Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Computación

# Red de sensores para control de contextos usando servicios web.

tesis que presenta
**Ing. José Abraham Bernal Gutiérrez**

para obtener el grado de
**Maestro en Ciencias en Computación**

directores de la tesis:
**Dr. José Guadalupe Rodríguez García**
**Dra. Maricela Claudia Bravo Contreras**

Ciudad de México
28 de noviembre de 2016

# Agradecimientos

Le agradezco al Centro de investigación y estudios avanzados del IPN (cinvestav). La oportunidad que me ha brindado al permitirme ingresar a uno de sus programas para incrementar mis conocimientos y aptitudes en pro de la ciencia.

También agradezco al Concejo nacional de Ciencia y Tecnología (ConaCyT). Por el apoyo que me brindo para continuar mis estudios y alcanzar el grado de maestro en ciencias.

# Resumen

El ser humano busca la forma de estar siempre en contacto con su medio ambiente, a partir de esto surge la necesidad de entender al entorno y que éste se adapte a nuestras necesidades. La forma de obtener datos del entorno es muy variable, y la cantidad de información que nos provee el ambiente en el que nos encontremos puede ser excesiva. Por lo tanto, se deben desarrollar técnicas que filtren los datos obtenidos del ambiente en el cual se encuentra el usuario y los procesen de manera adecuada, para que produzca información que sea de utilidad de acuerdo a sus necesidades.

En el presente trabajo el contexto del usuario está compuesto por datos intrínsecos a él (perfil) como por datos externos a él (datos provenientes del medio ambiente). Éstos últimos pueden ser capturados haciendo uso de una red de sensores. Una vez analizados ambos conjuntos de datos se puede identificar el contexto en el que el ser humano (usuario) se encuentra.

Para lograr lo anterior dos elementos son indispensables: por un lado el uso de dispositivos de hardware reconfigurable, que aportan la facilidad de poder cambiar su funcionamiento al integrar nuevas funcionalidades. Por otro lado las ontologías permiten hacer uso de la semántica, la cual otorga sentido y significado a los datos a fin de facilitar la interacción entre el usuario y el sistema y entre los sistemas también.

Una vez que se ha agregado semántica a los datos se hace uso de servicios Web para facilitar la interacción con el usuario, ya que al tener la capacidad de ser consumidos por distintas plataformas brindan un sentido de ubicuidad al sistema, además el uso de dispositivos móviles provee de servicios específicos a los usuarios, sin la necesidad de ocupar la total atención del usuario sobre los mismos.

# Abstract

Modern Human beings search to be always connected with his environment. This gives as result the need of understand the environment and customize it according their needs. Several ways can be used to gather data from environment, however the amount of information that can be obtained from the environment can be excessive. As result some mechanisms for filtering and processing the data are needed. These mechanisms should produce useful information for users.

This thesis aims to carry out a process of analysis and selection of information that can be obtained from the environment. So the context can be extracted from environment where the user is. The environmental data is gathered by the use of a sensors network. The external data is analyzed together wit the internal data (user's profile, for example) in order to identify the user's context.

To be capable of identifying the context, two elements are needed: the sensors network and the ontologies. The former for gathering the data from the environment, the latter, through the use of semantics, gives sense and meaning to the data. This eases the interaction between users and the systems and between systems too.

After the semantic has been supplied to data, web services are used to supply the interface with users. These web services can be exploited using different platforms, giving in this way an ubiquitous capability to the system. The use of mobile devices allows the selection of services to be consumed.

# Índice general


**1. Introducción** **13**
- 1.1. Hipótesis y planteamiento del problema . . . . . 14
- 1.2. Objetivos generales y específicos del proyecto . . . . . 14
  - 1.2.1. Objetivo general . . . . . 14
  - 1.2.2. Objetivos individuales . . . . . 14
- 1.3. Metodología . . . . . 15

**2. Estado del arte** **17**
- 2.1. Cómputo ubicuo . . . . . 17
- 2.2. Servicios web . . . . . 22
- 2.3. Ontologías . . . . . 25
- 2.4. Redes de sensores . . . . . 28

**3. Marco teórico** **35**
- 3.1. Cómputo ubicuo contexto . . . . . 35
- 3.2. Hardware reconfigurable . . . . . 36
  - 3.2.1. Arduino . . . . . 37
  - 3.2.2. Programa ejemplo de arduino . . . . . 38
- 3.3. Cómputo móvil . . . . . 39
  - 3.3.1. Android . . . . . 41
- 3.4. Ontologías . . . . . 43
  - 3.4.1. *Extensible Markup Languaje* (XML) . . . . . 43
  - 3.4.2. *Resource Description Framework* (RDF) . . . . . 45
  - 3.4.3. *Ontology Web Language* (OWL) . . . . . 46
- 3.5. Servicios Web . . . . . 47

**4. Propuesta de solución** **53**
- 4.1. Sistema propuesto . . . . . 54
- 4.2. Arquitectura propuesta . . . . . 56

**5. Implementación** **61**
5.1. Diseño de la meta-ontología . . . . . 61
5.1.1. Ontología de dispositivos . . . . . 61
5.1.2. Ontología espacio físico . . . . . 63
5.1.3. Ontología de personas . . . . . 67
5.1.4. Meta-Ontología . . . . . 69
5.2. Red de sensores con arduino . . . . . 70
5.2.1. Sensores utilizados . . . . . 71
5.2.2. Actuadores . . . . . 75
5.3. Servicios web . . . . . 77
5.3.1. Servicio de localización: . . . . . 79
5.3.2. Servicio de búsqueda: . . . . . 82
5.3.3. Servicio de control Manual-Automático: . . . . . 83
5.3.4. Servicio de ingreso de preferencias: . . . . . 85
5.3.5. Programa demonio: . . . . . 88
5.4. Consumo de servicios con android . . . . . 91
5.4.1. Consumo del servicio de localización. . . . . 91
5.4.2. Consumo del servicio de búsqueda. . . . . 94
5.4.3. Consumo del servicio de control manual. . . . . 96
5.4.4. Consumo del servicio de preferencias. . . . . 100

**6. Pruebas y resultados** **103**
6.1. Pruebas individuales de arduino . . . . . 103
6.2. Pruebas a los servicios . . . . . 109
6.2.1. Pruebas individuales a los servicios . . . . . 111
6.2.2. Escenario con el sistema completo . . . . . 117
6.3. Pruebas de modificación de preferencias . . . . . 121
6.4. Pruebas para ubicación de usuarios . . . . . 130
6.4.1. Escenario de usuario sin privilegios . . . . . 131
6.4.2. Escenario usuario con privilegios . . . . . 133

**7. Conclusiones y trabajo futuro** **137**
7.1. Conclusiones . . . . . 137
7.1.1. Conclusión del objetivo general . . . . . 139
7.1.2. Conclusión de la hipótesis . . . . . 139
7.2. Trabajo futuro . . . . . 140

# Índice de figuras


3.1. Android framework details . . . . . 42
3.2. arquitectura de los servicios web . . . . . 51

4.1. Capas de la arquitectura . . . . . 56
4.2. Arquitectura Gráfica . . . . . 57

5.1. Ontología Devices (estructura) . . . . . 62
5.2. Ontología PhysicalSpace (estructura) . . . . . 64
5.3. Ontología People (estructura) . . . . . 67
5.4. Meta-ontología Institution (estructura) . . . . . 69
5.5. Sensor con LDR (esquema) . . . . . 74
5.6. Etapa de potencia de estado sólido (esquema) . . . . . 76

6.1. Petición desde explorador Web . . . . . 105
6.2. 'Fan1' y 'Lum1' activos . . . . . 106
6.3. Consulta desde explorador Web . . . . . 107
6.4. Muestra de lecturas a través del puerto serial . . . . . 108
6.5. Servicios desde el explorador Web . . . . . 110
6.6. Pruebas al servicio de localización desde soap UI . . . . . 111
6.7. Pruebas al servicio de localización desde android . . . . . 112
6.8. Prueba al servicio de búsqueda desde soap UI . . . . . 112
6.9. Prueba al servicio de búsqueda desde android . . . . . 113
6.10. Obtención de datos de los actuadores con soap UI . . . . . 114
6.11. Control de actuadores a través de android . . . . . 115
6.12. Envío de preferencias desde soap UI . . . . . 116
6.13. Envío de preferencias desde android . . . . . 116
6.14. Etapa de potencia y control con arduino . . . . . 118
6.15. Lampara de corriente alterna . . . . . 119
6.16. ventiladores de corriente directa . . . . . 120
6.17. Etapa de potencia de corriente directa . . . . . 120

6.18. Etapa de potencia para motor de corriente alterna . . . . . 121
6.19. Compresor de aire . . . . . 122
6.20. Arduino para lectura de sensores . . . . . 122
6.21. Modo de operacion manual . . . . . 123
6.22. Ingreso de preferencias . . . . . 124
6.23. Valores Iniciales . . . . . 125
6.24. temperatura y humedad iniciales . . . . . 125
6.25. Incremento del valor de temperatura . . . . . 126
6.26. Cambio de estado del aire acondicionado . . . . . 127
6.27. Incremento del valor de humedad . . . . . 127
6.28. Encendido de *Fan1* . . . . . 128
6.29. Cambio de estado de *Fan1* . . . . . 128
6.30. Incremento del valor de luz . . . . . 129
6.31. Encendido de lamparas . . . . . 130
6.32. Cambio de estado de *Lun1* . . . . . 130
6.33. Búsqueda de persona . . . . . 131
6.34. Búsqueda de colega . . . . . 132
6.35. Control manual desde android . . . . . 133
6.36. Buscando mi ubicación . . . . . 135
6.37. Buscando a compañero . . . . . 136

# Capítulo 1

# Introducción

El desarrollo tecnológico en los últimos años ha permitido observar una tendencia hacia las plataformas abiertas, las cuales muestran un gran potencial para el desarrollo de sistemas distribuidos. Dichas plataformas permiten a los usuarios la conexión y desarrollo de sistemas conectados a través de una red, ya sea Internet o red de área local.

Como una introducción al cómputo ubicuo nos remontamos al tiempo en el que **Mark Weisser** hizo la primer referencia a esta área de la computación. En su artículo [1] expone una visión a futuro del mundo en el que las computadoras se vuelven parte del día a día del ser humano, al ocultarse en el trasfondo de las actividades cotidianas del usuario, por medio de los objetos de uso diario hasta volverse indistinguibles para éste. A diferencia de la realidad virtual, la cual el autor percibe como un mapa que además necesita de hardware especial para tener una interacción con el usuario, y que dicho hardware no se vuelve invisible ni mejora el mundo que ya existe. El objetivo del cómputo ubicuo es mejorar la vida del usuario al brindarle información útil y filtrada, para así dar un sentido de omnipresencia al sistema sin que el usuario tenga que ser absorbido por la interacción con el sistema mismo.

Recientes avances en diversas áreas han permitido plantear la realizacion de la idea de Weisser. Una de las áreas que ha tenido un impacto importante es el desarrollo de dispositivos embebidos, los cuales gracias a la miniaturización nos han permitido desarrollar las redes inalámbricas de sensores (*Wireless Sensor Networks - WSNs*) [2], estas redes en la actualidad están presentes en distintos ambientes y áreas, como son: el cuidado de la salud

(e-healthcare), y en la industria, por mencionar algunos. Las *WSNs* han protagonizado avances muy importantes, ya que proponen nodos conocidos como *clusters* de sensores inalámbricos. Los nodos o *clusters* pueden ser desplegados de forma aleatoria en un área de interés, para obtener información del ambiente en el que son desplegados.

## 1.1. Hipótesis y planteamiento del problema

El ser humano continuamente recibe una gran cantidad de información, la cual proviene del ambiente en el que se encuentra. En esos ambientes pueden existir diferentes contextos, y si estos ambientes proveen información variada, se podría obtener y procesar dicha información para así presentarla al usuario por algún medio que permita un fácil entendimiento y uso.

El problema a resolver es la adquisición de la información del ambiente, la cual deberá ser filtrada para así presentarla al usuario y que éste pueda hacer uso de dicha información, de acuerdo al ambiente en el cual se encuentra. Ademas, se buscará integrar un dispositivo que se mimetice en el día a día del usuario.

## 1.2. Objetivos generales y específicos del proyecto

### 1.2.1. Objetivo general

Demostrar cómo la adquisición de información del contexto a través de una red de sensores, puede ser utilizada para proveer de servicios adecuados a los usuarios, dependiendo de sus perfiles y necesidades, facilitando la interacción del usuario con el ambiente y con otros usuarios en forma inalámbrica.

### 1.2.2. Objetivos individuales

1. Diseñar e implementar una red de sensores locales, para la adquisición automática de datos del contexto relevantes.
2. Diseñar una ontología para la representación de datos relevantes, obtenidos por medio de la abstracción del contexto y del perfil usuario.
3. Diseñar una plataforma basada en servicios web con la cual se proporcionará información relevante del contexto al usuario.

4. Implementar una aplicación para distintos dispositivos móviles la cual probará la contribución en esta área.

## 1.3. Metodología

- Diseñar una red de sensores para adquirir datos del contexto y así poder obtener información del ambiente en donde el usuario se encuentra.
- Realizar pruebas rigurosas del funcionamiento de la red de sensores y de los datos adquiridos por medio los sensores, para asegurar la lectura adecuada de datos a enviar al sistema.
- Con base en la información obtenida del ambiente, se debe diseñar una ontología para evitar las ambigüedades en la información obtenida, y así mejorar la información que se obtiene del contexto.
- Definir los servicios Web que harán uso de ontologías; de acuerdo a los servicios a brindar y los dispositivos disponibles (por ejemplo: relevadores, solenoides, motores etc.) que se utilizarán para poder brindar los servicios adecuados al usuario.
- Una vez que se han identificado los actuadores, se procederá a montar los servicios y hacer pruebas minuciosas de su funcionamiento.
- Crear una aplicación móvil, la cual será el intermediario entre el usuario y el sistema, que conjuga los pasos anteriores. Dicha aplicación deberá llevar también pruebas rigurosas de funcionamiento.

# Capítulo 2

# Estado del arte

En algunas investigaciones desarrolladas se utilizan las *WSNs* junto con la integración de ontologías con el objetivo de mejorar el contenido la información obtenida del ambiente o contexto. Al combinar esta información con XML y OWL, se puede brindar significado y sentido a los datos para que sean presentados a los usuarios de forma clara, para que éstos puedan hacer un uso mas adecuado de los datos obtenidos. En este capítulo se presentan algunos trabajos relacionados con nuestra propuesta.

## 2.1. Cómputo ubicuo

Se han desarrollado diversos trabajos con diferentes enfoques, en algunos casos buscan mejorar la comunicación entre el hardware y el software, desarrollando mecanismos que permiten el descubrimiento dinámico de servicios [3], en este trabajo los investigadores discuten varios puntos importantes a tomar en consideración, como son el conocimiento a *priori* del hardware, su capacidad para la obtención de datos y sus características.

El objetivo es que cada dispositivo detectado se agregue de forma automática al sistema conforme es descubierto. Por consiguiente los investigadores proponen el **Rubicon Semántico**, como resultado de su investigación, el cual permite la interacción de todos los dispositivos que están disponibles en la red, con la posibilidad de que éstos se descubran unos a otros de forma dinámica. Esto es posible al registrar los servicios descubiertos en una tabla, la cual puede ser consultada para así lograr una interacción más adecuada entre los dispositivos y servicios disponibles en la red.

El incremento en la posibilidad de desarrollar *WSNs.* Plantea nuevos retos, de los cuales uno está enfocado a la administración de dichos sistemas. En el trabajo presentado en [4] se aborda el desarrollo del tema de la administración al introducir una serie de mÉtodos tomados de sistemas modernos. Estos sistemas presentan cinco características destacadas como son: independencia operativa, independencia gerencial, distribución geográfica, comportamiento emergente y el desarrollo evolutivo.

Los sistemas son ubicuos y agregan el concepto de sistemas federados o federación de sistemas, en los que hay poco poder autoritario y la participación de los miembros en estos sistemas está basada en la colaboración y coordinación. Por tanto, los sistemas federados poseen una autonomía significativa, además cuentan con cierto grado de heterogeneidad y una distribución geográfica de los distintos grupos que conforman al sistema.

Se ha propuesto que los programas con acceso a la Web, bases de datos, sensores, y una variedad de otros dispositivos físicos lleven a cabo servicios de comunicación e interconexión [5]. En la siguiente década, las computadoras serán más omnipresentes, y la mayoría de los dispositivos tendrán alguna computadora dentro de éstos [5].

En los últimos años los objetos listos para Internet se vuelven actores activos y compañeros de la Web, como lo vemos en el trabajo [6], este trabajo desarrollo un prototipo llamado "*Paraimpu*", el cual permite a las personas conectarse, utilizar, compartir y componer "cosas" físicas y virtuales, servicios y dispositivos en orden de crear aplicaciones personalizadas pervasivas. *Paraimpu* permite hacer abstracciones en el modelado de cosas y conexiones. Gracias a dos aspectos clave.

- Sensores: cualquier cosa capaz de producir datos de cualquier tipo (texto, numérico, JSON, XML, etc.).
- Actuadores: cualquier cosa capaz de llevar a cabo acciones al consumir los datos producidos por un sensor.

Los sensores y actuadores se comunican con *paraimpu* utilizando el protocolo HTTP. La abstracción de la conexión representa un flujo de datos en tiempo real entre dos objetos el cual siempre involucra un sensor como fuente de datos y un actuador como receptor de los datos; **Paraimpu** administra datos de tipos como: cadenas, números, JSON y XM, por tanto, los sensores

y actuadores producen o consumen esos tipos de datos. En caso de que los actuadores consuman tipos de estructuras de datos complejas, **Paraimpu** guía al usuario con widgets ad-hoc y GUI para facilitar la definición de filtros.

La demostración de **Paraimpu** está enfocada en el uso de áreas de trabajo del usuario y está compuesto principalmente de cinco áreas.

- *Amigos (área 1)*: al seleccionar a un amigo, es posible descubrir su perfil, sus objetos compartidos y marcarlos en orden de importar los objetos compartidos es nuestro propio espacio de trabajo.
- *El conjunto de sensores y actuadores (área 2)*: muestra los sensores y actuadores disponibles, además permite a los usuarios agregar objetos virtuales o reales al área de trabajo personal.
- *Sensores y actuadores agregados/marcados (áreas 3 y 4)*: es la lista de los sensores o actuadores agregados (utilizando la paleta) o marcados (objetos compartidos).
- *Conexiones (área 5)*: la lista de conexiones. Cada uno tiene los controles para configurar, activar y desactivar el flujo de datos entre el sensor y el actuador.

**Paraimpu** es una plataforma que tiene la facilidad de agregar, componer, compartir, filtrar y adaptar los datos provenientes de servicios en línea, redes sociales e incluso objetos listos para la web.

El Internet de las Cosas (IoT) es un paradigma reciente que ha ganado el momento al proponer la extensión de Internet a una variedad de "cosa", comúnmente censadas como entidades físicas de interés (EoI) para los humanos. En este trabajo [7], los EoIs son representados a través de un conjunto de propiedades que pueden ser observadas, medidas o disparadas a través de dispositivos tales como las etiquetas de *Radio Frequency Identification* (RFID), sensores, actuadores u otros componentes tecnológicos inteligentes.

Indudablemente, el llevar a cabo una mejor automatización de las tareas de los usuarios, por ejemplo, hacer la vida del usuario más fácil. Alineando con las ideas que conlleva el cómputo ubicuo o cómputo pervasivo [8], tal visión guía a los pensadores tempranos a la conclusión de que la automatización puede ser mejor activada si la interoperabilidad entre EoIs, dispositivos y el contexto en el que ellos deben proveer información, fuese mejorado [9].

En particular, varios investigadores se dieron cuenta de que el uso de las tecnologías Web semánticas es particularmente adecuado para mejorar dicha interoperabilidad, como también en la activación de representaciones ricas y flexibles de cualquier recurso capaz de ser descrito.

Ahora bien, el reciente interés en proveer sensores o actuadores ha guiado a billones de servicios o datos ofrecidos a través de diferentes plataformas. En ese caso, pese a la interoperabilidad que puede ser alcanzada con el uso de tecnologías web semánticas, el uso de una aproximación centralizada para procesar semánticamente, asociar, almacenar y buscar entre descripciones tanto de dispositivos o EoIs, lo que podría guiarnos a una visión del IoT fuertemente realizable.

El uso de las tecnologías Web semánticas debe de ser pensadas en consideración con el despliegue. Por ejemplo, integrar los aspectos distribuidos y ubicuos del IoT, en particular cuando involucra EoI o dispositivos que puedan ser puestos en un ambiente interior, promoviendo una alta movilidad. Cada nodo del framework tiene capacidades de razonamiento locales, que dicho conocimiento sea compartido a través de nodos cercanos geográficamente, considerando la ubicación como un criterio muy importante cuando se comparten o asocian EoIs con dispositivos.

Una de las peculiaridades del IoT es que involucra a los dispositivos ubicados en exteriores, como también a los desplegados en un ambiente interior [7]. Uno puede estar dispuesto a crear un nodo en cada habitación como también el administrar varios tipos de dispositivos conectados y diferentes perfiles de personas.

Para capturar dicho nivel de granularidad, se utilizan las tecnologías Web semánticas y en particular el *Web Ontology Language* (OWL) debido a su habilidad para proveer descripciones ricas para cualquier tipo de recursos. Cada nodo semántico ha sido desplegado como una aplicación Web alojada en un servidor *Tomcat*, en una computadora ejecutando una configuración de computadora personal estándar.

En el área en los sistemas de hogares inteligentes, se proveen capacidades de automatización que permiten a los propietarios el tener un control completo sobre su hogar, y promover la eficiencia energética que les permita ahorrar dinero en las cuentas de energía. Esas soluciones de hogares inteligentes integran varios dispositivos con diferentes capacidades tales como la

detección de intrusos, video vigilancia, detección de fuego, monitorización de la salud de pacientes y entretenimiento. Por lo tanto, en el trabajo [10] se propone una arquitectura de software de hogar inteligente basada en el framework OSGi (*Open Services Gateway Initiative*) que integra sin problemas protocolos heterogéneos y diversos tipos de dispositivos utilizados en las redes del hogar. El objetivo es permitir a los usuarios finales agregar nuevos dispositivos a petición, independientemente de las peculiaridades de descubrimiento impuestas por el protocolo de comunicación en particular.

Los estándares predominantes en las redes del hogar utilizan diferentes medios (por ejemplo, potencia, distintas bandas de RF(Radio Frecuencia)), de direccionamiento estáticos o dinámicos de dispositivos, y diferentes mecanismos de descubrimiento de dispositivos. Debido a esa diversidad y a menudo la incompatibilidad de los mecanismos desde diferentes estándares de red, los sistemas de hogares inteligentes en el mercado permanecen fragmentados y proveen solo soluciones parciales en el direccionamiento de protocolos simples y subconjuntos de dispositivos.

A pesar de la naturaleza heterogénea de los estándares de red en los hogares, la investigación existente en los hogares inteligentes generalmente asume una arquitectura subyacente homogénea [11] . Ya que la interoperabilidad de los dispositivos es lograda por una pila de dispositivos multicapa los cuales consisten rigurosamente de controladores, adaptadores comunes para dispositivos similares y una representación de alto nivel descrita semánticamente por un modelo de dominio [12]. Además, la aproximación de los autores integra servicios en la nube para incrementar la cobertura del proceso de descubrimiento de dispositivos y para entender la funcional del hogar inteligente con nuevas aplicaciones, controladores, y servicios computacionales intensivos.

El modelo semántico es formalizado como una ontología. Las Ontologías traducen a un dominio de interés en un conjunto de conceptos, propiedades y relaciones gobernadas por semánticas estrictamente formalizadas. La taxonomía de los dispositivos hogareños y ambientes hogareños permite la recuperación de la abstracción al vuelo de dispositivos semánticos, junto con el conocimiento contextual requerido por las aplicaciones consientes del contexto.

Se formalizo la ontología utilizando *Web Ontology Languaje* (OWL), porque es bien soportado por los *toolkits* y los motores de razonamiento. En

orden de obtener una amplia expresividad y también llevar a cabo un razonamiento rápido se utilizan diferentes componentes para el razonamiento en línea y fuera de línea.

## 2.2. Servicios web

La Web del presente fue diseñada primordialmente para uso e interpretación humanos, por tanto, es fundamental tener programas de computadora o agentes que confiablemente implementen, una interoperación a gran escala de los servicios web conforme sea necesario [13].

Algunos trabajos de descubrimiento automático *Automatic Web service discovery* [5, 14] involucran la ubicación de los servicios web que proveen un servicio particular y los agregan, a las propiedades de las solicitudes, como un "marcador" para los servicios semánticos. Por lo que se puede especificar la información necesaria para el descubrimiento de servicios web como una *marca* añadida para la interoperación computacional en los sitios de servicios, y un registro de servicios o (mejora ontológica) en los motores de búsqueda para automáticamente localizar los servicios apropiados.

La Web Semántica es una visión de una nueva arquitectura para la *World Wide Web* [15], la cual está caracterizada por la asociación formal semántica de máquinas con más contenido que la Web tradicional. La motivación original fue el incremento en el proceso de automatización en el procesamiento de la información basada en Web y el mejoramiento de la comunicación operacional de los sistemas de información basados en la Web.

Las técnicas web semánticas, consisten de la aplicación de técnicas para la representación del conocimiento en un ambiente distribuido [14, 16, 17], (potencialmente a escala Web), han probado ser de utilidad al proveer descripciones ricas de servicios web. Los servicios web semánticos, puede ser considerados como un nuevo objeto de investigación, ya que están generalmente definidos como servicios web con descripciones aumentadas [15] . Para facilitar una alta automatización del descubrimiento de servicios, composición, invocación, y monitoreo en ambientes abiertos, no regulados, e incluso caóticos (como es, la Web) [14]. Por lo que, la relación entre la Web semántica y la arquitectura actual de servicios web depende de para qué que se desea aplicar.

Existen varios tipos de dispositivos empotrados, tales como sensores inalámbricos, RFID (por sus siglas en ingles *Radio Frequency IDentification*), actuadores entre otros. Dichos dispositivos permiten el uso de protocolos de comunicación para conectarse a internet y así tener diferentes capacidades, incluidos varios recursos físicos, capacidades de cómputo, ancho de banda de comunicación y capacidades de procesamiento como lo vemos en [18], el nombre del proyecto es **RestThin,** está basado en *Representational State Transfer* (REST)que permite a los dispositivos el acceso a internet y a compartir información través éste.

El estilo arquitectónico de REST permite la mejor aceptación de escenarios en el manejo básico de los recursos. La meta de RestThin es que varias aplicaciones puedan coexistir y acceder a los dispositivos empotrados como recursos de información web. En esta infraestructura, los dispositivos empotrados y la información Web son considerados como recursos y manipulados por una interfaz uniforme.

Actualmente, existen varios frameworks y lenguajes para describir formalmente los servicios web, en [19] se mencionan algunos, cada uno con fortalezas y limitaciones, por ejemplo: Web Ontology Language for services (OWL-S) [20], Web Service Modeling Ontology (WSMO) [21] y Semantic Annotations for the Web Services Description Language (SAWSDL) [22] son algunos de los frameworks. En esencia, OWL-S facilita el significado, esto es, intercambio de mensajes semánticos.

La WSMO está basada en el trabajo temprano del método de solución unificado de problemas, el cual forma parte de un *framework* para desarrollar un sistema razonador de conocimiento intensivo basado en bibliotecas de los componentes de un problem-solving genérico. SAWSDL no especifica un lenguaje para representar modelos de servicio semánticos. En lugar, adopta un mecanismo ligero por el que los conceptos de los modelos semánticos pueden ser referidos utilizando anotaciones.

En esencia, los tres frameworks (OWL-S, WSMO y SAWDSL) son lenguajes de servicios web, y varios de sus constructores son sintácticamente similares. A pesar de que, OWL-S, WSMO y SAWSDL apuntan a mejorar los servicios web existentes al utilizar tecnologías web semánticas, por lo que llevan a cabo esta visión fundamentalmente de diferentes maneras.

En un esfuerzo para alcanzar este potencial, W3C sugiere el aumento semántico de servicios Web (WS), de acuerdo a los principios de diseño para un conjunto unitario, llamados, una colección de *'de facto'* para sintaxis de estándares web, con los protocolos de direccionamiento y comunicación. El esfuerzo previsto para estos resultados el colectivamente conocido como la web semántica, donde cada resultado es provisto con metadatos, haciéndolos fácilmente accesibles por los programas de computadora.

En [23] se hace la propuesta de un *framework* con el cual se administren los "objetos" o dispositivos pueden partir desde nodos simples de sensores hasta sistemas empotrados muy complejos, dependiendo de su propósito y rol. Las áreas posibles de aplicación incluyen varios campos, tales como infraestructura de administración, aplicaciones de seguridad, protección ambiental, agricultura inteligente, industria, tráfico, servicios de negocio, administración urbana etc.

Con el soporte de las nuevas tecnologías, tales como la comunicación móvil de banda ancha y el cómputo en la nube, los sistemas IoT son capaces de procesar el estado de los objetos y el proveer una interfaz de usuario inteligente para la administración para facilitar las decisiones necesarias basadas en las notificaciones en tiempo real [24, 25]. Una implementación de dichos sistemas, es crucial para proveer un camino eficiente para la recolección de datos crudos para procesarlos y analizarlos posteriormente. Además, muy necesario el uso de un middleware que pueda superar la complejidad al proveer la transparencia necesaria para el usuario final.

El sistema propuesto esta lógicamente dividido en las siguientes partes principales: sensores, *aggregators*, cores, y *generic data adquisition system* (GDAS). Los sensores son dispositivos de baja potencia que recolectan información sensorial del ambiente físico, y envían las lecturas a otros componentes del sistema. Los sensores están conectados a los *aggregators* a través del protocolo de comunicación inalámbrica para evitar la necesidad de construir una infraestructura cableada.

El sistema está diseñado para aceptar a sensores equipados con cualquier interfaz de comunicación. El rol del aggregator local es el interactuar directamente con los nodos de sensores inalámbricos, recolectar datos crudos, monitorear el estado de los sensores, actualizar la configuración individual de los sensores, etc. Por lo tanto, uno de los roles más importantes del Aggregator es el servir como un protocolo puente entre cualquier tipo de protocolo

de sensor inalámbrico, y el resto del sistema.

El rol de Core local es el monitorear o controlar a todos los aggregators locales y el conectar a la red local consistente de varios aggregators al servicio de datos en la nube, para la recolección de datos y la monitorización (Generic Data Acquisition System, GDAS) [23]. El Core esta típicamente basado en la potencia computacional del sistema empotrado el cual es capaz de ejecutar un sistema operativo de propósito general (por ejemplo Linux), por tanto, los sistemas embebidos basados en Linux proveen una excelente elección al permitir el uso de administradores de bases de datos ligeros (DBMS).

La parte central de todo el sistema es el *Generic Data Acquisition System* (GDAS) cuyo propósito principal es el proveer significado para la colección escalable de datos de sensores, procesamiento, y almacenamiento. GDAS se comunica con los Cores como mediadores para la administración de las redes locales de sensores.

*GDAS* está implementado como un servicio Web, y está ampliamente construido utilizando *frameworks open source* para proveer un servidor mínimo funcional para administrar al sistema completo. Los segmentos *GDAS* son micro servicios, y necesitan ser ejecutados en la misma computadora o en sistemas distribuidos, en un número ilimitado de instancias independientes. Esto asegura la escalabilidad y modularidad desde el punto inicial de la administración de recursos.

## 2.3. Ontologías

Varios de los avances tecnológicos en comunicación, computación, almacenamiento, sensores y dispositivos móviles han marcado el camino para hacer realidad la visión de Mark Wiesser [1]. A pesar de estos avances y la investigación desarrollada en estas áreas, los sistemas ubicuos siguen enfrentando nuevos retos conforme surgen nuevos desarrollos, lo que nos lleva a buscar la integración de esos desarrollos en la investigación para así consolidar a los sistemas ubicuos.

