UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA ELECTRÓNICO DE MONITOREO DE BIOSEÑALES PARA EL DIAGNÓSTICO MÉDICO DE COVID-19 EN PERSONAS MEDIANTE INTELIGENCIA ARTIFICIAL

Trabajo de Integración Curricular Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Comunicaciones

LÍNEA DE INVESTIGACIÓN: Tecnologías de Comunicación

AUTOR: Santiago Adolfo Gómez Lagua

TUTOR: Ing. Juan Pablo Pallo Noroña, Mg.

AMBATO – ECUADOR

septiembre - 2022

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Integración Curricular con el tema: SISTEMA ELECTRÓNICO DE MONITOREO DE BIOSEÑALES PARA EL DIAGNÓSTICO MÉDICO DE COVID-19 EN PERSONAS MEDIANTE INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Santiago Adolfo Gómez Lagua, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato y el numeral 7.4 del respectivo instructivo.

Ambato, septiembre 2022.

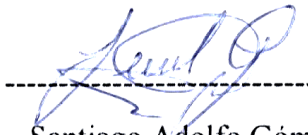
Firmado electrónicamente por:
JUAN PABLO PALLO NORONA

Ing. Juan Pablo Pallo Noroña, Mg.
TUTOR

AUTORÍA

El presente trabajo de Integración Curricular titulado: SISTEMA ELECTRÓNICO DE MONITOREO DE BIOSEÑALES PARA EL DIAGNÓSTICO MÉDICO DE COVID-19 EN PERSONAS MEDIANTE INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, septiembre 2022.



Santiago Adolfo Gómez Lagua

C.C. 1804959698

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Integración Curricular presentado por el señor Santiago Adolfo Gómez Lagua, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación titulado SISTEMA ELECTRÓNICO DE MONITOREO DE BIOSEÑALES PARA EL DIAGNÓSTICO MÉDICO DE COVID-19 EN PERSONAS MEDIANTE INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y sus reformas y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, septiembre 2022.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
JULIO ENRIQUE CUJI RODRIGUEZ

Ing. Julio Cuji, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FREDDY GEOVANNY BENALCAZAR PALACIOS

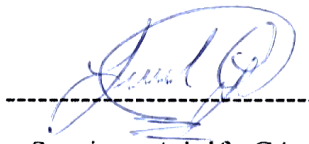
Dr. Freddy Benalcázar.
PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Integración Curricular como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Integración Curricular en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, septiembre 2022.



Santiago Adolfo Gómez Lagua

C.C. 180495968.

AUTOR

DEDICATORIA

Con mucho amor y cariño, este trabajo se lo dedico a mi madre Rosario, mi padre Alberto y mi hermano Carlos, quienes fueron los cimientos principales para formarme como un hombre de valores. Mi vida profesional se los debo a ellos.

Santiago Gómez

AGRADECIMIENTO

Principalmente mis agradecimientos a mis padres y a mi hermano, por sus apoyos incondicionales, por su paciencia, por sus consejos, ellos los que han sido mi motivo principal desde niño para superarme y salir adelante siempre.

A mis amigos, quienes estuvieron desde el principio de mi etapa universitaria y quienes con sus consejos y apoyos me han motivado a cumplir este logro personal, de corazón "Ingeniebrios".

A los ingenieros Santiago Manzano y Juan Pablo Pallo, los cuales fueron mi guía profesional en la realización de este proyecto.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

La presente investigación propone la implementación de un sistema electrónico para el monitoreo de los signos vitales y diagnóstico de COVID-19 en pacientes mediante Inteligencia Artificial. Para llegar a este fin se ha utilizado investigaciones bibliográficas en bases de datos médicas para recopilar información acerca de los síntomas que se presentan en pacientes con COVID-19, este punto es importante debido a que el algoritmo de Inteligencia Artificial tiene más eficiencia cuando existe la suficiente cantidad de datos para aprender. El conjunto de datos o "dataset" se compone de datos etiquetados los cuales son "Ambulatorio", "Revisión médica" y "Hospitalización", por lo que se opta por utilizar algoritmos de clasificación de aprendizaje supervisado. Se realiza una exploración de datos, para posteriormente con los signos vitales que son adquiridos desde los sensores MAX30100 y MLX90614, entrenar y elegir el algoritmo que más eficiencia tiene. Los algoritmos que son utilizados son el de Máquinas de Vectores de Soporte, Árboles de Decisiones y Naive Bayes Multinomial. Todo este proceso es realizado en una Raspberry Pi Zero aprovechando sus recursos y dimensión apto para una pulsera, los datos son almacenados en una base de datos local y enviados a la nube mediante peticiones HTTP, con la finalidad de que el médico tratante se mantenga informado del estado de salud de sus pacientes, así también pueda recibir correos de alerta indicando el nivel de riesgo de los mismos. Utilizando las métricas de evaluación para los algoritmos de Inteligencia Artificial se determinó que el algoritmo de Máquina de Vectores de Soporte es el que más se acopla en la clasificación de las 3 categorías, teniendo como resultados para la clase "Ambulatorio" un "Recall" igual a 0.75 y una precisión de 0.90, para la clase "Revisión médica" 0.92 y 0.83 y para la clase "Hospitalización" 1.00 y 0.95 concluyendo que el algoritmo maneja eficientemente la clasificación de las 3 clases con respecto a los otros algoritmos.

Palabras clave: Machine Learning, Inteligencia Artificial, COVID-19, Signos vitales.

ABSTRACT

The present research proposes the implementation of an electronic system for the monitoring of vital signs and diagnosis of COVID-19 in patients by means of artificial intelligence. To this end, bibliographic research in medical databases has been used to collect information about the symptoms that occur in patients with COVID-19, this point is important because the artificial intelligence algorithm is more efficient when there is enough data to learn. The dataset is composed of labeled data which are "Ambulatory", "Medical review" and "Hospitalization", so it is chosen to use supervised learning classification algorithms. A data exploration is performed, and then with the vital signs that are acquired from the MAX30100 and MLX90614 sensors, to train and choose the most efficient algorithm. The algorithms used are Support Vector Machines, Decision Trees and Multinomial Naive Bayes. All this process is performed on a Raspberry Pi Zero taking advantage of its resources and dimension suitable for a bracelet, the data are stored in a local database and sent to the cloud via HTTP requests, in order that the treating physician is kept informed of the health status of their patients, so you can also receive alert emails indicating the level of risk of the same. Using the evaluation metrics for the artificial intelligence algorithms, it was determined that the Support Vector Machine algorithm is the one that best fits in the classification of the 3 categories, having as results for the "Ambulatory" class a "Recall" equal to 0.75 and a precision of 0.90, for the "Medical Review" class 0.92 and 0.83 and for the "Hospitalization" class 1.00 and 0.95, concluding that the algorithm efficiently handles the classification of the 3 classes with respect to the other algorithms.

Keywords: Machine Learning, Artificial Intelligence, COVID-19, Vital signs.

CAPITULO I

MARCO TEÓRICO

1.1 Tema de investigación

Sistema electrónico de monitoreo de bioseñales para el diagnóstico médico de COVID-19 en personas mediante Inteligencia Artificial.

1.1.1 Planteamiento del problema

La enfermedad por coronavirus 2019 (COVID-19), causada por el síndrome respiratorio agudo severo coronavirus 2 (SARS-CoV-2), se ha convertido en una crisis de salud pública sin precedentes, teniendo como consecuencias que las infecciones por COVID-19 en todo el mundo superan los 450 millones de personas y aproximadamente 6 millones de fallecidos a causa de este virus, según estadísticas mostradas hasta el 10 de marzo del 2022 [1]. Esta enfermedad declarada como pandemia mundial a principios del año 2020 afectó profundamente en el ámbito social, económico y político, principalmente fue un desafío muy grande al área de la salud.

Según la Organización Mundial de la Salud (OMS), los síntomas más comunes del COVID-19 son fiebre, tos seca y cansancio y los síntomas menos comunes son dolores y molestias como, por ejemplo; dolor de garganta, diarrea, conjuntivitis, dolor de cabeza, entre otros, mientras que los síntomas graves incluyen dificultad para respirar, dolor en el pecho, pérdida del habla o del movimiento [2]. Pero, como se ha visto en los últimos meses, han aparecido nuevas variantes que afectan ya no en especial a personas adultas de edad mediana y a los mayores, sino que también afecta a jóvenes sanos y sin que se presente el síntoma más común, la pérdida de olfato y gusto. Para una identificación efectiva, el diagnóstico debe ampliarse para evaluar a millones de pacientes potenciales con síntomas, lo cual es crítico durante una pandemia, pues entre los desafíos incluyen la disponibilidad de botiquines médicos, trabajadores de la salud capacitados y centros de atención médica. De esta manera se ha visto la necesidad que la tecnología aborde dicho

desafío al integrar los dispositivos médicos con los sensores, algoritmos y aplicaciones móviles de salud incorporados [3].

Acorde a lo indicado, la OMS definió a la cibersalud como la única solución en Latinoamérica para cumplir con un acceso equitativo a servicios asistenciales de calidad [4], debido a que aún persisten considerables desigualdades en el acceso a los servicios de salud, debido a diversos factores como la falta de políticas sanitarias, grandes extensiones geográficas, escases en infraestructura tecnológica, entre otros.

De esta manera, la tecnología juega un papel importante en los tratamientos médicos y detección de enfermedades, ya que en el año 2020 aproximadamente el 55% de la población mundial tenía acceso a Internet y en casi 7 de cada 10 hogares, había al menos un teléfono celular [4], por lo que se afirma que el mundo está interconectado y evoluciona rápidamente a la era del "Internet de las cosas" en donde las señales son almacenadas, procesadas y analizadas mediante Inteligencia Artificial, para aplicaciones médicas de gran importancia para la salud social.

En concordancia con lo indicado se establece que si bien la atención médica profesional y la hospitalización son necesarias para los pacientes de alto riesgo de COVID-19, el aislamiento domiciliario es una estrategia efectiva para los pacientes de bajo y mediano riesgo. Sin embargo, esto requiere técnicas efectivas para monitorear de forma remota los síntomas de los pacientes. Los avances recientes en Machine Learning (ML) y Deep Learning (DL) han fortalecido el poder de las técnicas de imágenes y pueden usarse para realizar de forma remota varias tareas que anteriormente requerían la presencia física de un profesional médico [5].

El tratamiento de la salud mediante herramientas tecnológicas y la llamada Inteligencia Artificial pretenden la optimización de los diagnósticos y, eventualmente los tratamientos [6]. Su aplicación mediante sistemas que pretenden emular el razonamiento humano intenta mejorar las posibilidades de éxito, lo cual puede ser excelente para resolver problemas de diagnósticos con algoritmos unidos a base de datos y evidencia científica para mejorar el proceso diagnóstico o terapéutico.

Al inicio de la pandemia en Ecuador se observó un incremento exponencial en los hospitales de pacientes contagiados con COVID-19, por ende, estos centros de salud se saturaron y la atención médica estaba orientada en su totalidad a personas con sospechas de contagio. Debido a esta situación se empezó a implementar nuevos mecanismos de control basados en tecnologías de Inteligencia Artificial, la aplicación "SaludEC" es un ejemplo [7]. Sin embargo, estos sistemas que automatizan el diagnóstico de enfermedades no son accesible para la población geográficamente alejada de las grandes urbes.

El problema en Ecuador sigue latente por lo que un grupo de investigadores de la Universidad Espíritu Santo [8], sugieren que se debe fortalecer la vigilancia epidemiológica y la capacidad de localizar a personas contagiadas con el virus, los cuales son fundamentales para contener el avance de la enfermedad, por lo tanto, es vital proponer nuevas herramientas inteligentes de análisis de datos y mejorar las existentes para ayudar al personal médico a analizar la presencia del virus COVID-19.

Una respuesta al lento procesamiento de información de pacientes para detección de COVID-19, es el desarrollo de un sistema que solicite al usuario información médica para determinar si está infectado. Un sistema de monitoreo en tiempo real de los signos vitales del paciente exclusivo para esta enfermedad no está disponible y es poco accesible para la población. De modo que, el uso de un sistema electrónico para la monitorización de signos vitales es de suma importancia. El análisis de datos mediante el uso de Inteligencia Artificial puede reducir el tiempo de diagnóstico de COVID-19, evitando la propagación del virus y dando acceso a una atención médica a toda persona.

Este trabajo dará un aporte teórico y práctico al proyecto de investigación titulado "Sistema electrónico de monitoreo de bioseñales y alertas de distanciamiento social para la prevención de SARS-COV2 aplicando Inteligencia Artificial", aprobado con Resolución UTA-CONIN-2022-0011-R, con fecha 2 de febrero de 2022.

1.2 Antecedentes Investigativos

Una vez revisado diferentes bases de datos y repositorios a nivel nacional e internacional se encontraron trabajos similares referentes a la temática de estudio, mismos que sirvieron como guía para el desarrollo de la presente investigación, entre estos se mencionan:

En la Universidad de Cartagena, en el año 2020 se realizó un estudio titulado "Sistema IoT para la monitorización de la variabilidad del ritmo cardiaco en pruebas de usabilidad", el cual tuvo como finalidad desarrollar un sistema IoT para la monitorización de la variabilidad de ritmo cardiaco en pruebas de usabilidad. El sistema IoT recopila la variabilidad del ritmo cardiaco de manera periódica, así como el índice de estrés mental a lo largo de la experiencia del usuario. Este proyecto consta de 3 fases: la fase 1 consiste en el estudio de la variabilidad del ritmo cardiaco, la fase 2 permite explorar las herramientas y tecnologías de IoT que posibilitan la obtención y variabilidad del ritmo cardiaco, y la fase 3 consiste en el diseño e implementación del sistema IoT. Se evaluó la efectividad de los algoritmos de minería de datos tales como J48 y K-Means, los cuales permiten obtener una visión más clara del comportamiento del estrés mental en función del tiempo [1].

El trabajo investigativo desarrollado en la Escuela Colombiana de Ingeniería Julio Garavito, en el año 2021 con el tema "Aplicación de registro y reporte automático de parámetros fisiológicos para la trazabilidad psicofisiológica en personas post COVID-19" tuvo como objetivo desarrollar una aplicación de escritorio que permita el seguimiento clínicamente instrumentado de variables psicofisiológicas en múltiples pacientes. La aplicación fue desarrollada en lenguaje Python 3.8 y se definieron seis objetos para procesar siete bioseñales. Para cada objeto, se utilizaron diferentes métodos para extraer características psicofisiológicamente relevantes de las señales: ECG (Electrocardiograma), RESP (Respiración), EDA (Actividad Electrodérmica), TEMP (Temperatura), fNIRS (Espectroscopía en el infrarrojo cercano), PPG (Fotopletismografía) y EMG (Electromiografía). La interfaz de usuario se desarrolló con 3 ventanas: la ventana principal para registrar y visualizar pacientes y sus datos, una ventana para agregar mediciones en pacientes y una ventana para observar estas mediciones. La interfaz se validó procesando las señales adquiridas de un voluntario sano

y, en particular, validó la detección de complejos QRS (Combinación de las ondas Q, R y S) en el procesamiento de ECG y la detección de respiraciones en el procesamiento de RESP. La información del voluntario se almacenó correctamente y las características de las bioseñales adquiridas se extrajeron con éxito [2].

De igual forma en la investigación publicada en la Revista Cell en el año 2021, con el tema: "Sistema de Inteligencia Artificial clínicamente aplicable para el diagnóstico preciso, las mediciones cuantitativas y el pronóstico de la neumonía Covid-19 mediante tomografía computarizada" tuvo como objetivo desarrollar un sistema de diagnóstico de Inteligencia Artificial para ayudar al diagnóstico preciso del Covid-19, para su aplicación en un área epidémica y dos áreas no epidémicas en China. Usando una base de datos de tomografía computarizada (TC) de 3777 pacientes, se desarrolló un sistema de IA (Inteligencia Artificial) que puede diagnosticar neumonía por el nuevo coronavirus y diferenciarla de otras neumonías comunes y controles normales. El sistema de IA identificó marcadores clínicos de importancia que se correlacionaron con las características de lesión por el nuevo Coronavirus. Finalmente, con los datos clínicos y el sistema de IA se pudo proporcionar un pronóstico clínico preciso que puede ayudar a los médicos a considerar el manejo clínico temprano apropiado y asignar los recursos de manera adecuada [3].

En el artículo publicado en la Revista International Journal of Environmental Research and Public Healt en el año 2021, con el tema: *"Exploring an Efficient Remote Biomedical Signal Monitoring Framework for Personal Health in the COVID-19 Pandemic"*, se tuvo como objetivo diseñar un marco de monitoreo de señales biomédicas remotas que combina el Internet de las cosas (IoT), la comunicación 5G y las técnicas de Inteligencia Artificial. En el marco construido, los dispositivos IoT se utilizaron para recopilar señales biomédicas en la capa de percepción. Posteriormente, las señales biomédicas fueron transmitidas a través de la red 5G al servidor en la nube donde se despliega el modelo de aprendizaje profundo GRU-AE. Se destacó que el modelo GRU-AE propuesto puede analizar señales biomédicas multidimensionales en series temporales. Finalmente, en el artículo se describe un experimento de monitoreo de 24 semanas para 2000 sujetos de diferentes edades para obtener datos reales. En comparación con el método tradicional de monitoreo de señales biomédicas basado en el modelo AutoEncoder, el modelo GRU-AE

obtuvo un mejor rendimiento. La investigación tuvo un papel importante en la promoción del desarrollo de técnicas de monitoreo de señales biomédicas, que se pueden aplicar de manera efectiva a algunos tipos de escenarios de monitoreo de salud remotos [4].

En la Universidad Técnica de Ambato, en el año 2021 se desarrolló una investigación con el tema: "Sistema electrónico de monitoreo de signos vitales y alertas de distanciamiento social para la prevención de enfermedades respiratorias", la cual tuvo como objetivo principal construir un sistema electrónico de monitoreo de signos vitales y alertas de distanciamiento social para la prevención de enfermedades respiratorias, utilizando arquitectura IoT para el manejo de datos. Este sistema constó de un brazalete portable para la adquisición de datos, un servidor que almacena la información para mostrarla en un aplicativo web y un asistente en la aplicación Telegram para consultas y alertas con un celular inteligente. Como resultados se obtuvo que el sistema ayuda al sector de la salud, evitando aglomeraciones que generen contagios de estas enfermedades y monitorizar los signos vitales de los usuarios para recibir atención médica oportuna en caso de ser necesaria [5].

1.3 Fundamentación teórica

1.3.1 COVID-19

Los coronavirus son una gran familia de virus envueltos, pleomórficos o esféricos que presentan ARN como genoma y cuyo tamaño oscila entre los 80 a 120 nm de diámetro y en su superficie se puede observar proyecciones de glicoproteína spike. Estos virus generan enfermedades en los seres humanos y que puede ocasionar desde resfriados comunes hasta síndromes respiratorios agudos, se caracteriza por ser muy contagioso [6].

Las principales rutas de transmisión del COVID-19 son las siguientes [7]:

- **Transmisión por gotas:** Esto sucede cuando una persona que se encuentra infectada tose o estornuda y las gotas que se liberan son inhaladas o ingeridas por personas cercanas.

- **Transmisión por contacto:** Sucede cuando un sujeto tiene contacto con superficies u objetos que se encuentran contaminados y posteriormente se toca los ojos, nariz o boca.

- **Transmisión por aerosoles**: Se presenta cuando las gotas respiratorias se mezclan con el aire del ambiente de un lugar que se encuentra relativamente cerrado. Lo cual forma aerosoles que se inhalan en altas dosis causando infección.

1.3.2 Características del COVID-19 y sus efectos en la salud de las personas

Los signos vitales, tales como el ritmo cardiaco, la frecuencia respiratoria, la temperatura y la presión arterial representan las funciones esenciales del cuerpo. Estos varían con la edad, el sexo, el peso, la capacidad para ejercitarse y la salud en general. Los rangos normales de los signos vitales de un adulto sano promedio son [8]:

- Presión arterial: 90/60 mm Hg hasta 120/80 mm Hg.
- Respiración: 12 a 18 respiraciones por minuto.
- Pulsos por minuto: 60 a 100 latidos por minuto.
- Temperatura: 36.5° C a 37.3° C; promedio es 37° C.
- Saturación normal de oxígeno (SpO2): 95 a 100 %

Por medio de la medición de estos parámetros se puede diagnosticar si un paciente está contagiado con COVID-19 recopilando datos de sus signos vitales, ya que estos suelen ser muy pronunciados al momento de existir alguna enfermedad en el organismo.

Uno de los principales síntomas para identificar al COVID-19 en el organismo es una temperatura igual o mayor a 38ºC, pero esta temperatura debe ir acompañada de otros síntomas como la tos persistente, dificultad de respirar, dolor de cabeza intenso entre otros. [9]. A continuación, se describen los signos vitales y sus valores en un paciente con COVID-19 [10].

- Fiebre mayor a 38 ºC con tos seca y/o dificultad respiratoria.

- SpO2 < 94%.
- Pulsos por minuto: rango 102 a 150 por minuto.
- Respiración: 30 o más respiraciones por minuto.

Una de las características que destacan de este virus es que se estima que puede existir un 80 % de personas infectadas asintomáticas, un 15 % de grado moderado y un 5 % que llegan a la fase crítica, según Clifford Lane [19].

Como se describió, el COVID-19 se puede presentar por medio de síntomas que van desde respiratorios leves a moderados, hasta infecciones graves que concluyen con la muerte. El síntoma característico que se presenta es la fiebre con un porcentaje de casos del 99%, esta puede ser acompañado por tos seca (59%) y disnea (31%). La presencia de neumonía deriva en que el paciente esta grave, este síntoma se manifiesta debido a los síntomas recién descritos. Existen otros síntomas que se pueden presentar como lo son; la fatiga, anorexia, mialgias, odinofagia, expectoración y anosmia, aunque existen otros como náuseas y diarrea que se presentan con poca frecuencia.

El virus del COVID-19 exhibe tres grados de severidad creciente correspondientes con hallazgos clínicos distintos, respuesta de terapia y resultado clínico.

Etapa 1, infección temprana

La etapa inicial se presenta en el momento de la inoculación y el establecimiento temprano de la enfermedad. Esto implica, en la mayoría de las personas, un periodo de incubación asociado con síntomas leves y a menudo no específicos, como malestar general, fiebre y tos seca. A lo largo de este periodo, el SARS-CoV-2 se multiplica y establece la residencia en el huésped, centrándose principalmente en el sistema respiratorio.

El tratamiento en esta etapa está dirigido principalmente al alivio sintomático. Es decir, si una terapia antiviral viable se demuestra beneficiosa, dirigirse a pacientes seleccionados durante esta etapa puede reducir la duración de los síntomas, minimizar el contagio y prevenir la progresión de la gravedad.

Etapa 2

En la segunda etapa de la enfermedad, la multiplicación viral y la inflamación localizada en el pulmón es la norma:

- Durante esta etapa, los pacientes desarrollan una neumonía viral, con tos fiebre y posiblemente hipoxia (definida como una PaO2/FiO2 de < 300 mmHg).
- Las imágenes con radiografía de tórax o tomografía computarizada revelan infiltrados bilaterales u opacidades en vidrio esmerilado.
- Los análisis de sangre revelan un aumento de la linfopenia, junto con la transaminitis.
- Los marcadores de inflamación sistemática pueden estar elevados, pero no totalmente.

El tratamiento en esta etapa consiste principalmente en medidas de apoyo y terapias antivirales disponibles. En el caso de ausencia de hipoxia significativa, se puede evitar el uso de corticosteroides en pacientes con COVID-19. Sin embargo, si existe la presencia de hipoxia, probablemente los pacientes progresen y requieran ventilación médica, lo que conlleva a que el uso de la terapia antiinflamatoria, como los corticosteroides, puede ser útil y emplearse juiciosamente.

Etapa 3 (grave)

En esta etapa se manifiesta el síndrome de hiperinflamación sistemática extrapulmonar, en donde los marcadores de inflamación sistemática parecen estar elevados. En esta etapa, se pueden detectar shock, vasoplejia, insuficiencia respiratoria e incluso colapso cardiopulmonar. También existe la posibilidad de manifestar una afectación de los órganos sistémicos, incluso miocarditis.

El tratamiento consiste en la terapia personalizada, el cual depende del uso de agentes inmunomoduladores para reducir la inflamación sistémica antes de que resulte abrumadoramente en una disfunción multiorgánica. En general, el pronóstico y la recuperación de esta etapa crítica de la enfermedad es pobre, y el rápido reconocimiento y despliegue de dicha terapia puede tener el mayor rendimiento [20].

Según la OPS (Organización Panamericana de la Salud) los valores de los signos vitales de acuerdo a las etapas descritas se resumen en la tabla 1.

Tabla 1. Comportamiento de los signos vitales en cada etapa de desarrollo del COVID-19. Fuente: [21].

Estadio	Primera etapa (Leve)	Segunda etapa (Moderado)	Tercera etapa (Grave)
Saturación	95% o mayor	93% - 94%	92% o menor
Frecuencia respiratoria	≤ 20	21 - 24	≥ 25
Frecuencia cardiaca	≤ 90	91 – 130	≥ 131
Temperatura	38 ° C o mayor	38 ° C o mayor	38 ° C o mayor
Conducta	Considerar monitoreo remoto	Considerar internación hospitalaria / evaluación presencial.	Internación hospitalaria

1.3.3 Adquisición de bioseñales

Una señal es un medio de transmisión de información, cuya adquisición ayuda a obtener información acerca de la fuente que la generó. En el caso de las bioseñales, las fuentes son los sistemas fisiológicos del organismo. Las bioseñales han mostrado gran utilidad en el cuidado de la salud y dominios médicos, ya que permite al médico extraer información sobre el funcionamiento de los diferentes órganos para poder emitir un diagnóstico [11].

Las medidas médicas se encuentran agrupadas en diversas categorías que son: biopotenciales, mecánicas, acústicas, imágenes, impedancias, señales biomagnéticas y señales bioquímicas. Un sistema de adquisición y sistema de control de señales consiste de lo siguiente [12]:

- **Sensores:** Miden de manera física variables como temperatura, presión, flujo, fuerza y movimiento.
- **Acondicionamiento de señal:** Se encarga de convertir las salidas del sensor en señales que pueden ser leídas por el computador.
- **Entrada analógica:** se encarga de convertir las señales analógicas en un formato digital, de tal manera que pueda ser interpretado por el computador.
- **Computador:** Se requiere un ordenador que contenga una aplicación apropiada para el procesamiento, análisis y carga de datos en memoria. Por lo general el software puede proveer una interfaz gráfica para desplegar la información y validarla.
- **Interfaz de salida:** es la encargada de proveer un proceso apropiado de control y respuesta.

En la tabla 2 se hace un análisis de algunos dispositivos de monitoreo de signos vitales.

Tabla 2. Dispositivos electrónicos de monitores de bioseñales.

Elaborado por el investigador.

Nombre	Características	Precio	
CardiacSense	Está diseñado para uso intrahospitalario, detecta arritmias cardiacas como fibrilación auricular, taquicardia y bradicardia y paros cardiacos. La monitorización es continua durante 24 horas, 7 días de la semana a excepción del momento de carga del dispositivo. No necesita conexión a internet ya que se conecta por Bluetooth. Se puede sumergir al agua hasta una profundidad de 5 m, de acuerdo a la norma IP68. La duración de la carga es de 7 días [16].	-	
Monitor de presión arterial Yihou	Posee seguimiento de actividad de las actividades diarias como pasos, distancias, calorías quemadas, rastrea el ritmo cardiaco, el estado del sueño, etc. El brazalete utiliza CPU de alto rendimiento para controlar la presión arterial y la frecuencia cardiaca. Posee una batería de larga duración de hasta 7 días.	Amazon $ 39.99	

MorePro	Este dispositivo monitorea el nivel de oxígeno en la sangre y la presión arterial, al mismo tiempo rastrea la actividad diaria en su aplicación “FitCloudPro”. También monitorea la actividad física y rastrea los patrones de sueño para proporcionar un análisis completo de la calidad del sueño. Posee una batería de larga duración de entre 7 a 10 días de tiempo de trabajo. Es compatible con dispositivos Android 5.0 y iOS 10 con Bluetooth 4.0.	Amazon $ 47.99	
Sistema de monitoreo de signos vitales, ubicación, identificación y detector de caídas para adultos mayores	Almeida y Sánchez [17] desarrollaron una pulsera que permite monitorear los signos vitales de las personas adultas en tiempo real, tales como el pulso cardiaco, el nivel de oxígeno y la temperatura corporal, también envía alertas con la ubicación del paciente si ocurre alguna caída. La programación del dispositivo se lo hace en la tarjeta Node MCU ESP8296 y se envían datos a la nube utilizando el protocolo MQTT [17].	-	

Dispositivo wearable para el monitoreo de la oxigenación y ritmo cardiaco	Patiño Daniela [18] crea un dispositivo wearable en forma de muñequera para el monitoreo continuo y remoto de signos vitales para los pacientes afectados por la pandemia del COVID-19. Utiliza el sensor MAX30102 para registrar el ritmo cardiaco y el nivel de oxígeno en la sangre, este sensor es controlado mediante un microcontrolador ESP32 el cual transfiere los datos mediante el protocolo MQTT hacia la plataforma de Cayenne myDevices cada 10 minutos. Este dispositivo cuenta como suministro de energía dos pilas recargables de 4.2 V y 750 mAh.	-	

1.3.4 IoT e IA en telemedicina

La tecnología para telemedicina puede utilizar datos en tiempo real para posibilitar una atención remota de mejor calidad. Los pacientes pueden utilizar dispositivos que lleven puesto y otros dispositivos en el hogar para controlar la presión arterial, temperatura o frecuencia cardiaca y poder transmitir estos resultados hacia un médico para su posterior análisis.

La Inteligencia Artificial aporta nuevas capacidades a la telemedicina, ya que puede dar indicaciones para que resulte más fácil obtener el historial médico del paciente, pueden ayudar en el diagnóstico médico, ofrecer recordatorios personalizados para el suministro de medicamentos y recomendar controles rutinarios de condiciones sobre la base de datos de monitoreo personal [13].

1.3.5 Arquitectura IoT

El internet de las cosas (IoT) es el proceso que permite conectar elementos físicos cotidianos al internet, desde objetos domésticos comunes, como las bombillas de luz, hasta recursos para la obtención de salud, como los dispositivos médicos; también abarca prendas y accesorios personales inteligentes e incluso los sistemas de las ciudades inteligentes.

1.3.6 Inteligencia Artificial

La Inteligencia Artificial (IA) se refiere a sistemas o máquinas que imitan la inteligencia humana para realizar tareas y pueden mejorar iterativamente a partir de la información que recopilan. Según Marvin Minsky, uno de los pioneros de la IA, la Inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia [14].

Un aspecto importante que se puede mencionar de la Inteligencia Artificial es que abarca diversos campos, entre ellos se encuentran el reconocimiento de voz, procesamiento de lenguaje natural, visión por computadora, captura de conocimientos, robótica, planificación y optimización, entre otros, buscando que un sistema posea la capacidad de sentir, razonar, participar y aprender [15].

En la tabla 3 se presentan los algoritmos de Inteligencia Artificial más usados.

Tabla 3. Tipos de algoritmos de Inteligencia Artificial.

Fuente: [16] [17]**.**

TIPOS DE ALGORITMOS DE INTELIGENCIA ARTIFICIAL	
Algoritmos de Regresión	En las tareas de regresión, el programa de aprendizaje automático debe estimar y comprender las relaciones entre las variables. El análisis de regresión se enfoca en una variable dependiente y una serie de otras variables cambiantes, lo que lo hace particularmente útil para la predicción y el pronóstico.
Algoritmos Bayesianos	Este tipo de algoritmos por clasificación están basados en el teorema de Bayes y clasifican cada valor como independiente de cualquier otro. Lo que permite predecir una clase o categoría en función de un conjunto dado de características, utilizando la probabilidad.
Algoritmos de Agrupación	Se utilizan en el aprendizaje no supervisado, y sirven para categorizar datos no etiquetados, es decir, datos sin categorías o grupos definidos.
Algoritmos de Árbol de Decisión	Un árbol de decisión es una estructura de árbol similar a un diagrama de flujo que utiliza un método de bifurcación para ilustrar cada resultado posible de una decisión. Cada nodo dentro del árbol representa una prueba en una variable específica, y cada rama es el resultado de esa prueba.
Algoritmos de Redes Neuronales	Una red neuronal artificial (RNA) comprende unidades dispuestas en una serie de capas, cada una de las cuales se conecta a las capas anexas. Son extremadamente útiles para modelar relaciones no lineales en datos de alta dimensión, o donde la relación entre las variables de entrada es difícil de entender.
Algoritmos de Aprendizaje Profundo	Los algoritmos de aprendizaje profundo ejecutan datos a través de varias capas de algoritmos de redes neuronales, las cuales pasan a una representación simplificada de los datos a la siguiente capa.

