



Carpeta Técnica

7mo 2da Comision A

Año 2025

Bourlot Ezequiel
Fernandez Lorenzo
Domoñi Federico
Vignetta Joaquin

Índice general

1. Preámbulo	5
1.1. Nuestro contacto	5
1.2. ¿Quienes somos?	6
1.3. Docentes a cargo	9
1.4. Agradecimientos	9
1.5. Lenguajes usados	9
1.6. Programas utilizados	9
2. Introducción	11
2.1. ¿Qué es OIBsleep?	11
2.2. Utilidades del proyecto	12
2.3. Impacto en Argentina	12
2.3.1. Impacto en la salud pública	12
2.3.2. Impacto en hospitales y centros de salud	12
2.3.3. Impacto en el mercado argentino	13
2.4. ¿Por qué OIBsleep?	13
2.4.1. Usuarios que ayuda	13
2.4.2. Limitaciones de los métodos tradicionales	13
2.4.3. La importancia del confort térmico durante el sueño	13
2.4.4. Una solución personalizada y directa	14
2.4.5. Confort desde el primer momento	14
3. Estructura	15
3.1. ¿Cómo está estructurada la cama?	15
3.2. ¿Cómo está estructurado el brazalete?	16
3.3. ¿Cómo están estructurados los tanques?	21
3.3.1. ¿Cómo están colocados?	22
3.3.2. Cálculos de temperatura para el agua fría	22
4. Circuitos	24
4.1. Resumen	24
4.1.1. Pulsera	24
4.1.2. Bombas de agua	25
5. Código	26
5.1. Resumen	26
5.2. Explicación Técnica Detallada - Presence Detector	27
5.2.1. Arquitectura General del Sistema	27
5.2.2. Clase Principal: BedPresenceDetector	27
5.2.3. Método Principal: detect_presence()	28
5.2.4. Mecanismo de Histéresis	30
5.2.5. Gestión del Baseline Térmico	31

5.2.6. Métodos de Utilidad y Debugging	31
5.2.7. Consideraciones de Diseño	31
5.2.8. Retorno del Método Principal	32
5.3. large Diagrama presence_detector	33
5.4. Módulo de análisis de señales fisiológicas: analyzer.py	34
5.4.1. Arquitectura general	34
5.4.2. Funciones principales	34
5.4.3. Entradas y salidas	36
5.4.4. Supuestos y consideraciones temporales	36
5.4.5. Casos límite y estrategias de validación	37
5.4.6. Recomendaciones para uso en producción	37
5.4.7. Formato de salida y consumo de resultados	37
5.4.8. Ejemplo de uso	38
5.4.9. Referencias sugeridas	38

Índice de figuras

3.1.	Vista inferior del modelo 3D de la cama con sistema de tubos	16
3.2.	Diseño 3D del brazalete - Vista frontal con perforaciones de ventilación	17
3.3.	Diseño 3D del brazalete - Vista lateral detallada	18
3.4.	Brazaletes impresos y terminados	19
3.5.	Brazalete con compartimiento para sensores	20
3.6.	Modelo 3D del compartimiento de sensores del brazalete	20
3.7.	Tanque frío sin las celdas	21

Capítulo 1

Preámbulo

1.1. Nuestro Contacto



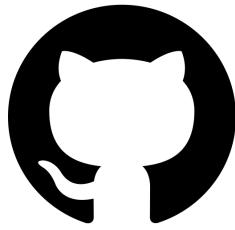
@oibsleep



oibsleep@gmail.com



oib-sleep.vercel.app



github.com/impatrqb/OIB_Sleep

1.2. Quienes somos?



Bourlot Ezequiel



Encargado del diseño electrónico, programación y control térmico del sistema.



Federico Domoñi



Responsable del diseño mecánico y estructura del sistema de tanques.

**Fernández Lorenzo**

Encargado de documentación técnica, cálculos térmicos y presentación.

**Vignetta Joaquín**

Responsable del diseño visual, edición de video y comunicación del proyecto.



1.3. Docentes a cargo

- ★ Medina Sergio
- ★ Carlassara Fabrizio
- ★ Palmier Diego
- ★ Bianco Carlos
- ★ Arguello Gabriel
- ★ Solomiewicz Federico

1.4. agradecimientos

Agradecemos a la Asociación Cooperadora del IMPA.

Agradecemos a Simmons Por el colchón, que fue lo principal en nuestro proyecto

Agradecemos a layer by layer por los acoples para las mangueras

Agradecemos al personal docente de la institución por el inmenso apoyo que nos brindaron.

1.5. Lenguajes usados

- HTML5
- CSS3
- JavaSCript
- Python
- C++
- LaTeX

1.6. Programas utilizados

- Visual Studio Code
- OverLeaf
- PlatformIO

- Raspberry PI Imager 
- PowerShell 
- Ubuntu WSL 
- KiCad 
- Blender 
- AutoCAD 
- Ultimaker cura 
- Fusion3D 
- Word 
- Figma 
- Ibis Paint 

Capítulo 2

Introducción

2.1. ¿Qué es OIBsleep?

La cama ortopédica OIB sleep Es un colchón de uso hospital o personal que cuenta con sistema de calefacción/refrigeración que se regula mediante diferentes mediciones del paciente. Esta cama sirve para mejorar la calidad de sueño y prevenir enfermedades como la migraña al regular la temperatura en búsqueda de mantener el sueño REM (profundo) la mayor cantidad de tiempo posible. También se busca solucionar diferentes problemas de salud mediante esta regulación como, por ejemplo:

- Pacientes con problemas de circulación entre ellos diabéticos o personas con problema de artritis suelen tener manos y pies fríos, y pueden no notar bien si están demasiado fríos o calientes (especialmente en el caso de la neuropatía diabética).
- Adultos mayores, ya que la regulación de la temperatura disminuye con la edad.
- Personas con fibromialgia o dolor crónico, el calor puede aliviar el dolor muscular y articular mientras que el frío reduce la inflamación.
- Personas con problemas de sueño como insomnio o despertares nocturnos debido a la temperatura

Se puede conseguir regular la temperatura del colchón mediante un sistema con tubos de agua que se regule con mediciones que se hacen con el colchón mismo o con una pulsera. Esto le ayudaría a los hospitales a mantener a varios pacientes en una misma habitación con las temperaturas necesarias para cada uno.

