DESARROLLO DEL SISTEMA DE SENSADO Y VISUALIZACIÓN DE VARIABLES PARA EL PROYECTO DE INVESTIGACIÓN “PROTOTIPO DE UN SISTEMA DE CARGA LIMPIA DE DISPOSITIVOS ELECTRÓNICOS UTILIZANDO ENERGÍAS RENOVABLES GENERADA MEDIANTE PEDALEO PARA EL BENEFICIO DE LA COMUNIDAD UNIVERSITARIA”

JHONATAN ALEXANDER SUAREZ MORENO

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGIA ELECTRÓNICA

MEDELLÍN

2025

DESARROLLO DEL SISTEMA DE SENSADO Y VISUALIZACIÓN DE VARIABLES PARA EL PROYECTO DE INVESTIGACIÓN “PROTOTIPO DE UN SISTEMA DE CARGA LIMPIA DE DISPOSITIVOS ELECTRÓNICOS UTILIZANDO ENERGÍAS RENOVABLES GENERADA MEDIANTE PEDALEO PARA EL BENEFICIO DE LA COMUNIDAD UNIVERSITARIA”

JHONATAN ALEXANDER SUAREZ MORENO

Trabajo de grado para optar al título de Tecnólogo en electrónica

Asesor Técnico
Sergio Hernando Ruíz Obando
Magister en Tecnologías Digitales Aplicadas a la Educación

Asesor Metodológico
Diego Hernando Orozco Gómez
Magister en Ingeniería – Automatización Industrial

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
TECNOLOGIA ELECTRÓNICA
MEDELLÍN
2025

Dedicatoria

Dedico este proyecto a mis tutores y profesores Diego Hernando Orozco Gómez y Sergio Hernando Ruiz Obando, quienes, con su apoyo constante, dedicación y esfuerzo hicieron posible el desarrollo de este trabajo. Gracias a sus enseñanzas y compromiso he logrado culminar con éxito esta etapa académica.

Agradecimientos

Expreso mi agradecimiento a los profesores y tutores Diego Hernando Orozco Gómez y Sergio Hernando Ruiz Obando, por su acompañamiento, esfuerzo y valiosa orientación en el desarrollo de este proyecto. Asimismo, agradezco a la Institución Universitaria Pascual Bravo por el respaldo y los recursos brindados para hacer posible la ejecución de esta investigación.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de anexos

Pág.

Resumen

DESARROLLO DEL SISTEMA DE SENSADO Y VISUALIZACIÓN DE VARIABLES PARA EL PROYECTO DE INVESTIGACIÓN “PROTOTIPO DE UN SISTEMA DE CARGA LIMPIA DE DISPOSITIVOS ELECTRÓNICOS UTILIZANDO ENERGÍAS RENOVABLES GENERADA MEDIANTE PEDALEO PARA EL BENEFICIO DE LA COMUNIDAD UNIVERSITARIA”

JHONATAN ALEXANDER SUÁREZ MORENO

El crecimiento en el consumo de dispositivos electrónicos portátiles ha generado la necesidad de buscar alternativas sostenibles para su recarga, con el fin de reducir la dependencia de fuentes de energía convencionales y mitigar el impacto ambiental. En este contexto, se identificó como problema la falta de un sistema eficiente de sensado y visualización de variables que permita monitorear los parámetros eléctricos y fisiológicos asociados al uso de un prototipo de generación de energía limpia mediante pedaleo.

La intervención consistió en el diseño e implementación de un sistema basado en un microcontrolador ESP32-WROOM-32, encargado de adquirir y procesar las señales de sensores de potencia, corriente, velocidad, ángulo de rotación y ritmo cardiaco del usuario. Estos datos fueron integrados en una interfaz de visualización en tiempo real, lo que facilita tanto el análisis del rendimiento del prototipo como la interacción del usuario con el sistema. Entre las ventajas de la implementación se destacan el bajo costo del sistema, la facilidad de integración con plataformas de visualización y la posibilidad de utilizar la información recolectada para investigaciones futuras en el ámbito de la eficiencia energética y el bienestar del usuario.

Los resultados obtenidos muestran que el sistema cumple con los objetivos planteados, permitiendo el registro y monitoreo confiable de las variables del proceso de pedaleo, así como su incorporación en el prototipo de carga limpia desarrollado en la Institución Universitaria Pascual Bravo. Este proyecto constituye un aporte significativo en el campo de la automatización y las energías renovables, orientado al beneficio de la comunidad universitaria.

Palabras claves: automatización, energías renovables, ESP32, sensado, visualización de variables

Abstract

DEVELOPMENT OF THE SENSING AND VISUALIZATION SYSTEM OF VARIABLES FOR THE RESEARCH PROJECT "PROTOTYPE OF A CLEAN CHARGING SYSTEM FOR ELECTRONIC DEVICES USING RENEWABLE ENERGY GENERATED THROUGH PEDALING FOR THE BENEFIT OF THE UNIVERSITY COMMUNITY"

JHONATAN ALEXANDER SUÁREZ MORENO

The growth in the use of portable electronic devices has generated the need to seek sustainable alternatives for recharging in order to reduce dependence on conventional energy sources and mitigate environmental impact. In this context, the problem identified was the lack of an efficient sensing and visualization system to monitor both the electrical and physiological parameters associated with the use of a clean energy generation prototype through pedaling.

The intervention consisted of designing and implementing a system based on the ESP32-WROOM-32 microcontroller, responsible for acquiring and processing the signals from sensors of power, current, speed, rotation angle, and the user's heart rate. These data were integrated into a real-time visualization interface, which facilitates both the analysis of prototype performance and user interaction with the system. The advantages of this implementation include the system's low cost, the ease of integration with visualization platforms, and the possibility of using the collected information for future research in the fields of energy efficiency and user well-being.

The results obtained show that the system meets the proposed objectives, allowing reliable recording and monitoring of the pedaling process variables, as well as its incorporation into the clean charging prototype developed at the Institución Universitaria Pascual Bravo. This project constitutes a significant contribution to the field of automation and renewable energies, oriented to the benefit of the university community.

Keywords: automation, ESP32, renewable energies, sensing, variables visualization

Glosario

Ángulo de rotación: medida del desplazamiento angular generado por el pedaleo, expresado en grados, que permite calcular la frecuencia y dinámica del movimiento.

Automatización: aplicación de tecnologías y sistemas de control para realizar procesos de manera automática, reduciendo la intervención humana y aumentando la eficiencia.

Corriente eléctrica: flujo de electrones a través de un conductor, medido en amperios (A), necesario para determinar la capacidad de generación de energía en el sistema.

Energías renovables: fuentes de energía obtenidas de procesos naturales inagotables, como la energía solar, eólica o cinética del pedaleo, que permiten reducir el impacto ambiental.

ESP32-WROOM-32: microcontrolador de alto rendimiento con conectividad Wi-Fi y Bluetooth, encargado de procesar señales y controlar los sensores del sistema de sensado.

Filtro de media móvil: es una técnica que se usa para suavizar una señal y eliminar el ruido de alta frecuencia, calculando el promedio de un número determinado de puntos consecutivos de la señal de entrada para obtener cada punto de la señal de salida.

Firmware: El firmware es un tipo de software especial que controla las funciones básicas y los circuitos electrónicos de un dispositivo, actuando como una capa de bajo nivel entre el hardware y el sistema operativo.

Interfaz gráfica: entorno visual que permite al usuario interactuar con el sistema y observar en tiempo real los datos obtenidos del proceso de pedaleo.

Microcontrolador: circuito integrado programable que ejecuta instrucciones para controlar dispositivos electrónicos, gestionar sensores y procesar información en tiempo real.

Potencia eléctrica: magnitud que relaciona la energía consumida o generada en un tiempo determinado, expresada en vatios (W), fundamental para evaluar la eficiencia del prototipo.

Ritmo cardíaco: número de pulsaciones por minuto que reflejan el estado fisiológico del usuario durante el pedaleo, obtenido mediante sensores biométricos.

Sensado: proceso de adquisición de datos mediante sensores que capturan variables físicas o biológicas para su posterior procesamiento y análisis.

Sensor de velocidad: dispositivo que mide la rapidez de giro del sistema de pedaleo, proporcionando datos necesarios para calcular la energía generada y el desempeño del usuario.

Visualización de variables: representación gráfica y en tiempo real de los datos obtenidos por los sensores, que facilita la comprensión y el análisis de la información recolectada en el sistema

Introducción

La creciente demanda de soluciones sostenibles para la generación y el aprovechamiento de energía ha impulsado el desarrollo de proyectos innovadores que integran fuentes renovables en actividades cotidianas. En este escenario, la energía mecánica generada por el ser humano a través del pedaleo surge como una alternativa viable y respetuosa con el medio ambiente para la recarga de dispositivos electrónicos portátiles. Esta práctica no solo reduce la dependencia de las fuentes convencionales de electricidad, sino que también fomenta la conciencia ambiental y la eficiencia en el uso de los recursos energéticos.

El presente proyecto de grado propone el desarrollo de un sistema de sensado y visualización de variables aplicado a un prototipo de carga limpia de dispositivos electrónicos, alimentado mediante la energía obtenida por pedaleo humano. La propuesta incluye la implementación de un microcontrolador ESP32-WROOM-32, encargado de adquirir y procesar las señales de sensores que miden potencia, corriente, velocidad, ángulo de rotación y ritmo cardíaco. Toda esta información será presentada en una interfaz de visualización en tiempo real, lo que facilitará el análisis del rendimiento energético y la interacción del usuario con el sistema.

Este proyecto, apoyado por la Institución Universitaria Pascual Bravo, busca fomentar el uso de energías renovables dentro de la comunidad universitaria y aportar en el campo de la automatización y el monitoreo de datos. Su desarrollo incluye el diseño, construcción y prueba de un sistema práctico, con el objetivo de demostrar que la energía generada por el pedaleo puede aprovecharse como una forma limpia de recarga. Además, integra sensores e interfaces de visualización que facilitan el control y la comprensión del proceso. En conjunto, representa un aporte académico que impulsa el desarrollo sostenible y fortalece la investigación en nuevas tecnologías.