1.3.7 Machine Learning (Aprendizaje máquina)

Según Arthur Samuel, el aprendizaje automático se define como el campo de estudio que da a las computadoras la capacidad de aprender sin estar explícitamente programado. Es utilizado para enseñar a las máquinas cómo manejar los datos más eficientemente. A veces se presentan problemas al interpretar la información extraída de los datos, entonces, se aplica el aprendizaje automático el cual tiene como objetivo aprender de los datos basándose en diferentes algoritmos. El tipo de algoritmo empleado depende del tipo de problema que se desee resolver, el número de variables, el tipo de modelo que mejor se adapte, etc. [5]. A continuación, se los describe:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semi-supervisado
- Aprendizaje por refuerzo

Aprendizaje supervisado

En el aprendizaje supervisado se trata de aprender a desempeñar una tarea partiendo de una distribución de datos previamente etiquetado por un supervisor. Por ejemplo, un conjunto de imágenes que contienen cada una algún tipo de metadato: (img00.jpg, “perro”), (img01.jpg, ”gato”), (img02.jpg, “vaca”), a partir de este conjunto de datos se pueden utilizar los diferentes tipos de algoritmos de clasificación de aprendizaje automático, con la finalidad de entrenar un modelo para posteriormente poder predecir la etiqueta correspondiente a una imagen que no está incluida en el conjunto de datos original [6].

Aprendizaje no supervisado

En el aprendizaje no supervisado se trata de aprender o encontrar alguna representación oculta en una distribución de datos sin etiquetar. La aplicación más conocida de este tipo de aprendizaje es el clustering (agrupamiento), cuyo objetivo es agrupar muestras, por ejemplo: obtener los diferentes tipos de clientes en un servicio online, agrupar productos en un comercio electrónico, identificar comportamiento en la conducción, etc [6].

Aprendizaje semisupervisado

El aprendizaje semisupervisado trabaja con conjuntos de datos en donde una porción de los datos está etiqueta y el resto no. Usualmente la cantidad de muestras etiquetadas es mucho más pequeña que las no etiquetadas.

Aprendizaje por refuerzo

Se lo utiliza principalmente en problemas relacionados con un agente interactuando con un entorno. Tal y como pasa con los seres vivos, el sistema es recompensado o penalizado cuando realiza una acción. Estos algoritmos se los llaman agentes y pueden descubrir que acciones tomar, dado el estado en el que se encuentra, y aprender guiándose en los principios de la evolución natural.

1.3.8 Proceso de Machine Learning

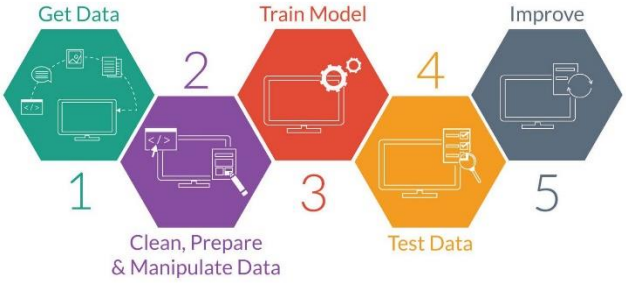


Figura 1. Ciclo de vida de Machine Learning.

Elaborado por el investigador.

Definición del problema a resolver

Se debe tener claro las entradas y salidas esperadas de acuerdo al problema que se tiene, para esto es necesario tomar en cuenta los siguientes aspectos:

- El objetivo principal del problema y que se intenta predecir.
- Conocer los datos de entrada y la disponibilidad, este aspecto es muy importante ya que el rendimiento del algoritmo de Inteligencia Artificial será mucho mayor mientras más datos aprenda.
- Saber con qué clase de problema se va a tratar y si es una clasificación binaria o agrupamiento.
- Conocer el estado actual de la variable objetivo y como se lo va a medir.

Adquisición de los datos

El primer paso para el desarrollo de un modelo de Machine Learning es la adquisición de los datos, el cual es un paso decisivo ya que influye directamente en la adecuación del modelo. Esto se lo puede realizar extrayendo información desde las páginas web, encuestas, entrevistas, etc. Cabe mencionar que también importará mucho la calidad de la información.

Elegir una medida o indicador de éxito

La medida tiene que estar directamente relacionada con el tipo de problema al que se enfrenta. Si se tiene problemas de regresión se utilizan métricas de evaluación como el error cuadrático medio. Si el problema es de clasificación se usan métricas de evaluación como la precisión, exactitud y recuerdo.

Se hace un análisis de las principales métricas de evaluación que se utilizan en los problemas de clasificación, los cuales son:

Matriz de confusión

Es una tabla con filas y columnas que contabilizan las predicciones en comparación con los valores reales. Se usa esta tabla para comprender si el modelo se comporta bien o mal, y si es de utilidad para mostrar explícitamente si una clase es confundida con otra [20]. Una matriz de confusión para un clasificador binario tiene la estructura que se presenta en la tabla 4.

La matriz de confusión consta de los siguientes elementos:

- Verdaderos positivos (VP): cuando la clase de observación es verdadera y la clase de predicción también es verdadera.
- Falsos negativos (FN): cuando la clase de observación es verdadera pero la clase de predicción es falsa.
- Falsos positivos (FP): cuando la clase de observación es falsa pero la clase de predicción es verdadera.
- Verdaderos negativos (VN): cuando la clase de observación es falsa y la clase de predicción también es falsa.

Tabla 4. Matriz de confusión para un clasificador binario.
Elaborado por el investigador.

	Predicción	
	Positivos	**Negativos**

Observación			
	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Accuracy (ACC)

Se refiere a una medida que determina que tan bueno es el algoritmo para predecir en general, es decir la puntuación general tomando en cuenta la clase positivo y la clase negativa. Esta medida se la calcula mediante la ecuación (**1)**. Es un valor que va entre 0 y 1, mientras más se acerca a 1, mejor es el algoritmo [35].

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \quad \textbf{(1)}$$

Es muy importante conocer que esta métrica es una buena medida si las clases de variables de salida están equilibradas.

Recall (TPR)

Se refiere a una medida que determina que tan bueno es el algoritmo para predecir la clase positiva. Esta métrica se diferencia de la "Precision" ya que es independiente al número de clasificaciones de muestras negativas. Si el enfoque está en minimizar los falsos negativos, entonces el "Recall" debe acercarse lo más posible a 1 [35]. Se lo calcula mediante la ecuación (2).

$$TPR = \frac{VP}{VP + FN} \quad \textbf{(2)}$$

Precisión

La precisión determina la relación entre las muestras positivas clasificadas correctamente (VP) y el total de, muestras positivas clasificadas (VP o FP) [36]. Esta métrica ayuda a visualizar la confiabilidad del modelo de aprendizaje automático al clasificar el modelo como positivo.

$$Precision = \frac{VP}{VP + FP} \quad \textbf{(3)}$$

F1-Score

Esta métrica es una de las más importantes en el aprendizaje automático ya que resume con elegancia el rendimiento predictivo de un modelo mediante la combinación de dos métricas, “Precision” y “Recall” [37]. Se lo calcula mediante la ecuación (4).

$$F1 = \frac{2VP}{2VP + FP + FN} \quad \textbf{(4)}$$

Se muestra un ejemplo en la tabla 5, cuyas métricas constan de valores aleatorios, suponiendo que se ha entrenado tres modelos distintos para la predicción de la diabetes, y cada modelo consta de valores diferentes tanto de precisión como de recuperación.

- Si se determina que los errores causados por FP son más indeseables, entonces se seleccionará un modelo basado en “Precision” y por lo tanto se elegirá el modelo C en donde, el valor igual a 0.70 indica que la cantidad de FP son menores con respecto a los otros modelos.
- Si se determina que los errores causados por FN son las indeseables, se seleccionará un modelo basado en el “Recall” y por lo tanto se elegirá el modelo B en donde, el valor igual a 0.90 indica que la cantidad de FN son menores con respecto a los otros modelos.
- Pero, si se determina que los errores causados por FP y FN son indeseables, se seleccionará un modelo basado en la puntuación F1 y se elegirá el modelo A [21].

Tabla 5. Selección de modelo mediante precisión, recuperación o puntuación F1.
Elaborado por el investigador.

	“Precision”	**“Recall”**	**F1 - Score**
Modelo A	0.50	0.80	0.62
Modelo B	0.25	0.90	0.39
Modelo C	0.70	0.40	0.51

Protocolo de evaluación

- ***Validación "Hold Out"***

Este método consiste en mantener una porción del set de datos como conjunto de datos de prueba. Es decir, se entrena el modelo con la fracción restante de datos y se evalúa su rendimiento con el conjunto de datos de prueba. La desventaja de este método es que, si hay poco volumen de datos disponible, los conjuntos de datos de validación y prueba tendrán pocos casos, lo que causará que el modelo no sea efectivo [22].

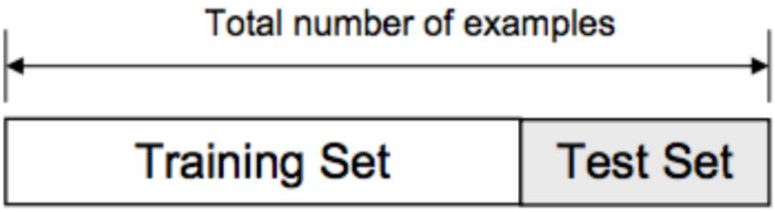


Figura 2. Método train-test.
Fuente: [16].

- ***Validación "K-Fold"***

Este método consiste en dividir los datos en K particiones de la misma dimensión. Para cada partición "i", el modelo es entrenado con las restantes K-1 particiones, y evaluado en la propia partición "i". La puntuación final es la media de las K puntuaciones obtenidas. El número de iteraciones necesarias está determinado por el valor *k* escogido, por lo general se recomienda un *k* entre 5 y 10.

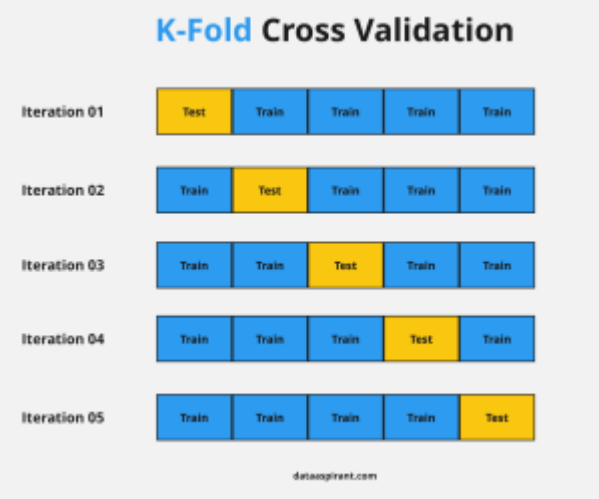


Figura 3. Validación "K-Fold".
Fuente: [40].

Preparación de los datos

En esta etapa se trata de hacer una lectura a profundidad de los datos que se disponga con el objetivo de comprender la función y los impactos que estos tienen con el objetivo de predicción que se ha establecido.

Análisis y exploración de los datos

El estudio de los datos empieza con su descripción (nombre, tipo, etc), así como por distintos procesos de tratamiento, entre los cuales están la eliminación de los datos que no sirven y la indagación de datos que falten, para posteriormente combinarlos con el fin de disponer de un conjunto de conocimientos apropiados para el aprendizaje y alcanzar el objetivo. Los campos vacíos generalmente se presentan con los indicadores "NaN" o "Null", esto genera un problema ya que la mayoría de algoritmos no manejan esos valores faltantes, por lo tanto, se reemplaza estos campos por algún dato mediante técnicas de estadística descriptiva [40].

Balanceo de los datos

La importancia del balanceo de datos se presenta debido a que un conjunto de datos desequilibrados representa un sesgo severo en la distribución de categorías. El sesgo se refiere a un peso desfavorable en la asignación de las categorías, lo que conlleva a ignorar por completo a la categoría minoritaria y, por ende, esto representa un gran problema ya que usualmente las predicciones más importantes están en esta minoría [41]. Este problema se puede resolver volviendo a muestrear aleatoriamente el conjunto de datos de entrenamiento de la siguiente manera:

- UnderSampling

  El submuestreo aleatorio involucra la selección aleatoria de ejemplos de la clase mayoritaria con la finalidad de eliminarlos de los datos de entrenamiento. Esto reduce el número de ejemplos en la clase mayoritaria del conjunto de datos de entrenamiento, proceso que se repite hasta lograr la distribución de clases deseada, por ejemplo, tener un número igual de ejemplos para cada clase [41]. En Python se lo puede realizar mediante la función "RandomUnderSampler" que provee la librería IMBLearn.

- OverSampling

  El sobremuestreo aleatorio involucra duplicar aleatoriamente los ejemplos contenidos en la clase minoritaria y agregarlos al conjunto de datos de entrenamiento. Esto se lo

hace varias veces seleccionando los ejemplos del conjunto de datos original hasta que el mismo esté balanceado [41]. En Python esto se lo puede realizar mediante la función "RandomOverSampler" que provee la librería IMBLearn.

Codificación de etiquetas de clases nominales

Cuando se trabaja con datos categóricos es frecuente encontrarse con datos que contengan características ordinales y nominales. Las características ordinales pueden ser ordenadas, por ejemplo, la talla de ropa: L > M > S, en cambio las características nominales no involucran ningún orden, por ejemplo, color de ropa: rojo, azul, verde. Los métodos para tratar este tipo de características son:

- Conversión de características ordinales

Consiste en convertir estos valores a número enteros: L:3, M:2, S:1

- Codificación de etiquetas nominales

Se trata de representar numéricamente los datos que contengan texto.

Codificación One-hot

Se utiliza para generar un vector con una longitud igual al número de categorías en un conjunto de datos, en donde una categoría es una sola palabra distinta [43]. Es decir, crea una columna para cada valor distinto que aparezca en la característica que se está codificando, tal como se muestra en la figura 4**Figura *4***.

Set of all the words in the corpus		R1: Great Restaurant and great service !	R2: They can do better to provide better service	R3: Only two thumbs up, worst service ever
great		1	0	0
restaurant		1	0	0
and		1	0	0
service		1	1	0
they		0	1	0
can		0	1	0
do		0	1	0
better		0	1	0
to		0	1	0
provide		0	1	0
only		0	0	1
Two		0	0	1
thumbs		0	0	1
up		0	0	1
worst		0	0	1
ever		0	0	1

Figura 4. Codificación One-Hot.

Fuente: [43].

Count Vector

Se utiliza para transformar un texto determinado en un vector en función de la frecuencia de cada palabra que aparece en todo el texto. Crea una matriz en la que cada palabra única se representa por una columna y cada muestra de texto del documento es una fila en la matriz. El valor de cada celda se refiere al recuento de la palabra en aquella muestra de texto en particular [4]. En la figura 5 se muestra un ejemplo en donde se utiliza una lista con dos párrafos para realizar la extracción de las características de cada palabra de estos párrafos mediante el método “CountVectorizer”.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
text = ["Amo escribir codigo en Python. Amo el código en Python",
        "Odio escribir codigo en Java. Odio el código en Java"]
df = pd.DataFrame({'review':['review1','review2'],'text':text})
cv = CountVectorizer()
cv_matrix = cv.fit_transform(df['text'])
df_dtm = pd.DataFrame(cv_matrix.toarray(), index=df['review'].values, columns=cv.get_feature_names())
print(df_dtm)
```

Figura 5. Ejemplo de extracción de características mediante "CountVectorizer".

Elaborado por el investigador.

En la figura 6 se observa la matriz de recuenta de palabras en donde existen 9 palabras únicas en el texto representadas como columnas de la matriz. Hay 2 ejemplos de texto, review1” y “review2”, los cuáles son los párrafos de la lista, cada uno representado como

filas de la matriz. Cada celda contiene un número que representa el conteo de la palabra en ese texto en particular.

	amo	codigo	código	el	en	escribir	java	odio	python
review1	2	1	1	1	2	1	0	0	2
review2	0	1	1	1	2	1	2	2	0

Figura 6. Matriz de recuento de palabras.
Elaborado por el investigador.

Tf – Idf (Term frecuency – Inverse document frecuency)

TF-IDF (Term Frequency Inverse Document Frequency) se lo define como el cálculo de la relevancia de una palabra en un texto. El significado aumenta proporcionalmente al número de veces que aparece una palabra en el texto, pero se compensa con la frecuencia de palabras en el conjunto de datos [4]. En la figura 7 se muestra un ejemplo utilizando este método, en donde se utiliza el mismo texto que el ejemplo de la figura 5.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
text = ["Amo escribir codigo en Python. Amo el código en Python",
        "Odio escribir codigo en Java. Odio el código en Java"]
df = pd.DataFrame({'review':['review1','review2'],'text':text})
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df['text'])
df_dtm = pd.DataFrame(tfidf_matrix.toarray(), index=df['review'].values, columns=tfidf.get_feature_names())
print(df_dtm)
```

Figura 7. Ejemplo de extracción de características mediante "TfidfVectorizer".
Elaborado por el investigador

En la figura 8 se muestra la matriz con las mismas columnas y filas que en la matriz de la figura 6, pero el contenido de cada celda, en este caso, representa la relevancia de las palabras únicas de cada párrafo del texto. En el primer párrafo (review1) la palabra “amo” tiene mayor valor que las demás ya que es la que menos se repite entre los dos párrafos, en cambio las demás palabras tienen menor valor debido a que se repiten entre los dos párrafos, es decir son comunes, por lo tanto, es más conveniente utilizar este método ya que no toma en cuentas las palabras comunes al momento de asociar cada párrafo a un contexto.

	amo	codigo	código	el	en	escribir	java	odio	python
review1	0.576152	0.204969	0.204969	0.204969	0.409937	0.204969	0.000000	0.000000	0.576152
review2	0.000000	0.204969	0.204969	0.204969	0.409937	0.204969	0.576152	0.576152	0.000000

Figura 8. Matriz de relevancia de palabras.

Elaborado por el investigador

Partición de datos

Los modelos de Machine Leaning aprenden a partir de los datos con que se los entrene. Con esto se intenta encontrar un patrón que les permita predecir el resultado para un nuevo caso. Pero, para poder comprobar si el modelo funciona correctamente, se necesita de un conjunto de datos de prueba, es por esto que el dataset se divide en dos partes: datos para el entrenamiento y datos para las pruebas. Normalmente para el conjunto de datos prueba se acostumbra asignar un 30 % de todo el dataset, quedando el 70 % para el entrenamiento, aunque hay que tener en cuenta la cantidad de datos que se dispone en el conjunto de datos dataset ya que, si esta es poca pues se necesitará de más porcentaje para el conjunto de entrenamiento con la finalidad de que el modelo pueda aprender bien. La proporción para el conjunto de datos tanto de entrenamiento como de prueba podría ser 80:20 o en casos extremos un 90:10. Hay que tener en cuenta que hay que evitar el sobreajuste u "overfitting" y el subajuste o "underfitting".

- Underfitting

  El modelo no se ajusta bien a los datos de entrenamiento cuando el rendimiento del modelo es bajo en los datos de entrenamiento. Esto es causado porque el modelo no puede relacionar los ejemplos de entrada (X) y los valores objetivo (Y).

- Overfitting.

  Se presenta cuando el modelo está sobreajustando los datos de entrenamiento, lo que quiere decir que el modelo está memorizando los datos que ha visto pero no puede generalizar a ejemplos no vistos [20]. Estos dos conceptos se lo pueden analizar en la figura 9.

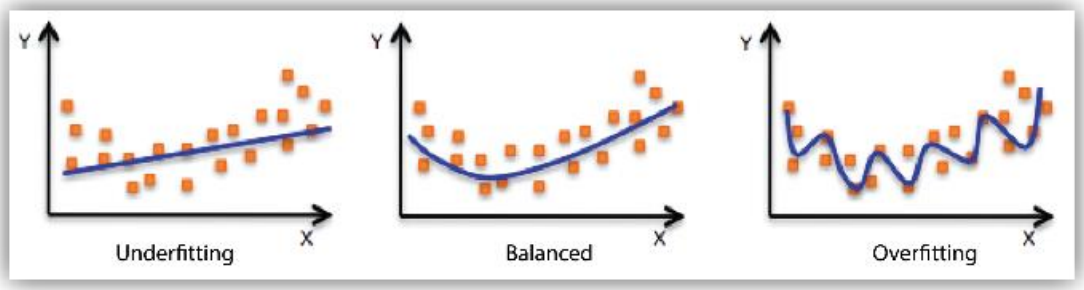


Figura 9. Ajuste del modelo de Machine Learning.

Fuente: [20].

Modelamiento

Una vez que la información que se tiene ha sido analizada es momento de elegir un modelo/algoritmo de Machine Learning que sea compatible con el objetivo [44].

- Clasificación binaria

  Realiza la predicción de un resultado binario, en pocas palabras, experimentos estadísticos con únicamente dos resultados posibles. Ejemplo: ¿Cuál es el estado de ánimo del cliente?, feliz o triste [44].

- Modelo de clasificación multiclase

  Realiza la predicción para varias clases, es decir, tiene la capacidad de predecir uno o más resultados. Ejemplo: ¿Cómo estuvo la prueba de manejo para el estudiante?, fácil, regular o difícil [44].

- Modelo de regresión

  Realiza la predicción de un valor numérico. Ejemplo: ¿Cuál será la temperatura en Quito mañana? [44].

A continuación, se presentan los modelos o algoritmos más utilizados:

- Regresión lineal,
- Regresión logística.
- Random Forest.
- Redes Bayesianas.

- Redes neuronales recurrentes
- K-Nearest Neighbors.

Evaluación del modelo

Se recomienda siempre evaluar un modelo de Machine Learning para determinar si éste hará un buen trabajo de predicción. Esto se lo hace de acuerdo a las métricas de avaluación presentados anteriormente en la parte de indicadores de éxito. Por ejemplo, si la métrica "Accuracy" es menor o igual al 50 % el modelo de será de gran utilidad, de lo contrario si alcanza un 90 % se puede deducir que el modelo tendrá una buena confianza en sus predicciones. Hay que tener en cuenta la métrica que con la que se desea evaluar el modelo.

Predicción

Finalmente, obtenido el modelo adecuado, se puede implementar para poder realizar las predicciones. En esta etapa se puede ya ingresar datos nuevos al modelo para obtener resultados adecuados

1.3.9 Algoritmos de clasificación de aprendizaje supervisado

Support Vector Machines (SVM).

SVM es un algoritmo de aprendizaje automático supervisado cuyo uso se presenta en problemas de clasificación o regresión. En este algoritmo se traza cada elemento de datos como un punto en un espacio n-dimensional, donde "n" es la cantidad de características que posee, el valor de cada una de estas es el valor de una coordenada particular. Posteriormente se realiza la clasificación encontrando el hiperplano que diferencia muy bien las dos clases [39].

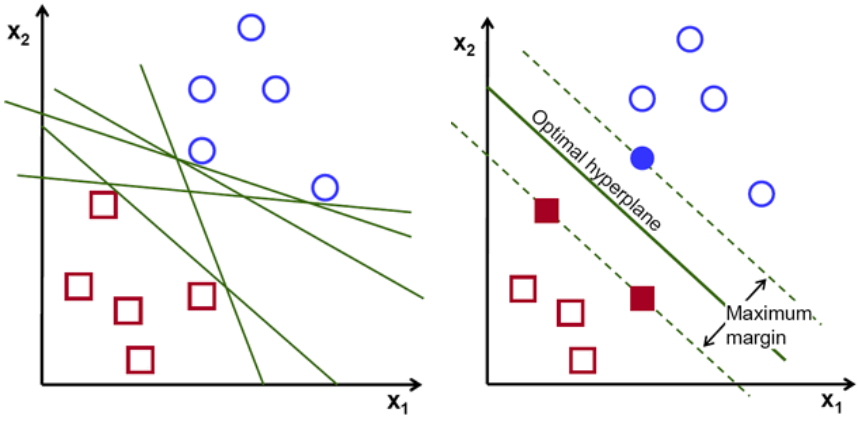


Figura 10. Hiperplanos de clasificación en SVM.

Fuente [40].

Si se requiere separar las dos clases de puntos de datos, se pueden escoger muchos hiperplanos posibles, estos son límites de decisión que ayudan en la clasificación de los puntos de datos. El objetivo de SVM es encontrar un plano que tenga el mayor margen, lo cual es la distancia máxima entre puntos de datos de ambas clases, mientras más es esta distancia, los puntos se podrán clasificar con más confianza.

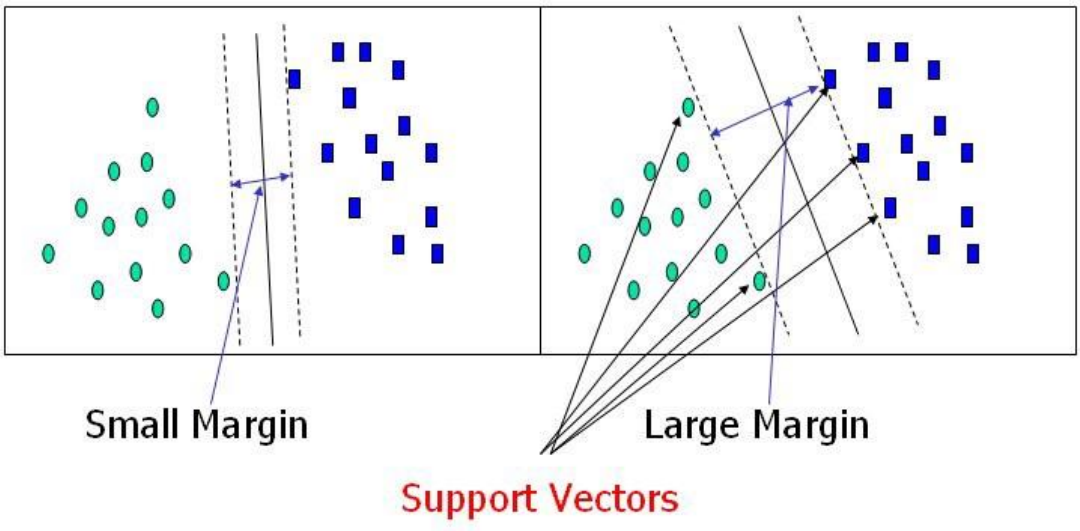


Figura 11. Vectores de soporte.

Fuente [40].

Los vectores de soporte, como se observa en la figura 11, son puntos de datos que se encuentran más cerca del hiperplano e influyen en la posición y orientación del hiperplano. Mediante estos vectores se puede maximizar el margen del clasificador, por lo tanto, estos son los puntos que ayudan a construir el algoritmo SVM [6].

Árboles de decisión

Este modelo realiza predicciones determinando una serie de pruebas sencillas sobre el punto dado. El proceso se lo puede representar mediante un árbol, en donde los nodos que no son hojas manifiestan las pruebas. En las hojas del árbol se localizan los valores a predecir tras descender a través del árbol de la forma prescrita por la prueba de cada nodo. Este modelo se lo puede utilizar tanto para la clasificación como para la regresión [30]. En pocas palabras, la estructura de un árbol decisión está dada por ramas y nodos de tipos diferentes:

- Nodos internos

  Manifiestan cada una de las características o propiedades a considerar para tomar una decisión.

- Ramas

  Manifiestan la decisión que se da en función de una condición puntual.

- Nodos finales

  Partiendo de un apto número de divisiones, un árbol de decisión puede representar un clasificador complejo y clasificar eficazmente cualquier conjunto de entrenamiento, tal y como se muestra en la figura 12.

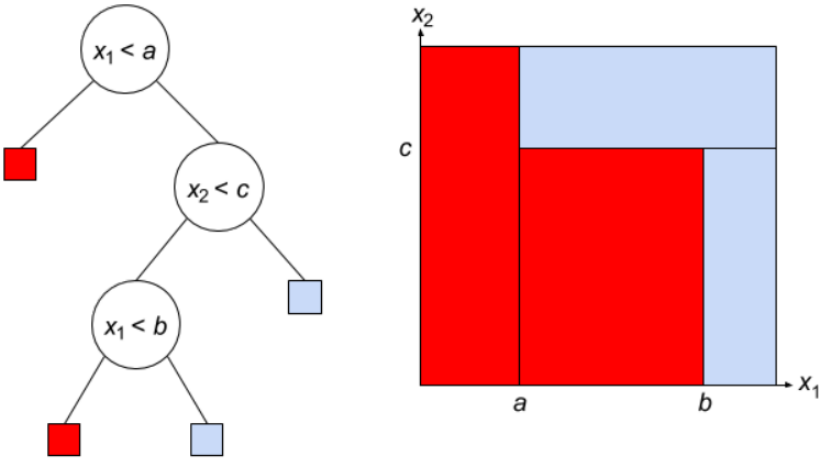


Figura 12. Comportamiento del modelo de Árbol de Decisiones.

Fuente: [30].

Clasificación de Naive Bayes

Este clasificador se basa en el teorema de Bayes y se lo utiliza para resolver problemas de clasificación. Principalmente se lo utiliza en clasificación de texto que incluye un conjunto de datos de entrenamiento de grandes cantidades. Es un algoritmo muy simple y uno de los más efectivos que ayuda a construir modelos de aprendizaje automático que puedan hacer predicciones rápidas. Se lo puede utilizar tanto para clasificaciones binarias como para clasificaciones múltiples .

Existen tres tipos de modelos de Naive Bayes, los mismo que se detallan a continuación:

- Gaussiano

  Este modelo asume que las características se presentan mediante una distribución normal, lo que significa que, si los predictores toman valores continuos en lugar de discretos, el modelo asume que estos valores se muestrean a partir de la distribución Gaussiana.

- Multinomial

  Este modelo se lo utiliza cuando se presentan datos con distribución multinomial. Es utilizado principalmente para la clasificación de documentos, lo cual quiere decir que determina a qué categorías como deportes, ciencias, etc. pertenece un documento.

- Bernoulli

  Este modelo tiene funciones similares al del modelo multinomial, con la diferencia de que las variables predictoras son variables booleanas independientes. Por ejemplo, si una palabra en particular está presente o no en un documento, lo que lo hace también muy utilizado para tareas de clasificación de texto [31].

1.3.10 Base de datos

Una base de datos es un instrumento que sirve para recopilar datos, los organiza y los relaciona con la finalidad de que se logre realizar una búsqueda rápida y recuperar con el apoyo de un ordenador. En la tabla 6 se hace un análisis de las ventajas y desventajas de las bases de datos más utilizadas.

1.3.11 Almacenamiento en la nube

El almacenamiento en la nube es un servicio que permite almacenar datos transfiriéndolos a través de Internet o de otra red a un sistema de almacenamiento externo que mantiene

un tercero. Hay cientos de sistemas de almacenamiento en la nube diferentes que abarcan desde almacenamiento personal, que guarda o mantiene copias de seguridad de correo electrónico, fotos, vídeos y otros archivos personales de un usuario, hasta almacenamiento empresarial, que permite a las empresas utilizar almacenamiento en la nube como solución comercial de copia de seguridad remota donde la compañía puede transferir y almacenar de forma segura archivos de datos o compartirlos entre ubicaciones [18].

Tabla 6. Parámetros de los principales sistemas gestores de bases de datos.
Elaborado por el investigador.

	MySQL	Microsoft SQL Server	PostgreSQL
Software	Libre y gratuito.	Es gratuito con limitaciones.	Libre y gratuito.
Lenguaje	SQL	SQL	SQL
Arquitectura	Cliente - Servidor	Cliente - Servidor	Cliente - Servidor
Plataformas	La mayoría de plataformas, incluyendo Linux.	Solo tiene soporte para el sistema operativo Windows.	La mayoría de plataformas, incluyendo Linux.
Seguridad	Posee un sistema de privilegios, el cual establece quien se puede conectar al servidor.	Posee funciones para recuperación y restauración de datos.	Es un sistema altamente confiable, estable, escalable y seguro.
Uso	Fácil de instalar y configurar.	Dificultad en la instalación y uso si el ordenador no cuenta con suficientes recursos.	Fácil de instalar y configurar.
Requisitos de hardware y software	No se necesita de muchos recursos en el ordenador para ejecutar el programa.	Se necesita de mucha memoria RAM.	No se necesita de muchos recursos en el ordenador para ejecutar el programa.

Se puede utilizar un servicio de hosting el cual además de ser un servicio de alojamiento en la nube, permite publicar un sitio web en internet. En la tabla 7 se hace una comparación de las empresas que proveen este servicio.

Tabla 7. Análisis de servicios de alojamientos online.
Elaborado por el investigador.

	Gratuidad	Base de datos	Seguridad	Website	Memoria	Correo
000webhost	Si	MySQL	Certificación SSL	1	300 MB	No
FreeHosting	Si	MySQL	Certificación SSL	1	10 GB	Si
Swhosting	Si	MySQL	Certificación SSL	1	10 GB	Si
Hostgator	No	MySQL	Certificación SSL	1	Ilimitado	Si

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema electrónico de bioseñales para el diagnóstico médico de COVID-19 mediante Inteligencia Artificial.

1.4.2 Objetivos específicos

- Analizar los sistemas electrónicos de adquisición de señales vitales en personas.
- Evaluar los algoritmos de Inteligencia Artificial para el diagnóstico médico de enfermedades respiratorias.
- Implementar el sistema de procesamiento de bioseñales para la detección y prevención del COVID-19 aplicando Inteligencia Artificial.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

2.1.1 Selección de software y hardware

2.1.1.1 Capa de sensorización

La selección del hardware para la capa de sensorización se lo realizó mediante un análisis técnico de los distintos tipos de dispositivos compatibles con el prototipo.

