ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Integración de endpoints para datos abiertos de internet de las cosas en red de ESPOL

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Nombre de la titulación
Ingeniero en Telemática

Presentado por:

Jandry Oswaldo Romero Arcentales

Franklin Jahir Parrales Quijije

GUAYAQUIL - ECUADOR

Año: 2022

DEDICATORIA

JANDRY OSWALDO ROMERO ARCENTALES

El presente proyecto lo dedico a mis padres, mi hermano y mi abuela que siempre estuvieron apoyándome en los momentos más difíciles de mi carrera universitaria. Su amor, esfuerzo y paciencia fueron incondicionales en toda mi vida.

FRANKLIN JAHIR PARRALES QUIJIJE

El presente proyecto lo dedico como símbolo de gratitud a mis padres: Martina Quijije y Franklin Parrales quienes con su amor, paciencia, esfuerzo y su apoyo incondicional durante toda mi vida me han permitido culminar mis estudios.

AGRADECIMIENTOS

JANDRY OSWALDO ROMERO ARCENTALES

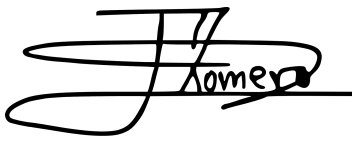
Agradezco a mi familia que siempre estuvo guiándome y apoyándome para lograr ser un profesional. A mis profesores, por toda su dedicación y horas compartiendo sus conocimientos. A mi tutor, que siempre estuvo para ayudarnos en algún requerimiento de nuestro proyecto y, finalmente, a mis amistades por compartir tiempo de estudio conmigo y brindarme su apoyo.

FRANKLIN JAHIR PARRALES QUIJIJE

Agradezco a mis padres, mi hermana, quienes fueron pilares fundamentales para conseguir este logro. A mis amistades, quienes me apoyaron y brindaron su ayuda durante todo este tiempo de estudio, y finalmente a mis docentes por sus conocimientos compartidos durante mi estadía en la carrera.

DECLARACIÓN EXPRESA

”Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Jandry Oswaldo Romero Arcentales y Franklin Jahir Parrales Quijije y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”



Jandry Romero Arcentales

Franklin Parrales Quijije

EVALUADORES

Firmado electrónicamente por:
WASHINGTON ADRIAN
VELASQUEZ VARGAS

Washington Velásquez PhD.

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
CHRISTOPHER JAVIER
VACCARO CEDILLO

Ing. Christopher Vaccaro

PROFESOR TUTOR

RESUMEN

La integración de dispositivos digitales a plataformas de *IoT* permite recopilar información medida en cualquier lugar que estos se encuentren para visualizarlas en portales abiertos y sea factible su análisis y procesamiento de los datos. Sin embargo, existe una gran polémica entre los usuarios cuando requieren acceder a información recopilada por dispositivos *IoT*, provocando que tengan escaso acceso a los datos medidos por parte de equipos comprados a fabricantes.

Para ello, en este proyecto se plantea integrar dispositivos digitales a la plataforma de registro de dispositivos *IoT* de ESPOL mediante el uso de hardware y software de código abierto para la recolección automática de los datos, consiguiendo que los usuarios puedan tener acceso libre a los datos medidos por parte de sus equipos.

El presente trabajo se encuentra dividido en 5 capítulos. El capítulo 1 comprende el origen de la problemática, los objetivos, alcance y limitaciones y el estado del arte con trabajos similares. El capítulo 2 presenta la metodología que se ha utilizado tanto en la arquitectura, dispositivos utilizados y protocolos de comunicación para dar cumplimiento a los objetivos del proyecto. El capítulo 3 describe la solución planteada, la cual consiste en el uso de una plataforma de registro de dispositivos *IoT* desarrollada por ingenieros de la carrera de Telemática. En el capítulo 4 se analizan los resultados obtenidos respecto a las pruebas realizadas para rendimiento y evaluación de satisfacción de los clientes. Finalmente, el capítulo 5 detalla las conclusiones, recomendaciones y trabajos futuros alrededor de la propuesta.

Palabras Clave: Estación meteorológica, sensores, accesibilidad a datos, endpoints, ngrok, IoT.

ABSTRACT

Integrating digital devices to IoT platforms allows the collection of information measured anywhere they are to view them in open portals, and its analysis and processing of the data is feasible. However, there is a great controversy among users when they require access to information collected by IoT devices, causing them to have little access to the data measured by Equipment purchased from manufacturers.

To do this, this project proposes to integrate digital devices to the platform of registration of ESPOL IoT devices by using code hardware and software open for automatic data collection, making it possible for users to have free access to the data measured by their equipment.

This work is divided into 5 chapters. Chapter 1 covers the origin of the problem, the objectives, scope, limitations, and state-of-the-art with similar jobs. Chapter 2 presents the methodology used in the architecture, devices, and communication protocols to comply with the project objectives. Chapter 3 describes the proposed solution, which uses an IoT device registration platform developed by engineers with Telematics degrees. Chapter 4 analyzes the results obtained regarding the tests carried out for performance and evaluation of customer satisfaction. Finally, chapter 5 details the conclusions, recommendations, and future work around the proposal.

Keywords: Metheorological station, sensors, data accessibility, endpoints, ngrok, IoT.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
IoT	Internet of Things
GEA	Granja Experimental Agrícola
CENAIM	Centro de Investigación Marina y Acuícola
SMS	Short Message Service
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
HTTP	Hyper Text Transport Protocol
IIoT	Industrial Internet of Things
IA	Inteligencia Artificial
LST	Laboratorio de Sistemas Telemáticos
FADCOM	Facultad de Arte, Diseño y Comunicación
JSON	JavaScript Object Notation
API	Application Programming Interfaces
UART	Universal Asynchronous Receiver/Transmitter
TCP	Transmission Control Protocol
FCV	Facultad de Ciencias de la Vida
LAN	Local Area Network

SIMBOLOGÍA

ms	milisegundos
kb/s	kilobytes por segundo
%	porcentaje

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ALGORITMOS

CAPÍTULO 1

1. INTRODUCCIÓN

El internet de las cosas (*Internet of Things* por sus siglas en inglés IoT) ha causado un gran cambio respecto a cómo las personas entendían la internet. Hace unos años, las personas del mundo consideraban que la internet era nada más que poder realizar búsquedas en páginas web e interactuar alrededor del mundo, sin embargo, ha evolucionado más allá a tal punto de permitir a las personas interactuar con dispositivos electrónicos que forman parte de la oficina, hogar, industrias, hospitales, entre otros [1].

El IoT ha cobrado un gran poder en los últimos años y es uno de los paradigmas más revolucionarios en la actualidad, en el que se asigna a cualquier dispositivo físico la capacidad de conectarse a una red; básicamente, es una nueva forma de entender la tecnología y su aplicación en la vida [2].

Este proyecto busca integrar endpoints de datos abiertos de internet de las cosas en la red de la Escuela Superior Politécnica del Litoral (ESPOL) mediante la programación y habilidades de networking, donde se pueda crear un procedimiento general de conexión de varios equipos digitales a una red y éstos se vean reflejados en una plataforma de registro de dispositivos IoT desarrollada por ingenieros telemáticos de ESPOL.

En este capítulo se analiza el problema de la baja accesibilidad a los datos obtenidos por redes de sensores debido a que los mismos proveedores cuentan con sus propias plataformas donde se obliga al usuario a contratar su servicio para con el objetivo de visualizar, descargar y analizar los datos. Para ello, se plantea utilizar una plataforma de datos abiertos para uso de los usuarios dentro de la universidad.

A continuación se detalla la problemática en el bajo acceso de los datos medidos por las estaciones metereológicas y la privacidad que sus proveedores manejan. Además, se presenta una justificación del por qué es necesario la resolución de este problema para los estudiantes y profesionales. Se exponen los objetivos para establecer de qué

manera de abarcar la problemática y definir las metas a cumplir con el fin de determinar una solución eficaz. El alcance y las limitaciones de la solución propuesta y se describe el estado del arte en el cuál se detallan antecendentes y trabajos relacionados con el presente proyecto.

1.1 Problemática

Las estaciones meteorológicas permiten el monitoreo en tiempo cuasi-real de parámetros meteorológicos los cuales representan las condiciones climáticas presentes en una localidad que puede abarcar desde una parcela, hasta una región costera. Gracias a estos monitoreos o estudios del clima, surgen implicaciones indirectas en actividades de salud pública, gestión de riesgos, agricultura, pesca, gestión del agua, turismo, transporte y energía, así mismo como se puede realizar el muestreo de variables meteorológicas tales como temperatura, humedad, precipitación, dirección y velocidad del viento, entre otras [3].

Hace varios años, la forma de operación de una estación meteorológica era tan símple y ordinaria como que un operador, en horarios definidos, registre en una bitácora de papel lo visto a través de un monitor analógico el cual muestra las variables medidas por la estación. Actualmente, este proceso ha evolucionado debido al uso de nuevos equipos, los cuales ahora son digitales y dan la posibilidad de integrar muchos más elementos para formar un sistema automático de evaluación meteorológica. El futuro de las estaciones meteorológicas es prometedor, dado que es evidente que la automatización de sistemas es un avance que ya está entre nosotros. Tener estaciones meteorológicas automatizadas puede transformar la manera tradicional de medir variables meteorológicas ya que la información recopilada puede ser accesible desde sitios remotos sin necesidad de visitar lugares alejados y de difícil acceso [4].

La aparición y evolución del IoT, sumado a la gran imaginación de ingenieros de hardware y software, ha permitido crear sistemas inteligentes interconectando una variedad de dispositivos físicos; solo basta con tener un requisito que está al alcance de prácticamente todas las personas, la internet. Estos dispositivos electrónicos al mantener comunicación a través de una red de intercambio de datos, hace posible la automatización de ambientes de hogar, sector industrial, agricultor, pesquero y ganadero. Sin embargo,

la internet no lo es todo ya que aún teniendo varios equipos trabajando en conjunto, la información o datos experimentales necesitan ser mostrados en alguna plataforma, por lo cual, se evidencia una baja accesibilidad de los datos por parte de equipos propietarios [5].

Esto se debe a varias razones: la descarga directa de la información desde la memoria flash se realiza desde un software especializado, conectando la estación meteorológica a un computador portátil; la capacidad del hardware de almacenado de datos es poca y limitada; el equipo no cuenta con una interfaz de red para transmisión de datos cableada o inalámbrica; el equipo envía la data a la nube propietaria del fabricante del producto por lo cual el cliente final debe pagar una mensualidad de subscripción de servicios en la nube [6]. Gracias al IoT y al desarrollo de software, se pueden crear métodos generalizados automatizados en donde la estación meteorológica tenga la capacidad de comunicarse con otros equipos de forma cableada o inalámbrica e informar su estado actual, así como la información que se encuentra midiendo en tiempo cuasi-real, la misma que puede ser procesada y administrada a través de una plataforma web propia del cliente y no depender de proveedores. Por tal motivo, se propone este proyecto integrador con el objetivo de eliminar la baja accesibilidad que tienen los datos y presentarlos a través de una plataforma web propia alojada en un servidor de datos dentro de la universidad para el posterior procesamiento, y análisis de esta información; sin la necesidad de utilizar los servicios en nubes privadas de las empresas que brindan el acceso a este tipo de equipos electrónicos de medición de variables meteorológicas.

1.2 Objetivos

1.2.1 Objetivo General

Integrar dispositivos digitales a la plataforma de registro de dispositivos IoT de ESPOL mediante el uso de hardware y software de código abierto para la recolección automática de los datos.

1.2.2 Objetivos Específicos

- Conectar la estación meteorológica del Centro Nacional de Acuicultura e Investigaciones Marinas (CENAIM) a una red de internet para la visualización de data en tiempo real de manera automatizada.
- Establecer conexión entre la estación meteorológica y la plataforma de registros de dispositivos IoT para monitorear la telemetría del equipo.
- Desarrollar endpoints de consumo de datos dentro de la plataforma de registro de dispositivos IoT para la difusión de los datos hacia el público en general.

1.3 Alcance y Limitaciones

El presente proyecto propone estudiar y brindar una solución más amigable, interactiva y accesible para todos los usuarios que requieran hacer uso de los datos para sus respectivos análisis, conocidos como endpoints de datos abiertos. En esta sección, se aclara el alcance que permita la integración de equipos en la plataforma local de la ESPOL para conceder acceso libre a los datos debido a que actualmente para acceder a esos datos es necesaria una suscripción mensual. Esta solución va ser desarrollada como una propuesta alterna al funcionamiento que actualmente poseen las estaciones meteorológicas, es decir, no se verá afectado el funcionamiento actual de visualización de los datos a través del proveedor. Por otra parte, se advierten que para el desarrollo de este proyecto existen limitaciones de equipo que: no existe información necesaria de los equipos utilizados, la data recopilada tiene una baja accesibilidad, el hardware de los equipos es muy limitado respecto a conectividad [7].

Adicionalmente, otras limitaciones en el desarrollo de este proyecto son la seguridad digital de los datos y el respaldo de los mismos. Según el PhD. Venkata Venugopal en su artículo *IoT Digital: Principales problemas de seguridad* [8], detalla sobre los problemas que tienen las empresas cuando contratan servicios basados en Internet, debido a que la red IoT puede ser vulnerada mediante espías. Este intruso puede recibir y enviar cada uno de los mensajes recibidos durante la comunicación y de este modo obtener información

confidencial. En cuanto al respaldo de la información, no es posible realizar aquello debido a la poca accesibilidad a los dispositivos por parte de los fabricantes ya que ellos esperan que los usuarios se subscriban a sus servicios de forma inmediata e indefinida. Para Yongmin Zhao y Ning Lu, el respaldo de datos es cada vez más importante y se garantiza que es confidencial, completo y eficaz, además de no requerir operaciones manuales para su creación [9].

1.4 Estado del Arte

Debido al gran desarrollo tecnológico, la adquisición de los datos para su análisis es un punto importante para las empresas que utilizan diferentes tipos de sensores para controlar de una manera automática sus campos [10]. Estos análisis permiten alcanzar un mejor rendimiento de los espacios utilizados mejorando la comodidad de los usuarios interesados. No obstante, las empresas que brindan los dispositivos cuentan con su propia plataforma para visualizar, descargar o extraer la información que estos recopilan con la finalidad de que los usuarios adquieran una subscripción a su nube privada y ofrecerles el servicio. [1]

El artículo *Automatización en agricultura e IoT* [11], señala que la automatización e IoT intervienen en la creación de una solución más eficiente y rentable para los problemas de las personas. Se detallan las razones por las cuales la automatización y tecnología son eficientes entre las cuales estan: acceso rápido, minizar labores humanas, conectividad, ahorro de tiempo, comunicación eficiente y análisis. Gracias a eso se plantea resolver diferentes problemas en la agricultura con el objetivo de administrar la mayor parte de las actividades agrícolas y que los usuarios dediquen tiempo a plantear ideas, estrategias para sus cultivos de acuerdo con las demandas del mercado actual.

La automatización de la agricultura se puede combinar con los nuevos sistemas agrícolas como: la hidroponía, aeroponía y la acuaponía. Es posible lograr un sistema totalmente automático, los cuales puedan ser cosechados y sembrados sin la intervención humana, haciendo así que dichos cultivos sean mas orgánicos y crezcan más rápido.

El trabajo expuesto por K. A. Patil y N. R. Kale, miembros del departamento de Tecnologías de la Información MET, *Un modelo para la agricultura usando IoT* [12]

[1] https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-private-cloud/

presenta un modelo para desarrollar un sistema de monitoreo en tiempo real del suelo, y propone u orienta al usuario en la toma de decisiones para sus cultivos mediante el análisis de imágenes y alertas basadas en SMS (Short Message Service), del mismo modo, se le permite controlar operaciones del entorno mediante el uso de una aplicación móvil o parámetros ya establecidos por él mismo. Este modelo consta de tres módulos: lado de la granja, del servidor y del cliente. El lado de la granja consta de identificación de sensores, detección de parámetros, transferencia de datos, soporte de decisiones basadas en el análisis de los datos, actuación basada en parámetros establecidos por el usuario, y seguimiento de los cultivos mediante una cámara.

