Aplicación de algoritmos de inteligencia artificial en sistemas de IoT orientado a la domótica

Mario Plana Casarrubios

Máster Universitario en Ingeniería de Telecomunicación
Smart Cities

Tutor/a de TF
Rubén Molina Casanovas
Profesor/a responsable de la asignatura
Carlos Monzo Sánchez
09/01/2023

Universitat Oberta de Catalunya

Ficha del Trabajo Final

Título del trabajo:	Aplicación de algoritmos de inteligencia artificial en sistemas de IoT orientado a la domótica
Nombre del autor/a:	Mario Plana Casarrubios
Nombre del Tutor/a de TF:	Rubén Molina Casanovas
Nombre del/de la PRA:	Carlos Monzo Sánchez
Fecha de entrega:	09/01/2023
Titulación o programa:	Máster Universitario en Ingeniería de Telecomunicación
Área del Trabajo Final:	Smart Cities
Idioma del trabajo:	Castellano
Palabras clave	Smart Cities, domótica, IoT, Inteligencia Artificial, machine learning

Resumen del Trabajo

La constante evolución de Internet ha derivado en una cuarta revolución industrial llevando la transformación digital a las industrias y a las empresas. Esto ha supuesto la aparición de tecnologías como computación en la nube, redes de internet móvil, robótica avanzada, Internet de las cosas, gemelos digitales o inteligencia artificial, teniendo un gran impacto socioeconómico.

Con el paso del tiempo, estas tecnologías están llegando a la vida cotidiana de las personas de la mano de smartphones y diferentes dispositivos que se instalan en los hogares como sensores de temperatura, de calidad del aire, sensores de movimiento o anti-incendio.

En el presente trabajo se pretende hacer uso de tecnologías en la nube para el procesamiento y almacenamiento de información, y para la ejecución de algoritmos de inteligencia artificial a partir de datos obtenidos mediante dispositivos IoT. El objetivo es crear un sistema capaz de ayudar en la toma de decisiones al usuario final facilitándole el desempeño en su vida diaria.

Abstract

The constant evolution of the Internet has led to a fourth industrial revolution bringing digital transformation to industries and companies. This has resulted in the emergence of technologies such as cloud computing, mobile internet networks, advanced robotics, the Internet of Things, digital twins and artificial intelligence, having a major socio-economic impact.

As time goes by, these technologies are reaching the everyday life of people through smartphones and different devices that are installed in homes, such as temperature sensors, air quality sensors, motion sensors or fire sensors.

This work aims to make use of cloud technologies for information processing and storage, and for the execution of artificial intelligence algorithms based on data obtained from IoT devices. The purpose is to create a system capable of helping the end user to make decisions, improving their performance in their everyday life.

Índice

1. Introducción 3
   1.1. Contexto y justificación del Trabajo 3
   1.2. Objetivos del Trabajo 3
   1.3. Impacto en sostenibilidad, ético-social y de diversidad 4
   1.4. Enfoque y método seguido 4
   1.5. Planificación del trabajo 5
   1.6. Breve sumario de productos obtenidos 8
   1.7. Breve descripción de otros capítulos de la memoria 8
2. Estado del arte 9
   2.1. Estado actual del IoT aplicado a la domótica 9
      2.1.1 Productos comerciales 9
      2.1.2 Proyectos 10
   2.2. Estado actual de la IA aplicada a la domótica 11
   2.4 Plataformas cloud para IoT 15
3. Solución propuesta 16
   3.1. Planteamiento formal del problema 16
   3.2. Justificación de la solución 17
   3.3. Implementación 20
      3.3.1 Implementación hardware 20
      3.3.2 Configuración de AWS IoT Core 23
      3.3.3 Implementación del código 24
      3.3.4 Uso cliente de prueba de MQTT 24
      3.3.5 Configuración de servicios de almacenamiento 26
      3.3.6 Configuración de reglas 30
      3.3.6 Visualización de los datos 32
      3.3.7 Análisis y ejecución de algoritmos 35
4. Resultados 38
   4.1 Resultados de la visualización de datos 39
   4.2 Resultados de los análisis y ejecución de algoritmos 41
      4.2.1 Cálculo de estadísticas móviles y prueba ADF 41
      4.2.2 Ejecución de predicción con *NeuralProphet* 42

4.2.3 Ejecución de regresión lineal simple 44
5. Conclusiones y líneas futuras 45
5.1 Conclusiones 45
5.2 Líneas futuras 47
6. Presupuesto 47
7. Glosario 49
8. Bibliografía 50
9. Anexos 52
9.1 Código proyecto Arduino 52
9.1.1 ESP8266_AWS 52
9.1.2 Secrets.h 55
9.2 Código scripts de Python 58
9.2.1 Ejecución de estadísticas móviles y pruebas ADF 58
9.2.2 Ejecución de predicción con *NeuralProphet* 59
9.2.3 Ejecución de regresión lineal simple 59

Lista de Figuras

Figura 1: Diagrama de Gantt 7
Figura 2: Entorno domótico [4] 10
Figura 3: Arquitectura de la solución 16
Figura 4: Conexión WIFI Arduino Uno [12] 17
Figura 5: NodeMCU Lolin V3 Modulo ESP8266 18
Figura 6: HiLetgo ESP8266 NodeMCU CP2102 ESP-12E 18
Figura 7: Esquema del montaje 20
Figura 8: Montaje hardware (1/5) 21
Figura 9: Montaje hardware (2/5) 21
Figura 10: Montaje hardware (3/5) 22
Figura 11: Montaje hardware (4/5) 22
Figura 12: Montaje hardware (5/5) 22
Figura 13: Objeto AWS IoT Core 23
Figura 14: Política AWS IoT Core 24
Figura 15: Cliente de prueba MQTT 25
Figura 16: Mensajes recibidos por MQTT 25
Figura 17: BBDD Amazon TimeStream 26
Figura 18: Tabla BBDD Amazon TimeStream 27
Figura 19: Canal IoT Analytics 28
Figura 20: Canalización IoT Analytics 28
Figura 21: Almacén de datos IoT Analytics 29
Figura 22: Conjunto de datos IoT Analytics 29
Figura 23: Instancia de bloc de notas Amazon SageMaker 30
Figura 24: Regla envío de datos a Amazon TimeStream 31
Figura 25: Regla envío de datos a AWS IoT Analytics 31
Figura 26: Pantalla de inicio de Grafana 32
Figura 27: Ejecución de la consulta 33
Figura 28: Resultado de la consulta 33
Figura 29: Gráfica en Grafana 34
Figura 30: Consulta A en Grafana 34
Figura 31: Consulta B en Grafana 34
Figura 32: Dashboard en Grafana 35
Figura 33: Consulta al conjunto de datos IoT Analytics 36
Figura 34: Datos procesados 37
Figura 35: Visualización de datos en Jupyter Notebook 37
Figura 36: Visualización de datos en Grafana 39
Figura 37: Visualización de datos en Jupyter Notebook 40
Figura 38: Estadísticas móviles de humedad 41
Figura 39: Prueba ADF de humedad 41
Figura 40: Estadísticas móviles de temperatura 42
Figura 41: Prueba ADF de temperatura 42

Figura 42: Predicción de humedad 43
Figura 43: Componentes encontradas de humedad 43
Figura 44: Predicción de temperatura 43
Figura 45: Componentes encontradas de temperatura 44
Figura 46: Resultados OLS de la regresión 44
Figura 47: Representación gráfica del modelo de regresión 45
Figura 48: Facturación total AWS 48

1. Introducción

En este trabajo se desea desarrollar un sistema de monitorización del hogar, donde se abarcará la recogida de datos, su envío y almacenamiento para finalmente emplearlos en la toma de decisiones.

1.1. Contexto y justificación del Trabajo

La cuarta revolución industrial, conocida también como industria 4.0, hace referencia a la transformación digital de la industria, donde la constante evolución de Internet sin duda ha tenido un impacto en la economía y en la sociedad del presente siglo.

La aparición de las tecnologías que la componen, como son la computación en la nube (del inglés, Cloud Computing), las redes de Internet móvil, la robótica avanzada, el Internet de las cosas (IoT, del inglés, Internet of Things) o el análisis de datos para predecir eventos futuros, están implantándose a través de la transformación digital en las industrias y en las empresas. Sin embargo, todavía no es común ver este tipo de tecnologías aplicadas al hogar o en el ámbito cotidiano, donde lo más fácil de encontrar son algunos sistemas de medida de temperatura, calidad del aire, sensores de movimiento o anti-incendio entre otros.

Con este trabajo lo que se quiere conseguir es ir un paso más allá, donde se hará uso de tecnologías en la nube para el procesamiento y almacenamiento de los datos, y para la ejecución de algoritmos de inteligencia artificial. Con ello lo que se pretende es disponer de un sistema capaz de ayudar en la toma de decisiones al usuario final facilitándole el desempeño en su vida diaria.

1.2. Objetivos del Trabajo

Los objetivos que se desean alcanzar con la realización del presente trabajo son los siguientes:

- Creación de un dispositivo que sea capaz de medir parámetros del hogar. Debe ser capaz de recoger los datos, y enviarlos para su posterior almacenamiento.
- Creación de una infraestructura en la nube que sirva para almacenar y soportar una visualización avanzada de los datos, así como ejecutar scripts de inteligencia artificial.

- El sistema completo debe ser capaz de ayudar en la toma de decisiones del usuario.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este trabajo tiene un gran impacto en sostenibilidad, en el apartado ético-social y de diversidad. De forma directa o indirecta cumple con algunos objetivos de desarrollo sostenible (ODS) de la Agenda 20/30 [1]. A continuación, se enumeran las ODS implicadas y se comenta el impacto del presente trabajo:

- ODS 3: Salud y bienestar

En este apartado, el proyecto podría tener futuras aplicaciones en el cuidado de ancianos y personas dependientes, así como vigilancia de pacientes que se encuentran en casa. También se podría integrar con sistemas de seguridad, ayudando a las personas tanto cuando estén dentro de casa como fuera. Estas aplicaciones impactarían positivamente en la mejora de calidad de vida de las personas y en su bienestar.

- ODS 9: Industria, innovación e infraestructura

Aunque este proyecto se ha centrado en la domótica, también se podría extrapolar al uso en Industria 4.0 como se comentaba, con aplicaciones como el uso inteligente de la energía o detección y predicción de averías con mantenimiento preventivo.

- ODS 11: Ciudades y comunidades sostenibles

Las aplicaciones comentadas en las anteriores ODS se podrían escalar perfectamente para un uso global en ciudades y comunidades, orientando a hacer un uso eficiente de los recursos y cumpliendo objetivos de desarrollo sostenible. Por ejemplo, se podrían introducir mejoras en las carreteras evitando atascos en el tráfico en medida de lo posible, o congestiones en el transporte público.

1.4. Enfoque y método seguido

El sistema consiste en un producto nuevo realizado desde cero, el cual se puede dividir en la parte hardware y la parte software.

Para la parte hardware será necesario disponer de una placa de desarrollo sobre la que se puedan conectar diferentes dispositivos y sensores para la recogida y en envío de los datos. En este caso se ha elegido la plataforma de creación electrónica Arduino. Esto se debe a que, a diferencia de otros competidores como Raspberry Pi, Arduino es de código abierto y está basada en software y hardware libre, lo que permite que sea más flexible y fácil de integrar.

Por otro lado, para la parte software se necesita disponer de una infraestructura cloud. En este caso se ha optado por la plataforma Amazon Web Services frente a rivales como Microsoft Azure o Google Cloud, ya que dispone de una gran variedad de servicios dentro de su plataforma, compatibilidad con un gran número de dispositivos, fácil integración de inteligencia artificial y además ofrece opciones gratuitas que son más atractivas que el resto.

1.5. Planificación del trabajo

Para poder llevar a cabo este trabajo es necesario disponer de los siguientes recursos:

- Placa de desarrollo Arduino y sus respectivos periféricos y programas.
- Cuenta en Amazon Web Services.
- PC.

A continuación, se enumeran las fases principales del proyecto y sus actividades:

- Hito 1: Definición de objetivos, plan de trabajo y resultados previstos.
- Hito 2: Estado del arte.
- Hito 3: Obtención del producto del proyecto:
    - Implementación del hardware.
    - Implementación de la infraestructura.
    - Integración.
    - Desarrollo de algoritmos y visualización de datos.
- Redacción de la memoria.
- Creación de la presentación y defensa del TFM.

Todos estos hitos y actividades del proyecto se encuentran dispuestos en el siguiente diagrama de Gant:

Actividad	Inicio	Final	Días	
Definición de objetivos, plan de trabajo y resultados previstos	**28-09-22**	**10-10-22**	**13.0**	**13.0**
Estado del arte	**11-10-22**	**26-10-22**	**16.0**	**16.0**
Obtención del producto del proyecto	**27-10-22**	**18-12-22**	**53.0**	**106.0**
Implementación del hardware	27-10-22	12-11-22	17.0	
Implementación de la infraestructura	13-11-22	24-11-22	12.0	
Integración	25-11-22	01-12-22	7.0	
Desarrollo de algoritmos y visualización de datos	02-12-22	18-12-22	17.0	
Redacción de la memoria	**19-12-22**	**09-01-23**	**22.0**	**22.0**
Creación de la presentación y defensa del TFM	**10-01-23**	**18-01-23**	**9.0**	**9.0**
				166.0

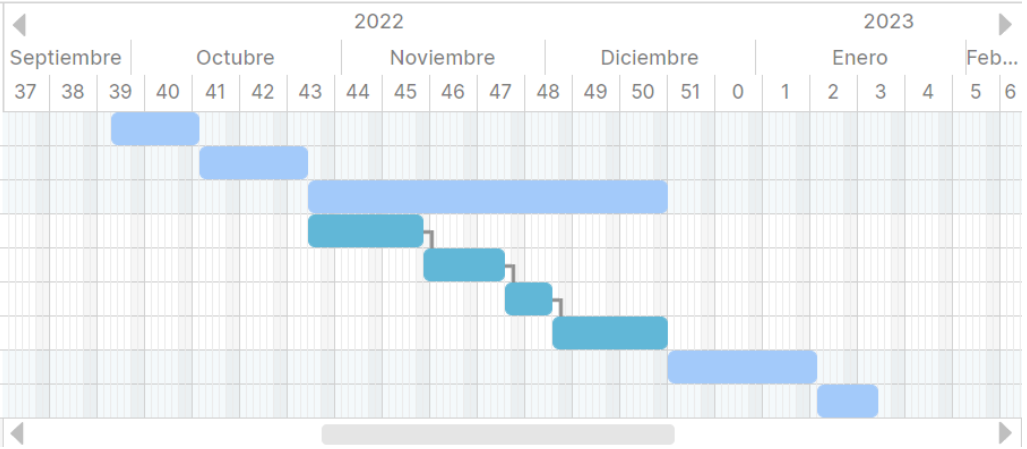


Figura 1: Diagrama de Gantt

1.6. Breve sumario de productos obtenidos

En este apartado se enumeran los productos obtenidos con la realización del trabajo, que, aunque se unifican en el proyecto pueden ser tratados como productos en sí mismos:

- Dispositivo IoT encargado de recoger los datos y parámetros del hogar, y enviarlos para su posterior almacenamiento.
- Infraestructura en la nube que almacena y soporta visualizaciones avanzadas de los datos, así como ejecución de scripts de inteligencia artificial.

Más adelante en la memoria se detallan más detalles de estos productos.

1.7. Breve descripción de otros capítulos de la memoria

La memoria consta de 6 capítulos, un glosario, la bibliografía y un anexo que se describen a continuación:

En el primer capítulo se encuentra la introducción del trabajo donde se comentan el contexto y justificación, los objetivos, el impacto en los objetivos de desarrollo sostenible de la Agenda 20/30, el enfoque y método seguido, la planificación y un breve sumario de los productos obtenidos.

En el segundo capítulo se hace un resumen del estado del arte señalando los campos más relevantes en el ámbito del trabajo, mostrando el estado actual del IoT aplicado a la domótica y las tecnologías de inteligencia artificial, así como un estudio de plataformas cloud.

En el tercer capítulo se hace un planteamiento formal del problema, una justificación del proyecto y se presentan las distintas soluciones propuestas para desarrollar el sistema y se comenta paso a paso la implementación.

En el cuarto capítulo se muestran los resultados obtenidos.

En el quinto capítulo se hace una conclusión de todos los resultados obtenidos y se comentan las posibles líneas futuras con las que poder mejorar los desarrollos.

Y finalmente en el sexto capítulo se muestra el presupuesto desglosando los distintos costes necesarios para la realización completa del trabajo.

2. Estado del arte

En este capítulo se hace un repaso de los diferentes proyectos y técnicas relacionadas con este trabajo. En este caso se estudia lo relativo al IoT y la inteligencia artificial aplicada a la domótica.

2.1. Estado actual del IoT aplicado a la domótica

2.1.1 Productos comerciales

En la actualidad ya se comercializan diferentes dispositivos con los que llegar a convertir una casa en un Smart Home [2] [3]. A continuación, se detallan algunos de los más interesantes:

- Asistentes personales: dispositivos IoT como Alexa o Google Assistant pueden ser los más populares hoy en día ya que están llegando cada vez más a los hogares. A partir del control por voz, permiten reproducir música, programar alarmas, añadir recordatorios y controlar algunos otros dispositivos del hogar que sean compatibles con ellos.

- Enchufes inteligentes: este tipo de dispositivos proporcionan información y control de los aparatos que se conecten al enchufe. Son muy útiles ya que pueden incluso controlarse por voz al integrarse con asistentes de Google o Alexa y permiten hacer un uso más eficiente de la energía reduciendo el consumo en determinados horarios. Pueden integrarse regletas o varios de ellos con aplicaciones que reciben los datos y se controlan mediante wifi, teniendo el control de varios dispositivos de la casa en la mano y pudiendo actuar sobre ellos a distancia mientras se está ausente.

- Bombillas inteligentes: al igual que los enchufes se integran mediante aplicaciones móviles y pueden funcionar con Alexa. Lo interesante de estos aparatos es que pueden variar su intensidad lumínica dependiendo de la situación, lo que también permite un gran ahorro energético.

- Calefacción inteligente: también se pueden introducir sistemas que permitan hacer un uso eficiente de la calefacción, ya que es muy importante en los tiempos que corren si se desea reducir la factura de la luz. A partir de los patrones de las personas, puede disminuir el consumo cuando no están en casa y optimizar la potencia en función de las franjas horarias.

- Cámaras inteligentes: este tipo de cámaras con capaces de detectar situaciones de peligro utilizando inteligencia artificial y notificar al usuario de ello. Son capaces de funcionar incluso en condiciones adversas de luz gracias a la multitud de sensores infrarrojos que incorporan. Además, son capaces de procesar sonidos de los micrófonos que tienen y emitir sonidos para comunicarse con familiares que estén en la casa o para ahuyentar a posibles ladrones.

- Jardinería inteligente: existen equipos que permiten regar jardines o plantas de forma inteligente, ya que a partir de sensores se puede medir la humedad de la tierra y las condiciones meteorológicas para controlar el caudal de agua necesario. También a partir de algoritmos se detectan patrones que ayudan a optimizar el uso de los recursos. Otro ejemplo existente serían los cortacéspedes automáticos.

