



UNIVERSIDAD DE MENDOZA
FACULTAD DE INGENIERÍA

TESIS DE
MAESTRÍA EN TELEINFORMÁTICA

WSN para alerta y monitoreo de
incendios forestales e
integración con sistemas de
predicción

Autor: Ing. Rodrigo Atilio Elgueta
Director: Dr. Miguel Mendez-Garabetti

Mendoza - 2023

Agradecimientos

A la Facultad de Ingeniería de la Universidad de Mendoza, al grupo humano que trabaja allí y que constantemente alienta a sus alumnos y profesores a continuar con su desarrollo profesional.

Agradezco mucho el apoyo y paciencia de mi director de tesis y amigo Miguel Mendez Gabaretti.

A mi familia por ser los principales promotores de mis sueños, sobre todo a mi madre por estar dispuesta a acompañarme a lo largo de todos mis procesos educativos y de mi vida.

Gracias a Dios y a la vida por la bendición diaria de estar acompañado por mis seres queridos y disfrutar de mi profesión.

A todas las personas que me apoyaron y colaboraron en la realización de esta Tesis.

Resumen

Los incendios forestales generan grandes pérdidas y daños anuales alrededor del mundo. En este contexto, el presente trabajo, plantea el desarrollo, planificación y despliegue de una red de sensores inalámbrica (WSN por sus siglas en inglés) y su infraestructura para la cuantificación de ciertas variables ambientales que permitan detectar incendios forestales de forma temprana. Esto permitiría, en primer lugar, anticipar situaciones críticas en relación al combate de incendios forestales, además de poder generar información útil para alimentar otros componentes de acción, como son los sistemas de monitoreo o predicción del avance de los incendios. Es decir, si se conoce el posible comportamiento de un incendio forestal una vez que éste ya se ha iniciado, se podrían tomar decisiones acertadas en el plan de acción de lucha contra el fuego. Para ello, se utilizan modelos de predicción de incendios forestales cuyo objetivo es disminuir incertidumbres: intentar conocer parámetros de entrada que determinan el comportamiento del modelo. Por lo tanto, el despliegue de una red inalámbrica de sensores permitiría la obtención de parámetros para inyectar en dichos sistemas. Asimismo, sin importar qué plataforma de recolección se utilice, el presente trabajo involucra la utilización del protocolo MQTT (Message Queue Telemetry Transport) a fin de enviar los datos censados hacia un Broker de mensajería que permite la suscripción de clientes para recibir los parámetros recolectados por la WSN e incorporarlos a una Base de Datos. Estos últimos elementos, forman parte de una infraestructura virtualizada mediante contenedores, particularmente se utilizó Docker, a fin de permitir su portabilidad, facilitar su implementación en plataformas cloud, soportar escalabilidad e incluso su despliegue y configuración necesaria para otros proyectos.

Índice de Contenido

Resumen	3
Índice de Contenido	4
Presentación	8
Capítulo I: Monitoreo de variables de incendios forestales mediante redes inalámbricas de sensores	9
Planteamiento del problema	9
Objetivos Planteados	13
Capítulo II: Marco Teórico y Antecedentes	15
Síntesis del capítulo	15
Incendios Forestales	16
El fuego y sus efectos sobre ecosistemas	16
Causales de incendios	18
Fases del incendio	20
Tipos de incendios.	20
La lucha del fuego en argentina	23
Datos actualizados al 2021 de incendios forestales en el País.	25
Incendios Forestales y el uso de la tecnología	26
Red Inalámbrica de Sensores	26
Arquitectura WSN	27
Nodos Inalámbricos.	28
Puertas de Enlace.	28
Estaciones Base.	29
Topologías de red.	29
Topología Estrella.	30
Topología en Malla.	30
Topología Híbrida Estrella-Malla.	31
Redes AD-HOC.	31
Tecnologías inalámbricas.	33
Redes Mesh.	35
IEEE 802.15.4.	37
ZigBee.	39
IEEE 802.11.	40
ESP8266	41
Arduino IDE.	42
Proyecto: Núcleo Arduino para chip ESP8266.	43

Placa de Desarrollo D1 Wemos Mini	43
Pinout WEMOS D1 Mini.	45
Programación en IDE Arduino.	46
Librería painlessMesh .	46
Limitaciones y advertencias de painlessMesh.	48
Captura de Datos	49
Captura de Datos mediante Sensores.	49
Sensor de Temperatura y Humedad: DHT11.	49
Sensor de Monóxido de Carbono MQ-7.	51
Destino de Datos Recolectados	52
Virtualización	52
Tipos de Virtualización.	53
Virtualización de servidores.	53
Virtualización de tipo 1.	54
Virtualización de tipo 2.	54
Virtualización de tipo 3 o híbrida.	55
Virtualización de red.	55
Virtualización de escritorios.	55
Virtualización y Cloud.	56
Proxmox.	56
Características de PROXMOX.	57
Contenedores	58
Docker	61
Docker Vs. LXC.	62
Portainer.	65
Infraestructura	65
MQTT.	65
Modelo de publicación y suscripción.	66
Protocolo de MQTT.	68
Suscripción de temas.	70
MQTT BROKER.	72
EMQX Dashboard	74
MariaDB	77
Características de MaríaDB.	77
Mecanismos de almacenamiento.	77
Mejoras de velocidad.	78
Mejoras en las pruebas.	78

Facilidad de uso.	78
Python	79
Eclipse Pahoo MQTT Python.	79
Layout	80
Apache.	80
PHP.	80
webdevops/php-apache-dev.	81
Despliegue de webdevops/php-apache-dev.	81
Google Cloud Platform.	82
API de Google Cloud.	82
Maps JavaScript API	83
Modelos de sistemas de predicción de comportamientos de Incendios forestales	84
Capítulo III: Desarrollo	87
WSN propuesta	87
Funcionamiento de la WSN y comunicación.	88
Programación de ESP8266 D1 Wemos Mini.	92
Programación del DHT11 mediante IDE Arduino.	93
MQ-7	94
Red mesh en esp8266 y envío de datos	94
Nodos de la Red Mesh.	95
Gateway.	98
Destino de los Datos	102
Despliegue de VM en Proxmox.	103
Despliegue de Docker y Containers.	104
Instalación de Docker.	104
Instalación de Portainer.	104
Deploy de Infraestructura.	105
Despliegue de EMQ para el proyecto.	105
Despliegue emqx/emqx4.2.3.	105
Despliegue de MariaDB para el proyecto.	106
Despliegue de Contenedor MaríaDB.	106
Construcción de la Base de Datos.	107
Despliegue de Código Python para el proyecto.	110
Despliegue de Python en contenedor.	111
Código a ejecutar.	111
Configuración de Apache.	114

Código del Layout.	114
Código de visión en Mapa.	121
App.php.	126
Conexion.php.	127
Info_Marcadores.php.	128
Marcadores.php.	128
Interacción con Sistemas de predicción	129
Capítulo IV: Resultados	134
Conclusiones	141
Bibliografía	143

Presentación

Este documento presenta una estructura coherente y clara. Comienza con el Capítulo I: “Monitoreo de variables de incendios forestales mediante redes inalámbricas de sensores”, donde se introduce la temática y se plantean los objetivos del presente trabajo, luego en el Capítulo II: “Marco Teórico”, se desglosan los componentes esenciales y se introducen las definiciones clave que servirán de base para el entendimiento del tema. A continuación, en el Capítulo III: “Desarrollo”, se profundiza en la implementación práctica de una Red Inalámbrica de Sensores (WSN) basada en el estándar IEEE 802.11. Además, se detalla el uso del protocolo MQTT para la transmisión de parámetros medidos, la incorporación de MariaDB como sistema de gestión de bases de datos para almacenar la información recabada, y la selección de Python como lenguaje de programación para la interfaz. Este capítulo, también incluye un ejemplo práctico de una interfaz de usuario construida utilizando PHP. Finalmente, en el Capítulo IV: “Resultados” se recopila las conclusiones y presenta evidencias que demuestran la funcionalidad de la infraestructura propuesta, basándose en experimentos realizados en un entorno de laboratorio a pequeña escala.

Capítulo I: Monitoreo de variables de incendios forestales mediante redes inalámbricas de sensores

Planteamiento del problema

Las zonas forestales son imprescindibles para la vida en el planeta. Además de ser parte fundamental en los ciclos de producción y distribución del agua, purifican el aire que respiramos al capturar bióxido de carbono y liberar oxígeno. También regulan la temperatura y la humedad, con lo que se equilibra el clima; proporcionan alimento, medicina y refugio a los seres vivos; y son fuente de materia prima en muchas actividades humanas.

Estos procesos vitales se ven amenazados por diversos factores ajenos a las actividades forestales como: la degradación de suelos, la deforestación, la tala inmoderada, los fuegos no controlados que están relacionados con otras actividades como la agricultura, la ganadería y el desarrollo urbano.

El fuego puede tener una influencia positiva en la naturaleza, pues ayuda a mantener la biodiversidad. Pero cuando se utiliza de forma irresponsable o se produce por alguna negligencia, puede convertirse en un incendio forestal de consecuencias devastadoras para el medio ambiente, incluso para la salud y seguridad de las personas (Comisión Nacional Forestal - México, 2010). En particular, los incendios forestales generan grandes pérdidas y daños anuales alrededor del mundo y se realizan grandes esfuerzos para mitigar el daño que producen.

En este sentido, Argentina posee un plan nacional de manejo del fuego (Decreto 746/2017, del Ministerio de Ambiente y Desarrollo Sustentable) y guías específicas respecto a cómo actuar ante la ocurrencia de incendios y métodos de prevención que fueron creadas en conjunto con servicios forestales de diferentes países. Dichas guías son eficientes en provincias que poseen territorio boscoso, donde se utiliza tecnología de detección por captura multiespectral de imágenes capturadas desde torres de observación terrestre, que consisten, a grandes rasgos, en puntos de observación y detección de columnas de humo típicas de incendios forestales de copas. Se consideran eficientes en dichas zonas, debido a que a través de su detección temprana, se pueden realizar acciones inmediatas para reducir la superficie afectada, lo que permite, además de su sofocación, conocer el origen del mismo. Ejemplo de ello es el desarrollo de la Red de Faros de Conservación

Sin embargo, debido a la extensión y diversidad de ecosistemas que posee nuestro país (Argentina Gobierno, 2021), es posible dividirlo en dieciocho ecorregiones diferentes donde se puede encontrar todo tipo de incendios forestales (Burkart et al., 1999). La zona Pampeana, por ejemplo, posee la mayor cantidad de superficie afectada por focos de incendios forestales de origen desconocido y por su vegetación oleosa, se requieren otros métodos de detección, ya que los incendios forestales son superficiales o subterráneos; Por lo que, el seguimiento del comportamiento y desplazamiento del incendio mediante métodos de imágenes puede ser errónea.

La provincia de Mendoza, posee un Plan Provincial de Manejo del Fuego y legislación declarando interés a prevención y lucha contra incendios en zonas rurales (Ley provincial 6099 - Declaracion interes provincial prevención lucha incendios zonas rurales bajo riego regadio secano

ministerio medio ambiente urbanismo vivienda creación programa
autoridad aplicaciòn -xvi- recursos naturales ecología medio ambiente).

En este contexto, para la cuantificación de ciertas variables ambientales que permitan detectar la ocurrencia de incendios de forma temprana y realizar seguimiento de los mismos, el presente trabajo plantea el diseño, desarrollo y validación de una red inalámbrica de sensores (WSN). Las WSN, en general, son construidas con elementos y funciones de ahorro energético y en general utilizan protocolos especialmente diseñados para dicha función. En este trabajo, se utilizó elementos de bajo costo, con la premisa de que ante la ocurrencia de un evento, puedan desplegarse, ser alcanzados por el fuego y sufrir daño total. En este sentido, para el mantenimiento de la comunicación, se utilizaron protocolos de red inalámbrica estándar adoptados por la industria (IEEE 802.11) a fin de simplificar la instalación y adaptación de gateways a diferentes infraestructuras de acceso a la red en zonas remotas.

Entre el abanico de uso, sobre todo para IoT (Internet de las Cosas), las WSN constituyen una herramienta tecnológica valiosa para recolectar información del entorno. Estas redes están formadas por plataformas de hardware con recursos limitados de memoria, cómputo y energía, motivo por el cual las WSN trabajan con protocolos especialmente diseñados para funcionar sobre estas plataformas, como el caso del estándar IEEE 802.15.4, ZigBee (Gislason, 2008), entre otros. Estos protocolos WSN, son propios de cada tecnología, lo que implica que para presentar los datos recolectados por las WSN al usuario mediante redes TCP/IP convencionales, deban implementarse distintas técnicas de hardware y software, por ello, este trabajo integra una WSN basada en el estándar IEEE 802.11 (Soto Sanchez, 2011), crucial para brindar información con

elementos masivamente utilizados a los equipos encargados de la sofocación de incendios.

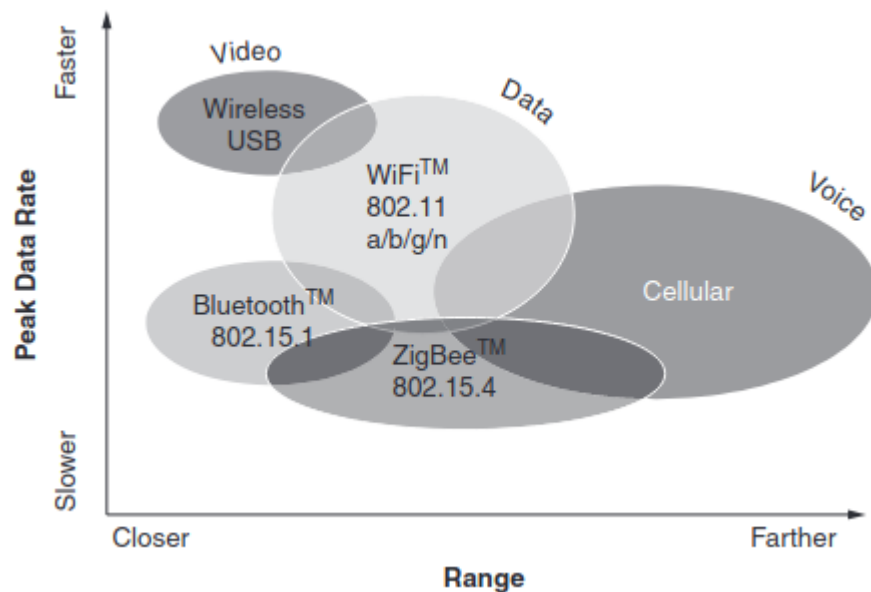


Figura 1: Comparativa de tecnologías inalámbricas

Además, la presente tesis, plantea que el uso y despliegue de una infraestructura basada en contenedores y cloud computing (detallada en el apartado “Destino de los Datos” del Desarrollo) que permitirá facilitar futuras implementaciones a desarrolladores, ya que podrían adaptar la misma y utilizarla para sus propios proyectos, ahorrando tiempo de desarrollo e investigación.

Por lo tanto, la incorporación de WSN junto con una infraestructura flexible para la lucha contra el fuego podría, mediante la captura de parámetros medidos en tiempo real, además de detectar incendios, colaborar en la prevención y predicción del desarrollo de incendios

forestales, alimentando además modelos de predicción de comportamiento de incendios.

Objetivos Planteados

En el presente, se planteó como objetivo principal la investigación para examinar el comportamiento y la eficacia de una Red Inalámbrica de Sensores (WSN, por sus siglas en inglés) que opera bajo el estándar IEEE 802.11 en el contexto de incendios forestales, con el propósito de detectar el inicio de estos eventos y proporcionar datos valiosos a sistemas de predicción del comportamiento de incendios forestales.

Como objetivos específicos:

1. Explorar la Diversidad de Controladores y Dispositivos WSN: Realizar un estudio exhaustivo de los distintos tipos de controladores y dispositivos de Red Inalámbrica de Sensores disponibles en el mercado.
2. Análisis de Desarrollos y Estudios Previos de WSN con IEEE 802.11: Evaluar investigaciones y desarrollos existentes que hayan empleado el estándar IEEE 802.11 en redes de sensores inalámbricos.
3. Implementación de una WSN Utilizando el Estándar IEEE 802.11: Diseñar y desarrollar una Red Inalámbrica de Sensores basada en el estándar IEEE 802.11.
4. Configuración y Administración de Dispositivos y Sensores: Establecer protocolos para la configuración y gestión de los dispositivos y sensores que formarán parte de la WSN.
5. Evaluación del Rendimiento y Resiliencia de la Red: Analizar el desempeño de la WSN en términos de eficacia de la comunicación, y evaluar su capacidad de recuperación y adaptación ante posibles fallos.
6. Examinar la Calidad de los Datos Capturados y su Integración en la Red: Evaluar la precisión y relevancia de la información obtenida por los dispositivos de captura, así como su integración eficiente en la red.

7. Documentación de Características y Análisis Crítico del Uso del IEEE 802.11 en WSN: Elaborar una documentación detallada que incluya características, arquitectura, ventajas y desventajas de utilizar el estándar IEEE 802.11 en redes de sensores inalámbricos sobre redes de área extensa (WAN).

8. Análisis de la Integración de Datos con Sistemas de Predicción: Investigar la manera en la que las variables resultantes pueden almacenarse e integrarse a diferentes sistemas de predicción

9. Evaluar el alcance de la solución y su viabilidad: A partir de los resultados obtenidos, generar un informe final.

Capítulo II: Marco Teórico y Antecedentes

Síntesis del capítulo

En esta sección, se presentan en detalle los conceptos fundamentales sobre los que se basa esta Tesis. Comenzando con una exploración sobre el elemento “Fuego”, se ofrece un panorama de su naturaleza, las implicaciones y el impacto devastador de los incendios forestales, enfocándose particularmente en Argentina. Además, se examinan las tecnologías actuales empleadas para detectar estos eventos.

A continuación, el enfoque se desplaza hacia las Redes Inalámbricas de Sensores, desglosando su mecánica operativa, las características distintivas, configuraciones, y el abanico de tecnologías a disposición.

Posteriormente se describe el chip ESP8266, el cual juega un papel fundamental en el desarrollo del proyecto. Presentando información pertinente acerca del chip, incluyendo su programación y configuración, que son vitales para el establecimiento de redes WSN.

Luego se aborda la, “Captura de Datos”, que brinda una descripción de los dispositivos y herramientas empleados en la captura de datos durante la fase experimental de este proyecto. En esta sección, se examinan los elementos que fueron utilizados en un entorno de laboratorio. También se describe las técnicas de “Virtualización” utilizadas, se introduce una descripción conceptual de los métodos de virtualización, además de una comparación perspicaz con las soluciones basadas en la nube (Cloud), se aborda el concepto de contenedor y se presenta una introducción a Docker, uno de los sistemas de contenerización más populares.

En la sección “Infraestructura”, habiendo establecido los conceptos precedentes, se profundiza en los contenedores de servicios específicos

que se incorporan en el proyecto. Asimismo, se describe el funcionamiento y despliegue de las aplicaciones que se utilizan.

Por último, se presenta un apartado dedicado a “Modelos de Sistemas de Predicción de Comportamiento de Incendios Forestales”, donde se hace referencia a la naturaleza de estos modelos, su funcionamiento, y cómo la incorporación de una Red Inalámbrica de Sensores puede potencialmente mejorar la precisión y fiabilidad de sus resultados.

Incendios Forestales

De acuerdo con la ISO (International Standards Organization, 1987; Cox, 1995) un incendio se define como:

1. Un proceso de combustión, caracterizado por la emisión de calor acompañada de humo, llamas o ambos.
2. Combustión que se propaga de manera incontrolada en el tiempo y el espacio

El fuego y sus efectos sobre ecosistemas

El fuego es un elemento esencial y natural en el funcionamiento de numerosos ecosistemas forestales. Los seres humanos vienen utilizando el fuego desde hace miles de años como instrumento de ordenación de la tierra. El fuego es uno de los elementos naturales que ha influido en las comunidades vegetales a lo largo del tiempo y como proceso natural cumple una función importante para mantener la salud de determinados ecosistemas (Dennis et al., n.d.).

Sin embargo, en la última parte del siglo XX, la modificación de la dinámica establecida entre la actuación humana y los incendios y la mayor frecuencia del fenómeno “El Niño” han dado lugar a una situación

en la que los incendios son una amenaza importante para los diferentes ecosistemas (*Efectos Del Fuego En Los Ecosistemas Forestales*, 2019).

Los incendios forestales tienen muchas repercusiones sobre la diversidad biológica. A escala mundial, son una fuente importante de emisión de carbono, contribuyendo al calentamiento mundial que podría modificar la biodiversidad. En los planos regional y local, modifican el volumen de biomasa, alteran el ciclo hidrológico con consecuencias sobre sistemas marinos como los arrecifes de coral, e influyen en el comportamiento de las especies biológicas. El humo procedente de los incendios puede reducir notablemente la actividad fotosintética y perjudicar la salud de los seres humanos y de los animales.

Uno de los efectos ecológicos más importantes de los incendios es la mayor probabilidad de que se produzcan nuevos episodios del mismo tipo en los años subsiguientes, al caer los árboles, lo que permite que la luz del Sol reseque el bosque y produzca una acumulación de combustible con un aumento de especies susceptibles a los incendios, como las herbáceas inflamables. La consecuencia de los incendios repetidos es perjudicial porque es uno de los factores principales del empobrecimiento de la biodiversidad en los ecosistemas de los bosques pluviales. Los incendios pueden ser seguidos de la colonización e infestación de insectos que perturban el equilibrio ecológico.

Los incendios son el enemigo más importante de las plantaciones forestales en más del 70% de los países. El fuego no solo destruye la madera, sino toda la fauna y flora que alberga y las construcciones aledañas.

Luego de un incendio, la sucesión vegetal es siempre de menor valor que el bosque original y el nivel freático se afecta sobre todo cuando el siniestro ocurre en cuencas hidrográficas y líneas divisorias.

El fuego que se extiende sin control en terreno forestal sostenido por especies vegetales o desechos vegetales que en general funcionan como combustibles, se denomina “Incendio Forestal” y se distingue de otros tipos de incendio por su amplia extensión, velocidad con la que se extiende desde su lugar de origen, su potencial para cambiar de dirección inesperadamente y su capacidad para superar obstáculos como carreteras, ríos y cortafuegos.

Causales de incendios

Si bien las causas inmediatas que dan lugar a los incendios forestales pueden ser muy variadas, en todos ellos se dan los mismos presupuestos, esto es, la existencia de grandes masas de vegetación en concurrencia con periodos más o menos prolongados de sequía. El calor solar provoca deshidratación en las plantas, que recuperan el agua perdida del sustrato, no obstante, cuando la humedad del terreno desciende a un nivel inferior al 30 % las plantas son incapaces de obtener agua del suelo, con lo que se van secando poco a poco. Este proceso provoca la emisión a la atmósfera de etileno, un compuesto químico presente en la vegetación y altamente combustible.

Tiene lugar entonces un doble fenómeno: tanto las plantas como el aire que las rodea se vuelven fácilmente inflamables, con lo que el riesgo de incendio se multiplica. Y si a estas condiciones se suma la existencia de períodos de altas temperaturas y vientos fuertes o moderados, la posibilidad de que una simple chispa provoque un incendio se vuelve significativa.

Por otro lado, al margen de que las condiciones físicas sean más o menos favorecedoras de un incendio, hay que destacar que en la gran mayoría de los casos no son causas naturales las que provocan el fuego, sino la acción humana, ya sea de manera intencionada o no.

Las causas que originan un incendio forestal pueden ser agrupadas en tres categorías principales:

1. **Intencionales:** Las motivaciones son variadas, siendo con diferencia las más comunes la quema no autorizada, ilegal e incontrolada de superficies agrícolas, ya sea para la eliminación de rastrojos o matorrales (“quema agrícola”) o para la regeneración de pastos para el ganado. Otras motivaciones menos corrientes detrás de un incendio provocado son la piromanía, usos cinegéticos, vandalismo, venganzas personales, especulación urbanística, bajar el precio de la madera, etc. Cabe señalar que el delito de incendio está tipificado en muchas legislaciones.
2. **Negligencias y otras causas accidentales:** Las quemas agrícolas autorizadas en las que los autores perdieron el control del fuego extendiéndose éste por la superficie forestal colindante están entre las causas habituales. Otras causas son las colillas y hogueras mal apagadas, quema de basuras, trabajos forestales, etc.
3. **Naturales:** Se deben, generalmente, a la acción de un rayo.

Por último, en contadas ocasiones, un incendio es una reproducción de un incendio anterior que no llegó a extinguirse del todo y se extiende a una nueva zona. Cabe aclarar que la frecuencia de la intencionalidad, por ejemplo, puede variar mucho de unas regiones a otras.

Fases del incendio

Cuando un incendio comienza, producido por causas naturales o mayoritariamente por la acción del hombre, está en fase de iniciación, cuando el mismo se extiende por la vegetación cercana, se encuentra en fase de propagación y finalmente entra en fase de extinción cuando se da la finalización del incendio por causas naturales (lluvia o falta de vegetación) o por acción humana (labores de extinción)

La propagación del fuego dependerá de las condiciones atmosféricas, de la topografía del lugar en el que se produzca y de la vegetación presente en el mismo. Normalmente se ocasionan en climas secos o sub-secos, como el mediterráneo, donde la vegetación sufre estrés hídrico y además algunas especies vegetales como los pinos contienen resinas que ayudan a que el incendio se propague mejor y sea más virulento. Asimismo, generalmente también poseen mecanismos de adaptación al fuego como por ejemplo las piñas serótinas.

Tipos de incendios.

El estudio de los incendios forestales distingue entre distintos tipos de fuegos, lo cual resulta útil a la hora de considerar las medidas más apropiadas de prevención y/o de extinción dado que pueden ser diferentes según se focalice el lugar de propagación; tamaño y elemento que rige el incendio.

1. Lugar de propagación (Ver Figura 2)

Uno de los factores determinantes para la aparición de grandes incendios forestales o de incendios de alta intensidad que superan la capacidad de extinción es: la presencia de elevadas cargas de combustible disponible para quemar y distribuidas a lo largo del territorio (Serra Davos & Plana Bach, 2016). En este sentido, se puede clasificar:

1.1. Fuego de suelo o subsuelo: El fuego se propaga por la materia orgánica en descomposición y las raíces. Casi siempre se queman despacio y en combustión incandescente (poca o ausencia de llama) al no disponer de suficiente oxígeno.

