

Estacion meteorologia

Diseño de sistemas electrónicos



Gorga Berganza

Mikel Ruiz de la illa

Jorge Romeral

Aitor Zambrano

Asier Vega

11 de enero de 2023

Resumen (Abstract)

Los 5 integrantes del grupo vamos a realizar un proyecto simulando una estación meteorológica. Esta, mediante un microcontrolador y distintos sensores, capturará datos para luego ser mostrados. Al estar la sensorica alejada, necesitamos un medio de comunicación para poder capturar estos datos. Los sensores compartirán su captura mediante una comunicación industrial inalámbrica, para la cual hemos elegido el wifi industrial.

Una vez hecha la comunicación, vamos a guardar en una base de datos todos los datos obtenidos con el fin de luego ser mostrados. A parte de este muestreo, se pretende utilizar todos los datos obtenidos para realizar una predicción a futuro y que está también esté plasmada visualmente. Para ello vamos a utilizar un software llamado Grafana que nos va a permitir utilizar distintas opciones de muestreo de datos como podrían ser gráficos o cuadros de mando, incluso a tiempo real.

Descriptores

- Meteorología
- Monitorización
- Wifi
- Arduino
- Kubernetes
- Prometheus
- Grafana
- Python
- Modelo predictivo
- Visualización

0. Índices

0.1 Índice de contenidos

Resumen (Abstract)	2
Descriptores	2
0. Índices	3
0.1 Índice de contenidos	3
0.2 Índice de figuras	4
1. Introducción	5
1.1 Justificación	5
1.1.1 El reto	5
1.1.2 Propuesta de solución	6
1.1.3 Importancia del proyecto	6
1.2 Antecedentes	7
1.2.2 Estado del arte	7
Percepción	7
Comunicación	7
Almacenamiento	8
Visualización	9
CNF	9
1.2.3 Riesgos del proyecto	10
1.2.4 Solución que creemos más óptima	10
2. Requisitos y Descripción técnica	11
2.1 Requisitos	11
2.2 Descripción técnica de la solución	12
2.2.0 Foto general del sistema	12
2.2.1 Percepción	13
Arduino	13
Placa de las estación meteorológica + wifi shield	15
Sensores	16
2.2.3 Comunicación	17
Red wifi	17
Emisión de datos (servidor arduino)	18
Recepción de datos (flask)	19
2.2.4 Arquitectura CNF, almacenado y predicción	20
Arquitectura CNF	20
Base de datos	21
Predicción	22
2.2.4 Visualización	25
Dashboard	25
Alertas	27
3. Metodológica	29
3.1 Metodología Kanban	29
4. Planificación	30
4.1 Estructura de desglose de trabajo	30
4.2 Diagrama de Gantt	31

0.2 Índice de figuras

Imagen 0: Canales de frecuencia WIFI	8
Tabla 0: requisitos	11
Imagen 1: Foto general del sistema	12
Imagen 2: Esquema del circuito	14
Imagen 3: Arquitectura de red	16
Imagen 4: Error de la librería PromLokiTransport	17
Imagen 5: Arquitectura CNF	20
Imagen 6: Matriz de correlación	22
Imagen 7: Valores anormalmente bajos	23
Imagen 8: Gráficos para valores actuales	24
Imagen 9: Gráfico para histórico de datos	25
Imagen 10: Gráficos para valores medios	25
Imagen 11: Alerta con aviso enviado	27
Imagen 12: Aleta restaurada con aviso sin enviar	27
Imagen 13: Ejemplo de correo enviado desde estación-2	27

1. Introducción

1.1 Justificación

1.1.1 El reto

Como principales objetivos de este proyecto tenemos la recogida y visualización de una serie de datos sobre una estación meteorológica. Para ello vamos a tener que superar ciertos retos separados por competencias, que unidos estas y completando los retos consigamos cumplir los objetivos.

Para afrontar el primer reto haremos uso de diversos sensores que nos aportarán información para poder empezar con la captación. Esta captura no es el único desafío de esta competencia de captación de datos sino que también tendremos que modificar esa lectura a las unidades que nosotros queramos manejar. Cada proveedor de cada sensor utiliza un estándar de unidades con las cuales el sensor capta los valores.

Una vez obtenidos los datos procedemos a guardar los en una base de datos. Para ello vamos a tener que afrontar el reto de extraer los datos del microcontrolador y poder guardarlos en una base de datos en la nube. Para esta extracción tenemos que utilizar una comunicación industrial la cual nos permita solicitar información cada cierto tiempo sucesivamente o cada vez que nosotros la solicitamos. Utilizar esta comunicación industrial directamente desde el controlador hasta la base de datos nos va a permitir en un futuro poder tener más de una estación meteorológica conectada simultáneamente en distintas localizaciones.

Siguiendo con los retos a cumplir, una vez dispongamos de los datos almacenados en la base de datos y con las unidades que nosotros queremos tratar, vamos a pasar al muestreo. En esta parte tenemos que enlazar un software de visualización con la base de datos. Aparte de esta conexión y de obtener los datos, también disponemos del desafío de realizar este muestreo en el software, de la manera en la que nosotros deseemos y lo más visual y legible posible.

1.1.2 Propuesta de solución

Para cumplir con todos los retos anteriormente mencionados vamos a completar una propuesta de funcionamiento general. Para ello, vamos a hacer uso de distintos componentes hardware y software mediante los cuales vamos a conseguir entrelazarlos y cumplir los objetivos.

Empezando con los dispositivos hardware, vamos a utilizar 6 distintos sensores, una placa de arduino y un módulo de comunicación industrial. Estos 6 sensores juntos van a simular nuestra estación meteorológica en la que capturaremos los siguientes parámetros: la temperatura ambiente, la presión atmosférica, la humedad, la velocidad del viento y la dirección de este, y por último la precipitación.

Por otro lado, para el uso del controlador vamos a utilizar la placa de Arduino UNO, en ella programaremos lo necesario para que mediante esta programación y el módulo de comunicación industrial podamos obtener los datos percibidos por los sensores. Para esta comunicación utilizaremos el módulo de comunicación Wifi que dispone arduino.

1.1.3 Importancia del proyecto

Al tratarse de un proyecto en el ámbito educativo, el principal objetivo y valor de este proyecto es el aprendizaje de todas las partes implicadas, los conocimientos que lograremos realizando esta implementación son los siguientes:

- Entenderemos mejor cómo se organizan las arquitecturas de aplicaciones IoT así como sus capas se interrelacionan.
- Aprenderemos a utilizar varios sensores así como conocer la manera en la que envía los datos.
- Aprenderemos en profundidad el funcionamiento de un módulo de comunicaciones wifi, así como sus beneficios y desventajas.
- Realizaremos una documentación detallada en la que se plasmará todo el aprendizaje e investigaciones realizadas.

1.2 Antecedentes

1.2.2 Estado del arte

Percepción

Para la realización de nuestro proyecto se ha realizado una elección de diversos sensores con los que se creará un símil a una estación meteorológica. Para esta elección hay que tener en cuenta dos conceptos. El primero y el más condicionante es que los sensores elegidos tienen que estar a disposición de Deusto para que nos los puedan proporcionar, el segundo concepto es que estos sensores tienen que estar relacionados con la meteorología para que así tengan una funcionalidad correcta.

Las distintas estaciones meteorológicas existentes hacen uso de diversos sensores para luego así mostrar sus valores. Unas estaciones hacen un análisis más exhaustivo que otras debido a la utilización de sensores complejos pero como base todas hacen uso de los siguientes sensores:

- Temperatura.
- Presión atmosférica.
- Dirección y velocidad del viento.
- Radiación solar.
- Grado de humedad.
- Precipitación.

En nuestro caso, Deusto nos ha proporcionado un equipo de meteorología en el que dispone de varios sensores de medición. En él podemos encontrar los principales sensores que se necesitan para una medición meteorológica básica comentados anteriormente. Es verdad que en el kit proporcionado no disponíamos de un sensor para la radiación solar pero se ha optado por utilizar un sensor de luminosidad con el cual podemos controlar la variación de luz.

Comunicación

La tecnología Wi-Fi (según la norma IEEE 802.11n) empezó a utilizarse en la frecuencia de 2.4GHz, aunque hoy en día se utiliza el espectro de frecuencia de 5GHz. Nuestro componente wifi trabaja bajo la frecuencia de 2.4GHz (2412-2472 MHz), esto hace que disponga de 13 canales (+1 alejado de este espectro) el ancho de banda de estos canales es de 20MHz, estos se solapan por lo que en realidad solo son 4 canales. Este tipo de tecnología se hizo muy popular y enseguida creció por lo que se vio saturada muy rápido, provocando mayor interferencia, teniendo en cuenta su menor velocidad de conexión. Además cuenta con un rango de cobertura muy amplio. [10][11][12]

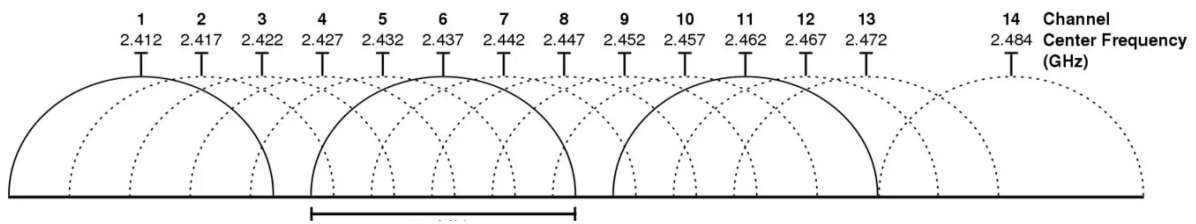


Imagen 0: Canales de frecuencia WIFI

Como podemos observar en la imagen realmente solo tenemos 4 canales, a 20 MHz por lo que puede transmitir simultáneamente (1,6,11 y 14) ya que al usar el resto causaría que otros canales se solapasen entre sí, logrando menos canales simultáneos en el proceso. [12]

Dependiendo de la distancia de los clientes inalámbricos haciendo uso de wifi, se hace uso de 2 tipos de modulación digital, Modulación por desplazamiento de fase (PSK) y Modulación de amplitud en cuadratura (QAM). Ciertamente que dentro de la modulación PSK se usa binario (BPSK) o de 4 estados (QPSK). Nuestro módulo de wifi hace uso de la modulación PSK. Está, manda la información en un enlace radio viaja en forma de ondas y con esta modulación lo que se hace es cambiar la fase de la onda para cambiar el símbolo o bit mandado. En BPSK tenemos dos posiciones para cambiar de fase representando el 0 y el 1. En QPSK tenemos cuatro posiciones representando 00, 01, 10 y 11, de esta manera en QPSK se manda más información que en BPSK utilizando en mismo espectro de frecuencias o ancho de canal.

