SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

PROGRAMA DE MAESTRÍA EN CIENCIAS
DE LA COMPUTACIÓN

TESIS

TEMA:

Middleware basado en una arquitectura IoT para granjas urbanas

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

EDUARDO VIDRIOS NUÑEZ

DIRECTORES:

M.C. FELIPE ALFONSO ORDOÑEZ GARCÍA

DRA. MARÍA GUADALUPE SÁNCHEZ CERVANTES

DR. MARIO ÁNGEL SILLER GONZÁLEZ PICO

CD. GUZMÁN JALISCO, MÉXICO, AGOSTO DE 2018

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Cd. Guzmán

Cd. Guzmán, Jal. a 09/Agosto/2018

Oficio No. DEPI/62/18

ASUNTO : AUTORIZACIÓN DE IMPRESIÓN

C. EDUARDO VIDRIOS NUÑEZ
N.C. M16290037

En cumplimiento con el documento normativo de las disposiciones para la operación de estudios de posgrado del Tecnológico Nacional de México y con base en la aprobación del Comité Tutorial comisionado para su revisión; la División de Estudios de Posgrado e Investigación le otorga la autorización de impresión de su trabajo de tesis intitulado:

"MIDDLEWARE BASADO EN UNA ARQUITECTURA IOT PARA GRANJAS URBANAS"

dirigido por el **M.C. Felipe Ordoñez García**, desarrollado como requisito parcial para la obtención del grado de Maestro en Ciencias de la Computación, de acuerdo al plan de estudios MCCOM-2011-05.

Sin otro asunto en particular, quedo de usted.

A T E N T A M E N T E

S.E.P. TecNM
INSTITUTO TECNOLÓGICO
DE CD. GUZMAN
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACIÓN

DR. HUMBERTO BRACAMONTES DEL TORO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.p. Archivo

Agradecimientos

Agradezco a Dios y a toda mi familia por su apoyo incondicional en especial mis padres Martha Núñez García y Juan Manuel Vidrios por su apoyo incondicional y sacrificios a lo largo de toda mi vida como estudiante, a mi abuelito Arturo Núñez y Martha García que a pesar de que no verán este logro cumplido fueron un pilar importante en mi formación como persona y estudiante, a mi hermano Víctor Vidrios por brindarme su ayuda cuando la necesite, a mi hermano Juan Carlos por los momentos vividos.

A mis compañeros de Generación Vero, Jair, Diego Héctor, Charlie y Blanca por su ayuda durante la formación académica y porque más allá de ser compañeros hubo una amistad con buenos recuerdos.

A mi asesor el maestro Alfonso Ordoñez por su guía que me brindó para la realización de mi tema de investigación.

A todos los Maestros que han contribuido en el conocimiento que he adquirido como estudiante y persona.

A CONACyT por el apoyo económico brindado en parte de la carrera.

Y a todas esas personas que no aparecen en este documento que siempre me brindaron un granito de arena en mi formación.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

La agricultura es considerada uno de los sectores productivos más relevantes en México, ya que produce la alimentación primaria para millones de personas. Los tratados de libre comercio han favorecido a las empresas agrícolas aumentando la exportación del producto mexicano al extranjero esto gracias a la calidad y a la diversidad de los productos, creando una gran cantidad de empleos en los últimos años [1]. La tendencia en la producción de cultivos dentro de las ciudades a través de las granjas urbanas donde la mayoría de las regiones ofrecen condiciones óptimas para la producción, sin embargo, es importante considerar los diferentes tipos de clima del país utilizando la agricultura de precisión para obtener el nivel de tecnificación requerido. Se prevé que la superficie de producción en ambientes controlados siga creciendo ya que cada vez la gente en espacios cerrados genera sus propios alimentos a través de innovación en técnicas agrícolas [2]. Para la manipulación de las variables con el fin de obtener un clima óptimo dentro de un ambiente controlado es necesario implementar un sistema de control y automatización en conjunto con una arquitectura del Internet de las Cosas (IoT). Para la implementación de un sistema IoT existen arquitecturas de propósito general, dicha arquitectura se encuentra en tres capas, las cuales son, capa física, capa de enlace y capa de aplicación, estas capas no cumplen las necesidades presentadas en la agricultura de precisión, por eso en este trabajo de investigación se propone una arquitectura para el Internet de las Cosas que cumpla con cada una de las necesidades tecnológicas que se presentan en la agricultura de precisión, como por ejemplo la automatización y control, monitoreo de las variables en tiempo real, ayuda en la toma de decisiones, etc. Otra de las necesidades de un sistema IoT es que el sistema funcione en tiempo real, para cumplir dicho requerimiento se implementa un sistema Web desarrollado con MEAN (MongoDB, Express, Angular, Node.js), MEAN se distribuye en las capas de la arquitectura IoT, debido a que implementa el patrón de arquitectónico de software, Modelo Vista Controlador (MVC), para el almacenamiento de los valores que genera el sistema IoT se utiliza mongoDB, node.js conceptualiza el middleware como un repositorio de servicios web agilizando el desarrollo con express y para la visualización de los datos se implementa angular.

Problema de investigación

La demanda que se presenta en México en la exportación de productos de agricultura va en aumento dado la calidad y variedad de los productos, por lo que se necesita cumplir con normativas internacionales, como el Análisis de Peligros y Puntos Críticos de Control (HACCP), en las que se requiere tener en supervisión estricta una gran cantidad de variables que pueden ser monitoreadas y controladas dentro de los invernaderos y granjas urbanas, lo cual presenta un nuevo reto por los grandes volúmenes de datos redundantes que deben ser almacenados en tiempo real y la heterogeneidad de los dispositivos utilizados.

Motivación

Para el año 2050, se prevé que la población mundial aumente y alcance los 9,700 millones de personas, donde perjudicará gravemente las perspectivas de desarrollo. Las comunidades locales dependen de la agricultura para el empleo y la generación de ingresos, y sin embargo, ésta no se puede desarrollar más por la presión a la que ya se encuentran sometidos las tierras y los recursos hídricos. Las altas temperaturas y un suministro de agua menos fiable crearán serias dificultades para la pequeña ganadería, especialmente en ecosistemas de pastos áridos y semiáridos en latitudes bajas. El impacto del cambio climático en la seguridad alimentaria mundial se notará no solo en el suministro de alimentos, sino también en la calidad, el acceso y la utilización de estos y en la estabilidad de la seguridad alimentaria. La adopción de prácticas de gestión sostenible de la tierra, el agua, la pesca y la silvicultura por parte de los pequeños productores será fundamental para avanzar en los esfuerzos de adaptación ante el cambio climático, la erradicación de la pobreza global y la eliminación del hambre [3].

Se debe aprovechar que se han desarrollado nuevas tecnologías en el área de automatización, que han generado un abaratamiento de los dispositivos de monitoreo y control de los procesos industriales, en especial en la agricultura de precisión [4]. Adicionalmente se han incrementado y mejorado las técnicas de programación Web en la nube, para lograr la obtención de grandes volúmenes de información de sensores para

posterior análisis, para ello se hace una propuesta de una arquitectura IoT de cinco capas Adhoc para los sistemas IoT agrícolas en granjas urbanas, que permita disminuir la redundancia de los datos en el almacenamiento y que disminuya la cantidad de procesamiento de los datos para un control de las variables.

Objetivo general

Diseñar y validar una arquitectura del Internet de las Cosas para granjas urbanas.

Objetivos específicos

1. Proponer una arquitectura IoT las capas necesarias para granjas urbanas.
2. Validar la arquitectura IoT propuesta.
   2.1 Diseñar instrumentos de hardware y software para la validación de la arquitectura IoT.
   2.2 Desarrollar el instrumento de validación en software.
   2.3 Desarrollar el instrumento de validación en hardware.
   2.4 Corroborar que el instrumento de validación en software sea el correcto.
3. Analisis de la arquitectura IoT propuesta.

Hipótesis

Por medio de una plataforma para sistemas Internet de las cosas en tiempo real, es posible la validación de una arquitectura del Internet de las cosas para la agricultura precisión, que permita disminuir la redundancia de los datos que deben ser almacenados, así como optimizar los procesos de control en el sistema.

Estructura del documento

En el capítulo 2 se describen y se analizan las características en trabajos relacionados al tema de investigación que han tratado de solucionar los problemas que se presentan al implementar una arquitectura del Internet de las Cosas dentro de granjas urbanas.

En el capítulo 3 se describen los conceptos y teorías que conforman de manera general las arquitecturas del Internet de las Cosas y la implementación y desarrollo de ésta.

En el capítulo 4 se presenta la propuesta de una arquitectura IoT, esta se describe capa a capa, con el funcionamiento que desempeña cada uno de los componentes, también se describen cada una de las actividades que se realizaron para implementar la propuesta y comprobar que la arquitectura desempeñará las tareas para la cual fue diseñada.

Capítulo 2. Estado del arte

En la literatura científica actual, existen una gran cantidad de trabajos sobre arquitecturas, algunos utilizan el paradigma Middleware pero en un dominio global, sin considerar las necesidades locales que presentan los sistemas de agricultura de precisión, otros usan algunos componentes de los sistemas IoT, no considerando todas las propiedades de dichos sistemas. En el año 2014, surge un middleware genérico denominado GSN que se encuentra entre la capa de sensores y la manipulación de datos, dichos datos son almacenados en un servicio en la nube llamado FireBase [5], el uso del middleware dentro de la arquitectura de un sistema de Internet de las Cosas, también es utilizado para la adaptación dinámica de la recolección de acuerdo a la QoS (Quality of service) [6]. Otro trabajo es sobre el uso de distintos protocolos de comunicación entre dispositivos, siendo una parte relevante en el desarrollo de sistemas del Internet de las Cosas en el año 2011, por Chunxiao Fan, Zhigang Wen, Fan Wang and Yuexin Wu [7], los cuales crean un sistema llamado SmartScene para obtener datos de la capa de sensores, utilizando un capa middleware para que cualquier tipo de sensor pueda proporcionar datos para analizar; este sistema trabaja en la capa de comunicación con los protocolos Zigbee y RFID.

Se han propuesto nuevas arquitecturas aplicando concepto SPF (Sieve, Process, and Forward), para el procesamiento de los grandes volúmenes de datos generados por dispositivos IoT [8], utilizando pre procesamiento antes de trasmitir los datos para su futuro análisis.

Una parte fundamental de los sistemas del Internet de las Cosas son las tecnologías web, para la creación de APIs de recolección de información, en el 2015 en la Conferencia Internacional del Futuro del Internet de las Cosas y la Nube, se muestra el diseño de una plataforma para las Ciudades Inteligentes llamada ALMANAC, donde utilizan web services con REST en java [9] en la plataforma EcoDiF también son utilizadas tecnologías como EEML, EMML y HTTP dentro de ésta, utilizando GET, PUT, POST y DELETE para la monitorización, control, procesamiento y almacenamiento de los datos obtenidos por los dispositivos físicos el [10]

El uso de tarjetas programables como Raspberry ha permitido el abaratamiento de los sistemas del Internet de las Cosas y también permitiendo la innovación como un

dispositivo móvil de recolección de datos en ciudades inteligentes, esto es posible a un middleware que adapta distintos sensores [11]

Para la implementación de un sistema IoT en la agricultura de precisión, donde se han propuesto arquitecturas específicas para este tipo de sistema IoT, donde consiste en tener una capa de sensores conectados a la nube con ZigBee y la nube conectada con los actuadores donde estos controlan la humedad y temperatura dentro del invernadero [12].

El dominio de IoT abarca una variedad en los ámbitos sociales como por ejemplo la agricultura, tráfico vehicular, información de una ciudad etc, por ello se han propuesto arquitecturas con computo en la niebla para unir distintas aplicaciones del internet de las cosas con dominios diferentes [13], el cómputo en la niebla también es utilizado para evitar procesos en los dispositivos que están conectados al Internet de las Cosas, para ello se han propuesto arquitecturas que unen las características de del cómputo en la niebla y el cómputo en la nube [14], una de las ventajas que se ha obtenido al utilizar un cómputo en la niebla es la disminución de tiempo en la espera de una respuesta en un servicio en la nube, esto a través de un ruteo de aplicaciones ya que este tipo de computo sirve como intermediario ente la capa de aplicación y la capa física de la arquitectura IoT[15]. El IoT también ha sido utilizado en el control y automatización de variables dentro de granjas urbanas esto a través de computo en la niebla [16]. Las capas y características que tienen las arquitecturas en los trabajos relacionados que se han implementado en granjas urbanas, se abstraen y se muestran en la tabla 2.1.

	Capa física	Capa de enlace	Middleware en software	Capa de aplicación	Cómputo en la niebla	Especializado para la agricultura
GSN	x		x	x		
ALMANAC	x	x		x		
SmartScene	x	x		x		
EcoDiF	x	x	x	x		
RT-Middleware	x	x		x		
Architecture base on QoS	x	x	x	x		
App for Green house	x	x		x		x
SPF	x	x		x	x	
FogSEA	x	x		x	x	
Fog-Cloud	x	x			x	
Fog computing based IoT architecture	x			x	x	
Control by fog	x	x			x	

computing						

Capítulo 3. Marco Teórico

3.1 Agricultura

La agricultura es una actividad económica en la cual se tienen técnicas y conocimientos para el cultivo de la tierra a través de la explotación de los recursos de la tierra mediante acciones humanas [17].

