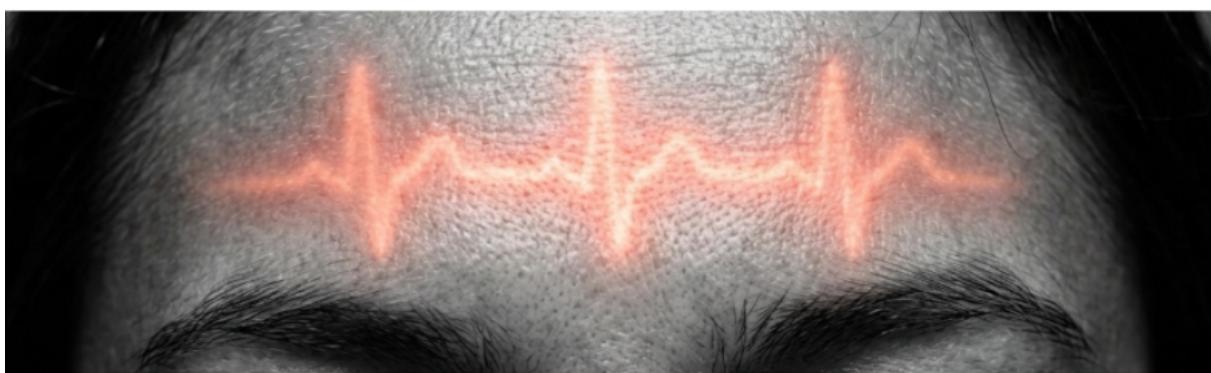
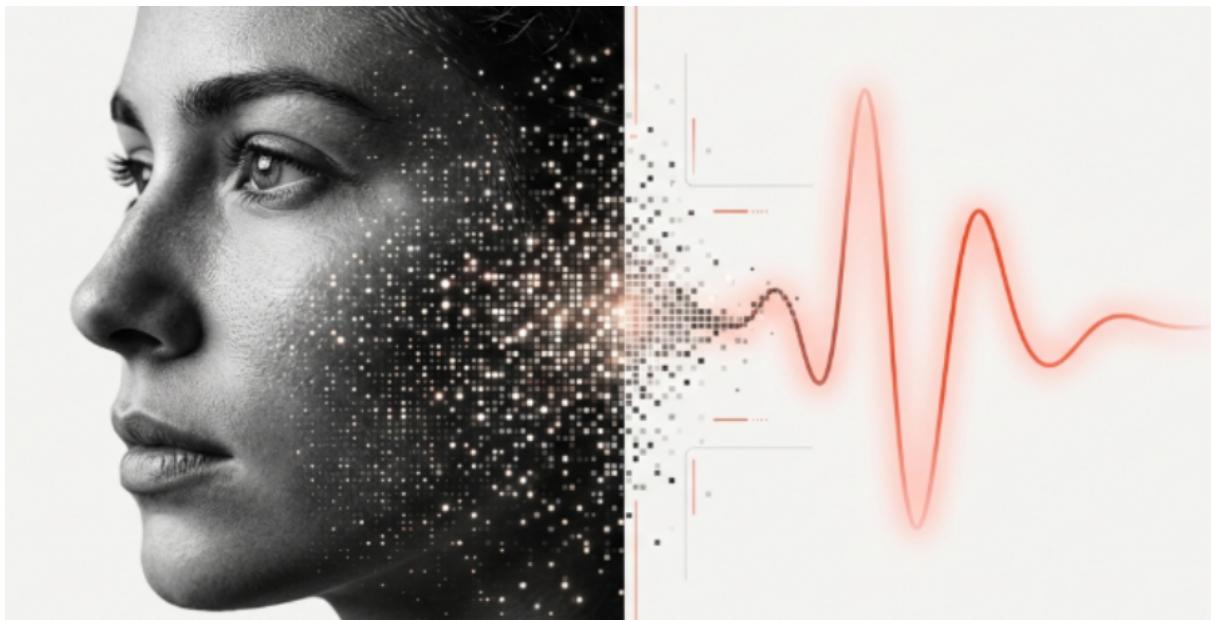


# Fotoplestimografía Remota (rPPG)

Proyecto de Visión por Computador para la Extracción de Signos Vitales en tiempo real mediante imagen