A nivel de protocolo, todos los protocolos wifi se basan en el primer protocolo wifi (IEEE 802.11) lanzado en 1997 que utiliza el protocolo de internet para el enrutamiento. A partir de este estándar se han ido creando los protocolos que usamos hoy en día.

A nivel de usuario, nuestro módulo de Wifi 4 (802.11n), lanzado en 2009, permite el uso de las bandas 2.4 y 5 GHz y puede transmitir a una velocidad máxima de 600Mbps (teórico). Este fue el primer estándar en permitir compatibilidad MIMO (múltiples entradas y múltiples salidas) permitiendo así el uso de múltiples antenas de forma simultánea.[13]

Almacenamiento

Para el almacenamiento de datos se ha utilizado Prometheus, que es una aplicación de base de datos libre de series temporales registradas en tiempo real. Esta aplicación emplea extracción de métricas a través de un modelo pull usando HTTP. Está escrito en GO y basado en la licencia de apache 2.0. Además de esto también tiene un sistema de alarmas que no emplearemos en esta práctica (alert manager)[1]

El almacenamiento de los datos se realiza de la siguiente manera: se definen una métrica a escuchar (nombre y descripción) los valores de esa métrica se van introduciendo de forma arbitraria a través de pares clave valor. Después empleando "promQL" (lenguaje de queries de prometheus) podemos representar los datos almacenados de distintas maneras.[1]

¿Por qué se ha utilizado?

Definimos nuestro requerimiento de almacenamiento de datos:

- Volumen: pequeño, solo se va recoger un dato numérico (valor actual).
- Velocidad: alta, sería conveniente conocer el valor con la mayor brevedad posible (se recoge cada 5s pero dependemos de la velocidad de actualización de Bitfinex)
- Variedad: ninguna, es siempre el mismo dato del último valor del bitcoin
- Veracidad: tenemos que ser precisos con la hora de recogida del dato.

Teniendo en cuenta lo de arriba, hemos decidido utilizar una base de datos de series temporales debido a que solo tenemos un dato de interés que recuperar y es interesante conocer la fecha y hora en la que recuperamos ese dato. Esta aplicación está pensada para hacer seguimiento a esta moneda y así poder prever los mejores momentos para adquirirla o venderla.

Visualización

Para la visualización se ha utilizado Grafana, que es un software de visualización libre de datos métricos, datos cuantitativos que permiten realizar cualquier tipo de operación matemática. Esta aplicación emplea bases de datos de registros temporales como la que nosotros empleamos. Está escrito en GO y se basa en la licencia de Apache 2.0. [2]

¿Por qué se ha utilizado?

Se ha empleado Grafana porque tiene una configuración sencilla, rápida y es una aplicación pensada para visualizar datos de registros temporales. Por lo tanto nos permite realizar múltiples gráficos de manera simple y muy visuales para el usuario.

CNF

Se ha decidido utilizar la arquitectura CNF (Cloud-Native Network Functions) debido a que las funciones de esta arquitectura son las perfectas para escalar las cargas de trabajo a una arquitectura en la nube. Por otro lado, esta arquitectura nos permite funcionalidades de automatización con las que se nos garantiza una mayor comodidad y disponibilidad.

Por otro lado, en Kubernetes, el rendimiento es siempre una preocupación. A medida que la industria adopta la arquitectura CNF para construir sistemas con contenedores y pods en Kubernetes, la preocupación por el rendimiento no desaparece por completo. El controlador F5 (gráfico de configuración que se envía al servidor API en el modo maestro de Kubernetes) proporciona un rendimiento amplificado mediante la descarga de determinadas funcionalidades seleccionadas al hardware. Esto ofrece a los proveedores de servicios lo mejor de ambos mundos: las características nativas de la nube con las características de mayor rendimiento de una plataforma Kubernetes.[14]

Esta es una de las principales justificaciones para nuestra elección pero la que más peso tiene es la elección de esta debido a nuestro previo conocimiento. En el aula hemos llegado a hacer uso de mini kube y Kubernetes lo que nos hace más fácil la elaboración del proyecto. Añadiendo a esto que esta herramienta es fácil en un entorno de aprendizaje.

1.2.3 Riesgos del proyecto

Se han localizado los siguientes riesgos que podrían afectar a la integridad y cumplimiento de este proyecto:

- Se ha detectado que el escudo wifi suministrado al inicio del proyecto, no cuenta con la documentación actualizada para hacerlo funcionar, el fabricante indica que este componente está catalogado como obsoleto pero que podría usarse.
- No se conoce la manera de conectar la base de datos al origen de los datos, ya que no existe una librería que haga el transporte a las métricas que prometió leer.
- No se ha proporcionado un requisito no funcional para las alertas, por lo que tenemos dos opciones o configurarlas en Grafana o en Prometheus usando el alertmanager.

1.2.4 Solución que creemos más óptima

explicar por qué el wifi no es la mejor opción y proponer una alternativa

A la hora de poner en funcionamiento el dispositivo wifi, nos hemos encontrado con el problema de que nuestro Wifly shield no se conectaba al router, o al menos no aparecía conectado. Sin embargo sí que nos daba señal al hacer ping a este. Debido a que pese a consultarlo con Jonatan, y a haber probado otras posibles opciones de conectarnos, no lo hemos conseguido, hemos considerado que lo importante era elegir un método de comunicación más óptimo y posible de realizar.

Finalmente nos hemos decantado por otra alternativa. Montar un flask en un ordenador local el cual servirá para tomar los datos de la estación meteorológica y ponerlo en una dirección IP para que desde una máquina virtual de Linux pueda acceder a estos al estar en una misma red interna dentro de la misma wifi.

2. Requisitos y Descripción técnica

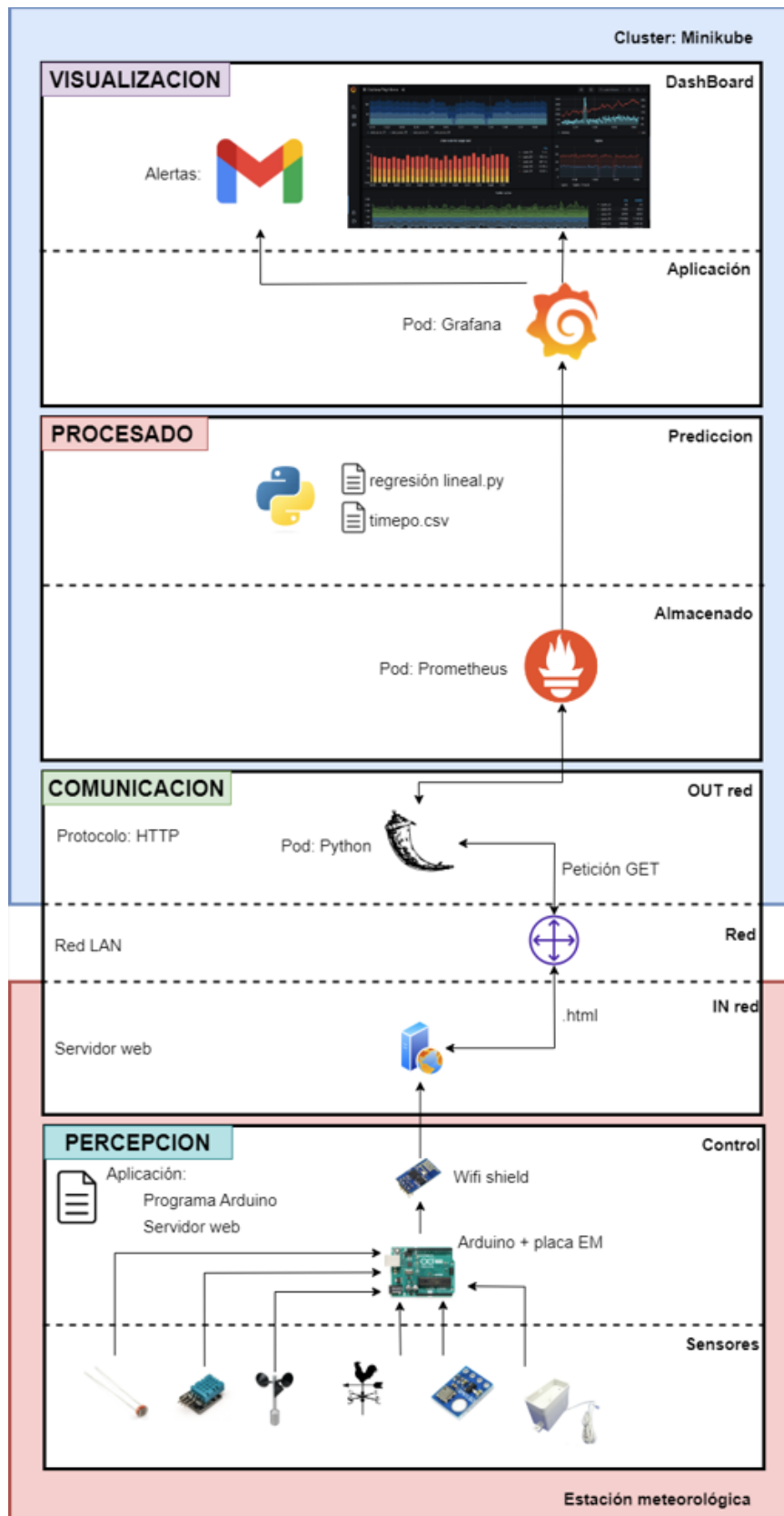
2.1 Requisitos

ID	Requisito	Tipo	Ámbito	Prioridad
0	La estación meteorológica está formada por un dispositivo para medir y enviar la información.	No func.	Arquitectura	
1	La estación meteorológica está formada por un sistema para visualizar y monitorizar la información que reciba.	No func.	Arquitectura	
2	El dispositivo cuenta con un sistema embebido arduino para el control y envío de datos de los sensores.	Funcional	Interacción	
3	El dispositivo cuenta con un sensor de temperatura (C°) para medir la temperatura medioambiental.	Funcional	Interacción	
4	El dispositivo cuenta con un sensor de humedad (%) para medir la humedad medioambiental.	Funcional	Interacción	
5	El dispositivo cuenta con un anemómetro (m/s) para medir la velocidad del viento.	Funcional	Interacción	
6	El dispositivo cuenta con una veleta para medir la dirección del viento.	Funcional	Interacción	
7	El dispositivo cuenta con un barómetro para medir la presión atmosférica (Pa) y la altitud.	Funcional	Interacción	
8	El dispositivo cuenta con un pluviómetro (l/m ²) para medir la cantidad de agua por metro cuadrado.	Funcional	Interacción	
9	El dispositivo cuenta con un wifi transceiver para poder enviar los datos via wifi.	Funcional	Interacción	
10	El dispositivo cuenta con un sensor de luminosidad para medir la cantidad de luz.	Funcional	Interacción	
11	El dispositivo cuenta con un medidor de CO2 para medir la cantidad de este gas.	Funcional	Interacción	
12	El dispositivo está alimentado vía una pila de 9V de larga duración.	No func.	Alimentación	
13	El dispositivo está diseñado de manera que se adapte a los estándares internacionales de las estaciones meteorológicas.	Funcional	Diseño	
14	El sistema pide los datos al dispositivo (arquitectura pull) cada 5 segundos.	Funcional	Interacción	
15	El sistema almacena los datos en una base de datos de series temporales para mantener un seguimiento de todos los datos recibidos.	Funcional	Interacción	
16	El sistema muestra la información en diferentes gráficos para que el usuario pueda ver la información de su estación meteorológica.	Funcional	Interacción	
17	El sistema muestra una serie de alertas para informar al usuario de distintas situaciones.	Funcional	Interacción	
18	El sistema está montado en una arquitectura de contenedores para que cada aplicación este dividida por microservicios.	No func.	Arquitectura	
19	El sistema debe usar una base de datos Prometheus para el almacenamiento ordenado de los datos.	No func.	Arquitectura	
20	El sistema debe usar la aplicación Gráfica para la visualización de los datos	No func.	Arquitectura	
21	El sistema debe usar Docker para el despliegue de contenedores.	No func.	Arquitectura	