2.2. Utilidades del proyecto

Utilidades del proyecto: OIB sleep es un proyecto planteado para hospitales que busca ayudar a las personas que tienen problemas debido a la falta de sueño y también a quienes la temperatura puede ayudar con sus respectivas enfermedades. Tomamos la temperatura como base, ya que El entorno térmico es uno de los factores más importantes que pueden afectar el sueño humano". Los efectos típicos de la exposición al calor o al frío son un aumento del estado de vigilia y una disminución del sueño de movimientos oculares rápidos (REM) y del sueño de ondas lentas.

2.3. Impacto en Argentina

En Argentina, los problemas relacionados con el sueño representan un desafío creciente para la salud pública. Según un estudio de la Universidad Favaloro, **el 75 % de la población presenta algún tipo de alteración del sueño**, lo cual evidencia la necesidad de soluciones innovadoras que contribuyan a mejorar el descanso y la calidad de vida.

2.3.1. Impacto en la salud pública

- Mejora la calidad del sueño, reduciendo riesgos asociados a insomnio, migrañas y problemas cardiovasculares.
- Favorece a pacientes con enfermedades crónicas como artritis, fibromialgia, diabetes y problemas de circulación, donde la regulación térmica es un factor clave.
- Contribuye al bienestar de adultos mayores, un grupo poblacional que representa más del 11 % de los argentinos y que suele tener dificultades para regular la temperatura corporal durante el descanso.

2.3.2. Impacto en hospitales y centros de salud

- Permite la **regulación de la temperatura individual en habitaciones compartidas**, lo cual mejora la atención personalizada y optimiza el uso de recursos.
- Potencial ahorro energético al evitar climatizar espacios completos, reduciendo los costos operativos.

2.3.3. Impacto en el mercado argentino

- Dada la alta prevalencia de trastornos del sueño, existe un **mercado amplio** para este tipo de tecnologías tanto en hospitales como en el uso domiciliario.
- La producción nacional del sistema puede fomentar la **industria tecnológica local**, generando empleo y reduciendo costos de importación.

2.4. ¿Por qué OIBsleep?

2.4.1. Usuarios que ayuda

OIB Sleep está diseñado para mejorar la salud del sueño tanto en entornos hospitalarios como domésticos. Esto se logra a través de la regulación inteligente de la temperatura, ayudando a grupos como:

- adultos mayores
- Personas diabéticas
- Personas con insomnio
- Personas con fibromialgia

2.4.2. Limitaciones de los métodos tradicionales

El aire acondicionado, la estufa y otros métodos permiten modificar la temperatura del ambiente, pero ninguno de estos sistemas se adapta a las necesidades térmicas específicas de tu cuerpo ni responde a los cambios que experimentás mientras dormís. Estos métodos actúan de forma general sobre toda la habitación, sin tener en cuenta las variaciones naturales de la temperatura corporal a lo largo de la noche. Además, al climatizar todo el espacio en lugar de centrarse únicamente en la persona, generan un consumo energético considerablemente mayor, lo que se traduce en un uso ineficiente de la energía y en un impacto ambiental más alto.

2.4.3. La importancia del confort térmico durante el sueño

La temperatura corporal no permanece constante durante el sueño: tiende a descender durante las fases más profundas y vuelve a elevarse de manera natural al

acerarse el momento del despertar. Este ciclo térmico cumple un rol fundamental en la regulación del descanso y en la transición entre las distintas etapas del sueño. Cuando el entorno no acompaña estos cambios —por ejemplo, si la habitación se mantiene demasiado cálida o demasiado fría—, el cuerpo debe esforzarse para compensar esa diferencia, lo que puede provocar microdespertares e interrumpir el descanso, reduciendo así la calidad y la continuidad del sueño.

2.4.4. Una solución personalizada y directa

OIB Sleep actúa directamente sobre la cama, en contacto con tu cuerpo, ajustando la temperatura del colchón en tiempo real según tus necesidades. De esta forma, el sistema responde de manera inteligente a los cambios térmicos que se producen durante el sueño, brindando una sensación de confort constante y personalizada. Al intervenir únicamente sobre el colchón y no sobre el ambiente completo, permite alcanzar el confort térmico ideal con un consumo energético significativamente menor y sin alterar la temperatura del entorno.

2.4.5. Confort desde el primer momento

Más allá de su control inteligente, OIB Sleep ofrece funciones pensadas para maximizar el confort desde el primer momento. Permite precalentar la cama en invierno o enfriarla en verano antes de acostarte, logrando así que el colchón alcance la temperatura ideal incluso antes de que te recuestes. De esta manera, transforma el momento de ir a dormir en una experiencia más placentera y personalizada, adaptándose a tus preferencias y al clima de cada estación.

Capítulo 3

Estructura

3.1. ¿Como esta estructurada la cama?

La cama cuenta con un colchón especialmente modificado en cuyo interior se instalaron Caños de cobre de 1/4 x 1 m esmaltados. Para colocarlos, utilizamos una plegadora que nos permitió insertar los tubos dentro del colchón realizando una serie de dobleces estratégicos, asegurando así una distribución uniforme del calor a lo largo de toda la superficie. El sistema está compuesto por dos circuitos independientes: uno por el que circula agua caliente y otro por el que circula agua fría. Ambos tubos están conectados a sus respectivos tanques, lo que permite regular la temperatura del colchón de manera precisa y eficiente.