Todos estos dispositivos comerciales, tienen en común que no se le saca demasiado partido a la inmensa cantidad de datos que recogen, donde la aplicación de algoritmos de inteligencia artificial o aprendizaje automático puede ser clave para dar un salto cualitativo en la calidad de vida de las personas.

2.1.2 Proyectos

Durante los últimos años han aparecido conceptos como DIY ("hágalo usted mismo", del inglés, "Did It Yourself"), que también se aplican en el mundo del IoT y en este caso aplicado a la domótica. En el artículo seleccionado muestran como comunicar Arduino con un servidor vía http [4] con el objetivo de controlar las luces de la casa y automatizar el control del aire acondicionado.

A continuación, se muestra la arquitectura desarrollada por los autores:

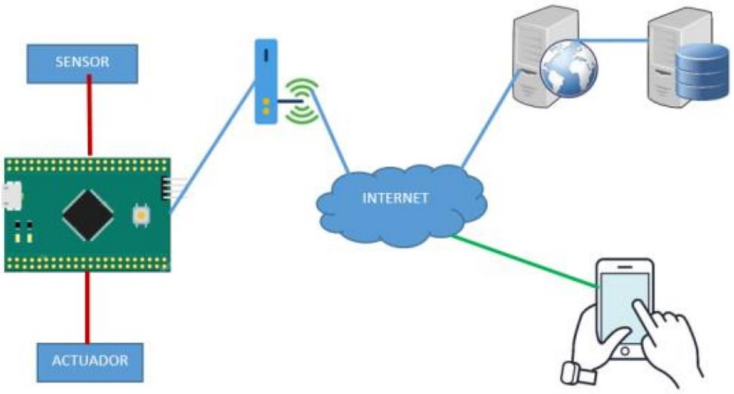


Figura 2: Entorno domótico *[4]*

En este caso utilizan una comunicación maestro-esclavo entre una placa NodeMCU (maestro) y Arduino Uno (esclavo), donde la comunicación se hace a partir de comandos HTTP (GET y PUT). Se utiliza un sensor y un actuador, la información recopilada se envía a un servidor web y se guarda en una base de datos SQL Server. Luego los datos se pueden consumir desde el teléfono por medio de aplicaciones ASP NET, permitiendo controlar el actuador.

Como conclusiones, tuvieron problemas al utilizar un relé con la placa NodeMCU, por lo que tuvieron que emplear Arduino Uno para integrar el dispositivo y recomiendan el uso de servidores gratuitos con hosting PHP ya que hay mayor documentación.

Para este trabajo de fin de máster se desea prescindir del servidor web y hacer uso de la plataforma de Amazon Web Services. AWS tiene un módulo dedicado a la integración con dispositivos IoT llamado AWS IoT Core [5], lo que facilita mucho su uso. Además, eso hace que se puedan integrar fácilmente con otros servicios de la plataforma como Amazon EC2 donde crear instancias de máquinas virtuales y disponer de capacidad de computación en la nube [6].

El buen posicionamiento de AWS en el mercado hace que haya una gran cantidad de documentación en Internet y numerosos tutoriales donde explican como conectar dispositivos como Arduino a AWS IoT [7]. En este caso se ha tomado un ejemplo donde se integra una placa Arduino ESP32, en la cual se recogen datos de humedad y temperatura, y se envían a AWS IoT Core mediante WiFi y MQTT, un protocolo muy utilizado en IoT basado en la publicación y suscripción de mensajes que destaca por ser muy ligero y por requerir de un ancho de banda muy pequeño.

2.2. Estado actual de la IA aplicada a la domótica

La inteligencia artificial (IA) es la ofrecida por las máquinas, convirtiéndose en elementos racionales que son capaces de percibir su entorno y adaptarse con el objetivo de ofrecer la máxima eficiencia a la hora de realizar una tarea. En este sentido imitan a las funciones cognitivas humanas, pero en este caso realizando tareas sencillas.

Nils J. Nilsson [8] hace referencia a los siguientes cuatro pilares básicos en los que se apoya la inteligencia artificial:

1. *Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles.*

2. *Algoritmos genéticos, similares al proceso evolutivo, donde las mejores soluciones continúan y las peores se descartan.*

3. *Redes neuronales artificiales, con una arquitectura donde su funcionamiento conceptualmente hablando es análogo al funcionamiento físico del cerebro de animales y humanos, donde las conexiones son similares a la sinapsis entre neuronas.*

4. *Razonamiento lógico comparable a la forma de pensar abstracta del cerebro humano.*

Una de las ramas más importantes de la IA es el aprendizaje máquina (del inglés, *machine learning*) cuyo objetivo principal es implementar técnicas que consigan que las computadoras aprendan, creando programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos o datos de entrada. Para ello, el aprendizaje máquina tiene como resultado un modelo que trata de resolver un problema o realizar una tarea. Los algoritmos de aprendizaje máquina pueden clasificarse en:

- Modelos geométricos: se construyen usando elementos geométricos como líneas o planos, pudiendo ser de una o varias dimensiones. Son muy útiles cuando los datos siguen una estructura geométrica. Este sería el caso de una regresión lineal, donde los datos pueden aproximarse a una recta.

- Modelos probabilísticos: determinan una función distribución enlazando los datos del conjunto de entrada con los valores definidos en la salida.

- Modelos lógicos: evalúan las probabilidades de que sea correcta una salida por un sistema de reglas estructurado jerárquicamente. Es el caso de los árboles de decisión.

Los diferentes algoritmos de aprendizaje máquina también pueden agruparse en función de si se conoce la salida de estos. Existiendo dos tipos, aprendizaje supervisado y no supervisado.

En algoritmos de aprendizaje supervisado se conocen las salidas de las muestras, donde éstas están debidamente etiquetadas y llevan a producir una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Un ejemplo de este tipo de algoritmo es un problema de clasificación, donde el algoritmo de aprendizaje trata de etiquetar o clasificar una serie de vectores, conociéndose una o varias salidas reales de las muestras.

Por otro lado, en los algoritmos de aprendizaje no supervisado, no se conoce la salida real de las muestras, de modo que el proceso de modelado de algoritmos se lleva a cabo partiendo tan solo de las entradas al sistema y no se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas utilizando algoritmos de agrupamiento (del inglés, *clustering*).

En este trabajo se quiere hacer uso de dispositivos IoT e inteligencia artificial, por lo que es necesario mostrar el estado actual de las tecnologías y últimos avances donde se combinen ambas áreas. Para ello se ha seleccionado un trabajo de fin de grado que trata sobre el diseño e implementación de una aplicación domótica para iluminación de la vivienda utilizando inteligencia artificial [9].

El objetivo principal del trabajo era optimizar la iluminación de la vivienda para reducir el consumo de energía y así lograr una mayor eficiencia sin dejar de lado el confort del usuario. Para ello implementaron un sistema de inteligencia artificial basado en redes neuronales multicapa, donde los datos de entrada eran enviados por sensores de proximidad y lumínicos, y para medir el nivel de confort se realizaban encuestas a los convivientes de la casa.

En este caso utiliza una red neuronal con varias capas y para calcular el valor de los pesos de cada una emplea el algoritmo retropropagación (del inglés, *backpropagation*), ya que se trata de un sistema no lineal y para poder entrenar esta red se necesitó que fuese supervisado. Este algoritmo es cíclico, donde a partir de unos datos de entrada como estímulo optimiza la función de coste por medio del método de descenso por gradiente. Esto quiere decir que la salida se compara con la salida deseada y se genera una señal de error por cada una de las muestras del conjunto de entrenamiento. Estas salidas de error se propagan hacia atrás desde la capa de salida, pasando por las neuronas de la capa anterior. Estas, reciben una parte de la señal total del error, lo correspondiente a lo aportado por cada neurona hacia la salida. De modo que se repite este proceso por cada capa, hasta llegar al inicio de la red. De esta forma se ajustan los pesos.

Por otro lado, en función de los datos de proximidad y datos lumínicos, por medio de actuadores conseguían variar la intensidad de luz del 0% al 100% además de poder variar el color de la luz emitida por las bombillas ya que podían conseguir hasta 16 millones de colores distintos pudiendo adaptarse a todo tipo de ambientes para asegurar el confort.

Finalmente, se consiguió una tasa de error del 5%, logrando un confort del 97% en los usuarios, por lo que se puede considerar un éxito. Establecieron las líneas futuras de este trabajo en la posibilidad de instalar cámaras y añadir más sensores al sistema para controlar niveles de temperatura y humedad entre otros.

A continuación, en el siguiente trabajo seleccionado [10] se hace un estudio de los principales algoritmos de aprendizaje máquina y sus diferentes aplicaciones, donde hay una sección dedicada al IoT, algo muy interesante ya que tiene relación directa con este trabajo de fin de máster y puede servir para contextualizar y aportar información. Por lo tanto, algunos de los algoritmos que pueden aplicarse en IoT se listan a continuación:

- Algoritmos de aprendizaje supervisado:

- Regresión lineal: es una técnica de estimación, donde se relacionan los datos de entrada y de salida, donde la relación entre la variable dependiente e independiente es lineal.

- Regresión logística: se aplica en problemas de clasificación binaria donde hay relación entre la variable dependiente y una o más variables independientes. Emplea una función logística para calcular una probabilidad de pertenecer a una clase u otra, donde es necesario establecer un umbral.

- Máquina de vectores soporte (SVM, del inglés, *Support Vector Machine*): clasifica datos encontrando el hiperplano de máximo margen que separa las clases en un problema de clasificación.

- Naïve Bayes: utiliza clasificadores probabilísticos simples y se emplea en casos donde supuestamente todas las características de los datos son independientes entre sí.

- K vecinos más cercanos (KNN, del inglés, *K Nearest Neighbors*): en este algoritmo las observaciones del conjunto de test son clasificadas según la clase mayoritaria de las K muestras de entrenamiento más próximas a la muestra de test, aplicándose por tanto una función distancia entre los datos. De forma que se calcula la distancia entre el elemento de test y cada uno de los datos de entrenamiento, ordenando de mayor a menor esos datos, y la clasificación de la muestra vendrá dada por el grupo más frecuente de los K primeros en esa ordenación, y en caso de empate, se elige uno de forma aleatoria.

- Algoritmos de aprendizaje no supervisado:

  - Clustering: trata de encontrar una estructura o patrón entre los datos no etiquetados agrupando en un número de clústeres. En este algoritmo, al igual que en el KNN se aplican funciones de distancia con el objetivo de determinar los grupos de datos más cercanos y similares entre sí.

  - Cuantificación vectorial: de forma similar al anterior, los datos se intentan organizar en vectores representados por sus centroides. Para ello, se emplea el algoritmo K-means.

- Algoritmos de aprendizaje profundo (del inglés, *deep learning*): se utilizan redes neuronales para resolver problemas que no son linealmente separables. Estas redes están formadas por capas de neuronas, donde cada una de ellas tiene una función de activación no lineal y entre ellas forman una relación entre los datos de entrada y de salida. Cabe destacar que las neuronas de cada capa no pueden conectarse con las de su misma capa.

2.4 Plataformas cloud para IoT

Elegir una buena plataforma cloud para el trabajo es muy importante, por lo que se ha tenido que hacer un estudio de todas las existentes en el mercado y ver cual se ajusta mejor. Para ello, se ha tenido en cuenta el trabajo de Rodrigo Martínez [11] y en la siguiente tabla se muestran una comparativa de las plataformas que podrían encajar en este proyecto, ya que algunas se han descartado por ser solo compatibles con algunos casos de uso muy concretos:

Plataforma	Lenguajes compatibles	Protocolos compatibles	Ventajas	Inconvenientes
AWS IoT	C, JavaScript, Java, Python, iOS, Android, C++	MQTT, HTTP	Tarifas muy asequibles y ofrece integración con todos los servicios de su plataforma	Hay que mantenerse al día con la documentación ya que hay avances constantemente
Azure IoT	C, Python, Node.js, Java, .NET	MQTT AMQP HTTP	Muy completa por los servicios que ofrece y buena interacción con el dispositivo	Tarifas elevadas y poca flexibilidad en ellas
Oracle IoT	Android, C, iOS, Java SE, JavaScript	MQTT, HTTP	Fácil conexión a dispositivos y buenas herramientas de visualización	Servicios limitados en comparación con los competidores
Watson IoT	Node.js, Java, Python, C#, C, C++	MQTT, HTTP	Muy completa en el análisis de datos y machine learning	No tiene un servicio propio de visualización de datos y en general ofrece pocas herramientas
Xively	iOS Android	MQTT WebSocket HTTP	Muy sencilla de gestionar y operar con dispositivos	Solo ofrece servicios de recolección, análisis y visualización básicos
Samsung Artik	C C++ Node.js	HTTP WebSockets MQTT CoAP	Muy fácil de conectar a sus propias placas de desarrollo y buenas integraciones con herramientas externas	Proceso de configuración muy cerrado, poco flexible y económica

Finalmente, la opción aconsejada por el autor es AWS IoT, principalmente por el gran número de servicios que ofrece en su plataforma, por su flexibilidad y la buena oferta de tarifas. Por lo que de cara a un trabajo como el presente es una opción más que recomendable.

3. Solución propuesta

3.1. Planteamiento formal del problema

Como se comentaba en el capítulo introductorio, se desea crear un dispositivo capaz de realizar mediciones en el entorno del hogar, recoger los datos, enviarlos y almacenarlos. Todo ello debe estar integrado con una infraestructura en la nube que soporte el almacenamiento y ofrezca la posibilidad de visualizar los datos, así como de ejecutar algoritmos de inteligencia artificial. Una vez el dispositivo sea completamente funcional debe ayudar en la toma de decisiones del usuario final.

Para satisfacer estas necesidades, se ha desarrollado la siguiente arquitectura:

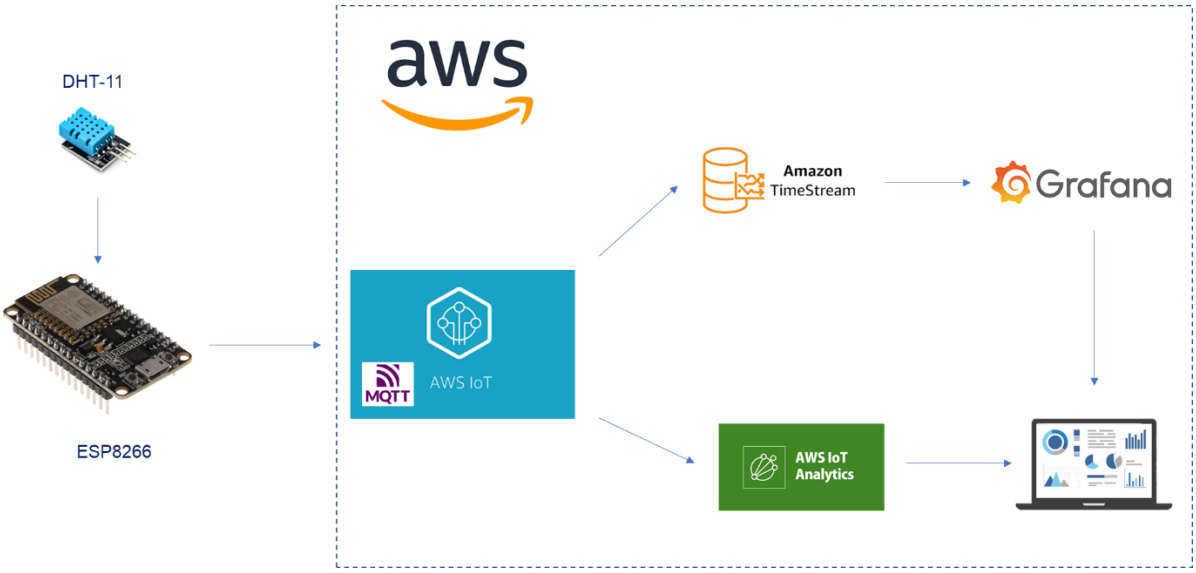


Figura 3: Arquitectura de la solución

A continuación, se detalla brevemente la funcionalidad de cada elemento del esquema:

- DHT-11: dispositivo conectado a la placa ESP8266 que integra sensores de temperatura y humedad.
- ESP8266: placa compatible con el Arduino IDE que recibe las mediciones de temperatura y humedad. Tiene wifi integrado utilizado para enviar los datos a la nube de Amazon Web Services.

- AWS IoT Core: servicio de AWS empleado para recibir información de dispositivos IoT. Tiene integrado el protocolo MQTT para recibir mensajes con los datos medidos y mediante un sistema de reglas, se reenvían los datos medidos hacía Amazon TimeStream y AWS IoT Analytics.

- Amazon TimeStream: servicio de almacenamiento que guarda datos de series temporales.

- Grafana: herramienta de visualización de datos y creación de dashboards.

- AWS IoT: módulo muy completo de AWS que integra servicios de almacenamiento, canalización de información, visualización y de ejecución de algoritmos de IA entre otros.

3.2. Justificación de la solución

En este apartado se realiza una justificación de la solución elegida. Para ello, se muestran una serie de dificultades que se han ido encontrando durante el desarrollo del trabajo y que son muy importantes a la hora de justificar las elecciones de los elementos que componen la solución.

En primera instancia, se contaba con una placa Arduino Uno con la idea de conectar un módulo wifi ESP8266-01 y montar un circuito similar al de la siguiente figura:

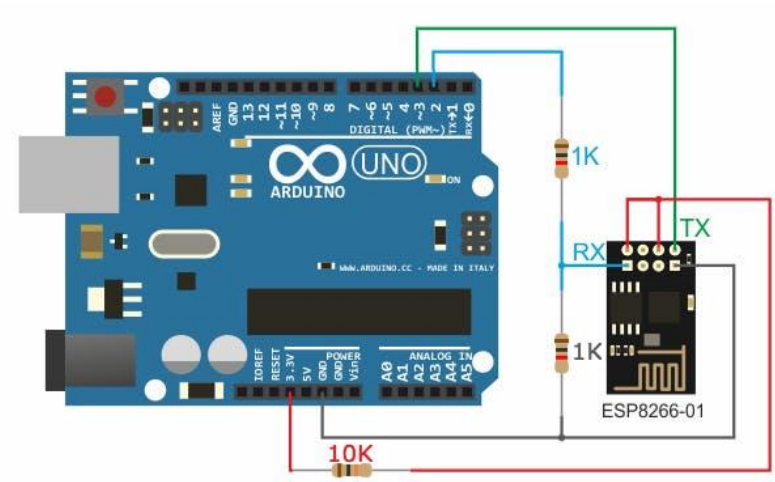


Figura 4: Conexión WIFI Arduino Uno *[12]*

Después de seguir investigando esta posible solución, se terminó descartando ya que había una gran cantidad de usuarios que tenían problemas con la configuración de la red wifi

después de implementar el código y cargarlo en la placa, donde tenían que configurar módulo ESP8266 por medio de comandos AT y no aseguraba la compatibilidad.

Por ello, se optó por buscar alguna placa que tuviese un módulo wifi integrado y fuese compatible con Arduino. Principalmente, se encontraron placas con el módulo ESP32 y el ESP8266 ya integrados, donde la segunda opción, tanto por especificaciones como por rango de precio, era más recomendable.