En filosofía, una ontología es una teoría acerca del ser en naturaleza [26]. La ontología como disciplina estudia dichas teorías [26, 5]. Sin embargo los investigadores de la inteligencia artificial y la Web han adoptado el término

para su propia jerga, ya que para ellos una ontología es un documento o archivo que formalmente define las relaciones entre términos [26, 27].

El trabajo presentado en [28] se presenta a la *"federation of ubicomp systems"* como una colección de nodos inteligentes, donde cada nodo está compuesto por un conjunto de objetos (dispositivos) heterogéneos, los cuales pueden ser hardware o software. Además, los autores desarrollan un *framework* que permite la interaccíon de distintos nodos, con los cuales el usuario puede interaccionar, al realizar tareas en un nodo y trasladarse a otro con la ventaja de poder resumir su trabajo o tarea pendiente en algún otro nodo del sistema. Por consiguiente, el sistema le da al usuario la libertad de trasladarse de un nodo a otro sin la necesidad de que el usuario tenga que regresar al nodo en el que dejó sus tareas activas, para resumir el trabajo pendiente.

Además, el tipo de ontología más común para la Web cuenta con una taxonomía y un conjunto de reglas de inferencia. La taxonomía define clases de objetos y las relaciones entre éstos [26]. Las clases, subclases y relaciones entre entidades son una muy poderosa herramienta para el uso en la Web. Podemos representar un gran número de relaciones entre entidades al asignar propiedades a las clases y permitiendo a las subclases el heredar dichas propiedades.

Otra área importante que ha tomado fuerza son los servicios Web semánticos [26]. En el año 2001 la mayoría del contenido web estaba diseñado para la lectura por los humanos, y no para que los programas de computadora los manipulen de forma significativa. Los autores argumentan que la Web semántica brindará la estructura para que el contenido sea significativo en las páginas web, al crear un ambiente en el que los agentes de software que iteran dentro de una página puedan llevar a cabo tareas sofisticadas para los usuarios. Por consiguiente, la Web semántica no está separada de la Web ya que es una extención de la actual, en la que la información está representada con un significado bien definido, permitiendo que los humanos y las computadoras trabajen en colaboración.

Para que la Web semántica funcione, las computadoras deben tener acceso a colecciones bien estructuradas de información y a un conjunto de reglas de inferencia, tal que, puedan ser utilizadas en conjunto para conducir a un razonamiento automatizado. Se puede decir que el principal reto de la Web semántica es el proveer un lenguaje que exprese datos y reglas para

el razonamiento acerca de los datos y que permita usar reglas provenientes desde cualquier sistema de representación del conocimiento existente, para ser exportado a la Web.

Para lograr el objetivo los autores mencionan dos tecnologías importantes para el desarrollo de la web semántica que ya están disponibles las cuales son: *eXtensible Markup Languaje*[1] (XML) y *Resource Description Languaje*[2] (RDF). XML Permite a los usuarios agregar una estructura arbitraria para sus documentos, pero no dice nada acerca del significado de la estructura del documento. Y el significado es expresado por RDF, el cual lo codifica en conjuntos de *tripleta*, donde cada *tripleta* debe ser el sujeto, verbo u objeto de una sentencia primaria.

El potencial real de la Web semántica será explotado por las personas que crearán los programas que recolectarán la información de la Web desde diversas fuentes, para procesar la información e intercambiar los resultados con otros programas [26]. La efectividad de dichos agentes de software se incrementará exponencialmente. Junto con el contenido Web legible por las cómputadoras y los servicios automatizados que estén disponibles. Además, la Web semántica promueve dicha sinergia; incluso los agentes que no fuesen expresamente diseñados para trabajar en conjunto podrán transferir datos entre sí, siempre y cuando los datos tengan connotación semántica [29, 27].

Los avances en las tecnologías inalámbricas, tales como los teléfonos inteligentes personales, han hecho un amplio rango de eficientes, ahorradoras y potentes aplicaciones para el cuidado de la salud en el hogar [30]. Cada uno de esos sensores está acompañado de un manejador médico el cual es responsable de recolectar datos de dichos sensores y entregar esos datos, a través de un protocolo propietario.

Un problema que incrementa poco el costo directamente es que las mediciones médicas están fragmentadas entre varios dispositivos, además de que presentan un problema de integración para el médico, en [31]. Con el crecimiento explosivo en el uso de teléfonos inteligentes, crea oportunidades interesantes para ser explotadas.

[1] https://www.w3.org/XML/ último acceso 10-11-16
[2] https://www.w3.org/RDF/ último acceso 10-11-16

El problema con la información del sistema de salud es la complejidad de la estructura de la información. Dentro del dominio de la salud existen subdominios e incluso cada uno de los subdominios tiene un vocabulario particular y relaciones específicas.

La solución está compuesta de dos ontologías y un mapeo entre ambas. *Healthcare Semantics Lite* (HSL) es una ontología que representa el contexto del consumidor, y Subset es una ontología que representa el contexto médico.

La ontología *Healthcare Semantcs Lite* (HSL) está compuesta de clases específicamente enfocadas hacia el conjunto de datos mínimo necesario para la lectura médica. La ontología es una representación de los datos que en los que el público en general se preocupa y los que un desarrollador de aplicaciones necesita [31].

## 2.4. Redes de sensores

Una tendencia visible, en la comunicación entre sensores y otros sistemas, es la posibilidad de que el sistema sea escalable. Las redes inalámbricas de sensores en estos casos son una excelente opción, además se busca que posean una interacción con otros sistemas. Al conectar los sensores por medio de las tecnologías de comunicación de red de bajo costo, se abre el camino al surgimiento de las *WSNs* y éstas a su vez dan lugar al surgimiento de nuevos ecosistemas los cuales cuentan con un número variado de aplicaciones diferentes [32].

En dicho trabajo los autores proponen montar pequeños servidores web en sistemas embebidos que ya cuentan con sensores y también con recursos limitados en poder de cómputo, memoria, etc. Además permiten la comunicación a través de protocolos de red (como: UDP, HTTP). Con la finalidad de poder comunicarse a través de mensajes entre los diferentes sistemas, se busca lograr un sistema heterogéneo a través de una comunicación libre entre dispositivos que se integren o dejen el sistema, sin importar el tipo del mismo.

Con la mejora de los dispositivos de uso diario, se pueden obtener beneficios significativos, ya que estos objetos no requieren de la total atención por parte del usuario, lo que facilita la realización de sistemas de cómputo ubicuo [33]. En este trabajo los investigadores proponen dispositivos de

hardware, los cuales son integrados en algunos objetos secundarios (tazas, escritorios, etc.) que se encuentran un ambiente de oficina.

Los objetos utilizados en este caso son unas tazas de café, las cuales se convierten en objetos **aumentados**[1]**,** con la capacidad de medir la temperatura del líquido dentro de la taza. Además si se encuentran en movimiento ya sea desplazamiento o inclinación, es capaz de proveer la ubicación geográfica dentro de la oficina.

El sistema desarrollado puede ser extendido gracias a los receptores que están ubicados en diferentes partes del edificio por lo que se puede saber en dónde está cada usuario, ya que el sistema cuenta con un identificador individual para cada dispositivo. Por consiguiente, al poner en práctica el diseño del sistema se puede saber, por ejemplo, si algún usuario está en su oficina o si la sala de juntas está ocupada.

En el área de aplicación podemos encontrar trabajos en los que se utilizan las ontologías y reglas para dar semántica a la Web como es el trabajo [34]. En este trabajo los autores proponen una ontología compuesta de cuatro clases principales que son: *canal*, *evento*, *acción* y *regla*. Además cuenta con algunas propiedades principales como son: son *hasParameter*, *hasCategory*, *activechannel* y *spinRule*.

- La clase *Canal* define individuos ya sea que generen eventos, provean acciones o ambos. Además, los sensores y actuadores también son modelados como canales dado que ellos producen eventos y/o acciones.
- *Evento* define un suceso particular de un proceso, el cual puede disparar reglas, en la web de eventos (EWE Event Web) los eventos son instantáneos, es decir, no tienen duración con respecto al tiempo [35].
- *Acción* define una operación o proceso provisto por un canal, las acciones producen efectos cuya naturaleza depende de la acción, también puede producir una acción por ejemplo activar el interruptor de una luz.
- *Regla* define una regla de condición de evento acción (Event-Condition-Action, ECA), la cual es disparada por un evento que produce la ejecu-

[1]Los objetos aumentados son objetos que no cambian su propósito ni apariencia, pero son modificados para agregar funcionalidades

ción de una acción. Las reglas definen interacciones particulares entre instancias de las clases evento y acción transfiriendo información desde el evento a la acción.

- *hasParameter* representa los parámetros de una acción o un evento las cuales diferencían a cada individuo. *hasCategory* indica que un canal, evento o acción pertenece a cierta categoría.
- *hasActiveChannel* asocia los usuarios al canal en el cual ellos tienen una cuenta.
- *hasCreator* asocia las reglas de instancia con la lógica de ejecución SAPRQL[1] (es un acrónimo recursivo del inglés
- *SPARQL Protocol and RDF Query Language*) descrita utilizando el giro del vocabulario. Por consiguiente, EWE permite definir reglas donde los canales, eventos y acciones involucradas no están acotados.

La capa de generación es necesaria para distribuir dichos eventos y los motores en la capa de procesamiento están a cargo de ejecutar las reglas cuando los aparecen sus disparadores. Cuando los servicios web y los sensores generan eventos, la red de sensores o el adaptador de servicios (API), se genera un mensaje el cual es empujado al *Message oriented Middleware* (MOM) en la capa de transporte; la red de sensores y los adaptadores de servicios API se suscriben a la *acción de mensajes*. Cuando estos dos módulos reciben la acción-mensaje proveniente de MOM, ellos reenvían dichos mensajes a los actuadores o a los servicios web, los cuales son responsables de interpretar y ejecutar dichas acciones.

El motor de reglas semánticas es el responsable de: procesar los mensajes de eventos entrantes, ejecutar las reglas, descartar los mensajes una vez que han sido procesados para evitar ejecutar la misma reglas múltiples veces para el mismo evento.

También existen sistemas de administración de red (NMSs - Network Managment System's) que automáticamente descubren todos los dispositivos desplegados en la red y reportan sus propiedades, como podemos observar en el trabajo [36].

[1] https://www.w3.org/TR/rdf-sparql-query/ último acceso acceso 18-11-16

El uso de ontologías en el dominio de la administración de redes ha sido sujeto de una variedad de investigaciones, muchas es éstas se han concentrado en redes homogéneas, como las cableadas, o WSNS. Por tanto, el uso de diferentes modelos de administración requiere interoperabilidad entre ellas en orden de tener una vista unificada de la red.

Una ontología común permite que la información sea compartida e integrada desde distintas fuentes y entre diferentes sistemas de administración. Por ejemplo, la ontología MonONTO [37] incluye información para aplicaciones en internet, incluyendo calidad del servicio, monitoreo del ambiente y perfiles de usuario.

Debido a las masivas cantidades de interacción humana y conocimiento experto requerido, la configuración del administrador puede ser el administrador de red quien toma la mejor ventaja de las ontologías. Debido a que una ontología puede reducir la cantidad de interacción necesaria para configurar las tareas complejas en una red de una sola capa.

Un beneficio de los nuevos NMS es que dependiendo de la información rutinariamente recolectada desde los dispositivos cableados a través de SNMP es que no requiere ninguna modificación al software que se está ejecutando en los dispositivos. Por lo que la meta de la investigación fue minimizar el *overhead* de procesamiento para esos dos componentes mientras de maximiza el beneficio de incorporar ontologías en la NMS. Además, las instancias solo es necesario agregarlas a la base del conocimiento cuando los dispositivos desplegados son descubiertos.

En décadas recientes, gracias al avance de los sistemas empotrados y las tecnologías inalámbricas, las redes inalámbricas de sensores (WSN por su siglas en ingles *Wireless Sensor Networks*) se han vuelto ampliamente utilizadas [38]. En el trabajo [39], se expone que las WSN usualmente consisten de un conjunto de nodos de sensores inalámbricos (desde unas decenas hasta unos cientos), los cuales adquieren, almacenan, transforman y comunican datos utilizando tecnologías inalámbricas.

Las WSN recolectan grandes cantidades de datos heterogéneos (lluvia, temperaturas, niveles de agua, etc.). Sin embargo, por su naturaleza, los nodos de WSN se caracterizan por los limitados recursos: energía, poder de cómputo y capacidad de almacenamiento. Además, los nodos con suficiente energía pueden comunicar todos los datos adquiridos con la misma frecuen-

cia con la que son adquiridos. Por el contrario, los nodos WSN que poseen un bajo nivel energético no pueden comunicarse de esa manera.

Por consiguiente, los nodos WSN deben de poder adaptarse a su contexto. El contexto es un conjunto de información que el nodo puede obtener de su ambiente, un nodo toma en cuenta su contexto para mejora su tiempo de vida y en consecuencia el funcionamiento en general de la red.

Las ontologías son una solución para describir a los sensores, sus datos y contexto, también definen vocabularios de metadatos. A través de las ontologías, será posible al asignar a los datos una descripción de su adquisición y políticas de comunicación en orden de mejorar la integración de los datos del sensor. Además, con el uso de reglas semánticas, las ontologías son componentes de sistemas razonadores.

Las WSN son ampliamente utilizadas para obtener datos de varios dominios. Todos esos cambios de estados pueden ser llevados a cabo por un sistema basado en reglas es cual es un tipo de sistema experto. El razonador puede ser modelado utilizando reglas de decisión a un conjunto de hechos. Ciertamente, algunos tipos de ontologías definen entidades utilizadas en motores basados en reglas, en este escenario, los nodos WSN deben:

Interoperar con otros nodos, los razonadores deben ser capaces de adaptarse a su contexto: el estado del WSN y el estado del fenómeno observado y deben cambiar inteligentemente la adquisición y comunicación de las políticas de datos bajo supervisión de ambos la red y el *Decision support system* (DSS).

Una red inalámbrica de sensores (WSN) consiste de pequeñas entidades llamadas nodos WSN. Estos nodos tienen recursos energéticos limitados, memoria y capacidad de cómputo. Un nodo WSN está compuesto de varios dispositivos como el dispositivo de comunicación, el dispositivo de censado, también llamado sensor, medidas de la propiedad o características de interés.

La ontología SSN (está basada en el patrón de diseño ontológico *Stimulus-Sensor-Observation*) [40], un dispositivo de censado es un dispositivo, una especialización del sistema. Un sistema está compuesto de varios sistemas. Un red de sensores puede estar compuesta de varios dispositivos como una batería y un sensor. En la ontología WSSN, un *Wireless Sensor Network*

*Node* está compuesto de cuatro dispositivos: processingDevice, CommunicatiDevice, EnergyDevice y ssn: SensigDevice. Una WSN es también un sistema compuesto de varios nodos WSN.

El dispositivo de censado adquiere los datos. El dispositivo de procesamiento produce nuevos datos, que pueden ser una agregación de un conjunto de datos adquiridos. El dispositivo de comunicación comunica los datos adquiridos por los dispositivos de Censado o el dispositivo de procesamiento.

Los artículos citados anteriormente nos permiten concebir una nueva idea, la cual será la base primordial del presente trabajo de tesis. Estos artículos representan sólo una muestra del incremento en el cuerpo de la investigación de los servicios web semánticos. Representando una fusión de áreas tan diversas como son: los de sistemas basados en agentes, lógica de descripción, mallas computacionales, y servicios web, Los servicios web semánticos serán parte esencial de la realización de servicios web semánticos ubicuos.

# Capítulo 3

# Marco teórico

En esta sección vamos va comenzar por el marco teórico que sustenta la base teórica en la cual esta desarrollado el presente trabajo. A continuación se presenta una introducción general de la teoría necesaria para la comprensión de la propuesta.

## 3.1. Cómputo ubicuo contexto

Como se ha mencionado anteriormente el cómputo ubicuo nace de la visión de Mark Weiser, y el cómputo ubicuo contiene a una disciplina conocida como cómputo sensible al contexto la cual es una disciplina de software que permite a la infraestructura ubicua proveer de manera proactiva servicios a los usuarios que tengan interacción bajo la misma situación contextual que el sistema.

Por tanto se puede decir que el contexto es un conjunto de circunstancias en que se sitúa un hecho ya que desde el punto de vista de computación los espacios sensibles e inteligentes, como los espacios computacionales que "sienten" y "reaccionan" a los estímulos del entorno en que se encuentra desplegado. También se debe reducir la interacción entre el usuario y el sistema haciendo uso del contexto por medio de nuevas interfaces de comunicación, las cuales deben pasar al trasfondo de la vida del usuario por medio computadoras empotradas en objetos que no demanden demasiada atención del usuario.

El contexto en cómputo ubicuo hace uso principalmente de computadoras, sensores y reglas, ya que los sensores pueden obtener datos del ambiente

e incluso con algunos más complejos se puede "identificar" al usuario tiene interacción indirecta con el sistema. Y los sistemas de reglas ayudan al modelado del comportamiento y reacciones que tiene el sistema a los estímulos de modo que estas reglas son parte de la "inteligencia" del sistema ubicuo que puede ser sensible al contexto o consiente del contexto.

Los sistemas ubicuos pueden ayudar a las actividades diarias del usuario, y son igualmente aplicables al trabajo, a la administración del hogar, e incluso se puede aplicar a juegos. Citando lo que dijo Weiser en su artículo [1] "Las tecnologías más profundas son aquellas que desaparecen. Se tejen a si mismas en la fabricación de la vida diaria hasta que se vuelven indistinguibles."

La esencia de la visión de Weiser es que las computadoras empotradas se puedan comunicar unas con otras y con la infraestructura que las rodea, al mismo tiempo que coordinan su operación para dar soporte a una amplia variedad de practicas de trabajo diarias.

Las aplicaciones consientes del contexto comprenden al ambiente en que son utilizadas, y adaptan su modo de operación para proveer la mejor experiencia de usuario posible. El contexto de un usuario o dispositivo es difícil de modelar ya que tiene muchas dimensiones, como la ubicación, la identidad de los dispositivos cercanos, quién más esta presente, la orientación, entre otras variables físicas, como el tiempo, movimiento, sonido y temperatura, muchas de las cuales pueden ser medidas a través de sensores.

## 3.2. Hardware reconfigurable

El hardware reconfigurable es una arquitectura que combina la flexibilidad del software con el alto rendimiento del hardware mediante el procesamiento de arreglos de compuertas programables. La principal diferencia en comparacion a los microprocesadores ordinarios es la capacidad de llevar a cabo cambios en el flujo de los datos y en el flujo de control. Y a diferencia del hardware dedicado, como los circuitos integrados, es abre la posibilidad de adaptar el hardware en tiempo de ejecución al "cargar" un circuito nuevo en el hardware reconfigurable.

El concepto de cómputo reconfigurable ha existido desde la década de

los 60's, cuando Gerald Estrin publica una propuesta del concepto de una computadora hecha con un procesador estandardizo y un arreglo de hardware reconfigurable.El procesador principal controlaría el comportamiento del hardware reconfigurable, de modo que más tarde se encolaría para llevar a cabo una tarea especifica, como el procesamiento de una imagen, igual de rápido que una pieza de hardware dedicado.

Una vez que se haya llevado a cabo la tarea, el hardware puede ser nuevamente ajustado para realizar otra tarea. Esto dio lugar a una estructura de equipo híbrido que combina la flexibilidad del software con la velocidad del hardware.

Más tarde en las décadas de los 80's y 90's se suscita el renacimiento de esta área de investigación con el surgimiento de muchas arquitecturas reconfigurables propuestas y desarrolladas en la industria y el mundo académico,Los diseños eran factibles gracias al constante avance en la tecnología de silicio que permite implementar diseños complejos en un chip.

### 3.2.1. Arduino

Arduino[1] es un proyecto de plataforma abierta(open-source) fundado en el año 2006 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea(Italia). Está basado en una plataforma de prototipado basada en hardware y software de fácil uso.

Las placas arduino son capaces de leer entradas analógicas o digitales, también son capaces de encender un sensor, un botón presionado por un dedo, o un mensaje de twitter e incluso es capaz de convertir una entrada en salida de datos para así activar un led, motores de baja tension en corriente directa, e incluso publicar algo en línea.

Todo esto es gracias al uso de un microcontrolador y al conjunto de instrucciones que se le envían, para poder lograr en envío de las instrucciones se debe utilizar el lenguaje de programación de arduino[2] el cual está basado en el lenguaje de programación C/C++. Este lenguaje de programación se liga con la biblioteca *'AVR libc'* y permite el uso de casi cualquiera de sus instrucciones, es decir, cuenta con estructura, variables y funciones para facilitar al desarrollo de aplicaciones.

[1] https://www.arduino.cc última vez consultado 11-10-16
[2] https://www.arduino.cc/en/Reference/HomePage última vez consultado 11-10-16

Arduino cuenta con una gran cantidad de productos diseñados para diferentes propósitos, como son: a nivel de principiantes con kits de desarrollo básicos, características mejoradas al incrementar la capacidad para procesar un número mayor de entradas o salidas, agregando la capacidad de conexión a Internet por medio de una conexión ethernet alámbrica o inalámbrica (Wi-Fi), y también la posibilidad de crear dispositivos *wearable*, es decir, dispositivos que podemos llevar puestos como si fuese ropa o un accesorio como un reloj de pulsera.

### 3.2.2. Programa ejemplo de arduino

A continuación se muestra un ejemplo de un programa[1] de arduino. El primer ejemplo es un sencillo programa para leer una entrada analógica (*véase script 3.1*) haciendo uso de un pin de entrada y mostrando lo que se lee a través del monitor serial.

```
1  /* AnalogReadSerial
2  Reads an analog input on pin 0, prints the result to the
       serial monitor.
3  Attach the center pin of a potentiometer to pin A0, and the
       outside pins to +5V and ground.
4
5  This example code is in the public domain. */
6
7  // the setup routine runs once when you press reset:
8  void setup() {
9  // initialize serial communication at 9600 bits per second:
10   Serial.begin(9600);
11 }
12 // the loop routine runs over and over again forever:
13 void loop() {
14 // read the input on analog pin 0:
15   int sensorValue = analogRead(A0);
16 // print out the value you read:
17   Serial.println(sensorValue);
18   delay(1);        // delay in between reads for stability
19 }
```

Script 3.1: Arduino: lectura analógica

[1]Los programas fueron tomados de IDE de arduino y también están disponibles en el sito de https://www.arduino.cc última vez consultado 11-10-16

La parte del código *void setup()* tiene como objetivo ayudar al usuario a definir que pines serán destinados como entrada o salida, en caso de ser entradas que tipo de entradas ya sea analógicas o digitales, e incluso si se agregan más bibliotecas para hacer uso de dispositivos adicionales en esta sección del programa se pueden inicializar los dispositivos que requieran iniciar para que puedan utilizar de manera adecuada. Cabe destacar de que por omisión los pines de cualquier arduino están definidos como entradas, además el primer programa que viene pre cargado es *Blink*, que enciende y apaga el led que esta integrado a la placa y conectado al pin 13 del arduino.

En la función (*void loop()*) es donde se ejecuta el código en un ciclo infinito ademas es donde se pueden declarar las variables para reducir el uso de memoria de programa. En este programa se declara la variable *sensorValue* de tipo entero y se indica que es igual a la entrada *A0* de la sección de pines analógicos. Hay que mencionar que arduino cuenta con una cantidad de pines tanto analógicos como digitales y que cada una de las secciones es dedicada, es decir, la sección digital no debe ser utilizada para hacer lecturas de datos analógicos y viceversa.

Por último la línea *Serial.println(sensorValue);* es la encargada de enviar al monitor serial los dato que fueron leídos a través del pin A0 y son mostrados en pantalla, para aquellas personas que son principiantes en arduino hay que recordar que arduino cuenta con una interfaz de conversión de puerto serial a USB, lo que hace posible conectar al arduino a través de un puerto USB y así obtener datos e incluso programarlo son la necesidad de tener un programador externo para la placa.

## 3.3. Cómputo móvil

El cómputo móvil o computación móvil es un campo de investigación relativamente nuevo con poco más de tres décadas de historia. Esto jha dado lugar al nacimiento de un área de diseño de interacción móvil en las intersecciones entre la computación móvil, ciencias sociales, la interacción humano-computadora, diseño y experiencia de usuario.

El cómputo móvil contribuye a la omnipresencia del cómputo ubicuo en la civilización moderna. Junco con la proliferación de las tecnologías empotradas y fijas en la sociedad, los dispositivos móviles tales como teléfonos

celulares, teléfonos inteligentes y otras tecnologías portátiles han creado un ambiente ubicuo en el que hay mas dispositivos de cómputo que personas.

De modo que abre la posibilidad a estos dispositivos para adaptarse y servir como apoyo en la ejecucion de tareas en la vida personal o laboral de los usuarios, por tanto es un reto para los desarrolladores, y "como consecuencia del cómputo ubicuo, el diseño en la interacción está a punto de convertirse en una de las principales vertientes del siglo XXI" [41].

Desde los inicios de la computación, siempre han existido aspiraciones para crear hardware de cómputo más pequeño, y desde entonces las computadoras se han vuelto más accesibles, ya que ha habido un gran interés en la potabilidad de los dispositivos por parte de los usuarios [42].

La historia del cómputo móvil se puede dividir en siete áreas, cada una caracterizada por su enfoque tecnológico en particular, los diseños de interacción, y el por que conducen a cambios fundamentales en el diseño y aplicación de dispositivos móviles sin olvidar el uso. A continuación se presentan las siete áreas del cómputo móvil, que proporcionan una buena visión general de la herencia sobre la cual se construye actualmente la investigaciń del cómputo móvi.

- Portabilidad
- Miniaturización
- Conectividad
- Convergencia
- Divergencia
- Aplicaciones
- Ecosistemas digitales

El enfoque de la portabilidad de caracteriza por la búsqueda en la reducción del tamaño del hardware para permitir el desarrollo de equipos que pudiesen ser transportados de un lado a otro fćilmente.

La miniaturización se enfoca en crear dispositivos móviles de factores más pequeños permitiendo así el uso de los dispositivos móviles mientras el

usuario se encuentra en movimiento.

En la conectividad se trata de desarrollar dispositivos y aplicaciones que permitan estar a los usuarios estar siempre en línea y comunicarse a través de redes inalámbricas mientras están en movimiento.

La convergencia trata acerca de la integración de dispositivos móviles digitales, tales como los asistentes personales (PDA), teléfonos móviles, reproductores de música, cámaras digitales, vídeo juegos, entre otros, en los dispositivos híbridos.

La divergencia toma un enfoque opuesto al diseño de integración al promover dispositivos móviles con funciones especializadas en lugar de generalizadas.

Para el área de las aplicaciones se trata de desarrollar aplicaciones que puedan ser ejecutadas en los dispositivos móviles, permitiendo el acceso fácil y amigable a dicho contenido o aplicación.

Finalmente en el área emergente de ecosistemas digitales se investigan los conjuntos de tecnologías ubicuas en las que, los sistemas móviles se están volviendo parte ellas.

### 3.3.1. Android

Es un sistema operativo basado en el núcleo de Linux. Fue diseñado en un principio para dispositivos móviles como los teléfonos inteligentes y tablets; En el presente también se puede encontrar en relojes inteligentes, televisores y automóviles. Android es una plataforma abierta de software que busca unificar al hardware y al software fue fundado inicialmente por Andy Rubin, Rich Miner, Chris White y Nick Sears en Palo Alto, California, EE.UU. en el año de 2003 y para 2005 es adquirido por Google.

Cuenta con cuatro capas en su arquitectura y cada una de ellas contiene diferentes caracteristicas de acuerdo a la version. Además es posible ejecutar una maquina virtual del java especializada, y diseñada específicamente para android y optimizada para dispositivos que funcionan con batería como los móviles ya que cuentan con recursos limitados en capacidad de procesamiento y memoria.

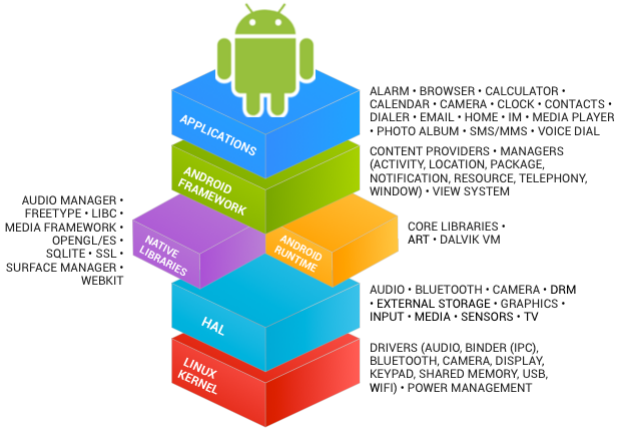


Figura 3.1: Android framework details

Las cuatro capas de android son: aplicaciones, framework de aplicaciones, bibliotecas y a éste mismo nivel se encuentra el Android Runtime, en la siguiente capa se encuentra Hal, y más al fondo podemos encontrar el núcle de Linux *véase figura 3.1.*

Android fue inicialmente originado por un grupo de compamĩas conocido como *Open Handset Alliance*, liderado por Google. Hoy en día, los mienbros originales y varias compañias tienen un gran interés en android. Esas compañias han reservado recursos de ingeniería para mejorar a los dispositivos android y para traer dispositivos android al mercado.

Las comapñias que se han interesado en android es por sus meritos puesto que los creadores creen que un plataforma abierta es necesaria. android es explícita e intencionalmente una plataforma abierta. Por lo que la personalización no controlada, puede por supuesto, llegar a alguna implementación incompatible, por lo que se creó al *Android Compatibility program*, el cual se encarga de decir lo que significa ser "Android Compatible" y que es lo que requieren los desarolladores para alcanzar ése meta.

## 3.4. Ontologías

La web semántica es una visión para el futuro de Web en la cual la información tendrá un significado explícito, siendo fácil para las computadoras el automáticamente procesar e integrar la información disponible en la web.

En el trabajo de (DeborahL.McGuinness [43]) en le año 2004. Los OWLS (*Ontology Web Languages*) están destinados a ser utilizados cuando la información contenida en documentos necesite ser procesada por aplicaciones, como en situaciones opuestas en las cuales el contenido sólo necesita ser presentado a los humanos. OWL (*ontology Web Laguage*) puede ser utilizado para representar el significado de los términos en vocabularios y relaciones entre dichos términos. Esta representacién de los términos y sus interrelaciones es llamada una Ontología.

### 3.4.1. *Extensible Markup Languaje* (XML)

XML proviene de *eXtensible Markup Language* ("lenguaje de marcas extensible). Es un lenguaje que se utiliza para "decir" algo de otro por medio de etiquetas, fue desarrollado por la W3C (World Wide Web Consortium), que elabora estándares y recomendaciones para la World Wide Web.

Es una adaptacion del Standard generalized markup language (SGML), el cual es un lenguaje para la organización y el etiquetado de documentos. Por tanto XML no es un lenguaje es sí mismo, es un sistema que permite definir etiquetas de acuerdo alas necesidades del usuario. el metalenguaje aparece como un estándar que estructura del intercambio de información entre diferentes plataformas.

XML surge como un lenguaje para sustituir a HTML, de modo que la principal diferencia radica en que HTML esta orientado a la presentación de datos, mientras que XML se enfoca en los datos mismos, por lo que cualquier software puede trabajar mejor con XML.

Al hacer uso de XML los programas producen "datos estructurados" y dado que el número de aplicaciones basada en documentos XML es bastante numeroso de modo que la palabra documento no se refiere solo a documentos tradicionales, sino tambien a otro tipo de formatos de datos en XML. Esto incluye gáficos vectoriales, transacciones de e-commerse, ecuaciones

matemáticas, objetos de metadatos, servidores de Aplicaciones entre otras clases de información estructurada.

XML provee una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas al significado de esos documentos. XML Schema es un lenguaje para restringir la estructura de los documentos XML y también para extender a XML con tipos de datos.

Los principales usos y aplicaciones de XML son las siguientes:

- Sitios Web: permite separa el contenido y presentación, de moto que los mismos datos se puedan presentar de distintas formas sin esfuerzo.
- Comunicación entre aplicaciones: representación de datos simple, fácil de trasmitir por la red.
- configuración: permite representar a los datos de modo simple y estándar.
- Ofrece mecanismos versátiles para mostrar datos. Es posible representar un documento XML en los navegadores Web de forma sofisticada a diferencias de HTML, ya que XML no solo proporciona sintaxis, sino también semántica.
- Los motores de búsqueda "inteligentes" pueden realizar una búsqueda más precisa, ya que XML etiqueta por su significado de modo preciso y se puede localizar de una forma más clara.
- Intercambio de información entre sistemas diversos y heterogéneos.

