

Anexo I: Estudio geotécnico

Mostrado en la figura 1, este mapa nos da un valor de la aceleración sísmica básica a_b , en función a la gravedad (g), y el coeficiente de contribución K, que tiene en cuenta la influencia de los distintos tipos de terreno.

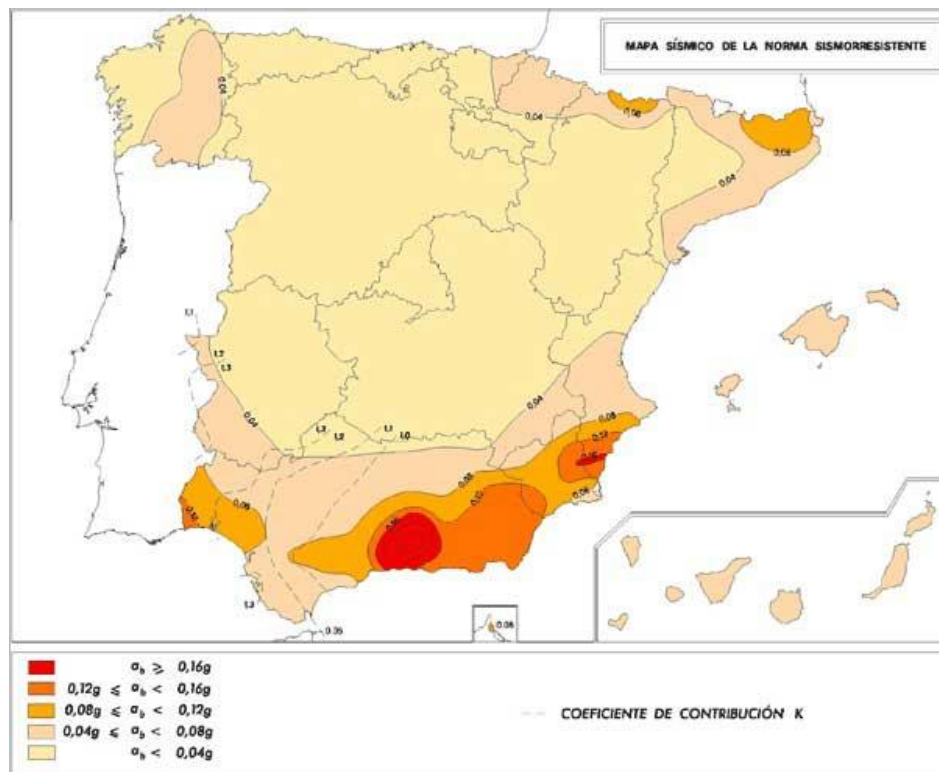


Figura 1. Mapa sísmico de la norma Sismorresistente

En nuestra zona el valor asignado será menor a 0,04g. Según la normativa usada, la aceleración sísmica de cálculo, a_c , se define como:

$$a_c = S * \rho * a_b$$

Donde:

- a_b : Aceleración sísmica de básica
- ρ : Coeficiente adimensional de riesgo

Toma los siguientes valores:

- Construcciones de importancia normal $\rho = 1,00$.

- Construcciones de importancia especial $\rho = 1,30$.

S: Coeficiente de amplificación del terreno. Toma el valor:

- Para $\rho \cdot ab < 0,1g$

$$S = C/1,25$$

El valor del coeficiente “C” varía en función a la clasificación del terreno:

- **Tipo I:** Roca compacta, suelo cementado o granular muy denso.
- **Tipo II:** Roca muy fracturada, suelos granulares densos o cohesivos duros
- **Tipo III:** Suelo granular de compacidad media, suelo cohesivo de consistencia firme o muy firme.
- **Tipo IV:** Suelo granular suelto, o cohesivo blando

Donde nuestro caso es un Tipo I. Conociendo los siguientes valores:

$$ab = 0,04$$

$$\rho = 1$$

$$S = 0.8$$

Y usando la fórmula:

$$ac = S * \rho * ab$$

Sabemos que el valor de $ac = 0,032$. Donde podemos decir que este valor es una aceleración apenas perceptible por personas sensibles, y que normalmente no causa daños estructurales por lo que no tendremos que tomar medidas frente a sismos.

Anexo II: Dimensionamiento del sistema de hidrógeno

En este anexo se desarrollará el dimensionamiento del electrolizador, almacenamiento y pila de combustible.

A partir de la producción eólica anual (73,546 GWh/año) y el consumo del centro de datos (52,56 GWh/año) se obtiene un excedente medio de 20,986 GWh/año. Este excedente se destina a la producción de hidrógeno verde.

Electrolizador

El electrolizador será de tipo PEM o de membrana de intercambio de protones, el cual produce hidrógeno de alta pureza a partir de agua, utilizando una membrana de intercambio de protones y un electrolito polimérico sólido. Son los más adecuados para adaptarse a las variaciones de las energías renovables y son fáciles de refrigerar. Cuenta con una potencia nominal de 5 MW y su consumo específico es de 55 kWh/kgH₂. Con ello, podemos calcular su producción horaria máxima:

$$\dot{m}_{H_2} = \frac{5\,000\text{ kW}}{55\text{ kWh/kg } H_2} = 91\text{ kg/h}$$

También la producción anual estimada:

$$\dot{m}_{H_2, \text{anual}} = \frac{20\,986\,000\text{ kWh}}{55\text{ kWh/kg } H_2} = 381\,600\text{ kg/año} \approx 382\text{ t/año}$$

Almacenamiento

Para dimensionar el sistema de almacenamiento de hidrógeno, se ha definido como objetivo garantizar una autonomía mínima de 72 horas para la carga de refrigeración del centro de datos. Dado que la potencia asociada a la alimentación es de 4,2 MW, la energía total necesaria para cubrir 3 días de operación ininterrumpida se calcula como el producto de la potencia por el tiempo de funcionamiento, siendo de 302,4 MWh.

$$E_{72h} = 4,2\text{ MW} \cdot 72h = 302,4\text{ MWh}$$

La conversión de este requerimiento energético en masa de hidrógeno depende del rendimiento de la pila de combustible seleccionada. Considerando una pila PEMFC con un rendimiento eléctrico del 50 % y tomando como base el poder calorífico inferior del hidrógeno, de 33,33 kWh/kg, la energía eléctrica obtenida por cada kilogramo de hidrógeno en la pila de combustible (E_{PC}) es de 16,665 kWh/kg.

$$E_{PC} = 0,5 \cdot 33,33 \text{ kWh/kg} = 16,665 \text{ kWh/kg}$$

Si dividimos la demanda de 302,4 MWh entre la energía específica, obtenemos que la masa total necesaria para alcanzar la autonomía que requerimos es de 18145,8 kg de hidrógeno.

$$m_{H_2(72h)} = \frac{302\,400 \text{ kWh}}{16,665 \frac{\text{kWh}}{\text{kg}}} = 18145,8 \text{ kg} \approx 18,14 \text{ t } H_2$$

En cuanto al volumen requerido para el almacenamiento, se plantea comprimir el gas a una presión de 350 bar, para estas condiciones la densidad del hidrógeno normalmente es alrededor de 23 kg/m^3 , con lo cual, el volumen de almacenamiento equivalente para 18,14 t H_2 asciende a $788,9 \text{ m}^3$.

$$V = \frac{18145,8 \text{ kg}}{23 \text{ kg/m}^3} \approx 788,9 \text{ m}^3$$

Pila de combustible

La pila de combustible seleccionada para el proyecto es de tipo PEMFC (Proton Exchange Membrane Fuel Cell), por su idoneidad en aplicaciones que requieren flexibilidad operativa, arranque rápido y alta densidad de potencia. Se ha dimensionado una potencia nominal de 4,5 MW, lo que permite cubrir de manera holgada la demanda de funcionamiento del centro de datos, estimada en 4,2 MW, y disponer de un margen de seguridad adicional del 5 %. Este margen resulta esencial en instalaciones críticas de categoría Tier III, donde la continuidad de servicio debe estar garantizada incluso frente a incrementos puntuales de carga o a ligeras pérdidas de rendimiento de los equipos.

$$\dot{m}_{H_2} = \frac{4200 \text{ kWh}}{16,665 \text{ kWh/kg}} \approx 252 \text{ kg/h}$$

El rendimiento eléctrico considerado para la pila de combustible es del 50 %, lo que reduce el poder calorífico inferior del hidrógeno (33,33 kWh/kg) a una energía eléctrica útil de 16,665 kWh por kilogramo. Con esta base, para cubrir la demanda de operación de 4,2 MW, la pila requiere un consumo aproximado de 252 kg de hidrógeno por hora, resultado de dividir la potencia entre la energía específica disponible por kilogramo.

Balance energético anual

A partir del excedente de producción eólica, el electrolizador de 5 MW es capaz de generar en torno a 382 toneladas de hidrógeno al año. Este hidrógeno, una vez reconvertido en electricidad a través de la pila de combustible con un rendimiento del 50 %, supone una energía útil de aproximadamente 6,36 GWh anuales.

$$E_{PC} = 381\,600\,kg\,H_2/año \cdot 16,665\,kWh/kg\,H_2 = 6,36\,GWh/año$$

Esta cantidad se destina a cubrir el consumo total del centro de datos, y a garantizar el funcionamiento de los sistemas durante periodos de baja generación eólica. Considerando que la demanda anual asciende a 52,56 GWh, la electricidad aportada por el hidrógeno permite cubrir alrededor del 12 % de estas necesidades, asegurando así una importante contribución a la estabilidad operativa y reduciendo de forma significativa la dependencia de la red.

$$Cobertura\,demanda = \frac{6,36\,GWh/año}{52,56\,GWh/año} = 12\%$$

O lo que se traduce en 44 días a lo largo del año de suministro íntegro ante la ausencia persistente de viento.