1.2. Fuego de superficie: El incendio se propaga por el combustible que encontramos sobre el suelo, incluye la hojarasca, hierbas, arbustos y madera caída pero no inmersa en la hojarasca en descomposición.

1.3. Fuego de copas: avanza por las copas de los árboles o de los arbustos.

1.3.1. Antorcheo: Paso de fuego de superficie a fuego de copas, pero solo de forma puntual en algunos pies.

1.3.2. Copas pasivo: Es el fuego que avanza por las copas de los árboles acoplado y dependiente de un fuego de superficie, si se extingue este se detiene el de copas.

1.3.3. Copas activo: Es el fuego que avanza por las coronas de los árboles independientemente de la superficie. Solo se puede atacar de forma indirecta y suele necesitar un viento mayor de 30 km/h y proximidad de copas (alta densidad aparente de copas y largas copas).

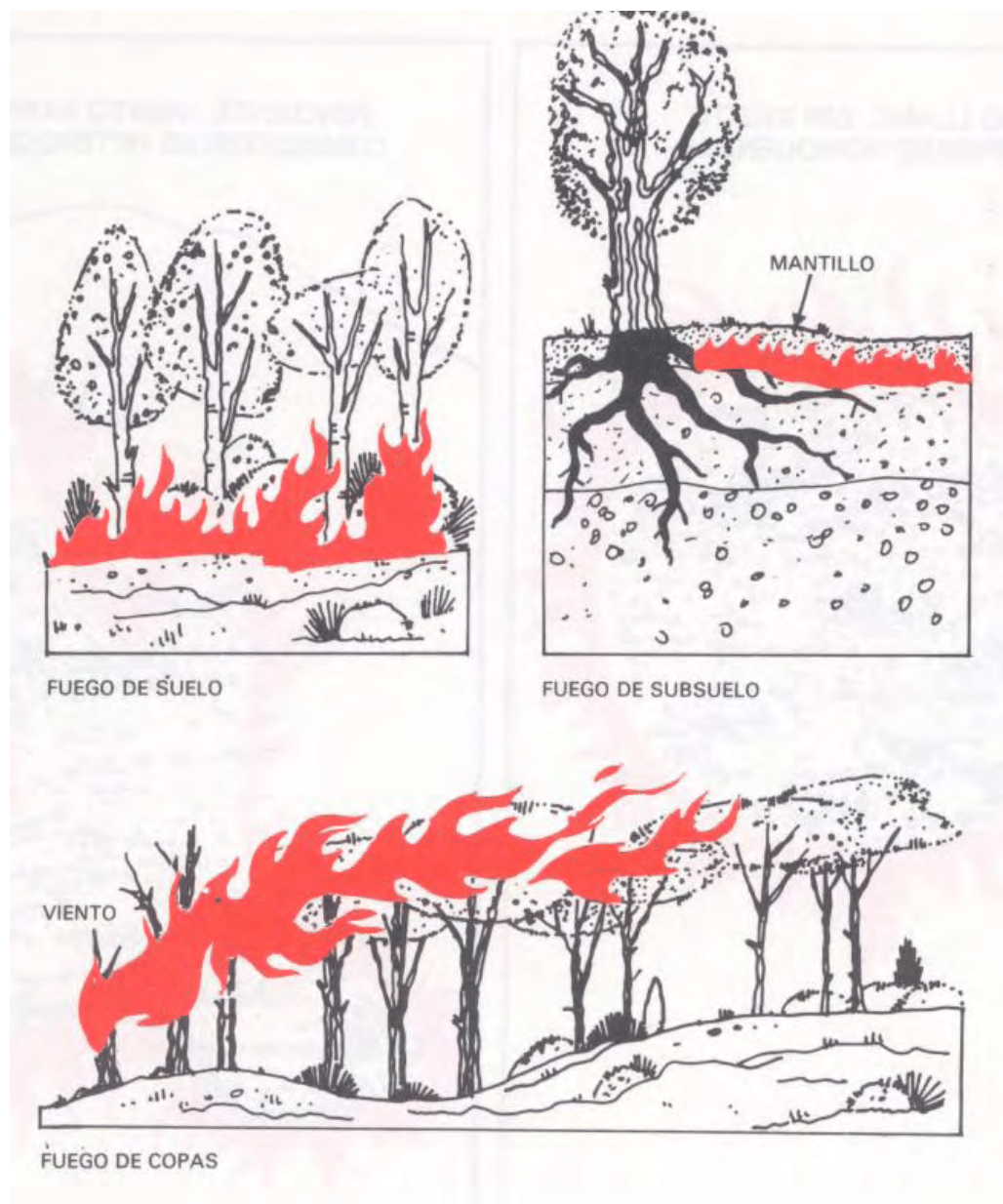


Figura 2: Tipos de incendios de acuerdo al estrato vegetacional afectado (Tomada de ICONA, 1993).

2. El tamaño

2.1. Los Grandes Incendios Forestales (de aquí en adelante llamados "GIF"). Se definen como aquellos que de manera continuada muestran un comportamiento que queda fuera de la capacidad del sistema de extinción, ya sea por las elevadas longitudes de llama, por las

altas velocidades de propagación o por la presencia de actividad de fuego de copas. Por actividad de copas entendemos que hay antorcheo importante (que prenden muchas de las copas del arbolado) o incluso que es fuego pasivo de copas o fuego activo.

2.2. Incendios de pequeño tamaño. Son aquellos que no alcanzan las características necesarias para ser un GIF.

3. El elemento que rige el incendio: Combustible, gas, topográfico, conducido por viento y hambriento.

La lucha del fuego en argentina

Existe un Plan Nacional de Manejo del Fuego (Argentina, n.d.), con guías específicas respecto a cómo actuar ante incendios y métodos de prevención que fueron creados en conjunto con el servicio forestal de British Columbia – Canadá.

Dicha guía fue elaborada en provincias que poseen bosques oleosos, en conjunto con especialistas de esas áreas, lo que explica que, en la zona Patagónica, existan múltiples focos de incendio y acciones para que la superficie afectada, si bien es extensa, no crezca demasiado. Particularmente, para estas zonas, se utilizan tecnologías de detección por imágenes, siendo estas a su vez las más utilizadas en el mundo.

Debido a que Argentina posee una gran extensión y diversos ecosistemas, es muy difícil contar con estadísticas generales ya que, a partir del año 2000, los datos se presentan agrupados según la regionalización del plan nacional de manejo del fuego.

Asimismo, según el informe presentado por el Ministerio de Ambiente y Desarrollo Sustentable de la Presidencia de la Nación durante el año

2014, se puede obtener un panorama general del “problema del fuego” sobre el territorio nacional.

En este aspecto, es interesante observar la relación existente entre cantidad de incendios, superficies afectadas por los mismos y su causa, expresados en la Ilustración que detallada en la Figura 3.

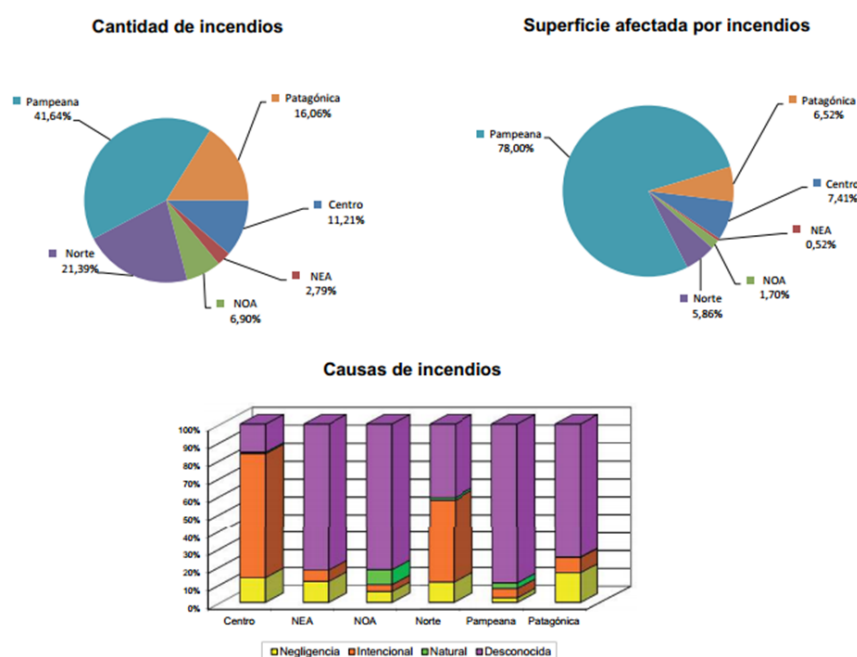


Figura 3: Cantidad de incendios, superficie afectada y causas en Argentina.
Fuente: Ministerio de Ambiente y Desarrollo Sustentable de la Presidencia de la Nación. Año: 2014.

Como puede apreciarse en los gráficos, la zona que posee mayor cantidad de superficie dañada por focos de incendio producidos por causas desconocidas es la pampeana.

A modo ilustrativo, se puede agregar que el clima de la región pampeana es templado con una temperatura media anual de 17° C. En cuanto a las lluvias, la pampa húmeda recibe un promedio de 1.000 mm. anuales de lluvias mientras que en la pampa seca llueve un promedio de 400 mm. anuales. Los vientos característicos de la Llanura Pampeana son “el Pampero”, frío, seco y violento proviene del sudoeste a velocidades de 100 km/h aproximadamente especialmente en verano y “la Sudestada”, que es un viento frío y húmedo.

Datos actualizados al 2021 de incendios forestales en el País.

Dentro del presente apartado, se detallan los datos de incendios del año 2021 según el último reporte del Ministerio de Ambiente y Desarrollo Sostenible del Gobierno Argentino.

En la figura que se muestra a continuación, puede observarse la superficie afectada por incendios por provincia:

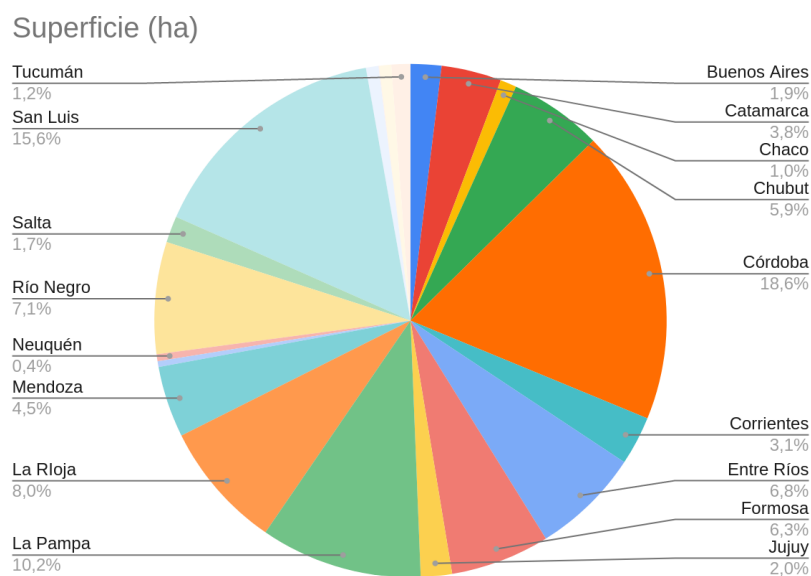


Figura 4: Superficie afectada por incendios forestales durante el 2021 en Argentina. Fuente: Ministerio de Ambiente y Desarrollo Sostenible del Gobierno Argentino. Año 2021.

Incendios Forestales y el uso de la tecnología

Actualmente el avance tecnológico y la creación de nuevos estándares hacen que problemas ambientales, como un incendio forestal, tengan nuevas y mejores alternativas de solución. Una de ellas son las Redes Inalámbricas de Sensores, compuestas por dispositivos de tamaño pequeño, que se integran a un sistema con capacidades de comunicación de radio. Los elementos principales que sirven de base para el funcionamiento de una WSN son: Nodos, que se encargan de la recolección de información mediante sensores de algún evento o fenómeno físico; Nodos Coordinadores que reciben toda la información enviada por cada nodo dentro de la WSN; Gateway, que interconecta la red WSN con una red TCP/IP y Estación Base, que recibe toda la información de la red para su procesamiento.

Red Inalámbrica de Sensores

Una Red Inalámbrica de Sensores (WSN), es una red de diminutos dispositivos, equipados con sensores, que colaboran en una tarea común. Las redes de sensores están formadas por un grupo de sensores con ciertas capacidades sensitivas y de comunicación inalámbrica que permiten formar redes ad-hoc sin infraestructura física preestablecida ni administración central.

La expresión ad-hoc hace referencia a una red en la que no hay un nodo central, sino que todos los dispositivos están en igualdad de condiciones. Ad-hoc es el modo más sencillo para crear una red, un tipo de red formada por un grupo de nodos móviles que forman una red temporal sin la ayuda de ninguna infraestructura externa. Para que esto se pueda llevar a la práctica es necesario que los nodos se puedan ayudar mutuamente para conseguir un objetivo común: que cualquier paquete

llegue a su destino aunque el destinatario no sea accesible directamente desde el origen. El protocolo de encaminamiento es el responsable de descubrir las rutas entre los nodos para hacer posible la comunicación.

Las redes de sensores poseen múltiples aplicaciones en distintos campos, tales como entornos industriales, domótica, entornos militares, detección ambiental. Esta clase de redes se caracterizan por su facilidad de despliegue y por ser autoconfigurables, pudiendo convertirse en todo momento en emisor, receptor, ofrecer servicios de encaminamiento entre nodos sin visión directa, así como registrar datos referentes a los sensores locales de cada nodo. Otra de sus características es su gestión eficiente de la energía, que les permite obtener una alta tasa de autonomía que las hacen plenamente operativas.

La miniaturización de los dispositivos, cada día mayor, dio a luz la idea de desarrollar dispositivos extremadamente pequeños y baratos que se comunican de forma inalámbrica y se organizan autónomamente. La idea de estas redes es repartir aleatoriamente estos nodos en un territorio amplio, del cual los nodos captan cierta información.

Arquitectura WSN

Las redes de sensores, como anteriormente se describió, están formadas por un conjunto de pequeños dispositivos denominados nodos sensores, con capacidad limitada de cómputo y comunicación. El tiempo de vida de la red de sensores dependerá del tiempo de vida de la batería de sus nodos. Estos dispositivos se encuentran dispersos de manera ad-hoc en una determinada área a monitorizar. En redes de comunicación, dicha expresión hace referencia a una red en la que no hay un nodo central, sino que todos los nodos están en igualdad de condiciones.

Típicamente, el modelo seguido por las aplicaciones es el siguiente: realizar una serie de mediciones sobre el medio, transformar dicha información en digital en el propio nodo y transmitirla fuera de la red de sensores vía un elemento gateway a una estación base, donde la información pueda ser almacenada y tratada temporalmente para acabar finalmente en un servidor con mayor capacidad que permite componer un histórico o realizar análisis de datos.

En una red de sensores inalámbricos, por lo tanto, podemos encontrarnos:

- nodos inalámbricos
- puertas de enlace
- estaciones base

Nodos Inalámbricos.

Los nodos inalámbricos se llaman motas, del inglés 'mote', por su ligereza y reducido tamaño. Son dispositivos electrónicos capaces de captar información proveniente del entorno en el que se encuentran, procesarla y transmitirla inalámbricamente hacia otro destinatario.

Puertas de Enlace.

Elementos para la interconexión entre la red de sensores y una red de datos (TCP/IP). Es un nodo especial sin elemento sensor, cuyo objetivo es actuar como puente entre dos redes de diferente tipo.

En este tipo de aplicaciones donde se usan redes de sensores, éstas no pueden operar completamente aisladas y deben contar con alguna forma de monitoreo y acceso a la información adquirida por los nodos de la red de sensores. De aquí surge la necesidad de conectar las redes de

sensores a infraestructuras de redes existentes tales como Internet, redes de área local (LAN) e intranets privadas. Los dispositivos que realizan la función de interconectar dos redes de diferente naturaleza se les llama dispositivo puerta de enlace; pero el término más conocido en el ambiente de las redes es gateway.

Estaciones Base.

Recolector de datos basado en un ordenador común o sistema empujado. En una estructura normal todos los datos van a parar a un equipo servidor dentro de una base de datos, desde donde los usuarios pueden acceder remotamente y poder observar y estudiar los datos.

Topologías de red.

Hay varias arquitecturas que pueden ser usadas para implementar una aplicación de WSN como pueden ser estrella, malla o una híbrida entre ellas dos. Cada topología presenta desafíos, ventajas y desventajas.

Para entender las diferentes topologías es necesario conocer los diferentes componentes de la WSN.

- Nodos finales: Compuesto por sensores y actuadores donde se capturan los datos sensores. Para las redes basadas en ZigBee son llamados RFD (Reduced Functional Devices).
- Routers: Dan cobertura a redes muy extensas pudiendo salvar obstáculos, problemas de congestión en la emisión de la información y posibles fallos en alguno de los aparatos.
- Puertas de enlace: Recoge los datos de la red, sirve como punto de unión con una red LAN o con Internet.

Topología se refiere a la configuración de los componentes hardware y como los datos son transmitidos a través de esa configuración. Cada topología es apropiada bajo ciertas circunstancias y puede ser inapropiada en otras. La idea de una red de sensores surge gracias a las posibilidades que nos da la tecnología

Topología Estrella.

Una topología en estrella es un sistema donde la información enviada sólo da un salto y donde todos los nodos sensores están en comunicación directa con la puerta de enlace. Todos los nodos sensores son idénticos, nodos finales, y la puerta de enlace capta la información de todos ellos. La puerta de enlace también es usada para transmitir datos al exterior y permitir la monitorización de la red.

Los nodos finales no intercambian información entre ellos, sino que usan la puerta de enlace para ello, si es necesario.

La topología en estrella es la que menor gasto de energía desarrolla, pero por el contrario está limitada por la distancia de transmisión vía radio entre cada nodo y la puerta de enlace. Tampoco tiene un camino de comunicación alternativo en caso de que uno de los nodos tenga obstruido el camino de comunicación, lo que lleva a que en este caso la información de ese nodo sea perdida.

Topología en Malla.

La topología en malla es un sistema multisalto, donde todos los nodos son routers y son idénticos. Cada nodo puede enviar y recibir información de otro nodo y de la puerta de enlace. A diferencia de la topología en estrella, donde los nodos solo pueden hablar con la puerta de enlace, en ésta los nodos pueden enviar mensajes entre ellos.

La propagación de los datos a través de los nodos hacia la puerta de enlace hace posible, por lo menos en teoría, crear una red con una extensión posible ilimitada. Este tipo, también es altamente tolerante a fallos ya que cada nodo tiene diferentes caminos para comunicarse con la puerta de enlace. Si un nodo falla, la red se reconfigura alrededor del nodo fallido automáticamente.

Dependiendo del número de nodos y de la distancia entre ellos, la red puede experimentar períodos de espera elevados a la hora de mandar la información.

Topología Híbrida Estrella-Malla.

Este tipo de red busca combinar las ventajas de los otros dos tipos, la simplicidad y el bajo consumo de una topología en estrella, así como la posibilidad de cubrir una gran extensión y de reorganizarse ante fallos de la topología en malla. Este tipo crea una red en estrella alrededor de routers pertenecientes a una red en malla. Los routers dan la posibilidad de ampliar la red y de corregir fallos en estos nodos y los nodos finales se conectan con los routers más cercanos ahorrando energía.

Redes AD-HOC.

Las redes más usadas en las implantaciones de redes de sensores inalámbricas son las redes malladas tipo ad-hoc. Son redes sin infraestructura, flexibles en las cuales todas las estaciones ofrecen servicios de encaminamiento para permitir la comunicación de estaciones que no tienen conexión inalámbrica directa.

La principal característica de las redes móviles ad-hoc es que todos los dispositivos que forman parte de la red, además de funcionar como

terminales finales, realizan también funciones de retransmisión de paquetes típicamente asociadas a routers.

Esta cualidad nos permite encaminar paquetes a destinos sin cobertura directa a través de otros nodos intermedios que se encuentren en la red. De este modo se nos ofrece la posibilidad de incrementar de una manera extraordinaria la movilidad y el tamaño de una red de datos inalámbrica. La funcionalidad principal de estas redes es la de crear de una manera rápida y eficaz una red temporal en lugares carentes de una infraestructura de red.

Las principales características de una red ad-hoc son:

- **Movilidad:** Este aspecto es la razón de ser de las redes ad-hoc. Los nodos se pueden reposicionar o simplemente ser móviles, siempre que no salgan del alcance radio. Se pueden desplegar rápidamente sin la necesidad de descubrir la zona o formar grupos, es decir, cada nodo es individual y solvente.
- **Multisalto (Multihopping):** Una red multihopping es una red donde el camino de la fuente al destino atraviesa varios nodos intermedios.
- **Autoorganización:** La red de forma autónoma debe determinar sus propios parámetros de configuración: dirección, encaminamiento, clustering, indicador de posición, etc.
- **Conservación de la energía:** Los nodos móviles, tienen una batería limitada y a no ser que dispongan de algún mecanismo de carga (por ejemplo, un panel solar), no tienen capacidad de recarga. Es muy importante diseñar unos protocolos (MAC, encaminamiento) eficientes, con la finalidad de mejorar el rendimiento y prolongar la autonomía de las baterías.

- Escalabilidad: En algunos tipos de redes, el número de nodos puede crecer hasta llegar a varios miles. Como no existe un access point concreto, la incorporación y descarte de nodos es un proceso sencillo y transparente.
- Seguridad: Las redes inalámbricas son vulnerables a ataques, y las redes ad-hoc lo son especialmente. Pueden padecer tanto ataques activos como pasivos y el atacante puede emular a un nodo legítimo y capturar paquetes de datos y control, destruir tablas de encaminamiento, etc.

Tecnologías inalámbricas.

La comunicación vía radio usa el espectro electromagnético para enviar información. Cuando una corriente eléctrica pasa por un cable crea un campo electromagnético que envía ondas en todas direcciones, en forma parecida a la luz, que también es parte del espectro, pero a frecuencias mucho más altas. Con una antena y energía adicional es posible transmitir las señales a gran distancia. La frecuencia de onda puede modificarse de forma que las señales no interfieran entre sí, lo que permite aprovechar una mayor parte del espectro.

Las tecnologías inalámbricas pueden clasificarse en cinco grandes grupos, de acuerdo con la distancia que viaja cada tipo de señal. Primero están las comunicaciones satelitales, como el sistema de posicionamiento global (GPS, por sus siglas en inglés), formado por 24 satélites manejados por las fuerzas armadas de Estados Unidos, las cuales envían constantemente señales a dispositivos en tierra. Sin embargo, estas señales sólo viajan del satélite al aparato receptor.

Otra categoría, y con señales de dos vías, son las tecnologías de telefonía celular de cobertura amplia como GSM y CDMA. Entre las

versiones avanzadas de 'tercera generación' (3G) destacan HSDPA y LTE 4G y 5G, desarrolladas por la industria de los celulares. Un contendiente es WiMax, tecnología basada en los estándares de Internet con respaldo de la industria informática.

Una tercera categoría incluye señales de menor alcance utilizadas para conectar dispositivos dentro de una habitación o un edificio, como los sistemas Wi-Fi para conectarse a Internet dentro de hoteles o aeropuertos, o Zigbee, protocolo de comunicaciones inalámbricas desarrollado específicamente para interconectar sensores.

Otra tecnología existente es de banda ultra ancha (UWB), que utiliza frecuencias sumamente altas de cobertura muy limitada para transmitir grandes volúmenes de información, por ejemplo, para enviar un video desde un iPod o un dispositivo similar a un televisor.

En cuarto lugar están los protocolos para enlazar dispositivos en una "red de área personal" (PAN, personal area network). Por ejemplo Bluetooth, utilizado para enviar la señal del teléfono celular a un auricular inalámbrico.

El último tipo de comunicaciones son las que se dan cerca de una antena transmisora (NFC, near-field communications). En este caso, el dispositivo receptor debe estar cerca del sistema emisor, por ejemplo, al pasar por un edificio o en el transporte público. Una variante son las etiquetas de identificación por radiofrecuencia (RFID, por sus siglas en inglés), utilizadas por tiendas departamentales y otros usuarios.

Cuando pasan por un lector, estas etiquetas envían la información que tienen almacenada. Estos sistemas de radio son tan diferentes entre sí

como la luz lo es del sonido; así, los satélites no pueden rastrear etiquetas RFID, lo cual permite descartar riesgos para la intimidad.

A continuación, podemos ver en forma gráfica cómo se distribuyen las diferentes tecnologías inalámbricas dependiendo de la velocidad de transmisión y de su utilización .

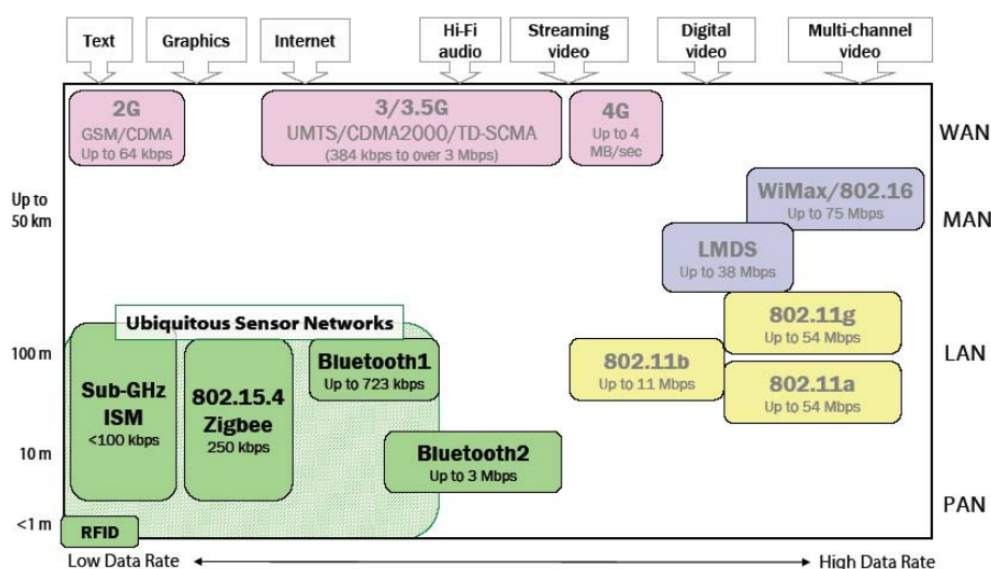


Figura 5: Tecnologías inalámbricas según su velocidad de transmisión y tamaño

Redes Mesh.