Los documentos XML están compuestos por unidades de almacenamiento llamadas objetos o entidades (entities), que contienen datos analizados (parsed) o sin analizar (unparsed). los datos analizados se componen de caracteres, de los cuales algunos forman parte de los datos del documento (data) y el resto forman las etiquetas (markups) o marcas. Las etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento.

Para "leer" los documentos XML y acceder a su contenido y estructura, se debe utilizar un software procesador de XML. Cualquier aplicación que utilice XML necesita dicho modulo o procesador XML (parser). Dicho modulo leer los documentos y proporciona accedo a su contenido y estructura.

### 3.4.2. *Resource Description Framework* (RDF)

RDF es un modelo de datos para objetos ("recursos") y relaciones entre éstos, provee semánticas simples para este modelo de datos, y esos modelos de datos pueden ser representados en una sintaxis XML. RDF Schema es un vocabulario para describir las propiedades y clases de los recursos RDF, con una semántica para la generalización de jerarquías de tales propiedades y clases.

Mientras que XML es un lenguaje para modelar datos, RDF es un lenguaje para especificar metadatos. XML falla en la estabilidad de los datos puesto que le orden de los elementos es antinatural y su mantenimiento difícil y costoso, por el contrario Resource Description Framework (RDF) permite la operación entre aplicaciones que intercambian información comprehensible por la página web, para proporcionar una infraestructura que de soporte a actividades de metadatos.

El lenguaje RDF es muy útil en situaciones en las que la información necesita ser procesada por aplicaciones que intercambian información legible por la máquina, más que por los humanos. Provee un marco de trabajo para expresar dicha información y para intercambiarla entre distintas aplicaciones mediante una serie de analizadores RDF. Puede utilizarse en diferentes áreas como la recuperación de recursos en los motores de búsqueda web, robots y agentes inteligentes o en una colección de documentos, ente otros.

El modelo de RDF está basado en tres elementos fundamentales:

- Recursos: todo aquello descrito por medio de expresiones RDF, Un recurso también puede ser un objeto que no sea directamente accesible desde la Web.
- Propiedades: son aspectos específicos, características atributos o relaciones que pueden ser utilizadas para describir un recurso. Cada propiedad tiene un significado especifico.
- Declaraciones: una declaracion RDF es una propiedad o relacion que puede ser utilizada para un recurso especifico y esta compuesta por tres partes individuales:
    1. Sujeto: recuro
    2. Predicado: propiedad

3. Objeto: valor de la propiedad, puede ser otro recuro o una cadena simple de caracteres o de otros tipos de datos primitivos definidos por XML.

Lo que marca la diferencia entre RDF y XML es que RDF fue diseñado para representar conocimiento en un mundo distribuido, que sea para conocimiento y no para datos significa que es concebido particularmente con significado, todo lo mencionado haciendo uso de RDF significa algo. Los estándares construidos por encima de RDF como: RDFS y OWL, agregan semántica a RDF para la elaboración de inferencias lógicas a partir de datos.

Formalmente los nombres para sujetos, predicados y objetos deben ser identificadores uniformes de recursos (URIs). Las URIs pueden tener el mismo formato que una dirección web, donde la URI es el nombre global para alguna identidad. No todas las URIs nombran a sitios web, esa es la diferencia entre una URI y una URL. Las URL son URIs que nombran cosas en la web que pueden ser recuperadas.

### 3.4.3. *Ontology Web Language* (OWL)

OWL por su siglas en inglés *(Ontology Eeb Language)*, es un lenguaje de marcas que sirve para publicar y compartir daros haciendo uso de ontologías. Tiene como objetivo facilitar el modelado de marcas construido sobre RDF y codificado en XML.

El lenguaje OWL es un lenguaje web semántico diseñado para representar conocimiento rico y complejo acerca de objetos, gripos de ongetos y relaciones entre objetos. OWL es un lenguaje basado en lógica computacional de modo que el conocimiento expresado en OEL puede ser explotado por programas de computadora.

OWL agrega más vocabulario para describir propiedades y clases, relaciones entre clases, cardinalidad, igualdad, tipificación de propiedades más ricas, características de propiedades, y clases enumeradas.

Cuenta con tres versiones las cuales son:

- OWL Lite da soporte a aquellos usuarios que ante todo necesitan una clasificación jerárquica y restricciones simples.
- OWL DL da soporte a los usuarios que buscan la máxima expresividad mientras retienen la completitud computacional y decidibilidad. OWL

es también llamada debido a su correspondencia con la lógica descriptiva, un campo de investigación que ha estudiado la lógica desde la fundación formal de OWL.

- OWL FULL es entendido por los usuarios que buscan la máxima expresividad y la libertad sintáctica de RDF sin garantía computacional. La elección entre OWL Lite y OWL DL depende de la extensión que requiere el usuario de los constructores más expresivos provistos por OWL DL.

La elección entre OWL DL y OWL Full depende principalmente de la extensión en la cual los usuarios requieren la facilidad provista por el metamodelo para el esquema RDF. Además, OWL Full puede ser visto como una extensión de RDF, mientras que OWL DL y OWL Lite pueden ser vistos como extensiones de una vista restringida de RDF.

Los prefijos de RDF: o RDFS: son utilizados cuando los términos ya están presentes en RDF o RDF Schema. De otro modo los términos son introducidos por medio de OWL. OWL Lite utiliza solamente algunas de las características del lenguaje OWL y tiene más limitaciones en el uso de las características que OWL DL u OWL Full. Las siguientes características se incluyen de OWL Lite a RDF Schema.

OWL Full extiende esta restricción para permitir descripciones de clases complejas arbitrarias, que consisten de clases numeradas, propiedades restrictivas, y combinaciones booleanas. Además, OWL Full permite que las clases sean utilizadas como instancias.

## 3.5. Servicios Web

Un servicio web es un programa accesible desde distintas plataformas que provee al usuario (el solicitante) acceso a una aplicación basada en web (proveedor) tanto de forma directa como a través de componentes terciarios [19]. Una vez disponible, tales metadatos pueden ser procesados a través de un razonador lógico en orden de interpretar las descripciones del servicio basándose en su significado semántico, en lugar de solo su sintaxis.

Básicamente un servicio Web es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. distintas aplicaciones

de software desarrolladas haciendo uso de distintos lenguajes de programación y ejecutadas sobre distintas plataformas, pueden consumir servicios web para intercambiar datos entre redes de dispositivos como los es Internet.

Las anotaciones en los esquemas WSDL (*Web Service Description Language*) y XML pueden ser utilizadas para publicar un servicio en un registro y también descubrir, componer e invocar servicios web. Las anotaciones semánticas hacen referencia a un concepto en una ontología o mapeo a un documento. El mecanismo de anotación es dependiente de la ontología y de los lenguajes de mapeo.

Las anotaciones en documentos XML, pueden ser utilizadas por varios componentes del esquema XML.

- Tipo simple de anotaciones XML: en el esquema de modelos de componentes XML/WSDL, una referencia a un modelo no vacío ,en un nivel superior es asignado a través de una propiedad de modelo Referencia.
- Tipo de anotación complejo XML: los tipos complejos pueden ser anotados utilizando el modelo Referencia igualmente a nivel inferior como en el superior. Las anotaciones en el nivel inferior son utilizadas cuando los miembros de un tipo complejo corresponden directamente a los conceptos de un modelo semántico. Las anotaciones en el nivel superior son utilizadas para anotar tipos complejos como un todo.
- Elemento de anotación XML: un modelo de referencia no vacío en un nivel superior de declaración de elementos WSDL es asignado a través de la propiedad del modelo Referenciado. Los valores del modelo referenciado asociados a la declaración de un elemento heredan dela propiedad del modelo referenciado asociado al tipo de elemento.
- Atributos de anotación XML: un modelo referenciado no vacío en el atributo de declaración a nivel superior está asignado vía un modelo de propiedad modelo referenciada.

Para WSDL 1.1, la principal anotación incluye lo siguiente.

- Anotación *portType*: un *portType* está denotado de la misma manera que una interfaz WSDL 2.0.
- Denotación de entrada y salida: los esquemas XML de entrada y salida pueden ser denotados de forma similar que en WSDL 2.0.

- Denotación de fallas: una falla en WSDL 1.1 está definida idénticamente para una entrada o salida. Denotación de operación: WSDL 1.1 no permite el atributo de extensión, una operación es denotada por agregar extensiones del elemento atributo (*attrExtensions*) como un hijo del elemento operación.

Para WSDL 2.0, los tipos de anotación principales incluyen los siguientes.

- Interfaces de anotación: un elemento de la interfaz WSDL puede ser anotado utilizando un modelo referenciado para proveer una liga a un concepto en una interfaz semántica descriptiva.
- Operadores de anotación: la anotación es el elemento de la operación que lleva una referencia a un concepto en un modelo semántico ya que provee una descripción de alto nivel en la operación, especifica los aspectos de su comportamiento o incluye otras definiciones semánticas.
- Anotación de fallas: la anotación del elemento de fallas lleva una referencia a un concepto en un modelo semántico que provee una descripción de alto nivel de la falla y puede incluir otras definiciones semánticas.

Acorde a la especificación de la IEEE, una visa de sistema es una representación de un sistema completo desde varias perspectivas relacionadas. En el caso de los *Semanic Web Services* (SWS), los actores/interesados más representativos con distintas perspectivas son el solicitante del servicio, quien invoca los servicios, el proveedor que responde a las peticiones, y el registro en el cual los servicios pueden ser publicados o anunciados.

Se podría decir de una manera más clara que un servicio web es una función que diferentes equipos o servicios utilizan; es decir, solo envían parámetros al servidor en que se encuentra alojado el servicio web y éste drá respuesta a la petición. Algunas de las ventajas de utilizar de los servicios web son las siguientes.

- Permiten la comunicación entre aplicaciones de software independientemente de sus propiedades o de las plataformas en la que ejecuten.
- Fomentan los estándares y protocolos basados en texto, de modo que facilitan el acceso a su contenido.

- Al apoyarse en el protocolo de HTTP, los servicios Web pueden aprovechar los firewall sin tener que cambiar las reglas del mismo.
- Permiten la integración de servicios y software que pueden estar localizados en Internet.
- Permiten la comunicación y operación de distintas plataformas haciendo uso de los protocolos estandarizados. cuyas especificaciones son gestionadas por la W3C.

Los servicios Web son independientes del lenguaje de programación y de la plataforma gracias a los estándares de servicios Web, su principal característica es la interoperabilidad y la extensibilidad de modo que servicios web simples se pueden combinar con otros para "crear" servicios más sofisticados. Por tanto un servicio web debe contener las siguientes características:

- Tiene que ser accesible a través de la Web. Por tanto debe utilizar los protocolos estandarizados de transporte como HTTP, y enviar los mensajes codificados en un lenguaje estándar que sea reconocido por cualquier cliente que desee consumir el servicio.
- Un servicio Web debe contener una descripción de sí mismo, es decir, una aplicación puede saber cual es la función de un determinado servicio Web, y cual es su interfaz, de modo que cualquier aplicación pueda utilizarlo de manera automática, sin la intervención del usuario.
- Debe poder ser localizado. Se debe tener algún mecanismo que permita encontrar algún servicio que lleve a cabo una función determinada.

La arquitectura de los servicios web es orientada a servicios ya que permite crear una definición abstracta de un servicio, proporcionar una implementación concreta de dicho servicio, publicar y localizar un servicio, seleccionar una instancia de un servicio, y utilizar dicho servicio con la ventaja de la interoperacion.

Las implementaciones concretas del un servicio pueden desacoplarse a nivel lógico y de transporte. La figura 3.2 presenta el diagrama de la arquitectura orientada a servicios.

El proveedor del servicio define la descripción abstracta de dicho servicio utilizando el lenguaje de descripción de servicios web (WSDL). En seguida se crea un servicio concreto a partir de la descripción abstracta del servicio,

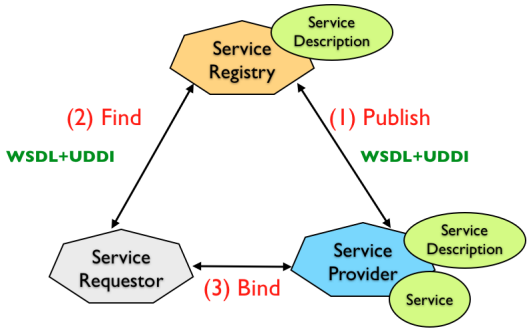


Figura 3.2: arquitectura de los servicios web

produciendo así una descripción concreta del servicio en WSDL.

Dicha descripción concreta puede entonces publicarse en un servicio de registro como en UDDI *(Universal Description, Descovery and Integration)*. El cliente de algún servicio puede entonces hacer uso de un servicio de registro para localizar una descripción de un servicio, a partir de la cual podrá seleccionar y utilizar una implementación concreta de dicho servicio.

Las teconogías básicas para los servicios web están basadas en XML,ya que son fundamentales para el desarrollo de servicios web. Estas tecnologías son independientes del sistema operativo como del lenguaje de programación utilizado para implementar los servicios. Por tanto, serán utilizadas para cualquier servicio Web, independientemente de la plataforma sobre la que se desarrollen los servicios.

# Capítulo 4

# Propuesta de solución

La red de sensores pretende resolver los problemas de adquisición de datos del ambiente al hacer uso de las tecnologías *open source* como Arduino, cabe mencionar que existen diferentes tecnologías del mismo tipo en el mercado puesto que permiten el crear prototipos de electrónica de forma rápida por lo que fue uno de los puntos fuertes en optar por este tipo de tecnología. Los datos que se pretende obtener del ambiente son aquellos que pueden llegar a ser de interés para el usuario en un ambiente académico, es decir, los datos relevantes que pueden ayudar al usuario a mejorar su experiencia dentro de la institución académica.

Entre los datos a obtener se tomaron en cuenta los siguientes: temperatura promedio, porcentaje de humedad, cantidad de luz externa e interna, número de personas que se encuentran dentro de un lugar determinado, y para casos especiales el conocer si uno o varios dispositivos se encuentran encendidos o apagados, además de la identificación de los individuos que se encuentran en algún lugar determinado.

El objetivo de obtener los datos anteriormente mencionados es el de otorgar un nivel de semántica a los datos obtenidos para convertirlos en información que sea fácil de entender por el usuario y que además el sistema pueda hacer uso de dicha información semántica para proveer un cierto grado de autonomía al sistema.

## 4.1. Sistema propuesto

A continuación se presenta la propuesta de solución a la hipótesis y al planteamiento de problema que se mencionaron en el primer capítulo de este trabajo, como punto de inicio en el presente capítulo, se debe comenzar por la adquisición de datos del ambiente, con la finalidad de conocer los distintos datos que se pueden obtener, es decir, para el caso de estudio propuesto se debe analizar el lugar y las distintas variables no solo ambientales que pueden llegar a existir, puesto que existe la posibilidad de que en un ambiente existan distintos contextos.

Para el caso de estudio se propone un ambiente académico a nivel universitario, dicho ambiente cuenta con distintos tipos de personal por lo que es deseable poder identificar a dicho personal para adecuar el tipo de información que se puede obtener, dado que no toda la información es útil o indispensable para el desarrollo del presente trabajo. Por tanto la mejor opción es montar una red de sensores inalámbricos los cuales nos auxilian en el proceso de obtener información del ambiente e incluso se puede tener la capacidad de identificar a las personas y el lugar en el que se encuentran. Por tanto se puede decir que la red inalámbrica de sensores es una de las capas del sistema en la arquitectura propuesta, la cual se explicará más adelante.

Para lograr convertir los datos en información es necesaria una capa que se encargue de otrogar semántica a la arquitectura por lo que es necesario hacer uso de las ontologías la cuales permiten agregar semántica e incluso relaciones y reglas a los datos, con el objetivo de crear una capa semántica que permita un manejo mas adecuado de la información y sus relaciones ya que al estar alimentada por los datos provenientes de la red de sensores es necesario identificar a que individuo pertenece la información proveniente de la red y así evitar inconsistencias en el sistema.

Algunas de la ventajas de hacer uso de las ontologías es que permiten agregar semántica a los datos que contienen, además de agrupar los datos en clases y cada clase puede contener a distintas clases que a su vez contienen a otras sub-clases o a individuos que cumplen con ciertas características especificas de la ontología, es decir, cumplen con reglas que comprenden el comportamiento de los individuos en el sistema. Las ontologías mejoran el rendimiento del sistema al clasificar a los individuos por sus características, también cabe destacar que la característica de modularidad (es decir, ontologías en módulos) permite agregar nuevas otologías a una meta-ontología

(ontología principal), para así agregar características al sistema y a la capa semántica sin la necesidad de cambiar todo su comportamiento ni perder la estructura original.

Como es deseable presentar a los usuarios la información que provee la meta-ontologá se propone el uso de los servicios Web ya que permiten acceder a la información del sistema prácticamente desde cualquier parte de la Internet, siempre y cuando se tenga una conexión disponible ya que es deseable acceder a los servicios desde cualquier lugar y tener acceso a la información que provee la meta-ontología, de modo que los servicios Web son la mejor opción ya que nos permiten hacer uso del lenguaje de programación JAVA para poder consultar a las ontologías.

También es posible cargar motores de reglas y razonadores para crear una estructura más "inteligente" ya que al crear las reglas y axiomas, y al cargar un razonador se abre el camino a la posibilidad de verificar la consistencia de los datos de la ontolgía, además es posible crear consultas por medio del motor de reglas/consultas, para obtener información relevante del sistema y así llevar acabo acciones que pudiesen ser del interés del usuario.

Por último hay que tomar en cuenta a los usuarios que no conocen el uso de los servicios Web ni su consumo por lo que se ha optado por crear una interfaz de usuario simple que ayude a los individuos a hacer uso de los servicios Web por medio de algún dispositivo. Para este objetivo se propone hacer uso de los *"Smart Phones"*, ya que han ganado popularidad y su omnipresencia es un punto clave en los sistemas ubicuos.

Hay que recordar que los teléfonos inteligentes han evolucionando de manera exponencial por lo que son una buena opción ya que cuentan con cierto nivel en poder de cómputo, lo que permite crear un cliente y consumir los servicios Web de forma que no se altere el funcionamiento ni la programación del teléfono, esto es posible, ya que permiten crear aplicaciones que se pueden ejecutar de manera independiente, y sin demasiada demanda de recursos del teléfono al aprovechar las tecnologías que integra como son: un microprocesador, memoria RAM, conexión inalámrica ya sea WiFi o red celular para acceso a Internet, entre otras.

## 4.2. Arquitectura propuesta

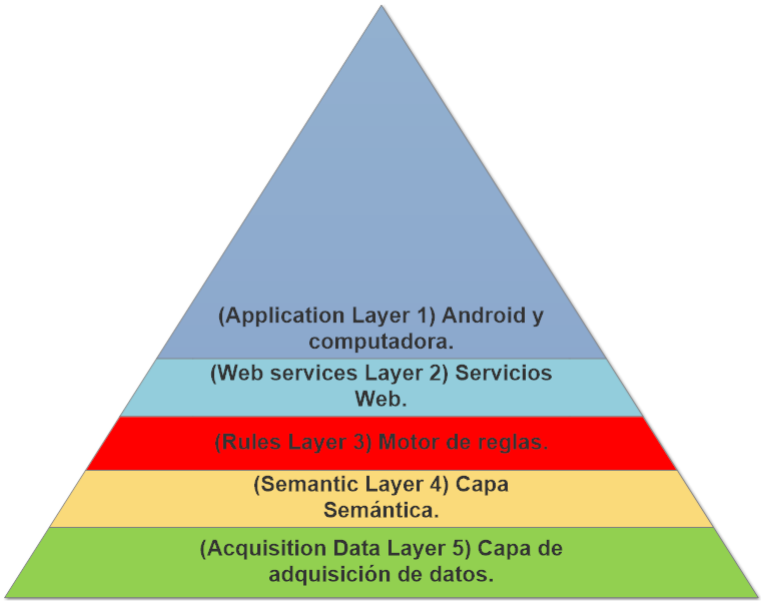


Figura 4.1: Capas de la arquitectura

La figura 4.2 muestra una representación gráfica más orientada hacia la forma de implementar la propuesta de solución, como se puede observar son visibles distintas capas delimitadas con colores y cada una de ellas tiene un nombre que la identifica en una posición más adecuada en la pirámide que fue diseñada para representar la arquitectura este trabajo *(véase figura 4.1)*, cada una de las capas comprende una sección del sistema completo y la forma en la que intercambian datos es delimitada con las flechas más obscuras y anchas.

La arquitectura propuesta consta de cinco diferentes niveles o capas y cada una de ellas involucra una tarea distinta para lograr un funcionamiento más armónico en el sistema.

1. La primer capa del sistema, es decir, la capa más alta en el sistema es la capa de aplicación, la representación de esta capa comprende el presentar la información que genera el sistema haciendo uso de alguna interfaz que sea fácil de entender y utilizar por el usuario final, puesto

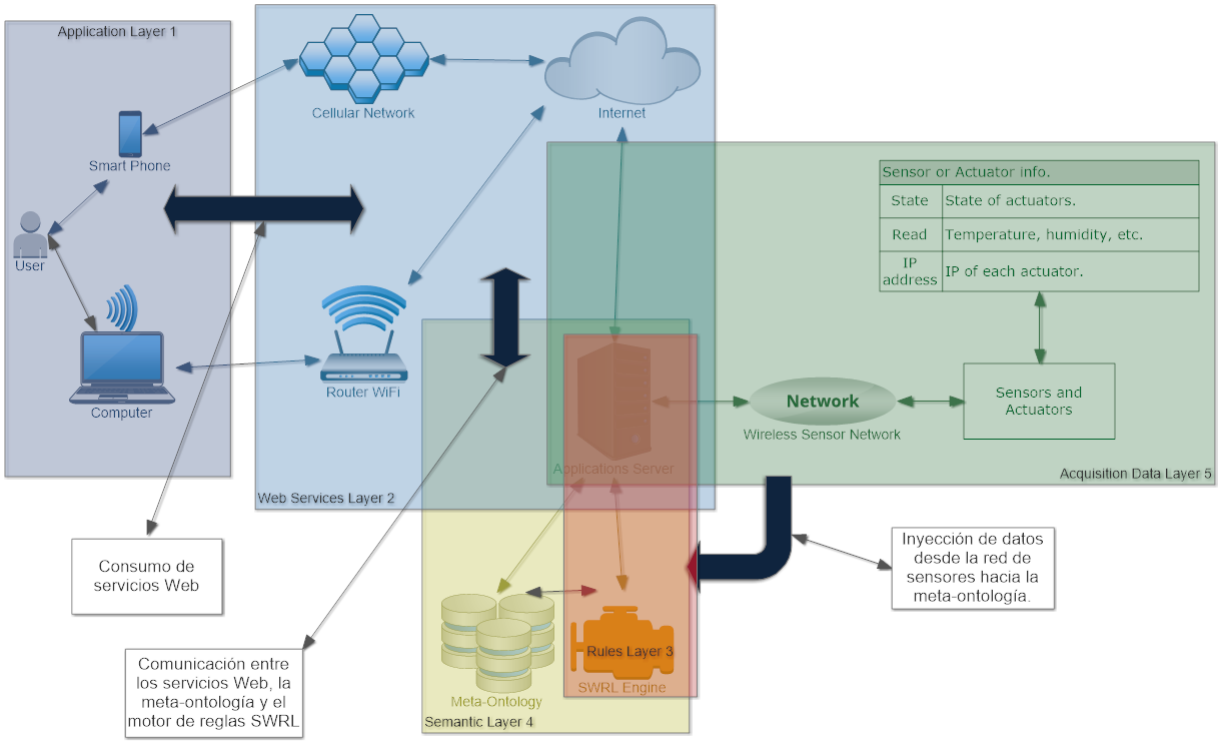


Figura 4.2: Arquitectura Gráfica

que es deseable que el sistema sea intuitivo y además cumpla con la principal característica de los sistemas ubicuos, es decir, que sea omnipresente sin acaparar la completamente atención del usuario.

Además es posible consumir los servicios desde un cliente que esté en una computadora con conexión a Internet o que se encuentre en la misma red que el sistema, por lo que el sistema puede ser consumido desde distintas plataformas sin comprometer la información que recibe del usuario como respuesta a la solicitud que se efectúa por parte del usuario hacia el servicio, como se puede observar el usuario puede hacer uso de cualquier plataforma que tenga conexión a Internet para comunicarse con la capa de servicios y consumirlos.

2. La segunda capa llamada capa de servicios Web es necesario, de preferencia, contar con una conexión a Internet es por éso que se muestran

dos tipos de conexión en la figura 4.2, ya que gracias a los avances tecnológicos es posible acceder a Internet desde la red celular por medio del servicio de datos o de forma tradicional con el servicio que se tenga en el lugar de trabajo ambas conexiones son inalámbricas y en el caso del la computadora puede ser alámbrica.

Esta capa contiene a los servicios Web que se desarrollarán ejecutando en un servidor remoto el cual debe tener la capacidad de manejar las peticiones que el usuario le solicite e incluso debe poder ejecutar programas demonio o monitores los cuales llevarán a cabo las acciones necesarias dentro de la red de sensores sin perder la capacidad de respuesta del servidor de aplicaciones, ya que la capacidad atender distintas peticiones es indispensable para aplicaciones Web.

3. La tercer capa de la arquitectura cuenta con el motor de reglas de SWRL *(Semantic Web Rules Languaje)*, este motor se encuentra en ejecución dentro del servidor y es invocado cada vez que llega alguna petición al los servicios que hacen uso de dicho motor. Ésta capa se encarga de ejecutar una serie de reglas e inferencias ya que el motor SWRL cuenta con una razonador y permite crear asociaciones e incluso puede inyectar los resultados de las reglas a la meta-ontología por medio de consultas que permiten generar nuevo conocimiento dentro de los individuos de meta-ontología.

4. La capa semántica que se encuentra en este caso en la cuarta posición, se encarga de crear la base del conocimiento sobre el cual se va a trabajar, para el caso de estudio en una institución académica es deseable usar módulos para los distintos contextos que pueden existir, por lo que hacer uso de las ontologías facilita el desarrollo de una meta-ontología más compleja y "fácil de manejar" por el motor de reglas, ya que al ser modular es posible agregar nuevas ontologías (nuevo conocimiento) al sistema, y abre la posibilidad de crear relaciones con los individuos ya existentes sin comprometer el rendimiento ni la posibilidad de expansión del sistema al agregar nuevas funcionalidades o características.

5. En la útima capa se encuentra la red de sensores y actuadores inalámbricos los cuales pueden estar asociados a distintos lugares en la ontología ya que cada uno de los dispositivos cuenta con distintas características técnicas como son: precision, velocidad de respuesta, capacidad de

lectura de distintas variables físicas, entre otras.

La red propuesta cuenta con distintos tipos de sensores y actuadores ya que por la gran variedad de interfaces que existen es deseable utilizar distintos tipos de éstos dispositivos para dar lugar a un disño de red de sensores más dinámico en cuanto a dispositivos que se pueden utilizar y controlar.

Además el uso del hardware reconfigurable Arduino permite montar servidores HTTP ligeros para así tomar lectura de los distintos dispositivos que se pueden controlar o leer desde la red y por medio algún programa demonio inyectar la información obtenida a la capa semántica para así lograr una adquisición de datos dinámica, sin la necesidad de generar grandes bases de datos y gracias a esto es posible economizar costos de infraestructura.

Cabe destacar que para el control de los dispositivos eléctricos que sea posible controlar se deberá hacer uso de electrónica de potencia para lograr dicha finalidad por lo que no todos los dispositivos pueden ser controlados, puesto que en algunos casos puede llegar a ser un método demasiado invasivo y no viable por el costo beneficio e incluso perdida del dispositivo.

También es importante tomar en cuenta las flechas que relacionan a algunas capas de la arquitectura, dichas flechas pueden ser visibles en la figura 4.2 en la que se hacen presentes dos flechas anchas que apuntan hacia ambos lados y una que sólo apunta en una sola dirección, cada una de estas flechas indica el intercambio de información entre capas del sistema y por tanto deben ser tomadas en cuenta. Para las capas que no cuentan con flechas pero se encuentran sobrepuestas se debe tomar a consideración que son capas que se encuentran fuertemente ligadas y por consiguiente el flujo de información es obligatorio ya que no es intercambio de información.

A continuación se presenta una breve explicación de la función que tiene cada una de las flechas azules que se presentan en la figura 4.2 y que se han comentado breve mente en el párrafo anterior.

- La primer flecha azul indica el flujo de los datos desde la red de sensores hacia la capa semántica, de modo tal que no se puede pasar por alto que es indispensable el obtener datos del ambiente y alimentar a la capa semántica para así modelar los cambios que suceden en el ambiente.

- La segunda flecha se indica el intercambio de información entre los servicios web y la capa semántica de modo que los servicios Web se comunican con la capa semántica para obtener la información solicitada por el cliente.

  Es de suma importancia aclarar que en este punto el servicio Web "crea" una instancia del motor de reglas que se encuentra en la tercer capa y lo utiliza para ejecutar una consulta a la capa semántica.

- La ultima flecha indica que hay un intercambio de información entre las dos capas más altas en el sistema de tal forma que la aplicación móvil o una computadora puede ser el cliente que va a consumir los servicios Web.

  Para esto debe enviar una petición con la información solicitada al servidor en el que se encuentran ejecutando los servicios Web y por tanto una ves que el servicio Web ejecute las tares necesarias en las capas inferiores de la arquitectura regresara como respuesta la información solicitada al cliente a través de Internet o de misma red.

La propuesta de solución y la arquitectura presentadas en este capítulo con uns descripción general de la forma en como está diseñado el sistema desarrollado en este trabajo, lo más importante que se puede decir de esta propuesta de solución es que su diseño esta basado en distintas áreas de estudio y por tanto comprende una compleja implementación y puesta en marcha de modo que se buscara la manera de implementar todo lo propuesto en la arquitectura utilizando herramientas existentes y abiertas.

# Capítulo 5

# Implementación

Las aportaciones del presente trabajo se diseñaron e implementaron utilizando diferentes herramientas de software. Uno de los componentes principales en la implementación fue la definición de una meta-ontología, la cual esta compuesta de diferentes ontologías. También se diseñó una red de sensores utilizando hardware *open source* para lo cual fue necesario diseñar algunos componentes electrónicos a fin de facilitar la interacción con el usuario.

En conjunto la meta-ontología y l red de sensores permiten una fácil interacción ente los usuarios y el sistema, gracias a su diseño de ubicuidad. En esta sección se explica el diseño de la meta-ontología y de cada una de las ontologías que la componen.

## 5.1. Diseño de la meta-ontología

Como se indica arriba para esta propuesta se diseño una meta-ontología compuesta por tres ontologías, esta sección se explica el diseño de la meta-ontología y de cada una de las ontologías que la componen.

### 5.1.1. Ontología de dispositivos

En esta ontología se definieron todos los dispositivos utilizados para éste trabajo, por lo que al definir las clases que componen a ésta ontología (*véase figura* 5.1) podemos observar una relación entre las clases que la componen.