Por otro lado, la plataforma de registro de dispositivos de mediciones, el acceso a los datos almacenados, la visualización de gráficos y dashboards de las variables tomadas en la recopilación de los datos de los sensores se encuentran en el servidor. Finalmente, el lado del cliente, se compone de una aplicación web o móvil, mediante la cual por credenciales de acceso del lado de servidor puede acceder a los datos obtenidos, configuraciones de las alarmas, y acciones a tomar, como: encendido o apagado de ventiladores, o luces.

Proporcionar información actualizada del tiempo, les permite a los agricultores a preparse y beneficiarse de los cambios en los patrones climáticos con el fin de lograr una buena gestión en sus cultivos y utilizar de manera eficiente los recursos.

Cuasquer Chiscueth [13], en su proyecto *Aplicación de la tecnología IoT para la medición de variables meteorológicas en la agricultura sostenible* expone que las observaciones metereológicas son las mediciones de todos los elementos integrados que permiten interpretrar las condiciones del tiempo atmosférico en un lugar y tiempo determinados. Es necesario que estas observaciones sean tomadas lo más exactas posible, por ello se deben realizar de forma ininterrumpida con horas preestablecidas.

Para el monitoreo de las variables meteorológicas, se consideraron la temperatura, se define como el grado de calor o frío presente en el aire cerca de la superficie; humedad relativa, cantidad da vapor presente en la atmósfera; precipitación, cantidad de lluvia que viene de la atmósfera; heliofanía, entendido también como insolación, es el número de horas de sol y radiación solar, la cual es la cantidad de energía que emana el sol hacia la superficie. En este documento, se utilizan tres diferentes estadísticas para el monitoreo de los datos: RMSE, NASH y BIAS, los cuales permiten comparar valores reales tomados

desde la observación y valores simulados desde un modelo. Adicionalmente, se utiliza una plataforma web de código abierto para la recolección de datos, procesamiento, visualización y administración de dispositivos IoT, la cual utiliza protocolos MQTT (*Message Queuing Telemetry Transport*), CoAP (*Constrained Application Protocol*) y HTTP (*Hyper Text Transport Protocol*), también admite implementaciones de manera local o en la nube, con tolerancia a fallas, escalabilidad, confiabilidad y buen rendimiento.

En el documento, *Sistema de monitoreo de agricultura inteligente utilizando IoT* [14], el modelo propuesto utiliza diferentes sensores y equipos IoT para la recolección, análisis y transferencia de los datos obtenidos. El modelo usa un sensor de lluvia, un sensor de PIR, un modulo Wi-Fi y un módulo Arduino, el sistema plantea enviar datos al sitio web; igualmente cuenta con un sistema de envío de notificaciones y acciones para eventos predeterminados como: disminución de la humedad del suelo, se enciende el motor de agua. El diseño ayuda al agricultor a monitorear de manera remota el cultivo para aumentar la producción de los cultivos en la tierra, a través de una plataforma web o aplicación móvil implementada por la misma empresa que oferta los dispositivos. La agricultura inteligente es una idea que se adapta rápidamente al campo agrícola gracias a que ayuda al agricultor a monitorear continuamente el cultivo para aumentar su producción general.

El proyecto *Aplicación de IoT en la Agricultura* [15] propone que los dispositivos basados en IoT tienen sus aplicaciones en la agricultura para mejorar varios aspectos del agricultor. Gracias al uso de dispositivos inteligentes, los usuarios obtienen un mejor control sobre sus propiedades debido a permiten recopilar y procesar datos para optimizar procesos agrícolas y permitir que los agricultores tengan en tiempo real el estado de sus plantaciones, y puedan actuar acorde a las condiciones presentadas.

La arquitectura de este proyecto propone una capa de red central, capa de red perimetral, capa de usuario, capa de dispositivo y almacenamiento y capa de administración. Todos los sensores colocados en el área y la estación meteorológica se emplean para recopilar la información recibida y transmitirla hacia su herramienta analítica para su procesamiento y análisis. La automatización de tareas lentas y repetitivas es un gran beneficio para los agricultores, ya que les permite concentrarse en mejorar los rendimientos generales de su industria.

Según Dagar en su título *Agricultura Inteligente - IoT en Agricultura* [16], expresa

que la agricultura inteligente es la implementación de varios dispositivos y tecnologías tales como: internet, la nube y los dispositivos IoT. Para ello, se plantea un modelo a base de sensor de humedad del suelo, temperatura del aire y detección de movimiento, para monitorear cambios de estas variables en los cultivos y de este modo alertar a sus usuarios mediante una aplicación móvil, y actúen según sea el caso.

La agricultura inteligente ayuda a disminuir los costos de producción gracias a un mejor control y manejo de los recursos utilizados, con el objetivo de obtener más ganacias para los agricultores.

En esta nueva era, la nueva revolución industrial ha traído consigo un sin número de nuevas ideas, conocimientos mejorados, profesionales visionarios, técnicas avanzadas de producción lo cual permite sin duda alguna optar por las tecnologías inteligentes que fácilmente se pueden acoplar a organizaciones mundiales, personas y activos.

Según Khan, Rehman, Zangoti, et al, en su documento titulado *Internet de las cosas industrial: avances recientes, tecnologías habilitadoras y desafíos abiertos* [17], sostiene que estas tecnologías emergentes y aplicaciones de IoT en sistemas industriales permite destruir esquemas pasados y abrir un nuevo camino hacia el desarrollo de un nuevo sistema llamado IIoT (Industrial IoT) el cual permitiría automatizar objetos industriales para detectar, recopilar, procesar y comunicar eventos en tiempo real para de esa forma lograr una alta eficicencia operativa, aumentando la productividad y tener una mejor gestión de activos, recursos y procesos industriales. La personalización de estos nuevos sistemas industriales puede lograr un monitoreo inteligente al estado de maquinaria, mantenimiento predictivo y preventivo de equipos electrónicos industriales [18].

Pero, ¿el sistema industrial inteligente solo abarca los objetos que intervienen en la producción? La respuesta es no. Un sistema industrial inteligente también corresponde al tomar en consideración la etapa inicial de aprovisonamiento de energía eléctrica, ya sea desde una línea pública o generadores locales. De esta manera, Salcedo, Suárez, Solano y Henriquez, en su documento *Sistema inteligente para la gestión automática de un generador eléctrico basado en arquitectura del IoT* [19], presentan un diseño y prototipado de la gestión automática del generador eléctrico a través del protocolo de comunicación MQTT, el cual permite automatizar las diversas funciones de un generador ante una situación de interrupción de fluido eléctrico. Esta gestión inteligente registra variables de nivel de combustible, temperatura, horas de uso y mantenimientos.

Por otro lado, el IoT también ha incursionado en lo que ha geopolítica se refiere. Así lo demuestra sus aplicaciones que actualmente se encuentran en municipalidades de distintas ciudades a nivel mundial, ya que lo que tienen en común tanto el IoT y los entes gubernamentales es que el monitoreo es aliado para ambos. De esta forma, Pinto, Icaza y Alexander en su trabajo titulado *Análisis, diseño y desarrollo de un sistema inteligente y automatizado de monitoreo y control de cultivos con IoT. Caso de estudio: huertos urbanos* [20], proponen una nueva aplicación de IoT al sector urbano de una ciudad. Este sistema inteligente supervisará de manera permanente las necesidades de las plantas mediante sensores, registrando en tiempo real información exacta con las cuales se podrá llevar un control de los indicadores y tomar acciones inmediatas de forma automatizada sobre las planificaciones de riego de agua y apertura o cierre de cortinas para salvaguardar la salud del huerto urbano.

Es impresionante como el IoT está tomando protagonismo en distintos ámbitos, como se ha evidenciado en la industria y sector productor agrícola. Tener una gran red de dispositivos interconectados funcionando en conjunto abre muchas posibilidades en el mundo y a su vez cambia el estilo de vida de los seres humanos, ya que en algún punto del desarrollo tecnológico que está en auge, brinda soluciones a las necesidades que surgen en la sociedad [21].

Actualmente existe una temática la cual ha causado mucho interés a nivel global, la inteligencia artificial o IA por sus siglas en español. La IA es un conjunto de algoritmos los cuales permiten programar a una máquina en base a megadatos de información. Tal programación es tan sofisticada que podría realizar las mismas tareas que una persona haría en su vida cotidiana. Ahora bien, ¿qué algoritmos son estos?. Uno de estos algoritmos es el llamado Machine Learning. Junto con el IoT, ambas tecnologías podrían crear proyectos fantásticos, es por esto que Bolatti, Karanik, Todt, et al, en su publicación profesional titulada *Sistema inteligente de detección de anomalías para IoT* reflejan el gran alcance de ambas tecnologías.

Se propone un proyecto que tiene como objetivo mitigar ataques de seguridad, detectando a través de Machine Learning, anomalías en los dispositivos finales IoT, aplicando técnicas innovadoras en donde el programa aprenda de manera automática qué mecanismo es el más óptimo para utilizarlo en dicha detección [22].

En base a los estudios previos para la automatización de la agricultura, procesos

industriales, hogares, oficinas, competencias gubernamentales, entre otras, gracias al uso de dispositivos finales IoT, es posible concluir que el mayor porcentaje de proyectos que utilizan IoT en sus implementaciones es muy alto y que las ideas innovadoras no se hacen esperar. Sin embargo, hay cierto punto que se debe analizar muy aparte de todos los beneficios que ofrece el IoT, y este es la forma de cómo se accede al monitoreo, control, recopilación de datos que todos estos equipos pueden realizar [23].

Según Ahmed Roshdy, et al. en su artículo *Plataforma de visualización de datos genéricos* presenta una plataforma propia para visualización de datos [24]. Esta idea se replicó en el Laboratorio de Sistemas Telemáticos, desarrollando una plataforma de registro de dispositivos y visualización de datos recolectados. La visualización de la información se genera en gráficos de tendencia a través del tiempo.

Cada proveedor tecnológico al comercializar dispositivos finales IoT, ofrece en manuales de guía de uso, el acceso a las diferentes plataformas o aplicativos móviles que brindan cada una de estas empresas. En otras palabras, todas estas variables medidas o información recopilada ya es accesible para el usuario final, pero no siempre es así. Existen fabricantes o proveedores que ofertan estos equipos electrónicos inteligentes para mediciones y monitoreo con opción a acceder a la información a través de una plataforma propia de fabricantes. De esta forma, surge cierto problema en la sociedad el cual se refleja en lo poco accesible que es para el usuario final acceder a esta información debido a la carencia de portales de datos abiertos en donde sea factible la visualización de la información para su posterior procesamiento y análisis a cargo del usuario final [25].

CAPÍTULO 2

2. METODOLOGÍA

En este capítulo se aborda temas relacionados a los dispositivos utilizados en la implementación del sistema, así como también protocolos de comunicación, software y métricas para evaluar tanto el rendimiento del sistema, como la satisfacción del usuario ante la propuesta.

2.1 Antecedentes

Para el desarrollo del presente proyecto se realizó una implementación de hardware y software del Laboratorio de Sistemas Telemáticos (*LST*) en la Escuela Superior Politécnica del Litoral (*ESPOL*). Se consideran los casos de estudios de la Granja Experimental Agrícola (GEA) y el Centro Nacional Acuicultura e Investigaciones Marinas (CENAIM), de los cuales se toman sus respectivas estaciones meteorológicas para consumir los datos obtenidos y lograr integrarlos con la plataforma Haystack implementada dentro del laboratorio.

GEA, ubicada dentro de ESPOL, está compuesta alrededor de 7 hectáreas de terreno donde se han cultivado especias agrícolas como cacao, yuca, plátano, café, maíz, papaya y frejol. Además, se han instalado sistemas de riego para hortalizas en general y cultivo de arroz [26]. En definitiva, GEA posee variedad de parcelas de cultivos las cuales son parte del proceso de aprendizaje de estudiantes universitarios de ESPOL.

Actualmente, GEA tiene instalada una estación meteorológica marca ”Davis Instruments” [1], modelo ”Vantage Pro 2” en el campus Gustavo Galindo de ESPOL, ubicada cerca de la Facultad de Arte, Diseño y Comunicación (FADCOM). Ésta estación meteorológica es utilizada para medir variables físicas que incidan en los estudios

[1] https://www.davisinstruments.com/pages/weather-stations

agrícolas desarrollados sobre los cultivos antes mencionados. Las variables físicas abarcan temperatura, humedad, precipitación, dirección del viento, velocidad del viento y radiación solar.

La estación meteorológica de GEA se encuentra operativa y enviando los datos medidos hacia la nube propietaria del fabricante del producto. Este modus operandis no es de agrado de los ingenieros encargados de este equipo, debido a que para descargar la información requerida, ellos deben pagar una membresía anual para por medio de un computador portátil, abrir el software brindado por el fabricante para acceder a la data recopilada por la estación.

El Centro Nacional Acuicultura e Investigaciones Marinas (CENAIM) perteneciente a ESPOL, actualmente posee una estación meteorológica instalada en la provincia de Santa Elena en la comuna Valdivia. Esta estación meteorológica funcional se encuentra midiendo en tiempo real variables físicas como precipitación de lluvia, temperatura, humedad, velocidad del viento, dirección del viento y radiación solar.

Al igual que el caso de GEA, la data recopilada por la estación de CENAIM se encuentra alojada en una nube propietaria del fabricante del equipo por lo cual la accesibilidad de los datos es baja ya que se depende del fabricante para acceder a la información requerida.

El método de conseguir y consumir los datos obtenidos por ambas estaciones meteorológicas (GEA y CENAIM) no es el más ideal dado que al ser un equipo digital, no está aprovechando todas las capacidades que posee, como por ejemplo, enviar data en tiempo real a cualquier plataforma donde se capturen los datos.

Es por esta razón que se plantea como una solución paralela a lo que ya se encuentra funcional, integrar estos equipos a una plataforma de registro de dispositivos IoT desarrollada por ingenieros telemáticos de ESPOL en los cuales se puedan brindar endpoints de consumo de datos al público en general.

2.2 Arquitectura tecnológica

En la etapa inicial del sistema, los sensores marca ”Davis Instruments” son los encargados de recolectar los datos respecto a las variables meteorológicas de la zona. Estos datos son leídos a través del *Arduino Uno* mediante su respectiva programación de

código para mostrar los datos en un monitor serial. Una vez con los datos recolectados, estos son enviados hacia la *Raspberry Pi3*, la cual cumple con su función de *gateway IoT* y procesa los datos medidos para a través de un *script*, parsear la información a *JSON*. Posteriormente, a través de un software de túnel *HTTP* llamado *NGROK*, permite al *gateway IoT* enviar los datos a donde se encuentra levantado la plataforma de Haystack, directamente a la *API* para posteriormente ser almacenado en PostgreSQL, una base de datos no relacional clave-valor. Adicionalmente, los datos registrados se cargarán a los dashboards de cada una de las variables y a la tabla histórica (ver figura 2.1).