En términos generales, la Agricultura Urbana (AU) considera el cultivo, procesamiento, distribución y consumo de productos agrícolas dentro del área de la ciudad, utilizando con fines productivos espacios que no son utilizados, como terrenos baldíos y azoteas; Incluye no sólo la producción de vegetales comestibles, como frutas y hortalizas, sino también una amplia gama de especies destinadas a medicina natural, fibras vegetales para cestería y floricultivos, entre otros.

El fenómeno de la AU se caracteriza por su gran adaptabilidad y movilidad, sirviendo de base alimentaria y económica a comunidades urbanas y periurbanas, a través del desarrollo creativo de estrategias agroproductivas que contribuyen a mejorar la calidad alimentaria. La Agricultura Urbana no se limita exclusivamente a un nivel de alimentación, también incorpora la posibilidad de cultivos recreativos y de autoconsumo en grupos socioeconómicos medios, de operaciones comerciales de pequeña escala para microempresarios, familias, actividades terapéuticas y educativas, mediante el desarrollo agrícola en patios traseros, terrazas, balcones, jardines escolares, hospitales, prisiones y otros establecimientos.

En la actualidad, la agricultura urbana tiene un rol creciente en la agenda internacional, donde de hecho se está reconociendo como parte esencial de una estrategia global para lidiar con los retos colocados por el rápido crecimiento urbano en los países en desarrollo. La agricultura urbana no es una solución total para los problemas que irán enfrentando en el futuro las ciudades de los países en desarrollo, pero sí es ingrediente fundamental de cualquier programa que busque tornar las ciudades en desarrollo y mejorar la vida de sus habitantes. A lo largo de los 25 años siguientes, el firme ascenso de la agricultura urbana en la agenda internacional estaría

acompañado por el creciente involucramiento en este tema de varias organizaciones del sistema de la ONU, a menudo en colaboración con investigaciones pioneras que estaban siendo apoyadas por el Centro Internacional de Investigaciones para el Desarrollo (IDRC) [18].

3.1.1 Agricultura de precisión

Según US National Research Council [19], la agricultura de precisión (AP) es una estrategia de manejo que utiliza la tecnología de la información para captar datos de múltiples fuentes para generar decisiones asociadas a la producción de cultivos. La agricultura de precisión se define como el proceso de poner el producto adecuado, la cantidad adecuada, en el lugar adecuado y en el momento adecuado.

La Agricultura de Precisión es una aproximación a un sistema para manejar cultivos y suelos en forma selectiva de acuerdo con sus necesidades. Utiliza la experiencia de muchas disciplinas e integra herramientas tecnológicas de información que permiten a los administradores agrícolas tener una mejor. Como se pueden visualizar las etapas del de la agricultura de precisión en la figura 3.1.

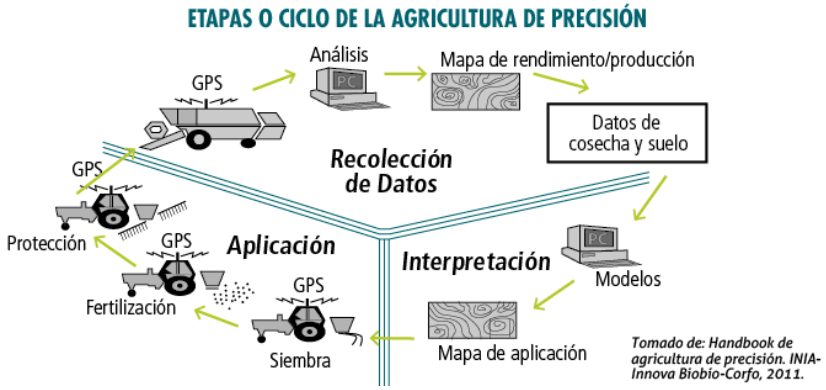


Figura 3. 1 Etapas de la agricultura de precisión.

En donde existen las etapas, Aplicación, Recolección de datos, e interpretación, que recopilan datos del sistema, para su posterior interpretación para retroalimentar el sistema.

3.2 Internet de las Cosas

Según la normatividad de ITU-T Y.2060 [20], el IoT puede verse como una infraestructura global para la sociedad de la información, que proporcionan servicios avanzados interoperables existentes y en evolución para interconectar (física y virtual) "cosas" utilizando las tecnologías de información y comunicación. Dichas "cosas", son objetos del mundo físico ("cosas físicas") o del mundo de la información ("mundo virtual") que pueden identificarse e integrarse en las redes de comunicación. Las "cosas" tienen información asociada, que puede ser estática y dinámica [21]. Las "cosas" físicas existen en el mundo físico y son capaces de ser detectadas, manipuladas y conectadas. Los ejemplos de cosas físicas incluyen el entorno circundante, robots industriales, electrodomésticos, etc. Existen cosas virtuales en el mundo de la información y se pueden almacenar, procesar y acceder. Los ejemplos de cosas virtuales incluyen contenido multimedia y software de aplicación.

Un dispositivo es una pieza de equipo con las capacidades obligatorias de comunicación y capacidades opcionales de detección, manipulación, captura de datos, almacenamiento de datos y procesamiento de datos. Los dispositivos recopilan diversos tipos de información y la proporcionan a las redes de información y comunicación para su posterior procesamiento.

Algunos dispositivos también ejecutan operaciones basadas en datos recibidos de las redes de información y comunicación.

3.2.1 Características fundamentales

Las características básicas y fundamentales de un sistema del Internet de las Cosas son las que se ilustran en la figura 3.2.

Figura 3. 2. Características de un sistema IoT.

Interconectividad: con respecto al IoT, cualquier cosa puede estar interconectada con la infraestructura global de información y comunicación.

Heterogeneidad: los dispositivos en el IoT son heterogéneos ya que se basan en diferentes plataformas y redes de hardware. Pueden interactuar con otros dispositivos o plataformas de servicio a través de diferentes redes.

Cambios dinámicos: el estado de los dispositivos cambia dinámicamente, por ejemplo, durmiendo y despertando, conectados y / o desconectados, así como el contexto de los dispositivos, incluida la ubicación y la velocidad. Además, la cantidad de dispositivos puede cambiar dinámicamente.

Gran escala: la cantidad de dispositivos que deben administrarse y comunicarse entre sí será al menos un orden de magnitud mayor que los dispositivos conectados a la Internet actual. La relación de comunicación activada por los dispositivos en comparación con la comunicación desencadenada por humanos se desplazará notablemente hacia la comunicación activada por el dispositivo.

Según pronósticos conservadores, por la tendencia de crecimiento de los dispositivos en los últimos años, en el 2050, existirán más de 50 Billones de dispositivos y casi siete dispositivos por persona interconectado a internet [19], como se puede visualizar en la figura 3.3.

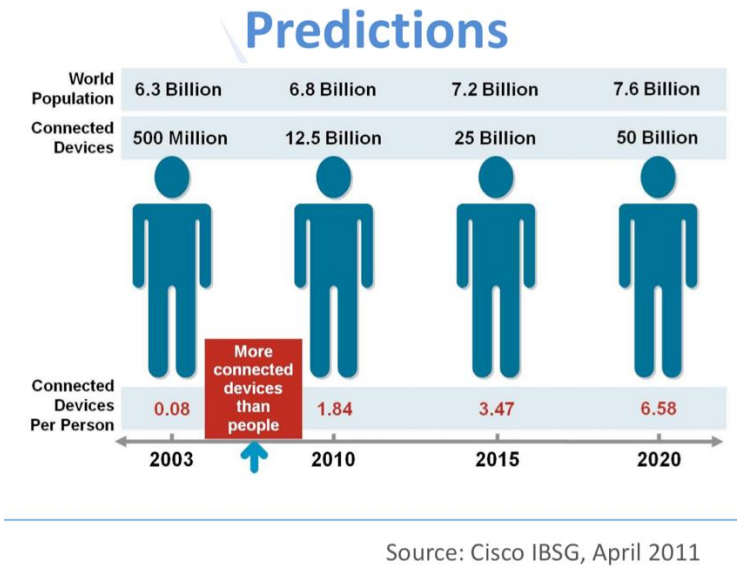


Figura 3. 3. Pronostico de dispositivos.

El internet de las cosas puede ser definido como una infraestructura global de red dinámica con capacidades auto-configurables basada en un estándar y protocolos de comunicación interoperables donde las “Cosas” físicas y virtuales tienen identidades, y personalidades virtuales usando interfaces inteligentes, y son integrados sin problemas dentro de la red de información, con frecuencia se asocia los datos con los usuarios y su entorno, como se puede conceptualizar en la figura 3.4.

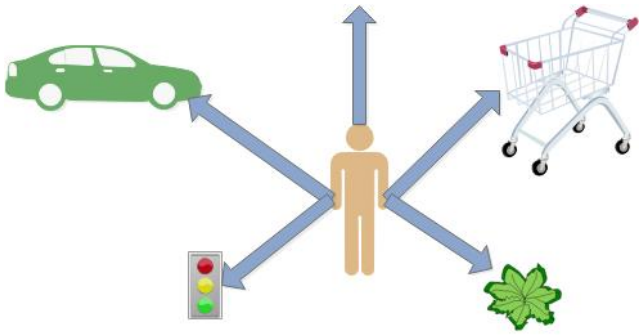


Figura 3. 4. Personas en su entorno.

En la figura 3.5, se muestra la clasificación del Internet de las Cosas, de los distintos dominios de aplicación de IoT [22].

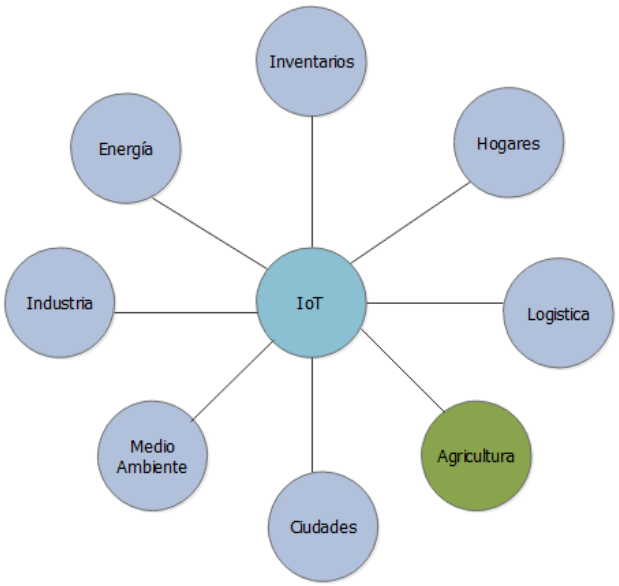


Figura 3. 5. Dominio del IoT.

3.2.2 Arquitectura del Internet de las Cosas

De acuerdo a Bahga[22], el Internet de las Cosas está compuesto por una arquitectura, que contiene capas, como se ilustra en la figura 3.6. La cual muestra en diagrama de bloques las tres capas que contiene.

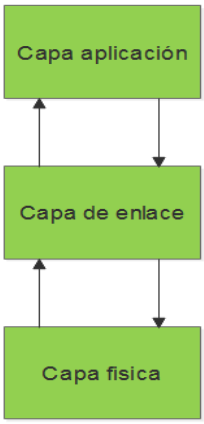


Figura 3. 6. Arquitectura IoT en 3 capas.

En la capa **física** se encuentran los dispositivos los cuales son encargados de recolectar información, actuar, monitorear, los dispositivos que recolectan información en el entorno que los rodean son sensores como temperatura, humedad, oxigeno etc. Y actuadores como bombas de riego, ventiladores, hornos etc. Ejemplo en la figura 3.7 de los componentes de un sistema que controla la temperatura de acuerdo a un sensor y el enfriamiento a partir de una hora establecida.

Figura 3. 7. Ejemplo de capa física.

La capa de **enlace** es la que se encarga de las comunicaciones utilizadas en el sistema y la encargada de que los datos de la capa física lleguen a la capa de aplicación, estas pueden ser alámbricas e inalámbricas (figura 3.8).

Figura 3. 8. Ejemplo de capa de enlace.

La visualización, control y monitoreo del sistema que el usuario tendrá, se encuentra en la capa de **aplicación,** aquí se muestran herramientas para la ayuda de la toma de decisiones como estadísticas y brinda un control manual sobre los actuadores, en la figura 3.9 se ejemplifica la presentación de los datos al usuario a través de graficas de barras**.**

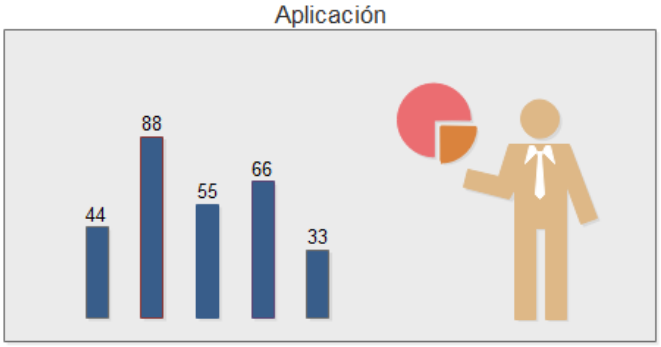


Figura 3. 9. Ejemplo de capa de aplicación.

Bahga [22] clasifica en seis niveles los tipos de sistemas que pueden ser representados con una arquitectura IoT, cada nivel cumple con distintas necesidades y alcances distribuyendo de manera distinta sus componentes, estos siendo distribuidos de forma local o externa, como por ejemplo las necesidades que se presentan en una arquitectura de nivel 1 todos los componentes se encuentran de forma local, en el nivel 2 el componente de aplicación se encuentra en la nube, en el nivel 3 componente de análisis se encuentra en la nube, en el nivel 4 el análisis se encuentra de forma local, en la figura 3.10 se muestra la distribución y componentes de una arquitectura de nivel 5.