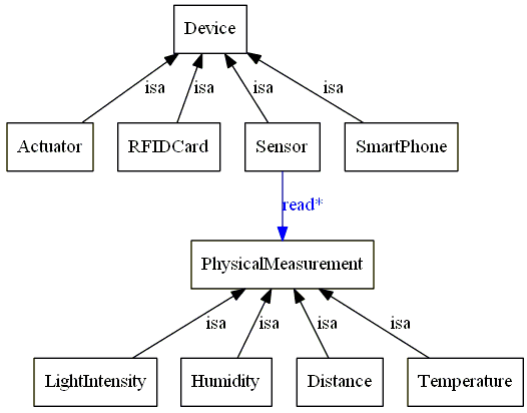


Figura 5.1: Ontología Devices (estructura)

### Clase dispositivo

Las clases que componen a esta clase principal son aquellas que tienen relación con los diferentes dispositivos que se pueden integrar, para nuestro caso particular tenemos los dispositivos:

- **Actuator**: esta clase contiene a todos los dispositivos que juegan el papel de *actuadores*. Los actuadores son dispositivos que sirven de interfaz entre el mundo digital y el mundo físico.
- **RFIDCard**: esta clase contiene a todas las tarjetas RFID utilizadas en este trabajo, ya que es uno de los medios para reconocer a los usuarios, se asigna una tarjeta a cada uno. Esta tarjeta posee un número de identificación único e información acerca del usuario.
- **Sensor**: en esta clase se encuentran todos los distintos tipos de sensores, los cuales deben leer al menos una magnitud física, también se incluyen dispositivos más complejos los que abren la posibilidad obtener información más compleja del ambiente.
- **Smartphone**: aquí clasifican los dispositivos conocidos como teléfonos inteligentes los cuales poseen un úmero de identificación único conocido como *International Mobile Station Equipment Identity IMEI*. Este número tiene de longitud quince dígitos y puede ser utilizado para identificación del usuario, y de esta forma poder proveer información relevante al usuario.

#### Clase de medidas físicas

Las clases que componen a esta clase principal y hermana de la clase *Dispositivos* hacen referencia a las distintas magnitudes físicas que podemos obtener en los diferentes sistemas me medición (o medida).

- **Intensidad luminosa**: en esta clase se pueden agregar las unidades de medida de la intensidad luminosa para así poder asociarla a un sensor en un sistema de medición con otras magnitudes.
- **Humedad**: en esta clase se hace uso del símbolo de porcentaje ( %) para hacer referencia a la cantidad de humedad censada del ambiente, la cual es obtenida a través de los sensores de humedad.
- **Distancia**: en esta clase se encuentran las distintas unidades de distancia ya que se consideran diferentes unidades de distancia, (centímetros y pulgadas).
- **Temperatura**: aquí van las unidades que tienen que ver con la temperatura y sus distintas unidades de medida para esta propuesta solo se consideran los grados Celsius (°C).

### 5.1.2. Ontología espacio físico

Esta ontología fue diseñada con base en los espacios físicos que conforman la estructura física de la institución, por tanto esta ontología esta compuesta por tres subclases que representan el modelo general de los espacios físcos que se pueden encontrar en la institucion (*véase figura* 5.2), por lo que es importante tomar en cuenta las características de los espacios físicos que se desean incluir. En este caso cabe destacar que hay propiedades que hacen referencia a espacios que están dentro de otros espacios más grandes por consiguiente se puede decir que hay espacios compuestos de otros más pequeños. A continuación abordamos las clases que componen a esta ontología.

#### Clase espacio interno

En esta clase clasifican todos los espacios físicos que están delimitados por una estructura, la cual los protege del medio ambiente, además cuentan con una serie de objetos que los complementan como son: pizarrón, ventanas, mesas, pupitres, etc. Enseguida se enlistan las subclases de esta clase y las propiedades de que debe cumplir un individuo para clasificar en alguna de éstas clases.

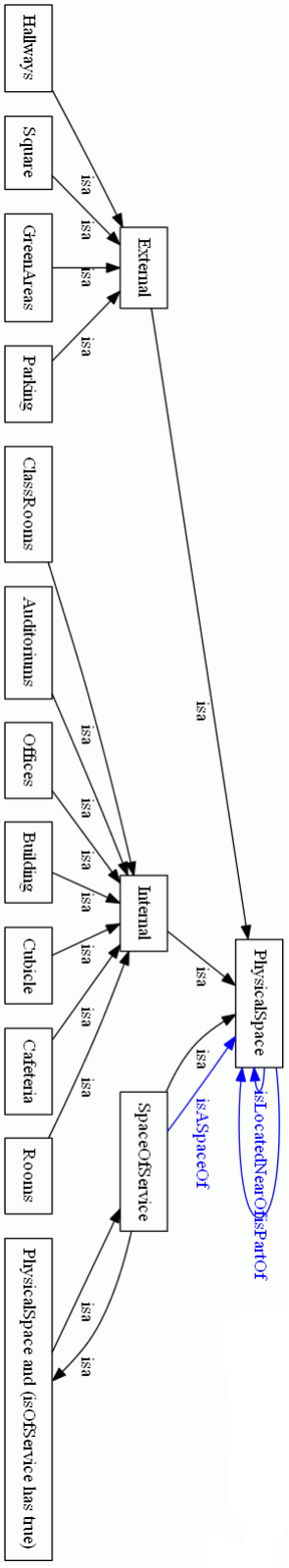


Figura 5.2: Ontología PhysicalSpace (estructura)

- **Auditorios:** aquí se encuentran los espacios que clasifican como auditorios, por ejemplo, en algunas instituciones podemos encontrar múltiples auditorios por lo que se tomó la decisión de incluir una clase para este tipo de espacios.

- **Edificio:** dentro de esta clase se encuentran los individuos que son contemplados como edificios ya que éstos están compuestos por otros espacios mas pequeños que los conforman y además cuentan con una estructura bien definida, que los protege del medio ambiente por lo que también son internos.

- **Cafetería:** las cafeterías son espacios de servicio, pero en algunas ocasiones pueden ser utilizadas para otros tipos de eventos por lo que pueden cambiar su función, por ejemplo, un evento en el que la superficie del espacio es la ideal para contener a una cantidad de personas al remover los muebles que tiene en su interior, para así dar lugar a un área de uso mayor. También hay instituciones que cuentan con más de una cafetería por lo que se decidió crear una clase para este tipo de espacios.

- **Salon de clases:** en esta categoría se encuentran todos los espacios que tienen que ver con un salón de clases, ya que dichos espacios solo tienen ese propósito y por lo tanto su arquitectura no cambia. Además este tipo de espacios proliferan en las instituciones educativas.

- **Cubiculo:** aquí podemos encontrar a todos los espacios designados como cubículos, estos espacios son pequeños y pueden existir múltiples de estos espacios en una sola área, piso o planta. Regularmente se organizan con un número para identificarlos o una combinación alfanumérica en los casos de que exista una gran cantidad de estos espacios.

- **Oficinas:** esta clase contiene a todas las oficinas, las cuales al ser un espacio físico designado para esta finalidad, permite que exista una cantidad variable de oficinas en uno o varios espacios internos de un edificio.

- **Salas:** se refiere a los espacios físicos como son salas, salas de espera, etc. Las cuales pueden contener una variedad de mobiliario dependiendo de la finalidad de la sala, por lo que en algunos casos pueden existir salas equipadas con una combinación de mobiliario de un salón de cla-

se y un laboratorio. Dichas salas, dependiendo de la organización de la institución, pueden variar su uso, pero al final siguen en esta clase.

**Clase espacio externo**

Las subclases que son parte de esta clase comprenden los espacios físicos que se encuentran en el exterior, ya que dichos espacios en algunas ocasiones están bien delimitados pero no cuentan con una estructura que los proteja del medio ambiente, cabe destacar que en esta clase sólo se toman en cuenta a los espacios abiertos y puesto que algunos pueden pertenecer a ambas clases, hay que tomar en cuenta que pueden llegar a existir estos casos especiales.

- **Areas verdes:** se tomaron las áreas verdes como un espacio físico externo ya que por lo regular están a la intemperie y cuentan con una superficie considerable, además en algunos lugares abarcan la mayor parte del terreno y deben ser identificados.
- **Pasillos:** los pasillos fueron tomados en cuenta, aunque son parte de la estructura de los espacios cerrados o internos, además, en algunas ocasiones, los pasillos conectan a los edificios ya sea a nivel de suelo o por niveles superiores de sin embargo son necesarios dar seguimiento a un individuo.
- **Estacionamientos:** el tipo de estacionamientos que aquí se contempla son los del tipo que se encuentran a la intemperie. Cabe destacar que sólo con unas modificaciones a la ontología se podrían agregar estacionamientos cerrados o subterráneos.
- **Plazas o explanadas:** en esta clase se contemplan las explanadas que se encuentran en algunos lugares, además éstas pueden ser muy grandes o muy pequeñas, y aquí son consideradas ya que pueden ser puntos de reunión para grupos de personas en distintas situaciones.

**Clase Espacio de servicio**

Esta clase fue creada con la finalidad de clasificar a los espacios que son designados como de servicio, ya que dichos espacios pueden tener múltiples usos en distintas actividades, dependiendo de la organización de la institución o el objetivo de la actividad. Por esto es importante identificar los espacios que pueden ser utilizados para estos fines.

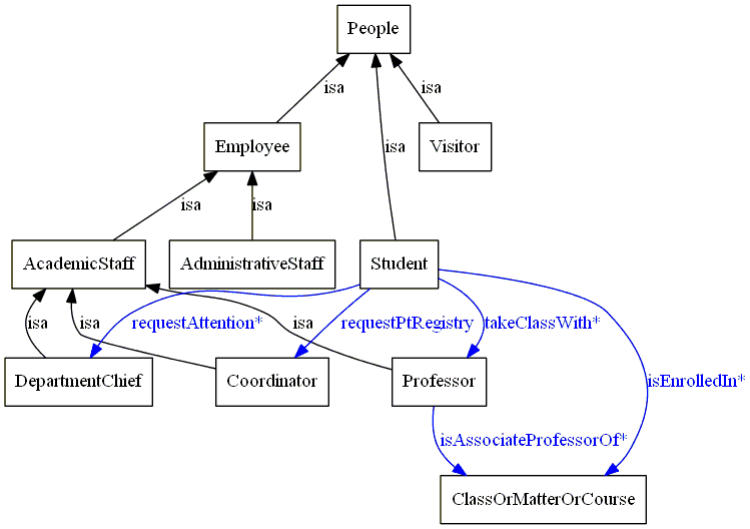


Figura 5.3: Ontología People (estructura)

### 5.1.3. Ontología de personas

Esta ontología fue creada con el objetivo de clasificar a las personas que se encuentran en una institución con base en el rol que desempeñan (*véase figura* 5.3) esto es para facilitar su identificación. Además se agregó una clase que define los cursos que se puede impartir o tomar en una institución académica, por lo que esta clase está más enfocada en una institución académica y los roles de las personas.

#### Clase persona

Aquí podemos apreciar una serie de subclases, que en algunos casos se anidan, incluso las subclases tienen subclases. Gracias a este diseño es posible crear clasificaciones con propiedades compartidas e incluso más específicas para nuestro objetivo.

#### Clase Empleado

En esta clase se encuentran todas las personas que clasifican como empleado, los empleados en algunas instituciones académicas son de dos tipos: académicos y administrativos. Como nosotros estamos trabajando en un espacio académico sólo nos vamos a enfocar en estos dos tipos de personas,

además la ontología comprende una serie de características las cuales pueden aplicar a otro lugares con unas pocas modificaciones.

- **Personal académico:** dentro de ésta clase encontraremos tres subclases, las cuales comprenden los roles principales de una institución académica.
  - **Coordinador:** los coordinadores pertenecen al personal académico y se encargan de llevar a cabo las revisiones de los planes de estudio en la institución.
  - **Jefe de departamento:** los individuos que entran en esta categoría realizan las tareas de coordinación de los profesores.
  - **Profesor:** los profesores clasifican aquí por su actividad de enseñanza, en algunas ocasiones también se dedican a la investigación además de las actividades de enseñanza.
- **Personal administrativo:** las personas que aquí encontraremos son aquellas que realizan las tareas de validación en los procesos administrativos y académicos.

### Clase Estudiante

Las personas que encontramos aquí son aquellas que están inscritas en un programa académico, ya sea de tiempo completo o medio tiempo. Recordemos que regularmente a los alumnos se les asigna una matrícula o número de identificación, esto es para conocer su estatus en la institución y para su historial en dicha institución.

### Clase Visitante

Aquí podemos encontrar a las personas que clasifican como visitantes en alguna institución, cabe destacar que se tomaron en cuenta ya que en algunos lugares se le asigna una tarjeta de identificación temporal y nosotros buscamos identificar el comportamiento del individuo poseedor de dicha tarjeta de identificación.

### 5.1.4. Meta-Ontología

La meta-ontología es una ontología que está compuesta de las ontologías anteriormente presentadas y además cuenta con todas sus características. A continuación abordaremos a la meto-ontología, para una explicación completa *véase figura 5.4*.

Esta ontología, cuenta con todas las características presentadas anteriormente agregando algunas propiedades adicionales, las cuales permiten crear relaciones entre los individuos de diferentes ontologías. Gracias a este diseño se tienen ontologías modulares, lo que nos permite relacionar por ejemplo, a la clase espacio físico con la clase dispositivos, para así poder asociar las lecturas que se toman por medio de los sensores definidos en la clase dispositivo a un espacio físico. Gracias a ésta posibilidad podemos dar un seguimiento a los cambios en el ambiente de un lugar e incluso dar seguimiento a los cambios que tienen distintos lugares por medio de la red de sensores.

Además, gracias a las redes de sensores que pueden ser desplegadas en distintos lugares y a la semántica que nos brinda la ontología, es posible detectar la cantidad de personas que encuentran en un espacio físico. Incluso existe la posibilidad de conocer la identidad de las personas que ahí se encuentran, ya que es deseable identificar a las personas y sus perfiles, los cuales han sido cargados previamente en la ontología *Persona*. Es posible crear servicios personalizados que utilicen la información proveniente de la meta ontología alimentada por la red de sensores, y así crear un sistema más complejo con un nuevo nivel de semántica.

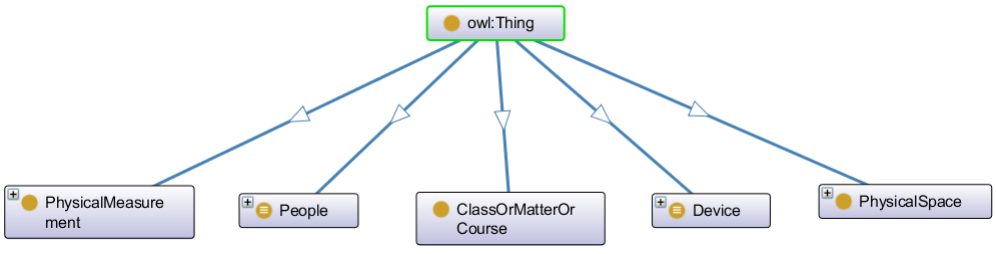


Figura 5.4: Meta-ontología Institution (estructura)

La meta-ontología tiene como objetivo el procesar de manera fácil y directa la datos que se obtienen por medio de la red de sensores para convertirlos en información. Dicha información se puede ser utilizada de forma directa, puesto que ya ha sido procesada por el hardware reconfigurable antes de llegar a la ontología, por lo que es más fácil entender dicha información y darle un uso práctico.

Como de indico anteriormente la meta-ontología conjuga a las otras ontologías, es decir, al ser éstas modulares se tiene la capacidad de agregar nuevas funcionalidades como son, la posibilidad de dar seguimiento a los usuarios con base en su ubicación, también es posible detectar a los dispositivos que se encuentran en la red y en algunos casos es posible consultar su estado; y además, en algunos casos, es posible controlarlos de remotamente.

## 5.2. Red de sensores con arduino

La red de sensores se diseñó haciendo uso del hardware *open source* de *arduino*, el cual permite crear distintos prototipos de forma rápida y sencilla sin la necesidad de comenzar desde cero un prototipo electrónico. Puesto que es deseable el poder cambiar la ubicación física del sensor, e incluso es posible el agregar nuevas funcionalidades a la red de sensores, se optó por utilizar éste tipo de tecnología. Para este trabajo se utilizaron distintos sensores los cuales abordamos a continuación.

Nuestra red de sensores está compuesta por dos arduinos modelo MEGA2560[1] los cuales están equipados con distintos sensores, además cuentan con interfaces adicionales de red WiFi (Shield WiFi[2] última vez consultado 9-9-16) para permitir la conexión a través de la red inalámbrica para así evitar el uso de cables. Cabe destacar que también se pueden utilizar los aditamentos para redes cableadas *(Ethernet)* para los casos en que el alcance de la red WiFi no sea suficiente.

Se eligió *Arduino Mega 2560* por sus capacidad de memoria de progra-

[1] https://www.arduino.cc/en/Main/ArduinoBoardMega2560 última vez consultado 9-9-16

[2] https://www.arduino.cc/en/Guide/ArduinoWiFiShield101 última vez consultado 9-9-16

ma y la cantidad de pines de entrada/salida, lo que se ve reflejado en la capacidad de manejar distintos componentes que pueden ser conectados y controlados por el arduino, las especificaciones técnicas de esta version en especial son las siguientes:

- Cuenta con un micro-controlador basado en el ATmega2560[1] de la compañia Atmel
- Tiene 54 pines digitales de entrada/salida de los cuales 15 cuentan con salida del tipo PWM[44] (pulse-width modulation) [2]
- De los 54 pines de entrada/salida 16 son entradas analógicas.
- Cuenta con cuatro puertos de comunicación del tipo serial y un cristal oscilador a 16 MHz (Mega Hertz)

Cabe destacar que cuenta con una fuente de alimentación de corriente directa desde los 7 volts hasta 20 como suministro de energía, puede alimentar a otros dispositivos por medio de los pies de estrada y salida a 5 volts y hasta 20mA (mili Aperes).

Además el lenguaje de programación que se utiliza para desarrollar programas en este tipo de plataforma es una variación de lenguaje de programación C, C++. Por lo que es de fácil uso y aprendizaje el desarrollar prototipos en este tipo de hardware.

Los sensores que se agregaron son diferentes para cada arduino; puesto que se busca lograr un ambiente distribuido en la misma red de sensores, por esta razón se optó por crear distintos programas para cada uno de los arduinos. Lo único que comparten en común es la opción de ejecutar pequeños servidores ligeros de páginas Web HTTP con los que es posible mostrar los datos obtenidos a través de los sensores montados en cada uno de los arduinos.

### 5.2.1. Sensores utilizados

Los sensores que fueron utilizados para el desarrollo de la plataforma poseen distintas características que se explican a continuación. Cabe resaltar

[1] http://www.atmel.com/devices/atmega2560.aspx última vez consultado 9-9-16

[2] la modulación por ancho de pulso de una señal o fuente de energía que sirve para modificar el ciclo de trabajo de una señal continua.

que los sensores que se eligieron son de gama baja por lo que en algunos casos cuentan con margenes de error considerablemente grandes y puesto que el sistema no requiere de demasiada precision ni ejecuta tareas criticas, estos sensores de gama baja son adecuados, sin mencionar que son bastante accesibles en cuanto a costo se refiere.

El primer tipo de sensor que se utilizo es el DHT11[1], este sensor tiene la capacidad de medir la temperatura y la humedad del ambiente con una precision de $\pm 2$°C de error, y el margen de error respecto a la medición de la humedad es del $\pm 5\,\%$ de error, este sensor es del tipo digital y funciona con una tensión de 5 voltios de entrada y un consumo de corriente de 2.5 mA con una velocidad de muestreo de un segundo, y es capaz de medir temperaturas desde los 0°C hasta los 50°C para temperatura y para la humedad el rango se encuentra entre el 20 % y el 90 %. La familia DHT cuenta con diferentes miembros y cada uno tiene capacidades diferentes en cuanto a precision y rangos de lectura para cada miembro, por lo que es posible utilizar a otros miembros de esta familia sin la necesidad de hacer cambios demasiado significativos en el código.

Otro sensor utilizado en esta propuesta, es el que mide la temperatura, en este caso se uso el LM35[2] este sensor funciona con un rango de tensión desde los 4 hasta 30 voltios, y puede medir temperaturas desde los -55°C hasta los 150°C con un error del $\pm 0.5$°C. Todo lo anterior con un consumo de corriente de $60\mu$, por lo que es la mejor opción si se desea tener una medición de la temperatura mas exacta para aplicaciones críticas en las que la temperatura juega un papel importante. además este sensor es de bajo costo y no requiere hardware adicional puesto, que su interfaz es analógica y su velocidad de muestreo es continua, por tanto el más mínimo cambio en la temperatura se ve reflejado directamente en su pin de salida de datos y no es necesario esperar para procesar los datos obtenidos.

Los dos sensores mencionados anteriormente se usaron para tener distintas fuentes de lectura de la temperatura, ademas se agregó el sensor DHT11 para obtener la lectura de humedad del ambiente, estos tres sensores son de bajo costo y consumo energético.

[1] *http : //www.micropik.com/PDF/dht*11.*pdf* última vez consultado 9-9-16
[2] http://www.ti.com/product/LM35 última vez consultado 9-9-16

También se uso un sensor llamado HC-SR04[1] para hacer mediciones de distancias haciendo uso del ultrasonido, es decir cuenta con un emisor y un receptor de ultrasonido, el uso de este tipo de sensor es para contar la cantidad de individuos (personas) que entran o salen de algún espacio físico, para así auxiliar al sistema en la toma de decisiones. Este sensor cuenta con las siguientes especificaciones técnicas: es capaz de medir distancias desde los 2 centímetros (cm) hasta 4 metros (m), con una precision de hasta 3 milímetros (mm), el consumo energético de este sensor es de 5 volts y una corriente de 15 mA, el ángulo de medición es de 15°.

Para trabajar con este dispositivo se debe enviar una señal en alto (1 lógico [2]) por un lapso de tiempo de $10\mu$S (micro segundos), el objetivo de hacer esto es para que el sensor envíe vario pulsos con una frecuencia de 40 kilo Hertz (kHz). Una vez hecho esto se esta a la escucha del regreso de dichos pulsos, si la señal regresa se lleva a cabo la operación de cálculo de la distancia con la siguiente fórmula *(véase fórmula 5.1)*

$$d = \frac{(t)(v)}{2} \tag{5.1}$$

donde $d$ = distancia censada, $t$ = tiempo de la señal en alto y $v$ = velocidad del sonido (340 m/s).

Para lograr la medición de la cantidad de luz que se encuentra al interior de un lugar cerrado y compararla con respecto a la hallada en el exterior, es necesario utilizar un sensor que pueda "leer" dichos cambios en la intensidad de la luz, por lo que se utilizó un sensor equipado con una LDR[45, 46][3] (light-Dependent Resistor) o mejor conocida como foto resistencia

El sensor cuenta con un selector que permite elegir el comportamiento de la salida de datos, es decir, permite elegir entre una salida de datos analógica, la cual está enviando constantemente los valores de luz censados o una salida digital en la que es posible establecer un límite de luz. Dicho límite lo establece el usuario, por tanto el sensor compara el valor "leído" del ambiente con respecto al establecido por el usuario, entonces el sensor envía un pulso en alto y lo mantiene indicando que el valor censado se encuentra

[1] *http://www.micropik.com/PDF/HCSR04.pdf* última vez consultado 9-9-16

[2] En electrónica se le conoce como señal en alto a un pulso continuo de entre 3.3 y 5 volts

[3] Una foto resistencia es un componente electrónico que cambia su resistencia con respecto a la incidencia de luz que recibe.

fuera del límite establecido por el usuario. La manera en la que compara el valor establecido con respeto al censado es por medio de un amplificador operacional que viene integrado. La figura 5.5 es una representación de un sensor de luz simple que hace uso de una LDR.

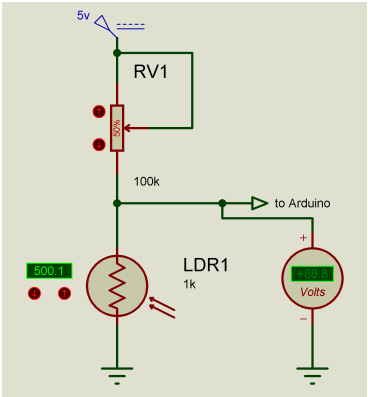


Figura 5.5: Sensor con LDR (esquema)

También se hizo uso de un lector de tarjetas RDIF que permite leer los datos que se encuentran guardados en una tarjeta de RFID (Radio Frequency Identification) y es utilizado para identificar al usuario, ya que este dispositivo permite grabar información en las tarjetas RFID y también leer la información que contienen las tarjetas.

Hay que tomar en cuenta que no todos los dispositivos de RFID son iguales, ya que por sus características y especificaciones técnicas, algunos son capaces de leer la información almacenada en las tarjetas a largas distancias e incluso en algunos casos son capaces de leer distintos tipos de tarjetas, cabe destacar que las tarjetas pueden ser grabadas con una ”llave”, la cual puede ser cambiada por el administrador del sistema, esto brinda un cierto grado de seguridad.

Como evitar la alteración de la información que puede contener la tarjeta y evitar el uso indebido de la información que contiene ya que puede ser información sensible. De igual forma que existen distintos tipos de lectores, también existen tarjetas con distintas velocidades de lectura/escritura, además de la capacidad de almacenamiento y la cantidad de bancos de me-

moria con los que cuentan; por lo que para nuestro caso sólo se utilizó un sólo banco de memoria y una llave genérica puesto que la parte de seguridad no es objetivo de este trabajo, pero puede ser contemplado como un trabajo futuro.

### 5.2.2. Actuadores

Un actuador es un dispositivo eléctrico o electrónico que transforma la energía eléctrica en un movimiento mecánico [47], por ejemplo: una lámpara, un motor, entre otros. Los actuadores están presentes en nuestra vida diaria, ya que se encuentran en casi todo lo que nos rodea, por lo que es deseable controlar dichos dispositivos por medio de las computadoras. En algunos casos los actuadores ya cuentan con una computadora empotrada en el dispositivo mismo, pero solo con el objetivo de controlar el comportamiento del dispositivo, mientras el usuario tiene interacción con el actuador por medio de alguna interfaz o medio de entrada.

Los actuadores pueden utilizar bajas tensiones eléctricas, ya que son más fáciles de controlar que aquellos que utilizan tensión media o alta, por ejemplo una lámpara incandescente, un ventilador e incluso un aire acondicionado. Para poder controlar estos últimos dispositivos es necesaria una etapa intermediaria, con la cual se puede aislar la parte de control (computadora empotrada) de la parte eléctrica, mejor conocida como etapa de potencia y, gracias al aislamiento que provee, permite proteger a los dispositivos electrónicos de las altas tensiones que utiliza el actuador.

#### Etapa de potencia

Una etapa de potencia tiene que ver con las aplicaciones de la electrónica de potencia para el control o conversión de la energía eléctrica [44]. Las técnicas de conversión requieren el encendido y apagado de semiconductores de potencia, donde un dispositivo ideal de potencia no debe tener limitaciones de conexión y desconexión, en términos de tiempo de encendido, apagado y las posibilidades de manejo de corriente y voltaje.

Esto se logra al hacer uso de microprocesadores y circuitos integrados de procesamiento de señales, que están reemplazando a los circuitos integrados y componentes discretos, que generaban las señales requeridas de excitación para los dispositivos.

Para nuestro objetivo de poder controlar algunos dispositivos se desarrolló una etapa de potencia basada en los principios anteriores por lo que se utilizaron algunos componentes de electrónica de potencia y se diseñó para cumplir con el principio de encendido y apagado que se mencionó anteriormente. El funcionamiento de la etapa de potencia *véase figura 5.6* consiste

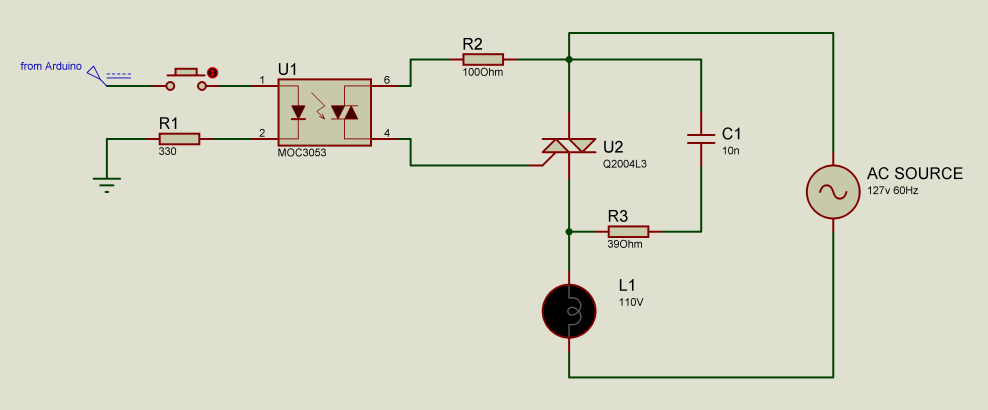


Figura 5.6: Etapa de potencia de estado sólido (esquema)

en recibir una señal en alto un 'uno' lógico enviado desde el arduino, dicha señal activa a un diodo Led (Light Emitier Diode) o diodo emisor de luz, que se encuentra encapsulado y excita a un foto transistor, el cual al recibir la luz emitida por el Led permite el flujo del electrones a través de sus terminales, es decir, se activa el interruptor. Una vez activado permite el flujo de tensión y corriente eléctrica a través del triristor [1] semiconductor (TRIAC), que tiene la capacidad de ser bidireccional, lo que permite el flujo eléctrico en ambas direcciones, por lo que es ideal para aplicaciones de control en actuadores que funcionan con corriente alterna.

Para dar un mejor manejo al activar o desactivar a los actuadores y por los distintos tipos que existen, se agregó una resistencia y un capacitor o condensador entre las terminales del TRIAC, para que sea posible manejar cargas tanto inductivas como resistivas con el objetivo de mejorar la respuesta de la etapa de potencia al activar o desactivar algún actuador. La potencia de carga que puede manejar esta etapa de potencia es de alrededor

[1] componente electrónico a base de elementos semiconductores que posee realimentación interna para producir conmutación

de 1000 Watts, la potencia puede ser calculada haciendo uso de la fórmula 5.2

$$p = V * I \tag{5.2}$$

donde $P$ = potencia en Watts, $V$ = voltaje o tensión $I$ = corriente soportada por el transistor (Amperes). Por ejemplo: los componentes utilizados para el desarrollo de la etapa de potencia soportan mínimo 120 volts y 10 Amperes, al hacer la operación con la fórmula nos da un resultado de 1200 Watts. Hay que tomar en cuenta que esa es la potencia límite y debemos trabajar por debajo de ésta para evitar daños a los actuadores, por lo que se decidió optar por dejar la potencia en 1000 Watts. Además hay que destacar que la red eléctrica no siempre suministra la misma tension por lo que dependiendo de la hora de día y la demanda esta puede variar.

## 5.3. Servicios web

Para el desarrollo de los servicios web se hizo uso del *framework* Apache CXF[1] que permite desarrollar servicios web haciendo uso de distintas APIs. Los servicios desarrollados con ese *framework* pueden hacer uso de una variedad de protocolos como SOAP, XML/HTTP, RestFul HTTP, o CORBA. Además puede trabajar sobre diferentes capas de transporte como HTTP entre otras. El *framework* se utilizó en el IDE Eclipse[2] para trabajar con el lenguaje de programación JAVA[3], ya que permite desarrollar aplicaciones de red y es un estándar para desarrollar aplicaciones móviles con contenido basado en Web entre otras aplicaciones.

También se hizo uso de la API de OWL version 4.1.4 para poder hacer uso de las ontologías desde JAVA ya que esta API permite crear, modificar, cargar en memoria, etcétera. También cuenta con un razonador ya integrado que puede verificar las inconsistencias en las ontolgías, lo que da paso a la posibilidad de cargar a las ontologías desde los archivos con extensión *.owl*, por lo que se puede decir que permite gestionar ontologías desde un ambiente de programación, facilitando el desarrollar programas o aplicaciones que hagan uso y explotación de ontologías.

[1] http://cxf.apache.org/ última vez consultado 3-6-16
[2] https://eclipse.org/ última vez consultado 3-6-16
[3] https://www.java.com/es/about/ última vez consultado 3-6-16

Otra API utilizada para el desarrollo de los servicios Web es SWRL versión 1.0.7, como se ha mencionado en capítulos anteriores. SWRL es un motor de reglas, que además cuenta con un razonador para crear las relaciones que se necesitan entre las ontologías, para así lograr la semántica necesaria para la explotación de los datos obtenidos por medio de la red de sensores para convertirlos en información útil para el usuario final.

Los servicios Web diseñados para este trabajo son cuatro, cada uno de ellos cuenta con un diseño similar con respecto a su estructura, pero se diferencian al momento de ejecutar la petición del cliente, es decir, cada uno recibe diferentes parámetros y ejecuta una consulta con el motor SWRL, cada consulta esta hecha para devolver información específica que se encuentra dentro de la ontología.