Por lo tanto, se compró el modelo NodeMCU Lolin V3 Modulo ESP8266 CP2102 ESP-12E:

Figura 5: NodeMCU Lolin V3 Modulo ESP8266

Sin embargo, por problemas de suministro se tuvo que cancelar el pedido y buscar un dispositivo similar que pudiese entregarse a tiempo. Finalmente, la placa elegida y empleada en el proyecto fue la HiLetgo ESP8266 NodeMCU CP2102 ESP-12E:

Figura 6: HiLetgo ESP8266 NodeMCU CP2102 ESP-12E

Por otro lado, se ha elegido el dispositivo DHT-11 ya que incorpora dos sensores, uno de temperatura y otro de humedad. Si bien es cierto que disponer de un gran número de medidas no es uno de los objetivos de este trabajo, disponer de al menos dos datos distintos es bastante útil y este elemento permite recibirlos muy fácilmente. Además, son dos parámetros muy importantes de cara a la monitorización del hogar ya que son claves a la hora de establecer un nivel de confort. Concretamente, el lugar donde se han realizado las

mediciones es un piso que no cuenta con buen aislamiento térmico y tiende a acumular bastante humedad en el ambiente, por lo que, si se realiza un buen uso de los datos, de cara a líneas futuras (como se comenta más adelante) puede ser interesante el uso de actuadores para controlar la calefacción y deshumidificadores para mantener buenos niveles de temperatura y humedad.

En cuanto a la elección plataforma en la nube, se tenido en cuenta lo mencionado en el apartado 2.4, optando por Amazon Web Services y por tanto por AWS IoT como servicio de IoT.

Dentro de la plataforma, de cara a aportar valor a los datos recibidos por MQTT en IoT sí que hay que tener cuidado a la hora de elegir los servicios de almacenamiento y analítica de datos, pero sobre todo hay que tener precisión al elegir la ubicación del servidor. Es muy importante que el servidor elegido dentro de la plataforma tenga disponible todos los servicios que se quieran utilizar ya que se puede dar el caso que un determinado servicio se pueda utilizar en unas ubicaciones, pero no en otras.

Durante el desarrollo de este proyecto, puesto que no se tenía claro en un principio todos los servicios implicados, se comenzó la configuración de AWS IoT en la región *eu-west-3* perteneciente a París con la idea de almacenar los datos en el servicio de almacenamiento no estructurado DynamoDB. Pero surgieron problemas durante la implementación de las reglas para reenviar los datos medidos y almacenarlos en esta base de datos, por lo que hubo que descartar este servicio. Más tarde, se vio que para este caso de uso era más adecuado utilizar los servicios Amazon TimeStream y AWS IoT Analytics.

El problema surgió cuando se vio que en la región de los servidores que se estaban utilizando no estaban disponibles estos servicios, por lo que hubo que repetir la configuración en una región donde todos los servicios estuviesen disponibles, que en este caso se optó por *eu-west-1* perteneciente a Irlanda.

Finalmente, como se ha indicado, los servicios de almacenamiento elegidos son Amazon TimeStream [13] y AWS IoT Analytics [14]. El primero de ellos se ha elegido por ser una opción muy eficiente, escalable y que encaja muy bien en el caso de uso del proyecto ya que es un servicio enfocado en series temporales. AWS IoT Analytics se ha elegido porque integra un producto que engloba desde el almacenamiento hasta la ejecución de análisis de datos y está expresamente dedicado al IoT. Por otro lado, se ha elegido Grafana [15] como herramienta de visualización de datos y creación de cuadros de mando ya que se conecta fácilmente a Amazon TimeStream y permite ver los datos medidos casi en tiempo real.

Cabe destacar que todos estos servicios empleados están administrados completamente por la plataforma dejando al usuario libre de tareas de gestión de la plataforma cloud, lo que facilita mucho su uso.

3.3. Implementación

Después de justificar la solución elegida, en este apartado se muestra todo el proceso de implementación de la solución final elegida.

3.3.1 Implementación hardware

El primer paso en este proyecto fue realizar el montaje del sistema de recogida de datos. Como se ha adelantado, para esta parte se han necesitado tan solo dos elementos, el módulo wifi ESP8266 compatible con Arduino y el sensor de temperatura y humedad DHT11. Para ello, se ha utilizado la documentación proporcionada en *How to Electronics* [16], una web que tiene documentación de todo tipo de proyectos IoT y que en este caso mostraban el proceso para conectar este hardware con AWS IoT.

A continuación, se muestra el esquema seguido para el montaje:

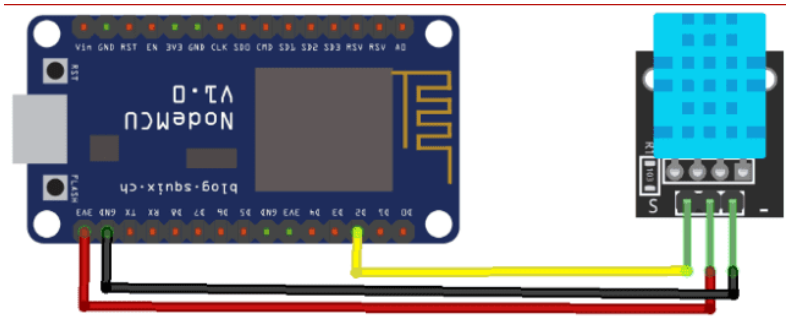


Figura 7: Esquema del montaje

Como se observa en el diagrama, simplemente hay que conectar los pines VCC y GND del sensor a los pines de 3.3 V y masa de la placa, así como el pin de señal del sensor al pin D2 de la placa. Posteriormente, habrá que conectar el puerto micro USB de la placa al ordenador y empezar a programar el código C++ en el Arduino IDE. Seguidamente en las siguientes figuras se muestran imágenes del hardware real montado:

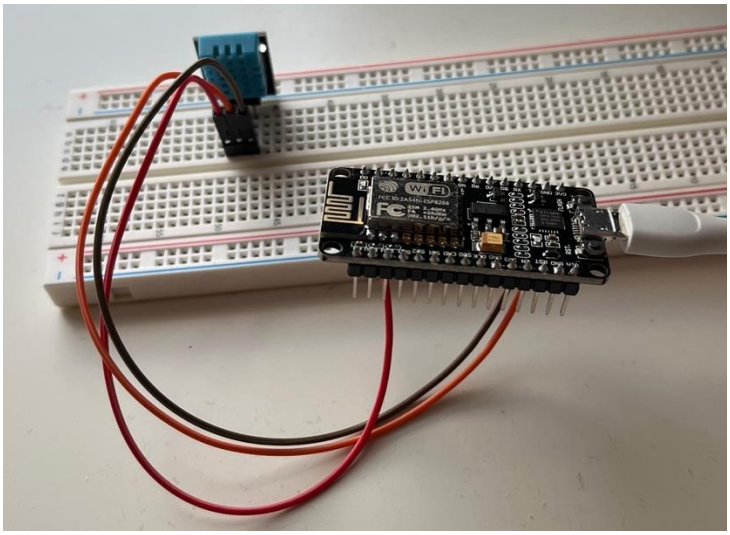


Figura 8: Montaje hardware (1/5)

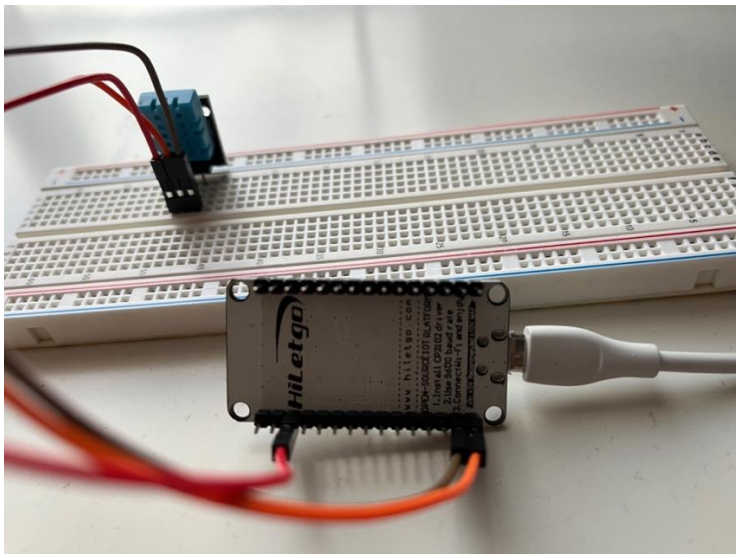


Figura 9: Montaje hardware (2/5)

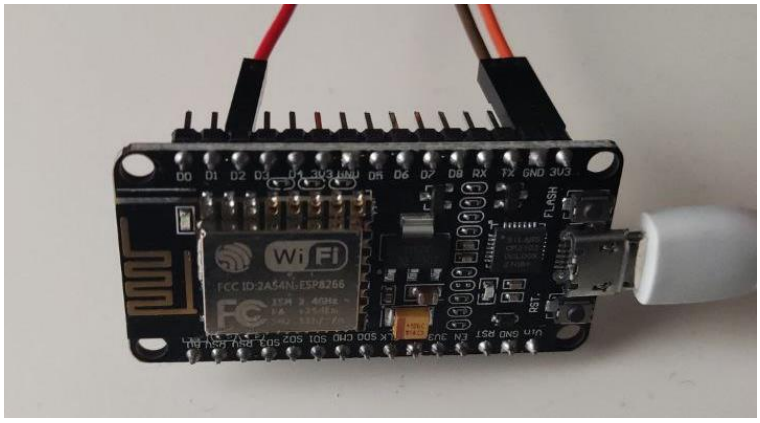


Figura 10: Montaje hardware (3/5)

Figura 11: *Montaje hardware (4/5)*

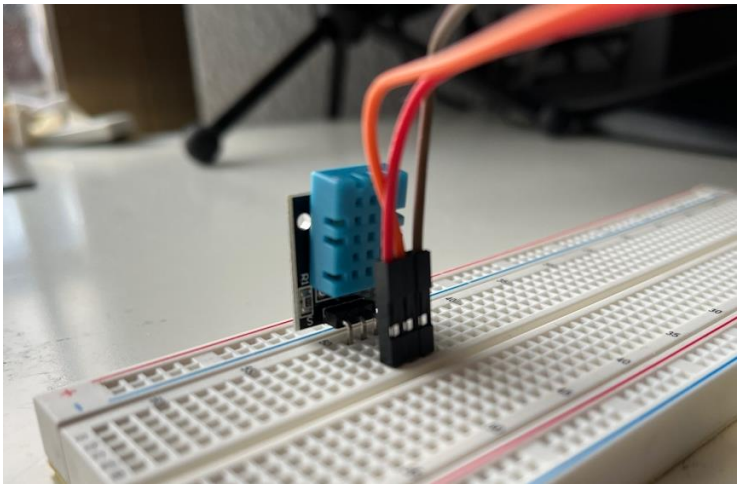


Figura 12: Montaje hardware (5/5)

3.3.2 Configuración de AWS IoT Core

Antes de crear el proyecto en el Arduino IDE, es necesario disponer de una cuenta en AWS y haber creado un objeto en AWS IoT Core para poder comunicar el dispositivo con la nube. Para ello, se ha seguido el proceso indicado en la documentación [16] mencionada y se muestran a continuación los pasos realizados en este proyecto.

Primero, como se comentaba en la justificación, es necesario elegir una región en AWS que tenga disponibles todos los servicios que se vayan a utilizar, en este caso se ha elegido *eu-west-1*, perteneciente a Irlanda. Después, dentro del servicio de AWS, hay que crear un objeto para el dispositivo que se desea conectar como el que se muestra en la siguiente figura:

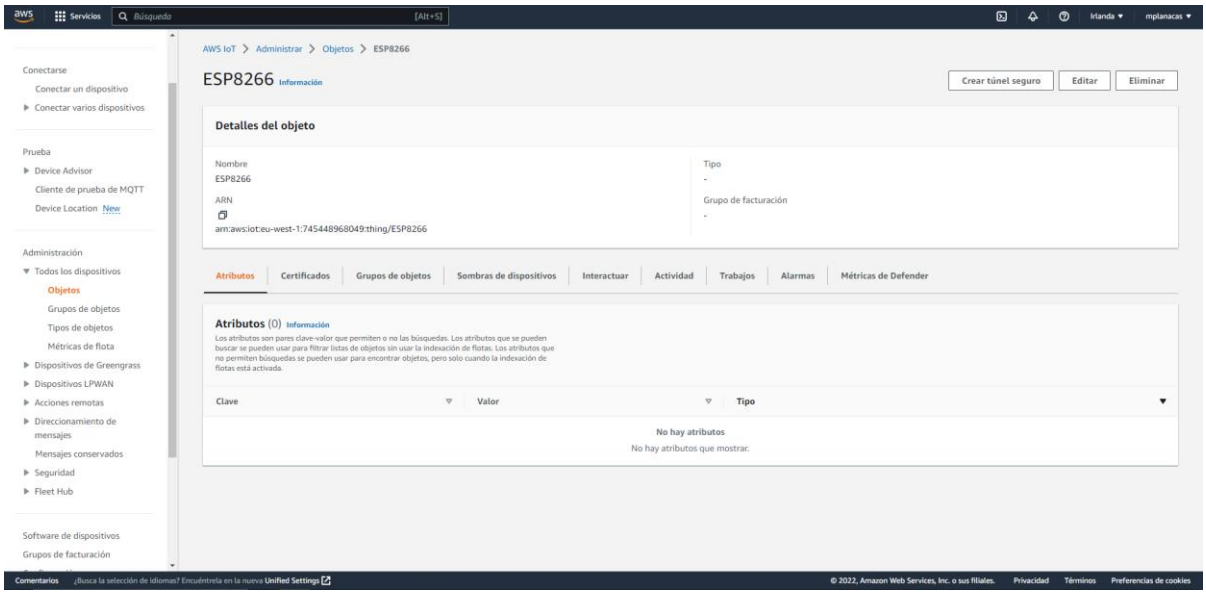


Figura 13: Objeto AWS IoT Core

Para que se permita la comunicación por MQTT del dispositivo, es necesario crear una política de seguridad y asociarla al objeto en el momento de su creación. En este caso, la política se crea para permitir las acciones de conectar, publicar, subscribir y recibir mensajes:

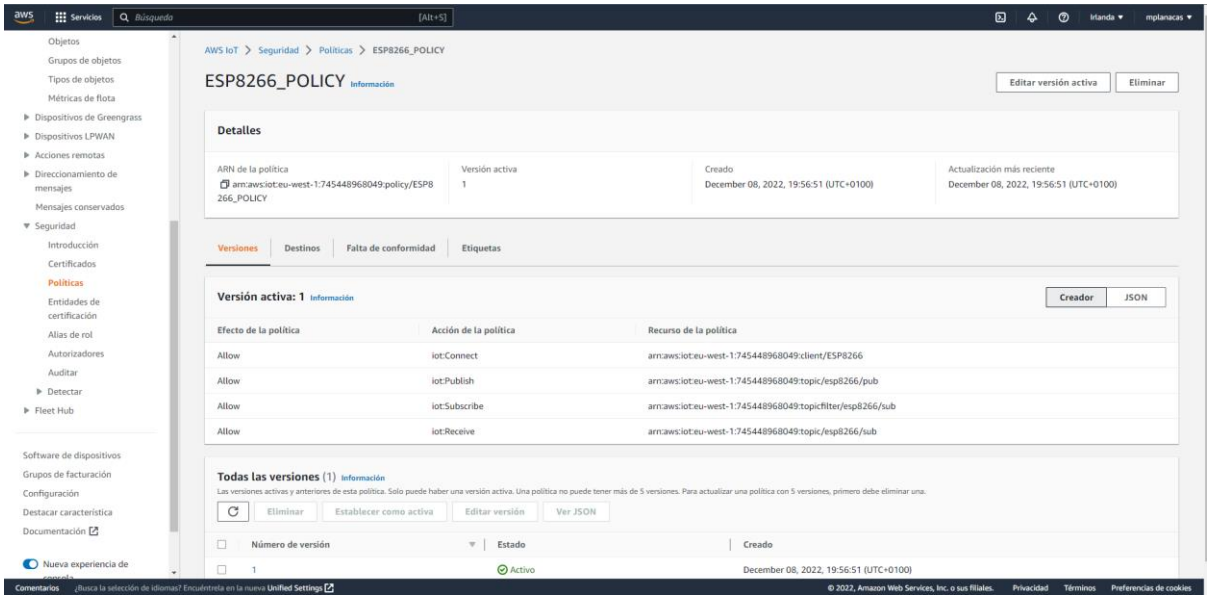


Figura 14: Política AWS IoT Core

Finalmente, después de la creación del objeto, es necesario descargar tres certificados de seguridad que se necesitan en el código que se carga en la placa ESP8266 a través del Arduino IDE. Estos tres certificados son la clave pública, la clave privada y el certificado Root CA1.

3.3.3 Implementación del código

Para la implementación del código, se han utilizado las plantillas proporcionadas en la documentación mencionada [16]. El proyecto está divido en dos ficheros, por un lado, está el main Arduino File, que en este caso se ha llamado *ESP8266_AWS* y, por otro lado, el fichero *secrets.h*. El primero de ellos contiene la definición de los pines que se utilizan, así como toda la funcionalidad necesaria para poder enviar los mensajes a AWS por medio de la red Wifi de la casa. El segundo fichero contiene las claves que se han descargado a la hora de crear el objeto en AWS IoT Core, y funciona mediante un cifrado RSA, la clave de la red Wifi y un identificador del host MQTT.

El código completo puede consultarse en el Anexo 9.1 Código proyecto Arduino.

