

# **ESCUELA POLITÉCNICA NACIONAL**

**FACULTAD DE INGENIERÍA ELÉCTRICA Y  
ELECTRÓNICA**

**IMPLEMENTACIÓN DE UN SISTEMA PROTOTIPO PARA  
LLEVAR A CABO PRONÓSTICOS METEOROLÓGICOS  
A CORTO PLAZO EN LAS ZONAS DE CUMBAYÁ,  
CALDERÓN Y CONOCOTO, BASADO  
EN UNA RED NEURONAL**

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE  
INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**

**COLLAGUAZO IPO OSCAR DANIEL**

**DIRECTOR: ING. PABLO ANIBAL LUPERA MORILLO, PhD.**

**Quito, febrero 2022**

## **AVAL**

Certifico que el presente trabajo fue desarrollado por Collaguazo Ipo Oscar Daniel, bajo mi supervisión.

---

**PhD. Pablo Aníbal Lupera Morillo**  
**DIRECTOR DEL TRABAJO DE TITULACIÓN**

## **DECLARACIÓN DE AUTORÍA**

Yo, Oscar Daniel Collaguazo Ipo declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

---

OSCAR DANIEL COLLAGUAZO IPO

## **DEDICATORIA**

Lleno de regocijo al culminar una valiosa, importante y anhelada meta en mi vida, quiero dedicar este proyecto con todo mi amor a:

Dios por brindarme sabiduría, fortaleza y perseverancia para recorrer con positivismo este camino día a día hasta llegar a la meta deseada y poder culminarla con éxito.

A mis padres Lucía y Álvaro, por su paciencia y amor infinito, por su apoyo incondicional y por ser el motivo para no rendirme cuando el camino se tornaba difícil, por ser la fuerza para levantarme y seguir luchando.

A mis pequeñas Melanie y Eliza por ser la alegría en momentos difíciles, por su cariño y por ser el motivo de salir adelante.

A mis tíos, hermanos y familia en general, por siempre brindarme muestras de apoyo para seguir luchando por mis sueños y metas anheladas.

A mi novia Gaby, por su apoyo incondicional, por su amor y por siempre estar a mi lado a pesar de todos los errores que como ser humano cometo.

A mi tía, por su respeto, admiración y apoyo, aunque no esté conmigo físicamente siempre estará presente en mi corazón.

## **AGRADECIMIENTO**

Agradezco infinitamente a Dios por todas las bendiciones recibidas, por guiar mis pasos, por darme la paciencia, la fortaleza y el coraje de seguir adelante para cumplir las metas propuestas.

A mis padres por todo su apoyo, por siempre estar para mí cuando más lo necesito, por ser mis guías y por nunca dejar de creer en mí.

A toda mi familia por el respeto, cariño, apoyo y motivación para continuar cumpliendo metas.

A mi director de tesis, PhD. Pablo Lupera por el apoyo brindado, la paciencia y confianza para la culminación de este proyecto.

Al Ing. Ricardo Llugsí por la ayuda brindada en todo el proceso de desarrollo del trabajo de titulación, por los conocimientos brindados y por su confianza.

A todas las personas que amablemente colaboraron para realizar con éxito este proyecto.

# ÍNDICE DE CONTENIDO

AVAL .....	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	IV
ÍNDICE DE CONTENIDO.....	V
RESUMEN .....	IX
ABSTRACT .....	X
1. INTRODUCCIÓN.....	1
1.1. OBJETIVOS .....	2
1.2. ALCANCE .....	2
1.3. MARCO TEÓRICO.....	3
1.3.1. COMPORTAMIENTO CLIMÁTICO EN QUITO .....	3
1.3.1.1. Clima en los Valles de Quito .....	4
1.3.2. METEOROLOGÍA.....	5
1.3.2.1. El tiempo atmosférico.....	5
1.3.2.2. Parámetros meteorológicos .....	6
1.3.2.3. Pronóstico del tiempo.....	8
1.3.3. ESTACIONES METEREOLÓGICAS .....	9
1.3.3.1. CWS (Conventional Weather Station, Estación Meteorológica Convencional).....	9
1.3.3.2. AWS (Automatic Weather Station, Estación Meteorológica Automática) .....	10
1.3.3.3. Componentes y requerimientos para las estaciones meteorológicas automáticas .....	12
1.3.4. REDES NEURONALES.....	25
1.3.4.1. Neurona o nodo .....	26
1.3.4.2. Tipos de redes neuronales.....	31
1.3.4.3. Redes neuronales recurrentes (RNN).....	32
1.3.4.4. Redes neuronales convolucionales.....	37
1.3.4.5. Modelo ARIMA.....	38
1.3.4.6. Entrenamiento de redes neuronales .....	40
1.3.4.7. Backpropagation en el entrenamiento de redes neuronales .....	42
1.3.4.8. Deep Learning (Aprendizaje Profundo).....	45

2. METODOLOGÍA.....	47
2.1. REQUERIMIENTOS DEL SISTEMA.....	47
2.1.1. HARDWARE DE AWS.....	48
2.1.2. NOTIFICACIONES DE EVENTOS.....	49
2.1.3. ALMACENAMIENTO DE INFORMACIÓN.....	49
2.1.4. APLICACIÓN DE ESCRITORIO Y ACCESO REMOTO.....	50
2.2. ARQUITECTURA DEL SISTEMA.....	50
2.2.1. DISEÑO DEL SISTEMA PROTOTIPO.....	50
2.2.1.1. Etapa de adquisición de datos.....	51
2.2.1.2. Etapa de transmisión y almacenamiento de datos.....	52
2.2.1.3. Etapa de tratamiento y predicción de datos.....	52
2.2.1.4. Etapa de visualización y monitoreo de datos.....	52
2.3. SELECCIÓN DE COMPONENTES.....	53
2.3.1. SELECCIÓN DE HARDWARE.....	53
2.3.1.1. Sensor de temperatura y humedad relativa.....	53
2.3.1.2. Sensor de presión.....	54
2.3.2. SELECCIÓN DE SOFTWARE.....	55
2.3.2.1. Acceso remoto.....	55
2.4. INSTALACIÓN Y CONFIGURACIÓN DE EQUIPOS.....	56
2.4.1. ETAPA DE ADQUISICIÓN DE DATOS.....	56
2.4.1.1. Sistema Operativo en Raspberry Pi.....	56
2.4.1.2. Sensor de Temperatura y Humedad DHT22.....	60
2.4.1.3. Sensor de Presión Atmosférica BMP180.....	62
2.4.2. ETAPA DE TRANSMISIÓN Y ALMACENAMIENTO DE DATOS.....	64
2.4.2.1. Instalación Apache.....	64
2.4.2.2. Instalación PHP.....	65
2.4.2.3. Instalación MySQL.....	66
2.4.2.4. Instalación de phpmyadmin.....	67
2.4.2.5. Creación de base de datos y usuarios en Hosting Web.....	69
2.4.2.6. Acceso remoto a la base de datos de Hosting Web.....	72
2.4.2.7. Instalación FTP.....	73
2.4.3. ETAPA DE TRATAMIENTO DE DATOS.....	73
2.4.3.1. Instalación de Pandas.....	74
2.4.3.2. Instalación de Numpy.....	74

2.4.3.3.	Instalación Anaconda .....	74
2.4.3.4.	Instalación de Keras.....	76
2.4.4.	ETAPA DE VISUALIZACIÓN Y MONITOREO DE DATOS .....	77
2.4.4.1.	Instalación de TeamViewer .....	77
2.4.4.2.	Tkinter para crear aplicaciones .....	79
2.4.4.3.	Configuración para envío de notificaciones.....	80
2.5.	REPRESENTACIÓN E IMPLEMENTACIÓN DE SCRIPTS .....	81
2.5.1.	LECTURA DE DATOS.....	81
2.5.2.	CHEQUEO DE DATOS .....	85
2.5.3.	INTERPOLACIÓN .....	87
2.5.4.	ALMACENAMIENTO EN BASE DE DATOS .....	91
2.5.5.	ALMACENAMIENTO EN SERVIDOR FTP.....	92
2.5.6.	APLICACIÓN DESARROLLADA PARA EL REGISTRO Y MANIPULACIÓN DE DATOS .....	95
2.5.7.	REDES NEURONALES.....	106
2.5.7.1.	Modelo ARIMA .....	106
2.5.7.2.	LSTM Simple.....	109
2.5.7.3.	LSTM Apilado.....	116
2.5.8.	AUTOMATIZACIÓN DE SCRIPTS .....	118
2.5.9.	CONFIGURACIÓN DE REDES WIFI .....	119
2.5.10.	CREACIÓN DE ACCESO DIRECTO A LA APLICACIÓN .....	120
3.	RESULTADOS Y DISCUSIÓN .....	123
3.1.	POSICIONAMIENTO DE LAS AWS.....	123
3.2.	CONSTRUCCIÓN E INSTALACIÓN DE LAS AWS.....	124
3.2.1.	CONSTRUCCIÓN DE CASETA METEOROLÓGICA.....	124
3.2.2.	INSTALACIÓN DE EQUIPOS.....	125
3.2.3.	INSTALACIÓN DE AWS EN VIVIENDAS.....	126
3.3.	PRUEBAS DE FUNCIONAMIENTO.....	127
3.3.1.	PRUEBAS DE ADQUISICIÓN DE DATOS.....	127
3.3.2.	PRUEBAS DE TRANSMISIÓN Y ALMACENAMIENTO DE DATOS.....	130
3.3.3.	PRUEBAS DE MONITOREO Y VISUALIZACIÓN DE DATOS.....	133
3.4.	CALIBRACIÓN DE SENSORES .....	138
3.5.	PRUEBAS DE LA PREDICCIÓN DE LOS PARÁMETROS METEOROLÓGICOS.....	147



3.5.1. MODELO ARIMA.....	147
3.5.2. LSTM SIMPLE.....	153
3.5.3. LSTM APILADA.....	157
3.6. PRESUPUESTO REFERENCIAL DE LOS EQUIPOS DEL SISTEMA ..	162
4. CONCLUSIONES Y RECOMENDACIONES.....	164
4.1. CONCLUSIONES.....	164
4.2. RECOMENDACIONES .....	165
5. REFERENCIAS BIBLIOGRÁFICAS .....	167
ANEXOS .....	173
ANEXO A. Funcionamiento del código de aplicación.....	173
ANEXO B. Guía de utilización de la aplicación.....	173
ANEXO C. Modelo de calibración de las AWS. ....	173
ANEXO D. Código de notificación de reinicio. ....	173
ANEXO E. Código de lectura de datos. ....	173
ANEXO F. Código de chequeo de datos.....	173
ANEXO G. Código de almacenamiento de lecturas en base de datos. ....	173
ANEXO H. Código de interpolación de datos.....	173
ANEXO I. Código de almacenamiento de archivos en servidor FTP. ....	173
ANEXO J. Código de aplicación. ....	173

## RESUMEN

En la ciudad de Quito, no todo el territorio presenta las mismas condiciones meteorológicas, por tanto, el estudio de los microclimas es un aspecto importante a considerar para la predicción precisa del clima, sin embargo, la instalación de equipos o estaciones meteorológicas en diversas zonas de la ciudad resulta muy costosa, el presente proyecto tiene como finalidad implementar un prototipo de sistema para llevar a cabo pronósticos meteorológicos de temperatura y humedad relativa a corto plazo en las zonas de Cumbayá, Calderón y Conocoto, en base al uso de una red neuronal cuya información de entrada será obtenida desde estaciones meteorológicas automáticas (AWS) de bajo costo.

En la actualidad, existen varias herramientas de hardware y software disponibles en el mercado de muy buenas prestaciones y bajo costo para el diseño e implementación de una AWS robusta y eficiente, que permita el monitoreo continuo de datos meteorológicos, para obtener un historial del comportamiento climático en las diferentes zonas, información que permitirá desarrollar un modelo de pronósticos del tiempo a corto plazo basado en la utilización del modelo de predicción ARIMA y redes neuronales como: LSTM simple y LSTM apilado (2 capas); con el fin de obtener el modelo con mayor rendimiento, precisión y confiabilidad en los pronósticos de temperatura y humedad relativa.

Finalmente, se obtuvo los errores más bajos con la red neuronal LSTM simple, determinando que las mejores predicciones se dan en las localidades con mayor estabilidad climática y continuidad en la adquisición de datos.

**PALABRAS CLAVE:** Parámetros meteorológicos, AWS, redes neuronales, ARIMA, LSTM.

## **ABSTRACT**

In the city of Quito, not all the territory presents the same meteorological conditions, therefore, the study of microclimates is an important aspect to consider for the precise prediction of the climate, however, the installation of equipment or meteorological stations in various areas. The purpose of this project is to implement a prototype system to carry out short-term meteorological forecasts of temperature and relative humidity in the areas of Cumbayá, Calderón and Conocoto, based on the use of a neural network. whose input information will be obtained from low-cost automatic weather stations (AWS).

Currently, there are several hardware and software tools available on the market with very good performance and low cost for the design and implementation of a robust and efficient AWS, which allows continuous monitoring of meteorological data, to obtain a history of climate behavior in the different areas, information that will allow the development of a short-term weather forecast model based on the use of the ARIMA prediction model and neural networks such as: simple LSTM and stacked LSTM (2 layers); in order to obtain the model with the highest performance, precision and reliability in temperature and relative humidity forecasts.

Finally, the lowest errors were obtained with the simple LSTM neural network, determining that the best predictions occur in localities with greater climatic stability and continuity in data acquisition.

**KEYWORDS:** Meteorological parameters, AWS, neural networks, ARIMA, LSTM.

# 1. INTRODUCCIÓN

El clima es un aspecto determinante para la organización de las actividades cotidianas de las personas, y dependiendo del área geográfica de una ciudad puede variar drásticamente, en el caso de la ciudad Andina de Quito y su accidentada geografía el comportamiento climático es diferente, es decir, cada sector posee un clima en particular (microclima), por tanto, el fenómeno de los microclimas no permite llevar a cabo un pronóstico preciso del tiempo en zonas extensas.

El ente encargado de la recolección y predicción de datos meteorológicos en Ecuador y en la ciudad de interés Quito en la actualidad es el INAMHI (Instituto Nacional de Meteorología e Hidrología) quien a través de REMMAQ (Red Metropolitana de Monitoreo Atmosférico de Quito) adquiere datos a través de las diferentes estaciones meteorológicas y en conjunto con potentes programas informáticos que ejecutan complejos modelos físico–matemáticos pronostican el estado futuro de la atmósfera.

Sin embargo, debido al fenómeno de los microclimas antes descrito es necesario el despliegue de varias estaciones meteorológicas en la ciudad según las normativas expedidas por la WMO (*World Meteorological Organization*, Organización Mundial de Meteorología), lo que implica costos muy elevados de adquisición, ubicación e instalación, montos de alrededor de USD 5000 y USD 15000 de inversión individual por estación, es decir, la adquisición de información de parámetros climáticos es muy compleja y costosa.

En el presente proyecto se plantea implementar un prototipo de sistema para llevar a cabo pronósticos de datos meteorológicos de temperatura y humedad relativa en base a una red neuronal LSTM (*Long-Short Term Memory*, Memoria a Corto y Largo Plazo) en las zonas de Calderón, Conocoto y Cumbayá, por falta de equipos de monitoreo meteorológico en estos sectores, el sistema va a ser alimentado con información de micro estaciones meteorológicas automáticas, implementadas y configuradas con ordenadores e instrumentos de bajo costo disponibles en el mercado, debido principalmente a las grandes ventajas que presentan como la portabilidad, conectividad a internet, la capacidad de adquisición y transmisión de datos en tiempo real.

## 1.1. OBJETIVOS

El objetivo general de este Proyecto técnico es: Implementar un prototipo de sistema para llevar a cabo pronósticos meteorológicos de temperatura y humedad relativa a corto plazo en las zonas de Cumbayá, Calderón y Conocoto, en base al uso de una red neuronal LSTM cuya información de entrada será obtenida desde micro estaciones meteorológicas automáticas de bajo costo.

Los objetivos específicos de este Proyecto técnico son:

- a. Investigar los fundamentos teóricos para el desarrollo e implementación de las estaciones meteorológicas automáticas de bajo costo; y, revisar la teoría relacionada a redes neuronales recurrentes para la implementación de sistemas básicos de pronóstico del clima.
- b. Realizar las configuraciones y programación necesarias para desarrollar una micro estación meteorológica para la adquisición de datos, generar una aplicación en la estación meteorológica para el monitoreo remoto de datos y eventos.
- c. Implementar un sistema básico para llevar a cabo el pronóstico de datos meteorológicos de temperatura y humedad relativa a corto plazo en base a una red neuronal.
- d. Construir una infraestructura adecuada para la protección de la micro estación meteorológica automática y adquisición adecuada de datos.
- e. Instalar y realizar las pruebas de funcionamiento de los prototipos de AWSs en las zonas predeterminadas expuestas a las condiciones climatológicas de cada sitio.

## 1.2. ALCANCE

Se implementarán prototipos de estaciones meteorológicas automáticas (AWS) de bajo costo capaces de tomar datos de temperatura, humedad relativa y presión atmosférica que serán instaladas en las zonas de Calderón, Conocoto y Cumbayá de acuerdo con las normativas de la WMO (*World Meteorological Organization*, Organización Mundial de Meteorología). Los datos serán almacenados en archivos csv (comma-separated values, valores separados por comas) cada minuto y en una base de datos cada diez minutos mediante la conexión a la red local de cada sitio a instalar. Los archivos serán almacenados en un servidor FTP (*File Transfer Protocol*, Protocolo de Transferencia de Archivos) en la nube al final de cada día.

Las estaciones tendrán la capacidad de enviar notificaciones de los siguientes tipos de eventos: reinicio al detectar un cero en la toma de datos, por manipulación física o por corte de energía, interpolación de datos al detectar datos faltantes al final del día, error de ceros persistentes y datos no detectados debido al fallo en la toma de datos de los sensores o software instalado, estos eventos serán almacenados en una base de datos para su respectivo monitoreo con lo cual se dará solución a los problemas que acontezcan en ellas.

Contarán con una aplicación de escritorio con varias funciones como: la búsqueda y visualización de la información almacenada en la base de datos y en los archivos csv, tendrá la capacidad de exportar datos a archivos de formato tipo Excel, también se tendrá la posibilidad de reiniciar la estación de forma inmediata o programarla para días determinados.

Las estaciones contarán con un host de acceso remoto para ingresar al sistema de cada nodo y monitorear los datos y eventos que ocurren en la estación a través de la aplicación de escritorio, con ello se permite al administrador de los dispositivos realizar cambios en las estaciones de manera sencilla para la implementación de futuras funciones.

Mediante los archivos con los datos obtenidos y almacenados en los archivos csv en el servidor FTP a diario, se obtendrá un historial del comportamiento climático de cada zona, con las cuales se realizará un modelo de pronósticos meteorológicos a corto plazo de temperatura y humedad relativa en base a redes neuronales LSTM.

La finalidad del proyecto es implementar y distribuir estaciones AWS de bajo costo en los tres valles cercanos a la ciudad de Quito; con dichas estaciones se adquirirán y transmitirán datos en tiempo real de acuerdo a las condiciones ambientales de cada sitio, se realizará el monitoreo continuo de datos y eventos que ocurren en la estación, mediante una aplicación de escritorio a través del acceso remoto y el envío de alertas de correo electrónico. Los datos recolectados por las estaciones se procesarán mediante redes neuronales LSTM que permitirán emitir pronósticos de temperatura y humedad relativa en los sectores mencionados.

### **1.3. MARCO TEÓRICO**

#### **1.3.1. COMPORTAMIENTO CLIMÁTICO EN QUITO**

El DMQ (Distrito Metropolitano de Quito) ubicado a 2850 msnm al norte de la Sierra en Ecuador, presenta diversidad de topoclimas o microclimas que son comportamientos

variados de parámetros meteorológicos como la temperatura, humedad, precipitación, nubosidad; entre otros, que pueden presentarse en un mismo día en zonas separadas por distancias cortas, este comportamiento se da principalmente por la ubicación de la ciudad, el relieve y otros factores que afectan al cambio atmosférico de las zonas, por ejemplo, la vegetación, los vientos que se producen, condiciones geomorfológicas <sup>1</sup>y sobre todo el nivel de intervención humana sobre la naturaleza, que en su efecto conjunto da origen a los denominados microclimas.

### **1.3.1.1. Clima en los Valles de Quito**

Un valle es una zona extensa y plana ubicada entre montañas o elevaciones naturales, los valles del DMQ por lo general están ubicados a las afueras de la ciudad, por ende, son conocidos como barrios o parroquias rurales, sin embargo, estas zonas tienen mucha demanda en cuanto a adquisición o compra de lotes y terrenos, principalmente se debe a que poseen un clima cálido, en este trabajo se describen; de forma breve tres zonas del DMQ: Cumbayá, Calderón y Conocoto.

- **Cumbayá**

Cumbayá se localiza en el Oriente de la ciudad de Quito a una altura de 2355 msnm y se conoce como: el valle de Cumbayá-Tumbaco. Cumbayá posee un microclima privilegiado, su clima es subtropical (cálido-seco) que llega hasta 32°C en verano, mientras que en las noches más frías de invierno baja hasta 6°C, se mantiene una temperatura media entre 14.1°C-17.3°C, por lo que esta parte del valle es la más abrigada, con excelentes cultivos de guabas, cítricos, aguacates, chirimoyas, hortalizas, pastos, maíz y hasta caña de azúcar [1].

- **Calderón**

Calderón se localiza en el Norte de la ciudad de Quito a una altura de 2610msnm, está caracterizada por numerosos ambientes microclimáticos, determinados ampliamente por la altitud y por las condiciones geomorfológicas locales [2], formado principalmente por suelos arenosos y cangagua, razón por la cual tiene un aspecto desértico.

El clima de Calderón está definido como clima templado y seco, lo cual limita el desarrollo agrícola, su temperatura media es de 21,7°C. Las lluvias son muy escasas por la falta de

---

<sup>1</sup> **Condiciones geomorfológicas.** – Son las formaciones o relieves de la superficie terrestre.

elevaciones próximas, ya que se encuentra sobre una meseta ubicada en el centro de la Hoya de Guayllabamba, a gran distancia de las Cordilleras [2].

- **Conocoto**

Conocoto cuyo nombre significa en lengua quecha “Loma abrigada”, se localiza en el Valle de los Chillos al Sureste de la ciudad de Quito a una altura de 2530 msnm, generalmente cuenta con un clima templado con un promedio de 16°C, esta zona goza de mantener lluvias dentro de un rango normal<sup>2</sup>, en su totalidad territorial mantiene una precipitación que oscila entre 1000mm-2000mm anuales [3].

### **1.3.2. METEOROLOGÍA**

Es la ciencia que estudia el estado del tiempo atmosférico, de los fenómenos que se producen en la atmósfera, así como de las causas y leyes que los rigen, en un plazo de tiempo cronológico determinado [4], su estudio se centra principalmente en los fenómenos que ocurren en la capa baja de la atmósfera (tropósfera) y sus variaciones en un tiempo corto, entre las que se encuentran fenómenos como cambios de temperatura, lluvias, vientos, entre otros.

Los parámetros usados por la meteorología para estudiar las variaciones atmosféricas son: temperatura, humedad, presión, nubosidad, precipitación; entre otras, con el objetivo de determinar las mejores entradas de datos para predecir o pronosticar dichos parámetros en un lazo de tiempo corto. Sin embargo, cada zona posee comportamientos diferentes con respecto a las variables atmosférica, en el caso de la ciudad de Quito los cambios son notables en cada área, por tanto, su pronóstico se dificulta.

#### **1.3.2.1. El tiempo atmosférico**

Se define como el estado en que se encuentra la atmósfera en un determinado lugar y momento [5], es decir, es la fluctuación de las variables atmosféricas en un tiempo dado, de manera empírica se dice que el clima está frío, caluroso o lluvioso, en determinadas horas en un sitio en el que nos encontramos, sin embargo, los conceptos entre tiempo atmosférico y clima son distintos.

El clima por su parte utiliza datos históricos generados por alrededor de treinta años de estudio para establecer un patrón climático de una determinada zona, mientras que el

---

<sup>2</sup> **Rango normal.** – Precipitaciones producidas en un rango de >500mm y < 3000mm anuales.



tiempo atmosférico es un estado cambiante e instantáneo y de cierto modo irrepetible [6], por tanto, estas variaciones repentinas afectan en general al pronóstico del tiempo. El despliegue de estaciones meteorológicas para la adquisición y tratamiento adecuado de datos de forma continua permite minimizar el error que se puede producir en las predicciones en distintas zonas.

### **1.3.2.2. Parámetros meteorológicos**

La caracterización del clima en un determinado lugar se debe a variables atmosféricas o parámetros meteorológicos como: temperatura, humedad, presión, precipitación, entre otros; condicionados por los factores climáticos descritos con anterioridad. Además, se debe comprender que los elementos climáticos que se describen a continuación son variables que pueden cambiar en tiempos cortos o en un mismo día y algunos de ellos son los que se estudiarán en el proyecto.

#### *1.3.2.2.1. Temperatura*

Es una de las propiedades básicas del aire y de gran importancia para la vida, la temperatura es un indicador del nivel de agitación molecular, es decir, es el parámetro que representa el grado de calentamiento del aire [7], donde a mayor agitación molecular mayor calentamiento; puede ser representada en tres diferentes escalas las cuales son: °C (Celsius o grados centígrados), °F (grados Fahrenheit) o k (Kelvin) y se mide mediante un termómetro.

Esta variable meteorológica depende de dos factores principales: la continentalidad que depende de la cercanía de un lugar a las grandes masas de agua que en su efecto actúan como reguladores naturales del clima, y la altitud que a medida que aumenta produce que la temperatura disminuya. En meteorología la temperatura del aire se define como: el valor indicado por un termómetro expuesto al aire en un lugar aislado de la radiación directa al sol [8].

#### *1.3.2.2.2. Humedad*

Se encuentra representada por la cantidad de vapor de agua que existe en el aire, la misma que depende de varios factores como: el nivel al que se realiza la medición, la existencia de lluvia y, vegetación en el sitio, etc. [9].

Existen diferentes conceptos de humedad como: absoluta, específica y relativa, sin embargo, en este estudio nos centramos en la humedad relativa [10].

- **Humedad relativa.** – Representa la relación entre la cantidad de vapor de agua existente y saturado para un mismo grado de temperatura y presión atmosférica [11], es decir es la cantidad de vapor de agua presente en el aire y se representa en porcentaje (%).

La humedad se puede medir a través de higrómetros, los cuales permiten determinar la humedad relativa del ambiente al cual esté expuesto, sin embargo, este aparato es muy utilizado en varios campos de investigación a pesar de no ser tan precisos como otros aparatos como el psicómetro.

#### 1.3.2.2.3. *Presión atmosférica*

Las personas a pesar de no percibir que el aire tiene peso, este ejerce una fuerza sobre la superficie terrestre, esta fuerza es representada por unidad de superficie y a la cual se la conoce como presión atmosférica [11], y se mide a través del barómetro. La presión atmosférica puede ser representada mediante tres unidades las cuales son: mmHg (milímetros de mercurio), atm (atmósferas) o Pa (Pascuales), y en donde se tiene que la relación existente entre estas es:  $1\text{atmósfera}=760\text{mmHg}=101300\text{Pa}$ .

La presión atmosférica depende de la altitud de la zona, a mayor altitud menor será la columna de aire y oxígeno en la atmósfera, por ende, la fuerza que ejerce el peso del aire será menor, por ello las personas al escalar zonas montañosas de gran altura experimentan náuseas, mareos y falta de aire, mientras que en las zonas costera o bajas la columna y peso del aire es mayor, por tanto, la presión atmosférica es mayor.

#### 1.3.2.2.4. *Precipitación*

La precipitación es la cantidad de agua líquida o sólida que cae sobre la superficie terrestre [11], el nivel de precipitación se mide a través de la cantidad de lluvia que recogen los aparatos meteorológicos como los pluviómetros en determinado periodo de tiempo que puede ser en un día, mes o año, etc., sus unidades son  $\text{l/m}^2$  (litros por metro cuadrado) y se puede expresar también en mm (milímetros).

### **1.3.2.3. Pronóstico del tiempo**

Los pronósticos del tiempo que pueden ser vistos a través de diversos dispositivos electrónicos como: celulares, televisiones, radio; computadores, son previsiones de datos meteorológicos que, de acuerdo al estudio de las condiciones atmosféricas de un determinado lugar y a través, de ciertos indicios se conoce información de un futuro a corto o largo plazo de los parámetros meteorológicos. Sin embargo, las predicciones no son acertadas en muchos de los casos, lo que se puede deber a las fluctuaciones de las condiciones atmosféricas sin tener indicios previos.

#### *1.3.2.3.1. Pronóstico del tiempo en Quito*

Las predicciones del tiempo están sujetas a estrictas normas expedidas por la OMM y vigiladas por esta misma organización, sin embargo, las organizaciones nacionales son las encargadas de las observaciones y predicciones de datos meteorológicos en cada país, en el caso de Ecuador es el INAMHI (Instituto Nacional de Meteorología e Hidrología), quien es la entidad técnico-científica regulada por la WMO responsable de generar y difundir información hidrometeorológica para la formulación y evaluación de los planes de desarrollo nacional y locales.

En la actualidad un pronóstico del tiempo se basa clásicamente en el uso de un modelo físico que computa la información de las CWS (*Conventional Weather Station*, Estaciones Meteorológicas Convencionales), las AWS (*Automatic Weather Station*, Estaciones Meteorológicas Automáticas) y las imágenes del GOES (*Geostationary Operational Environmental Satellite*, Satélite Ambiental Operativo Geoestacionario) [12]; El INAMHI por su parte utiliza dos modelos numéricos de sistemas para la predicción del estado de la atmósfera en diferentes períodos o escalas de tiempo como son: WRF (*Weather Reserch and Forecasting*, Investigación y Predicción Meteorológica) y MM5 (*Fifth-Generation Mesoscale Model*, Modelo de Mesoescala de Quinta Generación) [13].

- **WRF**

Es un sistema de cálculo numérico desarrollado gracias al aporte de diversas instituciones de investigación atmosférica en Estados Unidos, el software de WRF permite generar predicciones numéricas y producir simulaciones basadas en observaciones de variables atmosféricas reales o variables ficticias [14].

- **MM5**

El sistema de mesoescala MM5 desarrollado por la Universidad Estatal de Pensilvania y NCAR (*National Center for Atmospheric Research* – Centro Nacional de Investigación Atmosférica), es un sistema que permite realizar pronósticos de precipitación en tiempo real [13].

A pesar de que el INAMHI tiene desplegadas varias estaciones alrededor del país cumpliendo con las normas expedidas por la OMM, las observaciones de parámetros meteorológicos en la ciudad Andina de Quito son escasas porque solo se cuenta con 3 estaciones [15]: Iñaquito (Norte de la ciudad), Izobamba (Sur) y La Tola (Valle de Cumbayá, en las afueras de Quito), y el establecimiento de las condiciones iniciales es crucial y en el caso de Quito la escases de información de variables meteorológicas podría llevar a pronósticos incorrectos [12].

### **1.3.3. ESTACIONES METEREOLÓGICAS**

Una estación meteorológica es un lugar, sitio o infraestructura conformada por instrumentos de medición capaz de adquirir y registrar las variaciones de los diferentes parámetros meteorológicos que caracterizan el comportamiento atmosférico en una determinada zona de estudio.

Las estaciones meteorológicas se utilizan en múltiples sectores tanto industriales o de investigación, por tanto, existen diversos tipos de estaciones que se clasifican por sus características y finalidad, el tipo de dato que adquieren o según el lugar donde se encuentren ubicadas: sea mar, tierra o aire, sin embargo, según la tecnología se pueden mencionar dos tipos: CWS (*Conventional Weather Station*, Estación Meteorológica Convencional) y AWS (*Automatic Weather Station*, Estación Meteorológica Automática) o EMA por sus siglas en español.

#### **1.3.3.1. CWS (Conventional Weather Station, Estación Meteorológica Convencional)**

Una estación meteorológica convencional (CWS) o conocida como estación tradicional, son aquellas estaciones con la capacidad de adquirir datos y almacenarlos, este sistema no posee un método de transmisión de información, por tanto, es necesario la intervención humana para la recopilación y procesamiento de la información captada por los instrumentos de medición, en la Figura 1.1 se muestra un ejemplo de CWS.



**Figura 1. 1.** Estación Meteorológica Convencional [16].

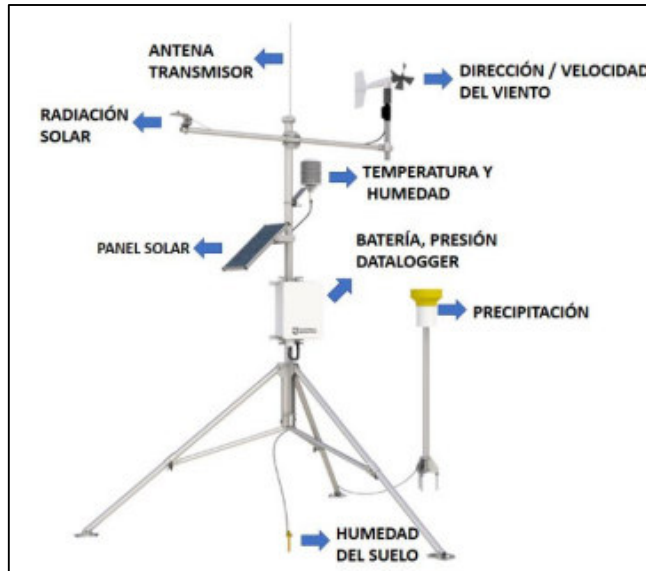
Las CWS están conformadas por instrumentos modulares o independientes, es así como en un momento de fallo de un aparato este no interfiere en el funcionamiento del resto de componentes, no obstante, en ciertos casos los dispositivos necesitan de complejos adaptadores y convertidores analógico/digital, por ende, el costo de adquisición e instalación de la estación es mayor [17].

Existen varios inconvenientes que se pueden producir en la CWS, uno de estos es tener horarios establecidos para el registro de las condiciones meteorológicas, la información no se actualizará de forma periódica y se pueden perder datos que pudieron ser de eventos críticos, otro error es la subjetividad del observador y el error de digitación [17].

#### **1.3.3.2. AWS (Automatic Weather Station, Estación Meteorológica Automática)**

Las estaciones meteorológicas automáticas (AWS) son aquellas estaciones conformadas por hardware y software capaces de adquirir, registrar, procesar, transmitir y acceder a la información de las diferentes variables meteorológicas de forma automática en tiempo real de una determinada zona de estudio, es decir, no necesitan de la intervención humana para su funcionamiento.

Los componentes de hardware de una AWS son aquellos dispositivos físicos compactos o circuitos integrados que permiten la adquisición y almacenamiento de datos del medio ambiente en el que se encuentran tales como: sensores, microprocesadores, minicomputadores, data logger, entre otros; encargados de controlar los procesos de adquisición, almacenamiento, procesamiento y transmisión de datos, como se muestra en la Figura 1.2.



**Figura 1. 2.** Estación Meteorológica Automática [18]

Los componentes de software son los sistemas operativos, aplicaciones y los diferentes programas ejecutables que permiten la adquisición, registro, procesamiento y visualización de la información de los diferentes parámetros meteorológicos propios de la zona de estudio.

Muchas de las estaciones meteorológicas están basadas en microcomputadores en los cuales la adquisición, procesamiento y transmisión de datos están controlados por códigos de programación, por tanto, las estaciones se pueden adaptar con facilidad y pueden procesar grandes cantidades de información, con ello la calidad y velocidad es superior respecto a las CWSs [17], generalmente, estas estaciones utilizan sensores de salida digital y se encuentran fácilmente en el mercado, eso significa que el costo es extremadamente menor al igual que el consumo de potencia en comparación a una estación convencional, con el objetivo de obtener información de los parámetros meteorológicos en lugares poco habitados o de difícil acceso.

A continuación, en la Tabla 1.1 se presentan las características de las estaciones meteorológicas y las ventajas que tiene las AWS respecto a las CWS.

**Tabla 1. 1.** Comparación entre AWS y CWS.

Características	CWS	AWS
Requieren de personal altamente capacitado.	Si	No
Costoso.	Si	No

Instalación en cualquier zona de estudio.	No	Si
Compuesto de instrumentos modulares o independientes.	Si	No
Baja probabilidad de fallas.	Si	Si
Operación con bajo consumo de potencia.	No	Si
Se compone de unidades de transmisión y recepción de datos.	No	Si
Almacenamiento continuo de variables meteorológicas.	No	Si
Disponibilidad y actualización de datos en tiempo real.	No	Si
Monitoreo remoto y aplicaciones de visualización de información meteorológica.	No	Si
Notificaciones de alertas por fallas presentadas en las estaciones.	No	Si

### **1.3.3.3. Componentes y requerimientos para las estaciones meteorológicas automáticas**

A continuación, se describen los componentes y requerimientos de hardware, software e infraestructura; para la implementación de una estación autónoma que no dependa de la intervención humana y que su información sea confiable para el proceso de pronósticos meteorológicos.

#### **1.3.3.3.1. Hardware**

- **Sensores**

Se cuenta con un gran número de sensores inteligentes y disponibles en el mercado de bajo costo, que facilitan la adquisición y procesamiento de datos de cualquier variable que tenga por objetivo medir, proporcionan salidas digitales por tanto son sencillos de implementar y fáciles de usar con las diferentes plataformas. En este caso se describen tres tipos de sensores principales de temperatura y humedad relativa y presión atmosférica a utilizar para implementar las estaciones meteorológicas automáticas que, de acuerdo a lo expedido por la WMO en la Guía de Instrumentos y Métodos de Observación de

Variables Meteorológicas [11], los sensores deben cumplir requerimientos como: fiabilidad, estabilidad, robustez, incertidumbre, facilidad de calibración y mantenimiento.

✓ **Sensor de temperatura y Humedad relativa**

Este tipo de sensores inteligentes nos permite cuantificar o medir las condiciones de temperatura y humedad del aire circundante o del medio en el que se encuentra instalado.

❖ **Temperatura**

Los requerimientos para los sensores que miden el fenómeno físico de temperatura deben cumplir con los requerimientos descritos en la tabla 1.2, además, este tipo de dispositivo debe instalarse sobre una garita o infraestructura adecuada para protegerlo de la radiación directa del sol, la lluvia y el viento.

**Tabla 1. 2.** Requerimientos para la medición de temperatura [11].

Rango <sup>3</sup>	Ordinario	Máximo	Mínimo
	-30°C – 45°C	-30°C – 50°C	-40°C – 40°C
Tiempo de respuesta <sup>4</sup>	20 s		
Tiempo medio de obtención <sup>5</sup>	1 min		
Resolución <sup>6</sup>	0,1 °C		
Incertidumbre <sup>7</sup>	≤ -40°C	> -40°C y ≤40°C	> 40°C
	0,3°C	0,1°C	0,3°C
Tolerancia máxima <sup>8</sup>	±0,2°C		

❖ **Humedad**

Las condiciones de instalación en general de este dispositivo deben ser sobre una garita para protegerlo, además, es importante que el material de la infraestructura no sea capaz de absorber o perder vapor de agua ya que esto cambiaría las condiciones de este parámetro meteorológico, los requerimientos de la WMO para este parámetro se describen en la tabla 1.3.

<sup>3</sup> **Rango.** – Intervalo de valores de una determinada variable que es capaz de medir un instrumento.

<sup>4</sup> **Tiempo de respuesta.** – Corresponde al tiempo necesario para que un instrumento responda a una señal dada, conocido como tiempo de muestreo.

<sup>5</sup> **Tiempo medio de obtención.** – Es el tiempo mínimo promedio en que un instrumento debe adquirir un dato, según especificaciones técnicas.

<sup>6</sup> **Resolución.** – Es la variación mínima de la magnitud medida, es decir, es el valor mínimo perceptible entre 2 magnitudes continuas.

<sup>7</sup> **Incertidumbre.** – Es una medida cuantitativa que representa la calidad de la medición realizada.

<sup>8</sup> **Tolerancia máxima.** – Es el máximo error permitido, que un instrumento debe cometer en las mediciones, según las especificaciones técnicas del organismo regulador.



**Tabla 1. 3.** Requerimientos para la medición de humedad relativa [11].

<b>Rango</b>	0% – 100%
<b>Tiempo de respuesta</b>	20 segundos
<b>Tiempo medio de obtención</b>	1 minuto
<b>Resolución</b>	1%
<b>Incertidumbre</b>	± 3%

De acuerdo a los diferentes requerimientos provistos por la WMO y a las condiciones descritas en la tabla 1.2 y 1.3, se ha previsto el uso de tres diferentes tipos de sensores a ser considerados para implementar las estaciones meteorológicas automáticas como son: DHT11, DHT21, DHT22.

Estos sensores permiten cuantificar las variables de temperatura y humedad relativa en un solo dispositivo, están compuestos por un sensor capacitivo de humedad y un termistor, instrumentos con salidas digitales gracias a que incluyen en su diseño microprocesadores, compatibles con la plataforma Raspberry pi y fáciles de implementar, en la Tabla 1.4 se presenta las características de cada uno de los sensores seleccionados.

**Tabla 1. 4.** Comparaciones de sensores DHT11, DHT21 y DHT22 [19] [20].

<b>Especificaciones</b>	<b>DHT11</b>	<b>DHT21</b>	<b>DHT22</b>
<b>Voltaje de operación<sup>9</sup></b>	3V – 5V	3,5V – 5,5V	3V – 6V
<b>Tiempo de muestreo</b>	1s	2s	2s
<b>Interfaz digital</b>	Single bus	I <sup>2</sup> C	Single bus
<b>Temperatura</b>			
<b>Rango de temperatura</b>	0°C - 50°C	-40°C - 80°C	-40°C - 80°C
<b>Precisión</b>	±2°C	±0,5°C	±0,5°C
<b>Resolución</b>	0,1°C	0,1°C	0,1°C
<b>Humedad</b>			
<b>Rango de humedad</b>	20% - 90%	0% - 100%	0% - 100%
<b>Precisión<sup>10</sup></b>	5%	3%	2%
<b>Resolución</b>	1%	0,1%	0,1%

✓ **Sensor de presión atmosférica**

Este tipo de sensores permiten medir la fuerza que ejerce la masa de aire sobre una superficie en particular, las recomendaciones que debe seguir este tipo de sensores es que debe ser instalado en una atmósfera limpia y seca, exenta de sustancias corrosivas, evitando vibraciones y choques mecánicos.

<sup>9</sup> **Voltaje de operación.** – Voltaje mínimo que un dispositivo necesita para su funcionamiento y operación.

<sup>10</sup> **Precisión.** – Capacidad de un instrumento de dar el mismo resultado en mediciones diferentes, realizadas a las mismas condiciones.

Los requerimientos que deben cumplir los sensores se muestran en la tabla 1.4, los cuales permiten identificar los sensores digitales de presión barométricas a utilizar para la implementación de la AWS.

**Tabla 1. 5.** Requerimientos para la medición de presión atmosférica [11].

<b>Rango</b>	500 hPa – 1080 hPa
<b>Tiempo de respuesta</b>	20 segundos
<b>Tiempo medio de obtención</b>	1 minuto
<b>Resolución</b>	0,1 hPa
<b>Incertidumbre</b>	± 0,1 hPa

Se toma en consideración los siguientes tipos de sensores BMP180, BMP280, BME280, sensores que permiten no solo cuantificar la presión atmosférica sino también medir la variación de temperatura y en algunos modelos la humedad relativa simultáneamente.

Los modelos BMP180 y BMP280 permiten medir la altura respecto al nivel del mar, la presión atmosférica y la temperatura del medio al mismo tiempo, funcionan de acuerdo a la relación existente entre la presión del aire y la altitud, basadas en la tecnología BOSCH piezo-resistivas de alta precisión y bajo consumo de potencia [21], diseñado para su fácil implementación y manejo en microcontroladores ya que utilizan la comunicación I<sup>2</sup>C.

Por otro lado, el sensor BME280 tiene la capacidad de medir los tres parámetros meteorológicos como temperatura, humedad relativa y presión atmosférica simultáneamente, asimismo, posee características similares a los modelos antes descritos, a continuación, se presenta una comparación de estos dispositivos en la Tabla 1.6 con énfasis en la variable a medir.

**Tabla 1. 6.** Comparación de sensores BMP180, BMP280 y BME280 [21].

<b>Especificaciones</b>	<b>BMP180</b>	<b>BMP280</b>	<b>BME280</b>
<b>Voltaje de operación</b>	3,3V – 5V	1,8V – 3,3V	1,8V – 3,3V
<b>Rango de presión</b>	300hPa – 1100 hPa	300hPa – 1100 hPa	300hPa – 1100 hPa
<b>Precisión</b>	±1 hPa	±1 hPa	±1 hPa
<b>Resolución</b>	0,1 hPa	0,16 hPa	0,16 hPa
<b>Frecuencia de muestreo<sup>11</sup></b>	120 Hz	157 Hz	157 Hz
<b>Interfaz digital</b>	I <sup>2</sup> C	I <sup>2</sup> C	I <sup>2</sup> C / SPI

<sup>11</sup> **Frecuencia de muestreo.** - Número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta.

La fiabilidad de los equipos electrónicos para la implementación e instalación de una AWS es de suma importancia, sin embargo, se conoce que no todo sistema trabaja con una fiabilidad<sup>12</sup> del 100%, pero se debe asegurar que la probabilidad de falla del equipo o dispositivos electrónicos sea muy baja, por ende, la calibración de los equipos en entidades reguladas y en periodos que oscilan de 1 a 5 años de acuerdo al fabricante es esencial.

- **Microcontroladores**
  - ✓ **Raspberry Pi**

En la actualidad existen innumerables microcomputadoras que facilitan las tareas de recolección de datos o automatización de labores, uno de los dispositivos más utilizados es la Raspberry Pi, que presenta varias ventajas como la portabilidad, conectividad a internet, la capacidad de adquisición, transmisión y visualización de datos en tiempo real [12], además de su disponibilidad en el mercado y bajo costo, por estas razones el presente trabajo se basa en esta plataforma la cual se describe a continuación.

Raspberry Pi es un ordenador o computador de placa reducida conformada por un SoC (*System on Chip*, Sistema en Chip), CPU, memoria RAM, puertos de entrada y salida de audio y video, conectividad de red; entre otras características, como se presenta en la Figura 1.3, no obstante, es necesario conectar periféricos como, mouse, teclado y una pantalla para interactuar con el dispositivo.

Tiene la capacidad de albergar un sistema operativo basado en Linux Open Source o de código abierto con aplicaciones que permiten realizar tareas comunes como: manipulación de herramientas ofimáticas, o a su vez utilizarlo como servidor, este dispositivo posee puertos dedicados y pines para interactuar de forma directa con los múltiples sensores electrónicos que en conjunto con el software facilitan la implementación y uso de estos.

A pesar de encontrar varios modelos de Raspberry Pi en el mercado, el presente proyecto ha tomado en consideración la eficiencia energética, rentabilidad, velocidad y rendimiento de las placas, por ello se optó por la Raspberry Pi 4, quien posee muchas mejoras en las características de modelos antiguos a esta, los cuales se presentan en la Tabla 1.7.

---

<sup>12</sup> **Fiabilidad.** – Es el grado de estabilidad que se consigue en los resultados cuando se repite una medición en condiciones idénticas.

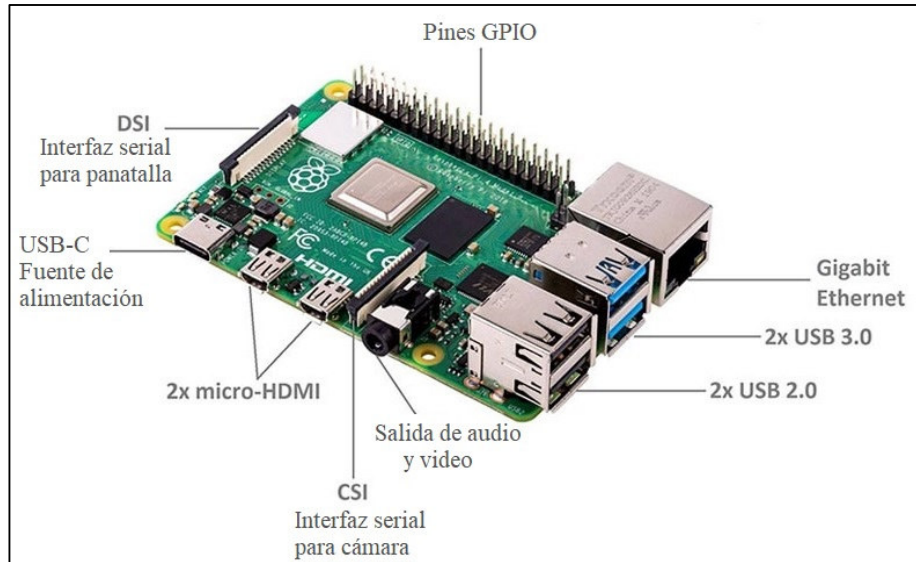


Figura 1. 3. Raspberry pi 4 modelo B [22].

Tabla 1. 7. Características de Raspberry Pi 4 modelo B [22] [23] [24].

Características	Especificaciones
<b>Procesador</b>	Microprocesador Broadcom BCM2711 de 64bits, un CPU ARM Cortex-A72 de cuatro núcleos BCM2837, núcleos ARM A72 de gran potencia a 1.5GHz, mejora el procesamiento en un 50% a Raspberry Pi 3B+.
<b>GPU (Graphics Processing Unit, Unidad de procesamiento de Gráficos.)</b>	Presenta mejoras con la nueva 3D Video Core VI que funciona hasta una frecuencia de 500MHz.
<b>Memoria RAM</b>	Cuenta con una LPDDR4-2400 SDRAM ( <i>Single Data Rate Synchronous Dynamic Random-Access Memory</i> , Memoria de Acceso Aleatorio Sincrónica y Dinámica), se encarga del rendimiento y gestión de las aplicaciones, tiene capacidad de 1 y 2 GB.
<b>Caché (3 tipos)</b>	<ul style="list-style-type: none"> <li>- Datos: 32KB</li> <li>- Caché de instrucciones L1: 48KB</li> <li>- Caché L2: 1MB</li> </ul>
<b>Tarjeta de red inalámbrica</b>	<ul style="list-style-type: none"> <li>- Wi-Fi 2,4GHz / 5GHz - IEEE 802.11.b/g/n/ac</li> <li>- Bluetooth 5.0, BLE (<i>Bluetooth Low Energy</i>, Bluetooth de Baja Energía)</li> </ul>
<b>GPIO (General Purpose Input/Output, Entrada / Salida de propósito General)</b>	<p>Cuenta con 40 pines de los cuales 26 son de propósito general para la comunicación con el medio exterior, los pines cumplen funciones de acuerdo a lo siguiente:</p> <ul style="list-style-type: none"> <li>- GPIO: 26 pines</li> <li>- GND: 8 pines</li> <li>- FUENTE (5V): 2 pines</li> <li>- FUENTE (3.3V): 2 pines</li> <li>- ID EEPROM: 2 pines</li> </ul>

	Esta distribución se puede observar en la Figura 1.4.
<b>USB</b>	USB 2.0: 2 puertos USB 3.0: 2 puertos
<b>Ethernet</b>	Puerto Gigabit Ethernet con capacidad de 10/100/1000 Mbps de transferencia, con soporte PoE ( <i>Power Over Ethernet</i> , Alimentación sobre Ethernet).
<b>HDMI (High Definition Multimedia Interface, Interfaz Multimedia de Alta Definición)</b>	Cuenta con 2 puertos HDMI con la capacidad de conectarse a pantallas o televisiones con resolución máxima de 4kp60.
<b>Puerto MIPI DSI</b>	DSI ( <i>Display Serial Interface</i> , Interfaz Serial para Pantalla), interfaz serial de alta velocidad que permite la conexión entre un módulo de pantalla y un procesador, reduce el número de pines y consumo de potencia, mejora el rendimiento y reduce la interferencia electromagnética.
<b>Puerto MIPI CSI</b>	CSI ( <i>Camera Serial Interface</i> , Interfaz Serial para Cámara), interfaz serial que permite la conexión entre una cámara y un procesador.
<b>Tarjeta SD</b>	Es recomendable utilizar una tarjeta SD ( <i>Secure Digital</i> , Seguro Digital) de mínimo 8 GB, aunque puede iniciarse con una SD superior a 256 GB.
<b>Alimentación</b>	La alimentación se realiza a través de un conector USB-C, una fuente de 5V con un mínimo de corriente de 2.5A y máximo 3A.

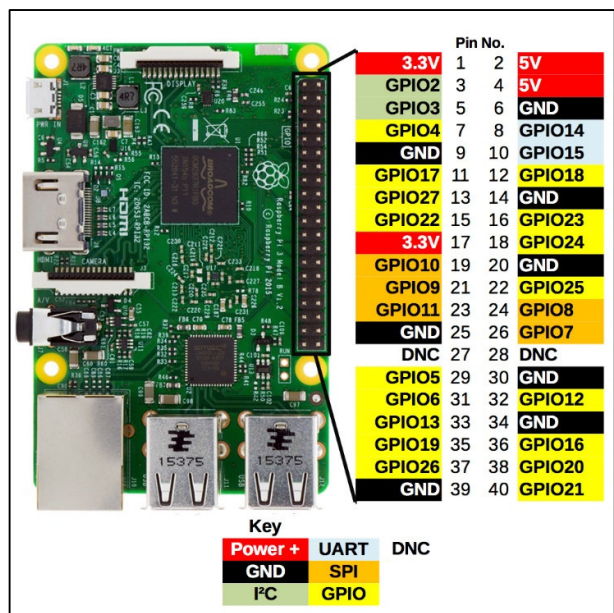


Figura 1. 4. Distribución de pines Raspberry Pi 4 B [25].

#### 1.3.3.3.2. Software

- **Raspbian**

Para la ejecución de diferentes tareas y aplicaciones en la AWS es necesario que esta cuente con un OS (*Operating System*, Sistema Operativo), el cual facilita el manejo de los periféricos, sistemas de almacenamiento, memorias, entre otros; es decir, un OS permite administrar y gestionar el ordenador. A pesar de que la tarjeta Raspberry Pi tiene varias opciones de OS para su instalación de acuerdo al tipo de proyecto o solución que se requiera solventar, el más común o recomendable para este microordenador es Raspbian.

Raspbian es un OS optimizado para el hardware de Raspberry Pi, se basa en la distribución gratuita y libre GNU/Linux denominada Debian y gracias al continuo desarrollo se ha convertido en un OS por excelencia, contiene un gran volumen en su repositorio, a los cuales los usuarios pueden acceder para descargar múltiples programas y controladores, como: procesadores de texto Office, servidores y clientes de correo electrónico [26].

- **Servidor LAMP**

Se cuenta con múltiples plataformas de desarrollo web que dependen del tipo de sistema operativo para la creación de sitios web, aplicaciones, entre otros, la elección de estos se liga al costo, flexibilidad y velocidad [24]. Las plataformas más comunes son: LAMP (Linux, Apache, MySQL, PHP), WAMP (Windows, Apache, MySQL, PHP), XAMP (X OS, Apache, MySQL, PHP), MAMP (MacOS, Apache, MySQL, PHP), entre otros.

LAMP es un conjunto conformado por una pila de cuatro tecnologías de software, como son: el sistema operativo Linux, servidor web Apache, servidor de base de datos MySQL y lenguaje de programación PHP, las cuales permiten desarrollar y alojar páginas web dinámicas que, por su disponibilidad, flexibilidad, sencillez de implementación, seguridad y bajo costo hacen de este servidor una opción preferida para los desarrolladores.

**Linux:** Es el sistema operativo que puede ser de cualquier distribución de GNU/Linux, que sirve de base para ejecutar el servidor web Apache y el resto de los componentes [24].

**Apache:** Servidor web de código abierto que aloja sitios webs estáticos y dinámicos, aunque no puede interpretar estos últimos, PHP es el lenguaje que interpreta el código fuente enviado por Apache [24].

**MySQL:** Servidor de base de datos más popular, el cual permite almacenar y enviar datos a través de un motor centralizado de datos [24].

**PHP:** Es un lenguaje de programación que se sigue utilizando para el desarrollo de páginas web dinámicas de forma simplificada y efectiva [24].

Sin embargo, existen opciones por las cuales se puede reemplazar una tecnología por otra que cumpla la misma función como por ejemplo el gestor de base de datos MySQL se puede reemplazar por MariaDB o Apache por Nginx, entre otros [27].

- **Hosting Web**

En la actualidad existe mucha demanda por contratar servicios en la nube como por ejemplo para: almacenamiento de información, administración de datos, alojamiento de sitios web; entre otros, que propicien rendimiento, escalabilidad, flexibilidad y asequibilidad para sus suscriptores, es por ello que un ISP (*Internet Service Provider*, Proveedor de Servicios de Internet) dispone de servicios como : SaaS (*Software as a Service*, Software como Servicio), PaaS (*Platform as a Service*, Plataforma como Servicio) e IaaS (*Infrastructure as a Service*, Infraestructura como Servicio), acorde a las necesidades de los usuarios.

**SaaS:** Es uno de los servicios más utilizados, en el cual el ISP aloja el software de aplicación en sus servidores, se encarga del mantenimiento, soporte y disponibilidad, el cliente puede acceder a ellos a través de un navegador web, tal como aplicaciones web de Google, Dropbox, Microsoft Office 365 [28].

**PaaS:** Servicio que proporciona una plataforma bajo demanda en el cual los usuarios o suscriptores pueden desarrollar, implementar, probar y distribuir sus aplicaciones [29].

**IaaS:** Es un servicio por suscripción, los usuarios ingresan al software y aplicaciones a través de navegadores web o APIs propias del proveedor [28].

El hosting web o alojamiento web se define como un servicio de acceso a través de Internet a herramientas informáticas como aplicaciones, almacenamiento de datos, herramientas de desarrollo, servidores; entre otros, alojadas en un conjunto de servidores que trabajan de forma conjunta y son gestionadas por un ISP [29].

Un hosting puede ser gestionado a través de cPanel (*Control Panel*, Panel de Control), basado en Linux que centraliza el control de todos los servicios en un solo lugar, facilita el monitoreo, administración, instalación, configuración de sitios web y aplicaciones albergados en el hosting [30].

Para el desarrollo de sitios web o aplicaciones el hosting y el dominio son dos partes esenciales, el hosting es el espacio alquilado o contratado en el servidor para el

almacenamiento de información y el dominio es la dirección de acceso al sitio, sin embargo, es necesario mencionar que en este trabajo solo se hace uso de los servicios de almacenamiento de información como base de datos y administrador de archivos.

- **FTP (*File Transfer Protocol*, Protocolo de Transferencia de Archivos)**

FTP es un protocolo de la capa de aplicación que trabaja con el modelo cliente-servidor que permite conectarse a un servidor remoto para intercambiar archivos a través de cualquier dispositivo cliente sin importar el sistema operativo de manera sencilla y rápida, es decir, FTP es una forma eficaz e independiente de enviar y descargar archivos utilizado comúnmente para guardar información en un hosting.

La conexión al servidor se lo realiza de forma general introduciendo ciertos datos como dirección del servidor, usuario y contraseña; este proceso abre dos canales, el canal de comandos o control y el canal de datos. El canal de control se establece de forma general en el puerto 21, donde el cliente envía comandos u órdenes al servidor, mientras que el canal de datos se establece en el puerto 20 y permite el intercambio de los archivos entre los dispositivos [31].

- **SMTP (*Simple Mail Transfer Protocol*, Protocolo de Transferencia Simple de Correo)**

Notificar algún evento mediante correos electrónicos a través de internet ha sido el método más simple, rápido y seguro de comunicación entre usuarios durante años, SMTP es un protocolo de la capa aplicación que permite enviar emails, este proceso utiliza el modelo cliente-servidor, es decir el cliente es quien inicia la comunicación y el servidor responde a las solicitudes.

SMTP trabaja en conjunto con protocolos como POP3 (*Post Office Protocol*, Protocolo de Oficina Postal) e IMAP (*Internet Message Access Protocol*, Protocolo de Acceso a Mensajes de Internet), para cumplir el proceso de envío y recepción de correos.