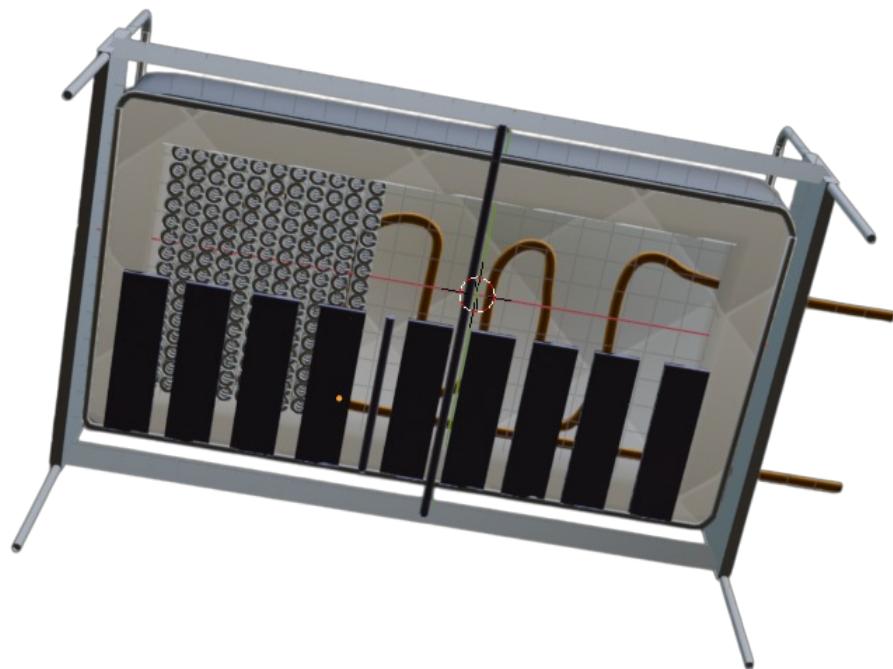


Figura 3.1: Vista inferior del modelo 3D de la cama con sistema de tubos

3.2. ¿Como esta estructurado el brazalete?

El brazalete fue diseñado mediante un modelo 3D desarrollado en Blender y fabricado con material PLA, elegido por su equilibrio entre solidez, ligereza y flexibilidad. El diseño incluye una serie de perforaciones distribuidas estratégicamente para permitir una adecuada ventilación y el paso del aire, garantizando así mayor comodidad durante su uso. En la siguiente imagen se puede observar la estructura y disposición de dichos orificios.



Figura 3.2: Diseño 3D del brazalete - Vista frontal con perforaciones de ventilación



Figura 3.3: Diseño 3D del brazalete - Vista lateral detallada

Además, el brazalete está recubierto con una capa de goma espuma, lo que mejora la sensibilidad al tacto y aporta una sensación más cómoda al contacto con la piel. Una vez impreso, el modelo fue lijado cuidadosamente para eliminar imperfecciones y lograr una superficie uniforme, optimizando tanto su estética como su funcionalidad.



Figura 3.4: Brazaletes impresos y terminados

El brazalete cuenta con un compartimiento destinado a alojar los diferentes sensores, diseñado para mantenerlos firmemente en su lugar y facilitar su conexión. Este compartimiento se fija al brazalete mediante abrojos (velcro), lo que permite retirarlo o ajustarlo fácilmente según sea necesario, garantizando al mismo tiempo estabilidad y comodidad durante el uso.

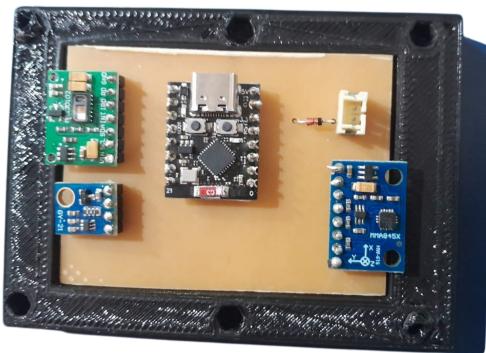


Figura 3.5: Brazalete con compartimiento para sensores

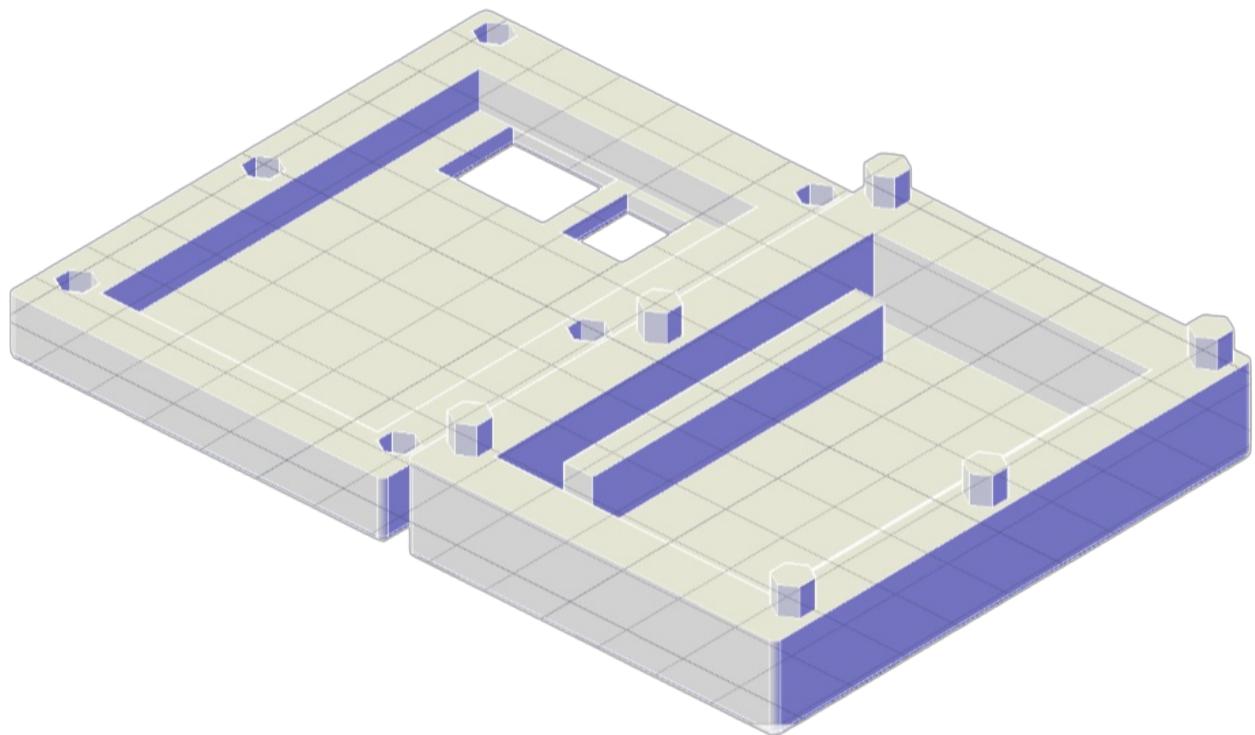


Figura 3.6: Modelo 3D del compartimiento de sensores del brazalete

3.3. ¿Como están estructurados los tanques?