Impacto ambiental

La integración del hidrógeno verde no solo refuerza la seguridad energética, sino que también tiene un efecto directo en la reducción de emisiones. Si se considera que la demanda anual del centro de datos incluida la refrigeración es de 52,56 GWh, y que un 44 % de la electricidad de la red procede de fuentes no renovables con un factor de emisión medio de 0,11 toneladas de CO₂ por MWh, la sustitución de esa fracción mediante hidrógeno renovable evita la emisión de unas 2544 toneladas de CO₂ al año. (Instituto para la Diversificación y Ahorro de la Energía, s.f.)

$$\text{Impacto ambiental} = 52,56 \frac{GWh}{\text{año}} \cdot 0,44 \cdot 0,11 \frac{t CO_2}{MWh} \approx 2544 t CO_2/\text{año}$$

Este resultado refleja el doble beneficio del sistema propuesto: por un lado, garantizar la resiliencia del centro de datos, y por otro, contribuir a la descarbonización del sector energético en línea con los objetivos de sostenibilidad marcados a nivel europeo y nacional.

Anexo III. Sistema de control de temperatura

El presente capítulo describe el diseño e implementación del sistema de control de la planta híbrida eólica-hidrógeno y del centro de datos. El objetivo es asegurar un suministro continuo, optimizar el uso de energías renovables y mantener el CPD en condiciones térmicas seguras (18–27 °C según ASHRAE).

Plataforma de control

El sistema se implementa en la placa NVIDIA Jetson Nano Developer Kit (modelo P3448, B01), un sistema embebido de bajo consumo ($\approx 5\text{--}10\text{ W}$) que integra CPU ARM Cortex-A57 de cuatro núcleos y GPU NVIDIA Maxwell de 128 núcleos CUDA con 4 GB de RAM. Estas características permiten combinar control en tiempo real, optimización energética y predicción con IA ligera. Las figuras 2, 3, 4 y 5 muestran distintas vistas de la Jetson Nano utilizada en el desarrollo.

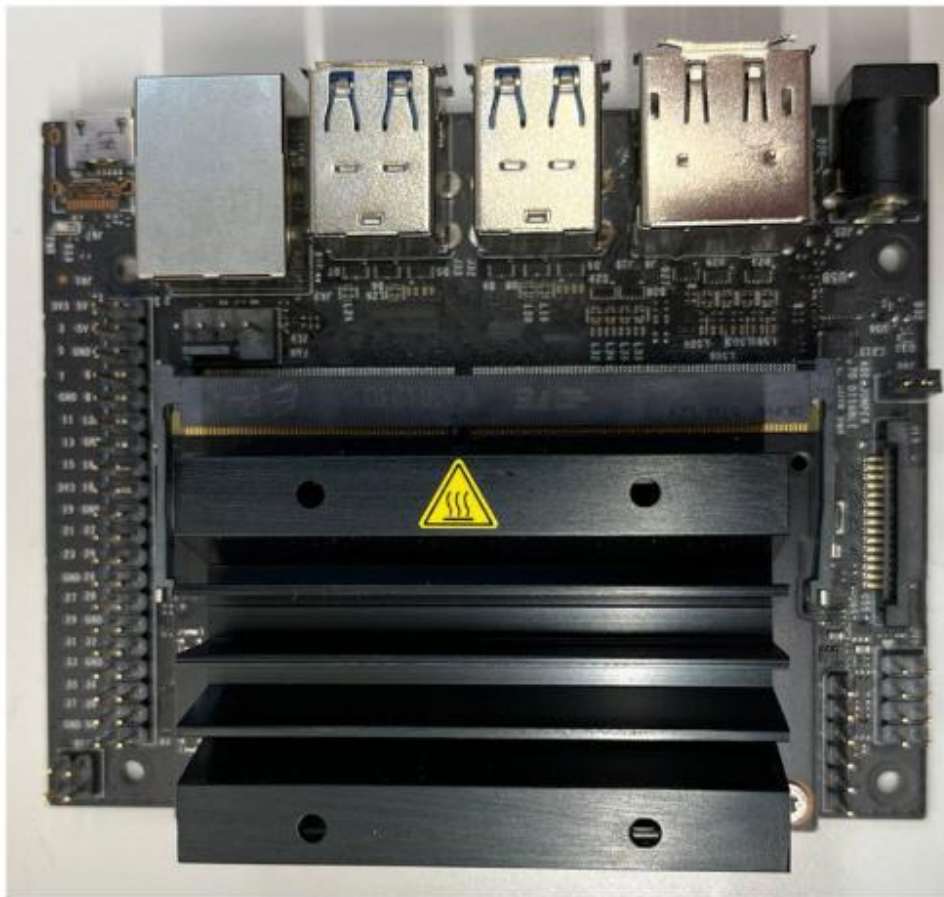


Figura 2. Vista de conjunto de la Jetson Nano con disipador. Fuente: Elaboración propia

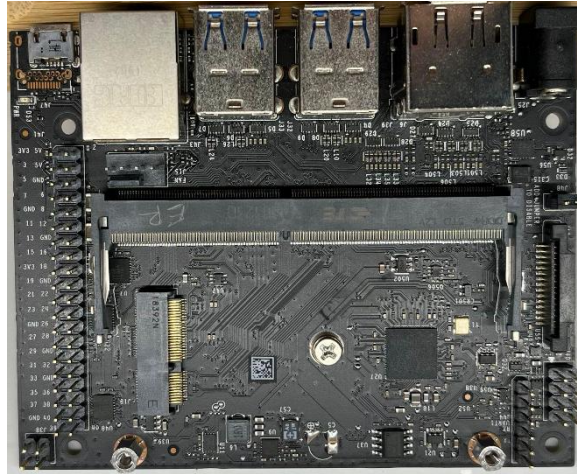


Figura 3. Vista superior de la placa base Jetson Nano. Fuente: Elaboración propia



Figura 4. Módulo Jetson Nano (SoM) con ranura microSD. Fuente: Elaboración propia









Figura 5. Vista inferior de la placa con serigrafía de pines GPIO. Fuente: Elaboración propia

Arquitectura del software

El software está compuesto por los siguientes módulos en Python:

- **config.py**: parámetros de operación (33,6 MW eólicos, 4,2 MW refrigeración, electrolizador 5 MW, pila de combustible 4,5 MW, umbrales térmicos 18–27 °C, pines GPIO).
- **functions.py**: utilidades de apoyo para formateo y gráficos.
- **main.py**: arranque de la aplicación y de la lógica de control.
- **main_window.py**: interfaz gráfica principal (inicio, gráficas, registro de eventos).
- **app_logic.py**: lógica de control en tiempo real, ejecutada cada segundo.

Estructura de los archivos:

	assets	
	icons	
	images	
	controllers	
	app_logic.py	Contiene la lógica del funcionamiento de la aplicación, el ciclo principal.
	dialogs	
	about_dialog.py	Diálogo "Acerca del programa"
	settings_dialog.py	Diálogo "Configuración"
	models	
	data.py	Almacena parámetros temporales del funcionamiento de la aplicación y los datos necesarios para simular el consumo de energía y la temperatura del bastidor del servidor.
	device.py	Contiene funciones para el control de los puertos de entrada/salida de la placa Jetson Nano.
	service.py	Contiene una función que obtiene datos meteorológicos de Internet.

📁 panels

energy_chart.py	Objeto que implementa el gráfico de consumo de energía.
panel_chart.py	Panel que contiene el gráfico en pantalla completa.
panel_events_log.py	Panel que contiene el registro de eventos.
panel_home.py	Panel que contiene la pantalla principal de la aplicación.
side_menu.py	Menú lateral de la aplicación. *
config.py	Contiene todos los parámetros y configuraciones.
functions.py	Funciones auxiliares adicionales.
main.py	Archivo principal de la aplicación, se ejecuta primero al iniciar el programa.
main_window.py	Ventana principal de la interfaz gráfica.

Enumeración de pines GPIO

Respecto a la gestión de entradas y salidas digitales en el sistema, se utilizan diversos pines GPIO usando la numeración lógica Broadcom (BCM) para gestionar las funciones principales relacionadas con la producción, almacenamiento y consumo de energía, así como el control térmico y la monitorización ambiental. A continuación, se detallan las funciones implementadas y su integración en el sistema, tal y como se refleja en la tabla 1:

Función controlada	GPIO (BCM)	Pin físico	Tipo de señal	Utilidad en el sistema
Entrada de potencia eólica (simulada)	26	37	Entrada digital (Digital In)	Lectura de la potencia del generador
Activación de la pila de combustible (PEMFC)	19	35	Salida digital (Digital Out)	Encender/apagar la pila
Activación del electrolizador PEM	13	33	Salida digital (Digital Out)	Encender/apagar el electrolizador
Entrada de red eléctrica	6	31	Entrada digital (Digital In)	Estado de la conexión a la red
Exportación a la red eléctrica	5	29	Salida digital (Digital Out)	Orden de exportación a la red
Ventiladores de free-cooling	11	17	Salida digital/PWM	Control de los ventiladores

			(Digital Out/PWM)	
Bomba de calor	12	32	Salida digital (Digital Out)	Arranque/parada de la bomba
Sensor de temperatura y humedad interior	17	11	Entrada digital (I2C/DHT) (Digital In (I2C/DHT))	Lectura ambiental interior

Tabla 1. Asignación final de pines de la Jetson Nano

Control de potencia y fuentes de energía

Entrada de potencia eólica (simulada): El pin 26 (físico 37) está configurado como entrada digital, permitiendo la lectura de la potencia generada por el simulador de generador eólico.

Activación de la pila de combustible (PEMFC): El pin 19 (físico 35) funciona como salida digital, encargado de activar y desactivar la pila de combustible.

Activación del electrolizador PEM: El pin 13 (físico 33), también configurado como salida digital, controla el funcionamiento del electrolizador PEM, permitiendo encenderlo y apagarlo según las necesidades del sistema.

Gestión de la red eléctrica

Entrada desde la red eléctrica: El pin 6 (físico 31) está definido como entrada digital y se utiliza para detectar el estado de conexión a la red eléctrica.

Exportación a la red eléctrica: El pin 5 (físico 29) funciona como salida digital, emitiendo la orden de exportación de energía excedente a la red.

Sistema de control térmico

Ventiladores de free-cooling: El pin 11 (físico 17) puede operar como salida digital o PWM, siendo responsable de controlar los ventiladores que ayudan en la gestión térmica del ambiente.

Bomba de calor: El pin 12 (físico 32), configurado como salida digital, enciende o apaga la bomba de calor para mantener la temperatura interna adecuada.

Monitorización ambiental

Sensor de temperatura y humedad interior: El pin 17 (físico 11) recibe señales del sensor, ya sea vía I2C o DHT, permitiendo la lectura de los parámetros ambientales internos esenciales para el control térmico y el registro del sistema.

