



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado
Facultad de Ingeniería de Sistemas e Informática
Unidad de Posgrado

**Técnica computacional en dispositivos embebidos
usando aprendizaje automático para el monitoreo
eficiente de señales fisiológicas**

TESIS

Para optar el Grado Académico de Doctor en Ingeniería de
Sistemas e Informática

AUTOR

Ricardo YAURI RODRIGUEZ

ASESOR

Dra. Nora Bertha LA SERNA PALOMINO

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Yauri, R. (2026). *Técnica computacional en dispositivos embebidos usando aprendizaje automático para el monitoreo eficiente de señales fisiológicas*. [Tesis de doctorado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Ricardo Yauri Rodriguez
Tipo de documento de identidad	DNI
Número de documento de identidad	41900467
URL de ORCID	https://orcid.org/0000-0001-9884-9317
Datos de asesor	
Nombres y apellidos	Nora Bertha La Serna Palomino
Tipo de documento de identidad	DNI
Número de documento de identidad	07665297
URL de ORCID	https://orcid.org/0000-0002-4292-344X
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	David Santos Mauricio Sánchez
Tipo de documento	DNI
Número de documento de identidad	06445495
Miembro del jurado 1	
Nombres y apellidos	José Alfredo Herrera Quispe
Tipo de documento	DNI
Número de documento de identidad	40362859
Miembro del jurado 2	
Nombres y apellidos	Frank Edmundo Escobedo Bailón
Tipo de documento	DNI
Número de documento de identidad	41671087
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento

Ubicación geográfica de la investigación	Edificio: Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Lima Latitud: -12.058333° Longitud: -77.083333°
Año o rango de años en que se realizó la investigación	2022 - 2025
URL de disciplinas OCDE	Ciencias de la Computación https://purl.org/pe-repo/ocde/ford#1.02.01 Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

**ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE
DOCTOR EN INGENIERÍA DE SISTEMAS E INFORMÁTICA**

A los diecisiete (17) días del mes de julio de 2026, siendo las 11:00 am., se reunieron en el Auditorio Alfredo Celso Alva Bravo, el Jurado de Tesis conformado por los siguientes docentes:

Dr. David Santos Mauricio Sánchez (Presidente)
Dr. José Alfredo Herrera Quispe (Miembro)
Dr. Frank Edmundo Escobedo Bailón (Miembro)
Dra. Nora Bertha La Serna Palomino (Miembro asesor)

Se inició la Sustentación invitando al candidato a Doctor **RICARDO YAURI RODRIGUEZ**, para que realice la exposición oral de la tesis para optar el Grado Académico de Doctor en Ingeniería de Sistemas e Informática, siendo la Tesis intitulada:

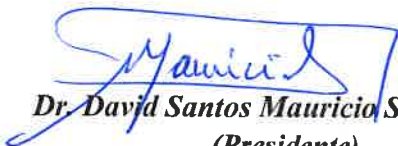
“TÉCNICA COMPUTACIONAL EN DISPOSITIVOS EMBEBIDOS USANDO APRENDIZAJE AUTOMÁTICO PARA EL MONITOREO EFICIENTE DE SEÑALES FISIOLÓGICAS”

Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, habiendo obtenido la siguiente calificación:

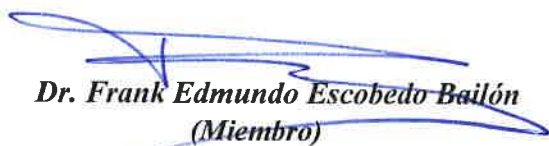
DIECISIETE (17) Muy BUENO

Por tanto, el presidente del Jurado, de acuerdo con el Reglamento General de Estudios de Posgrado, otorga al Magister **RICARDO YAURI RODRIGUEZ** el Grado Académico de Doctor en Ingeniería de Sistemas e Informática.

Siendo las 12:20 horas, el presidente del Jurado de Tesis, da por concluido el acto académico de Sustentación de Tesis.


Dr. David Santos Mauricio Sánchez
(Presidente)


Dr. José Alfredo Herrera Quispe
(Miembro)


Dr. Frank Edmundo Escobedo Bailón
(Miembro)


Dra. Nora Bertha La Serna Palomino
(Miembro Asesor)



Anexo 1

CERTIFICADO DE SIMILITUD

Yo, Nora Bertha La Serna Palomino en mi condición de asesor acreditado con Dictamen N° 000317-2025-UPG-VDIP-FISI/UNMSM de la tesis, cuyo título es **“Técnica Computacional en Dispositivos Embebidos usando Aprendizaje Automático para el Monitoreo Eficiente de Señales Fisiológicas”**, presentado por el egresado **Ricardo Yauri Rodriguez** para optar el **grado académico de Doctor en Ingeniería de Sistemas e Informática**.

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el **12 %** de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del **grado académico de Doctor en Ingeniería de Sistemas e Informática**.

Lima, 21 de agosto de 2026

Firma del Asesor

Nombres y Apellidos del asesor: Nora Bertha La Serna Palomino

DNI: 07665297

* Según corresponda: bachiller, magíster, doctor; título profesional o título de segunda especialidad

La vigencia del presente certificado corresponde a un período de noventa (90) días desde la fecha de su emisión; siempre que no haya observaciones en el texto durante el proceso de revisión por el jurado informante o surja alguna contingencia.

DEDICATORIA

A mi Esposa Edith, por su amor y constante apoyo incondicional.

A mis padres Ricardo y Nery por su compañía y dedicación constante

A mis abuelos Luisa, Sebastiana, Pablo y César por darme fuerza espiritual

Ricardo Yauri Rodriguez

Doctorando

AGRADECIMIENTO

A Dios por todo.

A la Doctora Nora La Serna, por su orientación, revisión y dedicación para el desarrollo de la presente investigación.

A los docentes de la Universidad Nacional Mayor de San Marcos por los conocimientos compartidos.

Al comité editorial de la revista indexada World Scientific and Engineering Academy and Society Transactions on Electronics, por la aceptación y publicación de los artículos de investigación que afianzaron la propuesta de tesis planteada.

Ricardo Yauri Rodriguez

Doctorando

ÍNDICE GENERAL

DEDICATORIA	ii
AGRADECIMIENTO	iii
ÍNDICE DE FIGURAS.....	viii
ÍNDICE DE TABLAS	xii
RESUMEN.....	xv
ABSTRACT.....	xvi
CAPÍTULO 1: INTRODUCCIÓN	1
1.1 Situación Problemática.....	1
1.2 Formulación del problema	4
1.3 Justificación Teórica	4
1.4 Justificación Práctica.....	5
1.5 Objetivos	6
1.5.1 Objetivo General	6
1.5.2 Objetivos Específicos.....	6
1.6 Contribuciones principales de la tesis	6
1.7 Organización de la tesis	8
CAPÍTULO 2: MARCO TEÓRICO.....	11
2.1 Marco filosófico o epistemológico	11
2.2 Antecedentes de investigación	13
2.3 Bases teóricas.....	15
2.3.1 Sistema de computación de borde.....	15
2.3.2 Nodos Sensores	18
2.3.3 Internet de las cosas.	19
2.3.4 Aprendizaje automático	20

2.3.5	Señales fisiológicas	32
2.3.6	Marco conceptual o Glosario	35
2.4	Estado del arte	37
2.4.1	Revisión sistemática de literatura.....	37
2.4.1.1	Metodología	37
2.4.1.2	Análisis.....	42
2.4.1.3	Discusión.....	56
2.4.1.4	Conclusiones	60
2.4.2	Trabajos relacionados.....	60
CAPÍTULO 3: METODOLOGÍA		66
3.1	Tipo y diseño de la investigación.....	66
3.1.1	Tipo de investigación	66
3.1.2	Diseño de investigación	66
3.2	Unidad de análisis	67
3.3	Población de estudio	67
3.4	Tamaño de la muestra	68
3.5	Selección de muestra.....	68
3.6	Técnicas de recolección de datos	68
3.6.1	Instrumentos.....	69
3.7	Técnicas de análisis e interpretación de datos.....	70
3.8	Hipótesis.....	71
3.8.1	Hipótesis general.....	71
3.8.2	Hipótesis específicas	71
3.9	Identificación y Operacionalización de variables	71
3.9.1	Identificación de variables	71
3.10	Matriz de consistencia.....	72

CAPÍTULO 4: DISPOSITIVO EMBEBIDO PARA MONITOREO DE SEÑALES FISIOLÓGICAS	74
4.1 Definición de la propuesta	75
4.1.1 Metodología para el desarrollo de la solución	75
4.1.2 Criterios de diseño.....	75
4.2 Etapa de desarrollo de dispositivo embebido.....	76
4.2.1 Características y Selección de tecnologías.....	76
4.2.2 Definición de componentes.....	78
4.2.3 Implementación.....	83
4.2.4 Resultados	94
4.2.5 Discusión de resultados.....	104
CAPÍTULO 5: DESARROLLO DE LA TÉCNICA COMPUTACIONAL EN EL DISPOSITIVO EMBEBIDO	106
5.1 Etapa de evaluación preliminar de modelos de clasificación.....	107
5.1.1 Preparación de datos y entrenamiento.....	109
5.1.2 Evaluación.....	112
5.1.3 Conversión	113
5.1.4 Ejecución en el dispositivo embebido.....	114
5.1.5 Resultados	115
5.1.6 Discusión de resultados.....	121
5.2 Etapa de despliegue de modelos con bases de datos integradas	122
5.2.1 Determinación de tecnologías	123
5.2.2 Modelo propuesto	127
5.2.3 Desarrollo del modelo	129
5.2.4 Validación de los modelos	133
CAPÍTULO 6: RESULTADOS Y DISCUSIÓN.....	138
6.1 Entorno de desarrollo	138

6.2	Resultados en el dispositivo embebido	142
6.3	Características de funcionamiento del dispositivo embebido	144
6.4	Pruebas de hipótesis	146
CAPÍTULO 7: CONCLUSIONES Y TRABAJOS FUTUROS		161
7.1	Conclusiones	161
7.2	Trabajos futuros	163
Bibliografía		164

ÍNDICE DE FIGURAS

Figura 1. Arquitectura de borde para dispositivos embebidos inteligentes	16
Figura 2. Pilares del Edge Computing (Mansouri & Babar, 2021).....	17
Figura 3. Modelo del Sistema (Hayyolalam et al., 2021)	17
Figura 4. Nivel del modelo IoT	20
Figura 5. Inteligencia cognitiva de borde para clasificación de señales médicas	20
Figura 6. Intersección de tecnologías para aplicaciones de TinyML	23
Figura 7. Consideraciones para implementación de TinyML en sistemas embebidos	25
Figura 8. Tareas delegadas a TensorFlow y TensorFlow Lite (Situnayake, 2020).....	26
Figura 9. ESP32-CAM (Cameron, 2021).....	29
Figura 10. Procesos de CRISP-DM (Huber et al., 2019).....	32
Figura 11. Esquema de un sistema de medición típico (Saltzman, 2010).....	33
Figura 12. Proceso de selección de artículos	40
Figura 13. Flujo de procesos para las etapas identificadas en la investigación	74
Figura 14. Propuesta de implementación de nodos sensores para el monitoreo	75
Figura 15. Bloques del nodo sensor inalámbrico.	77
Figura 16. Diagrama de interacción	77
Figura 17. Estructura de relaciones entre los componentes del NS1	78
Figura 18. Módulos de hardware utilizados para el prototipo: (a) ESP32, (b) Sensor DS18B20, (c) MAX30102 y (d) TP4056.....	80
Figura 19. Interacción de elementos del componente nodo sensor 1 (NS1).....	80
Figura 20. Ubicación del dispositivo de hardware – nodo sensor 1 (NS1).....	81
Figura 21. Elementos del nodo sensor 2 (NS2).....	81
Figura 22. Componentes que integran el nodo sensor 2 (NS2).....	82
Figura 23. Esquema de comunicación del nodo sensor con la plataforma de monitoreo.	83

Figura 24. Nodo sensor 1 implementado	84
Figura 25. Resultados de las lecturas de los sensores	84
Figura 26. Procesos de ejecución del dispositivo de hardware – nodo sensor 1	86
Figura 27. Diagrama de flujo para el cálculo de SPO2	87
Figura 28. Ubicaciones para el dispositivo de hardware - nodo sensor 2	87
Figura 29. Componentes del dispositivo de hardware - nodo sensor 2 integrados	88
Figura 30. Proceso de (a) derivada, (b) cuadratura (paso 3) y (c) suavizado (paso 4)	89
Figura 31. Pasos para el análisis de la señal ECG y cálculo de frecuencia cardiaca y respiratoria.....	90
Figura 32. Programa de detección de la frecuencia respiratoria	90
Figura 33. Programa de detección para la transmisión de datos usando MQTT	91
Figura 34. Datos capturados de la actividad de a) Corriendo (co), b) Caminando (ca) y c) estático (es)	92
Figura 35. Proceso de creación de la interfaz de visualización de datos	93
Figura 36. Resultados de la lectura de los parámetros de signos vitales.....	95
Figura 37. Salida de parámetros de signos vitales.	95
Figura 38. Ubicación del dispositivo de hardware – nodo sensor 1	96
Figura 39. Resultados de la implementación de los dos primeros pasos de procesamiento en el módulo (a) Nodo sensor 2 y (b) en Python	96
Figura 40. Resultado con (a) la derivada y (c) cuadratura- paso 3, (e) filtro suavizante- Paso 4; y en Python con (b) la derivada y (d) cuadratura-paso 3 y (4) filtro Suavizante-paso 4	97
Figura 41. Tiempos y umbrales utilizados para detección de picos.....	98
Figura 42. Visualización de datos de frecuencia respiratoria	98
Figura 43. Visualización de resultados generados por el dispositivo embebido-Nodo sensor2.	99
Figura 44. Ubicación del dispositivo embebido – nodo sensor 2	100
Figura 45. Visualización de datos del dispositivo embebido- sensor 1	101

Figura 46. Visualización de datos generados por el nodo sensor 2	102
Figura 47. Esquema de integración de los nodos sensores	102
Figura 48. Diagrama de interacción entre el DHC2, DHC1 y plataforma Adafruit	103
Figura 49. Esquema de pruebas de validación con integración de datos de los dos componentes de hardware	103
Figura 50. Visualización de datos de sensores enviados desde el dispositivo integrado	104
Figura 51. Diagrama del proceso de la técnica computacional.....	106
Figura 52. Esquema de la etapa de evaluación de modelos de clasificación	107
Figura 53. Procedimiento de implementación para la etapa de evaluación de modelos	108
Figura 54. Flujo de trabajo para el desarrollo de los modelos	109
Figura 55. Preparación y generación de modelos	112
Figura 56. Proceso de conversión para configuración 1 y 2	113
Figura 57. Proceso de conversión de los modelos	114
Figura 58. Conjunto de datos generados para evaluación en el nodo sensor.....	114
Figura 59. Exactitud promedio de los modelos evaluados de (a) árbol de decisión, (b) Bosques aleatorios, (c) SVM y (d) Bayes (GaussianNB) con muestras seleccionadas	116
Figura 60. Comparación de resultados de exactitud usando la primera configuración	117
Figura 61. Exactitud de modelos evaluados de (a) árbol de decisión, (b) Bosques aleatorios, (c) SVM y (d) Bayes (GaussianNB).....	118
Figura 62. Comparación de resultados de exactitud usando la configuración 2.....	119
Figura 63. Exactitud de modelos evaluados en software con muestras seleccionadas para los modelos de (a) ARN1, (b) ARN2, (c) ARN3 y (d) ARN4.....	120
Figura 64. Comparación de resultados de exactitud usando la configuración 3.....	121
Figura 65. Esquema de despliegue del proceso de clasificación	123

Figura 66. Base de datos CVD-Signal-Signs y Distribución de datos de cada variable	125
Figura 67. Base de datos utilizando los parámetros de NEWS y Distribución de datos de cada variable.....	126
Figura 68. Número de datos de clase de la Base de datos integrada.....	126
Figura 69. Proceso de clasificación.....	127
Figura 70. Proceso de generación de modelos propuesto	128
Figura 71. Muestra de la base de datos integrada	130
Figura 72. Proporción de cada clase	130
Figura 73. Preprocesamiento de datos	131
Figura 74. División de datos	131
Figura 75. Proceso de entrenamiento	132
Figura 76. Curvas de entrenamiento y validación de los algoritmos de SVM, árboles de decisión y bosques aleatorios	139
Figura 77. Curvas de entrenamiento y validación de los modelos basados en MLP y DNN	139
Figura 78. Comparación de resultados de exactitud para los modelos en el entorno de software	140
Figura 79. Comparación de resultados de exactitud en la etapa de validación y prueba de los modelos.....	141
Figura 80. Procesos para determinar el comportamiento de modelos en el dispositivo embebido	142
Figura 81. Comparación de resultados en el entorno de desarrollo y en el hardware.	144
Figura 82. Gráfico comparativo de datos obtenidos durante la evaluación en el dispositivo embebido	145
Figura 83. Matriz de resultados de las pruebas de comparaciones múltiples para tiempo de predicción	150

ÍNDICE DE TABLAS

Tabla 1 Frameworks TinyML (Sanchez-Iborra & Skarmeta, 2020).....	23
Tabla 2. Sensores utilizados en instrumentos biomédicos (Saltzman, 2010)	34
Tabla 3 Instrumentos médicos típicos (Saltzman, 2010)	34
Tabla 4. Cadenas de búsqueda según repositorios	38
Tabla 5. Criterio de selección y exclusión	39
Tabla 6. Número de artículos potenciales y seleccionados.....	40
Tabla 7. Distribución de revistas por cuartil	40
Tabla 8. Categorías de señales fisiológicas.....	42
Tabla 9. Señales fisiológicas en SMSFCB.....	43
Tabla 10. Sensores utilizados en dispositivos de borde	45
Tabla 11. Relación entre los sensores, y señales.....	47
Tabla 12. Algoritmos de aprendizaje en CB	49
Tabla 13. Algoritmos y su uso en el monitoreo	51
Tabla 14. Dispositivo de CB	53
Tabla 15. Comparación de trabajos representativos del estado del arte	59
Tabla 16. Operacionalización de las variables de la investigación	72
Tabla 17. Matriz de consistencia.....	73
Tabla 18. Señales fisiológicas relacionadas a cardiopatía (Meza, 2013).....	77
Tabla 19. Relación de colores y posición de los electrodos.....	82
Tabla 20. Descripción de etapas para el procesamiento de la señal ECG.....	89
Tabla 21 . Esquema de muestras captadas por el IMU	92
Tabla 22 . Relación entre los feeds y widgets generados.....	94
Tabla 23. Rangos de valores para estados de salud	108
Tabla 24. Rango de datos generados para cada parámetro	109

Tabla 25. Resultados de la evaluación de los modelos con datos aleatorios	112
Tabla 26. Características de los modelos evaluados con nuevos datos.....	115
Tabla 27. Resultados de comportamiento de los modelos en el dispositivo embebido	116
Tabla 28. Características de los modelos evaluados con nuevos datos.....	117
Tabla 29. Comportamiento de los modelos en el dispositivo embebido usando la configuración 2.....	118
Tabla 30. Resultado de evaluación de modelos con configuración 3	119
Tabla 31. Comportamiento de los modelos en el dispositivo embebido usando la configuración 3.....	120
Tabla 32. Matriz de confusión con datos de validación para el modelo SVM	133
Tabla 33. Métricas de validación con el modelo SVM.....	134
Tabla 34. Matriz de confusión con datos de validación para el modelo DT.....	134
Tabla 35. Métricas de validación con el modelo DT	134
Tabla 36. Matriz de confusión con datos de validación para el modelo RF	135
Tabla 37. Métricas de validación con el modelo RF.....	135
Tabla 38. Matriz de confusión con datos de validación para el modelo de MLP	135
Tabla 39. Métricas de validación con el modelo MLP	136
Tabla 40. Matriz de confusión con datos de validación para modelos basados en DNN	136
Tabla 41. Métricas de validación con el modelo de DNN	137
Tabla 42. Resumen de métricas con datos de validación en el entorno de desarrollo	137
Tabla 43. Resumen de métricas con datos de prueba en el entorno de desarrollo...	138
Tabla 44. Promedio de métricas obtenidas con datos de pruebas seleccionados en el entorno de software	143
Tabla 45. Promedio de las métricas obtenidas en el dispositivo embebido con los datos de pruebas seleccionados	143

Tabla 46. Promedio de características de funcionamiento en el dispositivo embebido	145
Tabla 47. Tiempos de predicción obtenidos	148
Tabla 48. Resultados de la verificación de supuestos	149
Tabla 49. Diferencias entre grupos para el tiempo de predicción.....	149
Tabla 50. Verificación de supuestos con datos de exactitud.....	151
Tabla 51. Determinación de diferencias entre grupos respecto a la exactitud	151
Tabla 52. Verificación de supuestos sobre el uso de memoria RAM	153
Tabla 53. Diferencias entre grupos respecto al uso de memoria RAM	153
Tabla 54. Matriz de resultados de comparaciones múltiples de memoria RAM	154
Tabla 55. Verificación de supuestos sobre el uso de memoria FLASH	155
Tabla 56. Diferencias entre grupos respecto al uso de memoria FLASH.....	155
Tabla 57. Matriz de resultados de comparaciones múltiples de memoria FLASH..	156
Tabla 58. Verificación de supuestos sobre el consumo de corriente eléctrica.....	157
Tabla 59. Diferencias entre grupos respecto al consumo de corriente eléctrica	157
Tabla 60. Matriz de resultados de comparaciones múltiples sobre corriente eléctrica	158
Tabla 61. Verificación de supuestos sobre el tamaño de archivo para cada modelo	159
Tabla 62. Diferencias entre grupos respecto al tamaño de los archivos	159
Tabla 63. Resultados de comparaciones múltiples sobre el tamaño de los archivos	160

RESUMEN

El monitoreo clínico continuo de los pacientes enfrenta limitaciones debido a la baja calidad de los datos obtenidos durante sus actividades cotidianas, la falta de infraestructura médica adecuada y la necesidad de detectar actividades anómalas en el momento de su ocurrencia. En estos casos el uso de dispositivos portátiles con modelos de inteligencia artificial permite el análisis de señales fisiológicas asociadas a cardiopatía para apoyo a la toma de decisiones. Es por esto que la investigación propone el desarrollo y validación de una técnica computacional basada en aprendizaje automático implementada en dispositivos embebidos de recursos limitados para el monitoreo de señales fisiológicas asociadas a cardiopatía mediante el uso de datos sintéticos y bases de datos fisiológicas públicas, sin involucrar recolección directa de datos de pacientes.

Para ello, se integran herramientas de computación de borde y Tiny machine learning (TinyML), que permiten el procesamiento y análisis de datos en un dispositivo portátil, evaluando los modelos mediante métricas de exactitud, consumo de corriente eléctrica, tiempo de inferencia y uso de memoria RAM y FLASH. Como primera etapa se realizó el desarrollo de un dispositivo embebido para monitoreo de señales con conectividad inalámbrica. La segunda etapa se enfoca en la creación y evaluación de modelos de aprendizaje automático para la clasificación del estado de salud en el dispositivo de hardware, organizada en dos fases: la primera que se encarga de la evaluación preliminar de modelos de clasificación utilizando datos sintéticos, y la segunda describe el despliegue y validación final de la técnica mediante conjunto de datos de señales fisiológicas reales. La investigación permitió elaborar y validar una técnica computacional en un dispositivo embebido para el monitoreo eficiente de señales fisiológicas asociadas a cardiopatía, determinando la variación del comportamiento de los modelos según las métricas de evaluación. Asimismo, los resultados muestran que los modelos desplegados son una alternativa viable considerando su uso en hardware embebido con recursos limitados.

Palabras clave: Monitoreo fisiológico, aprendizaje automático, computación de borde, TinyML, clasificación, TensorFlow Lite, ESP32

ABSTRACT

Continuous clinical monitoring of patients faces limitations due to the low quality of data obtained during their daily activities, the lack of adequate medical infrastructure, and the need to detect anomalous activity as it occurs. In these cases, the use of wearable devices with artificial intelligence models allows for the analysis of physiological signals to support decision-making. Therefore, this research proposes the development and validation of a computational technique based on machine learning, implemented in resource-constrained embedded devices, for monitoring physiological signals associated with heart disease using synthetic data and public physiological databases, without involving direct patient data collection.

This research integrates edge computing and Tiny Machine Learning (TinyML) tools, enabling data processing and analysis on a portable device. Models are evaluated using metrics such as accuracy, power consumption, inference time, and RAM and Flash memory usage. The first stage involved developing an embedded device for wireless signal monitoring. The second stage focuses on creating and evaluating machine learning models for health status classification on the hardware device. This stage is organized into two phases: the first involves the preliminary evaluation of classification models using synthetic data, and the second describes the deployment and final validation of the technique using a dataset of real physiological signals. The research successfully developed and validated a computational technique on an embedded device for the efficient monitoring of physiological signals associated with heart disease, determining the variation in model behavior according to the evaluation metrics. Furthermore, the results demonstrate that the deployed models are a viable alternative for use on embedded hardware with limited resources.

Keywords: Physiological monitoring, machine learning, edge computing, TinyML, classification, TensorFlow Lite, ESP32

CAPÍTULO 1: INTRODUCCIÓN

1.1 Situación Problemática

El monitoreo clínico de los pacientes requiere el seguimiento y toma de decisiones proactivas y reactivas donde, usualmente, la adquisición de las señales se realiza mientras las personas continúan con sus actividades diarias, lo cual afecta la calidad de los datos (Escalona et al., 2023) (J.-J. Wang et al., 2023). Además, cuando existe una alta demanda para la atención médica en hospitales es necesario fortalecer el uso de soluciones de monitoreo remoto y seguimiento para suplir las deficiencias actuales (R. Santos et al., 2024). En este caso también se ayuda a evitar el problema relacionado al síndrome post-hospitalario (V. Patel et al., 2021a) (Mangiameli et al., 2024) donde, para la vigilancia de parámetros de salud, se puede usar dispositivos portátiles para ayudar a reducir la prevalencia de este síndrome.

El proceso de monitoreo de parámetros fisiológicos, en centros médicos, necesita la infraestructura e intervención del personal médico adecuado en la zona donde el paciente debe movilizarse para recibir atención, siendo un proceso que se vuelve más complejo cuando estos se encuentran en zonas alejadas (Virag et al., 2025) (S. Singh et al., 2021). Además, este problema se agrava cuando las evaluaciones, que se realizan sobre el paciente, requieren de estancias prolongadas que pueden llegar a varios días para brindar una evaluación fiable y donde los pacientes usualmente abandonan el centro médico por la demora de atención (Dhanwanth et al., 2023)(Gallos et al., 2022). El efecto de este problema genera una sobrepoblación de pacientes y falta de recursos tecnológicos médicos utilizados para el monitoreo de señales fisiológicas asociadas a cardiopatía generando que el personal médico trabaje tiempo extra afectando la confianza del tratamiento y generando posteriormente un reingreso por afecciones más graves (Dall’Ora et al., 2025).

En el caso de sensores conectados sobre el paciente, estos se pueden integrar con un componente inteligente para su adquisición y procesamiento, utilizando una serie de elementos que dificultan el desplazamiento y alteran el estilo de vida de las personas durante la adquisición de señales fisiológicas (Jeong et al., 2022)(McMurray et al., 2025). Es aquí donde un dispositivo embebido (dispositivos de hardware con recursos limitados) puede utilizar algoritmos de inteligencia para apoyar a la toma de datos y realizar una transducción adecuada (por ejemplo, mediante el uso de sensores

flexibles) (Sreenilayam et al., 2020) teniendo como restricciones de funcionamiento el bajo consumo de energía, flexibilidad y facilidad de implementación (Z. Huang et al., 2024)(Suppiah et al., 2024). Para realizar la adquisición de estas señales se requieren largos períodos de tiempo para su adquisición y/o generación de una base de datos lo cual puede provocar un aburrimiento por parte del usuario y siendo necesario recurrir a procesos indirectos sin contacto de medición (por ejemplo con sensores de electrocardiograma o Glucosa) (Andellini et al., 2023)(X. Huang et al., 2024) .

Actualmente la cantidad de datos obtenidos de las señales fisiológicas permiten identificar patrones, pero su análisis es un problema que requiere de sistemas computacionales para descubrir nueva información (Y. Pan, 2025) y aportar a la implementación de aplicaciones de telesalud y monitoreo (Adolfo et al., 2021). En el caso de las tareas de clasificación, estas necesitan estrategias que reduzcan el costo computacional, como el uso de ventanas temporales o la fusión de sensores, lo que hace necesario categorizar y comprender dichos procedimientos para seleccionar el más adecuado según el contexto de aplicación (Karan & Kaygun, 2021) (Rim et al., 2020) (Bizzego et al., 2020) pero no realizan una evaluación del comportamiento del dispositivo de hardware durante su funcionamiento. A pesar de los avances reportados, aún existe una limitada sistematización de estos enfoques cuando se consideran las restricciones propias del hardware embebido.

Por otro lado, el monitoreo y transmisión continua de datos requiere el uso de grandes recursos de hardware, software y/o capacidad computacional, que no son siempre adecuados para las soluciones con dispositivos portátiles utilizadas en monitoreo remoto (Internet de las cosas) (Amin & Hossain, 2021) debido a las limitaciones de hardware reducido (Foukalas, 2020). En este caso existe un problema relacionado a la manera en la cual se integran y evalúan los mecanismos para acercar la capacidad computacional al borde de la red y generar soluciones en tiempo real que requieren baja latencia e integridad de los datos (Qadri et al., 2020) (Xia et al., 2025). Debido a la necesidad de aliviar los problemas críticos relacionados con el consumo de energía, seguridad, la utilización del ancho de banda y la escalabilidad, el uso de la computación de borde (Edge Computing) ayuda a identificar las condiciones críticas de salud de forma temprana y realizar el monitoreo apropiado por médicos, personal, hospitales y organismos de investigación (Kuchuk & Malokhvii, 2024)(K. Wang et al., 2024). Además, los procesos de computación de borde, tienen problemas

relacionados al acondicionamiento de señales obtenidas desde sensores, interpretarlas en el contexto del evento y deducir alguna respuesta lógica y racional orientada al seguimiento de la salud usando dispositivos IoT (Internet de las cosas) (Otoum et al., 2021) (Kokhazad et al., 2025).

Por lo tanto, en el presente trabajo se busca contribuir a resolver el problema específico relacionado al monitoreo de señales fisiológicas asociadas a cardiopatía usando dispositivos embebidos integrados con modelos de clasificación considerando el aspecto computacional. En este estudio la investigación no se limita al área de cardiopatía y se pueden utilizar en otras áreas como la respiratoria, neurológica y metabólica utilizando bases de datos públicas y anónimas.

En base a estudios revisados (Lahoti, 2024) (Kamalakaran et al., 2025) (Emetere & Akinlabi, 2020), se considera la técnica computacional como un marco metodológico donde se evalúan diferentes modelos de inteligencia artificial. En este sentido, una técnica computacional no se limita a un algoritmo aislado, sino que constituye un conjunto organizado de etapas que incluye el tratamiento de datos, la aplicación de modelos y la obtención de resultados.

Si bien, durante el desarrollo de la investigación, se evalúan distintos modelos de clasificación, estas son instancias evaluadas dentro de la técnica computacional. Esta evaluación comparativa permite determinar la relación de compromiso entre la exactitud, consumo de corriente eléctrica, uso de memoria RAM y FLASH y tiempo de inferencia, como parte del proceso de validación de la técnica.

La técnica computacional propuesta no es sólo una implementación aplicada, sino que integra componentes que representan aportes en el ámbito de la computación de borde aplicada al monitoreo fisiológico. En primer lugar, se aborda la contribución tecnológica como el aporte al diseño y construcción de un dispositivo embebido para la adquisición y procesamiento local de señales fisiológicas. Además, se tiene una contribución metodológica como un procedimiento sistemático de evaluación y despliegue de modelos de clasificación en un dispositivo embebido. Finalmente, se obtiene una contribución experimental, mediante una evaluación cuantitativa que analiza las métricas de comportamiento en el dispositivo de hardware.

1.2 Formulación del problema

Problema: ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía permite la selección de modelos eficientes?

Problemas específicos:

- ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de exactitud?
- ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de consumo de corriente eléctrica?
- ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de tiempo de inferencia ?
- ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de memoria RAM y FLASH?

1.3 Justificación Teórica

El trabajo se justifica teóricamente debido a la necesidad de generar un aporte a los conocimientos sobre procesos de monitoreo de señales fisiológicas asociadas a cardiopatía mediante el uso de tecnologías embebidas para procesar y analizar datos en tiempo real. Esto se basa en la revisión bibliográfica y búsqueda sistemática de la literatura que muestra los fundamentos teóricos que involucra la detección de eventos anómalos integrando inteligencia artificial en sistemas embebidos con pocos recursos de hardware. Debido a lo mencionado anteriormente, se analiza teóricamente métodos para la implementación de algoritmos de aprendizaje automático reducido utilizando el concepto de computación de borde.

En la investigación se analizan los principios de la computación de borde (Edge computing) y del aprendizaje automático de pequeña escala (Tiny machine learning, TinyML), para la descentralización del procesamiento de señales orientado a la reducción de la latencia y el consumo energético. De este modo, el uso de modelos de

inteligencia artificial optimizados para hardware se justifica como una estrategia teórica para evaluar la eficiencia energética, autonomía de funcionamiento y capacidad de análisis local de señales.

Finalmente, la investigación muestra los fundamentos teóricos que permiten comprender cómo, la integración de algoritmos de aprendizaje automático en hardware embebido, afecta la exactitud de modelos de clasificación, reduce el tiempo de inferencia, y afecta el consumo energético, sin que implique una validación clínica de los resultados obtenidos.

1.4 Justificación Práctica

El presente trabajo se justifica de manera práctica por su aplicabilidad para uso de sistemas de monitoreo con recursos de hardware limitados. En los centros hospitalarios y zonas rurales, la sobrepoblación de pacientes y la falta de dispositivos tecnológicos dificultan el seguimiento continuo de las señales fisiológicas por lo que una técnica computacional implementada en dispositivos embebidos es una alternativa que permite la vigilancia médica remota. La investigación describe el proceso de implementación para la ejecución de procesos de clasificación. Además, los algoritmos de aprendizaje automático, se evalúan e implementan en dispositivos de hardware basados en computación de borde, que no requieran la transmisión intensiva de datos hacia aplicaciones en Internet.

Asimismo, se presenta una aplicación práctica al integrar sensores fisiológicos, algoritmos de aprendizaje automático y hardware de bajo consumo energético en un sistema embebido portátil. Esta combinación permite la adquisición, procesamiento y análisis de datos fisiológicos mejorando la capacidad de respuesta ante emergencias. Finalmente, el trabajo se justifica prácticamente, dado que utiliza herramientas como microcontroladores ESP32, bibliotecas de aprendizaje automático (TensorFlow, Scikit-Learn) y tecnologías de TinyML optimizadas para hardware de recursos limitados. Su implementación permite evaluar métricas de desempeño, como el consumo de corriente eléctrica, el tiempo de inferencia y la memoria RAM y FLASH utilizada.

1.5 Objetivos

1.5.1 Objetivo General

Elaborar y validar una técnica computacional en dispositivos embebidos usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía, que permita seleccionar modelos eficientes con los mejores indicadores de comportamiento en hardware.

En este estudio se usaron señales fisiológicas asociadas a cardiopatía, sin embargo la investigación es potencialmente aplicable para señales fisiológica asociadas al área respiratoria, neurológica y metabólica.

1.5.2 Objetivos Específicos

- a) Determinar la exactitud alcanzada por la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.
- b) Determinar el consumo de corriente eléctrica asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.
- c) Determinar el tiempo de inferencia asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.
- d) Determinar el uso de memoria RAM y FLASH asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.

1.6 Contribuciones principales de la tesis

Esta tesis presenta contribuciones en el ámbito del monitoreo fisiológico de señales asociadas a cardiopatía mediante dispositivos embebidos y aprendizaje automático usando datos reales públicos. Las contribuciones que se proponen son las siguientes:

a) Contribución metodológica. En esta investigación el aporte metodológico es la Técnica computacional desarrollada para el monitoreo eficiente de señales fisiológicas cardíacas basada en aprendizaje automático en dispositivos embebidos de recursos limitados, que de manera general se describe a continuación.

La técnica empieza con la etapa de adquisición de señales fisiológicas cardíacas, que incluye la determinación de criterios para la selección de: sensores, el microcontrolador y los mecanismos de comunicación. Luego, se realiza la etapa del pre procesamiento de las diferentes señales que se manejan y la extracción de características, permitiendo generar los datasets adecuados. Seguidamente, se realiza la etapa del procesamiento automático con cinco algoritmos seleccionados, y se generan los modelos de clasificación correspondientes del estado de salud de las personas.

Luego, se establece un procedimiento para la evaluación de los modelos de clasificación compatibles con hardware embebido, el que se desarrolla en dos etapas complementarias: la primera etapa es la evaluación de los modelos orientado a analizar su viabilidad computacional, y la segunda etapa es la evaluación con base de datos integrada que permite validar el desempeño de los modelos bajo condiciones más representativas del escenario real de aplicación. Cabe indicar que, en esta última etapa de evaluación se utilizan las métricas de exactitud, consumo energético, uso de memoria RAM y FLASH y tiempo de inferencia.

b) Contribución experimental. La contribución empírica permite validar el funcionamiento de los modelos integrados en hardware con diferentes configuraciones, que permite un monitoreo eficiente de las señales fisiológicas asociadas a cardiopatías. Para ello, se realizan diferentes pruebas, y establece un método con varias fases.

En primer lugar, se realiza la evaluación en el entorno de desarrollo de los modelos que se han generado, utilizando datos de prueba que tienen condiciones equivalentes al despliegue posterior en hardware, y que selecciona los modelos con mejor desempeño. Luego, en las pruebas en hardware los modelos seleccionados y con formatos compatibles con el dispositivo embebido, se evalúa el desempeño obtenido con métricas de exactitud. Posteriormente, con otras pruebas se miden con las métricas

de tiempo de inferencia, consumo de corriente y uso de memoria RAM y FLASH respectivamente.

Como paso adicional, con los datos obtenidos en los dispositivos embebidos se realizó la comprobación estadística para determinar si existen diferencias significativas entre los modelos en hardware respecto a exactitud, tiempo de predicción, consumo de corriente y uso de memoria. El procedimiento incluye verificación de supuestos, pruebas paramétricas y no paramétricas, y comparaciones múltiples.

c) Contribución tecnológica. En esta tesis es el diseño e implementación de un dispositivo embebido para el monitoreo de señales fisiológicas, integrando sensores, procesamiento local y tecnologías de comunicación IoT. El dispositivo de hardware incluye sensores fisiológicos y definición de una arquitectura de hardware y software. Además, se integran mecanismos de procesamiento de señales en el borde y algoritmos para la ejecución local de los modelos de clasificación. Este prototipo es la plataforma de evaluación para la técnica computacional propuesta, y de esta manera, la contribución tecnológica demuestra su aplicación en sistemas embebidos de recursos limitados.

1.7 Organización de la tesis

En el capítulo 1 se describe la situación problemática en la sección 1.1, continuando con la Sección 1.2 que muestra la formulación del problema. La justificación teórica y práctica de la investigación se describe en la sección 1.3 y 1.4 respectivamente, mientras que en la sección 1.5 se plantea el objetivo general y específico. Finalmente, en la sección 1.6 se muestra la organización del trabajo de tesis.

El capítulo 2 muestra el marco teórico iniciando con la sección 2.1 que trata sobre el marco filosófico, continuando con la sección 2.2. de antecedentes de la investigación. Posteriormente se continúa con la sección 2.3 de bases teóricas. En la Sección 2.4 se realiza una revisión sistemática de la literatura mediante trabajos relacionados con métodos para implementar algoritmos de inteligencia artificial en sistemas de hardware embebidos para monitoreo de señales fisiológicas utilizando una técnica computacional. A continuación, se realiza una conceptualización de las principales formas para realizar la implementación de algoritmos de aprendizaje automático en sistemas embebidos con pocos recursos de hardware, de las cuales se elige las más

apropiada para el trabajo de investigación. Además, se muestra una evaluación de los algoritmos para clasificación y monitoreo de señales fisiológicas. Luego se plantea una revisión de los dispositivos de hardware apropiados para la implementación de soluciones en IoT con capacidades de computación de borde. Finalmente se realiza un resumen de la revisión de la literatura.

En el capítulo 3 se muestra la metodología usada para el desarrollo del trabajo de tesis, indicando el tipo y diseño de investigación, la unidad de análisis, población de estudio y tamaño de la muestra. Se continúa mostrando el enfoque usado para la recolección de datos, instrumentos utilizados, los métodos de análisis de datos. Se termina la sección las hipótesis, operacionalización de las variables y matriz de consistencia.

En el capítulo 4 se muestra el primer aporte realizado en la investigación enfocado a la selección y justificación de las tecnologías, propuesta de solución y el desarrollo del dispositivo de hardware embebido (nodo sensor). Posteriormente se realiza la implementación y evaluación del funcionamiento. Finalmente se concluye esta sección mostrando la descripción de pruebas y resultados obtenidos.

En el capítulo 5 se muestra otro aporte de la investigación relacionada a la técnica computacional. Se describe en la sección 5.1 la evaluación inicial de modelos de clasificación del estado de salud con datos generados en software. En esta etapa se describen las herramientas de software utilizadas para integrar los modelos de inteligencia artificial en dispositivos de hardware. Además, se muestran las métricas respecto a la evaluación del funcionamiento. Finalmente se termina con resultados y discusión de estas pruebas.

Posteriormente, en la sección 5.2 se realiza el despliegue de modelos de clasificación en dispositivos de hardware utilizando conjunto de datos con datos fisiológicos relacionados a cardiopatía obtenidos por medio de repositorios y mediante pruebas de laboratorio. Se presenta la elección de las tecnologías a utilizar, el modelo e implementación propuesto y su validación de funcionamiento generando resultados que se describen en el capítulo 6. Posteriormente en el Capítulo 7 se observan las conclusiones y trabajos futuros a realizar.

Finalmente, se presentan las referencias bibliográficas y anexos que complementan las evaluaciones realizadas, mostrando las publicaciones realizadas correspondientes a los resultados de la investigación.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Marco filosófico o epistemológico

El monitoreo de las señales fisiológicas y signos vitales es un método que aporta a la mejora del cuidado de la salud. Debido a los problemas que ocurren por la saturación de los centros hospitalarios, por personas de salud crítica o aquellas que requieran monitorearse de forma continua en modo ambulatorio, se ha identificado el uso de tecnologías de monitoreo en las áreas de seguridad, deporte y salud las cuales se relacionan con aspectos tecnológicos para la implementación de soluciones de ingeniería. La necesidad de tomar acciones inmediatas (para detectar eventos anómalos en las señales relacionadas a la salud de las personas) hace necesario utilizar dispositivos de hardware que realicen el análisis de forma local, lo cual se puede lograr mediante la implementación de dispositivos embebidos inteligentes en el campo denominado computación de borde y posteriormente realizar el desarrollo de dispositivos de hardware y software. La presente investigación considera como una herramienta importante las tecnologías de computación de borde para aportar soluciones al monitoreo de signos vitales y señales fisiológicas de una persona. Tomando como base este criterio es necesario hacer un análisis de la importancia de sistemas de computación de Borde que implementan algoritmos de inteligencia artificial (IA) en hardware con bajos recursos computacionales y aporten una solución al área del monitoreo eficiente de la salud desde el punto de vista filosófico o epistemológico.

En el contexto filosófico, se describe el conocimiento y entendimiento al empezar el estudio de investigación y saber la finalidad y justificación. En esta sección se trata el conocimiento producido, la interpretación de la realidad por el investigador y el enfoque adecuado al objeto de investigación (T. Liu & Mitcham, 2024) (Martínez, 1998). La fundamentación epistemológica es necesaria ya que permite estudiar la entidad del conocimiento científico que se trata, y da a conocer cuál ha sido su proceso de constitución y desarrollo. Además, se menciona el conocimiento a partir del análisis de la evolución de la inteligencia artificial, aprendizaje automático y su aplicabilidad para la detección de eventos fisiológicos.

De esta forma, el monitoreo de la salud mediante la implementación de dispositivos de computación de borde con algoritmos de aprendizaje automático en nodos sensores

con hardware reducido está orientada a la justificación de la investigación, vinculada a la salud y sistemas embebidos inteligentes. Cabe mencionar que el uso de dispositivos de computación de borde integrando tecnologías de inteligencia artificial, toma un importante papel en el monitoreo de la salud de las personas, lo cual aporta a generar estrategias de salud pública.

La inteligencia artificial y áreas relacionadas como el aprendizaje automático, entra en contacto con la filosofía en diversos puntos, pero particularmente en las áreas de la lógica y la epistemología. Así, la epistemología en el siglo XX no busca sólo imitar la inteligencia natural sino crear máquinas que piensen (Doroudi, 2024) . La evolución de esta tecnología inició cuando el pensamiento humano se intentó automatizar a través del cálculo matemático y técnicas computacionales (Guijarro & González, 2010) (Wickberg & Gärdebo, 2023). En relación a ello, durante la década de los 50, pioneros de la informática, como Donald Richie, inició el proyecto de construir inteligencias artificiales (Dreyfus, 2014) y continuó en 1966 con Joseph Weizenbaum que desarrolló un programa de computadora llamado ELIZA (Weizenbaum, 1977), capaz de interactuar con un ser humano mediante un teclado y un monitor.

Existen elementos éticos a considerar en el marco de la inteligencia artificial, lo que constituye un aspecto ampliamente debatido por los riesgos que su uso mal intencionado podría ocasionar con fines nocivos y maliciosos, lo que genera dilemas éticos en su uso y se puede observar en cuatro campos: identidad, invasión de los entornos, intimidad y seguridad (González Arencibia & Martínez Cardero, 2020). Una de las principales amenazas es la que se refiere a la privacidad, donde la tendencia es afectar la intimidad de las personas. Esto además se agrava por el empleo de sistemas de información, a la que se han dotado de poder para la toma de decisiones de manera autónoma en ocasiones riesgosas, presentándose un dilema moral.

El uso de dispositivos con tecnologías de computación de borde e inteligencia artificial permite tener una mayor autonomía de funcionamiento para lo cual es importante considerar el modelo de aprendizaje automático a desplegar, el tipo de tecnología inalámbrica, periodo de transmisión de datos y tipo de señales adquiridas por los sensores. Además, los algoritmos de aprendizaje automático permiten analizar los datos obtenidos por sensores relacionados al estado de los parámetros fisiológicos de una persona generando recomendaciones y conclusiones acerca de su diseño y

requisitos para su implementación, basado en el uso de algoritmos reducidos para la ejecución del proceso de inferencia local (Koufos et al., 2021).

Se debe tener en cuenta que la tecnología es un complemento y no el centro para resolver los usos negativos de la ciencia, teniendo la inteligencia artificial potencial para el desarrollo social y en la industria de software, lo cual es importante debido a que se están proyectando cuestiones éticas sobre la forma en que se está empleando la tecnología. Además, permite enfrentar las amenazas del contexto actual relacionadas a la salud, por lo que el uso de tecnologías de inteligencia artificial, aprendizaje automático y computación de borde, deben desarrollarse en función de los intereses humanos.

2.2 Antecedentes de investigación

Como resultado de la búsqueda realizada, se ha encontrado información sobre la situación problemática de la investigación, que ha servido para elaborar del presente trabajo. En el caso de los programas de inteligencia artificial (IA) se aplican a procedimientos de diagnóstico, desarrollo de protocolos de tratamiento y seguimiento de pacientes usando tecnologías como aprendizaje automático, Big Data y tecnología blockchain en la medicina para la atención médica (Supriya & Chattu, 2021). Además el crecimiento del Internet de las cosas médicas (IoMT) ha generado preocupaciones sobre la seguridad, la privacidad y la gestión de datos médicos centralizados en los sistemas de salud inteligentes donde se busca mejorar la eficiencia y la precisión del monitoreo médico, usando un marco federado Blockchain-IoT (FBCI-SHS) para detección proactiva de enfermedades (Bhasker et al., 2025).

Por otro lado, a pesar de los avances en los programas de recuperación pacientes, la estancia postoperatoria prolongada incrementa los costos y la ocupación hospitalaria por lo que existe la necesidad de evaluar estrategias que permitan el alta temprana sin comprometer la seguridad del paciente abordando el problema sobre si el monitoreo domiciliario, mediante telesalud, puede garantizar una recuperación segura en pacientes (Mangiameli et al., 2024). Además, en algunos casos es necesario un monitoreo constante y una intervención oportuna (como en embarazos de alto riesgo), sin embargo, los sistemas tradicionales de seguimiento tienen limitaciones en la continuidad del cuidado tras el alta hospitalaria (Nazari et al., 2024).

En el caso de las frecuentes hospitalizaciones y estancias prolongadas, estas representan una carga económica extremadamente alta para las instituciones sanitarias, donde el número de pacientes ancianos y con enfermedades crónicas aumenta continuamente siendo evidente la falta de médicos especializados. Es aquí donde la tele monitorización permite resolver tales necesidades investigando diferentes metodologías y dispositivos para la monitorización domiciliaria eficaz de los pacientes cardiológicos (Faragli et al., 2021). En el caso de la Organización Mundial de la Salud (OMS), esta reconoce los dominios ambientales, conductuales, fisiológicos y psicológicos que impactan negativamente en la salud, el bienestar y la calidad de vida (CdV) (Haghi et al., 2021) y es por esta necesidad que la evaluación en tiempo real de la salud del usuario es necesaria para la prestación de atención inmediata usando, en algunos casos, dispositivos portátiles para la vigilancia (WIHMS) (Memon et al., 2020).

El concepto de monitoreo de atención médica móvil (mHealth) ha sido apoyado por avances recientes en la fabricación de dispositivos móviles y el surgimiento de sensores portátiles (McMurray et al., 2025). El problema de estas tecnologías está asociado a la utilización de sensores integrados en dispositivos inteligentes para recopilar los datos fisiológicos, que carecen de precisión y exactitud. Además, al integrarse con tecnologías del Internet de las cosas, se realiza el monitoreo continuo de la salud (Alabdulqader et al., 2024) encontrando como dificultad la utilización de sensores adecuados.

En el aspecto relacionado a los problemas de implementación tecnológica para la atención médica inteligente, es necesario considerar los desafíos de la investigación actuales y futuros donde los análisis pocos exhaustivos, que no realizan comparaciones y evaluaciones de tecnologías, dificultan su integración, como en el caso del Internet de las cosas médicas (IoTMT), inteligencia artificial, computación de borde o nube para mejorar la privacidad y seguridad (Efendi et al., 2025) (M. Cao et al., 2024).

Las arquitecturas y enfoques que explotan la computación de borde y niebla son una alternativa para compensar las debilidades de la nube pero su implementación requiere una capacidad computacional elevada (Ghafari & Mansouri, 2025), es por eso que se realizan estudios para evaluar su factibilidad de despliegue considerando el uso eficiente de memoria y latencia en la respuesta (Rajagopal & Subramanian, 2025). En

el caso específico de los modelos basados en redes neuronales profundas con transmisión de datos hacia la nube, no es viable para aplicaciones que requieren baja latencia (Véstias et al., 2020) y carga de trabajo (Mehmood et al., 2021). Sin embargo, los dispositivos periféricos son menos potentes que los servidores en la nube y están sujetos a limitaciones de energía, por lo que se requiere evaluar modelos de aprendizaje específicos para estos dispositivos.

Los problemas de latencia, consumo de energía y ancho de banda están relacionados a las arquitecturas que permiten entrenar métodos de IA en dispositivos de borde y propagar la información a la nube, para tener una predicción global y final de manera distribuida (Berta et al., 2024). Estos escenarios implican utilizar módulos System on chip (SoC), como el ESP32, con capacidades de inteligencia artificial (Chang et al., 2025). En otros casos, el problema de implementar inteligencia artificial en dispositivos de borde es la complejidad de los algoritmos de análisis de datos y su implementación en microcontroladores, como en el caso de las aplicaciones de reconocimiento de actividades (Hayajneh et al., 2024) o detección de infarto (Gragnaniello et al., 2024).

2.3 Bases teóricas

2.3.1 Sistema de computación de borde

Un sistema de computación de borde es una tecnología que permite procesar datos en el lugar más cercano de su generación y evitar el envío de datos a dispositivos externos. Esto se utiliza por el incremento de datos producidos por dispositivos conectados y a la necesidad de reducir la latencia, el consumo de energía y el uso del ancho de banda (Qadri et al., 2020). En el caso de los sistemas de aplicación de tiempo crítico, como los de alerta de accidentes relacionados a la salud, requieren respuestas en tiempo real y baja latencia (Hameurlaine et al., 2025) (Ianculescu et al., 2025).

En el contexto del IoT, se considera que cuando se habla de computación de borde, este corresponde al nivel más cercano al mundo físico el cual incluye sensores, actuadores, gateways, concentradores y dispositivos llamados nodos de borde (Edge Node) (Amin & Hossain, 2021). Cuando se integran algoritmos de inteligencia artificial, para tomar decisiones a partir de los datos que procesan, se impulsa el desarrollo de aplicaciones de computación de Borde (Figura 1).

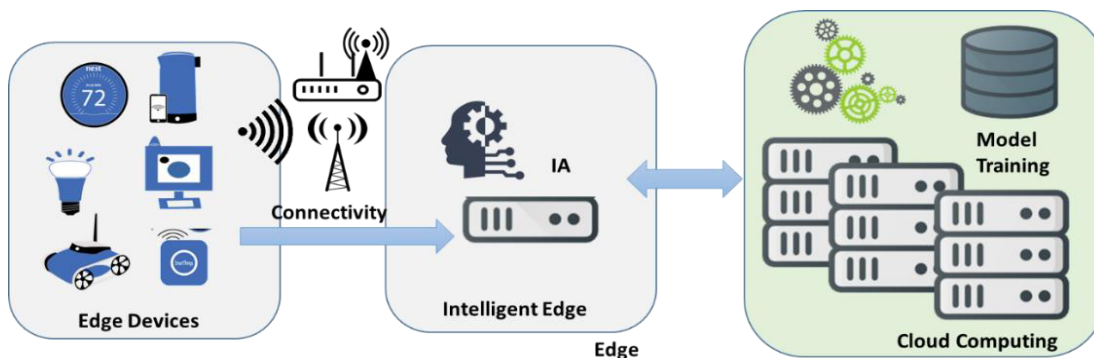


Figura 1. Arquitectura de borde para dispositivos embebidos inteligentes

Entre las ventajas de la computación de borde se tiene: proximidad (Los dispositivos se sitúan cerca de los equipos generadores de datos), localidad (Funciona de forma aislada al resto de la red), reducción de latencia (disminuyen los retardos en la red), ahorro de ancho de banda (Se procesan los datos de forma local para luego transmitir una versión reducida de estos) e inmediatez (Permite analizar los datos de cualquier dispositivo IoT casi en tiempo real) (Di Leo et al., 2025). Los pilares de la computación de borde (Mansouri & Babar, 2021) representan las razones por las que un dispositivo necesita inteligencia y procesar lo más cerca de donde se produce el dato (Figura 2) :

- **Autonomía.** En algunos casos la autonomía es necesaria por razones de seguridad, o por limitaciones en las telecomunicaciones, otras veces sencillamente el dispositivo no requiere conectarse.
- **Privacidad.** La protección de datos personales recolectados por un dispositivo limita mucho su utilidad debido a la necesidad de protección en diferentes entornos.
- **Alta disponibilidad.** Cualquier dispositivo que requiera de alta disponibilidad necesitará también la capacidad de tomar sus propias decisiones. La alta disponibilidad estaría limitada por cualquiera de los pilares anteriores.

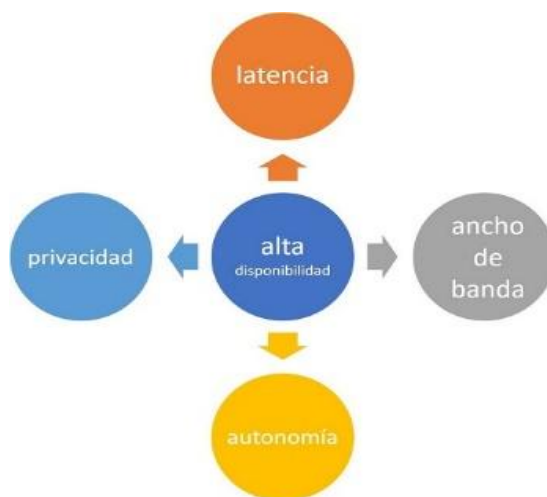


Figura 2. Pilares del Edge Computing (Mansouri & Babar, 2021)

En el caso que se requiera realizar tareas intensivas de procesamiento y clasificación, una opción es enviar los datos a servidores en nube, usando algoritmos avanzados de aprendizaje automático para análisis más complejos, integración de una etapa de borde para análisis de datos críticos de salud en centros médicos y conexión con sistemas en nube para acceso remoto (Suleiman et al., 2025) (Figura 3).

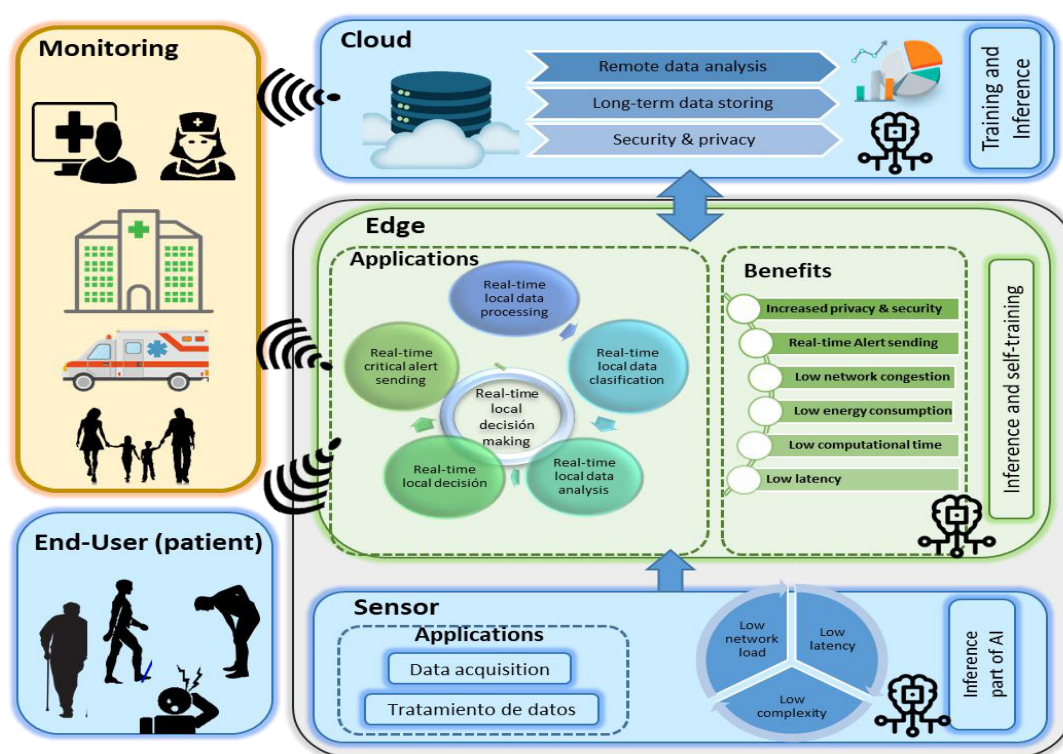


Figura 3. Modelo del Sistema (Hayyolalam et al., 2021)

La computación de borde se puede complementar con la tecnología en nube (Cloud) donde la primera se utiliza cuando la velocidad a la hora de transferir datos es un requisito fundamental, mientras que la tecnología en nube permite analizar grandes cantidades de datos de forma simultánea (J. Cao et al., 2018). En este contexto, existen varias herramientas de computación en nube para el monitoreo, almacenamiento, análisis de datos provenientes y el control remoto de dispositivos. Entre estas herramientas se tienen las plataformas como servicio (PaaS) e infraestructura como servicio (IaaS) que consideran servicios como Amazon web services (AWS IoT), Microsoft Azure IoT Hub y Google Cloud IoT (Andriulo et al., 2024) (Mocha Zhingri & Paltín Guamanrrigra, 2022). Además, existen herramientas para el desarrollo de soluciones con poca complejidad, denominadas Software como servicio (SaaS) y se encuentran tecnologías como Ubidots, ThingSpeak, Blynk, Adafruit IO y Losant, para la visualización de datos en tiempo real y uso de paneles de control (Eneque Suarez, 2025).

En esta investigación, para el monitoreo de datos se utiliza los dispositivos de borde en combinación con aprendizaje automático, en escenarios donde se requiere usar sensores y tecnología IoT para procesar información de manera local. Este concepto es fundamental en el desarrollo de la tesis.

2.3.2 Nodos Sensores

Es un componente inteligente el cual es usualmente un microcontrolador (MCU) para la ejecución de las tareas y permite optimizar el consumo de energía modificando su velocidad de trabajo (C. Zhang et al., 2022). La arquitectura del nodo sensor se puede dividir en: computación, energía, detección y comunicación. Además, los nodos sensores son dispositivos que se utilizan en aplicaciones denominadas redes de sensores inalámbricas (WSN) para el monitoreo y control (Duobiene et al., 2022).

En esta investigación se utiliza el diseño de los nodos sensores debido a que se requiere optimizar un bajo consumo de corriente eléctrica en su construcción. A diferencia de otros diseños de hardware con sensores, donde no es importante las restricciones de energía.

2.3.3 Internet de las cosas.

El Internet de las cosas (IoT) describe a los dispositivos inteligentes que monitorean, comunican e interpretan información de su entorno en tiempo real permitiendo la recopilación de datos. Actualmente, el IoT se compone de un conjunto heterogéneo de diferentes tipos de redes usado para distintos fines (Rivera et al., 2024) donde una de sus principales aplicaciones del IoT se encuentra en el campo de la salud (H-IoT) (M. Kumar et al., 2023).

El IoT se utiliza en esta investigación debido a que permite monitorear señales fisiológicas de pacientes de forma continua y con capacidades de comunicación inalámbrica. Esta elección tecnológica se justifica por su bajo consumo de corriente eléctrica, disminuye la latencia y limita el volumen de datos transmitidos, lo cual es adecuado para aplicaciones de monitoreo continuo de señales fisiológicas relacionadas a cardiopatía en salud.

La tecnología IoT aplicada a la salud se pueden organizar en tres niveles: dispositivos (responsable de recoger las propiedades físicas de los objetos), redes (enviar los datos recogidos por la capa de percepción) y plataformas (Armando et al., 2023) . En la Recomendación UIT-T Y.4051 (Itu-t, 2019) se presenta en términos generales Internet de los objetos (IoT), donde se aclara el concepto y su alcance (Figura 4). Además, existe un área de aplicación llamada “Ambient assisted living” (AAL), donde se usa el aprendizaje automático para escenarios de atención médica basados en IoT (Amin & Hossain, 2021), la monitorización de la salud de personas y mejorar la calidad de vida humana.(Figura 5).

Además, la tecnología IoT, se utiliza para integrar dispositivos embebidos inteligentes priorizando el procesamiento local en el dispositivo de borde y usando la red de Internet como infraestructura de interconexión para el envío de información. Es por ello, que el uso de esta tecnología es coherente con los objetivos de la tesis, orientados al monitoreo continuo de señales fisiológicas mediante sistemas embebidos de recursos limitados.