Tabla 0: requisitos

2.2 Descripción técnica de la solución

2.2.0 Foto general del sistema



2.2.1 Percepción

Arduino

La placa de arduino tiene las siguientes entradas análogas, pin A0 velocidad del viento, A1 sensor de luz, A3 referencia de voltaje de la placa. Por otro lado, los pines digitales que se emplean son los siguientes, pines 7 y 9-13 para la comunicación SPI del dispositivo wi-fi, pin 8 para mostrar el estado de la estación meteorológica, los pines 0 y 1 son para pin 2 para el pluviómetro, pin 3 para el anemómetro, los pines 4, 5 y 6 son empleados para la switch del UART del GPS, que pueden dar problemas a la hora de cargar el software a la placa si no están conectados o el switch se encuentra en modelo de HW-UART.

Por otro lado, la estación meteorológica emplea los pines de SCL SDA para realizar comunicaciones mediante I2C, también emplea el pin IORF para tener compatibilidad de tensiones entre la placa arduino y el escudo de la estación meteorológica.

El resto de pines, se usa de modo compartido, ya que son pines de alimentación, tierra, reset y el pin AREF que es una señal de referencia para sensores que trabajan en tensiones iguales o inferiores a 5V.

El sistema funcionará con una alimentación constante, es por eso que hemos prescindido de las funciones de nivel de carga del escudo de la estación meteorológica, además de esta manera se han podido liberar los pines necesarios para compatibilizar los dos escudos. Ninguno de ellos va conectado sobre la placa de arduino directamente, se ha optado por utilizar ambos sobre una protoboard.

Placa de las estación meteorológica + wifi shield

Para poder utilizar las dos placas, surge un problema, y es la incompatibilidad de las mismas para poder ir una encima de otra sobre la placa de arduino, por lo que para ello, ha habido que modificar el uso de algunos pines, para evitar que coincidan, y por otro lado, mediante una protoboard, empalmar aquellos puertos que sean compartidos, como pueden ser los de alimentación o I2C. El problema ha sido con los pines digitales, ya que la placa de wifi no utiliza ninguno analógico y la meteorológica no utiliza más que dos.

Hemos sacado provecho de los pines de la placa de la estación meteorológica que son empleados para sensores como el de el GPS que no son utilizados, para poder eliminarlos y así dejar pines libres para la placa de Wi-Fi.

Los cables verdes se conectan al escudo de la estación meteorológica, en su lugar respectivo.

Los cables azules se conectan al escudo del wifi, en su lugar respectivo

Los cables rojos son un empalme de pines que comparten ambas placas

La estación meteorológica se comunica mediante I2C, para ello emplea los pines que salen en la parte superior derecha de la placa de arduino en la imagen anterior, estos son los pines de SCL y SDA, que son las señales de reloj empleadas para esta comunicación. Para realizar una comunicación maestro esclavo entre la placa de arduino y el shield de la estación meteorológica.

El protocolo de comunicación empleado por la placa wifi es una interfaz I2C-bus/SPI esclava a un canal UART de alto rendimiento, para eso se emplean 8 pines digitales que son los que se pueden ver en la imagen anterior, que son los pines que van del 0-6 ambos incluidos y el pin 8.

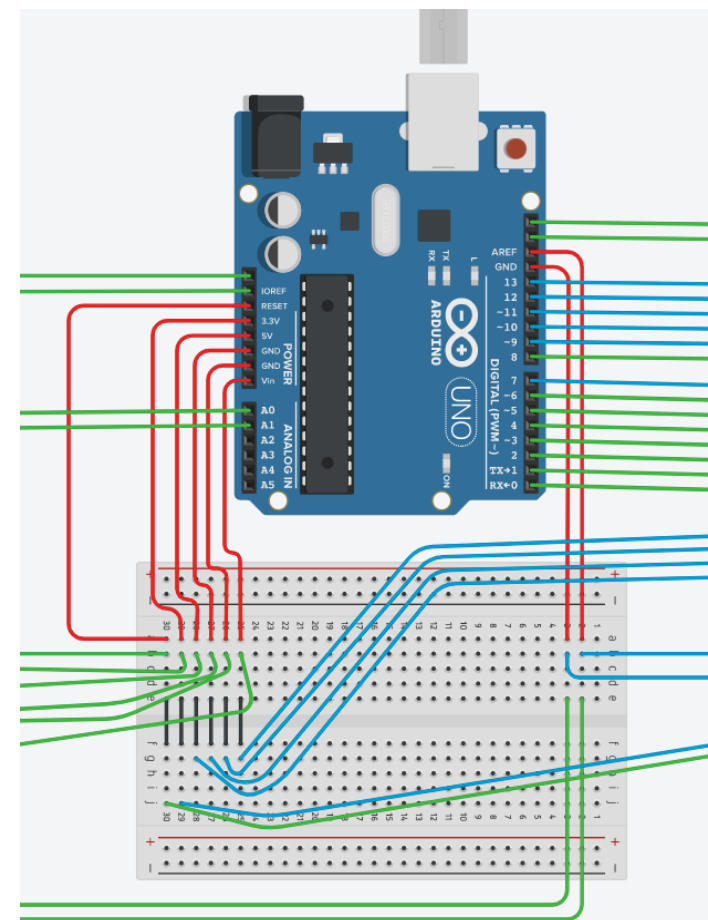


Imagen 2: Esquema del circuito

Sensores

La placa de arduino, emplea todo tipo de sensores para poder detectar y recibir señales del ambiente, todos ellos vienen acoplados mediante el “*Weather Shield*” de Sparkfun, el cual incorpora los siguientes sensores:

- Termómetro: Mide la temperatura del ambiente, lo hace mediante el mismo componente que se emplea para medir el porcentaje de humedad o el Barómetro, HTU21D o MPL3115A2 .
- Sensor de Humedad: Mide el porcentaje de humedad que hay en el ambiente, esto lo hace mediante un sensor HTU21D que está incorporado a la placa, este, además del porcentaje de humedad, como se ha mencionado anteriormente, mide la temperatura del ambiente. Utiliza una comunicación I2C, para ello, emplea los pines SDA y SCL, o en su defecto puede emplear dos pines digitales, y emplea la alimentación de 3.3V de la placa de arduino.
- Anemómetro: Se trata de un sensor analógico, que cada vuelta manda una señal, esto al ser comparado con un reloj interno en el sistema, se puede saber la velocidad a la que el viento se mueve, la medida de conversión es 2,4 km/h igual a 1 tic/segundo, con esta relación y aplicando una regla de 3, se puede sacar la velocidad del viento en todo momento.
- Pluviómetro: Es un sensor digital, ayudado por una estructura en forma de embudo, que recoge el agua, y cada vez que se procesan 0,2794 ml de agua, la balanza que tiene cede y se activa un pulsador que envía una señal digital, sabiendo el tiempo que pasa entre señales, y las dimensiones del embudo, se puede saber la cantidad de lluvia que hay por metro cuadrado.
- Barómetro: El barómetro, mide la presión atmosférica, y se hace mediante la incorporación del sensor MPL3115A2 a la placa, este sensor es capaz de medir tanto la presión atmosférica como la temperatura o la altitud del dispositivo, pero esta última opción no ha sido empleada. Para cumplir con su función, el sensor se alimenta a 3.3V, y emplea convertidores lógicos para limitar las señales de 5V. Aparte de eso, se comunica mediante I2C, por lo que emplea los pines SDA y SCL o en su defecto dos pines digitales.
- Luz ambiental: Es un sensor digital sencillo, que varía el voltaje entre 0-3.3V según la luz que percibe del ambiente, siendo 0 cuando el sensor no percibe ningún tipo de luz.
- Dirección del viento: Está formado por una serie de puertas magnéticas que se cierran según la veleta se mueve, obteniendo una orientación en grados, Tanto este como el anemómetro y el pluviómetro están conectados al escudo mediante RJ-11 y luego se comunican mediante I2C con la placa arduino.

Posteriormente, el “*Shield*” se comunica mediante I2C con la placa de arduino.

En la placa de arduino, los datos son procesados para obtener los datos en unidades del sistema universal legibles y entendibles. Para ello se emplean 3 librerías distintas, que son, la librería h.wire, encargada de las funciones para la comunicación I2C, y las librerías del barómetro y la del sensor de humedad.

2.2.3 Comunicación

Red wifi

La red wifi que planteamos para el proyecto es una red LAN compuesta por una (es posible ampliar a más) estación meteorológica (servidor web), un cluster de mini kube con un pod con Flask (aplicación web) en un ordenador y un router.