Esta estructura de entradas y salidas digitales garantiza una interacción eficiente entre los componentes físicos del sistema y el algoritmo de control, permitiendo respuestas automáticas a las condiciones de operación y facilitando la gestión integrada de energía y ambiente.

Lógica de control

La lógica de control implementa un ciclo de 1 segundo que puede explicarse en tres bloques principales:

Control térmico

- **Free-cooling:** activación de ventiladores si la temperatura exterior es inferior a 18 °C y la humedad relativa inferior al 70 %.
- **Bomba de calor:** activación si la temperatura interior del CPD supera los 27 °C; apagado al descender a 25 °C, aplicando histéresis para evitar conmutaciones rápidas.

Gestión energética

- **Eólica → CPD:** la energía eólica se destina prioritariamente a la carga del centro de datos.
- **Excedentes:** si tras cubrir la demanda aún existe potencia sobrante, se activa el electrolizador hasta un máximo de 5 MW para producción de hidrógeno.
- **Déficits:** si la generación es insuficiente y el nivel de hidrógeno almacenado supera el 20 %, se activa la pila de combustible hasta 4,5 MW.
- **Intercambio con red:** si persiste excedente y el tanque de H₂ está lleno, se realiza exportación a red; si persiste déficit y no hay hidrógeno disponible, se importa de la red.

Resumiendo, la lógica de control ejecuta un ciclo de 1 segundo que comprende:

1. Lectura: potencia eólica, carga TI, temperatura interior/exterior, humedad, nivel de H₂, estado de la red.

2. Gestión térmica: ventiladores si $T_{ext} < 18\text{ °C}$ y $HR < 70\%$. Bomba de calor si $T_{int} > 27\text{ °C}$ y apagado a 25 °C.

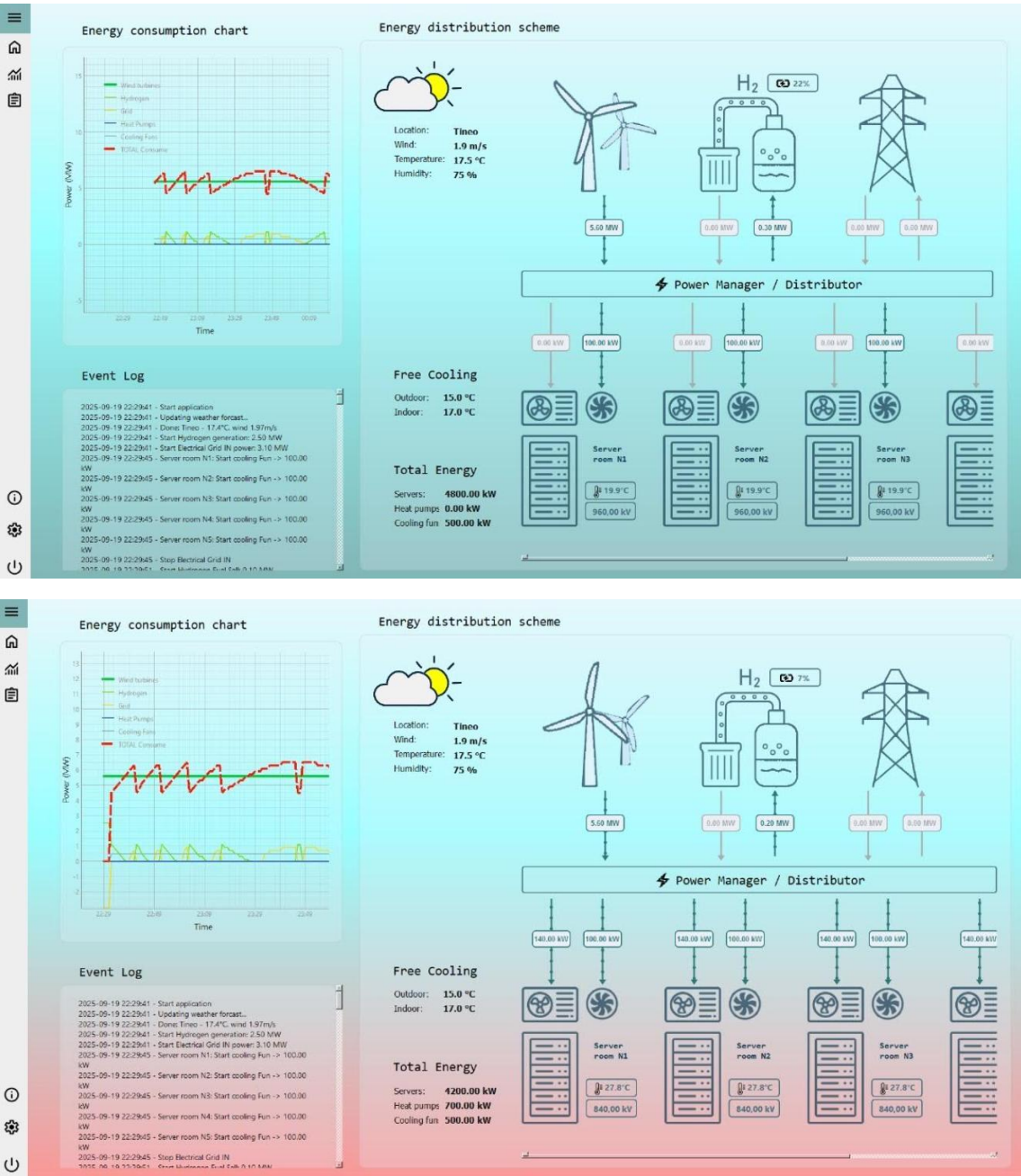
3. Gestión energética: eólica → CPD (prioridad); excedente → electrolizador (máx. 5

MW); déficit → pila de combustible (máx. 4,5 MW, si $H_2 > 20\%$); excedente con tanque lleno → exportación a red; déficit sin H_2 → importación de red.

Registro y visualización: actualización de la interfaz gráfica y registro de eventos

Todos los eventos y decisiones se reflejan en la interfaz gráfica, junto con gráficas de potencias y condiciones térmicas, garantizando trazabilidad operativa.

La figura 6 presenta un ejemplo de la interfaz utilizada en las simulaciones previas, donde se visualizan los flujos energéticos, el consumo y los registros de operación.



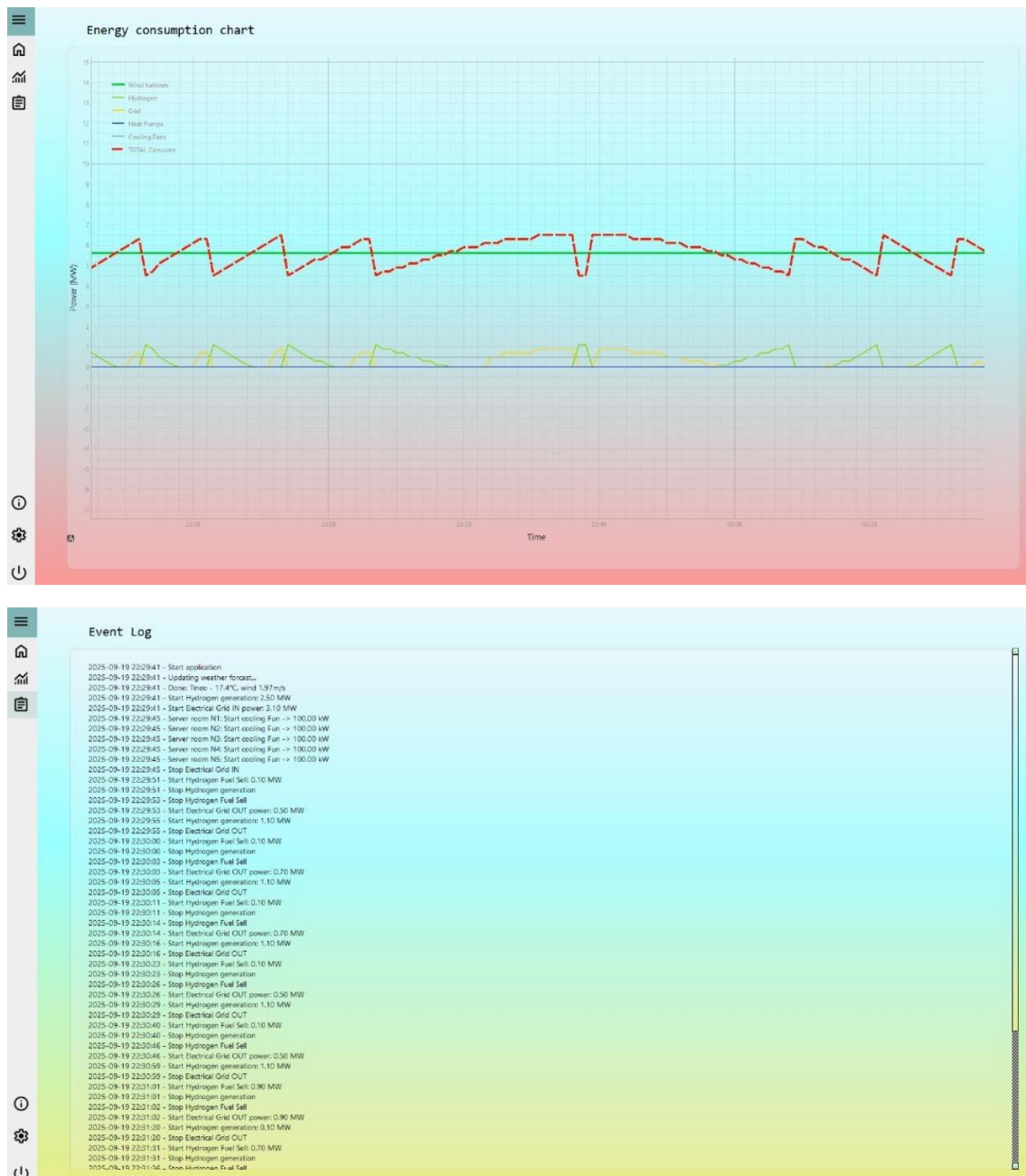


Figura 6. Interfaz utilizada en las simulaciones, registro de eventos y curva de consumo. Fuente: Elaboración propia