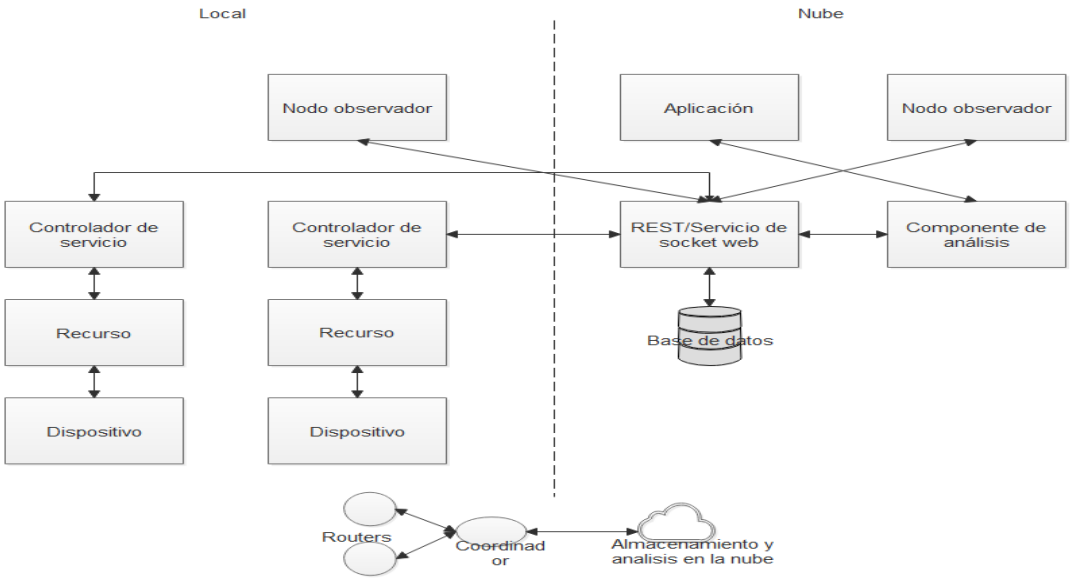


Figura 3. 10. Arquitectura IoT de nivel 5[22].

En el nivel 5 de una arquitectura IoT, los **dispositivos** son encargados de recolectar información, activar actuadores a través de funciones de control o de forma manual, los **recursos** son componentes de software en los dispositivos IoT para el acceso, procesamiento y almacenamiento de información, el **controlador de servicio**, es un servicio nativo que interactúa con los servicios Web, la **base de datos** es el almacenamiento en la nube de los datos generados por el sistema IoT, el **componente de análisis** es el responsable de analizar los datos generados por el sistema y presentarlos al usuario de una forma entendible y la **aplicación** proporciona visualización, control y monitoreo del sistema que el usuario tendrá.

Para poder representar la comunicación en IoT, se definen modelos, como lo son Petición-Respuesta y Publicar-Suscribir. En el modelo Petición-Respuesta el cliente envía una petición al servidor este prepara los datos solicitados para enviar una respuesta al cliente. Se muestra de forma gráfica el funcionamiento de este modelo de comunicación en la figura 3.11.

Figura 3. 11. Modelo Petición-Respuesta.

En el modelo Publicar-Suscribir se tiene un publicador (fuente de los datos) que envía respuestas a los clientes a través del agente (administrador), el agente de acuerdo a la información lanzada por el publicador tiene clasificados a los clientes y asi la información que lanza el publicador le llega solo a los clientes inscritos a ese tipo de información, como se muestra en la figura 3.12.

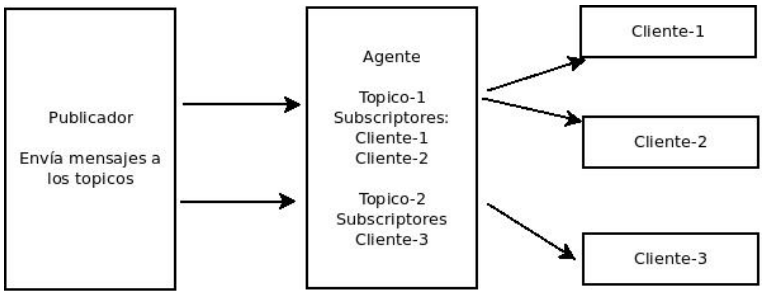


Figura 3. 12. Modelo Publicar-Suscribir.

3.3 Aplicaciones distribuidas

La presencia de redes de forma global permite la proliferación de aplicaciones, dado que las partes de una aplicación distribuida se ejecutan en diferentes ubicaciones físicas, estas aplicaciones ofrecen una serie de ventajas como la distribución geográfica, las aplicaciones se vuelven tolerantes a fallas a través de la replicación. A su vez, un sistema distribuido cuenta con heterogeneidad es decir, diferentes plataformas de hardware, tecnologías de red, sistemas operativos y lenguajes de programación pueden colaborar en un solo sistema [23].

Un sistema distribuido es un sistema de procesamiento de información que contiene una cantidad de equipos independientes que cooperan entre sí, a través de una red de comunicaciones para alcanzar un objetivo específico.

Los equipos están conectados entre sí a través de una red de comunicaciones que permite el intercambio de mensajes entre las computadoras (ver Figura 3.13). El objetivo de este intercambio de mensajes es lograr una cooperación entre computadoras con el propósito de alcanzar un objetivo común [23].

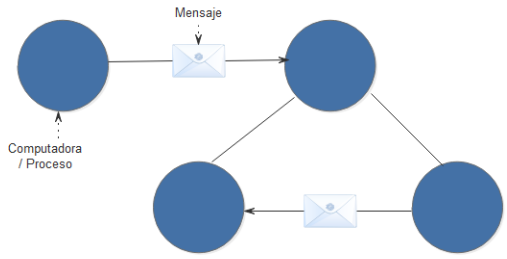


Figura 3. 13. Intercambio de mensajes entre clientes.

Una de las características de las aplicaciones distribuidas es la transparencia, existen distintos tipos de transparencia las cuales son: **transparencia de locación** es decir se puede acceder al sistema sin importar donde se encuentre físicamente, la **transparencia de acceso** se encarga de que el acceso remoto o local sea el mismo, el uso de distintos tipos de tecnología de programación, sistemas operativos o hardware dentro de un mismo sistema garantiza una **transparencia de tecnologías,** los usuarios no se dan cuenta que comparten información con otros usuarios dada la **transparencia de concurrencia** como se visualiza en la figura 3.14 .

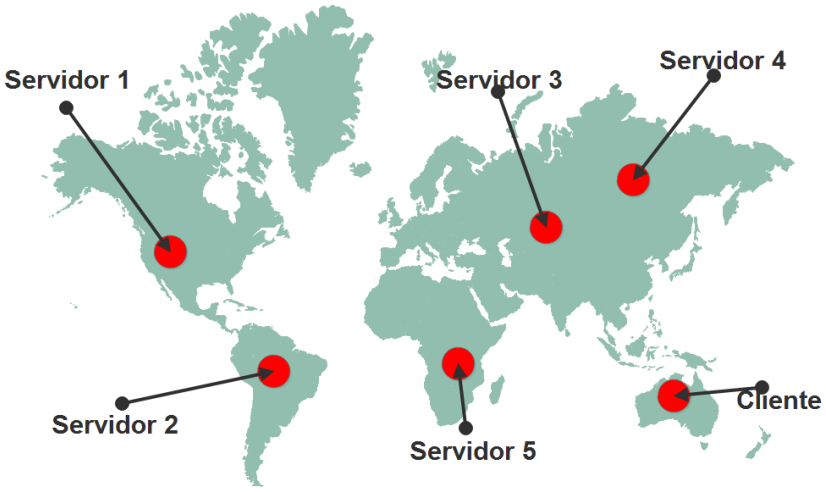


Figura 3. 14. Transparencia de locación.

3.3.2 Modelo Cliente/Servidor

El modelo cliente/servidor es un modelo de comunicaciones en la cual los clientes están enlazados a un servidor, en el cual centralizan los diversos recursos, servicios y aplicaciones poniendo a disposición cada vez que un servicio es solicitado por uno o más clientes (figura 3.15).

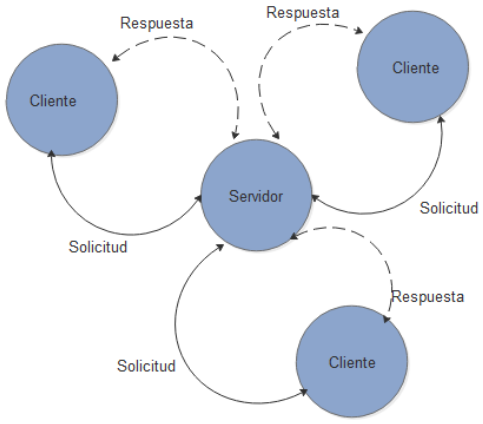


Figura 3. 15. Modelo Cliente-Servidor.

La separación entre cliente y servidor es una separación lógica, donde el servidor no se ejecuta necesariamente sobre una sola máquina, ni es necesariamente un sólo

servicio. Mientras que sus propósitos varían en los servicios, la arquitectura sigue siendo la misma como se observa en la figura 3.16.

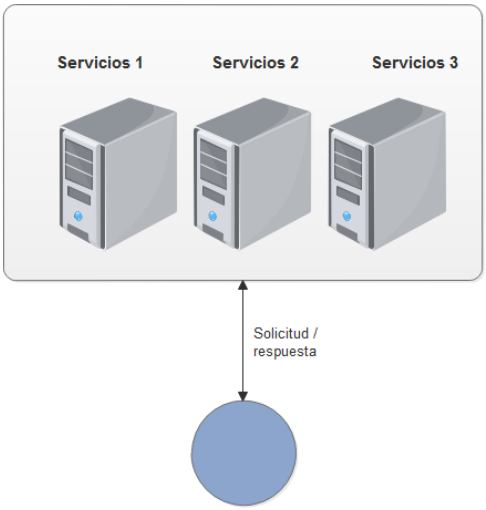


Figura 3. 16. Servidor de varios servicios.

3.3.3 Middleware

La implementación de un sistema distribuido es más compleja que en los sistemas centralizados. Existen diferentes opciones para reducir la complejidad de un sistema distribuido, desde en el nivel de hardware hasta los lenguajes de programación utilizados. Las soluciones de software suelen ser más sencillas. Estas condiciones conducen al concepto de un middleware en software. Un middleware ofrece servicios generales que soportan la ejecución distribuida de aplicaciones. El término middleware sugiere que es un software ubicado entre el sistema operativo y la aplicación, la función que desempeña es una capa de interpretación para comunicar aplicaciones o usuarios. El uso de middleware brinda la posibilidad a los usuarios de hacer solicitudes a servicios, visto de forma abstracta, el middleware se puede ver como un "mantel" que se propaga por una red heterogénea, ocultando la complejidad de la tecnología subyacente de la aplicación que se ejecuta en ella.

El middleware tiene la tarea de ser una interfaz entre los distintos tipos de clientes y servicios proporcionados por el servidor (ver figura 3.17), es decir que se puedan comunicar clientes con clientes y clientes con distintos servicios.

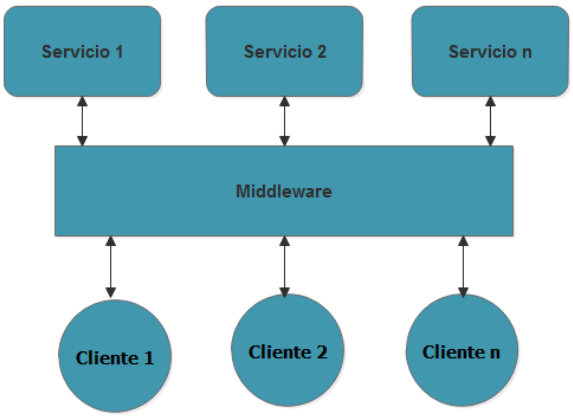


Figura 3. 17. Middleware como una interfaz.

3.3.4 Middleware para la computación ubicua

El término "computación ubicua" fue conceptualizado por Mark Weiser, imaginó un mundo de computadoras omnipresentes que se convierten invisibles al estar acoplados en el entorno físico con el objetivo de apoyar a la gente discretamente en el cumplimiento de sus tareas. Esto con dispositivos equipados con sensores y actuadores. Estos dispositivos pueden percibir y controlar ciertos parámetros de su entorno físico y puede comunicarse entre sí. Supongamos que se tiene contratado una serie de servicios sobre computación ubicua en nuestra casa, como riego de las plantas, encendido y apagado de luces del hogar y la vigilancia remota de las cámaras. Una persona al no estar cerca para activarlas lo hace a través de un celular para acceder a los servicios antes mencionados, un ejemplo se muestra en la figura 3.18.