La red de malla inalámbrica (red de malla inalámbrica 802.11- Mesh) también se denomina red "multisalto". Es una tecnología de red inalámbrica que es completamente diferente de las redes inalámbricas tradicionales. La red de malla inalámbrica es una nueva estructura de red basada en el enrutamiento de varios saltos y la tecnología de red de igual a igual. Tiene las características de la banda ancha móvil y, al mismo tiempo, puede expandirse, organizarse y administrarse de forma dinámica. autorreparación y autoequilibrio.

En comparación con WIFI, el método de red de malla inalámbrica, la distancia de transmisión y la movilidad se han mejorado considerablemente, especialmente tiene características de Wi-Fi compatibles, por lo que la red de malla inalámbrica aumentará la distancia de transmisión y la movilidad de WIFI. La aplicación de Wi-Fi extendida proporciona gran ayuda. Al mismo tiempo, la popular aplicación actual de terminales traerá beneficios para la rápida promoción de la malla inalámbrica. Por lo tanto, las redes WIFI e inalámbricas Mesh pueden complementarse y fusionarse entre sí.

En la red de área local inalámbrica tradicional (WLAN), cada cliente accede a la red a través de un enlace inalámbrico conectado al AP. Si los usuarios desean comunicarse entre sí, primero deben visitar un punto de acceso fijo (AP). Esta estructura de red es llamada red de un solo salto. En una red Mesh inalámbrica, cualquier nodo de dispositivo inalámbrico puede actuar como AP y enrutador al mismo tiempo, cada nodo de la red puede enviar y recibir señales y cada nodo puede comunicarse directamente con uno o más nodos pares.

La mayor ventaja de esta estructura es que si el AP más cercano está congestionado debido a un tráfico excesivo, los datos se pueden redirigir automáticamente a un nodo vecino con poco tráfico para la transmisión, etc. El paquete de datos también puede basarse en la situación de la red, Continúe enrutando al siguiente nodo más cercano a él para la transmisión hasta que llegue al destino final. Este método de acceso es el acceso de varios saltos.

Wireless Mesh es una solución de red de área inalámbrica que es muy adecuada para cubrir un área grande de área abierta (incluidos exteriores e interiores).

Las características de la red de malla inalámbrica son:

Consiste en un grupo de AP inalámbricos distribuidos en una malla. Los AP están interconectados a través de un enlace de retransmisión inalámbrico de manera punto a punto, expandiendo los "puntos calientes" inalámbricos en la WLAN tradicional en "puntos calientes" inalámbricos que cubren un área grande.

Además, debido a que la distancia de transmisión de cada salto corto es corta, la potencia requerida para transmitir datos también es pequeña. Dado que las redes de varios saltos suelen utilizar menos potencia para transmitir datos a los nodos vecinos, la interferencia de la señal inalámbrica entre los nodos también es pequeña, y la calidad del canal y la eficiencia de utilización del canal de la red se mejoran enormemente y, por lo tanto, pueden lograr una mayor capacidad de red.

Por ejemplo, en un entorno de red urbana de alta densidad, la red Mesh puede reducir la interferencia mutua de los usuarios adyacentes que utilizan la red inalámbrica y mejorar en gran medida la eficiencia de la utilización del canal.

IEEE 802.15.4.

Los dispositivos se relacionan entre sí a través de una red inalámbrica sencilla. La definición de los niveles se basa en el modelo OSI. Aunque los niveles inferiores se definen en el estándar, se prevé la interacción con el resto de niveles, posiblemente por medio de un subnivel de control de

enlace lógico basado en IEEE 802.2, que acceda a MAC a través de un subnivel de convergencia. La implementación puede basarse en dispositivos externos o integrarlo todo en dispositivos autónomos.

El nivel físico (PHY) provee el servicio de transmisión de datos sobre el medio físico propiamente dicho, así como la interfaz con la entidad de gestión del nivel físico, por medio de la cual se puede acceder a todos los servicios de gestión del nivel y que mantiene una base de datos con información de redes de área personal relacionadas. De esta forma, PHY controla el transceptor de radiofrecuencia y realiza la selección de canales junto con el control de consumo y de la señal. Opera en una de tres posibles bandas de frecuencia de uso no regulado.

- 868-868,8 MHz: Europa, permite un canal de comunicación (versión de 2003), extendido a tres en la revisión de 2006.
- 902-928 MHz: Norteamérica, hasta diez canales (2003) extendidos a treinta (2006).
- 2400-2483,5 MHz: uso en todo el mundo, hasta dieciséis canales (2003, 2006).

La versión original del estándar, de 2003, especifica dos niveles físicos basados en espectro ensanchado por secuencia directa (direct sequence spread spectrum, DSSS): uno en las bandas de 868/915 MHz con tasas de 20 y 40 kbps; y otra en la banda de 2450 MHz con hasta 250 kbps.

La revisión de 2006 incrementa las tasas de datos máximas de las bandas de 868/915 MHz, que permiten hasta 100 y 250 kbps. Aún más, define cuatro niveles físicos en base al método de modulación usado. Tres de ellas preservan el mecanismo por DSSS: las bandas de 868/915 MHz, que usan modulación en fase binaria o por cuadratura en offset

(offset quadrature phase shift keying, esta segunda opcional). En la banda de 2450 MHz se usa esta segunda técnica. Adicionalmente, se define una combinación opcional de modulación binaria y en amplitud para las bandas de menor frecuencia, basadas por lo tanto en una difusión de espectro paralela, no secuencial (PSSS). Si se usan estas bandas de menor frecuencia, se puede cambiar dinámicamente el nivel físico usado de entre los soportados.

El control de acceso al medio (MAC) transmite tramas MAC usando para ello el canal físico. Además del servicio de datos, ofrece una interfaz de control y regula el acceso al canal físico y al balizado de la red. También controla la validación de las tramas y las asociaciones entre nodos, y garantiza slots de tiempo. Por último, ofrece puntos de enganche para servicios seguros.

El estándar no define niveles superiores ni subcapas de interoperabilidad. Existen extensiones, como la especificación de ZigBee, que complementan al estándar en la propuesta de soluciones completas.

ZigBee.

ZigBee es un estándar que define un conjunto de protocolos para el armado de redes inalámbricas de corta distancia y baja velocidad de datos. Opera en las bandas de 868 MHz, 915 MHz y 2.4 GHz y puede transferir datos hasta 250Kbps.

Este estándar fue desarrollado por la Alianza ZigBee, que tiene a cientos de compañías desde fabricantes de semiconductores y desarrolladores de software a constructores de equipos OEMs e instaladores. Esta organización sin fines de lucro nace en el año 2002. Desarrolla un protocolo que adopta al estándar IEEE 802.15.4 para sus 2 primeras

capas, es decir la capa física (PHY) y la subcapa de acceso al medio (MAC) y agrega la capa de red y de aplicación.

La idea de usar una conexión inalámbrica para controlar sensores y adquirir datos tiene muchos años. Existen numerosas soluciones propietarias usadas en domótica pero el gran inconveniente que tienen es la incompatibilidad entre sensores, controles y equipos de procesamiento de datos que obliga a hacer pasarelas (gateways) para interconectar dispositivos de diferentes marcas. El estándar ZigBee fue diseñado con las siguientes especificaciones:

- Ultra bajo consumo que permite usar equipos a batería
- Bajo costo de dispositivos y de instalación y mantenimiento de ellos.
- Alcance corto (típico menor a 50 metros).
- Optimizado para ciclo efectivo de transmisión menor a 0.1 %
- Velocidad de transmisión menor que 250 kbps. Típica: menor que 20 kbps

Existen muchos estándares que se pueden usar en redes de corto alcance tales como el 802.11 y Bluetooth. Cada uno de estos está desarrollado para una clase de aplicación determinada. ZigBee es el estándar más aceptado hoy para usar en redes de sensores y actuadores que deban operar a batería.

IEEE 802.11.

WiFi: Es una tecnología de interconexión inalámbrica, que hace referencia a “Wireless Fidelity” (fidelidad inalámbrica). Wi-Fi Alliance es una asociación que se encarga de certificar que los dispositivos de WiFi

cumplen con el conjunto de IEEE de estándares inalámbricos 802.11. Este estándar define los niveles inferiores de la capa del Modelo OSI, la capa física y la capa de enlace de datos, para conexiones inalámbricas que utilizan ondas electromagnéticas. Dentro de este estándar existen distintas variaciones en las cuales se definen características específicas para cada caso, modificando principalmente la frecuencia en la cual se trabaja y la velocidad de

transferencia de datos. 802.11 b/g/n es el estándar de mayor utilización en los dispositivos, trabaja en la frecuencia de 2.4GHz y alcanza velocidades de hasta 300Mbps.

Su topología de red consiste en la creación de puntos de acceso en donde se conectan los diferentes dispositivos que quieren pertenecer a una red. Respecto a la seguridad, la conexión a los puntos de acceso puede ser o no protegida con algún tipo de contraseña.

ESP8266

Este microcontrolador tiene una serie de características clave que lo hacen asequible. Y es que partimos de una CPU RISC de 32 bits que trabaja a 80 MHz. Es capaz de soportar overclock, tanto en su CPU como en su memoria flash, lo que consigue impulsar su rendimiento hasta los 160 MHz en su procesador y entre los 40 y 80 MHz en su memoria flash.

Tiene una Caché de instrucciones de 64 KB y una de datos de 96 KB, capacidad externa de memoria QSPI que puede llegar hasta los 16 MB, soporte para IEEE 802.11 b/g/n , TR switch, balun, LNA, soporte para WEP y WPA/WPA2.

Los métodos de entrada se reducen a una conexión de 16 pines GPIO, que corren y que le otorga mayor flexibilidad. Aparte, tiene soporte para

UART, SPI e I2C, donde trabaja a un voltaje entre 3V y 3,6V con una intensidad de 80mA.

El chip ESP8266 comercialmente se lo encuentra en diferentes módulos cuya diferencia son la cantidad de pines disponibles y en algunos casos el tamaño de memoria flash. Las capacidades de este módulo, generalmente son utilizadas junto a una placa Arduino, ya que permite conectar la placa a cualquier red local, internet u aparatos compatibles con Wi-Fi.

Arduino IDE.

El entorno de desarrollo integrado (IDE) de Arduino es una aplicación multiplataforma que está escrita en el lenguaje de programación Java. Se utiliza para escribir y cargar programas en placas compatibles con Arduino, pero también, con la ayuda de núcleos de terceros, se puede usar con placas de desarrollo de otros proveedores.

El código fuente para el IDE se publica bajo la Licencia Pública General de GNU, versión 2.2 El IDE de Arduino admite los lenguajes C y C ++ utilizando reglas especiales de estructuración de códigos. El IDE de Arduino suministra una biblioteca de software del proyecto Wiring, que proporciona muchos procedimientos comunes de E/S. El código escrito por el usuario solo requiere dos funciones básicas, para iniciar el boceto y el ciclo principal del programa, que se compilan y vinculan con un apéndice de programa `main()` en un ciclo con el GNU toolchain, que también se incluye. El IDE de Arduino emplea el programa `avrdude` para convertir el código ejecutable en un archivo de texto en codificación hexadecimal que se carga en la placa Arduino mediante un programa de carga en el firmware de la placa.

Proyecto: Núcleo Arduino para chip ESP8266.

Este proyecto brinda soporte para el chip ESP8266 al entorno Arduino. Le permite escribir bocetos, utilizando funciones y bibliotecas familiares de Arduino, y ejecutarlos directamente en ESP8266, sin necesidad de un microcontrolador externo.

El núcleo Arduino ESP8266 viene con bibliotecas para comunicarse a través de WiFi usando TCP y UDP, configurar servidores HTTP, mDNS, SSDP y DNS, hacer actualizaciones OTA, usar un sistema de archivos en la memoria flash y trabajar con tarjetas SD, servos, SPI e I2C periféricos.

En pocas palabras podemos decir que un ESP8266 es un microcontrolador con WIFI incorporado. Y al igual que muchos microcontroladores, podemos reprogramarlo a nuestra necesidad, por defecto o de fábrica el ESP8266 viene con un firmware de comandos AT, que permite manipular el ESP8266 desde otro microcontrolador, pero, mediante la utilización de librerías especiales como la generada en Github por Iván Grokhotkov y abierta a la comunidad (<https://github.com/esp8266/Arduino>), es posible programar el ESP8266 usando el IDE Arduino añadiendo la librería: https://arduino.esp8266.com/stable/package_esp8266com_index.json.

Placa de Desarrollo D1 Wemos Mini

La placa Wemos D1 Mini es una placa de desarrollo con dimensiones de 34.2mm x 25.6mm y un peso de 3g, es una de las placas más pequeñas basadas en el ESP8266.

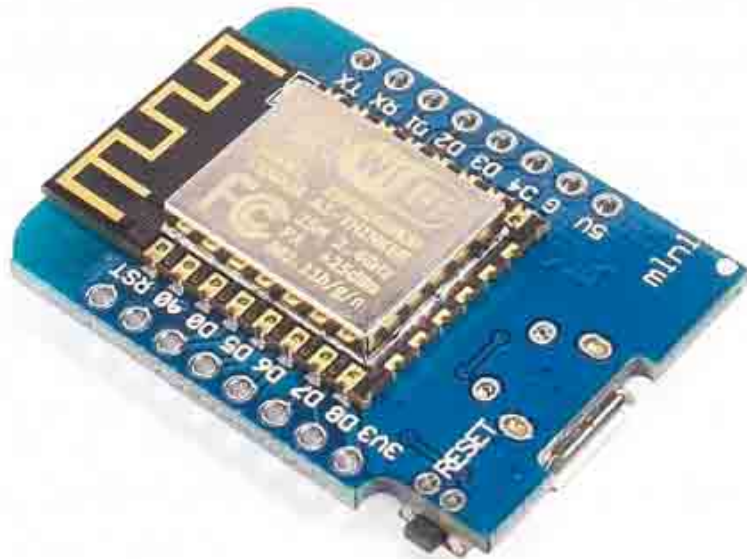


Figura 6: Placa D1 WeMos Mini

La Wemos D1 Mini es sólo ligeramente más grande que usar un módulo ESP12 directamente y aporta muchas ventajas, como incorporar un puerto Micro SD y conversor serial, regulador de tensión que permite alimentarlo a 5V, y terminales para conectar dispositivos. Además la Wemos D1 Mini permite ampliar su funcionalidad mediante la conexión de Shields. Existe una gran variedad de Shields como, por ejemplo, controlador de motores, módulo con rele, pantalla Oled, sensores de temperatura o humedad entre otros.

Como desventaja la Wemos D1 Mini tiene un número menor de pines disponible. Únicamente expone 11 pines GPIO frente a los 17 disponibles en el ESP12 o la NodeMCU.

La Wemos Mini D1 está basada en el ESP12E, por lo que las características vienen directamente del mismo. De forma muy resumida estas son algunas de las principales características:

- Velocidad: 80MHz/160MHz
- Flash: 4M bytes
- Tensión funcionamiento: 3.3V
- Entradas y salidas digitales: 11, todos (salvo el D0) con PWM, interrupciones, e I2C
- Entradas analógicas: 1 (Max. 3.2V)
- Conector Micro-USB

Pinout WEMOS D1 Mini.

En la imagen que se presenta a continuación, se observa la utilidad que posee cada pin de la placa en estudio.

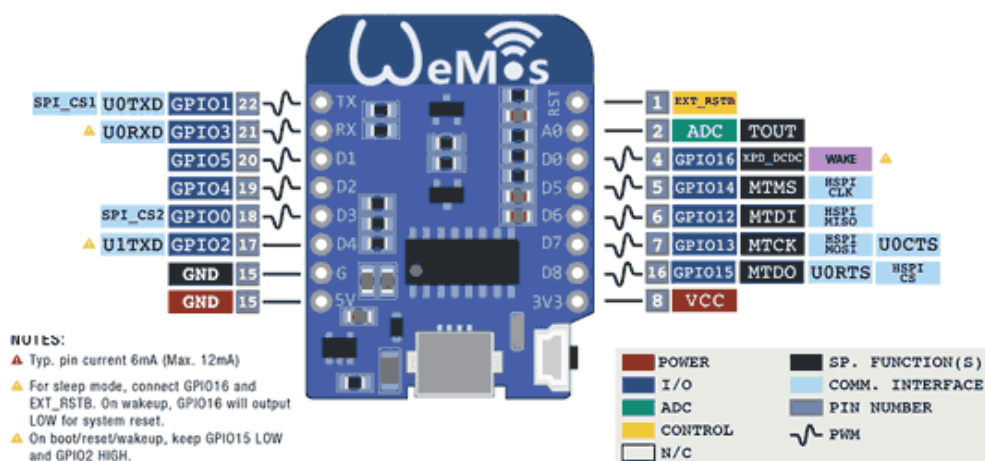


Figura 7: Pinout de placa D1 WeMos Mini

A continuación, se detalla la función específica de los pines de conexión.

Pin	ESP-8266 Pin	Función
TX	TXD	TXD
RX	RXD	RXD
A0	A0	Analog input (max 3.2V)
D0	GPIO16	IO

D1	GPIO5	IO, PWM, Interrupt, I2C, SCL
D2	GPIO4	IO, PWM, Interrupt, I2C, SDA
D3	GPIO0	IO 10k Pull-up, PWM, Interrupt, I2C
D4	GPIO2	IO 10k Pull-up, PWM, Interrupt, I2C, BUILTIN_LED
D5	GPIO14	IO, PWM, Interrupt, I2C, SCK
D6	GPIO12	IO, PWM, Interrupt, I2C,, MISO
D7	GPIO13	IO, PWM, Interrupt, I2C,, MOSI
D8	GPIO15	IO 10k Pull-down, PWM, Interrupt, I2C,, SS
G	GND	Ground
5V	-	5V
3V3	3.3V	3.3V
RST	RST	Reset

Programación en IDE Arduino.

Programar la Wemos Mini D1 desde el entorno de Arduino requiere la integración de las librerías vistas con anterioridad y al igual que el del resto de placas ESP8266 únicamente se debe elegir la placa "Wemos D1 R2 o Mini":



Figura 8: Librería D1 WeMos mini sobre IDE Arduino

Librería painlessMesh .

PainlessMesh es una librería para Arduino IDE que se ocupa de los detalles de la creación de una red de malla simple utilizando hardware esp8266 y esp32. El objetivo es permitir que el programador trabaje con

una red en malla sin tener que preocuparse por cómo está estructurada o administrada la red.

PainlessMesh es una red ad-hoc, lo que significa que no se requiere planificación, controlador central o enrutador. Cualquier sistema de 1 o más nodos se autoorganizará en una malla completamente funcional. El tamaño máximo de la malla está limitado por la cantidad de memoria en el montón que se puede asignar al búfer de subconexiones y, por lo tanto, debería ser bastante alto.

Usa objetos JSON para todos sus mensajes. Las razones para esto son: En primer lugar, hace que el código y los mensajes sean legibles por humanos y fáciles de entender y, en segundo lugar, hace que sea sencillo integrar indoloraMesh con front-end de JavaScript, aplicaciones web y otras aplicaciones. Se pierde algo de rendimiento, pero no críticos como para generar problemas de rendimiento.

Está diseñado para usarse con Arduino, pero no usa las bibliotecas WiFi de Arduino, ya que la latencia entre ellas sería muy alta. Más bien, la red se realiza utilizando las bibliotecas nativas del SDK esp32 y esp8266, que están disponibles a través del IDE de Arduino. Sin embargo, es de esperar que las bibliotecas de red que se utilicen no les importen mucho a la mayoría de los usuarios, ya que puede incluir `painlessMesh.h`, ejecutar `init()` y luego trabajar con la biblioteca a través de la API.

PainlessMesh no crea una red de nodos TCP / IP. Más bien, cada uno de los nodos se identifica de forma única por su `chipId` de 32 bits que se recupera del esp8266 / esp32 utilizando la `system_get_chip_id()` llamada en el SDK. Cada nodo tendrá un número único. Los mensajes pueden

transmitirse a todos los nodos de la malla o enviarse específicamente a un nodo individual que se identifica por su `nodeId`.

Limitaciones y advertencias de painlessMesh.

Debe evitarse el uso de `delay()` en el código. Para mantener la malla se necesita realizar algunas tareas en segundo plano. El uso `delay()` evitará que estas tareas sucedan y puede hacer que la malla pierda estabilidad o se deshaga. En su lugar, se recomienda la utilización de `TaskScheduler`, que se utiliza en `painlessMesh` en sí mismo.

`PainlessMesh` se suscribe a eventos WiFi. Como resultado, `painlessMesh` puede ser incompatible con programas de usuario / otras bibliotecas que intentan vincularse a los mismos eventos.

Se debe ser moderado en la cantidad de mensajes (y especialmente mensajes de difusión) que se envía por minuto. Esto es para evitar que el hardware se sobrecargue. Tanto `esp8266` como `esp32` están limitados en capacidad de procesamiento / memoria, lo que facilita sobrecargar la malla y desestabilizarla. Y aunque `painlessMesh` trata de evitar que esto suceda, no siempre es posible hacerlo.

Los mensajes pueden desaparecer o descartarse debido al alto tráfico y no se debe confiar en que se entregarán todos los mensajes. Para evitar esto, se pueden reenviar mensajes de vez en cuando. Incluso si algunos desaparecen, la mayoría debería pasar. Otra opción es que sus nodos envíen respuestas cuando reciben un mensaje. Los nodos emisores pueden reenviar el mensaje si no han recibido una respuesta en un período de tiempo determinado.

Captura de Datos

Captura de Datos mediante Sensores.

Debido a que el presente trabajo, tiene como objeto la medición de ciertos parámetros importantes para determinar la ocurrencia de incendios y realizar un seguimiento de los mismos, los elementos a incluir son los siguientes:

- Sensor de Temperatura y Humedad
- Sensor de Monóxido de Carbono

Sensor de Temperatura y Humedad: DHT11.

El DHT11 es un sensor con una alta fiabilidad y estabilidad debido a su señal digital calibrada. Se puede encontrar de dos maneras, de forma individual donde solo tenemos el sensor DHT11, o insertado en una PCB.

La diferencia es que la versión con PCB aporta una resistencia pull-up de 5 k Ω y un LED que nos avisa de su funcionamiento. Otra diferencia entre estas dos versiones del DHT11 son los pines.

En la versión sin PCB tenemos 4 pines y en la versión con PCB tenemos 3 pines.

En este trabajo, se optó por la opción con PCB. Los pines de la versión con PCB del DHT11 son:

- GND: conexión con tierra
- DATA: transmisión de datos
- VCC: alimentación

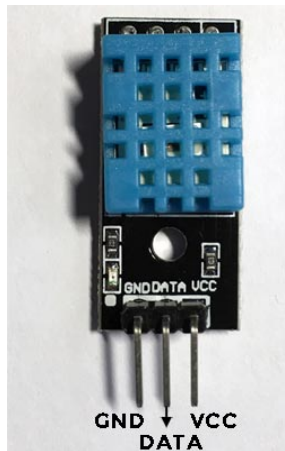


Figura 9: Sensor DHT 11

Dentro del propio dispositivo se hace una conversión entre analógico y digital.

Por lo tanto, partimos de una señal analógica que luego es convertida en formato digital y se enviará al microcontrolador. La trama de datos es de 40 bits correspondiente a la información de humedad y temperatura del DHT11.

<u>0011 0101</u>	<u>0000 0000</u>	<u>0001 1000</u>	<u>0000 0000</u>	<u>0100 1001</u>
8 bits humedad	8 bits humedad	8 bits temperatura	8 bits temperatura	bits de paridad

El primer grupo de 8-bit es la parte entera de la humedad y el segundo grupo la parte decimal. Lo mismo ocurre con el tercer y cuarto grupo, la parte entera de la temperatura y la parte decimal. Por último los bits de paridad para confirmar que no hay datos corruptos.

Estos bits de paridad lo único que hacen es asegurarnos de que la información es correcta, sumando los 4 primeros grupos de 8-bit. Esta

suma debe ser igual a los bit de paridad. Si nos centramos en la imagen anterior y sumamos los bits, comprobamos que todo está correcto.

$$0011\ 0101 + 0000\ 0000 + 0001\ 1000 + 0000\ 0000 = 0100\ 1101$$

Sensor de Monóxido de Carbono MQ-7.

El sensor MQ-7 es un sensor para la detección de monóxido de carbono (CO). Su funcionamiento se basa en medir la concentración de este gas en el aire. El MQ-7 puede medir concentraciones de gas monóxido entre 20 a 2000 partes por millón (ppm)

El sensor MQ7 posee una alta sensibilidad y rápido tiempo de respuesta. Se utiliza sobre un pin de entrada analógico para medir la concentración del gas.

Las conexiones que requiere el sensor son muy básicas solo requiere alimentación de 5V para el elemento calefactor



Figura 10: Sensor MQ-7

Los pines de la versión con PCB del sensor de gas MQ-7 son:

- GND: Tierra

- VCC: Alimentación 5V
- AOUT: Salida Analógica
- DOUT: Salida Digital

Destino de Datos Recolectados

Una vez recolectados los datos, los mismos deberán ser recolectados y distribuidos y almacenados en una base de datos a fin de generar clientes de consulta de los mismos. Para ello, se describe a continuación cada uno de los elementos a utilizar.

Virtualización

La virtualización es una tecnología que permite crear servicios de TI útiles mediante recursos que están ligados tradicionalmente al hardware. Además, distribuye sus funcionalidades entre diversos usuarios o entornos, lo que permite utilizar toda la capacidad de una máquina física.