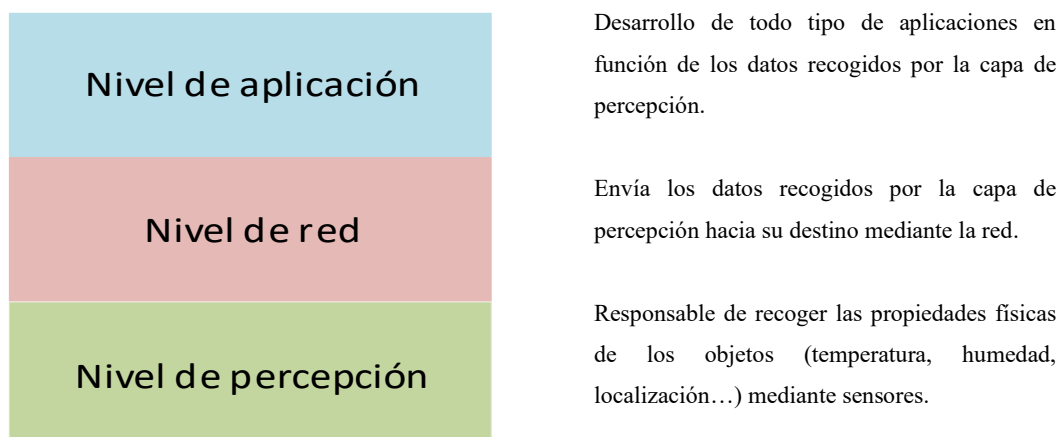


Figura 4. Nivel del modelo IoT

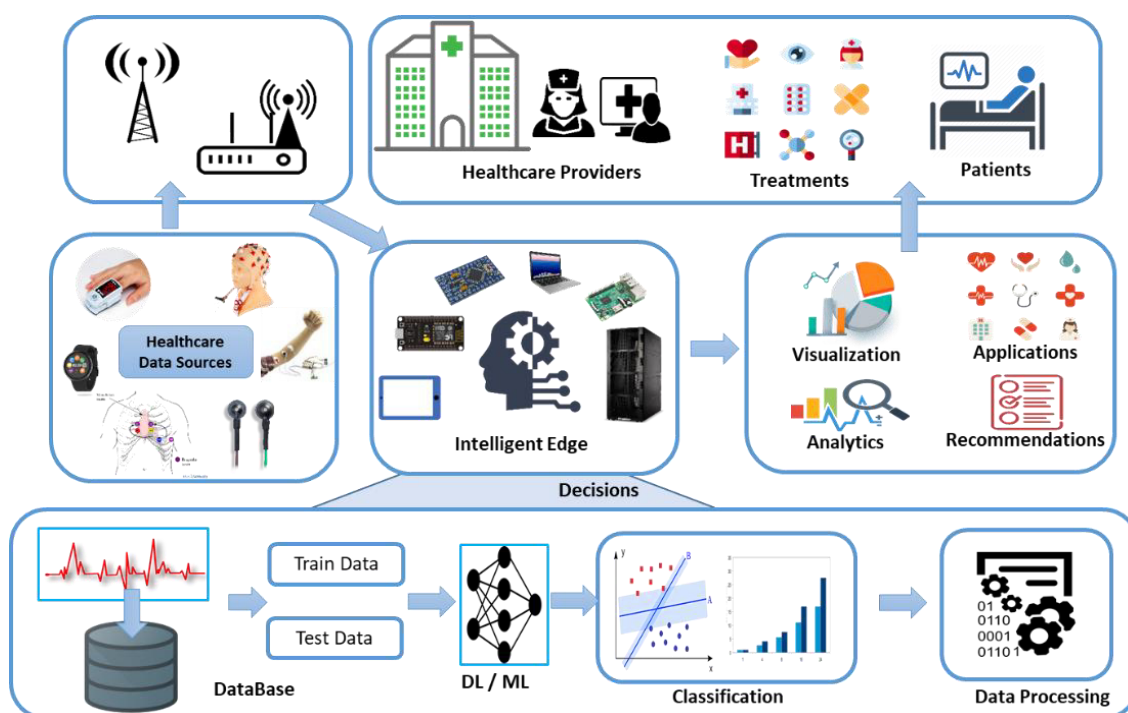


Figura 5. Inteligencia cognitiva de borde para clasificación de señales médicas

2.3.4 Aprendizaje automático

El aprendizaje automático brinda capacidades como el procesamiento del lenguaje natural, la percepción y la toma de decisiones basada en el conocimiento a sistemas computacionales y se implementa en escenarios donde se generan las reglas en base a la experiencia que actúan sobre nuevos datos para dar una respuesta (Bobadilla, 2021). Se puede clasificar estos algoritmos en tres clases: supervisado, no supervisado y

aprendizaje reforzado (RL). En el caso del aprendizaje profundo (DL), este es una subclase de ML que intenta imitar el proceso de toma de decisiones del cerebro humano mediante la implementación de redes neuronales.

A continuación, se describen los aspectos relacionados a los algoritmos de aprendizaje automático utilizados en la mayoría de las soluciones que implementan dispositivos de Borde, aprendizaje automático reducido (TinyML) y herramientas para su implementación (Iodice, 2023):

a) Aprendizaje automático utilizados en dispositivos embebidos. El aprendizaje automático está teniendo un impacto revolucionario en todas las áreas científicas, realizando la detección y predicción de trastornos que requerían pruebas médicas complejas (Qadri et al., 2020) mediante modelos entrenados. Los modelos son una parte del panorama general debido hay que considerar su ejecución en el hardware que permite que estos se ejecuten de manera eficiente. Para esto surgen marcos de trabajo (Frameworks) para su implementación en dispositivos de hardware como TensorFlow, TensorFlow Lite y TensorFlow Lite Micro (Banerjee, 2023) (Warden & Situnayake, 2019). Entre algunas tipos de algoritmos de aprendizaje automático se tienen:

- **Máquinas de vectores de soporte.** Desarrollada por Vladimir Vapnik y Corinna Cortes a principios de la década de 1990 en Bell Labs (Cortes & Vapnik, 1995). El algoritmo de SVM intenta obtener la separación máxima de dos clases utilizando un número mínimo de puntos de datos (Joshi, 2020) ,también llamados vectores de soporte, mediante la utilización de un hiperplano. Las SVM trabajan en escenarios lineales como no lineales y utilizan algoritmos de aprendizaje supervisado relacionados con problemas de clasificación y regresión (Roy & Chakraborty, 2023).
- **Redes neuronales.** Una red neuronal estándar consta de muchos procesadores simples conectados llamados neuronas, cada uno de los cuales produce una secuencia de activaciones. Una forma de utilizar las redes neuronales es mediante el aprendizaje profundo, el cual incluye métodos como las redes neuronales convolucionales (CNN) o redes neuronales profundas (DNN) y sus variaciones (Taye, 2023).
- **Métodos de reducción de componentes.** Unos de los métodos de transformación más conocidos es el PCA (análisis de componentes principales) consiste en la

reducción de dimensionalidad mediante la descomposición de valores de datos para proyectarlos sobre un espacio de dimensiones. PCA pertenece a la familia de algoritmos conocida como aprendizaje no supervisado siendo su objetivo extraer información empleando los predictores (Pernice, 2023).

- **K-Vecinos cercanos (KNN - K-Nearest-Neighbor).** Es un algoritmo supervisado que se usa para clasificar nuevas muestras (valores discretos) o para predecir nuevos valores. Sirve para clasificar valores buscando los puntos de datos similares que se aprendió en la sección de entrenamiento (Liquet et al., 2024).
- **Clustering KMeans.** K-Means es un algoritmo no supervisado de clustering utiliza cuyo objetivo es el de encontrar “K” grupos (clústers) entre los datos. KMeans tiene como función optimizar (minimizar) la suma de las distancias cuadráticas de cada objeto al centroide de su cluster (Skiena, 2017).

b) Tiny Machine Learning (TinyML). Tiene como propósito integrar tecnologías basadas en Aprendizaje automático (Machine Learning ML) dentro de dispositivos de hardware como los microcontroladores (MCU) (Banerjee, 2023) (Warden & Situnayake, 2019). Esto permite desarrollar aplicaciones y servicios que no necesitan conectarse a la nube, debido a que esto implica el consumo de energía y genera riesgos de seguridad y privacidad de los datos (Sanchez-Iborra & Skarmeta, 2020). TinyML puede proporcionar una solución enfocada en el análisis de los datos en el borde en dispositivos integrados de bajo consumo, proporcionando estadísticas resumidas inteligentes que toman en cuenta patrones y anomalías, siendo esto posible por la dualidad de sistemas embebidos y aprendizaje automático. Es por esto, que la intersección de las tecnologías de machine learning y sistemas embebidos involucra retos para habilitar algoritmos de complejidad reducida en dispositivos embebidos (Figura 6).

En esta investigación se utiliza el enfoque TinyML debido a que, no se requiere grandes recursos de memoria o velocidad de procesamiento y esto lo hace conveniente para su ejecución directa en microcontroladores. Además, esta tecnología permite la evaluación comparativa de distintos modelos en función de métricas de comportamiento eficiencia. Como parte de este enfoque, en la investigación se elige usar TensorFlow Lite Micro para el despliegue de modelos en base a redes neuronales y el software Micromlgen para implementar los modelos basados en SVM, RF y DT.

La principal limitación de TinyML es la potencia de procesamiento, memoria y frecuencia de reloj. Usualmente los modelos TinyML se ejecutan en microcontroladores de 32 bits con menos de 500 kB de memoria RAM y frecuencia de reloj menor a 200 MHz. Además TinyML se utiliza para la inferencia ML mientras el entrenamiento se ejecuta fuera del dispositivo (Koufos et al., 2021).

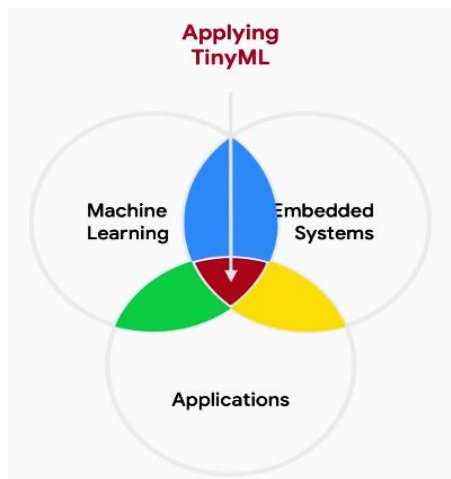


Figura 6. Intersección de tecnologías para aplicaciones de TinyML

Algunos de los Framework para implementar TinyML se muestran en la Tabla 1 mostrando que es posible integrar algoritmos como Redes neuronales (NN), Máquina de vectores de soporte (SVM), árboles de decisión o Bosque aleatorio (RF) en dispositivos con recursos de hardware reducidos. Por ejemplo, algunos frameworks compatibles con plataformas para Arduino son MicroMLGen, sklearnporter y emlearn (Abu-Samah et al., 2025). El flujo de trabajo seguido por estas herramientas es similar en todas ellas: en primer lugar, se entrena un algoritmo seleccionado utilizando una biblioteca de Aprendizaje automático basada en Python; luego el modelo generado es transformado a código C por la correspondiente herramienta de traducción TinyML. Finalmente, el código producido se integra dentro de un algoritmo importándolo como una biblioteca externa.

Tabla 1 Frameworks TinyML (Sanchez-Iborra & Skarmeta, 2020)

Framework	Algorithms	Compatible Platforms	Output Languages	Interoperable External	Publicly Available	Main Developer
-----------	------------	----------------------	------------------	------------------------	--------------------	----------------

				Libraries		
TensorFlow Lite	Neural networks	ARM Cortex-M	C++ 11	TensorFlow	Yes	Google
ARM-NN	Neural networks	ARM Cortex-A. ARM Mali. Graphics. Processors. ARM Ethos Processor	C	TensorFlow, Caffe, ONNX	Yes	ARM
STM 32Cube.	Neural networks	STM32	C	Keras. TensorFlow. Lite Caffe. ConvNetJs.	Yes	STMicroelectronics
MicroMLGen	SVM. RVM	Arduino. ESP32. ESP8266	C	Scikit-learn	Yes	Particular developer
sklearnporter	SVM Decision tree Random Forest Ada Boost Classifier k-NN Naive Bayes Neural networks	Multiple constrained & nonconstrained platforms	C GO Java JavaScript PHP Ruby	Scikit-learn	Yes	Particular developer
emlearn Decision trees	Decision trees Neural networks Naive Gaussian Bayes Random forest	AVR Atmega. ESP8266. Linux	C	Keras Scikit-learn	Yes	Particular developer
uTensor	Neural networks	mBed boards	C++11	TensorFlow	Yes	Particular developer
TinyMLgen	Neural networks	ARM Cortex-M ESP32	C	TensorFlow Lite	Yes	Particular developer

En la intersección de las tecnologías de aprendizaje automático y los sistemas integrados, se habilitan algoritmos de complejidad reducida, generando soluciones TinyML para analizar datos, identificando patrones y anomalías (Warden & Situnayake, 2019). Esto permite que los dispositivos creen nuevas aplicaciones y servicios (Tunggadewi et al., 2020), trasladando la inteligencia centralizada (sofisticada y menos compleja) a los bordes, siendo más autónomos y escalables (El Ghmary et al., 2020) (Figura 7). Además, es necesario considerar que existen beneficios en términos de latencia, seguridad y uso de energía y memoria

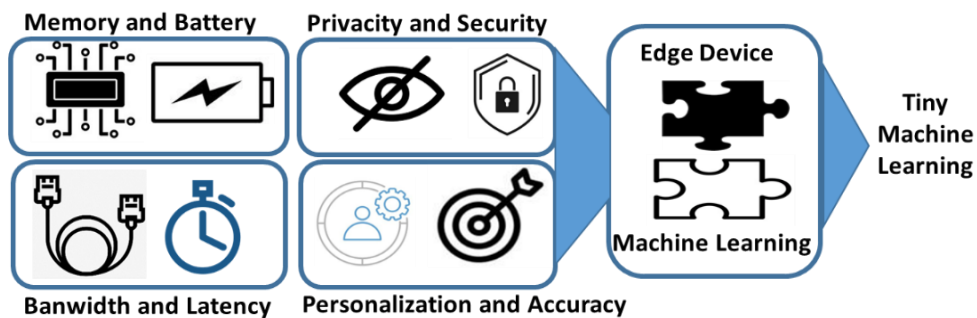


Figura 7. Consideraciones para implementación de TinyML en sistemas embebidos

c) Herramientas para implementación de aprendizaje automático embebido.

- **TensorFlow.** Es una biblioteca de software de código abierto usada para el aprendizaje automático utilizando el lenguaje Python para entrenar modelos de aprendizaje automático. Se pueden implementar en infraestructuras en la nube, dispositivos computacionales locales o mediante servicios que puede brindar una interfaz REST (Ansari, 2023). Las variantes de TensorFlow son:
 - **TensorFlow Lite.** Optimizado para sistemas más pequeños como Android, iOS y sistemas integrados que ejecutan una variante de Linux, como Raspberry Pi. TensorFlow Lite también incluye un conjunto de herramientas que lo ayudan a convertir y optimizar su modelo para este tiempo de ejecución (Warden & Situnayake, 2019).
 - **TensorFlow Lite Micro.** Construido sobre TensorFlow Lite y se puede usar para reducir el tamaño de modelo aún más para que funcione en microcontroladores y es una tecnología de habilitación central para TinyML (David et al., 2020). En el caso de esta investigación se utiliza esta herramienta para la creación de modelos usando redes neuronales.
 - **TensorFlow.js.** Proporciona una biblioteca basada en JavaScript que se puede utilizar tanto para entrenar modelos como para ejecutar inferencias en entornos basados en JavaScript, como los navegadores web o Node.js. <https://www.tensorflow.org/js>.

Entre algunos frameworks relacionados a la implementación de TensorFlow para creación de modelos tenemos a: Scikit-Learn, Pytorch, Caffè, theano (Joshi, 2020). En el caso de TensorFlow Lite, esta herramienta convierte un modelo grande y

robusto en algo ligero y pequeño, mediante un archivo llamado. tflite, el cual puede implementado en diferentes tipos de dispositivos. Estas tecnologías permiten integrar estos algoritmos desde grandes procesadores, teléfonos inteligentes, dispositivo de hardware reducidos (Dokic et al., 2020). En este último caso, sólo se centran en la inferencia, utilizando librerías de software de TensorFlow Lite (Figura 8), mientras que TensorFlow llega hasta la etapa de entrenamiento del modelo (Situnayake, 2020).

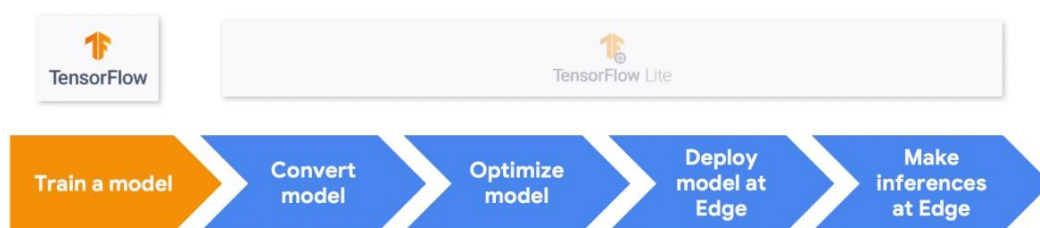


Figura 8. Tareas delegadas a TensorFlow y TensorFlow Lite (Situnayake, 2020).

TensorFlow Lite para microcontroladores requiere una plataforma de 32 bits y son compatibles con muchos procesadores basados en la arquitectura de la serie ARM Cortex-M y ESP32 (A. K. Gupta & Nandyala, 2023) entre las cuales se tiene:

- Arduino Nano 33 BLE Sense
- SparkFun Edge
- Kit de desarrollo STM32F746
- EdgeBadge de Adafruit
- Kit TensorFlow Lite de Adafruit para microcontroladores
- Circuit Playground Bluefruit de Adafruit
- ESP32-DevKitC de Espressif
- ESP-EYE de Espressif

Además, los avances recientes en los algoritmos de aprendizaje automático que se ejecutan en el borde y la nube, no se aplican a tinyML, donde el tamaño de las redes neuronales debe ser limitado y con modelos, probablemente, demasiados grandes para almacenarlo en sistemas embebidos. TF Lite Micro permite comprimir modelos de aprendizaje automático previamente entrenados generados en frameworks comunes, a un tamaño adecuado para TinyML lo cual es utilizado

en esta investigación para el despliegue de redes neuronales (Iodice, 2023)(Koufos et al., 2021).

- **Micromlgen.** Transforma los algoritmos de aprendizaje automático (como máquina de vector soporte (SVM), árboles de decisión (DT), bosques aleatorios (RF) o XGBOOST) entrenados en python con scikit-learn a código C que puede ejecutarse en una serie de microcontroladores como Arduino, ESP8266, ESP32 entre otros (Abu-Samah et al., 2025) (Sanchez-Iborra & Skarmeta, 2020). Surgió como una alternativa a Tensorflow para microcontroladores, para ejecutar inferencias incluso en microcontroladores de 8 bits. (Gloria & Sebastiao, 2021) (Luque Martínez & Bermejo Sánchez, 2020). Esta herramienta se usa para implementar los modelos basados en SVM, RF y DT en la presente investigación. Otros paquetes para implementar y entrenar algoritmos de aprendizaje en sistemas embebidos con bajos recursos de hardware se tiene:
 - **EloquentMicroML.** Realiza el entrenamiento, clasificación y la predicción en el mismo dispositivo embebido. Es posible su implementación en el dispositivo de hardware Arduino.
 - **EloquentTinyML.** Una librería que se basa en Tiny Tensorflow para entrenar modelos e implementarlos en sistemas de hardware basados en Arduino.
 - **Tinymlgen.** Es una herramienta que convierte el modelo de Tensorflow Lite a código en lenguaje C. Se complementa directamente con la herramienta EloquentTinyML (Salerno, 2021).
- **Python.** Es un lenguaje de "alto nivel" que se interpreta en tiempo de ejecución y orientada a objetos, caracterizado por su sintaxis clara y que ofrece clases de lectura e interfaces para llamadas al sistema operativo y bibliotecas de análisis de datos, procesamiento de señales e inteligencia artificial (Lutz, 2025). Puede ser extendido, usando el lenguaje C o C++ (Haslwanter, 2016) para incorporar nuevas características mediante la creación de librerías mediante la librería de Python “Jinja”, que permite acceder a un conjunto de plantillas para generar código de forma automática en otros lenguajes de programación y es usada por la librería “Micromlgen” (Ashley, 2020) (Pallets Project, 2021). Además, Python se complementa con bibliotecas de aprendizaje automático como Scikit-Learn (Abu-Samah et al., 2025), de tratamiento de matrices, gráficas y de procesamiento de

señales entre las cuales tenemos: NumPy (librería de matriz n-dimensional base), Pandas (estructura de datos y análisis), SciPy (librería fundamental para la informática científica), Matplotlib (Generación de gráficas) y SymPy (matemática simbólica).

- **Plataforma Arduino.** Es una plataforma electrónica de código abierto basada en hardware y software que diseña y distribuye un conjunto de placas de desarrollo para microcontroladores que simplifican la implementación de hardware integrado junto con un conjunto de herramientas que encapsulan la complejidad de la configuración de los dispositivos de hardware, para enfocarse en la aplicación (Ünsalan et al., 2025). Arduino IDE es un entorno de desarrollo liviano con características que permiten manipular rápidamente las placas de desarrollo de microcontroladores (Warden & Situnayake, 2019). Una de las principales ventajas que ofrece el ecosistema Arduino es:
 - La portabilidad del código que se escribe para tarjetas de hardware de familias similares o de fabricantes de terceros.
 - Disponibilidad de bibliotecas para realizar diversas tareas, como interactuar con un módulo sensor o manipular datos utilizando algoritmos comunes.
 - El "núcleo" (core) de Arduino es compatible con familias de dispositivos compatibles.
- **Dispositivo System On Chip ESP32.** El ESP32 es un dispositivo de hardware que incluye comunicación Bluetooth y Bluetooth Low Energy (BLE) y es adecuado para aplicaciones que requieren un procesamiento de datos intenso con conectividad Wi-Fi o Bluetooth (Zulfiqar, 2024) (Figura 9). En el caso particular de la placa de desarrollo ESP32 DEVKIT DOIT, se puede energizar con 3,3 voltios a través de una conexión micro-USB de 5 voltios. Dispone de dos núcleos (Tensilica Xtensa LX6 de 32 bits con hasta 240 MHz/600 MIPS de frecuencia de reloj) que se gestionan de forma independiente, 448 KB de ROM, 520 KB de SRAM y 16 KB de memoria RTC (Ozan, 2021). Entre los módulos que posee se tiene cuatro temporizadores independientes, un convertidor de digital a analógico (DAC), conversor Analógico/Digital (ADC), sensores táctiles capacitivos y un sensor de efecto Hall. Por otro lado, existe una tarjeta de desarrollo llamada ESP32-CAM fabricado por la empresa Ai Thinker (aunque existen otros similares) basado en un SoC ESP32-S de Espressif con 520 KB de memoria interna y 4 MB

externos y que incluye una cámara OV2640 de 2M píxeles y una micro-SD (Secure Ranura para tarjeta digital) (Cameron, 2021).



Figura 9. ESP32-CAM (Cameron, 2021)

d) Métodos de validación. Permiten medir la calidad del rendimiento de los modelos y garantizar que sean capaces de generalizar a datos no vistos. Cada método de evaluación se complementa con diferentes métricas según el enfoque y el propósito donde su elección depende del tipo de problema y de los datos específicos con los que se trabaje. Entre algunas de ellas se tienen:

- **Matriz de confusión.** Es una herramienta para calcular métricas de evaluación en modelos de clasificación como la precisión, recall, F1-score; y permite visualizar el rendimiento del modelo comparando las predicciones del modelo con las etiquetas reales (A. K. Gupta & Nandyala, 2023). La matriz de confusión se divide en cuatro cuadrantes:
 - **verdaderos positivos (TP).** Representa el número de muestras que fueron correctamente clasificadas como positivas por el modelo.
 - **falsos positivos (FP).** Indica el número de muestras que fueron incorrectamente clasificadas como positivas por el modelo
 - **verdaderos negativos (TN).** Muestra la cantidad de muestras que fueron correctamente clasificadas como negativas
 - **falsos negativos (FN).** Refleja el número de muestras que fueron incorrectamente clasificadas como negativas por el modelo.

- **Métricas de validación.** Son medidas utilizadas para evaluar el rendimiento y la calidad de un modelo en función de su capacidad para hacer predicciones precisas en datos no vistos, es decir, en datos que no se utilizaron durante el entrenamiento del modelo (Unpingco, 2019) (Joshi, 2020). Algunas de ellas son:
 - **Exactitud (Accuracy):** Es la proporción de predicciones correctas sobre el total de predicciones realizadas $((TP + TN) / (TP + TN + FP + FN))$. Es útil en problemas de clasificación balanceada, donde todas las clases tienen una cantidad similar de muestras.
 - **Precisión:** Mide la proporción de verdaderos positivos (instancias correctamente clasificadas como positivas) sobre el total de instancias clasificadas como positivas $(TP / (TP + FP))$. Es importante en problemas donde los falsos positivos son costosos, como en diagnósticos médicos.
 - **Recuperación (Recall o Sensibilidad):** Calcula la proporción de verdaderos positivos sobre el total de instancias que realmente son positivas $(TP / (TP + FN))$.
 - **Especificidad (Specificity):** Proporción de verdaderos negativos entre todas las instancias que son realmente negativas $(TN / (TN + FP))$.
 - **F1-Score:** Es la media armónica entre precisión y recuperación. Es útil cuando hay un equilibrio entre precisión y recuperación que se busca. Ayuda a tener una visión más completa del rendimiento del modelo.
 - **Área bajo la Curva ROC (AUC-ROC):** Es útil para evaluar la capacidad del modelo para distinguir entre clases en problemas de clasificación binaria. Cuanto mayor sea el AUC-ROC, mejor será el modelo para distinguir entre clases.
 - **Error cuadrado medio (MSE - Mean Squared Error):** Comúnmente utilizado en problemas de regresión, mide el promedio de los errores al cuadrado entre las predicciones y los valores reales.
 - **Error absoluto medio (MAE - Mean Absolute Error):** Similar al MSE pero sin elevar los errores al cuadrado. Es más robusto a valores atípicos y da una idea directa de la magnitud del error.
 - **R-cuadrado (R^2):** En problemas de regresión, mide la proporción de la variabilidad de la variable objetivo que es capturada por el modelo. Un valor más cercano a 1 indica un buen ajuste.

- **Método de Validación cruzada.** En el caso de los métodos de evaluación, una de las más importante es la validación cruzada. Esta es crucial en la evaluación de modelos de aprendizaje automático, siendo un enfoque para obtener métricas más confiables y estimar el rendimiento del modelo en datos no vistos y para evitar el sobreajuste (overfitting) del modelo a los datos de entrenamiento (Joshi, 2020) (Majumdar, 2023). Implica dividir el conjunto de datos en múltiples subconjuntos (llamados "folds") y luego entrenar y evaluar el modelo varias veces, utilizando diferentes combinaciones de subconjuntos como datos de entrenamiento y validación, obteniendo múltiples métricas de evaluación que permiten obtener una idea más precisa del rendimiento del modelo (Russell & Norvig, 2020).

En esta investigación, los modelos de aprendizaje automático se entrenan y validan mediante validación cruzada, exactitud y el F1-score porque permite evaluar el modelo de forma más consistente que otros métodos como la división simple de datos (hold-out), la cual depende de una sola partición. A diferencia de la validación en serie o la validación por grupos, la validación cruzada aprovecha mejor conjuntos de datos pequeños como un proceso estadístico y comparativo que aporta a identificar el algoritmo que mantiene estabilidad en diferentes particiones antes de ser integrado en hardware embebido.

e) Metodologías para el desarrollo de modelos de aprendizaje automático. Existen metodologías que proporcionan estructuras y pasos que los científicos de datos e ingenieros pueden seguir para crear y mejorar modelos de manera efectiva. Entre algunas de ellas se tienen: CRISP-DM (Cross-Industry Standard Process for Data Mining), MLOps (Machine Learning Operations) (Pote, 2023) y DataOps (Atwal, 2020). En el caso de CRISP-DM, se ha diseñado para minería de datos y es utilizada ampliamente en el desarrollo de algoritmos de aprendizaje automático en un entorno empresarial, brinda un enfoque iterativo para fases como la comprensión del negocio, de los datos, la preparación de los datos, la modelización, la evaluación y el despliegue (Huber et al., 2019) (Purbasari et al., 2021). Consta de seis fases (Figura 10):

- Comprensión del negocio: Comprender los objetivos del proyecto, los requisitos y las restricciones comerciales.
- Comprensión de los datos: Explorar y evaluar los datos disponibles, identificar problemas y oportunidades.

- Preparación de los datos: Seleccionar, limpiar y transformar los datos para que sean adecuados para el análisis.
- Modelado: Seleccionar y aplicar algoritmos de aprendizaje automático, ajustar parámetros y entrenar modelos.
- Evaluación: Evaluar el rendimiento de los modelos en datos no vistos y determinar si cumplen los objetivos del proyecto.
- Despliegue: Implementar los modelos y monitorear su rendimiento continuo.

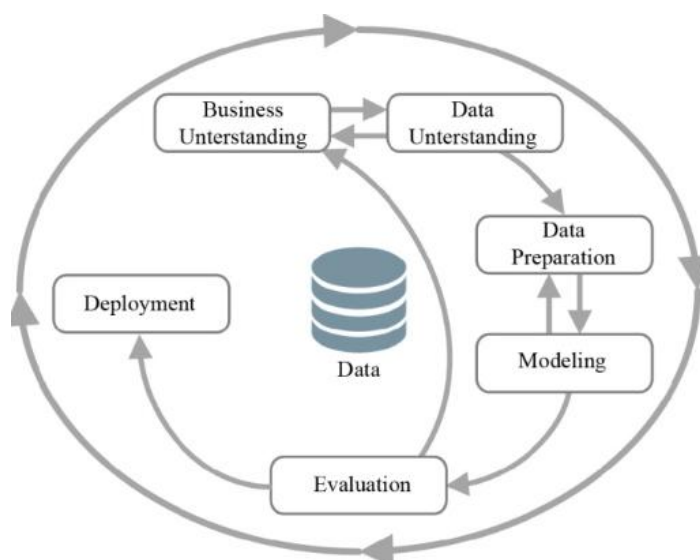


Figura 10. Procesos de CRISP-DM (Huber et al., 2019)

Aunque CRISP-DM es la metodología más utilizada, es importante tener en cuenta que el campo de la ciencia de datos es diverso y en constante evolución. Dependiendo de los objetivos del proyecto y del equipo involucrado, es posible que otras metodologías, como MLOps o KDD sean necesarios (Ninu, 2023)(Głowania et al., 2023). Esta investigación adapta la estructura CRISP-DM, al contexto de dispositivos embebidos, donde el despliegue en hardware representa la fase final del proceso, ya que permite medir variables físicas como latencia y consumo corriente eléctrica.

2.3.5 Señales fisiológicas

Son señales que permiten realizar análisis clínicos para diagnóstico e investigación médica utilizando información adquirida del sistema biológico de estudio. Entre los tipos de variables fisiológicas tenemos tanto Imágenes en 2D y 3D como señales

unidimensionales (presión, ECG, temperatura, movimiento humano, etc) (Arun et al., 2025) (Pramanik et al., 2018).

En esta investigación, las señales fisiológicas se abordan desde un enfoque de análisis computacional, usando datos para evaluar modelos de aprendizaje automático, y no con fines diagnósticos o clínicos.

Las señales fisiológicas o biomédicas pueden clasificarse usualmente según: Las propiedades de la señal, tipo de fuente y aplicación biomédica. Además estas se adquieren mediante sensores biomédicos que sirven como una interfaz entre los sistemas biológicos y electrónicos, adquiridas mediante métodos de bioinstrumentación, cuyos conceptos describimos a continuación.

a) Bioinstrumentación. El uso de sistemas de medición basados en sensores biomédicos hace referencia al término de Bioinstrumentación como se describe en (Tian & Jia, 2023)(Arun et al., 2025). La mayoría de los sistemas de instrumentación contienen elementos comunes, que están presentes independientemente del parámetro que se mida (Figura 11). Un sistema típico mide un parámetro de interés y genera una salida para que el usuario comprenda la información. Después de que se detecta un parámetro de entrada, un procesador lo modifica mediante la amplificación de la señal, el filtrado para eliminar información no deseada o la comparación con otras señales.

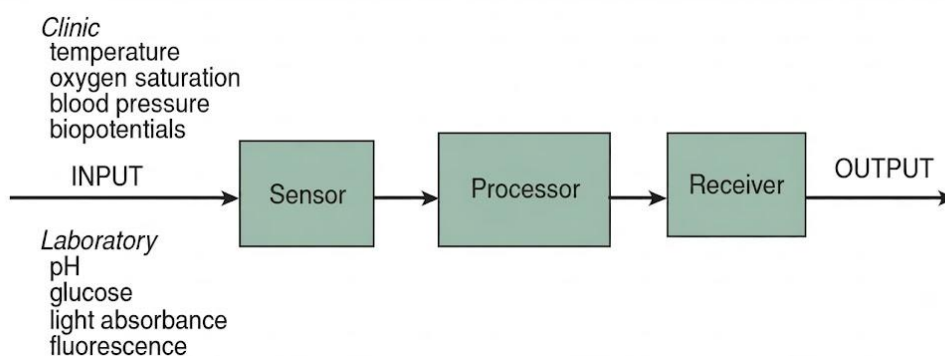


Figura 11. Esquema de un sistema de medición típico (Saltzman, 2010)

En el caso de señales como el pulso y la frecuencia cardíaca (FC), estas sufren variaciones normales desde recién nacido hasta la senectud. La fiebre es un signo importante de enfermedad en la edad avanzada y muchas veces es el único síntoma de enfermedad (Alcázar-Aguilar et al., 2021). Cuando hay aumento de la temperatura ambiental y corporal, se acelera la frecuencia respiratoria (OMS & OPS, 2020).

Además, existen sensores disponibles para medir muchos parámetros de interés clínico y de laboratorio. Algunos tipos de sensores se resumen en la Tabla 2.

Tabla 2. Sensores utilizados en instrumentos biomédicos (Saltzman, 2010)

Tipo de sensor	Elemento de adquisición	Ejemplo
Térmica	Termopar	Termómetro electrónico
Mecánica	extensómetro, sensor piezoeléctrico	Transductor de presión
Eléctrica	Electrodo	Electrocardiógrafo (ECG), electroencefalógrafo (EEG)
Químico	Electrodo	Medidor de pH
Óptico	Fotodiodo, fotomultiplicador	Oxímetro de pulso

Algunos parámetros fisiológicos se miden de forma rutinaria en los consultorios médicos, emergencias habitaciones, clínicas para pacientes ambulatorios y hospitales. La Tabla 3 enumera algunos parámetros fisiológicos comunes y los instrumentos utilizados para medirlos.

Tabla 3 Instrumentos médicos típicos (Saltzman, 2010)

Entrada	Instrumento	Sensor	Salida	Rango
Temperatura	Termómetro digital oral	Termistor	Visualización de temperatura	32–40°C
Presión arterial	Esfigmomanómetro digital	Estetoscopio o extensómetro	Presión	0–400 mmHg
Oxígeno en sangre	Oxímetro de pulso	Fotodiodo	Porcentaje de saturación de oxígeno	0–100% SpO ₂
Biopotenciales	Electrocardiógrafo (ECG)	Electrodos cutáneos	Electrocardiograma	0.5–5 mV
Biopotenciales cardíacos	Electroencefalógrafo (EEG)	Electrodos del cuero cabelludo	Electroencefalograma	5–300 mV
Biopotenciales neuronales	Electrorretinógrafo (ERG)	Electrodos de lentes de contacto	Electrorretinograma	0–900 mV
Biopotenciales retinianos	Electromiógrafo (EMG)	Electrodos de aguja	Electromiograma	0.1–5 mV
Biopotenciales musculares				

b) Señales Bioeléctricas. Este grupo de señales se propagan por el cuerpo, y es necesario un método invasivo para adquirirlas, siendo propias de sistemas biológicos,

donde el potencial transmembrana, ante ciertas condiciones, genera una diferencia de potencial, que puede ser medida con microelectrodos para obtener la señal (Dietzel et al., 2022). Una de ellas es el tipo de señal electrocardiograma ECG, que funciona como un registro de la actividad eléctrica del corazón medida entre dos puntos de la superficie corporal, permitiendo detectar anomalías en el corazón (Navarro, 2014) .

2.3.6 Marco conceptual o Glosario

- **Aprendizaje automático.** Se refiere a un programa de computadora que es capaz de aprender y generar un comportamiento para el cual no fue programado explícitamente (Joshi, 2020).
- **Arquitectura tecnológica.** Diseño del conjunto de relaciones que existen entre las partes de hardware, software, redes y/o datos que componen un sistema (Mallqui et al., 2019).
- **Balistocardiograma.** Método no invasivo que captura las vibraciones mecánicas generadas por los latidos del corazón (Z. Zhou et al., 2025).
- **Computación en nube (Cloud Computing).** Es el aprovisionamiento de capacidades informáticas (hardware, software o servicios) proporcionadas por un tercero a través de una red (Rong, 2019).
- **Computación de borde (Edge computing).** La Arquitectura que utiliza dispositivos perimetrales (Edge Devices) cercanos al usuario para almacenar cantidades grandes datos (Kuchuk & Malokhvii, 2024), hacer un procesamiento intensivo, tareas de comunicación y mejorar el rendimiento de la aplicación.
- **Computación en niebla (Fog Computing).** Es una arquitectura informática descentralizada donde los datos, las comunicaciones, el almacenamiento y las aplicaciones se distribuyen entre la fuente de datos y la nube realizando un procesamiento local de datos en tiempo real (Resende et al., 2021a).
- **Dispositivo de computación de borde (CB).** Brinda un punto de entrada a las redes de instituciones o proveedores de servicios (Z. Huang et al., 2024) (Dolui & Datta, 2017). Entre este tipo de dispositivos podemos encontrar: enrutadores, conmutadores, multiplexores y otros tipos de dispositivos de acceso a redes de área amplia y metropolitana.
- **Electrocardiograma.** Es el registro de la actividad eléctrica del corazón medida sobre la superficie corporal, para detectar anomalías del corazón (Navarro, 2014).

- **Inteligencia computacional.** Capacidad para que los sistemas se comporten de manera inteligente (Llanes et al., 2020). La inteligencia computacional se enfoca en resolver problemas a los cuales originalmente sólo los humanos pueden encontrar solución.
- **Inteligencia de las cosas.** Es la integración de IoT con la inteligencia artificial que permite analizar de manera eficiente la información obtenidas por los dispositivos (Atlam et al., 2019).
- **Internet de las cosas.** Objetos conformados por sensores y con capacidad para conectarse a internet, obtener y transmitir datos de manera autónoma (Rivera et al., 2024).
- **NEWS2. National Early Warning Score 2.** (En español puntuación nacional de alerta temprana 2). Sistema de puntuación para determinar el estado de salud en personas y determinar la urgencia de atención médica (Huespe et al., 2022a)(Liao et al., 2020a).
- **Nodo sensor.** Dispositivo autónomo que recolecta datos del ambiente y los transmite de forma inalámbrica hacia una estación base (Duobiene et al., 2022).
- **Saturación de oxígeno.** Indica el porcentaje de oxígeno presente en la hemoglobina que se transporta de los pulmones al cuerpo (Pathinarupothi et al., 2019).
- **Sistemas cognitivos.** Apoya el desempeño humano en el contexto de sistemas tecnológicos (Bisantz & Burns, 2014) que podrían resultar complejos, con incertidumbres en sus procesos y en entornos donde existe la presión del tiempo y riesgo en la toma de decisiones.
- **Sistema embebido.** Es un sistema de computacional diseñado para realizar una o algunas pocas funciones dedicadas en dispositivos electrónicos con bajos recursos de hardware (Berta et al., 2024) (Arora, 2016).
- **Sistemas inteligentes.** Son aquellos capaces de tomar decisiones en función de la entrada que reciben. Pueden aprender de interacciones realizadas en el pasado y cambiar para mejorar su comportamiento (Paulovich et al., 2018).
- **TinyML.** TinyML se encuentra en la intersección de aprendizaje automático y sistemas embebidos, y habilita capacidades de aprendizaje automático que se ejecutan en unidades de microcontroladores de baja potencia o en dispositivos IoT livianos (Koufos et al., 2021).

- **Unidad de procesamiento de señales.** Dispositivo electrónico encargado de realizar un procesamiento sobre las señales entregadas por sensores (Rahmani et al., 2021).

2.4 Estado del arte

2.4.1 Revisión sistemática de literatura

El monitoreo de señales fisiológicas (MSF) de pacientes en clínicas, hospitales o de forma ambulatoria requiere de medición de señales, su interpretación para una toma de decisiones oportunas y proactivas realizando estos procesos en salas especiales que requieren infraestructuras con los equipos de medición y personal (Mrozek et al., 2020) (G. L. Santos et al., 2019). Para aliviar estos problemas críticos, se está adoptando ampliamente el paradigma conocido como computación de borde (CB) (Alshehri & Muhammad, 2021). En el caso de un sistema de monitoreo de señales fisiológicas con computación de Borde (SMSFCB), esta realiza la adquisición de señales fisiológicas (SF) y con estos datos obtenidos se puede, a través de los algoritmos de aprendizaje automático (ML), diagnosticar (Faragli et al., 2021) y/o predecir (Supriya & Chattu, 2021).

Sin embargo, los estudios de estado del arte no identifican los SMSFCBs ni los algoritmos de aprendizaje automático (ML) que se vienen usando para el monitoreo de SF, a pesar de la importancia de los SMSFCBs y la tendencia del uso de los ML en el área de salud. La revisión sistemática de literatura realizada tiene como propósito responder a la pregunta de investigación ¿Cuál es el avance de los SMSFCB, en relación con los algoritmos de aprendizaje automático, sistemas de hardware, señales y sensores? Para ello se realiza una revisión sistemática de literatura (RSL) sobre los SMSFCB en revistas indizadas en Scopus, Web of Science (WoS) y PubMed. En este caso se revisan los aspectos del SMSFCB: señales fisiológicas, sensores para el monitoreo, algoritmos de aprendizaje automático y su conceptualización.

2.4.1.1 Metodología

La metodología que se siguió para la revisión sistemática de la literatura se basó en el procedimiento PRISMA (V. Patel et al., 2021b) y los metodológicos (Kitchenham, 2007), las cuales son usadas en artículos de revisión sistemática de literatura (Haghi Kashani et al., 2021) (Behmanesh et al., 2020) para tendencias de tecnologías de

sensores para monitoreo de la salud con computación de borde y tiene las siguientes fases:

- Planificación: se realiza la elaboración de las interrogantes de investigación y se precisó el protocolo.
- Desarrollo: En esta etapa, se utilizan pasos y reglas definidas en base a los criterios de inclusión y exclusión sobre los estudios primarios sobre los artículos.
- Resultados: Se determinan las estadísticas sobre los artículos seleccionados.
- Análisis: Se responde a las preguntas de investigación.

a) Planificación. Para responder a la pregunta de investigación, se plantean las siguientes sub-preguntas:

- RQ1: ¿Qué señales fisiológicas se monitorean en los SMSFCBs?
- RQ2: ¿Qué sensores son utilizados en los SMSFCBs?
- RQ3: ¿Cuáles son los algoritmos de aprendizaje automático utilizados en los SMSFCBs?
- RQ4: ¿Qué dispositivo de CB existen para el monitoreo de señales fisiológicas?
- RQ5: ¿Qué SMSFCB existen?

Para la búsqueda de artículos se considera los repositorios de Scopus, Web of Science (WoS) y PubMed, el periodo 2015-2025 (Setiembre), y la cadena de búsqueda:

("monitoring") AND ("Machine Learning" OR "System" OR "edge computing" OR sensor) AND ("physiological signals")

Esta cadena fue aplicada en tit-abs-key para Scopus y TS para WoS (Tabla 4). Además, se contemplaron criterios de inclusión y exclusión que son dados en la Tabla 5.

Tabla 4. Cadenas de búsqueda según repositorios

Repositorios	Estrategia de búsqueda
Scopus	TITLE-ABS-KEY(("monitoring") AND ("Machine Learning" OR "System" OR "edge computing" OR sensor) AND ("physiological signals"))
Web Of Science	TS=(("monitoring") AND ("Machine Learning" OR "System" OR "edge computing" OR sensor) AND ("physiological signals"))

PubMed	Search: ("monitoring") AND ("Machine Learning" OR "System" OR "edge computing" OR sensor) AND ("physiological signals")
--------	---

Tabla 5. Criterio de selección y exclusión

Criterios de inclusión	Criterios de exclusión
• Artículos al menos una subpregunta de investigación • Tipo: artículos de journals. • Idioma: Inglés y español • Periodo: 2015 - 2025 • Áreas: Ciencias de la computación, ingeniería, medicina	• Resúmenes de congresos (proceedings), Tesis, Capítulos de libro. • Artículos que no presenten validación estadística o computacional. • Artículo duplicado

b) Desarrollo. Los artículos se sometieron a un proceso de selección y revisión previa, utilizando la cadena de búsqueda en las bases de datos seleccionadas, y aplicando los criterios de inclusión y exclusión. Se aplicó la cadena de búsqueda y se encontró un total de 14805 artículos, luego se aplicaron los criterios de inclusión y exclusión y se eliminaron 9663 artículos, restando 5142. A continuación, se eliminaron los artículos duplicados en los repositorios de forma aleatoria y se procedió a la lectura de los resúmenes, eliminando principalmente artículos que no describen procesos de implementación de sistemas de computación de borde de manera práctica quedando 170 artículos. Luego, se realizó la lectura de la introducción y conclusiones, eliminando 44 artículos que en su mayoría correspondían a contextos como transporte, industria y seguridad, quedando 126 artículos. Finalmente se leyó el contenido completo de todos artículos restantes y se eliminaron 33 artículos porque no responden al menos una pregunta de investigación, quedando 93 artículos, Este proceso se muestra en la Figura 12.

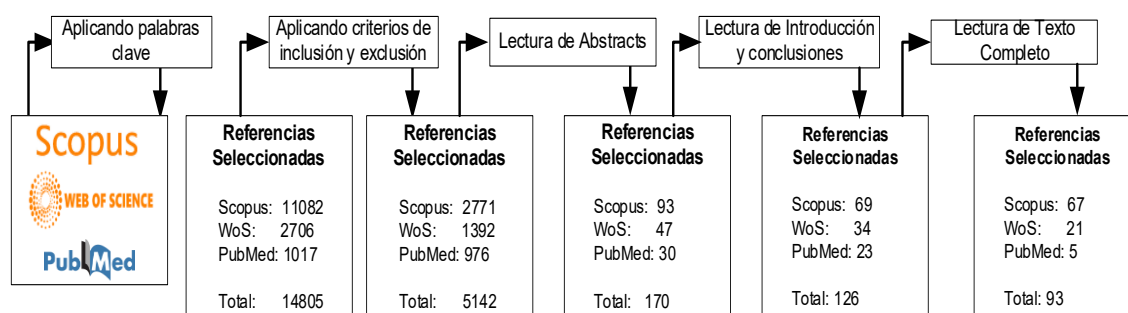


Figura 12. Proceso de selección de artículos

c) Resultados. Los 93 artículos seleccionados representan el 0.62% del total de artículos potenciales, y 23 artículos se encuentran en los 3 repositorios (Tabla 6), siendo Scopus el que presenta mayor número de artículos.

Tabla 6. Número de artículos potenciales y seleccionados

Repositorios	Artículos potenciales	Artículos seleccionados
Scopus	11082	67
Web Of Science	2706	21
PubMed	1017	5
Total	14805	93

* 23 artículos se repiten en los 3 repositorios

El 86% de los artículos (80 de 93) y el 85% de revistas (46 de 54) seleccionadas pertenecen a cuartiles Q1 y Q2 (Tabla 7), lo cual muestra el gran interés en SMSFCB, y garantiza la calidad de los estudios y de las inferencias que se pueda obtener de ellas. Las 54 revistas que publicaron los artículos seleccionados son en su mayoría revistas IEEE (11) y más del 33.3% de los trabajos se encuentran en las revistas Sensors (Q2), IEEE Access (Q1), IEEE Internet of Things Journal (Q1).

Tabla 7. Distribución de revistas por cuartil

Cuartil	Journal	# artículos	# artículos por cuartil
Q1	IEEE Access	8	41
	IEEE Internet of Things Journal	4	
	npj Flexible Electronics	2	
	IEEE journal of biomedical and health informatics	2	
	IEEE Network	2	
	IEEE Sensors Journal	2	
	IEEE Transactions on Mobile Computing	1	
	Computer Communications	1	
	Human-centric Computing and Information Sciences	1	
	Cognitive Computation	1	
	Information Sciences	1	
	IEEE Transactions on Emerging Topics in Computing	1	
	IEEE Transactions on Consumer Electronics	1	
	Internet of Things	1	
	Computers & Security	1	
	Journal of Parallel and Distributed Computing	1	

	IEEE Transactions on Industrial Informatics	1	
	International Journal of Medical Informatics	1	
	Automation in Construction	1	
	Biomedical Signal Processing and Control	1	
	Biosensors	1	
	Future Generation Computer Systems	1	
	IET Intelligent Transport Systems	1	
	Journal of Information Technology in Construction	1	
	Materials & Design	1	
	Microsystems & Nanoengineering	1	
	Sports	1	
Q2	Sensors	19	39
	Journal of Healthcare Engineering	2	
	International Journal of Electrical and Computer Engineering	2	
	Algorithms	1	
	IET Wireless Sensor Systems	1	
	The journal of supercomputing	1	
	Expert Systems	1	
	International Journal of Distributed Sensor Networks	1	
	IEEE Embedded Systems Letters	1	
	Microelectronics Journal	1	
	Medical hypotheses	1	
	Personal and Ubiquitous Computing	1	
	IEEE Journal of Translational Engineering in Health and Medicine	1	
	Wireless personal communications	1	
	Smart Health	1	
	Journal of Signal Processing Systems	1	
	ACM Transactions on Embedded Computing Systems	1	
	Microelectronic Engineering	1	
	Engineering Research Express	1	
Q3	Applied Sciences	5	10
	Indonesian Journal of Electrical Engineering and Computer Science	2	
	Webology	1	
	Microprocessors and Microsystems	1	
	EAI Endorsed Transactions on Pervasive Health and Technology	1	
Q4	Current Directions in Biomedical Engineering	1	3

	Studies in Health Technology and Informatics	1	
	Journal of Mechanics in Medicine and Biology	1	

2.4.1.2 Análisis

a) RQ1: ¿Qué señales fisiológicas se monitorean?

Se identificaron 18 señales fisiológicas en 80 estudios, las cuales se han agrupado en 6 categorías (Cat.) de señales de acuerdo con la literatura (Tabla 8).

Tabla 8. Categorías de señales fisiológicas

ID	Categoría	Descripción	Fuente
FB	Bioeléctrica	Incluye señales que representan la actividad eléctrica generada por el cuerpo, como el electrocardiograma (ECG) que registra la actividad eléctrica del corazón y el electroencefalograma (EEG) que mide la actividad cerebral.	(Oh et al., 2026) (Z. Zhou et al., 2025) (Jin et al., 2025) (S. Wang et al., 2025) (Umer et al., 2025) (N. Zhang et al., 2025) (X. Zhang, Zhan, et al., 2025) (Namazi et al., 2025) (M. A. Khan et al., 2025) (M. Khan et al., 2025) (Luzzani et al., 2025) (U. Verma et al., 2024) (Zholdas et al., 2024) (Y. Zhou et al., 2024) (Satija et al., 2018)
FI	Inercial	Comprende señales que miden la aceleración y la orientación del cuerpo, utilizadas en dispositivos de monitoreo del movimiento y la actividad física.	(Z. Zhou et al., 2025) (Park et al., 2025) (Oh et al., 2026) (Nguyen et al., 2024) (U. Verma et al., 2024) (Iosa et al., 2016)
FM	Biomecánica	Se refiere a señales relacionadas con las fuerzas y movimientos del cuerpo, como la fuerza de agarre, la presión plantar o la cinemática del movimiento.	(Z. Zhou et al., 2025) (Oh et al., 2026) (X. Zhang, Chai, et al., 2025) (X. Wang et al., 2025) (S. Wang et al., 2025) (M. Khan et al., 2025) (U. Verma et al., 2024)
FP	Bioóptica	Incluye señales ópticas que se generan o interactúan con el cuerpo, como la fotopletimografía (PPG) que mide el cambio en la absorción de luz por los tejidos y se utiliza para estimar el pulso y la saturación de oxígeno en sangre.	(Umer et al., 2025) (N. Zhang et al., 2025) (Namazi et al., 2025) (Chowdhury et al., 2025) (M. A. Khan et al., 2025) (Luzzani et al., 2025) (Park et al., 2025) (Nguyen et al., 2024) (Reguig & Labdelli, 2024) (Zholdas et al., 2024)
FC	Bioacústica	Comprende señales de sonido que provienen del cuerpo, como los sonidos cardíacos o pulmonares, utilizados en la auscultación médica.	(Oh et al., 2026) (Park et al., 2025)
FG	Biomagnética	Incluye señales generadas por la actividad magnética del cuerpo, como el magnetoencefalograma (MEG) que mide la actividad cerebral mediante campos magnéticos.	(U. Verma et al., 2024)

Las señales fisiológicas más estudiadas son la señal ECG (Electrocardiográfica) (P01), temperatura (P02) y aceleración (P09) correspondiendo al 32%, 29% y 27.5% de los estudios respectivamente agrupados por categoría (Cat.) (Tabla 9).

Tabla 9. Señales fisiológicas en SMSFCB

Cat.	ID	Señales	Descripción	Referencias
FB	P01	ECG	Señal de la actividad eléctrica del corazón que pueden ser detectadas mediante el electrocardiógrafo (Satija et al., 2017).	(Oh et al., 2026) (S. Wang et al., 2025) (N. Zhang et al., 2025) (X. Zhang, Zhan, et al., 2025) (Namazi et al., 2025) (M. A. Khan et al., 2025) (Luzzani et al., 2025) (U. Verma et al., 2024) (Y. Zhou et al., 2024) (Hamil et al., 2022)(Li et al., 2021)(Mastoi et al., 2019) (Pathinarupothi et al., 2019) (Ravi et al., 2017) (Satija et al., 2017) (J. Wu et al., 2018) (Elhadad et al., 2022) (Premalatha & Bai, 2022) (de Melo et al., 2022)(Scrugli et al., 2022)(Lee & Lee, 2021)(F. Wu et al., 2021)(Dong et al., 2020)(Djelouat et al., 2020) (Uddin, 2019)(Jahan et al., 2022)
	P02	Temperatura	Señal que corresponde al nivel de calor generado por el cuerpo humano (Hamil et al., 2022).	(Oh et al., 2026) (S. Wang et al., 2025) (Umer et al., 2025) (M. A. Khan et al., 2025) (M. Khan et al., 2025) (Park et al., 2025) (U. Verma et al., 2024) (Amudha & Murali, 2021)(Bhat et al., 2021)(Devi & Kalaivani, 2019)(Hamil et al., 2022)(A. Iqbal et al., 2020)(Marquez-Sanchez et al., 2021)(Mauldin et al., 2018)(Said et al., 2021)(Z. Zhang et al., 2020)(Elhadad et al., 2022)(Ashfaq et al., 2022) (Premalatha & Bai, 2022)(Malche et al., 2022)(Rahman et al., 2022)(A. Singh & Chatterjee, 2021)(Dong et al., 2020)
	P03	EEG	Señales eléctricas (Electroencefalograma) de la actividad generada en el cerebro por la generación de impulsos eléctricos de las neuronas (Asghar et al., 2021).	(Jin et al., 2025) (Y. Zhou et al., 2024) (Asghar et al., 2021)(Muhammad et al., 2019)(T. Iqbal et al., 2022)(Hassantabar et al., 2022)
	P04	EMG	Señal de la actividad eléctrica producida por los músculos esqueléticos (Hannan et al., 2021).	(Oh et al., 2026) (M. Khan et al., 2025) (Y. Zhou et al., 2024) (Hannan et al., 2021)(Ravi et al., 2017)(Yang et al., 2018)
FC	P05	Latido de corazón (heart beat)	señal periódica correspondiente a la que genera el corazón cuando late (Devi & Kalaivani, 2019).	(Amudha & Murali, 2021)(Devi & Kalaivani, 2019)(Ekiz et al., 2020)(P. M. Kumar & Devi Gandhi, 2018)
	P06	Sonido	Señal correspondiente a la propagación de ondas mecánicas por un medio (Mauldin et al., 2018).	(Park et al., 2025) (Mauldin et al., 2018)(Ukil et al., 2019)(Verde et al., 2019)(Malche et al., 2022)

	P07	Ultrasónico	Son señales sonoras con frecuencias superiores al límite audible del ser humano (Rosero-Montalvo et al., 2019)	(Rosero-Montalvo et al., 2019)
FG	P08	Campo magnético	campo vectorial que describe como la fuerza magnética afecta las cargas eléctricas en movimiento (Uddin, 2019)	(A. Iqbal et al., 2020)(Javed et al., 2020)(Uddin, 2019)(Yauri et al., 2022)
FI	P09	Aceleración	Señal relacionada a la tasa de cambio de la velocidad en un objeto (Mrozek et al., 2020)	(Oh et al., 2026) (Umer et al., 2025) (Park et al., 2025) (U. Verma et al., 2024) (El Attaoui et al., 2020)(A. Iqbal et al., 2020)(Javed et al., 2020)(Marquez-Sanchez et al., 2021)(Mrozek et al., 2020)(Muhammad et al., 2019)(Ravi et al., 2017)(G. L. Santos et al., 2019)(Sarabia-Jácome et al., 2020)(Satija et al., 2017)(Shiao et al., 2021)(Yacchirema et al., 2019)(Zaid Salah et al., 2022)(Premalatha & Bai, 2022)(Malche et al., 2022)(Dong et al., 2020)(Uddin, 2019)
	P10	Velocidad angular	Medida de la velocidad de rotación sobre qué tan rápido cambia la posición o la orientación de un objeto con el tiempo (El Attaoui et al., 2020)	(Albeshri, 2021)(El Attaoui et al., 2020)(Bhat et al., 2021)(Hannan et al., 2021)(A. Iqbal et al., 2020)(Javed et al., 2020)(Marquez-Sanchez et al., 2021) (Mortazavi et al., 2015) (Ravi et al., 2017)(Shiao et al., 2021)(Uddin, 2019)(Dong et al., 2020)
FM	P11	Movimiento	Señales generadas por el movimiento mecánico humano (Abid et al., 2021)	(X. Wang et al., 2025) (Jin et al., 2025) (Nguyen et al., 2024) (Albeshri, 2021) (Hamil et al., 2022)(Guo & Yang, 2022) (Abid et al., 2021)(Jeng et al., 2021)(Guo & Yang, 2022)
	P12	Respiración	Esta señal está relacionada a los procesos de La inhalación y exhalación de oxígeno (Noto et al., 2018).	(Oh et al., 2026) (X. Zhang, Chai, et al., 2025) (X. Wang et al., 2025) (Namazi et al., 2025) (Nguyen et al., 2024) (U. Verma et al., 2024) (Zholdas et al., 2024) (Amudha & Murali, 2021)(Bhat et al., 2021)(Hamil et al., 2022)(P. M. Kumar & Devi Gandhi, 2018)(A. Singh & Chatterjee, 2021)
	P13	Vibración	Efecto de la vibración mecánica del medio ambiente sobre el cuerpo humano (Rosero-Montalvo et al., 2019).	(Rosero-Montalvo et al., 2019)
FP	P14	Saturación de oxígeno en la sangre	fracción de hemoglobina saturada de oxígeno en relación con la hemoglobina total (Pathinarupothi et al., 2019)	(Oh et al., 2026) (Chowdhury et al., 2025) (M. A. Khan et al., 2025) (Luzzani et al., 2025) (Park et al., 2025) (Nguyen et al., 2024) (Zholdas et al., 2024) (Hamil et al., 2022)(Pathinarupothi et al., 2019) (Said et al., 2021) (Ashfaq et al., 2022) (Malche et al., 2022) (Rahman et al., 2022)(A. Singh & Chatterjee, 2021)
	P15	Ritmo cardíaco (heart rate)	velocidad de los latidos del corazón medida por la cantidad de contracciones del corazón por minuto (Marquez-Sanchez et al., 2021).	(Oh et al., 2026) (Z. Zhou et al., 2025) (X. Zhang, Chai, et al., 2025) (Umer et al., 2025) (N. Zhang et al., 2025) (X. Zhang, Zhan, et al., 2025) (Namazi et al., 2025) (M. Khan et al., 2025) (Nguyen et al., 2024) (Y. Zhou et

				al., 2024) (Zholdas et al., 2024) (Marquez-Sanchez et al., 2021) (Mauldin et al., 2018)(Said et al., 2021)(Velichko, 2021a)(J. Wu et al., 2018)(Elhadad et al., 2022)(Ashfaq et al., 2022)(Premalatha & Bai, 2022)(Malche et al., 2022)(Rahman et al., 2022)(A. Singh & Chatterjee, 2021)(Lee & Lee, 2021)(Pace et al., 2019)(Nath et al., 2022)
	P16	Presión de sangre	Señal relacionada a la fuerza que ejerce la sangre contra las paredes de tus arterias con el transcurso del tiempo (P. M. Kumar & Devi Gandhi, 2018)	(A. Iqbal et al., 2020)(P. M. Kumar & Devi Gandhi, 2018)(Mauldin et al., 2018)(Rosero-Montalvo et al., 2019)(Said et al., 2021)(Z. Zhang et al., 2020)(Elhadad et al., 2022)(Premalatha & Bai, 2022)(A. Singh & Chatterjee, 2021)
	P17	Imágenes	Señales bidimensionales que pueden ser usadas para representar imágenes estáticas o de video (Piątkowski & Walkowiak, 2022).	(Mulfari et al., 2017)(Rachakonda et al., 2020)(ŞİŞİK & SERT, 2020)(Piątkowski & Walkowiak, 2022)(Suresha et al., 2022)
	P18	Fotopleetismo grafía	Es una señal obtenida ópticamente que puede usarse para detectar cambios en el volumen sanguíneo microvascular del tejido (Alian & Shelley, 2014)	(N. Zhang et al., 2025) (Namazi et al., 2025) (Chowdhury et al., 2025) (Luzzani et al., 2025) (Reguig & Labdelli, 2024) (Y. Zhou et al., 2024) (Zholdas et al., 2024) (Pathinarupothi et al., 2019)(Lee & Lee, 2021)

Algunas señales fisiológicas son denominadas de forma diferente como en el caso de la saturación de oxígeno (P11) que es también denominada como SPO2 (Pathinarupothi et al., 2019); la velocidad angular (P10) como velocidad de rotación (El Attaoui et al., 2020); latido del corazón (P05) como pulso (Devi & Kalaivani, 2019); ritmo cardiaco (P15) como frecuencia cardiaca (Mauldin et al., 2018) (Said et al., 2021); movimiento (P12) como actividades de la vida diaria (Albeshri, 2021); respiración (P13) como frecuencia respiratoria (Hamil et al., 2022).

b) RQ2: ¿Qué sensores son utilizados en el monitoreo de señales fisiológicas?

Se identificó 19 sensores para la adquisición de señales fisiológicas en 35 estudios encontrados en esta sección, donde los más utilizados son el AD8232 (N02) para señales ECG y el sensor de aceleración como el 20948 (N05) con 20% y 12% de los estudios respectivamente (Tabla 10).