En este proyecto se utilizan dos tanques principales, uno destinado al agua fría y otro al agua caliente, los cuales desempeñan un papel fundamental en el sistema de regulación térmica. Ambos tanques están aislados térmicamente con espuma de poliuretano, un material de alta densidad que permite mantener la temperatura interna estable, reduciendo significativamente las pérdidas de calor o de frío hacia el exterior.

El tanque de agua fría está equipado con seis celdas Peltier, cada una acompañada de su disipador de calor y su correspondiente cooler, también tienen una placa de metal del la parte del tanque para que se pase mejor el calor. Estas celdas funcionan mediante un principio termoeléctrico: al recibir corriente, una de sus caras absorbe el calor del interior del tanque (enfriando el agua), mientras que la otra lo disipa hacia afuera con la ayuda de los coolers. De esta forma, se logra un proceso de refrigeración activa y controlada, manteniendo la temperatura del agua dentro de los rangos deseados.



Figura 3.7: Tanque frío sin las celdas

Por otro lado, el tanque de agua caliente cuenta con una resistencia eléctrica encargada de elevar la temperatura del agua. Este sistema está supervisado por un

sensor de temperatura sumergible, que mide de manera continua el valor térmico del agua y permite pausar automáticamente el calentamiento cuando se alcanza una temperatura preestablecida, evitando sobrecalentamientos y optimizando el consumo energético.

En conjunto, ambos tanques trabajan de forma complementaria dentro del sistema, proporcionando un control térmico eficiente y preciso, esencial para garantizar el confort y la estabilidad del entorno en el que se implementa el proyecto.

3.3.1. ¿Como están colocados?

3.3.2. Cálculos de temperatura para el agua fría

El sistema de enfriamiento del tanque de agua fría se basa en la utilización de celdas Peltier TEC1-12706, cada una con una capacidad aproximada de transferencia de calor (Q_c) de 60 W cuando opera con una diferencia de temperatura de 10 °C y alimentada a 12 V con un consumo de 6 A (potencia total 72 W). En el diseño propuesto se contempla un tanque con una capacidad total de 20 litros, debidamente aislado con espuma de poliuretano para minimizar las pérdidas térmicas. La temperatura ambiente promedio se considera de 30 °C, y el objetivo es reducir la temperatura del agua a 10 °C.

El agua, con una masa de aproximadamente 20 kg (considerando una densidad de 1 kg/L), requiere una extracción de energía térmica calculada mediante la ecuación:

$$Q = m \cdot c \cdot \Delta T$$

donde c es el calor específico del agua (4186 J/kg·°C) y $\Delta T = 20$ °C (de 30 °C a 10 °C). De esta manera, la cantidad total de energía a retirar del sistema es de aproximadamente:

$$Q = 20 \text{ kg} \times 4186 \frac{\text{J}}{\text{kg} \cdot \text{°C}} \times 20 \text{ °C} = 1,674,400 \text{ J}$$

Cada celda Peltier TEC1-12706 puede retirar alrededor de 60 W (60 J/s) en condiciones óptimas; sin embargo, considerando las ineficiencias y pérdidas térmicas del sistema, se estima una eficiencia real del 40%

Para el sistema implementado, se utilizan **6 celdas Peltier TEC1-12706**, por lo que la potencia total efectiva de enfriamiento es:

$$P = 6 \times 24 \text{ W} = 144 \text{ W} = 144 \text{ J/s}$$

En estas condiciones, el tiempo teórico requerido para reducir la temperatura del agua de 30 °C a 10 °C sería:

$$t = \frac{1\,674\,400 \text{ J}}{144 \text{ J/s}} \approx 11\,625 \text{ s} \approx 3 \text{ h } 13 \text{ min.}$$

No obstante, en la práctica, debido a pérdidas adicionales por conducción, eficiencia variable de las celdas y disipación incompleta del calor en la cara caliente, el tiempo real estimado de enfriamiento se sitúa entre **3,5 y 4 horas**. Con esta configuración, el sistema logra no solo enfriar el agua desde 30 °C hasta los 10 °C, sino también mantenerla estable a esa temperatura, incluso con una temperatura ambiente de 30 °C, gracias al aislamiento térmico del tanque.

Para el control térmico, el sistema utiliza sensores de temperatura **DS18B20** sumergidos en el tanque de agua fría y otro sensor en la parte interna del tanque de agua caliente para compensar el control térmico global del circuito hidráulico. Los sensores DS18B20 permiten una precisión de ±0,5 °C, facilitando una regulación exacta de la temperatura. La lectura de los sensores y el control de activación de las celdas Peltier se realiza mediante una **Raspberry Pi**, que además activa los relés asociados a cada grupo de celdas según las condiciones térmicas programadas.

Capítulo 4

Circuitos

4.1. Resumen

Para el correcto funcionamiento del proyecto fue necesario implementar distintos circuitos electrónicos encargados de controlar los diversos sistemas que lo componen. El centro de control principal está basado en una Raspberry Pi Zero 2 W, la cual coordina y administra el comportamiento de todos los componentes del sistema.

Esta placa actúa como el cerebro del proyecto, gestionando la activación de las bombas de agua, el control de las válvulas y el funcionamiento de los módulos de refrigeración y calefacción del colchón. Su tarea es asegurar que cada elemento se encienda o apague en el momento que se necesite, de acuerdo con las necesidades de temperatura determinadas por el algoritmo de control.

Además, la Raspberry Pi se comunica de forma inalámbrica mediante Wi-Fi con la pulsera inteligente del usuario, desde la cual recibe datos en tiempo real sobre parámetros relevantes. Esta información es utilizada por el sistema para ajustar dinámicamente la temperatura del colchón, logrando así un mayor confort y eficiencia energética durante el descanso.

4.1.1. Pulsera

La pulsera cuenta con un módulo ESP32 Mini que se encarga de transmitir de manera inalámbrica la información hacia una Raspberry Pi utilizando conectividad Wi-Fi, estableciendo así una comunicación eficiente y continua entre ambos dispositivos. A la ESP32 Mini se encuentran conectados tres sensores mediante el bus de comunicación I²C, cada uno con una función específica dentro del sistema: un

sensor de temperatura AHT10, encargado de registrar las variaciones térmicas del entorno y del cuerpo del usuario; un acelerómetro MMA845Q , utilizado para detectar y analizar los movimientos y la posición corporal; y un sensor de ritmo cardíaco Max30102, que permite medir la frecuencia cardíaca en tiempo real. Toda la información obtenida por estos sensores es procesada por la ESP32 Mini. Una vez procesados, los resultados son enviados a la Raspberry Pi, que se encarga de realizar el cálculo del estado de sueño del usuario y determinar la temperatura óptima requerida para mantener el confort durante el descanso, utilizando algoritmos de análisis de datos. garantizando un funcionamiento inteligente, autónomo y adaptable a las condiciones individuales de cada usuario.