Anexo IV: Código fuente del controlador de temperatura

A continuación, se presenta el código planteado para el sistema de control de temperatura:

1. app_logic.py

Contiene la lógica del funcionamiento de la aplicación, el ciclo principal.

```
from pyqtgraph.Qt.QtCore import QDateTime
from pyqtgraph.Qt.QtCore import QTimer
import config
import models.data as app_data
import models.device as app_device
import models.service as app_service

class AppLogic:
    def __init__(self, main_window):
        self._current_time_index = 0
        self._current_time_simulation_index = 0

        self._main_window = main_window
        self._home_panel = main_window.home_screen
        self._event_log_panel = main_window.log_screen
        self._chart_panel = main_window.chart_screen
        self._wind_generator_widget = self._home_panel.wind_generator_widget
        self._home_panel_energy_chart_widget = self._home_panel.energy_chart_widget
        self._chart_panel_energy_chart_widget = self._chart_panel.energy_chart_widget

        self._home_panel.update_weather()
        # Start the weather update cycle
```

```

self.app_weather_timer = QTimer()
self.app_weather_timer.timeout.connect(self._home_panel.update_weather)
self.app_weather_timer.start(config.WEATHER_TIMER * 60 * 1000) # 5 min

self.run_app_logic()
# Start the main application cycle
self.app_logic_timer = QTimer()
self.app_logic_timer.timeout.connect(self.run_app_logic)
self.app_logic_timer.start(1000) # 1 second

def run_app_logic(self):
    # print("run app_logic()")
    request_power = 0 # Request power consumes needs
    power_production_surplus = 0 # Energy production surplus

    # 1. Get current Wind generator power
    current_wind_power =
round(app_device.get_wind_generator_power(app_data.wind) *
config.WIND_GENERATORS_NUMBER / 1000, 1)
    # Update Wind Generator rotation power Info
    self._home_panel.update_wind_generator_power(current_wind_power)
    app_device.set_wind_generator_power_in(current_wind_power)

    # 3. Get current Servers Power Load
    current_server_rack_power_load
= app_device.get_server_power_load(self._current_time_simulation_index) * 4

    # 3. Get current Servers temperature and outdoor conditions
    current_servers_temperature =
app_device.get_servers_temperature(self._current_time_simulation_index)
    current_outdoor_temperature = app_device.get_outdoor_temperature()
    current_outdoor_humidity = app_device.get_outdoor_humidity()

```



```
current_indoor_temperature =  
app_device.get_indoor_temperature(self._current_time_simulation_index)
```

```
# Update Free Cooling Info
```

```
self._home_panel.update_free_cooling_info()
```

```
# 4. Calculate Total Cooling System Load
```

```
current_fan_power = 0
```

```
current_pump_power = 0
```

```
if current_server_rack_power_load > 0 and current_servers_temperature >  
config.SERVER_TEMPERATURE_MIN:
```

```
    # get current Free Cooling Fun power
```

```
    if current_outdoor_temperature < config.OUTDOOR_TEMPERATURE_MAX  
and current_outdoor_humidity < config.OUTDOOR_HUMIDITY_MAX:
```

```
        current_fan_power = config.COOLING_FUN_POWER
```

```
if current_servers_temperature > config.SERVER_TEMPERATURE_MAX:
```

```
    # get current Heat Pump power
```

```
    current_pump_power = config.HEAT_PUMP_POWER
```

```
# Update Cooling Funs power Info
```

```
self._home_panel.serverRack1.set_cooling_fun_power(current_fan_power * 4)
```

```
self._home_panel.serverRack2.set_cooling_fun_power(current_fan_power * 4)
```

```
self._home_panel.serverRack3.set_cooling_fun_power(current_fan_power * 4)
```

```
self._home_panel.serverRack4.set_cooling_fun_power(current_fan_power * 4)
```

```
self._home_panel.serverRack5.set_cooling_fun_power(current_fan_power * 4)
```

```
app_device.set_free_cooling_fun_power(current_fan_power)
```

```
# Update Heat Pump power Info
```

```
self._home_panel.serverRack1.set_heat_pump_power(current_pump_power * 4)
```

```
self._home_panel.serverRack2.set_heat_pump_power(current_pump_power * 4)
```

```

self._home_panel.serverRack3.set_heat_pump_power(current_pump_power * 4)
self._home_panel.serverRack4.set_heat_pump_power(current_pump_power * 4)
self._home_panel.serverRack5.set_heat_pump_power(current_pump_power * 4)
app_device.set_heat_pump_power(current_pump_power)

# Update Server Temperature Info
self._home_panel.serverRack1.set_temperature(current_servers_temperature)
self._home_panel.serverRack2.set_temperature(current_servers_temperature)
self._home_panel.serverRack3.set_temperature(current_servers_temperature)
self._home_panel.serverRack4.set_temperature(current_servers_temperature)
self._home_panel.serverRack5.set_temperature(current_servers_temperature)

# Update Server Load Power Info
self._home_panel.serverRack1.set_power(current_server_rack_power_load)
self._home_panel.serverRack2.set_power(current_server_rack_power_load)
self._home_panel.serverRack3.set_power(current_server_rack_power_load)
self._home_panel.serverRack4.set_power(current_server_rack_power_load)
self._home_panel.serverRack5.set_power(current_server_rack_power_load)

current_total_fan_power = current_fan_power * config.SERVER_NUMBERS
current_total_pump_power = current_pump_power *
config.SERVER_NUMBERS
app_data.total_cooling_funs_load = round(current_total_fan_power, 2)
app_data.total_heat_pumps_load = round(current_total_pump_power, 2)

# get current TOTAL power
current_total_servers_power =
app_device.get_server_power_load(self._current_time_simulation_index) *
config.SERVER_NUMBERS
app_data.total_servers_load = current_total_servers_power
current_total_load = round((current_total_servers_power +
current_total_fan_power + current_total_pump_power) / 1000, 2)

```

```

# Update current TOTAL power Info
self._home_panel.update_total_energy_info()

# Calculate request power covered by Wind generator
request_power = current_total_load - current_wind_power

# get current Hydrogen generator power STATE
current_hydrogen_power = 0.0

if app_data.hydrogen_storage < 100 and request_power < 0:
    # Start Generate H2
    power_delta = current_wind_power - current_total_load

    current_hydrogen_power = power_delta if power_delta <
config.ELECTROLYZER_POWER_MAX else
config.ELECTROLYZER_POWER_MAX

    # if power_delta < 2.5:
    #     current_hydrogen_power = power_delta
    #     request_power = 0
    # else:
    #     current_hydrogen_power = 2.5
    #     request_power += 2.5

    request_power += current_hydrogen_power
    app_device.set_hydrogen_generator(current_hydrogen_power)
else:
    # Stop Generate H2
    app_device.set_hydrogen_generator(0)

self._home_panel.update_hydrogen_generator_flow()

```

```

# get current Hydrogen Fuel Sell STATE
app_data.hydrogen_fuel_sell_power = 0.0

if request_power > 0 and app_data.hydrogen_storage >
config.H2_MIN_SOC_PERCENT:

    # Start Hydrogen Fuel Sell

    if request_power < config.HYDROGEN_FUEL_SELL_POWER:

        # app_device.set_hydrogen_fuel_sell(request_power)

        app_data.hydrogen_fuel_sell_power = request_power

        request_power = 0

    else:

        #
app_device.set_hydrogen_fuel_sell(config.HYDROGEN_FUEL_SELL_POWER)

        request_power -= config.HYDROGEN_FUEL_SELL_POWER


self._home_panel.update_hydrogen_fuel_sell_flow(app_data.hydrogen_fuel_sell_
power)

self._home_panel.update_hydrogen_storage_info()

app_device.set_hydrogen_fuel_sell(app_data.hydrogen_fuel_sell_power)


power_production_surplus = -request_power


# get current Electrical Grid power
app_data.electrical_grid_power_in = 0
app_data.electrical_grid_power_out = 0
current_grid_power = request_power


if request_power > 0:

    current_grid_power = request_power

    app_data.electrical_grid_power_out = request_power


if request_power < 0:

```

```

        current_grid_power = request_power

        app_data.electrical_grid_power_in -= request_power

        self._home_panel.update_electrical_grid_in_power(app_data.electrical_grid_power_in)

        self._home_panel.update_electrical_grid_out_power(app_data.electrical_grid_power_out)

    # Update Chart

    self._home_panel_energy_chart_widget.add_chart_data(self._current_time_index,
        current_wind_power, current_grid_power, current_hydrogen_power,
        current_total_pump_power / 1000, current_total_fan_power / 1000, current_total_load)

    self._home_panel_energy_chart_widget.update_chart_data()

    self._chart_panel_energy_chart_widget.add_chart_data(self._current_time_index,
        current_wind_power, current_grid_power, current_hydrogen_power,
        current_total_pump_power / 1000, current_total_fan_power / 1000, current_total_load)

    self._chart_panel_energy_chart_widget.update_chart_data()

    # New cycle

    self._current_time_index += 1

    self._current_time_simulation_index += 1

    if self._current_time_simulation_index == 600:

        self._current_time_simulation_index = 0

    self._main_window.update_app_background(current_servers_temperature)

    # print(f'self._current_time_simulation_index = {self._current_time_simulation_index}')

def add_event(self, event_text):

    now = QDateTime.currentDateTime().toString("yyyy-MM-dd hh:mm:ss")

    app_data.data_events_log += f'{now} - {event_text}<br>'

```

```
self._home_panel.update_log_event(event_text)

self._event_log_panel.update_log_event(event_text)
```

