UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE

"INTEGRACIÓN DE ESTACIONES DE MEDICIÓN MULTIPARAMÉTRICAS PARA MONITOREO METEOROLÓGICO REMOTO"

PABLO ANDRÉS PEÑA LEÓN

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO.

PROFESOR GUÍA: MANUEL OLIVARES.

PROFESOR CORREFERENTE: ALEX FLORES MARADIAGA.

SEPTIEMBRE - 2021

Agradecimientos

Quisiera darle las gracias a mucha gente, pero los que más se merecen mi agradecimientos es mi familia. Gracias papá y mamá por el apoyo incondicional en la parte de mi vida que formará el resto de mi futuro y por creer en mi cuando las metas eran cada vez más difíciles. Gracias a mis hermanos Christian y Felipe que siempre están presentes cuando necesito apoyo, motivación o simplemente para des-estresarse jugando a la pelota. Gracias a mi pareja, Mariel Opazo, por la motivación diaria y cariño constante durante uno de los periodos más esforzados que he vivido.

A mis compañeros, quisiera darles las gracias por toda la conversación y buenos tiempos que han salido durante nuestros años de estudio. En especial le doy las gracias a mis compañeros de laboratorio Diego Concha y Sebastián "Tota" Adamas que entre tantas risas compartidas logramos llegar al final.

Finalmente le doy las gracias a mis profesores, Manuel Olivares y Alex Flores por la orientación en un tema que no era familiar para mi. Les doy las gracias por apoyarme y por la paciencia durante este proceso largo causado por la pandemia.

Pablo Andrés Peña León

Resumen

El Laboratorio de Energías Renovables (LER) requiere de un sistema de adquisición de datos meteorológicos para proyectos futuros. Esta memoria de título consiste en un primer acercamiento a una base de datos de mediciones meteorológicas y su visualización a través de un cliente web para el LER. Se estudiaron distintos tipos de estaciones multiparamétricas con el fin de elegir cuatro estaciones que sean capaces de funcionar en una proximidad cercana sin causar interferencias en el envío inalámbrico de sus mediciones.

Considerando lo anterior, se eligieron cuatro estaciones modelo Vantage Pro 2 para ser utilizadas en el proyecto con el objetivo de ser integradas a una microcomputadora Raspberry Pi. La conexión de las cuatro estaciones a la Raspberry Pi fue posible al utilizar un dispositivo de adquisición de datos llamado Envoy8x. Así, fue posible concentrar las mediciones multiparamétricas inalámbricas de las cuatro estaciones en un dispositivo capaz de transmitir dichas mediciones de manera cableada.

Fue posible instalar el sistema operativo llamado Meteobridge a la Raspberry Pi, transformándola en un mini servidor capaz de transmitir las mediciones a una base de datos remota. Con esto, se logró crear un servidor con una base de datos mySQL que permite almacenar una gran capacidad de datos y entregarlos de manera rápida y eficiente al usuario final.

Finalmente, se desarrolló un interfaz para el usuario final a través de un cliente web que permite visualizar y descargar los datos en un formato conveniente. Esta plataforma quedará disponible para todos los usuarios y será utilizada para estudiar distintas características, como la serie de tiempo y el perfil del viento y otras variables meteorológicas de interés, contribuyendo así, a la docencia, la investigación científica y el desarrollo de proyectos en el LER.

Keywords: Estaciones multiparamétricas, Estaciones ISS, Automatic Weather Station (AWS), Base de datos, Servidor, MySQL, PHP, Meteobridge, Envoy8x, USB, API, Highcharts.

Abstract

The Renewable Energies Laboratory (REL) requires a data adquisition system of meteorological data for future projects. This thesis consists of a first approach towards creating a database of meteorological data and its visualization through an online client for the REL. Different multiparametric weather stations were studied with the purpose of selecting four stations capable of working in close proximity without interfering in the wireless transmission of the data.

For this project, four stations of model Vantage Pro 2 were chosen to be integrated into a Raspberry Pi microcomputer. The connection between the four stations and the Raspberry Pi was made possible by using a datalogger device called Envoy8x. With this, it was possible to log the wireless multiparametric data of the four stations into a device that has the capabilty of storing and transferring the data through a cabled connection.

The Raspberry Pi was flashed with an operating system named Meteobridge that transforms the Raspberry Pi into a microserver capable of transmitting the logged data to a remote database. This allowed for the creation of a server with a mySQL database which can hold large amounts of data and can supply the information to the end user in a quick and efficient manner.

Finally, a user interface was created through an online client in order to visualize and download the meteorological data in a convenient format. This platform will be available for all users and will be utilized to study different characteristics, like the time series and vertical profiles, of the wind and other meteorological variables of interest, leading to a contribution to teaching, scientific investigation and the development of projects for the REL.

Glosario

- **AWS:** Automatic Weather Station (Estación meteorológica automática).
- **ISS:** Integrated Sensor Suite (Estación meteorológica con múltiples sensores integrados).
- **Vantage Pro 2:** Nombre del modelo de la estación meteorológica ISS.
- **Meteobridge:** Sistema operativo de adquisición de datos meteorológicos para Raspberry Pi.
- **LAMP:** Stack Linux-Apache-MySQL-PHP para USB Webserver.
- **Envoy8x:** Receptor inalámbrico concentrador de hasta 8 ISS.
- **WDTU:** Weather Data Transfer Utility Software de configuraciónde la Envoy8x para PC.
- **USB Weatherlink:** Dispositivo para conector la Envoy8x con la Raspberry Pi mediante un puerto USB.
- **PWS-Wunderground:** Personal Weather Station Network- Weather Underground.

Índice general

1. Introducción **1**

1.1. Problema a resolver y objetivos 2

1.2. Requerimientos funcionales 2

1.2.1. Integración de múltiples estaciones meteorológicas 2

1.2.2. Adquisición de datos de múltiples estaciones en red 3

1.2.3. Desarrollo de servidor de base de datos en red 3

1.2.4. Desarrollo de cliente web HMI 3

1.2.5. Visualización de datos históricos y en tiempo real a través de la aplicación web 3

1.3. Estado del Arte 4

1.3.1. Estaciones meteorológicas automáticas AWS 4

1.3.2. Procesamiento de datos meteorológicos 9

1.3.3. Sistema actual de medición meteorológica en el LER . . . 9

2. Instrumentación de la solución propuesta **11**

2.1. Arquitectura de la solución propuesta 11

2.2. Componentes Hardware 13

2.2.1. Estación meteorológicas ISS - *Davis Vantage Pro 2* 13

2.2.2. Consola receptora *Davis Vantage Pro2* 14

2.2.3. Concentrador y adquisidor de datos *Davis Envoy8x* 15

2.2.4. Adaptador *Davis WeatherLink USB Datalogger* 16

2.2.5. Raspberry Pi 3 Modelo B+ 17

2.2.6. Pendrive USB 17

2.3. Software 18

2.3.1. *Weather Data Transfer Utility* (WDTU) 18
2.3.2. Sistema operativo Raspbian 19
2.3.3. Meteobridge 19
2.3.4. Webserver LAMP: Linux-APACHE-MySQL-PHP 21
2.3.5. Cliente Web: Navegador 22

3. Configuración de la Red de Estaciones Meteorológicas 24
3.1. Preparación de las estaciones Vantage Pro 2 24
3.1.1. Preparación de la estación transmisora (ISS) 24
3.1.2. Preparación de la consola receptora 32
3.2. Prueba de comunicación inalámbrica de la red de estaciones con las consolas Vantage Pro 2 34
3.3. Conexión de las consolas Vantage Pro 2 al concentrador Envoy8x 36

4. Adquisición de datos con la Envoy8x 37
4.1. Preparación de la Envoy8x 37
4.1.1. Configuración de hardware 37
4.1.2. Configuración de software 38
4.2. Prueba de almacenamiento de datos de múltiples estaciones 40
4.3. Conexión de la Envoy8x a la Raspberry Pi 41

5. Adquisición de datos en la Raspberry Pi 42
5.1. Instalación del sistema operativo Meteobridge 42
5.2. Configuración de Meteobridge 43
5.2.1. Asignación de IP para la Rasberry Pi mediante LAN/Ethernet 43
5.2.2. Selección de estaciones meteorológicas deseadas 44
5.2.3. Configuración de exportación de datos a servicios online . 44
5.2.4. Exportación a la base de datos MYSQL 46
5.3. Pruebas de adquisición de datos de la red de estaciones con la Raspberry Pi 49

6. Configuración de servidor y base de datos **50**

6.1. Desarrollo de servidor USB portátil LAMP 50

6.2. Configuración de base de datos MYSQL 51

6.3. Pruebas de accesibilidad remota a los datos adquiridos con la Raspberry Pi 54

7. Desarrollo de un cliente web y Resultados **55**

7.1. Creación del formato de la pagina web 55

7.2. Desarrollo de la visualización de los datos 57

7.3. Desarrollo de la interfaz para el usuario final 59

7.3.1. Visualización de rapidez de viento 62

7.3.2. Visualización de temperatura 68

7.3.3. Visualización de presión 73

7.3.4. Visualización de humedad 74

7.3.5. Rosa de viento 76

8. Conclusiones **84**

8.0.1. Conclusiones 84

8.0.2. Trabajo futuro 85

A. Códigos Desarrollados: **89**

A.1. Página principal 89

A.2. Códgio para la temperatura 90

A.3. Código para el viento 103

A.4. Código para la humedad 121

A.5. Código para la presión 133

A.6. Código para el estilo de la página 137

Índice de figuras

1.1. Diagrama conceptual para el desarrollo del proyecto. 3
1.2. Red de sensores integrados a un transmisor inalámbrico. 5
1.3. Configuración estándar de sensores de una AWS. 6
1.4. Transmisor de datos con panel solar. 7
1.5. Receptor inalámbrico de mediciones para AWS. 7
1.6. Servidor portátil para subir mediciones locales a la web. 8
1.7. Ejemplo de gráficos para el perfil del viento. 9
1.8. Diagrama de bloques de los instrumentos actualmente instalados en el LER. 10

2.1. Diagrama conceptual para la configuración final del proyecto. . . . 12
2.2. Estación multiparamétrica seleccionada para la memoria. 13
2.3. Consola receptora para la estación multiparamétrica. 14
2.4. Dispositivo de adquisición de datos para los receptores. 15
2.5. Adaptador USB WeatherLink para el dispositivo Envoy8x. 16
2.6. Microcomputadora Raspberry Pi con capacidad de comunicación vía WiFi. 17
2.7. Pendrive USB para el servidor portátil. 17
2.8. “Weather Data Transfer Utility” software para la Envoy8x. 18
2.9. Sistema operativo Meteobridge para la Raspberry Pi accesible en la web. 20
2.10. Software para la configuración de la pendrive USB. 22
2.11. Ejemplo de la interfaz de usuario de la página web Wunderground. 23

3.1. Orientación del soporte para el anemómetro y veleta de viento. . . 25

3.2. Extracción del imán del sensor de lluvia. 25
3.3. Reemplazo del imán del sensor de lluvia. 26
3.4. Levantar la tapa y deslizar hacía arriba. 26
3.5. Desconectando el cable del panel solar. 27
3.6. Conexión de los sensores. 27
3.7. Cableado interior de una estación ISS. 28
3.8. Esquemática de la caja transmisora y DIP switches. 29
3.9. Configuraciones posibles de los DIP switches. 30
3.10. Configuración de DIP switches para la estación número 3. 30
3.11. Ilustración de la estación ISS armada. 31
3.12. Estaciones armadas en el Laboratorio Shannon. 31
3.13. Dirección de giro de la antena. 32
3.14. Canal por defecto de recepción. 32
3.15. Indicación de canales recibidos. 33
3.16. Unidades de medición por defecto. 33
3.17. Consola receptora 1. 34
3.18. Consola receptora 2. 35
3.19. Consolas recibiendo distintas direcciones de viento, sin interferencia. 35

4.1. Interior de la Envoy8x, sin cable weatherlink USB incluido. 37
4.2. Cable Weatherlink USB. 38
4.3. Interior de la Envoy8x, con cable weatherlink USB instalado. . . . 38
4.4. Envoy8x lista para ser configurada a través de su software. 38
4.5. Esquema de conexión de la Envoy8x a un PC. 39
4.6. Configuración final de la Envoy8x con las cuatro estaciones ISS. . 39
4.7. Estado de conexión de las cuatro estaciones ISS, mostrado en pantalla PC y WDTU. 40
4.8. Recepción y almacenamiento de datos en la Envoy8x vista desde un PC y WDTU cada un minuto. 41
4.9. Esquema de conexión de la Envoy8x a la Raspberry Pi. 41

5.1. Interfaz del programa Etcher para montar la imagen a la micro SD. 42

5.2. Instalación de sistema operativo Meteobridge a través de tarjeta micro SD 43
5.3. Interfaz de Meteobridge, mostrando la configuración final de la Raspberry Pi. 43
5.4. Interfaz de Meteobridge, selección de estación/dispositivo meteorológico. 44
5.5. Registro de dispositivo en la página web Wunderground. 45
5.6. Interfaz de Meteobridge, configuración de servicio Wunderground. 45
5.7. Estado del dispositivo en la página Wunderground. 45
5.8. Configuración de la conexión a la base de datos. 46
5.9. Configuración de la exportación a la base de datos. 47
5.10. Últimas mediciones obtenidas por la Raspberry Pi. 49

6.1. Formato final del servidor portátil USB. 50
6.2. Archivos de configuración dentro de la carpeta “settings”. 51
6.3. Configuración del archivo “usbwebserver”. 51
6.4. Creación de la base de datos “meteobridge” con tablas para cada estación. 52
6.5. Interfaz para definir la estructura de la base de datos. 52
6.6. Estructura de la primera estación Vantage Pro 2. 53
6.7. Variables dentro de la base de datos con sus valores deseados. . . 54

7.1. Página inical básica generada a través de HTML. 57
7.2. Interfaz de selección de mediciones para el usuario. 62
7.3. Perfil vertical del viento simulado. 65
7.4. Serie de tiempo para la velocidad promedio del viento simulado. . 68
7.5. Perfil vertical de la temperatura. 70
7.6. Serie de tiempo para las temperaturas. 72
7.7. Captura de pantalla de la página web que muestra las opciones para seleccionar el rango de fechas y estaciones (alturas) a graficar en la serie de tiempo. La serie de tiempo visualiza las temperaturas a 2 y 50 metros de altura entre el 24 y 26 de junio. 72
7.8. Serie de tiempo para la presión de la estación de 2 metros de altura . 74
7.9. Serie de tiempo para las humedades. 76

7.10. Rosa de viento para la velocidad promedio de una estación. 82

7.11. Captura de pantalla de la página web que muestra las opciones para seleccionar el rango de fechas y estación (altura) a graficar en la rosa de viento. La rosa de viento visualiza la frecuencia del viento en cada punto cardinal entre el 24 y 27 de junio. 83

7.12. Detalle de la frecuencia de viento para la figura 7.11. 83

8.1. Gabinete de doble puerta con dimensiones 1000 x 800 x 250 mm. . 86

8.2. Configuración para una instalación apropiada de las pantallas receptoras y componentes del sistema. 87

Índice de tablas

1.1. Instrumentación actual de la torre patrón de 28 metros de altura. 10

2.1. Resoluciones e intervalos de actualización de los sensores. 13
2.2. Especificaciones de la consola receptora. 14
2.3. Especificaciones de la Envoy8x. 15
2.4. Especificaciones de la Raspberry 3 B+. 17
2.5. Especificaciones de la Pendrive USB. 17

5.1. Mapeo de variables para la base de datos mySQL. 48

7.1. Rangos de magnitud definidos para la rosa de viento. 77

Capítulo 1

Introducción

El monitoreo remoto de variables meteorológicas es de gran importancia para el desarrollo de las energías renovables modernas debido a que permite encontrar ubicaciones con condiciones meteorológicas óptimas mediante el estudio del potencial eólico. En los últimos años ha aumentado el uso de estaciones meteorológicas electrónicas debido a la prohibición mundial de la producción, importación y exportación de instrumentos que contienen mercurio desde el año 2017. Estas estaciones permiten la implementación automática de la transmisión de los parámetros medidos, pudiéndose convertir en *Estaciones Meteorológicas Automáticas* o AWS por su sigla en inglés. Las AWS sirven para obtener una mayor cantidad de mediciones en terreno y aumentar su confiabilidad debido a la capacidad para entregar información durante las 24 horas del día y la reducción de errores humanos. Además de esto, facilita la integración futura de estaciones adicionales permitiendo la expansión de la red de obtención de datos.

El Laboratorio de Energías Renovables (LER) requiere de un sistema de datos meteorológicos para desarrollar docencia, proyectos, asistencia técnica e investigación científica. En esta memoria, la capacidad de integrar varias estaciones es fundamental, puesto a que se utilizarán para generar perfiles verticales de varias variables de interés. Además, se realizará una unificación y automatización de la obtención de datos, creando una torre de AWS autónoma para el LER

En este documento se detallará el proceso para implementar un sistema de adquisición de datos meteorológicos. Se empezará por las estaciones que medirán los datos y la configuración inicial requerida para lograr capturar mediciones en los formatos deseados. Luego, se explicará el proceso para conectar las estaciones meteorológicas a un concentrador de datos. Siendo este dispositivo el que permite la conexión de las cuatro estaciones a una sola micro-computadora que estará encargada de capturar y transmitir los datos a un servidor remoto. Finalmente, se detallarán los pasos para configurar una base de datos que será utilizada para acceder a las mediciones mediante una pagina web con herramientas de análisis

para el usuario final.

1.1. Problema a resolver y objetivos

El Laboratorio de Energías Renovables requiere un sistema de adquisición de datos meteorológicos para generar investigación científica, desarrollar docencia y proyectos. El Proyecto pretende integrar, mediante un microcomputador Raspberry Pi, múltiples estaciones meteorológicas que se ubicarán a distintas alturas en el LER de la sede de Viña del Mar de la UTFSM. Se podrá acceder a la base de datos de manera remota online, con la finalidad de obtener datos estadísticos que permitan estudiar las distintas características del viento y otras variables meteorológicas de interés. Los resultados obtenidos en este trabajo buscan cumplir con los siguientes objetivos:

- Integrar cuatro estaciones de mediciones multiparamétricas a una Raspberry Pi
- Crear un sistema de adquisición periódica e historización de datos meteorológicos
- Permitir acceso a las mediciones de manera local a través de un Interfaz Hombre Maquina
- Permitir acceso a las mediciones de manera remota a través de internet para cualquier usuario
- Desarrollar herramientas que permitan un análisis posterior de las variables
- Desarrollar código que permita visualizar los perfiles de viento y temperaturas a diferentes alturas

1.2. Requerimientos funcionales

1.2.1. Integración de múltiples estaciones meteorológicas

Se busca obtener mediciones periódicas de variables meteorológicas como dirección y rapidez del viento, humedad, presión, precipitación y temperatura desde cuatro alturas distintas. Para lograr esto, se requiere de cuatro estaciones multiparamétricas instaladas en proximidad cercana. Las cuatro estaciones serán instaladas a 2, 10 , 30 y 50 metros de altura.

1.2.2. Adquisición de datos de múltiples estaciones en red

Los datos recibidos desde las estaciones deben ser almacenados en un sistema de adquisición de datos que permita guardar una gran cantidad de mediciones. Este sistema debe ser accesible de manera presencial, en terreno, para tener un respaldo físico de la información medida.

1.2.3. Desarrollo de servidor de base de datos en red

La información obtenida a través del sistema de adquisición de datos debe ser accesible de manera remota, online, para evitar tener que ir a terreno cada vez que se requiera obtener nueva data.

1.2.4. Desarrollo de cliente web HMI

Una vez obtenidas las mediciones online, el usuario final debe ser capaz de acceder a la información obtenida sin uso de terminales o protocolos avanzados. La pagina web final deber tener un interfaz hombre maquina que sea fácil de utilizar para todo tipo de usuario.

1.2.5. Visualización de datos históricos y en tiempo real a través de la aplicación web

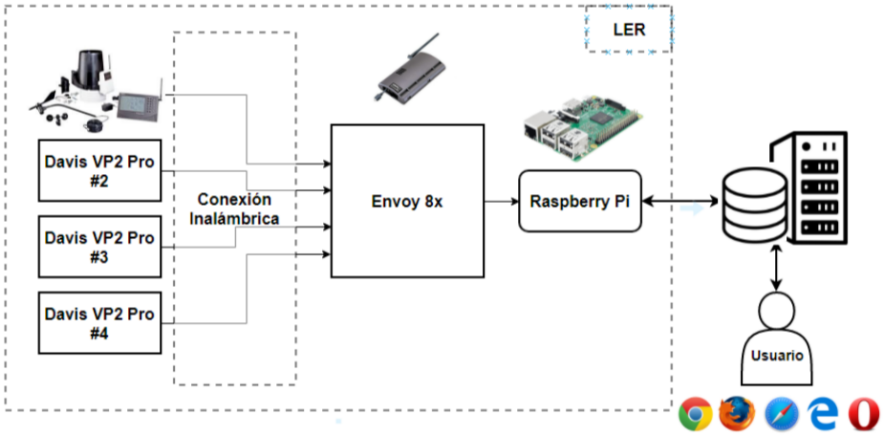


Figura 1.1: Diagrama conceptual para el desarrollo del proyecto.

Junto con obtener los datos medidos de manera tabulada, el usuario debe ser capaz de visualizar la información. Se deben graficar los datos de un periodo de tiempo deseado, permitiendo obtener perfiles de las variables medidas y ayudar

con el posprocesamiento de las mediciones obtenidas. En la figura 1.1 se muestra el diagrama conceptual del proyecto.

1.3. Estado del Arte

A continuación, se han de describir a grandes rasgos los usos, implementaciones, procesos y aplicaciones relevantes para la comprensión del estado del arte que circunda esta memoria.

1.3.1. Estaciones meteorológicas automáticas AWS

Existe una guía a nivel mundial para la creación e implementación de estaciones de calidad. Las recomendaciones y observaciones están detalladas en [1] donde se explica en detalle los tipos de sensores que deberían ser implementados y la calidad necesaria para lograr tener una estación certificada en USA. Se utiliza esta guía debido a que no existe un organismo certificador en Chile.

Una estación meteorológica automática consiste en una red de sensores que están generalmente conectados a un transmisor mediante cables *ethernet* o LAN. Estas estaciones están formadas por sensores de temperatura, humedad, presión, precipitación, anemómetros y veletas de viento, siendo alimentadas a través de una batería recargable conectada a un panel solar. En la figura 1.2 se visualiza una configuración típica de una AWS.

A continuación se describen los componentes de una AWS.

1.3.1.1. Sensores:

Una *AWS* genérica se conforma de dos sensores de viento, un sensor de temperatura, humedad, presión y un sensor de precipitación. Estos sensores son alimentados a través de un panel solar que permite el trabajo continuo autónomo de la AWS. Estaciones más avanzadas suelen tener sensores ultravioleta y de radiación solar.

Normalmente, los sensores de viento están integrados dentro de un solo dispositivo con una veleta para medir la dirección del viento y un anemómetro para medir su velocidad. Estos sensores están montados sobre el sensor de temperatura que va junto al sensor de humedad. En la siguiente figura se puede apreciar la configuración estándar.

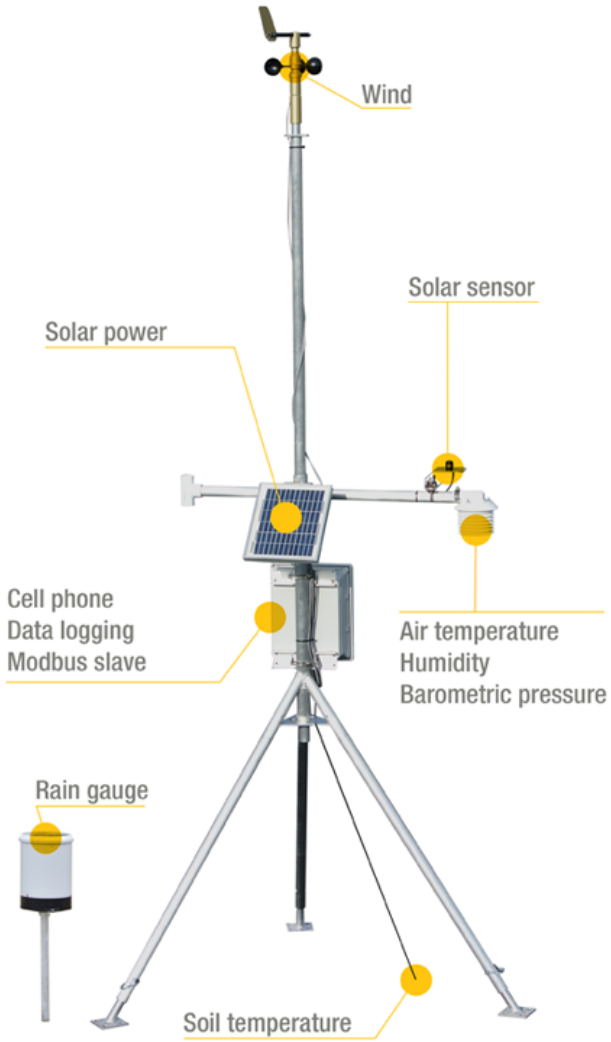


Figura 1.2: Red de sensores integrados a un transmisor inalámbrico.

Figura 1.3: Configuración estándar de sensores de una AWS.

1.3.1.2. Transmisión inalámbrica

Los sensores de las AWS van conectados a una caja transmisora que usualmente tiene un panel solar para su funcionamiento autónomo y alimenta la transmisión de los datos capturados. Esta caja solo transmite los datos y no los almacena, el receptor se encarga de almacenar los datos y mostrarlos en una pantalla táctil para sus usuarios.

1.3.1.3. Adquisición y almacenamiento de datos en un formato único

La adquisición de datos se realiza mediante un receptor inalámbrico ubicado en proximidad al transmisor, típicamente dentro de 300 metros de distancia, y cuenta con espacio para almacenar meses de información para una única estación meteorológica.

Figura 1.4: Transmisor de datos con panel solar.

Figura 1.5: Receptor inalámbrico de mediciones para AWS.

1.3.1.4. Servidores para estaciones meteorológicas

Existen servidores que se encargan de subir la información obtenida por las estaciones meteorológicas a una pagina web, sin embargo, generalmente funcionan para una estación a la vez y no permiten que el usuario haga procesamiento adicional al que ya entrega.

Figura 1.6: Servidor portátil para subir mediciones locales a la web.

Una de las páginas usadas para este tipo de proyecto es *Wunderground* [3], donde los usuarios pueden registrar sus propias estaciones meteorológicas y revisar las mediciones en la web pagando una suscripción de 50 USD por año. Otras páginas más completas son el Explorador Eólico de Chile [5], actualmente no activa, y la Dirección Meteorológica de Chile [4], donde parece ser posible contactarse con ellos y registrar estaciones dentro de todo el país.

1.3.2. Procesamiento de datos meteorológicos

Una de las variables más importantes del proyecto es la velocidad del viento y su perfil vertical debido a que muestra cambios por efecto de la altura o por obstáculos en su camino. Se busca generar gráficos de perfiles del viento mediante los datos medidos debido a que puede ser beneficioso en los cálculos para cuantificar el potencial energético de una ubicación En el paper "Cuantificación del perfil del viento hasta 100 m de altura desde la superficie y su incidencia en la climatología eólica" [13] se detalla la importancia de este tipo de análisis, obteniendo gráficos como los de la figura 1.7.

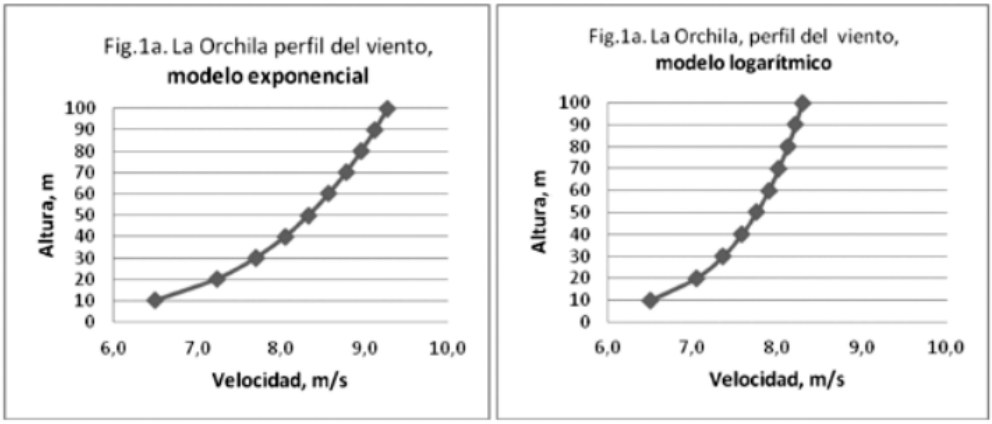


Figura 1.7: Ejemplo de gráficos para el perfil del viento.

1.3.3. Sistema actual de medición meteorológica en el LER

Actualmente en el LER la única torre con instrumentación es la de 28 metros. Esta torre dispone de dos anemómetros para medir la velocidad del viento, una veleta de dirección, un sensor de temperatura y un sistema de adquisición de datos en la base de la torre. A continuación, se detallan los modelos de los dispositivos instalados.

Instrumento	Modelo	Altura
Anemómetro de 3 copas	NRG#40C	28 metros
Anemómetro de 3 copas	NRG#40C	18 metros
Veleta de dirección	NRG#200P	28 metros
Adquisición de datos	Campbell Datalogger CR850	3 metros
Fuente de alimentación	Campbell Scientific PS100	3 metros
Sensor de temperatura	Termocupla Type K TC	-

Cuadro 1.1: Instrumentación actual de la torre patrón de 28 metros de altura.

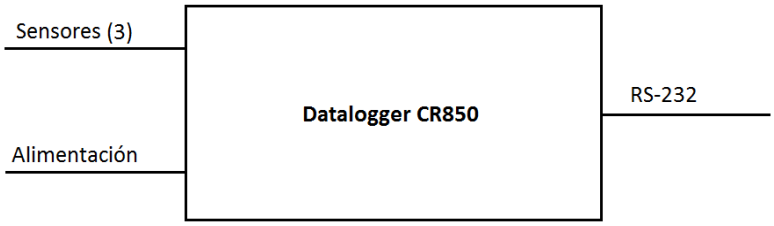


Figura 1.8: Diagrama de bloques de los instrumentos actualmente instalados en el LER.

El primer anemómetro de 3 copas y la veleta de dirección están instalados a 28 metros de altura con orientación hacía el norte, tal como lo indica el fabricante. Luego, el segundo anemómetro de 3 copas esta instalado a 18 metros de altura y el sistema de adquisición de datos, que incluye la fuente de poder Campbell Scientific PS100 y el Datalogger CR850, a una altura de 3 metros dentro de un gabinete accesible. La fuente de poder PS100 esta conectada a la red de la universidad (220 VAC) a través de un cargador de pared (wall charger) e incluye un regulador de carga y una batería interna de 7-Ah que alimenta el CR850 directamente.

Capítulo 2

Instrumentación de la solución propuesta

2.1. Arquitectura de la solución propuesta

Existen varios problemas que se deben solucionar en este proyecto para obtener perfiles verticales de variables meteorológicas. La más importante siendo que las estaciones elegidas no deben tener interferencia entre ellas. Para lograr esto, se utilizan estaciones que transmiten sus mediciones a distintas frecuencias o por distintos canales.

Otro problema es la concentración de los datos de las cuatro estaciones meteorológicas. Estas deben ser almacenadas en un solo dispositivo, integrando sus receptores que solo permiten el almacenamiento de una estación a la vez. Para esto, se utiliza un concentrador de datos diseñado para múltiples estaciones del mismo modelo llamado Envoy8x. Sin embargo, el dispositivo solamente funciona conectado a un computador y no de manera autónoma. Por esto se utiliza una microcomputadora con un sistema operativo dedicado para acceder a los datos de la Envoy8x y subirlos a un servidor web de base de datos *mySQL* online, en donde los datos en formato *RAW* podrán ser visualizados y descargados.

Inicialmente, se consideró utilizar la API de Google llamada *Google Charts* [7] para visualizar la base de datos *mySQL*, sin embargo, hacerlo de esta manera requería una restructuración de datos y poca variedad en los gráficos posibles. Es por esto que finalmente se decidió usar la API *Highcharts* [6], dado que permite utilizar los datos obtenidos sin grandes cambios en sus formatos y permite una mayor personalización de los gráficos finales.