4.1.2. Bombas de agua

El circuito de bombeo está formado por dos módulos idénticos que permiten la conmutación y alimentación de las bombas mediante control digital desde la Raspberry Pi, cada módulo integra un relé electromecánico modelo SRD-12VDC-SL-C accionado a 12 V para aislar la carga de la lógica de control, un transistor BC337 empleado como interruptor de baja impedancia para manejar la bobina del relé desde un pin GPIO de la Raspberry Pi y los componentes de protección y polarización necesarios para un funcionamiento seguro y fiable, entre estos se consideran la resistencia de base para limitar la corriente hacia la base del BC337, un diodo 1N4007 conectado en paralelo con la bobina del relé para absorber picos inductivos y proteger el transistor.

Capítulo 5

Código

5.1. Resumen

Una de las partes más vitales del proyecto es el desarrollo del código, ya que representa el núcleo lógico que permite la comunicación, el procesamiento de datos y la toma de decisiones dentro del sistema. La arquitectura del software se divide en cuatro módulos principales, cada uno con una función específica y complementaria que, en conjunto, garantizan el correcto funcionamiento de la cama inteligente.

El primer módulo es el Presence Detector, encargado de analizar en tiempo real si hay una persona sobre la cama. Para ello, realiza una fusión de datos provenientes de distintos sensores —como temperatura, movimiento y ritmo cardíaco— con el fin de estimar de manera precisa la presencia o ausencia del usuario. Este componente es fundamental, ya que su salida condiciona el funcionamiento de los demás módulos.

El segundo módulo, denominado `analyzer.py`, es responsable de ejecutar todos los cálculos del algoritmo principal. A partir de la información recibida del Presence Detector y de los sensores, este componente determina las acciones a tomar para mantener la temperatura del colchón dentro del rango de confort del usuario. En función de las condiciones detectadas, el módulo decide si se debe activar el sistema de enfriamiento o el de calentamiento del agua, buscando siempre optimizar la experiencia de descanso.

El tercer módulo es el Smart Bed Controller, el cual actúa como intermediario entre el software lógico y el hardware físico. Su función es enviar señales de control —en forma de valores binarios (1 o 0)— hacia los tanques de agua fría y caliente y a sus válvulas eléctricas, activando o desactivando los relés correspondientes, también apaga las bombas a cada cierto tiempo para que no se quemen. Gracias a este

controlador, las decisiones tomadas por el algoritmo pueden traducirse en acciones concretas sobre el sistema hidráulico.

Por último, se encuentra el módulo correspondiente a la pulsera inteligente, que cumple un rol clave en la adquisición de datos fisiológicos del usuario. Este dispositivo portátil recopila información proveniente de sus sensores (como frecuencia cardíaca, temperatura y movimiento) y la transmite a la Raspberry Pi mediante el protocolo MQTT, utilizando una conexión Wi-Fi local. De esta forma, la cama inteligente dispone de datos actualizados y precisos para alimentar el algoritmo de control térmico.

En conjunto, estos cuatro módulos conforman un sistema integral capaz de detectar la presencia del usuario, analizar su estado, tomar decisiones autónomas y ejecutar las acciones necesarias para ajustar la temperatura del colchón. Esta estructura modular permite un desarrollo escalable, fácil de mantener y adaptable a futuras mejoras del proyecto.

5.2. Explicación Técnica Detallada - Presence Detector

5.2.1. Arquitectura General del Sistema

El módulo `presence_detector.py` implementa un sistema de detección de presencia en cama inteligente mediante fusión multi-sensor, utilizando cinco indicadores independientes que se combinan para generar un nivel de confianza global.

5.2.2. Clase Principal: BedPresenceDetector

Atributos de Estado

Estado de presencia actual:

- `bed_occupied (bool)`: Indica si la cama está ocupada actualmente.
- `presence_confidence (float)`: Nivel de confianza de presencia en porcentaje (0-100).
- `presence_history (list)`: Buffer circular que almacena los últimos N valores de confianza.

- **presence_start_time** (float): Timestamp Unix del momento en que se detectó entrada a la cama.

Valores de referencia (baseline):

- **baseline_temperature** (float): Temperatura de referencia de la cama vacía, actualizable dinámicamente.
- **baseline_activity** (float): Nivel de actividad de referencia (actualmente no utilizado).

Parámetros configurables:

- **confidence_threshold_enter**: Umbral para detectar entrada (60 % por defecto).
- **confidence_threshold_exit**: Umbral para detectar salida (20 % por defecto).
- **thermal_threshold**: Incremento térmico sobre baseline (1.5°C por defecto).
- **activity_threshold**: Nivel mínimo de actividad (0.001 por defecto).
- **hr_min** y **hr_max**: Rango válido de frecuencia cardíaca (40–150 BPM).
- **history_size**: Tamaño del buffer de historial (30 muestras).
- **confirmation_time**: Muestras consecutivas necesarias para confirmar salida (15 muestras).

Todos los parámetros se cargan desde el módulo `bed_config`, permitiendo ajuste sin modificar el código.

5.2.3. Método Principal: `detect_presence()`

Flujo de Procesamiento

El método recibe un diccionario `sensor_data` con:

- **bed_temperature**: Lectura del sensor HTU21D.
- **activity**: Magnitud del vector del acelerómetro MMA8451.
- **heart_rate**: Frecuencia cardíaca calculada por el MAX30102.
- **hr_valid**: Booleano que indica si la medición de HR es confiable.
- **finger_present**: Booleano indicando contacto físico con el sensor óptico.

Sistema de Puntuación Multi-Indicador

1. Indicador Térmico (máximo 40 puntos)

```
temp_elevation = bed_temp - baseline_temperature
```

Criterios:

- Si `temp_elevation > thermal_threshold`: +30 puntos.
- Si `temp_elevation > thermal_threshold * 2`: +10 puntos adicionales.