1. Planteamiento del problema

1.1 Descripción

En la actualidad, las comunidades universitarias enfrentan desafíos relacionados con el consumo energético y la necesidad de acceder a sistemas de carga para dispositivos electrónicos de manera sostenible. A pesar del crecimiento en el uso de energías renovables, su integración a pequeña escala y en entornos específicos, como los campus universitarios, sigue siendo limitada. Este proyecto surge como respuesta a la necesidad de diseñar e implementar sistemas que permitan la generación y aprovechamiento de energía mediante pedaleo, haciendo uso de mediciones y visualización de variables eléctricas y fisiológicas, esenciales para el correcto desempeño del prototipo.

1.2 Formulación

¿Será posible que le sistema de sensado a desarrollar para el prototipo de carga limpia de dispositivos electrónicos provea la información real de las señales adquiridas?

2. Justificación

Este trabajo de grado busca fortalecer el desarrollo de un proyecto de investigación vigente en la Institución Universitaria Pascual Bravo, mediante la creación de un sistema electrónico que permita sensar variables como potencia, corriente, velocidad angular y ritmo cardíaco. La visualización de estos datos facilitará el análisis del desempeño del sistema de carga limpia, así como la interacción del usuario con el mismo. Además, este desarrollo representa una oportunidad para aplicar conocimientos en automatización y programación de microcontroladores, contribuyendo a la formación académica del estudiante y al avance del conocimiento institucional.

3. Objetivos

3.1 Objetivo general

Desarrollar un sistema de sensado y visualización de variables físicas, eléctricas y fisiológicas para ser integrado en un prototipo de carga limpia de dispositivos electrónicos basado en energía generada por pedaleo.

3.2 Objetivos específicos

Adecuar en el prototipo de carga limpia diversos sensores para la adquisición de señales físicas, eléctricas y fisiológicas.

Programar un microcontrolador que permita el procesamiento y la visualización de las variables adquiridas.

Evaluar el funcionamiento del sistema de sensado mediante pruebas realizadas en laboratorio y en campo al interior de la Institución Universitaria.

4. Marco teórico

La integración de energías renovables en sistemas de bajo consumo se ha convertido en una de las principales líneas de investigación en ingeniería, debido a la necesidad de reducir la dependencia de combustibles fósiles y mitigar el impacto ambiental. Una de las alternativas más exploradas consiste en la conversión de energía mecánica en energía eléctrica mediante generadores accionados por pedaleo, una solución viable en entornos educativos, deportivos y comunitarios. Este tipo de iniciativas permite no solo promover prácticas sostenibles, sino también desarrollar proyectos académicos con aplicaciones tecnológicas reales.

Para lograr una implementación efectiva, es indispensable contar con sistemas de sensado y visualización de variables. Estos permiten monitorear parámetros eléctricos como voltaje, corriente y potencia, así como indicadores fisiológicos asociados al esfuerzo humano, entre ellos la frecuencia cardíaca. La integración de estas mediciones a través de un microcontrolador como el ESP32-WROOM-32 posibilita el procesamiento en tiempo real y la presentación de los datos en interfaces visuales accesibles, facilitando tanto la interpretación como la interacción del usuario con el sistema.

4.1 Microcontrolador ESP-32

El ESP32**,** desarrollado por Espressif Systems**,** es un sistema en un chip SoC (System On a Chip o Sistema en un Chip)de bajo costo y bajo consumo energético que integra capacidades de conectividad inalámbrica, específicamente Wi-Fi y Bluetooth Classic y BLE (Bluetooth low energy o Bluetooth de Baja Energía) , lo que lo convierte en una de las plataformas más versátiles y adoptadas en el ámbito de la electrónica embebida (Espressif, Espressif, 2025). Este microcontrolador, sucesor del ESP8266, introduce mejoras sustanciales tanto en potencia de cálculo como en opciones de conectividad y disponibilidad de periféricos (Espressif, Espressif, 2025).

En cuanto a su arquitectura, el ESP32 emplea distintos núcleos de procesamiento, incluyendo el microprocesador Tensilica Xtensa LX6 (en variantes de uno o dos núcleos), Xtensa LX7 o

incluso RISC-V en modelos más recientes, operando en frecuencias de hasta 240 MHz (Espressif, Espressif, 2025). Incorporando componentes esenciales para las comunicaciones inalámbricas, posee interruptores de antena, balun RF, amplificadores de potencia, amplificadores de recepción de baja ruido, filtros y módulos de administración de energía, todo ello altamente integrado para optimizar el uso del PCB (Espressif, Espressif, 2025)

El ESP32 cuenta con una rica variedad de periféricos integrados: sensores táctiles capacitivos, sensor de efecto Hall, interfaces como SPI (Serial Peripheral Interface o Interfaz Periférica Serial), I2S (Inter-IC Sound), I2C (Circuito Inter-integrado), UART (Receptor/Transmisor Asíncrono Universal), tarjetas SD (Secure Digital o Tarjeta Digital Segura), Ethernet, ADC (Analog-to-Digital Converter o Conversor Analógico a Digital) , DAC (Digital to Analog Converter o Convertidor Digital a Analógico), PWM (Modulación por Ancho de Pulso), entre otros, lo que lo hace ideal para aplicaciones IoT (Internet de las Cosas), automatización, prototipado y entornos académicos (Espressif, 2025).

Asimismo, su elevada integración y rendimiento de bajo costo han acelerado su adopción tanto en la industria como en la comunidad maker. Espressif apoya esta adopción con herramientas robustas como el SDK ESP-IDF, el soporte para Arduino IDE, y un ecosistema amplio de bibliotecas, además de una comunidad activa. Esto lo hace especialmente atractivo para aplicaciones en tiempo real que requieren eficiencia, conectividad y flexibilidad a bajo costo (Michael Howard, 2023). En la tabla 1 se observan las especificaciones técnicas del ESP-32.

Tabla 1
Especificaciones técnicas el ESP-32

Nombre del parámetro	**Valor del parámetro**
Microprocesador	LX6 de 32 Bits de 1 o 2 nucleos Tensilica Xtensa
Voltaje de operación	3,3 V
Corriente CC en el pin de operación	50mA
Corriente CC en los pines E/S	40mA
Frecuencia máxima	240MHz
Osciladores de frecuencia	8 MHz(oscilador interno), Oscilador RC interno, Oscilador de cristal externo de 2MHz a 60 MHz (se requieren 40 MHz para WI-FI/BT

Tabla 1
(Continuación)

Temporizadores	2 temporizadores de 64 bits, 1 temporizador RTC

Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

En la tabla 2 se observa la distribución de los pines ESP-32.

Tabla 2
Distribución de pines ESP-32

DAC	2 canales (8 bits convertidor digital a análogo)
ADC	18 canales (12 bits, convertidor análogo a digital)
Sensores Tactiles Capacitivos	10
LED PWM	16 canales

Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

En la tabla 3 se observan los protocolos de comunicación ESP-32

Tabla 3
Protocolos de comunicación ESP-32

WI-FI	802.11 b/g/n (velocidad de hasta 150 Mbps)
Bluetooth	Compatible con Bluetooth clásico de v4.2 BR/EDR y Bluetooth de bajo consumo
Bluetooth de bajo consumo	Soporta BLE
Protocolo UART	4 canales
Protocolo SPI	4 canales
Protocolo I2C	2 canales
Protocolo I2S	2 canales (para audio digital)
Protocolo CAN	1 canales

Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

En la tabla 4 se observa la memoria incorporada ESP-32.

Tabla 4
Memoria incorporada ESP-32

Nombre del parámetro	**Valor del parámetro**
Memoria SRAM	52 kb
ROM (Memoria flash)	448 kb
SRAM de tiempo real	16 kb

Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

4.1.1 Pines de alimentación del ESP-32. La alimentación se realiza a través de Micro-USB, pin de 3.3 V, pin de 5 V y GND. Se suministran 5 V regulados a este pin, que a su vez se regulan a 3.3 V para alimentar la placa. El pin de 3.3 V suministra directamente los 3.3 V regulados a la placa. La tierra está conectada a GND. (Wilson, 2020). Adicional tenemos el pin en la placa y se utiliza para reiniciar el microcontrolador (Wilson, 2020).

4.1.2 Pines GPIO del ESP-32. El ESP32 cuenta con 36 pines GPIO (entrada/salida de propósito general) para realizar numerosas operaciones (normalmente una a la vez) (Wilson, 2020). El ESP32 tiene un total de 18 canales ADC (12 bits cada uno) utilizados para medir el voltaje analógico dentro del rango de 0 a 3,3 V. El ESP32 está equipado con dos módulos convertidores analógico-digitales SAR, denominados ADC1 y ADC2. El ADC1 tiene 10 canales, etiquetados de ADC2_1 a ADC2_7, mientras que el ADC2 tiene 10 canales, etiquetados de ADC2_0 a ADC2_9. El valor de salida del ADC varía de 0 a 4093 con una resolución de 12 bits. (Wilson, 2020). En la figura 1 se puede observar la distribución de cada uno de los pines ADC con su respectivo GPIO (Wilson, 2020). En la Figura 1, se observa la ubicación de los pines ADC en el microcontrolador ESP-32.