Microordenador

Debido a las características que se presentan en la tabla 8 de cada uno de los ordenadores de placa única, se procede a elegir a la Raspberry Pi Zero debido a sus dimensiones, recursos para la comunicación tanto alámbrica como inalámbrica, costo y disponibilidad en el mercado ecuatoriano.

Tabla 8. Comparación de los microordenadores.
Elaborado por el investigador.

	Raspberry PI Zero 2	Orange PI Zero 2	Banana Pi
Dimensiones	65 x 30 mm	53 x 60 mm	65 x 52.5 mm
Sistema Operativo	Raspbian	Android10, Ubuntu, Debian.	Raspbian, Bananian, Debian
Procesador	ARM Cortex-A53 Quad-core 1 GHz	ARM Cortex-A53 Quad-core 1.5 GHz	Allwinner H2+, Quad-core Cortex-A7
Puertos de comunicación	I2C, SPI, UART, Micro USB.	I2C, SPI, UART, USB.	I2C, SPI, UART, USB.
WiFi	2.4 GHz. 802.11 b/g/n	2.4 GHz. 802.11 a/b/g/n/ac	2.4 GHz. 802.11 b/g/n
Bluetooth	BT4.2	BT5.0	BT4.0
Memoria RAM	512 MB LPDDR2	512 MB DDR3	512 MB DDR3

Voltaje de operación	Interfaz Micro USB, 5V-2A	Interfaz tipo C, 5V-2A.	Interfaz Micro USB, 5V-2A
Costo	$ 19,91	$ 28,30	$ 36,17

Sensor de temperatura corporal

Se elige al sensor MLX90614 por las características necesarias presentadas en la tabla 9 para el desarrollo del prototipo como el voltaje de operación, la compatibilidad, el rango de operación y la disponibilidad en el mercado ecuatoriano.

Tabla 9. Comparación entre sensores de temperatura corporal.
Elaborado por el investigador.

	MLX90614	**MLX90640**
Variable física a sensar	Temperatura corporal	Temperatura corporal
Dimensiones	1.6 x 1.1 cm	2.7 x 1.6 cm
Compatibilidad	Arduino, Raspberry Pi	Arduino, Raspberry Pi
Rango de medición	40℃ a +170℃	-40 °C a +300 °C
Rango de error	± 1 °C	±1 °C
Interfaz de comunicación	Interfaz I2C (dirección 0x5A)	Interfaz I2C (dirección 0x33)
Voltaje de operación	3.3 V – 5 V	3.3 V – 5 V
Corriente de operación	1 mA	1 mA
Disponible en el mercado	Si	No
Costo	$16.00	$ 82.99

Sensor de pulsioximetría MAX30100

Se elige el sensor MAX30100 debido a las características necesarias presentadas en la tabla 10 para el desarrollo del proyecto, como lo son; las variables físicas a sensar, el voltaje de operación, la compatibilidad y la disponibilidad en el mercado.

Por lo tanto, los recursos necesarios para la capa de sensorización son los siguientes:

- Sensor MAX30100
- Sensor MLX90614
- Raspberry Pi Zero

Tabla 10. Comparación de sensores de pulsioximetría.
Elaborado por el investigador.

	MAX30100	**MAX30102**	**SFH7050**
Variables físicas a sensar	Frecuencia cardiaca, saturación de oxígeno	Frecuencia cardiaca, saturación de oxígeno	Frecuencia cardiaca, saturación de oxígeno
Longitud de onda de emisión	Rojo 660 nm IR 920 nm	Rojo 660 nm IR 920 nm	Rojo 640 nm IR 940 nm
Voltaje de operación	3.3 V – 5V	3.3 V – 5V	3.3 V – 5V
Corriente de operación	1.2 mA	1.2 mA	20 mA
Interfaz de comunicación	Interfaz I2C (dirección 0x57)	Interfaz I2C (dirección 0x57)	Interfaz I2C (dirección 0x57)
Temperatura de operación	- 40ºC - 85ºC	- 40ºC - 85ºC	-
Dimensiones	1.4 cm x 1.7 cm	2.06 cm x 1.5 cm	0.47 cm x 0.25 cm
Compatibilidad	Arduino, Raspberry Pi	Arduino, Raspberry Pi	Arduino
Disponible en el mercado	Si	Si	No
Costo	$ 9.00	$ 10.00	$ 39.00

2.1.1.2 Capa de aplicación

El procesamiento de los datos se lo hace en la Raspberry Pi mediante scripts desarrollados en Python, debido a que es el lenguaje de programación que viene instalado en el sistema operativo de Raspbian. Los datos adquiridos y el diagnóstico a realizarse se presentan en una pantalla OLED de 1.3 pulgadas, el cual es compatible con la Raspberry Pi Zero W y además sus dimensiones se ajustan a las que se requiere para una pulsera electrónica.

Suministro de energía del dispositivo

Debido a que el dispositivo utiliza tecnología wearable la característica importante es su suministro de energía, por ende, se utiliza un módulo de batería de litio de 1200 mAh construida específicamente para la Raspberry Pi Zero el cual puede funcionar hasta 4 horas. Tiene un puerto de carga Mini USB con el cual se puede cargar la batería y al mismo tiempo se puede hacer funcionar la Raspberry Pi.

Figura 13. Módulo de alimentación de batería de litio PiSugar para Rasberry Pi Zero W.

Por lo tanto, los materiales que se necesitan son:

- Lenguaje de programación Python
- Módulo de alimentación de batería de litio PiSugar
- Pantalla OLED 1.3 pulgadas

Almacenamiento de datos y administración en la nube

El almacenamiento de los datos se lo realiza tanto a nivel local como en la nube. Se eligió MySQL como sistema gestor de base de datos para el proyecto, de igual manera para el almacenamiento en la nube se tiene a MySQL, desde donde se enviará toda la información almacenada a la interfaz web. La sincronización entre las dos bases de datos se lo realiza mediante un script en Python haciendo peticiones HTTP mediante el método POST.

De acuerdo a la tabla 7 se determinó utilizar el servicio de "000webhost" debido a la compatibilidad con la base de datos que se está utilizando en la Raspberry Pi, la gratuidad y la memoria que posee, características que sirven para la visualización de los datos en la nube.

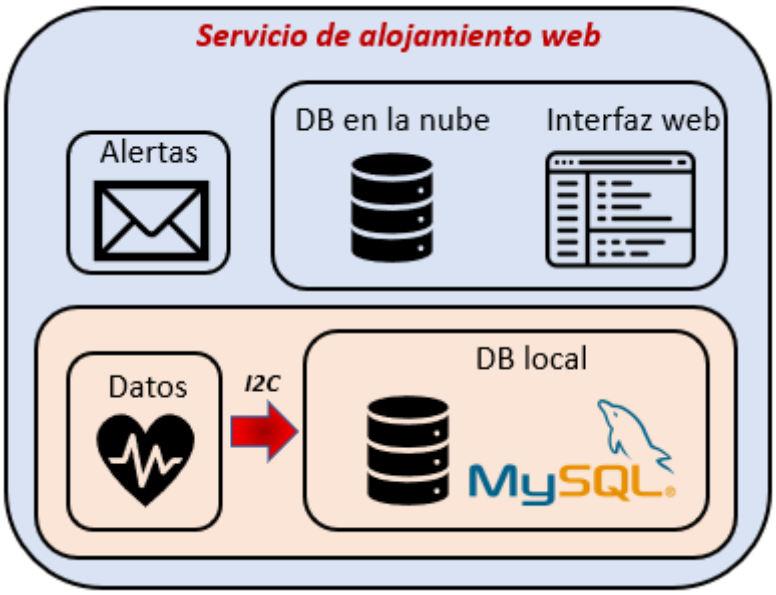


Figura 14. Administración en la nube.
Elaborado por el investigador.

2.2 Métodos

2.2.1 Modalidad de la Investigación

Investigación Aplicada

El presente proyecto se sustentó en una Investigación Aplicada ya que se empleó los conocimientos adquiridos durante la carrera y se realizaron investigaciones que sustentaron la realización del proyecto. Estas investigaciones se lo realizaron tanto en las

bases de datos como en las bibliotecas virtuales, los mismos que fueron proporcionados por la Universidad.

Investigación Bibliográfica

Investigación Bibliográfica, ya que se realizaron búsquedas, recopilaciones e información de datos bibliográficos como, por ejemplo; tesis, artículos científicos, publicaciones en revistas y libros de acuerdo a temas acerca de Inteligencia Artificial y tecnología IoT aplicados a la medicina, para conocer y explorar las fuentes que sirvieron para el proyecto.

Investigación de Campo

Investigación de Campo, ya que el desarrollo del proyecto implicó la participación del paciente y, mediante el sistema electrónico de bioseñales y la aplicación de algoritmos de Inteligencia Artificial, se pudo determinar el nivel de riesgo de contagio de COVID-19.

Investigación Experimental

Investigación Experimental, ya que el éxito del proyecto de investigación requirió de una serie de pruebas para determinar el desempeño del algoritmo de Inteligencia Artificial al realizar un diagnóstico médico de COVID-19.

2.2.2 Recolección de Información

Para lograr la recolección de información se emplearon libros, revistas, fuentes online y proyectos desarrollados, así como guías prácticas y fichas técnicas de componentes electrónicos por lo que se tomaron en cuenta bases de datos confiables que permitieron el desarrollo del proyecto.

2.2.3 Procesamiento y Análisis de Datos

Para el procesamiento y análisis de datos se realizaron mediante los siguientes pasos:

- Revisión de la información recopilada.
- Planteamiento de la propuesta de solución.

- Estudio de las propuestas de solución planteadas para el diagnóstico de COVID-19.
- Control y verificación de los datos obtenidos mediante el testeo del dispositivo.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados

3.1.1 Desarrollo de la propuesta

3.1.1.1 Implementación del sistema electrónico de monitoreo de bioseñales

En el presente proyecto se implementa un sistema electrónico de monitoreo de bioseñales para poder diagnosticar si el paciente está o no contagiado de COVID-19 mediante clasificación de riesgos utilizando aprendizaje automático. La arquitectura de este dispositivo se presenta en la figura 15, el mismo que consta de 3 capas; la capa de sensado, aplicación y de servicio en la nube.

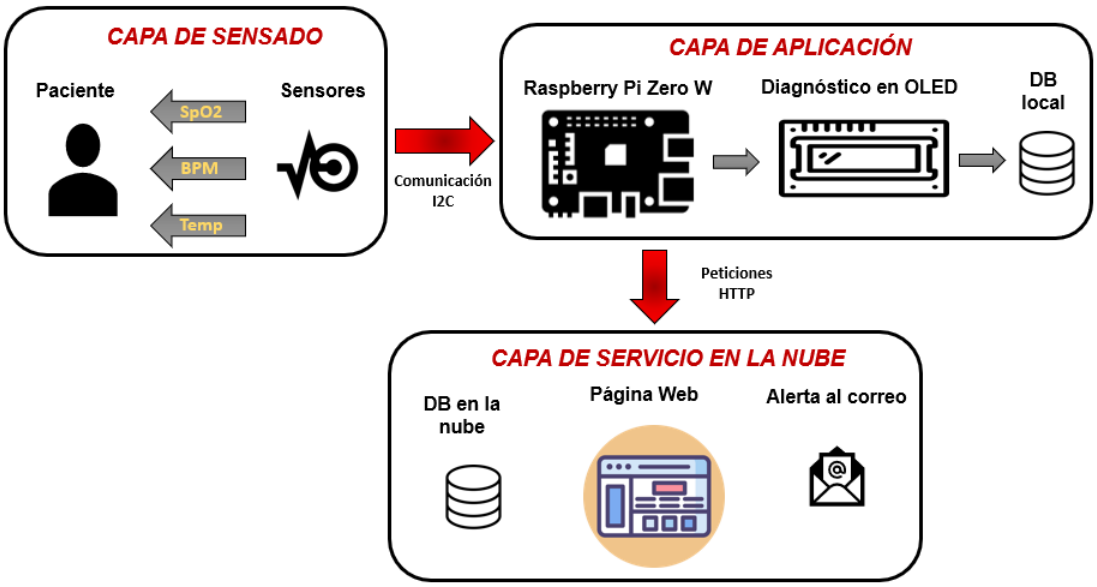


Figura 15. Arquitectura del dispositivo electrónico de monitoreo de bioseñales.
Elaborado por el investigador.

Capa de sensado

Se utiliza una pulsera wereable para la adquisición de los signos vitales en donde se instala el sensor de temperatura corporal y la Raspberry Pi Zero. Se utiliza también el sensor de oximetría, el cual se lo conecta a la pulsera mediante cables debido a que va a obtener los datos desde el dedo índice del paciente. Los sensores que se utilizan son el MAX30100 y MLX90614, estos trabajan con un voltaje de entrada de 3.3 V y se comunican a través del protocolo I2C, además se utilizan dos resistencias de 4.7 KΩ, las mismas que van conectadas desde el puerto SCL y SDA de los sensores hacia el pin que provee 3.3 V de

la Raspberry Pi. Los datos de los signos vitales son enviados a la Raspberry Pi para ser mostrados en una pantalla OLED de 1.3 pulgadas, el mismo que se comunica mediante el protocolo SPI y trabaja con 3.3 V de alimentación.

Capa de aplicación

La capa de aplicación consta de dos elementos, la Raspberry Pi y una pantalla OLED de 1.3 pulgadas. En el primer elemento se realiza todo el procedimiento de almacenamiento de los signos vitales, procesamiento de los mismos junto con otros datos más y diagnóstico de COVID-19 mediante un algoritmo de Inteligencia Artificial. Además, cuenta con conexión a WiFi y a Bluetooth, ambos en la banda de 2.4 GHz.

En la pantalla OLED se muestra la interfaz que se crea mediante la programación en un script de Python, el cual consta de un menú en donde se puede elegir varias opciones entre las cuales está el diagnóstico. Al elegir esta opción mediante el teclado de la pantalla se activan los sensores para tomar los datos de los signos vitales. Luego se ingresa el género del paciente, si está o no vacunado, los factores de riesgo y los síntomas que presentan al utilizar el dispositivo electrónico, todos estos datos son enviados al algoritmo de Inteligencia Artificial para que los analice y finalmente muestra en la pantalla la predicción. Toda la información es almacenada en una base de datos MySQL y es enviada luego al servidor en la nube utilizando peticiones HTTP.

Capa de servicio en la nube

En esta capa se realiza el almacenamiento de todos los datos que se alojan en la Raspberry Pi para ser mostrados en una interfaz web con la finalidad de que el médico a cargo pueda estar informado del estado de salud de sus pacientes y tomar las acciones necesarias para el bienestar de los mismos. Además, el médico puede ser alertado mediante una notificación al correo electrónico si el diagnóstico recomienda revisión médica u hospitalización del paciente. Esto se lo realiza mediante un protocolo de transferencia simple de correo (SMTP).

3.1.1.2 Recursos tecnológicos de red

El funcionamiento correcto del sistema depende de dos protocolos de comunicación, el I2C y el SPI, además para que el médico pueda estar informado de lo que sucede con sus pacientes el dispositivo debe contar con conexión a Internet mediante una red inalámbrica. Se debe tomar en cuenta que el dispositivo es portátil, por lo tanto, puede darse el caso de que se encuentre con redes públicas o privadas, en el último caso se debe

ingresar el nombre y la contraseña de la red, es por esto que se hace uso de la conexión a bluetooth que posee la Raspberry Pi, para que mediante una aplicación alojada en un smartphone pueda conectarse al dispositivo y enviar los datos necesarios para conectarse a la red privada. Estos recursos se presentan en la tabla 11 y tabla 12.

Tabla 11. Recursos de red para la comunicación alámbrica.
Elaborado por el investigador.

Dispositivo	Protocolo	Direccionamiento
Sensor MAX30100	I2C	0x57
Sensor MLX90614	I2C	0x3C
Pantalla OLED	SPI	

Tabla 12. Recursos de red para la comunicación inalámbrica.
Elaborado por el investigador.

Dispositivo	Tecnología	Frecuencia
Almacenamiento en la nube	WiFi 802.11 b/g/n	2.4 GHz
Conexión a red privada	Bluetooth 4.0	2.4 GHz

3.1.1.3 Implementación del circuito de monitoreo de bioseñales

3.1.1.3.1 Circuito para la comunicación entre los sensores y la Raspberry Pi

En la figura 16 se observa el diagrama para la conexión de los sensores en donde se puede observar 2 resistencias conectadas desde los puertos SCL y SDA a 3V, esto se lo realiza debido a que la comunicación entre sensores y la Raspberry Pi puede tornarse lenta a causa de la capacitancia en estos puertos, el cual depende de la cantidad de periféricos que están conectados y también la distancia entre estos y el microordenador. Por lo tanto, para evitar estos problemas se recomienda conectar dos resistencias cuyo valor se lo calcula mediante la siguiente ecuación:

$$R = \frac{V}{I_{DD}} \quad \textbf{(5)}$$

En donde:

- V: Voltaje que suministra la Raspberry Pi.

- I_{DD}: Corriente máxima de operación del sensor.

Se realiza el cálculo para cada sensor:

$$R_{MAX30100} = \frac{3V}{1200\mu A} = 2500\ \Omega$$

$$R_{MLX90614} = \frac{3V}{1.5mA} = 2000\ \Omega$$

El valor más alto, el cual pertenece al sensor MAX30100, es el elegido. Determinando un valor igual o aproximado a 2500 Ω para las resistencias que se van a utilizar.

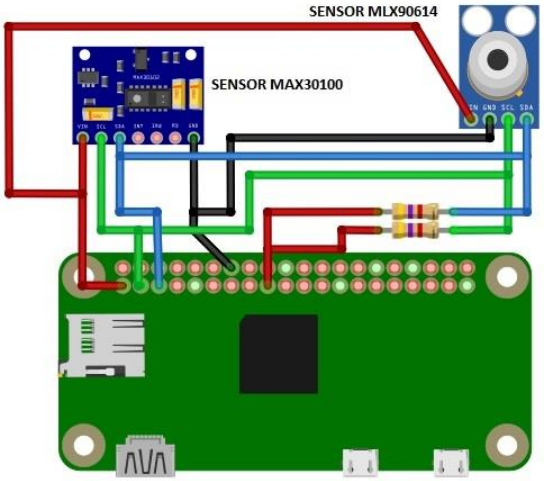


Figura 16. Diagrama para la conexión entre los sensores y la Raspberry Pi.
Elaborado por el investigador.

3.1.1.3.2 PCB del circuito para la conexión entre los sensores y la Raspberry Pi

Para la implementación del circuito en donde serán conectados los sensores con la Raspberry Pi se necesita observar el esquemático del mismo, para evitar conexiones erróneas lo que puede producir graves consecuencias al momento de suministrar de energía al dispositivo. En la figura 17 se muestra el esquemático para el circuito de los sensores.

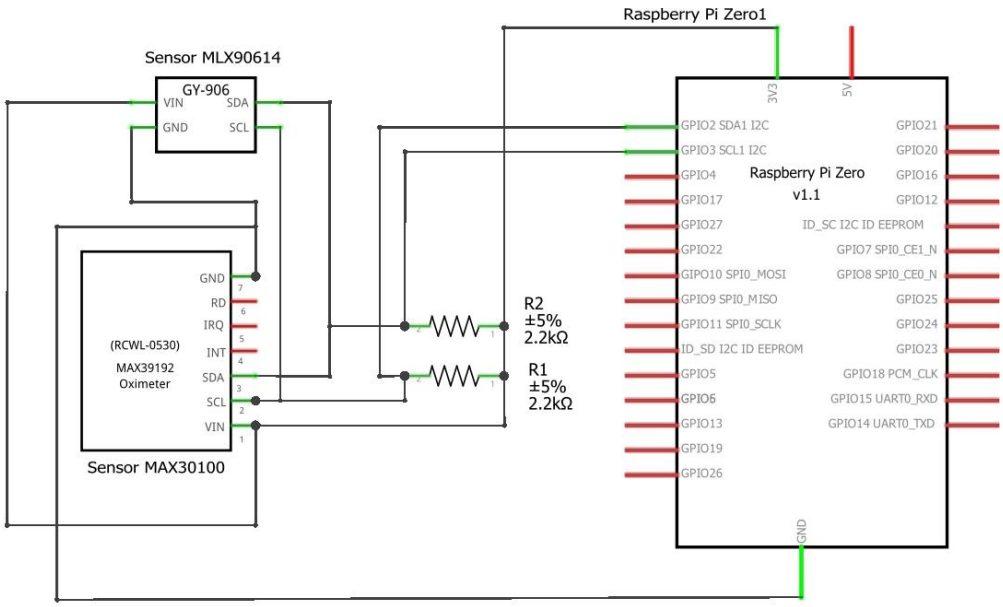


Figura 17. Esquemático para el circuito de los sensores.

Elaborado por el investigador.

En la figura 18 se muestra el diseño de la PCB hecha en Fritzing, el cual es de una sola cara y en la figura 19 se muestra la PCB ya realizada en placa de cobre mediante el método del planchado.

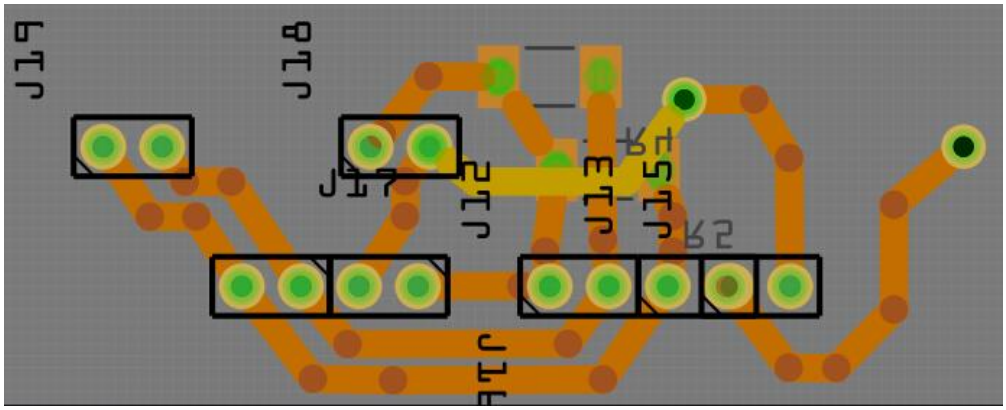


Figura 18. PCB en Fritzing.

Elaborado por el investigador.

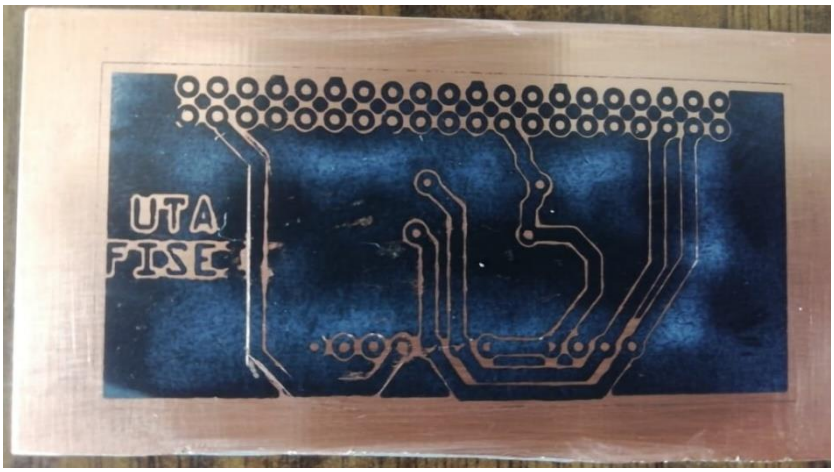


Figura 19. PCB realizada por el método del planchado.
Elaborado por el investigador.

3.1.1.3.3 Diseño del case

El case se lo obtuvo desde la página de la batería PiSugar. Se observa en la figura 20 las entradas para carga de la Raspberry Pi Zero y también para la carga de la batería.

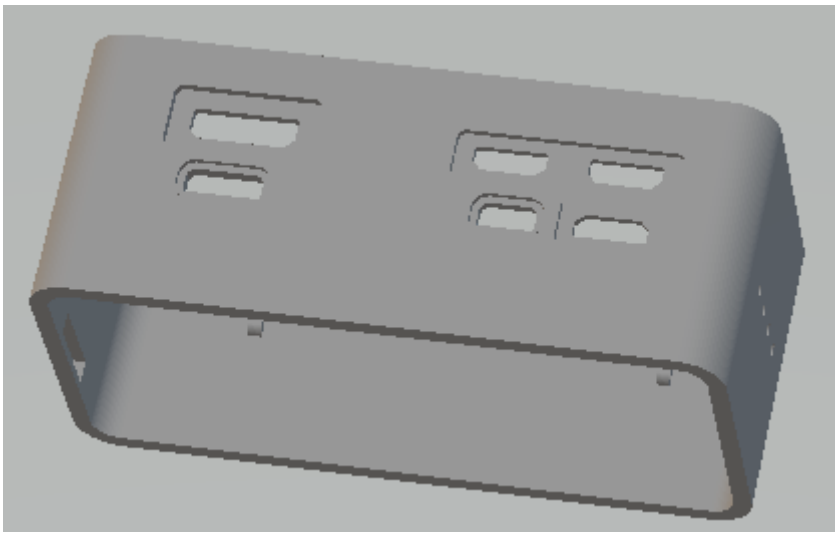


Figura 20. Case del prototipo.
Fuente: [53].

Finalmente, el prototipo queda como en la figura 21 en donde se puede observar la toma de datos de la temperatura corporal mediante el sensor instalado en la pulsera y los datos de pulsioximetría desde el dedo índice a través del sensor que se comunica mediante cable al dispositivo. La ubicación del sensor MAX30100 se determinó de acuerdo a la estabilidad en la lectura de los datos ya que, al ubicar el sensor en la muñeca hubo mucha diferencia en los valores de BPM y SpO2 para cada persona. En cambio, al ubicarlo en el dedo índice los valores fueron estables, esto permite que se pueda calibrar el sensor.

Figura 21. Sistema electrónico en funcionamiento.
Elaborado por el investigador.

3.1.1.4 Acondicionamiento de los sensores

3.1.1.4.1 Ajuste de datos del sensor MAX30100

Filtro de media móvil para la lectura de datos del sensor MAX30100

La media móvil es el cálculo realizado a un conjunto de datos en donde se crean una serie de promedios, es decir son una lista de números en donde cada uno es el promedio de un subconjunto de datos originales, se muestra un ejemplo en la figura 22.

N°	Datos	Periodo = 2	MM
1	14240		
2	14216	Promedio 1-2	14228
3	14208	Promedio 2-3	14212
4	14208	Promedio 3-4	14208
5	14184	Promedio 4-5	14196
6	14280	Promedio 5-6	14232
7	14232	Promedio 6-7	14256
8	14216	Promedio 7-8	14224
9	14160	Promedio 8-9	14188
10	14192	Promedio 9-10	14176
11	14088	Promedio 10-11	14140
12	14200	Promedio 11-12	14144

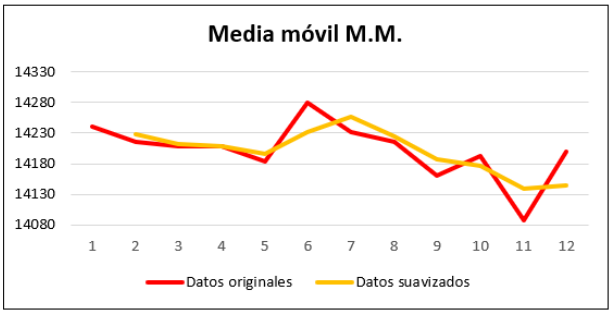


Figura 22. Suavizamiento de datos mediante aplicación un filtro de media móvil con un periodo igual a 2.
Elaborado por el investigador.

A pesar de que el sensor MAX30100 viene calibrado de fábrica, es necesario realizar una serie de pruebas a un grupo de personas para obtener mejores resultados en los datos, tanto del nivel de oxígeno como del pulso cardiaco. Se comienza aplicando un filtro de media móvil a los datos originales obtenidos por el sensor para suavizar los dientes de

sierra que se generan por el ruido presente en la lectura, este filtro se lo hará con un periodo de 4 datos evitando las pérdidas de los mismos. En la figura 23 y figura 24 se muestran los datos que son almacenados en el buffer tanto del LED infrarrojo como del rojo, los mismos que son medidas del pulso cardiaco y SpO2 respectivamente. Se puede apreciar la tendencia que tienen los datos originales en donde se observa el ruido presente en estos.

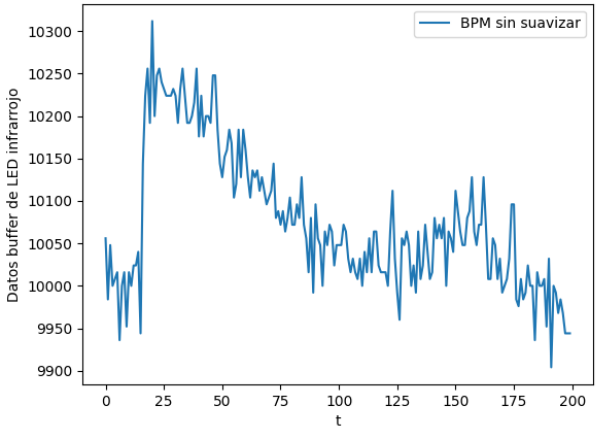


Figura 23. Datos almacenados en el buffer del LED infrarrojo.
Elaborado por el investigador.

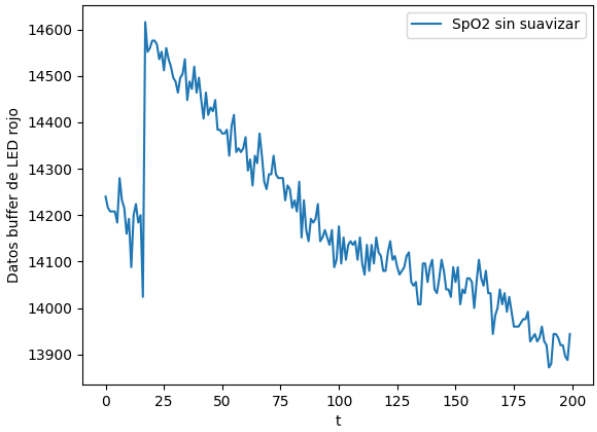


Figura 24. Datos almacenados en el buffer del LED rojo.
Elaborado por el investigador.

En la figura 25 se muestra la función *"media_movil()"* que se crea en el script en donde se realiza el proceso de adquisición de datos del sensor MAX30100. Esta función tiene como argumento la variable *"numbers"* en donde se recibirán los datos originales almacenados en los búferes. Se realiza el cálculo de la media móvil en un rango de 4 datos.

```
def media_movil(numbers):
    promSize = 4
    i = 0

    while i < len(numbers) - promSize + 1:
        porcion = numbers[i : i + promSize]
        promedioPorcion = sum(porcion) / promSize
        i += 1
    try:
        return int((promedioPorcion/100))
    except:
        pass
```

Figura 25. Aplicación de la media móvil para los datos contenidos en los búferes.
Elaborado por el investigador.

Una vez realizado el proceso de la figura 25 se corrigen los datos debido a que, si la lectura de BPM y SpO2 tiene valores menores a 10 los convierte a valores desconocidos o “null”, lo cual puede llegar a producir errores en el valor final del pulso y SpO2 ya que se debe trabajar con valores numéricos para realizar el ajuste mediante el ingreso de una ecuación lineal como se mostrará más adelante.

```
def corrDatos(promedio_bpm,promedio_sp02):
    try:
        if(promedio_bpm<10):
            promedio_bpm = 0
            promedio_sp02 = 0
        return promedio_bpm, promedio_sp02
    except:
        return promedio_bpm, promedio_sp02
```

Figura 26. Corrección de representación de datos.
Elaborado por el investigador.

En la figura 27 y 28 se muestran los datos suavizados mediante la aplicación del filtro, en donde se observa una tendencia estable en la lectura de los datos.

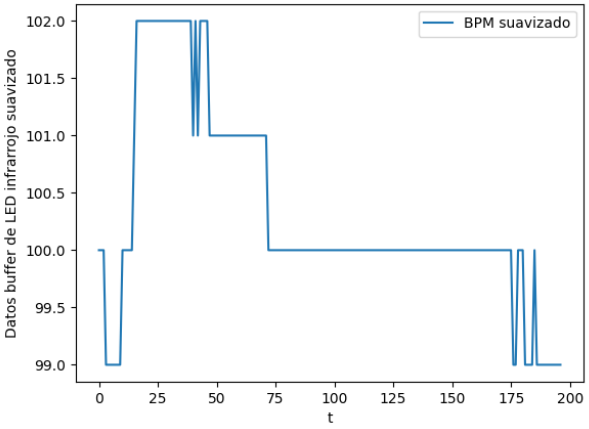


Figura 27. Datos del buffer LED infrarrojo suavizados.