El software denominado hipervisores separa los recursos físicos de los entornos virtuales, es decir, todo lo que necesitan los recursos. Los hipervisores pueden conformarse como elementos principales de un sistema operativo (como una computadora portátil) o se pueden instalar directamente en el hardware (como un servidor), que es la forma en que la mayoría de las empresas virtualiza. Los hipervisores toman los recursos físicos y los dividen de manera tal que los entornos virtuales puedan usarlos.

Los recursos se dividen según las necesidades, desde el entorno físico hasta los numerosos entornos virtuales. Los usuarios interactúan con los cálculos y los ejecutan dentro del entorno virtual (generalmente denominado máquina de guest o máquina virtual). La máquina virtual funciona como un archivo de datos único. Al igual que con cualquier

archivo digital, se puede mover de una computadora a otra, abrir en cualquiera de ellas y esperar que funcione de la misma manera.

Cuando el entorno virtual se está ejecutando y un usuario o programa emite una instrucción que requiere recursos adicionales del entorno físico, el hipervisor transmite la solicitud al sistema físico y guarda los cambios en la caché. Todo esto sucede prácticamente a la misma velocidad que habría si este proceso se realizara dentro de la máquina física (en especial, si la solicitud se envía a través de un hipervisor open source basado en KVM, la máquina virtual basada en el kernel).

Tipos de Virtualización.

Virtualización de servidores.

La virtualización de servidores permite ejecutar múltiples sistemas operativos en un solo servidor físico por medio de máquinas virtuales que ofrecen un elevado rendimiento. Entre las ventajas principales, se incluyen las siguientes:

- Mayor eficiencia del entorno de TI
- Reducción de los costes operativos
- Implementación más rápida de las cargas de trabajo
- Mejora del rendimiento de las aplicaciones
- Mayor disponibilidad del servidor
- Eliminación de la complejidad y la proliferación de servidores

La virtualización de sistemas operativos se suele clasificar en dos tipos, aunque dependiendo de la fuente el segundo tipo se puede desplegar en otros dos, así que vamos a clasificarlos de la siguiente forma:

Virtualización de tipo 1.

En este tipo de virtualización, el monitor de la máquina virtual (también llamado hipervisor) se ejecuta directamente sobre el hardware físico. Eso quiere decir que en este esquema instalaremos un sistema operativo dedicado única y exclusivamente a ejecutar máquinas virtuales, y que este sistema operativo permite la comunicación directa entre el hardware real del ordenador y el sistema operativo virtualizado. Esto hace que sea el tipo de virtualización más veloz de los tres tipos, ya que el sistema operativo principal (el que alberga las máquinas virtuales) no se dedica a más cosas que a gestionarlas correctamente y los sistemas operativos virtualizados pueden acceder a los recursos del ordenador sin ningún impedimento. Los sistemas más famosos para estas labores son VMWare ESXi y Citrix XenServer, aunque también hay otros menos conocidos como Proxmox (que usa KVM-QEMU como hipervisor) o XCP-ng (que es un fork libre de Citrix XenServer) pisándole los talones y ganando usuarios a pasos agigantados. El problema que podría tener un particular con este tipo de virtualización es que el ordenador no se puede usar para otra cosa más que para virtualizar.

Virtualización de tipo 2.

Aquí es necesario instalar el hipervisor en un sistema operativo que no está dedicado exclusivamente para estas tareas (como Windows), por lo que el sistema operativo que está en comunicación directa con el hardware es el principal que tenga instalado el equipo, no las máquinas virtuales, que no tienen comunicación directa con el hardware. Este formato se usa mucho hoy en día en programas que quieren separar sus procesos de los del resto del sistema operativo, como Java o .NET (Ojo: estos dos que pongo como ejemplo no son hipervisores de máquinas virtuales), pero la mayoría de hipervisores actuales que quieran instalarse

sobre un sistema previo adoptan lo que vamos a llamar «tipo 3» (que es una variante del tipo 2).

Virtualización de tipo 3 o híbrida.

Se tratan de programas que se instalan sobre un sistema operativo principal, de la misma forma que en la virtualización de tipo 2, pero que permiten la conexión directa entre el hardware y las máquinas virtuales que instalemos (algo llamado «Paravirtualización»). Aquí encontramos el software de virtualización para usuarios domésticos que más nos sonará, como Virtual Box, Parallels, VMWare Player o Microsoft Virtual PC. Este tipo de virtualización es el que más se usa a nivel doméstico para crear máquinas virtuales en nuestro propio ordenador sin tener que desinstalar nada y suele requerir algún tipo de instalación de «drivers» en las máquinas virtualizadas (como las «VMWare tools» en VMWare o las «Guest Additions» en Virtual Box).

Virtualización de red.

Al reproducir una red física en su totalidad, la virtualización de red permite ejecutar las aplicaciones en una red virtual del mismo modo que en una red física, pero con mayores ventajas operativas y con toda la independencia del hardware que ofrece la virtualización. La virtualización de red muestra los dispositivos y servicios de red lógicos (puertos lógicos, conmutadores, enrutadores, cortafuegos, equilibradores de carga, VPN, etc.) a las cargas de trabajo vinculadas.

Virtualización de escritorios.

Implementar los escritorios como un servicio gestionado permite a las organizaciones de TI responder más rápido a las necesidades cambiantes del entorno de trabajo y a las nuevas oportunidades. Los escritorios y las aplicaciones virtualizados también pueden distribuirse de forma rápida y

sencilla a sucursales, empleados subcontratados o en otros países y trabajadores móviles que utilizan tabletas iPad y Android.

Virtualización y Cloud.

A pesar de ser tecnologías que suscitan el mismo interés, la virtualización y la informática de cloud no son intercambiables. La virtualización utiliza el software para separar los entornos informáticos de la infraestructura física, mientras que la informática de cloud es un servicio que proporciona recursos informáticos compartidos (software o datos) según las necesidades por medio de Internet. Como son soluciones complementarias, las organizaciones pueden comenzar por virtualizar sus servidores y luego adoptar la informática de cloud para lograr una mayor agilidad y capacidad de autoservicio.

Proxmox.

PROXMOX VE ofrece beneficios similares a los productos para virtualización como VMware vSphere, Windows Hyper-V, Citrix XenServer, entre otros.

Siendo PROXMOX libre sin costo, se puede instalar en cualquier cantidad de "Servidores físicos", sin límite en uso de Procesadores y Sockets, Puentes de comunicación, o integración de NAS o SAN ya sea a través de Fibra Canal, iSCSI Over Ethernet o NFS.

"Proxmox VE" en su sitio oficial pone a su disposición una imagen ISO para generar el instalador en un CD o USB/Memory. La instalación se efectúa en un equipo vacío y en 15 minutos tiene preparado un servidor (Nodo) para colocar máquinas virtuales o integrarlo a un "Cluster".

Este tipo de instalación se conoce como "Bare-metal" lo cual significa que la instalación agrega todo lo necesario al equipo y lo configura para su disponibilidad en un ambiente productivo.

Esta solución trabaja con "Debian OS + KVM virtualization + Container-based Virtualization", toda la base es Libre y esto hace posible que el producto final sea libre.

Características de PROXMOX.

- Administrador Web HTML5. PROXMOX proporciona un interfaz Web para configurar los servidores físicos, cluster, máquinas virtuales, políticas de backups, restauración de backups, snapshots. No es necesario instalar aplicaciones clientes en su máquina para administrar y siendo HTML5 le permite conectarse y gestionar el entorno virtualizado desde su Smartphone Android, Iphone, tablet's, entre otros.
- Virtualización para la mayoría de Sistemas Operativos, en sus versiones 32/64bits: Linux en todas sus versiones, Microsoft Windows 10 / 2016 / 2012 / 7 / 8/ 2003 / xp, Solaris, AIX, entre otros.
- KVM (Máquina virtual basada en el núcleo) es una solución para implementar virtualización sobre Linux. Puede funcionar en hardware x86/x86_64 y es necesario que el microprocesador tenga soporte de virtualización Intel "VT" y en AMD "SVM".
- Container-based Virtualization (LXC), es una alternativa para ejecutar máquina "Linux" en espacios separados. A diferencia de la virtualización este funciona como un módulo agregado al servidor físico y hace uso directo del hardware (también conocido como Paravirtualización).
- Backup & Restore de "Máquinas Virtuales". En Proxmox el efectuar estas tareas es muy sencillo y se administra a través de su interfaz Web. Puede efectuar un backup de forma inmediata o dejarlo

programado. La restauración es simple, solo debe de seleccionar el backup a restaurar y listo.

- Snapshot Live. le permite hacer copias instantáneas de "Máquinas Virtuales" incluyendo el contenido de la RAM, su configuración y el estado de los discos virtuales. Usted puede retroceder en tiempo la "Maquina Virtual" restaurando spanshot's.

Contenedores

Un contenedor de Linux es un conjunto de uno o más procesos separados del resto del sistema. Todos los archivos que se necesitan para ejecutarlos provienen de una imagen diferente, lo cual significa que los contenedores de Linux son portátiles y uniformes durante todas las etapas: desarrollo, prueba y producción. Esto los hace mucho más rápidos que los canales de desarrollo que necesitan replicar entornos de prueba tradicionales. Debido a su popularidad y facilidad de uso, los contenedores también son una parte importante de la seguridad de TI.

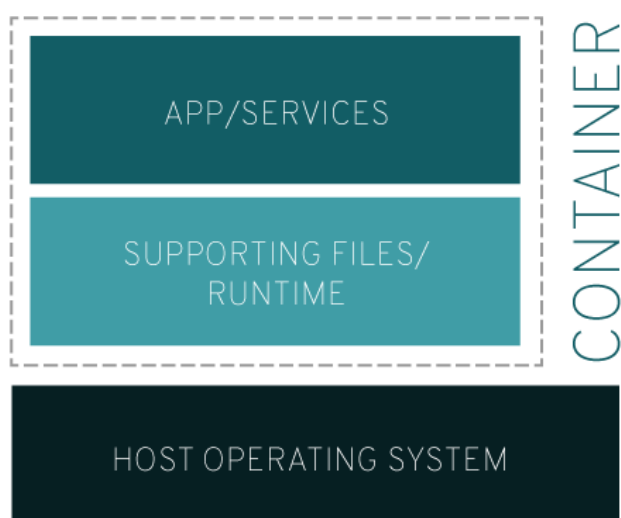


Figura 11: Esquema de funcionamiento de un contenedor

Durante el desarrollo de una aplicación, el entorno de desarrollo tiene una configuración específica. Estas configuraciones pueden variar respecto al entorno en el que trabaje cada desarrollador. La aplicación en proceso de desarrollo se basa en esa configuración y depende de archivos, bibliotecas y dependencias específicas. Por lo tanto, los entornos de desarrollo y producción que están estandarizados con sus propias configuraciones y sus propios conjuntos de archivos compatibles. Para emular esos entornos tanto como sea posible de forma local, pero sin los gastos que genera tener que recrear los entornos del servidor a fin de que pasen el control de calidad y que se implemente sin demasiadas dificultades, se puede hacer uso de los contenedores.

El contenedor donde se encuentra la aplicación tiene las bibliotecas, las dependencias y los archivos necesarios para que pueda pasar a la etapa de producción sin ninguna consecuencia grave. En realidad, el contenido de una imagen en contenedor puede considerarse como una instalación de una distribución de Linux, ya que incluye paquetes RPM, archivos de configuración, etc. Sin embargo, la distribución de las imágenes en contenedores es mucho más fácil que la instalación de nuevas copias de los sistemas operativos.

Este es un ejemplo sencillo, pero los contenedores de Linux se pueden aplicar a los problemas de muchas formas diferentes, donde se requiere la máxima portabilidad, capacidad de configuración y aislamiento. El objetivo de los contenedores de Linux es generar desarrollos con mayor rapidez y satisfacer las necesidades comerciales a medida que van surgiendo. En algunos casos, como con la transmisión de datos en tiempo real de Apache Kafka, los contenedores son fundamentales porque son la única forma de proporcionar la escalabilidad que necesita una aplicación. Los contenedores pueden desplegarse, sin importar la infraestructura en

que se encuentren (on-premise, en la nube o en una combinación de ambas).

Su diferenciación con la virtualización es que esta última permite que múltiples sistemas operativos se ejecuten simultáneamente en un solo sistema de hardware.

Los contenedores en cambio, comparten el mismo kernel del sistema operativo y separan los procesos de las aplicaciones del resto del sistema. Por ejemplo, los sistemas Linux ARM ejecutan contenedores de Linux ARM, los sistemas Linux x86 ejecutan contenedores de Linux x86, los sistemas Windows x86 ejecutan contenedores de Windows x86. Los contenedores de Linux son muy portátiles, pero deben ser compatibles con el sistema subyacente.

Es decir, la virtualización utiliza un hipervisor para emular el hardware que permite que varios sistemas operativos se ejecuten en paralelo. Pero ese hipervisor no es tan ligero como los contenedores. Si tiene recursos limitados con capacidades también limitadas, necesitará aplicaciones ligeras que se puedan implementar de forma masiva en un mismo entorno. Los contenedores de Linux se ejecutan de forma nativa en el sistema operativo y lo comparten con todos los demás contenedores, para que sus aplicaciones y servicios sigan siendo ligeros y se ejecuten rápidamente en paralelo.

Los contenedores de Linux cambian la forma de desarrollar, implementar y administrar las aplicaciones. Las imágenes en contenedores de Linux brindan portabilidad y control de versiones, lo cual garantiza que lo que funciona en la computadora portátil del desarrollador también funcione en la etapa de producción. En comparación con las máquinas virtuales, un

contenedor de Linux en ejecución consume menos recursos, tiene una interfaz estándar (inicio, interrupción, variables del entorno, etc.), mantiene el aislamiento de las aplicaciones y se gestiona con mayor facilidad como parte de una aplicación más grande (varios contenedores). Además, las aplicaciones en múltiples contenedores se pueden organizar en distintas nubes.

Docker

La palabra "DOCKER" se refiere a varias cosas. Esto incluye un proyecto de la comunidad open source; las herramientas del proyecto open source; Docker Inc., la empresa que es la principal promotora de ese proyecto; y las herramientas que la empresa admite formalmente. El hecho de que las tecnologías y la empresa compartan el mismo nombre puede ser confuso.

"Docker", el software de TI, es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux.

La comunidad open source Docker trabaja para mejorar estas tecnologías a fin de beneficiar a todos los usuarios de forma gratuita.

La empresa, Docker Inc., desarrolla el trabajo de la comunidad Docker, lo hace más seguro y comparte estos avances con el resto de la comunidad. También respalda las tecnologías mejoradas y reforzadas para los clientes empresariales.

DOCKER, utiliza los contenedores como máquinas virtuales extremadamente livianas y modulares. Además, obtiene flexibilidad con estos contenedores: puede crearlos, implementarlos, copiarlos y moverlos de un entorno a otro, lo cual le permite optimizar sus aplicaciones para la nube.

La tecnología Docker usa el kernel de Linux y las funciones de este, como Cgroups y namespaces, para segregar los procesos, de modo que puedan ejecutarse de manera independiente. El propósito de los contenedores es esta independencia: la capacidad de ejecutar varios procesos y aplicaciones por separado para hacer un mejor uso de la infraestructura y, al mismo tiempo, conservar la seguridad que se obtendría con sistemas separados.

Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.

Estas herramientas desarrolladas a partir de los contenedores de Linux, otorgan a los usuarios un acceso sin precedentes a las aplicaciones, la capacidad de implementar rápidamente y control sobre las versiones y su distribución.

Docker Vs. LXC.

Al principio, Docker se desarrolló a partir de la tecnología LXC, lo que la mayoría de las personas asocia con contenedores de Linux "tradicionales", aunque desde entonces se ha alejado de esa dependencia. LXC era útil como virtualización ligera, pero no ofrecía una buena experiencia al desarrollador ni al usuario. La tecnología Docker no solo aporta la capacidad de ejecutar contenedores; también facilita el proceso de creación y diseño de contenedores, de envío de imágenes y de creación de versiones de imágenes (entre otras cosas).

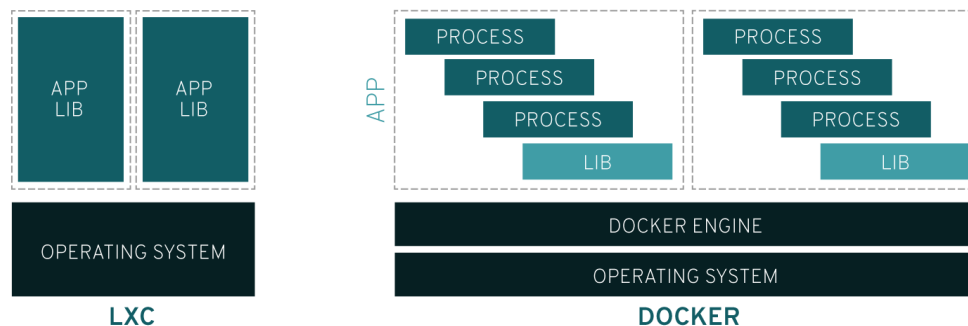


Figura 12: Diferencia entre Contenedores Linux y Docker

Los contenedores de Linux tradicionales usan un sistema init que puede gestionar varios procesos. Esto significa que las aplicaciones completas se pueden ejecutar como una sola. La tecnología Docker pretende que las aplicaciones se dividan en sus procesos individuales y ofrece las herramientas para hacerlo. Este enfoque granular tiene las siguientes particularidades:

- **Modularidad**

El enfoque Docker para la creación de contenedores se centra en la capacidad de tomar una parte de una aplicación, para actualizarla o repararla, sin necesidad de tomar la aplicación completa. Además de este enfoque basado en los microservicios, puede compartir procesos entre varias aplicaciones de la misma forma que funciona la arquitectura orientada al servicio (SOA).

- **Control de versiones de imágenes y capas**

Cada archivo de imagen de Docker se compone de una serie de capas. Estas capas se combinan en una sola imagen. Una capa se crea cuando la imagen cambia. Cada vez que un usuario especifica un comando, como ejecutar o copiar, se crea una nueva capa.

Docker reutiliza estas capas para construir nuevos contenedores, lo cual hace mucho más rápido el proceso de construcción. Los cambios intermedios se comparten entre imágenes, mejorando aún más la velocidad, el tamaño y la eficiencia -Keep it small: a closer look at Docker image sizing. El control de versiones es inherente a la creación de capas. Cada vez que se produce un cambio nuevo, básicamente, usted tiene un registro de cambios incorporado: control completo de sus imágenes de contenedor.

- Restauración

Probablemente la mejor parte de la creación de capas es la capacidad de restaurar. Toda imagen tiene capas. ¿No le gusta la iteración actual de una imagen? Restáurela a la versión anterior. Esto es compatible con un enfoque de desarrollo ágil y permite hacer realidad la integración e implementación continuas (CI/CD) desde una perspectiva de las herramientas.

- Implementación rápida

Solía demorar días desarrollar un nuevo hardware, ejecutarlo, proveerlo y facilitarlo. Y el nivel de esfuerzo y sobrecarga era extenuante. Los contenedores basados en Docker pueden reducir el tiempo de implementación a segundos. Al crear un contenedor para cada proceso, puede compartir rápidamente los procesos similares con nuevas aplicaciones. Y, debido a que un SO no necesita iniciarse para agregar o mover un contenedor, los tiempos de implementación son sustancialmente inferiores. Además, con la velocidad de implementación, puede crear y destruir la información creada por sus contenedores sin preocupación, de forma fácil y rentable.

Por lo tanto, la tecnología Docker es un enfoque más granular y controlable, basado en microservicios, que prioriza la eficiencia.

Portainer.

Se trata de una interfaz de usuario web que permite administrar fácilmente un host Docker y todos los contenedores que en él habitan.

Portainer tiene como una de sus ventajas principales su facilidad de uso. Consiste en un solo contenedor que puede ejecutarse en cualquier motor de Docker, tanto en GNU/Linux como en otros sistemas, como Microsoft Windows.

Esta herramienta permite administrar las pilas de Docker, los contenedores, las imágenes, volúmenes y redes. Es compatible con Docker CE, Docker EE y Docker Swarm.

Infraestructura

En este apartado, se describe cada una de las aplicaciones y servicios a utilizar. Cabe mencionar que cada uno de ellos estará contenido en un contenedor docker diferente en el trabajo.

MQTT.

MQTT (Message Queue Telemetry Transport) es un protocolo de transporte de mensajes Cliente/Servidor basado en publicaciones y suscripciones a los denominados “tópicos”. Cada vez que un mensaje es publicado será recibido por el resto de dispositivos adheridos a un tópico del protocolo.

Para dispositivos de Internet de las Cosas (IoT) es necesaria la conexión a Internet. La conexión a Internet permite que los dispositivos trabajen

entre ellos y con servicios backend. El protocolo de red subyacente de internet es TCP/IP. MQTT (Message Queue Telemetry Transport), que está construido sobre la pila de TCP/IP, se ha convertido en el estándar para las comunicaciones de IoT.

Originariamente, MQTT fue inventado y desarrollado por IBM a finales de los 90. Su aplicación original era conectar sensores de los oleoductos con satélites. Tal como sugiere su nombre, es un protocolo de mensajería que soporta la comunicación asíncrona entre las partes. Un protocolo de mensajería asíncrona disocia al emisor y al receptor de los mensajes tanto en espacio como en tiempo, y, por lo tanto, es escalable en entornos de red no confiables. A pesar de su nombre, no tiene nada que ver con las colas de mensajería, y, en cambio, utiliza un modelo de publicación y suscripción. A finales del 2014, se convirtió oficialmente en un estándar abierto de OASIS, y se soporta en lenguajes de programación populares mediante la utilización de varias implementaciones de código abierto.

MQTT es un protocolo de red liviano y flexible que logra el equilibrio adecuado para los desarrolladores de IoT:

- El protocolo liviano le permite implementarse en hardware de dispositivos altamente limitados y en redes con ancho de banda de alta latencia/limitado.
- Su flexibilidad hace que pueda soportar varios escenarios de aplicaciones para dispositivos y servicios de IoT.

Modelo de publicación y suscripción.

El protocolo MQTT define los tipos de entidades en la red: un intermediario de mensajes y un número de clientes. El intermediario es un servidor que recibe todos los mensajes de los clientes y luego los redirige

a clientes de destinos relevantes. Un cliente es cualquier cosa que pueda interactuar con el intermediario para enviar y recibir mensajes. Un cliente puede ser un sensor de IoT en el campo o una aplicación del centro de datos que procesa datos de IoT.

El cliente se conecta con el intermediario. Se puede suscribir a cualquier «tema» de mensajes de intermediario. Esta conexión puede ser una conexión TCP/IP simple o una conexión TLS cifrada para mensajes confidenciales.

El cliente publica el mensaje, sobre un tema, enviando el mensaje y el tema al intermediario.

Después, el intermediario redirige el mensaje a todos los clientes que están suscritos a ese tema.

Ya que los mensajes MQTT están organizados por temas, el desarrollador de aplicaciones tiene la flexibilidad de especificar que ciertos clientes solo puedan interactuar con determinados mensajes. Por ejemplo, los sensores publicarán sus lecturas sobre el tema «sensor_data» y se suscribirán al tema «config_change». Las aplicaciones de procesamiento de datos que guardan datos del sensor en una base de datos backend se suscribirán al tema «sensor_data». Una aplicación de consola de administración puede recibir los comandos del administrador del sistema para ajustar las configuraciones de los sensores, como la sensibilidad y la frecuencia de la muestra, y publicar esos cambios en el tema «config_change». (Vea El modelo de publicación y suscripción de MQTT para los sensores de IoT.)

El modelo de publicación y suscripción de MQTT para los sensores de IoT.

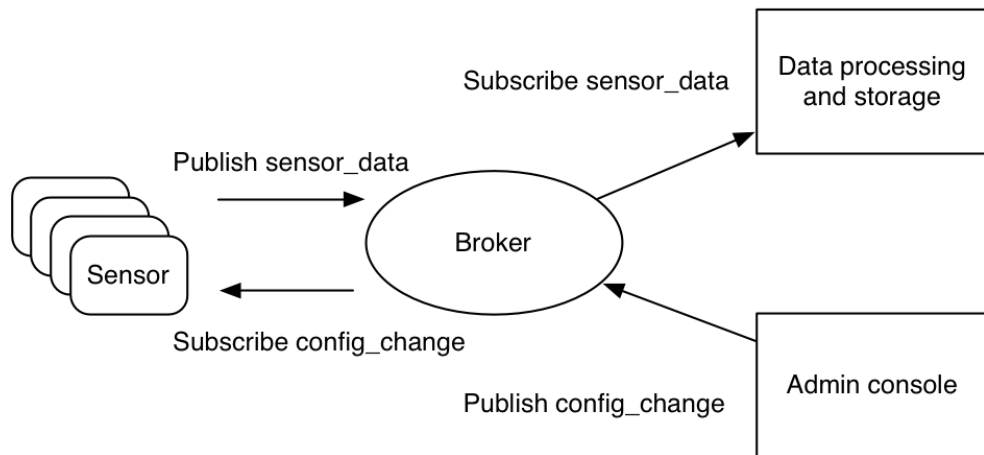


Figura 13: Funcionamiento de publicación y suscripción en MQTT

La imagen anterior, ejemplifica un diagrama de flujo que muestra mensajes de publicación y suscripción de datos del sensor utilizando un broker MQTT, almacenamiento de datos y una consola de administración

Al mismo tiempo, MQTT es liviano. Tiene una cabecera simple para especificar el tipo de mensaje, un tema basado en texto y, a continuación, una carga útil binaria y arbitraria. La aplicación puede utilizar cualquier formato de datos para la carga útil, como: JSON, XML, cifrado binario o Base64, siempre que los clientes destino puedan analizar la carga útil.

Protocolo de MQTT.

MQTT es un protocolo de cable que especifica cómo se organizan los bytes de datos y cómo se transmiten a través de la red TCP/IP. Pero para propósitos prácticos, los desarrolladores no necesitan entender el protocolo de cable. Todo lo que necesitan saber es que cada mensaje tiene un comando y una carga útil de datos. El comando define el tipo de mensaje (por ejemplo: un mensaje CONNECT o un mensaje

SUBSCRIBE). Todas las bibliotecas y herramientas de MQTT brindan formas sencillas de manipular directamente esos mensajes y pueden rellenar automáticamente algunos campos obligatorios, como los IDs del mensaje y del cliente.