En la tabla 1.8., se muestra una breve descripción de los servidores y puertos más utilizados por los usuarios [32]:

**Tabla 1. 8.** Descripción de servidores SMTP [32].

<b>Proveedor</b>	<b>Dirección</b>	<b>Puerto</b>
Yahoo	smtp.mail.yahoo.com	587
Outlook	smtp.office365.com	587
Gmail	smtp.gmail.com	587 (TLS/STARTTLS), 465 (SSL)



- **Acceso remoto**

La mayoría de las empresas en la actualidad trabajan mediante plataformas que ofrecen el servicio de acceso de forma remota a los equipos de oficina; con el objetivo de buscar la autonomía de sus empleados, reducción de costos, tener control sobre los procesos internos y brindar un servicio más ágil a los clientes [33].

El acceso remoto es la tecnología capaz de acceder y controlar a un ordenador o dispositivo desde otro equipo, en cualquier momento y desde cualquier parte del mundo a través de software o aplicaciones específicas e internet. La comunicación y conexión a los dispositivos se realiza a través de una red virtual que mediante autenticación entre los extremos evita problemas de pérdida de datos e invasiones [33].

Existen varias aplicaciones que permiten la conexión remota de forma gratuita o pagada, a continuación, en la Tabla 1.9, se presenta ciertas características de tres diferentes plataformas que ofrecen el servicio más utilizadas y conocidas.

**Tabla 1. 9.** Software de acceso remoto [34] [35].

<b>Software</b>	<b>Características</b>	<b>Precio</b>
<b>TeamViewer</b>	<ul style="list-style-type: none"> <li>- Software más popular.</li> <li>- Compatible con múltiples plataformas.</li> <li>- Ofrece autenticación de dos factores.</li> <li>- Permite la conexión y control de varios escritorios, servidores y dispositivos IoT.</li> <li>- Permite la transferencia de archivos, grabado de sesiones, comunicación por chat entre equipos e impresión.</li> <li>- Permite configurar Wake-on-LAN que posibilita encender el equipo de forma remota.</li> <li>- Escritorio remoto 4K<sup>13</sup>.</li> <li>- Fácil de usar e instalar.</li> </ul>	Gratuito
<b>AnyDesk</b>	<ul style="list-style-type: none"> <li>- Compatibilidad limitada con plataformas.</li> <li>- Ofrece autenticación de dos factores.</li> <li>- Es más ligero y rápido.</li> <li>- Cuenta con una versión portable.</li> <li>- Permite la transferencia de archivos, grabado de sesiones y comunicación por chat entre equipos.</li> </ul>	Gratuito
<b>VNC Connect</b>	<ul style="list-style-type: none"> <li>- Compatibilidad con múltiples plataformas.</li> <li>- Permite la transferencia de archivos, comunicación por chat e impresión.</li> </ul>	Pagada

<sup>13</sup> **4K.** – Resolución de imagen con mejor calidad, conocida como Ultra HD posee mayor cantidad de píxeles, tecnología capaz de alcanzar 3840x2160 píxeles.

	<ul style="list-style-type: none"> <li>- Autenticación multifactorial<sup>14</sup>.</li> <li>- Se debe trabajar con capas extras como SSH o VPN para mayor seguridad, ya que en si no se considera como un protocolo confiable [36].</li> </ul>	
--	---	--

#### 1.3.3.3.3. *Infraestructura*

La infraestructura de una estación o garita meteorológica debe estar diseñada y construida de tal modo que soporte los dispositivos implementados en ella, los proteja de fenómenos como: la radiación solar, la lluvia, el viento y la condensación; y tenga la ventilación suficiente para que la circulación del aire sea libre [37].

Varias recomendaciones estandarizadas por la OMM se deben tomar en cuenta para la construcción de garitas meteorológicas y que esta no interfiera en la adquisición de datos de parámetros meteorológicos y su exactitud, a continuación, se presentan algunos criterios para el diseño de garitas con ventilación natural o artificial.

- **Ventilación natural**

- ✓ La estructura puede ser de un material robusto, la mayoría de las garitas son fabricadas con madera tratada para soportar el medio externo o plástico.
- ✓ La garita debe ser pintada dentro y fuera con pintura blanca esmaltada anti higroscópica para reflejar los rayos directos del sol y que no permita la absorción de la humedad.
- ✓ Las paredes pueden ser dobles en forma de persianas con una inclinación a 45°, permite la circulación libre del aire dentro de la garita e impide la incidencia directa de los rayos del sol a los dispositivos.
- ✓ La cubierta debe tener dos niveles con espaciado entre ellas, la primera evita el calentamiento del aire en condiciones extremas, mientras que la segunda debe estar inclinada de tal manera que caiga el agua de lluvia.
- ✓ La construcción debe ser de tal manera que exista un espacio acorde entre las paredes y los dispositivos y que la capacidad de acumular calor sea baja [38].
- ✓ El mantenimiento de una garita debe ser periódica de dos a cuatro veces al año, para que la pintura y limpieza de esta estén en buenas condiciones y no afecte a los sensores.

---

<sup>14</sup> **Autenticación multifactorial.** – Es una tecnología de seguridad que requiere múltiples métodos de autenticación para el inicio de sesión de un usuario.

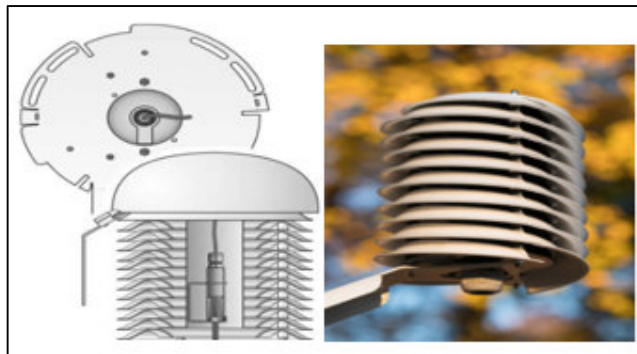
- ✓ Este diseño se conoce como pantalla de Stevenson que proporciona un modelo estandarizado para garitas meteorológicas, como se muestra en la Figura 1.5.



**Figura 1. 5.** Garitas de madera y plástico con ventilación natural [37] [38].

- **Ventilación artificial**

- ✓ Pantallas cilíndricas concéntricas en posición vertical.
- ✓ Fabricado con materiales de aislante térmico, con ello reduce la absorción de calor.
- ✓ Contiene un ventilador instalado para que el aire ingrese a la garita y no cruce por ella, por ello se conoce como ventilación forzada, como se muestra en la Figura 1.6.
- ✓ La velocidad de ingreso del aire requerida está comprendida entre 2m/s-10m/s.



**Figura 1. 6.** Pantalla o protección con ventilación artificial [39].

#### 1.3.3.3.4. *Ubicación de estaciones meteorológicas*

Las garitas deben estar ubicadas en sectores específicos de tal modo que las condiciones o infraestructuras del terreno no influyan en los parámetros meteorológicos a medir, como se muestra en la Figura 1.7, por ello se presentan a continuación varias recomendaciones de ubicación de casetas meteorológicas:

- ✓ La garita debe ubicarse entre 1,2m – 2m de altura desde la superficie o suelo a instalar, esto se debe a que en estos límites el gradiente o diferencia de temperatura es menor a 0.2 °C [11].
- ✓ Evitar ubicar las garitas en terrenos rocosos, de hormigón, césped o techos, si se da una de estas condiciones ubicar la garita a 1,5m de altura de la superficie.
- ✓ Procurar que el terreno esté nivelado, evitar que el agua se acumule y provoque inundaciones para garantizar las condiciones climáticas de la zona y la accesibilidad para el mantenimiento de la garita [40].
- ✓ No debe haber obstrucciones como: árboles, edificios u otros objetos a su alrededor que puedan alterar las condiciones térmicas del medio.
- ✓ Se debe evitar que el terreno tenga una pendiente exagerada u hondonada, si este fuera el caso, la garita debe estar nivelada, sin embargo, la información de las condiciones climáticas sería únicamente local debido a que las condiciones o parámetros difieren con respecto a su entorno más amplio [41].



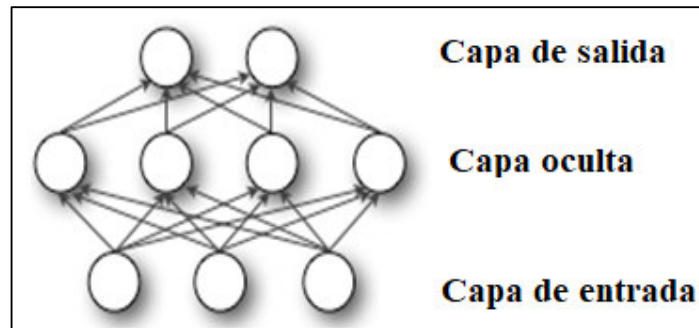
**Figura 1. 7.** Ubicación de garitas: terraza y terreno óptimo [42] [43].

#### **1.3.4. REDES NEURONALES**

Las redes neuronales es un sistema conformado por un conjunto de algoritmos potentes con los cuales podemos modelar el comportamiento de las neuronas del cerebro y dotar de inteligencia artificial a los ordenadores quienes pueden mejorar sus capacidades y resolver problemas por sí mismos [44].

Se considera a las redes neuronales como un modelo de capas, que están conformadas por una o varios nodos o neuronas en donde ocurre la computación o la automatización de algoritmos [45], se distinguen tres tipos de capas: entrada, ocultas y salida, el número de

capas de una red neuronal incluyendo la capa de entrada y salida dependerán de la complejidad del problema a resolver.



**Figura 1. 8.** Capas de una red neuronal [45].

En la actualidad las redes neuronales son eficientes y útiles al momento de evaluar enormes cantidades de datos, sus potentes algoritmos permiten identificar patrones. Han sido implementadas en diferentes aplicaciones como: reconocimiento de voz, imágenes, texto o análisis de series temporales [12]; en el área de la meteorología se han implementado redes neuronales para el pronóstico del tiempo.

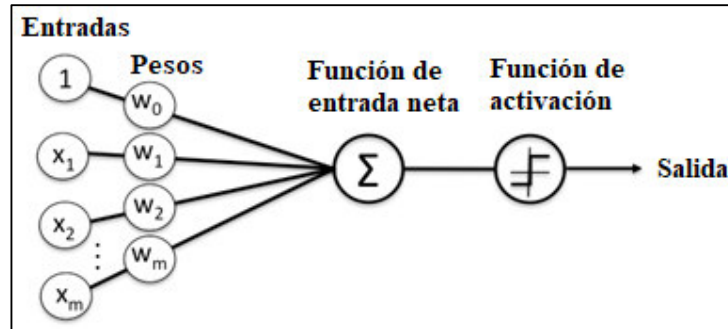
#### **1.3.4.1. Neurona o nodo**

Una neurona es la unidad básica de una red neuronal, en donde ocurre la automatización del algoritmo, que al igual que las capas se distinguen tres tipos de neuronas:

- **Entrada:** Son aquellas que reciben las señales del entorno.
- **Salida:** Son aquellas que entregan una respuesta de acuerdo a los estímulos de las señales de entrada.
- **Ocultas:** Son aquellas que realizan el proceso interno de la red y no están conectadas directamente con el entorno.

El funcionamiento de la neurona y procesamiento de la información dentro de las redes neuronales dependen básicamente de: las conexiones de las neuronas o nodos de cada capa, el peso numérico asignado y un umbral asociado a estas, es decir, los datos de entrada llegan hasta uno de los nodos que por su nivel de importancia o peso amplifica o atenúa las entradas acorde al algoritmo de aprendizaje [45], el producto de las entradas con sus respectivas ponderaciones se suma, este valor pasa a través de una función de activación y si sobrepasa el umbral designado la neurona se activa, de este modo se da la

transmisión de la señal a la siguiente capa, en otras palabras, si la señal pasa al siguiente nivel es porque la neurona se activó.



**Figura 1. 9.** Estructura de un nodo o neurona [45].

Matemáticamente la salida de una neurona se puede describir mediante la Ecuación 1.1 [46]:

$$y_k = \beta \sum_{i=1}^m x_i w_{ki} + bias \quad (1.1)$$

Donde:

$y_k$ : es la salida.

$\beta$ : es la función de activación.

$x_i$ : son las entradas.

$w_{ki}$ : son los pesos respectivos de las neuronas.

$bias$ : umbral de activación o no de la neurona.

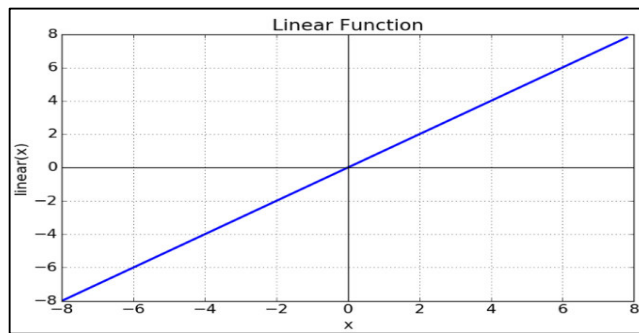
#### 1.3.4.1.1. Funciones de activación

Una función de activación en una red neuronal permite tomar decisiones a la salida de la neurona, en otras palabras, la función de activación actúa como un filtro que define un umbral o un límite de decisión lineal o no lineal que el nodo aprende [47] y que la señal debe sobrepasar para que se active la neurona y transmita la información a la siguiente neurona.

Por tanto, una función de activación permite tener el control del proceso de entrenamiento de la red, es decir, mediante la elección adecuada de una función el aprendizaje será más eficiente y de esta también dependen los tipos de predicciones a realizar.

- **Función de activación lineal**

Conocida también como función identidad, este tipo de función de activación permite obtener un valor único a la salida de la red neuronal y permite que la salida sea igual a la entrada como se muestra en la Figura 1.10, por ende, si se aplica este tipo de función de activación es porque se necesita de una regresión lineal [48], que no es más que un modelo matemático utilizado para analizar la relación existente entre dos o más variables dependientes.



**Figura 1. 10.** Función de activación lineal [48].

- **Función de activación no lineal**

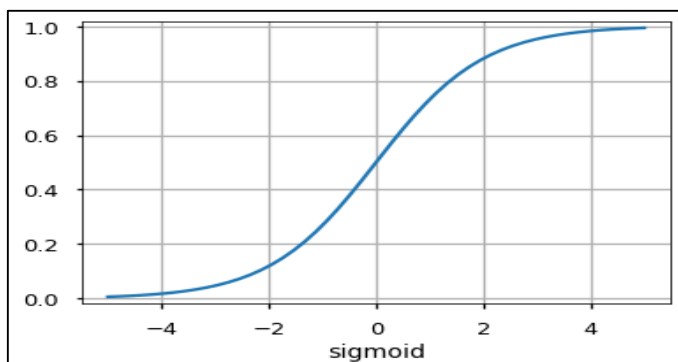
Se tienen varias opciones de funciones de activación no lineales, sin embargo, se presentan las funciones más utilizadas en redes neuronales:

- ✓ **Función de activación Sigmoide**

Denominada como función logística, los valores de salida de esta función están comprendidos en el rango de 0 y 1, por tanto, es considerada como una función de probabilidad, los valores altos tienden de forma asintótica a 1, mientras que los valores bajos tienden de manera asintótica a 0, la Ecuación 1.2 define este comportamiento, y se representa mediante la Figura 1.11; de manera general, se utiliza en la última capa para dividir los datos en dos clases [48].

$$F(x) = \frac{1}{1 - e^{-x}} \quad (1.2)$$

Las características de esta función son: tiene una lenta convergencia, si se evalúa en el punto 0 esta no es centrada, satura y “mata” el gradiente<sup>15</sup>, muestra un mejor rendimiento en la última capa, en la actualidad esta función no es muy usada [49].



**Figura 1. 11.** Función de activación Sigmoide [49].

✓ **Función de activación tangente hiperbólica (tanh)**

Los valores de salida de esta función están comprendidos en el rango de -1 y 1, los valores altos tienden de forma asintótica a 1, mientras que los valores bajos tienden de manera asintótica a -1, la Ecuación 1.3 define dicho comportamiento y se representa mediante la Figura 1.12, a pesar de ser una función centrada, esta función al igual que la sigmoide mata al gradiente [48].

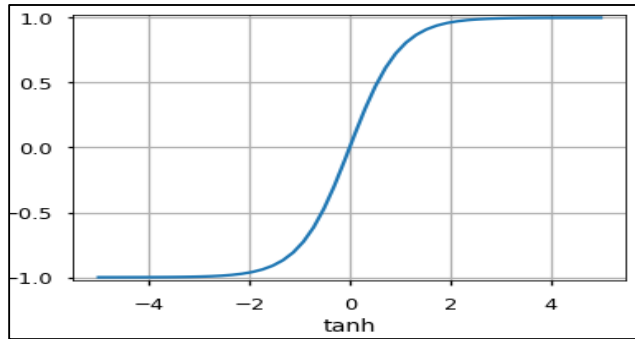
$$F(x) = \frac{2}{1 - e^{-2x}} - 1 \quad (1.3)$$

Se dice que la función tanh es de lenta convergencia, está centrada en 0, es utilizada para tomar decisiones entre una opción o su contraria, muestra un buen desempeño en las redes recurrentes [49].

---

<sup>15</sup> **Matar el gradiente.** – Es un problema que ocurre en el entrenamiento de las redes neuronales al ajustar los pesos de las neuronas en el cual si el gradiente desaparece o “muere” la red neuronal no aprende.





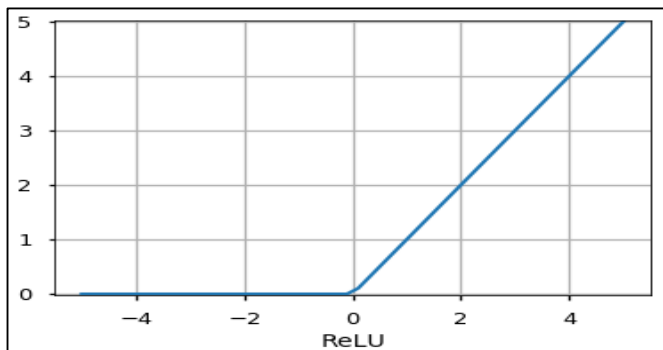
**Figura 1. 12.** Función de activación tanh(x) [49].

✓ **Función de activación ReLU (Rectificador Lineal Unitario)**

La función de activación ReLU es en la actualidad la más utilizada en el campo de Deep Learning (Aprendizaje Profundo) [50], se debe a que el aprendizaje en las redes neuronales ocurre más rápido, el comportamiento de esta función consiste en devolver un valor de 0 al tener entradas negativas, por tanto, desactiva la neurona; por contrario, entrega el mismo valor si este es mayor o igual a cero, la Ecuación 1.4 describe el comportamiento de esta función, que se representa mediante la Figura 1.13.

$$F(x) = \max(0, x) = \begin{cases} 0 & \text{para } x < 0 \\ x & \text{para } x \geq 0 \end{cases} \quad (1.4)$$

Las características de la función ReLU son: el gradiente de la función en el primer cuadrante es 1, mientras que en el segundo es 0 [48]; las neuronas se activan solamente si los valores son positivos, se usan de mejor manera en las redes convolucionales y en deep learning [50].



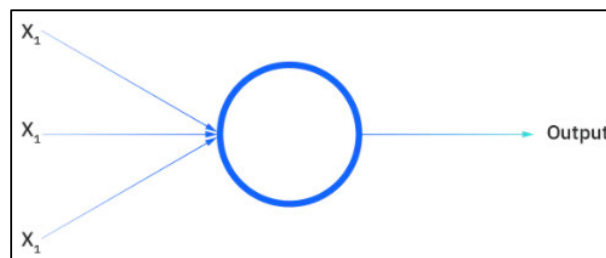
**Figura 1. 13.** Función de activación ReLU [49].

### 1.3.4.2. Tipos de redes neuronales

Existen estructuras de redes neuronales muy variadas que se las puede utilizar para múltiples propósitos, a continuación; se presentan las redes neuronales más comunes.

#### 1.3.4.2.1. Perceptrón

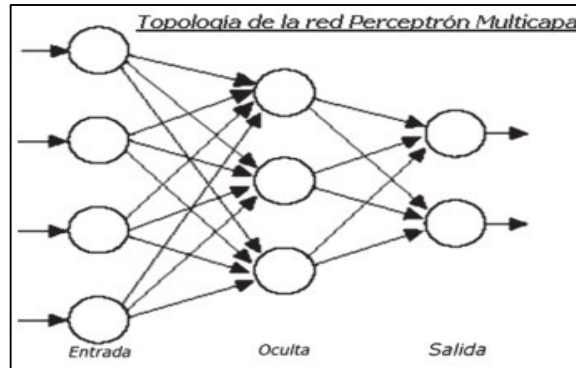
Es un modelo básico y el más antiguo, la red neuronal se conforma de una sola neurona o una sola capa, como se muestra en la Figura 1.14; que se modifica de acuerdo a los pesos o nivel de importancia de sus entradas y el valor del umbral, consistía en un discriminador lineal o clasificador binario que, a través de entrenamiento con datos, el perceptrón era capaz de tomar decisiones a través del reconocimiento de patrones, aunque este último era un proceso limitado porque permitía una clasificación lineal, es decir, separaba características de diferentes categorías mediante una línea sin que se mezclen [51].



**Figura 1. 14.** Perceptrón [52].

#### 1.3.4.2.2. Redes neuronales Feedforward o prealimentadas

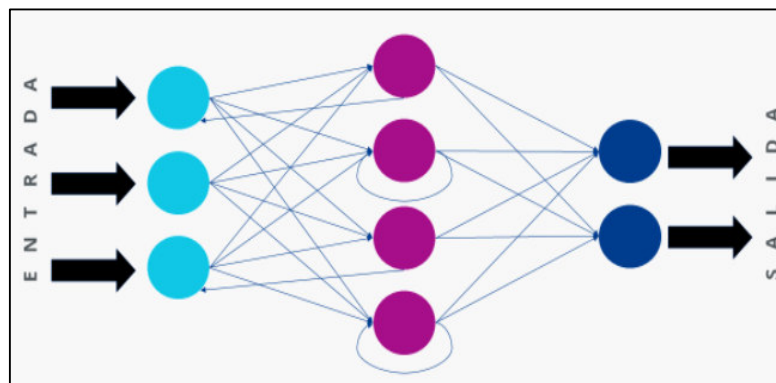
Son conocidas como redes neuronales MLP (*Multi-Layer Perceptrons*, Perceptrones Multicapa), este tipo de red está compuesta por una capa de entrada, una o varias capas ocultas y una capa de salida, como se muestra en la Figura 1.15, tiene la capacidad de transmitir o hacer fluir la información en una sola dirección [44], es decir, es unidireccional debido a que no se componen de conexiones de retroalimentación que formen ciclos.



**Figura 1. 15.** Red neuronal Feedforward [53].

### 1.3.4.3. Redes neuronales recurrentes (RNN)

Las RNN son un tipo de redes neuronales que se distinguen por la capacidad de incorporar una memoria para proporcionar la retroalimentación, es decir, utilizan la información de entradas anteriores almacenadas para influir tanto en las entradas como en las salidas actuales [54], por ello es de uso común para analizar datos de series secuenciales o series de tiempo, algunos ejemplos de aplicación de las RNN son: reconocimiento de voz, traductores de idiomas; entre otros, un ejemplo de red recurrente se presenta en la Figura1.16.



**Figura 1. 16.** Red Neuronal Recurrente [44].

Las RNN como se mencionó con anterioridad, son capaces de influir en los estados presentes y futuros de una red neuronal debido a que guardan en la memoria información relevante de acontecimientos pasados, es por ello, que la precisión de las predicciones en este tipo de red es más alta que en las redes feedforward y convolucionales.

La RNN tiene diferentes estructuras que en la actualidad se utilizan para resolver problemas de aprendizaje automático y las cuales se describen a continuación:

#### 1.3.4.3.1. LSTM (Long-Short Term Memory, Memoria a Corto y Largo Plazo)

Las redes LSTM son un tipo de RNN especiales que contienen celdas o etapas de memoria en las capas ocultas de la red neuronal, que tienen la capacidad de almacenar datos de dependencias a largo plazo. La red LSTM se introduce como una solución a los problemas de desaparición de gradiente y dependencia a largo plazo o memoria a corto plazo [54].

Las predicciones de los estados actuales de una RNN simple dependen de los estados anteriores de un pasado reciente, sin embargo, si estos estados anteriores no fueran del pasado reciente la predicción del estado actual no sería preciso [54], por tanto, LSTM juega un papel importante ya que recuerda información por largos periodos de tiempo y evita este inconveniente de dependencia a largo plazo.

El flujo de información que atraviesa la red LSTM puede ser controlado a través de estructuras especiales llamadas puertas, estas son: entrada, salida y olvido; como se muestra en la Figura 1.17, a las que se puede considerar como una red neuronal que aprende qué información se conserva o elimina [12].

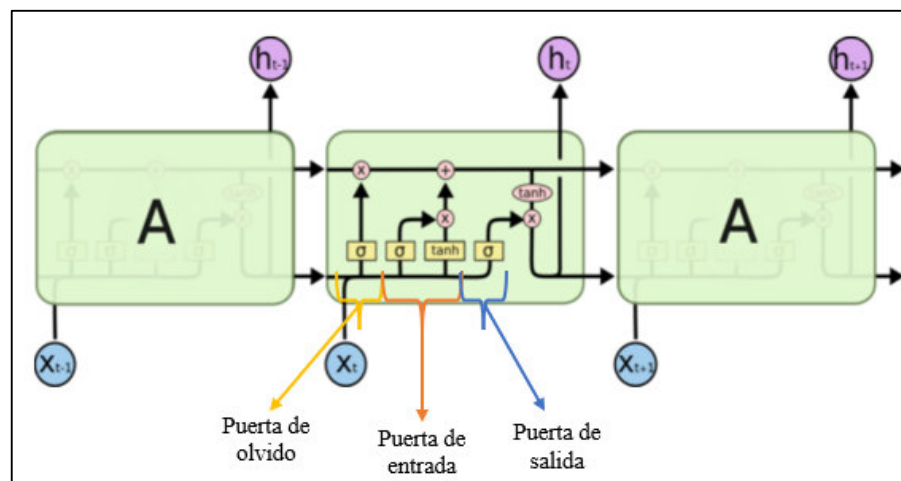
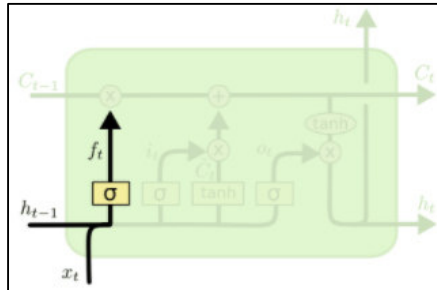


Figura 1. 17. Estructura de una celda LSTM [55].

- **Puerta de olvido**

La puerta del olvido de la Figura 1.18, es la primera fase de la celda LSTM que se encarga de seleccionar la información relevante a conservar o desechar [56], en esta etapa se

concatena el estado oculto anterior y la entrada actual que pasa a través de una función sigmoide, la cual limita a 0 que significa olvidar y 1 que significa conservar, la Ecuación 1.5 representa el funcionamiento de la puerta.



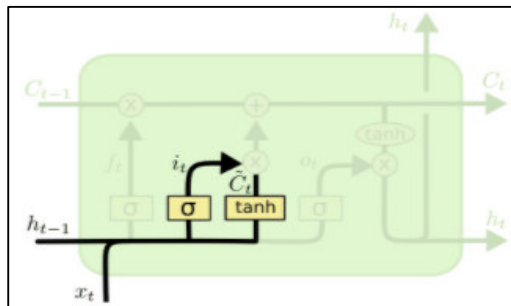
**Figura 1. 18.** Puerta de olvido LSTM [55].

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + bias_f) \quad (1.5)$$

- **Puerta de entrada**

La segunda fase es la puerta de entrada que de la Figura 1.19, que se encarga de actualizar el estado de la celda [56], en esta fase la concatenación del estado oculto anterior y la entrada actual pasa a través de una función sigmoide, la cual limita a 0 que significa “no importante” y 1 que significa “importante” a actualizar, este comportamiento se define en la Ecuación 1.6.

Por otro lado, la misma concatenación pasa a través de la función tangente hiperbólica que limita a -1 y 1 para determinar los valores posibles a agregarse al estado demostrado mediante la Ecuación 1.7. A través de una multiplicación de las salidas de las funciones se obtienen los nuevos valores para la actualización del estado de la celda LSTM.

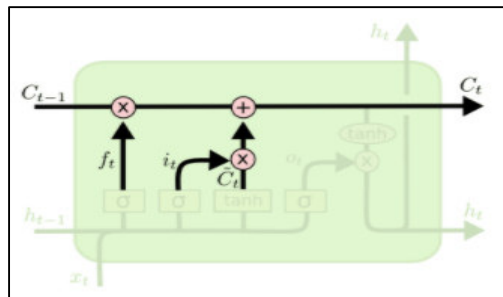


**Figura 1. 19.** Puerta de entrada LSTM [55].

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + bias_i) \quad (1.6)$$

$$\bar{c}_t = \tanh(W_c * [h_{t-1}, x_t] + bias_c) \quad (1.7)$$

La actualización del estado de la celda se representa mediante la Figura 1.20 y se realiza a través del proceso de multiplicación del estado anterior de la celda por el resultado de la función de la puerta de olvido, a estos valores se suma el producto de la función de la puerta de entrada o los nuevos valores de actualización de estado, con ello se actualiza el estado de la celda con nuevos valores que la red considere relevantes, comportamiento que describe la Ecuación 1.8.



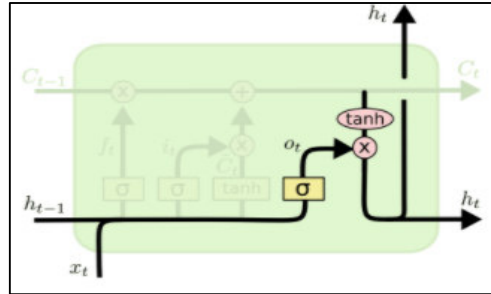
**Figura 1. 20.** Actualización de estado de la celda LSTM [55].

$$C_t = (C_{t-1} * f_t) + (i_t * \bar{c}_t) \quad (1.8)$$

- **Puerta de salida**

La última fase de la celda LSTM es la puerta de salida de la Figura 1.21 que determina el nuevo estado oculto de esta [56], se debe tener en cuenta que, el estado oculto contiene información de las entradas anteriores y permite además realizar las predicciones.

En esta etapa la concatenación del estado oculto anterior y la entrada actual pasan a través de la función sigmoide representada por la Ecuación 1.9, por otro lado, el estado actualizado de la celda pasa a través de la función tangente hiperbólica, de cuales se definieron sus funciones con anterioridad; mediante el producto de la salida de estas dos funciones se produce la información que contiene el nuevo estado oculto que con el estado actualizado de la celda pasan al siguiente punto de tiempo, proceso que se define mediante la Ecuación 1.10.



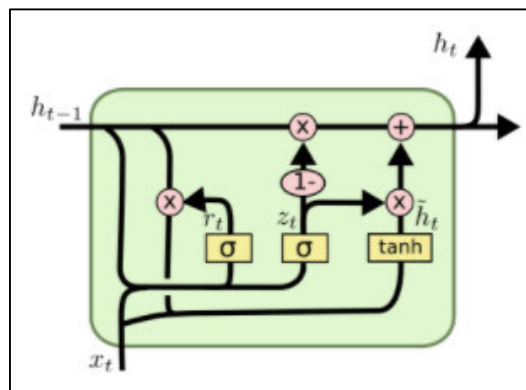
**Figura 1. 21.** Puerta de salida LSTM [55].

$$O_t = \sigma(W_o * [h_{t-1}, x_t] + bias_o) \quad (1.9)$$

$$h_t = O_t * \tanh(C_t) \quad (1.10)$$

#### 1.3.4.3.2. GRU (Gated Recurring Units, Unidades Recurrentes Cerradas)

Las GRU son otro tipo de redes RNN similares a LSTM que contienen celdas de memorias en la capa oculta de la red, la diferencia que sobresalta de estas redes es que la capacidad de procesamiento de la información se realiza a mayor velocidad [12], esta red combina la puerta de entrada y olvido en una puerta única de “actualización” y fusiona el estado de la celda con el estado oculto; entre otros cambios internos que permiten tratar las dependencias temporales de forma simplificada [12], la estructura de este tipo de RNN se presenta en la Figura 1.22.



**Figura 1. 22.** Estructura de una celda GRU [55].

#### 1.3.4.4. Redes neuronales convolucionales

La CNN (*Convolutional Neural Network*, Red Neuronal Convolucional) similar a la RNN es un modelo de red con retroalimentación, la CNN aprovecha los principios de álgebra lineal como la multiplicación de matrices y el uso de matrices bidimensionales para identificar patrones en las imágenes, es decir, las CNN de forma general se utilizan para el reconocimiento de imágenes y visión artificial [57].

La CNN es una arquitectura que no extrae características de forma manual, sino que aprende directamente de los datos ingresados, se conforma de varias capas ocultas con jerarquías, las cuales en el proceso de entrenamiento de la red aplican filtros a cada imagen [58]. Estos filtros que pueden variar según las características a distinguir, filtros para las primeras capas que se especializan en reconocer formas sencillas de una imagen como líneas, curvas, etc.; o filtros para las capas más profundas que reconocen formas complejas como rostros o siluetas [59].

La CNN se conforma de una capa de entrada, una capa de salida y varias capas ocultas, sin embargo, estas últimas tienen procesos definidos con el objetivo de aprender características de los datos de imágenes ingresadas a las cuales se conoce como: Convolución, Pooling y Clasificación [58].

- **Capa convolución**

Esta capa es la encargada de someter a la imagen de entrada a varios filtros de tamaño reducido que recorren todas las posiciones posibles de la imagen y activan características especiales u obtienen un valor de salida [60], es decir, el recorrido que realiza el filtro sobre la imagen permite recopilar características relevantes y reducir el tamaño de la imagen.

Además, luego de las capas convolucionales lo más habitual es aplicar funciones de activación tipo ReLU para una mayor velocidad de aprendizaje [60].

- **Capa pooling**

La función principal de esta capa es reducir el tamaño de parámetros que la red necesite aprender para que de esta forma sea manejable computacionalmente, este proceso se realiza disminuyendo la tasa de muestreo no lineal [58].

- **Capa de clasificación**

Es la capa final de una CNN que por su parte resuelve problemas de clasificación y se la conoce como *Fully Connected Layer*, es decir, es necesario contar con neuronas de salida para cada clase encargadas de distinguir entre varias características que pertenecen a una



imagen en particular, generalmente se utiliza una capa softmax que es una función de activación para este proceso [60].

En la Figura 1.23 se representa una CNN con los procesos para resolver problemas de clasificación de imágenes.

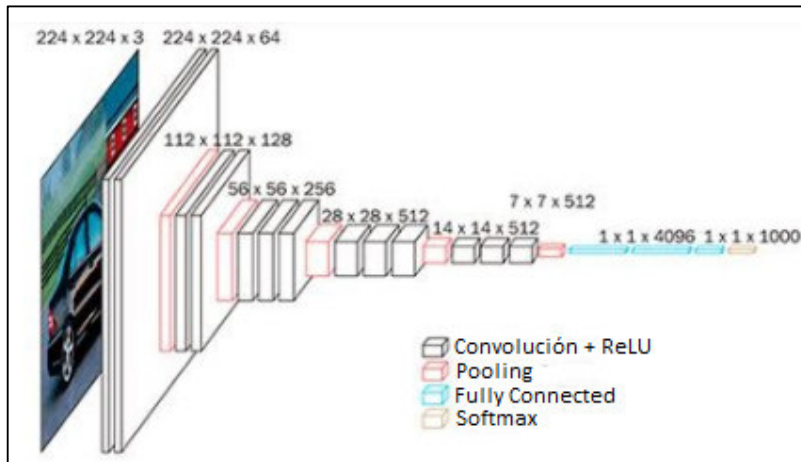


Figura 1. 23. Clasificación de imágenes con CNN [60].

#### 1.3.4.5. Modelo ARIMA

ARIMA (*Autoregressive Integrated Moving Average*, Media Móvil Integrada Autorregresiva) es un modelo estadístico que captura diferentes estructuras de datos de series temporales para el análisis y predicciones de estas [61], se conforma por tres componentes que cumplen funciones específicas y toman valores enteros o cero para indicar el modelo que se utiliza, se denota como ARIMA (p, d, q) y a las cuales se describen a continuación:

- **AR (Autoregressive):** Modelo de regresión lineal en la cual el valor de salida depende linealmente de las observaciones pasadas, la componente es “p” conocido como orden de retraso [61], representa el número de observaciones pasadas de las que depende la salida o valor actual.
- **I (Integrated):** Combina los modelos AR y MA, ayuda a que los datos sean estacionarios en todo momento mediante la aplicación de la técnica de diferenciación, la componente es “d” conocida como grado de diferenciación [61], representa el número de veces que se diferencian las observaciones.
- **MA (Movil Average):** Modelo en el cual se procesa cierto conjunto de datos para obtener una lista de valores promediados de acuerdo a un periodo de tiempo específico, emplea el error existente y la observación actual en la aplicación de MA de las observaciones pasadas, la componente es “q” conocida como orden de

media móvil, representa el tamaño de la ventana o el número de errores de pronósticos pasados que deben ser utilizados en el modelo [61].

El requisito necesario para que el modelo ARIMA(p, d, q) sea eficiente en las predicciones es que las series temporales sean estacionarias, es decir, antes de estimar valores futuros la media y varianza deben ser constantes en todo momento [62], a pesar que la mayoría de series son no estacionarias, es posible transformarlas mediante la aplicación de técnicas como: diferenciación, logaritmo natural, raíz cuadrada; entre otros, sin embargo, no todas se pueden aplicar a valores negativos.

Por tanto, la técnica más utilizada en el modelo ARIMA (p, d, q) es la diferenciación que puede ser [63]:

- **Diferenciación simple:** Proceso en el que se calcula la diferencia entre el valor actual y el valor anterior, sin tomar en cuenta el valor más antiguo de la serie.
- **Diferenciación estacional:** Proceso en el cual se calcula la diferencia entre el valor actual y el valor estacional anterior.

Para el cálculo de las componentes p, d, q es importante obtener las funciones de autocorrelación parcial, realizar la prueba ADF (*Augmented Dikey Fuller*, Dikey Fuller Aumentada) y la función de autocorrelación respectivamente. Estas funciones permiten saber la correlación existente entre los datos de las series temporales, es decir, expresan la relación lineal existente entre dos variables y definen los mejores coeficientes a utilizar en el modelo ARIMA (p, d, q) y con ello establecer las observaciones pasadas más útiles para pronosticar valores futuros.

- **FACP (Función de Autocorrelación Parcial)**

Es la autocorrelación existente entre dos variables separadas en k intervalos de distancia, eliminando el posible efecto de los residuos de intervalos intermedios. El coeficiente "p" del modelo AR(p) se determina a partir de esta función en donde los primeros p términos son distintos de 0 y el resto son nulos [64], el término p se elige de acuerdo a la significancia de los retardos, es decir, p es el número de términos que sobrepasan el límite de significancia (0,05) [65].

- **Prueba ADF**

Es una prueba estadística en la cual hipotéticamente se define que la serie es no estacionaria, por tanto, mediante esta prueba se verificará dicha hipótesis de lo contrario, será necesario un grado de diferenciación, la prueba consiste en verificar si el factor p

entregado por el modelo es menor a 0,05 que es el límite de significancia, por ende, la serie es estacionaria [65], caso contrario se debe buscar un nuevo factor de diferenciación para cumplir con la condición.

- **FAC (Función de Autocorrelación)**

Es la autocorrelación existente entre dos variables separadas en  $k$  intervalos de distancia. El coeficiente “ $q$ ” del modelo  $MA(q)$  se determina a partir de esta función en donde los primeros  $q$  términos son distintos de 0 y el resto son nulos [64], al igual que el término  $p$ , el término  $q$  se elige de acuerdo a la significancia de los errores de pronóstico retrasados, es decir,  $p$  es el número de términos que sobrepasan el límite de significancia (0,05) [65].

Los parámetros antes descritos pueden ser elegidos al azar, pero es necesario tener en cuenta que el valor  $p$  entregado por el modelo debe ser menor al límite de significancia (0,05).

#### **1.3.4.6. Entrenamiento de redes neuronales**

El entrenamiento previo de las redes neuronales permite calibrar tanto los pesos o niveles de importancia como los umbrales de la red para que las conexiones entre neuronas sean adecuadas y cumplan la función correcta [66], en otras palabras, el entrenamiento permite desarrollar algoritmos apropiados para resolver los problemas propuestos y que la red aprenda de forma automática, para ello existen métodos de aprendizaje que se presentan a continuación.

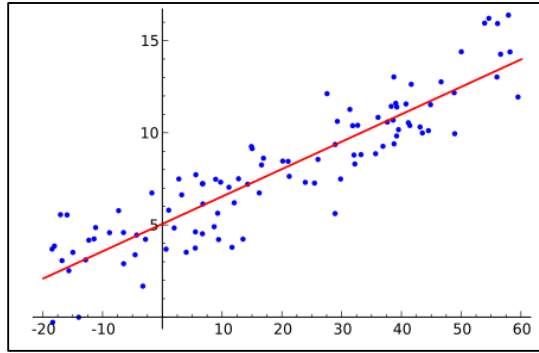
##### *1.3.4.6.1. Aprendizaje supervisado*

La modalidad de aprendizaje supervisado trabaja con datos introducidos por personas, es decir, para este método de aprendizaje es necesario la intervención humana para etiquetar, clasificar e introducir los datos a un algoritmo que permite encontrar la función correcta para designar etiquetas apropiadas, con ello, es posible asociar las entradas y salidas mediante el entrenamiento del algoritmo con datos previos o históricos [44].

Existen dos tipos de aprendizaje supervisado:

- **Regresión**

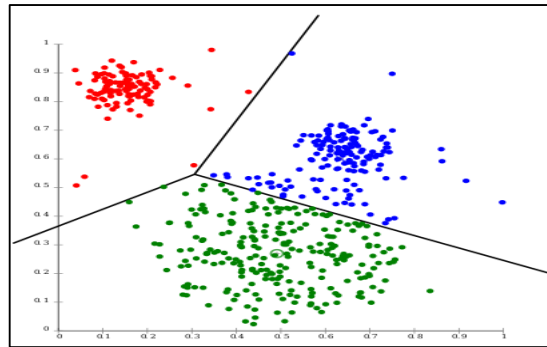
Se define el aprendizaje supervisado de regresión cuando el dato a predecir es un valor numérico [67], por ejemplo: las predicciones meteorológicas cómo se representa en la Figura 1.24.



**Figura 1. 24.** Aprendizaje supervisado de regresión [67].

- **Clasificación**

Por su parte, el aprendizaje supervisado de clasificación se tiene cuando el dato a predecir es un valor categórico como se muestra en la Figura 1.25, es decir clasifica objetos en clase de acuerdo a patrones establecidos [67].



**Figura 1. 25.** Aprendizaje supervisado de clasificación [67].

#### 1.3.4.6.2. *Aprendizaje no supervisado*

El método de aprendizaje no supervisado no requiere de la intervención humana, en este caso no es necesario asociar una entrada con una salida a través de etiquetas, tiene un carácter exploratorio en el cual solo se puede describir la estructura de los datos. Por tanto, el algoritmo en este caso trabaja con datos no etiquetados, pero cumple la función de agrupación que a través de similitudes crea grupos de datos sin conocer las individualidades de cada uno [67].

Existen dos tipos de aprendizaje no supervisado [68]:

- **Clustering**

Agrupar o segmentar los datos de salida de acuerdo a las similitudes, por ejemplo: agrupar las personas que compraron un auto.

- **Asociación**

Encuentra y establece reglas dentro de los conjuntos agrupados, por ejemplo: las personas que compraron un auto deben comprar un seguro.

#### 1.3.4.6.3. *Aprendizaje por refuerzo*

En este tipo de aprendizaje se trabaja con retroalimentación, es decir, el modelo mejora sus capacidades de acuerdo a la información de retroalimentación que obtiene en respuesta a las acciones tomadas, por tanto, el modelo aprende mediante el proceso de prueba-error.

### 1.3.4.7. **Backpropagation en el entrenamiento de redes neuronales**

Para entender que es la propagación hacia atrás y como este algoritmo matemático ayuda en el entrenamiento de la red neuronal se aborda dos conceptos fundamentales.

#### 1.3.4.7.1. *Función de pérdida*

La función de pérdida representa la desviación o error existente entre las predicciones entregadas por la red neuronal y los valores reales utilizados en su entrenamiento, por tanto, a menor desviación o error de la función de pérdida la red neuronal será más eficiente, esto se logra ajustando los pesos de la red.

Las métricas utilizadas en las tareas de regresión para evaluar el error de la red neuronal son las siguientes:

- ✓ **Error lineal**

Este error se denomina error local y resulta de la diferencia entre el valor real y el valor predicho por la red neuronal, como se presenta en la Ecuación 1.11.

$$\text{error} = \text{valor real} - \text{valor estimado} \quad (1.11)$$

✓ **MSE (Mean Squared Error, Error Cuadrático Medio)**

MSE es la métrica más utilizada en las tareas de regresión, su cálculo permite conocer de forma general el porcentaje de error que comete la red neuronal, sin embargo, es un parámetro preferido, ya que es diferenciable, por tanto, es optimizable. El MSE representa el promedio de la diferencia cuadrática de los datos reales y predichos por la red como se presenta en la Ecuación 1.12 [69].

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (\text{valor real}_i - \text{valor estimado}_i)^2 \quad (1.12)$$

✓ **RMSE (Root Mean Squared Error, Raíz del Error Cuadrático Medio)**

Otra métrica utilizada en la tarea de regresión es el RMSE que representa la desviación estándar de un conjunto de datos, RMSE representa la raíz cuadrada del promedio de la diferencia cuadrática de los datos reales y predichos por la red como se presenta en la Ecuación 1.13 [69]. Este método es útil para reducir errores grandes en las predicciones y evaluar el desempeño de la red neuronal.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\text{valor real}_i - \text{valor estimado}_i)^2} \quad (1.13)$$

A pesar de que, RMSE es más intuitivo que MSE este no es usado todo el tiempo, se debe al costo computacional, por ende, menor será la velocidad con la que calcula el error en cada iteración y más lento será el aprendizaje de la red. Además, se tiene que MSE permite a ciertos algoritmos de optimización encontrar soluciones a mayor velocidad.

1.3.4.7.2. *Gradiente*

El gradiente se puede definir como la generalización multivariable de la derivada de la función de pérdida de valor vectorial; que corresponde a la contraparte multidimensional de una derivada unidimensional que es de valor escalar, es decir, el gradiente es el conjunto de las derivadas parciales de la función de pérdida [70], que representa la pendiente de la recta tangente a la gráfica de una función y la rapidez con que el valor de una función cambia de acuerdo con el cambio que sufre su variable independiente [70].

El gradiente por su parte controla cuánto aprende la red neuronal en el entrenamiento, por tanto, si el valor del gradiente es muy bajo o nulo se concluye que la red neuronal poco o nada aprende, por tanto, las predicciones serían deficientes, este inconveniente ocurre en redes con múltiples capas y es conocido como: problema de gradiente descendiente (pendiente) [70].

Entendido los conceptos antes mencionados se puede definir qué, la propagación hacia atrás en el entrenamiento de las redes neuronales juega un papel importante, ya que es un algoritmo matemático que permite mejorar la precisión en las predicciones de dato; es decir, el algoritmo de backpropagation permite calcular el gradiente de la función de pérdida respecto a cada uno de los pesos para ajustarlos en función de la tasa de error de la red [71]. Los pesos ajustados adecuadamente generarán tasas de error más bajas por ende el modelo de red configurado será más confiable, para realizar este proceso de generar pesos cada vez mejores y solucionar el problema de gradiente descendiente se utilizan optimizadores [72].

#### 1.3.4.7.3. Optimizadores

Como se mencionó con anterioridad los optimizadores en las redes neuronales son una herramienta muy importante para encontrar los pesos adecuados para que la función de pérdida minimice su valor.

De acuerdo con el experimento realizado en [73], en el cual se proponen cuatro problemas tradicionales de aprendizaje automático que se muestran en la Tabla 1.10.

Se experimenta con los optimizadores disponibles en la biblioteca de redes neuronales Keras como: Adadelta, Adagrad, Adam, Adamax, Ftrl, Nadam, RMSprop, SGD (*Stochastic Gradiente Descendent*, Gradiente Descendiente Estocástico) y las funciones de pérdida como: MAE (*Mean Absolute Error*, Error Absoluto Medio) para la regresión, binary cross entropy como clasificador binario y categorical cross entropy como clasificador multiclase.

**Tabla 1. 10.** Problemas tradicionales de aprendizaje automático [73].

<b>Tipo de problema</b>	<b>Arquitectura de red</b>	<b>Juego de datos</b>
<b>Regresión simple</b>	Red neuronal simple	Precio de viviendas en Boston
<b>Clasificador (multiclase) de imágenes)</b>	Red convolucional 2D	Fashion-MNIST
<b>Clasificador (binario) de textos</b>	Red convolucional 1D + Embeddings	Análisis sentimiento IMDB
<b>Forecasting de series temporales</b>	Red neuronal recurrente con LSTM	Predicción temperatura

Para el experimento se mantiene constante el valor por defecto de Learning Rate <sup>16</sup> (tasa de aprendizaje) de los optimizadores, el cual tiene un valor de 0.001. Para el caso en particular que se estudia en este proyecto que tiene como objetivo la predicción de temperatura y humedad relativa mediante el uso de una red recurrente LSTM y de acuerdo a los experimentos realizados, se concluye que el optimizador Adam muestra los mejores resultados, por esto, se presenta a continuación una definición del optimizador Adam.

- **Optimizador Adam (*Adaptative moment estimation*)**

El optimizador Adam une las características de dos optimizadores, por un lado, AdaGrad (*Adaptive Gradient Algorithm*) que define un valor diferente de Learning Rate para cada uno de los pesos y los calcula a partir de la acumulación de gradientes en cada iteración obtenidos en base al factor de entrenamiento inicial adaptado a cada peso [73].

Y por otro, RMSprop (*Root Mean Square Propagation*) que mantiene valores diferentes de Learning Rate para cada peso, pero inserta un concepto de “ventana”, por tanto, considera un promedio móvil de los gradientes más recientes [73].

#### **1.3.4.8. Deep Learning (Aprendizaje Profundo)**

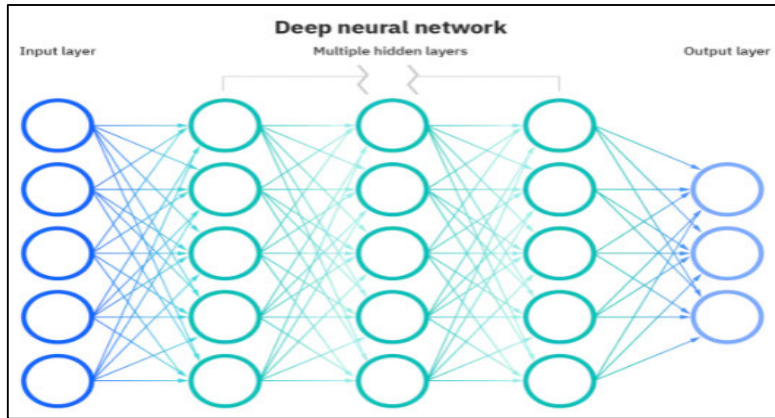
Deep Learning es una tecnología que está ganando espacio en el mundo de las redes neuronales y el aprendizaje automático que es utilizado en aplicaciones como reconocimiento de voz e imágenes y para la realización de predicciones, varios proyectos han dado fruto con la aplicación de esta técnica como, por ejemplo: el asistente de voz de Google o Siri [74].

Se define como Deep Learning a la estructura de red neuronal conformada por múltiples capas de procesamiento que puede aprovechar tanto un conjunto de datos etiquetados como no etiquetados, esto se debe a que Deep Learning entrena a la red para que reconozca automáticamente patrones básicos de los datos [74].

---

<sup>16</sup> **Learning Rate.** - Es un hiperparámetro que controla cuánto cambia el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo. Con un valor muy grande de learning rate será difícil encontrar coeficientes que minimicen la función de pérdida, mientras que un valor muy bajo provocará que el algoritmo demore más en encontrar la solución adecuada.





**Figura 1. 26.** Estructura Deep Learning [74].

## **2. METODOLOGÍA**

En el presente capítulo se describen los requerimientos y componentes necesarios de hardware y software para el diseño e implementación de estaciones meteorológicas de bajo costo en las zonas de Cumbayá, Calderón y Conocoto, las cuales servirán de fuente de datos para las predicciones meteorológicas de temperatura y humedad relativa mediante el uso de una red neuronal.

En primer lugar, se analizan los requisitos necesarios del sistema prototipo para cumplir las funciones previstas de adquisición, almacenamiento, transmisión, visualización y procesamiento de datos meteorológicos como son: temperatura, humedad relativa y presión atmosférica, con el fin de generar predicciones de temperatura y humedad relativa.

En segundo lugar, se examinan los diferentes componentes de hardware y software para su correspondiente selección de acuerdo a la disponibilidad en el mercado, además, se presenta de forma general la instalación e implementación de cada uno de los componentes electrónicos y programas o scripts asociados a la estación meteorológica para el funcionamiento automático y el monitoreo remoto y continuo de parámetros meteorológicos.

En tercer lugar, se desarrolla una aplicación de escritorio con múltiples funciones para el monitoreo de datos y eventos que ocurren en la estación, asimismo, se construye una red neuronal para las predicciones de temperatura y humedad relativa de las diferentes áreas previstas.

Finalmente, se construyen e instalan infraestructuras acordes para la protección de equipos en los valles mencionados, con el fin de realizar pruebas de funcionamiento del sistema prototipo expuestos a condiciones propias de cada zona y comprobar el correcto funcionamiento de las AWS para corregir posibles errores.

### **2.1. REQUERIMIENTOS DEL SISTEMA**

De manera general el prototipo debe cumplir con requisitos específicos para que el sistema se comporte de manera automática y permita adquirir, almacenar, transmitir, visualizar, pronosticar, monitorear y notificar de forma continua los datos de parámetros meteorológicos y eventos que ocurran en la estación. Además, la infraestructura debe cumplir con la mayoría de las características descritas en el capítulo 1 para que la AWS recopile información, no limite su funcionamiento y cuente con protección ante fenómenos naturales.

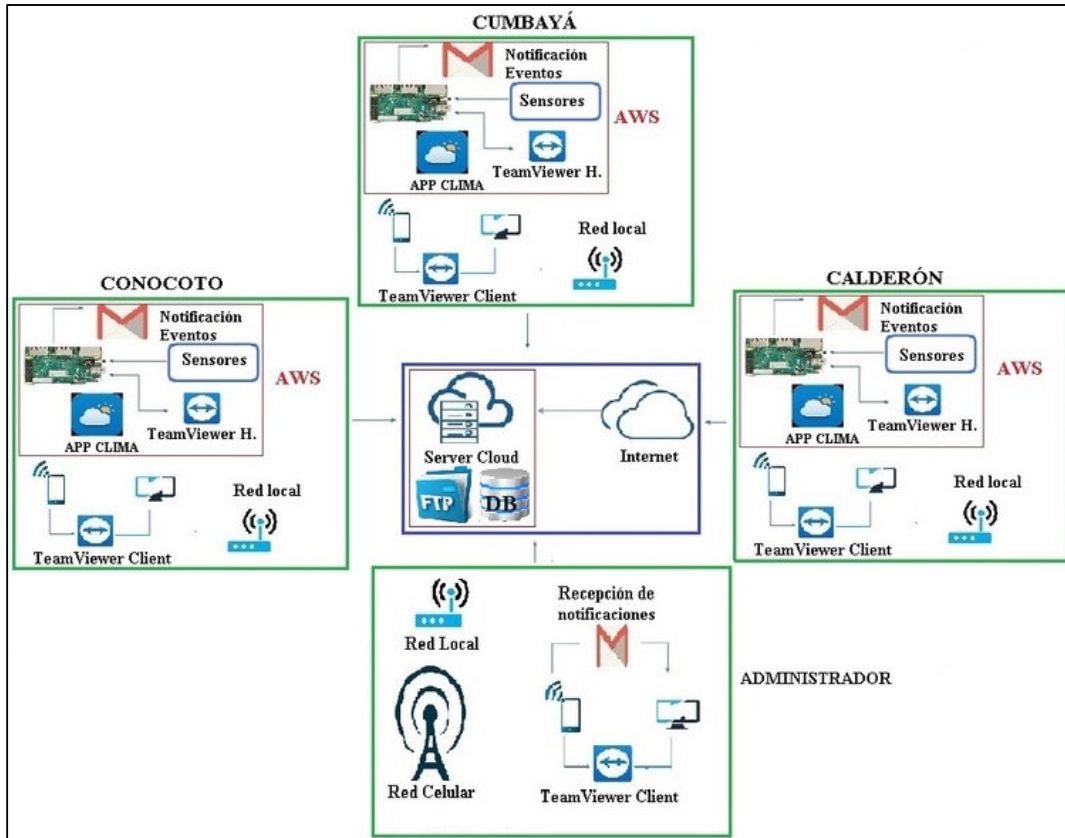


Figura 2. 1. Esquema de AWS para pronósticos meteorológicos.

Basado en el esquema de la Figura 2.1 se describen los requerimientos para la implementación de una AWS a continuación:

### 2.1.1. HARDWARE DE AWS

En base a soluciones previas de proyectos, experimentos o prácticas que de cierta manera tienen similitud con el presente trabajo con el objetivo de implementar una AWS de bajo costo, los componentes de hardware deben ser dispositivos robustos con poca probabilidad de fallo, fáciles de implementar y configurar que se encuentren disponibles en el mercado y sean asequibles.

Por otra parte, el ordenador principal del sistema prototipo de AWS debe cumplir con ciertas especificaciones de carácter obligatorio para que los procesos de adquisición, tratamiento y predicción de datos meteorológicos sean óptimos, por tanto, como se describió con anterioridad, el dispositivo debe tener las siguientes capacidades: adquirir y transmitir datos en tiempo real, tener conexión a internet, eficiencia energética, velocidad, asequibilidad,

disponibilidad y rendimiento. Por lo mencionado, la Raspberry Pi 4 B en comparación con sus versiones anteriores es el candidato ideal para la implementación de una estación meteorológica automática.

### 2.1.2. NOTIFICACIONES DE EVENTOS

Un método básico y efectivo de implementar alarmas para informar acontecimientos que ocurren en un sistema es a través del envío de correos electrónicos, la posibilidad de obtener información en tiempo real de los eventos de una estación meteorológica permite tomar medidas correctivas de los procesos de adquisición, almacenamiento y transmisión de datos para que el funcionamiento sea correcto y continuo.

El sistema de alarmas de la AWS permitirá comunicar los siguientes eventos:

- **Reinicio:** El dispositivo se reiniciará y notificará al administrador cuando la estación detecte un cero en el valor de temperatura o humedad, no adquiera datos en un tiempo determinado, no se detecten los sensores, exista alguna manipulación física o se produzca un corte de energía eléctrica.
- **Error de ceros persistentes y datos no detectados:** Este tipo de notificaciones se darán si ocurren los siguientes inconvenientes: fallo de sensores, conexiones incorrectas o inestables, scripts saturados.
- **Interpolación de datos:** Este evento se notificará al final del día si en la AWS existen datos faltantes.

### 2.1.3. ALMACENAMIENTO DE INFORMACIÓN

Los valores numéricos de parámetros meteorológicos de temperatura, humedad relativa y presión atmosférica se almacenarán en la memoria externa de la Raspberry Pi 4B, en archivos csv generados en la AWS.

El servicio de hosting web posee espacios de almacenamiento para que el administrador guarde toda la información tanto de datos como acontecimientos ocurridos en la estación meteorológica, por tanto, es necesario dos espacios de almacenamiento en los servidores alojados en la nube.

- **Base de datos:** El almacenamiento de datos meteorológicos en tiempos determinados y eventos que ocurren en la AWS se realizará en el servidor de base de datos alojado en el hosting web.

- **Administrador FTP:** El almacenamiento de archivos diarios e interpolados si fuera el caso se hará en el servidor FTP alojado en el hosting web.