En la figura 2.1 se muestra la configuración final del proyecto, con la base de datos implementada dentro del servidor del Laboratorio de Energías Renovables.

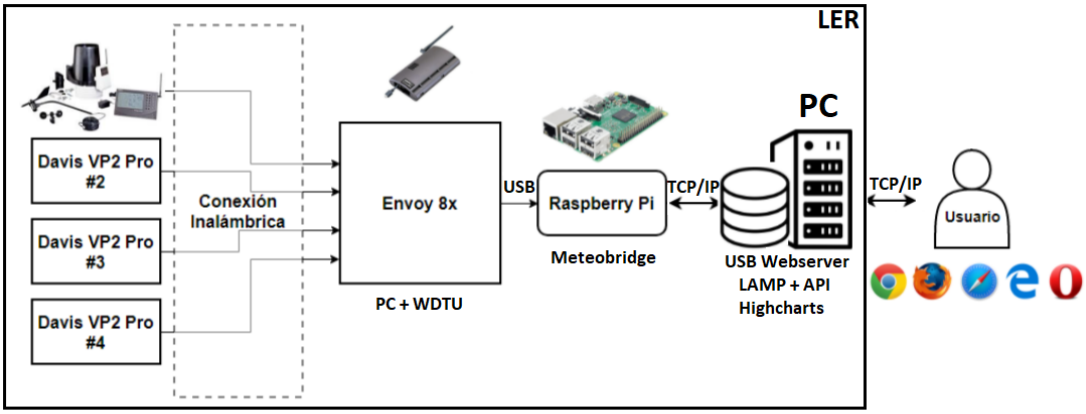


Figura 2.1: Diagrama conceptual para la configuración final del proyecto.

2.2. Componentes Hardware

2.2.1. Estación meteorológicas ISS - *Davis Vantage Pro 2*

En el cuadro 2.1 se detallan las especificaciones más relevantes de la Integrated Sensor Suite (ISS) Vantage Pro 2 según la información disponible del fabricante [8].

Figura 2.2: Estación multiparamétrica seleccionada para la memoria.

Instrumento	**Resolución**	**Actualización**
Anemómetro de 3 copas	0.4 m/s	2.5 - 3 s
Veleta de dirección	16 puntos (22.5°)	2.5 - 3 s
Temperatura	0.1 °C	10 - 12 s
Humedad	1 %	1 min
Presión	0.1 hPa/mb	1 min
Lluvia	0.1 mm	20 - 24 s

Cuadro 2.1: Resoluciones e intervalos de actualización de los sensores.

Los sensores y la transmisión de los datos se alimentan a través de un panel solar de 0.5 watts durante el día y mediante un supercondensador en la noche. Adicionalmente, se cuenta con una batería de 3 volts tipo “CR-123” como respaldo.

2.2.2. Consola receptora *Davis Vantage Pro2*

En el cuadro 2.2 se detallan las especificaciones más relevantes de la consola receptora de la estación Vantage Pro 2 según la información disponible del fabricante [8].

Figura 2.3: Consola receptora para la estación multiparamétrica.

Característica	**Valor**
Corriente de operación	0.9 mA promedio, 30 mA peak
Adaptador de poder AC	5 VDC, 1000 mA, regulado
Baterías	3 Tipo-C
Vida de baterías	Hasta 9 meses
Frecuencia de operación	921 - 928 MHz FHSS, <8 mW
Distancia Linea vista	300 m
Distancia a través de paredes	60 - 120 m

Cuadro 2.2: Especificaciones de la consola receptora.

El tiempo de actualización de los datos de la ISS en la consola varía según la medición. La velocidad del viento se actualiza cada 2.5 segundos y puede ser configurada por el usuario para obtener mediciones cada un minuto o obtener el promedio cada 10 minutos, mientras que otros sensores tienen tiempos de actualización fijos como la humedad, que se actualiza cada 50 segundos. El listado completo de las actualizaciones para cada sensor puede ser encontrado en [8], indicado por “Update Interval”.

2.2.3. Concentrador y adquisidor de datos *Davis Envoy8x*

En el cuadro 2.3 se detallan las especificaciones de la Envoy8x [9].

Figura 2.4: Dispositivo de adquisición de datos para los receptores.

Característica	**Valor**
Corriente de operación	0.9 mA promedio, 20 mA peak
Adaptador de poder AC	5 VDC, 200 mA, regulado
Baterías	3 tipo-AA
Vida de baterías	Hasta 4 meses
Cantidad de canales	8
Distancia Linea vista	300 m
Distancia a través de paredes	75 - 150 m

Cuadro 2.3: Especificaciones de la Envoy8x.

La actualización de las mediciones de datos de cada ISS en la Envoy8x puede ser configurada por el usuario. Los tiempos de actualización son de 10 segundos a 2 horas.

2.2.4. Adaptador *Davis WeatherLink USB Datalogger*

El adaptador *Davis WeatherLink USB Datalogger* [10] habilita la conexión de la Envoy8x a un computador mediante USB. Además de esto, actúa como un tipo de "buffer" que almacena una porción pequeña, 2560 mediciones, de los datos adquiridos y logra modificar el intervalo en que se registran las mediciones en la base de datos e la Envoy8x.

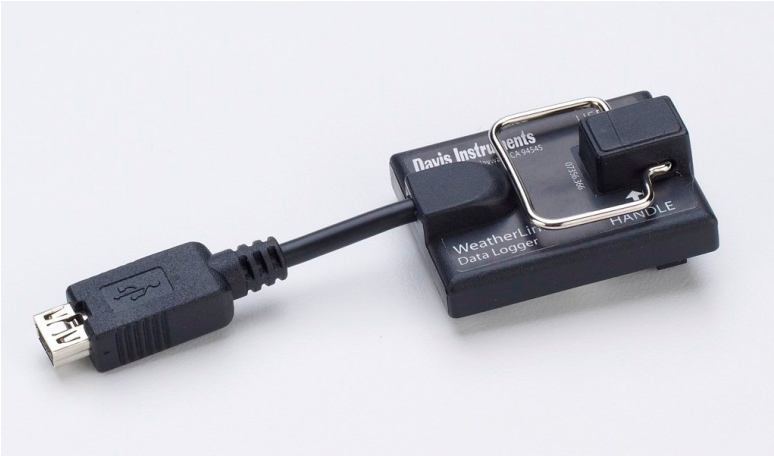


Figura 2.5: Adaptador USB WeatherLink para el dispositivo Envoy8x.

2.2.5. Raspberry Pi 3 Modelo B+

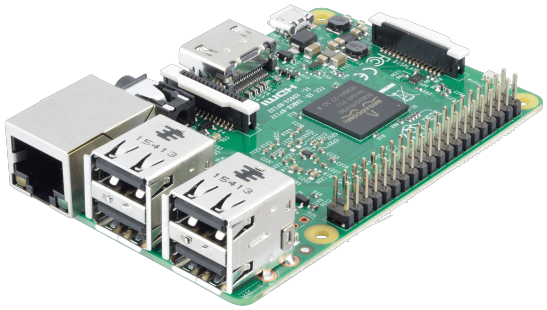


Figura 2.6: Microcomputadora Raspberry Pi con capacidad de comunicación vía WiFi.

Característica	**Raspberry Pi 3 B+**
Procesador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
RAM	1GB LPDDR2 SDRAM
LAN inalámbrica	2.4GHz y 5GHz IEEE 802.11.b/g/n/ac
Bluetooth	Bluetooth 4.2
Video	HDMI
Almacenamiento	Puerto Micro SD para instalación de sistema operativo
Alimentación	5V/2.5A DC

Cuadro 2.4: Especificaciones de la Raspberry 3 B+.

2.2.6. Pendrive USB

Figura 2.7: Pendrive USB para el servidor portátil.

Característica	**Pendrive USB**
Modelo	USB 2.0
Almacenamiento	8 GB

Cuadro 2.5: Especificaciones de la Pendrive USB.

2.3. Software

2.3.1. *Weather Data Transfer Utility* (WDTU)

El sistema de adquisición de datos Envoy8x requiere ser configurado a través de un software incluido llamado "Weather Data Transfer Utility" o *WDTU*. Este programa permite asignar distintos canales a las cuatro estaciones ISS Vantage Pro 2 y configurar las mediciones que serán almacenadas en la memoria interna. Según el fabricante, una Envoy8x conectada a dos estaciones ISS, midiendo cada 15 minutos, puede almacenar más de 365 días de datos. Este intervalo de adquisición de datos puede ser modificado a través de la WDTU.

Es necesario utilizar un computador para la configuración inicial de la Envoy8x, donde además se podrán visualizar las mediciones en tablas o ser exportadas en formato Excel o CSV. Aunque estas opciones son atractivas, solo se pueden acceder si un computador está encendido durante todo el día para evitar la perdida de datos históricos si se logra llenar la memoria interna, generando un alto consumo eléctrico. En las siguientes secciones se detalla como lograr una operación continua de bajo consumo.

Figura 2.8: "Weather Data Transfer Utility" software para la Envoy8x.

2.3.2. Sistema operativo Raspbian

Normalmente se necesita de un sistema operativo basado en Linux o Debian, como Raspbian, para utilizar una Raspberry Pi. Estos sistemas operativos transforman a la Raspberry Pi en un mini computador, permitiendo conectar una pantalla, teclado y mouse para usar todas sus funcionalidades. Con esto, es posible instalar múltiples aplicaciones o programas dentro del dispositivo.

Sin embargo, para este proyecto, se utilizará un sistema operativo dedicado llamado Meteobridge que transforma la Raspberry Pi en un *Mini Weather Server*. Este sistema operativo utiliza toda la capacidad de la Raspberry Pi y no permite la instalación de sistemas operativos adicionales. Es por esto que se pierden las capacidades normales de una Raspberry pi, tales como la conexión remota a través de SSH, pero se ganan beneficios importantes para este trabajo. En la siguiente sección se detallan los beneficios de Meteobridge.

2.3.3. Meteobridge

El sistema operativo Meteobridge habilita la integración de distintas estaciones meteorológicas en un solo dispositivo, dentro de estas se encuentra la Envoy8x. Meteobridge permite que la Raspberry Pi reciba los datos capturados por la Envoy8x mediante conexión USB sin necesidad de estar monitoreando a través de una pantalla. Se puede acceder a los comandos del sistema Meteobridge vía WiFi o LAN Ethernet, permitiendo que todo tipo de configuración o monitoreo sea realizado a través de un browser web remoto. Esto presenta una gran ventaja comparada al programa *WDTU* que viene incluido con la Envoy8x, ya que no es necesario estar físicamente al lado de la Envoy8x para configurar el intervalo de medición de los datos o acceder a cualquier tipo de información. El usuario puede definir el intervalo de actualización de la data obtenida por Meteobridge, este puede ser desde un segundo a una hora. Una vez definido el intervalo, el sistema opera de manera autónoma, aunque el usuario no este conectado al sistema Meteobridge.

Network | Weather Station | Weather Network | Services | System | License | Live Data

Weather Network Status

Weather Underground: 2016-05-13 08:52:01 Success!

Live Data

Sensor	Signal	Metric Data	Imperial Data
Outdoor	44 sec	30.3°C 19% (dew 4.1°C, heat 30.3°C)	86.5°F 19% (dew 39.4°F, heat 86.5°F)
Wind	4 sec	0.0m/s (avg 0.0m/s) 60° ENE	0.0mph (avg 0.0mph) 60° ENE
Rain	36 sec	rate 0.0mm/h	rate 0.00in/h
Indoor	48 sec	26.1°C 26% 1013.2hPa (970.0hPa)	79.0°F 26% 29.92inHg (28.64inHg)

Historical Data

Sensor	Now 08:52	Today Fri 13		Yesterday Thu 12		Month May		Year 2016		All Erase	
		min	max	min	max	min	max	min	max	min	max
Indoor temp	26.1°C 79.0°F	23.9°C 75.0°F	26.3°C 79.3°F	24.3°C 75.7°F	27.0°C 80.6°F	23.3°C 73.9°F	28.4°C 83.1°F	20.8°C 69.4°F	29.1°C 84.4°F	20.2°C 68.4°F	29.1°C 84.4°F
Indoor hum	26%	26%	28%	27%	31%	24%	32%	17%	35%	17%	35%
Indoor dew	5.0°C 41.0°F	3.0°C 37.4°F	5.0°C 41.0°F	4.0°C 39.2°F	8.0°C 46.4°F	2.0°C 35.6°F	9.0°C 48.2°F	-1.0°C 30.2°F	9.0°C 48.2°F	-1.0°C 30.2°F	9.0°C 48.2°F
Indoor press	970.0mb 28.64inHg	967.4mb 28.57inHg	970.0mb 28.64inHg	966.5mb 28.54inHg	971.1mb 28.68inHg	958.7mb 28.31inHg	971.3mb 28.68inHg	955.9mb 28.23inHg	982.7mb 29.02inHg	955.9mb 28.23inHg	982.7mb 29.02inHg
Indoor seapress	1013.2mb 29.92inHg	1010.6mb 29.84inHg	1013.2mb 29.92inHg	1009.7mb 29.82inHg	1014.7mb 29.96inHg	1001.7mb 29.58inHg	1016.0mb 30.00inHg	999.7mb 29.52inHg	1030.3mb 30.42inHg	999.7mb 29.52inHg	1030.3mb 30.42inHg
Outdoor temp	30.3°C 86.5°F	18.1°C 64.6°F	30.3°C 86.5°F	18.4°C 65.1°F	40.3°C 104.5°F	12.3°C 54.1°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F
Outdoor hum	19%	16%	49%	5%	45%	5%	91%	4%	96%	4%	96%
Outdoor dew	4.1°C 39.4°F	-1.1°C 30.0°F	8.5°C 47.3°F	-7.1°C 19.2°F	7.7°C 45.9°F	-7.1°C 19.2°F	13.9°C 57.0°F	-17.0°C 1.4°F	13.9°C 57.0°F	-17.0°C 1.4°F	13.9°C 57.0°F
Outdoor heatindex	30.3°C 86.5°F	18.1°C 64.6°F	30.3°C 86.5°F	18.4°C 65.1°F	40.3°C 104.5°F	12.3°C 54.1°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F
Wind wind	0.0m/s 0.0mph	0.0m/s 0.0mph	3.1m/s 6.9mph	0.0m/s 0.0mph	3.6m/s 8.1mph	0.0m/s 0.0mph	9.8m/s 22.0mph	0.0m/s 0.0mph	12.1m/s 27.1mph	0.0m/s 0.0mph	12.1m/s 27.1mph
Wind avgwind	0.0m/s 0.0mph	0.0m/s 0.0mph	0.9m/s 2.0mph	0.0m/s 0.0mph	0.9m/s 2.0mph	0.0m/s 0.0mph	3.6m/s 8.1mph	0.0m/s 0.0mph	4.0m/s 8.9mph	0.0m/s 0.0mph	4.0m/s 8.9mph
Wind chill	30.3°C 86.5°F	18.1°C 64.6°F	30.3°C 86.5°F	18.4°C 65.1°F	40.3°C 104.5°F	12.3°C 54.1°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F	-2.2°C 28.0°F	40.3°C 104.5°F
Rain total		0.0mm / 0.00in		0.0mm / 0.00in		4.5mm / 0.18in		31.2mm / 1.23in		34.5mm / 1.36in	
Rain rate	0.0mm/h 0.00in/h	0.0mm/h 0.00in/h	0.0mm/h 0.00in/h	0.0mm/h 0.00in/h	0.0mm/h 0.00in/h	0.0mm/h 0.00in/h	80.3mm/h 3.16in/h	0.0mm/h 0.00in/h	1828.8mm/h 72.00in/h	0.0mm/h 0.00in/h	1828.8mm/h 72.00in/h
UV index		11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi	11.3uvi

Figura 2.9: Sistema operativo Meteobridge para la Raspberry Pi accesible en la web.

2.3.4. Webserver LAMP: Linux-APACHE-MySQL-PHP

Para visualizar los datos adquiridos por la Envoy8x que el sistema Meteobridge almacena periódicamente en una base de datos, se necesita desarrollar un servidor de base de datos y una página web.

Una página web requiere de múltiples procesos trabajando en conjunto. Uno de los conjuntos más utilizados es el "LAMP Stack", donde cada letra representa uno de los procesos que forman el servidor completo.

El primero de estos es el sistema operativo Linux. Generalmente se programa dentro de un computador con este tipo de sistema operativo, ya que ofrece mayor flexibilidad y opciones de configuración, sin embargo, actualmente es posible crear un servidor en cualquier tipo de sistema operativo.

Segundo, se requiere de un servidor web que sea capaz de procesar solicitudes y entregar recursos web vía HTTP para que la aplicación esté disponible para cualquier usuario en el dominio publico a través de una dirección web simple. Para lograr esto se eligió el servidor web *Apache*. Este servidor web está mantenido por una comunidad abierta, posee grandes cantidades de configuraciones y actualmente es utilizado por una gran porción de páginas web.

Una de las partes más importantes en la adquisición de datos viene siendo la base de datos. *mySQL*permite almacenar toda la información deseada en un formato que es fácil de consultar con el lenguaje *SQL*. *SQL* es ideal para crear tablas estructuradas en el servidor y es recomendada para bases de datos complejas o en grande escala.

Finalmente, es necesario poder comunicarse con el servidor creado, especialmente para el usuario final. El lenguaje *PHP* permite lograr una comunicación dinámica con el servidor, facilitando la entrega y recepción de información. Esto es clave en el procesamiento de los datos almacenados, ya que no siempre es necesario ver toda la información disponible y limitando la información solicitada ayuda en mantener la página web funcionando de manera rápida.

Figura 2.10: Software para la configuración de la pendrive USB.

Normalmente se utiliza la Raspberry Pi como servidor junto a las funciones que uno requiera, sin embargo, en este caso no es posible utilizar la memoria de la Raspberry Pi para tareas adicionales ya que sus recursos están completamente asignados al sistema operativo Meteobridge. Debido a esto, se opto por utilizar un servidor "móvil" llamado *USBWebserver*. Este programa permite transformar cualquier pendrive USB en un servidor con "LAMP Stack" incluida. Luego, la base de datos del *USBWebserver* puede comunicarse con Meteobridge mediante el internet. De esta manera es posible desplazar el servidor y la base de datos e implementarlo en cualquier computador con conexión al internet en el momento, sin necedidad de tener la Raspberry Pi con Meteobridge conectada a la misma red del *USBWebserver*. Esto es especialmente útil durante la creación y modificación de la página web final.

2.3.5. Cliente Web: Navegador

El usuario debe ser capaz de interactuar con las mediciones obtenidas por las estaciones meteorológicas de manera intuitiva y robusta. El cliente web diseñado debe tener una interfaz que sea fácil de entender para el usuario, permitiendo ingresar las fechas deseadas de las mediciones elegidas y descargar partes de la base de datos que sean de importancia. En la figura 2.11 se puede apreciar un ejemplo de interfaz para el usuario desarrollada por Wunderground.

Figura 2.11: Ejemplo de la interfaz de usuario de la página web Wunderground.

Capítulo 3

Configuración de la Red de Estaciones Meteorológicas

3.1. Preparación de las estaciones Vantage Pro 2

Se requiere armar y configurar las 4 estaciones Davis Vantage Pro 2 antes de que se puedan realizar las pruebas prácticas. Para esto, se debe armar cada estación de la manera indicada por los manuales y asegurarse que los componentes incluidos funcionan como se espera. Una vez armadas las estaciones se deben configurar las consolas receptoras de cada estación para que reciban los datos de la estación ISS correspondiente y verificar que no exista interferencia entre ellas. A continuación, se detallan los pasos importantes para lograr la configuración correcta de dichas estaciones y sus consolas receptoras.

3.1.1. Preparación de la estación transmisora (ISS)

La estación transmisora ISS o "Integrated Sensor Suite" requiere ser armada y configurada con su propio canal de transmisión en caso de existir más de una en un ambiente. Se empieza montando el brazo que soporta el anemómetro y la veleta de viento, asegurándose de que la curva del brazo apunte alejándose de la estación. Este paso es de mayor importancia, ya que el brazo en esta configuración debería ir apuntado hacía el NORTE para lograr la medición correcta de la dirección del viento. En la figura 3.1 se muestra como debería quedar armado.

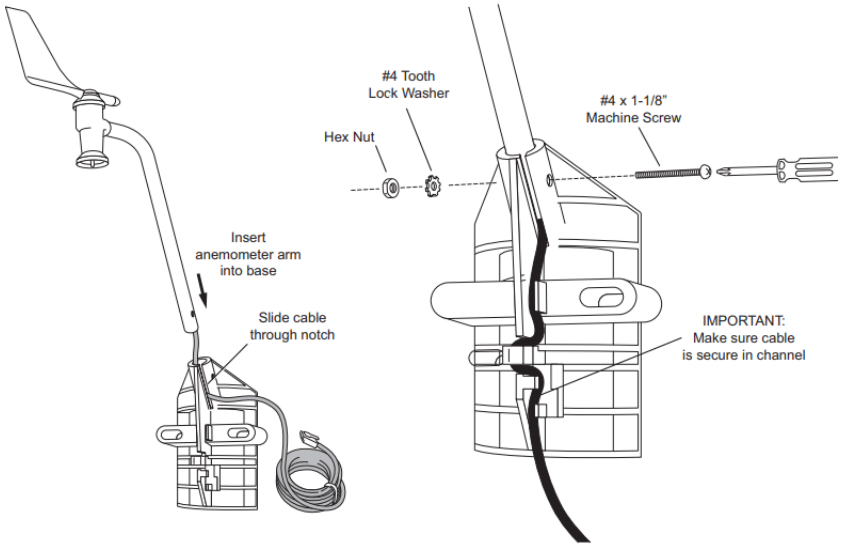


Figura 3.1: Orientación del soporte para el anemómetro y veleta de viento.

Una vez armado el soporte para las mediciones del viento, se procede a configurar el sensor de medición de precipitación. Éste utiliza un imán que está diseñado para capturar mediciones en pulgadas o "inches" y requiere ser modificado para lograr capturar datos en milímetros. Esto se logra colocando una carcasa (incluida) alrededor del imán y fijándolo en la orientación indicada de vuelta en el sensor. El proceso se encuentra ilustrado en la figura 3.2 y 3.3. Luego, es necesario que la base esté perfectamente horizontal para que el dispositivo de cazoletas basculantes junto al imán calculen la cantidad de precipitación actual.

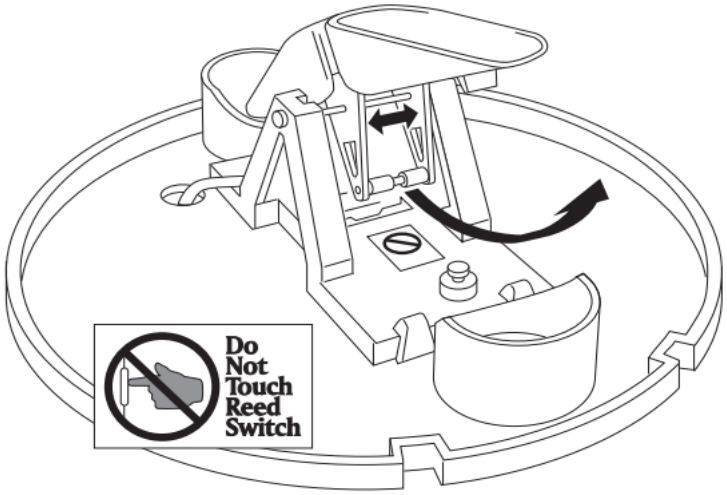


Figura 3.2: Extracción del imán del sensor de lluvia.

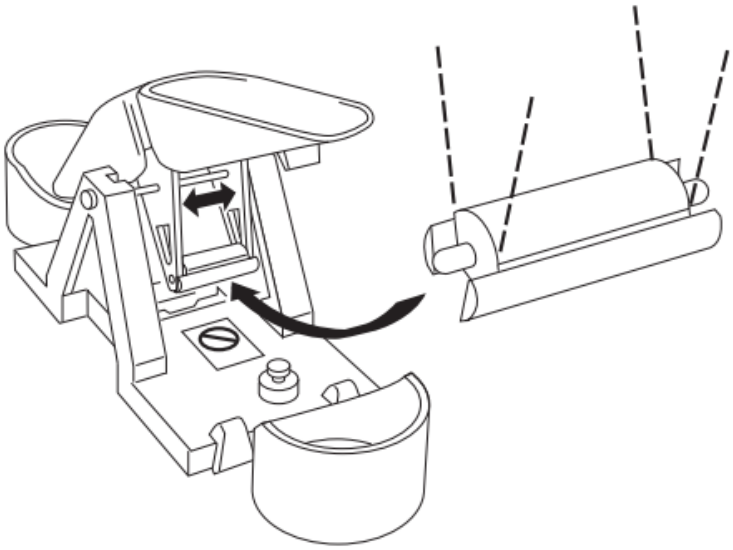


Figura 3.3: Reemplazo del imán del sensor de lluvia.

Luego se deben conectar los cables de los sensores en la caja transmisora. Para acceder el panel de configuración se debe levantar y deslizar hacia arriba la tapa que contiene el panel solar que alimenta la ISS, figura 3.4, teniendo cuidado a no dañar el cable del panel solar que se encuentra en su interior. Es recomendable desconectar el cable como se indica en la figura 3.5.

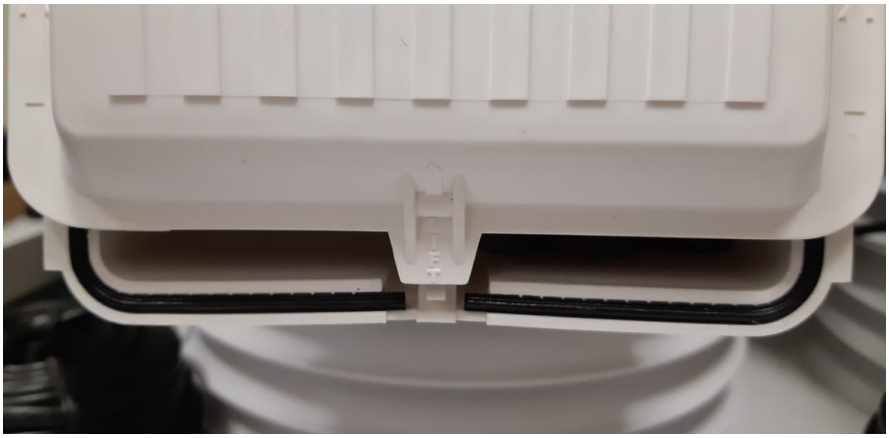


Figura 3.4: Levantar la tapa y deslizar hacía arriba.

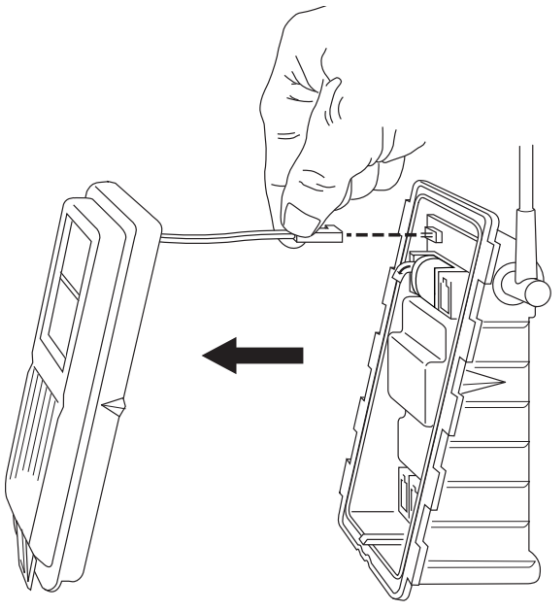


Figura 3.5: Desconectando el cable del panel solar.

Los cables de los sensores deben ser conectados por el lado inferior y sellados con la goma incluida para prevenir que entre lluvia e insectos. Esto se puede apreciar en la figura 3.6

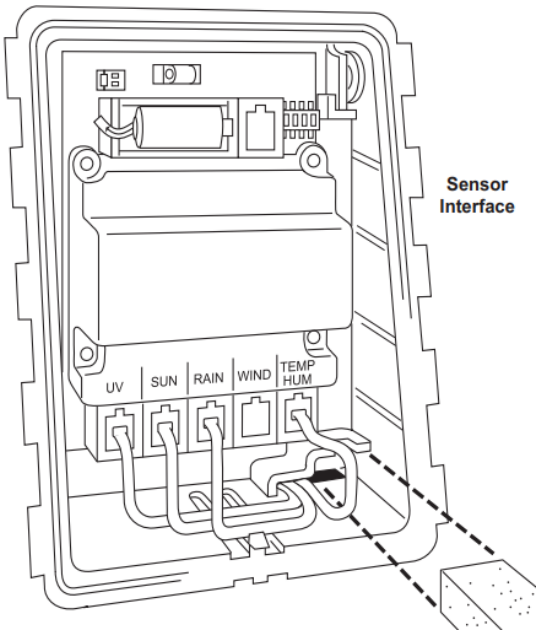


Figura 3.6: Conexión de los sensores.

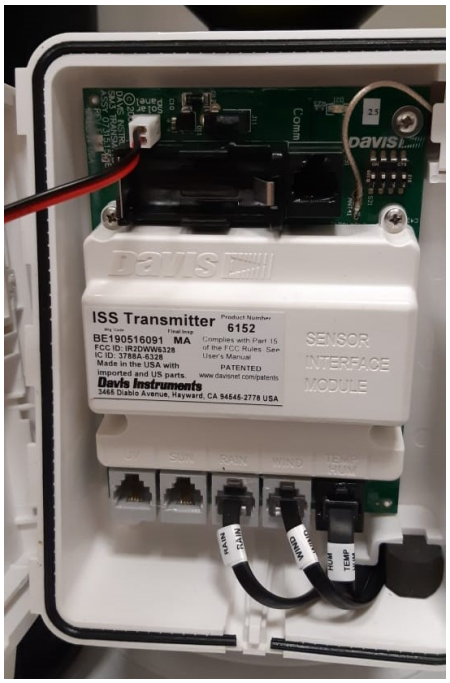


Figura 3.7: Cableado interior de una estación ISS.

En el interior de la caja transmisora se pueden ubicar cuatro DIP switches que mediante su configuración se puede definir un canal de transmisión para dicha estación. Los DIP switches pueden ser ubicados en el numero 7 de la figura 3.8. Los tres primeros switches determinan el canal de la estación, mientras que el cuarto switch solo se utiliza para colocar la estación en modo "prueba". Los primeros tres switches utilizan una configuración binaria y pueden generar los canales indicados en la figura 3.9. 3.9.

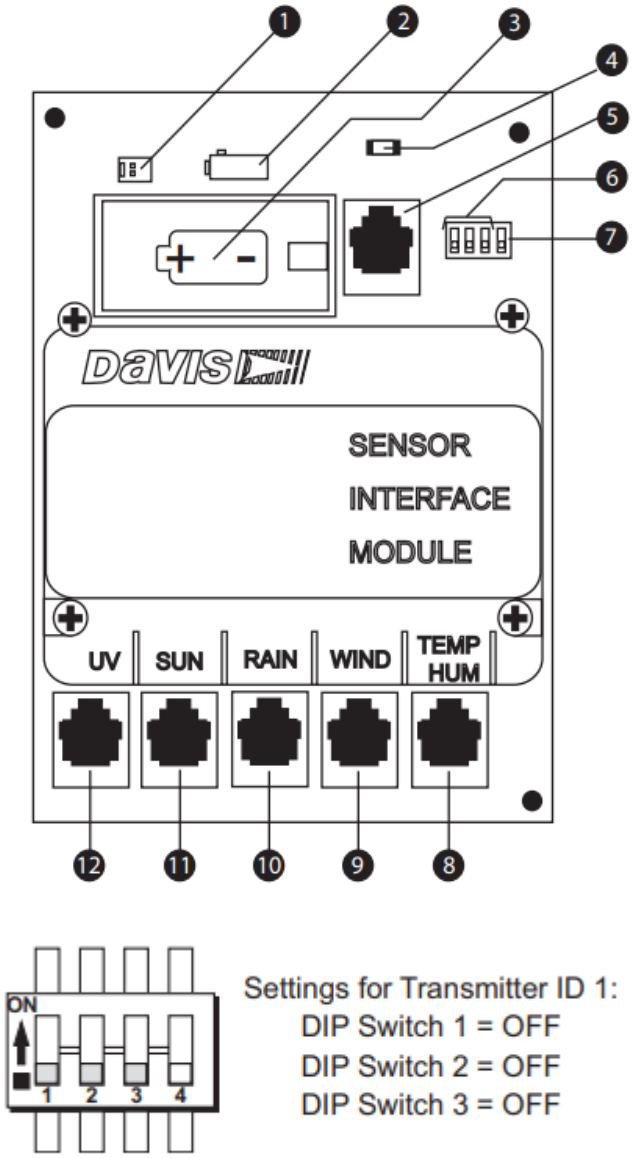


Figura 3.8: Esquemática de la caja transmisora y DIP switches.

