UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

"SISTEMA EMBEBIDO PARA LA DETECCION DEL ANGULO CIFOTICO Y LUMBAR POR MEDIO DE VISION ARTIFICIAL"

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

AUTOR: JONATHAN FABRICIO GARCÍA NAVARRO

DIRECTOR: MSC. JAIME ROBERTO MICHILENA CALDERÓN

ASESOR: MSC. LUIS EDILBERTO SUÁREZ ZAMBRANO

IBARRA - ECUADOR

2023

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL CONTACTO	
Cédula de identidad	100369542-4
Apellidos y nombres	García Navarro Jonathan Fabricio
Dirección	Atuntaqui, Andrade Marín “Eloy Alfaro e Imbabura #2”
E-mail	jfgarcia@utn.edu.ec
Teléfono móvil	0969091439
DATOS DE LA OBRA	
Titulo	**“SISTEMA EMBEBIDO PARA LA DETECCION DEL ANGULO CIFOTICO Y LUMBAR POR MEDIO DE VISION ARTIFICIAL”**
Autor	García Navarro Jonathan Fabricio
Fecha	10/05/2023
Programa	Pregrado
Titulo	Ingeniero en Electrónica y Redes de Comunicación
Director	Ing. Jaime Michilena Calderón, MSC

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 17 días del mes de mayo. de 2023

EL AUTOR

García Navarro Jonathan Fabricio

CI: 100369542-4

UNIVERSIDAD TÉCNICA DEL NORTE FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

MAGISTER JAIME MICHILENA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que, el presente trabajo de Titulación **"SISTEMA EMBEBIDO PARA LA DETECCION DEL ANGULO CIFOTICO Y LUMBAR POR MEDIO DE VISION ARTIFICIAL"** Ha sido desarrollado por el señor Jonathan Fabricio García Navarro bajo mi supervisión.

Es todo en cuanto puedo certificar en honor de la verdad

.....

MSc. Jaime Michilena Calderón

CI: 100219843-8

DIRECTOR

DEDICATORIA

Primeramente, dedico este trabajo de titulación a DIOS, quien en lo largo de mi vida me dio las fuerzas necesarias para superar los obstáculos que se han presentado a lo largo de mi vida personal y académica, quien con su bendición me a resguardado en esta vida para cumplir con mis propósitos.

Dedico este trabajo especialmente a mi madre por su apoyo, consejos, compresión y por ayudarme con los recursos necesarios para estudiar. Lo dedico también a mi familia que, con su amor incondicional, comprensión y nunca rendirse conmigo siempre me apoyaron.

También de manera muy especial a las personas que me apoyaron hasta llegar a mi meta, compartiendo sus conocimientos para mi formación profesional y personal, por su tiempo, sabiduría, paciencia y apoyo.

Jonathan Fabricio García Navarro

AGRADECIMIENTOS

Agradezco a mi familia por siempre animarme a continuar con mis estudios y proyectos, por ayudarme sin esperar nada a cambio, por siempre estar presentes en los buenos y aún más en los malos momentos, no lo habría logrado sin ustedes, gracias.

Agradezco también a mis amigos con quienes a lo largo de estos años e compartido experiencias, tristezas y alegrías obteniendo grandes recuerdos y anécdotas. También cabe resaltar un agradecimiento a las personas que formaron en algún momento parte de mi vida, pero se fueron quedando en el camino gracias.

A mi director de tesis por no solo ser un guía académico sino un amigo que gracias a sus consejos, orientación e incondicional disponibilidad me permitió crecer como persona y cumplir con el desarrollo de mi proyecto de titulación.

Por último, y más importante quiero agradecerme a mí mismo, por creer siempre en mí superando cada traba de la vida sin rendirme, por tropezar y levantarme siempre a pesar de las dificultades que se me presentaron, por trabajar duro y constantemente en cumplir mis objetivos, por ser como doy, gracias.

ÍNDICE

IDENTIFICACIÓN DE LA OBRA.....I
CONSTANCIAS.....II
CERTIFICADO.....III
DEDICATORIA.....IV
AGRADECIMIENTOS.....V
ÍNDICE.....VI
ÍNDICE DE FIGURAS.....X
ÍNDICE DE TABLAS.....XIV
ÍNDICE DE ANEXOS.....XV
RESUMEN.....XVI
ABSTRAC.....XVII
Capítulo 1. Antecedentes.....1
1.1. Tema.....1
1.2. Problema.....1
1.3. Objetivos.....3
1.3.1. Objetivo general.....3
1.3.2. Objetivos específicos.....3
1.4. Alcance.....4
1.5. Justificación.....6
Capítulo 2. Fundamentación Teórica.....9
2.1. Fisioterapia.....9
2.1.1 Columna Vertebral.....10
2.1.1 Curvatura Vertebral.....11

2.1.2 Anomalías en la columna vertebral 11

2.1.3 Alteraciones en el plano sagital 13

2.1.4 Métodos de diagnóstico Columna Vertebral 15

2.2. Sistemas Embebidos 19

2.2.1. Componentes 19

2.2.2. Tipos de Sistemas embebidos 20

2.2.3 Placa Embebidas 22

2.2.4 Sistema de Captura 25

2.2.5 Visión Embebida 25

2.3. Inteligencia Artificial 26

2.3.1 Machine Learning 27

2.3.2 Vision Artificial 31

2.3.3 Aplicaciones 32

2.3.4 Procesamiento de Imágenes 35

2.3.5 Etapas Vision Artificial 36

2.4. Software Libre 49

2.4.1 Lenguajes de Programación 49

2.4.2 Librerías de Vision Artificial 50

2.5. Estándar IEEE 29148 52

2.6. Metodología 53

2.6.1 Método de Cascada 53

Capítulo 3. Diseño del Sistema 56
3.1. Fase 1: Análisis de requerimientos 56
3.2. Clínica MediFisio 56
3.3. Análisis 57
3.3.1. Situación Actual 57
3.3.2. Técnicas de Investigación 58
3.3.3. Propósito y Ámbito del Sistema 58
3.3.4. Descripción General del Sistema 59
3.3.5. Características del Sistema 60
3.4. Especificación de Requerimientos 61
3.4.1. Stakeholders 62
3.4.2. Nomenclatura de los Requerimientos 62
3.4.3. Requerimientos de Stakeholders 63
3.4.4. Requerimientos del Sistema 64
3.4.5. Requerimientos de Arquitectura 65
3.5. Fase 2 Diseño del Sistema 66
3.5.1. Elección de Hardware 66
3.5.2. Elección de Software 69
3.6 Diagrama de bloques general del sistema 71
3.7. Diseño del Sistema 72
3.7.1. Arquitectura del Sistema 72
3.8. Diseño de Hardware 74

3.8.1 Subsistema de Adquisición.....75

3.8.2 Alimentación del Sistema.....77

3.9. Diseño de Software.....79

3.9.1 Subsistema de Procesamiento.....79

3.9.2 Subsistema de Almacenamiento.....83

3.9.3 Subsistema de Visualización.....90

Capítulo 4. Implementación y Pruebas de Funcionamiento.....98

4.1. Fase 3: Implementación del Sistema.....98

4.1.1 Prueba 1: Verificación del Hardware y Entorno de trabajo.....98

4.1.2 Prueba 2: Verificación del Software Sistema.....105

4.2. Fase 4: Validación del Sistema.....123

4.2.1. Pruebas 3: Adquisición de Datos.....124

4.2.2. Resultados Obtenidos.....132

4.3. Fase 5: Mantenimiento del Sistema.....133

Conclusiones.....134

Recomendaciones.....135

Referencias.....137

Anexos.....143

ÍNDICE DE FIGURAS

Figura 1 Columna vertebral 10

Figura 2 Anomalías Columna vertebral 14

Figura 3 Método de Ángulo de Cobb 16

Figura 4 Inclinómetro de fluido 17

Figura 5 Método Flecha Sagital 17

Figura 6 Nvidia Jetson Nano 22

Figura 7 Nicla Vision 23

Figura 8 Coral Dev Board 24

Figura 9 Cámara web 25

Figura 10 Aprendisaje Supervisado 28

Figura 11 Aprendisaje Supervisado 29

Figura 12 Etapas de Visión artificial 36

Figura 13 Proceso de adquisición de imágenes digitales 37

Figura 14 Imagen con 256 niveles de intensidad 38

Figura 15 Diferentes Ruidos afectando la imagen 42

Figura 16 Tipos de filtro y mecanismos de aplicación 43

Figura 17 Segmentación de una Imagen 44

Figura 18 Transformada de Hough circular 48

Figura 19 Diagrama Metodología de cascada 55

Figura 20 Diagrama de Bloques General del Sistema 71

Figura 21 Arquitectura del Sistema 72

Figura 22 Diagrama de Conexión 76

Figura 23 Diagrama de Flujo Subsistema de Adquisición 77

Figura 24 Entradas de carga Nvidia Jetson Nano 78

Figura 25 Conector J28 Nvidia Jetson Nano 78

Figura 26 Diagrama de flujo evaluación..... 83

Figura 27 Comando para actualizar repositorio..... 84

Figura 28 Comando instalar paquetes de sqlite3 84

Figura 29 Comando para instalar entorno grafico sqlite3..... 85

Figura 30 Verificación de instalación de sqlite3 85

Figura 31 Interfaz gráfica SQLite..... 86

Figura 32 Código de conexión a la BBDD..... 86

Figura 33 Diagrama de flujo Subsistema de Almacenamiento..... 87

Figura 34 Modelo de Relación BBDD 89

Figura 35 Registro del Paciente..... 90

Figura 36 Interfaz de adquisición de imágenes 91

Figura 37 Obtención de distancias..... 92

Figura 38 Interfaz datos del paciente..... 93

Figura 39 Visualización de información del paciente 94

Figura 40 Interfaz de modificación de Base de Datos. 95

Figura 41 Interfaz de edición de Datos. 96

Figura 42 Entorno de adquisición. 99

Figura 43 Entorno de Adquisición 99

Figura 44 Marcadores a utilizar. 100

Figura 45 Marcadores de ángulo cifolumbar..... 101

Figura 46 Marcadores Flecha Sagital 102

Figura 47 Interfaz de Adquisición 105

Figura 48 Librerías Python 106

Figura 49 Interfaz de registro..... 107

Figura 50 Código de adquisición imagen 108

Figura 51 Código guardar base de datos 109

Figura 52 Interfaz de registro 110

Figura 53 Código validaciones 110

Figura 54 Código patrones de validación 111

Figura 55 Código de guardar cambios 111

Figura 56 Código guardar en base de datos 112

Figura 57 Código de detección de ángulos 113

Figura 58 Interfaz de obtención de ángulos 113

Figura 59 Angulo Cifotico 114

Figura 60 Angulo Lumbar 115

Figura 61 Código de perspectiva 115

Figura 62 Código de detección de marcadores 116

Figura 63 Interfaz de flechas sagital 117

Figura 64 Detección de flechas sagitales 117

Figura 65 Interfaz de evaluación 118

Figura 66 Creación tabla de registro 119

Figura 67 Creación tabla de imágenes 120

Figura 68 Tabla de resultados 120

Figura 69 Verificación creación de tablas 121

Figura 70 Código de conexión a la base de datos 121

Figura 71 Código para guardar información en la base de datos 122

Figura 72 Verificación de la información en la base de datos 122

Figura 73 Verificación de la base de datos de imágenes 123

Figura 74 Método de medición Sistema 125

Figura 75 Método de medición Manual.....126

Figura 76 Método de Flechas Sagitales 128

Figura 77 Método de Flechas Sagitales manual 129

ÍNDICE DE TABLAS

Tabla 1 Clasificación de las etiologías de las desalineaciones Sagitales 13

Tabla 2 Campos aplicativos de la Visión Artificial 32

Tabla 3 Stakeholders del proyecto 62

Tabla 4 Nomenclatura de los requerimientos 62

Tabla 5 Prioridad de los Requerimientos del sistema 63

Tabla 6 Requerimientos Stakeholders STRS 63

Tabla 7 Requerimientos Stakeholders SYSR 64

Tabla 8 Requerimientos Stakeholders SYSH 65

Tabla 9 Elección de la placa visión artificial 67

Tabla 10 Características técnicas de Nvidia Jetson Nano 67

Tabla 11 Selección cámara de adquisición 68

Tabla 12 Selección lenguajes de programación 70

Tabla 13 Selección Base de Datos 70

Tabla 14 Cronograma de pruebas 104

Tabla 15 Rangos de evaluación 124

Tabla 16 Tabla de evaluación Cifosis 127

Tabla 17 Tabla de evaluación Lumbar 127

Tabla 18 Tabla Flechas Sagitales 130

Tabla 19 Tabla de Indicé de Cifosis 131

Tabla 20 Tabla de índice lumbar 131

ÍNDICE DE ANEXOS

ANEXO A. Certificación de pruebas de funcionamiento 143

ANEXO B. Pacientes del Centro de Fisioterapia 144

ANEXO C. Script Interfaz Principal 146

ANEXO D. Script Medición de Ángulos 154

ANEXO E. Medición Flechas Sagital 156

ANEXO F. Script Registro Pacientes 160

RESUMEN

En el presente proyecto está enfocado en la realización de un sistema embebido capaz de medir el ángulo cifolumbar en la región de la columna vertebral para la evaluación de fisioterapia, el cual estará desarrollado mediante visión artificial el cual nos permite la manipulación de imágenes o videos en tiempo real, es así como por medio de algoritmos podemos reconocer puntos de interés en la persona formando un esquema en 2D para su procesamiento.

De tal manera el sistema con los algoritmos previamente desarrollados puede detectar los marcadores colocados por el fisioterapeuta en la región de la espalda, permitiendo a este evaluar los diferentes curvaturas y distancias en el plano sagital, obteniendo así una lectura clara de los ángulos y distancias en tiempo real, al mismo tiempo establecer un registro medico en digital el cual permite guardar la información tanto del paciente como de su respectiva evaluación en una base de datos local.

Con lo referente a las pruebas de funcionamiento del sistema se realizaron bajo la supervisión de un experto en el área de fisioterapia, logrando de esta manera determinar la valides y funcionamiento del sistema, las pruebas en si fueron realizados en el Centro de Fisioterapia de la Clínica MediFisio, contando con la participación de 10 pacientes la mayoría personas adultas del sector de diferente edad, estatura y patología.

Los resultados que se obtuvieron a través de las pruebas han demostrado que son valores de datos coherentes permitiendo en si al valides y confiabilidad del sistema en un 91%, el cual para su obtención se comparó tanto las mediciones manuales realizadas como las del sistema permitiendo llegar a ese resultado, de tal manera que el sistema ayuda al fisioterapeuta a llevar un registro detallado de cada paciente para su evaluación de esta manera permitiendo realizar un diagnóstico más detallado y tratamiento adecuado.

ABSTRAC

The present project is focused on the development of an embedded system capable of measuring the cifolumbar angle in the region of the spinal column for physiotherapy evaluation, which will be developed through computer vision that allows us to manipulate images or videos in real time. Through algorithms, we can recognize points of interest in the person forming a 2D scheme for processing.

In this way, the system with the previously developed algorithms can detect the markers placed by the physiotherapist in the back region, allowing them to evaluate the different curvatures and distances in the sagittal plane, thus obtaining a clear reading of the angles and distances in real time. At the same time, a medical record can be established digitally, which allows storing both patient and evaluation information in a local database.

Regarding the system's functional tests, they were carried out under the supervision of an expert in the field of physiotherapy, thereby determining the validity and functionality of the system. The tests themselves were carried out at the MediFisio Clinic Physiotherapy Center, with the participation of 10 patients, mostly adult individuals from different ages, heights, and pathologies.

The results obtained through the tests have demonstrated that the data values are coherent, allowing for a system validity and reliability of 91%. This was obtained by comparing both the manual measurements and those of the system, reaching this result. In this way, the system helps the physiotherapist to keep a detailed record of each patient for their evaluation, allowing for a more detailed diagnosis and appropriate treatment.

Capítulo 1. Antecedentes

En el presente capitulo se dará a conocer el tema del presente proyecto junto a la descripción de la problemática de este, también se planteará los objetivos tanto general como específicos, se dará a conocer el alcance del proyecto delimitándolo para la resolución de este, finalmente se justificará con el fin de respaldar el desarrollo para solucionar el problema planteado.

1.1. Tema

SISTEMA EMBEBIDO PARA LA DETECCIÓN DEL ÁNGULO CIFÓTICO Y LUMBAR POR MEDIO DE VISIÓN ARTIFICIAL.

1.2. Problema

Las patologías en la columna vertebral a nivel cifótico y lumbar posee distinta etiología, pueden ser congénitas es decir se desarrolla durante los meses de gestación y se evidencian en el nacimiento, o de origen postural en la cual se adquieren comúnmente en la infancia o adolescencia por diferentes factores, ocasionando repercusiones en la salud física y psicológica de la persona, por lo cual pueden producir dolor siendo este de leve a intenso afectando la calidad de vida y limitando la realización de actividades cotidianas y laborales. Las curvaturas en la columna vertebral son de carácter hiper o hipo afectando comúnmente a la cifosis y lordosis, las cuales son producidas por alteraciones a nivel muscular, esquelético y articular (Leonidas, Navarro, Jorge Díaz, & Lizana, 2018).

Dichas patologías son causadas: por la mala posición al realizar una actividad constante, sobre esfuerzo físico o escases de actividad física entre otras causas, cada vez aumentando desde edades más tempranas. La lordosis es la curvatura fisiológica de la columna en la región cervical y lumbar, mientras que la cifosis consiste es la curvatura en la zona torácica y sacra, el aumento o disminución de las curvaturas produce alteraciones fisiológicas (Pastor & Santoja, 2021).

El proceso de evaluación y diagnóstico que ejecutan los fisioterapeutas hacia los pacientes que tienen problemas postulares son tardíos y en algunos casos complicados, debido a que se trata de medir los ángulos en la columna vertebral con herramientas o procesos de evaluación como el inclinómetro o cifómetro que son instrumentos de medición, también se los realiza por métodos manuales conocidos como la flecha sagital o el método de cobb, este último se emplea en radiografías para la medición de los ángulos (Singla & Veqar, 2014). Estos procesos al ser realizados manualmente proporcionan cierto margen de error de medición por estimación visual propio del ser humano o instrumentos, por otra parte la evaluación por medio de radiografías es un problema debido a su costo elevado en algunos casos y también a la exposición a la radiación para la obtención de estas, por lo que podrían afectar a la salud del paciente, debido a esto la inclusión de la tecnología toma importancia en los procesos de evaluación, de tal manera que el profesional encargado tenga una alternativa mucho más efectiva, rápida e integrando la comodidad y bienestar del paciente generando un entorno amistoso. La obtención de estos ángulos, depende de las herramientas o métodos manuales usados por el profesional encargado para evaluar y diagnosticar al paciente, seleccionando el método o instrumento a su disposición en el cual procederá a realizar una evaluación con métodos manuales para posteriormente rectificar con los ángulos obtenidos con las radiografías y así brindar un diagnóstico al paciente, la información del paciente así como el diagnóstico es llevado en fichas médicas e informes hechos manualmente por el fisioterapeuta.

Uno de los procesos de evaluación más utilizado que realizan los fisioterapeutas es el Test de la flecha sagital el cual es un método de mediciones lineales aceptado por su disponibilidad y validez, el cual tiene como función obtener las distancias de ciertos puntos específicos de la zona cervical, dorsal, lumbar y sacra, las cuales pasan por fórmulas matemáticas con el fin de determinar el ángulo cifótico y lumbar para el diagnóstico de la hiper o hipo lordosis e hiper o hipo cifosis (Santoja & Pastor, 2006).

La solución planteada a esta problemática se fundamenta en realizar un sistema embebido basado en visión artificial que permite la obtención de los ángulos en la columna vertebral para brindar un diagnóstico al paciente, al mismo tiempo que se puede visualizar los datos obtenidos por medio de una interfaz gráfica previamente elaborada, también en una base de datos se guardara el diagnóstico e información del paciente respectivamente, esto permitirá al profesional encargado optimizar recursos en herramientas, métodos de evaluación y tiempo, en la obtención de datos y registro del pacientes, la integración de la tecnología permitirá la optimización del método empelado para la evaluación, debido a que nos brinda una alternativa basada en la captura de imágenes por medio de una cámara, las cuales serán almacenadas en la placa para realizar el respectivo procesamiento el cual permitirá obtener ciertos parámetro de interés en la columna vertebral, los cuales se basaran en el principio de la Flecha Sagital para la obtención de las distancias, por consiguiente con estos datos desarrollar un proceso matemático para la obtención del ángulo cifótico y lumbar, posteriormente compararlos con referencias establecidas para la obtención del diagnóstico Este proceso se realizara en un corto periodo de tiempo, aplicando los principios de visión artificial para automatizar procesos de evaluación y diagnóstico de ciertos problemas postulares presentes en la columna vertebral.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar un sistema embebido basado en técnicas de visión artificial para la identificación del ángulo cifótico y lumbar en la columna vertebral para la evaluación y el diagnóstico de la Hiper o Hipo cifosis y lumbar en pacientes del Centro de Fisioterapia de la Clínica MediFisio.

1.3.2. Objetivos específicos

- Analizar los métodos que involucren el proceso de evaluación postural que utilizan los profesionales de fisioterapia.

- Diseñar el sistema embebido con los parámetros establecidos de hardware y desarrollar el sistema basado en visión artificial que permitirá realizar el proceso de evaluación postular.
- Implementar el sistema que permita medir el ángulo cifótico y lumbar usando el método de procesamiento de imágenes de visión artificial, previamente para ser visualizados por medio de una interfaz.
- Realizar las pruebas con pacientes del centro de fisioterapia para determinar el funcionamiento del sistema y efectuar las correcciones de errores para garantizar la implementación final.

1.4. Alcance

El proyecto se basa en el desarrollo de un sistema embebido que emplea el uso de visión artificial, el cual permitirá la obtención de imágenes por medio de una cámara las cuales serán guardadas en la placa de visión artificia, posteriormente se desarrollara un algoritmo de procesamiento el cual permite la detección del ángulo lumbar y cifotico de las imágenes obtenidas, permitiendo agilizar el método de evaluación y diagnóstico empleados en la hiper o hipo cifosis y lumbar, en personas con problemas postulares del centro de Fisioterapia de la Clínica MediFisio, convirtiéndose en una herramienta de apoyo para el profesional.

En la fase inicial se realiza un análisis literario individual, el cual permita comprender, sobre los procedimientos y métodos usados, los cuales dan a conocer la forma de evaluación y diagnóstico de ciertas alteraciones en la columna vertebral como son el aumento o disminución de la curvatura vertebral, también se realiza un análisis con los fisioterapeutas en la clínica MediFisio, mediante entrevistas y encuestas dirigidas al departamento de Fisioterapia, con el objetivo de comprender la metodología actual utilizada en la detección de los ángulos involucrados en estas alteraciones, con el fin de obtener la información necesaria para un punto de partida en la elaboración del sistema de medición basado en visión artificial.

Para la selección del hardware y software tanto del sistema embebido como de la interfaz se escogerá los elementos que se apeguen a las necesidades de la implementación y ubicación estratégica del sistema para su funcionamiento en base a la norma IEEE 29748, que sigue un proceso de selección adecuado para los requerimientos previos al desarrollo del sistema, con la intervención de los stakeholders y a través de un modelo en Cascada el cual permitirá el desarrollo del proyecto de forma secuencial comenzando con las fases de análisis, pasando al diseño y terminando con las de testeo e implementación del sistema para su funcionamiento.

Los requerimientos del sistema se tomarán a partir de la fundamentación teórica y de las recomendaciones de los especialistas del centro de fisioterapia, esto permitirá adquirir la placa y los módulos de cámara y display para su posterior implementación.

El sistema embebido, tiene como propósito a nivel de hardware ser desarrollo con la placa “NVIDIA Jetson Nano” la cual cumple la función de una microcomputadora, de gran alcance que permite correr modelos de visión artificial, en la cual se conectara los módulos para el funcionamiento que permitan realizar capturas por medio de un módulo de cámara, de modo que se obtendrán imágenes las cuales se guardaran en la memoria del dispositivo para ser procesadas posteriormente, al mismo tiempo a nivel de software se desarrollara un algoritmo basado en el método de la flecha sagital, el cual se apegue al funcionamiento de nuestro sistema que nos permitirá analizar las imágenes y procesarlas con el fin de extraer los cuatro puntos de interés de la columna vertebral para el cálculo de las distancias, las cuales pasaran por un proceso matemático previamente programado que permitirán la obtención de los “índices lordicos y cifóticos“. El ángulo dado se comparará con parámetros establecidos por el fisioterapeuta para la ejecución de un diagnóstico sobre la hiper o hipo cifosis lumbar, este diagnóstico se guardará en una base de datos con el registro del paciente que tendrá como información: los nombres, apellidos, número de cedula, edad y una breve descripción de su

patología, la cual permitirá llevar un registro continuo de las sesiones de fisioterapia. El sistema final permite visualizar la información del paciente y su respectivo diagnóstico a través de una interfaz.

Se realizarán pruebas de funcionamiento en dos escenarios, en una primera fase de integración con un cierto número de personas con alteraciones en la espalda para tener una referencia de la medición que realiza el sistema y la correcta funcionalidad en la toma y análisis de datos verificando errores tanto de hardware y software. Una segunda fase se realizará con pruebas de funcionamiento para efectuar la corrección de errores del sistema con el fin de garantizar su implementación y la evaluación de datos obtenidos correctamente con los pacientes.