#### **2.1.4. APLICACIÓN DE ESCRITORIO Y ACCESO REMOTO**

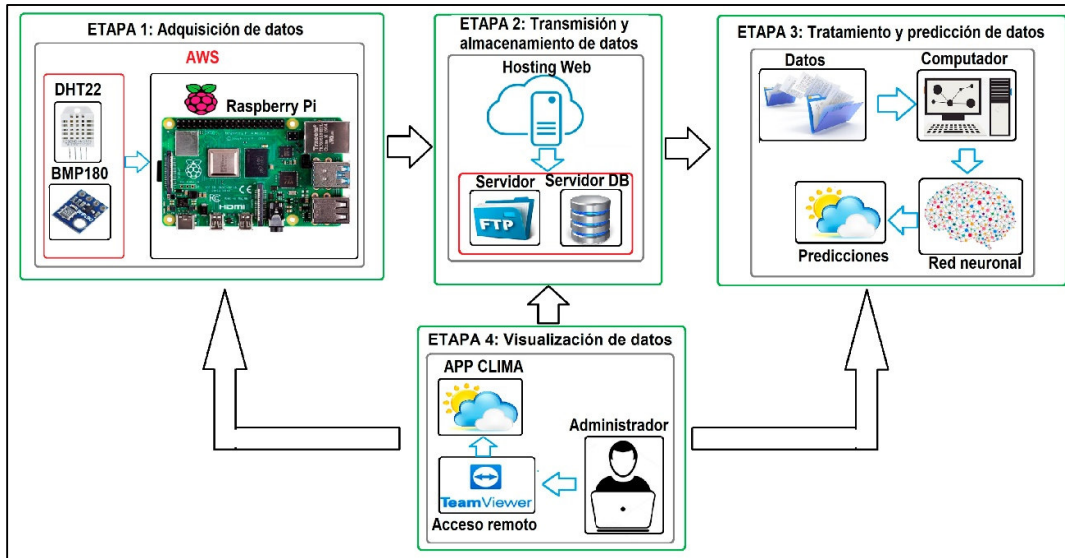
El diseño y la creación de una interfaz amigable al usuario es uno de los requisitos primordiales de la estación meteorológica automática, a través de la aplicación se podrá visualizar la información almacenada tanto en los archivos como en la base de datos, así como los sucesos ocurridos, lo que genera un historial del comportamiento de los nodos instalados en las diferentes zonas, información necesaria para el administrador, quien tomará las respectivas medidas correctivas en el caso de errores, entre otras funciones, mediante el acceso remoto implementado en la AWS.

## **2.2. ARQUITECTURA DEL SISTEMA**

En esta sección se presentan las etapas y componentes necesarios para diseñar una estación meteorológica automática de bajo costo y cumplir con los requerimientos antes mencionados, así como sus procesos para generar predicciones de temperatura y humedad relativa.

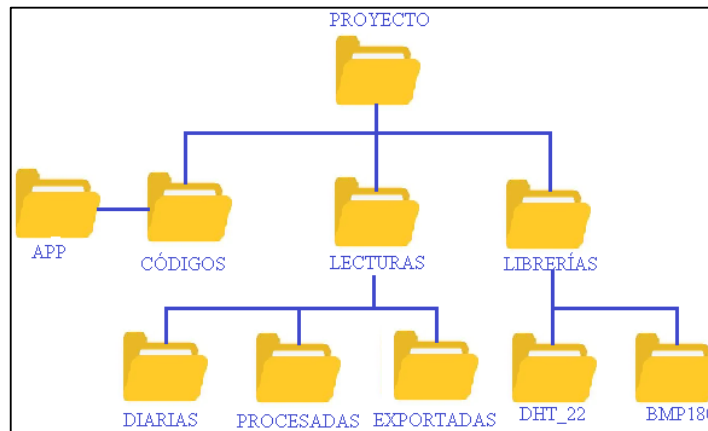
### **2.2.1. DISEÑO DEL SISTEMA PROTOTIPO**

De acuerdo a los recursos disponibles de hardware y software, el proceso de diseño del sistema prototipo se presenta en 4 componentes o etapas como se muestra en la Figura 2.2: etapa de adquisición datos, etapa de transmisión y almacenamientos de datos, etapa de tratamiento y predicción de datos, por último, la etapa de visualización y monitoreo de datos.



**Figura 2. 2.** Componentes del sistema prototipo.

Además, para la instalación, configuración y programación de los componentes y funciones a cumplir por la estación meteorológica, se genera carpetas y subcarpetas para mantener el orden y la jerarquía de programación y almacenamiento interno de datos meteorológicos como se muestra en la Figura 2.3, siendo la carpeta principal definida como “PROYECTO” quien abarcará todos los archivos y scripts.



**Figura 2. 3.** Jerarquía de carpetas en AWS.

### 2.2.1.1. Etapa de adquisición de datos

En esta etapa se recolectarán los datos de los parámetros meteorológicos de temperatura, humedad relativa y presión atmosférica a través de los sensores incorporados en el miniordenador Raspberry Pi 4B, información que se almacenará en archivos generados

por el ordenador de forma automática en la memoria externa del sistema, además se incorporarán funciones para minimizar errores en el funcionamiento continuo de la estación y en el caso de presentar problemas sea capaz de notificar al administrador de la AWS para su intervención inmediata.

#### **2.2.1.2. Etapa de transmisión y almacenamiento de datos**

El acceso a internet para la segunda etapa del sistema prototipo es fundamental, por tanto, la AWS debe contar con una conexión inalámbrica a la red local de cada zona. En esta fase se implementarán scripts que se ejecutarán de forma periódica en un tiempo definido, la comunicación entre el ordenador y los servidores alojados en el hosting web permitirá transmitir el último dato de las variables meteorológicas registrado en el archivo generado en la memoria externa de la Raspberry Pi 4B y será almacenado en una base de datos, asimismo, los archivos que contienen los datos recopilados durante todo el día se enviarán y guardarán en el servidor FTP al final de cada día.

#### **2.2.1.3. Etapa de tratamiento y predicción de datos**

En esta etapa se desarrolla un modelo adecuado de red neuronal para las predicciones de las variables meteorológicas de temperatura y humedad relativa y que la fiabilidad del prototipo sea óptima para generar resultados acordes a las condiciones climáticas propias de cada zona.

El tratamiento de datos de la información recopilada por la AWS y almacenada en el hosting web permitirá distribuir los datos en dos tipos:

- **Datos de entrenamiento:** Datos con volumen suficiente para generar información significativa, serán usados para entrenar la red neuronal, y la calidad de estos dependerá de la calidad de aprendizaje del modelo y el ajuste de los pesos y umbrales de la red neuronal.
- **Datos de prueba:** Datos de validación con los cuales se comprobará el funcionamiento del modelo y las predicciones del mismo.

#### **2.2.1.4. Etapa de visualización y monitoreo de datos**

En la etapa final del proyecto se desarrollará una interfaz amigable al usuario, a través de una aplicación de escritorio el administrador podrá visualizar todos los datos almacenados

en la tarjeta externa de la Raspberry o en la base de datos del hosting web, además, contará con varias funciones como: búsqueda de datos y archivos, conversor de archivos y programación de reinicio que facilitarán la administración de la AWS.

En esta etapa, además, se implementarán scripts para el monitoreo y notificaciones de errores o eventos que se pueden presentar en la estación meteorológica automática por las condiciones climáticas a las que este expuesta.

## **2.3. SELECCIÓN DE COMPONENTES**

En base al análisis de la información de componentes y requerimientos de hardware y software para la implementación de estaciones meteorológicas automáticas descrita en el capítulo 1, así como la disponibilidad y asequibilidad en el mercado de estos se eligen los dispositivos que integrarán el sistema prototipo.

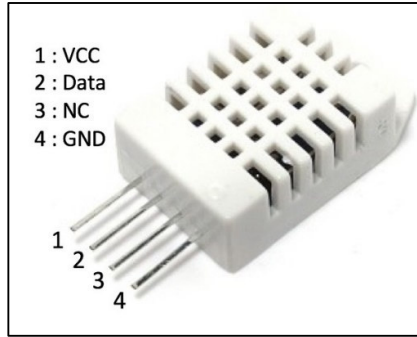
### **2.3.1. SELECCIÓN DE HARDWARE**

La Raspberry Pi 4B es el componente principal para la implementación de una estación meteorológica automática gracias a sus características y mejoras respecto a versiones pasadas, funciona como Unidad Central de todo el sistema prototipo que gestionará y controlará los procesos de adquisición, transmisión y visualización de datos.

#### **2.3.1.1. Sensor de temperatura y humedad relativa**

Para la selección del sensor de temperatura y humedad relativa se analiza los requerimientos de medición de estos parámetros presentados en las Tabla 1.2 y la Tabla 1.3 y la comparación de sensores descritos en la Tabla 1.4. La decisión de optar por el modelo DHT22 se debe a que presenta mejores prestaciones en comparación a otros sensores, la precisión del instrumento es mayor y el rango de operación es más amplio. Además, se encuentra con facilidad en el mercado, consume poca corriente, es fácil de implementar, configurar y reemplazar en caso de daño.



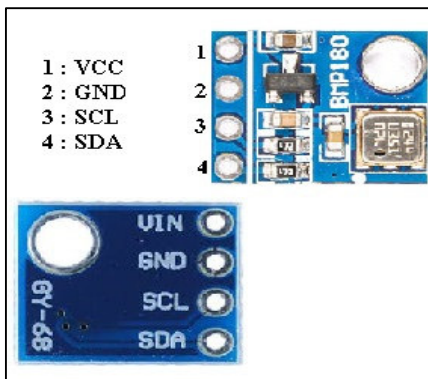


**Figura 2. 4.** Pines del sensor DHT22.

El sensor DHT22 (AM2302) como se muestra en la Figura 2.4, está conformado por cuatro pines: el pin1 para la fuente de alimentación (3V-6V), el pin2 permite la transmisión y recepción de datos mediante el protocolo single bus, el pin 3 no se utiliza y el pin 4 para conexión a tierra.

### 2.3.1.2. Sensor de presión

El análisis de requerimientos de la Tabla 1.5 y las comparaciones de la Tabla 1.6 respecto a sensores que cumplen las mismas funciones muestran características similares, sin embargo, cabe recalcar que la elección del instrumento se hace de acuerdo a la precisión que presentan, por tanto, gracias a este parámetro el sensor BMP180 es el adecuado para la implementación de la estación meteorológica. Al igual que el sensor DHT22, el sensor de presión atmosférica se encuentra con facilidad en el mercado a un precio accesible, es fácil de implementar, configurar y sustituir.



**Figura 2. 5.** Pines del sensor BMP180.

El sensor BMP180 cuenta con 4 pines como se observa en la Figura 2.5: el pin 1 para la fuente de alimentación (3.3V-5V), el pin 2 para conexión a tierra, el pin 3 (SCL-System Clock) para sincronización y el pin 4 (SDA-System Data) para movimiento de datos entre dispositivos. Cabe destacar que el sensor recibe y transmite datos a través del protocolo de comunicación serial I2C a través de 2 líneas de comunicación que las conforman el pin 3 y 4.

### 2.3.2. SELECCIÓN DE SOFTWARE

El capítulo 1 contiene toda la información del software que se va a utilizar para la implementación de una estación meteorológica automática, para dar idea de la elección de estos componentes se presenta un resumen en la Tabla 2.1.

**Tabla 2. 1.** Selección de software para AWS.

<b>Software</b>	<b>Descripción</b>
<b>Raspbian</b>	Sistema Operativo libre óptimo para el funcionamiento de Raspberry Pi.
<b>Python</b>	Lenguaje de programación interpretado y orientado a objetos, muy utilizado gracias a la facilidad de programación y contener ininidad librerías para desarrollar varios proyectos.
<b>LAMP</b>	Software conformado por una pila de cuatro tecnologías de software como: el sistema operativo Linux, servidor web Apache, servidor de base de datos MySQL y lenguaje de programación PHP.
<b>FTP</b>	Protocolo que permite la transferencia y descarga de archivos, cabe recalcar que el servidor FTP está alojado en el hosting web y a través de una cuenta FTP se realiza el proceso de almacenamiento y descarga de archivos en el servidor.
<b>SMTP</b>	Protocolo que permite el envío de notificaciones de alarmas por correo electrónico hacia el administrador y que este tome acciones correctivas en caso de errores.

#### 2.3.2.1. Acceso remoto

Mediante el análisis de la Tabla 1.9 en la cual se presentan características específicas de los diferentes servidores y aplicaciones que permiten acceder de forma remota a un ordenador desde cualquier dispositivo en cualquier momento y lugar con la única condición de contar con acceso a internet, se selecciona TeamViewer para el acceso remoto.

El administrador de las AWS no siempre llevará consigo un computador o laptop, sin embargo, el uso de Smartphones hoy en día es común y ayudarán a acceder a las estaciones meteorológicas automáticas y administrarlas desde cualquier lugar, es decir, la

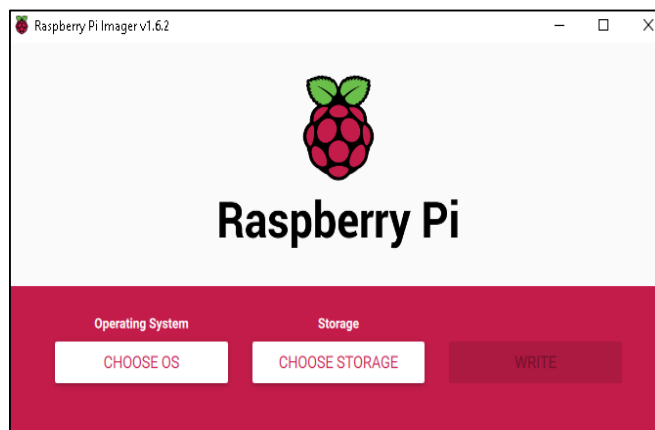
compatibilidad con múltiples dispositivos y la gratuidad son las ventajas que sobresalen de TeamViewer sobre los otros softwares de acceso remoto presentados.

## 2.4. INSTALACIÓN Y CONFIGURACIÓN DE EQUIPOS

### 2.4.1. ETAPA DE ADQUISICIÓN DE DATOS

#### 2.4.1.1. Sistema Operativo en Raspberry Pi

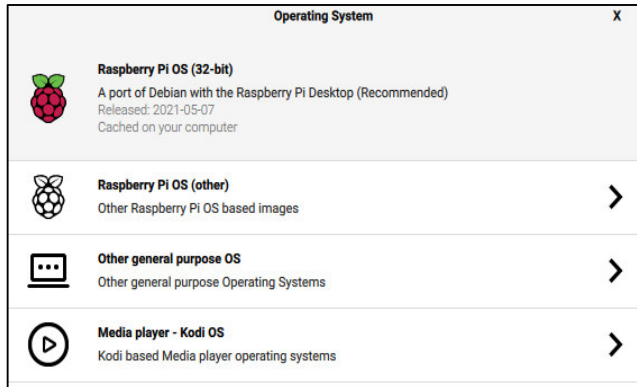
La instalación del Sistema Operativo (OS) del componente principal de una AWS (Raspberry Pi 4B) se realiza a través de la aplicación Raspberry Pi Imager (Figura 2.6) disponible para descargar desde la página oficial propio del microordenador en sistemas MAC, Windows y Linux, la aplicación hace que la instalación de la imagen del OS en una tarjeta SD sea rápida y fácil.



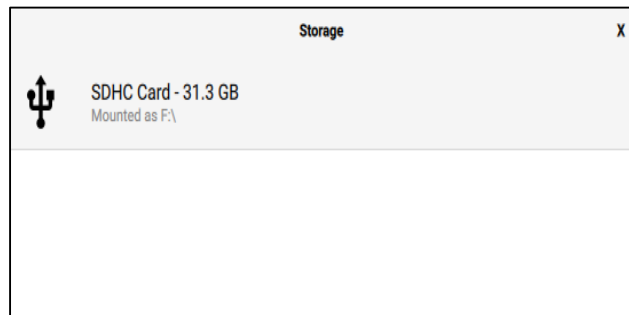
**Figura 2. 6.** Raspberry Pi Imager.

Antes de instalar el OS en la tarjeta SD se deben considerar las sugerencias de la página oficial de Raspberry como: la SD debe tener una capacidad mínima de 8Gb y formatear la tarjeta SD antes de su uso, este último proceso se realiza mediante el programa SD Card Formatter, que permite eliminar el contenido y las particiones de una SD de forma segura y rápida, sin que la tarjeta sufra daños.

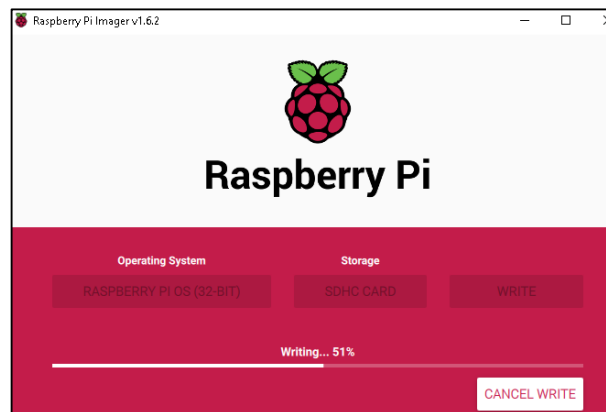
La aplicación presenta varios OS disponibles para diversos proyectos, entre otras funciones, el software por defecto es Raspberry Pi Desktop (Raspbian) basado en Linux, un entorno óptimo para el microordenador y fácil de utilizar, ofrece una interfaz gráfica de usuario, múltiples aplicaciones de productividad, herramientas de programación y configuración del sistema. Por tanto, los pasos a seguir para la escritura en la SD son: 1) Elegir el sistema (Choose OS) como la Figura 2.7, 2) Escoger el espacio de almacenamiento (Choose Storage) de acuerdo a la Figura 2.8 y 3) Escribir (Write) como la Figura 2.9.



**Figura 2. 7.** Paso 1: Sistema Operativo para Raspberry Pi.



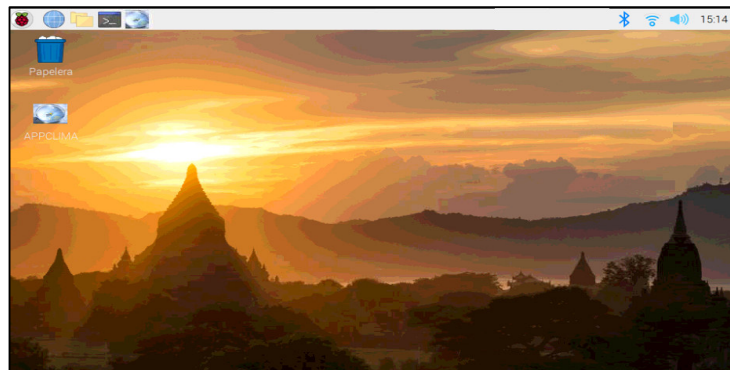
**Figura 2. 8.** Paso 2: Espacio de almacenamiento para OS.



**Figura 2. 9.** Paso 3: Escritura de OS en SD.

Una vez realizados los tres pasos antes mencionados, se inserta la tarjeta SD en la ranura de la Raspberry Pi 4B, para empezar a interactuar con el sistema operativo, es necesario conectar periféricos de entrada al microcontrolador como: pantalla, teclado y mouse, mediante el cable de alimentación eléctrica tipo C la Raspberry arranca o enciende y

muestra el entorno gráfico del sistema operativo de la Figura 2.10, base para el funcionamiento de la AWS.

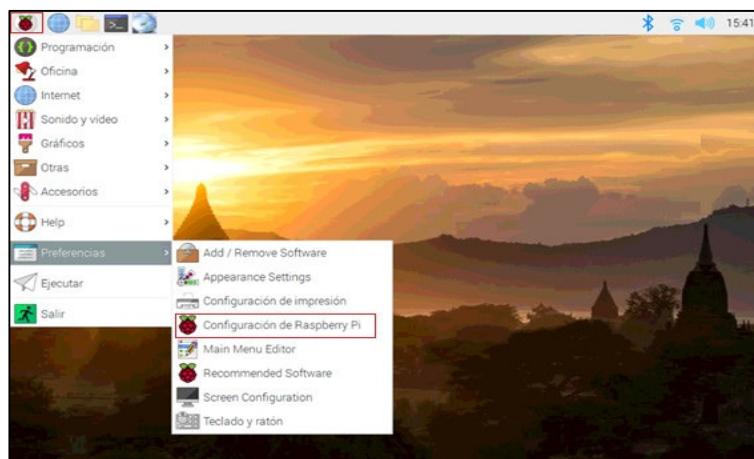


**Figura 2. 10.** Entorno gráfico de Raspberry Pi.

Al iniciar el microcontrolador muestra ventanas de configuraciones iniciales globales como zona horaria, contraseña, conexiones a internet ya sea mediante WI-FI o por cable, además, la versión lite de Raspbian instalada mediante la aplicación no incorpora ciertas funciones que necesariamente deben ser descargadas como las herramientas de Ofimática.

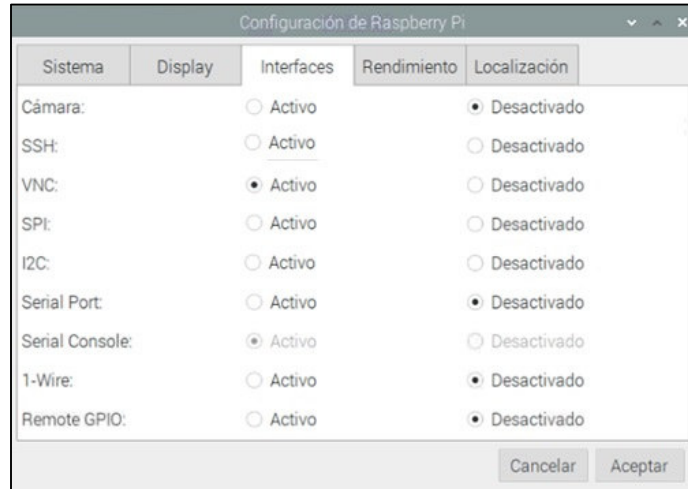
Para facilitar el control y trabajo sobre el microordenador, la Raspberry cuenta con un software libre basado en el modelo cliente-servidor para acceso remoto VNC (*Virtual Network Computing*), al cual se lo puede habilitar de dos maneras distintas.

La primera es a través del botón de inicio de la interfaz en donde nos dirigimos a la opción preferencias y configuración de Raspberry Pi respectivamente como se puede observar en la Figura 2.11.



**Figura 2. 11.** Opciones para habilitación de VNC en Raspberry Pi.

A continuación, se muestra una ventana en la cual nos dirigimos a la pestaña Interfaces, encontramos múltiples protocolos de comunicación que podemos habilitar para el desarrollo de diferentes proyectos, seleccionamos activo en la interfaz a utilizar (VNC) como en la Figura 2.12, aceptamos y reiniciamos el dispositivo.



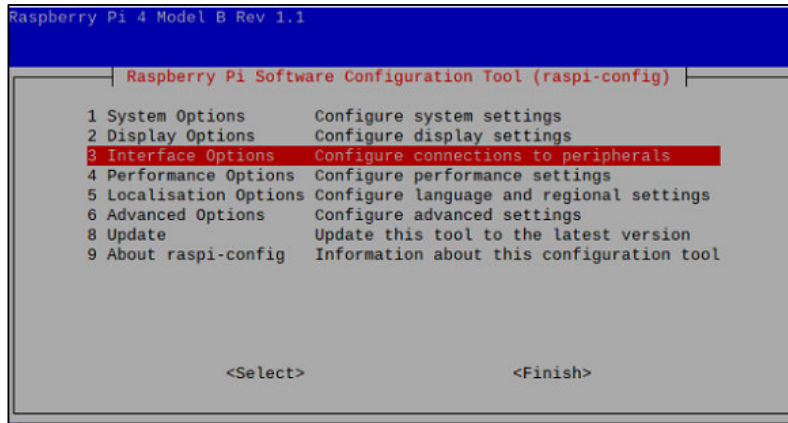
**Figura 2. 12.** Activación de VNC.

La segunda forma de activar VNC se puede lograr el Código 2.1 ejecutado en la aplicación LXTerminal de Raspberry.

```
pi@raspberrypi: ~ $ sudo raspi-config
```

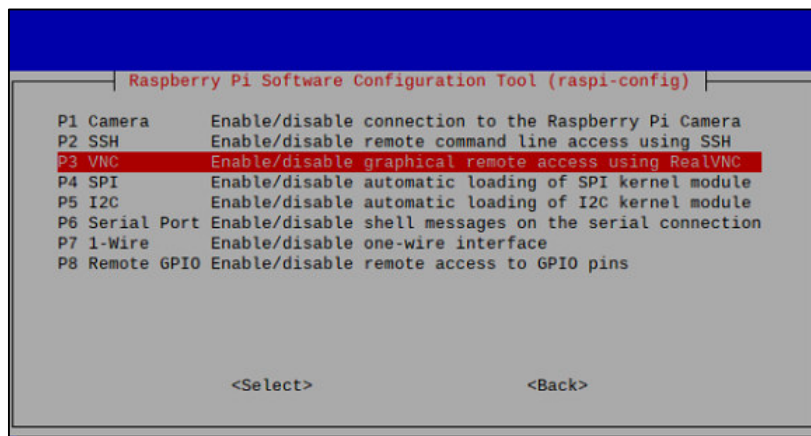
**Código 2. 1.** Acceso a la interfaz de configuración de Raspberry Pi.

Al ejecutar el comando se muestra una interfaz con múltiples opciones y herramientas de configuración, las cuales se pueden cambiar de acuerdo a las necesidades del proyecto, se selecciona la opción *Interface Options*, como se observa en la Figura 2.13.



**Figura 2. 13.** Interfaz de configuración de Raspberry Pi.

En la Figura 2.14 se muestra la interfaz que permite habilitar/deshabilitar VNC para el acceso remoto al microordenador.

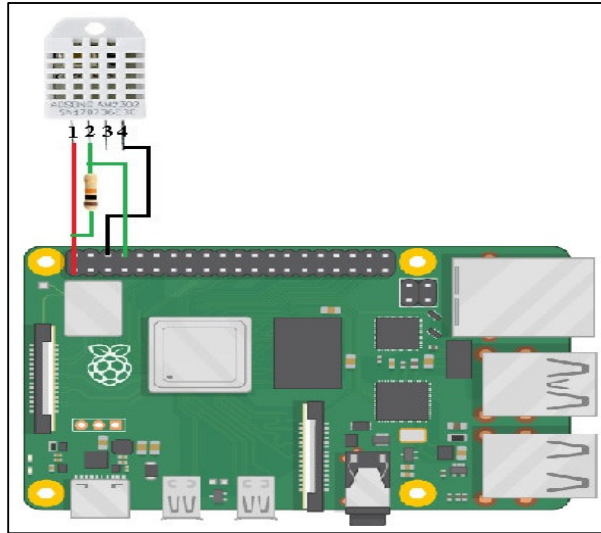


**Figura 2. 14.** Interfaz de habilitación de VNC.

Para acceder de forma remota a través de un dispositivo distinto se debe instalar el software libre VNC Viewer, al ingresar la dirección IP asignada al dispositivo se permitirá conectar a la interfaz gráfica de la Raspberry Pi mediante la autenticación con el usuario por defecto asignado “pi” y la contraseña definida en las configuraciones globales antes descritas.

#### **2.4.1.2. Sensor de Temperatura y Humedad DHT22**

El diagrama de conexión del sensor de temperatura y humedad se presenta en la Figura 2.15, de acuerdo a la distribución de pines presentados en la Figura 2.3.



**Figura 2. 15.** Diagrama de conexión DHT22.

Para el funcionamiento del sensor se procede a la instalación de la librería del sensor de humedad y temperatura DHT22 en base a las configuraciones de la página web github.com, y se procedió con la instalación de librerías actualizadas, por ejemplo, para el DHT22 se trabajará con la librería CircuitPython (Blinka).

- **Instalación CircuitPython**

Para la instalación de la librería CircuitPython se debe asegurar que el sistema operativo instalado en la Raspberry Pi pueda compilar y descargar extensiones de Python con pip, por tanto, para asegurar dicho procedimiento se ejecuta el Código 2.2 en LXTerminal.

```
pi@raspberrypi: ~$ sudo apt-get update && sudo apt-get upgrade -y
pi@raspberrypi: ~$ sudo apt-get install python3-pip
pi@raspberrypi: ~$ sudo python3 -m pip install --upgrade pip setuptools wheel
```

**Código 2. 2.** Comandos para la instalación de la herramienta pip.

Para la instalación de CircuitPython se debe ejecutar el Código 2.3:

```
pi@raspberrypi: ~$ pip3 install adafruit-circuitpython-dht
pi@raspberrypi: ~$ sudo apt-get install libgpiod2
```

**Código 2. 3.** Comandos para la instalación de CircuitPython-DHT

Para comprobar el funcionamiento del sensor y la librería instalada se crea un script como el Código 2.4 en base al código de [75] llamado simpletest.py dentro de la carpeta DHT\_22.



```

import time
import adafruit_dht
import board
import RPi.GPIO as GPIO

#Definicion del tipo de sensor y pin al que está conectado
DHT_SENSOR = adafruit_dht.DHT22
DHT_PIN = board.D14

dhtDevice = DHT_SENSOR(DHT_PIN)

while True:      #Ciclo infinito
    try:
        temperature = dhtDevice.temperature #Lectura de temperatura
        humidity = dhtDevice.humidity      #Lectura de humedad
        #Mostrar datos en pantalla
        print("Temp={0:0.2f}*C Humidity={0:0.2f}%".format(temperature, humidity))
    except RuntimeError as error:         #En caso de errores
        print("Fallo de lectura")
        time.sleep(2.0)
        continue
    time.sleep(5.0)

```

**Código 2. 4.** Script de comprobación de funcionamiento DHT22

Mediante LXTerminal se ejecuta el Código 2.5 para iniciar el script, como se puede observar el sensor funciona de forma correcta.

```

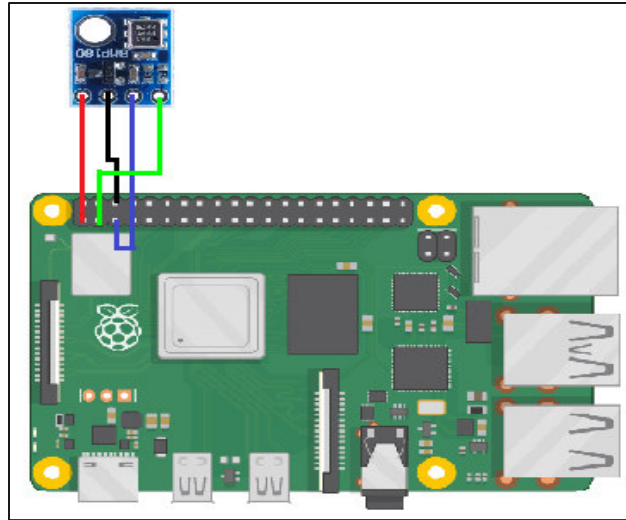
pi@raspberrypi:~/PROYECTO/LIBRERIAS/DHT_22 $ python3 simpletest.py
Temp=21.0°C Humidity=52.6%
Temp=20.8°C Humidity=52.5%
Temp=20.8°C Humidity=52.5%
Temp=20.8°C Humidity=52.5%
Temp=20.8°C Humidity=52.5%
Temp=20.7°C Humidity=52.3%
Temp=20.7°C Humidity=52.4%
Temp=20.7°C Humidity=52.4%
Temp=20.7°C Humidity=52.4%

```

**Código 2. 5.** Comando para verificar el funcionamiento del sensor DHT22.

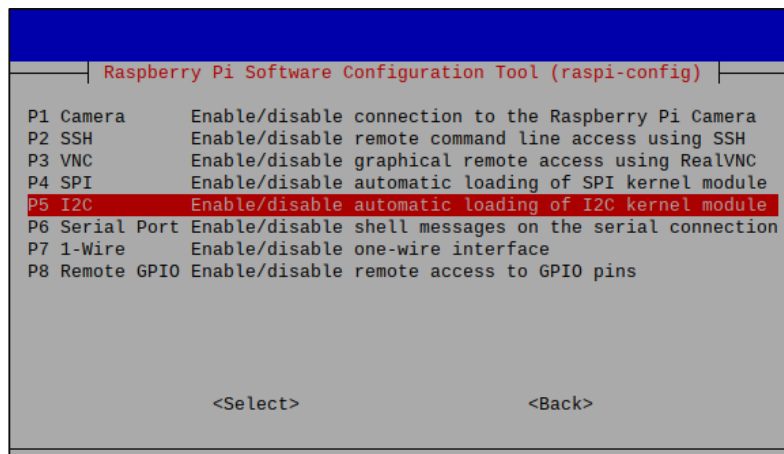
### 2.4.1.3. Sensor de Presión Atmosférica BMP180

Para la instalación y configuración del sensor BMP180 se utiliza de guía el diagrama de conexión de la Figura 2.16, tomando en cuenta la distribución de pines de la Figura 2.5.



**Figura 2. 16.** Diagrama de conexión BMP180.

Para la instalación y el funcionamiento del sensor BMP180 es necesario habilitar la comunicación I2C, para activarla se ejecuta el Código 2.1 y se elige la opción de la interfaz de la Figura 2.13, inmediatamente se muestra la interfaz de la Figura 2.17 y se lo habilita, es necesario reiniciar la Raspberry Pi para guardar los cambios realizados.



**Figura 2. 17.** Interfaz de habilitación I2C.

Para la instalación de la librería y las dependencias necesarias para utilizar el sensor de presión atmosférica se ejecuta el Código 2.6:

```
pi@raspberrypi: ~ $ sudo apt-get install git build-essential python-dev python-smbus
```

**Código 2. 6.** Comando para la instalación de dependencias para BMP180.

La librería para controlar el sensor BMP180 se encuentra disponible para descargar en la página oficial de github.com, la instalación de esta se basa en el proceso explicado en [76], a continuación, se presenta el Código 2.7 utilizado:

```
pi@raspberrypi: ~ $ cd PROYECTO/LIBRERIAS
pi@raspberrypi: ~ /PROYECTO/LIBRERIAS $ git clone https://github.com/adafruit/Adafruit_Python_BMP.git
pi@raspberrypi: ~ /PROYECTO/LIBRERIAS $ cd Adafruit_Python_BMP
pi@raspberrypi: ~ /PROYECTO/LIBRERIAS/Adafruit_Python_BMP $ sudo python3 setup.py install
```

**Código 2. 7.** Instalación de la librería BMP180.

Acto seguido se comprueba el funcionamiento del sensor y la instalación correcta de la librería, por ende, mediante el conjunto de Código 2.8 y el ejemplo ofrecido por la librería, se ejecuta el script *simpletest.py* y se observan los datos obtenidos del sensor.

```
pi@raspberrypi: ~ /PROYECTO/LIBRERIAS/Adafruit_Python_BMP $ cd examples
pi@raspberrypi: ~ /PROYECTO/LIBRERIAS/Adafruit_Python_BMP/examples $ python3 simpletest.py
Temp = 15.00 °C
Pressure = 72974.00 Pa
Altitude = 2682.55 m
Sealevel Pressure = 72994.00 Pa
```

**Código 2. 8.** Comandos de comprobación del funcionamiento del sensor BMP180.

## 2.4.2. ETAPA DE TRANSMISIÓN Y ALMACENAMIENTO DE DATOS

El segundo bloque es un conjunto de software que permitirán almacenar todos los datos y archivos en una base de datos y un servidor FTP respectivamente. Se ha decidido que, a pesar de instalar un servidor de base de datos en el nodo Raspberry Pi, este no alojará datos locales o tendrá páginas web desarrolladas, esto se debe a que toda la información será alojada en el Hosting Web. Para esto, la instalación del servidor LAMP es necesario para cumplir las funciones de comunicación y almacenamiento de datos.

### 2.4.2.1. Instalación Apache

La instalación del servidor web APACHE es muy sencilla y se lo realiza mediante el Código 2.9.

```
pi@raspberrypi: ~ $ sudo apt-get install apache2
```

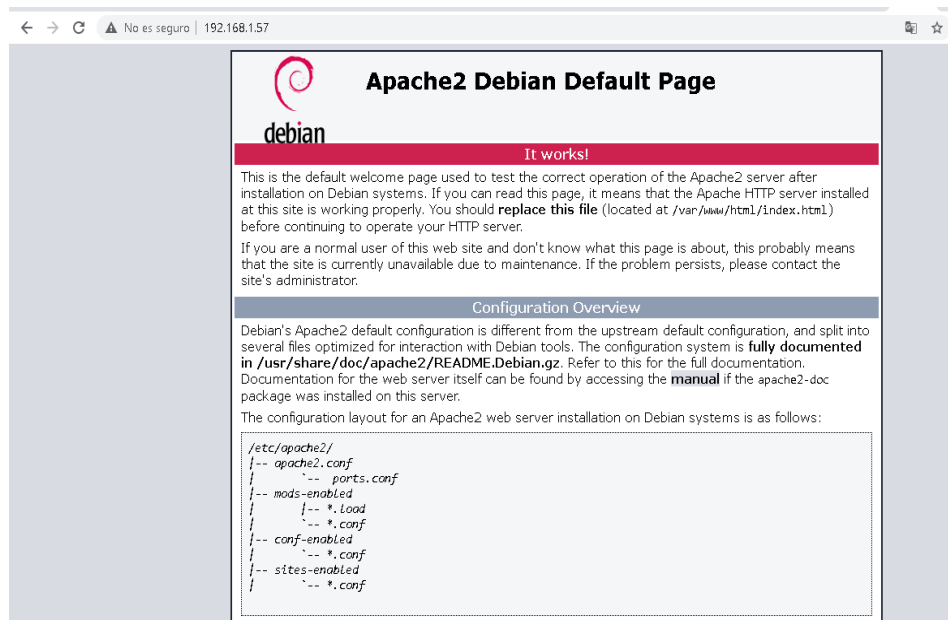
**Código 2. 9.** Instalación del servidor web Apache

Para comprobar el funcionamiento del servidor Apache se debe consultar la dirección IP asignada a la Raspberry mediante el Código 2.10, esta dirección será ingresada en un

buscador web teniendo como resultado la página de inicio de Apache como se muestra en la Figura 2.18.

```
pi@raspberrypi: ~$ hostname -I
```

**Código 2. 10.** Consulta de dirección IP.



**Figura 2. 18.** Página de inicio de Apache.

### 2.4.2.2. Instalación PHP

Junto con el servidor se instala el lenguaje de programación PHP que permite crear páginas web dinámicas y facilita la conexión con la base de datos para ello se ejecuta el Código 2.11:

```
pi@raspberrypi: ~$ sudo apt-get install php -y
```

**Código 2. 11.** Instalación de PHP.

Para la comprobación del enlace de php con apache se procede a crear un archivo de prueba nombrado "prueba.php" en la dirección /var/www/html con el Código 2.12:

```
pi@raspberrypi: ~$ sudo nano /var/www/html/prueba.php
```

**Código 2. 12.** Creación de prueba.php

El archivo se edita con sencillas líneas de comando como se muestra en la Figura 2.19.

```
GNU nano 3.2 /var/www/html/prueba.php
<?php
echo "Hola mundo!!!"; ?>
```

**Figura 2. 19.** Archivo de prueba PHP.

Posteriormente se reinicia apache con el Código 2.13 para enlazar php y apache.

```
pi@raspberrypi: ~$ sudo service apache2 restart
```

**Código 2. 13.** Reinicio del servidor Apache

En un navegador web, se coloca la dirección IP de la Raspberry / prueba.php y el resultado se presenta en la Figura 2.20.



The screenshot shows a web browser window with the address bar containing "192.168.1.57/prueba.php". The page content displays "Hola mundo!!!".

**Figura 2. 20.** Funcionamiento de Apache y PHP

### 2.4.2.3. Instalación MySQL

El servidor MARIADB es un sistema de gestión de base de datos que se deriva de MySQL, a pesar de que incorpora las funcionalidades de MySQL tiene algunas mejoras, como: motores de almacenamiento más eficientes. Para la instalación del gestor de base de datos y la dependencia para vincular php y mysql se ejecuta el Código 2.14.

```
pi@raspberrypi: ~$ sudo apt install mariadb-server mariadb-client php-mysql
```

**Código 2. 14.** Instalación del gestor de base de datos MariaDB.

MariaDB al igual que MySQL cuenta con la utilidad `mysql_secure_installation` para reconfigurar las opciones por defecto de fábrica y asegurar la instalación del servidor, esta utilidad permitirá agregar la clave root, además, afirmar presionando la "y" o negar presionando la "n" a diferentes operaciones desplegadas al ejecutar el Código 2.15 como: remover usuarios anónimos, no permitir que root inicie sesión remotamente, borrar la base de datos de prueba y recargar la tabla de privilegios.

```
pi@raspberrypi: ~$ sudo mysql_secure_installation
```

**Código 2. 15.** Reconfiguración de la instalación de MariaDB.

La comprobación del funcionamiento correcto, se realiza mediante las instrucciones del Código 2.16, para ello se ingresa la contraseña antes configurada y se comprueba que la conexión al servidor MariaDB esté activa con lo cual se pueden realizar diversas funciones, tal como crear una base de datos, agregar privilegios, entre otras; por ejemplo, el comando *show databases* consulta y muestra la base de datos por defecto del servidor.

```
pi@raspberrypi: ~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 57
Server version: 10.3.23-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.012 sec)
```

**Código 2. 16.** Comprobación de funcionamiento de MariaDB.

Finalmente, se debe instalar el conector del servidor MariaDB con Python a través de pip3 para lo cual se ejecuta el Código 2.17:

```
pi@raspberrypi: ~$ pip3 install mariadb
```

**Código 2. 17.** Instalación de conector MariaDB-Python

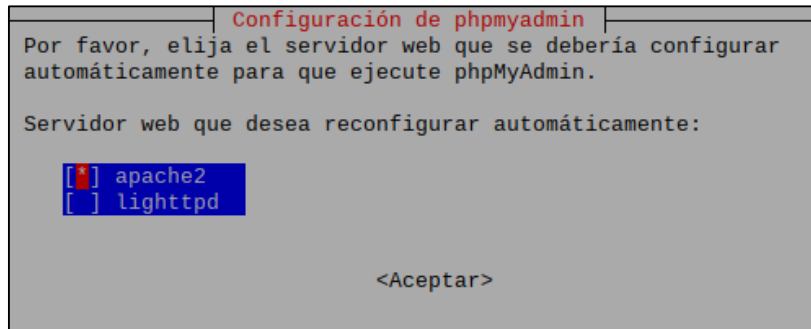
#### 2.4.2.4. Instalación de phpmyadmin

PhpMyAdmin es un software que facilita al administrador de la base de datos gestionar las tablas y la información a través de una interfaz gráfica y no mediante la línea de comandos. La instalación de las dependencias necesarias de phpmyadmin se realiza mediante el Código 2.18:

```
pi@raspberrypi: ~$ sudo apt install phpmyadmin php-mbstring php-gettext
```

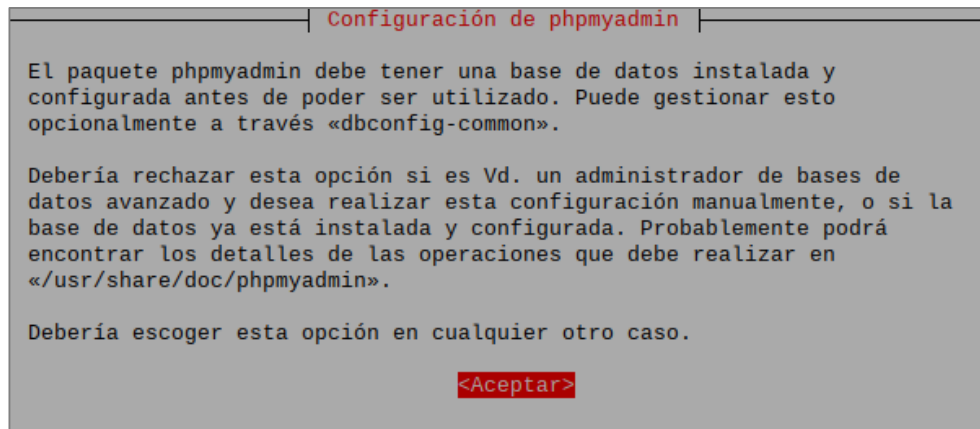
**Código 2. 18.** Dependencias para phpmyadmin

Acto seguido, despliega una pantalla para la selección del servidor para la ejecución de phpmyadmin, como se muestra en la Figura 2.21 se elige apache 2.



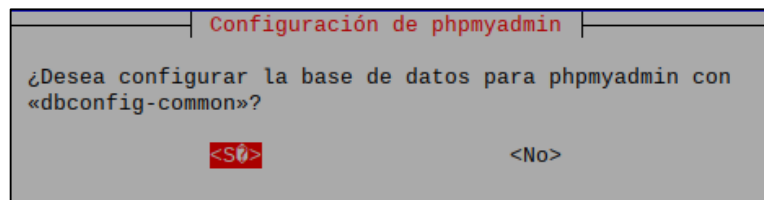
**Figura 2. 21.** Selección de servidor web.

Una vez seleccionado el servidor, se despliega una pantalla para la configuración de phpmyadmin, que brinda dos opciones de configuración como: automática o manual según las necesidades del instalador como se muestra en la Figura 2.22.



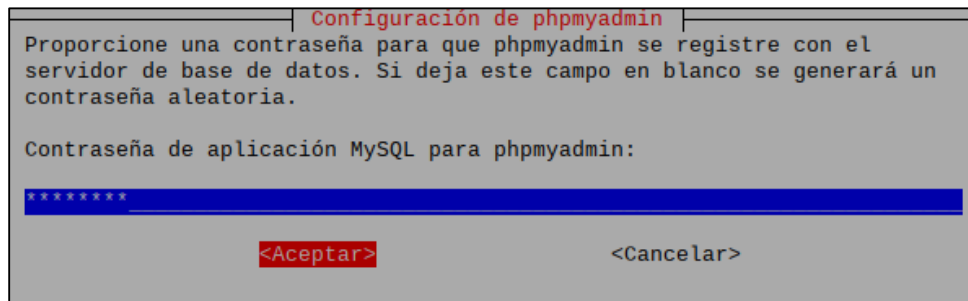
**Figura 2. 22.** Opciones de configuración.

Para el caso de estudio acepta la configuración automática a través de *dbconfig-common*, como se muestra en la Figura 2.23.



**Figura 2. 23.** Configuración de phpmyadmin con dbconfig-common.

El registro de la base de datos con phpmyadmin se realiza mediante el ingreso de una contraseña, como se muestra en la Figura 2.24.



The image shows a dialog box titled "Configuración de phpmyadmin". The text inside reads: "Proporcione una contraseña para que phpmyadmin se registre con el servidor de base de datos. Si deja este campo en blanco se generará un contraseña aleatoria." Below this, it says "Contraseña de aplicación MySQL para phpmyadmin:" followed by a blue input field containing eight asterisks. At the bottom, there are two buttons: "<Aceptar>" and "<Cancelar>".

Figura 2. 24. Contraseña de phpmyadmin.

Una vez realizado los procedimientos antes descritos, se reinicia el servicio de apache mediante el comando `sudo service apache2 restart`, para enlazar el servidor web y phpmyadmin, la comprobación del funcionamiento se realiza en un navegador web, se introduce la dirección IP del servidor/phpmyadmin, como se muestra en la Figura 2.25.

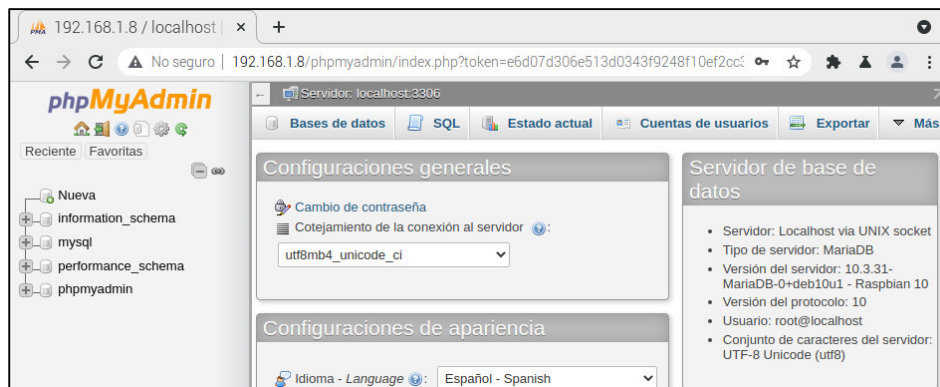


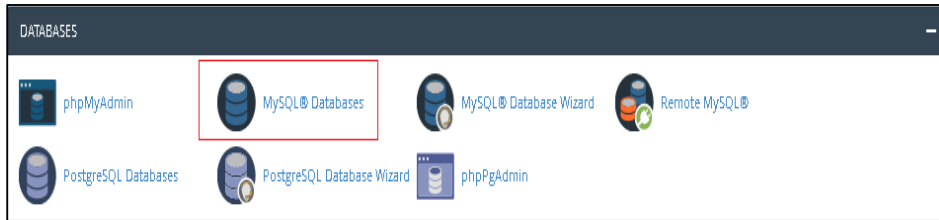
Figura 2. 25. Funcionamiento phpmyadmin.

#### 2.4.2.5. Creación de base de datos y usuarios en Hosting Web

El proceso para la creación de la base de datos y asociarla a un usuario administrador que tenga todos los privilegios para gestionarla es sencillo, ya que CPANEL presenta una interfaz amigable e intuitiva que facilita dichos procesos, por tanto, se describe de forma rápida la creación y configuración de estos dos parámetros importantes para el almacenamiento de datos.

1. Ir hacia el apartado *DATABASES* en el Hosting Web y elegir la opción *MySQL Databases* de acuerdo a la Figura 2.26.





**Figura 2. 26.** DATABASES Hosting Web

2. En la opción de crear nueva base de datos se ingresa el nombre de la base, la cual tendrá un prefijo “proysepn” por el nombre de usuario actual, que se trata de quien contrató el servicio de Hosting Web, de acuerdo a la Figura 2.27.

The image shows a form titled "Create New Database". It has a label "New Database:" followed by a text input field containing "proysepn\_". Below the input field is a blue button labeled "Create Database".

**Figura 2. 27.** Creación de una nueva base de datos.

3. Para la creación del usuario se dirige a la opción *MySQL Users* en la misma página, donde se ingresan las credenciales del usuario (nombre y contraseña), como se presenta en la Figura 2.28.

The image shows a form titled "MySQL Users" with the subtitle "Add New User". It contains several input fields: "Username" with "proysepn\_" entered, "Password", and "Password (Again)". Below these is a "Strength" indicator showing "Very Weak (0/100)" and a "Password Generator" button. At the bottom is a blue "Create User" button.

**Figura 2. 28.** Creación de usuario-administrador.

4. Para añadir o vincular al usuario a la base de datos, se debe dirigir al apartado *Add User To Database* en el cual se selecciona el usuario y la base de datos antes creados, como se muestra en la Figura 2.29. De acuerdo al procedimiento aparecerá, una nueva ventana como la de la Figura 2.30, permitiendo otorgar privilegios al administrador.

**Figura 2. 29.** Vinculación de usuario y base de datos creados.

**Figura 2. 30.** Selección de privilegios de administrador.

Finalmente, se debe comprobar que las bases de datos fueron creadas y están ligadas a un administrador que en este caso es “proysepn\_admin4”, como se muestra en la Figura 2.31.

Database	Size	Privileged Users	Actions
proysepn_Calderon	0 M	proysepn_admin4	Rename  Delete
proysepn_Conocoto	0 M	proysepn_admin4	Rename  Delete
proysepn_Cumbaya	0 M	proysepn_admin4	Rename  Delete

**Figura 2. 31.** Bases de datos creadas y asociadas al usuario-administrador.

### 2.4.2.6. Acceso remoto a la base de datos de Hosting Web

La configuración de acceso remoto a una base de datos permitirá al administrador acceder al servidor desde la AWS u otros dispositivos, para ello se procede con lo siguiente:

1. Ir al apartado DATABASES y MySQL Remote, de acuerdo a la Figura 2.32.

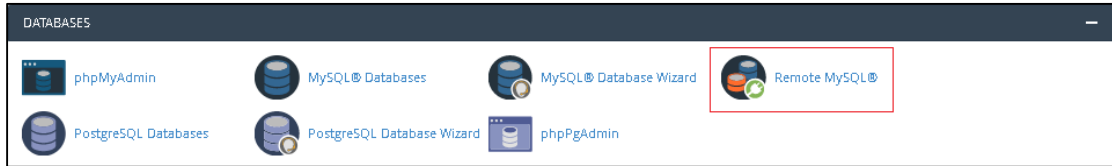


Figura 2. 32. MySQL Remoto.

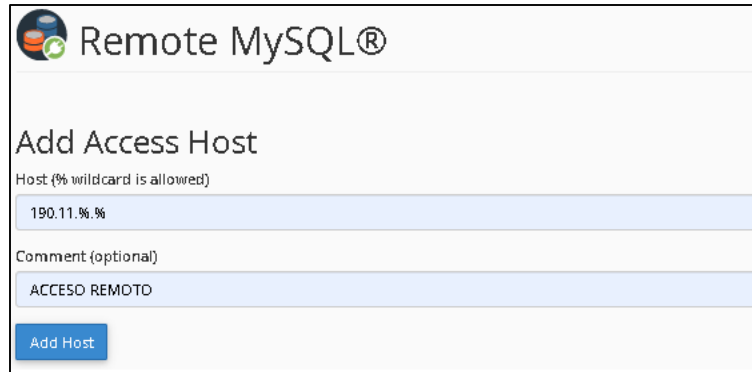
2. Consultar la IP pública de acceso a internet asignado por el ISP mediante la página [whatismyip.com](http://whatismyip.com), sitio web que muestra información de la red pública conectada, como se muestra en la Figura 2.33.



Figura 2. 33. Consulta de IP pública.

Sin embargo, como es de conocimiento, las IP no son constantes y cambian cada momento, no obstante, en pruebas de identificador de IP pública del proveedor CNT, los dos primeros octetos son de red y los dos siguientes para la dirección de host.

3. Configurar la IP pública en el apartado *Add Access Host* de acuerdo con la Figura 2.34.



**Figura 2. 34.** Agregar dirección IP.

A través de esta configuración, cualquier dispositivo conectado a una red que inicie con la dirección 192.11, sin importar el valor de los dos octetos restantes por el carácter % configurado, podrá acceder a la base de datos mediante las credenciales de usuario y contraseña.

Para acceder de forma remota desde la AWS hacia la base de datos alojada en el servidor en la nube se ejecuta el Código 2.19, mediante autenticación y en concordancia a los privilegios asignados el administrador podrá realizar cambios en la base de datos.

```
pi@raspberrypi: ~$ mysql -h "ip-servidor" -u "usuario" -p
```

**Código 2. 19.** Acceso remoto a base de datos.

#### **2.4.2.7. Instalación FTP**

La instalación del protocolo FTP para la conexión entre el cliente y el servidor FTP y el envío de archivos de información es sencilla, mediante la línea de Código 2.20:

```
pi@raspberrypi: ~$ sudo apt-get install ftp
```

**Código 2. 20.** Instalación protocolo FTP.

### **2.4.3. ETAPA DE TRATAMIENTO DE DATOS**

Para el tratamiento de datos es necesario instalar librerías que agilizarán el tratamiento de datos en funciones como: la interpolación de archivos con datos faltantes y los pronósticos de parámetros meteorológicos a través de la red neuronal.

### 2.4.3.1. Instalación de Pandas

La librería de código abierto Pandas permite el análisis y la manipulación de datos de forma rápida, potente y fácil de usar, la instalación se realiza con el Código 2.21.

```
pi@raspberrypi: ~$ sudo apt-get install python3-pandas
```

**Código 2. 21.** Instalación de librería Pandas.

Además, se instala una dependencia que permitirá optimizar las operaciones numéricas o cálculos científicos mediante el Código 2.22.

```
pi@raspberrypi: ~$ sudo apt-get install libatlas-base-dev
```

**Código 2. 22.** Optimizador de cálculos.

Finalmente, se instala el conector de la librería Pandas con el lenguaje de programación Python, mediante pip con el Código 2.23.

```
pi@raspberrypi: ~$ pip3 install pandas
```

**Código 2. 23.** Instalación del conector Pandas.

### 2.4.3.2. Instalación de Numpy

Pandas está desarrollado sobre la librería Numpy, por ello, para su correcto funcionamiento es necesario instalar esta librería, que permite realizar cálculos numéricos y procesar datos de matrices unidimensionales o multidimensionales más rápido, se instala mediante el código 2.24.

```
pi@raspberrypi: ~$ sudo apt-get install python3-numpy
```

**Código 2. 24.** Instalación de la librería Numpy.

Se procede a instalar el conector de Numpy con Python con el Código 2.25.

```
pi@raspberrypi: ~$ pip3 install numpy
```

**Código 2. 25.** Instalación del conector Numpy.

### 2.4.3.3. Instalación Anaconda

El procedimiento de predicción de datos se realiza en un computador por el administrador de las AWSs, por tanto, es necesario que el software Anaconda y la biblioteca Keras sean instaladas en el ordenador.

El software Anaconda es una distribución de Python que es principalmente usado para aplicaciones de ciencia de datos, por ejemplo: Machine Learning, análisis predictivo; entre otros. Cuenta con varias librerías preinstaladas por defecto y con su propio administrador de paquetes *conda* para la instalación de más librerías a través del código *conda install*.

La descarga del software con extensión *.exe* de *Anaconda Navigator* se realiza mediante la página oficial *anaconda.com* como se muestra en la Figura 2.35, la instalación en Windows es relativamente fácil, se procede como una instalación normal de un programa, en el cual se debe aceptar el acuerdo de licencia, elegir el software por defecto y la ubicación de la instalación.

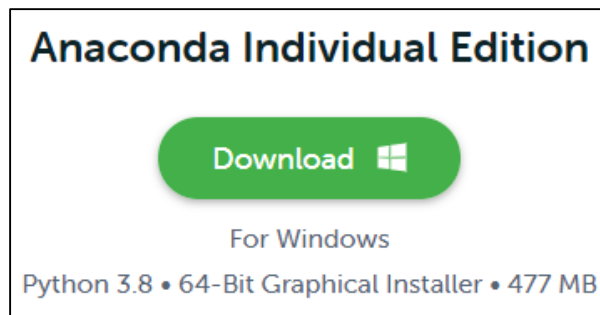


Figura 2. 35. Descarga de Anaconda para Windows.

Finalmente, se procede a abrir la aplicación de Anaconda Navigator como se muestra en la Figura 2.36, que posee múltiples herramientas de desarrollo de códigos basadas en Python como: Jupyter Lab, Jupyter Notebook; entre otros.

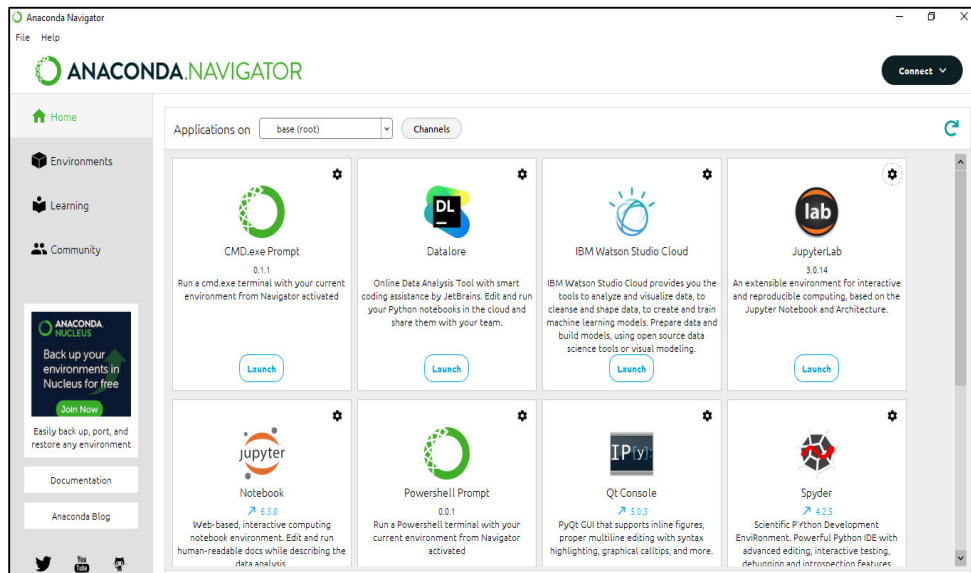


Figura 2. 36. Anaconda Navigator.

#### 2.4.3.4. Instalación de Keras

Keras es una biblioteca de código abierto escrita en Python que permite desarrollar complejos modelos de aprendizaje profundo, es decir, es una excelente biblioteca para construir por bloques diferentes arquitecturas de redes neuronales, incluidas convolucionales y recurrentes.

Una vez instalado el software Anaconda Navigator se procede a ejecutar *Anaconda Prompt* con los atributos de administrador, como se muestra en la Figura 2.37.

