Diseño e implementación de un lenguaje especifico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas Ciberfísicos (CPS)

Méndez Rodríguez, Juan Sebastián y Pilco Benavides, Bryan Alexander

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Alulema Flores, Darwin Omar

23 de Junio del 2021

Curiginal

Document Information

Analyzed document	TESIS_Mendez_Pilco.pdf (D109319641)
Submitted	6/20/2021 12:46:00 AM
Submitted by	Alulema Flores Darwin Omar
Submitter email	doalulema@espe.edu.ec
Similarity	2%
Analysis address	doalulema.espe@analysis.urkund.com

Firmado electrónicamente por:
DARWIN OMAR ALULEMA FLORES

Sources included in the report

W	URL: https://repositorio.espe.edu.ec/bitstream/21000/13474/1/T-ESPE-057380.pdf Fetched: 11/24/2019 10:36:22 AM	3
W	URL: https://docplayer.es/63577895-Titulo-autores-director-codirector-ingenieria-de-software-mda-pim-psm-uml-uml-profile-ocl-atl-framework-spring.html Fetched: 7/8/2020 6:04:08 PM	1
SA	**Universidad de las Fuerzas Armadas ESPE / PROYECTO_TITULACIÓN_MG_V10 (1) (Autoguardado)FINAL.docx** Document PROYECTO_TITULACIÓN_MG_V10 (1) (Autoguardado)FINAL.docx (D47309301) Submitted by: omiguelgp@hotmail.com Receiver: masoasti.espe@analysis.urkund.com	1
W	URL: http://sedici.unlp.edu.ar/bitstream/handle/10915/26667/INFORMATICA%2BPons-Giandini-Perez.pdf.txt%3Bjsessionid%3D3A8FD7418B2E253DE2F4F259BADE48FF%3Fsequence%3D2 Fetched: 1/11/2021 9:51:43 AM	1
SA	**Memoria_TFM_SantiagoJacome.pdf** Document Memoria_TFM_SantiagoJacome.pdf (D14890770)	2
W	URL: https://docplayer.es/9503923-Escuela-superior-de-ingenieria.html Fetched: 7/6/2020 7:04:34 PM	1
SA	**TT-JuanJumbo_Versión_Final.pdf** Document TT-JuanJumbo_Versión_Final.pdf (D99702548)	1
W	URL: https://docplayer.es/4114730-Un-analisis-critico-de-la-aproximacion-model-driven-architecture.html Fetched: 7/6/2020 6:57:28 PM	1
SA	**submission.docx** Document submission.docx (D62801079)	1
SA	**Escrito tesis5.docx** Document Escrito tesis5.docx (D51076136)	1
W	URL: https://dspace.unl.edu.ec/jspui/bitstream/123456789/23360/1/AlexNixon_SalinasGranda.pdf Fetched: 5/29/2021 4:31:00 AM	1

DEPARTAMENTO DE ELÉCTRIA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Diseño e implementación de un lenguaje específico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas Ciberfísicos (CPS)**" fue realizado por los señores **Méndez Rodríguez, Juan Sebastián y Pilco Benavides, Bryan Alexander** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 21 de junio del 2021

Firmado electrónicamente por:
DARWIN OMAR ALULEMA FLORES

.....

Ing. Alulema Flores, Darwin Omar

C. C. 1002493334

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRIA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Nosotros, **Méndez Rodríguez, Juan Sebastián y Pilco Benavides, Bryan Alexander**, con cedulas de ciudadanía n° 1803596392 y n° 1725024358, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Diseño e implementación de un lenguaje específico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas Ciberfísicos (CPS)** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 21 de junio del 2021

Méndez Rodríguez, Juan Sebastián
C.C.: 1803596392

Pilco Benavides, Bryan Alexander
C.C.: 1725024358

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRIA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Nosotros, **Méndez Rodríguez, Juan Sebastián y Pilco Benavides, Bryan Alexander,** con cédulas de ciudadanía n° 1803596392 y n° 1725024358, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un lenguaje específico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas Ciberfísicos (CPS)** en el Repositorio institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 21 de junio del 2021

.....

Méndez Rodríguez, Juan Sebastián

C.C.: 1803596392

.....

Pilco Benavides, Bryan Alexander

C.C.: 1725024358

Dedicatoria

Este trabajo lo dedico a mis padres y a mis hermanos que siempre han estado para mí en los buenos y malos momentos, por ser la mejor compañía en momentos difíciles, por darme el valor de seguir adelante y superarme a pesar de cualquier problema que se me presente.

Juan Sebastián Méndez Rodríguez

Dedicatoria

Dedico de manera muy especial este proyecto a mis padres Israel Pilco y Mercy Benavides quienes siempre han sabido guiarme y apoyarme a lo largo de toda mi vida, por haberme forjado como la persona que soy.

A mi hermano Brandon por estar siempre conmigo en los buenos y malos momentos, por poder contar contigo siempre.

Bryan Alexander Pilco Benavides

Agradecimiento

Agradezco a mis padres y hermanos, porque cada uno de ustedes son un ejemplo a seguir, ya sea por su carisma, por ser perfeccionistas en lo que hacen, por dar todo por la familia, y por mucho más y es así que por ustedes puedo seguir adelante y puedo culminar esta etapa de mi vida. A mi sobrina por ser la luz que alegra mis días. A mi compañero Bryan, por ser un gran amigo y un excelente compañero, sin ti no hubiéramos logrado esto. A mi tutor de tesis, por ser un gran guía, por apoyarnos en toda esta etapa y hacer que demos lo mejor hasta el último momento. A mi mejor amigo Francisco porque a pesar de la distancia siempre estuviste para darme cualquier consejo, por hacer que sueñe en grande, por como mi hermano. A mis amigos por hacerme reír con sus boberas y por las cervezas bien frías mientras hacíamos los proyectos. A cada persona que he conocido en el transcurso de la universidad porque de ellos aprendí muchas cosas. Por último, pero no menos importante, me quiero agradecer a mí por creer en mí, por realizar todo este duro trabajo, me quiero agradecer por nunca renunciar, por dar más de que recibo y por ser siempre el mismo.

Juan Sebastián Méndez Rodríguez

Agradecimiento

Agradezco infinitamente a mis padres por brindarme sus consejos y todo su apoyo incondicional siempre que lo he necesitado, gracias por enseñarme valores que me van a servir a lo largo de mi vida y ayudarme a conseguir una meta más. A mi hermano por todo su apoyo y cariño. A toda mi familia en general, abuelitos, tíos, primos ya que siempre he podido contar con ellos para cualquier cosa que necesite y sé que siempre lo podré hacer. A mi compañero de tesis Sebas, por ser un gran amigo y por todo el compromiso y dedicación de su parte para poder sacar este trabajo adelante. A todos los amigos de la universidad y de la carrera por todos los buenos momentos que hemos compartido a lo largo de este camino. Finalmente agradezco al ingeniero Darwin Alulema nuestro tutor, por el tiempo, dedicación y paciencia que ha tenido hacia nosotros a lo largo de estos meses para poder culminar con la elaboración del proyecto.

Bryan Alexander Pilco Benavides

Contenido

Índice de tablas

Índice de Figuras

Resumen

Dados los avances de los Sistemas Ciberfísicos y servicios web, más y más cosas están en capacidad para comunicarse con el internet y así transmitir y almacenar información, donde estos se han utilizado en diferentes campos como la industria, la domótica, la salud, la agroindustria, entre otros. Debido a estos grandes cambios tecnológicos, se presentan desafíos para los profesionales al momento de integrar varios procesos donde existen diferentes protocolos de comunicación, lenguaje de programación, procesamiento y almacenamiento de datos. Por lo cual se propone una arquitectura, un metamodelo, un editor gráfico, y un generador de código semi automático, que permite al desarrollador implementar distintos CPS sin la necesidad de tener un amplio conocimiento de las plataformas de hardware y software. Esta propuesta abarca objetos físicos (controladores, sensores y actuadores), protocolos de comunicación que integren mensajería MQTT y servicios REST, y diversas orquestaciones que maneja la lógica a tomar por los CPS. Para comprobar el funcionamiento de nuestro DSL se han realizado varias pruebas que van desde plantearse un escenario de smart home, hasta realizar encuestas para medir la aceptación por parte de un grupo de usuarios.

PALABRAS CLAVES:

- **LENGUAJE DE DOMINO ESPECIFICO**
- **EVENTOS COMPLEJOS**
- **SISTEMAS CIBERFÍSICOS**
- **ORQUESTACIÓN**
- **ARQUITECTURA**

Abstract

Given the advances of Cyber-Physical Systems and web services, more and more things are able to communicate with the internet and this transmit and store information, where these have been used in different fields such as industry, home automation, health, agribusiness, among others. Because of these great technological changes, there are challenges for professionals when integrating various processes where there are different communication protocols, programming language, data processing and storage. Therefore, an architecture, a metamodel, a graphic editor, and a semi-automatic code generator are proposed, which allows the developer to implement different CPS without the need to have extensive knowledge of hardware and software platforms. This proposal covers physical objects (controllers, sensors and actuators), communication protocols that integrate MQTT messaging and REST services, and various orchestrations that handle the logic to be taken by the CPS. To verify the operation of our DSL, several tests have been carried out, ranging from considering a smart home scenario, to conducting surveys to measure acceptance by a group of users.

KEYWORDS:

- **DOMAIN-SPECIFIC LANGUAGE**
- **COMPLEX EVENTS**
- **CYBER-PHYSICAL SYSTEMS**
- **ORCHESTRATION**
- **ARCHITECTURE**

Capítulo I: Marco Metodológico

Antecedentes

Con el avance de la tecnología, más y más “cosas” comienzan a conectarse entre sí, como: parlantes inteligentes, relojes inteligentes, refrigeradores inteligentes, aspiradoras inteligentes, luces inteligentes, etc. En este nuevo campo de dispositivos, IoT (Internet of Things) permite la integración de objetos físicos propios de las Tecnologías de Operación (OT) con servicios web propios de las Tecnologías de la Información (IT), para automatizar y mejorar la eficiencia de los procesos (Espada, Martínez, & Pelayo, 2011). El IoT se utiliza para entornos inteligentes en varios campos. Con respecto al campo de la salud, estas tecnologías proporcionan métodos efectivos y eficientes para mejorar los entornos de vida y el bienestar (Rahmani, y otros, 2018). En el ámbito del hogar el Internet de las Cosas tiene como objetivo mejorar la eficiencia energética, la seguridad, la comodidad y proporcionar sistemas de audio/ video/ control, todo esto mediante la comunicación de sensores y actuadores instalados en un hogar (Miguez, Fernández, Fraga, & Castedo, 2018).

Otro concepto que actualmente ha tomado realce junto con el IoT es el de los CPS (Sistemas Ciberfísicos), los cuales son visiones holísticas de las perspectivas de la informática y la comunicación. Sin embargo, CPS tiene raíces en el control, la informática, los sistemas en tiempo real y las redes de sensores. Mientras que IoT tiene raíces en las redes de comunicación y comunicación inalámbrica. Además, CPS enfatizan los sistemas híbridos y la verificación formal de sistemas dinámicos, e IoT enfatizan los protocolos de comunicación.

El objetivo de los Sistemas Ciberfísicos es percibir y comprender los cambios en el entorno físico que los rodea, analizar estos datos y tomar decisiones rápidas e

inteligentes para responder a dichos cambios de forma autónoma (Shi, Wan, & Suo, 2011). Por lo que se basan en arquitecturas orientadas a eventos lo que les permite actuar en tiempo real. Un evento se considera a cualquier cosa significativa que ocurre o que se considera que puede llegar a ocurrir y es significativo debido a que puede llegar a afectar alguna acción, los eventos pueden ser primitivos o también llamados simples y también pueden ser eventos complejos. Los eventos simples son eventos que no resumen, representan o denotan un conjunto de otros eventos como por ejemplo el cambio de temperatura de un sensor, el cambio de estado de un switch, el cambio de hora en un reloj, etc. Por otro lado un evento complejo es la agrupación de varios eventos simples previamente procesados para su análisis con el objetivo de detectar una situación relevante para el entorno, por ejemplo, se tendría un sensor de temperatura en la entrada de un laboratorio el cual activa una alarma si detecta una sobre temperatura, pero por otro lado al momento de llegar la hora en la que todos los empleados salen, al haber una gran cantidad de personas acumuladas se podría llegar a activar la alarma erróneamente, por lo cual para evitar esto se tendría que procesar dos eventos simples (temperatura, tiempo, humedad, presencia) para así, una vez validados evitar que la alarma se active equivocadamente, para lo cual se genera un evento complejo que a su vez puede ser parte de otros eventos complejo (Lee, Bagheri, & Kao, 2015).

Debido a que en los últimos años el avance de la tecnología ha permitido que el costo de dispositivos electrónicos se abarate significativamente, esto ocasiona que en una aplicación CPS puedan existir varios tipos de dispositivos conectados entre sí trabajando a una alta velocidad de comunicación lo cual ocasiona una sobrecarga de información y que las decisiones tomadas para responder a los cambios de entorno no sean las más acertadas, para lo cual se utiliza el Procesamiento de Eventos Complejos (CEP), el cual se encarga de procesar eventos primitivos o complejos y

obtener información valiosa de ellos, para así poder responder a estos lo más rápido posible. Por lo cual el motor CEP tiene un papel clave en el procesamiento de datos CPS (Cao, Li, & Wang, 2013).

En los sistemas CEP actuales los usuarios definen cuáles serán las reglas que van a procesar distintos eventos primitivos recibidos desde un entorno monitorizado y así poder detectar diferentes fenómenos compuestos combinando distintos eventos simples u otros eventos usando un conjunto de operadores de composición (Zacheilas, Kalogeraki, Zygouras, Panagiotou, & Gunopulos, 2015), sin embargo pese a la gran cantidad de motores CEP existentes, uno de los mayores problemas que se presenta es lograr una buena escalabilidad mientras se mantiene un alto rendimiento.

Un mejor nivel de automatización en CPS y CEP requiere de: (1) la reacción a los eventos del sensor del mundo físico; y (2) el procesamiento de datos debe ser completamente automatizado. Los sistemas actuales en los que se ejecutan los procesos han sido diseñados con un enfoque en los flujos de trabajo basados en la web que involucran Servicios Web, actividades humanas de alto nivel y procesos organizacionales (Seiger, Huber, & Schlegel, 2016), lo cual no permite una integración con otros sistemas ni actuar ante eventos en tiempo real. Para solucionar estos inconvenientes, propuestas como los de la Web de las Cosas (WoT) proponen emplear arquitecturas REST, las cuales permiten escalabilidad y robustez que tanto se requiere en sistemas con miles de nodos posibles como ocurrirá en los CPS.

Modelar los CPS utilizando principios REST orientados a la web puede ser una forma de comenzar a desarrollar una infraestructura global de objetos interconectados y fomentar el desarrollo de aplicaciones CPS escalables y robustas. La idea básica es considerar los objetos inteligentes como pequeños servidores que implementan aplicaciones utilizando hipermedia. Construir los CPS en torno al paradigma REST y modelarlo de acuerdo con los conceptos web permite reutilizar toda la experiencia

adquirida en las décadas de construcción de la web. La Web de las cosas (WoT) proporciona una capa de aplicación que simplifica la creación de CPS. Al llevar los patrones de la web al CPS, será posible crear aplicaciones robustas a largo plazo y construir una infraestructura diseñada para escalar indefinidamente en el tiempo. Sin embargo, dada la naturaleza REST de estos protocolos orientados a la web, adoptarán los mismos patrones, lo que dará como resultado interoperabilidad entre la web y CPS (Eassa, Elhoseny, El-Bakry, & Salama, 2018).

El desarrollo actual que ha tenido la Ingeniería Dirigida por Modelos (MDE) la convierte en uno de los principales candidatos para lograr la integración de tecnologías y acelerar el proceso de desarrollo. Esto debido a que, mediante un proceso de abstracción, un experto de dominio de CPS puede generar un Lenguaje Específico de Dominio (DSL) el cual abstraiga las características de los sistemas y por medio de una serie de transformaciones de modelos permita la generación de las instancias concretas. Un DSL es un lenguaje de programación dedicado a resolver un problema en particular, representar un problema específico y proveer técnicas para solucionar el mismo. Existen varios tipos de estos lenguajes tales como Verilog para modelado de sistemas electrónicos, R y S que están enfocados a la estadística, Matlab y Máxima para las matemáticas, se ha desarrollado un DSL con el fin de evaluar el aprendizaje de lenguas extranjeras mediante técnicas 3D, etc. La utilización eficiente de dichos DSL pasa por contar con herramientas o espacios, generalmente a modo de editores gráficos, que faciliten el trabajo del modelador. De esta forma, a lo largo de los últimos años, y con la intención de aliviar la complejidad inherente al desarrollo de este tipo de herramientas, dando así algunas de las propuestas que han aparecido para el desarrollo de editores gráficos para DSL visuales.

Justificación e importancia

La Arquitectura Orientada a Servicios no es un concepto nuevo, debido a que procede de la década de los 90, en si el SOA es un concepto de arquitectura de software que define la implementación de servicios para ofrecer soporte a ciertos requisitos, la cual se ocupa del diseño y desarrollo de sistemas distribuidos y es un potente aliado en el momento de realizar a cabo la administración de gigantes volúmenes de datos, datos en la nube y jerarquías de datos. Lo nuevo en esta situación es pasar del paradigma de una arquitectura monolítica a una arquitectura basada en microservicios, con lo que la arquitectura REST permite los procesos de interoperabilidad, robustez y escalabilidad, lo cual es empleado en los sistemas recientes como Amazon AWS, Google Cloud o Netflix.

Además, la Arquitectura Orientada por Eventos pertenece a los más actuales pasos en la evolución de los microservicios, es un patrón de diseño el cual permite a un conjunto de sistemas comunicarse entre sí de manera reactiva por medio de la publicación y el consumo de eventos, los cuales tienen la posibilidad de interpretar como cambios de estado de objetos, lo cual permite a los CPS actuar en tiempo real con base a dos posibles esquemas: a) basado en colas de mensajes y b) basado en tópicos. Los cuales en la actualidad son empleados en sistemas empresariales como los de BMW, MasterCard, Netflix, o en sistemas PaaS (Plataforma como Servicio) como los de IBM Whatson, Microsoft Azure.

Los analistas Blueberry Consultants alegan que el futuro de los negocios digitales está una convergencia digital, inteligente y expandida compuesta por aplicaciones dirigidas por eventos, IoT, Cloud, Blockchain, almacenamiento en memoria e IA (inteligencia artificial). Gartner concuerda con este pronóstico, y garantiza que para 2020 los usuarios demandarán que 8% de las aplicaciones digitales de los negocios sean en tiempo real y con un enfoque con base en eventos. Es allí que nace la necesidad de

continuar trabajando en los CPS para disponer de arquitecturas que busquen el desarrollo de la Industria 4.0.

Actualmente los Sistemas Ciberfísicos cuentan con dispositivos-herramientas de uso industrial que fabrica la compañía, cuentan ahora con un ahorro del 30% en los tiempos de desempeño, y del 50% en tiempo y esfuerzo para calcular los valores tecnológicos o la búsqueda de información fundamental, lo cual sumado con las capacidades de los sistemas de administración de TI, la unión de las plataformas de suministros, el procesamiento de enormes volúmenes de datos, el Cloud Computing, mejorarán el rendimiento de cada uno de los procesos.

El advenimiento de la Industria 4.0 es un hecho que más temprano que tarde se implementará. De ahí la necesidad de disponer de arquitecturas que permitan la integración de los CPS con toda la infraestructura desplegada en la Web. Para lo cual tecnologías como REST permite la estandarización en la integración de sistemas políglotas, que pueden implementar infinidad de protocolos propietarios. Además, al considerar las características de las arquitecturas orientas a eventos, permiten que los sistemas puedan lidiar con una gran cantidad de tráfico generado por una gran cantidad de nodos.

En el contexto propuesto surge la necesidad de disponer de una herramienta que permita a los desarrolladores crear aplicaciones CPS sin la complejidad de dominar todas las tecnologías que intervienen en el sistema. Sin embargo, hasta el momento no existe una herramienta que cubra todas las capas de las arquitecturas IoT, WoT o CPS. Por esta razón nuestra propuesta cubre esa necesidad empleando Ingeniería de modelos, lo cual permite crear un plano arquitectónico que permitirá continuar con el desarrollo y la incorporación de nuevas tecnologías. Además, se creará un prototipo de herramienta gráfica que permitirá a los desarrolladores crear aplicaciones mediante la interconexión

de bloques conceptuales con lo cual se modela la arquitectura particular de las aplicaciones. Con el diseño de la aplicación se creará código de forma automática que se podrá cargar y compilar en cada dispositivo de la aplicación. Por último, la propuesta pretende evaluar la herramienta en un dominio concreto que tiene mucha difusión, como es el caso de la Domótica. Lo cual no implica, aunque la herramienta se vea limitada de forma exclusiva para este dominio ya que en su diseño se pretende modelar de forma abstracta cada elemento, lo cual se puede aplicar a una gran variedad de dominios. Una vez en funcionamiento de la herramienta se pretende evaluar cada uno de los artefactos de software creados mediante una serie de pruebas que validan el funcionamiento del mismo.

Al utilizar una gran cantidad de tecnologías que deben integrarse es necesario realizar el trabajo en grupo, para disminuir el tiempo de realización del proyecto, además se realizará pruebas prácticas en un protoboard y sus respectivos acoplamientos a las distintas plataformas y técnicas a utilizar, con la cual se ejecutarán pruebas para el control de los eventos complejos que podrán suceder en el hogar. A continuación, se lista las razones por las cuales el presente proyecto requiere de dos personas:

1. Se manejan múltiples tecnologías de diversos ámbitos, como son: Arquitecturas Orientas a Eventos, Arquitecturas Orientas a Servicios, Procesamiento de Eventos Complejos, Sistemas Ciberfísicos, Ingeniería Dirigida por Modelos. Para así obtener una correcta evolución al implementar el proyecto y se pueda trabajar en un futuro como base para otro plan tecnológico.
2. Al ser la herramienta genérica los Sistemas Ciberfísicos, es necesario definir un escenario para realizar las pruebas, este escenario será la Domótica, debido a que se podrá utilizar diferentes sensores y actuadores, para el control de eventos que se pueden presentar en la vida diaria.

3. Se generará diversos programas y se cargaran a un servicio específico, realizando las pruebas de funcionamiento, individual y grupal de todos los servicios, comprobando los resultados y forma de acción de cada uno de los componentes implementados.
4. Se manejan varias herramientas de software, como son: EMF, Sirius, Acceleo, Visual Studio Code, Java Script, Node Red, Arduino, Python, MySQL. Debido a que cada uno de los procesos que tendrá el proyecto tiene sus distintas herramientas de lenguaje que pueden ser compatibles para la integración entre ella y facilidad de programación de las mismas.
5. Se manejan varias tecnologías de comunicación: MQTT, HTTP, REST. Que permiten el acceso, producción y tratamiento y comunicación de información presentada en diferentes códigos (sensores, actuadores, etc).
6. Se analizarán varios protocolos de comunicación como: 802.11, 802.15.4, 802.15 y SigFox. Permitiendo la comunicación entre los dispositivos empleados en el sistema.
7. Se abordan temáticas de Sistemas de Control, Domótica, Cloud Computing, Servidores de aplicaciones, Broker de mensajería, Internet de las Cosas, Web de las Cosas. Obteniendo la integración de cada una de estas temáticas, obteniendo un sistema dinámico al ser aplicado con los Sistemas Ciberfísicos.
8. Se evaluará el escenario con distintos tipos de eventos, para comprobar el funcionamiento del sistema y la eficiencia del mismo en los distintos ambientes que se pueden presentar en un hogar.