1.5. Justificación

Cuando surgió la pandemia, obligo a millones de personas a encerrarse en sus casas, continuando con las practicas cotidianas pero esta vez realizándolas desde su hogar, las cuales se fueron realizando diariamente, a causa de esto se han venido desarrollado ciertos problemas siendo el más principal los dolores de espalda debido a la poca actividad física y malas posturas al sentarse, siendo afectados desde los niños hasta los adolescentes por motivos de la educación virtual y de teletrabajo en personas adultas, estudios realizados prepandemia determinaron que 3 de cada 10 personan tiene problemas en la zona lumbar y 1 de cada 10 personas en la región cervical (Gallo, 2021), como resultado después de la pandemia los números de personas con problemas posturales aumentaron, los cuales impiden retomar las actividades diarias o en el trabajo de forma autónoma, siendo estos el motivo más frecuente de consultas en los centros de fisioterapia.

Este sistema de apoyo tiene como objetivo principal la automatización del método manual llamado Flecha sagital que es usado por los fisioterapeutas el cual permite la evaluación y diagnóstico de ciertas patologías como la Hiper o Hipo cifosis y lumbar debido a que estas

son desviaciones en los ángulos lumbar y cifotico, que se encuentran en la columna vertebral. Según la Organización mundial de la salud (OMS, 2020), de mil millones de personas, o sea en torno al 15% de la población mundial, tienen algún tipo de discapacidad por lo cual no pueden acceder a servicios sanitarios de calidad para ser evaluados, o tienen acceso a centros de salud deficientes y sin recursos para ser evaluados y diagnosticados correctamente, por lo tanto la Organización Mundial de la Salud ha permitido al desarrollo de proyectos innovadores que mejoren las calidad de vida de las personas que sufren con discapacidad, dando a conocer el artículo 1 el cual se refiere "El propósito de la presente Convención es promover, proteger y asegurar el goce pleno y en condiciones de igualdad de todos los derechos humanos y libertades fundamentales por todas las personas con discapacidad, y promover el respeto de su dignidad inherente." (Unidas, 2006)

En el Ecuador, se ha creado por medio del Consejo Nacional de planificación "El Plan del Buen Vivir", el cual aborda temáticas para la igualdad social en todos los sectores sociales permitiendo el acceso a servicios sanitarios para obtener una atención de calidad obteniendo acceso la a procedimientos de evaluación, diagnóstico y rehabilitación, a aquellas personas expuestas a vulnerabilidades como son las personas con discapacidad por lo que son grupos vulnerables de suma importancia que deben ser tomados en cuenta al momento de diseñar e implementar un proyecto, como se da a entender en el objetivo 1 el cual permite "Garantizar una vida digna con iguales oportunidades para todas las personas" (Consejo Nacional de Planificación, 2017).

Para el cumplimiento de una mejor evaluación y diagnóstico de un paciente con ciertas patologías, este debe acudir a un centro especializado que cuente con los equipos necesarios para la respectiva evaluación, a causa de esto muchas personas no puede acceder a ciertos procesos para su diagnóstico como son las radiografías por temas económicos o de salud. En las sesiones de fisioterapia el terapeuta es el encargado de realizar la evaluación con procesos

manuales que son necesarios para lograr una correcta evaluación y diagnostico postular. En la actualidad la tecnología permite brindar alternativas para el desarrollo de nuevas formas y métodos de evaluación en los pacientes permitiendo automatizar métodos tradicionales mejorándolos, facilitando la obtención de datos de una manera más rápida y precisa, acelerando los procesos de evaluación y obteniendo un entorno agradable para el paciente generando mayor motivación al interactuar visualmente a través de la tecnología, beneficiando directamente a los fisioterapeutas permitiendo disminuir del tiempo de evaluación y registro de los paciente, por otro lado, se tiene como beneficiarios indirectos a los pacientes los cuales obtienen un entorno agradable y ágil al momento de ser evaluados.

Capítulo 2. Fundamentación Teórica

El presente capítulo muestra los conceptos fundamentales para la elaboración del presente proyecto, como punto de inicio estableciendo conceptos fundamentales sobre la fisioterapia en la región de la columna vertebral con el fin de comprender la estructura y los ángulos que la conforman permitiéndonos entrar en los métodos de detección manual que permiten obtener el ángulo cifolumbar, el cual nos ayuda a diagnosticar las siguientes patologías: Hipo o Hiper lordosis/cifosis, así también se dará una breve introducción a los sistemas embebidos, haciendo énfasis en las placas embebidas dedicadas a la visión artificial, sistema de captura de imágenes y la visión embebida aplicada a estos sistemas, por último la explicación de visión artificial, sus aplicaciones, etapas de procesamiento de imagen y aspectos fundamentales como son los tipos de aprendizaje automáticos con el fin de explicar el uso de estas para brindar un diagnóstico, para terminar estos puntos serán explicados brevemente permitiendo desarrollar el sistema embebido para el presente proyecto.

2.1. Fisioterapia

La fisioterapia es una ciencia que ayuda a diagnosticar una discapacidad o movimientos funcionales que se encuentra afectados por una patología o problemas físicos, por consiguiente, también se encarga de la respectiva rehabilitación. en definitiva, se lo realiza por medio de métodos y técnicas que son utilizados por profesionales de este campo para la respectiva evaluación o rehabilitación. (Gallego, 2007)

El proceso de evaluación analiza los problemas causantes de la discapacidad, deficiencia o limitaciones funcionales de la persona, estos problemas pueden derivarse de ciertas causas como las lesiones, intervenciones quirúrgicas, enfermedades o malas posturas, la evaluación comprende la determinación y recomendación por el cual el fisioterapeuta realiza juicios clínicos con los datos obtenidos duramente la observación, palpación. ampliación de métodos y técnicas de valoración para ser sustentadas por exámenes complementarios para dar

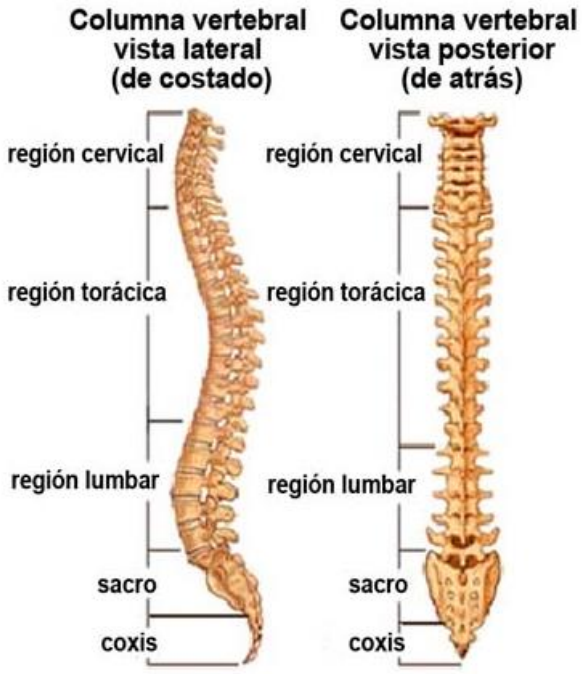
un diagnóstico y el tratamiento a seguir. Este proceso también puede identificar problemas que requieren ser referidos a otros profesionales de la salud.

2.1.1 Columna Vertebral

La columna vertebral o raquis se extiende desde la zona del cráneo hasta la pelvis unida por ligamentos y músculos, también en cada dos vertebras se encuentra conformada por discos cartilaginosos que amortiguan el movimiento de los huesos, esta se encuentra constituida de 33 a 34 vertebras y un hueso triangular "sacro" repartidas en 4 regiones que son: cervical, torácica, lumbar y pelviana como se muestra en la Figura 1, esta estructura permite actuar como protección y envolvente de la medula espinal y centro de gravedad y soporte del esqueleto (González, 2018)

Figura 1

Columna vertebral



Fuente: https://www.spineuniverse.com/espanol/anatomia/columna-vertebrall

En la columna vertebral debido a varios factores se desarrollan lesiones las cuales tienden a causar problemas como dolor o incomodidad, como resultado un gran número de

estos problemas, se debe a diversos factores como: posturas incorrectas , malos hábitos de vida , enfermedades entre otros, por consecuencia de estos factores la columna vertebral es afectada en sus curvaturas las cuales se desplazan de su posición original perjudicando los ángulos y ocasionando ciertas patologías como la Hiper /Hipo Cifosis o Lordosis.

2.1.1 Curvatura Vertebral

El ser humano a media que va envejeciendo empieza a desarrollar curvaturas naturales en toda la longitud de su espalda las cuales son conformadas en cuatro regiones que son: cervicales, torácicas, lumbar y sacra. Estas curvaturas ayudan a la columna vertebral con funciones como la flexibilidad y la fuerza. pero no siempre la columna vertebral va a presentar curvaturas normales debido a factores que se desarrollan con el tiempo o el nacimiento.

2.1.2 Anomalías en la columna vertebral

La columna vertebral cuenta con curvaturas fisiológicas que tiene como referencia un margen de normalidad, pero a veces por diversos factores estos pueden variar llamándose deformidades de raquis, las cuales pueden ser:

Anomalías Congénitas

Según la Organización mundial de la salud (OMS, 2020), las anomalías congénitas son denominadas como “Defectos de Nacimiento, Trastornos o Malformaciones congénitas”, estas malformaciones son presentes desde la gestación y son detectadas durante el embarazo, debido a que son heredadas de los padres.

Anomalías no Congénitas

La malformación es un defecto morfológico que resulta de un proceso de desarrollo anormal, estas son detectadas como deformaciones que conducen a el cambio de tamaño o desviaciones como en la columna vertebral. Un factor importante son los malos hábitos y estado físico de una persona los cuales influyen para el desarrollo de malformaciones, las personas han adquirido hábitos como el sedentarismo, la obesidad, la mala postura al sentarse

entre otros, también las deformaciones pueden ocurrir por ejercicios mal ejecutados o por embarazos, donde se ejerce una presión no acostumbrada a la zona de la columna la cual tiende a ceder los discos modificando la curvatura vertebral.

2.1.2.1 Enfermedades de la columna vertebral

Hernia discal

Las hernias discales son causadas por la sustancia tipo gel "núcleo pulposo" que sale a través de una ruptura en el anillo fibroso de nuestros discos, por lo cual tiende a irritar la pared externa de la columna vertebral y los nervios espinales afectando a las vértebras cervicales y lumbares produciendo un dolor insoportable, esta deformidad es una de las más comunes en la columna vertebral. (Romero, 2019)

Ausencia de vertebras

Esta enfermedad congénita en la columna vertebral se presenta desde el nacimiento, debido a que la persona puede nacer sin una o dos vertebras, esta enfermedad se puede presentar más en las regiones del coxis o sacro las cuales ocasionan una deformidad en la estructura ósea de la persona ocasionando riesgo en el futuro.

Espina Bífida

Es una malformación congénita del sistema nervioso más común en las personas. Se trata de una malformación desarrollada durante la gestación en el cual no se cierra correctamente las vértebras, por ende, la medula espinal no cuenta con una capa ósea que la proteja. Esta malformación es más común en la región lumbar o lumbosacra. (Alejandra, 2014)

Escoliosis

Es la desviación lateral de la columna vertebral más frecuente en los adolescentes. La columna se desvía hacia uno de los lados de la espalda, ya sea a la derecha o izquierda deformándola y presentando ciertas curvaturas dependiendo del grado de desviación en esta. Unos de los factores para el desarrollo de esta anomalía es la edad, sexo o estilo de vida, pero

cade destacar una las causas más frecuentes de la escoliosis es el factor hereditario. debido a esta anomalía se puede presentar problemas respiratorios, problemas en la espalda y aspecto de la persona.

Cifolordosis

Se representa como el incremento o disminución, abolición e inversión de la curvatura fisiológica, Lumbar: Cifosis e Cifolordosis, según (Moe & Winter, 2003) las causas de estas deformaciones en la columna vertebral se pueden clasificar según la Tabla 1.

Tabla 1

Clasificación de las etiologías de las desalineaciones Sagitales

CIFOSIS	LORDOSIS
Postular	Postural
Enfermedad de Scheüermann	Congénita
Congénita	Neuromuscular
Neuromuscular	Después de la minectomía
Mielomeningocele	Secundaria a la contractura en flexión de cadera
Traumática	
Posquirúrgica	
Después de irradiación	
Metabólicas	
Displasias esqueléticas.	
Enfermedades del colágeno.	
Tumores	

Fuente: (Moe & Winter, 2003) “Deformaciones de la columna vertebral”

2.1.3 Alteraciones en el plano sagital

Según (Chambi, 2010). La clasificación de las deformidades en la columna vertebral, se presentan por curvas de deformación adquiridas o alineamientos de tres tipos, cifosis, lordosis y rectificaciones, dependiendo del plano en el que se encuentren podemos localizarlas de la siguiente manera:

Desalineaciones en el plano sagital

- Incrementos/Disminución del grado de curvatura

  Hiper/Hipo cifosis

  Hiper/Hipo lordosis

  Cifolordosis

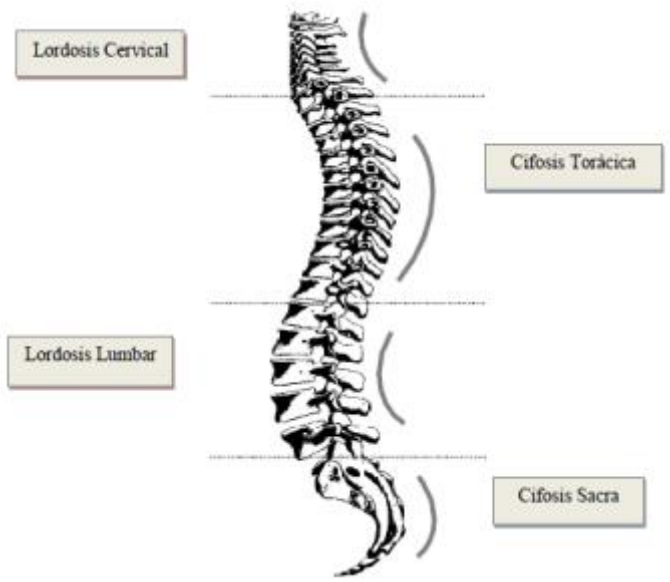
- Por no tener grado de curvatura

  Rectificación

La mayoría de estas alteraciones postulares en el plano sagital muestran curvaturas fisiológicas que varían en los márgenes de normalidad, poseen una naturaleza las cuales ocasionan una pérdida de la alineación normal en la columna vertebral ciertamente en la región dorso lumbar, en esta zona algunos tipos de alteraciones postulares son: Cifosis, Lordosis y Rectificación, como se muestra en la Figura 2, la cual podemos observar la columna vertebral la ubicación de sus diferentes anomalías.

Figura 2

Anomalías Columna vertebral



Fuente: Rouviere H “Anatomía humana” Tomo II Editorial Masson, Barcelona 1999

2.1.3.1 Cifosis

Para muchos expertos el concepto de Cifosis es el incremento de la curvatura raquídea en la convexidad posterior de la coluna vertebral en el plano sagital, se trata de una pequeña cantidad de curvatura anterior de la columna torácica es normal y se presenta debido a la forma de los cuerpos vertebrales y discos intervertebrales, un ángulo de cifosis mayor que 40º, se define como hipercifosis, aumenta el ángulo de cifosis, el rendimiento físico y calidad de vida a menudo disminuye, por lo que la intervención temprana para hipercifosis es una prioridad. (Katzman & Wanek, 2010)

2.1.3.2 Lordosis

Se define como lordosis a un incremento de la concavidad anterior de la curvatura lumbar, se puede manifestar con una anteversión de la pelvis, un abdomen prominente y nalgas salientes, donde "Es una angulación de convexidad anterior de la columna lumbar. El grado normal de lordosis ronda los 30º a 50 °" (Lyyn & Staheli, 2003)

2.1.3.3 Rectificación

La rectificación significa que su lordosis es menor de lo habitual o incluso ha desaparecido, de forma que la columna es recta vista de perfil. Una rectificación lumbar está causada por una retroversión permanente de la pelvis. La rectificación de una o varias de las curvaturas de la columna vertebral es relativamente frecuente**.**

2.1.3.4 Cifolordosis

Esta anomalía se caracteriza por tener las dos alteraciones en un mismo diagnóstico tanto la Hiper o Hipo lordosis/cifosis.

2.1.4 Métodos de diagnóstico Columna Vertebral

Algunos de los métodos de evaluación comúnmente utilizados en la región de la columna vertebral son instrumentos y test confiables que dan diagnósticos los cuales ayudan a

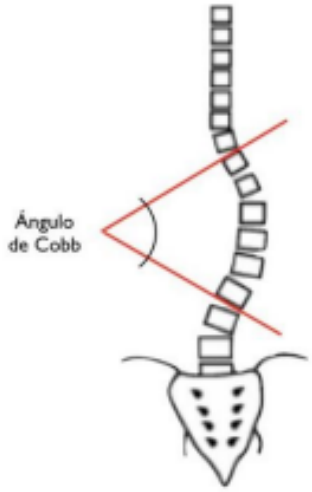
medir los ángulos de curvatura para algunos diagnósticos los cuales son la Hiper o Hipo Lordosis/Cifosis, estos métodos/ instrumentos pueden ser los siguientes.

2.1.4.1 Método de Cobb

El método de cobb se usa para la medición de las curvaturas vertebrales por medio de radiografías tomadas a la columna vertebral, en este método se trazan dos líneas las cuales nos ayudan a medir el ángulo como se demuestra en la Figura 3, la primeria se busca las vértebras terminales superior y la otra inferior, Primero se traza una perpendicular del borde superior de la vertebra hacia la concavidad, de igual manera se realiza el trazo de otra perpendicular al borde inferior ubicado en la vertebra inferior. (Aittor, 2000)

Figura 3

Método de Ángulo de Cobb



Fuente: https://www.columna-spine.com/espinograma/

2.1.4.2 Inclinómetro

El inclinómetro es un instrumento de medición de ángulos donde el goniómetro no se puede utilizar correctamente, estos instrumentos son ligeros y prácticos debido a que nos ayudan a la medición de la flexión-extensión en la columna vertebral, las mediciones de estos instrumentos son propensos a errores debido a que depende de la fuerza de gravedad como

punto de referencia para ser calibrado, estos errores técnicos arrojan lecturas inexactas en ciertas ocasiones, por ende es uno métodos con probabilidad de error.

Figura 4

Inclinómetro de fluido

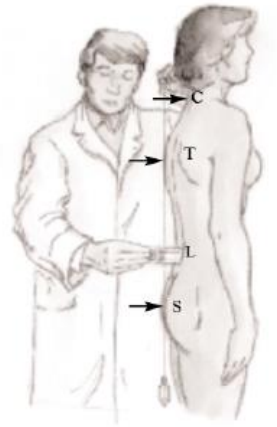
Fuente: https://inclinometro.com/fisioterapia/

2.1.4.3 Método Flecha sagital

Es un método de evaluación el cual nos ayuda determinar la cifosis y lordosis, por lo tanto, se consideran 4 medidas que se denominan flechas, para la obtención de estos parámetros el paciente al momento de realizar el test se debe encontrar en una postura de bipedestación con mirada al frente y piernas extendidas totalmente relajado.

Figura 5

Método Flecha Sagital



Fuente: *(Pastor & Santoja, 2021)* "Procedimientos Ortopédicos y de Traumatología"

Para ejecutar el test, se debe acercar un hilo de plomada hasta llegar al primer contacto de la columna, este se encuentra entre T7 y T9 o en el inicio del pliegue interglúteo como se muestra en la, Se miden la distancia desde el hilo con ciertos puntos los cuales son: (Pastor & Santoja, 2021)

- Flecha cervical (FC)= Distancia medida desde la apófisis espinosa de C7 hasta la plomada
- Flecha torácica (FT)= Distancia máxima de la concavidad del raquis dorsal hasta la plomada
- Flecha lumbar (FL)= Distancia máxima de la concavidad de la zona lumbar hasta la plomada
- Flecha sacra (FS)= Distancia desde el inicio del pliegue interglúteo hasta la plomada

Una vez obtenido estas 4 distancias se puede obtener los índices cifótico y lordótico del paciente aplicando ciertas ecuaciones para el cálculo de estas. (Santoja, 2022)

Índices para calcular:

- Índice cifótico (IC)
- Índice lordótico (IL)

El primer caso se puede presentar cuando el paciente tiene un eje adelantado en este caso: FT > 0 y FS = 0

$$IC = \frac{FC + FL + FS}{2} - FT \quad (1)$$

$$IL = FL - \left(\frac{1}{2} FT\right) \quad (2)$$

El segundo caso se puede presentar cuando el paciente tiene un eje adelantado en este caso: FT = 0 y FS > 0

$$IC = \frac{FC + FL + FT}{2} \quad (3)$$

$$IL = FL - \left(\frac{1}{2}FS\right) \quad (4)$$

2.2. Sistemas Embebidos

Según (Heath, 2003), define a los sistemas embebidos con el siguiente concepto. “Un sistema embebido es un sistema basado en un microprocesador construido para controlar una función o rango de funciones y no está diseñado para ser programado por el usuario final de la misma manera que una PC. Por consiguiente, un usuario puede elegir opciones relacionadas con la funcionalidad, pero no puede cambiar la funcionalidad del sistema agregando o reemplazando software. (p. 2) “

Como todo sistema este cumple funciones específicas ya programadas, optimizando el tamaño siendo este independiente o formando parte de un sistema más grande, que cumple con funciones específicas ejecutadas en tempo real permitiéndole almacenar información ya establecida en su programa.

2.2.1. Componentes

Un sistema embebido se distingue por sus componentes ya que posee hardware y un software embebido como unos de sus componentes principales, los cuales conjuntamente están dedicados al desarrollo de aplicaciones como sistemas independientes.

2.2.1.1. Hardware

Se refiere a todos los componentes físicos del sistema embebido que ayudan junto al software a realizar tareas previamente establecidas, estos componentes se distinguen por ciertas

características como el tamaño, funcionalidad, procesamiento, etc. Algunos elementos pueden ser:

Sensor: Son dispositivos que se encargan de captar acciones o estímulos externos los cuales brindan información, para luego transformarlos en señales eléctricas para pasar a las placas o microcontroladores.

Memoria: Consiste en un espacio de almacenamiento para la información previamente obtenida para su previo uso.

Microprocesadores: Son circuitos integrados que contienen los elementos de un procesador digital, los cuales brindan capacidad de cómputo al sistema, permitiéndole realizar tareas o funciones las cuales son programadas en el dispositivo.

2.2.1.2. Software

Es un sistema operativo que permite la supervisión y ejecución de tareas que pueden ser realizadas de manera individual o conjunta.

Según (David & Perez, 2009), define que "El software a ejecutarse dentro del sistema embebido tendrá restricciones importantes : 1º la cantidad de memoria para la ejecución, 2º capacidades de limitación para el procesamiento dependiendo de la velocidad del procesador, 3º el consumo de energía en el funcionamiento."

2.2.2. Tipos de Sistemas embebidos

Los sistemas embebidos de acuerdo con ciertas características como el rendimiento, procesamiento, bajo costo y consumo de potencia, pueden dividirse en cuatro tipos, a continuación, se detalla cada uno de estos.

2.2.2.1. Sistema Embebido Independiente

Estos sistemas se caracterizan por no depender de un anfitrión, esto quiere decir que no dependen de un procesador o un ordenador para realizar sus tareas establecidas (Blumenscheid,

2022), así pues, estos se encuentran en la tecnología embebida autónomas, los ejemplos más comunes son:

- Hornos microondas
- Lavadoras
- Consolas de videojuegos

2.2.2.2. Sistema Embebido en Red

Se trata de un sistema el cual permite conectar redes inalámbricas o alámbricas permitiendo la comunicación entre un nodo de dispositivos los cuales realizan tareas predeterminadas y dar salida a la información de estos dispositivos (Blumenscheid, 2022), estos dispositivos están conformados por microcontroladores o sensores, algunos ejemplos de estos son:

- Cajeros Automáticos
- Sistemas de seguridad

2.2.2.3. Sistema Embebido Tiempo Real

Estos sistemas tienen una característica única la cual son sistemas empotrados que se encargan de realizar tareas específicas dentro de un límite de tiempo establecido y un lugar predeterminado (Blumenscheid, 2022), los sistemas en tiempo real se dividen en dos los cuales son:

- Sistemas embebidos de tiempo real suave.- Estos sistemas priorizan el cumplimiento de su tarea sin importar el plazo de tiempo.
- Sistemas empotrados de tiempo real duro. - Estos sistemas dan prioridad al tiempo de ejecución sin la perdida de información.

2.2.2.4. Sistema Embebido Móviles

En este sistema sus características son portátiles, tamaño pequeño y fáciles de usar, en consecuencia, cuentan con un límite de memoria, algunos ejemplos de estos sistemas son:

- Cámaras Digitales.
- Teléfonos Móviles
- Reloj Inteligente.

2.2.3 Placa Embebidas

Son dispositivos tanto de hardware y software de ciertas características especiales las cuales conectados apropiadamente a los módulos electrónicos necesarios los cuales son utilizados en el diseño de los sistemas embebidos, teniendo en cuenta las necesidades de este para la fluides del sistema.

Una de las características más importantes de las placas embebidas es la potencia de procesamiento, debido a que es el núcleo central de dichos sistemas donde se encuentra el microprocesador, microcontrolador, DSP, o cualquier otro tipo de dispositivo encargado del procesamiento. permitiéndonos tener arquitecturas que combinadas con un software adecuado y hardware respectivamente se convierten en diseños integrados para un sistema embebido.

2.2.3.1 NVIDIA Jetson Nano

La placa de desarrollo NVIDIA Jetson Nano es un dispositivo IoT que cuenta con características favorables para su reducido tamaño cuenta con una capacidad de procesamiento potente debido a que nos ofrecer 472 GFLOPS de rendimiento de cómputo utilizando una potencia de solo 5 W, su finalidad es facilitar la ejecución de redes neuronales muy rápido. para diferentes aplicaciones en visión artificial o robótica., lo cual nos ayuda a resolver problemas relacionados con la conducción autónoma y el tráfico. de vigilancia, médico y agrícola, de navegación para drones.