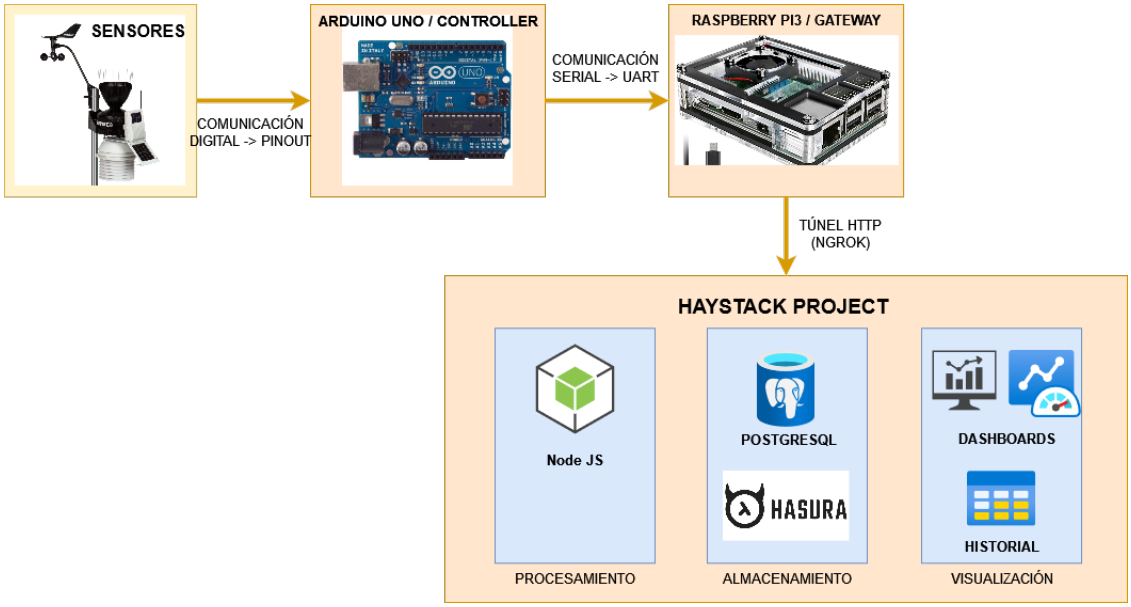


Figura 2.1: Arquitectura del sistema

2.2.1 Materiales y métodos

2.2.1.1 Sensores

Las estaciones meteorológicas de ambos casos de estudio (GEA y CENAIM) están conformadas por una variedad de instrumentos de medición meteorológica los cuales son los encargados de coleccionar los datos para su posterior envío. Los sensores que integran este sistema son:

1. **Anemómetro**

   Es un instrumento de medición meteorológica de la marca ”Davis Instruments” modelo ”6410” (ver figura 2.2), el cual permite conocer la velocidad del viento en kilómetros por hora (km/h) o a su vez en millas por hora (mph). Además, permite medir en qué dirección se encuentra circulando el viento de acuerdo a los polos magnéticos del planeta Tierra (Norte, Sur, Este, Oeste) con su respectivo grado sexagesimal. Su funcionamiento se basa en un transductor térmico que mide el flujo de velocidad a través del cambio del valor de la resistencia [27].

Figura 2.2: Anemómetro Davis 6410

2. **Colector de lluvia**

   Es un instrumento de medición meteorológica de la marca ”Davis Instruments” modelo ”7852” (ver figura 2.3), el cual permite conocer la cantidad de precipitación de lluvia en unidades de milímetros por hora (mm/h). Su funcionamiento se basa en contactos para cerrar circuito de acuerdo a la posición de las paletas recolectoras de lluvia las cuales tienen un aspecto de balanza [28].

Figura 2.3: Colector de lluvia Davis 7852

3. **Sensor de radiación solar**

   Es un instrumento de medición meteorológica de la marca ”Davis Instruments” modelo ”6450” (ver figura 2.4), el cual permite conocer la incidencia de radiación solar del momento en unidades de milivoltios (mV) y vátios por metro cuadrado (W/m2). Su funcionamiento se basa en los cambios de radiación que detecta el fotoresistor [29].

Figura 2.4: Sensor de radiación solar Davis 6450

4. **Sensor de temperatura y humedad**

   Es un sensor comercial, específicamente DHT22, el cual permite medir la temperatura y el porcentaje de humedad de la zona (ver figura 2.5). Según Adam Fikri Hishamudin et al., 2021 en su artículo *”Evaluación del sensor de temperatura DHT22 para la aplicación IoT”*[30] explica que dicho sensor proporciona una

medición precisa de la temperatura hasta a 0,1 grados centígrados. Sin embargo, es más lento en la detección de la tasa de cambio de temperatura, es decir, no es adecuado en una aplicación que requiere el monitoreo y retroalimentación de cambios rápidos de temperatura.

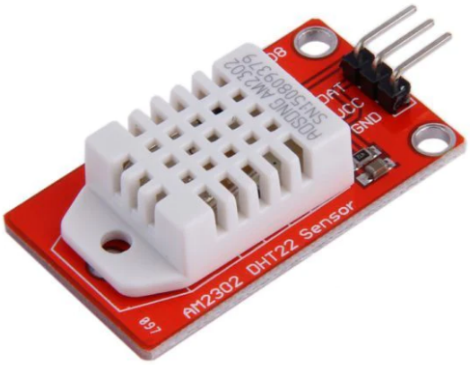


Figura 2.5: Sensor comercial DHT22

2.2.1.2 Arduino UNO

Arduino es una plataforma de electrónica para crear software de código abierto que al interactuar piezas de hardware entre circuitos, permite ejecutar programas utilizando una adaptación de los lenguajes de programación de C y C++ [31]. Gracias a su microprocesador y a su arquitectura, se puede controlar elementos electrónicos de índole digital o analógica (ver figura 2.6).

El framework de *Arduino* es una adaptación de los lenguajes de programación de C y C++, siendo una versión reducida y fácil de manejar este lenguaje. Consiste en interacturar los puertos de entradas y salidas, ya sean digitales, analógicas o avanzadas [32].

Para este proyecto, este lenguaje fue utilizado en la interconexión de todos los dispositivos (*sensores*), configurandolos de acuerdo con los parámetros establecidos, y así obtener los datos que cada uno de estos registre en su funcionamiento.

Figura 2.6: Microcontrolador Arduino UNO

2.2.1.3 Raspberry Pi 3

Raspberry Pi es un microordenador (ver figura 2.7), es decir, es una computadora de tamaño compacto y reducido como de una tarjeta de crédito. Su potencia de hardware no es similar al de un ordenador convencional debido a que todos sus componentes electrónicos son pequeños. Su uso está orientado al desarrollo de microsistemas informáticos [33].

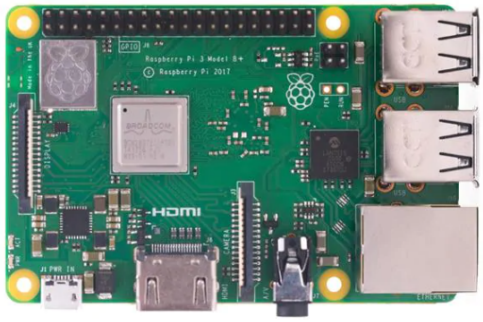


Figura 2.7: Raspberry Pi 3

2.2.2 Protocolos de comunicación

2.2.2.1 Señal Digital

Es una variable eléctrica con dos niveles que se alternan en el tiempo, trasmitiendo información en valores de 0 y 1. Este tipo de señales presenta inmunidad al ruido, facilidad de mezclar distintas fuentes, integración de múltiples servicios, monitoreo, adiconal

permite incorporar en el mensaje varias instrucciones con la finalidad de establecer, supervisar y gestionar la comuniación entre los distintos dispositivos[2].

2.2.2.2 Comunicación UART

Transmisor/receptor universal asincrónico (*Universal Asynchronous Receiver/Transmitter* por sus siglas en inglés), es un protocolo para el intercambio de información entre dos dispositivos. El uso de este protocolo es muy simple, debido a que utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones (ver figura 2.8). En la comunicación *UART*, los datos se transmiten en forma de tramas [34].

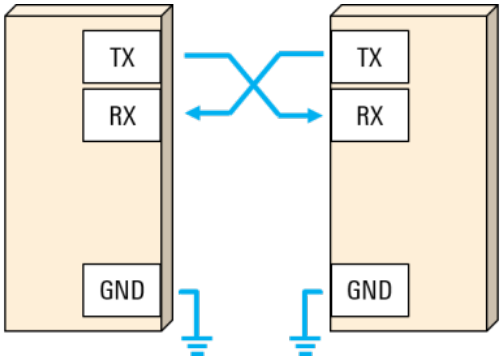


Figura 2.8: Conexión de la comunicación serial UART

2.2.3 Estandarización Haystack

Haystack es un marco de código abierto que tiene como objetivo estandarizar los modelos de datos tecnológicos para agilizar el trabajo con los dispositivos IoT utilizando taxonomías predeterminadas. Haystack incluye varias aplicaciones, entre las cuales se encuentran: automatización, iluminación, control, energía y varios sistemas ambientales [3].

Esto permitirá etiquetar todos los dispositivos IoT, y se detallará toda la información necesaria de cada uno de estos dispositivos, y pueda ser utilizada por parte del usuario final para realizar peticiones o consumo de los datos. Según Sarah A. Al-Qaseemi, en su artículo *Desafíos y problemas de la arquitectura de IoT: falta de estandarización* [35] explica que uno de los principales problemas importantes es la multitud de lenguajes,

[2] www.decu.com.mx/senal-digital-y-analoga-que-es
[3] https://project-haystack.org/

protocolos y estándares, así como la falta de acuerdo sobre cuál funciona mejor para las capas individuales de IoT.

2.2.3.1 Estructura de Datos

La estructura definida para el proyecto HayStack que se encuentra en LST, es recibir los datos en un formato tipo *JSON*, es decir, clave-valor. Donde los campos obligatorios son: ID, el cual es único para cada equipo registrado, unidades de medición de cada variable, descripción del equipo y valor medido. Adicionalmente, los demás campos estan etiquetados como opcionales, los cuales brindan información extra sobre el equipo.

2.2.4 NGROK Cloud Edge

Es una tecnología de sistemas en la nube de borde (o ”Edge” por su terminología en inglés) en la cual un túnel multiplataforma y un proxy inverso de software ayuda a establecer un túnel seguro desde un punto final público como Internet a una red privada. Ngrok permite que el servicio desplegado local sea disponible inmediatamente a través de la Internet permitiendo acceder realizando peticiones HTTPS al servidor de NGROK [36].

Para el sistema que se ha desarrollado, se utiliza la licencia gratuita de NGROK la cual ofrece las siguientes características [4]:

Licencia

- 1 miembro de equipo, 1 acceso a agente de túnel seguro, 1 dominio de la nube de borde.

Conectividad

- Soporta despliegue de tunel HTTPS.

Desarrollo

- Soporta agente de inspección web, re-hacer requerimientos/peticiones, servidor de archivos.

Rendimiento

[4] https://ngrok.com/pricing

- Soporta conexiones HTTP/2, HTTP Compression.
- No soporta balanceo de carga ni balanceo de peso para tráfico de borde.

Escalabilidad

- Soporta 2 tunel por agente.
- Soporta hasta 100 conexiones TCP (Transmission Control Protocol) simultáneas.
- El índice (rate) máximo de conexiones TCP simultáneas es hasta 120/min.
- Ancho de banda de hasta 1GB/mes.

2.3 Criterios de inclusión y exclusión

Para el desarrollo de un proyecto investigativo o documento de índole similar, es importante tomar en cosideración los factores que permiten garantizar los resultados más óptimos en el transcurso y finalización del estudio investigativo. De la misma forma, pueden aplicarse al desarrollo o implementación de un sistema tecnológico, es decir, analizar estos criterios durante el proceso de desarrollo de infraestructura tecnológica y de software que permita el correcto funcionamiento de todo el sistema.

Los criterios de inclusión del proyecto comprende desde la elección correcta de los sensores de medición para tener la mayor fidelidad de los datos medidos a través de cada uno, hasta el manejo de los datos medidos para presentarlos en una plataforma web. Por esta razón, los objetos de medición según el nivel de fidelidad que se necesita para este proyecto, fueron escogidos de la marca ”Davis Instruments”. Para el proceso de colección, manejo de datos, procesamiento y envío de los datos se eligieron el actuador Arduino Uno para las dos primeras tareas y la Raspberry Pi 3 para las dos restantes. Así mismo, los procotocolos de comunicación elegidos son los más adecuados para los equipos y el entorno de desarrollo de software es muy intuitivo y de fácil entendimiento.

Por otra parte, los criterios de exclusión del proyecto puede referirse a lo complejo y limitado que pueden ser los equipos de hardware del sistema. Un ejemplo claro es la compatibilidad de la estación meteorológica de GEA y lo muy limitado que es en conexiones físicas e inalámbricas debido a que las características de la estación

meteorológica pertenece a la versión más básica de los equipos, y además que en intentos de acoplar nuevo hardware para darle más facilidad de uso, todo fue imposible de manipular o acceder a la información medida. Lo que conlleva que todo el sistema desarrollado e implementado en este proyecto, solo sea aplicable a la estación de CENAIM y las pruebas respectivas son pertenecientes a la misma.

2.4 Métricas de Rendimiento

Las métricas consideradas son: la latencia y la tasa de transferencia efectiva, *throughput*, ya que estas métricas nos permiten comprobar el rendimiento de nuestra experimentación, con las cuales podemos ”estresar” con múltiples peticiones o solicitudes al servidor de *NGROK*.

La latencia se define como un retraso en la comunicación, es decir, el tiempo de transferencia de los datos a través de la red. Para medir la latencia, es posible utilizar peticiones HTTP, paquetes ICMP, tiempo de respuestas en milisegundos, entre otros [5].

Por otro lado, el throughput, es la cantidad de datos que se pueden enviar y recibir en el medio de almacenamiento. Se mide normalmente por bits por segundo (bps)[6].

Para realizar estas mediciones utilizaremos el aplicativo *Jmeter*, el cuál es una herramienta de código abierto desarrollada para comprobar el estado funcional de un aplicativo web o, actualmente, también utilizada para probar rendimiento de servidores o proyectos de redes.[7].

2.5 Métricas de Satisfacción del Usuario

Con el fin de medir obtener una retroalimentación por parte de los usuarios, se plantea realizar encuestas de satisfacción a cada uno que utilice el aplicativo. La muestra esta conformada por los estudiantes y profesores de la Facultad de Electricidad y Computación (FIEC) y un trabajador de la Facultad de Ciencias de la Vida (FCV) de la universidad ESPOL, los cuales podrían usar el sistema para utilizar la información medida en investigaciones o en planes de riego de cultivos. De acuerdo a la página web oficial

[5] aws.amazon.com/es/what-is/latency/
[6] www.techtarget.com/searchnetworking/definition/throughput
[7] https://jmeter.apache.org/index.html

de la FIEC[8], existen aproximadamente 2500 estudiantes pertenecientes a la facultad. Con esta información obtenida, se procede a calcular el tamaño de la muestra de una población finita a través de la fórmula (2.1).

$$muestra = \frac{\frac{z^2 p(1-p)}{e^2}}{1 + \frac{z^2 p(1-p)}{e^2 N}} \tag{2.1}$$

Donde:

- N: tamaño de la población.
- e: margen de error (%).
- p: probabilidad a favor (%).
- z: puntuación z para estimar el nivel de confianza.

El valor z es proporcional al nivel de confianza que se desea para tomar la muestra del proyecto. En la tabla 2.1 se puede visualizar los diferentes valores de z para los niveles de confianza más utilizados.

Tabla 2.1: Tabla valores Z y niveles de confianza

Niveles de Confianza	Valor z
80%	1,28
85%	1,44
90%	1,65
95%	1,96
99%	2,58

Considerando la población mencionada en el párrafo anterior, se decidió seleccionar un margen de error del 14%, y un nivel de confianza del 80%; donde luego de realizar los cálculos necesarios, se obtiene una muestra de aproximadamente 21 personas entre las que se incluyen estudiantes de la facultad y el trabajador de la FCV. Posteriormente, la cantidad de personas a favor de la variable que se busca medir, indicada por la letra p, es del 0,5 de los encuestados. Las preguntas propuestas hacia esta muestra en

[8] https://www.fiec.espol.edu.ec/

particular están orientadas a respuestas sencillas, tal como se muestra en la tabla 2.2:

Tabla 2.2: Tabla de respuestas

Respuestas
Sí
No
Otros

CAPÍTULO 3

3. DISEÑOS E IMPLEMENTACIÓN

En este capítulo se detalla cada uno de los diagramas de interacción entre el usuario, objetos, información de la base de datos, además comandos necesarios para instalación del software *NGROK* y pseudo-código de los *scripts* utilizados en la implementación del sistema.

3.1 Modelo de comunicación

Para el correcto funcionamiento entre el controlador *Arduino* y el *gatway IoT* (Raspberry Pi3), se deben crear sockets o puertos de comunicación serial entre ambas entidades. El concepto ligado a la apertura de puertos obedece al modelo de comunicación entre cliente-servidor programado en lenguaje C, tal y como se lo aplica en la programación de sistemas *UNIX* o *LINUX*. El medio de comunicación entre ambas placas es inalámbrico, utilizando tecnología *bluetooth*.

El algoritmo 1 nos muestra la apertura de un puerto de comunicación serial al cual se le asigna el nombre de *"arduino"*. Existen parámetros de configuración se deben aplicar para este tipo de comunicación entre los cuales destacan: el *baudrate* que obedece al número de señales por segundo al cual es enviada los datos y *port* que indica el puerto de comunicación serial que utilizan. A través de un comando de sistema, se *empareja* ambos dispositivos ligados al puerto *rfcomm1*. De la misma forma, se hace uso de librerías propias de python las cuales permiten instanciar funciones necesarias para la ejecución correcta del algoritmo.