Primero, el cliente se conecta con el intermediario enviando un mensaje CONNECT. El mensaje CONNECT pide establecer una conexión desde el cliente hacia el intermediario. El mensaje CONNECT tiene los siguientes parámetros de contenido.

Parámetros del mensaje CONNECT {: #parámetros-del-mensaje-connect}

Parámetro	Descripción
cleanSession	Este identificador especifica si la conexión es persistente o no. Una sesión persistente almacena todos los mensajes de la suscripción y los que potencialmente se han perdido (dependiendo de QoS) en el intermediario. (Vea Parámetros del mensaje SUBSCRIBE para obtener una descripción de QoS.)
username	La autenticación del intermediario y las credenciales de autorización.
password	La autenticación del intermediario y las credenciales de autorización.
lastWillTopic	Cuando una conexión se cae de forma inesperada, el intermediario publica automáticamente un mensaje de «última voluntad» en un tema.
lastWillQos	El mensaje de «última voluntad» en QoS. (Vea Parámetros del mensaje SUBSCRIBE para obtener una descripción de QoS.)
lastWillMessage	El propio mensaje de «última voluntad».
keepAlive	Este es el intervalo de tiempo para que el cliente compruebe la disponibilidad del intermediario para mantener la conexión viva.

El cliente recibirá un mensaje CONNACK del intermediario. El mensaje CONNACK tiene los siguientes parámetros de contenido.

Parámetros del mensaje CONNACK {: #parámetros-del-mensaje-connack}

Parámetro Descripción sessionPresent Esto indica si la conexión ya tiene una sesión persistente. Esto significa que la conexión ya tiene temas a los que se ha suscrito, y que recibirá la entrega de los mensajes que faltan. returnCode0 indica éxito. Otros valores identifican la causa de la falla.

Después de que se establece una conexión, el cliente puede enviar al intermediario uno o más mensajes SUBSCRIBE para indicar que recibirá mensajes del intermediario para determinados temas. Esos mensajes pueden tener una o varias repeticiones de los siguientes parámetros.

Parámetros del mensaje SUBSCRIBE {: #parámetros-del-mensaje-subscribe}

Parámetro Descripción qos El indicador qos (calidad de servicio, o QoS) indica la consistencia con la que los mensajes de este tema se tienen que entregar a los clientes.

- Valor 0: No confiable, el mensaje se entrega como mucho una vez, si el cliente no está disponible en ese momento se perderá el mensaje.
- Valor 1: el mensaje se debería entregar al menos una vez.
- Valor 2: el mensaje se debería entregar exactamente una vez.

Suscripción de temas.

Un tema puede tener varios niveles separados por el carácter de barra diagonal. Por ejemplo: «dw/demo» y «ibm/bluemix/mqtt» son temas válidos.

Después de que un cliente se haya suscrito correctamente a un tema, el intermediario devuelve un mensaje SUBACK con uno o más parámetros «returnCode».

Parámetros del mensaje SUBACK {: #parámetros-del-mensaje-suback}

Parámetro Descripción returnCode En el comando SUBSCRIBE existe un código de retorno para cada uno de los temas. Los valores de retorno son los siguientes.

Valores 0 – 2: éxito con el nivel de QoS correspondiente. (Vea Parámetros del mensaje SUBSCRIBE para saber más acerca de QoS.)

Valor 128: falla.

Corresponde al mensaje SUBSCRIBE, también puede dejar la suscripción (UNSUBSCRIBE) de un tema o de varios.

Parámetros del mensaje UNSUBSCRIBE {: #parámetros-del-mensaje-unsubscribe}

Parámetro Descripción tema Este parámetro se puede repetir para varios temas.

El cliente puede enviar mensajes PUBLISH al intermediario. El mensaje contiene un tema y una carga útil de datos. Después, el intermediario redirige el mensaje a todos los clientes que están suscritos a ese tema.

Parámetros del mensaje PUBLISH {: #parámetros-del-mensaje-publish}

Parámetro Descripción topicName El tema en que se publica el mensaje.qos La calidad del nivel de servicio de la entrega del mensaje. (Vea Parámetros del mensaje SUBSCRIBE para obtener una descripción de QoS.)retainFlag Este indicador indica si el intermediario retendrá el mensaje como el último mensaje conocido para este tema.carga útil Los datos reales que hay en el mensaje. Puede ser una cadena de texto o una bola binaria de datos (Yuan, 2017).

MQTT BROKER.

Es un gestor de datos que “vive” en algún lugar de la nube. Recibe todos los datos que están siendo publicados, y decide qué suscriptores deben recibirlos, según el interés de cada uno de ellos. Por lo tanto, una de las claves principales de la comunicación MQTT es el Broker, que es el programa que se encarga de recepcionar los mensajes enviados por los clientes y distribuirlos entre sí en un sistema pub-sub.

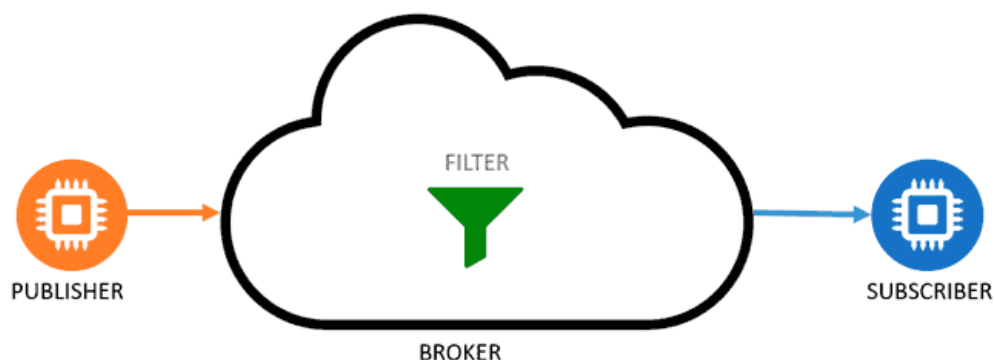


Imagen 14: Diagrama de funcionamiento en red de un MQTT Broker

Existe una gran cantidad de brokers MQTT disponibles cada uno con sus características, ventajas e inconvenientes. Los principales disponibles son:

- Mosquitto. Es el broker MQTT más conocido. Es un broker Open Source desarrollado por la fundación Eclipse y distribuido bajo licencia EPL/EDL. Está programado en C, y es multiplataforma. Es un broker liviano y adecuado para uso en servidores de baja potencia.
- Mosca. Es un broker MQTT Open Source para Node.js, desarrollado en Javascript por Matteo Collina. Puede ser empleado como aplicación independiente o embebido en cualquier proyecto de Node.js
- Aedes. Del mismo autor que Mosca, Aedes es un servidor broker MQTT Open Source para Node.js diseñado para ser un reemplazo de Mosca.
- HBMQTT. Es un broket MQTT Open Source escrito en Python que funciona sobre asyncio, introducido en Python 3.4.
- EMQTT. Erlan MQTT broker es Open Source, desarrollado en Erlang/OTP, está diseñado para aplicaciones con grandes exigencias en escalabilidad.
- RabbitMQ. Es un popular broker de mensajería AMQP Open Source, que también permite emplear el protocolo MQTT a través de un Adaptador.
- HiveMQ CE. La versión Community del popular HiveMQ es un broker Open Source basado en Java.
- ActiveMQ. Es un broker de mensajería JMS (Java Message Script) Open Source desarrollado por Apache, que también admite protocolo MQTT.
- Moquette. Un broker MQTT Open Source escrito en Java desarrollado por Eclipse, que destaca por ser muy liviano.
- MQTTnet. Un broker Open Source para .NET.

Son sólo algunos ejemplos de los broker más conocidos pero, existe todo un ecosistema, con un broker para cada lenguaje y cada proyecto, Mystique, Awesome MQTT, SurgeMQ, VerneMQ por nombrar algunos.

EMQX Dashboard

EMQ X (Erlang / Enterprise / Elastic MQTT Broker) es un broker de mensajes MQTT de código abierto basado en la plataforma Erlang/OTP. Erlang/OTP es una excelente plataforma de desarrollo distribuida, de baja latencia y en tiempo real.

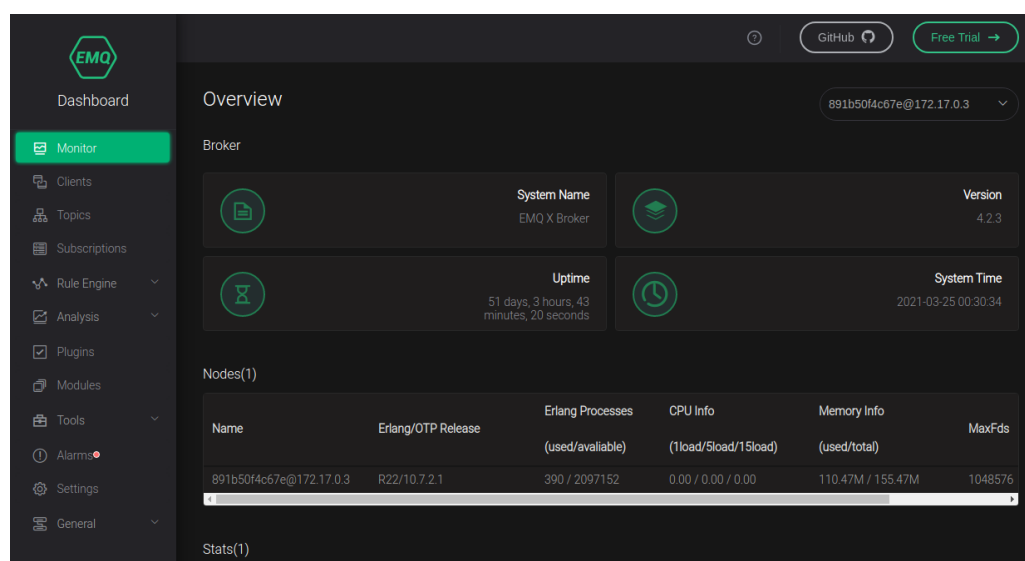


Figura 15: Consola de EMQ X Broker MQTT

EMQ X está diseñado para el acceso masivo de clientes y realiza un enrutamiento de mensajes rápido y de baja latencia entre dispositivos de red físicos masivos:

- Estable para alojar conexiones de cliente MQTT a gran escala y un nodo de servidor único que admite millones de conexiones.
- Clúster distribuido, enrutamiento de mensajes rápido y de baja latencia. Un solo clúster admite decenas de miles de rutas.
- Extensible, admite complementos personalizados, como autenticación y otras funciones.

- Compatibilidad completa con el protocolo IoT, incluidos MQTT, MQTT-SN, CoAP, LwM2M y otros protocolos patentados basados en TCP / UDP.

EMQ X Dashboard es una aplicación web a la que se puede acceder directamente a través del navegador sin instalar ningún otro software. EMQ X Broker proporciona un panel para facilitar a los usuarios la gestión de equipos y el seguimiento de los indicadores relacionados. A través de Dashboard, se puede ver la información básica del servidor, cargar y ver datos estadísticos, ver el estado de conexión de un cliente e incluso desconectarlo. Además, proporciona una interfaz de operación visual del motor de reglas y también integra una herramienta de cliente MQTT simple para pruebas de usuario. EMQ X Dashboard admite oyentes HTTP y HTTPS, pero solo los oyentes HTTP con un puerto de escucha de 18083 están habilitados de forma predeterminada.

La página de visualización predeterminada de Dashboard Overview proporciona información detallada de los nodos actuales de EMQ X Broker e información clave de otros nodos en el clúster para ayudar a los usuarios a obtener rápidamente el estado de cada nodo.

La página Clients proporciona una lista de clientes conectados al nodo especificado y también admite la búsqueda directa de clientes a través de Client ID. Además de ver la información básica del cliente, también al hacer clic en el Kick Outbotón en el lado derecho de cada registro, se puede expulsar al cliente. Esta operación desconectará al cliente y finalizará su sesión.

En la pestaña Subscriptions, en la parte de los detalles del cliente, se puede ver la información de suscripción del cliente actual y crear o

cancelar suscripciones. El motor de reglas de EMQ X Broker puede procesar mensajes y eventos de manera flexible, como convertir el mensaje a un formato específico y almacenarlo en una tabla de base de datos o reenviarlo a la cola de mensajes.

La página Plugins enumera todos los complementos que EMQ X Broker puede encontrar, incluidos los complementos oficiales de EMQ X y los complementos propios que se hayan desarrollado de acuerdo con los estándares oficiales de EMQ X.

La página Applications enumera las aplicaciones creadas actualmente. Se pueden realizar operaciones como crear aplicaciones, deshabilitar temporalmente o iniciar permisos de acceso para una aplicación determinada.

La página Websocket proporciona una herramienta de cliente WebSocket, incluye funciones de conexión, suscripción y publicación. Al mismo tiempo, se pueden ver los datos del mensaje que envía y recibe.

Adicionalmente, este Dashboard incluye las siguientes páginas API HTTP donde se muestran todas las API HTTP admitidas actualmente por EMQ X Broker y sus descripciones.

La página Users en la cual se pueden ver y administrar los usuarios que pueden acceder y operar en el Dashboard.

En la página Settings se puede configurar el idioma y el tema del panel y por último, la página Help donde se muestran los enlaces a foros y preguntas más frecuentes para ayudar a los usuarios en caso de problemas.

MariaDB

MariaDB es un sistema gestor de bases de datos (SGBD), es decir, un conjunto de programas que permiten modificar, almacenar, y extraer información de una base de datos. Disponiendo de otro tipo de funcionalidades como la administración de usuarios, y recuperación de la información si el sistema se corrompe, entre otras.

MariaDB surge a raíz de la compra, de la compañía desarrolladora de otro (SGBD) llamado MySQL, por la empresa Sun Microsystems. El desarrollador original, decide tomar el código fuente original de MySQL y genera un derivado con mejoras y cambios a los que llama MariaDB. Permitiendo así la existencia de una versión de este producto con licencia GPL (General Public License).

Características de MaríaDB.

Algunas de las características que diferencias a MariaDB de MySQL son:

Mecanismos de almacenamiento.

Amplían los paquetes binarios y fuentes, no solo con los mecanismos de almacenamiento standard (CSV,Blackhole, MyISAM ...) sino que añade los siguientes:

- Aria (alternativa a MyISAM resistente a caídas)
- XtraDB (reemplazo directo de InnoDB)
- PBXT (en MariaDB 5.1, 5.2 y 5.3. Deshabilitada en 5.5)
- FederatedX (reemplazo directo de Federated)
- OQGRAPH (nuevo en 5.2)
- SphinxSE (nuevo en 5.2)
- IBMDB2I Eliminada por Oracle de MySQL 5.1.55 pero se incluye en el código de MariaDB hasta la versión 5.5.

- TokuDB (Motor de almacenamiento de alto rendimiento basado en indexación de árboles fractales).
- Cassandra, en MariaDB 10.0 (otros mecanismos no-sql se incluirán en MariaDB)
- CONNECT (MariaDB 10.0)
- Sequence (10.0.3)
- Spider (MariaDB 10.0)

Mejoras de velocidad.

Se completan mejoras que aumentan la velocidad y el rendimiento, eliminando conversiones de caracteres innecesarias, uso del motor Aria que permite hacer consultas complejas con mayor rapidez. Mejoras sobre la tabla de chequeo de redundancia, replicación más rápida y segura.

Mejoras en las pruebas.

Reducen en la medida de lo posible el número de alertas del compilador y también mejoran el código de programación evitando introducción de nuevos errores o bugs.

Facilidad de uso.

Mejora la facilidad de uso e introduce actualizaciones del progreso en acciones como ALTER TABLE y LOAD DATA INFILE. Comandos como FLUSH y SHOW disponen de opciones para identificar la causa en la carga del SGBD. Y columnas dinámicas, que proporcionan al usuario columnas virtuales en las tablas.

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación

a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License

Eclipse Paho MQTT Python.

Como anteriormente se destacó, el protocolo MQTT es un protocolo de conectividad de máquina a máquina (M2M) / "Internet de las cosas". Diseñado como un transporte de mensajería de publicación / suscripción extremadamente liviano, es útil para conexiones con ubicaciones remotas donde se requiere una pequeña huella de código y / o el ancho de banda de la red es un bien escaso.

La biblioteca cliente Eclipse Paho MQTT Python, implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT.

Este código proporciona una clase de cliente que permite que las aplicaciones se conecten a un agente MQTT para publicar mensajes, suscribirse a temas y recibir mensajes publicados. También proporciona algunas funciones auxiliares para que la publicación de mensajes únicos en un servidor MQTT sea muy sencilla.

Es compatible con Python 2.7.9+ o 3.5+.

Layout

Mediante el Layout, se verán reflejados los datos recolectados y almacenados en la Base de Datos en un entorno gráfico. Si bien su construcción no es objeto del presente trabajo, es un complemento que

resuelve el modo de mostrar la información y será imprescindible para un desarrollo de producción final.

Particularmente, en el presente trabajo, se decidió utilizar un Layout basado en PHP que grafique por nodo los últimos 100 registros extraídos de la Base de Datos.

Apache.

El servidor HTTP Apache es un servidor web HTTP de código abierto, para múltiples plataformas, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual según la normativa RFC 2616.

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Apache tiene amplia aceptación en la red: desde 1996, Apache es el servidor HTTP más usado. Jugó un papel fundamental en el desarrollo de la World Wide Web.

PHP.

El acrónimo recursivo de PHP es: Hypertext Preprocessor. Se trata de un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo". El código de PHP está encerrado entre las etiquetas especiales de comienzo y final `<?php` y `?>` que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP.

PHP es un código simple y a su vez ofrece muchas características avanzadas.

webdevops/php-apache-dev.

Esta imagen se extiende de otra que ejecuta PHP en modo de desarrollo: webdevops/php-dev. A diferencia de ella, la presente imagen, cuenta con un demonio apache que se ejecuta en el puerto 80 y 443.

Además, la imagen seleccionada usa supervisor y ejecuta el demonio bajo el usuario application(UID 1000; GID 1000) de forma predeterminada. Si el contenedor se inicia con un usuario diferente, el demonio se ejecutará con el uid especificado.

Despliegue de webdevops/php-apache-dev.

Dentro de Portainer, se desplegará como nuevo contenedor teniendo en cuenta los puertos que se desea exponer en el servidor.

Name: Layout

Image configuration

Registry: DockerHub

Image: docker.io webdevops/php-apache-dev:8.0

Advanced mode

Always pull the image ☒

Network ports configuration

Publish all exposed network ports to random host ports ☐

Manual network port publishing [publish a new network port](#)

host	8081	→	container	80	TCP	UDP	
host	8082	→	container	443	TCP	UDP	
host	8083	→	container	9000	TCP	UDP	

Figura 16: Contenedor layout con los servicios integrados php y apache y los puertos del host en los que se exponen.

Google Cloud Platform.

Google Cloud Platform (GCP), ofrece todas las herramientas de Google disponibles en la nube. Este conjunto de servicios aporta todas las herramientas necesarias para diseñar, hacer testings y lanzar aplicaciones desde gcloud, sus herramientas son de código abierto y fomentan la innovación conjunta, la interoperabilidad y la portabilidad.

API de Google Cloud.

Las API de Cloud, permiten automatizar los flujos de trabajo utilizando utilizando el lenguaje que se prefiera. Es posible usar estas API de Cloud con llamadas REST o bibliotecas de cliente en los lenguajes de programación más extendidos. Entre ellas, se destaca la API que se describe a continuación.

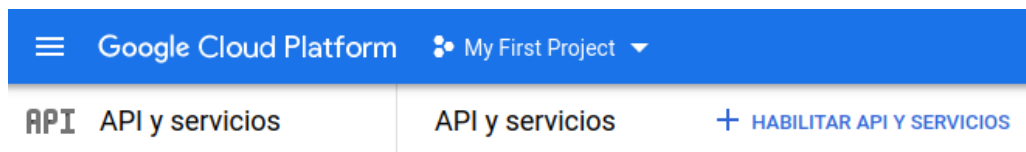


Figura 17: API y servicios de Google

Maps JavaScript API

La API de JavaScript de Maps, permite personalizar mapas con contenido e imágenes propios para mostrarlos en páginas web y dispositivos móviles. La API de JavaScript de Maps presenta cuatro tipos de mapas básicos (mapa de ruta, satélite, híbrido y terreno) que puede modificar utilizando capas y estilos, controles y eventos, y varios servicios y bibliotecas.

Google Maps Platform	API																														
Descripción general																															
API	API habilitadas Selecciona una API para ver los detalles. Las cifras corresponden a los últimos 30 días.																														
Métricas																															
Cuotas																															
Credenciales																															
Asistencia																															
Solución Locator Plus																															
Solución de autocompletar																															
Administración de mapas																															
Estilos de mapa																															
	<table> <tr> <th>API ↑</th><th>Solicitudes</th></tr> <tr> <td>Directions API</td><td>-</td></tr> <tr> <td>Distance Matrix API</td><td>-</td></tr> <tr> <td>Geocoding API</td><td>-</td></tr> <tr> <td>Geolocation API</td><td>-</td></tr> <tr> <td>Maps Elevation API</td><td>-</td></tr> <tr> <td>Maps Embed API</td><td>-</td></tr> <tr> <td>Maps JavaScript API</td><td>48</td></tr> <tr> <td>Maps SDK for Android</td><td>-</td></tr> <tr> <td>Maps SDK for iOS</td><td>-</td></tr> <tr> <td>Maps Static API</td><td>-</td></tr> <tr> <td>Places API</td><td>-</td></tr> <tr> <td>Roads API</td><td>-</td></tr> <tr> <td>Street View Static API</td><td>-</td></tr> <tr> <td>Time Zone API</td><td>-</td></tr> </table>	API ↑	Solicitudes	Directions API	-	Distance Matrix API	-	Geocoding API	-	Geolocation API	-	Maps Elevation API	-	Maps Embed API	-	Maps JavaScript API	48	Maps SDK for Android	-	Maps SDK for iOS	-	Maps Static API	-	Places API	-	Roads API	-	Street View Static API	-	Time Zone API	-
API ↑	Solicitudes																														
Directions API	-																														
Distance Matrix API	-																														
Geocoding API	-																														
Geolocation API	-																														
Maps Elevation API	-																														
Maps Embed API	-																														
Maps JavaScript API	48																														
Maps SDK for Android	-																														
Maps SDK for iOS	-																														
Maps Static API	-																														
Places API	-																														
Roads API	-																														
Street View Static API	-																														
Time Zone API	-																														

Figura 18: API habilitadas de Google y solicitudes por API

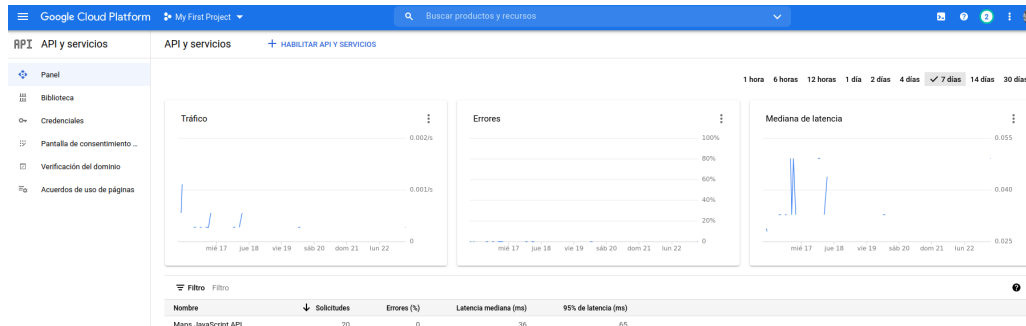


Figura 19: Estadísticas de la API de JavaScript de Maps utilizada

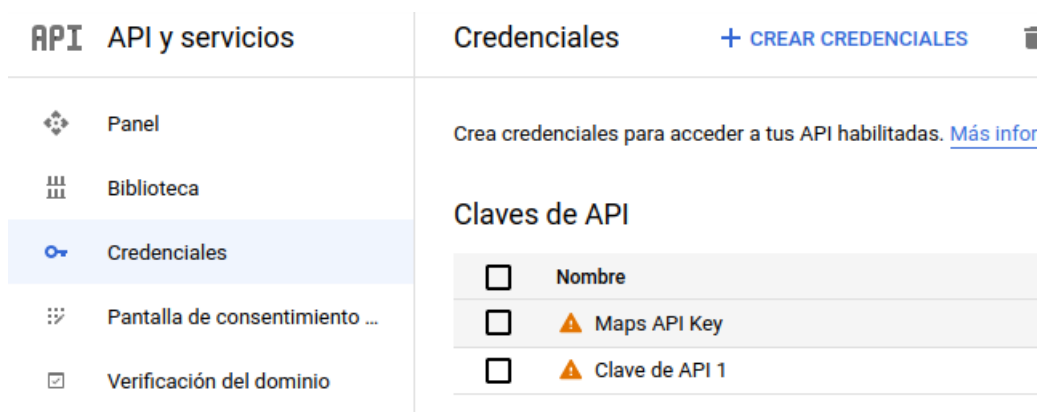


Figura 20: Credenciales para la API de JavaScript para Google Maps

Modelos de sistemas de predicción de comportamientos de Incendios forestales

Existen diversos modelos y sistemas de predicción de comportamiento de incendios forestales, y la elección del más eficiente dependerá de diversos factores, como la disponibilidad de datos, la complejidad del terreno, la precisión deseada, entre otros (Liu et al., s. f.). Algunos de los modelos más utilizados son:

Modelo FARSITE: Este es un modelo de propagación de fuego que utiliza datos topográficos, de combustible y meteorológicos para predecir la dirección y velocidad de propagación del fuego.

Modelo WFDS: Este es un modelo de dinámica de fluidos computacional (CFD) que utiliza la física del flujo de aire y la propagación del fuego para predecir la propagación del incendio.

Modelo BEHAVE: Este es un modelo de comportamiento de incendios que utiliza datos de combustible, meteorológicos y topográficos para predecir el comportamiento del fuego.

Modelo NEXUS: Este es un modelo de predicción de incendios forestales que utiliza datos de satélite y meteorológicos para predecir la propagación del fuego.