Figura 6

Nvidia Jetson Nano

Fuente: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

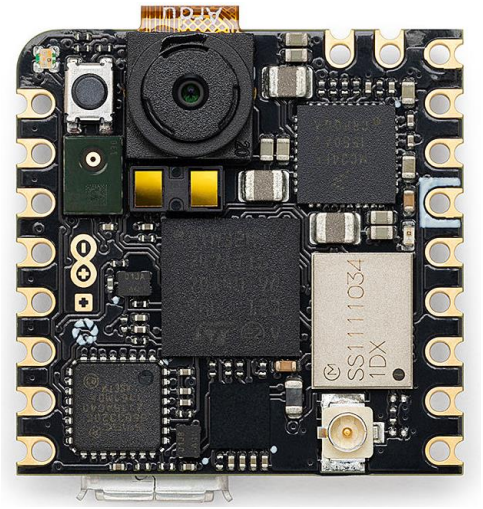
2.2.3.2 Nicla Vision

Por la parte de placas desarrolladoras de la marca de Arduino se cuenta con la siguiente placa llamada Nicla Vision, de manera que es una placa que cuenta con una cámara independiente a color de 2 MP compatible con TinyML, la cual analiza y procesa las imágenes en un cierto rango o perímetro, por consiguiente, Nicla Vision cuenta con un procesador ARM Cortex M7/M4 a la cual se integran ciertos sensores como: sensor de movimientos, micrófono y un sensor de distancia.

Cabe destacar que Nicla Vision gracias a su compatibilidad con: otras placas de Arduino, OpenMV, MicroPython y también ofrece conectividad WiFi y Bluetooth Low Energy, por lo cual es una opción factible para el desarrollo de prototipos referentes al procesamiento de imágenes, reconocimiento facial o de objetos entre otros. (Arduino, 2023)

Figura 7

Nicla Vision



Fuente: https://store-usa.arduino.cc/products/nicla-vision?selectedStore=us

2.2.3.3 Coral Dev Board

Coral Dev Board es una placa enfocada en realizar aprendizaje automático la cual ayuda a crear sistemas integrados junto a su modulo SOM, cabe destacar que el sistema integrado cuenta con las siguientes características: sistema en chip (SoC) iMX8M de NXP, memoria eMMC, RAM LPDDR4, Wi-Fi y Bluetooth. Igualmente cuenta con un coprocesador desarrollado por Google Edge TPU el cual es un pequeño ASIC que proporciona un alto rendimiento con bajo costo de energía. (ASUSIOT, 2023)

Figura 8

Coral Dev Board

Fuente: https://coral.ai/products/dev-board/

2.2.4 Sistema de Captura

La adquisición de la imagen se trata de la primera etapa del sistema para visión que consiste en capturar la imagen y digitalizarla. El primero es un dispositivo físico que es sensible a la energía irradiada por el objeto (espectro de energía electromagnética) del que queremos capturar la imagen. Y el segundo elemento, llamado digitalizador, es un dispositivo para convertir la salida del dispositivo de detección física en forma digital (Rafael & Woods, 2002)

Esta fase se la realiza para la obtención de una imagen en RGB la cual se captura por medio de un módulo de cámara que está conectado a la placa embedded, el cual obtiene como dato una imagen en formato digital que nos sirve para empezar a realizar el procesamiento, una de las partes más importantes del sistema de captura es la cámara.

Figura 9

Cámara web

Fuentehttps://www.bhphotovideo.com/images/images1000x1000/adesso_cybertrackh4_cybertrack_h4_1080p_desktop_1567510.jpg

2.2.5 Visión Embebida

En la actualidad la visión integrada ha tenido un progreso muy grande en tanto a los sistemas embebidos que emplean métodos de visión, de manera que estos con el tiempo han sido una solución viable por su bajo costo, fácil de instalación y manera de usar, permitiendo abarcar muchos campos en los sistemas embebidos de bajo nivel desde el procesamiento de

imágenes hasta el manejo de secuencias de video de alto nivel. al mismo tiempo se han desarrollado aplicaciones que satisface los requerimientos necesarios, por lo que en la actualidad hay diferentes aplicaciones de visión embebida para diferentes dispositivos. (Xu, 2010)

Los Sistemas Embebidos que utilizan visión artificial, son mejores al momento de realizar evaluaciones en magnitudes físicas, debido a que este no puede presentar dificultades como las ilusiones ópticas, estos sistemas puede mantenerse activos sin ningún inconveniente, sin afectar su desempeño, todo lo contrario en el caso de un ser humano en el que hay desgaste físico o agotamiento el cual puede afectar el desempeño y aumentando al probabilidad de errores, en resumen los sistemas embebidos de visión artificial no necesita de contacto físico para efectuar su funcionamiento permitiendo a las personas la simplificación de métodos de evaluación en el campo de la medicina.

2.3. Inteligencia Artificial

La inteligencia artificial se define como un sistema o máquina que emula la inteligencia humana al momento de realizar tareas de una manera de igual o mejor manera que una persona. Por consiguiente, el comportamiento inteligente se expresa como razonar, aprender, solucionar y actuar en entornos controlados.

Autores como Deyi Li, mencionan que la inteligencia artificial es: *"Un área de estudio en el campo de la informática en el cual la inteligencia artificial está preocupada por el desarrollo de computadoras capaces de involucrarse en procesos propios de los seres humanos de pensamiento como el aprendizaje, el razonamiento y la auto corrección"* (Li & Du, 2016). Es así como la inteligencia artificial desprende una variedad de ramas tales como:

- Vision Artificial
- Machine Learning
- Minería de datos

- Neural Networks

2.3.1 Machine Learning

Machine Learning se establece como un conjunto de técnicas que forman parte de la inteligencia artificial, es decir permiten a las maquinas aprender por medio de ejemplos o experiencias las cuales se adquieren a partir de un conjunto de datos. Por lo cual una de las características principales es la predicción de nuevos casos debido a la cantidad de datos que deben utilizar.

Los algoritmos de Machine learning por los datos que estos necesitan tienen la capacidad de predecir nuevos casos en base a la experiencia aprendida, por consiguiente, a esto se le llama aprendizaje automático los cuales hacen que un sistema aprenda a partir de un conjunto de datos en lugar de tener una programación lineal. (Hurwitz & Kirsch, 2018)

El aprendizaje automático según (Müller & Guido, 2016) permite extraer información de los datos, como resultado el aprendizaje automático se ha vuelto omnipresente en todas las actividades de la vida diaria, permitiendo obtener sistemas que basada en nuestra información hagan recomendaciones como: sistemas de mercado, seguridad automáticos, salud entre otros, por esto la mayoría de los dispositivos cuentan con algoritmos de aprendizaje automático en su funcionamiento.

El aprendizaje automático se centra más en el desarrollo de sistema que se ligan a la informática permitiendo cambiar o reaprender de acuerdo con los datos y condiciones que se presenten, puesto que ahí varios tipos de algoritmos basados en modelos matemáticos que permiten trabajar el aprendizaje de manera automática, estos son: aprendizaje supervisado, no supervisado, reforzado.

2.3.1.1 Aprendizaje Supervisado

Según (Müller & Guido, 2016) por cada entrada al sistema se establece la salida deseada, de tal manera que el algoritmo puede crear o predecir una salida para los datos que

ingresan sin la ayuda de una persona. El objetivo es tener una regla que intenta tener una relación y dependencia de los resultados predictivos objetivos y las características de entrada. El aprendizaje supervisado se puede dividir en:

Regresión

Este algoritmo es usado en la predicción de datos de salida basados en los datos de entrada, debido a esto el algoritmo basado en los datos de salida de los datos entrenados ayudando a predecir los valores para los nuevos datos. Los valores de salida en este caso son continuos y no discretos, este tipo de algoritmos tienen las siguientes características como se muestran en la Figura 10, Según la página de (AprendeIA, 2023) algunos algoritmos de regresión son:

- Regresión lineal
- Regresión polinomial
- Vectores de soporte regresión
- Arboles de decisión regresión
- Bosques aleatorios regresión

Figura 10

Aprendisaje Supervisado

Fuente:https://aprendeia.com/todo-sobre-aprendizaje-supervisado-en-machine-learning/

Clasificación

Por cada entrada del sistema al sistema son finitas y discretas, las cuales se enfocan en la predicción de una respuesta cualitativa, en resumen, se usan cuando los datos se encuentran etiquetado o separado en grupos específicos, este tipo de algoritmos tienen las siguientes características como se muestran en la Figura 11. Según la página de (AprendeIA, 2023), los tipos de algoritmos de clasificación incluyen:

- K Vecinos más cercanos
- Máquinas de vectores de soportes
- Arboles de decisión clasificación
- Redes Neuronales
- Regresión logística

Figura 11

Aprendisaje Supervisado

Fuente: https://aprendeia.com/todo-sobre-aprendizaje-supervisado-en-machine-learning/

2.3.1.2 Aprendizaje No Supervisado

Este tipo de aprendizaje permite que de un grupo de datos la maquina pueda explorar e identificar patrones que vinculan diferentes variables de salida, dado por hecho esto permite

clasificar a los datos en grupos basados en propiedades estadísticas permitiendo la conexión entre estos, de modo que los datos estén segmentados en (clústeres). Cabe destacar que los datos sin etiquetar crean los clústeres, por lo que el algoritmo de aprendizaje automático es más complejo y el procesamiento requiere de mucho tiempo. (Hurwitz & Kirsch, 2018)

Uno de los principales problemas de este aprendizaje es la agrupación de datos, debido a que a que no se encuentra una estructura a la recolección de datos sin etiquetas, por consiguiente, se debe usar un algoritmo que permita crear grupos donde todos los datos tengan características similares el uno del otro. Según (AprendeIA, 2023) algunos algoritmos de agrupación pueden ser:

Agrupamiento exclusivo: Los datos se agrupan exclusivamente de forma definida con las mismas características, permitiendo que otros datos diferentes no puedan ingresar a este clúster.

Superposición de clúster: Son grupos de datos que cuentan con niveles de similitud permitiendo a estos formar parte de uno o más grupos con el nivel de similitud apropiado en cada grupo.

Agrupamiento jerárquico: Es la unión de dos clústeres semejantes con la condición de establecer cada punto como un clúster individual para partir de este realizar las interacciones de agrupamiento para alcanzar los clústeres finales.

Agrupación probabilística: Son clústeres donde se aplica el enfoque probabilístico.

Agrupación K-Means: Divide datos en grupos distintos según el centro de un punto específico.

Modelos Gaussianos: Son agrupaciones de distintas intensidades normalmente multivariables.

2.3.1.3 Aprendizaje Reforzado

Este tipo de aprendizaje la maquina puede interactuar con su entorno permitiendo que esta aprenda de una manera lineal permitiendo que la maquina realice las acciones a su debido tiempo, debido a esto el algoritmo está diseñado con premios o castigos a medida que se van solucionando lo problemas permitiendo un autoaprendizaje si la maquina comete errores en consecuencia de sus decisiones. Este tipo de aprendizaje es más utilizado en la robótica con la integración del aprendizaje profundo.

2.3.2 Vision Artificial

La visión por computadora o visión artificial es una herramienta la cual está encargada de emular los procesamiento o extracción de ciertas características como la forma, el volumen, los colores tamaños entre otras como la percepción humana, esto se lo realiza por medio de métodos u algoritmos de aprendizaje, los cuales ayudan a construir tecnologías en donde se combina cámaras de video con computadoras, con el finde interpretar las capacidades visuales del ser humano, realizando métodos de manera automática, rápida y controlada.

La visión artificial se refiere al análisis de imágenes o videos por medio de un proceso establecido, en otras palabras se establecen ciertos algoritmos de programación los cuales ayudan al procesamiento o aprendizaje de imágenes, en los cuales su función principal es la de obtener parámetros establecidos en la imagen, en resumen estos ayudan a la obtención de una base sólida la cual es usada para la detección de ciertas características entre las más usadas tenemos: distancia , reconocimiento, edición de imagen entre otras.

Una de las ventajas de los sistemas de visión artificial es el procesamiento y análisis de imágenes de manera cuantitativa, por consiguiente, esta característica de procesamiento depende en su mayor parte al tiempo y obtención de las imágenes a través del sistema de captura, de manera las imágenes deben cumplir con ciertas características como: tener una resolución correcta y optima de las imágenes permitiendo detallar objetos grandes o pequeños

en forma de pixeles los cuales la visión humana no puede examinar a simple vista permitiendo realizar análisis más exactos, debido a esto se reduce significativamente la participación humana en ciertos procesos reduciendo tiempo, costos en los sistemas y el error humano. (COGNES Corporation, 2016)

2.3.3 Aplicaciones

En la actualidad la visión artificial ha sido protagonista de grandes avances tecnológicos, debido a que ejecuta acciones de manera más rápida, eficaz y autónoma en comparación a una persona. Uno de los campos a destacar de esta ciencia es el de la visión artificial, el cual cuenta su principal propósito es el de adquirir, procesar, analizar y comprender las imágenes. Esto ha permitido el desarrollo de aplicaciones, por ejemplo, en las industrias permitiendo: (contar productos, comprobación de defectos, elaboración de productos entre otros), en la medicina permitiendo realizar acciones como (diagnósticos, automatización de procesos, búsqueda de células etc.), en la Tabla 2, se puede visualizar otras áreas de aplicación.

Tabla 2

Campos aplicativos de la Visión Artificial

Área de Producción	Aplicaciones
Biomédicas	Análisis de imágenes de microscopia (virus, células, proteínas) Resonancias magnéticas, tomografías, genoma humano
Robótica	Control de Soldaduras Guiado de Robots
Control de calidad	Inspección de productos Identificación de piezas Etiquetado Inspección de circuitos Control de calidad
Navegación	Guiado de vehículos terrestres, aéreos y marítimos Vehículos autónomos
Realidad Aumentada	Aplicaciones móviles
Vigilancia y Seguridad	Reconocimiento de rostros Conteo de personas Rastreo de personas

Astronomía	Exploración del espacio
Control de trafico	Matriculas de vehículos Trafico
Agricultura	Análisis de cultivos Control de plantaciones
Militares	Seguimiento de objetos Vigilancia Satelital

Fuente: Adaptado de (Platero Dueñas, 2009)

2.3.3.1 Visión artificial aplicado a la medicina

Uno de los campos beneficiados gracias a la visión artificial es la medicina, así pues, permiten brindar tratamientos o diagnósticos o mejorando la mejorando la calidad vida de los pacientes de una manera rápida y eficaz, para los profesionales de la salud, la visión artificial ha permitido simplificar procesos manuales o automatizarlos, permitiendo el acceso a imágenes y analizándolas para brindar resultados que escapan de las habilidades humanas. (Vida.ip, 2021)

El campo de la visión artificial ha ayudado a mejorar los diagnósticos y rehabilitaciones físicas, de una manera más rápida y eficiente en algunos casos por la automatización de métodos para el diagnóstico, el diagnóstico dado por el especialista se basa en observaciones de imágenes que sugieren, en algunos casos, conclusiones subjetivas estos procesos se pueden realizar incorporando la visión artificial.

2.3.3.2 Visión artificial en la Fisioterapia

Una de las ramas del machi learning es la visión artificial, en la cual su principal propósito es enseñar a los ordenadores por medio de programas el poder ver y entender el contenido de una imagen, de tal manera simplificado algunos procesos de rehabilitación sin necesidad presencial del especialista en la fisioterapia y el paciente, por ende, se han desarrollado plataformas que permitan este trabajo como se muestra a continuación.

DyCare

DyCare es una empresa catalana dedicada a desarrollar soluciones tecnológicas especialmente en el campo de la rehabilitación física, por consiguiente, la empresa DyCare a desarrollado por medio de visión artificial una plataforma llamada ReHub la cual por medio de una cámara es capaz de analizar 74 puntos en el cuerpo humano proporcionando un feedback en tiempo real permitiendo realizar terapias de rehabilitación a distancia sin supervisión profesional. (Busquets, 2022)

La plataforma de ReHub cuenta con algoritmos de inteligencia artificial para ser una plataforma terapéutica para el tratamiento de lesiones musculoesqueléticas de tal manera que permita analizar y ofrecer una terapia según las condiciones del paciente, de tal manera que por medio de una cámara y un asistente virtual se guie al paciente en la ejecución correcta de los ejercicios, de modo que se realice el seguimiento del paciente de una manera más fácil identificando el progreso o permitiendo el cambio de la terapia.

Trak

Es una plataforma de rehabilitación digital que por medio de la visión artificial permite a los pacientes a la rehabilitación desde el hogar, en otros términos, es una plataforma de telerehabilitación la cual permite al fisioterapeuta pree escribir ejercicios de una manera más ágil. por consiguiente, el paciente los realizara de una manera correcta y controlada.

Trak es una aplicación que utiliza una cámara móvil o portátil que por medio de visiona artificial identifica 20 articulares las cuales permiten analizar las posiciones y el rango de movimiento del paciente, permitiendo corregir la mala ejecución de los ejercicios en tiempo real. Trak cuenta con 800 ejercicios permitiendo involucrar al paciente con su tratamiento. (Trak, 2023)

Fisify

Fisify es un software que tiene permite a los centros de fisioterapia ayudar con la calidad de tratamientos y sesiones de recuperación, por esto la fisioterapia virtual viene a la par con la

visión artificial que ayuda proporcionando una información y orientación a partir de un cuestionario que determina un diagnóstico de la sintomatología precisa. Fisify ayuda al paciente desde su diagnóstico hasta su recuperación dando un seguimiento en tiempo real gracias a su sistema que ayuda y corrige la postura por medio de una cámara.

Sus tratamientos van dirigidos a los dolores musculoesqueléticos: lesiones musculares, ligamentosas, tendinosas y recuperación de cirugías. Así Fisify basa los ejercicios de cada paciente para un tratamiento a largo tiempo, abordando de una manera más saludable y dinámica los resultados que se obtienen a lo largo de un tiempo. (Charaxes, 2023)

2.3.4 Procesamiento de Imágenes

El procesamiento de imágenes se refiere a al uso de diversas técnicas que se realizan a la imagen de entrada pudiendo procesarla o mejorarla, por medio del uso de datos cuantitativos o conjuntos de datos numéricos para cambiar el resultado visual. En resumen, la imagen final es diferente a la imagen de entrada. Los tipos de procesamiento de imágenes pueden ser digitales, donde el programa establece una matriz digital de pixeles para su aplicación, por último, para el procesamiento de imágenes analógicas, donde el software funciona desde una imagen física.

Según (Gonzalez & Woods, 2007), a la imagen digital se le aplica un conjunto de técnicas con el fin de que el resultado sea el más adecuado dependiendo de las aplicaciones a utilizar. De esta manera para un sistema no se le puede aplicar un determinado método de procesamiento debido a las necesidades de la imagen de salida.

El procesamiento de imagen se ha desarrollado debido a las necesidades que la imagen presente estas son: (Petrou, 2010)

- Digitalización y codificación de imágenes que faciliten el tratamiento de transmisión, impresión y almacenamiento.
- Restauración y mejoramiento de la imagen

- Descripción y segmentación de la imagen.

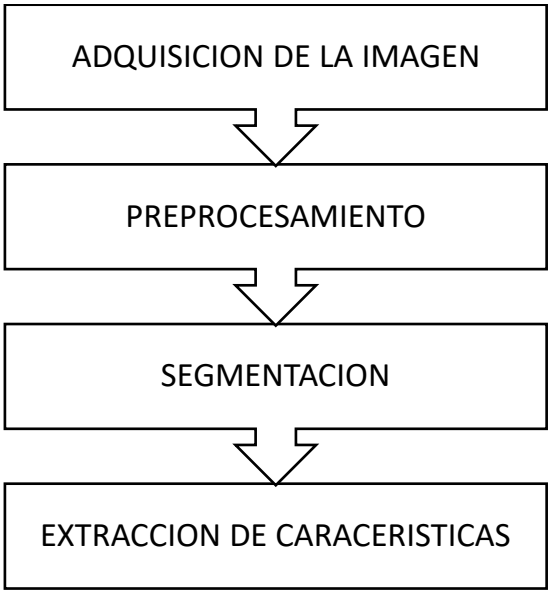
2.3.5 Etapas Vision Artificial

En la visión del ser humano la cual captura imágenes que se transforman en información las cuales circulan por los nervios ópticos hasta llegar al cerebro y ser interpretadas para la obtención de una respuesta de lo que se está observando, la visión artificial también cumple con etapas algo similares divididas en dos grupos.

En la primera fase se trata de etapas que ejecutan métodos de bajo nivel las cuales se encargarán de obtener las características básicas de la imagen, al mismo tiempo la segunda fase se encarga de un procesamiento de alto nivel, el cual consta de recoger las características extraídas del anterior nivel y construir una descripción del resultado, dado que estos sistemas de visión artificial usan diversa técnicas y cada aplicación tiene sus propias características específicas, existe una serie de etapas que son comunes en todo proceso como se muestra en la Figura 12, las cuales son :

Figura 12

Etapas de Visión artificial



Fuente: Autoría

Estas etapas son comunes en casi todos los sistemas de visión artificial, pero dependiendo del sistema empelado puede aumentar o disminuir, las etapas no siempre van en secuencia como en el diagrama, por el contrario, si en alguna etapa se falla esta regresara al proceso anterior o inicial para corregir los fallos.

2.3.5.1 Adquisición

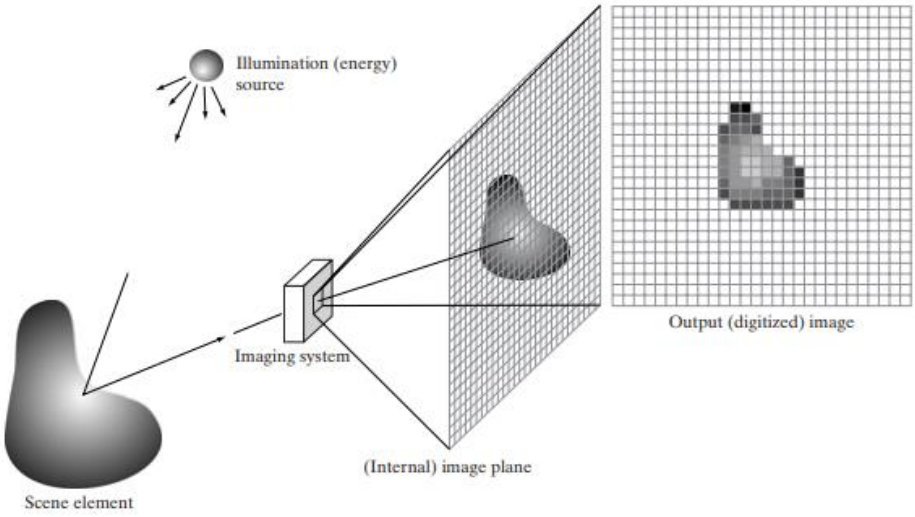
Es considerada la primera etapa de un sistema de visión artificial, la cual consiste en capturar la imagen por métodos de fotografía a través de un sensor a cámara las cuales nos permite transformar las señales luminosas de la escena, en señales analógicas capaces de ser transmitidas al sistema permitiéndonos obtener las características visuales del objeto a color o B/N.

Algunos de los aspectos importante en cuanto a la adquisición de la imagen, se trata de la iluminación por lo cual las cámaras no ven a los objetos en si sino a la luz reflejada de estos, debido a las características del contraste que simplifican la integración de esta cuando una iluminación es adecuada.

La digitalización de la imagen se lograr gracias dos elementos importantes en la etapa inicial del sistema de visión artificial, el primero consta de un dispositivo físico que capta cierta banda del espectro electromagnético del que pertenece la imagen, y el segundo elemento que es un digitalizador que convierte la detección física en digital. (Rafael & Woods, 2002)

Figura 13

Proceso de adquisición de imágenes digitales



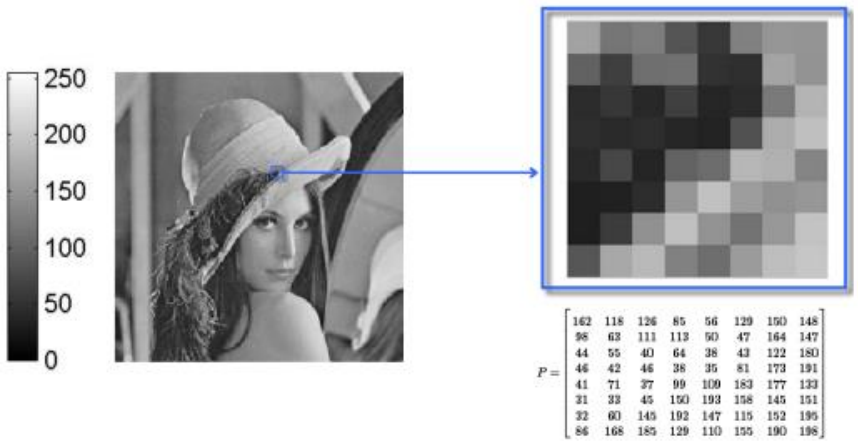
. Nota: (a) Fuente de energía ("iluminación"). (b) Un elemento de una escena. (c) Sistema de imágenes. (d) Proyección de la escena en el plano de la imagen. (e) Imagen digitalizada Fuente: *(Rafael & Woods, 2002)*

Imagen

Una imagen digital se define como la representación bidimensional que utiliza f(x, y), donde x e y están representadas en un plano de coordenadas y f se considera la intensidad o el nivel de gris en ese punto, en concreto si las coordenadas y la intensidad sean finitos se conocerá como imagen digital, es decir la imagen siempre estará compuesta por un número finito de pixeles, los cuales tienen valor y posición particulares. (Garcia, 2008)

Figura 14

Imagen con 256 niveles de intensidad



Fuente: *(Aguirre, 2013)*

Según (Garcia, 2008) las imágenes pueden dividirse en 2 tipos:

- Mapas de Bits: Son imágenes codificadas en una matriz de puntos donde se ordenan los bits que representan el color de la imagen, de manera que la imagen consta de un número fijo de pixeles, en consecuencia, esta depende de la resolución, debido a esto cuando es modificado su tamaño esta pierde resolución, cada píxel de la imagen se le asigna un valor y ubicación especifico.
- Vectoriales: Son imágenes formadas a partir de curvas y líneas especificados como vectores, estas imágenes no sufren perdida de resolución, por lo que conservan su nitidez sin importar la modificación de su tamaño, cada uno de sus pixeles son representado por una ecuación matemática que se relaciona con el resto de los pixeles en la imagen.