## Autoría del Proyecto:

Iván Pérez Díaz

Asia Gatta

9 de enero de 2026

# Índice

---

<b>1</b>	<b>Introducción: El Principio Físico (rPPG)</b>	<b>3</b>
<b>2</b>	<b>Configuración del Sensor (Cámara)</b>	<b>3</b>
<b>3</b>	<b>Procesamiento de Región de Interés (ROI)</b>	<b>3</b>
<b>4</b>	<b>Estimación Geométrica de Distancia</b>	<b>4</b>
<b>5</b>	<b>Algoritmo POS (<i>Plane-Orthogonal-to-Skin</i>)</b>	<b>4</b>
5.1	Normalización Temporal . . . . .	4
5.2	Proyección Ortogonal y Fusión . . . . .	5
<b>6</b>	<b>Procesamiento Digital de Señales: Frecuencia Cardíaca</b>	<b>5</b>
6.1	Interpolación . . . . .	5
6.2	Filtrado Pasa-Banda ( <i>Butterworth</i> ) . . . . .	6
6.3	Análisis Espectral (FFT) . . . . .	6
<b>7</b>	<b>Procesamiento Digital de Señales: Frecuencia Respiratoria (RPG)</b>	<b>6</b>
7.1	Filtrado Respiratorio . . . . .	6
7.2	Cálculo de RPM . . . . .	7
<b>8</b>	<b>Informe de Viabilidad y Condiciones Operativas</b>	<b>7</b>
8.1	Condiciones Ideales del Entorno . . . . .	7
8.2	Análisis de Variables Críticas . . . . .	7
8.2.1	Iluminación (Factor Crítico) . . . . .	7
8.2.2	Tono de Piel (Fototipo) . . . . .	7
8.2.3	Movimiento y Gesticulación . . . . .	8
8.2.4	Oclusión (Maquillaje, Barba y Flequillo) . . . . .	8
8.3	Casos de Fallo ( <i>Failure Cases</i> ) . . . . .	8
8.4	Resumen de Fiabilidad . . . . .	8
<b>9</b>	<b>Consideraciones Éticas y Protección de Datos</b>	<b>8</b>
9.1	Consentimiento Informado . . . . .	9
9.2	Privacidad por Diseño (Local Processing) . . . . .	9
9.3	Limitación de Responsabilidad (Descargo Médico) . . . . .	9
<b>10</b>	<b>Possibles Mejoras Futuras</b>	<b>9</b>
10.1	Implementación de <i>Deep Learning</i> (rPPG Neuronal) . . . . .	9
10.2	Nuevas Métricas: HRV y SpO2 . . . . .	10
10.3	Cancelación de Ruido Adaptativa . . . . .	10
10.4	Optimización y Portabilidad . . . . .	10
<b>11</b>	<b>Stack Tecnológico y Entorno de Desarrollo</b>	<b>10</b>
11.1	Entorno de Ejecución . . . . .	10
11.2	Librerías Principales . . . . .	11
<b>12</b>	<b>Bibliografía y Referencias</b>	<b>12</b>

12.1	Papers y Fundamentos Científicos . . . . .	12
12.2	Uso de la IA Genarativa . . . . .	12
12.3	Documentación Técnica y Librerías . . . . .	12

# 1. Introducción: El Principio Físico (rPPG)

La base del proyecto es la **Fotoplestimografía Remota (rPPG)**. Técnica que permite detectar los cambios, a simple vista invisibles, en el color de la piel causados por el flujo sanguíneo mediante el uso de la **Visión por Computador** y el **Procesamiento Digital de Señales (DSP)**.

El principio fisiológico de la **rPPG** se basa en la absorción de la luz por la **Hemoglobina** y se pueden repartir en los siguientes puntos principales:

- **Absorción:** La hemoglobina presente en la sangre absorbe la luz en el espectro visible, especialmente en las longitudes de onda verdes ( $\approx 540$  nm) y, en menor medida, en las azules ( $\approx 450\text{--}495$  nm) y rojas ( $\approx 620\text{--}750$  nm).
- **El Pulso:** Con cada contracción cardíaca, el volumen de sangre en los vasos sanguíneos de la piel varía, causando cambios periódicos en la cantidad de luz absorbida.
- **Variación Cromática:** Cuando el volumen de sangre aumenta, la absorción de luz incrementa (la piel se oscurece imperceptiblemente). El proyecto consiste en detectar estas micro-variaciones temporales en el color de la piel.