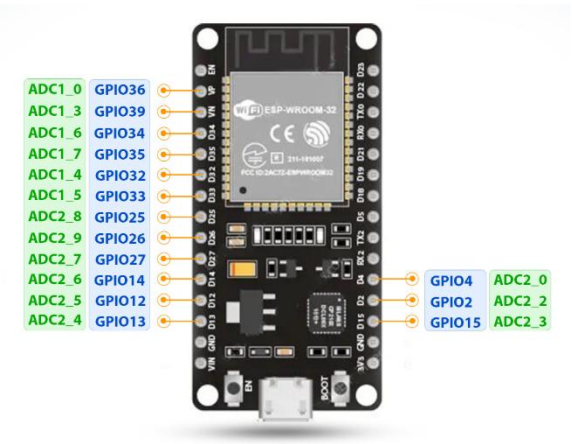

Figura 1. Pines ADC en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Pines DAC. El ESP32 cuenta con dos convertidores digital-analógicos de 8 bits (DAC1 y DAC2) para convertir valores digitales en señales analógicas (Wilson, 2020), que están conectados a los pines GPIO (DAC1-GPIO25) y (DAC2-GPIO26).El DAC utiliza una fuente de alimentación como voltaje de referencia de entrada y cuenta con una red de resistencia interna (Wilson, 2020). En la Figura 2 se observan los pines DAC1 y DAC2 con sus respectivas conexiones a los GPIO25 y GPIO26.

Figura 2. Pines DAC en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Pines PWM. El controlador PWM del ESP32 cuenta con 16 canales PWM independientes con frecuencia y ciclos de trabajo configurables. Cualquier pin GPIO puede utilizarse como pin PWM. Los pulsos PWM se utilizan para controlar la velocidad de los motores o el brillo de los LED. Se puede configurar la frecuencia, el canal, el pin GPIO y el ciclo de trabajo de la señal PWM (Wilson, 2020). En la Figura 3 se observa la distribución de los pines PWM en el ESP-32.

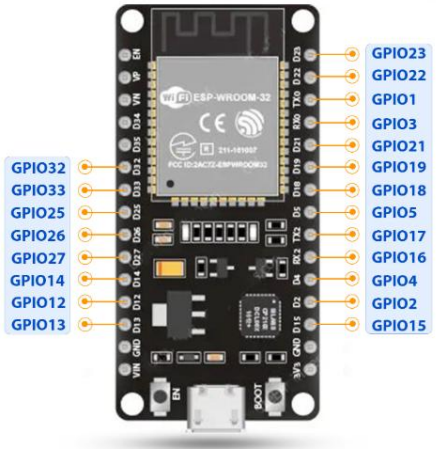

Figura 3. Pines PWM en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Pines SPI. El ESP32 tiene tres bloques SPI que operan en modo maestro y esclavo, llamados SPI, HSPI y VSPI (Wilson, 2020). Entre estos tres bloques, SPI se utiliza como interfaz para la memoria flash. Por lo tanto, nos quedan VSPI y HSPI para uso normal:

VSPI. Los pines VSPI del ESP32 son GPIO23 (MOSI), GPIO19 (MISO), GPIO18 (CLK) **y** GPIO5 (CS) utilizados para la comunicación SPI-1 (Wilson, 2020).

HSPI. Los pines HSPI del ESP32 son GPIO13 (MOSI), GPIO12 (MISO), GPIO14 (CLK) y GPIO15 (CS) utilizados para la comunicación SPI-2 (Wilson, 2020). A continuación, se observa en la figura 4 la distribución de los pines SPI en el ESP-32.

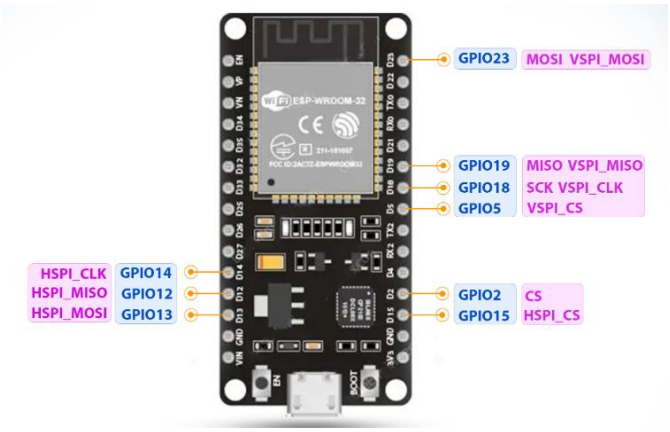

Figura 4. Pines SPI en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Pines I2C. El ESP-32 tiene dos interfaces I2C. Los pines SCL y SDA de ambas interfaces I2C pueden ser asignados por el usuario en el programa. Los pines I2C predeterminados son SDA-GPIO21 y SCL-GPIO22 (Wilson, 2020). A continuación, se observa la posición de los pines SDA y SCL.

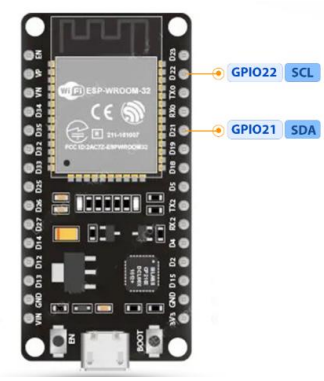

Figura 5. Pines SDA y SCL en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Sensores táctiles capacitivos**.** El ESP32 tiene 10 pines GPIO de detección táctil capacitiva (T0 a T9), que se cargan electrostáticamente cuando un dedo toca el pin GPIO respectivo. Sin hardware adicional, estos GPIO táctiles pueden utilizarse para crear paneles táctiles capacitivos. Las variaciones de capacitancia son evidentes. En la figura 6 se observa la posición de los pines GPIO táctiles en el ESP-32 (Wilson, 2020).

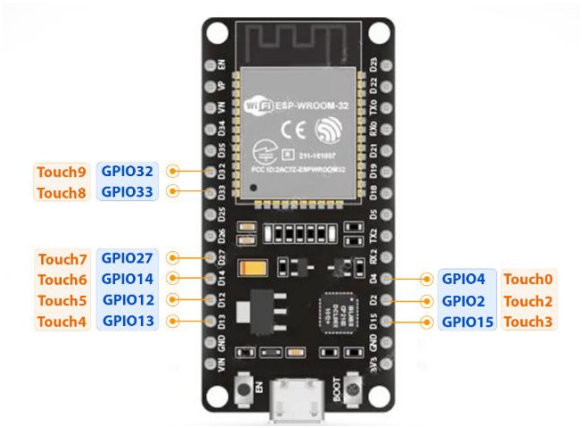

Figura 6. Pines táctiles capacitivos
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

GPIO de tiempo real**.** El ESP32 tiene 18 pines GPIO RTC de bajo consumo (RTCIO-0 a RTCIO-17) que se utilizan para despertar la placa ESP32 del modo de suspensión profunda (Wilson, 2020). A continuación, se observa en la figura 7 la posición de los pines GPIO de tiempo real en el ESP-32.

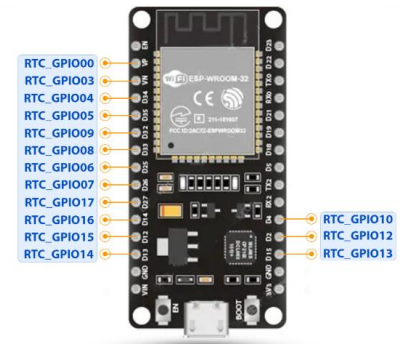

Figura 7. Pines GPIO de tiempo real.
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

Pines serie. En las placas, dos pines serie se representan como Tx y Rx . El Tx se utiliza para transmitir datos serie, mientras que el Rx se utiliza para recibirlos (Wilson, 2020). En la figura 8 se nota la ubicación de los pines serie en el ESP-32.

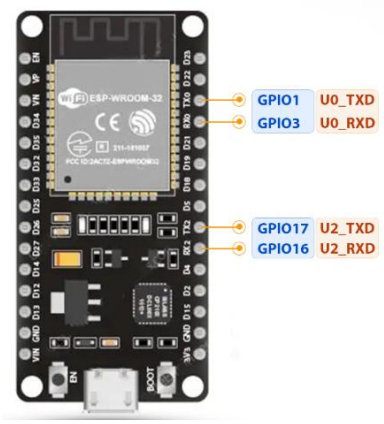

Figura 8. Pines GPIO serie en el ESP-32
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

4.2 Sensor de ritmo cardiaco MAX 30102

El MAX30102 es un módulo biosensor integrado para monitoreo de frecuencia cardíaca y oximetría de pulso (SpO_2), diseñado para aplicar en dispositivos portátiles y de salud wearable (Devices, 2025). Este sensor incorpora LEDs (uno rojo y otro infrarrojo), fotodiodos, óptica y electrónica de bajo ruido con rechazo de luz ambiente, lo que le permite medir la absorción de luz por parte de la sangre al pasar pulsos, utilizando el principio de fotopletismografía (PPG). (Devices, 2025). Opera a través de una interfaz I^2C estándar, lo que facilita su conexión con microcontroladores como el ESP32, y permite configurar la corriente de los LEDs, la frecuencia de muestreo y modos de bajo consumo para optimizar el uso energético (Devices, 2025).

En la parte posterior del módulo, hay dos LED: uno rojo y otro infrarrojo. En el lado opuesto, hay un fotodetector de alta sensibilidad. La idea es iluminar un solo LED a la vez, detectar la luz reflejada del sensor y, según la señal adquirida, medir el nivel de oxígeno en la sangre y la frecuencia cardíaca (instructables, 2025). En la figura 9, se observa la parte superior de un sensor MAX 30102, con sus respectivos nombres para cada pin.

Figura 9. Sensor de ritmo cardiaco MAX30102.
Fuente: extraído de https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

4.3 Sensor de velocidad HC-020K

Es un codificador de rueda, que mide el giro de una rueda, suele emplear un disco dividido en segmentos o ranuras. El disco codificador está montado en la parte giratoria de la rueda, ya sea en la propia rueda o en el eje del motor. A medida que la rueda o el motor giran, también gira el disco codificador (Hareendran, 2022).

El sensor óptico transmisivo del módulo del sensor de velocidad detecta la transición de un segmento/ranura a otro para representar la cantidad de vueltas que ha dado la rueda, dependiendo de cuántas ranuras dividan el disco codificador. Si son 20 ranuras, si el sensor detecta el paso de 20 ranuras, la rueda ha dado una vuelta (Hareendran, 2022).