3.3.4 Uso cliente de prueba de MQTT

Para monitorizar los mensajes enviados desde la placa ESP8266 a la cuenta de AWS, se hace uso del cliente de prueba de MQTT. En este caso, se ha configurado para que los mensajes lleguen a través del topic “esp8266/pub”, por lo que se empezarán a recibir los mensajes subscribiéndose a dicho topic tal y como se puede observar en la siguiente figura:

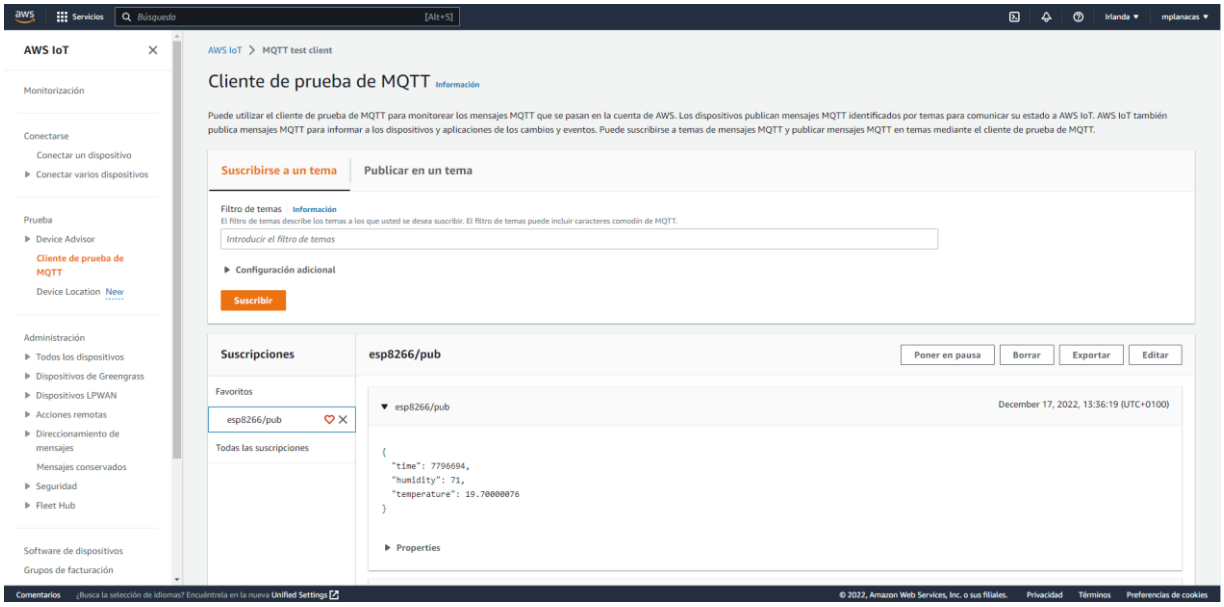


Figura 15: Cliente de prueba MQTT

A continuación, se muestran los últimos mensajes recibidos en el momento de la captura donde se pueden ver los datos de temperatura y humedad:

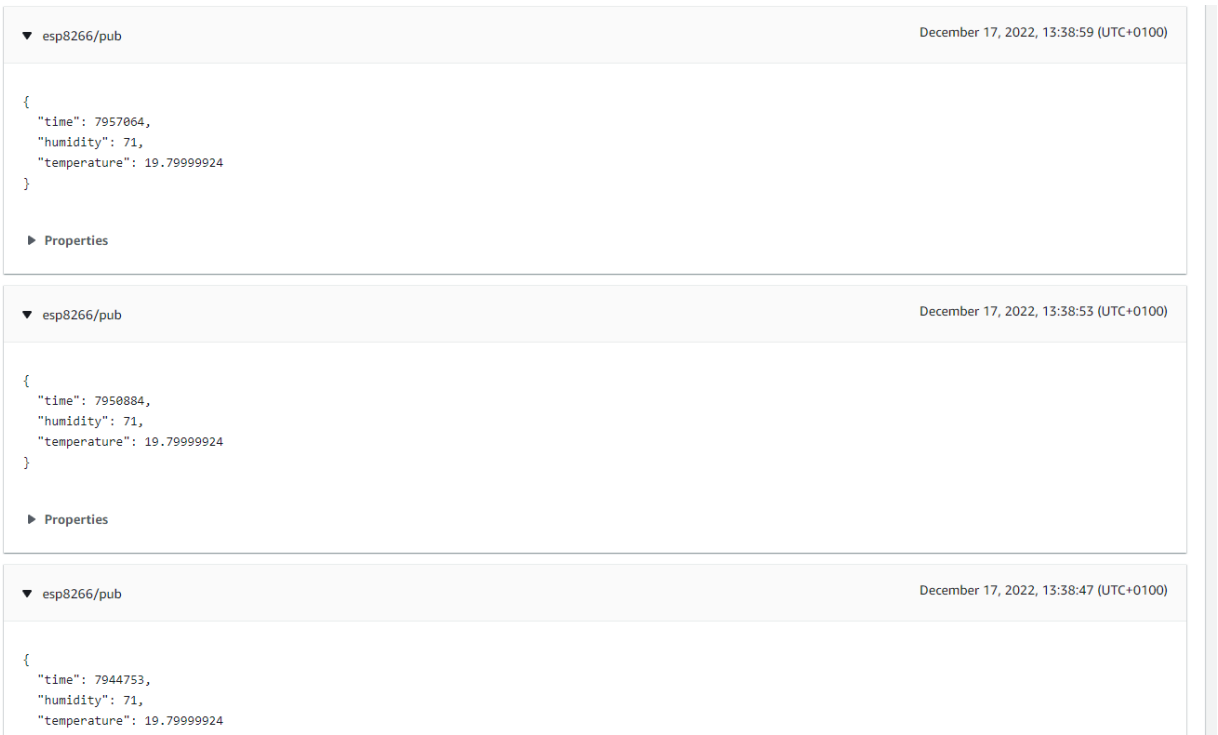


Figura 16: Mensajes recibidos por MQTT

Cabe destacar que con la opción publicar en un tema, se pueden enviar mensajes desde el cliente MQTT y aparecen por la consola del Arduino IDE, aunque más allá de realizar alguna prueba no tiene mayor relevancia actualmente en el proyecto. Si en el futuro se añadiesen actuadores a la placa, en ese caso sí que cobraría sentido el uso de la publicación de mensajes hacia la placa ESP8266.

3.3.5 Configuración de servicios de almacenamiento

Los datos recibidos a través del cliente de MQTT si no se hace nada con ellos se perderían, por lo que es necesario crear reglas que reenvíen estos datos a los servicios de almacenamiento de datos, pero antes de crear estas reglas los servicios deben estar activos y configurados. Por lo que a continuación se muestra la configuración de los servicios Amazon TimeStream y AWS IoT Analytics.

Primero, se creó dentro del servicio de Amazon TimeStream una base datos llamada *IoT_bbdd*:

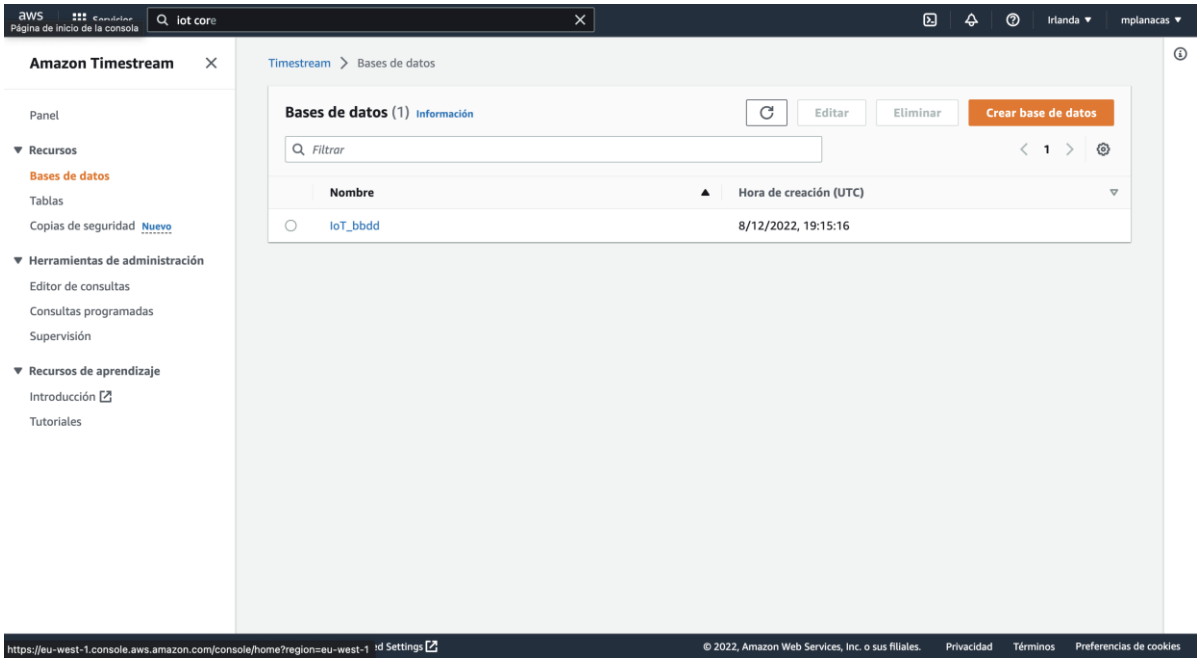


Figura 17: BBDD Amazon TimeStream

Dentro de esta base de datos, fue necesario crear una tabla llamada *ESP8266*. El proceso de creación de estas tablas es muy rápido y sencillo, ya que no funciona como una base de datos estructurada convencional. En este caso, no es necesario indicar la estructura de la tabla, ya que el número, nombre y tipo de datos de las columnas se genera a partir de la regla que se cree posteriormente para el envío de los datos de la serie temporal. A continuación, se muestra una imagen con información de esta tabla en el servicio:

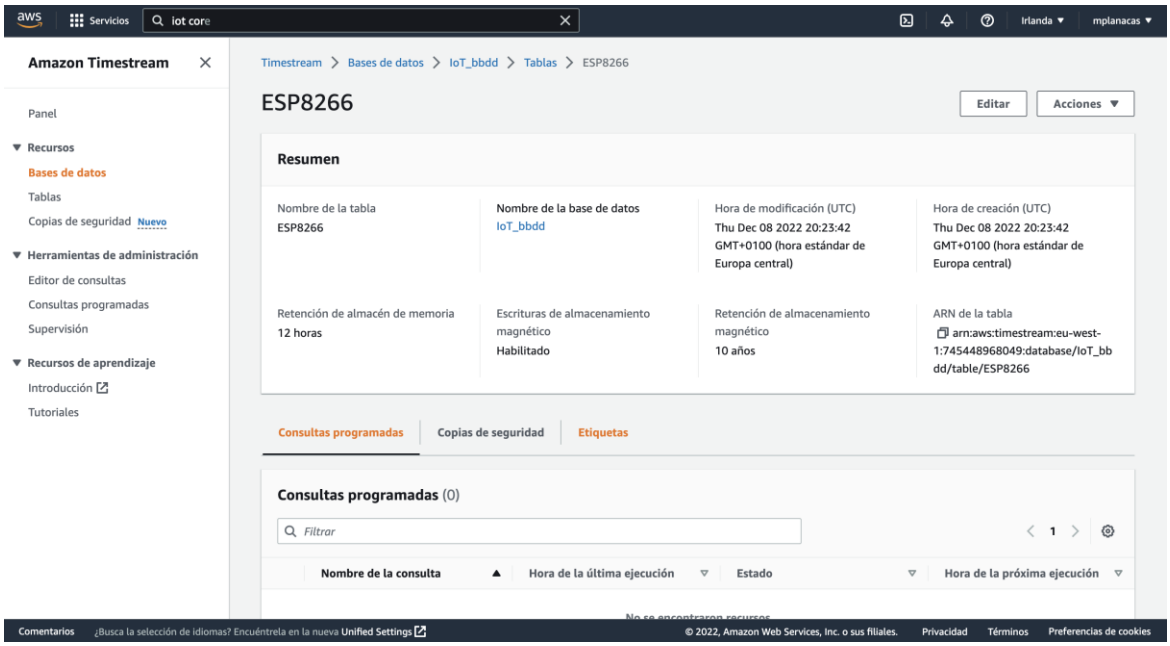


Figura 18: Tabla BBDD Amazon TimeStream

Paralelamente, de cara a disponer de un servicio con el que poder ejecutar scripts de inteligencia artificial como se ha comentado anteriormente, se ha configurado el servicio IoT Analytics. Para ello, en la ventana mostrada en la siguiente figura, simplemente hay que indicar un prefijo de recursos para describir el proyecto y un tema de MQTT. Una vez indicados dos campos, pasados unos minutos el servicio crea automáticamente un canal, una canalización, un almacén de datos, un conjunto de datos asociados al tema de MQTT, que en este caso se ha indicado “esp8266” y una instancia de bloc de notas:

