesamiento en segundo plano. Se establece la librería a utilizar, la conexión a la base de datos que va a guardar la cola de procesos y los procesos a ejecutar en segundo plano.
- **server.js**: Este es el archivo principal para ejecutar ProximiThings Server.

3.8 Extensión de funcionalidades de ProximiThings Server

ProximiThings Server es un sistema abierto y que puede ser utilizado por cualquier persona para su propio entorno de IoT. ProximiThings Server fue desarrollado y diseñado para que cualquier desarrollador pueda extender sus funcionalidades de diversas maneras.

Supongamos que un dispositivo de IoT envía la distancia en pulgadas. ProximiThings Server hasta este momento procesa las zonas proxémicas en centímetros, por lo que un desarrollador puede cambiar o agregar el procesamiento de las zonas proxémicas en pulgadas.

Por otra parte, supongamos que un dispositivo envía los datos en una cadena con diferentes valores separados por coma, podemos agregarle funcionalidades a ProximiThings Server para que procese esa cadena y descomponerla en sus valores y atributos para procesar las dimensiones proxémicas.

Existen casos en donde el uso de hardware es inherente a ProximiThings Server. Supongamos que un dispositivo de IoT que desarrolló una persona utiliza un sensor de tipo LIDAR para medir la distancia en centímetros. ProximiThings Server no necesita cambiar ninguna parte del código para soportar este tipo de sensores. Por lo que se puede usar cualquier tipo de hardware para medir las dimensiones proxémicas siempre y cuando se envíen los datos de la forma en que ProximiThings Server ya está programado.

3.8.1 Agregar nuevas formas de procesar las dimensiones proxémicas

Para agregar nuevas formas de procesar la interacción proxémica, es necesario crear un nuevo archivo javascript en la ruta `lib/proxemics_interaction`. En esta misma ruta se encuentra el archivo `BlankProxemicsDimension.js` con el código inicial del archivo. Este archivo recibe como parámetro el objeto de la entidad de la que se recibió la notificación. Este parámetro tiene dos objetos: `last_notification` y `proxemics_data`. La primera guarda la información de la última notificación recibida del OCB y `proxemics_data` guarda la interacción proxémica procesada hasta el momento en que el archivo nuevo fue ejecutado.

Una vez programado este archivo, hay que darlo de alta en `config.js`. En este archivo se encuentra un arreglo y en el parámetro `config.interaction_processing` agregamos al arreglo el nombre del archivo (sin extensión). Si necesitamos algún orden específico, entonces ubicar en el arreglo antes o después de que archivo hay que ejecutar el nuevo.

3.9 Modelos de datos del API de ProximiThings Server

Antes de empezar a utilizar el API de ProximiThings Server, es necesario conocer los modelos de datos utilizados en el gestor de base de datos. Es fundamental conocer estos modelos porque cuando se vaya a realizar una actualización en el API REST de ProximiThings Server, en la petición se debe incluir una estructura de datos en formato JSON basado en el modelo de datos según el recurso que se vaya a actualizar. Así mismo, las respuestas de estos recursos están en este mismo modelo de datos.

En la **Figura 18** tenemos el modelo de datos para las entidades. En la **Figura 19** está el modelo de datos para las reglas de interacción proxémica. En la **Figura 20** tenemos el modelo de datos para las acciones a ejecutar en las reglas de interacción proxémica y en la **Figura 21** tenemos el modelo de datos del historial de interacción proxémica.

Figura 18 Modelo de datos de una entidad

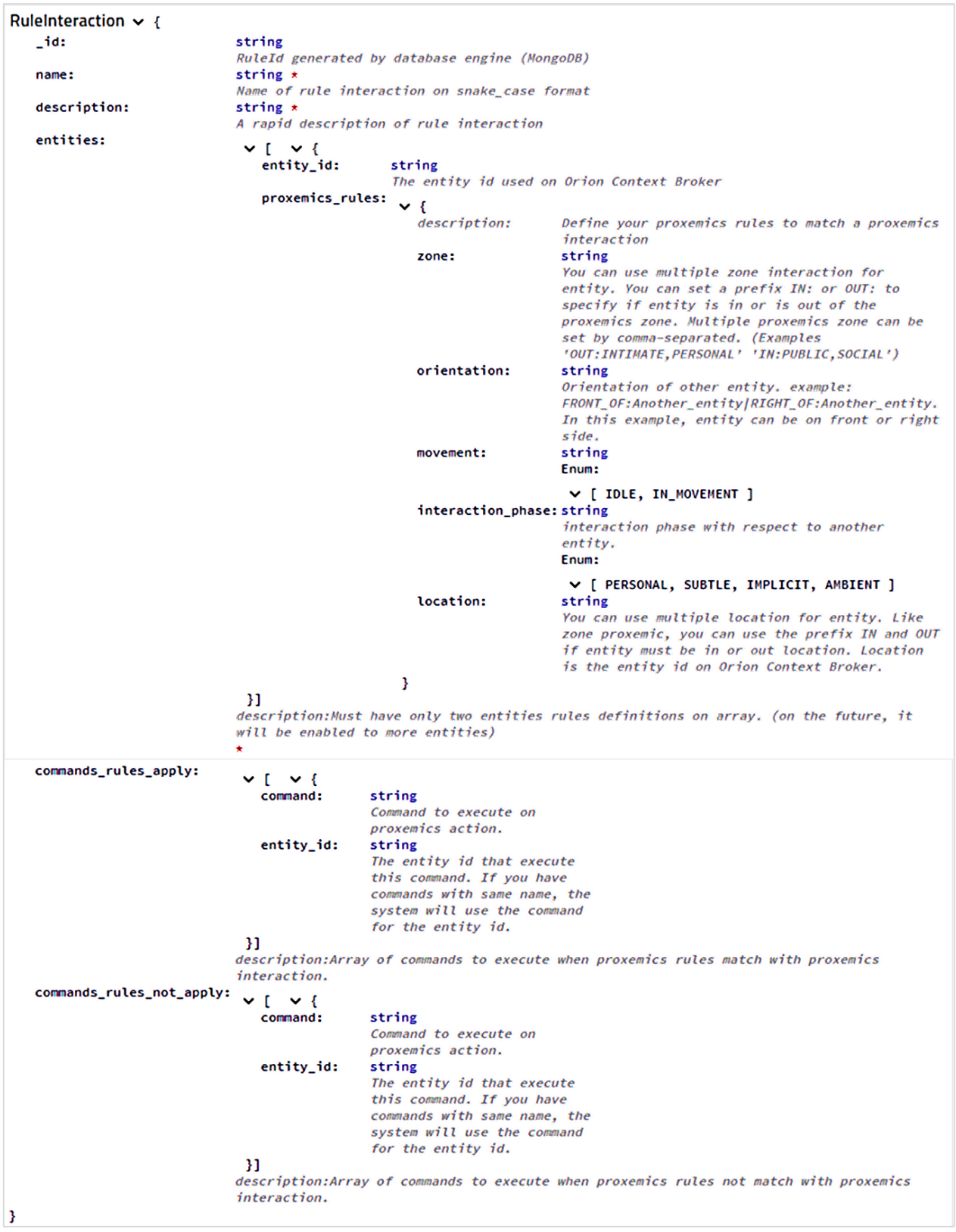


Figura 19 Modelo de datos de las reglas de interacción proxémica

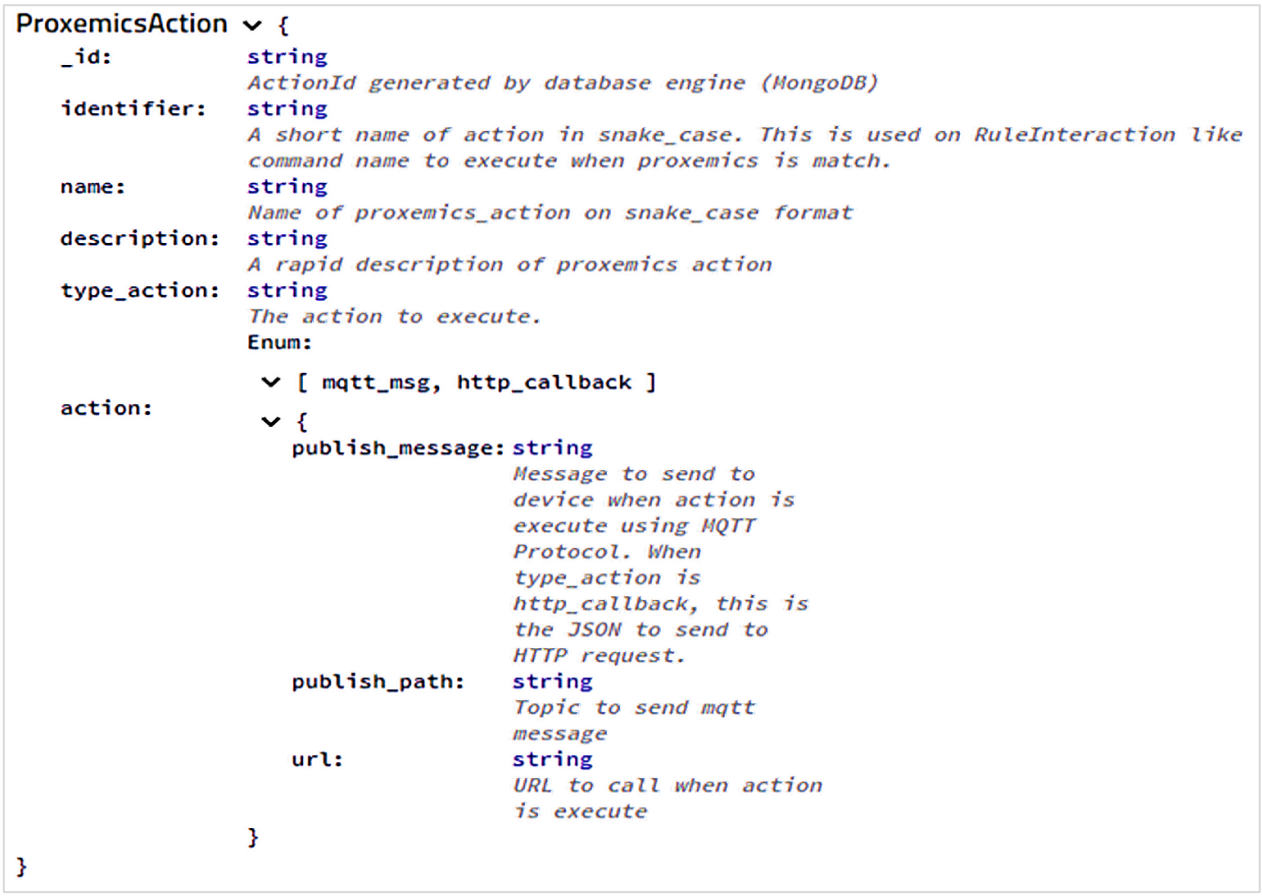


Figura 20 Modelo de datos de una acción

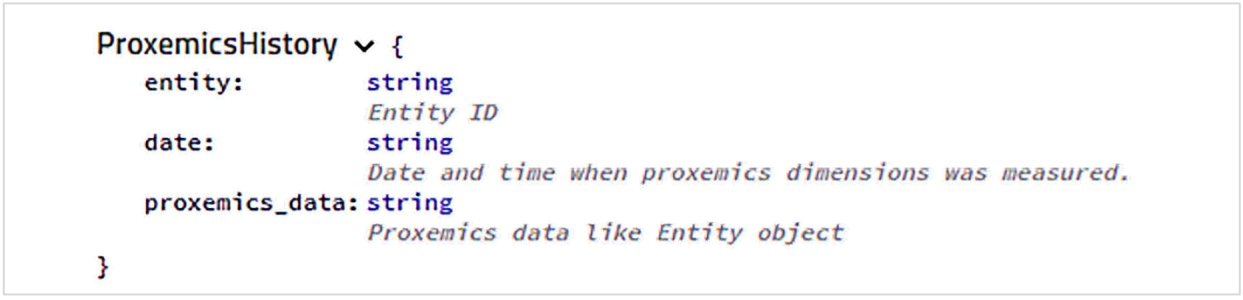


Figura 21 Modelo de datos del historial de interacción proxémica

Capítulo 4. Implementación y pruebas de concepto

En este capítulo se muestra como se implementó ProximiThings Server en un entorno de Internet de las Cosas para realizar las pruebas de concepto en tres escenarios. Estas pruebas de concepto involucraron el desarrollo de 4 dispositivos para medir las dimensiones proxémicas. Además, se describe cómo un usuario desarrollador implementó ProximiThings Server en su propio entorno de IoT.

