UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Análisis y predicción de datos obtenidos de un AGV

Presentado por Gonzalo Burgos de la Hera
en Universidad de Burgos — 5 de julio de 2023
Tutor: Bruno Baruque Zanón y Jesús Enrique Sierra García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanón, profesor del departamento de Digitalización, área de Ciencia de la Computación e Inteligencia Artificial, y D. Jesús Enrique Sierra García, profesor del departamento de Digitalización, área de Ingeniería de Sistemas y Automática.

Exponen:

Que el alumno D. Gonzalo Burgos de la Hera, con DNI 71312090S, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Análisis y predicción de datos obtenidos del funcionamiento de un AGV.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 5 de julio de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. Bruno Baruque Zanón	D. Jesús Enrique Sierra García

Resumen

En cualquier entorno industrial, poder predecir el comportamiento de un sistema de manera precisa ofrece una gran ventaja, pues permite ahorrar costes mejorando la productividad.

En este trabajo se propone por tanto el desarrollo de un sistema capaz de almacenar y predecir datos de un AGV (Autonomous Guided Vehicle), concretamente la velocidad de sus ruedas. Con el fin de conseguir el sistema más óptimo y eficiente posible, se ha realizado un estudio comparativo entre diferentes sistemas gestores de bases de datos y de modelos predictivos. Con el fin de obtener la mayor modularidad posible, este sistema ha sido desarrollado como un conjunto de microservicios que interaccionan entre sí.

Descriptores

AGV, series temporales, gestores de bases de datos, modelos predictivos, aprendizaje automático, microservicios, ...

Abstract

In any industrial environment, being able to predict the behaviour of a system in an accurate way offers a great advantage, since it allows cost savings by improving productivity.

This work proposes the development of a system capable of storing and predicting data from an AGV (Autonomous Guided Vehicle), specifically the speed of its wheels. In order to achieve the most optimal and efficient system possible, a comparative study has been carried out between different database and predictive model management systems. In order to obtain the greatest possible modularity, this system has been developed as a set of microservices that interact with each other.

Keywords

AGV, time series, database management systems, predictive models, machine learning, microservices, ...

Índice general

Índice de figuras

Índice de tablas

Introducción

Los AGV, Autonomous Guided Vehicles por sus siglas en inglés, son complejos sistemas robóticos, capaces de moverse en un entorno concreto, cuyo uso es transportar cargas pesadas en fábricas o almacenes, y que están diseñados para mejorar la eficiencia y la productividad en la logística y el transporte de materiales. Debido a sus ventajas en seguridad, flexibilidad y velocidad, esta tecnología está consiguiendo una mayor relevancia [1].

Aunque estos sistemas pueden mejorar la productividad, desajustes en su configuración u otros errores operacionales pueden producir una reducción de su rendimiento, y, en casos extremos, causar una detención de la línea de producción. Por este motivo, es necesario extraer información de los sistemas en marcha para analizar el rendimiento de las máquinas y las aplicaciones logísticas. Esta información puede usarse para predecir comportamientos futuros del sistema, realizar mantenimiento predictivo y proveer retroalimentación con el fin de diseñar mejoras continuas de las máquinas. Estas predicciones pueden ser conseguidas con el uso de algoritmos de análisis de series temporales, que permitan anticipar futuras condiciones del sistema. [2]

Algunos de estos datos obtenidos del sistema tienen una baja frecuencia de actualización, como puede ser la temperatura y voltaje de la batería, pero otros cambian cada pocos milisegundos, como la corriente eléctrica, la velocidad, la posición del vehículo, errores y estado, etc. Toda esta información proveída por el AGV debe estar relacionada con el tiempo en el que fue generada, por lo que puede ser agrupada en series temporales [3]. Cualquier tipo de base de datos puede usarse para almacenar esta información generada por los AGV, sin embargo, ya que se trata de series temporales, es preferible utilizar bases de datos para series temporales para optimizar el rendimiento del sistema.

Estos datos pueden ser posteriormente analizados y utilizados para entrenar un modelo basado en técnicas estadísticas o en redes neuronales, que sirva para predecir los indicadores de rendimiento de estos vehículos. Con estas predicciones, se pueden detectar errores con mucha mayor antelación, lo que puede permitir tomar medidas que no sería factible tomar de otra manera. De esta forma, se pueden conseguir reducir el número de errores críticos que supongan una parada del sistema, reduciendo considerablemente los costes y aumentando la productividad.

Este proyecto está enfocado a utilizarse en un entorno industrial, por lo que su principal objetivo es claro: reducir costes a partir de una mejora de la eficiencia del funcionamiento de los AGV. Para ello, se desarrollará un sistema capaz de almacenar la información recibida de forma eficiente y capaz de predecir datos futuros con precisión. Por ello, escoger la base de datos más óptima para esta tarea, así como el mejor modelo para realizar las predicciones son tareas esenciales. Es por esto que este trabajo está centrado precisamente en dichas tareas de investigación.

Estructura de la memoria

Esta memoria seguirá la siguiente estructura:

- **Introducción:** contiene una breve descripción del trabajo, así como una guía de contenidos del mismo.
- **Objetivos del proyecto:** detalla los objetivos principales del proyecto.
- **Conceptos teóricos:** explicación de varios conceptos necesarios para una mejor comprensión del proyecto.
- **Técnicas y herramientas:** metodologías y técnicas utilizadas durante todo el desarrollo del trabajo.
- **Aspectos relevantes del desarrollo:** descripción del proceso de desarrollo, con los aspectos más relevantes del mismo.
- **Trabajos relacionados:** exposición y comparativa con otros trabajos relacionados.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas durante el desarrollo, así como ideas a futuro para la mejora del mismo.

Junto a la memoria, se incluye además unos anexos que incluyen:

- **Plan de proyecto:** contiene la planificación temporal y un estudio de viabilidad legal y económica.
- **Requisitos:** describe los requisitos funcionales y no funcionales del sistema, así como una comparativa de gestores de bases de datos y modelos de predicción, basándose en dichos requisitos.
- **Pruebas de rendimiento:** describe la metodología y pruebas realizadas para comparar los diferentes sistemas gestores de bases de datos y modelos de predicción planteados.
- **Diseño:** se define el diseño de los datos, procedimental y arquitectónico del sistema.
- **Manual del programador:** recoge los aspectos más importantes para futuros programadores del sistema.
- **Manual de usuario:** detalla las funcionalidades del proyecto para futuros usuarios del mismo.
- **Publicaciones derivadas:** se incluye el trabajo presentado en el SOCO 2023 [4], el cual está derivado de este proyecto.

Materiales adjuntos

Junto con esta memoria y anexos, se incluye:

- Prototipos. Recoge el conjunto de prototipos realizados durante el desarrollo.
- Análisis de rendimiento. Contiene los programas utilizados para hacer las pruebas de rendimiento de las bases de datos y de los modelos de predicción.
- Código del sistema desarrollado.
- Conjunto de datos de prueba. Se incluye en los directorios de los servicios un conjunto de datos de un AGV real con el que realizar pruebas.

Objetivos del proyecto

El objetivo principal de este trabajo es desarrollar un sistema capaz de almacenar, visualizar y predecir datos producidos por un AGV. Este sistema está planteado para permitir la realización de mantenimiento predictivo o alertas de fallos en el funcionamiento de manera automática en entornos industriales, por lo que su objetivo final será el de reducir costes y mejorar la eficiencia en estos entornos.

Se pretendo conseguir también un sistema eficiente, por lo que será necesario una extensa investigación y comparación de sistemas gestores de bases de datos a utilizar, así como de diferentes modelos predictivos.

Este proyecto ha sido diseñado como un conjunto de microservicios [5]. Las aplicaciones diseñadas de esta manera se dividen en elementos más pequeños e independientes entre sí, y que deben comunicarse y coordinarse. De esta forma, se consigue que la aplicación sea modular, más fácil de escalar, de mantener y de desarrollar.

Al dividir la aplicación en lo que a efectos prácticos son aplicaciones más simples, se consigue un código mucho más sencillo, por lo que el diseño del propio código se simplifica. En nuestro caso, se ha utilizado un enfoque de programación procedimental, en vez de programación orientada a objetos, pues la simplicidad del propio código hace que esta última metodología no sea la más apropiada.

Los objetivos pueden quedar resumidos en la siguiente lista:

- Desarrollar sistema que permita la monitorización de un sistema de AGVs enviando información en tiempo real.