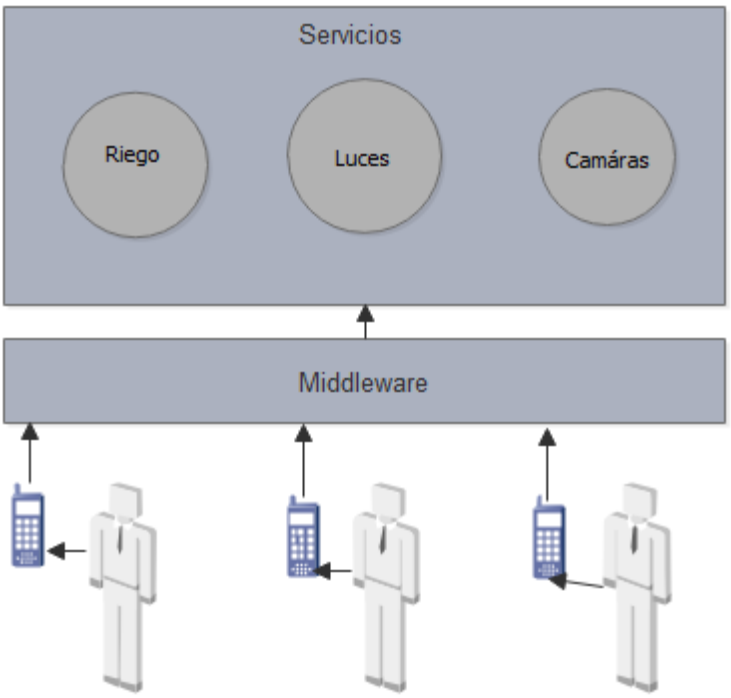


Figura 3. 18. Personas utilizando servicios.

3.4 Tecnologías Web

Los servicios web y las arquitecturas orientadas a servicios son tecnologías con el objetivo común de hacer componentes de software y aplicaciones disponibles a través de interfaces estandarizadas. Estas tecnologías facilitarán el desarrollo más simple de distribuido. Básicamente, los servicios web permiten el acceso a una funcionalidad a través de la Web usando un conjunto de estándares abiertos que hacen que la interacción sea independiente de los aspectos de implementación, como la plataforma del sistema de operaciones y el lenguaje de programación utilizado, en la figura 3.19 se muestra la arquitectura de un servicio web [25].

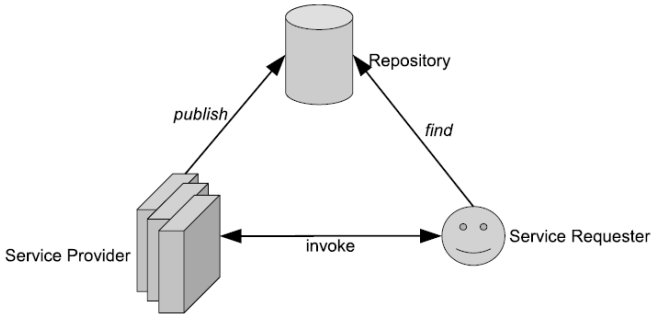


Figura 3. 19. Arquitectura de un servicio web.

3.4.1 REST

Hay varias formas de implementar comunicación entre aplicaciones para que estas sean heterogéneas, una de ellas es el uso de estándares enfocados en servicios web basados en un protocolo estándar que define cómo dos objetos en diferentes procesos pueden interactuar a través del intercambio de datos como lo es SOAP (Simple Object Access Protocol), y WSDL (Web Services Description Language) describe los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios proporcionados por un servidor; junto a estos estándares hay una solución simple basada en HTTP denominado Representational State Transfer (REST).

REST utiliza un conjunto de reglas y especificaciones para que varias aplicaciones se comuniquen entre ellas, de esta manera, podemos garantizar el intercambio de mensajes o datos en formato estándar. La principal función de REST es que puede utilizarse como interfaz entre sistema operativo que contenga bases de datos y un sistema que use protocolos HTTP.

REST se identifica por los principios de recursos direccionables, interfaces restringidas utilizando los métodos HTTP, los principios clave de REST son:

- Asociar ID a recursos
- Uso de métodos HTTP estándar
- Múltiples formatos enviados por un recurso
- Protocolos sin estado

La arquitectura de REST se basa en los mensajes de solicitud y respuesta transferidos entre clientes y servidores sin que ninguno de los nodos participantes realice un seguimiento del estado de las sesiones anteriores.

La representación del recurso que se envía al cliente depende de la solicitud y cómo el servidor envía los datos, como se ilustra en la figura 3.20 [26].

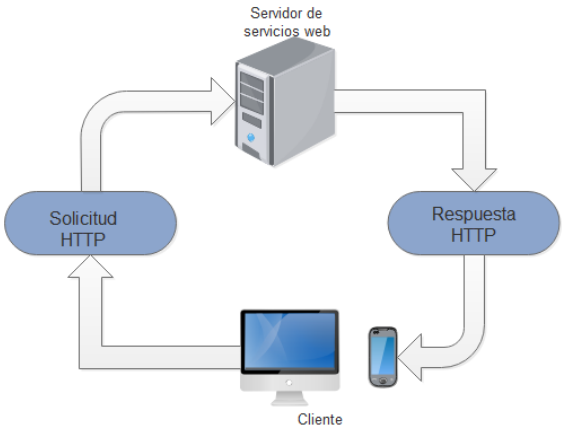


Figura 3. 20. Ejemplo de solicitud de un servicio.

3.4.2 Métodos usados en REST

- GET: La solicitud GET regresa una representación de un recurso desde un servidor al cliente.
- POST: La solicitud POST es usada para crear un recurso en un servidor basado en la representación de un recurso que un cliente a enviado.
- PUT: La solicitud PUT es usada para actualizar o crear referencia a un recurso en un servidor.
- DELETE: La solicitud DELETE puede borrar un recurso en un servidor
- HEAD: La solicitud HEAD solicita revisar un recurso sin recuperarlo.

3.4.3 Aplicaciones asíncronas

En las aplicaciones que implementan un modelo cliente-servidor se utilizan escenarios de respuesta a solicitudes normales, en donde se mantiene un hilo en ejecución para cada solicitud hasta que la respuesta esté disponible. Esto se convierte en un cuello de botella en los casos en que el servidor tarda mucho tiempo en procesar las solicitudes, y el proceso de subproceso espera que el servidor finalice la preparación de la respuesta requerida y, por lo tanto, no puede aceptar ninguna nueva solicitud entrante. Existen aplicaciones asíncronas es decir no bloqueantes, esto permitiendo la aceptación de nuevas solicitudes agregándolas a una cola.

La programación asíncrona establece la posibilidad de hacer que algunas operaciones devuelvan el control al programa llamante antes de que hayan terminado mientras siguen operando en segundo plano. Esto agiliza el proceso de ejecución y en general permite aumentar la escalabilidad, pero complica el razonamiento sobre el programa.

3.5 Metodología de desarrollo de software

Una Metodología para el desarrollo de software, consiste principalmente en el uso de herramientas, técnicas, métodos y modelos para el desarrollo. Normalmente este tipo de metodologías, tienen que estar documentadas, para que los desarrolladores que estarán incluidos en un proyecto, entiendan la metodología y en algunos casos el ciclo de vida del software que se pretende seguir.

La metodología de desarrollo de software en cascada es lineal es decir, que no se puede avanzar a la siguiente fase de desarrollo si la anterior no se encuentra totalmente terminada, las fases de desarrollo de la metodología en cascada se muestran en la figura 3.21 [26].

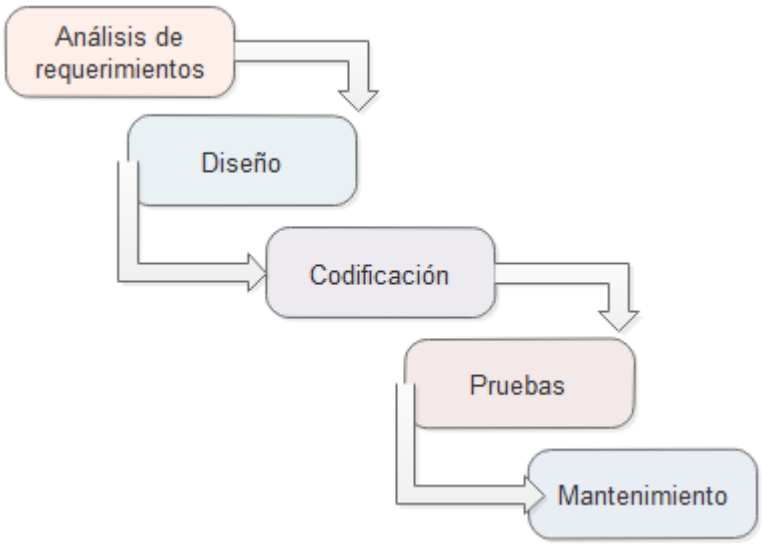


Figura 3. 21. Metodología de cascada.

Cada paso de la metodologia tiene un objetivo dentro del desarrollo de un sistema de software, para el cual se explican a continuación dichos objetivos.

1. **Análisis de requerimientos**. El primer nivel del modelo cascada, es el análisis de requisitos. Son los objetivos que el software debe alcanzar y cubrir.

2. **Diseño**. Aquí se elabora la la estructura del sistema y se determinarán las especificaciones para cada una de las partes que lo compondrán

3. **Codificación**. El diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

4. **Pruebas**. En esta fase se verifica que el sistema funciona antes de entregarlo o mostrarlo al usuario final para su uso.

5. **Mantenimiento**. En el mantenimiento de software, se solucionan errores y se añaden funcionalidades.

3.6 Arquitectura de software

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema es decir comprender los elementos de software, relaciones entre ellos, y propiedades de ambos [27].

3.6.1 Patrón Modelo Vista Controlador

El Modelo Vista Controlador es un patrón de diseño arquitectónico de software, que sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario. En este tipo de arquitectura existe un sistema central o controlador que gestiona las entradas y la salida del sistema, uno o varios modelos que se encargan de buscar los datos e información necesaria y una interfaz que muestra los resultados al usuario final. Es muy usado en el desarrollo web porque al tener que interactuar varios lenguajes para crear un sitio es muy fácil generar confusión entre cada componente si estos no son separados de la forma adecuada. Este patrón permite modificar cada uno de sus componentes si necesidad de afectar a los demás. Los componentes de un sistema son:

- **Modelo**, este componente se encarga de manipular, gestionar y actualizar los datos. Si se utiliza una base de datos aquí es donde se realizan las consultas, búsquedas, filtros y actualizaciones.

- La **Vista** se encarga de mostrarle al usuario final las pantallas, ventanas, páginas y formularios; el resultado de una solicitud. Desde la perspectiva del programador este componente es el que se encarga del frontend.

- El **Controlador** se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. Por medio de él se comunican el modelo y la vista: solicitando los datos necesarios; manipulándolos para obtener los resultados; y entregándolos a la vista para que pueda mostrarlos (figura 3.22).

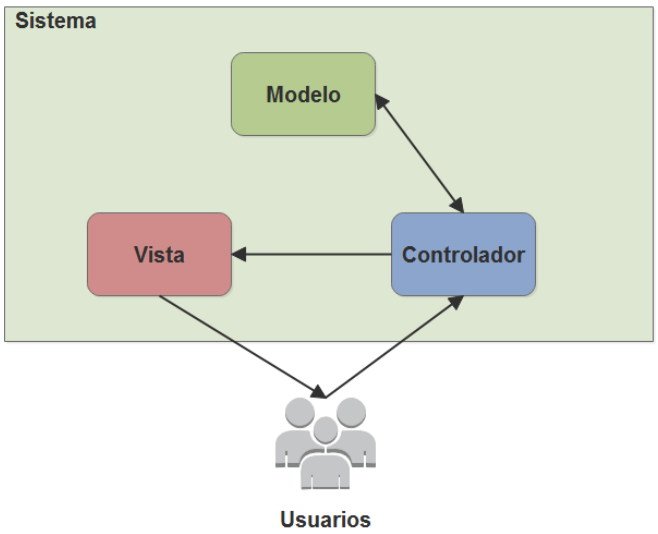


Figura 3. 22. Interacción MVC.

3.7 Sistemas de automatización y control

El control automático ha desempeñado un papel importante en el avance de la ingeniería. El control automático se ha convertido en una parte importante e integral en los sistemas que requieran el control de variables como por ejemplo la temperatura, presión, humedad, flujo, etc.

Un sistema es una combinación de componentes que actúan juntos y realizan un objetivo determinado. Un sistema no está necesariamente limitado a los sistemas físicos. El concepto de sistema se puede aplicar a fenómenos abstractos y dinámicos, como los que se encuentran en la economía. Por tanto, la palabra sistema debe interpretarse en un sentido amplio que comprenda sistemas físicos, biológicos, económicos y similares. Un sistema que mantiene una relación establecida entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control, se denomina sistema de control realimentado o de lazo cerrado. También existen los sistemas de lazo abierto en los cuales la salida no afecta la acción de control.

Los sistemas de control realimentados se denominan también sistemas de control de lazo cerrado. En un sistema de control en lazo cerrado, se alimenta al controlador con la señal de error de actuación, que es la diferencia entre la señal de entrada y la salida de realimentación con el fin de reducir el error y llevar la salida del sistema a un valor esperado. Los elementos que componen a un sistema de lazo cerrado son:

- Elemento de **comparación**, este elemento compara el valor requerido con el valor medido de lo que se obtiene a la salida, y produce una señal de error la cual indica la diferencia del valor obtenido a la salida y el valor requerido.
- Elemento de **control**, este elemento decide que acción tomar cuando se recibe una señal de error.
- Elemento de **corrección**, este elemento se utiliza para producir un cambio en el proceso al eliminar el error,

- Elemento de **proceso**, el proceso o planta, es el sistema dónde se va a controlar la variable.
- Elemento de **medición**, este elemento produce una señal relacionada con la condición de la variable controlada, y proporciona la señal de realimentación al elemento de comparación para determinar si hay o no error.