4.1 Implementación de requerimientos de ProximiThings Server

En el capítulo anterior se describió cuáles son los requerimientos de ProximiThings Server y la arquitectura cliente-servidor que se necesita implementar. Para implementar ProximiThings Server en la nube del FIWARE Lab, es necesario crear una cuenta y solicitar recursos computacionales al INFOTEC[20] que es la institución que coordina todo lo relacionado con FIWARE en México.

Por otro lado, para implementar ProximiThings Server en su propia infraestructura o en su computadora, se deben instalar todos los componentes del FIWARE Platform necesarios para el funcionamiento de ProximiThings Server. Más información para la instalación de los componentes y GEs en la página web de FIWARE[21].

ProximiThings Server fue desarrollado utilizando el entorno de ejecución de NodeJS, por lo que también es necesario instalarlo. Una vez instalado NodeJS, ya puede instalar ProximiThings Server en su servidor o en el FIWARE Lab. Para instalar y configurar ProximiThings Server, hay que seguir las instrucciones proporcionadas en el archivo README.md del repositorio de ProximiThings Server en Github[22].

4.2 Visualización de la información de ProximiThings Server

El API REST de ProximiThings Server puede incorporarse a otras herramientas y desarrollos propios o de terceros. En este caso desarrollamos ProximiThings Front-end, el cual es una interfaz gráfica para visualización de la información de las dimensiones proxémicas entre los diferentes dispositivos del entorno de Internet de las Cosas. El objetivo principal de esta interfaz es ver, de manera gráfica, como van

[20] http://lanif.infotec.mx/solicitarcuenta.php.
[21] http://fiware.org
[22] http://github.com/faxterol/ProximiThings-Server

cambiando con el tiempo las mediciones de las dimensiones proxémicas y la conversión de las medidas continuas a discretas. Esta interfaz gráfica, que se muestra en la **Figura 22**, despliega en pantalla la información de dos formas:

1) Por medio de tarjetas para cada dimensión proxémica proporcionando los valores continuos y discretos en cada entidad.
2) Por medio de un radar en el que se muestra un dispositivo central y los demás dispositivos alrededor de él.

ProximiThings Front-end fue desarrollado utilizando la interfaz de Material Design[23] y el framework Angular 2 utilizando Javascript, el cual facilita la adopción del API RESTful directamente del servidor.

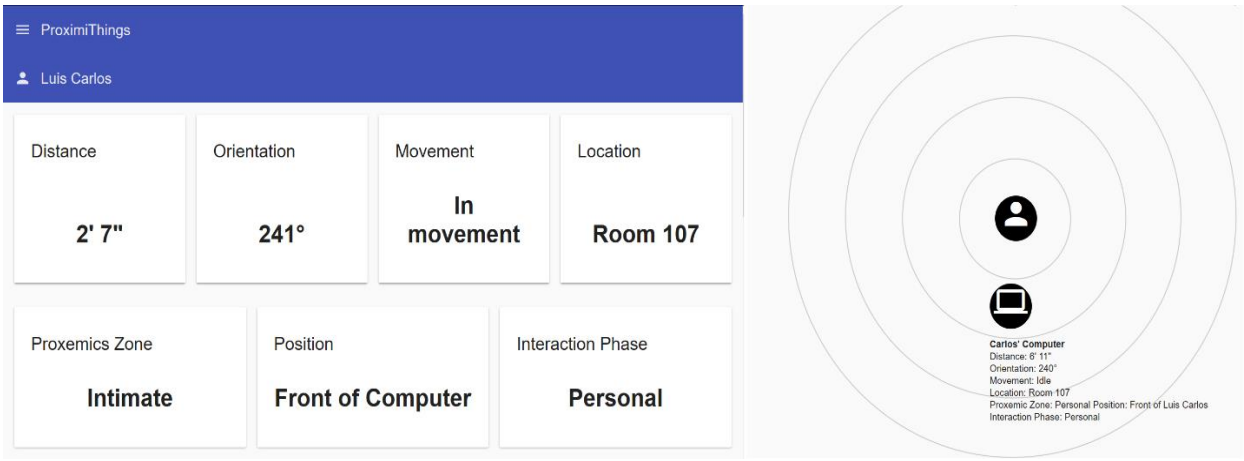


Figura 22 ProximiThings Front-end

4.3 Escenario General

Además de desarrollar ProximiThings Server, para las pruebas de concepto se desarrollaron algunos dispositivos de Internet de las Cosas para nuestro entorno. Estos dispositivos permiten medir una o más dimensiones proxémicas según sea necesario. Este escenario es como el de la **Figura 23**, y está compuesto de 4 tipos de dispositivos (ver **Figura 24**):

[23] AngularJS with Material Design: https://material.angularjs.org/latest/

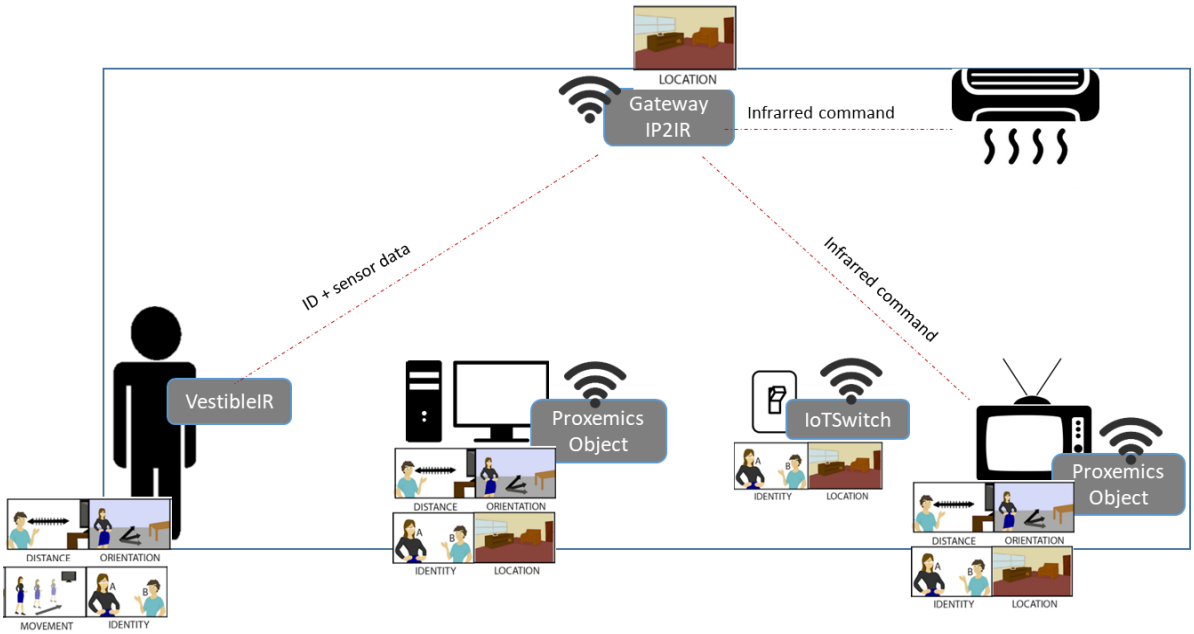


Figura 23 Entorno de prueba de ProximiThings Server

- **Gateway IR2IP:** Es un dispositivo fijo ubicado en el centro de un lugar o habitación. Tiene la capacidad de emitir y recibir señales en infrarrojo, lo que le permite enviar comandos a dispositivos que no están comunicados a internet para que ejecuten funciones en beneficio de las personas. La recepción de señales en infrarrojo le permite al dispositivo recibir información de otros dispositivos, principalmente del VestibleIR. Para poder enviar señales en infrarrojo, el servidor deberá decirle al dispositivo el comando a enviar. Para ello, el dispositivo se comunica al servidor por medio de WiFi utilizando el protocolo MQTT. De la misma manera, si el dispositivo recibe una señal en infrarrojo, esta deberá ser enviada al servidor por medio de la conexión WiFi. Por lo tanto, Gateway IR2IP es un Gateway que permite transformar señales en infrarrojo en paquetes IP (IP2IR). Se utiliza un receptor infrarrojo modelo TSOP38238[24]. Este dispositivo está pensado para medir la dimensión proxémica de la ubicación.

- **VestibleIR:** Este dispositivo está diseñado para usarse como cómputo vestible en las personas. Está equipado con sensores de acelerómetro y giroscopio que permite saber si la persona se está moviendo, magnetómetro para saber su orientación, sensor de distancia por ultrasonido para saber la distancia entre la persona y el objeto más cercano y un emisor infrarrojo. Este emisor infrarrojo envía la información de los sensores, así como el identificador de la persona al Gateway IP2IR o al ProxemicsObject. La estructura del mensaje está dividida en rangos de bits como se

[24] TSOP-38238: https://www.sparkfun.com/products/10266

muestra en la **Tabla 7**. Cada mensaje es de 32 bits enviados a una frecuencia de 38kHz. El Gateway IP2IR por su parte envía la información al servidor. Este dispositivo está pensado para medir las dimensiones proxémicas: identificación, movimiento, orientación y distancia.

Tabla 7 Estructura de bits que contiene el mensaje de infrarrojo

0 bit	1 – 9 bit	10 – 19 bits	20 – 31 bits
1 bit	9 bits	10 bits	12 bits
Libre (siempre cero)	Orientación	Distancia	ID infrarrojo de la persona

- **IoTSwitch:** Este dispositivo está diseñado para activar o desactivar otros dispositivos por medio de la interrupción del paso de energía eléctrica y su uso más sencillo es para el interruptor del alumbrado de un lugar. IoTSwitch está equipado con un relevador y conexión Wi-Fi para que el servidor envíe la señal de interrupción o paso de energía a este dispositivo. IoTSwitch está pensado para medir las dimensiones proxémicas: identidad y ubicación.

- **ProxemicsObject** Este dispositivo está diseñado para abstraer objetos con los que las personas van a interactuar constantemente. Este dispositivo está equipado con sensores para la medición de la distancia del objeto más cercano, acelerómetro y giroscopio para saber si el objeto está en movimiento, magnetómetro para conocer la orientación del dispositivo y un receptor de infrarrojo, en cuanto a la conectividad utiliza Wi-Fi para el intercambio de información con el servidor. Este dispositivo puede ser utilizado como un complemento para otros dispositivos electrónicos o no electrónicos. Este dispositivo permite medir las dimensiones proxémicas: identidad, localización, orientación y distancia.