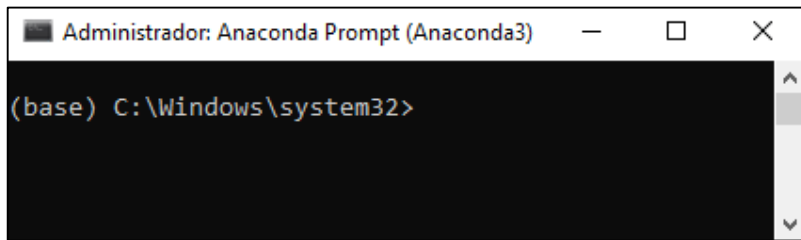


Figura 2. 37. Ejecución de prompt de Anaconda.

Posterior, se crea un entorno que permite tener un espacio para la instalación de paquetes y el desarrollo de las redes neuronales, este entorno se crea con el comando *conda create --name "nombre del entorno"*, como se muestra en el Código 2.26, donde *deeplearning* es el nombre del entorno creado.

```
(base) C:\Windows\system32>conda create --name deeplearning
```

Código 2. 26. Creación de entorno "deeplearning".

En el proceso de instalación se solicita especificar la ubicación del entorno en el cual se tipea "y" para aceptar la ubicación predeterminada y el entorno ha sido creado con éxito como se muestra en la Figura 2.38.

```
environment location: C:\ProgramData\Anaconda3\envs\deeplearning

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Figura 2. 38. Ubicación y creación del entorno.

Se procede a activar el entorno creado mediante la función *activate "nombre del entorno"*, como se muestra en el Código 2.27.

```
(base) C:\Windows\system32>activate deeplearning
```

**Código 2. 27.** Activación de entorno deeplearning.

La instalación de la biblioteca de redes neuronales Keras se realiza mediante el Código 2.28, durante este proceso se debe aceptar la descarga de los paquetes necesarios, para esto se debe tipear “y”.

```
(deeplearning) C:\Windows\system32>conda install -c anaconda keras
```

**Código 2. 28.** Instalación de Keras.

Sin embargo, se debe tomar en cuenta que a pesar de que la instalación de la biblioteca keras sea correcta, esta puede dar errores al momento de importar ya que es un entorno nuevo creado, por lo cual, se deben instalar herramientas adicionales como:

- Jupyter para desarrollar los scripts de predicción como se muestra en el Código 2.39.

```
(deeplearning) C:\Windows\system32>conda install jupyter
```

**Código 2. 29.** Instalación de Jupyter

- La librería Matplotlib para la generación visualización de gráficos a partir de datos con el Código 2.30.

```
(deeplearning) C:\Windows\system32>conda install matplotlib
```

**Código 2. 30.** Instalación de Matplotlib.

- Finalmente, la instalación de Pandas para la manipulación y análisis de datos con el Código 2.31.

```
(deeplearning) C:\Windows\system32>conda install pandas
```

**Código 2. 31.** Instalación de Pandas.

## 2.4.4. ETAPA DE VISUALIZACIÓN Y MONITOREO DE DATOS

### 2.4.4.1. Instalación de TeamViewer

El software TeamViewer es la manera más sencilla de acceder a ordenadores, servidores; entre otros, sin configuraciones largas y tediosas, la descarga del paquete se realiza a través de la página oficial mediante el Código 2.32.



```
pi@raspberrypi: ~$ wget https://download.teamviewer.com/download/linux/teamviewer-host_armhf.deb
```

**Código 2. 32.** Descarga de paquete TeamViewer.

La instalación de TeamViewer Host, que está diseñado para recibir conexiones y actuar como servidor, se realiza ejecutando el Código 2.33.

```
pi@raspberrypi: ~$ sudo dpkg -i teamviewer-host_armhf.deb
```

**Código 2. 33.** Instalación de TeamViewer.

Sin embargo, en el proceso de instalación se pueden presentar errores de ciertos paquetes específicos que no se pueden instalar correctamente, para corregir esto se ejecuta el Código 2.34.

```
pi@raspberrypi: ~$ sudo apt --fix-broken install
```

**Código 2. 34.** Corrección de errores en la instalación de TeamViewer.

En el proceso de corrección de errores aparece un acuerdo de licencia de TeamViewer que se debe aceptar, luego se procede a asignar una contraseña para el software con el Código 2.35 con la finalidad de otorgar seguridad de acceso remoto a la AWS.

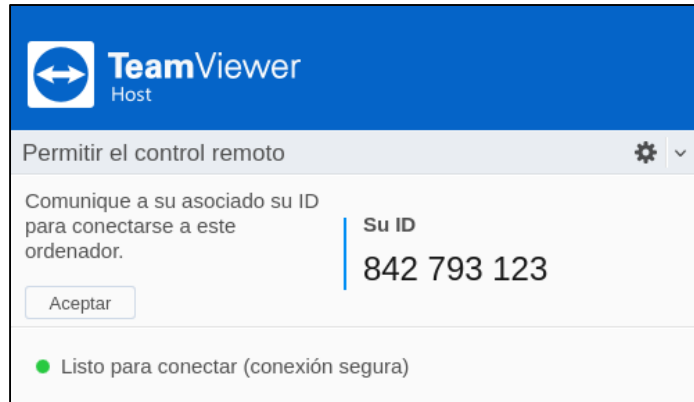
```
pi@raspberrypi: ~$ sudo teamviewer passwd "contraseña"
```

**Código 2. 35.** Asignación de contraseña a TeamViewer.

Finalmente, mediante el Código 2.36 se obtiene la ID única de 9 dígitos, como se muestra en la Figura 2.39, identificador con el cual se accede a la estación meteorológica automática ingresando la contraseña asignada anteriormente.

```
pi@raspberrypi: ~$ teamviewer status
```

**Código 2. 36.** Consulta de ID de AWS.



**Figura 2. 39.** ID AWS.

#### 2.4.4.2. Tkinter para crear aplicaciones

Existen librerías nativas e integrales de Python para la creación de aplicaciones, la librería por defecto instalada es Tkinter, que ofrece una infinidad de herramientas de control gráfico o widgets para la creación de GUIs (*Graphical Unit Interfaces*, Interfaces Gráficas de Usuario).

Los widgets son miniaplicaciones interactivas como botones, menús; entre otras herramientas gráficas, que presentan funcionalidades que permiten interactuar y facilitan el acceso a ciertas funciones y características de una GUI creada, por ejemplo, un botón programado para salir de la aplicación al hacer clic.

La descripción de los widgets utilizados en la aplicación de escritorio de la AWS se presenta en la Tabla 2.2.

**Tabla 2. 2.** Descripción de Widgets.

Widget	Descripción
<b>Button</b>	Control común que permite ingresar a una función o aceptar una operación al presionar sobre este.
<b>Label</b>	Etiquetas de texto estático que permiten nombrar partes de la pantalla en una aplicación.
<b>LabelFrame</b>	Actúa como un contenedor de otros widgets para mantener en orden las diferentes funciones de una aplicación.
<b>Combobox</b>	Es una lista de opciones que un usuario puede escoger para realizar distintas operaciones.
<b>MessageBox</b>	Son distintos tipos de mensajes que permiten verificar e informar sobre operaciones a realizar por el usuario de la aplicación, estos mensajes pueden ser: <ul style="list-style-type: none"> <li>- Showinfo</li> <li>- Showwarning</li> <li>- Showerror</li> <li>- Askyesno, entre otros.</li> </ul>

<b>Entry</b>	Es un cuadro de texto de una línea, que permite ingresar cualquier valor o texto que el usuario desee.
<b>Treewiew</b>	Son ventanas que permiten representar información tabulada y jerárquica.

### 2.4.4.3. Configuración para envío de notificaciones

Un protocolo que viene instalado por defecto o es nativo del OS es SMTP, mediante el cual se pueden enviar notificaciones al administrador de la AWS en caso de que un evento ocurra, por tanto, se presenta a continuación el Código 2.37 de prueba para la implementación del sistema de alarmas por envío de correo electrónico y su respectivo funcionamiento mostrado en la Figura 2.40.

Sin embargo, para que sea posible este proceso es necesario crear una cuenta de correo electrónico; gmail en el presente caso, y activar la función de acceso a aplicaciones poco seguras de esta, ya que por defecto Google bloquea esta función para inicios de sesión de dispositivos poco seguros.

```
import smtplib
from email.mime.text import MIMEText

def notificacioncorreo():
    corigen = ' ' #Correo de origen del mensaje
    contraseña = ' ' #Contraseña del correo de origen
    cdestino = ' ' #Correo/s destino
    msg = MIMEText("Prueba email con Raspberry Pi") #Mensaje
    msg['Subject'] = 'Notificacion' #Asunto
    msg['From'] = corigen #Quien envía
    msg['To'] = cdestino #Quien recibe
    serv = smtplib.SMTP('smtp.gmail.com:587') #Conexion por puerto 587 con gmail
    serv.ehlo()
    serv.starttls() #Protocolo de seguridad TLS (Transport Layer Security)
    serv.login(corigen, contraseña) #Acceso con correo y contraseña
    serv.sendmail(corigen, cdestino, msg.as_string()) #Envio de mensaje
    serv.close() #Desconexión del servidor gmail

notificacioncorreo()
```

**Código 2. 37.** Envío de notificación de prueba.



**Figura 2. 40.** Recepción de notificación de prueba.

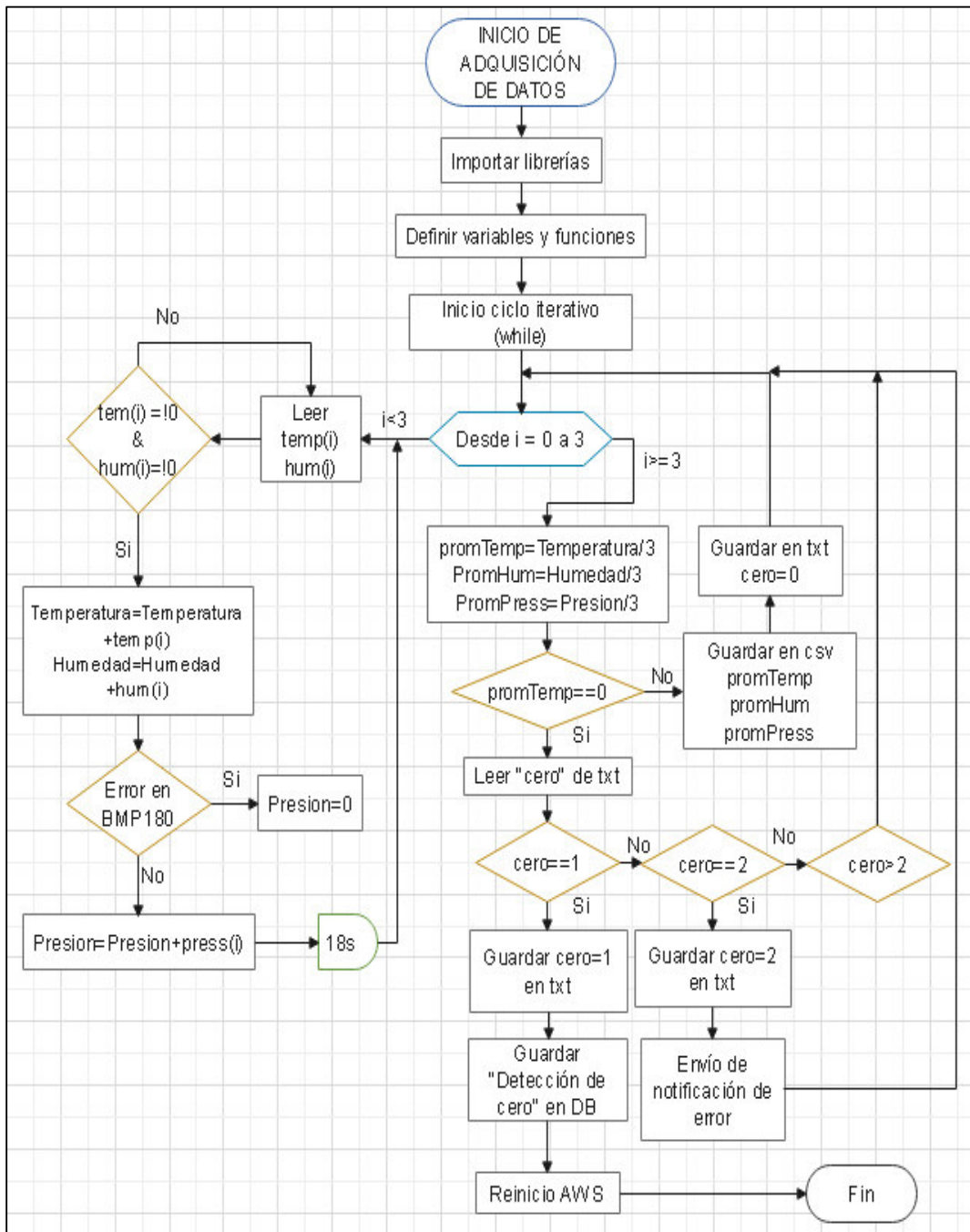
El Código 3.37 es el modelo de función base para la implementación de las diferentes alarmas por correo electrónico de la AWS. Una vez importadas las librerías necesarias para estructurar y enviar un correo electrónico, se procede a definir todas las variables necesarias como: correo emisor, contraseña y correo o correos receptores y el asunto de dicho correo, las funciones que permiten que el proceso se cumpla de manera automática y a cabalidad son:

- ***Smtplib.SMTP()***: permite la conexión con el servidor de correo electrónico en este caso gmail, a través del puerto 587.
- ***Ehlo()***: función de inicialización de saludo con el servidor de correo electrónico.
- ***Starttls()***: función definida por el protocolo TLS (*Transport Layer Security*, Seguridad de la capa Transporte), que da seguridad a la información mediante la encriptación para garantizar la comunicación por correo electrónico a través de internet.
- ***Login()***: función para acceder al correo emisor a través de la cuenta y contraseña.
- ***Sendmail()***: función que permite el envío del correo electrónico entre cuentas de correo electrónico emisor y receptor.
- ***Close()***: función que permite el cierre de la conexión al servidor.

## **2.5. REPRESENTACIÓN E IMPLEMENTACIÓN DE SCRIPTS**

### **2.5.1. LECTURA DE DATOS**

Para la implementación del script de lectura de datos de los sensores que miden las variables meteorológicas cada minuto sin interrupciones, se toma de referencia el diagrama de flujo de la Figura 2.41.



**Figura 2. 41.** Diagrama de flujo de adquisición de datos.

El script de adquisición de datos contiene funciones para definir el tipo de sensor a utilizar en la Raspberry Pi de acuerdo a las librerías instaladas. Para el sensor DHT22 se define el sensor y se inicializa mediante el comando `dhtDevice= adafruit_dht.DHTx(board.Dy)`,

donde “x” es el tipo de sensor DHT(11 o 22) a utilizar y “y” es el pin de propósito general conectado, como se presenta en el Código 2.38.

```
DHT_SENSOR = adafruit_dht.DHT22 #Tipo de sensor a utilizar
DHT_PIN = board.D14 #PIN al que está conectado
dhtDevice = DHT_SENSOR(DHT_PIN) #Inicialización del sensor
```

**Código 2. 38.** Inicialización de DHT22.

La adquisición de datos de temperatura y humedad respectivamente se realiza mediante el Código 2.39.

```
temperatura = dhtDevice.temperature #Lectura de valor de Temperatura
humedad = dhtDevice.humidity #Lectura de valor de Humedad
```

**Código 2. 39.** Adquisición de temperatura y humedad.

La inicialización del sensor de presión y la adquisición del dato de presión se realiza mediante le Código 2.40.

```
bmp = BMP085.BMP085() #Inicialización del sensor
presion = bmp.read_pressure() #Lectura de presión
```

**Código 2. 40.** Inicialización y adquisición del sensor de presión.

Los promedios de las variables meteorológicas adquiridas se almacenarán en archivos csv cada minuto, archivo diario que es generado automáticamente por la AWS, se abre cada que va a almacenar un dato mediante las funciones *open()* y *write()*, este proceso se presenta en el Código 2.41.

```
#Conjunto de codigos para apertura y escritura de
#datos en archivos .csv
now = datetime.now()
current_time = now.strftime("%H:%M:%S")
today = date.today()
nombreFile = "/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor, today)
file1 = open(nombreFile, "a")
str1 = "%s,%s,%d,%.1f,%.1f,%.1f\n"%(today, current_time, idSensor, promTemp, promHum, promPress)
file1.write(str1)
file1.close()
```

**Código 2. 41.** Escritura de datos meteorológicos.

Se debe tomar en cuenta que, los errores pueden presentarse en la adquisición de datos y generar valores nulos, por tanto, se implementa un registro de este evento que se almacena y se lee de un archivo de texto plano (.txt), en este caso se van a ejecutar eventos con sus valores correspondientes: 1 para almacenar el evento en la base de datos y

reiniciar el dispositivo, 2 para notificar al administrador acerca del error y 3 o más para tratar de recopilar información hasta lograrlo, la lectura y almacenamiento del registro se realiza mediante el Código 2.42 y el Código 2.43 respectivamente.

```
#Lectura de variable almacenada en caso de 0
leer=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt", 'r')
ceros=int(leer.read())
```

**Código 2. 42.** Lectura de registro de 0.

```
var=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt", "w")
var.write(str(ceros))
var.close()
```

**Código 2. 43.** Almacenamiento de registro de 0.

Por ende, si el valor del promedio de temperatura es 0 y el registro es 1, entonces quiere decir que se guarda el evento en la base de datos y se reinicia la AWS para tratar de resolver este inconveniente, por tanto, se presenta en el Código 2.44 la conexión a la base de datos que; se realiza con la instrucción *mariadb.connect()* de acuerdo a las credenciales de host, usuario, contraseña y base de datos. El almacenamiento del evento mediante el comando *INTO* en la tabla “Eventos” proceso que se realiza mediante la función *cursor.execute()* y el reinicio de la AWS con la instrucción *os.system('sudo shutdown -r now')*.

```
try:
    #Conexion con base de datos
    conexion = mariadb.connect(host="      ",
                               user="      ",
                               password="  ",
                               database="  ")
except mariadb.Error as e:
    os.system('sudo shutdown -r now')
    sleep(10)
#Almacenamiento de informacion en la tabla Eventos
cursor = conexion.cursor()
ceros = "INSERT INTO Eventos(Fecha, Hora, Sensor,Evento)VALUES
        ('%s', '%s', '%s', 'Cero detectado')"%(today,current_time, idSensor)
try:
    cursor.execute(ceros)
    conexion.commit()
except:
    conexion.rollback()
conexion.close()
os.system('sudo shutdown -r now') #Reinicio de AWS
sleep(10)
```

**Código 2. 44.** Conexión y almacenamiento de evento en la base de datos.

Por otro lado, si el inconveniente continúa luego del reinicio, entonces el registro tendrá un valor de 2 con el cual se envía una notificación de error al correo del administrador mediante la función modelo presentada en la configuración de envío de notificaciones del Código 2.37 y la AWS tratará de adquirir datos hasta su revisión correspondiente.

### 2.5.2. CHEQUEO DE DATOS

El script de chequeo de datos se implementa para supervisar de forma automática la adquisición de información que se ejecutará cada 5 min, con lo cual se tratará de detectar el caso de que por algún factor no conocido dé por terminado el proceso de recolección de datos de las variables meteorológicas, por tanto, el diagrama de flujo de este script se muestra en la Figura 2.42.

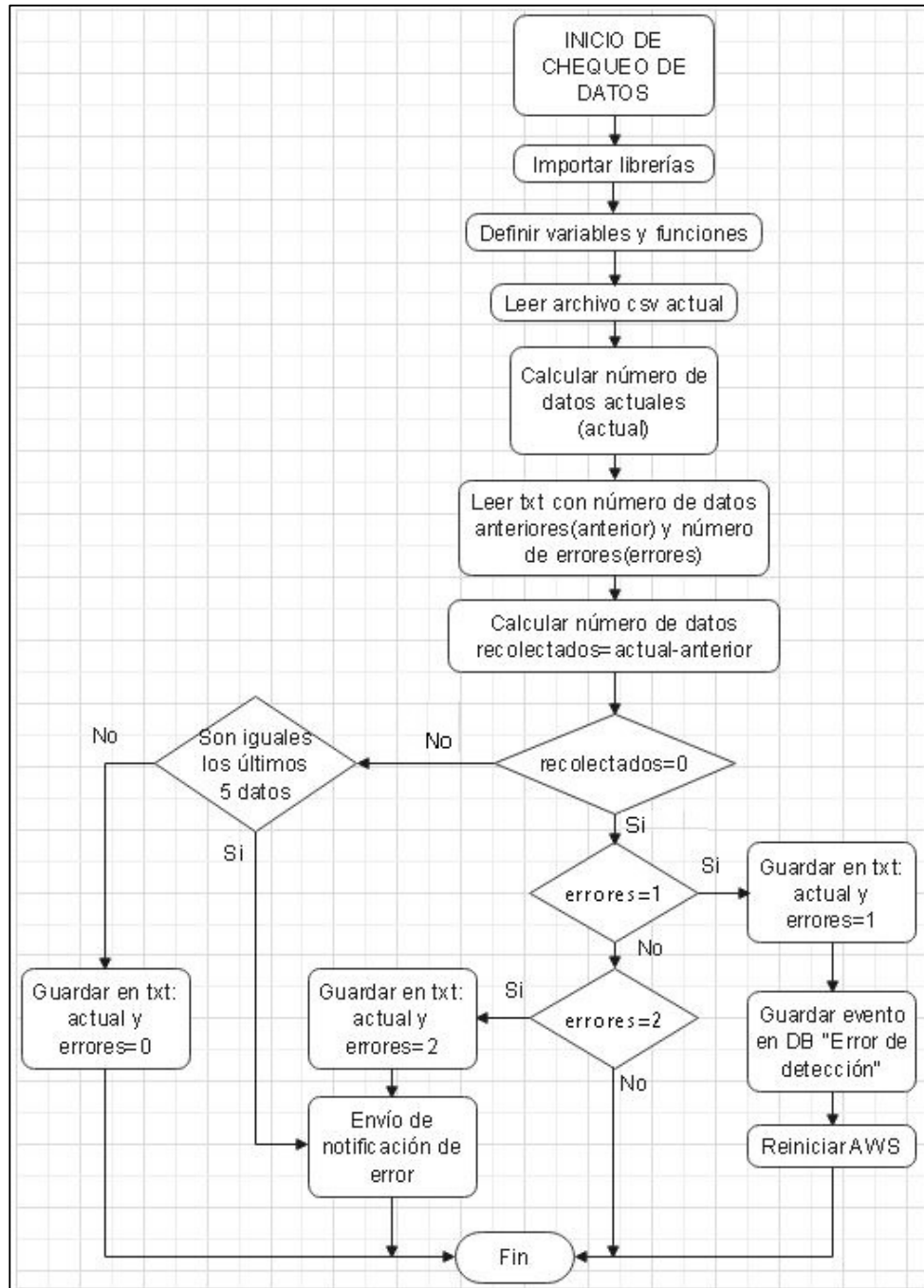


Figura 2. 42. Diagrama de flujo de chequeo de datos.



El script de chequeo de datos permite a la AWS monitorear la recopilación de información de las variables meteorológicas, en el caso de que un evento externo interrumpa en la adquisición de datos, las librerías Pandas y Numpy son las encargadas de verificar dicho proceso, para tomar acciones de acuerdo al registro de datos y errores que se almacenarán en un archivo de texto plano (.txt) como en el caso anterior.

La apertura del archivo para verificar los datos se realiza mediante la librería Pandas (pd) con la función `datos=pd.read_csv()`, y se almacena la información en columnas de datos estructurados mediante la función `pd.DataFrame()`, para calcular los datos totales del archivo csv actuales mediante la función `len()`, proceso que se muestra en el Código 2.45.

```
#Nombre del archivo a abrir
nombreFile = "/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,today)
file = open(nombreFile, "+a")
#Conteo del número de datos actuales
def lectura_datos():
    global datos_act
    cabecera = ["Fecha", "Hora", "Nodo", "Temperatura", "Humedad", "Presion"]
    datos=pd.read_csv(nombreFile, names = cabecera)
    df=pd.DataFrame(datos)
    print(df)
    datos_act = len(df)
    return(datos_act)
```

**Código 2. 45.** Lectura de datos totales actuales.

Además, se define una función para la lectura de datos anteriores y el número de errores almacenados como *strings* en el archivo de texto plano, a los cuales a través de la función *int()* se convierten en números enteros como se muestra en el Código 2.46, mediante estos y los datos actuales se determina el número de datos adquiridos y los errores cometidos, como se muestra en el Código 2.47.

```
#Lectura de datos anteriores y registro de errores
def lectura_variable():
    global datos_ant
    global num_errores
    with open("/home/pi/PROYECTO/CODIGOS/VARIABLES/check_datos.txt", "r") as f:
        for line in f:
            datos_ant, num_errores = line.split(" ")
            datos_ant = int(datos_ant)
            num_errores = int(num_errores)
    return(datos_ant, num_errores)
```

**Código 2. 46.** Lectura de datos pasados y número de errores.

```
datos_ant, num_errores = lectura_variable()
datos_act = lectura_datos()
numero_datos = datos_act - datos_ant
```

**Código 2. 47.** Número de datos adquiridos y errores cometidos.

Por tanto, como en el caso anterior, si se tienen 0 datos adquiridos, el valor de error 1 se guarda en el evento “Error de detección” en la base de datos y la AWS se reinicia. Por otro lado, si el valor de error es 2, entonces se notificará al administrador a través del correo electrónico, mediante las configuraciones descritas en el Código 2.44 y el Código 2.37 respectivamente.

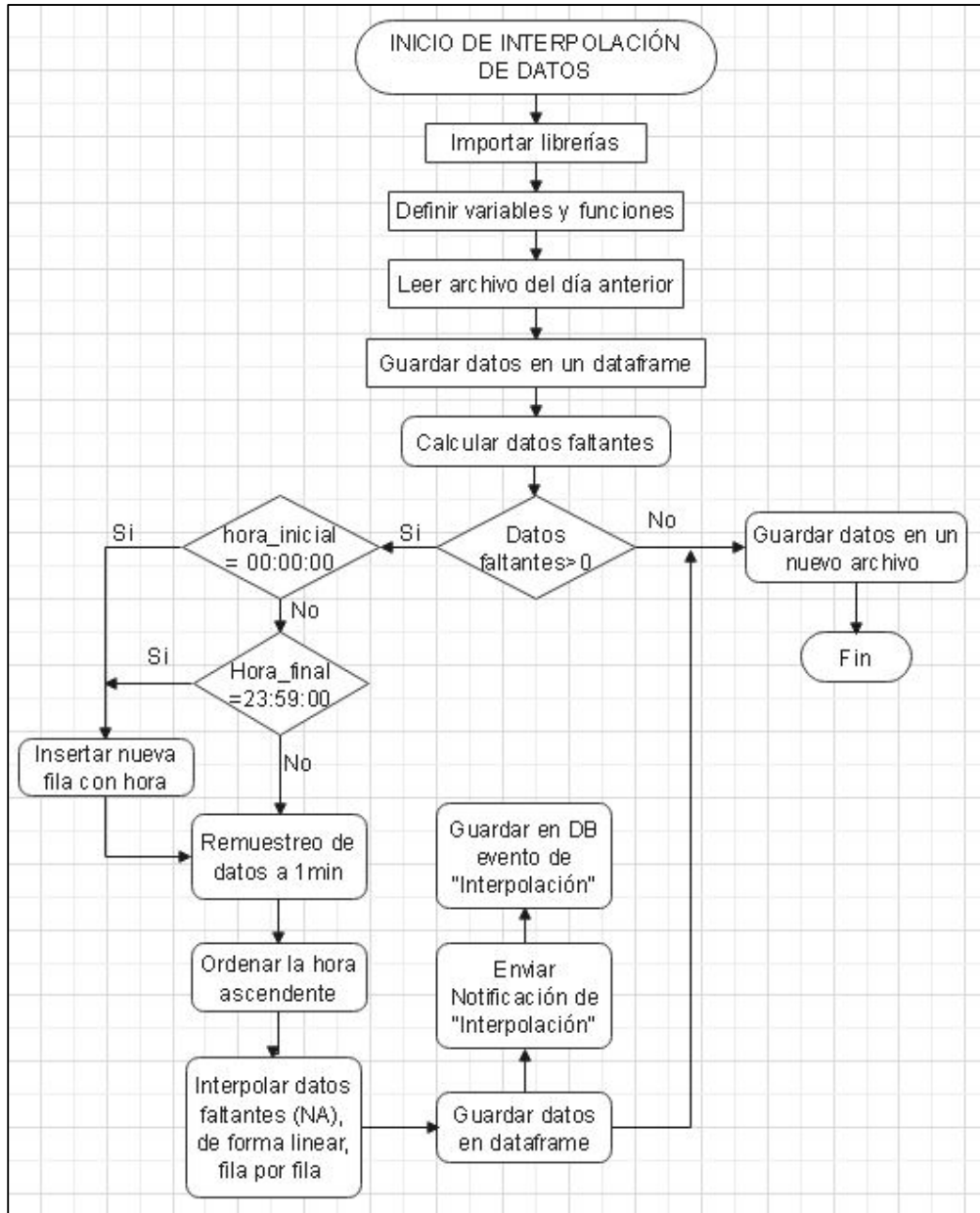
Además, puede haber fallos de conexión de los sensores o un evento desconocido que resulte en que, el último dato se esté repitiendo un número indefinido de veces, por ello se incorpora una metodología básica de comparación de los últimos cinco datos de temperatura y humedad del archivo seleccionados con la función *iloc[]* de acuerdo al Código 2.48 y en el caso de ser iguales se informará mediante correo al administrador.

```
for i in range (0, 5):
    datoT[i]=datos.iloc[-i-1, 3]
    datoH[i]=datos.iloc[-i-1, 4]
if datoT[0]==datoT[1]==datoT[2]==datoT[3]==datoT[4]:
    if datoH[0]==datoH[1]==datoH[2]==datoH[3]==datoH[4]:
        print("Fallo")
        notificacionesensor()
```

**Código 2. 48.** Comparación de últimos datos.

### 2.5.3. INTERPOLACIÓN

El diagrama de flujo de la Figura 2.43 representa el proceso para la elaboración del script de interpolación para el caso de presentarse en la AWS datos faltantes, este proceso se ejecutará al final de cada día.



**Figura 2. 43.** Diagrama de flujo para la interpolación de datos.

Un sistema no es eficiente al 100% y en este caso la AWS no siempre va a adquirir los 1440 datos al día, por ello, se incorpora un script de interpolación de datos con el cual se rellenan los datos faltantes en el archivo.

En primer lugar, se identifican los archivos en los cuales se comprobará si existen los datos faltantes al final de cada día (nombreFile) y el nuevo archivo (processFile) en el que se almacenarán todos los datos adquiridos e interpolados mediante el Código 2.49.

```
#archivo actual
nombreFile = "/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,yesterday)
#Nuevo archivo
processFile="/home/pi/PROYECTO/LECTURAS/PROCESADAS/Sensor%s_Lectura%s.csv"%(idSensor,yesterday)
file = open(processFile, "+a")
```

**Código 2. 49.** Definición de archivos.

Los procesos de lectura del archivo csv, almacenamiento de datos en un *dataframe* y el cálculo de datos faltantes es similar a los descritos en los códigos presentados con anterioridad, el script incorpora nuevas funciones las cuales permiten definir horas, definir filas e insertar estas, ya sea el caso en que la hora inicial (00:00:00) y final (23:59:00) difieran en el archivo.

Por ejemplo: la definición de una hora se realiza mediante el Código 2.50 con la función *astype()*.

```
#hora de inicio
horainicio = '%s 00:00:00'%yesterday
horainicio = np.datetime64(horainicio)
horainicio = horainicio.astype(datetime)
```

**Código 2. 50.** Definición de una hora.

La inclusión de una nueva fila en el archivo se realiza mediante el Código 2.51 con la función *pd.Series()* que contiene valores *numpy* nulos (*np.nan*) que serán reemplazados por datos o valores interpolados, el Código 2.52 con la función *append()* permite insertar la nueva fila en el archivo.

```
#Nueva fila
filanueva=pd.Series([horainicio, horainicio,
                    idSensor, np.nan,np.nan, np.nan],
                    index=['Fecha', 'Hora', 'Nodo',
                          'Temperatura', 'Humedad', 'Presion'])
```

**Código 2. 51.** Definición de una fila.

```
#Nueva fila insertada
df=df.append(filanueva, ignore_index=True, sort=False)
```

**Código 2. 52.** Inserción de nueva fila.

La información de variables meteorológicas se adquiere cada minuto, por ende, si existen datos faltantes y es necesario interpolar, el intervalo de tiempo entre datos debe ser el

mismo, mediante la función *resample()* se remuestrea las series de tiempo en 1min a la columna “Fecha” definida como índice por el comando *set\_index()*, terminado dicho proceso se reestablece al índice por defecto del archivo con el comando *reset\_index()* y se ordenan los datos de forma ascendente con la función *sort\_values()*, este proceso se muestra en el Código 2.53.

```
#Remuestreo y orden ascendente de datos
df=df.set_index('Fecha').resample('1min').mean()
df=df.reset_index(level=['Fecha'])
df = df.sort_values(by=['Fecha'], ascending=[True])
```

**Código 2. 53.** Remuestreo de series de tiempo.

Los datos remuestreados se guardan con valores numpy nulos, los cuales se interpolan mediante la función *interpolate()*, que utiliza varias técnicas de interpolación como: lineal, polinomio, baricéntrico; entre otros métodos y se lo configura con el comando *method()*; en este caso; se elige lineal, ya que las variaciones de los datos meteorológicos en 1min no son drásticos, el relleno de datos se realiza fila por fila de acuerdo a la función *axis()* con valores cercanos a estos mediante las funciones *ffill()* y *bfill()*, relleno hacia adelante y hacia atrás respectivamente, como se muestra en el Código 2.54.

```
df = df.interpolate(method="linear", axis=1).ffill().bfill()
```

**Código 2. 54.** Interpolación de datos.

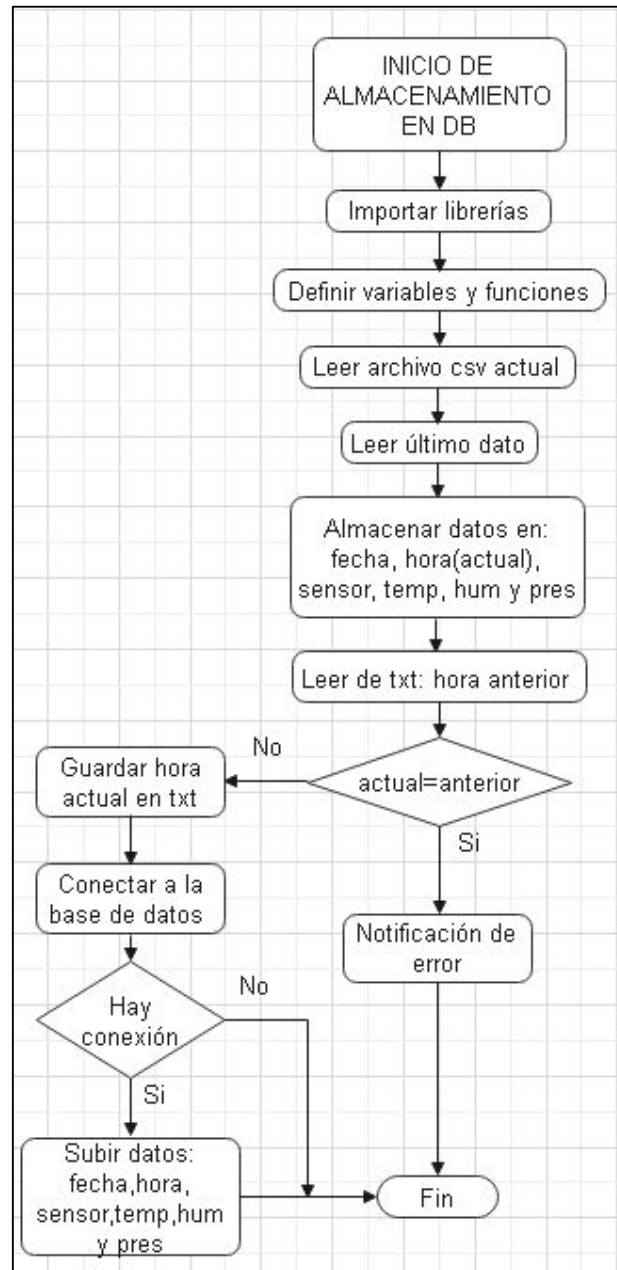
Finalmente, luego del proceso de interpolación los datos se guardan en una estructura de datos o *dataframe* mediante la función *Dataframe()* en una nueva ubicación y nombre del archivo definido en “processFile” a través del comando *to\_csv()*, este proceso se muestra en el Código 2.55.

```
df=pd.DataFrame(df)
df.to_csv(processFile, index=False)
```

**Código 2. 55.** Archivo interpolado.

### 2.5.4. ALMACENAMIENTO EN BASE DE DATOS

El diagrama de flujo de la Figura 2.44 representa el proceso para la implementación de un script que permite la conexión y almacenamiento de datos en la base de datos alojada en el Hosting Web.



**Figura 2. 44.** Diagrama de flujo de almacenamiento en la base de datos.

Los procesos son similares a los códigos anteriores al leer los datos desde el archivo y almacenarlos en un *dataframe*, la diferencia que radica en este código es la lectura del

último dato del archivo que se realiza cada 10min y su almacenamiento en variables, el cual se realiza mediante el Código 2.56 con la función *iloc[]* que permite escoger un valor de una determinada columna, por ejemplo: *iloc[-1,0]* escoge el último valor de la columna 0 y lo almacena en la variable “fecha”.

```
#Lectura último dato
fecha = df.iloc[-1,0]
hora = df.iloc[-1,1]
temp = df.iloc[-1,3]
hum = df.iloc[-1,4]
pres = df.iloc[-1,5]
```

**Código 2. 56.** Lectura del último dato

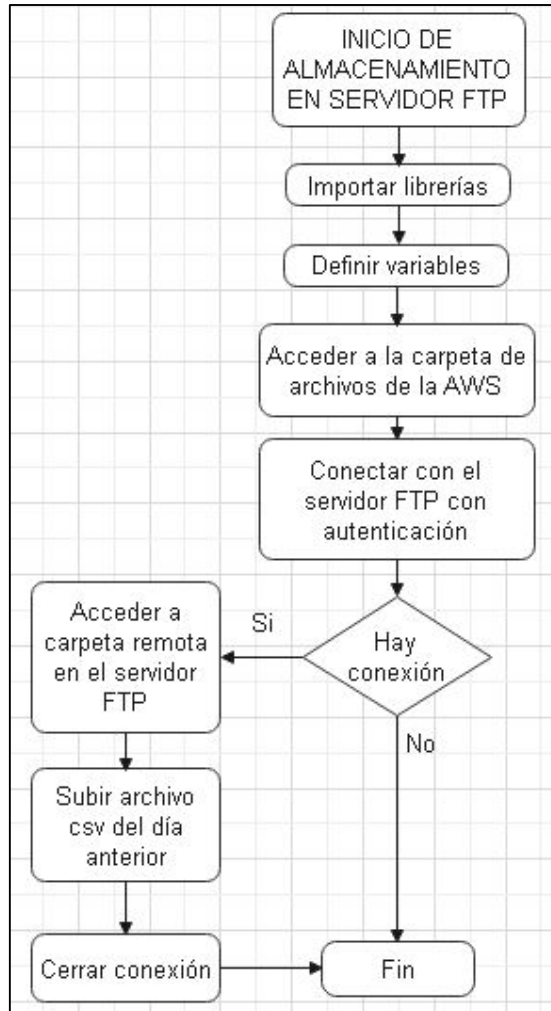
La conexión y el almacenamiento de los datos de estas variables en la base de datos se realiza de acuerdo al conjunto de funciones del Código 2.44, la diferencia es que se guarda en una nueva tabla de valores denominada “LecturaSensor” mediante la función *INTO()* y el comando *cursor.execute()*, presentado en el Código 2.57.

```
cursor = conexion.cursor()
datos = "INSERT INTO LecturaSensor(Fecha, Hora, Sensor, Temperatura, Humedad, Presion)
        VALUES('%s', '%s', '%s', '%s', '%s', '%s')"%(fecha, hora, idSensor, temp, hum, pres)
cursor.execute(datos)
```

**Código 2. 57.** Almacenamiento de datos en DB.

### 2.5.5. ALMACENAMIENTO EN SERVIDOR FTP

Los archivos diarios generados por la AWS y que contienen la información de las variables meteorológicas que servirán de entrada para las redes neuronales, serán almacenadas en el servidor FTP al final de cada día luego del proceso de interpolación, por tanto, el desarrollo del script se basa en el diagrama de flujo de la Figura 2.45.



**Figura 2. 45.** Diagrama de flujo del almacenamiento en el servidor FTP.

Para este proceso se implementan tres scripts los cuales permiten el proceso de almacenamiento de archivos en las respectivas carpetas en el servidor FTP, el primer script se basa en la ejecución del intérprete de órdenes escritas en modo de texto *bash*, definiendo los archivos de extensión *sh* que contienen las órdenes y los archivos *csv* con los datos como se muestra en el Código 2.58.

```

nombreFile ="bash /home/pi/PROYECTO/CODIGOS/Archivo_diario.sh Sensor%s_Lectura%s.csv >
/home/pi/PROYECTO/CODIGOS/NOTIFICACIONES/Salidadiaria.txt" %(idSensor,yesterday)
nombreprocess ="bash /home/pi/PROYECTO/CODIGOS/Archivo_procesado.sh Sensor%s_Lectura%s.csv >
/home/pi/PROYECTO/CODIGOS/NOTIFICACIONES/SalidaProcess.txt" %(idSensor, yesterday)
  
```

**Código 2. 58.** Definición de órdenes para el almacenamiento en el servidor FTP.



Este proceso se ejecuta mediante la función `subprocess.call()`, como se muestra en el Código 2.59.

```
subprocess.call(nombreFile,shell=True)
subprocess.call(nombreprocess, shell=True)
```

**Código 2. 59.** Ejecución de órdenes para el almacenamiento en el servidor FTP.

El segundo y tercer script con extensión sh contienen instrucciones en lenguaje `bash`, los cuales se estructuran de acuerdo a las siguientes instrucciones: el Código 2.60 representa la función `echo` con el que se cumplen diferentes operaciones como; \$0 ejecución del script actual (Archivo\_diario.sh) y \$1 elección del primer argumento (Sensor%\_Lectura%.csv) definidos en el Código 2.58.

```
echo $0
echo $1
```

**Código 2. 60.** Ejecución de script y elección de archivo.

El acceso a la carpeta en donde se encuentra alojado el archivo a subir se lo realiza mediante el comando `cd` y una ruta absoluta, a través del Código 2.61.

```
cd /home/pi/PROYECTO/LECTURAS/DIARIAS
ls
aux="/home/pi/PROYECTO/LECTURAS/DIARIAS"
```

**Código 2. 61.** Acceso a la ruta del archivo.

Se define las credenciales de acceso al servidor FTP y se define el archivo del primer argumento dentro de la variable "FILE", a través del Código 2.62.

```
HOST='IP del servidor'
USER="Usuario del servidor"
PASSWD="Contraseña del servidor"
FILE=$1
```

**Código 2. 62.** Credenciales de acceso al servidor FTP.

Posteriormente, se define la ruta absoluta remota del servidor FTP en el cual se guardarán los archivos diarios o procesados (interpolados) según sea el caso, mediante el Código 2.63.

```
REMOTEPATH='public_html/Lecturas_Diarias/NODO_3_CUMBAYA/SIN_PROCESAR'
```

**Código 2. 63.** Ruta en el servidor FTP.

La conexión al servidor FTP remoto se realiza con el comando *ftp* y el *host* definido, en el caso de ocurrir un error esta conexión terminará con la instrucción *END\_SCRIPT*, además, a través de la instrucción *ftp* interna *quote* se envían los comandos al servidor para una correcta conexión, como se muestra en el código 2.64.

```
ftp -n $HOST <<END_SCRIPT
quote USER $USER
quote PASS $PASSWD
```

**Código 2. 64.** Conexión al servidor FTP.

Para almacenar el archivo en la dirección definida en el servidor se; ingresa a dicho directorio con el comando *cd*, se guarda el archivo con el comando *put* y se procede a terminar la conexión con el servidor con la instrucción *quit*, como se muestra en el código 2.65.

```
cd $REMOTE_PATH
put $FILE
quit
```

**Código 2. 65.** Almacenamiento de archivo en el servidor FTP.

Finalmente, se da por terminado el script y se ejecuta la salida con el comando *exit 0*, ver le Código 2.66.

```
END_SCRIPT
exit 0
```

**Código 2. 66.** Finalización de script.

## **2.5.6. APLICACIÓN DESARROLLADA PARA EL REGISTRO Y MANIPULACIÓN DE DATOS**

La aplicación presenta dos ventanas: la primera muestra la interfaz de bienvenida y contiene dos botones, los cuales permiten salir del sistema o ingresar a la segunda ventana como se muestra en la Figura 2.46, para esto, se desarrolla un diagrama de flujo para la implementación del script, el cual se muestra en la Figura 2.47.



Figura 2. 46. Ventana de presentación.

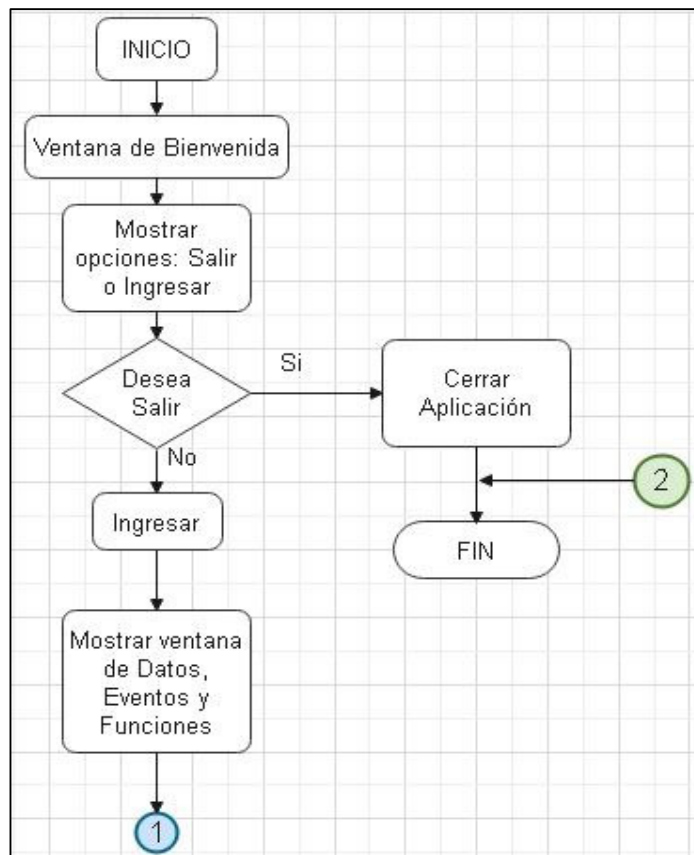


Figura 2. 47. Diagrama de flujo ventana 1.

La ventana de presentación de la aplicación de escritorio desarrollada en la AWS se crea a través de la función *tk()* en la que se definen varias características como: título con la función *title()*, las dimensiones con *geometry()*, el color de fondo o *background (bg)* con *configure()* y por último con la función *resizable()* que no permite cambiar las dimensiones de la pantalla, esto se puede observar en el Código 2.67.

```
principal = Tk()
principal.title("NODO 3 - CUMBAYÁ")
principal.geometry("700x500")
principal.configure(bg='black')
principal.resizable(False, False)
```

**Código 2. 67.** Creación de la ventana de presentación.

Para insertar una imagen se utiliza, el Código 2.68 en la que se define la ruta en la que se encuentra; a través de la función *PhotoImage()* que permite insertar dicha imagen en una sección de pantalla *Label()* o botón y se acomoda mediante la función *pack()*.

```
caratula_dir = "/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/caratula.gif"
caratula = PhotoImage(file = caratula_dir)
fondo = Label(principal, image=caratula)
fondo.pack()
```

**Código 2. 68.** Imagen de la ventana principal.

La creación de botones con imágenes incluidas en estos se realiza de acuerdo al Código 2.69, el cual se diferencia en la función *subsample()* que permite reducir a una imagen y que se ajuste al botón creado, además, se puede asociar una función con la instrucción *command* y colocarlo en un determinado lugar con la instrucción *place()*.

```
ing_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/entrar.PNG"
ing = PhotoImage(file = ing_dir)
ing_red = ing.subsample(3)
entrar=Button(text="INGRESAR",image=ing_red, command=Pantalla_in)
entrar.place(x=450, y=420)
```

**Código 2. 69.** Creación de botones.

La visualización de la ventana se realiza a través del Código 2.70, mediante la función *mainloop()*.

```
principal.mainloop()
```

**Código 2. 70.** Visualización de ventanas.

La segunda interfaz presenta tres secciones para el control y monitoreo de la AWS como se muestra en la Figura 2.48 y para su desarrollo se presenta el diagrama de flujo de la Figura 2.49; estas secciones se describen a continuación.



**Figura 2. 48.** Interfaz de control y monitoreo

- **Datos y consultas**

En esta sección se realiza la consulta y visualización de datos tanto de la base de datos del Hosting Web como datos de los archivos csv o excel de las AWS para su respectiva revisión, exportación de archivos csv a excel; entre otras funciones.

- **Eventos**

En esta sección de la aplicación se presentan todos los eventos ocurridos en la AWS a lo largo del periodo de tiempo de funcionamiento.

- **Programación de reinicio**

En esta sección el usuario puede reiniciar la AWS en ese instante o programar el reinicio en un rango de 1 a 5 días.

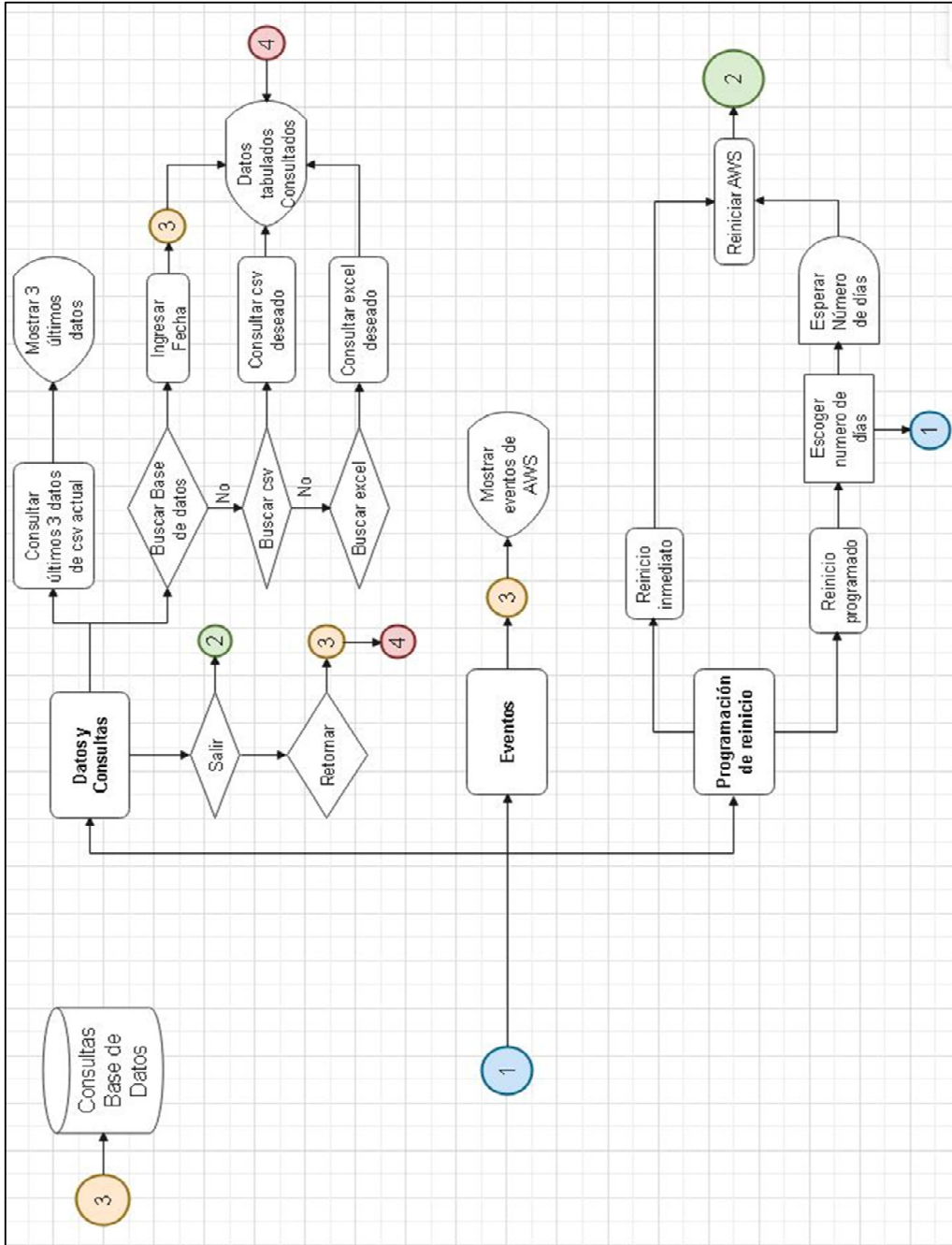


Figura 2. 49. Diagrama de flujo de la ventana 2 de la aplicación.

El desarrollo de la segunda ventana es similar a la primera, sin embargo, se conforma de tres secciones en la misma pantalla definida en este caso como “ventana”, las cuales se crean con el widget `LabelFrame()` como se muestra en el Código 2.71, que presenta funciones para nombrar la sección con `text` con el tipo y tamaño de letra con `font`. Las

opciones dentro de la instrucción `pack()` permiten rellenar los espacios por el widget y tener una aspecto centrado y atractivo en la aplicación.

```
#DIVISIONES DE LA PANTALLA
div1=LabelFrame(ventana, text="DATOS NODO CUMBAYÁ",font=("Arial",10))
div1.pack(fill="both", expand="yes")
```

**Código 2. 71.** Creación de secciones.

- **Datos y consultas**

Esta sección se ha subdividido en partes, las cuales permiten diferenciar las operaciones o funciones que la aplicación cumple, por ejemplo: la subsección BASE DE DATOS se muestra en la Figura 2.50, se crea mediante el Código 2.72, a la cual se nombra con la función `Label()` y se la ubica con la función `grid()` en una determinada fila y columna dentro del `LabelFrame` creado.



**Figura 2. 50.** Subsección BASE DE DATOS.

```
#BASE DE DATOS
etiqueta= Label(div1_1, text="BASE DE DATOS", fg="red", font=("Arial", 12))
etiqueta.grid(row=0, column=1, padx=2, pady=2)
```

**Código 2. 72.** Ubicación de etiqueta con grid.

En esta subsección se presenta un widget denominado `Entry()`, función que permite ingresar una fecha como `string` con la función `StringVar()`, este proceso se define en el Código 2.73, que al asociar a un botón denominado BUSCAR y este a su vez a un comando, como en el Código 2.69, que al presionarlo realiza la función de búsqueda de la información en la base de datos.

```
fecha1=StringVar()
fecha = Entry(div1_1, textvariable=fecha1)
fecha.grid(row=1, column=1, padx=2, pady=2)
```

**Código 2. 73.** Widget para ingresar valores o texto.

En la misma pantalla el resto de los botones se implementan de forma similar al Código 2.69 como se muestra en la Figura 2.51.



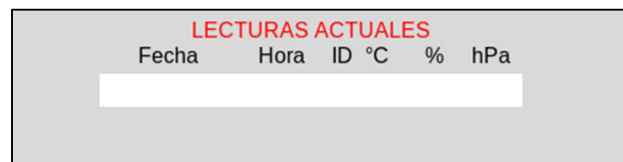
**Figura 2. 51.** Botones de la sección 1.

Los botones de la Figura 2.51 cumplen funciones diferentes y las cuales se describen en la Tabla 2.3.

**Tabla 2. 3.** Funciones de los botones.

Botón	Función
<b>Buscar (base de datos)</b>	Buscar información según la fecha ingresada en Entry() y mostrar en la ventana Treeview.
<b>Buscar(importar/exportar)</b>	Buscar archivos csv o Excel en la AWS y mostrarlos en la ventana Treeview.
<b>Exportar</b>	Convertir la información de la ventana Treeview a un archivo tipo Excel.
<b>Retornar</b>	Mostrar información de la base de datos luego de un proceso.
<b>Salir</b>	Salir de la aplicación.

Por otro lado, existen dos formas de presentar la información de la base de datos y de los archivos internos, una de estas es a través de la subsección LECTURAS ACTUALES que se muestra en la Figura 2.52, que permite leer los 3 últimos datos del archivo actual generado por la AWS y se actualizan cada minuto, se implementan a través del widget *Label()* con el Código 2.74.



**Figura 2. 52.** Subsección LECTURAS ACTUALES.

```
lec1=Label(div1_1)
lec1.grid(row=8, column=1)
```

**Código 2. 74.** Definición de label para mostrar lecturas.



El *label* está asociado a una función de lectura y visualización de datos con la función *config()*, donde la información se almacena en un array *rows2[ ]* y se muestra en el *label* "lec1", como se muestra en el Código 2.75.

```
lec1.config(text=rows2[0], bg="white",  
            fg="midnight blue", font=("Arial", 14))
```

**Código 2. 75.** Almacenamiento y visualización de datos.

Finalmente, a través de la función *after()* se actualizan los valores de las lecturas cada 1min, mediante el Código 2.76.

```
lec1.after(60000,lecturas)
```

**Código 2. 76.** Actualización de datos.

Otra forma de presentar la información de los parámetros meteorológicos tanto de la base de datos como de los archivos internos (csv o Excel) de la AWS consultados, se realiza mediante una forma tabulada en el widget *Treeview* de la Figura 2.53, y se genera de acuerdo al Código 2.77 mediante la función *ttk.Treeview()* insertada en la sección 1, posee 6 columnas, muestra las cabeceras de estas columnas y tiene un espacio para mostrar información en 4 filas.

Fecha	Hora	Nodo	Temperatura(°C)	Humedad(%)	Presion(hPa)

**Figura 2. 53.** Tabla de visualización de datos.

```
data=ttk.Treeview(div1, columns=(1,2,3,4,5,6),  
                  show="headings", height="4")  
data.pack()
```

**Código 2. 77.** Generación de tabla de datos.

Se generan las columnas de acuerdo a la función *column()*, la cual presenta dos instrucciones que permiten definir el ancho de la columna *width* y la anchura mínima de un elemento *minwidth*, como se presenta en el Código 2.78.

```

data.column("1", width=100, minwidth=100)
data.column("2", width=100, minwidth=100)
data.column("3", width=100, minwidth=90)
data.column("4", width=140, minwidth=140)
data.column("5", width=130, minwidth=130)
data.column("6", width=130, minwidth=130)

```

**Código 2. 78.** Generación de columnas en Treeview.

Los nombres de las columnas se definen mediante la función *heading()* de acuerdo al número de columna descrita anteriormente, como se muestra en el Código 2.79.

```

data.heading(1, text="Fecha")
data.heading(2, text="Hora")
data.heading(3, text="Nodo")
data.heading(4, text="Temperatura(°C)")
data.heading(5, text="Humedad(%)")
data.heading(6, text="Presion(hPa)")

```

**Código 2. 79.** Definición de nombres a columnas de Treeview.

- **EVENTOS**

En esta sección se describe el proceso de creación de una ventana Treeview, que se presentó anteriormente, para presentar la información de los eventos ocurridos en la AWS, mediante la conexión a la base de datos con el Código 2.80.

```

conexion = mariadb.connect(host=" ",
                           user=" ",
                           password=" ",
                           database=" ")
cursor = conexion.cursor()

```

**Código 2. 80.** Conexión a la base de datos.

Se realiza la consulta y se eligen los eventos mediante la función *SELECT*, desde la tabla "Eventos" con la función *from*, la información se presenta de manera descendente, es decir, desde el último evento que ocurrió. Para ello, se utiliza la función *ORDER BY* y *DESC*, de acuerdo a las columnas que se desean ordenar, además se ejecuta el anterior proceso mediante la función *execute()*, para elegir todas las filas de la base de datos se ejecuta el comando *fetchall()* que se almacena en la variable *rows1* y la información se envía a una función llamada *datosevent(rows1)*, como se muestra el Código 2.81.