Alcance

El presente proyecto pretende alcanzar los siguientes hitos:

Se creará un DSL que permita la integración de las Arquitecturas Basadas en Eventos y las Arquitecturas Basadas en Servicios, para el modelamiento de sistemas CPS. Para lo cual se empleará Eclipse Modeling Framework. Además, mediante Sirius se creará un editor gráfico que permita el modelamiento de aplicaciones con CPS. Por último, se empleará Acceleo para generar las instancias específicas (código fuente) de cada uno de los nodos.

Para el diseño del DSL se establece el dominio de la domótica, sin que sea un limitante exclusivo se utilizará distintos tipos de sensores como (presencia, temperatura, humo, humedad, asistente Alexa) y actuadores como (motor, servo motor, led, buzzer o zumbadores) los cuales se conectarán con una plataforma propia en la que se desplegarán los servicios. Además, para el CPE se modelarán mediante lógica de control booleana.

El Editor Gráfico generado permitirá establecer: las conexiones que tendrán los sensores y los actuadores, establecer los métodos que dispondrán los servicios REST, la creación de los servicios orquestadores que implementarán el CPE, y los servicios que capturarán los eventos MQTT provenientes de un tópico.

Al implementar un Metamodelo del DSL tendremos una sintaxis concreta el cual será el Editor Gráfico, donde podrá ser una base para futuros proyectos, teniendo la gran ventaja de trabajo con ingeniería de modelos donde se tendrá un mapa arquitectónico, vamos a tener sus limitantes debido al tiempo o conocimiento, pero cualquier persona que conozca sobre ingeniería de modelos podrá utilizar este mapa arquitectónico para mejorarlo o modificarlo, es decir, ser una herramienta abierta que puede crecer con el tiempo.

Para evaluar el sistema se efectuarán diferentes pruebas de validación como: pruebas de funcionamiento a cada uno de los elementos para verificar que se entregue el código correcto y realice la función dada por el programador, pruebas de carga en los servidores, y pruebas de usabilidad para comprobar cuál es el porcentaje de dificultad al usar el software.

Objetivos

Objetivo General

Diseñar e implementar un lenguaje especifico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas Ciberfísicos (CPS).

Objetivos Específicos

- Investigar los trabajos relacionados con EDA, SOA, REST, CPS, CPE, MDE.
- Generar el modelo de la arquitectura para la estructuración del sistema.
- Crear el editor gráfico simulando escenarios que cumplan eventos requeridos.
- Generar el código para cada elemento que trabajará en el sistema.
- Procesar los eventos complejos para conseguir respuestas eficientes al problema detectado.
- Evaluar el sistema por medio de pruebas de carga en los servidores, pruebas de usabilidad del editor gráfico, análisis de puntos de función, pruebas de desempeño funcional del sistema.

Estado del Arte

El estado del arte está basado en el método de Mapeo Sistemático de la Literatura (Systematic Mappyng Study, SMS) el cual intenta cubrir un amplio campo pero sin adentrarse a detalle y una Revisión sistemática de la Literatura (Systematic Literature Review, SLR) con el fin de cubrir mayor profundidad a un tema concreto (Márquez, 2018).

En la Tabla 1 se presentan las preguntas para la investigación del método SMS, la cual permite una visión del entorno actual que se encuentra el campo de investigación, por medio de la recopilación de información en libros, revistas y artículos.

Tabla 1

Preguntas para el Estudio de Mapas Sistemáticos (SMS)

Pregunta de investigación	**Motivación**
SMS1: ¿Cuáles son las arquitecturas usualmente utilizadas para el procesamiento de eventos complejos en aplicaciones IoT?	Conocer que arquitecturas pueden ser utilizadas para el procesamiento de eventos complejos.
SMS2: ¿Qué sistemas se usan para el desarrollo de DLS en aplicaciones IoT?	Conocer los sistemas que permitan el desarrollo de DSL.
SMS3: ¿Qué técnicas y/o tecnologías son utilizadas en sistemas IoT?	Conocer la base tecnológica de los sistemas IoT para su implementación.
SMS4: ¿Qué tecnologías permiten la comunicación de sistemas ciberfísicos para aplicaciones IoT?	Conocer las tecnologías de comunicación en sistemas ciberfisico con aplicación al IoT.

Nota: Preguntas referentes al Estudio de Mapas Sistemáticos enfocadas al proyecto.

En la Tabla 2 se puede apreciar las preguntas previstas para la metodología SLR, permitiendo así conocer el campo de investigación, pero en un ambiente más técnico.

Tabla 2

Preguntas para la Revisión Sistemática de la Literatura (SLR)

Pregunta de investigación	**Motivación**
SLR1: ¿Qué tecnologías se utilizan para el desarrollo de un DSL gráfico?	Conocer las tecnologías que se usan para el desarrollo de DSL gráficos.
SLR2: ¿Qué tecnologías se usan para la gestión de procesamiento de eventos complejos?	Conocer las tecnologías que se usan para la gestión de procesamiento de eventos complejos.
SLR3: ¿Qué tecnologías se usan para el desarrollo de sistemas cibersificos?	Conocer las tecnologías que se usan para el desarrollo de sistemas ciberfisicos.

Nota: Preguntas referentes a la Revisión Sistemática de la Literatura enfocadas al proyecto.

Una vez planteadas las preguntas se procede a investigar información en artículos científicos, tesis, proyectos de investigación, desde el 2017 hasta el 2021. Realizada esta investigación se encontró 50 documentos con estudios verídicos, esto con el fin de que la información a tomar sea confiable, por esto la investigación solo se realizó en buscadores como GoogleScholar, IEEE, SciELO, Springer Link. A continuación, se citarán algunos trabajos de investigación que aportaron al desarrollo del presente proyecto de titulación.

Según (Salman, AL.Jawad, & Al Tameemi, 2020) el Internet of Things (IoT) es un instrumento multidisciplinaria usada por profesionales de campo para crear tecnología en su disciplina, por consiguiente, los profesionales en dichos campos requieren un lenguaje

de programación sencilla, comprensible y simple de utilizar para este objetivo. Es por esto que se han desarrollado idiomas específicos de dominio (DSL) para facilitar el camino para que estos profesionales integren las habilidades de IoT en sus labores cotidianas.

La introducción de nuevos procedimientos de desarrollo de aplicaciones y la utilización de estándares y herramientas de ingeniería permitan a los usuarios finales desarrollar sus propias aplicaciones de IoT lo cual ha recibido atención en la última década; no obstante, este aspecto necesita más trabajo por la comunidad investigadora.

De acuerdo a lo que indican (Bračevac, Salvaneschi, Erdweg, & Mezini, 2019) el campo de la correlación de eventos se ocupa del diseño, la teoría y la aplicación de lenguajes de patrones para asociar eventos de fuentes de datos distribuidos. La naturaleza es cada vez más impulsada por eventos de las aplicaciones, donde se han desencadenado una gran cantidad de esfuerzos de investigación y desarrollo que han dado como resultado diferentes familias de enfoques de correlación de eventos, como: procesamiento de flujo, programación reactiva, procesamiento de eventos complejos (CEP). Referente a lo investigado por (Ulvi & Ozdemir, 2018) los CEP analiza grandes flujos de datos de flujo de anotaciones oportunas recibidos de un entorno monitoreado para detectar eventos complejos en tiempo real se puede emplear en diferentes dominios como redes sociales, monitorización del tráfico, gestión de crisis, etc.

En la investigación realizada por (Wortmann, Combemale, & Barais, 2017) los lenguajes específicos de dominio son la segunda técnica de modelado más popular para desarrollo de sistemas computacionales con características particulares de un dominio dado. Donde la mayoría (82%) de las contribuciones en relación con DSL se publicaron a partir de 2010 e incrementaron desde entonces, donde (Carbonell, Medeiros, Silva, Rodrigues, & Bernardino, 2017) especifica que un DSL abarca una amplia gama de usos,

que van desde aplicaciones con campos geológicos hasta ayudar a administrar bases de datos SQL y se espera ver más técnicas de modelado dirigidas a la Industria 4.0.

La Ingeniería basada en modelos (MDE) ha sido reconocida a lo largo de la última década como una técnica de ingeniería de software dedicada al diseño, gestión y evolución de lenguajes informáticos que permiten la generación automática de código. Siguiendo esta tendencia (Amrani, Gilson, Debieche, & Englebert, 2017) presenta IoTDSL, el cual es un prototipo de lenguaje específico de dominio (DSL) designado a capturar las capacidades de los dispositivos IoT y sus configuraciones de implementación, al tiempo que ofrece a los usuarios finales crear sus propios escenarios.

Capítulo II: Marco Conceptual

Ingeniería Dirigida por Modelos

Abstracción

La abstracción es un proceso importante para el desarrollo de software, en donde se aíslan los elementos o detalles que no resultan relevantes de un objeto determinado, con el fin de establecer un modelo más simplificado y general (Rodenas, 2010). Cabe destacar que dependiendo de la aplicación se pueden definir varios niveles de abstracción, y a mayor nivel son menores los detalles del objeto en cuestión. En la Figura 1 se puede observar de manera gráfica dicho proceso de abstracción, donde se toman características puntuales de un objeto determinado.

Figura 1

Proceso de abstracción.

Nota. Obtenido de (Martinez, 2014)

Modelos

Un modelo define y ayuda a dar respuesta a un sistema en estudio y viceversa. El modelo debe ser una versión simplificada del sistema original, por lo cual no todas las características del sistema deben estar representadas en el modelo y este debe ser útil,

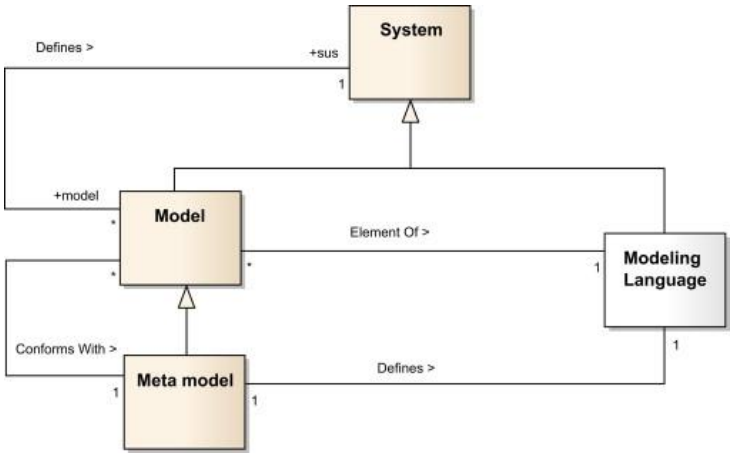
es decir debe ser capaz de sustituir al sistema original para determinados fines (Ludewig, 2003). También los modelos nos ayudan a visualizar el sistema, ya sea como es o lo que queremos visualizar, y nos proporcionan plantillas que ayudan a guiar el proceso de desarrollo.

Metamodelos

Un metamodelo se detalla como un modelo que define la estructura de un lenguaje de modelado (Rodrigues, 2015). Como se observa en la Figura 2, por las relaciones, un lenguaje de modelado es un conjunto de modelos, se observa también que un metamodelo es un modelo de un conjunto de modelos o modelo de modelos, lo que significa que el modelo debe satisfacer las reglas definidas al nivel de su metamodelo.

Figura 2

Definición de metamodelo.



Nota. Obtenido de (Rodrigues, 2015).

Arquitectura Dirigida por Modelos

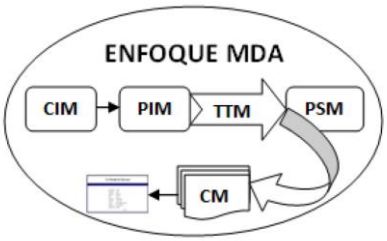
La Arquitectura Dirigida por Modelos (MDA) es un enfoque impulsado por modelos (MD) que nace de la iniciativa de Object Management Group (OMG) a partir de 2000, y

se centra en principalmente en definir estándares de modelos y sus transformaciones (Frankel, 2003).

MDA admite la definición de diferentes modelos en varios niveles de abstracción, entre ellos: Modelo Independiente de Computación (CIM) el cual expresa las necesidades de una manera general y poco técnica; Modelo Independiente de la Plataforma (PIM) este se encuentra en un mayor nivel de abstracción y debe estar construido de tal manera que defina la solución de manera precisa y formal; Tecnologías de la Transformación de Modelos (TTM) son las tecnologías usadas para poder pasar de un modelo a otro con un nivel de abstracción mayor o menor dependiendo del caso; Modelo Específico de la Plataforma (PSM) en estos modelos se ven involucradas algunas máquinas virtuales, sistemas operativos, hardware e interconexiones; Modelo del Código (CM) es conocido también como el código generado (Martinez, 2014). En teoría las aplicaciones desarrolladas con el método MDA son independientes de la plataforma debido a estas conversiones, especialmente las transformaciones PIM-PSM, PSM-PSM y PSM-CM, que se pueden instalar en diferentes plataformas informáticas y admitir completamente diferentes tecnologías (Frankel, 2003).

Figura 3

Pasos MDA para el desarrollo de software.



Nota. Obtenido de (Martinez, 2014).

MDA se relaciona con múltiples estándares, tales como Meta-Object Family (MOF), Unified Modeling Languaje (UML), XML Metadata Interchange (XMI), Enterprise Distributed Object Computing (EDOC) y Common Warehouse Metamodel (CWM).

Desarrollo De Software Dirigido por Modelos

El Desarrollo de Software Dirigido por Modelos (MDD) se centra principalmente en los requisitos, el análisis, el diseño y las disciplinas de implementación. Este enfoque MD tiende a definir lenguajes de modelado en diferentes niveles de abstracción y así proporciona transformaciones Modelo a Modelo (M2M) y Modelo a Texto (M2T) todo esto con el fin de mejorar la productividad y la calidad del software, es decir, en lugar de requerir que los desarrolladores que utilicen un lenguaje de programación que explique cómo se implementa el sistema, les permite usar modelos para especificar funciones requeridas del sistema y la arquitectura que se utilizará (Atkinson & Kühne, 2003).

Dentro de las ventajas de MDD destacan las siguientes:

- Velocidad en el desarrollo: Esto se debe a que el código se genera automáticamente, lo que reduce el tiempo en el desarrollo de aplicaciones.
- Calidad del software: Los componentes se generan automáticamente a partir del modelo formal, lo que reduce el error humano.
- Reusabilidad: El código fuente que es generado y las herramientas pueden ser usados para la construcción de varias aplicaciones.
- Evitar la redundancia: Se evita la implementación de código varias veces porque se generan y distribuyen para las implementaciones específicas.
- Portabilidad: Los artefactos que se generan son independientes a las plataformas, por lo que pueden ser portados fácilmente.

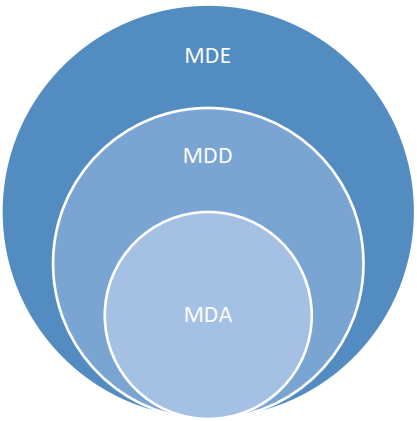
- Interoperabilidad: Los metamodelos no obedecen a la tecnología sino a la lógica de desarrollo.

Ingeniería Dirigida por Modelos

La Ingeniería Dirigida por Modelos (MDE) se encuentra en el nivel más alto de abstracción de todos los enfoques MD. MDE considera que los modelos pueden servir no solo como documentación o información, sino que pueden usarse en todas las disciplinas de ingeniería y en cualquier dominio de aplicación (Rodrigues, 2015).

Figura 4

Estructura Ingeniería Dirigida por Modelos



Nota. En esta figura se observa la jerarquía de los diferentes enfoques impulsados por modelos.

.

Dentro de los principios de MDE se pueden considerar como fundamentales los siguientes:

- Un modelo representa total o parcialmente un aspecto de un sistema software.
- Los modelos son representados mediante Lenguajes de Dominio Específico.

- Un metamodelo es usado para representar formalmente un Lenguaje de Dominio Específico.
- La automatización de código se consigue gracias a la traducción de los modelos a código mediante transformaciones de modelos.

Un desafío importante en el uso generalizado de MDE es la naturaleza diferente de los lenguajes de programación, la tecnología y las herramientas de soporte de MDE, lo cual hace que sea responsabilidad de los desarrolladores de MDE identificar y poder "combinar" las técnicas y habilidades necesarias para proporcionar uniformidad, cohesión e integración perfecta (Abrahão, y otros, 2017).

Lenguaje Específico de Dominio

Los Lenguajes Específicos de Dominio (DSL) son lenguajes que se adaptan al dominio de una aplicación o problema en específico. Estos lenguajes no están destinados a solucionar todo tipo de problemas, como Java o C que son lenguajes de propósito general. Pero si el domino de un problema está cubierto por un DSL en particular, este se podrá resolver de manera más fácil y rápida usando un DSL en lugar de un lenguaje de propósito general (Bettini, 2016).

Algunos ejemplos de DSL son SQL (que sirve para realizar consultas a bases de datos relacionales), XML (para el transporte de datos), CSS (para la definición de interfaces de usuarios), etc. Un programa que este escrito en un DSL, se puede compilar o interpretar en un lenguaje de propósito general; y en otros casos se los pueden ser procesados por otros sistemas.

Los DSL desde el punto de vista de la construcción de lenguaje se pueden clasificar en dos tipos: los DSL internos que utilizan un lenguaje anfitrión para dar la apariencia de ser otro lenguaje concreto, y los DSL externos que son aquellos que tienen

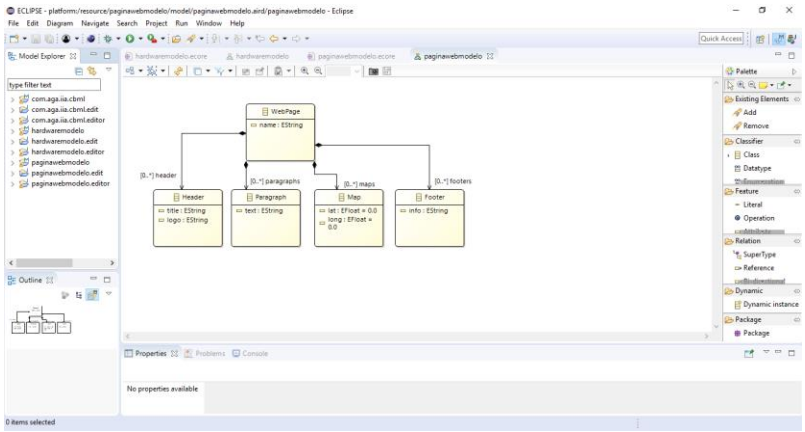
su propia sintaxis y se necesita de un parser para poder procesarlos (Montenegro, Cueva, Martinez, & Gaona, 2010).

Herramientas del MDE

- Eclipse: Es una plataforma de desarrollo, que es diseñada para que pueda ser extendida de forma ilimitada a través del uso de plug-ins. No posee un lenguaje específico, sino que es un IDE genérico.
- Eclipse Modeling Framework: Eclipse Modeling Framework (EMF) es un framework de modelado y generador de código para poder construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado. Proporciona varias herramientas o plug-ins para así poder producir un conjunto de clases de Java para el modelo, junto con otras clases de adaptador que permiten la visualización y edición de los modelos basados en comandos.
- Eclipse Modeling Project: Se centra en la evolución y en promover tecnologías de desarrollo basadas en modelos al proporcionar un conjunto unificado de estructuras de modelado, herramientas e implementaciones de estándares.

Figura 5

Vista general de la ventana de EMF.

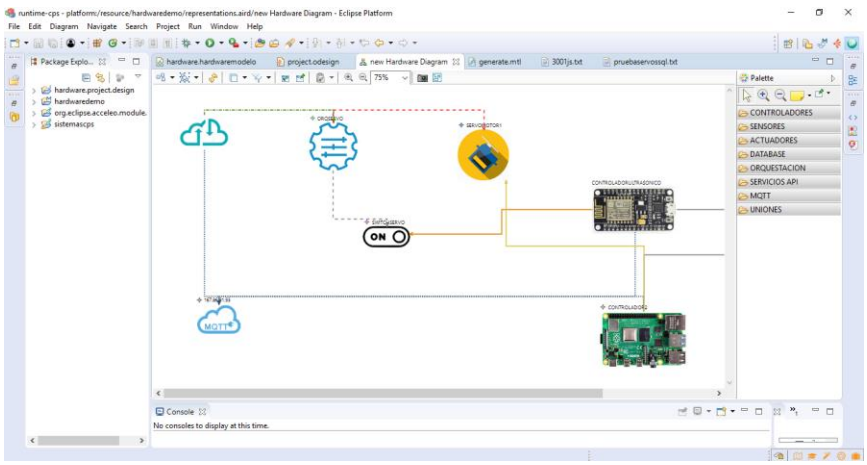


Nota. En la figura se observa la vista general de EMF para desarrollar metamodelos.

- Acceleo: Es una tecnología basada en plantillas que sirve para la creación de código personalizado y automático. Permite producir automáticamente cualquier tipo de código fuente a partir de los datos disponibles tomados en formato EMF, es decir nos sirve para realizar las transformaciones de modelo a texto.
- Sirius: Es un proyecto de Eclipse el cual permite la creación de un banco de trabajo de modelado grafico aprovechando la tecnología EMF de Eclipse. Un banco de trabajo de Sirius está compuesto por un conjunto de editores gráficos (diagramas, tablas, árboles) que permite crear, editar y visualizar los modelos EMF.

Figura 6

Vista general de una ventana de proyecto en Sirius.



Nota. En la figura se observa la ventana de Sirius para representar de manera gráfica un metamodelo.