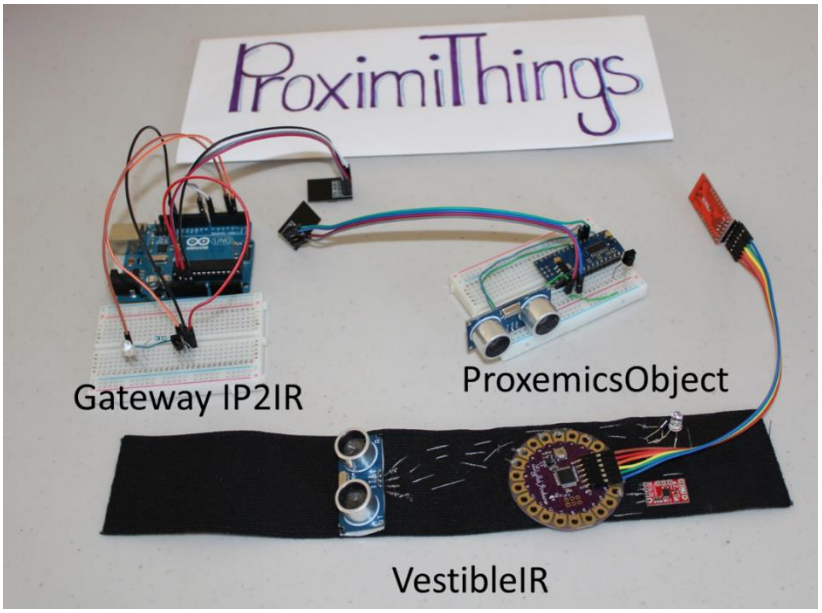


Figura 24 Dispositivos para el entorno de prueba

4.4 Escenarios de implementación para las pruebas de concepto

Con los dispositivos del escenario general, se plantea utilizarlos para los siguientes escenarios específicos.

4.4.1 Escenario 1

En el primer escenario se utilizan dos de los dispositivos presentados anteriormente, el VestibleIR y el ProxemicsObject. Como se muestra en la **Figura 25**, en el piso se marcaron las zonas proxémicas con respecto al ProxemicsObject. El VestibleIR se va a ir acercando al ProxemcisObject, pasando por cada una de las zonas desde la zona pública hasta la zona íntima. Vamos a observar los datos procesados por ProximiThings Server en ProximiThings Front-end, para verificar cómo van cambiando las mediciones conforme el tiempo. Con esto, se comprueba el funcionamiento de la emisión de señales en infrarrojo y la medición de la distancia en el dispositivo.

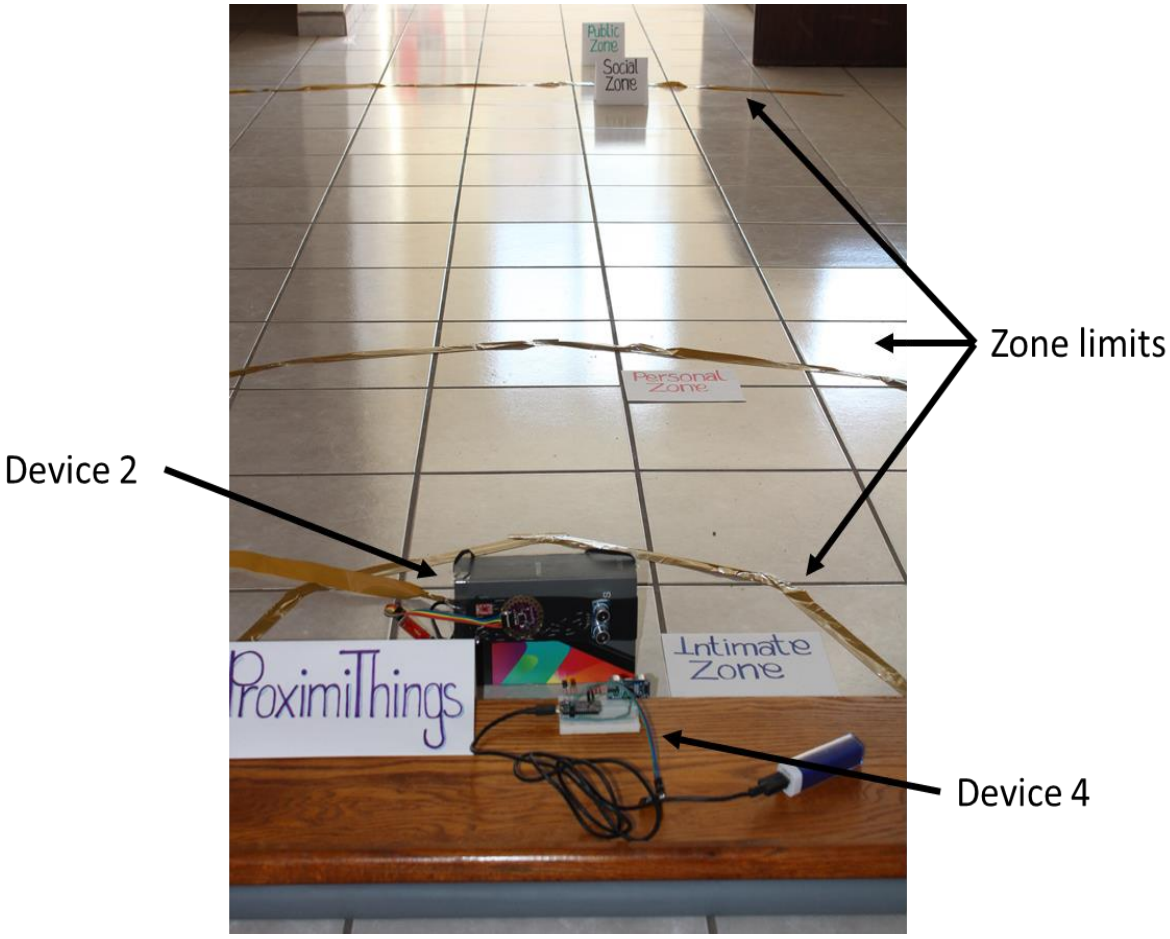


Figura 25 Escenario 1 de prueba

4.4.2 Escenario 2

Como segundo escenario, mostrado en la **Figura 26**, se utiliza también el VestibleIR y el ProxemicsObject. En este escenario, se quiere mostrar un mensaje de texto para una persona en específico. Este mensaje es privado, por lo que deberá mostrarse únicamente cuando el usuario correspondiente se encuentre en la zona proxémica íntima y frente al dispositivo. Si el usuario se aleja o le da la espalda al dispositivo el mensaje se ocultará.

Las reglas de interacción se utilizan para definir las condiciones específicas que deben cumplirse para que el mensaje sea mostrado y ocultado. Mientras que las acciones se usan para indicar al dispositivo que muestre y oculte el mensaje.

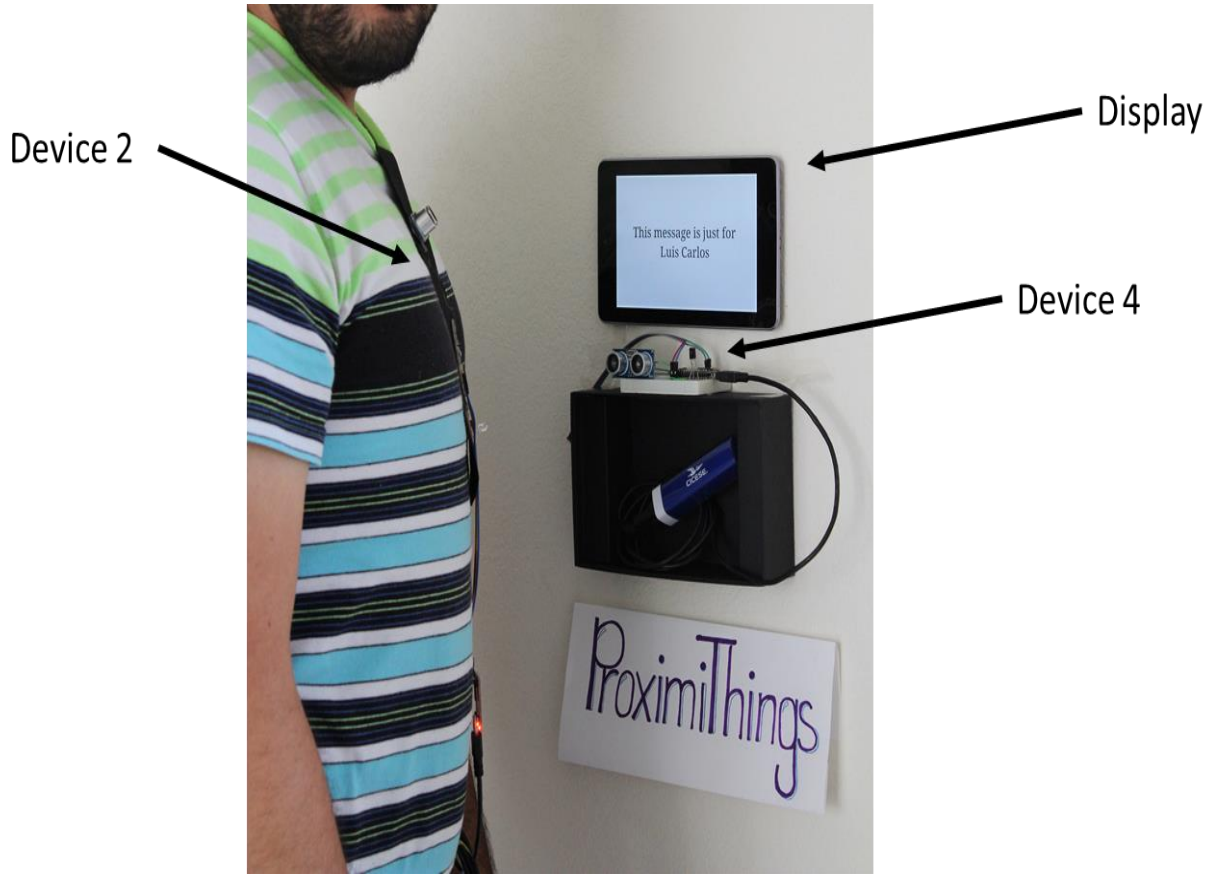


Figura 26 Escenario 2 de prueba

4.4.3 Escenario 3

En este escenario, como se muestra en la **Figura 27**, además del VestibleIR y del ProxemicsObject, se utiliza el IoTSwitch. El cual está representado con una cerradura electrónica y se incorpora al entorno de IoT que se está probando. En este caso se trata de una puerta que incluye una cerradura electrónica, las cual se abrirá o cerrará dependiendo de las interacciones de las personas. Cuando la persona se acerca lo suficiente para estar en la zona intima de la puerta, la cerradura abrirá. Mientras que si la persona se aleja volverá a cerrarse. Si una persona que no tiene permisos para abrir la puerta se acerca, la cerradura no abrirá.