Lo primero que se debe declarar en cada uno de los servicios, una vez importadas las ontologías y cargadas en memoria, es el motor de reglas SWRL, ya que este permite cargar las reglas ya contenidas en la meta-ontología e incluso crear nuevas reglas que no existan en la meta-ontología.

El script 5.1 se muestran las líneas de código para declarar al motor de reglas y así poder utilizarlo para hacer inferencias en la ontolgía, además puede importar las reglas contenidas en la meta-ontología.

```
1 SWRLRuleEngine ruleEngine =
2         SWRLAPIFactory.createSWRLRuleEngine(ontology);
3         ruleEngine.getSWRLRules();
4         ruleEngine.infer();
```

Script 5.1: Declaración del motor SWRL

La línea 1 de código *SWRLRuleEngine* permite crear al motor de reglas que en este caso fue nombrado como *ruleEngine*. La siguiente línea, que es la continuación de la primera, indica en donde se va a crear el motor de reglas por lo que en este caso se indica que debe ser en la copia de la meta-ntolgía cargada en memoria llamada *ontology*.

Al ejecutar estas líneas de código se crea el motor de reglas en la copia que se encuentra en memoria. Se crea la copia de la meta-ontología en memoria para mejorar el rendimiento al momento de consultar a la meta-ontolgía y al cargar las reglas e incluso al crear y ejecutar nuevas reglas o consultas, ya que la velocidad de lectura es mayor que hacerlo en directamente en disco.

Una vez creado el motor de reglas se debe crear el motor de consultas, el cual es el encargado de regresar la información adecuada, con base en las consultas que se ejecutan en cada uno de los servicios. Cabe destacar que al igual que el motor de reglas, el motor de consultas es capaz de cargar consultas guardadas previamente en la meta-ontología, además de ser capaz de crear nuevas consultas desde código que pueden ser guardadas directamente en la meta-ontolgía.

```
1 SQWRLQueryEngine queryEngine =
2         SWRLAPIFactory.createSQWRLQueryEngine(ontology);
```

Script 5.2: Declaración del motor de consultas de SWRL

El script 5.2 presenta el comando con el que se debe declarar el motor de consultas, en este caso *SQWRLQueryEngine* es el que nos indica el tipo de motor de va a crear y el motor de consultas a crear se nombra *queryEngine*. De igual manera que el motor de reglas se debe indicar en donde se va a crear por lo que de la misma manera se indica que debe ser en la copia cargada en memoria llamada *ontology*.

Para el caso de los servicios los motores tanto de reglas como de consultas se nombran igual y ambos deben ser declarados ya que en la meta-ontología se encuentran las seis reglas ya mencionadas anteriormente en este capítulo. Por tanto es necesario crear y ejecutar a ambos motores puesto que las reglas crean inferencias entre los individuos y las consultas permiten obtener información incluida la que se crea al ejecutar las inferencias de las reglas.

### 5.3.1. Servicio de localización:

El objetivo de este servicio es el de ubicar en dónde se encuentra el usuario, es decir, sirve para ubicarnos a nosotros mismos dentro de la institución, puede deseable saber en dónde se localiza el usuario y las variables ambientales del lugar. Es deseable que el usuario mismo vaya conociendo los nombres de los lugares en los que se encuentra y cómo moverse dentro de la institución, para ello el usuario debe ingresar su nombre para saber en dónde se localiza.

```
1 @WebResult(name="location")
2         public Location getLocationInfo(@WebParam(name="name"
              )String Name)
```

```
3        throws SQWRLException, SWRLParseException;
```

Script 5.3: Servicio de localización

En el script 5.3 "name" es el parámetro del tipo cadena *(String)* que recibe el servicio, por lo tanto es el nombre del usuario el parámetro que se debe enviar al servicio. Una vez recibido en paŕametro, el servicio ejecuta una consulta a la meta-ontología y busca el perfil del usuario dentro de la meta-ontología, una vez encontrado hace las asociaciones pertinentes con los demás individuos dentro de la meta ontología.

Una vez encontrado el perfil del usuario se puede conocer su ubicación y gracias a esto se pueden conocer los distintos sensores que se encuentran en el espacio físico en que se encuentra el usuario, por consiguiente se pueden obtener las variables ambientales del lugar con base en el nombre del usuario, admás de obtener la ubicación y demás información del lugar en el que se encuentra.

```
1  SQWRLResult Q1 = queryEngine.runSQWRLQuery("Q1", "
       ontologypeople:hasPersonName(?p, ?name) ^ ontologydevices
       :itIsLocatedIn(?a, ?located) ^ ontologyphysicalspace:
       hasName(?login, namephys) ^ ontologydevices:Actuator(?a)
       ^ ontologyinstitution:hasLight(?login, ?lgt) ^
       ontologyphysicalspace:hasNumberPhysicalSpace(?login, ?
       numphys) ^ ontologyinstitution:hasTemp(?login, ?tmp) ^
       ontologyinstitution:hasHum(?login, ?hum) ^
2   ontologyinstitution:hasLoginIn(?p, ?login) ^
       ontologyphysicalspace:hasLetter(?login, ?letter) ^
       ontologyphysicalspace:hasLevel(?login, ?lvl) ^
       ontologypeople:People(?p) ->
3   sqwrl:select(?name, ?namephys, ?letter, ?lvl, ?numphys) ^
       sqwrl:avg(?tmp)^sqwrl:avg(?hum) ^ sqwrl:avg(?lgt)");
```

Script 5.4: Consulta para localización

El script 5.4 es la consulta elaborada en *Semantic Query-Enhanced Web Rule Language* [1] que es un lenguaje de consultas basado en SWRL, lo cual permite extraer información de las ontolgías. Al momento de ejecutar una o varias consultas, cada consulta debe hacer referencia a las clases y por tanto a las propiedades que se desea obtener por lo que en el código del script 5.4 se específica, lo siguiente:

[1] https://github.com/protegeproject/swrlapi/wiki/SQWRL última vez consultado 20-7-16

- Para *ontologypeople:hasPersonName(?p,?name)* el comando *ontologypeople* hace referencia a la ontología people que es parte de la meta ontología.
- *hasPersonName* es una *data property* (propiedad de dato) que existe dentro de la ontología *people*.
- *(?p, ?name)* **?p** quiere decir que se va a ejecutar una búsqueda en todos los individuos persona y los que cumplan con todas las características (propiedades) de una persona en la ontolgía serán seleccionados.
- *(hasPersonaName)* y es la propiedad que debe cumplir el individuo en la ontología para ser considerado y **?name** es la variable en la que se almacenan los individuos obtenidos que cumplan con dicha propiedad.

El carácter ^ que se utiliza como conector para delimitar a los resultados conforme van cumpliendo las condiciones, para así reducir la cantidad de resultados a sólo algunos más exactos, facilitando así la búsqueda y reduciendo el tiempo de ejecución en la consulta. Al agilizar la obtención de información por medio de los servicios Web al momento de ejecutar la petición del cliente.

### 5.3.2. Servicio de búsqueda:

El servicio de búsqueda es muy similar al servicio anterior, pero, a diferencia de su antecesor, este servicio utiliza una consulta más sencilla, es decir, se necesitan menos condiciones para obtener la información solicitada por el usuario al momento de ejecutar la consulta.

El servicio tiene como objetivo dar respuesta a la petición de un cliente que desea conocer la ubicación de alguna persona dentro de la institución. Hay que tomar en cuenta que sólo es la ubicación dentro de la institución; el servicio no toma en cuenta las variables ambientales ni ninguna otra información acerca del lugar en donde se encuentra la persona a localizar por lo que solo cuenta con informaciín sencilla.

El servicio debe recibir como parámetro de búsqueda el nombre del usuario al que se desea ubicar este parámetro es del tipo cadena, al recibir el nombre del usuario, el servicio crea los motores de reglas y de consultas. Una vez hecho lo anterior se procede a ejecutar la consulta en el motor de consultas.

```
1  @WebResult(name="search")
2          public Search getSearchInfo(@WebParam(name="nom")
               String Nombre) throws SQWRLException,
               SWRLParseException;
```

Script 5.5: Servicio de búsqueda

El script 5.5 muestra la forma en la que fue declarado el servicio, en la línea 1 se declara el nombre del servicio que para este caso tiene el nombre de *"search"*. En la línea 2 se declara que es público como ya se ha mencionado antes y el nombre de la variable del tipo cadena que es el parámetro que recibe el servicio, para este servicio el nombre de la variable es *"nom"*. Como resultado este servicio regresa sólo la ubicación del usuario que se desea localizar, es decir devuelve el nombre del lugar, el nivel, el número que tiene asignado el espacio físico[1] y la letra que identifica al edificio que contiene al lugar en el que se encuentra al usuario.

```
1  SQWRLResult Q2 = queryEngine.runSQWRLQuery("Q2", "
       ontologypeople:People(?p)␣^␣ ontologypeople:hasPersonName
```

[1]El número de un espacio físico se refiere a la numeración asignada a las divisiones que tiene un nivel

```
(?p, ?name) ^ ontologyinstitution:hasLoginIn(?p, ?login)
^ ontologyphysicalspace:hasLetter(?login, ?letter) ^
ontologyphysicalspace:hasNumberPhysicalSpace(?login, ?phy
) ^ ontologyphysicalspace:hasLevel(?login, ?lvl) ^
ontologyphysicalspace:hasName(?login, ?namephys) -> sqwrl
:select(?name, ?namephys, ?letter, ?lvl, ?phy)");
```

Script 5.6: Consulta de búsqueda

El script 5.6 presenta la consulta desarrollada para la búsqueda de un usuario dentro de la institución académica esta consulta hace uso de las ontolgías *"People"* y *"Physicalspace"*. Para esta consulta en el último comando de la consulta se encuentra **sqwrl:select(?name, ?namephys, ?letter, ?lvl, ?phy)");** en que, *sqwrl:select* tiene como función seleccionar las distintas variables que hayan sido utilizadas para manejar la información obtenida de la consulta, para este caso se seleccionan *?name, ?namephys, ?letter, ?lvl y ?phy* donde: *?name* es el nombre del usuario que se desea localizar, *?namephys* es el nombre del espacio físico (lugar) en caso de tener uno por ejemplo: "Sala audiovisual", *?lvl* es en nivel en el que se encuentra el espacio físico, donde '0' es planta baja, 1 = primer piso, etcétera; *?letter* es la letra del edificio (en caso de que el lugar sea parte de algún edificio) en el que se encuentra el espacio físico en el que se ubica el usuario que se desea localizar y *?phy* que hace referencia a el número que tiene asignado el espacio físico dentro de la institución, es decir, puede tener un número asignado dependiendo de su nivel y la cantidad de espacios en los que esté dividido el lugar.

### 5.3.3. Servicio de control Manual-Automático:

El servicio de control manual-automático tiene como objetivo principal obtener información acerca de un espacio físico, pero los datos que regresa como respuesta el servicio tienen un doble propósito el cual se explica más adelante.

```
1 @WebResult(name="actuator")
2         public Actuator getActuatorInfo(@WebParam(name="lugar
              ")String Lugar)throws SQWRLException,
              SWRLParseException;
```

Script 5.7: Servicio de manual-automático

Primero se debe declarar el servicio de la misma manera que sus antecesores por lo que en el script 5.7 muestra como se crea el servicio, en la línea

1 se declara el nombre con el que se va a identificar al servicio en cuestión, para este caso el nombre elegido hace referencia a la infromacion que se puede obtener con dicho servicio, el nombre del servicio es *"actuator"*. En la línea 2 se declara el nombre del parámetro a obtener y el tipo por lo que para este servicio el nombre del parámetro es "lugar" y es del tipo cadena.

El servicio debe recibir el nombre del lugar, el nombre del lugar puede contener espacios y debe ser igual al que se encuentra en el lugar, es decir, si el lugar se llama "b003" el cliente debe enviar exactamente el nombre tal cual está escrito.

Puesto que se crearon los individuos y sus nombres en la ontolía tal cual están escritos, si se ingresara una letra de manera errónea el nombre producirá una inconsistencia al momento de ejecutar la consulta.

Las dos aplicaciones que tiene este servicios son: que el usuario pueda controlar de forma manual a los dispositivos en un lugar dado, es decir, el usuario puede activar o desactivar a cualquier actuador que se encuentre en el lugar que se ingrese al servicio, y debe contener al menos un dispositivo actuador que pueda ser controlado.

La segunda opción de aplicación del servicio es que se consume con un programa demonio que está ejecutando en segundo plano y ejecuta la consulta directamente en ontología y no consume al servicio, lo que disminuye el tiempo de respuesta del programa demonio para ejecutar las tareas solicitadas, el programa demonio se explica más adelante.

```
1  SQWRLResult Q3 = queryEngine.runSQWRLQuery("Q3", "
       ontologydevices:Actuator(?a) ^ ontologydevices:
       itIsLocatedIn(?a, ?loc) ^ ontologyphysicalspace:
       hasNumberOfPeople(?loc, ?numpeople) ^ ontologydevices:
       hasIPAddress(?a, ?ip) ^ "ontologydevices:hasState(?a, ?
       state) ^ ontologyphysicalspace:hasName(?loc, ?name) ^
       ontologydevices:hasNameDevice(?a, ?namedev) ^
       ontologyinstitution:hasTemp(?loc, ?temp) ^
       ontologyinstitution:hasHum(?loc, ?hum) ^
       ontologyinstitution:hasLight(?loc, ?lux) -> sqwrl:select
       (?numpeople, ?ip, ?state, ?name, ?namedev) ^ sqwrl:avg(?
       temp) ^ sqwrl:avg(?hum) ^ sqwrl:avg(?lux)");
```

Script 5.8: Servicio de manual-automático

El script 5.8 representa la consulta generada para obtener información de los actuadores disponibles en algún espacio físico, esta consulta hace uso de la ontologías "Dispositivos y Espacio físico" que a su vez son importadas por la meta-ontología *Institución*.

Para esta consulta las propiedades que son de utilizad son las siguientes;

- La dirección IP de cada dispositivo que se puede controlar ya que puede haber distintos arduinos que controlen uno o varios actuadores.
- El estado en el que se encuentra el dispositivo para conocer si esta activo o inactivo.
- El nombre del dispositivo ya que cada dispositivo debe estar registrado en la ontolgía y una de sus propiedades es un nombre que lo identifique.
- El número de personas que se encuentran dentro de un lugar ya que este dato es utilizado por el programa demonio.
- los promedios de las lecturas de los distintos sensores que se encuentran en el espacio físico.

Cabe recordar que pueden existir distintos tipos de sensores que censen la misma magnitud (por ejemplo, la temperatura), por lo que se obtienen los promedios de temperatura, humedad y cantidad de luminosidad que hay disponible en el lugar.

Los promedios se obtienen al ejecutar los comandos *sqwrl:avg(?temp) ŝqwrl:avg(?hum) ŝqwrl:avg(?lux)* y el comando que se encarga de calcular el promedio directamente al ejecutar la consulta es *sqwrl:avg()*, así como este comando permite calcular el promedio.
Existen muchos otros comandos ya incluidos en la API de SWRL que permiten ejecutar de forma rápida otro tipo de funciones que simplifican el trabajo al momento de obtener información de la ontolgía, facilitando así la explotación de ontologías en aplicaciones variadas.

### 5.3.4. Servicio de ingreso de preferencias:

El último servicio que se implemento es el de ingreso de las preferencias del usuario al sistema ya que las preferencias pueden cambiar y no todos los usuarios están registrados por lo que este servicio cumple con el propósito de obtener las preferencias de usuario, además de permitir al usuario elegir el modo de operación del programa demonio, esto es, que el programa puede funcionar de modo automático o manual.

```
1  @WebResult(name="userpreferences")
2  public Selector getUserInfo(@WebParam(name="userInfo")
       UserInfo userInfo) throws SQWRLException,
       SWRLParseException;
```

Script 5.9: Servicio de preferencias

En el script 5.9 muestra la declaración del servicio cuyo nombre para identificarlo es *"userpreferences"*, para poder consumirlo y publicarlo. Como se muestra en la línea 1, esta línea de código es igual que en los servicios anteriores lo único que cambia es el nombre de cada uno de los servicios. En la línea 2 se declaran las variables que el servicio debe recibir como parámetros por lo que para satisfacer el objetivo de éste servicio y complementar la información se recibe un objeto llamado *"sel"* del tipo *"getUserInfo"* cuyo contenido se muestra en el script 5.10.

Este objeto presenta las declaraciones del tipo de dato que va a recibir, nombre de cada dato y tipo, por lo que es recomendable y más factible utilizar los objetos de JAVA ya que facilitan el manejo de la información y permiten al cliente enviar de forma rápida las peticiones al servicio, con la ventaja de poder enviar distintos tipos de datos a través de un objeto declarado con distintos tipos de elementos que lo componen.

```
1  @XmlRootElement(name = "userInfo")
2  @XmlAccessorType(XmlAccessType.FIELD)
3  public class UserInfo {
4          @XmlElement(name = "sel", required = true, nillable =
               false)
5          private int sel;
6          @XmlElement(name = "temp", required = true, nillable
               = false)
7          private int temp;
8          @XmlElement(name = "hum", required = true, nillable =
               false)
9          private int hum;
10         @XmlElement(name = "lux", required = true, nillable =
               false)
11         private int lux;
12         @XmlElement(name = "place", required = true, nillable
               = false)
13         private String place;
```

Script 5.10: Obejto getUserInfo

Como se puede observar el objeto *getUserInfo* contiene cuatro elementos que son indispensables para el consumo de este servicio, cada elemento es del tipo *XmlElement* que posee características que pueden ser definidas por el usuario al momento de crear el objeto, por ejemplo:
En la línea 1 se declara el elemento raíz cuyo nombre para este caso es *userInfo* por consiguiente de sebe declarar el tipo de acceso que se va a tener el objeto raíz, en la línea 2 se define el tipo de acceso por lo que se utiliza el tipo campo *FIELD*.

Una vez hecho lo anterior se procede a declarar al objeto y su contenido, en la línea 4 se denota el primer elemento llamado "sel" que es del tipo *XmlElement*, para el caso del servicio se declara que todos los parámetros deben ser necesarios y ademas no deben tener contenido nulo ya que éso causaría inconsistencias al tratar de consumir el servicio.

Cada uno de los elementos contenidos en este objeto tienen un tipo asignado que puede ser utilizado para identificar los datos enviados por el cliente al hacer la petición, los tiposde datos son: de tipo entero para los elementos *sel, temp, hum, lux* y del tipo cadena para el elemento *place* ya que cada uno de estos datos está asociado a una variable que se encuentra en la meta-ontología y son obtenidas con base en el valor promedio leído por todos los sensores que se encuentran en un lugar.

Para el caso de este servicio se asocian esas variables a las preferencias del usuario por lo que los datos que envía el usuario al servicio por medio del cliente se utilizan como datos que el usuario puede modificar sin la necesidad de inyectar información a la meta-ontolgía, facilitando así el manejo del sistema al permitir la interacción indirecta con el programa demonio.

El servicio devuelve como respuesta un objeto con los datos que el cliente ha enviado y una confirmación de la información que ha recibido. También crea un archivo de texto en el que se guardan las preferencias del usuario. Dicho archivo es utilizado por el programa demonio para llevar a cabo las acciones necesarias, dependiendo de las preferencias del usuario y el contenido del archivo, por lo que es importante mencionar que el archivo siempre es creado en caso de que no exista y cada que el cliente hace una petición se modifica con los nuevos valores enviados por el usuario como sus nuevas preferencias.

### 5.3.5. Programa demonio:

El propósito del desarrollo de este programa es el mantener una interacción con el usuario y su ambiente de manera que no se demande mucha atención del usuario, de esta forma el sistema pueda ejecutar acciones en los dispositivos actuadores sin que el usuario se distraiga. Esto permite tener un control del ambiente que no causa distracciones e incluso permitiendo controlar desde Internet a los dispositivos.

El programa se desarrolló utilizando el lenguaje de programación JAVA. Debe poder consultar la meta-ontolgía y también hacer uso del archivo de texto creado por el servicio web de ingreso de preferencias, ya que al consultar la meta-ontolgía para obtener información debe poder asociarla a las preferencias del usuario y compararlas para llevar a cabo las acciones pertinentes.

```
1  SQWRLResult Q1 = queryEngine.runSQWRLQuery("Q3", "
       ontologydevices:Actuator(?a) ^ ontologydevices:
       itIsLocatedIn(?a, ?loc) ^ ontologyphysicalspace:
       hasNumberOfPeople(?loc, ?numpeople) ^ ontologydevices:
       hasIPAddress(?a, ?ip) ^ ontologydevices:hasState(?a, ?
       state) ^ ontologyphysicalspace:hasName(?loc, ?name) ^
       ontologydevices:hasNameDevice(?a, ?namedev) ^
       ontologyinstitution:hasTemp(?loc, ?temp) ^
       ontologyinstitution:hasHum(?loc, ?hum) ^
       ontologyinstitution:hasLight(?loc, ?lux) -> sqwrl:select
       (?numpeople, ?ip, ?state, ?name, ?namedev) ^ sqwrl:avg(?
       temp) ^ sqwrl:avg(?hum) ^ sqwrl:avg(?lux)");
```

Script 5.11: Consulta programa demonio

En el script 5.11 se puede apreciar la consulta que se hizo para obtener la información necesaria de la meta-ontologia, la información que devuelve esta consulta tiene que ver no solo con las variables ambientales como temperatura, humedad, etcétera. También con la información adicional asociada a un lugar en especifico, es decir, un espacio físico tiene asociados a otros dispositivos o personas de la meta-ontología y por consiguiente a otros datos de dispositivos o personas que tienen relevancia al momento de ejecutar al programa demonio.

Uno de esos datos es el número de personas que se encuentran en un espacio físico, los datos que devuelve esta consulta son presentados después del símbolo "->". Para esta consulta se utilizan los datos de ńumero de per-

sonas *"?numpeople"*, el estado de los dispositivos que se encuentran en un lugar *"?state"*, la dirección IP de cada uno de los dispositivos actuadores *"?ip"*, el nombre de cada uno de los dispositivos *"?namedev"* y el nombre del lugar en el que se encuentran cada uno de los dispositivos *"?name"*, hay que recordar que cada actuador modelado en la meta-ontología se encuentra asociado a un espacio físico, por lo que es necesario conocer el lugar al que están asociados, el programa demonio hace uso del nombre del espacio físico para buscar en la meta-ontologia y obtener a los dispositivos que puede controlar.

Además se hace uso de las funciones ya definidas directamente en la API de SWRL para obtener el promedio de las variables ambientales, hay que recordar que en un lugar pueden existir distintos sensores, y se desea obtener el promedio de las lecturas de todos los sensores que se encuentran asociados al espacio físico. Por lo que la función *sqwrl:avg()* permite obtener los promedios de propiedades de dato ya definidas y asociadas a un sensor en la meta-ontología.

El funcionamiento del programa demonio se explica a continuación.

- Como primer paso el programa abre el archivo generado por el servicio Web llamado ingreso de preferencias.
- La línea 2 invoca a los métodos ya establecidos en JAVA para manejar archivos.

Se indica el nombre de la variable que para este caso es *'f'* y la ubicación del archivo dentro de la computadora, por lo que se debe ingresar la ruta hacia el directorio que contiene a dicho archivo, el nombre del archivo es *User.txt* y contiene las preferencias que el usuario ingreso por medio del servicio Web.

El algoritmo 1 es una representación de las tareas que lleva a cabo el programa demonio, dependiendo del modo de operación se ejecutan diferentes acciones. La variable *sel* es la que se encarga del modo de operación, ésta es solicitada al usuario al momento de enviar su petición al servicio, es decir, el usuario debe ingresar sus preferencias de acuerdo a lo que se le solicite en el campo correspondiente.

El modo de operación en general es el siguiente: primero el programa demonio verifica el modo de operación utilizando las condiciones en los ciclos *while*, es decir, el primer ciclo *while* verifica al momento de la primer

**Algorithm 1** Algoritmo del programa demonio

```
1: while personas>0||sel == 1 do
2:    cargar al archivo de preferencias FileReader f = new FileReader("c:/Ontologias/User.txt");
3:    Consulta a la meta-ontología
4:    if var >varuser && estado == 0 then
5:       enciende al actuador
6:    end if
7:    if var <varuser && estado == 1 then
8:       Apaga al actuador
9:    end if
10: end while
11: while personas == 0 && sel == 0 do
12:    cargar al archivo de preferencias FileReader f = new FileReader("c:/Ontologias/User.txt");
13:    consulta a la meta-ontología
14: end while
```

ejecución del programa demonio la cantidad de personas dentro del espacio físico seleccionado por el usuario a través del servicio Web.

Una vez hecho esto se procede a comparar el modo de operación (la variable *sel*) y el número de personas que se encuentran dentro del espacio físico, el segundo *while* entra en operación si el nuéro de personas es igual a cero o que cambie el modo de operación a automático. Sí se cumple cualquiera de las dos condiciones: ya sea que el número des personas sea mayor a cero o que el modo de operación sea igual a "1" (automático), se cumple se procede a ejecutar la consulta de la figura 5.11 ya que se necesita toda ésa información para llevar a cabo las acciones necesarias para mantener las preferencias del usuario.

Una vez ejecutada la consulta y obtenida la información se procede a comparar cada una de las variables ambientales con respecto a las preferencias del usuario para llevar a cabo las acciones necesarias enviando los comandos necesarios a los controladores de los actuadores, por lo que se utilizan dos condiciones *if*.

La primer condición *if* compara la variable ambiental obtenida de la ontología con respecto a la preferida por el usuario y verifica que el estado

del actuador se encuentre apagado, si lo anterior se cumple se envía la orden al arduino que controla la etapa de potencia para encender al actudor correspondiente. En el segundo *if* se verifica si la variable obtenida por la consulta es menor que la preferencia del usuario y que el estado del actuador sea activo, si se cumplen las condiciones anteriores se envía la orden al controlador correspondiente de apagar la etapa de potencia que controla al actuador.

El segundo ciclo *while* verifica que el número de personas y el modo de operación sean cero y modo de operación manual (cero) respectivamente ya que este ciclo solo se encarga de hacer consultas continuas a la ontología con el objetivo de actualizar la información que maneja, por ejemplo, que ingrese una persona al lugar y se regrese al ciclo anterior, además de que permite la activar o desactivar a los actuadores de manera manual desde la aplicación móvil.

## 5.4. Consumo de servicios con android

Uno de los objetivos es presentar la información por algún medio que sea fácil de entender y utilizar por el usuario, por lo que un dispositivo móvil e inteligente *(smart phone)* es la mejor opción por su versatilidad y presencia en la vida de los usuarios.

La aplicación desarrollada en la plataforma Android cuenta con cuatro sub funciones, que están orientadas al consumo de cada uno de los cuatro servicios Web, de modo que el usuario puede seleccionar a cada una de ellas para obtener la información que desee consultar. Además que permite controlar de manera directa a los dispositivos actuadores del lugar en el que se encuentra.

Por tanto se va a explicar cada una de las cuatro funciones que posee la aplicación y el cómo se envían las peticiones al servicio, además de la forma en la que se presentan los datos al usuario en su dispositivo.

### 5.4.1. Consumo del servicio de localización.

Esta primer función hace llamadas al servicio de localización, hay que recordar que este servicio muestra al usuario la información ambiental del lugar en el que se encuentra.

```
1  private static final String NAMESPACE = "http://nfc.
       ubiquitous.cinvestav.edu/";
2  private static final String URL = "http://aisii.azc.uam.mx
       :8080/Semantic_WS/soap/semantic_ws?wsdl"?
3  private static final String METHOD_NAME = "name";
```

Script 5.12: Consumo del primer servicio

En el script 5.12 se muestran las líneas de código necesarias en todas las demás funciones para el consumo de los servicios. En la línea 1 se presenta la cadena del nombre del espacio sobre el cual se va a trabajar. Para este caso se crea un espacio que hace referencia al lugar de trabajo en el que se encuentra el usuario, el *namespace* puede ser cualquier lugar que el usuario defina.

En la siguiente línea es visible la URL del servidor, que ejecuta a los servicios, es necesario especificar todo la ruta hasta el archivo **WSDL** que contiene la información de los servicios (*véase línea 2 del script 5.12*). También se debe de especificar el nombre del método que se va a invocar, por ejemplo, en la línea 3 se presenta el nombre el método que para nuestro caso es *name*, hay que recordar que el nombre del servicio es *getLocationInfo* y su método es llamado *name*.

Por consiguiente estos son los parámetros necesarios para poder consumir a cualquiera de los servicios desarrollados, cabe recalcar que cada uno de los servicios tiene un nombre diferente y por consiguiente el nombre del parámetro también cambia para evitar inconsistencias al momento de invocar al servicio que se desee.

```
1  SoapSerializationEnvelope envelope = new
       SoapSerializationEnvelope(SoapEnvelope.VER11);
2  envelope.implicitTypes = true;
3  envelope.setAddAdornments(false);
4  SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
5  request.setInnerText(name);
6  envelope.setOutputSoapObject(request);
7  envelope.skipNullProperties = true;
8
9  HttpTransportSE androidHttpTransport = new HttpTransportSE(
       URL, 300000);
10 androidHttpTransport.debug = true;
11 androidHttpTransport.setXmlVersionTag("<!--?xml version
       =\"1.0\" encoding= \"UTF-8\" ?-->");
```

```
12
13 try{
14     androidHttpTransport.call(SOAP_ACTION, envelope);
15     Vector response = (Vector)envelope.getResponse();
16
17     SoapPrimitive data = (SoapPrimitive)response.get(0);
18     locationInfo.setHum(data.toString());
19     }
```

Script 5.13: Envío de parámetros al servicio

Para poder consumir servicios desde android es necesaria una biblioteca llamada *kSOAP2,*[1] ya que permite el consumo de servicios tipo SOAP, el script 5.13 presenta la estructura de la función que envía la petición al servicio. En la línea 1 se crea un "sobre" que contiene la estructura con los parámetros necesarios para consumir al servicio. Para este caso el sobre tiene el nombre de *envelope* y es del tipo SOAP version 1.1, lo que le permite ser usado por casi por cualquier cliente que soporte al menos ésa versión del protocolo SOAP.

Para crear la petición en la línea 4 se indica que se va a crear una petición del tipo *SoapObjet* llamada *request*, cuyos parámetros de entrada son las dos cadenas *NAMESPAE, METHOD_NAME* mencionadas anteriormente en el script 5.12, después en la línea 5 se agrega la cadena asociada al servicio que se va a consumir, en este caso es *"name"* y el servicio es *"location"*, la cadena *name* contiene al nombre del usuario que se ha ingresado al dispositivo móvil para consultar la localización de uno mismo como usuario del sistema.

Para hacer posible el envío de la petición se debe enviar en un "sobre" por lo que en la línea 6 se indica que el objeto de salida será un objeto SOAP con el contenido de *request*, además se indica que se van a ignorar las propiedades nulas en la línea 7.

Para llevar a cabo la comunicación a través de la red ya sea Internet o una red de área local se utiliza el protocolo de transporte HTTP. En la línea 9 se crea el objeto llamado *androidHttpTransport* haciendo uso de *HttpTransportSE*, que se encuentra de forma nativa en el *SDK* (Software Development Kit) de Android. Además de los parámetros que se ingresan, como son la URL del archivo WSDL y el tiempo de espera máximo. En la línea 11 se indica que la versión es la 1.0 de *XML* con una codificación del

[1] http://kobjects.org/ksoap2/index.html última vez consultado 20-7-16

tipo *UTF-8*.

Una vez que se han definido los parámetros principales para que el cliente pueda consumir al servicio, se procede a utilizar el mecanismo de control de excepciones por medio de los bloques *(try , catch)*. En la línea 14 se invoca al servicio haciendo uso de *adroidHttpTransport* con los parámetros *"SOAP_ACTION" y "ENVELOPE"* para procesar la respuesta del servicio se crea un objeto del tipo "Vector", que tiene la propiedad de cambiar su tamaño con base en el número de elementos que tiene un objeto, es decir, si la respuesta del servicio fuese un objeto con dos elementos el vector tendrá un tamaño de dos, en la línea 15 se presenta la manera en la que se hizo uso del vector llamado *"response"* al asociar el comando *.getResponse()* se indica que *response* va almacenar todo el contenido que viene en la respuesta del servicio.