Elaborado por el investigador

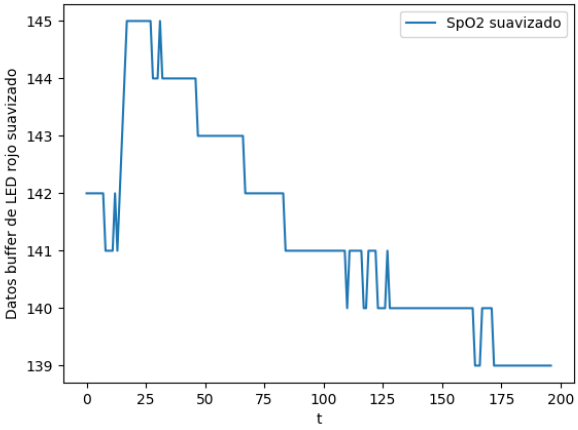


Figura 28. Datos del buffer LED rojo suavizados.

Elaborado por el investigador

Calibración del sensor mediante la obtención de una ecuación lineal

Para esto se requiere adicionalmente un oxímetro comercial con el fin de comparar los valores entre este con el sensor MAX30100, y calibrarlo mediante la obtención de un modelo matemático con respuesta lineal para posteriormente introducirla en el script de toma de datos del sensor. Tomado en cuenta los requerimientos para el uso correcto del oxímetro mencionados anteriormente, se procede a realizar pruebas en 10 personas de distinta edad y sexo, obteniendo los resultados que se muestran en la tabla 13.

Tabla 13. Valores de las mediciones del sensor y el oxímetro.

Elaborado por el investigador.

	Sensor MAX30100		**Oxímetro comercial**	
	Pulso	SpO2	Pulso	SpO2
1	82.96	124.08	78	92
2	84.38	125.52	78	93
3	86.71	127.96	79	93
4	87.48	130.61	79	93
5	89.71	132.82	80	94
6	88.44	130.8	83	91
7	87.02	120.04	87	90
8	82.68	120.09	81	92
9	94.78	135.49	85	93
10	90.73	129.74	81	94

Para interpretar los datos de la tabla 13 de manera adecuada se obtiene el rango de error de los valores de cada fila, de acuerdo a la ecuación (6).

$$E_p = \left|\frac{X_i - X_v}{X_v}\right| * 100\% \quad \textbf{(6)}$$

Donde:

- X_i: $Valor\ real$
- X_v: $Valor\ medido$

Tabla 14. Porcentaje de error para valores de BPM.

Elaborado por el investigador.

	Datos oxímetro comercial (X_i)	**Datos sensor MAX30100 (X_v)**	$X_i - X_v$	$\left\|\frac{X_i - X_v}{X_i}\right\|$	*** 100%**	**% de error**
1	78.00	82.96	-4.96	0.06	100	6.36
2	78.00	84.38	-6.38	0.08	100	8.18
3	79.00	86.71	-7.71	0.10	100	9.76
4	79.00	87.48	-8.48	0.11	100	10.73
5	80.00	89.71	-9.71	0.12	100	12.14
6	83.00	88.44	-5.44	0.07	100	6.55
7	87.00	87.02	-0.02	0.00	100	0.02
8	81.00	82.68	-1.68	0.02	100	2.07
9	85.00	94.78	-9.78	0.12	100	11.51
10	81.00	90.73	-9.73	0.12	100	12.01
					Promedio	7.93

Tabla 15. Porcentaje de error para valores de SpO2.

Elaborado por el investigador.

	Datos oxímetro comercial (X_i)	**Datos sensor MAX30100 (X_v)**	$X_i - X_v$	$\left\|\frac{X_i - X_v}{X_i}\right\|$	*** 100%**	**% de error**
1	92.00	124.08	-32.08	0.35	100	34.87
2	93.00	125.52	-32.52	0.35	100	34.97
3	93.00	127.96	-34.96	0.38	100	37.59
4	93.00	130.61	-37.61	0.40	100	40.44
5	94.00	132.82	-38.82	0.41	100	41.30
6	91.00	130.80	-39.80	0.44	100	43.74
7	90.00	120.04	-30.04	0.33	100	33.38
8	92.00	120.09	-28.09	0.31	100	30.53
9	93.00	135.49	-42.49	0.46	100	45.69
10	94.00	129.74	-35.74	0.38	100	38.02
					Promedio	38.05

De acuerdo a la tabla 14 y tabla 15 se puede deducir que existe un promedio de error porcentual significativo, razón por la cual es muy necesario realizar un ajuste de estos valores para obtener datos confiables del sensor. Se hace uso de Excel en donde se ingresan los datos de la tabla 13 para obtener una gráfica y trazar la curva ajustada

mediante una línea fina y continua. Cabe mencionar que es necesario emplear un gráfico de dispersión en donde se representan cada uno de los datos mediante puntos discretos.

Las ecuaciones que se muestran en la figura 29 y 30 son ecuaciones lineales que representan la línea de ajuste de tendencia de los datos. Se obtienen a partir del gráfico de dispersión realizada en Excel y determinan el grado de dependencia de los valores en el eje X e Y, con la finalidad de predecir los valores del oxímetro comercial en comparación con los valores del sensor MAX30100. De esta manera se realiza el ajuste de los datos que se obtienen mediante el sensor MAX30100 los mismos que, mediante la ecuación lineal, serán equivalentes a los valores del oxímetro comercial.

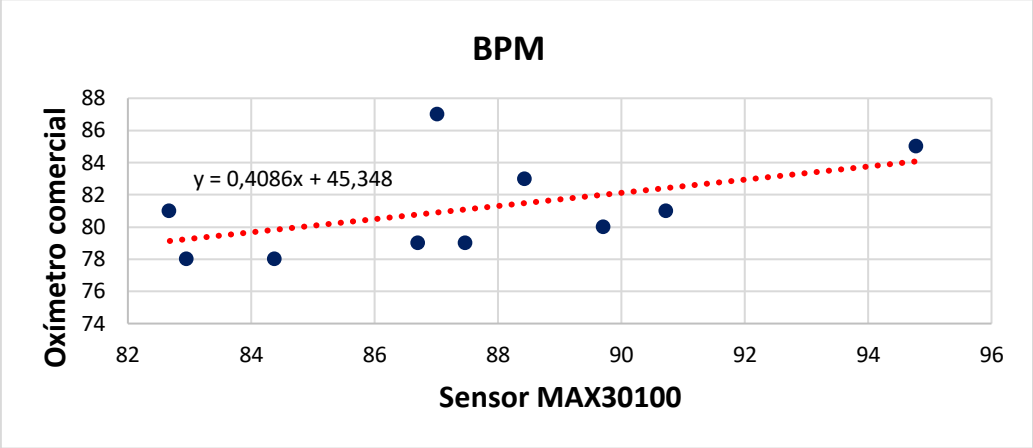


Figura 29. Ecuación lineal para la calibración de los valores de BPM.
Elaborado por el investigador.

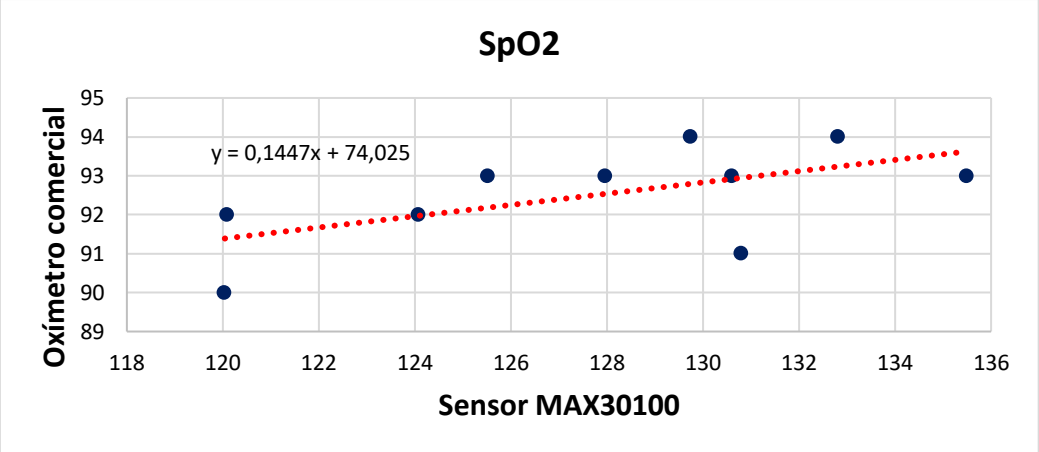


Figura 30. Ecuación lineal para la calibración de los valores de SpO2.
Elaborado por el investigador.

Una vez obtenidas las ecuaciones se procede a ingresarlas en el script que realiza la función de extracción de BPM y SpO2. En la figura 31se puede observar que la variable x de la ecuación lineal toma el valor de la media de los datos obtenidos sin ajustar.

```
for i in spo2List:
  suma2 = suma2+i
  suma2 = suma2/len(spo2List)
  ajusteSpo2  = round((0.1447*suma2+74.025),2)
  print("SpO2 con ajuste: ",ajusteSpo2)
```

Figura 31. Ingreso de la ecuación lineal para el nivel de oxígeno.
Elaborado por el investigador.

```
for i in pulsosList:
  suma1 = suma1+i
  suma1 = round(suma1/len(pulsosList),2)
  ajustePulsos = round((0.4086*suma1+45.348),2)
  print("Pulsos con ajuste: ",ajustePulsos)
```

Figura 32. Ingreso de la ecuación lineal para el pulso cardiaco.
Elaborado por el investigador.

Se realiza las respectivas pruebas entre el sensor MAX30100 calibrado y el oxímetro comercial para realizar la comparación entre estos dos valores, los mismos que se observan en la tabla 16 en donde se puede observar que los datos que se obtienen mediante los modelos matemáticos se ajustan a los valores que provee el oxímetro comercial.

Tabla 16. Datos ajustados con el modelo matemático.
Elaborado por el investigador.

	Sensor MAX30100 (Ajustado)		**Oxímetro comercial**	
	Pulso	SpO2	Pulso	SpO2
1	79	91	79	92
2	80	93	75	94
3	80	92	75	94
4	78	93	75	93
5	78	93	75	93
6	78	93	74	93
7	78	94	76	92
8	77	94	74	93
9	78	93	76	92
10	79	94	78	94

Para corroborar que los datos medidos por el sensor MAX30100 son confiables, se hace uso del diagrama de "Blant-Altman" con el objetivo de comparar las mediciones entre el sensor y el oxímetro comercial. En la figura 33, para los valores de frecuencia cardiaca, se observa un sesgo debido a que la línea promedio está por debajo del cero, esta línea está en -2.8 lo que quiere decir que el prototipo mide aproximadamente 2.8 unidades más que el equipo médico con respecto a la frecuencia cardiaca. Los datos se encuentran dentro del intervalo de confianza del 95 % y la mayoría de los datos se encuentran cerca del promedio. Lo mismo sucede en la figura 34, con la diferencia de que la línea promedio se encuentra cerca del cero, lo que significa que los valores para el nivel de oxígeno en la sangre son similares a la del oxímetro comercial.

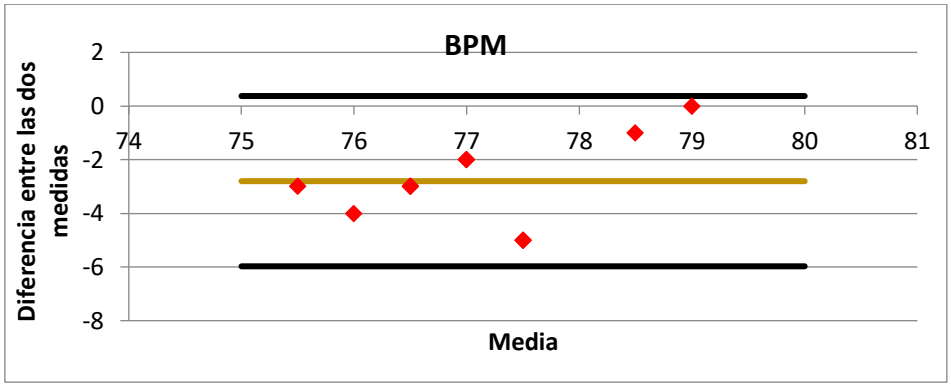


Figura 33. Diferencias de mediciones de BPM mediante gráfico de Blant Altman. Elaborado por el investigador.

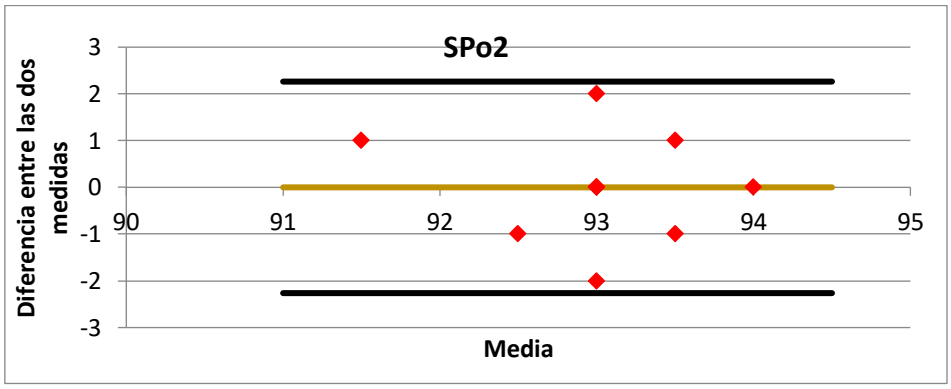


Figura 34. Diferencias de mediciones de SPo2 mediante gráfico de Blant Altman. Elaborado por el investigador.

Finalmente, se puede deducir como será el proceso de adquisición de datos por parte del sensor MAX30100 mediante un diagrama de flujo, el mismo que se muestra en la figura 35.

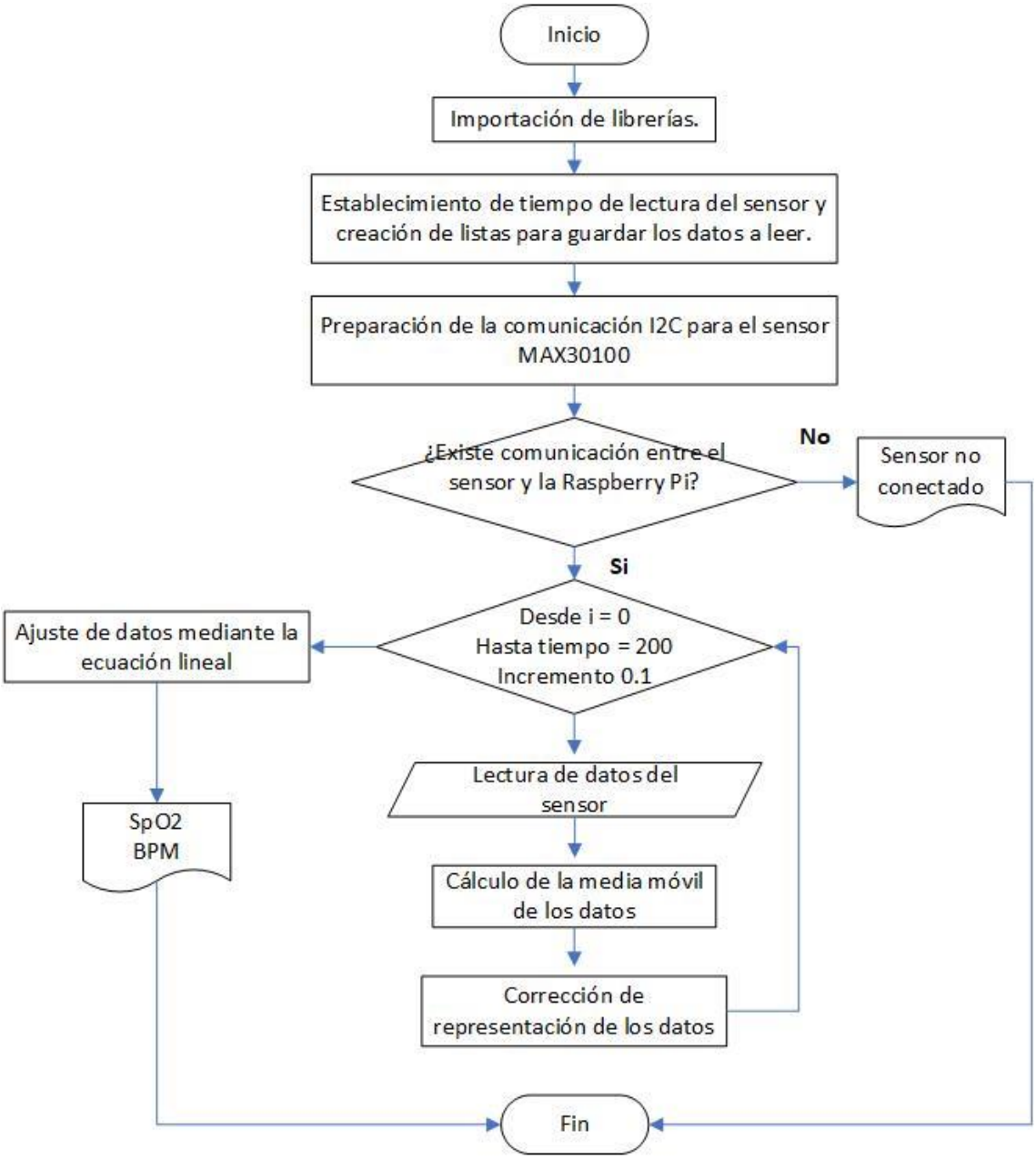


Figura 35. Diagrama de flujo de adquisición de datos por parte del sensor MAX30100. Elaborado por el investigador.

3.1.1.4.2 Ajuste de datos del sensor MLX90614

El procedimiento para la calibración del sensor MLX90614 es el mismo que se hizo para el sensor MAX30100, por lo tanto, se realiza las mediciones respectivas con el objetivo de encontrar la ecuación lineal para calibrar el sensor.

Tabla 17. Mediciones entre el sensor MLX90614 y un termómetro comercial.
Elaborado por el investigador.

	Sensor MLX90614	**Termómetro comercial**
	Temperatura °C	Temperatura °C
1	32.69	36.2
2	33.45	36.4
3	33.88	36.4
4	33.48	36.3
5	32.77	36.5
6	33.24	36.5
7	33.12	36.3
8	32.89	36.3
9	33.21	36.3
10	33.82	36.5

A partir de los datos que se presentan en la tabla 17 se obtiene el gráfico de dispersión. Tal y como se realizó en el juste de datos para el sensor MAX30100, la ecuación de la figura 36 ayuda a que los valores obtenidos por el sensor MLX90614 tiendan a los valores del termómetro comercial. Esta ecuación se lo ingresa al respectivo script en donde la variable "x" tomará los valores originales del sensor, produciendo así el ajuste o calibración del sensor de temperatura corporal.

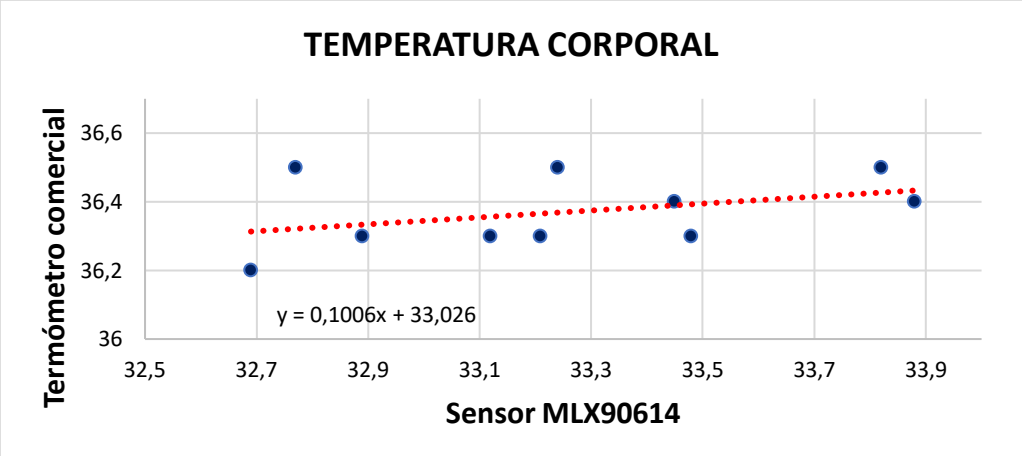


Figura 36. Gráfico de dispersión de los datos de temperatura.
Elaborado por el investigador.

Se procede a ingresar al script del sensor de temperatura la ecuación lineal obtenida como se observa en la figura 37.

```
for i in tempList:
  suma = suma + i
  medTemp = suma/len(tempList)
  ajusteTemp = round((0.1006*medTemp+33.026),2)
```

Figura 37. Ingreso de la ecuación lineal para la temperatura.
Elaborado por el investigador.

Una vez ingresado la ecuación lineal en el script, se realizan las respectivas pruebas entre el sensor y el termómetro comercial tal y como se muestran en la tabla 18.

Tabla 18. Medidas entre el sensor ajustado y el termómetro comercial.
Elaborado por el investigador

	Sensor MLX90614	**Termómetro comercial**
	Temperatura	Temperatura
1	36.4	36.4
2	36.3	36.3
3	36.4	36.4
4	36.3	36.2
5	36.3	36.4
6	36.3	36.4
7	36.3	36.5
8	36.3	36.4
9	36.4	36.5
10	36.4	36.4

Para observar la diferencia entre las medidas de los dos dispositivos se grafica el diagrama de “Blant Altman”, en donde se observa que existen 6 puntos discretos debido a que de acuerdo a la tabla 18, existen 4 mediciones iguales entre los dos dispositivos y por lo tanto su diferencia es igual a cero, lo que indica que van a existir 4 valores en la misma coordenada, en este caso va a ser la coordenada (36.4, 0). De igual manera se observa que estos están dentro del intervalo de confianza.

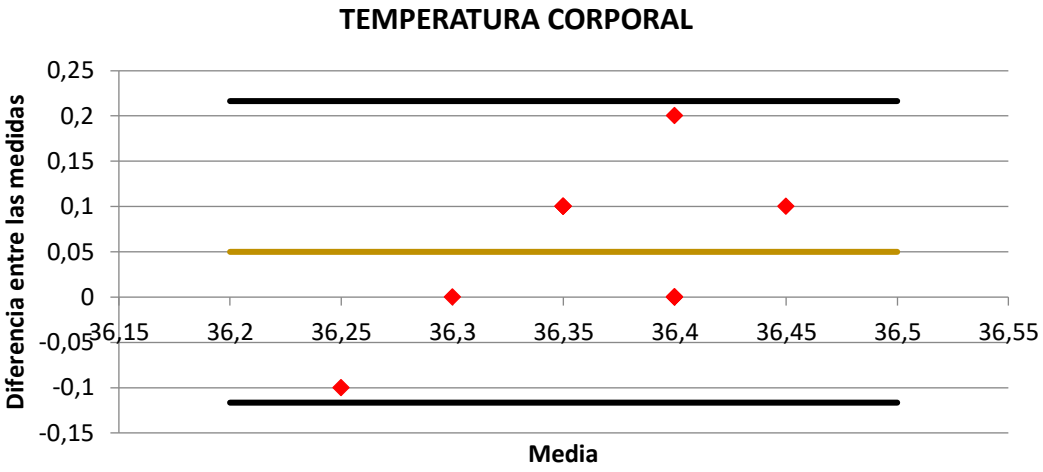


Figura 38. Diferencias de mediciones de temperatura mediante gráfico de Blant Altman. Elaborado por el investigador.

Con el sensor calibrado se procede a mostrar el diagrama de flujo de la figura 38 en donde se observa el procedimiento para la toma de datos por parte del sensor de temperatura corporal.

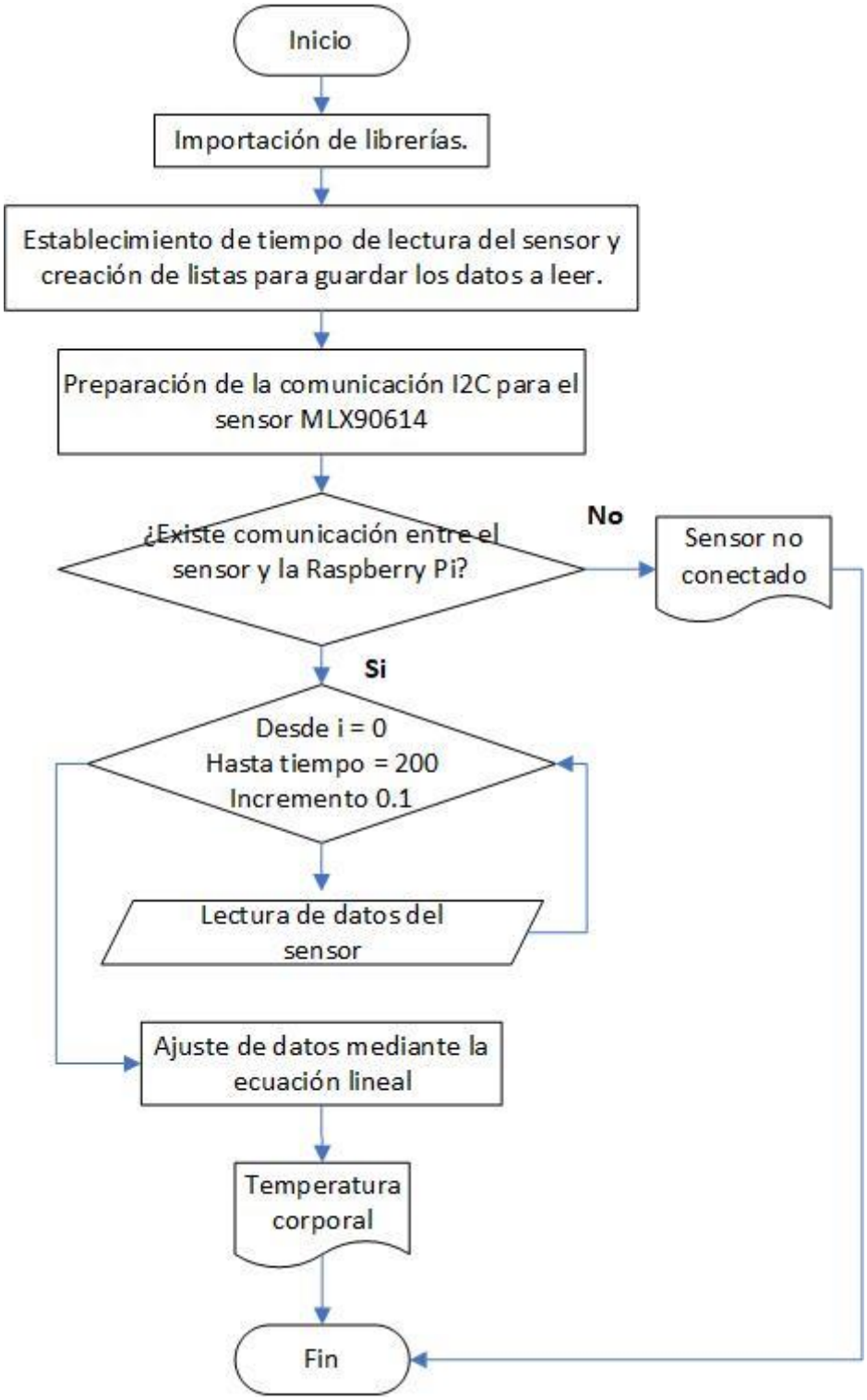


Figura 39. Diagrama de flujo de adquisición de datos por parte del sensor MLX90614. Elaborado por el investigador.

3.1.1.5 Diseño de la interfaz para la pantalla OLED

Se realiza un diagrama de flujo como se muestra en la figura 39 para la creación de la interfaz, el cual se mostrará en la pantalla OLED. Se observa en el flujograma que la creación parte de la importación de las librerías necesarias para la comunicación entre la pantalla y la Raspberry Pi, y para que la interfaz se observe en la pantalla. El menú principal consta de 6 opciones, Diagnóstico, Signos vitales, Información del sistema, Conectar a red y Apagar. Cabe mencionar que el diagnóstico es con respecto al nivel de riesgo de COVID-19 que puede tener el paciente, para que posteriormente el médico pueda dar su diagnóstico indicando si el paciente puede o no tener COVID-19. Se crean además 6 contadores los mismos que se describen a continuación:

- "cursor"

  Este contador se inicia en cero y sirve para indicar en qué posición de la pantalla se encuentra el cursor con la finalidad de que el desplazamiento se visualice. Toma valor de acuerdo a la posición en que se encuentra. Cabe mencionar que en la pantalla se muestra texto en 7 líneas, las mismas que van cambiando de acuerdo a la selección que se haga por medio del teclado de la pantalla.

- "página"

  Este contador se inicia en cero e indica en qué opción del menú creado se encuentra. Debido a que en la pantalla se muestra un máximo de 7 opciones del menú, la página siguiente será 7 dado a que el conteo empieza desde cero, la siguiente página será la opción 14 del menú y así sucesivamente.

- "menú"

  Este contador sirve para mantener en la misma posición el cursor.

- "selección"

  Este contador toma el valor de 1 cuando se presiona la tecla derecha (KEY_LEFT) indicando que se ha seleccionado una opción de la página principal.

- "diag"

  Este contador toma el valor de 1 al momento de presionar la tecla KEY1, indicando que se ha seleccionado una opción de las páginas siguientes al menú principal.

- "envío"

Este contador toma el valor de 1 si se presiona la tecla KEY3, esta tecla se lo utiliza para mostrar el nivel de riesgo de COVID-19 cuando ya se han ingresado los datos respectivos para el mismo.

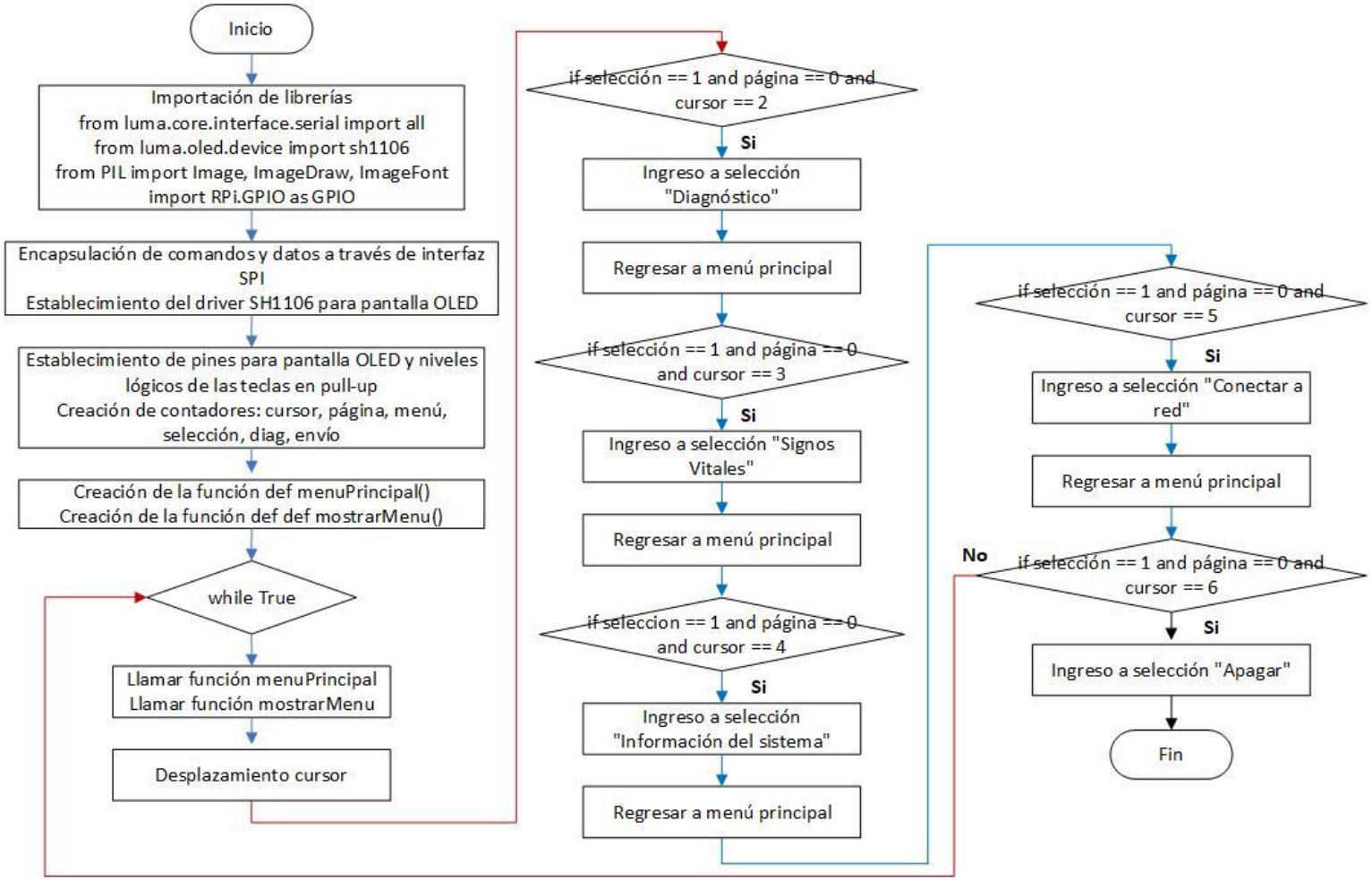


Figura 40. Flujograma de la interfaz gráfica para la pantalla OLED.

Elaborado por el investigador.

El desplazamiento del cursor se lo hace mediante el diagrama de flujo que se presenta en la figura 40.