Fundamento: La presencia humana eleva la temperatura de la cama entre 1.5°C y 4°C.

2. Indicador de Movimiento (máximo 25 puntos)

```
activity_score = min(25, int(activity * 25 / 0.1))
```

Fundamento: Los micro-movimientos durante el sueño generan actividad detectable.

3. Indicador Cardiovascular (máximo 40 puntos) Criterios:

- Si HR válida y en rango 40–150 BPM: +35 puntos.
- Si HR en rango de sueño (50–80 BPM): +5 puntos adicionales.

Fundamento: La frecuencia cardíaca válida confirma presencia humana con alta certeza.

4. Indicador de Contacto (máximo 20 puntos) Criterios:

- Si `finger_present == True`: +20 puntos.

Fundamento: Detecta contacto físico mediante señal infrarroja reflejada.

5. Indicador Temporal (máximo 10 puntos)

```
avg_confidence = sum(presence_history[-5:]) / 5
```

Criterio: Si el promedio de las últimas 5 muestras >50 %, +10 puntos.

Fundamento: La presencia real genera señales consistentes en el tiempo.

Cálculo de Confianza Global

```
confidence = min(sum(all_indicators), 100)
```

La suma ponderada de indicadores se limita a 100 %, siendo tolerante a fallos parciales.

5.2.4. Mecanismo de Histéresis

Propósito

Evita transiciones rápidas entre estados de presencia.

Implementación en `_update_presence_state()`

Entrada (vacía → ocupada):

- Condición: `confidence >= threshold_enter`
- Acción inmediata: activa ocupación y registra timestamp

Salida (ocupada → vacía):

- Condición: `confidence <= threshold_exit`
- Confirmación: últimos `confirmation_time` valores $\leq 30\%$
- Acción: cambia a vacía tras baja confianza sostenida

Nota: La asimetría evita falsas salidas durante quietud prolongada.

5.2.5. Gestión del Baseline Térmico

Inicialización

El baseline se toma automáticamente con la primera lectura térmica.

Actualización Dinámica

Cuando la cama está vacía:

$$\text{baseline} = (1 - \alpha) \times \text{baseline}_{\text{old}} + \alpha \times \text{temp}_{\text{current}}$$

con $\alpha = 0,05$.

Calibración Manual

- Toma N lecturas en 5 minutos.
- Calcula la mediana para eliminar outliers.
- Reemplaza el baseline actual.

5.2.6. Métodos de Utilidad y Debugging

- `get_presence_summary()`: Retorna estado global.
- `get_detailed_indicators()`: Análisis granular de indicadores.
- `reset_presence_state()`: Limpia historial y banderas.
- `_get_time_occupied()`: Calcula minutos desde detección de entrada.

5.2.7. Consideraciones de Diseño

Fusión de Sensores: Suma ponderada de sensores heterogéneos.

Tolerancia a Fallos: El sistema sigue operativo ante fallos parciales.

Adaptabilidad Temporal: Baseline térmico adaptable a cambios ambientales.

Escalabilidad: Parámetros configurables vía `bed_config`.

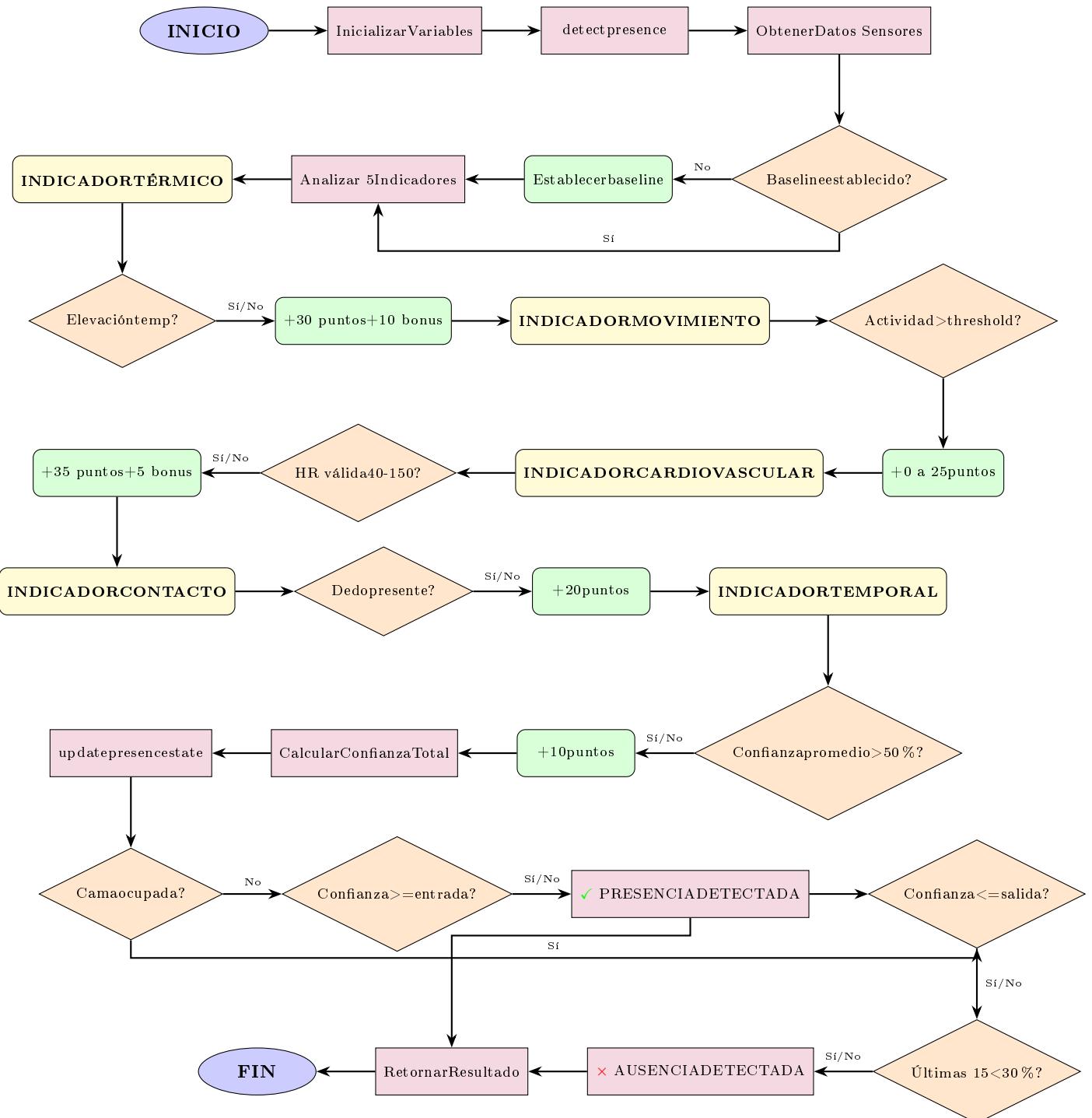
5.2.8. Retorno del Método Principal

El método `detect_presence()` retorna:

- `occupied`: estado binario actual.
- `confidence`: nivel de confianza (%).
- `indicators`: detalle de cada indicador.
- `time_occupied`: minutos desde entrada.
- `temp_elevation`: incremento térmico sobre baseline.
- `presence_changed`: bandera de transición de estado.

Este formato estructurado facilita la integración con módulos superiores del sistema de cama inteligente.

5.3. Diagrama presence_detector



5.4. Módulo de análisis de señales fisiológicas: analyzer.py

5.4.1. Arquitectura general

El módulo `analyzer.py` constituye el núcleo de procesamiento analítico del sistema, centralizando las funciones encargadas del análisis de señales fisiológicas y series temporales de estados de sueño. Este módulo se diseñó siguiendo el principio de separación de responsabilidades: no implementa clases de estado ni gestiona operaciones de entrada/salida, limitándose exclusivamente a recibir datos previamente muestreados y preprocesados para devolver métricas cuantificables y eventos de interés clínico.

La única dependencia externa es la biblioteca NumPy, empleada para cálculo vectorizado y operaciones estadísticas básicas, lo que garantiza eficiencia computacional y portabilidad del código.

5.4.2. Funciones principales

A continuación se describen las funciones principales implementadas en el módulo, detallando sus parámetros de entrada, comportamiento esperado y fundamento teórico cuando aplique.

`calculate_rmssd(ibi)` Calcula el RMSSD (*root mean square of successive differences*) a partir de una secuencia de intervalos inter-latido (IBI). Esta métrica cuantifica la variabilidad de corto plazo del ritmo cardíaco y es un indicador reconocido de la actividad del sistema nervioso parasimpático [?].

La función retorna `None` si la ventana temporal no contiene suficientes muestras para el cálculo. Matemáticamente, el RMSSD se define como:

$$\text{RMSSD} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N-1} (IBI_{i+1} - IBI_i)^2}$$

La implementación utiliza diferencias sucesivas (`np.diff`), elevación al cuadrado y cálculo de la raíz de la media aritmética.

`calculate_sdnn(ibi)` Calcula la desviación estándar (SDNN, *standard deviation of NN intervals*) de la serie completa de IBI. Esta métrica describe la variabilidad global del ritmo cardíaco en la ventana temporal considerada y se obtiene mediante `np.std`. La función retorna `None` si la ventana contiene menos de dos muestras válidas.

`calculate_stress_score(hr, rmssd, sdnn, ...)` Construye un índice heurístico de estrés en escala 0–100 mediante la integración ponderada de tres componentes fisiológicos: frecuencia cardíaca instantánea (`hr`), RMSSD y SDNN. El proceso de normalización se estructura así:

- **Frecuencia cardíaca (HR):** Se normaliza respecto a un umbral inferior fijo y un tope configurable, asumiendo una relación monotónica entre FC y nivel de estrés.
- **RMSSD y SDNN:** Se normalizan frente a valores máximos esperados. La contribución al índice de estrés se calcula como $1 - \text{valor normalizado}$, de modo que una mayor variabilidad cardíaca (asociada a mejor regulación autonómica) reduce la puntuación de estrés.

Los pesos por defecto asignados son 0.4 para HR, 0.3 para RMSSD y 0.3 para SDNN. Las normalizaciones se recortan al intervalo $[0, 1]$ para evitar valores fuera de rango.

Nota: Este índice es una aproximación heurística útil para sistemas de alarma temprana, pero requiere calibración experimental con población objetivo y validación clínica antes de su uso en entornos de producción.

`calculate_sleep_quality(sleep_states, hr_values=None, activity_levels=None)`
Genera una puntuación de calidad del sueño en escala 0–100 mediante la integración de tres dimensiones fisiológicas y conductuales:

1. **Distribución de estados de sueño:** Se asume una codificación estándar (0=WAKE, 1=LIGHT_SLEEP, 2=REM_SLEEP, 3=DEEP_SLEEP) con resolución temporal de un minuto por muestra. La función compara la distribución observada contra proporciones heurísticas óptimas documentadas en la literatura del sueño (aproximadamente 20 % sueño profundo, 25 % REM, 50 % sueño ligero, <5 % vigilia) [?].
2. **Consistencia de frecuencia cardíaca:** Si se proporciona el vector `hr_values`, se calcula la variabilidad (desviación estándar) durante las fases de sueño. Una menor variabilidad se interpreta como indicador de sueño consolidado y se mapea a una puntuación favorable.
3. **Nivel de actividad motora:** Si se proporciona `activity_levels`, se penaliza la actividad media elevada durante períodos de sueño, dado que refleja fragmentación o inquietud nocturna.

La puntuación final combina estos tres componentes con pesos de 0.5, 0.25 y 0.25 respectivamente, y se recorta al intervalo $[0, 100]$. En caso de ausencia de datos de HR o actividad, la función emplea valores neutros para mantener la computabilidad con información parcial.

`analyze_sleep_transitions(sleep_states)` Cuantifica las transiciones entre estados de sueño consecutivos y retorna una tupla (`transitions, fragmentation_index`), donde el índice de fragmentación se expresa como transiciones por hora de sueño (asumiendo 1 muestra = 1 minuto). Un índice elevado sugiere sueño fragmentado o inestable. La función retorna (None, None) si la entrada contiene menos de dos muestras.

`detect_sleep_onset(sleep_states, window_size=10, threshold=0.8)` Identifica el índice temporal correspondiente al inicio del sueño mediante un algoritmo de ventana deslizante. Se considera iniciado el sueño cuando al menos el 80 % de las muestras en una ventana de tamaño configurable (por defecto 10 minutos)

corresponden a estados de sueño (valor > 0). Retorna el índice de inicio o `None` si no se satisface el criterio en toda la serie.