- Diseñar una metodología que sirva como base para realizar una comparativa de sistemas gestores de bases de datos.
- Crear un sistema que almacene de manera eficiente la información, utilizando para ello el sistema gestor de bases de datos que mejor se adapte a los requisitos según la metodología diseñada.
- Realizar predicciones sobre los datos almacenados a diez segundos en el futuro.
- Diseñar una metodología para realizar una comparativa de modelos de predicción de datos, de manera que se escoja el más eficiente.
- Desarrollar el sistema como un conjunto de microservicios que interaccionan entre sí para conseguir una mayor modularidad.
- Diseñar un código mantenible y que se adapte a las convenciones estilísticas establecidas a partir del uso de herramientas de integración continua como CodeClimate o Pylint.
- Mantener un control de las versiones del proyecto a partir de un uso correcto de Git.
- Desarrollar el sistema de manera organizada aplicando la metodología ágil.

Conceptos teóricos

Como se ha mencionado en apartados anteriores, los datos recibidos por un AGV se agrupan en series temporales. Conviene por tanto explicar primero que es exactamente un AGV, que son las series temporales, así como los modelos que se utilizarán para intentar predecirlas.

También, como el proyecto se ha desarrollado como un conjunto de microservicios, se explicará brevemente en que consiste brevemente este estilo de estructurado de aplicaciones.

3.1. Autonomous Guided Vehicles

Un Autonomous Guided Vehicle, o AGV, es un robot que se mueve generalmente por una banda magnética que utiliza como guía, aunque también pueden ubicarse utilizando ondas de radio, visión artificial a través de cámaras, o mediante el uso de láseres.

Su uso principal es el de transporte de cargas pesadas en entornos industriales, como puede ser una fábrica o un almacén. Como estos sistemas se integran en entornos en los que trabajan personas, han de incluirse con rigurosas medidas de seguridad para evitar accidentes.

Generalmente, estos vehículos se organizan en flotas, por lo que puede haber varios vehículos en un mismo circuito. Esto requiere que estos AGV sean capaces de comunicarse y coordinarse entre sí, por lo que tecnologías como el 5G son prácticamente necesarias para su correcto funcionamiento [6].

Debido a su prevalencia en entornos industriales, existen numerosas soluciones diseñadas para optimizar el funcionamiento de dichos vehícu-

los: optimización de las misiones que han de cumplir [7], optimización de trayectorias [8], etc.

Existen también soluciones diseñadas para detectar anomalías en el funcionamiento [9]. Este proyecto se diferencia sin embargo en el uso de predicciones del funcionamiento a partir de datos anteriores, por lo que se propone una solución poco investigada en este aspecto.

3.2. Series temporales

Definiciones

Una serie temporal es una colección de observaciones obtenidas mediante mediciones repetidas a lo largo del tiempo [10].

Normalmente, las series temporales presentan patrones que pueden utilizarse para realizar predicciones. Estos patrones son:

- **Tendencia**. La tendencia existe cuando hay un incremento o decremento del valor medido a largo plazo. Esta tendencia no tiene por qué ser lineal.
- **Estacionalidad**. El patrón de estacionalidad se presenta cuando una serie temporal se ve afectada por factores estacionales, como puede ser el día de la semana. Siempre tiene una frecuencia fija y conocida.
- **Ciclos**. Un ciclo ocurre cuando los datos muestran incrementos o decrementos a una frecuencia no fija.

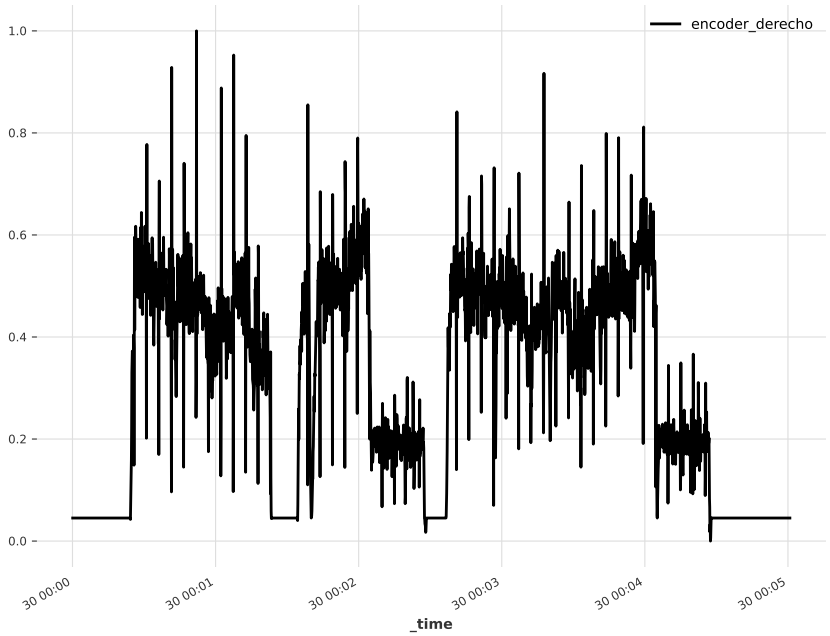


Figura 3.1: Ejemplo de serie temporal

Existen distintos tipos de clasificaciones para las series temporales, según varios puntos de vista [11], de las cuales destacan:

- **Continuas o discretas**. Las series temporales continuas son aquellas en las que la información se obtiene, valga la redundancia, de forma continua, normalmente por un dispositivo analógico, como podría ser los datos recibidos de un sismógrafo. Por otro lado, las series temporales discretas son aquellas en las que la información se obtiene en intervalos de tiempo concretos. Estos intervalos pueden ser equidistantes, o bien ser irregulares. Normalmente, las series temporales medidas por medios digitales son discretas. En nuestro caso, las series temporales enviadas por el AGV son discretas espaciadas en intervalos irregulares, pues el AGV manda dicha información cada varios milisegundos, de manera irregular.

- **Univariantes o multivariantes**. Aquellas series temporales que tengan solo una observación por cada momento del tiempo son series univariantes. Por contra, aquellas en las que se obtengan de manera simultánea mediciones de dos o más fenómenos son multivariantes. Esto

será importante a la hora de escoger que modelo utilizar para realizar predicciones, pues hay modelos que solo soportan series temporales univariantes.

- **Estacionarias o no estacionarias**. Una serie estacionaria [12] es aquella en la que sus propiedades no dependen del momento en el que se observan. Por ello, aquellas que presenten tendencias o estacionalidad no son estacionarias. Por otra parte, una serie de ruido blanco es estacionaria: no importa cuándo se observe, debería tener el mismo aspecto en cualquier momento. De manera general, las series estacionarias no tendrán patrones predecibles en el largo plazo, por lo que conviene convertir las series estacionarias en no estacionarias.

Predicción de series temporales

La predicción de series temporales es una actividad muy importante en muchos sectores: predicción de datos financieros, predicciones del clima, etc. Debido a esto, existe una gran cantidad de modelos para realizar dichas predicciones. Hay que tener en cuenta sin embargo que predecir datos futuros es una tarea especialmente complicada, y no siempre se obtiene una gran precisión.

Los siguientes modelos se tendrán en cuenta en este trabajo:

- **ARIMA.** Este modelo (Autoregressive Integrated Average) [12] es un enfoque estadístico utilizado para el análisis y pronostico de series temporales. Es una combinación de tres componentes principales:
  - Componente autorregresivo (AR): este componente utiliza la información de valores pasados de la serie temporal. Se basa en la idea de que los valores pasados tienen influencia en el futuro. Este modelo indica cuantos valores pasados se utilizan en la predicción.
  - Componente de media móvil (MA): tiene en cuenta el error residual de las predicciones anteriores para mejorar la precisión, haciendo una media de los errores pasados para predecir los futuros. Indica cuantos errores pasados se tienen en cuenta.
  - Componente de Integración (I): se refiere al proceso de diferenciación de la serie temporal para hacerla estacionaria. El orden de Integración indica cuántas veces se diferencia la serie temporal.

  La combinación de estos tres componentes forman el modelo ARIMA(p, d, q), donde “p” representa el orden del componente autorregresivo,

"d" es el orden del componente de integración y "q" es el orden del componente de media móvil. Al tratarse de un modelo estadístico no requiere de entrenamiento como el resto de modelos expuestos. Por ello, resulta interesante compararlo con modelos basados en redes neuronales. Es también un modelo muy extendido y que suele utilizarse de base para realizar pruebas de este tipo.