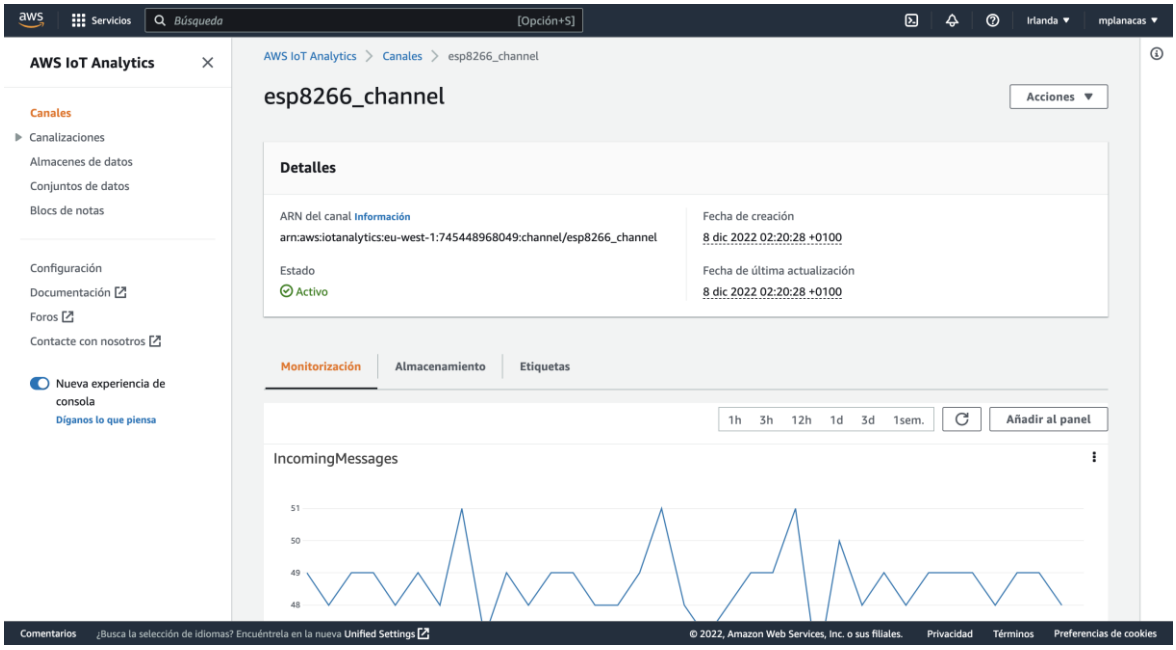


***Figura 19**: Canal IoT Analytics*

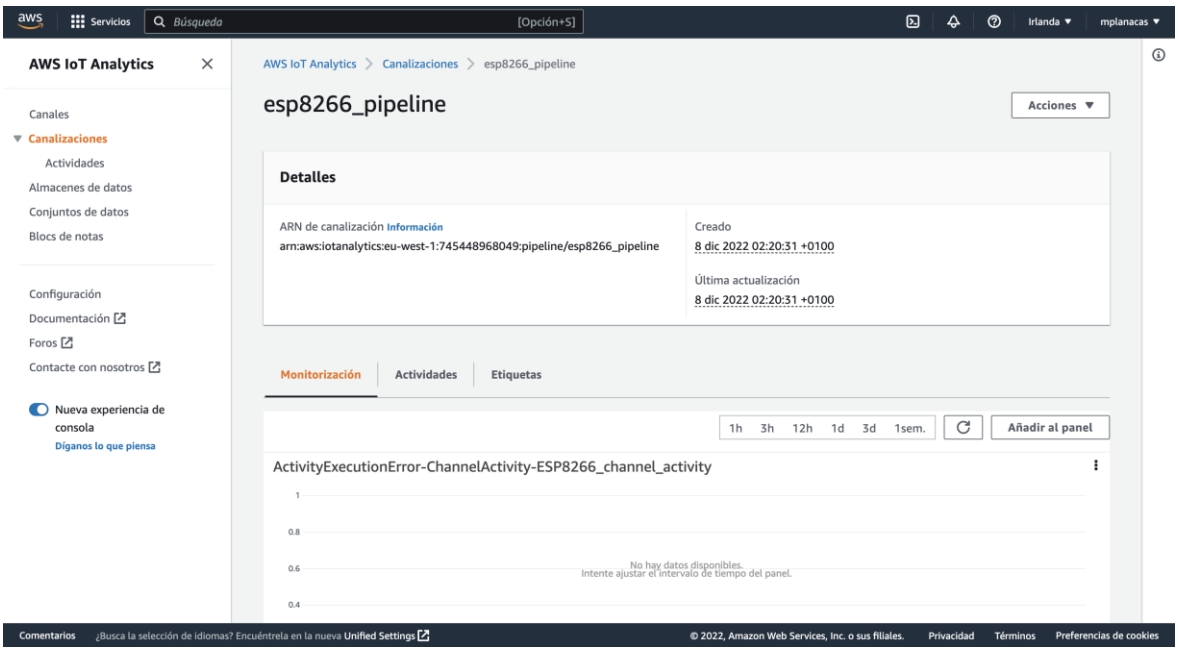


Figura 20: Canalización IoT Analytics

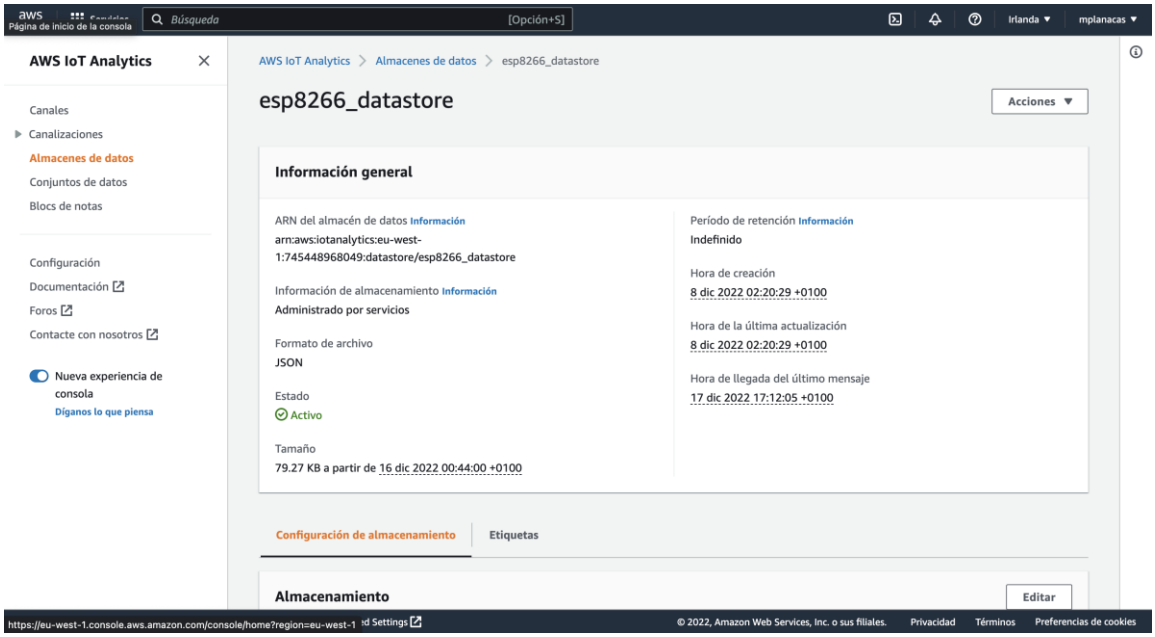


Figura 21: Almacén de datos IoT Analytics

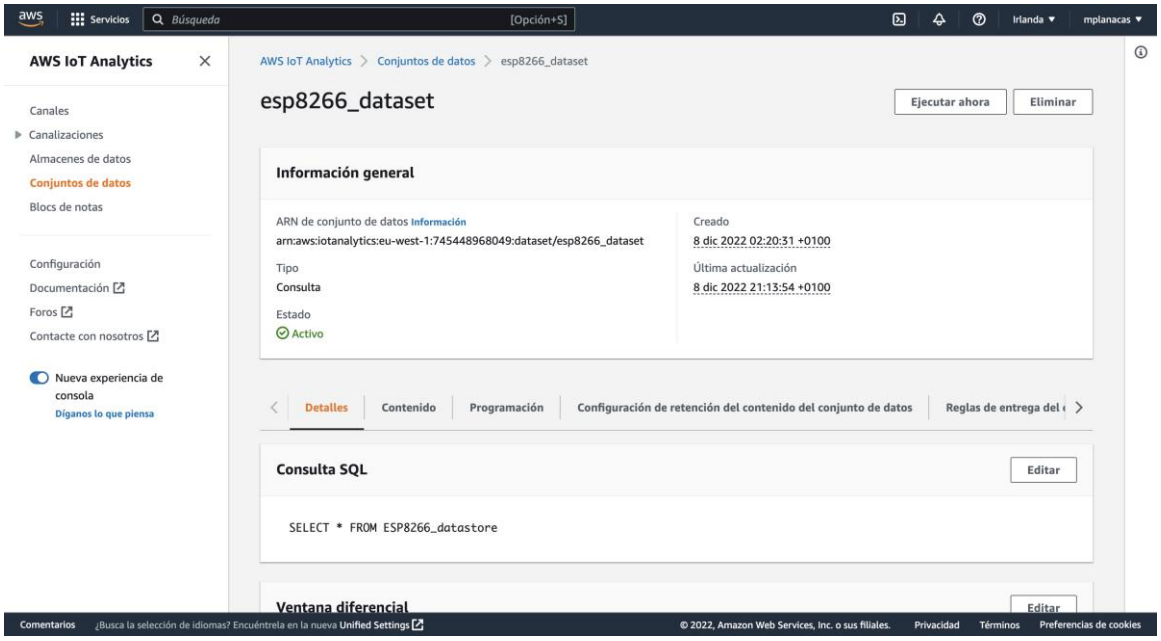


Figura 22: Conjunto de datos IoT Analytics

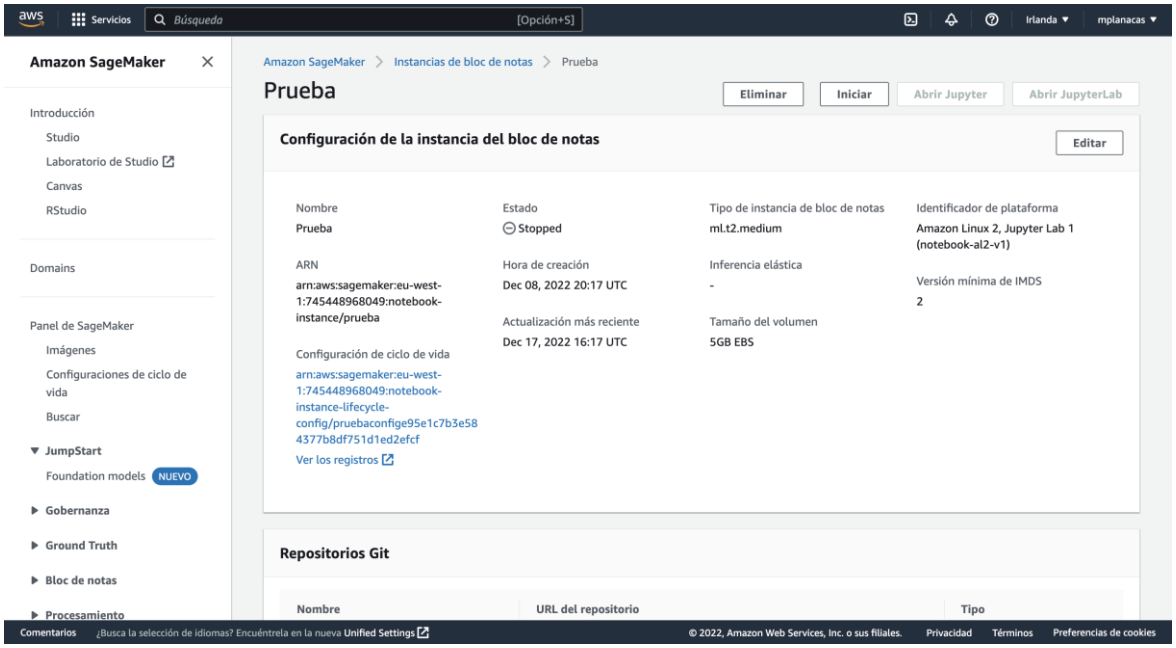


Figura 23: *Instancia de bloc de notas Amazon SageMaker*

Pese a que todos los recursos tienen aplicaciones muy interesantes, de cara a este proyecto los más relevantes son el conjunto de datos y la instancia de bloc de notas. El conjunto de datos da forma a información reenviada por la regla que se comentará en el siguiente apartado y la prepara para distintas aplicaciones. En este caso, este conjunto de datos puede consultarse a partir de un Jupyter Notebook disponible en la instancia de bloc de notas mencionada, para posteriormente ejecutar algoritmos de inteligencia artificial o de análisis de datos.

Como se puede observar, el servicio de Jupyter correspondiente a la instancia de bloc de notas llamado “Prueba”, aparece en estado parado. Esto se debe a que se hizo alguna prueba y seguidamente se vio que era un servicio de pago y estaban cobrando por cada ejecución. Este aspecto se comentará más adelante.

3.3.6 Configuración de reglas

Una vez se tienen disponibles los servicios de almacenamiento de datos, como se ha anticipado, es necesario disponer de reglas que reenvíen los datos recibidos por el cliente de MQTT. Para ello, hay que volver al servicio AWS IoT Core, al apartado de direccionamiento de mensajes y crear las reglas correspondientes.

A continuación, se muestran imágenes de ambas reglas:

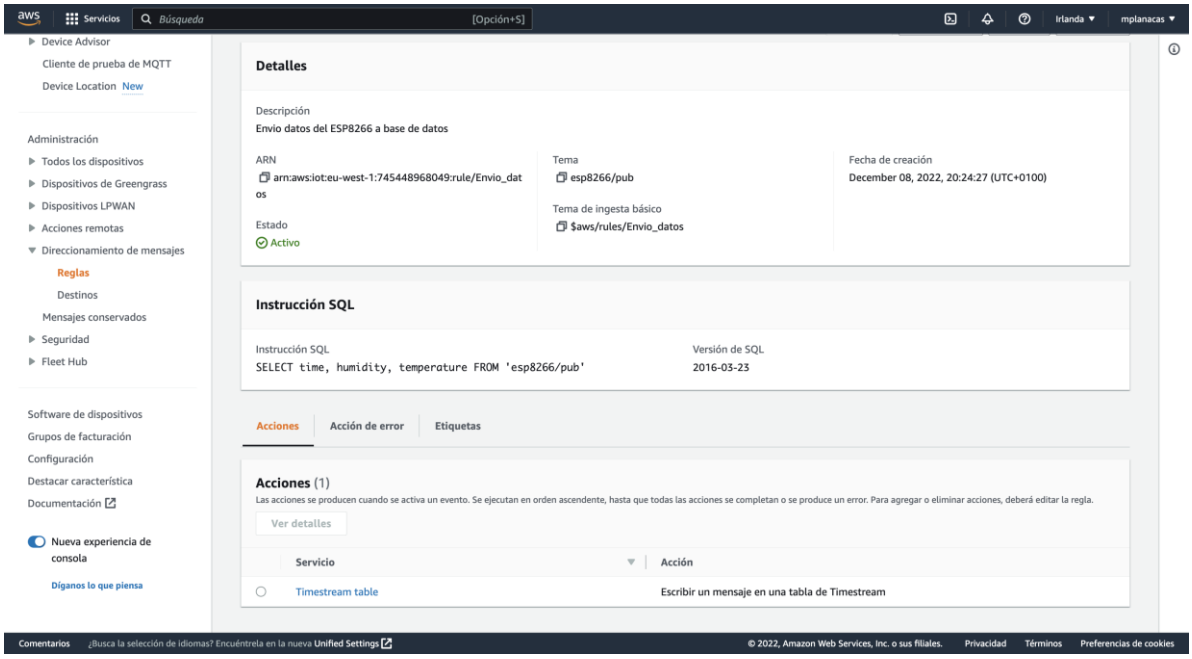


Figura 24: Regla envío de datos a Amazon TimeStream

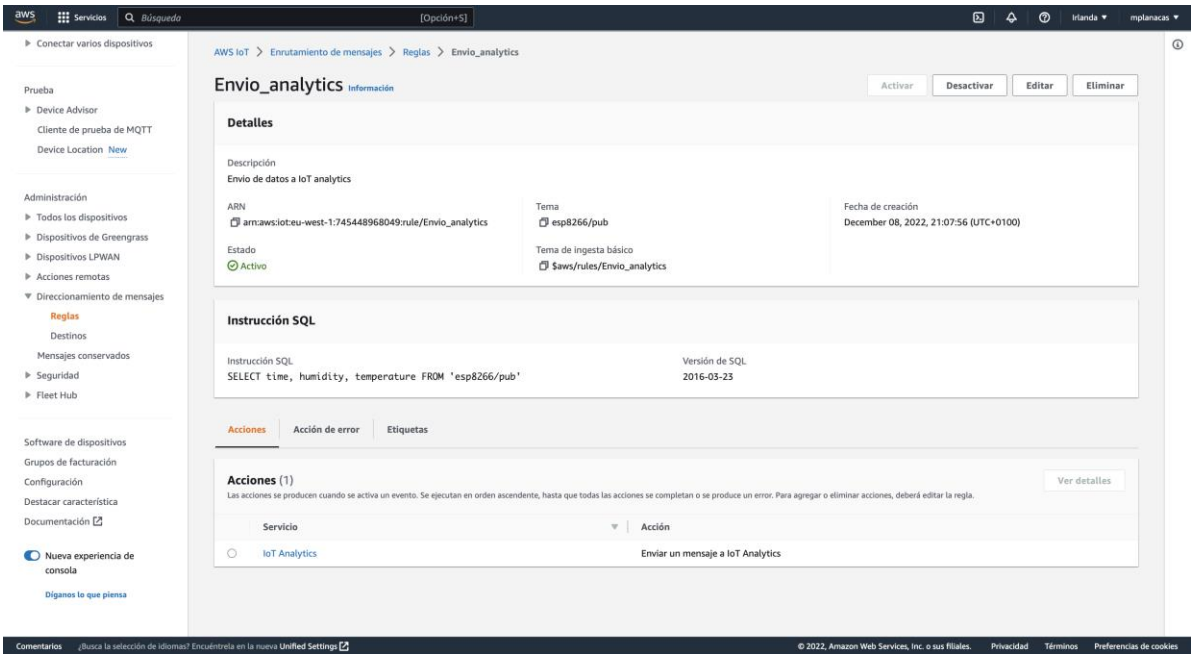


Figura 25: Regla envío de datos a AWS IoT Analytics

Como se puede observar, ambas reglas son muy similares, donde lo único que cambia es el campo de acción, donde se envía la información a un servicio o a otro. Cabe destacar la instrucción SQL donde se indican los datos que se quieren reenviar del mensaje indicando

el tema de MQTT de origen. En este caso se envía los datos de tiempo, humedad y temperatura. Esta sentencia es muy importante, ya que gracias a ella se da forma a los datos en los sistemas de almacenamiento.

3.3.6 Visualización de los datos

De cara a aplicar inteligencia artificial o algoritmos de machine learning, el primer paso es entender los datos y ver cómo se comportan, para ello lo más fácil es crear visualizaciones. En este caso, se ha utilizado la herramienta Grafana administrada por AWS, conectándose a la base de datos creada en Amazon TimeStream.

Cuando se abre la herramienta tiene el siguiente aspecto:

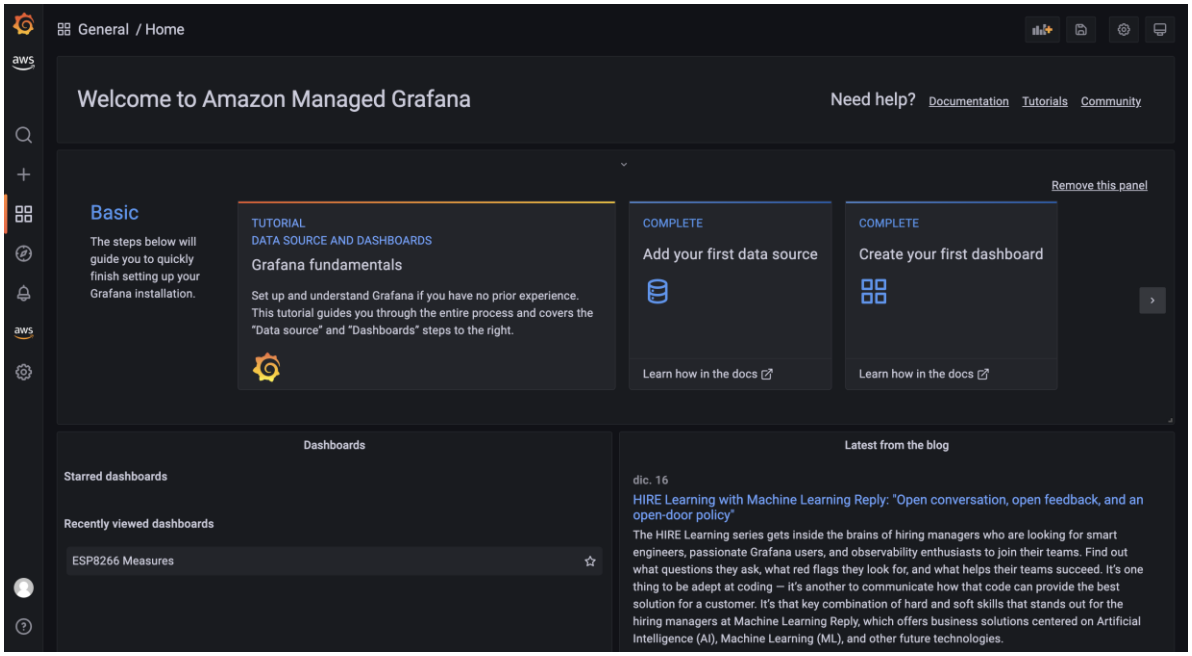


Figura 26: Pantalla de inicio de Grafana

Antes de crear un dashboard y ver cómo se comportan los datos casi a tiempo real, lo mejor es hacer alguna consulta en Amazon TimeStream para ver la forma que tienen y de esta forma orientar las consultas que deben lanzar desde Grafana para cada visualización. Para ello, se accede desde al editor de consultas y se lanza una consulta para visual los últimos datos:

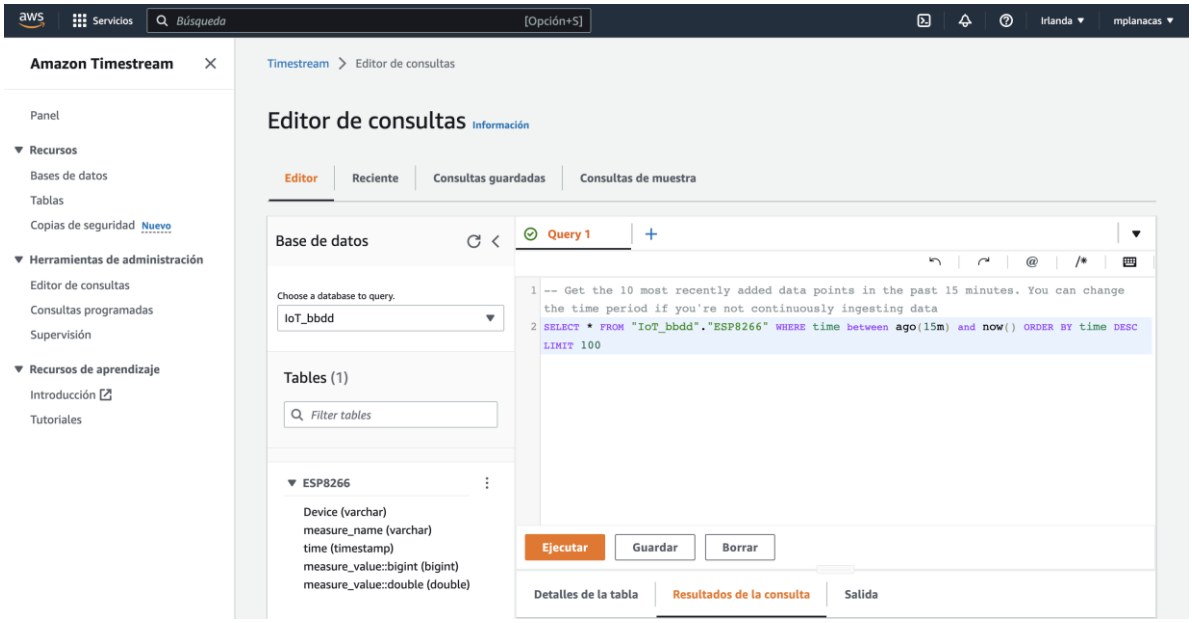
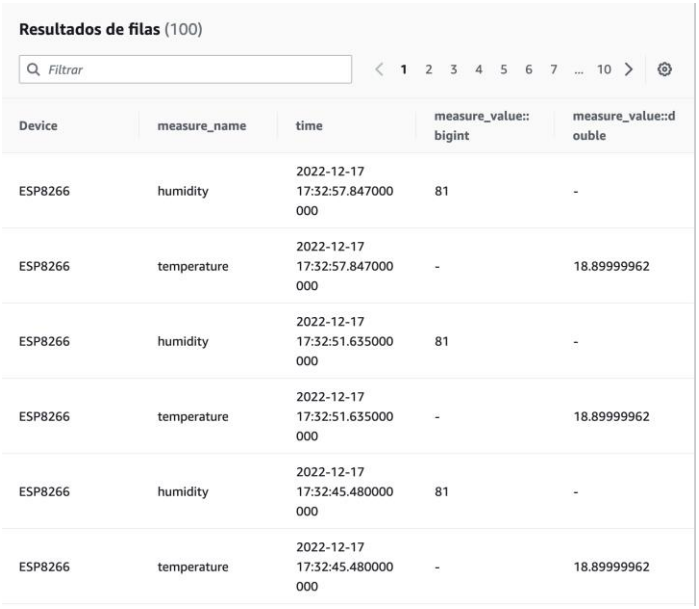


Figura 27: Ejecución de la consulta



Resultados de filas (100)

Filtrar

1 2 3 4 5 6 7 ... 10

Device	measure_name	time	measure_value::bigint	measure_value::double
ESP8266	humidity	2022-12-17 17:32:57.847000000	81	-
ESP8266	temperature	2022-12-17 17:32:57.847000000	-	18.89999962
ESP8266	humidity	2022-12-17 17:32:51.635000000	81	-
ESP8266	temperature	2022-12-17 17:32:51.635000000	-	18.89999962
ESP8266	humidity	2022-12-17 17:32:45.480000000	81	-
ESP8266	temperature	2022-12-17 17:32:45.480000000	-	18.89999962