Con estas configuraciones, se les asigna un canal independiente a las cuatro estaciones armadas. En la figura 3.10 se muestra como quedó configurada la estación número 3 con los DIP switches en posición "0100". Debido a que esta asignación es configurada con switches físicos, no es necesario reconfigurar al ser des-energizadas.

ID Code	Switch 1	Switch 2	Switch 3
#1 (default)	off	off	off
#2	off	off	ON
#3	off	ON	off
#4	off	ON	ON
#5	ON	off	off
#6	ON	off	ON
#7	ON	ON	off
#8	ON	ON	ON

Figura 3.9: Configuraciones posibles de los DIP switches.

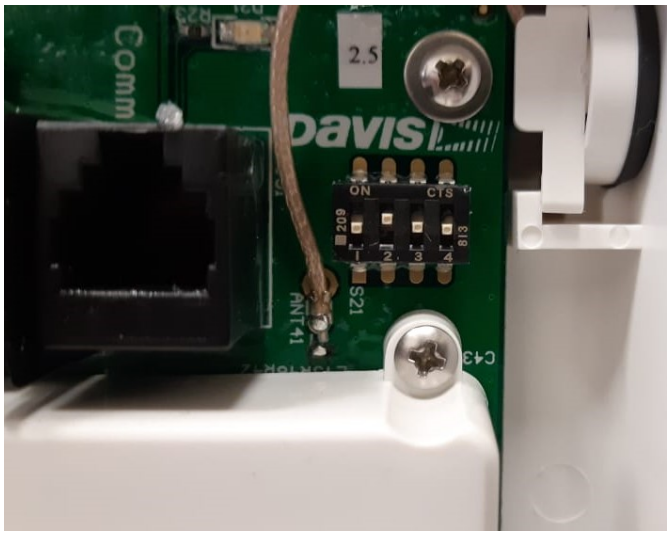


Figura 3.10: Configuración de DIP switches para la estación número 3.

Finalmente, las estaciones quedan armadas como se indica en la ilustración de la figura 3.11 y se puede comenzar el trabajo de configurar sus pantallas receptoras correspondientes.

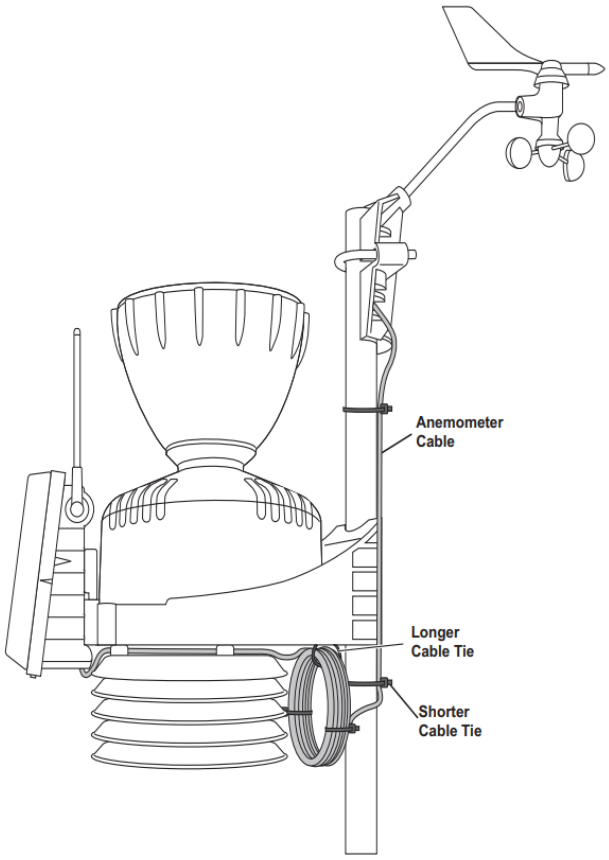


Figura 3.11: Ilustración de la estación ISS armada.

Figura 3.12: Estaciones armadas en el Laboratorio Shannon.

3.1.2. Preparación de la consola receptora

Una vez armadas las estaciones ISS, se deben configurar sus consolas receptoras. Para empezar, se debe tener cuidado con las antenas de los receptores, ya que no giran en 360 grados y forzándolas podría causarles daño. Por lo tanto, solo se deben girar en la dirección indicada en la figura 3.13.

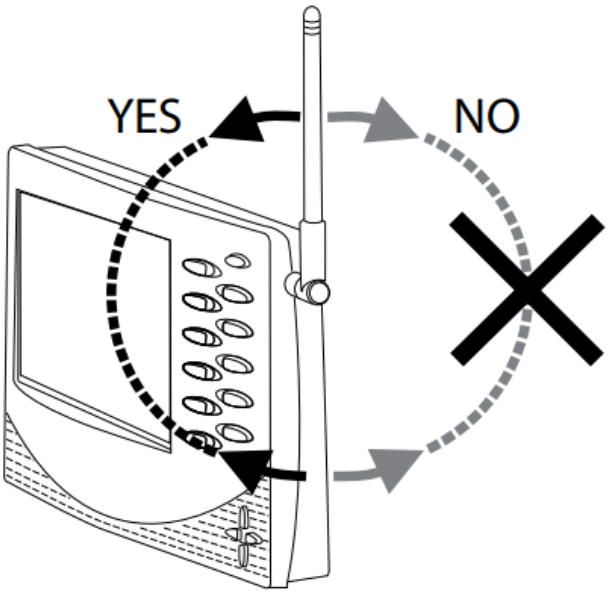


Figura 3.13: Dirección de giro de la antena.

Al energizar las consolas, utilizando el adaptador incluido para conectarlas a la red eléctrica o 3 baterías tipo C, comienzan a recibir datos de la estación en el canal uno por defecto (3.14). Debido a que las pantallas se des-configuran cada vez que se des-energicen, se deben cambiar los canales preestablecidos. Para lograr esto se utilizan los botones con flechas para cambiar entre canales y los botones "+" y "-" para prender o apagar el canal que sale en la pantalla. Esto se puede verificar por los números indicados a la derecha de "Station No." en la pantalla (ver figura 3.15). Si se reciben datos de más de una estación, es posible cambiar cual estación se muestra en la pantalla.

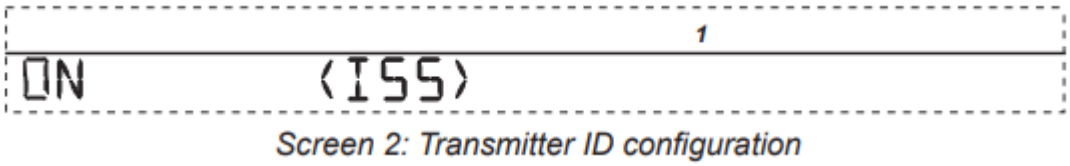


Figura 3.14: Canal por defecto de recepción.

Figura 3.15: Indicación de canales recibidos.

Siguiendo las instrucciones en pantalla se configuraron los receptores para tener la hora y fecha correcta al igual que la longitud y latitud deseada. Se fija la longitud en -71.5 y latitud en 33.1 correspondiente a Valparaíso ya que solo tenia un decimal de precisión. Luego se fijó la elevación en 101 metros de altura y se cambiaron las unidades de medición que por defecto estaban en las unidades utilizadas en EEUU como se indica en la figura 3.16.

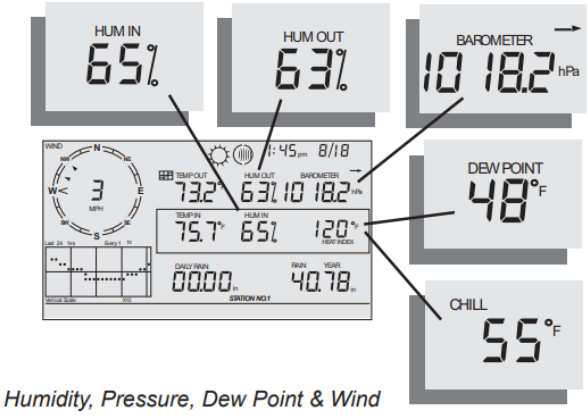


Figura 3.16: Unidades de medición por defecto.

3.2. Prueba de comunicación inalámbrica de la red de estaciones con las consolas Vantage Pro 2

A continuación, se muestran dos consolas configuradas por separado y ambas juntas verificando que están recibiendo datos de dos estaciones distintas. Esto es más fácil ver en la dirección del viento, ya que están orientadas en distintas direcciones. Las figuras 3.17, 3.18 y 3.19 muestran que las consolas Vantage Pro 2 están recibiendo datos de las ISS respectivas.

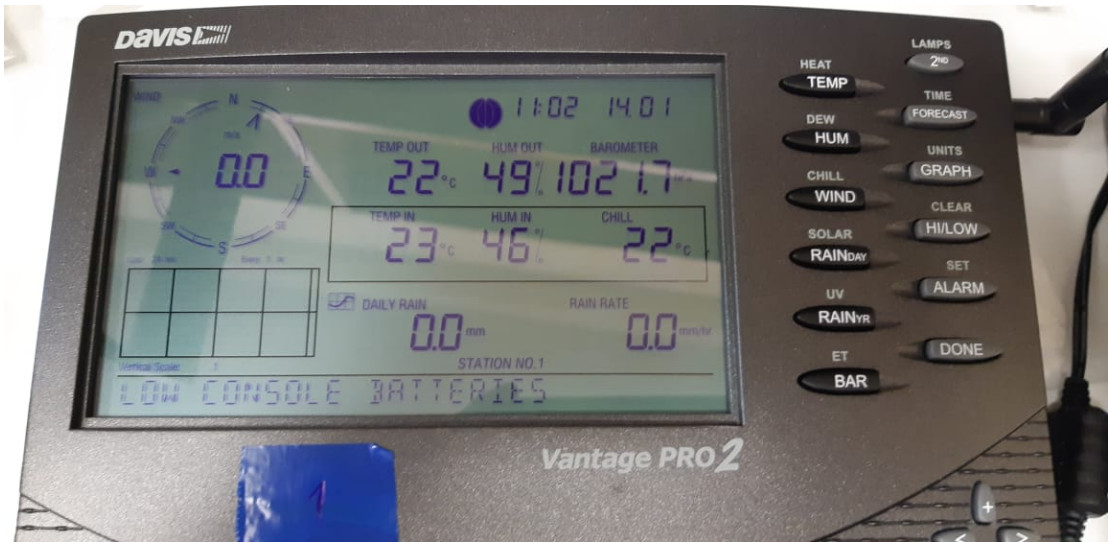


Figura 3.17: Consola receptora 1.

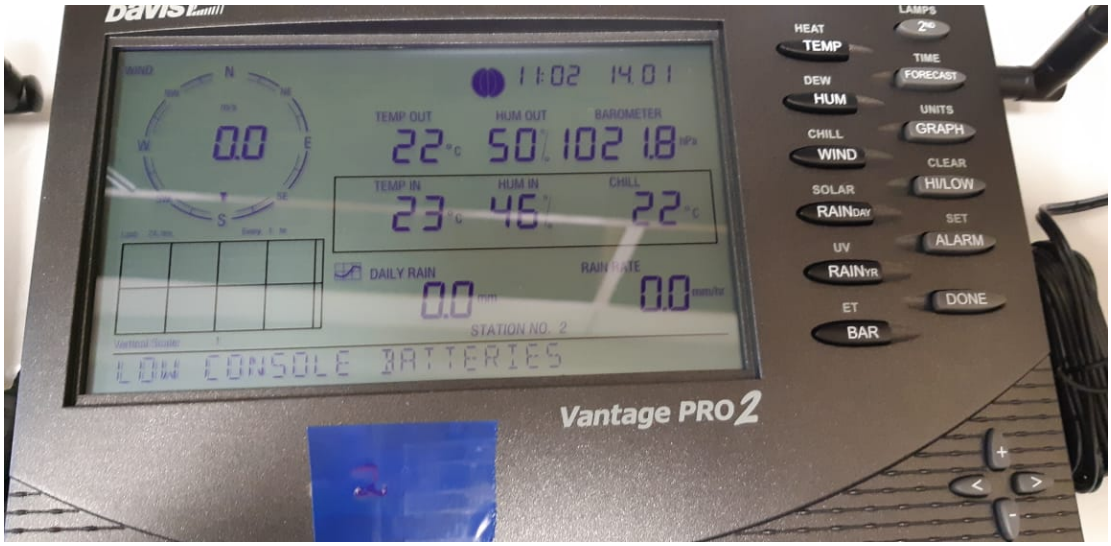


Figura 3.18: Consola receptora 2.

Figura 3.19: Consolas recibiendo distintas direcciones de viento, sin interferencia.

3.3. Conexión de las consolas Vantage Pro 2 al concentrador Envoy8x

Una vez configuradas las consolas receptoras con un canal único, se procede a utilizar el software *WDTU* para lograr el enlace inalámbrico entre el concentrador Envoy8x y cada estación Vantage Pro 2. En el capítulo 4 se detalla el procedimiento para preparar el concentrador Envoy8x y configurar la recepción de datos de cada estación ISS.

Al finalizar la configuración, las estaciones ISS quedan transmitiendo directamente al Envoy8x. Las consolas receptoras no son necesarias para recibir las mediciones, sin embargo, sirven como pantallas adicionales para monitorear la información localmente. La Envoy8x sustituye a las consolas permitiendo concentrar los datos de las 4 ISS.

Capítulo 4

Adquisición de datos con la Envoy8x

4.1. Preparación de la Envoy8x

4.1.1. Configuración de hardware

Antes de utilizar el software incluido con la Envoy8x para configurar sus parámetros a través de un PC, se debe conectar el cable "Weatherlink USB". En el interior de la Envoy8x se puede apreciar un espacio diseñado para conectar el cable. Las siguientes figuras muestran ambos dispositivos y como debe quedar la Envoy8x una vez se instala el cable.

Figura 4.1: Interior de la Envoy8x, sin cable weatherlink USB incluido.

Figura 4.2: Cable Weatherlink USB.

Figura 4.3: Interior de la Envoy8x, con cable weatherlink USB instalado.

Figura 4.4: Envoy8x lista para ser configurada a través de su software.

4.1.2. Configuración de software

Para lograr la comunicación entre la Envoy8x y la Raspberry Pi, primero se debe configurar la Envoy8x usando su software incluida *Weather Data Transfer Utility*, o WDTU, junto a un PC. Con este software se define la Envoy8x como el receptor número uno y a las cuatro estaciones ISS como transmisoras distintas,

asegurándose de definir sus unidades deseadas. En la figura 4.6 se puede apreciar el programa WDTU mostrando la configuración final de los dispositivos y en la figura 4.7 se puede verificar que el receptor esta recibiendo señales desde las cuatro estaciones, donde “ACQUIRED” significa “adquirida”.

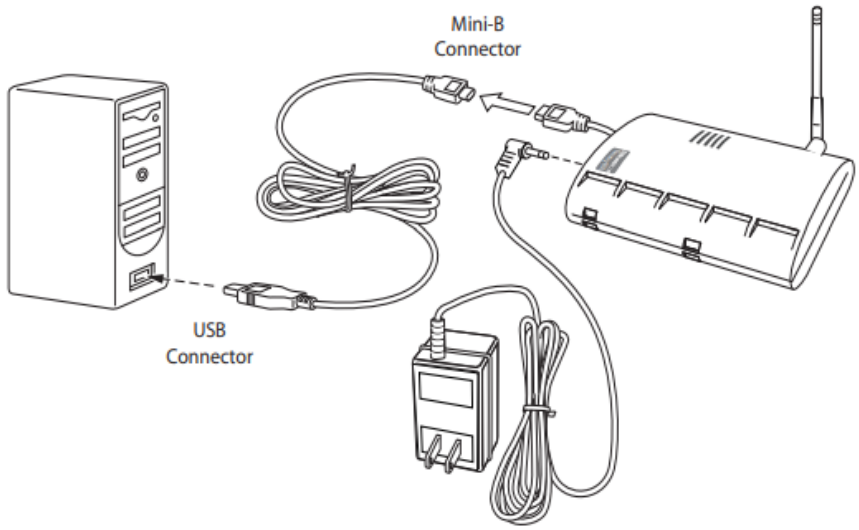
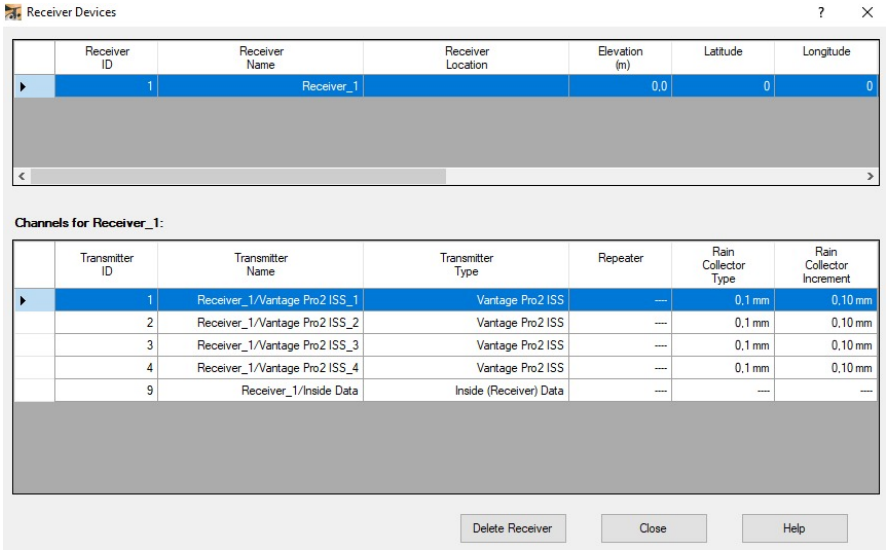


Figura 4.5: Esquema de conexión de la Envoy8x a un PC.



Receiver Devices

	Receiver ID	Receiver Name	Receiver Location	Elevation (m)	Latitude	Longitude
▸	1	Receiver_1		0,0	0	0

Channels for Receiver_1:

	Transmitter ID	Transmitter Name	Transmitter Type	Repeater	Rain Collector Type	Rain Collector Increment
▸	1	Receiver_1/Vantage Pro2 ISS_1	Vantage Pro2 ISS	---	0,1 mm	0,10 mm
	2	Receiver_1/Vantage Pro2 ISS_2	Vantage Pro2 ISS	---	0,1 mm	0,10 mm
	3	Receiver_1/Vantage Pro2 ISS_3	Vantage Pro2 ISS	---	0,1 mm	0,10 mm
	4	Receiver_1/Vantage Pro2 ISS_4	Vantage Pro2 ISS	---	0,1 mm	0,10 mm
	9	Receiver_1/Inside Data	Inside (Receiver) Data	---	---	---

Delete Receiver | Close | Help

Figura 4.6: Configuración final de la Envoy8x con las cuatro estaciones ISS.

```
Transmitter Status for Receiver_1

            Date: lun, 08-02-2021
      Checked At: 11:49:04
   Receiver Date: lun, 08-02-2021
   Receiver Time: 11:48:58
Receiver Battery: 4,18 V

TX 1: Receiver 1/Vantage Pro2 ISS 1
            Status: ACQUIRED
   Total Resynchs:      1
   Total Received:    114
     Total Misses:    600
          Percent:    16%
       Max Streak:    114
   Current Streak:    114
   Max Save After:      0
        Hop Index:     43
             RSSI:     59
              AFC:      0
       AFC Adjust:      1

TX 2: Receiver 1/Vantage Pro2 ISS 2
            Status: ACQUIRED
   Total Resynchs:      0
   Total Received:    158
     Total Misses:      0
          Percent:   100%
       Max Streak:    158
   Current Streak:    158
   Max Save After:      0
        Hop Index:     36
             RSSI:     59
              AFC:      1
       AFC Adjust:      0

TX 3: Receiver 1/Vantage Pro2 ISS 3
            Status: ACQUIRED
   Total Resynchs:      0
   Total Received:     21
     Total Misses:    240
          Percent:     8%
       Max Streak:     21
   Current Streak:     21
   Max Save After:      0
        Hop Index:      2
             RSSI:     59
              AFC:      0
       AFC Adjust:      0

TX 4: Receiver 1/Vantage Pro2 ISS 4
            Status: ACQUIRED
   Total Resynchs:      0
   Total Received:    153
     Total Misses:      5
          Percent:    97%
       Max Streak:     40
   Current Streak:      0
   Max Save After:      2
        Hop Index:     36
             RSSI:   ----
              AFC:      0
       AFC Adjust:      1
```

Figura 4.7: Estado de conexión de las cuatro estaciones ISS, mostrado en pantalla PC y WDTU.

4.2. Prueba de almacenamiento de datos de múltiples estaciones

A continuación se verifica que la Envoy8x esta recibiendo datos desde las cuatro estaciones ISS. Desde el software *WDTU* se puede apreciar que se reciben múltiples lecturas de temperatura para las distintas estaciones ISS. En la figura 4.8 se muestran mediciones para las estaciones 2, 3 y 4, identificadas por TX2, TX3 y TX4, respectivamente. La estación 1 no aparece en la figura debido al tamaño máximo de la pantalla. Las múltiples mediciones indica que efectivamente se están almacenando datos dentro de la Envoy8x cada un minuto. Es posible configurar el intervalo de tiempo, desde 10 segundos a una hora.

File Setup View Database Help

Receiver Devices:

Receiver_1

Transmitters:

1: Receiver_1/Vantage Pro2 ISS_1
2: Receiver_1/Vantage Pro2 ISS_2
3: Receiver_1/Vantage Pro2 ISS_3
4: Receiver_1/Vantage Pro2 ISS_4

From: lun. , 08-02-2021
To: lun. , 08-02-2021

Include all dates
Include all transmitters

Select Data Fields | Export to Excel | Refresh | Delete Records

Date	Time	iside erature 2 (°C)	Outside Humidity TX2 (%)	Wind Speed TX2 (m/s)	Rain Amount TX2 (mm)	Outside Temperature TX3 (°C)	Outside Humidity TX3 (%)	Wind Speed TX3 (m/s)	Rain Amount TX3 (mm)	Outside Temperature TX4 (°C)	Outside Humidity TX4 (%)	Wind Speed TX4 (m/s)
lun, 08-02-2021	11:42:00	20,9	---	---	0,0	---	---	---	0,0	---	---	---
lun, 08-02-2021	11:43:00	20,9	63	0,0	0,0	---	---	---	0,0	20,9	---	0,0
lun, 08-02-2021	11:44:00	20,9	63	0,0	0,0	---	---	0,0	0,0	21,0	63	0,0
lun, 08-02-2021	11:45:00	20,9	63	0,0	0,0	---	---	---	0,0	21,0	63	0,0
lun, 08-02-2021	11:46:00	20,9	63	0,0	0,0	---	---	---	0,0	21,0	63	0,0
lun, 08-02-2021	11:47:00	21,0	63	0,0	0,0	---	---	---	0,0	21,0	63	0,0
lun, 08-02-2021	11:48:00	20,9	63	0,0	0,0	---	---	---	0,0	21,1	63	0,0
lun, 08-02-2021	11:49:00	21,0	63	0,0	0,0	---	---	---	0,0	21,1	63	0,0
lun, 08-02-2021	11:50:00	21,0	63	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:51:00	21,0	63	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:52:00	21,0	63	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:53:00	21,1	63	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:54:00	21,0	64	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:55:00	21,0	64	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:56:00	21,0	64	0,0	0,0	21,5	61	0,0	0,0	21,1	63	0,0
lun, 08-02-2021	11:57:00	21,1	64	0,0	0,0	21,5	62	0,0	0,0	21,2	63	0,0
lun, 08-02-2021	11:58:00	21,0	64	0,0	0,0	21,5	62	0,0	0,0	21,2	63	0,0

Figura 4.8: Recepción y almacenamiento de datos en la Envoy8x vista desde un PC y WDTU cada un minuto.

4.3. Conexión de la Envoy8x a la Raspberry Pi

Con la configuración del software finalizada, la Envoy8x queda lista para almacenar datos y ser conectada a la Raspberry Pi. Utilizando el mismo cableado que se muestra en la figura 4.5, se reemplaza el PC por la Raspberry Pi y queda conectada la Envoy8x a través de un adaptador Mini USB a USB como se indica en la figura 4.9.

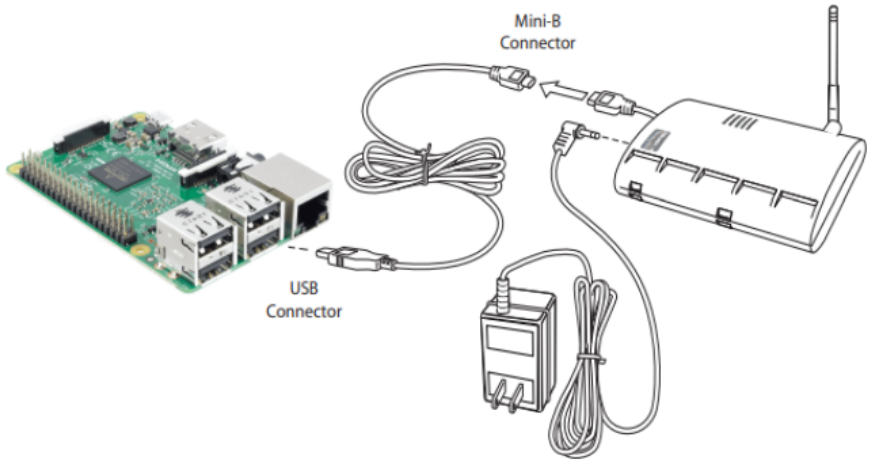


Figura 4.9: Esquema de conexión de la Envoy8x a la Raspberry Pi.

Capítulo 5

Adquisición de datos en la Raspberry Pi

5.1. Instalación del sistema operativo Meteobridge

Meteobridge es el sistema operativo para la Raspberry Pi que permite trabajar con los datos obtenidos y facilita el acceso remoto de estos. Para instalar Meteobridge se debe primero montar la imagen a una tarjeta micro SD que será insertada a la Raspberry Pi. La imagen de Meteobridge se puede descargar desde la página web principal del desarrollador [2] y utilizando un programa como *Etcher*, figura 5.1, se puede montar la imagen a la micro SD. Finalmente, se inserta la tarjeta dentro de la Raspberry Pi, se conecta a la red deseada mediante un cable ethernet y se energiza. La instalación finaliza una vez la LED roja de la Raspberry Pi termina de parpadear.

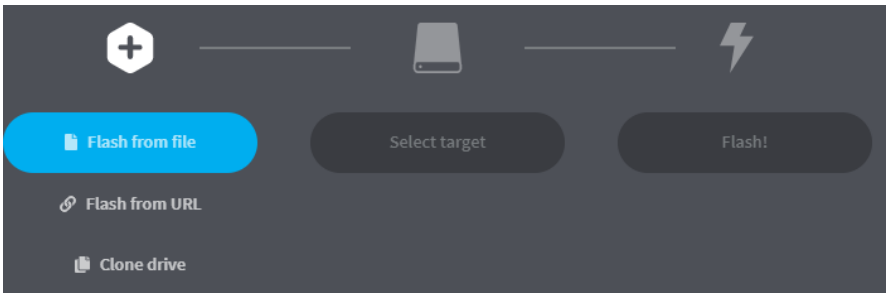


Figura 5.1: Interfaz del programa Etcher para montar la imagen a la micro SD.

Figura 5.2: Instalación de sistema operativo Meteobridge a través de tarjeta micro SD .

5.2. Configuración de Meteobridge

5.2.1. Asignación de IP para la Rasberry Pi mediante LAN/Ethernet

Es importante tener a la Raspberry Pi conectada a la red, ya que durante la instalación se le asigna una IP que debe ser utilizada para acceder a la interfaz Meteobridge. Debido a que la Raspberry Pi no tiene pantallas para desplegar información se debe utilizar un *IP scanner* para encontrar la IP que fue asignada durante la instalación. Finalmente, mediante la IP, se puede acceder a la interfaz de Meteobridge mediante cualquier navegador de web.

Figura 5.3: Interfaz de Meteobridge, mostrando la configuración final de la Raspberry Pi.

En la figura 5.3 se puede apreciar la IP local asignada, 10.2.51.30, dentro

de la red de computadores del Departamento de Electrónica de la Universidad Técnica Federico Santa María, Casa Central. Es importante señalar que la IP se debe cambiar una vez que se instale en la sede de Viña del Mar.

5.2.2. Selección de estaciones meteorológicas deseadas

En la interfaz de Meteobridge se especifica las estaciones que el usuario desea conectar a la Raspberry Pi. Para este proyecto, se están utilizando cuatro estaciones Davis Vantage Pro2 conectadas al concentrador Envoy8x. Por lo tanto, se debe seleccionar el modelo del concentrador como la estación conectada en la pestaña *Weather Station* dentro de la interfaz web.

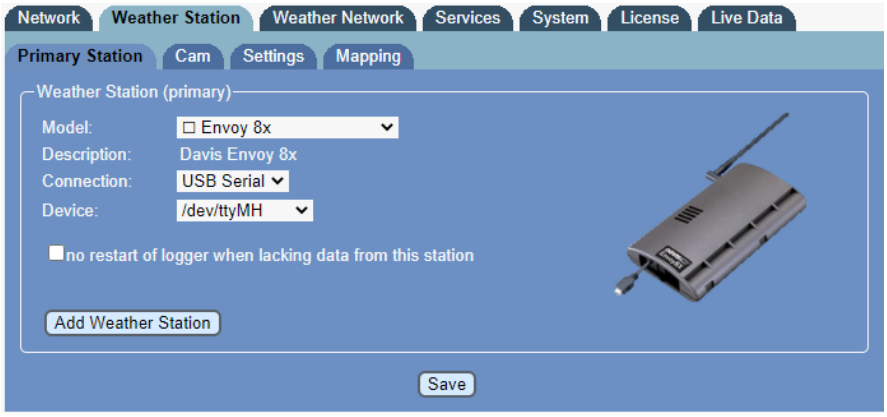


Figura 5.4: Interfaz de Meteobridge, selección de estación/dispositivo meteorológico.

5.2.3. Configuración de exportación de datos a servicios online

La interfaz Meteobridge permite subir mediciones a distintos servicios online, como por ejemplo *Weather Underground*, y a bases de datos *mySQL*. *Weather Underground* o más conocida como *Wunderground* permite a sus usuarios subir datos de sus dispositivos meteorológicos, monitorear y generar gráficos simples. Para lograr esto, basta con crear una cuenta en la página web y registrar el dispositivo en *Mis Dispositivos*, figura 5.5. Wunderground le asigna un *Station ID* que puede ser usado dentro de Meteobridge para subir mediciones en un intervalo de tiempo deseado.

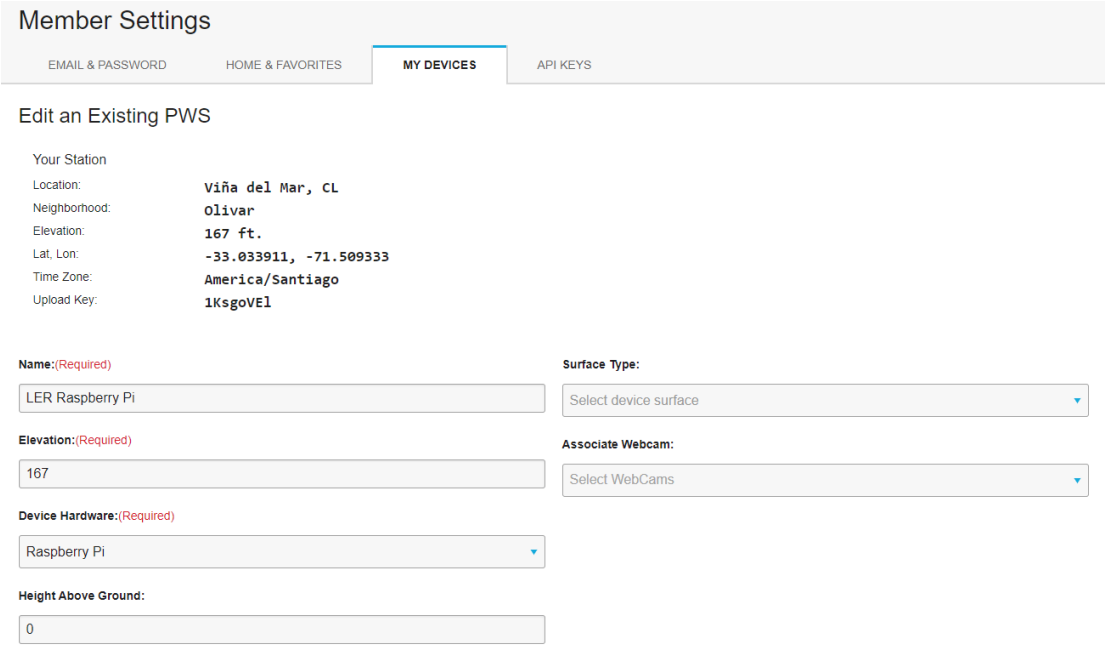


Figura 5.5: Registro de dispositivo en la página web Wunderground.