- **TCN.** Una Red neuronal Convolucional Temporal (Temporal Convolutional Network en inglés) [13] es un tipo de red neuronal utilizada para analizar series temporales. Las TCN tienen en cuenta la estructura temporal de los datos y aplican operaciones convolucionales para capturar patrones. Una convolución [14] es un operador matemático que transforma dos funciones en una nueva, y se define como la integral del producto de ambas funciones después de desplazar una de ellas una distancia t (Fórmula 3.1). Un ejemplo de convolución es la media móvil, o un filtro de aumento de nitidez para imágenes.

$$\int_{-\infty}^{\infty} f(\eta)g(t-\eta)d\eta \tag{3.1}$$

3.1: Convolución

TCN utiliza capas convolucionales de una dimensión para aprender características de la serie temporal. Estas capas son aplicadas sobre ventanas deslizantes de la secuencia para extraer características en diferentes puntos de tiempo. Este modelo se ha escogido debido a que las redes neuronales convolucionales son utilizadas típicamente en problemas de visión artificial o que requieran de tratamiento de imágenes, por lo que pretende comprobarse su utilidad para predecir series temporales.

- **N-HiTS.** Este modelo (Neural Hierarchical interpolation for Time Series) es una extensión del modelo N-BEATS, mejorando su rendimiento y velocidad de entrenamiento [15]. N-BEATS [16] está formado por dos componentes: stack y bloque. Un bloque está formado por una red multicapa que predice valores futuros y pasados. Estos bloques se organizan en pilas (stacks), que agregan las predicciones y errores residuales. Este modelo es interesante porque utiliza "predicciones" del pasado para calcular el error que está teniendo la red al entrenarse con el fin de compensarlo en predicciones futuras. Por ello, se pretende comparar con otras soluciones más típicas y populares.

- **Transformer Model.** El Modelo Transformador [17] es una red neuronal que aprende el contexto de la información. Este modelo usa una arquitectura codificador-decodificador, en la que el codificador procesa la entrada de forma iterativa, y el decodificador hace lo mismo con la salida del codificador. Este modelo está siendo muy utilizado en la actualidad, especialmente en modelos de generadores de lenguaje como GPT (Generative Pre-Trained Transformer). Al escoger este modelo, se pretende por tanto comprobar su viabilidad para la predicción de series temporales.

Estos modelos serán los utilizados para la comparación en el desarrollo del proyecto. Para ello, se utilizan las siguientes métricas:

- **MAE**. Siglas de Mean Absolute Error [18], o Error Absoluto Medio en español. Mide el error medio de una predicción sin tener en cuenta la dirección de dicho error.
- **MASE**. El MASE [19] (Mean Absolte Scaled Error), o error absoluto escalado medio, mide como de bueno es el modelo comparado con un modelo "ingenuo" (modelo que predice un valor como su valor previo). Un valor por encima de 1 significa que nuestro modelo es peor que dicho modelo ingenuo.
- **DTW**. Siglas de Dynamic Time Warping [20], es utilizado para medir la similitud entre dos series temporales. Por ejemplo, una predicción con forma de ecuación lineal puede tener el mismo MAE que otra predicción más irregular, pero esta segunda puede parecerse más a los datos reales.

3.3. Microservicios

Un microservicio es una pequeña aplicación que puede ser desplegada, escalada y probada de manera independiente, además de caracterizarse por tener una única responsabilidad. De esta manera, las aplicaciones diseñadas siguiendo este tipo de arquitectura están formadas por un conjunto de microservicios, en vez de desarrollarse de manera monolítica.

La arquitectura de microservicios es una arquitectura orientada a servicios, es decir, que dicta cómo deben trazarse los límites de los servicios, y una en la que la independencia en la capacidad de despliegue es el concepto clave. Esta manera de organizar las aplicaciones permite una mayor modularidad,

lo que a su vez facilita su mantenimiento y escalabilidad. Permite también desarrollar y desplegar aplicaciones de manera mucho más rápida, ya que modificar la implementación de uno de estos microservicios no debería tener ningún efecto sobre el resto.

Algunos conceptos clave de los microservicios son [21]:

- Capacidad de despliegue independiente: esta idea consiste en que pueden realizarse cambios sobre un microservicio, desplegarlo y lanzar dicho servicio a nuestros usuarios, sin necesidad de tener que desplegar ningún otro microservicio.
- Modelado en torno a un dominio de negocio: cada microservicio debe tener una única responsabilidad, de esta manera se simplifica el desarrollo del mismo, así como la combinación de diferentes microservicios para entregar nuevas funcionalidades.
- Tamaño: ya que no existe una buena forma de medir el tamaño del software, como norma general se dice que el tamaño de un microservicio debe mantenerse en un tamaño en el que pueda ser fácilmente entendido.
- Flexibilidad: al dividir la aplicación en un conjunto de microservicios independientes, la flexibilidad aumenta.

Idealmente, estos microservicios se ejecutan de manera aislada del resto. De esta forma, un fallo en uno de ellos no afecta al resto. Para ello, se recurren a técnicas de virtualización que crean entornos de ejecución aislados. Sin embargo, las técnicas clásicas de virtualización, como las Máquinas Virtuales, suelen ser demasiado pesadas cuando se tiene en cuenta el tamaño de los microservicios. Por ello, se utilizan Contenedores, como los proveídos por Docker [22]. Estos contenedores proveen entornos de ejecución independientes, igual que las Máquinas Virtuales, pero son mucho más ligeros.

En cuanto a las ventajas ofrecidas por los microservicios, se pueden destacar las siguientes:

- Heterogeneidad de la tecnología: como los microservicios son independientes entre sí, podemos escoger tecnologías diferentes para cada uno de ellos, permitiendo así escoger la herramienta adecuada para cada tarea.

- Robustez: ya que los errores en un servicio no se propagan al resto, se consigue un sistema mucho más robusto.
- Escalado: en aplicaciones monolíticas, todo a de ser escalado en conjunto. Sin embargo, al dividir la aplicación en partes más pequeñas, podemos simplemente escalar aquella parte que lo necesite.
- Facilidad del despliegue: un cambio en una aplicación monolítica requiere de volver a desplegar toda la aplicación, lo que puede ser una operación peligrosa. Un cabio en un microservicio, sin embargo, solo provoca un nuevo despliegue en dicho servicio.
- Productividad: ya que los microservicios son pequeños en tamaño, es mucho más fácil coordinar a un grupo de desarrolladores para trabajar en dicha aplicación.

Técnicas y herramientas

4.1. Metodologías

Las siguientes metodologías han sido utilizadas:

- **SCRUM** es un marco de trabajo cuyas características principales son: desarrollo incremental en lugar de una ejecución completa, basar la calidad en el conocimiento de los integrantes del equipo en vez de en la calidad de los procesos y solapar las fases del proyecto.

  En SCRUM, el trabajo se divide en sprints, periodos de tiempo de entre 1 y 4 semanas. Al principio de cada sprint se planifica el trabajo a desarrollar, y se mantienen reuniones diarias que sirven para mantener la comunicación entre desarrolladores. Al final del sprint se tiene una última reunión en la que se analiza el trabajo realizado.

  En nuestro caso, los sprints han sido de entre 1 y 2 semanas, y no se han tenido las reuniones diarias.

- **Pomodoro** es un método diseñado para mejorar la productividad mediante una correcta gestión del tiempo. Se establece un tiempo de trabajo con una duración de entre 20 y 30 minutos, y un tiempo de descanso de 5 minutos. Después de cuatro ciclos de trabajo/descanso, se realiza un descanso más largo, de unos 15 a 30 minutos.

  Este método ha sido utilizado especialmente a la hora de escribir esta memoria y los anexos.

4.2. Herramientas y librerías

Desarrollo

El proyecto ha sido desarrollado íntegramente en el lenguaje de programación Python en su versión 3.10.6.

Además, todos los módulos que más tarde se explican han sido desarrollados como contenedores de Docker. Para ello se han utilizado:

- Docker [22], en su versión 24.0.2. Docker permite el despliegue de aplicaciones dentro de contenedores, similares a máquinas virtuales, de forma que cada contenedor está aislado del resto del sistema. Docker permite también la ejecución de dichas aplicaciones de forma independiente al sistema operativo, por lo que permite una gran compatibilidad.

- Docker compose [23] versión 2.16.0. Esta herramienta sirve para definir y ejecutar aplicaciones multi-contenedor.

Sistema Gestor de Base de Datos

El sistema gestor de bases de datos escogido ha sido InfluxDB. Otros gestores planteados para su uso han sido TimescaleDB, Prometheus, Graphite y kdb+.