Tabla 10. Sensores utilizados en dispositivos de borde

ID	Sensor	Descripción	Fabricante	Referencias
----	--------	-------------	------------	-------------

N01	ADS1292	Adquisición de señales EEG. Aplicaciones con hasta 6 derivaciones, resolución de 24 bits y comunicación de interfaz periférica serial.	Texas Instruments	(T. Iqbal et al., 2022)(Pathinarupothi et al., 2019)(Hassantabar et al., 2022)
N02	AD8232	Adquisición de señales ECG. Integrado con amplificadores de señal y filtrado de ruido y con salida de datos analógica.	Analog Devices	(M. A. Khan et al., 2025) (Y. Zhou et al., 2024) (J. Wu et al., 2018)(Premalatha & Bai, 2022)(Scrugli et al., 2022)(F. Wu et al., 2021)(Jahan et al., 2022) (Pathinarupothi et al., 2019)
N03	ADS1198	Adquisición de señales ECG. Con comunicaciones de interfaz periférica serial y adquisición de datos analógica con resolución de 12 bits.	Texas Instruments	(Yang et al., 2018)
N04	MAX30102	Adquisición de SPO2 y pulso. Protocolo de comunicación entre circuitos con una resolución de 16 bits y filtro de 60Hz.	Maxim Integrated	(M. A. Khan et al., 2025) (Zholdas et al., 2024) (Ashfaq et al., 2022)(Premalatha & Bai, 2022)
N05	INERTIAL 20948	Adquisición de patrones de movimiento. Sensor inercial de 9 grados de libertad con protocolo de comunicación inter-circuito a 100 kHz.	TDK INVENSENSE	(Abid et al., 2021)(Ahamed et al., 2020)(Albeshri, 2021)(Yauri et al., 2022)
N06	MMA8452 Q	Adquisición de señales de aceleración. Sensor con 3 ejes de medición y bajo consumo de energía con salida digital de 12 bits.	Freescall Semiconductor	(El Attaoui et al., 2020)(Yacchirema et al., 2019)(Guo & Yang, 2022)
N07	BMI160	Adquisición de patrones de movimiento. Con interfaz de comunicación de periférico serial e inter circuito y resolución de 16 bits de bajo consumo de energía.	Bosch Sensortec	(Marquez-Sanchez et al., 2021)
N08	MPU6050	Adquisición de patrones de aceleración y giro con protocolo de comunicación inter circuito y frecuencia de muestreo de hasta 8 kHz	MicroPower Direct	(El Attaoui et al., 2020)
N09	MPU6500	Adquisición de patrones de aceleración y giro. Compatible con comunicación de periférico serial e inter circuito con frecuencia de muestreo de hasta 32 kHz	MicroPower Direct	(Hannan et al., 2021)
N10	LSM6DS0	Adquisición de patrones de aceleración y giro. Unidad de movimiento inercial con resolución de 16 bits y detección de 6 ejes de	STMicroelectronics	(Yacchirema et al., 2019)

		movimiento de bajo consumo de energía.		
N11	ADXL345	Adquisición de patrones de aceleración de ejes. Con ajuste de sensibilidad de hasta 16g y recepción de datos analógica de 13 bits.	Analog Devices	(Yacchirema et al., 2019) (Sarabia-Jácome et al., 2020)(Zaid Salah et al., 2022)
N12	RPi-infrared (IR) camera	Adquisición de imágenes con campo de visión de 75.7 grados. Con leds infrarrojos y puerto de comunicación de interfaz serie para cámaras.	Unistorm	(Pace et al., 2019)(Suresha et al., 2022)
N13	CMS50E	Adquisición de SPO2 y pulso. Con almacenamiento automático de datos y una resolución de 1%	Sumida Corporation	(Devi & Kalaivani, 2019)(Hamil et al., 2022)(Rahman et al., 2022)
N14	APDS-9008	Adquisición de señal de pulso. Con salida de datos analógica y consumo de corriente de hasta 20 miliamperios.	AVAGO TECHNOLOGIES	(Kalantarian et al., 2017)(Lee & Lee, 2021)
N15	MAX31855	Adquisición de temperatura mediante un termopar con protocolo de comunicación serial periférica.	Maxim Integrated	(Marquez-Sanchez et al., 2021)
N16	MLX90614	Adquisición de temperatura infrarroja sin contacto, con adquisición de datos analógica con 17 bits de resolución.	Melexis	(Ashfaq et al., 2022)
N17	Si7020-A20	Adquisición de temperatura en formato digital con protocolo de comunicación inter circuito, 12 bits de resolución y precisión de 4%	Silicon Laboratories	(Z. Zhang et al., 2020)
N18	LM35	Adquisición de temperatura analógica, rango de -55°C a 150°C, bajo consumo de energía y precisión de 0.5°C.	Texas Instruments	(M. A. Khan et al., 2025) (Amudha & Murali, 2021) (Premalatha & Bai, 2022)
N19	HC-SR04	Detección de distancia en base a ultrasonidos de 40 kHz con distancia de 2 a 450 centímetros.	ETC2	(Rosero-Montalvo et al., 2019)

La relación entre las señales fisiológicas, los sensores y sus usos en el monitoreo se muestran en la Tabla 11, donde se observan que los sensores usualmente se utilizan para medir más de 1 señal.

Tabla 11. Relación entre los sensores, y señales

Sensor	Señal	Aplicaciones de monitoreo
--------	-------	---------------------------

N01	P03	Afecciones cardíacas (Pathinarupothi et al., 2019) , stress (T. Iqbal et al., 2022), trastornos de salud mental: esquizoafectivo, depresivo mayor y bipolar (Hassantabar et al., 2022), patologías (Muhammad et al., 2019)
N02	P01	enfermedades cardíacas (J. Wu et al., 2018), monitoreo de signos vitales (M. A. Khan et al., 2025) (Y. Zhou et al., 2024) (Premalatha & Bai, 2022), enfermedades cardiovasculares (Scrugli et al., 2022), monitoreo de atención médica (F. Wu et al., 2021), arritmias cardíacas (Jahan et al., 2022)
N03	P01	rehabilitación de accidentes cerebrovasculares (Yang et al., 2018)
N04	P15	enfermedades cardiovasculares (Ashfaq et al., 2022) , monitoreo parámetros vitales (M. A. Khan et al., 2025) (Zholdas et al., 2024) (Premalatha & Bai, 2022)
	P11	enfermedades cardiovasculares (Ashfaq et al., 2022), monitoreo parámetros vitales (Premalatha & Bai, 2022)
N05	P12	reconocimiento de actividad humana (Abid et al., 2021), demencia (Ahamed et al., 2020), reconocimiento de actividades (Albeshri, 2021), reconocimiento de actividades (Yauri et al., 2022)
N06	P09	Detección de caídas (El Attaoui et al., 2020), detección de caídas (Yacchirema et al., 2019), monitoreo de señales fisiológicas (Guo & Yang, 2022)
N07	P09	Monitoreo de actividad humana (Marquez-Sanchez et al., 2021)
	P12	Monitoreo de actividad humana (Marquez-Sanchez et al., 2021)
N08	P10	Detección de caídas (El Attaoui et al., 2020)
	P09	Detección de caídas (El Attaoui et al., 2020)
N09	P10	Corrección de postura (Hannan et al., 2021)
	P09	Corrección de postura (Hannan et al., 2021)
N10	P10	Detección de caídas (Yacchirema et al., 2019)
	P09	Detección de caídas (Yacchirema et al., 2019)
N11	P09	Detección de anomalías en movimiento (Sarabia-Jácome et al., 2020), detección de caídas (Yacchirema et al., 2019), detección de caídas (Zaid Salah et al., 2022)
N12	P17	Variabilidad de latidos cardíacos (Pace et al., 2019), Monitoreo de variables clínicas (Suresha et al., 2022)
N13	P15	arritmia cardíaca (Devi & Kalaivani, 2019), monitoreo de signos vitales (Hamil et al., 2022), variabilidad de frecuencia cardíaca (Rahman et al., 2022)
	P18	arritmia cardíaca (Devi & Kalaivani, 2019)
N14	P05	Monitoreo de latidos de corazón (Kalantarian et al., 2017), estimación de frecuencia cardíaca (Lee & Lee, 2021)
N15	P02	Detección de anomalías en la temperatura (Marquez-Sanchez et al., 2021)
N16	P02	Anomalías en temperatura corporal (Ashfaq et al., 2022)
N17	P02	Monitoreo de actividades deportivas (Z. Zhang et al., 2020)
N18	P02	Análisis predictivo de salud (Amudha & Murali, 2021), monitoreo de signos vitales (M. A. Khan et al., 2025) (Premalatha & Bai, 2022)
N19	P06	Detección de posturas (Rosero-Montalvo et al., 2019)
	P07	Detección de posturas (Rosero-Montalvo et al., 2019)

c) RQ3: ¿Cuáles son los algoritmos de aprendizaje automático utilizados para el monitoreo de señales fisiológicas?

En la etapa de revisión de literatura se encontraron 37 tipos de algoritmos (Tabla 12) en 74 trabajos encontrados en esta sección (los cuales se repiten en la distribución de frecuencia), que se aplican en el procesamiento de señales fisiológicas en dispositivos de Borde. Se observa que los algoritmos más utilizadas son Máquina de Vector Soporte (SVM) con 24% (18 de 74 trabajos) de los estudios, seguida por las redes neuronales convolucionales (CNN) con 29.7% (22 de 74 trabajos).

Tabla 12. Algoritmos de aprendizaje en CB

ID	Algoritmos	Referencias	Frecuencia
A01	Redes neuronales convolucionales (CNN)	(Oh et al., 2026) (X. Zhang, Chai, et al., 2025) (X. Wang et al., 2025) (Jin et al., 2025) (Namazi et al., 2025) (Chowdhury et al., 2025) (U. Verma et al., 2024) (Abid et al., 2021) (Bhat et al., 2021) (Hamil et al., 2022) (Jordan et al., 2020) (Li et al., 2021) (Mauldin et al., 2018) (Muhammad et al., 2019) (Mulfari et al., 2017) (Rachakonda et al., 2020) (Resende et al., 2021b) (G. L. Santos et al., 2019) (Suresha et al., 2022) (Zaid Salah et al., 2022) (de Melo et al., 2022) (Scrugli et al., 2022)	22
A02	Redes neuronales artificiales (ANN)	(Asghar et al., 2021) (Gembaczka et al., 2019) (Hamil et al., 2022) (Huifeng et al., 2020) (Mrozek et al., 2020) (ŞİŞİK & SERT, 2020) (Velichko, 2021a) (Malche et al., 2022)	9
A03	Aprendizaje profundo (DL)	(Umer et al., 2025) (Chowdhury et al., 2025) (Albeshri, 2021) (Hannan et al., 2021) (Mauldin et al., 2018) (Ravi et al., 2017) (Sarabia-Jácome et al., 2020) (Yacchirema et al., 2019) (P. Verma et al., 2022) (Lee & Lee, 2021)	10
A04	Red neuronal perceptrón multicapa (MLP)	(X. Zhang, Zhan, et al., 2025) (El Attaoui et al., 2020) (Suresha et al., 2022) (A. Iqbal et al., 2020) (Yang et al., 2018)	5
A05	redes neuronales recurrentes (RNN)	(Namazi et al., 2025) (U. Verma et al., 2024) (Huifeng et al., 2020) (Marquez-Sanchez et al., 2021) (Mastoi et al., 2019) (Mauldin et al., 2018) (Morán et al., 2021) (Musci et al., 2021) (Rachakonda et al., 2020) (Sarabia-Jácome et al., 2020) (Uddin, 2019)	11
A06	Aprendizaje por refuerzo profundo	(Lim et al., 2021)	1
A07	Memoria a corto plazo largo (LSTM)	(Namazi et al., 2025) (Marquez-Sanchez et al., 2021) (Musci et al., 2021) (Sarabia-Jácome et al., 2020) (Zaid Salah et al., 2022)	5
A08	Red neuronal a corto plazo (LSTN)	(Huifeng et al., 2020)	1
A09	K-Vecinos Cercanos (KNN)	(Abid et al., 2021) (Ahamed et al., 2020) (Asghar et al., 2021) (El Attaoui et al., 2020) (Ekiz et al., 2020) (Hamil et al., 2022) (Hannan et al., 2021) (A. Iqbal et al., 2020)	14

		(Javed et al., 2020) (Suresha et al., 2022) (Zaid Salah et al., 2022) (Ashfaq et al., 2022)	
A10	Máquina de Vector Soporte (SVM)	(U. Verma et al., 2024) (Abid et al., 2021) (Amudha & Murali, 2021) (El Attaoui et al., 2020) (Hamil et al., 2022) (Hannan et al., 2021) (A. Iqbal et al., 2020) (Jeng et al., 2021) (Mauldin et al., 2018) (Mortazavi et al., 2015) (Mrozek et al., 2020) (Said et al., 2021) (Shiao et al., 2021) (ŞİŞİK & SERT, 2020) (Ukil et al., 2019) (Yang et al., 2018) (Zaid Salah et al., 2022) (Ashfaq et al., 2022)	18
A11	Redes de Vector Soporte (SVN)	(Asghar et al., 2021)	1
A12	SVM de mínimos cuadrados (LSSVM)	(Premalatha & Bai, 2022)	1
A13	Regresión Logística (RL)	(Amudha & Murali, 2021) (Ekiz et al., 2020) (Hannan et al., 2021) (Kalantarian et al., 2017) (P. M. Kumar & Devi Gandhi, 2018) (Yacchirema et al., 2019) (Ashfaq et al., 2022)	7
A14	Regresión Lineal (LIR)	(Reguig & Labdelli, 2024) (Ashfaq et al., 2022)	2
A15	Árboles de decisión (DT)	(Amudha & Murali, 2021) (Ashfaq et al., 2022) (Yacchirema et al., 2019)	3
A16	Bosques aleatorios (RF)	(M. Khan et al., 2025) (U. Verma et al., 2024) (Y. Zhou et al., 2024) (Albeshri, 2021) (El Attaoui et al., 2020) (Ekiz et al., 2020) (Hamil et al., 2022) (Hannan et al., 2021) (A. Iqbal et al., 2020) (Suresha et al., 2022) (Rahman et al., 2022)	11
A17	Redes Bayesianas (BN)	(Kalantarian et al., 2017)	1
A18	K Means (KM)	(Lim et al., 2021) (Marquez-Sanchez et al., 2021) (Piątkowski & Walkowiak, 2022)	3
A19	Árboles de decisión potenciados (BDT)	(Mrozek et al., 2020) (Verde et al., 2019)	2
A20	bayes ingenuos (NB)	(Asghar et al., 2021) (Ekiz et al., 2020) (Ashfaq et al., 2022)	3
A21	AdaBoost (AB)	(Albeshri, 2021) (Suresha et al., 2022)	2
A22	XGBoost (XGB)	(M. Khan et al., 2025) (N. Zhang et al., 2025) (Suresha et al., 2022)	3
A23	Ensemble boosted (EB)	(Ahamed et al., 2020) (Yacchirema et al., 2019)	2
A24	RUSBoosted (RB)	(Ahamed et al., 2020)	1
A25	Máquina de aumento de gradiente (GBM)	(Amudha & Murali, 2021)	1
A26	Soft-Voting y Autoaprendizaje (SVSL)	(Albeshri, 2021)	1
A27	Modelo lineal generalizado (GLM)	(Albeshri, 2021)	1
A28	Enet (EN)	(Amudha & Murali, 2021)	1
A29	Optimización Mínima Secuencial (SMO)	(Kalantarian et al., 2017)	1
A30	Aprendizaje federado (FL)	(Lim et al., 2021)	1

A31	Reconocimiento óptico de caracteres (OCR)	(Rachakonda et al., 2020)	1
A32	Análisis de componentes (PCA)	(Rachakonda et al., 2020) (Rosero-Montalvo et al., 2019) (P. Verma et al., 2022)	3
A33	agrupamiento difuso de C-medias (FCMC)	(ŞİŞİK & SERT, 2020)	1
A34	Red neuronal LogNet (LN)	(Velichko, 2021a)	1
A35	Análisis Discriminante Lineal (LDA)	(Yang et al., 2018)	1
A36	Detección de compresión (DC)	(Djelouat et al., 2020)	1
A37	softmax (SM)	(Suresha et al., 2022)	1

Por otro lado, se han identificado 21 usos de 31 algoritmos de aprendizaje automático sobre 13 señales fisiológicas, como se muestran en la Tabla 13, en donde el mayor uso se da en la detección de caídas con el 23% de los trabajos.

Tabla 13. Algoritmos y su uso en el monitoreo

#	Uso	Alg.	SF	Fuente
1	Reconocimiento de actividad física	A01	P09	(U. Verma et al., 2024) (X. Wang et al., 2025) (Abid et al., 2021) (Z. Zhang et al., 2020)
		A02	P09	(Malche et al., 2022)
		A04	P09	(A. Iqbal et al., 2020)
		A05	P09	(Uddin, 2019)
		A09	P09, P12	(A. Iqbal et al., 2020) (Javed et al., 2020) (Abid et al., 2021)
		A10	P09	(Abid et al., 2021) (A. Iqbal et al., 2020) (Shiao et al., 2021)
		A16	P09	(A. Iqbal et al., 2020)
P01, P03, P04, P15, P18	(Y. Zhou et al., 2024)			
2	Detección de caídas	A01	P09	(U. Verma et al., 2024) (Mauldin et al., 2018) (G. L. Santos et al., 2019) (Zaid

#	Uso	Alg.	SF	Fuente
6	Detección de signos vitales	A01	P01, P14	(U. Verma et al., 2024) (Oh et al., 2026) (X. Zhang, Chai, et al., 2025) (Namazi et al., 2025) (Hamil et al., 2022) (de Melo et al., 2022) (Scrugli et al., 2022) (Chowdhury et al., 2025)
		A09	P02, P03	(Hamil et al., 2022)
		A14	P18	(Reguig & Labdelli, 2024)
7	Reconocimiento de posturas	A03	P09	(Umer et al., 2025) (Yacchirema et al., 2019)
		A10	P09	(Hannan et al., 2021) (Jeng et al., 2021) (Mortazavi et al., 2015)
		A13	P09	(Hannan et al., 2021)
		A16	P09	(Hannan et al., 2021) (Jeng et al., 2021)
8	Detección de stress	A01	P01, P17	(Oh et al., 2026) (Rachakonda et al., 2020)
		A05	P17	(Rachakonda et al., 2020)
		A31	P17	(Rachakonda et al., 2020)
		A32	P17	(Rachakonda et al., 2020)

3				Salah et al., 2022)
		A02	P09	(Mrozek et al., 2020)
		A03	P09	(Umer et al., 2025) (U. Verma et al., 2024)(Albeshri, 2021) (Mauldin et al., 2018) (Sarabia-Jácome et al., 2020)
		A04	P01	(El Attaoui et al., 2020)
		A05	P09	(Mauldin et al., 2018) (Musci et al., 2021) (Sarabia-Jácome et al., 2020)
		A07	P09	(Musci et al., 2021) (Sarabia-Jácome et al., 2020) (Zaid Salah et al., 2022)
		A09	P09	(El Attaoui et al., 2020) (Zaid Salah et al., 2022)
		A10	P09	(El Attaoui et al., 2020) (Mauldin et al., 2018) (Zaid Salah et al., 2022)
		A13	P09	(A. Iqbal et al., 2020)
		A15	P09	(Yacchirema et al., 2019)
		A16	P09	(Mrozek et al., 2020)
		A19	P09	(Yacchirema et al., 2019)
		A21	P09	(Albeshri, 2021)
		A23	P09	(Yacchirema et al., 2019)
		A26	P09	(Albeshri, 2021)
		A27	P09	(Albeshri, 2021)
3	Corrección de postura	A09	P11	(Hannan et al., 2021) (Rosero-Montalvo et al., 2019)
		A32	P12	(Rosero-Montalvo et al., 2019)
4	Monitoreo de deportistas	A02	P01, P09	(Huifeng et al., 2020)
		A05	P01, P02, P09	(Namazi et al., 2025) (Marquez-Sanchez et al., 2021)
		A07	P02, P09	(Marquez-Sanchez et al., 2021)
		A08	P01, P09	(Huifeng et al., 2020)

		A24	P12	(Ahamed et al., 2020)
9	Detección de patologías	A01	P03	(Jin et al., 2025) (Namazi et al., 2025) (Muhammad et al., 2019)
10	Reconocimiento de emociones	A02	P03	(Asghar et al., 2021)
		A09	P03	(Asghar et al., 2021)
		A11	P03	(Asghar et al., 2021)
		A20	P03	(Asghar et al., 2021)
11	Detección de enfermedades	A02	P01, P02, P15	(Supriya & Chattu, 2021) (Velichko, 2021a)
		A10	P11, P14, P02, P05, P13,	(Hamil et al., 2022) (Mrozek et al., 2020) (Amudha & Murali, 2021) (Ukil et al., 2019)
		A13	P16	(Amudha & Murali, 2021) (P. M. Kumar & Devi Gandhi, 2018)
		A15	P02, P15	(Hamil et al., 2022)
		A16	P01, P02	(Y. Zhou et al., 2024) (Amudha & Murali, 2021)
		A21	P14	(Ukil et al., 2019)
		A25	P02, P15	(Amudha & Murali, 2021)
		A28	P02, P15	(Amudha & Murali, 2021)
		A34	P15	(Velichko, 2021a)
12	Predicción de asma	A01	P01	(Hamil et al., 2022) (de Melo et al., 2022) (Scrugli et al., 2022)
13	Diagnóstico de tumores	A02	P17	(ŞİŞİK & SERT, 2020)
		A10	P17	(Said et al., 2021)
		A33	P17	(ŞİŞİK & SERT, 2020)
14	Extracción de bioinformación	A03	P15, P01	(Chowdhury et al., 2025) (Umer et al., 2025) (Lee & Lee, 2021)
15	Autenticación de personal	A04	P15	(Ekiz et al., 2020)
		A09	P15	(Ekiz et al., 2020)
		A13	P15	(Ekiz et al., 2020)
		A16	P15	(Ekiz et al., 2020)
		A20	P15	(Ekiz et al., 2020)
16	Rehabilitación	A04	P16, P02	(X. Zhang, Zhan, et al., 2025) (Yang et al., 2018)
		A10	P16, P02	(Yang et al., 2018)
		A35	P16, P02	(Yang et al., 2018)
17	Clasificación de latidos del corazón	A05	P05	(Mastoi et al., 2019)

		A18	P02, P09	(Marquez-Sanchez et al., 2021)
5	Detección de enfermedades cardiovasculares	A09	P09, P02, P15	(Ashfaq et al., 2022)
		A10	P02, P15	(Ashfaq et al., 2022)
		A13	P02, P15	(Ashfaq et al., 2022)
		A14	P02, P15	(Ashfaq et al., 2022) (Reguig & Labdelli, 2024)
		A15	P02, P15	(Ashfaq et al., 2022)
		A20	P09, P02, P15	(Ashfaq et al., 2022)

18	Identificación de signos de demencia	A23 A24	P12 P12	(Ahamed et al., 2020) (Ahamed et al., 2020)
19	Reconocimiento de máscara facial	A18	P17	(Rachakonda et al., 2020)
20	Detección de trastornos de la voz	A19	P06	(Verde et al., 2019)
21	Clasificación de señales	A01	P01	(U. Verma et al., 2024) (Oh et al., 2026) (Namazi et al., 2025) (Li et al., 2021)

d) RQ4: ¿Qué dispositivo de CB existen para el monitoreo de señales fisiológicas?

Se identificaron 31 dispositivos (sistemas) de CB en 76 estudios seleccionados para esta sección, donde, el 28% de los estudios usan un teléfono inteligente y el 18% la microcomputadora Raspberry Pi, como se muestra en la Tabla 14. Además, se observa que el 40% de los estudios usan dispositivos basados en microcontroladores, y solo el 9% consideran la computadora estándar.

Tabla 14. Dispositivo de CB

ID	Dispositivo	Descripción	Referencias	Frec
S01	Teléfono inteligente	Dispositivo que combina las características de una computadora personal y de un teléfono convencional (Bhat et al., 2021).	(Z. Zhou et al., 2025) (Jin et al., 2025) (Umer et al., 2025) (X. Zhang, Zhan, et al., 2025) (M. A. Khan et al., 2025) (Park et al., 2025) (Bhat et al., 2021) (Devi & Kalaivani, 2019) (Hannan et al., 2021) (A. Iqbal et al., 2020) (Javed et al., 2020) (Jeng et al., 2021) (Mauldin et al., 2018) (Mrozek et al., 2020) (Ravi et al., 2017) (Rachakonda et al., 2020) (Satija et al., 2017) (Ukil et al., 2019) (Verde et al., 2019) (Rahman et al., 2022) (Dong et al., 2020) (Djelouat et al., 2020)	22
S02	Raspberry pi	La Raspberry Pi es una placa de computadora de bajo costo, pequeña y portátil (El Attaoui et al., 2020).	(Jin et al., 2025) (M. A. Khan et al., 2025) (El Attaoui et al., 2020) (Bhat et al., 2021) (Hamil et al., 2022) (Marquez-Sanchez et al., 2021) (Mulfari et al., 2017) (Sarabia-Jácome et al., 2020) (ŞİŞİK & SERT, 2020) (Elhadad et al., 2022) (P. Verma et al., 2022) (Ashfaq et al., 2022) (Pace et al., 2019) (Suresha et al., 2022)	14
S03	Computadora	tipo de ordenador personal, diseñado y fabricado para ser instalado en una ubicación estática (Rath et al., 2009)	(Oh et al., 2026) (Luzzani et al., 2025) (U. Verma et al., 2024) (Albeshri, 2021) (Asghar et al., 2021) (Musci et al., 2021) (Uddin, 2019)	7

S04	Reloj inteligente	es un reloj de pulsera dotado capaz de acceder a Internet, realizar y recibir llamadas telefónicas (Ekiz et al., 2020)	(Ekiz et al., 2020) (Mortazavi et al., 2015) (Lee & Lee, 2021)	3
S05	MSP430	familia de microcontroladores fabricados por Texas Instruments basado en un CPU de 16 bits (Ghasemzadeh et al., 2015)	(Ghasemzadeh et al., 2015) (Kalantarian et al., 2017) (Pathinarupothi et al., 2019)	3
S06	FPGA	dispositivo programable con bloques de interconexión programados usando un lenguaje de descripción (Li et al., 2021).	(Li et al., 2021) (Morán et al., 2021) (de Melo et al., 2022)	3
S07	Intel Edison	Es un microprocesador basado en un sistema linux yocto embebido (S. M. Patel & Kanawade, 2017)	(Ravi et al., 2017) (Elhadad et al., 2022)	2
S08	Pulsera inteligente	pulsera que debe monitorear y dar seguimiento a los datos relacionados con la forma física (Thamaraiselvi et al., 2019)	(Devi & Kalaivani, 2019) (Ekiz et al., 2020)	2
S09	Arduino Uno	placa electrónica basada en el microcontrolador ATmega328P (Badamasi, 2014).	(Reguig & Labdelli, 2024) (Hamil et al., 2022) (Velichko, 2021a)	3
S10	Arduino nano	placa electrónica compacta, compatible con protoboards, basada en el microcontrolador ATmega328P (Rosero-Montalvo et al., 2019)	(Rosero-Montalvo et al., 2019) (A. Singh & Chatterjee, 2021)	2
S11	STM32L451	microcontrolador de propósito general de 32 bit basado en la versión v7-M de los microprocesadores ARM (Yacchirema et al., 2019)	(Jordan et al., 2020) (Yacchirema et al., 2019)	2
S12	nRF52840	Es un dispositivo microcontrolador System On Chip basado en la CPU ARM CortexM4 de 32 bits (F. Wu et al., 2021)	(J. Wu et al., 2018) (F. Wu et al., 2021) (Guo & Yang, 2022)	3
S13	Ordenadores de una sola placa	computadora construida en una sola placa de circuito, con microprocesador, memoria y otras características de una computadora funcional (Won et al., 2020).	(Rachakonda et al., 2020) (Yauri et al., 2022)	2
S14	BeagleBone Black	placa computadora de hardware libre de bajo consumo con una CPU ARM Cortex-A8 (que puede usar Linux) (Revision C, 2014)	(Said et al., 2021)	1
S15	Arm Cortex A9	procesador multinúcleo de 32 bits de alto el rendimiento con una arquitectura basada en RISC (Resende et al., 2021b)	(Resende et al., 2021b)	1
S16	ARM STM32L4	El microcontrolador se basa en una nueva plataforma de ultra bajo consumo basado en el núcleo RISC de 32 bits ARM M4 (Lee & Lee, 2021)	(Lee & Lee, 2021)	1
S17	Arduino nano 33	placa electrónica con conectividad WiFi y Bluetooth con el criptochip ATECC608A y basada en el microcontrolador SAMD21G18A (Zaid Salah et al., 2022)	(Zaid Salah et al., 2022)	1
S18	ESP32	Tarjeta electrónica de desarrollo con conectividad WiFi y Bluetooth, basada en un CPU 32-bit de doble núcleo Tensilica Xtensa LX6 (Babiuch et al., 2019).	(X. Zhang, Chai, et al., 2025) (Zholdas et al., 2024) (Z. Zhang et al., 2020)	3
S19	ESP8266	Microcontrolador diseñado por una Espressif Systems con conectividad WiFi basada en un Tensilica L106 32 bits (Amudha & Murali, 2021).	(Amudha & Murali, 2021)	1
S20	NodeMCU	placa de desarrollo basada en el ESP8266 para facilitar la programación de un microcontrolador (Kashyap et al., 2018)	(Amudha & Murali, 2021)	1
S21	STM32F411	Microcontrolador del núcleo Cortex M4 con unidad de punto flotante, y frecuencia de hasta 100 MHz (Piątkowski & Walkowiak, 2022).	(Piątkowski & Walkowiak, 2022)	1
S22	nRF5340	Es un microcontrolador System On Chip con dos procesadores Arm CortexM33 con conectividad Bluetooth (Malche et al., 2022)	(Malche et al., 2022)	1
S23	Cortex-M4	Microcontroladores de STMicroelectronics con una unidad de punto flotante (FPU) y procesamiento de señal digital (DSP) (Kim et al., 2021)	(Morán et al., 2021) (Park et al., 2025)	2

S24	SensorTile	módulo IoT basado en un microcontrolador STM32L476 de 80 MHz y conectividad Bluetooth (Scrugli et al., 2022)	(Scrugli et al., 2022) (Nath et al., 2022) (Hassantabar et al., 2022)	3
S25	Plataforma de sensor e-Health	Es una placa de desarrollo que permite realizar aplicaciones médicas y biométricas con 9 sensores diferentes para tarjetas Arduino y Raspberry (Hamil et al., 2022)	(Hamil et al., 2022)	1
S26	IoT Vestible	Dispositivos vestibles inteligentes con capacidades de procesar datos generados por los usuarios que los usan (P. M. Kumar & Devi Gandhi, 2018)	(P. M. Kumar & Devi Gandhi, 2018)	1
S27	Banda inteligente Hexoskin	Es una camisa inteligente con conectividad Bluetooth capaz de monitorear Datos cardíacos, pulmonares, de actividad y de sueño (Abid et al., 2021)	(Abid et al., 2021)	1
S28	STC8A8K64D4	Se basa en un núcleo 8051 mejorado. No requiere un oscilador externo ni circuito de reset externo	(S. Wang et al., 2025)	1
S29	STM32103C8T6	microcontrolador de 32 bits basado en ARM Cortex-M3, fabricado por STMicroelectronics.	(Y. Zhou et al., 2024)	1
S30	STM32F407VE T6	microcontrolador de 32 bits de STMicroelectronics, basado en el núcleo ARM Cortex-M4	(X. Zhang, Zhan, et al., 2025)	1
S31	AFE4950	microcontrolador integrado para medición de señales biomédicas, de Texas Instruments	(Nguyen et al., 2024)	1

e) RQ5: ¿Qué SMSFCB existen?

Considerando que los SMSFCB incluyen sensores, señales fisiológicas, algoritmos y dispositivos de computación de borde, se han identificado siete SMSFCB en los artículos seleccionados.

El primer integra tecnologías textiles, electrónicas e inteligentes para realizar el monitoreo de la actividad cardíaca, usando una banda torácica con electrodos que registran señales ECG transmitidas a un dispositivo móvil para su análisis posterior. Mediante una red neuronal que detecta la intensidad del ejercicio (X. Zhang, Zhan, et al., 2025). Por otro lado, en el caso del Internet de las Cosas Médicas (IoMT) se identificó una solución para monitorear los signos vitales de los conductores y prevenir accidentes compuesto por un sistema embebido que adquiere datos fisiológicos en una placa Raspberry Pi que se complementa con una aplicación móvil que muestra los datos y emite alertas (M. A. Khan et al., 2025). También se encontró un sistema que monitorea el estado fisiológico de los trabajadores en obras de construcción usando un IMU para detectar caídas, junto con sensores PPG, de temperatura corporal y acústicos que permiten evaluar signos vitales, donde los datos se procesan en una aplicación conectada a un servidor, conformando un sistema ciberfísico (Park et al., 2025). En el caso de uso de un dispositivo portátil se integró capacidad del Internet de las cosas

para detectar el nivel de estrés utilizando un microcontrolador ESP32-PICO con conectividad Wi-Fi para procesar las señales captadas por un sensor óptico MAX30102 (frecuencia cardíaca y oxigenación) y un sensor GSR (respuesta galvánica de la piel) que se analizan en una aplicación FireBase en la nube (Zholdas et al., 2024).

Se identificó el sistema denominado "Whoops", es un aplicativo móvil que monitorea a adultos mayores para detectar caídas y notificar a cuidadores, usa árboles de decisión y datos de la señal de aceleración y movimiento angular obtenidos de los sensores integrados al smartphone (Mrozek et al., 2020). También se identificó un sistema que utiliza aprendizaje profundo basado en árboles de decisión para la detección de patologías, que usa señales EEG obtenido por un sensor ADS1292 que está conectado por Bluetooth a un dispositivo IoT (Muhammad et al., 2019). Finalmente, el tercer sistema permite la supervisión remota de señales de ritmo cardíaco y ECG, con el sensor ADS1292 transmitiendo datos a un smartphone, en donde se ejecuta un aplicativo móvil que usa aprendizaje automático SVM (Pathinarupothi et al., 2019).

2.4.1.3 Discusión

La revisión sistemática cubre varias categorías de relacionadas a las tecnologías utilizadas, pero la validación desarrollada en esta tesis se delimita al dominio de señales fisiológicas asociadas con eventos de cardiopatía, y los resultados son usados en un escenario representativo, sin enfocarse en una generalización hacia otras categorías. El resultado de esta revisión sistemática es un catálogo de aspectos básicos para la construcción de un SMSFCB. Los investigadores pueden utilizar las diferentes señales, sensores, algoritmos de aprendizaje automático y dispositivos de CB presentados aquí para comprender y determinar los enfoques óptimos para su aplicación específica. La confiabilidad y relevancia de los hallazgos se refuerza por la presencia predominante de estudios en revistas de alto cuartil (Q1, Q2) con un 86%. Cada pregunta de investigación se discute a continuación:

a) Señales fisiológicas. Se identificó que las señales fisiológicas cubiertas por los sistemas de monitoreo automático con CB cubren 6 categorías de señales (bioeléctrica, bioacústica, biomagnética, inercial, biomecánica y bio óptica), pero no cubre la categoría de señales fisiológicas químicas que incluyen, por ejemplo, las hormonas, las citocinas y las feromonas. Esto podría explicarse, porque en general los sensores para esta categoría de señales son más complejos, donde en algunos casos puede ser

invasivo o requerir agentes químicos para obtener la información. La categoría de señales fisiológicas más estudiada es la Bioeléctrica, con 41% de los estudios, y las dos señales fisiológicas más estudiadas son ECG (P01) y temperatura (P02), ello puede explicarse por su diversidad de usos (Tabla 13), sin embargo, las señales ECG así como la EEG son más difíciles de capturar y procesar debido a su naturaleza bioeléctrica delicada.

b) Sensores utilizados en el monitoreo de señales fisiológicas. Se identificó que en los estudios están contemplados sensores establecidos, es decir no hay variedad sensores para una señal fisiológica. Ello se podría explicar debido a que los sensores requieren de validación y/o certificación para su uso. Además no hay un fabricante que predomine en todos los sensores, siendo los más usados el AD8232 (señales ECG) y INERTIAL 20948 (para señales de aceleración de giro) utilizados para detección de eventos vinculados comportamiento anómalo y al movimiento de pacientes en actividades diarias respectivamente. Sin embargo, los sensores ADS1198 y AD8232 son los que capturan más señales fisiológicas (ECG, pulso cardiaco, frecuencia respiratoria). El número de estudios muestran que hay una tendencia en el uso de sensores digitales (29) en vez de analógicos (15), ello puede explicarse por la facilidad de interpretación y explotación de los datos.

c) Algoritmos de aprendizaje automático. Se ha identificado una variedad de 37 tipos de algoritmos de aprendizaje automático en 74 estudios seleccionados, que se han usado en los SMSFCB, siendo CNN y SVM los más estudiados con 29.7% y 24% respectivamente. Además, se han identificado 21 usos de estos algoritmos, siendo la detección de caídas, el reconocimiento de actividades físicas y la detección de enfermedades las más estudiadas con 23%, 14.6% y 11.6% de los estudios encontrados para esta sección. Estos intereses se podrían explicar: primero porque las caídas son una de las principales causas de la mortalidad en adultos mayores (Zaid Salah et al., 2022); segundo, las actividades físicas son parte de muchos tratamientos sean correctivo como las terapias de rehabilitación (Yang et al., 2018) o preventivo como el sedentarismo (Abid et al., 2021); tercero, el monitoreo continuo es necesario, en algunos casos, para la detección de enfermedades (Ukil et al., 2019).

d) Dispositivos de computación de borde. Se ha identificado un conjunto de 31 tipos de dispositivos de CB en 76 estudios, donde el teléfono inteligente y el Raspberry PI

son los más usados con un 28% y 18% respectivamente. El teléfono inteligente es utilizado para aplicaciones de detección riesgo de asma (Bhat et al., 2021) y arritmia cardíaca (Devi & Kalaivani, 2019) siendo su fortaleza la potencia de procesamiento, portabilidad y sensores integrados, mientras que el Raspberry Pi se usa para implementación de redes de sensores (El Attaoui et al., 2020) o plataformas de salud con sensores externos integrados (M. A. Khan et al., 2025) (Marquez-Sanchez et al., 2021) debido a que es más personalizable y económica. A pesar de sus ventajas, los dispositivos de CB también presentan limitaciones significativas debido a que algunos tienen ineficiencia energética en proyectos de movilidad cuando se requieren aplicaciones intensivas en cómputo.

e) Sistemas. Los SMSFCB identificados se destacan por su capacidad para utilizar sensores y algoritmos avanzados en dispositivos móviles, detectar caídas en adultos mayores, diagnosticar patologías a través de señales EEG, y supervisar el ritmo cardíaco y ECG de forma remota. Se muestra de forma general como la convergencia de la tecnología móvil y el análisis de datos está siendo aprovechada para mejorar el cuidado de la salud y el bienestar de las personas. Las tecnologías más utilizadas implican el uso de sensores integrados en dispositivos móviles, como smartphones, junto con el uso de algoritmos avanzados, como árboles de decisión y aprendizaje automático SVM.

f) Discusión comparativa. Para la evaluación comparativa se seleccionaron aquellos estudios que abarcan la mayoría de los aspectos considerados en esta tesis, aunque no necesariamente todos de manera simultánea. Se priorizaron trabajos que describen de forma explícita el tipo de dispositivo utilizado, algoritmos, señales, métricas, evaluación en microcontrolador (MCU) y limitaciones de la investigación. Por lo tanto, se excluyen aquellos artículos que abordan algunos de estos elementos de manera parcial aunque y tienen una relación con los criterios considerados en la investigación, mostrados en la Tabla 15.

A partir de la comparación presentada en la Tabla 15, se evidencia que los trabajos más representativos utilizan métricas, como la exactitud, precisión o F1-score, ejecutados en dispositivos como smartphones, circuitos de placa única (Raspberry PI) o microcontroladores. En varios casos, aunque se generan métricas de desempeño del

modelo, no se muestran variables del comportamiento en hardware, tales como consumo corriente eléctrica, tiempo de inferencia o uso de memoria RAM o FLASH.

Por otro lado, en esta investigación se aborda estas comparaciones de varios modelos múltiples algoritmos (SVM, RF, DT, MLP y DNN) en un microcontrolador ESP32, evaluando la exactitud, latencia de inferencia, consumo de corriente y uso de memoria RAM o FLASH. Esta integración metodológica y tecnológica permite superar las limitaciones encontradas en trabajos previos, analizando el comportamiento de los modelos y la eficiencia del uso de recursos de hardware.

Tabla 15. Comparación de trabajos representativos del estado del arte

Autor / Año	Dispositivo	Algoritmo	Señales	Métricas	Evaluación en MCU	Limitaciones
(Zaid Salah et al., 2022)	Arduino nano 33	SVM, KNN, CNN, LSTM	Inercial	Exactitud	Si	No se realiza análisis comparativo de modelos y memoria.
(Devi & Kalaivani, 2019)	Smartphone	SVM	Temperatura , latido de corazón	Precisión, sensibilidad	No	Procesamiento centralizado, dependencia de conectividad
(El Attaoui et al., 2020)	Raspberry Pi	Red neuronal multicapa (MLP), KNN, SVM, RF	Inercial	Exactitud, sensibilidad, especificidad, precisión, F1-score, Error cuadrático medio	No	No hay evaluación de métricas de comportamiento en hardware
(M. A. Khan et al., 2025).	Raspberry Pi	Filtros digitales	ECG, Temperatura , saturación de oxígeno, latido de corazón	Error absoluto medio (En ingles Mean absolute error - MAE)	No	Algoritmos dependientes de valores umbrales para detección del estado de salud. Enfoque no basado en aprendizaje automático.
(Marquez-Sanchez et al., 2021)	Microcontrolador ESP32 (adquisición de sensores) y Raspberry PI (Centralizador)	LSTM, KNN	Temperatura , aceleración, ritmo cardiaco	Exactitud	Si	No evalúa tiempo de inferencia ni métricas relacionadas al hardware
(Velichko, 2021b)	Microcontrolador basado en Arduino UNO	LogNet (NN ligera), ANN	Datos cardiográficos, Ritmo cardiaco	Exactitud, Precisión, Sensibilidad, F1-Score, memoria RAM	Si	Si evalúa la memoria RAM utilizada pero no se analiza métricas de comportamiento del hardware.
(Novac et al., 2020)	Microcontrolador ARM Cortex-M4F (SparkFun Edge)	CNN	Inercial	Exactitud, memoria RAM y FLASH	Si	Compara la memoria RAM y FLASH pero no analiza variaciones del modelo ni métricas de comportamiento del hardware

Propuesta actual	Microcontrolador ESP32	SVM, RF, DT, MLP, DNN	Señales fisiológicas (SPO2, ritmo cardíaco, temperatura, frecuencia respiratoria)	Exactitud, latencia de inferencia, RAM, FLASH, corriente eléctrica	Sí	Las limitaciones anteriores se han implementado en este proyecto de tesis
-------------------------	------------------------	-----------------------	---	--	----	---

2.4.1.4 Conclusiones

Se hizo una revisión sistemática de la literatura de 14805 estudios primarios relacionados con relación a los SMSFCBs, de los cuales se seleccionaron 93 estudios relevantes. El análisis de la información recolectada identifica 37 algoritmos de aprendizaje automático, 31 tipos de dispositivos, 18 señales y 19 tipos de sensores.

Los resultados indican que el consumo de energía y tiempo de inferencia son los más utilizados para la implementación de SMSFCBs pero muy pocos autores lo relacionan con la optimización del tamaño del modelo de aprendizaje automático desplegado en el dispositivo de borde, por lo que sigue siendo un tema para futuras investigaciones. Con respecto a algoritmos de aprendizaje automático el algoritmo SVM es la más utilizada dentro del contexto de monitoreo de señales fisiológicas representando un total del 22% de los estudios. Esto se debe a que el algoritmo que lo implementa está ampliamente difundido en diferentes sistemas de software y hardware.

A diferencia de otras revisiones de literatura (V. Patel et al., 2021b), que dan a conocer los aspectos relacionados a las variables medidas y dispositivos usados para la adquisición de datos en soluciones de IoT, este contempla los aspectos de análisis de señales fisiológicas procesadas y analizadas en el mismo dispositivo de adquisición de datos. Por lo que esta investigación se encuentra limitada en el aspecto que sólo se han revisado artículos indexados relacionados a señales fisiológicas sin considerar su integración en soluciones de computación en niebla o nube. Como trabajo futuro se sugiere investigar las herramientas de software en nube que se integran con los SMSFCBs.

2.4.2 Trabajos relacionados

a) Un sistema basado en correa de pecho para la monitorización del electrocardiograma. Este estudio aborda el problema de la y movilidad limitada de los sistemas de monitorización de ECG durante la actividad física, proponiendo como

desarrollar un sistema portátil para capturar señales de alta calidad sin restringir el movimiento (X. Zhang, Zhan, et al., 2025). Para su desarrollo, se creó un Sistema de monitorización de ECG con correa de pecho (CEMS) utilizando hilo de poliamida recubierto de plata como bioelectrodo textil integrado en una banda elástica. El sistema embebido en el CEMS recoge las señales, las filtra (con un filtro de rechazo y un filtro paso banda) y digitaliza, usando un microcontrolador (MCU) para el procesamiento inicial de la señal. Además, un módulo Bluetooth permite la visualización de datos en tiempo real y el almacenamiento de datos en memoria flash para análisis posterior.

El resultado principal es que se garantiza la captura de señales de ECG de alta calidad, incluso con la presencia de ruido ambiental, gracias a su eficiente filtrado. Este sistema no solo rastrea la señal cardíaca sino que también permite la evaluación de la intensidad del ejercicio a través de un modelo de clasificación basado en una red neuronal completamente conectada. Este algoritmo embebido o borde de clasificación demostró una precisión del 98.7% para la intensidad de carrera después de 100 épocas de entrenamiento.

b) Volante inteligente: Diseño de un sistema de monitorización de la salud del conductor no invasivo basado en el Internet de las Cosas (IoMT) para mejorar la seguridad vial. Este artículo aborda el problema del creciente número de accidentes de tráfico causados por condiciones médicas que requieren una solución para mejorar la seguridad vial (M. A. Khan et al., 2025). El objetivo fue proponer y validar un sistema de monitoreo de salud del conductor (DHMS) no invasivo y basado en el Internet de las cosas médicas (IoMT) para vigilar signos vitales y detectar condiciones de salud anormales.

El desarrollo del sistema se estructuró en cuatro bloques: un sistema embebido (con sensores en el volante conectados a una placa como Raspberry Pi 4), computación de borde (edge computing), computación en la nube y una interfaz de usuario. El bloque de computación de borde se encarga del preprocesamiento y análisis de los datos de los sensores utilizando múltiples algoritmos y filtros, lo que ayuda a reducir los problemas de latencia y pérdida de paquetes. Los resultados experimentales demostraron un rendimiento adecuado del DHMS, con sus mediciones de ECG, saturación de oxígeno, frecuencia cardíaca y temperatura corporal.

c) Desarrollo de un monitor portátil para identificar niveles de estrés mediante internet de las cosas. El objetivo de este proyecto fue desarrollar un dispositivo wearable con IoT capaz de identificar los niveles de estrés en tiempo real (Zholdas et al., 2024). Para ello se utilizó la plataforma de computación de borde ESP32-PICO como monitor, integrando sensores como el MAX30105 (ritmo cardíaco/oximetría) y sensores de respuesta galvánica de la piel (GSR). Este monitor captura las señales fisiológicas relevantes y las transmite a la plataforma Firebase mediante Wi-Fi.

Las pruebas mostraron valores coherentes, con la frecuencia cardíaca entre 76 y 97 latidos por minuto y el GSR entre 71% y 79%, observándose desviaciones de 5% a 10% para los niveles de estrés. El dispositivo se considera adaptable para la gestión del estrés en escenarios, como oficinas y centros de salud, ya que permite la monitorización continua de la conductividad de la piel además de las tasas cardíaca y respiratoria.

d) Un método para el análisis de datos médicos que utiliza LogNNet para sistemas de apoyo a la toma de decisiones clínicas y borde informática en el cuidado de la salud. Este trabajo (Velichko, 2021b) propone un método para implementar inteligencia artificial en microcontroladores de bajo consumo con restricciones de memoria usando algoritmos para aplicaciones con redes neuronales. Se presenta un método para analizar datos médicos basados en la red neuronal LogNNet, que resuelve los problemas de clasificación y calcula los factores de riesgo de una enfermedad en pacientes. El sistema desarrollado permite que los dispositivos con una pequeña cantidad de memoria RAM (2–32 kB) con un procesador de poca velocidad, pueda recibir información diversa sobre el estado de un paciente desde diferentes sensores. Estos sensores pueden ser sensores portátiles, que pueden medir la frecuencia cardíaca, la presión arterial, la temperatura corporal y los niveles de glucosa, y se instalan sensores externos en los puntos de control.

e) Hacia el reconocimiento de actividad humana sin supervisión en unidades de microcontroladores. El presente trabajo (Novac et al., 2020) menciona que para aplicaciones de reconocimiento de actividad humana, las redes neuronales artificiales han demostrado su capacidad para proporcionar un mejor rendimiento en comparación con otros métodos de aprendizaje automático. El objetivo de este trabajo es estudiar el rendimiento de la clasificación y las limitaciones de implementación en sistemas

embebidos, para lo cual se utilizan datos de un acelerómetro y giroscopio de los teléfonos inteligentes.

Las redes neuronales utilizadas en esta investigación se implementan en la placa SparkFun Edge, la cual tiene un microcontrolador basado en ARM Cortex-M4F de bajo consumo de energía. Los modelos de aprendizaje supervisado se entrenan en TensorFlow 2.0 en una computadora y luego se migran al hardware usando con TensorFlow Lite para microcontroladores. Como conclusión se tiene que la implementación de redes neuronales en un microcontrolador es viable en el contexto del reconocimiento de la actividad humana.

f) FANN-on-MCU: un kit de herramientas de código abierto para la inferencia de redes neuronales energéticamente eficientes en el borde del Internet de las cosas. En este trabajo (X. Wang et al., 2020) se describe como el creciente número de dispositivos inteligentes de bajo consumo en Internet de las cosas se combina con el concepto de computación de borde para mover parte de la inteligencia hacia el borde de la red. La ejecución de algoritmos de aprendizaje automático en dispositivos de hardware con recursos limitados es un problema en términos de hardware y de firmware para realizar las implementaciones algorítmicas. Este artículo presenta un conjunto de herramientas de código abierto construido sobre la biblioteca de redes neuronales artificiales rápidas (FANN) para ejecutar redes neuronales ligeras y eficientes en microcontroladores basados tanto en la serie ARM Cortex-M.

g) Robustecimiento de la implementación de modelos tinyML para Mini-Vehículos autónomos. Este trabajo describe como los vehículos autónomos han mejorado rápidamente gracias a los avances del aprendizaje profundo (de Prado et al., 2021). Se propone en este artículo un flujo de aprendizaje de circuito cerrado para mini vehículos de conducción autónoma que incluye para lo cual se utiliza una familia de herramientas de aprendizaje automático ligero (tinyCNN), compactos y de alto rendimiento. Por lo tanto, se presenta un predictor en línea que puede elegir entre diferentes modelos de tinyCNN en tiempo de ejecución que minimiza el consumo de energía de la inferencia. Para ello se utiliza un microcontrolador (MCU) con ejecución de procesos en paralelo basado en tecnología RISC-V y de baja potencia, para cumplir con los requisitos de inferencia en tiempo real.

h) Inteligencia computacional ligera para IoT Salud Monitoreo de vehículos todoterreno: selección de Mutación mejorada a escala logarítmica GA Structured ANN. El uso de dispositivos de borde, como teléfonos inteligentes, permite implementar soluciones con capacidad de computación a nivel local pero aún no se ha desarrollado a nivel comercial (N. Gupta et al., 2022). Con la creciente demanda del usuario por tecnologías económicas y fáciles de usar, se motiva a cambiar sensores costosos y complejos a económicos. En este artículo, se presenta la metodología de inteligencia computacional híbrida para desarrollar una solución de inteligencia artificial habilitada para dispositivos de borde para salud, monitoreo y diagnóstico (HM&D) de los vehículos todoterreno, utilizando micrófonos económicos como sensores.

i) Detección de anomalías de datos en el borde de los sistemas de IoT generalizados. Este artículo muestra el uso de métodos estadísticos para identificar anomalías de datos en dispositivos de borde que se conectan a servicios en la nube (Mishra et al., 2021). Se estudia dos tipos de anomalías: anomalías de sesgo de datos y anomalías de corte de sensores. El enfoque principal en este documento es presentar y evaluar métodos de control estadístico como el método T2 transformado para anomalías de sesgo de datos siendo robusto cuando existen valores de datos faltantes. Ambos métodos se pueden utilizar en el dispositivo de borde, demostrando que son muy precisos y detectan anomalías casi de inmediato en cuanto ocurren. En los experimentos de simulación, se evalúa la efectividad de usar gráficos de control, el monitoreo de procesos estadísticos, a ambos tipos de anomalías. Los resultados muestran que el uso de gráficos de control como métodos estadísticos para la detección de anomalías en sistemas IoT brinda un alto rendimiento en precisión y potencia (probabilidad de detectar la anomalía).

j) Sistema de atención médica inteligente basado en aprendizaje profundo para la detección de molestias del paciente en el Borde de Internet de las cosas. En este artículo, se presenta un sistema de monitoreo y detección no invasivo de malestar del paciente utilizando un algoritmo basado en aprendizaje profundo (Ahmed et al., 2021). El sistema usa una cámara IP para detectar el movimiento y la postura del cuerpo del paciente. El método Mask-RCNN se emplea para la extracción de diferentes puntos del cuerpo del paciente. Para analizar el malestar del paciente, se mide la información obtenida de las coordenadas de puntos clave detectados, la distancia y el umbral

temporal para asociar los movimientos con condiciones normales o de malestar. La posición y la postura del cuerpo del paciente se controlan continuamente, en función del nivel de comodidad y malestar.

l) Monitoreo y clasificación de arritmias en tiempo real basado en Edge Computing y DNN. Este trabajo muestra una forma para incorporar una etapa de inteligencia para detectar arritmias, una afección cardíaca en dispositivos IoT de Borde (Edge) (M. Liu et al., 2021). Se propone un algoritmo de clasificación basado en el modelo de red neuronal convolucional intrapaciente y el modelo de red residual de atención entre pacientes para identificar automáticamente el tipo de arritmia en el borde. El proceso de algoritmo de diagnóstico de arritmias basado en la red neuronal profunda se puede dividir en los siguientes pasos: preprocesamiento de datos de ECG y clasificación de arritmias. Los resultados de la investigación tienen un significado positivo para mejorar la precisión del diagnóstico de arritmias, pero limitado por la pequeña cantidad de datos. Además, se indica que es necesario aumentar el número de conjuntos de datos e intentar combinar otras estructuras de redes neuronales para explorar el algoritmo de diagnóstico de arritmias.

CAPÍTULO 3: METODOLOGÍA

3.1 Tipo y diseño de la investigación

3.1.1 Tipo de investigación

El enfoque de la investigación es cuantitativa debido a que se necesita recolectar datos numéricos determinando patrones de comportamiento y permitiendo probar las hipótesis mediante un análisis estadístico (Kothari, 2004) (Rodríguez Rodríguez et al., 2021) (Hernández Sampieri & Mendoza Torres, 2018). Esta investigación tiene un enfoque cuantitativo, porque se obtiene un conjunto de observaciones cuantificables de actividades relacionadas a generar e implementar algoritmos de aprendizaje automático en nodos sensores. Además, se obtienen datos sobre el uso de recursos de hardware y energía en los nodos sensores, por medición directa con instrumentos externos, almacenamiento y lectura de archivos de texto de forma local y/o registros de datos enviados a servicios web remotos para la prueba de hipótesis.

El alcance o nivel de la investigación explicativa identifica relaciones causales entre las variables y mediante un proceso estadístico inferencial se valida la hipótesis (Caballero Romero, 2014) (Dahnke et al., 1986). Por lo tanto, esta investigación es explicativa ya que plantea la propuesta de implementar un sistema de computación de Borde con algoritmos de aprendizaje automático y explicar cómo afecta al monitoreo de señales fisiológicas de forma experimental para demostrar la hipótesis mediante estadística inferencial.

Además, la investigación también es de alcance exploratorio porque se ha realizado una revisión sistemática de la literatura obteniendo bibliografía identificando la relevancia al relacionar las variables de la investigación. La investigación exploratoria permite organizar y consolidar la información de los estudios y su relación con el problema de investigación, identificando tópicos y áreas de aplicación (Hernández Sampieri et al., 2014).

3.1.2 Diseño de investigación

En la investigación experimental se altera una o varias variables al mismo tiempo, en un ambiente controlado con el objetivo de describir el modo y la causa de cómo se produce un fenómeno (Caballero Romero, 2014) (Palella & Martins, 2012). Esta investigación corresponde a un estudio cuantitativo de tipo cuasiexperimental, dado

que se realiza la manipulación de modelos de aprendizaje automático para analizar su impacto en las métricas de comportamiento en hardware, se realizan pruebas de hipótesis, un muestreo no probabilístico y no requieren necesariamente experimentos controlados con grupos aleatorizados.

En el contexto de esta investigación se realiza la manipulación de variables independientes y la observación de sus efectos sobre variables dependientes. Es por esto que el diseño es cuasiexperimental al utilizar la técnica computacional para evaluar el comportamiento de modelos de aprendizaje automático en hardware, y no se realiza el diseño de un software o artefacto, lo cual es propio de la ciencia del diseño.

3.2 Unidad de análisis

Esta investigación utiliza señales fisiológicas de personas como temperatura, frecuencia respiratoria, pulso cardíaco, electrocardiográficas, saturación de oxígeno e inerciales. Estas señales capturadas están representadas en series de tiempo y permite aplicar modelos de aprendizaje automático para detección de patrones utilizando nodos sensores con recursos de hardware reducido utilizando tecnologías de computación de borde. En este caso se considera lo siguiente:

- Unidad de análisis. Señales fisiológicas asociadas a cardiopatía en dispositivo embebido.

3.3 Población de estudio

Corresponden a la base de datos de señales fisiológicas de personas encontradas en la literatura. Por lo tanto, se realiza la selección de hasta dos bases de datos para adaptar a un formato compatible a los algoritmos de aprendizaje durante su entrenamiento.

Para realizar la generación de un conjunto de datos se utiliza dos fuentes de información. La primera de ellas a partir del del conjunto de datos de enfermedades cardiovasculares (CVD, por sus siglas en inglés) con dos 2 estados de salud en sus categorías (Louis Uzoegwu Farah, 2023)(Chidozie, 2022). Mientras que la otra fuente de datos fue generada de manera sintética con los rangos de parámetros descritos en NEWS-2 (3 estados de salud) (Huespe et al., 2022a)(Liao et al., 2020a). Posteriormente estos dos resultados son adaptados e integrados para ser utilizados en procesos de clasificación de 3 niveles del estado de la salud.

- Población diana: Bases de datos de señales fisiológicas.
- Población accesible: Bases de datos de señales fisiológicas asociadas a cardiopatía construidos en el proceso de investigación

3.4 Tamaño de la muestra

En este proyecto de investigación se emplea el muestreo no probabilístico intencional (por conveniencia) (Vara Horna, 2010) (Kothari, 2004) (Hernández Sampieri et al., 2014), debido a que las muestras seleccionadas serán aquellas que correspondan a las bases de datos de señales fisiológicas relacionadas a cardiopatía para su prueba en dispositivos de hardware construidos.

3.5 Selección de muestra

La selección de muestras se realiza en base a la población universo descrita anteriormente. Se considera en primer lugar, la selección de la población diana teniendo en cuenta los elementos sobre los cuales se van a generalizar los resultados de la investigación (Offarm, 2011) (Arias-Gómez et al., 2016). La población accesible se obtiene considerando aquel subconjunto de la población diana sobre los cuales podemos recabar información y realizar la experimentación (Arias, 2016).

Para las muestras se selecciona sólo aquellos elementos de la población accesible que consisten en dispositivos de hardware desarrollados durante el trabajo de investigación a incluir en la experimentación (Vara Horna, 2010).

3.6 Técnicas de recolección de datos

Para la recolección de datos relacionadas a las métricas de comportamiento en hardware, se utilizan dispositivos embebidos integrados con modelos de aprendizaje automático. Según Vara (2010) y Caballero (2014) existen técnicas como revisión documental y observación de campo experimental, las cuales se usan de la siguiente manera:

- Análisis documental: Para la recolección de datos se considera el análisis realizado con la revisión sistemática de literatura el cual ayuda a la investigación a profundizar sobre las herramientas tecnológicas, aprendizaje automático y bases de datos públicas de señales fisiológicas.

- Observación experimental. Se realiza la evaluación de los modelos de aprendizaje automático en la etapa de entrenamiento para obtener métricas acerca de exactitud, consumo de corriente eléctrica, tiempo de inferencia y uso de memoria RAM y FLASH. Además, se realiza pruebas de laboratorio acerca de los métodos de clasificación sobre señales fisiológicas, para obtener datos de los recursos de hardware utilizados y el comportamiento de los algoritmos.

3.6.1 Instrumentos

Para el desarrollo de la investigación, los instrumentos permiten obtener métricas durante el entrenamiento de modelos de aprendizaje automático y su ejecución en un dispositivo embebido. Estos instrumentos consideran aspectos como el software de desarrollo, herramientas para la evaluación en dispositivos de hardware y bases de datos fisiológicos utilizados en el análisis.

En la etapa de entrenamiento y validación de los modelos, se utiliza el lenguaje de programación Python, usando librerías como Scikit-learn para implementar y evaluar modelos basados en algoritmos como máquinas de soporte vectorial (SVM), Random Forest, y árboles de decisión. Para el diseño y entrenamiento de redes neuronales multicapa (MLP) y redes profundas (DNN), se recurre a TensorFlow y Keras, herramientas que permiten convertir los modelos entrenados al formato TensorFlow Lite compatible con hardware embebido. Para el despliegue de los modelos en hardware, se utiliza como plataforma el microcontrolador ESP32, donde se cargan versiones optimizadas de los modelos mediante usando las librerías de MicroMLgen y TensorFlow Lite.

Como fuente de señales fisiológicas, se emplean bases de datos públicas de PhysioNet, CVD (Cardio-Vascular Diseases) (Louis Uzoegwu Farah, 2023) e información de señales características descritas en NEWS-2(National Early Warning Score) (Huespe et al., 2022a). Estas contienen datos de frecuencia cardíaca, saturación de oxígeno (SpO2), temperatura corporal y frecuencia respiratoria, que son preprocesadas con Python para usarse durante el entrenamiento y validación. Además, se utilizan como entrada para las pruebas en el sistema embebido.

Además, se usan instrumentos integrados y externos al dispositivo embebido para obtener las métricas de evaluación. Los resultados de clasificación de los modelos y su comparación con etiquetas de prueba son registrados en la memoria del dispositivo

embebido y se usan para evaluar la exactitud. Además, se usa un multímetro digital en serie con el ESP32 para determinar el consumo energético durante el funcionamiento del dispositivo. El tiempo de inferencia se calcula directamente en el programa del dispositivo embebido utilizando funciones de temporización, midiendo el lapso exacto entre la entrada de la señal y la salida del resultado del modelo. Por otro lado, el uso de memoria (RAM y Flash) se obtiene mediante reportes generados durante la compilación en entornos de Arduino IDE.

3.7 Técnicas de análisis e interpretación de datos

Para realizar la validación de las hipótesis, se emplean métodos estadísticos para comparar el rendimiento de distintos modelos de aprendizaje automático embebidos en dispositivos de hardware. En primer lugar, se debe verificar los supuestos de normalidad y homogeneidad de varianzas mediante las pruebas de Shapiro-Wilk y Levene, respectivamente en todas las pruebas realizadas con las diferentes métricas obtenidas.

Dependiendo del cumplimiento de estos supuestos, se deben aplicar pruebas paramétricas como ANOVA o pruebas no paramétricas como Kruskal-Wallis. Esto se realiza con la finalidad de determinar las diferencias significativas entre los algoritmos respecto al consumo de corriente, tiempo de procesamiento, uso de memoria y exactitud de clasificación.

Cuando se identifican diferencias significativas, se aplican pruebas de comparaciones múltiples, como la prueba de Tukey, Bonferroni o pruebas post-hoc no paramétricas como la prueba de Dunn, obteniendo los modelos con un mejor rendimiento en relación con cada métrica evaluada. Esto permite analizar estadísticamente la influencia de la técnica computacional embebida en el monitoreo eficiente de señales fisiológicas relacionadas a cardiopatía.

El análisis de los resultados usando procesos estadísticos es coherente con un diseño cuasiexperimental, ya que permite contrastar las hipótesis mediante la comparación de los modelos de aprendizaje automático en hardware embebido. Además, el análisis de los resultados no se orienta a la construcción de un artefacto, producto o sistema lo cual se relaciona a la ciencia del diseño.

3.8 Hipótesis

3.8.1 Hipótesis general

La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía, permite seleccionar modelos eficientes en condiciones de aplicación con dispositivos embebidos.

3.8.2 Hipótesis específicas

- La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de exactitud.
- La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de consumo de corriente eléctrica.
- La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de tiempo de inferencia.
- La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de memoria RAM y FLASH.