El módulo HC-020K tiene 3 pines: uno para alimentación (VCC/5 V), otro para tierra (GND/0 V) y otro para la salida de señal (OUT). En cada transición, la salida de señal cambia de alto a bajo o de bajo a alto (H→L / L→H) (Hareendran, 2022).

Los elementos básicos de un sensor óptico transmisivo, también conocido como fotointerruptor, son un emisor (transmisor de luz) y un fotodetector (receptor de luz). Normalmente, se utiliza un diodo emisor de infrarrojos (IRED) y un fototransistor, y el emisor se coloca frente al detector. Un sensor transmisivo es adecuado para distancias cortas y objetos estrechos (Hareendran, 2022).

El sensor óptico transmisivo utilizado en el módulo HC-020K es el sensor fotoeléctrico WYC-H206. A continuación, en la figura 10 se observa el sensor óptico transmisivo WYC-H206.

Figura 10. Sensor óptico transmisivo WYC-H206.
Fuente: extraído de https://www.edn.com/motor-speed-sensor/

Un módulo HC-020K típico funciona con 5 V CC y produce una salida digital en el rango de 0 V a 5 V. Proporciona una salida ALTA/5 V cuando el haz se ve afectado y una salida BAJA/0 V cuando no hay obstrucciones. La mayoría de los microcontroladores pueden leer el tren de pulsos de salida para calcular la distancia recorrida por el motor y a qué velocidad (Hareendran, 2022).

Conociendo el diámetro de la rueda del encoder se puede hacer un cálculo para saber la velocidad y la distancia recorrida. A continuación, se observa el sensor HC-020K con sus respectivos elementos.

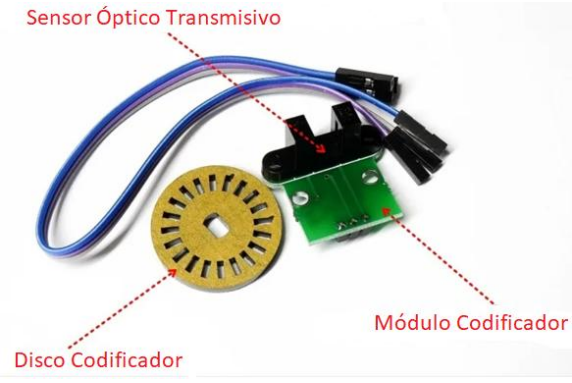

Figura 11. Sensor HC-020K
Fuente: extraído de https://www.edn.com/motor-speed-sensor/

4.4 Sensor de corriente y voltaje SEN0291

Wattmetro digital de medición a gran escala, alta precisión y alta resolución que puede medir el voltaje, la corriente y la potencia de varios módulos electrónicos y equipos eléctricos dentro de 26V8A, y el error relativo máximo es inferior a ± 0.2 % (Se requiere una calibración manual simple antes de su uso). Se puede utilizar para evaluar el consumo de energía o la duración de la batería de sistemas de energía solar, baterías, motores, controladores o módulos electrónicos (JLVE, s.f.). En la condición de medición a escala completa, el error relativo máximo de la medición de voltaje y corriente es inferior a ±0.2%. También proporciona cuatro direcciones I2C que se pueden configurar mediante el interruptorDIP2P.El módulo mide con precisión las corrientes bidireccionales del lado alto (corriente que fluye a través de la fuente de alimentación o del positivo de la batería), lo que es especialmente útil en aplicaciones de medición de combustible solar o de batería donde es necesario cargar y descargar la batería. Este estado puede determinarse simplemente mediante lecturas de corriente positivas o negativas. En las aplicaciones de motor, la corriente se puede monitorear en tiempo real monitoreando si la corriente del motor es demasiado grande debido a una sobrecarga. Además, puede utilizar este módulo para medir el consumo de energía de varios módulos electrónicos o de todo el proyecto para evaluar la duración de la batería. (JLVE, s.f.). En la figura 12, se observa el sensor de corriente y voltaje SEN0291.

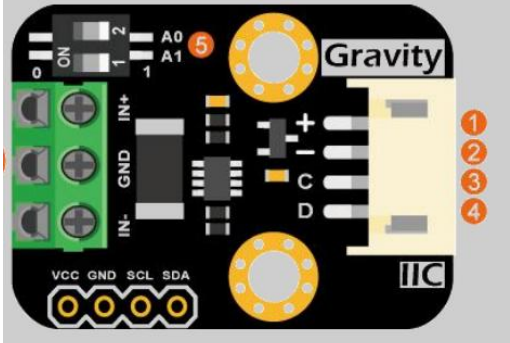

Figura 12. Sensor de corriente SEN0291
Fuente: extraído de https://agelectronica.lat/pdfs/textos/S/SEN0291.PDF

En la tabla 5, se observan las especificaciones del sensor SEN0291.

Tabla 5.
Especificaciones del sensor SEN0291.

Parámetros	**Descripción**
Voltaje de alimentación	3.3 ~ 5.5V
Rango de voltaje (IN+ o IN- relativo a GND)	0 ~ 26V
Resolución de voltaje	4mV
Error relativo de voltaje	<±0.2% (típico)
Rango de corriente	0 ~ ±8A (corriente bidireccional) 1mA
Rango de potencia	20mW (hardware) /4mW (software) 0.7mA

Fuente: extraído de https://agelectronica.lat/pdfs/textos/S/SEN0291.PDF

4.5 Pantalla Nextion NX1060P101-011C—1.

La Nextion NX1060P101-011C—1 es una pantalla HMI (Human-Machine Interface) de la serie *Intelligent* (Nextion) de 10.1 pulgadas, con pantalla IPS y panel táctil capacitivo, pensada para crear interfaces gráficas interactivas que delegan parte de la lógica de la GUI a la propia pantalla (microcontrolador interno y sistema de archivos para proyectos HMI). Esto facilita el diseño visual con el Nextion Editor y reduce la carga de procesamiento en el microcontrolador huésped. (nextion, s.f.).

4.5.1 Especificaciones principales de hardware. Tamaño de pantalla: 10.1"; tipo: IPS**.** Resolución: 1024*600. Panel táctil: capacitivo. Número de colores: 65536. Memoria flash: 128 MB. RAM: 512 KB. Reloj del microcontrolador: 200 MHz. Memoria EEPROM: 1024 bytes. Número de GPIO: 8. Reloj en tiempo real: SÍ (elty.pl, s.f.).

4.5.2 Arquitectura y funcionamiento. Arquitectura HMI «inteligente»**:** la pantalla integra una pequeña CPU, memoria y un sistema de reproducción de la interfaz diseñado en Nextion Editor. El microcontrolador externo (ej. ESP32, Arduino, Raspberry Pi) se conecta por UART para enviar/recibir comandos y eventos. Esto permite que la GUI funcione de forma autónoma en la pantalla, mientras el MCU externo se ocupa de la lógica de control y sensores (nextion, s.f.).

Protocolo y comandos**:** la comunicación se hace mediante comandos ASCII/hex definidos por el *Nextion Instruction Set* — operaciones como `page`, `get`, `set`, actualizaciones de texto, control de componentes y notificaciones de eventos táctiles se gestionan por este conjunto de instrucciones. El manual de instrucción detalla formato de comandos, respuestas y buenas prácticas (nextion, s.f.).

Editor Nextion (herramienta de desarrollo)**:** el Nextion Editor es la aplicación oficial para diseñar páginas, componentes, animaciones y compilar el proyecto (.HMI) que se carga en la pantalla (por SD o micro USB/depuración). Existe guía de uso oficial y versiones LTS para compatibilidad (nextion, s.f.).

4.5.3 Interconexión con microcontroladores. Cableado físico: conector típico 4 pines: VCC (+5V), GND, TX (salida de Nextion), RX (entrada Nextion). Es necesario un GND común entre Nextion y el MCU (sonoff, s.f.).

Niveles lógicos y compatibilidad 3.3 V / 5 V: Nextion funciona alimentado a 5 V; sin embargo, en la práctica hay amplia documentación y experiencias que indican que la entrada RX del Nextion reconoce niveles lógicos de 3.3 V (por lo que muchos usuarios conectan ESP32/RPi sin conversor), y la salida TX de Nextion suele estar en niveles compatibles con 3.3 V. Aun así, no es universalmente garantizado para todas las unidades/series, por lo que la práctica de ingeniería recomendable es verificar la hoja de datos de la unidad específica y, si se busca máxima seguridad, usar conversores de nivel bidireccionales entre MCU 3.3V y Nextion 5V (DroneBot Workshop , s.f.).

Velocidad UART (baudrate)**:** configurable desde el Editor/firmware; valores comunes 9600, 115200 bps, etc. Se debe sincronizar el MCU y la pantalla (nextion, s.f.).

Librerías y ejemplos**:** existen librerías para Arduino/ESP (Nextion library, EasyNextionLibrary) y ejemplos en GitHub y foros para integrar con ESP32 (uso de Serial1/Serial2, nexInit, NexText, etc.) (Seitanis, 2019-2020).

5. Metodología

5.1 Tipo de proyecto

Este trabajo corresponde a un proyecto de desarrollo tecnológico con enfoque investigativo, enmarcado en la línea de automatización.

5.2 Método

Se adopta un enfoque cuantitativo-experimental, que permita recolectar y analizar datos a partir de la implementación del sistema propuesto, midiendo su efectividad mediante pruebas de campo controladas.

5.3 Instrumentos de recolección de información

5.3.1 Fuentes primarias. Documentación técnica de los sensores y microcontrolador utilizados. Foros y páginas web avaladas por la comunidad científica.

5.3.2 Fuentes secundarias. Artículos científicos sobre energías renovables, sensado y automatización. Trabajos de grado de proyectos similares. Libros con temáticas especificas del proyecto a desarrollar.

6. Resultados del proyecto

6.1 Instalación y programación de los sensores para la medición de las variables

La realización del proyecto se desarrolló en una serie de pasos que permitieron cumplir con los objetivos específicos planteados.