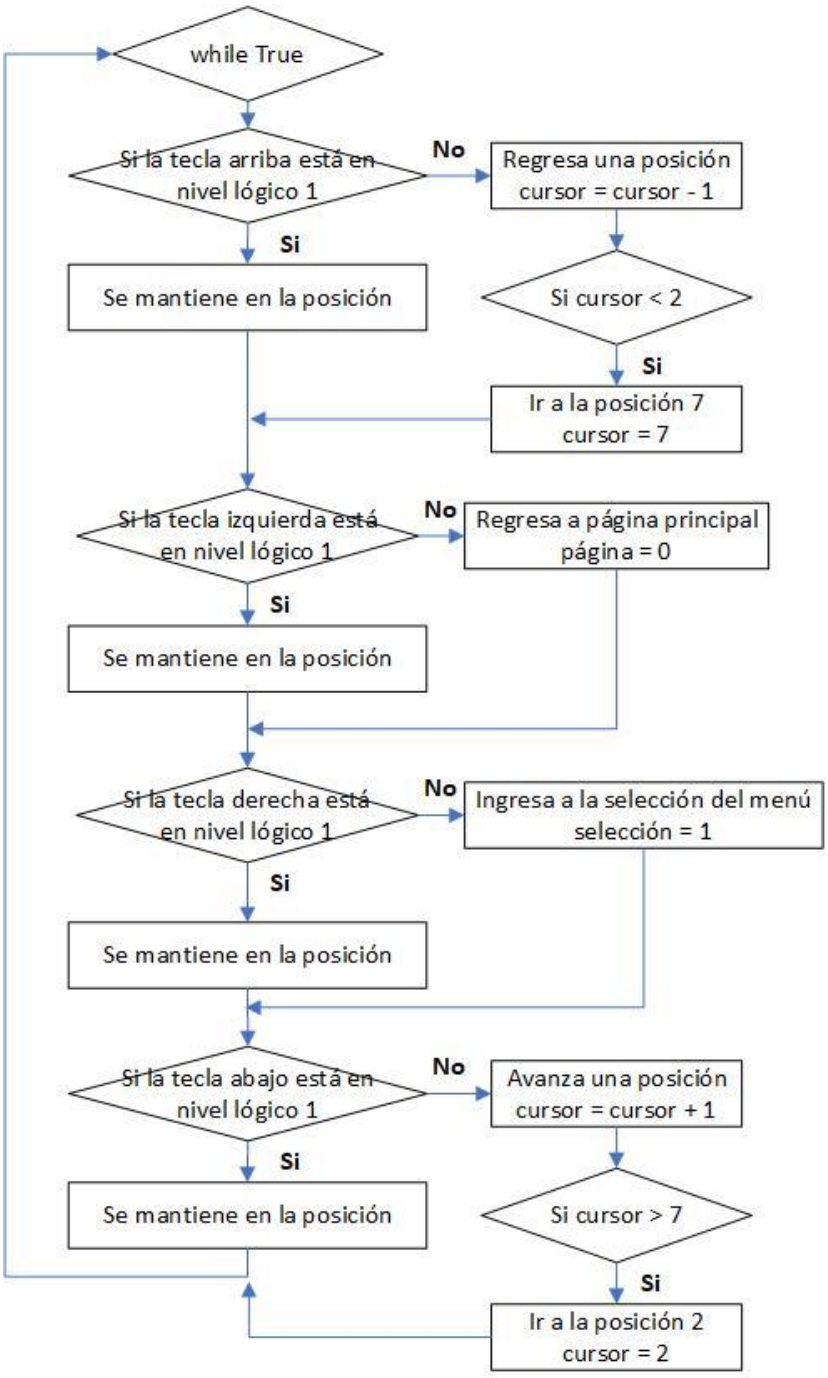


Figura 41. Diagrama de flujo para el movimiento del cursor.
Elaborado por el investigador.

Se observa en el diagrama de flujo de la figura 41 los parámetros que se ingresan, tanto para información como para el diagnóstico, estos fueron tomados en cuenta de acuerdo al criterio médico de la Dra. Gaviria Bolaños Angélica Paola.

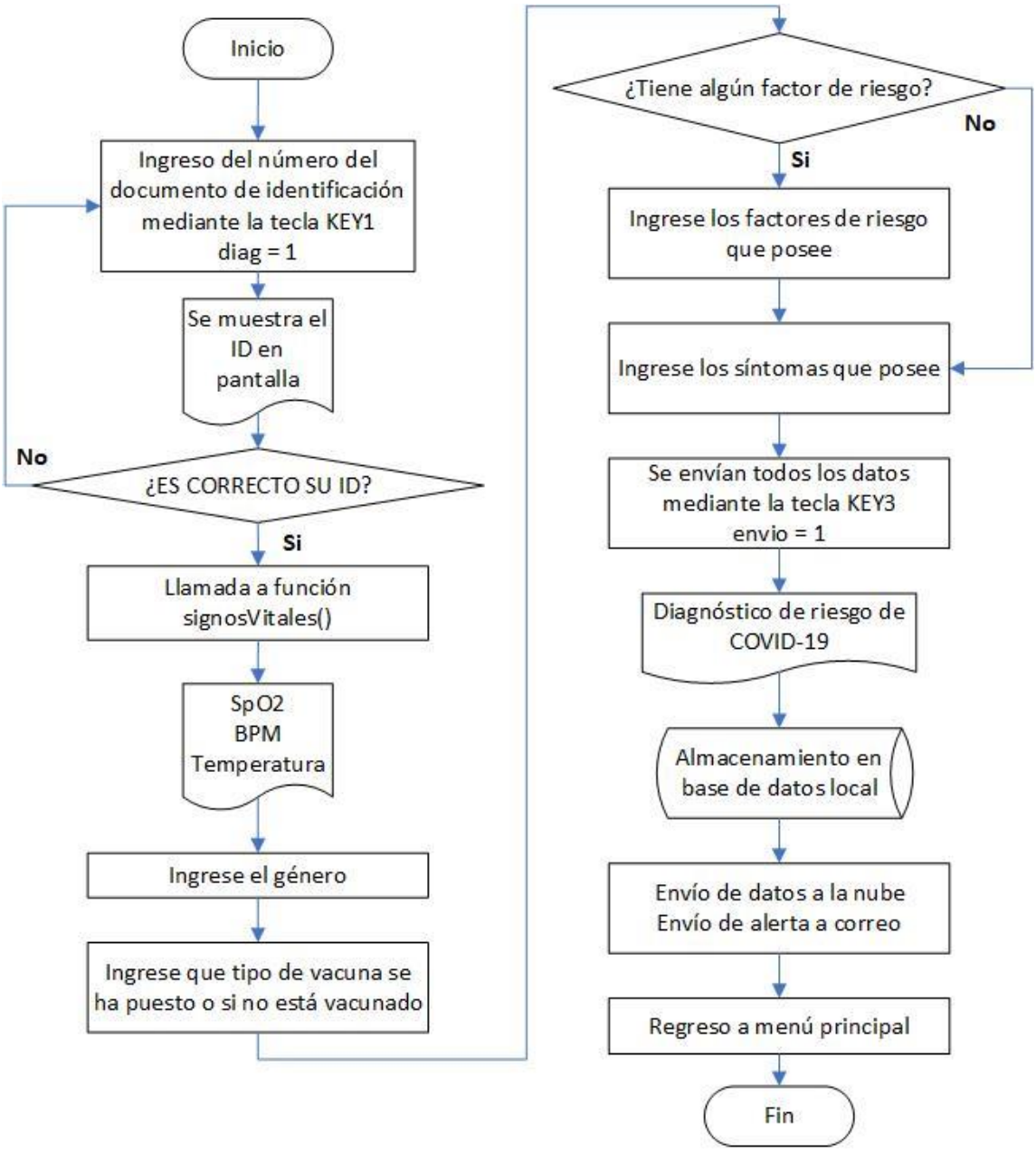


Figura 42. Diagrama de flujo para el diagnóstico de riesgo de COVID-19.
Elaborado por el investigador.

3.1.1.6 Implementación de la etapa de sensorización en el dispositivo

Se crean dos scripts, *"sensor_max30100.py"* y *"sensor_mlx90614.py",* uno para la activación del sensor de SpO2 y el pulso cardiaco y el otro para la activación del sensor de temperatura corporal respectivamente. En cada uno de estos se establecen las funciones que van a retornar los valores de cada medida, con la finalidad de que estos puedan ser enviados al script principal *"main.py"* mediante la importación de estas funciones.

```
def varPulso():
    return ajustePulsos

def varSpo2():
    return ajusteSpo2
```

```
def varTemp():
    return ajusteTemp
```

En el script principal se importa las funciones creadas para adquirir los datos de BPM, SpO2 y temperatura corporal del paciente enviadas desde los scripts *"sensor_max30100.py"* y *"sensor_mlx90614.py"*. Esto se lo realiza mediante la sintaxis *" from file import function"*.

```
from sensor_mlx90614 import sensor_mlx90614, varTemp
from sensor_max30100 import sensor_max30100, varPulso, varSpo2
```

Se crea una función llamada *"signosVitales()"* en donde se recopila los valores de cada sensor y son procesados para ser enviados a una función que realiza el diagnóstico. Se utiliza la clase *"Thread"* para activar al mismo tiempo los dos sensores y minimizar el tiempo de sensado ya que si no se lo hace, primero se ejecutará un sensor y luego de finalizar se ejecutará el otro sensor aumentando el tiempo de diagnóstico. Mientras los sensores están tomando datos, se muestra una leyenda en la pantalla cuyo texto es "Tomando los signos vitales" y, debido a que esto se está ejecutando en segundo plano se establece un tiempo de espera de 23 segundos para que los datos sean recibidos correctamente y puedan ser enviados a las variables que se crearán luego. Se determina este tiempo debido a que la toma de datos de cada sensor está establecida en 20 segundos, por lo tanto, se añade 3 segundos más para evitar problemas en recibir estos datos.

```
def signosVitales():
  hilo1 = threading.Thread(target=sensor_max30100)
  hilo2 = threading.Thread(target=sensor_mlx90614)
  hilo1.start()
  hilo2.start()

  font10 = ImageFont.truetype('/home/santiago/Documentos/Tesis/pruebas_interfaz/Font.ttf',11)

  with canvas(device) as draw:
    draw.rectangle(device.bounding_box, outline="white", fill="black")
    draw.text((10,10), 'Tomando los signos', font=font10, fill = "white")
    draw.text((40,25), 'vitales', font=font10,  fill = "white")

  time.sleep(23)
```

Los datos se visualizan en la pantalla OLED del dispositivo tal y como se ve en la figura 42.

Figura 43. Visualización de signos vitales en la pantalla OLED.
Elaborado por el investigador.

3.1.1.7 Selección del algoritmo de Inteligencia Artificial

3.1.1.7.1 Preparación del dataset

Tal y como se describe en el diagrama de la figura 1 se comienza con la etapa de preparación de los datos, para esto se debe contar con un dataset el cual contenga la información etiquetada, es decir con sus clases en este caso con las clases “ambulatorio”, “revisión médica” y “hospitalización”.

El dataset se lo realizó de la siguiente manera:

- El análisis de los algoritmos médicos para la clasificación del nivel de riesgo de COVID-19 encontrados en la página web de la Organización Panamericana de Salud y del Instituto Mexicano del Seguro Social, los cuales se muestran en el anexo B y C.

- Datos proporcionados de pacientes con COVID-19 almacenados en el computador de la doctora que ayudó con la realización de este proyecto.

- Dataset proporcionado por investigadores de la Universidad de Guayaquil en un curso de Procesamiento de Lenguaje Natural con Python.

El dataset tiene cuatro columnas, género, vacuna, síntomas y casos, estos son transformados a datos que sirvan para la clasificación, en el caso del género se puede observar en la figura 43 que existen iniciales F y M que corresponden a Femenino y Masculino respectivamente, por lo tanto, se transforman estos iniciales a palabras que tengan sentido semántico. De igual manera sucede con la columna "vacuna" en donde se está especificando que tipo de vacuna posee, pero para la predicción se tomará en cuenta únicamente si está o no vacunado.

genero	vacuna	sintomas leves	intensidad
F	Pfizer	perdida de gusto y olfato	leve
M	Sinovac	dolor leve de espalda y fiebre	leve
M	No tenia ninguna vacuna puesta	dolor de pecho	leve
F	No tenia ninguna vacuna puesta	dolor de cabeza tos seca y fiebre no tenia gusto ni olfato	leve
F	Pfizer	sin sabor	leve
F	No tenia ninguna vacuna puesta	solo no tenia sabor	leve
M	No tenia ninguna vacuna puesta	perdida de olfato total	leve
F	No tenia ninguna vacuna puesta	fiebre, perdida del olfato y gusto	leve
F	Sinovac	perdida de gusto y olfato, dolor de cabeza, fatiga, fiebre	medio

Figura 44. Columnas del dataset a utilizar.
Elaborado por el investigador.

La columna de síntomas se puede decir que es la más importante, ya que posee los datos con más peso y que ayudarán en la eficiencia del algoritmo al clasificar el nivel de riesgo de COVID-19. Se observa en la quinta fila de esta columna el texto "sin sabor", debido a que desde el dispositivo se van a agregar palabras específicas mediante selección, por lo tanto, se debe modificar este texto tal y como se observa en la figura 44.

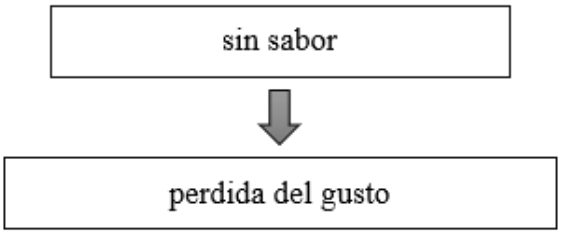


Figura 45. Modificación de la sintáxis de los datos.

Elaborado por el investigador.

Una vez transformado el texto, es necesario también combinar las 3 primeras columnas para que el algoritmo de Machine Learning pueda considerar toda la información. Finalmente, el dataset queda como se muestra en la figura 45.

index	sintomas	intensidad
0	femenino, perdida del gusto, perdida del olfato	ambulatorio
1	masculino, dolor de espalda, fiebre	ambulatorio
2	masculino, no vacuna, dolor de pecho	ambulatorio
3	femenino, perdida del gusto	ambulatorio
4	femenino, no vacuna, perdida del gusto	ambulatorio
5	masculino, no vacuna, perdida del olfato	ambulatorio
6	femenino, no vacuna, fiebre, perdida del olfato, perdida del gusto	ambulatorio
7	masculino, aun nada	ambulatorio
8	dolor de articulaciones, fiebre	ambulatorio

Figura 46. Datos procesados en Excel.

Elaborado por el investigador.

Una vez procesados los datos mediante Excel, se realiza una exploración de los mismos en Python para conocer los tipos de datos que contienen y si existen celdas en blanco que generen errores en el proceso de vectorización de los mismos. En la figura 46 se observan que las dos columnas tienen datos de tipo “object” y no existen datos nulos.

```
datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 430 entries, 0 to 429
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sintomas    430 non-null    object
 1   intensidad  430 non-null    object
dtypes: object(2)
memory usage: 6.8+ KB
```

```
datos_null = datos.isnull().any().sum()
datos_null
```

```
0
```

Figura 47. Exploración de datos del dataset.

Elaborado por el investigador.

3.1.1.7.2 Balanceo de datos

En la figura 47 se puede observar la cantidad de casos que existen para cada categoría, de estas la que más tiene es la categoría “leve”, por lo tanto, se procede a modificar la distribución original del dataset mediante el método de *“RandomOverSampler”* para replicar los casos que contienen la categoría minoritaria. Se realiza esto para evitar que el algoritmo tienda a clasificar más la clase con mayores casos. Por ejemplo, de acuerdo a la figura 47, la predicción se centraría más en la clase “ambulatorio” ya que contiene 226 casos lo que generaría falsas predicciones ya que casi no se tomaría cuenta la clase “hospitalización”, el cual tiene 20 casos.

```
datos.value_counts(['intensidad'])
```

```
intensidad
ambulatorio        226
revision_medica    184
hospitalizacion     20
dtype: int64
```

Figura 48. Cantidad de casos en cada categoría.

Elaborado por el investigador.

En la figura 48 se observa el procedimiento para el balanceo de los casos, en donde se tiene 226 casos para cada categoría.

```
from imblearn.over_sampling import RandomOverSampler
balanceo = RandomOverSampler()
datosBal, datosBal['intensidad'] = balanceo.fit_resample(datos[['sintomas']],datos[['intensidad']])
datosBal.value_counts(['intensidad'])
```

```
intensidad
ambulatorio        226
hospitalizacion    226
revision_medica    226
dtype: int64
```

Figura 49. Casos balanceados mediante "RandomOverSampler".

Elaborado por el investigador.

3.1.1.7.3 Entrenamiento de datos

El entrenamiento se lo hace con el método *"train_test_split"*. Este método recibe como argumentos un conjunto de datos, el tamaño de muestra para el testeo y un estado aleatorio, este último argumento se recomienda utilizar un número entero con la finalidad de que al ejecutar varias veces el código no se generen nuevos valores para el conjunto de datos de entrenamiento y de prueba, lo cual puede afectar al rendimiento del modelo final.

- Conjunto de datos: datos balanceados.
- Tamaño de la muestra para el testeo: 10 % de la cantidad del conjunto de datos balanceados.
- Estado aleatorio: 42.

Se observa en la siguiente imagen el número de muestras que toma para el entrenamiento y testeo de acuerdo al porcentaje que se estableció en el parámetro respectivo.

```
entrenamiento, testeo = train_test_split(datosBal, test_size=0.10, random_state=42)
entrenamiento.value_counts("intensidad")
```

```
intensidad
hospitalizacion    208
ambulatorio        202
revision_medica    200
dtype: int64
```

```
testeo.value_counts("intensidad")
```

```
intensidad
revision_medica    26
ambulatorio        24
hospitalizacion    18
dtype: int64
```

Figura 50. Entrenamiento de los datos.

Elaborado por el investigador.

Cabe mencionar que debido a la cantidad de datos que se tienen es recomendable utilizar una proporción 90:10 para el entrenamiento y testeo respectivamente con la finalidad de que el modelo pueda predecir eficientemente. Luego se procede a separar las columnas de los datos entrenados, “train_x” y “train_y” serían las columnas de síntomas e intensidad respectivamente del conjunto de datos para el entrenamiento. De igual manera sucede con el conjunto de datos para el testeo.

```
train_x, train_y = entrenamiento['sintomas'], entrenamiento['intensidad']
test_x, test_y = testeo['sintomas'],testeo['intensidad']
```

3.1.1.7.4 Codificación de los datos

Debido a que los modelos que se van a utilizar son matemáticos, es decir, utilizan datos numéricos, surge la necesidad de convertir los ejemplos los cuales son textos, en vectores numéricos. Esto se lo realiza mediante la utilización del módulo de sklearn, *“TfidVectorizer”*, el cual se encarga de definir la relevancia de una palabra en el texto tal y como se lo definió anteriormente. La codificación se lo realiza a la columna de síntomas tanto del entrenamiento como del testeo, debido a que esta va a ser la que se va a utilizar en el algoritmo de Inteligencia Artificial para realizar la clasificación.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
train_x_vector = tfidf.fit_transform(train_x)
test_x_vector = tfidf.transform(test_x)
```

```
print(train_x_vector)
```

```
  (0, 42)       0.37080657996874405
  (0, 80)       0.28089102038578634
  (0, 106)      0.31803537587711644
  (0, 166)      0.40949021468645513
  (0, 86)       0.37858095919596313
  (0, 270)      0.3756313199220596
```

```
print(test_x_vector)
```

```
  (0, 361)      0.12301002685473125
  (0, 270)      0.42186644215494323
  (0, 253)      0.23761553698413748
  (0, 239)      0.1579061911576508
  (0, 166)      0.22994645388313595
  (0, 145)      0.18047466074821
```

Figura 51. Vectorización del conjunto de datos de entrenamiento y prueba.

Elaborado por el investigador.

3.1.1.7.5 Construcción y evaluación del algoritmo

Realizado el proceso de codificación queda hacer pruebas con los algoritmos de Inteligencia Artificial para que, mediante los métodos de evaluación se elija el mejor. Se utilizarán los siguientes algoritmos:

- Support Vector Machine

Este método se lo realiza mediante la función *"SVC()"*, la función *"fit"* se lo usa para poder encontrar los valores óptimos en el entrenamiento de este algoritmo.

```
lr = LogisticRegression()
lr.fit(train_x_vector,train_y)
```

- Árbol de decisiones

Este método se lo realiza mediante la función *"DecisionTreeClassifier()"*.

```
dec_tree = DecisionTreeClassifier()
dec_tree.fit(train_x_vector, train_y)
```

- Naive Bayes

Este método se lo realiza mediante la función *"GaussianNB()"*.

```
mn = MultinomialNB()
mn.fit(train_x_vector,train_y)
```

Evaluación de los algoritmos

La evaluación de los algoritmos se lo hace mediante 3 métricas, las cuales son; "*Precision (P)", "Recall (R)" y "F1 Score (F1)"*. Estos se los obtiene partiendo de la matriz de confusión de cada algoritmo, cabe tomar en cuenta que debido a que existen 3 clases para la clasificación, la matriz que se obtiene mediante Python es de 3x3, por lo tanto, es necesario comprender su estructura con la finalidad de evitar errores en su interpretación. En la figura 51, 52 y 53 se presentan los ejemplos de cómo se distribuyen los verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN).

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">Clase de predicción</td></tr>
<tr><td>Clase A</td><td>Clase B</td><td>Clase C</td></tr>
<tr><td rowspan="3">Clase real</td><td>Clase A</td><td>VP</td><td colspan="2">FN</td></tr>
<tr><td>Clase B</td><td rowspan="2">FP</td><td colspan="2" rowspan="2">VN</td></tr>
<tr><td>Clase C</td></tr>
</table>

Figura 52. Determinación de métricas para la clase A.
Elaborado por el investigador.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">Clase de predicción</td></tr>
<tr><td>Clase A</td><td>Clase B</td><td>Clase C</td></tr>
<tr><td rowspan="3">Clase real</td><td>Clase A</td><td>VN</td><td>FP</td><td>VN</td></tr>
<tr><td>Clase B</td><td>FN</td><td>VP</td><td>FN</td></tr>
<tr><td>Clase C</td><td>VN</td><td>FP</td><td>VN</td></tr>
</table>

Figura 53. Determinación de métricas para la clase B.
Elaborado por el investigador.

		Clase de predicción		
		Clase A	Clase B	Clase C
Clase real	Clase A	VN	VN	FP
	Clase B	VN	VN	FP
	Clase C	FN	FN	VP

Figura 54. Determinación de métricas para la clase C.
Elaborado por el investigador.

Las matrices que se obtienen en Python se lo hacen mediante el comando "*confusion_matrix(test_y,svc.predict(test_x_vector),labels=['ambulatorio','revision_medica,'hospitalizacion])"* , a los cuales se les agrega etiquetas las mismas que son los 3 tipos de clases para que la matriz se muestre ordenada. En la figura 54 se muestra las matrices de confusión de cada algoritmo.

```
[[18  5  1]
 [ 2 24  0]
 [ 0  0 18]]
```

a)

```
[[18  4  2]
 [ 6 18  2]
 [ 0  0 18]]
```

b)

```
[[ 6  1 17]
 [ 1 12 13]
 [ 0  0 18]]
```

c)

Figura 55. Matrices de confusión: a) SVM, b) Árbol de decisiones, c) Naive Bayes.
Elaborado por el investigador.

Se procede a calcular los valores de las métricas, esto se lo hace mediante las ecuaciones (2), (3) y (4). En la tabla 19 se muestran los elementos que conforman la matriz de confusión y los valores de todas las métricas de evaluación para cada algoritmo.

Tabla 19. Desempeño de los algoritmos de clasificación (SVM, Árbol de Decisiones y Naive Bayes). Elaborado por el investigador.

VP	VN	FP	FN	Clase	P	R	F1
SVC							
18	42	2	6	Ambulatorio	0.9	0.75	0.82
24	37	5	2	Revisión médica	0.83	0.92	0.87
18	49	1	0	Hospitalización	0.95	1	0.97
Árbol de decisiones							
18	38	6	6	Ambulatorio	0.75	0.75	0.75
18	38	4	4	Revisión médica	0.82	0.69	0.75
18	46	4	0	Hospitalización	0.82	1.00	0.90
Naive Bayes							
18	20	30	0	Ambulatorio	0.86	0.25	0.39
12	41	1	14	Revisión médica	0.92	0.46	0.62
6	43	1	18	Hospitalización	0.38	1.00	0.55

3.1.1.7.6 Selección del algoritmo para el diagnóstico

De acuerdo a los reportes de clasificación de cada uno de los algoritmos se puede determinar que el algoritmo que se puede utilizar para la clasificación del riesgo de COVID-19 es el de Máquina de Vectores de Soporte debido a que, para la clase ambulatorio se tiene un "Recall" de 0.75 y "Precision" de 0.9, para la clase "Revisión médica" 0.92 y 0.83 y para la clase "Hospitalización" 1 y 0.95, valores que son más altos que para los otros modelos, lo que quiere decir que la clasificación de cada clase es garantizada. De igual manera, se puede observar que, mediante F1-Score, el algoritmo clasifica mejor la clase "Hospitalización".

3.1.1.7.7 Almacenamiento del modelo final

Una vez escogido el modelo final se utiliza la librería *"pickle"* con el objetivo de almacenar los datos entrenados basados en el modelo final, esto se lo realiza mediante la serialización de objetos lo cual convierte objetos a cadenas de bytes para posteriormente escribirlos en un archivo con extensión *".pickle"*. Este proceso se lo realiza para poder utilizar el archivo independientemente en otro script sin la necesidad de realizar todos los pasos para la elección del algoritmo final, reduciendo el consumo de los recursos de la Raspberry Pi Zero y aumentando la rapidez del diagnóstico.

Cabe mencionar que se deben serializar tanto el modelo final como los datos vectorizados.

```
datosVec = 'datos.pickle'
pickle.dump(tfidf, open(datosVec, 'wb'))

modeloFinal = 'modeloDT.pickle'
pickle.dump(dec_tree, open(modeloFinal, 'wb'))
```

Figura 56. Serialización de datos del dataset y del algoritmo final.
Elaborado por el investigador.

En la figura 56 se muestra el código para poder utilizar los archivos, en donde se debe únicamente realizar la lectura de los mismos para posteriormente utilizar el algoritmo elegido.

```
import pickle

modeloFinal = pickle.load(open('modeloDT.pickle', 'rb'))
datoVec = pickle.load(open('datos.pickle', 'rb'))
sintoma = "Fiebre"

prediccion = modeloFinal.predict(datoVec.transform([sintoma]))
print(prediccion)
```

Figura 57. Ejecución del algoritmo almacenado mediante la librería Pickle.
Elaborado por el investigador.

En el script *"main.py"* se crea una función llamada *"diagnostico(sintoma)"* cuyo argumento recibe todos los síntomas y datos que se eligen a través del mismo menú que se visualiza en la pantalla del dispositivo electrónico.

```
def diagnostico(sintoma):
  modeloFinal = pickle.load(open('/home/santiago/Documentos/Tesis/pruebas_ia/modeloDT.pickle', 'rb'))
  datosVec = pickle.load(open('/home/santiago/Documentos/Tesis/pruebas_ia/datos.pickle', 'rb'))
```

Figura 58. Creación de la función para realizar el diagnóstico en el dispositivo electrónico.
Elaborado por el investigador.

3.1.1.8 Almacenamiento de datos en base de datos local

Todos los datos que se ingresan para realizar el diagnóstico se deben alojar en alguna base de datos. De acuerdo a la tabla 6 se utiliza MySQL debido a su compatibilidad con Linux,

es de código abierto y puede ser administrado mediante la aplicación PhpMyAdmin de forma sencilla y con una interfaz muy amigable.

De acuerdo al diagrama de flujo de la figura 41 se genera una tabla llamada "*diagnostico_pacientes*" el cual consta de 11 campos, tal y como se muestra en la figura 58.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	cedula	varchar(15)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	fecha	varchar(20)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	4	pulso	float			No	*Ninguna*			Cambiar Eliminar Más
☐	5	spo2	float			No	*Ninguna*			Cambiar Eliminar Más
☐	6	temperatura	float			No	*Ninguna*			Cambiar Eliminar Más
☐	7	sexo	text	utf8mb4_general_ci		No				Cambiar Eliminar Más
☐	8	vacuna	text	utf8mb4_general_ci		No				Cambiar Eliminar Más
☐	9	condiciones_especiales	text	utf8mb4_general_ci		No				Cambiar Eliminar Más
☐	10	sintomas	text	utf8mb4_general_ci		No				Cambiar Eliminar Más
☐	11	diagnostico	text	utf8mb4_general_ci		No				Cambiar Eliminar Más

Figura 59. Tabla de diagnóstico de pacientes.

Elaborado por el investigador.

Se describen cada uno de los campos de la tabla.

- "id"

  Este campo contiene un valor autoincrementable el cual ayuda a generar un número único al insertar un nuevo registro en la tabla, con el objetivo de contabilizar el número de registros.

- "cedula"

  Es de suma importancia registrar la identificación del paciente debido a que estos datos servirán al profesional de la salud para identificar de quien se trata y cuál es su estado de salud.

- "fecha"

  Este campo indica la fecha en que se ha realizado el diagnóstico del nivel de riesgo de COVID-19.

- "pulso"

  Indica el valor de la frecuencia cardiaca del paciente.

- "spo2"

  Indica el valor de la saturación del nivel de oxígeno en la sangre.

- “temperatura”

Indica la temperatura corporal del paciente.

- “sexo”

Indica la identidad del género del paciente.

- “vacuna”

Indica si el paciente ha sido vacunado y en el caso de que sea así, qué tipo de vacuna tiene.

- “condiciones_especiales”

Indica las condiciones especiales que tiene el paciente, tales como diabetes, cáncer, enfermedades cardiacas, embarazo, etc.

- “sintomas”

Este campo alberga todos los síntomas que el paciente posee, estos pueden ser desde una tos o fiebre hasta la condición en la que está el paciente de acuerdo a la frecuencia cardiaca (bradicardia o taquicardia) y el SpO2 (hipoxia leve, moderada o severa).

- “diagnostico”

Este campo alberga el nivel de riesgo de COVID-19 que tiene el paciente, el mismo que se obtiene mediante la predicción del algoritmo de Inteligencia Artificial.

Conexión de la base de datos con la Raspberry Pi Zero

Una vez creada la tabla se procede a realizar la conexión mediante un script hecho en Python tal como se muestra en la figura 59. Se crea en el archivo *“main.py”* una función llamada *“db(id, fecha, pulso, spo2, temperatura, sexo, vacuna, condiciones_especiales, sintomas, diagnostico)”* cuyos argumentos recibirán los datos respectivos una vez ejecutado la función *“diagnostico(sintomas)”* en donde se realiza la clasificación de riesgo de COVID-19.

Se añade el sitio de alojamiento el cual es “localhost”, el nombre de la base de datos, el usuario y password con el que se creó la misma.