Internet de las Cosas (IoT)

Definición

El Internet de las Cosas es un cambio novedoso en el ámbito de las tecnologías de la información, aun así, esta no es una definición acertada, debido a que no existe una única definición disponible que sea aceptable para la comunidad mundial de usuarios. Todas las definiciones tienen algo en común y es la idea que la primera versión de internet trataba sobre datos creados por personas, mientas que la siguiente versión trata sobre datos creados por cosas. Con la conceptualización de varios académicos, investigadores, profesionales, innovadores, desarrolladores y personas corporativas la mejor definición para el Internet de las Cosas seria: “Una red abierta y completa de objetos inteligentes que tienen la capacidad de organizarse automáticamente, compartir información, datos y recursos, reaccionando y actuando ante situaciones y cambios en el entorno”. (Madakam, Ramaswamy, & Tripathi, 2015)

Sistemas Ciberfísicos (CPS)

Los CPS son integraciones de la computación con procesos físicos. Las computadoras y las redes integradas monitorean y controlan los procesos físicos, generalmente con lazos de retroalimentación donde los procesos físicos afectan al sistema. Algunas aplicaciones de los CPS son dispositivos y sistemas médicos de alta confianza, asistentes virtuales, control y cuidado del tráfico, sistemas automotrices avanzados, procesos de control, conservación de energía, control del ambiente, instrumentación, sistemas de comunicación, manufactura, robótica, etc. (Lee E. , 2008)

La investigación de sistemas ciberfísicos tiene como objetivo la integración de conocimientos y principios de ingeniería (control, redes, electrónica, mecánica,

biomédica, etc.) para así desarrollar nueva ciencia de CPS y tecnología de apoyo. (Baheti & Gill, 2011)

Servidores MQTT

MQTT es un protocolo de comunicación que trabaja en base al protocolo TCP y así asegurar la entrega de mensajes de un nodo a un servidor. Debido a que es un mensaje orientado a la entrega de información, este protocolo es ideal para el uso del IoT.

MQTT es un protocolo basado en publicación y subscripción. Toda conexión MQTT involucra dos partes: clientes MQTT y brokers o servidores MQTT. Cualquier dispositivo o programa que se conecta a la red e intercambia datos por aplicaciones MQTT se conoce como clientes MQTT, cabe recalcar que el cliente MQTT puede ser de subscripción o publicación. Mientras que el bróker o servidor MQTT es un dispositivo o programa que interconecta a los clientes MQTT. (Kodali & Soratkal, 2016).

Sensores

Sensores o también conocido como transductores, los sensores son dispositivos que detectan los cambios en el entorno, donde este convierte un fenómeno físico en una señal analógica o digital, así esta señal puede ser procesada. Existen varios tipos de sensores, pero para este proyecto se definieron los siguientes para su utilización:

- Presencia: dispositivo electrónico que detecta cualquier movimiento en el área del trabajo del sensor.
- Temperatura: dispositivo electrónico que permite medir la variación de temperatura del aire o del agua y transformarla a una señal eléctrica determinada.
- Humo: dispositivo electrónico que detecta la presencia de humo en un área definida.

- Humedad: dispositivo electrónico que detecta la cantidad de humedad que existe en el ambiente.
- Asistente Alexa: dispositivo electrónico que presenta un asistente de voz interactivo, con capacidad de integrar diferentes funciones por medio de rutinas programadas.

Actuadores

Un actuador es un dispositivo substancialmente mecánico, su función es dar fuerza para mover o actuar otro dispositivo mecánico. Esta fuerza puede ser por tres fuentes: presión neumática, presión hidráulica, y fuerza motriz eléctrica. Por medio del tipo de fuerza que actúa, al actuador se lo denomina neumático, hidráulico o eléctrico. Para la presentación el sistema se definió los siguientes actuadores:

- Motor DC: es un motor de corriente continua, su propiedad es la conversión de energía eléctrica a energía mecánica, donde se produce un movimiento rotatorio por medio de un campo magnético.
- Servo motor: son dispositivos de accionamiento para el control de precisión de velocidad, par motor y posición.
- Led: es un diodo que además de permitir el paso de la corriente desde un solo sentido, este emite luz.
- Buzzer: también conocido como zumbador, es en transductor capaz de convertir la energía eléctrica en sonido.

Controladores

El controlador es un dispositivo en el cual se definirá diferentes acciones de funcionamiento por medio de un programa, para la implementación del sistema se utiliza dos controladores, el Arduino ESP8266 y la Raspberry Pi.

Arduino se ha vuelto muy popular con personas con enfoque electrónico, la ventaja de Arduino es su placa a la cual solo se debe cargar el programa deseado y hacer las conexiones deseadas para que este pueda funcionar. La adaptabilidad de este dispositivo es lo que lo llego a ser conocido a nivel mundial por su forma simple, compacta y la variedad de placas que se fabrican, también por su fácil estudio y programación, lo convierten en un micro controlador accesible. (Basamasi, 2014)

Raspberry Pi es una placa de bajo costo, pequeña y portable. Esta puede ser conectada a un monitor o televisión, teclado, mouse, etc. Esta tiene un software integrado que permite a los usuarios programar diferentes comandos usando lenguaje de programación “Python”. (Zhao, Jegatheesan, & Loon, 2015)

Domótica

Domótica implica el control y seguimiento de electrodomésticos en un sistema unificado. Estos sistemas de control envuelven iluminación, control de clima, seguridad e incluso electrónica del hogar. La domótica está estrechamente relacionada con la automatización de edificios. Hay muchos tipos de sistemas disponibles para la domótica. Estos sistemas suelen ser diseñados para distintos propósitos. La domótica puede ir desde simples interruptores de luz hasta redes que controlan todos los dispositivos en un edificio. La domótica es una aplicación extremadamente atractiva del IoT, donde se visualiza un futuro entorno de sensores y actuadores integrados, en productos y sistemas electrónicos de consumo que son auto configurables y se puede controlar de forma remota por medio del internet, permitiendo así una variedad de aplicaciones de control y monitoreo. (Miori & Russo, 2014).

Herramientas

Para el presente proyecto se pretende utilizar las siguientes herramientas que permitan utilizar el IoT, esto se presenta en la Tabla 3.

Tabla 3

Herramientas para el uso de IoT

Herramienta	Detalle
Arduino IDE	El programa Arduino (IDE) de código abierto permite la escritura de código y su carga en la placa. Este programa se puede usar con cualquier placa Arduino. Admite un soporte multiplataformas, detección de la placa a usar y autoguardado al compilar y cargar el script.
Python	Lenguaje de programación interpretado y orientado a objetos. Es de lenguaje interpretado con un formato de código estructural, es multiplataforma y multiparadigma.
Mosquitto	Es un mánager de mensajes de código abierto, es liviano y conveniente para su uso en todos los dispositivos. El protocolo MQTT otorga un procedimiento ligero para realizar mensajes por medio de un modelo de publicación/suscripción. Esto lo hace apropiado para la mensajería de Internet de las cosas, como con sensores de baja potencia o dispositivos móviles como smartphones, computadoras integradas o microcontroladores.

Nota: Definición de las herramientas necesarias en el proyecto para la integración de un sistema IoT

Arquitectura Orientada a Servicios (SOA)

Definición

La arquitectura orientada a servicios es un paradigma para organizar un conjunto de capacidades, a menudo distribuidas a través de la red y posiblemente bajo el control de diferentes propiedades de dominio. Las capacidades organizadas se pueden utilizar para proporcionar soluciones a problemas empresariales. Un problema empresarial se define ampliamente como cualquier problema, en cualquier dominio de interés, encontrado por un individuo u organización a medida que avanza en sus actividades. SOA se aplica a problemas empresariales que son susceptibles a soluciones de la tecnología de información. Los servicios se hacen visibles para los usuarios potenciales, interactuar con los usuarios a través de una serie de intercambios de información y producir efectos en el mundo real. La invocación y el intercambio de información se basan en lenguajes estándar y protocolos que facilitan la interoperabilidad. La secuencia detallada de acciones llevadas a cabo por el servicio puede involucrar cualquier número de operaciones como consultas de bases de datos, transformaciones de datos, ejecución de modelos, y formateo de pantallas. Estas operaciones pueden estar compuestas por operaciones de nivel inferior. Los módulos de datos o softwares necesarios para realizar las operaciones pueden residir en diferentes ubicaciones físicas y ser controladas por diferentes Organizaciones. SOA libera que los usuarios se concentren en su problema empresarial, dejando los detalles de solución al servicio. (Laskey & Laskey, 2009)

Arquitectura Dirigida por Eventos

La Arquitectura Dirigida por Eventos (EDA) es un patrón de arquitectura de software en la que cada vez que ocurre algún evento notable dentro de la arquitectura, este se difunde hacia todas las partes interesadas, las mismas que se encargan de

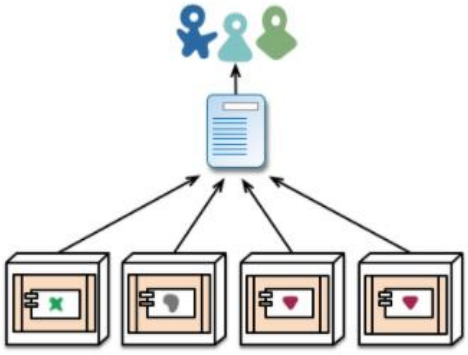
evaluar el evento y realizan una acción, la cual puede ser la invocación de un servicio, la activación de algún proceso local y/o la publicación o distribución de información adicional. En los casos de EDA, el creador del evento solo dicho la ocurrencia de dicho evento, esto no tiene conocimiento del procesamiento posterior al evento, ni de las partes que están interesadas del mismo, por lo cual EDA se utilizan de mejor manera para flujos asíncronos de trabajo e información (Theorin, y otros, 2017).

Microservicios

Los microservicios son un patrón arquitectónico emergente fuera de la Arquitectura Orientada a Servicios (SOA), enfatizando la autogestión y la ligereza, estos han surgido recientemente como un estilo arquitectónico que aborda cómo construir, administrar y evolucionar arquitecturas a partir de unidades pequeñas e independientes. Específicamente en la nube, el enfoque de la arquitectura de microservicios parece ser una complementación ideal de la tecnología de contenedores a nivel de PaaS. Los microservicios se pueden empaquetar idealmente, aprovisionados y orquestados a través de la nube. (Pahl & Jamshidi, 2016)

Figura 7

Microservicios corriendo en diferentes procesos.



Nota. Obtenida de (Lewis & Fowler, 2014).

Según (Lewis & Fowler, 2014) los microservicios se pueden caracterizar por medio de un número de principios:

- Organización en torno a la capacidad empresarial.
- Diseño evolutivo.
- Automatización de infraestructura.
- Inteligencia en los endpoints.
- Heterogeneidad y control descentralizado.
- Control descentralizado de datos.
- Diseño para fallas.

Servicios REST

REST, Representational State Transfer, es un estilo arquitectónico que determina un conjunto de limitaciones como la extensibilidad funcional (código a pedido), interacción cliente – servidor sin estado (el servidor no almacena información sobre el cliente), visibilidad de interacción, datos que proporcionan información de control y una interfaz uniforme entre componentes arquitectónicos, manipulación de recursos, mensajes auto descriptivos, etc.

Los datos son el elemento principal en REST, siendo el recurso la abstracción de información básica; un recurso es un mapeo conceptual a un conjunto de entidades. Los recursos son abstractos y se realizan mediante representaciones que hacen innecesario clasificar los recursos según un tipo o implementación (la representación únicamente sirve al cliente, mientras que el recurso en sí está oculto por el servidor). Las representaciones comprenden datos y metadatos que indican el estado actual o previsto del recurso, o el valor de otro recurso. (Alarcon, Saffie, Bravo, & Cabello, 2015)

Las solicitudes del cliente incluyen datos de control, un identificador de recursos y una representación opcional. Los servidores realizan acciones sobre los recursos de acuerdo con la solicitud y brindan una respuesta que puede incluir control y metadatos de recursos, y una representación opcional. Los datos de control determinan el propósito del mensaje y las acciones de los componentes intermedios, como los cachés. El formato de datos de la representación se conoce como tipo de medio. Las representaciones transmiten el conjunto de transiciones de estado disponibles para un recurso en un estado determinado en forma de enlaces y controles. Los agentes de usuario (por ejemplo, navegadores web) se comportan como motores que mueven una aplicación web de un estado al siguiente según las acciones del usuario. (Alarcon, Saffie, Bravo, & Cabello, 2015)

La arquitectura REST según (Neumann, Laranjeiro, & Bernardino, 2018) cubre estas principales características:

- Número de operaciones.
- Esquema de diseño utilizado en los URI (contiene información de recursos o también contiene información sobre la operación).
- La técnica de la selección del formato de salida.
- La información de alcance (informa al servidor sobre qué datos debe operar).
- Soporte de versiones de API.
- Soporte para el almacenamiento en caché de respuestas.
- Uso de enlaces a recursos relacionados en los mensajes de respuesta.

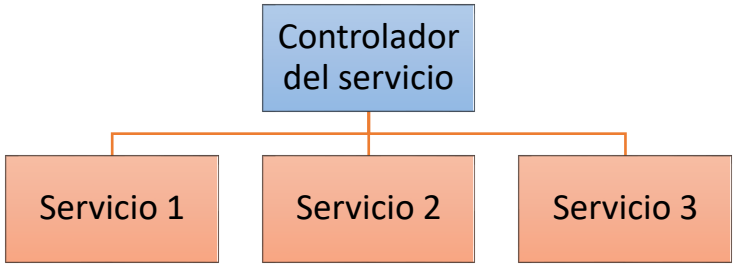
Orquestación

Para una solución basada en la web, la lógica empresarial debe distribuirse en diferentes servicios. Estos servicios deben estar organizados y ajustados, lo que

comúnmente se denomina orquestación. Un proceso de orquestación presenta diferentes servicios que se pueden componer de manera eficiente a través de un flujo para ejecutar un proceso de negocio. (Karande, Karande, & Meshram, 2011)

Figura 8

Orquestación del servicio.



Nota. En la figura se observa que distintos servicios pueden involucrarse con un mismo orquestador o controlador.

En la Figura 8, los distintos servicios involucrados se coordinan a través de un controlador. La capa de servicios de orquestación proporciona un medio poderoso por el cual las soluciones orientadas a servicios pueden obtener algunos beneficios claves como:

- Las aplicaciones y servicios se pueden diseñar libremente.
- Ser independientes del proceso y reutilizables.
- La lógica del proceso está centralizada en una ubicación, en lugar de estar distribuida e integrada en distintos servicios.

Para la orquestación del servicio se pueden integrar diferentes tipos de control, el control usado en el proyecto está enfocado a la lógica boolena, la cual es una lógica de conjuntos, su función principal es definir formas de intersección entre conjuntos.

Herramientas

Para el presente proyecto se pretende utilizar las siguientes herramientas que cumplan con la lógica de la arquitectura orientada a servicios y eventos, esto se presenta en la Tabla 4.

Tabla 4

Herramientas para la aplicación EDA y SOA

Herramienta	Detalle
Node-RED	Es una herramienta de programación para conectar dispositivos, APIs y servicios en línea. El cual se basa por la interconexión de flujos, utilizando diferentes nodos. En si Node-RED es una programación de bajo nivel para aplicaciones impulsadas por eventos.
Node.js	Es un entorno de ejecución para JavaScript orientado a eventos asíncronos, diseñada para crear aplicaciones network escalables.
MySQL	Es un sistema de gestión de bases de datos. Esta utiliza tablas múltiples que se interconectan mutuamente, para así almacenar información y organizarla correctamente.

Nota: Definición de las herramientas necesarias en el proyecto para la integración de la arquitectura orientada a servicios y la arquitectura orientada a eventos.

Pruebas

Pruebas de funcionamiento

Las pruebas de funcionamiento incluyen un conjunto de actividades que son realizadas para la identificación de posibles fallos de funcionamiento, configuraciones o disponibilidad de algún programa, por medio de pruebas sobre el comportamiento del mismo.

Prueba de carga

Las pruebas de carga sirven para la medición de las capacidades operativas bajo diferente nivel de carga de las solicitudes de los usuarios hacia los servicios web, identificando así la capacidad operativa máxima y averiguar que factor está causando la degradación, evaluando también el rendimiento de los servicios web y por lo tanto las aplicaciones orientadas a servicios. Para realiza una prueba de carga se realiza una simulación de características reales de solicitudes masivas de usuarios, incluida la concurrencia real, la diversidad de ubicación geográfica y la configuración del sistema (Minzhi, Hailong, Xudong, Ting, & Xu, 2014).

Puntos de función

Puntos de función es una técnica de medición del tamaño funcional del software, desde el punto de vista del cliente, este análisis no considera ningún aspecto de implementación de la solución, y por último es un método estándar ISO / IEC 20926 de medición de software, el cual cuantifica los requisitos funcionales del usuario.

Para los puntos de función se toma en cuenta 5 componentes, los cuales son:

1. Entrada de usuario: hace referencia a todos los procesos que ingresan datos, o actualizan cualquier archivo interno.

2. Salida de usuario: son todos los procesos donde se envían datos ya sean informes o listados al exterior de la aplicación.
3. Consulta de usuario: son todos los procesos donde se realizan las consultas de datos de los archivos internos.
4. Archivos lógicos internos: son todos los grupos de datos ya sean tablas en la base de datos o archivos internos al sistema.
5. Interfaces externas: datos referenciados a otros sistemas.

Prueba de usabilidad

La prueba de usabilidad consiste en el análisis dado por los usuarios de un producto o servicio. Este análisis permite determinar la utilidad del producto o servicio, para así considerar la capacidad que tiene con el fin de cumplir el propósito para el cual fue diseñado.

Capítulo III: Diseño

Requisitos de diseño

Los requisitos de diseño del sistema se conocen por ser el medio de comunicación entre el cliente y el desarrollador. Se debe iniciar con los requerimientos que tiene el proyecto para así estructurar el diseño de la arquitectura y cumplir así con los objetivos planteados en el Capítulo I, plasmando así los propósitos con los que fue planteado el proyecto inicialmente.

Requisitos funcionales

En la Tabla 5 se encuentra la información de los requisitos funcionales.

Tabla 5

Requisitos funcionales de diseño

Requisitos funcionales	
Requerimientos del cliente	Interfaz de operación amigable para el usuario, con información para ser fácil de manejar. El usuario debe tener conocimientos básicos sobre sensores, actuadores, controladores y programación. El usuario debe tener varios softwares de programación como Eclipse, Node-Red, Visual Studio Code, Arduino IDE, Python.
Necesidades y requisitos del software que debe cumplir de manera satisfactoria	La recolección y transferencia de datos debe ser en tiempo real. El usuario tiene acceso de la base de datos y los servicios rest.

	La base de datos debe ser incremental debido a la acumulación de datos que puede existir.
Funciones que el software será capaz de realizar	El programa debe ser capaz de entregar los programas necesarios según el esquema realizado en el editor gráfico.
Detalles técnicos	Sistema operativo Windows 7 o superior. Procesador: Intel from 1.2 GHz ó equivilent AMD family. Memoria: 4 GB de RAM. Gráficos: DirectX 9 compatible Tarjeta gráfica card.
Manipulación de datos de entrada y salida	El orquestador será el encargado de administrar los datos y producir los diferentes eventos que se haya definido en el editor gráfico.

Nota: Listado de los requisitos funcionales para la utilización de la aplicación a desarrollar.

Requisitos no funcionales

En la Tabla 6 se encuentra la información de los requisitos no funcionales.

Tabla 6

Requisitos no funcionales de diseño

Requisitos no funcionales	
Propiedades o cualidades que el software debe tener	El programa se realizará en eclipse por lo cual debe tener una interfaz sencilla para la utilización de los usuarios y así programar el entorno que desea.

	La aplicación podrá entregar distintos códigos, dependiendo de los elementos integrados en el desarrollador grafico realizado.
Restricción del sistema operativo	Windows 7 o posterior, es recomendable usar Windows 10.
Restricción de ambiente	Debe utilizar diferentes microservicios por lo cual se aplica servidores mqtt, base de datos, servicios rest, de los cuales se utilizara sus usuarios y contraseñas.
Restricciones o condiciones del producto	Programas solo para arduino esp8266 y raspberry pi de los sensores y actuadores definidos.
Restricción de rapidez	Toma de datos a tiempo real. Conexión a internet indispensable para el funcionamiento.
Restricción de seguridad	Los datos de la base de datos no son editables por el usuario. Se guardaran solo datos de lo que requiera el usuario.
Restricción de usabilidad	La arquitectura se basa en microservicios.

Nota: Listado de los requisitos no funcionales para la utilización de la aplicación a desarrollar.

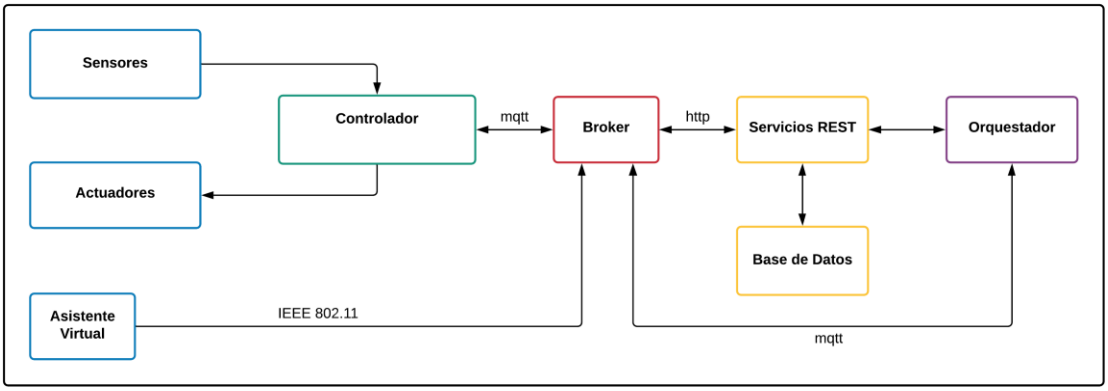
Arquitectura

La base del presente proyecto de titulación es la generación de códigos por medio del editor gráfico “Sirius”, iniciando con el programa del controlador donde se realizará la adquisición de datos de los sensores y se activaran los actuadores, estos dos elementos están conectados físicamente al controlador, a su vez se comunicará por medio del

protocolo mqtt con el bróker de mensajería, el cual es se encarga de administrar todos los datos provenientes del controlador o el asistente virtual. Los datos del bróker mediante servicios REST y protocolos http podrán ser almacenados en la base de datos. El orquestador podrá acceder a los datos almacenados previamente realizando consultas http y enviar la acción a ser realizada por el controlador mediante el bróker por protocolos mqtt. Toda esta estructura esta descrita en la Figura 9.

Figura 9

Arquitectura del sistema.