2. **data.py**

Almacena parámetros temporales del funcionamiento de la aplicación y los datos necesarios para simular el consumo de energía y la temperatura del bastidor del servidor.

```
data_events_log = ""

# Weather Data
temperature = 0
pressure = 0
humidity = 0
cloudiness = 0
wind = 0

# Hydrogen System
hydrogen_storage = 0
hydrogen_generate = 0
hydrogen_fuel_sell_power = 0
is_hydrogen_generate_started = False
is_hydrogen_fuel_sell_started = False

# Electrical Grid
electrical_grid_power_in = 0
electrical_grid_power_out = 0
is_electrical_grid_power_in_started = False
is_electrical_grid_power_out_started = False

# Free Cooling
```

```
outdoor_temperature = 0
```

```
indoor_temperature = 0
```

```
indoor_humidity = 0
```

```
# Total Energy Info
```

```
total_servers_load = 0
```

```
total_heat_pumps_load = 0
```

```
total_cooling_funs_load = 0
```

```
# ===== Simulation data: =====
```

```
# Simulation period: 10 min - 600 seconds
```

```
# One SERVER power load
```

```
data_server_load = [
```

```
    0.0000,
```

```
    0.0000,
```

```
    0.0000,
```

```
    7.5000,
```

```
    7.4895,
```

```
    7.5000,
```

```
    ...n
```

```
]
```

```
# SERVERs indor temperature
```

```
data_servers_temperature = [
```

```
    19.0000,
```

```
    19.0533,
```

```
    19.1067,
```

```
    ...n
```

```
    30.8325,
```

30.8473,
30.8614,
30.8749,
30.8876,
30.8997,
30.9111,
30.9218,
30.9319,
30.9413,
30.9499,
30.9579,
30.9652,
30.9718,
30.9777,
30.9829,
30.9875,
30.9913,
30.9944,
30.9969,
30.9986,
30.9997,
31.0000,
31.0000,
30.9378,
30.8756,
30.8134,
30.7512,
30.6890,
30.6268,
30.5646,

30.5024,
30.4402,
30.3780,
30.3158,
30.2536,
30.1914,
30.1292,
30.0670,
30.0048,
29.9426,
29.8804,
29.8182,
29.7560,
29.6938,
29.6316,
29.5694,
29.5072,
29.4450,
29.3828,
29.3206,
29.2584,
29.1962,
29.1340,
29.0718,
29.0096,
28.9474,
28.8852,
28.8230,
28.7608,
28.6986,

28.6364,
28.5742,
28.5120,
28.4498,
28.3876,
28.3254,
28.2632,
28.2010,
28.1388,
28.0766,
28.0144,
27.9522,
27.8900,
27.8278,
27.7656,
27.7033,
27.6411,
27.5789,
27.5167,
27.4545,
27.3923,
27.3301,
27.2679,
27.2057,
27.1435,
27.0813,
18.9915,
19.0000,

]

3. **device.py**

Contiene funciones para el control de los puertos de entrada/salida de la placa Jetson Nano:

```
import config
import models.data as app_data

is_jetson_nano = True

try:
    import Jetson.GPIO as GPIO # pip3 install Jetson.GPIO

    GPIO.setmode(GPIO.BCM)
    GPIO.setup(config.GPIO_WIND_GENERATOR_POWER_IN, GPIO.OUT)

    GPIO.setup(config.GPIO_HYDROGEN_FUEL_SELL_POWER_ON, GPIO.OUT)
    GPIO.setup(config.GPIO_HYDROGEN_GENERATION_ON, GPIO.OUT)

    GPIO.setup(config.GPIO_ELECTRICAL_GRID_POWER_IN, GPIO.OUT)
    GPIO.setup(config.GPIO_ELECTRICAL_GRID_POWER_OUT, GPIO.OUT)

    GPIO.setup(config.GPIO_FREE_COOLER_FUN_POWER_ON, GPIO.OUT)
    GPIO.setup(config.GPIO_HEAT_PUMP_POWER_ON, GPIO.OUT)

    import Adafruit_DHT # pip3 install Adafruit_DHT

except ImportError:
    print("GPIO module not available (not running on Jetson?)")
    is_jetson_nano = False

# Wind generator V112-3.0 MW
```

```

# Returns the power (kW) delivered by the V112-3.0 MW wind turbine generator
# depending on wind speed (m/s)
def get_wind_generator_power(wind_speed_mps: float) -> float:
    if config.SIMULATION_WIND_IS_USED:
        wind_speed_mps = config.SIMULATION_WIND_SPEED

    if wind_speed_mps < 3.5:
        return 0.0

    elif 3.5 <= wind_speed_mps < 12.5:
        # Proportional power growth from 0 to 3000 kW
        return 3000.0 * ((wind_speed_mps - 3.5) / (12.5 - 3.5))

    elif 12.5 <= wind_speed_mps <= 25.0:
        # Maximum power
        return 3000.0

    else:
        # When the threshold is exceeded, it is switched off
        return 0.0

def get_indoor_temperature(simulation_index) -> float:
    if config.SIMULATION_TEMPERATURE_IS_USED:
        app_data.indoor_temperature =
config.SIMULATION_INDOOR_TEMPERATURE

    else:
        # Get Indoor Temperature from Sensor
        if is_jetson_nano:
            sensor = Adafruit_DHT.DHT11

            # Reading sensor data
            app_data.indoor_humidity, app_data.indoor_temperature =
Adafruit_DHT.read_retry(sensor,
config.GPIO_INDOOR_TEMP_HUMIDITY_SENSOR)

```

```

        else:

            app_data.indoor_temperature =
app_data.data_servers_temperature[simulation_index] - 1.5

        return app_data.indoor_temperature

def get_outdoor_temperature() -> float:
    if config.SIMULATION_TEMPERATURE_IS_USED:
        app_data.outdoor_temperature = config.SIMULATION_TEMPERATURE

    else:
        app_data.outdoor_temperature = app_data.temperature

    return app_data.outdoor_temperature

def get_outdoor_pressure() -> float:
    return app_data.pressure

def get_outdoor_humidity() -> float:
    if config.SIMULATION_WIND_IS_USED:
        app_data.humidity = config.SIMULATION_HUMIDITY

    else:
        app_data.humidity = app_data.humidity

    return app_data.humidity

def get_server_power_load(simulation_index = 0) -> float:
    return round(app_data.data_server_load[simulation_index], 2)

```

```

def get_servers_temperature(simulation_index = 0) -> float:
    return app_data.data_servers_temperature[simulation_index]

def set_hydrogen_generator(power):
    app_data.hydrogen_generate = power

    # Set Jetson Nano Hydrogen H2 Generation Power management signal
    if is_jetson_nano:
        GPIO.output(config.GPIO_HYDROGEN_GENERATION_ON, power > 0)

def set_hydrogen_fuel_sell(power):
    app_data.hydrogen_fuel_sell_power = power

    # Set Jetson Nano Hydrogen Fuel Sell Power management signal
    print(f"set_hydrogen_fuel_sell(power = {power})")
    if is_jetson_nano:
        GPIO.output(config.GPIO_HYDROGEN_FUEL_SELL_POWER_ON, power >
0)

def set_wind_generator_power_in(power):
    app_data.wind_generator_power_in = power

    # Set Jetson Nano Wind Generator Power IN management signal
    print(f"set_wind_generator_power_in(power = {power})")
    if is_jetson_nano:
        GPIO.output(config.GPIO_WIND_GENERATOR_POWER_IN, power > 0)

def set_electrical_power_in(power):
    app_data.electrical_grid_power_in = power

    # Set Jetson Nano Electrical Grid Power IN management signal

```

```

if is_jetson_nano:
    GPIO.output(config.GPIO_ELECTRICAL_GRID_POWER_IN, power > 0)

def set_electrical_power_out(power):
    app_data.electrical_grid_power_out = power

    # Set Jetson Nano Electrical Grid Power OUT management signal
    if is_jetson_nano:
        GPIO.output(config.GPIO_ELECTRICAL_GRID_POWER_OUT, power > 0)

def set_free_cooling_fun_power(power):
    # Set Jetson Nano Free Cooling Fun management signal
    if is_jetson_nano:
        GPIO.output(config.GPIO_FREE_COOLER_FUN_POWER_ON, power > 0)

def set_heat_pump_power(power):
    # Set Jetson Nano Heat Pump management signal
    if is_jetson_nano:
        GPIO.output(config.GPIO_HEAT_PUMP_POWER_ON, power > 0)

```

4. **service.py**

Contiene una función que obtiene datos meteorológicos de Internet:

```

import requests
import config
import models.data as app_data

def get_weather():
    response = requests.get(config.URL)
    datas = response.json()
    app_data.temperature = datas['main']['temp']

```

```

app_data.pressure = datas['main']['pressure']
app_data.humidity = datas['main']['humidity']
app_data.cloudiness = datas['clouds']['all']
app_data.wind = datas['wind']['speed']

```

5. config.py

Contiene todos los parámetros y configuraciones:

```

# Parameters de openweathermap.org API
API_KEY = '13e4657fc0d134982cf3f04e66916d2f'
CITY = 'Tineo'
URL =
f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}&un
its=metric"

WEATHER_TIMER = 5 #min

# Simulation settings
SIMULATION_WIND_IS_USED = True # False -> Use real weather Forecast wind
speed
SIMULATION_WIND_SPEED = 5.6 # m/s

SIMULATION_TEMPERATURE_IS_USED = True # False -> Use real weather
Forecast temperature
SIMULATION_TEMPERATURE = 15.0 # °C
SIMULATION_HUMIDITY = 60
SIMULATION_INDOOR_TEMPERATURE = 17.0 # °C

# Wind Ggenerator V136-4.2 MW parameters
WIND_SPEED_MIN = 4
WIND_SPEED_MAX = 25

```


WIND_POWER_MIN = 50 # kW - 4 m/s

WIND_POWER_MAX = 3000 # kW - 25 m/s

WIND_GENERATORS_NUMBER = 8

Hydrogen Generator

HYDROGEN_FUEL_SELL_POWER = 2.5 # MW

SERVERS Operating range according to ASHRAE: 18–27 °C

SERVER_TEMPERATURE_MIN = 18

SERVER_TEMPERATURE_MAX = 27

SERVERS power load

SERVER_POWER = 10 # kW

SERVER_NUMBERS = 20

SERVER_USE_RATE = 0.85 # 85%

SERVER_ADDITIONAL_LOAD = 8.5 # kW

FREE COOLING FUN load

COOLING_FUN_POWER = 0.5 # 500 W

OUTDOOR_TEMPERATURE_MAX = 18

OUTDOOR_HUMIDITY_MAX = 70 # 70%

HEAT PUMP load

HEAT_PUMP_POWER = 2.86 # > SERVER LOAD = 10 kW / 3.5 (EER) \approx 2.86 kW

Debug mode

DEBUG_MODE = False # Set False to disable the debugging lights

Configuration de GPIO (BCM)