Algoritmo 1 Estracto de Algoritmo escrito en Python para apertura de puerto serial

```
import serial
import time
import requests
import json
import os

os.system("sudo rfcomm bind rfcomm1 20:13:09:24:18:90")

time.sleep(10)


arduino = serial.Serial()
arduino.baudrate = 9600
arduino.port = "/dev/rfcomm1"
arduino.open()
```

3.2 Modelo de obtención de datos de los sensores

A continuación se detalla la sección de código *setup()* cargado en la placa *Arduino*, en donde se observa variables utilizadas en el código, la velocidad de transmisión de los datos, inicio de transmisión de datos por parte de los sensores y establecer que pines *GPIO* utilizará cada sensor, además de un temporizador de 5 segundos para la interrupción con el fin de evitar falsos contactos.

Algoritmo 2 Estracto de Algoritmo Arduino función $setup()$

```
void setup() {
    LastValue = 0;
    IsSampleRequired = false;
    TimerCount = 0;
    Rotations = 0; // Set Rotations to 0 ready for calculations
    Serial.begin(9600);
    dht.begin();

    pinMode(WindSensorPin, INPUT);
    pinMode(interruptPin, INPUT_PULLUP);
    pinMode(SOLAR, INPUT);
    analogReference(EXTERNAL);
    attachInterrupt(digitalPinToInterrupt(WindSensorPin), isr_rotation, FALLING);
    attachInterrupt(digitalPinToInterrupt(interruptPin), count, FALLING);

    // Setup the timer interupt
    Timer1.initialize(500000);// Timer interrupt every 2.5 seconds
    Timer1.attachInterrupt(isr_timer);
}
```

3.3 Modelo de envío de datos hacia gateway IoT

El algoritmo 3 presenta el modelo mediante el cuál se envian los datos recolectados por los sensores implementados y conectados a la placa *Arduino* hacia nuestro módulo *gateway IoT*. El formato utilizado es *JSON*, dados por parámetros de *clave-valor*; donde la clave es cada una de las variables medidas y su valor, la medida tomada por los dispositivos. Así mismo, se presenta el *ID* y el nombre del equipo (*base*) al que pertenecen los puntos o sensores utilizados.

Algoritmo 3 Esquema de envío de datos de Arduino en tipo JSON

```
//PRINT OF DATA

Serial.print("{\"id\": \"base\""); Serial.print(", ");
Serial.print("\"WindSpeed[MPH]\": ");
Serial.print(WindSpeed);
Serial.print(", ");
Serial.print("\"Knots\": ");
Serial.print(getKnots(WindSpeed));
Serial.print(", ");
Serial.print("\"Direction\": \"");
Serial.print(CalDirection); getHeading(CalDirection);
Serial.print("\", ");
Serial.print("\"Strength\": \""); getWindStrength(WindSpeed);
Serial.print("\", ");
Serial.print("\"Humidity[%]\": ");
Serial.print(humidity); Serial.print(", ");
Serial.print("\"Temperature[C]\": ");
Serial.print(temperature); Serial.print(", ");
Serial.print("\"Temperature[F]\": ");
Serial.print(fahrenheit); Serial.print(", ");
Serial.print("\"Heat Index[C]\": ");
Serial.print(hic); Serial.print(", ");
Serial.print("\"Heat Index[F]\": ");
Serial.print(hif); Serial.print(", ");
Serial.print("\"Voltage[mV]\": ");
Serial.print(outputSolar * 1000.00); Serial.print(", ");
Serial.print("\"Watts[W/m2]\": ");
Serial.print((int)(outputSolar * 1000.00 / 1.67));
Serial.println("}");
```

3.4 Líneas de comandos de terminal Linux para desplegar un túnel HTTP

Comandos necesarios para implementar *NGROK* en linux con el objetivo de obtener un dirección IP pública del sistema Haystack que se encuentra implementado en un servidor del Laboratorio de Sistemas Telemáticos.

- # sudo -s: Ingresar a usuario root
- # sudo apt install ngrok: Instalar NGROK en la máquina
- # ./ngrok http 8082: Hacer pública la ip local

3.5 Funcionalidad del gateway IoT y envío de datos hacia la API

Se declara una variable que contenga la dirección web pública que brinda el servidor de NGROK para el envío de la información. Además, se crea una sentencia *while* para que el programa se ejecute de forma ininterrumpida, y realice la lectura de los datos de la placa *Arduino*, asignarlos a la variable, y enviarlos hacia la *API* del sistema de registro IoT.

Algoritmo 4 Estracto de Algoritmo Python función $main()$

```
url = "https://094b-2801-0-20-8c9-ba79-d8d-2df4-2d56.ngrok.io/v1/registrar-datos"
headers = {'Content-type': 'application/json', 'Accept': '*/*'}

while True:
getData = arduino.readline()
dataString = getData.decode("utf-8")
data = dataString[:-2]
data1 = json.dumps(data)
jsondata = json.loads(data1)
page = requests.post(url, jsondata, headers=headers)
print(jsondata)
print("Sending data.....")
```

3.6 Funcionalidad del sistema

El siguiente diagrama detalla la manera correcta de proceder para realizar la correcta implementación de todo el sistema, desde la integración de cada uno de los sensores, realización de pruebas, conexión hacia el *gateway IoT*, y hacia a la plataforma web. La funcionalidad del sistema se visualiza en la figura 3.1, la cual se divide en 6 secciones: implementación de los dispositivos, recolección de datos, estandarización, conectividad hacia la plataforma web, actualización en tiempo real, y consumo de los datos obtenidos de la estación meteorológica.

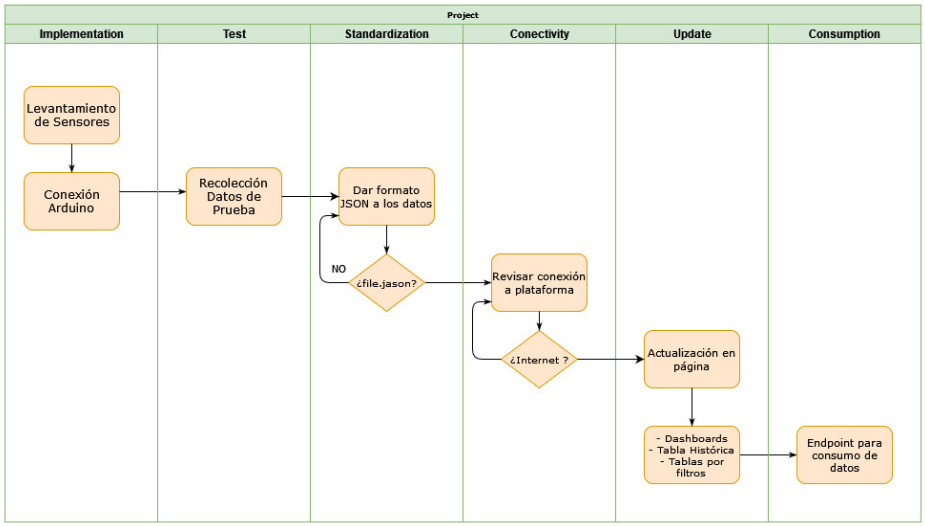


Figura 3.1: Diagrama de funcionalidades

3.7 Diagramas de Comunicación

- **Diseños de interacción de usuarios**

  En la siguiente figura (3.2) se puede visualizar el algoritmo que el usuario debe seguir para un correcto uso del sistema implementado. Como primer paso es la implementación de los dispositivos con el gateway, la Raspberry Pi 3. Luego, el gateway enviará los datos obtenidos hacia la plataforma Haystack en formato JSON, donde serán almacenados en la base de datos. Finalmente para visualizar estos datos, se leerán desde el almacenamiento y presentados en sus respectivos dashboards dentro de la plataforma.

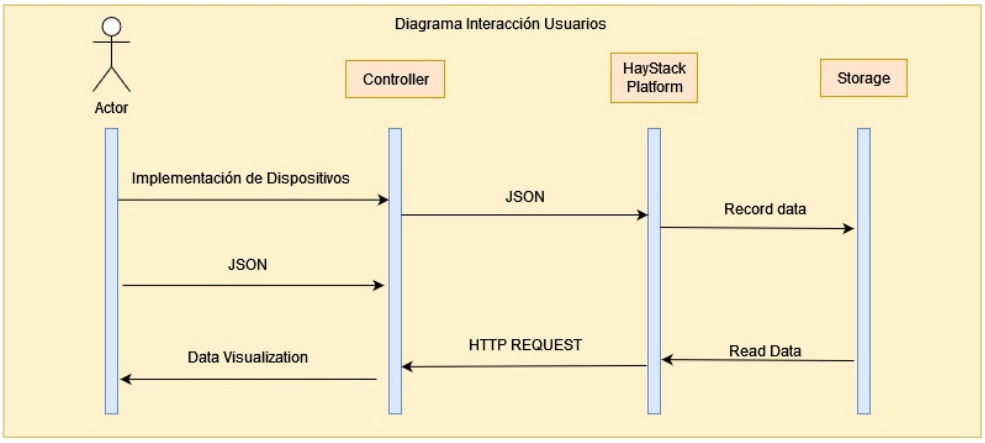


Figura 3.2: Diagrama de Interacción del Usuario

- **Diseños de interacción de datos**

  En cuanto a la interacción de los objetos, se divide en 3 partes, tal como se muestra en la figura 3.3: la recolección de los datos, conversión y envío. En la primera parte los datos obtenidos mediante los sensores conectados y programados mediante la placa arduino. A continuación, mediante la Raspberry Pi 3, estos datos son convertidos a formato JSON para finalmente ser enviados a la plataforma y ser visualizados. Gracias a este diagrama se puede visualizar los pasos que siguen cada uno de los datos recolectados por los sensores, estandarización en formato *JSON* para al final ser leída por la *API* de Haystack.

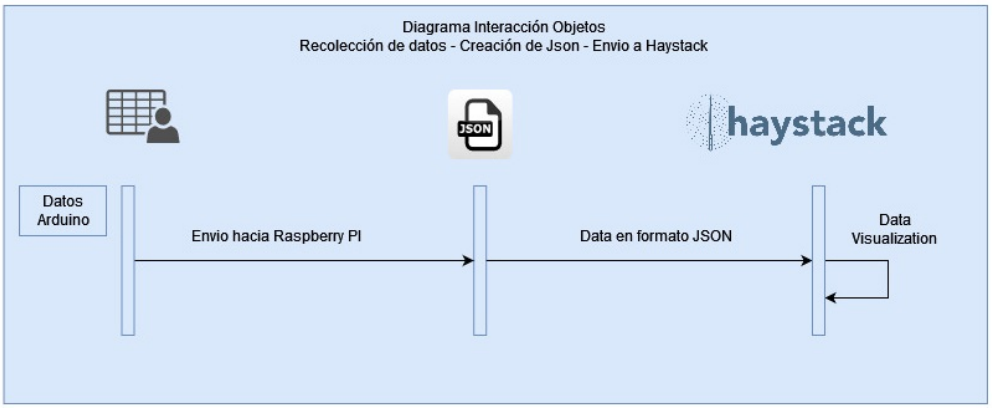


Figura 3.3: Diagrama de interacción de objetos

3.8 Diagramas de Base de Datos

El propósito de este diagrama es dar a conocer al usuario final, el correcto formato en el que se van a registrar cada uno de los valores obtenidos por los sensores. Para el almacenamiento de la información se detalla el siguiente diagrama. La base de datos del sistema es no relacional, definida por los parámetros de clave-valor, la cual nos permite registrar utilizando la clave como identificador único (figura 3.4).

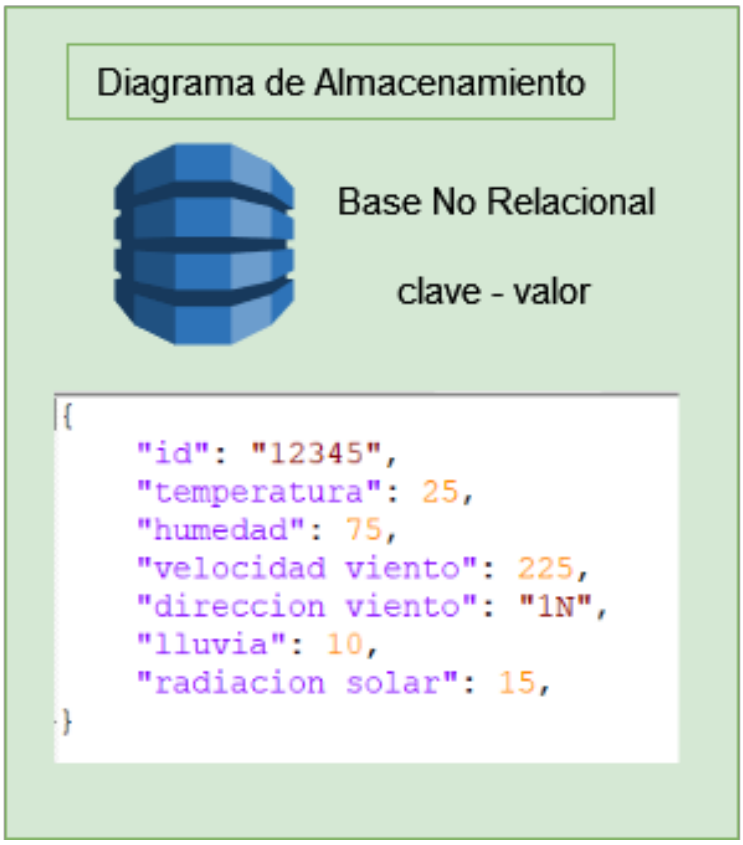


Figura 3.4: Diagrama de Almacenamiento

3.9 Diseño de plataforma o aplicación web (endpoints)

Para esta sección, se detalla el desarrollo del *front-end* en el Sistema de Registro de Módulos *IoT* implementada en el Laboratorio de Sistemas Telemáticos. La figura 3.5 presenta la vista general de la página para la visualización de los datos abiertos de cada una de las estaciones meteorológicas implementadas dentro del área de la ESPOL. Cada estación cuenta con su propio direccionamiento.