Para hacer la uso de la información contenida en *response* se indica que es del tipo *SoapPrimitive* y que se guardara en una variable llamada *"data"*, la línea 17 se indica que se va a copiar el contenido de *response* en su posición cero a *data*.

Una vez hecho esto se asocia el valor obtenido a la propiedad correspondiente del objeto *"locationInfo"*, que cuenta con las distintas variables con las que se cuenta en la respuesta del servicio, por ejemplo, en la línea 18 se asocia el valor de *data* a la humedad del lugar en que se encuentra el usuario. Se usa la misma estructura para los demás datos dentro del objeto *locationInfo* que cuenta con: humedad, temperatura, luz, nombre del lugar, letra, número que tiene asignado el lugar, nivel y el nombre del usuario que hizo la solicitud al servicio.

### 5.4.2. Consumo del servicio de búsqueda.

En esta segunda función se hace el consumo del servicio de búsqueda llamado *"search"* cuyo parámetro de entrada es una cadena llamada *"nom"*, este servicio es similar a su predecesor llamado *"location"* con la diferencia principal en la cantidad de información que el servicio devuelve, como respuesta a la petición del cliente. A continuación se presentan los parámetros que marcan la diferencia entre el consumo del servicio anterior y éste (*véase script 5.14*).

```
1  private static final String NAMESPACE = "http://nfc.
       ubiquitous.cinvestav.edu/";
2  private static final String URL = "http://aisii.azc.uam.mx
       :8080/Semantic_WS/soap/semantic_ws?wsdl"?
3  private static final String METHOD_NAME = "nom";
```

Script 5.14: Consumo del segundo servicio

Los parámetros para invocar al servicio y hacer una petición no cambian, como se pueden observar las líneas 1 y 2 siguen siendo las mismas que en el servicio anterior, la única diferencia entre estos dos servicios es el nombre del método. Para llevar a cabo el consumo del servicio de búsqueda de personas se debe utilizar en método *"nom"* que se muestra en la línea 3.

En ese parámetro de entrada se debe enviar el nombre de la persona que se desea localizar, debe ser el nombre completo, ya que este servicio está pensado para su consumo por los estudiantes que desean buscar a algún profesor dentro de la institución.

```
1  try{
2      androidHttpTransport.call(SOAP_ACTION, envelope);
3      Vector response = (Vector)envelope.getResponse();
4      SoapPrimitive data = (SoapPrimitive)response.get(0);
5      searchInfo.setAlfabeto(data.toString());
6
7      data = (SoapPrimitive)response.get(1);
8      searchInfo.setLugar(data.toString());
9
10     data = (SoapPrimitive)response.get(2);
11     searchInfo.setNum(data.toString());
12
13     data = (SoapPrimitive)response.get(3);
14     searchInfo.setPiso(data.toString());
15     }
```

Script 5.15: Respuesta del segundo servicio

En el script 5.15 se presenta la forma de la respuesta del servicio, los datos que vienen dentro del paquete respuesta se guardan dentro de un objeto llamado *"searchInfo"*. El objeto *searchInfo* contiene a las variables llamadas *alfabeto*, *lugar*, *num* y *piso*. Donde:

- "alfabeto" hace referencia a la letra con la que se identifica al edificio.
- "lugar" es el nombre que tiene asignado el espacio físico ya que puede contar con un nombre adicional a la nomenclatura que tiene el edificio.

- "num" es el número asignado al espacio físico
- "piso" es el nivel en que se encuentra el lugar en el cual se ubica la persona que se esta buscando.

Toda esa información puede ser de utilidad para saber como moverse dentro de la institución. De igual forma se utilizan los mismos métodos para hacer una petición al servicio y para procesar la respuesta se sigue haciendo uso de vector para guardar el contenido de la respuesta que posteriormente se asocia a una variable llamada *"data"*, la cual se utiliza para asignar los valores a las variables contenidas en el objeto *searchInfo*.

### 5.4.3. Consumo del servicio de control manual.

Para hacer uso de la información que el servicio de control manual-automático devuelve como respuesta, el usuario debe ingresar el nombre del lugar en el que se encuentra para poder hacer uso de los dispositivos actuadores que se encuentran a su disposición en el espacio físico que lo rodea *(véase script 5.16)*.

```
1 private static final String NAMESPACE = "http://nfc.
      ubiquitous.cinvestav.edu/";
2 private static final String URL = "http://aisii.azc.uam.mx
      :8080/Semantic_WS/soap/semantic_ws?wsdl"?
3 private static final String METHOD_NAME = "lugar";
```

Script 5.16: Consumo del tercer servicio

Como en los servicios anteriores la línea 3 es la más importante para este servicio ya que define el nombre del servicio que se va a consumir por lo que para este caso es "lugar". El nombre del lugar es lo más importante para obtener la información de los dispositivos actuadores y controlarlos de manera manual por medio de la aplicación en el dispositivo móvil. Además hay que recordar a las líneas 1 y 2 que no cambian para nada ya que los servicios se encuentran en ejecución en el mismo servidor. En la línea 3 el parámetro de entrada es del tipo cadena y es necesario convertir los datos el tipo indicado para evitar errores.

```
1 HttpTransportSE androidHttpTransport = new HttpTransportSE(
      URL, 300000);
2 androidHttpTransport.debug = true;
```

```
androidHttpTransport.setXmlVersionTag("<!--?xml version
    =\"1.0\" encoding= \"UTF-8\" ?-->");

try{
    androidHttpTransport.call(SOAP_ACTION, envelope);
    Vector response = (Vector)envelope.getResponse();

    SoapPrimitive data = (SoapPrimitive)response.get(0);
    actuatorInfo.setEstado1(data.toString());
    }
```

Script 5.17: Respuesta del tercer servicio servicio

Cabe resaltar que en el script 5.17 se muestra la forma en la que se consume este servicio y es similar a los servicios anteriores, para este caso y los anteriores cabe destacar que se ha creado un objeto con las propiedades necesarias para almacenar los datos que devuelve el servicio como respuesta, el objeto utilizado para llevar a cabo las acciones deseadas por el usuario.

En el caso del servicio de control manual-automático hay que recordar que devuelve información variada acerca de los dispositivos y las condiciones ambientales del lugar en que el usuario se encuentra, por lo que se debe guardar dicha información para activar una serie se botones que permiten la interacción del usuario con el sistema de manera directa sin la necesidad de acercarse directamente al dispositivo actuador.

Se muestra solo una parte del bloque de respuesta ya que la información que devuelve el servicio contiene demasiados datos en los que podemos encontrar, el nombre del dispositivo, la dirección IP del dispositivo que controla al actuador, su estado ya sea activo o inactivo, y los promedios de temperatura, humedad y luz del lugar que el usuario a consultado por medio del servicio.

```
b8 = (Button) findViewById(R.id.button8);
b8.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) {
 if(actuatorInfo.getEstado1() != null && actuatorInfo.getIp1
     () != null){
  int pin1 = 2;
  edo = Integer.parseInt(actuatorInfo.getEstado1());
  Compara(actuatorInfo.getIp1(), pin1, edo, actuatorInfo.
      getNombre1());
  }
```

```
10    }
11  }) ;
```

Script 5.18: Botón de control manual en la aplicación

Para hacer uso de los botones que se presentan en la interfaz gráfica de la aplicación, primero se deben declarar dentro de la aplicación, asignarles un identificador y posteriormente agregar las acciones que se llevarán a cabo cuando el botón es "presionado" por parte del usuario para ejecutar una acción, (los botones se pueden observar en las pruebas del capítulo siete). En script 5.18 se presenta la forma en que se usan los botones en la aplicación, por lo que los demás tienen una estructura similar al presentado en esta figura.

En la línea 1 se crea el objeto de tipo *button* llamado **b8** el cual puede ser encontrado por su identificador en la parte de *(R.id.button8)*, además en la línea 2 se agrega al mecanismo de escucha, el cual se encarga de estar al pendiente de las acciones que se llevan a cabo en el panel táctil y asocia dichos eventos al botón, es decir, si el usuario toca la pantalla y la región en la que hizo contacto hay un botón el sistema automáticamente ejecuta las acciones que tiene dicho botón u objeto.

Cabe resaltar que una vez que el usuario ha hecho la consulta y los datos de la respuesta se han almacenado, sólo en ese momento los botones se activan. Una vez obtenidos los datos se procede a verificar que no sean elementos nulos para así activar el bottón y las acciones que puede llevar a cabo. En la línea 5 se verifica que tanto el estado de los dispositivos como su dirección IP no sean valores nulos.

A continuación se debe asingar el número de pin que tiene el controlador para enviar directamente el comando a dicho pin en el que está conectado la etapa de potencia que activa o desactiva al actuador.

Una vez hecho lo anterior se debe convertir el valor de estado al tipo entero para facilitar la comparación que se lleva a cabo en la función que se encuentra en la línea 8, por tanto, la línea para llevar a cabo la conversión a tipo entero es la línea 7 ya que en la línea 8 se deben enviar los como parámetros la IP, el pin, el estado y el nombre del dispositivo que se desea controlar a la función *Compara* que es la encargada de construir el comando que se enviará al controlador por medio de una URL.

```
1  public void Compara(String ip, int pin, int edo, String
       device){
2   if(edo == 0 ){
3    dev = "http://" + ip + "/arduino/digital/" + pin + "/1";
4   }
5
6   if(edo == 1){
7    dev = "http://" + ip + "/arduino/digital/" + pin + "/0";
8   }
9
10  Intent i = new Intent();
11  i.setData(Uri.parse(dev));
12  startActivity(i);
```

Script 5.19: Función compara

El script 5.19 muestra las acciones llevadas a cabo para construir dicha URL, como ya se dijo antes, esta función recibe cuatro parámetros y a través de una comparación se construye la URL adecuada para la acción del usuario. En las líneas 2 y 6 se lleva a cabo la comparación del estado del dispositivo, para saber si se encuentra activo o inactivo, en ambos casos se construye una cadena en la variable **dev** y para los demás botones se sigue el mismo procedimiento.

Como se puede observar se utiliza la dirección IP del dispositivo controlador del actuador por eso es un parámetro indispensable para nuestro objetivo, como se puede observar en las líneas 3 y 7 son prácticamente iguales lo único que cambia es número al final el cual indica el estado de *pin* del controlador. Si el numero es un cero el *pin* del controlador debe cambiar a estado "bajo" y por tanto la etapa de potencia se desactiva, apagando al actuador.

Una vez formada la cadena se crea un objeto llamado *i* que es una instancia de *Intent*, como se puede observar en la línea 10; *intent* nos auxilia al hacer llamadas a otras aplicaciones fuera de la nuestra, por lo que es útil para enviar el comando a través de un explorador web integrado en el *smartphone*. En la línea 11 se hace el *parser* de la cadena contenida en la variable **dev** y porsteriormente en la línea 12 se hace una llamada a la aplicación que va a enviar el comando directamente al dispositivo para activar o desactivar al actuador que el usuario desee.

### 5.4.4. Consumo del servicio de preferencias.

Por último se presenta el consumo del servicio de ingreso de preferencias el cual tiene como objetivo recibir las preferencias ambientales de usuario y hacerlas disponibles para el programa demonio. hay que recordar que este servicio recibe cinco parámetros diferentes, los cuales debe ingresar el usuario de forma manual y el servicio debe responder con las mismas variables como confirmación acerca de los datos que el usuario ingreso.

Los parámetros son obligatorios y en caso de que el usuario de no ingrese alguno la misma aplicación le indicará que falta alguno y no lleva a cabo ninguna acción, por lo tanto el usuario debe ingresar el nombre del lugar, el modo de operación del programa demonio, la cantidad de temperatura, humedad y luz preferidos.

```
1 private static final String NAMESPACE = "http://nfc.
      ubiquitous.cinvestav.edu/";
2 private static final String URL = "http://aisii.azc.uam.mx
      :8080/Semantic_WS/soap/semantic_ws?wsdl"?
3 private static final String METHOD_NAME = "userInfo";
```

Script 5.20: Consumo del cuarto servicio

El script 5.20 muestra la información necesaria para el consumo del servicio de ingreso de preferencias en la línea 3 se debe ingresar el nombre "userInfo" que es el objeto que contiene las variables de todas las preferencias del usuario.

```
1  SoapSerializationEnvelope envelope = new
       SoapSerializationEnvelope(SoapEnvelope.VER11);
2  envelope.implicitTypes = true;
3  envelope.setAddAdornments(false);
4  SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
5
6  PropertyInfo propertyInfo = new PropertyInfo();
7
8  propertyInfo.setName("sel")
9  propertyInfo.setValue(select);
10 propertyInfo.setType(Integer.TYPE);
11 request.addProperty(propertyInfo);
```

Script 5.21: Propiedades del cuarto servicio

El objeto *userInfo* está definido con las preferencias mencionadas, por lo que cada una de ellas debe ser puesta de forma explicita, al momento

de crear el paquete, por lo que se deben de agregar como propiedades del objeto. De principio el script 5.21 es una muestra del programa para llevar a cabo el consumo en la línea 6 se crea una instancia llamada *propertyInfo*, la cual es un objeto que cuenta con las variables que va a recibir el servicio por lo que es necesario utilizar un objeto que cumpla con las propiedades necesarias.

Una vez creado el objeto en la línea 8 se declara el nombre de una de las variables, en la siguiente línea se debe asignar un valor, el cual es capturado por medio de un cuadro de texto asociado a la variable *"select"*.Aademás se debe indicar que tipo de variable es por lo que en la línea 10 se hace la indicación al tipo entero y por último se agrega dicha propiedad al *request* que se ha creado en la línea 4.

Cabe destacar que lo anterior se debe realizar para los parámetros que sea necesario enviar ya que podrían existir servicios que requieran más de dos parámetros.

```
1  try{
2   androidHttpTransport.call(SOAP_ACTION, envelope);
3   Vector response = (Vector)envelope.getResponse();
4
5   SoapPrimitive data = (SoapPrimitive)response.get(0);
6   prefset.setHum(Integer.parseInt(data.toString()));
7
8   data = (SoapPrimitive)response.get(1);
9   prefset.setLux(Integer.parseInt(data.toString()));
10
11  data = (SoapPrimitive)response.get(2);
12  prefset.setPlace(data.toString());
13
14  data = (SoapPrimitive)response.get(3);
15  prefset.setSel(Integer.parseInt(data.toString()));
16
17  data = (SoapPrimitive)response.get(4);
18  prefset.setTemp(Integer.parseInt(data.toString()));
19 }
```

Script 5.22: Respuesta del cuarto servicio

Una vez que se han agregado todas las propiedades y características correspondientes a *request*, se procede a hacer la petición al servicio como se ha hecho anteriormente empleando los bloques *try*, En el script 5.22 se puede observar que en las líneas 5, 8, 11, 14 y 17 se lleva a cabo la obtención de los

datos provenientes de la respuesta del servicio, cada uno es la confirmación de las preferencias del usuario. También para las líneas 6, 9, 12, 15 y 18 se lleva a cabo la asociación de cada uno de los datos guardados en la variable *data* a su variable correspondiente, para poder visualizarla de manera adecuada. En la línea 6 se asigna el valor de la humedad como del tipo entero al realizar el análisis sintáctico, dicha tarea se lleva a cabo también en las demás líneas a excepción de la línea 12 en la que es un cadena lo que devolvió el servicio, la cual es el nombre del lugar que el usuario ingreso como parámetro.

Este conjunto de aplicaciones tiene como objetivo el consumir a los servicios por medio de un dispositivo móvil, permitiendo así que el usuario no se pierda con aplicaciones demasiado llamativas ni demandantes que puedan distraer su atención. ada uno de los servicios es invocado sólo cuando el usuario lo requiere y la forma en la que se le solicita y presenta la información al usuario es sencilla para que no exista ambigëdad en los datos, ya que todos los servicios tienen objetivos diferentes y por tanto información distinta que puede ser de utilidad para el usuario.

# Capítulo 6

# Pruebas y resultados

En este capítulo se presentan las pruebas y resultados de nuestra propuesta. Cabe destacar que la red de sensores cuenta con la capacidad de controlar, algunos dispositivos actuadores por medio de etapas de potencia. Para el caso del sistema completo se propone un escenario de prueba con el que se simula el ingreso de datos y la obtención de información lo que contempla las inferencias en la meta-ontología por medio de las reglas, las consultas directas a la meta-ontoloía, además de la interacción con el usuario por medio de los servicios Web. Para obtener la información necesaria a fin de controlar los dispositivos de forma manual o enviar sus preferencias al programa demonio en modo de operación automático.

## 6.1. Pruebas individuales de arduino

Para las pruebas a los servidores ligeros en cada arduino se llevaron a cabo consultas directas a las direcciones IP de éstos por medio de exploradores Web, para obtener la información de los sensores y actuadores según sea el caso. También se presentan capturas de pantalla de los datos que se obtienen y del código fuente de la página web como los presenta el servidor ligero.

A continuación se presentan las respuestas que devuelven los servidores ligeros de HTTP montados en cada uno de los arduinos. Cada arduino cuenta con distintos sensores y por tanto los programas para cada uno son diferentes. Además se desea presentar los datos de forma estructurada para que sean fácil de obtener. Por lo que se hace uso de la etiqueta "<a>[1]" que permite crear atributos con los que es posible representar a los dispositivos

[1] https://www.w3.org/TR/html-markup/a.html

y ligar los datos obtenidos por medio de los sensores. Para el caso de los actuadores el atributo *estado* es el encargado de indicar si se encuentra activo o inactivo el dispositivo actuador.

```
1  <html><body>
2  <a class="Sensor" id="DHT11A"> DHT11A </a>
3  <a name="DHT11A" id="hasTempValueA"> 25.00 </a>
4  <a name="DHT11A" id="hasHumidityValueA"> 38.00 </a>
5  <a class="Sensor" id="DHT11B"> DHT11B </a>
6  <a name="DHT11B" id="hasTempValueB"> 41.00 </a>
7  <a name="DHT11B" id="hasHumidityValueB"> 9.00 </a>
8  <a class="Sensor" id="LightA"> LightA </a>
9  <a name="LightA" id="hasLightValueA"> 886.00 </a>
10 <a class="Sensor" id="LightB"> LightB </a>
11 <a name="LightB" id="hasLightValueB"> 712.00 </a>
12 <a class="Actuator" id="Lum1"> Lum1 </a>
13 <a name="Lum1" id="hasStateLum1"> 0 </a>
14 <a name="Lum1" id="hasIPAddressLum1"> 192.168.0.109 </a>
15 <a class="Actuator" id="Fan1"> Fan1 </a>
16 <a name="Fan1" id="hasStateFan1"> 0 </a>
17 <a name="Fan1" id="hasIPAddressFan1"> 192.168.0.109 </a>
18 </body></html>
```

Script 6.1: Respuesta del servidor de sensores y actuadores

El resultado de la ejecución del programa del servidor ligero se muestra en el script 6.1. Se muestra la respuesta que devuelve el servidor de HTTP que está ejecutando en arduino. Para esta primer prueba se usaron dos sensores de temperatura que, además de medir la temperatura, también miden la humedad relativa en porcentaje del lugar en el que se encuentran.

Estos sensores tienen un grado de error considerable y puesto que para este trabajo no es crítico tener mayor exactitud en las lecturas de temperatura se optó por éstos sensores de gama baja. Además se agregaron dos sensores que miden la intensidad luminosa, pero a diferencia de los anteriores estos sensores son mucho más sensibles e incluso tiene una doble funcionalidad, no solo miden la intensidad luminosa también pueden disparar un pulso al llegar a un umbral de luz, el cual puede ser definido por el usuario. A continuación se muestra la respuesta de las lecturas tomadas y mostradas por medio del servidor Web.

También se han agregado dos actuadores el primero llamado "Lum1" el cual tiene asignado un bit de salida del arduino, con el cual se controla la etapa de potencia para activar o desactivar las luminarias que se controlan por medio del programa demonio o en modo manual haciendo uso del dispo-

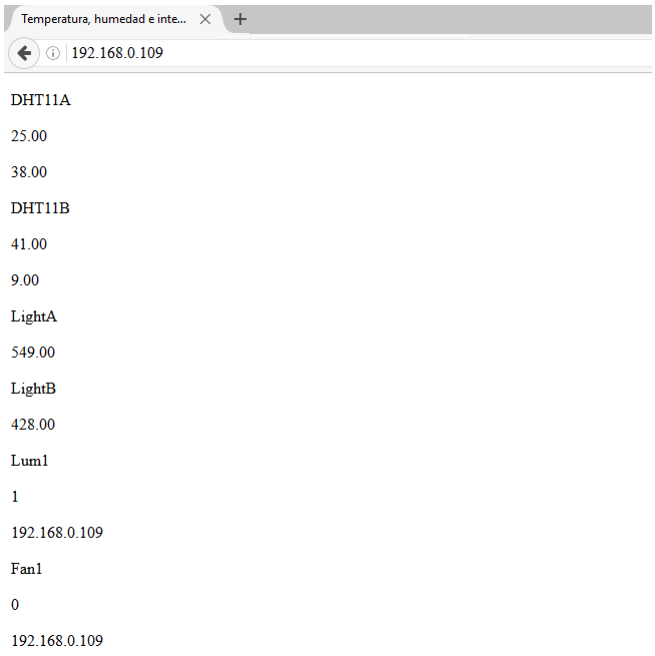


Figura 6.1: Petición desde explorador Web

sitivo móvil. El segundo dispositivo actuador es llamado "Fan1" e igual que "Lum1" tiene asignado un bit de salida del arduino, con el cual también se controla a una etapa de potencia para activar o desactivar, en este caso, un ventilador. En el scritp 6.1 se pueden observar las lecturas de los sensores, junto con su identificador y a qué clase en la meta-ontología pertenecen.

Se presentan los valores de las lecturas de humedad, temperatura, cantidad de luz, la cual puede tomar valores entre 0 y 1000 en los que cero es una gran cantidad de luz y mil (1000) significa obscuridad absoluta. Para el caso de los actuadores se presenta la dirección IP del arduino y el estado del dispositivo, ya que son los datos necesarios que utiliza el programa demonio para llevar a cabo las acciones de acuerdo a las preferencias del usuario.

En las fíguras 6.1 y 6.2 se muestran capturas de pantalla de las respuestas del servidor de HTTP ligero que ejecuta en el primer arduino. Como se puede observar en las dos figuras, para los sensores de temperatura nombrados "DHT11A" y "DHT11B" se tienen distintas lecturas de temperatura y hu-

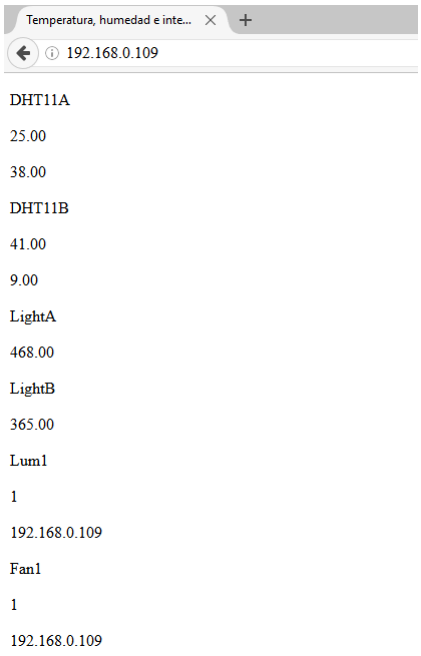


Figura 6.2: 'Fan1' y 'Lum1' activos

medad, ya que ambos fueron colocados alejados uno de otro y por tanto sus lecturas son diferentes. Lo mismo sucede para los sensores de luz nombrados "LightA" y "LightB" en este caso el lugar cuenta con paredes blancas por lo que refleja más luz y es por eso que las lecturas no presentan tanta diferencia.

Para los actuadores "Lum1" y "Fan1" se presentan sus direcciones IP, en la figura 6.1 sólo esta activo en actuador "Lum1" ya que "Fan1" esta inactivo indicado por los numeros "uno y "cero" respectivamente. En la fígura 6.2 ambos actuadores se encuentran activos, para llevar a cabo la activación o desactivación de un actuador es necesario enviar el comando por medio de la URL por lo se debe crear dicha URL con base en la dirección IP del arduino al que se encuentra conectado el actuador, se debe usar el nombre del dispositivo y el estado que se deseado.

Para el segundo servidor de HTTP que esta ejecutando en el segundo arduino MEGA 2560, se presenta la respuesta del servidor en el script 6.2, la respuesta cuenta con un conjunto de sensores los cuales nos ayudan a saber

cuando una persona entra o sale del espacio físico que se está censando, por lo que podemos saber la cantidad de personas que se encuentran dentro del lugar.

El programa que se diseñó para este objetivo además, de contar a las personas que se encuentran en un lugar, tambíen asocia un espacio físico a una etiqueta de *html* la cual hace referencia al lugar que se está censando.

```
1 <html><body>
2 <a class="Sensor" id="RFIDReader"> RFID </a>
3 <a name="RFIDReader" id="idCardRead"> 16319618317200015 </a>
4 <a name="RFIDReader" id="idNumberRead"> 141270010 </a>
5 <a class="Cubicle" id="H254"> H254 </a>
6 <a name="H254" id="hasNumberOfPeople"> 1 </a>
7 </body></html>
```

Script 6.2: Lectura de tarjetas RFID y contador de personas

Además, el diseño contempla un lector de tarjetas RFID con las que es posible identificar a los usuarios que hacen un "inicio de sesión", es decir, el usuario debe acercar su tarjeta o credencial RFID al lector antes de entrar al lugar, para que el sistema asocie su número de identificación al lugar en que se encuentra el sensor y así localizar de forma adecuada al usuario.

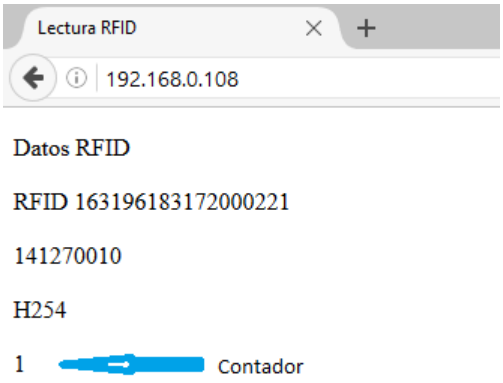


Figura 6.3: Consulta desde explorador Web

Al hacer la lectura del contenido de la tarjeta se obtienen dos valores uno de ellos es el de *"idCardRead"*, es el número de identificación de la tarjeta que tiene asignado por el fabricante y no puede ser cambiado. El segundo valor es el llamado *"idNumberRead"* y es asignado por la institución, ya que puede ser un número de empleado o matrícula de estudiante por lo que con base en este segundo número se puede identificar al usuario de forma más certera ya que está asociado a su perfil en la ontología.

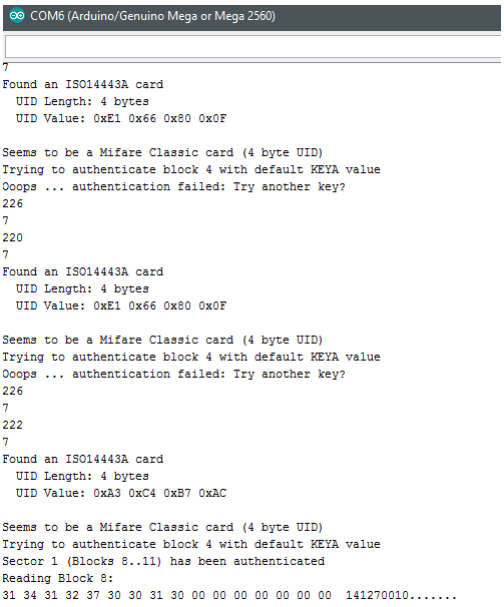


Figura 6.4: Muestra de lecturas a través del puerto serial

En las figuras 6.3 y 6.4 se presentan una captura de pantalla de como se ve la respuesta del servidor ligero de HTTP a través del explorador Web. Como se puede observar cuenta con los valores de las lecturas y a que propiedades y clases pertenecen e incluso el individuo que esta representado en la ontología. En la figura 6.3 se muestra la lectura de una tarjeta RFID y el incremento en el *contador* al censar que el individuo ingreso al espacio físico (indicado con el número uno), además de su número de matrícula el cual es el segundo número con menor número de dígitos .

Para la figura 6.4 se muestra la información que envía el arduino por medio del puerto serie-usb a la computadora, en dicha figura se muestra la lectura de algunas tarjetas no registradas o con llaves de acceso diferentes, al final de la figura de muestra la lectura de la misma tarjeta que en la figura anterior y se puede observar que el número asignado por el fabricante esta en hexadecimal y el número de matrícula está almacenado en el bloque ocho de la tarjeta RFID y se muestra en hexadecimal y en decimal.

También se puede observar las lecturas de los sensores de distancia de ultrasonido y el contador de personas al interior del espacio físico. Los valores de las lecturas de los sensores son las que tienen valores de doscientos en adelante y el contador de personas es el de menor valor, para este caso es el valor de siete. Esta muestra del puerto serial presenta mayor cantidad de datos acerca de las lecturas tanto de los sensores como del lector de tarjetas RFID y no puede ser utilizada para inyectar información a menos que se desee hacer por ese medio.

## 6.2. Pruebas a los servicios

El escenario de pruebas comprende a un usuario de nuevo ingreso que desea saber en qué lugar de la institución está ubicado. La institución le otorgó la aplicación móvil, con la cual puede obtener la información necesaria de su ubicación e incluso el buscar a alguna persona que se encuentre dentro de la institución, Para esta prueba se han montado los servicios en un servidor apache Tomcat 9.0, el cual es ejecutado en una computadora de escritorio con Linux. El equipo cuenta con un procesador AMD Athlon 64 X2 de doble núcleo y 4 Gb de memoria RAM a 800 MHz.

La figura 6.5 muestra una captura de pantalla de la visualización del archivo WSDL a través del explorador Web. En dicha imagen es visible la información acerca de los distintos servicios que se tienen, los tipos de datos que se manejan e incluso los tipos de datos que devuelve como respuesta al recibir una petición de algún cliente.

Entre los servicios que se ofrecen podemos encontrar a los mencionados anteriormente con su nombre, por ejemplo, *"userInfo"* el cual invoca a *"location"* con el objetivo de recibir el nombre del usuario en forma de cadena y devolver como respuesta a las variables ambientales con base en su tipo de dato. Para este servicio se devuelve la humedad del lugar, temperatu-


Figura 6.5: Servicios desde el explorador Web

ra, cantidad de luz, el piso en que se encuentra el lugar, nombre del lugar, número del lugar y letra del edificio en que está localizado el lugar. Todos los datos tienen que ver con la localización del usuario al momento de la consulta. También se pueden observar los servicios *"actuator"* y *"userInfo"*, los cuales ya han sido mencionados y explicados en el capítulo anterior.

### 6.2.1. Pruebas individuales a los servicios

Las pruebas que se llevaron a cabo a los servicios se realizaron por medio de de la aplicación móvil y de un un programa llamado SoapUI[1], que permite realizar pruebas a servicios de tipo REST y SOAP. Para corroborar la información obtenida se ejecutaron las mismas consultas en el software protégé y se compararon los resultados. Del mismo modo la información de las lecturas de los sensores se ingresó para verificar que se obtienen los promedios de temperatura, humedad y luz, y que el usuario se encuentra en el lugar en que hace chequeo de entrada.