Dentro de los tipos de imágenes se encuentran varios formatos, algunos de los cuales son más prácticos y rápidos para el procesamiento del sistema, siendo el tipo de imágenes con las que vamos a trabajar, entre los formatos más conocidos tenemos:

- GIF: Fueron creadas por Compuserve, son imágenes compactas de animación en movimiento que cuentan con 256 colores, estas cuentan con grandes áreas de un mismo color, las cuales afectan a la calidad que se muestra en pantalla.
- JPG: Fueron creadas por el Grupo Unido de Expertos en Fotografía (Joint Photographic Experts Group), es el formato más utilizado y de mejor calidad para las imágenes al contener millón de colores, de diferentes dimensiones, pero poco eficiente al momento de comprimirlas por la pérdida de información.
- MBP: Formato de imagen que utiliza Windows de tamaño muy grandes para su procesamiento:
- PNG: Es el formato de archivo par la web PNG (Portable Network Graphics: gráfico de red portable), tiene similitud al formato JPG.

2.3.5.2 Pre-procesamiento

El pre-procesamiento de las imágenes se da por un grupo de métodos en el cual se trabaja la imagen obtenida con el fin de modificarla en ciertos aspectos o manejar la visibilidad de ciertas características naturales o artificiales de esta, para la obtención de información, se describirá las etapas de procesamiento que se realiza, la imagen adquirida la imagen esta debe tener un formato apropiado para su procesamiento en el cual se debe evitar el consumo de memoria del dispositivo.

El objetivo del pre-procesamiento es mejorar la calidad de la imagen original, donde podemos eliminar ciertas características de la imagen las cuales son: (ruido, contraste, degradados, etc.), por consiguiente, eliminando esos factores se puede obtener de una mejor manera los bordes, segmentos, características. Con una imagen ya tratada se puede mejorar el procesamiento para la obtención de mejores resultados, según (Mehl & Peinado, 2021)ahí dos tipos de operaciones que se realizan en la imagen las cuales son:

Operaciones Globales: La imagen al contener varios pixeles se le aplica la operación a cada uno de estos, obteniendo un nuevo valor que depende del obtenido anteriormente permitiendo modificar cada píxel de forma independiente, por lo tanto, es una de las operaciones globales más usadas es la de mejorar el contraste, en el cual la imagen original tiene un contraste poco nítido y se procede a mojarlo utilizando el método de estirar el rango de valores de gris al máximo posible.

Operaciones Locales: Esta operación depende de los pixeles vecinos para su modificación, esta operación se utiliza para aplicar un filtro de imagen como filtrado espacial o de convolución, los cuales adoptan valores que se asemejen o se diferencien de los pixeles de sus vecinos.

Imágenes a escala de grises

Se trata de la conversión de una imagen a escala de grises para poder trabajar con ellas con mayor facilidad, pues los tres canales se reducen a un solo canal que representa la intensidad de la imagen. La conversión de una imagen RGB a escala de grises se realiza a través del cálculo de un equivalente que depende de cada plano de color del modelo RGB.

Este proceso busca la optimización del umbral de la imagen eliminando las imperfecciones como: ruido, manchas, fondos entre otros, los cuales impiden el proceso de segmentación de las imágenes, además según (Garcia, 2008) una imagen en escala de grises tiene ciertas características las cuales son:

- Utilizan distintos tonos de grises
- Las imágenes de 8 bits tienen 256 tonos de grises
- Cada pixel tiene un valor comprendido entre 0 y 255, tomando el 0 como el color (negro) y el 255 como el color (blanco).

Filtrado

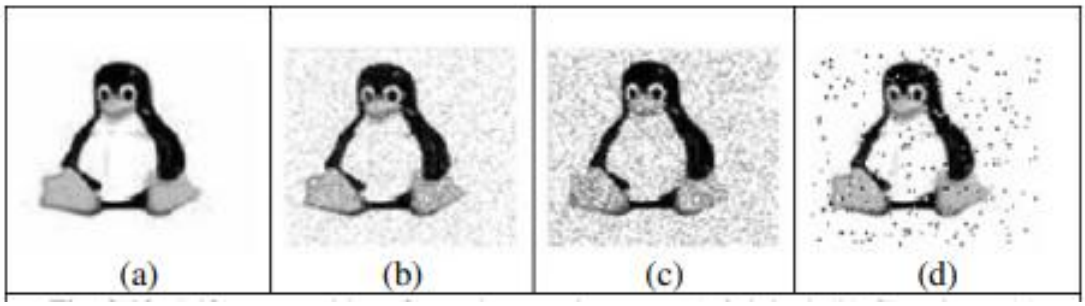
Las imágenes obtenidas siempre cuentan con cantidades de ruido mínimas, debido al sistema de adquisición, por lo cual la imagen entrante no tendrá un umbral nítido debido a los pixeles erróneos, según (Garcia, 2008) ahí diferentes ruidos que se presentan en una imagen en la etapa de adquisición entre ellos tenemos:

- **Ruido Gaussiano**: Son pequeñas variaciones de la imagen que afecta a los pixeles de esta, este ruido es producido por los componentes electrónicos del sistema de adquisición como los sensores, en consecuencia, cada pixel tiene un valor agregado correspondiente del error “variable aleatoria gaussiana”
- **Ruido Sal y Pimienta:** Toma valores altos o bajos y se presenta en los pixeles aleatoriamente, apreciando pixeles blancos y negros, se lo interpreta como valores máximos “Sal” y valores mínimos “Pimienta”.

- **Multiplicativo:** La señal que se obtiene es el resultado de las multiplicaciones de dos o más señales.

Figura 15

Diferentes Ruidos afectando la imagen



Nota: (a) Original, (b)Gaussiano. (c) Multiplicativo, (d) Sal y Pimienta. Fuente: *(Garcia, 2008)*

Uno de los métodos para el procesamiento de las imágenes digitales es la aplicación de filtros debido a que estos transformar pixel a pixel la nitidez de la imagen, el principal objetivo de la aplicación de filtros es la siguiente: (Pezonaga, 2015)

- **Suavizar la imagen:** Limitar la intensidad de los pixeles vecinos.
- **Eliminar ruido:** Modificar los píxeles cuya intensidad es muy distinta al de los vecinos.
- **Realzar bordes**: Destacar el borde de la figura en la imagen.
- **Detectar bordes**: Detectar cambios bruscos en los pixeles.

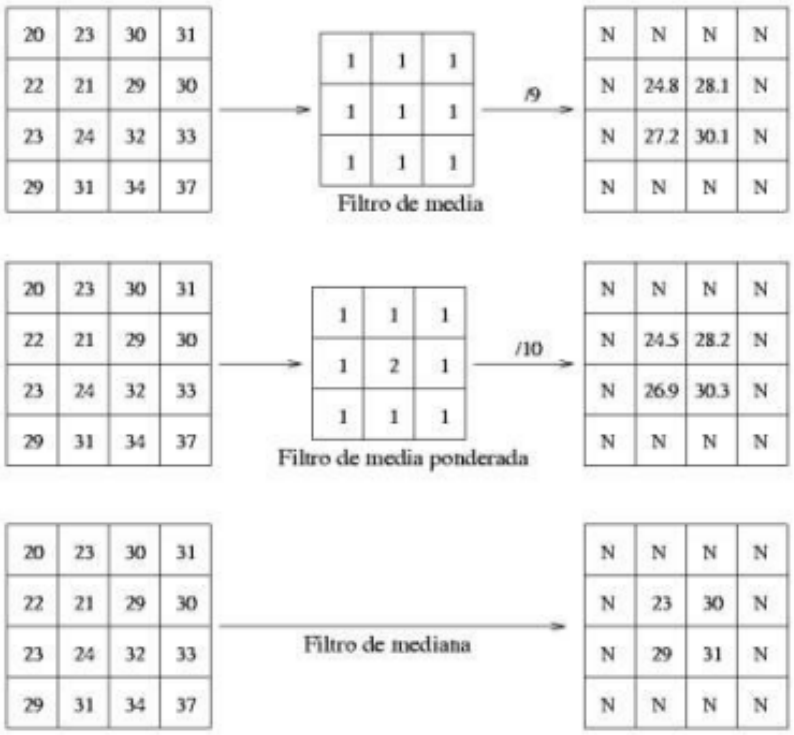
Uno de los métodos para el procesamiento de las imágenes digitales es la aplicación de filtros debido a que La aplicación de estos filtros puede ayudar a la disminución del ruido y suavizar la imagen, algunos tipos de filtros más usados son: (Anonimo, 2022)

- **Filtro de la media:** De la imagen original aplicando este filtro, el cual asigna una media de todos los pixeles de la vecindad, por consiguiente, generando una nueva imagen con efectos de difuminado o suavizado. (Navarrete, 2010)

- **Filtrado de la media ponderada:** Por la matriz principal los números obtenidos no son todos 1, sino con este método se da más peso al número central, por lo cual se obtendrá una imagen muy parecida a la original.
- **Filtro de la mediana:** Se obtiene un valor real y no un promedio el cual es parecido al de la imagen original reduciendo el efecto borroso, su aplicación es difícil debido al cálculo de los valores vecinos con el central de la ventana.

Figura 16

Tipos de filtro y mecanismos de aplicación



Fuente: *(Anonimo, 2022)*

- **Filtro Gaussiano:** El valor máximo se encuentra en el pixel central disminuyendo mientras más alejado se encuentre del centro. El resultado será el conjunto de valores entre 0 y 1. Se divide la matriz por el menor de los valores obtenidos, de manera que este filtro además de remover el ruido presenta empañaduras en la imagen perdiendo la nitidez de esta.

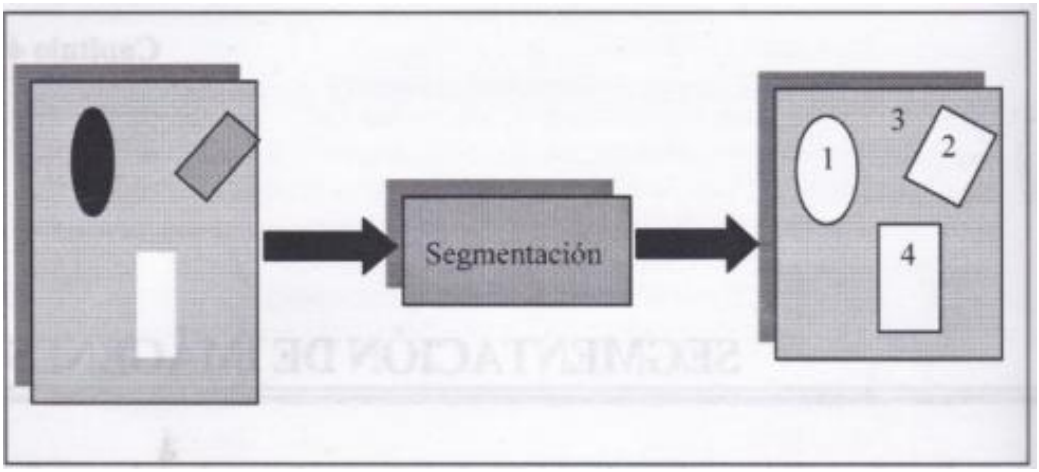
2.3.5.3 Segmentación

Este proceso la imagen se divide en regiones de interés, En esta nueva fase se trata de agrupar los píxeles, por algún criterio de homogeneidad, para particionar en regiones de similares como gris, contraste o texturas.

Es un proceso donde la imagen original produce otra, este proceso se termina cuando se extrae de la imagen distintas regiones, por el proceso de segmentación completa o parcial de la imagen o por partes de interés. En una escena compleja, el resultado es un conjunto de regiones homogéneas superpuestas parcialmente segmentada deberá ser sometida después a una segmentación completa. (José Francisco, Ana Belén, & Ángel Sánchez, 2016)

Figura 17

Segmentación de una Imagen



Fuente: (Garcia, 2008)

Este proceso nos ayuda determinar los objetos que se encuentran en la imagen, pero no existe un método universal de segmentación, por ello se utilizan diferentes métodos para realizar combinaciones de varios de estos métodos y realizar ajustes con los métodos para tratar un problema en particular, algunos de estos métodos son: (Garcia, 2008)

Segmentación por umbral: Este proceso nos permite convertir imágenes de color en binarias, marcando los objetos de interés de los pixeles de fondo, por lo cual en estas condiciones se logra distinguir las características estructurales de la imagen. esta técnica de

segmentación es rápida debido a que se puede realizarla en tiempo real mientras se captura la imagen pro el bajo costo de procesamiento que necesita. (Garcia, 2008)

Segmentación de regiones: Separa las regiones de interés en diferentes regiones agrupadas con las mismas características y conjuntos de pixeles con la misma similaridad, a partir de estas segmentaciones se realizan las medidas de cada región y las que se encuentran adyacentes a esta.

Transformada de Hough: Este método de segmentación permite la detección de curvas en la imagen basándose en las características geométricas de los objetos, esta técnica de segmentación es utilizada por su robustez al ruido y por la existencia de huecos en los objetos. La aplicación se debe realizar en una imagen binaria permitiendo el manejo de los pixeles que forman parte de la frontera, el objetivo de Hough es encontrar puntos alineados lo cual se puede realizar con la siguiente ecuación (5) de la recta en coordenadas polares. (Aguirre, 2013)

$$\rho = x.cos\theta + y.sen\theta \quad (5)$$

2.3.5.4 Extracción de características

Cuando se ha aplicado la segmentación después se procede a aplicar diferentes métodos en al cual se toma una imagen como entrada para la extracción de características o datos de interés, los cuales proporcionan información cuantitativa o permitan la diferenciación de objetos entre clases. Estas características pueden ser área, perímetro, esqueletos, o basadas en la textura. (Pajares & Garcia, 2007)

Los datos que las imágenes contienen son elevados, pero estos no aportan información concreta, los sistemas de visión artificial deben extraer de forma más robusta, eficaz y rápida posible las características de la escena que permitan obtener información adecuada para el proceso de interpretación. Estos sistemas deben cumplir con ciertas características. (Aguirre, 2013)

- La extracción de información debe ser en tiempo real y no presentar un costo excesivo de procesamiento para el sistema.
- La localización de las características debe ser precisa para su extracción, no obstante, si se presentan errores estos deben ser mínimos.
- El método que se emplea para la extracción debe ser confiable, robusto y estable para el sistema.
- Los datos deben tener información de la escena y también información geométrica de la misma.

En este apartado se dará a conocer algunos de los métodos más comunes en la extracción de características.

Detección de Bordes

En visión artificial es importante para el reconocimiento o segmentar regiones, realizar la detección de bordes en las imágenes debido a que permite filtrar la información de los contornos preservando las características de la imagen, en la detección de bordes se detectan los cambios bruscos de intensidad para fijar el contorno de la imagen, igualmente en el método basada en regiones se segmenta para la obtención de regiones homogéneas al hacer la detección de contornos (Ana, 2004), las funciones que permiten obtener los bordes son:

- **Filtros de Prewitt:** Es un operador de diferenciación discreto que usa operadores horizontales y verticales que detectan los bordes en sí, basado en el método gradiente. este vector se basa en la convolución de la imagen aplicando un filtro pequeño, separable y de valor entero. Fue desarrollado por M.S Prewitt. (Priyam & Dipanjan, 2016)
- **Filtros de Burns:** Este filtro tiene como propósito realiza un barrido en sentido lexicográfico (arriba a abajo e izquierda a derecha) en busca de pixeles pertenecientes al mismo grupo, etiquetado a cada pixel que será analizado con sus vecinos en todas las

direcciones resultantes del gradiente, etiquetando a pixeles similares para denominarlos bordes, obteniendo una agrupación en función de su gradiente. (Aguilera, 2010)

- **Detector de bordes de Canny:** Fue creado en 1986 por Candy, este método usa la gradiente máxima para la detección del borde de la figura obteniendo una imagen binaria, este método se basa en tres criterios los cuales son: (Valverde, 2007)
    - **Detección:** Este criterio evita eliminar los bondes importantes para no suministrar falsos bordes.
    - **Localización:** Este criterio establece que la distancia entre el borde y la posición real debe eliminarse.
    - **Respuesta:** Este criterio determina que al integrar varias respuestas múltiples de un solo borde se obtenga una sola respuesta.

  Para este método se emplea la primera derivada con la cual se obtiene el valor de cero de las regiones donde la intensidad no varía, por lo tanto, si se presenta un cambio brusco de intensidad esta aplicara a la derivada, por lo cual se aplicaría tres pasos para aplicación del algoritmo de Canny: (Valverde, 2007)
    - **Obtención de la gradiente:** En cada píxel de la imagen se calcula la magnitud y posición del vector.
    - **Supresión no máxima:** Por los bordes obtenidos con el gradiente se reduce el ancho hasta un píxel.
    - **Histéresis de umbral:** Se aplica la función de histéresis de dos umbrales la cual educe la aparición de entornos falsos.

Detección de Círculos

En visión artificial el método para detectar características geométricas en las imágenes es por medio de la transformada de Hough, de tal manera sesto se realiza remplazando la ecuación del círculo, El círculo es más simple de representar en el espacio, en comparación con

la línea, ya que los parámetros del círculo se pueden transferir directamente al espacio. La ecuación (6) del círculo es la siguiente.

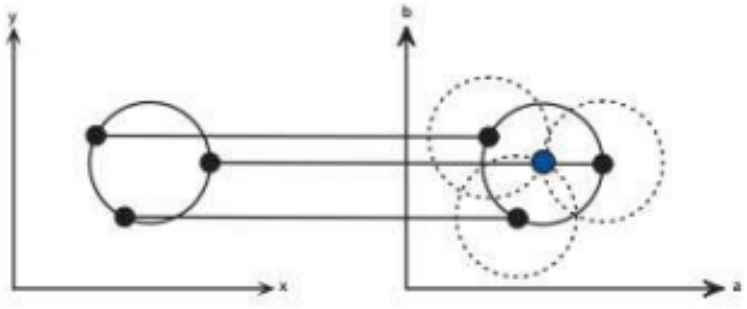
$$r^2 = (x - a)^2 + (y - b)^2 \quad (6)$$

El proceso para detectar círculos en una imagen utilizando la Transformada de Hough es el siguiente: (Pedersen, 2007)

- Detectar los bordes de la imagen usando otro método como Canny
- En los puntos de borde se dibuja un círculo con centro en el punto del radio deseado de tal manera tal que nuestro eje x es el valor a y el eje y es el valor b mientras que el eje z eje son los radio.
- En las coordenadas del perímetro del círculo, se incrementa una matriz acumuladora la cual tiene el tamaño de los parámetros. permitiéndonos barrer sobre cada punto del borde con el radio deseado e incrementando los valores en el acumulador.
- El acumulador tendrá un numero de datos correspondiente al círculo que pasan a través de estas coordenadas individuales.
- Los números más altos, corresponden con el centro de los círculos en la imagen.

Figura 18

Transformada de Hough circular



Fuente: *(Pedersen, 2007)*

2.4. Software Libre

El software libre se considera a todo software en el cual el usuario cuanta con la libertad de: copiar, estudiar y modifica, dando como resultado la posibilidad de crear o controlar un programa, dentro de estas plataformas que permiten la ejecución o elaboración de proyectos en el campo de la visión artificial se pueden encontrar: los lenguajes de programación, librerías de visión artificial, bases de datos los cuales serán explicados a continuación.

2.4.1 Lenguajes de Programación

Los lenguajes de programación se encargan de construir aplicaciones las cuales permiten la comunicación entre el programador y la máquina, de manera que una persona pueda establecer instrucciones permitiendo interactuar directamente con el hardware. Estas instrucciones se adjuntan en grupos permitiendo desarrollar programas que son los pasoso que sigue el ordenador para completar una tarea (Guevara, 2008).

Uno de los lenguajes de programación que se aplica en el campo de la visión artificial es el de Python por su versatilidad a continuación se detallara más de este.

2.4.1.1 Python

Es un lenguaje de alto nivel el cual se caracteriza por sencillo y legible, se trata de un lenguaje de programación orientada a objetos o programación funcional, por lo cual este lenguaje a ganando campo en el desarrollo de aplicaciones y web, por consiguiente, cuenta con algunas características que son: su robustez, soporte a múltiples plataformas, fácil de aprender y constante actualización. El cual el permite ser utilizado en las aplicaciones web, desarrollo de software, ciencia de datos y el machine learning.

Debido a que este lenguaje se utiliza en diferentes campos, entre ellos el de visión artificial o visión por computadora el cual cuenta con una amplia gama de librerías permitiendo el manejo de matrices e imágenes. Este lenguaje cuenta con diferentes bibliotecas las cuales

agregan muchas más herramientas para el desarrollo de proyectos o prototipos. (Pajankar, 2017).

2.4.2 Librerías de Vision Artificial

En la actualidad existen diversos paquetes de visión por computadora y procesamiento de imágenes las cuales destacan por su robustez y flexibilidad, debido a esto se han desarrollado paquetes tanto para el entorno comercial que están ligados a una plataforma para su desarrollo entre estos tenemos: Khoros de Linux, Exbem de MacOS y eVision, Aphelion de Windows. Todos ellos proporcionan funciones básicas y avanzadas de procesamiento y análisis de imágenes, reconocimiento de patrones, estadísticas, calibración de la cámara. Por otro lado, también existen paquetes no comerciales "Software Libre" entre estos estan: OpenCV, Gandalf, TargetJr, ImageLib, ImLib3D, LookingGlass, NeatVision, TINA. Todos estos al igual que el comercial tienen las mismas herramientas excluyendo a OpenCV y Gandalf los cuales no cuentan con un marco de trabajo completo en visión artificial. (Ambrosio, González, & Arévalo, 2004)

Las funciones de las librerías de visión artificial se pueden clasificar dependiendo de la información a obtener, las cuales pueden ser:

- **Visualización:** La función a destacar es la detección de objetos no visibles que se encuentran en una imagen.
- **Reconocimiento:** Son usadas con el fin de reconocer o detectar objetos en las imágenes como: personas, objetos, animales o características de interés de la persona.
- **Nitidez y restauración:** Son usadas con el fin de mejorar o restaurara la imagen de entrada mejorando su calidad.
- **Reconocimiento de patrones:** Ayuda con la detección de ciertos patrones en las imágenes.

- **Recuperación:** Se encarga de buscar y encontrar imágenes similares en un cierto rango

Con las funciones y métodos establecidos se puede determinar ciertas librerías de visión artificial con el propósito de obtener información que pueda ser procesada por una computadora. Entre estas librerías tenemos:

2.4.2.1. OpenCV

OpenCV The Open Computer Vision. Fue creada por Intel Corporation, es una librería libre especializada en el procesamiento de imágenes el cual cuenta con aproximadamente 500 funciones que abarcan varias áreas en el proceso de visión artificial, de modo que se le aplica en proyectos como reconocimiento de objetos o reconocimiento facial, calibración de cámaras, visión estereoscópica, visión robótica, entre otras. (Arévalo, González, & Ambrosio, 2023)

2.4.2.2. Metaplotlib

Esta librería utiliza trazos en 2D permitiendo el manejo de matrices y el uso de funciones como leer y mostrar imágenes, de tal manera con el uso de la biblioteca de numpy se pueden visualizar los datos en multiplataforma, algunos ejemplos de la aplicación de esta librería son: histogramas, diagramas de barra, potencia con sus espectros y diagrama de errores.

2.4.2.3. PyTorchCV

Está basado en Pytorch bajo un modelo de aprendizaje profundo enfocado para visión artificial. Es una librería enfocada en los modelos de clasificación, segmentación, detección y estimación de poses en imágenes, permitiendo desarrollar un sistema de etiquetas para problemas de clasificación. Está compuesta por una serie de modelos los cuales son: AlexNet, ResNet, ResNeXt, PyramidNet, SparseNet, DRN-C/DRN-D.

2.4.2.4. SimpleCV

Se especializa en crear aplicaciones de visión artificial de una manera sencilla, permitiendo el uso de diversas bibliotecas para el procesamiento de imágenes, entre las cuales

esta OpenCV, de tal manera que se puedan tratar las imágenes de una manera más simple sin tener que establecer de una manera más profunda cierto parámetros como profundidades de bits, formatos de archivo, espacios de color, administración de búfer, valores propios o almacenamiento de matrices, por consiguiente permitiendo al usuario trabajar con imágenes o secuencias de video que provienen de cámaras web, Kinects, FireWire y cámaras IP, o teléfonos móviles. (Sight, 2023)

2.4.2.5. NumPy

Es una biblioteca de código abierto desarrollada en Python, la cual permite el desarrollo numérico de matrices enfocado en los pixeles para el procesamiento de imágenes, por lo cual se enfoca en ciertos parámetros como: recorte de imágenes, manipulación de pixeles, enmascaramiento de valores. Por lo tanto, debido al manejo más amplio que ofrece en las matrices ayuda a reducir el color, binarización, nitidez de una imagen. (NumPy, 2023)

2.5. Estándar IEEE 29148

El estándar IEEE 29148, es una norma internacional que provee de parámetros y herramientas para el desarrollo de proyectos en el campo de la ingeniería, el cual permite la identificación de ciertos requerimientos en sistemas y productos de software. Esta norma es el resultado de ciertos estándares involucrados para un solo objetivo, los cuales son:

- ISO/IEC 12207:2008, Systems and software engineering − Software life cycle processes.
- ISO/IEC 15288:2008 Systems and software engineering − System life cycle processes.
- ISO/IEC/IEEE 15289:2011, Systems and software engineering − Content of lifecycle information products (documentation).
- ISO/IEC TR 19759, Software Engineering − Guide to the Software Engineering Body of Knowledge (SWEBOK).