## 2. Configuración del Sensor (Cámara)

Uno de los mayores obstáculos para el rPPG es el **ajuste automático** de la cámara (*Auto-Exposure* o *Auto-Gain*). Dicho suceso consiste en que si la cámara ajusta el brillo automáticamente para compensar cambios de luz, se destruye la señal biológica, ya que el cambio de color de la piel se enmascara con el ajuste del sensor, fenómeno que puede suceder con cámaras antiguas o de baja calidad.

En el método `setup_camera`, se fuerza una exposición manual baja para **evitar la saturación** (*clipping*), motivo por el cuál al ejecutar en tiempo real se aprecia una iluminación local de entrada muy baja. Arreglando así el problema de que algún píxel alcance el valor máximo (255), perdiendo así la capacidad de registrar cambios sutiles.

```
1 def setup_camera(self):  
2     cap = cv2.VideoCapture(0)  
3     # Evitar el ajuste automático que elimina la señal del pulso  
4     cap.set(cv2.CAP_PROP_AUTO_EXPOSURE, 0.25)  
5     cap.set(cv2.CAP_PROP_EXPOSURE, -4.0)  
6     return cap
```

## 3. Procesamiento de Región de Interés (ROI)

Para extraer la señal cruda, se opta por la utilización de parches de píxeles de zonas altamente vascularizadas (frente y mejillas). Para ello, la operación matemática clave es el **promedio espacial** (Dada una región de interés con  $N$  píxeles, se calcula la señal  $S(t)$  en el instante  $t$ ).

$$S(t) = \frac{1}{N} \sum_{i=1}^N p_i(t) \quad (1)$$

Donde  $p_i(t)$  es el valor de un píxel individual.

**Justificación Matemática:** Las cámaras web (como las de algunos portátiles) presentan **ruido de disparo (ruido fotónico)** que sigue una distribución aproximadamente Gaussiana. Al promediar  $N$  píxeles, la señal del pulso (coherente en toda la región) se mantiene, mientras que el ruido aleatorio se reduce por un factor de  $\sqrt{N}$ . Esto mejora drásticamente la **Relación Señal-Ruido (SNR)**.

```

1 # Fragmento de get_roi_average
2 crop = frame[y1:y2, x1:x2]
3 if crop.size > 0:
4     # Promedio espacial para reducir ruido de disparo
5     mean = np.mean(crop, axis=(0, 1))
6     accum_color += mean

```

## 4. Estimación Geométrica de Distancia

El sistema implementa una validación de distancia utilizando el [modelo de cámara Pinhole](#) y la distancia interpupilar promedio.

$$D = \frac{W_{real} \cdot F}{W_{pixel}} \quad (2)$$

Donde:

- $D$ : Distancia del usuario a la cámara.
- $W_{real}$ : Ancho real promedio entre los ojos humanos ( $\approx 11,5$  cm).
- $F$ : Longitud focal de la cámara (en píxeles).
- $W_{pixel}$ : Distancia euclíadiana calculada entre los *landmarks* de los ojos.

```

1 # Fragmento de estimate_distance_and_check
2 # Distancia Euclidiana en pixeles
3 pixel_dist = np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
4 # Formula de proyeccion
5 distance_cm = (REAL_EYE_WIDTH * FOCAL_LENGTH) / pixel_dist

```

## 5. Algoritmo POS (*Plane-Orthogonal-to-Skin*)

El núcleo del procesamiento es el algoritmo POS (Wang et al., 2017). Este método separa matemáticamente las variaciones de color inducidas por el pulso de las inducidas por el movimiento o cambios de iluminación.