GPIO_WIND_GENERATOR_POWER_IN = 26

GPIO_HYDROGEN_FUEL_SELL_POWER_ON = 19

GPIO_HYDROGEN_GENERATION_ON = 13

GPIO_ELECTRICAL_GRID_POWER_IN = 6

GPIO_ELECTRICAL_GRID_POWER_OUT = 5

GPIO_FREE_COOLER_FUN_POWER_ON = 11

GPIO_HEAT_PUMP_POWER_ON = 12

GPIO_INDOOR_TEMP_HUMIDITY_SENSOR = 17

Types: int, str, float, bool

PARAMETERS = {

 "Open Weather Map parameters": [

 ("CITY", str, 200, "City name"),

 ("API_KEY", str, 400, "API key"),

 ("URL", str, 400, "API URL address"),

 ("WEATHER_TIMER", int, 80, "Weather update timer, min"),

],

 "Simulation settings": [

 ("SIMULATION_WIND_IS_USED", bool, None, "Use simulation wind speed"),

 ("SIMULATION_WIND_SPEED", float, 80, "Wind speed, in m/s"),

 ("spacer", None, None, None),

 ("SIMULATION_TEMPERATURE_IS_USED", bool, None, "Use simulation temperature"),

 ("SIMULATION_TEMPERATURE", float, 80, "Outdoor temperature"),

 ("SIMULATION_HUMIDITY", int, 80, "Outdoor humidity"),

 ("SIMULATION_INDOOR_TEMPERATURE", float, 80, "Indoor temperature"),

],

 "Wind Ggenerator V112-3.0 MW parameters": [

 ("WIND_SPEED_MIN", float, 80, "Minimum wind speed, m/c"),

 ("WIND_SPEED_MAX", float, 80, "Maximum wind speed, m/c"),

```

("WIND_POWER_MIN", int, 80, "Min wind Power, in kW"),
("WIND_POWER_MAX", int, 80, "Max wind Power, in kW"),
("spacer", None, None, None),
("WIND_GENERATORS_NUMBER", int, 60, "Wind generator number"),
],
"Hydrogen Generator": [
    ("HYDROGEN_FUEL_SELL_POWER", float, 80, "Hydrogen fuel sell power,
MW"),
],
"SERVERS Operating range according to ASHRAE": [
    ("SERVER_TEMPERATURE_MIN", float, 80, "Server rack temperature min,
°C"),
    ("SERVER_TEMPERATURE_MAX", float, 80, "Server rack temperature max,
°C"),
],
"SERVER rack power load": [
    ("SERVER_NUMBERS", int, 80, "Server rack number"),
    ("SERVER_POWER", float, 80, "Server rack power, kW"),
    ("SERVER_USE_RATE", float, 80, "Server rack use rate, %"),
    ("SERVER_ADDITIONAL_LOAD", float, 80, "Server rack additional load,
kW"),
],
"Free cooling fun load": [
    ("COOLING_FUN_POWER", float, 80, "Cooling fun power, kW"),
    ("OUTDOOR_TEMPERATURE_MAX", float, 80, "Maximum outdoor
temperature, °C"),
    ("OUTDOOR_HUMIDITY_MAX", float, 80, "Maximum outdoor humidity, %"),
],
"HEAT PUMP load": [
    ("HEAT_PUMP_POWER", float, 80, "Heat pump power, kW"),
],
}

```

HYDROGEN_FUEL_SELL_POWER = 1.5

ELECTROLYZER_POWER_MAX = 5.0

H2_MIN_SOC_PERCENT = 20

HP_OFF_TEMPERATURE = 25

6. **functions.py**

Funciones auxiliares adicionales de configuración:

```
from pyqtgraph.Qt.QtWidgets import QWidget
```

```
import random
```

```
import config
```

```
def colored_container(layout_type, label="", color=None, margins=(0, 0, 0, 0),  
spacing=0):
```

```
    """
```

```
    Creates a QWidget with a given layout and optional color debugging.
```

```
    :param debug_mode:
```

```
    :param layout_type: Layout class (e.g. QHBoxLayout or QVBoxLayout)
```

```
    :param label: Label for debugging
```

```
    :param color: Background color (if not specified, generated automatically)
```

```
    :param margins: Indents (left, top, right, bottom)
```

```
    :param spacing: Distance between elements
```

```
    :return: (QWidget, layout)
```

```
    """
```

```
    container = QWidget()
```

```
    layout = layout_type()
```

```
    layout.setContentsMargins(*margins)
```

```
    layout.setSpacing(spacing)
```

```
    container.setLayout(layout)
```

```

if config.DEBUG_MODE:
    if not color:
        # Generating a random pastel color
        r = lambda: random.randint(150, 220)
        color = f'rgb({r()}, {r()}, {r()})'

    container.setStyleSheet(
        f"""
        background-color: {color};
        border: 1px dashed gray;
        border-radius: 0px;
        margin: 0px;
        font-size: 10px;
        color: black;
        """
    )

    if label:
        from pyqtgraph.Qt.QtWidgets import QLabel
        layout.addWidget(QLabel(f'[{label}]'))

return container, layout

def formatValues(value):
    return f'{value:.2f}'.replace('.', ',')

# PowerFlowIndicator - Setup parameters:
# flow.setText("4.2 MW")
# flow.setLineLength(120)

```

```
# flow.setLineColor("#FF0000")

def parseFloat(value_str):
    # Remove units and spaces
    cleaned = value_str.replace("MW", "").replace("kW", "").strip()

    # Replace the comma with a period
    cleaned = cleaned.replace(",", ".")

    # Convert to a number
    value = float(cleaned)

    # Convert to a integer
    value_int = int(value)

    return value
```

7. **main.py**

Archivo principal de la aplicación, se ejecuta primero al iniciar el programa:

```
import sys

from pyqtgraph.Qt.QtWidgets import (QApplication)
from main_window import MainWindow

from controllers.app_logic import *

import config
config.DEBUG_MODE = False

if __name__ == '__main__':
```

```

app = QApplication(sys.argv)

window = MainWindow()
window.show()

# QTimer.singleShot(10, lambda: run_app_logic(
#     window.home_screen.energy_chart_widget,
#     window.home_screen.wind_generator_widget
# ))

# Application start
window.home_screen.add_log_event("Start application")
app_logic = AppLogic(window)

sys.exit(app.exec())

```

8. main_window.py

Ventana principal de la interfaz gráfica:

```

# from PySide6.QtCore import Qt
# from PySide6.QtWidgets import (QLabel, QMainWindow,
#     QHBoxLayout, QStackedWidget, QWidget)

from pyqtgraph.Qt.QtCore import Qt
from pyqtgraph.Qt.QtWidgets import (QLabel, QMainWindow,
    QHBoxLayout, QStackedWidget, QWidget)

from panels.side_menu import SideMenu
from panels.panel_home import PanelHome
from panels.panel_chart import PanelChart
from panels.panel_events_log import PanelEventsLog

```

```
import config
```

```
class MainWindow(QMainWindow):
```

```
    def __init__(self):
```

```
        super().__init__()
```

```
        self.setWindowTitle("Energy Interface")
```

```
        self.setGeometry(0, 0, 1920, 1080)
```

```
        self.showFullScreen()
```

```
        self.main_widget = QWidget()
```

```
        self.setCentralWidget(self.main_widget)
```

```
        h_layout = QHBoxLayout()
```

```
        h_layout.setContentsMargins(0, 0, 0, 0)
```

```
        h_layout.setSpacing(0)
```

```
        self.main_widget.setLayout(h_layout)
```

```
        self.main_widget.setStyleSheet("""
```

```
            background: qlineargradient(
```

```
                x1: 0, y1: 0,
```

```
                x2: 0, y2: 1,
```

```
                stop: 0 #E9F0FA,
```

```
                stop: 0.4 #d7f6f6,
```

```
                stop: 1.0 #80B3B3
```

```
            );
```

```
        """)
```

```
        self.side_menu = SideMenu(navigation_callback=self.display_screen)
```

```
        h_layout.addWidget(self.side_menu)
```



```

self.stack = QStackedWidget(self.main_widget)

self.home_screen = PanelHome(self.stack)

# self.scheme_screen = QLabel("[Scheme screen]")

self.chart_screen = PanelChart(self.stack)

self.log_screen = PanelEventsLog(self.stack)


# for screen in [self.home_screen, self.scheme_screen, self.chart_screen,
self.log_screen]:
    for screen in [self.home_screen, self.chart_screen, self.log_screen]:
        if isinstance(screen, QLabel):
            screen.setAlignment(Qt.AlignmentFlag.AlignCenter)
            self.stack.addWidget(screen)

h_layout.addWidget(self.stack)


def display_screen(self, index):
    self.stack.setCurrentIndex(index)

    if index == 2:
        # Update Event Log
        self.log_screen.update_log_event()


def update_app_background(self, current_servers_temperature):
    if current_servers_temperature > config.SERVER_TEMPERATURE_MAX:
        # HOT
        bg_color = "#FE9494"
    else:
        if current_servers_temperature > ((config.SERVER_TEMPERATURE_MAX -
config.SERVER_TEMPERATURE_MIN) / 2 +
config.SERVER_TEMPERATURE_MIN):
            # NORMAL

```

```
        bg_color = "#EBEC81"
    else:
        # COLD
        bg_color = "#80B3B3"

self.main_widget.setStyleSheet(f"""
    background: qlineargradient(
        x1: 0, y1: 0,
        x2: 0, y2: 1,
        stop: 0 #E8F8FA,
        stop: 0.4 #99FFFF,
        stop: 1.0 {bg_color}
    );
""")
```

Anexo V: Viabilidad económica

Se considera que cada aerogenerador debe producir al menos 750 kW en todo momento para que el centro de datos esté correctamente alimentado, tal y como se refleja en la figura 7.

$$P\left(\frac{Kw}{\text{aerogenerador}}\right) = \frac{52\,560\,000 \frac{kWh}{\text{año}}}{8 \text{ aerogeneradores} * 24 \frac{h}{\text{día}} * 365 \frac{\text{días}}{\text{año}}} = 750 \text{ kW}$$

Figura 7. Potencia que debe aportar cada aerogenerador para abastecer el centro de datos en todo momento

Se tiene en cuenta la curva de capacidad de los aerogeneradores, tal y como se aprecia en la figura 8 y sabiendo que para que no se requiera el funcionamiento del bloque de hidrógeno cada aerogenerador debe trabajar al menos a 750 kW.