Figura 28: Resultado de la consulta

Observando la forma en la que se devuelven los datos, en Grafana hay que crear dos consultas diferentes, una para cada medida ya que se guardan los datos de las medidas en dos columnas diferentes y se coloca un valor nulo cuando nombre de la medida no coincide. A continuación, se muestra la creación de una gráfica de líneas en la que se muestran los valores de temperatura y humedad:

Figura 29: Gráfica en Grafana

Para poder visualizar ambas medidas, ha sido necesario crear las siguientes dos consultas dentro del editor de gráficos:

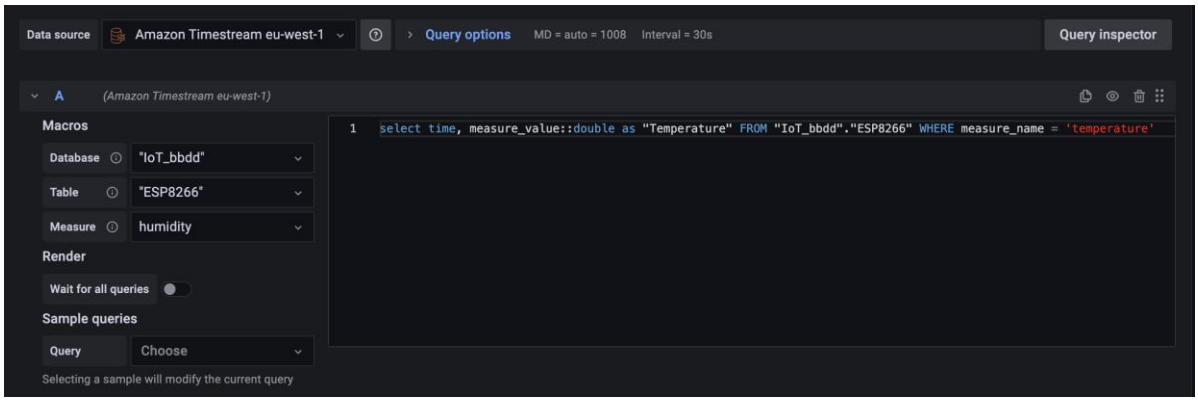


Figura 30: Consulta A en Grafana

Figura 31: *Consulta B en Grafana*

A partir de estos pasos de forma análoga, se crean varias gráficas y se monta el siguiente dashboard:

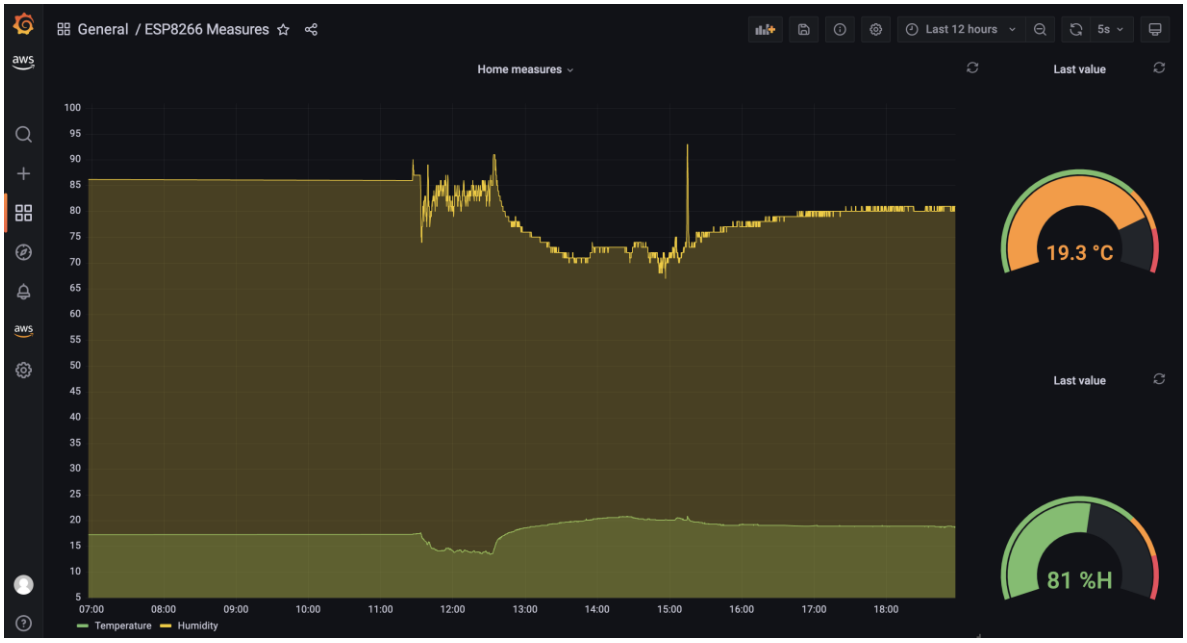


Figura 32: Dashboard en Grafana

Se trata de un dashboard muy sencillo que muestra en las últimas 12 horas, la evolución de las medidas y los últimos valores de temperatura y humedad. Se puede observar claramente como a partir de las 11:30 es cuando se vuelve a empezar a recibir información, ya que durante la implementación no se ha tenido el dispositivo todo el rato funcionando.

3.3.7 Análisis y ejecución de algoritmos

De cara a realizar un análisis profundo de los datos y ejecutar algoritmos de inteligencia artificial para sacar un verdadero valor de los datos y ofrecer un producto que facilite la vida diaria de las personas y les ayude en la toma de decisiones, se ha elegido el servicio IoT Analytics. Como se comentaba, en este proyecto se ha creado una instancia de prueba de bloc de notas para la ejecución de Jupyter Notebooks, además se han analizado otros servicios dentro de Amazon SageMaker como Canvas, que en función del conjunto de datos que se hayan creado, ofrece análisis y creación de modelos de inteligencia artificial y machine learning totalmente automáticos sin la necesidad de que el usuario programe código.

Amazon SageMaker es un producto muy interesante que encaja perfectamente con este trabajo final, pero sus servicios son de pago y son más recomendables en un entorno de producción con el producto final consolidado y con los algoritmos implementados, ya que existe un coste por la creación de cada servicio y además por cada ejecución. Por todo esto, no sale rentable utilizarlo en un entorno académico y de pruebas, ya que son necesarias muchas ejecuciones hasta dar con un producto consolidado. Es por este motivo que el servicio se encuentra parado en la **Figura *23***: Instancia de bloc de notas Amazon SageMaker.

Para simular el uso de este servicio de analítica, se ha instalado un entorno local el software Anaconda para poder utilizar el Jupyter Notebook y realizar algún análisis de una forma muy similar a como se haría en este servicio en la nube. Para ello, primero hubo que disponer de los datos en el PC, para poder descargarlos desde los conjuntos de datos de IoT Analytics se puede lanzar una consulta de toda la información que tiene almacenada y después permite descargarlo en CSV:

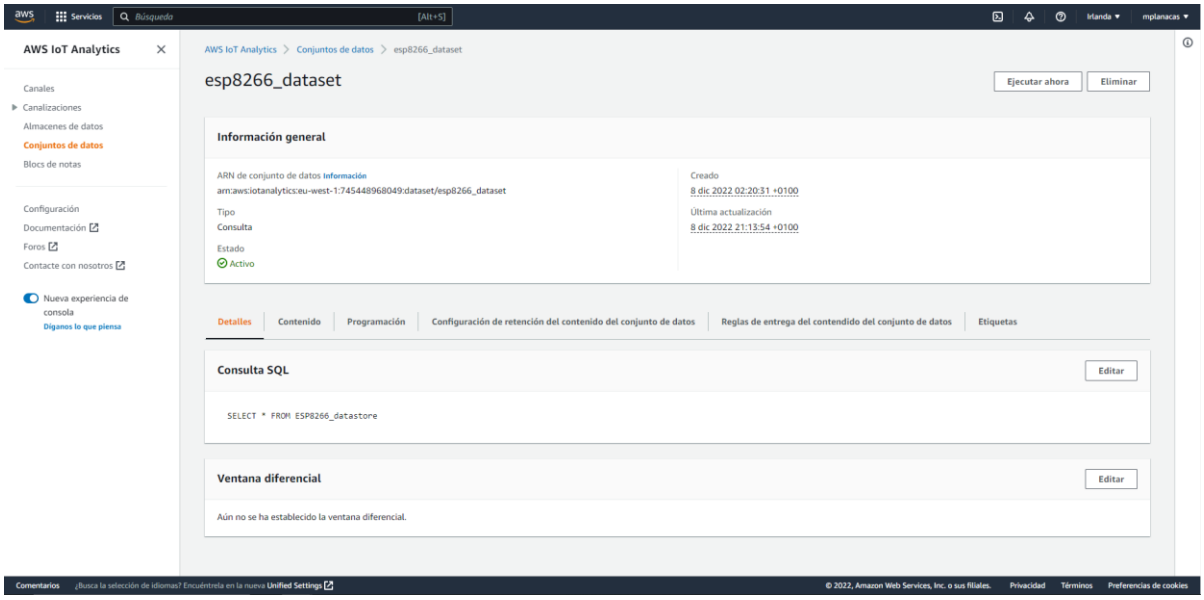


Figura 33: Consulta al conjunto de datos IoT Analytics

Una vez se dispone de los datos, para que sea más sencilla la lectura de dan forma con el editor de texto Notepad++ y se guarda en formato Excel con el aspecto de la siguiente figura. Cabe destacar que se han filtrado los datos, utilizando la información recogida entre los días 17 y 18 de diciembre de 2022. Esto se debe a que el dispositivo se ha tenido enviando información de forma irregular mientras se realizaban las distintas pruebas y el periodo en el que estuvo funcionando de forma continuada fue en el tiempo que va desde la fecha y hora 17/12/2022 11:26:42 hasta 18/12/2022 1:50:05.

	A	B	C
1	time	Humidity	Temperature
2	17/12/2022 11:26:42	86	17,3
3	17/12/2022 11:26:48	90	17,3
4	17/12/2022 11:26:54	90	17,3
5	17/12/2022 11:27:00	89	17,4
6	17/12/2022 11:27:06	89	17,4
7	17/12/2022 11:27:13	88	17,4
8	17/12/2022 11:27:19	88	17,4
9	17/12/2022 11:27:25	88	17,4
10	17/12/2022 11:27:31	88	17,4
11	17/12/2022 11:27:37	88	17,4
12	17/12/2022 11:27:44	87	17,4
13	17/12/2022 11:27:50	87	17,4
14	17/12/2022 11:27:56	87	17,4

Figura 34: Datos procesados

El primer paso tras disponer de los datos es visualizarlos en Jupyter Notebook:

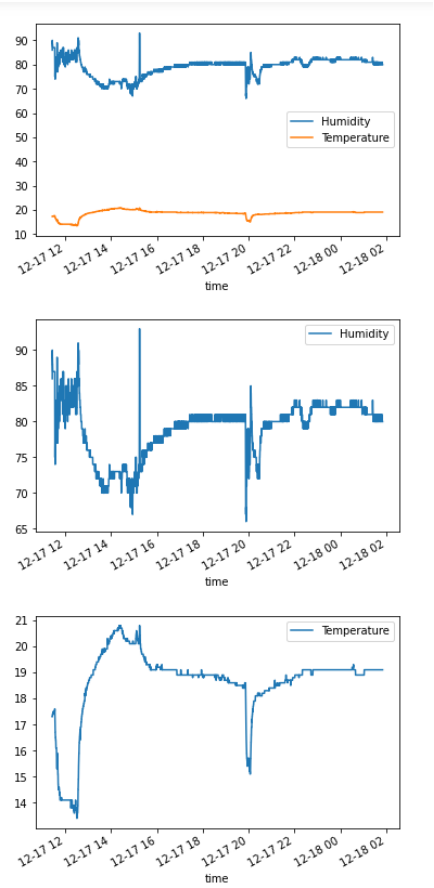


Figura 35: Visualización de datos en Jupyter Notebook

Viendo que los datos son coherentes y se observa lo mismo que en Grafana, el siguiente paso es pensar que se desea conseguir con los datos. Puesto que el tiempo disponible para el desarrollo de esta parte de la implementación es muy reducido, se optado por hacer algunos análisis sencillos. Además, los datos no son lo suficientemente extensos como para realizar implementaciones muy profundas. Por ello, se ha investigado y se ha probado con distintos métodos que se han encontrado en internet y se han adaptado a este proyecto para ofrecer una primera aproximación.

En primer lugar, se ha consultado un artículo de *Towards Data Science* [17], donde se ha adaptado parte del código para profundizar en los datos recolectados. Para ello, se han calculado estadísticas móviles, en concreto la media móvil y la desviación estándar móvil con el objetivo de encontrar estacionariedad en los datos. También se ha realizado la prueba de Dickey-Fuller aumentada (ADF) para calcular el P-valor y valores críticos en los intervalos de confianza 1%, 5% y 10%.

El objetivo de este artículo era aplicar un modelo autorregresivo bajo la premisa de encontrar que los valores pasados tienen efecto en los futuros y así realizar una predicción. Puesto que no se dispone de unos datos lo suficientemente ricos y extensos ni del tiempo suficiente para profundizar en este método, se opta por tratar de realizar una predicción haciendo uso de las librerías de *NeuralProphet*, unas librerías basadas en *Prophet* desarrollado por Facebook donde básicamente funciona como una caja negra que contiene redes neuronales y que con base a los datos que se introduzcan genera predicciones. Para ello, se ha adaptado el código provisto por Sandeep S en la web Medium.com [18].

Finalmente, el último análisis que se realiza es una regresión lineal simple con el objetivo de esclarecer si la temperatura hace que varíe la cantidad de humedad en el ambiente. A simple vista se puede observar que cuando varía la temperatura la humedad lo hace también, por lo que se ha considerado interesante profundizar en ello. En este caso, se ha empleado el código provisto por Joaquín Amat Rodrigo en la web *Ciencia de datos* [19].

Los códigos empleados se encuentran en el Anexo y los resultados se muestran en el siguiente capítulo.

4. Resultados

En este capítulo se muestran los resultados obtenidos en este trabajo final de máster. Concretamente, se muestra lo obtenido referente a visualización y a análisis de los datos ya que el resto de los resultados correspondientes a la implementación del hardware y la integración con los servicios en la nube no produce ningún resultado más allá de lo mostrado en el apartado de implementación, donde se han mostrado imágenes de que se ha integrado correctamente y los servicios se comunican entre sí.

4.1 Resultados de la visualización de datos

Como se ha mostrado en la implementación, hay dos partes del trabajo donde se visualizan los datos, Grafana y Jupyter Notebook. A continuación, se muestran las visualizaciones de los datos empleados en los análisis en ambas herramientas:

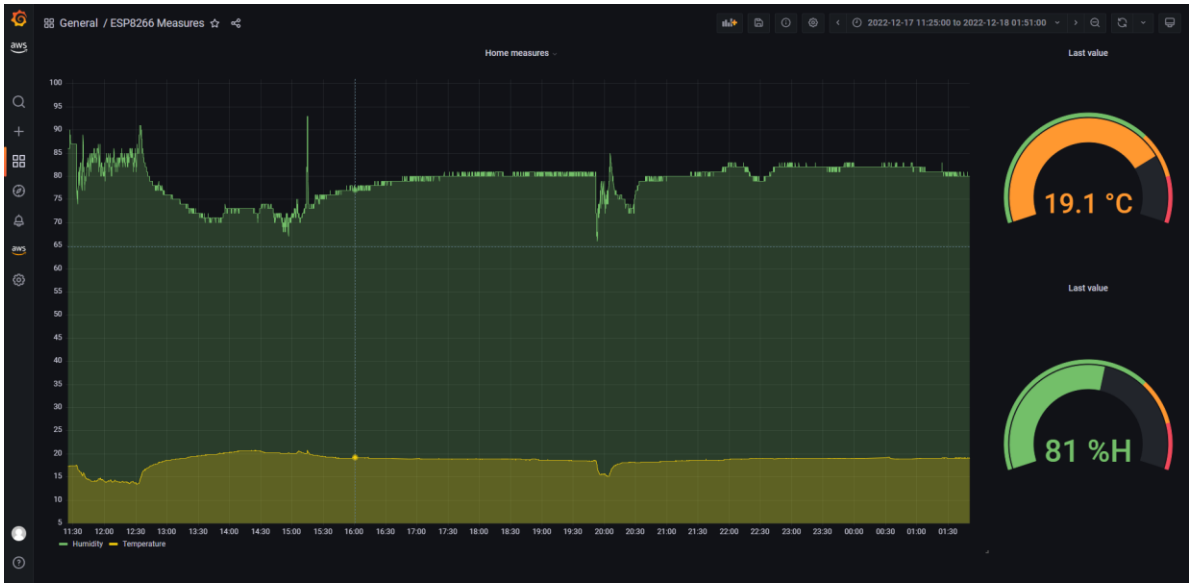


Figura 36: Visualización de datos en Grafana

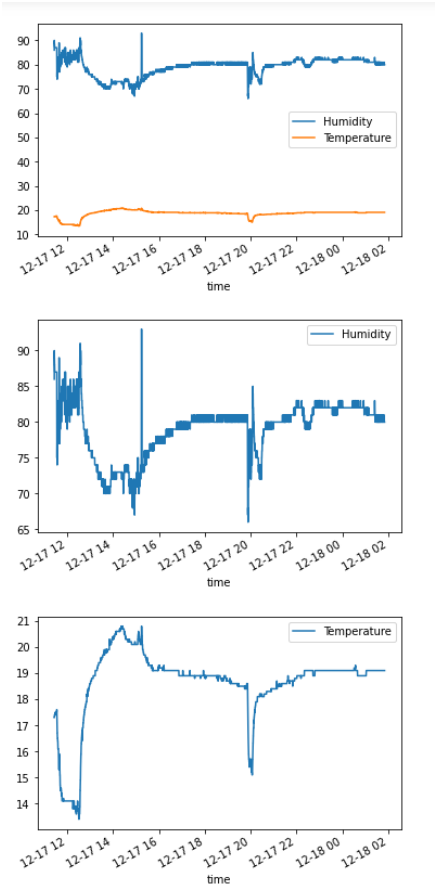


Figura 37: Visualización de datos en Jupyter Notebook

En ambas figuras se puede observar cómo los datos recogidos por el ESP8266 son finalmente visualizados tras su paso por los distintos servicios de la nube de AWS. Cabe destacar que estos datos se han tomado en un piso en Madrid en el mes de diciembre del año 2022 cerca de la ventana, tomando unos valores mínimos de humedad del 67% y máximos de 93%, y 13.5ºC y 20.7ºC de temperatura.

Son valores que se han contrastado que con correctos ya que en el domicilio se dispone de una pantalla proporcionada por una empresa de seguridad que también marca valores muy cercanos de humedad y temperatura. Quizá esta pantalla, marque unos valores más bajos de humedad y más altos en temperatura porque su sensor está ubicado en el pasillo de la casa y el ESP8266 está situado cerca de la ventana para poder realizar pruebas.