### 5.1. Normalización Temporal

Primero, se normaliza cada canal de color ( $C$ ) dividiéndolo por su media temporal ( $\mu_C$ ) para eliminar el color estático de la piel ([melanina](#)) y obtener solo las variaciones relativas tal que:

$$C_n(t) = \frac{C(t)}{\mu_C} - 1 \quad (3)$$

```

1 # Normalizacion en pos_algorithm
2 mean_color = np.mean(rgb_array, axis=0)
3 Cn = rgb_array / (mean_color + 1e-6) - 1

```

## 5.2. Proyección Ortogonal y Fusión

El algoritmo proyecta las señales RGB en un plano ortogonal al tono de piel. Se definen dos señales de proyección  $S_1$  y  $S_2$ :

$$S_1 = G - B \quad (4)$$

$$S_2 = G + B - 2R \quad (5)$$

Posteriormente, se calcula un factor de sintonización  $\alpha$  basado en las desviaciones estándar ( $\sigma$ ) para fusionar ambos componentes, cancelando el ruido espectral:

$$\alpha = \frac{\sigma(S_1)}{\sigma(S_2)} \quad (6)$$

$$P_{ppg} = S_1 + \alpha \cdot S_2 \quad (7)$$

```

1 # Proyeccion y fusion alpha
2 S1 = g - b
3 S2 = g + b - 2*r
4 alpha = np.std(S1) / (np.std(S2) + 1e-6)
5 P = S1 + alpha * S2

```

## 6. Procesamiento Digital de Señales: Frecuencia Cardíaca

Una vez extraída la señal cruda  $P_{ppg}$ , se aplica una cadena de procesamiento para obtener la frecuencia cardíaca (BPM, *Beats Per Minute*).

### 6.1. Interpolación

Las cámaras web no capturan frames a intervalos perfectos (*jitter temporal*). La FFT requiere un muestreo uniforme. Para lograrlo, se opta por utilizar la interpolación lineal para remuestrear la señal a una frecuencia objetivo ( $FS_{target} = 30$  Hz).

```

1 # Interpolacion para corregir jitter de la camara
2 f_interp = interp1d(raw_times, raw_rgb[:, i], kind='linear')
3 interpolated_rgb[:, i] = f_interp(uniform_times)

```

## 6.2. Filtrado Pasa-Banda (*Butterworth*)

Se aplica un filtro *Butterworth* para aislar el rango cardíaco generado por el usuario, eliminando ruido de alta frecuencia y movimientos lentos.

- **Corte inferior:** 0.7 Hz ( $\approx$  42 BPM).
- **Corte superior:** 3.0 Hz ( $\approx$  180 BPM).

```
1 # Filtro Butterworth de orden 3
2 b, a = signal.butter(3, [0.7/nyquist, 3.0/nyquist], btype='band')
3 filtered_ppg = signal.filtfilt(b, a, ppg_signal)
```

## 6.3. Análisis Espectral (FFT)

Se realiza una transformación de la señal al dominio de la frecuencia usando la [Transformada Rápida de Fourier \(FFT\)](#). En donde se identifica el pico de mayor energía ( $f_{peak}$ ) dentro del rango válido para calcular los BPM.

$$\text{BPM} = f_{peak} \times 60 \quad (8)$$

# 7. Procesamiento Digital de Señales: Frecuencia Respiratoria (RPG)

Adicionalmente al pulso, la señal **PPG** contiene información respiratoria conocida como **Modulación de Línea Base**. La respiración provoca cambios de presión torácica que modulan el retorno venoso a baja frecuencia.

Para detectar esto, se aumenta el *buffer* a 900 frames ( $\approx$  30 segundos) para capturar ciclos respiratorios completos.

## 7.1. Filtrado Respiratorio

Se aísla la componente respiratoria aplicando un filtro *Butterworth* centrado en frecuencias mucho más bajas que el **pulso cardíaco**:

- **Rango:** 0.1 Hz (6 RPM) a 0.5 Hz (30 RPM).

```
1 # 1. FILTRADO (Butterworth de 2 orden, 0.1-0.5 Hz)
2 low = 0.1 / nyquist
3 high = 0.5 / nyquist
4 b, a = signal.butter(2, [low, high], btype='band')
5
6 # Aplicar filtro
7 filtered_rr = signal.filtfilt(b, a, ppg_signal)
```

## 7.2. Cálculo de RPM

Al igual que con el pulso, se aplica una ventana ([Hamming](#)) y se calcula la FFT. El pico dominante en este rango de baja frecuencia corresponde a la Frecuencia Respiratoria (RPM).

$$\text{RPM} = f_{resp} \times 60 \quad (9)$$

```
1 # 3. IDENTIFICACION DE PICO (Rango 0.1 - 0.5 Hz)
2 mask = (xf >= 0.1) & (xf <= 0.5)
3 valid_yf = np.abs(yf[mask])
4 peak_idx = np.argmax(valid_yf)
5 freq_dominante = valid_xf[peak_idx]
6
7 # Conversion Hz -> Respiraciones Por Minuto
8 rpm = freq_dominante * 60.0
```

## 8. Informe de Viabilidad y Condiciones Operativas

A continuación, se detallan las consideraciones operativas, limitaciones y variables que afectan al sistema de rPPG implementado. Estas condiciones son críticas para obtener una Relación Señal-Ruido (SNR) adecuada.