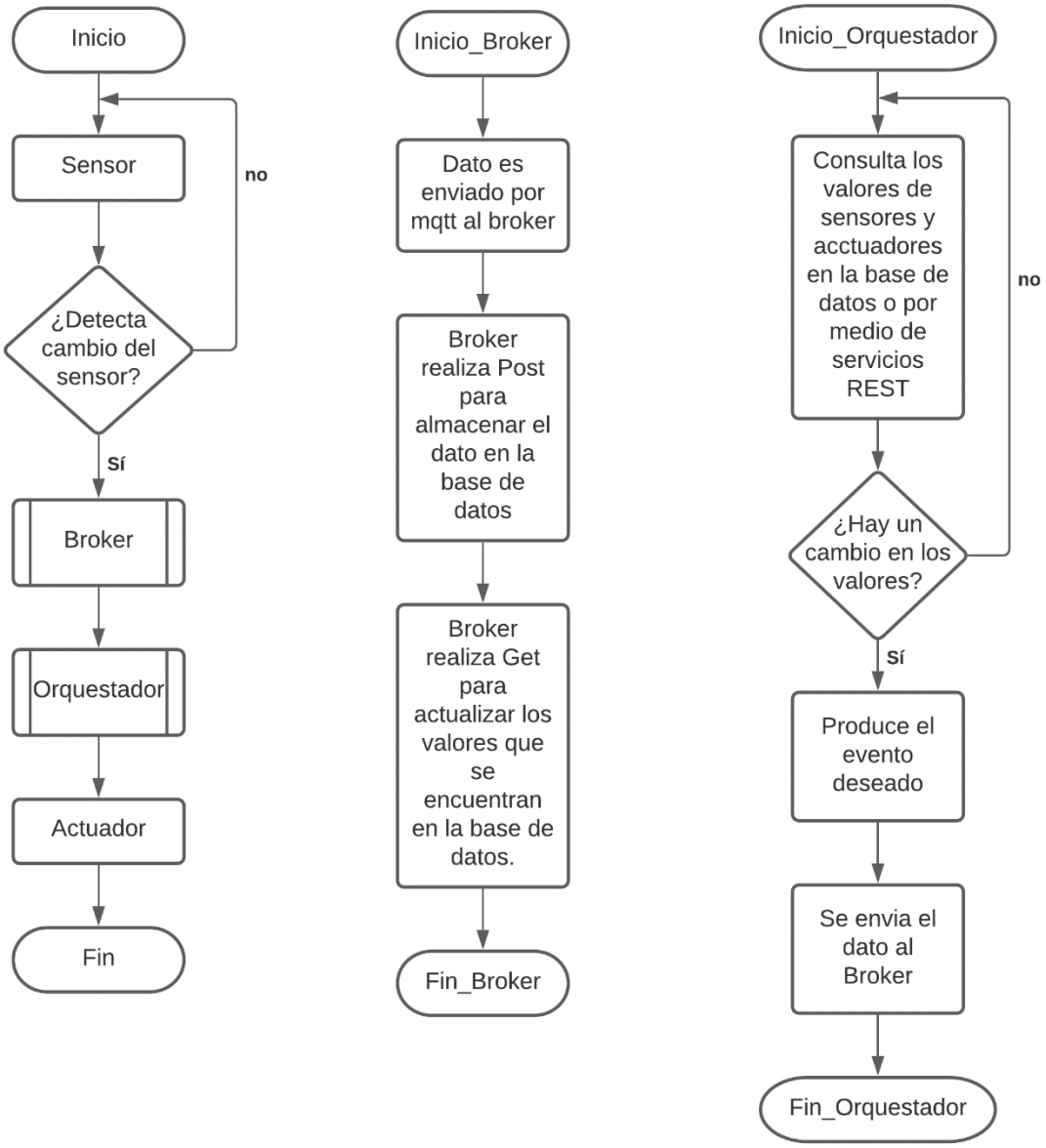


Nota. En la figura se muestra de la arquitectura a ser tomada por el sistema.

El la Figura 10, se presenta un diagrama de flujo explicando el proceso que cumple el sistema ya implementado.

Figura 10

Diagrama de proceso del proyecto de titulación.



Nota. En la figura se muestra de manera breve un diagrama de flujo explicando el funcionamiento del sistema.

Lenguaje de dominio especifico

Los lenguajes de dominio especifico constan de 3 partes las cuales son: la sintaxis abstracta la cual se encarga de describir tanto el software como el hardware del sistema y está representado por los metamodelos; la sintaxis especifica o concreta que es encargada de la representación de la sintaxis abstracta mediante una herramienta grafica para su mejor comprensión y visualización; finalmente la transformación modelo a texto la cual permite que a través de los modelos creados mediante la sintaxis concreta se puedan crear las instancias específicas de software y hardware que componen nuestro sistema IoT.

Metamodelo

Para el diseño del metamodelo debemos conocer todas las características de los elementos a utilizar, para así poder definir las clases y atributos que presentara el metamodelo. Por ello a continuación se detallan las especificaciones de sensores, actuadores, controladores, asistente virtual, base de datos, servidores mqtt, servicios REST y orquestación.

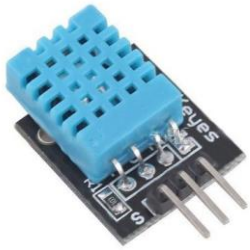
Características de los elementos del sistema

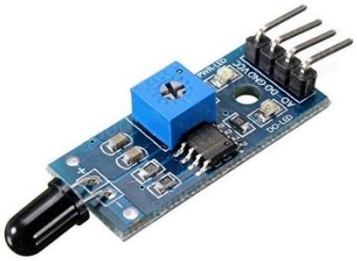
Sensores

Para el diseño del presente proyecto se trata de utilizar distintos tipos de sensores por lo cual en la Tabla 7 se enlistará los sensores, con sus respectivas características.

Tabla 7

Lista de sensores

Nombre	Imagen	Características
DHT11		- Sensor de humedad y temperatura. - Salida de tipo digital utilizando un pin de datos. - Alimentación de 3-5 V. - 1 segundo de tiempo de sensado. - Rango de temperatura de 0 a 50°C - Rango de humedad de 20% a 90% RH (Humedad relativa)
LM35		- Sensor de temperatura - Salida analógica - Rango de temperatura de -55°C a 150°C - Alimentación de 4 a 20 V
PIR		- Sensor de movimiento - Alimentación de 4.5 a 20 V - Salida digital (Alto) - Rango de detección de 3 a 7 m y ángulo operativo de 120°

IR (proximidad)		- Sensor de proximidad - Alimentación de 3.3 a 5 V - Salida digital (Bajo) - Rango de detección de 2 a 30 cm
IR (llama)		- Sensor de llama - Alimentación de 3.3 a 5 V - Salida digital (Bajo) - Rango de detección de longitud de ondas de luz de 760nm a 1100nm
Ultrasónico		- Sensor de distancia - Alimentación de 5 V - Rango de medición de 2 a 450cm - Pin TRIG (Disparo del ultrasonido) y ECHO (Recepción del ultrasonido)

Nota: Listado de los sensores a utilizar, exponiendo sus principales características.

Actuadores

En el diseño se pretende utilizar varios actuadores, en la Tabla 8, se enlistara los actuadores utilizados para el diseño del proyecto.

Tabla 8

Lista de actuadores

Nombre	Imagen	Características
Diodo Led		- Paso de corriente de un solo sentido - Polarizado emite un haz de luz - Alimentación de trabajo aproximadamente de 2 V
Motor DC		- Voltaje nominal 3 V - Rango de operación 1.5 a 5 V.
Servomotor		- Voltaje de funcionamiento 3 a 7.2 V - Ángulo de rotación 180° - Velocidad 0.1 sec/60° a 4.8 V

Nota: Listado de los actuadores a utilizar, exponiendo sus principales características

Asistente virtual

El asistente virtual Alexa Echo Dot se configuro con la skill de Node-RED como se presenta en la Figura 11.

Figura 11

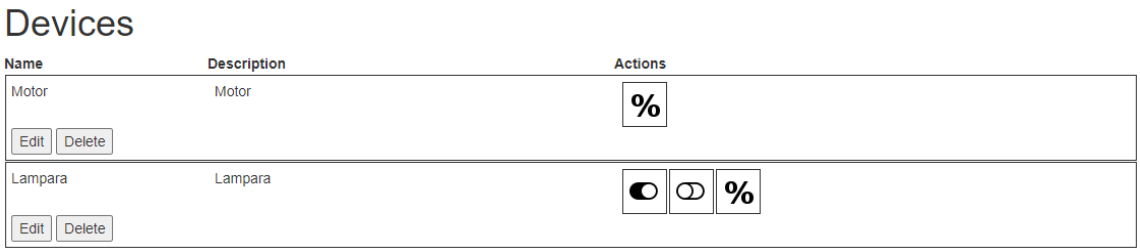
Skill Node-RED para uso de Alexa Echo Dot.

Nota. En la figura se ve el ícono y el nombre de la Skill de Alexa en Amazon.

Esta skill permite interactuar con dispositivos virtuales, a través de la página web de https://alexa-node-red.bm.hardill.me.uk/ esta web permite crear sus propios "dispositivos" para trabajar con el sistema Home Skill de Amazon Echo, dándole control de voz sobre básicamente cualquier cosa con la que pueda interactuar con Node-RED, es decir, se puede crear dispositivos los cuales pueden ser configurados para una acción on – off, o con un porcentaje de activación como se muestra en la Figura 12.

Figura 12

Creación de dispositivos para Alexa Echo Dot.



Nota. En la figura se ve de formar breve como lucen los dispositivos creados virtualmente en la página.

Controladores

Para los controladores nos enfocamos en dos plataformas, Arduino y Raspberry Pi, debido a su capacidad de adaptación con sensores, actuadores y que permiten utilizar internet para el envío y recepción de datos. En la Tabla 9, vamos a enlistar las componentes que se planea utilizar y sus características

Tabla 9

Lista de Controladores

Nombre	Imagen	Características
Arduino ESP8266		- Alimentación por puerto USB - Pin analógico - GPIO pins para entrada o salida - PCB Antenna, conexión WiFi
Raspberry Pi 3		- Alimentación por puerto USB de 2.5 A - 40 pins GPIO - HDMI - 4 puertos USB - Sistema Operativo Raspbian

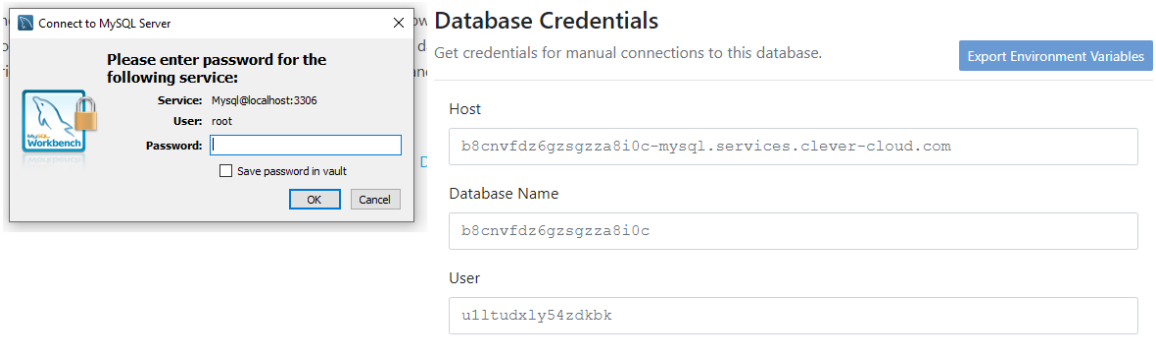
Nota. En esta tabla se muestran los controladores con sus respectivas características.

Base de datos

Dentro de las características ineludibles que se necesitan para poder acceder a una base de datos ya sea que esta se encuentre de manera local o en un servidor son: el host por el cual se quiere acceder, el nombre de usuario del gestor de la base de datos y la contraseña del mismo.

Figura 13

Credenciales de un gestor de base de datos de manera local o remota.



Nota. En la figura se observan todas las credenciales necesarias de una base de datos para poder acceder a ellas.

Una vez que se pueda ingresar al gestor de las bases de datos con las credenciales previas, las siguientes características que no se pueden pasar por alto son: el nombre de la base datos en la cual se desean trabajar para la creación de querys o tablas y los nombres de las mismas para asociarlas a sus respectivas bases.

En la

Figura ***14*** se puede observar un ejemplo de cómo es la creación de una base de datos y una tabla asociada con sus respectivas columnas, cabe destacar que el código usado en este ejemplo se lo puede implementar a un gestor de manera local o remota.

Figura 14

Creación de una base de datos y una tabla asociada.

```
CREATE DATABASE if not exists actuadores1;
USE actuadores1;
    CREATE TABLE if not exists led12 (
    n INT primary key auto_increment not null,
    id VARCHAR (11) NOT NULL,
    lugar VARCHAR(45) NOT NULL,
    estado FLOAT (2) NOT NULL,
    nombre VARCHAR(45) NOT NULL,
    fecha TIMESTAMP
    );
```

Nota. En la figura se observa el código necesario para crear una base de datps y una tabla.

Para nuestro aplicativo el programador estará en la libertad de crear diversas bases de datos y asociarlas con diferentes sensores, actuadores o asistentes virtuales para así poder almacenar las diferentes características y estados de los componentes.

Servidores Mqtt

Las características que son indispensables para poder acceder hacia un broker de mensajería Mqtt son: el host del servidor en el que se encuentra montado el broker, ya sea este de manera local o remota; el puerto por el cual se puede acceder al servidor; el nombre de usuario, y en caso de tenerla, una contraseña, como se muestra en la

Figura 15.

Figura 15

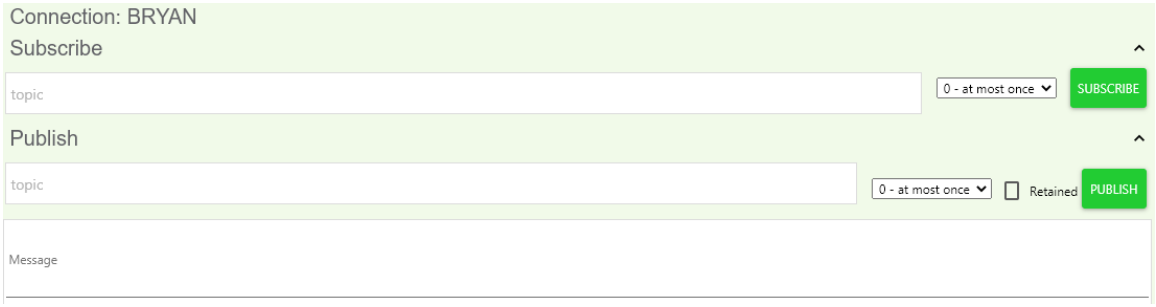
Características para acceder a un broker mqtt.

Nota. En la figura se observan las credenciales necesarias para acceder remotamente a un broker mqtt.

Adicional a las credenciales previas es necesario el nombre de uno o varios tópicos a los cuales se puedan suscribir para recibir la información que llega o puedan publicar cualquier tipo de mensaje que viaje a través del mismo.

Figura 16

Suscripción y publicación de tópicos mqtt.



Nota. En la figura se muestra los paneles para realizar publicaciones o suscripciones a tópicos mqtt mediante MqttLens.

En nuestro caso en particular, como se observó en la arquitectura mostrada en la Figura 16 los controladores serán los encargados de publicar en un tópico diferente el estado y las características de cada sensor, el orquestador estará suscrito a estos tópicos para la toma de decisiones y dependiendo de si existe algún cambio en el actuador publicará el resultado en un tópico diferente para cada actuador, el controlador también estará suscrito a los tópicos de los actuadores y así realizara la activación o desactivación de los mismos. El broker de mensajería Node Red se suscribirá a los tópicos tanto de actuadores como de sensores y enviará la información a través de servicios rest para el almacenamiento de los cambios de los dispositivos en sus bases de datos.

Servicios REST

Para poder crear o acceder a los servicios rest, es necesario un host y un puerto por los cuales se realizará la comunicación, posterior a esto son necesarias las diversas url de los servicios a los cuales se quiere acceder.

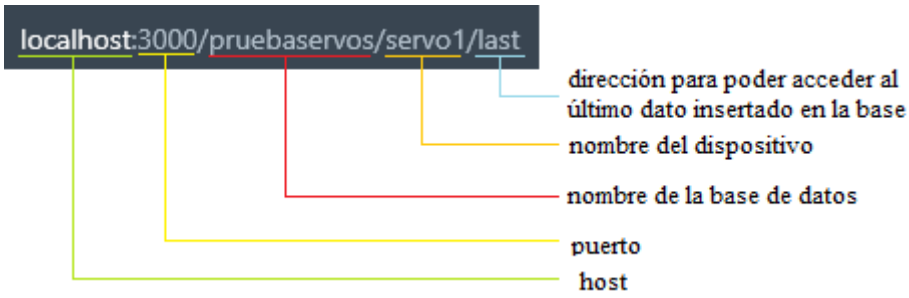
Los servicios rest utilizan los métodos del protocolo HTTP, los cuales son: GET para recoger la información de un recurso; POST para crear un nuevo recurso en el servidor; PUT para modificar algún recurso y DELETE para eliminar algún recurso.

En nuestro caso particular, se crearán distintas url´s para poder acceder a la base de datos de cada sensor o actuador y mediante el método GET realizar una consulta del último estado de los componentes, y con el método POST realizar un posteo de algún cambio de un componente realizado por los controladores o el orquestador. El la

Figura 17 se muestra un ejemplo de una url generada para los servicios rest, en donde se quiere acceder al último registro de la tabla “servo1” de la base de datos “pruebaservos”.

Figura 17

Estructura de la url de los servicios rest.



Nota. En la figura se observa el formato que hemos dado a la url que se genera automáticamente para los microservicios.

Orquestador

Para el orquestador al igual que en los otros servidores, es necesario un host y un puerto por el cual se pueda acceder a la orquestación, cabe recalcar que en un mismo puerto de orquestación pueden alojarse distintas funciones de orquestación para diferentes actuadores.

Debido a que se puede controlar los actuadores por control on-off o por PWM, antes es necesario observar antes cuál de estos tipos de control se va a implementar, ya

que para el caso del control on-off en la función de orquestación cuando esta se cumpla se enviará un “1” al controlador por medio de mqtt, mientras que para el caso del control por PWM al cumplirse la función de orquestación se enviara un porcentaje seteado previamente para que el controlador trabaje sobre el actuador.

También se puede implementar en el orquestador que este actúe automáticamente en la toma de decisiones sin la necesidad de haber cambios previos de sensores, esto gracias a poder setear una fecha para el inicio y apagado de los actuadores o a su vez poder setear un valor de delay en el cual se especifica el tiempo que el actuador permanecerá prendido y apagado.

Por lo cual dadas las especificaciones que puede tener un orquestador en la toma de decisiones (obviando el host y puerto de conexión) este tendría las siguientes características:

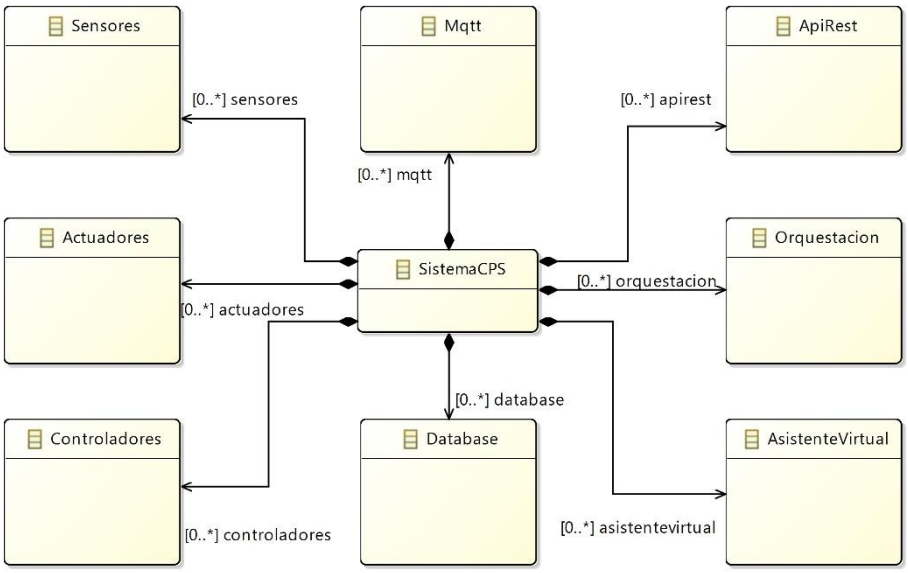
- Función de orquestación.
- Fecha de encendido.
- Fecha de apagado.
- Tiempo de encendido.
- Tiempo de apagado.
- Porcentaje de PWM (en caso de actuadores análogos o con un servo motor).

Clases del metamodelo

Una vez explicadas y detalladas todas las características y elementos que intervienen en un CPS, a un nivel muy grande de abstracción se obtendría el siguiente esquema:

Figura 18

Esquema de elementos de un CPS.



Nota. En la figura se observa un metamodelo del sistema a un gran nivel de abstracción.

Luego de entender de manera general los elementos que intervienen en los CPS, se necesita disminuir el nivel de abstracción y ver detalladamente que atributos poseen sus componentes para poder llegar a un metamodelo, que abarque todos los elementos con sus respectivas características.

Sensores

Para los sensores se ha previsto que se implemente una clase padre llamada “Sensores” la cual tendrá los atributos nombreSensor (String), id (String) y lugar (String), los cuales son los mismos para todos los sensores, no importa si estos son digitales o analógicos.

De esta clase abstracta heredaran sus atributos cuatro clases hijas con atributos diferentes unas de otras.

En primer lugar, la clase “SensoresVirtuales” en donde se podrán configurar los diferentes sensores que se agreguen al asistente virtual.

Seguido la clase “sensoresDig” la cual hace referencia a todos los sensores digitales que se puedan utilizar, tiene como único atributo a “pin” (String) en donde se podrá configurar al pin del controlador al que se conecte dicho sensor. Debido a que los únicos valores que puede tomar un sensor digital son 0 y 1, el controlador al detectar un cambio de estado enviará automáticamente el estado actual del sensor.

Posterior dado a que los controladores ESP a utilizar poseen una entrada analógica, se ha creado la clase “sensoresAnag” en la cual se pueden referenciar a todos los sensores analógicos que tengan un solo pin de salida, esta clase tiene como atributos propios a “pin” (String) en donde se podrá configurar al pin del controlador al que se conecte dicho sensor y a “delay” (Int), en donde el programador podrá colocar un tiempo de delay en el que desee que el controlador envié las variaciones del sensor, dependiendo del caso para el cual este siendo utilizado.

Finalmente se ha creado una segunda clase abstracta llamada “SensoresEspeciales” de la cual podrán desprenderse diferentes clases hijas de sensores

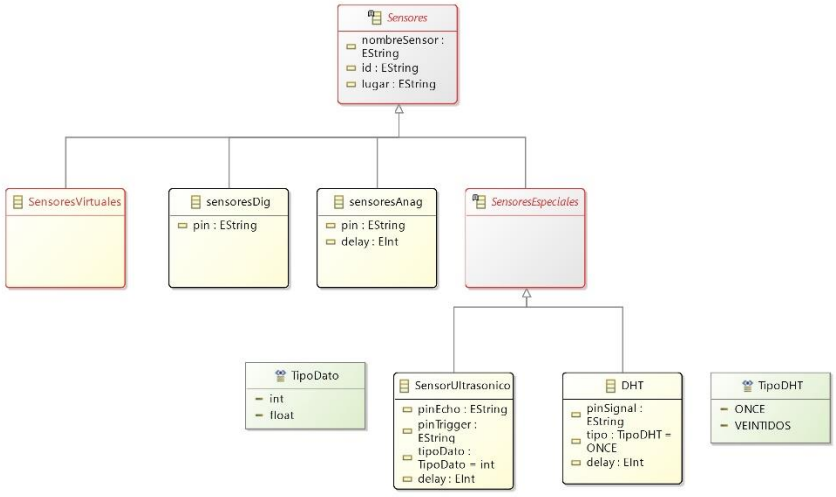
que tengan pines en específico o requieran de alguna configuración en particular en su implementación.