InfluxDB es un sistema gestor de bases de datos de series temporales. En InfluxDB, los datos se guardan en series. Una serie es un conjunto de puntos que comparten:

- Medida (Measurement): equivalente en SQL a una tabla.

- Conjunto de etiquetas (tag set): equivalente en SQL a una columna indexada. Solo pueden ser cadenas de texto. Sirven para almacenar metadatos.

- Conjunto de valores (field set): equivalente en SQL a una columna sin indexar. Guardan los datos.

Por último, estas series se guardan en “Buckets”, que son el equivalente en SQL a una base de datos.

Predicción

Como librería utilizada para la predicción de las series temporales se ha utilizado Darts [24]. Se consideraron otras alternativas, como Loud ML, herramienta desarrollada específicamente para InfluxDB, y Facebook Prophet. La primera fue descartada debido a que el proyecto está abandonado, y la segunda fue descartada debido a que Darts incluye muchos más modelos, incluido el mismo Prophet.

Se ha utilizado también la librería Optuna[25] para la optimización de los hiperparámetros de los modelos de predicción.

4.3. Entorno de desarrollo

Todo el trabajo ha sido realizado con el editor de texto de Microsoft Visual Studio Code [26]. Esto ha sido así por varios motivos, principalmente mi comodidad con él, pues es el editor de texto que estoy acostumbrado a usar, y la gran cantidad de extensiones existentes que permitan una mejor experiencia de usuario.

En cuanto a las extensiones utilizadas, caben destacar:

- Docker: utilizada para un manejo más simple de los contenedores de Docker.
- Python: extensión esencial para el desarrollo en Python, pues ofrece características como un Debugger, resaltado de la sintaxis, IntelliSense (autocompletado), etc.
- Todo Tree: utilizada para marcar elementos no terminados, de forma que luego sean fáciles de encontrar.
- LaTeX Workshop: permite compilar de forma automática al guardar los archivos de LaTeX, autocompletado, etc.
- LTeX - LanguageTool grammar/spell checkingLTex: como su nombre en inglés indica, esta extensión analiza errores gramaticales en los archivos de LaTeX.

4.4. Otras herramientas

Control de versiones

El proyecto se aloja en la plataforma GitHub [27], y la herramienta utilizada para el control de versiones es git. Inicialmente se empezó a usar ZenHub [28] para llevar el control de los sprints. Sin embargo, debido a un problema de licencias a mitad del desarrollo se dejó de utilizar en favor de simplemente usar el apartado de Issues del repositorio para dicha labor.

Documentos

La memoria y los anexos han sido desarrollados en LaTeX, utilizando la distribución TeX Live[29] para la compilación de dichos documentos.

Análisis de código

Se han utilizado dos herramientas para el análisis estático del código:

- Codeclimate [30]: Analiza la mantenibilidad del código, cobertura de pruebas, etc.
- Pylint [31]: Analiza y puntúa el estilo del código.

Estas herramientas se integran de forma que se ejecutan cada vez que se hace un nuevo commit en el repositorio del proyecto. Se pueden ver también en el README los resultados de dichas pruebas.

Aspectos relevantes del desarrollo del proyecto

El desarrollo de este proyecto puede decirse que ha estado dividido principalmente en dos partes. La primera fase estuvo relacionada con el diseño de la arquitectura, así como la elección del sistema gestor de bases de datos. En la segunda fase, se diseñó todo lo relativo a la predicción de datos y a la elección de las herramientas necesarias para dicha tarea.

5.1. Diseño de la arquitectura y comparativa de los gestores de bases de datos

Como ha sido mencionado en el párrafo anterior, en la primera fase se llevó a cabo el diseño de la arquitectura del sistema, así como el desarrollo de los servicios que forman dicho sistema. Además, se ha realizado una comparativa de diferentes sistemas gestores de bases de datos con el fin de elegir el que mejor se adapte a nuestros requisitos.

Esta parte del desarrollo del proyecto ha sido publicada en la conferencia SOCO 2023 [4]. Esta publicación se incluye también en la sección G de los anexos.

Diseño de la arquitectura

La arquitectura del sistema estudiado se representa en la figura 5.1. El sistema está formado por los siguientes sistemas software:

AGV Coordinator Es un servicio encargado de recibir la información enviada por los AGV a través de una conexión 5G/Wifi. Esta información está codificada como una cadena de bytes, que es decodificada y transformada a formato JSON. Estos mensajes son enviados al servicio "Receiver". Este servicio ha sido desarrollado como parte de otro proyecto, por lo que no se detallará en este.

Simulator Es, como su nombre en inglés indica, un servicio encargado de simular un AGV en caso de no disponer de AGV reales y sea necesario realizar pruebas.

Receiver Este servicio recibe mensajes a través del protocolo UDP bien del "AGV Coordinator" o bien del simulador, y se encarga de insertar dichos datos en la base de datos.

Database Como su nombre indica, este servicio será la base de datos encargada de almacenar todos los datos recibidos.

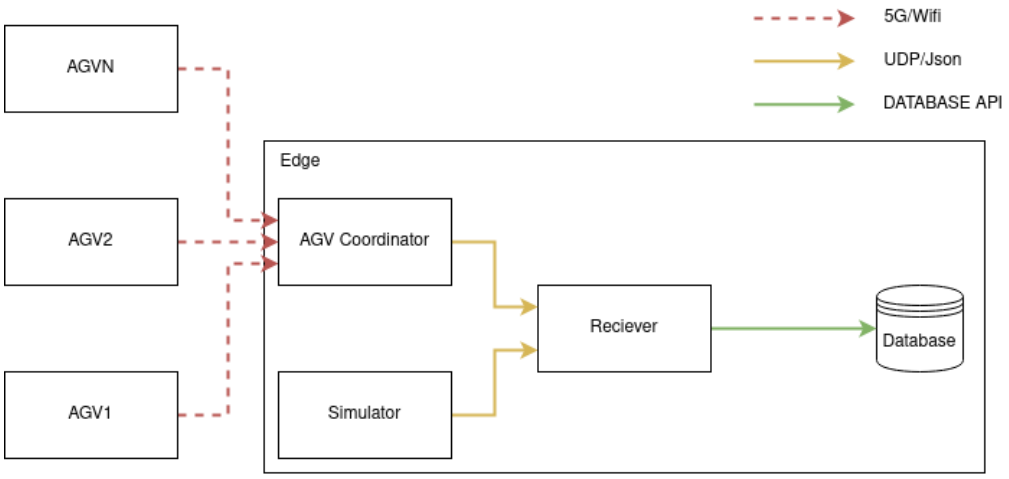


Figura 5.1: Diseño de la arquitectura

Comparativa de Sistemas Gestores de Bases de Datos

Inicialmente, se seleccionaron los cinco sistemas gestores de bases de datos para series temporales más populares según el ranking DB-Engines [32]:

1. InfluxDB

2. Prometheus
3. Kdb+
4. Graphite
5. TimescaleDB

A partir de esta lista de gestores, se ha realizado una comparación exhaustiva de los mismos según los requisitos del proyecto. Esta comparativa puede verse en el apartado C.1 de los anexos que se incluyen de forma complementaria a esta memoria.

Como resumen de dicha comparativa, tanto InfluxDB como TimescaleDB cumplen con los requisitos especificados. Finalmente, el sistema escogido ha sido InfluxDB debido a su menor uso de recursos con un rendimiento suficiente y a la implementación por defecto de una herramienta de visualización de la información.

Desarrollo de los servicios

Una vez diseñada la arquitectura y el sistema gestor de bases de datos escogido, se procedió a desarrollar los sistemas definidos. Todos estos sistemas se ejecutan cada uno en su contenedor de Docker independiente, los cuales se comunican entre sí a través de una red especificada en el archivo de configuración de docker compose.

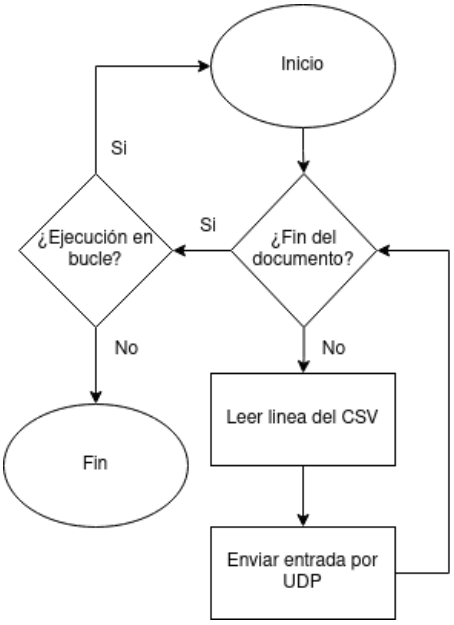
Simulator

Inicialmente, no disponía de datos reales del AGV, por lo que la primera versión (Figura 5.2a) simplemente generaba datos aleatorios con campos aleatorios, y los enviaba al nodo "Receiver" por UDP utilizando el puerto 5004.