Con el dispositivo registrado dentro de Wunderground, es posible configurar la Raspberry Pi para transmitir datos a la página. Esto se realiza a través de la pestaña *Weather Network* y seleccionando Weather Underground como el servicio deseado. Se procede a utilizar el "Station ID" que se obtuvo y la clave respectiva. Finalmente, se puede definir el intervalo de subida de los datos. A continuación, se presenta la configuración final del servicio por ejemplo, cada 5 segundos.

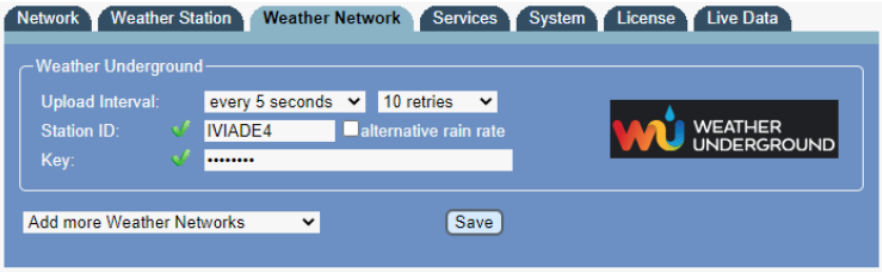


Figura 5.6: Interfaz de Meteobridge, configuración de servicio Wunderground.

Name	Location	Status	ID	Key	Type	Manage
LER Raspberry Pi	Viña del Mar (Olivar), CL	Online	IVIADE4	1KsgoVEl	PWS	Edit | Delete | Copy credentials

Figura 5.7: Estado del dispositivo en la página Wunderground.

5.2.4. Exportación a la base de datos MYSQL

Para tener más control sobre el procesamiento de la data obtenida, se crea una base de datos *mySQL* en el servidor *USBWebserver* (detallado en el capitulo 6) donde Meteobridge podrá transmitir los datos recibidos. Con esta base de datos será posible elegir qué variables graficar, en qué intervalo de tiempo y si se mostrarán múltiples estaciones ISS a la vez. En la pestaña *Services*, dentro de la interfaz de Meteobridge, se procede a definir la conexión a la base de datos (ver figura 5.8) y un evento *mySQL*. Este evento puede ser configurado como el usuario desee, especificando que variables serán exportadas a la base de datos mediante una *Query*. En la figura 5.9 se muestra la configuración final de las cuatro estaciones.

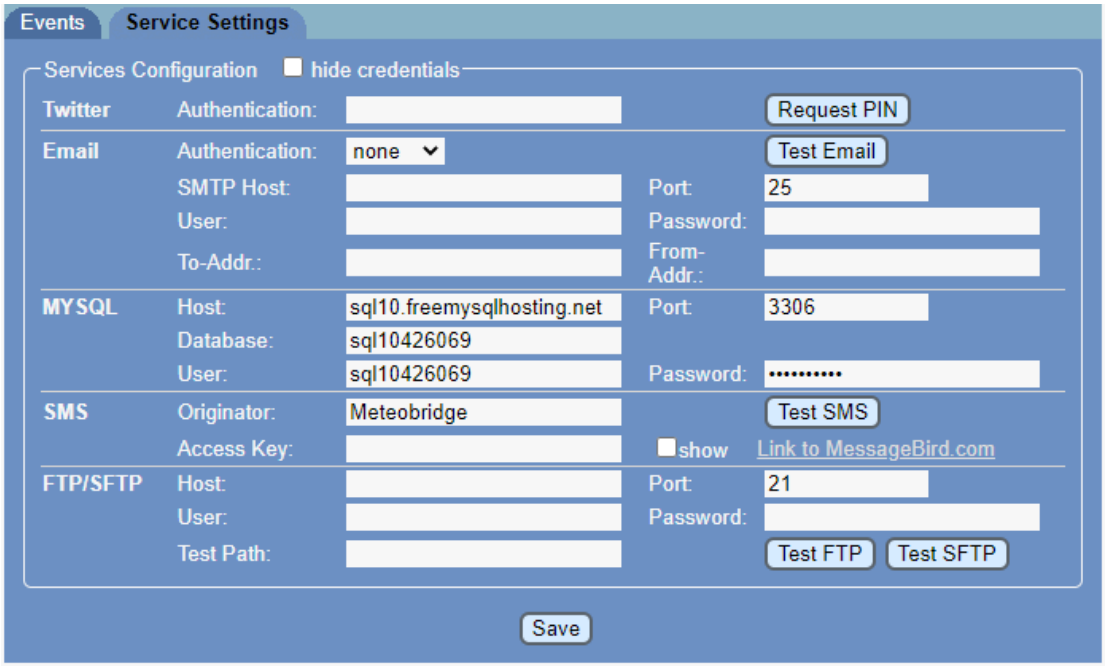


Figura 5.8: Configuración de la conexión a la base de datos.

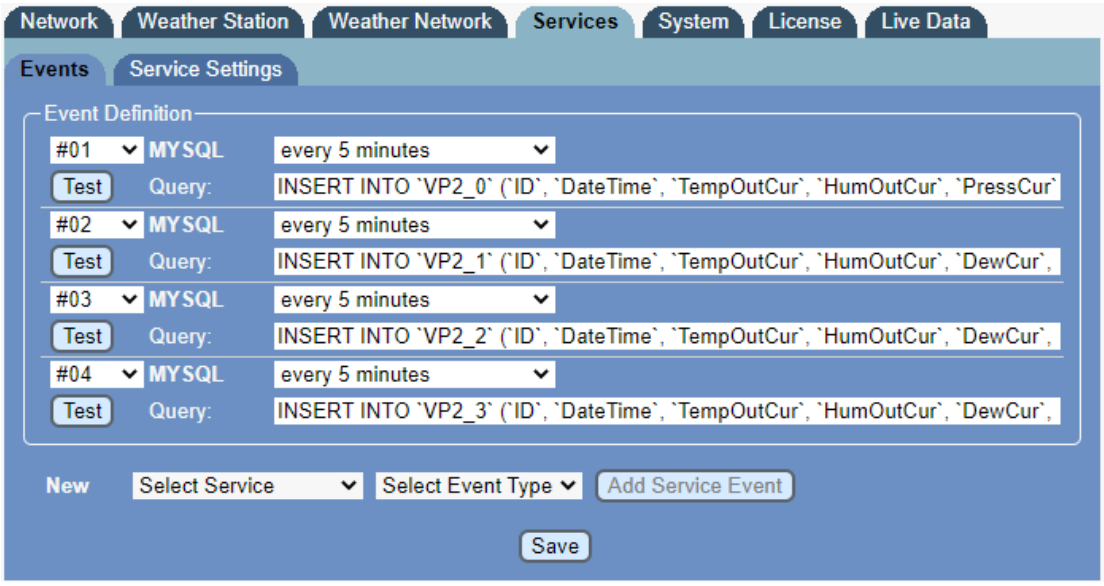


Figura 5.9: Configuración de la exportación a la base de datos.

Para cada estación se crea una *QUERY* o consulta, donde se indica cuales mediciones van dentro de cada columna de la base de datos *mySQL*. Cada estación Vantage Pro 2 tiene una consulta y tabla distinta donde se ingresarán sus datos. Las estaciones tienen sus propias tablas dentro de la base de datos denominadas por "VP2_0", "VP2_1", "VP2_2" y "VP2_3". Luego, las variables que están definidas dentro de cada tabla deben ser mapeadas a una variable interna de cada estación, definiendo la precisión que se desea de cada variable. En el cuadro 5.1 se detallan las consultas de la primera estación.

Variable Base de datos	**Variable Lógica**	**Descripción**
ID	NULL	Identificador
DateTime	[YYYY]-[MM]-[DD] [hh]:[mm]:[ss]	Formato de día/hora
TempOutCur	th0!0temp-act=.2	Temperatura actual
HumOutCur	th0!0hum-act=.3	Humedad actual
PressCur	thb0!0press-act=.3	Presión actual
DewCur	th0!0dew-act=.3	Punto de rocío
HeatIdxCur	th0!0heatindex-act=.3	Índice de calor
WindChillCur	wind0!0chill-act=.3	Factor de viento frío
WindSpeedCur	wind0!0wind-act=.3	Vel. viento actual
WindAvgSpeedCur	wind0!0avgwind-act=.3	Vel. viento promedio
WindDirCur	wind0!0dir-act=.3	Dirección de viento
WindDirCurEng	wind0!0dir-act=endir	Dirección de viento en inglés
WindGust10	wind0!0wind-max10	Máx vel. viento en 10 min
WindDirAvg10	wind0!0dir-avg10	Promedio Dir. viento en 10 min
WindDirAvg10Eng	wind0!0dir-avg10=endir	Promedio Dir. viento en 10 min (inglés)
RainRateCur	rain0!0rate-act=.3	Promedio de lluvia actual
RainDay	rain0!0total-daysum=.3	Total de lluvia en el día
RainYest	rain0!0total-ydaysum=.3	Total de lluvia ayer
RainMonth	rain0!0total-monthsum=.3	Total de lluvia del mes
RainYear	rain0!0total-yearsum=.3	Total de lluvia en un año

Cuadro 5.1: Mapeo de variables para la base de datos mySQL.

Las variables lógicas de las estaciones Vantage Pro 2 pueden ser encontradas en la página web de Meteobridge [12]. Estas variables pueden ser modificadas para obtener información desde los sensores de manera personalizada, donde "0!X" es el comando que elige la estación correspondiente. Por ejemplo, para seleccionar la lluvia de la segunda estación, se necesita modificar " rain0!Xrate-act=.3" a " rain0!1rate-act=.3". Las únicas excepciones en estos casos son para la temperatura, humedad, punto de rocío y factor de calor en el cual el número identificador para la segunda estación es "0!20", "0!40" para la tercera estación y "0!60" para la cuarta estación.

5.3. Pruebas de adquisición de datos de la red de estaciones con la Raspberry Pi

Dentro de la página de configuración de Meteobridge se puede acceder a los datos medidos en vivo. Aquí se puede verificar el estado de la subida de mediciones a la base de datos *mySQL* al igual que las últimas mediciones que han ingresado a la Raspberry Pi.

			min	max	min	max	min	max	min	max
0	indoor temperature	15.9°C	15.3°C	16.2°C	14.2°C	16.6°C	14.2°C	31.3°C	14.2°C	32.0°C
0	indoor humidity	61%	59%	61%	57%	64%	27%	69%	27%	69%
0	indoor dew point	8.4°C	7.3°C	8.7°C	6.5°C	9.4°C	4.9°C	17.5°C	4.9°C	17.5°C
0	pressure	1011.3mb	1005.4mb	1011.6mb	1005.4mb	1019.9mb	993.8mb	1022.0mb	991.4mb	1022.0mb
0	sealevel pressure	1011.3mb	1005.4mb	1011.6mb	1005.4mb	1019.9mb	993.8mb	1022.0mb	991.4mb	1022.0mb
0	outdoor temperature	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	32.7°C	12.1°C	36.8°C
0	outdoor humidity	67%	64%	68%	62%	70%	26%	77%	26%	77%
0	outdoor dewpoint	8.2°C	7.0°C	8.8°C	6.0°C	9.5°C	4.3°C	17.4°C	4.3°C	17.4°C
0	outdoor heat index	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	32.7°C	12.1°C	36.8°C
0	wind speed	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.4m/s	0.0m/s	0.4m/s
0	average wind speed	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	rain fall (total amount)		0.0mm		0.0mm		0.0mm		0.5mm	
0	rain rate (hourly)	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.3mm/h
0	wind chill	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	32.7°C	12.1°C	36.0°C
0	wind0!1wind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.4m/s	0.0m/s	0.4m/s
0	wind0!1avgwind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	rain0!1total		0.0mm		0.0mm		0.0mm		0.0mm	
0	rain0!1rate	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h
0	wind0!1chill	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	29.9°C	12.1°C	29.9°C
0	data0!1num	1554684	1473416	1554684	605816	1554684	-2147474	2147465	-2147474	2147465
0	wind0!2wind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	wind0!2avgwind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	rain0!2total		0.0mm		0.0mm		0.0mm		0.0mm	
0	rain0!2rate	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h
0	wind0!2chill	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	29.9°C	12.1°C	29.9°C
0	data0!2num	-27	-27	-27	-27	-27	-27	-27	-27	-27
0	wind0!3wind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	wind0!3avgwind	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s	0.0m/s
0	rain0!3total		0.0mm		0.0mm		0.0mm		0.0mm	
0	rain0!3rate	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h	0.0mm/h
0	wind0!3chill	14.2°C	13.3°C	15.3°C	12.1°C	15.4°C	12.1°C	29.9°C	12.1°C	29.9°C

Network | Weather Station | Weather Network | Services | System | License | Live Data

Upload Status | Raw Sensor Data | Min/Max Data | Weather Dashboard

Weather Network Status

Weather Underground: 2020-12-28 01:59:29 Success: 2020-12-28 01:59:28

MYSQL Upload: 2020-12-28 01:57:03 Success: 2020-12-28 01:57:01

Figura 5.10: Últimas mediciones obtenidas por la Raspberry Pi.

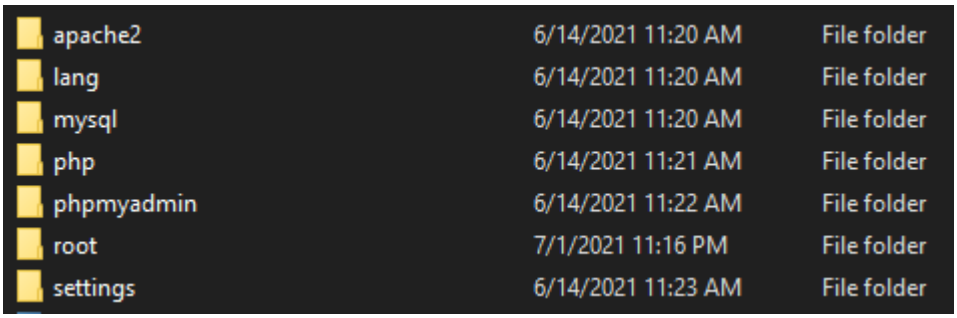
Capítulo 6

Configuración de servidor y base de datos

6.1. Desarrollo de servidor USB portátil LAMP

La configuración inicial del servidor web portátil es bastante fácil, sin embargo, existen pasos claves que deben ser realizados para lograr la interconexión del servidor y la base datos *mySQL*.

Al iniciar la aplicación *USBWebserver*,se debe seleccionar la unidad USB que se desea utilizar. Una vez elegida la unidad, la aplicación procede a generar las carpetas necesarias para la "LAMP Stack" dentro del pendrive USB. Junto a estas carpetas se generan archivos de configuración en los cuales se debe explícitamente indicar cuales son los puertos necesarios para la conexión entre el servidor, la página web y la base de datos. Estos archivos se pueden encontrar en la carpeta llamada "settings", tal como se muestra en la figura 6.1.



apache2	6/14/2021 11:20 AM	File folder
lang	6/14/2021 11:20 AM	File folder
mysql	6/14/2021 11:20 AM	File folder
php	6/14/2021 11:21 AM	File folder
phpmyadmin	6/14/2021 11:22 AM	File folder
root	7/1/2021 11:16 PM	File folder
settings	6/14/2021 11:23 AM	File folder

Figura 6.1: Formato final del servidor portátil USB.

Dentro de esta carpeta se encuentran cuatro archivos de configuración (figura 6.2). Es importante modificar el archivo llamado "usbwebserver" y cambiar

el puerto “apache” a 80 y el puerto “mysql” a 3306. Esto se puede apreciar en la figura 6.3.

httpd.conf	12/22/2020 3:25 PM	CONF File
my	6/26/2021 6:05 PM	Configuration settings
php	12/22/2020 3:25 PM	Configuration settings
usbwebserver	6/26/2021 6:02 PM	Configuration settings

Figura 6.2: Archivos de configuración dentro de la carpeta “settings”.

```
[apache]
port=80
[mysql]
port=3306
[algemeen]
slocal=0
hide=0
local=0
root={path}/root
lang=English
```

Figura 6.3: Configuración del archivo “usbwebserver”.

Es importante asignar los puertos correctos para permitir la comunicación entre el servidor y la base de datos *mySQL*. Una vez terminada la configuración, los datos de la base de datos se empezarán a guardar dentro del servidor USB. Esto permite una capacidad de almacenamiento de datos directamente proporcional a la capacidad de memoria de la unidad USB.

6.2. Configuración de base de datos MYSQL

Para facilitar la configuración y administración de la base de datos, se utiliza *phpMyAdmin* que viene incluida en la instalación del *USBWebserver*. Esta herramienta, creada en lenguaje PHP, permite crear una base de datos *mySQL* y administrarla a través de un browser web.

Al iniciar *phpMyAdmin* se requiere el nombre del usuario y la clave correspondiente. Se definió el usuario como “root” y la clave “stationLER” para ser las credenciales de la herramienta. Se procede a crear una base de datos con el nombre “meteobridge” y se define las tablas con los nombres “VP2_0”, “VP2_1”, “VP2_2” y “VP2_3” para las cuatro estaciones, respectivamente. Esto se puede apreciar en la figura 6.4.

VP2_0	Browse	Structure	Search	Insert	Empty	Drop	4,943
VP2_1	Browse	Structure	Search	Insert	Empty	Drop	4,928
VP2_2	Browse	Structure	Search	Insert	Empty	Drop	4,927
VP2_3	Browse	Structure	Search	Insert	Empty	Drop	4,926

Figura 6.4: Creación de la base de datos "meteobridge" con tablas para cada estación.

La estructura determina los parámetros y mediciones obtenidas desde la Raspberry Pi. Meteobridge tiene una librería de variables lógicas que pueden ser utilizadas para rescatar distintos valores de las mediciones dentro de la Raspberry Pi [12]. Utilizando las variables definidas en el capitulo 5 se puede configurar la estructura de cada tabla por separado. Las estructuras de las tablas deben tener los mismos formatos que las consultas generadas en el capitulo anterior y se debe definir el tipo de variable asociado a cada variable lógica. Estas variables pueden ser de tipo *INT, BOOL, DEC, STRING, DATETIME* entre otros.

Table name: Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes
	INT		None		
	INT		None		
	INT		None		
	INT		None		

Structure

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition:

Partition by: (Expression or column list)

Partitions:

Figura 6.5: Interfaz para definir la estructura de la base de datos.

Table structure | Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**ID**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**DateTime**	datetime			No	*None*			Change Drop More
☐	3	**TempOutCur**	decimal(10,3)			No	*None*			Change Drop More
☐	4	**HumOutCur**	decimal(6,3)			No	*None*			Change Drop More
☐	5	**PressCur**	decimal(10,3)			No	*None*			Change Drop More
☐	6	**DewCur**	decimal(6,3)			No	*None*			Change Drop More
☐	7	**HeatIdxCur**	decimal(6,3)			No	*None*			Change Drop More
☐	8	**WindChillCur**	decimal(6,3)			No	*None*			Change Drop More
☐	9	**WindSpeedCur**	decimal(6,3)			No	*None*			Change Drop More
☐	10	**WindAvgSpeedCur**	decimal(6,3)			No	*None*			Change Drop More
☐	11	**WindDirCur**	int(11)			No	*None*			Change Drop More
☐	12	**WindDirCurEng**	varchar(3)	utf8_unicode_ci		No	*None*			Change Drop More
☐	13	**WindGust10**	decimal(6,3)			No	*None*			Change Drop More
☐	14	**WindDirAvg10**	int(11)			No	*None*			Change Drop More
☐	15	**WindDirAvg10Eng**	varchar(3)	utf8_unicode_ci		No	*None*			Change Drop More
☐	16	**RainRateCur**	decimal(6,3)			No	*None*			Change Drop More
☐	17	**RainDay**	decimal(6,3)			No	*None*			Change Drop More
☐	18	**RainYest**	decimal(6,3)			No	*None*			Change Drop More
☐	19	**RainMonth**	decimal(6,3)			No	*None*			Change Drop More
☐	20	**RainYear**	decimal(6,3)			No	*None*			Change Drop More

Figura 6.6: Estructura de la primera estación Vantage Pro 2.

En cada variable se debe definir su largo correspondiente, asegurando tener suficientes dígitos para capturar la precisión de cada sensor.

6.3. Pruebas de accesibilidad remota a los datos adquiridos con la Rapsberry Pi

Una vez finalizada la configuración de la base de datos, basta con acceder a ésta para verificar que los datos estén siendo almacenados en el formato deseado para cada variable.

DateTime	TempOutCur	HumOutCur	PressCur	DewCur	HeatIdxCur	WindChillCur
2021-06-24 14:56:15	15.200	67.000	1017.000	9.100	15.200	15.200
2021-06-24 15:01:16	15.200	67.000	1017.000	9.100	15.200	15.200
2021-06-24 15:06:19	15.300	67.000	1017.000	9.200	15.300	15.300
2021-06-24 15:11:20	15.300	67.000	1017.000	9.200	15.300	15.300
2021-06-24 15:16:23	15.400	67.000	1017.000	9.300	15.400	15.400
2021-06-24 15:21:24	15.400	67.000	1016.000	9.300	15.400	15.400

WindSpeedCur	WindAvgSpeedCur	WindDirCur	WindDirCurEng	WindGust10	WindDirAvg10
0.000	0.000	161	SSE	0.000	161
0.000	0.000	161	SSE	0.000	161
0.000	0.000	161	SSE	0.000	161
0.000	0.000	161	SSE	0.000	161
0.000	0.000	161	SSE	0.000	161
0.000	0.000	161	SSE	0.000	161

Figura 6.7: Variables dentro de la base de datos con sus valores deseados.

Se puede observar en la figura 6.7 que los valores de las variables tienen ceros adicionales. Esto es para asegurar que los sensores estén entregando su precisión máxima y no se trunquen los valores. Además, al momento de obtener estas figuras, las estaciones Vantage Pro 2 se encuentran dentro de un laboratorio. Por lo tanto, no es posible obtener mediciones de viento para estas tablas, pero se confirma que efectivamente se está ingresando un valor cero, no nulo, a la base de datos.

Capítulo 7

Desarrollo de un cliente web y Resultados

7.1. Creación del formato de la pagina web

La página web es el lugar donde todo el trabajo realizado hasta este punto podrá ser visualizado. Es importante desplegar la información disponible en un formato limpio y fácil de entender. Es por esto que se optó por crear una página web cuya barra de navegación tenga pestañas, donde cada pestaña contiene información relevante de las variables requeridas en el estudio meteorológico.

Para lograr lo anterior, se utilizó el lenguaje *HTML* para crear el formato de la página y el lenguaje *CSS* para modificar su estilo. Es conveniente utilizar *HTML* ya que el lenguaje es estático y el usuario no necesita enviar información al servidor para navegar a través de la página web.

A la página principal se le denomina "Index", está dentro de la carpeta "root" del *USBwebserver* y funciona como un punto de inicio para llegar a las otras pestañas.

```
<div class="navbar">
  <a href="index.php">Home</a>
   <div class="dropdown"> <!--PESTAÑA PARA LA HUMEDAD -->
    <button class="dropbtn">Humedad
      <i class="fa fa-caret-down"></i>
    </button>
    <div class="dropdown-content"><!--CONTENIDO DEL MENU PARA LA
    HUMEDAD -->
      <a href="highchartsHum.php">Serie de Tiempo</a>
      <a href="highchartsHumPerf.php">Perfil Vertical </a>
    </div>
  </div>
  <div class="dropdown">
```

```
13      <button class="dropbtn">Precipitación <!—PESTAÑA PARA LA LLUVIA
          —>
14        <i class="fa fa-caret-down"></i>
15      </button>
16      <div class="dropdown-content"> <!—CONTENIDO DEL MENU PARA LA
        LLUVIA —>
17        <a href="highchartsPrecipDay.php">Diaria</a>
18        <a href="highchartsPrecipMonth.php">Mensual</a>
19        <a href="highchartsPrecipYear.php">Anual</a>
20      </div>
21    </div>
22    <a href="highchartsPres.php">Presión</a> <!—PESTAÑA PARA LA
        PRESIÓN —>
23    <div class="dropdown"> <!—PESTAÑA PARA LA TEMPERATURA —>
24      <button class="dropbtn">Temperatura
25        <i class="fa fa-caret-down"></i>
26      </button>
27      <div class="dropdown-content"> <!—CONTENIDO DEL MENU PARA LA
        TEMPERATURA —>
28        <a href="highchartsTemp.php">Serie de Tiempo</a>
29        <a href="highchartsTempPerf.php">Perfil Vertical </a>
30      </div>
31    </div>
32    <div class="dropdown"> <!—PESTAÑA PARA EL VIENTO —>
33      <button class="dropbtn">Viento
34        <i class="fa fa-caret-down"></i>
35      </button>
36      <div class="dropdown-content"><!—CONTENIDO DEL MENU PARA EL
        VIENTO —>
37        <a href="highchartsWindDir.php">Dirección del Viento</a>
38        <a href="highchartsWindSpd.php">Velocidad Instantánea</a>
39        <a href="highchartsWindSpdAvg.php">Velocidad Promedio</a>
40        <a href="highchartsWindPerf.php">Perfil Vertical de la
        Velocidad</a>
41      </div>
42    </div>
43  </div>
44
45  <!—LIBRERIAS NECESARIAS PARA GRAFICAR CON HIGHCHARTS API —>
46  <script src="https://code.jquery.com/jquery.js"></script>
47      <!— Importo el archivo Javascript de Highcharts directamente
        desde su servidor —>
48  <script src="http://code.highcharts.com/stock/highstock.js"></script>
49  <script src="http://code.highcharts.com/modules/exporting.js">
        </script>
50  <script src="https://code.highcharts.com/highcharts.js"></script>
51  <script src="https://code.highcharts.com/modules/series-label.js">
        </script>
52  <script src="https://code.highcharts.com/modules/exporting.js">
        </script>
53  <script src="https://code.highcharts.com/modules/export-data.js">
```

```
    </script>
54 <script src="https://code.highcharts.com/modules/accessibility.js">
    </script>
```

El código de arriba logra generar una página simple para iniciar el desarrollo de las pestañas. El resultado se puede apreciar en la siguiente figura.

Figura 7.1: Página inical básica generada a través de HTML.

Con este formato simple es posible crear códigos distintos para cada variable medida, mejorando el rendimiento de la página y evitando largos tiempos de carga para la información desplegada. En la línea número 8 del código se puede apreciar que al hacer clic en la pestaña de temperatura, nos llevará a una página dinámica *PHP* donde el usuario será capaz de recibir y enviar información al servidor. Todas las pestañas repiten el código de más arriba, ya que este contiene las ubicaciones de las demás pestañas. Luego, para poder avanzar en el desarrollo de la página web, es necesario utilizar una *API* o "Application Programming Interface" que nos permitirá usar las mediciones obtenidas desde la base de datos para generar gráficos. En la siguiente sección se detalla el proceso para generar dichos gráficos.

7.2. Desarrollo de la visualización de los datos

La visualización de datos es clave en este trabajo y para lograr una visualización robusta fue necesario encontrar una *API* capaz de manejar los datos meteorológicos que se midieron. Sin embargo, independiente de la *API* utilizada, la manipulación del formato de la data obtenida es siempre necesaria. Es por esto que se decidió usar la *API* "Highcharts", ya que es rápida, robusta, moderna y gratis para fines de educación.

Para obtener los datos deseados, es necesario crear un código *PHP* para consultar o hacer un "QUERY" al servidor con las condiciones requeridas. Esto se logra utilizando el lenguaje *SQL* dentro del código *PHP*. Las siguientes líneas de código permiten solicitar al servidor la data según las variables indicadas. El código presentado es el de la pestaña de temperatura, pero está porción se repite en los códigos de todas las variables.

```
//CONEXIÓN A LA BASE DE DATOS
$conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
    'Cdfuzpvpan', 'sql10439029');
```

Primero, se debe conectar a la base de datos utilizando la función *mysqli_connect(dirección IP, usuario, clave, base de datos)*, indicando la dirección de la base de datos, el nombre del usuario, la clave del usuario y el nombre de la base de datos. Las credenciales del código, *mysqli_connect('sql10.freemysqlhosting.net', 'sql10426069', 'KWtlHW3C8u' , 'sql10426069')*, son valores de prueba mientras se desarrolla la página web y deben ser cambiados para la configuración final.

Luego, se deben solicitar las columnas de las variables deseadas para ser graficadas. En este caso nos interesa obtener la fecha, hora y temperatura actual. Esto se logra con las siguientes líneas de código.

```
//CONSULTA A LA BASE DE DATOS PARA CADA ESTACION VANTAGE PRO
   2//////////////////////////////////////////////////

$sql_station1 =
' SELECT DateTime, TempOutCur AS Temp1 FROM VP2_0
  WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
';

$sql_station2 =
' SELECT DateTime, TempOutCur AS Temp2 FROM VP2_1
  WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
';

$sql_station3 =
' SELECT DateTime, TempOutCur AS Temp3 FROM VP2_2
  WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
';

  $sql_station4 =
' SELECT DateTime, TempOutCur AS Temp4 FROM VP2_3
  WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
';
```

Donde el comando "SELECT" nos permite seleccionar las variables *DateTime* y *TempOutCur*. La palabra "AS" nos permite modificar el nombre de las variables seleccionadas para ser utilizadas dentro del código, esto es útil cuando la variable tiene un nombre complejo o debe aparecer en el gráfico. El comando "FROM" nos permite indicar desde que tabla se desea extraer los datos, en este caso nos interesa las mediciones de la primera estación y su tabla llamada *VP2_0*. El comando "WHERE" nos permite restringir la información solicitada desde el servidor. En donde se está escogiendo data "DONDE" la variable *DateTime* está entre los valores $sql_StartDate y $sql_EndDate. Estas variables serán explicadas en mayor detalle en la siguiente sección. Finalmente, se utiliza la misma estructura de consulta para extraer los datos de las otras estaciones.

7.3. Desarrollo de la interfaz para el usuario final

En principio, se tiene toda la información disponible desde la base de datos, pero es necesario que el usuario pueda filtrar las mediciones desplegadas en los gráficos. Para lograr esto, se necesita crear código dinámico en lenguaje *PHP* que le permita al usuario enviar sus opciones al servidor y obtener así las estaciones que desea revisar y el periodo de tiempo en que quisiera visualizar.