Dentro de los sensores especiales que vamos a usar y se han descrito previamente se encuentra el sensor ultrasónico, para su configuración se ha creado una clase “SensorUltrasonico” la cual tiene los siguientes atributos: pinEcho (String) donde se podrá configurar el pin del sensor con el mismo nombre; pinTrigger (String) donde se indica a donde ira conectado el pin Trigger del sensor; tipoDato (puede ser Int o Float) donde el programador podrá indicar si desea que los datos arrojados por el sensor sean enteros o flotantes; y delay (Int), en donde el programador podrá colocar un tiempo de delay en el que desee que el controlador envié las variaciones del sensor.

Finalmente se encuentra la clase “DHT” en la cual se puede configurar dicho sensor, posee los siguientes atributos: pinSignal (String) donde se indica a donde ira conectado el pin Signal que posee el sensor; tipo (puede ser “ONCE” o “VEINTIDOS”) donde se puede seleccionar con cuál de los dos tipos de sensores DHT se está trabajando; y delay (Int), en donde el programador podrá colocar un tiempo de delay en el que deseé que el controlador envié las variaciones del sensor.

Figura 19

Esquema de las clases usadas para representar los sensores.



Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los sensores.

Actuadores

Para los actuadores se ha previsto que se implemente una clase padre llamada “Actuadores” la cual tendrá los atributos nombreActuador (String), id (String) y lugar (String), los cuales son los mismos para todos los actuadores, no importa si estos son digitales o analógicos.

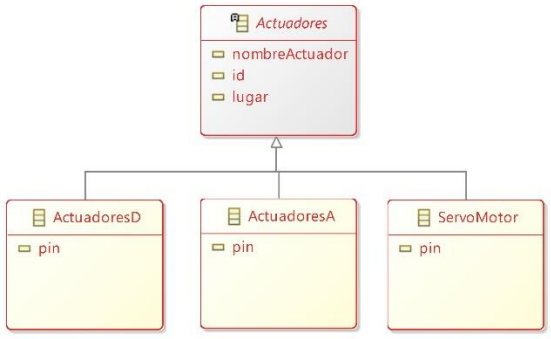
Esta clase padre tendrá 3 clases hijas llamadas “ActuadoresD”, “ActuadoresA” y “ServoMotor”, estas tres clases tendrán un solo atributo llamado pin (String) que sirve para definir el pin al cual se conectarán los actuadores en el controlador.

Es necesario que a pesar de que las tres clases tengan el mismo atributo separarlas, esto para poder identificar los diferentes tipos de actuadores a usar de diferente manera, ya que la clase “ActuadoresD” hace referencia a los diferentes actuadores digitales a los cuales solo se les envía un dato de 0 o 1, la clase “ActuadoresA” hace referencia a todos los actuadores que se los puede controlar mediante una señal PWM, a estos actuadores es necesario enviarles un porcentaje de PWM con el cual

trabajar. Finalmente, la clase “ServoMotor” es con la cual se va a identificar a los servomotores, que a pesar de que también se les puede enviar un porcentaje de PWM para que trabajen, para más facilidad del programador se enviará un grado de movilidad entre 0 y 180 que es el rango en el cual trabaja un servomotor.

Figura 20

Esquema de las clases usadas para representar los actuadores.



Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los actuadores.

Controladores

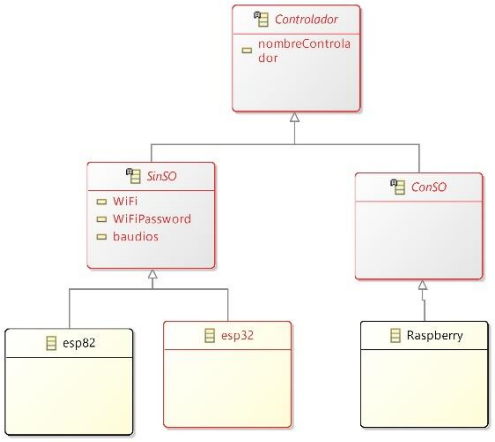
Para los controladores se ha previsto que se implemente una clase padre llamada “Controlador” la cual tendrá un atributo llamado nombreControlador (String) para con este poder identificar a cualquier tipo de controlador que vayamos a implementar.

Debido a los controladores que se van a poder implementar en el desarrollo, se ha distribuido a dos clases hijas las cuales son: “SinSo” en donde se identificaran a los controladores que no cuentan con un sistema operativo propio como la ESP82 y la ESP32, debido a que estos controladores para ejecutar la conexión a internet se la realiza mediante el código de programación a esta clase se le han cargado dos atributos llamados: WiFi (String) para poder definir el nombre a la red a la cual se van a conectar;

WiFiPassword (String) en donde se puede definir la contraseña de la red WiFi; y baudios (Int) esto se los define para poder realizar una comunicación serial con la computadora.

Figura 21

Esquema de las clases usadas para representar los controladores.



Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los controladores.

La otra clase hija de Controlador a ser implementada es “ConSo” en donde se podrá identificar a los controladores que tengan su propio sistema operativo implementado, en nuestro caso en específico a la Raspberry.

Bases de datos

Para las bases de datos se ha visto que es necesaria la implementación de una clase llamada “Database” la cual tendrá como atributos todos los elementos necesarios para poder acceder a una base de datos los cuales son: host (String) en donde se coloca el host al cual se quiere acceder; user (String) donde se define el usuario; password (String) es la variable donde se puede definir la contraseña del usuario para poder acceder al host; y database (String) donde se define el nombre de la base de datos que queremos crear para almacenar los datos de los elementos que vamos a almacenar.

Figura 22

Clase usada para representar a las bases de datos.

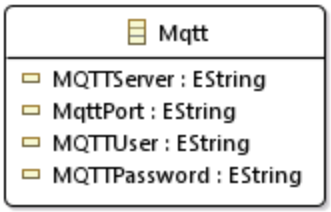
Nota. En la figura se observa un metamodelo para poder describir de mejor manera a las bases de datos.

Mqtt

Para los servicios Mqtt se ha visto conveniente implementar una clase llamada “Mqtt” la cual tiene por atributos todos los elementos necesarios para poder realizar la conexión con estos servicios, los atributos de la clase son: MQTTServer (String) donde se define el host o el servidor en donde se encuentran los servicios mqtt; MqttPort (String) donde se define el puerto por el cual acceder a los servicios mqtt; MQTTUser (String) en donde se describe en caso de ser necesario un usuario mqtt; y MQTTPassword (String) en donde se puede incluir la contraseña en caso de ser necesaria para poder acceder al host especificado previamente.

Figura 23

Clase usada para representar a los servicios mqtt.



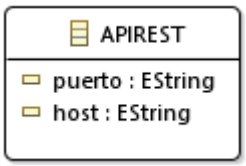
Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los servicios mqtt.

Servicios Rest

Para poder representar los servicios Rest se ha visto conveniente crear una clase llamada “APIREST” con los siguientes atributos: host (String) en donde se define el host en el cual se quieren montar los servicios rest; y puerto (String) en donde se determina el puerto por el cual se puede acceder al host.

Figura 24

Clase usada para representar los servicios rest.



Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los servicios rest.

Orquestador

Para los orquestadores se ha previsto la implantación de diversas clases entre estas “PuertoOrquestacion” que es en donde se definirá el host y el puerto en el cual estará montado el servicio de orquestación. Esta clase tendrá los siguientes atributos: host (String) donde se indica el host de servicio para la orquestación; y puerto (String) donde se indica porque puerto se puede acceder a nuestro host.

También se ha implementado una clase padre llamada “Orquestacion” la cual tendrá atributos que son indistintos de si la orquestación se la va a implementar de manera digital, por PWM o el ángulo de abertura de un servomotor. Estos atributos son: funcion (String) donde se define la función boleana de orquestación a ser implementada

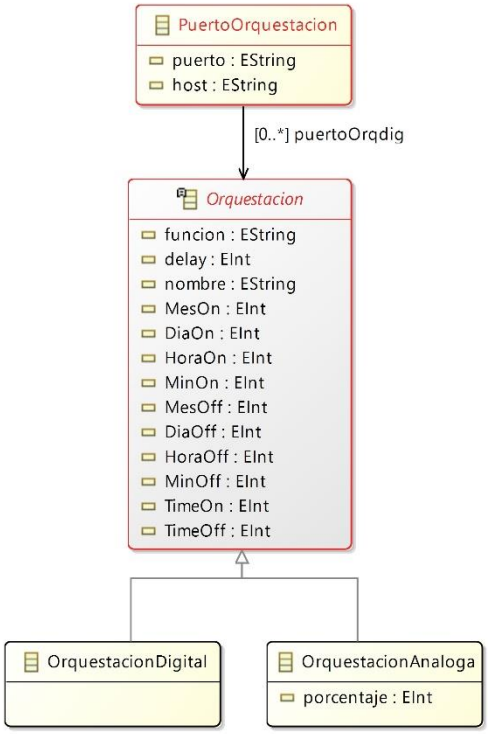
en nuestro sistema; nombre (String) donde se indica un nombre a la orquestación para poder ser identificada; delay (Int) para poder definir el tiempo en milisegundos en el cual el orquestador enviará los datos una vez que se cumpla la función; debido a que también se puede definir una fecha exacta para que el orquestador actué se implementan los atributos MesOn (Int), DiaOn (Int), HoraOn (Int) y MinOn (Int); para poder indicar en qué fecha exacta se desea que el orquestador deje de actuar se añaden los atributos MesOff (Int), DiaOff (Int), HoraOff (Int) y MinOff (Int); finalmente ya que se puede definir un tiempo específico en el cual se desea que el orquestador opere se inserta el atributo TimeOn (Int) y el atributo TimeOff (Int) donde se indica el tiempo que el orquestador no operará.

De la clase “Orquestacion” se desprenden dos clases hijas llamadas “OrquestacionDigital” y “OrquestacionAnaloga” donde la primera es usada para representar todas orquestaciones digital que se requieran implementar (es decir las que simplemente tengas una lógica de encendido o apagado), mientras que la clase “OrquestacionAnaloga” hace referencia a las orquestaciones donde se requiera enviar un porcentaje de PWM o ángulo de apertura de un servomotor por lo cual cuenta con un atributo llamado porcentaje (Int) en donde se indicará el valor a ser enviado una vez que se cumpla la función de orquestación.

Adicional dado que un mismo host puede albergar diferentes funciones de orquestación la clase “PuertoOrquestacion” tiene una relación de “cero a muchos” con la clase “Orquestacion” esto con el fin de no utilizar diferentes hosts o puertos para cada orquestación en específico.

Figura 25

Esquema de las clases usadas para representar a la orquestación.



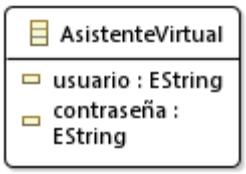
Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los orquestadores.

Asistente virtual

Para poder usar la skill de nuestro asistente virtual y vincularla a node-red se ha implementado una clase llamada “AsistenteVirtual” la cual cuenta con los siguientes métodos: usuario (String) donde se define nuestro usuario del asistente virtual, y contraseña (String) donde se determina la contraseña para acceder a nuestro usuario creado previamente.

Figura 26

Clase usada para representar al asistente virtual.



Nota. En la figura se observa un metamodelo para poder describir de mejor manera a los asistentes virtuales.

Relaciones entre clases del metamodelo

Para poder identificar las relaciones que existen entre las distintas clases que conforman al metamodelo es necesario hacer una visión general acerca de cómo se relacionarían estos elementos en un entorno real.

A los controladores (esp32, esp82 o raspberry) pueden conectarse diversos números de sensores o actuadores, dependiendo de sus pines de entrada y salida, pero los sensores y actuadores solo pueden estar conectados a un solo controlador, es por eso que la clase padre “Controlador” tiene una relación de cero a muchos con las clases padre “Sensores” y “Actuadores” y estas a su vez una relación de cero a uno con la clase “Controlador”.

Los controladores deben publicar y suscribirse a los servicios MQTT para poder enviar el estado de los sensores y poder recibir la información de los actuadores, sin embargo, un controlador solo podrá publicar en un servicio MQTT, dado esto es que la clase “Controlador” tiene una relación de cero a uno con la clase “Mqtt”.

Puesto que se crearán distintos microservicios por cada sensor o actuador que conste en nuestro sistema, es necesario que la clase “Controlador” tenga una relación de cero a muchos con la clase “APIREST” para así saber cuántos sensores y actuadores tiene nuestro sistema.

Las bases de datos pueden albergar datos de diferentes sensores y actuadores, pero estos solo pueden conectarse a una base de datos, dado esto la clase “Database” tiene una relación de cero a muchos con las clases “Sensores” y “Actuadores”, y estas a su vez una relación de cero a uno con la clase “Database”.

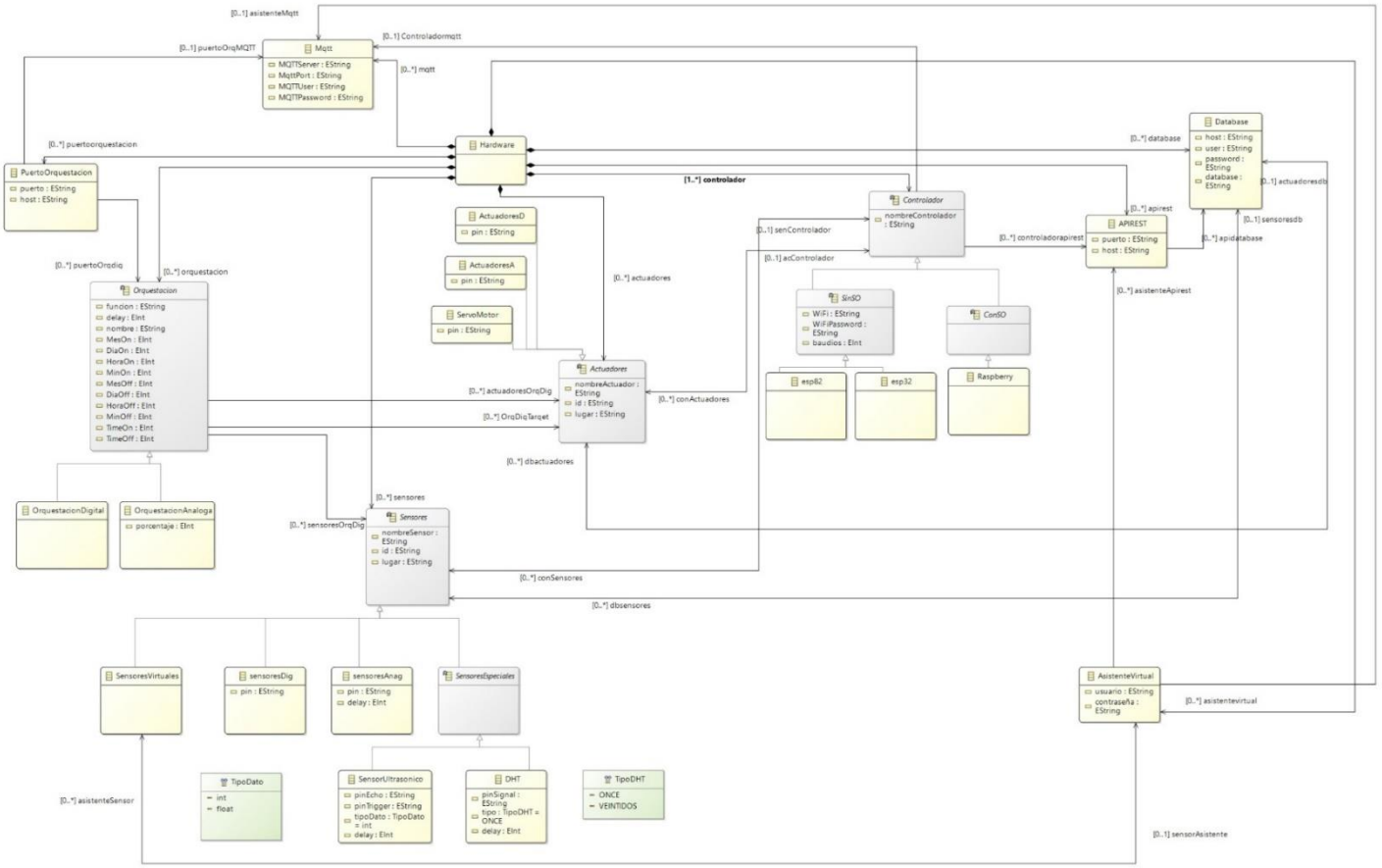
Los microservicios pueden realizar consultas o generar data para diferentes bases de datos, por lo cual la clase “APIREST” tiene una relación de cero a muchos con la clase “Database”.

La orquestación va a depender de los estados de los diferentes sensores o actuadores y esta puede afectar el comportamiento de varios actuadores, por lo que la clase “Orquestacion” tiene una relación de cero a muchos con las clases “Sensores” y “Actuadores”, donde se define que sensores intervienen en la orquestación y a que actuadores se modificará su estado.

Puesto que en un miso host y puerto de orquestación pueden existir varias funciones de orquestación y se necesita que estas publiquen sus respuestas por servicios MQTT, la clase “PuertoOrquestacion” tiene una relación de cero a uno con la clase “Mqtt”.

Figura 27

Esquema del metamodelo



Nota: Metamodelo desarrollado para que cumpla con los requerimientos planteados en el proyecto.

Capítulo IV: Implementación

Editor gráfico

Para la sintaxis específica del DSL que es la cual conforma al editor gráfico, se realiza su implementación por medio de Sirius de Eclipse, es necesario que los gráficos representados vayan acordes con las clases del metamodelo para una mejor comprensión por parte del programador y así pueda desarrollar las aplicaciones a su conveniencia.

Tabla 10

Íconos que representan de manera gráfica las clases del metamodelo

Controladores	
	Ícono usado para representar al controlador ESP32.
	Ícono usado para representar al controlador Raspberry.
	Ícono usado para representar al asistente virtual ALEXA.

Sensores	
	Ícono usado para representar a los sensores digitales.
	Ícono usado para representar a los sensores analógicos.
	Ícono usado para representar a los sensores virtuales para ser configurados con el asistente virtual.
	Ícono usado para representar los sensores de humedad y temperatura DHT11 y DHT22.
	Ícono usado para representar al sensor ultrasónico,
Actuadores	
	Ícono usado para representar a los actuadores digitales.

	Ícono usado para representar a los actuadores a ser controlados por una señal PWM.
	Ícono usado para representar a los servo-motores.
Orquestación	
	Ícono usado para representar el puerto de configuración de las orquestaciones.
	Ícono usado para representar las orquestaciones digitales.
	Ícono usado para representar las orquestaciones por PWM o hacia los servo-motores.

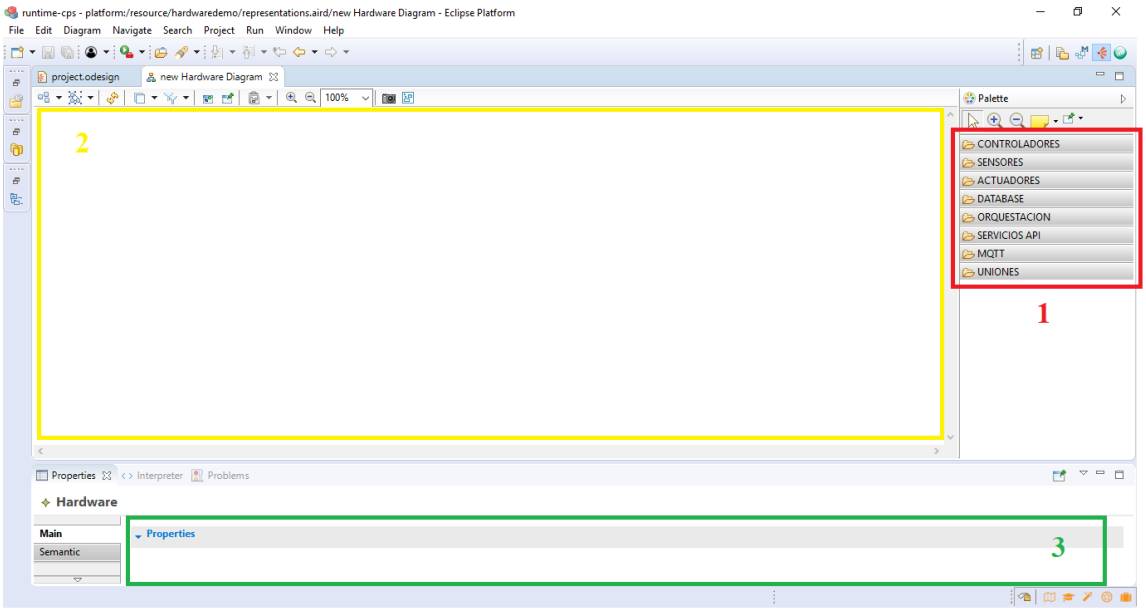
Database	
	Ícono usado para representar las configuraciones de las bases de datos.
Servicios API REST	
	Ícono usado para representar las configuraciones de los servicios api rest.
Servicios MQTT	
	Ícono usado para representar las configuraciones de los servicios MQTT.

Nota. En esta tabla se muestran todos los íconos usados para respresentar las diferentes clases en el editor gráfico.

Además la pantalla en donde el usuario puede programar las distintas configuraciones de los Sistemas Ciberfísicos cuenta con las partes de la Figura 28.

Figura 28

Partes de la pantalla de programación.



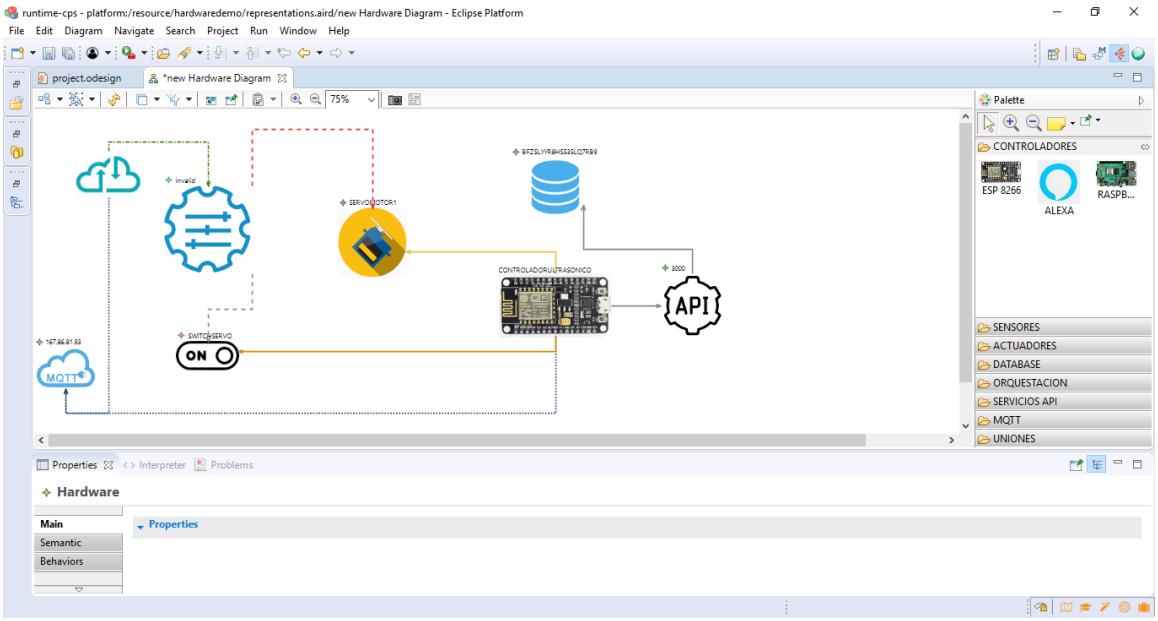
Nota: Interfaz de uso del editor gráfico, indicando los paneles utilizados al momento de programar.