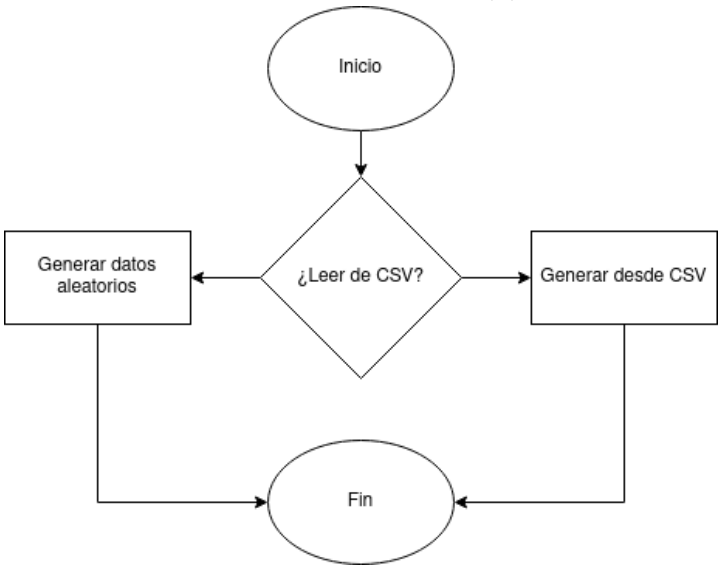
Después, se intentó desarrollar un simulador capaz de, valga la redundancia, simular el comportamiento de un AGV. Sin embargo, una vez dispuse de datos reales del AGV, esta idea se descartó, pues simular dicha información de forma precisa iba a ser demasiado complejo, y se escapa del objetivo de este proyecto. Por tanto, se decidió simular el comportamiento del AGV leyendo los datos de un CSV (Figura 5.2b) obtenido a partir de uno real.

Por último, en la versión final se unificaron los dos procedimientos, de forma que el comportamiento del simulador se decide según lo especificado en un archivo de configuración.

(a) Generación aleatoria



(b) Generación por CSV



(c) Versión final

Figura 5.2: Diagramas de flujo del simulador

Receiver

El funcionamiento del nodo "Receiver" es muy simple (Figura 5.3): escucha el puerto UDP 5004, y cuando recibe información, la inserta en la base de datos. Inicialmente, el servicio intentará conectarse a la base de datos. Si esta conexión falla un número determinado de veces, el servicio fallará informando de que la conexión no ha podido realizarse.

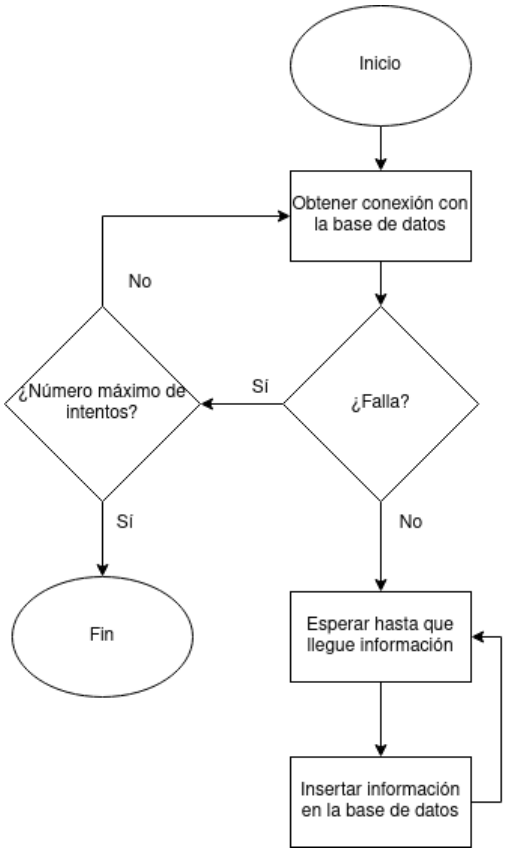


Figura 5.3: Diagrama de flujo del servicio "Receiver"

Database

Este servicio simplemente inicia un contenedor de Docker con la base de datos. Es necesario sin embargo especificar las credenciales del usuario administrador que tendrá los permisos necesarios para insertar información en dicha base de datos.

5.2. Modelo de predicción de datos

La segunda fase se centró en el desarrollo y diseño del servicio de predicción de nuevos datos. Se ha realizado un análisis comparativo de diferentes modelos predictivos, así como un análisis de los datos recibidos por el AGV para hacer la mejor elección.

Modificación de la arquitectura

Ya que se ha pretendido desarrollar una arquitectura lo más modular posible, la predicción de la información se realiza desde un nuevo servicio. Por ende, la arquitectura modificada queda de la siguiente manera:

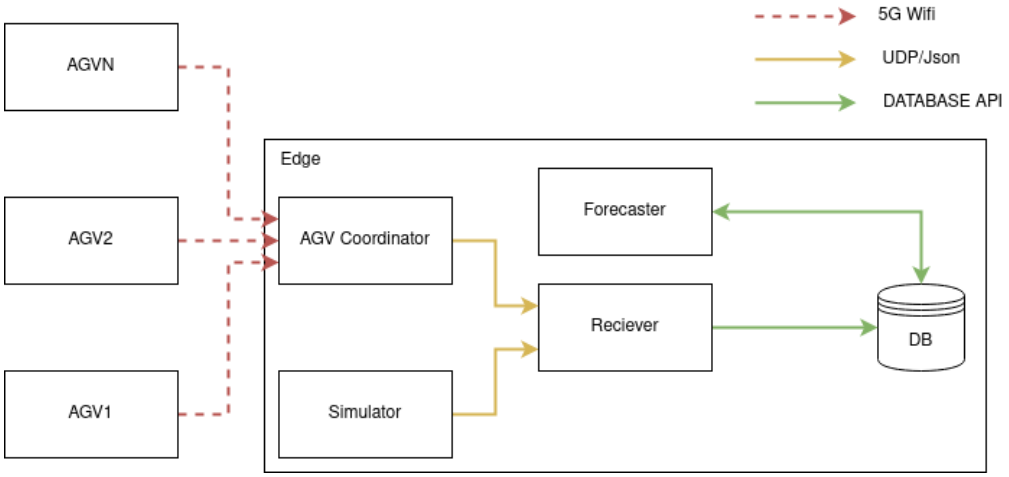


Figura 5.4: Arquitectura final

Este nuevo servicio se conecta directamente a la base de datos de la que extrae la información necesaria para hacer las predicciones. Una vez hechas se insertan en la base de datos para poder ser visualizadas en la utilidad web de la base de datos, Chronograf [33].

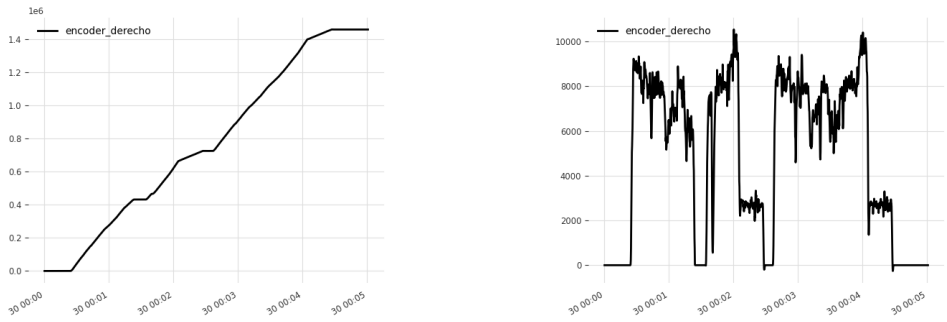
Análisis de los datos

Antes de hacer nada relativo a la predicción de los datos, necesitamos conocer que datos tenemos y cuáles queremos predecir. Del AGV recibimos los siguientes datos:

- Tiempo: tiempo que ha pasado en segundos desde que se inicia el AGV.