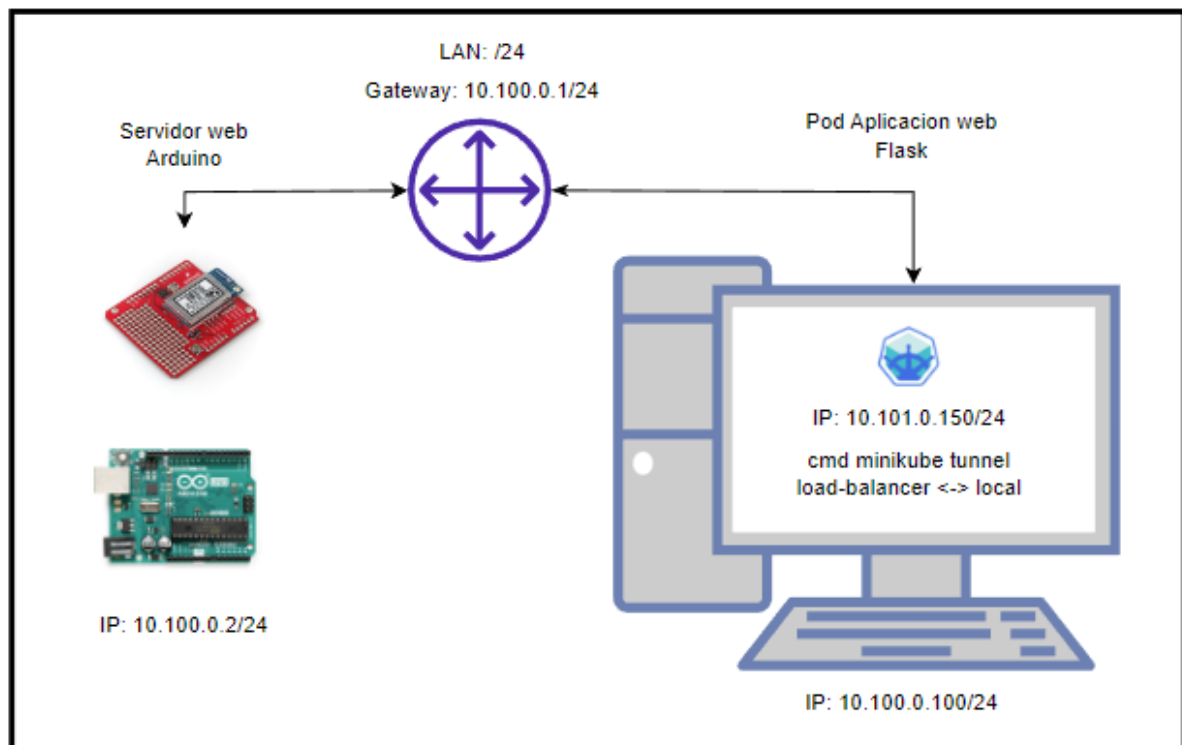


Imagen 3: Arquitectura de red

Como se puede observar en la imagen se crea una red LAN en el rango de 10.100.0.0/24 donde se conectan los dos dispositivos a través del mismo router. En el ordenador se levanta el cluster de kubernetes (minikube) y a través de un servicio tipo “load balancer” en el rango de ip definido en el cluster para estos servicios. Podemos realizar peticiones al servidor de arduino, si antes habilitamos el tunnel en minikube.

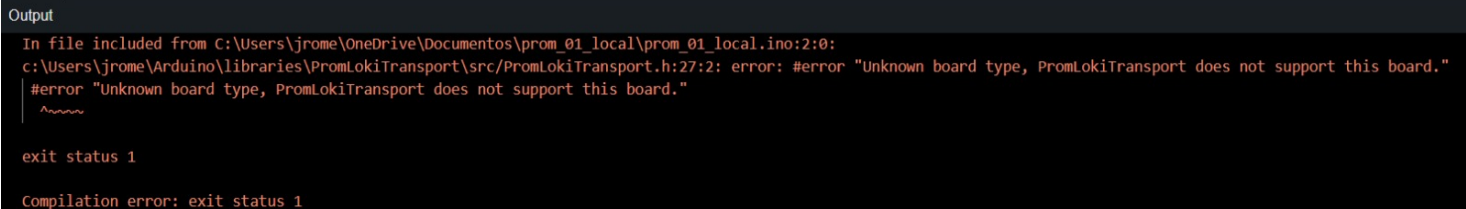
Lo que realiza el comando “minikube tunnel” es ejecutar un proceso que crea una ruta de red al HOST usando la ip del cluster como gateway que expone las ips definidas como “loadBalancer” a cualquier programa del ordenador. [8]

En una situación real, donde la red no podría ser de área local debido a que la estación se encuentre demasiado lejos del servidor, tendríamos que asignar una ip pública fija por estación (además de tener un punto de acceso a internet). Cosa que aumentaría muchos los costes “run” del proyecto, sería recomendable cambiar el tipo de comunicación a una de largo alcance por radio frecuencia.

Emisión de datos (servidor arduino)

- Problema encontrado:

El problema que hemos detectado a la hora de realizar la solución nos ha surgido a la hora de intentar conectar la placa de arduino con la base de datos para pasar los datos simultáneamente y que no tenga que haber un PC haciendo de intermediario entre estos dos. A la hora de cargar el programa y lanzarlo en el arduino nos daba el siguiente problema:



```
Output
In file included from C:\Users\jrome\OneDrive\Documents\prom_01_local\prom_01_local.ino:2:0:
c:\Users\jrome\Arduino\libraries\PromLokiTransport\src\PromLokiTransport.h:27:2: error: #error "Unknown board type, PromLokiTransport does not support this board."
#error "Unknown board type, PromLokiTransport does not support this board."
~~~~~
exit status 1
Compilation error: exit status 1
```

Imagen 4: Error de la librería PromLokiTransport

Este problema es debido a que a la hora de realizar la conexión con la placa de arduino UNO, no la realiza correctamente por problemas de compatibilidad. La librería a utilizar para conseguir cumplir el reto de conectar directamente el controlador con la base de datos se llama "PromLokiTransport". El objetivo era poder dejar el sistema disponible para que en un futuro poder escalar en el número de estaciones meteorológicas. El problema de esta librería es que solo es compatible con las siguientes 3 placas bases:

- ESP32
- MKRWIFI1010
- MKRGSM1400

Debido a este problema que nos ha surgido tenemos que descartar la posibilidad de compartir la información desde el arduino. Para ponerle una solución a este problema, vamos a realizar una conexión intermedia. El arduino trabajará junto a un ordenador el cual mediante python publicará esa captura de datos. El ordenador al estar conectado también a la base de datos, cojera del local los datos y los plasmarlos para dejarlos guardados. Así posteriormente serán manejados mediante grafana para su muestreo.

- Solución propuesta:

Nuestra orientación para solucionar este problema es hacer que el arduino tenga un servidor web donde publique sus datos. Esto es que este tenga una ip fija dentro de la red LAN y publique un archivo ".html". Ese archivo está compuesto por el siguiente código:

```
"<html><body>1_"+str(temperatura)+"_"+str(presion)+"_"+str(humedad)+"_"+str(velocidadviento)+"_"+str(direccionviento)+"_"+str(lluvia)+"_"+str(luz)+"</body></html>"
```

En él se puede observar que concatenamos los valores a unas etiquetas "html" y "body" e indicamos un ID como primer parámetro. De esta manera cada vez que llamemos a este html a través de un petición "get" este html devolverá los valores actuales de la estación.

Recepción de datos (flask)

Para la recepción y petición de datos de las estaciones meteorológicas se emplea una aplicación web desarrollada con Flask, que realiza peticiones tipo “pull” a los servidores que levanta cada estación.

- Ejecución de la petición: la aplicación flask realiza una petición html “GET” a la url del respectivo servidor web de cada estación. Para ello utiliza la librería de python “requests”.
- Archivo html descargado: la página html que descargamos de cada estación tiene el siguiente formato: `<html><body>1_10.5_885_0_0_0_0_0</body></html>` los valores de dentro de las etiquetas “body” corresponden a las siguientes variables en el siguiente orden:
`“id_temperatura_presion_humedad_velocidadviento_direccionviento_lluvia_luz”`
Esta información será desglosada y convertida en un formato que Prometheus pueda entender.
- Construcción de la aplicación web (server.py):
 - Variables:
 - Índices, se debe definir por adelantado el número de estaciones y el ID que van a tener.
 - urls, se debe definir por adelantado el número de estaciones y la url por la que estos van a transmitir.
 - métricas, por cada índice que se defina se definen 7 variables de tipo “gauge” a través de la librería de “prometheus_client”. Estas nos valdrán para transmitir los datos de manera que prometheus entienda.
 - Rutas (métodos):
 - ruta “/”: página de prueba de funcionamiento de Flask, devuelve un html con un “hola mundo”.
 - ruta “/prueba”: página de prueba para extracción de html de página web externa, devuelve el html cargado y también lo pinta en los logs.
 - ruta “/metrics”: página a la que Prometheus va a recoger las métricas y su valor. Desde aquí se realiza un get de las páginas externas definidas en “urls”, se preparan los datos y se asignan a las métricas previamente establecidas. Devuelve una página web cargada a través de la librería “prometheus_client” con el formato para que prometheus las lea.

En conclusión este servidor web nos vale para centralizar y preparar los datos de las distintas estaciones. De esta manera evitamos hacer operaciones que consumen la batería del arduino de forma innecesaria.

2.2.4 Arquitectura CNF, almacenado y predicción

Arquitectura CNF

Para la organización de la base de datos y de la aplicación de visualización se ha empleado una arquitectura CNF empleando docker, kubernetes y el cluster minikube.

A continuación se muestran los distintos recursos empleados para el despliegue.

Configuración de grafana:

- grafana-deployment.yaml : este recurso de tipo deployment, configura el despliegue de los pods según la imagen de base de grafana [5], expone el puerto 3000 y monta varios volúmenes que cargan distintos archivos de configuración en distintas rutas, son los siguientes 5 archivos.
- grafana-ini-configmap.yaml: este recurso configura distintos parámetros que el Grafana debe tener de manera inicial. El que más nos interesa es el de la configuración de un servidor smtp, que más adelante explicaremos su funcionamiento.
- grafana-datasources-configmap.yaml: este recurso configura el origen de los datos, esto es el servicio de Prometheus (<http://prometheus.monitoring.svc:9090>).
- grafana-dashboards-configmap.yaml: este recurso contiene el .json que hemos generado configurando manualmente el dashboard de grafana.
- grafana-dashboardproviders.yaml: ese recurso carga el contenido del recurso grafana-dashboards-configmap.yaml en la ruta “/var/lib/grafana/dashboards” del pod generado de Grafana.
- grafana-notifier.yaml: este archivo configura un notificador de alertas, este servirá para que las alertas puedan ser enviadas a una dirección de correo definida también en este archivo.
- grafana-service.yaml: este recurso de tipo service, define la manera en la que exponemos el puerto definido en grafana-deployment.yaml, en este caso será de tipo “LoadBalancer” ya que necesitamos una ip externa para que el usuario pueda hacer vía navegador.

Configuración de prometheus:

- prometheus-deployment.yaml: este recurso de tipo deployment, configura el despliegue de los pods según la imagen de base de prometheus [4], expone el puerto 9090, monta un volumen para almacenar los datos de forma interna (si quisiéramos almacenar los datos en el equipo local y no dentro del pod este volumen tendría un volumen persistente y enlazado a una ruta en local) y otro volumen para cargar el archivo prometheus-configmap.yaml.
- prometheus-configmap.yaml: este recurso contiene la configuración para que prometheus realice el raspado de datos a la ruta de flask (ip-interna-flask:5000) y también se configura el tiempo de raspado que es de 5s.
- prometheus-service.yaml: este recurso de tipo service, define la manera en la que exponemos el puerto definido en prometheus-deployment.yaml, en este caso será de tipo “cluster ip” ya que solo se necesita que sea visible desde dentro del cluster.

Configuración de Python

- dockerfile: partiendo de la imagen oficial de Python [9] instalamos “Flask (2.2)” y las librerías “prometheus_client (0.15)” y “requests (2.22)”. Por último copiamos la aplicación “server.py” e indicamos que el Flask arranque esa aplicación “FLASK_APP = server.py” y ejecutamos Flask en 0.0.0.0 con el puerto 5000.
- server.py: aplicación web escrita en Python.
- flask-deployment.yaml: este recurso de tipo deployment, configura el despliegue los pods según la imagen modificada de Python (para ello se construye la imagen en el registry de mini kube no en el local) y expone el puerto 5000.
- flask-service.yaml: este recurso de tipo service, define la manera en la que exponemos el puerto definido en flask--deployment.yaml, en este caso será de tipo “LoadBalancer” ya que necesitamos una ip externa para que el falk pueda realizar las peticiones get al exterior.

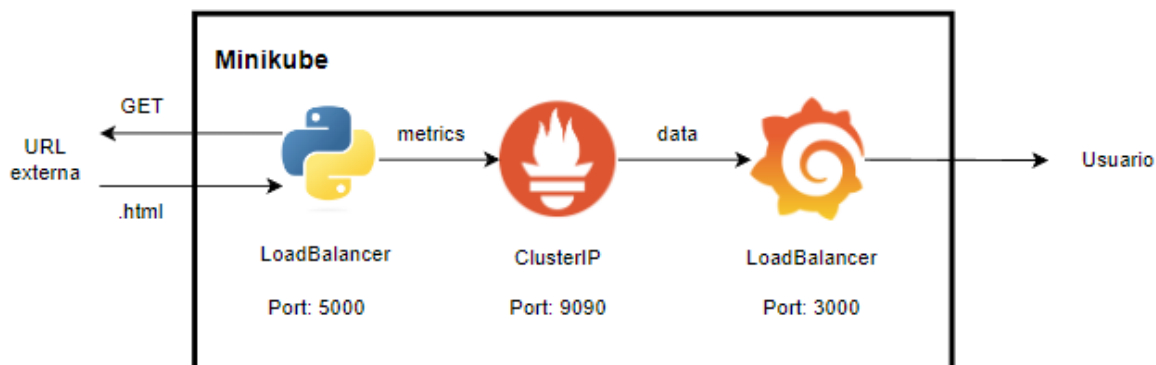


imagen 5: Arquitectura CNF

Base de datos

- Carga de datos desde flask:

Las métricas que cargamos en Prometheus son de tipo “gauge”, esto es son valores numéricos que suben y bajan de manera arbitraria, prometheus almacena su valor y su timestamp cada 5s (tiempo de scraping definido). Decidimos que para los gráficos que íbamos a representar en Grafana este tipo de métricas eran más que suficiente. Las métricas son las siguientes: `temperature_in_celsius_#`, `atmospheric_pressure_#`, `humidity_#`, `wind_speed_#`, `wind_direction_#`, `rain_precipitation_#`, `light_level_#`, sustituyendo # por el id de la estación.

- Descarga de datos desde grafana:

Para la descarga de los datos desde Prometheus a Grafana utilizamos el siguiente modelo de queries: `temperature_in_celsius_1`, como se puede observar es la misma métrica que subimos a Prometheus. Se decidió que para los gráficos que se utilizan no es necesario realizar ninguna agrupación y nos valen las métricas tipo “gauge” como tal.

Predicción

Una de las funcionalidades añadidas en este proyecto es la predicción de la temperatura máxima y mínima, que más adelante será mostrada en un dashboard de Grafana. Para ello se ha decidido usar una regresión lineal. Los argumentos de peso para dicha elección han sido la formación previa en esta materia y la familiarización del equipo con esta.

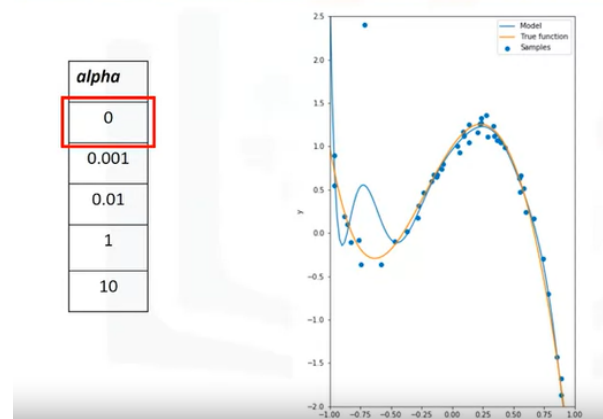
Se debe aclarar en primera instancia que la regresión lineal no es una inteligencia artificial, sino que pertenece a machine learning. El aprendizaje automático (machine learning) es una rama de la informática que se dedica al estudio y desarrollo de algoritmos y técnicas que permiten a las máquinas aprender de forma automática sin ser explícitamente programadas. El aprendizaje automático se basa en el uso de algoritmos y técnicas que pueden detectar patrones y tendencias en grandes conjuntos de datos, y utilizar esa información para mejorar el rendimiento de un sistema de manera autónoma. Hay diferentes tipos de aprendizaje automático, entre ellos el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo. El aprendizaje automático se utiliza en una amplia variedad de aplicaciones, como el reconocimiento de patrones, la clasificación de datos, la predicción de resultados y la toma de decisiones.

La regresión lineal es un método de análisis de datos que se utiliza para entender la relación entre dos variables. Esta técnica se basa en la idea de que hay una relación lineal entre la variable independiente y la variable dependiente. La regresión lineal trata de ajustar una línea recta a los datos de manera que se minimice la diferencia entre los valores observados y los valores predichos por la línea. La pendiente de la línea es una medida de cómo cambia la variable dependiente con respecto a la variable independiente. La intersección de la línea con el eje -y- es un término de interceptación que representa el valor de la variable dependiente cuando la variable independiente es cero.