- IEEE Std 830, IEEE Recommended Practice for Software Requirements Specifications.

Esta normativa proporciona los requerimientos relacionados con el sistema permitiendo el levantamiento y gestión de actividades para el desarrollo del sistema y software, con ciertos estándares que son:

- Stakeholders. - Es el conjunto de requisitos del sistema que son solicitados por los usuarios, clientes o partes involucradas.
- System requirements specification Este documento describe las características y el comportamiento del sistema.
- Software requirements specification.- Este documento establece información de los equipos involucrados como: desarrollo, garantía de calidad, operaciones y mantenimiento.

2.6. Metodología

Según (Torres, 2009), se define a la metodología como " Un conjunto de técnicas y métodos que permite abordar de forma homogénea cada una de las actividades del ciclo de vida de un proyecto", por consiguiente se considera siempre el uso de las metodologías en el desarrollo de un proyecto la cual permite trazar un punto de inicio donde se puede tratar y gestionar el proyecto con el objetivo de tener éxito en este, a continuación, se referencia la metodología para realizar un proyecto de prototipo.

2.6.1 Método de Cascada

Este modelo fue propuesto por Winston Royce, el cual ordena de forma rigurosa las etapas de vida del proyecto, debido a que este método se basa en un desarrollo secuencial de modo que al inicio de cada etapa esta debe esperar que la anterior se encuentre finalizada para continuar con el desarrollo (Torres, 2009). El modelo en cascada se divide en 5 fases según (Cervantes & Gómez, 2012) las cuales son:

Análisis y definición de requerimientos

En esta fase de inicio se trabaja con los clientes y usuarios finales a quienes beneficiara el sistema, estos ayudaran a determinar los parámetros y restricciones para el desarrollo del proyecto, elaborando el documento de Requerimientos del sistema.

Diseño del sistema y del software

Debido a los parámetros del anterior paso en esta fase de diseño del sistema se establecen los requerimientos. Después se establece la arquitectura completa del sistema especificando los subsistemas y cómo funciona cada uno.

Implementación y validación de unidades

Esta fase consiste en la prueba de los subsistemas por separado verificando el correcto funcionamiento establecido anteriormente en las especificaciones previamente establecidas.

Integración y validación del sistema

Probado cada subsistema se unifican para crear el prototipo completo verificando que cada requerimiento se haya cumplido tanto en la parte de hardware como la de software, permitiendo así pasar a las pruebas de funcionamiento, de manera que si se presenta algún error cualquier sistema se pueda corregirlo.

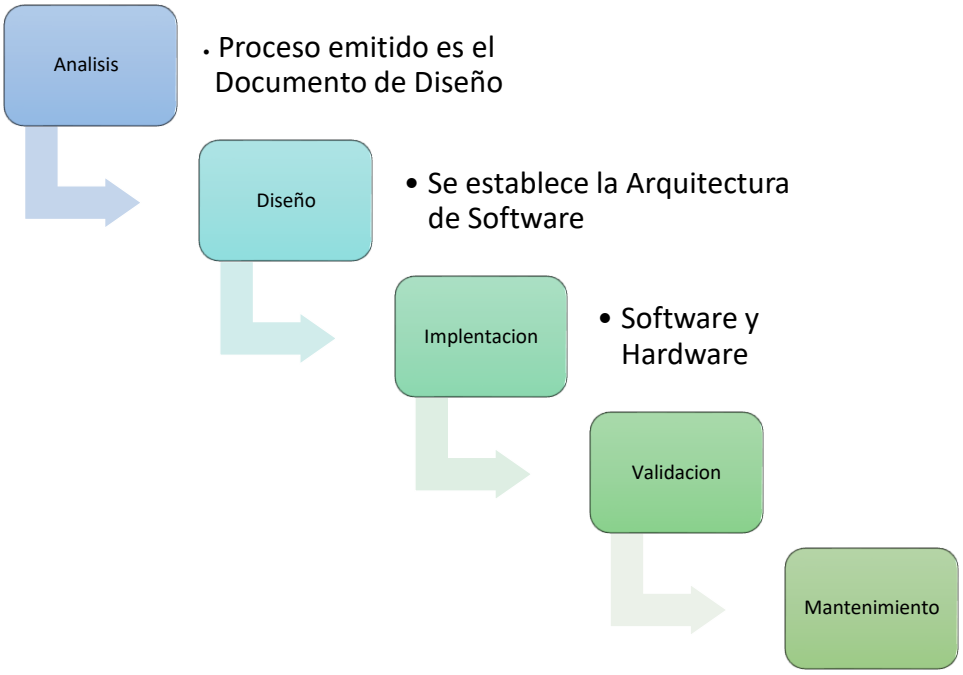
Funcionamiento y mantenimiento

El sistema se instala y se verifica el funcionamiento, dado que si se presenta algún error en alguna de las etapas anteriores se pueda corregir y mejorarlo dando una mayor robustes al funcionamiento del sistema.

Para finalizar este método se implica seguir una serie de pasos secuenciales para completar una tarea. Cada paso tiene que ser completado antes de pasar al siguiente. En el contexto de la investigación, los pasos pueden incluir la definición del problema, la revisión de la literatura existente, la formulación de una hipótesis, el diseño del estudio, la recolección de datos, el análisis de los resultados y la presentación de las conclusiones.

Figura 19

Diagrama Metodología de cascada



Fuente: Adaptado de *(Desarrolladores Web , 2022)*

Capítulo 3. Diseño del Sistema

En el presente capitulo se establecieron las dos primeras fases del método de cascada, en el cual se describió a la Clínica MediFisio la cual ayudó permitiendo acceder al centro de Fisioterapia para el desarrollo de este proyecto de investigación. Con la ayuda de este departamento se realizó un documento que contenía los requerimientos y especificaciones de diseño del sistema. Por consiguiente, también se delimitaron a los usuarios para realizar las pruebas de funcionamiento del sistema.

En el siguiente apartado se explicaron la fase 1 del proyecto en la cual consta de la investigación y análisis del problema identificado anteriormente en la sección 1.2. Para esto se empezó realizando un análisis de los requerimientos.

3.1. Fase 1: Análisis de requerimientos

En lo referente a la fase inicial, se dieron a conocer los requerimientos propuestos por el departamento de fisioterapia para el desarrollo del proyecto de investigación. Se establecieron los parámetros tanto de hardware como de software para el desarrollo del dispositivo, con el fin de establecer un sistema versátil y funcional para la adquisición del ángulo Cifolumbar, además de permitir que la información obtenida se guarde en una base de datos. En cuanto a las pruebas de funcionamiento, se tomó a un grupo de pacientes del centro de fisioterapia, los cuales ayudaron a realizar las pruebas de funcionamiento para la verificación y rendimiento del sistema, con el objetivo de validarlo o corregir los errores, lo que permitió mejorar el dispositivo para un óptimo uso. Después, se procedió al análisis de los requerimientos y especificaciones del sistema embebido, a continuación, se da una breve introducción de la Clínica MediFisio.

3.2. Clínica MediFisio

La Clínica MediFisio es un centro médico que brinda servicios integrales de salud a las personas de manera oportuna y cordial con los mayores estándares de calidad para atender de

manera eficaz y eficiente las necesidades de los ciudadanos, brindando la mejor atención medica basada en la evidencia científica y contenido ético, por lo cual cuenta con un equipo médico de calidad.

A continuación, se realizó un análisis sobre la situación actual del departamento, tomando en cuenta el grupo de pacientes tratados en un periodo de tiempo junto con un análisis de las patologías que estos presentaron y si estos requerían un proceso de evaluación.

3.3. Análisis

3.3.1. Situación Actual

La Clínica MediFisio es un centro de salud el cual brinda servicios, con profesionales capacitados que brindan sus servicios con la intención de mejorar la calidad de vida de los pacientes, por esto un servicio que brinda la clínica es el de evaluación y rehabilitación en el campo fisioterapia, así pues, la clínica recibe un sin número de pacientes que necesitan ser evaluados y diagnosticados para determinar el tipo de patología que tienen.

Con datos obtenidos del Centro de Rehabilitación en los meses de Enero - Mayo del 2022, se han atendido a un aproximado de 18 pacientes, que corresponden a un 35 % mujeres y un 65% hombres que viven por el sector donde se encuentra la clínica.

Por otro lado, en los pacientes se determinó que un grupo de estos acuden debido a problemas o malestares físico para realizarse una evaluación, solo un 40% los pacientes se les practico un proceso de evaluación con el cual se concluyó un diagnóstico. El otro 60% han ingresado para los tratamientos de rehabilitación.

En el análisis realizado, se sustentó la ejecución del proyecto, el cual permitió el desarrollo de una herramienta alternativa de evaluación con el uso de la tecnología. Esta herramienta permitió la obtención de datos valederos en un menor tiempo tomando en cuenta la comodidad del paciente en el proceso de evaluación. Por otro lado, el sistema contó con una

base de dato personalizada la cual permitió el control de la información y almacenar los resultados obtenidos.

En lo referente al centro de fisioterapia, se llevaban a cabo procesos de evaluación manuales, Sin embargo, con la propuesta del proyecto se automatizaron ciertos métodos de evaluación, haciendo que fueran más rápidos tanto para el paciente como para el fisioterapeuta. A continuación, se presentó el sustento en el que se basó el desarrollo del proyecto.

3.3.2. Técnicas de Investigación

En el área de Fisioterapia, se realizan procesos de terapias o evaluaciones. El dispositivo que se desarrolló se utilizó como instrumento alternativo de evaluación no invasivo para ciertas patologías en la región de la columna vertebral, las cuales fueron Hipo/Hiper Cifosis o Lordosis. La mayoría de estas patologías se desarrollan debido a malos hábitos, pero en el último tiempo, debido a la pandemia y a la realización de ciertas actividades, como clases en línea o teletrabajo, muchas personas presentaron deformidades en la columna vertebral, lo que degradó su calidad de vida. Como resultado, muchos pacientes han acudido a la clínica para recibir evaluaciones y tratamiento.

A partir de lo anteriormente expuesto, se establecieron condiciones para el desarrollo y funcionamiento del sistema, las cuales se obtuvieron mediante una entrevista al fisioterapeuta. Con la cual, se establecieron los requerimientos del sistema que permitieron dar un punto de partida al desarrollo del proyecto.

3.3.3. Propósito y Ámbito del Sistema

Las alteraciones postulares se presentan a lo largo de la vida, afectando las actividades cotidianas de una persona. Los métodos manuales, como la flecha sagital, el método de Cobb, el inclinómetro, el cifómetro y el método de examen subjetivo/objetivo, se emplearon para la realización de evaluación postulares. Sin embargo, con el desarrollo de este sistema electrónico se propuso mejorar el método de la flecha sagital mediante imágenes para obtener ciertos

parámetros en la columna vertebral, por ejemplo, las flechas en las zonas cervical, lumbar sacro y toráxico. Además, se desarrolló un método programado para la identificación de estos puntos, lo que permitió aplicar el principio de la Flecha Sagital para obtener las distancias de estos puntos, los cuales ayudaron a la obtención de los índices cifóticos y lordicos. Posteriormente, se compararon con referencias ya establecidas para diagnosticar el hipo/hiper lordosis o cifosis.

En cuanto al dispositivo embebido, este fue desarrollado en una placa específica de visión artificial, lo que permitió un mejor procesamiento de imágenes en un corto tiempo. Además, esta placa permitía realizar funciones de adquisición y visualización de datos. Se conectó una cámara para permitir la adquisición de imágenes y posteriormente, la placa se encargó del procesamiento y envío de los datos del paciente hacia la base de datos, obteniendo así un historial clínico digitalizado con el cual se puede llevar un seguimiento del paciente.

Este proyecto se limitó a proponer una alternativa de evaluación diferente a la tradicional, por la cual, por medio de un dispositivo, se permitió la obtención del ángulo cifolumbar para el diagnóstico de la Hipo/Hiper lordosis o cifosis, como se explicó anteriormente. A continuación, se realizó una descripción general del dispositivo a desarrollar.

3.3.4. Descripción General del Sistema

Como se abordó en el apartado 1.4, el dispositivo se desarrolló usando la placa de visión artificial Nvidia Jetson Nano, debido a que esta realiza operaciones de procesamiento en el campo de la visión artificial de una manera más rápida y compleja. Igualmente, gracias a su versatilidad, es apta para el uso en sistemas embebidos, por lo cual se le conectó un grupo de dispositivos los cuales permitieron el uso de la placa de una manera más ágil y adecuada.

Con la imagen previamente capturada, se procedió al almacenamiento en la placa, lo que permitió usarla para realizar el respectivo procesamiento. Estas imágenes se procesaron por medio de un método de binarización, que consistió en un proceso de conversión de RGB a "Blanco y Negro". A continuación, se delimitaron los bordes en la imagen y se identificaron

los puntos de interés. Para finalizar, se suavizó la imagen eliminando el ruido y se procedió a su segmentación.

Teniendo en cuenta el método de la flecha sagital, se elaboró un proceso que permitió obtener los puntos de interés correspondientes a las distancias tomadas de los 4 puntos de la columna vertebral. Una vez dados estos datos, se aplicaron las fórmulas establecidas para la obtención del ángulo cifolumbar. Estos ángulos podían ser utilizados para dar un diagnóstico y determinar la situación actual del paciente.

Los datos del paciente se procesados de la siguiente manera: se elaboró una ficha de ingreso donde se tomó la información principal del paciente antes de iniciar el proceso de evaluación. Esto permitió llevar un registro personalizado de cada evaluación. Posteriormente, se llevó a cabo el proceso de evaluación utilizando la imagen o el video obtenidos, aplicando el algoritmo previamente programado para obtener tanto el ángulo e índice cifolumbar. Finalmente, tanto el registro como el proceso de evaluación se guardaron en una base de datos local.

En cuanto al sistema electrónico, este estuvo conformado por una cámara digital que permitió la obtención de imágenes, las cuales pasaron a ser procesadas. En la parte de la visualización, esta se realizó por medio de un monitor o display que estaba conectado a la placa, donde también estaban conectados otros dispositivos como el teclado y ratón que permitieron el uso del sistema de una manera más rápida.

Dada una descripción general del sistema, funcionamiento y el objetivo, se procedió a establecer las características, las cuales permitieron establecer los riesgos y restricciones, Estas se definieron a continuación.

3.3.5. Características del Sistema

Como punto de partida, se desarrolló una investigación que ayudó a crear una alternativa de evaluación dentro del área de fisioterapia, la cual permitió mejorar los métodos

manuales mediante la Visión Artificial, como en este caso el método de la Flecha Sagital. El proyecto se desarrolló en una placa específica de visión artificial como base, y se establecieron algunas restricciones como se detalla a continuación.

3.3.5.1. Restricciones del Sistema

- El sistema necesita una adquisición de imágenes optimas obtenidas por una cámara de resolución aceptable, cuya imagen permitiera diferenciar los puntos de interés en el paciente.
- El sistema embebido debía tener un tamaño adecuado para su uso.
- El sistema debía apegarse al método mencionado anteriormente para las respectivas mediciones.

3.3.5.2. Riesgos del Sistema

- Problemas de compatibilidad entre la cámara y la placa.
- Tener un margen de error elevado al momento de realizar las pruebas.
- Aumento de carga en el dispositivo que haga fallar el funcionamiento de este.

3.4. Especificación de Requerimientos

Para el presente proyecto se realizó el análisis de los requerimientos tomando como referencia el estándar ISO/IEC/IEEE 29148-2018, debido a que especificaba cada función de los procesos requeridos para sistemas y productos de software a lo largo del ciclo de vida de los proyectos.

Teniendo en cuenta el estándar, se diseñaron las tablas de requerimientos relevantes en base a criterios de prioridad para el desarrollo del sistema, así como de la arquitectura y los stakeholders. Esto permitió presentar dicha información de una manera práctica en cuanto a la selección de elementos del sistema.

3.4.1. Stakeholders

Para el desarrollo del siguiente proyecto se tomó en cuenta a las personas implicadas en este, denominadas Stakeholders los cuales son: el fisioterapeuta quien es el que determino las características fundamentales del sistema. Además, se involucraron a los responsables de este proyecto, como el director, asesor y el autor del trabajo de titulación, los cuales se muestran en la siguiente Tabla 3.

Tabla 3

Stakeholders del proyecto

Lista de Stakeholders	
Fisioterapeuta	Lic. Estefanía Hidalgo
Director del Proyecto	Msc. Jaime Michilena
Asesor del Proyecto	Msc. Luis Suárez
Desarrollador del Proyecto	Fabricio García

Fuente: Autoría

Una vez que se estableció a las personas involucradas en el desarrollo del proyecto, se procedió a explicar los niveles y nomenclatura que se establecieron en el siguiente apartado, los cuales se usaran en este proyecto.

3.4.2. Nomenclatura de los Requerimientos

Los Stakeholders del proyecto fueron considerados por las personas que participaron directa o indirectamente de este. Por consiguiente, era primordial conocer la nomenclatura de cada uno de los requerimientos en la Tabla 4, a continuación, se mostró los acrónimos empleados.

Tabla 4

Nomenclatura de los requerimientos

No	**Requerimientos**	**Nomenclatura**
1	Especificaciones del Stakeholders	STRS
2	Especificaciones del Sistema	SYSR
3	Especificaciones de Arquitectura	SRSH

Fuente: Autoría

Se debió incluir ciertos parámetros en el diseño propuesto donde se indicó la prioridad del requerimiento, siendo estos Alta, Media y Baja. Esta valoración se visualizó en la Tabla 5**,** debido a la importancia para la selección de los elementos del sistema. Adicionalmente, se añadió una columna de relación que se usaría en el caso de que un requerimiento fuese totalmente dependiente de otro.

Tabla 5

Prioridad de los Requerimientos del sistema

Prioridad	Descripción
ALTA	Se trato de un requerimiento critico que se incluye en el desarrollo del sistema, si no se considera puede afectar la funcionalidad del sistema.
MEDIA	Si no se incluye este tipo de requerimientos puede afectar al funcionamiento del sistema, pero pueden omitirse en caso de fuerza mayor.
BAJA	Estos requerimientos son circunstanciales pueden omitirse debido a que no tienen un gran impacto en el sistema.

Fuente: Autoría

3.4.3. Requerimientos de Stakeholders

Los requerimientos de stakeholders fueron considerados como los requerimientos dados por el grupo o individuo a cargo del desarrollo del proyecto. La definición de estos requerimientos (STRS) fue un análisis realizado de las necesidades de los pacientes y usuarios. Cabe resaltar que se limitó a realizar una propuesta de evaluación alternativa, en otras palabras, se digitalizó un método de evaluación manual con ayuda de la tecnología. A continuación, en la Tabla 6, se mostraron los requisitos de los implicados o stakeholders.

Tabla 6

Requerimientos Stakeholders STRS

REQUERIMIENTOS STAKEHOLDERS STRS			
#	**REQUERIMIENTOS**	**PRIORIDAD**	**Relación**

		Alta	Media	Baja	
REQUERIMIENTOS OPERACIONALES					
STRS 1	El sistema debe implementarse en el área de Fisioterapia.	X			
STRS 2	El dispositivo no debe presentar interrupciones durante su funcionamiento.	X			
STRS 3	El dispositivo debe contar con conexión a internet.		X		
STRS 4	El sistema de adquisición debe contar con un rango entre el paciente y el dispositivo para ejecutar la evaluación.	X			STRS 5
STRS 5	Tanto los datos obtenidos por el sistema y el diagnostico se deben visualizar.	X			
REQUERIMIENTOS DE USUARIO					
STRS 6	La visualización de los datos obtenidos debe ser en tiempo real.	X			
STRS 7	La información del paciente debe ser registrada adecuadamente en una ficha médica.	X			
STRS 8	El proceso de evaluación se contemplará en el plano sagital.	X			

Fuente: Autoría

3.4.4. Requerimientos del Sistema

En los requerimientos del sistema (SYSR), se concretaron las limitaciones, operaciones y funciones del sistema, como se muestra en la Tabla 7, los SYSR son requisitos que se transmitieron del usuario o paciente a la persona encargada del proyecto.

Tabla 7

Requerimientos Stakeholders SYSR

REQUERIMIENTOS SISTEMA SYSR					
#	**REQUERIMIENTO**	**PRIORIDAD**			**Relación**
		Alta	**Media**	**Baja**	
REQUERIMIENTOS SISTEMA					
SYRS 1	El proceso de adquisiciones de imágenes debe ser rápida.	X			
SYRS 2	El sistema debe contar con la capacidad de soporte.		X		
SYRS 3	El sistema debe cotar con una base de datos para almacenar la información del paciente.	X			
REQUERIMIENTOS INTERFAZ					

SYRS 4	El sistema embebido debe contar con una interfaz visible.	X		
SYRS 5	La interfaz del sistema debe ser intuitiva y fácil de usar.	X		
SYRS 6	Capacidad de almacenamiento y visualización de datos.	X		
SYRS 7	El sistema debe mostrar la detección del ángulo Cifolumbar a través del software.	X		
REQUERIMIENTO DE PERFORMANCE				
SYRS 8	Reconocimiento de las distancias mediante la flecha sagital para el cálculo del ángulo cifolumbar.	X		
REQUERIMIENTOS FÍSICO				
SYRS 9	La ubicación del sistema debe estar situado a la distancia establecida para la obtención de la imagen.	X		
SYRS10	La cámara debe tomar una imagen nítida del paciente, donde los aspectos como la luminosidad y ruido en sean mínimos	X		

Fuente: Autoría

3.4.5. Requerimientos de Arquitectura

En los requerimientos de arquitectura (SYSH), se definieron los requerimientos solicitados, tales como: software, hardware, diseño y sistema eléctrico. Estos hitos fueron analizados para cumplir la petición del operador y los usuarios, permitiendo el correcto diseño y funcionamiento del dispositivo. Como se puede observar en la Tabla 8, se presentan estos requerimientos.

Tabla 8

Requerimientos Stakeholders SYSH

REQUERIMIENTOS SISTEMA SYSH					
#	**REQUERIMIENTO**	**PRIORIDAD**			**Relación**
		Alta	**Media**	**Baja**	
REQUERIMIENTOS DE DISEÑO					
SYSH 1	El dispositivo debe ser compacto en su mayoría.		X		
SYSH 2	La cámara debe estar ubicada en una base para la medición de los ángulos.		X		
SYSH 3	El dispositivo debe contar con una pantalla adecuada que permita la visualización del sistema.	X			
REQUERIMIENTOS HARDWARE					
SYSH 4	Sistema embebido debe contar con una entrada para la conexión de la cámara de video.	X			
SYSH 5	La placa debe contar con un procesador capaz de analizar imágenes en tiempo real.	X			
SYSH 6	Verificar las especificaciones del hardware antes de su elección.	X			
SYSH 7	El sistema debe contar con una cámara de alta resolución para la efectividad de las tareas de detección y reconocimiento.	X			

SYSH 8	Las imágenes deben tener un procesamiento rápido en el GPU.	X	
SYSH 9	La placa debe contar con una capacidad de almacenamiento de 128 GB para el sistema.	X	
SYSH10	Adquisición de los dispositivos rápida		X
SYSH11	Sistema embebido debe tener conexión a internet.	X	
	REQUERIMIENTO DE SOFTWARE		
SYSH12	El dispositivo debe contar con un sistema operativo basado en Open Source.	X	
SYSH13	Lenguaje de programación de código abierto.	X	
SYSH14	Compatibilidad de OpenCV y el dispositivo de captura.	X	
SYSH15	El software debe ser compatible con la placa seleccionada.	X	
SYSH16	Se requiere una base de datos de bajo impacto o costo de procesamiento para el rendimiento del sistema.	X	
	REQUERIMIENTOS ELÉCTRICOS		
SYSH17	Fuente de voltaje de 5V	X	

Fuente: Autoría

3.5. Fase 2 Diseño del Sistema

Una vez culminado el análisis con el personal de fisioterapia, se determinó los respectivos requerimientos del sistema. En resumen, se procedió a realizar una comparativa de algunos elementos para el desarrollo del proyecto. Los componentes se evaluaron dependiendo de los atributos correspondientes en (STRS, SYSR, SRSH), definiendo que los valores se puntuaban de “1” si cumplían el requerimiento y de “0” si no cumplían, lo que permitió optar por el valor más alto para el desarrollo del proyecto.

3.5.1. Elección de Hardware

Los elementos de hardware fueron determinados de acuerdo con los requerimientos solicitados previamente en la Tabla 8. En esta sección se especificó la placa que se utilizaría, que contaba con capacidades necesarias para el desarrollo del sistema. Además, se detalló la cámara que se utilizaría para adquirir las imágenes del sistema. Posteriormente se describió a fondo las características de esta placa en la siguiente sección.

3.5.1.1. Selección de placa

Para la realización de este sistema embebido se llevó a cabo un análisis previo en el cual se encontró la placa de visión artificial que se ajustaba a los requisitos para el desarrollo

del sistema, los cuales se pueden observar en la Tabla 9**Tabla *9*** Elección de la placa visión artificial. En consecuencia, la placa en al cual se desarrollo fue la NVIDIA Jetson Nano, ya que es una placa recomendada en este campo debido a sus capacidades de procesamiento y especificaciones.