- Encoder derecho: valor del codificador de la rueda derecha. Cuando el AGV avanza este valor se incrementa, y cuando retrocede se decrementa.
- Encoder izquierdo: igual al anterior, pero con la rueda izquierda.
- Corriente: dividido en límites inferior y superior.
- Medida batería: corriente medida en la batería.
- Error de guiado: distancia que el AGV está desviado de la banda magnética por la que se guía.
- Consigna de velocidad derecha: valor de la velocidad enviado a la rueda derecha.
- Consigna de velocidad izquierda: similar al anterior pero con la rueda izquierda.
- Display: valor recibido con información del AGV.

Cabe mencionar que, como los encoders derecho e izquierdo no nos muestran directamente la velocidad de las ruedas (Figura 5.5a), antes hay que preprocesar dichos datos, restando cada valor con su anterior, obteniendo así el dato de la velocidad (Figura 5.5b).



(a) Sin procesar (b) Después de diferenciación

Figura 5.5: Datos del encoder derecho

El objetivo es realizar una predicción de los valores encoder derecho y encoder izquierdo para poder compararlos con los obtenidos y detectar posibles errores. Para ello, utilizamos valores anteriores de los mismos datos, así como los valores de las consignas de velocidad derecha e izquierda, pues

su correlación es muy alta. Para predecir los valores derechos, y viceversa, se usan también los valores izquierdos, pues aunque su correlación no sea tan alta, también están correlacionados. Ya que estos datos se mandan en intervalos de tiempo irregulares, el primer paso es agruparlos en ventanas de 200 milisegundos. Esto es así porque para realizar las predicciones es necesario que los datos de la serie temporal estén distribuidos de manera uniforme.

Las pruebas realizadas a los modelos han sido realizadas únicamente con el encoder derecho, pues al ser muy similar al encoder izquierdo se van a obtener resultados prácticamente idénticos en los dos casos, por lo que no merece la pena realizar las pruebas sobre los dos datos.

Comparativa de modelos de predicción

Al igual que con la comparativa de sistemas gestores de bases de datos, una comparativa exhaustiva de los modelos de predicción puede encontrarse en la sección C.2 de los anexos complementarios.

De dicha comparación se sacan las siguientes conclusiones:

- La optimización de los modelos mejora ligeramente los resultados a costa de subir tanto el tiempo de ajuste como el de predicción.
- El mejor modelo para predicciones a largo plazo es el Transformer por defecto y con covariables, o N-HiTS con optimización y con multivariables. Aunque la optimización de este primer modelo ofrece mejores resultados en alguna de las métricas, no son lo suficientemente buenas como para justificar el tiempo de entrenamiento extra.
- Para predicciones a corto plazo, ARIMA resulta el mejor modelo.
- El conjunto de datos no es lo suficientemente grande.

Ya que nuestra intención es realizar predicciones a relativamente largo plazo, el modelo escogido ha sido el modelo Transformer. Otro de los motivos por el que se escoge este modelo es que, al usar covariables en vez de multivariables, su implementación resulta más sencilla.

Desarrollo del servicio

Como se puede ver en la siguiente imagen (Figura 5.6), este servicio carga un modelo desde un archivo o bien lo entrena desde cero. Para poder

entrenar dicho modelo, se necesita que la base de datos ya tenga datos cargados, por lo que la rutina de entrenamiento espera algunos minutos antes de empezar a entrenar para que haya los datos necesarios para poder realizar dicha tarea. La cantidad de tiempo a esperar antes de entrenar se especifica en un archivo de configuración.

En el caso de cargar el modelo desde un archivo, también es necesario esperar un tiempo a que la base de datos se cargue de información, pues es necesario escalar los datos antes de hacer la predicción. Dicha escala se ajusta según los datos introducidos, por lo que cuanto más tiempo esperamos en este paso, más parecida será a la especificada a la hora de entrenar el modelo.

Los datos generados en la predicción son insertados a un "bucket" de la base de datos diferente al que están los datos. Este "bucket" tiene un periodo de retención bajo. Esto significa que los datos insertados se eliminan después de cierto tiempo, pues no nos interesa guardar predicciones antiguas.

Existen dos versiones de este servicio: una utiliza una GPU de Nvidia para entrenar y la otra utiliza la CPU. Entrenar con GPU es mucho más rápido que con CPU, por lo que siempre que se disponga de una es muy recomendable utilizar esta versión. En caso de no disponer de una GPU, o de disponer de una que no soporte la versión 11.8 de CUDA, siempre puede utilizarse el servicio complementario.

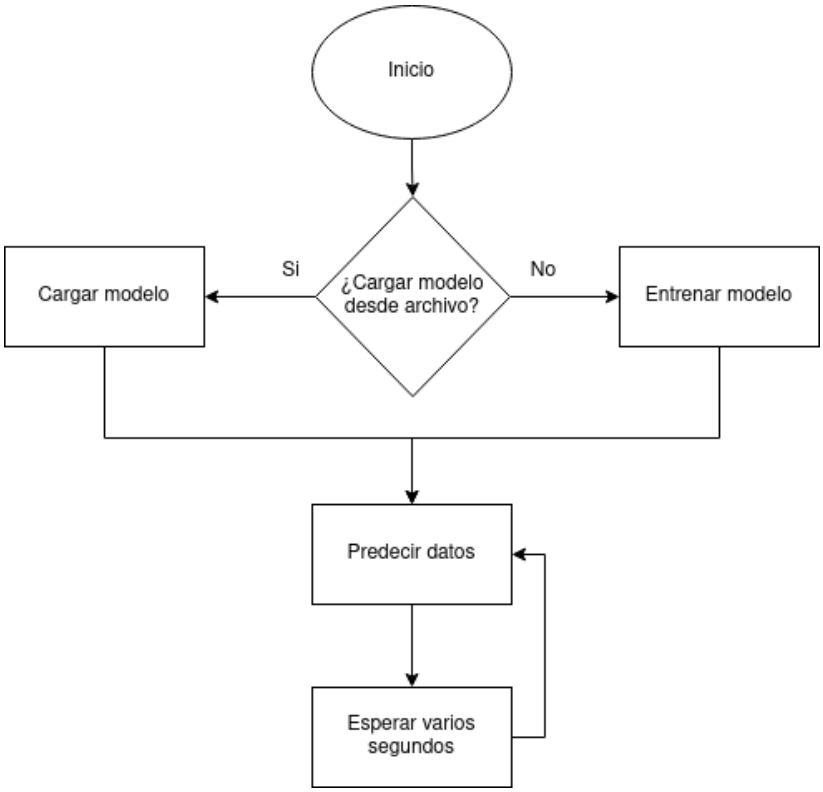


Figura 5.6: Diagrama del servicio de predicción.

5.3. Despliegue del proyecto

Cada servicio queda definido en un archivo "Dockerfile", que definirá el contenido y acciones de cada uno de los contenedores.

Todos estos servicios se inician utilizando la herramienta docker compose. Con esta herramienta podremos crear una red virtual a la que se conectan los diferentes servicios para poder comunicarse entre sí, especificar los puertos que usa cada contenedor, variables de entorno, volúmenes, etc. Con esta herramienta podemos también ver los registros de cada contenedor, lo que nos permitirá ver posibles errores y trazas de información.

Para interaccionar con la aplicación, se accede a una aplicación web contenida en el servicio "Database" llamada Chronograf [33]. Esta aplicación se aloja en el puerto 8086, por lo que, suponiendo que el sistema se ejecuta de forma local, se accede a través de la url "localhost:8086". En caso de ejecutarse en un servidor remoto, basta con cambiar "localhost" por la IP o nombre de dominio de dicho servidor.

Figura 5.7: Plantilla con los gráficos con las predicciones

En la Figura 5.7 podemos observar dicha utilidad, en la que se muestran las predicciones realizadas en naranja y los datos recibidos por el AGV en azul. Una explicación en detalle de como acceder y utilizar esta interfaz se da en la sección F.4 de los anexos.

5.4. Mantenimiento del código

Al final del desarrollo, se integraron en el proyecto las herramientas de integración continua Code Climate [30] y Pylint [31]. Gracias a estas herramientas, se localizaron y resolvieron errores estilísticos y de mantenibilidad del código.

Trabajos relacionados

En un entorno industrial, aumentar la eficiencia de los sistemas utilizados es un proceso esencial, pues permite mejorar la productividad de los procesos y reducir costes. Por ello, hay una gran cantidad de artículos científicos que tratan temas de mantenimiento predictivo utilizando técnicas como el aprendizaje automático.