Los elementos se visualizan en la figura 3.23.

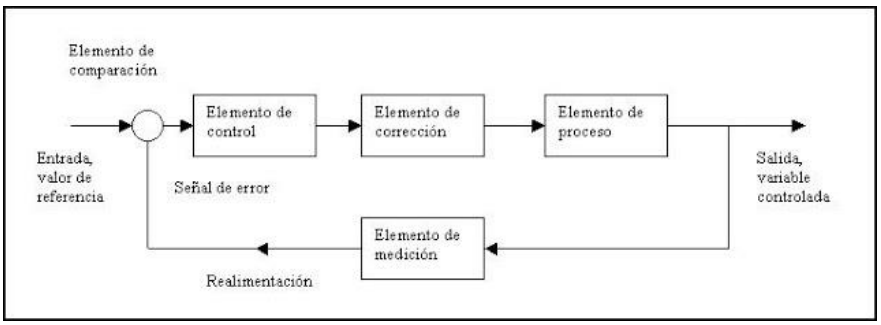


Figura 3. 23. Elementos de control de lazo cerrado.

Los sistemas de lazo abierto son en los cuales la salida no afecta la acción de control. En un sistema en lazo abierto no se mide la salida ni se realimenta para compararla con la entrada.

En cualquier sistema de control en lazo abierto, la salida no se compara con la entrada de referencia. Por tanto a cada entrada de referencia le corresponde una condición operativa fija; como resultado, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones, un sistema de control en lazo abierto no realiza la tarea deseada. En la práctica, el control en lazo abierto sólo se utiliza si se conoce la relación entre la entrada y la salida y si no hay perturbaciones internas ni externas. Los elementos que conforman un sistema de lazo abierto son:

- Elemento de control, este elemento determina qué acción se va a tomar dada una entrada al sistema de control.
- Elemento de corrección, este elemento responde a la entrada que viene del elemento de control e inicia la acción para producir el cambio en la variable controlada al valor requerido.

- Elemento de proceso, el proceso o planta en el sistema en el que se va a controlar la variable.

Como se observa en la figura 3.24, los sistemas no utilizan la retroalimentación de señales.

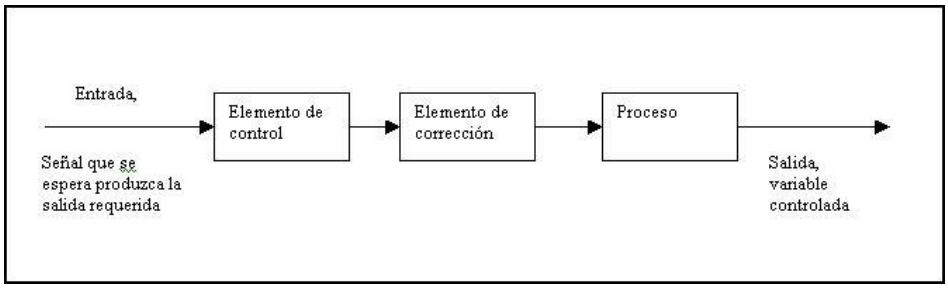


Figura 3. 24. Elementos de control de lazo abierto.

Capítulo 4. Propuesta y validación de una arquitectura IoT

4.1 Propuesta de una arquitectura IoT para granjas urbanas.

Se han identificado las características que se deben presentar en una arquitectura diseñada e implementada en granjas urbanas, en la que en distintos trabajos el común denominador son la capa física, capa de enlace y la capa de aplicación; en el presente trabajo se integra un middleware diseñado para la agricultura y una capa con el cómputo en la niebla. Lo cual se explicará en el transcurso de este capítulo, y que se observa en la Figura 4.1 de la propuesta, donde la capa física se encuentran todos los dispositivos que interactúan con el mundo real, el computo en la niebla y la capa de enlace conceptualizan un middleware local que tiene la tarea del control y automatización y el enlace entre sensores y actuadores, el middleware global es la interfaz para que las granjas urbanas se acoplen al sistema IoT ideado para la agricultura de precisión y en la capa de aplicación se encuentra herramientas de software que ayudan a un usuario final con la toma de decisiones .

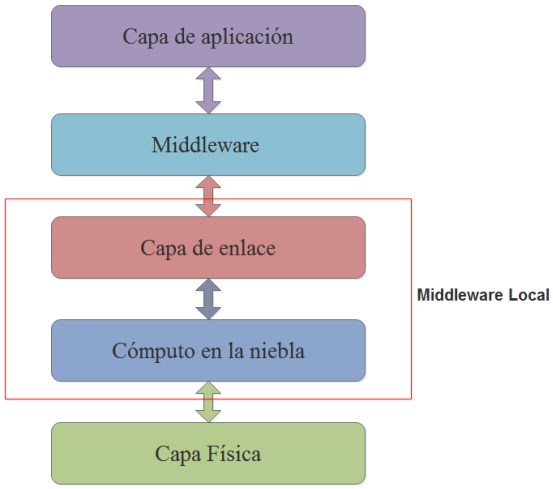


Figura 4. 1. Arquitectura IoT propuesta.

En la figura 4.2 se muestra una vista general de la arquitectura propuesta para un sistema del Internet de las Cosas implementada en granjas urbanas. En donde en la capa física se encuentra en forma local dentro de las granjas urbanas, conceptualizando en primera que dentro de cada granja se encuentran sensores y actuadores conectados a través de un middleware local así como el middleware localizado en forma global, se

encuentra de forma externa para que sin importar la geo posición de las granjas, puedan acoplarse al sistema de forma general, y así poder intercambiar información entre granjas.

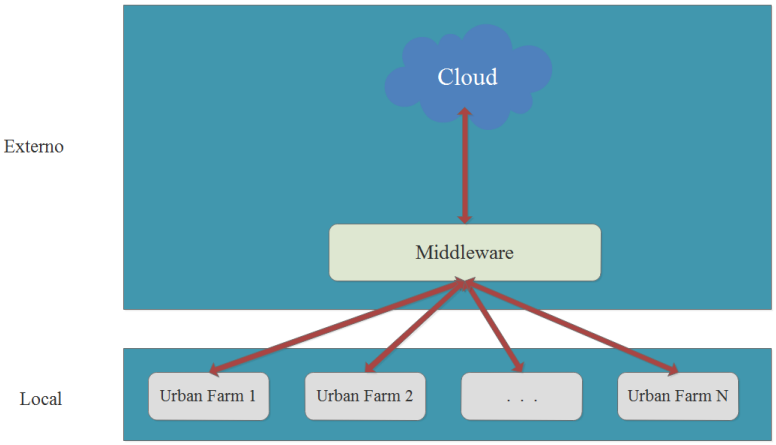


Figura 4. 2. Vista general de la propuesta.

La arquitectura propuesta en este trabajo de investigación permite que se puedan ir acoplando de forma sencilla cualquier número de granjas urbanas, es decir, se puede conceptualizar que existen *N* cantidad de granjas, que conformarán el sistema final y cada granja urbana puede tener, *N* cantidad de sub-granjas (figura 4.3), dando resultado un sistema de sistemas.

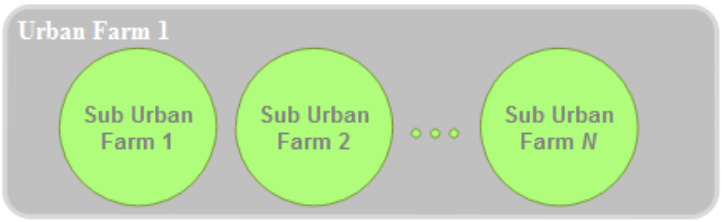


Figura 4. 3. Granja urbana de *N* sub granjas.

Lo que conforma cada subsistema de la granja urbana es un middleware local, sensores y actuadores, el middleware local hace la función de interfaz entre los sensores y actuadores como se observa en la figura 4.4.

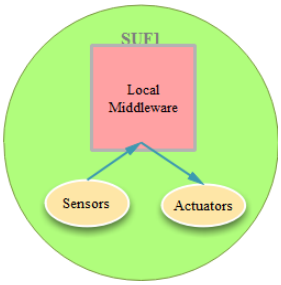


Figura 4. 4. Componentes de una sub granja urbana.

El middleware local está conformado por un componente que hace cómputo en la niebla, que de acuerdo a los valores que se generan a través de los sensores aplica las funciones de control y el filtro de datos, que luego serán enviados a su almacenamiento, este filtrado previo pretende disminuir la redundancia de datos, ya que son enviados una menor cantidad de información al almacenamiento global, para su futuro análisis. La arquitectura del middleware local muestra en la figura 4.5.

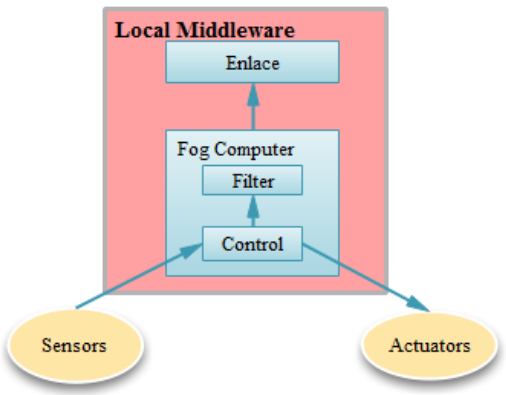


Figura 4. 5. Componentes del Middleware local.

El Middleware global está desarrollado para que sea la interfaz entre los clientes (granjas urbanas) y la capa de aplicación dentro de un sistema distribuido. Está desarrollado con tecnologías o servicios de programación Web, cuya interacción se lleva a cabo a través de los métodos de HTTP, el cual es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. En este caso utiliza los métodos POST y GET, como se puede visualizar en la figura 4.6.

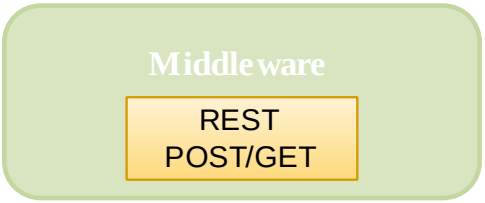


Figura 4. 6. Middleware global.

La capa de aplicación se encuentra en la nube, conteniendo la aplicación Web, que contiene las herramientas gráficas que presentan los datos en tiempo real, para apoyar y agilizar la toma de decisiones al usuario final, estos datos que son representados en la capa de aplicación se encuentran almacenados de forma centralizada en una base de datos que se encuentra en la capa de aplicación. Por lo que en el diseño de los requerimientos funcionales de la arquitectura se contempló que las características de la base de datos, debe ser capaz de almacenar hasta varios miles de millones de datos, debido a la naturaleza del sistema a desarrollar en donde utilizan una gran cantidad de variables (figura 4.7).

Figura 4. 7. Contenido de la nube.

4.2 Diseño de los instrumentos de validación para la arquitectura IoT

Para el instrumento de validación en software y de acuerdo a las necesidades presentadas en la agricultura de precisión en conjunto con la arquitectura propuesta, se establecieron los requerimientos de los cuales los más relevantes fueron que el software debería ser en tiempo real, aceptar *N* cantidad de conexiones para solicitud de servicios, utilizar herramientas de visualización para la ayuda en la toma de decisiones, que hiciera uso de la arquitectura de software REST, por su implementación en sistemas distribuidos, en especial sistemas IoT y su sintaxis universal; así como interactuar y almacenar grandes volúmenes de datos en forma distribuida con la capacidad de

escalabilidad lineal y disponibilidad, para su posterior análisis. Por lo cual se investigaron distintos tipos de tecnologías de desarrollo de software; las tecnologías analizadas que se podían utilizar para cumplir con las necesidades son LAMP (Linux, Apache, MySQL, PHP) y MEAN (MongoDB, Express, Angular, Node.js).

De acuerdo a las características de cada tecnología se optó por elegir MEAN, ya que **MongoDB** es una base de datos orientada a documentos, proporcionando un almacenamiento de datos potente, flexible y escalable. Tiene la capacidad de escalar con características de las bases de datos relacionales, como índices secundarios, consultas de rango y clasificación. MongoDB también tiene características útiles, como soporte integrado para agregación de estilo MapReduce e índices geoespaciales [29]. **Express** es un framework de desarrollo en JavaScript para crear aplicaciones web y está diseñado para Node.js [30], ofrece ruteo de URL con los métodos Get, Post, Put y Delete, y se le pueden integrar plantillas para la vista y así facilitar el uso del patrón de arquitectura de software MVC, para el diseño web se utiliza **Angular** [31] ya que para la aplicación del MVC en la parte de la vista facilita las pruebas de la aplicación y por sus características para la muestra de objetos dinámicos en tiempo real, la necesidad de aceptar miles de conexiones concurrentes para consumir los servicios web que van a ser proporcionados, se eligió **Node.js** [32], ya que está diseñado para crear aplicaciones que utilizan eventos asíncronos.

Una vez elegidas las herramientas con las cuales se iba a desarrollar el software para la implementar las capas propuestas en la arquitectura y los requerimientos, se procedió con el diseño general de los componentes que iban a formar el sistema en cuestión de software y las herramientas que lo iban a conformar, la distribución de las tecnologías que se usaran se muestra en la figura 4.8.