Al igual que en el escenario 2, las reglas de interacción se utilizan para definir las condiciones específicas que deben cumplirse para que la cerradura se abra y cierre. Mientras que las acciones se usan para indicar

a la cerradura que abra o cierre. Para este escenario, vamos a implementar un protocolo de comunicación con la cerradura, llamado zigbee[25]

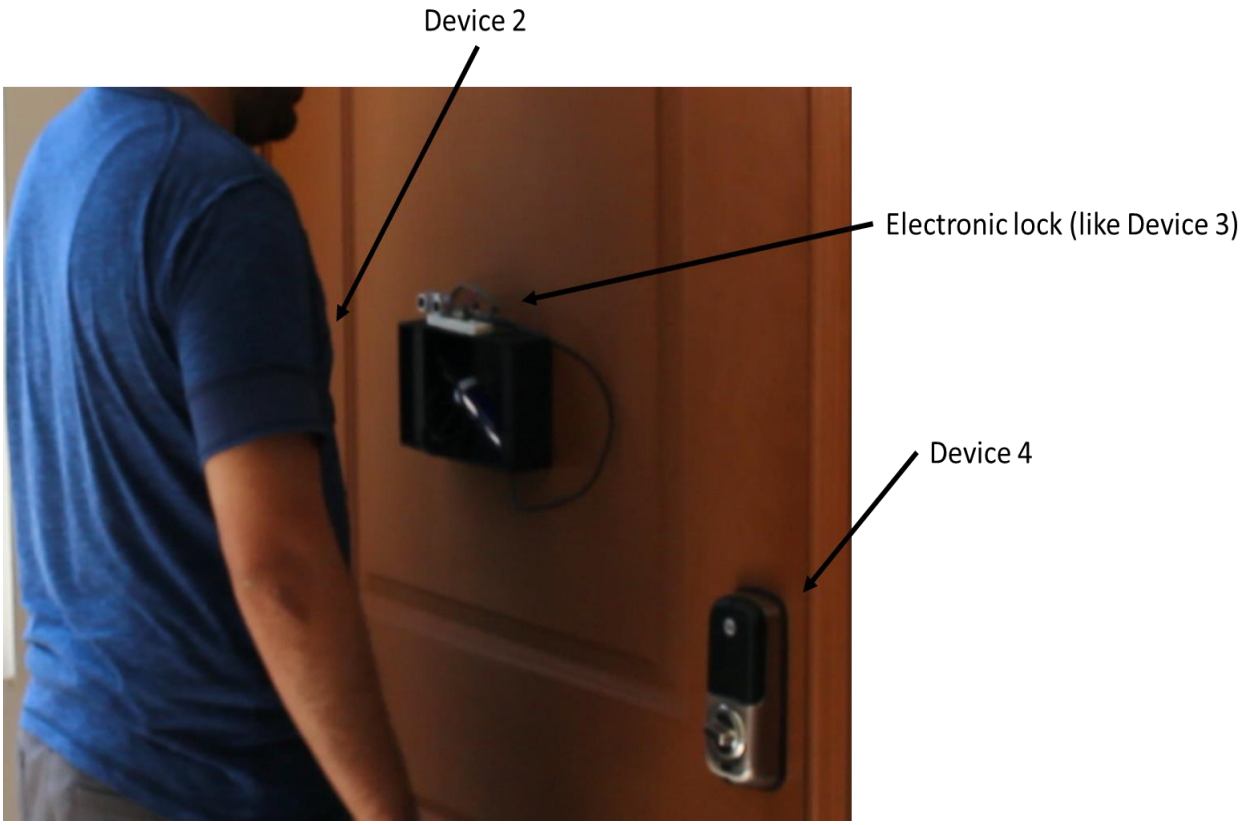


Figura 27 Escenario 3 de prueba

4.5 Implementación de los escenarios

En esta sección se va a describir cómo se implementó ProximiThings Server para el entorno de IoT de prueba que desarrollamos, partiendo del supuesto que el lector ya tiene instalado y configurado ProximiThings Server en su servidor o instancia en la nube. También se supone que los dispositivos envían los datos a ProximiThings Server a través del IoT Agent y el OCB. De lo contrario, se debe consultar el capítulo anterior o el repositorio de Github[26].

[25] Zigbee: http://www.zigbee.org
[26] ProximiThings Server repositorio de Github: http://github.com/faxterol/ProximiThings-Server

4.5.1 Dar de alta los dispositivos de IoT en el ProximiThings Server

Para dar de alta a los dispositivos, es necesario conectarnos al API REST de ProximiThings Server. Para hacerlo, podemos usar un cliente como Postman[27] o la línea de comandos con cURL[28]. En este caso dejaremos las peticiones como última opción.

En la **Figura 28** se muestran las peticiones a realizar en ProximiThings Server. La persona va a utilizar VestibleIR, en el capítulo anterior mostramos para qué es cada parámetro de la estructura de datos de la entidad. Sin embargo, para este escenario de prueba en `entity_attributes` agregamos un parámetro adicional que es el ID de la persona en código infrarrojo. Más adelante vamos a mostrar como extendimos la funcionalidad para procesar información en infrarrojo, pues necesitaremos de este parámetro.

En la misma figura mostramos como damos de alta la abstracción de un objeto para el *ProxemicsObject*. Este dispositivo no necesita parámetros adicionales. Por último, damos de alta el lugar. Esta entidad en ProximiThings Server podemos usarlo para el Gateway IP2IR y para hacer referencia al lugar en la dimensión proxémica de la ubicación en los demás dispositivos.

[27] Postman: http://getpostman.com
[28] cURL: https://curl.haxx.se/

```
curl -X POST \
  http://fiware.faxterol.com:1026/v1/entities \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: bc359a35-6893-7538-75e7-292d799d509c' \
  -d '{
    "name" : "LuisCarlos",
    "description" : "LuisCarlos entity for proximithings",
    "entity_id" : "LuisCarlos",
    "entity_type" : "PersonAbstraction",
    "entity_attributes" : {
        "infrared_entity_id" : "abc"
    },
    "service_path" : "/"
}
'
```

a)

```
curl -X POST \
  http://fiware.faxterol.com:1026/v1/entities \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: c7870a68-fe9a-34a3-bd92-4eed5eb567b8' \
  -d '{
    "name" : "NotifierDisplay",
    "description" : "",
    "entity_id" : " NotifierDisplay_ubilogix3",
    "entity_type" : "ObjectAbstraction",
    "service_path" : "/"
}
'
```

b)

```
curl -X POST \
  http://fiware.faxterol.com:1026/v1/entities \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 51fac14a-15c3-1f1a-12e4-a0b0c3ad780b' \
  -d '{
    "name" : "Ubilogix Dept 3",
    "description" : "",
    "entity_id" : "Ubilogix3",
    "entity_type" : "Location",
    "service_path" : "/"
}
'
```

c)

Figura 28 Dar de alta los dispositivos (entidades) en ProximiThings Server a) Persona b) Objeto (display y puerta) c) Lugar

4.5.2 Recibir información por infrarrojo

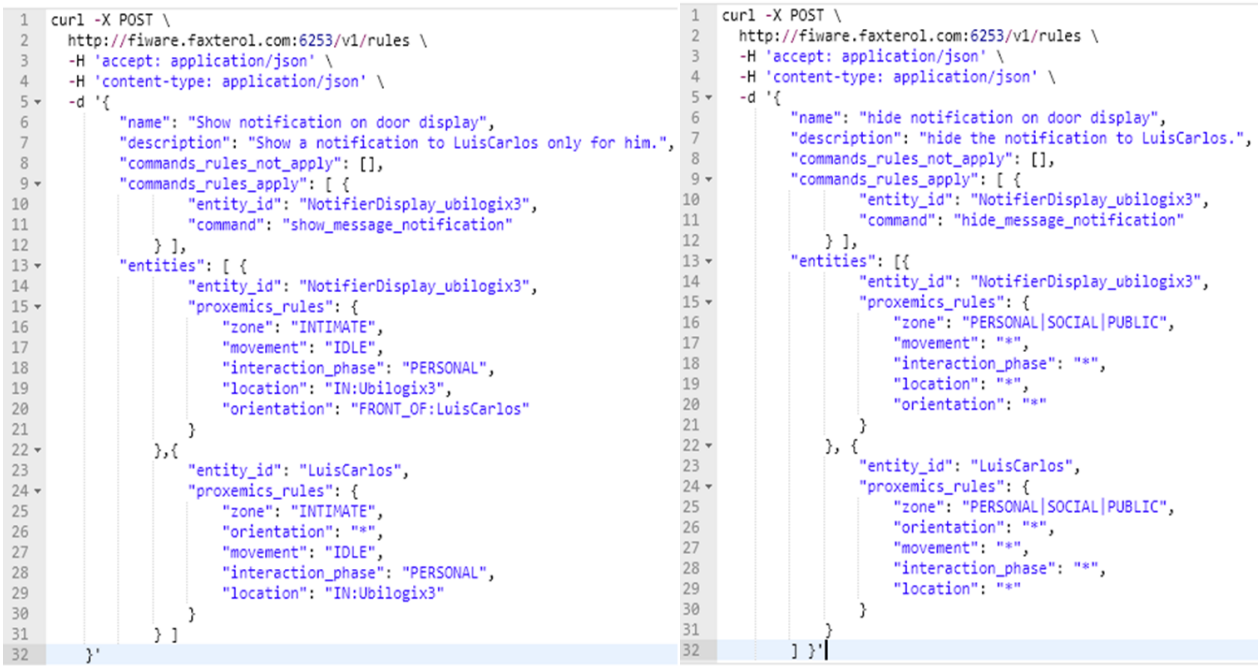
Como en el escenario general mencionamos que vamos a utilizar infrarrojo para recibir la información de la persona en el sistema de cómputo vestible, es necesario extender las funcionalidades de ProximiThings Server para procesarlo. Para ello, seguimos los pasos descritos en el capítulo anterior. El primer paso es crear el archivo `InfraredPerson.js` en la carpeta `lib/proxemics_interaction` de ProximiThings Server. Agregamos en el `config.js` el nombre del archivo y el orden para procesarlo y en el archivo `InfraredPerson.js` ingresamos el código fuente para procesar el infrarrojo. Como el infrarrojo tiene capacidad muy limitada en datos, entonces debemos crear un ID corto. Cuando dimos de alta a la persona, pusimos también el ID infrarrojo, por lo que cada vez que Gateway IP2IR o ProxemicsObject reciban un código infrarrojo, entonces van a procesar una actualización en la medición de las dimensiones proxémicas de la persona. El código fuente para procesar el infrarrojo es el que se muestra en la **Figura 29**. Para conocer las librerías utilizadas en ProximiThings Server, consultar el archivo `package.json` en la raíz del repositorio.

```
var mongoose = require('mongoose'), //Mongodb library
  Entity = mongoose.model('Entity'),  //Entity schema
  sv = require('../UnitVectorSum'),  //Unit Vector Sum function
  config = require('../../config'), //config file
  rp = require("request-promise"); //for http request
//infrared person
module.exports = function(entity) {
    //there is the "infrared_person" variable
    if(typeof entity.last_notification.infrared_person != "undefined"){
        /*infrared_person is an infrared command using 32 bits in this format:
        <12 bits for people id><10 bits for distance><9 bits for orientation>*/
        var ir = parseInt("0x"+entity.last_notification.infrared_person.value);

        var ir_base2 = ir.toString(2);
        var ir_bits = [];
        for(var i = 0;i<32-ir_base2.length;i++){
            ir_bits[i] = 0;
        }
        for(var i = 0;i<ir_base2.length;i++){
            ir_bits[32-ir_base2.length+i] = ir_base2[i];
        }

        var ir_person = parseInt(ir_bits.slice(20, 32).join(""),2).toString(16);
        var ir_distance = parseInt(ir_bits.slice(10, 20).join(""),2);
        var ir_orientation =  parseInt(ir_bits.slice(1, 10).join(""),2);

        var entity_ir = Entity.findOne({"entity_attributes.infrared_entity_id" : ir_person}).exec();
        entity_ir = entity_ir.then(function ( entity_query) {
            if(entity_query != null){
                entity.proxemics_data.orientation.in_front_of.push(entity_query.entity_id);
                console.log(JSON.stringify(entity.proxemics_data));
                entity.save();
                return entity_query;
            }else{
                throw new Error("Infrared entity ID not exists");
            }
        })
        .then(function(entity_query){
            return rp({
                method: 'POST',
                uri: 'http://'+config.orion.server+':'+config.orion.port+'/v2/entities/'+entity_query.entity_id+'/attrs',
                body: {
                    "distance_cm": {
                        "type": "integer",
                        "value" : ir_distance
                    },
                    "orientation_degrees" : {
                        "type" : "integer",
                        "value" : ir_orientation
                    },
                    "location_id" : {
                        "type" : "Location",
                        "value" : entity.proxemics_data.location_id
                    }
                },
                headers: {
                    "Content-type" : "application/json",
                    "Fiware-Service" : config.ProximiThings.service,
                    "Fiware-ServicePath" : entity_query.service_path,
                },
                json : true
            })
            .catch(function (err) {
                console.log("Failed to send Infrared data to OCB Entity:" + err);
                return entity_query;
            });
        })
        .catch(function(err){
            console.log(err);
        });

    }
    console.log("Infrarojo procesado");
    entity.markModified('proxemics_data');
    return entity.save();
};
```