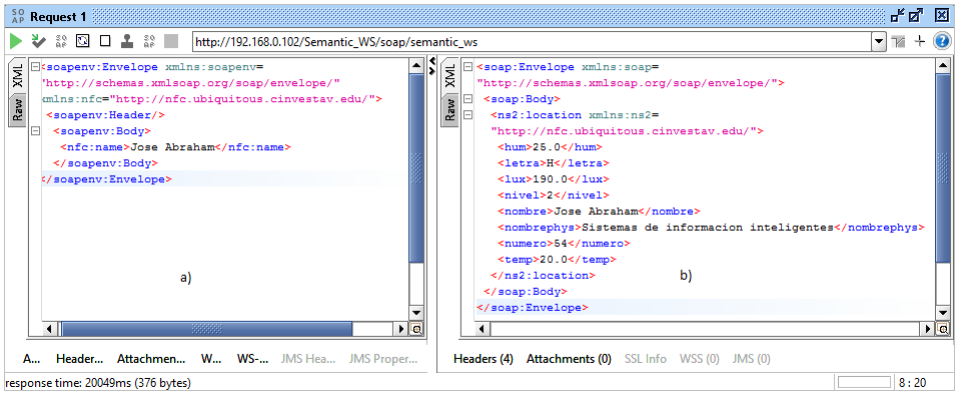


Figura 6.6: Pruebas al servicio de localización desde soap UI

La figura 6.6 es una captura de pantalla de la respuesta a la petición del usuario para localizarse dentro de la institución. Del lado *"a)"* de la figura se muestra que en la etiqueta *"<nfc:name>"*, el usuario ingresa su nombre con el objetivo conocer su localización dentro de la institución. Del lado *"b)"* se presenta la respuesta que devuelve el servicio con diferentes etiquetas, las cuales identifican a la información en la respuesta, por ejemplo, la etiqueta *"<hum>"* contiene la humedad promedio del lugar en que se encuentra el usuario, *"<nombrephys>"* contiene el nombre del espacio físico, que puede estar identificado por un número y un nombre o en algunos casos sólo por el número que tiene asignado.

La figura 6.6 es la respuesta que se obtiene al hacer una petición al servicio *getLocatioInfo* y al consumirlo desde una aplicación móvil que juega

[1]https://www.soapui.org/

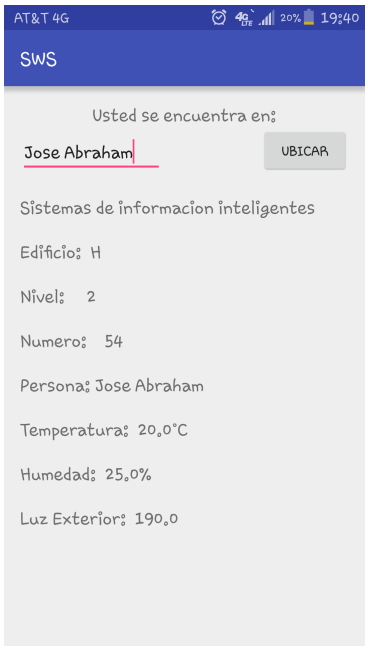


Figura 6.7: Pruebas al servicio de localización desde android

el papel de cliente. Se presenta información con más detalle ya que el cliente se diseñó pensando en que el usuario final necesita la información clara y directa. Por lo que en la figura 6.7 presenta una captura de pantalla de la aplicación móvil al ejecutar la misma consulta con los mismos datos que en la figura 6.6.

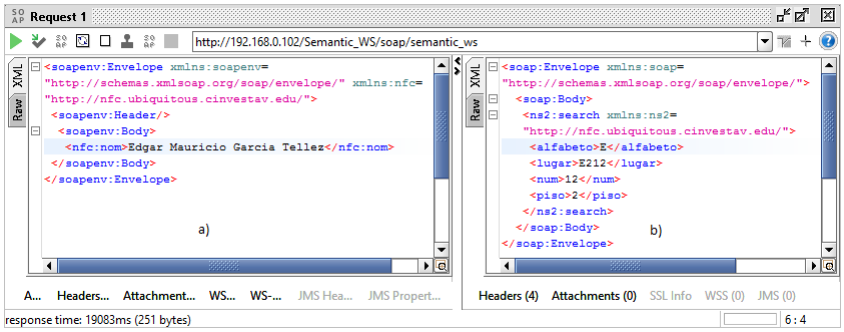


Figura 6.8: Prueba al servicio de búsqueda desde soap UI

Para la prueba del servicio de búsqueda se presenta la figura 6.8 en la que se puede observar que el parámetro que se envía es el nombre de la persona que se desea buscar. Del lado *"a)"* de la figura se encuentra la etiqueta *"<nfc:nom>"*, con el nombre de un usuario en la ontología llamado *Edgar Mauricio Garcia Tellez*. Hay que tomar en cuenta que los acentos en los nombre o palabras se deben omitir ya que no todos los idiomas los manejan ni reconocen.

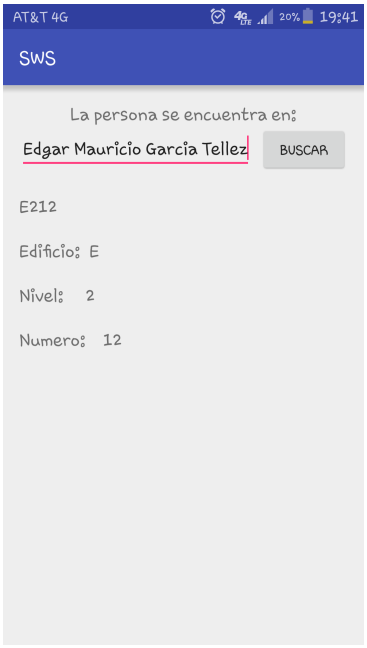


Figura 6.9: Prueba al servicio de búsqueda desde android

Los Resultados que se pueden observar en la figura 6.8*"b)"* al ejecutar las peticiones desde el programa soap UI *(véase figura 6.8)* y android *(véase figura 6.9)*, es que la información que se obtiene es las misma para al consumir el servicio ya que el servicio devuelve información directa de la meta-ontología por lo que no hay diferencia en la información que el servicio devuelve como respuesta.

De las misma manera que se han llevado a cabo las pruebas a los servicios anteriores, para esta prueba se desea obtener la información de los dispositivos actuadores que se encuentran en un espacio físico dentro de la


Figura 6.10: Obtención de datos de los actuadores con soap UI

institución, En la figura 6.10 *"a)"* se muestra el cliente que envía el nombre del lugar a través de la etiqueta *"<lugar>"*, para este caso se envía el nombre *"Sistemas de informacion inteligentes"*.

En la figura 6.10 *"b)"* se presenta la información que se regresa como respuesta, dicha respuesta contiene los estados de los tres dispositivos encontrados en el espacio físico, sus direcciones IP y demás información que puede ser de utilidad para el usuario "experto" que desee llevar a cabo acciones más complejas con dicha información.

En la aplicación móvil dicha información tiene como objetivo el activar a los botones de que tienen la leyenda "accion", con el objetivo de que el usuario pueda activar o desactivar al dispositivo deseado al "presionar" la imagen del botón "accion" que se encuentra en frente del nombre del dispositivo. Por lo que la información contenida en la respuesta no puede ser visualizada tan directamente en la aplicación móvil como en los otros servicios.

En la captura de pantalla de la aplicación móvil (figura 6.11) se pueden observar tres dispositivos actuadores llamados *"Fan1"*, *"Lum1"* y *"aire acondicionado"*; de igual modo debajo del cada nombre de dispositivo se muestra el estado que tiene, es decir, si se encuentra activo o inactivo.

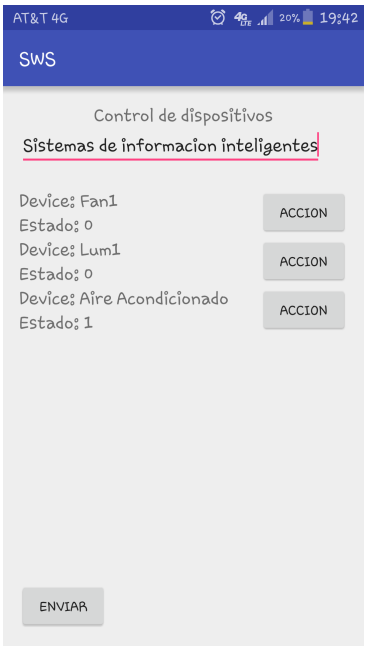


Figura 6.11: Control de actuadores a través de android

Para la prueba del último servicio *(véase figura 6.12 "a)" )* el cual recibe como parámetros las preferencias del usuario por medio de las etiquetas siguientes: *"<sel>"* se utiliza para que el usuario ingrese el modo de operación del programa monitor o demonio que ya se ha explicado anteriormente, la etiqueta "<temp>" la cual contiene la temperatura que es "ideal" para el usuario, en *"<hum>"* se encuentra la humedad que el usuario desea que se mantenga en el lugar, En *"<lux>"* el usuario ingresa la "cantidad de luz" que el desea en el interior del lugar y por último en *"<place>"* el usuario debe ingresar el nombre del lugar que el desea controlar y al cual se aplicaran los valores anteriores.

Como respuesta a la petición que hace el cliente que ha ingresado sus preferencias al sistema por medio del servicio, el usuario obtiene las mismas preferencias como una confirmación de que el servicio ha escrito los valores de las preferencias de manera adecuada.

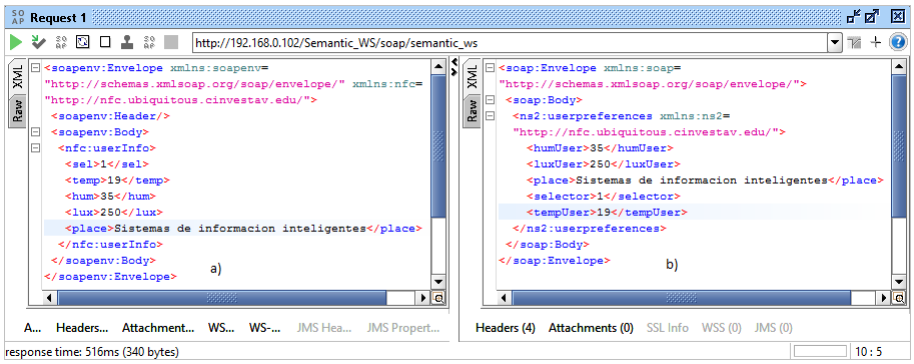


Figura 6.12: Envío de preferencias desde soap UI

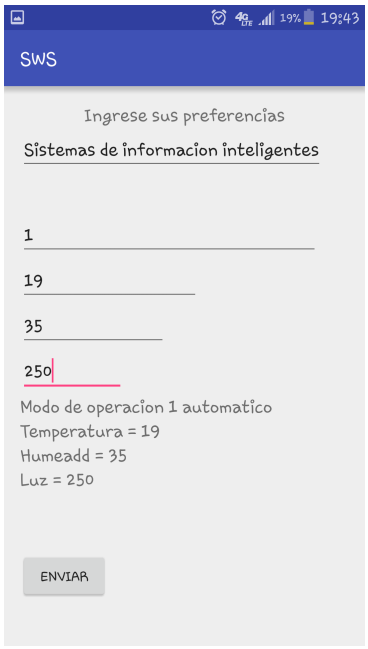


Figura 6.13: Envío de preferencias desde android

Al levar a cabo el ingreso de las mismas preferencias por medio de la aplicación móvil se pude observar en la figura 6.13 que posee los los mismos valores que en la figura 6.12 "b)" y de la misma forma debajo de los campos en los que el usuario ingresa sus preferencias se presenta la confirmación del

servicio al presentar de forma mś detallada los valores que el usuario ingreso.

### 6.2.2. Escenario con el sistema completo

Para el escenario de pruebas se propone una simulación de una persona de nuevo ingreso que desea conocer su ubicación dentro de la institución y desea buscar a un compañero con el que tiene una cita. Y para el caso de un profesor que se encuentra dentro de la institución y que desea buscar a algún colega y conocer la variables ambientales del lugar en que se encuentra, para enviar sus preferencias al sistema por medio del servicio.

Para las pruebas se usaron dispositivos de corriente alterna y directa como focos para incrementar la cantidad de luz del lugar, par el caso del ventilador de utiliza uno que es de corriente directa y para el caso del aire acondicionado se utiliza un compresor pequeño que simula la activación del aire acondicionado. Hay que tomar en cuenta que estas simulaciones se llevaron a cabo fuera del centro de investigación, ya que el conectar las etapas de potencia a los dispositivos conlleva desarmar a los dispositivos o tomas de corriente en algunos casos y se deben pedir los permisos adecuados a las autoridades pertinentes para llevar a cabo dicha acción. por lo que, el lugar de pruebas fue la residencia del estudiante.

En la figura 6.14 se presenta una etapa de potencia de estado sólido, la cual es accionada por medio del arduino que se encuentra en la imagen. Este arduino es el encargado de controlar a los dispositivos llamados *Lum1* y *Fan1* los cuales van a simular a un conjunto de lámparas que pueden ser accionadas de modo manual por el usuario o por el programa demonio al enviar los comandos necesarios al arduino que las controla y un ventilador de corriente alterna que cuenta con su etapa de potencia.

La conexión se ve realizada al conectar la etapa de potencia de la lámpara al arduino correspondiente y a la red electrica por medio del apagador que utiliza. Cabe destacar que la conexion se realiza en paralelo para seguir utilizando el apagador manual que posee la lampara, en la figura 6.15 se presenta como queda la conexión del arduino y la etapa de potencia a la lámpara.

En seguida se presentan los ventiladores que van a simular al de corriente alterna, en la figura 6.16 se muestra un arreglo de seis ventiladores de que

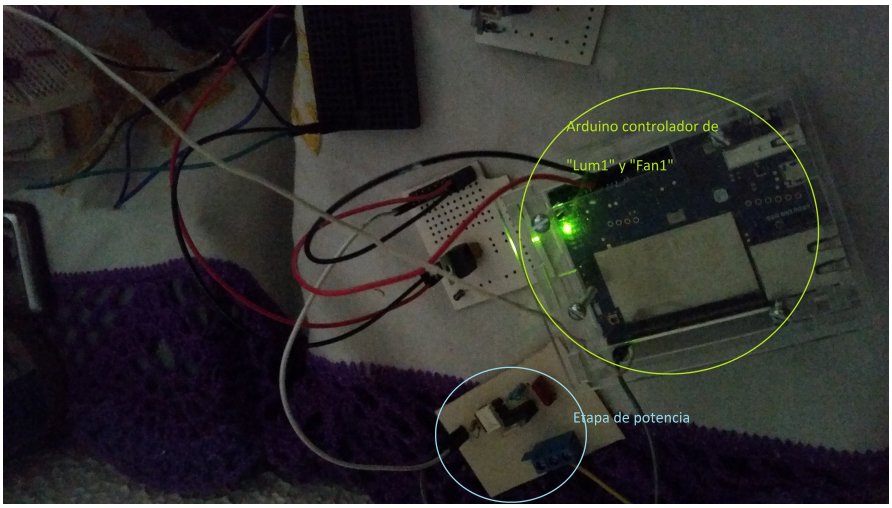


Figura 6.14: Etapa de potencia y control con arduino

trabajan con doce volts de corriente directa, de modo que es necesaria una etapa de potencia para poder suministrar el voltaje adecuado y la corriente necesaria para poder encender los ventiladores. Sólo se encienden dos de los seis ventiladores ya que la etapa de potencia se conecto para controlar solo a dos.

La etapa de potencia que controla a los ventiladores es un dispositivo conocido como puente 'H' que tiene por características la capacidad de activar o desactivar hasta cuatro dispositivos con un consumo eléctrico de hasta 36 volts de corriente directa y hasta 2 amperes. En la figura 6.17 se presenta el dispositivo LM293D[1], que también es capaz de llevar a cabo más acciones al conectar los dispositivos de acuerdo a las necesidades del usuario.

El siguiente dispositivo es la etapa de potencia que acciona a un compresor, esta etapa tiene un diseño a nivel componente igual a la primera presentada, con la diferencia de que en esta se agrega un disipador de calor de gran tamaño con el objetivo de proteger del sobrecalentamiento al transistor *Triac*. En la figura 6.18 se muestra una toma de la etapa de potencia que ya cuenta con suministro eléctrico y además cuenta con una clavija para conectar a distintos dispositivos.

[1]www.ti.com/lit/ds/symlink/l293.pdf última vez consultado 2-11-16

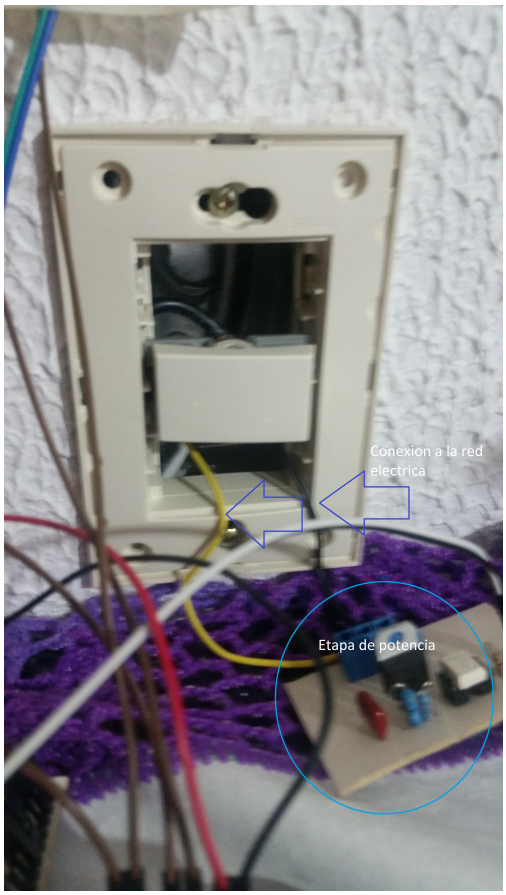


Figura 6.15: Lampara de corriente alterna

El compresor que se uso para la simulación cuenta con un consumo de alrededor de 150 Watts de potencia y la etapa de potencia es capaz de soportar actuadores con consumos de hasta 1000 Watts. En la figura 6.19 se presenta el compresor y la etapa de potencia ya conectados y también se puede observar un arduino que se encuentra conectado por medio de un cable USB a una batería. Dicho arduino es el encargado de activar a la etapa de potencia que activa al compresor.

Por último se muestra en la figura 6.20 un arduino mega el cual es el encargado de llevar a cabo las lecturas de los sensores de luz, temperatura y humedad, por lo que también cuenta con un módulo de conexión WiFi.

Figura 6.16: ventiladores de corriente directa

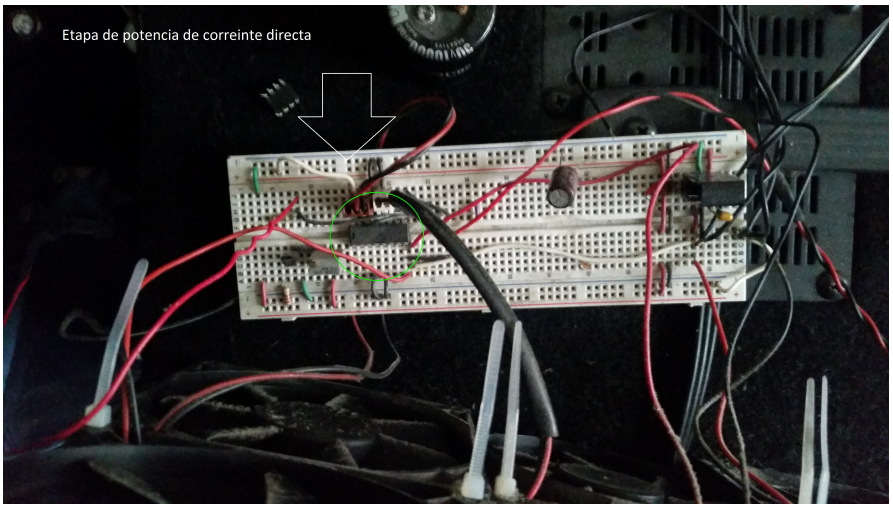


Figura 6.17: Etapa de potencia de corriente directa

Este arduino se utiliza para obtener los datos del ambiente.

Cabe destacar que cada uno de los arduinos cuenta con una dirección IP asignada por el servicio DHCP del *switch* de red inalámbrico, también


Figura 6.18: Etapa de potencia para motor de corriente alterna

para la simulación se obtuvieron las IP de cada arduino y se asignaron a su respectivo actuador en la meta-ontología. Las siguientes imágenes son capturas de pantalla del programa demonio y de cómo se va actualizando la información dentro de la meta-ontología.

## 6.3. Pruebas de modificación de preferencias

De primera instancia se presenta una toma de la ejecución del programa monitor, cuando el usuario ha ingresado el modo de operación manual, el

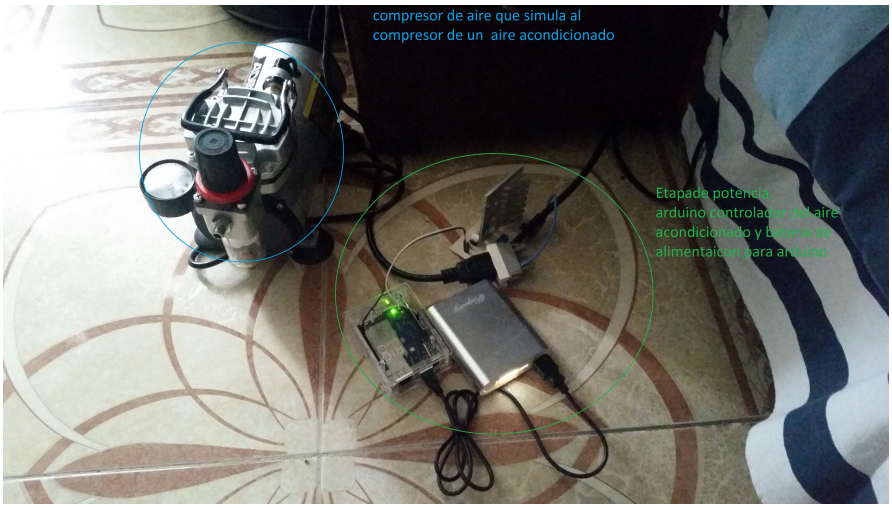


Figura 6.19: Compresor de aire

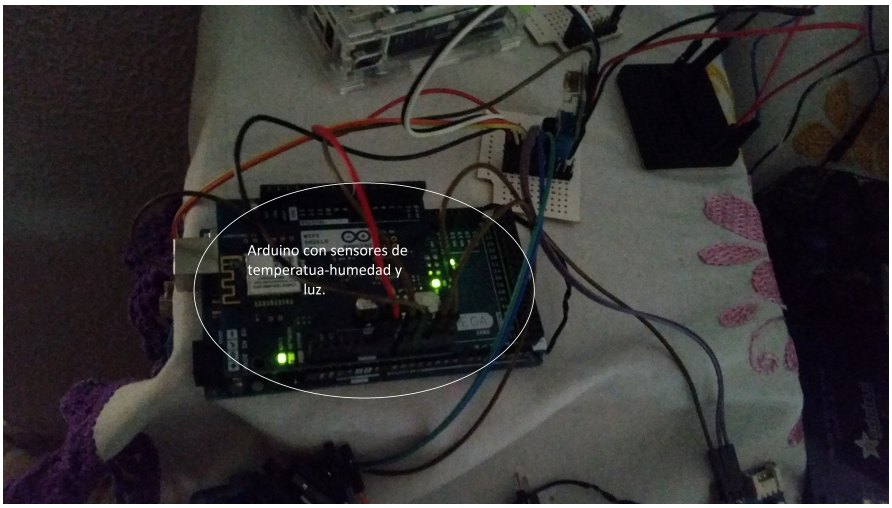


Figura 6.20: Arduino para lectura de sensores

programa monitor se cicla en esperar a que cambie la condición de entrada de una persona al lugar que se va monitorizar o que el usuario ingrese el modo de operación automático, el cual es un modo de operación forzado, ya que no importa que no haya ninguna persona dentro del lugar el sistema va

a mantener las condiciones ingresadas por el usuario.

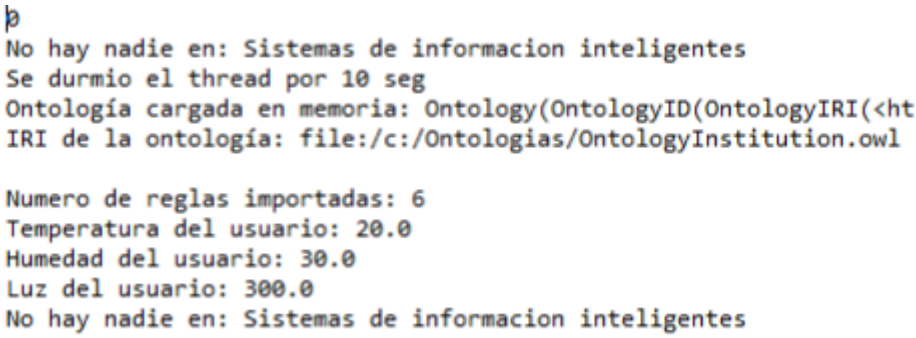


Figura 6.21: Modo de operacion manual

Al observar la captura de pantalla (figura 6.21) se puede observar que el programa monitor presenta información acerca del número de personas que se encuentran en ese lugar y de las preferencias ambientales del usuario como son: temperatura, humedad y luz. También se presenta el nombre del lugar que es controlado por el programa monitor o demonio. Los valores de las variables ambientales al momento de ejecutar el programa son las que se obtienen del archivo de preferencias del usuario y no se presentan las variables ambientales hasta el momento que se detecta a una persona dentro del lugar o que se cambie el modo de operación del programa.

Un ejemplo de ingreso se preferencias se presenta en la imagen 6.22, en la que se presenta el modo de operación manual del programa demonio que es indicado por el valor en 'cero'. La temperatura preferida del usuario es el valor de '25', la humedad es el valor correspondiente a '30' y la cantidad de luz mínima que desea el usuario corresponde a '300'. Además el lugar que se desea controlar es el denominado "Sistemas de información inteligentes", el cual ya se ha modelado en la ontología con sus respectivos sensores y actuadores.

Al ejecutar al programa demonio, en modo de operación automático se puede observar que hace una consulta a la meta-ontolgía y obtiene la información necesaria acerca de los actuadores, variables ambientales, número de personas y las preferencias del usuario con el fin de llevar a cabo las acciones pertinentes en los dispositivos actuadores correctos, cuando se cumple una

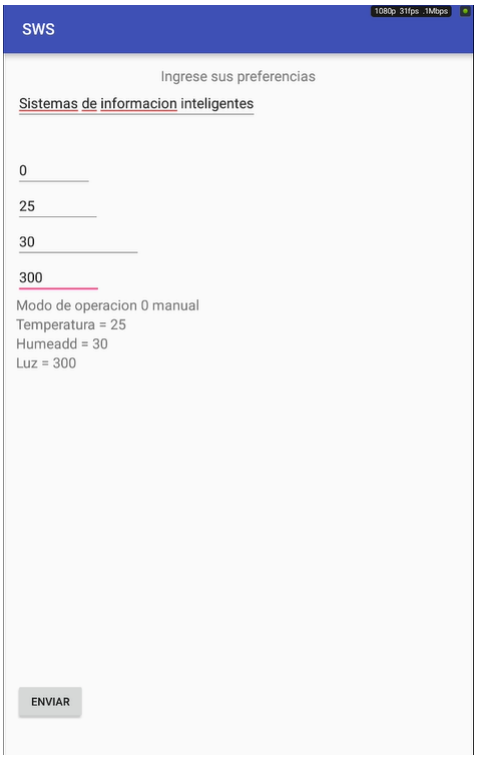


Figura 6.22: Ingreso de preferencias

condición.

En la figura 6.23 se puede observar que la información obtenida por el programa demonio presenta en consola las direcciones IP de tres dispositivos actuadores que se encuentran en el lugar que se está censando continuamente y cuyo ambiente se desea controlar. Además se presentan las preferencias del usuario, las cuales han sido ingresadas previamente, y también se muestra el modo de operación seleccionado por el usuario junto con las variables ambientales del lugar.

Como se mencionó antes, el usuario con "privilegios" puede controlar el ambiente de su lugar de trabajo ingresando los siguientes valores: temperatura de 19 °, humedad del 35 % y cantidad de luz mínima de 250, con un modo de operación automático. Para llevar a cabo la prueba de cambio en la temperatura se modifica el valor de uno de los sensores de temperatura, en

```
192.168.0.148
0
Lum1
--------------------
192.168.0.148
0
Fan1
--------------------
192.168.0.149
0
Aire Acondicionado
--------------------
Numero de personas en el lugar: 0
--------------------
Temperatura del lugar: 19.0
Humedad del lugar: 30.0
Luz del lugar: 240.0
----------------------
Modo de operacion: 1
Automatico
Temperatura del usuario: 19.0
Humedad del usuario: 35.0
Luz del usuario: 250.0
Lugar: Sistemas de informacion inteligentes
```

Figura 6.23: Valores Iniciales

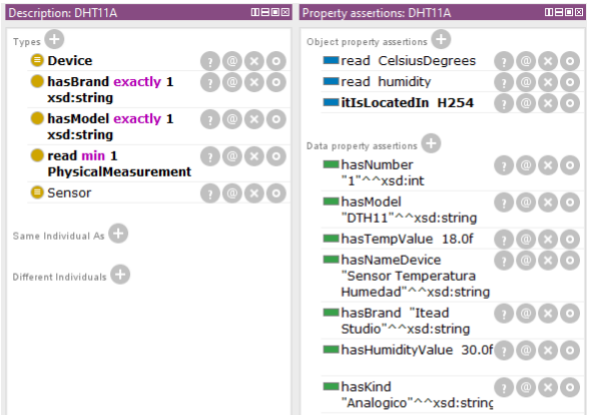


Figura 6.24: temperatura y humedad iniciales

la figura 6.24 presenta la codificación del valor de temperatura que originalmente era de 18°a 20°. Cabe recordar que se tiene un arreglo de dos sensores de temperatura-humedad de modo que solo se modifica el parámetro de uno y el otro sensor mantiene su lectura de 20°, el valor de humedad en este sensor no se ve afectado.

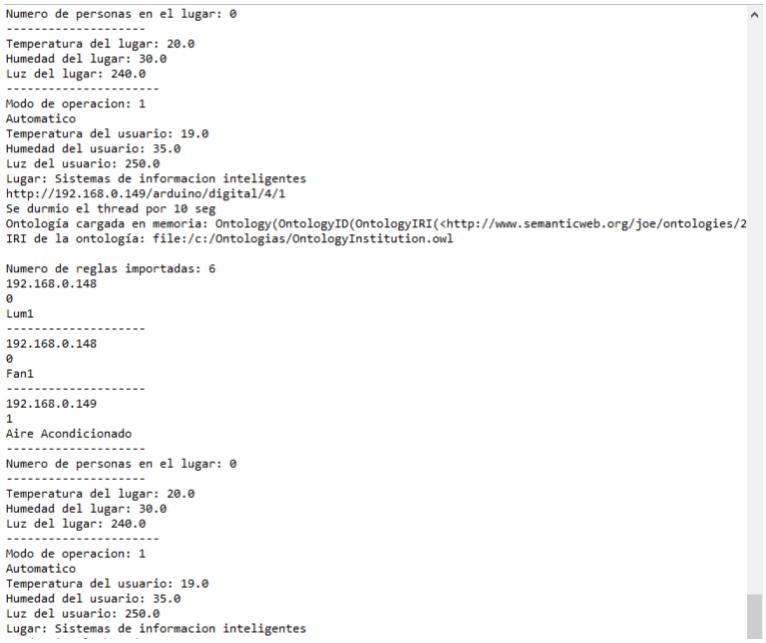


Figura 6.25: Incremento del valor de temperatura

Como se ha mencionado antes para obtener la temperatura se lleva a cabo la operación de obtener el promedio de temperaturas, con base en el número de sensores asociado al espacio físico que se está consultando. De modo que al obtener el promedio de temperatura del lugar cuyas lecturas de temperatura son de 20°en ambos sensores, se obtiene una temperatura promedio de 20°, y como la preferida el usuario es 19°el programa demonio enciende al aire acondicionado para reducir la temperatura del lugar. La activación se lleva a cabo al enviar el comando al arduino de la figura 6.19 que controla la etapa de potencia que activa al aire acondicionado. El comando se envía al construir una url compuesta de la IP que tiene el aire acondicionado y su nombre.

Como se puede observar en la figura 6.25 se muestra que la temperatura del lugar ha subido de 19°a 20°, por lo que más abajo se presenta el cambio de estado del dispositivo aire acondicionado cuyo estado en la figura 6.23 es 'cero' y en a figura 6.25 es de 'uno', lo que indica que se ha encendido.