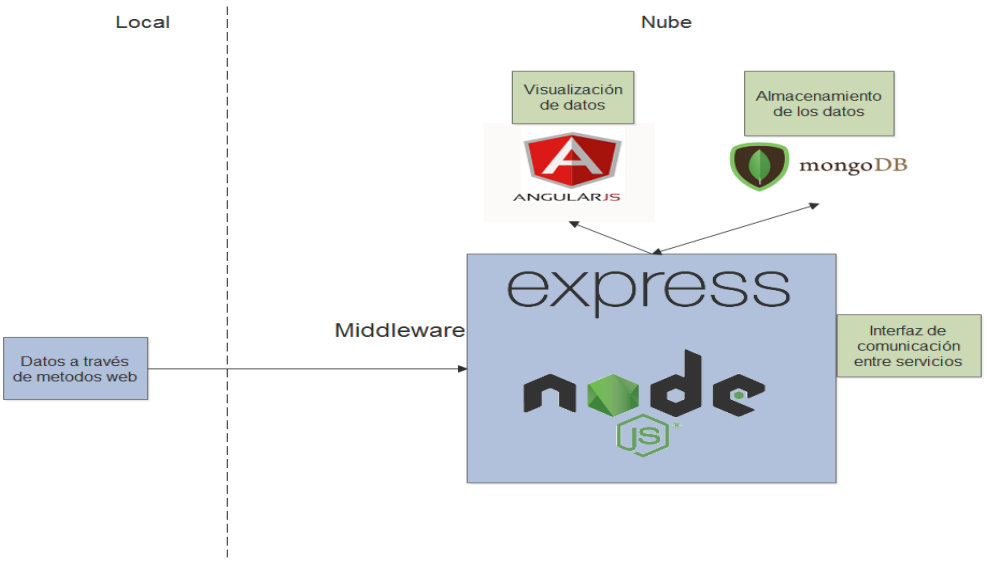


Figura 4. 8. Distribución de las tecnologías en software.

Para continuar con validación de la arquitectura IoT en la parte de hardware, se diseñó un instrumento de validación, el cual se decidió que sea un sistema hidropónico de interior, este tipo de sistemas son una alternativa al cultivo en tierra; las características que se planearon fueron que tuviera un sensor de temperatura y humedad relativa, riego automatizado, luces RGB configurada para la ayuda en el crecimiento del cultivo y una forma de disminuir la temperatura en el sistema hidropónico. El cultivo que producirá el sistema hidropónico serán lechugas, el boceto del diseño del sistema es el que se muestra en la figura 4.9.

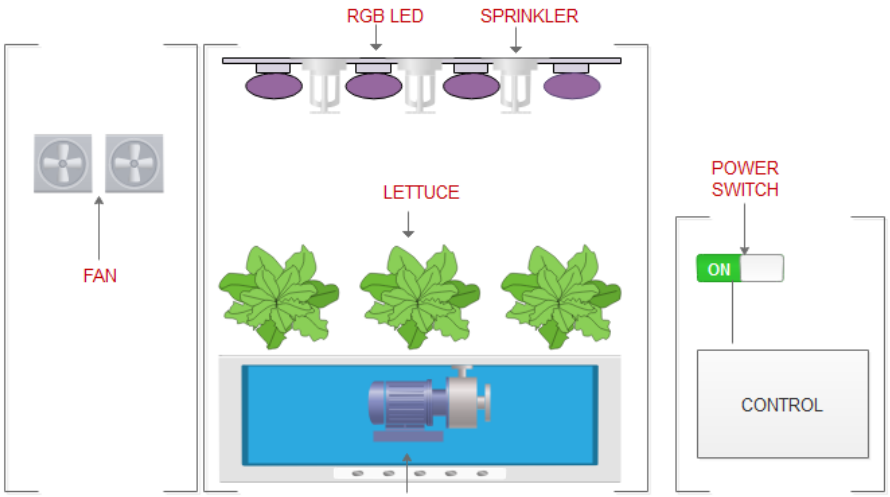


Figura 4. 9. Boceto del sistema hidropónico.

4.3 Desarrollo del instrumento de validación en software

La primera fase del desarrollo del software y distribución de los componentes fue totalmente de forma local como se muestra en la figura 4.10. En donde en la primera etapa, el middleware, la capa de aplicación y el almacenamiento quedaron de forma local, el middleware se desarrolló como un repositorio de servicios web para el funcionamiento general del sistema IoT, de forma siguiente se diseñaron los esquemas de la base de datos en mongoDB necesarios para los servicios desarrollados, de esta forma comenzó la programación del sistema, sobre un sistema operativo Ubuntu 16.04 en un equipo de cómputo con las características de 4gb de ram y un procesador con i5 segunda generación.

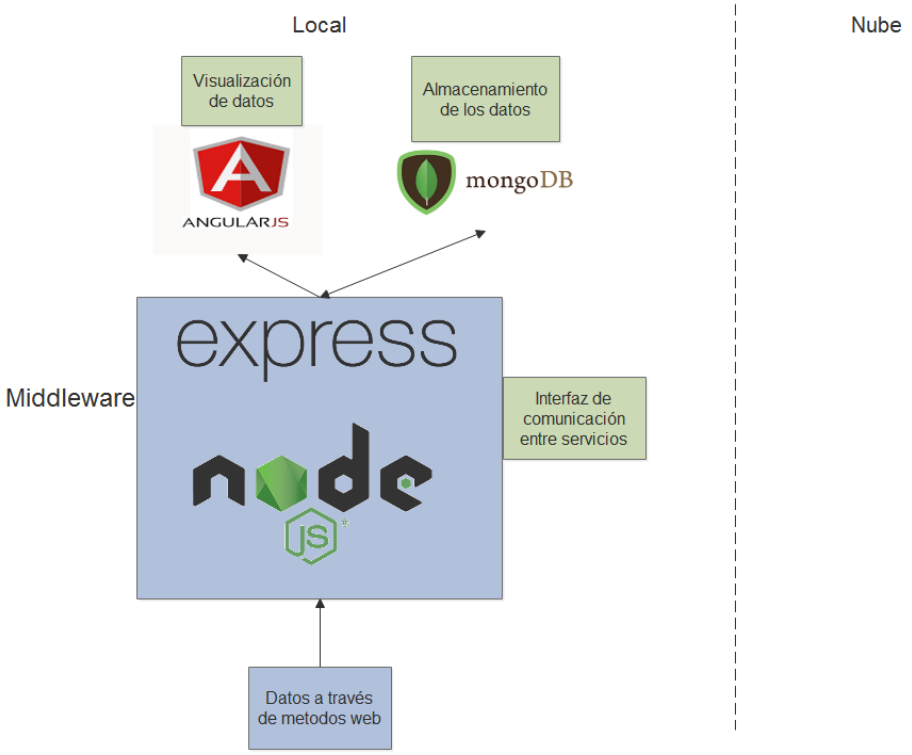


Figura 4. 10. Distribución de los componentes en la primera etapa.

Una vez desarrollado el software se hizo la simulación de registrar granjas urbanas y sub granjas a través de la vista del registro de una Urban Farm que se muestra en la figura 4.11.

Figura 4. 11. Registro de una Granja Urbana.

Posteriormente para el almacenamiento de los datos generados por las granjas urbanas se tienen que enviar a los valores de las variables a través del método POST, el nombre de las variables necesarias y su descripción de lo que almacenará se muestran en la tabla 4.1.

Variable	Descripción
urbanFarm	El nombre de la granja urbana a la cual pertenece la sub granja urbana.
subUrbanFarm	El nombre de la sub granja urbana a la cual pertenece la variable.
date	La fecha y hora en la cual se hace el registro del valor.
ipMicrocontroller	IP del microcontrolador a la cual se encuentra conectado el actuador que dicha variable puede activar.
sensorModel	Modelo del sensor del cual se captura la información.
dataType	El tipo de variable que captura el sensor, por ejemplo temperatura, humedad, etc.
data	El valor que está generando el sensor para ser almacenado.

Tabla 4. 1. Esquema para el almacenamiento de los datos.

Para el apoyo en la toma de decisiones se desarrolló un componente de software, el cual proporciona información de los datos de forma visual, a través de gráficas y tablas, el componente de visualización está desarrollado en Angular, que se encuentra en la parte de la vista del modelo arquitectónico de software Modelo-Vista-Controlador, los datos que se presentan son extraídos del almacenamiento que se encuentra en mongoDB. Un ejemplo de la representación en forma de gráfica de los datos se muestra en la figura 4.12.

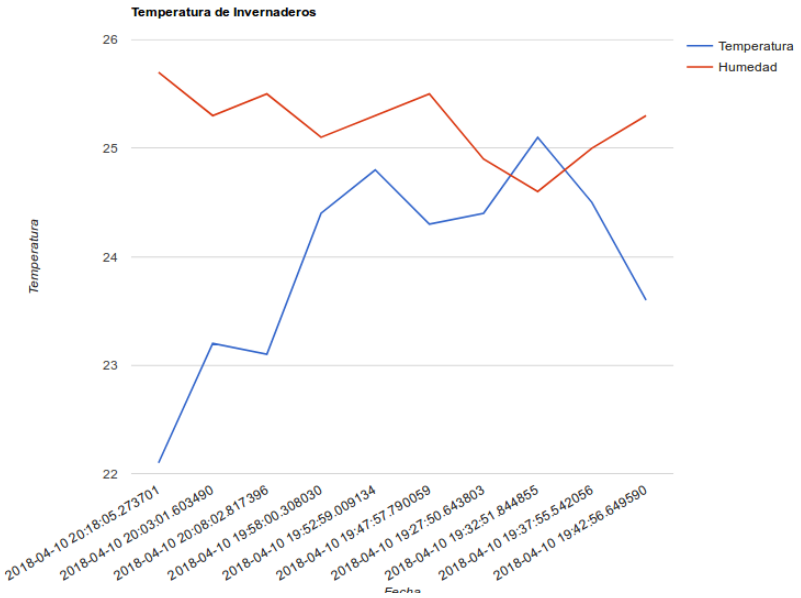
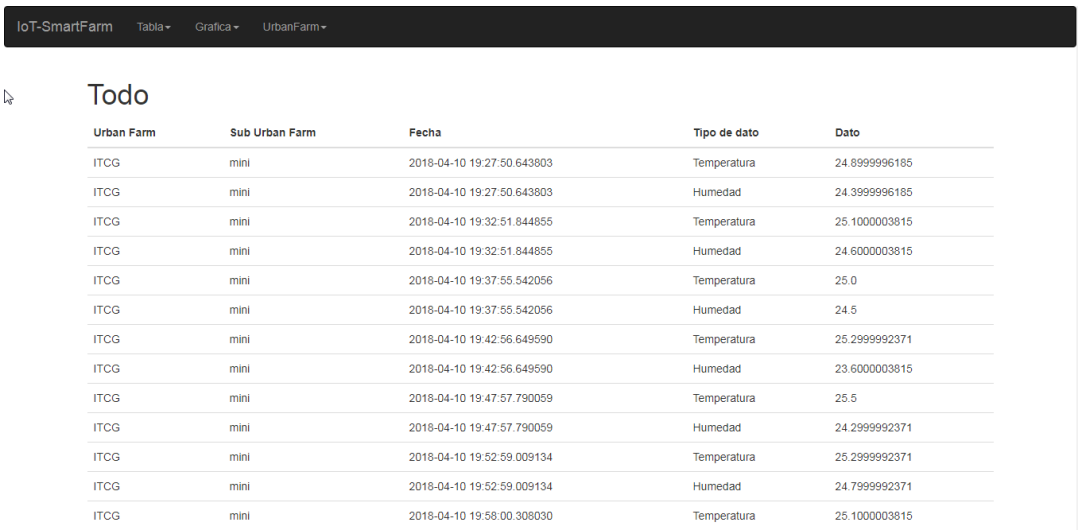


Figura 4. 12. Visualización de los datos en forma de gráficas.

Los datos también se pueden representar en forma de tabla para la ayuda en la toma de decisiones como se observa en la figura 4.13



Urban Farm	Sub Urban Farm	Fecha	Tipo de dato	Dato
ITCG	mini	2018-04-10 19:27:50.643803	Temperatura	24.8999996185
ITCG	mini	2018-04-10 19:27:50.643803	Humedad	24.3999996185
ITCG	mini	2018-04-10 19:32:51.844855	Temperatura	25.1000003815
ITCG	mini	2018-04-10 19:32:51.844855	Humedad	24.6000003815
ITCG	mini	2018-04-10 19:37:55.542056	Temperatura	25.0
ITCG	mini	2018-04-10 19:37:55.542056	Humedad	24.5
ITCG	mini	2018-04-10 19:42:56.649590	Temperatura	25.2999992371
ITCG	mini	2018-04-10 19:42:56.649590	Humedad	23.6000003815
ITCG	mini	2018-04-10 19:47:57.790059	Temperatura	25.5
ITCG	mini	2018-04-10 19:47:57.790059	Humedad	24.2999992371
ITCG	mini	2018-04-10 19:52:59.009134	Temperatura	25.2999992371
ITCG	mini	2018-04-10 19:52:59.009134	Humedad	24.7999992371
ITCG	mini	2018-04-10 19:58:00.308030	Temperatura	25.1000003815

Figura 4. 13. Representación de los datos en forma de tabla.