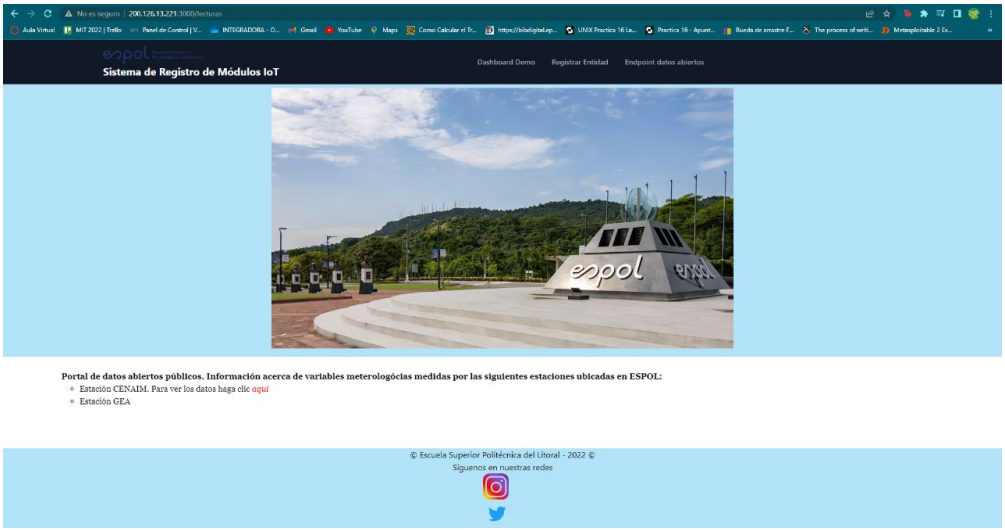
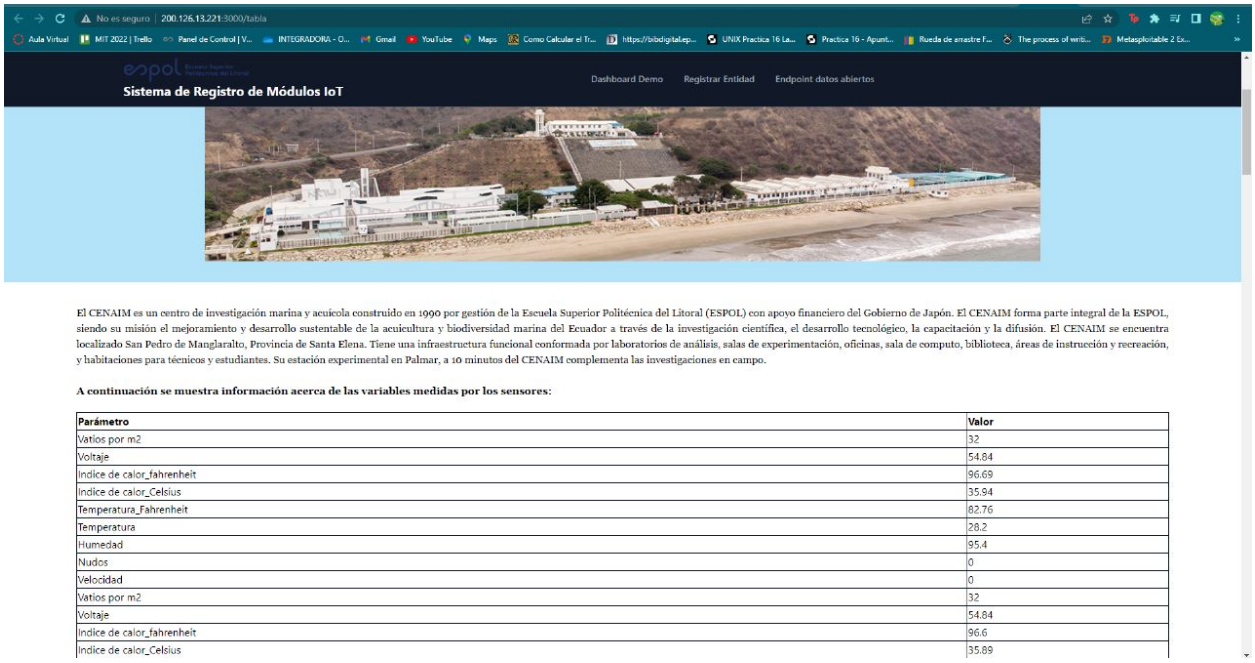


Figura 3.5: Vista general de la página de datos abiertos

Al elegir la estación metereológica para visualizar sus datos medidos, el aplicativo redirecciona al usuario a una nueva página web donde se presenta una breve descripción de CENAIM, quien ha sido parte del proyecto, y además una tabla histórica en donde se encuentran reflejados los valores recolectados por los sensores, tal como se presenta en la figura 3.6, donde podemos visualizar las variables que la estación está midiendo en ese instante de tiempo; variables como: *voltaje, índices de calor, temperatura en grados celcius, temperatura en grados fahrenheit, índice de calor, humedad, velocidad, nudos(pluviosidad)*.

Estos datos, gracias a los *scripts* desarrollados en los algoritmos 2 y 3, se actualizan en intervalos de 10 segundos. Esta configuración de lectura de datos, se realiza para una mejor precisión de la información en los sitios donde se encuentran dichas estaciones metereológicas, y de esta manera, poder realizar análisis más certeros y predicciones más confiables.



El CENAIM es un centro de investigación marina y acuícola construido en 1990 por gestión de la Escuela Superior Politécnica del Litoral (ESPOL) con apoyo financiero del Gobierno de Japón. El CENAIM forma parte integral de la ESPOL, siendo su misión el mejoramiento y desarrollo sustentable de la acuicultura y biodiversidad marina del Ecuador a través de la investigación científica, el desarrollo tecnológico, la capacitación y la difusión. El CENAIM se encuentra localizado San Pedro de Manglaralto, Provincia de Santa Elena. Tiene una infraestructura funcional conformada por laboratorios de análisis, salas de experimentación, oficinas, sala de computo, biblioteca, áreas de instrucción y recreación, y habitaciones para técnicos y estudiantes. Su estación experimental en Palmar, a 10 minutos del CENAIM complementa las investigaciones en campo.

A continuación se muestra información acerca de las variables medidas por los sensores:

Parámetro	Valor
Vatios por m2	32
Voltaje	54.84
Indice de calor_fahrenheit	96.69
Indice de calor_Celsius	35.94
Temperatura_Fahrenheit	82.76
Temperatura	28.2
Humedad	95.4
Nudos	0
Velocidad	0
Vatios por m2	32
Voltaje	54.84
Indice de calor_fahrenheit	96.6
Indice de calor_Celsius	35.89

Figura 3.6: Vista de tabla histórica de datos recolectados

Finalmente, la figura 3.7 representa un diagrama de *unidad de medida vs tiempo* en donde se puede apreciar la tendencia o comportamiento de la variable de humedad mientras transcurre el tiempo. Este *dashboard* es parte del *front-end* del Sistema de Registro de Módulos *IoT*, sin embargo, su desarrollo fue realizado los ingenieros Bryan Pilco y Erick García [37] en su proyecto integrador titulado *Integración de módulos IoT con Sistemas Hardware-Software para Edificios Inteligentes mediante la estandarización de datos*. Nuestra propuesta acogió su funcionalidad para integrarlos a los dispositivos IoT registrados permitiendo la visualización de datos en tiempo real.

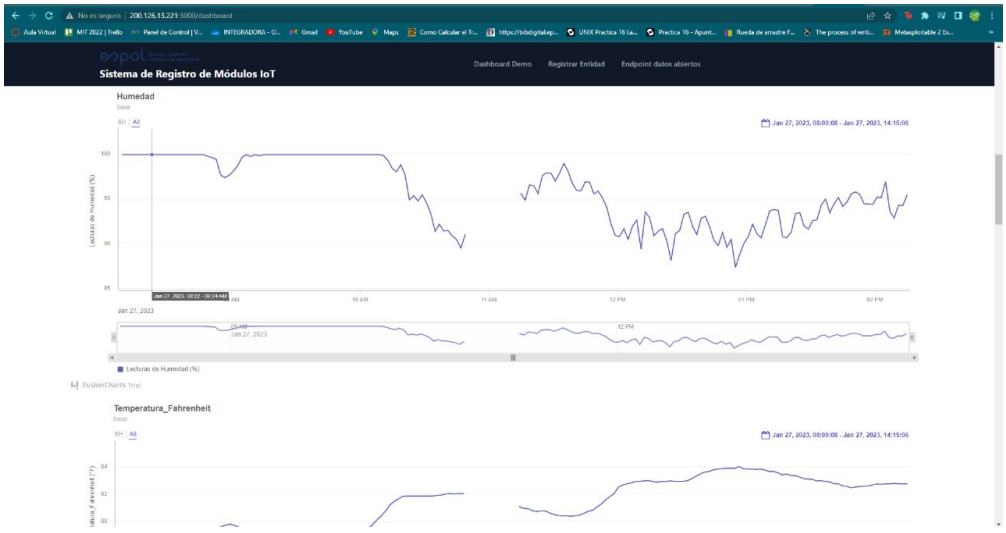


Figura 3.7: Dashboard de temperatura y humedad

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

En este capítulo se detallan las evaluaciones que se realizaron al sistema de integración de endpoints IoT, para conocer su rendimiento frente a registros múltiples de dispositivos IoT, pruebas que son redireccionadas al servidor de NGROK donde se encuentra funcionando la API de Haystack en una URL pública. Además, se realiza la respectiva encuesta de satisfacción de la plataforma web por parte de los usuarios finales. Para finalizar, se analiza el presupuesto del proyecto a nivel de hardware, software y mano de obra humana.

4.1 Escenarios de prueba

En capítulos anteriores se detalló el funcionamiento del sistema y objetivamente las peticiones HTTP realizadas a la plataforma Haystack. De esta forma, a través de 4 escenarios de pruebas, se puede evaluar las métricas de rendimiento para este proyecto que se destacó en el capítulo 2.

4.1.1 Métricas de rendimiento

Todo sistema o aplicación debe poder satisfacer las necesidades del usuario final o consumidor, es decir, que el funcionamiento integral sea el esperado por el cliente con el objetivo de cumplir las demandas que ellos proponen. Es así que la calidad de cualquier aplicación puede ser controlada a partir de estimaciones respecto a diferentes criterios de rendimiento, para lo cual mediante frameworks de evaluación, se puede contemplar estos criterios como por ejemplo la latencia de la aplicación, la velocidad, la capacidad de atender tantos requerimientos como se pueda y los errores de red que pudiesen

presentarse [38]. Para esta aplicación web de registro de dispositivos IoT, es importante conocer el rendimiento que refleja este sistema a través del análisis de las siguientes métricas:

- Latencia: Tiempo promedio de la transferencia de los datos a través de la red en la ruta definida desde el usuario final hasta la API de Haystack. Es medido en ms.
- Rendimiento: Determina la cantidad de datos que pueden ser atendidos por el sistema en simultáneo. Su medida es en kb/s.

4.1.2 Escenario 1: Envío de 40 requerimientos HTTP

En el primer escenario de prueba con 40 peticiones de manera simultánea, los resultados de la tabla 4.1 reflejan que los valores de latencia son elevados, tanto el mínimo como el máximo valor, tiempo que tardan las peticiones HTTP en llegar a su destino.

Por otro lado, el rendimiento del sistema es muy bueno dado que se indica cuantas peticiones HTTP por unidad de tiempo fueron atendidas exitosamente, lo cual refleja un valor de 8.82 kb/s. En la figura 4.1, se observa un detalle de todos los requerimientos realizados en donde no hubo pérdidas de niguno de ellos.

Tabla 4.1: Tabla de resultado del escenario 40 solicitudes

Métrica	Escenario: 40 HTTP Requests	Unidades
Mínimo de Latencia	1012	ms
Máximo de Latencia	1715	ms
Media de Latencia	1455	ms
Rendimiento	8.82	kb/s
Error	0.0	%

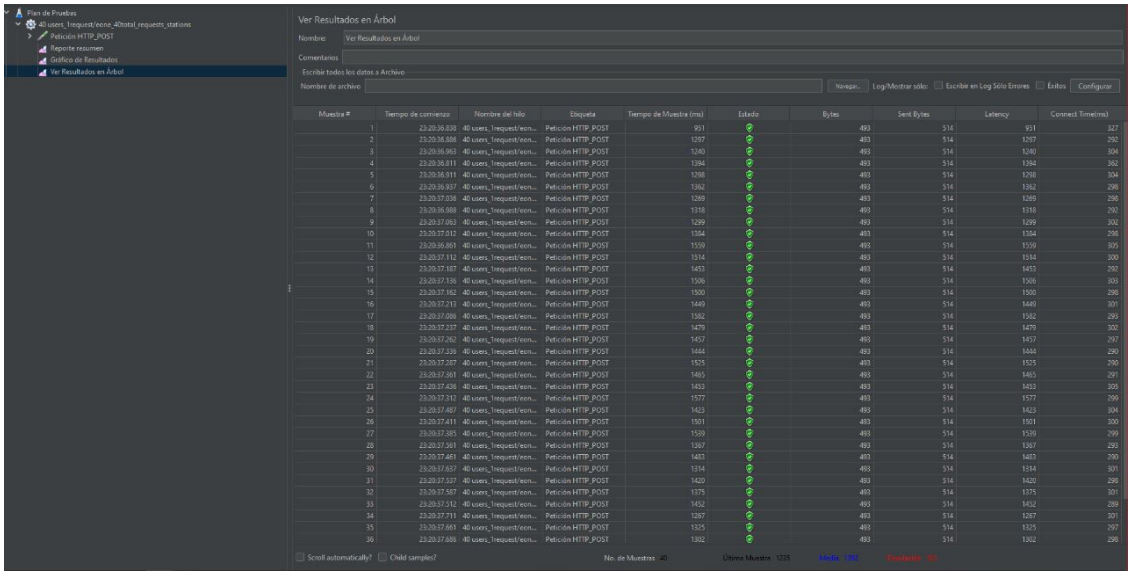


Figura 4.1: Detalle de envío de 40 peticiones simultáneas

4.1.3 Escenario 2: Envío de 50 requerimientos HTTP

En el segundo escenario de prueba con 50 peticiones de manera simultánea, los resultados obtenidos en la tabla 4.2 indican que existen indicios de saturación del puerto 443 que recibe las peticiones HTTP. De acuerdo a lo especificado en el capítulo 2 acerca del rendimiento de los servidores de NGROK, se puede verificar que la escalabilidad es limitada, por lo cual, al realizar las 50 solicitudes en simultáneo, el puerto 443 empieza a saturarse y existe pérdida de solicitudes, reflejado en el porcentaje de error alcanzando un valor de 8.0%. El rendimiento en comparación al escenario 1, es mayor debido a que las solicitudes aumentaron, es decir, el índice máximo de conexiones (especificado en el capítulo 2) es directamente proporcional al número de solicitudes realizadas.

Por otro lado, los valores de latencia también aumentaron en comparación a las pruebas realizadas anteriormente. La tolerancia a fallos del servidor de ngrok al tratar de recuperar las solicites fallidas, provoca que exista este aumento en el tiempo que toman los paquetes en llegar a su destino. En la figura 4.2 se muestra un detalle de los paquetes con estado exitoso y fallido al realizar las pruebas.

Tabla 4.2: Tabla de resultado del escenario 50 solicitudes

Métrica	Escenario: 50 HTTP Requests	Unidades
Mínimo de Latencia	436	ms
Máximo de Latencia	2178	ms
Media de Latencia	1695	ms
Rendimiento	11.39	kb/s
Error	8.0	%

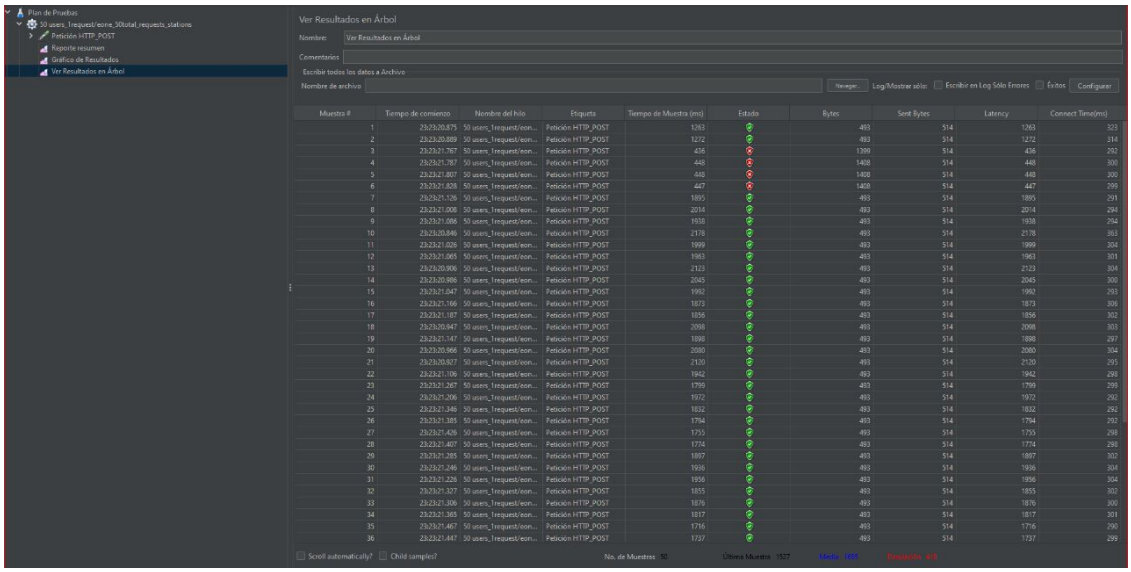


Figura 4.2: Detalle de envío de 50 peticiones simultáneas

4.1.4 Escenario 3: Envío de 55 requerimientos HTTP

Este escenario de prueba al igual que el anterior, se encuentra en una situación de estrés debido a que la cantidad de solicitues aumentaron. Los resultados obtenidos en la tabla 4.3 muestran valores casi similares que en el escenario anterior si comparamos el promedio de latencia en las peticiones realizadas.