- 1) Es en donde se encuentra un menú desplegable para poder acceder fácilmente a los componentes que conforman un CPS.
- 2) La pantalla de programación donde se colocarán los componentes a ser usados.
- 3) La zona de propiedades donde se indica o modifica uno o varios de los atributos de algún componente en específico.

En la Figura 29, se puede observar de mejor manera como luce una implementación de una configuración de un CPS con sus distintos componentes.

Figura 29

Ejemplo de implementación de un CPS.



Nota: Interfaz de la herramienta de Sirius como Editor gráfico del software Eclipse, demostrando un ejemplo base entre un sensor y un actuador conectados a un controlador ESP 8266.

Transformación modelo-texto

La transformación modelo-texto se basa en tomar la sintaxis concreta creada con Sirius y mediante Acceleo poder interpretar todas y cada una de las clases implementadas con sus relaciones para obtener el código que se usará para los controladores, bases de datos, orquestación, servicios REST y Node-Red.

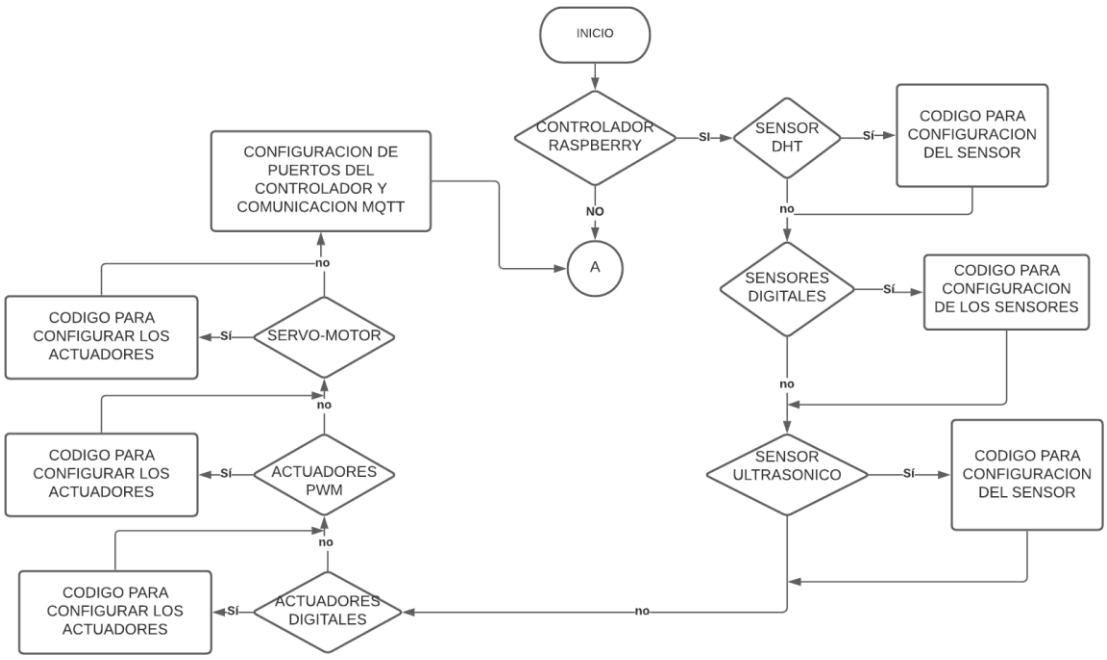
Dentro de Acceleo se implementan dos clases de lenguaje, el lenguaje de propósito general que describe el funcionamiento lógico para cada programa independiente de cada componente que conforma un CPS; y el lenguaje descriptor que

es propio de Acceleo y está programado en Java, el cual se encarga de interpretar los modelos de Sirius y unirlos con los lenguajes de propósito general.

En la Figura 30, se muestra el diagrama de flujo de Acceleo para la instancia de una clase de controlador Raspberry y como es la lógica para que se genere el código necesario para la correcta configuración del controlador.

Figura 30

Diagrama de flujo de Acceleo para los controladores Raspberry.

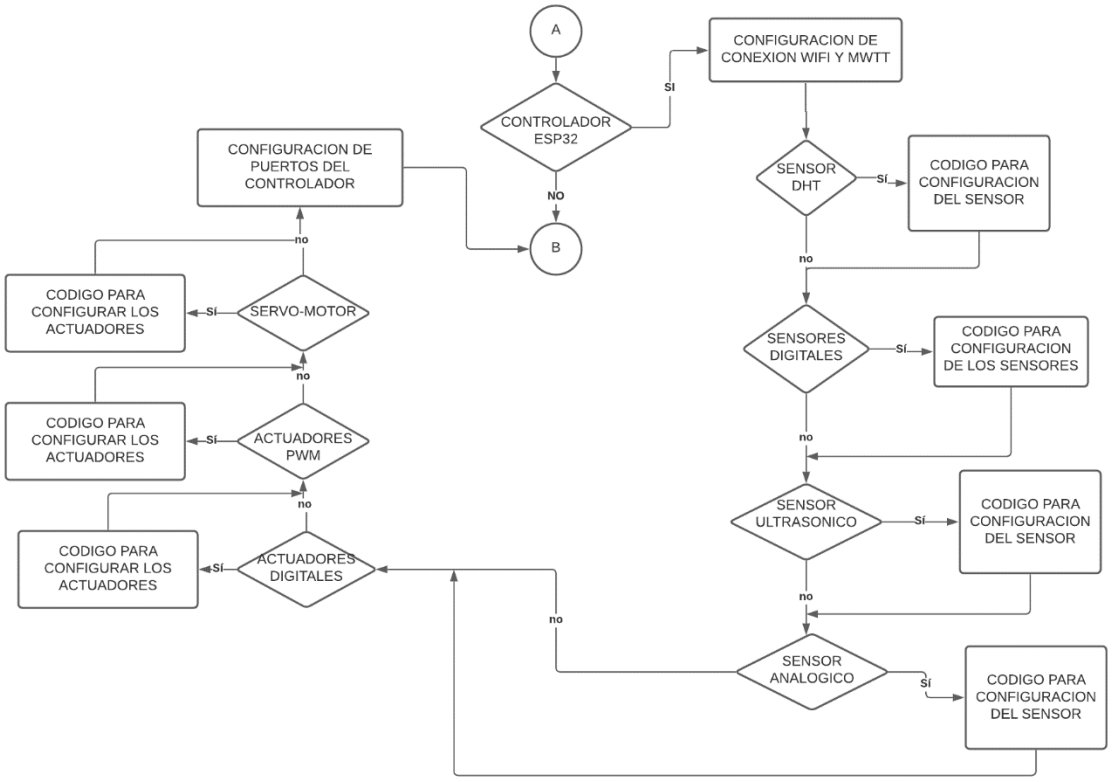


Nota: Diagrama de flujo para la obtención del código del controlador Raspberry Pi por medio de la herramienta Acceleo del software Eclipse,

Seguido de verificar si existe o no alguna Raspberry en el DSL se procede a configurar los controladores ESP32 y ESP82, cuya lógica de Acceleo se encuentra representada en el diagrama de flujo de la Figura 31.

Figura 31

Diagrama de flujo de Acceleo para los controladores ESP32 y ESP82.



Nota: Diagrama de flujo para la obtención del código de Arduino para los respectivos controladores por medio de la herramienta Acceleo del software Eclipse,

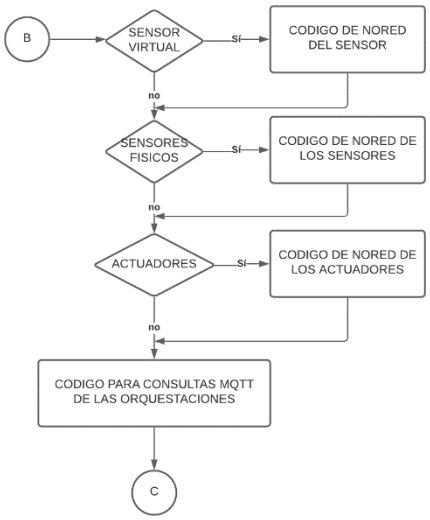
En la Figura 31 continua el proceso que realiza acceleo, en el cual se verifica si existe un controlador ESP8266, si existe dicho controlador, se agregara los datos para la configuración del wifi y el servidor mqtt, seguido analiza que sensores y actuadores posee y dependiendo eso agrega el código, por último, se agregara el puerto del controlador.

Mientras que en la Figura 32, se entregará el código para Node-RED dependiendo de los sensores, actuadores y asistentes virtuales agregados en el editor gráfico. Y en la Figura 33, se obtendrá el código de los servicios REST, donde se configura la conexión

a las bases de datos agregadas y se realizaran las peticiones get y post para sensores y actuadores.

Figura 32

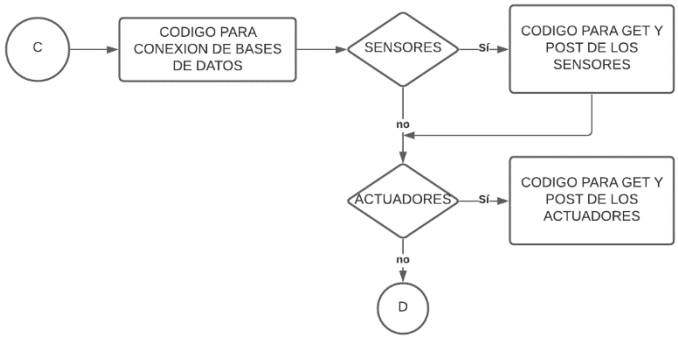
Diagrama de flujo de Acceleo para Node-Red.



Nota: Diagrama de flujo para la obtención del código de la estructura del bróker de mensajería Node-RED por medio de la herramienta Acceleo del software Eclipse,

Figura 33

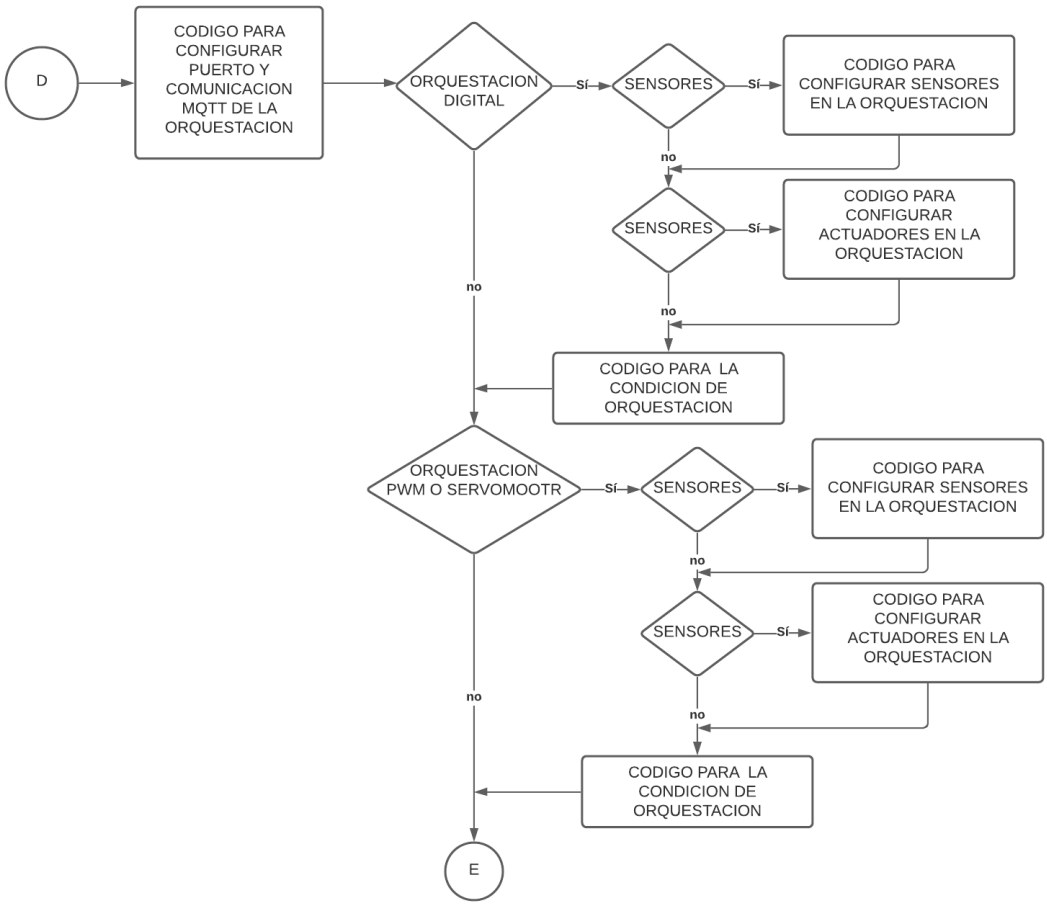
Diagrama de flujo de Acceleo para los servicios Rest.



Nota: Diagrama de flujo para la obtención del código de los servicios REST por medio de la herramienta Acceleo del software Eclipse,

Figura 34

Diagrama de flujo de Acceleo para la orquestación.



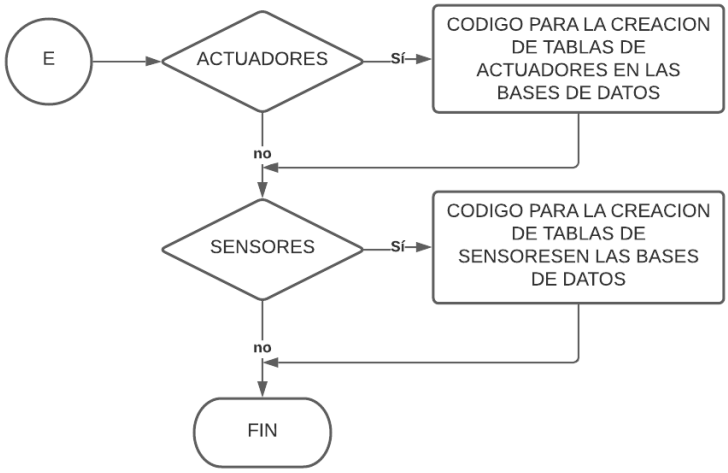
Nota: Diagrama de flujo para la obtención del código de la orquestación por medio de la herramienta Acceleo del software Eclipse,

En la Figura 34, se observa el proceso para la entrega del código de la orquestación en la cual tendremos primero la configuración del puerto y del servidor mqtt, y segundo se verifica si la orquestación es digital o una orquestación que actúa con un pwm o un servomotor, donde el proceso es similar si cumple cualquier condición se configura las publicaciones/subscriciones de los sensores y actuadores, seguido se agregara la condición “evento” para el trabajo de dicho actuador, esta condición depende

del tipo de orquestación a trabajar si es digital será por medio de lógica boolen, solo utilizando and u or, mientras que para pwm o servomotor, se condiciona con sentencias de mayor que, menor que o igual.

Figura 35

Diagrama de flujo de Acceleo para las bases de datos.



Nota: Diagrama de flujo para la obtención del código de la base de datos por medio de la herramienta Acceleo del software Eclipse,

Por último, en la Figura 35 se obtiene el código para crear la base de datos de todos los sensores y actuadores agregados en el editor gráfico.

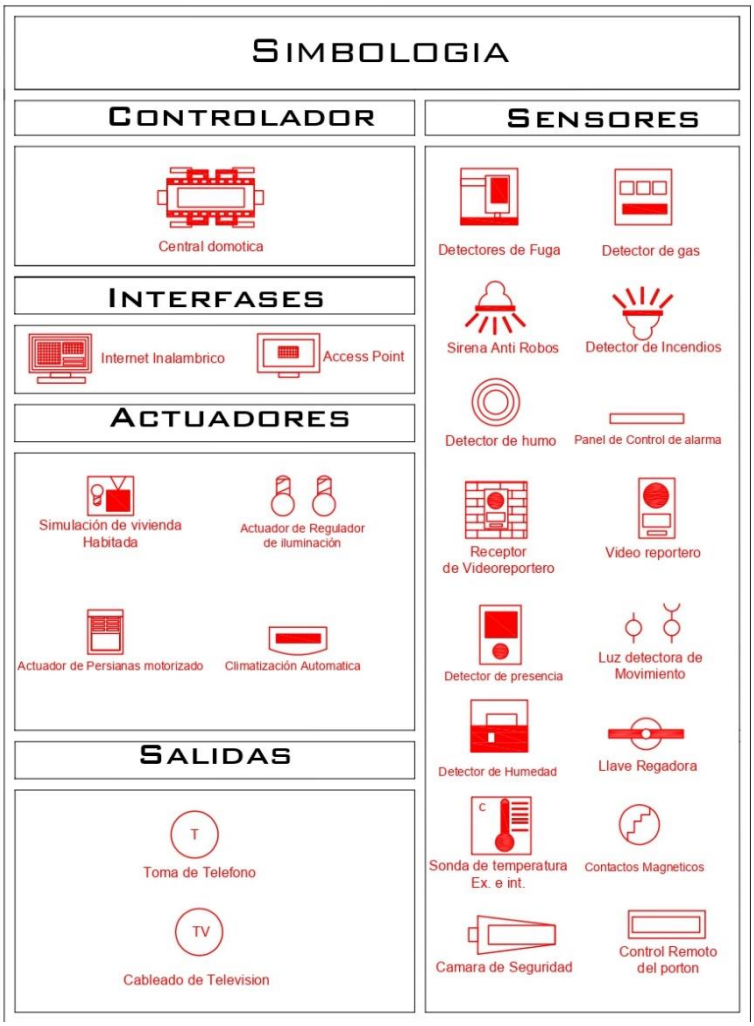
Capítulo V: Pruebas de Validación

Pruebas de funcionamiento

Para las pruebas de funcionamiento, realizamos el esquema de una casa domótica con las normativas KNX (Estándar ISO/IEC 14543-3), con los componentes que se plantean utilizar, con esto comprobaremos el funcionamiento del software, entregando los códigos correspondientes para cada elemento de la casa domótica.

Figura 36

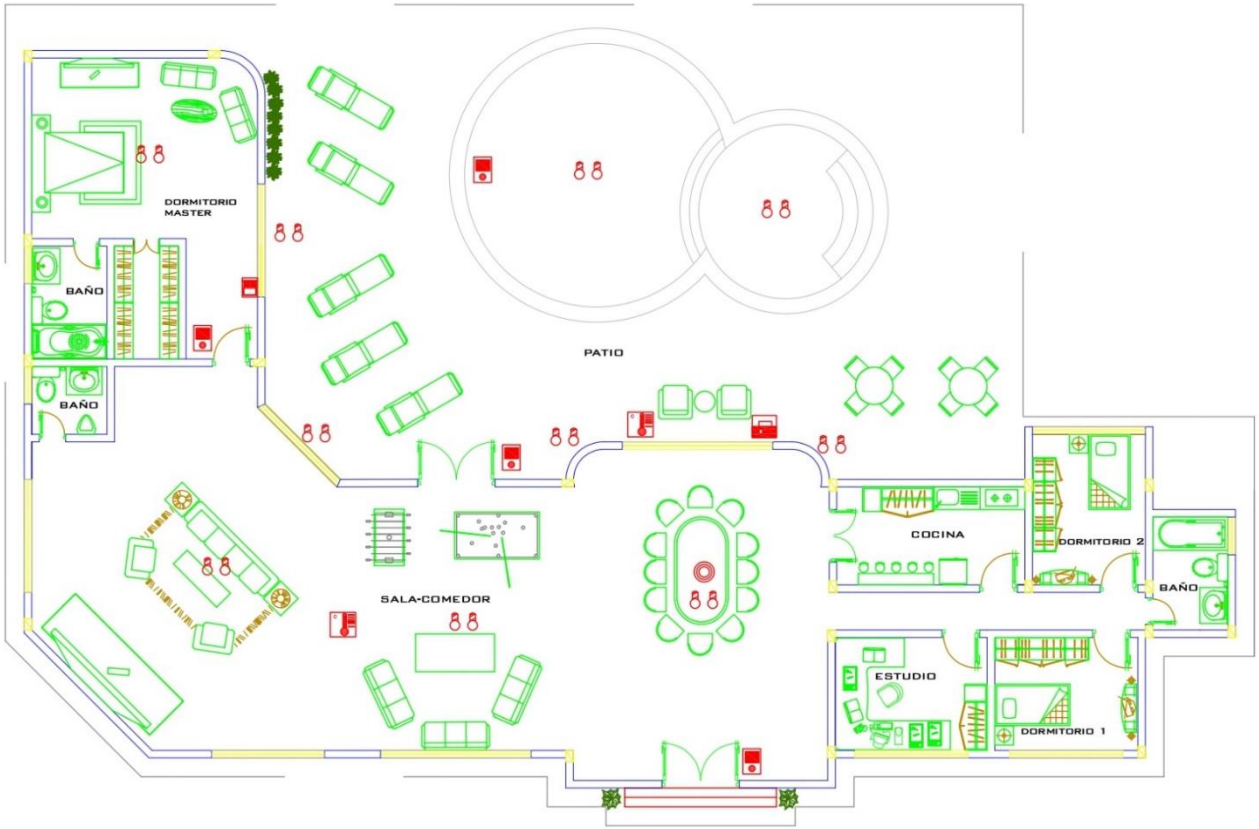
Simbología de la normativa KNX.



Nota: Simbología KNX para el diseño de planos domóticas.

Figura 37

Esquema casa domótica.



Nota: Esquema de la casa domótica, donde se puede apreciar los sensores y actuadores a utilizar en la prueba de funcionamiento.

Como se muestra en la Figura 37, se encuentra la implementación de distintos sensores y actuadores, en 3 diferentes áreas de la casa domótica, como es el dormitorio master, la sala-comedor y el patio. Dando así la distribución de los sensores y actuadores a diferentes controladores, en la siguiente tabla, se visualizará como se realiza la distribución de estos sensores y actuadores, tanto como el id que se dará y los pines en los cuales serán conectados.