Con todo ello, realizaremos una regresión lineal para saber que velocidad de viento mínima se requiere en cada momento, siendo ésta 5,743 m/s. (The Wind Power, s.f.)

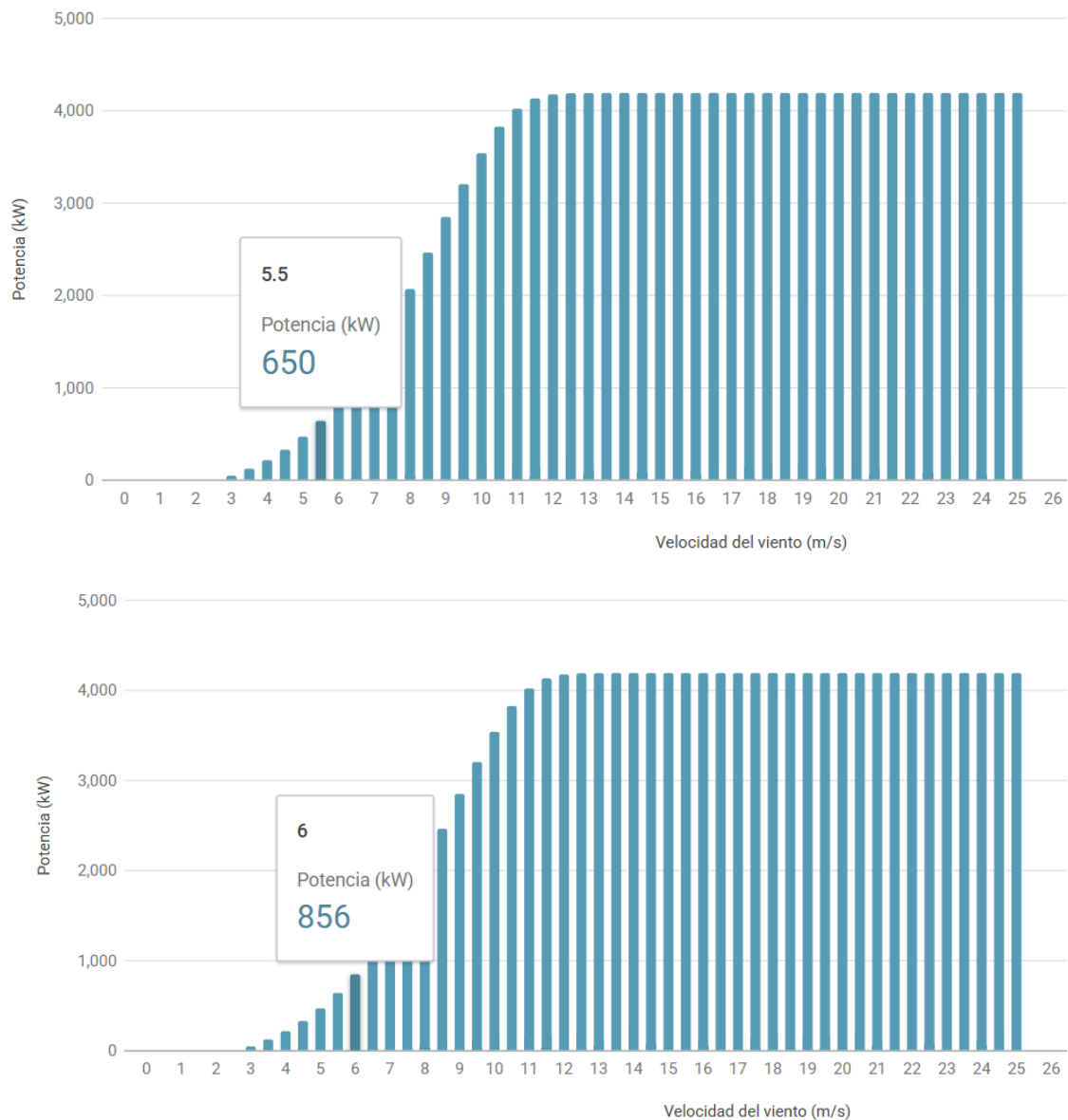


Figura 8. Curva de potencia del aerogenerador Vestas V136-4.2MW

Se tiene en cuenta que el bloque de hidrógeno consume 5 MW que, al considerar 365 días al año y 24 horas al día, se traducen en 43,8 GWh/año si se tuviera funcionando el sistema el 100% del tiempo. Así mismo, teniendo en cuenta los parámetros calculados para la distribución Weibull usando Windographer que nos indican la probabilidad de vientos futuros inferiores a 5.743 m/s, se realiza el cálculo plasmado en la figura 9.

$$P\left(\text{viento} < 5.743 \frac{\text{m}}{\text{s}}\right) = 1 - e^{\left(-\left(\frac{5.743}{7.072}\right)^{2.5}\right)} = 0.4480 \cong 44.8 \%$$

Figura 9. Tiempo que el viento es inferior a 5.743 m/s

Por todo ello, el consumo del bloque de hidrógeno será como máximo del 44.8 % de 43,8 GWh/año, lo que suponen 19,624 GWh/año.

A continuación, se ha realizado un “cash flow” descrito tal y como se aprecia en la tabla. Hay que destacar que no se han tenido en cuenta como un 5% los gastos derivados de los costes de arrendamiento del terreno al pertenecer dichos terrenos al ayuntamiento para el que se está realizando el proyecto. Se considera también que al tratarse de un ahorro y no de ingresos como tal, el beneficio neto no está sujeto a impuestos y puede ser considerado directamente como “cash flow”. Se aprecia en la tabla 7, los valores de producción con una degradación anual del 0,4%, y consumos, así como excedentes si hubiera. (3E, s.f.)

Año	Producción anual (GWh/año)	Energía centro (GWh/año)	Energía hidrógeno (GWh/año)	Excedentes (GWh/año)
1	73,55	52,56	19,62	1,36
2	73,25	52,56	19,62	1,07
3	72,96	52,56	19,62	0,77
4	72,67	52,56	19,62	0,48
5	72,38	52,56	19,62	0,19
6	72,09	52,56	19,62	-0,10
7	71,80	52,56	19,62	-0,39
8	71,51	52,56	19,62	-0,67
9	71,23	52,56	19,62	-0,96
10	70,94	52,56	19,62	-1,24
11	70,66	52,56	19,62	-1,53
12	70,37	52,56	19,62	-1,81
13	70,09	52,56	19,62	-2,09
14	69,81	52,56	19,62	-2,37
15	69,53	52,56	19,62	-2,65
16	69,25	52,56	19,62	-2,93
17	68,98	52,56	19,62	-3,21
18	68,70	52,56	19,62	-3,48
19	68,43	52,56	19,62	-3,76
20	68,15	52,56	19,62	-4,03

Tabla 2. Flujo de caja a 20 años del proyecto

A continuación, en las tablas 8, 9, 10 ,11, 12, y 13, aparece realizado el desglose de dicho “cash flow” para las diferentes casuísticas estudiadas. Considerando un escenario conservador, realista y optimista sin subvenciones y con subvenciones respectivamente.

Dicho escenario conservador consta con un precio de la electricidad de 59 euros, y un 10% de subvención si aplica. El escenario realista consta con un precio de la electricidad de 62 euros, y un 20% de subvención si aplica. Y por último, el optimista consta con un precio de la electricidad de 65 euros, y un 30% de subvención si aplica.

Precio electricidad [€/MWh]	Coste electricidad de la red [€/año]	Ahorro neto [€/año]	Gasto O&M con el 2% de subida del IPC anual [€/año]	Gasto terreno (5% del ahorro) [€/año]	Gastos totales [€/año]	Resultado bruto [€/año]	Beneficio neto [€/año]	Cash Flow [€/año]
59,0	3101040,0	3101040,0	104000,0	155052,0	259052,0	2841988,0	2841988,0	2841988,0
59,0	3101040,0	3101040,0	106080,0	155052,0	261132,0	2839908,0	2839908,0	2839908,0
59,0	3101040,0	3101040,0	108201,6	155052,0	263253,6	2837786,4	2837786,4	2837786,4
59,0	3101040,0	3101040,0	110365,6	155052,0	265417,6	2835622,4	2835622,4	2835622,4
59,0	3101040,0	3101040,0	112572,9	155052,0	267624,9	2833415,1	2833415,1	2833415,1
59,0	3101040,0	3095305,2	114824,4	154765,3	269589,7	2825715,6	2825715,6	2825715,6
59,0	3101040,0	3078292,7	117120,9	153914,6	271035,5	2807257,2	2807257,2	2807257,2
59,0	3101040,0	3061348,3	119463,3	153067,4	272530,7	2788817,6	2788817,6	2788817,6
59,0	3101040,0	3044471,6	121852,6	152223,6	274076,2	2770395,5	2770395,5	2770395,5
59,0	3101040,0	3027662,5	124289,6	151383,1	275672,8	2751989,7	2751989,7	2751989,7
59,0	3101040,0	3010920,6	126775,4	150546,0	277321,4	2733599,1	2733599,1	2733599,1
59,0	3101040,0	2994245,6	129310,9	149712,3	279023,2	2715222,4	2715222,4	2715222,4
59,0	3101040,0	2977637,4	131897,1	148881,9	280779,0	2696858,4	2696858,4	2696858,4
59,0	3101040,0	2961095,6	134535,1	148054,8	282589,9	2678505,7	2678505,7	2678505,7
59,0	3101040,0	2944619,9	137225,8	147231,0	284456,8	2660163,1	2660163,1	2660163,1
59,0	3101040,0	2928210,2	139970,3	146410,5	286380,8	2641829,4	2641829,4	2641829,4
59,0	3101040,0	2911866,1	142769,7	145593,3	288363,0	2623503,1	2623503,1	2623503,1
59,0	3101040,0	2895587,4	145625,1	144779,4	290404,5	2605182,9	2605182,9	2605182,9
59,0	3101040,0	2879373,7	148537,6	143968,7	292506,3	2586867,4	2586867,4	2586867,4
59,0	3101040,0	2863225,0	151508,4	143161,2	294669,6	2568555,4	2568555,4	2568555,4