Respecto al rendimiento, los resultados muestran que hubo un muy pequeño decremento pero es muy cercano al que se obtuvo en la prueba anterior. En la figura 4.3, se visualiza el estado de cada una de las peticiones HTTP realizadas en donde existen más peticiones que no llegaron al destino debido a la saturación del puerto 443 lo cual se puede verificar en el porcentaje de error de la tabla 4.3.

Tabla 4.3: Tabla de resultado del escenario 55 solicitudes

Métrica	Escenario: 55 HTTP Requests	Unidades
Mínimo de Latencia	834	ms
Máximo de Latencia	1976	ms
Media de Latencia	1678	ms
Rendimiento	10.37	kb/s
Error	9.09	%

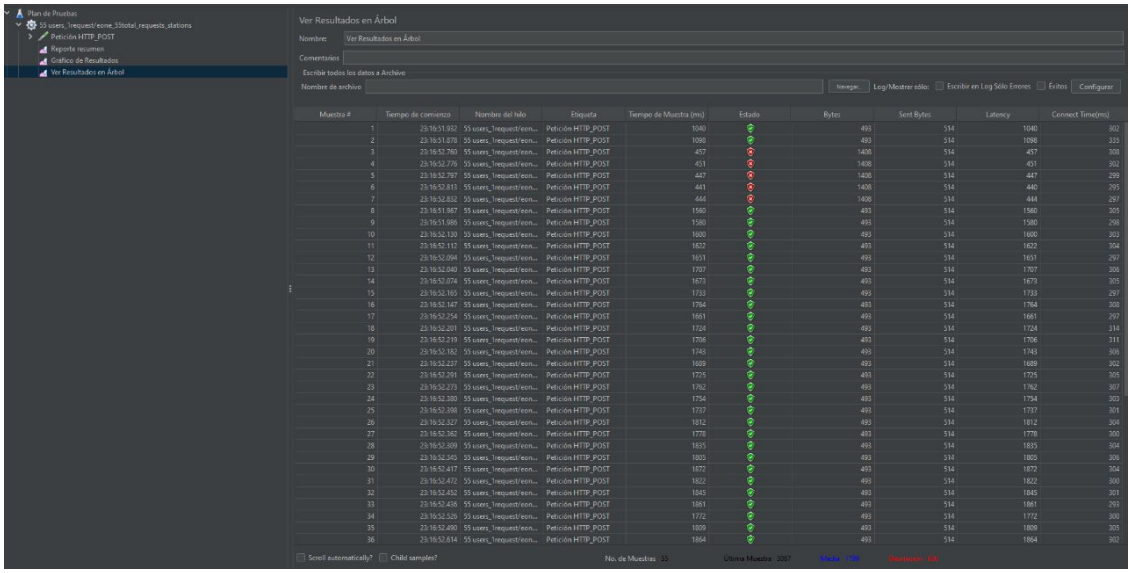


Figura 4.3: Detalle de envío de 55 peticiones simultáneas

4.1.5 Escenario 4: Envío de 60 requerimientos HTTP

Este último escenario de prueba muestra resultados aún más críticos. De acuerdo a lo que se observa en la tabla 4.4, se obtuvieron resultados de un estado crítico de saturación del puerto 443 debido a los requerimientos HTTP. Debido a este estado, el promedio de latencia alcanzado es mucho menor que en las 3 pruebas anteriores, dado que muchas peticiones simplemente no llegaron al destino y, por lo tanto, hubo más rapidez en cuanto al tiempo de respuesta de los requerimientos que si lograron ser atendidos.

Respecto al rendimiento, en esta prueba hubo un aumento considerable debido a que el número de solicitudes aumentó de la misma manera, ya que ambos son directamente proporcionales. En la figura 4.4, se visualiza el escenario crítico de las solicitudes HTTP en donde su gran mayoría poseen estado fallido en su intento de ser atendidas por el

servidor de NGROK, valor que se puede corroborar en el porcentaje de error el cual es muy alto.

Tabla 4.4: Tabla de resultado del escenario 60 solicitudes

Métrica	Escenario: 60 HTTP Requests	Unidades
Mínimo de Latencia	434	ms
Máximo de Latencia	1262	ms
Media de Latencia	772	ms
Rendimiento	41.42	kb/s
Error	60.0	%

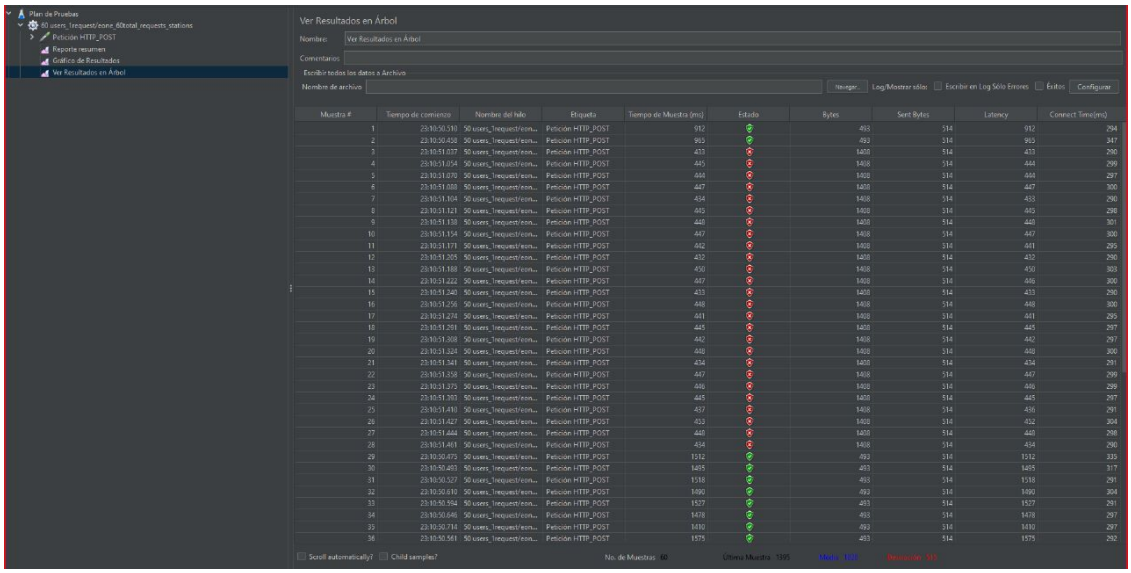


Figura 4.4: Detalle de envío de 60 peticiones simultáneas

Luego de realizarse las pruebas anteriormente presentadas, se logró medir la latencia y el rendimiento (*throughput*) durante las peticiones HTTP realizadas al servidor de NGROK, peticiones que simulaban el envío de datos medidos por parte de estaciones meteorológicas hacia el sistema implementado en LST.

Tal y como muestran los resultados de cada escenario, la latencia promedio en cada uno de ellos tuvo un valor elevado. Tener una latencia alta no es óptimo para un sistema que interactúa con dispositivos IoT, debido a que en una red IoT, cada uno de estos equipos que pueden ser sensores, recopilan y envían grandes volúmenes de datos, por lo cual, tener baja latencia es un factor primordial para estos sistemas [39].

Como primicia se planteó una hipótesis de la razón por la cual en todas las pruebas

se obtuvo como resultado latencia alta. Esta hipótesis contempla que es debido a la tecnología de tunel HTTP que ofrece NGROK. La conexión TCP que se establece entre el origen (ordenador de pruebas con software Jmeter) y el destino (servidor de NGROK) obedece a una ruta muy distante. La ruta por la que viajan los paquetes portadores de las banderas o ”flags” de SYN (*Synchronize*) y ACK (*Acknowledge*) comienzan en el ordenador de pruebas hacia los servidores de NGROK y finalmente hacia LST donde se encuentra levantado el sistema [40]. Sin embargo, al realizar unas pruebas de conectividad de redes, se descarta esta hipótesis y se comprueba que los valores de latencia altos obedecen a cómo está configurado la red de conectividad a internet de LST.

Como se puede apreciar en las figuras 4.5, 4.6, 4.7, al realizar pruebas de conectividad de red con líneas de comandos de consola en el servidor linux donde se encuentra implementado el sistema, se puede observar la traza de ruteo de paquetes para los dominios públicos de Google y Amazon, devuelve valores altos respecto a la latencia.

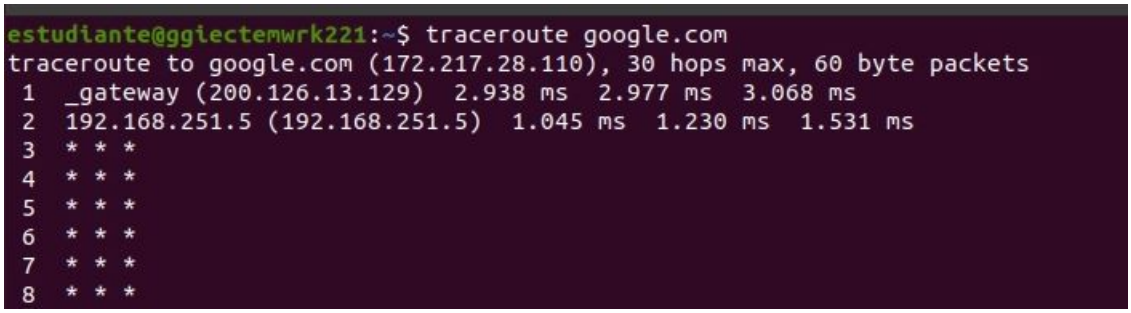


Figura 4.5: Shell linux, comando traceroute a destino www.google.com

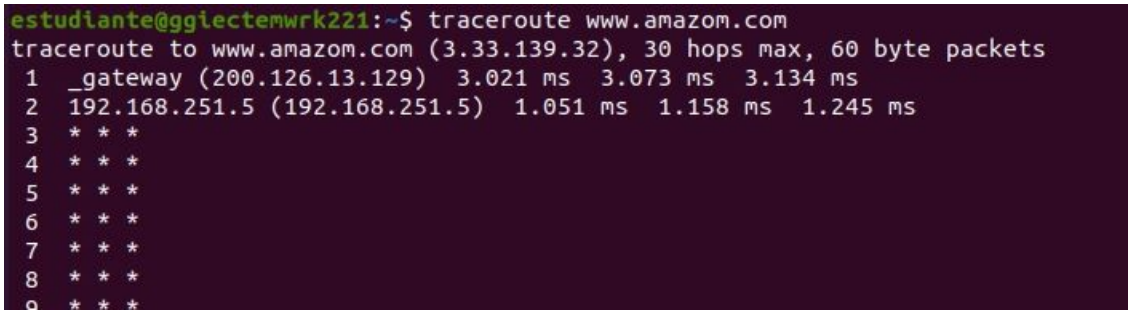


Figura 4.6: Shell linux, comando traceroute a destino www.amazon.com

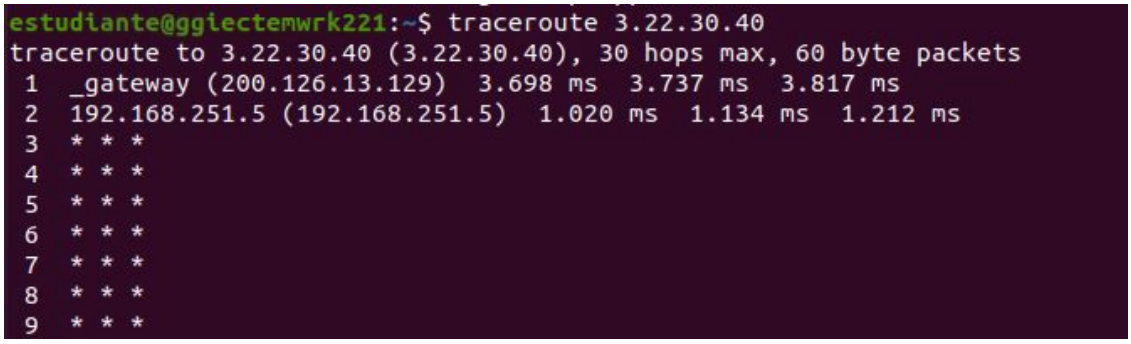


Figura 4.7: Shell linux, comando traceroute a destino servidor NGROK

Por otra parte, al realizar las mismas pruebas de conectividad de red en un ordenador diferente ubicado en una red LAN (Local Area Network) diferente, con las mismas líneas de comandos de consola como se muestra las figuras 4.8, 4.9, 4.10, los resultados obtenidos reflejan que la latencia es considerablemente baja en comparación al servidor linux de LST. Esto se debe a que los servidores DNS son diferentes en ambos casos, por lo cual, el configurar un servidor DNS adecuado provoca que el tiempo de respuesta con el dominio del servidor de NGROK sea menor y por consiguiente la latencia de los requerimientos HTTP también disminuirá [41].

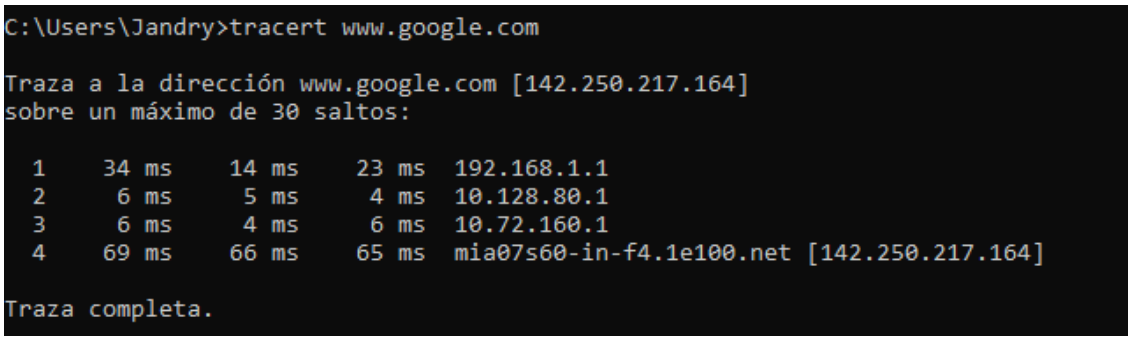


Figura 4.8: Consola de comandos windows, comando tracert a destino www.google.com

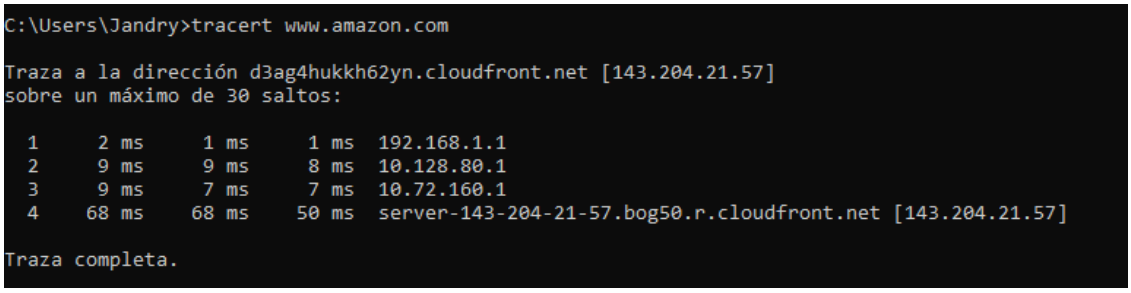


Figura 4.9: Consola de comandos windows, comando tracert a destino www.amazon.com

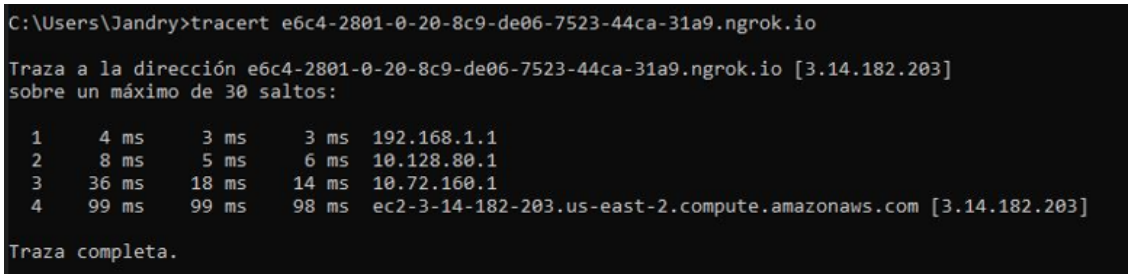


Figura 4.10: Consola de comandos windows, comando tracert a destino servidor NGROK

4.2 Análisis de satisfacción del cliente

Realizando una encuesta dirigida a 21 personas, entre estudiantes, profesores e investigadores, que conforman la muestra para la evaluación, se pretende analizar la aceptación, conformidad, retroalimentación de los usuarios frente a la propuesta que se ha presentado en este proyecto. La estructura con la que fue desarrollada la encuesta consta de 3 respuestas de opción múltiple: ”*Si*”, ”*No*” y ”*Otras*”; esta última es una casilla de respuesta abierta para un posible comentario. De esta manera, podremos conocer las diferentes posturas de los usuarios y concluir acerca de la satisfacción de los clientes.