Tabla 11

Elementos para la implementación de las pruebas de funcionamiento

Dormitorio Master				
ID	**ESP8266A**	**Raspberry Pi**	**ESP8266B**	**Pin**
FocoD1	X			D0
PersinaD1	X			D1
PresenciaD1		X		8
AlexaB				
Patio				
ID	**ESP8266A**	**Raspberry Pi**	**ESP8266B**	**Pin**
PresenciaP1	X			D2/D3
PresenciaP2		X		10
FocoP1		X		12
FocoP2		X		13
FocoP3		X		15
FocoP4		X		16
FocoP5		X		18
FocoP6		X		19

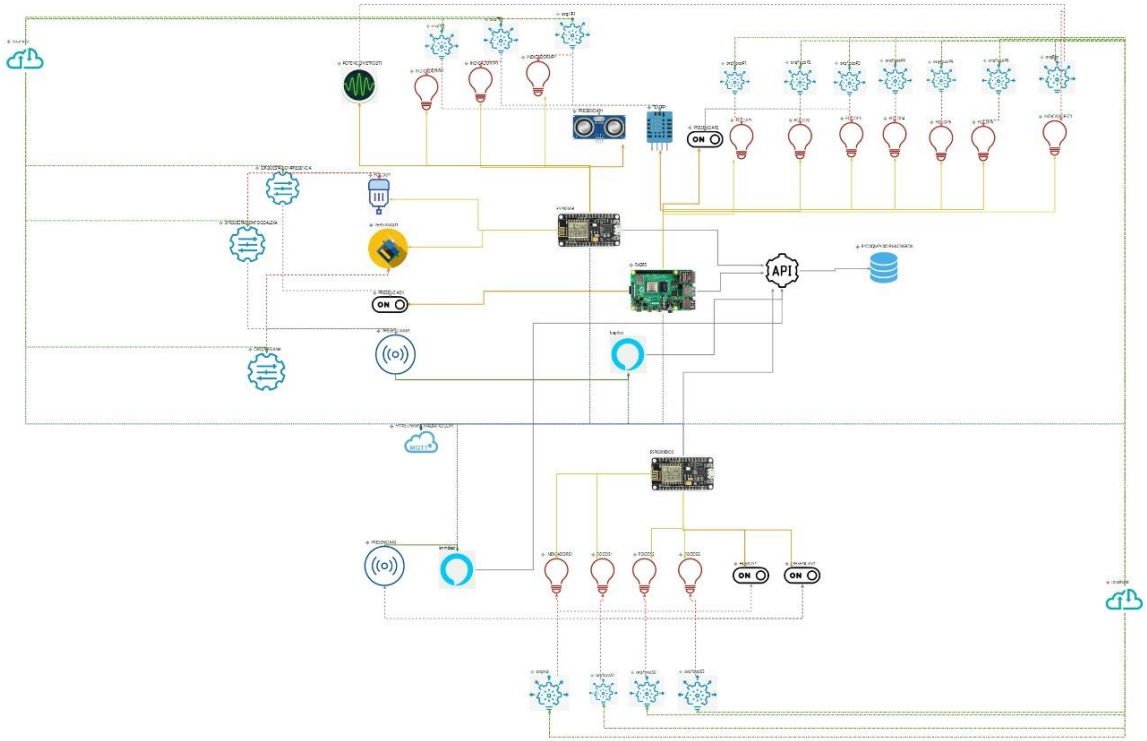
TemperaturaP1		X		11
HumedadP1		X		17
IndicadorPP1	X			D5
IndicadorTP1	X			D6
IndicadorHP1	X			D7
Sala Comedor				
ID	**ESP8266A**	**Raspberry Pi**	**ESP8266B**	**Pin**
FocoS1			X	D3
FocoS2			X	D4
FocoS3			X	D5
HumoS1			X	D0
PresenciaS1			X	D2
InidadorS1			X	D1
PotenciometroST1	X			A0
IndicadorST1		X		22
AlexaA				

Nota. En esta tabla se muestran todos los elementos usados para las pruebas de funcionamiento y los controladores a cuáles se los conecta con sus respectivos pines.

En el editor gráfico realizaremos la programación tomando en cuenta los controladores y los elementos detallados en la Tabla 11. Obteniendo los distintos programas que proporciona el sistema.

Figura 38

Programa final de la prueba del DSL.



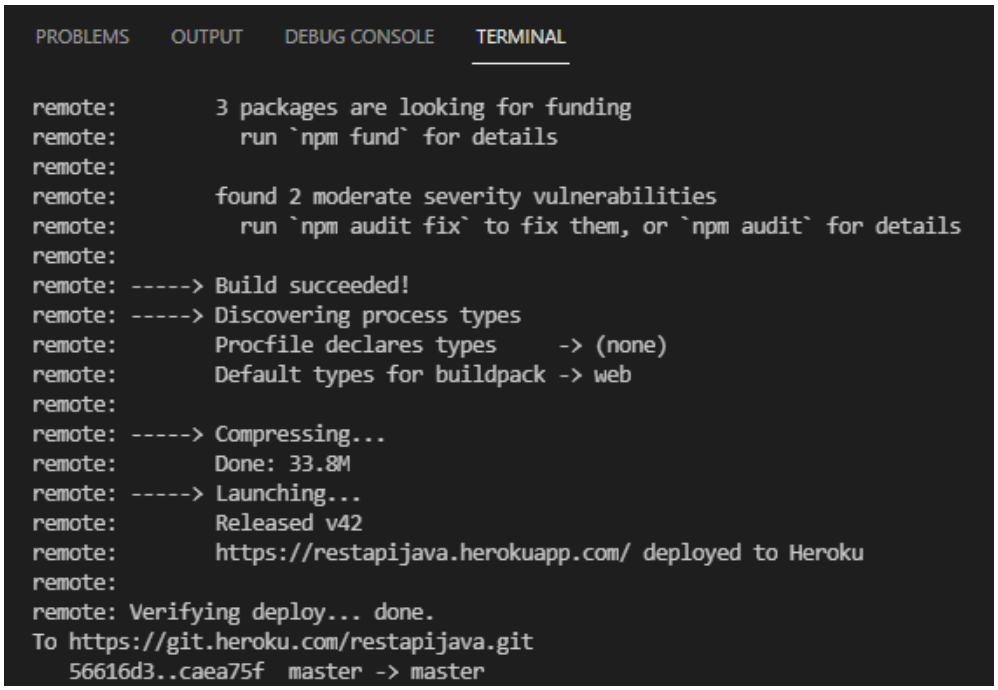
Nota: Diseño del editor gráfico realizada en la aplicación propuesta.

Realizaremos la carga de cada uno de los programas y comprobaremos el funcionamiento del sistema, observando los datos enviados en los distintos controladores, tanto como en la base de datos y en los servicios REST.

Se cargará los servicios REST en la plataforma Heroku, y la base de datos en Clever Cloud, como se puede mostrar en las Figuras 39 y Figura 40 respectivamente.

Figura 39

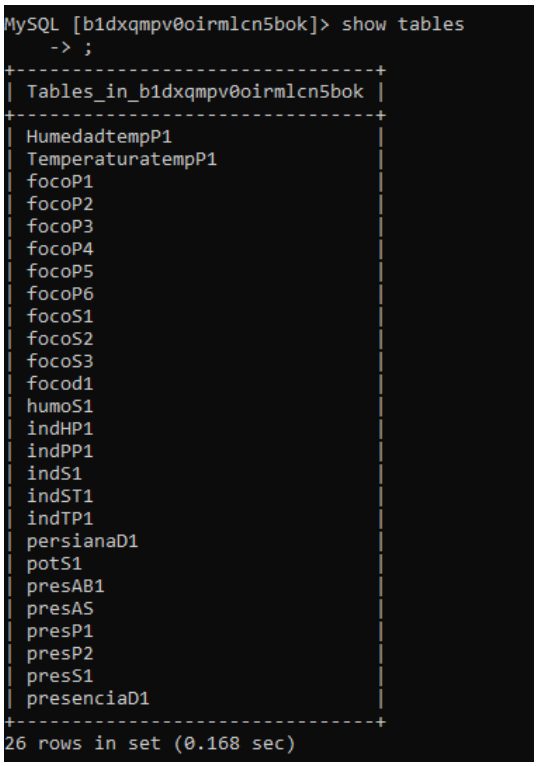
Carga de datos de manera local a la plataforma de heroku.



Nota: Proceso para cargar los datos “Servicios REST” de manera local por medio de la plataforma heroku.

Figura 40

Tablas creadas en Clever Cloud.

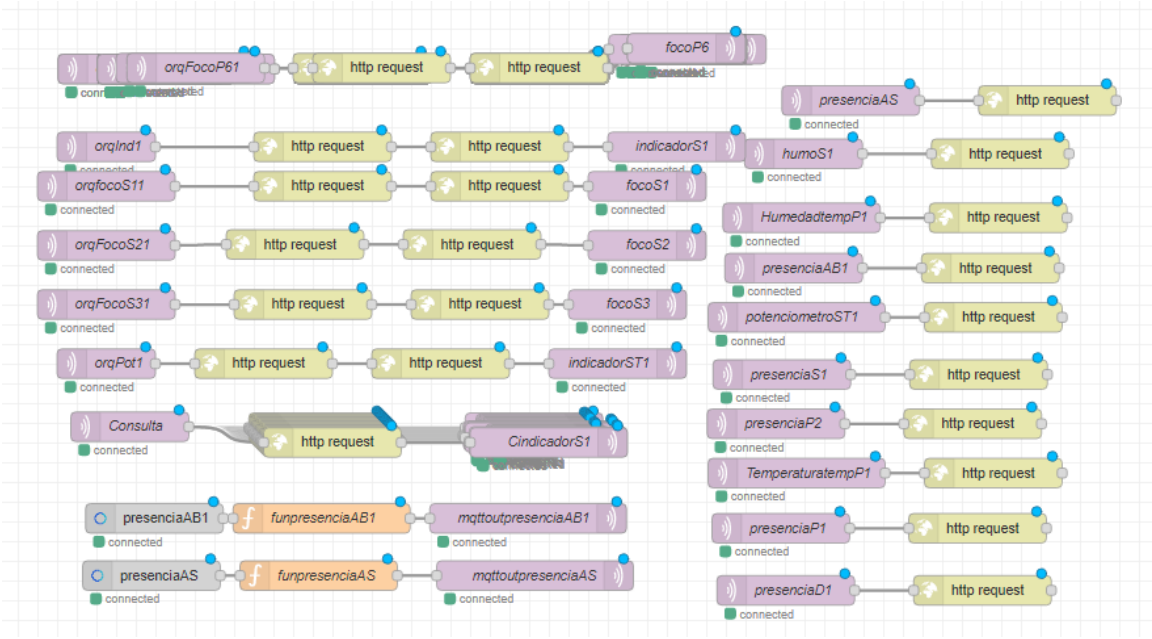


Nota: Visualización de las tablas creadas en Clever Cloud después de utilizar el programa .sql entregado por el DSL.

Al subir el archivo para Node-RED, tendremos que verificar que los datos servidor mqtt con el que vamos a trabajar. Ingresando si hace falta el usuario y contraseña de dicho servidor. *Una* vez verificando el servidor organizamos los bloques de envío y recepción de datos, como se muestra en la Figura 41.

Figura 41

Diagrama creado en Node-RED

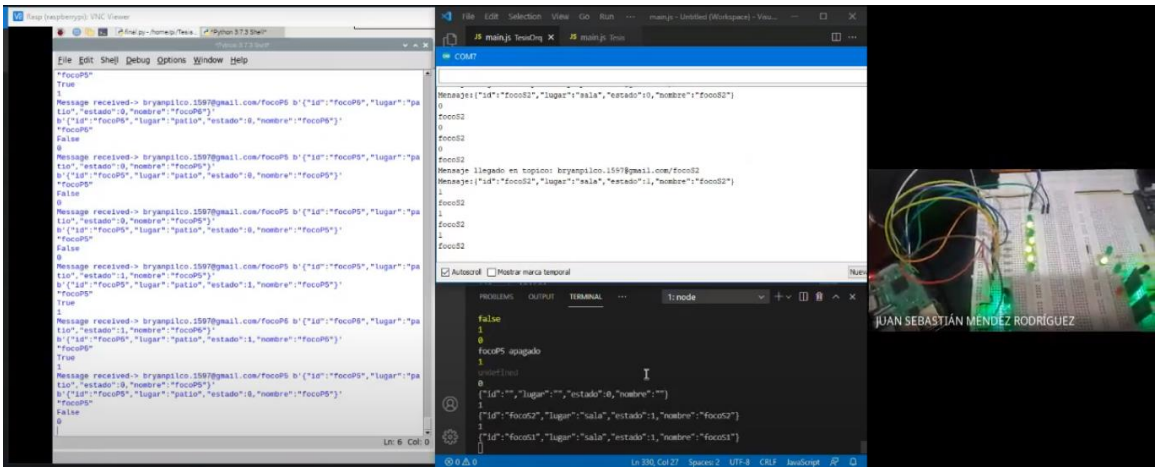


Nota: Resultado de cargar el archivo .json a Node-RED, desplegando así todos los elementos necesarios para el funcionamiento del mismo.

Una vez cargada los programas de arduino y de phyton se procede a realizar las pruebas de funcionamiento del sistema, verificando que el envío y recepción de datos se realiza utilizaremos los terminales y monitores seriales de cada dispositivo, esto lo observaremos en la Figura 42

Figura 42

Integración y visualización del funcionamiento del sistema.



Nota: Visualización del envío y recepción de los datos en los diferentes terminales y monitores seriales que utiliza cada sistema.

Por medio de la dirección web que se proporciona por los servicios REST se procede a revisar si los datos de los sensores llegan, en el ejemplo utilizamos el DTH11 para observar el valor de temperatura en ese momento, esto se muestra en la Figura 43.

Figura 43

Comprobación de recepción de la base de datos.

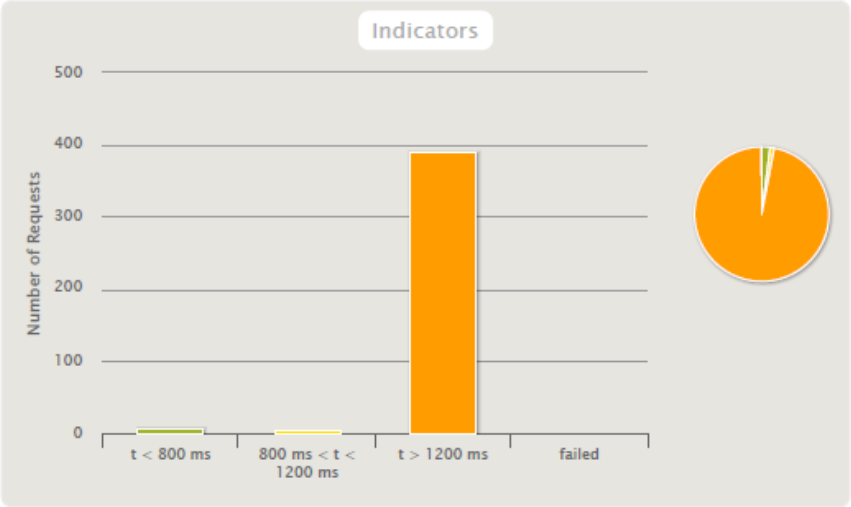
Nota: Ingreso del url para comprobar que la información enviada por el DHT11 sea haya guardado en la base de datos.

Pruebas de carga

Para las pruebas de carga utilizamos el software libre Gatling permitiendo obtener las gráficas de rendimiento del sistema, el entorno de desarrollo IntelliJ IDEA y la herramienta Maven que permiten realizar el escenario en el cual se va a dirigir el sistema. Esta prueba contara con un escenario que interactuaran con el sensor PresenciaD1 y el actuador PersianaD1, donde se realizara la petición GET y POST para cada uno, este escenario será estresado con 6 distintas cantidades de usuarios, los cuales son: 100, 300, 400, 425, 450, 500. Dando como resultado:

Figura 44

Prueba de Carga con 100 usuarios.



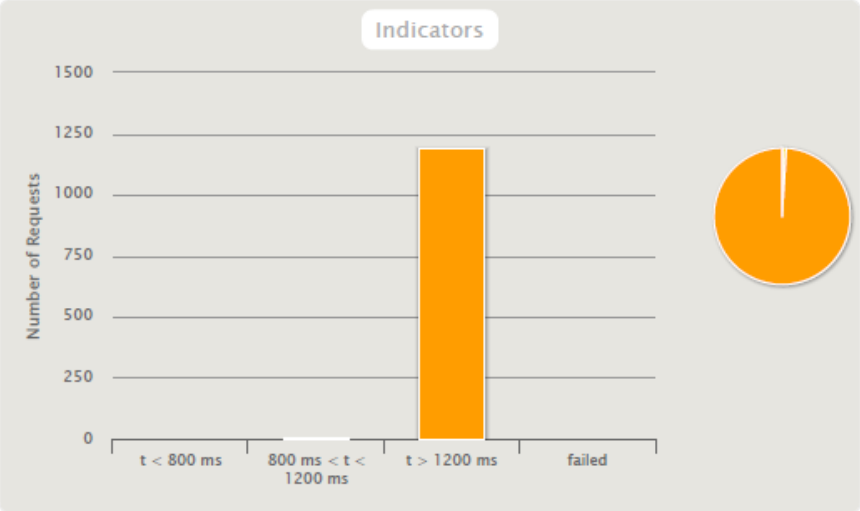
Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 100 usuarios.

Como se puede observar en la Figura 44, existe un error de 0%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 100 muestra y una media del tiempo de respuesta de 3382 ms.

Analizando la Figura 45, se puede identificar que existe un error de 0%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 300 muestra y una media del tiempo de respuesta de 17381 ms.

Figura 45

Prueba de carga con 300 usuarios.

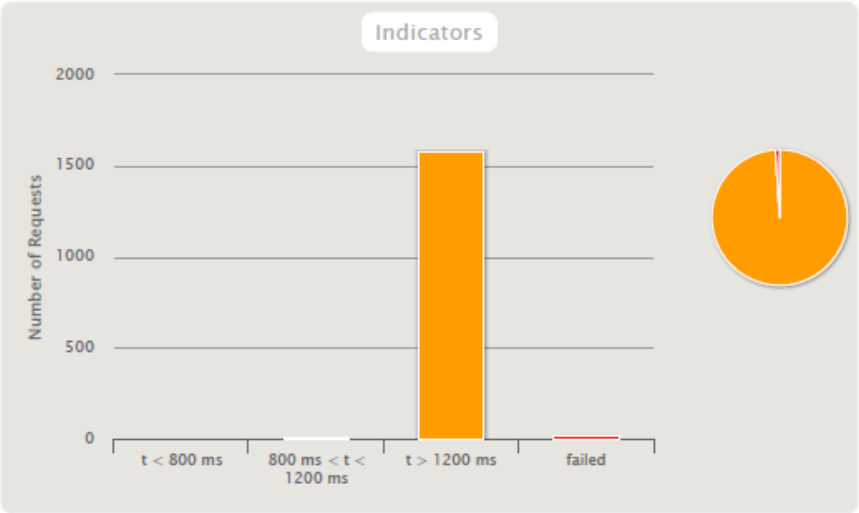


Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 300 usuarios.

Observando la Figura 46, se identificó que existe un error de 1%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 400 muestra y una media del tiempo de respuesta de 24414 ms

Figura 46

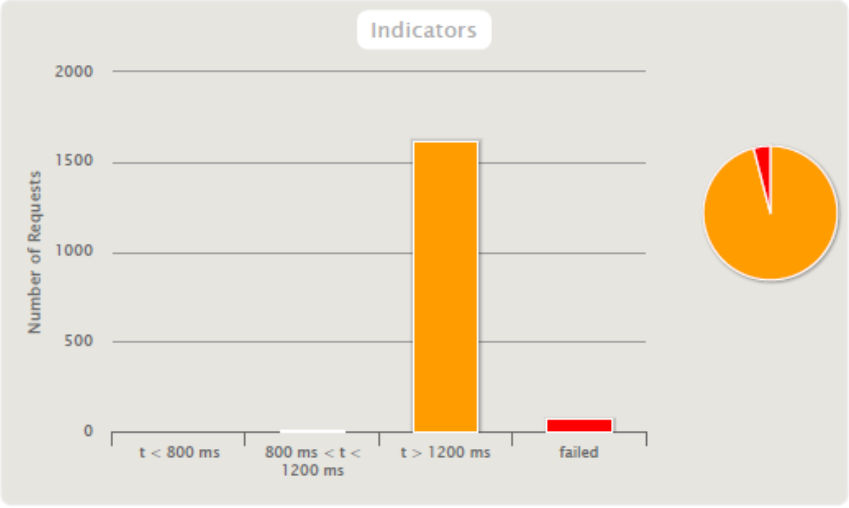
Pruebas de carga con 400 usuarios



Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 400 usuarios.

Figura 47

Prueba de carga con 425 usuarios.

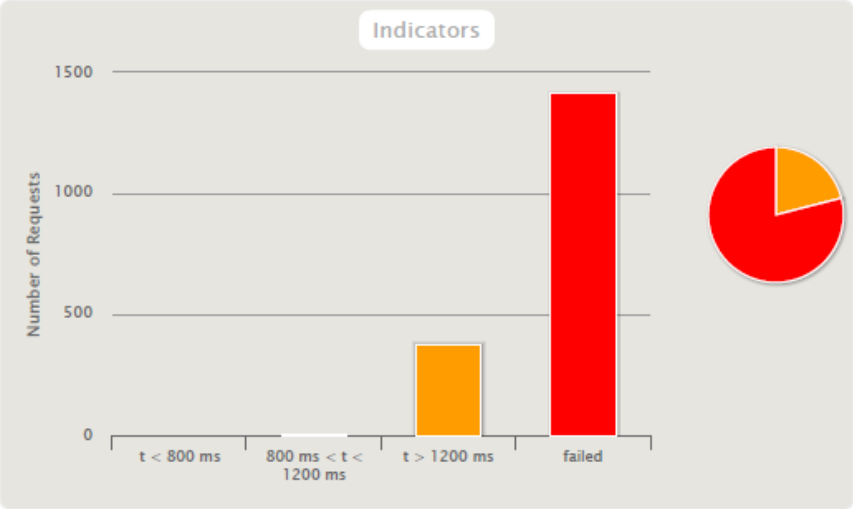


Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 425 usuarios.

Mediante la Figura 47, se extrajo los datos y existe un error de 4%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 425 muestra y una media del tiempo de respuesta de 26041 ms.

Figura 48

Prueba de carga con 450 usuarios.

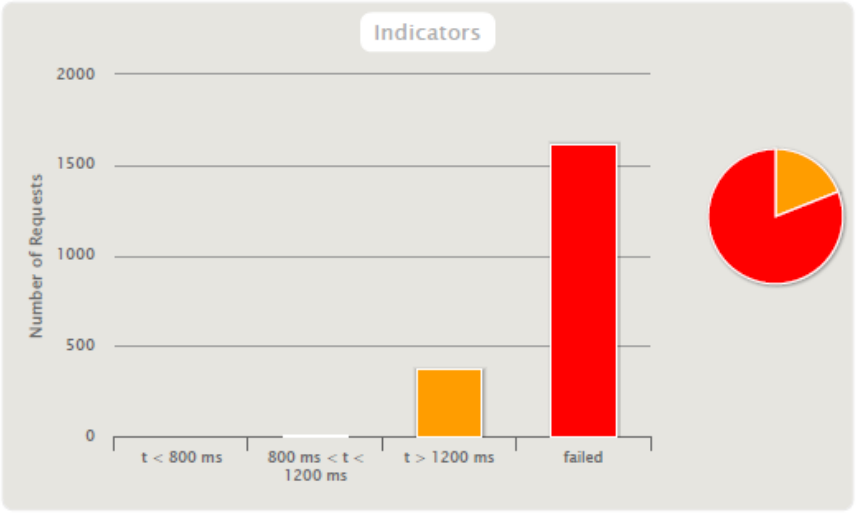


Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 450 usuarios.