Figura 29 Código fuente para procesar la información en infrarrojo desde ProximiThings Server

4.5.3 Reglas de interacción proxémica y acciones

Para los escenarios necesitaremos dar de alta las reglas de interacción proxémica y las acciones en el API REST de ProximiThings Server. Para el escenario 2, tenemos que dar de alta en el API REST las reglas de interacción que deben cumplir dos entidades para mostrar y ocultar el mensaje en la pantalla como se muestra en la **Figura 30**. Para cada una de las reglas, tenemos una acción que se debe ejecutar cada que estás se cumplan, como se muestran en la **Figura 31**. Para el escenario 3, tenemos que dar de alta en el API las reglas de interacción que deben cumplir las dos entidades para abrir y cerrar la puerta como se muestran las peticiones en la **Figura 32**. Del mismo modo, debemos dar de alta las acciones a ejecutar para cuando se cumpla cada una de las reglas, pero en este caso vamos a utilizar la acción del mensaje MQTT. Las peticiones para dar de alta las acciones se muestran en la **Figura 33**.



```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/rules \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
        "name": "Show notification on door display",
        "description": "Show a notification to LuisCarlos only for him.",
        "commands_rules_not_apply": [],
        "commands_rules_apply": [ {
                "entity_id": "NotifierDisplay_ubilogix3",
                "command": "show_message_notification"
            } ],
        "entities": [ {
                "entity_id": "NotifierDisplay_ubilogix3",
                "proxemics_rules": {
                    "zone": "INTIMATE",
                    "movement": "IDLE",
                    "interaction_phase": "PERSONAL",
                    "location": "IN:Ubilogix3",
                    "orientation": "FRONT_OF:LuisCarlos"
                }
            },{
                "entity_id": "LuisCarlos",
                "proxemics_rules": {
                    "zone": "INTIMATE",
                    "orientation": "*",
                    "movement": "IDLE",
                    "interaction_phase": "PERSONAL",
                    "location": "IN:Ubilogix3"
                }
            } ]
    }'
```

a)

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/rules \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
        "name": "hide notification on door display",
        "description": "hide the notification to LuisCarlos.",
        "commands_rules_not_apply": [],
        "commands_rules_apply": [ {
                "entity_id": "NotifierDisplay_ubilogix3",
                "command": "hide_message_notification"
            } ],
        "entities": [{
                "entity_id": "NotifierDisplay_ubilogix3",
                "proxemics_rules": {
                    "zone": "PERSONAL|SOCIAL|PUBLIC",
                    "movement": "*",
                    "interaction_phase": "*",
                    "location": "*",
                    "orientation": "*"
                }
            }, {
                "entity_id": "LuisCarlos",
                "proxemics_rules": {
                    "zone": "PERSONAL|SOCIAL|PUBLIC",
                    "orientation": "*",
                    "movement": "*",
                    "interaction_phase": "*",
                    "location": "*"
                }
            }
        ] }'
```

b)

Figura 30 Reglas de interacción para el escenario 2. a) Reglas para mostrar el mensaje b) Reglas para ocultar el mensaje

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/actions \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -d '{
        "identifier": "show_message_notification",
        "entity_id": "NotifierDisplay_ubilogix3",
        "name": "Show message to LuisCarlos",
        "description": "Show an special message to LuisCarlos",
        "type_action": "http_callback",
        "action": {
            "publish_message": "{\"value1\":1}",
            "url": "http://proximithings.faxterol.com/write.php"
        }
    }
'
```

a)

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/actions \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -d '{
        "identifier": "hide_message_notification",
        "entity_id": "NotifierDisplay_ubilogix3",
        "name": "Hide message to LuisCarlos",
        "description": "hide the special message to LuisCarlos",
        "type_action": "http_callback",
        "action": {
            "publish_message": "{\"value1\":0}",
            "url": "http://proximithings.faxterol.com/write.php"
        }
    }
'
```

b)

Figura 31 Acciones para ejecutar en el escenario 3. a) Cuando se cumpla la regla de mostrar b) Cuando se cumpla la regla de no mostrar

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/rules \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
        "name": "open door ubilogix3",
        "description": "open main door ubilogix3",
        "commands_rules_not_apply": [],
        "commands_rules_apply": [ {
                "entity_id": "NotifierDisplay_ubilogix3",
                "command": "open_main_door"
            }],
        "entities": [{
                "entity_id": "NotifierDisplay_ubilogix3",
                "proxemics_rules": {
                    "zone": "INTIMATE",
                    "movement": "FRONT_OF:LuisCarlos",
                    "interaction_phase": "PERSONAL",
                    "location": "IN:Ubilogix3",
                    "orientation": "*"
                }
            },{
                "entity_id": "LuisCarlos",
                "proxemics_rules": {
                    "zone": "INTIMATE",
                    "orientation": "FRONT_OF:NotifierDisplay_ubilogix3",
                    "movement": "*",
                    "interaction_phase": "*",
                    "location": "IN:Ubilogix3"
                }
            } ]
    }'
```

a)

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/rules \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
        "name": "close door ubilogix3",
        "description": "close main door ubilogix3",
        "commands_rules_not_apply": [],
        "commands_rules_apply": [{
                "entity_id": "NotifierDisplay_ubilogix3",
                "command": "close_main_door"
            }],
        "entities": [ {
                "entity_id": "NotifierDisplay_ubilogix3",
                "proxemics_rules": {
                    "zone": "*",
                    "movement": "*",
                    "interaction_phase": "*",
                    "location": "IN:Ubilogix3",
                    "orientation": "*"
                }
            },{
                "entity_id": "LuisCarlos",
                "proxemics_rules": {
                    "zone": "*",
                    "orientation": "*",
                    "movement": "*",
                    "interaction_phase": "*",
                    "location": "NOT_IN:Ubilogix3"
                }
            }]
    }'
```

b)

Figura 32 Reglas de Interacción para el escenario 3. a) Reglas para abrir la puerta b) Reglas para cerrar la puerta

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/actions \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: cc16460a-541a-daa4-66e9-7667c3f6c2f6' \
  -d '{
        "identifier": "open_main_door",
        "entity_id": "NotifierDisplay_ubilogix3",
        "name": "Open main door ublilogix 3",
        "description": "open the main door of ubilogix3.",
        "type_action": "mqtt_msg",
        "action": {
            "publish_message": "{\"access\":\"yes\"}",
            "publish_path": "agrsMZG/access"
        }
  }
'
```

a)

```
curl -X POST \
  http://fiware.faxterol.com:6253/v1/actions \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -d '{
        "identifier": "close_main_door",
        "entity_id": "NotifierDisplay_ubilogix3",
        "name": "close main door ublilogix 3",
        "description": "close the main door of ubilogix3.",
        "type_action": "mqtt_msg",
        "action": {
            "publish_message": "{\"access\":\"no\"}",
            "publish_path": "agrsMZG/access"
        }
    }
'
```

b)

Figura 33 Acciones para ejecutar en el escenario 3 a) Cuando se cumplan las reglas para abrir la puerta b) Cuando se cumplan las reglas para cerrar la puerta

4.6 Resultados y pruebas de concepto de la implementación de ProximiThings Server

Con la implementación de ProximiThings Server, pudimos comprobar la utilidad de la aplicación de las dimensiones proxémicas en un entorno de Internet de las Cosas. En el escenario 2, utilizamos las dimensiones proxémicas para mostrar un mensaje a un usuario, lo cual denota una utilidad en cuanto a la privacidad de información. De la misma forma con la puerta que tiene una utilidad para permitir el acceso a un lugar para una persona en específico.

ProximiThings Server ha sido desarrollado siguiendo el formato del *Orion Context Broker* para recibir notificaciones y actualizar información proxémica según lo proporcionado por la documentación[29]. ProximiThings Server puede formar parte de la plataforma FIWARE ya que con las notificaciones y las actualizaciones de información de contexto en *Orion Context Broker,* ProximiThings Server es un consumidor y proveedor de información de contexto[30].

De acuerdo con el repositorio de Github de la plataforma FIWARE[31], 22 de 75 repositorios con componentes de la plataforma FIWARE son desarrollados utilizando Javascript como lenguaje de programación. Esto significa que el 30% de los componentes utilizan este lenguaje, 70% restante utilizan Python, Java y C++.

[29] http://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html#subscriptions
[30] http://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/#context-management
[31] https://github.com/Fiware

Esto denota una preferencia por utilizar Javascript para desarrollar componentes de la plataforma FIWARE. ProximiThings Server fue desarrollado en el lenguaje de programación Javascript, utilizando como *runtime* NodeJS, pues NodeJS se define como una plataforma “diseñada para construir aplicaciones de red escalables”.

ProximiThings Server utiliza sensores electrónicos para la medición de las dimensiones proxémicas, mientras que otros trabajos de investigación no los utilizan. Sin embargo, no es la finalidad de estas pruebas de concepto comparar el rendimiento del procesamiento de las mediciones.

En la prueba de concepto encontramos que utilizar sensores tiene algunas limitaciones, uno de ellos es el uso del ultrasonido como forma de medir la distancia. Si tenemos dos sensores de ultrasonido midiendo la distancia al mismo tiempo, entonces las ondas de ultrasonido que rebotan entre los dos sensores no arrojan la medida correcta. Sin embargo, el uso de las zonas proxémicas a partir de la distancia, como parámetro para establecer las reglas de interacción permite disminuir el error.

Como se muestra en la **Figura 34**, cuando la persona se encuentra más alejada que el límite superior de la zona intima, la acción de mostrar el mensaje en pantalla no se ejecuta. Mientras que en la **Figura 35** se puede ver que si la persona se acerca lo suficiente para estar dentro del rango de distancia de la zona intima, el mensaje correspondiente se muestra en pantalla.