### 8.1. Condiciones Ideales del Entorno

Para que el **algoritmo POS** y el **análisis espectral** funcionen con máxima precisión, el entorno debe cumplir las siguientes características:

- **Iluminación:** Luz difusa, constante y frontal. La luz natural indirecta es ideal.
- **Sujeto:** Sentado, en reposo (sin hablar ni gesticular) y mirando a la cámara.
- **Hardware:** Webcam capaz de mantener 30-60 FPS constantes y soporte para control manual de exposición.

### 8.2. Análisis de Variables Críticas

#### 8.2.1. Iluminación (Factor Crítico)

La luz es la portadora de la señal.

- **Baja Luz:** Aumenta el ruido de disparo (grano) del sensor, enmascarando el pulso.
- **Inestabilidad:** Luces fluorescentes baratas pueden introducir parpadeo (*flickering*) a 50Hz, generando armónicos falsos en la FFT.

#### 8.2.2. Tono de Piel (Fototipo)

- **Piel Clara (Tipos I-III):** Alta reflectancia, mejor SNR.
- **Piel Oscura (Tipos IV-VI):** La alta concentración de melanina absorbe más luz incidente, reduciendo la modulación de intensidad reflejada. Requiere mayor iluminación externa para funcionar correctamente.

### 8.2.3. Movimiento y Gesticulación

El movimiento es el mayor desafío del rPPG.

- **Rígido:** El algoritmo POS tolera leves inclinaciones.
- **Gesticulación:** Hablar o reír deforma la malla facial y desplaza la sangre por presión muscular, invalidando la lectura cardíaca momentáneamente.

### 8.2.4. Oclusión (Maquillaje, Barba y Flequillo)

El maquillaje denso actúa como una capa opaca que bloquea la visión de los capilares. Si la ROI (mejillas) está cubierta por barba densa, el sistema no podrá extraer la señal PPG, debiendo recurrir exclusivamente a la frente si está despejada.

## 8.3. Casos de Fallo (*Failure Cases*)

El sistema arrojará datos erróneos en las siguientes situaciones:

1. **Ajuste Automático (*Auto-Exposure*):** Si la cámara compensa los cambios de brillo automáticamente, cancela la señal del pulso.
2. **Distancia Incorrecta:**
  - Muy lejos (>80cm): Ruido domina sobre la señal por falta de píxeles.
  - Muy cerca (<30cm): Pequeños movimientos se traducen en grandes desplazamientos.
3. **Compresión de Video:** Cámaras IP o codecs agresivos eliminan los micro-cambios de color necesarios.

## 8.4. Resumen de Fiabilidad

Variable	Impacto en Precisión	Solución Implementada
Luz Variable	Alto (Destructivo)	<b>Algoritmo POS + Detrending</b>
Movimiento	Alto	Validación de Distancia + <i>Buffer</i>
Piel Oscura	Medio	Normalización temporal ( $C_n$ )
Auto-Exposición	Crítico (Fallo Total)	Control manual de driver ( <a href="#">OpenCV</a> )
Ruido Cámara	Medio	Promedio Espacial (ROI averaging)

Cuadro 1: Matriz de impacto de variables y soluciones

## 9. Consideraciones Éticas y Protección de Datos

El desarrollo de sistemas basados en visión por computador para la extracción de datos biométricos conlleva una responsabilidad ética significativa. En este proyecto se han adoptado estrictas medidas para garantizar la privacidad y los derechos de los participantes.

## 9.1. Consentimiento Informado

Todas las imágenes y grabaciones de vídeo utilizadas para el entrenamiento, validación y demostración del sistema han sido obtenidas con el **consentimiento explícito e informado** de los participantes.