Las grandes variaciones que aparecen en los gráficos son producidas por el momento de apertura de las ventanas con el objetivo de ventilar las estancias. Se puede observar también, que el momento donde hay menos humedad y hay una temperatura más alta, y por lo tanto es el momento más confortable, es el periodo comprendido entre 14:30 y las 15:00. Esto se produce gracias a que la habitación ya se ha ventilado correctamente y además la

calefacción ya ha caldeado el ambiente. Es por esto por lo que los resultados son coherentes y se verifica que el sistema funciona correctamente.

4.2 Resultados de los análisis y ejecución de algoritmos

Como se ha indicado en la implementación, se han utilizado tres scripts distintos. En los siguientes subapartados se muestran los resultados de cada uno.

4.2.1 Cálculo de estadísticas móviles y prueba ADF

Se calculan las medias y estadísticas móviles, y se realizan las pruebas ADF tanto para la temperatura como para humedad y se obtienen los siguientes resultados:

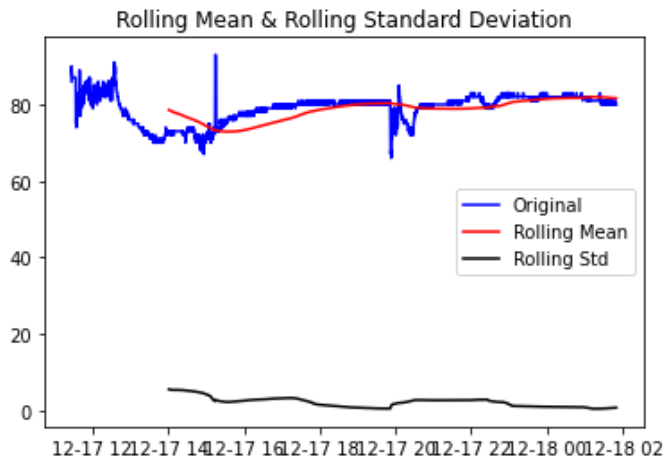


Figura 38: Estadísticas móviles de humedad

```
ADF Statistic: -3.966006586722441
p-value: 0.0015993528228769476
Critical Values:
        1%: -3.4311299335352365
        5%: -2.8618846769283564
        10%: -2.566953466665446
```

Figura 39: Prueba ADF de humedad

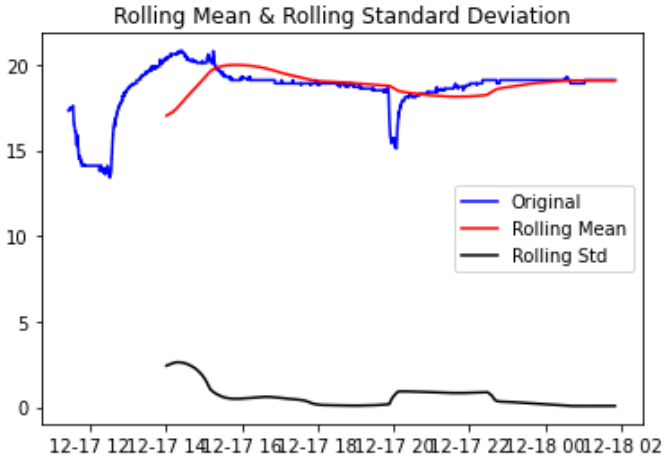


Figura 40: Estadísticas móviles de temperatura

```
ADF Statistic: -2.1696555409117586
p-value: 0.21744245626267222
Critical Values:
        1%: -3.4311314247270106
        5%: -2.8618853358458023
        10%: -2.5669538174128843
```

Figura 41: Prueba ADF de temperatura

Pese a que los datos son reducidos, se puede observar cómo en ambas gráficas de estadísticas móviles, tanto la media móvil como la desviación estándar móvil aparecen muy paralelas al eje horizontal, por lo que indica que hay indicios de estacionariedad en los datos. Por otro lado, los resultados de las pruebas ADF marcan valores del estadístico ADF muy próximos a los valores críticos. Sin embargo, el valor del P-valor solo está por debajo del umbral de 0.05 para el caso de la humedad. Por lo que se concluye que tan solo podría considerase estacionaria la medida de humedad.

4.2.2 Ejecución de predicción con *NeuralProphet*

Después de adaptar el código indicado para los datos de humedad y temperatura, se indica al script la frecuencia de los datos, que en este caso con 6 segundos (tiempo entre cada medición). Además, se indica el número de muestra que se quieren predecir. Como resultado, esta red neuronal grafica la predicción de los datos de temperatura y humedad, y muestra las líneas de tendencia y estacionariedad que haya encontrado.

A continuación, se muestran los resultados obtenidos:

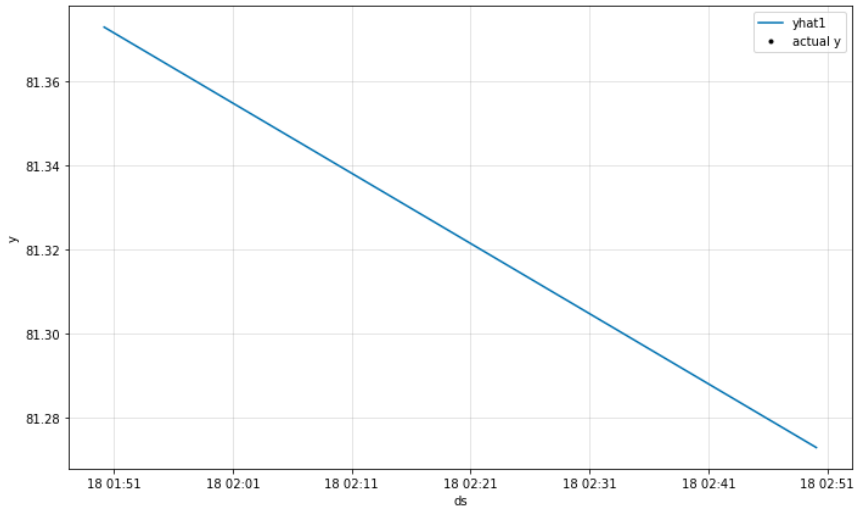


Figura 42: Predicción de humedad

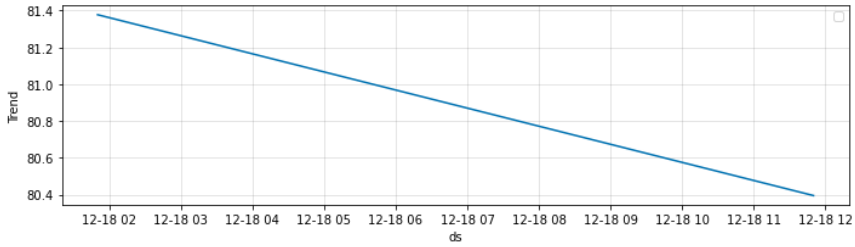


Figura 43: Componentes encontradas de humedad

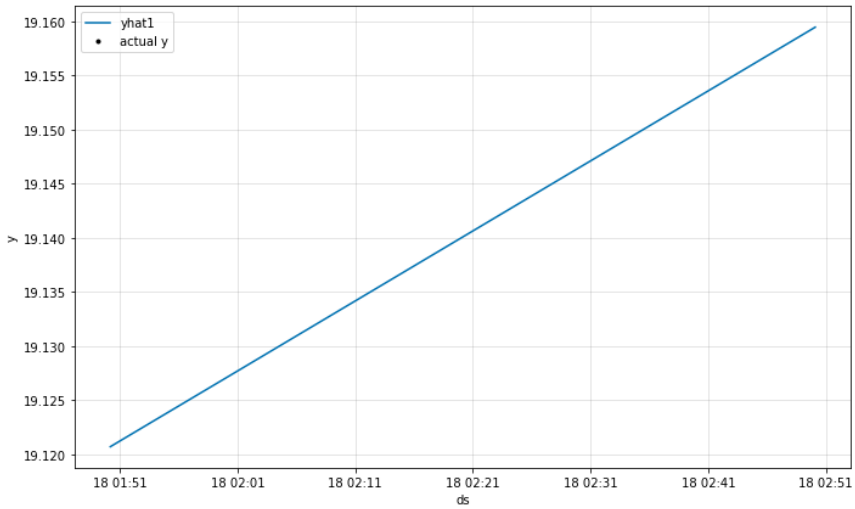


Figura 44: Predicción de temperatura

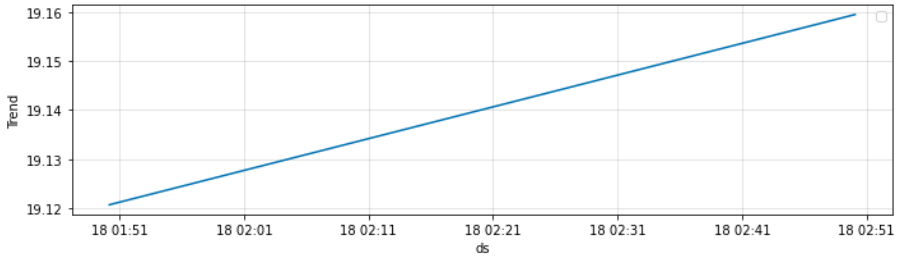


Figura 45: Componentes encontradas de temperatura

Se puede observar como el algoritmo consigue predecir los valores de humedad y temperatura a una hora vista desde las 01:50 del 18 de diciembre. Sin embargo, las componentes encontradas en ambos casos para la predicción son tan solo líneas de tendencia. De esta forma, predice una tendencia decreciente en la humedad y una tendencia ascendente en la temperatura.

4.2.3 Ejecución de regresión lineal simple

El objetivo de la regresión lineal simple en este caso es comprobar si la humedad depende la temperatura del ambiente. Después de ejecutar el script, se obtiene un error cuadrático medio en el conjunto de test del 3.33 y la siguiente tabla de estadísticas:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.224
Model:                            OLS   Adj. R-squared:                  0.224
Method:                 Least Squares   F-statistic:                     1939.
Date:                Sun, 18 Dec 2022   Prob (F-statistic):               0.00
Time:                        15:57:10   Log-Likelihood:                -17503.
No. Observations:                6724   AIC:                         3.501e+04
Df Residuals:                    6722   BIC:                         3.502e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        101.9049      0.520    195.952      0.000     100.885     102.924
x1            -1.2262      0.028    -44.036      0.000      -1.281      -1.172
==============================================================================
Omnibus:                      430.871   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              516.511
Skew:                          -0.658   Prob(JB):                    6.93e-113
Kurtosis:                       3.334   Cond. No.                         244.
==============================================================================
```

Figura 46: Resultados OLS de la regresión

También se obtiene una representación gráfica del modelo:

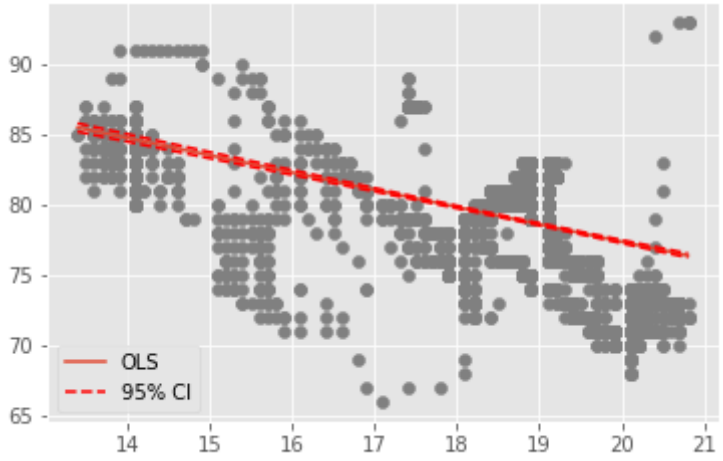


Figura 47: Representación gráfica del modelo de regresión

La interpretación de los resultados es la siguiente. Por un lado, el valor de la tabla R-squared=0.224, indica que el modelo es capaz de explicar el 22.4% de variabilidad observada en los valores de humedad. Además, el valor del p-valor = 0.00 < 0.05 obtenido en el test Prob (F-statistic), indica que varianza explicada por el modelo es superior a la varianza total esperada por azar. Hay que destacar que el error cuadrático medio de 3.33 indica que las predicciones realizadas en el modelo final solo se alejan en promedio 3.33 unidades de humedad

Observando la representación del modelo, se puede ver como la línea de mínimos cuadrados ordinarios (OLS) está ajustada a los intervalos de confianza. Todo esto hace indicar que la humedad depende claramente de la temperatura.

5. Conclusiones y líneas futuras

Después de analizar todo el proceso de implementación del trabajo y ver los resultados obtenidos, en este capítulo se hace una conclusión y se comentan los trabajos futuros.

5.1 Conclusiones

Al inicio de este trabajo final de máster, se enumeraban los siguientes objetivos que se querían lograr con el desarrollo de este proyecto:

- Creación de un dispositivo que sea capaz de medir parámetros del hogar. Debe ser capaz de recoger los datos, y enviarlos para su posterior almacenamiento.

- Creación de una infraestructura en la nube que sirva para almacenar y soportar una visualización avanzada de los datos, así como ejecutar scripts de inteligencia artificial.
- El sistema completo debe ser capaz de ayudar en la toma de decisiones del usuario.

Después de ver el trascurso del trabajo, se puede concluir que se han cumplido todos los objetivos con algún matiz. Todo lo relativo a la implementación del dispositivo y su integración con los servicios de AWS se ha realizado con éxito. Si bien es cierto, que cada una de las implementaciones por separado pueden parecer sencillas, como se ha visto en la justificación de la solución, al tratar de integrar y juntar cada una de las partes pueden surgir numerosos inconvenientes apareciendo distintos puntos de error que podrían haber impedido la realización del proyecto.

Se puede tomar como ejemplo de contratiempo el hecho de que no ha sido posible utilizar el servicio de ejecución de algoritmos en la nube. Por suerte, esto ha ocurrido en la parte final de la cadena y se ha suplido con el uso de Anaconda de forma local después de la obtención de los datos. Pese a que este no era uno de los objetivos, puede justificarse ya que el uso de la nube para ejecutar algoritmos no es estrictamente necesario, debido a que el marco del proyecto es un entorno académico.

Hay que mencionar también una dificultad encontrada que no se ha mencionado en la justificación de la solución y es que AWS en la mayoría de sus servicios no indica si es un servicio de pago o no, ni tampoco las tarifas de cada servicio. El ejemplo más notable fue que realizando pruebas para orientar la arquitectura, se probó a crear un dominio de OpenSearch (anterior ElasticSearch) para ver si podía incluirse en la solución, y simplemente por tener el dominio activo varios días, sin hacer uso de él, se han cobrado 27.49$.

Por otro lado, como se ha visto en los resultados obtenidos, ha faltado tiempo para cumplir el último de los objetivos de manera completa. Pese a que se han obtenido resultados interesantes como que la humedad depende de la temperatura y que, en este caso, se alcanzan buenos niveles de humedad cuando la temperatura sube, es un hecho que solo ha habido tiempo para reutilizar y adaptar códigos de otros casos de uso. Todo esto también está condicionado por los datos, donde se han utilizado datos de menos de 24 horas. En un entorno de producción, lo ideal sería crear un buen histórico de datos, lo más completo posible, con un mayor número de parámetros donde también se refleje las acciones tomadas dentro del hogar. Indicando, por ejemplo, cuando se abren las ventanas y momentos de calefacción activa entre otros.

Como conclusión, el balance final es muy positivo y se considera un éxito, ya que se había planteado un proyecto muy ambicioso en relación con el tiempo que se disponía. Además, se han solventado numerosos contratiempos y dificultades consiguiendo sacar adelante una solución y consiguiendo un producto funcional.

5.2 Líneas futuras

Para poder cumplir con los objetivos del trabajo al 100% se proponen a continuación algunas líneas de trabajo futuras que se han ido mencionando indirectamente a lo largo de la memoria.

Una de las líneas de trabajo más clara, es conformar un conjunto de datos más completo. Esto pasa por añadir más sensores al sistema como pueden ser sensores de calidad del aire, sensores de movimiento, sensores que identifiquen acciones de los convivientes de la casa para saber por ejemplo cuándo hay alguna ventana abierta, cuando está encendida la calefacción o el aire acondicionado y además recolectar estos datos durante el tiempo necesario para crear un histórico donde se puedan detectar periodicidades.

Esta mejora en los datos también será útil de cara a implementar unos algoritmos especializados en este caso de uso. Por ejemplo, si finalmente de detectan claramente comportamientos estacionarios se podría crear algún algoritmo ARIMA identificando bien sus parámetros para predecir todo tipo de comportamientos, así como introducir algoritmos de clasificación para detectar distintas situaciones. Una vez creados los algoritmos finales, ya se podría plantear el uso de la nube para ejecutarlos. Esto unido al uso de actuadores para controlar la calefacción, diferentes dispositivos en el hogar o el uso de luces, puede hacer que se logre obtener un entorno de hogar inteligente y eficiente que mejore la calidad de vida de las personas. Además, puede combinarse con el ámbito de las alarmas y la seguridad para detectar situaciones de peligro como robo o incendio y poder socorrer a los convivientes.

6. Presupuesto

A continuación, se muestra una tabla con el presupuesto desglosando los precios de todos elementos con coste asociado necesarios para la realización del proyecto. Cabe destacar que se han descartado los que finalmente no se han incluido en el sistema:

Elemento	Subsistema	Precio
ESP8266	Hardware	7.5 €
DHT11	Hardware	2.6 €
Amazon TimeStream	Almacenamiento	0.33 €
Total		**10.43 €**

También en caso de que se desee consultar se adjunta una figura con el desglose de precios obtenido de la propia facturación de AWS. De esta forma, aunque no todos se hayan utilizado de forma destacable, puede ser útil para ver que ver qué servicios llevan un coste asociado:

Cargos por servicios de AWS	$33.95
CloudWatch	$0.00
Data Transfer	$0.00
DynamoDB	$0.00
Elastic File System	$0.00
IoT	$0.00
IoT Analytics	$0.00
Key Management Service	$0.00
Managed Grafana	$0.00
OpenSearch Service	$27.49
SageMaker	$0.22
Service Catalog	$0.00
Simple Notification Service	$0.00
Simple Storage Service	$0.00
Timestream	$0.35
Impuestos	
IVA que se recaudará	$5.89

Figura 48: Facturación total AWS

7. Glosario

IOT – Internet de las cosas
IA – Inteligencia artificial
ODS – Objetivos de desarrollo sostenible
DIY – Hágalo usted mismo, del inglés, *Do It Yourself*
AWS – Amazon Web Services
SVM – Máquina de vectores soporte, del inglés, *Support Vector Machine*
KNN – K vecinos más cercanos, del inglés, *K Nearest Neighbors*
IDE – Entorno de desarrollo
BBDD – Base de datos
SQL – Lenguaje de consulta estructurado, del inglés, *Structured Query Language*
CSV – Valores separados por comas, del inglés, *Comma Separated Values*
ADF – Dickey Fuller aumentada
OLS – Mínimos cuadrados ordinarios, del inglés, Ordinary Least Squares
ARIMA – Modelo autorregresivo integrado de media móvil

8. Bibliografía

[1] Naciones Unidas, "un.org," [Online]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[2] Iot World Online, "iotworldonline.es," 21 junio 2020. [Online]. Available: https://www.iotworldonline.es/iot-en-casa-seleccion-de-varios-dispositivos-iot-para-convertir-tu-hogar-en-una-smarthome/. [Accessed 20 octubre 2022].

[3] Lanner, "lanner-america.com," [Online]. Available: https://www.lanner-america.com/es/blog-es/5-ejemplos-de-dispositivos-iot-en-su-proxima-casa-inteligente/. [Accessed 20 octubre 2022].

[4] L. M. Amaya Fariño, A. Tumbaco Reyes, E. Roca Quirumbay, T. Villón González, B. Mendoza Morán and Á. Reyes Quimís, "El IoT aplicado a la Domótica," *Revista Científica y Tecnológica UPSE,* vol. 7, no. 1, pp. 21-28, 2020.

[5] Amazon, "aws.amazon.com," [Online]. Available: https://aws.amazon.com/es/iot-core/getting-started/. [Accessed 23 octubre 2022].

[6] Amazon, "docs.aws.amazon.com," [Online]. Available: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html. [Accessed 23 octubre 2022].

[7] Explore Embedded, "exploreembedded.com," [Online]. Available: https://exploreembedded.com/wiki/AWS_IOT_with_Arduino_ESP32. [Accessed 23 octubre 2022].

[8] N. J. Nilsson, Inteligencia artificial. Una nueva síntesis, S. A. Mcgraw-Hill, 2000.

[9] J. O. M. CHAUX, "ARTIFICIAL, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN DOMÓTICA PARA ILUMINACIÓN USANDO INTELIGENCIA," [Online]. Available: https://ciencia.lasalle.edu.co/ing_automatizacion/38/. [Accessed 23 octubre 2022].

[10] U. Shankar Shanthamallu, A. Spanias, C. Tepedelenlioglu and M. Stanley, "A Brief Survey of Machine Learning Methods," SenSIP Center, School of ECEE, Arizona State University, NXP Semiconductors, Arizona, 2020.

[11] R. M. Jacobson, "Comparativa y estudio de plataformas IoT," Escola Universitària Politècnica de Mataró, 2017.

[12] J. Paredes, "Project Hub," 1 Febrero 2019. [Online]. Available: https://create.arduino.cc/projecthub/imjeffparedes/add-wifi-to-arduino-uno-663b9e. [Accessed 13 Diciembre 2022].

[13] Amazon, "Amazon Web Services," [Online]. Available: https://aws.amazon.com/es/timestream/. [Accessed 14 Diciembre 2022].

[14] Amazon, "Amazon Web Services," [Online]. Available: https://aws.amazon.com/es/iot-analytics/. [Accessed 14 Diciembre 2022].

[15] Amazon Web Services, "Amazon Web Services," [Online]. Available: https://aws.amazon.com/es/grafana/. [Accessed 16 Diciembre 2022].

[16] How to Electronics, "How to Electronics," [Online]. Available: https://how2electronics.com/connecting-esp8266-to-amazon-aws-iot-core-using-mqtt/. [Accessed 16 Diciembre 2022].

[17] Towards Data Science, "Towardsdatascience.com," [Online]. Available: https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7. [Accessed 18 Diciembre 2022].

[18] Sandeep, "Medium," [Online]. Available: https://medium.com/analytics-vidhya/neuralprophet-a-neural-network-based-time-series-model-3c74af3b0ec6. [Accessed 18 Diciembre 2022].

[19] J. A. Rodrigo, "Ciencia de Datos," Octubre 2022. [Online]. Available: https://www.cienciadedatos.net/documentos/py10-regresion-lineal-python.html. [Accessed 18 Diciembre 2022].

9. Anexos

9.1 Código proyecto Arduino

9.1.1 ESP8266_AWS

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <time.h>
#include "secrets.h"
#include "DHT.h"

#define DHTPIN 4        // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

float h ;
float t;
unsigned long lastMillis = 0;
unsigned long previousMillis = 0;
const long interval = 5000;

#define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"

WiFiClientSecure net;

BearSSL::X509List cert(cacert);
BearSSL::X509List client_crt(client_cert);
BearSSL::PrivateKey key(privkey);

PubSubClient client(net);

time_t now;
time_t nowish = 1510592825;


void NTPConnect(void)
{
```

```
  Serial.print("Setting time using SNTP");
  configTime(TIME_ZONE * 3600, 0 * 3600, "pool.ntp.org", "time.nist.gov");
  now = time(nullptr);
  while (now < nowish)
  {
    delay(500);
    Serial.print(".");
    now = time(nullptr);
  }
  Serial.println("done!");
  struct tm timeinfo;
  gmtime_r(&now, &timeinfo);
  Serial.print("Current time: ");
  Serial.print(asctime(&timeinfo));
}


void messageReceived(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Received [");
  Serial.print(topic);
  Serial.print("]: ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}


void connectAWS()
{
  delay(3000);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.println(String("Attempting to connect to SSID: ") + String(WIFI_SSID));

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(1000);
  }
```

```
  NTPConnect();

  net.setTrustAnchors(&cert);
  net.setClientRSACert(&client_crt, &key);

  client.setServer(MQTT_HOST, 8883);
  client.setCallback(messageReceived);


  Serial.println("Connecting to AWS IOT");

  while (!client.connect(THINGNAME))
  {
    Serial.print(".");
    delay(1000);
  }

  if (!client.connected()) {
    Serial.println("AWS IoT Timeout!");
    return;
  }
  // Subscribe to a topic
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);

  Serial.println("AWS IoT Connected!");
}


void publishMessage()
{
  StaticJsonDocument<200> doc;
  doc["time"] = millis();
  doc["humidity"] = h;
  doc["temperature"] = t;
  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client

  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}


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