```

query = "SELECT Fecha, Hora, Sensor, Evento
        "from Eventos ORDER BY Fecha DESC, Hora DESC"
cursor.execute(query)
rows1 = cursor.fetchall()
datosevent(rows1)

```

**Código 2. 81.** Consulta y envío de información de los eventos en la AWS.

La función que permite visualizar la información de eventos ocurridos en la tabla se realiza mediante la función `datosevent(rows1)`, a través de la función `eventos.delete(*eventos.get_children())`, donde "eventos" es el *Treeview* creado, se eliminan todos los elementos antes guardados en la tabla y a través de un lazo *for* y mediante la función `eventos.insert()`, se pueden actualizar y visualizar los datos en la tabla *Treeview*, como se muestra en el Código 2.82.

```

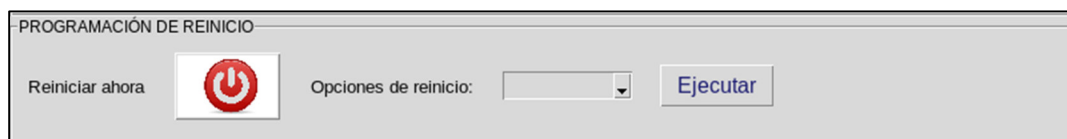
#Datos eventos
def datosevent(rows1):
    global event
    event = rows1
    eventos.delete(*eventos.get_children())
    for i in rows1:
        eventos.insert('', 'end', values=i)

```

**Código 2. 82.** Visualización de datos

- **PROGRAMACIÓN DE REINICIO**

En la tercera y última sección de la Figura 2.54 se implementa un botón de reinicio inmediato de acuerdo a los códigos descritos, que está asociado a una función, la cual se muestra en el Código 2.83.



**Figura 2. 54.** Sección de reinicio

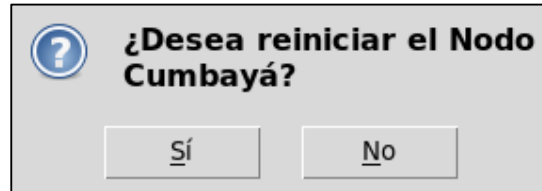
```

#Reinicio inmediato
def reinicio():
    reboot=ms.askyesno("Mensaje", "¿Desea reiniciar el Nodo Legarda?")
    if reboot==True:
        subprocess.call("shutdown -r now", shell=True)
    else:
        ms.showinfo("Mensaje", "Reinicio cancelado")

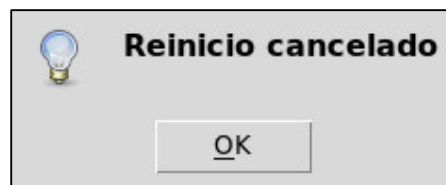
```

**Código 2. 83.** Función de reinicio.

La función del Código 2.83 incorpora un *askyesno()* que permite afirmar (Si) o negar (No) a una operación que la AWS quiera realizar como se muestra en la Figura 2.55 y un *showinfo()* que informa sobre dicho proceso, estas cajas de mensajes pertenecen al widget *Messagebox*. Por ende, si la aceptación es verdadera a través de *subprocess.call("shutdown -r now", Shell=True)* la estación se reinicia de forma inmediata, y en el caso de negarse *showinfo()* informará la cancelación de dicha operación, como se muestra en la Figura 2.52.



**Figura 2. 55.** Askyesno para reinicio de AWS.



**Figura 2. 56.** Showinfo por cancelación de reinicio.

Esta sección, además, cuenta con opciones de programación de reinicio de la AWS, lo cual se implementa con el widget Combobox como se muestra en el Código 2.84, que permite tener varias opciones para que el usuario elija el más conveniente, las opciones se enlistan en un array, el combobox se genera con la función *ttk.Combobox()*, en el cual se ingresa la información de la ubicación *div3*, las opciones a tomar o *values*, que el usuario puede elegir y que se definen con *state="readonly"*, y *textvariable* que permite guardar la información del número de días en que la estación se reiniciará en la variable "día".

```
opciones=["Sin reinicio", "1 día", "2 días",
          "3 días", "4 días", "5 días"]
listopciones=ttk.Combobox(div3, width=10, values=opciones,
                          state="readonly", textvariable=dia)
listopciones.grid(row=0, column=4, padx=10, pady=10)
```

**Código 2. 84.** Programación de reinicio.

Las funciones a las cuales se encuentran asociadas cada uno de los widgets definidos en la aplicación se encuentran en el Anexo I: Funcionamiento del código de aplicación, en el

cual se presentan todos los códigos utilizados para el funcionamiento y monitoreo de la AWS (*Automatic Weather Station*, Estación Meteorológica Automática).

### 2.5.7. REDES NEURONALES

En este apartado se presentan tres configuraciones para la predicción de datos meteorológicos como: el modelo ARIMA y las redes neuronales LSTM simple y LSTM apilado de dos capas, con las cuales se quiere probar el modelo de mejor eficiencia para las predicciones de temperatura y humedad de las distintas zonas de estudio.

#### 2.5.7.1. Modelo ARIMA

En primera instancia mediante el Código 2.85 se importan las librerías necesarias para analizar, procesar y graficar datos como: *numpy*, *pandas* y *matplotlib*, además, se importa la librería *statsmodels* que proporciona la capacidad de adaptarse a un modelo ARIMA.

```
import warnings
warnings.filterwarnings('ignore')
from pandas import read_csv
from statsmodels.tsa.arima.model import ARIMA
import numpy
import matplotlib.pyplot as plt
from matplotlib import pyplot
```

**Código 2. 85.** Importación de librerías del modelo ARIMA

El Código 2.86 permite cargar los datos en un *DataFrame* en la variable *series* del archivo *Temperatura\_Cumbaya.csv*, datos adquiridos durante 75 días y remuestreados por hora para el entrenamiento de la red neuronal, que mediante la función *series.values* devuelve los valores de temperatura en un array y se guarda en una variable X.

Finalmente se carga los datos en un *DataFrame* en la variable *real* del archivo *Temperatura\_prueba\_Cumbaya.csv* para comparar con las predicciones realizadas por la red neuronal. Las funciones: *infer\_datetime\_format* permiten inferir el formato de fecha y hora para un análisis más rápido, *parse\_dates* permite elegir la columna con un formato de fecha adecuado para su análisis.

Por otro lado, se define el valor de datos por día en *days\_in\_year* parámetro necesario para el proceso de diferenciación de la función *difference(args.)*.

```

# Lectura de datos de entrenamiento
series = read_csv('Temperatura_Cumbaya.csv', header=0, index_col=0)
# Diferenciación estacional
X = series.values
days_in_year = 24
differenced = difference(X, days_in_year)
# Lectura de datos de prueba
real = read_csv('Temperatura_prueba_Cumbaya.csv', header=0,
                infer_datetime_format=True,
                parse_dates=['datetime'],
                index_col=['datetime'])

```

**Código 2. 86.** Lectura de datos para el entrenamiento.

La función *difference(args.)* del Código 2.87 realiza el proceso de diferenciación de los datos guardados en la variable X y definidos en esta función como *dataset* para que los datos de entrenamiento sean estacionales y el modelo ARIMA implementado sea eficiente.

La función realiza la diferenciación de un valor actual con otro valor hace 24 horas, por Ejemplo: *dataset[24]-dataset[0]*, mediante la función *append* se agrega cada valor obtenido en *value* a la lista *diff* definida, lista que se retorna como un array o matriz numpy.

```

# Diferenciación de series
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return numpy.array(diff)

```

**Código 2. 87.** Función de diferenciación de las series.

De acuerdo a la interpretación del *script* de la red neuronal ARIMA, el Código 2.88 define los parámetros para llamar a la función de *prediccion(args.)* de datos, parámetros tales como: X array de los datos leídos, *days\_in\_year* número de datos por día, *differenced* matriz retornada del proceso de diferenciación y los parámetros *p*, *d*, *q* respectivamente que especifican el modelo ARIMA configurado.

```

# Predicción de valores
predict=prediccion(X, days_in_year, differenced,1,1,1)

```

**Código 2. 88.** Predicciones y MEAN.

La función de predicción *prediccion(args.)* del Código 2.89 define el modelo ARIMA con los datos de diferenciación estacional como entrada y los parámetros p, d, q mediante la función *ARIMA(args.)*, modelo que se entrena mediante la función *model.fit()*, además,

mediante la función `model_fit.forecast(args.)` se realizan las predicciones, siendo `steps` los pasos de tiempo a pronosticar, en este caso es 24, es decir se obtienen las predicciones de las próximas 24 horas, sin embargo, son pronósticos de valor invertido.

Los pronósticos de valor invertido se transforman en valores utilizables mediante la función `inverse_difference(args.)` del Código 2.90, donde `history` es el seguimiento manual de los datos de entrenamiento y predicción que se insertan en cada iteración, `yhat` toma cada valor de pronóstico invertido y se suma al dato del historial de hace 24 horas definido en `days_in_year`. La variable `predict` alberga los datos pronosticados que van a ser comparados con los datos de prueba del día 76.

```
def prediccion(X, days_in_year, differenced,P1,P2,P3):
    # Definición del modelo ARIMA
    model = ARIMA(differenced, order=(P1,P2,P3))
    # Entrenamiento de La red
    model_fit = model.fit()
    # Predicciones de varios pasos
    forecast = model_fit.forecast(steps=24)
    # Seguimiento de observaciones
    history = [x for x in X]
    day = 1
    predict = []
    for yhat in forecast:
        # invertir el pronóstico diferenciado en algo utilizable
        inverted = inverse_difference(history, yhat, days_in_year)
        predict.append(inverted)
        history.append(inverted)
        day += 1
    return predict
```

**Código 2. 89.** Función de predicción de datos con el modelo ARIMA.

```
# Invertir el valor de diferenciación
def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]
```

**Código 2. 90.** Función de diferenciación inversa.

Por lo general, para medir el desempeño o la precisión de los pronósticos realizados se usa MAPE<sup>17</sup> (*Mean Absolute Porcentaje Error*, Error Porcentual Absoluto Medio), por tanto, mediante el Código 2.91 se definen las series temporales a comparar entre los datos reales y pronosticados.

---

<sup>17</sup> **MAPE.** - Es una medida de precisión del pronóstico de un modelo que mide el tamaño del error en términos porcentuales, es decir es un indicador de desempeño o eficiencia de los pronósticos realizados.

```
# Indicador de desempeño
output=mean_absolute_percentage_error(real, predict)
```

**Código 2. 91.** Indicador de desempeño MAPE.

Mediante el Código 2.92 se define la función MAPE que retorna la media del error porcentual que puede variar entre 0% y 100% existente entre los datos reales y pronosticados.

```
# MAPE-Indicador de desempeño
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = numpy.array(y_true), numpy.array(y_pred)
    result=numpy.mean(numpy.abs((y_true - y_pred) / y_true)) * 100
    return result
```

**Código 2. 92.** Función para el cálculo de MAPE.

Finalmente, para tener una idea de la variación de los pronósticos realizados y los valores reales del día 76 obtenidos en la AWS se procede a graficar estas series temporales de datos mediante el Código 2.93, *matplotlib* que ofrece una serie de herramientas como la función *plt.plot(args.)* que grafica datos tanto en el eje X que contiene las horas como en el eje Y que contine los datos reales y pronosticados.

```
e=list(range(len(real)))
plt.figure(figsize=(8,4))
plt.title('PRONOSTICO DE TEMPERATURA AWS 3 - CUMBAYA')
plt.xlabel('Horas')
plt.ylabel("'°C")

plt.plot(e,real,label='Temperatura Real')
plt.plot(e,predict,label='Prediccion con ARIMA')
plt.legend(['Real','Prediccion con ARIMA'], loc='center left')
```

**Código 2. 93.** Proceso para graficar datos.

### 2.5.7.2. LSTM Simple

En este caso inicialmente se importan las librerías necesarias para la construcción del modelo LSTM simple (una capa) del Código 2.94, como se puede observar la librería Keras tiene un mayor peso debido a que se pueden desarrollar complejos modelos de aprendizaje profundo, es decir, es una excelente biblioteca para construir por bloques diferentes arquitecturas de redes neuronales incluidas convolucionales y recurrentes, así como pandas, numpy, scipy y sklearn.



```

# Librerías
import warnings
warnings.filterwarnings('ignore')
import numpy
import math
from math import sqrt
from numpy import split
from numpy import array
from pandas import read_csv
from sklearn.metrics import mean_squared_error, mean_absolute_error
from scipy.special import logsumexp
from matplotlib import pyplot
from keras.models import Sequential
from keras.layers import RepeatVector, Dense, Flatten, LSTM, TimeDistributed, Dropout, Input, GRU
from keras.layers.convolutional import Conv1D, MaxPooling1D
from keras.regularizers import l2
from keras.models import Model
import matplotlib.pyplot as plt
from matplotlib import pyplot
from keras.utils.vis_utils import plot_model
from keras.optimizers import SGD
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import LSTM

```

**Código 2. 94.** Librerías para LSTM.

En el Código 2.95 se define el número de datos u horas de entrada y salida del modelo LSTM en este caso es 24 y el número de días de datos utilizados para el entrenamiento de la red.

```

# Cantidad de horas entrada y salida
n_input = 24
n_out= 24
days=75

```

**Código 2. 95.** Parámetros especificados para LSTM.

El Código 2.96 permite cargar los datos en un *DataFrame* en la variable *dataframe* del archivo *Temperatura\_Cumbaya.csv*, datos adquiridos durante 75 días y remuestreados por hora para la red neuronal, la función *index\_col* permite utilizar una columna en particular como índice, a través de la función *dataframe.values* que devuelve los valores de temperatura en un array, los cuales son convertidos en valores de punto flotante que son adecuados para modelar una red neuronal y se guarda en la variable *dataset* .

Los datos se dividen en tres grupos que son: entrenamiento (*train*) con el 70% de los datos, validación (*val*) con el 16% de datos y prueba (*test*) con el 14% de datos, los cuales permitirán ajustar el modelo de red neuronal con pesos y umbrales adecuados para generar predicciones con errores mínimos.

```

# Lectura de datos
dataframe = read_csv('Temperatura_Calderon.csv',
                    header=0, infer_datetime_format=True,
                    parse_dates=['datetime'], index_col=['datetime'])
dataset=dataframe.values
dataset = dataset.astype('float32')
# División de datos
train_limit=math.ceil((len(dataset)/n_out)*0.7)
middle_limit=math.ceil((len(dataset)/n_out-train_limit)/2)
train = dataset[1:(train_limit*n_out)+1]
val = dataset[(train_limit*n_out):((train_limit+middle_limit+1)*n_out)]
test = dataset[((train_limit+middle_limit)*n_out):len(dataset)]

```

**Código 2. 96.** Lectura y división de datos.

Una red neuronal LSTM recibe una estructura específica de matriz 3D de la forma [muestras, paso de tiempo, características], por tanto, el Código 2.97 permite reestructurar esta matriz que inicialmente se presenta como una matriz 2D de la forma [muestras, características], los datos de los tres grupos antes definidos se agrupan en ventanas de 24 datos cada una, que a su vez se dividen en subconjuntos de datos de entrada (*train\_x*, *val\_x*) y salida (*train\_y*, *val\_y*).

La conversión de las entradas X de una matriz 2D a una matriz 3D se realiza mediante la función *to\_supervise(args.)* del Código 2.98 y por último se define la matriz 3D de entrada a la red neuronal mediante la función *shape[]*.

```

# Reestructuración en ventanas de datos
train = array(split(train, len(train)/n_out))
val = array(split(val, len(val)/n_out))
test = array(split(test, len(test)/n_out))
# Etiquetado de datos
train_x, train_y = to_supervised(train, n_input, n_out)
val_x, val_y = to_supervised(val, n_input, n_out)
# Definición de parámetros matriz LSTM
n_timesteps = train_x.shape[1] # Paso de tiempo
n_features = train_x.shape[2] # Características
n_outputs = train_y.shape[1] # Salidas

```

**Código 2. 97.** Reestructuración para matriz LSTM.

Como se mencionó la conversión 2D a 3D se realiza mediante la función *to\_supervise(args.)* del Código 2.98, dentro de la cual se transforma a la estructura esperada mediante la función *reshape()*, la red neuronal repasará el historial de datos paso por paso, por ende, mediante el bucle *for* se define el final de la secuencia de entrada que va a recorrer la red y a través de un nuevo bucle se guardarán los datos hasta las límites

definidos en las matrices numpy en las variables “X” y “y” que la red neuronal ha recorrido de acuerdo a la reestructuración dada anteriormente.

```
# Conversión del historial en entradas y salidas
def to_supervised(train, n_input, n_out):
    # Reestructuración de datos
    data = train.reshape((train.shape[0]*train.shape[1],
                          train.shape[2]))

    print(data.shape)
    X, y = list(), list()
    in_start = 0
    # Repaso de la red paso a paso por el historial
    for _ in range(len(data)):
        # Límite de la entrada de la secuencia
        in_end = in_start + n_input
        out_end = in_end + n_out
        # aseguramiento de datos
        if out_end <= len(data):
            x_input = data[in_start:in_end, 0]
            x_input = x_input.reshape((len(x_input), 1))
            X.append(x_input)
            y.append(data[in_end:out_end, 0])
        # mover un paso de tiempo
        in_start += 1
    return array(X), array(y)
```

**Código 2. 98.** Transformación de datos.

De acuerdo a la interpretación del código se realiza una reestructuración de los datos de entrenamiento y validación de salida, el mismo proceso que a los datos originales para convertirlos en una matriz 3D (muestra, paso de tiempo, características) como se muestra en el Código 2.99.

```
## Reestructurar salidas
#[muestras, pasos de tiempo, características]
train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
val_y = val_y.reshape((val_y.shape[0], val_y.shape[1], 1))
```

**Código 2. 99.** Reestructuración de valores de salida.

Se define el modelo de la red neuronal a utilizar, en este caso para el pronóstico de los datos de temperatura y humedad, la configuración es una red LSTM simple de una capa oculta, el Código 2.100, que se configura mediante la función *model.add(LSTM(args.))*.

La red neuronal consta de una capa de entrada, una capa oculta con 300 nodos, la forma de entrada a esta capa se define mediante la función *input\_shape()* que refleja 24 pasos de tiempo con una característica y se configura a las neuronas con una función de

activación Relu. La capa de salida se configura mediante la función `model.add(Dense())` que posee 150 nodos conformados con una función de activación ReLU, las cuales realizan predicciones de 24 salidas definidas por la función `model.add(Dense(n_outputs))` y a través de la función `model.compile()` se configura el modelo para la utilización de 2 parámetros en el entrenamiento como: la función de pérdida (RMSE) y el optimizador (Adam) para que el proceso de aprendizaje sea más rápido.

```
# Definición del modelo
#=====
model = Sequential()
model.add(LSTM(300, activation='relu', input_shape=(n_timesteps, n_features)))
model.add(Dense(150, activation='relu'))
model.add(Dense(n_outputs))
model.compile(loss='mse', optimizer='adam')
print(model.summary())
```

**Código 2. 100.** Definición del modelo LSTM simple.

El Código 2.101 prepara a la red neuronal LSTM simple con los datos de entrada para su respectivo entrenamiento mediante la función `model.fit(args.)`, los parámetros definidos para el entrenamiento son: `verbose=0` que no mostrará el progreso del entrenamiento para cada época, `epochs=20` que ajusta el modelo para 20 épocas de entrenamiento, es decir, todos los datos completan 20 ciclos y `batch_size=16` es el tamaño del lote, que representa la cantidad de muestras a utilizar en cada actualización de peso, a este proceso se le conoce como descenso de gradiente que permite adecuar los pesos y umbrales para predicciones más acertadas.

Finalmente, se realiza un seguimiento manual de todas las observaciones en una lista llamada predicción historial (*hystory*) que se inserta con los datos de entrenamiento tomando todas las columnas en grupos de 24 filas, es decir se tiene 53 grupos de 24 filas de datos.

```
verbose, epochs, batch_size = 0, 20, 16
# entrenamiento de la red
model.fit(train_x, train_y, epochs=epochs,
          batch_size=batch_size,
          validation_data=(val_x, val_y), verbose=verbose)

# Seguimiento de datos diarios
history = [x for x in train]
yhat_sequence = forecast(model, history, n_input)
```

**Código 2. 101.** Entrenamiento de la red LSTM simple

Para realizar el pronóstico de datos se crea una función denominada *forecast(args.)*, las variables que ingresan como parámetros de predicción son el modelo de la red, los datos de entrenamiento y el número de salidas que se requiere, como se muestra en el Código 2.102, predicciones que serán almacenadas en la variable *yhat\_sequence*.

A continuación se define una matriz, la cual engloba todos los datos del historial en una sola matriz que se guarda en la variable *data*, con los que se procede a reestructurar dichos datos en una sola matriz, posteriormente mediante la función *input\_x=data[-n\_input:, 0]* se extrae los últimos 24 datos de la columna 0 de la matriz reestructurada para ocuparlos como datos de entrada definidos en 2D, esta matriz es transformada a una matriz 3D con la función *input\_x=input\_x.reshape(1, len(input\_x), 1)*. Finalmente, mediante la función *yhat=model.predict(input\_x, verbose=0)*, se realizan las predicciones que se almacenan en la variable *yhat*, sin embargo, se debe tomar en cuenta que solo se desea el vector de predicción, por ende, *yhat=yhat[0]* generando el vector con los 24 datos pronosticados.

```
def forecast(model, history, n_input):
    »# almacenamiento de datos
    »data = array(history)
    »data = data.reshape((data.shape[0]*data.shape[1], data.shape[2]))
    »# almacenar ultimos datos como entradas
    »input_x = data[-n_input:, 0]
    »#Reestructurar a 3D
    »input_x = input_x.reshape((1, len(input_x), 1))
    »# Predicciones
    »yhat = model.predict(input_x, verbose=0)
    »# vector de prediccion
    »yhat = yhat[0]
    »return yhat
```

**Código 2. 102.** Predicciones de valores.

Los errores que se presentan al comparar las predicciones realizadas por la red neuronal y los valores reales se calculan de acuerdo a RMSE, para ello, se implementa una función para en efecto se disponga del reporte los errores y se pueda reconfigurar, si es necesario la red. El Código 2.103 presenta la función de cálculo de dicho parámetro de error, donde se realiza el cálculo de MSE mediante la función *mean\_square\_error(args.)* que se encarga de comparar cada valor pronosticado con el valor real y obtiene un error del cual se calcula la raíz cuadrada, para así obtener el RMSE de cada predicción, estos errores se guardan en la variable *scores*. El otro parámetro de evaluación es el RMSE general de la red neuronal, es decir, se comparan los 24 datos actuales y de predicción con los que se obtiene un promedio general del parámetro y se almacena en la variable *score*.

```

def evaluate_forecasts(actual, predicted):
    scores = list()
    # Calculo RMSE para cada datos
    for i in range(actual.shape[1]):
        # Calculo MSE
        mse = mean_squared_error(actual[:, i], predicted[:, i])
        # Calculo RMSE
        rmse = sqrt(mse)
        # Almacenamiento en scores
        scores.append(rmse)
    # Calcular RMSE general de la red
    s = 0
    for row in range(actual.shape[0]):
        for col in range(actual.shape[1]):
            s += (actual[row, col] - predicted[row, col])**2
    score = sqrt(s / (actual.shape[0] * actual.shape[1]))
    return score, scores

```

**Código 2. 103.** Cálculo del RMSE.

Las funciones antes definidas se citan de acuerdo a las líneas de funciones que se muestran en el Código 2.104, que permiten validar cada una de las predicciones realizadas por la red neuronal y almacenarlas en el historial para los próximos pronósticos.

```

# Validacion de cada dato
predictions = list()
for i in range(len(test)):
    # Validacion progresiva de cada dato
    yhat_sequence = forecast(model, history, n_input)
    # almacenamiento de predicciones
    predictions.append(yhat_sequence)
    # dato real de testeo y almacenamiento para la proxima prediccion
    history.append(test[i, :])
# evaluacion de las predicciones por cada dato
predictions = array(predictions)
score, scores = evaluate_forecasts(test[:, :, 0], predictions)

```

**Código 2. 104.** Validación de datos.

Finalmente, se presenta el gráfico de comparación entre los datos reales y las predicciones realizadas mediante el Código 2.105, el proceso es similar al descrito en el modelo ARIMA, en la cual se definen los datos, títulos y leyendas a mostrar con las funciones que brinda la librería matplotlib.

```

real = read_csv('Temperatura_prueba_Calderon.csv',
                header=0, infer_datetime_format=True,
                parse_dates=['datetime'], index_col=['datetime'])
e=list(range(len(real)))
plt.figure(figsize=(8,4))
plt.title('PRONOSTICO DE TEMPERATURA AWS 4 - CALDERON')
plt.xlabel('Hora')
plt.ylabel('°C')

plt.plot(e,real,label='Real Temperature')
plt.plot(e,yhat_sequence,label='Forecast with LSTM')
plt.legend(['Real','Predicción con LSTM_Simple'], loc='center left')

```

**Código 2. 105.** Proceso para graficar los datos obtenidos mediante LSTM.

### 2.5.7.3. LSTM Apilado

En este tipo de configuración solamente se cambia el modelo a utilizar, es decir, el resto de las funciones antes presentadas en la configuración de LSTM simple sirven para pronosticar los valores de temperatura y humedad relativa.

Por tanto, en el Código 2.106 se presenta la configuración del modelo LSTM apilado o multicapa, que cuenta con una capa de entrada, 2 capas ocultas de 150 nodos cada una configuradas con la función de activación ReLU y una capa de salida con 100 neuronas configuradas con una función de activación ReLU.

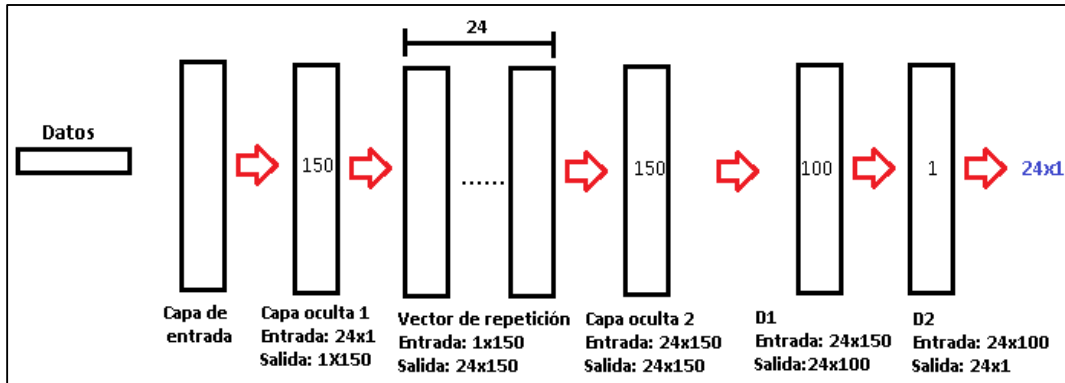
```

#Definición del modelo
#-----
model = Sequential()
model.add(LSTM(150, activation='relu', input_shape=(n_timesteps, n_features)))
model.add(RepeatVector(n_outputs))
model.add(LSTM(150, activation='relu', return_sequences=True))
model.add(TimeDistributed(Dense(100, activation='relu')))
model.add(TimeDistributed(Dense(1)))
model.compile(loss='mse', optimizer='adam')
print(model.summary())

```

**Código 2. 106.** Configuración del modelo LSTM multicapa.

Para explicar el funcionamiento del modelo se presenta la Figura 2.1, la cual describe el procesamiento de los datos y como estos entran y salen de las capas de conforman la red neuronal LSTM apilada para generar las predicciones de datos.



**Figura 2. 57.** Funcionamiento de la red neuronal LSTM multicapa.

Para el modelo stack LSTM o multicapa, como se mencionó anteriormente se diseña un modelo secuencial con una capa de entrada, 2 capas ocultas de 150 nodos cada una, la forma de entrada a la primera capa oculta es de 24 pasos de tiempo con una característica, es decir, con una matriz de entrada de forma (1x24), la primera capa lee los datos de entrada y genera 150 funciones generando una salida de 1x150, las cuales a través de la función `model.add(RepeatVector(noutputs))` genera una matriz de salida de 24x150, esta función permite repetir un vector de acuerdo al número de pasos generando una matriz acorde para la siguiente capa. La capa oculta 2, toma la matriz 24x150 y a su salida genera una matriz de 24x150 de acuerdo a los pesos y función de activación configurada, además en esta capa se habilita la función `return_sequences=True`, la cual hace que cada celda emita una señal por cada paso de tiempo, es decir, cada celda o nodo entrega una salida.

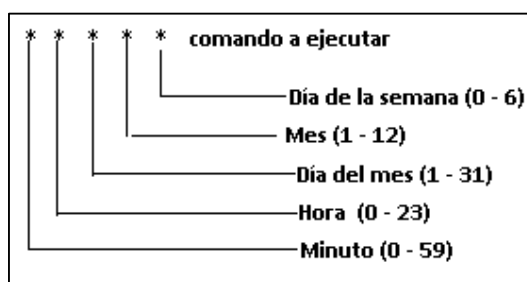
Por tanto, cada nodo de la capa 2 entrega un valor a los nodos de la capa de salida, la cual está conformada por 100 neuronas y una función de activación ReLU configurada con la función `model.add(TimeDistributed(Dense(100, activation='relu'))` generando una matriz de salida de 24x100. Finalmente, la última capa configurada con la función `model.add(TimeDistributed(Dense(1)))`, la cual crea un vector de longitud igual al número de entidades generadas desde la capa anterior, es decir 100, y se configura de acuerdo al número de características de los datos de entrada que en este caso es 1, generando una matriz de (100x1). A continuación, mediante la multiplicación de estas 2 últimas capas se obtiene como resultado final una matriz de 24x1 con los datos de predicción de las 24h.



## 2.5.8. AUTOMATIZACIÓN DE SCRIPTS

La automatización de los scripts es de suma importancia para que la estación sea automática y no necesite de la intervención humana para la adquisición de datos, envío de notificaciones por eventos que ocurra o procesamiento de datos cuando este lo requiera.

Para la automatización de los scripts se utiliza el demonio cron, que ejecuta tareas en segundo plano en fechas y horas definidas que se configuran en el archivo crontab, la forma de programar tareas periódicas que se desea ejecutar automáticamente debe cumplir con el modelo de la Figura 2.58, que muestra la estructura de implementación de una tarea específica a ejecutar.



**Figura 2. 58.** Estructura de programación en crontab.

En la Tabla 2.4 se muestra la información de las tareas que cron va a ejecutar en la AWS en tiempos definidos.

**Tabla 2. 4.** Establecimiento de horarios para las tareas.

Script	Horario
Notificación de reinicio	Reinicio de la AWS
Lectura de datos	Reinicio de la AWS
Chequeo de datos	Cada 5 min
Almacenamiento en la base de datos	Cada 10 min
Interpolación	Todos los días a las 00:05:00
Almacenamiento en el servidor FTP.	Todos los días a las 00:10:00

La implementación de la automatización de los scripts antes descritos se realiza como se muestra en la Figura 2.59 en el archivo crontab. El acceso a este archivo se realiza mediante el comando “*crontab -e*”, como se puede observar las tareas tienen el formato antes mencionado, las cuales se ejecutan mediante la instrucción “*python3*” acompañada de la ruta absoluta del script, sin embargo, se puede además identificar que la notificación de reinicio y la lectura de datos esperan 1 min y luego se ejecutan una sola vez al iniciar el dispositivo mediante la instrucción “*@reboot sleep 60 &&*”, este tiempo de espera tiene la finalidad de que la Raspberry Pi logre estabilizar su acceso a la red de internet.

```

# m h dom mon dow   command
@reboot sleep 60 && python3 /home/pi/PROYECTO/CODIGOS/Notificacion_reinicio.py
@reboot sleep 60 && python3 /home/pi/PROYECTO/CODIGOS/LecturasDatos.py
*/5 * * * * python3 /home/pi/PROYECTO/CODIGOS/CheckDatos.py
*/10 * * * * python3 /home/pi/PROYECTO/CODIGOS/Subir_DB.py
05 00 * * * * python3 /home/pi/PROYECTO/CODIGOS/Interpolacion.py
10 00 * * * * python3 /home/pi/PROYECTO/CODIGOS/Subir_archivos.py

```

**Figura 2. 59.** Automatización de scripts.

### 2.5.9. CONFIGURACIÓN DE REDES WIFI

La configuración de red Wifi con las credenciales de cada localidad antes de instalar es de suma importancia porque así la AWS se conectará de forma automática a la red y tendrá acceso a internet para informar sobre el inicio del sistema y si existe algún inconveniente en la inicialización para adquirir datos.

La red se puede configurar de dos formas diferentes: la primera es ejecutando en el terminal el Código 2.107 que permite acceder al archivo de configuración de la nueva red Wifi.

```

pi@raspberrypi:~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

```

**Código 2. 107.** Acceso al archivo de configuración de red Wifi.

La nueva red Wifi se configura de acuerdo a la Figura 2.60, dentro del archivo en donde se define el nombre de la red y la contraseña, una vez configurados se presiona Ctrl+o para guardar y Ctrl+x para salir y la AWS se podrá conectar a la red del sitio donde se instalará.

```

GNU nano 3.2 /etc/wpa_supplicant/wpa_supplicant.conf

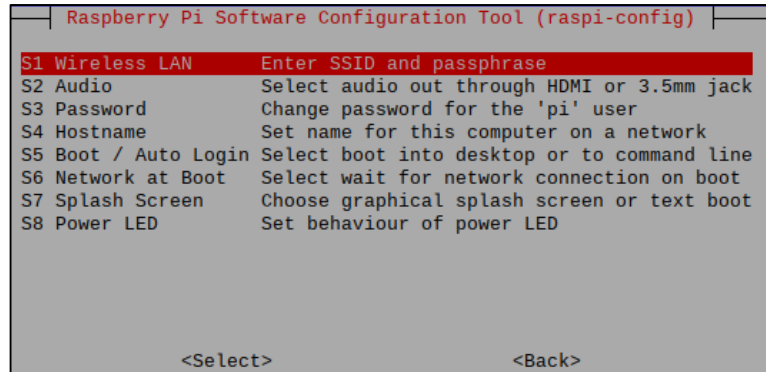
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=EC

network={
    ssid="Nombre de la red"
    psk=" Contraseña "
}

```

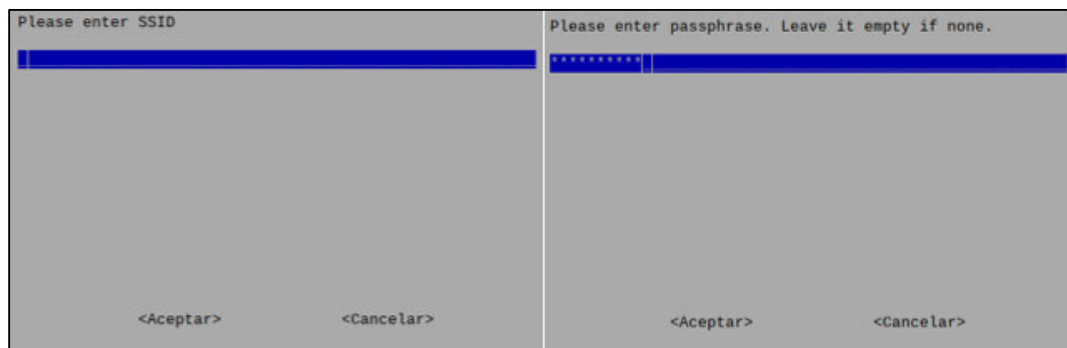
**Figura 2. 60.** Configuración de la red Wifi.

Otra forma de configura se realiza mediante el comando “*sudo raspi-config*” que permite ingresar a la interfaz con múltiples opciones, en el cual se dirige al apartado *System Options* donde se encuentra la opción *Wireless LAN*, como se muestra en la Figura 2.61



**Figura 2. 61.** Opción de configuración de red Wifi.

Al seleccionar esta opción se despliega una primera pantalla para ingresar el nombre de la red, luego otra pantalla para ingresar la contraseña como se muestra en la Figura 2.62 y la red quedará configurada.



**Figura 2. 62.** Configuración de la red Wifi.

### 2.5.10. CREACIÓN DE ACCESO DIRECTO A LA APLICACIÓN

La creación de un acceso directo a la aplicación facilita al usuario que al hacer un clic se despliegue toda la información que esta contenga, por tanto, para la creación del acceso directo de la aplicación CLIMA se debe en primer lugar a cambiar los permisos de ejecución para todos los usuarios sin excepción al script que contiene las líneas de código de la aplicación mediante el Código 2.111.

```
pi@raspberrypi: ~ $ chmod +x PROYECTO/CODIGOS/APP/AppCumbaya.py
```

**Código 2. 108.** Cambio de permisos de ejecución.

El segundo paso es crear un archivo desktop en el directorio `.local/share/applications/` con el Código 2.112, en el cual se editan las líneas de comandos que se muestran en la Figura

2.63, donde se define el nombre de la aplicación, la ruta del ícono de aplicación y la ruta del archivo para ejecutar.

```
pi@raspberrypi: ~$ nano .local/share/applications/APPCLIMA.desktop
```

**Código 2. 109.** Creación de archivo desktop

```
GNU nano 3.2 .local/share/applications/APPCLIMA.desktop
[Desktop Entry]
Name=APPCLIMA
Version=v0.8.5
Icon=/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/logoapp.PNG
Exec=python3 /home/pi/PROYECTO/CODIGOS/APP/AppCumbaya.py
Terminal=false
Type=Application
Hidden=false
StartupNotify=false
```

**Figura 2. 63.** Edición del archivo desktop.

Posteriormente, se agregan permisos de ejecución al archivo desktop creado para que cualquier usuario pueda ejecutar y ver el contenido de la aplicación creada, este proceso se realiza mediante el Código 2.113.