Esto se puede realizar utilizando un objeto dentro del código llamado "form". Este tipo de formulario permite al usuario enviar sus elecciones al servidor. Para especificar las estaciones que el usuario quiere ver, se le solicita rellenar, mediante unas checkboxes, cuales de las cuatro estaciones quiere visualizar. Además, se le pide al usuario elegir entre dos fechas en donde quiere solicitar los datos mediante un calendario que se despliega en la interfaz.

```
1 //DEFINICION DEL FORMULARIO, CAJAS PARA REVISAR ESTACIONES Y FECHA
      INICIAL Y FINAL PARA VISUALIZAR//////////////////
2   ?>
3   <form action="highchartsTemp.php" method="POST">
4   Estación 2m: <input type="checkbox" name="Estacion1" value="true"
      <?php if(isset($_POST['Estacion1'])) echo 'checked="checked"';
      ?>/>
5   Estación 10m : <input type="checkbox" name="Estacion2" value="true"
       <?php if(isset($_POST['Estacion2'])) echo 'checked="checked"';
      ?>/>
6   Estación 30m: <input type="checkbox" name="Estacion3" value="true"
      <?php if(isset($_POST['Estacion3'])) echo 'checked="checked"';
      ?>/>
7   Estación 50m : <input type="checkbox" name="Estacion4" value="true"
       <?php if(isset($_POST['Estacion4'])) echo 'checked="checked"';
      ?>/>
8   <br>
```

En el código se puede apreciar que existen cuatro estaciones para seleccionar a distintas alturas. Una vez el usuario selecciona las estaciones deseadas, estas toman el valor de "true" después que la página se actualice. La variable de cada estación puede ser revisada por el servidor utilizando el comando *$_POST['EstacionX']*. Cabe destacar que al estar marcada una caja, entrega un valor verdadero, sin embargo, si no se selecciona dicha caja, el servidor recibe un valor tipo nulo para esa estación y no un valor de tipo falso o cero. Debido a esto, es importante definir casos especiales donde se revise en especifico por estos casos nulos y corregirlos a falso o cero. Además se debe corregir y entregar el comportamiento por defecto (mostrar todas las estaciones) en caso que ninguna estación haya sido seleccionada.

```php
<?php
    // REVISAR SI SE ESPECIFICO CUAL ESTACIÓN SE DEBE GRAFICAR, SI NO, GRAFICAR TODAS//////////////////////////////////////////////
    $Estacion1 = isset($_POST['Estacion1']);
    if (empty($Estacion1)){
      $Estacion1=0;
      }

    $Estacion2 = isset($_POST['Estacion2']);
    if (empty($Estacion2)){
      $Estacion2=0;
    }

    $Estacion3 = isset($_POST['Estacion3']);
    if (empty($Estacion3)){
      $Estacion3=0;
    }

    $Estacion4 = isset($_POST['Estacion4']);
    if (empty($Estacion4)){
      $Estacion4=0;
    }

    if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & ($Estacion4==0)){
      $Estacion1=1;
      $Estacion2=1;
      $Estacion3=1;
      $Estacion4=1;
    }

    $Estacion1=boolval($Estacion1) ? 'true' : 'false';
    $Estacion2=boolval($Estacion2) ? 'true' : 'false';
    $Estacion3=boolval($Estacion3) ? 'true' : 'false';
    $Estacion4=boolval($Estacion4) ? 'true' : 'false';
```

En el mismo formulario para seleccionar las estaciones, se le solicita al usuario entregar una “Fecha Inicial” y una “Fecha Final’ para las estaciones seleccionadas. Estas se pueden revisar por el servidor con los comandos *$_POST['StartDate']*, para la fecha de inicio y *$_POST['EndDate']* para la fecha final. Como al inicio no existe una fecha ingresada, se le fija una fecha deseada por el servidor. En este caso se optó por graficar una semana desde la fecha presente.

```
Fecha Inicial: <input type="date" id="StartDate" name="StartDate" value="<?php if (!isset($_POST['StartDate'])) {echo date_format($date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['StartDate'])) ?>" />
Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
<br>

<input type="submit" value="Guardar Cambios" />
</form>
```

Estas fechas no quedan definidas automáticamente al ingresar por primera vez a la página web, por lo tanto se debe programar una fecha de inicio y una fecha final que este activa por defecto. Sin embargo, una vez el usuario ingrese una fecha, estas deben quedar guardadas para ser utilizadas en la consulta para la base de datos. A continuación, se muestra el código creado para cumplir estos objetivos, guardando los valores en las variables $sql_StartDate y $sql_EndDate.

```
//DEFINICIÓN DE INTERVALO DE TIEMPO PARA GRAFICAR, ESPECIFICANDO CASO NULO//////////////////////
  $StartDayOffset=0;
  $EndDayOffset=0;
  if (!isset( $_POST['StartDate'])){
      $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
  }
  else {
    $sql_StartDate = new DateTime($_POST['StartDate']);
    $StartDateDiff = date_diff(date_create(),$sql_StartDate);
    $StartDayOffset=$StartDateDiff->days;
    $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY)";

  }
  if (!isset($_POST['EndDate'])){
      $sql_EndDate="NOW()";
  }
  else {
    $sql_EndDate = new DateTime($_POST['EndDate']);
    $EndDateDiff = date_diff(date_create(),$sql_EndDate);
    $EndDayOffset=$EndDateDiff->days;
    $EndDayOffset=$EndDayOffset-1;
    $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
  }
```

Toda esta información ingresa al servidor cuando el usuario hace clic en el botón de “Guardar Cambios” mediante un botón de tipo *submit*. Con esta operación se le solicita las mediciones deseadas al servidor y se actualiza la página. En la figura 7.2 se muestra el resultado del código.

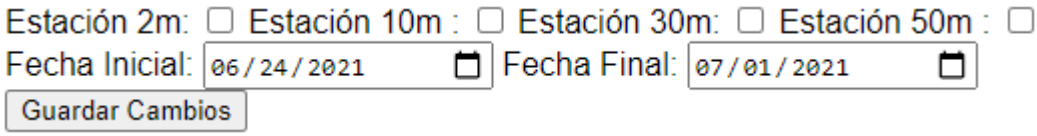


Figura 7.2: Interfaz de selección de mediciones para el usuario.

Con este paso finalizado es posible empezar a generar los gráficos deseados en un intervalo de tiempo definido. En la siguiente sección se detalla el proceso para graficar todas las variables de interés.

7.3.1. Visualización de rapidez de viento

La visualización de la rapidez del viento viene siendo uno de los objetivos más importantes de este trabajo, ya que gráficos como el perfil vertical del viento permiten analizar el potencial de energía eólica de un sector. Como las estaciones se encuentran dentro de un laboratorio y no al aire libre, es imposible obtener mediciones reales de las cuatro estaciones. Sin embargo, es posible utilizar datos obtenidos de otras estaciones en la web y poblar la base de datos con estas mediciones. De esta manera, se puede dejar el código funcionando para cuando las estaciones sean instaladas en sus ubicaciones finales.

Los datos obtenidos desde el Explorador Eólico [5] no pudieron ser importados directamente a la base de datos, esto debido a permisos necesarios para generar grandes cambios dentro de una base de datos con servidor externo. Este problema nace debido a las dificultades generadas por la pandemia de COVID-19 y no será un problema cuando la base de datos este dentro del LER. Debido a esto, se ingresaron 50 datos manualmente a la base de datos basados en mediciones reales y se sumó un delta a cada estación para simular el aumento de velocidad del viento con el cambio de altura.

7.3.1.1. Perfil vertical del viento

Se creó el siguiente código para generar el gráfico de perfil del viento.

```
$sql_station1 =
' SELECT WindAvgSpeedCur FROM VP2_0
  WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
';

```

```
6   $sql_station2 =
7   ' SELECT WindAvgSpeedCur FROM VP2_1
8     WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
9   ';
```

```
1 //ESTACIÓN 1 ALTURA: 2 METROS
      ////////////////////////////////////////////////////////
2   $result_station1 = mysqli_query($conn, $sql_station1);
3
4
5   $prom_2m=0;
6   $i=0;
7   $data_station1=[[null]];
8   while ($row1 = mysqli_fetch_array($result_station1))
9   {
10    $data_station1[$i] = $row1;
11     $i++;
12  }
13  for($i = 0 ;$i<count($data_station1);$i++)
14  {
15      $prom_2m+= $data_station1[$i][0];
16  }
17  $total_values1=sizeof($data_station1);
18  $prom_2m=$prom_2m/$total_values1;
19
20
21 //ESTACIÓN 2 ALTURA: 10 METROS
      ////////////////////////////////////////////////////////
22 $result_station2 = mysqli_query($conn, $sql_station2);
23
24
25  $prom_10m=0;
26  $i=0;
27  $data_station2=[[null]];
28  while ($row2 = mysqli_fetch_array($result_station2))
29  {
30    $data_station2[$i] = $row2;
31     $i++;
32  }
33  for($i = 0 ;$i<count($data_station2);$i++)
34  {
35      $prom_10m+= $data_station2[$i][0];
36  }
37  $total_values2=sizeof($data_station2);
```

En el código, se puede apreciar la obtención de datos para dos alturas. Primero se deben extraer las mediciones mediante una consulta a la base de datos, estos datos quedan almacenados en una lista tipo *array* llamado "$result_stationX". Luego, es necesario extraer cada fila que contiene una medición de velocidad del viento ya que la lista contiene pares de valores con fecha y la medición correspondiente, esta nueva lista queda llamada "$data_stationX". En este caso,

para el perfil del viento se necesita promediar la velocidad de cada estación y se debe contar la cantidad de mediciones obtenidas. Este promedio queda guardado en la variable "$prom_Xm" que va sumando cada valor numérico de la velocidad y luego es divido por el total de datos contados en la variable "$total_valuesX".

Para este gráfico es suficiente entregarle unos pares de coordenadas en formato ["Altura", "Promedio velocidad"] a la API de Highcharts. Esto se ingresa en la sección *series:* del siguiente código.

```

<input type="submit" value="Guardar Cambios" />
</form>

<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'
    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Perfil del Viento'
    },
    xAxis: {
        type: 'line',
        minPadding: 0.2,
        maxPadding: 1,
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000,
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Altura (m)',
            margin: 10
        }
    },
    plotOptions: {
        line: {
            dataLabels: {
                enabled: false
            },
            enableMouseTracking: true
        }
    },
    tooltip: {
            valueSuffix: ' m',
            followPointer: false
        },
```

```
    series: [{
        name: 'Perfil del viento (m/s)',
        data: [[<?php echo $prom_2m;?>, 2],
                [<?php echo $prom_10m;?>, 10],
                [<?php echo $prom_30m;?>, 30],
                [<?php echo $prom_50m;?>, 50]]
    }],
    credits: {
```

Con este código se obtiene el siguiente gráfico de la figura 7.3, en donde se visualiza el aumento de velocidad del viento a medida que aumenta la altura. En la implementación final, el usuario es capaz de ingresar el intervalo de fechas que quiera, pudiendo generar gráficos del perfil del viento para un día, una semana, un mes, etc.

Figura 7.3: Perfil vertical del viento simulado.

7.3.1.2. Serie de tiempo

Una representación de la velocidad del viento más común es su serie de tiempo. En el siguiente código se detallan los pasos para poder crear este gráfico con los datos utilizados en la gráfica del perfil del viento.

```
//ESTACIÓN 1 ALTURA: 10 METROS
    ////////////////////////////////////////////////////////////
  $result_station1 = mysqli_query($conn, $sql_station1);
  $data_station1 = array();

  $i=0;
```

```
while ($row1 = mysqli_fetch_array($result_station1))
{
  $data_station1[$i] = $row1;
   $i++;
}
$valoresArray1;
$timeArray1;

for($i = 0 ;$i<count($data_station1);$i++)
{
    $valoresArray1[$i]= $data_station1[$i][1];
    //OBTENEMOS EL TIMESTAMP
    $time1= $data_station1[$i][0];
    $timeArray1[$i] = strtotime($time1)*1000;
    //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
}
```

Se puede apreciar que el proceso para extraer los datos es similar al del perfil del viento, sin embargo, para este gráfico nos interesa la fecha y tiempo de cada medición. Por lo tanto, se guardan ambos valores de la lista en variables distintas "$valoresArrayX" y "$timeArrayX". Lo más importante de este tipo de gráfico es el formato de la fecha y hora medida ya que deben tener el mismo formato que requiere la API de Highcharts. La API utiliza el formato *UNIX* y se define como la cantidad de milisegundos transcurridos desde la medianoche UTC del 1 de enero de 1970. Luego, debido a que el formato de fecha y hora medido por las estaciones es *Datetime*, se requiere convertir este formato de tipo *string* a *time* para luego ser multiplicado por mil.

Con esta manipulación de datos completa, se pueden entregar los datos a la API con coordenadas ["Tiempo Unix", "Velocidad"]. A continuación, se muestra la sección *series:* donde se crean estas coordenadas mediante una función que utiliza el comando *push* para unir el tiempo y valor medido.

```
    series: [{
        name: 'Vel. Viento Prom. (2m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station1);$i++){
                ?>
                data.push([<?php echo $timeArray1[$i];?>,<?php echo
 $valoresArray1[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion1;?>,
        showInLegend:<?php echo $Estacion1;?>
    },{
```

```
17          name: 'Vel. Viento Prom. (10m)',
18          data: (function () {
19
20
21                  var data = [];
22                  <?php
23                      for($i = 0 ;$i<count($data_station2);$i++){
24                  ?>
25                  data.push([<?php echo $timeArray2[$i];?>,<?php echo
     $valoresArray2[$i];?>]);
26                  <?php } ?>
27                  return data;
28              })(),
29          visible:<?php echo $Estacion2;?>,
30          showInLegend:<?php echo $Estacion2;?>
31      },{
32          name: 'Vel. Viento Prom. (30m)',
33          data: (function () {
34
35
36                  var data = [];
37                  <?php
38                      for($i = 0 ;$i<count($data_station3);$i++){
39                  ?>
40                  data.push([<?php echo $timeArray3[$i];?>,<?php echo
     $valoresArray3[$i];?>]);
41                  <?php } ?>
42                  return data;
43              })(),
44          visible:<?php echo $Estacion3;?>,
45          showInLegend:<?php echo $Estacion3;?>
46      },{
47          name: 'Vel. Viento Prom. (50m)',
48          data: (function () {
49
50
51                  var data = [];
52                  <?php
53                      for($i = 0 ;$i<count($data_station4);$i++){
54                  ?>
55                  data.push([<?php echo $timeArray4[$i];?>,<?php echo
     $valoresArray4[$i];?>]);
56                  <?php } ?>
57                  return data;
58              })(),
59          visible:<?php echo $Estacion4;?>,
60          showInLegend:<?php echo $Estacion4;?>
61      },
```

Finalmente, el código resulta en el gráfico obtenido en la figura 7.4, en donde se pueden ver claramente los 50 datos entrados manualmente a la base de datos en esta sección para simular condiciones exteriores a distintas alturas.

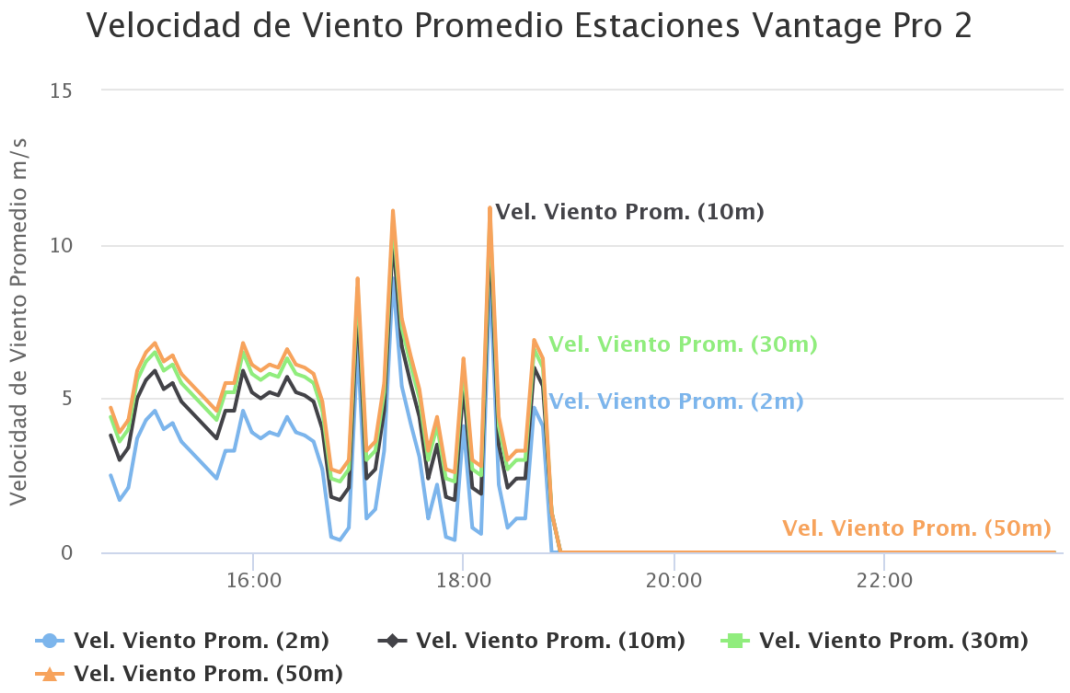


Figura 7.4: Serie de tiempo para la velocidad promedio del viento simulado.

7.3.2. Visualización de temperatura

La visualización de la temperatura con series de tiempo es posible hacerla con datos reales, sin embargo, para gráficos como el perfil vertical es necesario modificar las mediciones con un delta pequeño para simular el cambio de temperatura que tendrían de estar instaladas a distintas alturas. Esto, debido a que actualmente las cuatro estaciones se encuentran sobre una mesa dentro del laboratorio Shannon del Departamento de Electrónica en Casa Central.

7.3.2.1. Perfil vertical de la temperatura

Se agregó un delta pequeño en la base de datos de las estaciones 2, 3 y 4 para crear un cambio visible en el gráfico del perfil vertical. Tomando en cuenta lo anterior, se puede modificar la consulta de la sección 7.3.1.1 para extraer la temperatura en vez de la velocidad del viento. La sección modificada del código queda de la siguiente manera.

```
$sql_station1 =
' SELECT TempOutCur FROM VP2_0
  WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
';

$sql_station2 =
' SELECT TempOutCur FROM VP2_1
  WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
';
```

Luego se modifican los nombres de los títulos y las unidades para representar la información medida. A continuación se muestran los cambios necesarios.

```
<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Perfil Vertical de la Temperatura'
    },
    xAxis: {
        type: 'line',
        minPadding: 0.2,
        maxPadding: 1,
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000,

    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Altura (m)',
            margin: 10
        }
    },
    plotOptions: {
        line: {
            dataLabels: {
                enabled: false
            },
            enableMouseTracking: true
        }
    },
    tooltip: {
            valueSuffix: ' m',
```

```
40                 followPointer: false
41             },
42         series: [{
43             name: 'Perfil de la temperatura °C',
44             data: [[<?php echo $prom_2m;?>, 2],
45                     [<?php echo $prom_10m;?>, 10],
46                     [<?php echo $prom_30m;?>, 30],
47                     [<?php echo $prom_50m;?>, 50]]
48         }],
49         credits: {
50                 enabled: false
51         }
52 });
53 </script>
54 </body>
```

El resultado simulado para el perfil vertical de la temperatura se puede ver en la figura 7.5.

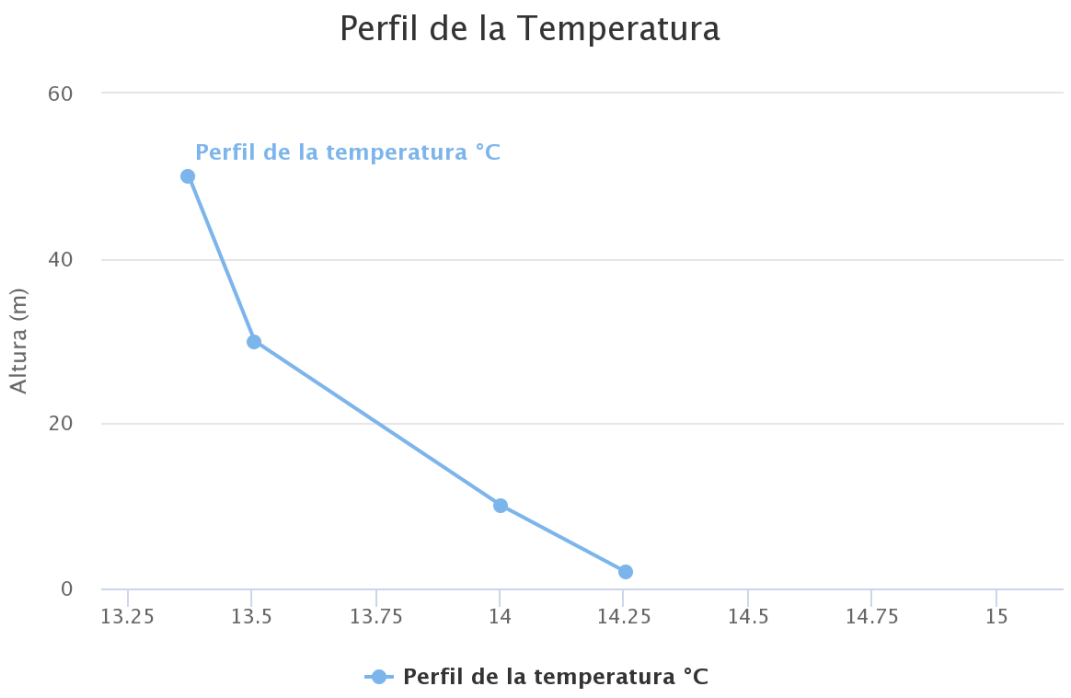


Figura 7.5: Perfil vertical de la temperatura.

7.3.2.2. Serie de tiempo

Utilizando el mismo código para la serie de tiempo de la velocidad del viento en la sección 7.3.1.2 y modificando la consulta a la base de datos como se hizo en la sección 7.3.2.1, solo falta cambiar los nombres de los títulos y unidades para generar el gráfico para la temperatura.

```
<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Temperaturas Estaciones Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime',
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Temperatura °C',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' °C',
            followPointer: true
        },
    series: [{
        name: 'Temperatura (2m)',
        data: (function () {


                var data = [];
```

Con estos cambios, se generaron los siguientes gráficos en las figuras 7.6 y 7.7.

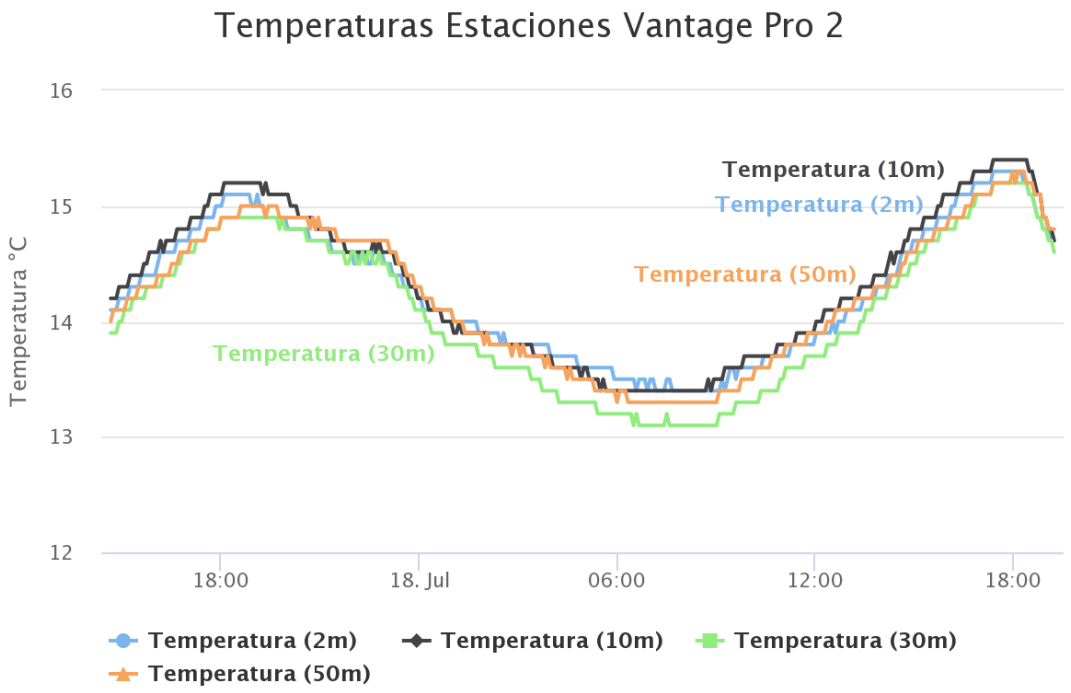


Figura 7.6: Serie de tiempo para las temperaturas.

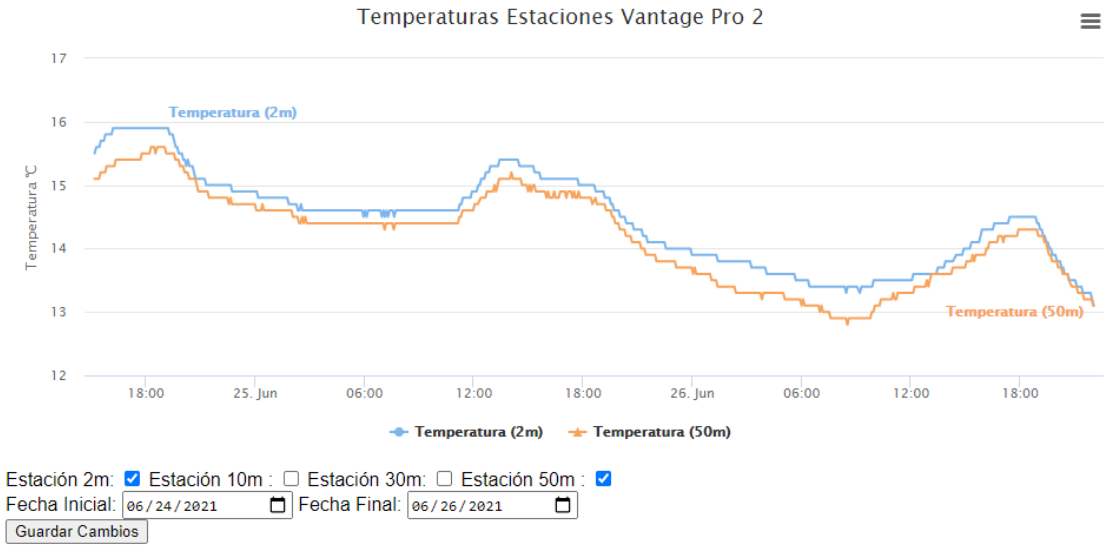


Figura 7.7: Captura de pantalla de la página web que muestra las opciones para seleccionar el rango de fechas y estaciones (alturas) a graficar en la serie de tiempo. La serie de tiempo visualiza las temperaturas a 2 y 50 metros de altura entre el 24 y 26 de junio.

7.3.3. Visualización de presión

Dado que el sistema de adquisición de datos solo permite almacenar la presión de una de las estaciones a la vez, debido a la configuración del Envoy8x con WDTU limitando la obtención de múltiples mediciones de presión por motivos desconocidos, es imposible generar un gráfico del perfil vertical. Solo se puede generar la serie de tiempo de dicha estación. En este caso, se puede obtener la presión de la estación a dos metros de altura. Luego, modificando el mismo código de la sección 7.3.1.2 se obtiene lo siguiente.

```
<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Presión Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime'
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Presión hPa',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' hPa',
            followPointer: true
        },
    series: [{
        name: 'Presión (2m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station1);$i++){
                ?>
                data.push([<?php echo $timeArray1[$i];?>,<?php echo
 $valoresArray1[$i];?>]);
```

```
                <?php } ?>
                return data;
            })()
    },
    credits: {
            enabled: false
    }
});
</script>
```

Con este código se obtiene la serie de tiempo de la presión de la figura 7.8 a una altura de dos metros.

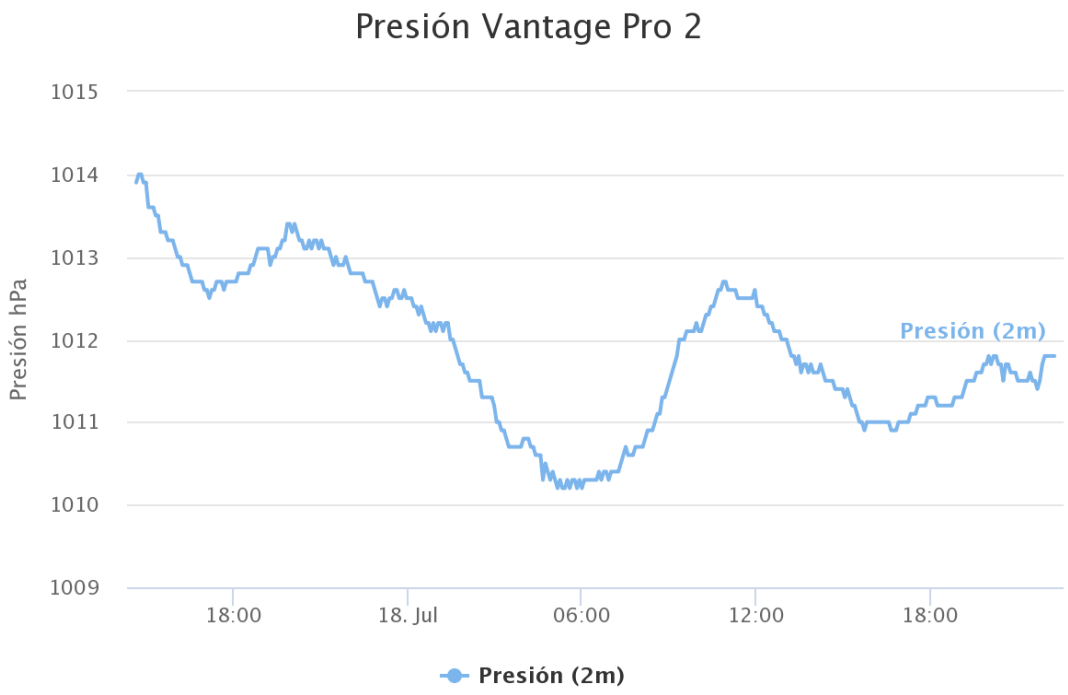


Figura 7.8: Serie de tiempo para la presión de la estación de 2 metros de altura .

7.3.4. Visualización de humedad

La visualización de la humedad queda programada para entregar el perfil vertical y la serie de tiempo de la variable, sin embargo, se opta por mostrar solo la serie de tiempo en esta sección. Esto debido a la baja resolución de las mediciones obtenidas combinado con la falta en cambio de altura de las estaciones.

La serie de tiempo se obtiene modificando la consulta a la base de datos y utilizando el código de la sección 7.3.1.2.

```
<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Humedad Estaciones Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime'
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Humedad %',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' %',
            followPointer: true
        },
    series: [{
        name: 'Humedad (2m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station1);$i++){
                ?>
                data.push([<?php echo $timeArray1[$i];?>,<?php echo
 $valoresArray1[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion1;?>,
        showInLegend:<?php echo $Estacion1;?>
```

Se obtiene finalmente el gráfico de la figura 7.9. Se puede apreciar la resolución baja de las mediciones mediante los escalones generados en el gráfico, esto es debido a la precisión de los sensores de humedad que son del 1 %.

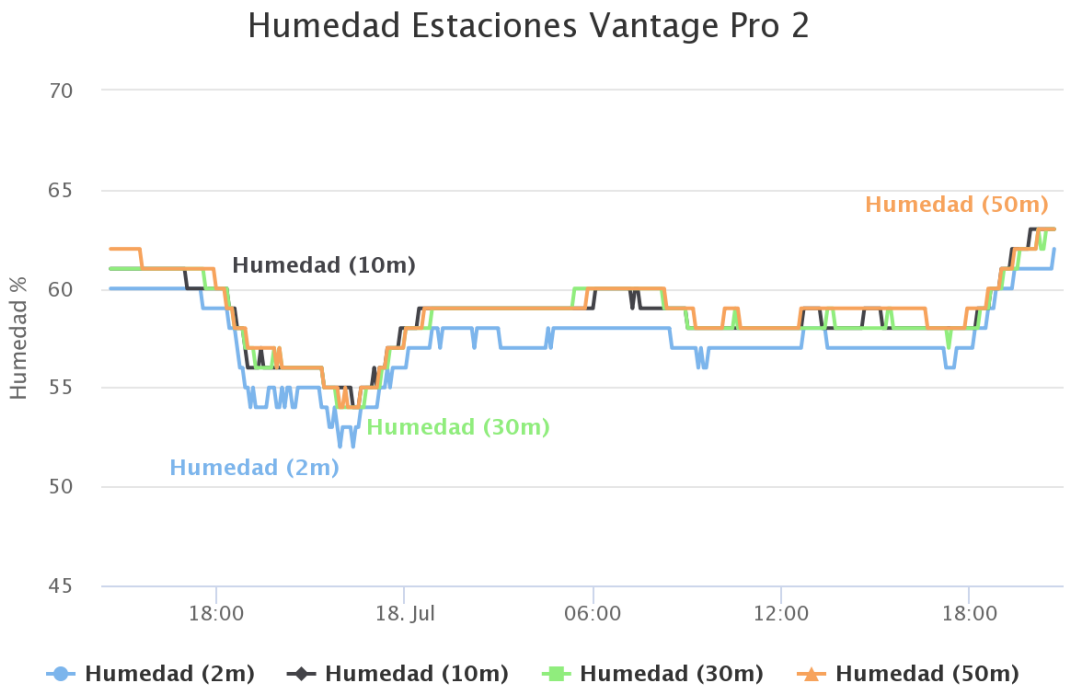


Figura 7.9: Serie de tiempo para las humedades.