Figura 34 Medición de las dimensiones proxémicas previo a la ejecución de una acción (Escenario 2)

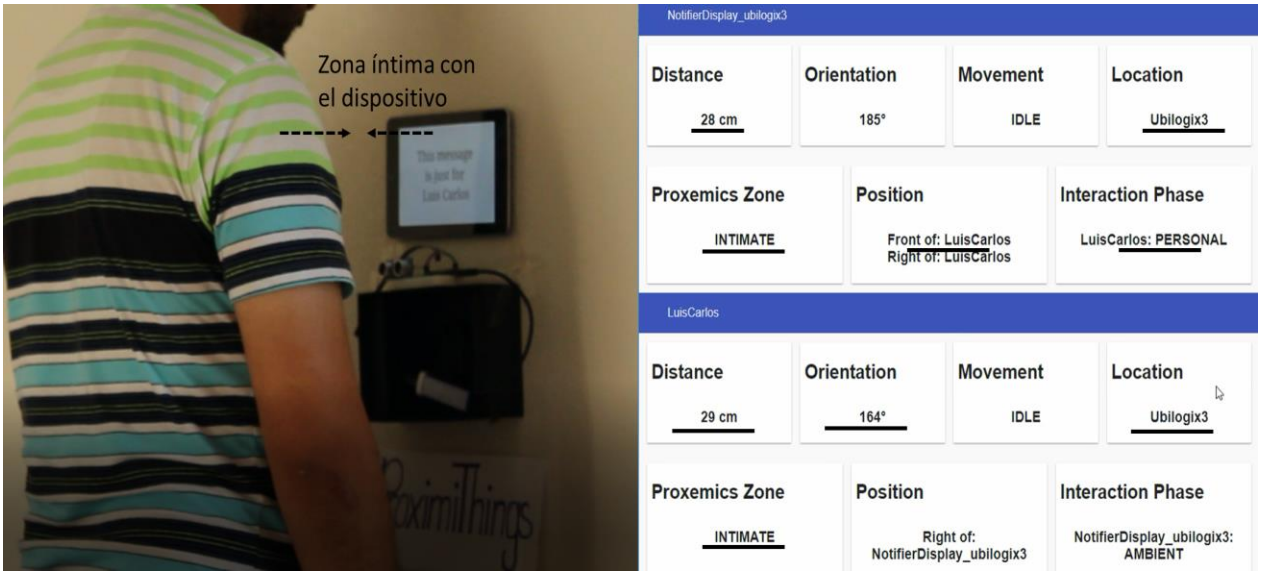


Figura 35 Medición de las dimensiones proxémicas luego de la ejecución de una acción

Otra limitación que encontramos es el uso de infrarrojo. Ya que durante las pruebas de concepto se tuvieron ciertos problemas con el receptor de infrarrojo, el cual no siempre recibía correctamente las mediciones. Por lo que es necesario explorar otra tecnología de implementación para el envío de la información proxémica de las personas a ProximiThings Server.

ProximiThings Server fue desarrollado con la idea de que cualquier desarrollador pueda extender sus funcionalidades agregando código para interpretar de otras formas las dimensiones proxémicas. De esta manera fue como se agregó el soporte para la recepción de datos en infrarrojo en ProximiThings Server. Con esto, pudimos comprobar que es posible extender las funcionalidades de ProximiThings Server a través de infrarrojo y utilizando otro formato de cadenas de texto o de bits. Si, por ejemplo, una persona quisiera extender las funcionalidades de ProximiThings Server para utilizar *beacons* con bluetooth que encapsulen los datos de las dimensiones proxémicas dentro del *beacon*, es posible hacerlo de la misma manera como se hizo con el infrarrojo.

De acuerdo con la literatura, para que un entorno de Internet de las Cosas sea centrado en las personas, los dispositivos deben tener soporte para que puedan ser utilizados con la mínima intervención de las personas (Miranda et al., 2015a). En el escenario 2, se muestra un mensaje personalizado para una persona con sólo acercarse a la pantalla y se oculta al alejarse o darle la espalda. La persona no necesita tocar la pantalla o hacer un gesto adicional para ver el mensaje. En el escenario 3, se permite el acceso a una persona a un lugar detectando que existe un acercamiento hacia la puerta. Para disminuir la

intervención de la persona con la puerta, utilizamos las dimensiones proxémicas para quitar el seguro a la puerta. De lo contrario, la persona tendría que ingresar la llave para quitar el seguro. Cuando se detecta que la persona ha pasado la puerta, esta se vuelve a bloquear para impedir el paso de personas no autorizadas.

En cuanto a las dimensiones proxémicas, ProximiThings Server convirtió de forma adecuada la distancia medida en la zona proxémica correspondiente. En el escenario dos y tres fue donde probamos la distancia y con la transición entre la zona íntima y personal se ejecutaron las acciones como se esperaba, como se muestra en la **Figura 36**.

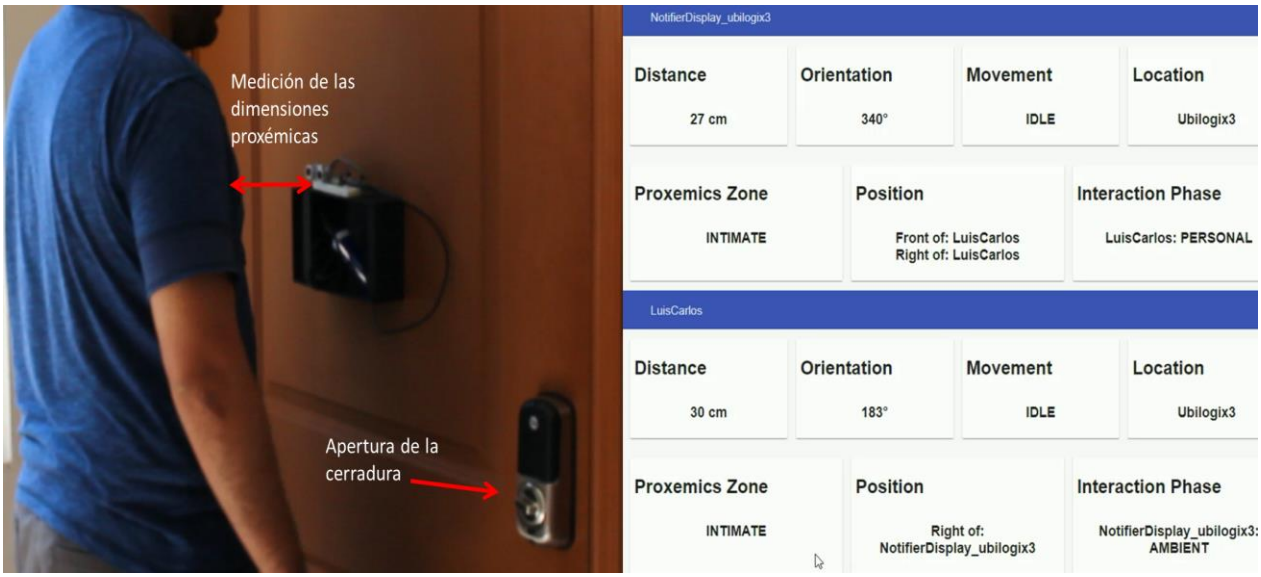


Figura 36 Medición y conversión de las dimensiones proxémicas en el escenario 3 para activar una cerradura electrónica

En cuanto a la orientación, el envío de información a través de infrarrojo permitió procesar la orientación para obtener sólo una posición “de frente”. Con el magnetómetro también se pudo encontrar la orientación de un dispositivo con respecto a otro. Sin embargo, se encontró que cuando se procesa la orientación, en los dos dispositivos de los escenarios 2 y 3 arroja la misma posición. En el caso del escenario 3, la orientación en grados sexagesimales indicaba que el dispositivo estaba a la izquierda de la persona, lo cual es incorrecto. El ángulo resultante es de 8 grados sexagesimales, que está en el rango de “en frente”, lo cual refuerza que el dispositivo está frente a la persona (ver **Figura 37**).

Estos errores encontrados, motivaron una revisión del código de ProximiThings Server para mejorar la inferencia de la orientación. Lo que generó mejoras en el funcionamiento correcto de ProximiThings Server en cuanto a la orientación de personas y dispositivos.

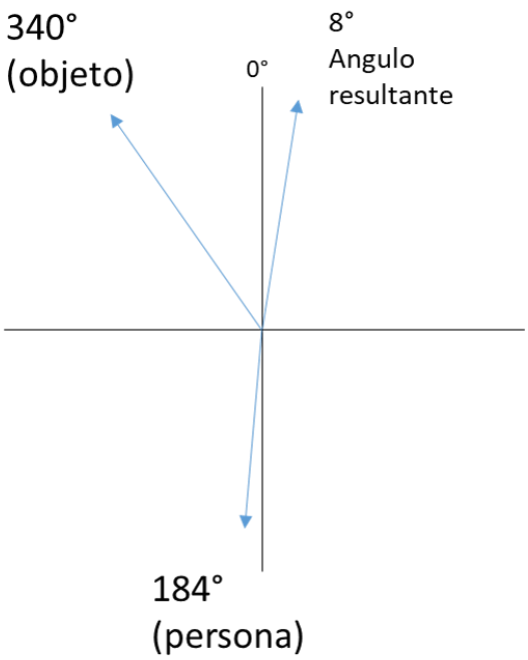


Figura 37 Inferencia de la posición del objeto con respecto a la persona utilizando la orientación del magnetómetro

Para el caso de la ubicación, se pudo obtener la ubicación de la persona con el uso del infrarrojo. Inicialmente, se estableció que si un dispositivo recibía la señal de infrarrojo de una persona, entonces se infería que estaban en el mismo lugar. En el escenario 3, con la interpretación de la ubicación, se pudo permitir al usuario tener acceso a la puerta cuando estuvieran en el mismo lugar.

En cuanto a las fases de interacción, la fase directa, personal y sutil utilizan la distancia y orientación. La conversión de estas fases se realizó correctamente. La fase de interacción ambiental por otro lado utilizó la dimensión proxémica de la ubicación.

Por último, las acciones que se ejecutaban cuando se cumplían las interacciones proxémicas se ejecutaban cada vez que se hacía una medición de las dimensiones proxémicas. Por lo que, si se hacían 10 mediciones en un minuto y una regla de interacción coincidía con una acción, entonces la acción se ejecutaba 10 veces. Esto implicó que, principalmente al incorporar acciones de tipo *webhook*, el servidor del servicio web en el que se hacía la petición bloqueaba peticiones futuras, debido a la saturación.

En general, los tres escenarios convirtieron correctamente las medidas continuas en discretas como se estableció en los requerimientos, así como con la extensión de funcionalidades a través de infrarrojo.

4.7 Uso de ProximiThings Server por desarrolladores

ProximiThings Server ya ha sido utilizado por desarrolladores en un entorno de Internet de las Cosas. En este entorno, se incorporaron otros dispositivos de IoT para modificar la temperatura del ambiente de un lugar a través de las reglas de interacción proxémicas y las acciones.