A continuación, se muestran varios artículos que tratan este problema, ya sea en entornos en los que se utilicen AGV, o entornos puramente industriales.

6.1. Comparativas de sistemas gestores de bases de datos

A Comparison Of Relational, NoSQL and NewSQL Database Management Systems For The Persistence Of Time Series Data Este trabajo [34] evalúa diferentes sistemas de gestión de bases de datos en el contexto del almacenamiento de datos relacionados con el tiempo utilizando diferentes modelos de datos, como modelos relacionales clásicos, modelos no relacionales que utilizan sistemas de bases de datos NoSQL y el grupo de bases de datos NewSQL de reciente aparición. La evaluación muestra que una base de datos de series temporales altamente optimizada como InfluxDB es capaz de superar a los otros sistemas probados en cuanto a rendimiento de escritura y utilización de RAM y disco en una configuración de servidor único.

Evaluation of modern tools and techniques for storing time-series data Según los autores [35], este trabajo se centra en la importancia creciente del análisis y las aplicaciones de los datos de series temporales

en diversas áreas y dominios. Se menciona que muchos campos científicos e industriales dependen del almacenamiento y procesamiento de grandes cantidades de series temporales, como la economía y las finanzas, la medicina, Internet de las Cosas, la protección ambiental, el monitoreo de hardware, entre otros. El objetivo de este trabajo es presentar un enfoque teórico y experimental para seleccionar una herramienta apropiada para el análisis de series temporales.

6.2. Modelos de predicción en entornos industriales

Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry En este artículo [36], se realiza un estudio y clasificación de estudios que proponen la aplicación de aprendizaje automático para la realización de mantenimiento predictivo de en el campo de la industria automovilística. Se proponen retos a resolver, así como posibles direcciones de investigación. Las conclusiones obtenidas finalmente son que disponer de datos públicos impulsaría las actividades de investigación, la mayoría de los artículos se basan en métodos supervisados que requieren datos etiquetados, la combinación de múltiples fuentes de datos puede mejorar la precisión, y el uso de métodos de aprendizaje profundo seguirá aumentando, pero requiere métodos eficientes e interpretables y la disponibilidad de grandes cantidades de datos.

Availability assessment for a multi-AGV system based on simulation modeling approach Este artículo [37] propone el desarrollo de una simulación de Monte Carlo de un sistema multi-AGV utilizado para evaluar la disponibilidad del sistema. El análisis de sensibilidad para investigar las relaciones directas entre el mantenimiento y los parámetros operativos y el nivel de ratio de disponibilidad se realiza basándose en el modelo desarrollado. El análisis detallado de los logros en este ámbito permite identificar las lagunas en la investigación y las posibles líneas de investigación futuras para los procesos de diseño de almacenes autónomos.

Combining empirical mode decomposition and deep recurrent neural networks for predictive maintenance of lithium-ion battery Los autores proponen [38] un método híbrido de ciencia de datos basado en la descomposición empírica de modos, el análisis relacional gris y las redes neuronales recurrentes profundas para la predicción de la vida útil restante

de las baterías de iones de litio. Los resultados experimentales obtenidos con los datos de las baterías de iones de litio del Repositorio de Datos de Prognosis Ames de la NASA muestran que el modelo híbrido de ciencia de datos propuesto puede predecir con precisión el estado de salud y la vida útil restante de las baterías de iones de litio.

Novel methodology for optimising the design, operation and maintenance of a multi-AGV system Según los autores [39], los problemas de fiabilidad y las estrategias de mantenimiento de los AGV no se han estudiado suficientemente, por lo que han realizado una investigación considerando un sistema multi-AGV, compuesto por tres AGVs, con el fin de desarrollar una metodología científica para optimizar el diseño del layout, la operación y el mantenimiento de un sistema multi-AGV. Los resultados de simulación obtenidos muestran claramente que la ubicación de los puntos de mantenimiento y las estrategias de mantenimiento tienen una influencia significativa en el rendimiento de un sistema multi-AGV, donde el mantenimiento correctivo es una medida eficaz para mantener la fiabilidad y estabilidad del sistema a largo plazo.

6.3. Otros proyectos

Si bien existen soluciones como Amazon Forecast encargadas de hacer predicciones sobre datos, hasta donde se ha podido observar, no existen soluciones de software dedicadas a la predicción de comportamientos específicamente de AGV, por lo que este proyecto ofrece un sistema original e innovador en su campo.

Conclusiones y líneas de trabajo futuras

A continuación, se exponen las conclusiones obtenidas tras la finalización del proyecto, así como mejoras a realizar en el futuro.

7.1. Conclusiones

El objetivo general del proyecto ha sido cumplido de manera satisfactoria: se ha conseguido el desarrollo de un sistema capaz de almacenar los datos recibidos por un AGV o por el simulador y de llevar a cabo predicciones precisas que permitan realizar mantenimiento predictivo.

Se ha desarrollado un sistema modular, basado en el desarrollo de microservicios independientes entre sí. Gracias a esto, se ha conseguido también que el sistema se adapte a las limitaciones de hardware del cliente.

Gracias a la utilización de Python y Docker, se ha conseguido crear un sistema sobre el que es muy fácil de realizar modificaciones a futuro: Python agiliza mucho el desarrollo al ser un lenguaje interpretado, que no necesita de compilaciones que resten tiempo al proceso de despliegue, y Docker permite modificar solo aquellos servicios en los que haya cambios.

Durante el desarrollo del proyecto, se han utilizado metodologías y conocimientos obtenidos durante el grado. Se ha seguido una metodología ágil para el desarrollo, se han aprovechado los conocimientos obtenidos sobre bases de datos para la elección del sistema gestor de bases de datos, se han utilizado los conocimientos obtenidos sobre inteligencia artificial y aprendizaje automático, etc.

El desarrollo del proyecto ha servido también para ampliar mis conocimientos en dichos campos. Me ha servido también para adquirir nuevos conocimientos, como herramientas de Integración continua, nuevos métodos de aprendizaje automático, uso de contenedores de Docker para la creación de microservicios, etc.

Por el trabajo de investigación realizado durante el desarrollo del proyecto, mi capacidad para buscar información de calidad ha mejorado de manera sustancial, así como mi capacidad para leer y redactar documentos científicos.

Quiero destacar también la labor realizada por mis tutores, guiándome de manera efectiva durante todo el desarrollo del trabajo. También, por sugerencia suya, una parte de este trabajo ha sido publicado en la conferencia SOCO 2023 [4].

7.2. Líneas de trabajo futuras

Con el fin de mejorar la modularidad del proyecto, se sugieren los siguientes cambios:

- Crear un nuevo servicio que haga de intermediario para otros servicios que quieran interaccionar con la propia base de datos (Figura 7.1). Para ello, se propone el desarrollado de una API Rest, mediante la cual se hagan las peticiones a este servicio para hacer consultas e inserciones. De esta forma, se consigue desacoplar los diferentes servicios de la base de datos, lo que resultaría en un diseño mucho más modular, y que simplificaría mucho el desarrollo de nuevos servicios en el futuro.
- Modificar el servicio "Forecaster", para que pueda detectar comportamientos anómalos de los AGV de manera automática utilizando la detección de anomalías de Darts [40], herramienta utilizada para realizar las predicciones.
- Modificar el servicio "Simulator" de manera que permita simular más de un AGV. De manera similar, se sugiere modificar también el servicio "Forecaster" para que pueda realizar predicciones de varios vehículos.
- Ejecutar la optimización de los modelos de predicción comparados durante más tiempo. Por limitaciones de plazos, no fue posible optimizar dichos modelos durante mucho tiempo, por lo que todavía se pueden obtener mejores modelos potenciales.