7.3.5. Rosa de viento

La visualización de la dirección de viento viene siendo unas de las tarea más difíciles del proyecto. Esto debido a que se debe considerar la dirección del viento, el promedio de las velocidades y un intervalo de tiempo predeterminado. Como no existen mediciones para el viento debido a la ubicación de las estaciones durante el desarrollo de esta tesis, se utilizarán los 50 datos entrados manualmente que fueron utilizados en la sección 7.3.1.

La rosa de viento es una herramienta que permite visualizar la frecuencia del viento en cualquier dirección, cuantificadas por un intervalo de magnitud de las velocidades. Estos rangos de magnitudes son determinados previamente. Para esta rosa de viento se separaron las magnitudes en los rangos definidos del cuadro 7.1.

Rango	$< 0,5$	$\geq 0,5 < 2$	$\geq 2 < 4$	$\geq 4 < 8$	≥ 8

Cuadro 7.1: Rangos de magnitud definidos para la rosa de viento.

Con los rangos definidos, se deben crear distintas consultas a la base de datos para cada uno de ellos. Esto varía de las secciones anteriores ya que debemos hacer una comparación de valores antes de extraer las mediciones. Por lo tanto, las cinco consultas quedan definidas de la siguiente manera.

```
    $Estacion="VP2_0";
    }
  else $Estacion = $_POST["Estacion"];


  // write query for data

  $sql_station_LT05 =
  ' SELECT DateTime, WindAvgSpeedCur, WindDirCurEng, COUNT(*)
    FROM '.$Estacion.'
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
      AND WindAvgSpeedCur < 0.5
    GROUP BY WindDirCurEng
  ';

  $sql_station_GT05_LT2 =
  ' SELECT DateTime, WindAvgSpeedCur, WindDirCurEng, COUNT(*)
    FROM '.$Estacion.'
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
      AND WindAvgSpeedCur BETWEEN 0.5 AND 2
    GROUP BY WindDirCurEng
  ';

  $sql_station_GT2_LT4 =
  ' SELECT DateTime, WindAvgSpeedCur, WindDirCurEng, COUNT(*)
    FROM '.$Estacion.'
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
      AND WindAvgSpeedCur BETWEEN 2 AND 4
    GROUP BY WindDirCurEng
  ';

  $sql_station_GT4_LT8 =
  ' SELECT DateTime, WindAvgSpeedCur, WindDirCurEng, COUNT(*)
    FROM '.$Estacion.'
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
      AND WindAvgSpeedCur BETWEEN 4 AND 8
    GROUP BY WindDirCurEng
  ';
```

Las variables $sql_station_X_X usan la nomenclatura en ingles de *LessThan* o “LT” para especificar la operación “Menor A” y *GreaterThan* o

“GT” para especificar la operación “Mayor A”. Dentro de cada consulta se solicita la fecha, velocidad promedio, nombre de la dirección y cuantos resultados existen. Todos estos valores son de una sola estación y con los distintos rangos definidos se compara la velocidad promedio para luego ser agrupadas según el nombre de la dirección.

La API de Highcharts debe recibir una lista con las 16 direcciones posibles aunque no existan valores en algunas de ellas. Esto significa que se debe crear una lista con todos los nombres de direcciones posibles y manipular los datos obtenidos para que las direcciones reales queden dentro de su categoría respectiva. Para el caso entre magnitudes de 2 y 4 m/s, a continuación se detalla el código para generar un par de coordenadas con el formato [“Dirección”, “Magnitud”].

```
        $i++;
    }

    $dataArray_GT05_LT2 = [[]];
    for($i = 0 ;$i<count($data_station_GT05_LT2);$i++)
    {
        $total_count+=intval($data_station_GT05_LT2[$i][3]);
        $dataArray_GT05_LT2[$i] = [$data_station_GT05_LT2[$i][2],intval
       ($data_station_GT05_LT2[$i][3])];
    }

//TERCER RANGO: DATA MAYOR QUE 2 MENOR QUE 4
       ////////////////////////////////////////////
    $result_station_GT2_LT4 = mysqli_query($conn, $sql_station_GT2_LT4)
       ;

    $i=0;
    $data_station_GT2_LT4=[[null,null,null,null]];
    while ($row1 = mysqli_fetch_array($result_station_GT2_LT4))
```

En la variable $total_count se almacena la cantidad total de datos, independiente del rango de velocidad. Este valor es necesario para posteriormente calcular el porcentaje de las mediciones extraídas que cada rango y dirección contiene.

El resultado del código anterior es una lista con las direcciones y el número de repeticiones correspondientes, pero no necesariamente se obtuvieron datos en las 16 direcciones que existen. Para considerar estos casos, se define una lista en el mismo formato con las 16 direcciones y valores 0, luego, se reescribe el valor 0 al correspondiente para cada dirección obtenida originalmente. En el siguiente código se detalla el procedimiento anterior y se observa la conversión del número de ocurrencias a porcentaje del total.

```
    for($i = 0 ;$i<count($dataArray_GT05_LT2);$i++)
    {
      $j=0;
      for($j = 0 ;$j<count($finalArray_GT05_LT2);$j++)
```

```
    {
      if ($dataArray_GT05_LT2[$i][0]==$finalArray_GT05_LT2[$j][0]){

        $finalArray_GT05_LT2[$j][1]=$dataArray_GT05_LT2[$i][1]/
     $total_count*100;

      }
    }
  }
$finalArray_GT05_LT2 = json_encode($finalArray_GT05_LT2);

//TERCER RANGO: MANIPULACIÓN DE DATOS
     ////////////////////////////////////////////
  $finalArray_GT2_LT4 = [["N",0],["NNE",0],["NE",0],["ENE",0],["E"
     ,0],["ESE",0],["SE",0],["SSE",0],["S",0],["SSW",0],["SW",0],["
    WSW",0],["W",0],["WNW",0],["NW",0],["NNW",0]];
```

En el código se puede apreciar que la lista final queda almacenada en una variable llamada $finalArray_X_X. Esta variable debe ser transformada mediante la función *json_encode* para que quede en un formato que la API de Highcharts pueda utilizar.

Una vez terminado este paso para cada rango definido, se procede a entregar la información a la API con el siguiente código.

```
        },

        subtitle: {
            text: 'Laboratorio de Energías Renovables',
            align: 'left'
        },

        pane: {
            size: '95%'
        },

        legend: {
            align: 'right',
            verticalAlign: 'top',
            y: 100,
            layout: 'vertical'
        },

        xAxis: {
            tickmarkPlacement: 'on',
            labels: {
              formatter: function() {

                let label;
                switch (this.value) {
                  case 0:
                    label = 'N';
```

```
                    break;
                  case 1:
                    label = 'NNE';
                    break;
                  case 2:
                    label = 'NE';
                    break;
                  case 3:
                    label = 'ENE';
                    break;
                  case 4:
                    label = 'E';
                    break;
                  case 5:
                    label = 'ESE';
                    break;
                  case 6:
                    label = 'SE';
                    break;
                  case 7:
                    label = 'SSE';
                    break;
                  case 8:
                    label = 'S';
                    break;
                  case 9:
                    label = 'SSW';
                    break;
                  case 10:
                    label = 'SW';
                    break;
                  case 11:
                    label = 'WSW';
                    break;
                  case 12:
                    label = 'W';
                    break;
                  case 13:
                    label = 'WNW';
                    break;
                  case 14:
                    label = 'NW';
                    break;
                  case 15:
                    label = 'NNW';
                    break;
                }

                return label;
              }
    }
        },
```

```
        yAxis: {
            min: 0,
            endOnTick: false,
            showLastLabel: true,
            title: {
                text: 'Frequency (%)'
            },
            labels: {
                formatter: function () {
                    return this.value + '%';
                }
            }
        },

        tooltip: {
          valueDecimals: 2,
            valueSuffix: '%',
            followPointer: true
        },

        plotOptions: {
            series: {
                stacking: 'normal',
                shadow: false,
                groupPadding: 0,
                pointPlacement: 'on'
            }
        },
        series: [{
            "name": "&lt; 0.5 m/s",
            "data":<?php echo $finalArray_LT05 ;?>,
            "_colorIndex": 0
        },
        {
            "name": "0.5-2 m/s",
            "data":<?php echo $finalArray_GT05_LT2 ;?>,
            "_colorIndex": 1
        },
        {
            "name": "2-4 m/s",
            "data":<?php echo $finalArray_GT2_LT4 ;?>,
            "_colorIndex": 2
        },
        {
            "name": "4-8 m/s",
            "data":<?php echo $finalArray_GT4_LT8 ;?>,
            "_colorIndex": 3
        },
        {
            "name": "&gt; 8 m/s",
            "data":<?php echo $finalArray_GT8 ;?>,
```

```
                "_colorIndex": 4
            }
            ]
        });
});
</script>
</body>
</html>
```

Finalmente, se pueden apreciar algunos gráficos obtenidos con este código en las figuras 7.10, 7.11 y 7.12.

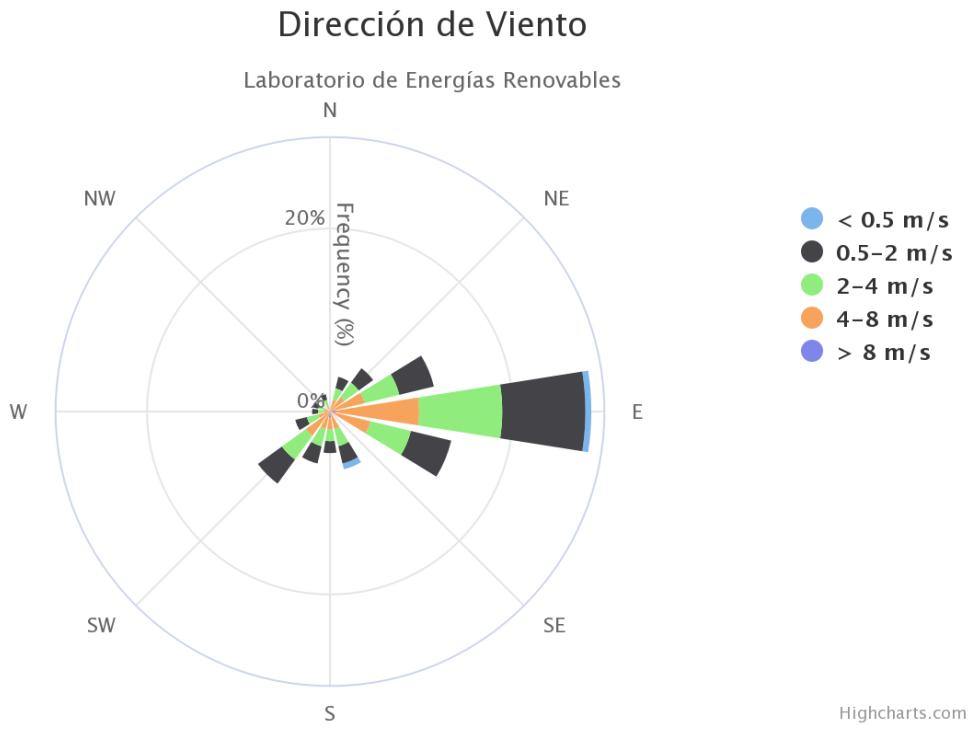


Figura 7.10: Rosa de viento para la velocidad promedio de una estación.

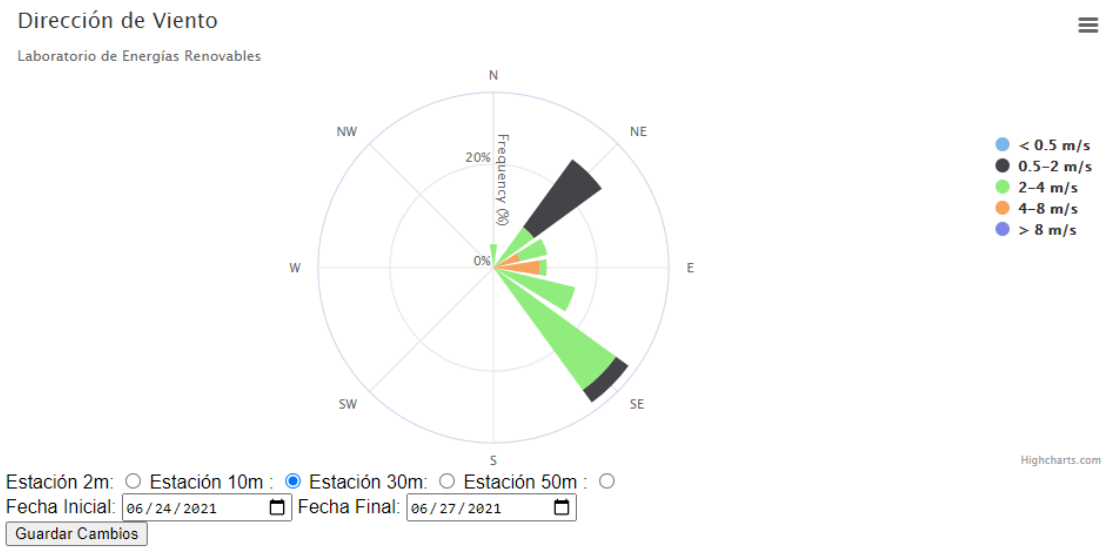


Figura 7.11: Captura de pantalla de la página web que muestra las opciones para seleccionar el rango de fechas y estación (altura) a graficar en la rosa de viento. La rosa de viento visualiza la frecuencia del viento en cada punto cardinal entre el 24 y 27 de junio.

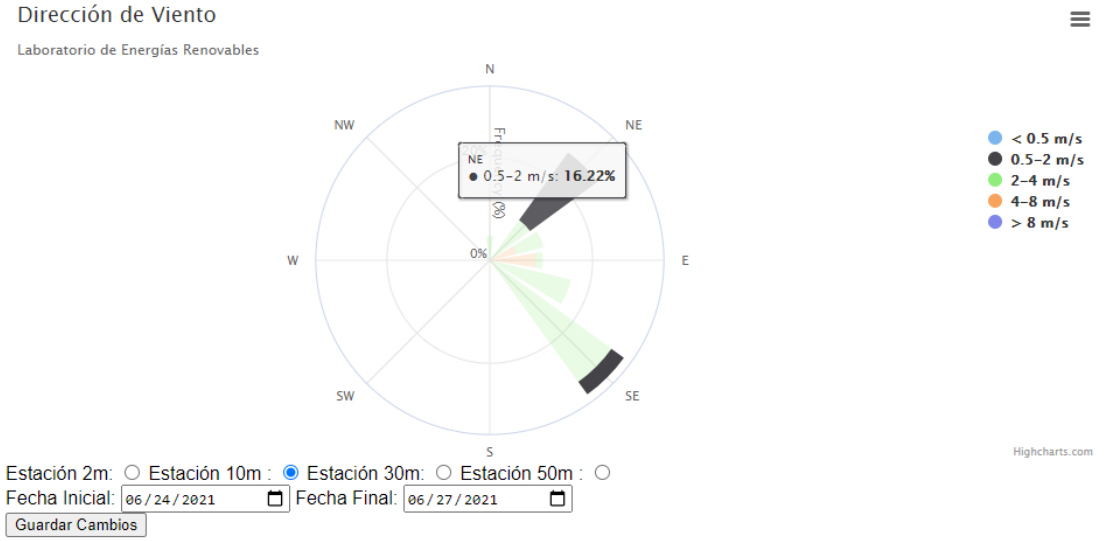


Figura 7.12: Detalle de la frecuencia de viento para la figura 7.11.

Capítulo 8

Conclusiones

8.0.1. Conclusiones

En esta memoria se presentó una primera iteración del proceso para generar una página web para la visualización de datos de una red de estaciones meteorológicas. Durante la tesis, se describió el contexto y proceso necesario para: la integración y configuración de las estaciones ISS; adquisición de datos mediante la Envoy8x; integración de todos los dispositivos mediante la Raspberry Pi; configuración de un servidor con base de datos *mySQL*; y el desarrollo de códigos para la visualización de datos mediante una *API* externa, entre las más importantes. Todo esto, para finalmente culminar en una pagina web que posee herramientas para facilitar el análisis de datos meteorológicos y la descarga de esos datos.

En particular, se logró integrar cuatro estaciones multiparamétricas a una sola Raspberry Pi. Esto fue logrado utilizando cuatro estaciones idénticas Vantage Pro 2 que permitían ser configuradas para transmitir en cuatro canales distintos, evitando la interferencia en la comunicación con el sistema de adquisición de datos llamado Envoy8x. Dicho sistema permitió crear un puente entre la comunicación de las cuatro estaciones multiparamétricas a la Raspberry Pi, ya que toma como entrada cuatro señales inalámbricas y las convierte en una pura señal cableada. Con esta señal cableada fue posible integrar las cuatro estaciones a la Raspberry Pi mediante un sistema operativo dedicado llamado *Meteobridge* que convierte a la Raspberry Pi en un mini servidor donde se pudieron enviar los datos obtenidos a una base de datos a través de internet. Con las mediciones disponibles online, fue posible desarrollar un cliente web que le permite al usuario final generar un análisis adicional de las mediciones con las herramientas implementadas. Con este sistema desarrollado es posible almacenar una gran cantidad de mediciones dentro de la base de datos *mySQL*, permitiendo generar una historización de los datos meteorológicos obtenidos. Según el fabricante, se requiere de 3.9 MB de memoria

para almacenar datos de un mes para una ISS con un intervalo de un minuto. Con esto se determina que se necesitan aproximadamente 1.2 GB de memoria para almacenar los datos de 4 estaciones ISS con un intervalo de 10 segundos por un año. Se destaca que, según los resultados obtenidos, no existe una pérdida de datos cuando se pierde la conexión a la red de internet debido al segundo almacenamiento disponible por parte de la Envoy8x. Este dispositivo no depende de la red para comunicarse con las estaciones y almacena los datos de manera cíclica dentro de su memoria interna. Esto resulta útil, ya que no se perderán datos por caídas o mantenciones programadas de la red.

De los resultados obtenidos, se puede confirmar que cualquier usuario o estudiante podrá acceder, de manera remota, a los datos medidos mediante el cliente web una vez sea instalado en el servidor del Laboratorio de Energías Renovables. Dentro del cliente web se encuentran herramientas que permiten un análisis posterior de las variables de interés, como el perfil vertical del viento a diferentes alturas. Por otro lado, el acceso de manera local también queda habilitado mediante las pantallas táctiles de cada estación.

Finalmente, con esta memoria quedan detallados los pasos para crear una base de datos con mediciones meteorológicas mediante un sistema de adquisición de datos de bajo consumo energético. Con un interfaz fácil de entender y usar para cualquier usuario conectado de manera remota.

8.0.2. Trabajo futuro

Este trabajo se realizó durante la pandemia de COVID-19, debido a esto, no fue posible realizar todo el trabajo presencial y experimental asociado. En esta sección se detallarán los pasos para completar algunas de las tareas pendientes e indicar sugerencias para mejorar el funcionamiento del sistema.

8.0.2.1. Instalación del sistema en el LER

Una de las pruebas importantes a realizar presencialmente es el alcance de cada estación con su pantalla receptora y la Envoy8x. Para lograr esto, se debe alejar la pantalla receptora desde cada estación transmisora. Las estaciones deben tener un alcance de 300 metros linea vista. Sin embargo, el rango con obstáculos en el camino suele variar dependiendo del objeto. El alcance de cada estación debe probarse en linea vista con la pantalla receptora en el piso y la estación transmisora instalada en la torre del LER. Una vez verificada la distancia de alcance alrededor de 300 metros, se debe probar el alcance efectivo cuando la pantalla receptora está dentro un cuarto o un gabinete eléctrico. Esta distancia de alcance determinará donde se podrá instalar el gabinete con la Raspberry Pi, las consolas y Envoy8x, ya que las antenas de las consolas y Envoy8x no pueden ser desmontadas.

Las pantallas receptoras tienen dimensiones de 270 mm de largo, 156 mm de ancho y 41 mm de profundidad. Para colocar las pantallas en una configuración apropiada para que el usuario pueda acceder la información, es necesario tener un gabinete que pueda almacenar dos pantallas lado a lado como el de la figura 8.1.

Figura 8.1: Gabinete de doble puerta con dimensiones 1000 x 800 x 250 mm.

La configuración ideal de estas pantallas se puede apreciar en la figura 8.2 junto a sus componentes necesarias del sistema. Es importante detallar que no es necesario tener la Raspberry Pi y Envoy8x al lado de las pantallas receptoras pero ayuda en la comunicación inalámbrica entre las estaciones transmisoras y la Envoy8x. También es importante tener conexión a la red de internet con la Raspberry Pi, esto puede ser cableado o a través de WiFi.

El espacio libre del gabinete puede quedar para la alimentación del sistema o si se decide, incorporar alguna pantalla adicional en el futuro.

Para instalar la página web y la base de datos en el Laboratorio de Energías Renovables es necesario contactarse con el encargado del servidor del laboratorio. Como el sistema completo de la página web fue diseñado para ser traspasado a través de una pendrive USB, debería ser tan simple como “arrastrar” los archivos de la USB al servidor. Esto siempre cuando los archivos se coloquen en la ubicación necesaria para la configuración del servidor actual en el LER. Es posible que los

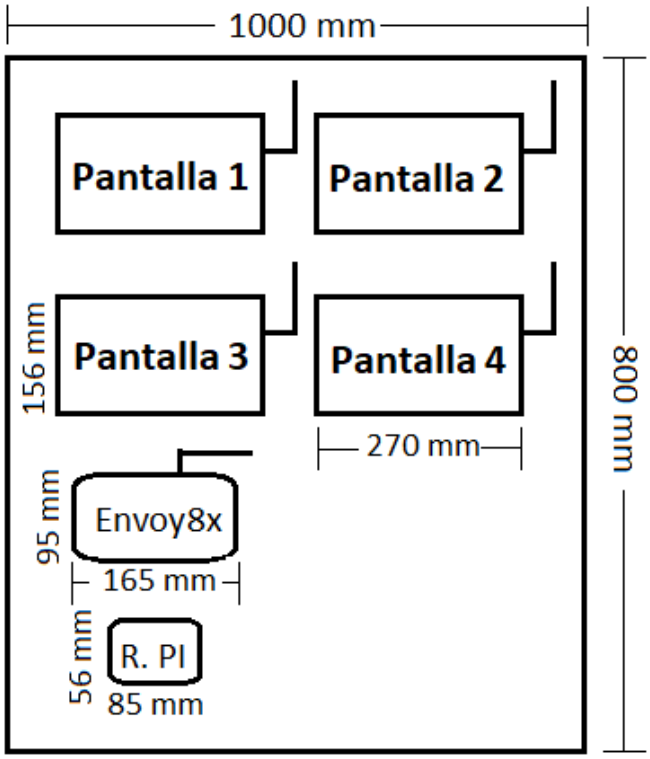


Figura 8.2: Configuración para una instalación apropiada de las pantallas receptoras y componentes del sistema.

puertos del servidor sean distintos a los definidos en este trabajo. Si esto es el caso, en el capitulo 6 se indican los pasos para generar los cambios.

8.0.2.2. Interferencia debido a la torre transmisora AM

Las cuatro estaciones ISS serán instaladas en la torre radio-emisora AM de 52 metros. Dado que es una torre AM, esto significa que la señal que emite es transmitida mediante las superficies alrededores. Esto puede causar interferencia en los datos medidos o en la transmisión de estos datos al sistema receptor. Es recomendado instalar las estaciones en un brazo o soporte aislante y no directamente en contacto con la torre AM. Utilizando este tipo de brazo ayuda en eliminar la posibilidad de corrientes inducidas en los circuitos dentro de los sensores de las estaciones ISS.

Es importante realizar las pruebas de alcance antes de las pruebas de interferencia, con la información previa será posible verificar si ademas de posibles corrientes inducidas, la torre afecta el alcance máximo de las estaciones.

8.0.2.3. Perfil vertical físicamente y análisis adicional

Debido a la pandemia y la imposibilidad de instalar las estaciones en el exterior, los perfiles verticales de este proyecto fueron creados con mediciones de estaciones reales encontradas online. Los códigos quedaron preparados para utilizar los datos de las estaciones Vantage Pro 2 y es necesario probar que los perfiles verticales de las variables medidas estén funcionando de manera correcta. Basta con esperar un par de días desde la instalación de las estaciones en el LER para que la base de datos se llene con mediciones reales y luego revisar los gráficos obtenidos desde la página web. Alternativamente, se pueden ubicar las cuatro estaciones en distintas alturas, como por ejemplo en el edificio "A" de la Casa Central y aumentar el muestreo de datos para la Raspberry Pi a 10 segundos. De esta manera sera posible llenar la base de datos con mediciones reales en unos pocos minutos.

8.0.2.4. Creación de herramientas adicionales

Al pasar el tiempo se actualizarán y crearán nuevos métodos para analizar las variables que existen dentro de la base de datos y sera necesario realizar cálculos nuevos. En esta sección se mostrara como agregar nuevas funciones a los códigos existentes o nuevos.

Como se ha explicado en las sección 7, siempre es necesario conectarse a la base de datos antes de poder acceder los datos mediante una consulta. Esto se hace dentro de un código de formato PHP con la siguiente función.

$$\$conn = new\ mysqli(\$nombreServidor, \$nombreUsuario, \$clave);$$

Una vez conectado, se procede a generar una consulta a la base de datos tal como se indica en la sección 7. La mayoría de los cálculos resultan ser más fáciles en formato PHP, por lo tanto, se pueden realizar todas las operaciones aritméticas dentro de un código PHP para finalmente ser entregados a cualquier API deseada.

Una referencia para ver lo que se puede lograr dentro del lenguaje PHP es [14], donde se puede aprender a utilizar el lenguaje sin necesidad de guardar los códigos o tener un desarrollador especifico.

Anexo A

Códigos Desarrollados:

A.1. Página principal

Código para generar la página principal y sus pestañas correspondientes:

```
<div class="navbar">
  <a href="index.php">Home</a>
   <div class="dropdown"> <!--PESTAÑA PARA LA HUMEDAD -->
    <button class="dropbtn">Humedad
      <i class="fa fa-caret-down"></i>
    </button>
    <div class="dropdown-content"><!--CONTENIDO DEL MENU PARA LA
     HUMEDAD -->
      <a href="highchartsHum.php">Serie de Tiempo</a>
      <a href="highchartsHumPerf.php">Perfil Vertical </a>
    </div>
  </div>
  <div class="dropdown">
    <button class="dropbtn">Precipitación <!--PESTAÑA PARA LA LLUVIA
      -->
      <i class="fa fa-caret-down"></i>
    </button>
    <div class="dropdown-content"> <!--CONTENIDO DEL MENU PARA LA
     LLUVIA -->
      <a href="highchartsPrecipDay.php">Diaria</a>
      <a href="highchartsPrecipMonth.php">Mensual</a>
      <a href="highchartsPrecipYear.php">Anual</a>
    </div>
  </div>
  <a href="highchartsPres.php">Presión</a> <!--PESTAÑA PARA LA
     PRESIÓN -->
  <div class="dropdown"> <!--PESTAÑA PARA LA TEMPERATURA -->
    <button class="dropbtn">Temperatura
      <i class="fa fa-caret-down"></i>
    </button>
```

```
27      <div class="dropdown-content"> <!--CONTENIDO DEL MENU PARA LA
        TEMPERATURA -->
28        <a href="highchartsTemp.php">Serie de Tiempo</a>
29        <a href="highchartsTempPerf.php">Perfil Vertical </a>
30      </div>
31    </div>
32    <div class="dropdown"> <!--PESTAÑA PARA EL VIENTO -->
33      <button class="dropbtn">Viento
34        <i class="fa fa-caret-down"></i>
35      </button>
36      <div class="dropdown-content"><!--CONTENIDO DEL MENU PARA EL
        VIENTO -->
37        <a href="highchartsWindDir.php">Dirección del Viento</a>
38        <a href="highchartsWindSpd.php">Velocidad Instantánea</a>
39        <a href="highchartsWindSpdAvg.php">Velocidad Promedio</a>
40        <a href="highchartsWindPerf.php">Perfil Vertical de la
        Velocidad</a>
41      </div>
42    </div>
43  </div>
44
45  <!--LIBRERIAS NECESARIAS PARA GRAFICAR CON HIGHCHARTS API -->
46  <script src="https://code.jquery.com/jquery.js"></script>
47      <!-- Importo el archivo Javascript de Highcharts directamente
        desde su servidor -->
48  <script src="http://code.highcharts.com/stock/highstock.js"></script>
49  <script src="http://code.highcharts.com/modules/exporting.js">
        </script>
50  <script src="https://code.highcharts.com/highcharts.js"></script>
51  <script src="https://code.highcharts.com/modules/series-label.js">
        </script>
52  <script src="https://code.highcharts.com/modules/exporting.js">
        </script>
53  <script src="https://code.highcharts.com/modules/export-data.js">
        </script>
54  <script src="https://code.highcharts.com/modules/accessibility.js">
        </script>
```