Como primera interrogante se presentó: *¿Le parece intuitivo la forma de registrar los dispositivos IoT? Déjenos su comentario en la casilla de texto abierto.* Los resultados observados en la figura 4.11 señalan que, el 84% de los encuestados, está de acuerdo que el método para registrar los dispositivos IoT es intuitivo siempre y cuando el usuario conozca acerca de la tecnología IoT; de otra forma, llenar las casillas de información de registro puede llegar a ser confuso y se requiera un soporte de atención para una explicación del funcionamiento de los formularios de registro de sensores, equipo principal y sitio de funcionamiento de los equipos.

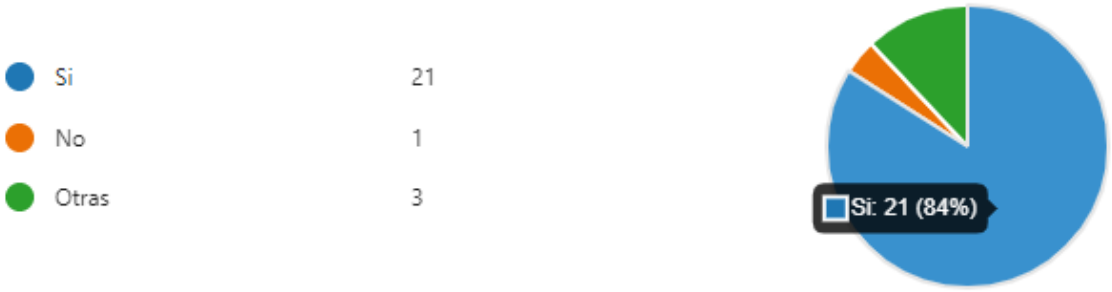


Figura 4.11: ¿Le parece intuitivo la forma de registrar los dispositivos IoT? Déjenos su comentario en la casla de texto abierto.

Luego de que los usuarios logren registrar sensores y equipos de manera exitosa, el proceso de funcionamiento del sistema indica que a través de la pestaña *Dashboard* se podrá visualizar los datos medidos en tiempo real. Por esta razón se realizó la segunda interrogante: *¿Le parece intuitivo la forma de interacturar con las gráficas que muestran la información medida por lo sensores? Déjenos su comentario en la casilla de texto abierto.* Los resultados obtenidos en la figura 4.12 exponen que, el 83% de los encuestados pueden interpretar e interactuar correctamente las gráficas presentadas en la plataforma, sin embargo, proponen que para una persona no tan experimentada en el campo estadístico, sería correcto explicar qué sucede en los picos altos y bajos de las gráficas.

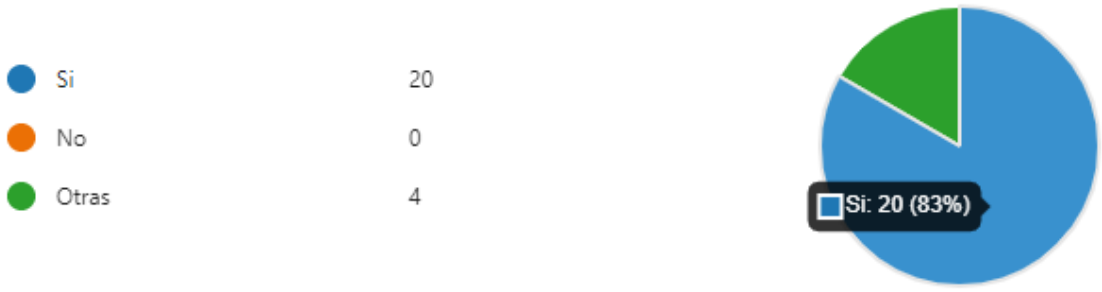


Figura 4.12: ¿Le parece intuitivo la forma de interacturar con las gráficas que muestran la información medida por lo sensores? Déjenos su comentario en la casilla de texto abierto.

Una vez que se obtienen los datos medidos en gráficas de unidad de medida vs tiempo, los usuarios tienen la opción de acceder a la pestaña *Endpoint de datos abiertos* donde se encuentra publicado el redireccionamiento URL o *endpoint* para obtener los datos medidos en otro formato. De esta manera se plantea la tercera interrogante: *¿Le parece adecuado la forma en en que los datos son publicados mediante una página web o endpoint? Déjenos su comentario en la casilla de texto abierto.* De acuerdo a las

valoraciones reflejadas en la figura 4.13, el 60% de los encuestados considera que es adecuada y formal la manera en la que se presenta el portal web introductorio y donde se informa qué estaciones están ligadas a la propuesta para el posterior redireccionamiento a su *endpoint* de consumo de datos. Además, de acuerdo a la retroalimentación obtenida, sería acorde especificar si la estación se encuentra en una fase *demo* o es la estación oficial de CENAIM.

Figura 4.13: ¿Le parece adecuado la forma en en que los datos son publicados mediante una página web o endpoint? Déjenos su comentario en la casilla de texto abierto.

Para finalizar, los usuarios pueden ingresar a la URL donde se presentan los datos medidos en formato tabla, además de una breve introducción de la entidad que está colaborando con la propuesta. Es así que, se realiza la última interrogante: *¿Le parece agradable y formal la interfaz donde se presentan los datos medidos por las estaciones metereológicas? Déjenos su comentario en la casilla de texto abierto.* La retroalimentación de los usuario, de acuerdo a la figura 4.14, muestra que el 46% de los encuestados, sostiene que la interfaz presenta de forma correcta los datos medidos en el nuevo formato establecido. Por otro lado, también mencionan que sería mejor elegir otros colores para combinación del fondo y mejorar la calidad de la imagen mostrada.

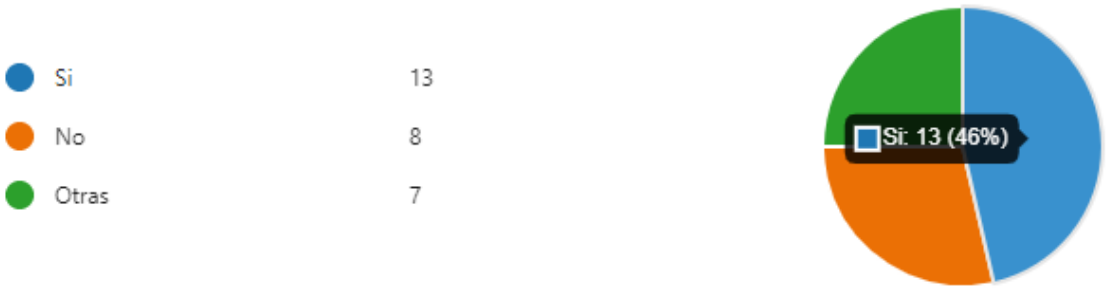


Figura 4.14: ¿Le parece agradable y formal la interfaz donde se presentan los datos medidos por las estaciones metereológicas? Déjenos su comentario en la casilla de texto abierto.

4.3 Análisis de costos

Esta sección contiene los valores a considerar en desarrollo e implementación del proyecto. Esto cubre costos de *Hardware, Software y Humanware* tanto en el desarrollo de los scripts y conexión de los dispositivos hacia la API de Haystack.

La tabla 4.5 representa la inversión en equipos de Hardware para la implementación del sistema propuesto descrito en la sección 2.2.

Tabla 4.5: Tabla de materiales: *Hardware*

Descripción	Cantidad	Costo Unitario	Total
Placa Arduino UNO	1	$20	$20
Placa Raspberry Pi 3	1	$60	$60
Placa Base Shield	1	$14	$14
Valor Total:			**$94**

Por otra parte, la tabla 4.6 detalla los programas necesarios y utilizados para el desarrollo de scripts y realización de pruebas de rendimiento para el sistema. En su mayoría se ha utilizado software libre o sus versiones gratuitas y no caducables. Sin embargo, la adquisición de estos programas no influye en el costo final de la propuesta.

Tabla 4.6: Tabla de materiales: *Software*

Descripción	Cantidad	Costo Unitario	Total
GNU/Linux	1	-	-
Python	1	-	-
Arduino	1	-	-
Apache JMeter	1	-	-
Valor Total:			-

Finalmente, los costos de producción en el desarrollo de software se calculan mediante la tabla salarial del Ministerio de Trabajo [1]. La tabla 4.7 muestra los pagos a considerar por mano de obra que incluye el desarrollo de los scripts necesarios para operabilidad del sistema, tanto en lenguaje Arduino y Python para las placas

[1] https://www.ecuadorlegalonline.com/laboral/tabla-de-salarios-minimos-sectoriales/

respectivamente. La tabla 4.8 muestra el costo total del proyecto, valorado en $204, el cual incluye implementación e instalación de los dispositivos, desarrollo de srcipts para su funcionamiento, soporte técnico y manual de usuario.

Tabla 4.7: Tabla salarial

Trabajador	Rol	Total
Programador	Desarrollo y Conectividad	$110
Valor Total:		$110

Tabla 4.8: Tabla costos finales

Recursos	Costo Total
Hardware	$94
Software	-
Humanware	$110
Valor Total:	*$204*

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Implementar sistemas alternos de análisis, procesamiento y visualización de datos es sin duda una gran propuesta referente a plataformas que cumplen las mismas funciones, pero es necesario acceder a una membresía para gozar de las mismas. La recolección automática de datos representa cierta dificultad al momento de integrar los equipos de mediciones debido a que obedecen a algoritmos matemáticos fundamentados por los mismos proveedores, señales digitales o análogas según la naturaleza del componente electrónico de medición y además al formato de intercambio de datos con el cuál son transferidos a través de una red.

La arquitectura tecnológica del sistema implementado destaca por lo primordial que es elegir los canales o protocolos de comunicación que se utilizan para el intercambio de información entre todos los componentes, desde los sensores hasta los microordenadores. Los pulsos eléctricos que envían los equipos de medición responden a cambios de estado entre *alto y bajo*, característica que define a una señal digital. Por otro lado, el protocolo *UART* permitió establecer un canal de comunicación entre la placa de *Arduino Uno* con rol de transmisor y la placa *Raspberry Pi 3* con rol de receptor. Estos datos transmitidos obedecen a una comunicación serial en donde, mediante los puertos de comunicación *RX* y *TX*, conectados de manera cruzada, permite establecer el envío y recepción de los datos hacia el gateway *IoT* (*Raspberry Pi 3*).

El análisis y procesamiento de los datos alojados en el gateway *IoT* fue posible gracias al desarrollo de *scripts* en lenguaje de programación *Python*. Estos *scripts* se encargan de inicializar las funciones anteriormente mencionadas mediante el uso de librerías que facilitan la apertura de puertos de comunicación serial, decodificar la información y darle

un formato *JSON* para su posterior envío a un servicio web realizando peticiones *HTTP* a la *API* del *back-end*.

La aplicación web o plataforma de visualización de datos implementada en un servidor del LST presentó un correcto funcionamiento sin importar cuantos equipos *IoT* se deseen registrar. Todos los datos recibidos en esta plataforma son representados en gráficos individuales para cada variable medida. Gracias al estándar de etiquetado para dispositivos IoT llamado *Haystack*, todo el proceso de registro de estaciones metereológicas fue sencillo debido al correcto procesamiento de la información realizada por el gateway IoT.

El desarrollo de endpoints de datos abiertos consiste en presentar una dirección URL pública o redireccionamiento en el *front-end* de la aplicación web. Estos endpoints ofrecen una descripción de la finalidad del desarrollo de esta aplicación web, así mismo información de quiénes estan colaborando con el proyecto y finalmente toda la información recibida de los sensores es presentada en un formato de tabla.

La tecnología de sistemas en la nube que ofrece *NGROK* enfocada a los equipos de borde o *"edge"* facilitó la conectividad entre dispositivos *IoT* desplegados en cualquier parte del mundo y la plataforma o aplicación web desarrollada en un entorno de productividad local. La creación de un túnel *HTTP* posibilita que la URL de la *API* con su método *POST* sea pública a través de la internet.

El rendimiento obtenido por parte del sistema, utilizando esta tecnología del software de *NGROK*, presenta resultados positivos en cuanto a la latencia y el *throughput*. Pese a que se puede interpretar que por tener una latencia alta el sistema no puede cumplir con las demandas del cliente, esto no es asi ya que en un escenario real los dispositivos *IoT* enviarán los datos de manera asíncrona, por lo cual todos los requerimientos *HTTP* van a ser atendidos dentro de la inmediatez posible. Por otro lado, el *throughput* indica que a medida de que aumente la cantidad de registro de dispositivos *IoT*, gracias a la escalabilidad de *NGROK*, la capacidad de recepción de datos va a aumentar en la misma medida.

5.2 Recomendaciones

- El voltaje de los sensores utilizados es una característica fundamental en el despliegue del sistema propuesto, pues es necesario verificar el correcto funcionamiento de cada uno de estos dispositivos al momento de medir su variable establecida. Un sensor en mal estado perjudica la lectura de los datos del ambiente.

- En cuanto a la conectividad de las placas Arduino y Raspberry, otra opción viable es el uso del microcontrolador *ESP32*, el cual incluye tecnologías Bluetooth y Wi-Fi. Siendo esta última necesaria para el envío de los datos hacia la plataforma de visualización.

- Es necesario realizar una actualización de los paquetes instalados en la placa Raspberry Pi 3 para su correcta utilización e instalación de nuevas librerías necesarias en la ejecución de los scripts para este poryecto.

- Se necesita utilizar *Contrab*, el cual es un archivo de texto especial, diseñado para ser leído por *CRON* y proceder con la ejecución de scripts en segundo plano, esto para evitar que el usuario deba acudir al administrador.

- Verificar la procedencia de cada una de las librerías de Arduino para los diferentes sensores implementados en el proyecto. Debido a que existen sensores que miden pulsos de señales (digitales) o funcionan como un potenciómetro, midiendo la variable en un rango determinado (analógicos).

5.3 Líneas Futuras

La propuesta descrita en este proyecto esta ligada a las aplicaciones de IoT en entornos domésticos e industriales, por lo cual en la Internet existe un amplio repositorio referente a nuevos proyectos o documentos investigativos en donde el *IoT* sea un actor principal. Durante la realización de este trabajo experimental surgieron varias ideas de posibles mejoras o trabajos futuros que se pueden aplicar para optimizar el sistema implementado. Entre las ideas que se pueden abordar se presentan las siguientes:

- Diseñar un *front-end* distinto para la visualización de los datos de manera más intuitiva con el usuario, donde se visualicen gráficos a través del tiempo de cada una de las variables, así como también gráficos de calibración (o *"gauge"* por su traducción al inglés) dentro del rango permitido por cada uno de los sensores utilizados en el despligue del presente proyecto.

- Implementar el servicio de registros IoT dentro de un servidor dedicado dentro de la universidad con el objetivo de que se encuentre dentro de un dominio público y pueda ser accedido desde cualquier punto donde se encuentren los estudiantes, profesores, ingenieros, u otros.

- Los algoritmos de predicción juegan un papel escencial dentro de este sistema, ya que mediante estos podemos anticiparnos a futuros cambios climáticos que afecten o beneficien a los otros entornos, como por ejemplo: sector agrícola, control de riesgos, navegación marina, entre otros.

- La realización de algoritmos de correlación entre las variables metereológicas, para establecer si existen alguna relación entre ellas y qué tipo de relación mantienen.

BIBLIOGRAFÍA

[1] E. P. Yadav, E. A. Mittal, and H. Yadav, “Iot: Challenges and issues in indian perspective,” in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pp. 1–5, 2018.