Cada modelo tiene sus ventajas y desventajas y se pueden combinar para obtener una mayor precisión. Además, existen técnicas para optimizar los resultados de estos modelos mediante la selección de características y calibración de parámetros del modelo. Entre ellas, se puede nombrar a los métodos de reducción de incertidumbre basados en HPC, Evolución Diferencial y modelos de Islas (Miguel et al., 2014).

La técnica de evolución diferencial puede ser utilizada por muchos modelos de predicción de incendios forestales, como los modelos basados en redes neuronales, máquinas de soporte vectorial, y modelos basados en lógica difusa, entre otros. Tiene ventajas como una buena capacidad de convergencia, robustez, facilidad de implementación y puede ayudar a mejorar la precisión y eficiencia del modelo de predicción al ajustar automáticamente los parámetros del modelo. Además, es una

técnica que se puede adaptar a diferentes tipos de problemas de predicción de incendios forestales.

Sin embargo, la eficacia del modelo de evolución diferencial dependerá de la calidad de los datos de entrada y de la elección adecuada de los parámetros del modelo. Por lo tanto, se requiere una evaluación cuidadosa y una optimización de los parámetros para lograr los mejores resultados.

Cuando se utiliza esta técnica utilizando islas, su optimización es metaheurística y se utiliza para resolver problemas de optimización en los que se dispone de múltiples procesadores o unidades de procesamiento. Por lo tanto, puede ser utilizada en combinación con otros modelos para mejorar su eficiencia y precisión. La evolución diferencial por islas se ha utilizado en diversos campos de aplicación, incluyendo la optimización de sistemas energéticos, el diseño de sistemas de control, la predicción de enfermedades y la clasificación de imágenes, entre otros (Méndez Garabetti et al., 2020).

En función del cálculo necesario para la obtención de variables requeridas por los modelos de predicción de comportamiento de incendios forestales, el uso de tecnologías como las redes de sensores inalámbricos y la inteligencia artificial pueden mejorar aún más la eficiencia de los sistemas de predicción de comportamientos de incendios forestales

Capítulo III: Desarrollo

En este capítulo se detalla la solución propuesta para la captura de datos, su transporte, integración a una base de datos y publicación de los mismos en una página web.

El presente está ordenado desde la captura hacia el usuario final, de modo tal que las temáticas se presentan del siguiente modo:

- WSN Propuesta: se detalla la WSN, configuración y modo de trabajo de los equipos, programación y sensores dispuestos.
- Destino de los Datos: En este punto se detalla la infraestructura utilizada a fin de procesar la información enviada por la WSN. Para ello, se describe el despliegue de la máquina virtual que contendrá cada contenedor en el que se generará el despliegue de: Docker, Portainer, broker EMQX, María DB, Python y Layout.

WSN PROPUESTA

En virtud de lo visto, el presente trabajo, propone una red WSN utilizando el protocolo 802.11. La misma estará formada por una red de malla WiFi de modo tal que todos los nodos que tengan conectados sensores, puedan comunicarse entre sí y compartir datos a través de sus nodos vecinos o circundantes hacia un Gateway que reciba la información y la procese o bien la reenvíe a internet a fin de que sea procesada.

En este caso en particular, el Gateway estará conectado a Internet y servirá de nexo entre la WSN e Internet.

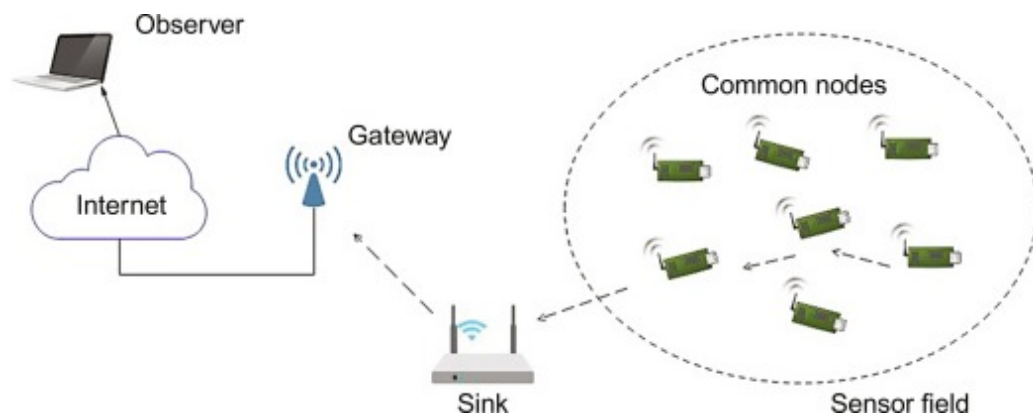


Figura 21: Red WSN Típica

Funcionamiento de la WSN y comunicación.

A fin de evaluar la regeneración y configuración automática de los nodos y a fin de que los datos sensados, sean enviados a una estación base a pesar de fallas o cambios en la red, se montó una WSN en laboratorio formada por cuatro nodos conectados mediante el protocolo 802.11, conformando un clúster. Los nodos siguieron el siguiente esquema que se observa en la siguiente imagen.

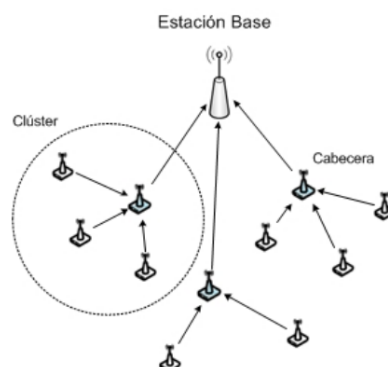


Figura 22: Esquema de cluster WSN

Debido a que se representa un solo clúster, uno de los nodos hace de receptor de la estación base, la estación base es una PC, y de los tres nodos restantes uno tiene el rol de cluster-head con dos nodos equipados con sensores. Cada nodo posee un identificador de nodo representado por un número entero de 6 cifras. Particularmente, en esta experimentación, se emplearon cuatro nodos esp8266-D1 WeMos.

Nodo	ID	Rol
0	Sin ID, receptor estación base.	Se encuentra conectado al puerto USB de una PC y recibe la información del nodo más próximo
1	534789 (gris claro)	Sin sensores
2	529993 (gris oscuro)	Con un sensor de Humedad y Temperatura conectado
3	535864 (gris)	Con un sensor de Humedad y Temperatura conectado

En las pruebas de laboratorio realizadas, se observa la captura de datos de una WSN donde se refleja el comportamiento de la red ante los cambios de topologías y pérdidas de nodos realizados en la experimentación.

Las líneas “adopt” representan los cambios de topologías, donde un valor igual a 1 indica que el actual nodo cluster-head ha dejado de estar operativo y otro nodo ha tomado su rol, y 0 cuando se ha incorporado otro nodo como cluster-head. En el resto de las líneas sólo se reciben mediciones de los sensores conectados en los nodos 2 y 3 siguiendo el formato: se recibe información de un nodo “X” a través de un nodo “Y” (la última columna indica los valores medidos).

Los datos de la figura que se muestra a continuación, fueron capturados desde el Nodo 0, conectado a la PC mediante el puerto USB. Por lo tanto, los mensajes: “Received from”, provienen del nodo más próximo conectado a éste mientras que “Msg=Nodo:” indica el nodo original de donde provienen los datos.

/dev/ttyUSB0				
BLOQUE "A"	startHere: Received from	534789	msg=Nodo: 529993	25.00% 32.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 535864	25.00% 30.00
	startHere: Received from	534789	msg=Nodo: 529993	25.00% 18.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
BLOQUE "B"	startHere: New Connection, adopt=1			
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 42.00
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 43.00
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 18.00
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 27.00
BLOQUE "C"	startHere: New Connection, adopt=0			
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 27.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	529993	msg=Nodo: 529993	27.00% 28.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	529993	msg=Nodo: 529993	25.00% 28.00
	startHere: Received from	529993	msg=Nodo: 529993	nan% nan
	startHere: Received from	529993	msg=Nodo: 529993	32.00% 27.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	529993	msg=Nodo: 529993	33.00% 26.00
BLOQUE "D"	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 535864	25.00% 45.00
	startHere: Received from	534789	msg=Nodo: 534789	nan% nan
	startHere: Received from	534789	msg=Nodo: 535864	25.00% 42.00
	startHere: Received from	534789	msg=Nodo: 535864	25.00% 18.00
	startHere: New Connection, adopt=1			
BLOQUE "E"	startHere: Received from	535864	msg=Nodo: 535864	24.00% 43.00
	startHere: Received from	535864	msg=Nodo: 529993	33.00% 41.00
	startHere: Received from	535864	msg=Nodo: 529993	24.00% 31.00
	startHere: Received from	535864	msg=Nodo: 529993	25.00% 18.00
	startHere: Received from	535864	msg=Nodo: 535864	24.00% 44.00
	startHere: Received from	535864	msg=Nodo: 535864	24.00% 44.00
	startHere: Received from	535864	msg=Nodo: 529993	23.00% 30.00
	startHere: Received from	535864	msg=Nodo: 535864	24.00% 21.00
	startHere: Received from	535864	msg=Nodo: 535864	29.00% 17.00
	startHere: Received from	535864	msg=Nodo: 529993	25.00% 30.00

Figura 23: Datos recibidos por el Gateway de la WSN

La experimentación se puede ver gráficamente en la siguiente figura, donde se puede apreciar en bloques los cambios de topología graficados.

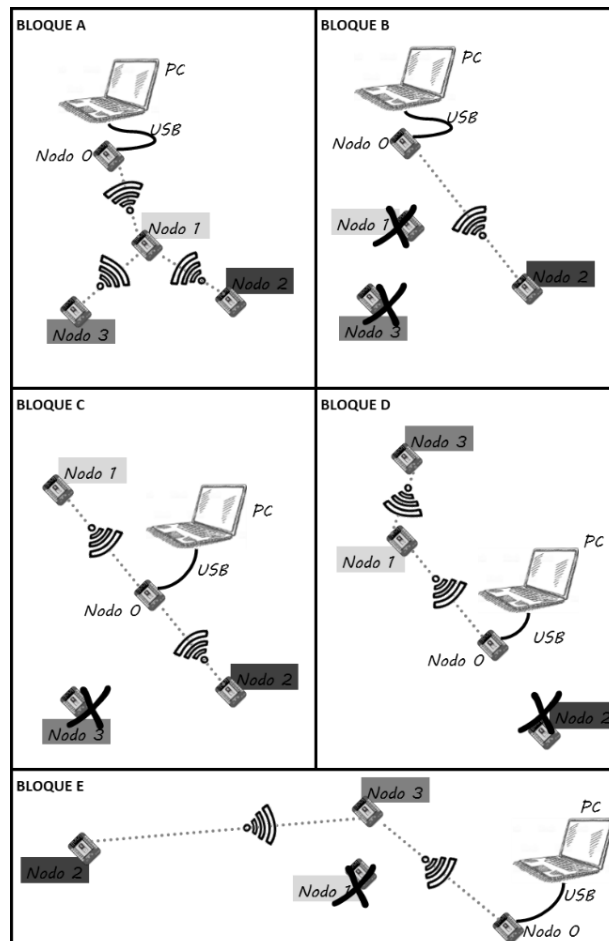


Figura 24: Disposición física y experimentación con la WSN descrita

Los bloques representan distintos escenarios, descritos a continuación:

BLOQUE “A”: El *Nodo1* es el más próximo al *Nodo0*. Mediante el *Nodo1*, se reciben los datos capturados por los *Nodos 2 y 3*.

BLOQUE “B”: Cambio de Topología. Se apaga el Nodo 1. Automáticamente el Nodo2 advierte que es el más cercano al Nodo0, se configura como “Head” y envía la información de sus sensores.

BLOQUE “C”: Cambio de Topología. Se coloca y se enciende el Nodo 1 en un espacio físico de modo tal que sea más cercano al Nodo0 que al Nodo2 y se enciende. Por lo que el Nodo0 recibe la información del mismo en forma directa.

BLOQUE “D”: Cambio de topología. Se coloca y se enciende el Nodo 3 cercano al Nodo1. Se apaga el Nodo2. No se visualiza una nueva adopción debido a que el Nodo1 ya estaba conectado al Nodo0. Mediante el Nodo1 se reciben los datos enviados por el Nodo 3.

BLOQUE “E”: Cambio de Topología. Se coloca y se enciende el Nodo 2 en un espacio físico cercano al Nodo 3. Se apaga el Nodo1. Se visualiza la adopción del Nodo3 como el más cercano al Nodo0 y desde el Nodo3 se reciben los datos enviados por el Nodo2

Programación de ESP8266 D1 Wemos Mini.

Visto y teniendo en cuenta todo lo necesario para deployar una WSN tipo mesh en un ESP8266 D1 Wemos Mini utilizando el IDE Arduino y considerando las librerías que serían necesarias para que ello funcione, veremos en este apartado el código utilizado.

En este trabajo se utilizó dos tipos de sensores: DHT11 para medir temperatura y humedad de ambiente y MQ7 a fin de obtener mediciones de Monóxido de Carbono.

Programación del DHT11 mediante IDE Arduino.

Para la misma, fué necesaria la integración de la Librería DHT.h al IDE Arduino. A continuación se detalla la programación del sensor en IDE Arduino.

```
////////CAPTURA DE DATOS DE HUMEDAD Y TEMPERATURA Y MUESTRA POR
PUERTO SERIE

#include "DHT.h"

// Crea el objeto DHT - temperatura y humedad
DHT dht1(D4, DHT11);

void setup(void) {
  Serial.begin(115200); //Velocidad del Puerto
  dht1.begin(); //Inicia DHT
}

void loop() {
  // Read DHT temperature and humidity values
  float DHT11_t = dht1.readTemperature();
  float DHT11_h = dht1.readHumidity();

  Serial.print("DHT11 ");
  Serial.print(DHT11_t,1); Serial.print("°C ");
  Serial.print(DHT11_h,1); Serial.println("% RH");
  Serial.println();

  delay(1000);
}
////////
```

Este Código lee y muestra en el puerto serial la temperatura y la humedad cada 1 segundo.

MQ-7

A continuación se muestra la programación en IDE Arduino para el funcionamiento y sensado del Sensor.

```
////////// CAPTURA DE DATOS DE MONÓXIDO DE CARBONO Y MUESTRA POR
PUERTO SERIE
float sensorVoltage;
float sensorValue;

sensorValue = analogRead(A0);
sensorVoltage = sensorValue/1024*5.0;
/* Líneas a Mostrar*/
Serial.print("sensor MQ-7 nvoltage = ");
Serial.print(sensorVoltage);
Serial.println(" V");
//////////
```

Red mesh en esp8266 y envío de datos

Como hemos visto, ya contamos con el nodo, los sensores y los datos a medir. En este apartado, utilizando lo descrito respecto a la WSN y mesh 802.11, se analizará el código utilizado en los nodos para formar una red Mesh y el utilizado en Gateway que será el encargado de recoger la información.

Además en este apartado, se puede observar el formato que se le dará a los datos que se enviarán mediante MQTT de modo tal que puedan ser comprendidos por otros clientes para el análisis de datos.

En este sentido, se utilizó el siguiente formato de datos separado por comas:

ID DEL NODO	TEMPERATURA	HUMEDAD	CO (monóxido de C)
-------------	-------------	---------	--------------------

Por lo tanto, si el nodo 38 de la WSN en un momento dado envía un mensaje MQTT con los datos obtenidos por sus sensores: 27.34°C, 79.14%RH y 0.99CO, siguiendo el formato descrito, se enviará el siguiente dato en el mensaje:

```
0038,27.34,79.14,0.99
```

Como se puede observar, el contenido de un mensaje MQTT será equivalente a un archivo separado por comas, lo que será más fácil de decodificar por los clientes que necesiten analizar los datos o bien almacenarlos en una Base de Datos.

Nodos de la Red Mesh.

En cada nodo de la red mesh, deberá cargarse los datos de conexión e identificación de nodo correspondiente en el código que se presenta a continuación, donde se detalla su funcionamiento.

```
#include "DHT.h"
#include "painlessMesh.h"

#define LED 2 // GPIO number of connected LED, ON ESP-12 IS GPIO2
#define BLINK_PERIOD 3000000 // microseconds until cycle repeat
#define BLINK_DURATION 100000 // microseconds LED is on for

#define MESH_PREFIX "SSID DE LA MESH"
```

```

#define MESH_PASSWORD "CONTRASEÑA DE LA MESH"
#define MESH_PORT 5555

//ID Nodo Destino (QUIEN SERÁ EL GATEWAY)
int destsvr = XXXXXXXXXXXX;

painlessMesh mesh;

// Create the DHT temperature and humidity sensor object
DHT dht1(D4, DHT11); //Pin del WeMos y el elemento

void setup() {
  Serial.begin(115200);
  dht1.begin();

  pinMode(LED, OUTPUT);

  // sincronizar();
  mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION ); // set before init()
  so that you can see startup messages

  mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT );
  mesh.onReceive(&receivedCallback);
}

void loop() {

  // Read DHT temperature and humidity values
  float DHT11_t = dht1.readTemperature();
  float DHT11_h = dht1.readHumidity();

  /* Lineas a mostrar
  Serial.print("DHT11 ");

```

```

Serial.print(DHT11_t,1); Serial.print("°C ");
Serial.print(DHT11_h,1); Serial.println("% RH");
Serial.println();
*/
//MQ-7
float sensorVoltage;
float sensorValue;

sensorValue = analogRead(A0);
sensorVoltage = sensorValue/1024*5.0;
/* Lineas a Mostrar
Serial.print("sensor MQ-7 nvoltage = ");
Serial.print(sensorVoltage);
Serial.println(" V");
*/
Serial.println();
Serial.println();
delay(3000);
*/
//Serial.println(mesh.getNodeId());

//Mensaje para enviar
//Destapar en nodos
int nid = mesh.getNodeId();
//Nodo:
String nidu = "02"; //Identificación del nodo que enviará el mensaje - Más adelante se
utilizará. En la línea siguiente, se genera el mensaje que se enviará a través de la red
mesh:
String msglm = nidu + "," + DHT11_t + "," + DHT11_h + "," + sensorVoltage;
mesh.sendSingle(destsvr, msglm);
//Serial.print(msglm);

delay(2000);

bool error;

```

```

mesh.update();

// Ejecutar el blink led
bool onFlag = true;
uint32_t cycleTime = mesh.getNodeTime() % BLINK_PERIOD;
for (uint8_t i = 0; i < (mesh.getNodeList().size() + 1); i++) {
    uint32_t onTime = BLINK_DURATION * i * 2;

    if (cycleTime > onTime && cycleTime < onTime + BLINK_DURATION)
        onFlag = false;
}
digitalWrite(LED, onFlag);
//mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC |
COMMUNICATION | GENERAL | MSG_TYPES | REMOTE );
//Serial.println(mesh.getNodeList().size());
// sincronizar();
}

void receivedCallback( uint32_t from, String &msg ) {
    //Serial.printf("Nodo de eco: Recibido de %u msg=%s\n", from, msg.c_str());
    //Serial.printf("STARTUP %u msg=%s\n", STARTUP);
    mesh.sendSingle(destsvr, msg);
}

```

Gateway.

En el código que se muestra a continuación, se explica el funcionamiento y se puede observar que una vez recibidos los datos de las estaciones, éste mediante una conexión a internet, enviará los datos al broker MQTT.

```

#include <Arduino.h>
#include <painlessMesh.h>
#include <PubSubClient.h>
#include <WiFiClient.h>

```

```

//Datos de Conexión de la Red Mesh
#define MESH_PREFIX "SSID DE LA MESH"
#define MESH_PASSWORD "CONTRASEÑA DE MESH"
#define MESH_PORT 5555
//Datos de Conexión WiFi
#define STATION_SSID "SSID DE WIFI"
#define STATION_PASSWORD "CONTRASEÑA WIFI"

#define HOSTNAME "MQTT_Bridge"

// Prototypes
void receivedCallback( const uint32_t &from, const String &msg );
void mqttCallback(char* topic, byte* payload, unsigned int length);

IPAddress getlocalIP();

IPAddress myIP(0,0,0,0);
IPAddress mqttBroker(XX, XX, XX, XX); //IP del Broker Mqtt (se verá más adelante)

painlessMesh mesh;
WiFiClient wifiClient;
PubSubClient mqttClient(mqttBroker, 1883, mqttCallback, wifiClient);

void setup() {
  Serial.begin(115200);

  mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION ); // set before init()
  so that you can see startup messages

  // Channel set to 6. Make sure to use the same channel for your mesh and for you
  other
  // network (STATION_SSID)
  mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, WIFI_AP_STA, 6 );
  mesh.onReceive(&receivedCallback);

```

```

mesh.stationManual(STATION_SSID, STATION_PASSWORD);
mesh.setHostname(HOSTNAME);

    // Bridge node, should (in most cases) be a root node. See [the
wiki](https://gitlab.com/painlessMesh/painlessMesh/wikis/Possible-challenges-in-mesh-
formation) for some background
    mesh.setRoot(true);
    // This node and all other nodes should ideally know the mesh contains a root, so call
this on all nodes
    mesh.setContainsRoot(true);
}

void loop() {
    mesh.update();
    mqttClient.loop();

    if(myIP != getLocalIP()){
        myIP = getLocalIP();
        Serial.println("My IP is " + myIP.toString());

        if (mqttClient.connect("painlessMeshClient")) {
            mqttClient.publish("painlessMesh/from/gateway", "Ready!");
            mqttClient.subscribe("painlessMesh/to/#");
        }
    }
}

void receivedCallback( const uint32_t &from, const String &msg ) {
    Serial.printf("bridge: Received from %u msg=%s\n", from, msg.c_str());
    String topic = "topic"; //aquí va el topic en el se publicará (más adelante, en el tema
MQTT Broker se detallará este punto)
    // String topic = "painlessMesh/from/" + String(from);
    mqttClient.publish(topic.c_str(), msg.c_str()); //En esta línea se publica el valor
sensado en un topic determinado.
}

```

```

void mqttCallback(char* topic, uint8_t* payload, unsigned int length) {
    char* cleanPayload = (char*)malloc(length+1);
    memcpy(cleanPayload, payload, length);
    cleanPayload[length] = '\0';
    String msg = String(cleanPayload);
    free(cleanPayload);

    String targetStr = String(topic).substring(16);

    if(targetStr == "gateway")
    {
        if(msg == "getNodes")
        {
            auto nodes = mesh.getNodeList(true);
            String str;
            for (auto &&id : nodes)
                str += String(id) + String(" ");
            mqttClient.publish("painlessMesh/from/gateway", str.c_str());
        }
    }
    else if(targetStr == "broadcast")
    {
        mesh.sendBroadcast(msg);
    }
    else
    {
        uint32_t target = strtoul(targetStr.c_str(), NULL, 10);
        if(mesh.isConnected(target))
        {
            mesh.sendSingle(target, msg);
        }
        else
        {
            mqttClient.publish("painlessMesh/from/gateway", "Client not connected!");
        }
    }
}

```

```

    }
  }
}

IPAddress getlocalIP() {
    return IPAddress(mesh.getStationIP());
}

```

Destino de los Datos

Hasta aquí, se ha evaluado y se ha desarrollado el despliegue de una WSN Mesh a fin de recolectar datos que son enviados hacia un Gateway. En este capítulo, se dará detalle de la infraestructura necesaria y modo en el que los datos puedan ser consultados, ingresados a otros sistemas o bien almacenados. Para ello, se utilizó Docker sobre una VM creada en Proxmox.

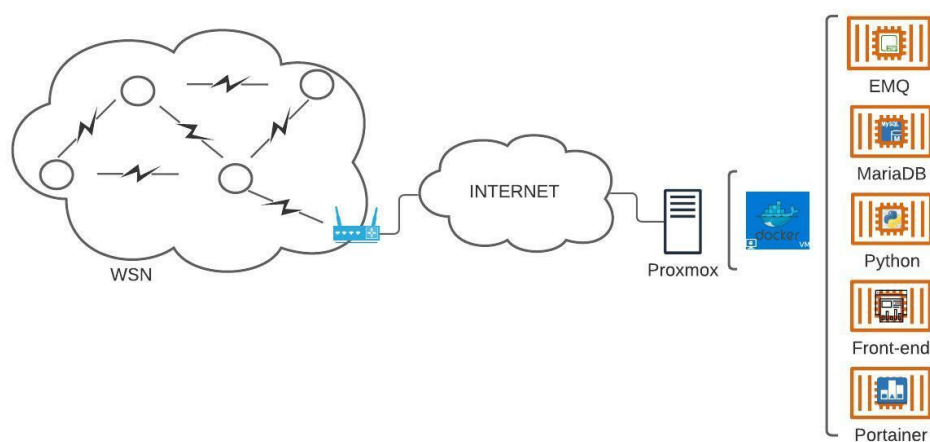


Figura 25: Diagrama de la Infraestructura planteada

A continuación, se dará detalle de la generación de la infraestructura.

Despliegue de VM en Proxmox.

En este trabajo de Tesis, se generó una VM sobre Proxmox basada en un Sistema Operativo Debian 10 con las siguientes características:

- 1 CPU
- 2GB de RAM
- 60GB de Disco Rígido
- 1 Placa de Red publicada en Internet

Dentro de la misma, estará desplegada toda la infraestructura necesaria.