Después de tener desarrollados los componentes del software, el sistema IoT se migró a un servidor externo, esto a través de una plataforma de trabajo colaborativo con control de versiones llamada GitHub. Después de la migración, la distribución de las capas de la arquitectura se distribuyeron quedando un sistema IoT de nivel 5, donde la capa del middleware local, la capa física y la capa de enlace quedaron de forma local, la

capa del middleware global y la capa de la aplicación se encuentran en la nube, la distribución del sistema se observa en la figura 4.14

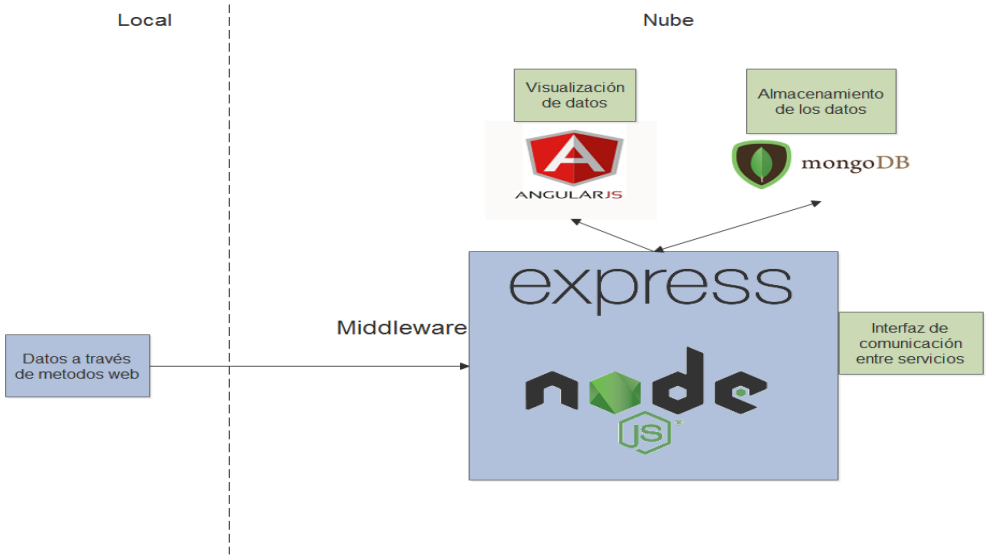


Figura 4. 14. Distribución de las tecnologías.

4.4 Desarrollo del instrumento de validación en hardware

Para el sistema hidropónico como un instrumento de validación de la arquitectura se integraron ventiladores para cumplir con la función de disminuir la temperatura, este es un actuador que se activa dependiendo del valor arrojado por el sensor DTH22, dicho sensor es alimentado por voltaje de corriente directa de 5 volts y obtiene información sobre la temperatura y humedad relativa del sistema hidropónico, el riego automatizado se conforma por una hidro bomba de corriente alterna de una fuerza de 0.1 HP, esta se activa cada cierto tiempo y para enviar agua a cuatro aspersores distribuidos para un riego uniforme, las luces son dos tiras de led RGB de 12 volts de corriente directa están configuradas para ser color violeta, son activadas una cantidad de horas al día, para la alimentación de energía se integró una fuente de 12 volts de 10 amperes. Para la función de automatización y control es desarrollado en una tarjeta RaspBerry Pi 3 ya que ofrece su programación en lenguajes de alto nivel y con las características que tiene [33], esta tarjeta también se encarga de ser el middleware local, es decir cumple con la función de filtro de datos y automatización del sistema hidropónico, la programación desarrollada para el filtro y control está en el lenguaje

python en la versión 2.7. El uso de la función de control para la variable de temperatura es un sistema de lazo cerrado, el diagrama de función se muestra en la figura 4.15.

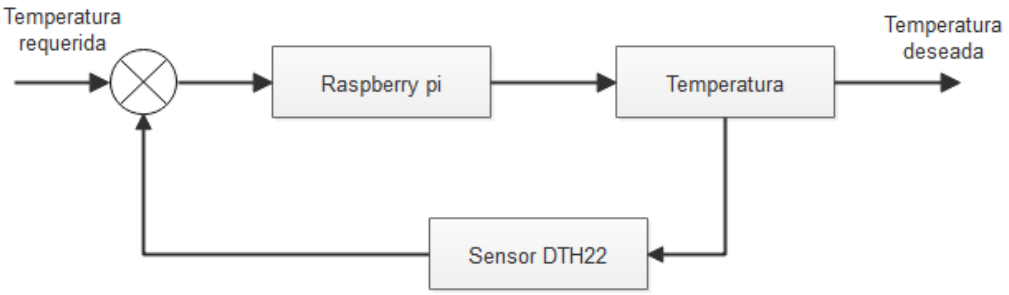


Figura 4. 15. Modelo de control para enfriamiento.

Para la automatización del riego se utilizó un sistema de control de lazo abierto que se muestra en la figura 4.16.

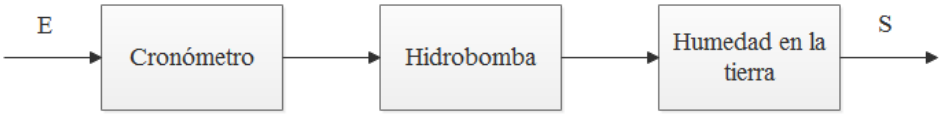


Figura 4. 16. Modelo de control para el riego.

El control de la iluminación está diseñado en un control de lazo abierto el cual se muestra en la figura 4.17.

Figura 4. 17. Modelo de control para la iluminación.

En la figura 4.18 se encuentra el resultado real del sistema hidropónico como un instrumento de validación para implementar la capa física, el cómputo en la niebla y enlace de la arquitectura propuesta, a dicho sistema hidropónico se nombró como "MiniUrbanFarm".

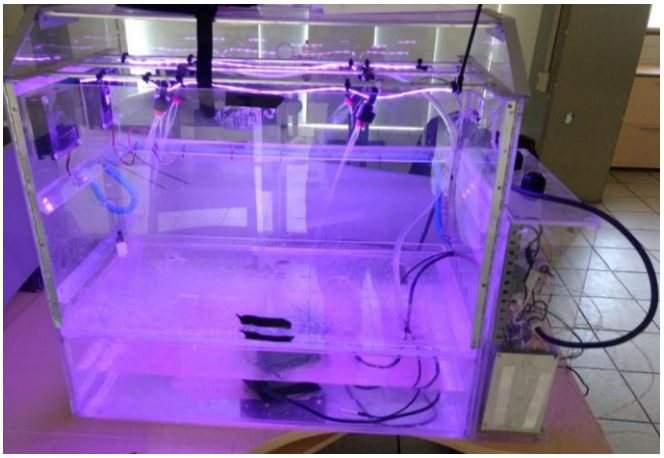


Figura 4. 18. Sistema hidropónico de interior.

En el Instituto Tecnológico de Ciudad Guzmán se cuenta con una estación meteorológica Davis Pro 2, la cual se muestra encerrada en un círculo en la figura 4.19, dicha estación capta información del clima en el exterior y almacena el valor de estas variables de forma local en un archivo de texto plano a través de un sistema propio de la estación meteorológica, dicho sistema se tiene instalado en una computadora, el valor de las variables se registran cada minuto, para la monitorización en tiempo real del clima, se desarrolló una librería en el lenguaje de programación Java para buscar y extraer el valor de las variables deseadas para su futura manipulación, esto a través de una depuración en el archivo de texto.

Figura 4. 19. Estación meteorológica DavisPro2.

La siguiente etapa de la validación de la arquitectura es el acoplamiento de los instrumentos de validación de software con los instrumentos de validación en hardware, la distribución real del sistema IoT con los componentes que lo conforman se muestran en la figura

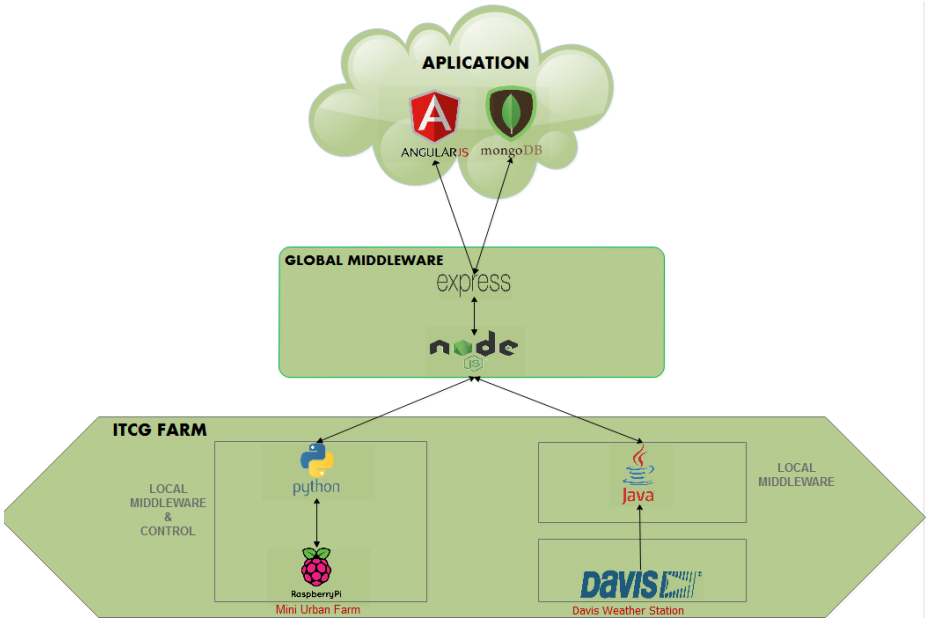


Figura 4. 20. Esquema y distribución de las tecnologías utilizadas.

4.5 Comparativa de tecnologías Web.

Para corroborar que las tecnologías en tiempo real que se utilizaron son las adecuadas se desarrolló el sistema con las mismas funciones pero con otro tipo de tecnologías las cuales fueron en LAMP (Linux, Apache, MySQL, PHP). La comparación realizada de las tecnologías implementadas se muestra en la figura 4.21.

Figura 4. 21. Comparativa de tecnologías.

La primera etapa realizada para hacer la comparativa es tener la misma cantidad y los mismos valores de datos en las dos tecnologías de almacenamiento, este paso se realizó con la exportación los datos de mongoDB a MySQL, la segunda parte fue desarrollar en PHP el middleware con las funciones para registrar y manipular datos en MySQL a través de los métodos GET y POST, para la visualización gráfica de los datos almacenados en MySQL se utilizó HighChart.

De acuerdo a la naturaleza del proyecto y el crecimiento que puede tener, una de las características más relevantes es que el tiempo de respuesta sea el mínimo al registrar datos, por eso la primera prueba realizada fue hacer solicitudes registrando de forma simultánea datos y cronometrando el tiempo de la respuesta. En el lenguaje de programación Java se desarrolló una librería para crear solicitudes por el método POST, posteriormente se realizaron hilos para simular la concurrencia de la solicitud del servicio que registra un dato, se realizaron pruebas en 10, 20, 50 y 100 datos. La tendencia que se muestra al incrementar el registro de los datos es que PHP ocupa más tiempo para responder como se observa en la figura 4.22.

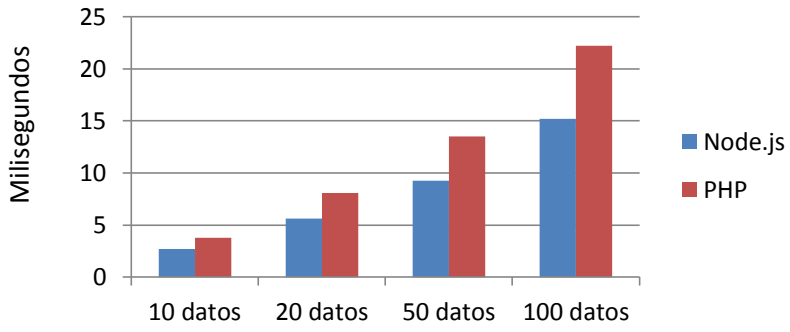


Figura 4. 22. Resultados de la prueba Petición-Respuesta.

Las siguientes pruebas fueron en el almacenamiento de los datos en la cual se cronometraron las acciones de respaldar y restaurar de forma total la base de datos, la consulta de todos los datos almacenados y una búsqueda específica, como se observa en los resultados obtenidos en la figura 4.23 MySQL tuvo un mejor desempeño, ya que al consultar toda la información hubo una notable diferencia en el tiempo de respuesta.

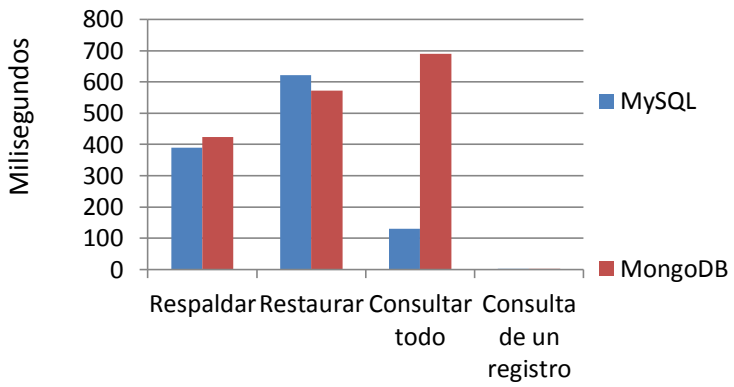


Figura 4. 23. Pruebas en las bases de datos.

Otra métrica de la comparativa es el peso que ocupa en disco el respaldo de la base de datos, de acuerdo al resultado mongoDB necesita más espacio para el almacenamiento del respaldo como se observa en la figura 4.24.