3.9 Identificación y Operacionalización de variables

3.9.1 Identificación de variables

Las variables seleccionadas son susceptibles a ser cambiadas y su variación se puede medir. En este caso se tienen variables dependiente e independiente como se describen a continuación y mencionadas en la Tabla 16:

Variable independiente

- Algoritmos de aprendizaje automático

Variable dependiente

- Eficiencia en el monitoreo de señales fisiológicas asociadas a cardiopatía

Variable interviniente

- Señales fisiológicas

Tabla 16. Operacionalización de las variables de la investigación

Variables	Definición Conceptual	Definición Operacional	Dimensiones	Indicadores
Independiente: Algoritmos de aprendizaje automático	Métodos computacionales para aprender automáticamente a partir de datos, sin necesidad de ser programado explícitamente para cada tarea.	Modelos computacionales implementados en dispositivos embebidos que, a partir del entrenamiento con datos de señales fisiológicas	Algoritmos: a) SVM b) Bosques aleatorios c) Árboles de decisión d) Redes neuronales multicapa e) Redes neuronales convolucionales	Indicador común: Exactitud del modelo a) Parámetro C y gamma b) Número de árboles, Muestras por hoja c) Estrategia de división, profundidad máxima d) Capas ocultas, función de activación e) Capas, tamaño del filtro
Dependiente: Eficiencia en el monitoreo de señales fisiológicas asociadas a cardiopatía	Capacidad del sistema para procesar señales fisiológicas orientada a la evaluación, clasificación y visualización del estado de salud de un paciente	Medición del desempeño del sistema que ejecuta tareas de monitoreo evaluando métricas de eficiencia	a) Exactitud de clasificación en dispositivo embebido b) Consumo energético c) Tiempo de inferencia d) Uso de memoria RAM e) Uso de memoria FLASH	a) Exactitud (Accuracy = $(TP + TN) / (TP + TN + FP + FN)$) b) Corriente promedio (mA). $I_{prom} = (\sum I_i) / N$, donde I_i es la corriente medida en cada muestra durante ejecución c) Tiempo de inferencia (ms). $T_{inf} = (\sum t_i) / N$, donde t_i es el tiempo de inferencia por muestra. d) Memoria RAM usada (KB) por algoritmo e) Memoria FLASH usada (KB) por algoritmo

3.10 Matriz de consistencia

Está conformada por las siguientes columnas, para los aspectos generales: problema general, objetivo e hipótesis generales, para el caso de los aspectos específicos: problemas específicos, objetivos específicos, hipótesis específicas, variables de las hipótesis específicas y técnicas de recolección de datos para cada una de las variables (Tabla 17).

Tabla 17. Matriz de consistencia

Problema	Objetivos	Hipótesis	Variables	Metodología
Problema General: ¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía permite la selección de modelos eficientes?	Objetivo general: Elaborar y validar una técnica computacional en dispositivos embebidos usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía, que permita seleccionar modelos eficientes con los mejores indicadores de comportamiento en hardware.	Hipótesis general: La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas asociadas a cardiopatía, permite seleccionar modelos eficientes en condiciones de aplicación con dispositivos embebidos.	Variable independiente • Algoritmos de aprendizaje automático	Tipo de investigación: Enfoque Cuantitativa con nivel de investigación explicativa. Alcance exploratorio (Revisión sistemática de literatura) Diseño de investigación: Cuasiexperimental Población: Bases de datos de señales fisiológicas públicas asociadas a cardiopatía Muestra: No probabilística
Problemas específicos	Objetivos específicos	Hipótesis específicas	Variable dependiente • Eficiencia en el Monitoreo de señales fisiológicas asociadas a cardiopatía	
¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de exactitud?	Determinar la exactitud alcanzada por la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.	La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de exactitud.		
¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de consumo de corriente eléctrica?	Determinar el consumo de corriente eléctrica asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.	La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de consumo de corriente eléctrica.		
¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de tiempo de inferencia ?	Determinar el tiempo de inferencia asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.	La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de tiempo de inferencia.		
¿De qué manera la técnica computacional en dispositivos embebidos usando aprendizaje automático contribuye en la selección de modelos eficientes considerando el indicador de memoria RAM y FLASH?	Determinar el uso de memoria RAM y FLASH asociado a la ejecución de la técnica computacional, mediante la evaluación de modelos de aprendizaje automático en el dispositivo embebido para el monitoreo eficiente de las señales fisiológicas.	La técnica computacional usando aprendizaje automático para el monitoreo de señales fisiológicas, permite seleccionar modelos eficientes considerando el indicador de memoria RAM y FLASH.		

CAPÍTULO 4: DISPOSITIVO EMBEBIDO PARA MONITOREO DE SEÑALES FISIOLÓGICAS

El presente capítulo describe el diseño y construcción del dispositivo embebido. Este componente conforma la plataforma de hardware donde se valida la técnica computacional mediante la evaluación de modelos de aprendizaje automático. Para realizar las actividades de la investigación se divide el desarrollo en dos etapas (Figura 13), donde en el caso de este capítulo, se describen las actividades realizadas para la etapa 1.

- En el caso de la Etapa 1 se inicia con los procesos relacionados al dispositivo embebido para su construcción con conectividad inalámbrica (nodo sensor con tecnologías de computación de borde). Posteriormente se realiza la preparación de los datos a partir de los valores obtenidos de las señales fisiológicas asociadas a cardiopatía.
- En el caso de la etapa 2, su desarrollo se describe en el capítulo 5 y muestra la técnica computacional que permite la evaluación de modelos de aprendizaje automático para la clasificación del estado de salud. En este caso se usan métodos de clasificación para su despliegue en el dispositivo embebido (nodo sensor) y evaluar parámetros de comportamiento durante su funcionamiento.

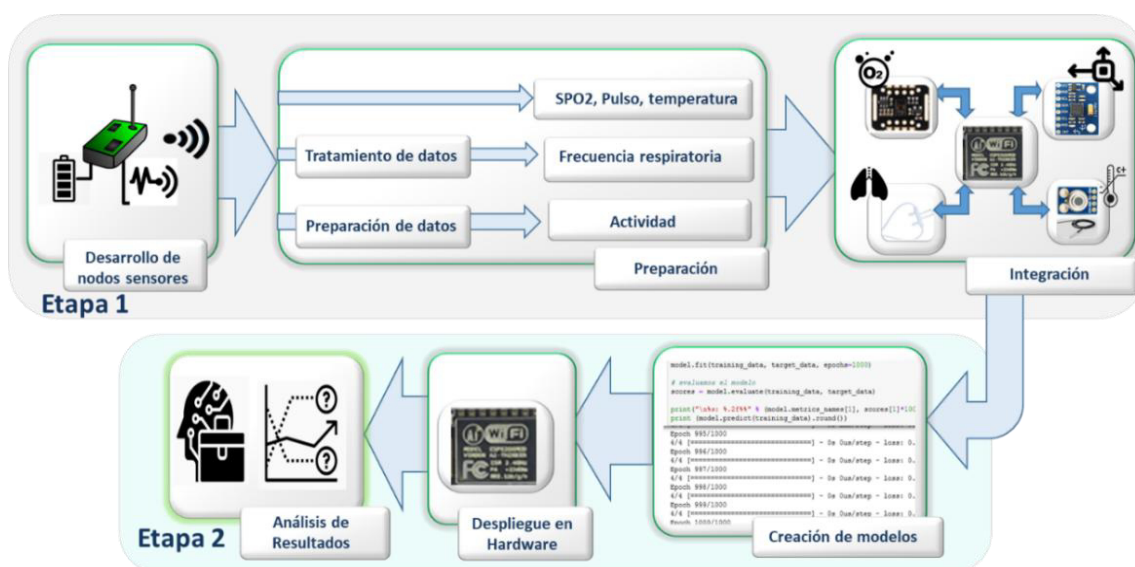


Figura 13. Flujo de procesos para las etapas identificadas en la investigación

4.1 Definición de la propuesta

En el desarrollo de la etapa 1 se utiliza tecnología de monitorización remota mediante dispositivos embebidos (nodo sensor) portátiles con transmisión inalámbrica hacia centros de datos en Internet. Su uso permite una supervisión continua, monitoreo y alerta automática, contribuyendo a una reacción preventiva y proactiva para el cuidado de pacientes de riesgo.

El protocolo de uso del sistema se inicia cuando el dispositivo embebido o nodo sensor, se coloca sobre el paciente realizando las capturas de datos de forma automática, realizando transmisiones de datos en ventanas de tiempo (Figura 14).

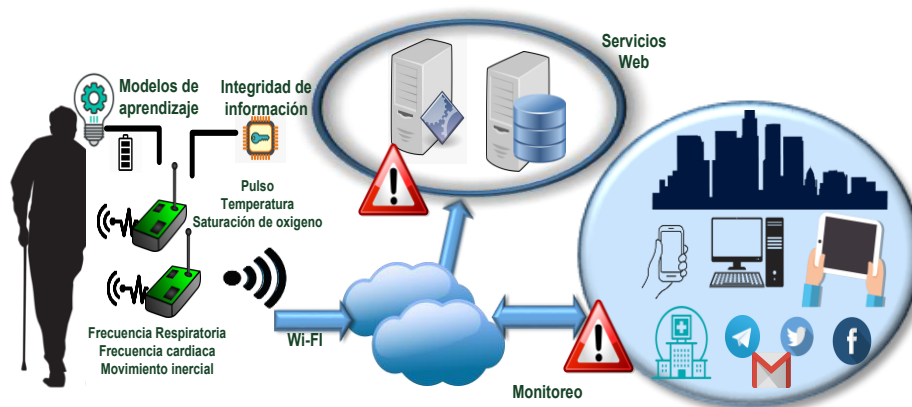


Figura 14. Propuesta de implementación de nodos sensores para el monitoreo

4.1.1 Metodología para el desarrollo de la solución

Se evalúa el funcionamiento del dispositivo embebido y la validación de integración inicial de modelos de inteligencia artificial. Es por ello que en esta etapa 1 se describe el desarrollo del dispositivo, mediante la integración de dos componentes de hardware diferentes.

4.1.2 Criterios de diseño

Para el desarrollo de estas actividades se consideran los siguientes criterios:

- **Criterio 1:** Envío de datos desde la superficie corporal exige que se utilice módulos inalámbricos.
- **Criterio 2:** Implementar la electrónica correspondiente para la adaptación de las señales.

- **Criterio 3:** Aplicación remota que este activa durante todo momento y almacene la información.
- **Criterio 4:** Implementación de un protocolo con estado como MQTT (Message Queing Telemetry Transport).

4.2 Etapa de desarrollo de dispositivo embebido

4.2.1 Características y Selección de tecnologías

De forma esta etapa tiene las siguientes características:

- **Adquisición y procesamiento de datos.** En el dispositivo embebido, las señales fisiológicas son capturadas realizando un procesamiento para eliminar el ruido relacionado a señales externas.
- **Control de bajo consumo.** El consumo de corriente se controla mediante interruptores digitales que desactivan todos los componentes cuando no se transmite información.
- **Transmisión de datos.** Se envían los datos inalámbricamente hacia un dispositivo centralizador o gateway estableciendo comunicación con los servicios en la nube.
- **Acceso remoto a la información.** Un servicio remoto (aplicación web) recibe los datos y luego se envía los datos a la interfaz de monitoreo web.

En la etapa 1 se desarrollan dos tipos de dispositivos embebidos los cuales se construyen con componentes comerciales de bajo consumo de energía y alimentación con baterías y el control de la etapa de comunicación. Posteriormente estos dispositivos se integran en uno sólo. Ambos dispositivos varían solo en los sensores que se integran y están compuesto por los siguientes bloques (Figura 15.):

- Bloque de sensores.
- Bloque procesamiento y control
- Bloque de energía.
- Bloque de comunicación.

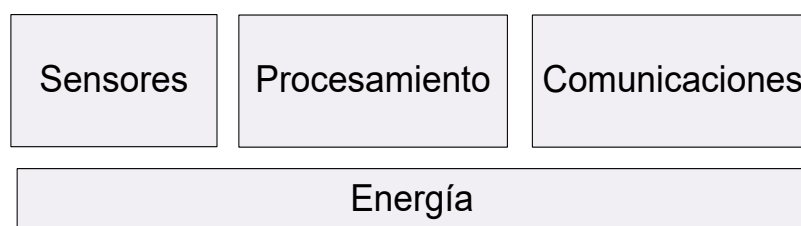


Figura 15. Bloques del nodo sensor inalámbrico.

Los dos dispositivos de hardware (NS1 y NS2) tienen las siguientes características generales básicas en un esquema de despliegue mostrado en la Figura 16:

- Autonomía del dispositivo embebido.
- Comunicaciones. Comunicación WiFi
- Sensores: Sensor de saturación de oxígeno, pulso, ECG, movimiento inercial, frecuencia respiratoria, frecuencia cardiaca y temperatura corporal.
- Modo de uso
 - Al encender cada nodo sensor automáticamente se realiza transmisiones periódicas hacia servicios Web.
 - Periodo de envío ajustable, con ventanas de tiempo.

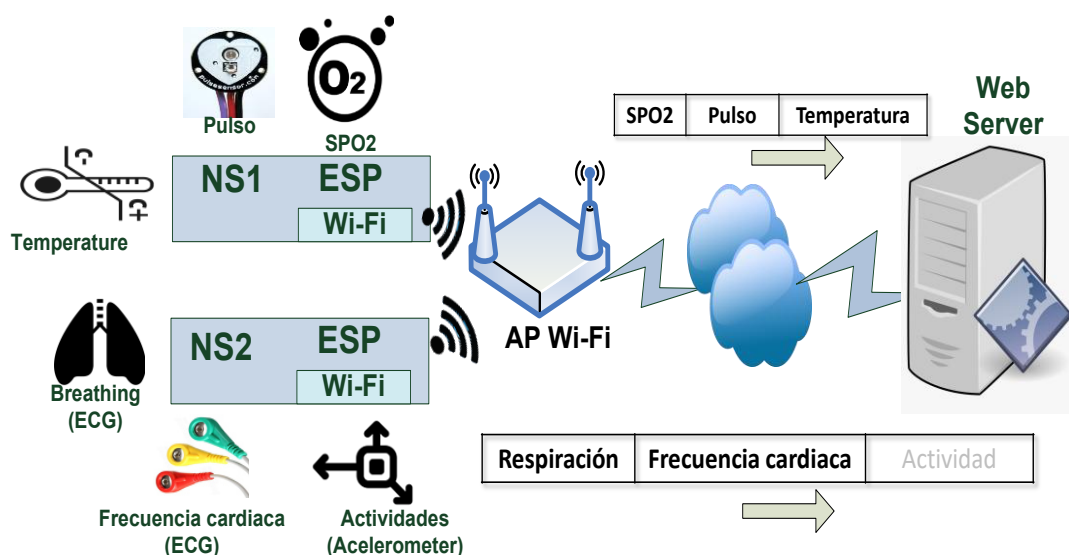


Figura 16. Diagrama de interacción

Para la adquisición de los datos de sensores se considera un rango de frecuencias para el monitoreo de la salud mostrados en la Tabla 18

Tabla 18. Señales fisiológicas relacionadas a cardiopatía (Meza, 2013)

Parámetros	Rango de frecuencias
ECG	0.5 Hz – 100 Hz
Temperatura del cuerpo	1 HZ
Latidos del corazón	40 Hz
Electrocardiográfica	10 Hz – 5 kHz
Actividad Humana	0 Hz – 20 Hz

4.2.2 Definición de componentes

a) Dispositivo embebido - Nodo sensor 1. Para realizar la implementación del dispositivo embebido correspondiente al nodo sensor 1 (NS1) es necesario definir los tipos de sensores, microcontrolador, dispositivo de carga de batería y módulo de comunicación inalámbrico. Está conformado por una etapa de energía controlada por el dispositivo TP4056, el cual sirve para cargar una batería, y generar controlar la salida de energía que va hacia el regulador de 3.3V TC1262. Este regulador es de bajo Low-Dropout y puede estabilizar tensiones de hasta 3.5Voltios para generar 3.3V. Este valor de tensión sirve para energizar el módulo ESP32 y el dispositivo MAX30105 (Figura 17). A continuación, se describen los elementos de cada bloque:

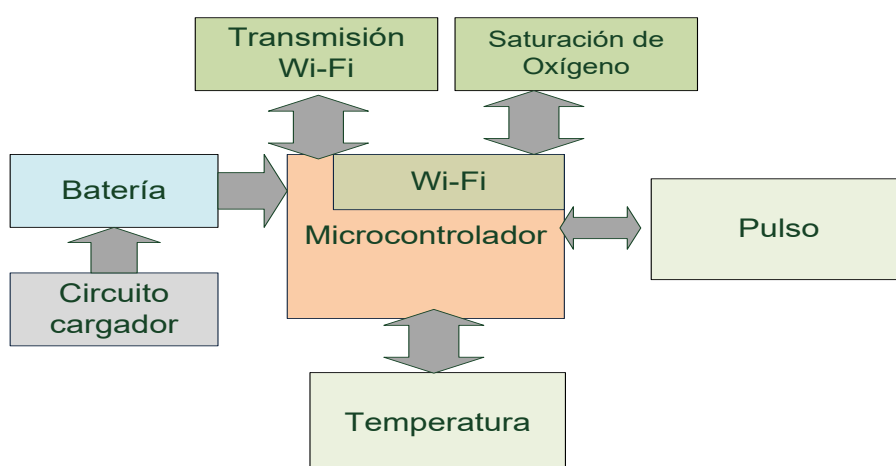


Figura 17. Estructura de relaciones entre los componentes del NS1

i) Elementos del Bloque de sensores. Este bloque está conformado por los siguientes parámetros:

- **Concentración de oxígeno.** Para adquirir este parámetro se usa el circuito integrado MAX30105 el cual es compatible con un nivel de tensión de 3.3V usando una librería compatible con el módulo ESP.
- **Pulso.** Corresponde al latido de las arterias que se provoca cuando el corazón es contraído. El valor de pulso es adquirido también a través del mismo módulo MAX30105 y su valor se infiere de la variabilidad de los picos de la señal analógica capturada.
- **Temperatura.** Este parámetro se obtiene mediante el módulo sensor MAX3010 el cual integra un sensor de temperatura con una resolución inherente de 0,0625°C. Otro sensor de temperatura complementario utiliza el protocolo 1-Wire para comunicarse, que necesita solo un pin de datos para comunicarse (Ramesh Saha et al., 2021) (SARI & KIRINDI, 2019).

ii) **Elementos del Bloque procesamiento y control.** Se encarga de almacenar y/o procesar las señales de temperatura y saturación de oxígeno. En el diseño circuital de este bloque se realiza la polarización del microcontrolador ESP32, que tiene chips SoC de bajo costo y consumo de energía, comunicación Wi-Fi y Bluetooth. Dispone de un microprocesador Tensilica Xtensa LX6 (Maier et al., 2017) y utiliza el protocolo I2C para la conexión con el módulo SPO2 MAX30100 (Zim, 2021).

iii) **Elementos del Bloque de energía.** Se utiliza la etapa de regulación de tensión basada en un Circuito integrado TC1262-3.3. Además, se complementa con el circuito cargador TP4056 para baterías de Ion-Litio de 3.7V (Gowda et al., 2020).

iv) **Elementos del Bloque de comunicación.** El dispositivo dispone de módulos de comunicación WiFi y Bluetooth. En el caso de este prototipo se usa la comunicación WiFi con protocolos de comunicación Websocket y HTTP.

Los componentes seleccionados se muestran en la Figura 18 donde el dispositivo ESP32 es el encargado de coordinar todas las acciones de los componentes involucrados considerando las conexiones de la Figura 19.

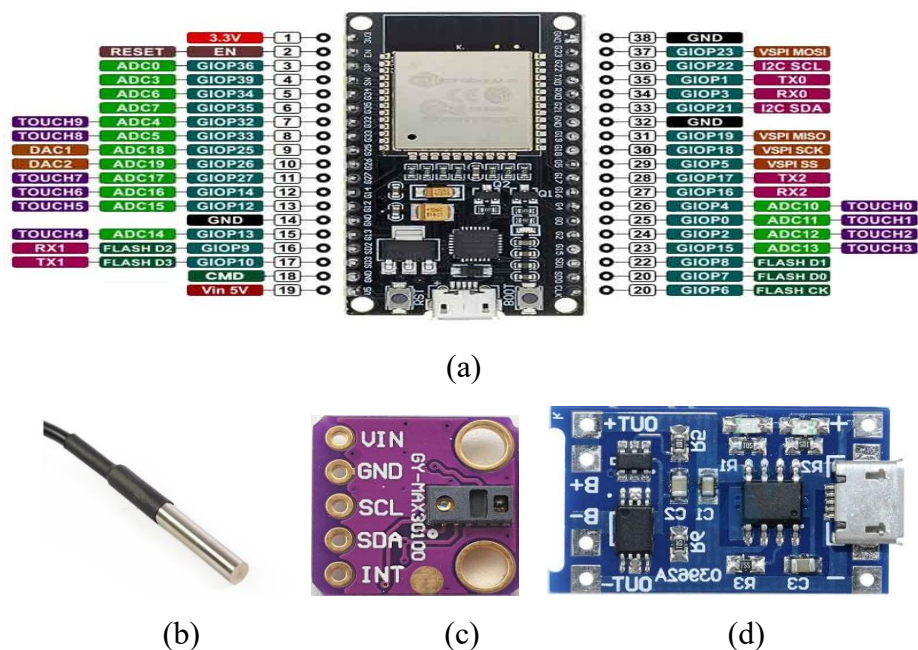


Figura 18. Módulos de hardware utilizados para el prototipo: (a) ESP32, (b) Sensor DS18B20, (c) MAX30102 y (d) TP4056

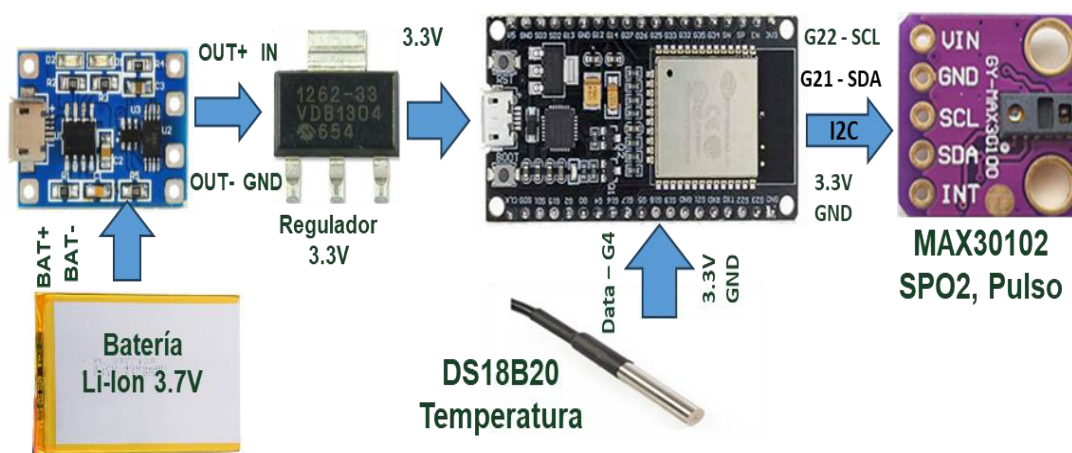


Figura 19. Interacción de elementos del componente nodo sensor 1 (NS1)

Todos estos componentes son integrados en una estructura para que se adapten a la muñeca de la persona y para que los sensores tengan contacto con la superficie necesaria para realizar las lecturas (Figura 20).



Figura 20. Ubicación del dispositivo de hardware – nodo sensor 1 (NS1)

b) Dispositivo embebido - Nodo sensor 2. En este caso de este dispositivo NS2, se encarga de adquirir la señal ECG, se detectan patrones de movimiento y se monitorea la frecuencia respiratoria para lo cual se usa el módulo AD8232 que adapta y acondiciona las señales. La descripción de los bloques que lo conforman se menciona a continuación (Figura 21) seleccionando los dispositivos integrados en la Figura 22:

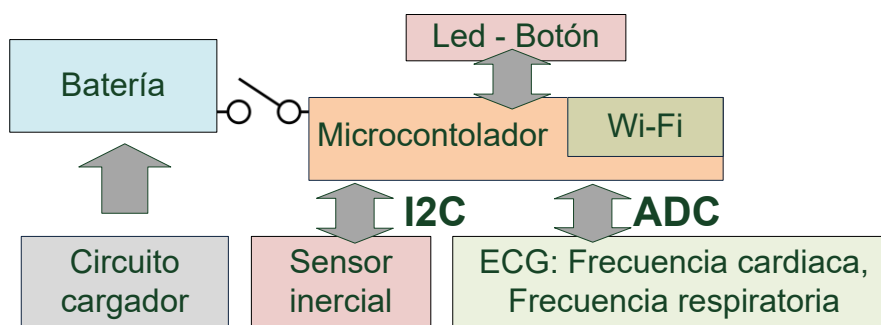


Figura 21. Elementos del nodo sensor 2 (NS2)

i) **Elementos del Bloque de sensores.** Este bloque permite obtener los siguientes parámetros:

- **Frecuencia cardiaca** (heart rate). Es el número de veces que el corazón se contrae en un minuto y se adquiere usando un módulo AD8232 y analizando la señal ECG.

- **Frecuencia Respiratoria.** Su valor se calcula mediante un análisis temporal de la señal ECG dentro del módulo ESP32 utilizando filtros digitales usando el mismo módulo AD8232
- **Aceleración.** Se obtiene mediante el uso del sensor inercial MPU6050 de 6 grados de libertad para obtener datos relacionadas al movimiento inercial a través del procesamiento de la señal de aceleración.

ii) **Elementos del Bloque procesamiento y control.** Se encarga de procesar las señales electrocardiográficas utilizando el circuito integrado AD8232. Se usa un circuito de polarización para amplificar y eliminar el ruido de la señal que ingresa al módulo ESP32. Este bloque realiza la inferencia del valor de la frecuencia respiratoria y cardiaca usando filtros digitales.

iii) **Elementos del Bloque de energía.** Se utiliza el mismo Circuito integrado TC1262-3.3 usado en el nodo sensor 1, junto con un circuito cargador TP4056.

iv) **Elementos del Bloque de comunicación.** La transmisión de datos se realiza usando el protocolo WiFi mediante el módulo integrado al ESP32 mediante el protocolo MQTT usando conexiones WebSockets a servicios Web.

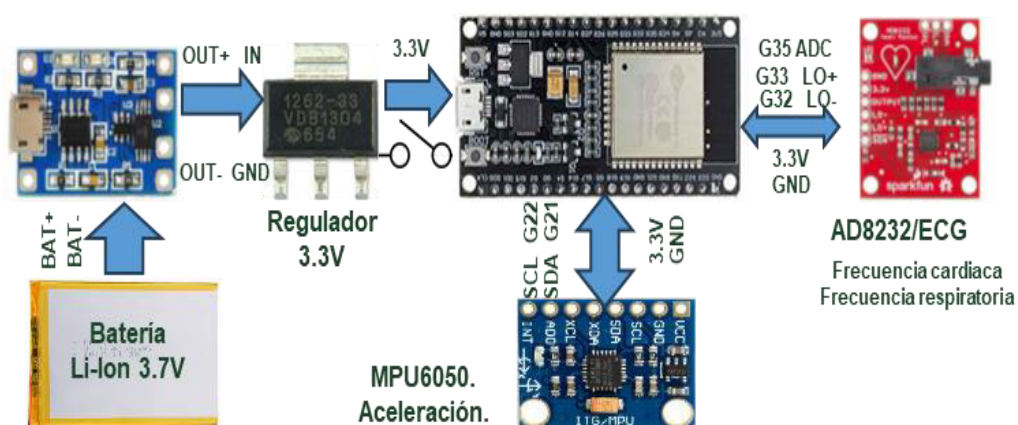


Figura 22. Componentes que integran el nodo sensor 2 (NS2)

Se considera que la ubicación de los electrodos es importante debido a que las señales captadas pueden cambiar y su análisis se vería afectado. Por lo tanto, se consideró el esquema de conexiones de la Tabla 19.

Tabla 19. Relación de colores y posición de los electrodos

Rojo	Verde	Amarillo
RA	LA	RL

Brazo derecho	Brazo izquierdo	Pierna derecha
---------------	-----------------	----------------

c) Aplicación de monitoreo Web. Los parámetros obtenidos son transmitidos a un servicio Web para su visualización y posterior almacenamiento. En este caso es necesario utilizar un conjunto de componentes de software con las siguientes características:

- Protocolo de mensajería estándar para dispositivos del Internet de las cosas
- Clientes MQTT que utilicen recursos mínimos para su en microcontroladores
- Transporte de mensajería de publicación/suscripción ligera.
- Sesiones persistentes para disminuir el tiempo de reconexión con el cliente.
- Uso de una interfaz de programación basada en cliente MQTT.

Debido a las características mencionadas anteriormente, se plantea el esquema de funcionamiento. Este consiste en integrar el protocolo de publicación MQTT en el dispositivo de hardware (nodo sensor) y realizar el envío de datos al Broker MQTT de Adafruit el cual gestiona las conexiones para su visualización en una aplicación Web (Figura 23).

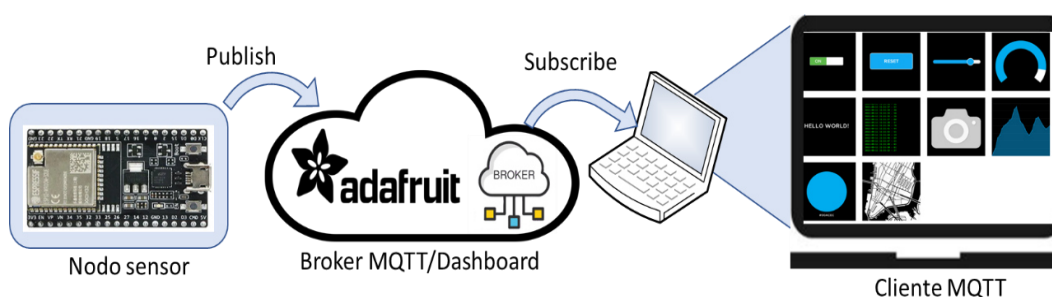


Figura 23. Esquema de comunicación del nodo sensor con la plataforma de monitoreo.

4.2.3 Implementación

La solución propuesta está conformada por una etapa de hardware y software donde los sensores para monitoreo de pulso cardíaco, saturación de oxígeno y temperatura integrados en el nodo sensor 1, se colocan en un brazalete. En el caso del segundo nodo sensor, este se coloca sobre un cinturón alrededor del pecho para permitir una mejor ubicación de los electrodos de la señal ECG, obtención de la frecuencia respiratorio. Estos dos componentes son integrados en una etapa posterior.

a) Implementación del dispositivo de hardware - nodo sensor 1. Se realizó la construcción del dispositivo de hardware utilizando el módulo ESP32 para realizar la adquisición, procesamiento y envío de datos de señales fisiológicas. El proceso de implementación está compuesto por una etapa de hardware y software descritas a continuación:

i) Hardware. El proceso de construcción del hardware se realizó en varias etapas iterativas incrementales. El prototipo del nodo sensor 1, se usa en forma pulsera en el paciente monitoreado y está integrado por los módulos de energía como el TC1262 (regulador de bajo nivel de entrada con salida de 3.3V), módulo TP4050 (Dispositivo de carga de batería) y sensores DS18B20 (temperatura corporal) y MAX30105 (SPO2 y pulso)(Figura 24)

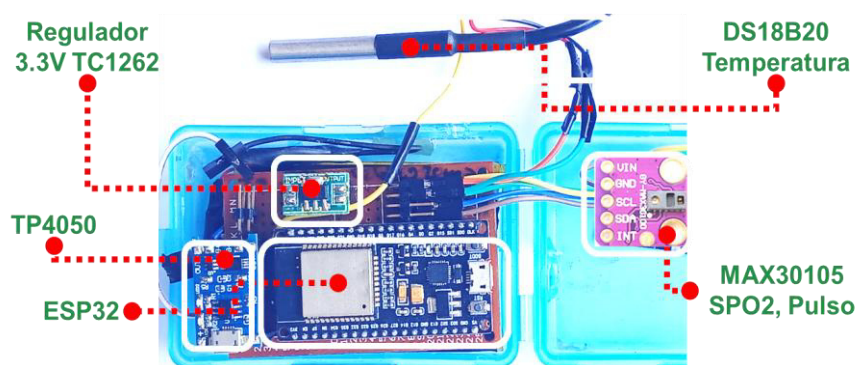


Figura 24. Nodo sensor 1 implementado

ii) Firmware. El programa implementado permite acceder a los datos generados por el sensor de temperatura DS18B20 y el sensor MAX3010, usando librerías de lectura construidas en lenguaje C, obteniendo la lectura del pulso y del nivel de saturación de oxígeno. Las primeras pruebas de lectura obtienen los valores del nivel de saturación de oxígeno (BPM), presencia de señal de luz infrarroja (IR), sensor de temperatura del sensor DS18B20 (Figura 25).

```

IR=3323, BPM=78.23, Avg BPM=0
IR=3322, BPM=78.23, Avg BPM=0
IR=3321, BPM=78.23, Avg BPM=0
IR=3318, BPM=78.23, Avg BPM=0
Temperatura Ambiente= 29.23 °C
Temperatura del Objeto= 30.11 °C
Temp DS18B2035.94 °C
Temperature: 25.20
red=4483, ir=3314, HR=-499, HRvalid=0, SPO2=-999, SPO2Valid=0
red=4491, ir=3327, HR=-499, HRvalid=0, SPO2=-999, SPO2Valid=0
red=4500, ir=3321, HR=-499, HRvalid=0, SPO2=-999, SPO2Valid=0

```

☒ Autoscroll ☐ Mostrar marca temporal

Figura 25. Resultados de las lecturas de los sensores

Con las pruebas mencionadas anteriormente se procede a desarrollar el programa basado en la creación de funciones para obtener el cálculo de pulsos por minuto (BPM). El proceso de cálculo de la saturación de oxígeno (SPO2) y pulso (HR) implica tomar un cierto número de muestras para actualizar estas variables. Las funciones utilizadas dentro del programa son:

- **procesoTemp()**. Esta función se encarga de realizar la comunicación con el sensor DS18B20 para obtener el valor de temperatura.
- **procesoSPO2HR()**. Esta actividad tiene un tiempo de procesamiento de 4 segundos. Se toman 75 muestras de forma inicial y luego se adquieren en bloques de 25 muestras.
- **detectarBeat()**. Este proceso se ejecuta cada 12 segundos. Es necesario que este proceso se ejecute de manera continua durante 4 segundos para tener buenas lecturas.
- **mqtxperiod()**. Realiza la transmisión de datos adquiridos de temperatura, SPO2 y pulso cada 20 segundos. Además, las transmisiones de datos se ejecutan en forma de tópicos grupales para evitar más de una conexión por transmisión.
- **txperiod()**. Este procesamiento se realiza cada 20 segundos para transmitir los datos adquiridos de temperatura, SPO2 y pulso usando el puerto serie, para verificación.
- **procesoMQTT()**. Este proceso realiza las conexiones con el servidor MQTT de adafruit.io en Internet.

Para realizar la transmisión de los datos procesados, se utiliza el protocolo de comunicación MQTT (Message Queing Telemetry Transport) el cual permite realizar transmisión de datos de sensores hacia servicios en nube como Adafruit.io. Para realizar la transmisión de datos se usa las librerías llamadas “PubSubClient” que permite crear elementos llamados tópicos y configurar la conexión hacia los servicios en nube (O’Leary, 2017) usando la conexión WiFi y MQTT. Los procesos de adquisición de datos se realizan en dos etapas diferenciadas, donde la primera de ellas se realiza en la sección de setup para inicializar parámetros y variables usadas en las funciones. Luego, en el bucle principal se invocan a las funciones mencionadas anteriormente para su ejecución cada 20 segundos (Figura 26).

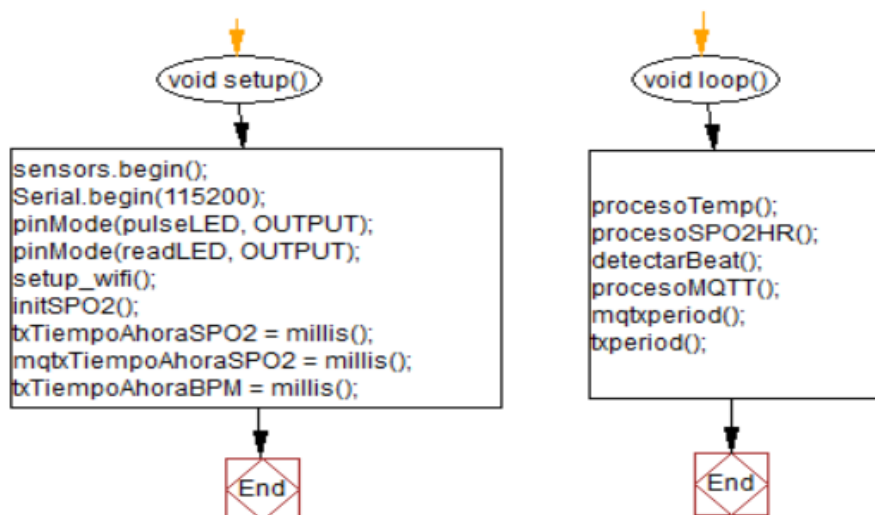


Figura 26. Procesos de ejecución del dispositivo de hardware – nodo sensor 1

Uno de los procesos más complejos corresponde al cálculo del SPO2 el cual inicia con el uso de un buffer (1), donde se eliminan las 25 muestras más antiguas y se reorganizan los datos restantes para agregar los nuevos datos. Posteriormente, se realiza la captura de datos (2), donde el sensor MAX30105 genera valores de los canales rojo e infrarrojo y temperatura, aplicando filtros para reducir el ruido. Luego, se procede con el cálculo de la frecuencia cardíaca (HR) y la saturación de oxígeno (SPO2) (3) mediante un algoritmo que determina la validez de los datos para almacenar los resultados. Finalmente se verifica (4) si el dedo está correctamente colocado y si son válidas, se almacenan los valores (Figura 27).

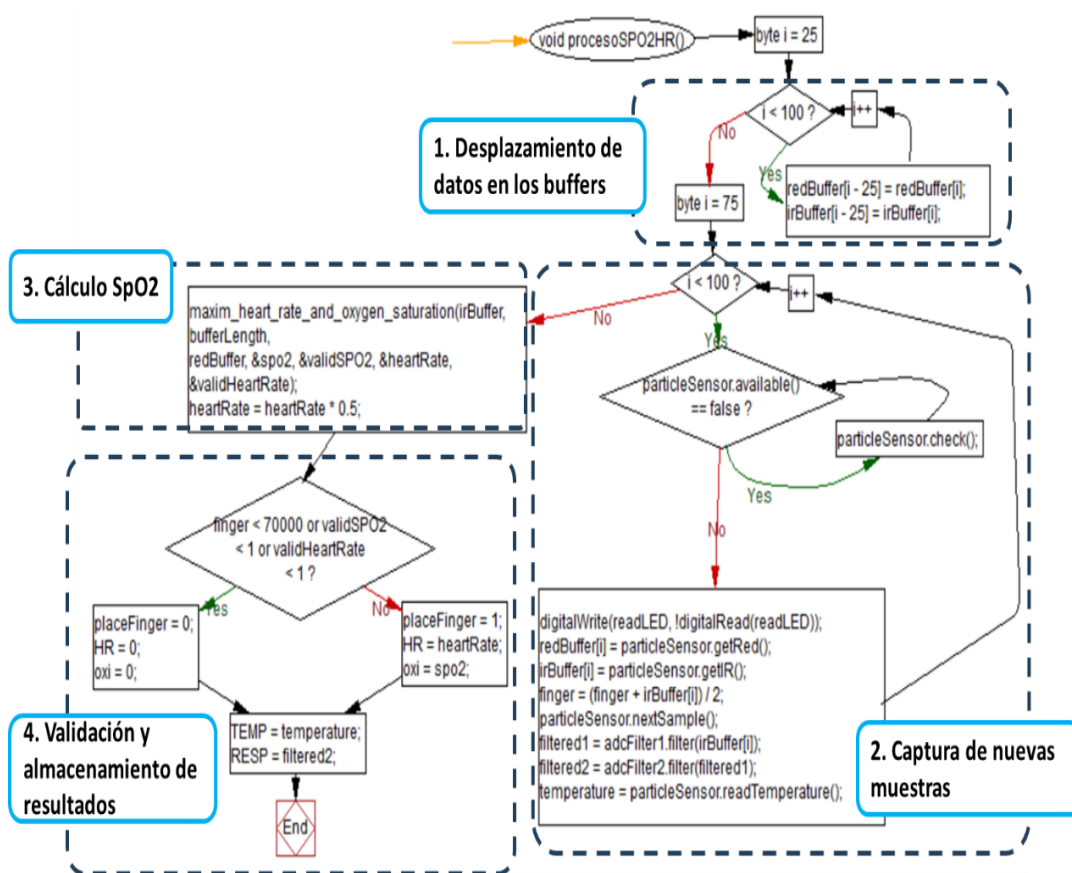


Figura 27. Diagrama de flujo para el cálculo de SPO2

b) Implementación del dispositivo de hardware - nodo sensor 2. Tiene como tarea principal el procesamiento de la señal ECG mediante el uso de filtros digitales y analógicos y adquisición de datos del acelerómetro. Estos componentes se pueden colocar en diferentes posiciones usando un cinturón sobre el torso de la persona cuya diferencia se basa en la distancia de los electrodos respecto al nodo realizando las pruebas con la propuesta de posición 1 que se muestra en la Figura 28.

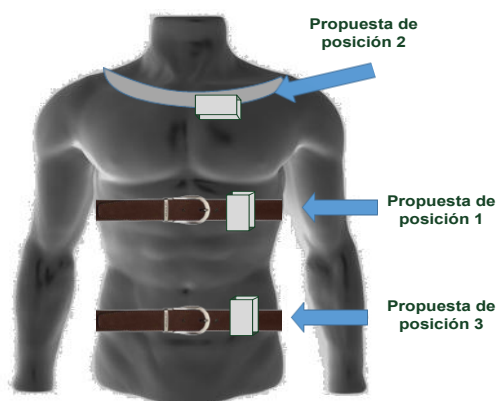


Figura 28. Ubicaciones para el dispositivo de hardware - nodo sensor 2

La construcción de este nodo sensor se realizó utilizando el módulo ESP32 estando compuesto por una etapa de firmware y de hardware los cuales se describen a continuación:

i) Hardware. Se realizó la construcción del dispositivo de hardware con las características del nodo sensor 2. Para ello se utilizaron los módulos de hardware que integran los componentes observados en la siguiente figura, y conectados usando una placa base donde se coloca el módulo ESP32 (Figura 29).

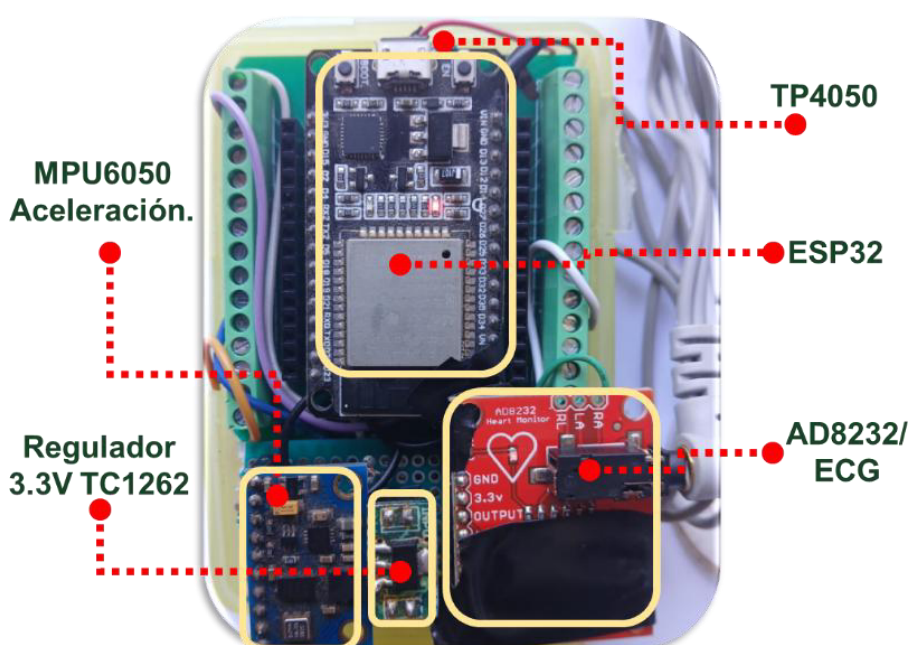


Figura 29. Componentes del dispositivo de hardware - nodo sensor 2 integrados

ii) Firmware. El firmware implementa el algoritmo de detección de la frecuencia cardíaca y respiratoria. Para ello, se utilizan los datos capturados a un tiempo de muestreo (T_s) igual a 5ms (200 Hz). Se sigue un esquema de pasos para el procesamiento (Hussain, 2021) e implementación del firmware (J. Pan & Tompkins, 1985)(Hamilton & Tompkins, 1986) describiendo a continuación los pasos programados para el procesamiento de los datos y el cálculo de la frecuencia cardíaca (Figura 30).

- **Preprocesamiento.** La señal ECG es capturada a una frecuencia de muestreo de 100 Hz. De esta forma se evita adquirir el ruido de la línea eléctrica de 60Hz usando un ADC para la recuantización a 10bits.
- **Limpieza de la señal.** Se aplican filtros digitales para el suavizado de la señal y eliminación de la señal DC. Para el proceso de suavizado se utiliza un filtro promediador implementado con la librería de Arduino “EWMA” (Paso 1) y luego se elimina la señal continua mediante un filtro pasalta (Paso 2)
- **Detección de frecuencia cardiaca.** Se realiza la derivación de la señal (paso 3). Además, se aplica un Filtro suavizado (Con librerías “Ewma” que corresponde al Paso 4 (Figura 30).

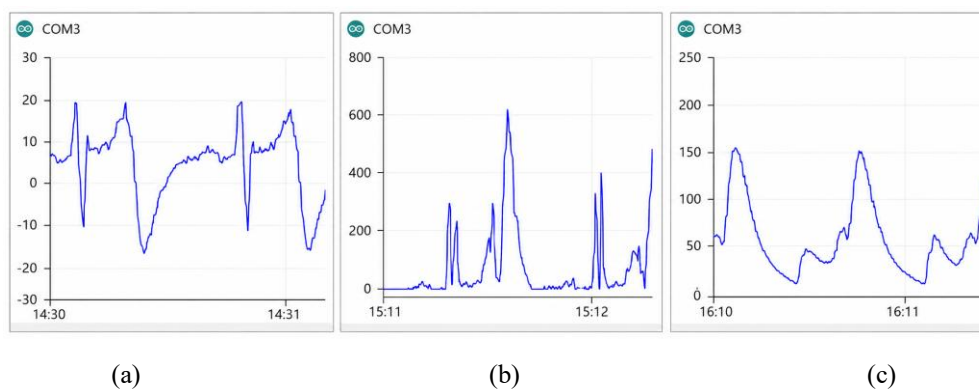


Figura 30. Proceso de (a) derivada, (b) cuadratura (paso 3) y (c) suavizado (paso 4)

Para detectar latido del corazón para una persona en movimiento, en la Tabla 20 se muestra cada uno de los pasos para procesar la señal (Figura 31). Las secciones del programa para cada uno de los pasos se aplican a las señales entrantes adquiridas por el módulo AD8232.

Tabla 20. Descripción de etapas para el procesamiento de la señal ECG

Pasos	Procesos	Objetos de programación	Descripción
1	Ruido de línea [notch] y filtro de paso bajo	Ewma ADCfilter y KickFiltersRT NOTCH	suavizado Filtro notch de 50Hz y filtro pasa baja
2	Filtro de paso alto para deshacerse de la compensación de CC	KickFiltersRT de paso alto	Eliminar señal DC
3	Rectificación y amplificación	Derivada y cuadrada [sq (x)]	Cálculo de pulsos por minuto
4	suavizante e integral	Ewma ADCfilter1 [media móvil]	Suavizado de la señal

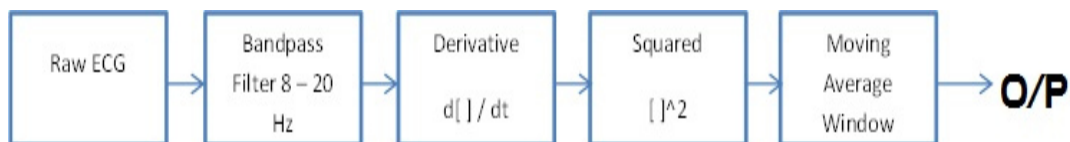


Figura 31. Pasos para el análisis de la señal ECG y cálculo de frecuencia cardiaca y respiratoria

Con las señales resultantes de las etapas descritas, se aplica un algoritmo de detección de la frecuencia respiratoria a través de una máquina de estados mediante el cálculo de cada ciclo respiratorio (Figura 32). El procedimiento se puede resumir de la siguiente forma:

- En el estado “0” se detecta si la señal de respiración se incrementa. Siempre que se esté incrementando se asigna el valor “**rtimeResp**” para que almacene el valor del pico máximo de la señal. En el momento que se detecte que la señal se decrementa y se pasa al estado 1.
- En el estado “1” se detecta si la señal decrece. Este estado continúa hasta que se detecta que la señal se incrementa. En este momento se realiza el cálculo de la frecuencia respiratoria restando los tiempos de los dos picos de la señal.

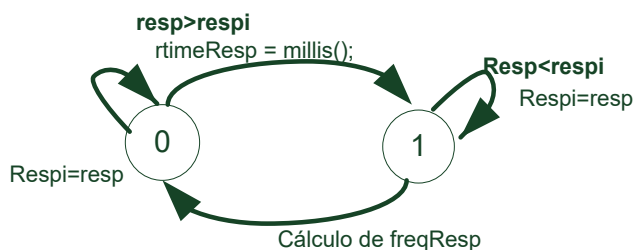


Figura 32. Programa de detección de la frecuencia respiratoria

Para realizar la transmisión de los datos se utilizan ventanas de tiempo para enviar los picos de la señal SPO, temperatura, pulso y los parámetros de frecuencia respiratoria y los valores picos de la señal ECG filtrada realizando hasta 30 transmisiones por minuto.

El Proceso final de transmisión de datos se realiza en ventanas de tiempo de 12 segundos para todos los datos, incluyendo los datos a graficar de la frecuencia

respiratoria y el valor de frecuencia calculada. Esto se realiza usando una máquina de estados para la controlar la ventana de transmisión de datos (Figura 33).

- En el estado “0” se realiza la transmisión de datos mientras no haya transcurrido un tiempo igual a la ventana de tiempo de 12 segundos. Si ya paso el tiempo, se cambia al estado “1”.
- En el estado “1”, corresponde a un proceso en el cual se encontrará el nodo sensor mientras no haya transcurrido un tiempo mayor de 60 segundos.

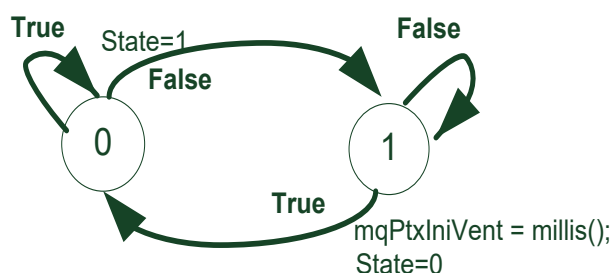


Figura 33. Programa de detección para la transmisión de datos usando MQTT

En el caso de la señal de aceleración, esta es capturada por el módulo ADC a una frecuencia de muestreo de 100 Hz (cada 10 ms), utilizando un bucle que dura 1 segundo antes pasar a un estado de inactividad por un tiempo de 3 segundo. La adquisición del sensor inercial acelerómetro se hace de la siguiente manera:

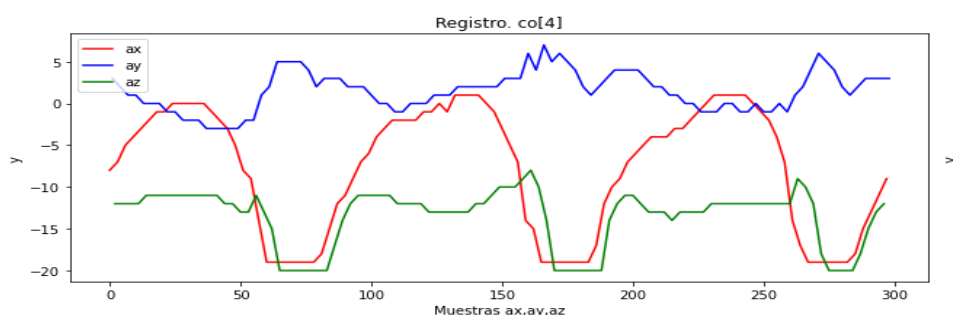
- Puede registrar señales inerciales con una frecuencia de muestreo estable de hasta 200 Hz (para detectar patrones del movimiento con frecuencia mínima de movimiento de 40 Hz)
- El IMU MPU6050 tiene 3 sensores triaxiales considerando un acelerómetro, un giroscopio y un magnetómetro triaxiales,
- Incorpora elementos de filtrado analógico integrados al dispositivo para minimizar posibles interferencias.

Se utiliza un algoritmo de detección de actividades considerando tres clases a clasificar las cuales son: Corriendo, Caminando y Estático (parado) denominadas como “co”, “ca” y “es”. Cada registro de datos representa una serie temporal capturada en una ventana de tiempo de 3 segundos con 300 muestras (columnas) alternando datos en

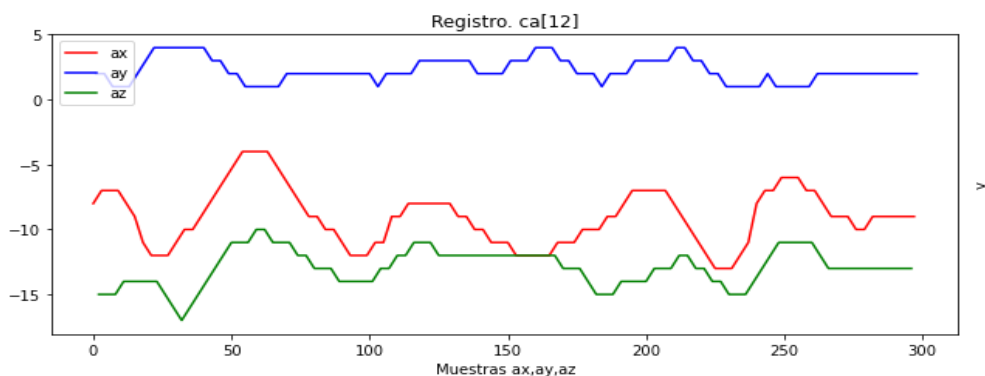
cada uno de los 3 ejes del acelerómetro (Tabla 21), que tienen como etiqueta las clases Running (co), Walking (ca) y Static (es). Una muestra de las capturas realizadas se observa en la Figura 34.

Tabla 21 . Esquema de muestras captadas por el IMU

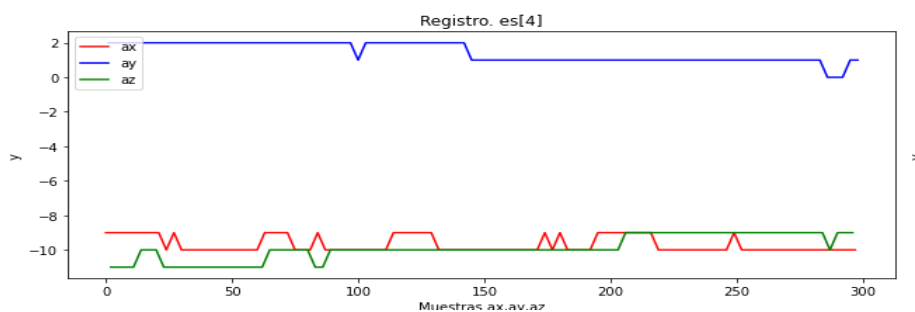
X1	,	y1	,	y1	,	...	X100	,	y100	,	Z100
-2	,	3	,	-11	,	...	2	,	-2	,	-12
0	,	-1	,	-11	,	...	-2	,	0	,	-10



(a)



(b)



(c)

Figura 34. Datos capturados de la actividad de a) Corriendo (co), b) Caminando (ca) y c) estático (es)

c) Implementación de la Aplicación de gestión y monitoreo de datos. Se realiza la configuración de diferentes componentes de recepción y visualización en el servicio Web de Adafruit para la recepción de datos enviados desde los dos nodos sensores (Figura 35). Esto implica realizar una serie de pasos para preparar el entorno de recepción y monitoreo de datos:

- Se crean componentes denominados “feeds”, los cuales están asociados a variables creadas en el programa de cada nodo sensor.
- Se crea un entorno de visualización denominado dashboard, sobre el cual se agregan componentes llamados Widgets.
- Cada widget se asocia a un feed para visualizar los datos recibidos
- El servicio Web genera unas credenciales de acceso las cuales son ingresadas en el dispositivo de hardware del nodo sensor para establecer la conexión persistente.

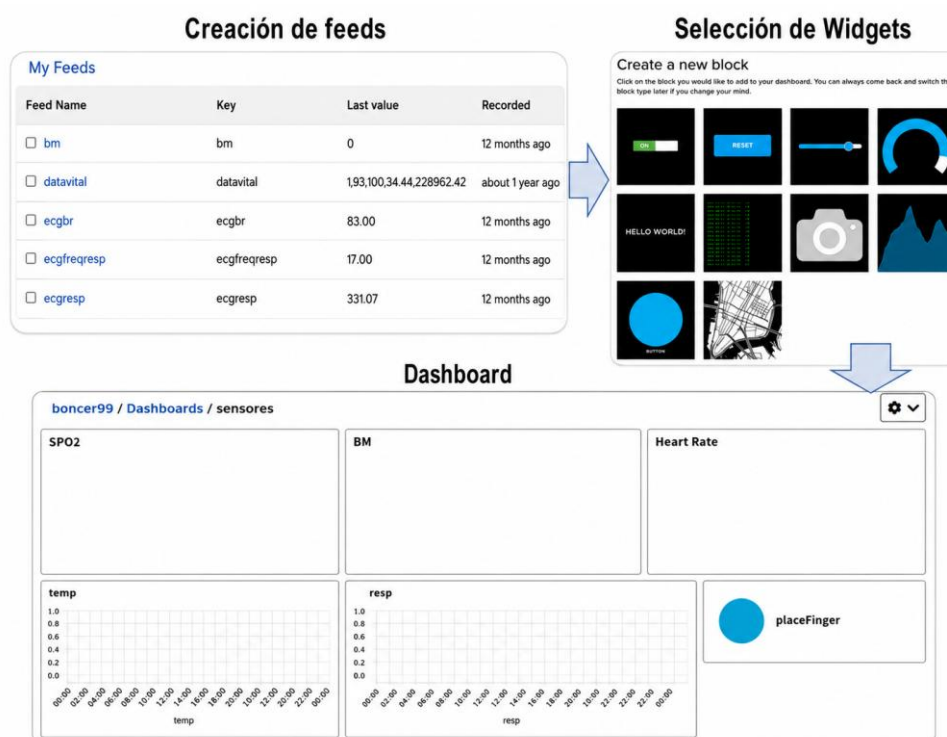


Figura 35. Proceso de creación de la interfaz de visualización de datos

La asociación entre las variables de sensores (feeds) y los componentes de visualización (widgets) se muestra en la siguiente Tabla 22 junto con los parámetros de configuración al momento de integrarlos en la pantalla de visualización

(Dashboard) y el dispositivo embebido relacionado al nodo sensor 1 o nodo sensor 2 (Nodo sensor NS).

Tabla 22 . Relación entre los feeds y widgets generados

feed	widget	parámetro	NS
hr	stream	Frecuencia cardiaca	1
Spo2	stream	Saturación de oxígeno	1
temp	Line chart	Temperatura corporal	1
bm	stream	Pulsos por minuto	1
placefinger	indicator	Detección de superficie de lectura	1
ecgbr	stream	Frecuencia cardiaca	2
ecgresp	Line chart	Amplitud de la señal ECG	2
ecgfreqresp	stream	Frecuencia respiratoria	2
estadoacel	Stream	Identificador de actividad	2
stringacel	stream	Nombre de la actividad	2

4.2.4 Resultados

En esta sección se describe los resultados de la implementación de los nodos sensores inalámbricos y la transmisión de parámetros obtenidos desde los sensores a una aplicación web para la gestión de datos, con la finalidad de realizar el monitoreo de pacientes ambulatorios.

a) Dispositivo embebido– Nodo sensor 1. En base al algoritmo desarrollado se realizan pruebas de evaluación para verificar la extracción de información de la señal que se va generando por diferentes filtros, usada para obtener el valor de SPO2. De manera específica se muestran los resultados de la cantidad de luz reflejada en la longitud de onda roja (red), ir (luz reflejada en la longitud de onda infrarroja), frecuencia cardíaca (HR), saturación de oxígeno (SPO2) y valores filtrados (filtered1 y filtered2). En esta prueba se imprime inicialmente en formato JSON valores de temperatura (temp), frecuencia respiratoria estimada (resp) y una variable que indica si el dedo está correctamente colocado (placeFinger) (Figura 36).

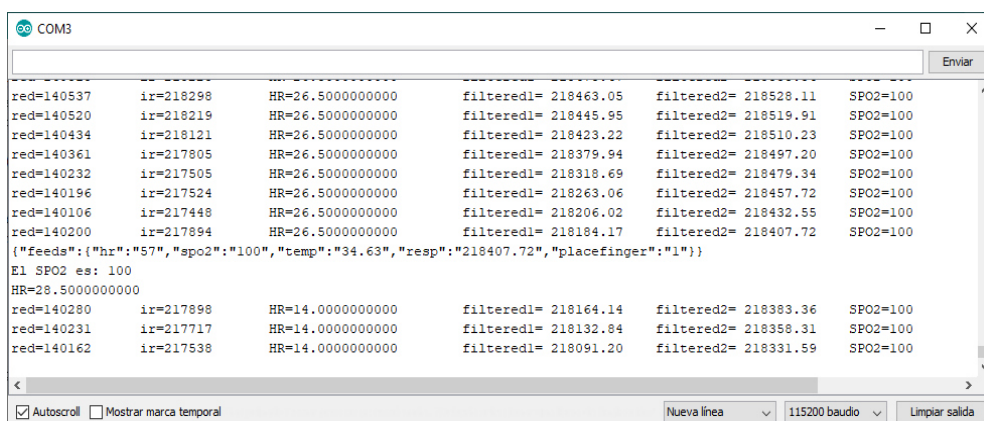


Figura 36. Resultados de la lectura de los parámetros de signos vitales.

Posteriormente se adapta la salida sólo para mostrar la información ya procesada de las mediciones de datos enviadas al servidor MQTT de Adafruit. La información se estructura en formato JSON que luego son publicadas en un feed de Adafruit IO para visualización remota. Los identificadores enviados contienen las claves “spo2” (Saturación de oxígeno), “temp” (Temperatura corporal estimada) y “bm” (Pulso cardíaco). Además, se imprime la frecuencia cardíaca medida (Heart Rate) (Figura 37).

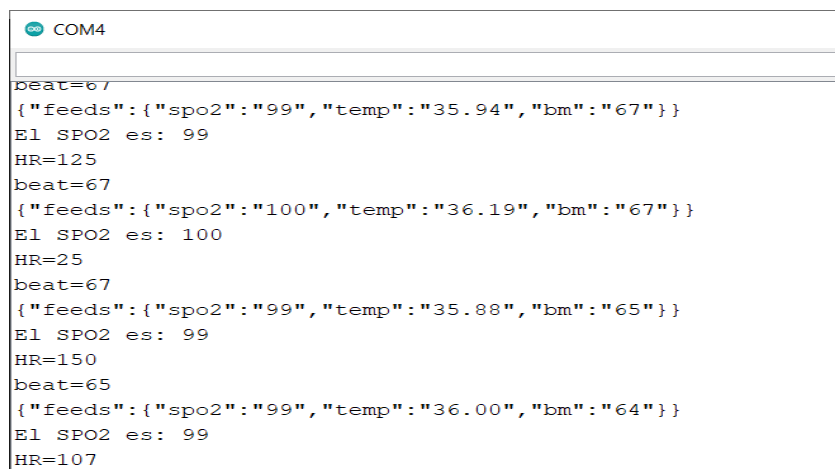


Figura 37. Salida de parámetros de signos vitales.

En el caso del dispositivo de hardware del nodo sensor 2, se integraron los componentes para ser colocado entre el antebrazo y la muñeca de la persona a monitorear. Esta electrónica está dentro de una estructura que expone el sensor MAX30105 y permite la salida del sensor de temperatura DS18B20. Este dispositivo fue sujeto con una muñequera (Figura 38).