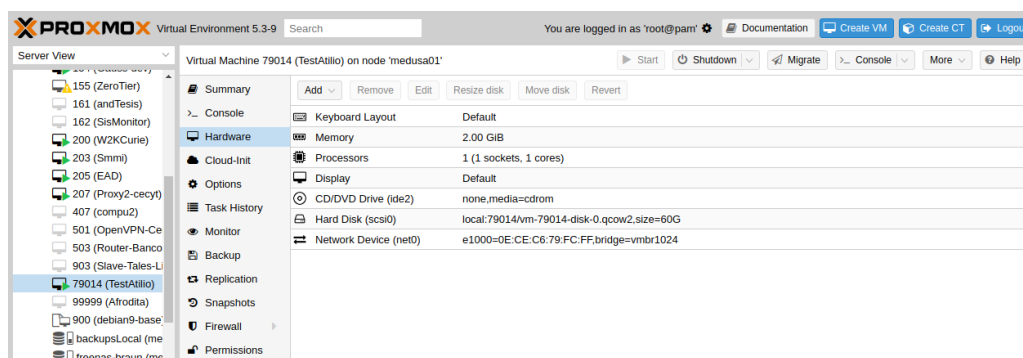


Figura 26: Detalle de componentes de la VM donde corre Docker

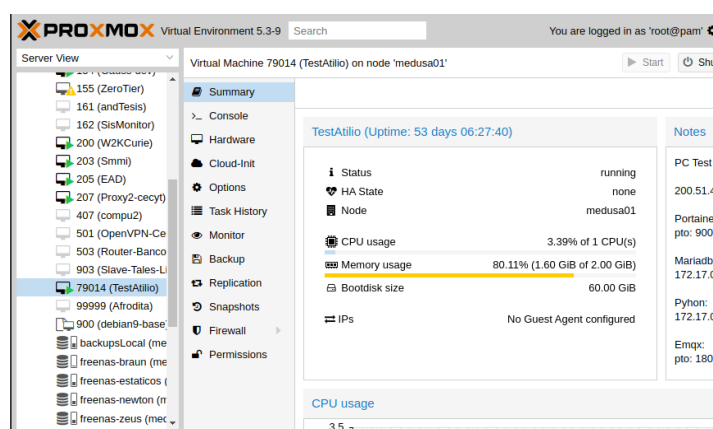


Figura 27: Utilización de recursos de la VM donde corre docker

Una vez creada la VM, se instaló un sistema Operativo GNU Linux Debian donde corren los servicios.

Despliegue de Docker y Containers.

Como anteriormente se detalló, este proyecto posee una VM disponible desde Internet. Dentro de la misma, se desplegará la infraestructura en Containers Docker.

Para ello, dentro de la VM creada simplemente deployremos Docker e Instalaremos Portainer a fin de obtener un entorno gráfico y amigable para la administración de contenedores.

Instalación de Docker.

```
#apt install docker.io  
#systemctl enable docker
```

Instalación de Portainer.

```
#docker pull portainer/portainer-ce  
#docker run --restart=always --name=portainer -d -p 9000:9000 -v  
/var/run/docker.sock:/var/run/docker.sock portainer/portainer-ce
```

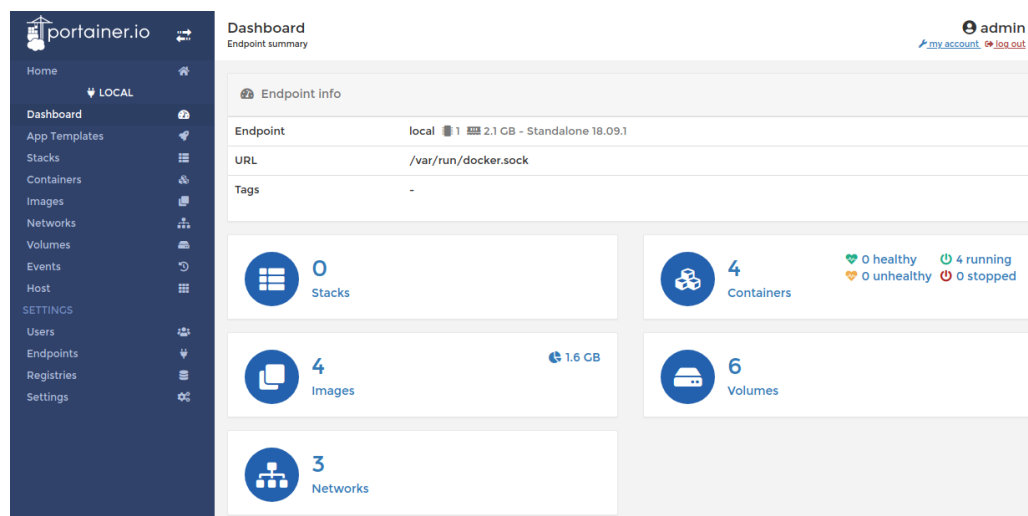


Figura 28: Panel de control de Portainer

Deploy de Infraestructura.

En este proyecto, se utilizará la siguiente Infraestructura:

- Broker MQTT - emqx
- Base de Datos - mariadb
- Cliente MQTT de Escritura de Base de Datos - Python
- Web de Visualización - Layout

Para ello, se analizará el marco teórico de cada tecnología para comprender su funcionamiento.

Despliegue de EMQ para el proyecto.

Como Broker MQTT seleccionado, EMQ posee una versión oficial que permite su utilización en docker. La misma se encuentra en [hub.docker.io](https://hub.docker.io/emqx/emqx) bajo el proyecto: `emqx/emqx`.

Despliegue `emqx/emqx4.2.3`.

Name

Image configuration

Registry

Image

[Advanced mode](#)

Always pull the image ☒

Network ports configuration

Publish all exposed network ports to random host ports ☐

Manual network port publishing [publish a new network port](#)

host	1883	→	container	1883	TCP	UDP	✕
host	18083	→	container	18083	TCP	UDP	✕

Figura 29: Contenedor de EMQ x y los puertos donde expone el servicio

Despliegue de Maria DB para el proyecto.

En este apartado se dará detalle del despliegue de MaríaDB en Docker y su configuración.

Despliegue de Contenedor MaríaDB.

Name mariadb

Image configuration

Registry DockerHub

Image docker.io mariadb:10.5.8-focal

[Advanced mode](#)

Always pull the image ☒

Network ports configuration

Publish all exposed network ports to random host ports ☒

Manual network port publishing [publish a new network port](#)

host	container	Protocol
3306	3306	TCP

Figura 30: Contenedor de María DB y los puertos donde expone el servicio

En este punto, es importante, además de desplegar el contenedor, pasar algunas variables necesarias para su funcionamiento:

Advanced container settings

Env

Environment variables [add environment variable](#)

name	value
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin/
GOSU_VERSION	1.12
PGP_KEYS	177F4010FE56CA3336300305F1656F24C74CD1D8
MARIADB_MAJOR	10.5
MARIADB_VERSION	1:10.5.8+maria~focal
MYSQL_ROOT_PASSWORD	CONTRASEÑA
MYSQL_DATABASE	NOMBRE DE LA BASE DE DATOS
MYSQL_USER	USUARIO

Figura 31: Parámetros de configuración del contenedor

Construcción de la Base de Datos.

Este proyecto en particular, se desarrolló con una base de datos que posee dos tablas:

- google_maps_php_mysql
- mqttm

La primera de ellas, contiene los datos de geolocalización por nodo a fin de visualizar en modo gráfico la distribución de los mismos en el territorio, más adelante se dará detalle de ello.

La segunda tabla, se creó para almacenar los parámetros sensados, por lo que contiene las siguientes columnas:

- id: Identificador único de fila
- nodo: Identificador de nodo que envía la información
- temperatura: medición de temperatura en un momento dado
- humedad: medición de humedad en un momento dado
- monóxido: medición de monóxido de carbono en un momento dado
- fecha: fecha y hora en la que se realizó la medición

Para la creación de la estructura de la base de datos, se puede utilizar un archivo con extensión .sql que contenga un dump para la creación de la siguiente estructura:

```
-- MariaDB dump 10.18 Distrib 10.5.8-MariaDB, for debian-linux-gnu (x86_64)
--
-- Host: localhost      Database: mqttdb
--
-- Server version      10.5.8-MariaDB-1:10.5.8+maria~focal

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101 SET
```

```

@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `google_maps_php_mysql`
--

DROP TABLE IF EXISTS `google_maps_php_mysql`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `google_maps_php_mysql` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) CHARACTER SET latin1 NOT NULL,
  `direccion` varchar(100) CHARACTER SET latin1 NOT NULL,
  `lat` float NOT NULL,
  `lng` float NOT NULL,
  `pais` varchar(100) CHARACTER SET latin1 NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=7 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `mqttm`

```

```
--

DROP TABLE IF EXISTS `mqttm`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `mqttm` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nodo` varchar(20) DEFAULT NULL,
  `temperatura` varchar(20) DEFAULT NULL,
  `humedad` varchar(20) DEFAULT NULL,
  `monoxido` varchar(20) DEFAULT NULL,
  `fecha` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2911 DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Con ello, todo lo necesario para la utilización de la base de datos ya estaría desplegado. Se podría para futuras implementaciones o proyectos diferentes, agregar campos o bien tablas, como por ejemplo, una tabla: nodos que permitiría tener información sobre los mismos y una identificación más precisa.

Despliegue de Código Python para el proyecto.

Con lo visto hasta aquí, a partir de una WSN Mesh, se está capturando datos que son enviados a un Gateway que publica mensajes utilizando el protocolo MQTT hacia un Broker MQTT.

Si dichos datos desean ser utilizados por alguna Aplicación, o bien se desea su almacenamiento para un historial, es necesaria la recolección de los mismos por un cliente MQTT y su inserción en la Base de Datos.

Despliegue de Python en contenedor.

Name: Python

Image configuration

Registry: DockerHub

Image: docker.io python:3.9.1-buster

Advanced mode

Always pull the image: ☒

Network ports configuration

Publish all exposed network ports to random host ports: ☒

Manual network port publishing: [publish a new network port](#)

host	container	protocol
22222	22222	TCP UDP

Figura 32: Contenedor de Python y los puertos donde expone el servicio

Código a ejecutar.

En el código que se presenta a continuación, se explica su funcionamiento.

```
# python3.9

import random

from paho.mqtt import client as mqtt_client

#db
import mariadb
#Abrir conexión
try:
    conn = mariadb.connect(
        user="USUARIO",
        password="CONTRASEÑA",
```

```

        host="172.17.0.5",
        port=3306,
        database="mqttdb"
    )

except mariadb.Error as e:
    print(f"Error connecting to MariaDB Platform: {e}")
    sys.exit(1)

cursor = conn.cursor()

#BROKER
broker = 'IP O DIRECCION DEL BROKER'
port = 1883
topic1 = "testtopic"
# generate client ID with pub prefix randomly

client_id = f'python-mqtt-{random.randint(0, 100)}'

def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port, 15)
    return client

def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):

```

```

print(f"Received `{msg.payload.decode()}` from `{msg.topic}` topic")
ttop = msg.topic
#ttop = "hola"
tmse = msg.payload.decode()
#tmse = "mundo"
separado = tmse.split(',')
senodo = separado[0]
setemp = separado[1]
sehume = separado[2]
semono = separado[3]

sql = "INSERT INTO `mqtttdb`.`mqttm` (`id`, `nodo`, `temperatura`, `humedad`,
`monoxido`, `fecha`) VALUES (NULL, '"+senodo+"', '"+setemp+"', '"+sehume+"',
 '"+semono+"',CURRENT_TIMESTAMP);"

try:
    cursor.execute(sql)
    conn.commit()
    print("Guardando datos...OK")
except:
    conn.rollback()
    print("Falla en guardar datos")

client.subscribe(topic1)
# client.subscribe(topic2)
client.on_message = on_message

def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

```

```
if __name__ == '__main__':  
    run()
```

Configuración de Apache.

Una vez desplegado, como consideración especial, deberá activarse Apache. Para ello, definido el lugar de almacenamiento del sitio, debe habilitarse el sitio dentro de la carpeta /etc/apache2/sites-available y copiar la configuración en /etc/apache2/sites-enable. Realizado esto, se debe reiniciar el demonio a fin de que el sitio sea publicado.

Particularmente para el presente trabajo se utilizó para el almacenamiento del sitio la carpeta por defecto: /var/www/html/.

Código del Layout.

A continuación, se presenta el código incluido en el archivo: index.php. Dentro del mismo, se encuentra la explicación de su funcionamiento.

```
for($i=0;$i<count($rawdata);$i++){  
    $time = $rawdata[$i]["fecha"];  
    $date = new DateTime($time);  
    $rawdata[$i]["fecha"]=$date->getTimestamp()*1000;  
    // $rawdata["fecha"]=$date->getTimestamp()*1000;  
}  
  
?>  
  
<HTML>  
<BODY>  
  
<meta charset="utf-8">  
  
<!-- Latest compiled and minified JavaScript -->
```

```

</script src="https://code.jquery.com/jquery.js"></script>

    <!-- Importo el archivo Javascript de Highcharts directamente desde su servidor -->
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script <?php
//Conexión a DB
function conectarBD(){
    $server = "IP O NOMBRE DEL SERVIDOR DE DB";
    $usuario = "USUARIO";
    $pass = "CONTRASEÑA";
    $BD = "BASE DE DATOS";
    //variable que guarda la conexión de la base de datos
    $conexion = mysqli_connect($server, $usuario, $pass, $BD);
    //Comprobamos si la conexión ha tenido éxito
    if(!$conexion){
        echo 'Ha sucedido un error inesperado en la conexión de la base de
datos<br>';
    }
    //devolvemos el objeto de conexión para usarlo en las consultas
    return $conexion;
}

/*Desconectar la conexión a la base de datos*/
function desconectarBD($conexion){
    //Cierra la conexión y guarda el estado de la operación en una variable
    $close = mysqli_close($conexion);
    //Comprobamos si se ha cerrado la conexión correctamente
    if(!$close){
        echo 'Ha sucedido un error inesperado en la desconexión de la base de
datos<br>';
    }
    //devuelve el estado del cierre de conexión
    return $close;
}

//Devuelve un array multidimensional con el resultado de la consulta
function getArraySQL($sql){

```

```

//Creamos la conexión
$conexion = conectarBD();
//generamos la consulta
//if(!$result = mysqli_query($conexion, $sql)) die(mysqli_error($conexion));
$result = mysqli_query($conexion, $sql);
#if(!$result = mysqli_query($conexion, $sql)) die();

$rawdata = array();
//guardamos en un array multidimensional todos los datos de la consulta

$i=0;

while($row = mysqli_fetch_array($result))
{
    //guardamos en rawdata todos los vectores/filas que nos devuelve la consulta
    $rawdata[$i] = $row;
    $i++;
}

//    $rawdata = mysqli_fetch_array($result);

#var_dump($rawdata);
//Cerramos la base de datos
desconectarBD($conexion);
//devolvemos rawdata
//    var_dump($rawdata);
    return $rawdata;
}

$nodo = "";
if (isset($_POST)) {

if (isset($_POST['nodo'])) {
    $nodo = $_POST['nodo'];

```

```

    }
    }

    //Sentencia SQL
    //$sql = "SELECT energy,water,temperature,time from tabla;";

    $sql = "SELECT temperatura,humedad,monoxido,fecha from mqttdb.mqttm where
    nodo = '$nodo' order by fecha desc limit 100;";

    //Array Multidimensional

    $rawdata = getArraySQL($sql);

    //Adaptar el tiempo
    //
    //var_dump($rawdata);

    src="http://code.highcharts.com/modules/exporting.js"></script>

    <form name = 'fsarabia' method = 'POST' action = ">
    <script type="text/javascript">
    <!--
        var wait=setTimeout("document.fsarabia.submit();",5000);
    //-->
    </script>
    <select name = 'nodo'>
    <option value = ">seleccionar</option>
    <option value = '01' <?php if($_POST['nodo']=='01'){echo ' selected';}?>>01</option>
    <option value = '02' <?php if($_POST['nodo']=='02'){echo ' selected';}?>>02</option>
    <option value = '03' <?php if($_POST['nodo']=='03'){echo ' selected';}?>>03</option>

    </select>
    <input type = 'submit' value = 'seleccionar nodo:'>
    </form>

```

```
<div id="container">
```

```
</div>
```

```
<script type='text/javascript'>
```

```
$(function () {
```

```
    $(document).ready(function() {
```

```
        Highcharts.setOptions({
```

```
            global: {
```

```
                useUTC: false
```

```
            }
```

```
        });
```

```
var chart;
```

```
$('#container').highcharts({
```

```
    chart: {
```

```
        type: 'spline',
```

```
        animation: Highcharts.svg, // don't animate in old IE
```

```
        marginRight: 10,
```

```
        events: {
```

```
            load: function() {
```

```
            }
```

```
        }
```

```
    },
```

```
    title: {
```

```
        text: 'Medicion de Valores del nodo <?php echo $nodo; ?>',
```

```
    },
```

```
    xAxis: {
```

```
        type: 'datetime',
```

```
        tickPixelInterval: 150
```

```
    },
```

```
    yAxis: {
```

```
        title: {
```

```

        text: 'Value'
    },
    plotLines: [{
        value: 0,
        width: 1,
        color: '#808080'
    }]
},
tooltip: {
    formatter: function() {
        return '<b>'+ this.series.name +'</b><br/>'+
            Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) +'<br/>'+
            Highcharts.numberFormat(this.y, 2);
    }
},
legend: {
    enabled: true
},
exporting: {
    enabled: true
},
series: [{
    name: 'Temperatura',
    data: (function() {
        var data = [];
        <?php
            for($i = 0 ;$i<count($rawdata);$i++){
                ?>
                    data.push([<?php echo $rawdata[$i]["fecha"];?>,<?php echo
$rawdata[$i]["temperatura"];?>]);
                <?php } ?>
            return data;
        })()
    },{
        name: 'Humedad',

```

```

        data: (function() {
            var data = [];
            <?php
                for($i = 0 ;$i<count($rawdata);$i++){
                    ?>
                        data.push([<?php echo $rawdata[$i]["fecha"];?>,<?php echo
$rawdata[$i]["humedad"];?>]);
                    <?php } ?>
                return data;
            })()
        },{
            name: 'Monoxido',
            data: (function() {
                var data = [];
                <?php
                    for($i = 0 ;$i<count($rawdata);$i++){
                        ?>
                            data.push([<?php echo $rawdata[$i]["fecha"];?>,<?php echo
$rawdata[$i]["monoxido"];?>]);
                        <?php } ?>
                    return data;
                })()
            }]
        });
    });

</script>
</html>

```

Código de visión en Mapa.

A fin de visualizar los dispositivos en el mapa, como anteriormente se mencionó, se utilizó una API de GCP. al código anterior, se le agregó una redirección a /map a fin de visualizar el resultado en una página diferente y mantener ordenados los datos.

```
<a href="http://200.51.41.149:8081/map" target="_blank">VER DISTRIBUCION EN MAPA</a>
```

Dentro de la carpeta map, se encuentra el index.php con el siguiente código explicado:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <meta name="description" content="">
    <meta name="author" content="">

    <meta name="theme-color" content="#000000" />

    <title>Cargar una ubicación desde la Base de Datos MySQL con PHP y Google Maps Javascript API </title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvORxT2MZw1T" crossorigin="anonymous">
```

```

<style type="text/css">
#mapa {
height: 50vh;
}
.h2s {
font-size: 3vh;
}
</style>

</head>

<body>

    <main role="main">

        <div class="container text-center mt-5">

            <div class="row">

                <div class="col-md-12">

                    <h1 class="mb-4">DISTRIBUCION DE NODOS WSN </h1>

                    <!-- Contenedor del Mapa de Google -->
                    <div id="mapa"></div>

                </div>

            </div>

            <div class="row mt-3">

                <div class="col-md-12">

                    <h2 class="h2s">Distribucion de Nodos por Sector</h2>

```

```

<!-- Archivo PHP con la lógica para mostrar una tabla con las ubicaciones -->
<?php include('php/app.php'); ?>

</div>

</div>

<hr>

</div>

</main>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abt
TE1Pi6jizo" crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4
sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIlly6OrQ6VrjIEaEfF/nJGzIxFDsf4x0xIM+
B07jRM" crossorigin="anonymous"></script>

<script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AquiVaLaKeyQueNosBrindòGCP&
callback=initMap"></script>

<script type="text/javascript">
function initMap() {
var map;
var bounds = new google.maps.LatLngBounds();

```

```

var mapOptions = {
  mapTypeId: 'roadmap'
};

map = new google.maps.Map(document.getElementById('mapa'), {
  mapOptions
});

map.setTilt(50);

// Crear múltiples marcadores desde la Base de Datos
var marcadores = [
  <?php include('php/marcadores.php'); ?>
];

// Creamos la ventana de información para cada Marcador
var ventanaInfo = [
  <?php include('php/info_marcadores.php'); ?>
];

// Creamos la ventana de información con los marcadores
var mostrarMarcadores = new google.maps.InfoWindow(),
  marcadores, i;

// Colocamos los marcadores en el Mapa de Google
for (i = 0; i < marcadores.length; i++) {
  var position = new google.maps.LatLng(marcadores[i][1], marcadores[i][2]);
  bounds.extend(position);
  marker = new google.maps.Marker({
    position: position,
    map: map,
    title: marcadores[i][0]
  });

  // Colocamos la ventana de información a cada Marcador del Mapa de Google

```

```

        google.maps.event.addListener(marker, 'click', (function(marker, i) {
            return function() {
                mostrarMarcadores.setContent(ventanaInfo[i][0]);
                mostrarMarcadores.open(map, marker);
            }
        })(marker, i));

        // Centramos el Mapa de Google para que todos los marcadores se puedan
ver
        map.fitBounds(bounds);
    }

    // Aplicamos el evento 'bounds_changed' que detecta cambios en la ventana
del Mapa de Google, también le configuramos un zoom de 14
    var    boundsListener    =    google.maps.event.addListener((map),
'bounds_changed', function(event) {
        this.setZoom(14);
        google.maps.event.removeListener(boundsListener);
    });

}

// Lanzamos la función 'initMap' para que muestre el Mapa con Los
Marcadores y toda la configuración realizada
    google.maps.event.addDomListener(window, 'load', initMap);
</script>

</body>
<br>
<br>
    <footer class="text-muted mt-3 mb-3">
    <div align="center">
    <a href="http://200.51.41.149:8081" target="_blank">VER DATOS DE
NODOS</a>
    </div>

```

```
</footer>
</html>
```

Como se aprecia en el código, index, llama funciones adicionales que se encuentran en archivos contenidos en la carpeta llamada php. Los mismos son:

App.php.

Este archivo, generará la tabla de dispositivos a partir de los registrados en la base de datos y mostrará sus coordenadas.

```
?php

// Archivo de Conexión a la Base de Datos
include('conexion.php');

// Listamos las direcciones con todos sus datos (lat, lng, dirección, etc.)
$result = mysqli_query($con, "SELECT * FROM google_maps_php_mysql");

// Creamos una tabla para listar los datos
echo "<div class='table-responsive'>";

echo "<table class='table'>
    <thead class='thead-dark'>
    <tr>
        <th scope='col'>Nombre</th>
        <th scope='col'>Dirección</th>
        <th scope='col'>Latitud</th>
        <th scope='col'>Longitud</th>
        <th scope='col'>País</th>
    </tr>
    </thead>
```

```

        <tbody>";

while ($row = mysqli_fetch_array($result)) {
    echo "<tr>";
    echo "<td scope='col'>" . $row['nombre'] . "</td>";
    echo " <td scope='col'>" . preg_replace('/\\\\u([da-fA-F]{4})/', '&#x1;',
$row['direccion']) . "</td>";
    echo "<td scope='col'>" . $row['lat'] . "</td>";
    echo "<td scope='col'>" . $row['lng'] . "</td>";
    echo "<td scope='col'>" . $row['pais'] . "</td>";
    echo "</tr>";
}
echo "</tbody></table>";
echo "</div>";

mysqli_close($con);

?>

```

Conexion.php.

En este archivo se encuentran los datos de conexión a la tabla específica de ubicación de Base de Datos.

```

<?php

// Nos conectamos a la Base de Datos MySQL
$con = mysqli_connect("200.51.41.149", "consulta", "consulta", "mqttodb");

// Verificamos la conexión a la Base de Datos MySQL
if (mysqli_connect_errno()) {
    echo "Error al Conectar a la base de Datos: " . mysqli_connect_error();
}

```

```
?>
```

Info_Marcadores.php.

Este archivo contiene la información que será desplegada al pasar el mouse por encima de cada punto.

```
<?php

// Agregamos el nombre y dirección a la ventana de información de cada marcador,
puedes agregar la información que desees, nosotros agregaremos 'nombre' y
'direccion'

if($result->num_rows > 0){

    while($row = $result->fetch_assoc()){ ?>

        ['<div class="info_content">' + '<h3><?php echo $row['nombre']; ?></h3>' +
        '<p><?php echo $row['direccion']; ?></p>' + '</div>'],

        <?php }

    }

?>
```

Marcadores.php.

A partir de este archivo, se generan los marcadores de google maps.

```
<?php

// Archivo de Conexión a la Base de Datos
```

```

include('conexion.php');

// Listamos las direcciones con todos sus datos (lat, lng, dirección, etc.)
$result = mysqli_query($con, "SELECT * FROM google_maps_php_mysql");

// Seleccionamos los datos para crear los marcadores en el Mapa de Google serian
direccion, lat y lng
while ($row = mysqli_fetch_array($result)) {
    echo '[' . $row['nombre'] . ', ' . $row['direccion'] . ', ' . $row['lat'] . ', ' . $row['lng']
    . '],';
}
?>

```

Interacción con Sistemas de predicción

Una WSN, al proporcionar datos en tiempo real y una cobertura más amplia del área de interés, puede mejorar sistemas de predicción de incendios forestales ya que pueden desplegarse en un área afectada y recopilar datos sobre sus condiciones ambientales, como la temperatura, la humedad, la velocidad del viento, la dirección del viento, la concentración de gases peligrosos, entre otros.

Estos datos se pueden utilizar para alimentar modelos de predicción de incendios para mejorar la precisión y la eficacia de estos modelos.

En este sentido, al almacenar los valores recogidos por la WSN planteada en una base de datos, modelos de predicción de comportamiento de incendios forestales, podrían acceder a los mismos a fin de utilizar dichas variables para su mejorar su cálculo.

En el laboratorio montado para el presente trabajo, se utilizó una cantidad limitada de sensores. Si bien se podría crecer en este sentido

aumentando la cantidad de sensores por nodo, sería necesaria una interfaz para la conversión de los valores registrados en la base de datos a los valores necesarios de entrada a sistemas de predicción.

Esto se debe a que agregar un sensor a cada elemento combustible sería una tarea muy difícil e impráctica debido al gran número de elementos combustibles presentes en un área forestal. En su lugar, se podrían utilizar sensores distribuidos estratégicamente en diferentes puntos de la zona forestal, capaces de proporcionar una imagen global de las condiciones ambientales y de los materiales combustibles presentes. Además, existen técnicas de predicción que no requieren medición directa de todas las variables, sino que utilizan modelos matemáticos y estadísticos que pueden predecir la temperatura del material combustible en función de otras variables ambientales medidas por los sensores, como la humedad y la temperatura del aire, la velocidad del viento y la cantidad de lluvia, entre otros.