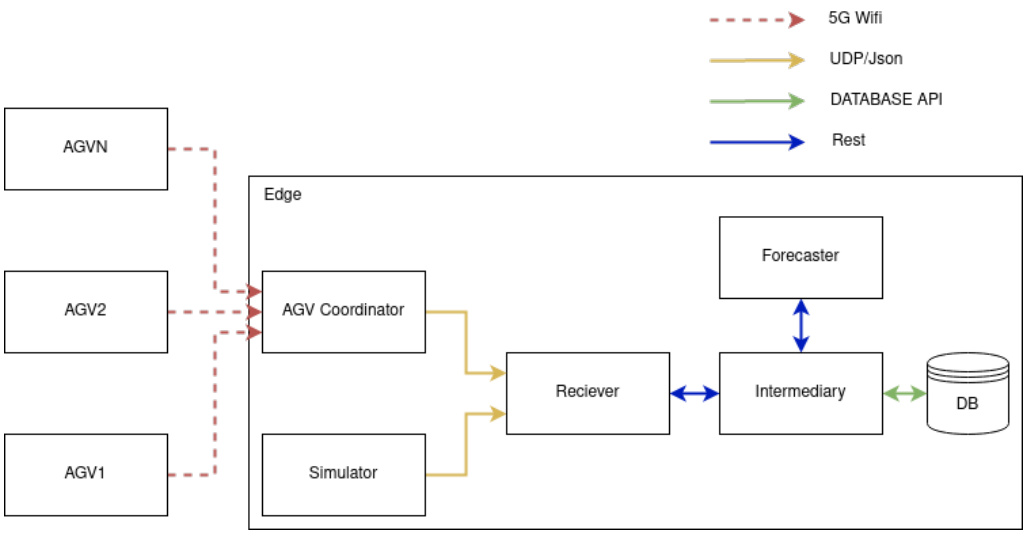


Figura 7.1: Diseño futuro de la arquitectura

Bibliografía

[1] F Espinosa, C Santos y JE Sierra-García. "Transporte multi-AGV de una carga: estado del arte y propuesta centralizada". En: *Revista Iberoamericana de Automática e Informática industrial* 18.1 (2020), págs. 82-91.

[2] Bruno Baruque et al. "Geothermal heat exchanger energy prediction based on time series and monitoring sensors optimization". En: *Energy* 171 (2019), págs. 49-60. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2018.12.207. URL: https://www.sciencedirect.com/science/article/pii/S0360544218325817.

[3] Fatoumata Dama y Christine Sinoquet. "Analysis and modeling to forecast in time series: a systematic review". En: *CoRR* abs/2104.00164 (2021). arXiv: 2104.00164. URL: https://arxiv.org/abs/2104.00164.

[4] Gonzalo Burgos, Enrique Sierra-García y Bruno Baruque-Zanón. "Comparative study of open source database management systems to enable predictive maintenance of Autonomous Guided Vehicles". En: *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)*. Ed. por Pablo García Bringas et al. Cham: Springer Nature Switzerland, 2023. ISBN: Pending Publication.

[5] Johannes Thönes. "Microservices". En: *IEEE Software* 32.1 (2015), págs. 116-116. DOI: 10.1109/MS.2015.11.

[6] Rafal Cupek et al. "Autonomous Guided Vehicles for Smart Industries – The State-of-the-Art and Research Challenges". En: *Computational Science – ICCS 2020*. Ed. por Valeria V. Krzhizhanovskaya et al.

Cham: Springer International Publishing, 2020, págs. 330-343. ISBN: 978-3-030-50426-7.

[7] Elias K. Xidias y Philip N. Azariadis. "Mission design for a group of autonomous guided vehicles". En: *Robotics and Autonomous Systems* 59.1 (2011), págs. 34-43. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2010.10.003. URL: https://www.sciencedirect.com/science/article/pii/S0921889010001715.

[8] Tim Mercy, Ruben Van Parys y Goele Pipeleers. "Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment". En: *IEEE Transactions on Control Systems Technology* 26.6 (2018), págs. 2182-2189. DOI: 10.1109/TCST.2017.2739706.

[9] Ozan Çatal et al. "Anomaly Detection for Autonomous Guided Vehicles using Bayesian Surprise". En: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, págs. 8148-8153. DOI: 10.1109/IROS45743.2020.9341386.

[10] James D Hamilton. *Time series analysis*. Princeton university press, 2020.

[11] Genshiro Kitagawa. *Introduction to time series modeling*. CRC press, 2010.

[12] Rob J. Hyndman y George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd. OTexts.com/fpp2. Melbourne, Australia: OTexts, 2018.

[13] Shaojie Bai, J. Zico Kolter y Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". En: *CoRR* abs/1803.01271 (2018). arXiv: 1803.01271. URL: http://arxiv.org/abs/1803.01271.

[14] Isidore Isaac Hirschman y David V Widder. *The convolution transform*. Courier Corporation, 2012.

[15] Cristian Challu et al. "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting". En: *CoRR* abs/2201.12886 (2022). arXiv: 2201.12886. URL: https://arxiv.org/abs/2201.12886.

[16] Boris N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". En: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=r1ecqn4YwB.

[17] Ashish Vaswani et al. "Attention Is All You Need". En: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[18] Alexei Botchkarev. "A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms". En: *Interdisciplinary Journal of Information, Knowledge, and Management* 14 (2019), págs. 045-076. DOI: 10.28945/4184. URL: https://doi.org/10.28945%2F4184.

[19] Rob J Hyndman et al. "Another look at forecast-accuracy metrics for intermittent demand". En: *Foresight: The International Journal of Applied Forecasting* 4.4 (2006), págs. 43-46.

[20] "Dynamic Time Warping". En: *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, págs. 69-84. ISBN: 978-3-540-74048-3. DOI: 10.1007/978-3-540-74048-3_4. URL: https://doi.org/10.1007/978-3-540-74048-3_4.

[21] Sam Newman. *Building microservices*. .º'Reilly Media, Inc.", 2021.

[22] Docker. *Docker*. 2023. URL: https://www.docker.com/.

[23] Docker. *Docker Compose overview*. 2023. URL: https://docs.docker.com/compose/.

[24] Julien Herzen et al. "Darts: User-Friendly Modern Machine Learning for Time Series". En: *Journal of Machine Learning Research* 23.124 (2022), págs. 1-6. URL: http://jmlr.org/papers/v23/21-1177.html.

[25] Preferred Networks. *Optuna: A Hyperparameter Optimization Framework*. https://optuna.org/. Accedido el 4 de julio de 2023.

[26] *Visual Studio Code*. https://code.visualstudio.com/. Accessed: 5th July 2023.

[27] *GitHub*. https://github.com/. Accessed: 5th July 2023.

[28] *ZenHub*. https://www.zenhub.com/. Accessed: 5th July 2023.

[29] TeX user groups. *Tex Live - Tex Users Group*. URL: https://www.tug.org/texlive/.

[30] Code Climate. *CodeClimate*. https://codeclimate.com/. Accessed: 5th July 2023.

[31] *Pylint*. https://www.pylint.org/. Accessed: 5th July 2023.

[32] DB-Engines. *DB-Engines Ranking of Time Series DBMS*. [Internet; read on 22-march-2023]. Mar. de 2023. URL: https://db-engines.com/en/ranking/time+series+dbms.

[33] InfluxData. *Chronograf 1.10 documentation*. 2023. URL: https://docs.influxdata.com/chronograf/v1.10/.

[34] Christoph Praschl et al. “A Comparison Of Relational, NoSQL and NewSQL Database Management Systems For The Persistence Of Time Series Data”. En: *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 2022, págs. 1-6. DOI: 10.1109/ICECCME55909.2022.9988333.

[35] Alexey Struckov et al. “Evaluation of modern tools and techniques for storing time-series data”. En: *Procedia Computer Science* 156 (2019). 8th International Young Scientists Conference on Computational Science, YSC2019, 24-28 June 2019, Heraklion, Greece, págs. 19-28. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.08.125. URL: https://www.sciencedirect.com/science/article/pii/S1877050919310439.

[36] Andreas Theissler et al. “Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry”. En: *Reliability Engineering & System Safety* 215 (2021), pág. 107864. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.107864. URL: https://www.sciencedirect.com/science/article/pii/S0951832021003835.

[37] Anna Jodejko-Pietruczuk y Sylwia Werbińska-Wojciechowska. “Availability assessment for a multi-AGV system based on simulation modeling approach”. En: *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 2021, págs. 1-6. DOI: 10.1109/ICECCME52200.2021.9590979.

[38] James C. Chen et al. “Combining empirical mode decomposition and deep recurrent neural networks for predictive maintenance of lithium-ion battery”. En: *Advanced Engineering Informatics* 50 (2021), pág. 101405. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2021.101405. URL: https://www.sciencedirect.com/science/article/pii/S1474034621001579.

[39] Rundong Yan, S.J. Dunnett y L.M. Jackson. “Novel methodology for optimising the design, operation and maintenance of a multi-AGV system”. En: *Reliability Engineering & System Safety* 178 (2018), págs. 130-139. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2018.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0951832017311559.

[40] Unit8co. *Darts - Anomaly Detection*. 2023. URL: https://unit8co.github.io/darts/generated_api/darts.ad.html.