```
pi@raspberrypi: ~$ chmod +x .local/share/applications/APPCLIMA.desktop
```

**Código 2. 110.** Asignación de permisos al archivo desktop.

El siguiente paso a seguir consiste en dirigirse a la barra de tareas de la interfaz de Raspberry Pi dar clic derecho para desplegar las opciones en las cuales se escoge el apartado “Barra de Aplicaciones Settings” como se muestra en la Figura 2.64.



**Figura 2. 64.** Barra de aplicaciones.

Con lo que se despliega una nueva ventana que permite mostrar o no la aplicación en la barra de herramientas de la interfaz de Raspberry Pi. Se debe considerar que, para encontrar la aplicación configurada mediante el archivo desktop se debe dirigir a la opción “Otras”, y se escoge la aplicación “APPCLIMA” y se presiona en “añadir”, como se muestra en la Figura 2.65.



**Figura 2. 65.** Añadir APPCLIMA a la barra de tareas.

Finalmente se tiene el acceso directo creado como se muestra en la Figura 2.66 que solo con un clic mostrará la información requerida.



**Figura 2. 66.** Acceso directo a la APPCLIMA.

### 3. RESULTADOS Y DISCUSIÓN

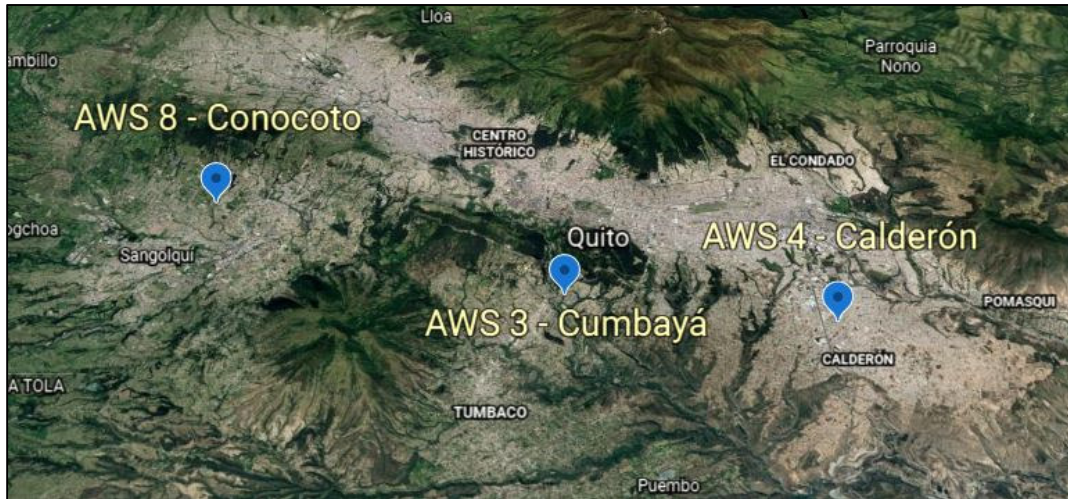
En este capítulo se presenta la instalación de las AWS (*Automatic Weather Stations*, Estaciones Meteorológicas Automáticas) en las zonas de Cumbayá, Conocoto y Calderón con las respectivas pruebas de funcionamiento de cada una de las etapas que conforman el sistema, además, se realiza la calibración de los sensores que conforman la AWS de acuerdo a las pruebas desarrolladas en el INAMHI (Instituto Nacional de Meteorología e Hidrología), el análisis de los resultados y los errores cometidos se adquieren mediante la comparación de los pronósticos de temperatura y humedad realizadas por los tres modelos de redes neuronales implementados y los valores reales adquiridos por la estación, con los cuales se pretende elegir la red de mejor desempeño, finalmente se presenta un presupuesto referencial del sistema prototipo que se conforma de tres AWS.

#### 3.1. POSICIONAMIENTO DE LAS AWS

El sistema prototipo se compone de tres AWS instaladas en tres valles de la ciudad de Quito como Cumbayá, Conocoto y Calderón, la instalación se realizó en espacios urbanos teniendo en cuenta las recomendaciones de la WMO, las coordenadas de las diferentes AWS se presentan en la Tabla 3.1, al igual que la ubicación geográfica presentada en la Figura 3.1, por tanto, las variaciones de parámetros meteorológicos de las AWS serán evidentes debido a factores como: suelo, geomorfología, vegetación; entre otros, los cuales influyen en la formación de microclimas característicos de cada zona.

**Tabla 3. 1.** Coordenadas geográficas de las AWS.

<b>AWS</b>	<b>Latitud</b>	<b>Longitud</b>	<b>Altitud (m)</b>
3 - Cumbayá	0°11'25.53"S	78°26'38.11"W	2476
4 - Calderón	0°06'04.0"S	78°26'17.0"W	2673
8 - Conocoto	0°18'55.0"S	78°28'35.0"W	2506



**Figura 3. 1.** Ubicación geográfica de las AWS.

## **3.2. CONSTRUCCIÓN E INSTALACIÓN DE LAS AWS**

### **3.2.1. CONSTRUCCIÓN DE CASETA METEOROLÓGICA**

Con el fin de brindar protección a los equipos y sensores que conforman la AWS de fenómenos naturales como: la lluvia, radiación solar, viento, entre otros; se procedió a construir una caseta meteorológica que cumpla las recomendaciones de infraestructura descritas en el Capítulo 1.

El material utilizado para la construcción de la garita o caseta meteorológica es madera tratada recubierta por una mezcla de agua y cola de madera quien actúa como sellador para cerrar los poros de la madera y crear una superficie fácil de pintar, posee cuatro persianas con una inclinación de 45° para la circulación libre del aire e impedir la radiación directa del sol, además, se compone de dos espacios separados entre sí para evitar el calentamiento de la garita y el otro inclinado para que el agua caiga.

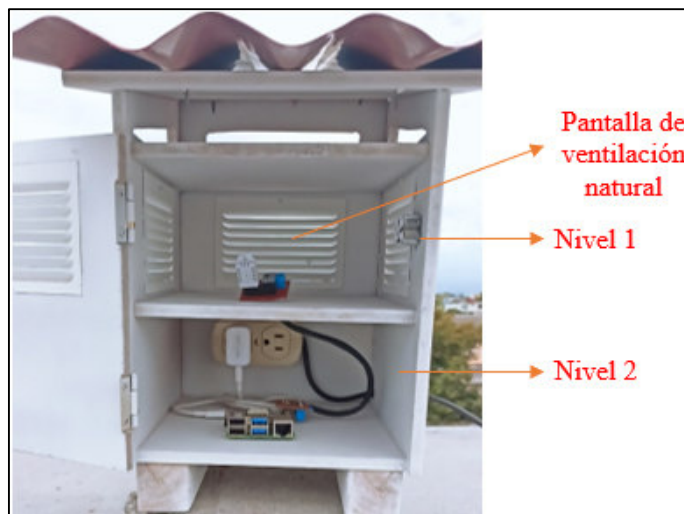
La pintura utilizada es de tipo esmaltada adecuada para superficies que están expuestas a condiciones naturales externas, por tanto, este tipo de pintura brinda mayor resistencia, refleja los rayos directos del sol y no absorbe humedad, además cuenta con un techo de plástico que permite proteger a la garita de la lluvia y dicha cubierta está inclinada de tal forma que el agua caiga, la caseta terminada se presenta en la Figura 3.2.



**Figura 3. 2.** Garita meteorológica terminada.

### 3.2.2. INSTALACIÓN DE EQUIPOS

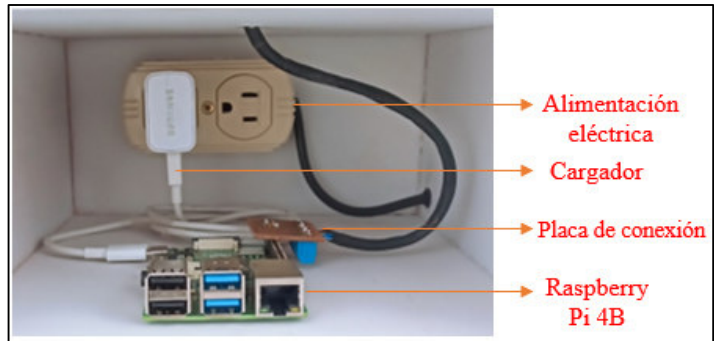
La caseta o garita meteorológica está compuesta de dos niveles como se muestra en la Figura 3.3.



**Figura 3. 3.** Instalación de equipos.

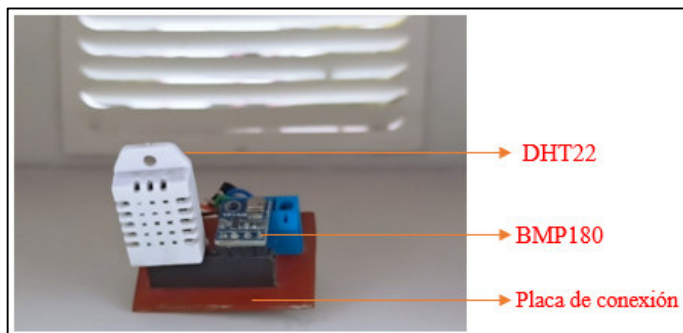
El primer nivel (inferior) alberga la Raspberry Pi 4B, el tomacorriente de alimentación y la placa diseñada para la conexión a la Raspberry Pi, como se observa en la Figura 3.4.





**Figura 3.4.** Instalación de equipos en el nivel 1.

El segundo nivel construido con pantallas de ventilación natural aloja la placa de conexión a los sensores y los sensores de temperatura/humedad relativa y presión atmosférica, como se muestra en la Figura 3.5, la conexión entre placas se realiza mediante cable UTP multipar.



**Figura 3. 5.** Instalación de equipos en el nivel 2.

### 3.2.3. INSTALACIÓN DE AWS EN VIVIENDAS

La instalación en las diferentes localidades de las AWS se las realizó sobre terrazas a la máxima altura posible, no contaban con obstrucciones a su alrededor, estaban niveladas, no se provocaba inundaciones y sobre todo los lugares son accesibles para la manipulación y mantenimiento de las AWS, el resultado de la instalación en las zonas se presenta en la Figura 3.6 como referencia.



**Figura 3. 6.** Instalación de AWS en localidades.

### **3.3. PRUEBAS DE FUNCIONAMIENTO**

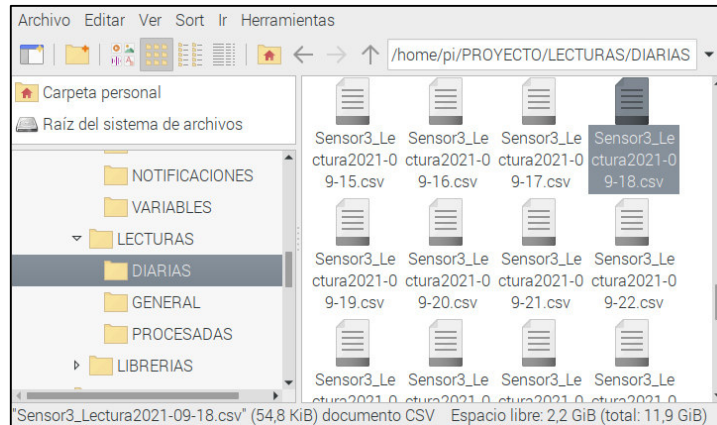
A continuación, se muestran las pruebas de funcionamiento de los scripts automatizados, proceso realizado en el Capítulo 2 en la sección 2.5.8.

#### **3.3.1. PRUEBAS DE ADQUISICIÓN DE DATOS**

- **Lectura de datos y almacenamiento en archivos csv.**

El script automatizado para este proceso se ejecuta 1 min después de que la AWS se enciende, el tiempo de espera se da para estabilizar la conexión a internet, con el fin de notificar al administrador de las AWS sobre eventos que dificulten la adquisición de datos de las variables meteorológicas.

En la Figura 3.7 se observa la creación de archivos tipo csv en la dirección /home/pi/PROYECTO/LECTURAS/DIARIAS, que se identifican de acuerdo al número de estación designados y la fecha de creación correspondientes.



**Figura 3. 7.** Creación de archivos csv.

Al ingresar al archivo se puede observar el almacenamiento de las variables atmosféricas de temperatura, humedad relativa y presión, así como la fecha, hora de adquisición de datos y la numeración del nodo al que pertenece en diferentes columnas, en la Figura 3.8 se presenta la adquisición y almacenamiento de datos del día 18/09/2021 perteneciente al nodo 3 de Cumbayá.

The screenshot shows a spreadsheet application window with the following data in the table:

	A	B	C	D	E	F
1	2021-09-18	00:00:43	3	15.9	58.2	761.8
2	2021-09-18	00:01:43	3	15.9	57.8	761.8
3	2021-09-18	00:02:43	3	15.9	57.3	761.8
4	2021-09-18	00:03:43	3	15.9	57.2	761.8
5	2021-09-18	00:04:43	3	15.9	57.5	761.8
6	2021-09-18	00:05:43	3	15.9	57	761.8
7	2021-09-18	00:06:43	3	15.7	57.6	761.8
8	2021-09-18	00:07:44	3	15.6	58	761.7
9	2021-09-18	00:08:43	3	15.6	57.4	761.7

The spreadsheet title is 'Sensor3\_Lectura2021-09-18' and it is 'Sheet 1 of 1'.

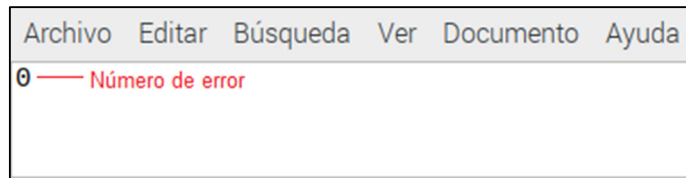
**Figura 3. 8.** Almacenamiento de datos en csv.

- **Notificación de eventos al adquirir datos**

Los eventos que pueden ocurrir en la adquisición de datos son:

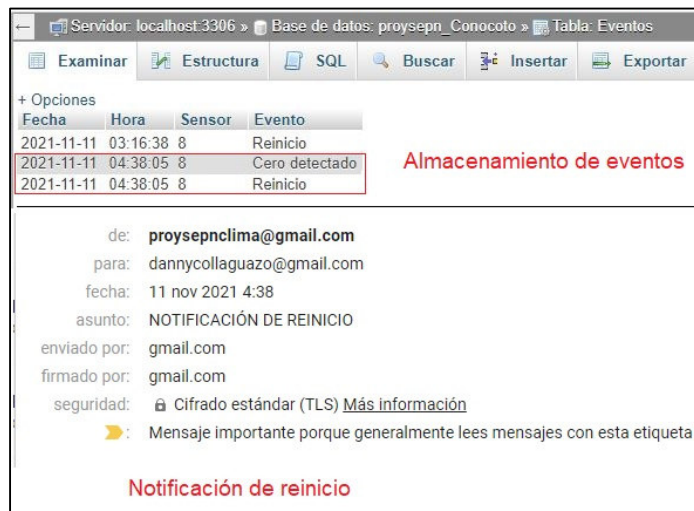
1. Detección de 0 en los valores de temperatura o humedad, en este caso la información del evento se almacena en la base de datos propia de la AWS en la tabla definida como Eventos; se debe mencionar que al ocurrir este evento la AWS se reinicia.
2. Persistencia en lectura de 0 en los sensores, en este tipo de suceso se procede a informar mediante correo electrónico al administrador de las AWS.

De acuerdo a la información del número de error almacenado en un archivo de texto plano, el cual se presenta en la Figura 3.9, la AWS tomará la acción correspondiente, este evento se dio en el nodo 8 Conocoto.



**Figura 3.9.** Error en adquisición de datos.

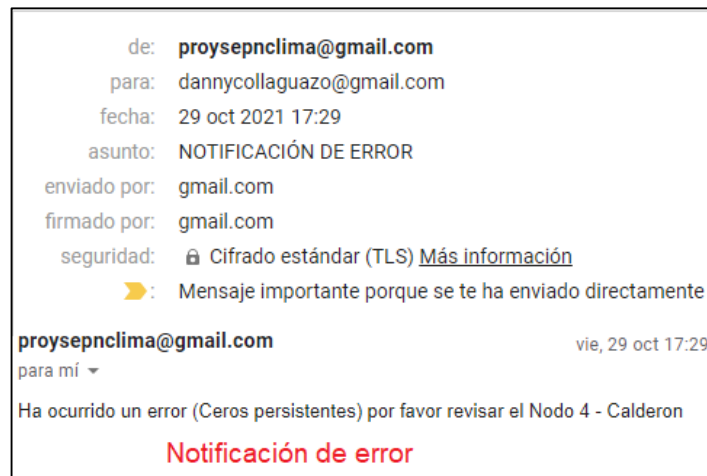
Por tanto, si el número de error es igual a 1 la AWS almacenará el evento de “Detección de cero” y reiniciará la AWS con el fin de corregir dicho error como se presentó en el nodo 8 Conocoto el 11/11/2021, proceso que se muestra en la Figura 3.10.



**Figura 3.10.** Almacenamiento de evento y notificación de reinicio.

Por otro lado, si el valor es igual a 2, entonces el problema 1 sigue persistiendo, por ello, la AWS informará acerca del error “Ceros persistentes” al administrador mediante correo electrónico y tratará de adquirir datos hasta su respectiva revisión, este evento ocurrió en el nodo 4 Calderón el 29/10/2021, sin embargo, se debe mencionar que por falla de conexión a internet no se almacenaron los eventos anteriores a este como: cero detectado

y reinicio, por tanto, en la Figura 3.11 se presenta el correo electrónico notificando el suceso ocurrido.



**Figura 3.11.** Notificación de ceros persistentes.

### 3.3.2. PRUEBAS DE TRANSMISIÓN Y ALMACENAMIENTO DE DATOS

Las pruebas de funcionamiento de la segunda etapa se realizan de acuerdo al tiempo establecido en la automatización de los scripts para el almacenamiento de parámetros meteorológicos en la base de datos y almacenamiento de archivos en el servidor FTP presentados en el Capítulo 2.

- **Almacenamiento de datos en el servidor MySQL alojado en la nube**

Los datos de fecha, hora, número de estación, valores de temperatura, humedad y presión atmosférica son almacenados cada 10 min en una tabla definida como LecturaSensor dentro de una base de datos propia de la AWS en el servidor MySQL del hosting web alojado en la nube. En la Figura 3.12 se pueden observar los datos almacenados correspondientes al día 18/09/2021 de la AWS 3 de Cumbayá en una base de datos definida con el mismo nombre.

Fecha	Hora	Sensor	Temperatura	Humedad	Presion
2021-09-18	10:06:46	3	18.7	59.8	762.0
2021-09-18	10:16:46	3	19.5	56.6	762.0
2021-09-18	10:26:46	3	20.2	53.9	762.0
2021-09-18	10:36:46	3	19.2	54.5	761.8
2021-09-18	10:46:46	3	20.7	50.8	761.8
2021-09-18	10:56:46	3	20.4	49.6	761.7
2021-09-18	11:06:46	3	22.0	48.0	761.7
2021-09-18	11:16:46	3	21.9	46.6	761.5
2021-09-18	11:26:46	3	22.9	44.9	761.4
2021-09-18	11:36:47	3	23.6	41.6	761.2
2021-09-18	11:46:47	3	24.1	39.6	761.1
2021-09-18	11:56:47	3	25.6	38.5	760.9
2021-09-18	12:06:47	3	24.7	36.2	760.6
2021-09-18	12:16:47	3	26.2	33.0	760.5
2021-09-18	12:26:47	3	26.1	31.5	760.3
2021-09-18	12:36:47	3	24.8	32.9	760.2
2021-09-18	12:46:47	3	24.7	32.0	760.0
2021-09-18	12:56:47	3	25.0	32.9	759.9
2021-09-18	13:06:47	3	25.4	32.7	759.7
2021-09-18	13:16:47	3	24.5	33.1	759.7
2021-09-18	13:26:47	3	25.3	31.6	759.4
2021-09-18	13:36:47	3	23.5	35.0	759.4
2021-09-18	13:46:47	3	22.7	35.9	759.4
2021-09-18	13:56:47	3	23.7	34.1	759.3
2021-09-18	14:06:47	3	23.5	35.8	759.2

Figura 3.12. Almacenamiento de datos en el servidor MySQL.

Estos datos se pueden corroborar en el archivo almacenado en la memoria SD de la AWS la cual se presenta en la Figura 3.13.

	A	B	C	D	E	F	G	H
607	2021-09-18	10:06:46	3	18.7	59.8	762		
608	2021-09-18	10:07:47	3	18.9	59.7	762		
609	2021-09-18	10:08:46	3	19.1	58.9	762		
610	2021-09-18	10:09:46	3	19.3	59	762		
611	2021-09-18	10:10:46	3	19.4	58.6	762		
612	2021-09-18	10:11:46	3	19.5	57.7	762		
613	2021-09-18	10:12:46	3	19.6	57.5	762		
614	2021-09-18	10:13:46	3	19.6	57.2	762		
615	2021-09-18	10:14:46	3	19.5	56.6	762		
616	2021-09-18	10:15:46	3	19.5	56.5	762.1		
617	2021-09-18	10:16:46	3	19.5	56.6	762		
618	2021-09-18	10:17:47	3	19.4	56	762		
619	2021-09-18	10:18:46	3	19.5	55.8	762		
620	2021-09-18	10:19:46	3	19.5	55.4	762		
621	2021-09-18	10:20:46	3	19.6	55.6	762		
622	2021-09-18	10:21:46	3	19.7	55.6	762		
623	2021-09-18	10:22:46	3	19.9	54.8	761.9		
624	2021-09-18	10:23:46	3	20	54.6	762		
625	2021-09-18	10:24:46	3	20.1	54.2	762		
626	2021-09-18	10:25:46	3	20.2	53.9	761.9		
627	2021-09-18	10:26:46	3	20.2	53.9	762		

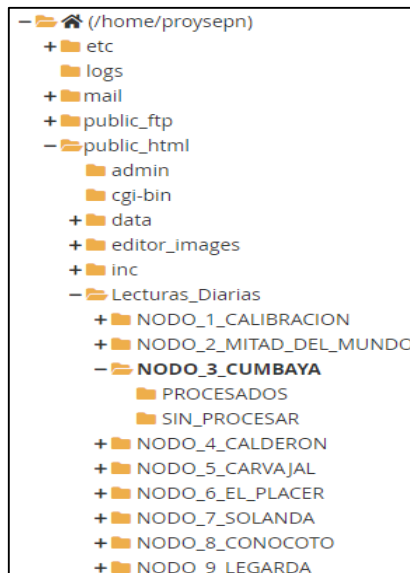
Figura 3.13. Datos subidos a MySQL.

- Almacenamiento de archivos en el servidor FTP alojado en la nube

Los archivos que contienen los datos de los parámetros meteorológicos son almacenados diariamente en un servidor FTP alojado en la nube al final de cada día, son dos tipos de

archivos que almacena el servidor, el primero es el archivo diario original, es decir, sin interpolar y el otro es el archivo interpolado en caso de haber datos faltantes, por tanto, los archivos se suben luego de realizar el proceso de interpolación de acuerdo a la hora programada para cada proceso. Los archivos pueden ser descargados desde el Panel de Control al cual se ingresa con las credenciales entregadas por el proveedor del servicio contratado, como dirección, usuario y contraseña.

Los archivos subidos se mantienen en orden y se los puede encontrar fácilmente gracias al administrador de archivos que incorpora el Panel de Control, quiere decir, que al igual que la AWS, el servidor FTP contiene jerarquías en sus directorios para albergar subdirectorios y archivos como se puede observar en la Figura 3.14.



**Figura 3.14.** Jerarquía de directorios en el servidor FTP.

Como se puede observar las AWS contienen dos subdirectorios llamados PROCESADOS y SIN\_PROCESAR, donde se guardan los archivos interpolados y originales respectivamente, en la Figura 3.15 se muestran los archivos originales almacenados al final de cada día de acuerdo a la automatización del script de almacenamiento de archivos, lo mismo ocurre con los archivos interpolados.

Name	Size	Last Modified	Type	Permissions
Sensor3_Lectura2021-09-18.csv	54.84 KB	Sep 19, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-19.csv	54.81 KB	Sep 20, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-20.csv	54.81 KB	Sep 21, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-21.csv	54.84 KB	Sep 22, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-22.csv	54.81 KB	Sep 23, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-23.csv	54.84 KB	Sep 24, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-24.csv	54.81 KB	Sep 25, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-25.csv	54.81 KB	Sep 26, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-26.csv	54.84 KB	Sep 27, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-27.csv	54.84 KB	Sep 28, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-28.csv	54.84 KB	Sep 29, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-29.csv	54.84 KB	Sep 30, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-09-30.csv	54.84 KB	Oct 1, 2021, 12:10 AM	text/x-generic	0644
Sensor3_Lectura2021-10-01.csv	54.84 KB	Oct 2, 2021, 12:10 AM	text/x-generic	0644

Figura 3.15. Almacenamiento de archivos originales en el servidor FTP.

### 3.3.3. PRUEBAS DE MONITOREO Y VISUALIZACIÓN DE DATOS

En esta sección se presentan las pruebas realizadas sobre el monitoreo continuo de datos mediante scripts como el chequeo de datos de los archivos originales, la interpolación por datos faltantes con sus respectivas notificaciones de acuerdo a los eventos que pueden ocurrir en una AWS y la visualización de datos en la aplicación.

- **Chequeo de datos**

El chequeo de datos se realiza mediante un script que se ejecuta cada 5 min, que cumple la función de conteo de datos totales y los compara con los guardados anteriormente, la Figura 3.16, muestra el proceso de almacenamiento de dos variables en un archivo de texto plano para seleccionar las acciones requeridas y mitigar el error que se presenta:

1. Almacenamiento de eventos en la base de datos y reinicio de la AWS
2. Envío de notificación de error.

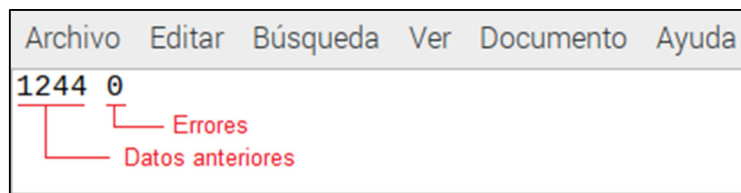


Figura 3.16. Almacenamiento de datos anteriores y errores.

La opción 1 se da cuando la comparación entre datos actuales y datos anteriores da un valor de 0 y el número de errores es 1, por tanto, en la Figura 3.17. se muestra el



almacenamiento del evento en la base de datos, el almacenamiento de reinicio por el acontecimiento ocurrido y la notificación de reinicio del nodo enviado al administrador, evento ocurrido en el nodo 4 de Calderón el día 29/10/2021.

The screenshot shows a database interface for 'Tabla: Eventos' on 'localhost:3306'. It displays a table with the following data:

Fecha	Hora	Sensor	Evento
2021-10-29	17:50:12	4	Reinicio
2021-10-29	18:00:02	4	Datos no detectados
2021-10-29	18:01:22	4	Reinicio

Below the table, there is a section titled 'Almacenamiento de eventos' and an email notification. The email is from 'proysepnclima@gmail.com' to 'dannycollaguazo@gmail.com', dated '29 oct 2021 18:01', with the subject 'NOTIFICACIÓN DE REINICIO'. The body of the email states: 'El nodo 4 - Calderon ha sido reiniciado el 2021-10-29'. Below the email content is the title 'Notificación de reinicio'.

Figura 3.17. Almacenamiento de evento y notificación de reinicio.

La opción 2, que corresponde al número de error 2, se da cuando la AWS no puede resolver el error con el reinicio, entonces la acción de notificar al administrador se activa, en la Figura 3.18. se muestra la notificación del evento "Error de adquisición de datos", evento ocurrido en el nodo 4 de Calderón el día 30/10/2021.

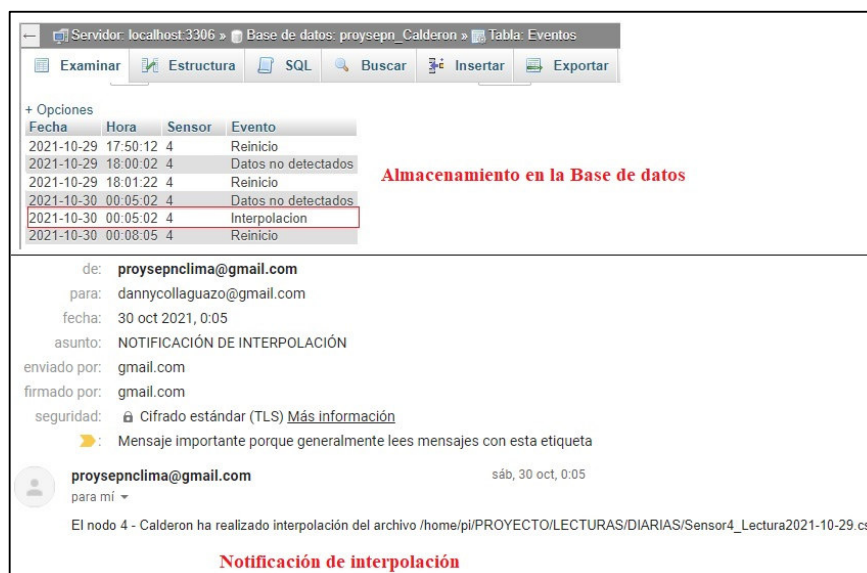
The screenshot shows an email notification from 'proysepnclima@gmail.com' to 'dannycollaguazo@gmail.com', dated '30 oct 2021, 0:10', with the subject 'NOTIFICACIÓN DE ERROR'. The body of the email states: 'Revisar el Nodo 4 - Calderon, presenta errores de adquisición de datos'. Below the email content are buttons for 'Responder' and 'Reenviar', and the title 'Notificación de error'.

Figura 3.18. Notificación de error de adquisición de datos.

- **Interpolación de datos**

Como bien se mencionó la AWS tiene la capacidad de rellenar datos faltantes <sup>18</sup>en los archivos diarios, es decir, en el caso de que un archivo no contenga los 1440 datos el script lo solucionará interpolando datos en la hora y minuto en que la AWS no adquirió datos de temperatura, humedad relativa y presión al final de cada día.

Este evento se almacena en la base de datos propia de la AWS de cada zona en la tabla Eventos en el servidor MySQL del hosting web alojado en la nube y además se informa al administrador mediante correo electrónico sobre el proceso de interpolación, la Figura 3.19 muestra este proceso de interpolación del archivo Sensor4\_Lectura2021-10-29 en el nodo 4 de Calderón debido al evento previo “Error de detección” dado el mismo día.



**Figura 3.19.** Almacenamiento y notificación de interpolación.

- **Acceso remoto por TeamViewer**

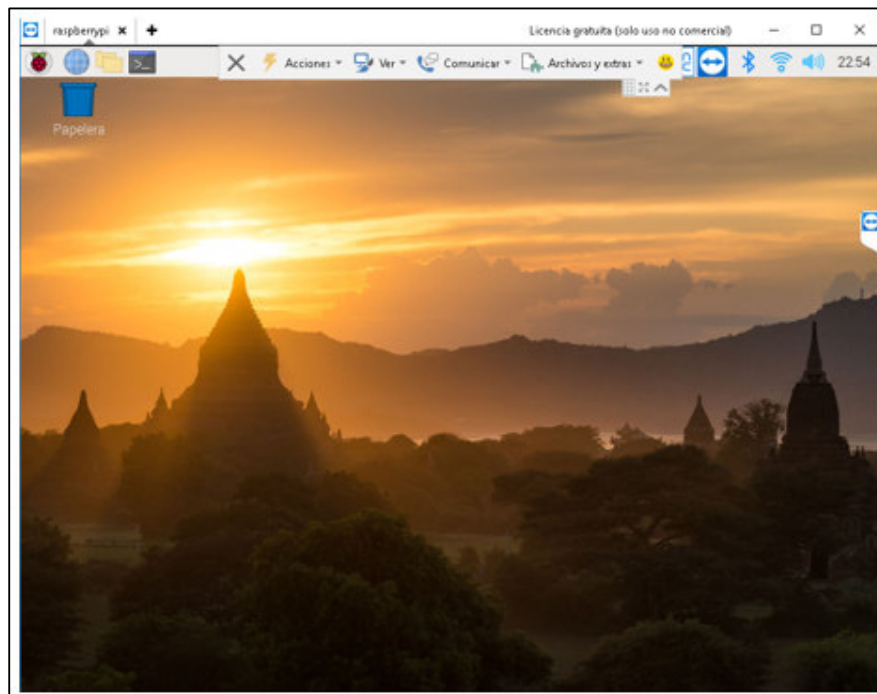
La aplicación de TeamViewer es sencilla de usar, para acceder de forma remota a las estaciones es necesario introducir el ID único de la AWS y la contraseña configurada en la instalación, este proceso se muestra en la Figura 3.20.

<sup>18</sup> **Datos faltantes.** - Este inconveniente puede presentarse por la falta de alimentación eléctrica a las AWS, saturación de los scripts, mala conexión de los dispositivos electrónicos y falta de mantenimiento a las estaciones por largos periodos de tiempo.



**Figura 3.20.** Acceso remoto por TeamViewer.

Si los datos son correctos entonces se puede visualizar la pantalla o interfaz de la AWS como se muestra en la Figura 3.21, el acceso remoto permite controlar la estación, monitorear y visualizar los datos, integrar nuevas funciones a la estación y realizar varias operaciones mediante la aplicación integrada.



**Figura 3.21.** Acceso remoto a la AWS.

- **Aplicación desarrollada**

La aplicación cuenta con un acceso directo en la barra de herramientas de la interfaz de la Raspberry Pi, como se muestra en la Figura 3.22, esto proporciona la facilidad de dar un clic e ingresar a la pantalla principal del sistema de almacenamiento y visualización de datos como se presentó en la Figura 2.46 de acuerdo al proceso desarrollado en el Capítulo 2.



**Figura 3.22.** Acceso directo a la aplicación.

El uso de la aplicación es muy sencillo ya que se desarrolló una interfaz básica e intuitiva para el usuario, muchas opciones en la interfaz cuentan con mensajes a los cuales se puede aceptar o negar de acuerdo a las necesidades del administrador del dispositivo o en vista de los mensajes de información de alguna actividad realizada, como, por ejemplo, la presentada en la pantalla inicial del sistema de la Figura 3.23.



**Figura 3.23.** Modelo de mensajes en la aplicación.

Al ingresar a la segunda interfaz se puede visualizar todos los datos almacenados en la base de datos del servidor MySQL alojado en la nube, los archivos csv almacenados en la memoria SD y las lecturas que adquiere en ese instante la AWS, como se puede observar en la Figura 3.24, la interfaz se conforma de tres partes que cumplen diversas operaciones.

En la primera parte denominada DATOS Y CONSULTAS se encuentra funciones de búsqueda e importación de archivos o base de datos, retorno de información de la base de

datos, salida del sistema y espacios de visualización de datos. La segunda parte definida como EVENTOS cuenta con espacio de visualización de los acontecimientos ocurridos en la AWS y almacenados en la base de datos, mientras que la última sección nombrada como PROGRAMACIÓN DE REINICIO cuenta con las opciones de reinicio que incorpora la aplicación.

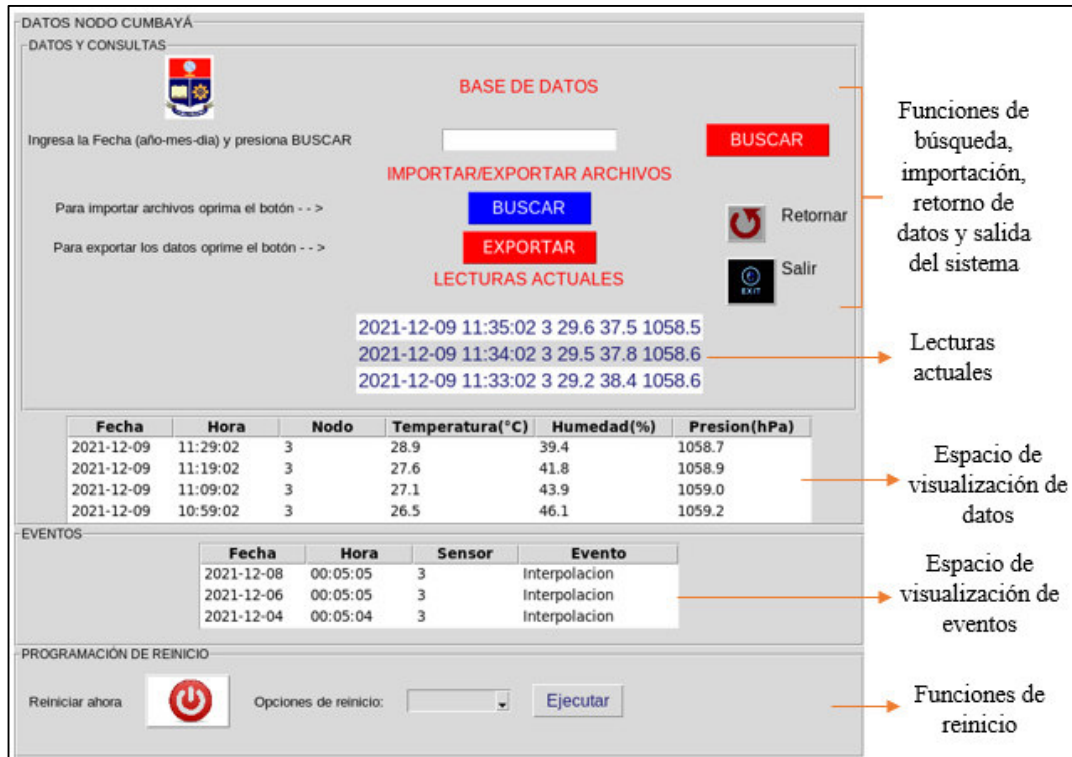


Figura 3.24. Interfaz de datos.

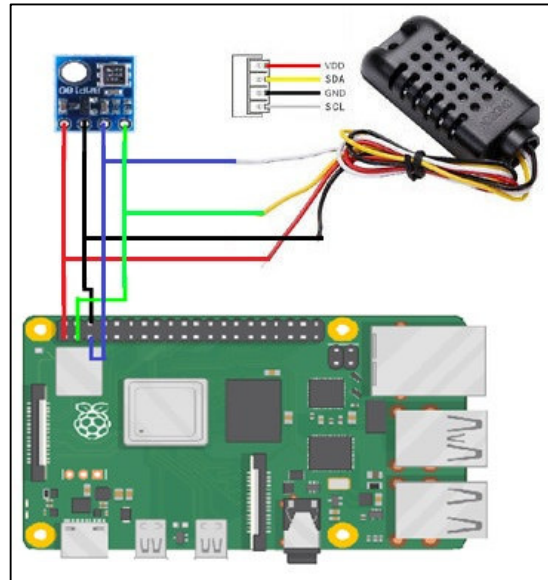
En el Anexo II: Guía de utilización de la aplicación, se presenta el funcionamiento de cada uno de los botones y las funciones que estos incorporan para el monitoreo y visualización de datos, además, la programación de reinicio.

### 3.4. CALIBRACIÓN DE SENSORES

Los sensores DHT22 y BMP180 utilizados en las AWS vienen calibrados por defecto de fábrica, sin embargo, para asegurar que los valores de las variables meteorológicas sean precisos se ejecutó el proceso de calibración de una AWS que sirvió como referencia para el resto de las estaciones.

La AWS de calibración se integró con dos tipos de sensores: el sensor digital DHT21 que adquiere valores de temperatura/humedad relativa en un solo dispositivo, sensor de buenas prestaciones para trabajar en exteriores por la robustez de su empaque y el sensor

BMP180 que adquiere valores de presión atmosférica. Cabe mencionar que las especificaciones técnicas de estos sensores que trabajan con el protocolo de comunicación I2C se presentaron en el Capítulo 1, la Figura 3.25 presenta la conexión de los dos sensores a la Raspberry Pi.



**Figura 3.25.** Diagrama de conexión de los sensores de calibración.

La calibración de los sensores descritos se la realizó en el INAMHI ubicado en Quito en la calle Núñez de Vela N36-15 y Corea, institución que cuenta con equipos y profesionales certificados para calibrar dispositivos que miden variables meteorológicas, el proceso se realizó en el área de CALIBRACIÓN Y MANTENIMIENTO DE INSTRUMENTAL HIDROMETEOROLÓGICO en aproximadamente 15 días laborales.

- **Calibración del parámetro temperatura**

La calibración de este parámetro se realizó en el laboratorio de Temperatura y Humedad mediante el dispositivo de calibración de horno de bloque seco FLUKE. El proceso de calibración consiste en, ingresar el dispositivo en su interior y variar la temperatura dentro de este, con el fin de adquirir los valores en ciertos puntos de temperatura y efectuando el método de comparación directa como el método matemático de mínimos cuadrados se puede determinar la recta de regresión lineal y obtener los coeficientes de calibración del sensor DHT21. La Figura 3.26 se muestra el instrumento de calibración utilizado en el INAMHI para la calibración del sensor DHT21 en la variable meteorológica de temperatura.



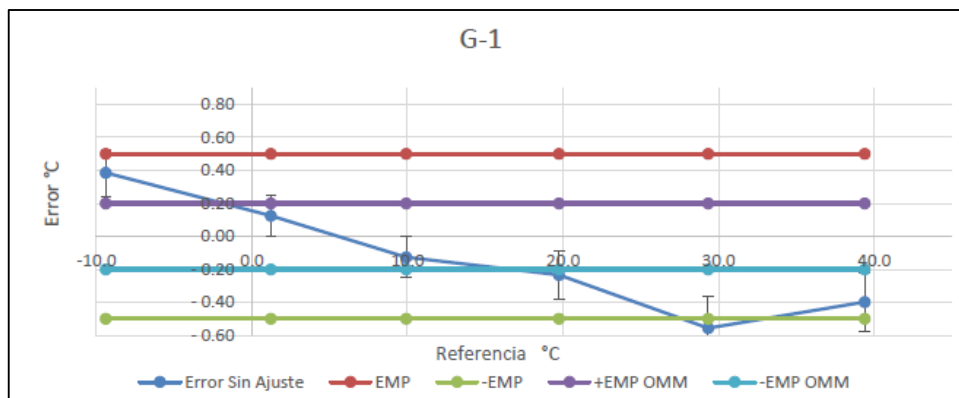
**Figura 3.26.** Bloque seco FLUKE.

El intervalo de tiempo necesario para la adquisición de los datos entre cada punto de temperatura es de aproximadamente de 2h a 3h, una vez definido el punto de temperatura para adquirir los datos es necesario esperar hasta que el instrumento de calibración se estabilice en dicho valor y proceder a almacenar un valor de temperatura por minuto durante los próximos 10 minutos y obtener un promedio general de dichos datos. La Tabla 3.2 muestra los valores promedio obtenidos en los puntos de temperatura estándar del INAMHI, tanto del instrumento de calibración FLUKE (Lp) y del sensor a calibrar DHT21 (Li), además, se presenta la comparación a través del cálculo de error absoluto (EA) del parámetro antes del ajuste.

**Tabla 3. 2.** Comparación de datos de temperatura sin ajuste.

Puntos (°C)	Lp	Li	EA
-10	-9.785	-9.4	0.385
-1	1.085	1.2	0.125
10	10.047	9.9	-0.127
20	19.974	19.7	-0.234
30	29.887	29.887	-0.557
40	39.798	39.798	-0.398

De acuerdo a la tabla antes descrita y a la Figura 3.27 en la cual se representa el intervalo de error medio probable que debe cumplir el sensor para estar dentro de los límites permitidos por la WMO ( $\pm 0.2^{\circ}\text{C}$ ), se observa que el sensor no cumple con dichas especificaciones, pero sí con la precisión ( $\pm 0.5^{\circ}\text{C}$ ), por tanto, es necesario calcular los coeficientes de ajuste.



**Figura 3.27.** Error de temperatura sin ajuste.

Los coeficientes de ajuste para los datos se adquieren mediante el proceso de mínimos cuadrados, la Ecuación 3.1 representa la recta de ajuste para la temperatura del sensor DHT21, donde: “y” es el valor de temperatura ajustado y “x” es el valor de temperatura adquirido, es decir sin ajuste.

$$y = 1.01790345x - 0.13471825 \quad (3.1)$$

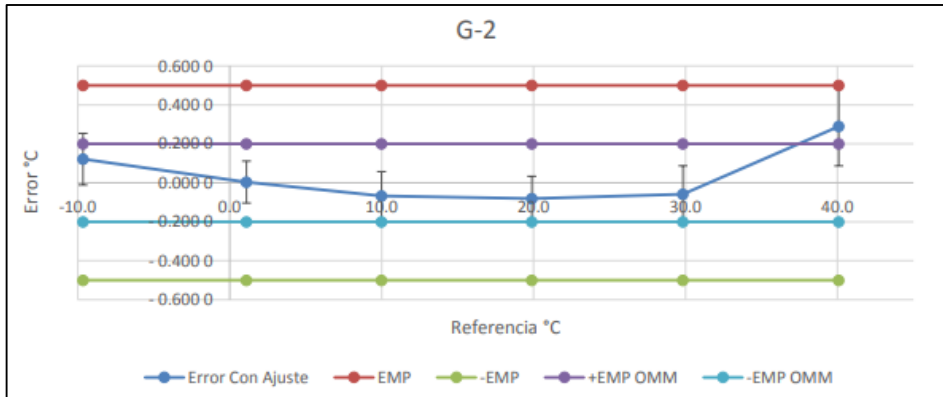
Las pruebas realizadas para verificar si los datos luego del ajuste minimizan los errores en cada punto siguen el mismo proceso antes descrito, la Tabla 3.3 describe los datos y errores obtenidos al implementar la ecuación en el código de lectura de datos de la AWS de calibración.

**Tabla 3.3.** Comparación de datos de temperatura con ajuste

Puntos (°C)	Lp	Li	EA
-10	-9.781	-9.7	0.121
-1	1.096	1.1	0.004
10	10.057	10	-0.067
20	19.961	19.9	-0.081
30	29.889	29.8	-0.059
40	39.801	40.1	-0.289

La comparación entre los datos de calibración y obtenidos por el sensor da a entender que el proceso de calibración se dio con éxito debido a que los errores están dentro de los establecidos por la WMO, sin embargo, el comunicado por parte de los técnicos del INAMHI recomiendan no operar en condiciones cercanas a los 40°C caso que no ocurre en Quito, ya que, aunque el error es mínimo, la ecuación no cumple su objetivo de ajuste, como se representa en la Figura 3.28.





**Figura 3.28.** Error de temperatura con ajuste.

- **Calibración del parámetro de humedad**

La calibración de este parámetro sigue el mismo proceso; se puede mencionar que toma de 2h a 3h la estabilización y adquisición de datos en cada punto estándar establecido por el INAMHI. La calibración del sensor se realiza mediante la cámara de humedad y temperatura THUNDER SCIENTIFIC del laboratorio de Temperatura y Humedad presentada en la Figura 3.29, que especifica los puntos de humedad y los genera.

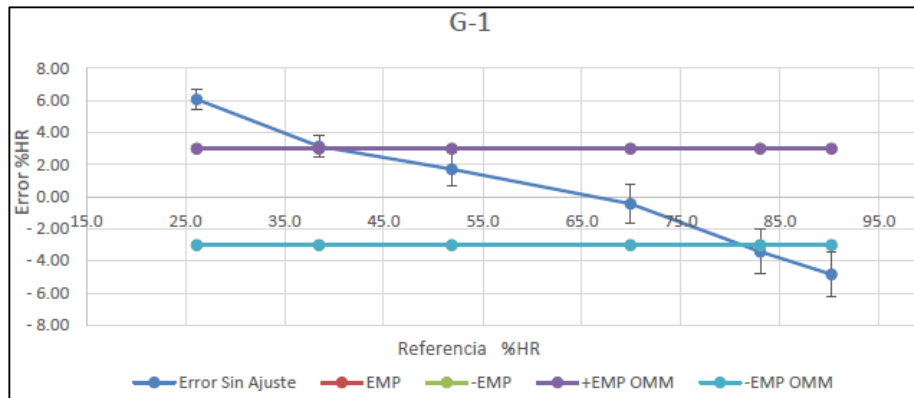


**Figura 3.29.** Cámara de humedad y temperatura.

La Tabla 3.2 muestra los valores promedio obtenidos en los puntos de humedad ajustados tanto del instrumento de calibración (Lp) y del sensor a calibrar (Li), el cálculo del error absoluto (EA) del parámetro antes del ajuste. Mediante el análisis de la Figura 3.30, se concluye que el sensor no cumple con las condiciones de la WMO ( $\pm 3\%$ ) y es necesario determinar los coeficientes de ajuste.

**Tabla 3. 4.** Comparación de datos de humedad sin ajuste.

Puntos (%)	Lp	Li	EA
20	20.08	26.1	6.06
35	35.35	38.5	3.13
50	50.17	51.9	1.72
70	70.34	69.9	-.043
85	86.43	83	-3.41
92	95.02	90.2	-4.83



**Figura 3.30.** Error de humedad sin ajuste.

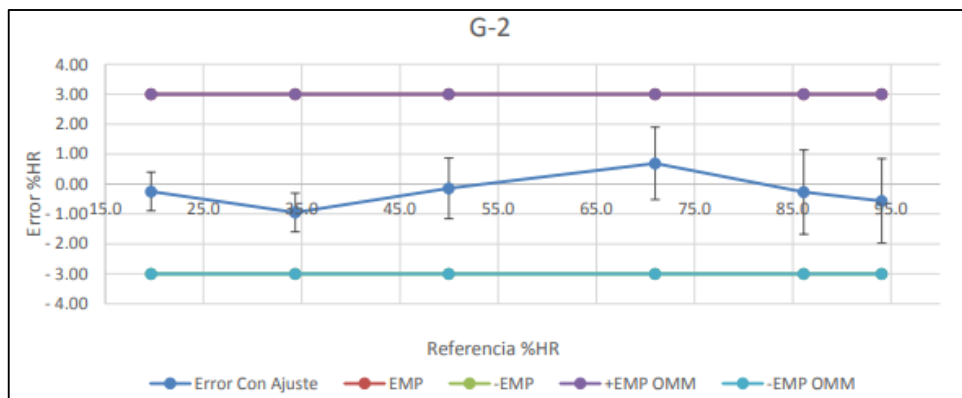
La ecuación 3.2 presenta la recta de ajuste, donde: “y” es el valor de humedad ajustado y “x” es el valor de humedad adquirido, es decir sin ajuste. La comparación de los datos luego de implementar la ecuación en el código de lectura de datos se presenta en la Tabla 3.5 con sus respectivos errores.

$$y = 1.16021050x - 9.97681594 \quad (3.2)$$

**Tabla 3. 5.** Comparación de datos de humedad con ajuste

Puntos (%)	Lp	Li	EA
20	19.92	19.7	-0.25
35	35.29	34.3	-0.95
50	50.12	50	-0.14
70	70.30	71	0.69
85	86.40	86.1	-0.27
92	94.62	94.1	-0.56

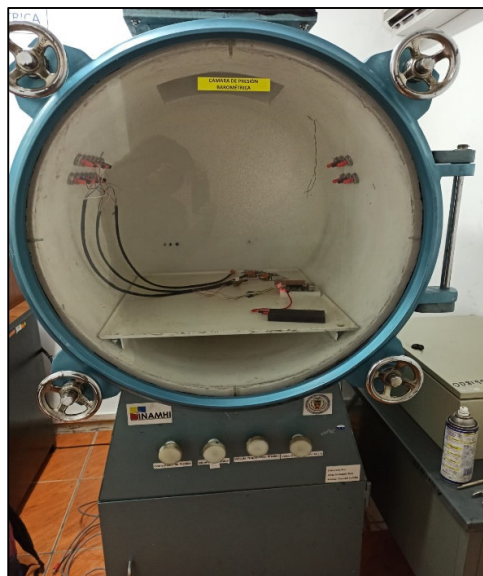
Los errores cometidos luego del ajuste de los coeficientes en los datos adquiridos presentan errores muy por debajo de los establecidos por la WMO, como se puede observar en la Figura 3.31, por ende, el parámetro de humedad fue ajustado con éxito.



**Figura 3. 31.** Error en la medición de humedad con ajuste.

- **Ajuste del parámetro de presión atmosférica**

El ajuste del sensor BMP180 se realizó mediante la cámara de presión barométrica THEODOR FRIEDRICHS presentado en la Figura 3.32, que al igual que los anteriores instrumentos se ajusta en determinados puntos de presión barométrica estándar del INAMHI para realizar las pruebas pertinentes de los sensores con la finalidad de determinar los errores cometidos y obtener la ecuación que ajuste las mediciones, el proceso toma de 2h a 3h por cada punto definido



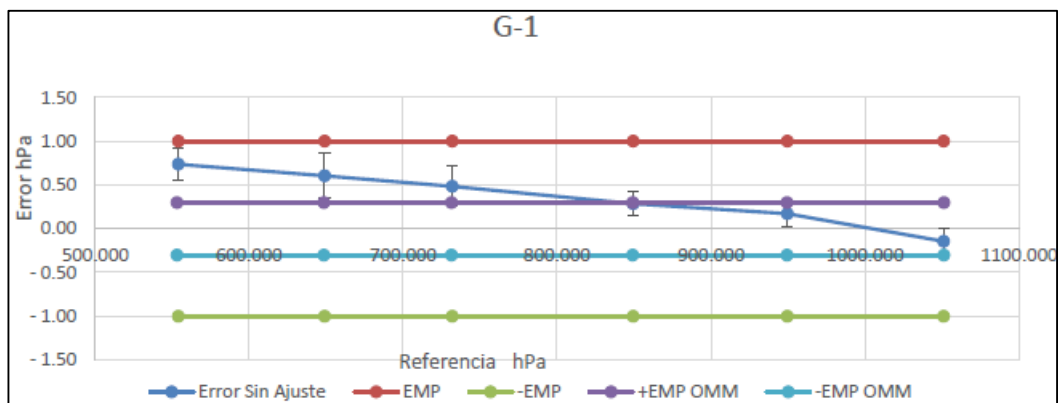
**Figura 3.32.** Cámara de presión barométrica.

La Tabla 3.5 y la Figura 3.33 muestran los errores del sensor BMP180 en comparación con los datos mostrados por los sensores calibrados del INAMHI, estos errores, aunque son

mínimos, requieren de una ecuación de ajuste para estar dentro de las condiciones de la WMO.

**Tabla 3. 6.** Comparación de datos de presión sin ajuste.

Puntos (hPa)	Lp	Li	EA
1050	1051.2929	1051.2	-0.143
950	949.4691	949.6	0.171
850	849.2018	849.5	0.288
730	731.3939	731.9	0.486
650	648.4245	649	0.606
550	553.1629	553.9	0.737



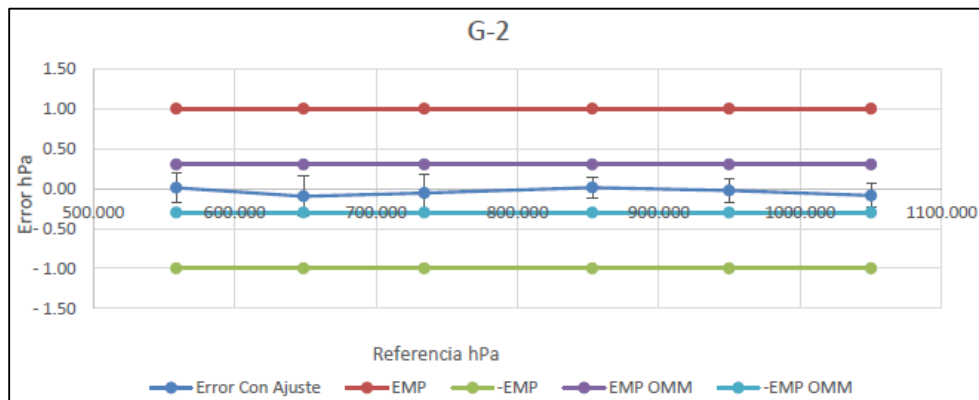
**Figura 3.33.** Error de la medición de presión sin ajuste.

La ecuación 3.3 representa la recta de ajuste para la medición de presión barométrica, donde: “y” es el valor de presión ajustado y “x” es el valor de presión adquirido, la implementación de esta ecuación da como resultado los datos de la Tabla 3.7 y se representa el error de la comparación en la Figura 3.34 con la cual se obtiene que la calibración se ha realizado satisfactoriamente y está dentro de las condiciones especificadas por la WMO.

$$y = 1.0016867x - 1.7026344 \quad (3.3)$$

**Tabla 3. 7.** Comparación de datos de presión con ajuste.

Puntos (hPa)	Lp	Li	EA
1050	1050.3869	1050.3	-0.087
950	949.8745	949.9	-0.025
850	853.0081	853	0.012
730	734.2650	734.2	-0.055
650	648.8080	648.7	-0.098
550	558.7806	558.8	0.009



**Figura 3. 34.** Error de la medición de la presión con ajuste.

Una vez calibrados los sensores de la AWS de calibración se procede a instalar en cada localidad para adquirir valores de temperatura, humedad y presión ajustados a las normativas de la WMO, como se muestra en la Figura 3.35.



**Figura 3.35.** Instalación de AWS de calibración.

El procedimiento para la calibración de las AWS de Cumbayá, Conocoto y Calderón se realiza de acuerdo a los patrones de referencia de la AWS de calibración, la comparación directa de las 2 AWS se realiza con valores adquiridos durante 5 días que permiten modelar una ecuación de ajuste para las diferentes variables meteorológicas.

Para encontrar un modelo de ecuación que ajuste los datos de las AWS sometidas a condiciones naturales, es necesario que la información recopilada sea analizada cuidadosamente debido a que existen datos erróneos o irrelevantes, por tanto, es necesario aplicar el proceso de depuración para eliminar los datos no acordes a los demás.

En el Anexo III: Modelo de calibración de las AWS, se presenta el procedimiento que se siguió para calibrar la AWS de Calderón.

Las pruebas de verificación de las ecuaciones implementadas en el script de lectura de datos se realizaron durante 4 días para analizar la eficiencia de las lecturas luego de la calibración de la AWS de Calderón, sin embargo, se debe mencionar que el sistema no es 100% efectivo por la variabilidad de datos en un solo punto, teniendo como resultado la eficiencia del 92.9% en las lecturas de temperatura con un error de  $\pm 1^{\circ}\text{C}$ , 94.6% en las lecturas de humedad con un error de  $\pm 4\%$  y 100% en las lecturas de presión, como se muestra en la Tabla 3.8.

**Tabla 3. 8.** Eficiencia de calibración.

Errores	Día 1	Día 2	Día 3	Día 4	Total, de errores	Total, datos	Eficiencia (%)
Temperatura	158	37	146	59	400	5658	92,9
Humedad	66	43	125	71	305	5658	94,6
Presión	0	0	0	0	0	5658	100

### 3.5. PRUEBAS DE LA PREDICCIÓN DE LOS PARÁMETROS METEOROLÓGICOS

Las pruebas que se presentan en esta sección son las comparaciones y errores producidos entre los datos reales adquiridos y los pronósticos realizados por los 3 modelos de redes neuronales con el fin de analizar que configuración de red neuronal es más eficiente para predicciones futuras de las variables de temperatura y humedad relativa, cabe destacar que el entrenamiento de las redes se realizó con los datos adquiridos durante 75 días a los cuales se remuestreó cada hora, por tanto, la comparación se realizó con los datos del día 76 remuestreados cada hora y las predicciones realizadas por las redes neuronales.

#### 3.5.1. MODELO ARIMA

De acuerdo a las pruebas realizadas con el modelo de predicción ARIMA en el cual se fijan 3 parámetros ARIMA (p, d, q) con los que se define la configuración del modelo a utilizar y los mismos que deben cumplir con la condición de que los errores residuales o límite de significancia deben ser menores a 0.05. Estos valores se determinan a través de las funciones descritas en el Capítulo 1 como: FAC (Función de Autocorrelación) y FACP (Función de Autocorrelación Parcial), sin embargo, en este caso se utilizó la prueba ADF (*Augmented Dickey Fuller*) en la cual se asigna 0 al valor de diferenciación ya que

hipotéticamente la serie temporal es estacionaria y se comprueba la condición del nivel de significancia sea menor a 0.05, como se muestra en la Figura 3.36, para ello se incluye la función de `print(model_fit.summary())` en la sección de entrenamiento del modelo como se muestra en el Código 3.1.

```
# Entrenamiento de La red
model_fit = model.fit()
print(model_fit.summary())
```

**Código 3. 1.** Función de resumen de modelo configurado.

SARIMAX Results						
-----						
Dep. Variable:	y	No. Observations:	1776			
Model:	ARIMA(1, 0, 1)	Log Likelihood	-3332.517			
Date:	Thu, 09 Dec 2021	AIC	6673.035			
Time:	20:36:21	BIC	6694.963			
Sample:	0	HQIC	6681.135			
	- 1776					
Covariance Type:	opg					
-----						
	coef	std err	z	P> z	[0.025	0.975]
-----						
const	0.0225	0.150	0.150	0.881	-0.271	0.316
ar.L1	0.6469	0.013	48.979	0.000	0.621	0.673
ma.L1	0.4078	0.018	22.996	0.000	0.373	0.443
sigma2	2.4950	0.040	61.660	0.000	2.416	2.574
-----						
Ljung-Box (L1) (Q):		0.08	Jarque-Bera (JB):	3402.58		
Prob(Q):		0.78	Prob(JB):	0.00		
Heteroskedasticity (H):		0.94	Skew:	0.06		
Prob(H) (two-sided):		0.48	Kurtosis:	9.78		
-----						

Límites de significancia

**Figura 3.36.** Comprobación de límite de significancia.

Como se puede observar hay una constante que informa que la serie no es estacionaria y que necesita de un valor de diferenciación, es decir, se debe asignar un valor a d, por ende, el valor de d será igual a 1, el resultado se comprueba en la Figura 3.37.

SARIMAX Results						
-----						
Dep. Variable:	y	No. Observations:	1776			
Model:	ARIMA(1, 1, 1)	Log Likelihood	-3473.673			
Date:	Thu, 09 Dec 2021	AIC	6953.347			
Time:	20:48:18	BIC	6969.791			
Sample:	0	HQIC	6959.421			
	- 1776					
Covariance Type:	opg					
-----						
	coef	std err	z	P> z	[0.025	0.975]
-----						
ar.L1	-0.3234	0.056	-5.744	0.000	-0.434	-0.213
ma.L1	0.5611	0.053	10.600	0.000	0.457	0.665
sigma2	2.9331	0.047	62.423	0.000	2.841	3.025
-----						
Ljung-Box (L1) (Q):		0.99	Jarque-Bera (JB):	3635.76		
Prob(Q):		0.32	Prob(JB):	0.00		
Heteroskedasticity (H):		0.94	Skew:	-0.08		
Prob(H) (two-sided):		0.48	Kurtosis:	10.01		
-----						

Límites de significancia

**Figura 3.37.** Serie temporal estacionaria.

Ya definidos los parámetros con los cuales se comprueba la conversión de la serie temporal a estacionaria se procede a pronosticar los valores de temperatura y humedad de las AWS de Cumbayá, Conocoto y Calderón, con sus respectivos valores de eficiencia y errores obtenidos en el pronóstico.

En la Figura 3.38 se presentan las comparaciones mediante gráficos de las predicciones realizadas por el modelo ARIMA y los valores reales adquiridos por las AWS del parámetro de temperatura, como se puede observar esta variable atmosférica difiere debido a los factores propios de cada zona.

Los valores de las predicciones realizadas tienen cierta tendencia a los valores reales, sin embargo, las predicciones con mayor similitud o precisión de acuerdo a los gráficos son las realizadas por la AWS 4 - Calderón, mientras que las otras dos estaciones difieren mucho de sus datos en ciertas horas del día, sin embargo, la eficiencia de los pronósticos realizados se mide de acuerdo a MAPE (*Mean Absolute Percentage Error*, Error Porcentual Absoluto Medio).

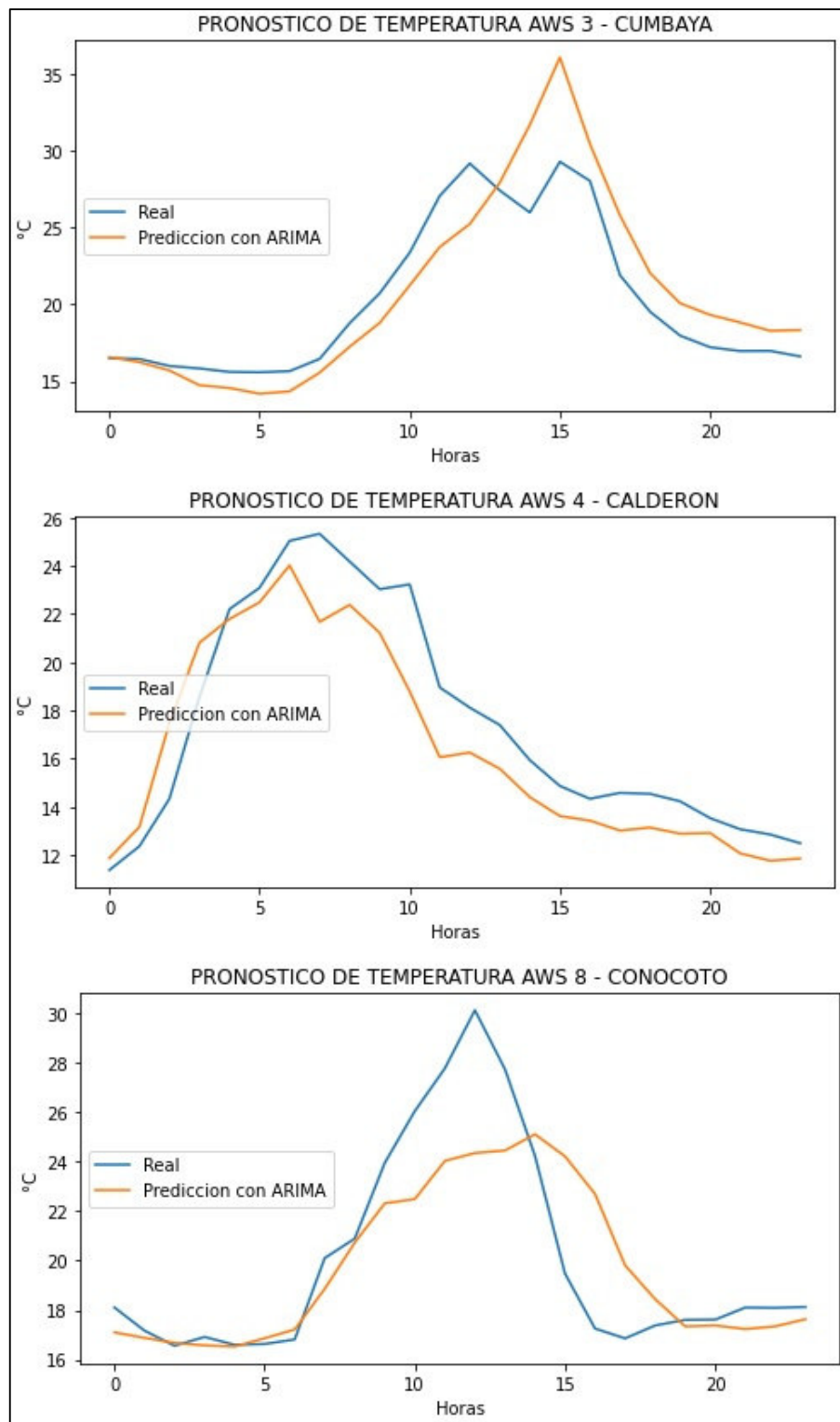
La Tabla 3.9 muestra la eficacia de cada una de las AWS al realizar pronósticos de sus valores, por tanto, se obtiene como resultado que la AWS 8 – Conocoto con un 92.36% supera en precisión de predicción en un aproximado de 2% a las AWS Cumbayá con 90.9% y Calderón con 90.4%, es decir, la estación de Conocoto tiene menor número de predicciones erradas.

El rango de error en la misma tabla representa el mínimo y máximo error respectivamente que se comete en las predicciones con el modelo ARIMA de la variable temperatura, donde la AWS 3 - Cumbayá presenta un error promedio de 2.1°C, la AWS 4 - Calderón de 1.6°C y la AWS 8 - Conocoto de 1.7°C.

**Tabla 3. 9.** Eficiencia de la predicción de temperatura con ARIMA en las 3 AWS.

<b>Nodo</b>	<b>AWS</b>	<b>MAPE (%)</b>	<b>Eficiencia AWS (%)</b>	<b>Error (°C)</b>
<b>3</b>	<b>Cumbayá</b>	9.64	90.36	0.06 – 6.8
<b>4</b>	<b>Calderón</b>	9.09	90.90	0.4 - 4
<b>8</b>	<b>Conocoto</b>	7.66	92.34	0.07 – 5.8





**Figura 3.38.** Predicciones de temperatura por ARIMA.

A continuación, la Figura 3.39 representa las comparaciones mediante gráficos de las predicciones realizadas y los valores reales adquiridos por las AWS del parámetro

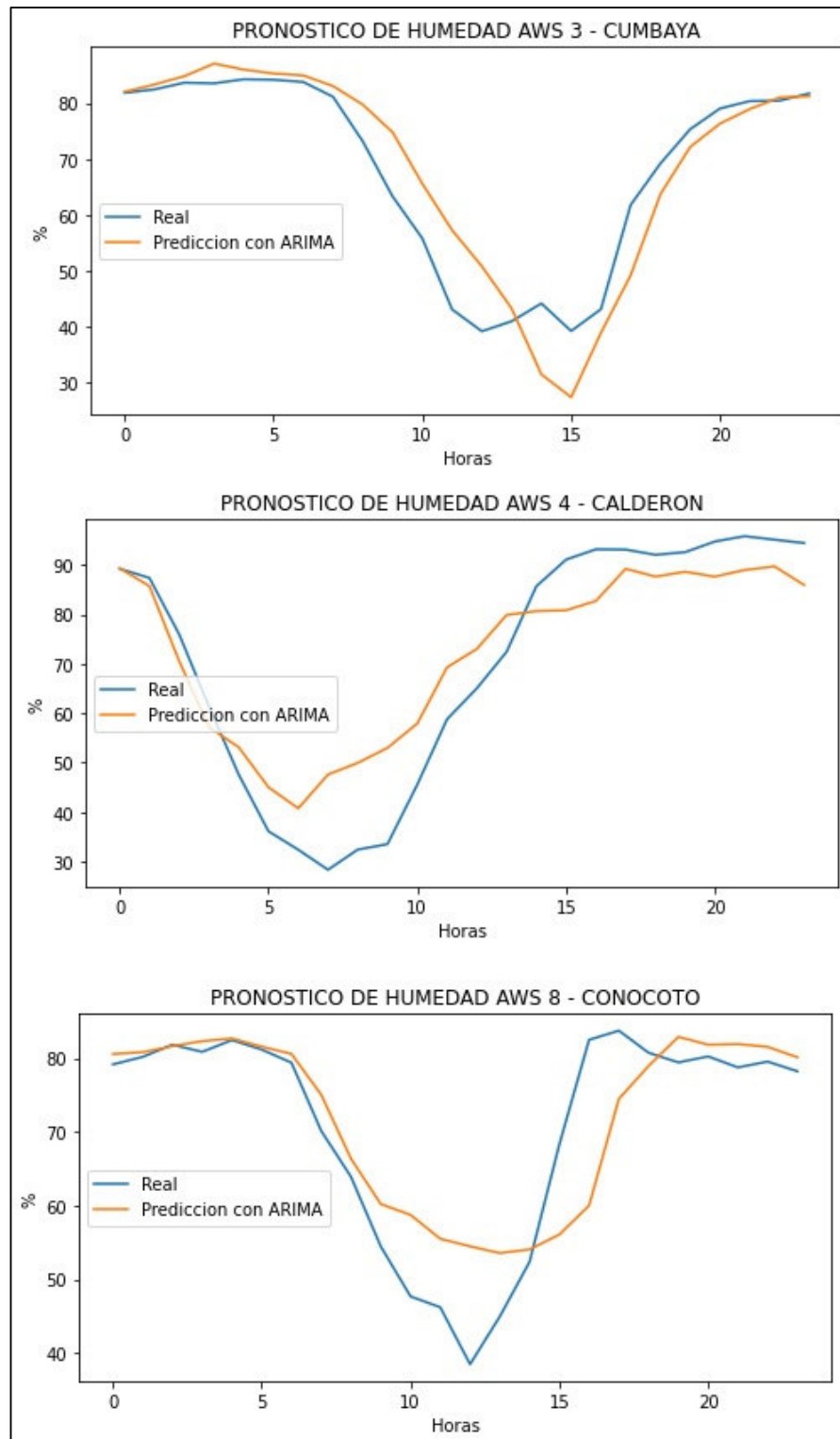
humedad relativa, como se puede observar los datos de predicción tratan de ajustarse a los datos reales, pero al igual que en las predicciones de temperatura los valores difieren en cierto rango de tiempo, siendo la AWS 3 – Cumbayá la que mayor precisión obtiene en la predicción de esta variable meteorológica.

Sin embargo, en la Tabla 3.10 se presenta la eficiencia de predicción de cada una de las AWS y se tiene como resultado que las AWS de mayor eficiencia de predicción son la AWS 8 - Conocoto con el 91.23% con un error promedio de 5.1% y la AWS 3 - Cumbayá con el 90.3% con un error promedio de 5.1% en comparación con la AWS 4 - Calderón con un 83.5% con un error promedio de 8.1%, aunque muestran que el error máximo cometido es elevado en las 3 estaciones.

**Tabla 3. 10.** Eficiencia de predicción de humedad con ARIMA de las 3 AWS.

<b>Nodo</b>	<b>AWS</b>	<b>MAPE (%)</b>	<b>Eficiencia AWS (%)</b>	<b>Error (%)</b>
<b>3</b>	<b>Cumbayá</b>	9.68	90.31	0.12 – 14.4
<b>4</b>	<b>Calderón</b>	16.53	83.47	0.09 – 19.5
<b>8</b>	<b>Conocoto</b>	8.73	91.27	0.2 – 22.5

Los errores en las predicciones del modelo ARIMA(p, d, q) pueden ser ocasionados por los mismos datos de la AWS, este modelo no posee memoria para almacenar valores previos y utilizarlos en pronósticos de estados futuros, por ende, los pronósticos dependen de la suma de las predicciones invertidas o errores de predicciones pasados de las últimas 24 horas de entrenamiento, por tanto, si el día 75 fue un día lluvioso y el 76 fue un día con mucho calor o viceversa, fenómenos de variación de parámetros meteorológicos que ocurren con frecuencia en Quito debido a los microclimas característicos de la ciudad, las predicciones van a ser erradas o existirán variaciones de acuerdo al rango de tiempo en que ocurrió el fenómeno natural de lluvia o calor, es decir, los errores se pueden dar por el cambio del tiempo atmosférico en instantes no determinados.



**Figura 3.39.** Predicciones de humedad en las 3 AWS con ARIMA.

### 3.5.2. LSTM SIMPLE

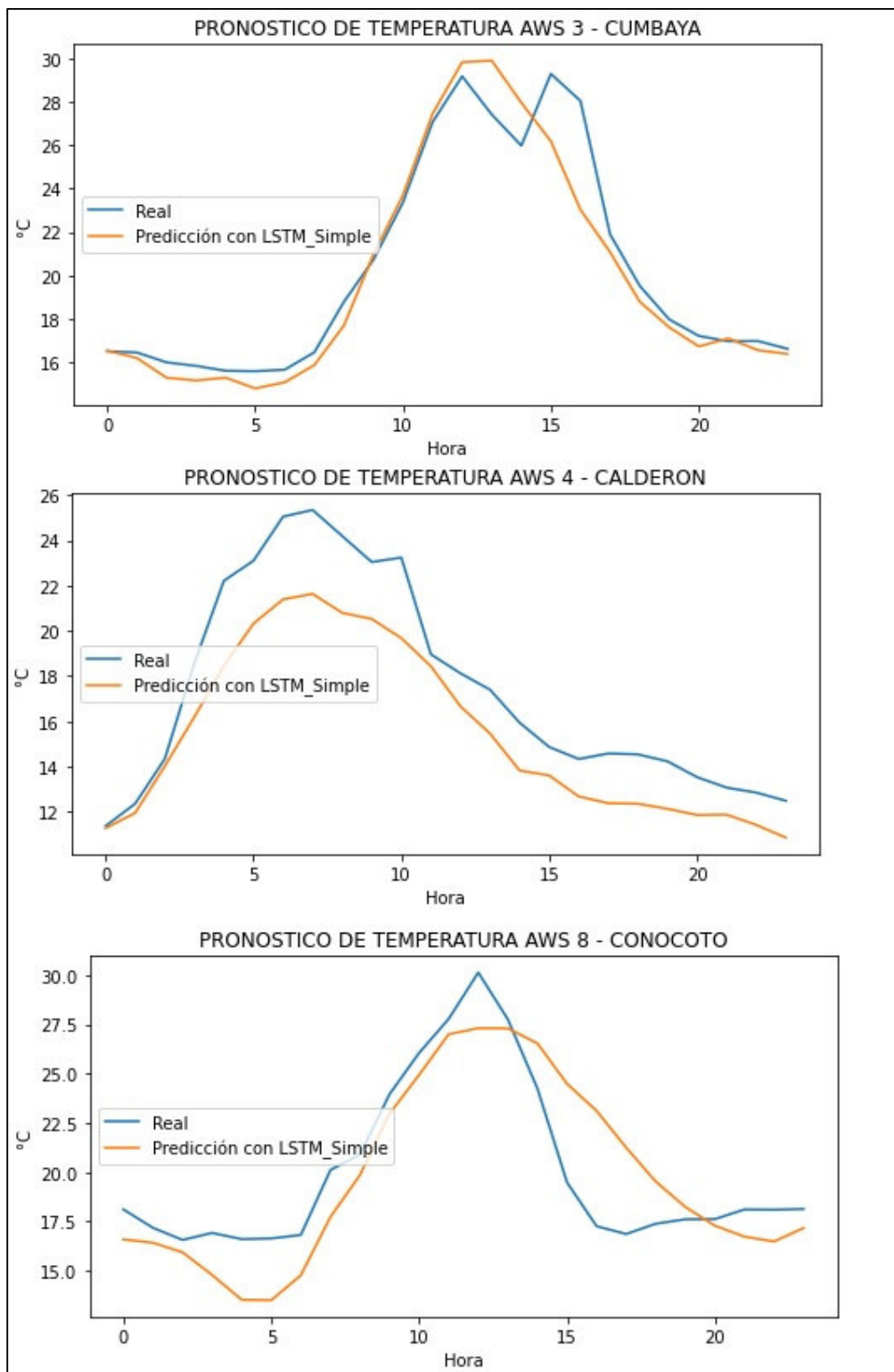
Para tratar de minimizar los errores cometidos por el modelo ARIMA (p, d, q) se implementa la red neuronal LSTM simple, es decir, tiene una capa oculta, para pronosticar valores futuros, esta configuración de red neuronal recurrente implementa memorias que le permiten recordar valores pasados y utilizarlos para pronosticar valores futuros.

La configuración del modelo LSTM se puede verificar gracias a la función `print(model.summary())` que proporciona información de las capas que conforman la red neuronal y las matrices de salida que proporcionan cada una, como se muestra en la Figura 3.40, donde, la capa oculta que se conforma de 300 neuronas entrega a la capa de salida una matriz de 300x1 hacia la capa de salida que está conformada de 150 neuronas y que entregan una matriz de salida de 24x1 con datos de predicciones realizadas para las 24 horas.