- Se informó a los voluntarios sobre la naturaleza académica del proyecto.
- Se explicó que el objetivo exclusivo es la estimación de parámetros fisiológicos (rPPG) sin fines de identificación biométrica (reconocimiento facial).

## 9.2. Privacidad por Diseño (Local Processing)

El sistema ha sido diseñado bajo el principio de *Privacy by Design*.

- **Procesamiento Local:** Todo el cómputo (detección de rostro, extracción de ROI y análisis de señal) se realiza localmente en el dispositivo (*Edge Computing*).
- **No Almacenamiento:** El código no guarda ni transmite las imágenes de la cámara a servidores externos. Los frames se procesan en la memoria RAM y se descartan inmediatamente después de extraer el valor numérico del color promedio.

## 9.3. Limitación de Responsabilidad (Descargo Médico)

Es importante destacar que este software es un prototipo de ingeniería con fines de investigación.

- **No es un Dispositivo Médico:** Los resultados obtenidos (BPM, RPM) son estimaciones y no deben utilizarse para diagnóstico, tratamiento o prevención de enfermedades.
- El sistema **no cuenta** con la certificación de las autoridades sanitarias pertinentes.

# 10. Posibles Mejoras Futuras

Aunque el sistema actual implementa un algoritmo robusto (POS) y validaciones de seguridad, el campo del rPPG está en constante evolución. A continuación, se proponen las mejoras más significativas que podrían llevarse a cabo en futuras iteraciones del proyecto.

## 10.1. Implementación de *Deep Learning* (rPPG Neuronal)

El método actual (POS) es un algoritmo basado en modelos matemáticos. La tendencia actual del [estado del arte \("SOTA"\)](#) es utilizar Redes Neuronales Convolucionales (CNNs).

- **DeepPhys / PhysNet:** Implementar arquitecturas de *Deep Learning* entrenadas de extremo a extremo (*End-to-End*) que reciben los *frames* de video y devuelven la onda de pulso directamente, aprendiendo a ignorar mejor los cambios de iluminación no lineales que el algoritmo POS no puede filtrar.
- **Ventaja:** Mayor robustez ante movimientos de cabeza y condiciones de luz extremas.

## 10.2. Nuevas Métricas: HRV y SpO<sub>2</sub>

Actualmente, el sistema mide la frecuencia cardíaca (BPM) y respiratoria (RPM). Se podría expandir para calcular:

- **Variabilidad de la Frecuencia Cardíaca (HRV):** Mide la variación de tiempo entre latidos consecutivos (intervalos R-R). Es un indicador crucial para detectar estrés, fatiga y el estado del sistema nervioso autónomo. Requiere una interpolación temporal muy precisa (*cubic spline*) para compensar los bajos FPS de una webcam.
- **Saturación de Oxígeno (SpO<sub>2</sub>):** Aunque difícil con cámaras RGB estándar (normalmente requiere infrarrojos), existen técnicas experimentales que comparan el ratio de absorción entre el canal rojo y el azul para estimar el porcentaje de oxígeno en sangre.

## 10.3. Cancelación de Ruido Adaptativa

Para mejorar la resistencia al movimiento, se puede implementar un filtrado adaptativo en lugar de filtros estáticos (*Butterworth*).

- **Filtros LMS/RLS:** Utilizar una señal de referencia de ruido (por ejemplo, tomando el promedio de píxeles del fondo estático o de una zona de la cara sin piel) y restar matemáticamente ese ruido de la señal de la piel utilizando filtros adaptativos (*Least Mean Squares*).

## 10.4. Optimización y Portabilidad

- **Aceleración por GPU (CUDA):** Migrar el procesamiento de imágenes y la FFT a la tarjeta gráfica utilizando librerías como OpenCV-CUDA para liberar la CPU y permitir el procesamiento de imágenes en resolución 4K.
- **Aplicación Móvil:** Portar el núcleo del algoritmo a C++ (Android NDK o iOS) para ejecutar el sistema en dispositivos móviles, aprovechando las cámaras de alta calidad de los teléfonos modernos.

# 11. Stack Tecnológico y Entorno de Desarrollo

Para el desarrollo e implementación del proyecto, se ha utilizado el lenguaje de programación **Python**, apoyado en un ecosistema de computación científica y visión artificial.