Figura 38. Ubicación del dispositivo de hardware – nodo sensor 1

b) Dispositivo embebido - Nodo sensor 2. El firmware, para el caso de la extracción de la frecuencia respiratoria, implica un proceso de limpieza de la señal (paso 1 y paso 2), donde, en la Figura 39, se muestra la señal obtenida en el dispositivo de hardware. Se observa que los rangos de la señal resultante respecto a los obtenidos en la evaluación realizada en la plataforma de software para análisis tienen el mismo rango de variación de datos. Además, estas señales son el resultado de aplicar el filtro de suavizado (paso 1) utilizando la función “Ewma” y filtro pasa alta para eliminar la señal continua (paso 2)

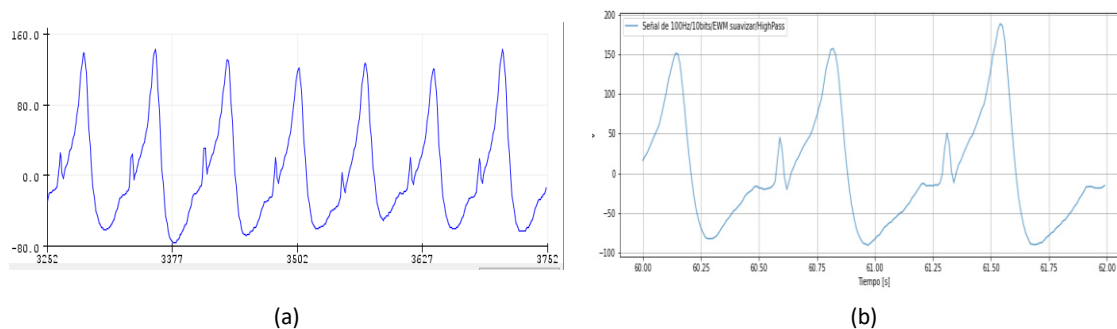
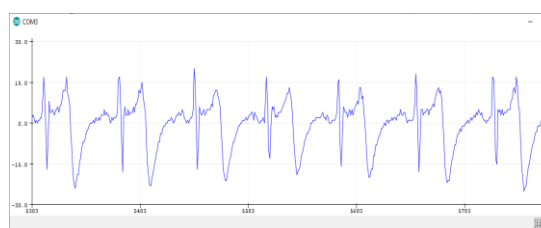


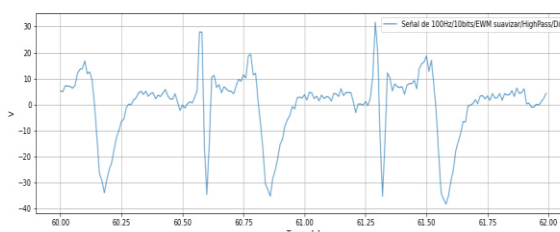
Figura 39. Resultados de la implementación de los dos primeros pasos de procesamiento en el módulo (a) Nodo sensor 2 y (b) en Python

Se obtienen los pulsos por minuto (paso 3 y paso 4) extraídos de la señal anterior, aplicando procesos de derivación, cuadratura y suavizado de la señal. Estos permiten aplicar un algoritmo de detección de picos y de duración de los pulsos por minuto. Los resultados obtenidos están en el rango de 0 a 250 mientras que los obtenidos por el software de Python, al aplicar los filtros, están en el rango de 0 a 900 (Figura 40).

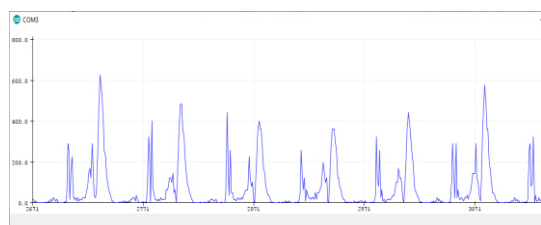
En la etapa final, se realiza el cálculo de los pulsos por minuto detectando un nuevo ciclo cuando existe un cambio de tiempo de validación de la señal QRS de 40 ms a 150 ms (Hussain, 2021). Además, para la detección del inicio o final de la señal QRS (StopTime menos StartTime) se usa un valor umbral de 50. Esto permite detectar los verdaderos picos, considerando que, si los falsos picos fueran detectados, estos son eliminados (Figura 41).



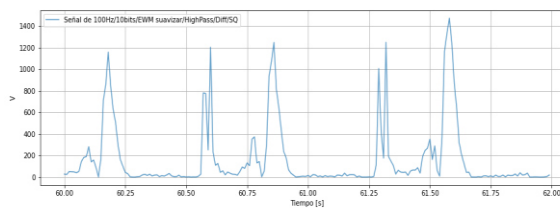
(a) Paso 3. Derivada en hardware



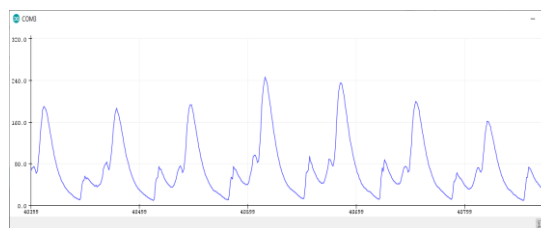
(b) Paso 3. Derivada en python



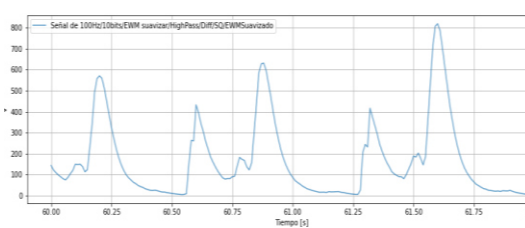
(c) Paso 3. Cuadratura en hardware



(d) Paso 3. Cuadratura en python



(e) Paso 4. Suavizante en hardware



(f) Paso 4. Suavizante en python

Figura 40. Resultado con (a) la derivada y (c) cuadratura- paso 3, (e) filtro suavizante- Paso 4; y en Python con (b) la derivada y (d) cuadratura-paso 3 y (f) filtro Suavizante- paso 4

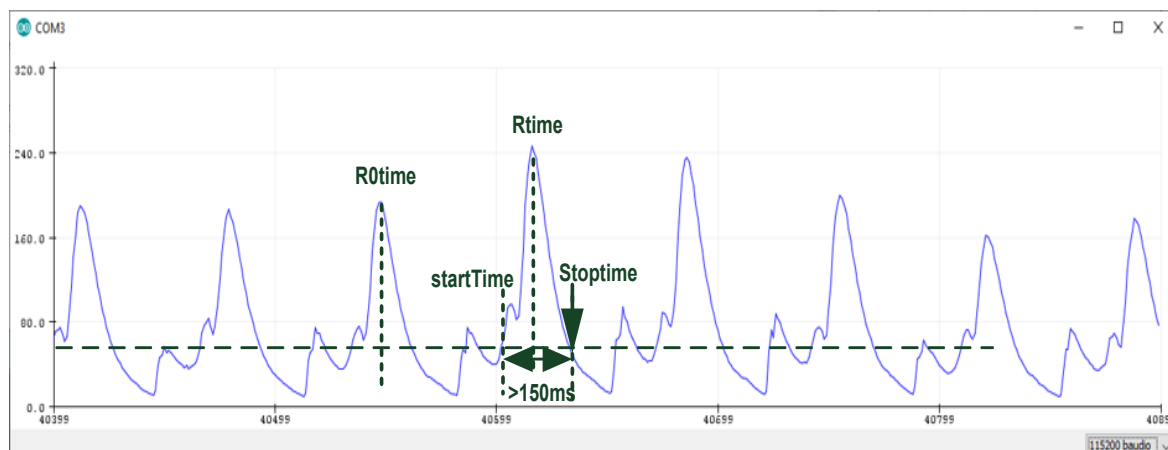


Figura 41. Tiempos y umbrales utilizados para detección de picos

El programa de detección actualiza la variable “RESP” con el valor del pico de la señal QRS que se ha obtenido en ese momento. En este caso, observando los picos de las señales QRS, hay 18 respiraciones en 100 segundos, por lo tanto, se tiene 5.55 respiraciones por minuto y una Frecuencia respiratoria de 0.18 Hz (Figura 42).

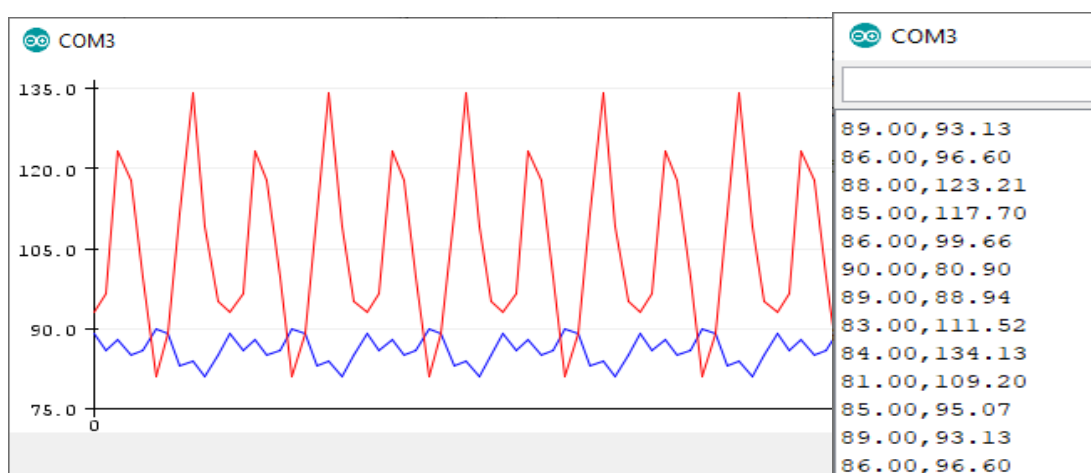


Figura 42. Visualización de datos de frecuencia respiratoria

Con los datos ya procesados respecto a la frecuencia respiratoria y actividad del movimiento se realiza la verificación del funcionamiento del programa visualizando la información en un monitor serial (Figura 43) considerando que la información más relevante enviada corresponde a: ECGRESP, ECGRESP, stringacel y estadoacel.

```

COM4
89.00,81.28,10.00
{"feeds":{"ecgbr":"89.00","ecgresp":"81.28","ecgfreqresp":"10.00"}}
229.00,98.72,10.00
{"feeds":{"ecgbr":"229.00","ecgresp":"98.72","ecgfreqresp":"10.00"}}
141.00,111.28,10.00
{"feeds":{"ecgbr":"141.00","ecgresp":"111.28","ecgfreqresp":"10.00"}}
228.00,105.55,10.00
{"feeds":{"ecgbr":"228.00","ecgresp":"105.55","ecgfreqresp":"10.00"}}
86.00,128.13,23.00
{"feeds":{"ecgbr":"86.00","ecgresp":"128.13","ecgfreqresp":"23.00"}}
Ciclo activo iniciado
1
{"feeds":{"estadoacel":"1","stringacel":"caminando"}}
2
{"feeds":{"estadoacel":"2","stringacel":"corriendo"}}
1
{"feeds":{"estadoacel":"1","stringacel":"caminando"}}
0
{"feeds":{"estadoacel":"0","stringacel":"estatico"}}
Ciclo activo finalizado
reconnectando

```

Autoscroll ☐ Mostrar marca temporal Nueva línea 115200 baudio Limpiar salida

Figura 43. Visualización de resultados generados por el dispositivo embebido-Nodo sensor2.

El dispositivo de hardware fue adaptado para poder colocarse sobre la persona a monitorear considerando que debe estar dentro de una estructura cerrada, y con espacio de salida para colocar los electrodos. Este dispositivo fue sujeto con una correa de ajuste al torso de la persona donde se observan los electrodos ya distribuidos (Figura 44).

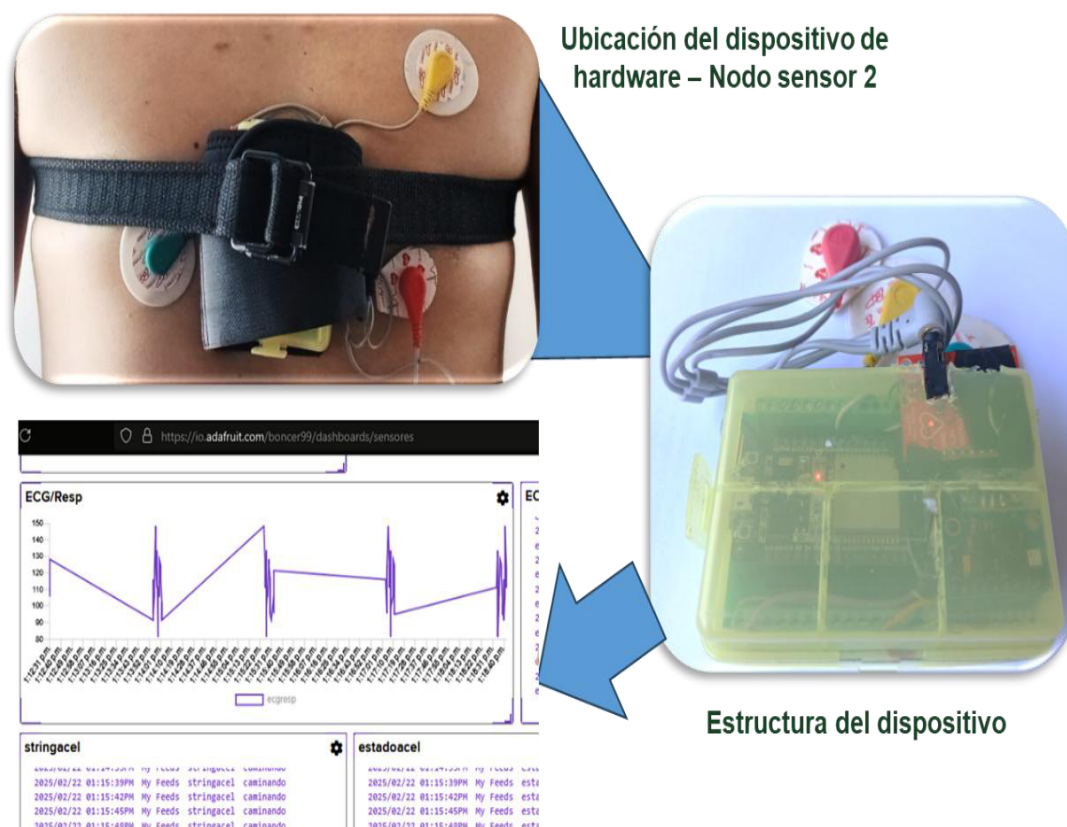


Figura 44. Ubicación del dispositivo embebido – nodo sensor 2

c) Aplicación de monitoreo. Los datos son transmitidos a la plataforma Adafruit.io utilizando el esquema de transmisión de variables por grupos. Con esto se obtiene una sola conexión para transmitir los datos y no realizar una conexión específica para cada variable. En la Figura 45 se observa como la frecuencia respiratoria varia observando la forma de la señal, además también la evolución de frecuencia cardiaca. Finalmente la aplicación de monitoreo se realiza con el servicio de Adafruit.io donde los parámetros del nodo sensor 1 están asociado a los feeds disponibles:

- SPO2. Oxígeno de la sangre
- TEMP. Temperatura
- BM. Pulsos por minuto

boncer99 / Dashboards / sensores

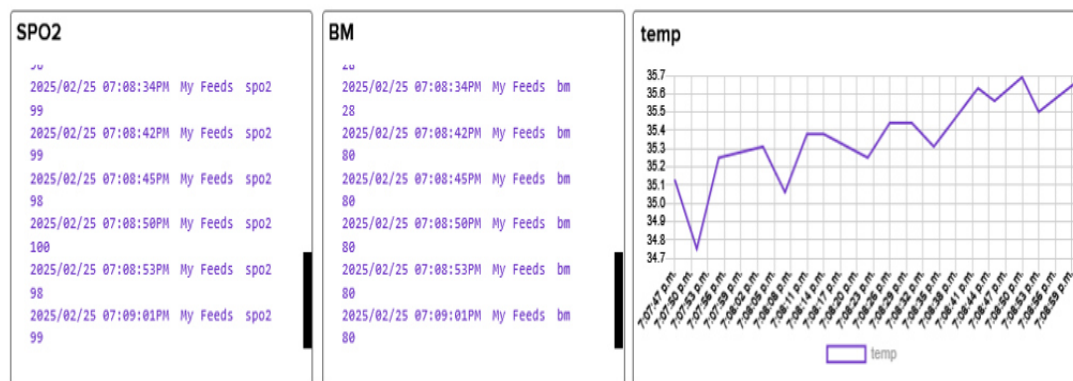


**Figura 45. Visualización de datos del dispositivo embebido- sensor 1
en la plataforma Adafruit.io**

Si bien los nodos sensores realizan la transmisión de datos al servicio Web, se realiza la verificación comparándolos con los generados localmente mediante la consola serial. En el caso del nodo sensor 2, los parámetros adquiridos fueron enviados en una sola conexión generado mediante el protocolo MQTT. Los componentes gráficos de la señal permitieron visualizar datos en forma de un gráfico temporal o una consola de texto (Figura 46). Para estos se seleccionaron algunos de los feeds disponibles en el dispositivo:

- ECGRESP. Valor numérico de la amplitud de la señal
- ECGRESP. Valor de la frecuencia respiratoria
- stringacel. Nombre de la actividad de movimiento
- estadoacel. Identificador de actividad

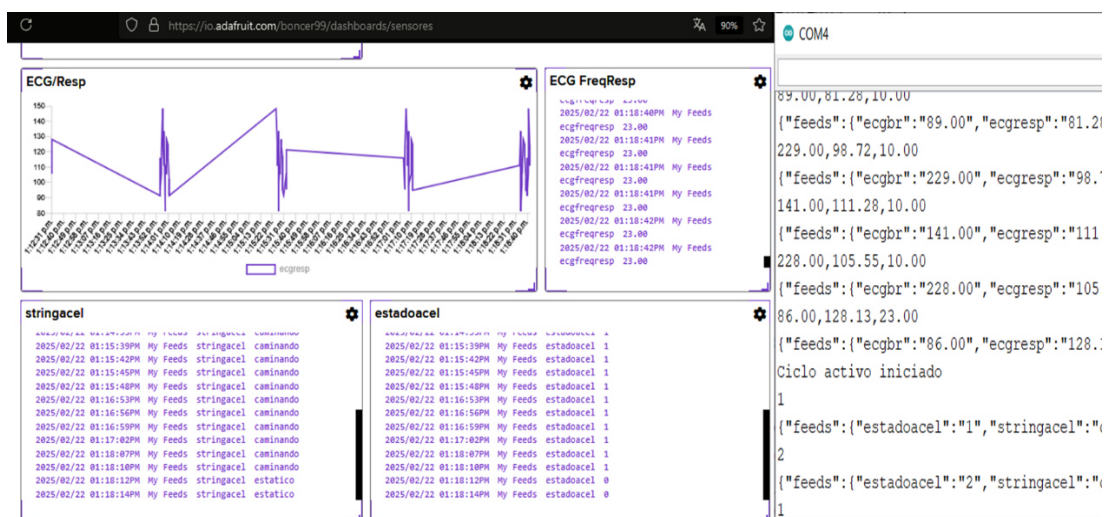


Figura 46. Visualización de datos generados por el nodo sensor 2

d) Dispositivo embebido integrado. Se realiza la integración de los componentes basados en los nodos sensores anteriores, los cuales se comunican usando el puerto serial número 2 de cada dispositivo. En este caso la información de los sensores de aceleración y frecuencia respiratorio se envió al otro dispositivo embebido para la consolidación de datos y envío de información a la nube (Figura 47).

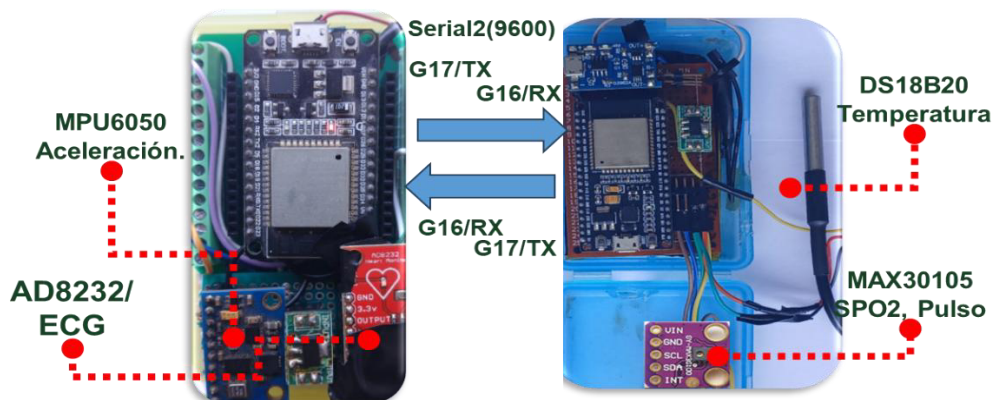


Figura 47. Esquema de integración de los nodos sensores

Para coordinar la lectura de los datos de sensores del dispositivo de hardware del nodo sensor 2 (DHC2) utilizando el dispositivo de hardware del nodo sensor 1 (DHC1) se envía un comando de solicitud "R". Luego de este envío el DHC1 se queda esperando la respuesta con información de los sensores del DHC2 (frecuencia respiratoria y aceleración). Cuando se recibe la respuesta en el DHC1 se realiza la consolidación de

la información con los datos de sensores de SPO2, pulso y temperatura para su envío a la plataforma Web de Adafruit utilizando el protocolo MQTT (Figura 48).

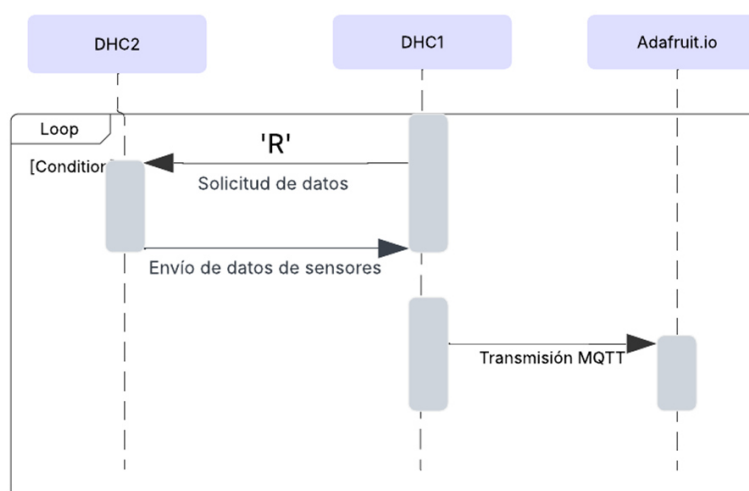


Figura 48. Diagrama de interacción entre el DHC2, DHC1 y plataforma Adafruit

Se implementó el programa en base al algoritmo anterior realizando pruebas parciales y de integración, validando los resultados de lectura, transmisión y recepción en cada uno de los dos componentes (Figura 49). Finalmente se visualiza la información integrada en una trama enviada a la plataforma web (Figura 50).

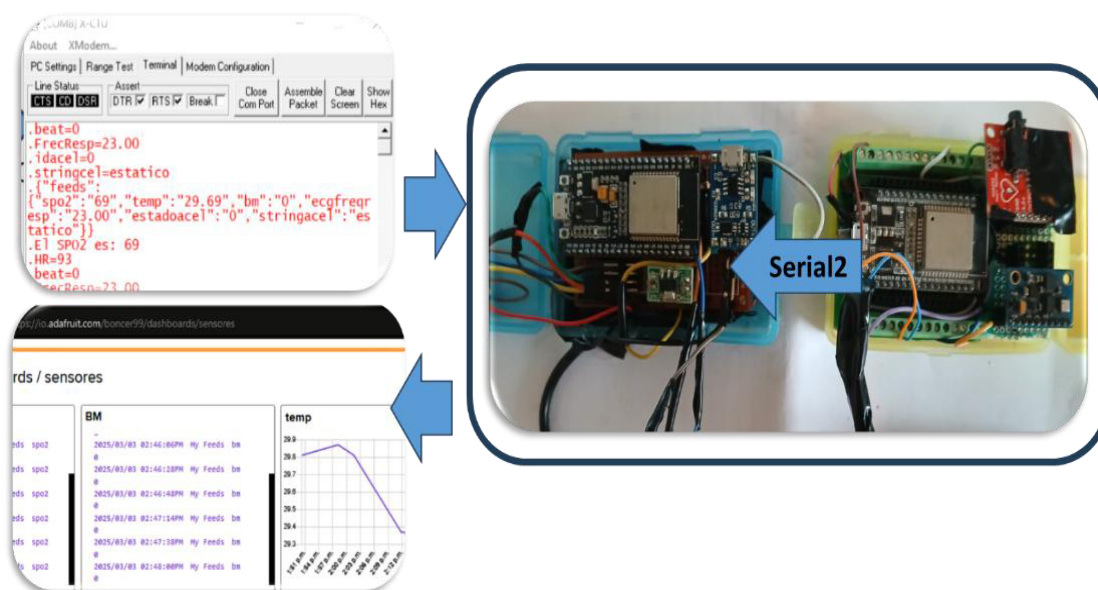


Figura 49. Esquema de pruebas de validación con integración de datos de los dos componentes de hardware

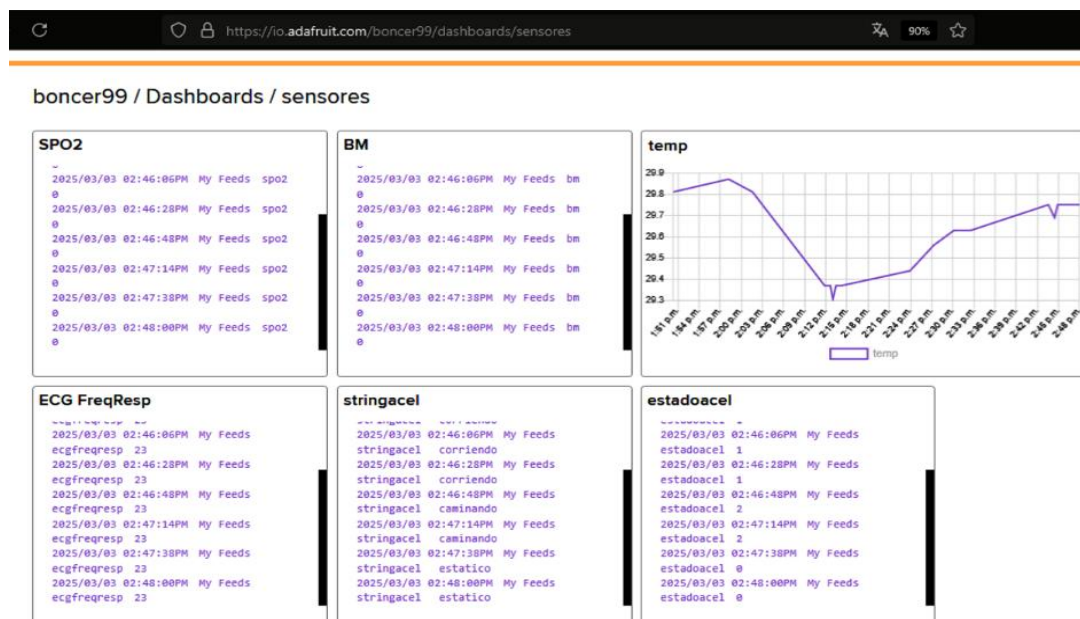


Figura 50. Visualización de datos de sensores enviados desde el dispositivo integrado

4.2.5 Discusión de resultados

En el caso del nodo sensor 1, es necesario agregar un filtro de datos para suavizar las lecturas de saturación de oxígeno obtenidas. Por otro lado, las señales obtenidas en el nodo sensor 2 son distintas a las tratadas en el análisis con Python, es por esta razón que se adaptó el programa del nodo para que los niveles de las señales sean similares a las obtenidas durante su análisis fuera de línea y minimizar los errores en la evaluación e implementación de los algoritmos.

Según lo visto para el procesamiento y cálculo de los valores de frecuencia respiratoria y pulso cardiaco es necesario que se cambie el tiempo de validación de QRS de 40 ms a 150 ms. En el caso de la validación de detección de la señal QRS, el valor de umbral es 50 lo que permite ayudar a detectar los picos verdaderos e incluso si los falsos picos fueran detectados, estos son eliminados. Esto corresponde al tiempo que debe superar el complejo QRS para ser considerado como tal (subida de 50 y bajada menor a 50.)

En el desarrollo del dispositivo de hardware o nodo sensor 2, se debe configurar el tiempo usado para transmitir los datos considerando que las muestras a enviar, en una ventana de tiempo, deben ser menores a 30 en un rango de tiempo de 60 segundos. En los ejemplos evaluados se tienen 5 segundos para un ciclo respiratorio y si el ciclo cardiaco ocurriera cada 700ms se obtiene aproximadamente 7 ciclos de pulso cardiaco

por ciclo respiratorio considerando que se tiene un rango de duración aproximado de 4 a 8 segundos. Si se realiza la transmisión en una ventana de tiempo de 12 segundos, se transmiten de 10 a 12 muestras. En el nodo sensor 2, se transmiten los datos totales de picos de la señal QRS, con una duración cercana a los 700ms en ventanas de 12 segundos.

Como trabajo futuro, se puede considerar el uso de algoritmos de aprendizaje automático para detectar automáticamente los valores de frecuencia respiratoria sin utilizar máquinas de estado y calcular su valor a partir de las señales de SPO2.

CAPÍTULO 5: DESARROLLO DE LA TÉCNICA COMPUTACIONAL EN EL DISPOSITIVO EMBEBIDO

Se presenta el desarrollo de la técnica computacional orientada al monitoreo eficiente de señales fisiológicas relacionadas a cardiopatía. La técnica se basa en la integración y evaluación del funcionamiento de modelos de aprendizaje automático integrados en el dispositivo embebido (nodo sensor), permitiendo la clasificación del estado de salud de un paciente ambulatorio en base a parámetros como temperatura corporal, frecuencia cardiaca, saturación de oxígeno (SpO_2) y frecuencia respiratoria, sin depender de infraestructura de procesamiento en la nube. Además, se considera el uso de métodos de preprocesamiento, donde las señales fisiológicas capturadas por los sensores son filtradas, adaptadas y transformadas para extraer las características que los modelos de clasificación del estado de salud utilizan.

El desarrollo de la técnica se organiza en dos fases, donde en primer lugar se describe una etapa de evaluación de modelos de clasificación, considerando métricas de exactitud, consumo energético, tiempo de inferencia y uso de memoria RAM y FLASH, usando datos sintéticos generados en el dispositivo embebido desarrollado en el capítulo 4. Posteriormente, como segunda etapa, se realiza el desarrollo, despliegue y validación final de la técnica computacional usando conjunto de datos encontrados en la literatura para el entrenamiento realizando las pruebas de hipótesis para validar el cumplimiento de los objetivos (Figura 51).

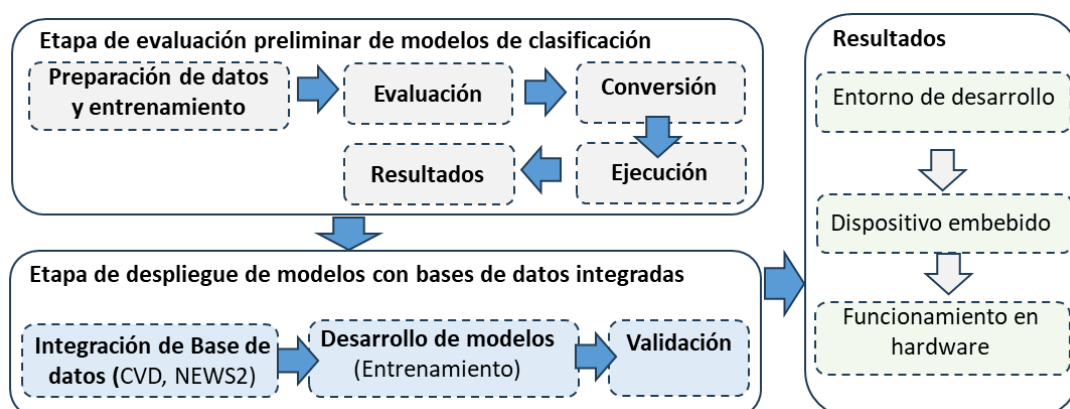


Figura 51. Diagrama del proceso de la técnica computacional

5.1 Etapa de evaluación preliminar de modelos de clasificación

En esta sección se describe el proceso de evaluación inicial de modelos de aprendizaje automático y redes neuronales para determinar del estado de salud, usando datos generados aleatoriamente en el dispositivo embebido (integrado con los dos nodos sensores descritos anteriormente) considerando los procesos de preparación y clasificación y para la evaluación computacional. Para ello se usan datos de temperatura, pulso cardíaco, SPO2 y frecuencia respiratoria (Figura 52). En este caso se consideran que se utiliza un solo dispositivo embebido producto de la integración de sensores y componentes descritos anteriormente en 2 nodos sensores (capítulo 4).

Los datos sintéticos empleados en esta etapa permiten evaluar el comportamiento computacional de los modelos de clasificación y no se utilizan para validar resultados clínicos, sino para la evaluación de la técnica computacional.

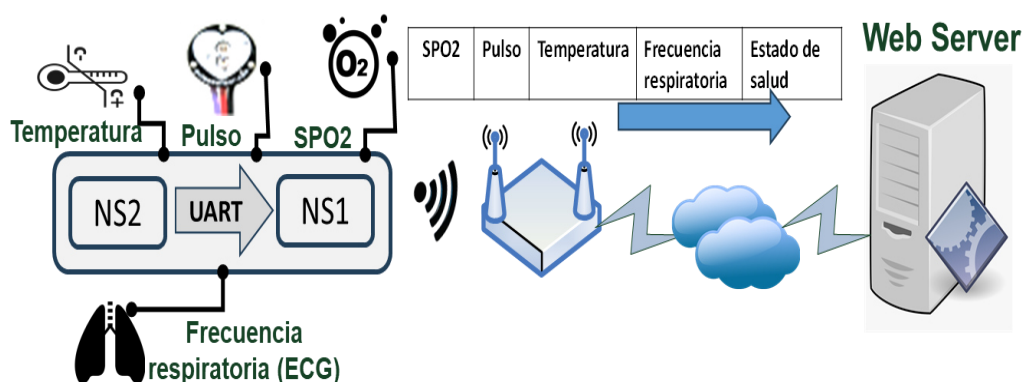


Figura 52. Esquema de la etapa de evaluación de modelos de clasificación

Las señales para evaluar el comportamiento de los modelos, tanto en hardware como el proceso de entrenamiento, son SPO2 (saturación de oxígeno), frecuencia respiratoria (rr), pulso y temperatura corporal (temp). Por otro lado, para las evaluaciones en esta sección se establecieron 4 estados para el nivel de la persona: estable (stable), leve (mild), moderado (moderate) y crítico (critical) (Figura 53).

Para esta primera evaluación se hace uso de un conjunto de datos utilizado como referencia trabajos de investigación que indican las puntuaciones para las diferentes condiciones del estado de salud según las señales fisiológicas relacionadas a cardiopatía. Los valores asignados en la literatura (Huespe et al., 2022b)(Liao et al.,

2020b) indica 0 para un valor estable, 1 para leve, 2 es moderada y la puntuación 3 indica crítica. En este caso se ha tomado los valores de puntuación donde, por ejemplo, el estado crítico indica que los valores de la persona están con niveles de SPO2 entre 80% y 90% indicando que necesita atención médica (Tabla 23).

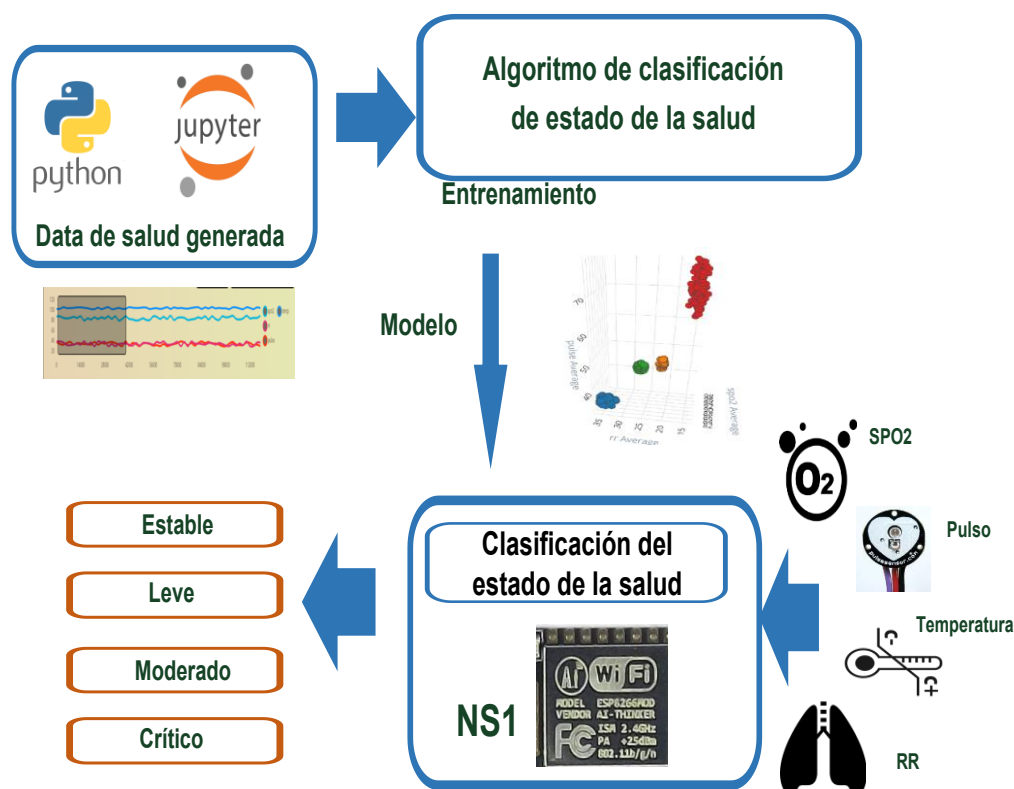


Figura 53. Procedimiento de implementación para la etapa de evaluación de modelos

Tabla 23. Rangos de valores para estados de salud

Parámetro	Estado			
	estable	leve	Moderado	Crítico
SPO2 (%)	96 - 100	93 - 95	90 - 92	80 - 90
Frecuencia respiratoria (respiraciones por minuto)	12- 20	21- 24	25 - 30	30 - 40
Pulso (latidos por minuto)	51 - 90	41 - 50	41 - 50	30 - 40
Temperatura corporal (°Fahrenheit)	96 - 100	100 - 102	102 - 106	102 - 106

Para evaluar el comportamiento y la generalización de los modelos, se realiza la simulación de casos extremos para medir la respuesta a escenarios no contemplados luego de haber sido entrenados. Los nuevos datos generados corresponden a valores

diversificados y realistas que representen los rangos normales y variaciones (Tabla 24).

Tabla 24. Rango de datos generados para cada parámetro

Parámetro	Rango de valores
SPO2 (%)	75-100
Frecuencia respiratoria (respiraciones por minuto)	5-30
Pulso (latidos por minuto)	30-90
Temperatura corporal (°Fahrenheit)	90-110

Para desarrollar los modelos se consideran las etapas de creación e implementación del modelo en el dispositivo embebido (nodo sensor integrado). Además, se ha determinado el flujo de trabajo en base a implementaciones descritas en sistemas embebidos para aplicaciones del computación de Borde y TinyML (Situnayake, 2020) (Situnayake & Plunkett, 2023)(Lea & Safari, 2020)(Iodice, 2022) (Warden & Situnayake, 2019) donde se describen metodologías conocidas para el desarrollo de modelos como KDD (Głowania et al., 2023) y CRIPS-DM (Purbasari et al., 2021) (Figura 54).

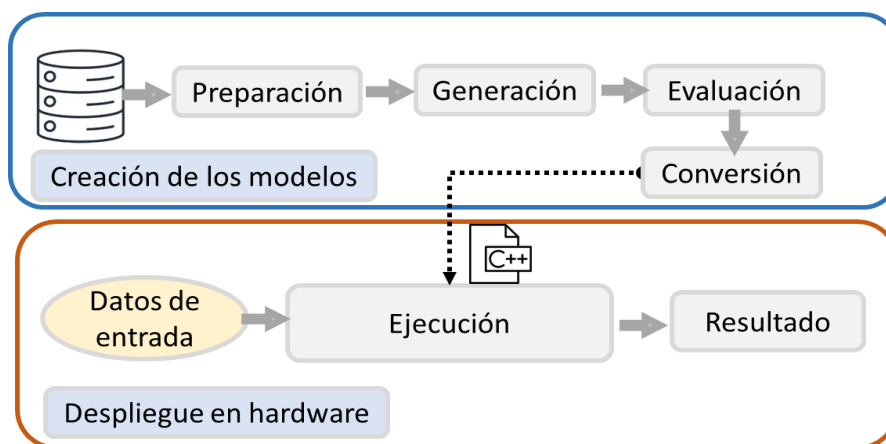


Figura 54. Flujo de trabajo para el desarrollo de los modelos

5.1.1 Preparación de datos y entrenamiento

Para realizar la evaluación de los modelos se consideran las siguientes configuraciones de implementación de modelos en el dispositivo embebido:

a) Configuración 1 (C1) . Se utiliza los modelos de clasificación encontrados en la literatura científica de manera recurrente: Árboles de decisión (DT), Bosques aleatorios (RF), Máquinas de Vector Soporte para clasificación (SVM/SVC) y Bayes (Naive Bayes o GNB).

Para la preparación de datos y generación de modelos se sigue la siguiente secuencia de pasos: Recopilación de datos (Lectura del conjunto de datos que contienen información acerca del estado de salud), selección de características (Se seleccionan los 4 parámetros del estado de la salud sin realizar el proceso de normalización y escalado) y división de datos (Se divide el conjunto de datos en 70% para entrenamiento y 30% para datos de prueba).

b) Configuración 2 (C2). En este caso, a diferencia del caso anterior, se extraen las características en ventanas de tiempo usando secuencia de registros contiguas en lugar de un solo registro independiente. Para ello se realiza las evaluaciones de los siguientes modelos de aprendizaje automático DT, RF, SVM y GNB considerando datos alterados con parámetros diversificados:

- **Recopilación de datos.** Se realiza la lectura de los archivos que contienen los datos de los parámetros a evaluar.
- **Selección de características.** En este caso se realiza la extracción y cálculo de características usando una ventana deslizante para analizar series temporales. Las características extraídas son: valor mínimo, valor máximo, valor absoluto mínimo, valor absoluto máximo, valor promedio, desviación estándar, cantidad de valores por encima del valor promedio y por debajo.
- **División de datos:** Se seleccionan datos de entrenamiento y datos de prueba siendo su configuración del 70% para prueba y el 30%.

Finalmente, se realiza la generación y entrenamiento para cada modelo de la misma manera que se hizo para la configuración 1, considerando en este caso las características generadas al aplicar la ventana.

c) Configuración 3 (C3). Se evalúan modelos basados en redes neuronales usando el algoritmo para extracción de características mediante la método de ventanas de tiempo usado anteriormente. En este caso se usan cuatro variaciones del modelo de una red

neuronal totalmente conectada, donde el primer valor indica el número neuronas de entrada, luego el número de neuronas en cada capa oculta y las neuronas de salida:

- Modelo 1 (RN1). (32,32,16,8,4,4).
- Modelo 2 (RN2). (32,16,8,4,4)
- Modelo 3 (RN3). (32,8,4,4).
- Modelo 4 (RN4). (32,8,4).

De acuerdo con el modelo establecido se siguen los mismos pasos de recopilación de datos (datos con los parámetros del estado de salud), selección de características (usando la ventana deslizante) y división de datos (70% de datos para entrenamiento y 30% para datos de prueba), agregando las etapas de:

- **Codificación de salidas:** Se transforman las etiquetas categóricas en formatos numéricos tanto para los conjuntos de datos de entrenamiento y prueba.
- **Escalamiento de datos de entrada:** Se escalan los datos de entrenamiento y prueba, y se mapean dentro de un rango definido entre -1 a 1. En el caso de modelos basados en redes neuronales, el escalado de los datos de entrada es una etapa importante para el entrenamiento y no sólo la optimización relacionada a la implementación en hardware embebido.

La normalización de las características uniformiza los rangos de las variables fisiológicas, aportando a la estabilidad del descenso por gradiente a través de las funciones de activación utilizadas en arquitecturas basadas en redes neuronales. En este caso, se utiliza un proceso de escalado de datos previo a la inferencia, para adaptar los rangos numéricos de las señales fisiológicas a las limitaciones de los dispositivos embebidos de bajos recursos, debido a que valores con alta precisión decimal puede incrementar el uso de memoria RAM y FLASH y el tiempo de procesamiento, además de generar riesgos de desbordamiento numérico. Es por ello que el escalado de datos disminuye estos efectos, y permite realizar el proceso de inferencia más adecuado en términos de consumo de recursos de hardware y velocidad de ejecución.

Posteriormente, en la etapa de generación de los modelos, se entrenan y obtienen cada uno de ellos siguiendo los procesos mencionados anteriormente (Figura 55). Las capas ocultas utilizan funciones de activación “relu” y la salida “sigmoid” y la red se

configura disminuyendo el número de neuronas de entrada de 32 a 4 de salida a lo largo de 200 épocas.

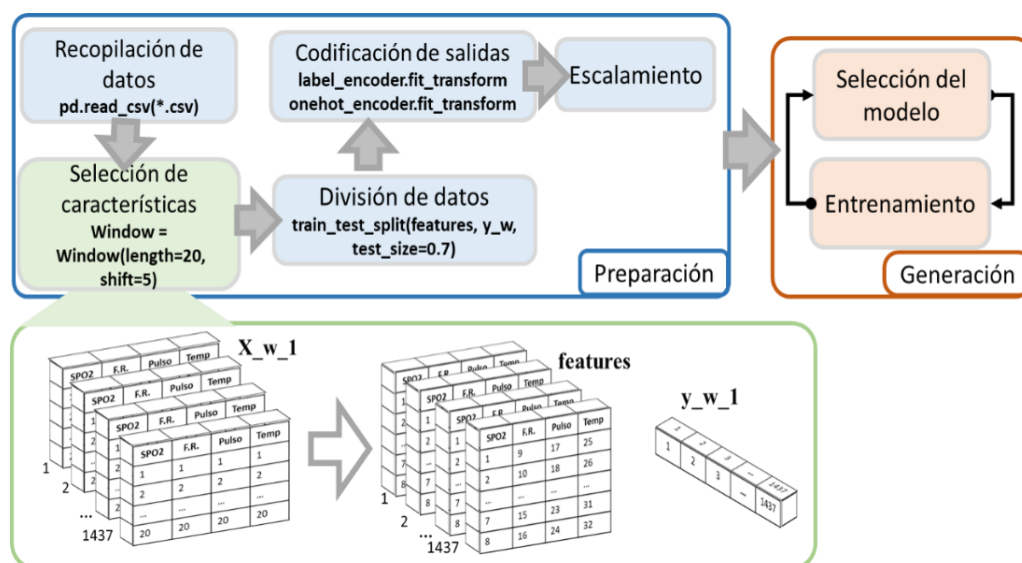


Figura 55. Preparación y generación de modelos

5.1.2 Evaluación

Se usa la matriz de confusión para obtener el detalle de los aciertos y errores del modelo en la clasificación de cada clase calculando métricas como la exactitud en el entorno de software (Tabla 25) (Géron, 2017).

Tabla 25. Resultados de la evaluación de los modelos con datos aleatorios

Parámetro	Modelos para configuración 1 y 2	Exactitud Configuración 1	Exactitud configuración 2	Modelos para configuración 3	Exactitud configuración3
SPO2	Decision Tree	0.255	0.75	RN1	0.748
	Random Forest	1	0.999	RN2	0.997
	SVC	0.636	0.976	RN3	0.786
	GaussianNB	0.529	0.249	RN4	0.754
rr	Decision Tree	1	0.999	RN1	0.752
	Random Forest	0.958	0.998	RN2	0.498
	SVC	0.555	0.5	RN3	0.498
	GaussianNB	0.732	0.498	RN4	0.757
Pulso	Decision Tree	1	0.999	RN1	0.496
	Random Forest	1	0.983	RN2	0.432
	SVC	0.509	0.253	RN3	0.374
	GaussianNB	0.726	0.994	RN4	0.249
Temp	Decision Tree	1	0.504	RN1	0.751
	Random Forest	1	0.751	RN2	0.998
	SVC	0.895	0.997	RN3	0.998
	GaussianNB	0.801	0.752	RN4	0.995

5.1.3 Conversión

Para la transformación de los modelos utilizando la configuración 1 (T1) y configuración 2 (T2), se utilizó el framework Micromlgen obteniendo un archivo en código C/C++ que representa el modelo generado. Para ello la instrucción **"from micromlgen import port"** se utiliza para implementar modelos de aprendizaje automático en dispositivos de recursos de hardware limitados. (Figura 56).

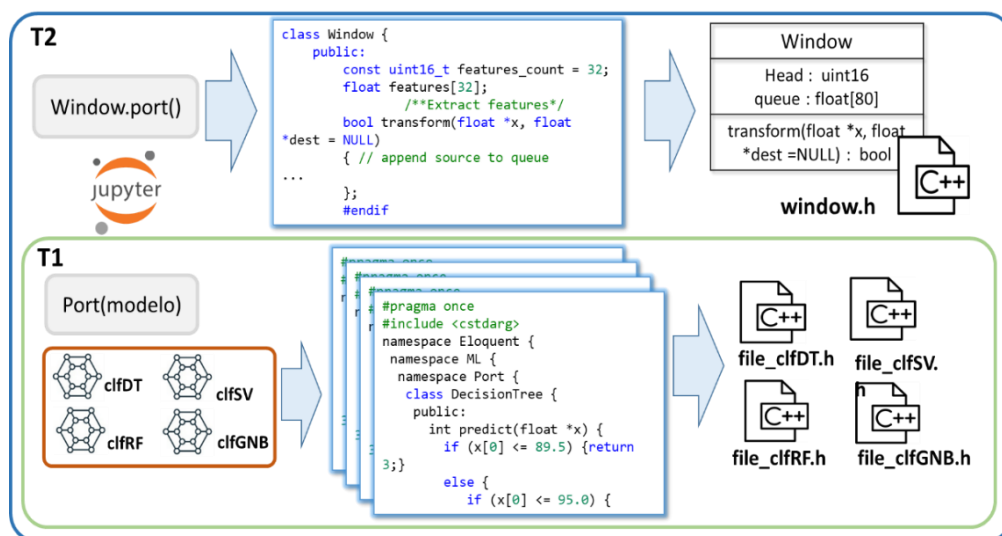


Figura 56. Proceso de conversión para configuración 1 y 2

En el caso de los modelos basados en redes neuronales se utilizó el framework de Tensorflow para generar el código en C/C++. La librería carga el modelo (`loaded_model`) y lo convierte a un modelo TensorFlow Lite (TFLite) utilizando el `TFLiteConverter`. Después de la conversión, el modelo TFLite se guarda en un archivo con extensión `"tflite"`. Finalmente, se utiliza el comando `xxd` para generar una representación hexadecimal del modelo TFLite (Figura 57).

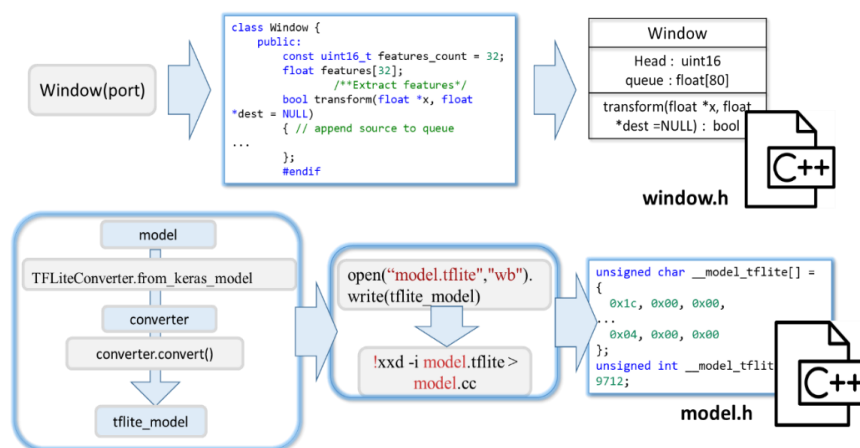


Figura 57. Proceso de conversión de los modelos

5.1.4 Ejecución en el dispositivo embebido

En esta sección se describe el proceso de despliegue de los modelos de clasificación en el dispositivo embebido para cada configuración.

a) Ejecución de los modelos con la configuración 1. Se seleccionan 20 muestras espaciados cada 300 elementos usando el desplazamiento de datos los cuales sirven como entrada a los modelos desplegados en el firmware. El firmware del dispositivo de hardware (nodo sensor) usa los datos obtenidos anteriormente en una matriz 2D (20 filas y 4 columnas).

b) Ejecución de los modelos usando la configuración 2. Se integra al firmware el algoritmo de extracción de características con ventanas de tiempo. Para ello se seleccionan 65 filas de entrada y salida. Luego se crea un objeto “Window” con una longitud de ventana con 20 datos aplicando la transformación de ventana a las variables “Xm” e “ym” obteniendo como resultado “Xm_w” con la totalidad de las ventanas. Además, se generan las 8 características para cada una de las ventanas, almacenadas en la variable “features_m_w” (Figura 58).

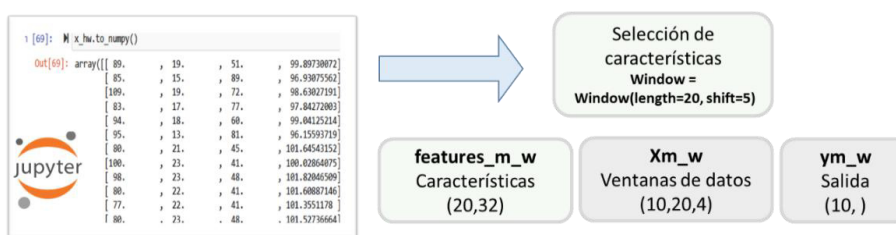


Figura 58. Conjunto de datos generados para evaluación en el nodo sensor

Los datos se almacenan en una matriz de 3 dimensiones (10 ventanas de 20 filas que contienen 4 columnas), de los cuales luego se extraen las características. Posteriormente se realizan las predicciones y estos resultados son enviados por el puerto de comunicación serial hacia una computadora.

c) Proceso de despliegue de los modelos de clasificación usando la configuración

3. Se realiza en el dispositivo embebido, usando el método de ventana para extraer las características de la misma forma que en el paso anterior, pero en este caso, los resultados de predicción se obtienen con las características escaladas. El programa realiza el proceso de características, normalización, predicción y visualización de los datos a través del puerto serie.

5.1.5 Resultados

Se obtienen métricas de evaluación de los modelos durante el entrenamiento, pruebas y despliegue en el dispositivo embebido integrado. Para ello se hizo uso de los datos con parámetros diversificados obteniendo 16 resultados de evaluación en el dispositivo embebido para las 4 variaciones de señales y los 4 tipos de modelos generados.

a) En el caso de la configuración 1 se realizó la evaluación de los modelos durante su entrenamiento, observando la exactitud promedio y el tamaño del archivo que contiene los modelos generados (Tabla 26).

Tabla 26. Características de los modelos evaluados con nuevos datos

Modelo	Exactitud	Tamaño (KB)
Decision Tree	0.81375	1.02
Random Forest	0.9895	7.57
SVC	0.64875	39.2
GaussianNB	0.697	3.75

Se obtienen los resultados del comportamiento de los modelos usando el entorno de software durante el proceso de validación, considerando sólo muestras seleccionadas. Estos datos se utilizan como entrada al dispositivo embebido (nodo sensor integrado) para su clasificación (Figura 59).

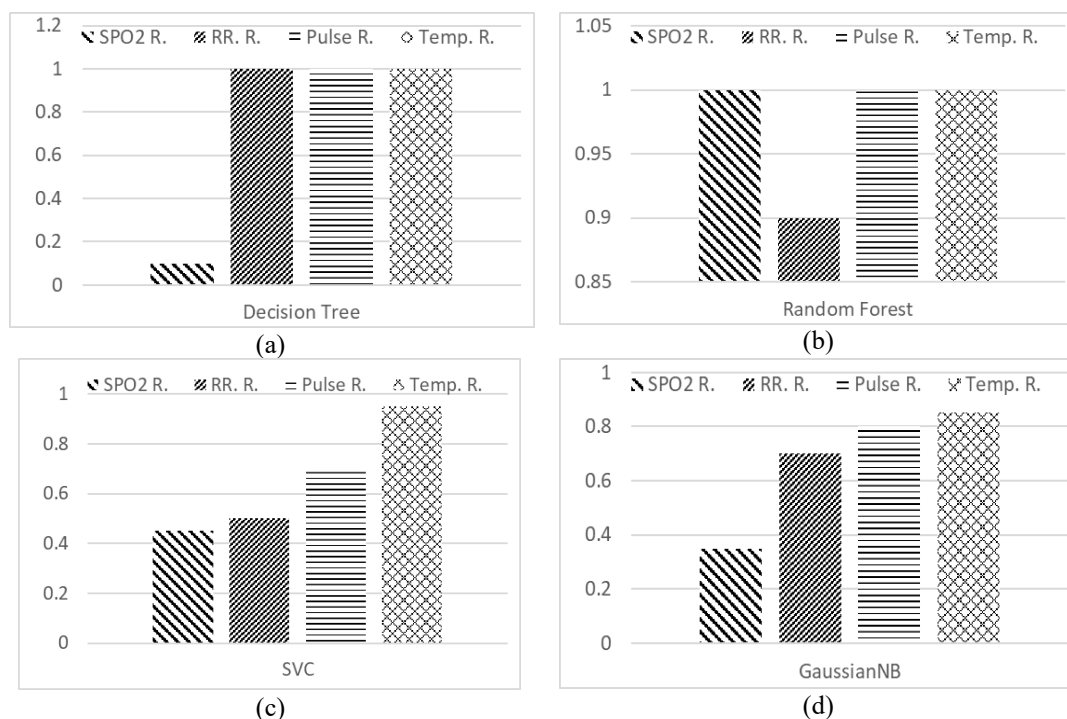


Figura 59. Exactitud promedio de los modelos evaluados de (a) árbol de decisión, (b) Bosques aleatorios, (c) SVM y (d) Bayes (GaussianNB) con muestras seleccionadas

Se implementan los modelos en el dispositivo embebido teniendo como resultados datos de exactitud, el consumo de la memoria de programa FLASH, la memoria dinámica RAM y el tiempo de clasificación (Tabla 27). En el sistema embebido, la memoria FLASH se empleó para el almacenamiento del firmware y del modelo entrenado, mientras que la memoria RAM se usa durante la ejecución del algoritmo, incluyendo la carga de variables y arreglos de dato. En este contexto, la optimización del consumo de memoria RAM es crítica, ya que su disponibilidad es limitada. Por esta razón, en los resultados presentados a continuación se reportan de manera separada el consumo de memoria FLASH y RAM para cada modelo evaluado.

Tabla 27. Resultados de comportamiento de los modelos en el dispositivo embebido

	Modelo	Exactitud	Memoria FLASH %	Memoria FLASH (3145728) Bytes	Memoria RAM %	Memoria RAM (327680) Bytes	Tiempo de inferencia (us)
SPO2	Decision Tree	0.5	8	270026	4	14044	0.95
	Random Forest	1	8	271134	4	14052	2.65
	SVC	0.4	9	292286	4	14056	4021.6
	GaussianNB	0.3	8	272138	4	14056	206

rr	Decision Tree	0.9	8	269998	4	14044	1
	Random Forest	0.85	8	270338	4	14020	3.5
	SVC	0.5	9	292286	4	14056	4014.75
	GaussianNB	0.75	8	271342	4	14024	206.4
Pulso	Decision Tree	1	8	270026	4	14044	0.9
	Random Forest	1	8	271134	4	14052	2.85
	SVC	0.65	9	292286	4	14056	4172.1
	GaussianNB	1	8	272138	4	14056	205.95
Temp	Decision Tree	0.85	8	270042	4	14044	0.95
	Random Forest	1	8	271134	4	14052	2.65
	SVC	0.9	9	292286	4	14056	4000.8
	GaussianNB	0.8	8	272138	4	14056	206

Se muestran los resultados obtenidos durante el entrenamiento (Exactitud SW TD), resultados de exactitud con datos aleatorios diversificados en el software de evaluación (Exactitud SW) y resultados en el dispositivo embebido (Exactitud HW) (Figura 60).

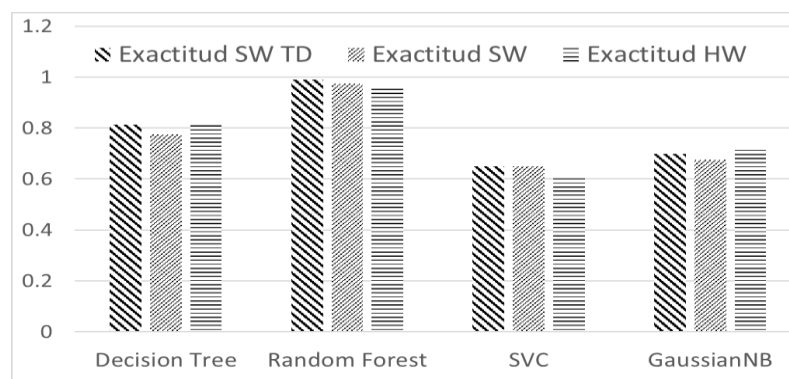


Figura 60. Comparación de resultados de exactitud usando la primera configuración

b) Etapa de entrenamiento, en base los procesos indicados en la configuración 2, se obtienen los valores de exactitud promedio (para los 4 resultados obtenidos del modelo) y el tamaño del archivo para los modelos generados (Tabla 28).

Tabla 28. Características de los modelos evaluados con nuevos datos

Modelo	Exactitud	Tamaño (KB)
Decision Tree	0.813	1.04
Random Forest	0.93275	7.94
SVC	0.6815	19.8
GaussianNB	0.62325	8.64

Considerando el uso los datos seleccionados se evalúa el comportamiento de los modelos usando la plataforma de software obteniendo resultados de exactitud según los datos de señales modificadas de manera diversificada (Figura 61).

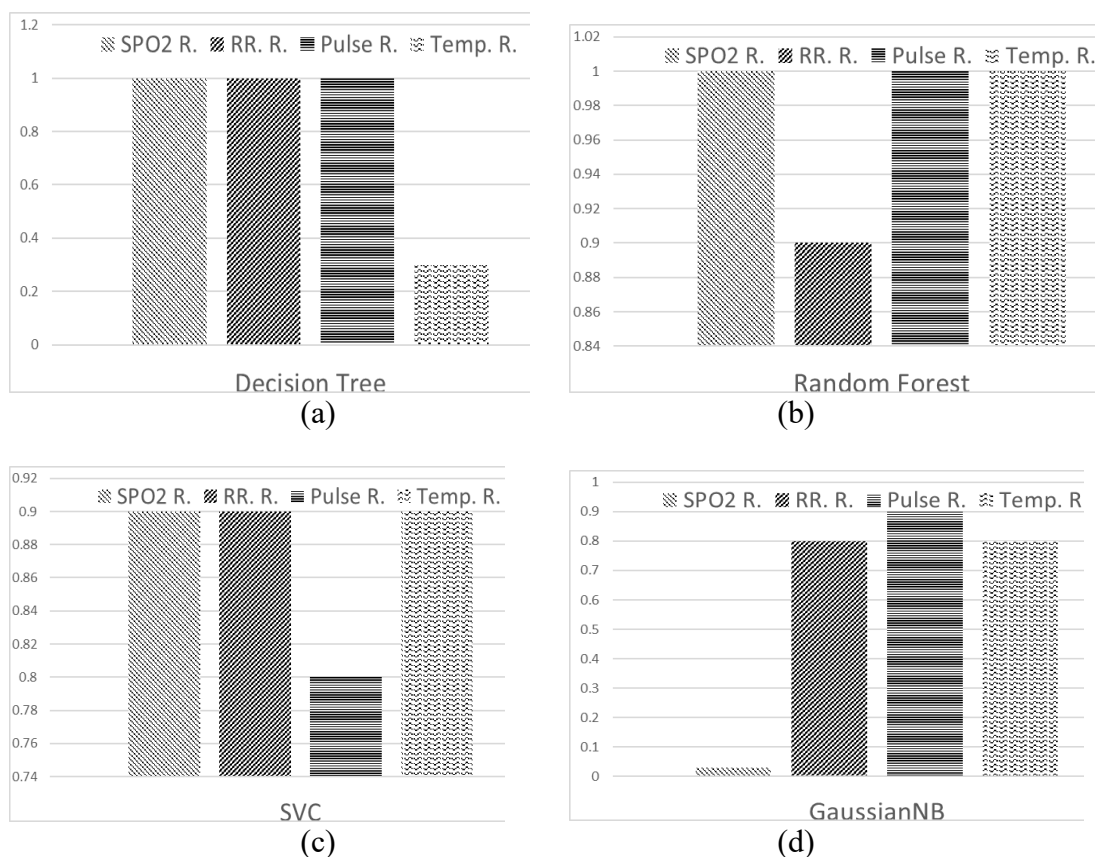


Figura 61. Exactitud de modelos evaluados de (a) árbol de decisión, (b) Bosques aleatorios, (c) SVM y (d) Bayes (GaussianNB)

Posteriormente se realiza el despliegue de modelos entrenados en el dispositivo embebido obteniendo valores de exactitud, el consumo de la memoria de programa, la memoria dinámica y el tiempo de inferencia (tiempo para realizar la clasificación), utilizando los datos previamente seleccionados (Tabla 29).