```
Model: "sequential"
Layer (type)                Output Shape              Param #
-----
lstm (LSTM)                  (None, 300)              362400
dense (Dense)                 (None, 150)              45150
dense_1 (Dense)              (None, 24)               3624
-----
Total params: 411,174
Trainable params: 411,174
Non-trainable params: 0
```

**Figura 3.40.** Información de configuración LSTM.

En la Figura 3.41 se presenta la comparación de los valores de temperatura real y pronosticada por la red neuronal LSTM simple, en el cual se obtiene como resultado visual la tendencia de los datos pronosticados a los valores reales del día 76, aunque como todo modelo posee errores, ya que las variables meteorológicas tienden a cambiar rápidamente.



**Figura 3.41.** Predicciones de temperatura por LSTM simple.

Los resultados obtenidos de los pronósticos de la red neuronal LSTM simple se presentan en la Tabla 3.11, en las cuales se incluyen los parámetros como RMSE que permiten conocer el error existente entre los datos reales y pronosticados por la red neuronal y el valor del MAPE que permite conocer la eficiencia de la predicción. Se puede afirmar que la AWS 3 - Cumbayá por sus resultados está en mejores condiciones de predecir valores futuros de temperatura con un valor de error promedio de 0.93°C en comparación a las AWS 4 - Calderón con un promedio de error de 1.99°C y la AWS 8 - Conocoto con el 1.98°C.

**Tabla 3. 11.** Resultados de predicciones de temperatura con LSTM simple en las 3 AWS.

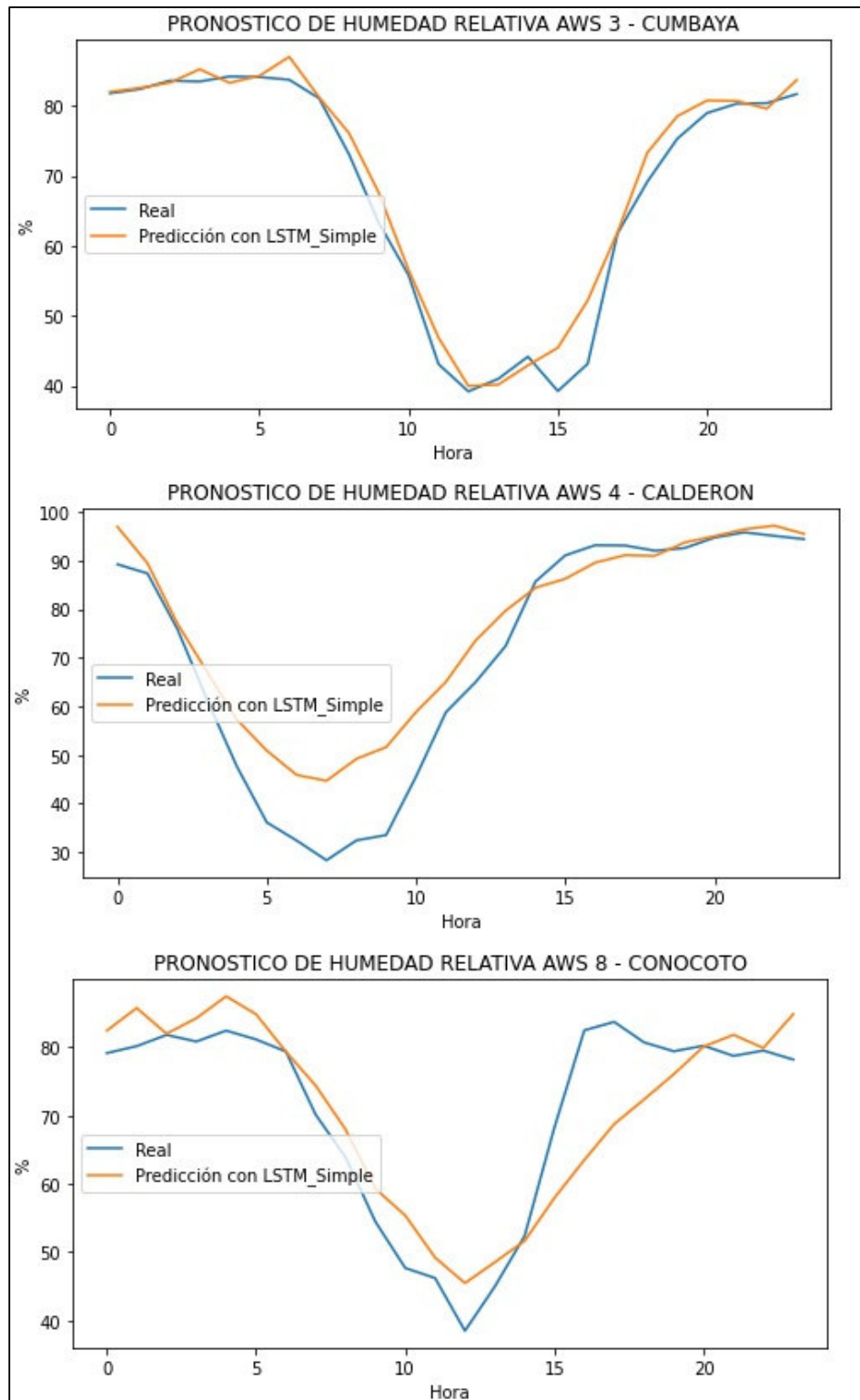
<b>Nodo</b>	<b>AWS</b>	<b>RMSE(°C)</b>	<b>MAPE (%)</b>	<b>Error (°C)</b>
<b>3</b>	<b>Cumbayá</b>	1.46	4.12	0.03 – 3.1
<b>4</b>	<b>Calderón</b>	2.26	10.97	0.09 – 3.8
<b>8</b>	<b>Conocoto</b>	2.44	10.52	0.3 – 5.8

A continuación, en la Figura 3.42 se presentan las comparaciones entre los valores reales y los pronósticos de humedad realizados por la red neuronal LSTM Simple, que a simple vista se deduce que la AWS 3 – Cumbayá va a presentar mejores resultados para las predicciones de la humedad relativa.

En la Tabla 3.12 se presentan los resultados obtenidos con sus respectivos errores para afirmar o rechazar lo que en perspectiva se dedujo, la AWS 3 – Cumbayá muestra mejores condiciones para pronosticar datos de humedad teniendo un error promedio de 2.1% en comparación a la AWS 4 - Calderón con un 5.7% y la AWS 8 - Conocoto con el 5.2%.

**Tabla 3. 12.** Resultados de predicciones de humedad con LSTM simple en las 3 AWS.

<b>Nodo</b>	<b>AWS</b>	<b>RMSE (%)</b>	<b>MAPE (%)</b>	<b>Error (%)</b>
<b>3</b>	<b>Cumbayá</b>	3	3.4	0.15 – 4
<b>4</b>	<b>Calderón</b>	7.5	10.17	0.04 – 15.6
<b>8</b>	<b>Conocoto</b>	6.18	7.33	0.14 – 15.6



**Figura 3.42.** Predicciones de humedad por LSTM Simple.

### 3.5.3. LSTM APILADA

La razón de implementar una red LSTM multicapa consiste en tener una mayor capacidad de la red neuronal para que ésta aprenda características más complejas de las entradas ingresadas y de esta manera el ajuste los datos sea de mejor manera, minimizando los errores obtenidos en la LSTM simple, sin embargo, apilar capas LSTM no siempre mejora el rendimiento de la red neuronal por un ajuste excesivo que puede llegar a tener [77].

La configuración de la red LSTM apilada se puede verificar gracias a la función `print(model.summary())` que presenta información de las capas que conforman la red neuronal y las matrices de entrada y salida que proporcionan cada una, como se muestra en la Figura 3.43 y que fue explicada en el Capítulo 2.

```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
lstm (LSTM)                  (None, 150)              91200
-----
repeat_vector (RepeatVector) (None, 24, 150)          0
-----
lstm_1 (LSTM)                (None, 24, 150)         180600
-----
time_distributed (TimeDistri (None, 24, 100)         15100
-----
time_distributed_1 (TimeDist (None, 24, 1)           101
-----
Total params: 287,001
Trainable params: 287,001
Non-trainable params: 0
```

**Figura 3.43.** Configuración LSTM apilada.

En la Figura 3.44 se presenta la comparación de los valores de temperatura real y pronosticados por la red neuronal LSTM apilada, en el cual se obtiene como resultado visual la tendencia de los datos pronosticados a los valores reales, sin embargo, como se puede visualizar los datos a pesar que la red neuronal tiene un mayor procesamiento de datos sigue teniendo errores en sus pronósticos para el día 76, por tanto, se deduce que no siempre se tiene un mejor rendimiento al aumentar capas intermedias en las redes neuronales.

Los resultados obtenidos de los pronósticos de temperatura de la red neuronal LSTM apilado se presentan en la Tabla 3.13 con los parámetros antes especificados como RMSE y MAPE para el análisis de cada AWS, donde se tiene como resultado que la AWS 4 – Caderón presenta mejoras en los pronósticos y condiciones para predecir valores futuros con el modelo configurado LSTM de doble capa, con un error promedio de 1.35°C y un



error de predicción general de 1.6°C en comparación a la AWS 3 - Cumbayá con un promedio de error de 1.77°C y la AWS 8 - Conocoto con el 2.29°C de error.

**Tabla 3. 13.** Resultados de las predicciones de temperatura con LSTM de doble capa

<b>Nodo</b>	<b>AWS</b>	<b>RMSE(°C)</b>	<b>MAPE (%)</b>	<b>Error (°C)</b>
<b>3</b>	<b>Cumbayá</b>	2.26	9.06	0.16 – 5
<b>4</b>	<b>Calderón</b>	1.6	7.76	0.08 – 2.9
<b>8</b>	<b>Conocoto</b>	2.75	12.92	0.4 – 5.4

El comportamiento de los valores de pronóstico de la variable humedad en la red neuronal LSTM de doble capa se muestran en la Figura 3.45, misma que denota un acercamiento mayor a los valores reales en comparación a la red neuronal LSTM simple, es decir, se espera que mediante esta configuración los valores pronosticados tengan menor error y sirvan de modelo para futuras predicciones.

Los resultados obtenidos de las predicciones de humedad en las AWS se presentan en la Tabla 3.14, donde se tiene como resultado que la AWS 3 – Cumbayá con un promedio de error de datos de 3.11%, presenta mejores pronósticos y menores errores en comparación a las AWS 4 - Calderón con un promedio de 9.75% y la AWS 8 - Conocoto con 7.69%.

**Tabla 3. 14.** Resultados de las predicciones de humedad con LSTM de doble capa.

<b>Nodo</b>	<b>AWS</b>	<b>RMSE (%)</b>	<b>MAPE (%)</b>	<b>Error (%)</b>
<b>3</b>	<b>Cumbayá</b>	5.16	6.19	0.13 – 16
<b>4</b>	<b>Calderón</b>	12.51	21.11	0.2 – 23.08
<b>8</b>	<b>Conocoto</b>	9.93	10.93	0.74 – 26.39

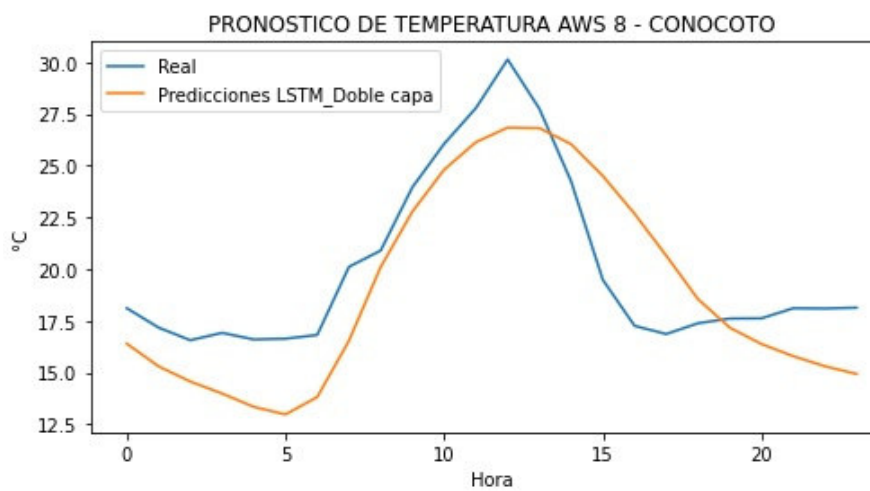
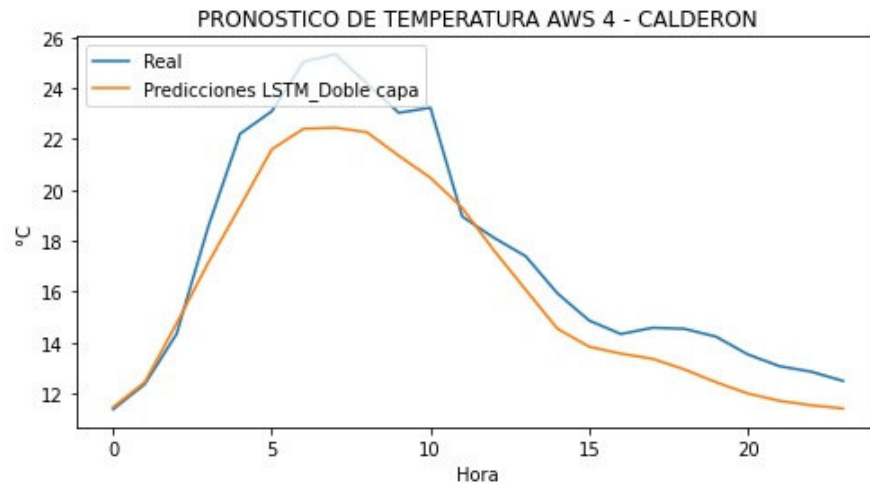
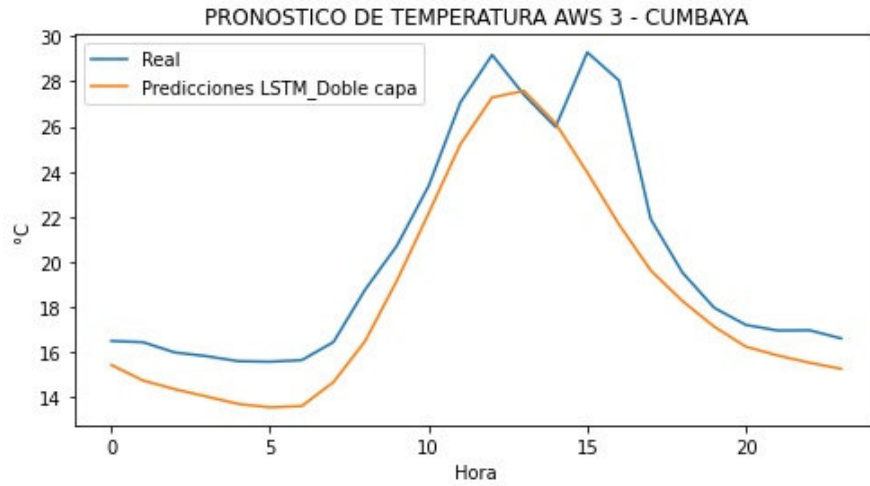
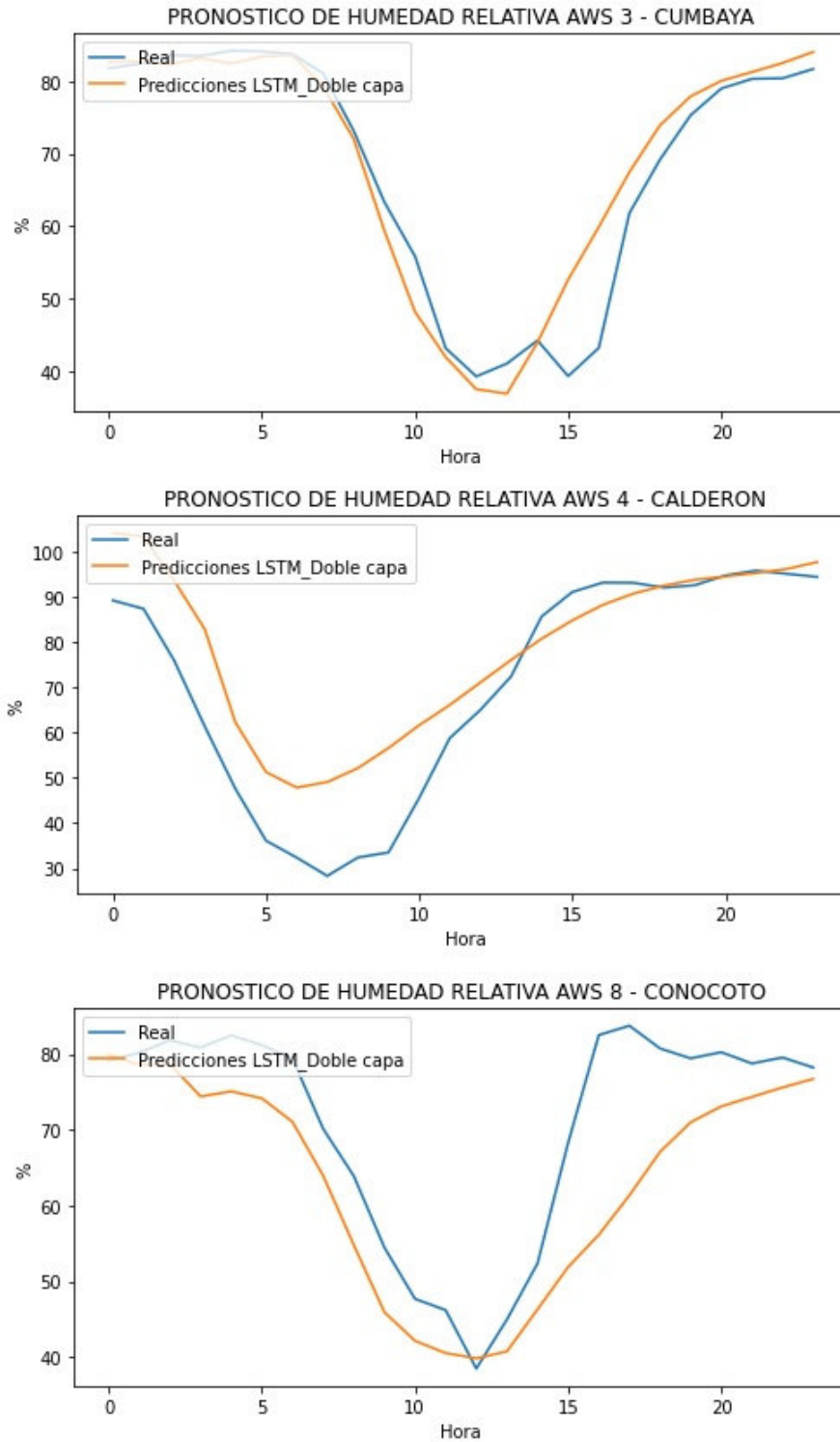


Figura 3.44. Predicciones de Temperatura por LSTM apilada.



**Figura 3.45.** Predicciones de humedad con LSTM apilado.

Las comparaciones de las 3 configuraciones de redes neuronales en la variable temperatura se realiza mediante la Tabla 3.15, en la cual se definen todos los parámetros

obtenidos durante las pruebas de predicción de la temperatura, por tanto, se tiene como resultado que la AWS 3 – Cumbayá presenta errores mínimos posibles con la configuración LSTM simple con un rango de error entre 0.03°C y 3.1°C. Se concluye que el modelo LSTM con una capa oculta presenta mejores prestaciones para realizar predicciones de temperatura.

**Tabla 3. 15.** Comparación de modelos de redes neuronales en la predicción de la temperatura.

AWS	Tipo	Errores		
		ARIMA	LSTM Simple	LSTM apilado
Cumbayá	Promedio (°C)	2.1	0.93	1.77
	MAPE (%)	9.64	4.12	9.06
	RMSE (°C)		1.46	2.26
Calderón	Promedio (°C)	1.6	1.99	1.35
	MAPE (%)	9.09	10.97	7.76
	RMSE (°C)		2.26	1.6
Conocoto	Promedio (°C)	1.7	1.98	2.29
	MAPE (%)	7.66	10.52	12.92
	RMSE (°C)		2.44	2.75

Mediante la Tabla 3.16, en la cual se definen todos los parámetros obtenidos durante las pruebas de predicción de la humedad relativa, se tiene como resultado que la AWS 3 – Cumbayá presenta errores mínimos posibles con la configuración LSTM simple con un rango de error entre 0.15% y 4%. Se concluye que el modelo LSTM con una capa oculta presenta mejores prestaciones para realizar predicciones de humedad relativa.

**Tabla 3. 16.** Comparación de modelos de redes neuronales en la predicción de la humedad relativa

AWS	Tipo	Errores		
		ARIMA	LSTM Simple	LSTM apilado
Cumbayá	Promedio (%)	5.1	2.1	3.11
	MAPE (%)	9.68	3.4	6.19
	RMSE (%)		3	5.16
Calderón	Promedio (%)	8.1	5.7	9.75
	MAPE (%)	16.53	10.17	21.11
	RMSE (%)		7.5	12.51
Conocoto	Promedio (%)	5.1	5.2	7.69
	MAPE (%)	8.73	7.33	10.93
	RMSE (%)		7.33	9.93

De forma general se concluye que de acuerdo a los resultados obtenidos de las 3 configuraciones realizadas para el caso de estudio se tiene una mejor eficiencia en la predicción de datos meteorológicos de temperatura y humedad relativa con la red neuronal LSTM simple, que de acuerdo a la información entregada por la AWS 3 – Cumbayá el intervalo de error respecto a la temperatura está entre 0.03°C y 3.1°C. Mientras que el intervalo de error respecto a la humedad relativa está entre 0.15% y 4%, hay varios factores que pueden influir en el resultado obtenido y que se diferencia del resto de estaciones como por ejemplo: la adquisición de datos donde la AWS 3 – Cumbayá ha mostrado mayor estabilidad que la AWS 4 – Calderón y la AWS 8 – Conocoto en donde se perdieron datos en varios periodos de tiempo, por tanto el análisis de estos datos no se realizó de forma continua con dicho nodo, además, otra de las ventajas del nodo Cumbayá es que en el sector es bastante estable las condiciones meteorológicas por ende las predicciones son mejores en este caso.

La mayor eficiencia de la red LSTM simple en la predicción de las variables meteorológicas en comparación a las 2 configuraciones probadas se debe a que el modelo ARIMA, si bien se puede aplicar en datos no estacionarios, esta no posee una memoria y los datos futuros dependen de la relación lineal existente en los datos pasados, por tanto, para predecir un valor futuro no se toma en cuenta si el evento ha sucedido alguna vez. Por otro lado, la red neuronal LSTM apilada presentó errores con una mínima diferencia que la LSTM simple, lo que puede deberse a que, en este caso de estudio, los datos no necesitan un alto procesamiento y al implementarse más capas se puede presentar el problema de sobreajuste en el cual la red memoriza las posibles salidas en vez de adecuar las entradas nuevas para el entrenamiento de la red.

### **3.6. PRESUPUESTO REFERENCIAL DE LOS EQUIPOS DEL SISTEMA**

A continuación, se presenta el costo de implementación del sistema que se conforma de 4 AWS: Cumbayá, Calderón, Conocoto y Calibración, en esta sección se presentan los costos de los dispositivos y materiales utilizados para la implementación de las AWS que se detallan en la Tabla 3.17 y el costo de calibración para cada variable meteorológica que difieren de acuerdo a los equipos utilizados y el tiempo que toma calibrar cada parámetro presentados en la Tabla 3.18.

**Tabla 3. 17.** Costos de materiales de implementación.

<b>Material</b>	<b>Cantidad</b>	<b>Precio unitario</b>	<b>Precio total</b>
Raspberry Pi 4B	4	\$ 110,00	\$ 440,00
Sensor DHT22	3	\$ 10,00	\$ 30,00
Sensor DHT21	1	\$ 10,00	\$ 10,00
Sensor BMP180	4	\$ 3,00	\$ 12,00
Placas de conexión	8	\$ 4,00	\$ 32,00
Casetas meteorológicas	4	\$ 20,00	\$ 80,00
Cable de extensión eléctrica	20	\$ 0,60	\$ 12,00
<b>Costo total de materiales</b>			<b>\$ 616,00</b>
<b>Costo por AWS</b>			<b>\$ 154, 00</b>

**Tabla 3. 18.** Costo de calibración.

<b>Parámetro</b>	<b>Costo</b>
Temperatura	\$ 63,91
Humedad relativa	\$ 205,45
Presión barométrica	\$ 139,10
<b>Costo de calibración</b>	<b>\$ 408,46</b>

Finalmente, la Tabla 3.19 presenta el costo total invertido en el proyecto para implementar un sistema prototipo de bajo costo para llevar a cabo pronósticos meteorológicos a corto plazo.

**Tabla 3. 19.** Costo total del proyecto.

<b>Detalle</b>	<b>Costo</b>
Materiales	\$ 616,00
Calibración	\$ 408,46
<b>Costo total del proyecto</b>	<b>\$ 1024,46</b>

## **4. CONCLUSIONES Y RECOMENDACIONES**

### **4.1. CONCLUSIONES**

La exposición de las AWS a las condiciones normales de cada sitio generó varias dificultades en la adquisición de datos, interrumpiendo la continuidad de los mismos, inconvenientes que se resolvieron mediante la reconfiguración y creación de nuevos scripts, actualización del software para mejorar el rendimiento y la instalación de nuevas librerías para el procesamiento de datos a través del software TeamViewer de acceso remoto, una herramienta esencial para el monitoreo de las AWS y que permite resolver problemas desde cualquier sitio y a través de cualquier dispositivo sin necesidad de transportarse hacia la estación.

El porcentaje de error en la calibración de los sensores se dio por la utilización del sensor DHT21 en la AWS de calibración, sensor que mostró un comportamiento distinto al DHT22; creemos que esto se debe a que éste entra en modo de suspensión, por tanto, el muestreo de las variables meteorológicas se realizó cada 2s y no cada 20s para evitar esta conducta y no tener una excesiva pérdida de datos durante el día, además, en los datos de validación de calibración no se realizó el proceso de depuración de datos, dando un margen de error total de alrededor 7% en los datos de temperatura y humedad relativa.

El almacenamiento de información en archivos de texto permitió a la AWS tomar decisiones en el caso de ocurrir eventos internos o externos, y a través del sistema de alarmas por correo electrónico dar aviso al administrador para tomar las medidas correctivas y de mantenimiento necesarias para el sistema e infraestructura y que el funcionamiento de este sea adecuado, continuo y genere los mínimos errores posibles.

En el modelo ARIMA se tienen 3 factores de configuración para realizar los pronósticos, dichos parámetros definen la estacionalidad de los datos de entrenamiento y la dependencia de auto regresión y errores de los eventos pasados para pronosticar eventos futuros.

Del estudio realizado se observa que la red LSTM simple presentó mejor eficiencia en realizar pronósticos de las series temporales de temperatura y humedad relativa en comparación al modelo ARIMA por la implementación de memorias, permitiendo aprender sobre los eventos relevantes del pasado y aplicar dicha relevancia en los pronósticos de eventos futuros.

La AWS de Cumbayá presentó un margen de error menor en las predicciones en comparación a las AWS de Calderón y Conocoto, con un rango de error en la temperatura

de entre 0.03°C y 3.1°C; y, humedad relativa entre 0.15% y 4%; con la configuración de red neuronal LSTM simple, debido a que presentó mayor estabilidad climática y mejores condiciones para la continuidad de adquisición de datos.

Las predicciones de LSTM apilado (2 capas) de acuerdo a los resultados obtenidos muestran similitud a los pronósticos de LSTM simple con una mínima diferencia en los parámetros de eficiencia presentados; sin embargo, a pesar que en ciertas aplicaciones presenta mejores resultados en este caso fue la excepción, debido al posible sobreajuste que puede resultar en la red, es decir, a pesar que la red muestra un buen rendimiento en el conjunto de datos de entrenamiento, en los datos de prueba decae.

El tiempo que toma realizar las predicciones en la red LSTM apilada es mucho mayor en comparación a las otras dos configuraciones realizadas, debido a la mayor cantidad de capas y procesamiento de datos que realiza internamente, siendo el modelo ARIMA el que menor tiempo toma en pronosticar, pero con una mayor cantidad de errores en el pronóstico.

Una desventaja que se presenta en el proyecto es que la instalación de las diferentes AWS se las realizó en propiedades privadas, por tanto, para el mantenimiento de las estaciones en el caso de errores físicos o de conexión, el administrador de las AWS debe ajustarse al tiempo libre de los propietarios de los sitios de instalación, esto genera excesiva pérdida de datos por el retraso en resolver los inconvenientes presentados.

La implementación de prototipos de estaciones meteorológicas automáticas de bajo costo fue posible gracias a los dispositivos electrónicos económicos disponibles en el mercado.

## **4.2. RECOMENDACIONES**

Se sugiere el mantenimiento de las garitas meteorológicas en un periodo de tiempo determinado (alrededor 4 meses), ya que al estar expuestas a las condiciones ambientales exteriores se deteriora tanto la infraestructura como las placas de conexión entre dispositivos debido a la corrosión.

Es recomendable instalar las garitas meteorológicas en zonas no expuestas a inundaciones y a una suficiente altura con el fin de no generar humedad en la caseta y que esta dañe los equipos electrónicos internos.

Se debe instalar la garita en zonas sin obstrucciones a su alrededor para recopilar información de las variables meteorológicas sin alterar sus condiciones térmicas.



La utilización de cable multipar flexible es importante para la conexión de las placas, con esto se evita la desconexión por rompimiento del cable, generando pérdida de datos.

Se recomienda analizar detenidamente el comportamiento de los sensores de calibración y ajustar el modelo a uno más adecuado que genere menos errores y por tanto mayor precisión en la adquisición de información meteorológica acorde a los límites de error especificados.

Es recomendable mejorar la caseta meteorológica con el fin de instalar un sistema de alimentación eléctrica renovable y un sistema de transmisión de datos independiente de la localidad en la que están instaladas, debido al corte de energía que en muchas ocasiones se pueden presentar.

Es necesario y recomendable la continuidad en la recolección de datos para el entrenamiento de la red neuronal, ya que el ajuste de pesos, umbrales y las predicciones dependen directamente de los ajustes que se da en este proceso en base a los datos disponibles.

## 5. REFERENCIAS BIBLIOGRÁFICAS

- [1] M. Murriaga, "Expansión urbana y demanda de transporte público de buses: caso de estudio de parroquias de Cumbayá, Tumbaco y Puembo," Quito, 2016.
- [2] L. León, de "*Propuesta de un modelo urbano macro frente al incremento demográfico en el sector de Calderón.*", Quito, 2015, pp. 30-31.
- [3] R. Yajaira, "Estudio de la factibilidad para la creación e implementación de una empresa Ecoturística en el Valle de Los Chillos," Quito, 2012, pp. 16-18.
- [4] Cazatormentas, "La web de los aficionados a la meteorología," 15 Agosto 2018. [En línea]. Available: <https://cazatormentas.net/meteorologia-y-climatologia-dos-ramas-de-la-ciencia-relacionadas-pero-no-iguales/>. [Último acceso: 02 Julio 2021].
- [5] M. Andrades y M. Carmen, "Fundamentos de Climatología," de *Material Didáctico: Agricultura y Alimentación*, La Rioja-España, Universidad de la Rioja Servicios Publicaciones, 2012, p. 7.
- [6] I. Fonseca, "Sociedad Andaluza de educación matemática Thales," [En línea]. Available: <https://thales.cica.es/rd/Recursos/rd99/ed99-0151-01/capitulos/cap2.htm>. [Último acceso: 02 Julio 2021].
- [7] D. Rivas, "El clima, caracteres, causas, clasificación, fenómenos y alteraciones climáticas," Lima-Perú, 2018.
- [8] E. Báez y G. Nicolalde, "Construcción de un prototipo inalámbrico de monitoreo meteorológico conformado por plataformas de hardware y software libre, asociado al análisis de sus datos para la predicción de temperatura mediante un método basado en Machine Learning.," Quito, 2018.
- [9] P. Ramírez, "Implementación de un sistema de medición, almacenamiento y transmisión de variables eléctricas para una estación meteorológica con mecanismos de ahorro de energía.," Quito, 2021.
- [10] J. Barros y A. Troncoso, "Atlas climatológico del Ecuador," Quito, 2010.
- [11] OMM, "Medición de Variables Meteorológicas," de *Guía de Instrumentos y Métodos de Observación Meteorológica.*, 2017, pp. 6-199.
- [12] R. Llugsí, A. Fontaine, P. Lupera y S. El Yacoubi, "Deep Learning to implement a Statistical Weather Forecast for the Andean City of Quito," *La Granja*, pp. 1-6, 2020.
- [13] A. Suárez, "Implementación de un sistema aéreo de medición y almacenamiento de parámetros meteorológicos Geo-referenciados para zonas pequeñas," Quito, 2017.
- [14] NCAR, "NCAR - UCAR," National Science Foundation, [En línea]. Available: <https://www.mmm.ucar.edu/weather-research-and-forecasting-model>. [Último acceso: 07 Julio 2021].

- [15] S. Serrano, J. C. Ruiz y F. Berbosa, "Heavy rainfall and temperature projections in a climate change scenario over Quito, Ecuador," *La Granja*, pp. 16-32, 2016.
- [16] "Ciencias de la Tierra," 29 Enero 2021. [En línea]. Available: <https://www.esearth.com/compact-weather-sensors-and-traditional-weather-sensors/>. [Último acceso: 12 Julio 2021].
- [17] F. Ureña, "ResearchGate," Marzo 2011. [En línea]. Available: [https://www.researchgate.net/publication/325919482\\_Utilizacion\\_de\\_estaciones\\_meteorologicas\\_automaticas\\_como\\_nueva\\_alternativa\\_para\\_el\\_registro\\_y\\_transmision\\_de\\_datos](https://www.researchgate.net/publication/325919482_Utilizacion_de_estaciones_meteorologicas_automaticas_como_nueva_alternativa_para_el_registro_y_transmision_de_datos). [Último acceso: 12 Julio 2021].
- [18] G. Novoa y G. Byron, "Desarrollo de una estación agro-meteorológica automática remota para el levantamiento de información climática en la cuenta del río Pisque.," Quito, 2018.
- [19] Omniblug, "Omniblug - Sensor de temperatura y humedad DHT11-DHT22," [En línea]. Available: [omniblug.com/sensor-temperatura-humedad-DHT11-DHT22.html](https://omniblug.com/sensor-temperatura-humedad-DHT11-DHT22.html). [Último acceso: 15 Julio 2021].
- [20] MicroJMP, "MicroJMP - Sensor de temperatura y humedad DHT21," [En línea]. Available: <https://www.microjpm.com/products/ad28820/>. [Último acceso: 15 Julio 2021].
- [21] N. MECHATRONICS, "Naylamp Mechatronics - Sensor de presión BMP280," [En línea]. Available: <https://naylampmechatronics.com/sensores-posicion-inerciales-gps/358-sensor-de-presion-bmp280.html>. [Último acceso: 15 Julio 2021].
- [22] J. Pastor, "XATAKA," 25 Julio 2019. [En línea]. Available: <https://www.xataka.com/ordenadores/raspberry-pi-4-analisis-caracteristicas-precio-especificaciones>. [Último acceso: 16 Julio 2021].
- [23] R. P. F. "Raspberry Pi Documentation," [En línea]. Available: <https://www.raspberrypi.org/documentation/computers/processors.html#bcm2711>. [Último acceso: 16 Julio 2021].
- [24] J. Loaiza, de "Diseño y construcción de un prototipo de control y monitoreo inalámbrico de un inventario de activos," Quito, 2020, pp. 20-24.
- [25] SOLECTRO, "Guía Raspberry Pi," [En línea]. Available: <https://solectroshop.com/es/content/60-5-pines-gpio-y-su-programacion>. [Último acceso: 16 Julio 2021].
- [26] D. G. I. "Sistemas operativos para Raspberry," 09 Octubre 2020. [En línea]. Available: <https://www.ionos.es/digitalguide/servidores/know-how/sistemas-operativos-para-raspberry-pi/>. [Último acceso: 19 Julio 2021].
- [27] D. G. I. "Servidor LAMP," 26 Mayo 2016. [En línea]. Available: <https://www.ionos.mx/digitalguide/servidores/know-how/servidor-lamp-la-solucion-para-webs-dinamicas/>. [Último acceso: 19 Julio 2021].

- [28] "Clinic Cloud," [En línea]. Available: <https://clinic-cloud.com/blog/servicios-en-la-nube-tipos-ejemplos/>. [Último acceso: 20 Julio 2021].
- [29] S. Vennam, "IBM," 18 Agosto 2020. [En línea]. Available: <https://www.ibm.com/mx-es/cloud/learn/cloud-computing/>. [Último acceso: 20 Julio 2021].
- [30] A. Deyimar, "Hostiger Tutoriales," 12 Agosto 2021. [En línea]. Available: <https://www.hostinger.es/tutoriales/que-es-cpanel/>. [Último acceso: 20 Julio 2021].
- [31] I. Ramírez, "Xataka," 16 Septiembre 2019. [En línea]. Available: <https://www.xataka.com/basics/filezilla-que-sirve-primeros-pasos-este-cliente-ftp/>. [Último acceso: 21 Julio 2021].
- [32] "Digital Guide IONOS," 4 Diciembre 2018. [En línea]. Available: <https://www.ionos.es/digitalguide/correo-electronico/cuestiones-tecnicas/sntp/>. [Último acceso: 21 Julio 2021].
- [33] D. Da Silva, "Zendesk," Web Content & SEO Associate, LATAM, 18 Marzo 2021. [En línea]. Available: <https://www.zendesk.com.mx/blog/acceso-remoto-que-es/>. [Último acceso: 22 Julio 2021].
- [34] Y. Fernández, "XATAKA," 23 Noviembre 2020. [En línea]. Available: <https://www.xataka.com/basics/programas-escritorio-remoto/>. [Último acceso: 22 Julio 2021].
- [35] A. Sánchez, "XENTIC," 3 Septiembre 2020. [En línea]. Available: <https://xentic.com.pe/top-10-progama-escritorio-remoto/>. [Último acceso: 22 Julio 2021].
- [36] R. García, "AZ adsl zone," 09 Agosto 2021. [En línea]. Available: <https://www.adslzone.net/como-se-hace/internet/servidor-vnc/>. [Último acceso: 22 Julio 2021].
- [37] G. Prácticas, "Guías prácticas.com," [En línea]. Available: <https://www.guiaspracticas.com/estaciones-meteorologicas/garita-meteorologica/>. [Último acceso: 26 Julio 2021].
- [38] S. Díaz, "Análisis CFD de diferentes diseños para una estación meteorológica de pequeñas dimensiones.," 2012.
- [39] Cometsystem, "Manualzz, the universal manuals library," [En línea]. Available: <https://manualzz.com/doc/56016230/comet-f8900-cometeo-professional-radiation-shield-for-wea...> [Último acceso: 27 Julio 2021].
- [40] IDEAM, "Guía para le emplazamiento de las estaciones meteorológicas convencionales.," Bogotá, 2021.
- [41] I. Maldonado, R. Ruiz y M. Fuentes, "Selección del sitio," de *Manual de instalación de estaciones meteorológicas automáticas*, Chillán, INIA, 2010, pp. 11-15.
- [42] X. Abajo y J. Veciana, "Construcción de una garita meteorológica artesanal.," *Revista Aficionados a la Meteorología.*, Catalunya, 2004.

- [43] J. Pelayo, "Meteorología para todos," 21 Noviembre 2011. [En línea]. Available: <http://ojaizmet.blogspot.com/2011/11/la-estacion-meteorologica-i.html>. [Último acceso: 29 Julio 2021].
- [44] D. G. IONOS, "Digital Guide IONOS," 10 Marzo 2020. [En línea]. Available: <https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-una-neural-network/>. [Último acceso: 2 Agosto 2021].
- [45] C. Nicholson, "Pathmind Inc.," 2020. [En línea]. Available: <https://wiki.pathmind.com/neural-network#element>. [Último acceso: 2 Agosto 2021].
- [46] C. Nicholson, "pathmind," [En línea]. Available: <https://wiki.pathmind.com/multilayer-perceptron>. [Último acceso: 3 Agosto 2021].
- [47] S. Mallick, "LearnOpenCV," 09 Octubre 2017. [En línea]. Available: <https://learnopencv.com/understanding-feedforward-neural-networks/>. [Último acceso: 3 Agosto 2021].
- [48] E. Freire y S. Silva, "Bootcamp AI," 14 Noviembre 2019. [En línea]. Available: <https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb>. [Último acceso: 4 Agosto 2021].
- [49] D. Calvo, "Diego Calvo," 07 Diciembre 2018. [En línea]. Available: <https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/>. [Último acceso: 4 Agosto 2021].
- [50] A. Rubiales, "Rubiales Alberto," 16 Octubre 2020. [En línea]. Available: <https://rubialesalberto.medium.com/explicaci%C3%B3n-funciones-de-activaci%C3%B3n-y-pr%C3%A1ctica-con-python-5807085c6ed3>. [Último acceso: 5 Agosto 2021].
- [51] X. Islas, "Crehana," 05 Febrero 2021. [En línea]. Available: <https://www.crehana.com/mx/blog/desarrollo-web/que-es-perceptron-algoritmo/#que-es-perceptron>. [Último acceso: 5 Agosto 2021].
- [52] I. C. Education, "IBM," 17 Agosto 2020. [En línea]. Available: <https://www.ibm.com/cloud/learn/neural-networks>. [Último acceso: 5 Agosto 2021].
- [53] G. Toro y L. Iván, "Evaluación de las Redes Neuronales Artificiales Perceptron Multicapa y Fuzzy-Artmap en la Clasificación de Imágenes Satelitales.," UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS, Bogotá - Colombia, 2012.
- [54] I. C. Education, "IBM," 14 Septiembre 2020. [En línea]. Available: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>. [Último acceso: 6 Agosto 2021].
- [55] C. Olah, "Blog de Colah," 27 Agosto 2015. [En línea]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Último acceso: 9 Agosto 2021].

- [56] M. Phi, "towards data science," 24 Septiembre 2018. [En línea]. Available: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>. [Último acceso: 9 Agosto 2021].
- [57] I. C. Education, "IBM," 14 Septiembre 2020. [En línea]. Available: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>. [Último acceso: 11 Agosto 2021].
- [58] MathWorks, "MathWorks," [En línea]. Available: <https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html>. [Último acceso: 11 Agosto 2021].
- [59] J. Bagnato, "Aprende Machine Learning en español," 29 Noviembre 2018. [En línea]. Available: <https://www.aprendemachinlearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/>. [Último acceso: 11 Agosto 2021].
- [60] P. Parada, "IEBS," 21 Mayo 2021. [En línea]. Available: <https://www.iebschool.com/blog/redes-neuronales-convolucionales/>. [Último acceso: 11 Agosto 2021].
- [61] J. Brownlee, "Machine Learning Mastery," 10 Diciembre 2020. [En línea]. Available: <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>. [Último acceso: 12 Agosto 2021].
- [62] J. Collantes, G. Colmenares, G. Orlandoni y F. Rivas, "SCIELO," Diciembre 2004. [En línea]. Available: [http://ve.scielo.org/scielo.php?script=sci\\_arttext&pid=S0254-07702004000300002](http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0254-07702004000300002). [Último acceso: 12 Agosto 2021].
- [63] IBM, "IBM," [En línea]. Available: <https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=data-series-transformations>. [Último acceso: 12 Agosto 2021].
- [64] S. De La Fuente, "Series Temporales, Modelo ARIMA, Metodología de Box-Jenkins," Universidad Autónoma de Madrid (UÁM), Madrid.
- [65] S. Prabhakaran, "Machine Learning +," 22 Agosto 2021. [En línea]. Available: <https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>. [Último acceso: 13 Agosto 2021].
- [66] A. Zayegh y N. Al Bassam, "Neural Network Principles and Applications," Clarivate Analytics, Muscat-Omán, 2018.
- [67] J. Zambrano, "Medium," 30 Marzo 2018. [En línea]. Available: <https://medium.com/@juanzambrano/aprendizaje-supervisado-o-no-supervisado-39ccf1fd6e7b>. [Último acceso: 16 Agosto 2021].
- [68] F. Sancho, "Fernando Sancho Caparrini," 14 Diciembre 2020. [En línea]. Available: <http://www.cs.us.es/~fsancho/?e=77>. [Último acceso: 16 Agosto 2021].

- [69] N. Acevedo, "Natalia Acevedo," 3 Junio 2020. [En línea]. Available: <https://nataliaacevedo.com/como-evaluar-modelos-de-aprendizaje-automatico-regresion/>. [Último acceso: 17 Agosto 2021].
- [70] J. Martínez, "IArtificial.net," 20 Septiembre 2020. [En línea]. Available: <https://www.iartificial.net/gradiente-descendiente-para-aprendizaje-automatico/>. [Último acceso: 18 Agosto 2021].
- [71] D. Johnson, "GURU99," 27 Agosto 2021. [En línea]. Available: <https://www.guru99.com/backpropogation-neural-network.html>. [Último acceso: 18 Agosto 2021].
- [72] Saurabh, "Backpropagation – Algorithm For Training A Neural Network," 2020.
- [73] L. Velasco, "Blog Luis Velasco," 26 Abril 2020. [En línea]. Available: <https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5>. [Último acceso: 19 Agosto 2021].
- [74] K. Borne, "SAS," [En línea]. Available: [https://www.sas.com/es\\_mx/insights/analytics/deep-learning.html](https://www.sas.com/es_mx/insights/analytics/deep-learning.html). [Último acceso: 2021].
- [75] ADAFRUIT, "Adafruit," [En línea]. Available: <https://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging/python-setup>. [Último acceso: 30 Agosto 2021].
- [76] ADAFRUIT, "Adafruit," [En línea]. Available: <https://learn.adafruit.com/using-the-bmp085-with-raspberry-pi/using-the-adafruit-bmp-python-library>. [Último acceso: 30 Agosto 2021].
- [77] R. De La Vega, "PHAROS," [En línea]. Available: <https://pharos.sh/resolucion-de-problemas-de-secuencias-con-lstm-en-keras/>. [Último acceso: 12 Diciembre 2021].
- [78] E. Freire y S. Silva, "Bootcamp AI," 14 Noviembre 2019. [En línea]. Available: <https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb>. [Último acceso: 2021].

## **ANEXOS**

ANEXO A. Funcionamiento del código de aplicación.

ANEXO B. Guía de utilización de la aplicación.

ANEXO C. Modelo de calibración de las AWS.

ANEXO D. Código de notificación de reinicio.

ANEXO E. Código de lectura de datos.

ANEXO F. Código de chequeo de datos.

ANEXO G. Código de almacenamiento de lecturas en base de datos.

ANEXO H. Código de interpolación de datos

ANEXO I. Código de almacenamiento de archivos en servidor FTP.

ANEXO J. Código de aplicación.



# ANEXO A

## Funcionamiento del código de aplicación

En primera instancia para el funcionamiento correcto de la aplicación es necesario instalar las librerías `xlrd` y `openpyxl`, librerías que permiten almacenar correctamente los datos en los archivos tipo Excel con extensión “.xlsx”, proceso que se realiza mediante el comando “`pip3 install xlrd`” y “`pip3 install openpyxl`” respectivamente.

Importamos las librerías para el desarrollo y funcionamiento de la aplicación que contiene la librería `tkinter`, así como variables tipo vectores que permiten guardar datos y mostrarlos por pantalla, como se muestra en el Código 6.1.

```
import mariadb
import subprocess
import time
import csv
import os
import subprocess
import pandas as pd
from tkinter import *
from tkinter import messagebox as ms
from tkinter import filedialog
from tkinter import ttk
from datetime import date
from datetime import datetime
from datetime import timedelta
from time import sleep

datos = []
event = []
fecha=[0.00,0.00,0.00]
hora=[0.00,0.00,0.00]
sensor=[0.00,0.00,0.00]
temp=[0.00,0.00,0.00]
hum=[0.00,0.00,0.00]
pres=[0.00,0.00,0.00]
rows2=[0.00,0.00,0.00,0.00,0.00,0.00,0.00]
```

**Código 5. 1.** Importación de librerías y variables definidas.

En el Capítulo 2 se presentó la creación de ventanas, botones y las configuraciones de cada widget implementado en la aplicación, en este anexo se presentan las funciones que los widgets están asociados. La pantalla principal se presenta en la Figura 5.1, esta pantalla se conforma de 2 tipos de botones: el botón “Salir” quien está asociado al código 5.2 e “Ingresar” asociado al Código 5.3.



Figura 5. 1. Pantalla principal.

Con un clic en el botón “Salir” la aplicación presenta un mensaje de aceptación o negación que se configura con la función *ms.askyesno()*, en el caso de aceptar la petición abandona el sistema con la función *quit()*.

```
#Salir
def salir():
    exit=ms.askyesno("Mensaje", "¿Desea salir del sistema?")
    if exit==True:
        quit()
```

Código 5. 2. Función salir

El botón “Ingresar” está asociada a la segunda ventana de la Figura 5.2, que se define mediante la función *Pantalla\_in()*, en la cual se visualizan los datos y se configuran múltiples herramientas, el Código 5.3 muestra un pequeño segmento de la función en la que se crea la ventana con el nombre “NODO 3 – CUMBAYÁ”.



**Figura 5. 2.** Pantalla de visualización y herramientas.

```

#PANTALLA DE OPCIONES Y FUNCIONES
def Pantalla_in():
    global ventana
    global data
    global fecha1
    global cursor
    global eventos
    global etiquetareinicio
    global dia
    global lec1, lec2, lec3
    global listopciones

    ms.showinfo("Mensaje", "Está ingresando a los datos del nodo Cumbayá")
    ventana = Toplevel(principal)
    ventana.title("NODO 3 - CUMBAYÁ")
    ventana.geometry("800x700")
    ventana.configure(bg="black")
    ventana.resizable(False, False)

```

**Código 5. 3.** Función de pantalla de visualización y funciones.

El Código 5.4 presenta las divisiones de la pantalla de datos de la Figura 5.2, estas divisiones se crean mediante la función *LabelFrame()* y se ajustan a la ventana mediante la función *pack()*.

```
#DIVISIONES DE LA PANTALLA
div1=LabelFrame(ventana, text="DATOS NODO CUMBAYÁ",
                font=("Arial",10))
div1.pack(fill="both", expand="yes")

div1_1=LabelFrame(div1, text="DATOS Y CONSULTAS",
                  font=("Arial",9))
div1_1.pack(fill="both", expand="yes", padx=5, pady=5)

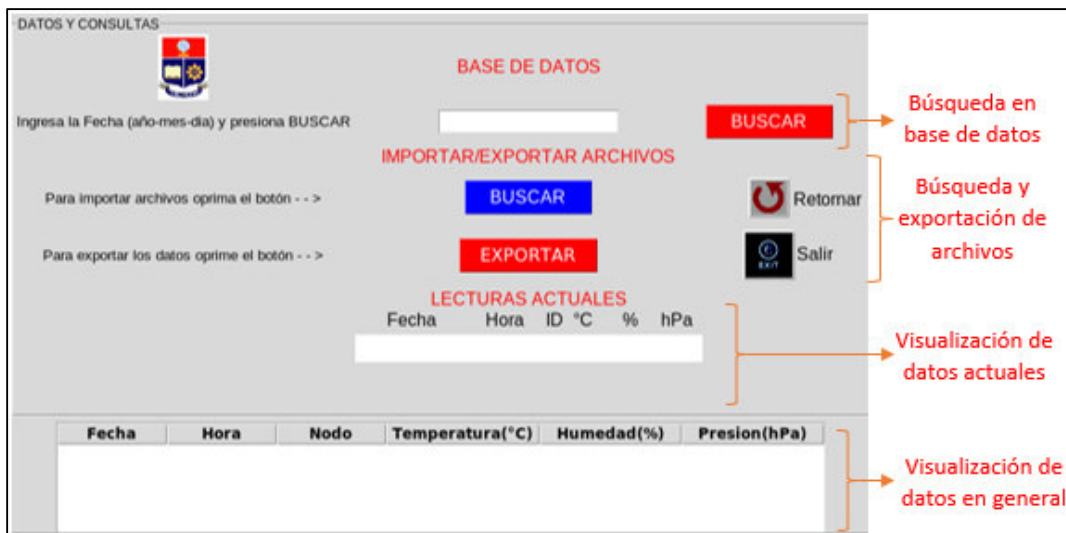
div2=LabelFrame(ventana, text="EVENTOS",font=("Arial",9))
div2.pack(fill="both", expand="yes")

div3=LabelFrame(ventana, text="PROGRAMACIÓN DE REINICIO",
                font=("Arial",9))
div3.pack(fill="both", expand="yes")

logo_dir = "/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/escudo.gif"
logo = PhotoImage(file=logo_dir)
logo_red = logo.subsample(4)
fondo1 = Label(div1_1, image=logo_red)
fondo1.grid(row=0, column=0)
```

**Código 5. 4.** Divisiones de pantalla.

La primera división se presenta En la Figura 5.3, cuenta con herramientas de búsqueda, exportación y visualización de datos y se compone de 4 subdivisiones.



**Figura 5. 3.** División 1.

La primera subdivisión se presenta como: “Búsqueda en base de datos”, el Código 5.5 realiza el proceso de búsqueda, se obtiene la fecha ingresada en el widget “Entry” y se verifica si hay un valor ingresado, se conecta a la base de datos mediante las credenciales con la función *mariadb.connect()*, definimos el cursor y realizamos un proceso de búsqueda con las funciones *SELECT*, *from* y *WHERE* de MySQL, ejecutamos el proceso con la función *execute()*, finalmente se obtiene todas la filas de la consulta realizada y devuelve una lista de tuplas mediante la función *cursor.fetchall()*. Los datos se visualizan en la sección “Visualización de datos en general” a través de la función *datosmdb()*, en el caso de no ingresar una fecha y se quiere hacer una búsqueda mostrará un mensaje de advertencia configurado con la función *ms.showwarning()*.

```
#Buscar datos en DB por fecha
def buscardb():
    fecha = fecha1.get()
    if len(fecha)>0:
        conexion = mariadb.connect(host="      ",
                                   user="      ",
                                   password="      ",
                                   database="      ")

        cursor = conexion.cursor()
        query = "SELECT Fecha, Hora, Sensor, Temperatura,Humedad,
                Presion from LecturaSensor WHERE Fecha LIKE '%" + fecha + "%'"
        cursor.execute(query)
        rows = cursor.fetchall()
        datosmdb(rows)
    else:
        ms.showwarning("Error", "Ingresa una fecha válida por favor")
```

**Código 5. 5.** Función de búsqueda en base de datos.

La visualización de datos se realiza en la ventana *treeview* definida como “data” de la sección “Visualización de datos en general”, el envío de datos a la ventana se realiza mediante la función *datosmdb()* del Código 5.6, los datos mostrados con anterioridad se borran mediante la función *data.delete(\*data.get\_children())* para insertar los nuevos datos mediante la función *data.insert()*.

```
#Datos en Treeview
def datosmdb(rows):
    global datos
    datos = rows
    data.delete(*data.get_children())
    for i in rows:
        data.insert('', 'end', values=i)
```

**Código 5. 6.** Visualización de datos.

La segunda sección “Búsqueda y exportación de archivos” define 4 botones: Buscar, Exportar, Retorno y Salir, donde, el botón Retorno se conecta a la base de datos y muestra la información en la ventana *data*, y el botón Salir ya se presentó con anterioridad.

El botón Buscar de esta sección permite elegir los archivos con extensión “.csv” o “.xlsx” y mostrarlos en la ventana *data*, la función se presenta en el Código 5.7, mediante la función *filedialog.askopenfilename()* abre un gestor de archivos para elegir el que se requiere mostrar, en el cual se define el directorio por defecto a ingresar, el título del gestor y los tipos de archivos a escoger. El archivo se asigna como lectura del formato elegido con la función *archivo=r"{}*”.*format()*, la separación de la extensión para su comparación se realiza con la función *fnopen,fileextension=os.path.splitext()*, los procesos que diferencian a los archivos con extensión “.csv” y “.xlsx” es la lectura con la librería pandas mediante las funciones *pd.read\_csv()* y *pd.read\_excel* respectivamente.

```
#Importar archivos csv o excel a ventana
def import_arch():
    datos.clear()
    fnopen = filedialog.askopenfilename(initialdir = "/home/pi/PROYECTO/LECTURAS/",
                                       title = "Gestor de archivos",
                                       filetypes = (("All Files", "*.*"),
                                                  ("csv Files", "*.csv"),
                                                  ("xlsx Files", "*.xlsx")))

    file_path = fnopen
    try:
        archivo = r"{}”.format(file_path)
        fnopen,fileextension=os.path.splitext(archivo)
        if fileextension == ".csv":
            df = pd.read_csv(archivo)
            rows = df.to_numpy().tolist()
            for i in rows:
                datos.append(i)
            datosmdb(datos)
        elif fileextension == ".xlsx":
            df = pd.read_excel(archivo)
            rows = df.to_numpy().tolist()
            for i in rows:
                datos.append(i)
            datosmdb(datos)
```

**Código 5. 7.** Importación de archivos a ventana data.

El botón “Exportar” permite leer los datos de la ventana *data* y exportarlos a un archivo Excel de extensión “.xlsx”, el Código 5.8 muestra dicho proceso. Se define una nueva variable que guardará los datos como una lista con la función *list()*, se verifica si hay datos existentes en la ventana *data* en el caso de no existir presenta un mensaje de advertencia, por contrario si hay datos se presenta un mensaje de confirmación o negación para la

exportación de los datos, si se niega el proceso un mensaje de información se despliega configurado con la función *ms.info()*, al afirmar el proceso se abre un gestor de almacenamiento configurado con la función *filedialog.asksavefilename()* que se define el directorio por default a ingresar, el titulo del gestor y el tipo de la extensión “.xlsx” a guardar, los datos se almacenan en la variable *new*, se almacenan en un DataFrame pandas y se guarda en Excel con la función *to\_excel()* si el proceso se realiza con éxito muestra un mensaje de información confirmando el proceso de almacenamiento correcto de datos.

```
#Exportar datos de ventana a un archivo tipo EXCEL
def export_datos():
    new=list()
    if len(datos) < 1:
        ms.showwarning("Error",
            "No existen datos para exportar")
        return False
    export=ms.askyesno("Mensaje",
        "¿Desea exportar los datos seleccionados a excel?")
    if export==True:
        flsave = filedialog.asksaveasfilename(initialdir=
            "/home/pi/PROYECTO/LECTURAS/ARCHIVOS EXPORTADOS/",
            title="EXPORTAR A EXCEL",
            filetypes=(("xlsx File", "*.xlsx"),
                ("All Files", "*.*")))
        for i in datos:
            new.append(i)
        df=pd.DataFrame(new)
        df.to_excel(flsave, index=None, header=False)
        ms.showinfo("Mensaje", "Datos exportados con éxito, revisar carpeta destino")
        retornar()
    else:
        ms.showinfo("Mensaje", "Operación cancelada")
        retornar()
```

**Código 5. 8.** Exportación de datos a Excel.

En la tercera subsección “Visualización de datos actuales” se lee los últimos 3 datos del archivo actual y estos se actualizan cada minuto, el Código 5.9 muestra el proceso de visualización de datos actuales, la lectura de datos del archivo se realiza mediante la función *pd.read\_csv()*, se guarda estos en un *DataFrame* pandas y se define un rango de 0 a 3 para definir los 3 últimos valores de cada variable mediante la función *iloc()* y se almacenan en la variable *rows2*, valores que se guardan los vectores definidos al inicio, mediante la función *config()* se muestran los valores almacenados en *rows2* en los *Label* definidos como *lec1*, *lec2* y *lec3*, estos últimos se actualizan cada minuto mediante la función *after()* que llama a la función *lecturas()*, un error se produce en el caso de que la aplicación no encuentre el archivo actual o no se ha creado por falla de la AWS presentando el mensaje de “Ups! : Error al encontrar datos actuales”.

```

def lecturas():
    try:
        fecha_act = datetime.now()
        time_act = fecha_act.strftime("%H:%M:%S")
        today = date.today()
        idSensor = 3
        nombreFile = "/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor, today)
        datos = pd.read_csv(nombreFile, names=cabecera)
        df=pd.DataFrame(datos)
        for i in range (0,3):
            fecha[i]=df.iloc[-i-1, 0]
            hora[i]=df.iloc[-i-1, 1]
            sensor[i]=df.iloc[-i-1, 2]
            temp[i]=df.iloc[-i-1, 3]
            hum[i]=df.iloc[-i-1, 4]
            pres[i]=df.iloc[-i-1, 5]
            rows2[i]=[fecha[i], hora[i], sensor[i], temp[i], hum[i], pres[i]]
        lec1.config(text=rows2[0], bg="white", fg="midnight blue", font=("Arial", 14))
        lec2.config(text=rows2[1], fg="midnight blue", font=("Arial", 14))
        lec3.config(text=rows2[2], bg="white", fg="midnight blue", font=("Arial", 14))
        lec1.after(60000,lecturas)
        lec2.after(60000,lecturas)
        lec3.after(60000,lecturas)
    except FileNotFoundError as error:
        lec1.config(text="Ups! : Error al encontrar datos actuales", bg="white",
                    fg="midnight blue", font=("Arial", 14))
        lec1.after(60000, lecturas)

```

**Código 5. 9.** Visualización de datos actuales.

La segunda división definida como “EVENTOS” de la Figura 5.4 muestra los eventos ocurridos en la AWS almacenados en la base de datos, el Código 5.10 muestra el proceso se realiza mediante la conexión a la base de datos y la búsqueda con las funciones *SELECT* y *from* de MySQL, esta información se presenta de forma descendente con la función *ORDER BY* y *DESC* de MySQL, datos que se guardan en la variable *rows1* y se muestran en la ventana definida como eventos a través de la función *datosevent()* del Código 5.11.

Fecha	Hora	Sensor	Evento

**Figura 5. 4.** Eventos.

```

query = "SELECT Fecha, Hora, Sensor, Evento from Eventos ORDER BY Fecha DESC, Hora DESC"
cursor.execute(query)
rows1 = cursor.fetchall()
datosevent(rows1)

```

**Código 5. 10.** Búsqueda de eventos.



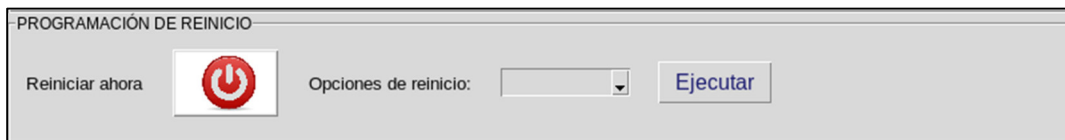
```

#Datos eventos
def datosevent(rows1):
    global event
    event = rows1
    eventos.delete(*eventos.get_children())
    for i in rows1:
        eventos.insert('', 'end', values=i)

```

**Código 5. 11.** Visualización de eventos.

La tercera sección definida como “PROGRAMACIÓN DE REINICIO” se muestra en la Figura 5.5 quien contiene dos opciones de reinicio como: reinicio inmediato y reinicio programado.



**Figura 5. 5.** Programación de reinicio.

El botón “Reiniciar ahora” permite reiniciar la AWS de forma inmediata, este se configura de acuerdo a la función *reinicio()* del Código 5.12, presenta un mensaje de confirmación o negación para realizar el proceso, al aceptar la AWS se reinicia mediante la función *subprocess.call(“shutdown -r now”, Shell=True)*, si se niega muestra un mensaje de información sobre la cancelación del proceso.

```

#Reinicio inmediato
def reinicio():
    reboot=ms.askyesno("Mensaje", "¿Desea reiniciar el Nodo Cumbayá?")
    if reboot==True:
        subprocess.call("shutdown -r now", shell=True)
    else:
        ms.showinfo("Mensaje", "Reinicio cancelado")

```

**Código 5. 12.** Función de reinicio.

El reinicio programado se realiza mediante la función *reinicioprg()*, las opciones definidas en la lista denominada *listopciones* se programan de acuerdo al widget *ttk.Combobox()* presentado en el Capítulo 2, las opciones y la programación de reinicio se presenta en el Código 5.13, la opción elegida por el usuario mediante al hacer clic en el botón “Ejecutar” se obtiene mediante la función *listopciones.get()* a quienes se realizan las comparaciones para definir el número de días para que la AWS se reinicie, valor que se guarda en la

variable n, posterior se llama a la función *guardar(n)* que lee el número de días y almacena en la *Variable\_reincio.txt* la fecha y hora en la cual la AWS se reiniciará.