Se realizó la integración de los sensores seleccionados en el prototipo. Cada dispositivo fue conectado al microcontrolador ESP32, teniendo en cuenta la compatibilidad de sus salidas analógicas y digitales con los pines disponibles en la tarjeta.

El resultado de este proceso se presenta en la tabla de conexiones, donde se detallan los pines del ESP32 empleados para cada sensor, así como el tipo de señal correspondiente. Este recurso permite visualizar de manera clara la asignación de hardware realizada durante la etapa de adecuación.

6.1.1 sensor de velocidad. La Tabla 6 presenta la correspondencia de pines entre el microcontrolador ESP-32 y el sensor óptico HC-020, donde se especifican las conexiones de alimentación, tierra y señal digital utilizadas para la adquisición de pulsos del disco encoder.

Tabla 6.
Pines de conexión entre el sensor HC-020 y el microcontrolador ESP-32.

Pines sensor HC-020	Pines ESP-32-WROOM-32	Descripción pines de conexión ESP-32
5V	VIN	Conexión de 5V
GND	GND	Conexión tierra
OUT	GPIO15	Conexión de señal

Fuente: Diseño propio

En la figura 13, se observa el diagrama de conexión del microcontrolador ESP32-WROOM-32 con el sensor óptico HC-020. El pin de salida digital del sensor se conecta al GPIO15 del ESP32, mientras que la alimentación se realiza a 5V y tierra a (GND).

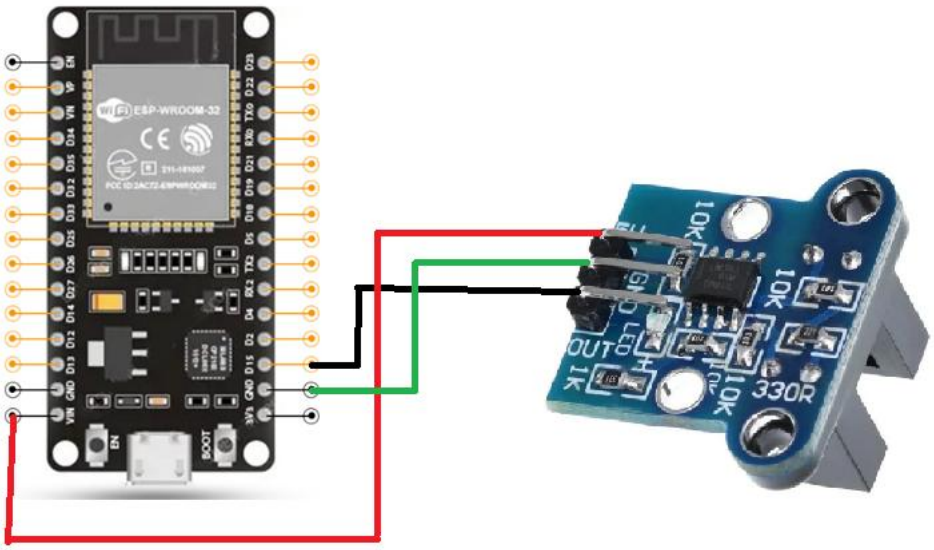

Figura 13. Diagrama de conexión entre el sensor HC-020 y el microcontrolador ESP-32
Fuente: Diseño propio

La figura 14 muestra el diagrama flujo que representa el funcionamiento de la función revoluciones_por_minuto(), la cual calcula la velocidad de rotación (RPM) y la velocidad lineal (km/h) a partir de las señales enviadas por el sensor HC-020**.**

6.1.2 sensor de ritmo cardiaco. En la tabla 7, se presentan las conexiones realizadas entre el sensor de ritmo cardiaco MAX30102 y el microcontrolador ESP32**.** El sensor se comunica mediante el protocolo I2C, utilizando los pines GPIO 21 (SDA) **y** GPIO 22 (SCL)**,** además de las líneas de alimentación 5V y GND**.**

Tabla 7.
Descripción y conexiones del sensor MAX 30102 con el microcontrolador ESP-32

Conexión pines MAX 30102	**Conexión pines ESP-32**	**Descripción pines de conexión ESP-32**
VIN	VIN	Conexión de 5V
GND	GND	Conexión tierra
SCL	GPIO 22	TOUCH7/ADC2_7
SDA	GPIO 21	DAC1/ADC2_8

Fuente: Diseño propio

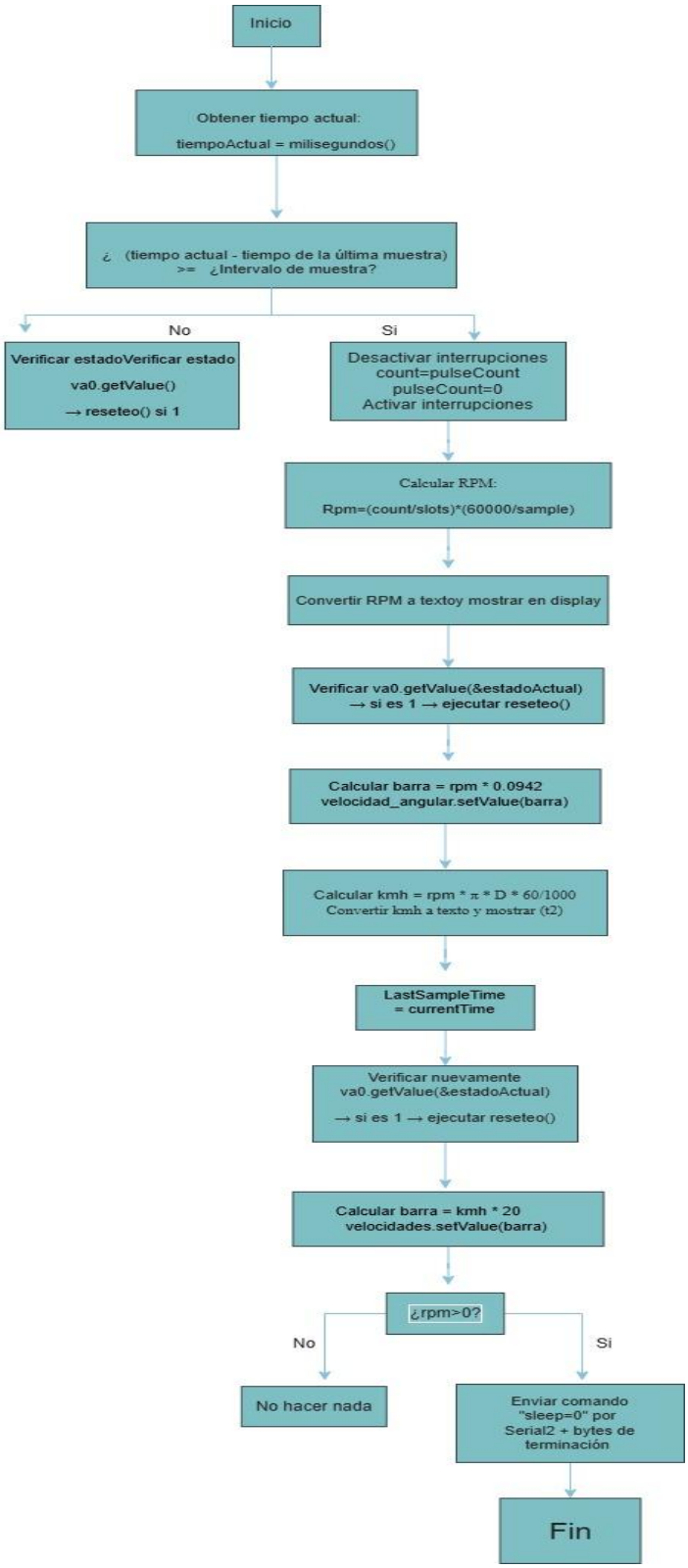

Figura 14. Diagrama de flujo sensor HC-020
Fuente: diseño propio desarrollado en Draw.io

En la figura 15, se observa el diagrama de conexión del sensor de ritmo cardiaco MAX30102 con el microcontrolador ESP32**.** El sensor se comunica mediante el protocolo I2C, utilizando el pin GPIO 21 para SDA y el pin GPIO 22 para SCL. La alimentación se realiza con 5V provenientes del ESP32, mientras que la línea GND se conecta a tierra común. Esta configuración permite la adquisición en tiempo real de la señal infrarroja, necesaria para el cálculo de la frecuencia cardíaca.

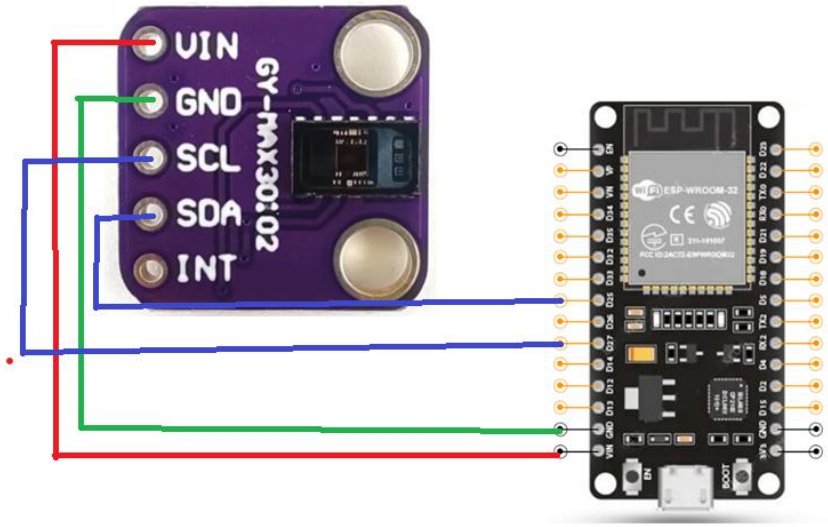

Figura 15. Diagrama de conexión entre el sensor MAX 30102 y el microcontrolador ESP-32
Fuente: Diseño propio

La figura 16 muestra el diagrama de flujo que representa el funcionamiento del programa encargado de la lectura y procesamiento de la señal del sensor MAX30102, utilizado para medir el ritmo cardíaco mediante un ESP32. En él se muestra la secuencia de inicialización del sistema, la detección del dedo, el cálculo de las pulsaciones por minuto (BPM) y la visualización de los resultados promedio en el monitor serial.