[2] T. Mendieta, J. Herrera, and A. Jimenez Peña, “La capacidad del iot de transformar el futuro,” *Revista Avenir*, vol. 1, pp. 15–18, sep. 2019.

[3] F. E. Espinoza Moran, Aldrin Marcos y Herrera Yerovi, “Estación meteorológica de la escuela superior naval cmdte. “rafael morán valverde”,” vol. 1, no. 1, p. 22, 2020.

[4] F. Ureña Elizondo, “Utilización de estaciones meteorológicas automáticas como nueva alternativa para el registro y transmisión de datos,” *Posgrado y Sociedad Revista Electrónica del Sistema de Estudios de Posgrado*, vol. 11, p. 33–49, sep. 2017.

[5] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, “Internet of things: Evolution and technologies from a security perspective,” *Sustainable Cities and Society*, vol. 54, p. 101728, 2020.

[6] E. B. Villamil, “Desarrollo de estación meteorológica con iot, redundante a fallos y para condiciones ambientales de alta montaña,” vol. 1, no. 1, pp. 12–14, 2021.

[7] M. Rico Vañó, *Integración de plataformas para el desarrollo de una aplicación IoT de riego inteligente*. PhD thesis, Universitat Politècnica de València, 2022.

[8] V. V. R. G. Saigopal and V. Raju, “Iot digital forensics and major security issues,” in *2020 International Conference on Computational Intelligence (ICCI)*, pp. 233–236, 2020.

[9] Y. Zhao and N. Lu, “Research and implementation of data storage backup,” in *2018 IEEE International Conference on Energy Internet (ICEI)*, pp. 181–184, 2018.

[10] E. A. Q. Montoya, S. F. J. Colorado, W. Y. C. Muñoz, and G. E. C. Golondrino, “Propuesta de una arquitectura para agricultura de precisión soportada en iot,” *Revista Ibérica de Sistemas e Tecnologias de Informação*, no. 24, pp. 39–56, 2017.

[11] V. Puranik, Sharmila, A. Ranjan, and A. Kumari, “Automation in agriculture and iot,” in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pp. 1–6, 2019.

[12] K. A. Patil and N. R. Kale, “A model for smart agriculture using iot,” in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 543–545, 2016.

[13] D. L. Cuasquer Chiscueth, “Aplicación de la tecnología iot (internet of things) para la medición de variables meteorológicas en la agricultura sostenible: aplicación del iot en sensores de bajo costo que monitorean las variables de precipitación, temperatura y humedad ambiental para optimizar el uso del recurso hídrico en la agricultura sostenible.,” vol. 1, no. 1, 2022.

[14] T. Manglani, A. Vaishnav, A. S. Solanki, and R. Kaushik, “Smart agriculture monitoring system using internet of things (iot),” in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, pp. 501–505, 2022.

[15] N. Kumar, A. K. Dahiya, K. Kumar, and S. Tanwar, “Application of iot in agriculture,” in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 1–4, 2021.

[16] R. Dagar, S. Som, and S. K. Khatri, “Smart farming – iot in agriculture,” in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 1052–1056, 2018.

[17] W. Z. Khan, M. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, “Industrial internet of things: Recent advances, enabling technologies and open challenges,” *Computers & Electrical Engineering*, vol. 81, p. 106522, 2020.

[18] M. H. ur Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, “The role of big data analytics in industrial internet of things,” *Future Generation Computer Systems*, vol. 99, pp. 247–259, 2019.

[19] D. Salcedo, D. Suarez, J. Solano, and C. Henriquez, “Iot motors sistema inteligente para para la gestión automática de un generador eléctrico basado en la arquitectura del iot,” *Computer and Electronic Sciences: Theory and Applications*, vol. 1, no. 1, pp. 1–10, 2020.

[20] C. A. Icaza Guamba, “Análisis, diseño y desarrollo de un sistema inteligente y automatizado de monitoreo y control de cultivos con iot. caso de estudio: huertos urbanos,” B.S. thesis, Quito: UCE, 2021.

[21] H. Duque Gómez, “Arquitecturas inteligentes para gestión de sistemas ciberfísicos en ambientes iot,” 2020.

[22] D. Bolatti, M. J. Karanik, C. Todt, R. J. R. Scappini, and S. D. Gramajo, “Sistema inteligente de detección de anomalías para iot,” in *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*, 2021.

[23] “The dual effects of the internet of things (iot): A systematic review of the benefits and risks of iot adoption by organizations,” *International Journal of Information Management*, vol. 51, p. 101952, 2020.

[24] A. Roshdy, N. Sharaf, M. Saad, and S. Abdennadher, “Generic data visualization platform,” in *2018 22nd International Conference Information Visualisation (IV)*, pp. 56–57, 2018.

[25] S. Royo-Montañés and A. Benítez-Gómez, “Portales de datos abiertos. metodología de análisis y aplicación a municipios españoles,” *El profesional de la información (EPI)*, vol. 28, no. 6, 2019.

[26] ESPOL, “Granja experimental de espol, espacio que beneficia a estudiantes, investigadores y agricultores del país,” *noticias*, vol. 1, no. 1, p. 1, 2019.

[27] K. Okamoto, T. Ohhashi, M. Asakura, and K. Watanabe, “A digital anemometer,” *IEEE Transactions on Instrumentation and Measurement*, vol. 43, no. 2, pp. 116–120, 1994.

[28] O. Omoruyi, S. N. John, O. Chinonso, O. Robert, A. A. Adewale, and K. O. Okokpujie, “Wireless sensor network for rainfall measurement using a tipping bucket

rain gauge mechanism," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 740–744, 2017.

[29] A. Argeseanu, E. Ritchie, and K. Leban, "New low cost structure for dual axis mount solar tracking system using adaptive solar sensor," in *2010 12th International Conference on Optimization of Electrical and Electronic Equipment*, pp. 1109–1114, IEEE, 2010.

[30] Y. A. Ahmad, T. Surya Gunawan, H. Mansor, B. A. Hamida, A. Fikri Hishamudin, and F. Arifin, "On the evaluation of dht22 temperature sensor for iot application," in *2021 8th International Conference on Computer and Communication Engineering (ICCCE)*, pp. 131–134, 2021.

[31] A. Hasibuan, A. Qodri, M. Isa, *et al.*, "Temperature monitoring system using arduino uno and smartphone application," *Bulletin of Computer Science and Electrical Engineering*, vol. 2, no. 2, pp. 46–55, 2021.

[32] Johnny Novillo-Vicuña, Dixys Hernández Rojas, Bertha Mazón Olivo, Jimmy Molina Ríos, Oscar Cárdenas Villavicencio, *Arduino y el Internet de las cosas*. 2018.

[33] C. Sun, F. Zheng, G. Zhou, and K. Guo, "Design and implementation of cloud-based single-channel lora iiot gateway using raspberry pi," in *2020 39th Chinese Control Conference (CCC)*, pp. 5259–5263, 2020.

[34] Eric Peña, Mary Grace Legaspi, "Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter," vol. 54, no. 4, pp. 1–5, 2020.

[35] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "Iot architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*, pp. 731–738, 2016.

[36] K. Praghash, V. D. S. Eswar, J. Y. Roy, A. Alagarsamy, and S. Arunmetha, "Tunnel based intra network controller using ngrok framework for smart cities," in *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 39–43, 2021.

[37] B. Pilco and E. García, “Integración de módulos iot con sistemas hardware-software para edificios inteligentes mediante la estandarización de datos,” *IEEE*, p. 21, 2022.

[38] P. K. Aggarwal, P. S. Grover, and L. Ahuja, “A performance evaluation model for mobile applications,” in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pp. 1–3, 2019.

[39] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, “An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment,” *PloS one*, vol. 14, no. 11, p. e0224934, 2019.

[40] A. A. Anto and D. Ogi, “Implementation of hybrid password authentication scheme based on shape-text on raspberry pi as a client-server-based access control system to overcome shoulder surfing attack,” in *2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI)*, pp. 207–212, 2019.

[41] E. F. Noviani, B. Kembara, B. A. Yudha Pratama, D. A. Permata Sari, A. M. Shiddiqi, and B. J. Santoso, “Performance analysis of aws and gcp cloud providers,” in *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, pp. 236–241, 2022.

APÉNDICES

A Código Arduino para lectura de datos

```
#include "TimerOne.h" // Timer Interrupt set to 2 second for read sensors
#include <math.h>
#include "DHT.h"

//PINOUT
#define WindSensorPin (2) // The pin location of the anemometer sensor
#define WindVanePin (A4) // The pin the wind vane sensor is connected to
#define VaneOffset 0; // define the anemometer offset from magnetic north
#define interruptPin (3) // The pin location of the rain collector sensor
#define DHTPIN (7)  // Digital pin connected to the DHT sensor
#define SOLAR (A0) // Digital pin connected to solar sensor

#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//-----------------------------------------------------------------------------

//VARIABLES
int VaneValue; // raw analog value from wind vane
int Direction; // translated 0 - 360 direction
int CalDirection; // converted value with offset applied
int LastValue; // last direction value

volatile bool IsSampleRequired; // this is set true every 2.5s. Get wind speed
volatile bool IsCollected;
volatile unsigned int TimerCount; // used to determine 2.5sec timer count
volatile unsigned long Rotations; // cup rotation counter used in interrupt routine
volatile unsigned long ContactBounceTime; // Timer to avoid contact bounce in isr

float WindSpeed; // speed miles per hour

const int interval = 500;
volatile unsigned long tiptime = millis();

//-----------------------------------------------------------------------------

DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  LastValue = 0;
  IsSampleRequired = false;
  TimerCount = 0;
  Rotations = 0; // Set Rotations to 0 ready for calculations
  Serial.begin(9600);
  dht.begin();

  pinMode(WindSensorPin, INPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  pinMode(SOLAR, INPUT);
  analogReference(EXTERNAL);
  attachInterrupt(digitalPinToInterrupt(WindSensorPin), isr_rotation, FALLING);
  attachInterrupt(digitalPinToInterrupt(interruptPin), count, FALLING);

  // Setup the timer interupt
  Timer1.initialize(500000);// Timer interrupt every 2.5 seconds
  Timer1.attachInterrupt(isr_timer);
}

void loop() {

  delay(10000);
  getWindDirection();

  int radsolar = analogRead(SOLAR);
  float outputSolar = 3.30 / 1023.0 * radsolar;

  float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temperature = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float fahrenheit = dht.readTemperature(true);

  // Only update the display if change greater than 5 degrees.
  if(abs(CalDirection - LastValue) > 5) {
    LastValue = CalDirection;
  }
```

```
if (isnan(humidity) || isnan(temperature) || isnan(fahrenheit)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
}

// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(fahrenheit, humidity);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(temperature, humidity, false);

if(IsSampleRequired == true || IsCollected == true) {
  // convert to mp/h using the formula V=P(2.25/T)
  // V = P(2.25/2.5) = P * 0.9
  WindSpeed = Rotations * 0.9;
  Rotations = 0; // Reset count for next sample

  IsSampleRequired = false;
  IsCollected = false;

  //PRINT OF DATA
  Serial.print("{\"id\": \"base\"");
  Serial.print(", ");
  Serial.print("\"WindSpeed[MPH]\": ");
  Serial.print(WindSpeed);
  Serial.print(", ");
  Serial.print("\"Knots\": ");
  Serial.print(getKnots(WindSpeed));
  Serial.print(", ");
  Serial.print("\"Direction\": \"");
  Serial.print(CalDirection);
  getHeading(CalDirection);
  Serial.print("\", ");
  Serial.print("\"Strength\": \"");
  getWindStrength(WindSpeed);
  Serial.print("\", ");
  Serial.print("\"Humidity[%]\": ");
  Serial.print(humidity);
  Serial.print(", ");
  Serial.print("\"Temperature[C]\": ");
```

```
    Serial.print(temperature);
    Serial.print(", ");
    Serial.print("\"Temperature[F]\": ");
    Serial.print(fahrenheit);
    Serial.print(", ");
    Serial.print("\"Heat Index[C]\": ");
    Serial.print(hic);
    Serial.print(", ");
    Serial.print("\"Heat Index[F]\": ");
    Serial.print(hif);
    Serial.print(", ");
    Serial.print("\"Voltage[mV]\": ");
    Serial.print(outputSolar * 1000.00);
    Serial.print(", ");
    Serial.print("\"Watts[W/m2]\": ");
    Serial.print((int)(outputSolar * 1000.00 / 1.67));
    Serial.println("}");
  }
}

void count() {
  // Grab the current ms count for common calculations
  unsigned long curtime = millis();

  // Make sure we don\"t record bounces
  if ((curtime - tiptime) < interval) {
    return;
  }

  // How long since the last tip?
  unsigned long tipcount = curtime - tiptime;
  tiptime = curtime;

  // Calculate mm/hr from period between cup tips
  double rainrate = 914400.0 / tipcount;

  IsCollected = true;

  //Serial.print("Cup tip: ");
```

```
  Serial.print("{\"id\": \"base\"");
  Serial.print(", ");
  Serial.print("\"Cup tip[ms]\": ");
  Serial.print(tipcount);
  Serial.print(", ");

  //Serial.print("Rain rate: ");
  Serial.print("\"Rain Rate[mm/hr]\": ");
  Serial.print(rainrate);
  Serial.println("}");
}

// isr handler for timer interrupt
void isr_timer() {

  TimerCount++;

  if(TimerCount == 6)
  {
    IsSampleRequired = true;
    TimerCount = 0;
  }
}

// This is the function that the interrupt calls to increment the rotation count
void isr_rotation() {

  if((millis() - ContactBounceTime) > 15 ) { // debounce the switch contact.
    Rotations++;
    ContactBounceTime = millis();
  }
}

// Convert MPH to Knots
float getKnots(float speed) {
  return speed * 0.868976;
}

// Get Wind Direction
```

```
void getWindDirection() {

  VaneValue = analogRead(WindVanePin);
  Direction = map(VaneValue, 0, 1023, 0, 359);
  CalDirection = Direction + VaneOffset;

  if(CalDirection > 360)
    CalDirection = CalDirection - 360;

  if(CalDirection < 0)
    CalDirection = CalDirection + 360;

}

// Converts compass direction to heading
void getHeading(int direction) {

  if(direction < 22)
  Serial.print(" N");
  else if (direction < 67)
  Serial.print(" NE");
  else if (direction < 112)
  Serial.print(" E");
  else if (direction < 157)
  Serial.print(" SE");
  else if (direction < 212)
  Serial.print(" S");
  else if (direction < 247)
  Serial.print(" SW");
  else if (direction < 292)
  Serial.print(" W");
  else if (direction < 337)
  Serial.print(" NW");
  else
  Serial.print(" N");
}

// converts wind speed to wind strength
void getWindStrength(float speed) {
```

```
  if(speed < 2)
  Serial.print("Calm");
  else if(speed >= 2 && speed < 4)
  Serial.print("Light Air");
  else if(speed >= 4 && speed < 8)
  Serial.print("Light Breeze");
  else if(speed >= 8 && speed < 13)
  Serial.print("Gentle Breeze");
  else if(speed >= 13 && speed < 18)
  Serial.print("Moderate Breeze");
  else if(speed >= 18 && speed < 25)
  Serial.print("Fresh Breeze");
  else if(speed >= 25 && speed < 31)
  Serial.print("Strong Breeze");
  else if(speed >= 31 && speed < 39)
  Serial.print("Near Gale");
  else
  Serial.print("RUN");
}
```

B Código Python utilizado en el *gateway IoT*

```
import serial
import time
import requests
import json

arduino = serial.Serial()
arduino.baudrate = 9600
arduino.port = "/dev/ttyUSB0"
arduino.open()

lista = []
print("Initializing gateway IoT.....")

url = "https://094b-2801-0-20-8c9-ba79-d8d-2df4-2d56.ngrok.io/v1/registrar-datos"
```

```
headers = {'Content-type': 'application/json', 'Accept': '*/*'}
print("Running instructions.....")

while True:
   getData = arduino.readline()
   dataString = getData.decode("utf-8")
   data = dataString[:-2]
   data1 = json.dumps(data)
   jsondata = json.loads(data1)
   page = requests.post(url, jsondata, headers=headers)
   print(jsondata)
   print("Sending data.....")
```