Tabla 29. Comportamiento de los modelos en el dispositivo embebido usando la configuración 2

	Modelo	Exactitud	Memoria FLASH %	Memoria FLASH (3145728) Bytes	Memoria RAM %	Memoria RAM (327680) Bytes	Tiempo de inferencia (us)
SPO2	Decision Tree	1	8	274782	5	17360	0.9
	Random Forest	1	8	276006	5	17368	4
	SVC	1	9	292910	5	17368	3.7

	GaussianNB	0.8	8	278498	5	17368	1685.4
rr	Decision Tree	1	8	274782	5	17360	1
	Random Forest	0.9	8	276006	5	17368	3.7
	SVC	0.9	9	292910	5	17368	2191.4
	GaussianNB	0.8	8	278498	5	17368	1683.3
Pulso	Decision Tree	1	8	274782	5	17360	1
	Random Forest	1	8	276006	5	17368	3.6
	SVC	0.8	9	292910	5	17368	2211
	GaussianNB	0.9	8	278498	5	17368	1683.9
Temp	Decision Tree	0.3	8	274782	5	17360	1
	Random Forest	1	8	276006	5	17368	4
	SVC	0.9	9	292910	5	17368	2192.6
	GaussianNB	0.8	8	278498	5	17368	1684.1

Finalmente se realiza la comparación de los valores de exactitud considerando los datos totales para entrenamiento (Exactitud SW TD), datos seleccionados (Exactitud SW) y la exactitud obtenida con los resultados de clasificación en el nodo sensor (Exactitud HW) (Figura 62).

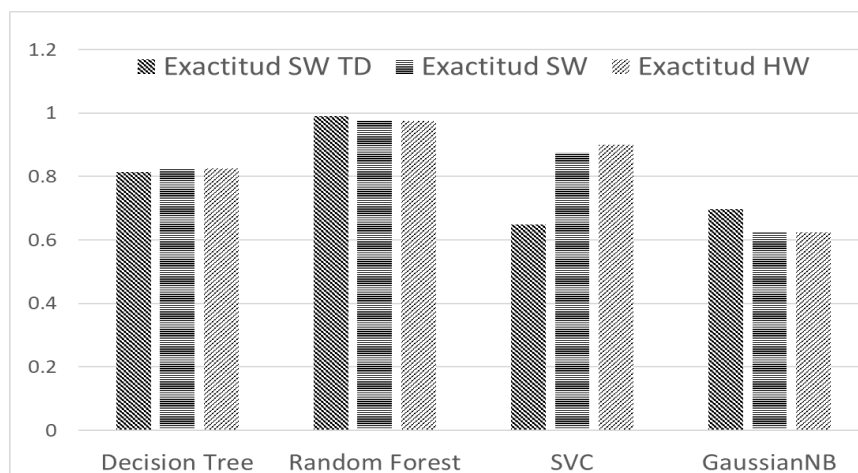


Figura 62. Comparación de resultados de exactitud usando la configuración 2

c) Usando los modelos basados en redes neuronales (configuración 3) se obtuvieron resultados durante el entrenamiento, observando la exactitud promedio (para las 4 variaciones de datos con parámetros aleatorios) y el tamaño del archivo para los modelos generados (Tabla 30).

Tabla 30. Resultado de evaluación de modelos con configuración 3

Modelo	Exactitud promedio (Accuracy)	Desviación estándar	Tamaño (KB)
32,8,4	0.68675	0.1101394	16.5
32,8,4,4	0.73125	0.26727081	19.2
32,16,8,4,4	0.664	0.24397541	31.3
32,32,16,8,4,4	0.68875	0.27206835	59.3

Usando las muestras seleccionadas se obtienen los resultados de exactitud para los 4 tipos de modelos (Figura 63).

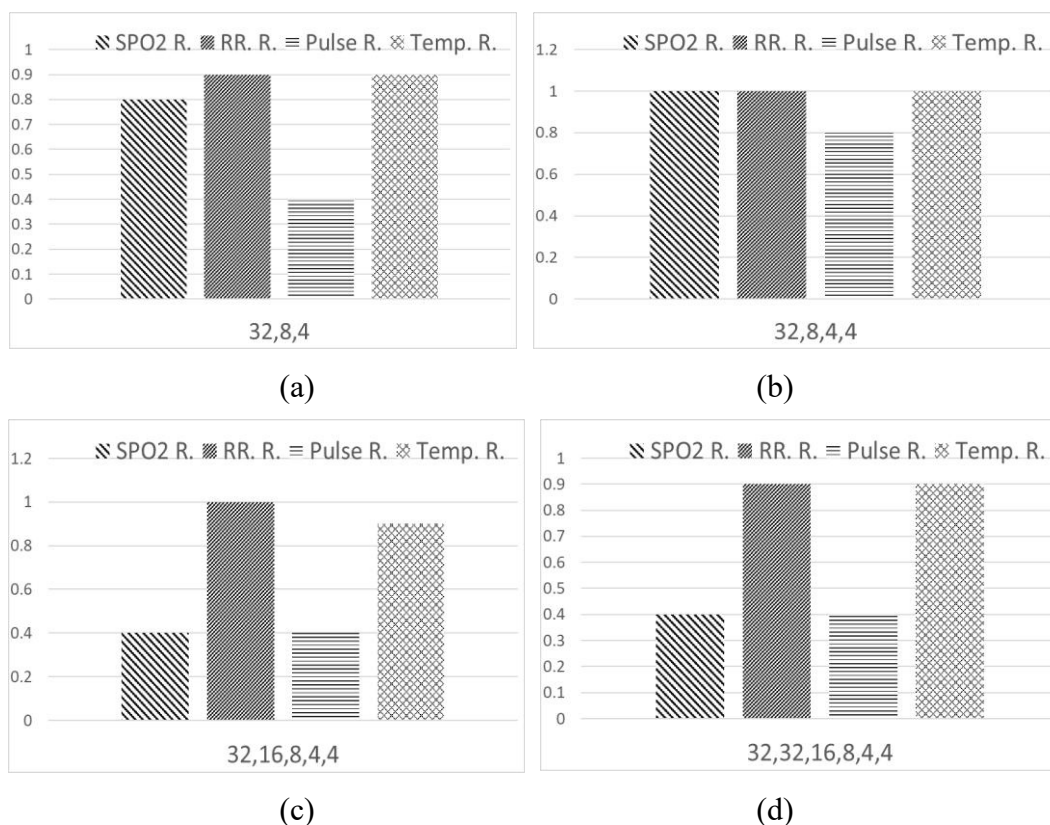


Figura 63. Exactitud de modelos evaluados en software con muestras seleccionadas para los modelos de (a) ARN1, (b) ARN2, (c) ARN3 y (d) ARN4

En el dispositivo embebido se cargan los modelos y los datos mencionados anteriormente obteniendo los resultados de exactitud, uso de la memoria de programa, memoria dinámica y el tiempo para realizar la clasificación en el nodo sensor (Tabla 31).

Tabla 31. Comportamiento de los modelos en el dispositivo embebido usando la configuración 3

Modelo	Exactitud	Memoria FLASH %	Memoria FLASH	Memoria RAM %	Memoria RAM	Tiempo de
--------	-----------	-----------------	---------------	---------------	-------------	-----------

				(3145728) Bytes		(327680) Bytes	inferenci a (us)
SPO2	32,8,4	0.9	15	472298	15	49280	303.9
	32,8,4,4	1	15	472738	15	49280	312.5
	32,16,8,4,4	0.5	15	474706	15	49280	395
	32,32,16,8,4,4	0.6	15	479314	15	49280	585.6
rr	32,8,4	0.9	15	472290	15	49280	280.7
	32,8,4,4	1	15	472738	15	49280	312.7
	32,16,8,4,4	1	15	474706	15	49280	396.5
	32,32,16,8,4,4	0.9	15	479314	15	49280	585.9
Pulso	32,8,4	0.4	15	472290	15	49280	280.7
	32,8,4,4	0.8	15	472738	15	49280	313.6
	32,16,8,4,4	0.4	15	474706	15	49280	395.2
	32,32,16,8,4,4	0.4	15	479314	15	49280	587.2
Temp	32,8,4	0.9	15	472290	15	49280	280.3
	32,8,4,4	1	15	472738	15	49280	313.5
	32,16,8,4,4	1	15	474706	15	49280	396.9
	32,32,16,8,4,4	0.9	15	479314	15	49280	586.7

En la Figura 64 se muestran los valores promedios de exactitud obtenidos en el software de análisis (SW TD) usando los datos totales y resultados con los datos seleccionados en el dispositivo embebido (HW)

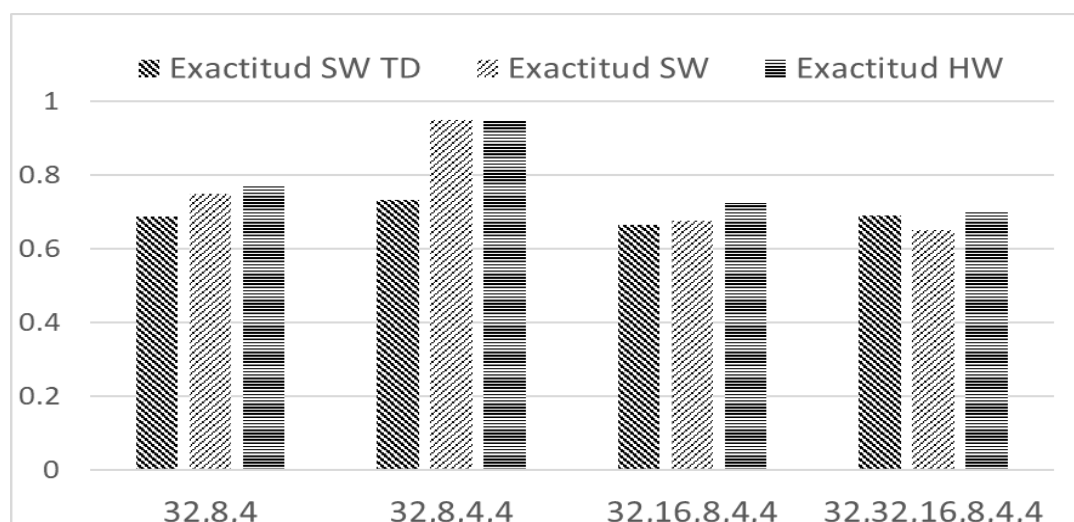


Figura 64. Comparación de resultados de exactitud usando la configuración 3

5.1.6 Discusión de resultados

En el caso de la configuración 1 se observa que los árboles de decisión tienen una mayor estabilidad (superando el 85% de exactitud en todos los casos) mientras que cuando se usan los valores atípicos de la señal SPO2 se reduce significativamente la exactitud de los modelos a excepción de los bosques aleatorios. Por otro lado, los

archivos de modelos generados para el algoritmo de árboles decisiones son más ligeros, mientras que los SVMs, generan archivos de mayor tamaño. Durante la evaluación en el dispositivo embebido, se determinó que el SVM es el que utiliza más memoria de programa (9%), la exactitud del árbol de decisión se mantiene estable ante los cambios del SPO2 y los bosques aleatorios tienen un mejor comportamiento de clasificación con los datos de entrenamiento. Por otro lado, durante la evaluación del tiempo de clasificación se obtuvo que el árbol de decisión consume menor tiempo (0.95 microsegundos).

En el proceso de entrenamiento realizado con la configuración 2, el modelo RF tuvo un mejor comportamiento en los resultados de exactitud mientras que en el dispositivo embebido se obtuvo que el modelo RF obtuvo un 0.975 de exactitud seguido del DT con 0.825. Si se compara el comportamiento del modelo en validación y en el dispositivo embebido, el SVC mejora su exactitud mientras que el modelo RF es el más estable.

En el caso de la configuración 3, se tiene que el Modelo 2 tiene un mayor valor de exactitud, tanto en la etapa de entrenamiento como en la etapa de prueba, seguido del modelo 1 con 99%. Durante el proceso de conversión, el modelo 2 es un 50% menor en tamaño que el modelo 1. Por lo tanto, el modelo 3 se recomienda para implementarse en un hardware considerando el tamaño del modelo y su exactitud. En el caso de los resultados agregando datos aleatorios, se observa que el ingreso de variables sintéticas de temperatura no afecta más del 0.1% la variación de la exactitud, la señal SPO2 tiene una exactitud que no supera el 78%. Se observa que no necesariamente los modelos más complejos tienen una mayor exactitud en hardware, concluyendo que el modelo 3 tiene la mejor compensación en tiempo y exactitud.

Aunque algunos modelos tienen un mayor consumo de memoria FLASH por el tamaño del archivo generado, el criterio crítico para su implementación en el dispositivo embebido es también la memoria RAM durante la inferencia, dado que afecta la ejecución del modelo.

5.2 Etapa de despliegue de modelos con bases de datos integradas

En esta sección se describe el proceso de desarrollo, implementación y pruebas de modelos en el dispositivo embebido. En este contexto, se aborda la adaptación de un

conjunto de datos, a través de métodos de preprocesamiento diferenciado, las cuales corresponden a los mismos rangos de las señales obtenidas de los sensores para obtener las características más relevantes antes de ser enviados al modelo final de predicción (Figura 65).

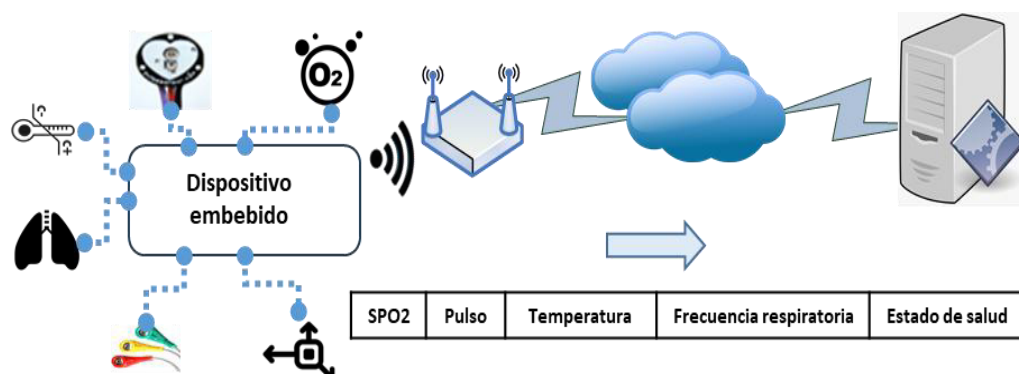


Figura 65. Esquema de despliegue del proceso de clasificación

Para esta etapa se emplean bases de datos fisiológicas reales provenientes de repositorios científicos, las cuales son de acceso público y se encuentran debidamente anonimizadas. Estas bases de datos se utilizan con fines experimentales para evaluar el desempeño de la técnica computacional, por lo que no se requiere consentimiento informado. Además es necesario indicar que durante el desarrollo del prototipo de hardware se usaron datos para validación técnica con señales del propio autor.

Se construyó un conjunto de datos propio, sobre el cual se aplicaron procesos de tratamiento de datos antes de la evaluación de los modelos de clasificación. Dichos procesos incluyen la detección y eliminación de valores atípicos (outliers), la transformación de variables y el balanceo de clases. El tratamiento de outliers se realizó mediante un criterio estadístico orientado a reducir el efecto del ruido o lecturas anómalas propias de la adquisición de señales fisiológicas.

5.2.1 Determinación de tecnologías

a) Base de datos. Considerando la revisión de la literatura se selecciona la base de datos de CVD-Vital-Signs (Louis Uzoegwu Farah, 2023)(Chidozie, 2022) y la base de datos que se genera a partir de la categorización del estado de salud de NEWS2 (Huespe et al., 2022a) (Liao et al., 2020a). La base de datos CVD-Vital-Signs está basada en el conjunto de datos MIMIC-III, etiquetada considerando los rangos fisiológicos (Zhao

et al., 2023) y tiene 210000 registros de cinco parámetros corporales fisiológicos (frecuencia cardíaca, la frecuencia de respiración, la temperatura, la saturación de oxígeno y la presión arterial) cuyos rangos normales (Lindsey, 2022) y fuentes médicas (Jacob et al., 2022) (Sapra et al., 2023) suelen estar dentro de parámetros específicos (frecuencia cardíaca - 60-100 latidos por minuto, frecuencia respiratoria - 12-18 respiraciones por minuto, presión arterial - 120/80 mmHg, saturación de oxígeno - 95-100%, y temperatura corporal - 98.6° F (37°C)), mientras que los niveles de saturación de oxígeno anormal están por debajo del 95% (Windisch et al., 2023)(Bhardwaj et al., 2022). Además, la salida está condicionada a dos estados: "Saludable" (0) y "No saludable" (1), la cual será adaptada posteriormente para tres niveles de riesgo (Figura 66).

En el caso de la escala NEWS2, surge como un esfuerzo por crear un conjunto de datos representativo de signos vitales (Liao et al., 2020a) que pueda ser utilizado para entrenar modelos de inteligencia artificial orientados a predecir el deterioro clínico en pacientes, considerando 4 niveles de riesgo y 7 tipos de señales (la frecuencia respiratoria, la saturación de oxígeno, calidad de aire, presión arterial, pulso, estado neurológico y temperatura) (Huespe et al., 2022a). La distribución de los datos de cada variable de NEWS se observa en la Figura 67. Ambas fuentes de datos son integradas en un sólo conjunto de datos con tres categorías de salida de severidad relacionadas a la salud o nivel de riesgo (0- normal o estable, 1- leve y 2- crítica) para el desarrollo de los modelos de clasificación. En el caso del conjunto de datos CVD-Vital-Signs, sólo se consideró los datos procesados incluyendo el tratamiento de outliers, transformación y balanceo (sin escalado). Por otro lado, para el conjunto de datos de NEWS2, se consideró el tratamiento de datos hasta la etapa de outliers (sin balanceo y sin escalado). Una vez integrados, el nuevo conjunto de datos es tratado por una etapa de eliminación de duplicados, tratamiento de outliers, y un preprocesamiento para el tratamiento del desbalanceo (Figura 68). En esta última etapa se eliminan los datos de la clase más alta (submuestreo) y un sobre muestreo de la clase más baja usando SMOTE (sobre muestreo de minorías sintéticas).

Durante la construcción del conjunto de datos, se aplicó un método estadístico basado en el rango intercuartílico (IQR) en todas las etapas para la identificación de valores atípicos en las características extraídas. Para cada variable se calculó el primer cuartil (Q1) y el tercer cuartil (Q3), definiendo el rango como $IQR = Q3 - Q1$. Se consideran

como valores atípicos aquellos que se encuentran fuera del intervalo $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$. Este criterio eliminó mediciones extremas a errores de adquisición o ruido. De esta manera se mejoró la estabilidad del conjunto de datos.

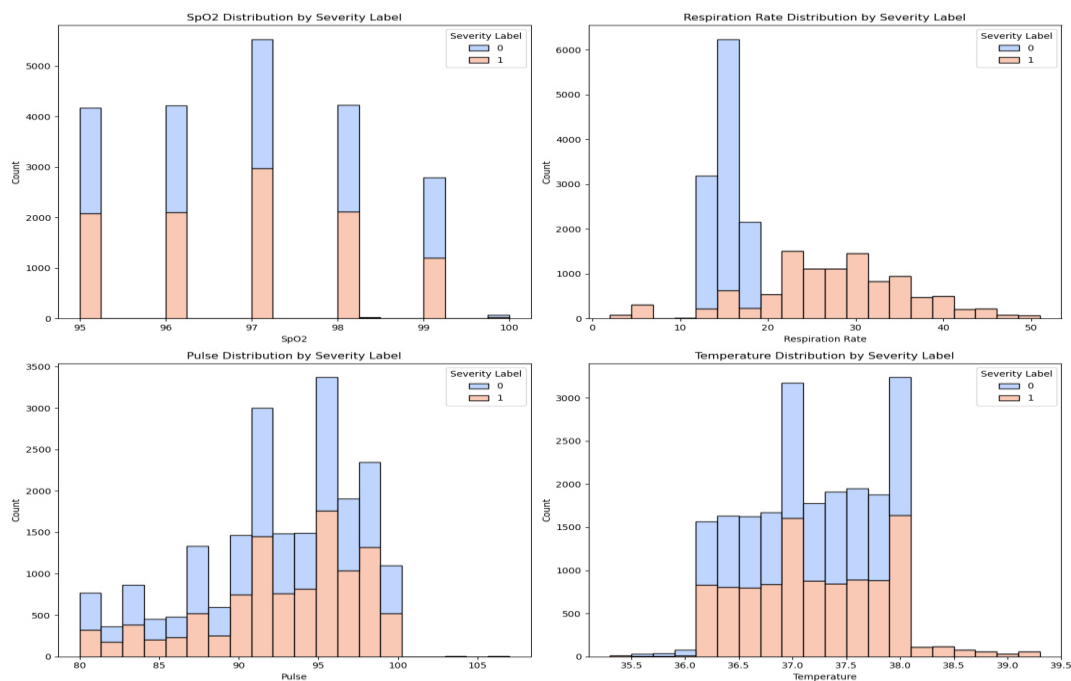


Figura 66. Base de datos CVD-Signal-Signs y Distribución de datos de cada variable

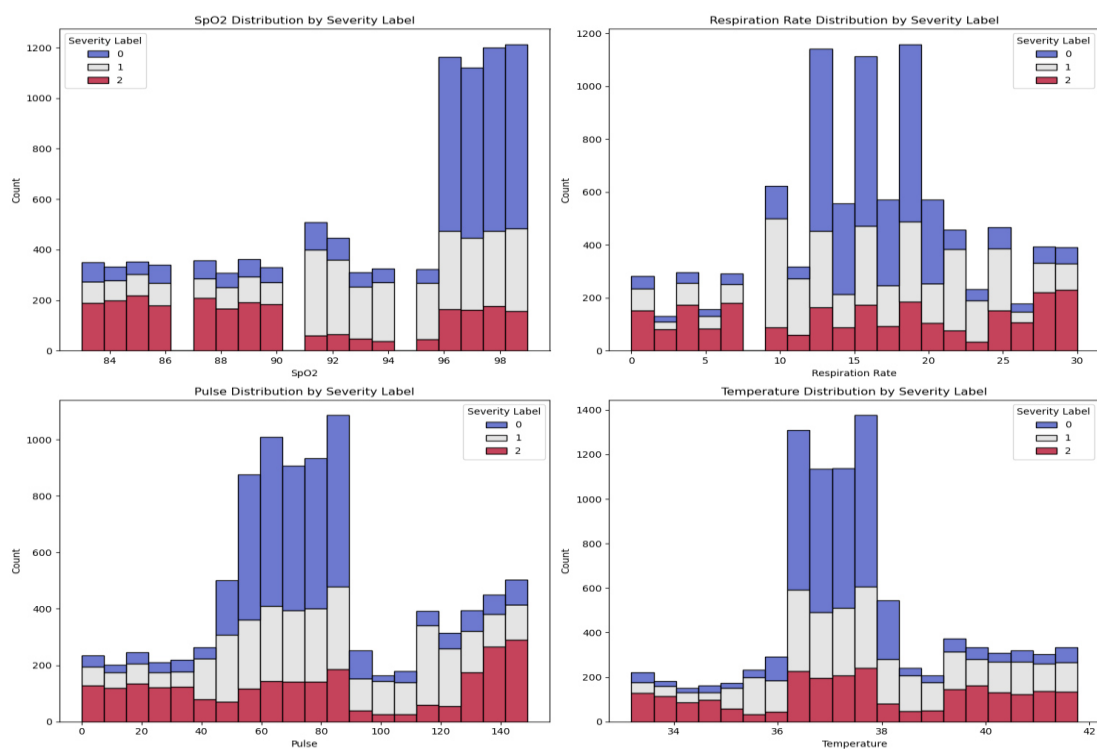


Figura 67. Base de datos utilizando los parámetros de NEWS y Distribución de datos de cada variable

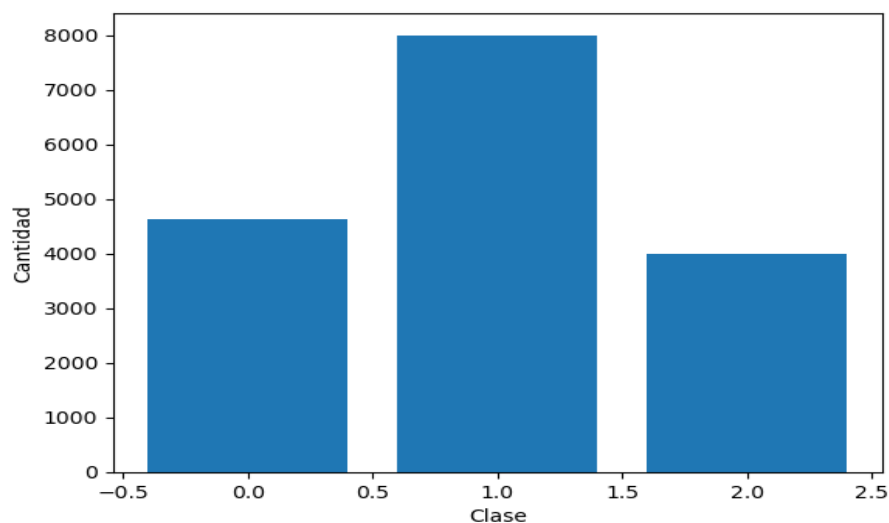


Figura 68. Número de datos de clase de la Base de datos integrada

b) Algoritmos de clasificación. Actualmente se dispone de una variedad de algoritmos de clasificación para señales de signos vitales y en base la revisión sistemática de la literatura se determinó que los más relevantes son aquellos basados en redes neuronales (MLP-perceptrón multicapa y DNN-redes neuronales profundas), máquina de vector soporte (SVM), Bosques aleatorios (RF) y árboles decisión (DT). En el caso de estos modelos, su evaluación consiste en seleccionar y preprocesar adecuadamente los datos, asegurando su calidad y representatividad. La elección del mejor modelo puede optimizarse mediante métodos como GridSearchCV (para ajustar sus hiperparámetros), validación cruzada K-Fold (minimizar el riesgo de sobreajuste) y métricas basadas en la matriz de confusión.

c) Proceso de clasificación. Para la implementación del proceso de clasificación es necesario realizar la extracción de características la cual se puede realizar en base a la modelos concatenados, que funcionan en paralelo. Otro método usa procesos basados en modelos híbridos multimodales, donde los datos pasan por una etapa de preparación y preprocesamiento y luego se ingresan a modelos separados, concatenando las salidas (características). Finalmente, también se puede extraer las características en base a un tratamiento de señales, uso de métodos estadísticas y un preprocesamiento diferenciado con un solo modelo de salida (Figura 69).

Por lo tanto, debido a las características propias del hardware, se usa el preprocesamiento diferenciado por tener una complejidad baja a media, su simplicidad

(requiere menor potencia computacional y fácil interpretación de las características combinadas). En el caso de la señal de ECG, se utilizan los filtros y métodos descritos en la sección 4, para extraer los patrones característicos y determinar la frecuencia respiratoria en el dispositivo embebido. Por otro lado, para las otras señales de SPO2, temperatura y frecuencia respiratoria, se utilizan los métodos clásicos para preparación, balanceo, detección de outliers y nulos.

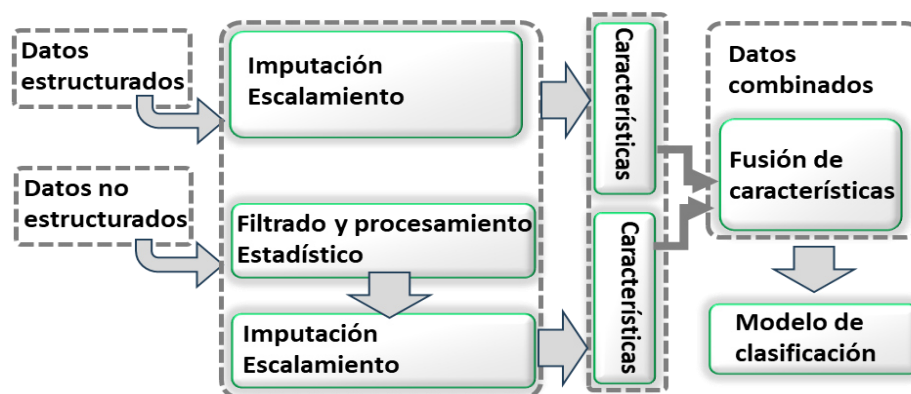


Figura 69. Proceso de clasificación

El proceso de clasificación se realizó utilizando las características sometidas al tratamiento de datos de la sección anterior, considerando la eliminación de valores atípicos mediante el método del rango intercuartílico. Este enfoque permitió que los resultados no se afecten por irregularidades en los datos de entrada.

d) Dispositivo embebido integrado. En base a la literatura revisada sobre la implementación de inteligencia artificial en dispositivos embebidos se identifica aquellos que tienen el equilibrio entre potencia computacional y recursos de hardware para procesar las señales fisiológicas de cardiopatía. Por lo tanto, tras revisar las opciones disponibles descritas en el estado de arte, se opta por integrar en el microcontrolador ESP32 (versión DOIT ESP32 DevKitV1) los modelos. Este dispositivo se describe en el capítulo 4.

5.2.2 Modelo propuesto

Como parte de la técnica computacional para la clasificación de señales fisiológicas relacionadas a cardiopatía se propone un proceso de integración donde se integran varias etapas de preprocesamiento para adaptar tanto datos estructurados como no estructurados durante el entrenamiento y las pruebas. Este flujo de trabajo ha sido desarrollado tomando como base la literatura revisada, que abarca desde la creación y

la implementación del modelo, su despliegue en el dispositivo embebido (nodo sensor integrado) y el análisis de los resultados. Este enfoque ha sido elaborado a partir de investigaciones previas en el ámbito de sistemas embebidos para aplicaciones de computación de borde y TinyML, haciendo uso de metodologías reconocidas como KDD y CRISP-DM (Situnayake, 2020) (Lea & Safari, 2020)(Iodice, 2022) (Purbasari et al., 2021) (Figura 70).

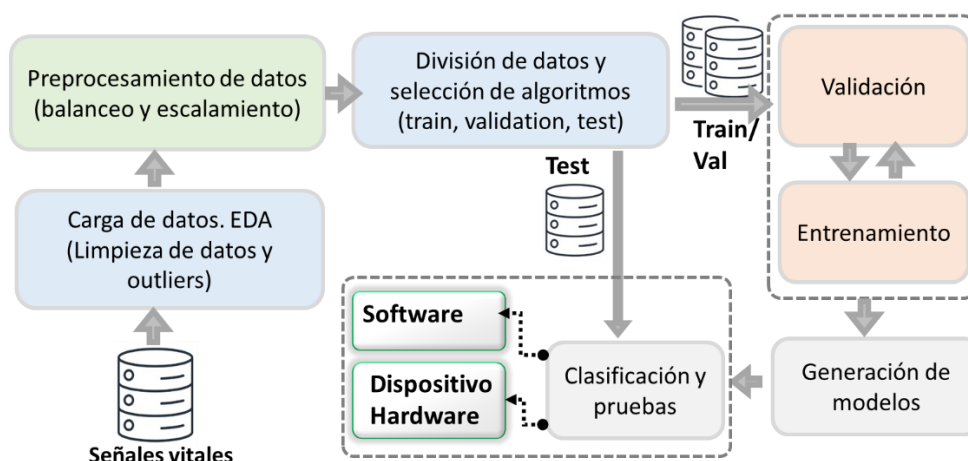


Figura 70. Proceso de generación de modelos propuesto

Las etapas de entrenamiento y pruebas consideran los siguientes componentes que involucra el uso de la Base de datos de signos vitales:

- **Carga y exploración del conjunto de datos.** El proceso inicia con la importación de los datos proveniente de la integración de un conjunto de datos previos en un archivo CSV. Se realiza el análisis exploratorio de datos (EDA). Además, se realiza la limpieza de datos, mediante la eliminación y tratamiento de Outliers.
- **Preprocesamiento de datos.** Se evalúa si se necesita convertir los datos en variables numéricas u otras transformaciones. Si las clases están desbalanceadas, se aplican métodos de sobre muestreo o submuestreo y si es necesario se escalan las características.
- **División de datos y selección de los algoritmos.** Se separan los datos en conjuntos de entrenamiento, validación y prueba, permitiendo que los algoritmos se entrenen con una parte de los datos.
- **Entrenamiento.** Se realiza el entrenamiento de los algoritmos para obtener los modelos, configurando sus hiperparámetros de manera iterativa.

- **Validación.** Se utiliza el conjunto de datos de validación para evaluar el rendimiento de los modelos durante el entrenamiento aplicando métodos de validación cruzada K-Fold.
- **Clasificación y pruebas.** Se evalúan los modelos con el conjunto de datos de pruebas utilizando la plataforma de software de entrenamiento para obtener métricas finales de rendimiento. Además, se implementan los modelos en el dispositivo embebido, evaluando el tiempo de procesamiento, consumo de corriente eléctrica y uso de memoria FLASH y RAM.

5.2.3 Desarrollo del modelo

Para el desarrollo de los procesos de clasificación de señales se realizó el desarrollo de scripts de programación en Python y en lenguaje C++ para el dispositivo embebido. En el caso del entorno entrenamiento y validación se usan librerías como tensorflowLite, tensorflow y Scipy. Mientras que en el caso del despliegue de modelos y programación en el dispositivo embebido se usa micromlgen (conversión del modelo al lenguaje C), librerías de lectura de sensores y de transmisión inalámbrica. A continuación, se describe los mecanismos y métodos usados en cada etapa.

a) Carga y exploración del conjunto de datos. Se utilizan la base de datos de CVD-Vital-Signs y la base de datos que se genera a partir de la categorización del estado de salud de NEWS2, las cuales son integradas para generar nuevo conjunto de datos (Figura 71). Luego, se realiza un análisis exploratorio de datos, obteniendo que no hay variables que sean dependientes entre ellas. Este conjunto de datos resultante tiene 16637 registros con 4 características y 3 niveles de severidad teniendo 4637 registros para el nivel de severidad 0 (salud normal), 8000 para la severidad 1 (leve) y 4000 para la severidad 2 (salud crítica) (Figura 72).

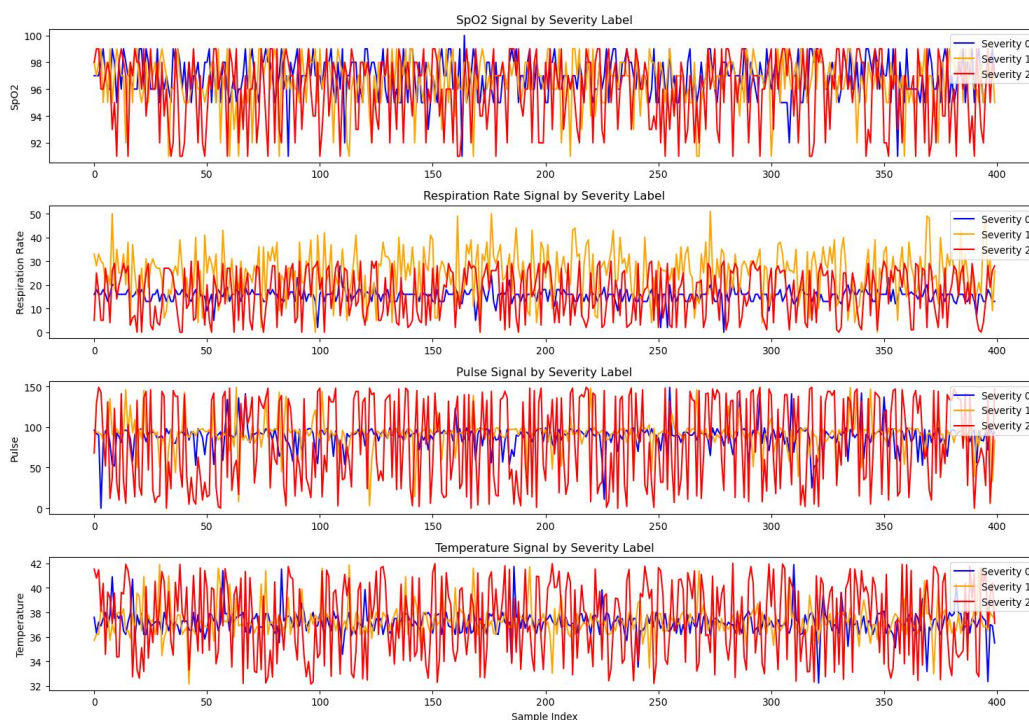


Figura 71. Muestra de la base de datos integrada

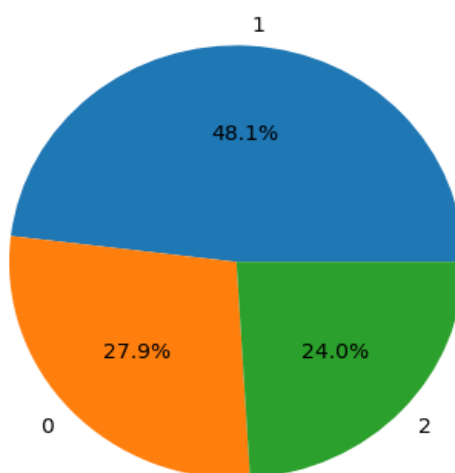


Figura 72. Proporción de cada clase

b) Preprocesamiento de datos. Los datos son adaptados para su uso en la etapa de entrenamiento y validación de modelos de aprendizaje automático realizando la limpieza de datos, mientras que para las clases desbalanceadas se utiliza el método del submuestreo para la más frecuente. Además, se utiliza scikit-learn para escalar las

características de manera uniforme (Figura 73). Como resultado se tienen el conjunto de datos con y sin escalamiento.

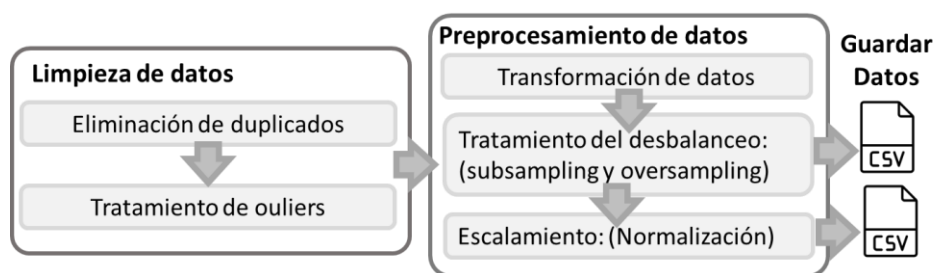


Figura 73. Preprocesamiento de datos

c) División de datos y selección de los algoritmos. En esta etapa, los datos se dividen en conjuntos de entrenamiento, validación y prueba, con una proporción de 64%, 16% y 20%, respectivamente usando la librería de Scikit-learn. Para esto se importan el conjunto de datos no escalados anteriores y son utilizados con los algoritmos seleccionados de clasificación según la literatura revisada, incluyendo SVM, bosques aleatorios, árboles de decisión y redes neuronales, utilizando bibliotecas como scikit-learn y TensorFlow. Los datos escalados son particionados y se guardan para el proceso de entrenamiento, validación y pruebas (Figura 74).

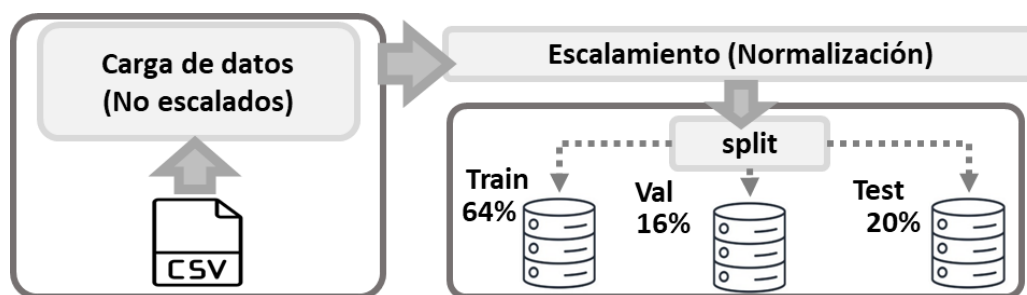


Figura 74. División de datos

d) Entrenamiento. El entrenamiento de los algoritmos se realiza ajustando los parámetros de cada algoritmo, optimizando los hiperparámetros mediante métodos como GridSearchCV y RandomizedSearchCV, proporcionadas por scikit-learn y otras creadas de manera iterativa. Durante este proceso, se busca maximizar el rendimiento de los modelos generados mediante la optimización de parámetros clave en algoritmos como árboles de decisión, configuración de parámetros de la profundidad de los árboles en Random Forest, los parámetros de regularización en SVM o número de

neuronas y capas en redes neuronales. Los procedimientos generales realizados en esta etapa son:

- Se cargan los datos ya procesados en la etapa anterior para entrenamiento, validación y pruebas. En este caso los datos no escalados fueron cargados y preprocesados antes de su ingreso al proceso de entrenamiento.
- Se realiza la optimización de hiperparámetros usando métodos como GridSearchCV (explora las combinaciones posibles de los hiperparámetros en un rango determinado) o RandomizedSearchCV (búsqueda aleatoria sobre un conjunto de combinaciones).
- De manera interna se utilizan métodos de validación cruzada para evaluar el rendimiento de los modelos a los que sea aplicables. Al final, se promedian los resultados de todas las validaciones para obtener una métrica de rendimiento.
- Las métricas como exactitud y F1-Score se calculan sobre el conjunto de validación de cada iteración. Estas métricas se promedian a lo largo de varias iteraciones para obtener una estimación del rendimiento del modelo.
- Posteriormente se prueba el modelo con los datos de validación para comprobar su funcionamiento y volver a iniciar si fuera el caso.
- Finalmente se evalúa el modelo con los datos de prueba (test) y se guarda el modelo entrenado, sus hiperparámetros y las métricas (Figura 75).

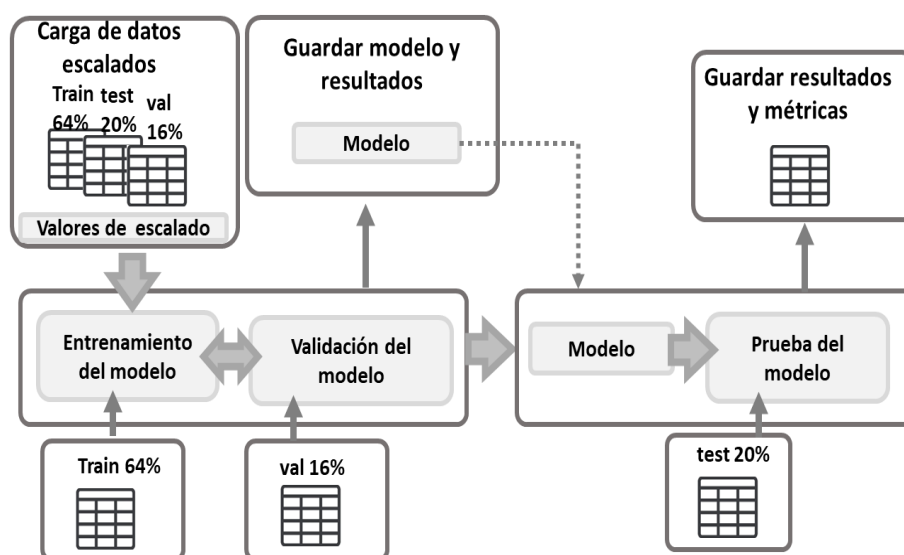


Figura 75. Proceso de entrenamiento

5.2.4 Validación de los modelos

En esta sección se presentan los resultados obtenidos durante la etapa de validación de los diferentes modelos desarrollados para la clasificación de señales fisiológicas. Para ello se utilizan diferentes configuraciones de hiperparámetros para cada modelo basado en SVM, RF y DT. Entre algunas métricas utilizadas para esta evaluación incluyen exactitud y F1-score.

Por otro lado, en el caso de modelos basados en redes neuronales, el uso de MLP (Perceptrón multicapa) o DNN (redes neuronales profundas) son más adecuados para estos tipos de datos tabulares, debido a que no presentan una estructura espacial ni secuencial que justifique el uso de CNN (redes neuronales convoluciones) o RNN (redes neuronales recurrentes) es por ello su elección para estas pruebas.

Se generan diez diferentes conjuntos de datos para el entrenamiento, validación y pruebas a partir del conjunto de datos originales los cuales, mediante un proceso de validación cruzada, son usados para obtener los modelos y las métricas de validación. A continuación, se describen los resultados de validación para cada tipo de algoritmo

a) SVM. El mejor valor de exactitud obtenido fue del 91.66%. No obstante 4 de las pruebas tuvieron un valor por debajo del 91% mientras que el resultado con menor valor de exactitud fue de 90.5%. En la siguiente tabla se muestran los 3 mejores resultados relacionados a la configuración del algoritmo (obtenido durante la búsqueda de hiperparámetros con 5 diferentes combinaciones mediante validación cruzada), exactitud y otras métricas (Tabla 32 y Tabla 33).

Tabla 32. Matriz de confusión con datos de validación para el modelo SVM

Actual \ Pred.	Clase 0	Clase 1	Clase 2
'C': 100, 'gamma': 5.0, 'kernel': 'rbf'			
Clase 0	700	29	0
Clase 1	127	1142	52
Clase 2	1	13	598
'C': 100, 'gamma': 5.0, 'kernel': 'rbf'			
Clase 0	704	36	0
Clase 1	150	1098	51
Clase 2	0	16	607
'C': 100, 'gamma': 5.0, 'kernel': 'rbf'			
Clase 0	698	42	0
Clase 1	127	1086	41
Clase 2	0	17	651

Tabla 33. Métricas de validación con el modelo SVM

Dataset	Config	Exactitud	F1-Score
DD4	'C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9166	0.9166
DD6	'C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9151	0.9149
DD7	'C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9147	0.9147

b) Árboles de decisión. Se evaluaron tres hiperparámetros para el modelo de árboles de decisión y se muestran las métricas de validación para cada configuración. El mejor valor de exactitud obtenido fue del 93.8%. Por otro lado, 4 de las pruebas tuvieron un valor por debajo del 93.5% mientras que el resultado con menor valor de exactitud fue 92.7%. En la Tabla 34 y Tabla 35 se muestran los 3 mejores resultados para este algoritmo

Tabla 34. Matriz de confusión con datos de validación para el modelo DT

Actual \ Pred.	Clase 0	Clase 1	Clase 2
Configuración: 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2			
Clase 0	709	31	0
Clase 1	84	1144	26
Clase 2	0	24	644
Configuración: 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2			
Clase 0	728	28	0
Clase 1	93	1145	32
Clase 2	0	12	624
Configuración: 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2			
Clase 0	729	25	0
Clase 1	88	1156	27
Clase 2	0	27	610

Tabla 35. Métricas de validación con el modelo DT

Dataset	Config	Exactitud	F1-Score
DD7	'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.938	0.9381
DD9	'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.938	0.938
DD10	'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.9373	0.9373

c) Bosques aleatorios. En este caso el mejor valor de exactitud obtenido fue del 94.8% considerando la matriz de confusión de la Tabla 36. Además, 6 de las pruebas con diferentes conjuntos de datos, tuvieron una exactitud mayor al 94.5% mientras que el

resultado con menor valor de exactitud fue 93.4%. En la Tabla 37 se muestran los 3 mejores resultados relacionados a la configuración del modelo (obtenido usando 5 diferentes combinaciones), exactitud y otras métricas (obtenida con el conjunto de datos de validación para cada caso).

Tabla 36. Matriz de confusión con datos de validación para el modelo RF

Actual \ Pred.	Clase 0	Clase 1	Clase 2
Configuración: 'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 30			
Clase 0	728	23	0
Clase 1	84	1177	31
Clase 2	0	10	609
Configuración: 'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30			
Clase 0	715	26	0
Clase 1	88	1197	31
Clase 2	0	10	595
Configuración: 'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 30			
Clase 0	748	24	0
Clase 1	75	1158	37
Clase 2	0	7	613

Tabla 37. Métricas de validación con el modelo RF

Dataset	Config	Exactitud	F1-Score
DD5	'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 30	0.9482	0.9482
DD2	'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30	0.9463	0.9462
DD10	'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 30	0.9455	0.9455

d) MLP. Se hicieron combinaciones considerando cambios en 3 hiperparámetros para la creación de los modelos para lo cual se tienen los parámetros de: 4 configuraciones de capas ocultas ([16] , [32], [32,16], [64,32]), tasa de aprendizaje (learning_rate: 0.001 y 0.01) y batch_size (16 y 32). En base a estas configuraciones se obtuvieron los resultados de los 3 mejores modelos en la Tabla 38 y Tabla 39.

Tabla 38. Matriz de confusión con datos de validación para el modelo de MLP

Actual \ Pred.	Clase 0	Clase 1	Clase 2
Configuración: 'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20			
Clase 0	738	34	0

Clase 1	98	1117	55
Clase 2	0	14	606
Configuración: 'layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20			
Clase 0	692	59	0
Clase 1	110	1123	59
Clase 2	0	4	615
Configuración: 'layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 32, 'epochs': 20			
Clase 0	723	31	0
Clase 1	124	1089	58
Clase 2	0	10	627

Tabla 39. Métricas de validación con el modelo MLP

Dataset	Config	Exactitud	F1-Score
DD3	'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.924493	0.92426
DD5	'layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.912847	0.9125
DD10	'layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 32, 'epochs': 20	0.916228	0.91592

e) **DNN**. Las redes neuronales profundas fueron entrenadas con diferentes arquitecturas de 3, 4 y 5 capas, mostrando los 3 mejores resultados considerando sus resultados en la etapa de validación tanto en la Tabla 40 como en la

Tabla 41.

Tabla 40. Matriz de confusión con datos de validación para modelos basados en DNN

Actual \ Pred.	Clase 0	Clase 1	Clase 2
Configuración: 'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20			
Clase 0	734	43	0
Clase 1	113	1068	49
Clase 2	0	9	646
Configuración: 'layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20			
Clase 0	693	48	0
Clase 1	126	1148	42
Clase 2	0	20	585
Configuración: 'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20			
Clase 0	705	24	0
Clase 1	127	1134	60
Clase 2	0	2	610

Tabla 41. Métricas de validación con el modelo de DNN

Dataset	Config	Exactitud	F1-Score
DD1	'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20	0.919609	0.91931
DD2	'layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.911345	0.91143
DD4	'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20	0.919985	0.91978

f) Resumen de evaluaciones de los modelos. Considerando los resultados anteriores obtenidos en el entorno de desarrollo de software, se evalúan las métricas y estas muestran un mejor rendimiento con el conjunto de entrenamiento en comparación con el conjunto de validación. De esta manera se detecta el sobreajuste en cuyo caso se simplificaba el modelo o se detiene el entrenamiento.

Entre los modelos basados en aprendizaje automático, el mejor rendimiento lo tiene el modelo basado en bosques aleatorios con una exactitud de 0.9482, y un F1-score de 0.948 seguido del modelo basado en árboles de decisión (0.938). Por otro lado, el modelo SVM, fue el que mostró el rendimiento más bajo con exactitud del 0.9166. En el caso de los modelos basados en redes neuronales, el modelo MLP está en tercer lugar de exactitud con 0.9245, destacando su configuración usando sólo 2 capas siendo superior al modelo DNN (Tabla 42).

Tabla 42. Resumen de métricas con datos de validación en el entorno de desarrollo

Modelo	Dataset	Configuración	Exactitud	F1-Score
SVM	DD4	'C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9166	0.9166
Árboles de Decisión	DD7	'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.9380	0.9381
Bosques aleatorios	DD5	'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 30	0.9482	0.9482
MLP	DD3	'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.9245	0.9243
DNN	DD4	'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20	0.920	0.9198

CAPÍTULO 6: RESULTADOS Y DISCUSIÓN

Cada uno de los modelos generados en la etapa anterior se evalúan utilizando los datos de pruebas (test) en el entorno de desarrollo y en el dispositivo embebido que implementa el nodo sensor. Los resultados se interpretan considerando el comportamiento computacional y no son interpretados como resultados de validación clínica.

6.1 Entorno de desarrollo

Luego de realizar el entrenamiento y validación, se procede con la evaluación del funcionamiento de los modelos usando el conjunto de datos de prueba, bajo condiciones similares de despliegue que en el dispositivo embebido. Los resultados obtenidos en el entorno de desarrollo muestran un rendimiento consistente entre los diferentes modelos evaluados. Según el resumen de métricas, se observa en la Tabla 43 los mejores modelos para cada tipo de algoritmo, donde el modelo de bosques aleatorios presenta la mejor exactitud global (0.9504), seguido del modelo árboles de decisión (0.9450) y el MLP (0.9303). A continuación, se describen los resultados obtenidos en esta etapa:

Tabla 43. Resumen de métricas con datos de prueba en el entorno de desarrollo

Modelo	Dataset	Configuración	Exactitud	F1-Score
SVM	DD3	'C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9213	0.9211
Árboles de Decisión	DD8	'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.9450	0.9450
Bosques aleatorios	DD3	'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30	0.9504	0.9504
MLP	DD3	'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.9303	0.9301
DNN	DD8	'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20	0.9231	0.9231

- **Comportamiento de los Modelos.** La evolución de las métricas se evalúa considerando los datos obtenidos en el capítulo anterior relacionados con las fases de entrenamiento, validación y considerando los datos de prueba obtenido en este capítulo, para determinar la convergencia o el sobreajuste. Para generar el gráfico de la evolución del comportamiento se utiliza los modelos con mayor exactitud visualizando que, tanto la exactitud en entrenamiento como en validación aumentaron de manera constante, indicando convergencia. Sin

embargo, mientras más datos de entrenamiento se utilizan, la exactitud de validación se acerca a la de entrenamiento, que podrían indicar señales de sobreajuste que no ocurren durante este proceso. Esto confirma que el modelo generaliza de manera aceptable (Figura 76 y Figura 77).

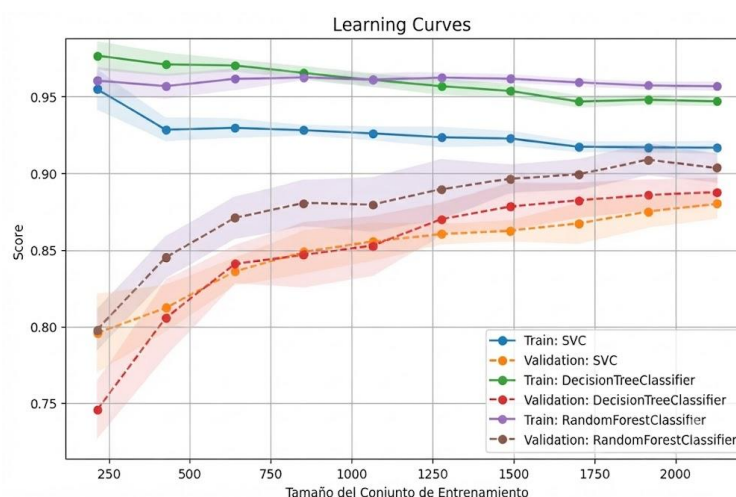


Figura 76. Curvas de entrenamiento y validación de los algoritmos de SVM, árboles de decisión y bosques aleatorios

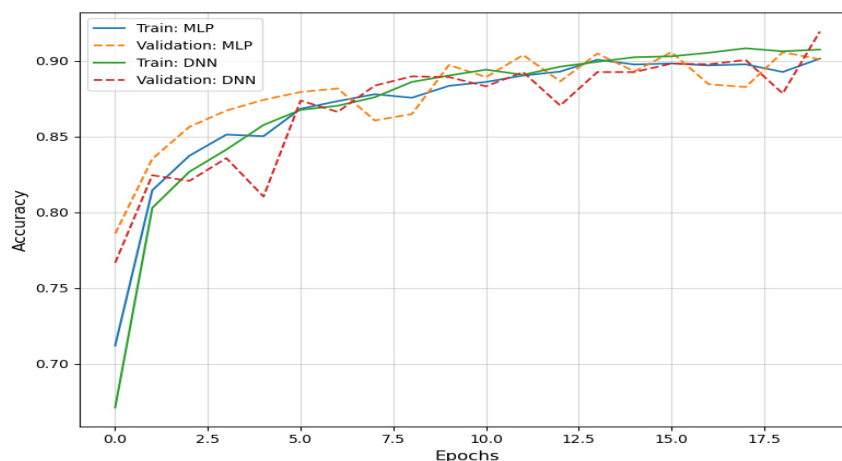
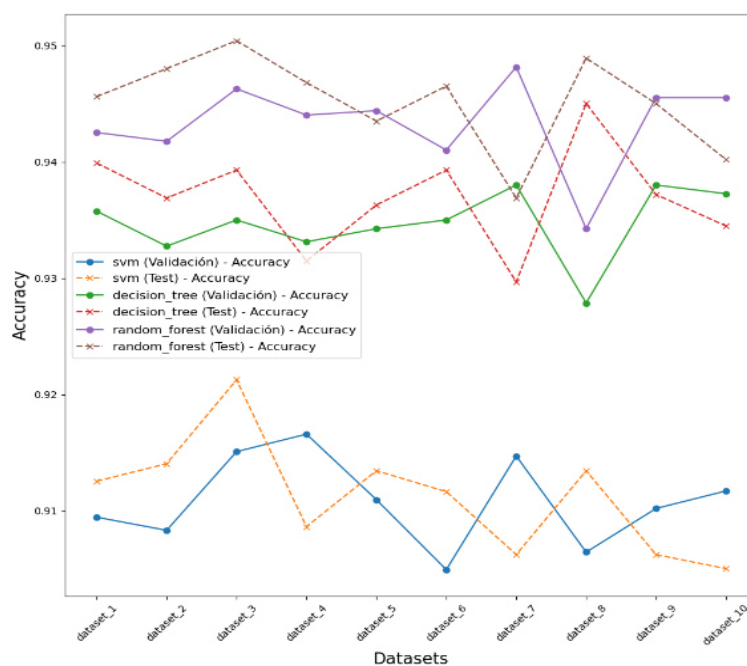


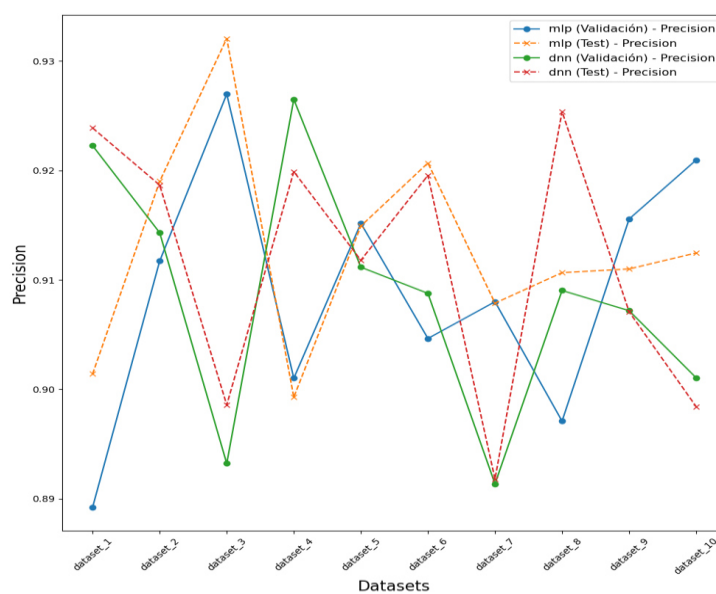
Figura 77. Curvas de entrenamiento y validación de los modelos basados en MLP y DNN

- **Comportamiento con diferentes secciones del conjunto de datos.** En esta sección se describen los resultados obtenidos cuando se utilizan una sección de datos seleccionados de cada conjunto de datos, los cuales también se utilizan para la evaluación en el dispositivo embebido. En la Figura 78 (a) se evidencia que el modelo basado en RF es el modelo más consistente y preciso, logrando

un equilibrio entre validación y prueba en diferentes con diferente conjunto de datos, donde, además el DT presenta un comportamiento aceptable, pero con variabilidad mayor. En cambio, los modelos MLP y DNN (Figura 78 (b)) tienen una mayor variabilidad en la exactitud lo que indica que son más sensibles a la distribución de los datos por lo que el RF demostró ser el modelo con mejor desempeño global.



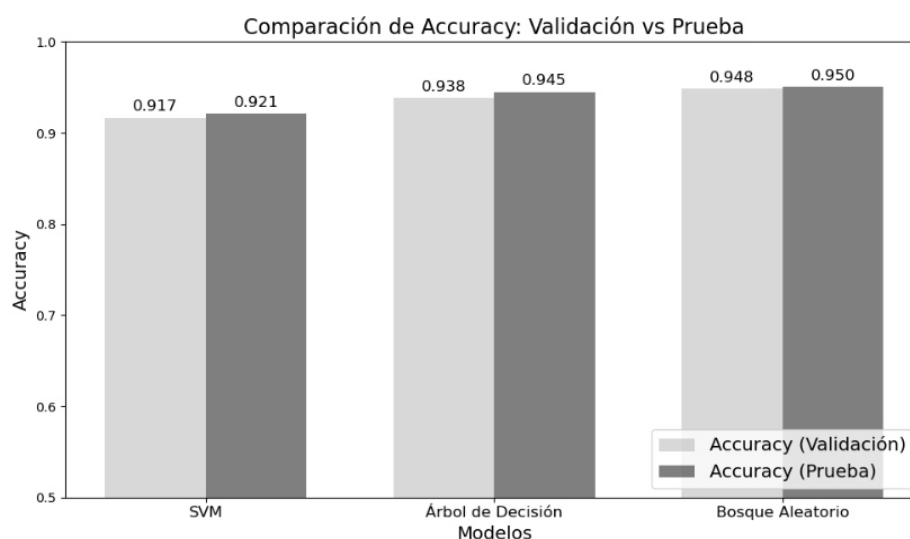
(a) Exactitud de los modelos SVM, DT y RF



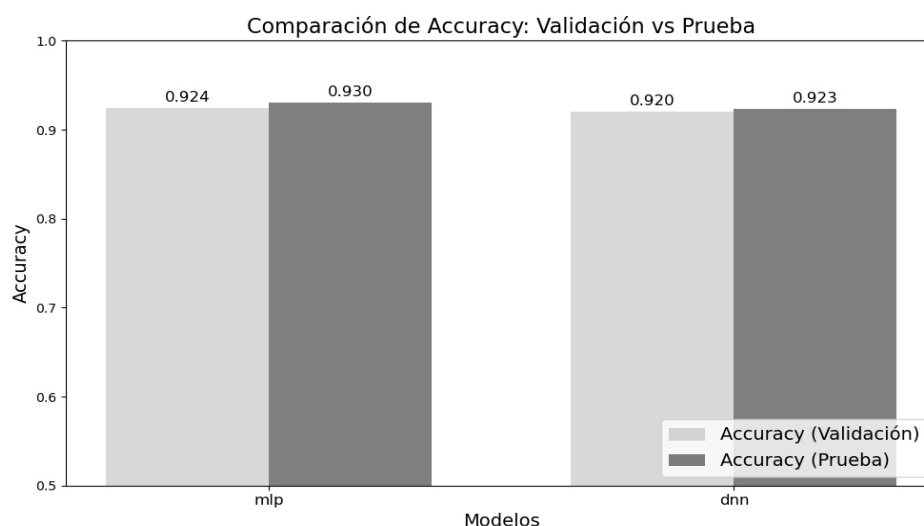
(b) Exactitud de los modelos MLP y DNN

Figura 78. Comparación de resultados de exactitud para los modelos en el entorno de software

En la Figura 79 se visualizan las mejores métricas obtenidas al evaluar los modelos donde se comprueba un rendimiento consistente a lo largo de las fases de validación y pruebas. Se evidencia que las diferencias entre estos resultados son reducidas, por lo que se puede inferir que los modelos generalizan adecuadamente y que su desempeño no se deteriora por lo que no hay evidencia de sobreentrenamiento.