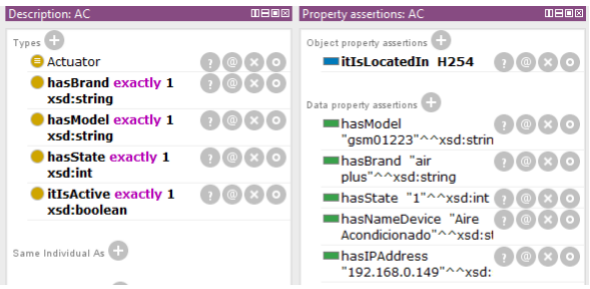


Figura 6.26: Cambio de estado del aire acondicionado

El cambio se ve reflejado en la meta-ontología *(véase figura 6.26)*, en el modelo del aire acondicionado en la meta-ontología se puede observar que la propiedad *"hasState"* tiene como valor un 'uno'. Lo que le indica al programa demonio que el dispositivo actuador está ahora activado y así evitar seguir enviando varias veces el comando de encendido.

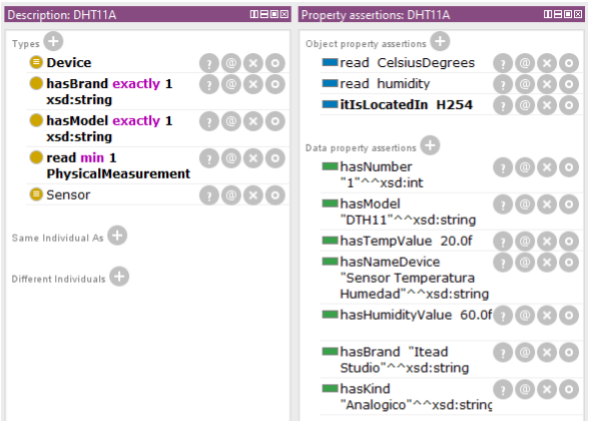


Figura 6.27: Incremento del valor de humedad

Del mismo modo que se cambió el valor de temperatura de uno de los sensores, se puede cambiar el valor de la humedad en el mismo sensor. En este ejemplo el valor sera el doble del que se tenia, por ejemplo. En la figura 6.23 la propiedad *"hasHumidityValue"* tiene un valor de 30 y, al hacer el cambio, en la figura 6.27 se puede observar que el nuevo valor es de 60. hay

que recordar que esos valores son en porcentaje ya que el valor que devuelve el censor es directamente el porcentaje de humedad que se ha censado.

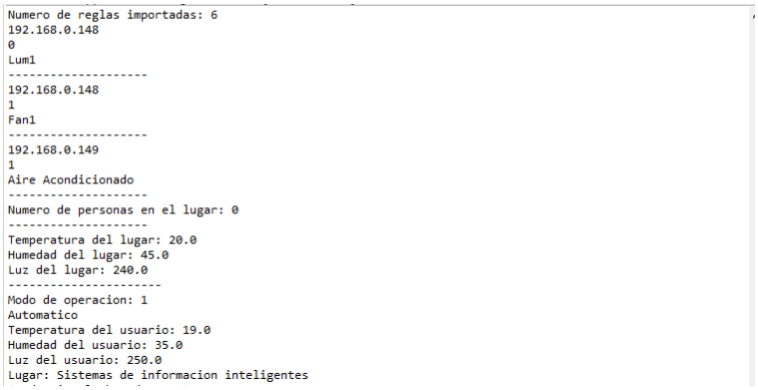


Figura 6.28: Encendido de *Fan1*

El programa demonio, al obtener el nuevo valor promedio de humedad, lleva a cabo la acción de encender el dispositivo *"Fan1"* que es el encargado de "reducir" la humedad ambiental. En la figura 6.28 se presenta la captura de pantalla del programa demonio en consola, en dónde se puede observar el estado de *"Fan1"* ha cambiado de cero a uno, lo que indica que está ahora activo.

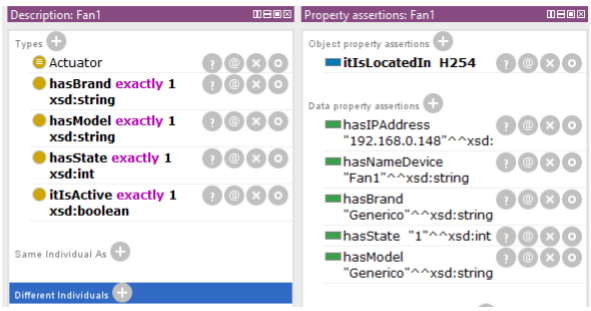


Figura 6.29: Cambio de estado de *Fan1*

El cambio de estado del dispositivo *"Fan1"* se ve reflejado en la terminal del programa demonio, y también se presenta en la figura 6.29 que representa

el modelo del dispositivo *"Fan1"* en la ontología, como se puede observar la propiedad *"hasState"* tiene un valor de uno, lo que indica que el estado del dispositivo es activo o encendido.

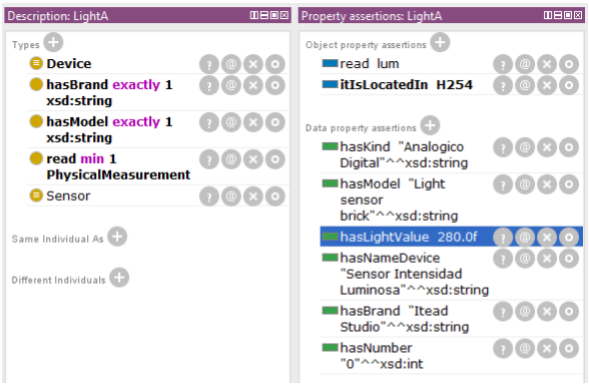


Figura 6.30: Incremento del valor de luz

Por último y como se ha llevado a cabo anteriormente para simular un cambio rápido en la cantidad de luz que se está censando, se procede a cambiar el valor de uno de los censores de luz, para este caso se cambia el valor del sensor llamado *"LightA"* el cual se encuentra relacionado con el lugar que se está simulando por medio de la propiedad *"itIsLocatedIn"*. Por consiguiente el valor original luz que se tenia era de 180 y se ha cambiado por 280, como se puede observar en la figura 6.30. Al incrementar el nivel de "obscuridad" el programa demonio enciende las luces para alcanzar un nivel de luz igual o mayor al que el usuario prefiere.

Al llevar a cabo el cambio en el valor de la luz el programa demonio presenta el nuevo valor promedio obtenido desde la meta-ontología, de modo que al compararlo con el de la figura 6.28 que es "Luz del lugar:" con un valor de 240 con el presentado en la figura 6.31 se puede ver que ha cambiado a 290. Por lo que al cumplirse la condición de que la cantidad de luz censada es menor que la preferida por el usuario se debe encender la luz para cumplir con las preferencias del usuario.

De igual manera el cambio de estado se debe ver reflejado en la meta-ontología, de modo que al observar al modelo de la lámpara llamado *"Lum1"* en su propiedad *"hasState"* se ha cambiado por un 'uno' que también puede

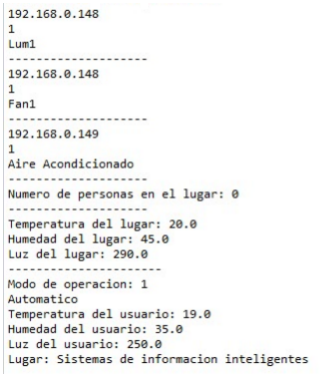


Figura 6.31: Encendido de lamparas

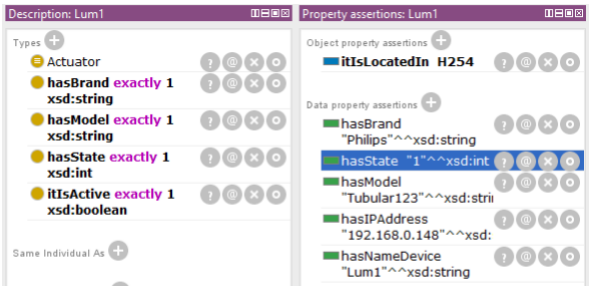


Figura 6.32: Cambio de estado de *Lun1*

ser observado en la captura de pantalla de la terminal del programa monitor en la figura 6.31.

Las capturas de pantalla anteriores son las que se han obtenido en la prueba del ingreso de preferencias por parte de un usuario que tiene el privilegio de "controlar" su ambiente de trabajo a través de Internet por medio del consumo de uno de los servicios Web.

## 6.4. Pruebas para ubicación de usuarios

En esta sección se presentan pruebas a los distintos servicios al simular situaciones de la vida real que se pueden presentar en el sistema, de primera

instancia se presenta un escenario en que un usuario sin privilegios desea saber en donde se encuentra y desea buscar a una persona. Posteriormente el siguiente escenario es el de un usuario con privilegios que desea controlar su entorno de forma manual, además de localizarse a si mismo y buscar a algún conocido.

### 6.4.1. Escenario de usuario sin privilegios

Tomando en cuenta que ya se han ingresado preferencias ahora el usuario desea saber su ubicación dentro de la institución, para saber exactamente en qué lugar se encuentra. De modo que al iniciar su aplicación móvil ingresa a la sección de búsqueda e ingresa su nombre.


Figura 6.33: Búsqueda de persona

Una vez que ingresa su nombre en el cuadro te texto y "presiona" el botón

con la leyenda "buscar" la aplicación móvil envía el nombre que ingresó el usuario y espera una respuesta del servicio Web que se está consumiendo. Una vez trascurrido el tiempo de espera se presenta en pantalla la ubicación del usuario *(véase figura 6.33)* que en este caso se encuentra en un lugar llamando *"Babage"* que es parte del edificio "J", se encuentra en el segundo nivel y tiene asignado el número "02".

Como se puede observar el servicio consumido es *"search"* cuyo parámetro te entrada es el nombre de la persona que se desea buscar, a diferencia del servicio *"location"* que devuelve mayor cantidad de información, el servicio *"search"* solo responde a la petición con los datos directos de la ubicación sin contemplar temperatura, humedad y luz.

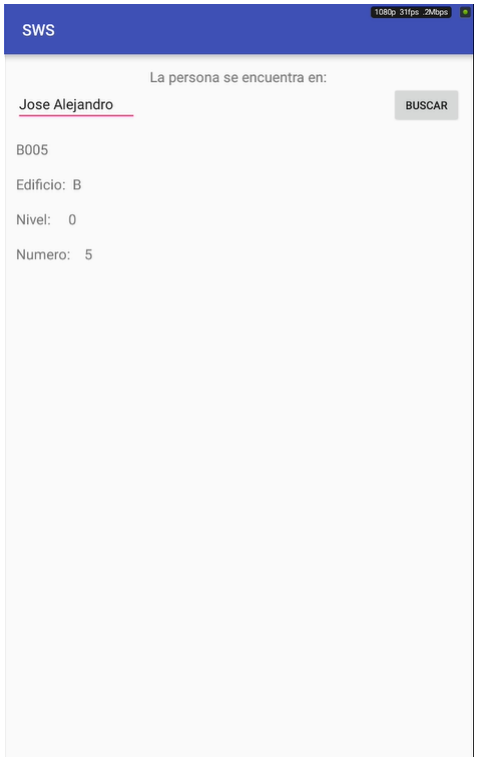


Figura 6.34: Búsqueda de colega

Además el usuario puede localizar a un colega con el que trabaja, por tanto decide buscar a su colega haciendo uso de la aplicación de búsqueda en la que debe ingresar el nombre de su colega y esperar la respuesta del servicio. Como se puede observar en la figura 6.34 el servicio devuelve la ubicación del colega, quien se encuentra en un lugar llamado "B005" que se ubica en el edificio "B" planta baja y es el quinto espacio físico que compone al edificio "B".


Figura 6.35: Control manual desde android

### 6.4.2. Escenario usuario con privilegios

Como se presentó en la figura 6.22 el usuario ha ingresado previamente sus preferencias haciendo uso de la aplicación móvil, de modo que al terminar las actividades que debe realizar fuera de su laboratorio u oficina, desea

tomar el control manual de los dispositivos actuadores que se encuentran en su laboratorio u oficina. Por tanto decide hacer uso de la aplicación móvil que le permite obtener la información de los dispositivos que se encuentran en el lugar que el ingrese.

La figura 6.35 presenta el nombre del lugar que el usuario desea controlar en modo manual. Al ingresar el nombre del lugar que desea controlar manualmente en el cuatro de texto y "presionar" el botón con la leyenda "enviar" la aplicación envía la petición al servicio *"actuator"*, el cual devuelve los dispositivos que se encuentran en dicho lugar y también sus estados actuales. Los botones de acción se activan una vez recibida la información proveniente del servicio Web. El usuario será capaz de activar o desactivar los dispositivos por medio de la aplicación móvil al "presionar" el botón que se encuentra en frente de cada uno de los nombres de los dispositivos y por lo tanto la aplicación será capaz de enviar el comando adecuado para activar o desactivar al dispositivo que el usuario desee.

Un **usuario con privilegios** puede hacer uso de todas las aplicaciones si problemas ni restricciones de modo que al ser un **visitante destacado** de la institución, desea saber en dónde se encuentra ya que desconoce la distribución de la institución. El visitante *privilegiado* puede utilizar la aplicación móvil de localización, para saber en dónde de encuentra y que variables ambientales existen en dicho, lugar ya que le han comentado que puede controlar a algunos dispositivos del lugar en el que se encuentra.

Por tanto en la figura 6.36 se presenta la información que devuelve el servicio una vez que el usuario ha ingresado su nombre en el cuadro de texto y ha hecho "click" en el botón ubicar. El servicio responde con la localización del usuario en el último lugar en que fue "censado" por medio de la tarjeta de RFID, que se le asigno al momento de llegar a la institución. por tanto, siempre debe tener consigo dicha tarjeta de RFID ya que es indispensable para su localización. Además la información que recibe al hacer uso de la aplicación móvil de localización es más completa ya que incluye los promedios de temperatura, humedad y cantidade de luz.

También debe localizar a un copañero que también es visitante en la institución con el cual debe trabajar. Dicho compañero se encuentra tomando un curso en uno de los salones de clases, por lo que el usuario debe hacer uso de la aplicación móvil para localizar a su compañero para ir a buscarlo.

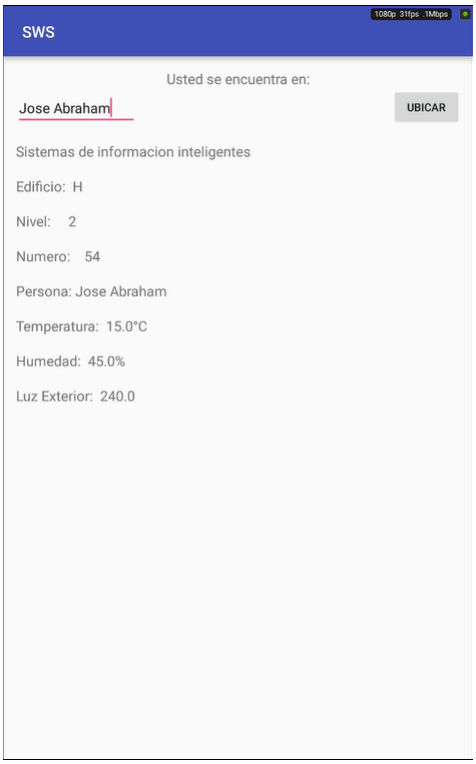


Figura 6.36: Buscando mi ubicación

En la figura 6.37 el usuario ha ingresado el nombre completo del compañero que desea localizar y el servicio le devuelve como respuesta que la persona que busca se encuentra en un lugar llamado "E212" el cual se encuentra en el edificio "E" segundo piso y es el doceavo lugar que se encuentra en ése nivel.

El usuario **privilegiado** puede hacer uso del todos los servicios, es decir, los puede consumir desde la aplicación móvil sin restricciones. El privilegio es otorgado directamente por la la institución a quien sea necesario darle dicha capacidad, los usuarios sin privilegios solo pueden hacer uso del servicio de localización y búsqueda. A diferencia del usuario privilegiado que puede hacer uso de esos dos además de los servicios de control manual de los dispositivos actuadores y del ingreso de preferencias.

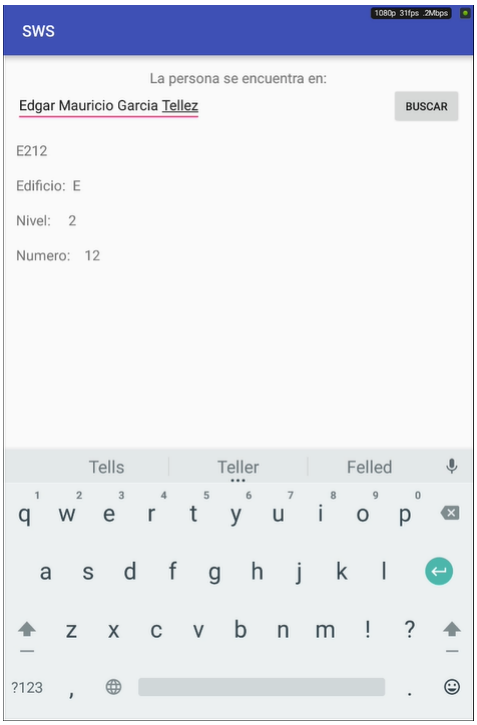


Figura 6.37: Buscando a compañero

Las simulaciones anteriores comprenden una pequeña demostración de la capacidad del sistema que se ha diseñado para este trabajo. Cabe destacar que las tomas que se presentan se llevaron a cabo con distintos dispositivos, de modo que sea evidente la amplia aplicación que puede tener el sistema. Además el poblar la meta-ontología conlleva demasiado tiempo, por lo que no fue posible agregar más usuarios para las pruebas ya que no se cuenta con suficientes sensores ni actuadores con los que se pueda establecer mayor interacción, hasta el momento la meta-ontología cuenta con una variedad de personas, dispositivos y espacios físicos que pueden ser consultados.

# Capítulo 7

# Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones obtenidas por cada objetivo específico del proyecto, del objetivo general y de la hipótesis. También se presenta el trabajo a futuro que puede hacer crecer ya sea en mejoras o agregar nuevas funcionalidades al presente proyecto. Por consiguiente el presente capítulo se va a dividir en dos secciones la primera en la que se presentan conclusiones y la segunda que habla acerca del trabajo futuro.

## 7.1. Conclusiones

Las conclusiones se presentarán en el siguiente orden, primero se verán las conclusiones del los objetivos específicos y después del objetivo general seguida de una conclusión para la hipótesis.

- Es posible diseñar e implementar una o varias redes de sensores, al hacer uso de tecnologías abiertas *Open source* que reducen los costos de los dispositivos de *hardware* reconfigurable, además permiten crear prototipos en hardware de forma rápida y fácil de utilizar, ya que poseen gran variedad de aditamentos los cuales agregan funcionalidades, entre ellas la posibilidad de comunicación a través de *ethernet* o Wi-Fi. Las redes de sensores que se pueden implementar haciendo uso de dicho *hardware* con la ventaja de pueden ser fácilmente desplegadas gracias al bajo consumo energético que poseen estos dispositivos.

  Los datos que se pueden obtener del contexto por medio de sensores

pueden ser desde muy simples como la temperatura, hasta muy complejos como el censar si una persona entra o sale de un lugar. Al tener acceso a distintos sensores y definir un contexto es posible censar datos que se pueden usar para crear información más compleja y completa acerca del ambiente en que se encuentra desplegada la red de sensores.

- Gracias al uso de las ontologías es posible modelar a casi cualquier objeto ya sea físico o conceptual haciendo uso de las características del mismo, es decir, se pueden agregar propiedades las cuales permiten crear un modelo relacional con otros objetos dentro de la misma ontología. Además las ontolgías tienen la capacidad de cargar ontologías adicionales en forma de módulos, lo que permite la ampliación de las características y por ende brindan la posibilidad de crear ontologías más grandes y complejas.

  Al hacer uso de ontologías más complejas, llamadas meta-ontologías, que están compuestas de dos o más ontologías se abre el camino a crear sistemas más complejos de conocimiento. La desventaja de tener ontologías demasiado grandes es que poblarla se vuelve una tarea demasiado complicada y pesada. Además la explotación de otoloías demasiado complejas y con demasiadas relaciones vuelve más lento el proceso de explotación por medio de la API *OWLAPI*.

- Las tecnologías Web permiten crear servicios capaces de ejecutar consultas a las ontologías por medio de las API's adecuadas. Las plataformas web juegan un papel importante en el desarrollo de este trabajo, ya que permiten recibir y enviar información de manera adecuada por medio de los protocolos estandarizados como *SOAP o REST*, ofreciendoo así un ambiente más amigable y fácil de consumir por otras plataformas que hagan uso de dichos estándares.

- Los dispositivos móviles en este caso los *smart phones* se han vuelto parte de la vida cotidiana del ser humano por lo que el desarrollar aplicaciones para esos dispositivos móviles se vuelve una tarea sencilla. En la actualidad tienen mucho potencial ya que cuentan con una gran variedad de censores y actuadores integrados sin olvidar su capacidad de comunicación por medio de distintos protocolos estandarizados como el IEEE 802.11, *bluetooth* entre otros. Por lo que se han vuelto una

herramienta indispensable en áreas como el cómputo ubicuo

Además, al consumir los servicios que se diseñaron para este trabajo se demuestra que la capacidad de enviar y recibir datos por medio de estos dispositivos es infinita ya que cuentan con suficiente poder de cómputo para procesar la información y presentarla al usuario haciendo uso de gráficos que no demandan demasiada atención del usuario, permitiendo la usuario elegir cuando ejecutar una tarea.

### 7.1.1. Conclusión del objetivo general

Ciertamente al obtener suficiente información del ambiente por medio de sensores y al delimitar un contexto es posible definir la información que puede ser relevante para los usuarios. Al definir la información que puede contener el perfil general de los usuarios se abre el camino a la posibilidad de reducir la interacción que tiene físicamente el usuario con los dispositivos se encuentran en su entrono.

El obtener datos del ambiente por medio de censores y al agregar un poco se semántica a dichos datos abre la posibilidad de crear sistemas con un cierto nivel de inteligencia, ya que se combinan distintas tecnologías como la Web y los dispositivos móviles para crear un ambiente ubicuo que no demanda demasiada atención por parte del usuario, cumpliendo así el objetivo de que se esconda en el trasfondo de la vida diaria del ser humano.

### 7.1.2. Conclusión de la hipótesis

Es posible adquirir datos del ambiente y transformarlos en información útil, que puede ser fácilmente entendida por el usuario, al presentar dicha información por medio de una interfaz amigable para el mismo. Un ambiente puede contener a varios contextos, en el que existen distintos usuarios que tienen interacción unos con otros, y la información en ocasiones es confusa de tal modo que al agregar un poco de semántica y presentar la información de forma fácil de entender, el usuario se vuelve el beneficiario de un sistema que se encarga de entregarle información adecuada en un medio que no requiere la total atención del usuario.

## 7.2. Trabajo futuro

Las propuestas de trabajo a futuro que aquí se presentan tienen como objetivo ampliar el alcance el presente trabajo, ya sea mejorando las características o las tecnologías utilizadas para el presente desarrollo. A continuación se abordan las propuestas.

- Se deben probar distintas tecnología de comunicación de sensores para crear redes de sensores remotas que utilicen distintos protocolos de comunicación.
- Desde el punto de vista de los sensores, se deben desarrollar nuevos sensores, además se pueden utilizar los sensores que tiene el *smartphone* con el objetivo de obtener suficientes datos para reconocer al usuario y poder hacer uso de los mismos para que el usuario tenga una mejor interacción con el sistema.
- Hay que contemplar la ampliación de la meta-ontología, con nuevas características que se pueden agregar a las ontoloías que la componen, e incluso la posibilidad de agregar nuevas ontologías para incrementar el número de servicios que se pueden ofrecer.
- Incrementar el número de servicios disponibles a través de distintas plataformas Web y con distintos protocolos web. Se debe tomar en cuenta la posibilidad de transformar a los servicios actuales en servicios web semánticos.
- La aplicación móvil actual fue diseñada para el sistema operativo móvil android y debe ser ampliada a otros sistemas operativos móviles.
- Las características actuales de la aplicación móvil no contemplan el uso de notificaciones, ni el intercambio directo de información con el sistema. Debe ampliarse la aplicación móvil para mejorar su aspecto visual y funcionalidades contemplando la aplicación de nuevas características.

El prototipo presentado en este trabajo tiene muchas áreas de oportunidad por lo que se debe dejar volar la imaginación para ampliar su funcionalidad y aplicación.

# Bibliografía


[1] M. Weisser, "The Computer for the 21st Century," *Sci Am*, vol. 265, no. 3, pp. 94–104, 1988.

[2] L. Nachabe, M. Girod-Genet, and B. El Hassan, "Unified Data Model for Wireless Sensor Network," *IEEE Sensors Journal*, vol. 15, no. 7, pp. 3657–3667, 2015.

[3] T. Kindberg and A. Fox, "System Software for Ubiquitous Computing," *Pervasive Computing*, vol. 1, no. 1, pp. 70–81, 2002.

[4] A. P. Sage and C. D. Cuppan, "On the Systems Engineering and Management of Systems of Systems and Federations of Systems," *Inf. Knowl. Syst. Manag.*, vol. 2, no. 4, pp. 325–345, 2001.

[5] S. A. McIlraith, T. C. San, and H. Zeng, "Semantic Web services," *IEEE Intelligent Systems*, vol. 16, pp. 46–53, 2001.

[6] A. Pintus, D. Carboni, and A. Piras, "Paraimpu," *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, p. 401, 2012.

[7] B. Christophe, "Managing massive data of the internet of things through cooperative semantic nodes," in *Proceedings - IEEE 6th International Conference on Semantic Computing, ICSC 2012*, 2012, pp. 93–100.

[8] M. Satyanarayanan, "Pervasive computing: Vision and challenges," pp. 10–17, 2001.

[9] R. Masuoka, B. Parsia, and Y. Labrou, "Task computing–the semantic web meets pervasive computing," *The Semantic Web-ISWC 2003*, pp. 866–881, 2003.

[10] J. E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, and D. Mosse, "Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes," *2012 Eighth International Conference on Intelligent Environments*, pp. 206–213, 2012.

[11] D. Cook, "MavHome: An agent-based smart home," *... 2003.(PerCom 2003 ...*, no. January, pp. 521–524, 2003.

[12] V. Peláez, R. González, L. Á. S. Martín, A. Campos, and V. Lobato, "Multilevel and Hybrid Architecture for Device Abstraction and Context Information Management in Smart Home Environments," in *First International Join Conference on Ambient Intelligence*, vol. 6439, 2010, pp. 207–216.

[13] J. Hendler, "Agents and the semantic web," *IEEE Intelligent Systems and Their Applications*, vol. 16, no. 2, pp. 30–37, 2001.

[14] G. Borriello and R. Want, "Embedded computation meets the World Wide Web," *Communications of the ACM*, vol. 43, no. 5, pp. 59–66, 2000.

[15] T. R. Payne and O. Lassila, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2004.

[16] R. Cáceres, A. Friday, R. Caceres, and A. Friday, "Ubicomp Systems at 20: Progress, Opportunities, and Challenges," *IEEE Pervasive Computing*, vol. 11, no. 1, pp. 14–21, 2012.

[17] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture," *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pp. 1361–1367, 2013.

[18] W. Qin, Q. Li, L. Sun, H. Zhu, and Y. Liu, "RestThing: A restful web service infrastructure for mash-up physical and web resources," in *Proceedings - 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing, EUC 2011*, 2011, pp. 197–204.

[19] H. H. Wang, N. Gibbins, T. Payne, and A. Patelli, "A survey of semantic web services formalisms," in *Proceedings - 2013 9th International Conference on Semantics, Knowledge and Grids, SKG 2013*, 2013, pp. 135–142.

[20] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web Service Description for the Semantic Web," pp. 348–363, 2002.

[21] R. Dumitru, K. Uwe, L. Holger, B. de Jos, L. Rubén, S. Michael, P. Axel, F. Cristina, B. Cristoph, and F. Dieter, "Web Service Modeling Ontology," *Applied Ontology*, vol. 1, no. 1, pp. 77–106, 2005.

[22] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic annotations for WSDL and XML schema," *IEEE Internet Computing*, vol. 11, no. 6, pp. 60–67, 2007.

[23] S. Tepic, P. Pejic, J. Domsic, H. Mihaldinec, and H. Dzapo, "IBMS - Intelligent Building Management System Framework," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 143–148.

[24] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp. 347–352, 2010.

[25] I. Bose and R. Pal, "Auto-ID: Managing Anything, Anywhere, Anytime in the Supply Chain," *BT Technology Journal*, vol. 22, no. 3, pp. 100–106, 2004.

[26] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[27] K. Park, Y. Kim, and J. Chang, "Semantic reasoning with contextual ontologies on sensor cloud environment," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[28] G. Moschoglou, T. Eveleigh, T. Holzer, and S. Sarkani, "A semantic mediation framework for architecting federated ubiquitous systems," *Proceedings - 2012 7th International Conference on System of Systems Engineering, SoSE 2012*, no. July, pp. 485–490, 2012.

[29] R. Yus, E. Mena, S. Ilarri, and A. Illarramendi, "Sherlock: Semantic management of location-based services in wireless environments," *Pervasive and Mobile Computing*, vol. 15, pp. 87–99, 2014.

[30] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mobile Networks and Applications*, vol. 12, no. 2-3, pp. 113–127, 2007.

[31] M. Hennessy, C. Oentojo, and S. Ray, "A framework and ontology for mobile sensor platforms in home health management," in *2013 1st International Workshop on the Engineering of Mobile-Enabled Systems, MOBS 2013 - Proceedings*, 2013, pp. 31–35.

[32] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. M. Leung, "Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 402–411, 2015.

[33] M. Beigl, H.-W. Gellersen, and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artefacts," *Computer Networks*, vol. 35, no. 4, pp. 401–409, 2001.

[34] M. Coronado, C. A. Iglesias, and E. Serrano, "Modelling rules for automating the Evented WEb by semantic technologies," *EXPERT SYSTEMS WITH APPLICATIONS*, vol. 42, no. 21, pp. 7979–7990, 2015.

[35] Y. Raimond and S. Abdallah, "The Event Ontology," *Event (London)*, pp. 3–6, 2011.

[36] L. Frye and L. Cheng, "A network management system for a heterogeneous, multi-tier network," *GLOBECOM - IEEE Global Telecommunications Conference*, pp. 1–5, 2010.

[37] P. S. Moraes, L. N. Sampaio, J. A. S. Monteiro, and M. Portnoi, "MonONTO a domain ontology for network monitoring and recommendation for advanced Internet applications users," in *2008 IEEE Network Operations and Management Symposium Workshops - NOMS 08*, 2008, pp. 116–123.

[38] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[39] R. Bendadouche, C. Roussey, G. De Sousa, J. P. Chanet, and K. M. Hou, "Extension of the Semantic Sensor Network ontology for Wireless Sensor Networks: The stimulus-WSNnode-communication pattern," in *CEUR Workshop Proceedings*, vol. 904, 2012, pp. 49–64.

[40] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, "The SSN ontology of the W3C semantic sensor network incubator group," *Journal of Web Semantics*, vol. 17, pp. 25–32, 2012.

[41] M. McCullough, *Digital Ground: Architecture, Pervasive Computing, and Environmental Knowing*, 2004.

[42] P. Atkinson, "Man in a briefcase: The social construction of the laptop computer and the emergence of a type form," *Journal of Design History*, vol. 18, no. 2, pp. 191–205, 2005.

[43] F. v. H. Deborah L. McGuinness, "Owl web ontology language overview," *W3C recommendation 10.2004-03*, vol. 2004, no. February, pp. 1–12, 2004.

[44] M. Rashid H, "Electrónica de Potencia. Circuitos, Dispositivos y Aplicaciones," *Electrónica de Potencia. Circuitos, Dispositivos y Aplicaciones*, pp. 1–878, 2004.

[45] L. Boylestad, Robert, "Introducción al análisis de circuitos," *México : Prentice-Hall, 2011.*, vol. decima seg, 2011.

[46] R. L. Boylestad and N. Louis, *Electronic devices and circuit theory*, 2013, vol. 1, no. 9.

[47] W. Bolton, *Mechatronics*. Pearson Education Limited, 2013.