```
def db(id, fecha, pulso, spo2, temperatura, sexo, vacuna, condiciones_especiales, sintomas, diagnostico):
  try:
    db = mysql.connector.connect(host='localhost', database='borrador', user='admin', password='1234')
    cursor = db.cursor()
    cursor.execute( '''INSERT INTO diagnostico_pacientes (id, fecha, pulso, spo2, temperatura, sexo,
                                                vacuna, condiciones_especiales, sintomas, diagnostico)
                      VALUES (NULL,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s);
                    ''',
                  (fechaActual,sigPulso,sigSpo2,sigTemp,sexo,vacuna,condEsp,sintomas,diagnostico))
    db.commit()
    db.close()
    print("Envío positivo")

  except:
    print("No se envió a la base de datos")
    pass
```

Figura 60. Código para la conexión del dispositivo con la base de datos local.

Elaborado por el investigador.

Se ejecuta el archivo principal y se muestran los datos que han sido enviados a la base de datos local, esto se observa en la figura 60.

id	cedula	fecha	pulso	spo2	temperatura	sexo	vacuna	condiciones_especiales	sintomas	diagnostico
1	1801234567	24/07/2022 23:42:51	45.35	74.03	35.29	Masculino	Aztrazeneca	Hipotiroidismo, Obesidad, Cancer	Tos, Dolor de cabeza, Dolor de articulaciones, Bra...	Revisión médica

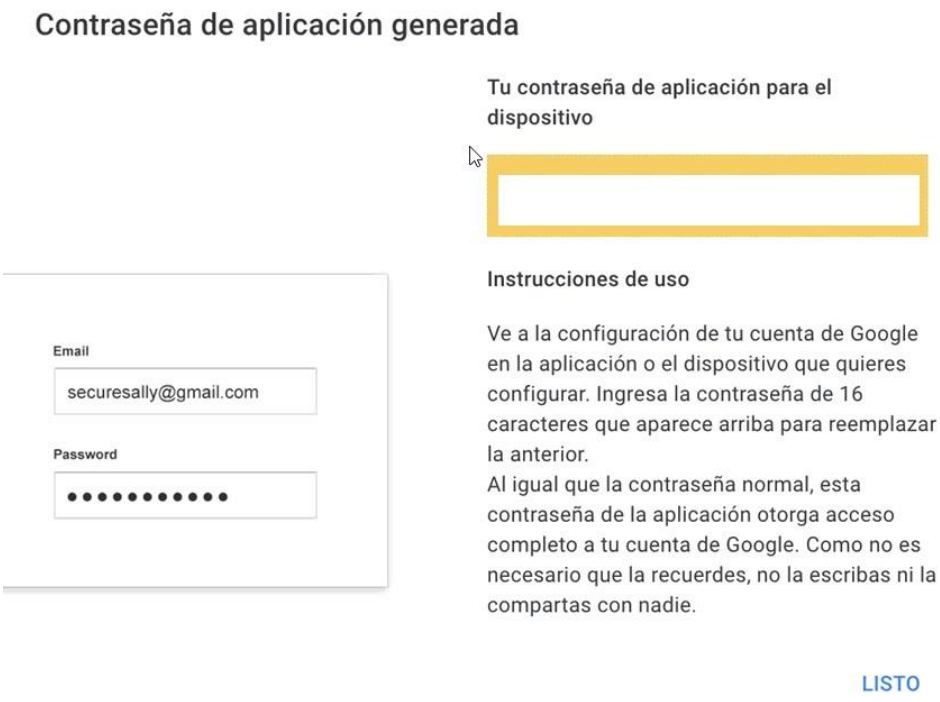
Figura 61. Datos almacenados en la base de datos local.

Elaborado por el investigador.

3.1.1.9 Sistema de alertas al correo

Para el envío de alertas al correo se crea una función llamada *"enviarCorreo(asunto,cuerpo)"* cuyos argumentos reciben el diagnóstico de riesgo de COVID-19 y la recomendación respectiva. Esto se lo realiza utilizando la librería *"email.message"* el cual configura el encabezado y los cuerpos de los mensajes para crear o modificar mensajes estructurados. También se utilizan las librerías *"ssl"* y *"smtplib"*.

Para enviar correos por Python se debe dirigir a GoogleAcount en la pestaña de seguridad para activar la verificación en dos pasos, el cual provee una contraseña de 16 dígitos mismo que se utiliza en el script de Python.



La librería *"ssl"* brinda seguridad en la conexión hacia internet y envío de datos tanto del lado del cliente como del lado del servidor, ya que proporciona acceso a la capa de seguridad de transporte o más conocido como SSL (Secure Sockets Layer) y permite la autenticación, encriptación y desencriptación de datos que se envían por Intenert. La librería *"smtplib"* se utiliza para enviar correos electrónicos mediante el protocolo SMTP (Simple Mail Transfer Protocol), lo que permite que un cliente se comunique con un servidor para poder enviar un correo a uno o más receptores.

```
def enviarCorreo(asunto, cuerpo):
  email_emisor = '                    .com'
  email_contrasena = '                '
  email_receptor = '                  .es'

  asunto0 = asunto
  cuerpo0 = cuerpo

  em = EmailMessage()
  em['From'] = email_emisor
  em['To'] = email_receptor
  em['Subject'] = asunto0
  em.set_content(cuerpo0)

  contexto = ssl.create_default_context()
```

Se hace la conexión al servidor de correo Gmail ya que la dirección de correo electrónico del emisor usa este servidor. El puerto de comunicación que utiliza el protocolo SMTP es el 465 ya que trabaja sobre SSL lo cual brinda mayor seguridad en el envío de los mensajes.

```
try:
  with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=contexto) as smtp:
    smtp.login(email_emisor, email_contrasena)
    smtp.sendmail(email_emisor, email_receptor, em.as_string())
except:
  pass
```

La función es llamada al momento de ejecutarse la función *"diagnostico"* en donde se envían los respectivos datos a los dos argumentos.

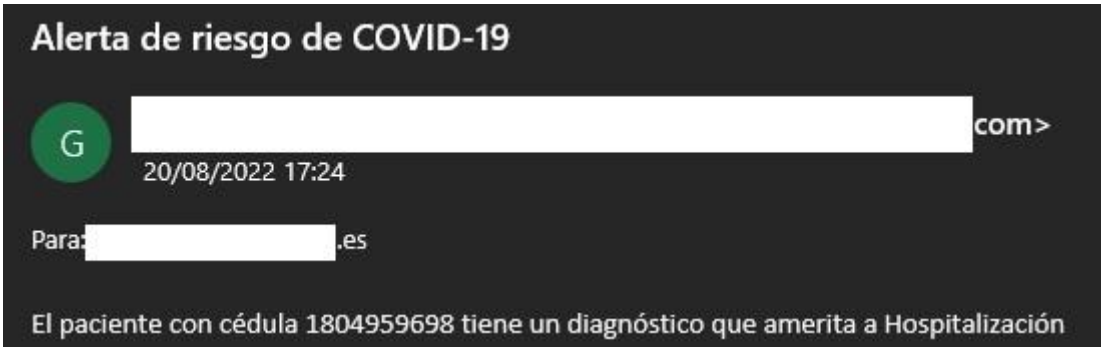


Figura 62. Alerta recibida al correo electrónico.
Elaborado por el investigador.

3.1.1.10 Almacenamiento en la nube

La necesidad del almacenamiento de los datos en la nube surge debido a que el médico tratante pueda observar los resultados de las pruebas realizadas y así determine las acciones necesarias para el cuidado del mismo. Por lo tanto, se elige un servicio de almacenamiento en la nube mediante un web hosting gratuito.

Para poder obtener el hosting gratuito se ingresa a la página *"https://www.000webhost.com/"* y mediante un correo se inicia sesión para contratar el servicio gratuito. El sitio web tendrá como nombre "covidsintomas" junto con el dominio que provee este servidor.

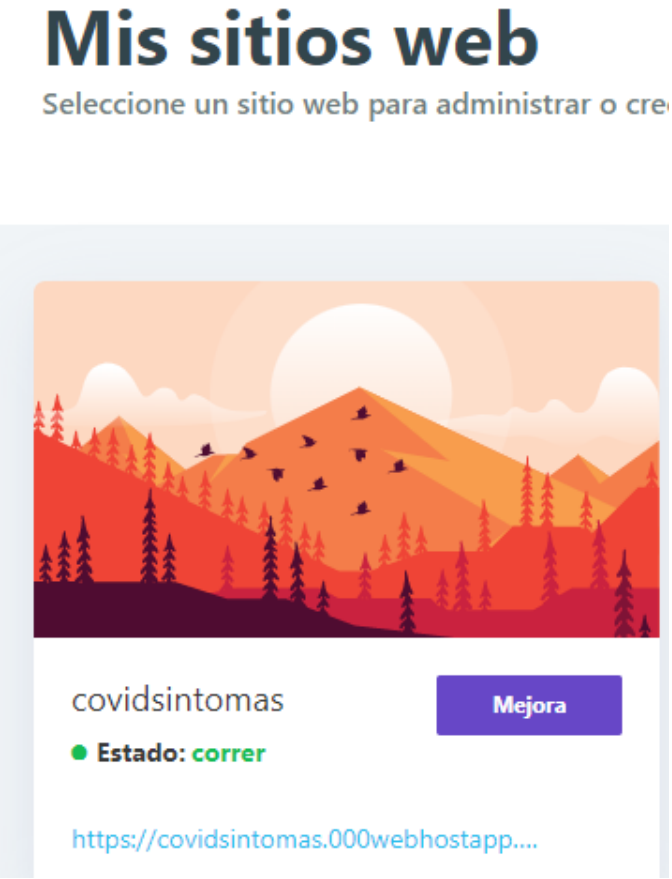


Figura 63. Servicio de web hosting creado en "000webhost".
Elaborado por el investigador.

Sincronización entre la base de datos local y en la nube

Se comienza creando una base de datos y una tabla que contenga los mismos campos que tiene la tabla de la base de datos local como se muestra en la figura 63.

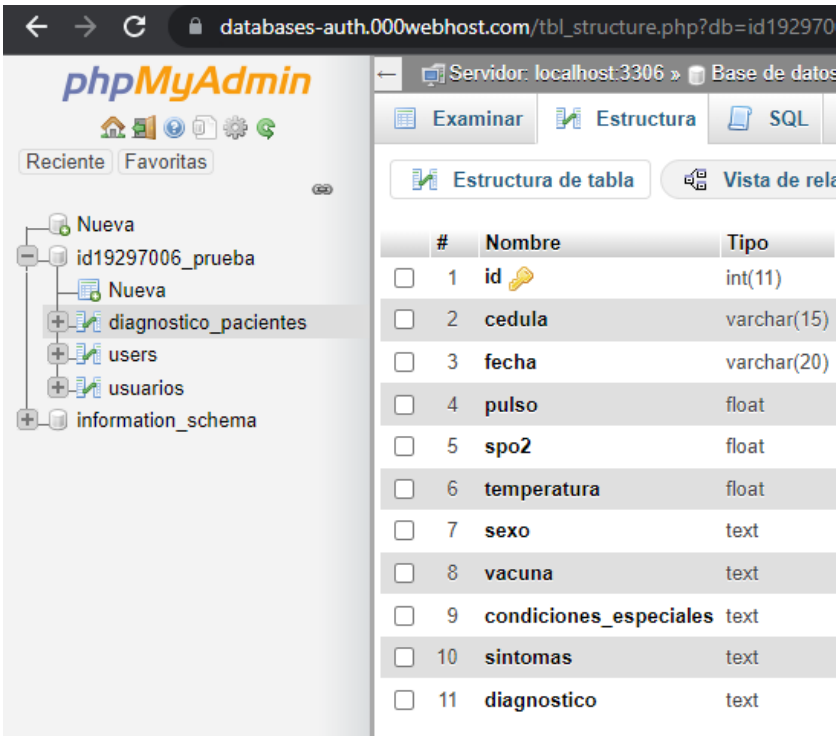


Figura 64. Creación de la base de datos en la nube.
Elaborado por el investigador.

Luego se crean los archivos *"conexion.php"*, *"insertar.php"* y *"eliminar.php"*. El primer archivo se encarga de realizar la conexión con la base de datos, el segundo ingresa los datos provenientes de la Raspberry a la base de datos en la nube y el tercero elimina los datos contenidos en la tabla.

Posteriormente se crea un script en Python llamado *"sincronizar.py"* en donde se realiza el código para la conexión entre las dos bases de datos. Para esto es necesario la utilización de la librería *"requests"* la cual permite enviar solicitudes HTTP de forma muy simple,

de esta librería se utiliza el método post() para enviar los datos al servidor web como se muestra en la figura 64.

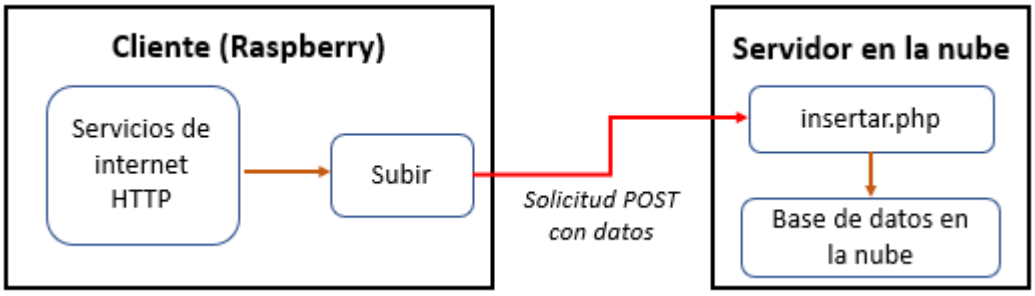


Figura 65. Método "requests.post" mediante Python.
Elaborado por el investigador.

El método request.post recibe como argumentos la dirección web donde se encuentra el archivo *"insertar.php"* y los parámetros que se envían son los datos que se encuentran en la base de datos local como se muestra en la figura 65. El archivo *"sincronizar.py"* se va a ejecutar en la función *"diagnóstico()"* del script principal.