(a) Resultados de validación y prueba de los modelos SVM, DT y RF



(b) Resultados de validación y prueba de los modelos MLP y DNN

Figura 79. Comparación de resultados de exactitud en la etapa de validación y prueba de los modelos

6.2 Resultados en el dispositivo embebido

Los modelos evaluados en el dispositivo de hardware embebido permitieron evaluar las métricas relacionadas al procesamiento, memoria y consumo de corriente. La evaluación se centra en verificar si el modelo mantiene un rendimiento eficiente considerando las restricciones de hardware utilizando una sección de los datos de prueba.

Para esto se importan los modelos y los datos escalados (junto con los valores usados para el escalamiento) y luego se convierten los modelos a formatos compatibles con el hardware (PHW1). Se exportan los modelos utilizando las librerías de software de micromlgen y tensorflowlite que traducen los modelos de Python a C++ y luego son importados desde el dispositivo embebido (nodo sensor integrado) para obtener las métricas del comportamiento (PHW2). Luego se considera una etapa de conversión de datos no escalados (Proceso NE/ES) que se ingresan a los modelos implementados en el hardware (Figura 80).

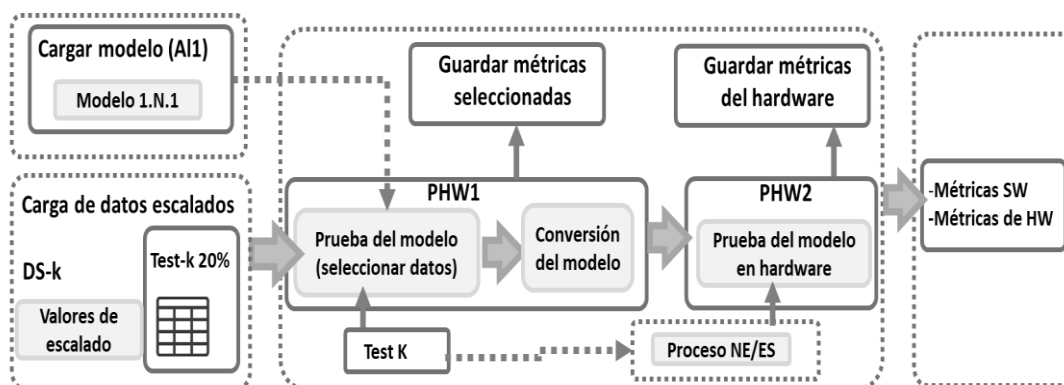


Figura 80. Procesos para determinar el comportamiento de modelos en el dispositivo embebido

Comparando los modelos evaluados (Tabla 44), se observa que, en términos de exactitud, los RF y DT se destacaron con un 96.6% y 95.6% respectivamente, mientras que el valor de F1-score de los RF fue el más alto, alcanzando 97%, lo que indica un equilibrio entre exactitud y sensibilidad. Por otro lado, los modelos basados en redes neuronales lograron buenos resultados con exactitudes de 94.6 % y 94.3%, pero su F1-score fue ligeramente inferior, lo que sugiere un desempeño menos consistente en la clasificación. En general, los modelos RF resaltan como el más robusto en términos de rendimiento general en el entorno de software seguido de los DT.

Tabla 44. Promedio de métricas obtenidas con datos de pruebas seleccionados en el entorno de software

Modelo	Configuración	Exactitud	F1-Score
SVM	C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.9466	0.9532
Árboles de Decisión (DT)	max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.9566	0.9598
Bosques aleatorios (RF)	max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30	0.9666	0.9707
MLP	layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.94666	0.9524
DNN	layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 32, 'epochs': 20	0.94333	0.9500

En la Tabla 45 se observan diferencias en el comportamiento de los modelos cuando se ejecutan en el hardware del dispositivo embebido, donde nuevamente el RF, muestra el mejor rendimiento, con una exactitud alta de 96.6% seguido del modelo basado en MLP. Por otro lado, los modelos más livianos, como los DT tienen una exactitud ligeramente inferior (95.6). El SVM, por su parte, mostró el menor impacto en el hardware, con una exactitud de 91%.

Tabla 45. Promedio de las métricas obtenidas en el dispositivo embebido con los datos de pruebas seleccionados

Modelo	Configuración	Exactitud
SVM	C': 100, 'gamma': 5.0, 'kernel': 'rbf'	0.910
Árboles de Decisión	max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2	0.956
Bosques aleatorios	max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 30	0.96601
MLP	layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.957
DNN	layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20	0.942

Durante la comparación de las métricas de exactitud con datos seleccionados se observa que en el entorno de desarrollo, el RF (96.6%), mantiene un valor similar al obtenido en el dispositivo de embebido. Además, los DT mantienen también su valor de 95.66% tanto en hardware como en el entorno de desarrollo lo que indica una mayor adaptabilidad a las limitaciones del dispositivo y presentaron el menor cambio entre los dos entornos. Esto lo hace una opción robusta para implementaciones en dispositivos con recursos limitados (a diferencia de los bosques aleatorios con mayor

complejidad computacional). Por otro lado, con menores valores se tiene los modelos basados en MLP, DNN y por último las SVM (Figura 81).

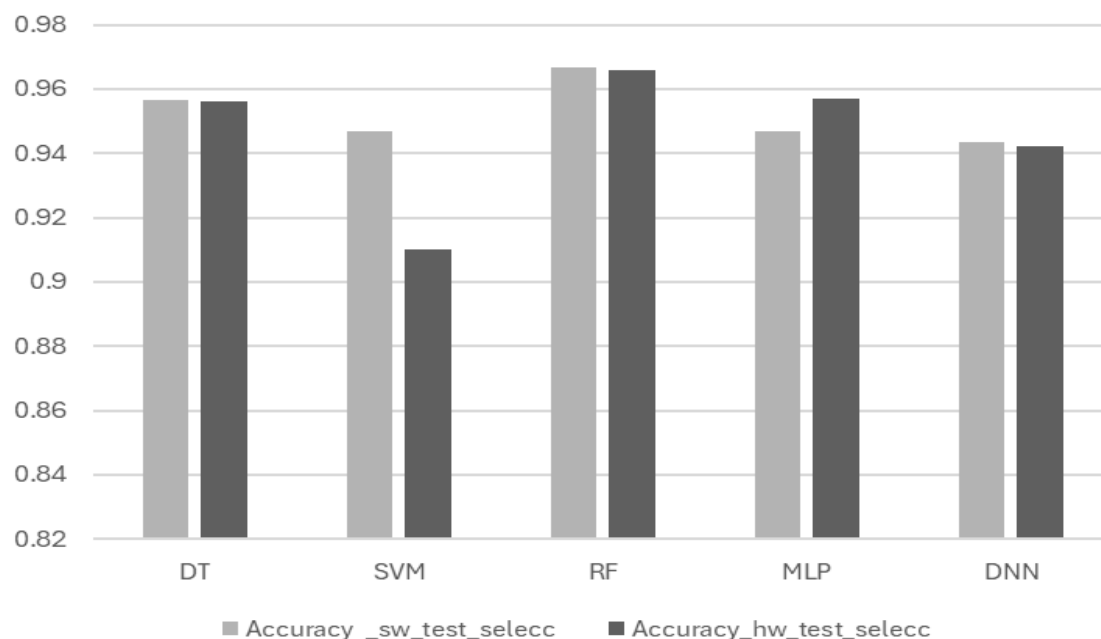


Figura 81. Comparación de resultados en el entorno de desarrollo y en el hardware.

6.3 Características de funcionamiento del dispositivo embebido

En esta sección se muestra el comportamiento del dispositivo embebido relacionadas con el consumo de corriente eléctrica, tiempo usado para realizar la predicción (T.P.), tiempo de normalización (T.N.), el tamaño de los modelos en memoria, FLASH y RAM considerando la evaluación en el contexto computacional y no clínico. Para ello se considera que los datos no escalados fueron cargados dentro del firmware y se tiene una etapa de conversión a datos escalados, con la finalidad de determinar la latencia en la conversión como parte de las métricas de evaluación (Figura 80, proceso NE/ES).

Aunque anteriormente se observa que los modelos son diferentes a los descritos en la Tabla 29 (los cuales no son modelos optimizados) y mantuvieron una exactitud aceptable (alrededor del 90%) en el dispositivo embebido, hay una notable diferencia en el caso del tiempo de inferencia y el consumo de corriente eléctrica, lo que refleja las limitaciones de procesamiento del dispositivo embebido para ciertos tipos de modelos donde algunos utilizan más recursos de hardware que otros.

El modelo basado en árbol de decisión ofrece un equilibrio razonable entre exactitud y eficiencia computacional al consumir la menor cantidad de corriente eléctrica (39.3 mA) y tiempo de inferencia (8.665 us), lo que los convierte en la opción más viable para dispositivos con recursos limitados (Tabla 46). Por otro lado, en el caso de los bosques aleatorios, a pesar de que su comportamiento en el entorno de software fue adecuado con relación a la exactitud, este consume más recursos de memoria y tiempo de inferencia que todos los modelos. Finalmente, sobre el consumo de corriente eléctrica, los modelos basados en redes neuronales son los que tienen los valores más altos (Figura 82).

Tabla 46. Promedio de características de funcionamiento en el dispositivo embebido

Modelo	Consumo de corriente (mA)	T. N. (us)	T.P (us)	RAM usada (%)	FLASH usada (%)
SVM	40.3	4.8	73.02	82.22	46.1523
Árboles de Decisión (DT)	39.3	4.818	8.665	6.9	6.771
Bosques aleatorios (RF)	41.46	4.8212	720.2	100	100
MLP	59.6	4.677	123.2358	6.838	9.942
DNN	57.8	4.692	409.502	7.305	10.495

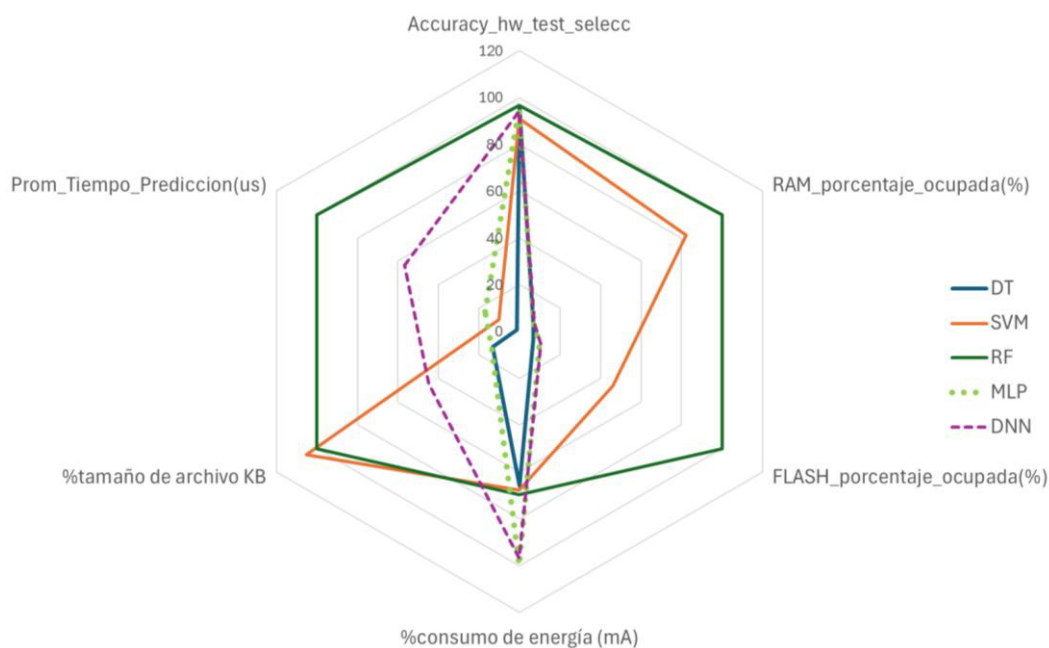


Figura 82. Gráfico comparativo de datos obtenidos durante la evaluación en el dispositivo embebido

6.4 Pruebas de hipótesis

Para realizar la comprobación de la hipótesis se utilizan los siguientes pasos para cada característica a evaluar acerca del comportamiento de los modelos de clasificación en el dispositivo de hardware embebido:

a) Selección de modelos: Se identificaron los modelos que se evaluaron en el dispositivo para la detección del estado de salud obteniendo métricas relacionadas a la exactitud, tiempo de inferencia, tamaño del archivo generado, consumo de memoria FLASH y RAM y consumo de corriente eléctrica.

b) Calculo de las métricas a evaluar: Para cada modelo se consideran múltiples grupos de muestras (10 muestras correspondientes a variaciones de modelos para cada uno de los 5 algoritmos)

c) Definición de las hipótesis: Se plantea una hipótesis nula (H_0) y una hipótesis alternativa (H_1) relacionadas con las características a comparar en los diferentes modelos:

- H_0 : No hay diferencia entre las características evaluadas de los diferentes modelos.
- H_1 : Existe una diferencia significativa entre las características evaluadas de los diferentes modelos.

d) Verificación de supuestos. Se verifican dos supuestos: normalidad de los datos y homogeneidad de varianzas (se cumple la hipótesis nula con un valor p mayor que 0.05). Para ello se realizan pruebas estadísticas de Shapiro-Wilk para verificar la normalidad y la prueba de Levene para verificar la homogeneidad de varianzas. Si el valor p es menor o igual a 0.05 en cualquiera de las pruebas entonces se tiene evidencia para rechazar la hipótesis nula, por lo tanto, los datos no siguen una distribución normal y las varianzas no son homogéneas. En este último caso se debe utilizar pruebas no paramétricas (como Kruskal-Wallis) y si fuera lo contrario la prueba de ANOVA.

e) Determinar diferencias entre los modelos: Según los resultados anteriores se selecciona el tipo de prueba paramétrica (por ejemplo, ANOVA) o no paramétrica (Kruskal-Wallis) para evaluar si existen diferencias entre los modelos. Al aplicar la prueba anterior, se compara el resultado con el nivel de significancia establecido obtenido (0.05). Si el valor p es menor que el nivel de significancia, se rechaza la

hipótesis nula (que indica que no hay diferencia entre características) y se acepta la hipótesis alternativa, donde existe una diferencia significativa en al menos uno de los modelos.

f) Pruebas de comparaciones múltiples: Si el resultado anterior indica que hay diferencias significativas se realizan pruebas de comparaciones múltiples. Para ello se pueden usar la prueba de Tukey y la prueba de Bonferroni, en el caso que se hayan usado pruebas paramétricas. Por otro lado, si se usan pruebas no paramétricas (debido a que no se cumplieron los supuestos de normalidad y homogeneidad) se puede usar pruebas post-hoc, como la de Dunn con corrección de Bonferroni, para identificar los pares de grupos que difieren.

g) Interpretación de resultados: Para detectar el modelo con el mejor comportamiento, según la característica analizada, se interpreta la salida generada por el software Python en caso se use la prueba de Tukey:

- Identificar las comparaciones significativas mediante la columna reject donde se identifican las comparaciones en las que se rechaza la hipótesis nula (True).
- Se revisan las diferencias promedio (meandiff). Esto se realiza entre las medias de los grupos comparados. Un valor positivo indica que el primer modelo tiene una exactitud promedio mayor que el segundo modelo.
- Finalmente se analiza el intervalo de confianza (columna lower y upper). Si el intervalo de confianza no incluye el cero, significa que la diferencia entre las medias de los grupos comparados es estadísticamente significativa.

Por otro lado, en el caso de la prueba de Dunn, los resultados mediante las herramientas estadísticas de Python (librería `scikit_posthocs`) obtenidos, muestran el esquema en forma de matriz para cada par de combinaciones de los modelos, con valores de p en cada celda. Su interpretación se realiza de la siguiente manera:

- Los resultados de la prueba de Dunn se muestran en formato matriz con valores de p. Además, se debe tener los valores promedio de la característica a analizar por cada modelo
- Ordenar los modelos en orden de desempeño con sus valores promedio dependiendo del criterio elegido (mayor o menor)

- Interpretación de la matriz de Dunn. En este caso se revisa las combinaciones identificado aquellos con diferencia significativa (valor p menor que 0.05). Luego se determina los modelos con diferencia estadísticamente significativa.
- Identificar el mejor modelo. Se selecciona el mejor modelo según el criterio de desempeño. Además, se complementa este análisis con comparaciones utilizando los valores p de la prueba de Dunn.

En base al procedimiento descrito anteriormente se aplican estos pasos para evaluar cada una de las características de comportamiento obtenidas para los modelos. A continuación, se describen los resultados para cada caso:

a) Tiempo de predicción. Para este caso se siguen los pasos previamente descritos:

- **Hipótesis nula y alternativa.** Como primer paso se determinan las hipótesis a evaluar:
 - Hipótesis nula (H0): No hay diferencia en los tiempos de predicción entre los diferentes modelos.
 - Hipótesis alternativa (H1): Existe al menos una diferencia significativa en los tiempos de predicción entre los diferentes modelos.
- **Datos por evaluar.** Los datos de tiempo de predicción se obtuvieron durante el funcionamiento de los cada uno de los modelos en el dispositivo embebido. Los datos obtenidos para las evaluaciones se muestran en la Tabla 47

Tabla 47. Tiempos de predicción obtenidos

	DNN	DT	MLP	RF	SVM
0	617.03	7.67	7.238	653.6	66.36
1	202.17	8.77	165.030	701.6	80.16
2	616.87	8.30	78.200	744.0	61.40
3	617.00	6.97	78.300	567.6	55.76
4	202.23	9.07	78.230	775.6	92.56
5	202.00	10.17	165.100	813.6	84.36
6	201.93	9.50	165.130	810.0	69.00
7	616.93	9.30	164.830	724.0	84.40
8	202.03	7.70	165.130	676.0	56.60
9	616.83	9.20	165.170	736.0	79.60

- **Verificación de supuestos.** En este caso, la hipótesis nula para ambas pruebas es:
 - Prueba de Shapiro-Wilk: Los datos siguen una distribución normal.

- Prueba de Levene: Las varianzas son iguales entre los grupos.

Observando los resultados de software al aplicar las pruebas (Tabla 48), las hipótesis nulas se rechazan y en este caso se debe utilizar pruebas no paramétricas debido a que se tiene valores de p menores a 0.05.

Tabla 48. Resultados de la verificación de supuestos

Modelo DNN: Prueba Shapiro-Wilk - Valor p: 0.00025679569807834923 Los datos no siguen una distribución normal. Modelo DT: Prueba Shapiro-Wilk - Valor p: 0.8186915516853333 Los datos siguen una distribución normal. Modelo MLP: Prueba Shapiro-Wilk - Valor p: 0.002305279253050685 Los datos no siguen una distribución normal. Modelo RF: Prueba Shapiro-Wilk - Valor p: 0.682458221912384 Los datos siguen una distribución normal. Modelo SVM: Prueba Shapiro-Wilk - Valor p: 0.4398563504219055 Los datos siguen una distribución normal. Prueba de Levene - Valor p: 1.0638143467730896e-17 Recomendación: Considerar pruebas no paramétricas (datos no normales).
--

- **Determinar diferencias entre los modelos.** Se aplica la prueba Kruskal-Wallis para analizar las diferencias entre los grupos considerando la Hipótesis nula y alternativa del tiempo de predicción. Los resultados al aplicar las funciones estadísticas en Python muestran que el valor p es menor que el nivel de significancia 0.05, por lo tanto, se rechaza la hipótesis nula y se concluye que existen diferencias significativas en las exactitudes entre los algoritmos (Tabla 49).

Tabla 49. Diferencias entre grupos para el tiempo de predicción

Prueba de Kruskal-Wallis: Estadístico = 43.4430272762197, Valor p = 8.373436819448492e-09 Existen diferencias significativas entre los grupos. Proceder con pruebas post-hoc.

- **Pruebas de comparaciones múltiples.** En base a las pruebas de comparaciones múltiples se obtuvieron los siguientes resultados:

- El análisis de los resultados de la prueba de Dunn indica que el modelo DT mostró el menor tiempo promedio de predicción (8.67 μ s), ubicándose como el modelo más eficiente en este aspecto, seguido por SVM (73.02 μ s) y MLP (123.24 μ s).
- Por otro lado, los modelos DNN y RF presentaron tiempos mayores, con 409.50 μ s y 720.20 μ s, respectivamente.
- Además, se encontró diferencias significativas entre ciertos pares de modelos en la matriz de resultados usando la prueba de Dunn (Figura 83). En este caso el modelo DT presentó diferencias con DNN ($p=0.0001$) y RF ($p=0.0000$), lo que indica que DT tiene tiempos de predicción significativamente menores. Sin embargo, no se observaron diferencias importantes entre DT y otros modelos como SVM ($p=0.7517$) y MLP ($p=0.1603$).
- Se observa que el modelo MLP tiene una diferencia importante respecto a RF ($p=0.0044$), con diferencias importantes respecto a SVM ($p=0.0003$).

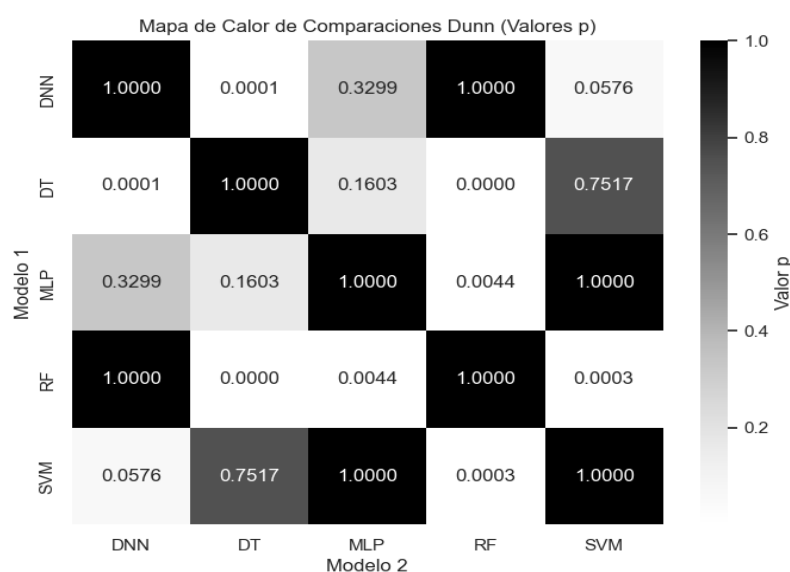


Figura 83. Matriz de resultados de las pruebas de comparaciones múltiples para tiempo de predicción

b) Exactitud de modelos. Se seleccionan los datos a evaluar para realizar el análisis de varianza y respecto a las exactitudes entre los modelos con pruebas de comparaciones múltiples. A continuación, se describen los pasos realizados:

- **Hipótesis nula y alternativa.** En primer lugar, se determinan las hipótesis:
 - Hipótesis nula (H_0): No hay diferencia en la exactitud entre los diferentes modelos.

- Hipótesis alternativa (H1): Existe al menos una diferencia significativa en la exactitud entre los diferentes modelos.
- **Verificación de supuestos.** Se determina la hipótesis nula para ambas pruebas:
 - Prueba de Shapiro-Wilk: Los datos de exactitud siguen una distribución normal.
 - Prueba de Levene: Las varianzas son iguales entre los grupos de modelos.

Observando los resultados (Tabla 50), se tiene que no todos los datos son normales, por lo tanto, las hipótesis nulas se rechazan y en este caso se debe utilizar pruebas no paramétricas.

Tabla 50. Verificación de supuestos con datos de exactitud

Modelo DNN:
Prueba Shapiro-Wilk - Valor p: 0.03418070450425148
Los datos no siguen una distribución normal.
Modelo DT:
Prueba Shapiro-Wilk - Valor p: 0.08221086114645004
Los datos siguen una distribución normal.
Modelo MLP:
Prueba Shapiro-Wilk - Valor p: 0.012787408195436
Los datos no siguen una distribución normal.
Modelo RF:
Prueba Shapiro-Wilk - Valor p: 0.05282573401927948
Los datos siguen una distribución normal.
Modelo SVM:
Prueba Shapiro-Wilk - Valor p: 0.13250669836997986
Los datos siguen una distribución normal.
 Prueba de Levene - Valor p: 0.07149706352765917
Recomendación: Considerar pruebas no paramétricas (datos no normales).

- **Determinar diferencias entre los modelos.** Una vez verificado los supuestos, se aplica la prueba Kruskal-Wallis para analizar las diferencias entre los grupos. Los resultados muestran que el valor p es mayor que 0.05, por lo tanto, no existen diferencias significativas en las exactitudes entre los algoritmos (Tabla 51).

Tabla 51. Determinación de diferencias entre grupos respecto a la exactitud

Prueba de Kruskal-Wallis: Estadístico = 7.001632268307161, Valor p = 0.13580199263262174 No se encontraron diferencias significativas entre los grupos.
--

Esto implica que no hay suficiente evidencia estadística para rechazar la hipótesis nula, por lo que los modelos tienen un desempeño comparable de exactitud, y su elección debe basarse en otros elementos como el tiempo de predicción o recursos de memoria. A pesar que el modelo Random Forest tiene la mayor exactitud promedio (0.96601), esto no permite establecer una superioridad estadística frente a los otros modelos (Figura 81).

El resultado de la prueba de Kruskal–Wallis ($p = 0.135$), indica que los algoritmos presentan un desempeño comparable en términos de exactitud de clasificación. La falta de diferencias estadísticamente significativas entre los modelos evaluados, permite inferir que el criterio de selección del modelo para dispositivos embebidos es dependiente de los otras métricas evaluadas. Dado que todos los algoritmos alcanzan valores similares de exactitud, la decisión de implementación se basa en métricas de eficiencia, tales como el consumo de corriente eléctrica, el uso de memoria RAM, memoria FLASH y el tiempo de inferencia.

c) Memoria RAM. En este caso se evalúa el porcentaje de uso de memoria RAM en el dispositivo embebido. Se definen la hipótesis nula y alternativa, para luego seguir los pasos de validación de hipótesis:

- **Hipótesis nula y alternativa.** Como primer paso se determinan las hipótesis a evaluar:
 - Hipótesis nula (H_0): No hay diferencia en el uso de memoria RAM por los modelos.
 - Hipótesis alternativa (H_1): Existe al menos una diferencia significativa en el uso de la memoria RAM de los modelos.
- **Verificación de supuestos.** Se realiza la verificación de los supuestos de normalidad y de distribución normal considerando la hipótesis nula para ambas pruebas:
 - Prueba de Shapiro-Wilk: Los datos de uso de memoria RAM siguen una distribución normal.
 - Prueba de Levene: Las varianzas de datos de uso de memoria RAM son iguales entre los grupos.

Observando los resultados (Tabla 52), las hipótesis nulas se rechazan y se debe utilizar pruebas no paramétricas debido a que se tiene valores de p menores a 0.05.

Tabla 52. Verificación de supuestos sobre el uso de memoria RAM

Modelo DNN: Prueba Shapiro-Wilk - Valor p: 0.00025396436103619635 Los datos no siguen una distribución normal. Modelo DT: Prueba Shapiro-Wilk - Valor p: 1.0 Los datos siguen una distribución normal. Modelo MLP: Prueba Shapiro-Wilk - Valor p: 0.00016867939848452806 Los datos no siguen una distribución normal. Modelo RF: Prueba Shapiro-Wilk - Valor p: 1.0 Los datos siguen una distribución normal. Modelo SVM: Prueba Shapiro-Wilk - Valor p: 1.0 Los datos siguen una distribución normal. Prueba de Levene - Valor p: 3.4746793998196943e-31 Recomendación: Considerar pruebas no paramétricas (datos no normales).

- **Determinar diferencias entre los modelos.** Se considera el uso de la prueba Kruskal-Wallis obteniendo que el valor de p es menor que el nivel de significancia 0.05 (Tabla 53). En este caso se rechaza la hipótesis nula por lo que existen diferencias significativas en el uso de memoria RAM.

Tabla 53. Diferencias entre grupos respecto al uso de memoria RAM

Prueba de Kruskal-Wallis: Estadístico = 48.40701407754998, Valor p = 7.762675410772137e-10 Existen diferencias significativas entre los grupos. Proceder con pruebas post-hoc.

- **Pruebas de comparaciones múltiples.** Los resultados muestran variaciones en los recursos de memoria RAM debido a que la prueba de Dunn indica diferencias significativas en varias comparaciones. Por ejemplo, MLP y RF tiene un valor de p muy pequeño, lo que indica un consumo mucho menor por parte de MLP, al igual que entre el modelo DT y RF. Sin embargo, no se encontraron diferencias significativas entre DNN y DT ni entre RF y SVM, lo que sugiere comportamientos similares en estas comparaciones (Tabla 54).

Por lo tanto, MLP es el modelo con el menor uso de memoria RAM, seguido por DT, que son más eficientes que RF y SVM. Este comportamiento hace que MLP sea más eficiente en uso de este recurso en dispositivos embebidos. Sin embargo, si se requiere priorizar la simplicidad de implementación, DT también es una opción viable, dado que no presenta diferencias significativas con MLP y mantiene un bajo consumo de recursos.

Tabla 54. Matriz de resultados de comparaciones múltiples de memoria RAM

	DNN	DT	MLP	RF	SVM
DNN	1.000000	1.000000	1.861425e-02	1.861425e-02	1.000000
DT	1.000000	1.000000	1.000000e+00	3.052684e-05	0.018614
MLP	0.018614	1.000000	1.000000e+00	4.877668e-09	0.000031
RF	0.018614	0.000031	4.877668e-09	1.000000e+00	1.000000
SVM	1.000000	0.018614	3.052684e-05	1.000000e+00	1.000000

d) Memoria Flash. Se evalúa el porcentaje de uso de memoria FLASH utilizada para lo cual, en primer lugar, se definen la hipótesis nula y alternativa, siguiendo los pasos descritos anteriormente:

- **Hipótesis nula y alternativa.** Como primer paso se determinan las hipótesis a evaluar:
 - Hipótesis nula (H_0): No hay diferencia en el uso de memoria FLASH por los modelos.
 - Hipótesis alternativa (H_1): Existe al menos una diferencia significativa en el uso de la memoria FLASH de los modelos.
- **Verificación de supuestos.** La hipótesis nula para ambas pruebas es:
 - Prueba de Shapiro-Wilk: Los datos de uso de memoria FLASH siguen una distribución normal.
 - Prueba de Levene: Las varianzas de datos de uso de memoria FLASH son iguales entre los grupos.

Luego de aplicar las pruebas, se considera que se rechazan las hipótesis nulas por lo que se debe utilizar pruebas no paramétricas (valores obtenidos p menores a 0.05) (Tabla 55).

Tabla 55. Verificación de supuestos sobre el uso de memoria FLASH

Modelo DNN: Prueba Shapiro-Wilk - Valor p: 0.00025396436103619635 Los datos no siguen una distribución normal. Modelo DT: Prueba Shapiro-Wilk - Valor p: 0.008083764463663101 Los datos no siguen una distribución normal. Modelo MLP: Prueba Shapiro-Wilk - Valor p: 0.00016867939848452806 Los datos no siguen una distribución normal. Modelo RF: Prueba Shapiro-Wilk - Valor p: 1.0 Los datos siguen una distribución normal. Modelo SVM: Prueba Shapiro-Wilk - Valor p: 0.25402361154556274 Los datos siguen una distribución normal. Prueba de Levene - Valor p: 1.1193718410538054e-26 Recomendación: Considerar pruebas no paramétricas (datos no normales).
--

- **Determinar diferencias entre los modelos.** Según los resultados obtenidos anteriormente se utilizan pruebas no paramétricas de Kruskal-Wallis obteniendo que el valor de p es menor que el nivel de significancia 0.05 (Tabla 56). Por lo tanto, se rechaza la hipótesis nula indicando que existen diferencias significativas.

Tabla 56. Diferencias entre grupos respecto al uso de memoria FLASH

Prueba de Kruskal-Wallis: Estadístico = 47.71642808452622, Valor p = 1.0813821998726499e-09 Existen diferencias significativas entre los grupos. Proceder con pruebas post-hoc.

- **Pruebas de comparaciones múltiples.** Estas pruebas muestran variaciones importantes entre modelos MLP y RF, mientras que modelos con menor complejidad como DT y MLP, son más eficientes en el uso de memoria FLASH en comparación con RF. Por otro lado, los modelos DT y DNN tienen diferencias significativas, mientras que entre los modelos MLP y DNN tienen comportamientos similares. Por lo tanto, el modelo DT fue el modelo más eficiente, utilizando solo el 6.77 % de la memoria FLASH, seguido por MLP

(9.942 %) y DNN (10.49 %). Por otro lado, SVM y RF registraron un uso mucho mayor de memoria FLASH (Tabla 57).

La interpretación de los resultados muestra que los modelos más complejos, como RF y SVM, son menos adecuados para dispositivos embebidos con restricciones de memoria FLASH. En contraste, modelos más simples como DT y MLP son más eficientes en el uso de memoria FLASH, y muestran diferencias estadísticamente importantes. Por otro lado, RF y SVM deben evitarse en estos contextos, a menos que el dispositivo cuente con recursos suficientes.

Tabla 57. Matriz de resultados de comparaciones múltiples de memoria FLASH

	DNN	DT	MLP	RF	SVM
DNN	1.000000	2.006822e-02	1.000000	2.006822e-02	1.000000
DT	0.020068	1.000000e+00	1.000000	6.473736e-09	0.000036
MLP	1.000000	1.000000e+00	1.000000	3.589595e-05	0.020068
RF	0.020068	6.473736e-09	0.000036	1.000000e+00	1.000000
SVM	1.000000	3.589595e-05	0.020068	1.000000e+00	1.000000

e) Consumo de corriente eléctrica. Se definen la hipótesis nula y alternativa, siguiendo los pasos descritos anteriormente para determinar el comportamiento del consumo de corriente eléctrica al implementar los modelos:

- **Hipótesis nula y alternativa.** Se determinan las hipótesis a evaluar:
 - Hipótesis nula (H0): No hay diferencia en el consumo de corriente eléctrica al implementar los modelos en hardware.
 - Hipótesis alternativa (H1): Existe al menos una diferencia significativa en el consumo de corriente eléctrica al implementar los modelos en hardware.
- **Verificación de supuestos.** En este caso, la hipótesis nula para ambas pruebas es:
 - Prueba de Shapiro-Wilk: Los datos de uso de consumo de corriente eléctrica siguen una distribución normal.
 - Prueba de Levene: Las varianzas de datos de consumo de corriente eléctrica son iguales entre los grupos.

Al obtener los resultados de verificación, se rechazan las hipótesis nulas y se utilizan pruebas no paramétricas (Tabla 58).

Tabla 58. Verificación de supuestos sobre el consumo de corriente eléctrica

Modelo DNN:
Prueba Shapiro-Wilk - Valor p: 0.05530498921871185
Los datos siguen una distribución normal.
Modelo DT:
Prueba Shapiro-Wilk - Valor p: 0.02513469010591507
Los datos no siguen una distribución normal.
Modelo MLP:
Prueba Shapiro-Wilk - Valor p: 0.2742747962474823
Los datos siguen una distribución normal.
Modelo RF:
Prueba Shapiro-Wilk - Valor p: 0.02539902552962303
Los datos no siguen una distribución normal.
Modelo SVM:
Prueba Shapiro-Wilk - Valor p: 0.02513469010591507
Los datos no siguen una distribución normal.
 Prueba de Levene - Valor p: 0.0017851717332900564
Recomendación: Considerar pruebas no paramétricas (datos no normales).

- **Determinar diferencias entre los modelos.** Según los resultados obtenidos anteriormente se utilizan pruebas no paramétricas de Kruskal-Wallis. En esta prueba se rechaza la hipótesis nula por lo que existen diferencias significativas entre los grupos de datos de cada modelo (Tabla 59).

Tabla 59. Diferencias entre grupos respecto al consumo de corriente eléctrica

Prueba de Kruskal-Wallis: Estadístico = 42.57295169082121,
Valor p = 1.2689438352652965e-08
Existen diferencias significativas entre los grupos. Proceder con pruebas post-hoc.

- **Pruebas de comparaciones múltiples.** Los resultados muestran que existen diferencias significativas en algunos casos. Según los valores obtenidos, el modelo más eficiente fue DT, con un consumo de 39.3 mA, seguido por SVM (40.3 mA) y RF (41.46 mA). Por otro lado, los modelos DNN (57.8 mA) y MLP (59.6 mA) registraron los consumos más altos. Además, las pruebas de Dunn muestran diferencias significativas, considerando los p valores, entre el modelo de DT con DNN y MLP y entre el modelo SVM con MLP y DNN. Sin embargo, no hubo

diferencias importantes entre DT con SVM y RF, de lo cual se infiere que estos últimos tienen comportamientos de consumo similares (Tabla 60).

Los resultados confirman que DT es el modelo más eficiente en relación con el consumo de corriente eléctrica, seguido por SVM y RF, que no presentan diferencias significativas entre sí. Estos tres modelos podrían ser utilizados en dispositivos con restricciones de consumo eléctrico. Por otro lado, el uso de DNN o MLP debería considerarse sólo si el sistema embebido es energizado de manera constante y si las ventajas de exactitud son más importantes que el consumo de corriente eléctrica.

Tabla 60. Matriz de resultados de comparaciones múltiples sobre corriente eléctrica

	DNN	DT	MLP	RF	SVM
DNN	1.000000	1.034759e-05	1.000000e+00	0.232460	0.002027
DT	0.000010	1.000000e+00	9.762441e-07	0.089086	1.000000
MLP	1.000000	9.762441e-07	1.000000e+00	0.066166	0.000316
RF	0.232460	8.908582e-02	6.616607e-02	1.000000	1.000000
SVM	0.002027	1.000000e+00	3.157630e-04	1.000000	1.000000

f) Tamaño del archivo de modelo. Se determina el tamaño del archivo que contiene el programa codificado de los modelos. Se definen la hipótesis nula y alternativa, siguiendo los pasos descritos anteriormente:

- **Hipótesis nula y alternativa.** Se determinan las hipótesis a evaluar:
 - Hipótesis nula (H_0): No hay diferencia en el tamaño de los archivos que corresponden a cada modelo
 - Hipótesis alternativa (H_1): Existe al menos una diferencia significativa en el tamaño de los archivos que corresponden a cada modelo.
- **Verificación de supuestos.** En este caso se verifican para este caso el supuesto de normalidad y varianzas, donde la hipótesis nula para ambas pruebas es:
 - Prueba de Shapiro-Wilk: Los datos el tamaño del archivo de cada modelo siguen una distribución normal.
 - Prueba de Levene: Las varianzas de datos relacionados al tamaño de cada modelo son iguales entre los grupos.

Al obtener los resultados de verificación, se rechazan las hipótesis nulas y se utilizan pruebas no paramétricas (Tabla 61).

Tabla 61. Verificación de supuestos sobre el tamaño de archivo para cada modelo

Modelo DNN:
Prueba Shapiro-Wilk - Valor p: 0.00026556203374639153
Los datos no siguen una distribución normal.
Modelo DT:
Prueba Shapiro-Wilk - Valor p: 0.05587879568338394
Los datos siguen una distribución normal.
Modelo MLP:
Prueba Shapiro-Wilk - Valor p: 5.0543858378659934e-05
Los datos no siguen una distribución normal.
Modelo RF:
Prueba Shapiro-Wilk - Valor p: 0.032429520040750504
Los datos no siguen una distribución normal.
Modelo SVM:
Prueba Shapiro-Wilk - Valor p: 0.107515849173069
Los datos siguen una distribución normal.
 Prueba de Levene - Valor p: 8.326986383449957e-23
Recomendación: Considerar pruebas no paramétricas (datos no normales).

- **Determinar diferencias entre los modelos.** Los siguientes resultados se obtuvieron usando pruebas no paramétricas de Kruskal-Wallis (Tabla 62). En esta prueba se rechaza la hipótesis nula por lo que existen diferencias significativas entre los grupos de datos de cada modelo.

Tabla 62. Diferencias entre grupos respecto al tamaño de los archivos

Prueba de Kruskal-Wallis: Estadístico = 43.69186326432353, Valor p = 7.434321064035097e-09 Existen diferencias significativas entre los grupos. Proceder con pruebas post-hoc.
--

- **Pruebas de comparaciones múltiples.** Estas pruebas muestran diferencias significativas entre algunos pares de modelos. En el caso del modelo DT este tiene el menor tamaño de archivo que contienen el modelo (59.7 KB), lo que lo hace eficiente en términos de almacenamiento mientras que el modelo MLP, tiene valores similares. Sin embargo, tanto el modelo DT como MLP muestran diferencias respecto a los modelos RF y SVM, que tienen tamaños mayores. Sobre los modelos basados en DNN, tiene un tamaño promedio de 199.98 KB, lo cual hace que esté en una posición intermedia entre los modelos más ligeros (Tabla 63).

En aplicaciones donde el tamaño del archivo del modelo es importante el modelo DT es la opción adecuada, seguido de MLP. Finalmente, sobre el modelo DNN, se puede usar si la restricción de almacenamiento no crítica, mientras que RF y SVM no son recomendables debido a su gran tamaño.

Tabla 63. Resultados de comparaciones múltiples sobre el tamaño de los archivos

	DNN	DT	MLP	RF	SVM
DNN	1.000000	9.024323e-02	0.458526	0.653069	5.697041e-02
DT	0.090243	1.000000e+00	1.000000	0.000084	7.621860e-07
MLP	0.458526	1.000000e+00	1.000000	0.001231	1.921833e-05
RF	0.653069	8.417408e-05	0.001231	1.000000	1.000000e+00
SVM	0.056970	7.621860e-07	0.000019	1.000000	1.000000e+00

CAPÍTULO 7: CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

En esta investigación se elaboró y validó una técnica computacional implementada en un dispositivo embebido, basada en algoritmos de clasificación (SVM, RF, DT, MLP y DNN), para el monitoreo de señales fisiológicas relacionadas a cardiopatía. En relación a los objetivos específicos, se determinó la exactitud de los modelos en el dispositivo de hardware, consumo de corriente, tiempo de inferencia y el uso de memoria RAM y memoria FLASH empleado. Estas métricas se usaron para comparaciones cuantitativas y estadísticas que evalúan el comportamiento de los modelos.

Se realizó la evaluación de tiempo de predicción de los modelos en el dispositivo embebido obteniendo que el modelo DT es el que ofrece el menor tiempo de predicción y tiene diferencias significativas en su desempeño respecto a algunos modelos, como DNN y RF. Esto respalda su selección como el modelo más eficiente en términos de tiempo. No obstante, las diferencias no son significativas frente a modelos como SVM y MLP, lo que sugiere que podrían considerarse como alternativas razonables dependiendo de otros criterios de evaluación.

Sobre la evaluación de la exactitud resultante al evaluar los modelos en dispositivo embebido, se recomienda considerar otros aspectos relevantes para la toma de decisiones, como el tiempo de predicción o el consumo de memoria. Aunque las pruebas estadísticas no identificaron diferencias significativas entre los modelos, la comparación visual destaca al modelo Random Forest (RF) como el mejor candidato, gracias a su exactitud promedio más alta (0.96601), seguido del modelo MLP y DT siendo la opción más adecuada para ser implementada en un sistema que priorice la exactitud como criterio principal.

El análisis de uso de memoria RAM demuestra que existen diferencias significativas entre los modelos evaluados, confirmando la hipótesis alternativa. En este caso MLP se destaca como el modelo más eficiente en términos de uso de memoria RAM, siendo una opción adecuada para dispositivos de hardware con recursos limitados. Además, también se puede considerar el uso del modelo DT por su simplicidad, evitando el uso de modelos como RF o SVM debido a su elevado consumo de memoria.

El análisis del uso de memoria FLASH confirma la hipótesis alternativa (existen diferencias significativas en el uso de memoria FLASH entre los modelos evaluados) donde los modelos DT y MLP se muestran como las mejores opciones para dispositivos de hardware limitados, debido a su bajo consumo de memoria FLASH y su eficiencia estadística comprobada en comparación a otros modelos como RF y SVM. Se concluye que DT es la mejor elección para optimizar el uso de memoria FLASH en aplicaciones embebidas, seguido de MLP.

Sobre los resultados del consumo de corriente eléctrica, existen diferencias significativas. donde el modelo DT es el más eficiente energéticamente, y no tiene diferencias significativas frente a SVM y RF superando ampliamente a DNN y MLP. Por lo tanto, en este caso, se recomienda el uso de DT en aplicaciones donde el consumo de corriente eléctrica y la autonomía de funcionamiento es un factor crítico. En cambio, DNN y MLP deben reservarse para escenarios donde el rendimiento justifique su mayor demanda energética.

En el caso de la diferencia en el tamaño de los archivos de los modelos, se determina que la hipótesis nula (H_0) se rechaza ya que se encontraron diferencias significativas en el tamaño de los archivos. Los modelos DT y MLP son los modelos más eficientes en términos de tamaño de archivo, siendo ideales para aplicaciones con almacenamiento limitado. Finalmente, los modelos RF y SVM, son recomendables solo en casos donde el espacio de almacenamiento no sea una limitación.

En resumen, en términos de costo-beneficio, el modelo DT tiene mejores resultados de comportamiento en tiempo de predicción, bajo consumo de memoria (RAM y FLASH), tamaño reducido de archivo y consumo de corriente eléctrica mientras que el modelo MLP (aunque es un poco menos eficiente en consumo de corriente eléctrica, memoria FLASH y RAM), tiene un buen equilibrio entre exactitud, uso de memoria y tamaño. Por otro lado, los modelos RF y SVM destacan sólo en la exactitud, pero su elevado consumo de memoria y corriente eléctrica los limita a aplicaciones donde la exactitud es lo más importante. Finalmente, los modelos DNN, son adecuados para casos donde se prioriza el tiempo de predicción y exactitud sobre la optimización de recursos.

Finalmente, a partir de evaluación y selección tecnológica y metodológica, se indica de manera explícita la relación entre los conceptos de computación de borde, TinyML,

métodos de validación y metodologías como CRISP-DM con la técnica computacional propuesta en esta investigación. A diferencia de enfoques que abordan estas tecnologías de forma aislada, en esta investigación se utiliza un hardware embebido para la evaluación de modelos. Es por ello, que no sólo aplica modelos de clasificación, sino que delimita un procedimiento organizado para su entrenamiento, validación e integración en hardware.

7.2 Trabajos futuros

Se recomienda explorar métodos de compresión y cuantización de modelos para reducir el uso de memoria RAM y FLASH considerando procesos como el truncamiento de redes neuronales en modelos complejos, como DNN o RF, sin afectar significativamente el comportamiento del hardware. Otro aspecto relevante es el análisis de nuevos algoritmos de aprendizaje automático que puedan ofrecer un equilibrio entre exactitud y consumo de recursos de hardware.

Una línea de investigación puede estar relacionada a considerar la optimización del consumo de energía y corriente eléctrica mediante la implementación de estrategias de apagado de componentes cuando no se encuentren en uso. Esto puede permitir aumentar la autonomía de los dispositivos sin comprometer la capacidad de procesamiento. Además, también se puede evaluar el impacto de diferentes arquitecturas de hardware y el rendimiento de los modelos, como microcontroladores, computadoras de placa reducida, procesadores digitales de señales, FPGA o aplicando métodos de procesamiento neuromórfico.

Finalmente, se pueden realizar pruebas en entornos reales con pacientes o entornos clínicos, lo cual permite consolidar la eficiencia de la solución basadas en aprendizaje automático en dispositivos embebidos con bajos recursos de hardware, permitiendo su adopción en sistemas de monitoreo fisiológico en tiempo real.

Bibliografía

- Abid, M., Khabou, A., Ouakrim, Y., Watel, H., Chemcki, S., Mitiche, A., Benazza-Benyahia, A., & Mezghani, N. (2021). Physical Activity Recognition Based on a Parallel Approach for an Ensemble of Machine Learning and Deep Learning Classifiers. *Sensors* 2021, Vol. 21, Page 4713, 21(14), 4713. <https://doi.org/10.3390/S21144713>
- Abu-Samah, A., Ghaffa, D., Abdullah, N. F., Kamal, N., Nordin, R., Dela Cruz, J. C., Magwili, G. V., & Mercado, R. J. (2025). Deployment of TinyML-Based Stress Classification Using Computational Constrained Health Wearable. In *Electronics (Switzerland)* (Vol. 14, Issue 4). Multidisciplinary Digital Publishing Institute. <https://doi.org/10.3390/electronics14040687>
- Adolfo, C. M. S., Chizari, H., Win, T. Y., & Al-Majeed, S. (2021). Sample reduction for physiological data analysis using principal component analysis in artificial neural network. *Applied Sciences (Switzerland)*, 11(17). <https://doi.org/10.3390/APP11178240>
- Ahamed, F., Shahrestani, S., & Cheung, H. (2020). Internet of things and machine learning for healthy ageing: Identifying the early signs of dementia. *Sensors (Switzerland)*, 20(21), 1–25. <https://doi.org/10.3390/s20216031>
- Ahmed, I., Jeon, G., & Piccialli, F. (2021). A Deep-Learning-Based Smart Healthcare System for Patient's Discomfort Detection at the Edge of Internet of Things. *IEEE Internet of Things Journal*, 8(13), 10318–10326. <https://doi.org/10.1109/JIOT.2021.3052067>
- Alabdulqader, N., Riad*, K., & Almarri, B. (2024). Empowering Home Care: Utilizing IoT and Deep Learning for Intelligent Monitoring and Management of Chronic Diseases. *International Journal of Advanced Computer Science and Applications*, 15(12), 983–995. <https://doi.org/10.14569/IJACSA.2024.0151298>
- Albeshri, A. (2021). SVSL: A human activity recognition method using soft-voting and self-learning. *Algorithms*, 14(8). <https://doi.org/10.3390/a14080245>
- Alcázar-Aguilar, O. O., Castro-Yanahida, J. E., Rodríguez-Vargas, M. C., Gil-Cueva, S. L., & Cebrian-Centeno, E. L. (2021). Recomendaciones dirigidas a los familiares responsables del cuidado domiciliario de un paciente diagnosticado con Covid-19. *Revista Peruana de Investigación En Salud*, 5(1), 40–49. <https://doi.org/10.35839/repis.5.1.790>
- Alian, A. A., & Shelley, K. H. (2014). Photoplethysmography. *Best Practice & Research Clinical Anaesthesiology*, 28(4), 395–406. <https://doi.org/10.1016/J.BPA.2014.08.006>
- Alshehri, F., & Muhammad, G. (2021). A Comprehensive Survey of the Internet of Things (IoT) and AI-Based Smart Healthcare. *IEEE Access*, 9, 3660–3678. <https://doi.org/10.1109/ACCESS.2020.3047960>
- Amin, S. U., & Hossain, M. S. (2021). Edge Intelligence and Internet of Things in Healthcare: A Survey. *IEEE Access*, 9, 45–59. <https://doi.org/10.1109/ACCESS.2020.3045115>
- Amudha, S., & Murali, M. (2021). Implementation of Energy Efficient Fog based Health Monitoring and Emergency Admission Prediction System Using IoT.

- Webology*, Volume 18(Special Issue on Information Retrieval and Web Search), 171–189. <https://doi.org/10.14704/WEB/V18SI02/WEB18065>
- Andellini, M., Haleem, S., Angelini, M., Ritrovato, M., Schiaffini, R., Iadanza, E., & Pecchia, L. (2023). Artificial intelligence for non-invasive glycaemic-events detection via ECG in a paediatric population: study protocol. *Health and Technology*, 13(1), 145–154. <https://doi.org/10.1007/s12553-022-00719-x>
- Andriulo, F. C., Fiore, M., Mongiello, M., Traversa, E., & Zizzo, V. (2024). Edge Computing and Cloud Computing for Internet of Things: A Review. *Informatics 2024*, Vol. 11, Page 71, 11(4), 71. <https://doi.org/10.3390/INFORMATICS11040071>
- Ansari, S. (2023). Building Computer Vision Applications Using Artificial Neural Networks. In *Building Computer Vision Applications Using Artificial Neural Networks* (second). Apress. <https://doi.org/10.1007/978-1-4842-9866-4>
- Arias-Gómez, J., Ángel Villasís-Keever, M., & Guadalupe Miranda-Novales, M. (2016). Metodología de la Investigación. In *Revista Alergia México* (Vol. 63, pp. 201–206). www.nietoeditores.com.mx
- Arias, F. (2016). *El proyecto de investigación Introducción a la investigación* (EPISTEME (ed.)).
- Armando, R., Herrera, J., & Sandoval, S. (2023). *Internet de las cosas Teoría y práctica enfoque académico*. Universidad De Colima. <https://doi.org/10.53897/LI.2023.0001.UCOL>
- Arora, M. (2016). *Embedded System Design: Introduction to SoC System Architecture*. Learning Bytes Publishing. <https://www.amazon.com/Embedded-System-Design-Introduction-Architecture-ebook/dp/B01FY3LLHI>
- Arun, S., Sykes, E. R., & Tanbeer, S. (2025). MobiVitalsConnect: A Comprehensive Mobile Healthcare System for Real-Time Patient Monitoring and Data Visualization. *Communications in Computer and Information Science*, 2259 CCIS(1), 49–63. https://doi.org/10.1007/978-3-031-85908-3_4/FIGURES/9
- Asghar, M. A., Khan, M. J., Shahid, H., Shorfuzzaman, M., Xiong, N. N., & Mehmood, R. M. (2021). Semi-skipping layered gated unit and efficient network: Hybrid deep feature selection method for edge computing in EEG-based emotion classification. *IEEE Access*, 9, 13378–13389. <https://doi.org/10.1109/ACCESS.2021.3051808>
- Ashfaq, Z., Mumtaz, R., Rafay, A., Zaidi, S. M. H., Saleem, H., Mumtaz, S., Shahid, A., De Poorter, E., & Moerman, I. (2022). Embedded AI-Based Digi-Healthcare. *Applied Sciences (Switzerland)*, 12(1). <https://doi.org/10.3390/APP12010519>
- Ashley, D. (2020). Using Flask and Jinja. In *Foundation Dynamic Web Pages with Python* (pp. 159–181). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6339-6_5
- Atlam, H. F., Walters, R. J., & Wills, G. B. (2019, January 25). Intelligence of things: Opportunities challenges. *3rd Cloudification of the Internet of Things Conference, CIoT 2018*. <https://doi.org/10.1109/CIOT.2018.8627114>
- Atwal, H. (2020). *Practical DataOps : delivering agile data science at scale*. Apress.
- Babiuch, M., Foltyniek, P., & Smutny, P. (2019). Using the ESP32 microcontroller for

- data processing. *Proceedings of the 2019 20th International Carpathian Control Conference, ICCC* 2019. <https://doi.org/10.1109/CARPATHIANCC.2019.8765944>
- Badamasi, Y. A. (2014). The working principle of an Arduino. *Proceedings of the 11th International Conference on Electronics, Computer and Computation, ICECCO 2014*. <https://doi.org/10.1109/ICECCO.2014.6997578>
- Banerjee, R. (2023). HANDS-ON TINYML. In B. Online (Ed.), *Hands-on TinyML* (p. 2023). BPB PUBLICATIONS.
- Behmanesh, A., Sayfour, N., & Sadoughi, F. (2020). Technological Features of Internet of Things in Medicine: A Systematic Mapping Study. In *Wireless Communications and Mobile Computing* (Vol. 2020). Hindawi Limited. <https://doi.org/10.1155/2020/9238614>
- Berta, R., Dabbous, A., Lazzaroni, L., Pau, D. Pietro, & Bellotti, F. (2024). Developing a TinyML Image Classifier in an Hour. *IEEE Open Journal of the Industrial Electronics Society*, 5(1), 946–960. <https://doi.org/10.1109/OJIES.2024.3451959>
- Bhardwaj, V., Rajat Joshi, ., Anshu, ., & Gaur, M. (2022). IoT-Based Smart Health Monitoring System for COVID-19. *SN Computer Science* 2022 3:2, 3(2), 1–11. <https://doi.org/10.1007/S42979-022-01015-1>
- Bhasker, B., Rao, P. M., Saraswathi, P., Patro, S. G. K., Bhutto, J. K., Islam, S., Kareemullah, M., & Emma, A. F. (2025). Blockchain framework with IoT device using federated learning for sustainable healthcare systems. *Scientific Reports*, 15(1), 26736. <https://doi.org/10.1038/s41598-025-06539-z>
- Bhat, G. S., Shankar, N., Kim, D., Song, D. J., Seo, S., Panahi, I. M. S., & Tamil, L. (2021). Machine learning-based asthma risk prediction using IoT and smartphone applications. *IEEE Access*, 9, 118708–118715. <https://doi.org/10.1109/ACCESS.2021.3103897>
- Bisantz, A., & Burns, C. (2014). *Cognitive Systems Engineering in Health Care* (p. 224). CRC Press. <https://doi.org/10.1201/b17737>
- Bizzego, A., Gabrieli, G., Furlanello, C., & Esposito, G. (2020). Comparison of wearable and clinical devices for acquisition of peripheral nervous system signals. *Sensors (Switzerland)*, 20(23), 1–16. <https://doi.org/10.3390/S20236778>
- Bobadilla, J. (2021). Machine Learning y deep learning usando python, scikit y keras. In D. Editores (Ed.), *Editorial Ra-ma (España)*. Ra-Ma. <https://books.google.es/books?hl=es&lr=&id=iAAyEAAAQBAJ&oi=fnd&pg=PA111&dq=deep+learning&ots=QiA2yVIO3u&sig=P5souOlap80IcBm5zOgrRyQ555Q#v=onepage&q=deep+learning&f=false>
- Caballero Romero, A. (2014). *Metodología integral innovadora para planes y tesis: LA METODOLOGÍA DEL CÓMO FORMULARLOS* (M. R. S. Ramírez (ed.); 1st ed.). Cengage Learning. <https://www.grupoases.pe/libros/4.pdf>
- Cameron, N. (2021). Electronics Projects with the ESP8266 and ESP32. In *Electronics Projects with the ESP8266 and ESP32*. Apress. <https://doi.org/10.1007/978-1-4842-6336-5>
- Cao, J., Zhang, Q., & Shi, W. (2018). *Edge Computing: A Primer*. Springer International Publishing. <https://doi.org/10.1007/978-3-030-02083-5>

- Cao, M., Ramezani, R., Katakwar, V. K., Zhang, W., Boda, D., Wani, M., & Naeim, A. (2024). Developing remote patient monitoring infrastructure using commercially available cloud platforms. *Frontiers in Digital Health*, 6(1), 10. <https://doi.org/10.3389/fdgth.2024.1399461>
- Chang, Y.-H., Wu, F.-C., & Lin, H.-W. (2025). Design and Implementation of ESP32-Based Edge Computing for Object Detection. *Sensors*, 25(6), 1656. <https://doi.org/10.3390/s25061656>
- Chidozie, L. (2022). *CVD_Vital_Signs*. Kaggle. <https://www.kaggle.com/datasets/chidozieuzoegwu/cvd-vital-signs>
- Chowdhury, M. H., Reaz, M. B. I., Ali, S. H. M., Khan, M. S., & Chowdhury, M. E. H. (2025). ROSE-Net: Leveraging remote photoplethysmography to estimate oxygen saturation using deep learning. *Biomedical Signal Processing and Control*, 100(1), 107105. <https://doi.org/10.1016/j.bspc.2024.107105>
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. <https://doi.org/10.1007/bf00994018>
- Dahnke, G. L., Fernández-Collado, C., & Macotela, F. (1986). *La comunicación humana: ciencia social* (D. F.: M.-H. México (ed.)). McGraw-Hill. https://ulatina.metabiblioteca.org/cgi-bin/koha/opac-detail.pl?biblionumber=4951&shelfbrowse_itemnumber=7050
- Dall’Ora, N., Felli, L., Aldegheri, S., Vicino, N., & Giuliano, R. (2025). LumiCare: A Context-Aware Mobile System for Alzheimer’s Patients Integrating AI Agents and 6G. *Electronics*, 14(17), 3516. <https://doi.org/10.3390/electronics14173516>
- David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2020). *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*. <https://arxiv.org/abs/2010.08678v3>
- de Melo, F., Neto, H. C., & da Silva, H. P. (2022). System on Chip (SoC) for Invisible Electrocardiography (ECG) Biometrics. *Sensors (Basel, Switzerland)*, 22(1). <https://doi.org/10.3390/S22010348>
- de Prado, M., Rusci, M., Capotondi, A., Donze, R., Benini, L., & Pazos, N. (2021). Robustifying the Deployment of tinyML Models for Autonomous Mini-Vehicles. *Sensors* 2021, Vol. 21, Page 1339, 21(4), 1339. <https://doi.org/10.3390/S21041339>
- Devi, R. L., & Kalaivani, V. (2019). Machine learning and IoT-based cardiac arrhythmia diagnosis using statistical and dynamic features of ECG. *The Journal of Supercomputing* 2019 76:9, 76(9), 6533–6544. <https://doi.org/10.1007/S11227-019-02873-Y>
- Dhanwanth, B., Saravanakumar, R., Tamilselvi, T., & Revathi, K. (2023). A Smart Remote Monitoring System for Prenatal Care in Rural Areas. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(3), 30–36. <https://doi.org/10.17762/IJRITCC.V11I3.6196>
- Di Leo, S., De Cicco, L., & Mascolo, S. (2025). Real-Time Speech-to-Text on Edge: A Prototype System for Ultra-Low Latency Communication with AI-Powered NLP. *Information*, 16(8), 685. <https://doi.org/10.3390/info16080685>
- Dietzel, I. D., Happel, P., & Schäffer, T. E. (2022). The Evolution of Scanning Ion

- Conductance Microscopy. In *Bioanalytical Reviews* (Vol. 3, pp. 1–21). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/11663_2022_14
- Djelouat, H., Al Disi, M., Boukhenoufa, I., Amira, A., Bensaali, F., Kotronis, C., Politi, E., Nikolaidou, M., & Dimitrakopoulos, G. (2020). Real-time ECG monitoring using compressive sensing on a heterogeneous multicore edge-device. *Microprocessors and Microsystems*, 72, 102839. <https://doi.org/https://doi.org/10.1016/j.micpro.2019.06.009>
- Dokic, K., Martinovic, M., & Mandusic, D. (2020, September 1). Inference speed and quantisation of neural networks with TensorFlow Lite for Microcontrollers framework. *SEEDA-CECNSM 2020 - 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference*. <https://doi.org/10.1109/SEEDA-CECNSM49515.2020.9221846>
- Dolui, K., & Datta, S. K. (2017, August 23). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. *GloTS 2017 - Global Internet of Things Summit, Proceedings*. <https://doi.org/10.1109/GIOTS.2017.8016213>
- Dong, P., Ning, Z., Obaidat, M. S., Jiang, X., Guo, Y., Hu, X., Hu, B., & Sadoun, B. (2020). Edge Computing Based Healthcare Systems: Enabling Decentralized Health Monitoring in Internet of Medical Things. *IEEE Network*, 34(5), 254–261. <https://doi.org/10.1109/MNET.011.1900636>
- Doroudi, S. (2024). On the paradigms of learning analytics: Machine learning meets epistemology. *Computers and Education: Artificial Intelligence*, 6(1), 100192. <https://doi.org/10.1016/j.caeai.2023.100192>
- Dreyfus, H. (2014). *INTELIGENCIA ARTIFICIAL: CONDICIONES DE POSIBILIDAD TÉCNICAS Y SOCIALES PARA LA CREACIÓN DE MÁQUINAS PENSANTES* [Universidad Complutense de Madrid]. <https://dialnet.unirioja.es/servlet/tesis?codigo=99269&info=resumen&idioma=SPA>
- Duobiene, S., Ratautas, K., Trusovas, R., Ragulis, P., Šlekas, G., Simniškis, R., & Račiukaitis, G. (2022). Development of Wireless Sensor Network for Environment Monitoring and Its Implementation Using SSAIL Technology. *Sensors (Switzerland)*, 22(14). <https://doi.org/10.3390/s22145343>
- Efendi, A., Ammarullah, M. I., Isa, I. G. T., Sari, M. P., Izza, J. N., Nugroho, Y. S., Nasrullah, H., & Alfian, D. (2025). IoT-Based Elderly Health Monitoring System Using Firebase Cloud Computing. *Health Science Reports*, 8(3), 10. <https://doi.org/10.1002/hsr2.70498>
- Ekiz, D., Can, Y. S., Dardagan, Y. C., & Ersoy, C. (2020). Can a Smartband be Used for Continuous Implicit Authentication in Real Life. *IEEE Access*, 8, 59402–59411. <https://doi.org/10.1109/ACCESS.2020.2982852>
- El Attaoui, A., Largo, S., Kaissari, S., Benba, A., Jilbab, A., & Bourouhou, A. (2020). Machine learning-based edge-computing on a multi-level architecture of WSN and IoT for real-time fall detection. *IET WIRELESS SENSOR SYSTEMS*, 10(6), 320–332. <https://doi.org/10.1049/iet-wss.2020.0091>
- El Ghmary, M., Hmimz, Y., Chanyour, T., & Cherkaoui Malki, M. O. (2020). Time and resource constrained offloading with multi-task in a mobile edge computing

- node. *International Journal of Electrical and Computer Engineering*, 10(4), 3757–3766. <https://doi.org/10.11591/IJECE.V10I4.PP3757-3766>
- Elhadad, A., Alanazi, F., Taloba, A. I., & Abozeid, A. (2022). Fog Computing Service in the Healthcare Monitoring System for Managing the Real-Time Notification. *Journal of Healthcare Engineering*, 2022, 1–11. <https://doi.org/10.1155/2022/5337733>
- Emetere, M. E., & Akinlabi, E. T. (2020). Introduction to Computational Techniques. In *Lecture Notes in Networks and Systems* (Vol. 58, pp. 63–77). Springer. https://doi.org/10.1007/978-3-030-36207-2_3
- Eneque Suarez, G. V. (2025). Análisis comparativo de servicios y funcionalidades de AWS, Azure y Elastic Cloud en el entorno de Cloud Computing [Universidad Señor de Sipán]. In *Repositorio Institucional - USS*. <https://repositorio.uss.edu.pe/handle/20.500.12802/14113>
- Escalona, O., Cullen, N., Weli, I., McCallan, N., Ng, K. Y., & Finlay, D. (2023). Robust Arm Impedocardiography Signal Quality Enhancement Using Recursive Signal Averaging and Multi-Stage Wavelet Denoising Methods for Long-Term Cardiac Contractility Monitoring Armbands. *Sensors*, 23(13), 5892. <https://doi.org/10.3390/s23135892>
- Faragli, A., Abawi, D., Quinn, C., Cvetkovic, M., Schlabs, T., Tahirovic, E., Düngen, H. D., Pieske, B., Kelle, S., Edelmann, F., & Alogna, A. (2021). The role of non-invasive devices for the telemonitoring of heart failure patients. In *Heart Failure Reviews* (Vol. 26, Issue 5, pp. 1063–1080). Springer. <https://doi.org/10.1007/s10741-020-09963-7>
- Foukalas, F. (2020). Cognitive IoT platform for fog computing industrial applications. *Computers and Electrical Engineering*, 87, 106770. <https://doi.org/10.1016/j.compeleceng.2020.106770>
- Gallos, P., Menychtas, A., Panagopoulos, C., Kaselimi, M., Temenos, A., Rallis, I., Doulamis, A., Doulamis, N., Bimpas, M., Aggeli, A., Protopapadakis, E., Sardis, E., & Maglogiannis, I. (2022). Using mHealth Technologies to Promote Public Health and Well-Being in Urban Areas with Blue-Green Solutions. *Studies in Health Technology and Informatics*, 295, 566–569. <https://doi.org/10.3233/SHTI220791>
- Gembaczka, P., Heidemann, B., Bennertz, B., Groeting, W., Norgall, T., & Seidl, K. (2019). Combination of sensor-embedded and secure server-distributed artificial intelligence for healthcare applications. *Current Directions in Biomedical Engineering*, 5(1), 29–32. <https://doi.org/10.1515/CDBME-2019-0008>
- Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. In *O'Reilly Media*. <https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>
- Ghafari, R., & Mansouri, N. (2025). Swarm intelligence techniques and their applications in fog/edge computing: an in-depth review. *Artificial Intelligence Review*, 58(11), 359. <https://doi.org/10.1007/s10462-025-11351-2>
- Ghasemzadeh, H., Amini, N., Saeedi, R., & Sarrafzadeh, M. (2015). Power-aware computing in wearable sensor networks: An optimal feature selection. *IEEE Transactions on Mobile Computing*, 14(4), 800–812.