En función de la cantidad de métodos de predicción existentes y la diferencia en la obtención de datos, como prueba de laboratorio, planteó el modo en que la WSN elaborada en el presente proyecto, podría integrarse a la ecuación de Rothermel.

La misma es ampliamente utilizada para modelar la propagación del fuego en un combustible forestal. Fue desarrollada por el científico Mark Rothermel en 1972 y se basa en la idea de que la propagación del fuego depende de varios factores, incluyendo la densidad del combustible, la humedad, la velocidad del viento y la topografía del terreno.

La ecuación de Rothermel utiliza estos factores para calcular la tasa de propagación del fuego y la cantidad de combustible que se quema en un

determinado período de tiempo. La ecuación se expresa matemáticamente como:

$$R = \frac{I_R \xi + (1 + \phi_W + \phi_S)}{\rho_{be} \varepsilon Q_{ig}}$$

$$R = \frac{a^{\frac{1}{2}} U^{m+1}}{b}$$

Donde:

R es la tasa de propagación del fuego

I_R es la intensidad de reacción

ξ es el radio del flux de propagación

ϕ_W es el coeficiente de viento,

ϕ_S es un factor de la pendiente,

ρ_{be} es la densidad efectiva,

ε es número efectivo de calentamiento

Q_{ig} es el calor de pre-ignición.

El modelo de Rothermel, entre muchos parámetros de los que depende, el más importante es el combustible que se presenta en la zona forestal. Teniendo en cuenta lo anterior, Rothermel plantea modelos de combustión que representan terrenos típicos que se pueden presentar en un área forestal, los cuales tienen descripciones completas de los parámetros importantes para aplicar el modelo matemático de propagación de incendios adecuadamente. Esto permite escoger el modelo que mejor se aplique al terreno y a su clima correspondiente.

La ecuación de Rothermel se ha utilizado ampliamente en la gestión de incendios forestales para predecir la propagación del fuego y ayudar a tomar decisiones sobre cómo combatirlo.

Como se observa en la misma, una de las variables a obtener, sea cual fuere el caso, es la dirección del viento en función de su coeficiente.

En el laboratorio planteado en la presente, no se previó la utilización de sensores de viento, asimismo, para la creación de dicha interfaz se plantea la posibilidad de utilizar la información recopilada por la WSN para estimar la dirección del viento en una zona de incendio forestal. Si se conoce la ubicación de los nodos de la red y se mide el monóxido de carbono en diferentes momentos en cada nodo, sería posible estimar la velocidad y la dirección del viento que transporta el humo y el gas generado por el incendio.

Para hacer esto, se puede utilizar un algoritmo de triangulación o multilateración que calcule la posición del foco del incendio a partir de las mediciones de monóxido de carbono en diferentes nodos. Una vez que se conoce la posición del foco, se puede determinar la dirección del viento que transporta el humo y el gas utilizando modelos meteorológicos y datos de sensores de viento cercanos.

Es importante tener en cuenta que la estimación de la dirección del viento a partir de mediciones de monóxido de carbono es una técnica que puede ser afectada por factores como la topografía del terreno y la altura de los sensores, por lo que se debería combinar con otras técnicas de estimación para obtener resultados más precisos.

A pesar de ello, se concluyó que es posible la integración de los datos obtenidos por la WSN a modelos de predicción de comportamiento de incendios forestales mediante la utilización de interfaces de conexión y conversión de valores.

Capítulo IV: Resultados

Para realizar las pruebas, se programó y desplegó la WSN descrita con anterioridad tomando datos de sensores de temperatura, humedad, monóxido de carbono y registrando su ubicación (coordenadas) en una tabla anexa a la DB a fin de poder visualizar el despliegue en un mapa dinámico utilizando los servicios de Google Cloud Platform.

Si accedemos al puerto serial del gateway de la WSN podremos observar cómo se conectarán los nodos más cercanos y comenzarán a enviar su información y la de los nodos que estén conectados en la mesh. Además, se puede observar su la conexión a al AP de la red WiFi que provee conexión a internet al Gateway a fin de enviar la información recibida hacia el broker MQTT.

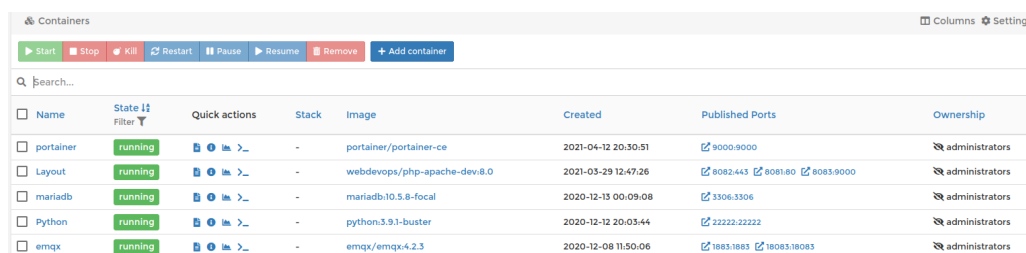
```

/dev/ttyUSB0

CONNECTION: newConnectionTask(): adding 2785553720 now= 4470924
bridge: Received from 2785553720 msg=03,28.00,17.00,0.66
bridge: Received from 2785553720 msg=03,28.00,17.00,0.65
bridge: Received from 2785553720 msg=03,28.00,17.00,0.64
bridge: Received from 2785553720 msg=03,29.00,16.00,0.63
bridge: Received from 2785553720 msg=03,28.00,30.00,0.62
bridge: Received from 2785553720 msg=03,28.00,30.00,0.62
bridge: Received from 2785553720 msg=03,28.00,17.00,0.61
CONNECTION: stationScan(): CRISTINA
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete():-- > Cleared old APs.
CONNECTION: scanComplete(): num = 4
CONNECTION:      found : CRISTINA, -66dBm
CONNECTION:      Found 1 nodes
CONNECTION: connectToAP(): Best AP is 922244932<---
CONNECTION: connectToAP(): Trying to connect, scan rate set to 4*normal
bridge: Received from 2785553720 msg=03,29.00,16.00,0.61
CONNECTION: Event: Station Mode Connected
bridge: Received from 2785553720 msg=03,28.00,28.00,0.60
CONNECTION: Event: Station Mode Got IP (IP: 192.168.100.15  Mask: 255.255.255.0  Gateway: 192.168.100.1)
My IP is 192.168.100.15
bridge: Received from 2785553720 msg=03,28.00,30.00,0.60
bridge: Received from 2785553720 msg=03,28.00,35.00,0.60
```

Figura 33: Conexión de la WSN al Gateway y del Gateway a Internet

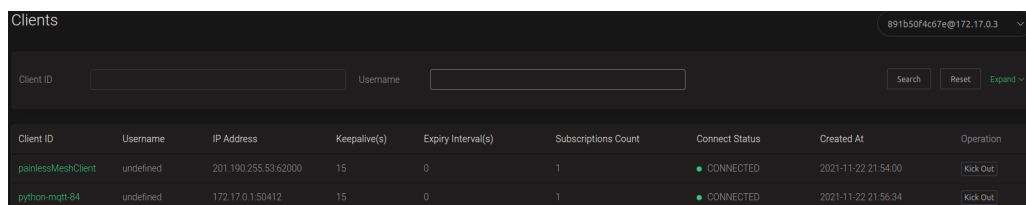
Una vez inicializada la WSN, mediante Portainer, se puede observar de modo gráfico que cada contenedor con los elementos de infraestructura se encuentren disponibles.



Name	State	Quick actions	Stack	Image	Created	Published Ports	Ownership
portainer	running	[Icons]	-	portainer/portainer-ce	2021-04-12 20:30:51	9000:9000	administrators
Layout	running	[Icons]	-	webdevops/php-apache-dev:8.0	2021-03-29 12:47:26	8082:443 8081:80 8083:9000	administrators
mariaadb	running	[Icons]	-	mariaadb:10.5.8-focal	2020-12-13 00:09:08	3306:3306	administrators
Python	running	[Icons]	-	python:3.9.1-buster	2020-12-12 20:03:44	22222:22222	administrators
emqx	running	[Icons]	-	emqx/emqx:4.2.3	2020-12-08 11:50:06	1883:1883 18083:18083	administrators

Figura 34: Visualización de contenedores activos desde Portainer

Verificada la disponibilidad, se puede observar mediante el broker MQTT que se encuentre conectada tanto la WSN para transmitir datos como el contenedor que posee el código que leerá estos datos y los escribirá en la Base de datos.



Client ID	Username	IP Address	Keepalive(s)	Expiry Interval(s)	Subscriptions Count	Connect Status	Created At	Operation
painlessMeshClient	undefined	201.190.255.53:62000	15	0	1	CONNECTED	2021-11-22 21:54:00	[Kick Out]
python-mqtt-84	undefined	172.17.0.1:50412	15	0	1	CONNECTED	2021-11-22 21:56:34	[Kick Out]

Figura 35: Control de Clientes sobre el dashboard EMQ x

Además, desde el broker se podrá observar la cantidad de mensajes recibidos desde la WSN y adquiridos desde el contenedor que posee el código que los incorpora a la DB.

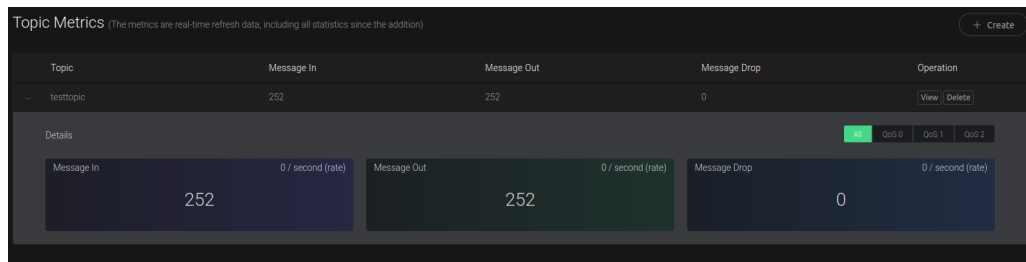


Figura 36: Cantidad de mensajes MQTT recibidos y enviados por el Broker

Como paso posterior, se puede observar dentro del contenedor que posee el código de escritura de mensajes en la DB, el log de dicha acción:

```
Guardando datos...OK
Received `03,29.00,35.00,0.66` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,17.00,0.65` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,18.00,0.65` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,18.00,0.65` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,17.00,0.66` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,37.00,0.65` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,23.00,0.61` from `testtopic` topic
Guardando datos...OK
Received `03,29.00,19.00,0.61` from `testtopic` topic
Guardando datos...OK
```

Figura 37: Logs del contenedor de Python respecto a la recepción de mensajes y almacenamiento de los mismos en la base de datos.

Cotejado esto, se podría observar si los datos fueron correctamente escritos en la base de datos realizando una consulta en la misma.

```

1 • SELECT * FROM mqtttdb.mqttm
2   order by fecha desc

```

Result Grid						
		Filter Rows:		Edit:		
#	id	nodo	temperatura	humedad	monoxido	fecha
1	2770	03	29.00	17.00	0.59	2021-11-22 22:06:00
2	2769	03	29.00	19.00	0.59	2021-11-22 22:05:58
3	2768	03	29.00	20.00	0.58	2021-11-22 22:05:56
4	2767	03	29.00	30.00	0.59	2021-11-22 22:05:54
5	2766	03	29.00	22.00	0.59	2021-11-22 22:05:52
6	2765	03	29.00	36.00	0.59	2021-11-22 22:05:50
7	2764	03	29.00	17.00	0.60	2021-11-22 22:05:48
8	2763	03	29.00	17.00	0.59	2021-11-22 22:05:46
9	2762	03	30.00	17.00	0.60	2021-11-22 22:05:44
10	2761	03	29.00	20.00	0.60	2021-11-22 22:05:42
11	2760	03	29.00	34.00	0.59	2021-11-22 22:05:40
12	2759	03	29.00	19.00	0.60	2021-11-22 22:05:38
13	2758	03	29.00	19.00	0.61	2021-11-22 22:05:36
14	2757	03	29.00	18.00	0.60	2021-11-22 22:05:34
15	2756	03	29.00	19.00	0.60	2021-11-22 22:05:32
16	2755	03	29.00	32.00	0.60	2021-11-22 22:05:30
17	2754	03	29.00	18.00	0.60	2021-11-22 22:05:28
18	2753	03	30.00	18.00	0.60	2021-11-22 22:05:26
19	2752	03	29.00	18.00	0.60	2021-11-22 22:05:24
20	2751	03	29.00	19.00	0.61	2021-11-22 22:05:21

Figura 38: Consulta a la base de datos sobre la tabla que almacena los datos registrados.

Finalmente, con los datos escritos en la base de datos, podríamos acceder a ellos mediante un usuario de consulta y a partir de allí, podríamos, o bien capturarlos mediante un web-service a fin de incorporarlos como entradas de sistemas de predicción de

comportamiento de incendios forestales o bien mostrar los datos mediante una página web que podría ser consultada por personal de lucha contra el fuego, o bien a modo estadístico.

En el presente trabajo, a modo de ejemplo, se presenta una web que muestra los datos descritos por nodo.

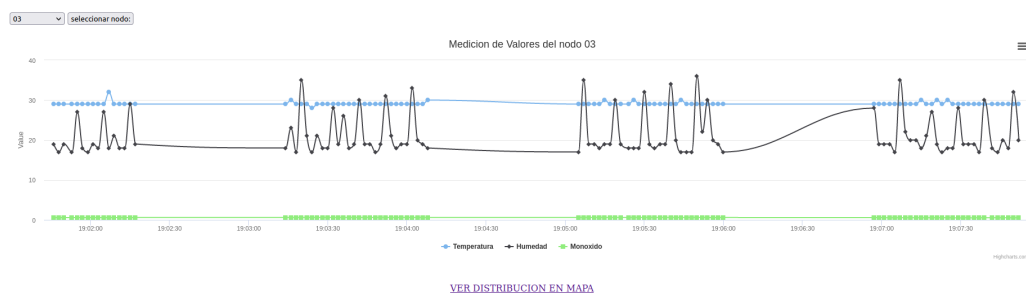


Figura 39: Layout. Valores de datos sensados por nodo

Además, se incluyó un link a fin de poder visualizar dentro de un mapa, la ubicación de cada nodo.



Distribucion de Nodos por Sector

Nombre	Dirección	Latitud	Longitud	País
Nodo 1	Villa Nueva - Glén	-32.9095	-68.7796	Argentina
Nodo 2	Villa Nueva - Glén	-32.9097	-68.7792	Argentina
Nodo 3	Villa Nueva - Glén	-32.9094	-68.78	Argentina
Gateway	Villa Nueva - Glén	-32.9097	-68.7795	Argentina

Figura 40: Layout. Distribución geográfica de los nodos de la WSN

De esta manera, la integración a modelos de predicción propuesta, se realizaría mediante el acceso de los mismos a los valores obtenidos y registrados en la base de datos.

Por lo tanto, tomando como ejemplo un modelo de predicción que utilice técnicas de evolución diferencial por islas, en la predicción del comportamiento de los incendios forestales, puede acceder a la información de las variables ambientales relevantes que influyen en la propagación del fuego, como la temperatura, la humedad, la velocidad y dirección del viento, la cantidad de combustible, entre otras.

Esta información permitiría calibrar y ajustar los parámetros del modelo de manera más precisa, mejorando la calidad de las predicciones y

reduciendo la incertidumbre en los resultados. Además, una WSN también permitiría una cobertura más amplia del área en estudio, lo que a su vez aumentaría la resolución espacial y temporal de los datos obtenidos y, por lo tanto, mejoraría la precisión de la predicción.

Conclusiones

El presente proyecto, pretende tener la flexibilidad necesaria que le permita implementarse tanto en equipos ubicados permanentemente en ubicaciones estratégicas como equipos desplegados sobre frentes de incendio o bien integrados a drones de monitoreo.

Además, por el uso del protocolo MQTT para el envío de mensajes con los datos recolectados, puede integrarse y formar parte de otras tecnologías y topologías de red. Un ejemplo de ello sería la formación de pequeñas células WSN distribuidas en los extremos de otra red de mayor alcance para la recolección de datos.

Respecto a la incorporación de datos a sistemas de predicción, el almacenamiento de datos en tiempo casi real (no se utiliza el término tiempo real en virtud del tiempo necesario entre la captura, distribución y escritura de los datos) permitiría que a partir de la información, se realicen conversiones de medidas y se inyecten los datos necesarios mediante un web service a cualquier sistema a fin de mejorar los tiempos de predicción de comportamientos.

En cuanto a la infraestructura, al utilizarse contenedores, sería posible utilizar orquestadores y balanceadores de carga ante eventuales necesidades de escalabilidad de los proyectos.

En cuanto a los objetivos planteados, en el apartado “Red Inalámbrica de Sensores” se alcanzó el primer objetivo:

- Estudiar los diferentes tipos de controladores y dispositivos WSN existentes.

Respecto a los siguientes puntos:

- Evaluar desarrollos e investigaciones de WSN utilizando el estándar IEEE 802.11.
- Implementar una WSN basada en el Estándar IEEE 802.11.
- Configurar y gestionar los dispositivos y sensores.
- Evaluar rendimiento de la red y capacidad de reacción ante fallos.
- Evaluar comportamiento e información obtenida por dispositivos de captura de datos a la red.

Como se observó a lo largo del trabajo, se armó un laboratorio con dispositivos que utilizan el estándar IEEE 802.11 y se realizaron test de comportamiento y reacción ante diferentes escenarios.

En cuanto a los puntos:

- Documentar características, arquitectura, ventajas y desventajas de la utilización del estándar IEEE 802.11 sobre WAN.
- Análisis de datos y compatibilidad con sistemas de predicción de comportamiento de incendios.

En el resumen del trabajo, se teorizó respecto a soluciones híbridas WAN y se mencionó cómo debería ser la interacción de la Base de Datos con Sistemas de Predicción. Asimismo, esto no pudo ser logrado prácticamente, por lo que se investigará en una línea de trabajo futuro.

Finalmente, la evaluación del alcance de la solución y su viabilidad fue satisfactoria y se concluye que el despliegue del presente proyecto, podría colaborar en gran medida en la generación de alertas de ocurrencia de incendios forestales así como en el planeamiento estratégico de la lucha contra el fuego, generación de estadísticas y facilitaría la incorporación de datos reales en sistemas de predicción de comportamiento de incendios forestales.

Bibliografía

- Argentina, J. d. g. d. m. (n.d.). *Plan Nacional de Manejo del Fuego*. Infoleg. Retrieved May 26, 2023, from http://www.infoleg.gob.ar/basehome/actos_gobierno/actosdegobier/actosdegobierno/16-3-2009-1.htm
- Argentina Gobierno, M. d. A. y. D. S. (Ed.). (2021). *Red de faros de conservación*. argentina.gob.ar. https://www.argentina.gob.ar/sites/default/files/faros_de_conservacion-descripcion_e_informacion_general.pdf
- Argentina Gobierno - Ministerio de Ambiente y Desarrollo Sostenible. (2021, 11 17). *Manejo del Fuego - Reporte de Incendios*.
- Aslan, Y. E., Korpeoglu, I., & Ulusoy, Ö. (2012, 04 04). A framework for use of wireless sensor networks in forest fire detection and monitoring. *Computers, Environment and Urban Systems*, 36, 614-625.
- Atempa Camacho, J. A., Díaz-Ramírez,, A., Ayala Figueroa, R. I., & Quintero Rosas, V. (2015). Red de sensores inalámbrica para la detección de incendios forestales. *Research in Computing Science*, 107, 73-85.
- Bahit, E. (n.d.). *Python para principiantes*. Uniwebsidad. Retrieved 10 23, 2020, from <https://uniwebsidad.com/libros/python>
- Burkart, R., Bárbaro, N. O., Sánchez, R. O., & Gómez, D. A. (1999).

Eco-regiones de la Argentina. Administración de Parques Provinciales. Secretaría de Recursos Naturales y Desarrollo Sustentable. Presidencia de la Nación Argentina.

Cai A, L., He B, H. S., Liang A, Y., Wu A, Z., & Huang, C. (2019). Analysis of the uncertainty of fuel model parameters in wildland fire modelling of a boreal forest in north-east China. *International Journal of Wildland Fire*, 28, 205–215.

<https://doi.org/10.1071/WF18083>

Comisión Nacional Forestal - México. (2010). *Incendios Forestales - Guía práctica para comunicadores* (3rd ed.). Comisión Nacional Forestal - México.

Dennis, R., Meijaard, E., Applegate, G., & Moore, P. (n.d.). *Diversidad biológica forestal*. DIVERSIDAD BIOLÓGICA FORESTAL - Unasylva 209. Retrieved May 26, 2023, from <https://www.fao.org/3/y3582s/y3582s08.htm>

Dentoni, M.C., Zacconi, R.G., & Marcuzzi, E.A. (2014, 01). *El Incendio de Baradero de 2008: un caso de comportamiento extremo del fuego* (Informe Técnico N° 9 ed.) [Evaluación de peligro de incendios Informes técnicos - ISSN 2313-9420]. Programa Nacional de Evaluación de Peligro de Incendios y Alerta Temprana - Plan Nacional de Manejo del Fuego - Secretaría de Ambiente y Desarrollo Sustentable - Argentina Gobierno.

Docker Documentation. (n.d.). Docker Documentation. Retrieved 06 20,

- 2019, from <https://docs.docker.com/>
- Docker run reference*. (n.d.). Docker Documentation. Retrieved 11 15, 2019, from <https://docs.docker.com/engine/reference/run/>
- Efectos del fuego en los ecosistemas forestales*. (2019, February 22). Asemafor. Retrieved May 26, 2023, from <https://www.asemafor.cl/efectos-del-fuego-en-los-ecosistemas-forestales/>
- Elgueta, R. A., & Méndez-Garabetti, M. (2017). Red de sensores inalámbrica aplicada a alertas y predicción del comportamiento de incendios forestales. *Encuentro del Centro Internacional de Ciencias de la Tierra*, 12, 111-119.
- Elgueta, R. A., & Méndez-Garabetti, M. (2017). Red Inalámbrica de Sensores Aplicada a la Predicción del Comportamiento de Incendios Forestales. *Workshop de Investigadores en Ciencias de la Computación*, XIX, 914-918.
- EMQX Documentation. (n.d.). EMQ X Broker | EMQ Docs. Retrieved 12 02, 2019, from <https://docs.emqx.io/en/broker/v4.3/>
- Garbarino, J. (2011, 11 7). *Protocolos para redes inalámbricas de sensores* [Tesis]. Facultad de Ingeniería - Universidad de Buenos Aires.
- Gislason, D. (2008). *Zigbee Wireless Networking*. Elsevier Science.
- Iacono, L., Godoy, P., Marianetti, O., García Garino, C., & Párraga, C. (2012, 07). *A Survey on the integration between WSN and TCP/IP*

networks (10th ed.) [ISSN 1510-7450 • ISSN (en línea) 1688-9584].

Memoria de Trabajos de Difusión Científica y Técnica.

K, J., P, E., A, C., & P, H. (2016, 01 05). Sistema de detección de incendios forestales mediante redes sensoriales inalámbricas (Zigbee). *MASKANA, I+D+ingeniería*, 5, 1-5.
<https://publicaciones.ucuenca.edu.ec/ojs/index.php/maskana/article/view/532>

Lajara, J. R., & Pelegrí Sebastián, J. (2014). *Sistemas integrados con Arduino*. Alfaomega.

MariaDB Knowledge Base. (n.d.). MariaDB Knowledge Base. Retrieved 03 03, 2021, from <https://mariadb.com/kb/es/>

Méndez-Garabetti, M., Bianchini, G., Caymes-Scutari, P., Elgueta, R., Caylá, I. B., Córdoba, D., Peñasco, A., Morelli, J., Carrizo, M., Varela, P., Giorlando, A., Luján, A., & Carballo, N. (2018). Predicción y alerta temprana de incendios forestales mediante integración de WSN e imágenes satelitales en un método de reducción de incertidumbre guiado por datos. *Workshop de Investigadores en Ciencias de la Computación, XX*, 179-183.

Méndez-Garabetti, M., Bianchini, G., Caymes-Scutari, P., & Tardivo, M. L. (2016, 03 09). Increase in the quality of the prediction of a computational wildfire behavior method through the improvement of the internal metaheuristic. *Fire Safety Journal*, 82, 49-62.

Minera, F. (2014). *PHP* (1st ed.). Red Users.

- Minera, F. (2014). *PHP + SQL* (1st ed.). Red Users.
- Python para Todos: Explorando la información con Python 3* (J. d. D. Dougnac & J. C. P. Castellanos, Trans.). (2020). Independently Published.
- Serra Davos, M., & Plana Bach, E. (2016). *Los incendios forestales: guía para comunicadores y periodistas* (Centre Tecnològic Forestal de Catalunya, Ed.). Centro Tecnológico Forestal de Cataluña.
- Soto Sanchez, O. A. (2011). *Comparación de la eficiencia volumétrica entre redes inalámbricas WiFi y WiMAX* [(Tesis de Licenciatura)]. UNAM. <https://repositorio.unam.mx/contenidos/3443390>
- Stallings, W. (2004). *Comunicaciones y redes de computadores* (J. E. Díaz Verdejo, Trans.). Pearson Educación.
- Stallings, W. (2014). *Data and Computer Communications*. Pearson.
- Tardivo, M. L., Caymes-Scutari, P., Bianchini, G., & Méndez-Garabetti, M. (2016, 11 08). Reducción de incertidumbre con evolución diferencial en la predicción de incendios forestales: sintonización y análisis de parámetros. *Mecánica Computacional, XXXIV*, 2903-2917.
- Torrente Artero, Ó. (2013). *Arduino: curso práctico de formación*. Alfaomega.
- Universidad Tecnológica Nacional -FRD- Instituto Linux. (n.d.). *Virtualización de redes Linux*. carreralinux.com.ar.
blog.carreralinux.com.ar

Yuan, M. (2017, May 12). *Getting to know MQTT*. IBM Developer.

Retrieved October 8, 2023, from

<https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>