En este caso se ha decidido hacer la predicción usando la regresión Ridge. La regresión de Ridge es un tipo de regresión lineal que se utiliza para evitar el sobreajuste (Overfitting) de los datos. A veces, cuando se entrena un modelo de regresión lineal, es posible que el modelo se ajuste muy bien a los datos de entrenamiento, pero que tenga un rendimiento pobre en los datos de prueba. Este fenómeno se conoce como sobreajuste. Una forma de evitar el sobreajuste es utilizar la regresión de Ridge,

que introduce una penalidad en el error del modelo para evitar que el modelo se ajuste demasiado a los datos. La penalidad se aplica mediante el uso de un parámetro llamado alpha (0.1 en nuestro modelo), que controla la intensidad de la penalidad. Cuanto mayor

Ridge Regression



sea α , mayor será la penalidad y el modelo será menos propenso al sobreajuste. La regresión de Ridge es una técnica de regularización que se utiliza a menudo en el aprendizaje automático para mejorar el rendimiento del modelo en problemas de generalización.

Remarcar también que un aspecto decisivo de la elección de este modelo fue su capacidad para evitar el overfitting.

La fuente de datos que se ha utilizado para entrenar la regresión lineal es el GHCN (Global Historical Climatology Network) esta elección de datos es una manera de trazar los pasos necesarios una vez se haya generado suficiente información en la estación, hasta entonces se expondrá un ejemplo de lo que podría ser la predicción.. GHCN es una página web que contiene datos e información climatológica de todo el mundo. Estos datos incluyen mediciones de temperatura, precipitación, nieve y viento, entre otras variables, que se recogen desde finales del siglo XVIII hasta la actualidad. GHCN es una red global de estaciones meteorológicas que se encuentran en todos los continentes y que proporcionan datos sobre el clima a nivel local. Estos datos se utilizan para estudiar cambios en el clima a lo largo del tiempo y para predecir el clima futuro. El GHCN es una fuente importante de información climática y es ampliamente utilizada por científicos y otros investigadores para estudiar el clima y sus efectos en el mundo, y por esto mismo ha sido la elegida en este proyecto, por su fiabilidad y popularidad.

*GHCN adjunta la documentación de sus datos cuando descargamos cualquier dataset.

Aunque los datos descargados del GHCN sean fiables, requieren de una normalización o ajuste para poder usarse y estudiarse. Estos han sido los pasos a seguir para escoger las variables de mayor importancia y ajustar la muestra de datos para que sea válida.

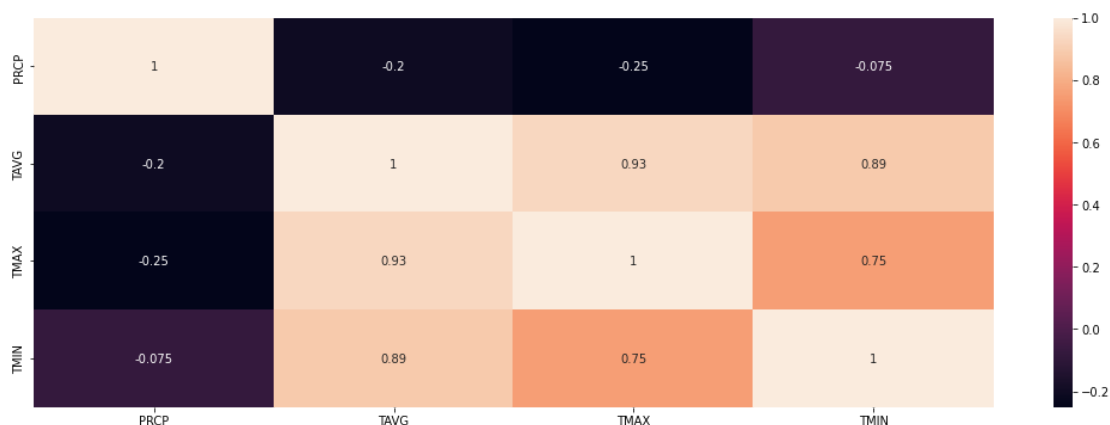


Imagen 6: Matriz de correlación

La muestra de datos inicial cuenta con 4 columnas, PRCP (Precipitaciones), TAVG (Temp. Media), TMAX (Temp. Máxima) y TMIN (Temp. Mínima), el primer problema que debe afrontarse son los datos "Null" nulos, datos que no son válidos para la predicción. En este caso particular la columna de TAVG, contiene un número elevado de valores nulos y tras estudiar los diferentes valores existentes en esta columna se ha decidido sustituir los valores nulos, por la suma de TMAX y TMIN partida en dos. La siguiente columna en la que existen un gran número de nulos, es la columna de PRCP, en este

caso tras estudiar la tendencia de los valores se observa que el valor más común es el -0- por lo que se decide sustituir dichos valores nulos por ceros.

Por último únicamente restan algunos valores nulos en la columnas TMAX y TMIN, en este caso y tras estudiar el comportamiento de los valores, se ha decidido utilizar el método “fillna” en la configuración “ffill”, este método asignará valores anteriores al valor nulo.

EJ. El día 2004-11-20 es un valor nulo, por lo que buscará si el día 19 es nulo, y en caso de que no lo sea, asignará el valor del día 19 al día 20, en caso de que el día anterior sea nulo, seguirá subiendo hasta encontrar un valor que no lo sea.

Una vez eliminados todos los valores nulos, se estudiarán los valores para intentar detectar valores que puedan entorpecer la predicción, normalmente valores muy dispares a la media. En este caso la TAVG supone un problema ya que existe un gran número de valores que son anormalmente bajos. Para que estos datos no afecten a la predicción se ha decidido eliminarlos.

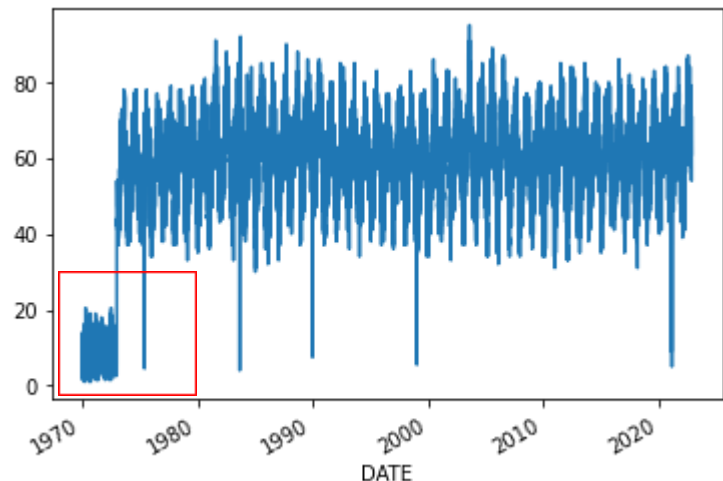


Imagen 6: Valores anómalamente bajos

Una vez modificados los datos se inicia con la predicción. Para ello se dividirá el dataset en 2 la parte “Train” y la parte “Test”, cada una de estas partes contará con los datos utilizados para predecir y el target, el valor que se quiere conseguir (Para generar target se ha desplazado el valor de TMAX un día). Tras esto se generan los coeficientes utilizando Train y se comprueba su efectividad utilizando Test.

Con el fin de mejorar la predicción se han generado más columnas a utilizar en la predicción y finalmente la tabla de correlación resultante es la siguiente.

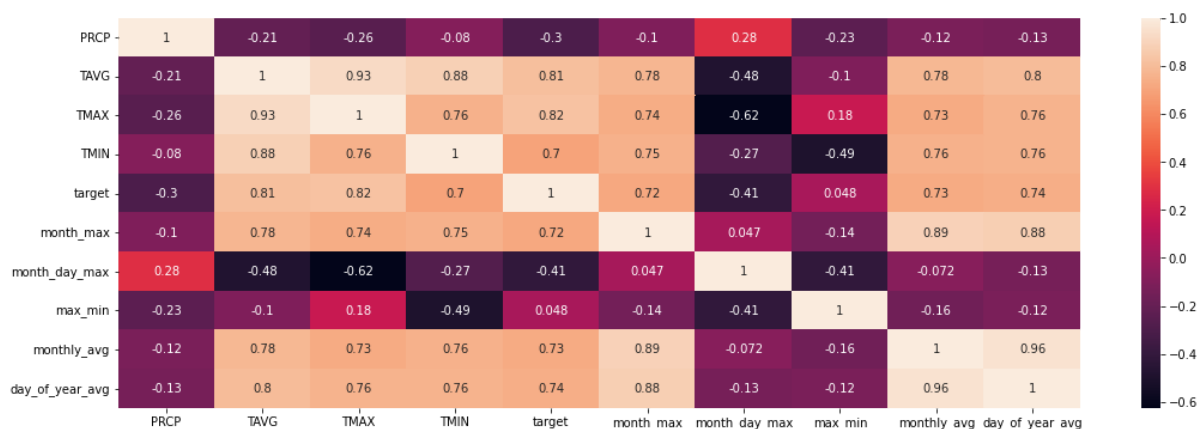


Imagen 7: Matriz de correlación

Tras generar la predicción con estos valores, se consigue un fallo medio de 4 grados Fahrenheit.

2.2.4 Visualización

La visualización de los datos de las estaciones se compone de un dashboard en Grafana y de unas alertas a la cuenta de correo electrónico del usuario.

Dashboard

El dashboard está repartido por pestañas, cada pestaña es una estación, de tal manera que cada una represente sus datos recogidos de las siguiente maneras:

- Indicadores de valores actuales: se cuenta con 6 gráficos tipo “Gauge” (valor + barra indicadora de máximos y mínimos) con los que se representa el valor actual de los sensores. Cada uno de este gráfico cambia el color del texto y de su barra en función del valor de la métrica.

Ejemplo: la temperatura se representa en °C. El color del número y de su barra varía en función del valor que recoja la métrica, mostrando:

- color azul si se encuentra entre -10°C y 23°C
- color verde si se encuentra entre 23°C y 35°C
- Color naranja si se encuentra entre 35°C y 50°C
- color rojo si se encuentra entre 50°C y 70°C.

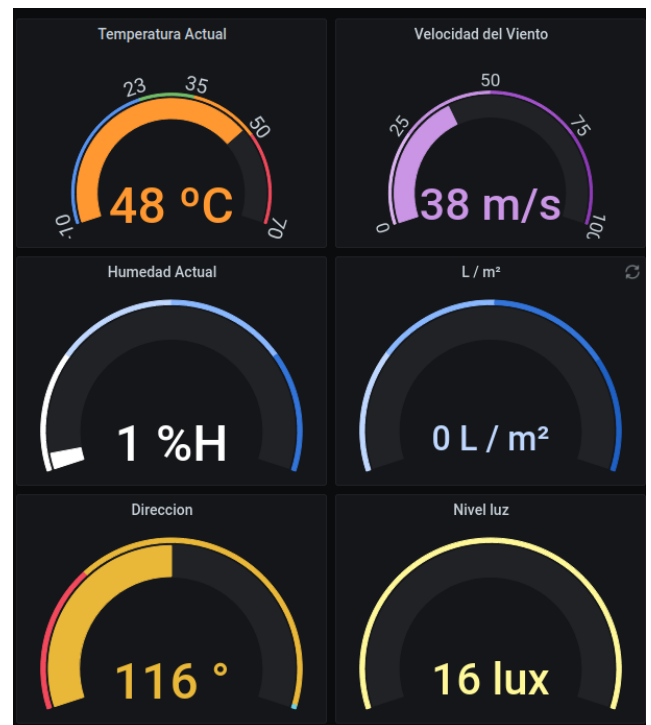


Imagen 8: Gráficos para valores actuales

Este ejemplo se replica de forma similar para cada una de las 6 métricas que recogemos en esta parte del dashboard.

Estos gráficos valen para que el usuario pueda distinguir fácilmente qué valores está recogiendo la estación actualmente y si se encuentra entre unos rangos normales.

- Indicadores de histórico y alertas: se cuenta con 3 gráficos de línea continua en los que se representa como el valor de la temperatura, humedad y presión atmosférica cambia conforme pasa el tiempo. En el eje Y se representa el valor y en el X el tiempo (este es configurable por el usuario). Además de esto también representa unos límites máximos y mínimos (líneas rojas) que se encuentran sincronizados con las alertas, el funcionamiento de esta se explica en el siguiente apartado.

Estos gráficos valen para que el usuario reciba alertas, ya que en los otros tipos de gráficos no es posible configurarlas, debido a que estas necesitan tener constancia de los valores desde hace 5 minutos, para comprobar si la alerta es correcta antes de enviar el aviso.



Imagen 9: Gráfico para histórico de datos

- Indicadores de medias: se cuenta con 4 gráficos de barras en cada pestaña que nos muestra el valor medio de la temperatura, l/m^2 , humedad y velocidad del viento que ha tenido esa estación desde que empezó a recoger datos. El color del texto y de la barra también cambia como lo hacen los gráficos de valores actuales.

Estos gráficos valen para que el usuario sea capaz de conocer los valores medios de la zona que recoge esa estación de forma intuitiva.

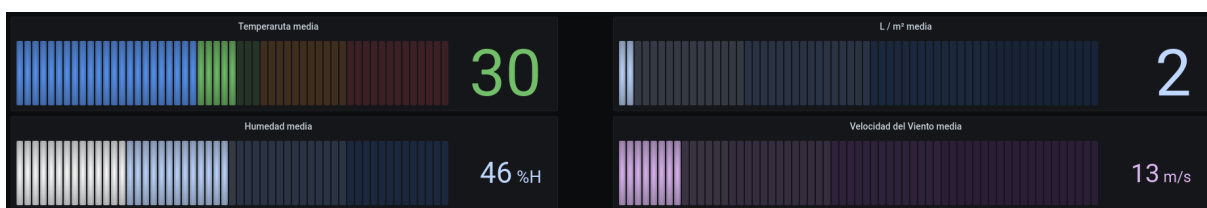


Imagen 10: Gráficos para valores medios

Alertas

Las alertas se configuran en los gráficos de tipo “Graph” ya que se necesita verificar el histórico desde que se detecta un valor hasta que se da la alerta. Estas se configuran dentro del dashboard por lo que van atadas al archivo: grafana-dashboards-configmap.yaml y grafana-dashboardproviders.yaml

Con las alertas definidas controlas los siguiente parámetros:

1. Rango de temperatura entre 45 y -5 °C (alerta temperatura anómala)
2. Rango de humedad entre 75 y 5 % (alerta humedad anómala)
3. Rango menor que 900 Pa (alerta “mañana hará mal tiempo”)

Para el envío de esta alertas se ha configurado una verificación en dos pasos de manera que una vez se realice un escaneo (se realizan cada minuto) de los valores actuales, si estos salen de los límites la alerta pasará a “pending” durante 5 minutos, entonces se realizará un nuevo escáner para verificar al anterior y entonces pasará la alerta a esta “alerting” y enviará el aviso al usuario. También se controla si se envían o no datos.

El medio de notificación de las alertas está configurado mediante los siguientes archivos:

- grafana-ini-confimap.yaml: este recurso configura entre otras cosas el servidor smtp de gmail que se va a usar para enviar un correo electrónico. Para ello se definen los siguientes parámetros en la configuración:

```
[smtp]
enabled = true
host = smtp.gmail.com:587
user = yosoifalso@gmail.com
password = kquvuqgftbnrvidp
```

Durante la configuración de esta parte se descubrió que el servidor smtp de gmail desde mayo de 2022 no permite la activación en una cuenta del uso de aplicaciones poco seguras. Como es nuestro caso ya que pasamos la contraseña por texto plano al servidor. [6]

Para solucionarlo, se habilitó una contraseña de aplicación. Este tipo de configuración requiere activar la doble autenticación en una cuenta de gmail y después genera una de estas contraseñas. De esta manera damos acceso al Grafana de manera más segura a nuestra cuenta de correo. [7]

- grafana-notifier.yaml: con este recurso indicamos los correos receptores a los que se deben enviar las alertas.

```
name: email_notifications
type: email
uid: notifier1
is_default: true
settings:
  addresses: asier.vega@gmail.com
```

Al configurar estos parámetros Grafana genera una instancia de canal de notificación y la establece como predeterminada bajo un nombre. En el dashboard se ha configurado este mismo nombre por lo que al coincidir no es necesario realizar ningún cambio manual al generar el pod.

Ejemplo:

En el siguiente ejemplo se puede observar como la alerta pasa de la línea roja de 45°C y esta pasa a pendiente en el escaneo habitual (cada minuto), el momento exacto del cambio viene marcado por el de la línea naranja vertical. A los 5 minutos se puede observar que el valor sigue siendo el mismo y la línea roja vertical marca el punto en el que se envía el correo con la notificación al usuario. En el caso de que el valor vuelva a la normalidad después de 5 minutos, se marcará con una línea verde el momento en el que la alerta se haya solucionado.

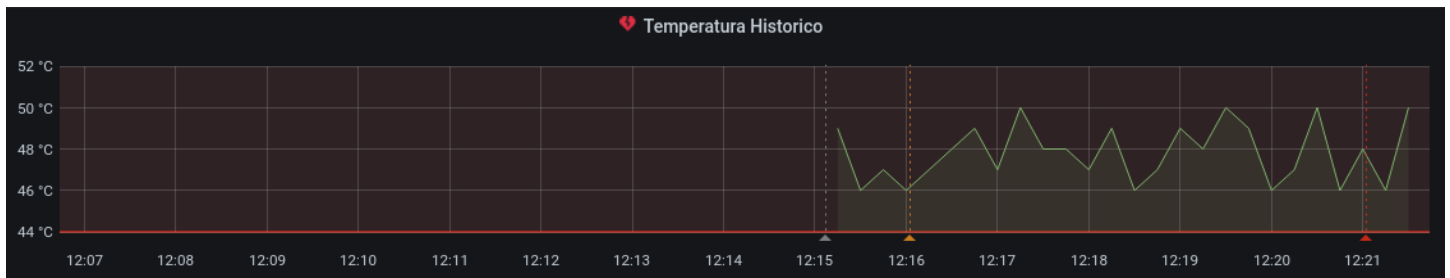


Imagen 11: Alerta con aviso enviado

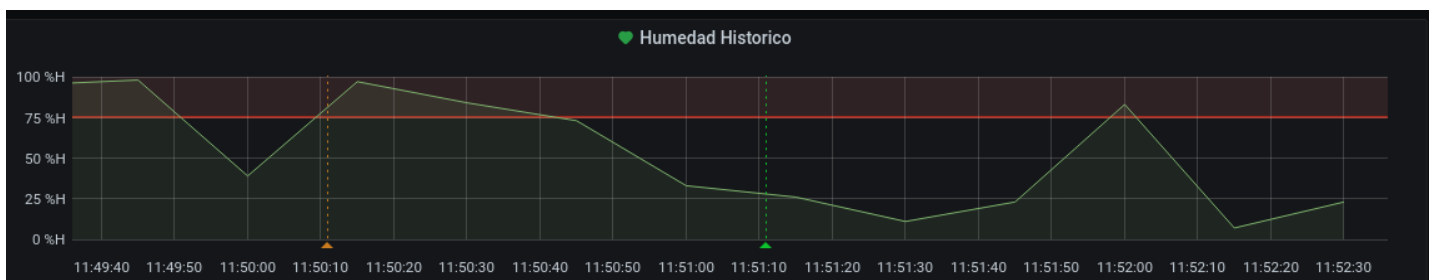


Imagen 12: Aleta restaurada con aviso sin enviar

En la siguiente foto se puede ver la alerta que se envía al correo electrónico del usuario:

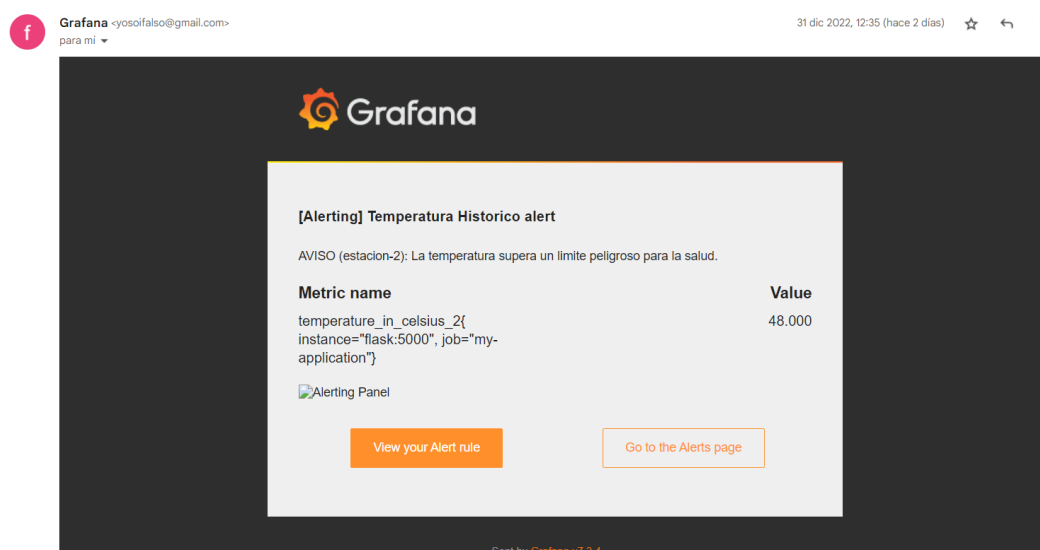


Imagen 13: Ejemplo de correo enviado desde estación-2

3. Metodológica

3.1 Metodología Kanban

Kanban es una metodología de gestión de proyectos que se basa en la visualización de tareas y el control del flujo de trabajo. El objetivo principal de Kanban es mejorar la eficiencia y la efectividad del proceso de trabajo al permitir una mejor planificación y una mayor flexibilidad para adaptarse a los cambios.

La metodología Kanban se originó en los años 1950 en las fábricas de Toyota, donde se utilizaba para optimizar la producción de vehículos. Desde entonces, ha sido adoptada por muchas otras empresas en diferentes industrias y ha sido adaptada para su uso en proyectos de software.

En Kanban, se utiliza un tablero visual para representar el flujo de trabajo a lo largo del proyecto. El tablero está dividido en columnas que representan diferentes etapas del proceso de trabajo, y las tareas se representan mediante tarjetas que se mueven a lo largo del tablero a medida que avanzan en el proceso.



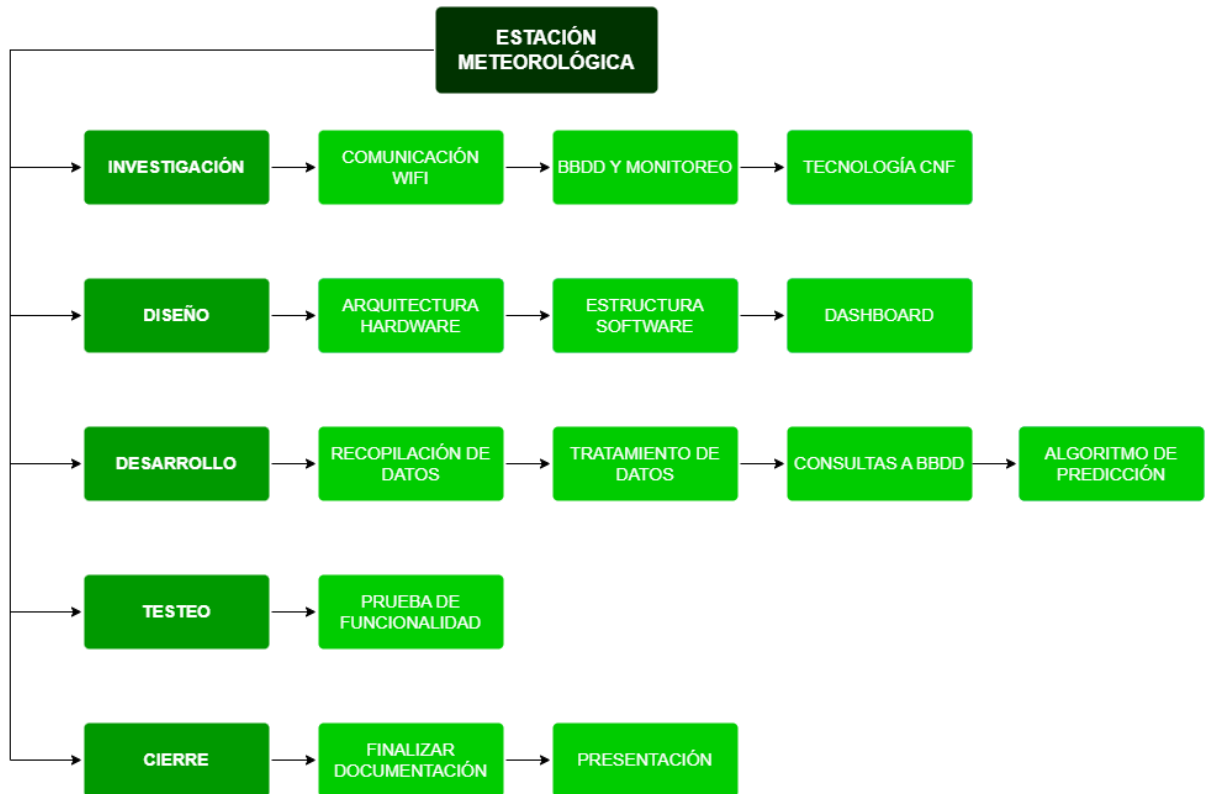
Una de las principales ventajas de Kanban es que permite una mayor flexibilidad y adaptabilidad a los cambios. En lugar de seguir un plan predeterminado, se pueden añadir o eliminar tareas en función de las necesidades del proyecto y el progreso de las tareas existentes. Además, Kanban permite a los miembros del equipo ver claramente el progreso del proyecto y detectar problemas o cuellos de botella en el flujo de trabajo.

Otra ventaja de Kanban es que promueve una comunicación más efectiva dentro del equipo, ya que todas las tareas y responsabilidades quedan claramente definidas en el tablero visual. Esto permite a los miembros del equipo trabajar de manera más colaborativa y reducir la sobrecarga de trabajo en una sola persona.

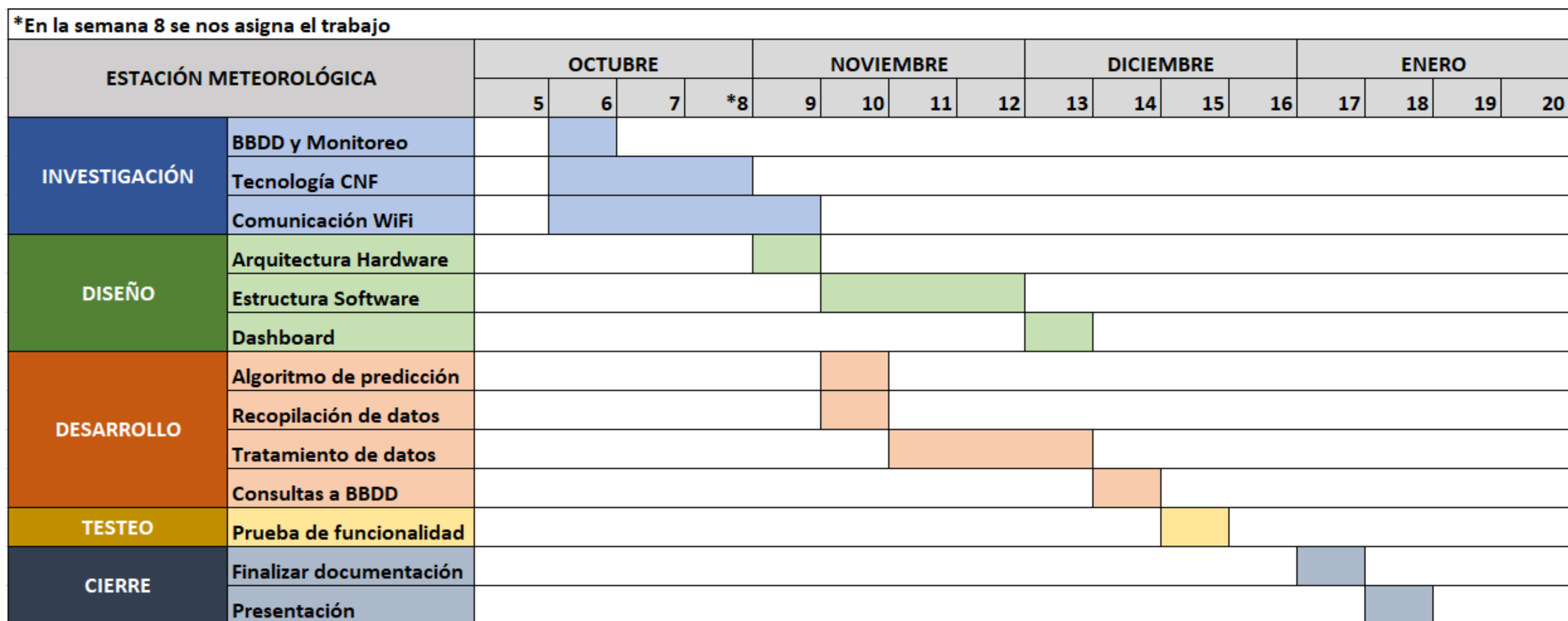
En resumen, la metodología Kanban es una herramienta útil para mejorar la eficiencia y la efectividad del proceso de trabajo al permitir una mejor planificación y una mayor flexibilidad para adaptarse a los cambios. Es especialmente útil en proyectos que requieren un alto grado de adaptabilidad y colaboración entre los miembros del equipo.

4. Planificación

4.1 Estructura de desglose de trabajo



4.2 Diagrama de Gantt



5. Bibliografía

Citas del documento:

[1] *Accessing apps*. minikube. (2022, December 28). Retrieved January 10, 2023, from <https://minikube.sigs.k8s.io/docs/handbook/accessing/>

[2] Docker. (n.d.). Retrieved January 10, 2023, from <https://hub.docker.com/r/grafana/grafana>

[3] Docker. (n.d.). Retrieved January 10, 2023, from <https://hub.docker.com/r/prom/prometheus>

[4] Docker. (n.d.). Retrieved January 10, 2023, from https://hub.docker.com/_/python

[5] Google. (n.d.). *Aplicaciones Menos Seguras y la cuenta de google - ayuda de cuenta de google*. Google. Retrieved January 10, 2023, from <https://support.google.com/accounts/answer/6010255?hl=es>

[6] Google. (n.d.). *Iniciar Sesión Con contraseñas de Aplicación - Ayuda de Cuenta de Google*. Google. Retrieved January 10, 2023, from <https://support.google.com/accounts/answer/185833?hl=es>