```
#Reinicio programado
def reinicioprg():
    if listopciones.get() == "Sin reinicio":
        variable=open("/home/pi/PROYECTO/CODIGOS/APP/Variable_reincio.txt", "w")
        variable.close()
        ms.showinfo("Mensaje", "Nodo sin reinicio programado")
    elif listopciones.get() == "1 día":
        n=1
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
    elif listopciones.get() == "2 días":
        n=2
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
    elif listopciones.get() == "3 días":
        n=3
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
    elif listopciones.get() == "4 días":
        n=4
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
    elif listopciones.get() == "5 días":
        n=5
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
```

**Código 5. 13.** Opciones de reinicio.

```
#FUNCION DE ALMACENAMIENTO DE FECHA DE REINICIO
def guardar(n):
    now=datetime.now()
    reiniciodia=now+timedelta(days=n)
    variable=open("/home/pi/PROYECTO/CODIGOS/APP/Variable_reincio.txt", "w")
    variable.write(str(reiniciodia))
    variable.close()
```

**Código 5. 14.** Almacenamiento de fecha y hora de reinicio.

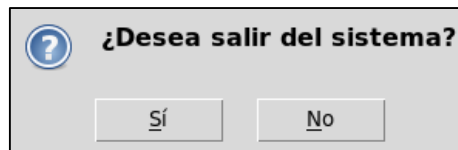
## ANEXO B

### Guía de utilización de la aplicación APPCLIMA

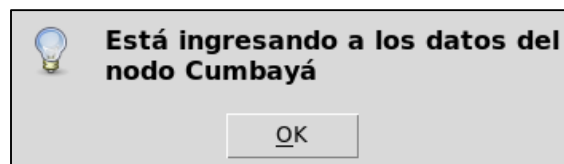
- **Presentación pantalla de presentación.**



Con un clic en el botón “Salir” se despliega un mensaje para confirmar o negar la expulsión del sistema, al negar se mantiene en la pantalla principal hasta realizar alguna otra acción.

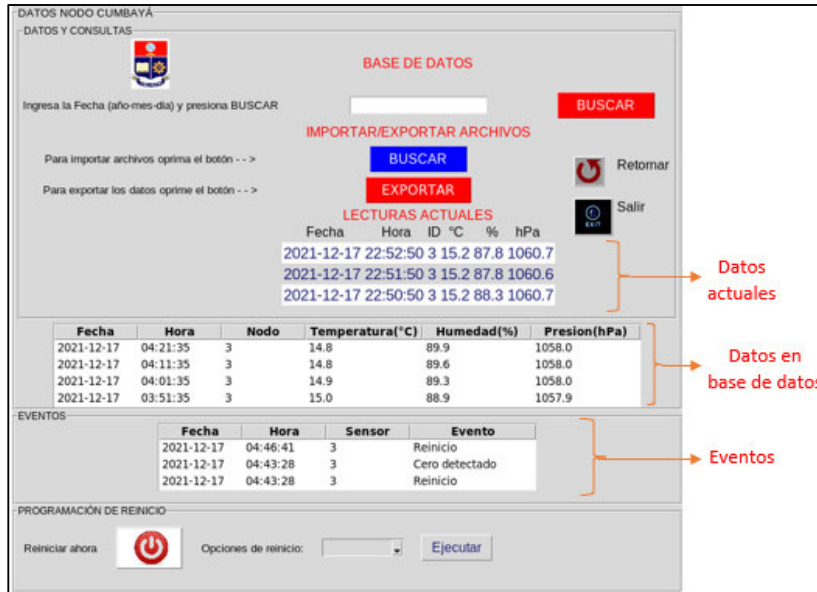


Con un clic en el botón “Ingresar” la aplicación despliega un mensaje de información de ingreso al sistema de búsqueda de datos, visualización de datos y programación de reinicio, presionar OK.



- **Presentación pantalla de datos y programación.**

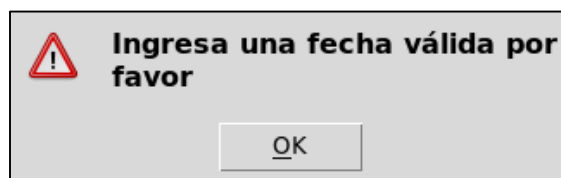
La ventana inicia y se presenta los datos actuales, los datos almacenados en la base de datos y los eventos ocurridos en la AWS.



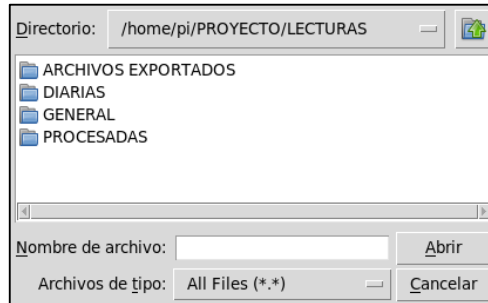
Para consultar los datos de una determinada fecha en la base de datos ingresar la fecha requerida en el espacio en blanco de acuerdo al formato “año-mes-día”, por ejemplo: 2021-11-19 y dar clic en el botón “BUSCAR” (rojo).



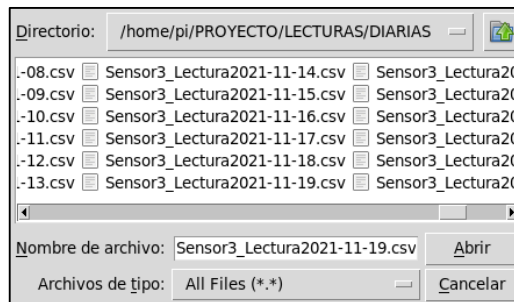
Si no ingresa una fecha y presiona el botón “BUSCAR” la aplicación despliega un mensaje de advertencia.



Para importar un archivo de extensión “.csv” o “.xlsx” a la ventana de datos, hacer clic en el botón “BUSCAR” (azul), se despliega un gestor de archivos en la carpeta “LECTURAS” por default ya que aquí se almacenan los archivos de datos.



Elegir el archivo del directorio requerido y dar clic en “Abrir”.



Los datos se muestran en la ventana mientras que los actuales se muestran continuamente y se actualizan.

**DATOS NODO CUMBAYÁ**  
DATOS Y CONSULTAS

Ingresar la Fecha (año-mes-día) y presiona **BUSCAR**

Para importar archivos oprima el botón - -> **BUSCAR**  
Para exportar los datos oprima el botón - -> **EXPORTAR**

**LECTURAS ACTUALES**

Fecha	Hora	ID	°C	%	hPa
2021-12-17	22:55:50	3	15.2	87.8	1060.6
2021-12-17	22:54:50	3	15.2	87.6	1060.6
2021-12-17	22:53:50	3	15.2	87.8	1060.7

**EVENTOS**

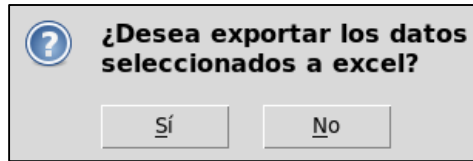
Fecha	Hora	Sensor	Evento
2021-12-17	04:46:41	3	Reinicio
2021-12-17	04:43:28	3	Cero detectado
2021-12-17	04:43:28	3	Reinicio

**PROGRAMACIÓN DE REINICIO**

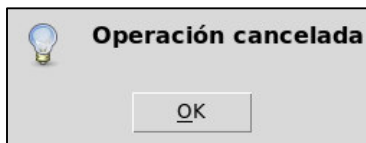
Reiniciar ahora Opciones de reinicio:  **Ejecutar**

Datos del archivo importado

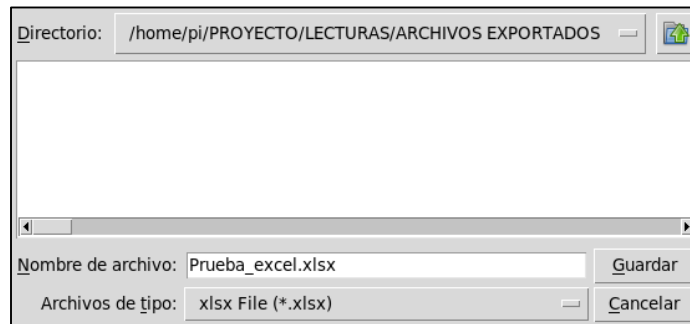
Para exportar los datos de la ventana hacer clic en el botón “EXPORTAR”, se despliega un mensaje de aceptación o negación.



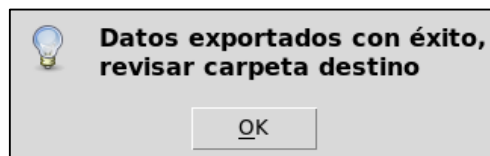
Al negar el proceso se despliega un mensaje de información en el cual informa que la operación ha sido cancelada.



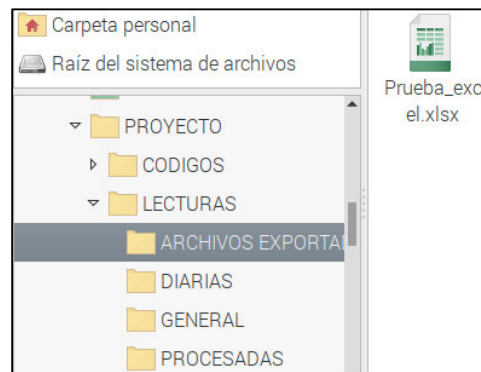
Por contrario al aceptar el proceso, se despliega una ventana de almacenamiento dentro de la carpeta "ARCHIVOS EXPORTADOS" por default aquí se guardarán los archivos tipo Excel, ingresar el nombre del archivo con el que se requiere guardar y la extensión ".xlsx", por ejemplo: "Prueba\_excel.xlsx" y clic en "Guardar".



Una vez realizado el proceso la aplicación informará que el proceso ha sido exitoso.



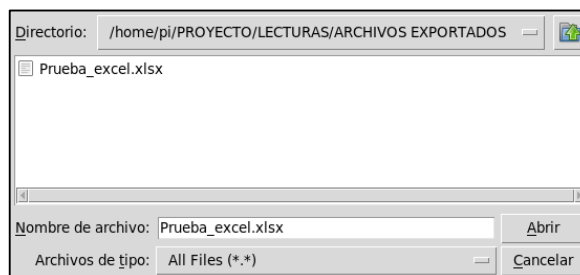
Se puede revisar la carpeta que contiene el archivo tipo Excel creado.



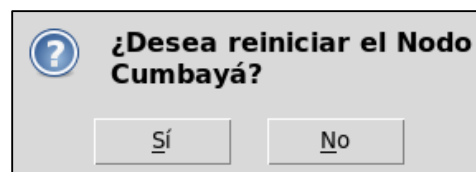
La verificación de los datos almacenados en el archivo Excel consultados de la base de datos en la fecha 2021-11-19 en una computadora externa se muestra a continuación.

	A	B	C	D	E	F
1	2021-11-19 00:06:40	3		14,9	86,9	1060,8
2	2021-11-19 00:16:40	3		14,8	86,6	1060,7
3	2021-11-19 00:26:40	3		14,9	86,4	1060,6
4	2021-11-19 00:36:40	3		15	86,2	1060,4
5	2021-11-19 00:46:40	3		14,8	86,3	1060,3
6	2021-11-19 00:56:40	3		14,8	86,3	1060,1
7	2021-11-19 01:06:40	3		14,8	86,5	1060
8	2021-11-19 01:16:40	3		14,7	87,2	1059,9
9	2021-11-19 01:26:40	3		14,6	87,5	1059,9
10	2021-11-19 01:36:41	3		14,6	87,1	1059,9
11	2021-11-19 01:46:41	3		14,4	87,3	1059,9
12	2021-11-19 01:56:41	3		14,4	87,4	1059,9
13	2021-11-19 02:06:41	3		14,3	87,2	1059,8
14	2021-11-19 02:16:41	3		14,1	88,2	1059,6

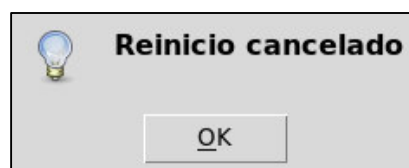
O se puede abrir en la misma aplicación mediante el gestor de archivos, seleccionar el archivo y clic en “Abrir”.



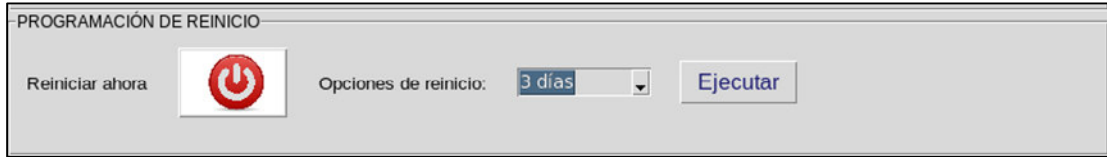
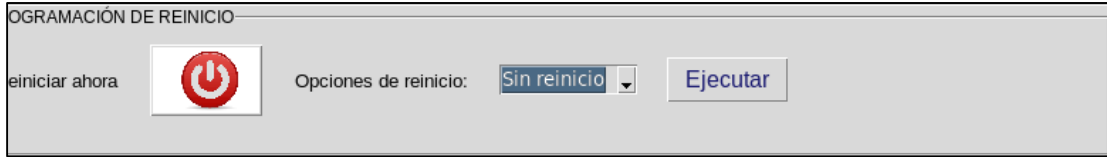
Para reiniciar la AWS dar clic en el botón “Reiniciar ahora”, se despliega un mensaje para aceptar o negar el proceso, si acepta la AWS se reinicia inmediatamente.



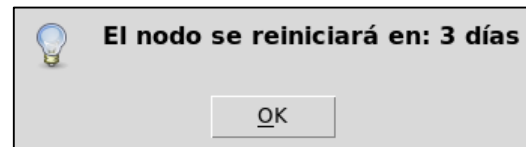
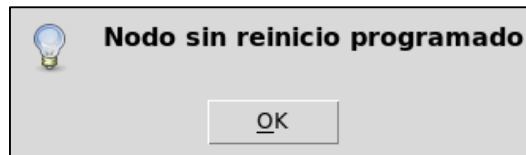
Si niega el proceso un mensaje de cancelación se despliega y regresa a la ventana.



Para programar que la AWS se reinicie en un día determinado se tiene opciones como: “Sin reinicio”, “1 día”, “2 días”, “3 días”, “4 días” y “5 días”, escoger una de estas opciones.



Hacer clic en el botón “Ejecutar”, la aplicación muestra un mensaje de información de la elección realizada y guarda esta información para reiniciar el dispositivo según lo configurado.



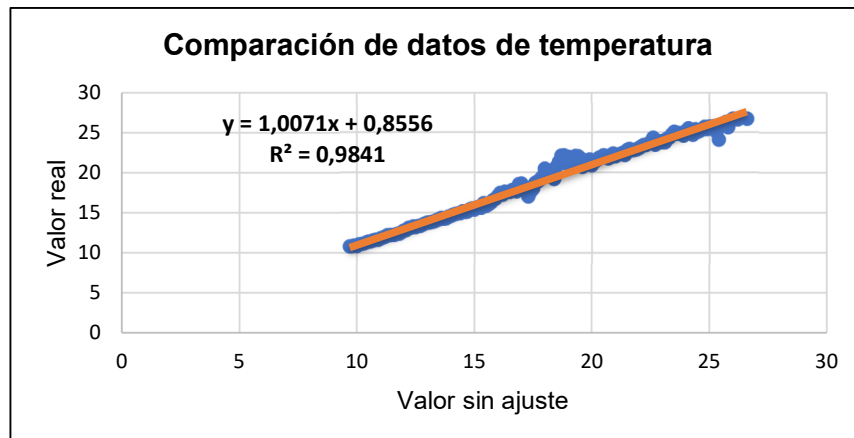
Esta ventana también consta de un botón “Salir” para expulsar al usuario del sistema y muestra las mismas opciones antes mencionadas para este botón.



## ANEXO C

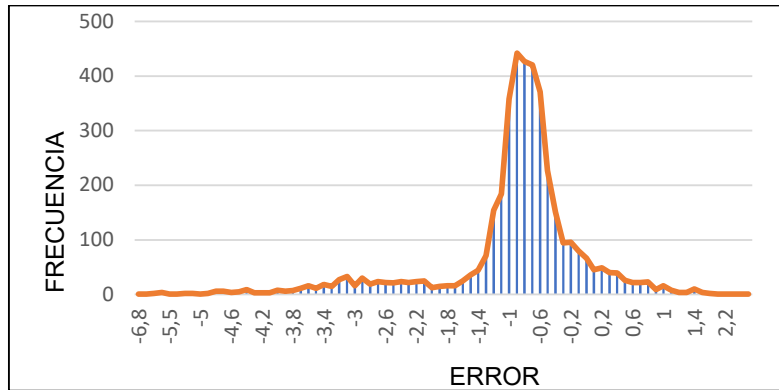
### Modelo de calibración de las AWS

Para la determinación de la ecuación de ajuste de temperatura se eligieron puntos específicos de temperatura y se obtuvo el promedio, siguiendo el proceso realizado en el INAMHI, con la ayuda del software Excel se pudo determinar la ecuación de regresión lineal y la distribución de datos como se muestra en la Figura III.1, siendo  $R^2$  el factor de determinación que indica la eficiencia de los valores al ajustar la línea de tendencia.



**FIGURA III. 1.** Ecuación de ajuste de temperatura en la estación de Calderón.

Sin embargo, se puede notar en el gráfico que existe un pequeño intervalo que no se ajusta a la línea de tendencia, por tanto, es necesario el análisis de la distribución de errores mediante los cuales se puede determinar una nueva variable para ajustar de mejor forma todos los datos existentes. En la Figura III.2 se muestra la distribución de error de los datos ajustados a la ecuación presentada anteriormente, los errores presentes de mayor frecuencia están en el intervalo de  $[-1.4, 1]$  que se los puede minimizar de acuerdo a la aplicación de las variables estadísticas como la media o mediana de los errores presentados. Además, se pueden observar valores aún mayores, pero con frecuencias bajas porque aún existieron datos erróneos y que no se eliminaron en el proceso de depuración, pero permite que el proceso realice el ajuste de los modelos.

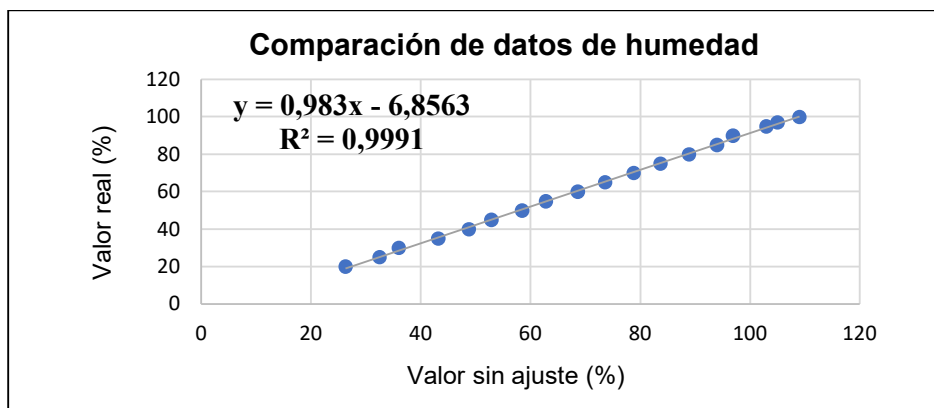


**FIGURA III. 2.** Distribución de error de frecuencia.

Al aplicar el proceso antes mencionado se presenta la Ecuación 1 mediante la cual se ajustaron de mejor forma los datos, aunque siguen existiendo errores, esto se debe a la variabilidad de valores en un determinado punto que no puede ser estabilizada en un valor constante.

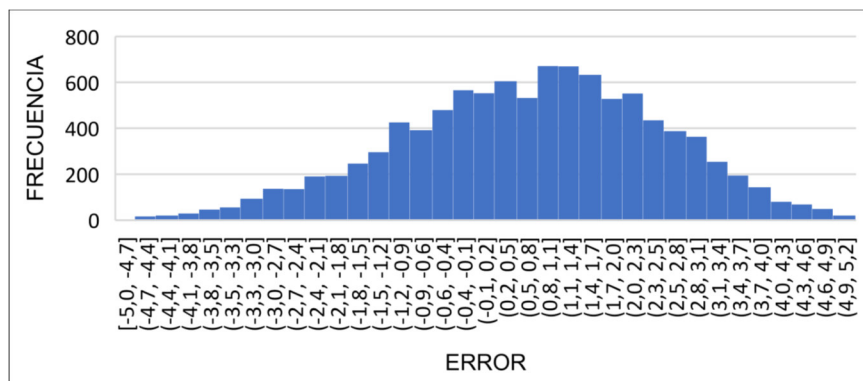
$$y = 1.0031 + 0.8808 \quad (1)$$

El procedimiento de ajuste para la variable de humedad es el mismo, como se puede observar en la Figura III.38 se podría deducir que se ajustan a todos los datos u obtienen errores mínimos, sin embargo, en este caso se analizaron los datos en ciertos intervalos en los cuales se encontraron variables a implementar luego de la ecuación, ya que la ecuación entregaba valores de humedad que sobrepasaban el rango del sensor [0%, 99.9%].



**FIGURA III. 3.** Ecuación de ajuste de Humedad.

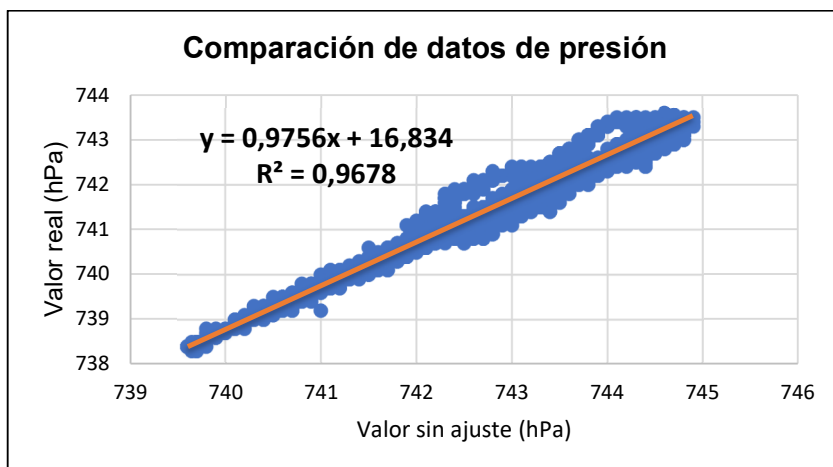
Luego de dicho análisis y la identificación de la distribución del error presentada en la Figura 3.39, se presentan las condiciones en la Ecuación 3.5 que describe de mejor forma el comportamiento del sensor en el parámetro de humedad.



**FIGURA III. 4.** Distribución de frecuencia de error.

$$y = \begin{cases} 0.983x - 6.8563 & x < 96 \\ 0.983x - 6.1663 & x = 96 \\ 0.983(x + 2) - 6.1663 & 96 < x < 99.6 \\ x - 0.69 & x \geq 99.6 \end{cases} \quad (2)$$

Por último, se ajusta el parámetro de presión, en este caso solo bastó introducir los datos totales y obtener la línea de tendencia con su respectiva ecuación y coeficiente de determinación como se presenta en la Figura 3.40 y la Ecuación 3.6.



**FIGURA III. 5.** Ecuación de ajuste de humedad.

$$y = 0.9756 + 16.834 \quad (3)$$

## ANEXO D

### Código de notificación de reinicio

```
import os
import sys
import smtplib
from email.mime.text import MIMEText
import time
from datetime import date
from datetime import datetime
from time import sleep
import mariadb

idSensor=3
fecha_act = datetime.now()
time_act = fecha_act.strftime("%H:%M:%S")
hoy = date.today()

def reiniciomail():
    origen = ' ' #Correo de origen del mensaje
    contraseña = ' ' #Contraseña del correo de origen
    destino = ' ' #Correo/s destino

    #Mensaje
    mensg = MIMEText("El nodo 3 - Cumbayá ha sido reiniciado el {}".format(hoy))
    mensg['Subject'] = 'NOTIFICACIÓN DE REINICIO ' #Asunto
    mensg['From'] = origen #Quien envía
    mensg['To'] = destino #Quien recibe

    server = smtplib.SMTP('smtp.gmail.com:587') #Conexion con el servidor por puerto 587(gmail)
    server.ehlo()
    server.starttls() #Protocolo de seguridad TLS (Transport Layer Security)
    server.login(origen,contraseña) #Acceso con correo y contraseña
    server.sendmail(origen,destino,mensg.as_string()) #Envio de mensaje

    #Almacenamiento de evento de base de datos
    try:
        conex = mariadb.connect(host=" ",
                                user=" ",
                                password=" ",
                                database=" ")
    except:
        server.close()
    cur = conex.cursor()
    evento = "INSERT INTO Eventos(Fecha, Hora, Sensor,
                                Evento)VALUES('%s','%s','%s','Reinicio')"%(hoy, time_act, idSensor)

    try:
```

```
        cur.execute(evento)
        conex.commit()
    except:
        conex.rollback()
        server.close()          #Desconexión del servidor gmail
        conex.close()
try:
    reiniciomail()
except:
    print("Error de conexion")
```

## ANEXO E

### Código de lectura de datos

```
import sys
import os
import board
import adafruit_dht
import Adafruit_BMP.BMP085 as BMP085
import smtplib
import subprocess
from email.mime.text import MIMEText
import time
from datetime import datetime
from datetime import date
from time import sleep
import mariadb

DHT_SENSOR = adafruit_dht.DHT22 #Definicion del tipo de sensor a utilizar
DHT_PIN = board.D14 #Definicion PIN al que está conectado
dhtDevice = DHT_SENSOR(DHT_PIN) #Inicializacion de dispositivo DHT22
idSensor = 3

def DATOS_DHT22():
    temperatura = dhtDevice.temperature
    humedad = dhtDevice.humidity
    return(temperatura, humedad)

def DATOS_BMP():
    try:
        bmp = BMP085.BMP085() #Inicializacion de sensor BMP en modo Standar
        presion = bmp.read_pressure()
        return(presion)
    except Exception as error:
        print(error)
        presion = 0
        return(presion)

#PROMEDIOS
def promedios():
    humedades = [0.00, 0.00, 0.00]
    presiones = [0.00, 0.00, 0.00]
    temperaturas = [0.00, 0.00, 0.00]
    for i in range(0, 3):
        try:
            temperatura, humedad = DATOS_DHT22()
            presion = DATOS_BMP()
            if humedad is not None and temperatura is not None:
                humedades[i] = humedad
```

```

        temperaturas[i] = temperatura
        presiones[i] = presion
        time.sleep(18)
except RuntimeError as error:
    print(error)
    temperatura, humedad = DATOS_DHT22()
    presion = DATOS_BMP()
    if humedad is not None and temperatura is not None:
        humedades[i] = humedad
        temperaturas[i] = temperatura
        presiones[i] = presion
        time.sleep(18)
promTemp = (temperaturas[0]+temperaturas[1]+temperaturas[2])/3
promPress = (presiones[0]+presiones[1]+presiones[2])/(3*100)
promHum = (humedades[0]+humedades[1]+humedades[2])/3
return(promTemp, promHum, promPress)

#Notificacion de error al correo
def notificacioncorreo():
    corigen = ' ' #Correo de origen del mensaje
    contraseña = ' ' #Contraseña del correo de origen
    cdestino = ' ' #Correo/s destino

    msg = MIMEText("Ha ocurrido un error (Ceros persistentes) por favor revisar el Nodo 3
- Cumbayá") #Mensaje
    msg['Subject'] = 'NOTIFICACIÓN DE ERROR ' #Asunto
    msg['From'] = corigen #Quien envía
    msg['To'] = cdestino #Quien recibe
    server = smtplib.SMTP('smtp.gmail.com:587') #Conexion por puerto 587(gmail)
    server.ehlo()
    server.starttls() #Protocolo de seguridad TLS (Transport Layer Security)
    server.login(corigen, contraseña) #Acceso con correo y contraseña
    server.sendmail(corigen, cdestino, msg.as_string()) #Envio de mensaje
    server.close() #Desconexión del servidor gmail

while True:
    inicio = time.time()

    #Conjunto de codigos para apertura de archivos .csv al iniciar ejecutar el programa
    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    today = date.today()
    nombreFile =
"/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,
today)
    nombrePermanente =
"/home/pi/PROYECTO/LECTURAS/GENERAL/LecturaGeneralSensor_%s.csv"%(idSen
sor)
    file1 = open(nombreFile, "+a")

```

```

file2 = open(nombrePermanente, "a+")

promTemp, promHum, promPress=promedios()
print(promTemp)
print(promHum)
print(promPress)

if promTemp > 0:
#Condiciones para escritura de datos
    str1 = "%s,%s,%d,% .1f,% .1f,% .1f\n"%(today, current_time, idSensor, promTemp,
promHum, promPress)
#Conjunto de codigos para escritura de datos en los archivos creados
    vart = time.time()-inicio
    t = 60 - vart
    time.sleep(t)
    file1.write(str1)
    file2.write(str1)
    file1.close()
    file2.close()
    print("Datos almacenados en CSV")

if promTemp == 0:
    leer=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt",
'r')
    cero=int(leer.read())
    cero=cero+1
    print(cero)
    leer.close()
    if cero == 1:

variable=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt",
"w")
    variable.write(str(cero))
    variable.close()
    try:
        conexion = mariadb.connect(host=" ",
                                user=" ",
                                password=" ",
                                database=" ")
    except mariadb.Error as e:
        print(e)
        os.system('sudo shutdown -r now')
        sleep(60)
        cursor = conexion.cursor()
        ceros = "INSERT INTO Eventos(Fecha, Hora, Sensor,Evento)VALUES('%s', '%s',
'%s', 'Cero detectado')"%(today,current_time, idSensor)
        try:
            cursor.execute(ceros)
            conexion.commit()

```



```

    except:
        conexion.rollback()
    conexion.close()
    os.system('sudo shutdown -r now')
    sleep(60)
elif cero == 2:

variable=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt",
"w")
    variable.write(str(cero))
    variable.close()
    try:
        notificacioncorreo()
    except Exception as e:
        print(e)
        sleep(10)

elif cero > 2:
    print("Tratando de tomar datos")
    sleep(10)
    continue
else:
    cero = 0

variable=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Deteccion_ceros.txt",
"w")
    variable.write(str(cero))
    variable.close()

```

## ANEXO F

### Código de Chequeo de datos

```
import sys
import os
import smtplib
import subprocess
import time
import mariadb
import pandas as pd
import numpy as np
from email.mime.text import MIMEText
from datetime import datetime
from datetime import date
from datetime import timedelta
from time import sleep

#DATOS PARA LECTURA DE ARCHIVOS
fecha_act = datetime.now()
time_act = fecha_act.strftime("%H:%M:%S")
today = date.today()
idSensor = 3
nombreFile
="/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,today)
file = open(nombreFile, "a")

datos_ant = 0
num_errores = 0
datos_act = 0

def lectura_datos():
    global datos_act
    cabecera = ["Fecha", "Hora", "Nodo", "Temperatura", "Humedad", "Presion"]
    datos=pd.read_csv(nombreFile, names = cabecera)
    df=pd.DataFrame(datos)
    print(df)
    datos_act = len(df)
    datos_act = datos_act
    return(datos_act)

def lectura_variable():
    global datos_ant
    global num_errores
    with open("/home/pi/PROYECTO/CODIGOS/VARIABLES/check_datos.txt", "r") as f:
        for line in f:
            datos_ant, num_errores = line.split(" ")
            datos_ant = int(datos_ant)
```

```

    num_errores = int(num_errores)
    return(datos_ant, num_errores)

#NOTIFICACION DE ERROR AL CORREO
def notificacioncorreo():
    corigen = ' ' #Correo de origen del mensaje
    contraseña = ' ' #Contraseña del correo de origen
    cdestino = ' ' #Correo/s destino
    #Mensaje
    msg = MIMEText("Revisar el Nodo 3 - Cumbayá presenta errores de adquisición de
datos")
    msg['Subject'] = 'NOTIFICACIÓN DE ERROR ' #Asunto
    msg['From'] = corigen #Quién envía
    msg['To'] = cdestino #Quién recibe
    server = smtplib.SMTP('smtp.gmail.com:587') #Conexion por puerto 587(gmail)
    server.ehlo()
    server.starttls() #Protocolo de seguridad TLS (Transport Layer Security)
    server.login(corigen, contraseña) #Acceso con correo y contraseña
    server.sendmail(corigen, cdestino, msg.as_string()) #Envío de mensaje
    print("El correo de error ha sido enviado")
    server.close() #Desconexión del servidor gmail

#NOTIFICACION DE ERROR AL CORREO
def error_deteccion():
    try:
        conexion = mariadb.connect(host=" ",
                                user=" ",
                                password=" ",
                                database=" ")
    except:
        sleep(10)
        os.system('sudo shutdown -r now')
        sleep(20)
        cursor = conexion.cursor()
        deteccion = "INSERT INTO Eventos(Fecha, Hora, Sensor,Evento)VALUES('%s', '%s',
'%s', 'Datos no detectados')"%(today,
                                time_act,
                                idSensor)
    try:
        cursor.execute(deteccion)
        conexion.commit()
    except:
        conexion.rollback()
    conexion.close()

datos_ant, num_errores = lectura_variable()
datos_act = lectura_datos()
numero_datos = datos_act - datos_ant
print("Datos Actuales =",datos_act)

```

```

print("Datos Anteriores =", datos_ant)
print("Datos Tomados =", numero_datos)

if numero_datos == 0:
    num_errores = num_errores + 1
    print("Numero de errores =", num_errores)
    if num_errores == 1:
        with open("/home/pi/PROYECTO/CODIGOS/VARIABLES/check_datos.txt", "w")
as f:
    f.write("%s %s" % (datos_act, num_errores))
    error_deteccion()
    sleep(10)
    os.system('sudo shutdown -r now')
    sleep(20)
    elif num_errores == 2:
        with open("/home/pi/PROYECTO/CODIGOS/VARIABLES/check_datos.txt", "w")
as f:
    f.write("%s %s" % (datos_act, num_errores))
    notificacioncorreo()
else:
    num_errores = 0
    with open("/home/pi/PROYECTO/CODIGOS/VARIABLES/check_datos.txt", "w") as
f:
    f.write("%s %s" % (datos_act, num_errores))

```

## ANEXO G

### Código de almacenamiento de base de datos

```
import sys
import os
import subprocess
import time
import mariadb
import pandas as pd
import numpy as np
from datetime import datetime
from datetime import date
from datetime import timedelta
from time import sleep

#DATOS PARA LECTURA DE ARCHIVOS
fecha_act = datetime.now()
time_act = fecha_act.strftime("%H:%M:%S")
today = date.today()
idSensor = 3
nombreFile
="/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,to
day)

cabecera = ["Fecha", "Hora", "Nodo", "Temperatura", "Humedad", "Presion"]
datos = pd.read_csv(nombreFile, names=cabecera)
print(datos)
df=pd.DataFrame(datos)

#Lectura de última hora almacenada
leer=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Verif_DB.txt", 'r')
dato_ant=int(leer.read())
leer.close()

fecha = df.iloc[-1,0]
hora = df.iloc[-1,1]
temp = df.iloc[-1,3]
hum = df.iloc[-1,4]
pres = df.iloc[-1,5]

if hora != hora_ant:
    try:
        #Almacenamiento de hora actual
        variable=open("/home/pi/PROYECTO/CODIGOS/VARIABLES/Verif_DB.txt", "w")
        variable.write(str(hora))
        variable.close()
        conexion = mariadb.connect(host=" ",
                                    user=" ",
```

```

        password=" ",
        database=" ")
except mariadb.Error as e:
    print(e)
    conexion.close()
cursor = conexion.cursor()
ceros = "INSERT INTO LecturaSensor(Fecha, Hora, Sensor, Temperatura, Humedad,
Presion)VALUES('%s', '%s', '%s', '%s', '%s', '%s')"%(fecha, hora, idSensor, temp, hum,
pres)
try:
    cursor.execute(ceros)
    conexion.commit()
except:
    conexion.rollback()
    conexion.close()

```

# ANEXO H

## Código de Interpolación

```
#IMPORTACION DE LIBRERIAS
import pandas as pd
import numpy as np
from datetime import datetime
from datetime import timedelta
from datetime import date
import time
import csv
import mariadb
import smtplib
from email.mime.text import MIMEText

#DATOS PARA LECTURA DE ARCHIVOS
fecha_act = datetime.now()
time_act = fecha_act.strftime("%H:%M:%S")
today = date.today()
yesterday=today-timedelta(days=1)
print(yesterday)
idSensor=3
nombreFile
="/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,y
esterday)
processFile="/home/pi/PROYECTO/LECTURAS/PROCESADAS/Sensor%s_Lectura%s.c
sv"%(idSensor,yesterday)
file = open(processFile, "a")
#Notificacion al correo
def interpolmail():
    origen = ' ' #Correo de origen del mensaje
    contraseña = ' ' #Contraseña del correo de origen
    destino = ' ' #Correo/s destino

    mensg = MIMEText("El nodo {}-Cumbaya ha realizado interpolación del archivo
{}".format(idSensor, nombreFile)) #Mensaje
    mensg['Subject'] = 'NOTIFICACIÓN DE INTERPOLACIÓN ' #Asunto
    mensg['From'] = origen #Quien envía
    mensg['To'] = destino #Quien recibe
    server = smtplib.SMTP('smtp.gmail.com:587') #Conexion con el servidor por puerto
587(gmail)
    server.ehlo()
    server.starttls() #Protocolo de seguridad TLS (Transport Layer Security)
    server.login(origen,contraseña) #Acceso con correo y contraseña
    server.sendmail(origen,destino,mensg.as_string()) #Envio de mensaje

conexion = mariadb.connect(host=" ",
```

```

        user=" ",
        password=" ",
        database=" ")
    cursor = conexion.cursor()
    interpol = "INSERT INTO Eventos(Fecha, Hora, Sensor,
Evento)VALUES('%s','%s','%s','Interpolacion')"%(today, time_act,idSensor)
    try:
        cursor.execute(interpol)
        conexion.commit()
    except:
        conexion.rollback()
    conexion.close()
    server.close()

#Definición de cabecera, insercion y lectura de datos
cabecera = ["Fecha", "Hora", "Nodo", "Temperatura", "Humedad", "Presion"]
datos=pd.read_csv(nombreFile, names=cabecera)
print(datos)
df=pd.DataFrame(datos)
datosfalt=1440-len(df)
print(datosfalt)

if datosfalt > 0:
    df["Fecha"] = pd.to_datetime(df["Fecha"] + " " + df["Hora"])
    print(df)
    df=pd.DataFrame(df)
    print(df)

    iniciosdoc=df.iloc[0,0]
    print(iniciosdoc)
    finaldoc=df.iloc[-1,0]
    print(finaldoc)

#Definición de hora inicial
horainicio = '%s 00:00:00'%yesterday
horainicio = np.datetime64(horainicio)
print(horainicio)
horainicio = horainicio.astype(datetime)
print(horainicio)

#Definición de hora final
horafinal= '%s 23:59:00'%yesterday
horafinal = np.datetime64(horafinal)
print(horafinal)
horafinal = horafinal.astype(datetime)
print(horafinal)

#Definición de nuevas filas
filanueva=pd.Series([horainicio, horainicio, idSensor, np.nan, np.nan, np.nan],

```



```

        index=['Fecha','Hora', 'Nodo', 'Temperatura', 'Humedad', 'Presion'])
filanueva1=pd.Series([horafinal, horafinal, idSensor, np.nan, np.nan, np.nan],
        index=['Fecha','Hora', 'Nodo', 'Temperatura', 'Humedad', 'Presion'])

#Comparación de horas iniciales y finales
if finaldoc.hour != 23 or finaldoc.minute != 59:
    print("Aregar fila")
    df=df.append(filanueva1, ignore_index=True, sort=False)
print(df)
if iniciodoc.hour != 00 or iniciodoc.minute != 00:
    df=df.append(filanueva, ignore_index=True, sort=False)
print(df)

df=df.set_index('Fecha').resample('1min').mean() #Remuestreo a 1min
df=df.reset_index(level=['Fecha'])
df = df.sort_values(by=['Fecha'], ascending=[True]) #Ordenar ascendente
print(df)
#Interpolación lineal, hacia adelante y hacia atrás
df = df.interpolate(method="linear", axis=1).ffill().bfill()
print(df)
df=pd.DataFrame(df)
print(df['Fecha'])
df["Hora"] = df["Fecha"].dt.time
df["Fecha"] = df["Fecha"].dt.date
df=df[['Fecha', 'Hora', 'Nodo', 'Temperatura', 'Humedad', 'Presion']]
#Almacenamiento en CSV
df.to_csv(processFile, index=False)
print(df)
interpolmail()
else:
    df.to_csv(processFile, index=False)

```

# ANEXO I

## Código de almacenamiento en servidor FTP

```
import time
from datetime import datetime,timedelta
from datetime import date
import sys
import subprocess

today = date.today()
yesterday=today-timedelta(days=1)
idSensor=3
nombreFile ="bash /home/pi/PROYECTO/CODIGOS/Archivo_diario.sh
Sensor%s_Lectura%s.csv >
/home/pi/PROYECTO/CODIGOS/NOTIFICACIONES/Salidadiaria.txt"
%(idSensor,yesterday)
nombreprocess ="bash /home/pi/PROYECTO/CODIGOS/Archivo_procesado.sh
Sensor%s_Lectura%s.csv >
/home/pi/PROYECTO/CODIGOS/NOTIFICACIONES/SalidaProcess.txt" %(idSensor,
yesterday)
subprocess.call(nombreFile,shell=True)
subprocess.call(nombreprocess, shell=True)
```

- **Archivo\_diario.sh**

```
#!/bin/bash
echo "No Ejecutado"
echo $0
echo $1

cd /home/pi/PROYECTO/LECTURAS/DIARIAS
ls
aux="/home/pi/PROYECTO/LECTURAS/DIARIAS"
HOST=' ' #IP HOST
USER=" " #USER HOST
PASSWD=" " #PASSWORD
FILE=$1
REMOTEPATH='public_html/Lecturas_Diarias/NODO_3_CUMBAYA/SIN_PROCESA
R'
echo "Ejecutado"
ftp -n $HOST <<END_SCRIPT
quote USER $USER
quote PASS $PASSWD
cd $REMOTEPATH
put $FILE
quit
END_SCRIPT
```

exit 0

- **Archivo\_procesado.sh**

```
#!/bin/bash
echo "No Ejecutado"
echo $0
echo $1

cd /home/pi/PROYECTO/LECTURAS/PROCESADAS
ls
aux="/home/pi/PROYECTO/LECTURAS/PROCESADAS"
HOST=' ' #IP HOST
USER=" " #USER HOST
PASSWD=" " #PASSWORD
FILE=$1
REMOTEPATH='public_html/Lecturas_Diarias/NODO_3_CUMBAYA/PROCESADOS'
echo "Ejecutado"
ftp -n $HOST <<END_SCRIPT
quote USER $USER
quote PASS $PASSWD
cd $REMOTEPATH
put $FILE
quit
END_SCRIPT
exit 0
```

## ANEXO J

### Código de Aplicación

```
import mariadb
import subprocess
import time
import csv
import os
import subprocess
import pandas as pd
from tkinter import *
from tkinter import messagebox as ms
from tkinter import filedialog
from tkinter import ttk
from datetime import date
from datetime import datetime
from datetime import timedelta
from time import sleep

datos = []
event = []
fecha=[0.00,0.00,0.00]
hora=[0.00,0.00,0.00]
sensor=[0.00,0.00,0.00]
temp=[0.00,0.00,0.00]
hum=[0.00,0.00,0.00]
pres=[0.00,0.00,0.00]
rows2=[0.00,0.00,0.00,0.00,0.00,0.00,0.00]

#FUNCIONES ESTÁNDAR
#Salir
def salir():
    exit=ms.askyesno("Mensaje", "¿Desea salir del sistema?")
    if exit==True:
        quit()

#Actualizar
def retornar():
    conexion = mariadb.connect(host=" ",
                               user=" ",
                               password=" ",
                               database=" ")
    cursor = conexion.cursor()
    query = "SELECT Fecha, Hora, Sensor, Temperatura, Humedad, Presion from
LecturaSensor ORDER BY Fecha DESC, Hora DESC"
    cursor.execute(query)
    rows = cursor.fetchall()
```

```

datosmdb(rows)

#FUNCIONES DE DATOS ACTUALES, BASE DE DATOS Y CONSULTAS
#Datos actuales
def lecturas():
    try:
        fecha_act = datetime.now()
        time_act = fecha_act.strftime("%H:%M:%S")
        today = date.today()
        idSensor = 3
        nombreFile
        ="/home/pi/PROYECTO/LECTURAS/DIARIAS/Sensor%s_Lectura%s.csv"%(idSensor,
today)
        datos = pd.read_csv(nombreFile, names=cabecera)
        df=pd.DataFrame(datos)
        for i in range (0,3):
            fecha[i]=df.iloc[-i-1, 0]
            hora[i]=df.iloc[-i-1, 1]
            sensor[i]=df.iloc[-i-1, 2]
            temp[i]=df.iloc[-i-1, 3]
            hum[i]=df.iloc[-i-1, 4]
            pres[i]=df.iloc[-i-1, 5]
            rows2[i]=[fecha[i], hora[i], sensor[i], temp[i], hum[i], pres[i]]
            lec1.config(text=rows2[0], bg="white", fg="midnight blue", font=("Arial", 14))
            lec2.config(text=rows2[1], fg="midnight blue", font=("Arial", 14))
            lec3.config(text=rows2[2], bg="white", fg="midnight blue", font=("Arial", 14))
            lec1.after(60000,lecturas)
            lec2.after(60000,lecturas)
            lec3.after(60000,lecturas)
        except FileNotFoundError as error:
            lec1.config(text="Ups! : Error al encontrar datos actuales", bg="white",
                fg="midnight blue", font=("Arial", 14))
            lec1.after(60000, lecturas)

#Datos en Treeview
def datosmdb(rows):
    global datos
    datos = rows
    data.delete(*data.get_children())
    for i in rows:
        data.insert("", 'end', values=i)

#Datos eventos
def datosevent(rows1):
    global event
    event = rows1
    eventos.delete(*eventos.get_children())
    for i in rows1:

```

```

    eventos.insert("", 'end', values=i)

#FUNCIONES APP
#Buscar datos en DB por fecha
def buscardb():
    fecha = fecha1.get()
    if len(fecha)>0:
        conexion = mariadb.connect(host=" ",
                                   user=" ",
                                   password=" ",
                                   database=" ")
        cursor = conexion.cursor()
        query = "SELECT Fecha, Hora, Sensor, Temperatura,Humedad,
                "Presion from LecturaSensor WHERE Fecha LIKE '%"+fecha+"%"
        cursor.execute(query)
        rows = cursor.fetchall()
        datosmdb(rows)
    else:
        ms.showwarning("Error", "Ingresa una fecha válida por favor")

#Importar archivos csv o excel a ventana
def import_arch():
    datos.clear()
    flnopen = filedialog.askopenfilename(initialdir =
    "/home/pi/PROYECTO/LECTURAS/",
        title = "Gestor de archivos",
        filetypes = (("All Files", "*.*"),
                    ("csv Files", "*.csv"),
                    ("xlsx Files", "*.xlsx")))

    file_path = flnopen
    try:
        archivo = r"{}".format(file_path)
        flnopen,fileextension=os.path.splitext(archivo)
        if fileextension == ".csv":
            df = pd.read_csv(archivo)
            rows = df.to_numpy().tolist()
            for i in rows:
                datos.append(i)
                datosmdb(datos)
        elif fileextension == ".xlsx":
            df = pd.read_excel(archivo)
            rows = df.to_numpy().tolist()
            for i in rows:
                datos.append(i)
                datosmdb(datos)
    except:
        ms.showinfo("Mensaje", "No se puede encontrar el archivo")
        return None

```

```

#Exportar datos de ventana a un archivo tipo EXCEL
def export_datos():
    new=list()
    if len(datos) < 1:
        ms.showwarning("Error",
            "No existen datos para exportar")
        return False
    export=ms.askyesno("Mensaje",
        "¿Desea exportar los datos seleccionados a excel?")
    if export==True:
        flnsave = filedialog.asksaveasfilename(initialdir=
            "/home/pi/PROYECTO/LECTURAS/ARCHIVOS
EXPORTADOS/",
            title="EXPORTAR A EXCEL",
            filetypes=(("xlsx File", "*.xlsx"),
                ("All Files", "*.*")))
        for i in datos:
            new.append(i)
        df=pd.DataFrame(new)
        df.to_excel(flnsave, index=None, header=False)
        ms.showinfo("Mensaje", "Datos exportados con éxito, revisar carpeta destino")
        retornar()
    else:
        ms.showinfo("Mensaje", "Operación cancelada")
        retornar()

#FUNCIONES DE PROGRAMACIÓN DE REINICIO
#Reinicio inmediato
def reinicio():
    reboot=ms.askyesno("Mensaje", "¿Desea reiniciar el Nodo Cumbayá?")
    if reboot==True:
        subprocess.call("shutdown -r now", shell=True)
    else:
        ms.showinfo("Mensaje", "Reinicio cancelado")

#Reinicio programado
def reinicioprg():
    if listopciones.get() == "Sin reinicio":
        variable=open("/home/pi/PROYECTO/CODIGOS/APP/Variable_reinicio.txt", "w")
        variable.close()
        ms.showinfo("Mensaje", "Nodo sin reinicio programado")
    elif listopciones.get() == "1 día":
        n=1
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
    elif listopciones.get() == "2 días":
        n=2
        guardar(n)
        ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())

```

```

elif listopciones.get() == "3 días":
    n=3
    guardar(n)
    ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
elif listopciones.get() == "4 días":
    n=4
    guardar(n)
    ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())
elif listopciones.get() == "5 días":
    n=5
    guardar(n)
    ms.showinfo("Mensaje", "El nodo se reiniciará en: "+dia.get())

#FUNCION DE ALMACENAMIENTO DE FECHA DE REINICIO
def guardar(n):
    now=datetime.now()
    reiniciodia=now+timedelta(days=n)
    variable=open("/home/pi/PROYECTO/CODIGOS/APP/Variable_reinicio.txt", "w")
    variable.write(str(reiniciodia))
    variable.close()

#PANTALLA DE OPCIONES Y FUNCIONES
def Pantalla_in():
    global ventana
    global data
    global fecha1
    global cursor
    global eventos
    global etiquetareinicio
    global dia
    global lec1, lec2, lec3
    global listopciones

    conexion = mariadb.connect(host=" ",
                                user=" ",
                                password="",
                                database=" ")
    cursor = conexion.cursor()

    ms.showinfo("Mensaje", "Está ingresando a los datos del nodo Cumbayá")
    ventana = Toplevel(principal)
    ventana.title("NODO 3 - CUMBAYÁ")
    ventana.geometry("800x700")
    ventana.configure(bg="black")
    ventana.resizable(False, False)

#DIVISIONES DE LA PANTALLA
div1=LabelFrame(ventana, text="DATOS NODO CUMBAYÁ",

```



```

        font=("Arial",10))
div1.pack(fill="both", expand="yes")

div1_1=LabelFrame(div1, text="DATOS Y CONSULTAS",
        font=("Arial",9))
div1_1.pack(fill="both", expand="yes", padx=5, pady=5)

div2=LabelFrame(ventana, text="EVENTOS",font=("Arial",9))
div2.pack(fill="both", expand="yes")

div3=LabelFrame(ventana, text="PROGRAMACIÓN DE REINICIO",
        font=("Arial",9))
div3.pack(fill="both", expand="yes")

logo_dir = "/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/escudo.gif"
logo = PhotoImage(file=logo_dir)
logo_red = logo.subsample(4)
fondo1 = Label(div1_1, image=logo_red)
fondo1.grid(row=0, column=0)

#BOTON SALIR
etiquetaatr=Label(div1_1, text="Salir", fg="black", font=("Arial",12))
etiquetaatr.place(x=711, y=190)
atr_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/salir.PNG"
atr = PhotoImage(file = atr_dir)
atras_red = atr.subsample(5)
botonatr = Button(div1_1, image=atras_red, command=salir)
botonatr.place(x=660, y=190)

#LECTURAS ACTUALES
lec=Label(div1_1, text="LECTURAS ACTUALES",fg="red", font=("Arial",12))
lec.grid(row=6, column=1)
lec1=Label(div1_1)
lec1.grid(row=7, column=1)
fech=Label(div1_1, text="Fecha", fg="black", font=("Arial",12))
fech.place(x=340, y=250)
hor=Label(div1_1, text="Hora", fg="black", font=("Arial",12))
hor.place(x=430, y=250)
sen=Label(div1_1, text="ID", fg="black", font=("Arial",12))
sen.place(x=485, y=250)
tmp=Label(div1_1, text="°C", fg="black", font=("Arial",12))
tmp.place(x=510, y=250)
hmd=Label(div1_1, text="%", fg="black", font=("Arial",12))
hmd.place(x=555, y=250)
pss=Label(div1_1, text="hPa", fg="black", font=("Arial",12))
pss.place(x=590, y=250)
lec1=Label(div1_1)
lec1.grid(row=8, column=1)
lec2=Label(div1_1)

```

```

lec2.grid(row=9,column=1)
lec3=Label(div1_1)
lec3.grid(row=10, column=1)
lecturas()

#RETORNO A DATOS DB (ACTUALIZAR)
etiqueta5=Label(div1_1, text="Retornar", fg="black", font=("Arial",12))
etiqueta5.place(x=711, y=140)
act_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/actualizar.PNG"
actualizar = PhotoImage(file = act_dir)
actualizar_red = actualizar.subsample(4)
boton4 = Button(div1_1, image=actualizar_red, command = retornar)
boton4.place(x=660, y=140)

#VARIABLES DE ENTRADA
fecha1=StringVar()

#BASE DE DATOS
etiqueta= Label(div1_1, text="BASE DE DATOS", fg="red", font=("Arial", 12))
etiqueta.grid(row=0, column=1, padx=2, pady=2)
etiqueta1 = Label(div1_1, text="Ingresa la Fecha (año-mes-día) y presiona BUSCAR",
fg="black", font=("Arial", 10))
etiqueta1.grid(row=1, column=0, padx=2, pady=2)
fecha = Entry(div1_1, textvariable=fecha1)
fecha.grid(row=1, column=1, padx=2, pady=2)
boton1 = Button(div1_1, text="BUSCAR", command=buscardb, bg="red", fg="white",
height="1", width="10", font=("Arial", 12))
boton1.grid(row=1, column=2, padx=2, pady=2)

#IMPORTAR ARCHIVOS A VENTANA

etiqueta2 = Label(div1_1, text="IMPORTAR/EXPORTAR ARCHIVOS", fg="red",
font=("Arial",12))
etiqueta2.grid(row=2, column=1, padx=2, pady=2)
etiqueta3 = Label(div1_1, text="Para importar archivos oprima el botón - - >",
fg="black", font=("Arial", 10))
etiqueta3.grid(row=3, column=0, padx=2, pady=2)
boton2 = Button(div1_1, text="BUSCAR", command = import_arch, bg="blue",
fg="white", height="1", width="10", font=("Arial", 12))
boton2.grid(row=3, column=1, padx=2, pady=2)

#EXPORTAR DATOS A EXCEL
etiqueta4 = Label(div1_1, text="Para exportar los datos oprime el botón - - >",
fg="black", font=("Arial", 10))
etiqueta4.grid(row=4, column=0, padx=2, pady=2)
boton3 = Button(div1_1, text="EXPORTAR", command=export_datos, bg="red",
fg="white", height="1", width="10", font=("Aria", 12))
boton3.grid(row=4, column=1, padx=2, pady=2)

```

#### #DEFINICION DE TABLA DE VALORES DE LECTURAS

```
data=ttk.Treeview(div1, columns=(1,2,3,4,5,6), show="headings", height="4")
data.pack()
```

```
data.column("1", width=100, minwidth=100)
data.column("2", width=100, minwidth=100)
data.column("3", width=100, minwidth=90)
data.column("4", width=140, minwidth=140)
data.column("5", width=130, minwidth=130)
data.column("6", width=130, minwidth=130)
```

```
data.heading(1, text="Fecha")
data.heading(2, text="Hora")
data.heading(3, text="Nodo")
data.heading(4, text="Temperatura(°C)")
data.heading(5, text="Humedad(%)")
data.heading(6, text="Presion(hPa)")
```

#### #DATOS DATABASE INICIAL

```
query = "SELECT Fecha, Hora, Sensor, Temperatura, Humedad, Presion from
LecturaSensor ORDER BY Fecha DESC, Hora DESC"
cursor.execute(query)
rows = cursor.fetchall()
datosmdb(rows)
```

#### #EVENTOS

#### #DEFINICION DE TABLA DE EVENTOS

```
eventos=ttk.Treeview(div2, columns=(1,2,3,4), show="headings", height="3")
eventos.pack()
```

```
eventos.column("1", width=100, minwidth=100)
eventos.column("2", width=100, minwidth=100)
eventos.column("3", width=100, minwidth=90)
eventos.column("4", width=150, minwidth=100)
```

```
eventos.heading(1, text="Fecha")
eventos.heading(2, text="Hora")
eventos.heading(3, text="Sensor")
eventos.heading(4, text="Evento")
```

```
query = "SELECT Fecha, Hora, Sensor, Evento from Eventos ORDER BY Fecha
DESC, Hora DESC"
cursor.execute(query)
rows1 = cursor.fetchall()
datosevent(rows1)
```

#### #CONFIGURACIÓN DE REINICIO

```
etiquetareinicio=Label(div3, text="Reiniciar ahora", fg="black", font=("Arial",10))
```

```

etiquetareinicio.grid(row=0, column=0, padx=10, pady=10)
img_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/reinicio.png"
reinicio_imag = PhotoImage(file = img_dir)
reinicio_red=reinicio_imag.subsample(10)
reiniciob=Button(div3, image=reinicio_red, command=reinicio)
reiniciob.grid(row=0, column=1, padx=10, pady=10)

#PROGRAMACIÓN DÍAS DE REINICIO
dia = StringVar()
etiquetareinicio=Label(div3, text="Opciones de reinicio:", fg="black",font=("Arial",10))
etiquetareinicio.grid(row=0, column=3, padx=10, pady=10)
opciones=["Sin reinicio", "1 día", "2 días", "3 días", "4 días", "5 días"]
listopciones=ttk.Combobox(div3, width=10, values=opciones, state="readonly",
textvariable=dia)
listopciones.grid(row=0, column=4, padx=10, pady=10)
ejec=Button(div3, text="Ejecutar", command=reinicioprg, fg="midnight blue",
font=("Arial", 12))
ejec.grid(row=0,column=5, padx=10, pady=10)

ventana.mainloop()

#PANTALLA DE PRESENTACIÓN
def pantalla_principal():
    global reinicio
    global Pantalla_in
    global datosmdb
    global buscardb
    global retornar
    global fecha1
    global principal
    global salir
    global reinicioprg

    principal = Tk()
    principal.title("NODO 3 - CUMBAYÁ")
    principal.geometry("700x500")
    principal.configure(bg='black')
    principal.resizable(False, False)

    caratula_dir = "/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/caratula.gif"
    caratula = PhotoImage(file = caratula_dir)
    fondo = Label(principal, image=caratula)
    fondo.pack()

    #Botón Ingresar
    etiquetain=Label(principal, text="Ingresar",
                    bg="light cyan", fg="black", font=("Arial",15))
    etiquetain.place(x=350, y=430)
    ing_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/entrar.PNG"

```

```

ing = PhotoImage(file = ing_dir)
ing_red = ing.subsample(3)
entrar=Button(text="INGRESAR",image=ing_red, command=Pantalla_in)
entrar.place(x=450, y=420)
#Botón Salir
etiquetasal=Label(principal, text="Salir",
                  bg="light cyan", fg="black", font=("Arial",15))
etiquetasal.place(x=140, y=430)
sal_dir="/home/pi/PROYECTO/CODIGOS/APP/IMAGENES/salir.PNG"
salir_img = PhotoImage(file = sal_dir)
sal_red = salir_img.subsample(3)
sal=Button(text="SALIR", image=sal_red, command=salir)
sal.place(x=200, y=415)

principal.mainloop()

pantalla_principal()

```

## ORDEN DE EMPASTADO