Como se puede observar en la Figura 48, existe un error de 79%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 450 muestra y una media del tiempo de respuesta de 27151 ms.

Figura 49

Prueba de carga con 500 usuarios.



Nota: Gráfica resultante al momento de estresar el sistema con un ingreso simultaneo de 500 usuarios.

Observando la Figura 49, se aprecia que existe un error de 81%, dando a entender que las peticiones realizadas al servidor fueron exitosas durante las 500 muestra y una media del tiempo de respuesta de 27447 ms.

En la Tabla 12 se presenta un resumen de los resultados de las pruebas realizadas con los usuarios trabajados, la media en el tiempo de respuesta y su error, donde se analiza que mientas se incrementa el número de usuarios o muestras el incrementa el error y la media tiempo de respuesta.

Tabla 12

Resumen resultado pruebas con GATLING

Número de muestras	Media de tiempo de respuesta (ms)	% Error
100	3382	0
300	17381	0
400	24414	1
425	26041	4
450	27151	79
500	27447	81

Nota. En esta tabla se resumen los resultados de las pruebas de carga.

Puntos de función

Para realizar el cálculo de los puntos de función se ha utilizado el ejemplo realizado en la sección Pruebas de Funcionamiento, donde se enlista los materiales y archivos que fueron requeridos.

Para el cálculo de puntos de función es necesario definir un nivel de complejidad que va a tener el sistema, para el estudio del caso se determinó un nivel de dificultad simple. El cálculo del factor de peso se muestra en la Tabla 13.

Tabla 13

Cálculo del factor de peso

	Factor de Peso				
	Parámetros de Medida (1)			Contador (2)	Total Multiplicador (1)*(2)
Factores Funcionales de Peso	Simple	Media	Compleja		
N° Entrada de usuario	7	10	15	1	7
N° Salida de usuario	5	7	10	0	0
N° Consultas Usuario	3	4	6	1	3
N° Archivos Lógicos Internos (tablas)	4	5	7	1	4
N° Interfaces externas	3	4	6	6	18
					Factores de Peso = 32

Nota. En esta tabla se muestran los diferentes puntajes a considerar para calcular el factor de peso.

Una vez obtenido el factor de peso, es necesario calcular un factor de ajuste el cual se lo determina mediante niveles de influencia, los cuales tendrán un valor entre 0 y 5. Y si cálculo se muestra en la Tabla 14.

Tabla 14

Cálculo del factor de ajuste

Factor de Ajuste	Puntaje
Comunicación de Datos	5
Procesamiento Distribuido	4
Objetivos de Rendimiento	1
Configuración del equipamiento	1
Tasa de transacciones	0
Entrada de Datos en Línea	5
Interfaces con el usuario	3
Actualizaciones en Línea	3
Procesamiento Complejo	3
Reusabilidad del Código	4
Facilidad de Implementación	3
Facilidad de Operación	3
Instalaciones Múltiples	3
Facilidad de Cambios	3
Factor de Ajuste	41

Nota. En esta tabla se muestran los diferentes puntajes para calcular el factor de ajuste.

Una vez obtenido los valores finales del factor de ajuste y del factor de peso, se realiza el cálculo de los puntos de función ajustado, donde tendremos:

$$PFA = 32 * [0.65 + (0.01 * 41)]$$

$$PFA = 33.92$$

$$PFA \approx 34$$

Para el cálculo del número de horas que a un desarrollador le tomaría construir todo este sistema sin el uso de la aplicación presentada como proyecto de titulación, se realiza de la siguiente manera:

Tabla 15

Tabla de selección de lenguaje para los puntos de función

Lenguaje	**Horas PF promedio**	**Líneas de código por PF**
Ensamblador	25	300
COBOL	15	100
Lenguajes de 4ta Generación	8	20

Nota. En esta tabla se muestras las diferentes horas y líneas de código promedio de acuerdo a cada tipo de lenguaje.

$$H/H = PFA * Horas\ PF\ promedio$$

$$H/H = 34 * 8$$

$$H/H = 272\ horas\ \ hombre$$

Se ha calculado el tiempo estimado que una persona se demoraría en realizar el sistema, que son 272 horas de trabajo, mientras que, usando la aplicación desarrollada, el tiempo estimado de trabajo sería de 3 horas.

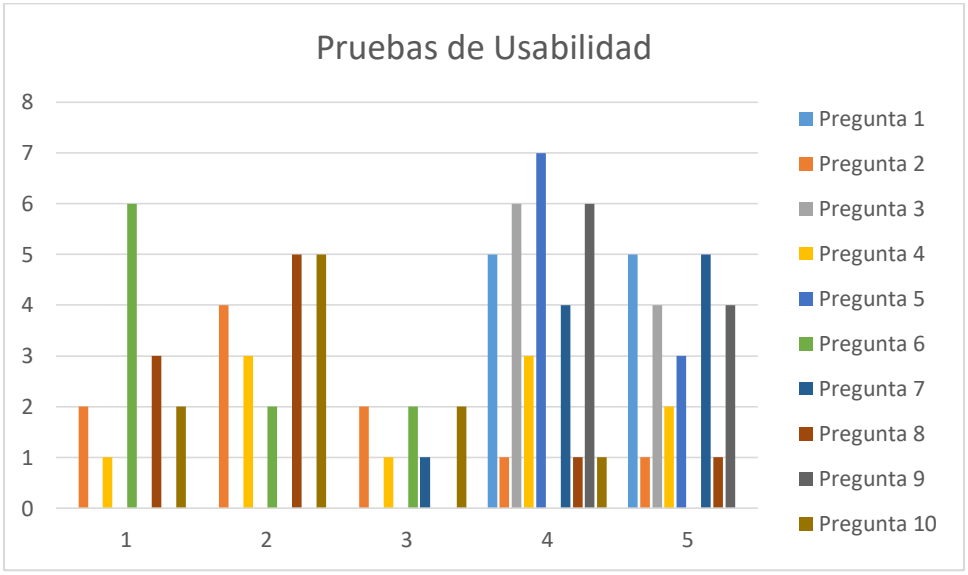
Pruebas de usabilidad

La prueba de usabilidad se realizó a partir de 10 preguntas, las cuales son puntuadas del 1 al 5, teniendo en cuenta que 1 significa total desacuerdo y 5 significa total acuerdo.

Esta prueba se realizó con 10 estudiantes de la Universidad de las Fuerzas Armadas, todos de la carrera de Ing. Electrónica, permitiendo conocer la valoración de la aplicación desarrollada. En el Anexo A, se presenta la encuesta realizada, y el desarrollo de la misma. A continuación, se presentará los resultados obtenidos para cada pregunta.

Figura 50

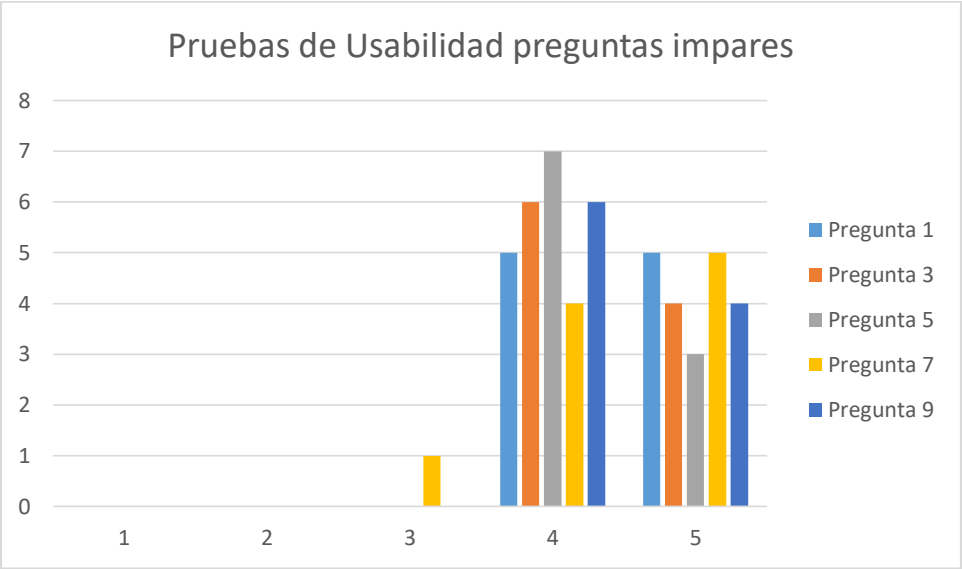
Resultados de las encuestas realizadas para la Prueba de Usabilidad



Nota: Gráfica resultante de las preguntas de la prueba de usabilidad.

Figura 51

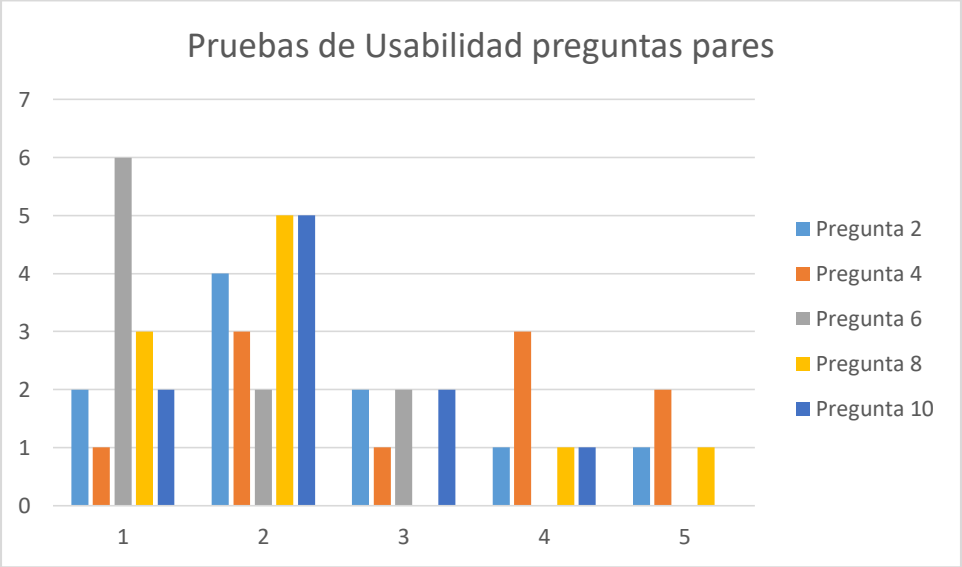
Resultados de las preguntas impares para la Prueba de Usabilidad



Nota: Gráfica resultante de las preguntas impares de la prueba de usabilidad.

Figura 52

Resultados de las preguntas pares para la Prueba de Usabilidad.



Nota: Gráfica resultante de las preguntas pares de la prueba de usabilidad.

Obtenidos estos resultados se realiza el algoritmo para obtener el valor de usabilidad de la aplicación, para esto se procede a obtener el promedio de cada pregunta, para las preguntas impares se tomará el valor obtenido del promedio y se restará 1, mientras que para las preguntas pares el promedio será restado de 5, obtenido el resultado se multiplicará por 2.5. Esto se presenta en la Tabla 16.

Tabla 16

Promedio y resultado de la prueba de usabilidad

Usuario	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Puntaje SUS
1	4	2	4	5	5	3	4	2	4	2	67,5
2	4	3	5	2	4	2	5	2	4	2	77,5
3	4	2	4	5	5	1	5	1	4	1	80
4	5	1	5	2	5	1	5	1	5	2	95
5	4	4	4	4	4	3	4	1	4	4	60
6	5	1	5	1	5	1	4	2	4	2	90
7	5	3	4	3	5	1	4	4	5	3	72,5
8	5	2	5	2	5	1	5	5	5	2	82,5
9	5	2	4	4	4	1	3	2	4	1	75
10	4	5	4	4	5	2	5	2	5	3	67,5
Promedio											76,75

Nota. En esta tabla se muestra el puntaje de cada pregunta y el promedio de usabilidad.

Se obtiene un valor de 76.75%, lo cual nos indica que la aplicación está en el rango favorable para su uso, a pesar de esto los casos que permitirán aumentar el porcentaje es realizar una capacitación de la aplicación debido a que varios encuestadores indican que es complejo su uso por los distintos conocimientos técnicos, esto se ve reflejado en la Figura 52 donde muestra una distribución inconsistente de los valores de las preguntas pares.

Capítulo VI: Conclusiones y Recomendaciones

Conclusiones

Tras la investigación de las múltiples tecnologías se desarrolló un metamodelo en el cual se tomó en cuenta todos los factores que afectan al sistema, ya sean estos físicos (sensores, actuadores, controlador, asistente virtual) o virtuales (protocolo mqtt, servicios REST, bróker de mensajería, base de datos), creando así los modelos necesarios para la estructura funcional del metamodelo, conociendo cual es la clase principal y secundaria del sistema, tomando en cuenta los atributos que deberán trabajar en cada una de ellas. También es importante tener en cuenta los niveles de abstracción para poder simplificar la estructura del modelo.

Se desarrolló un sistema que permita la comunicación de distintos elementos en la nube entre colas de mensajes y servicios web, almacenando información, realizando un análisis de los datos los cuales son objetos tipo JSON y ejecutando los eventos deseados por el desarrollador, para así llegar a la integración de un sistema IoT.

Se implementó un DSL que permite a los desarrolladores definir un sistema IoT, sin tener que profundizar en aspectos específicos de lenguajes de programación para las diferentes plataformas, siendo este una herramienta grafica para estandarizar la estructura de los dispositivos IoT, los enlaces a los servicios REST, el puente con los protocolos MQTT y el orquestador que define la lógica que tomara el sistema IoT. Obteniendo así de forma semi automática los códigos del orquestador, servicios REST, MySQL, Arduino, Python (Raspberry Pi) y Node-RED. Se dice que estos programas se obtienen de forma semi automática, ya que el programador tiene que implementar la infraestructura con los parámetros que se plantea para su aplicación específica.

Se realizó varias pruebas de estrés del sistema, dando a conocer que, al interactuar con 400 usuarios al mismo tiempo, existirá un error de 1%, pero esto no es un problema debido a que el sistema en el ambiente que se va a trabajar no se ejecutaran acciones que estresen al sistema, es decir la cantidad datos enviados de forma simultanea no llegara a valores tan altos.

Por medio de las pruebas de usabilidad se dio a conocer que el sistema tiene un 76.75%, lo cual es un valor aceptable para la implementación del sistema, hay que tomar en cuenta que este valor puede variar.

Realizadas las pruebas de puntos de función se ha observado que un desarrollador sin el uso de nuestra aplicación se demoraría un tiempo de 272 horas, sin embargo, haciendo de la misma el tiempo de trabajo sería un aproximado de 3 horas, reduciendo así significativamente las horas de trabajo que representa para un desarrollador programar una aplicación en específico.

Recomendaciones

Al momento de implementar los sistemas que se obtuvieron mediante la aplicación, se recomienda contratar servidores externos, o usar servidores gratuitos en la nube, para que así la comunicación no sea solo de manera local, sino también pueda dar de forma remota.

Al utilizar la aplicación, se recomienda una visión de manera general del sistema que se quiera implementar en la misma, para así reducir tiempos de programación, hay que tomar en cuenta que para la orquestación el desarrollador ya debe tener preparada la función boolena que interactuara con el o los actuadores.

Instalar todas las librerías de mqtt, sensores, actuadores y controlador en arduino y raspberry, también instalar las librerías de MySQL, mqtt, delay en Visual Studio Code,

para el desarrollo de servicios REST y orquestación, de la misma forma verificar que la versión de jdk sea la 1.8 permitiendo así la utilización del software Eclipse.

Trabajos Futuros

Para los trabajos futuros se propone:

- Realizar un encapsulado del proyecto, es decir, crear un plug-in que se pueda instalar de manera rápida en cualquier ordenador, mejorando así el tiempo de carga del programa, a parte se desea mejorar el DSL actual, para que, al momento de compilar el proyecto, en lugar de generar los archivos separados de cada tecnología, este se encargue de publicarlos y subirlos a sus plataformas correspondientes.
- Implementar protocolos de seguridad como cuentas de usuarios para mayor privacidad, dando mecanismos y certificados de autenticidad, como de dos vías o dos pasos, también realizar un encriptado de información dado que el proyecto solo envía datos planos.
- Implementar un DSL que entregue front ends para así tener una mayor visualización de la transmisión de datos, y obtener un mayor control sobre los controladores, permitiendo enviar información que sea beneficioso para el usuario.
- Ampliar el número de tecnologías utilizadas para así manejar una mayor cantidad de controladores (ASUS Tinker Board S, Banana Pi M64, Orange Pi Plus, etc), sensores (magnéticos, de contacto, químicos), actuadores (hidráulicos y neumáticos) y asistentes virtuales (Asistente de Google).

Acrónimos

- CPE. Procesamiento de Eventos Complejos
- CPS. Sistemas Ciberfísicos
- DSL. Lenguaje Especifico de Dominio
- EDA. Arquitectura dirigida a eventos
- EMF. Eclipse Modeling Framework
- HTTP. Hypertext Transfer Protocol
- IoT. Internet de las Cosas
- IT. Tecnologías de la Información
- MDE. Ingeniería Dirigida por Modelos
- MQTT. Messege Queuing Telemetry Transport (Transporte de telemetría)
- OT. Tecnologías de Operación
- REST. representational state transfer (Transferencia de estado representacional)
- SLR. Revisión Sistemática de la Literatura
- SMS. Mapeo Sistemático de la Literatura.
- SOA. Arquitectura Orientada a Servicios
- WoT. Web de las Cosas

Referencias Bibliográficas

Abrahão, S., Bourdeleau, F., Cheng, B., Kokaly, S., Paige, R., Stöerrle, H., & Whittle, J. (2017). User Experience for Model-Driven Engineering: Challenges and Future Directions. *IEEE*.

Alarcon, R., Saffie, R., Bravo, N., & Cabello, J. (2015). REST web service description for graph-based service discovery. *In International Conference on Web Engineering* , 461-478.

Amrani, M., Gilson, F., Debieche, A., & Englebert, V. (2017). Towards User-centric DSLs to Manage IoT Systems.

Atkinson, C., & Kühne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE Software*.

Baheti, R., & Gill, H. (2011). Cyber-physical systems. *The impact of control technology*, 161-166.

Basamasi, Y. (2014). The working principle of an Arduino. *IEEE, International confrence on electronics, computer and computation*, 1-4.

Bettini, L. (2016). Implementing Domain-Specific Languages with Xtect and Xtend. *Packt Publishing Ltd.*

Bračevac, O., Salvaneschi, G., Erdweg, S., & Mezini, M. (2019). Type-safe Polyvariadic Event Correlation.

Cao, K., Li, R., & Wang, F. (2013). The Research on CEP Based on Query Rewriting for CPS.

Carbonell, J., Medeiros, B., Silva, P., Rodrigues, E., & Bernardino, M. (2017). Proposta de Linguagem de Modelagem Gráfica com o Sirius: Uma Versno Open Source da DSL Canopus.

Eassa, A., Elhoseny, M., El-Bakry, H., & Salama, A. (2018). NoSQL Injection Attack Detection in Wev Applications Using RESTful Service. *Programming and Computer Software*, 435-444.

Espada, J., Martínez, O., & Pelayo, C. (2011). Virtual Objects on the Internet of Things.

Frankel, D. (2003). Model Driven Architecture: Applying MDA to Enterprise Computing. *Wiley*.

Karande, A., Karande, M., & Meshram, B. (2011). Choreography and Orchestration using Business Process Execution Language for SOA with Web Services. *International Journal of Computer Science Issues*, 224-232.

Kodali, R., & Soratkal, S. (2016). MQTT based home automation system using ESP8266. *IEEE Region 10 Humanitrian Technology Conference*, 1-5.

Laskey, K., & Laskey, K. (2009). Service oriented architecture. *Wiley Interdisciplinary Reviews: Computational Statistics*, 101-105.

Lee, E. (2008). Cyber physical systems: Design challenges. *IEEE*, 363-369.

Lee, J., Bagheri, B., & Kao, H. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems.

Lewis, J., & Fowler, M. (2014). Microservices. *https://martinfowler.com/articles/microservices.html*.

Ludewig, J. (2003). Models in Software Engineering – An Introduction. *Softw Syst Model*, 5 - 14.

Madakam, S., Ramaswamy, R., & Tripathi, S. (2015). Internet of Things . *Journal of Computer and Communications*, 164-173.

Márquez, J. E. (2018). *Una arquitectura orientada a servicios y dirigida por eventos para el control inteligente de UAV's multipropósito.*

Martinez, D. (2014). Herramienta para la generación automática del código fuente para aplicaciones con arquitectura modelo vista controlador (MVC) bajo desarrollo dirigido por modelos textuales (MDD.

Martínez, D. (2014). HERRAMIENTA PARA LA GENERACIÓN AUTOMÁTICA DEL CÓDIGO FUENTE PARA APLICACIONES CON ARQUITECTURA MODELO VISTA CONTROLADOR (MVC) BAJO DESARROLLO DIRIGIDO POR MODELOS TEXTUALES (MDD).

Miguez, I., Fernández, T., Fraga, P., & Castedo, L. (2018). Desing, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes. *Sensors*.

Minzhi, Y., Hailong, S., Xudong, L., Ting, D., & Xu, W. (2014). Delivering Web service load testing as a service with a global cloud.

Miori, V., & Russo, D. (2014). Domotic evolution towards the IoT. *IEEE 28th International Conference on Advanced Information Networking and Applications Workshops* , 809-914.

Montenegro, C., Cueva, J., Martinez, O., & Gaona, P. (2010). Desarrollo de un lenguaje de dominio específico para sistemas de gestión de aprendizaje y su herramienta de implementación “KiwiDSM” mediante ingeniería dirigida por modelos.

Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An analysis of public REST web service APIs. *IEEE Transactions on Services Computing.*

Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. *In CLOSER*, 137-146.

Rahmani, A., Guia, T., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., & Liljeberg, P. (2018). Exploiting smart e-Healt gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*.

Rodenas, F. (2010). Metodologies de Desenvolupament Dirigides per Models.

Rodrigues, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*.

Salman, A., AL.Jawad, M., & Al Tameemi, W. (2020). Domain-Specific Languages for IoT: Challenges and Opportunities.

Seiger, R., Huber, S., & Schlegel, T. (2016). Towards an Execution System for Self-healing Workflows in Cyber-Physical Systems.

Shi, J., Wan, J., & Suo, H. (2011). A Survey of Cyber-Physical Systems. *IEEE*.

Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., & Lennartson, B. (2017). An event-driven manufacturing information system architecture for Industry 4.0. *International journal of production research*.

Ulvi, M., & Ozdemir, S. (2018). CEP Rule Extraction from Unlabeled Data in IoT.

Wortmann, A., Combemale, B., & Barais, O. (2017). A Systematic Mapping Study on Modeling for Industry 4.0.

Zacheilas, N., Kalogeraki, V., Zygouras, N., Panagiotou, N., & Gunopulos, D. (2015). Elastic Complex Event Processing exploiting Prediction. *IEEE*.

Zhao, C., Jegatheesan, J., & Loon, S. (2015). Exploring IoT application using raspberry pi. *International Journal of Computer Networks and Applications*, 27-34.

Anexos