VAN (cash)	Inversión inicial [€]	VAN (6%)	TIR
31.671.204,75 €	46250000	-14.578.795,25 €	1,68%

Tabla 3. Cash flow considerando la inversión inicial sin subvenciones y 59 €/MWh

Precio electricidad [€/MWh]	Coste electricidad de la red [€/año]	Ahorro neto [€/año]	Gasto O&M con el 2% de subida del IPC anual [€/año]	Gasto terreno (5% del ahorro) [€/año]	Gastos totales [€/año]	Resultado bruto [€/año]	Beneficio neto [€/año]	Cash Flow [€/año]
62,0	3258720,0	3258720,0	104000,0	162936,0	266936,0	2991784,0	2991784,0	2991784,0
62,0	3258720,0	3258720,0	106080,0	162936,0	269016,0	2989704,0	2989704,0	2989704,0
62,0	3258720,0	3258720,0	108201,6	162936,0	271137,6	2987582,4	2987582,4	2987582,4
62,0	3258720,0	3258720,0	110365,6	162936,0	273301,6	2985418,4	2985418,4	2985418,4
62,0	3258720,0	3258720,0	112572,9	162936,0	275508,9	2983211,1	2983211,1	2983211,1
62,0	3258720,0	3252693,6	114824,4	162634,7	277459,1	2975234,5	2975234,5	2975234,5
62,0	3258720,0	3234816,1	117120,9	161740,8	278861,7	2955954,4	2955954,4	2955954,4
62,0	3258720,0	3217010,1	119463,3	160850,5	280313,8	2936696,3	2936696,3	2936696,3
62,0	3258720,0	3199275,3	121852,6	159963,8	281816,3	2917458,9	2917458,9	2917458,9
62,0	3258720,0	3181611,4	124289,6	159080,6	283370,2	2898241,2	2898241,2	2898241,2
62,0	3258720,0	3164018,2	126775,4	158200,9	284976,3	2879041,9	2879041,9	2879041,9
62,0	3258720,0	3146495,4	129310,9	157324,8	286635,7	2859859,7	2859859,7	2859859,7
62,0	3258720,0	3129042,7	131897,1	156452,1	288349,3	2840693,4	2840693,4	2840693,4
62,0	3258720,0	3111659,8	134535,1	155583,0	290118,1	2821541,7	2821541,7	2821541,7
62,0	3258720,0	3094346,4	137225,8	154717,3	291943,1	2802403,3	2802403,3	2802403,3
62,0	3258720,0	3077102,2	139970,3	153855,1	293825,4	2783276,8	2783276,8	2783276,8
62,0	3258720,0	3059927,1	142769,7	152996,4	295766,1	2764161,0	2764161,0	2764161,0
62,0	3258720,0	3042820,6	145625,1	152141,0	297766,1	2745054,5	2745054,5	2745054,5
62,0	3258720,0	3025782,6	148537,6	151289,1	299826,7	2725955,8	2725955,8	2725955,8
62,0	3258720,0	3008812,7	151508,4	150440,6	301949,0	2706863,7	2706863,7	2706863,7

VAN (cash)	Inversión inicial [€]	VAN (6%)	TIR
33.352.555,19 €	46250000	-12.897.444,81 €	2,22%

Tabla 4. Cash flow considerando la inversión inicial sin subvenciones y 62 €/MWh

Precio electricidad [€/MWh]	Coste electricidad de la red [€/año]	Ahorro neto [€/año]	Gasto O&M con el 2% de subida del IPC anual [€/año]	Gasto terreno (5% del ahorro) [€/año]	Gastos totales [€/año]	Resultado bruto [€/año]	Beneficio neto [€/año]	Cash Flow [€/año]
65,0	3416400,0	3416400,0	104000,0	170820,0	274820,0	3141580,0	3141580,0	3141580,0
65,0	3416400,0	3416400,0	106080,0	170820,0	276900,0	3139500,0	3139500,0	3139500,0
65,0	3416400,0	3416400,0	108201,6	170820,0	279021,6	3137378,4	3137378,4	3137378,4
65,0	3416400,0	3416400,0	110365,6	170820,0	281185,6	3135214,4	3135214,4	3135214,4
65,0	3416400,0	3416400,0	112572,9	170820,0	283392,9	3133007,1	3133007,1	3133007,1
65,0	3416400,0	3410082,0	114824,4	170504,1	285328,5	3124753,5	3124753,5	3124753,5
65,0	3416400,0	3391339,5	117120,9	169567,0	286687,9	3104651,6	3104651,6	3104651,6
65,0	3416400,0	3372671,9	119463,3	168633,6	288096,9	3084575,0	3084575,0	3084575,0
65,0	3416400,0	3354078,9	121852,6	167703,9	289556,5	3064522,4	3064522,4	3064522,4
65,0	3416400,0	3335560,4	124289,6	166778,0	291067,6	3044492,7	3044492,7	3044492,7
65,0	3416400,0	3317115,9	126775,4	165855,8	292631,2	3024484,7	3024484,7	3024484,7
65,0	3416400,0	3298745,2	129310,9	164937,3	294248,2	3004497,0	3004497,0	3004497,0
65,0	3416400,0	3280448,0	131897,1	164022,4	295919,5	2984528,4	2984528,4	2984528,4
65,0	3416400,0	3262223,9	134535,1	163111,2	297646,3	2964577,7	2964577,7	2964577,7
65,0	3416400,0	3244072,8	137225,8	162203,6	299429,4	2944643,4	2944643,4	2944643,4
65,0	3416400,0	3225994,3	139970,3	161299,7	301270,0	2924724,3	2924724,3	2924724,3
65,0	3416400,0	3207988,1	142769,7	160399,4	303169,1	2904818,9	2904818,9	2904818,9
65,0	3416400,0	3190053,9	145625,1	159502,7	305127,8	2884926,1	2884926,1	2884926,1
65,0	3416400,0	3172191,4	148537,6	158609,6	307147,2	2865044,2	2865044,2	2865044,2
65,0	3416400,0	3154400,4	151508,4	157720,0	309228,4	2845172,0	2845172,0	2845172,0

VAN (cash)	Inversión inicial [€]	VAN (6%)	TIR
35.033.905,63 €	46250000	-11.216.094,37 €	2,75%

Tabla 5. Cash flow considerando la inversión inicial sin subvenciones y 65 €/MW

VAN (cash)	Inversión inicial con 10% de subvención [€]	VAN (6%)	TIR
33.352.555,19 €	41625000	-8.272.444,81 €	3,37%

Tabla 6. Cash flow considerando la inversión inicial con subvenciones del 10% y 62 €/MWh

VAN (cash)	Inversión inicial con 10% de subvención [€]	VAN (6%)	TIR
33.352.555,19 €	37000000	-3.647.444,81 €	4,73%

Tabla 7. Cash flow considerando la inversión inicial con subvenciones del 20% y 62 €/MWh

VAN (cash)	Inversión inicial con 10% de subvención [€]	VAN (6%)	TIR
33.352.555,19 €	32375000	977.555,19 €	6,38%

Tabla 8. Cash flow considerando la inversión inicial con subvenciones del 30% y 62 €/MWh

Todo ello puede resumirse en el análisis de sensibilidad plasmado en la tabla 21. Destacar que se ha considerado el payback como el año en el que la suma de los beneficios netos obtenidos anteriormente supera a la inversión inicial.

Sin subvenciones	Conservador	Realista	Optimista
Precio energía	59	62	65
TIR	1,68%	2,22%	2,75%
Payback	17	16	16

A 62 €/MWh	Conservador	Realista	Optimista
Subvención (% de la inversión recibida en ayudas)	10%	20%	30%
TIR	3,37%	4,73%	6,38%
Payback	16	13	12

Tabla 9. Estudio de sensibilidad del proyecto

El ROI (Retorno de Inversión) del proyecto para el supuesto de una subvención del 30% del valor inicial invertido y con un precio de la electricidad de 62 €/MWh es el que se aprecia en la figura 29, de que se sabe que se ahorrará un 77,76 % del coste de la implementación del proyecto, respecto a usar electricidad directamente de la red.

$$ROI (\%) = \frac{57550136 - 46\,250\,000 * 70\%}{46\,250\,000 * 70\%} * 100 = 77,76\%$$

$$ROI \left(\frac{\%}{año} \right) = \frac{77,76\%}{20\,años} = 3,88 \frac{\%}{año}$$

Figura 10. Retorno de la inversión

Todo ello, sumado a las ganancias que ocasiona un centro de datos, cuya rentabilidad no es objeto de este informe, hacen pensar que la reducción de los costes en un 3,88 % anual, lo cual de, sí sería motivo suficiente para justificar la implementación del proyecto. Hay que destacar que, en la rentabilidad no se ha tenido en cuenta la venta de excedentes a la red, debido a la escasa cantidad de los mismos, y a que los gastos derivados de la infraestructura necesaria para realizarlos serían superiores a los ingresos que se pudieran obtener. Por lo que incluso en el caso más favorable el proyecto no saldría rentable.

Si se tienen en cuenta posibles subvenciones que se pudiera acoger nuestro proyecto, como la concesión de ayudas a las inversiones en transformación, comercialización y/o desarrollo que proporciona el Principado de Asturias cofinanciadas por: Fondos europeos Feder (60 %), Fondos del Principado de Asturias (28 %) y Fondos AGE (12 %), de un mínimo del 25 % del valor del proyecto, se podría obtener una rentabilidad superior, tal y como se plasma en la figura 30. (Gobierno del Principado de Asturias, 2025)

$$ROI (\%) = \frac{57\,550\,136 - 46\,250\,000 * 75\%}{46\,250\,000 * 75\%} * 100 = 65,91\%$$

$$ROI \left(\frac{\%}{año} \right) = \frac{65,91\%}{20\,años} = 3,29 \frac{\%}{año}$$

Figura 11. Retorno de la inversión con subvenciones

Con todo ello, se consigue un ahorro anual mayor, que sí posibilita la rentabilidad de una implementación renovable para el abastecimiento eléctrico del centro de datos. Una vez más se ha dejado fuera de los cálculos los 5 000 000 de euros que constituyen en centro

de datos, ya que la rentabilidad del mismo no es objeto de este proyecto, solamente la rentabilidad de un suministro eléctrico de energía renovable en lugar de uno que la tome directamente de la red.