La implementación de ProximiThings Server en este entorno consisten en el rastreo del usuario a través de las dimensiones proxémicas, se detecta cuando un usuario entra a un espacio. Al momento de entrar, se compara la temperatura actual con la preferencia del usuario, establecida con anterioridad, y si la temperatura actual no coincide, entonces se activa un ventilador para refrescar el ambiente. Con la utilización de las acciones MQTT, se creó una aplicación para convertir mensajes de MQTT a Zigbee. Esto debido a que el protocolo Zigbee es el utilizado por los dispositivos en este entorno de IoT.

4.8 Integración de las pruebas de concepto

Con las pruebas de concepto, mostramos cómo puede funcionar ProximiThings Server en un entorno de Internet de las Cosas y cómo se puede beneficiar a las personas al utilizar los dispositivos con la mínima intervención.

A través de la interacción proxémica e infiriendo una intensión de uso hacia los objetos, podemos apoyar a que los adultos mayores utilicen estos objetos con la mínima intervención. Esto representa un beneficio, ya que los adultos mayores pueden presentar limitaciones físicas y mentales propias de su edad. Con esta finalidad, ProximiThings Server puede ser utilizado en un entorno de Internet de las Cosas de una casa de asistencia para adultos mayores. Si se infiere una intención de uso hacia la puerta para salir de la casa de asistencia por parte de un adulto mayor, se puede bloquear el seguro de la puerta y notificar a un cuidador sobre la situación actual.

De igual forma, el uso de la interacción proxémica puede ayudar a personas con debilidad visual para navegar y moverse en espacios interiores. Por ejemplo, si una persona quiere ir de una habitación a otra, se pueden habilitar las puertas para permitir su acceso, ayudarle a moverse evitando obstáculos y comunicar por voz a la persona qué objetos o personas se encuentran cerca, así como su orientación.

Otro entorno de IoT donde puede ser usado ProximiThings Server es en la manipulación de maquinaria dentro de una fábrica. Con las dimensiones proxémicas se puede revisar que un operador no este distraído en la operación de las maquinas a través de la orientación del cuerpo y la mirada del operador, así como la distancia entre el operador y la máquina. Adicionalmente, esto también puede ayudar a evitar que personas no autorizadas manipulen la maquinaria, o de ser necesario, desactivarlas completamente.

4.9 Conclusiones

En conclusión, ProximiThings Server funcionó como se esperaba en el procesamiento de las dimensiones proxémicas. También se extendieron las funcionalidades de ProximiThings Server con el uso de infrarrojo como forma de enviar información. De igual manera, las acciones se pudieron procesar adecuadamente y se logró activar otros dispositivos para beneficio de las personas utilizando otros protocolos de comunicación.

Surgieron algunos detalles durante la implementación, pero se pueden arreglar mejorando los dispositivos como el trabajo a futuro. ProximiThings Server puede ser agregado a la plataforma FIWARE para procesar las dimensiones proxémicas en un entorno de Internet de las Cosas.

Capítulo 5. Conclusión

En este capítulo se resumen las conclusiones obtenidas a partir del desarrollo de este trabajo de investigación, las contribuciones hechas y las limitaciones resultantes. También se discute el trabajo futuro que puede inspirar o derivar de este trabajo.

5.1 Conclusiones

ProximiThings Server mostró que es posible procesar las dimensiones proxémicas en un entorno de Internet de las Cosas y utilizar las reglas de interacción proxémica para disminuir la intervención de las personas en la utilización de los dispositivos. Con las reglas de interacción u el disparo de acciones con respecto a estas reglas, se cubren el trabajo futuro propuesto por OpenThings (Calderon et al., 2016b).

De igual forma, se mostró que es posible la incorporación de nuevas funcionalidades para procesar mediciones de las dimensiones proxémicas a ProximiThings Server por otros medios. ProximiThings Server puede ser incorporado a las plataformas de IoT de los desarrolladores para procesar las interacciones proxémicas siempre y cuando se envíen los datos a ProximiThings Server como los necesita actualmente o extendiendo sus funcionalidades como se especifica en los capítulos anteriores.

5.2 Aportaciones

Este trabajo de investigación realiza sus aportes en el campo del Internet de las Cosas, principalmente en el aspecto del monitoreo de las intenciones de usuario a través de la medición de las dimensiones proxémicas, un tema que ha sido poco abordado desde que se propuso como un reto de investigación por (Satyanarayanan, 2001).

Este trabajo también realiza sus aportes en la medición de las dimensiones proxémicas en el campo del Internet de las Cosas, así como el uso de sensores agregados a los dispositivos para realizar las mediciones de las dimensiones proxémicas, pues en otros trabajos de investigación en el cómputo ubicuo (Ballendat, Marquardt, y Greenberg, 2010; Marquardt et al., 2011) se realizan las mediciones utilizando visión por

computadora. Con el uso de sensores en los objetos se sigue la visión orientada a los objetos en el Internet de las Cosas.

Este trabajo de investigación aporta al desarrollo de funcionalidades para interacción proxémica en entornos de IoT para la plataforma *SmartSDK*[32], que es una plataforma basada en FIWARE. Este trabajo también aporta conocimientos para que otros autores se interesen en hacer investigación y desarrollo en el ámbito del Internet de las Cosas centrado en las personas.

Por último, como contribución a la comunidad científica, se publicó el artículo "*ProximiThings: Implementing Proxemics Interactions in the Internet of Things*" que se presenta en el *8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks* y que, al momento de escribir este trabajo de tesis, está aceptado y pendiente de publicar (Garcia-Macias y Cardenas, 2017).

5.3 Limitaciones

ProximiThings Server es un sistema es un sistema que solo procesa las interacciones proxémicas a través de la medición de las dimensiones proxémicas de cada entidad, por lo tanto, carece de algunos aspectos que las plataformas de Internet de las Cosas tienen.

La primera limitación de ProximiThings Server es la carencia de un módulo de permisos, autenticación y autorización de usuarios para la configuración de ProximiThings Server y los dispositivos del entorno de Internet de las Cosas donde puede ser aplicado.

Otra de las limitaciones es que este sistema está centrado en usuarios desarrolladores, por lo que es necesario adaptar al sistema de tal forma que pueda ser utilizado por los usuarios finales de un entorno de Internet de las Cosas desarrollando paneles de control u otras formas de configuración a través de una interfaz gráfica. De igual manera, con el análisis de los grandes datos generados por ProximiThings Server se pueden proponer algunas reglas de interacción para activar dispositivos y así el usuario pueda activarlos.

[32] SMARTSDK https://www.smartsdk.eu/

De la misma forma, ProximiThings Server necesita de un mecanismo para la identificación de lugares, pues hasta este momento los dispositivos necesitaban ser configurados manualmente con la ubicación en donde se encontraban. Si los dispositivos se movían de lugar, entonces se debía definir nuevamente en la configuración.

Por último, el hardware que se utilizó para probar ProximiThings Server mostró muchas limitaciones en cuanto a capacidades de almacenamiento en memoria RAM, uno de ellos es el procesamiento de emisión de códigos en infrarrojo para activar o desactivar dispositivos. Otra de las limitaciones fue con el magnetómetro, que no proporcionaba correctamente la orientación.

5.4 Trabajo futuro

Como trabajo futuro, se propone que ProximiThings Server se incorpore al ecosistema de componentes de la plataforma FIWARE para Internet de las Cosas como un DSE (Domain Specific Enabler) para el procesamiento de las dimensiones proxémicas. En todo caso, también puede incorporarse al ecosistema de SmartSDK para el procesamiento de las dimensiones proxémicas en entornos de IoT.

ProximiThings Server es un desarrollo en fase beta, por lo que se propone seguir realizando pruebas en entornos de IoT que permitan mejorar su funcionamiento. Esto puede ser realizado por parte de la comunidad científica del país o por cualquier persona en la comunidad Open Source, pues este proyecto y su código es libre.

Debido a los problemas presentados con el hardware, como trabajo futuro también se propone hacer un desarrollo especializado de hardware para medir mejor las dimensiones proxémicas. Así como utilizar otro tipo de sensores para medir las dimensiones proxémicas y realizar una comparativa práctica entre los sensores y poder discriminar cual es mejor.

En ProximiThings Server es necesario establecer las reglas de interacción para accionar dispositivos. Con la finalidad de que el usuario final no tenga que hacer esto, se propone que, como trabajo futuro, se incorporen funcionalidades de análisis de datos en big data o inteligencia artificial. Esto permitirá saber qué reglas proxémicas se aplican cada vez que un usuario utiliza algún dispositivo y que se le propongan al usuario estas reglas proxémicas para así disparar beneficios de forma proactiva. Con esto se permite mejorar la capacidad de reacción de los dispositivos de IoT a las intenciones de usuario.

También como trabajo futuro se propone que las dimensiones proxémicas sean medidas en todas las direcciones. Principalmente medir la distancia desde una entidad a las demás entidades.

ProximiThings Server también puede apoyar en sistemas de IoT orientados a contexto, la plataforma FIWARE tiene algunos componentes que pueden ser implementados para la inferencia de contexto. De igual forma, a futuro, se pueden extender las funcionalidades de ProximiThings Server para que las dimensiones proxémicas procesen información de contexto más detallado acerca de las intenciones de usuario. Por ejemplo: en el movimiento se puede extender la inferencia a otros datos como “caminando hacia”, “saliendo de” o “entrando a un lugar”.

Para hacer ProximiThings Server más centrado en las personas, se propone que, como trabajo futuro, se puedan utilizar las variables espacio-temporales para restringir o acceder al uso de dispositivos de IoT así como el disparo de funcionalidades proactivas guiadas por estas variables. Y en un contexto más amplio, hacer que los dispositivos también sean parte del modelo social (*Social IoT*) con las personas para guiar las restricciones y permisos de uso de los dispositivos por parte de las personas y la comunicación entre dispositivos.

Para los desarrolladores, se propone que, como trabajo a futuro, se realice una evaluación de usabilidad para conocer como fue la adopción de ProximiThings Server y el API REST con sus entornos de Internet de las Cosas.

Literatura citada

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., y Steggles, P. 1999. Towards a Better Understanding of Context and Context-Awareness (pp. 304–307). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-48157-5_29

Ashton, K. 2009. That "Internet of Things" Thing. RFIDJournal. Recuperado a partir de http://www.rfidjournal.com/articles/view?4986

Atzori, L., Iera, A., y Morabito, G. 2014. From "smart objects" to "social objects": The next evolutionary step of the internet of things. IEEE Communications Magazine, 52(1), 97–105. https://doi.org/10.1109/MCOM.2014.6710070

Bâce, M., Sörös, G., Staal, S., y Corbellini, G. 2017. HandshakAR. En *Proceedings of the 8th Augmented Human International Conference on - AH '17* (pp. 1–5). New York, New York, USA: ACM Press. https://doi.org/10.1145/3041164.3041203

Ballendat, T., Marquardt, N., y Greenberg, S. 2010. Proxemic interaction: Designing for a Proximity and Orientation-Aware Environment. En *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*. New York, New York, USA: ACM Press. https://doi.org/10.1145/1936652.1936676

Botta, A., de Donato, W., Persico, V., y Pescapé, A. 2016. Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems, 56, 684–700. https://doi.org/10.1016/j.future.2015.09.021

Calderon, M. A., Delgadillo, S. E., y Garcia-Macias, J. A. 2016a. A more human-centric Internet of Things with temporal and spatial context. Procedia Computer Science, 83(Ant), 553–559. https://doi.org/10.1016/j.procs.2016.04.263

Calderon, M. A., Delgadillo, S. E., y Garcia-Macias, J. A. 2016b. A More Human-centric Internet of Things with Temporal and Spatial Context. Procedia Computer Science, 83(Ant), 553–559. https://doi.org/10.1016/j.procs.2016.04.263

Delgadillo, S. E. 2015. Plataforma semántica para internet de las cosas. Tesis de Maestría. Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.