Tabla 9

Elección de la placa visión artificial

	REQUERIMIENTOS						
HARDWARE	**SYSH 4**	**SYSH 5**	**SYSH 8**	**SYSH 9**	**SYSH12**	**SYSH17**	**Valoración**
NVIDIA Jetson Nano	1	1	1	1	1	1	6

Fuente: Autoría

La placa cuenta con un procesador Quad-core ARM A57 y una GPU NVIDIA Maxwell de 128 núcleos, y es altamente recomendable en el campo de la visión artificial debido a sus capacidades de procesamiento y especificaciones. Asimismo, se destaca su soporte para múltiples redes neuronales y frameworks de aprendizaje profundo, capacidad para procesar múltiples streams de video simultáneamente, soporte para interfaces de cámara de alta resolución, compatibilidad con sistemas operativos populares de Linux, bajo consumo de energía y tamaño compacto, lo que la hace fácil de integrar en proyectos personalizados. En la siguiente sección se presentarán las características detalladas de esta placa.

Tabla 10

Características técnicas de Nvidia Jetson Nano

NVIDIA Jetson Nano	
CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	128-core Maxwell
Memoria	4 GB 64-bit LPDDR4 25.6 GB/s
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

Fuente: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

3.5.1.2. Selección de Cámara

Una vez que se había seleccionado la placa para el sistema embebido, se exploraron opciones de compatibilidad con la NVIDIA Jetson Nano y las librerías a utilizar. Por lo tanto, se decidió utilizar el puerto USB para la cámara, lo cual proporcionaría una opción adicional de movilidad para el sistema. Los detalles de esta opción se detallan en la *Tabla 11*.

Tabla 11

Selección cámara de adquisición

HARDWARE	REQUERIMIENTOS			Valoración
	SYSH 2	SYSH 7	SYSH10	
Módulo cámara para Raspberry Pi de 5MP 1080p OV5647	0	1	1	2
Webcam camera logitech c270	1	0	1	2
Cámara web w77 1080p	1	1	1	3

Fuente: Autoría

Entre las tres opciones comparadas en la tabla anterior, se seleccionó la cámara web W77 1080p debido a que cumplía con todos los requerimientos establecidos en la arquitectura. La parte principal era la resolución de 1080p/30 fps, con la cual permitía capturar imágenes de calidad óptima. Esta cámara resultó ser una buena elección debido a su precio accesible y su movilidad, lo que hizo que el dispositivo fuese más ágil y rápido al momento de tomar fotografías. Es importante destacar que tanto la placa como el dispositivo utilizan una conexión USB.

3.5.2. Elección de Software

De acuerdo con las selecciones de hardware establecidas previamente, se llevó a cabo un análisis para determinar el sistema operativo compatible para la placa. La página de Nvidia proporcionó uno basado en Ubuntu con ciertas librerías preinstaladas para visión artificial. Una vez que se estableció lo anterior, se procedió a la selección del software basada en los requerimientos establecidos en la Tabla 8..

3.5.2.1. Lenguaje de programación

Se realizó un análisis para seleccionar el lenguaje de programación que mejor se adaptara a los requerimientos establecidos. Se evaluaron opciones que ofrecían compatibilidad para el procesamiento de imágenes y se adecuaban para el uso de la biblioteca OpenCV. Es importante mencionar que estos lenguajes de programación son de código abierto, como se muestra en la Tabla 12, donde se presentan las alternativas propuestas.

Tabla 12

Selección lenguajes de programación

SOFTWARE	Requerimientos			Valoración
	SYSH13	SYSH14	SYSH15	
Python	1	1	1	3
Matlab	1	0	1	2
Java	1	1	1	3

Fuente: Autoría

Se analizaron los lenguajes de programación en la Tabla *12* Selección lenguajes de programación, se encontró que ambos satisfacían los requerimientos establecidos, específicamente Python y Java. Como resultado, se seleccionó Python debido a que es óptimo para visión artificial como para el desarrollo de aplicaciones y dispone de amplios recursos que ayudaron a acelerar el desarrollo del proyecto.

3.5.2.2. Base de Datos

Se llevó a cabo un análisis para elegir una opción de servidor de datos que se adaptara y fuera compatible con los requerimientos previamente mencionados. Se compararon opciones que ofrecían compatibilidad tanto para el lenguaje de programación como para los recursos consumidos en el sistema, y se consideró que estas bases de datos fueran de código abierto. En la siguiente tabla se muestran las opciones que se consideraron.

Tabla 13

Selección Base de Datos

SOFTWARE	Requerimientos			Valoración
	SYSH12	SYSH15	SYSH16	
SQLite	1	1	1	3
MySQL	1	1	0	2
PostgreSQL	1	1	0	2

Fuente: Autoría

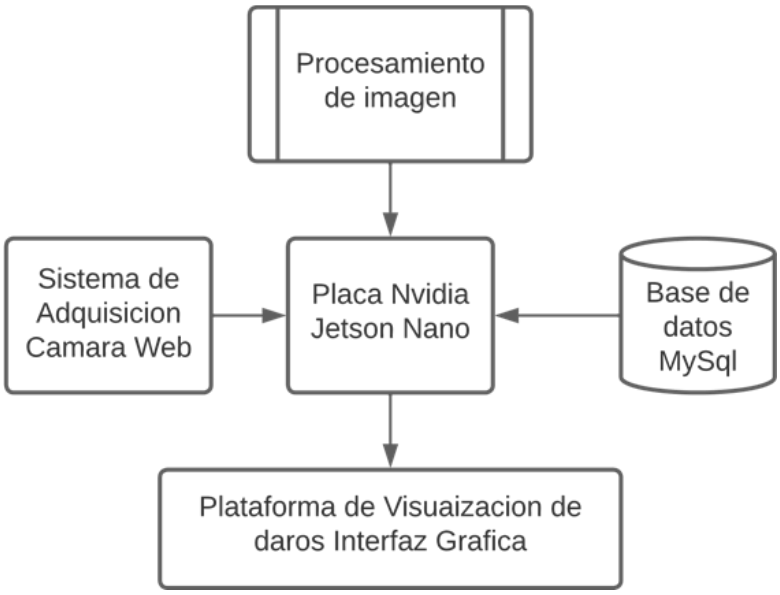
Después de comparar las bases de datos en la Tabla 13, se determinó que la opción más adecuada para el proyecto era SQLite debido a que presentaba ventajas tales como: operación de memoria, compatibilidad con diversos lenguajes y no requería de un proceso separado funcionando como servidor, ya que lee y escribe directamente sobre archivos que están almacenados en la memoria, lo que era optimo tanto para no consumir recursos del dispositivo, así como para no interferir con los procesos de visión artificial.

3.6 Diagrama de bloques general del sistema

En la Figura 20, se presenta un diagrama de bloques que ilustra la propuesta del proyecto. En este diagrama, cada proceso se separó en un bloque, considerando los componentes esenciales para el desarrollo del dispositivo basado en la placa mencionada en la sección anterior.

Figura 20

Diagrama de Bloques General del Sistema



Fuente: Autoría

El objetivo de este proyecto fue desarrollar un dispositivo que pudiera ser utilizado como alternativa de evaluación y permitiera detectar posibles anomalías estructurales en el

paciente, como lordosis o hiperlordosis, cifosis o hipercifosis, mediante el análisis de imágenes del paciente.

El dispositivo pasó por una fase de desarrollo en la que se dividieron en 4 subsistemas, que fueron integrados en uno solo para obtener un único sistema capaz de detectar el ángulo cifolumbar. Luego se procedió a la especificación del diseño.

3.7. Diseño del Sistema

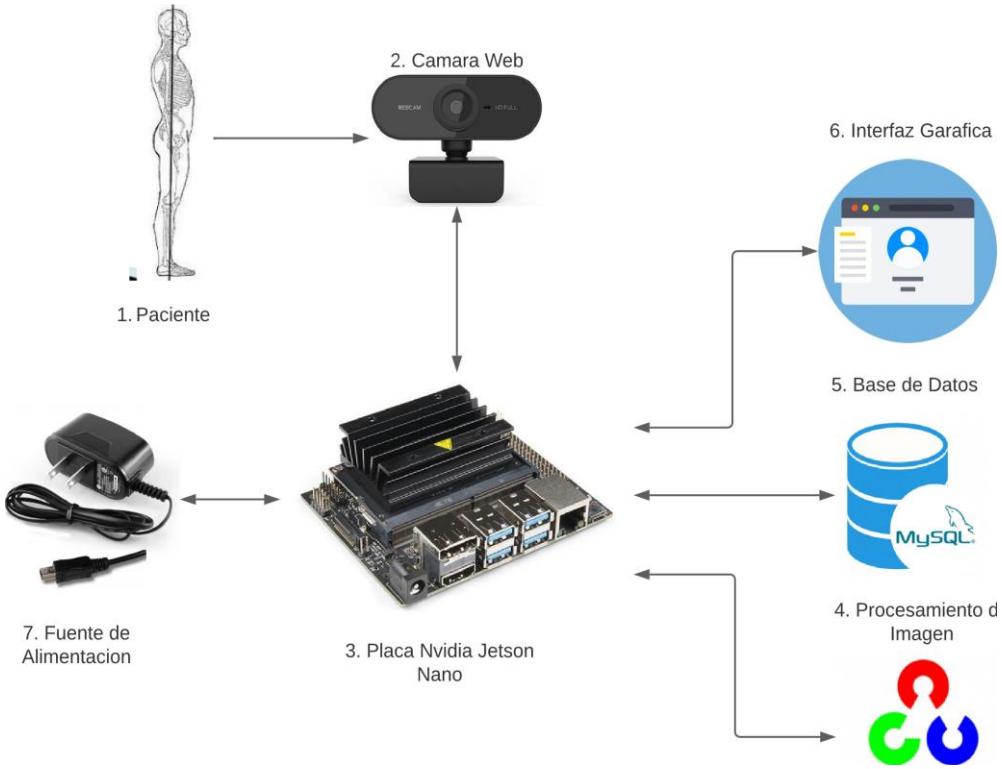
Después de analizar y seleccionar los requisitos en términos de hardware y software, considerando la estructura general del sistema, se procedió al diseño del sistema con sus componentes respectivos.

3.7.1. Arquitectura del Sistema

En la Figura 21, se presenta la arquitectura del sistema, que facilita la comprensión de la estructura y el funcionamiento de cada subsistema para obtener el ángulo e índice cifolumbar. El principal requisito para el desarrollo del sistema fue el uso de la placa Nvidia Jetson Nano. Para facilitar la comprensión del dispositivo, el diseño se subdividió en subprocesos, los cuales se explicarán de manera resumida y específica.

Figura 21

Arquitectura del Sistema



Fuente: Autoría

Sistema de Adquisición: El subsistema de adquisición estaba compuesto principalmente por una cámara web que se conectaba a la placa mediante un puerto USB, lo que permitía obtener las imágenes necesarias del paciente. Asimismo, se controlaba el entorno para eliminar factores que pudieran interferir con una buena adquisición de la imagen, como la baja luminosidad, el fondo de la imagen, objetos no deseados y la distancia del paciente a la cámara del dispositivo. Por lo tanto, se controló el entorno de pruebas lo más posible para minimizar la mayoría de los errores en la obtención de las imágenes. Una vez obtenida la imagen, se almacenaba en el dispositivo para su posterior uso por otros subsistemas.

Sistema de Procesamiento: Este subsistema dependía en gran medida de la capacidad de procesamiento de la placa. Se encargaba tanto del procesamiento de imágenes como de video del sistema. Después de adquirir las imágenes o videos, se creó un algoritmo en Python basado en el test de flechas sagitales para analizar la imagen, lo que permitía corregir errores y eliminar el ruido para obtener una imagen adecuada. Se identificaron características o regiones

de interés que ayudaron a determinar el ángulo y las distancias necesarias. Con los datos anteriores se realizaron procesos matemáticos para calcular el índice y ángulo cifolumbar, lo que ayudó a determinar la presencia de hipo/hiperlordosis o cifosis en la persona analizada.

Sistema de Almacenamiento: Este subsistema estaba compuesto por una base de datos local que fue elaborada en SQLite. Es decir, permitía el almacenamiento local de la información del paciente al momento de su registro. A través de una interfaz, se mostraba una ficha en formato digital en la que se pedía cierta información del paciente y se permitió registrar la información de manera controlada. También se permitió agregar los resultados obtenidos por los métodos de evaluación. De esta manera, se logró un control de la información de manera más rápida.

Sistema de Visualización: Este subsistema se trataba de la interfaz gráfica desarrollada en Python y sus librerías, que resultó en una interfaz intuitiva tanto para el fisioterapeuta como para el paciente. Permitía la visualización de la información obtenida por la evaluación, de tal manera que el fisioterapeuta pudiera visualizar de manera más sencilla la información obtenida por el sistema, incluyendo la ficha médica, las imágenes analizadas y los resultados obtenidos en la evaluación.

Tras establecer los subsistemas que serían tratados en el dispositivo, se diseñó el hardware, teniendo en cuenta los componentes y requerimientos solicitados en las secciones correspondientes. Se detallo el diseño a fin de asegurar que cumpliera con los requerimientos especificados.

3.8. Diseño de Hardware

Para el desarrollo del proyecto se utilizó como base principal la placa Nvidia Jetson Nano, la cual es específica para el desarrollo de aplicaciones en visión artificial. Gracias a sus características, permitió agilizar la adquisición y procesamiento de imágenes en poco tiempo.

Por lo tanto, dado que fue el punto principal del sistema, se describirá cómo funcionó cada parte del sistema.

Seguidamente, se inició el diseño del subsistema de adquisición, que fue crucial ya que posibilitó la adquisición de imágenes en un entorno controlado.

3.8.1 Subsistema de Adquisición

El objetivo de este subsistema era obtener una imagen en RGB mediante una cámara web conectada a la placa. Esto posibilitaba la captura de la imagen del paciente, y como resultado se obtenía una imagen en formato digital para almacenarla posteriormente. Una vez guardada la imagen en el dispositivo, se podía revisar o corregir en caso de ser necesario antes de ser utilizada en la siguiente fase. Es importante destacar que el tamaño de la imagen no debía ser muy grande para poder ser procesado por el dispositivo, pero la calidad de la imagen era un factor muy importante. Por lo tanto, la imagen debía ser clara para evitar fallos, lo cual requería un ambiente controlado que favoreciera la adquisición de la imagen.

Uno de los aspectos a considerar fue la correcta iluminación, ya que podía afectar la calidad de la imagen produciendo opacidad o reflejos. Además, el fondo utilizado tenía que ser diferente al de los marcadores, y se debía asegurar una adecuada distancia del paciente a la cámara para poder capturar todo el cuerpo del paciente sin que los marcadores de la espalda quedaran desenfocados. Con estos puntos en mente, se debía enfocar al paciente y en particular los marcadores colocados en cada región de la espalda. Es importante mencionar que el formato de la imagen no afectaba el procesamiento.

3.8.1.1 Diagrama de conexión

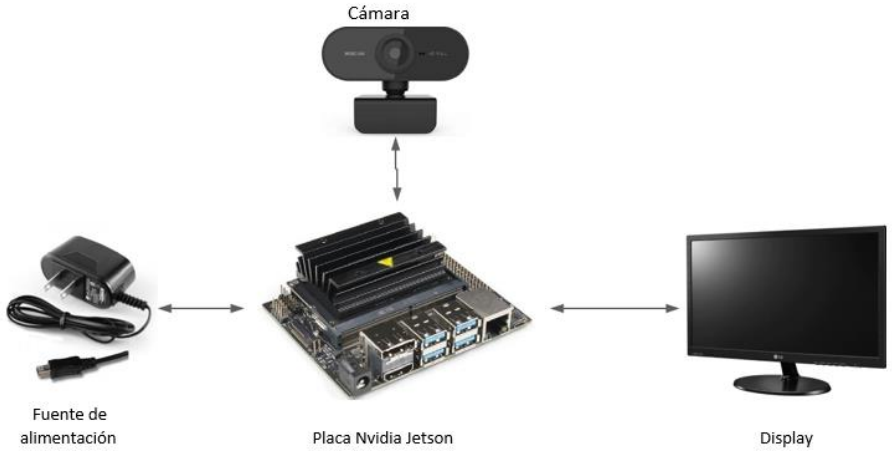
El subsistema de adquisición contaba con tres entradas importantes: cámara web, placa de visión artificial y el display de salida. En primer lugar, se estableció una conexión entre la cámara web y la placa mediante un cable USB. La cámara se colocó en un lugar fijo que permitía observar al paciente en el momento de capturar la imagen, pero que proporcionaba

movilidad en caso de ser necesaria. A su vez, la placa se conectó a una fuente de alimentación mediante un cargador de 5V a 4A para obtener la máxima capacidad de procesamiento de imágenes gracias a su configuración.

Es importante destacar que la placa base no tenía movilidad debido a la limitación de la conexión a la fuente de alimentación, lo cual restringía mucho el espacio donde se podía colocar. Por último, se conectó un monitor a la placa para poder visualizar tanto el funcionamiento del sistema como las interfaces del proyecto. La unión de estos tres elementos permitió la adquisición y visualización del sistema, como se muestra en la Figura 22.

Figura 22

Diagrama de Conexión



Fuente: Autoría

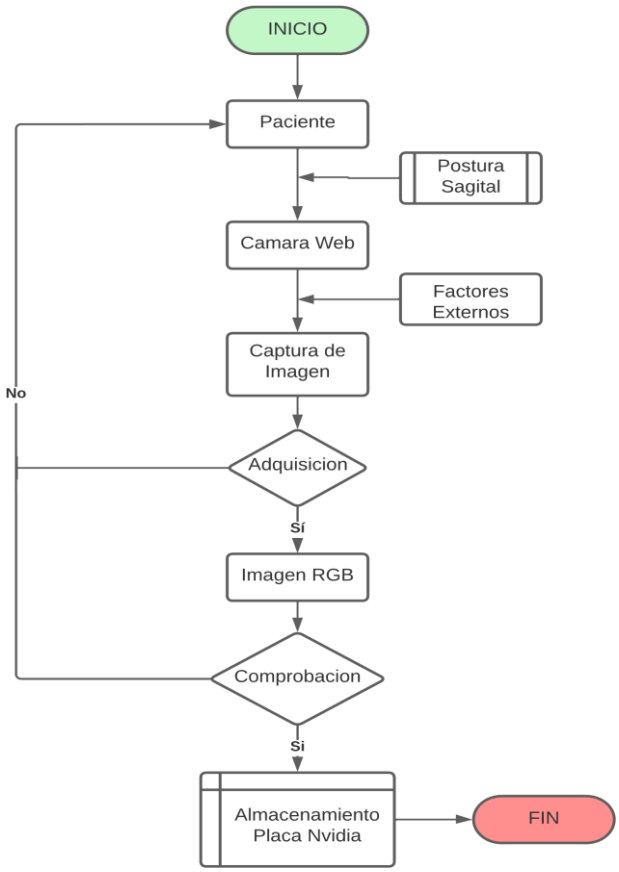
3.8.1.2 Diagrama de Flujo Subsistema de Adquisición

En el siguiente diagrama de flujo, que se muestra en la Figura 23, se ilustra el proceso de captura de la imagen llevado a cabo por el subsistema de adquisición. En la primera etapa, el paciente que será evaluado debía estar en bipedestación y de perfil a la cámara. Era importante considerar la distancia a la que se encontraba el paciente; se recomendaba que estuviera en un rango de 1.50 a 2 metros, aunque esta distancia podía variar dependiendo de la

altura de la persona para evitar problemas de enfoque y mala iluminación del entorno. Una vez que se cumplían estos parámetros, se tomaba la imagen con ciertas características de dimensión y resolución requeridas para ser guardada posteriormente en una carpeta con el nombre del paciente correspondiente.

Figura 23

Diagrama de Flujo Subsistema de Adquisición



Fuente: Autoría

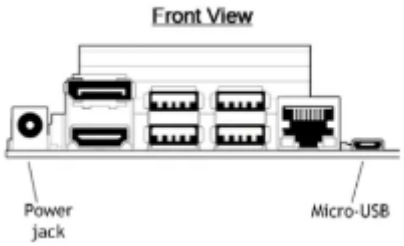
3.8.2 Alimentación del Sistema

En este apartado se describe el diseño del sistema de alimentación usado para suministrar corriente a la placa Nvidia Jetson Nano. Se tuvo en cuenta las dos opciones de alimentación disponibles en la placa: el puerto micro-USB, configurado para una alimentación

de 5V y 2.5A, y el puerto de 2.1mm "Power Jack", configurado para una alimentación de 5V y 4A. El diseño del sistema se muestra en la Figura 24.

Figura 24

Entradas de carga Nvidia Jetson Nano

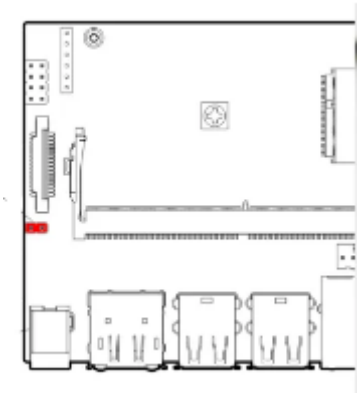


Fuente: https://stackdata.com/best-power-solution-for-your-nvidia-jetson-nano/

Al revisar las dos opciones anteriores y con el objetivo de aumentar la capacidad de procesamiento de la placa, se optó por colocar un puente en el conector J28 para activar el conector Barrel 5V 4A, tal y como se muestra en la Figura 25. Esta nueva configuración de alimentación posibilitaba el uso de fuentes de alimentación convencionales, como por ejemplo un cargador que contenga un conector jack DI 2.1mm.

Figura 25

Conector J28 Nvidia Jetson Nano



Fuente: https://stackdata.com/best-power-solution-for-your-nvidia-jetson-nano/

Una vez terminado el diseño del hardware y sus subsistemas, se procedió con el desarrollo del software según lo establecido en el apartado 3.7.1. En el siguiente apartado se explica de manera más detallada en qué se basa y qué función cumple cada subsistema.

3.9. Diseño de Software

En el apartado anterior se describió la elaboración del software para el dispositivo. Se tuvieron en cuenta tres subsistemas importantes, siendo uno de ellos el de procesamiento, encargado de analizar la imagen obtenida y pasarla por un proceso lógico matemático para obtener el índice y ángulo cifolumbar. En segundo lugar, se habló del subsistema de almacenamiento, el cual constó de una base de datos desarrollada en SQLite, lo que permitió el almacenamiento de información de cada paciente. Por último, el subsistema de visualización se estableció como una interfaz programada en Python, lo que permitió al fisioterapeuta registrar información y observar la evaluación del paciente.

3.9.1 Subsistema de Procesamiento

En lo que respecta al subsistema de procesamiento, este comprendió desde la captura de la imagen mediante la cámara hasta el algoritmo de evaluación. Para la adquisición de la imagen en la parte del software, se desarrolló un método de captura utilizando la biblioteca OpenCV. Esto permitió obtener una imagen del paciente en la cual se pudieran visualizar los puntos de interés en la columna vertebral para posteriormente pasar por el algoritmo de evaluación.

3.9.1.1 Evaluación

En lo referente al algoritmo de evaluación, este se realizó en el lenguaje de Python, el cual, debido a su compatibilidad con la biblioteca OpenCV, permitió procesar las imágenes. Antes de empezar con el desarrollo del algoritmo de evaluación, se debió tener en cuenta la imagen de entrada en la que se visualizaría al paciente en posición sagital. Otro factor importante fue la visibilidad de los marcadores colocados, ya que estos constituyeron los

puntos de referencia en el cuerpo para la obtención de los ángulos y distancias a calcular. Para iniciar con el algoritmo de evaluación se necesitó que la cámara pasara por un proceso de calibración, el cual eliminó los efectos de distorsión y permitió que la imagen fuera adecuada para usarla posteriormente.

Durante la adquisición de la imagen fue importante tener en cuenta la estructura de la cámara web, ya que muchas de ellas tienen un lente de ojo de pez, el cual produce una distorsión visual que captura las imágenes de forma panorámica o hemisférica amplia. Sin embargo, dicha distorsión afecta la medición de las distancias y puede hacer que se obtengan datos erróneos. Para corregir este problema se desarrolló un algoritmo usando la biblioteca OpenCV que nos ayudó a calcular los parámetros intrínsecos y extrínsecos de la cámara. Los parámetros intrínsecos describen las características de la cámara, como la distancia focal, el punto principal y los coeficientes de distorsión; mientras que los parámetros extrínsecos describen la posición y orientación de la cámara en el espacio.

A continuación, se describen los pasos generales para calibrar la cámara:

1. Como punto de partida se tomaron un total de 20 imágenes al patrón de calibración "Tablero de ajedrez". Para ello, se utilizó una cámara web y se aseguró que las imágenes tuvieran diferentes ángulos y posiciones, siempre teniendo en cuenta que el patrón de calibración fuera visible en cada una de ellas.
2. Después de obtener el conjunto de imágenes, se seleccionaron 12 de ellas sin repetición y se utilizaron para detectar los puntos del patrón de calibración en las imágenes seleccionadas. Para la detección de los puntos del patrón de ajedrez se utilizó la función cv2.findChessboardCorners() de la biblioteca de OpenCV.
3. En tercer lugar, se utilizaron las funciones de OpenCV para estimar los parámetros intrínsecos de la cámara, tales como la distancia focal y el punto

principal. Para esto, se empleó la función cv2.calibrateCamera() que permitió el cálculo de dichos parámetros a partir de las imágenes y los puntos detectados en ellas.

4. Una vez que se calcularon los parámetros intrínsecos de la cámara, se utilizó la función cv2.undistort() para corregir la distorsión de las imágenes en base a los parámetros calculados. Esta función tomó como entrada una imagen y los parámetros intrínsecos de la cámara; y devolvió una versión corregida de la imagen sin distorsión.
5. Después de calcular los parámetros intrínsecos de la cámara y corregir la distorsión de las imágenes, se procedió a estimar los parámetros extrínsecos utilizando con funciones de OpenCV. Para ello, se utilizó cv2.solvePnP(). Esta función toma como entrada los puntos del patrón de calibración detectados en una sola imagen y sus coordenadas en el mundo real, así como los parámetros intrínsecos de la cámara, y devuelve los parámetros extrínsecos de la cámara que describen su posición y orientación en el espacio.