6.1.3 sensor de corriente y voltaje. En la tabla 8, se presentan las conexiones realizadas entre el sensor de corriente y voltaje SEN- 0291 y el microcontrolador ESP32**.** El sensor se comunica mediante el protocolo I2C, utilizando los pines GPIO 21 (SDA) **y** GPIO 22 (SCL)**,** además de las líneas de alimentación 5V y GND**.**

Tabla 8.
Descripción y conexión del sensor SEN-0291 con el microcontrolador ESP-32

Conexión pines SEN 0291	**Conexión pines ESP-32**	**Descripcion pines de conexión ESP-32**
Pin positivo (+)	VIN	Conexión de 5V
Pin negativo (−)	GND	Conexión tierra
Pin SCL (C)	GPIO 22	SCL
Pin SDA (D)	GPIO 21	SDA

Fuente: Diseño propio

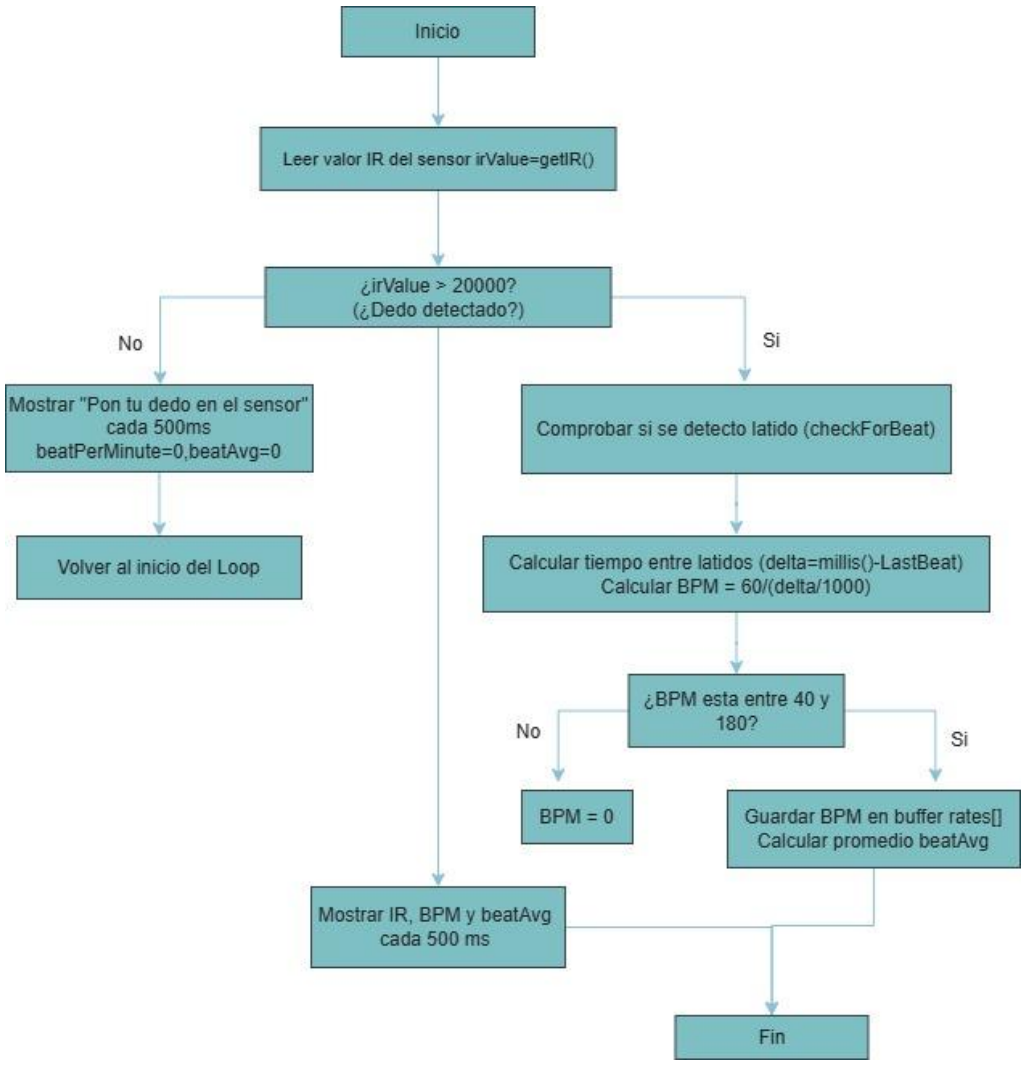

Figura 16. Diagrama de flujo de sensor de ritmo cardiaco MAX 30102
Fuente: diseño propio desarrollado en Draw.io

En la figura 17, se observa el diagrama de conexión del microcontrolador ESP32-WROOM-32 con el sensor SEN 0291. El sensor se comunica mediante el protocolo I2C, utilizando los pines GPIO 21 (SDA) **y** GPIO 22 (SCL)**,** además de las líneas de alimentación 5V y GND**.**

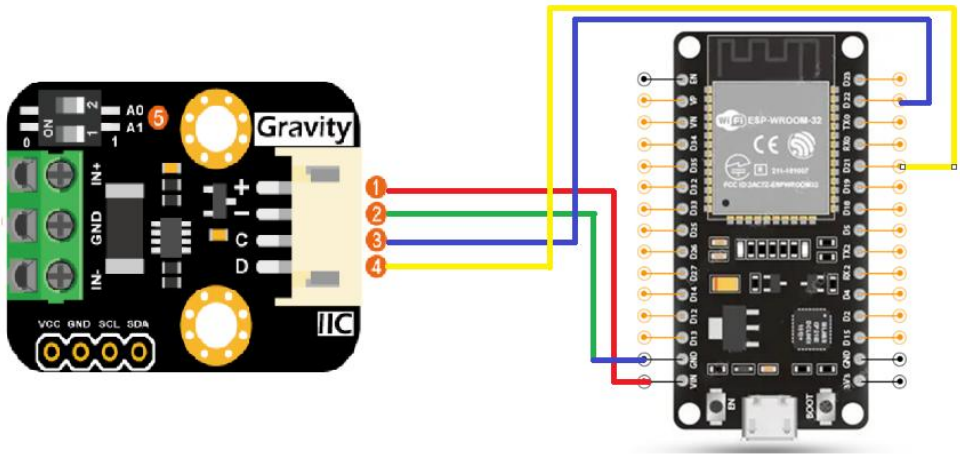

Figura 17. Diagrama de conexión entre el sensor SEN 0291 y el microcontrolador ESP-32
Fuente: Diseño propio

En la figura 18 se muestra el diagrama de flujo que representa el funcionamiento de la función corriente_dc(), la cual se encarga de medir el voltaje de salida mediante el sensor INA 0219, mostrar el valor en la pantalla Nextion.

En la figura 19 se muestra el diagrama de flujo que representa el funcionamiento de la función voltaje_dc(), la cual se encarga de medir el voltaje de salida mediante el sensor INA 0219, mostrar el valor en la pantalla Nextion.

6.2 Evaluación el funcionamiento del sistema de sensado

Cada sensor fue programado de forma independiente con el fin de validar su correcto funcionamiento antes de integrarlos todos en el sistema. En este proceso, los diagramas de flujo por sensor jugaron un papel clave, ya que permitieron esquematizar la lógica de adquisición y procesamiento de las señales. Asimismo, se incluyó la visualización en tiempo real de las variables adquiridas mediante la comunicación serial.