A.2. Códgio para la temperatura

Códigos para la pestaña de temperatura, con envió de datos y gráficos dinámicos:

```
1  <?php
2
3    mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
4    //CONEXIÓN A LA BASE DE DATOS
5    $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
         'Cdfuzpvpan', 'sql10439029');
```

```
6
7
8   //DEFINICIÓN DE INTERVALO DE TIEMPO PARA GRAFICAR, ESPECIFICANDO
      CASO NULO//////////////////////
9     $StartDayOffset=0;
10    $EndDayOffset=0;
11    if (!isset( $_POST['StartDate'])){
12        $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
13    }
14    else {
15      $sql_StartDate = new DateTime($_POST['StartDate']);
16      $StartDateDiff = date_diff(date_create(),$sql_StartDate);
17      $StartDayOffset=$StartDateDiff->days;
18      $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
      )";
19
20    }
21    if (!isset($_POST['EndDate'])){
22        $sql_EndDate="NOW()";
23    }
24    else {
25      $sql_EndDate = new DateTime($_POST['EndDate']);
26      $EndDateDiff = date_diff(date_create(),$sql_EndDate);
27      $EndDayOffset=$EndDateDiff->days;
28      $EndDayOffset=$EndDayOffset-1;
29      $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
30    }
31
32  // REVISAR CONEXIÓN A BASE DE DATOS
      ////////////////////////////////////////////////
33  if(!$conn){
34    echo 'Connection error: ' . mysqli_connect_error();
35  }
36
37
38  //CONSULTA A LA BASE DE DATOS PARA CADA ESTACION VANTAGE PRO
      2////////////////////////////////////////////////
39
40  $sql_station1 =
41  ' SELECT DateTime, TempOutCur AS Temp1 FROM VP2_0
42    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
43  ';
44
45  $sql_station2 =
46  ' SELECT DateTime, TempOutCur AS Temp2 FROM VP2_1
47    WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
48  ';
49
50  $sql_station3 =
51  ' SELECT DateTime, TempOutCur AS Temp3 FROM VP2_2
52    WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
```

```
53    ';
54
55      $sql_station4 =
56    ' SELECT DateTime, TempOutCur AS Temp4 FROM VP2_3
57      WHERE DateTime BETWEEN '.$sql_StartDate.' AND '.$sql_EndDate.'
58    ';
59
60
61  //ESTACIÓN 1 ALTURA: 2 METROS
        ////////////////////////////////////////////////////////
62    $result_station1 = mysqli_query($conn, $sql_station1);
63    $data_station1 = array();
64
65
66    $i=0;
67    while ($row1 = mysqli_fetch_array($result_station1))
68    {
69      $data_station1[$i] = $row1;
70       $i++;
71    }
72    $valoresArray1;
73    $timeArray1;
74
75    for($i = 0 ;$i<count($data_station1);$i++)
76    {
77        $valoresArray1[$i]= $data_station1[$i][1];
78        //OBTENEMOS EL TIMESTAMP
79        $time1= $data_station1[$i][0];
80        $timeArray1[$i] = strtotime($time1)*1000;
81        //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
82    }
83  //ESTACIÓN 2 ALTURA: 10 METROS
        ////////////////////////////////////////////////////////
84    $result_station2 = mysqli_query($conn, $sql_station2);
85    $data_station2 = array();
86
87    $i=0;
88    while ($row2 = mysqli_fetch_array($result_station2))
89    {
90      $data_station2[$i] = $row2;
91       $i++;
92    }
93    $valoresArray2;
94    $timeArray2;
95
96    for($i = 0 ;$i<count($data_station2);$i++)
97    {
98        $valoresArray2[$i]= $data_station2[$i][1];
99        //OBTENEMOS EL TIMESTAMP
100       $time2= $data_station2[$i][0];
101       $timeArray2[$i] = strtotime($time2)*1000;
```

```
102         //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
103     }
104 //ESTACIÓN 3 ALTURA: 30 METROS
        ////////////////////////////////////////////////////////
105     $result_station3 = mysqli_query($conn, $sql_station3);
106     $data_station3 = array();
107 
108     $i=0;
109     while ($row3 = mysqli_fetch_array($result_station3))
110     {
111       $data_station3[$i] = $row3;
112        $i++;
113     }
114     $valoresArray3;
115     $timeArray3;
116 
117     for($i = 0 ;$i<count($data_station3);$i++)
118     {
119         $valoresArray3[$i]= $data_station3[$i][1];
120         //OBTENEMOS EL TIMESTAMP
121         $time3= $data_station3[$i][0];
122         $timeArray3[$i] = strtotime($time3)*1000;
123         //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
124     }
125 
126 //ESTACIÓN 4 ALTURA: 50 METROS
        ////////////////////////////////////////////////////////
127     $result_station4 = mysqli_query($conn, $sql_station4);
128     $data_station4 = array();
129 
130     $i=0;
131     while ($row4 = mysqli_fetch_array($result_station4))
132     {
133       $data_station4[$i] = $row4;
134        $i++;
135     }
136     $valoresArray4;
137     $timeArray4;
138 
139     for($i = 0 ;$i<count($data_station4);$i++)
140     {
141         $valoresArray4[$i]= $data_station4[$i][1];
142         //OBTENEMOS EL TIMESTAMP
143         $time4= $data_station4[$i][0];
144         $timeArray4[$i] = strtotime($time4)*1000;
145         //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
146     }
147 ?>
148 
149 <!DOCTYPE html>
150 
```

```
151
152
153 <html>
154 <head>
155   <meta charset="utf-8">
156   <meta name="viewport" content="width=device-width, initial-scale=1"
      >
157 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
      font-awesome/4.7.0/css/font-awesome.min.css">
158 <?php include "./websiteStyle.html" ?>
159 </head>
160 <body>
161
162 <?php include "./websiteHeader.html" ?>
163
164 <div id="contenedor"></div>
165
166   <?php
167       // REVISAR SI SE ESPECIFICO CUAL ESTACIÓN SE DEBE GRAFICAR, SI
      NO, GRAFICAR TODAS/////////////////////////////////////////////
168       $Estacion1 = isset($_POST['Estacion1']);
169       if (empty($Estacion1)){
170         $Estacion1=0;
171         }
172
173       $Estacion2 = isset($_POST['Estacion2']);
174       if (empty($Estacion2)){
175         $Estacion2=0;
176       }
177
178       $Estacion3 = isset($_POST['Estacion3']);
179       if (empty($Estacion3)){
180         $Estacion3=0;
181       }
182
183       $Estacion4 = isset($_POST['Estacion4']);
184       if (empty($Estacion4)){
185         $Estacion4=0;
186       }
187
188       if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & (
      $Estacion4==0)){
189         $Estacion1=1;
190         $Estacion2=1;
191         $Estacion3=1;
192         $Estacion4=1;
193       }
194
195       $Estacion1=boolval($Estacion1) ? 'true' : 'false';
196       $Estacion2=boolval($Estacion2) ? 'true' : 'false';
197       $Estacion3=boolval($Estacion3) ? 'true' : 'false';
```

```
198         $Estacion4=boolval($Estacion4) ? 'true' : 'false';
199
200
201     $date=date_create();
202     date_add($date,date_interval_create_from_date_string("-7 days"));
203
204 //DEFINICION DEL FORMULARIO, CAJAS PARA REVISAR ESTACIONES Y FECHA
        INICIAL Y FINAL PARA VISUALIZAR//////////////////
205     ?>
206     <form action="highchartsTemp.php" method="POST">
207     Estación 2m: <input type="checkbox" name="Estacion1" value="true"
        <?php if(isset($_POST['Estacion1'])) echo 'checked="checked"';
        ?>/>
208     Estación 10m : <input type="checkbox" name="Estacion2" value="true"
         <?php if(isset($_POST['Estacion2'])) echo 'checked="checked"';
        ?>/>
209     Estación 30m: <input type="checkbox" name="Estacion3" value="true"
        <?php if(isset($_POST['Estacion3'])) echo 'checked="checked"';
        ?>/>
210     Estación 50m : <input type="checkbox" name="Estacion4" value="true"
         <?php if(isset($_POST['Estacion4'])) echo 'checked="checked"';
        ?>/>
211     <br>
212
213     Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
        value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
        $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
        StartDate'])) ?>" />
214     Fecha Final: <input type="date" id="EndDate" name="EndDate" value="
        <?php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
        echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
215     <br>
216
217     <input type="submit" value="Guardar Cambios" />
218     </form>
219
220     <script>
221     chartCPU = new Highcharts.Chart({
222         chart: {
223             renderTo: 'contenedor'
224             //defaultSeriesType: 'spline'
225
226         },
227         rangeSelector : {
228             enabled: false
229         },
230         title: {
231             text: 'Temperaturas Estaciones Vantage Pro 2'
232         },
233         xAxis: {
234             type: 'datetime',
235             //tickPixelInterval: 150,
```

```
236             //maxZoom: 20 * 1000
237         },
238         yAxis: {
239             minPadding: 0.2,
240             maxPadding: 0.2,
241             title: {
242                 text: 'Temperatura °C',
243                 margin: 10
244             }
245         },
246         tooltip: {
247                 valueSuffix: ' °C',
248                 followPointer: true
249             },
250         series: [{
251             name: 'Temperatura (2m)',
252             data: (function () {
253
254
255                 var data = [];
256                 <?php
257                     for($i = 0 ;$i<count($data_station1);$i++){
258                 ?>
259                 data.push([<?php echo $timeArray1[$i];?>,<?php echo
        $valoresArray1[$i];?>]);
260                 <?php } ?>
261                 return data;
262             })(),
263             visible:<?php echo $Estacion1;?>,
264             showInLegend:<?php echo $Estacion1;?>
265         },{
266             name: 'Temperatura (10m)',
267             data: (function () {
268
269
270                 var data = [];
271                 <?php
272                     for($i = 0 ;$i<count($data_station2);$i++){
273                 ?>
274                 data.push([<?php echo $timeArray2[$i];?>,<?php echo
        $valoresArray2[$i];?>]);
275                 <?php } ?>
276                 return data;
277             })(),
278             visible:<?php echo $Estacion2;?>,
279             showInLegend:<?php echo $Estacion2;?>
280         },{
281             name: 'Temperatura (30m)',
282             data: (function () {
283
284
285                 var data = [];
```

```
                    <?php
                        for($i = 0 ;$i<count($data_station3);$i++){
                    ?>
                    data.push([<?php echo $timeArray3[$i];?>,<?php echo
        $valoresArray3[$i];?>]);
                    <?php } ?>
                    return data;
                })(),
            visible:<?php echo $Estacion3;?>,
            showInLegend:<?php echo $Estacion3;?>
        },{
            name: 'Temperatura (50m)',
            data: (function () {


                    var data = [];
                    <?php
                        for($i = 0 ;$i<count($data_station4);$i++){
                    ?>
                    data.push([<?php echo $timeArray4[$i];?>,<?php echo
        $valoresArray4[$i];?>]);
                    <?php } ?>
                    return data;
                })(),
            visible:<?php echo $Estacion4;?>,
            showInLegend:<?php echo $Estacion4;?>
        }],
        credits: {
                enabled: false
        }
    });
</script>
</body>
</html>
```

```
<?php

  mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
  // connect to database
  //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
     wWtdTEvd5', 'DwIOid9JFJ');
  $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');



    $StartDayOffset=0;
    $EndDayOffset=0;
    if (!isset( $_POST['StartDate'])){
        $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
    }
    else {
```

```
16        $sql_StartDate = new DateTime($_POST['StartDate']);
17        $StartDateDiff = date_diff(date_create(),$sql_StartDate);
18        $StartDayOffset=$StartDateDiff->days;
19        $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
     )";
20
21      }
22      if (!isset($_POST['EndDate'])){
23          $sql_EndDate="NOW()";
24      }
25      else {
26        $sql_EndDate = new DateTime($_POST['EndDate']);
27        $EndDateDiff = date_diff(date_create(),$sql_EndDate);
28        $EndDayOffset=$EndDateDiff->days;
29        $EndDayOffset=$EndDayOffset-1;
30        $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
31      }
32
33    // check connection
34    if(!$conn){
35      echo 'Connection error: ' . mysqli_connect_error();
36    }
37
38
39    // write query for data
40
41    $sql_station1 =
42    ' SELECT TempOutCur FROM VP2_0
43      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
44    ';
45
46    $sql_station2 =
47    ' SELECT TempOutCur FROM VP2_1
48      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
49    ';
50
51    $sql_station3 =
52    ' SELECT TempOutCur FROM VP2_2
53      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
54    ';
55
56    $sql_station4 =
57    ' SELECT TempOutCur FROM VP2_3
58      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
59    ';
60
61
62  //ESTACIÓN 1 ALTURA: 2 METROS
     ////////////////////////////////////////////////////////////
63    $result_station1 = mysqli_query($conn, $sql_station1);
64
```

```
65
66    $prom_2m=0;
67    $i=0;
68    $data_station1=[[null]];
69    while ($row1 = mysqli_fetch_array($result_station1))
70    {
71      $data_station1[$i] = $row1;
72       $i++;
73    }
74    for($i = 0 ;$i<count($data_station1);$i++)
75    {
76        $prom_2m+= $data_station1[$i][0];
77    }
78    $total_values1=sizeof($data_station1);
79    $prom_2m=$prom_2m/$total_values1;
80
81
82    //echo "Promedio 2m: "; print_r($prom_2m); echo "<br>";
83    //echo "Number of values: "; print_r($total_values1); echo "<br>";
84 //ESTACIÓN 2 ALTURA: 10 METROS
       ////////////////////////////////////////////////////////
85 $result_station2 = mysqli_query($conn, $sql_station2);
86
87
88    $prom_10m=0;
89    $i=0;
90    $data_station2=[[null]];
91    while ($row2 = mysqli_fetch_array($result_station2))
92    {
93      $data_station2[$i] = $row2;
94       $i++;
95    }
96    for($i = 0 ;$i<count($data_station2);$i++)
97    {
98        $prom_10m+= $data_station2[$i][0];
99    }
100   $total_values2=sizeof($data_station2);
101   $prom_10m=$prom_10m/$total_values2;
102
103 //ESTACIÓN 3 ALTURA: 30 METROS
       ////////////////////////////////////////////////////////
104 $result_station3 = mysqli_query($conn, $sql_station3);
105
106
107   $prom_30m=0;
108   $i=0;
109   $data_station3=[[null]];
110   while ($row3 = mysqli_fetch_array($result_station3))
111   {
112     $data_station3[$i] = $row3;
113      $i++;
```

```
114   }
115   for($i = 0 ;$i<count($data_station3);$i++)
116   {
117       $prom_30m+= $data_station3[$i][0];
118   }
119   $total_values3=sizeof($data_station3);
120   $prom_30m=$prom_30m/$total_values3;
121
122 //ESTACIÓN 4 ALTURA: 50 METROS
      ////////////////////////////////////////////////////////
123 $result_station4 = mysqli_query($conn, $sql_station4);
124
125
126   $prom_50m=0;
127   $i=0;
128   $data_station4=[[null]];
129   while ($row4 = mysqli_fetch_array($result_station4))
130   {
131     $data_station4[$i] = $row4;
132      $i++;
133   }
134   for($i = 0 ;$i<count($data_station4);$i++)
135   {
136       $prom_50m+= $data_station4[$i][0];
137   }
138   $total_values4=sizeof($data_station4);
139   $prom_50m=$prom_50m/$total_values4;
140 ?>
141
142 <!DOCTYPE html>
143
144 <html>
145 <head>
146   <meta charset="utf-8">
147 </head>
148   <meta name="viewport" content="width=device-width, initial-scale=1"
      >
149 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
      font-awesome/4.7.0/css/font-awesome.min.css">
150 <?php include "./websiteStyle.html" ?>
151 </head>
152 <body>
153
154 <?php include "./websiteHeader.html" ?>
155
156 <div id="contenedor"></div>
157 <?php
158     // if data is posted, set value to 1, else to 0
159     $Estacion1 = isset($_POST['Estacion1']);
160     if (empty($Estacion1)){
161       $Estacion1=0;
```

```
      }

    $Estacion2 = isset($_POST['Estacion2']);
    if (empty($Estacion2)){
      $Estacion2=0;
    }

    $Estacion3 = isset($_POST['Estacion3']);
    if (empty($Estacion3)){
      $Estacion3=0;
    }

    $Estacion4 = isset($_POST['Estacion4']);
    if (empty($Estacion4)){
      $Estacion4=0;
    }

    if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & ($Estacion4==0)){
      $Estacion1=1;
      $Estacion2=1;
      $Estacion3=1;
      $Estacion4=1;
    }

    $Estacion1=boolval($Estacion1) ? 'true' : 'false';
    $Estacion2=boolval($Estacion2) ? 'true' : 'false';
    $Estacion3=boolval($Estacion3) ? 'true' : 'false';
    $Estacion4=boolval($Estacion4) ? 'true' : 'false';


$date=date_create();
date_add($date,date_interval_create_from_date_string("-7 days"));

?>

<form action="highchartsTempPerf.php" method="POST">
Fecha Inicial: <input type="date" id="StartDate" name="StartDate" value="<?php if (!isset($_POST['StartDate'])) {echo date_format($date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['StartDate'])) ?>" />
Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
<br>

<input type="submit" value="Guardar Cambios" />
</form>

<script>
chartCPU = new Highcharts.Chart({
    chart: {
```

```
208         renderTo: 'contenedor'
209         //defaultSeriesType: 'spline'
210 
211     },
212     rangeSelector : {
213         enabled: false
214     },
215     title: {
216         text: 'Perfil Vertical de la Temperatura'
217     },
218     xAxis: {
219         type: 'line',
220         minPadding: 0.2,
221         maxPadding: 1,
222         //tickPixelInterval: 150,
223         //maxZoom: 20 * 1000,
224 
225     },
226     yAxis: {
227         minPadding: 0.2,
228         maxPadding: 0.2,
229         title: {
230             text: 'Altura (m)',
231             margin: 10
232         }
233     },
234     plotOptions: {
235         line: {
236             dataLabels: {
237                 enabled: false
238             },
239             enableMouseTracking: true
240         }
241     },
242     tooltip: {
243             valueSuffix: ' m',
244             followPointer: false
245         },
246     series: [{
247         name: 'Perfil de la temperatura °C',
248         data: [[<?php echo $prom_2m;?>, 2],
249                 [<?php echo $prom_10m;?>, 10],
250                 [<?php echo $prom_30m;?>, 30],
251                 [<?php echo $prom_50m;?>, 50]]
252     }],
253     credits: {
254             enabled: false
255     }
256 });
257 </script>
258 </body>
259 </html>
```

A.3. Código para el viento

Códigos para la pestaña de viento, con envió de datos y gráficos dinámicos:

```
<?php


  // connect to database
  //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
     wWtdTEvd5', 'DwIOid9JFJ');
  $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');
  // check connection
  if(!$conn){
    echo 'Connection error: ' . mysqli_connect_error();
  }

  $StartDayOffset=0;
    $EndDayOffset=0;
    if (!isset( $_POST['StartDate'])){
        $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
    }
    else {
      $sql_StartDate = new DateTime($_POST['StartDate']);
      $StartDateDiff = date_diff(date_create(),$sql_StartDate);
      $StartDayOffset=$StartDateDiff->days;
      $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
     )";

    }
    if (!isset($_POST['EndDate'])){
        $sql_EndDate="NOW()";
    }
    else {
      $sql_EndDate = new DateTime($_POST['EndDate']);
      $EndDateDiff = date_diff(date_create(),$sql_EndDate);
      $EndDayOffset=$EndDateDiff->days;
      $EndDayOffset=$EndDayOffset-1;
      $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
    }

  // write query for data
  $sql_station1 =
  ' SELECT DateTime, WindSpeedCur AS VelViento1 FROM VP2_0
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
  ';

  $sql_station2 =
  ' SELECT DateTime, WindSpeedCur AS VelViento2 FROM VP2_1
    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
  ';

```

```
46   $sql_station3 =
47   ' SELECT DateTime, WindSpeedCur AS VelViento3 FROM VP2_2
48     WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
49   ';
50
51     $sql_station4 =
52   ' SELECT DateTime, WindSpeedCur AS VelViento4 FROM VP2_3
53     WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
54   ';
55
56
57 //ESTACIÓN 1 ALTURA: 10 METROS
       ////////////////////////////////////////////////////////
58   $result_station1 = mysqli_query($conn, $sql_station1);
59   $data_station1 = array();
60
61   $i=0;
62   while ($row1 = mysqli_fetch_array($result_station1))
63   {
64     $data_station1[$i] = $row1;
65      $i++;
66   }
67   $valoresArray1;
68   $timeArray1;
69
70   for($i = 0 ;$i<count($data_station1);$i++)
71   {
72       $valoresArray1[$i]= $data_station1[$i][1];
73       //OBTENEMOS EL TIMESTAMP
74       $time1= $data_station1[$i][0];
75       $timeArray1[$i] = strtotime($time1)*1000;
76       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
77   }
78 //ESTACIÓN 2 ALTURA: 30 METROS
       ////////////////////////////////////////////////////////
79   $result_station2 = mysqli_query($conn, $sql_station2);
80   $data_station2 = array();
81
82   $i=0;
83   while ($row2 = mysqli_fetch_array($result_station2))
84   {
85     $data_station2[$i] = $row2;
86      $i++;
87   }
88   $valoresArray2;
89   $timeArray2;
90
91   for($i = 0 ;$i<count($data_station2);$i++)
92   {
93       $valoresArray2[$i]= $data_station2[$i][1];
94       //OBTENEMOS EL TIMESTAMP
```

```
95          $time2= $data_station2[$i][0];
96          $timeArray2[$i] = strtotime($time2)*1000;
97          //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
98      }
99  //ESTACIÓN 3 ALTURA: 40 METROS
        ////////////////////////////////////////////////////////
100     $result_station3 = mysqli_query($conn, $sql_station3);
101     $data_station3 = array();
102
103     $i=0;
104     while ($row3 = mysqli_fetch_array($result_station3))
105     {
106       $data_station3[$i] = $row3;
107        $i++;
108     }
109     $valoresArray3;
110     $timeArray3;
111
112     for($i = 0 ;$i<count($data_station3);$i++)
113     {
114         $valoresArray3[$i]= $data_station3[$i][1];
115         //OBTENEMOS EL TIMESTAMP
116         $time3= $data_station3[$i][0];
117         $timeArray3[$i] = strtotime($time3)*1000;
118         //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
119     }
120
121 //ESTACIÓN 4 ALTURA: 50 METROS
        ////////////////////////////////////////////////////////
122     $result_station4 = mysqli_query($conn, $sql_station4);
123     $data_station4 = array();
124
125     $i=0;
126     while ($row4 = mysqli_fetch_array($result_station4))
127     {
128       $data_station4[$i] = $row4;
129        $i++;
130     }
131     $valoresArray4;
132     $timeArray4;
133
134     for($i = 0 ;$i<count($data_station4);$i++)
135     {
136         $valoresArray4[$i]= $data_station4[$i][1];
137         //OBTENEMOS EL TIMESTAMP
138         $time4= $data_station4[$i][0];
139         $timeArray4[$i] = strtotime($time4)*1000;
140         //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
141     }
142
143 ?>
```

```
144   <!DOCTYPE html>
145 <html>
146 <head>
147   <meta charset="utf-8">
148 </head>
149 <body>
150
151 <meta name="viewport" content="width=device-width, initial-scale=1">
152 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
      font-awesome/4.7.0/css/font-awesome.min.css">
153 <?php include "./websiteStyle.html" ?>
154 </head>
155 <body>
156
157 <?php include "./websiteHeader.html" ?>
158
159 <div id="contenedor"></div>
160
161 <?php
162     // if data is posted, set value to 1, else to 0
163     $Estacion1 = isset($_POST['Estacion1']);
164     if (empty($Estacion1)){
165       $Estacion1=0;
166       }
167
168     $Estacion2 = isset($_POST['Estacion2']);
169     if (empty($Estacion2)){
170       $Estacion2=0;
171     }
172
173     $Estacion3 = isset($_POST['Estacion3']);
174     if (empty($Estacion3)){
175       $Estacion3=0;
176     }
177
178     $Estacion4 = isset($_POST['Estacion4']);
179     if (empty($Estacion4)){
180       $Estacion4=0;
181     }
182
183     if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & (
      $Estacion4==0)){
184       $Estacion1=1;
185       $Estacion2=1;
186       $Estacion3=1;
187       $Estacion4=1;
188     }
189
190     $Estacion1=boolval($Estacion1) ? 'true' : 'false';
191     $Estacion2=boolval($Estacion2) ? 'true' : 'false';
192     $Estacion3=boolval($Estacion3) ? 'true' : 'false';
193     $Estacion4=boolval($Estacion4) ? 'true' : 'false';
```

```
194
195
196 $date=date_create();
197 date_add($date,date_interval_create_from_date_string("-7 days"));
198
199 ?>
200
201 <form action="highchartsWindSpd.php" method="POST">
202 Estación 2m: <input type="checkbox" name="Estacion1" value="true" <?
      php if(isset($_POST['Estacion1'])) echo 'checked="checked"';
      ?>/>
203 Estación 10m : <input type="checkbox" name="Estacion2" value="true"
      <?php if(isset($_POST['Estacion2'])) echo 'checked="checked"';
      ?>/>
204 Estación 30m: <input type="checkbox" name="Estacion3" value="true" <?
      php if(isset($_POST['Estacion3'])) echo 'checked="checked"';
      ?>/>
205 Estación 50m : <input type="checkbox" name="Estacion4" value="true"
      <?php if(isset($_POST['Estacion4'])) echo 'checked="checked"';
      ?>/>
206 <br>
207
208 Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
      value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
      $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
      StartDate'])) ?>" />
209 Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
      php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
      echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
210 <br>
211
212 <input type="submit" value="Guardar Cambios" />
213 </form>
214
215 <script>
216
217 chartCPU = new Highcharts.Chart({
218     chart: {
219         renderTo: 'contenedor'
220         //defaultSeriesType: 'spline'
221
222     },
223     rangeSelector : {
224         enabled: false
225     },
226     title: {
227         text: 'Velocidad de Viento Instantánea Estaciones Vantage Pro
      2'
228     },
229     xAxis: {
230         type: 'datetime'
231         //tickPixelInterval: 150,
```

```
232         //maxZoom: 20 * 1000
233     },
234     yAxis: {
235         minPadding: 0.2,
236         maxPadding: 0.2,
237         title: {
238             text: 'Velocidad de Viento m/s',
239             margin: 10
240         }
241     },
242     tooltip: {
243             valueSuffix: ' m/s',
244             followPointer: true
245         },
246     series: [{
247         name: 'Vel. Viento (2m)',
248         data: (function () {
249
250
251                 var data = [];
252                 <?php
253                     for($i = 0 ;$i<count($data_station1);$i++){
254                 ?>
255                 data.push([<?php echo $timeArray1[$i];?>,<?php echo
     $valoresArray1[$i];?>]);
256                 <?php } ?>
257                 return data;
258             })(),
259         visible:<?php echo $Estacion1;?>,
260         showInLegend:<?php echo $Estacion1;?>
261     },{
262         name: 'Vel. Viento (10m)',
263         data: (function () {
264
265
266                 var data = [];
267                 <?php
268                     for($i = 0 ;$i<count($data_station2);$i++){
269                 ?>
270                 data.push([<?php echo $timeArray2[$i];?>,<?php echo
     $valoresArray2[$i];?>]);
271                 <?php } ?>
272                 return data;
273             })(),
274         visible:<?php echo $Estacion2;?>,
275         showInLegend:<?php echo $Estacion2;?>
276     },{
277         name: 'Vel. Viento (30m)',
278         data: (function () {
279
280
281                 var data = [];
```

```
                <?php
                    for($i = 0 ;$i<count($data_station3);$i++){
                ?>
                data.push([<?php echo $timeArray3[$i];?>,<?php echo
     $valoresArray3[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion3;?>,
        showInLegend:<?php echo $Estacion3;?>
    },{
        name: 'Vel. Viento (50m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station4);$i++){
                ?>
                data.push([<?php echo $timeArray4[$i];?>,<?php echo
     $valoresArray4[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion4;?>,
        showInLegend:<?php echo $Estacion4;?>
    }],
    credits: {
            enabled: false
    }
});

</script>
</body>
</html>
```

```
<?php


  // connect to database
  //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
     wWtdTEvd5', 'DwIOid9JFJ');
  $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');
  // check connection
  if(!$conn){
    echo 'Connection error: ' . mysqli_connect_error();
  }

  $StartDayOffset=0;
    $EndDayOffset=0;
    if (!isset( $_POST['StartDate'])){
```

```
15          $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
16      }
17      else {
18        $sql_StartDate = new DateTime($_POST['StartDate']);
19        $StartDateDiff = date_diff(date_create(),$sql_StartDate);
20        $StartDayOffset=$StartDateDiff->days;
21        $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
     )";
22
23      }
24      if (!isset($_POST['EndDate'])){
25          $sql_EndDate="NOW()";
26      }
27      else {
28        $sql_EndDate = new DateTime($_POST['EndDate']);
29        $EndDateDiff = date_diff(date_create(),$sql_EndDate);
30        $EndDayOffset=$EndDateDiff->days;
31        $EndDayOffset=$EndDayOffset-1;
32        $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";
33      }
34
35
36    // write query for data
37    $sql_station1 =
38    ' SELECT DateTime, WindAvgSpeedCur AS VelVientoProm1 FROM VP2_0
39      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
40    ';
41
42    $sql_station2 =
43    ' SELECT DateTime, WindAvgSpeedCur AS VelVientoProm2 FROM VP2_1
44      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
45    ';
46
47    $sql_station3 =
48    ' SELECT DateTime, WindAvgSpeedCur AS VelVientoProm3 FROM VP2_2
49      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
50    ';
51
52      $sql_station4 =
53    ' SELECT DateTime, WindAvgSpeedCur AS VelVientoProm4 FROM VP2_3
54      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
55    ';
56
57
58  //ESTACIÓN 1 ALTURA: 10 METROS
     ////////////////////////////////////////////////////////////
59    $result_station1 = mysqli_query($conn, $sql_station1);
60    $data_station1 = array();
61
62    $i=0;
63    while ($row1 = mysqli_fetch_array($result_station1))
```

```
64   {
65     $data_station1[$i] = $row1;
66      $i++;
67   }
68   $valoresArray1;
69   $timeArray1;
70
71   for($i = 0 ;$i<count($data_station1);$i++)
72   {
73       $valoresArray1[$i]= $data_station1[$i][1];
74       //OBTENEMOS EL TIMESTAMP
75       $time1= $data_station1[$i][0];
76       $timeArray1[$i] = strtotime($time1)*1000;
77       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
78   }
79 //ESTACIÓN 2 ALTURA: 30 METROS
       ////////////////////////////////////////////////////////
80   $result_station2 = mysqli_query($conn, $sql_station2);
81   $data_station2 = array();
82
83   $i=0;
84   while ($row2 = mysqli_fetch_array($result_station2))
85   {
86     $data_station2[$i] = $row2;
87      $i++;
88   }
89   $valoresArray2;
90   $timeArray2;
91
92   for($i = 0 ;$i<count($data_station2);$i++)
93   {
94       $valoresArray2[$i]= $data_station2[$i][1];
95       //OBTENEMOS EL TIMESTAMP
96       $time2= $data_station2[$i][0];
97       $timeArray2[$i] = strtotime($time2)*1000;
98       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
99   }
100 //ESTACIÓN 3 ALTURA: 40 METROS
       ////////////////////////////////////////////////////////
101  $result_station3 = mysqli_query($conn, $sql_station3);
102  $data_station3 = array();
103
104  $i=0;
105  while ($row3 = mysqli_fetch_array($result_station3))
106  {
107    $data_station3[$i] = $row3;
108     $i++;
109  }
110  $valoresArray3;
111  $timeArray3;
112
```

```
113   for($i = 0 ;$i<count($data_station3);$i++)
114   {
115       $valoresArray3[$i]= $data_station3[$i][1];
116       //OBTENEMOS EL TIMESTAMP
117       $time3= $data_station3[$i][0];
118       $timeArray3[$i] = strtotime($time3)*1000;
119       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
120   }
121
122 //ESTACIÓN 4 ALTURA: 50 METROS
      ////////////////////////////////////////////////////////////
123   $result_station4 = mysqli_query($conn, $sql_station4);
124   $data_station4 = array();
125
126   $i=0;
127   while ($row4 = mysqli_fetch_array($result_station4))
128   {
129     $data_station4[$i] = $row4;
130      $i++;
131   }
132   $valoresArray4;
133   $timeArray4;
134
135   for($i = 0 ;$i<count($data_station4);$i++)
136   {
137       $valoresArray4[$i]= $data_station4[$i][1];
138       //OBTENEMOS EL TIMESTAMP
139       $time4= $data_station4[$i][0];
140       $timeArray4[$i] = strtotime($time4)*1000;
141       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
142   }
143
144 ?>
145 <!DOCTYPE html>
146 <html>
147 <head>
148   <meta charset="utf-8">
149 </head>
150 <body>
151
152
153 <meta name="viewport" content="width=device-width, initial-scale=1">
154 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
      font-awesome/4.7.0/css/font-awesome.min.css">
155 <?php include "./websiteStyle.html" ?>
156 </head>
157 <body>
158
159 <?php include "./websiteHeader.html" ?>
160
161 <div id="contenedor"></div>
```

```
<?php
    // if data is posted, set value to 1, else to 0
    $Estacion1 = isset($_POST['Estacion1']);
    if (empty($Estacion1)){
      $Estacion1=0;
      }

    $Estacion2 = isset($_POST['Estacion2']);
    if (empty($Estacion2)){
      $Estacion2=0;
    }

    $Estacion3 = isset($_POST['Estacion3']);
    if (empty($Estacion3)){
      $Estacion3=0;
    }

    $Estacion4 = isset($_POST['Estacion4']);
    if (empty($Estacion4)){
      $Estacion4=0;
    }

    if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & ($Estacion4==0)){
      $Estacion1=1;
      $Estacion2=1;
      $Estacion3=1;
      $Estacion4=1;
    }

    $Estacion1=boolval($Estacion1) ? 'true' : 'false';
    $Estacion2=boolval($Estacion2) ? 'true' : 'false';
    $Estacion3=boolval($Estacion3) ? 'true' : 'false';
    $Estacion4=boolval($Estacion4) ? 'true' : 'false';


$date=date_create();
date_add($date,date_interval_create_from_date_string("-7 days"));

?>

<form action="highchartsWindSpdAvg.php" method="POST">
Estación 2m: <input type="checkbox" name="Estacion1" value="true" <?php if(isset($_POST['Estacion1'])) echo 'checked="checked"'; ?>/>
Estación 10m : <input type="checkbox" name="Estacion2" value="true" <?php if(isset($_POST['Estacion2'])) echo 'checked="checked"'; ?>/>
Estación 30m: <input type="checkbox" name="Estacion3" value="true" <?php if(isset($_POST['Estacion3'])) echo 'checked="checked"'; ?>/>
```

```
Estación 50m : <input type="checkbox" name="Estacion4" value="true"
     <?php if(isset($_POST['Estacion4'])) echo 'checked="checked"';
     ?>/>
<br>

Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
     value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
     $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
     StartDate'])) ?>" />
Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
     php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
     echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
<br>

<input type="submit" value="Guardar Cambios" />
</form>

<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Velocidad de Viento Promedio Estaciones Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime'
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Velocidad de Viento Promedio m/s',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' m/s',
            followPointer: true
        },
    series: [{
        name: 'Vel. Viento Prom. (2m)',
        data: (function () {