Con los pasos antes mencionados, se establece un algoritmo de calibración que, mediante la captura de imágenes de un patrón, permite detectar los puntos del patrón de calibración. De esta manera, se logra estimar los parámetros intrínsecos de la cámara, corregir la distorsión de las imágenes y estimar los parámetros extrínsecos de la cámara. Los parámetros intrínsecos son utilizados para eliminar el efecto de ojo de pez en las imágenes, lo que permite utilizarlas en los siguientes métodos.

En la primera parte de los métodos de evaluación, nos enfocamos en medir los ángulos en la columna vertebral utilizando puntos específicos como C7, el punto más sobresaliente, y T12 para formar el ángulo cifótico, y entre los puntos T12, el trocante mayor y, por último, la espina iliaca anterosuperior para formar el ángulo lumbar. Para medir los ángulos, se usó un

método para formar un triángulo utilizando estos tres puntos, los cuales tienen coordenadas rectangulares (X, Y). Una vez obtenidos los tres puntos, se calculó la diferencia entre los puntos extremos y el vértice. Luego, utilizamos la función math. atan2() para obtener el ángulo entre los dos vectores en radianes. Esta función toma los componentes (X, Y) del vector como argumentos y devuelve el ángulo en radianes.

En cuanto al cálculo de las distancias, se tuvieron en cuenta tres puntos para el desarrollo del algoritmo: la extracción de la región de interés de la imagen, la detección de los marcadores y el cálculo de la distancia entre el marcador y la imagen. Para extraer la región de interés, se utilizó la transformada de perspectiva con la función ROI. Esta última permitió obtener la región de interés de la imagen principal, generando una nueva ventana de la región seleccionada para su análisis. La función ROI solucionó el problema de los ángulos presentes en la imagen, los cuales pueden modificar la distancia medida. Además, la función ROI necesitó la relación de aspecto para establecer la distancia en píxeles.

Con lo anterior, se obtuvo la imagen a analizar que se utilizó para detectar los marcadores mediante la función de Hough. En esta función se definió el radio y el color del marcador a detectar. Finalmente, con los marcadores detectados, se procedió a calcular la distancia con respecto al eje Y. A continuación, se procedió a explicar el diagrama de flujo de este subsistema.

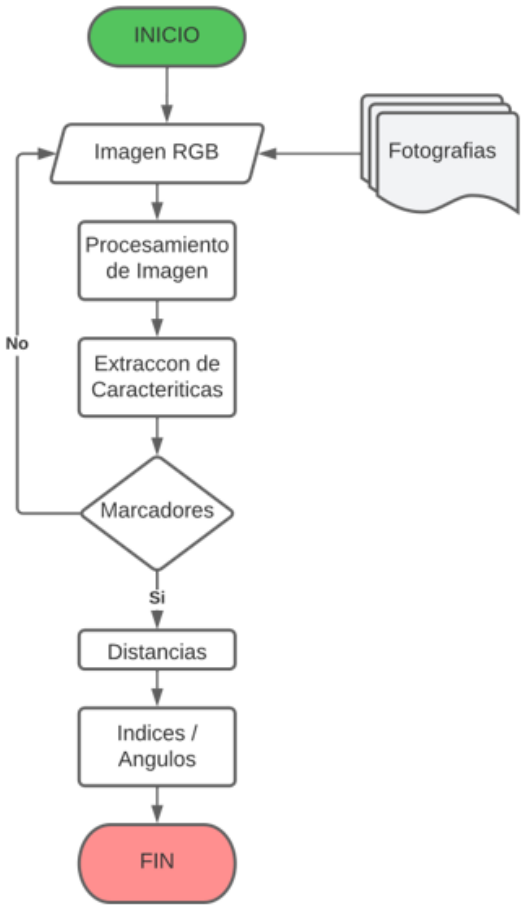
3.9.1.2 Diagrama de Flujo Proceso de Evaluación

En la Figura 26, se presenta un diagrama de flujo que muestra el proceso de evaluación. En primer lugar, se tomó la imagen del paciente, la cual fue sometida a un tratamiento mediante los algoritmos de OpenCV. La imagen RGB se convirtió a escala de grises y se eliminaron los ruidos mediante un filtro para mejorar su calidad y nitidez. A continuación, se procedió a segmentar la imagen y a seleccionar los puntos de interés, que en este caso correspondieron a los marcadores ubicados en la espalda del paciente. Una vez obtenidos estos puntos de

referencia, el algoritmo calculó las distancias de las flechas sagitales y se sometió a un proceso matemático para obtener el índice cifolumbar. Además, se utilizó otra imagen para calcular el ángulo del mismo nombre.

Figura 26

Diagrama de flujo evaluación



Fuente: Autoría

3.9.2 Subsistema de Almacenamiento

El subsistema de almacenamiento fue desarrollado utilizando la librería de Python sqlite3 con el propósito de obtener un sistema de gestión de bases de datos que permitiera almacenar y manipular la información relacionada con el registro, evaluación y resultados de cada paciente de forma adecuada y óptima. Esto posibilitó realizar un análisis detallado de la

situación actual del paciente y guardar la información de manera local para una administración eficiente y rápida por parte del encargado del sistema.

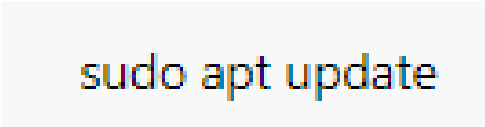
3.9.2.1 Creación Base de Datos

En esta sección se explicó el proceso de creación de la base de datos local. Se empleó la librería SQLite de Python como punto de partida para crear un sistema de gestión de bases de datos que permitiera almacenar y manipular la información relacionada con el registro, evaluación y resultados de los pacientes. SQLite es una base de datos de software libre reconocida por su sencillez, eficacia, potencia y rapidez, siendo ideal para ser utilizada en la Placa Nvidia. Es importante destacar que esta librería de base de datos se puede llamar a través de un script programado en Python.

Para descargar los paquetes de SQLite, se ejecutó un comando específico que permite instalar la librería. Se comenzó por actualizar los repositorios de la Nvidia Jetson Nano, para lo cual se introdujeron las siguientes líneas de comando en el terminal, tal como se muestra en la Figura 27.

Figura 27

Comando para actualizar repositorio

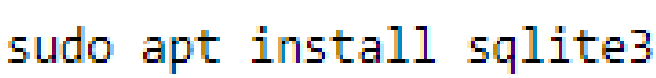


Fuente: https://ubunlog.com/sqlite-3-y-sqlitebrowser-como-instalarlos-en-ubuntu/

Llegado a este punto, se procedió a instalar el paquete SQLite3. Para ello, se ejecutó el siguiente comando en la terminal, como se muestra en la Figura 28.

Figura 28

Comando instalar paquetes de sqlite3



Fuente: https://ubunlog.com/sqlite-3-y-sqlitebrowser-como-instalarlos-en-ubuntu/

En consecuencia, al tratarse de una librería, esta no contaba con una interfaz gráfica. Por lo tanto, se procedió a instalar DB Browser, una herramienta visual de código abierto que permitió la creación, diseño y edición de archivos de bases de datos compatibles con SQLite. Para instalar esta interfaz gráfica, se siguió el mismo proceso de ejecutar la línea de comandos en la terminal, tal como se muestra en la Figura 29.

Figura 29

Comando para instalar entorno grafico sqlite3

```
sudo apt install sqlitebrowser
```

Fuente: https://ubunlog.com/sqlite-3-y-sqlitebrowser-como-instalarlos-en-ubuntu/

Después de instalar DB Browser, se accedió a esta herramienta visual de código abierto donde se visualizó un icono de base de datos, como se muestra en la Figura 30, Posteriormente, se procedió con la ejecución del programa.

Figura 30

Verificación de instalación de sqlite3

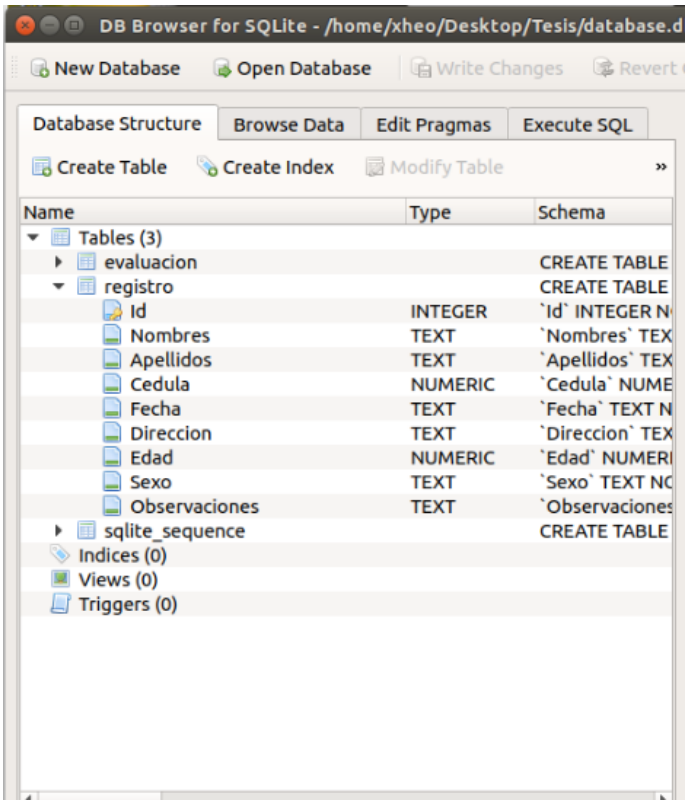
Fuente: Autoría

l iniciar DB Browser, se abrió una interfaz gráfica que permitió la creación de una base de datos de manera rápida mediante la creación de tablas con sus correspondientes propiedades y parámetros, como integer, numeric, text y blob (este último se usó para imágenes). De esta

manera, se agregaron ciertos campos para que la información fuera almacenada de forma estructurada y organizada, lo que resultó en un registro más detallado, tal como se muestra en la Figura 31.

Figura 31

Interfaz gráfica SQLite



Fuente: Autoría

Después de haber creado la base de datos mediante DB Browser, se importó la librería al script principal. De esta forma, se pudieron importar los paquetes necesarios para el uso de sqlite3 y establecer una conexión entre la interfaz y la base de datos. Para lograr esto, se desarrolló un método programado, que se muestra en la Figura 32.

Figura 32

Código de conexión a la BBDD

```
# Function to Execute Database Querys
def run_query(self, query, parameters = ()):
    with sqlite3.connect(self.db_name) as conn:
        cursor = conn.cursor()
        result = cursor.execute(query, parameters)
        conn.commit()
    return result
```

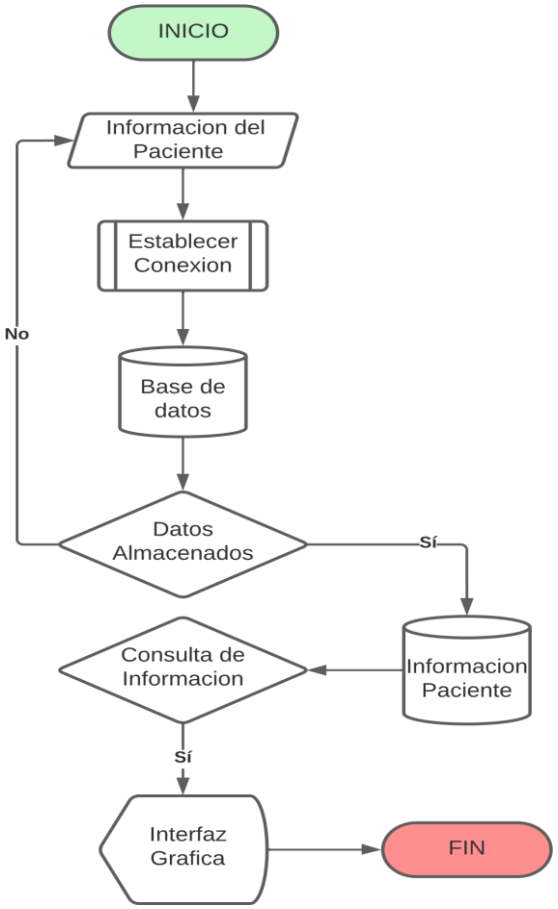
Fuente: Autoría

3.9.2.2 Diagrama de Flujo Subsistema de Almacenamiento

En la Figura 33, se muestra el diagrama de flujo del subsistema de almacenamiento, el cual describe de manera general su funcionamiento. La primera etapa consistió en la recopilación de información del paciente, incluyendo su registro y evaluación. Por medio de un script, se estableció la conexión entre la interfaz y la base de datos correspondiente. Si la conexión se estableció correctamente, se permitió guardar la información en nuestra tabla, lo que permitió llevar un registro detallado de cada paciente. Consecuentemente, esta información se utilizó para llevar un registro de las consultas, al que el fisioterapeuta podía acceder mediante la interfaz del sistema.

Figura 33

Diagrama de flujo Subsistema de Almacenamiento



Fuente: Autoría

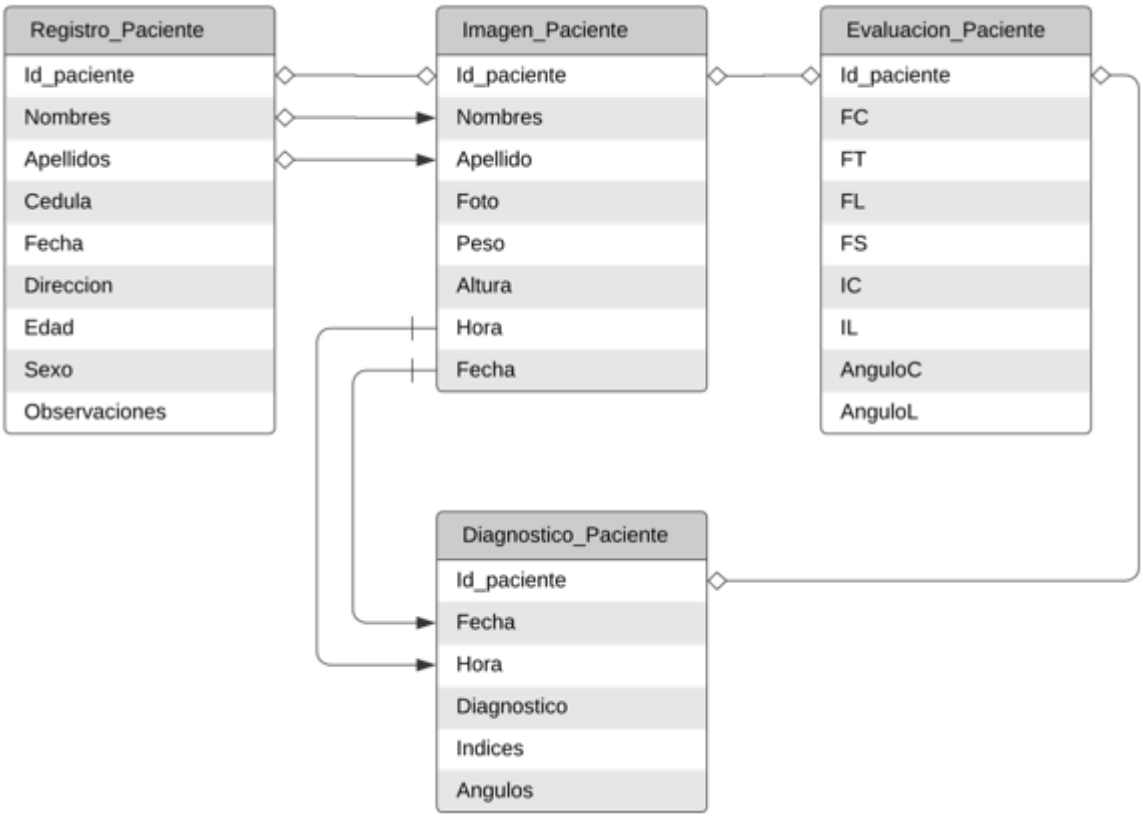
Después de haber concluido la explicación del diagrama anterior que estableció el proceso requerido para el registro y almacenamiento de la información en la base de, se llevó a cabo el diseño correspondiente de las tablas. Como punto de partida para el diseño digital de estas, se utilizó el formato de una ficha manual. En la siguiente sección, se explicará el modelo de relación de la base de datos.

3.9.2.3 Modelo relacional de la Base de datos

Para el control y administración de la información que fue ingresada tanto de pacientes como de evaluaciones, se creó una base de datos utilizando SQLite y DBrowser. Esta base de datos contiene cuatro tablas, tal como se puede observar en la Figura 34, lo cual permite la registración, recuperación y manipulación de la información de las imágenes analizadas y de los registros previamente realizados en el sistema.

Figura 34

Modelo de Relación BBDD



Fuente: Autoría

En cuanto al desarrollo, se creó una tabla llamada "Registro_Paciente" donde se registraron todos los datos relevantes de la persona analizada, tales como nombre, apellido, edad, sexo, entre otros. Cabe destacar que existe una relación entre la tabla "Registro_Paciente" y las demás tablas a través del identificador (id) del paciente. La tabla "Imagen_Paciente" contiene el nombre y apellido de la tabla de registro, así como el identificador del paciente y los campos de peso, altura, entre otros. Por otro lado, en la tabla "Evaluacion_Paciente" se registra el identificador del paciente y se agregan las distancias obtenidas de cada punto ("Flechas"), así como los resultados de la evaluación, tales como índices y ángulos. Para finalizar, en la tabla "Diagnostico_Paciente" se puede observar el identificador y el diagnóstico general proporcionado por el fisioterapeuta.

Luego de haber concluido la explicación de la creación y manejo de la base de datos, se procede en la siguiente sección a detallar el sistema de visualización que permitió mostrar los datos a través de una interfaz gráfica. En el siguiente apartado se entró en detalle sobre este subsistema.

3.9.3 Subsistema de Visualización

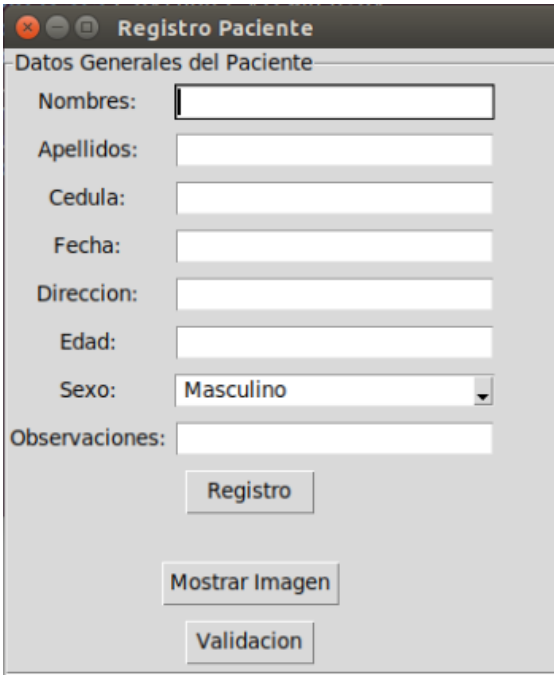
En la sección siguiente se describe el desarrollo de la interfaz de usuario. Para ello, se utilizó la librería de Tkinter, que es ofrecida por Python para el desarrollo de interfaces. Dicha librería contiene un conjunto de herramientas intuitivas y fáciles de entender en la programación, lo que permitió desarrollar una GUI raíz. Este GUI tenía acceso a cada función implementada, en línea con el diseño del sistema, como el registro del paciente, la visualización de la base de datos, el sistema de evaluación y otros. La ejecución del programa se permitió a través de un script en el intérprete de Python, y el código puede ser visualizado en el ANEXO C. Script Interfaz Principal

3.9.3.1 Registro de Paciente

En la opción de la interfaz correspondiente, se desplegó una ventana en forma de ficha médica que permitía el ingreso de la información del paciente perteneciente al área de fisioterapia, similar a una ficha médica. Esta información incluía nombres, apellidos, cédula, sexo, fecha, edad, dirección y observaciones, como se observa en la Figura 35**Figura *35*** Registro del Paciente. Estos campos son llenados de manera manual por el fisioterapeuta.

Figura 35

Registro del Paciente



Fuente: Autoría

Una vez que la información estaba completa y verificada, se seleccionaba la opción de Registro, la cual guardaba los datos de la ficha en una base de datos perteneciente a SQLite. Como resultado, se mostraba un mensaje que verificaba si la información se registraba correctamente o si había un error en el proceso de registro. El código correspondiente puede visualizarse en el ANEXO F. Script Registro Pacientes

3.9.3.2 Toma de Datos

En esta opción, el uso de una cámara era imprescindible, ya que permitía capturar la imagen del paciente para su posterior procesamiento en la opción de evaluación. En la sección correspondiente de la interfaz, como se muestra en la Figura 36, era posible verificar si la cámara funcionaba correctamente y preparar al paciente para la adecuada toma de la imagen.

Figura 36

Interfaz de adquisición de imágenes

Fuente: Autoría

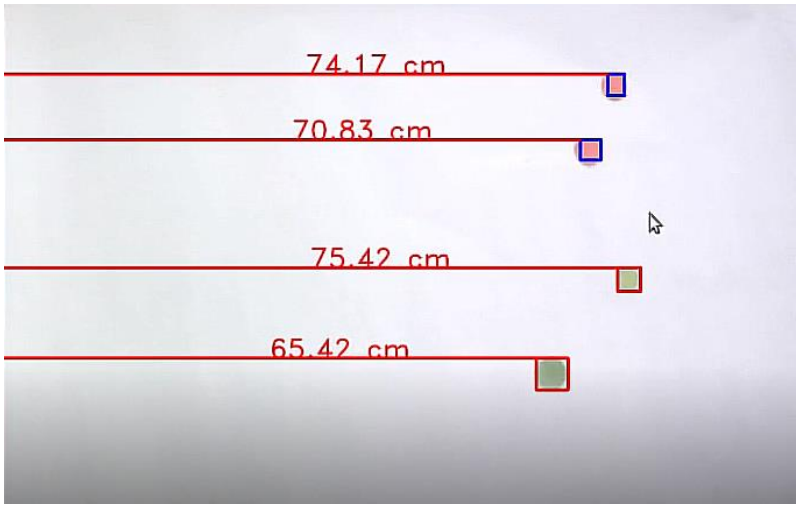
Esta sección registró los datos del paciente, incluyendo su nombre, apellido y un identificador (ID), con el objetivo de archivar la imagen junto con la información del paciente en una base de datos utilizando la función de guardar. En este sentido, se ingresaron los datos pertinentes para garantizar un registro completo y preciso de la información del paciente

3.9.3.2 Evaluación

En esta sección se presentó la opción de evaluar la imagen adquirida por el sistema. Se mostró en la interfaz la imagen del paciente con información importante, como la distancia de cada flecha sagital. Además, se presentaron los resultados de los índices cifótico y lumbar, junto con la información relacionada al ángulo cifótico y lumbar. Todo esto se hizo para garantizar que se tenga una visión completa y detallada de los datos del paciente.

Figura 37

Obtención de distancias



Fuente: Autoría

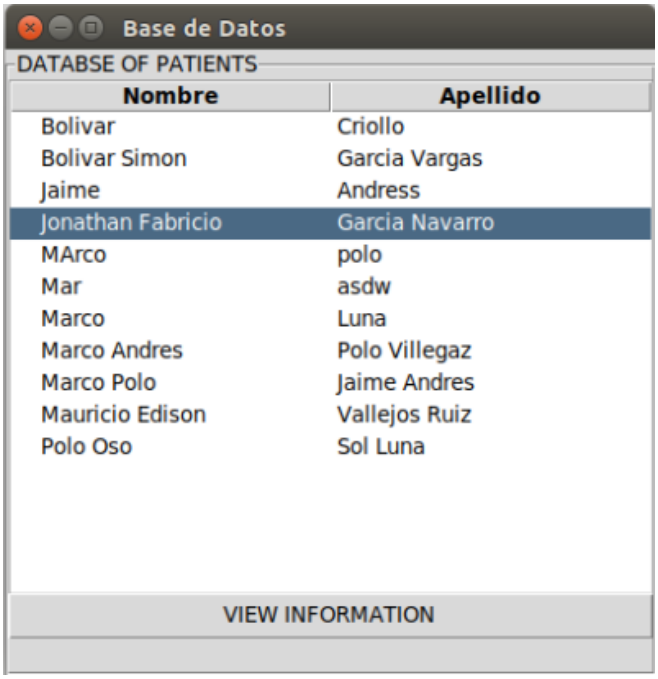
Como se visualizó en la Figura 37, se presentaron las 4 distancias de los marcadores hacia el eje Y mediante el método de evaluación. Esto permitió que los datos obtenidos se utilizaran para calcular el índice cifolumbar. El código correspondiente se encuentra en el ANEXO E. Medición Flechas Sagital

3.9.3.3 Visualización de Base de Datos de Pacientes

En esta sección se mostró la base de datos previamente establecida en la sección 3.7.1. La información de cada paciente fue visualizada de manera simplificada, con solo los campos de Nombre y Apellido, como puede observarse en la Figura 38**Figura *38*** Interfaz, Todo esto se hizo para garantizar una presentación más clara y ordenada de los datos de los pacientes.