```
  connectAWS();
  dht.begin();
}


void loop()
{
  h = dht.readHumidity();
  t = dht.readTemperature();

  if (isnan(h) || isnan(t) )  // Check if any reads failed and exit early (to try again).
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));
  delay(2000);

  now = time(nullptr);

  if (!client.connected())
  {
    connectAWS();
  }
  else
  {
    client.loop();
    if (millis() - lastMillis > 5000)
    {
      lastMillis = millis();
      publishMessage();
    }
  }
}
```

9.1.2 Secrets.h

```
#include <pgmspace.h>
```

```
#define SECRET

const char WIFI_SSID[] = "DIGIFIBRA*****";
const char WIFI_PASSWORD[] = "**********";

#define THINGNAME "ESP8266"

int8_t TIME_ZONE = 1; //España: GMT +1

const char MQTT_HOST[] = "a2mjls9d3u8csf-ats.iot.eu-west-1.amazonaws.com";


static const char cacert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExB
BbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFow
OTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";


// Copy contents from XXXXXXXX-certificate.pem.crt here ▼
static const char client_cert[] PROGMEM = R"KEY(
-----BEGIN CERTIFICATE-----
MIIDWjCCAkKgAwIBAgIVAPx1Vd/RnIhKNRxIiJEkJQRI0+48MA0GCSqGSIb3DQEB
CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t
IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yMjEyMDgxODU1
```

MDhaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2VydGlmaWN
h
dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDrP6k6qaFYMAgw6icV
ZLFGZBxHQPsUsNypiIJT0OM2cbJfO59WCLleX1JbA5eexiOvCBDO6iHelozz6qB+
xOxHOHUVyormd6xeXs0rRwP6Lq9QmheMSDk7NGQI3A6D9E4oALbjH5nvKC058/t/
eXaZ64Fx4HwJ4iBJ3X1rEdmAZoBs867Su6lhizXF/cEGNV4NvubDV8m5TtcC5Alq
T9Tb3qZuO+xAF3VBmu3JFSpfy3CSs+Jb9gvrmGOyJW1Gy0HtFKvToDBoXI36AhUw
9xH7KJ1sx0lnu1fih9ovV9P76BsV5MSlolhV7Vb5HL7Gxeblgg9VLDgfmW0AS/oF
rTQtAgMBAAGjYDBeMB8GA1UdIwQYMBaAFCMy7u8jk+8/+QlX1r/faetB/mk+MB0G
A1UdDgQWBBQh89J4gTcp8UQgL3HQCr+3prbJPjAMBgNVHRMBAf8EAjAAMA4GA1Ud
DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAAWENHZPsjBM3FkT7Bg4+4Cib
Zvz2CqwozjZG2KkyRteU9d5sI8W4uVXmVZO0FXlkvFIzYqctEFc3IColjNcDJSCI
xe0lCA2c0cNttGt1JH8/BlnPxGh7kSHLhlXFoxcB6q6GOU7H72ktPPCaXjm1J1Gm
+UfTAO49+sk4XtEGZkQb+y/CkZ8CICXl/QVHEaX4Ps6E/5DNKfD5GwC8ixeNQJ1B
i/3P1w55NEo5XRizB02DcUfVG6p/WZ5mjgzEF5S7bOk5LHXc7RtF6ExKzvaKiTbb
Ha+KKQJdABzs+hvx0zlZxHir5RsbX9TTfMkj2mOOPCAqrbuRzvIC7VCdPvREWw==
-----END CERTIFICATE-----
)KEY";

// Copy contents from XXXXXXXX-private.pem.key here ▼
static const char privkey[] PROGMEM = R"KEY(
-----BEGIN RSA PRIVATE KEY-----

MIIEpQIBAAKCAQEA6z+pOqmhWDAIMOonFWSxRmQcR0D7FLDcqYiCU9DjNnGyXzuf
Vgi5Xl9SWwOXnsYjrwgQzuoh3paM8+qgfsTsRzh1FcqK5nesXl7NK0cD+i6vUJoX
jEg5OzRkCNwOg/ROKAC24x+Z7ygtOfP7f3l2meuBceB8CeIgSd19axHZgGaAbPOu
0rupYYs1xf3BBjVeDb7mw1fJuU7XAuQJak/U296mbjvsQBd1QZrtyRUqX8twkrPi
W/YL65hjsiVtRstB7RSr06AwaFyN+gIVMPcR+yidbMdJZ7tX4ofaL1fT++gbFeTE
paJYVe1W+Ry+xsXm5YIPVSw4H5ltAEv6Ba00LQIDAQABAoIBAQCEVSFbMITkWNCi
w3UM4cGfOT13D6St9PeMHpBBmeXWgIq4a4+sE+hCoBhxvkmSNR9pnH3oB2jaD5Jm
u3thFGFdhvLGOVN9ZaB+zZFYYPBp0c0o6CWK8i3Pkh8bpL49mDZG5m/6v5IWKGGY
khjNkoQW/ZWpe4ALBzrmkkO08SuKwAn6TPVmT0siuSZIxn//9IzQ6pB5OT4Q8HoW
k4bVCo2/lnG5CAagNE/uk6VTyIeGW66GSYVgg2v57wZZzdBN69NmCsKwr/xUWPSe
Ntx2UK0KvYurpNxalhdJajsnjtkzV0bLnmhEFrGrI4/1SPCR7C02Z/9lUJaBBlp6
52hdB295AoGBAP9R7A9/FL4ybWDK9DmxmWYTpyHkTp2ivp71886I/1JLH+XywNPL
pLnFG1jpWMomB1HKs+4ZMCWJfYK6n+/jvLYFZBU/eg50ULWp6jyRNed5oetZRuK4
nPzd7Ityz4/Hz2HAyR6djaXK4gyE9w19nQbTWGnzjA+XoTTTvCqZD+JbAoGBAOvg
DeNUAiA8kniuOvqZT17H1gNQUsU3qzz1qiB2AVB72yTekw4QoKxiPSQwibsMQpt0
i7Lquwk4Ga6zMNAl7yn+njWIQYpucobysvVEYGlv6XOHlru2CCKp0sJJhk2MhN6y
/1FL7n4sEYhqq1bMcGQrvHkHBQBXcEINvqj3CfoXAoGAYIbPUViBNhsIgSSZysob
LZ1WfZi8JBqoKJ7h6v6eULV4PM5XGnuOLnGXkji+Scaw07pn5id9rWNaoTb/Sol2
CfblzKav9ZECBnxTw0w+7eFtVvXst8CGEY+xY1Mnp7sp61Igd0OnjruyVHtcj6wi
1rbGYWAjXq2706YNKKmveKsCgYEAwS640SWdtB9H6xTvcw0bJ0pN6lrf32xPj9a4
JmvfebZm7hRopk1R+r1ezAQA2bQZ3U4fmEKNOui6iA10fhWDgoZptvhIjYMVw5aE

oUIGIHSq45b6CfT1RsaFueAT4fGjZO1/FGs3Ahk/ptWOTMGVj0UO0KTCINx8MW0m
aSYjArMCgYEA9+8ob4kOfyd23sYxqjD1jqzL3MxrY6mVJ0Cs+oDWR1scS2bbn7IY
gNB5XVG3LQBLJLjxpQd9uCUWKJvPXjFKyioG6pa40e9roCmIiQplD9gvkeWPsdJ1
f5EsED79/lHaTrGiOsOatLLtb75LfzcQNoFRDtzIl3dDRQ4H5T792mU=
-----END RSA PRIVATE KEY-----

)KEY";

9.2 Código scripts de Python

9.2.1 Ejecución de estadísticas móviles y pruebas ADF

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

df = pd.read_excel('datos_excel.xlsx')
df ['time'] = pd.to_datetime(df['time'])
df.head()
df = df.set_index('time')
df_humidity = df[['Humidity']]
df_temperature = df[['Temperature']]

df.plot()
df_humidity.plot()
df_temperature.plot()

rolling_mean = df_temperature.rolling(window = 1500).mean() #Se cambia el dataframe de
humedad y tempertura para obtener unos datos u otros
rolling_std = df_temperature.rolling(window = 1500).std()
plt.plot(df_temperature, color = 'blue', label = 'Original')
plt.plot(rolling_mean, color = 'red', label = 'Rolling Mean')
plt.plot(rolling_std, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title('Rolling Mean & Rolling Standard Deviation')
```

```
plt.show()

result = adfuller(df_temperature['Temperature'])
print('ADF Statistic: {}'.format(result[0]))
print('p-value: {}'.format(result[1]))
print('Critical Values:')
for key, value in result[4].items():
    print('\t{}: {}'.format(key, value))
```

9.2.2 Ejecución de predicción con *NeuralProphet*

```
import pandas as pd
from neuralprophet import NeuralProphet
from matplotlib import pyplot as plt
import pickle

df = pd.read_excel('datos_excel.xlsx')
df ['time'] = pd.to_datetime(df['time'])
df.head()
#df = df.set_index('time')
df_humidity = df[['time','Humidity']]
df_temperature = df[['time','Temperature']]

data = df[['time','Temperature']]#esta linea se modifica para tomar un df u otro
data.dropna(inplace=True)
data.columns = ['ds', 'y']
data.head()

m = NeuralProphet()
model = m.fit(data, freq='6s', epochs=100)

future = m.make_future_dataframe(data, periods=600)
forecast = m.predict(future)
forecast.head()

plot1 = m.plot(forecast)

plt2 = m.plot_components(forecast)
```

9.2.3 Ejecución de regresión lineal simple

```
# Tratamiento de datos
```

```
# ==============================================================
import pandas as pd
import numpy as np

# Gráficos
# ==============================================================
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y modelado
# ==============================================================
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Configuración matplotlib
# ==============================================================
plt.rcParams['image.cmap'] = "bwr"
#plt.rcParams['figure.dpi'] = "100"
plt.rcParams['savefig.bbox'] = "tight"
style.use('ggplot') or plt.style.use('ggplot')

# Configuración warnings
# ==============================================================
import warnings
```

```
warnings.filterwarnings('ignore')

df = pd.read_excel('datos_excel.xlsx')
df ['time'] = pd.to_datetime(df['time'])
df.head()
#df = df.set_index('time')
datos = df[['Temperature','Humidity']]
#df_temperature = df[['time','Temperature']]

# Correlación lineal entre las dos variables
#
================================================================
===============
corr_test = pearsonr(x = datos['Temperature'], y =  datos['Humidity'])
print("Coeficiente de correlación de Pearson: ", corr_test[0])
print("P-value: ", corr_test[1])

# División de los datos en train y test
#
================================================================
===============
X = datos[['Temperature']]
y = datos['Humidity']

X_train, X_test, y_train, y_test = train_test_split(
                                   X.values.reshape(-1,1),
                                   y.values.reshape(-1,1),
                                   train_size   = 0.8,
                                   random_state = 1234,
                                   shuffle      = True
                                 )

# Creación del modelo
#
================================================================
===============
modelo = LinearRegression()
modelo.fit(X = X_train.reshape(-1, 1), y = y_train)

# Información del modelo
```

```
#
==================================================================
===============
print("Intercept:", modelo.intercept_)
print("Coeficiente:", list(zip(X.columns, modelo.coef_.flatten(), )))
print("Coeficiente de determinación R^2:", modelo.score(X, y))

# Error de test del modelo
#
==================================================================
===============
predicciones = modelo.predict(X = X_test)
print(predicciones[0:3,])

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print("")
print(f"El error (rmse) de test es: {rmse}")

# División de los datos en train y test
#
==================================================================
===============
X = datos[['Temperature']]
y = datos['Humidity']

X_train, X_test, y_train, y_test = train_test_split(
                                   X.values.reshape(-1,1),
                                   y.values.reshape(-1,1),
                                   train_size   = 0.8,
                                   random_state = 1234,
                                   shuffle      = True
                                 )

# Creación del modelo utilizando matrices como en scikitlearn
```

```
#
==================================================================
===============
# A la matriz de predictores se le tiene que añadir una columna de 1s para el intercept
del modelo
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())

# Intervalos de confianza para los coeficientes del modelo
#
==================================================================
===============
modelo.conf_int(alpha=0.05)

# Predicciones con intervalo de confianza del 95%
#
==================================================================
===============
predicciones = modelo.get_prediction(exog = X_train).summary_frame(alpha=0.05)
predicciones.head(4)

# Predicciones con intervalo de confianza del 95%
#
==================================================================
===============
predicciones = modelo.get_prediction(exog = X_train).summary_frame(alpha=0.05)
predicciones['x'] = X_train[:, 1]
predicciones['y'] = y_train
predicciones = predicciones.sort_values('x')

# Gráfico del modelo
#
==================================================================
===============
fig, ax = plt.subplots(figsize=(6, 3.84))

ax.scatter(predicciones['x'], predicciones['y'], marker='o', color = "gray")
ax.plot(predicciones['x'], predicciones["mean"], linestyle='-', label="OLS")
```

```
ax.plot(predicciones['x'], predicciones["mean_ci_lower"], linestyle='--', color='red',
label="95% CI")
ax.plot(predicciones['x'], predicciones["mean_ci_upper"], linestyle='--', color='red')
ax.fill_between(predicciones['x'], predicciones["mean_ci_lower"],
predicciones["mean_ci_upper"], alpha=0.1)
ax.legend();

# Error de test del modelo
#
================================================================
===============
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print("")
print(f"El error (rmse) de test es: {rmse}")
```