## 11.1. Entorno de Ejecución

- **Python (3.10.19):** Lenguaje de alto nivel seleccionado por su extensa colección de librerías para ciencia de datos y prototipado rápido.
- **Anaconda:** Distribución de Python utilizada para la gestión de dependencias y entornos virtuales. Permite aislar las versiones de las librerías para evitar conflictos.
- **Jupyter Notebook:** Entorno de desarrollo interactivo basado en web. Se utilizó para las fases de prueba y calibración de algoritmos (visualización de gráficas FFT y depuración de filtros), antes de pasar el código a producción en scripts .py.

- **GitHub:** Plataforma utilizada para el control de versiones, permitiendo el trabajo colaborativo y el seguimiento de cambios en el código fuente.
- **Visual Studio Code:** Editor de código fuente empleado para la escritura y edición del código Python, con soporte para extensiones de linting y depuración.
- **Google Drive:** Almacenamiento en la nube utilizado para guardar datasets de video y resultados experimentales.

## 11.2. Librerías Principales

El núcleo del software se apoya en cuatro pilares fundamentales:

- **OpenCV (cv2):** Librería estándar de la industria para visión por computador. Se encarga de la captura de video desde la webcam, la conversión de espacios de color (BGR a RGB) y el renderizado de la interfaz gráfica (UI) sobre la imagen.
- **MediaPipe:** *Framework* desarrollado por Google. Del cuál se utiliza el módulo FaceMesh para generar una malla facial de 468 puntos en tiempo real. Esto permite localizar las regiones de interés (ROI) en la piel de forma robusta, incluso si el usuario inclina la cabeza.
- **NumPy:** Biblioteca fundamental para la computación científica. Se utiliza para el manejo eficiente de *arrays* multidimensionales, cálculo de promedios de píxeles y operaciones vectoriales necesarias en el algoritmo POS.
- **SciPy:** Librería de algoritmos matemáticos avanzados. Es crucial para el módulo DSP (Procesamiento Digital de Señales):
  - `scipy.signal`: Para el diseño y aplicación de filtros Butterworth y Detrending.
  - `scipy.fft`: Para ejecutar la Transformada Rápida de Fourier y obtener el espectro de frecuencias.

## 12. Bibliografía y Referencias

---

### 12.1. Papers y Fundamentos Científicos

- **Algoritmo POS:** Wang, W., den Brinker, A. C., Stuijk, S., & de Haan, G. (2017). *Algorithmic Principles of Remote PPG*. IEEE Transactions on Biomedical Engineering. [Enlace al Paper \(IEEE Xplore\)](#)
- **Fisiología rPPG:** Verkruyse, W., Svaasand, L. O., & Nelson, J. S. (2008). *Remote plethysmographic imaging of skin perfusion with digital cameras*. Optics Express. [Enlace al Paper \(Optica Publishing\)](#)

### 12.2. Uso de la IA Generativa

- **Gemini:** Se ha usado Gemini para aprendizaje y documentación propia acerca de las funciones y librerías utilizadas a lo largo del proyecto, siendo de ayuda para la posterior redacción del presente documento con dicha información y con las consideraciones necesarias para cumplir el RGPD (Reglamento General de Protección de Datos). [Enlace a Gemini](#)
- **ChatGPT:** Se ha empleado ChatGPT para obtener una lista de utilidades que permite LaTex así como la correcta configuración del entorno Visual Studio Code para trabajar con dicha herramienta en la creación del presente documento. [Enlace a ChatGPT](#)
- **NotebookLM:** Se ha utilizado NotebookLM para la creación de las imágenes utilizadas para la portada. [Enlace a NotebookLM](#)

### 12.3. Documentación Técnica y Librerías

- Python Software Foundation: <https://www.python.org/>
- OpenCV Documentation: <https://docs.opencv.org/master/>
- MediaPipe Face Mesh: [https://google.github.io/mediapipe/solutions/face\\_mesh.html](https://google.github.io/mediapipe/solutions/face_mesh.html)
- SciPy Reference (Signal Processing): <https://docs.scipy.org/doc/scipy/reference/signals.html>
- Anaconda Distribution: <https://www.anaconda.com/>
- Project GitHub Repository: <https://github.com/ivanperezdiaz829/rPPG-Remote-Photoplethysmography>