[7] Wikimedia Foundation. (2022, December 13). *Prometheus (software)*. Wikipedia. Retrieved January 10, 2023, from [https://en.wikipedia.org/wiki/Prometheus_\(software\)](https://en.wikipedia.org/wiki/Prometheus_(software))

[8]Wikimedia Foundation. (2022, September 6). *Grafana*. Wikipedia. Retrieved January 10, 2023, from <https://es.wikipedia.org/wiki/Grafana>

[9]¿Qué es wi-fi?: Definición, Significado y Explicación. verizon.com. (n.d.). Retrieved January 10, 2023, from <https://espanol.verizon.com/articles/internet-essentials/wifi-definiton/>

[10]Admin. (1970, March 31). *WIFI 4 vs. WIFI 5 vs. WIFI 6 vs. Wigig: ¿cual es la diferencia? 802.11ax vs. 802.11ad vs. 802.11ac vs. 802.11n? PC Ahora*. Retrieved January 11, 2023, from <https://pcahora.com/wifi-4-vs-wifi-5-vs-wifi-6-vs-wigig/>

[11]Fernández, Y. (2018, March 16). *WIFI 2.4ghz y 5GHz: Cuáles Son Las diferencias YCuál Elegir*. Xataka. Retrieved January 11, 2023, from <https://www.xataka.com/basics/wifi-2-4g-y-5g-cuales-son-las-diferencias-y-cual-elegir>

[12]Las próximas cloud-native network functions (cnfs) de big-IP. F5. (n.d.). Retrieved January 11, 2023, from https://www.f5.com/es_es/products/cloud-native-network-functions

[13]López, A. (2022, October 4). *Aprende todo sobre las bandas de Frecuencia Wi-Fi: 2.4GHz, 5GHz Y 6ghz*. RedesZone. Retrieved January 11, 2023, from <https://www.redeszone.net/tutoriales/redes-wifi/bandas-frecuencias-wi-fi/>

[14]Research, T. (2022, November 2). *Análisis del Espectro WIFI, que es y cómo realizarlo*. Acrylic WiFi. Retrieved January 11, 2023, from <https://www.acrylicwifi.com/blog/analisis-del-espectro-wifi/>

Citas útiles para el desarrollo del sistema:

Ahnaf-Zamil. (n.d.). *Ahnaf-Zamil/k8s-flask-tutorial: A practical overview of deploying a flask application on kubernetes*. GitHub. Retrieved January 10, 2023, from <https://github.com/ahnaf-zamil/k8s-flask-tutorial>


Gallego, M. (2018, August 19). *Prometheus: Exportando Datos del Sistema*. MoiDev · MoiDev. Retrieved January 10, 2023, from <https://moidev.com/posts/prometheus-exportando-datos-del-sistema/>

Grafana. (n.d.). *Grafana/Prometheus-Arduino*. GitHub. Retrieved January 10, 2023, from <https://github.com/grafana/prometheus-arduino>

Hadieht. (n.d.). *HADIEHT/Kubernetes-Monitoring-Prometheus-Grafana: Montior Kuberntese with Prometheus and Grafana and alert with email with using Alertmanager*. GitHub. Retrieved January 10, 2023, from <https://github.com/hadieht/Kubernetes-Monitoring-Prometheus-Grafana>

Host access. minikube. (2021, May 12). Retrieved January 10, 2023, from <https://minikube.sigs.k8s.io/docs/handbook/host-access/>

Prometheus. (n.d.). *Prometheus/client_python: Prometheus Instrumentation Library for python applications*. GitHub. Retrieved January 10, 2023, from https://github.com/prometheus/client_python

Rashid. (2022, June 11). *Setup grafana with prometheus for python projects using Docker*. DEV Community . Retrieved January 10, 2023, from <https://dev.to/thepylot/setup-grafana-with-prometheus-for-python-projects-using-docker-4o5g>

ShubhamKhairnar. (2020, November 19). *Monitoring/configmap.yml at master · Shubhamkhairnar/monitoring*. GitHub. Retrieved January 10, 2023, from <https://github.com/ShubhamKhairnar/monitoring/blob/master/configmap.yml>

Valla, S. (2022, August 14). *Alertmanager setup on Kubernetes for Prometheus Monitoring*. DevOps Counsel. Retrieved January 10, 2023, from <https://devopscounsel.com/alertmanager-setup-on-kubernetes-for-prometheus-monitoring/>

▷ *tutorial de prometheus Y grafana - monitorización de sistemas: [puerto53.com]*. puerto53.com. (2022, December 4). Retrieved January 10, 2023, from <https://puerto53.com/linux/tutorial-de-prometheus-monitorizacion-de-sistemas/>

ATTRIBUTEERROR module 'requests' has no attribute 'get'. bobbyhadz. (n.d.). Retrieved January 10, 2023, from <https://bobbyhadz.com/blog/python-attributeerror-module-requests-has-no-attribute-get>

Sparkfun WiFly Shield. WRL-09954 - SparkFun Electronics. (n.d.). Retrieved January 10, 2023, from <https://www.sparkfun.com/products/retired/9954>

Sparkfun. (n.d.). *Sparkfun/sparkfun_wifly_shield_arduino_library: Arduino Library for the Sparkfun WiFly Shield*. GitHub. Retrieved January 10, 2023, from https://github.com/sparkfun/SparkFun_WiFly_Shield_Arduino_Library

user1639431, user1639431user1639431 3, Jon ClementsJon Clements 136k3232 gold badges242242 silver badges275275 bronze badges, Martin AtkinsMartin Atkins 53.8k55 gold badges108108 silver badges123123 bronze badges, & PoLoMoToPoLoMoTo 2144 bronze badges. (1960, March 1). *How can I send a get request from my flask app to another site?* Stack Overflow. Retrieved January 10, 2023, from <https://stackoverflow.com/questions/15463004/how-can-i-send-a-get-request-from-my-flask-app-to-another-site>

WiFly command reference, advanced features ... - SparkFun Electronics. (n.d.). Retrieved January 10, 2023, from <https://cdn.sparkfun.com/datasheets/Wireless/WiFi/rn-wiflycr-ug-v1.2r.pdf>

YouTube. (2017, December 11). *18.- requests (get, API to API) - curso flask*. YouTube. Retrieved January 10, 2023, from <https://www.youtube.com/watch?v=8mH-ujnV0Zc&t=114s>