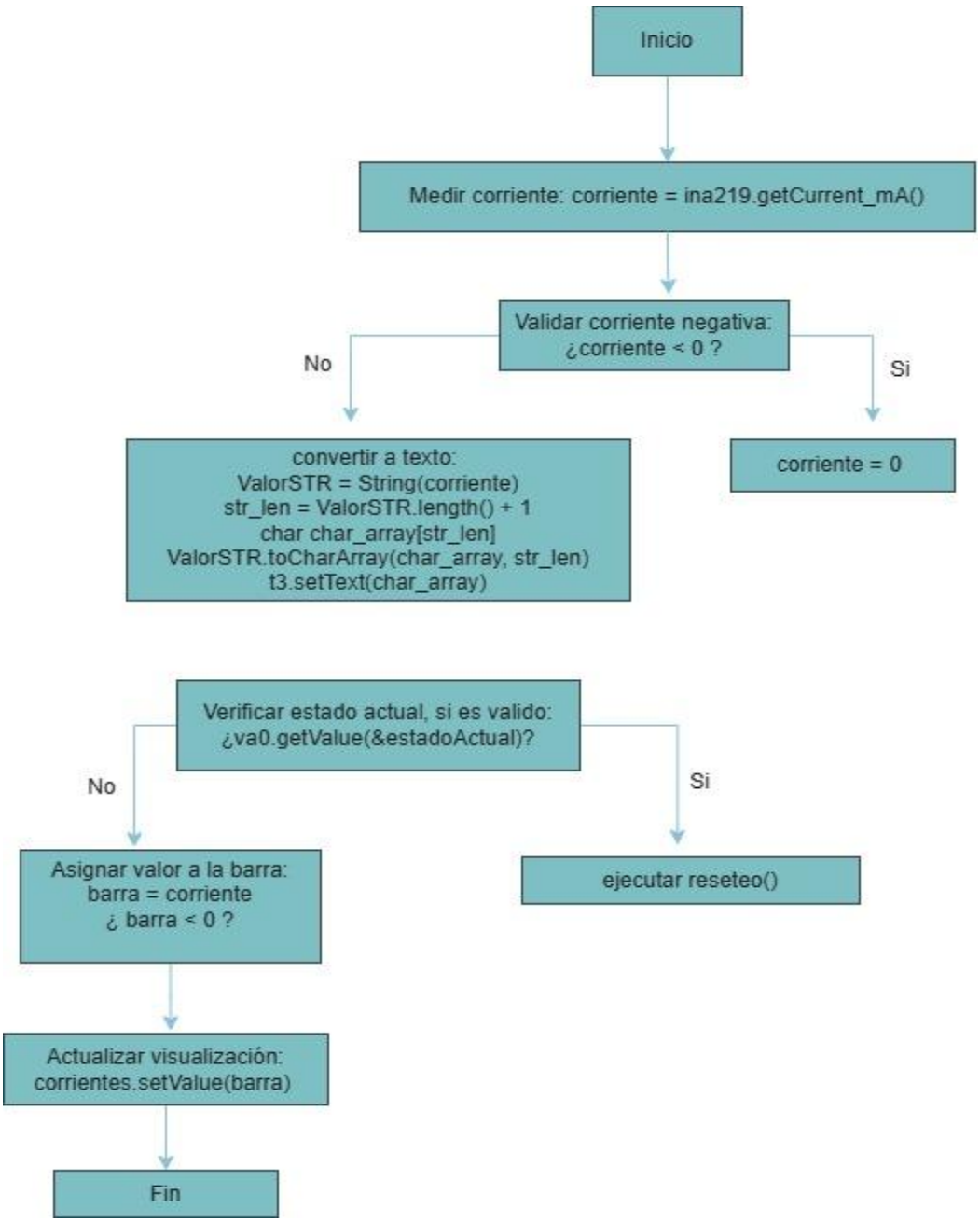

Figura 18. Diagrama de flujo de corriente, de sensor SEN 0291.
Fuente: diseño propio desarrollado en Draw.io

Finalmente, se llevaron a cabo pruebas para verificar el desempeño del sistema en dos fases: laboratorio y campo. En la primera fase se comprobó el comportamiento individual de los sensores y la estabilidad de las lecturas en condiciones controladas. En la segunda fase se evaluó el sistema integrado en condiciones reales de uso, simulando la interacción de los estudiantes con el prototipo de carga limpia.

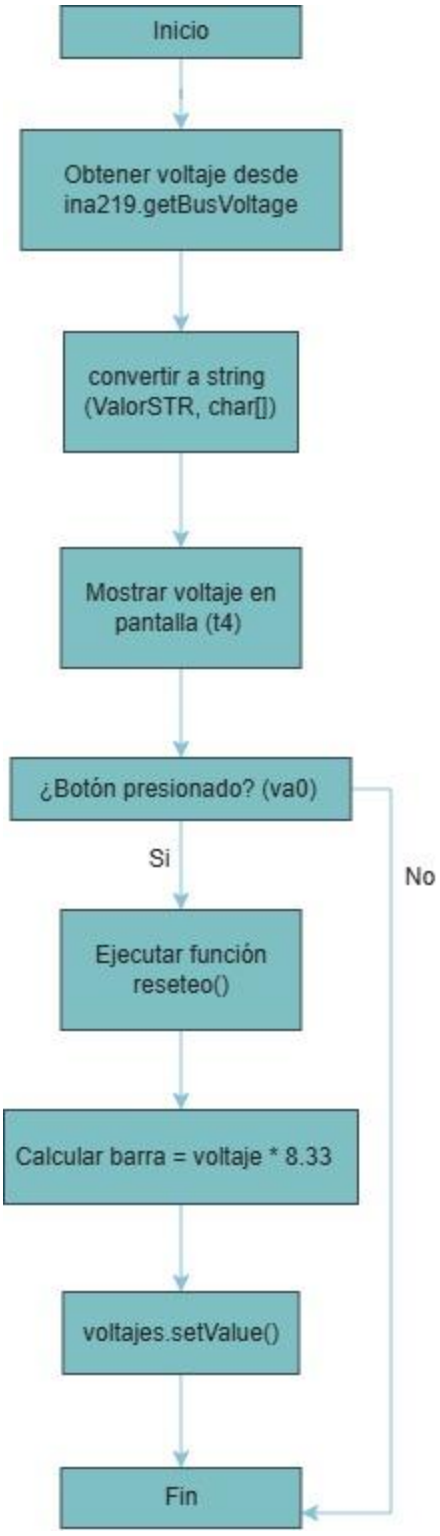

Figura 19. Diagrama de flujo de voltaje de sensor SEN 0291.
Fuente: diseño propio desarrollado en Draw.io

Los resultados evidenciaron que los sensores lograron captar de manera satisfactoria las señales físicas, eléctricas y fisiológicas. En el caso del sensor de ritmo cardíaco se observaron ligeras fluctuaciones debidas al movimiento del usuario y ruidos externos en el ambiente, pero en general se pudieron visualizar resultados muy buenos, mientras que los sensores eléctricos y mecánicos presentaron lecturas estables y consistentes.

En conjunto, la programación del ESP32 y la correcta integración de los sensores demostraron que el sistema de sensado es confiable para su aplicación en el contexto del proyecto.

7. Conclusiones

El desarrollo de este proyecto permitió comprobar que la energía generada por el pedaleo puede aprovecharse de manera práctica para la carga de dispositivos electrónicos, aportando una alternativa limpia y sostenible. La implementación del sistema de sensado y visualización de variables demostró que, con recursos accesibles y una buena planificación, es posible integrar tecnologías de automatización que midan y presenten datos en tiempo real de forma confiable.

El uso del microcontrolador ESP32-WROOM-32 junto con los sensores eléctricos, mecánicos y fisiológicos permitió registrar información precisa sobre el funcionamiento del sistema. Esta integración fue clave para entender cómo se comportan las diferentes variables durante el pedaleo y, a la vez, optimizar la eficiencia de la energía generada. Además, la interfaz de visualización diseñada facilitó el análisis de la información y mejoró la experiencia del usuario, haciéndola más intuitiva y cercana.

A lo largo del proceso se destacó la importancia de aplicar conocimientos teóricos en un entorno práctico. Cada etapa del proyecto fue una oportunidad para aprender, corregir errores y fortalecer habilidades tanto técnicas como analíticas. El apoyo de los docentes y el trabajo constante fueron fundamentales para alcanzar los resultados propuestos y para lograr un producto que realmente funcione.

En el contexto institucional, este proyecto representa un aporte a la Institución Universitaria Pascual Bravo, ya que promueve el uso responsable de la energía y la conciencia ambiental dentro de la comunidad académica. A nivel local, demuestra que la innovación no depende de grandes recursos, sino de la creatividad y el compromiso. A nivel nacional, este tipo de iniciativas aportan al desarrollo de tecnologías limpias y a la formación de profesionales más conscientes con el medio ambiente.

Finalmente, se puede concluir que el pedaleo como fuente de energía es una alternativa viable y con gran potencial para seguir explorando. Este proyecto deja abierta la posibilidad de continuar mejorando el sistema, incorporando nuevas tecnologías y extendiendo su uso a otros

espacios. Más allá de los resultados técnicos, el trabajo deja una enseñanza sobre el valor del esfuerzo, la perseverancia y la responsabilidad con el entorno.

8. Recomendaciones

Durante el desarrollo de este proyecto surgieron varios aprendizajes y también algunos aspectos que podrían mejorarse si en el futuro se quiere seguir ampliando o fortaleciendo esta propuesta.

En primer lugar, sería conveniente realizar una calibración más precisa de los sensores, especialmente el que registra las pulsaciones por minuto del corazón. Aunque el sistema cumplió con su función, en algunas pruebas se notaron pequeñas variaciones que podrían corregirse con un proceso de ajuste más detallado o con el uso de sensores de mayor precisión.

En cuanto al sistema de visualización, sería interesante incorporar nuevas funciones que permitan almacenar y comparar datos a lo largo del tiempo. Por ejemplo, una interfaz que muestre gráficas históricas o que pueda conectarse a una base de datos en la nube abriría la posibilidad de hacer análisis más profundos del rendimiento del sistema.

De igual forma, se sugiere seguir explorando la integración de otras fuentes de energía renovable, como la solar o la eólica, para complementar la energía generada por pedaleo. Esto permitiría ampliar el alcance del proyecto y convertirlo en una estación de carga híbrida totalmente sostenible.

Finalmente, es importante resaltar la necesidad de seguir apoyando este tipo de iniciativas dentro de la Institución Universitaria Pascual Bravo. Este proyecto demostró que, con dedicación, creatividad y el acompañamiento de los docentes, es posible convertir una idea en una propuesta funcional que combina aprendizaje, tecnología y compromiso ambiental. Continuar impulsando estos trabajos fortalecerá la formación de los estudiantes y fomentará una cultura más consciente y responsable con el uso de la energía.