<https://doi.org/10.1109/TMC.2014.2331969>

- Gloria, A. F. X., & Sebastiao, P. J. A. (2021). Autonomous Configuration of Communication Systems for IoT Smart Nodes Supported by Machine Learning. *IEEE Access*, 9, 75021–75034. <https://doi.org/10.1109/ACCESS.2021.3081794>
- Głowania, S., Kozak, J., & Juszczuk, P. (2023). Knowledge Discovery in Databases for a Football Match Result. *Electronics* 2023, Vol. 12, Page 2712, 12(12), 2712. <https://doi.org/10.3390/ELECTRONICS12122712>
- González Arencibia, M., & Martínez Cardero, D. (2020). Dilemas éticos en el escenario de la inteligencia artificial. *Economía y Sociedad*, 25(57), 1–18. <https://doi.org/10.15359/ey.25-57.5>
- Gowda, M., Gowda, J., Iyer, S., Pawar, M., & Gaikwad, V. (2020). Power Consumption Optimization in IoT based Wireless Sensor Node Using ESP8266. *ITM Web of Conferences*, 32, 03048. <https://doi.org/10.1051/ITMCONF/20203203048>
- Gragnaniello, M., Borghese, A., Marrazzo, V. R., Maresca, L., Breglio, G., Irace, A., & Riccio, M. (2024). Real-Time Myocardial Infarction Detection Approaches with a Microcontroller-Based Edge-AI Device. *Sensors*, 24(3), 828. <https://doi.org/10.3390/s24030828>
- Guijarro, V., & González, L. (2010). *La quimera del autómatas matemático. Del calculador medeal a la máquina analítica de Babbage*. Cátedra. <https://dialnet.unirioja.es/servlet/libro?codigo=499951>
- Guo, H., & Yang, T. (2022). A remote consultation system for sports injury based on wireless sensor network. *EAI Endorsed Transactions on Pervasive Health and Technology*, 8(31). <https://doi.org/10.4108/EETPHT.V8I31.701>
- Gupta, A. K., & Nandyala, S. P. (2023). *Deep learning on microcontrollers : learn how to develop embedded AI applications using TinyML*.
- Gupta, N., Khosravy, M., Patel, N., Dey, N., & Crespo, R. G. (2022). Lightweight Computational Intelligence for IoT Health Monitoring of Off-Road Vehicles: Enhanced Selection Log-Scaled Mutation GA Structured ANN. *IEEE Transactions on Industrial Informatics*, 18(1), 611–619. <https://doi.org/10.1109/TII.2021.3072045>
- Haghi Kashani, M., Madanipour, M., Nikravan, M., Asghari, P., & Mahdipour, E. (2021). A systematic review of IoT in healthcare: Applications, techniques, and trends. In *Journal of Network and Computer Applications* (Vol. 192, p. 103164). Academic Press. <https://doi.org/10.1016/j.jnca.2021.103164>
- Haghi, M., Danyali, S., Ayasseh, S., Wang, J., Aazami, R., & Deserno, T. M. (2021). Wearable devices in health monitoring from the environmental towards multiple domains: A survey. *Sensors*, 21(6), 1–24. <https://doi.org/10.3390/s21062130>
- Hameurlaine, M., Moussaoui, A., & Bensalah, M. (2025). Real-Time Smart Healthcare System Based on Edge-Internet of Things and Deep Neural Networks for Heart Disease Prediction. *Informatica*, 49(1), 93–104. <https://doi.org/10.31449/inf.v49i1.6875>
- Hamil, H., Zidelmal, Z., Azzaz, M. S., Sakhi, S., Kaibou, R., Djilali, S., & Ould Abdeslam, D. (2022). Design of a secured telehealth system based on multiple biosignals diagnosis and classification for IoT application. *Expert Systems*, 39(4).

<https://doi.org/10.1111/EXSY.12765>

- Hamilton, P. S., & Tompkins, W. J. (1986). Quantitative Investigation of QRS Detection Rules Using the MIT/BIH Arrhythmia Database. *IEEE Transactions on Biomedical Engineering*, *BME-33*(12), 1157–1165. <https://doi.org/10.1109/TBME.1986.325695>
- Hannan, A., Shafiq, M. Z., Hussain, F., & Pires, I. M. (2021). A Portable Smart Fitness Suite for Real-Time Exercise Monitoring and Posture Correction. *Sensors* 2021, Vol. 21, Page 6692, 21(19), 6692. <https://doi.org/10.3390/S21196692>
- Haslwanter, T. (2016). An Introduction to Statistics with Python - With Applications in the Life Sciences | Thomas Haslwanter | Springer. In *Springer*. <https://www.springer.com/gp/book/9783319283159>
- Hassantabar, S., Zhang, J., Yin, H., & Jha, N. K. (2022). MHDeep: Mental Health Disorder Detection System Based on Wearable Sensors and Artificial Neural Networks. *ACM Transactions on Embedded Computing Systems*, 21(6). <https://doi.org/10.1145/3527170>
- Hayajneh, A. M., Hafeez, M., Zaidi, S. A. R., & McLernon, D. (2024). TinyML Empowered Transfer Learning on the Edge. *IEEE Open Journal of the Communications Society*, 5(1), 1656–1672. <https://doi.org/10.1109/OJCOMS.2024.3373177>
- Hayyolalam, V., Aloqaily, M., Ozkasap, O., & Guizani, M. (2021). Edge Intelligence for Empowering IoT-Based Healthcare Systems. *IEEE Wireless Communications*, 28(3), 6–14. <https://doi.org/10.1109/MWC.001.2000345>
- Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación. In P. Baptista (Ed.), *Online Learning Center*. McGraw-Hill Education, 2014.
- Hernández Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta*. McGraw-Hill Education. https://books.google.com/books/about/METODOLOGÍA_DE_LA_INVESTIGACIÓN.html?hl=es&id=5A2QDwAAQBAJ
- Huang, X., Schmelter, F., Uhlig, A., Irshad, M. T., Nisar, M. A., Piet, A., Jablonski, L., Witt, O., Schröder, T., Sina, C., & Grzegorzec, M. (2024). Comparison of feature learning methods for non-invasive interstitial glucose prediction using wearable sensors in healthy cohorts: a pilot study. *Intelligent Medicine*, 4(4), 226–238. <https://doi.org/10.1016/j.imed.2024.05.002>
- Huang, Z., Herbozo Contreras, L. F., Leung, W. H., Yu, L., Truong, N. D., Nikpour, A., & Kavehei, O. (2024). Efficient Edge-AI Models for Robust ECG Abnormality Detection on Resource-Constrained Hardware. *Journal of Cardiovascular Translational Research*, 17(4), 879–892. <https://doi.org/10.1007/s12265-024-10504-y>
- Huber, S., Wiemer, H., Schneider, D., & Ihlenfeldt, S. (2019). DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model. *Procedia CIRP*, 79, 403–408. <https://doi.org/10.1016/J.PROCIR.2019.02.106>
- Huespe, I., Carboni Bisso, I., Di Stefano, S., Terrasa, S., Gemelli, N. A., & Las Heras,

- M. (2022a). COVID-19 Severity Index: A predictive score for hospitalized patients. *Medicina Intensiva*, 46(2), 98–101. <https://doi.org/https://doi.org/10.1016/j.medin.2020.12.001>
- Huespe, I., Carboni Bisso, I., Di Stefano, S., Terrasa, S., Gemelli, N. A., & Las Heras, M. (2022b). COVID-19 Severity Index: A predictive score for hospitalized patients. *Medicina Intensiva*, 46(2), 98–101. <https://doi.org/10.1016/J.MEDIN.2020.12.001>
- Huifeng, W., Kadry, S. N., & Raj, E. D. (2020). Continuous health monitoring of sportsperson using IoT devices based wearable technology. *Computer Communications*, 160, 588–595. <https://doi.org/10.1016/J.COMCOM.2020.04.025>
- Hussain, K. (2021). *Design For A Cause 2021*. Element14. <https://community.element14.com/challenges-projects/design-challenges/design-for-a-cause-2021/b/blog/posts/vital-care---07-final-summary>
- Ianculescu, M., Constantin, V.-Ștefan, Gușatu, A.-M., Petrache, M.-C., Mihăescu, A.-G., Bica, O., & Alexandru, A. (2025). Enhancing Connected Health Ecosystems Through IoT-Enabled Monitoring Technologies: A Case Study of the Monit4Healthy System. *Sensors*, 25(7), 2292. <https://doi.org/10.3390/s25072292>
- Iodice, G. M. (2022). *TinyML Cookbook*. Packt Publishing Limited. <https://www.packtpub.com/product/tinyml-cookbook/9781801814973>
- Iodice, G. M. (2023). *TinyML cookbook: combine machine learning with microcontrollers to solve real-world problems*. Packt Publishing Ltd.
- Iosa, M., Picerno, P., Paolucci, S., & Morone, G. (2016). Wearable inertial sensors for human movement analysis. *Expert Review of Medical Devices*, 13(7), 641–659. <https://doi.org/10.1080/17434440.2016.1198694>
- Iqbal, A., Ullah, F., Anwar, H., Ur Rehman, A., Shah, K., Baig, A., Ali, S., Yoo, S., & Kwak, K. S. (2020). Wearable Internet-of-Things platform for human activity recognition and health care: *International Journal of Distributed Sensor Networks*, 16(6). <https://doi.org/10.1177/1550147720911561>
- Iqbal, T., Simpkin, A. J., Roshan, D., Glynn, N., Killilea, J., Walsh, J., Molloy, G., Ganly, S., Ryman, H., Coen, E., Elahi, A., Wijns, W., & Shahzad, A. (2022). Stress Monitoring Using Wearable Sensors: A Pilot Study and Stress-Predict Dataset. *Sensors*, 22(21). <https://doi.org/10.3390/S22218135>
- Itu-t. (2019). *ITU-T Rec. Y.4051 (07/2019) Vocabulary for smart cities and communities*. <http://handle.itu.int/11.1002/1000/11>
- Jacob, V., Reynolds, J. A., Chattopadhyay, S. K., Hopkins, D. P., Therrien, N. L., Jones, C. D., Durthaler, J. M., Rask, K. J., Cuellar, A. E., Clymer, J. M., & Kottke, T. E. (2022). Pharmacist Interventions for Medication Adherence: Community Guide Economic Reviews for Cardiovascular Disease. *American Journal of Preventive Medicine*, 62(3), e202–e222. <https://doi.org/10.1016/J.AMEPRE.2021.08.021>
- Jahan, M. S., Mansourvar, M., Puthusserypady, S., Wiil, U. K., & Peimankar, A. (2022). Short-term atrial fibrillation detection using electrocardiograms: A comparison of machine learning approaches. *International Journal of Medical Informatics*, 163. <https://doi.org/10.1016/J.IJMEDINF.2022.104790>

- Javed, A. R., Sarwar, M. U., Beg, M. O., Asim, M., Baker, T., & Tawfik, H. (2020). A collaborative healthcare framework for shared healthcare plan with ambient intelligence. *Human-Centric Computing and Information Sciences*, 10(1), 1–21. <https://doi.org/10.1186/S13673-020-00245-7/TABLES/4>
- Jeng, P.-Y., Wang, L.-C., Hu, C.-J., Chaur-Jong, C.-J., Wu, D., Jeng, P.-Y. ;, & Wang, L.-C. ; (2021). A Wrist Sensor Sleep Posture Monitoring System: An Automatic Labeling Approach. *Sensors* 2021, Vol. 21, Page 258, 21(1), 258. <https://doi.org/10.3390/S21010258>
- Jeong, J. W., Lee, W., & Kim, Y. J. (2022). A real-time wearable physiological monitoring system for home-based healthcare applications. *Sensors*, 22(1). <https://doi.org/10.3390/S22010104>
- Jin, E., Wang, Q., Wang, X., Li, Y., & Shao, Z. (2025). AN IOT-ENABLED MULTIMODAL FRAMEWORK FOR REAL-TIME SLEEP QUALITY ASSESSMENT AND HEALTH MONITORING. *Journal of Mechanics in Medicine and Biology*, 25(8), 10. <https://doi.org/10.1142/S0219519425400743/ASSET/IMAGES/LARGE/S0219519425400743FIGF2.JPEG>
- Jordan, A. A., Pegatoquet, A., Castagnetti, A., Raybaut, J., & Le Coz, P. (2020). Deep Learning for Eye Blink Detection Implemented at the Edge. *IEEE Embedded Systems Letters*. <https://doi.org/10.1109/LES.2020.3029313>
- Joshi, A. V. (2020). Machine Learning and Artificial Intelligence. In *Machine Learning and Artificial Intelligence*. Springer International Publishing. <https://doi.org/10.1007/978-3-030-26622-6>
- Kalantarian, H., Sideris, C., & Sarrafzadeh, M. (2017). A Hierarchical Classification and Segmentation Scheme for Processing Sensor Data. *IEEE Journal of Biomedical and Health Informatics*, 21(3), 672–681. <https://doi.org/10.1109/JBHI.2016.2526679>
- Kamalakannan, M., Venkataramana, P., Shalini, V., Firdous, S., Reddy, N., & Iyyanar, I. (2025). Optimization and Non-Destructive Examination of High-Speed Computational Technique for Talent Identification in Sports. *2025 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 1–6. <https://doi.org/10.1109/i-PACT65952.2025.11307993>
- Karan, A., & Kaygun, A. (2021). Time series classification via topological data analysis. *Expert Systems with Applications*, 183, 115326. <https://doi.org/10.1016/J.ESWA.2021.115326>
- Kashyap, M., Sharma, V., & Gupta, N. (2018). Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. *Procedia Computer Science*, 132, 1611–1618. <https://doi.org/10.1016/J.PROCS.2018.05.126>
- Khan, M. A., Chen, M., Nawaz, T., Sedky, M., Sheikh, M., Bashir, A. K., & Hassan, S. (2025). Smart Steering Wheel: Design of IoMT-Based Non-Invasive Driver Health Monitoring System to Enhance Road Safety. *IET Intelligent Transport Systems*, 19(1), e70012. <https://doi.org/10.1049/itr2.70012>
- Khan, M., Anjum, S., Ibrahim, A., Nnaji, C., Aryal, A., & Koh, A. S. (2025). Wearable Sensor-Based Fatigue Classification Under Diverse Thermal Conditions. *Journal of Information Technology in Construction*. *Journal of Information Technology in Construction*, 30(1), 875–902. <https://doi.org/10.36680/j.itcon.2025.036>

- Kim, S., Fayazi, M., Daftardar, A., Chen, K. Y., Tan, J., Pal, S., Ajayi, T., Xiong, Y., Mudge, T., Chakrabarti, C., Blaauw, D., Dreslinski, R., & Kim, H. S. (2021). Versa: A Dataflow-Centric Multiprocessor with 36 Systolic ARM Cortex-M4F Cores and a Reconfigurable Crossbar-Memory Hierarchy in 28nm. *IEEE Symposium on VLSI Circuits, Digest of Technical Papers, 2021-June*. <https://doi.org/10.23919/VLSICIRCUITS52068.2021.9492391>
- Kitchenham, B. (2007). Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*.
- Kokhazad, Z., Gkountelos, D., Kokhazadeh, M., Bournas, C., Keramidas, G., & Kelefouras, V. (2025). Low-Memory-Footprint CNN-Based Biomedical Signal Processing for Wearable Devices. *IoT*, 6(2), 29. <https://doi.org/10.3390/iot6020029>
- Kothari, C. (2004). Research methodology: methods and techniques. In *New Age International*. <https://doi.org/http://196.29.172.66:8080/jspui/bitstream/123456789/2574/1/Research%20Methodology.pdf>
- Koufos, K., El Haloui, K., Dianati, M., Higgins, M., Elmirghani, J., Imran, M. A., & Tafazolli, R. (2021). Trends in Intelligent Communication Systems: Review of Standards, Major Research Projects, and Identification of Research Gaps. *Journal of Sensor and Actuator Networks*, 10(4), 60. <https://doi.org/10.3390/jsan10040060>
- Kuchuk, H., & Malokhvii, E. (2024). INTEGRATION OF IOT WITH CLOUD, FOG, AND EDGE COMPUTING: A REVIEW. *Advanced Information Systems*, 8(2), 65–78. <https://doi.org/10.20998/2522-9052.2024.2.08>
- Kumar, M., Kumar, A., Verma, S., Bhattacharya, P., Ghimire, D., Kim, S., & Hosen, A. S. M. S. (2023). Healthcare Internet of Things (H-IoT): Current Trends, Future Prospects, Applications, Challenges, and Security Issues. *Electronics*, 12(9), 2050. <https://doi.org/10.3390/electronics12092050>
- Kumar, P. M., & Devi Gandhi, U. (2018). A novel three-tier Internet of Things architecture with machine learning algorithm for early detection of heart diseases. *Computers and Electrical Engineering*, 65, 222–235. <https://doi.org/10.1016/J.COMPELECENG.2017.09.001>
- Lahoti, S. (2024). Genome Based Computational Technique to Identify the Functional RNA in Protozoan Parasites. *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, 1035–1039. <https://doi.org/10.1109/IC2PCT60090.2024.10486718>
- Lea, P., & Safari, an O. M. C. (2020). *IoT and Edge Computing for Architects - Second Edition*.
- Lee, E., & Lee, C. Y. (2021). PPG-Based Smart Wearable Device with Energy-Efficient Computing for Mobile Health-Care Applications. *IEEE Sensors Journal*, 21(12), 13564–13573. <https://doi.org/10.1109/JSEN.2021.3069460>
- Li, W., Chu, H., Huang, B., Huan, Y., Zheng, L., & Zou, Z. (2021). Enabling on-device classification of ECG with compressed learning for health IoT. *Microelectronics Journal*, 115, 105188. <https://doi.org/10.1016/J.MEJO.2021.105188>

- Liao, X., Wang, B., & Kang, Y. (2020a). Novel coronavirus infection during the 2019–2020 epidemic: preparing intensive care units—the experience in Sichuan Province, China. *Intensive Care Medicine*, 46(2), 357–360. <https://doi.org/https://doi.org/10.1007/s00134-020-05954-2>
- Liao, X., Wang, B., & Kang, Y. (2020b). Novel coronavirus infection during the 2019–2020 epidemic: preparing intensive care units—the experience in Sichuan Province, China. *Intensive Care Medicine*, 46(2), 357–360. <https://doi.org/10.1007/S00134-020-05954-2>
- Lim, W. Y. B., Garg, S., Xiong, Z., Niyato, D., Leung, C., Miao, C., & Guizani, M. (2021). Dynamic Contract Design for Federated Learning in Smart Healthcare Applications. *IEEE Internet of Things Journal*, 8(23), 16853–16862. <https://doi.org/10.1109/IIOT.2020.3033806>
- Lindsey, H. (2022). *What Are Vital Signs, and Why Are They Important?* Healthline Media LLC. <https://www.healthline.com/health/what-are-vital-signs>
- Liquet, B., Moka, S., & Nazarathy, Y. (2024). Principles of Machine Learning. In *Mathematical Engineering of Deep Learning*. <https://doi.org/10.1201/9781003298687-2>
- Liu, M., Shao, N., Zheng, C., & Wang, J. (2021). Real Time Arrhythmia Monitoring and Classification Based on Edge Computing and DNN. *Wireless Communications and Mobile Computing*, 2021. <https://doi.org/10.1155/2021/5563338>
- Liu, T., & Mitcham, C. (2024). Toward Practical Hermeneutics of Fourth Paradigm AI for Science. *Technology and Language*, 5(1), 89–105. <https://doi.org/10.48417/TECHNOLANG.2024.01.07>
- Llanes, O., Carlos, S., Corona, C., Jos, S. N., & Verdegay, L. (2020). *Computational Intelligence in Emerging Technologies for Engineering Applications* (O. Llanes Santiago, C. Cruz Corona, A. J. Silva Neto, & J. L. Verdegay (eds.); Vol. 872). Springer International Publishing. <https://doi.org/10.1007/978-3-030-34409-2>
- Louis Uzoegwu Farah, C. (2023). Comparative Analysis for Predicting Cardiovascular Diseases Using Machine Learning and Deep Learning Approaches. *International Journal of Science and Research (IJSR)*, 12(8), 945–964. <https://doi.org/10.21275/sr23809044938>
- Luque Martínez, V., & Bermejo Sánchez, S. (2020). *Smart low-cost motion controllers* [Universitat Politècnica de Catalunya]. <https://upcommons.upc.edu/handle/2117/334668>
- Lutz, M. (2025). *Learning Python* (6th ed.). O'Reilly Media.
- Luzzani, G., Pogliano, M., Buraioli, I., Colavincenzo, M., Martorana, S., Guglieri, G., & Demarchi, D. (2025). ECG, Respiration, fNIRS, and Eye Tracking for Stress and Mental Workload Monitoring in Human-Machine Interaction. *IEEE Access*, 13(1), 122726–122741. <https://doi.org/10.1109/ACCESS.2025.3588384>
- Maier, A., Sharp, A., & Vagapov, Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. *2017 Internet Technologies and Applications, ITA 2017 - Proceedings of the 7th International Conference*, 143–148. <https://doi.org/10.1109/ITECHA.2017.8101926>

- Majumdar, P. (2023). *Solutions, Mastering Classification Algorithms for Machine Learning: Learn how to apply Classification algorithms for effective Machine Learning*. bpbonline.
- Malche, T., Tharewal, S., Tiwari, P. K., Jabarulla, M. Y., Alnuaim, A. A., Hatamleh, W. A., & Ullah, M. A. (2022). Artificial Intelligence of Things- (AIoT-) Based Patient Activity Tracking System for Remote Patient Monitoring. *Journal of Healthcare Engineering*, 2022. <https://doi.org/10.1155/2022/8732213>
- Mallqui, J. A., Martinez, L. P., & Aguirre, J. A. (2019). Technological architecture based on internet of things to monitor the journeys of artisanal fishing. *Advances in Intelligent Systems and Computing*, 931, 24–33. https://doi.org/10.1007/978-3-030-16184-2_3
- Mangiameli, G., Bottoni, E., Tagliabue, A., Giudici, V. M., Crepaldi, A., Testori, A., Voulaz, E., Cariboni, U., Re Cecconi, E., Luppichini, M., Alloisio, M., Brascia, D., Morengi, E., & Marulli, G. (2024). Early Hospital Discharge on Day Two Post-Robotic Lobectomy with Telehealth Home Monitoring. *Journal of Clinical Medicine*, 13(20), 6268. <https://doi.org/10.3390/jcm13206268>
- Mansouri, Y., & Babar, M. A. (2021). A review of edge computing: Features and resource virtualization. *Journal of Parallel and Distributed Computing*, 150, 155–183. <https://doi.org/10.1016/J.JPDC.2020.12.015>
- Marquez-Sanchez, S., Campero-Jurado, I., Robles-Camarillo, D., Rodriguez, S., & Corchado-Rodriguez, J. M. (2021). BeSafe B2.0 Smart Multisensory Platform for Safety in Workplaces. *SENSORS*, 21(10). <https://doi.org/10.3390/s21103372>
- Martínez, E. (1998). *Estudio de la Integración de los Meios Informáticos en los currículos de Educación Infantil y Primaria: sus Implicaciones en la Practica Educativa* [Universidad Complutense de Madrid]. <https://dialnet.unirioja.es/servlet/tesis?codigo=10166&info=resumen&idioma=SPA>
- Mastoi, Q. ul ain, Wah, T. Y., & Raj, R. G. (2019). Reservoir Computing Based Echo State Networks for Ventricular Heart Beat Classification. *Applied Sciences* 2019, Vol. 9, Page 702, 9(4), 702. <https://doi.org/10.3390/APP9040702>
- Mauldin, T. R., Canby, M. E., Metsis, V., Ngu, A. H. H., & Rivera, C. C. (2018). Smartfall: A smartwatch-based fall detection system using deep learning. *Sensors (Switzerland)*, 18(10). <https://doi.org/10.3390/S18103363>
- McMurray, J. P., DeVries, A., Frazee, K., Sizemore, B., Branan, K. L., Jennings, R., & Coté, G. L. (2025). A Novel Wearable Device for Continuous Blood Pressure Monitoring Utilizing Strain Gauge Technology. *Biosensors*, 15(7), 413. <https://doi.org/10.3390/bios15070413>
- Mehmood, M. Y., Oad, A., Abrar, M., Munir, H. M., Hasan, S. F., Muqet, H. A. U., & Golilarz, N. A. (2021). Edge Computing for IoT-Enabled Smart Grid. *Security and Communication Networks*, 2021. <https://doi.org/10.1155/2021/5524025>
- Memon, S. F., Memon, M., & Bhatti, S. (2020). Wearable technology for infant health monitoring: A survey. *IET Circuits, Devices and Systems*, 14(2), 115–129. <https://doi.org/10.1049/iet-cds.2018.5447>
- Meza, R. (2013). *ECG con diagnostico por sistema experto y comunicación GPRS*.
- Mishra, A., Cohen, A., Reichherzer, T., & Wilde, N. (2021). Detection of data

- anomalies at the edge of pervasive IoT systems. *Computing*, 103(8), 1657–1675. <https://doi.org/10.1007/s00607-021-00927-9>
- Mocha Zhingri, E. M., & Paltín Guamanrrigra, L. F. (2022). *Análisis comparativo de las plataformas Amazon, Google, Azure, para internet de las cosas “IoT” usando Node-RED como middleware* [Universidad Señor de Sip+an]. <http://dspace.ups.edu.ec/handle/123456789/24045>
- Morán, A., Canals, V., Galan-Prado, F., Frasser, C. F., Radhakrishnan, D., Safavi, S., & Rosselló, J. L. (2021). Hardware-Optimized Reservoir Computing System for Edge Intelligence Applications. *Cognitive Computation* 2021, 1–9. <https://doi.org/10.1007/S12559-020-09798-2>
- Mortazavi, B., Nemati, E., Wall, K. Vander, Flores-Rodriguez, H. G., Cai, J. Y. J., Lucier, J., Naeim, A., & Sarrafzadeh, M. (2015). Can smartwatches replace smartphones for posture tracking? *Sensors (Switzerland)*, 15(10), 26783–26800. <https://doi.org/10.3390/s151026783>
- Mrozek, D., Koczur, A., & Malysiak-Mrozek, B. (2020). Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge. *Information Sciences*, 537, 132–147. <https://doi.org/10.1016/j.ins.2020.05.070>
- Muhammad, G., Alhamid, M. F., & Long, X. (2019). Computing and processing on the edge: Smart pathology detection for connected healthcare. *IEEE Network*, 33(6), 44–49. <https://doi.org/10.1109/MNET.001.1900045>
- Mulfari, D., Palla, A., & Fanucci, L. (2017). Embedded Systems and TensorFlow Frameworks as Assistive Technology Solutions. *Studies in Health Technology and Informatics*, 242, 396–400. <https://doi.org/10.3233/978-1-61499-798-6-396>
- Musci, M., De Martini, D., Blago, N., Facchinetti, T., & Piastra, M. (2021). Online Fall Detection Using Recurrent Neural Networks on Smart Wearable Devices. *IEEE Transactions on Emerging Topics in Computing*, 9(3), 1276–1289. <https://doi.org/10.1109/TETC.2020.3027454>
- Namazi, A., Modiri, E., Blesić, S., Knežević, O. M., & Mirkov, D. M. (2025). Comparative Analysis of Machine Learning Techniques for Heart Rate Prediction Employing Wearable Sensor Data. *Sports*, 13(3), 87. <https://doi.org/10.3390/sports13030087>
- Nath, R. K., Thapliyal, H., & Caban-Holt, A. (2022). Machine Learning Based Stress Monitoring in Older Adults Using Wearable Sensors and Cortisol as Stress Biomarker. *Journal of Signal Processing Systems*, 94(6), 513–525. <https://doi.org/10.1007/S11265-020-01611-5>
- Navarro, R. (2014). *Sistemas de Acondicionamiento y Adquisición de Señales Bioeléctricas*. <https://silo.tips/download/universidad-de-alcala-departamento-de-electronica>
- Nazari, M., Moayed Rezaie, S., Yaseri, F., Sadr, H., & Nazari, E. (2024). Design and analysis of a telemonitoring system for high-risk pregnant women in need of special care or attention. *BMC Pregnancy and Childbirth*, 24(1), 817. <https://doi.org/10.1186/s12884-024-07019-4>
- Nguyen, T., Park, S., Sodager, A., Park, J., Gallo, D. M., Luo, G., Romero, R., & Gandjbakhche, A. (2024). A Wireless and Wearable Multimodal Sensor to Non-Invasively Monitor Transabdominal Placental Oxygen Saturation and Maternal

- Physiological Signals. *Biosensors*, 14(10), 481. <https://doi.org/10.3390/BIOS14100481/S1>
- Ninu, S. B. (2023). An intrusion detection system using Exponential Henry Gas Solubility Optimization based Deep Neuro Fuzzy Network in MANET. *Engineering Applications of Artificial Intelligence*, 123, 105969. <https://doi.org/10.1016/j.engappai.2023.105969>
- Noto, T., Zhou, G., Schuele, S., Templer, J., & Zelano, C. (2018). Automated analysis of breathing waveforms using BreathMetrics: a respiratory signal processing toolbox. *Chemical Senses*, 43(8), 583–597. <https://doi.org/10.1093/CHEMSE/BJY045>
- Novac, P. E., Castagnetti, A., Russo, A., Miramond, B., Pegatoquet, A., Verdier, F., & Castagnetti, A. (2020). Toward unsupervised Human Activity Recognition on Microcontroller Units. *Proceedings - Euromicro Conference on Digital System Design, DSD 2020*, 542–550. <https://doi.org/10.1109/DSD51259.2020.00090>
- Offarm. (2011). Curso de metodología básica de investigación en AF. Tema 8. *Offarm*, 30(6), 164–167. <https://www.elsevier.es/es-revista-offarm-4-articulo-curso-metodologia-basica-investigacion-af--13081763>
- Oh, K., Park, H., Lee, G. H., & Choi, J. K. (2026). Supervised contrastive learning-based stress detection for wearable sensor-based healthcare applications. *Future Generation Computer Systems*, 175(1), 108058. <https://doi.org/10.1016/j.future.2025.108058>
- OMS, & OPS. (2020). Aspectos técnicos y regulatorios sobre el uso de oxímetros de pulso en el monitoreo de pacientes con COVID-19. *Organización Mundial de La Salud*, 1–20. <https://iris.paho.org/handle/10665.2/52551>
- Otoun, S., Al Ridhawi, I., & Mouftah, H. T. (2021). Preventing and Controlling Epidemics through Blockchain-Assisted AI-Enabled Networks. *IEEE Network*, 35(3), 34–41. <https://doi.org/10.1109/MNET.011.2000628>
- Ozan, V. (2021). *Developing IoT projects with ESP32 automate your home or business with inexpensive wi-fi devices*.
- Pace, P., Aloï, G., Gravina, R., Caliciuri, G., Fortino, G., & Liotta, A. (2019). An Edge-Based Architecture to Support Efficient Applications for Healthcare Industry 4.0. *IEEE Transactions on Industrial Informatics*, 15(1), 481–489. <https://doi.org/10.1109/TII.2018.2843169>
- Palella, S., & Martins, F. (2012). *Metodología de la investigación*. <https://metodologiaecs.files.wordpress.com/2015/09/metodologc3ada-de-la-investigac3b3n-cuantitativa-3ra-ed-2012-santa-palella-stracuzzi-feliberto-martins-pestana.pdf>
- Pallets Project. (2021). *Jinja — Jinja Documentation (3.0.x)*. <https://jinja.palletsprojects.com/en/3.0.x/>
- Pan, J., & Tompkins, W. J. (1985). A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3), 230–236. <https://doi.org/10.1109/TBME.1985.325532>
- Pan, Y. (2025). Emotion Recognition Algorithm Based on Multi-Modal Physiological Signal Feature Fusion Using Artificial Intelligence and Deep Learning. *International Journal of Advanced Computer Science and Applications*, 16(6),

- 105–118. <https://doi.org/10.14569/IJACSA.2025.0160613>
- Park, S., Youm, M., & Kim, J. (2025). IMU Sensor-Based Worker Behavior Recognition and Construction of a Cyber-Physical System Environment. *Sensors*, 25(2), 442. <https://doi.org/10.3390/s25020442>
- Patel, S. M., & Kanawade, S. Y. (2017). Internet of Things based smart home with intel edison. *Advances in Intelligent Systems and Computing*, 508, 385–392. https://doi.org/10.1007/978-981-10-2750-5_40/COVER
- Patel, V., Orchanian-Cheff, A., & Wu, R. (2021a). Evaluating the validity and utility of wearable technology for continuously monitoring patients in a hospital setting: Systematic review. In *JMIR mHealth and uHealth* (Vol. 9, Issue 8, p. e17411). JMIR mHealth and uHealth. <https://doi.org/10.2196/17411>
- Patel, V., Orchanian-Cheff, A., & Wu, R. (2021b). Evaluating the validity and utility of wearable technology for continuously monitoring patients in a hospital setting: Systematic review. *JMIR MHealth and UHealth*, 9(8). <https://doi.org/10.2196/17411>
- Pathinarupothi, R. K., Durga, P., & Rangan, E. S. (2019). IoT-based smart edge for global health: Remote monitoring with severity detection and alerts transmission. *IEEE Internet of Things Journal*, 6(2), 2449–2462. <https://doi.org/10.1109/IIOT.2018.2870068>
- Paulovich, F. V., De Oliveira, M. C. F., & Oliveira, O. N. (2018). A Future with Ubiquitous Sensing and Intelligent Systems. *ACS Sensors*, 3(8), 1433–1438. <https://doi.org/10.1021/acssensors.8b00276>
- Pernice, S. A. (2023). El problema de la reducción dimensional. Análisis de Componentes Principales (PCA). *Revista Mutis*, 14(1), 1–21. <https://doi.org/10.21789/22561498.2057>
- Piątkowski, D., & Walkowiak, K. (2022). TinyML-Based Concept System Used to Analyze Whether the Face Mask Is Worn Properly in Battery-Operated Conditions. *Applied Sciences (Switzerland)*, 12(1). <https://doi.org/10.3390/APP12010484>
- Pote, S. (2023). *MACHINE LEARNING IN PRODUCTION*. BPB PUBLICATIONS. <https://www.perlego.com/book/4087368/machine-learning-in-production-master-the-art-of-delivering-robust-machine-learning-solutions-with-mlops-english-edition-pdf>
- Pramanik, P. K. D., Pal, S., & Choudhury, P. (2018). Beyond automation: The cognitive IoT. artificial intelligence brings sense to the internet of things. In *Lecture Notes on Data Engineering and Communications Technologies* (Vol. 14, pp. 1–37). Springer, Cham. https://doi.org/10.1007/978-3-319-70688-7_1
- Premalatha, G., & Bai, V. T. (2022). Wireless IoT and Cyber-Physical System for Health Monitoring Using Honey Badger Optimized Least-Squares Support-Vector Machine. *Wireless Personal Communications*. <https://doi.org/10.1007/S11277-022-09500-9>
- Purbasari, A., Rinawan, F. R., Zulianto, A., Susanti, A. I., & Komara, H. (2021). CRISP-DM for Data Quality Improvement to Support Machine Learning of Stunting Prediction in Infants and Toddlers. *Proceedings - 2021 8th International Conference on Advanced Informatics: Concepts, Theory, and Application*,

- ICAICTA 2021. <https://doi.org/10.1109/ICAICTA53211.2021.9640294>
- Qadri, Y. A., Nauman, A., Zikria, Y. Bin, Vasilakos, A. V., & Kim, S. W. (2020). The Future of Healthcare Internet of Things: A Survey of Emerging Technologies. *IEEE Communications Surveys & Tutorials*, 22(2), 1121–1167. <https://doi.org/10.1109/COMST.2020.2973314>
- Rachakonda, L., Mohanty, S. P., & Kougianos, E. (2020). iLog: An Intelligent Device for Automatic Food Intake Monitoring and Stress Detection in the IoMT. *IEEE Transactions on Consumer Electronics*, 66(2), 115–124. <https://doi.org/10.1109/TCE.2020.2976006>
- Rahman, M. J., Morshed, B. I., Harmon, B., & Rahman, M. (2022). A pilot study towards a smart-health framework to collect and analyze biomarkers with low-cost and flexible wearables. *Smart Health*, 23, 100249. <https://doi.org/10.1016/J.SMHL.2021.100249>
- Rahmani, M. H., Berkvens, R., & Weyn, M. (2021). Chest-worn inertial sensors: A survey of applications and methods. *Sensors*, 21(8). <https://doi.org/10.3390/s21082875>
- Rajagopal, D., & Subramanian, P. K. T. (2025). AI augmented edge and fog computing for Internet of Health Things (IoHT). *PeerJ Computer Science*, 11(1), e2431. <https://doi.org/10.7717/peerj-cs.2431>
- Ramesh Saha, Biswas, S., Sarmah, S., Karmakar, S., & Das, P. (2021). A Working Prototype Using DS18B20 Temperature Sensor and Arduino for Health Monitoring. *SN Computer Science*, 2(1), 1–21. <https://doi.org/10.1007/S42979-020-00434-2/FIGURES/26>
- Rath, A. S., Devaurs, D., & Lindstaedt, S. N. (2009). UICO: An ontology-based user interaction context model for automatic task detection on the computer desktop. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/1552262.1552270>
- Ravi, D., Wong, C., Lo, B., & Yang, G. Z. (2017). A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices. *IEEE Journal of Biomedical and Health Informatics*, 21(1), 56–64. <https://doi.org/10.1109/JBHI.2016.2633287>
- Reguig, M. A. B., & Labdelli, N. (2024). Blood glucose prediction using non-invasive optical system based on photoplethysmography. *International Journal of Electrical and Computer Engineering (IJECE)*, 14(5), 5200. <https://doi.org/10.11591/ijece.v14i5.pp5200-5208>
- Resende, C., Folgado, D., Oliveira, J., Franco, B., Moreira, W., Oliveira-Jr, A., Cavaleiro, A., & Carvalho, R. (2021a). Tip4.0: Industrial internet of things platform for predictive maintenance. *Sensors*, 21(14), 4676. <https://doi.org/10.3390/s21144676>
- Resende, C., Folgado, D., Oliveira, J., Franco, B., Moreira, W., Oliveira-Jr, A., Cavaleiro, A., & Carvalho, R. (2021b). Tip4.0: Industrial internet of things platform for predictive maintenance. *Sensors*, 21(14). <https://doi.org/10.3390/s21144676>
- Revision C. (2014). *REF: BBONEBLK_SRM BeagleBone Black System Reference Manual BeagleBone Black System Reference Manual*.

http://circuitco.com/support/index.php?title=BeagleBoneBlack#LATEST_PRO
DUC

- Rim, B., Sung, N. J., Min, S., & Hong, M. (2020). Deep learning in physiological signal data: A survey. *Sensors (Switzerland)*, 20(4). <https://doi.org/10.3390/S20040969>
- Rivera, J. G., Ramiro, B., & Lopera Sánchez, A. (2024). *Internet de las Cosas* (1st ed.). Red Descartes. <https://proyectodescartes.org/iCartesiLibri/PDF/IoT.pdf>
- Rodríguez Rodríguez, C., Breña Oré, J. L., & Esenarro Vargas, D. (2021). Las variables en la metodología de la investigación científica. In *Las variables en la metodología de la investigación científica* (1st ed.). ÁREA DE INNOVACIÓN Y DESARROLLO, S.L. <https://doi.org/10.17993/ingytec.2021.78>
- Rong, M. Z. M. E. P. M. · C. (2019). *Cloud Computing and Big Data: Technologies, Applications and Security* (M. Zbakh, M. Essaïdi, P. Manneback, & C. Rong (eds.); Vol. 49, Issue September 2018). Springer International Publishing. <https://doi.org/10.1007/978-3-319-97719-5>
- Rosero-Montalvo, P. D., Peluffo-Ordonez, D. H., Lopez Batista, V. F., Serrano, J., & Rosero, E. A. (2019). Intelligent system for identification of wheelchair user's posture using machine learning techniques. *IEEE Sensors Journal*, 19(5), 1936–1942. <https://doi.org/10.1109/JSEN.2018.2885323>
- Roy, A., & Chakraborty, S. (2023). Support vector machine in structural reliability analysis: A review. *Reliability Engineering & System Safety*, 233(1), 109126. <https://doi.org/10.1016/j.ress.2023.109126>
- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson.
- Said, A. M., Yahyaoui, A., & Abdellatif, T. (2021). Efficient Anomaly Detection for Smart Hospital IoT Systems. *Sensors 2021, Vol. 21, Page 1026*, 21(4), 1026. <https://doi.org/10.3390/S21041026>
- Salerno, S. (2021). *GitHub - eloquentarduino/tinymlgcn: Generate C code for microcontrollers from Tensorflow models*. <https://github.com/eloquentarduino/tinymlgcn>
- Saltzman, W. M. (2010). Biomedical engineering: bridging medicine and technology. *Choice Reviews Online*, 47(06), 47-3188-47-3188. <https://doi.org/10.5860/choice.47-3188>
- Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits and Systems Magazine*, 20(3), 4–18. <https://doi.org/10.1109/MCAS.2020.3005467>
- Santos, G. L., Endo, P. T., Monteiro, K. H. de C., Rocha, E. da S., Silva, I., & Lynn, T. (2019). Accelerometer-Based Human Fall Detection Using Convolutional Neural Networks. *Sensors 2019, Vol. 19, Page 1644*, 19(7), 1644. <https://doi.org/10.3390/S19071644>
- Santos, R., Ribeiro, B., Sousa, I., Santos, J., Guede-Fernández, F., Dias, P., Carreiro, A. V., Gamboa, H., Coelho, P., Fragata, J., & Londral, A. (2024). Predicting post-discharge complications in cardiothoracic surgery: A clinical decision support system to optimize remote patient monitoring resources. *International Journal of Medical Informatics*, 182(1), 105307. <https://doi.org/10.1016/j.ijmedinf.2023.105307>

- Sapra, A., Malik, A., & Bhandari, P. (2023). Vital Sign Assessment. In *StatPearls*. StatPearls Publishing. <https://www.ncbi.nlm.nih.gov/books/NBK553213/>
- Sarabia-Jácome, D., Usach, R., Palau, C. E., & Esteve, M. (2020). Highly-efficient fog-based deep learning AAL fall detection system. *Internet of Things*, 11, 100185. <https://doi.org/10.1016/J.IOT.2020.100185>
- SARI, U., & KIRINDI, T. (2019). Using Arduino in Physics Teaching: Arduino-based Physics Experiment to Study Temperature Dependence of Electrical Resistance. *Journal of Computer and Education Research*, 7(14), 698–710. <https://doi.org/10.18009/JCER.579362>
- Satija, U., Ramkumar, B., & Manikandan, S. M. (2017). Real-Time Signal Quality-Aware ECG Telemetry System for IoT-Based Health Care Monitoring. *IEEE Internet of Things Journal*, 4(3), 815–823. <https://doi.org/10.1109/JIOT.2017.2670022>
- Satija, U., Ramkumar, B., & Sabarimalai Manikandan, M. (2018). A Review of Signal Processing Techniques for Electrocardiogram Signal Quality Assessment. *IEEE Reviews in Biomedical Engineering*, 11, 36–52. <https://doi.org/10.1109/RBME.2018.2810957>
- Scrugli, M. A., Loi, D., Raffo, L., & Meloni, P. (2022). An Adaptive Cognitive Sensor Node for ECG Monitoring in the Internet of Medical Things. *IEEE Access*, 10, 1688–1705. <https://doi.org/10.1109/ACCESS.2021.3136793>
- Shiao, Y., Hoang, T., & Chang, P. Y. (2021). Real-time exercise mode identification with an inertial measurement unit for smart dumbbells. *Applied Sciences (Switzerland)*, 11(23). <https://doi.org/10.3390/APP112311521>
- Singh, A., & Chatterjee, K. (2021). Securing smart healthcare system with edge computing. *Computers & Security*, 108, 102353. <https://doi.org/10.1016/J.COSE.2021.102353>
- Singh, S., Koirala, B., Thami, R., Thapa, A., Thapa, B., Kayastha, A., & Dahal, P. (2021). Length of stay in the emergency department during COVID-19 pandemic in a tertiary care hospital: A descriptive cross-sectional study. *Journal of the Nepal Medical Association*, 59(237), 490–493. <https://doi.org/10.31729/JNMA.6281>
- ŞİŞİK, F., & SERT, E. (2020). Brain tumor segmentation approach based on the extreme learning machine and significantly fast and robust fuzzy C-means clustering algorithms running on Raspberry Pi hardware. *Medical Hypotheses*, 136. <https://doi.org/10.1016/J.MEHY.2019.109507>
- Situnayake, D. (2020). *Machine Learning on Mobile and Edge Devices with TensorFlow Lite*. <https://www.infoq.com/presentations/tensorflow-lite>
- Situnayake, D., & Plunkett, J. (2023). *AI at the Edge*. O'Reilly Media.
- Skiena, S. (2017). Data science design. In *Springer*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-55444-0>
- Sreenilayam, S. P., Ahad, I. U., Nicolosi, V., Acinas Garzon, V., & Brabazon, D. (2020). Advanced materials of printed wearables for physiological parameter monitoring. *Materials Today*, 32, 147–177. <https://doi.org/10.1016/J.MATTOD.2019.08.005>

- Suleiman, H. J., Hamid, I. R. A., & Olaniran, O. R. (2025). Smart Health Monitoring for Predicting Heart Disease using IoT-Fog-Cloud Computing Model. *Engineering, Technology & Applied Science Research*, 15(3), 22565–22572. <https://doi.org/10.48084/etasr.10048>
- Suppiah, R., Noori, K., Abidi, K., & Sharma, A. (2024). Real-time edge computing design for physiological signal analysis and classification. *Biomedical Physics & Engineering Express*, 10(4), 045034. <https://doi.org/10.1088/2057-1976/ad4f8d>
- Supriya, M., & Chattu, V. K. (2021). A review of artificial intelligence, big data, and blockchain technology applications in medicine and global health. In *Big Data and Cognitive Computing* (Vol. 5, Issue 3, p. 41). Multidisciplinary Digital Publishing Institute. <https://doi.org/10.3390/bdcc5030041>
- Suresha, P. B., Hegde, C., Jiang, Z., & Clifford, G. D. (2022). An Edge Computing and Ambient Data Capture System for Clinical and Home Environments. *Sensors* 2022, Vol. 22, Page 2511, 22(7), 2511. <https://doi.org/10.3390/S22072511>
- Taye, M. M. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. *Computation*, 11(3), 52. <https://doi.org/10.3390/computation11030052>
- Thamaraiselvi, K., Rinesh, S., Ramaparthi, L., & Karthick, V. (2019). Internet of Things (IOT) based smart band to ensure the security for women. *Proceedings of the 2nd International Conference on Smart Systems and Inventive Technology, ICSSIT 2019*, 1093–1096. <https://doi.org/10.1109/ICSSIT46314.2019.8987928>
- Tian, Z., & Jia, Y. (2023). A method for portable PPG signal dynamic acquisition and denoise based on CEEMDAN and wavelet transform. *Journal of Instrumentation*, 18(01), P01023. <https://doi.org/10.1088/1748-0221/18/01/P01023>
- Tunggadewi, E. A., Agustin, E. I., & Yunardi, R. T. (2020). A smart wearable device based on internet of things for the safety of children in online transportation. *Indonesian Journal of Electrical Engineering and Computer Science*, 22(2), 100–108. <https://doi.org/10.11591/IJEECS.V22.I2.PP100-108>
- Uddin, M. Z. (2019). A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system. *Journal of Parallel and Distributed Computing*, 123, 46–53. <https://doi.org/10.1016/J.JPDC.2018.08.010>
- Ukil, A., Jara, A. J., & Marin, L. (2019). Data-Driven Automated Cardiac Health Management with Robust Edge Analytics and De-Risking. *Sensors* 2019, Vol. 19, Page 2733, 19(12), 2733. <https://doi.org/10.3390/S19122733>
- Umer, W., Mehmood, I., Qarout, Y., Antwi-Afari, M. F., & Anwer, S. (2025). Deep learning-based fatigue monitoring of construction workers using physiological signals. *Automation in Construction*, 177(1), 106356. <https://doi.org/10.1016/j.autcon.2025.106356>
- Unpingco, J. (2019). Python for Probability, Statistics, and Machine Learning. In *Python for Probability, Statistics, and Machine Learning*. Springer International Publishing. <https://doi.org/10.1007/978-3-030-18545-9>
- Ünsalan, C., Höke, B., & Atmaca, E. (2025). Embedded Machine Learning with Microcontrollers. In *Embedded Machine Learning with Microcontrollers* (1st ed.). Springer. <https://doi.org/10.1007/978-3-031-70912-8>
- Vara Horna, A. A. (2010). ¿CÓMO EVALUAR LA RIGUROSIDAD CIENTÍFICA DE

LAS TESIS DOCTORALES? Criterios teóricos y metodológicos aplicados a la educación.

https://www.researchgate.net/profile/Aristides-Vara-Horna/publication/215797775_Como_evaluar_la_rigurosidad_cientifica_de_las_tesis_doctorales/links/56456f0408aef646e6cc3b21/Como-evaluar-la-rigurosidad-cientifica-de-las-tesis-doctorales.pdf

- Velichko, A. (2021a). A method for medical data analysis using the lognnet for clinical decision support systems and edge computing in healthcare. *Sensors*, 21(18). <https://doi.org/10.3390/s21186209>
- Velichko, A. (2021b). A method for medical data analysis using the lognnet for clinical decision support systems and edge computing in healthcare. *Sensors*, 21(18), 6209. <https://doi.org/10.3390/s21186209>
- Verde, L., De Pietro, G., Alrashoud, M., Ghoneim, A., Al-Mutib, K. N., & Sannino, G. (2019). Leveraging artificial intelligence to improve voice disorder identification through the use of a reliable mobile app. *IEEE Access*, 7, 124048–124054. <https://doi.org/10.1109/ACCESS.2019.2938265>
- Verma, P., Tiwari, R., Hong, W. C., Upadhyay, S., & Yeh, Y. H. (2022). FETCH: A Deep Learning-Based Fog Computing and IoT Integrated Environment for Healthcare Monitoring and Diagnosis. *IEEE Access*, 10, 12548–12563. <https://doi.org/10.1109/ACCESS.2022.3143793>
- Verma, U., Tyagi, P., & Aneja, M. K. (2024). Multi-head CNN-based activity recognition and its application on chest-mounted sensor-belt. *Engineering Research Express*, 6(2), 025210. <https://doi.org/10.1088/2631-8695/AD43B9>
- Véstias, M. P., Duarte, R. P., de Sousa, J. T., & Neto, H. C. (2020). Moving deep learning to the edge. *Algorithms*, 13(5), 125. <https://doi.org/10.3390/A13050125>
- Virag, M., Kovacs, R., Marovics, G., Toth, L., Sandor, B., Voros, P., Gyori–Dani, V., Nagy, F., Eorsi, D., Sandor, J., Kiss, I., Vincze, F., Palinkas, A., Perjes, A., Rendeki, S., & Maroti, P. (2025). Bridging healthcare gaps through specialized mobile healthcare services to improve healthcare access and outcomes in rural Hungary. *Scientific Reports* 2025 15:1, 15(1), 1–17. <https://doi.org/10.1038/s41598-025-97447-9>
- Wang, J.-J., Liu, S.-H., Tsai, C.-H., Manousakas, I., Zhu, X., & Lee, T.-L. (2023). Signal Quality Analysis of Single-Arm Electrocardiography. *Sensors*, 23(13), 5818. <https://doi.org/10.3390/s23135818>
- Wang, K., Kong, S., Chen, X., & Zhao, M. (2024). Edge Computing Empowered Smart Healthcare: Monitoring and Diagnosis with Deep Learning Methods. *Journal of Grid Computing*, 22(1), 30. <https://doi.org/10.1007/s10723-023-09726-2>
- Wang, S., Dong, M., He, J., Wu, G., Li, X., Pan, X., Wu, J., Bao, R., & Pan, C. (2025). A multi-user wearable waistband system for real-time health monitoring of respiration, ECG, and body temperature. *Microelectronic Engineering*, 299(1), 112346. <https://doi.org/10.1016/j.mee.2025.112346>
- Wang, X., Bai, Z., Lin, K., & Hu, H. (2025). A hierarchically structured fibrous sensor with temperature-responsive adhesion for wearable applications. *Materials & Design*, 258(1), 114694. <https://doi.org/10.1016/j.matdes.2025.114694>
- Wang, X., Magno, M., Cavigelli, L., & Benini, L. (2020). FANN-on-MCU: An Open-

- Source Toolkit for Energy-Efficient Neural Network Inference at the Edge of the Internet of Things. *IEEE Internet of Things Journal*, 7(5), 4403–4417. <https://doi.org/10.1109/JIOT.2020.2976702>
- Warden, P., & Situnayake, D. (2019). *TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers*. <https://www.oreilly.com/library/view/tinyml/9781492052036/>
- Weizenbaum, J. (1977). La frontera entre el ordenador y la mente. In *Cibernética*. Pirámide.
- Wickberg, A., & Gärdebo, J. (2023). Computation, data and AI in Anthropocene history. *History and Technology*, 39(3–4), 328–346. <https://doi.org/10.1080/07341512.2024.2330724>
- Windisch, P., Schröder, C., Förster, R., Cihoric, N., & Zwahlen, D. R. (2023). Accuracy of the Apple Watch Oxygen Saturation Measurement in Adults: A Systematic Review. *Cureus*, 15(2). <https://doi.org/10.7759/CUREUS.35355>
- Won, J., Park, J., Park, J. W., & Kim, I. (2020). BLESeis: Low-Cost IoT Sensor for Smart Earthquake Detection and Notification. *Sensors* 2020, Vol. 20, Page 2963, 20(10), 2963. <https://doi.org/10.3390/S20102963>
- Wu, F., Qiu, C., Wu, T., & Yuce, M. R. (2021). Edge-Based Hybrid System Implementation for Long-Range Safety and Healthcare IoT Applications. *IEEE Internet of Things Journal*, 8(12), 9970–9980. <https://doi.org/10.1109/JIOT.2021.3050445>
- Wu, J., Feng, Y., & Sun, P. (2018). Sensor Fusion for Recognition of Activities of Daily Living. *Sensors* 2018, Vol. 18, Page 4029, 18(11), 4029. <https://doi.org/10.3390/S18114029>
- Xia, F., Han, L., Qiao, W., Su, S., & Song, B. (2025). Design of Intelligent Building Environment Control System Based on Internet of Things and <scp>6G</scp> Network. *Internet Technology Letters*, 8(5), 10. <https://doi.org/10.1002/itl2.70040>
- Yacchirema, D., de Puga, J. S., Palau, C., & Esteve, M. (2019). Fall detection system for elderly people using IoT and ensemble machine learning algorithm. *Personal and Ubiquitous Computing* 2019 23:5, 23(5), 801–817. <https://doi.org/10.1007/S00779-018-01196-8>
- Yang, G., Deng, J., Pang, G., Zhang, H., Li, J., Deng, B., Pang, Z., Xu, J., Jiang, M., Liljeberg, P., Xie, H., & Yang, H. (2018). An IoT-Enabled Stroke Rehabilitation System Based on Smart Wearable Armband and Machine Learning. *IEEE Journal of Translational Engineering in Health and Medicine*, 6. <https://doi.org/10.1109/JTEHM.2018.2822681>
- Yauri, R., Castro, A., Espino, R., & Gamarra, S. (2022). Implementation of a sensor node for monitoring and classification of physiological signals in an edge computing system. *Indonesian Journal of Electrical Engineering and Computer Science*, 28(1), 98. <https://doi.org/10.11591/IJEECS.V28.I1.PP98-105>
- Zaid Salah, O., Selvaperumal, S. K., & Abdulla, R. (2022). Accelerometer-based elderly fall detection system using edge artificial intelligence architecture. *International Journal of Electrical and Computer Engineering (IJECE)*, 12(4), 4430. <https://doi.org/10.11591/IJECE.V12I4.PP4430-4438>

- Zhang, C., Lu, L., Song, Y., Meng, Q., Zhang, J., Shao, X., Zhang, G., & Hou, M. (2022). Butterfly: μ W Level ULP Sensor Nodes with High Task Throughput. *Sensors (Switzerland)*, 22(8). <https://doi.org/10.3390/s22083082>
- Zhang, N., Zhang, Y., Chen, J., Qiu, S., Ma, J., Tan, L., & Dan, G. (2025). Multi-Signal Acquisition System for Continuous Blood Pressure Monitoring. *Sensors*, 25(18), 5910. <https://doi.org/10.3390/S25185910/S1>
- Zhang, X., Chai, J., Xu, L., Mei, S., Wang, X., Zhao, Y., Xue, C., Wang, Y., Cui, D., Zhang, Z., Zhang, H., & Gao, L. (2025). Direct extraction of respiratory information from pulse waves using a finger-inspired flexible pressure sensor system. *Microsystems & Nanoengineering* 2025 11:1, 11(1), 1–13. <https://doi.org/10.1038/s41378-025-00924-4>
- Zhang, X., Zhan, Y., Wang, X., & Yang, J. (2025). A Chest Strap-Based System for Electrocardiogram Monitoring. *Applied Sciences* 2025, Vol. 15, Page 5920, 15(11), 5920. <https://doi.org/10.3390/APP15115920>
- Zhang, Z., He, T., Zhu, M., Sun, Z., Shi, Q., Zhu, J., Dong, B., Yuce, M. R., & Lee, C. (2020). Deep learning-enabled triboelectric smart socks for IoT-based gait analysis and VR applications. *Npj Flexible Electronics* 2020 4:1, 4(1), 1–12. <https://doi.org/10.1038/s41528-020-00092-7>
- Zhao, X., An, Y., Xu, N., & Geng, X. (2023). Continuous label distribution learning. *Pattern Recognition*, 133, 109056. <https://doi.org/10.1016/J.PATCOG.2022.109056>
- Zholdas, N., Postolache, O., Mansurova, M., Belgibaev, B., Kunelbayev, M., & Sarsembayeva, T. (2024). Development of a wearable monitor to identify stress levels using internet of things. *Indonesian Journal of Electrical Engineering and Computer Science*, 33(3), 1486–1499. <https://doi.org/10.11591/ijeecs.v33.i3.pp1486-1499>
- Zhou, Y., Chen, P., Fan, Y., & Wu, Y. (2024). A Multimodal Feature Fusion Brain Fatigue Recognition System Based on Bayes-gcForest. *Sensors* 2024, Vol. 24, Page 2910, 24(9), 2910. <https://doi.org/10.3390/S24092910>
- Zhou, Z., Jin, Y., Fu, J., Si, S., Liu, M., Hu, Y., Gan, J., Deng, Y., Li, R., & Yang, J. (2025). Smart wireless flexible sensing system for unconstrained monitoring of ballistocardiogram and respiration. *Npj Flexible Electronics* 2025 9:1, 9(1), 1–8. <https://doi.org/10.1038/s41528-025-00388-6>
- Zim, M. Z. H. (2021). *TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC*. <https://doi.org/10.13140/RG.2.2.28602.11204>
- Zulfiqar, A. (2024). *HANDS-ON ESP32 WITH ARDUINO IDE: unleash the power of IoT with ESP32 and build exciting projects with this practical guide* (1st ed.). Packt Publishing.

Anexos

Anexo 1: Producción de artículos científicos.

Artículo de investigación (Scopus). Estado: publicado

- Yauri, R., & Espino, R. (2023). Edge device for movement pattern classification using neural network algorithms. Indones. J. Electr. Eng. Comput. Sci, 30(1), 229-236.

Indonesian Journal of Electrical Engineering and Computer Science
Vol. 30, No. 1, April 2023, pp. 229~236
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v30.i1.pp229-236

□ 229

Edge device for movement pattern classification using neural network algorithms

Ricardo Yauri¹, Rafael Espino²

¹Faculty of Systems and Informatic Engineering, Universidad Nacional Mayor de San Marcos, Lima, Perú
²Department of Electronic Engineering, Faculty of Engineering, Universidad Tecnológica del Perú, Lima, Perú

Artículo de investigación (Scopus). Estado: publicado

- Yauri, R., & La Serna, N. (2026). Edge Computing Device with Embedded Intelligence for Health Status Classification. WSEAS Transactions on Electronics

Edge Computing Device with Embedded Intelligence for Health Status Classification

RICARDO YAURI¹, NORA BERTHA LA SERNA PALOMINO²

¹ Facultad de Ingeniería de Sistemas e
Informática,

Universidad Nacional Mayor de San Marcos,
Lima,
PERÚ

Abstract: - Advances in artificial intelligence and embedded systems processing capabilities have contributed to monitoring people's health status, enabling the development of wearable devices and edge computing