```

```
                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station1);$i++){
                ?>
                data.push([<?php echo $timeArray1[$i];?>,<?php echo $valoresArray1[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion1;?>,
        showInLegend:<?php echo $Estacion1;?>
    },{
        name: 'Vel. Viento Prom. (10m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station2);$i++){
                ?>
                data.push([<?php echo $timeArray2[$i];?>,<?php echo $valoresArray2[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion2;?>,
        showInLegend:<?php echo $Estacion2;?>
    },{
        name: 'Vel. Viento Prom. (30m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station3);$i++){
                ?>
                data.push([<?php echo $timeArray3[$i];?>,<?php echo $valoresArray3[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion3;?>,
        showInLegend:<?php echo $Estacion3;?>
    },{
        name: 'Vel. Viento Prom. (50m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station4);$i++){
                ?>
```

```
                data.push([<?php echo $timeArray4[$i];?>,<?php echo
    $valoresArray4[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion4;?>,
        showInLegend:<?php echo $Estacion4;?>
    },
    credits: {
            enabled: false
    }
});
</script>
</body>
</html>
```

```
<?php

  mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
  // connect to database
  //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
    wWtdTEvd5', 'DwIOid9JFJ');
  $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');



    $StartDayOffset=0;
    $EndDayOffset=0;
    if (!isset( $_POST['StartDate'])){
        $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
    }
    else {
      $sql_StartDate = new DateTime($_POST['StartDate']);
      $StartDateDiff = date_diff(date_create(),$sql_StartDate);
      $StartDayOffset=$StartDateDiff->days;
      $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
    )";

    }
    if (!isset($_POST['EndDate'])){
        $sql_EndDate="NOW()";
    }
    else {
      $sql_EndDate = new DateTime($_POST['EndDate']);
      $EndDateDiff = date_diff(date_create(),$sql_EndDate);
      $EndDayOffset=$EndDateDiff->days;
      $EndDayOffset=$EndDayOffset-1;
      $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
    }

  // check connection
```

```
34  if(!$conn){
35    echo 'Connection error: ' . mysqli_connect_error();
36  }
37
38
39  // write query for data
40
41  $sql_station1 =
42  ' SELECT WindAvgSpeedCur FROM VP2_0
43    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
44  ';
45
46  $sql_station2 =
47  ' SELECT WindAvgSpeedCur FROM VP2_1
48    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
49  ';
50
51  $sql_station3 =
52  ' SELECT WindAvgSpeedCur FROM VP2_2
53    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
54  ';
55
56  $sql_station4 =
57  ' SELECT WindAvgSpeedCur FROM VP2_3
58    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
59  ';
60
61
62 //ESTACIÓN 1 ALTURA: 2 METROS
       ////////////////////////////////////////////////////////////
63  $result_station1 = mysqli_query($conn, $sql_station1);
64
65
66  $prom_2m=0;
67  $i=0;
68  $data_station1=[[null]];
69  while ($row1 = mysqli_fetch_array($result_station1))
70  {
71    $data_station1[$i] = $row1;
72     $i++;
73  }
74  for($i = 0 ;$i<count($data_station1);$i++)
75  {
76      $prom_2m+= $data_station1[$i][0];
77  }
78  $total_values1=sizeof($data_station1);
79  $prom_2m=$prom_2m/$total_values1;
80
81
82 //ESTACIÓN 2 ALTURA: 10 METROS
       ////////////////////////////////////////////////////////////
```

```
83 $result_station2 = mysqli_query($conn, $sql_station2);
84
85
86   $prom_10m=0;
87   $i=0;
88   $data_station2=[[null]];
89   while ($row2 = mysqli_fetch_array($result_station2))
90   {
91     $data_station2[$i] = $row2;
92      $i++;
93   }
94   for($i = 0 ;$i<count($data_station2);$i++)
95   {
96       $prom_10m+= $data_station2[$i][0];
97   }
98   $total_values2=sizeof($data_station2);
99   $prom_10m=$prom_10m/$total_values2;
100
101 //ESTACIÓN 3 ALTURA: 30 METROS
        ////////////////////////////////////////////////////////
102 $result_station3 = mysqli_query($conn, $sql_station3);
103
104
105   $prom_30m=0;
106   $i=0;
107   $data_station3=[[null]];
108   while ($row3 = mysqli_fetch_array($result_station3))
109   {
110     $data_station3[$i] = $row3;
111      $i++;
112   }
113   for($i = 0 ;$i<count($data_station3);$i++)
114   {
115       $prom_30m+= $data_station3[$i][0];
116   }
117   $total_values3=sizeof($data_station3);
118   $prom_30m=$prom_30m/$total_values3;
119
120 //ESTACIÓN 4 ALTURA: 50 METROS
        ////////////////////////////////////////////////////////
121 $result_station4 = mysqli_query($conn, $sql_station4);
122
123
124   $prom_50m=0;
125   $i=0;
126   $data_station4=[[null]];
127   while ($row4 = mysqli_fetch_array($result_station4))
128   {
129     $data_station4[$i] = $row4;
130      $i++;
131   }
```

```
  for($i = 0 ;$i<count($data_station4);$i++)
  {
      $prom_50m+= $data_station4[$i][0];
  }
  $total_values4=sizeof($data_station4);
  $prom_50m=$prom_50m/$total_values4;
?>

<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8">
</head>
  <meta name="viewport" content="width=device-width, initial-scale=1"
    >
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
    font-awesome/4.7.0/css/font-awesome.min.css">
<?php include "./websiteStyle.html" ?>
</head>
<body>

<?php include "./websiteHeader.html" ?>

<div id="contenedor"></div>
<?php
    // if data is posted, set value to 1, else to 0
    $Estacion1 = isset($_POST['Estacion1']);
    if (empty($Estacion1)){
      $Estacion1=0;
      }

    $Estacion2 = isset($_POST['Estacion2']);
    if (empty($Estacion2)){
      $Estacion2=0;
    }

    $Estacion3 = isset($_POST['Estacion3']);
    if (empty($Estacion3)){
      $Estacion3=0;
    }

    $Estacion4 = isset($_POST['Estacion4']);
    if (empty($Estacion4)){
      $Estacion4=0;
    }

    if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & (
     $Estacion4==0)){
      $Estacion1=1;
      $Estacion2=1;
      $Estacion3=1;
```

```
181         $Estacion4=1;
182     }
183 
184     $Estacion1=boolval($Estacion1) ? 'true' : 'false';
185     $Estacion2=boolval($Estacion2) ? 'true' : 'false';
186     $Estacion3=boolval($Estacion3) ? 'true' : 'false';
187     $Estacion4=boolval($Estacion4) ? 'true' : 'false';
188 
189 
190 $date=date_create();
191 date_add($date,date_interval_create_from_date_string("-7 days"));
192 
193 
194 ?>
195 
196 <form action="highchartsWindPerf.php" method="POST">
197 Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
      value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
      $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
      StartDate'])) ?>" />
198 Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
      php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
      echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
199 <br>
200 
201 
202 
203 <input type="submit" value="Guardar Cambios" />
204 </form>
205 
206 <script>
207 chartCPU = new Highcharts.Chart({
208     chart: {
209         renderTo: 'contenedor'
210         //defaultSeriesType: 'spline'
211     },
212     rangeSelector : {
213         enabled: false
214     },
215     title: {
216         text: 'Perfil del Viento'
217     },
218     xAxis: {
219         type: 'line',
220         minPadding: 0.2,
221         maxPadding: 1,
222         //tickPixelInterval: 150,
223         //maxZoom: 20 * 1000,
224     },
225     yAxis: {
226         minPadding: 0.2,
227         maxPadding: 0.2,
```

```
228         title: {
229             text: 'Altura (m)',
230             margin: 10
231         }
232     },
233     plotOptions: {
234         line: {
235             dataLabels: {
236                 enabled: false
237             },
238             enableMouseTracking: true
239         }
240     },
241     tooltip: {
242             valueSuffix: ' m',
243             followPointer: false
244         },
245     series: [{
246         name: 'Perfil del viento (m/s)',
247         data: [[<?php echo $prom_2m;?>, 2],
248                 [<?php echo $prom_10m;?>, 10],
249                 [<?php echo $prom_30m;?>, 30],
250                 [<?php echo $prom_50m;?>, 50]]
251     }],
252     credits: {
253             enabled: false
254     }
255 });
256 </script>
257 </body>
258 </html>
```

A.4. Código para la humedad

Códigos para la pestaña de humedad, con envió de datos y gráficos dinámicos:

```
1  <?php
2
3
4    // connect to database
5    //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
       wWtdTEvd5', 'DwIOid9JFJ');
6    $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
       'Cdfuzpvpan', 'sql10439029');
7    // check connection
8    if(!$conn){
9      echo 'Connection error: ' . mysqli_connect_error();
10   }
```

```
11
12      $StartDayOffset=0;
13      $EndDayOffset=0;
14      if (!isset( $_POST['StartDate'])){
15          $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
16      }
17      else {
18        $sql_StartDate = new DateTime($_POST['StartDate']);
19        $StartDateDiff = date_diff(date_create(),$sql_StartDate);
20        $StartDayOffset=$StartDateDiff->days;
21        $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
     )";
22
23      }
24      if (!isset($_POST['EndDate'])){
25          $sql_EndDate="NOW()";
26      }
27      else {
28        $sql_EndDate = new DateTime($_POST['EndDate']);
29        $EndDateDiff = date_diff(date_create(),$sql_EndDate);
30        $EndDayOffset=$EndDateDiff->days;
31        $EndDayOffset=$EndDayOffset-1;
32        $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
33      }
34
35    // write query for data
36    $sql_station1 =
37    ' SELECT DateTime, HumOutCur AS Humedad1 FROM VP2_0
38      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
39    ';
40
41    $sql_station2 =
42    ' SELECT DateTime, HumOutCur AS Humedad2 FROM VP2_1
43      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
44    ';
45
46    $sql_station3 =
47    ' SELECT DateTime, HumOutCur AS Humedad3 FROM VP2_2
48      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
49    ';
50
51      $sql_station4 =
52    ' SELECT DateTime, HumOutCur AS Humedad4 FROM VP2_3
53      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
54    ';
55
56
57  //ESTACIÓN 1 ALTURA: 2 METROS
     ////////////////////////////////////////////////////////////
58    $result_station1 = mysqli_query($conn, $sql_station1);
59    $data_station1 = array();
```

```
60
61  $i=0;
62  while ($row1 = mysqli_fetch_array($result_station1))
63  {
64    $data_station1[$i] = $row1;
65     $i++;
66  }
67  $valoresArray1;
68  $timeArray1;
69
70  for($i = 0 ;$i<count($data_station1);$i++)
71  {
72      $valoresArray1[$i]= $data_station1[$i][1];
73      //OBTENEMOS EL TIMESTAMP
74      $time1= $data_station1[$i][0];
75      $timeArray1[$i] = strtotime($time1)*1000;
76      //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
77  }
78 //ESTACIÓN 2 ALTURA: 10 METROS
       ////////////////////////////////////////////////////////
79  $result_station2 = mysqli_query($conn, $sql_station2);
80  $data_station2 = array();
81
82  $i=0;
83  while ($row2 = mysqli_fetch_array($result_station2))
84  {
85    $data_station2[$i] = $row2;
86     $i++;
87  }
88  $valoresArray2;
89  $timeArray2;
90
91  for($i = 0 ;$i<count($data_station2);$i++)
92  {
93      $valoresArray2[$i]= $data_station2[$i][1];
94      //OBTENEMOS EL TIMESTAMP
95      $time2= $data_station2[$i][0];
96      $timeArray2[$i] = strtotime($time2)*1000;
97      //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
98  }
99 //ESTACIÓN 3 ALTURA: 30 METROS
       ////////////////////////////////////////////////////////
100  $result_station3 = mysqli_query($conn, $sql_station3);
101  $data_station3 = array();
102
103  $i=0;
104  while ($row3 = mysqli_fetch_array($result_station3))
105  {
106    $data_station3[$i] = $row3;
107     $i++;
108  }
```

```
109   $valoresArray3;
110   $timeArray3;
111
112   for($i = 0 ;$i<count($data_station3);$i++)
113   {
114       $valoresArray3[$i]= $data_station3[$i][1];
115       //OBTENEMOS EL TIMESTAMP
116       $time3= $data_station3[$i][0];
117       $timeArray3[$i] = strtotime($time3)*1000;
118       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
119   }
120
121 //ESTACIÓN 4 ALTURA: 50 METROS
      ////////////////////////////////////////////////////////
122   $result_station4 = mysqli_query($conn, $sql_station4);
123   $data_station4 = array();
124
125   $i=0;
126   while ($row4 = mysqli_fetch_array($result_station4))
127   {
128     $data_station4[$i] = $row4;
129      $i++;
130   }
131   $valoresArray4;
132   $timeArray4;
133
134   for($i = 0 ;$i<count($data_station4);$i++)
135   {
136       $valoresArray4[$i]= $data_station4[$i][1];
137       //OBTENEMOS EL TIMESTAMP
138       $time4= $data_station4[$i][0];
139       $timeArray4[$i] = strtotime($time4)*1000;
140       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
141   }
142
143 ?>
144   <!DOCTYPE html>
145 <head>
146   <meta charset="utf-8">
147 </head>
148   <meta name="viewport" content="width=device-width, initial-scale=1"
      >
149 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
      font-awesome/4.7.0/css/font-awesome.min.css">
150 <?php include "./websiteStyle.html" ?>
151 </head>
152 <body>
153
154 <?php include "./websiteHeader.html" ?>
155
156 <div id="contenedor"></div>
```

```
157
158
159 <?php
160     // if data is posted, set value to 1, else to 0
161     $Estacion1 = isset($_POST['Estacion1']);
162     if (empty($Estacion1)){
163       $Estacion1=0;
164       }
165
166     $Estacion2 = isset($_POST['Estacion2']);
167     if (empty($Estacion2)){
168       $Estacion2=0;
169     }
170
171     $Estacion3 = isset($_POST['Estacion3']);
172     if (empty($Estacion3)){
173       $Estacion3=0;
174     }
175
176     $Estacion4 = isset($_POST['Estacion4']);
177     if (empty($Estacion4)){
178       $Estacion4=0;
179     }
180
181     if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & (
         $Estacion4==0)){
182       $Estacion1=1;
183       $Estacion2=1;
184       $Estacion3=1;
185       $Estacion4=1;
186     }
187
188     $Estacion1=boolval($Estacion1) ? 'true' : 'false';
189     $Estacion2=boolval($Estacion2) ? 'true' : 'false';
190     $Estacion3=boolval($Estacion3) ? 'true' : 'false';
191     $Estacion4=boolval($Estacion4) ? 'true' : 'false';
192
193
194 $date=date_create();
195 date_add($date,date_interval_create_from_date_string("-7 days"));
196
197 ?>
198
199 <form action="highchartsHum.php" method="POST">
200 Estación 2m: <input type="checkbox" name="Estacion1" value="true" <?
         php if(isset($_POST['Estacion1'])) echo 'checked="checked"';
         ?>/>
201 Estación 10m : <input type="checkbox" name="Estacion2" value="true"
         <?php if(isset($_POST['Estacion2'])) echo 'checked="checked"';
         ?>/>
202 Estación 30m: <input type="checkbox" name="Estacion3" value="true" <?
         php if(isset($_POST['Estacion3'])) echo 'checked="checked"';
```

```
    ?>/>
Estación 50m : <input type="checkbox" name="Estacion4" value="true"
    <?php if(isset($_POST['Estacion4'])) echo 'checked="checked"';
    ?>/>
<br>

Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
    value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
    $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
    StartDate'])) ?>" />
Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
    php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
    echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
<br>

<input type="submit" value="Guardar Cambios" />
</form>

<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Humedad Estaciones Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime'
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Humedad %',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' %',
            followPointer: true
        },
    series: [{
        name: 'Humedad (2m)',
        data: (function () {

```

```
247
248                 var data = [];
249                 <?php
250                     for($i = 0 ;$i<count($data_station1);$i++){
251                 ?>
252                 data.push([<?php echo $timeArray1[$i];?>,<?php echo
     $valoresArray1[$i];?>]);
253                 <?php } ?>
254                 return data;
255             })(),
256         visible:<?php echo $Estacion1;?>,
257         showInLegend:<?php echo $Estacion1;?>
258     },{
259         name: 'Humedad (10m)',
260         data: (function () {
261
262
263                 var data = [];
264                 <?php
265                     for($i = 0 ;$i<count($data_station2);$i++){
266                 ?>
267                 data.push([<?php echo $timeArray2[$i];?>,<?php echo
     $valoresArray2[$i];?>]);
268                 <?php } ?>
269                 return data;
270             })(),
271         visible:<?php echo $Estacion2;?>,
272         showInLegend:<?php echo $Estacion2;?>
273     },{
274         name: 'Humedad (30m)',
275         data: (function () {
276
277
278                 var data = [];
279                 <?php
280                     for($i = 0 ;$i<count($data_station3);$i++){
281                 ?>
282                 data.push([<?php echo $timeArray3[$i];?>,<?php echo
     $valoresArray3[$i];?>]);
283                 <?php } ?>
284                 return data;
285             })(),
286         visible:<?php echo $Estacion3;?>,
287         showInLegend:<?php echo $Estacion3;?>
288     },{
289         name: 'Humedad (50m)',
290         data: (function () {
291
292
293                 var data = [];
294                 <?php
295                     for($i = 0 ;$i<count($data_station4);$i++){
```

```
                ?>
                data.push([<?php echo $timeArray4[$i];?>,<?php echo
    $valoresArray4[$i];?>]);
                <?php } ?>
                return data;
            })(),
        visible:<?php echo $Estacion4;?>,
        showInLegend:<?php echo $Estacion4;?>
    }],
    credits: {
            enabled: false
    }
});
</script>
  </body>
  </html>
```

```
<?php

  mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
  // connect to database
  //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
    wWtdTEvd5', 'DwIOid9JFJ');
  $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');



    $StartDayOffset=0;
    $EndDayOffset=0;
    if (!isset( $_POST['StartDate'])){
        $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
    }
    else {
      $sql_StartDate = new DateTime($_POST['StartDate']);
      $StartDateDiff = date_diff(date_create(),$sql_StartDate);
      $StartDayOffset=$StartDateDiff->days;
      $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
     )";

    }
    if (!isset($_POST['EndDate'])){
        $sql_EndDate="NOW()";
    }
    else {
      $sql_EndDate = new DateTime($_POST['EndDate']);
      $EndDateDiff = date_diff(date_create(),$sql_EndDate);
      $EndDayOffset=$EndDateDiff->days;
      $EndDayOffset=$EndDayOffset-1;
      $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
    }

```

```
33  // check connection
34  if(!$conn){
35    echo 'Connection error: ' . mysqli_connect_error();
36  }
37
38
39  // write query for data
40
41  $sql_station1 =
42  ' SELECT HumOutCur FROM VP2_0
43    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
44  ';
45
46  $sql_station2 =
47  ' SELECT HumOutCur FROM VP2_1
48    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
49  ';
50
51  $sql_station3 =
52  ' SELECT HumOutCur FROM VP2_2
53    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
54  ';
55
56  $sql_station4 =
57  ' SELECT HumOutCur FROM VP2_3
58    WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
59  ';
60
61 //ESTACIÓN 1 ALTURA: 2 METROS
       ////////////////////////////////////////////////////////
62  $result_station1 = mysqli_query($conn, $sql_station1);
63
64
65  $prom_2m=0;
66  $i=0;
67  $data_station1=[[null]];
68  while ($row1 = mysqli_fetch_array($result_station1))
69  {
70    $data_station1[$i] = $row1;
71     $i++;
72  }
73  for($i = 0 ;$i<count($data_station1);$i++)
74  {
75      $prom_2m+= $data_station1[$i][0];
76  }
77  $total_values1=sizeof($data_station1);
78  $prom_2m=$prom_2m/$total_values1;
79
80 //ESTACIÓN 2 ALTURA: 10 METROS
       ////////////////////////////////////////////////////////
81 $result_station2 = mysqli_query($conn, $sql_station2);
```

```
82
83
84    $prom_10m=0;
85    $i=0;
86    $data_station2=[[null]];
87    while ($row2 = mysqli_fetch_array($result_station2))
88    {
89      $data_station2[$i] = $row2;
90       $i++;
91    }
92    for($i = 0 ;$i<count($data_station2);$i++)
93    {
94        $prom_10m+= $data_station2[$i][0];
95    }
96    $total_values2=sizeof($data_station2);
97    $prom_10m=$prom_10m/$total_values2;
98
99  //ESTACIÓN 3 ALTURA: 30 METROS
        ////////////////////////////////////////////////////////
100 $result_station3 = mysqli_query($conn, $sql_station3);
101
102
103   $prom_30m=0;
104   $i=0;
105   $data_station3=[[null]];
106   while ($row3 = mysqli_fetch_array($result_station3))
107   {
108     $data_station3[$i] = $row3;
109      $i++;
110   }
111   for($i = 0 ;$i<count($data_station3);$i++)
112   {
113       $prom_30m+= $data_station3[$i][0];
114   }
115   $total_values3=sizeof($data_station3);
116   $prom_30m=$prom_30m/$total_values3;
117
118 //ESTACIÓN 4 ALTURA: 50 METROS
        ////////////////////////////////////////////////////////
119 $result_station4 = mysqli_query($conn, $sql_station4);
120
121
122   $prom_50m=0;
123   $i=0;
124   $data_station4=[[null]];
125   while ($row4 = mysqli_fetch_array($result_station4))
126   {
127     $data_station4[$i] = $row4;
128      $i++;
129   }
130   for($i = 0 ;$i<count($data_station4);$i++)
```

```
  {
      $prom_50m+= $data_station4[$i][0];
  }
  $total_values4=sizeof($data_station4);
  $prom_50m=$prom_50m/$total_values4;
?>

<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8">
</head>
  <meta name="viewport" content="width=device-width, initial-scale=1"
    >
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
    font-awesome/4.7.0/css/font-awesome.min.css">
<?php include "./websiteStyle.html" ?>
</head>
<body>

<?php include "./websiteHeader.html" ?>

<div id="contenedor"></div>
<?php
    // if data is posted, set value to 1, else to 0
    $Estacion1 = isset($_POST['Estacion1']);
    if (empty($Estacion1)){
      $Estacion1=0;
      }

    $Estacion2 = isset($_POST['Estacion2']);
    if (empty($Estacion2)){
      $Estacion2=0;
    }

    $Estacion3 = isset($_POST['Estacion3']);
    if (empty($Estacion3)){
      $Estacion3=0;
    }

    $Estacion4 = isset($_POST['Estacion4']);
    if (empty($Estacion4)){
      $Estacion4=0;
    }

    if (($Estacion1==0) & ($Estacion2==0) & ($Estacion3==0) & (
    $Estacion4==0)){
      $Estacion1=1;
      $Estacion2=1;
      $Estacion3=1;
      $Estacion4=1;
```

```
    }

    $Estacion1=boolval($Estacion1) ? 'true' : 'false';
    $Estacion2=boolval($Estacion2) ? 'true' : 'false';
    $Estacion3=boolval($Estacion3) ? 'true' : 'false';
    $Estacion4=boolval($Estacion4) ? 'true' : 'false';


$date=date_create();
date_add($date,date_interval_create_from_date_string("-7 days"));


?>

<form action="highchartsHumPerf.php" method="POST">
Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
     value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
     $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
     StartDate'])) ?>" />
Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
     php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
     echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
<br>



<input type="submit" value="Guardar Cambios" />
</form>

<script>

chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Perfil Vertical de la Humedad'
    },
    xAxis: {
        type: 'line',
        minPadding: 0.2,
        maxPadding: 1,
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000,

    },
    yAxis: {
```

```
227         minPadding: 0.2,
228         maxPadding: 0.2,
229         title: {
230             text: 'Altura (m)',
231             margin: 10
232         }
233     },
234     plotOptions: {
235         line: {
236             dataLabels: {
237                 enabled: false
238             },
239             enableMouseTracking: true
240         }
241     },
242     tooltip: {
243             valueSuffix: ' m',
244             followPointer: false
245         },
246     series: [{
247         name: 'Perfil de la humedad %',
248         data: [[<?php echo $prom_2m;?>, 2],
249                 [<?php echo $prom_10m;?>, 10],
250                 [<?php echo $prom_30m;?>, 30],
251                 [<?php echo $prom_50m;?>, 50]]
252     }],
253     credits: {
254             enabled: false
255     }
256 });
257
258 </script>
259 </body>
260 </html>
```

A.5. Código para la presión

Código para la pestaña de presión, estación a dos metros de altura:

```
1 <?php
2
3   mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
4   // connect to database
5   //$conn = mysqli_connect('remotemysql.com', 'DwIOid9JFJ', '8
      wWtdTEvd5', 'DwIOid9JFJ');
6   $conn = mysqli_connect('sql10.freemysqlhosting.net', 'sql10439029',
      'Cdfuzpvpan', 'sql10439029');
7
8
```

```
9       $StartDayOffset=0;
10      $EndDayOffset=0;
11      if (!isset( $_POST['StartDate'])){
12          $sql_StartDate="DATE_SUB(NOW(), INTERVAL  7 DAY)";
13      }
14      else {
15        $sql_StartDate = new DateTime($_POST['StartDate']);
16        $StartDateDiff = date_diff(date_create(),$sql_StartDate);
17        $StartDayOffset=$StartDateDiff->days;
18        $sql_StartDate = "DATE_SUB(NOW(), INTERVAL  $StartDayOffset DAY
       )";
19
20      }
21      if (!isset($_POST['EndDate'])){
22          $sql_EndDate="NOW()";
23      }
24      else {
25        $sql_EndDate = new DateTime($_POST['EndDate']);
26        $EndDateDiff = date_diff(date_create(),$sql_EndDate);
27        $EndDayOffset=$EndDateDiff->days;
28        $EndDayOffset=$EndDayOffset-1;
29        $sql_EndDate = "DATE_SUB(NOW(), INTERVAL  $EndDayOffset DAY)";;
30      }
31
32    // check connection
33    if(!$conn){
34      echo 'Connection error: ' . mysqli_connect_error();
35    }
36
37
38    // write query for data
39
40    $sql_station1 =
41    ' SELECT DateTime, PressCur FROM VP2_0
42      WHERE DateTime BETWEEN  '.$sql_StartDate.' AND '.$sql_EndDate.'
43    ';
44
45  //ESTACIÓN 1 ALTURA: 2 METROS
       ////////////////////////////////////////////////////////////
46
47    $result_station1 = mysqli_query($conn, $sql_station1);
48
49    $data_station1 = array();
50
51
52    $i=0;
53    while ($row1 = mysqli_fetch_array($result_station1))
54    {
55      $data_station1[$i] = $row1;
56       $i++;
57    }
```

```
58   $valoresArray1;
59   $timeArray1;
60
61   for($i = 0 ;$i<count($data_station1);$i++)
62   {
63       $valoresArray1[$i]= $data_station1[$i][1];
64       //OBTENEMOS EL TIMESTAMP
65       $time1= $data_station1[$i][0];
66       $timeArray1[$i] = strtotime($time1)*1000;
67       //ALMACENAMOS EL TIMESTAMP EN EL ARRAY
68   }
69 ?>
70
71 <!DOCTYPE html>
72
73
74
75 <html>
76 <head>
77   <meta charset="utf-8">
78 </head>
79   <meta name="viewport" content="width=device-width, initial-scale=1"
       >
80 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
       font-awesome/4.7.0/css/font-awesome.min.css">
81
82 <?php include "./websiteStyle.html" ?>
83 </head>
84 <body>
85
86 <?php include "./websiteHeader.html" ?>
87
88 <div id="contenedor"></div>
89 <?php
90
91 $date=date_create();
92 date_add($date,date_interval_create_from_date_string("-7 days"));
93
94 ?>
95
96 <form action="highchartsPres.php" method="POST">
97 Fecha Inicial: <input type="date" id="StartDate" name="StartDate"
       value="<?php if (!isset($_POST['StartDate'])) {echo date_format(
       $date,"Y-m-d");} else echo date('Y-m-d',strtotime($_POST['
       StartDate'])) ?>" />
98 Fecha Final: <input type="date" id="EndDate" name="EndDate" value="<?
       php if (!isset($_POST['EndDate'])) {echo date("Y-m-d");} else
       echo date('Y-m-d',strtotime($_POST['EndDate'])) ?>" />
99 <br>
100
101
102
```

```
<input type="submit" value="Guardar Cambios" />
</form>

<script>
chartCPU = new Highcharts.Chart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: false
    },
    title: {
        text: 'Presión Vantage Pro 2'
    },
    xAxis: {
        type: 'datetime'
        //tickPixelInterval: 150,
        //maxZoom: 20 * 1000
    },
    yAxis: {
        minPadding: 0.2,
        maxPadding: 0.2,
        title: {
            text: 'Presión hPa',
            margin: 10
        }
    },
    tooltip: {
            valueSuffix: ' hPa',
            followPointer: true
        },
    series: [{
        name: 'Presión (2m)',
        data: (function () {


                var data = [];
                <?php
                    for($i = 0 ;$i<count($data_station1);$i++){
                ?>
                data.push([<?php echo $timeArray1[$i];?>,<?php echo
 $valoresArray1[$i];?>]);
                <?php } ?>
                return data;
            })()
    }],
    credits: {
            enabled: false
    }
});
```

```
154 </script>
155 </body>
156 </html>
```

A.6. Código para el estilo de la página

```
1 <style>
2 body {
3   font-family: Arial, Helvetica, sans-serif;
4 }
5
6 .navbar {
7   overflow: hidden;
8   background-color: #333;
9 }
10
11 .navbar a {
12   float: left;
13   font-size: 16px;
14   color: white;
15   text-align: center;
16   padding: 14px 16px;
17   text-decoration: none;
18 }
19
20 .dropdown {
21   float: left;
22   overflow: hidden;
23 }
24
25 .dropdown .dropbtn {
26   font-size: 16px;
27   border: none;
28   outline: none;
29   color: white;
30   padding: 14px 16px;
31   background-color: inherit;
32   font-family: inherit;
33   margin: 0;
34 }
35
36 .navbar a:hover, .dropdown:hover .dropbtn {
37   background-color: red;
38 }
39
40 .dropdown-content {
41   display: none;
42   position: absolute;
43   background-color: #f9f9f9;
```

```
44   min-width: 160px;
45   box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
46   z-index: 1;
47 }
48
49 .dropdown-content a {
50   float: none;
51   color: black;
52   padding: 12px 16px;
53   text-decoration: none;
54   display: block;
55   text-align: left;
56 }
57
58 .dropdown-content a:hover {
59   background-color: #ddd;
60 }
61
62 .dropdown:hover .dropdown-content {
63   display: block;
64 }
65 </style>
```

Bibliografía

[1] World Meteorological Organization, Guide to Meteorological Instruments and Methods of Observation. Part 2 Chapter 1, Preliminary 2018 Edition.

[2] Meteobridge Official Website [en linea] <https://www.meteobridge.com/wiki/index.php/Preparing_Hardware> [consulta: 20 diciembre 2019]

[3] Weather Underground [en linea] <https://www.wunderground.com/pws/overview> [consulta: 27 diciembre 2019]

[4] Dirección Meteorológica de Chile [en linea] <https://climatologia.meteochile.gob.cl/> [consulta: 05 febrero 2021]

[5] Explorador Eólico de Chile [en linea] <http://eolico.minenergia.cl/> [consulta: 05 febrero 2021]

[6] Documentación del API Highcharts [en linea] <https://api.highcharts.com/highcharts/> [consulta: 15 junio 2021]

[7] Documentación del API Google Charts [en linea] <https://api.highcharts.com/highcharts/> [consulta: 20 mayo 2021]

[8] Especificaciones de la estación Vantage Pro 2 [en linea] <https://support.davisinstruments.com/article/1fdqvb7bj0-spec-sheet-vantage-pro-2-wireless-stations-specifications> [consulta: 11 febrero 2021]

[9] Especificaciones de la Envoy8x [en linea] <https://support.davisinstruments.com/article/930hjmskdx-spec-sheet-envoy-8-x-for-vantage-pro-2-vue-specifications> [consulta: 11 febrero 2021]

[10] Especificaciones del WeatherLink [en linea] <https://cdn.shopify.com/s/files/1/0515/5992/3873/files/6510_SS.pdf> [consulta: 11 febrero 2021]

[11] Especificaciones de la Raspberry Pi 3B+ [en linea] <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/> [consulta: 11 febrero 2021]

[12] Variables lógicas disponibles para las estaciones Vantage Pro 2 [en linea] <https://www.meteobridge.com/wiki/index.php/Templates> [consulta: 2 marzo 2021]

[13] Guevara Díaz, José Manuel. Cuantificación del perfil del viento hasta 100 m de altura desde la superficie y su incidencia en la climatología eólica. Terra, 29(46), 81-101 [en linea] <http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1012-70892013000200006&lng=es&tlng=es> [consulta: 7 julio 2021]

[14] Documentación del lenguaje PHP [en linea] <https://www.w3schools.com/php/> [consulta: 20 febrero 2021]