9. Referencias bibliográficas

EDS Robotics. (s.f.). *Tipos de sensores más usados: características y funciones. Obtenido de* https://www.edsrobotics.com/blog/tipos-sensores-mas-usados/

Elty.pl. (s.f.). *Pantalla táctil capacitiva Nextion Intelligent de* 10,1" y 1024x600 NX1060P101-011C-I. Recuperado el 3 de Noviembre de 2025, Elty.pl elektronica https://elty.pl/en_US/p/Display-Nextion-Intelligent-10.1-1024x600-NX1060P101-011C-I-capacitive-touch-panel/2473

Espressif Systems. (2023). ESP32-WROOM-32 *Datasheet. Obtenido de* https://www.espressif.com/sites/default/files/documentation/esp32 wroom-32_datasheet_en.pdf

How2Electronics.com. (s.f.). *How to use INA226 DC Current Sensor with Arduino*. Recuperado de https://how2electronics.com/how-to-use-ina226-dc-current-sensor-with-arduino/

Luis Llamas. (s.f.). *Cómo usar las entradas analógicas ADC en un ESP32*. Recuperado el 28 de agosto de 2025, de https://www.luisllamas.es/esp32-adc/

Microcontrollers Lab. (s.f.). *MAX30102 pulse oximeter and heart rate sensor with ESP32*. Recuperado el 28 de agosto de 2025, de MicrocontrollersLab. https://microcontrollerslab.com/esp32-heart-rate-pulse-oximeter-max30102/

Nextion. (s. f.). *Intelligent series introduction*. Recuperado el 3 Noviembre de 2025, de Nextion https://nextion.tech/intelligent-series-introduction/

Seithan.(2019-2020). *Guía para recibir y enviarcomandos desde pantallas Nextion.* Recuperado el 3 de Noviembre de 2025. Proyectos seitanis de https://seithan.com/Easy-Nextion-Library/Use-Nextion-General-View/

SunFounder Universal Maker Sensor Kit. (s.f.). *Lesson 14: Pulse oximeter and heart rate sensor module (MAX30102)*. Recuperado el 28 de agosto de 2025, de SunFounder Docs https://docs.sunfounder.com/projects/umsk/en/latest/03_esp32/esp32_lesson14_max30102.html

Tecnotizate. (s.f.). *Básicos ESP32: Mapeo de pines y sensores internos*. Recuperado el 28 de agosto de 2025, de https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/

10. Bibliografía

Canada Robotix. (2021, 22 de julio). *HC-020K Encoder Module Guide*. Recuperado el 28 de agosto de 2025, de https://www.canadarobotix.com/blogs/how-to/hc-020k-encoder-module-guide

DIYmore. (2019, 23 de diciembre). *2 Set HC-020K Double Speed Measuring Sensor Module with Photoelectric Encoders Kit*. Recuperado el 28 de agosto de 2025, de https://www.diymore.cc/products/2-set-hc-020k-double-speed-measuring-sensor-module-with-photoelectric-encoders-kit-top

Luis Llamas. (s.f.). *Cómo leer el sensor de Hall integrado en el ESP32*. Recuperado el 28 de agosto de 2025, de https://www.luisllamas.es/esp32-sensor-hall-integrado/

Luis Llamas. (s.f.). *ESP32 programming guide in Arduino environment*. Recuperado el 28 de agosto de 2025, de https://www.luisllamas.es/en/programming-guide-for-esp32-in-arduino-environment/

SOSLab. (2025, 23 de junio). *How to connect ESP32 with MAX30102: A comprehensive guide*. Recuperado el 28 de agosto de 2025, de SOSLab https://soslab.net/articles/how-to-connect-esp32-with-max30102-a-comprehensive-guide

11. Anexos

Anexo A. Código de revoluciones por minuto RPM de sensor de velocidad HC – 020.

```
void revoluciones_por_minuto() {
  //Velocidad
  unsigned long currentTime = millis();
  if (currentTime - lastSampleTime >= sampleInterval) {
    noInterrupts();
    unsigned int count = pulseCount;
    pulseCount = 0;
    interrupts();
    // Esta fórmula se usa para calcular las revoluciones por minuto (RPM), basándose en los
pulsos generados por el sensor y el tiempo de muestreo.
    rpm = (count / (float)slots) * (60000.0 / sampleInterval);

    ValorSTR = String(rpm);
    str_len = ValorSTR.length() + 1;
    char char_array[str_len];
    ValorSTR.toCharArray(char_array, str_len);
    t1.setText(char_array);
  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
    float barra = rpm * 0.0942 ;
    velocidad_angular.setValue(barra);


    // Calcular velocidad lineal en km/h
    float circumference = 3.1416 * wheelDiameter; // metros
    kmh = rpm * circumference * 60.0 / 1000.0;    // km/h
    ValorSTR = String(kmh);
    str_len = ValorSTR.length() + 1;
    char_array[str_len];
    ValorSTR.toCharArray(char_array, str_len);
    t2.setText(char_array);
    //delay(500);
    //    Serial.print("RPM: ");
    //    Serial.print(rpm);
    //    Serial.print(" | Velocidad: ");
    //    Serial.print(kmh);
    //    Serial.println(" km/h");

    lastSampleTime = currentTime;
  }
```

```
  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
  float barra = kmh * 20;
  velocidades.setValue(barra);


  if (rpm > 0) {
    Serial2.print("sleep=0");
    Serial2.write(0xFF); // Byte de terminación 1
    Serial2.write(0xFF); // Byte de terminación 2
    Serial2.write(0xFF); // Byte de terminación 3
  }
}
```

Anexo B. Código de tiempo de pedaleo de sensor de velocidad HC – 020.

```
void tiempo_pedaleo() {
  if (rpm > 0) {
    // Si la variable de control es mayor que cero, el tiempo AVANZA
    // Calculamos el tiempo transcurrido sumando el tiempo activo y el tiempo pausado
    tiempoTranscurridoActual = (millis() - tiempoInicio) + tiempoPausado;

    // --- Realizamos las conversiones a minutos y segundos ---
    segundosTotales = tiempoTranscurridoActual / 1000;
    minutos = segundosTotales / 60;
    segundos = segundosTotales % 60;
    digitalWrite(relevo,HIGH);

  } else {
    // Si la variable de control es CERO o NEGATIVA, el tiempo se PAUSA
    // Aquí actualizamos tiempoInicio para que, al reanudar, no haya un salto de tiempo
    // y el tiempoPausado acumule lo que no se contó.
    tiempoPausado = tiempoTranscurridoActual; // Guardamos el tiempo acumulado hasta ahora
    tiempoInicio = millis(); // Reiniciamos el "inicio" para la próxima vez que se active
  }

    if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }

  ValorSTR = String(minutos);
  str_len = ValorSTR.length() + 1;
  char char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t14.setText(char_array);

  // Calcular los segundos restantes (el resto de la división por 60)
  unsigned long segundos = segundosTotales % 60;

  ValorSTR = String(segundos);
  str_len = ValorSTR.length() + 1;
  char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t16.setText(char_array);

    if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
```

```
  //Cálculo cal
  //3.5 ejercicio suave
  //73 peso promedio de los colombianos
  horas=segundosTotales/3600.0;
  cal=3.5*73*horas;
  //kcal=kcal/1000;

  ValorSTR = String(cal);
  str_len = ValorSTR.length() + 1;
  char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t18.setText(char_array);


  //Cálculo energía en kWh
  if ((voltaje!=0)and(corriente!=0)){
  energia=voltaje*corriente;
  //Serial.println(energia);
  //energia=energia/1000.0;
  //Serial.println(energia);
  energia=energia*horas;
  //Serial.println(energia);
  }

  ValorSTR = String(energia);
  str_len = ValorSTR.length() + 1;
  char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t21.setText(char_array);

  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
}
```

Anexo C. Código de sensor de ritmo cardiaco MAX30102.

```
void ritmo_cardiaco_f() {
  // Ritmo Cardiaco
  long irValue = particleSensor.getIR();

  if (irValue > 20000) {  // Se detecta dedo (umbral más bajo)
    if (checkForBeat(irValue) == true) {
      long delta = millis() - lastBeat;
      lastBeat = millis();

      beatsPerMinute = 60 / (delta / 1000.0);

      // Filtramos valores fuera de rango fisiológico
      if (beatsPerMinute < 40 || beatsPerMinute > 180) {
        //Serial.print("Latido detectado pero descartado: ");
        //Serial.println(beatsPerMinute);
        beatsPerMinute = 0;
      } else {
        //Serial.print("Latido detectado: ");
        //Serial.println(beatsPerMinute);

        rates[rateSpot++] = (byte)beatsPerMinute;
        rateSpot %= RATE_SIZE;

        int sum = 0;
        for (byte x = 0; x < RATE_SIZE; x++) sum += rates[x];
        beatAvg = sum / RATE_SIZE;
      }
    }

    // Mostrar valores cada 500 ms
    static unsigned long lastPrint = 0;
    if (millis() - lastPrint > 500) {
      //Serial.print("IR: ");
      //Serial.print(irValue);
      //Serial.print("\tBPM: ");
      //Serial.print(beatsPerMinute);
      //Serial.print("\tPromedio BPM: ");
      //Serial.println(beatAvg);
      lastPrint = millis();
    }
  }
  else {
    // No hay dedo
    static unsigned long lastPrint = 0;
```

```
    if (millis() - lastPrint > 500) {
      //Serial.print("IR: ");
      //Serial.print(irValue);
      //Serial.println("\tPon tu dedo en el Sensor...");
      lastPrint = millis();
    }
    beatsPerMinute = 0;
    beatAvg = 0;
  }

  if (irValue < 10000) {
    irValue = 0;
  }
  else {
    irValue = irValue * 0.00067;
    if (irValue < 60) {
      irValue = 60;
    }
    else {
      if (irValue > 90) {
        irValue = 90;
      }
    }
  }

  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
  ValorSTR = String(irValue);
  str_len = ValorSTR.length() + 1;
  char char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t9.setText(char_array);
  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
}
```

Anexo D. Código de medición de corriente de sensor SEN 0291.

```
void corriente_dc() {
  //Corriente
  corriente = ina219.getCurrent_mA();
  if (corriente < 0) {
    corriente = 0;
  }
  ValorSTR = String(corriente);
  str_len = ValorSTR.length() + 1;
  char char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t3.setText(char_array);

  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }

  float barra = corriente;
  if (barra < 0) {
    barra = 0;
  }
  corrientes.setValue(barra);
}
```

Anexo E. Código de medición de voltaje de sensor SEN 0291.

```
void voltaje_dc() {
  //voltaje
  voltaje = ina219.getBusVoltage_V();
  ValorSTR = String(voltaje);
  str_len = ValorSTR.length() + 1;
  char char_array[str_len];
  ValorSTR.toCharArray(char_array, str_len);
  t4.setText(char_array);
  if (va0.getValue(&estadoActual)) {
    //Serial.println(estadoActual);
    reseteo();
  }
  float barra = voltaje * 8.33;
  voltajes.setValue(barra);
}
```