`detect_wake_periods(sleep_states, min_duration=5)` Detecta segmentos contiguos de vigilia (estado `WAKE = 0`) y retorna una lista de tuplas (`start_index, duration`) para aquellos períodos cuya duración supera el umbral mínimo configurable (por defecto 5 minutos). Esta información es útil para caracterizar despertares nocturnos y estimar eficiencia del sueño.

5.4.3. Entradas y salidas

El módulo sigue convenciones consistentes para el manejo de datos:

- Las funciones aceptan secuencias numéricas en formato lista Python o `numpy.ndarray`; internamente se realiza conversión implícita cuando es necesario.
- Las funciones `calculate_rmssd` y `calculate_sdnn` retornan un valor `float` o `None` en caso de datos insuficientes.
- Las funciones `calculate_stress_score` y `calculate_sleep_quality` retoran un `float` en el intervalo $[0, 100]$.
- La función `analyze_sleep_transitions` retorna una tupla (`int, float`) o (`None, None`).
- La función `detect_sleep_onset` retorna un índice entero o `None`.
- La función `detect_wake_periods` retorna una lista de tuplas (`start_index, duration`).

Importante: El módulo no implementa funcionalidades de lectura de sensores ni parsing de protocolos. Se asume que los datos de entrada están previamente alineados temporalmente, libres de valores no numéricos y han pasado por etapas de preprocessamiento básico.

5.4.4. Supuestos y consideraciones temporales

El diseño del módulo se basa en los siguientes supuestos operacionales:

- La serie `sleep_states` está uniformemente muestreada con resolución de un minuto por muestra.
- Los umbrales, pesos de ponderación y valores máximos empleados en las normalizaciones son aproximaciones heurísticas derivadas de la literatura científica disponible. Estos parámetros requieren calibración específica con datos reales del hardware implementado y características de la población objetivo.
- Las funciones no aplican filtrado de artefactos de forma automática. Señales con picos espurios o artefactos de medición en los IBI pueden introducir sesgos significativos en las métricas RMSSD y SDNN.

5.4.5. Casos límite y estrategias de validación

El módulo implementa manejo básico de casos límite mediante retorno de valores `None` o puntuaciones neutras cuando los datos son insuficientes. No obstante, no se realizan operaciones de interpolación, imputación ni recuperación de series corruptas.

Para garantizar robustez en entornos de producción, se recomienda:

- Validar las entradas antes de invocar las funciones: verificar tipo numérico (`dtype`), longitudes mínimas y eliminar outliers en secuencias de IBI mediante criterios fisiológicos (por ejemplo, rechazar $IBI < 300$ ms o > 2000 ms).
- Desarrollar un conjunto de pruebas unitarias que incluya:
 - Ventanas con datos insuficientes.
 - Episodios de sueño continuo sin interrupciones.
 - Despertares breves y frecuentes.
 - Ausencia de señales opcionales (HR, actividad).
 - Valores extremos de HR para verificar la saturación correcta en `calculate_stress_score`.

5.4.6. Recomendaciones para uso en producción

Para la transición del módulo a entornos operacionales se sugieren las siguientes mejoras:

1. **Validación explícita de entradas:** Implementar funciones auxiliares que verifiquen tipo numérico, rangos fisiológicamente plausibles y conversión forzada a `np.array` con manejo de excepciones informativo.
2. **Filtrado de artefactos:** Incorporar un filtrado ligero o truncado estadístico (por ejemplo, Tukey's fences) en las secuencias de IBI previo al cálculo de RMSSD, reduciendo así la sensibilidad a artefactos de medición.
3. **Parametrización configurable:** Externalizar umbrales y pesos a un archivo de configuración (por ejemplo, `config/bed_config.py`) para facilitar ajustes por entorno sin modificación del código fuente.
4. **Documentación ejecutable:** Incluir un script demostrativo en `examples/demo_analyzer.py` y un conjunto de pruebas con `pytest` en `tests/test_analyzer.py` que cubra tanto casos de uso típicos como escenarios límite.

5.4.7. Formato de salida y consumo de resultados

Los módulos consumidores del análisis pueden esperar los siguientes formatos de retorno:

- **Métricas escalares:** Valor float o None (para RMSSD, SDNN, índice de estrés, calidad del sueño).
- **Análisis de transiciones:** Tupla (int, float) representando conteo de transiciones e índice de fragmentación.
- **Periodos de vigilia:** Lista de tuplas (*start_index, duration*) con índices temporales y duraciones en minutos.

Estos valores son directamente integrables en pipelines de logging, sistemas de visualización en tiempo real y algoritmos de fusión multimodal implementados en módulos superiores de la arquitectura.

5.4.8. Ejemplo de uso

A continuación se presenta un fragmento de código ilustrativo del uso típico del módulo:

Listing 5.1: Ejemplo de invocación de funciones del módulo analyzer.py

```
from analyzer import (
    calculate_rmssd, calculate_sdnn, calculate_stress_score,
    calculate_sleep_quality, analyze_sleep_transitions,
    detect_sleep_onset, detect_wake_periods
)

# Datos de ejemplo: IBI en milisegundos
ibi = [800, 810, 795, 805, 790]
rmssd = calculate_rmssd(ibi)
sdnn = calculate_sdnn(ibi)
stress = calculate_stress_score(hr=72, rmssd=rmssd, sdnn=sdnn)

sleep_states = [0, 0, 1, 1, 3, 3, 2, 2, 1, 0]
onset = detect_sleep_onset(sleep_states)
wake_periods = detect_wake_periods(sleep_states)
quality = calculate_sleep_quality(sleep_states)

print(f"RMSSD: {rmssd:.2f} ms")
print(f"Stress Score: {stress:.1f}/100")
print(f"Sleep Onset: {minuto}{onset}")
print(f"Sleep Quality: {quality:.1f}/100")
```

5.4.9. Referencias sugeridas

Para fundamentar las métricas y umbrales empleados en este módulo, se recomienda consultar:

- Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology. "Heart rate variability: standards of measurement, physiological interpretation and clinical use." *Circulation*, 1996.
- Ohayon et al. "National Sleep Foundation's sleep quality recommendations." *Sleep Health*, 2017.