```
userdata = {"id": fila[0], "fecha": fila[1],"pulso": fila[2], "spo2": fila[3],"temperatura": fila[4],
resp = requests.post('https://covidsintomas.000webhostapp.com/insertar.php', params=userdata)
```

Figura 66. Sincronización entre la base de datos local y en la nube.
Elaborado por el investigador.

3.1.1.11 Interfaz web para la visualización de datos de pacientes

El diseño comienza con la creación de un sistema de ingreso mediante usuario y contraseña con la finalidad de que los datos de los pacientes sean confidenciales y pueda acceder únicamente el médico autorizado. Para esto se crea un archivo *"login.php"* en donde se ingresarán los datos respectivos para ingresar a observar los datos de los pacientes. Se crea un formulario con 3 campos, el primer campo es para ingresar el usuario, el segundo para la ingresar la contraseña y el tercero es un botón el cual, si los datos ingresados son los correctos, direccionará a la interfaz en donde están los datos de los pacientes.

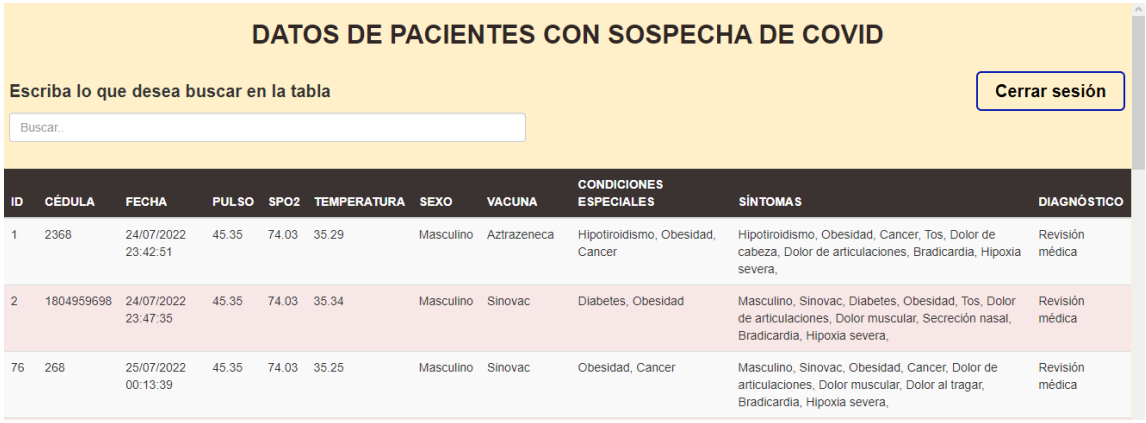
```
<form action="login.php" method="post">
  <input type="text" name="email" placeholder="Ingresa tu email">
  <input type="password" name="password" placeholder="Ingresa tu contraseña">
  <input type="submit" value="Enviar">
</form>
```

La interfaz queda como se muestra en la figura 66, el usuario y la contraseña es establecida por el investigador y no puede ser revelada por motivos de seguridad.

Figura 67. Interfaz de ingreso mediante usuario y contraseña.
Elaborado por el investigador.

Luego se crea otro archivo llamado *"index.php"*, en donde se diseña la interfaz la cual mostrará todos los datos que se encuentran en la base de datos de la nube, adicionalmente se añade un bloque de filtro en donde se ingresa el número de cédula del paciente para encontrar de manera eficaz algún paciente específico.

Una vez creado las dos interfaces se evalúa el desempeño de la página web. Si el usuario y la contraseña son correctas, al momento de dar click en el botón "enviar" se dirige a la interfaz donde se encuentran los datos de los pacientes.



ID	CÉDULA	FECHA	PULSO	SPO2	TEMPERATURA	SEXO	VACUNA	CONDICIONES ESPECIALES	SÍNTOMAS	DIAGNÓSTICO
1	2368	24/07/2022 23:42:51	45.35	74.03	35.29	Masculino	Aztrazeneca	Hipotiroidismo, Obesidad, Cancer	Hipotiroidismo, Obesidad, Cancer, Tos, Dolor de cabeza, Dolor de articulaciones, Bradicardia, Hipoxia severa,	Revisión médica
2	1804959698	24/07/2022 23:47:35	45.35	74.03	35.34	Masculino	Sinovac	Diabetes, Obesidad	Masculino, Sinovac, Diabetes, Obesidad, Tos, Dolor de articulaciones, Dolor muscular, Secreción nasal, Bradicardia, Hipoxia severa,	Revisión médica
76	268	25/07/2022 00:13:39	45.35	74.03	35.25	Masculino	Sinovac	Obesidad, Cancer	Masculino, Sinovac, Obesidad, Cancer, Dolor de articulaciones, Dolor muscular, Dolor al tragar, Bradicardia, Hipoxia severa,	Revisión médica

Figura 68. Visualización de datos del paciente en la interfaz web.

Elaborado por el investigador.

3.1.1.12 Costo del prototipo

Precios de Hardware

Los costos que conllevaron la parte de los componentes electrónicos para la implantación del sistema electrónico se detallan en la tabla 20.

Tabla 20. Precios del hardware para el avance del proyecto.

Elaborado por el investigador.

Ítem	Recursos Económicos	Unidad	Cantidad	P. Unit. (USD)	P. Total (USD)
1	Módulo de alimentación de batería de litio PiSugar	c/u	1	51.75	51.75
2	Raspberry Pi Zero W	c/u	1	30.00	30.00
3	Pantalla OLED 1.3´´	c/u	1	16.00	16.00
4	Sensor MAX30100	c/u	1	10.00	10.00
5	Sensor MLX90614	c/u	1	15.00	15.00
6	PCB	c/u	1	14.00	14.00
7	Estaño	c/u	1	2.50	2.50
8	Pegamento instantáneo	c/u	2	0.50	1.00
9	Correa de cuero	c/u	1	3.00	3.00
10	Case (Hora de impresión)	Horas	5	1.50	7.50
11	Materiales de oficina	c/u	1	10.00	10.00
12	Transporte	-	50	0.3	25
13	Alimentación	c/u	50	2	10.00
14	Impresión 3D Case	c/u	1	20.00	20.00
	Subtotal	-	-		215.75
	Imprevistos (5%)	-	-	-	10.79
	TOTAL	**-**	**-**	**-**	**226.54**

Precios del software

Los programas que se utilizaron para el desarrollo del proyecto tienen software libre y gratuito, los cuales a continuación se describen:

- Sistema operativo Raspbian 32 bits.
- MySQL.
- Python 3.
- 000webhosting.

Precio de mano de obra

El cálculo del costo de la mano de obra consiste en dos factores, el tiempo utilizado para la elaboración del proyecto y el salario de un Ingeniero en Telecomunicaciones, el cual

según el ministerio de Trabajo es de 858 dólares mensuales. Tomando en cuenta que el proyecto se lo desarrolló en 6 meses, el precio de la mano de la obra sería de 3416,00 dólares.

3.2 Pruebas de funcionamiento

Se realizaron pruebas de funcionamiento del dispositivo electrónico para calcular la fiabilidad del mismo, con la finalidad de comprobar si los valores obtenidos por los sensores cumplen con los niveles normales para un control adecuado del estado de salud del paciente. Se hace la comparación entre un pulsioxímetro comercial y los valores del sensor MAX30100 y, mientras que para la temperatura se utiliza un termómetro digital de axila con el sensor MLX90614, sensores que están instalados en el dispositivo electrónico.

La determinación de las medidas se lo realiza partiendo de la ecuación (7), la cual determina el valor para el error absoluto, en donde se utiliza el Valor exacto (valor del dispositivo médico comercial) y el Valor real (valor de los sensores). Posteriormente se calcula el error relativo mediante la ecuación (8) y este resultado se lo resta 100 para obtener la fiabilidad, tal como se observa en la ecuación (9).

$$Error\ absoluto = |Valor\ exacto - Valor\ real| \quad \textbf{(7)}$$

$$Error\ relativo = \frac{Error\ absoluto}{Valor\ exacto} * 100\% \quad \textbf{(8)}$$

$$Fiabilidad = 100 - Error\ relativo \quad \textbf{(9)}$$

Frecuencia cardiaca

Se hace el cálculo de la fiabilidad de las lecturas de frecuencia cardiaca del dispositivo electrónico mediante pruebas realizadas en 15 personas. Se observa en la tabla 21 un

porcentaje de error relativo promedio de 5.02 %, con lo que se obtiene una fiabilidad de 94.98 % para la medición de la frecuencia cardiaca.

Tabla 21. Pruebas de frecuencia cardiaca.

Elaborado por el investigador.

FRECUENCIA CARDIACA					
Pacientes	**Pulsioxímetro comercial**	**Dispositivo electrónico**	**Error absoluto**	**Error Relativo**	**Error porcentual**
1	83	81	2	0.02	2.41%
2	82	82	0	0.00	0.00%
3	87	81	6	0.07	6.90%
4	83	80	3	0.04	3.61%
5	84	80	4	0.05	4.76%
6	83	81	2	0.02	2.41%
7	85	81	4	0.05	4.71%
8	79	80	1	0.01	1.27%
9	85	82	3	0.04	3.53%
10	80	83	3	0.04	3.75%
11	76	83	7	0.09	9.21%
12	77	83	6	0.08	7.79%
13	79	83	4	0.05	5.06%
14	75	82	7	0.09	9.33%
15	76	84	8	0.11	10.53%
			Promedio de error		5.02%
			Fiabilidad		94.98%

De igual manera se puede observar la tendencia que tienen las medidas de la frecuencia cardiaca de los dos dispositivos en la figura 68. La tendencia que tiene la medición de este signo vital por parte del sensor no se asemeja mucho a la del pulsioxímetro comercial, razón por la cual se obtuvo el porcentaje de fiabilidad presentado en la tabla 21.

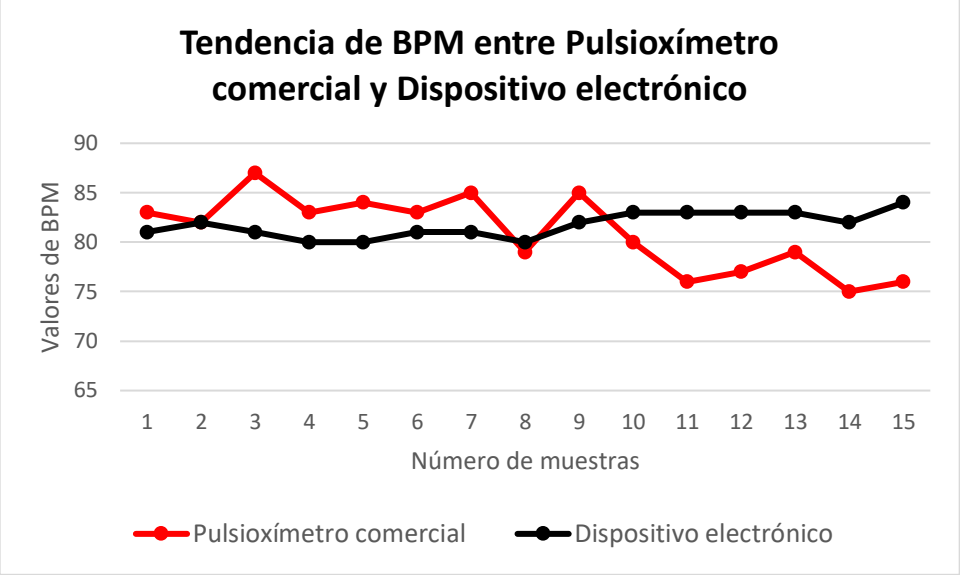


Figura 69. Tendencias de la frecuencia cardiaca en los dos dispositivos.
Elaborado por el investigador.

En la tabla 22 se muestran los valores del nivel de oxígeno en la sangre, obteniendo un porcentaje de error relativo promedio de 1.04 %, y, por lo tanto, la fiabilidad para medir este signo vital es del 98.96 %.

Tabla 22. Pruebas del nivel de oxígeno en la sangre.
Elaborado por el investigador.

NIVEL DE OXÍGENO EN LA SANGRE

Pacientes	Pulsioxímetro comercial	Dispositivo electrónico	Error absoluto	Error Relativo	Porcentaje de error
1	94	92	1.62	0.01723404	1.72%
2	93	94	0.59	0.00634409	0.63%
3	93	93	0.3	0.00322581	0.32%
4	92	93	0.57	0.00619565	0.62%
5	92	94	1.61	0.0175	1.75%
6	93	93	0.05	0.00053763	0.05%
7	92	94	1.78	0.01934783	1.93%
8	93	93	0.39	0.00419355	0.42%
9	93	94	0.58	0.00623656	0.62%
10	93	94	0.53	0.00569892	0.57%
11	94	92	1.61	0.01712766	1.71%
12	94	93	0.56	0.00595745	0.60%
13	95	93	2.17	0.02284211	2.28%
14	94	92	1.58	0.01680851	1.68%
15	94	93	0.66	0.00702128	0.70%
			Promedio de error		1.04%
			Fiabilidad		98.96%

En la figura 69 se observa la tendencia que tienen los valores del nivel de oxígeno en la sangre medidos por cada dispositivo. La medición por parte del sensor tiende más a aparecerse al pulsioxímetro comercial, lo que corrobora el porcentaje de fiabilidad presentado en la tabla 22.

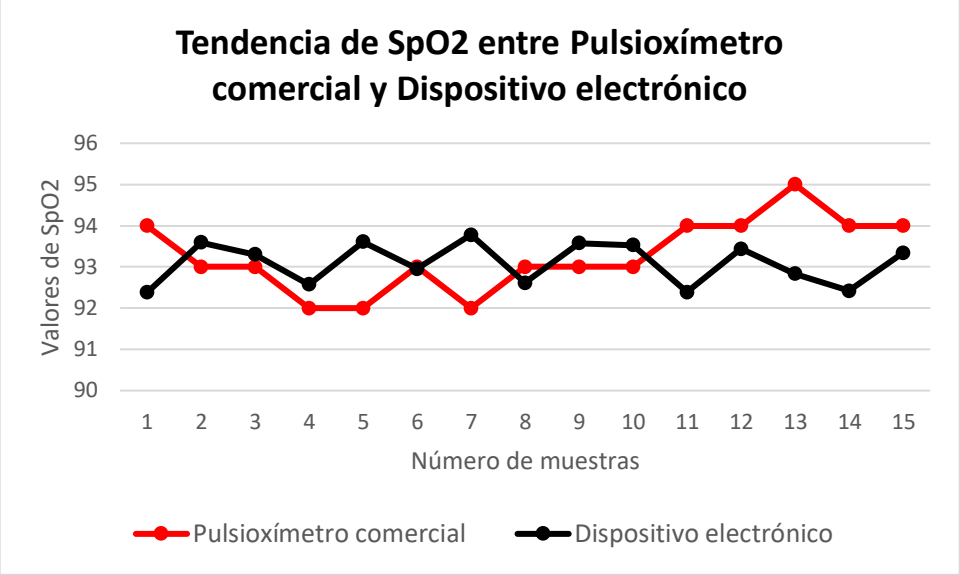


Figura 70. Tendencias del nivel de oxígeno en los dos dispositivos.

Elaborado por el investigador.

En la tabla 23 se muestran las medidas de la temperatura corporal por parte del termómetro comercial como del sensor MLX90614 instalado en el dispositivo electrónico. Se puede observar que el porcentaje de fiabilidad es igual a 99.78 % siendo este signo vital el que más fiabilidad tiene con respecto a los otros al momento de la sensorizacion.

Tabla 23. Pruebas de la temperatura corporal.

Elaborado por el investigador.

TEMPERATURA CORPORAL					
Pacientes	**Termómetro comercial**	**Dispositivo electrónico**	**Error absoluto**	**Error Relativo**	**Porcentaje de error**
1	36.5	36.6	0.14	0.00383562	0.38%
2	36.6	36.6	0.03	0.00081967	0.08%
3	36.6	36.7	0.1	0.00273224	0.27%
4	36.6	36.7	0.05	0.00136612	0.14%
5	36.5	36.6	0.1	0.00273973	0.27%
6	36.6	36.6	0.01	0.00027322	0.03%
7	36.7	36.6	0.1	0.0027248	0.27%
8	36.6	36.6	0	0	0.00%
9	36.6	36.6	0.03	0.00081967	0.08%
10	36.5	36.5	0	0	0.00%
11	36.6	36.6	0.01	0.00027322	0.03%
12	36.7	36.6	0.09	0.00245232	0.25%
13	36.7	36.6	0.09	0.00245232	0.25%
14	36.5	36.6	0.12	0.00328767	0.33%
15	36.7	36.6	0.08	0.00217984	0.22%
16	36.4	36.3	0.11	0.00302198	0.30%
17	36.5	36.3	0.22	0.0060274	0.60%
18	36.4	36.3	0.11	0.00302198	0.30%
19	36.4	36.3	0.11	0.00302198	0.30%
20	36.4	36.3	0.12	0.0032967	0.33%
			Promedio de error		0.22%
			Fiabilidad		99.78%

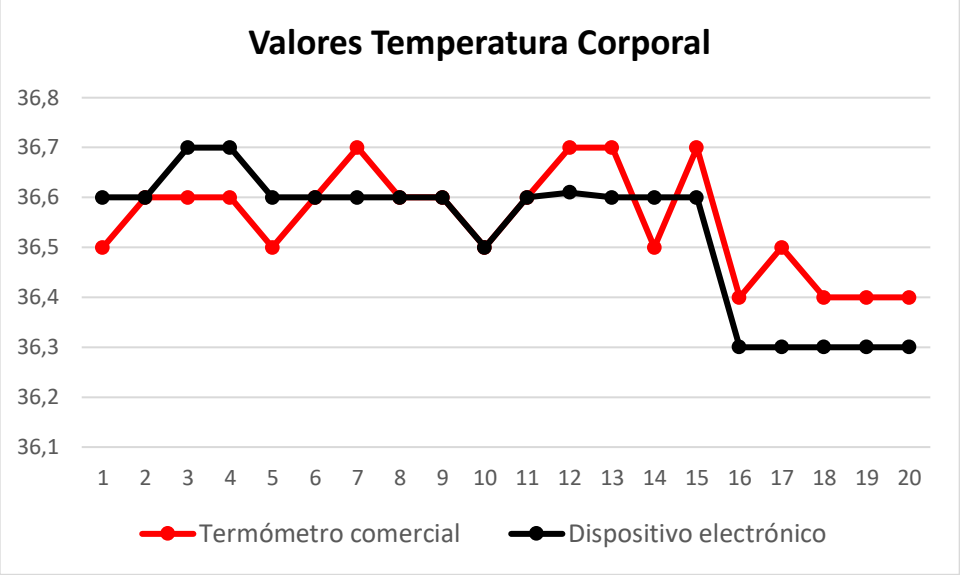


Figura 71. Tendencias de la temperatura corporal en los dos dispositivos.

Elaborado por el investigador.

Una vez verificado la fiabilidad de cada uno de los signos vitales se procede a realizar el análisis del diagnóstico del nivel de riesgo de COVID-19, esto se lo realizó en pacientes cuyas identidades son clasificadas debido al protocolo de privacidad que maneja la doctora con la que se hicieron las visitas. En la figura 71 se observan los pacientes y como se usó la pulsera para poder realizar el diagnóstico. En la tabla 24 se muestran las pruebas realizadas con sus respectivos campos que son necesarios para el diagnóstico de riesgo de COVID-19

Figura 72.Pruebas de diagnóstico de riesgo de COVID-19 en pacientes sospechosos.

Elaborado por el investigador.

Tabla 24. Pruebas realizadas mediante el dispositivo electrónico.

Elaborado por el investigador.

N°	Género	Edad	Temp °C	BPM	SpO2	Vacuna	Condiciones especiales	Síntomas	Recomendación del dispositivo	Diagnóstico por parte del médico
1	Masculino	26	38.1	100.2	93.7	Si	No	Pérdida del gusto, pérdida del olfato.	Ambulatorio	Correcta
2	Femenino	20	36.1	86.6	95.3	Si	No	Dolor de articulaciones, dolor de cabeza, malestar general.	Ambulatorio	Correcta
3	Masculino	25	37.9	100.1	92.3	Si	No	Pérdida del gusto, pérdida del olfato, falta de aire.	Rev. Med.	Incorrecta
4	Masculino	70	38	100.8	89.4	No	No	Malestar general, , tos, dolor de cabeza.	Rev. Med.	Correcta
5	Femenino	60	38.2	98.9	92.6	No	No	Dolor de cabeza, tos, gripe, dolor de articulaciones.	Ambulatorio	Correcta
6	Masculino	38	36.2	87.07	94.96	Si	No	Pérdida del olfato, pérdida del gusto, falta de aire, cansancio.	Rev. Med.	Incorrecta
7	Femenino	42	36.1	81.71	91.9	No	No	Dolor de cabeza, malestar general.	Ambulatorio	Correcta
8	Masculino	23	36.1	81.81	90.85	Si	No	Malestar general, tos seca, dolor de cabeza.	Rev. Med.	Correcta

9	Femenino	25	38.1	103.8	92.5	No	No	Malestar, tos, dificultad respiratoria	Ambulatorio	Correcta
10	Masculino	25	38.2	99.4	86.3	Si	Diabetes	Tos, dolor de pecho, falta de aire, malestar general, presión alta, cianosis	Rev. Med	Correcta
11	Femenino	26	36.1	81.53	91.4	No	No	Pérdida del gusto, pérdida del olfato, cansancio extremo, malestar general, vómito.	Ambulatorio	Correcta
12	Femenino	23	36.2	81.76	91.5	No	No	Falta de aire, hipotermia, dolor de articulaciones.	Emergencia	Correcta
13	Masculino	23	38.1	100.7	92.2	Si	No	Tos, malestar general, dolor de cabeza, hipoxia leve	Rev. Med.	Correcta
14	Masculino	29	36.3	79.81	87.2	No	No	Dolor cabeza fatiga falta de aire, perdida del olfato, perdida del gusto.	Emergencia	Correcta
15	Femenino	31	37.9	98.2	91.4	No	No	Dolor de articulaciones, vómito, diarrea, dolor de cabeza, dolor de pecho, falta de aire,	Ambulatorio	Incorrecta

								pérdida del gusto, pérdida de olfato.		
16	Femenino	42	38.1	102.7	90.3	Si	No	Malestar general, taquicardia, falta de aire y ardor de garganta.	Rev. Med.	Correcta
17	Masculino	35	36.2	82.05	92.9	No	No	Dolor de cabeza, escalofríos, dolor de articulaciones, pérdida del olfato, pérdida del gusto.	Rev. Med.	Correcta
18	Masculino	36	38.1	105.2	91.6	No	No	Dolor muscular, , perdida del gusto, diarrea, dolor de cabeza, vómito y cansancio.	Rev. Med.	Correcta
19	Femenino	35	38.2	104.3	85.8	No	No	Tos, dolor de cabeza, dolor de pecho, abdomen, falta de aire, , presión alta, fatiga, dolor de pecho y arritmia.	Emergencia	Correcta
20	Femenino	29	38	101.7	92.3	Si	No	Pérdida del olfato perdida del gusto, falta de aire, y, malestar general.	Rev. Med.	Correcta

En la figura 72 se observa la proporción de las pruebas correctas e incorrectas, teniendo de 20 pruebas, 17 correctas y 3 incorrectas.

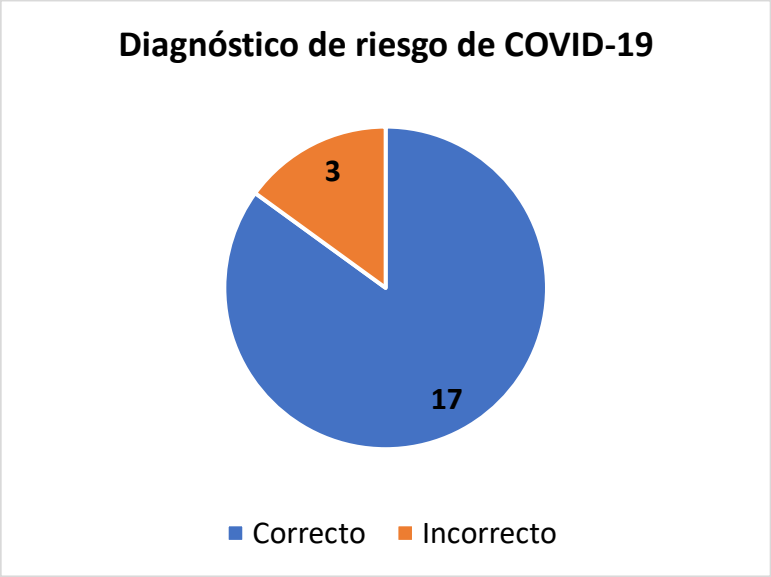


Figura 73. Diagrama de pastel de los diagnósticos de riesgo de COVID-19.
Elaborado por el investigador.

Se observa en la figura 73 la cantidad de clases predichas por el algoritmo, en donde la clase Revisión Médica es la que mayor número tiene, pero esto no quiere decir que todas las predicciones sean correctas. Esto se muestra en la figura 74 que indica 1 error y 7 aciertos para la clase Ambulatorio, 2 errores y 8 aciertos para la clase Revisión Médica y 3 aciertos con 0 errores para la clase Emergencia Médica. Con lo que se podría deducir que, de acuerdo a la tabla 19, el algoritmo clasifica mejor la clase Emergencia médica tal y como se indicaba en la métrica de evaluación F1-Score.

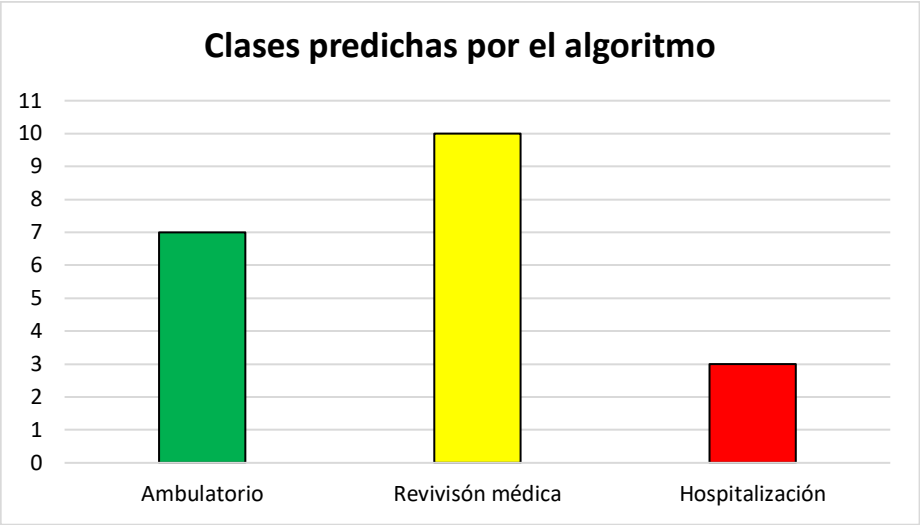


Figura 74. Clases predichas por el algoritmo.

Elaborado por el investigador.

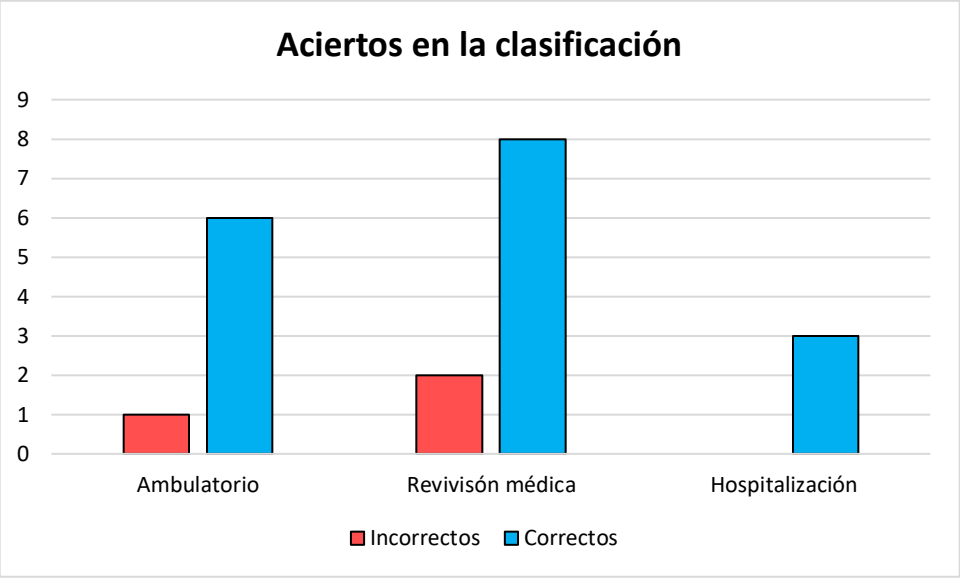


Figura 75. Aciertos en la clasificación.

Elaborado por el investigador.

De las 20 pruebas que se realizaron se obtuvieron 17 correctas, esto lo determinó la doctora quien ayudó en la validación del prototipo. La clase que más cantidad de predicciones tiene es la clase de Revisión Médica con un total de 10, pero de estos dos fueron erróneas. Luego sigue la clase Ambulatorio con 7 predicciones de las cuales una fue equivocada. La clase Hospitalización no tuvo ningún error de las tres predicciones que se obtuvo. Estos resultados muestran que el algoritmo está obteniendo la mayor cantidad de Verdaderos Positivos minimizando así los Falsos Negativos y Falsos Positivos, lo que constata la deducción obtenida a partir de los valores de las métricas “Recall” y “Precisión” presentadas en la tabla 19.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se analizaron los sistemas electrónicos de adquisición de señales vitales en personas en donde se analizaron tanto dispositivos comerciales y dispositivos desarrollados mediante investigaciones. Estos últimos fueron de gran relevancia debido a que se posee la información completa de su arquitectura y materiales utilizados. A partir de esto se procedió a realizar un dispositivo utilizando en este caso una Raspberry Pi Zero W, el cual debido a su característica de microordenador puede realizar el almacenamiento, procesamiento de datos y diagnóstico del nivel de riesgo de COVID-19.

- Se hizo un estudio de los algoritmos de Inteligencia Artificial que se utilizan en el campo de la medicina, destacando el uso de Redes Neuronales convolucionales para la detección del COVID-19 mediante imágenes de radiografías. El uso de los algoritmos de clasificación también toma importancia en este campo, especialmente los algoritmos para la clasificación de texto, razón por la cual se los eligió para el diagnóstico de riesgo de COVID-19 debido a que los datos de entrada son signos vitales y síntomas.

- El modelo de Máquina de Soporte de Vectores tuvo mayor eficiencia que los otros modelos en la clasificación del nivel de riesgo de COVID-19. Se obtuvo un "Recall" y "Precision" de 0.75 y 0.9 respectivamente para la clase "Ambulatorio", para la clase "Revisión Médica" valores iguales a 0.92 y 0.83 y para la clase "Hospitalización" 1 y 0.95, por lo que concluye que el modelo puede manejar eficientemente las 3 clases. La clase que más puede predecir el algoritmo es la clase "Hospitalización" debido a que la métrica F1 Score tiene un valor igual a 0.97.

- Los signos vitales obtenidos por el prototipo tuvieron una fiabilidad de 94.98 % para la frecuencia cardiaca, 98.96 % para el nivel de oxígeno en la sangre y 99.78 % para la temperatura corporal, con lo que se determina que los valores se acercan a los que se obtienen mediante los equipos médicos.

- Se implementó el sistema electrónico de diagnóstico de COVID-19 en personas mediante Inteligencia Artificial partiendo de un análisis técnico de los dispositivos electrónicos con las mismas características que se pretendió realizar. Se tomó en cuenta las necesidades tanto del paciente como del médico tratante y también los recursos del ordenador, en donde se va a realizar el procesamiento de los datos y el diagnóstico de riesgo de COVID-19, eligiendo la Raspberry Pi Zero W. De esta manera el paciente puede tomar sus signos vitales y diagnosticar el riesgo de COVID-19 desde su casa, evitando la propagación del virus debido a que todos los datos se visualizan en un servidor web, con la finalidad de que el médico tratante pueda conocer del estado de salud del paciente y en caso de que el dispositivo electrónico recomiende Revisión Médica u Hospitalización, pueda recibir una alerta a su correo electrónico personal indicando su nivel de riesgo.

4.2. Recomendaciones

- Tener en cuenta la disponibilidad de los datos, los mismos que deben tener concordancia con la aplicación que se quiere dar al modelo de Machine Learning. Otro factor importante es la calidad de estos datos, por lo tanto, es necesario hacer una exploración de estos y eliminar datos irrelevantes o vacíos que generen problemas al momento de realizar el modelamiento.

- La capacitancia incrementa en función a la cantidad de periféricos, cables o distancia, esto conlleva a que los flancos de subida y de bajada en la carga del capacitor sean más lentos, por lo tanto, se recomienda utilizar una resistencia cuyo valor se lo calcula con la ecuación de la ley de ohm, en donde el valor de la corriente será la máxima del puerto de los periféricos.

- Para la implementación del suministro de energía portátil para la Raspberry Pi Zero W es de suma importancia conocer sus puertos de alimentación para evitar sobrecargas que dañen el procesador u otros componentes. Se recomienda conectar la salida de la fuente de alimentación al puerto mini USB, debido a que este tiene un circuito de protección en caso de sobrecargas o caídas de voltaje.

- Se recomienda balancear los datos del dataset para que cada clase tenga la misma cantidad de datos, debido a que el desbalanceo puede dar aparición a un sesgo en la clasificación, es decir que el modelo tenga preferencia de predecir la clase con mayor cantidad de datos en contraste con las que tengan menos datos. Esto también puede afectar en las métricas de evaluación especialmente con el método "Accuracy", ya que su medida puede llevar a interpretar mal la eficiencia del modelo cuanto se presenta un desbalanceo de los datos.

- Se recomienda analizar los resultados del prototipo con un profesional de la salud debido a que el prototipo puede presentar errores en sus resultados, ya que este sirve como ayuda para le médico tratante, mas no lo reemplaza.

C. MATERIALES DE REFERENCIA

Referencias Bibliográficas

[1] Statista, «Statista,» 11 03 2022. [En línea]. Available: https://es.statista.com/estadisticas/1104227/numero-acumulado-de-casos-de-coronavirus-covid-19-en-el-mundo-enero-marzo/. [Último acceso: 14 03 2022].

[2] V. Chamola, V. Hassija, V. Gupta y M. and Guizani, «A comprehensive review of the covid-19 pandemic and the role of iot, drones, ai, blockchain, and 5g in managing its impact,» *EEE Access*, vol. 8, p. 90225–90265, 2020.

[3] V. Pappakrishnan, R. Mythili, V. Kavitha y N. Parthiban, «Role of Artificial Intelligence of Things (AIoT) in Covid-19 Pandemic: A Brief Survey,» *Proceedings of the 6th International Conference on Internet of Things, Big Data and Securit,* pp. 229-236, 2021.

[4] C. Graf, «Tecnologías de información y comunicación (TICs). Primer paso para la implementación de TeleSalud y Telemedicina,» *Revista Parguaya de Reumatología,* p. 4, 2020.

[5] H. Rohmetra, N. Raghunath, P. Narang, V. Chamola y M. Guizani, «AI-enabled remote monitoring of vital signs for COVID-19: methods, prospects and challenges,» *Nature Public Health Emergency Collection,* p. 1–27, 2021.

[6] C. Echeverría B, A. Rojas O., A. Serani M., A. Arriagada U., G. Ruiz Esquide, R. Salinas R. y P. Taboada R., «Una reflexión ética sobre la telemedicina,» *Revista médica de Chile,* p. 6, 2021.

[7] H. Chávez y J. Gaybor, «COVID-19, tecnología y poder: los peligros del optimismo tecnológico y el surgimiento del omnióptico global,» *F-iLIA,* p. 39, 2020.

[8] D. A. Haro Esparza, A. D. Aldáz Ibujes, A. E. Santana Alarcón, E. A. Torres Constante, L. R. Aranha Reyes, L. M. Gómez Albán y L. R. Calderón Layedra,

«revalencia de Enfermedades Respiratorias y Comportamiento Epidemiológico de COVID-19 en Pacientes del Centro de Salud “Centro Histórico”,» *INVESTIGATIO,* pp. 25-36, 2020.

[9] G. E. Chanchi, M. A. Ospina y J. L. Pérez, «Sistema IoT para la monitorización de la variabilidad del ritmo cardiaco en pruebas de usabilidad.,» *ESPACIOS,* p. 14, 2020.

[10] J. Moreno, Aplicación de registro y reporte automático de parámetros fisiológicos para la trazabilidad psicofisiológica en personas post COVID-19, Bogotá, 2021.

[11] K. Zhang, X. Liu, J. Shen, Z. Li, Y. Sang, X. Wu y G. Wang, «Clinically Applicable AI System for Accurate Diagnosis, Quantitative Measurements, and Prognosis of COVID-19 Pneumonia Using Computed Tomography,» *Cell,* vol. 181, p. 1423–1433, 2020.

[12] T. Zhongyun, H. Haiyang, X. Chonghuan y Z. Kaidi, «Exploring an Efficient Remote Biomedical Signal Monitoring Framework for Personal Health in the COVID-19 Pandemic,» *Int. J. Environ. Res. Public Health,* vol. 18, nº 9037, pp. 1-23, 2021.

[13] S. Pérez Fernández, Artist, *Sisteme electrónico de monitoreo de signos vitales y alertas de distanciamiento social para la prevención de enfermedades respiratorias.* [Art]. Universidad Técnica de Ambato, 2021.

[14] C. Vargas, R. Acosta y A. Tequen, «El nuevo Coronavirus y la pandemia del Covid-19,» *Rev Med Hered,* vol. 31, pp. 125-131, 2020.

[15] A. Vargas, V. Schreiber, E. Ochoa y A. López, «SARS-CoV-2: una revisión bibliográfica de los temas más relevantes y evolución del conocimiento médico sobre la enfermedad,» *Neumología y Cirugía de Tórax,* vol. 79, nº 3, pp. 185-196, 2020.

[16] N. Patricia y J. Pilicita, «Signos Vitales y cuiddos de enfermería,» *Universidad Central del Ecuador,* 2019.

[17] Redacción médica, «Redacción médica,» [En línea]. Available: https://www.redaccionmedica.com/recursos-salud/faqs-covid19/cual-es-la-temperatura-si-tienes-covid.

[18] Ministerio de Salud Pública , Consenso Multidisciplinario informado en la edidencia sobre el tratamiento de Covid-19, Quito: MSP, 2020.

[19] N. Sebastián, «Gaceta Médica,» 23 09 2020. [En línea]. Available: https://gacetamedica.com/investigacion/caracteristicas-investigacion-y-tratamientos-la-covid-19-de-la-a-a-la-z/.

[20] H. K. &. M. M. R. Siddiqi, «COVID-19 illness in native and immunosuppressed states: A clinical–therapeutic staging proposal,» *The journal of heart and lung transplantation,* vol. 39, nº 5, pp. 405-407, 2020.

[21] Organización Panamericana de la Salud, «Aspectos técnicos y regulatorios sobre el uso de oxímetros de pulso en el monitoreo de pacientes con COVID-19,» Washington, 2020.

[22] J. Guerrero, Bioseñales, Valencia: Universitat de Valencia, 2011.

[23] M. Gómez y C. Lara, «Análisis de bioseñales: Enfoque técnico de la adquisición, procesamiento y sus aplicaciones,» *Innovación y Desarrollo Tecnológico Revista Digital,* vol. 10, nº 2, 2018.

[24] Cardiac Sense, «Cardiac Sense,» [En línea]. Available: https://www.cardiacsense.com/el-brazalete/#:~:text=El%20brazalete%20CardiacSense%20monitoriza%20signos, directamente%20al%20control%20de%20enfermer%C3%ADa.. [Último acceso: 03 08 2022].

[25] P. A. Almeida Vivanco y S. X. Sánchez Velasco, Artists, *Diseño y construcción de un sistema de monitoreo de signos vitales, ubicación, identificación y detección de caídas para adultos mayores.*. [Art]. Universidad de las Fuerzas Armadas (ESPE), 2020.

[26] D. Patiño Vélez, «Diseño de un dispositivo wearable para el monitoreo de la oxigenación y ritmo cardiaco,» *Memorias del Congreso Nacional de Ingeniería Biomédica,* vol. 7, nº 1, pp. 485-492, 2020.

[27] Intel Latinoamérica, «Intel Latinoamérica,» 2020. [En línea]. Available: https://www.intel.la/content/www/xl/es/healthcare-it/telemedicine.html.

[28] A. Galipenso, M. Isabel y M. Quevedo, Inteligencia artificial: modelos, técnicas y áreas de aplicación., Madrid: Paraninfo, 2003.

[29] J. Márquez, «Inteligencia artificial y Big Data como soluciones frente a la COVID-19,» *Revista de Bioética y Derecho,* nº 50, 2020.

[30] V. Román, «Algoritmos Naive Bayes: Fundamentos e Implementación,» 2019. [En línea]. Available: https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fudamentos-e-implementaci%C3%B3n-4bcb24b307f. [Último acceso: 2022].

[31] E. Sucar, Métodos de de inteligencia artificial, Tecnologías de Información UPAEP, 2017.

[32] M. Batta, «Machine Learning Algorithms - A Review,» *International Journal of Science and Research (IJSR),* vol. 9, pp. 381-386, 2018.

[33] J. Bobadilla, Machine Learning y Deep Learning, Bogotá: Ediciones de la U, 2020.

[34] J. Torres, Python Deep Learning: Introducción práctica con Keras y TensorFlow 2, Barcelona: Marcocombo, 2020.

[35] Universidad de Galileo, «Introducción a la Inteligencia Aritificial,» Ciudad de Guatemala, 2020.

[36] J. I. Barrios Arce, 26 07 2019. [En línea]. Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/. [Último acceso: 16 07 2022].

[37] V. Portela, «Solofaq,» 30 05 2022. [En línea]. Available: https://solofaq.com/cuando-usar-de-manera-concisa-en-una-oracion/. [Último acceso: 17 07 2022].

[38] Z. LT, «Towards Data Science,» 23 11 2021. [En línea]. Available: https://towardsdatascience.com/essential-things-you-need-to-know-about-f1-score-dbd973bf1a3. [Último acceso: 01 08 2022].

[39] V. Roman, «Medium,» 18 02 2019. [En línea]. Available: https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48. [Último acceso: 02 08 2022].

[40] A. Polamuri , «Dataaspirant,» 03 12 2020. [En línea]. Available: https://dataaspirant.com/10-k-fold-cross-validation/. [Último acceso: 12 07 2022].

[41] J. J. Sarango Jumbo, Artist, *DESARROLLO DE UNA APLICACIÓN WEB PARA EL ANÁLISIS DEL MODELO DE APRENDIZAJE AUTOMÁTICO QUE MIDE LA EFECTIVIDAD DE RUTINAS DE EJERCICIOS FÍSICOS EN PACIENTES.* [Art]. Universidad de Guayaquil, 2022.

[42] J. Brownlee, «Machine Learning Mastery,» 23 12 2019. [En línea]. Available: https://machinelearningmastery.com/what-is-imbalanced-classification/. [Último acceso: 21 07 2022].

[43] Z. Keita, «Towards Data Science,» 19 02 2021. [En línea]. Available: https://towardsdatascience.com/text-data-representation-with-one-hot-encoding-tf-idf-count-vectors-co-occurrence-vectors-and-

f1bccbd98bef#:~:text=This%20algorithm%20is%20used%20to,being%20a%20single%20distinct%20word.. [Último acceso: 15 07 2022].

[44] «Geeks For Geeks,» 17 07 2020. [En línea]. Available: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/. [Último acceso: 16 06 2022].

[45] Amazon, «Amazon,» [En línea]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html. [Último acceso: 01 08 2022].

[46] Redacción España, «Agencia B12,» 09 11 2020. [En línea]. Available: https://agenciab12.com/noticia/etapas-proceso-machine-learning. [Último acceso: 01 08 2022].

[47] S. Ray, «Analytics Vidhya,» Understanding Support Vector Machine(SVM) algorithm from examples (along with code), 12 09 2017. [En línea]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. [Último acceso: 21 06 2022].

[48] Nadeem, «Medium,» 17 10 2017. [En línea]. Available: https://medium.com/mlearning-ai/support-vector-machine-svm-algorithm-a5acaa48fe3a. [Último acceso: 02 08 2022].

[49] R. Gandhi, «Towards Data Science,» Support Vector Machine — Introduction to Machine Learning Algorithms, 07 06 2018. [En línea]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Último acceso: 22 06 2022].

[50] N. Soroush, T. Garrett, W. Yang y A. Yang, A Comprehensive Guide to Machine Learning, California, 2019.

[51] JavatPoint, «JavatPoint,» [En línea]. Available: https://www.javatpoint.com/machine-learning-naive-bayes-classifier. [Último acceso: 03 08 2022].

[52] Azure, «Azure,» 2021. [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-cloud-storage/.

[53] PiSugar, [En línea]. Available: https://github.com/PiSugar/PiSugar/tree/master/model. [Último acceso: 22 08 2022].

[54] R. Ramirez, «Circuit Basics,» 13 06 2020. [En línea]. Available: https://www.circuitbasics.com/how-to-power-your-raspberry-pi-with-a-lithium-battery/.

Anexos

Anexo A: Certificación médica

CERTIFICACIÓN

Ingeniero, Mg.

Pilar Urrutia

DECANA

Facultad de Ingeniería en Sistemas Electrónica e Industrial

Presente

Señora Decana:

Por medio de la presente, yo Dra. GAVIRIA BOLAÑOS ANGÉLICA PAOLA con cédula de identidad N° 0502347370 con registro del Senescyt N° 1010-2016-1763581, Médico General con especialidad en Medicina Familiar y Comunitaria, certifico que he revisado el proyecto técnico "SISTEMA ELECTRÓNICO DE MONITOREO DE BIOSEÑALES PARA EL DIAGNÓSTICO MÉDICO DE COVID-19 EN PERSONAS MEDIANTE INTELIGENCIA ARTIFICIAL", el mismo que fue diseñado en su total autoría por el Sr. Santiago Adolfo Gómez Lagua, con cédula de identidad N° 1804959698, estudiante de la carrera de Telecomunicaciones de la Universidad Técnica de Ambato.

Después de haber realizado las pruebas médicas necesarias me permito verificar la funcionabilidad del equipo y puedo validar que los resultados obtenidos hasta la fecha son aceptables y de gran ayuda en el área de detección y prevención contra el COVID-19, además indico que los sensores se visualizan dentro de los rangos médicos normales.

Saludos cordiales,

Atentamente,

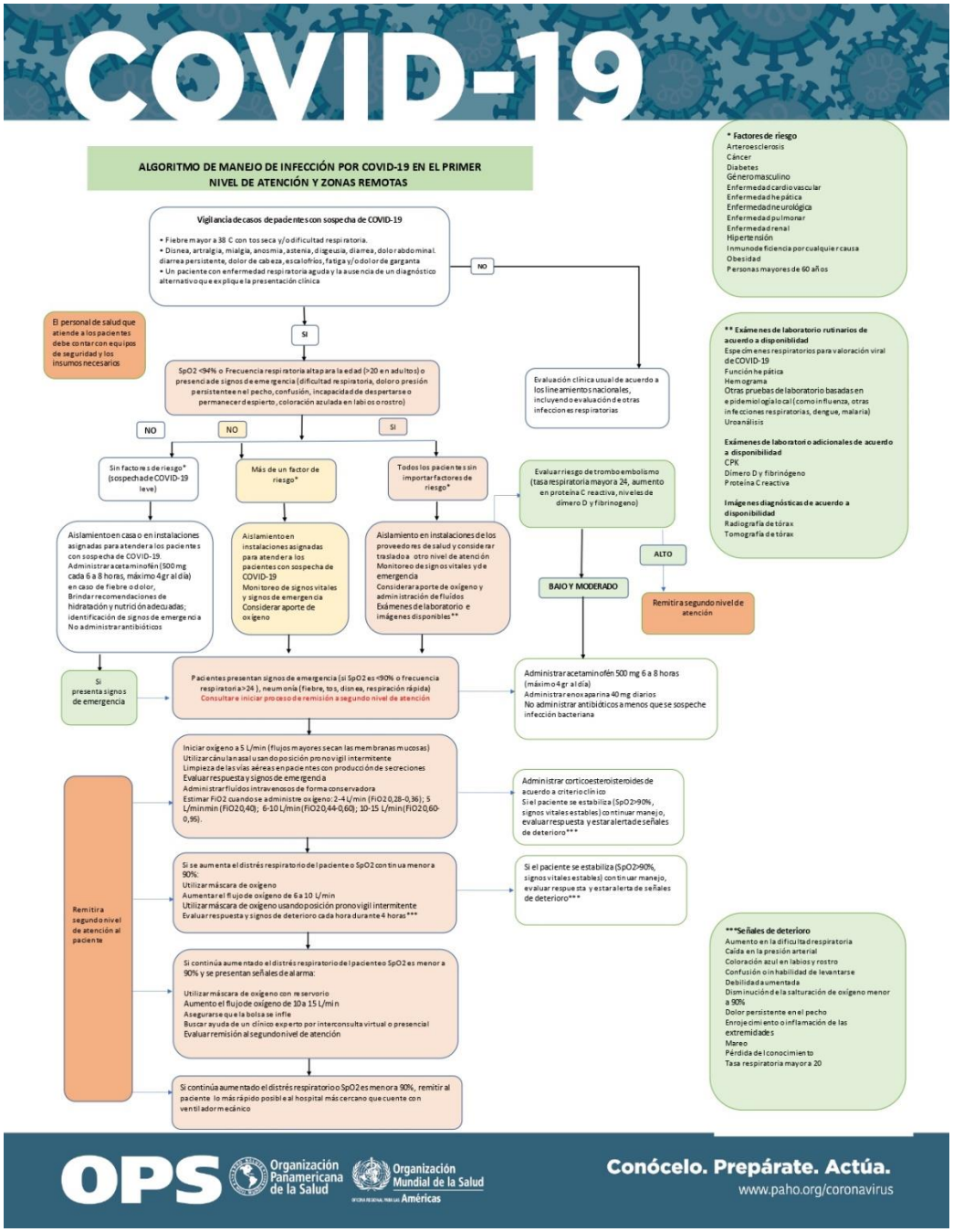
Dra. Paola Gaviria
ESP. MEDICINA FAMILIAR
MED. [illegible] Y CIRUGIA
MSP [illegible] N 403

Dra. Angélica Paola Gaviria Bolaños

Médico Familiar

C.I.: 0502347370

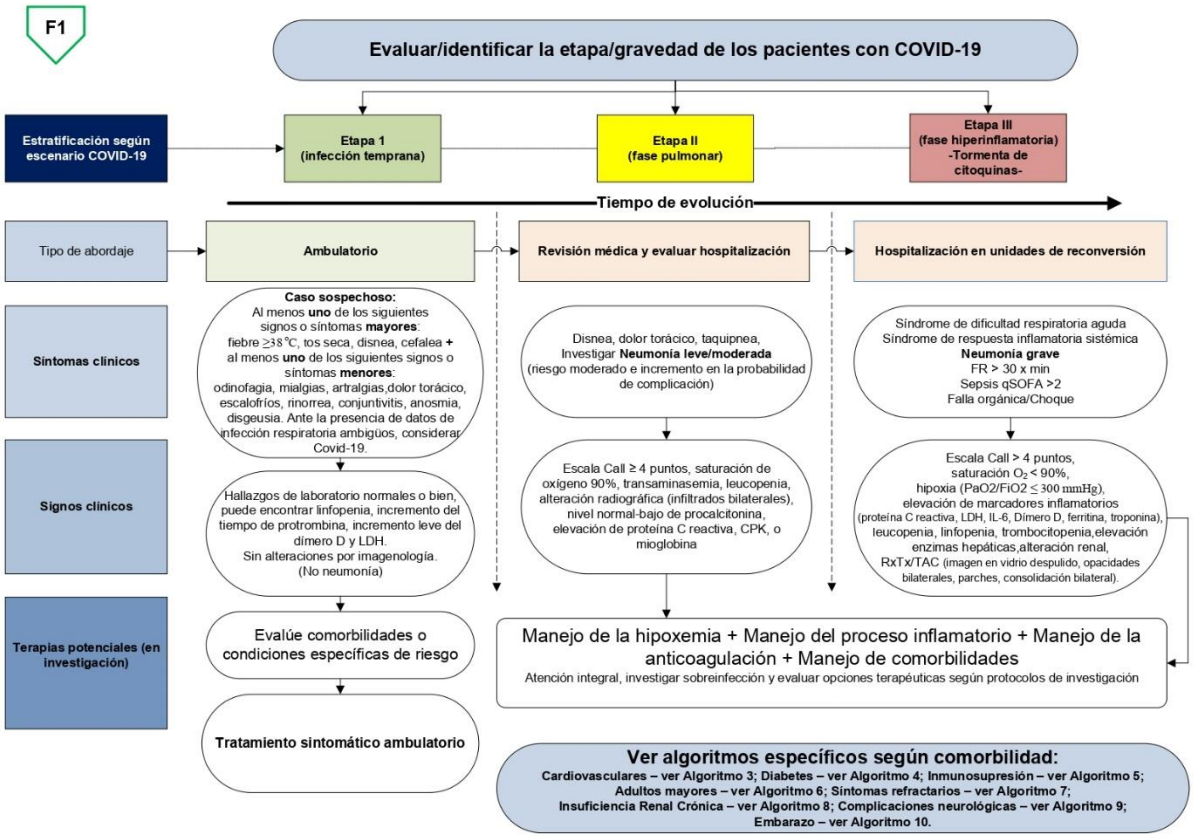
Anexo B: Algoritmo de detección de COVID-19 proporcionado por la Organización Panamericana de Salud



Anexo C: Algoritmo de detección de COVID-19 proporcionado por el Instituto de Seguridad Social de México

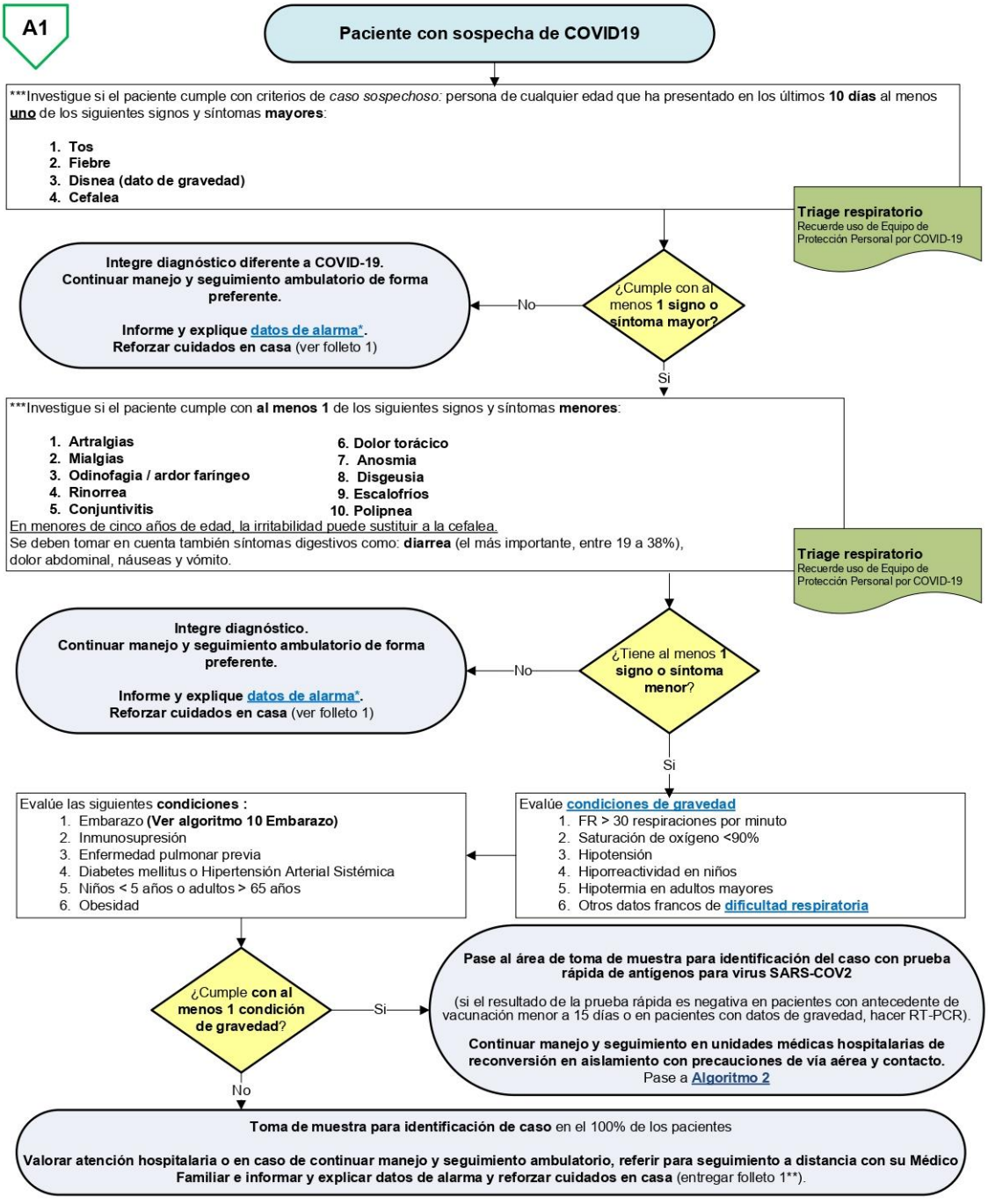
Figura 1. Identificación de casos según gravedad y opciones terapéuticas

Información integrada a partir de consenso



Algoritmo 1. Procedimiento para la atención médica de primer contacto en servicios de salud

Definición de casos sospechoso



*Datos de alarma (mencione los siguientes datos de alarma expresados en lenguaje sencillo): dificultad para respirar, dolor o presión persistente en el pecho, confusión, incapacidad de despertarse o permanecer despierto, coloración azulada en los labios o el rostro.

**Folleto disponible en: http://educacionensalud.imss.gob.mx/es/system/files/Cuidados-en-casa-personas-COVID-19.pdf

***Definición operacional de acuerdo a CONAVE (Comité Nacional para la vigilancia Epidemiológica) 21/08/2020.

Anexo D: Código para la adquisición de datos del sensor MAX30100