Ebling, M. R. (IBM T. J. W. R. C.). 2016. Revisiting Satya's Vision and Challenges. IEEE Pervasive Computing. Recuperado de: https://www.computer.org/csdl/mags/pc/2016/02/mpc2016020002.pdf

Garcia-Macias, J. A., y Cardenas, C. 2017. ProximiThings: Implementing Proxemic Interactions in the Internet of Things. 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks.

Gartner. 2013. Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. Recuperado de http://www.gartner.com/newsroom/id/2636073

Gil, D., Ferrández, A., Mora-Mora, H., y Peral, J. 2016. Internet of Things: A Review of Surveys Based on Context Aware Intelligent Services. Sensors, 16(7), 1069. https://doi.org/10.3390/s16071069

Greenberg, S., y Kuzuoka, H. 1999. Using digital but physical surrogates to mediate awareness, communication and privacy in media spaces. Personal and Ubiquitous Computing, 3(4), 182–198. https://doi.org/10.1007/BF01540552

Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., y Wang, M. 2011. Proxemic Interactions: The New Ubicomp? Interactions, 18(1), 42. https://doi.org/10.1145/1897239.1897250

Gubbi, J., Buyya, R., Marusic, S., y Palaniswami, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Hall, E. T. 1966. The Hidden Dimension. New York: Doubleday.

Kortuem, G., Kray, C., y Gellersen, H. 2005. Sensing and visualizing spatial relations of mobile devices. En *Proceedings of the 18th annual ACM symposium on User interface software and technology UIST 05* (pp. 93–102). https://doi.org/10.1145/1095034.1095049

Kosmatos, E. A., Tselikas, N. D., y Boucouvalas, A. C. 2011. Integrating RFIDs and Smart Objects into a UnifiedInternet of Things Architecture. Advances in Internet of Things, 1(1), 5–12. https://doi.org/10.4236/ait.2011.11002

Li, S., Xu, L. Da, y Zhao, S. 2010. The internet of things: a survey. Information Systems Frontiers, 17(2), 243–259. https://doi.org/10.1007/s10796-014-9492-7

Li, S., Xu, L. Da, y Zhao, S. 2015. The internet of things: a survey. Information Systems Frontiers, 17(2), 243–259. https://doi.org/10.1007/s10796-014-9492-7

Marquardt, N., Diaz-Marino, R., Boring, S., y Greenberg, S. 2011. The proximity toolkit. En *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11* (p. 315). New York, New York, USA: ACM Press. https://doi.org/10.1145/2047196.2047238

Miranda, J., Makitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., y Murillo, J. M. 2015a. From the Internet of Things to the Internet of People. IEEE Internet Computing, 19(2), 40–47. https://doi.org/10.1109/MIC.2015.24

Miranda, J., Makitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., y Murillo, J. M. 2015b. From the Internet of Things to the Internet of People. IEEE Internet Computing, 19(2), 40–47. https://doi.org/10.1109/MIC.2015.24

Papadopoulou, E., Abu Shaaban, Y., Gallacher, S., Taylor, N., y Williams, M. H. 2010. Two Approaches to Handling Proactivity in Pervasive Systems (pp. 64–75). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-12035-0_8

Presser, M., y Gluhak, A. 2009. The Internet of Things - Connecting the real world with the digital world. Recuperado el 7 de noviembre de 2016, de: https://filestore.eurescom.eu/message/messageSep2009/The-Internet-of-Thing -Connecting-the-real-world-with-the-digital-world.asp

R. Ebling, M. (IBM T. J. W. R. C. 2016. Pervasive Computing and the Internet of Things. IEEE Pervasive Computing, pp. 2–4. Recuperado de https://www.computer.org/csdl/mags/pc/2016/01/mpc2016010002.pdf

Rautmare, S., y Bhalerao, D. M. 2017. MySQL and NoSQL database comparison for IoT application. En *2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016* (pp. 235–238). IEEE. https://doi.org/10.1109/ICACA.2016.7887957

Satyanarayanan, M. 2001. Pervasive computing: vision and challenges. IEEE Personal Communications, 8(4), 10–17. https://doi.org/10.1109/98.943998

Sousa, J. P., y Garlan, D. 2002. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. En *Software Architecture* (pp. 29–43). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-35607-5_2

Srivastava, M., Abdelzaher, T., y Szymanski, B. 2011. Human-centric sensing. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 370(1958).

Sun, M., Pappu, A., Chen, Y.-N., y Rudnicky, A. I. 2016. Weakly supervised user intent detection for multi-domain dialogues. En *2016 IEEE Spoken Language Technology Workshop (SLT)* (pp. 91–97). IEEE. https://doi.org/10.1109/SLT.2016.7846250

Uckelmann, D., Harrison, M., y Michahelles, F. 2011. An Architectural Approach Towards the Future Internet of Things. En *Architecting the Internet of Things* (pp. 1–24). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19157-2_1

Vogel, D., y Balakrishnan, R. 2004a. Interactive public ambient displays. En *Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04* (p. 137). New York, New York, USA: ACM Press. https://doi.org/10.1145/1029632.1029656

Vogel, D., y Balakrishnan, R. 2004b. Interactive public ambient displays. Proceedings of the 2004 ACM Symposium on User Interface Software and Technology, 137. https://doi.org/10.1145/1029632.1029656

Weiser, M. 1991. The Computer for the 21st Century. Scientific American, 265(3), 94–104. https://doi.org/10.1038/scientificamerican0991-94

Wilson, H. J. 2014. Make the Internet of Things More Human-Friendly. Recuperado el 31 de julio de 2017, a partir de https://hbr.org/2014/10/make-the-internet-of-things-more-human-friendly

Yang, X., He, H., Wu, Y., Tang, C., Chen, H., y Liang, J. 2016. User intent perception by gesture and eye tracking. Cogent Engineering, 3(1). https://doi.org/10.1080/23311916.2016.1221570

Zhuang, J., Mei, T., Hoi, S. C. H., Xu, Y.-Q., y Li, S. 2011. When recommendation meets mobile. En *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11* (p. 153). New York, New York, USA: ACM Press. https://doi.org/10.1145/2030112.2030134

Apéndice 1: Tecnologías de comunicación utilizadas en IoT

Web of Things

Web of Things (WoT) es una implementación de protocolos Web para la comunicación entre objetos. (Atzori, Iera, y Morabito, 2014) menciona que una forma de utilizar WoT es a través de una Interfaz API (*Application Programming Interface*) RESTful (*Representational State Transfer*). Con un API RESTful, el dispositivo puede proporcionar información a un cliente a través de una dirección URL, proporcionando como respuesta una estructura de datos en XML o JSON y dando códigos de respuesta similares a HTTP.

Protocolo de Internet

El Protocolo de Internet (IP) es un protocolo que permite proporcionar de una dirección lógica a un dispositivo de Internet de las Cosas. El Internet de las Cosas implica que cada objeto puede ser accedido desde cualquier parte del mundo con una conexión a Internet, por ello es primordial que cada objeto tenga una dirección única. En estos momentos, empresas de telecomunicaciones están actualizando este estándar a la versión 6 (IPv6), la cual va a permitir que se conecten hasta 2^{128} dispositivos en el mundo. Esta nueva versión surgió precisamente con el auge del Internet de las Cosas, pues cada persona puede tener muchos objetos conectados a Internet para obtener un beneficio de ellos.

Wi-Fi y LTE

Wireles Fidelity y *Long-Term Evolution* son dos estándares de comunicación inalámbrica. Aunque ambos permiten la comunicación inalámbrica, la amplitud de área que pueden cubrir para conectar los dispositivos es diferente.

Por su parte Wi-Fi permite una comunicación inalámbrica para una red de área local. Dispositivos de Internet de las Cosas que tiene soporte para Wi-Fi son para utilizarse dentro de una casa o el piso de un edificio. Wi-Fi está basado en el protocolo 802.11 de la IEEE.

Por otro lado, LTE permite una comunicación inalámbrica en un área más amplia como una ciudad. Este protocolo es el utilizado para la transmisión de datos en los teléfonos inteligentes. Este estándar garantiza la conectividad de los dispositivos con total movilidad. Utiliza IP como protocolo de direccionamiento para la transmisión de datos.

MQTT

MQTT por sus siglas *Message Queue Telemetry Transport* es un protocolo de comunicaciones ligero basado en el patrón *publish-subscribe*. Este protocolo permite un flujo de mensajes ligeros entre dispositivos utilizando paquetes de tamaño limitado. Es ideal para dispositivos cuya conectividad es intermitente o y limitada. Debido al patrón arquitectónico en el que está basado, es necesario contar con un bróker o agente. Este es el servicio responsable de distribuir los mensajes desde o hacia el extremo solicitado. Cuando un cliente envía un mensaje, hace una "publicación" a un tópico en el bróker. Si un cliente necesita recibir un mensaje, este cliente debe realizar una "suscripción" en el bróker, por lo que cada vez que un cliente realiza una publicación a un tópico, los clientes que están suscritos a ese mismo tópico reciben el mensaje. Este protocolo es usado ampliamente como un protocolo para comunicación entre maquinas (*Machine to Machine* o M2M).

6LoWPAN

6LoWPAN es un estándar de comunicación propuesto por la IETF, el cual combina el uso de los protocolos IPv6 y WPAN (*Wireless Personal Area Network*). En puntos anteriores ya se comentó lo que es el protocolo IP. WPAN por su parte es un estándar de comunicación inalámbrica que tiene una amplitud de área personal. Esta amplitud es más pequeña que la de área local como la que utiliza Wi-Fi. WPAN está basado en el estándar IEEE 802.15.4. La aplicación comercial principal de este estándar es la tecnología Bluetooth, la cual a partir de la versión 4 (Bluetooth 4 o *Bluetooth Low Energy*) ya brinda la transmisión de paquetes IPv6 y otras mejoras como la disminución en el uso de la energía.

Otros métodos de comunicación

Existen otros métodos de comunicación que permiten enviar información de un dispositivo a otro. Uno que es muy utilizado en la actualidad es el infrarrojo, el cual permite enviar una secuencia de bits que forman un comando en hexadecimal. Esta forma de comunicación es usada en televisiones, climas artificiales, aparatos de ventilación y reproductores de audio por mencionar algunos. Utilizando el infrarrojo, podemos incorporar dispositivos de Internet de las Cosas que no necesariamente están conectados a Internet, por lo que es necesario un dispositivo intermedio que envié estos comandos a estos dispositivos.

Apéndice 2: Características de MongoDB

De acuerdo con (Rautmare y Bhalerao, 2017), algunas características de las bases de datos de tipo No-SQL como MongoDB para las plataformas de Internet de las Cosas son las siguientes.

- Escalabilidad horizontal, permite incrementar el número de nodos o servidores para compartir la carga del sistema.
- Obtención rápida de los datos, ya que los datos se almacenan como objetos (documentos) con toda la información relacionada. A diferencia de una base de tipo SQL como MySQL, se tienen cruzar las tablas con la cláusula JOIN.
- No hay restricción de esquemas, si se necesita almacenar un nuevo parámetro en una colección de documentos, simplemente se especifica a la hora de insertar o actualizar un documento.