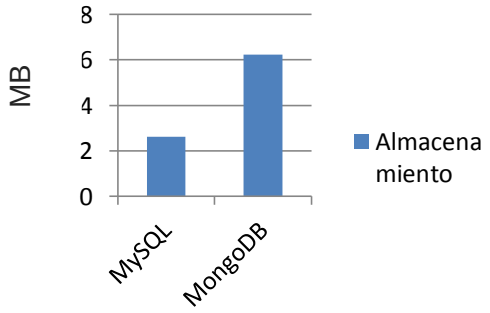


Figura 4. 24. Espacio necesario para el respaldo.

La última comparativa realizada fue el tiempo que requieren las tecnologías para mostrar de forma gráfica los datos almacenados, se hizo un límite de los datos a graficar de 10, 100, 1000 y 10000 y se cronometro el tiempo necesitado para cada cantidad de los datos, de acuerdo a los resultados que se presentan en la figura 4.25 Angular muestra una tendencia de necesitar menos tiempo para la graficación.

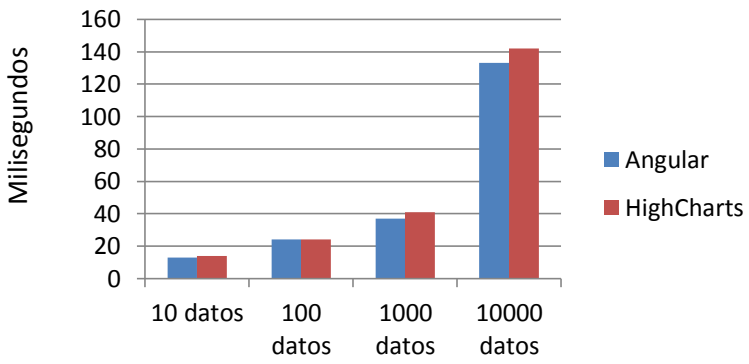


Figura 4. 25. Resultados en la graficación.

Los resultados muestran que las dos tecnologías comparadas necesitan muy poco tiempo. En algunos experimentos presentados LAMP tuvo mejores resultados pero para las características que se presentan en un sistema IoT en granjas urbanas se opta por seguir con las tecnologías MEAN.

4.6 Análisis de la arquitectura IoT propuesta.

Las funciones de automatización y control en una arquitectura IoT de tres capas se encuentran en la capa de aplicación, para la ejecución de las funciones de control y automatización los datos son enviados de forma directa a la capa de aplicación a través de la capa de enlace, creando un ciclo entre la capa física, de enlace y aplicación como se ilustra en la figura 4.26.

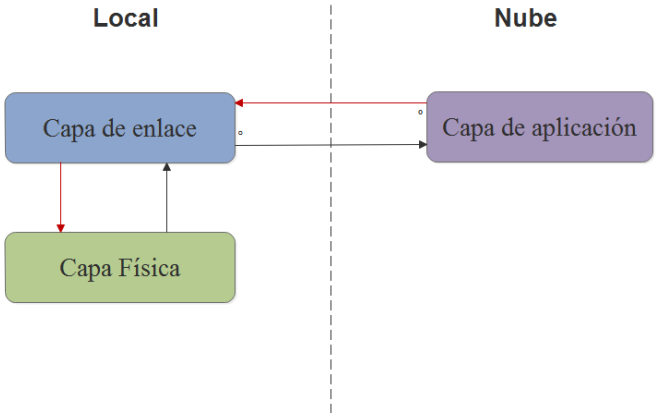


Figura 4. 26. Ciclo para control y automatización.

En la arquitectura propuesta se evita generar un ciclo que incluya la nube, para ello se realizan las funciones de control y automatización en la capa del cómputo en la niebla el cómo se ve en la figura 4.27, con la función de control también se disminuye la redundancia de los datos que llegan para un almacenamiento global ya que si un valor es repetido el computo en la niebla se encarga de hacer el filtrado de ese dato.

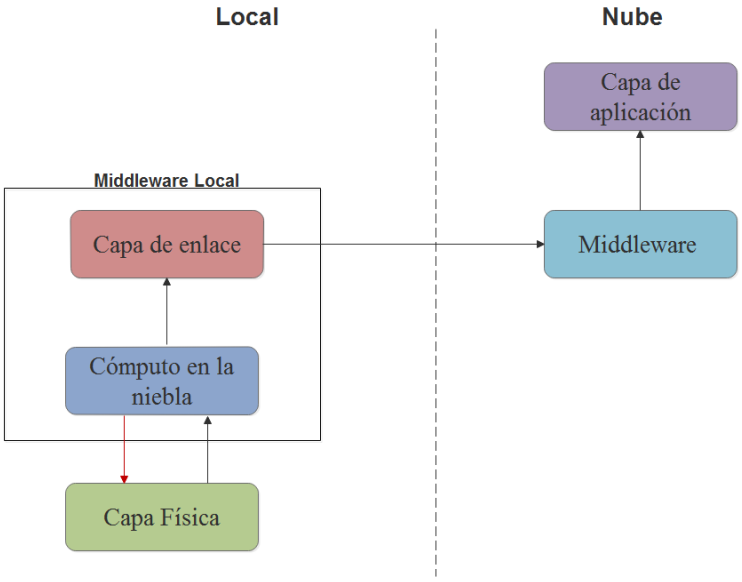


Figura 4. 27. Ciclo para automatización y control.

Conclusiones y trabajo futuro

En el presente trabajo de investigación, se presentó una arquitectura del Internet de las Cosas, la cual fue diseñada para cubrir las necesidades que se presentan en el dominio de la agricultura de precisión, la validación de dicha arquitectura fue con la implementación de las capas propuestas, donde en la capa física, se diseñó el sistema hidropónico "MiniUrbanFarm" y el acoplamiento de una "estación meteorológica", marca Davis pro2.

En la capa del middleware local para los instrumentos de validación estuvieron desarrollados en los lenguajes de programación Python y Java, el middleware global se desarrolló con tecnologías de programación web en tiempo real y para la agricultura de precisión, para implementar la capa de aplicación se desarrolló una herramienta para la visualización de datos en tiempo real e historial de los mismos. Con la implementación de la arquitectura propuesta se demostró que se facilita la integración de más granjas urbanas y que se disminuye la redundancia de los datos por el uso del cómputo en la niebla. También brinda un apoyo para las tomas de decisiones con las gráficas que se generan a partir de los datos almacenados. El uso de las herramientas de desarrollo de software web como lo es MEAN, agiliza el desarrollo de un sistema IoT debido a la utilización de un patrón arquitectónico de software como lo es el MVC.

Como trabajo futuro y con las ventajas que presenta MongoDB al ser una base de datos documental se deberá implementar técnicas de analítica de datos como MapReduce para darle un mayor valor a la información adquirida. Así como añadir más granjas urbanas localizadas en otros puntos del país.

Bibliografía

[1] Gay, C., Estrada, F., Conde, A. C., & Eakin, H. (2004). Impactos potenciales del Cambio Climático en la agricultura: escenarios de producción de café para el 2050 en Veracruz (México).

[2] Moran Alonso, N. (2010). Agricultura urbana: un aporte a la rehabilitación integral. *Papeles de relaciones ecosociales y cambio global*, (111), 99-111.

[3] LA FAO, U. C. E. (1990). Organización de las Naciones Unidas para la Alimentación y la Agricultura.

[4] Ortega, R., Flores, L., INIA, C. Q., de Recursos Naturales, D., & Ambiente, M. (1999). Agricultura de Precisión: Introducción al manejo sitio-específico. *Ministerio de Agricultura, Instituto de investigaciones agropecuarias. CRI Quilamapu.(Chile)*, 13-46.

[5] J. Boman, J., Taylor, J., & Ngu, A. H. (2014, October). Flexible IoT middleware for integration of things and applications. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on* (pp. 481-488). IEEE.

[6] Banouar, Y., Reddad, S., Diop, C., Chassot, C., & Zyane, A. (2015, November). Monitoring solution for autonomic Middleware-level QoS management within IoT systems. In *Computer systems and applications (AICCSA), 2015 IEEE/ACS 12th international conference of* (pp. 1-8). IEEE.

[7] Fan, C., Wen, Z., Wang, F., & Wu, Y. (2011). A middleware of internet of things (iot) based on Zigbee and RFID.

[8] Tortonesi, M., Michaelis, J., Morelli, A., Suri, N., & Baker, M. A. (2016, June). SPF: an SDN-based middleware solution to mitigate the IoT information explosion. In *Computers and Communication (ISCC), 2016 IEEE Symposium on* (pp. 435-442). IEEE.

[9] Bonino, D., Alizo, M. T. D., Alapetite, A., Gilbert, T., Axling, M., Udsen, H., ... & Spirito, M. (2015, August). Almanac: Internet of things for smart cities. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on* (pp. 309-316). IEEE.

[10] Pires, P. F., Cavalcante, E., Barros, T., Delicato, F. C., Batista, T., & Costa, B. (2014, August). A platform for integrating physical devices in the Internet of Things. In *2014 12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*(pp. 234-241). IEEE.

[11] Sekiyama, M., Kim, B. K., Irie, S., & Tanikawa, T. (2015, October). Sensor data processing based on the data log system using the portable IoT device and RT-Middleware. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2015 12th International Conference on* (pp. 46-48). IEEE.

[12] Vatari, S., Bakshi, A., & Thakur, T. (2016, May). Green house by using IOT and cloud computing. In *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on* (pp. 246-250). IEEE.

[13] Chen, N., Yang, Y., Li, J., & Zhang, T. (2017, October). A Fog-based service enablement architecture for cross-domain IoT applications. In *Fog World Congress (FWC), 2017 IEEE* (pp. 1-6). IEEE.

[14] Chen, Y. C., Chang, Y. C., Chen, C. H., Lin, Y. S., Chen, J. L., & Chang, Y. Y. (2017, May). Cloud-fog computing for information-centric Internet-of-Things applications. In *Applied System Innovation (ICASI), 2017 International Conference on* (pp. 637-640). IEEE.

[15] Kum, S. W., Moon, J., & Lim, T. B. (2017, September). Design of fog computing based IoT application architecture. In *Consumer Electronics-Berlin (ICCE-Berlin), 2017 IEEE 7th International Conference on* (pp. 88-89). IEEE.

[16] Vatari, S., Bakshi, A., & Thakur, T. (2016, May). Green house by using IOT and cloud computing. In *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on* (pp. 246-250). IEEE.

[17] Moreno Flores, O. (2007). Agricultura Urbana: Nuevas Estrategias de Integración Social y Recuperación Ambiental en la Ciudad.

[18] Mougeot, L. J. (2006). *Cultivando mejores ciudades: agricultura urbana para el desarrollo sostenible*. IDRC.

[20] National Research Council (US). Transportation Research Board. Committee for a Study of Transportation, & Sustainable Environment. (1997). *Toward a sustainable future: addressing the long-term effects of motor vehicle transportation on climate and ecology* (Vol. 251). National Academy Press.

[21] Ashton, K. (2009). That 'internet of things' thing. *RFID journal*, *22*(7), 97-114.

[22] Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, *1*(2011), 1-11.

[23] Bahga, A., & Madisetti, V. (2014). *Internet of Things: A hands-on approach*. Vpt.

[24] Puder, A., Römer, K., & Pilhofer, F. (2011). *Distributed systems architecture: a middleware approach*. Elsevier.

[25] Fensel, D., Lausen, H., Polleres, A., De Bruijn, J., Stollberg, M., Roman, D., & Domingue, J. (2006). *Enabling semantic web services: the web service modeling ontology*. Springer Science & Business Media.

[26] Webber, J., Parastatidis, S., & Robinson, I. (2010). *REST in practice: Hypermedia and systems architecture*. " O'Reilly Media, Inc.".

[26] Rogers, P. (2005). Ingeniería de Software un Enfoque Práctico. *Editorial McGraw-Hill, Madrid*.

[27] Sommerville, I. (2005). Ingeniería del software séptima edición. *Madrid: PearsonEducation SA*.

[28] Ogata, K., & Yang, Y. (2002). *Modern control engineering* (Vol. 4). India: Prentice hall.

[29] Chodorow, K. (2013). *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. " O'Reilly Media, Inc.".

[30] Cantelon, M., Harter, M., Holowaychuk, T. J., & Rajlich, N. (2014). *Node. js in Action* (pp. 17-20). Greenwich: Manning.

[31] Darwin, P. B., & Kozlowski, P. (2013). *AngularJS web application development*. Packt Publ..

[32] Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), 80-83.

[33]Chin, S., & Weaver, J. (2015). *Raspberry Pi with Java: Programming the Internet of Things (IoT)(Oracle Press)*. McGraw Hill Professional.

Glosario

HTTP. Protocolo de comunicación que permite las transferencias de información en la World Wide Web.

Interfaz. Conexión funcional entre dos sistemas, programas, dispositivos o componentes de cualquier tipo, que proporciona una comunicación de distintos niveles permitiendo el intercambio de información.

Periurbano. Son espacios que se sitúan en los alrededores de una ciudad y que, aunque no se emplean para el desarrollo urbano, tampoco se usan para actividades rurales.

Plantas. Una planta puede ser una parte de un equipo, tal vez un conjunto de los elementos de una máquina que funcionan juntos, y cuyo objetivo es efectuar una operación particular. En este libro se llamará planta a cualquier objeto físico que se va a controlar (como un dispositivo mecánico, un horno de calefacción, un reactor químico o una nave espacial).

RFID. Es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transponedores que utilizan identificación por radiofrecuencia

RGB. RGB es un modelo de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores de luz primarios.

ZigBee. Es un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área persona.