```
import time
import os
import max30100
tiempo = 200
def media_movil(numbers):
 promSize = 4
 i = 0
 while i < len(numbers) - promSize + 1:
 porcion = numbers[i : i + promSize]
 promedioPorcion = sum(porcion) / promSize
 i += 1
 try:
 return int((promedioPorcion/100))
 except:
 pass
def corrDatos(promedio_bpm,promedio_sp02):
 try:
 if(promedio_bpm<10):
 promedio_bpm = 0
 promedio_sp02 = 0
 return promedio_bpm, promedio_sp02
 except:
 return promedio_bpm, promedio_sp02
def sensor_max30100():
 pulsosList = []
 spo2List = []
 try:
 mx30 = max30100.MAX30100()
 mx30.enable_spo2()
 for i in range (0,tiempo):
 i-=1
 mx30.read_sensor()
 hb = int(mx30.ir / 100)
 spo2 = int(mx30.red/100)
 if mx30.ir != mx30.buffer_ir :
 promedio_bpm = (media_movil(mx30.buffer_ir))
 if mx30.red != mx30.buffer_red:
 promedio_sp02 = (media_movil(mx30.buffer_red))
 bpm,spo2 = corrDatos(promedio_bpm,promedio_sp02)
 pulsosList.append(bpm)
105
 spo2List.append(spo2)
 time.sleep(.1)
 #print(pulsosList)
 #print(spo2List)
 pulsosList = pulsosList[20:]
 spo2List = spo2List[20:]
 print(pulsosList)
 print(spo2List)
```

```
 global ajusteSpo2
 global ajustePulsos
 global suma1
 suma1=0
 suma2=0
 for i in pulsosList:
 suma1 = suma1+i
 suma1 = round(suma1/len(pulsosList),2)
 ajustePulsos = round((0.4086*suma1+45.348),2)
 print("Pulsos sin ajuste: ",suma1)
 print("Pulsos con ajuste: ",ajustePulsos)
 for i in spo2List:
 suma2 = suma2+i
 suma2 = suma2/len(spo2List)
 ajusteSpo2 = round((0.1447*suma2+74.025),2)
 print("SpO2 sin ajuste: ",suma2)
 print("SpO2 con ajuste: ",ajusteSpo2)
 #return ajustePulsos
 #return ajusteSpo2
 except:
 print("Sensor MAX30100 desconectado")
 ajustePulsos = 0
 ajusteSpo2 = 0
#sensor_max30100()
def varPulso():
 return ajustePulsos
def varSpo2():
 return ajusteSpo2
```

Anexo E: Código para la adquisición de datos del sensor MLX90614

```
import time
from smbus2 import SMBus
from mlx90614 import MLX90614
def sensor_mlx90614():
 tiempo = 200
 tempList = []
 try:
 for i in range(0,tiempo):
 i -= 1
 bus = SMBus(1)
 sensor = MLX90614(bus, address=0x5A)
 tempCorp = sensor.get_obj_temp()
 bus.close()
 time.sleep(.1)
 tempList.append(round(tempCorp,2))
 tempList = tempList[10:]
 #print(tempList)
 global medTemp
 global ajusteTemp
 suma = 0
 for i in tempList:
 suma = suma + i
 medTemp = suma/len(tempList)
 ajusteTemp = round((0.1006*medTemp+33.026),2)
 except:
 ajusteTemp = 0
 print("Sensor no conectado")
def varTemp():
 return ajusteTemp
```

Anexo F: Código para la sincronización entre la base de datos local y en la nube

```
import mysql.connector
import requests
import threading
import time
def sincronizar():
 while True:
 conexion1=mysql.connector.connect(host="localhost",
 user="admin",
passwd="1234",
107
 database="borrador")
 cursor1=conexion1.cursor()
 cursor1.execute("select id, cedula, fecha, pulso, spo2,
temperatura, sexo, vacuna, condiciones_especiales, sintomas,
diagnostico from diagnostico_pacientes")

 resp =
requests.post('https://covidsintomas.000webhostapp.com/eliminar.php')
 for fila in cursor1:
 #print(fila[0])
 userdata = {"id": fila[0], "cedula": fila[1],"fecha":
fila[2],"pulso": fila[3], "spo2": fila[4],"temperatura": fila[5],
"sexo": fila[6],"vacuna": fila[7], "condiciones_especiales":
fila[8],"sintomas": fila[9], "diagnostico": fila[10]}

 resp =
requests.post('https://covidsintomas.000webhostapp.com/insertar.php',
params=userdata)
 conexion1.close()
 time.sleep(2)
 break
```

Anexo G: Código de la función para la sensorización de los signos vitales

```
def signosVitales():
    hilo1 = threading.Thread(target=sensor_max30100)
    hilo2 = threading.Thread(target=sensor_mlx90614)
    hilo1.start()
    hilo2.start()
    font10 =
ImageFont.truetype('/home/santiago/Documentos/Tesis/pruebas_interfaz/Fo
nt.ttf',11)
    with canvas(device) as draw:
        draw.rectangle(device.bounding_box, outline="white",
fill="black")
        draw.text((10,10), 'Tomando los signos', font=font10, fill =
"white")
        draw.text((40,25), 'vitales', font=font10,  fill = "white")

    time.sleep(23)

    global varPulso_0, varSpo2_0, varTemp_0, varTemp_1, varPulso_1,
varSpo2_1, varPsa, varSsa, varTsa
    varPulso_1 = ""
    varSpo2_1 = ""
    varTemp_1 = ""

    varPulso_0 = float(varPulso())
    varSpo2_0 = float(varSpo2())
    varTemp_0 = float(varTemp())

    a = "*Pulso: " + str(varPulso())
    print(a)
    b = "*SpO2: " + str(varSpo2())
    print(b)
    c = "*Temperatura: " + str(varTemp())
    print(c)

    if  varTemp_0 > 37.4:
       varTemp_1 = "Fiebre"
       print(varTemp_1)

    if 39.5 < varTemp_0 <= 40:
       varTemp_1 = "Fiebre alta"
       print(varTemp_1)
       asunto = "Alerta de signos vitales ({})".format(varTemp_1)
       mensaje = "El paciente con cédula {} tiene una temperatura igual
a {}".format(identif, varTemp_0)
       enviarCorreo(asunto, mensaje)

    if varTemp_0 > 40:
       varTemp_1 = "Fiebre muy alta"
       print(varTemp_1)
```

```
        asunto = "Alerta de signos vitales ({})".format(varTemp_1)
        mensaje = "El paciente con cédula {} tiene una temperatura igual
a {}".format(identif, varTemp_0)
        enviarCorreo(asunto, mensaje)

    """
    if varTemp_0 == 0:
        c = "Sensor de temperatura"
        varTemp_1 = "desconectado"
        print(varTemp_1)
    """

    if 87 < varSpo2_0 <= 91:
        varSpo2_1 = "Hipoxia leve"
        print(varSpo2_1)
    if 83 < varSpo2_0 <= 87:
        varSpo2_1 = "Hipoxia moderada"
        asunto = "Alerta de signos vitales ({})".format(varSpo2_1)
        mensaje = "El paciente con cédula {} tiene su nivel de oxígeno
igual a {}".format(identif, varSpo2_0)
        print(varSpo2_1)

    if 1 < varSpo2_0 <= 83:
        varSpo2_1 = "Hipoxia severa"
        asunto = "Alerta de signos vitales ({})".format(varSpo2_1)
        mensaje = "El paciente con cédula {} tiene su nivel de oxígeno
igual a {}".format(identif, varSpo2_0)
        print(varSpo2_1)

    if varPulso_0 > 100:
        varPulso_1 = "Taquicardia"
        print(varPulso_1)
    if 0 < varPulso_0 < 60:
        varPulso_1 = "Bradicardia"
        print(varPulso_1)

    if varSpo2_0 == 0 and varPulso_0 == 0:
        a = "Sensor de pulso"
        b = ""
        varSpo2_1 = ""
        varPulso_1 = "desconectado"
        print(varSpo2_1)

    with canvas(device) as draw:
        draw.rectangle(device.bounding_box, outline="white",
fill="black")
        draw.text((0,0), 'Sus signos vitales:', font=font, fill =
"white")
        draw.text((1,8), a, font=font, fill = "white")
        draw.text((15,16), varPulso_1, font=font, fill = "white")
        draw.text((1,25), b, font=font, fill = "white")
        draw.text((15,34), varSpo2_1, font=font, fill = "white")
```

```
    draw.text((1,43), c, font=font, fill = "white")
    draw.text((15,52), varTemp_1, font=font, fill = "white")
time.sleep(5)

print(varPulso_1)
return varTemp_1
return varSpo2_1
return varPulso_1

return varPulso_0
return varSpo2_0
return varTemp_0
```

Anexo F: Código de la función para el diagnóstico de riesgo de COVID-19

```
def diagnostico(sintoma):
    decision_tree =
pickle.load(open('/home/santiago/Documentos/Tesis/pruebas_ia/modeloDT.p
ickle', 'rb'))
    loaded_vectorizer =
pickle.load(open('/home/santiago/Documentos/Tesis/pruebas_ia/datos.pick
le', 'rb'))

    listaDiag = sintoma
    print("---------------------------------")
    print("Lista original",listaDiag)

     signos = listaDiag[1]
    sigPulso = signos[0]
    sigSpo2 = signos[1]
    sigTemp = signos[2]
    sigPulso_1 = signos[3]
    sigSpo2_1 = signos[4]
    sigTemp_1 = signos[5]

    print("a: ",signos)
    print("b: ",sigPulso)
    print("c: ",sigSpo2)
    print("d: ",sigTemp)
    print("e: ",sigPulso_1)
    print("f: ",sigSpo2_1)
    print("g: ",sigTemp_1)
    ##############################################
    sexo = str(listaDiag[2])[2:-2]
    print("h: ", sexo)
    ##############################################
    vacuna = str(listaDiag[3])[2:-2]
    print("i: ", vacuna)
    ##############################################
    condEsp = str(listaDiag[4])[2:-2]
    condEsp = condEsp.replace("'",'')
    print("j: ", condEsp)
    ##############################################
    sintomas = str(listaDiag[5])[2:-2]
    sintomas = sintomas.replace("'",'')
    print("k: ", sintomas)
    ##############################################

    listaDiag_1 = listaDiag[2:]
    print("Primera lista: ", listaDiag_1)
#       if sigPulso_1 != "":
#          listaDiag_1 = [sigPulso_1] + [sigSpo2_1] + listaDiag_1
    if sigPulso_1 != "desconectado" and sigPulso_1 != '':
       listaDiag_1 = [sigPulso_1] + [sigSpo2_1] + listaDiag_1
```

```
#        if sigTemp_1 != "":
#           listaDiag_1 = [sigTemp_1] + listaDiag_1
    if sigTemp_1 != "desconectado" and sigTemp_1 != '':
       listaDiag_1 = [sigTemp_1] + listaDiag_1

    print("Lista sin sensores desconectados",listaDiag_1)

    listaDiag_1 = str(listaDiag_1)[1:-1]
    listaDiag_1 = listaDiag_1.replace("'",'')
    listaDiag_1 = listaDiag_1.replace("[",'')
    listaDiag_1 = listaDiag_1.replace("]",'')
    print("--------------------------------")
    print("Lista a diagnosticar: ", listaDiag_1)

    prediccion =
decision_tree.predict(loaded_vectorizer.transform([listaDiag_1]))
    prediccion = prediccion[0]

    if prediccion == "ambulatorio":
       rec = "Tome cuidado en casa"
       prediccion = "Ambulatorio"
       #enviarCorreo(prediccion, rec)
    if prediccion == "revision_medica":
       rec = "Consulte con un médico"
       prediccion = "Revisión médica"
       asunto = "Alerta de riesgo de COVID-19"
       mensaje = "El paciente con cédula {} tiene un diagnóstico que
amerita a {}".format(identif, prediccion)
       enviarCorreo(asunto, mensaje)
    if prediccion == "hospitalizacion":
       rec = "Acuda a urgencias"
       prediccion = "Hospitalización"
       asunto = "Alerta de riesgo de COVID-19"
       mensaje = "El paciente con cédula {} tiene un diagnóstico que
amerita a {}".format(identif, prediccion)
       enviarCorreo(asunto, mensaje)

    print("El diagnóstico es: ", prediccion)
    print("Antepenultimo: ",condEsp)

    with canvas(device) as draw:
        draw.rectangle(device.bounding_box, outline="white",
fill="black")
        draw.text((5,0), 'Su diagnótico es:', font=font, fill =
"white")
        draw.text((5,15), prediccion, font=font,  fill = "white")
        draw.text((5,30), rec, font=font, fill = "white")
    time.sleep(3)

    fechaActual = datetime.datetime.now()
    fechaActual = fechaActual.strftime('%d/%m/%Y %H:%M:%S')
    print(fechaActual)
```

```
    diagnostico = prediccion


dbLocal(identif,fechaActual,sigPulso,sigSpo2,sigTemp,sexo,vacuna,condEs
p,sintomas,diagnostico)

    try:
       sincronizar()
    except:
       print("No se pudo sinronizar")
       pass

    pagina = 0
    cursor = 2
```

Anexo G: Ficha técnica del sensor MAX30100

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
IR ADC Count—PSRR (V_{DD})	$PSRR_{VDD}$	Propriety ATE setup 1.7V < V_{DD} < 2.0V, LED_PW = 0x03, SPO2_SR = 0x01, IR_PA = 0x09, IR_PA = 0x05, T_A = +25°C		0.25	2	%
		Frequency = DC to 100kHz, 100mV_{P-P}		10		LSB
RED/IR ADC Count—PSRR (X_LED+)	$PSRR_{LED}$	Propriety ATE setup 3.1V < X_LED+ < 5V, LED_PW = 0x03, SPO2_SR = 0x01, IR_PA = 0x09, IR_PA = 0x05, T_A = +25°C		0.05	2	%
		Frequency = DC to 100kHz, 100mV_{P-P}		10		LSB
ADC Integration Time	INT	LED_PW = 0x00		200		µs
		LED_PW = 0x03		1600		µs
IR LED CHARACTERISTICS (Note 4)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	870	880	900	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		30		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		1.4		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		6.5		mW
RED LED CHARACTERISTICS (Note 4)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	650	660	670	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		20		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		2.1		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		9.8		mW
TEMPERATURE SENSOR						
Temperature ADC Acquisition Time	T_T	T_A = +25°C		29		ms
Temperature Sensor Accuracy	T_A	T_A = +25°C		±1		°C
Temperature Sensor Minimum Range	T_{MIN}			-40		°C
Temperature Sensor Maximum Range	T_{MAX}			85		°C

Absolute Maximum Ratings

V_{DD} to GND .....-0.3V to +2.2V
GND to PGND .....-0.3V to +0.3V
x_DRV, x_LED+ to PGND .....-0.3V to +6.0V
All Other Pins to GND .....-0.3V to +6.0V
Output Short-Circuit Current Duration .....Continuous
Continuous Input Current into Any Terminal .....±20mA

Continuous Power Dissipation (T_A = +70°C)
OESIP (derate 5.8mW/°C above +70°C) 464mW
Operating Temperature Range..... -40°C to +85°C
Soldering Temperature (reflow).....+260°C
Storage Temperature Range..... -40°C to +105°C

Package Thermal Characteristics (Note 1)

OESIP
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....150°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) .....170°C/W

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

Electrical Characteristics

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY							
Power-Supply Voltage	V_{DD}	Guaranteed by RED and IR count tolerance		1.7	1.8	2.0	V
LED Supply Voltage (R_LED+ or IR_LED+ to PGND)	V_{LED+}	Guaranteed by PSRR of LED Driver		3.1	3.3	5.0	V
Supply Current	I_{DD}	SpO_2 and heart rate modes, PW = 200µs, 50sps			600	1200	µA
		Heart rate only mode, PW = 200µs, 50sps			600	1200	
Supply Current in Shutdown	I_{SHDN}	T_A = +25°C, MODE = 0x80			0.7	10	µA
SENSOR CHARACTERISTICS							
ADC Resolution					14		bits
Red ADC Count (Note 3)	RED_C	Propriety ATE setup RED_PA = 0x05, LED_PW = 0x00, SPO2_SR = 0x07, T_A = +25°C		23,000	26,000	29,000	Counts
IR ADC Count (Note 3)	IR_C	Propriety ATE setup IR_PA = 0x09, LED_PW = 0x00, SPO2_SR = 0x07, T_A = +25°C		23,000	26,000	29,000	Counts
Dark Current Count	DC_C	RED_PA = IR_PA = 0x00, LED_PW = 0x03, SPO2_SR = 0x01			0	3	Counts
DC Ambient Light Rejection (Note 4)	ALR	Number of ADC counts with finger on sensor under direct sunlight (100K lux) LED_PW = 0x03, SPO2_SR = 0x01	RED LED		0		Counts
			IR LED		0		

EVALUATION KIT AVAILABLE

MAX30100

Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health

General Description

The MAX30100 is an integrated pulse oximetry and heart-rate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals.

The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

Applications

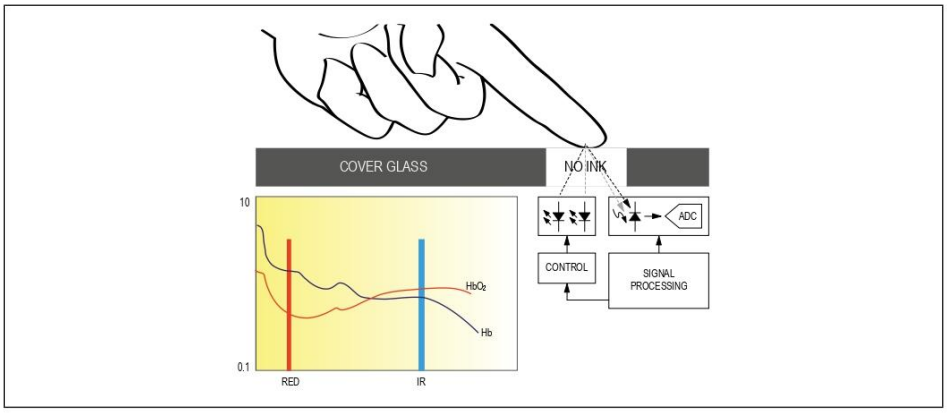
- Wearable Devices
- Fitness Assistant Devices
- Medical Monitoring Devices

Benefits and Features

- Complete Pulse Oximeter and Heart-Rate Sensor Solution Simplifies Design
  - Integrated LEDs, Photo Sensor, and High-Performance Analog Front -End
  - Tiny 5.6mm x 2.8mm x 1.2mm 14-Pin Optically Enhanced System-in-Package
- Ultra-Low-Power Operation Increases Battery Life for Wearable Devices
  - Programmable Sample Rate and LED Current for Power Savings
  - Ultra-Low Shutdown Current (0.7µA, typ)
- Advanced Functionality Improves Measurement Performance
  - High SNR Provides Robust Motion Artifact Resilience
  - Integrated Ambient Light Cancellation
  - High Sample Rate Capability
  - Fast Data Output Capability

Ordering Information **appears at end of data sheet.**

System Block Diagram



19-7065; Rev 0; 9/14

maxim integrated™

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{IR_LED+} = V_{R_LED+} = 3.3V, T_A = +25°C, min/max are from T_A = -40°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DIGITAL CHARACTERISTICS (SDA, SDA, $\overline{INT}$)						
Output Low Voltage SDA, $\overline{INT}$	V_{OL}	I_{SINK} = 6mA			0.4	V
I^2C Input Voltage Low	V_{IL_I2C}	SDA, SCL			0.4	V
I^2C Input Voltage High	V_{IH_I2C}	SDA, SCL	1.4			V
Input Hysteresis	V_{HYS}	SDA, SCL		200		mV
Input Capacitance	C_{IN}	SDA, SCL		10		pF
Input Leakage Current	I_{IN}	V_{IN} = 0V, T_A = +25°C (SDA, SCL, INT)		0.01	1	µA
		V_{IN} = 5.5V, T_A = +25°C (SDA, SCL, INT)		0.01	1	µA
I^2C TIMING CHARACTERISTICS (SDA, SDA, $\overline{INT}$)						
I^2C Write Address				AE		Hex
I^2C Read Address				AF		Hex
Serial Clock Frequency	f_{SCL}		0		400	kHz
Bus Free Time Between STOP and START Conditions	t_{BUF}		1.3			µs
Hold Time (Repeated) START Condition	$t_{HD,START}$		0.6			µs
SCL Pulse-Width Low	t_{LOW}		1.3			µs
SCL Pulse-Width High	t_{HIGH}		0.6			µs
Setup Time for a Repeated START Condition	$t_{SU,START}$		0.6			µs
Data Hold Time	$t_{HD,DAT}$		0		900	ns
Data Setup Time	$t_{SU,DAT}$		100			ns
Setup Time for STOP Condition	$t_{SU,STOP}$		0.6			µs
Pulse Width of Suppressed Spike	t_{SP}		0		50	ns
Bus Capacitance	C_B				400	pF
SDA and SCL Receiving Rise Time	t_R		20 + 0.1C_B		300	ns
SDA and SCL Receiving Fall Time	t_{RF}		20 + 0.1C_B		300	ns
SDA Transmitting Fall Time	t_{TF}		20 + 0.1C_B		300	ns

Note 2: All devices are 100% production tested at T_A = +25°C. Specifications over temperature limits are guaranteed by Maxim Integrated's bench or proprietary automated test equipment (ATE) characterization.

Note 3: Specifications are guaranteed by Maxim Integrated's bench characterization and by 100% production test using proprietary ATE setup and conditions.

Note 4: For design guidance only. Not production tested.

Anexo G: Ficha técnica del sensor MLX90614

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
SMBus compatible 2-wire interface[2]						
Input high voltage	V_{IH}		1.6	2	2.4	V
Input high voltage	V_{IH}(Ta,V)	Over temperature and supply	1.2	2	2.8	V
Input low voltage	V_{IL}		0.7	1.0	1.3	V
Input low voltage	V_{IL}(Ta,V)	Over temperature and supply	0.5	1.0	1.5	V
Output low voltage	V_{OL}	SDA pin in open drain mode, over temperature and supply, Isink = 2mA			0.25	V
SCL leakage	I_{SCL},leak	V_{SCL}=3V, Ta=+85°C			20	uA
SDA leakage	I_{SDA},leak	V_{SDA}=3V, Ta=+85°C			0.25	uA
SCL capacitance	C_{SCL}				10	pF
SDA capacitance	C_{SDA}				10	pF
Slave address	SA	Factory default		5Ah		hex
SMBus Request	t_{REQ}	SCL low	1.024			ms
Timeout,low	$T_{timeout,L}$	SCL low			30	ms
Timeout, high	$T_{timeout,H}$	SCL high			50	us
Acknowledge setup	Tsuac(MD	8-th SCL falling edge, Master	0.5		1.5	us
Acknowledge hold	Thdac(MD	9-th SCL falling edge, Master	1.5		2.5	us
Acknowledge setup	Tsuac(SD)	8-th SCL falling edge, Slave	2.5			us
Acknowledge hold	Thdac(SD	9-th SCL falling edge, Slave	1.5			us
EEPROM						
Data retention		Ta = +85°C	10			years
Erase/write cycles		Ta = +25°C	100,000			Times
Erase/write cycles		Ta = +125°C	10,000			Times
Erase cell time	Terase			5		ms
Write cell time	Twrite			5		ms

Note: refer to MLX90614Axx notes.

6.2 MLX90614Bxx

All parameters are preliminary for T_A = 25 ˚C, **V_{DD} =3V** (unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
Supplies						
External supply	V_{DD}		2.4	3	3.6	V
Supply current	I_{DD}	No load			1	mA
Supply current (programming)	I_{DDpr}	No load, erase/write EEPROM operations			1.5	mA
Power-down supply current	Isleep	no load	1	2.5	5	uA
Power-down supply current	Isleep	Full temperature range	1	2.5	6	uA
Power On Reset						
POR level	V_{POR}	Power-up, power-down and brown-out	1.6	1.85	2.1	V
V_{DD} rise time	T_{POR}	Ensure POR signal			1	ms
Output valid	Tvalid	After POR		0.15		s
Pulse width modulation[1]						
PWM resolution	PWMres	Data band		10		bit
PWM output period	$PWM_{T,def}$	Factory default, internal oscillator factory calibrated		1.024		ms
PWM period stability	$dPWM_T$	Internal oscillator factory calibrated, over the entire operation range and supply voltage	-4		+4	%
Output high Level	PWM_{HI}	I_{source} = 2 mA	V_{DD}-0.25			V
Output low Level	PWM_{LO}	I_{sink} = 2 mA			V_{SS}+0.25	V
Output drive current	$Idrive_{PWM}$	Vout,H = V_{DD} - 0.8V		15		mA
Output sink current	$Isink_{PWM}$	Vout,L = 0.8V		15		mA
Output settling time	Tset	100 pF capacitive load, full operating Ta range			150	ns
Output settling time	$Tset_{RC}$	220 Ohm in series with 47nF load on the wire, full Ta operating range		500	TBD	ns

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
SMBus compatible 2-wire interface[2]						
Input high voltage	V_{IH}		1.8	2	2.2	V
Input high voltage	V_{IH}(Ta,V)	Over temperature and supply	1.6		2.4	V
Input low voltage	V_{IL}		0.7	1.0	1.3	V
Input low voltage	V_{IL}(Ta,V)	Over temperature and supply	0.5		1.5	V
Output low voltage	V_{OL}	SDA pin in open drain mode, over temperature and supply, Isink = 2mA			0.2	V
SCL leakage	I_{SCL},leak	V_{SCL}=4V, Ta=+85°C			30	uA
SDA leakage	I_{SDA},leak	V_{SDA}=4V, Ta=+85°C			0.3	uA
SCL capacitance	C_{SCL}				10	pF
SDA capacitance	C_{SDA}				10	pF
Slave address	SA	Factory default		5Ah		hex
SMBus Request	t_{REQ}	SCL low	1.024			ms
Timeout, low	$T_{imeout,L}$	SCL low			30	ms
Timeout, high	$T_{imeout,H}$	SCL high			50	us
Acknowledge setup time	Tsuac(MD)	8-th SCL falling edge, Master	0.5		1.5	us
Acknowledge hold time	Thdac(MD)	9-th SCL falling edge, Master	1.5		2.5	us
Acknowledge setup time	Tsuac(SD)	8-th SCL falling edge, Slave	2.5			us
Acknowledge hold time	Thdac(SD)	9-th SCL falling edge, Slave	1.5			us
EEPROM						
Data retention		Ta = +85°C	10			years
Erase/write cycles		Ta = +25°C	100,000			Times
Erase/write cycles		Ta = +125°C	10,000			Times
Erase cell time	Terase			5		ms
Write cell time	Twrite			5		ms

Notes: All the communication and refresh rate timings are given for the nominal calibrated HFO frequency and will vary with this frequency's variations.
1. All PWM timing specifications are given for single PWM output (factory default for MLX90614xAx). For the extended PWM output (factory default for the MLX90614xBx) each period has twice the timing specifications (refer to the PWM detailed description section). With large capacitive load lower PWM frequency is recommended. Thermal relay output (when configured) has the PWM DC specification and can be programmed as push-pull, or NMOS open drain. PWM is free-running, power-up factory default is SMBus, refer to 7.6, "Switching between PWM and SMBus communication" for details..
2. For SMBus compatible interface on 12V application refer to Application information section. SMBus compatible interface is described in details in the SMBus detailed description section. Maximum number of MLX90614xxx devices on one bus is 127, higher pullup currents are recommended for higher number of devices, faster bus data transfer rates, and increased reactive loading of the bus.
MLX90614xxx is always a slave device on the bus. MLX90614xxx can work in both low-power and high-power SMBus communication.
All voltage are with respect to the Vss (ground) unless otherwise noted.

Power saving mode is not available on the 5V version (MLX90614Axx).

6 Electrical Specifications

6.1 MLX90614Axx

All parameters are preliminary for T_A = 25 °C, **V_{DD} =5V** (unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
Supplies						
External supply	V_{DD}		4.5	5	5.5	V
Supply current	I_{DD}	No load			1	mA
Supply current (programming)	I_{DDpr}	No load, erase/write EEPROM operations			1.5	mA
Zener voltage	Vz	Iz = 75…400 uA	5.6	5.75	5.8	V
Zener voltage	Vz(Ta)	Iz=70…400uA, full temperature range	5.15	5.75	6.24	V
Power On Reset						
POR level	V_{POR}	Power-up, power-down and brown-out	2.7	3.0	3.3	V
V_{DD} rise time	T_{POR}	Ensure POR signal			3	ms
Output valid (result in RAM)	Tvalid	After POR		0.15		s
Pulse width modulation[1]						
PWM resolution	PWMres	Data band		10		bit
PWM output period	$PWM_{T,def}$	Factory default, internal oscillator factory calibrated		1.024		ms
PWM period stability	$dPWM_T$	Internal oscillator factory calibrated, over the entire operation range and supply voltage	-4		+4	%
Output high Level	PWM_{HI}	I_{source} = 2 mA	V_{DD}-0.2			V
Output low Level	PWM_{LO}	I_{sink} = 2 mA			V_{SS}+0.2	V
Output drive current	$Idrive_{PWM}$	Vout,H = V_{DD} - 0.8V		20		mA
Output sink current	$Isink_{PWM}$	Vout,L = 0.8V		20		mA
Output settling time	Tset	100 pF capacitive load, full operating Ta range		500	TBD	ns
Output settling time	$Tset_{RC}$	220 Ohm in series with 47nF load on the wire, full Ta operating range	20		50	us