Figura 38

Interfaz datos del paciente

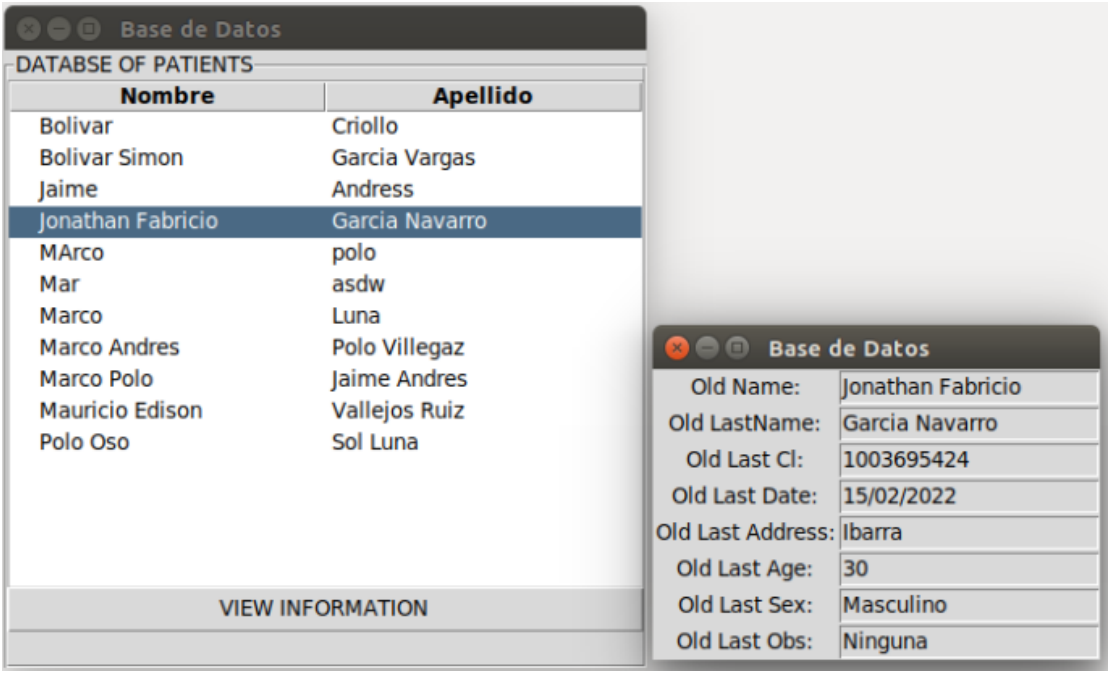


Fuente: Autoría

En la interfaz, la opción "Visualizar Información" permitía abrir una ventana emergente que presentaba toda la información del paciente almacenada en la base de datos, como se puede observar en la Figura 39**Figura *39*** Visualización de información del paciente. Esta opción se diseñó con el fin de que se pudiera acceder fácilmente a la información de cada paciente.

Figura 39

Visualización de información del paciente



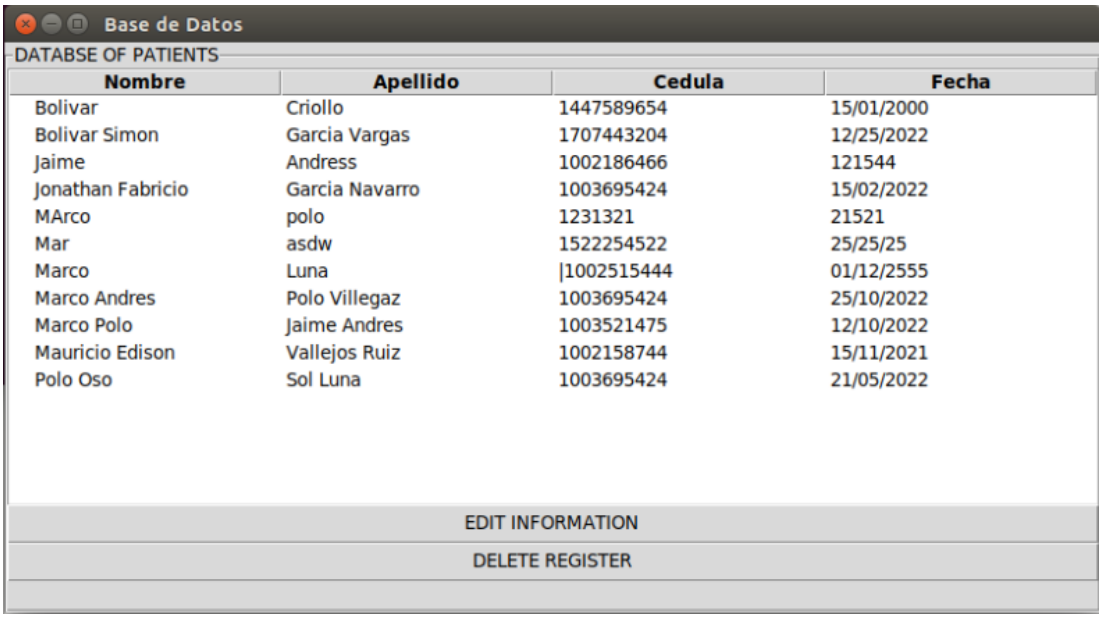
Fuente: Autoría

3.9.3.4 Modificación de Base de Datos

En esta sección se utilizaron determinadas variables de la sección anterior que permitieron la observación y modificación directa de la base de datos. De esta manera, se logró la capacidad de eliminar o modificar registros, tal como se observa en la Figura 40.

Figura 40

Interfaz de modificación de Base de Datos.

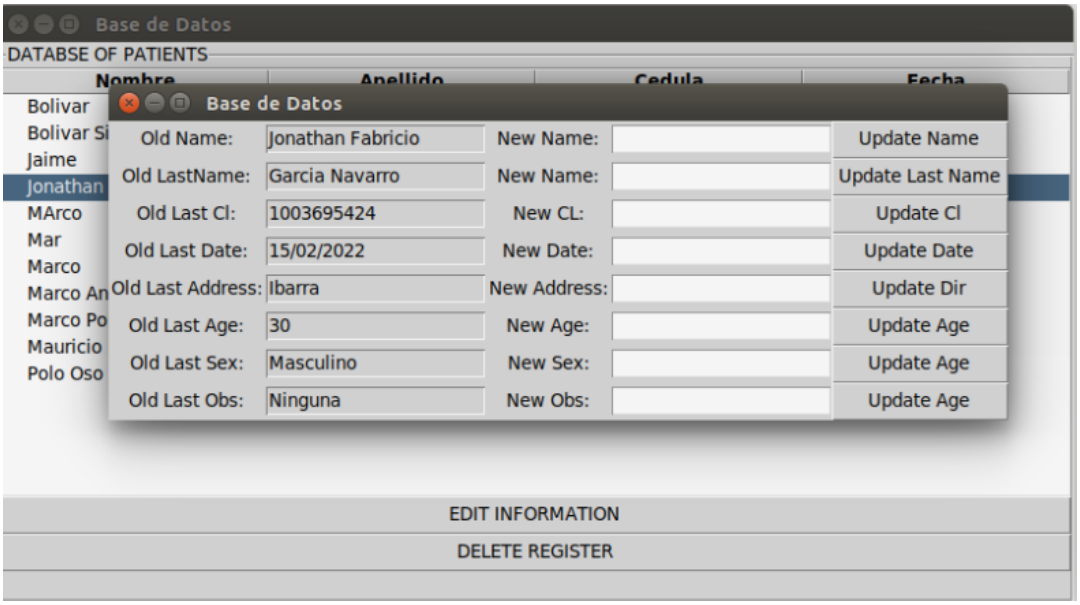


Fuente: Autoría

En la opción de modificación se brindó la posibilidad de editar la información almacenada en la base de datos, generando una nueva ventana que presentaba la información previa y los campos para ser modificados con la información actualizada. Esto se hizo para ofrecer una gestión eficiente y precisa de la información guardada

Figura 41

Interfaz de edición de Datos.



Fuente: Autoría

Una vez concluido el diseño del sistema propuesto en la sección 3.7, se procedió a su implementación y prueba, constatando que el dispositivo (tanto en la adquisición de datos, evaluación, interfaz gráfica y base de datos) se combinaba adecuadamente para la evaluación del ángulo cifolumbar en la región espinal. Se llevaron a cabo pruebas de campo para determinar el rendimiento y la facilidad de uso del sistema por parte del usuario, verificando que los datos recogidos fueran coherentes y en tiempo real. Estos resultados se compararon con los datos manuales para determinar el porcentaje de error y validar si el dispositivo desarrollado era una herramienta viable para la evaluación de las patologías mencionadas en la sección 2.1.3. En el siguiente capítulo se procedió al desarrollo de las fases 3, 4 y 5.

Capítulo 4. Implementación y Pruebas de Funcionamiento

En el capítulo presente se describió la fase de validación e implementación de cada subsistema tratado anteriormente, con el fin de unirlos en un solo sistema. Para cumplir con estas últimas fases, se estableció un cronograma de pruebas con el objetivo de obtener y comparar los datos obtenidos por el sistema con los datos ya existentes, comprobando así la funcionalidad y viabilidad del dispositivo y determinando si el sistema logró o no cumplir con los objetivos propuestos al inicio del proyecto.

Una vez finalizado el diseño del dispositivo propuesto en la sección 3.7, se procedió con su implementación y verificación, tal como se explica en la fase 3, donde se llevó a cabo la revisión individual de cada componente.

4.1. Fase 3: Implementación del Sistema

En este apartado se llevó a cabo el análisis con sus respectivas pruebas para validar el sistema. Se realizaron verificaciones individuales para cada componente a fin de comprobar el funcionamiento y en algunos casos, la adquisición de datos. El objetivo era verificar el correcto registro digital, la base de datos y los métodos de evaluación del sistema, y comparar los resultados con la forma manual de registro. En el siguiente apartado, se efectuaron pruebas en el dispositivo tanto en la parte del hardware como del software.

4.1.1 Prueba 1: Verificación del Hardware y Entorno de trabajo

Una vez se finalizó y adaptó el entorno de trabajo para el sistema, se verificaron ciertas características del ambiente de adquisición. Resultó necesario que el espacio estuviera diseñado con los parámetros correctos, tales como la iluminación, el fondo, la distancia de la cámara al paciente, la señalización, entre otros aspectos, como se exhibe en la Figura 42. De esta forma, se estableció un entorno de adquisición óptimo. Es importante destacar que el área de trabajo adecuada tanto para el paciente, el fisioterapeuta como para el sistema, fue mostrada.

Figura 42

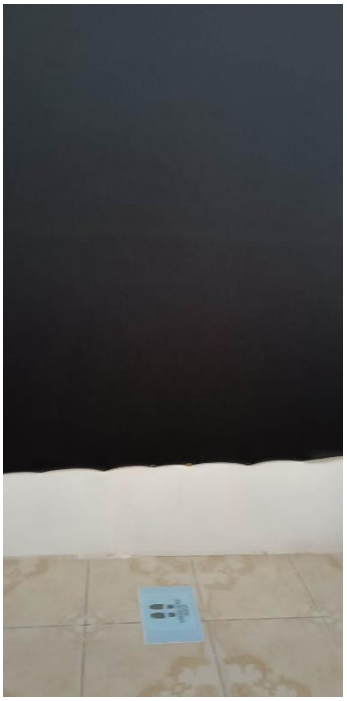
Entorno de adquisición.

Fuente: Autoría

En la parte del hardware se verificó la ubicación de los componentes en el entorno. Se aseguró de que la cámara del sistema estuviera en un lugar estático que pudiera enfocar al paciente y evitar el efecto a contraluz que pudiera afectar la imagen resultando oscura o borrosa. Con la correcta iluminación, se lograron distinguir los marcadores. Por esta razón, se colocaron algunos indicadores para servir como guía de distancia y la foto a tomar. Gracias a las recomendaciones antes mencionadas, se logró implementar el sistema en el área de trabajo, tal como se muestra en la siguiente Figura 43. De esta forma, se completó el sistema en la parte de entorno y dispositivos, permitiendo realizar las diferentes pruebas.

Figura 43

Entorno de Adquisición.



Fuente: Autoría

Con respecto a los marcadores a utilizar, se consideró necesario que éstos tuvieran dos características específicas. En primer lugar, debían tener un color distinto al entorno para evitar confusiones en el sistema al procesar la imagen. En segundo lugar, debían ser identificables, en este caso debían ser circulares, para que el sistema pudiera reconocerlos rápidamente. En cuanto a su ubicación, el fisioterapeuta los colocó en las zonas que habían sido previamente establecidas para el cálculo de distancias y ángulos, respectivamente. Como resultado final, se llevó a cabo la captura de la imagen que debía incluir la región de la espalda con los marcadores utilizados, los cuales eran claramente visibles para el dispositivo. Se muestra un ejemplo de los marcadores a utilizar a continuación.

Figura 44

Marcadores a utilizar.

Fuente: Autoría

En cuanto a la colocación de los marcadores, se tuvieron en cuenta ciertos factores que podían ocasionar problemas, como la contextura del paciente, el IMC y la fisiología de la columna vertebral. En consecuencia, se consideró necesario que los marcadores fueran visibles para el sistema, por lo que se implementó una extensión para que sobresalieran y pudieran ser detectados fácilmente, como se exhibe en la Figura 45. Tanto para la medición de los índices como de los ángulos.

Figura 45

Marcadores de ángulo cifolumbar

Fuente: Autoría

Como se visualizó en la imagen anterior, los marcadores fueron colocados en la región de la espalda en puntos específicos para cada evaluación. Para medir el ángulo cifótico, los puntos fueron dispuestos en el proceso espinoso C7 y el proceso espinoso de T12, teniendo en cuenta el punto sobresaliente entre esta región. Para el siguiente, los puntos se situaron en el proceso espinoso de T12, el trocánter mayor y la espina ilíaca superior.

Figura 46

Marcadores Flecha Sagital

Fuente: Autoría

Como se observa en la imagen anterior, se colocaron los marcadores de manera prominente para el cálculo de los índices. Dado que en el método de la flecha sagital el paciente debía estar en posición lateral y los marcadores debían sobresalir del cuerpo independientemente de la fisonomía del paciente, se implementaron unos marcadores que sobresalieran del cuerpo para poder ser visibles para el sistema. En el caso de la contextura del paciente, los marcadores podían o no ser visibles para el sistema. Como se mencionó anteriormente, se optó por realizar unos marcadores con mayor prominencia.

Para el método de las flechas sagitales, los marcadores se ubicaron en cuatro regiones diferentes: el primer punto en C7, el segundo en el punto más destacado entre C7 y T12, el tercer marcador en la parte menos destacada entre T12 y S1, y el último marcador en el punto

de S1. Es importante destacar que se recomendaba que el paciente llevara ropa negra para mayor comodidad y evitar confusiones con los marcadores.

Una vez establecidos los puntos anteriores, se dio por finalizada la fase de comprobación del entorno. Con los parámetros y especificaciones adecuados, se obtuvo una imagen apropiada para el sistema.

Una vez concluida la validación del hardware y entorno físico, se procedió a la validación del software mediante la explicación y realización de pruebas en cada subsistema: adquisición, pre-procesamiento, evaluación y registro. El objetivo era verificar el cumplimiento y funcionamiento de cada subsistema. Es importante mencionar que las pruebas se llevaron a cabo en un ambiente controlado, supervisadas por un fisioterapeuta. Además, se realizó una comparación entre la evaluación manual y el método de evaluación del dispositivo. Para realizar las pruebas, se elaboró un cronograma que indicaba los momentos en que se llevarían a cabo cada prueba del sistema. En la Tabla 14, se puede observar el cronograma de pruebas.

Tabla 14

Cronograma de pruebas

CRONOGRAMA DE PRUEBAS				
TIPO DE PRUEBA	**LUGAR PARA DESARROLLAR**	**RESULTADO ESPERADO**	**DURACIÓN**	**OBSERVACIONES**
Prueba 1.- Verificar el entorno de adquisición de datos	Centro de fisioterapia de la Clínica MediFisio	Se espera que el entorno se encuentre con una iluminación y espacio adecuado para la adquisición de imágenes evitando los factores que puedan bajar la calidad de esta	Del 6 de febrero hasta el 10 de febrero	
Prueba 2.- Verificar los métodos de evaluación establecidos	Centro de fisioterapia de la Clínica MediFisio	Se espera que el sistema y los métodos programados cumplan con sus objetivos midiendo así los índices y ángulos cifolumbar del paciente	Del 13 de febrero hasta el 17 de febrero	
Prueba 3.- Verificación de la interfaz y registro de los pacientes	Centro de fisioterapia de la Clínica MediFisio	Se espera que nuestra interfaz tanto en el registro, consultas o datos de la evaluación funcionen correctamente pudiendo ser guardados en nuestra base de datos local	Del 20 de febrero hasta el 24 de febrero	
Prueba 4.- Verificación final del dispositivo y pruebas	Centro de fisioterapia de la Clínica MediFisio	Se espero que el sistema funcione completamente para la realización de las pruebas finales con 10 pacientes del centro de fisioterapia, realizando las pruebas tanto manuales del fisioterapeuta y las pruebas del sistema esperando que el sistema tenga un índice de error bajo para que sea viable.	Del 27 de febrero hasta el 3 de marzo	Se realizará con los pacientes disponibles en el centro de fisioterapia

4.1.2 Prueba 2: Verificación del Software Sistema

En esta sección se verificó el funcionamiento del software mediante la explicación y la realización de pruebas en cada uno de los cuatro subsistemas: el primero era el de adquisición, que se encargaba de capturar la imagen; el segundo era el de registro, que obtenía información del paciente como su nombre, apellidos, edad y otros datos relevantes, funcionando como una ficha de registro digital; el tercero era el de evaluación, que procesaba la imagen para proporcionar los resultados del test; y el último era el de base de datos, que almacenaba la información y evaluación del paciente para futuras consultas.

4.1.2.1 Verificación de Adquisición de imágenes

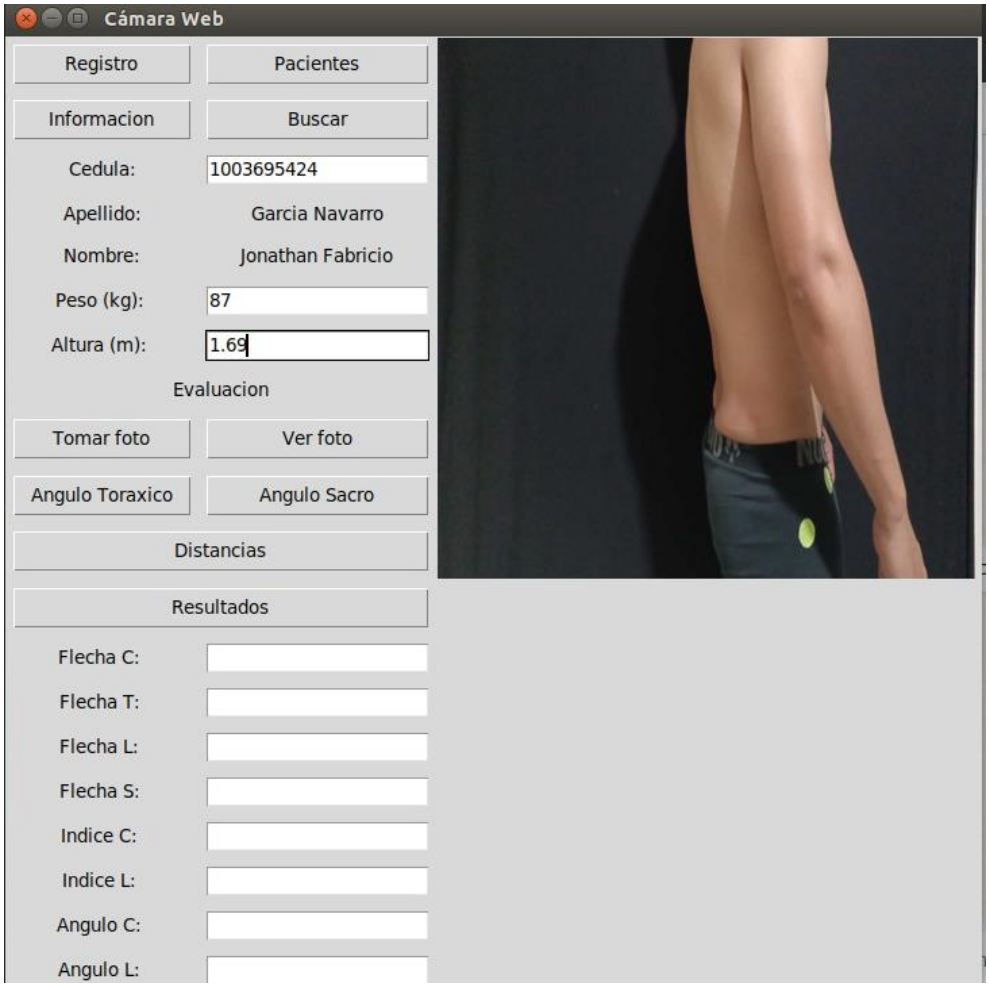
esta parte, las imágenes se obtenían utilizando la cámara y un código en Python. Para comenzar, se conectaba la cámara al dispositivo y este componente era prioritario para iniciar el proceso. A continuación, se ejecutaba el sistema para verificar la conexión con la cámara, de modo que permitiera capturar o corregir la imagen.

En resumen, como se muestra en la Figura 47, se realizó el llenado del campo "Cedula" requerido para buscar en la base de datos y obtener ciertos campos que se utilizarían para dar un nombre a la imagen. Además, se disponía de un botón "Tomar Foto" en la pantalla para obtener la imagen.

La verificación del subsistema se llevó a cabo mediante la ejecución del script para la captura y visualización de imágenes, de modo que se pudiera ver la imagen obtenida a través de un monitor. Como resultado, se obtuvo la imagen y se guardó en una ubicación específica para utilizarla en los procesos posteriores.

Figura 47

Interfaz de Adquisición



Fuente: Autoría

Para comenzar la prueba, se explicaron ciertos fragmentos del script que ayudarían a comprender y verificar mejor el funcionamiento del mismo. Tal y como se muestra en Figura 48, se podían visualizar las bibliotecas utilizadas en el sistema, incluyendo Tkinter, SQLite y OpenCV, las cuales ayudaban en el desarrollo tanto de la interfaz como en la conexión a la base de datos o cámara.

Figura 48

Librerías Python

```
test3.py | test2.py | test4
import tkinter as tk
import cv2
import sqlite3
import math
import numpy as np
import imutils
import requests
import subprocess
import os

from tkinter import ttk
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from tkcalendar import DateEntry
from PIL import Image, ImageTk
from PIL import Image
from PIL import ImageTk
#####################################
```

Fuente: Autoría

A continuación, se verificaron las líneas del script en las cuales se encontraban tanto los cuadros de texto como las etiquetas, tal y como se muestra en la Figura 49, Luego, la interfaz de registro se visualizó en la ventana, lo que permitió capturar la información ingresada como identificativo para diferenciarla de las imágenes de otros pacientes.

Figura 49

Interfaz de registro

```
class VentanaPrincipal:
    db_name = 'database.db'

    def __init__(self, root):
        self.database = Database(self.db_name)
        self.root = root
        self.root.title("Cámara Web")
        ###########################################################################
        # Crear etiquetas para el nombre y apellido del usuario
        label_nombre = tk.Label(root, text="Cedula:")
        label_nombre.grid(row=2, column=0, padx=5, pady=5)

        label_apellido = tk.Label(root, text="Apellido:")
        label_apellido.grid(row=3, column=0, padx=5, pady=5)

        label_cedula = tk.Label(root, text="Nombre:")
        label_cedula.grid(row=4, column=0, padx=5, pady=5)

        label_Methodos = tk.Label(root, text="Evaluacion")
        label_Methodos.grid(row=7, column=0, columnspan=2, padx=5, pady=5, sticky = W + E)

        label_peso = tk.Label(root, text="Peso (kg): ")
        label_peso .grid(row=5, column=0, padx=5, pady=5, sticky = W + E)

        label_altura = tk.Label(root, text="Altura (m): ")
        label_altura.grid(row=6, column=0, padx=5, pady=5, sticky = W + E)

        # Crear cuadros de entrada para el Peso y Altura
        self.entry_peso = tk.Entry(root)
        self.entry_peso.grid(row=5, column=1, padx=5, pady=5)
        self.entry_altura = tk.Entry(root)
        self.entry_altura.grid(row=6, column=1, padx=5, pady=5)

        # Crear cuadros de entrada para cedula
        self.entry_nombre = tk.Entry(root)
        self.entry_nombre.focus()
        self.entry_nombre.grid(row=2, column=1, padx=5, pady=5)
```

Fuente: Autoría

Por otro lado, se revisó el método principal de visualización en el que se estableció la conexión entre la cámara y el sistema, permitiendo iniciarlo y posteriormente ser empleado en otros métodos, como el de comprobación. Este método permitió ver la posición en tiempo real del paciente para luego pasar al método de captura. Como su nombre lo indica, el método de captura se encargó de tomar la foto del paciente y guardarla en la ruta especificada. Todo esto se muestra en la Figura 50.

Figura 50

Código de adquisición imagen

```
################################################################################
    def tomar_foto(self):
        directory = r'/home/xheo/Desktop/Tesis/Imagenes'
        os.chdir(directory)

        nombre = self.entry_nombre.get()

        IMAGE_WIDTH = 1920
        IMAGE_HEIGHT = 1080
        # Capturar imagen de la cámara
        url = "http://192.168.1.3:8080/shot.jpg"
        #cap = cv2.VideoCapture(0)
        #ret, frame = cap.read()
        img_resp = requests.get(url)
        img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
        img = cv2.imdecode(img_arr, -1)
        img = imutils.resize(img, width=IMAGE_WIDTH, height=IMAGE_HEIGHT)


        # Guardar imagen
        filename = f"{nombre}.jpg"
        cv2.imwrite(filename, img)

        # Habilitar botón para ver la foto
        self.button_ver_foto.config(state="normal")

    def ver_foto(self):
        directory = r'/home/xheo/Desktop/Tesis/Imagenes'
        os.chdir(directory)

        nombre = self.entry_nombre.get()

        # Mostrar imagen en el label correspondiente
        filename = f"{nombre}.jpg"
        image = Image.open(filename)
        image = image.resize((400, 400))
        photo = ImageTk.PhotoImage(image)
        self.image_label.configure(image=photo)
        self.image_label.image = photo
################################################################################
```

Fuente: Autoría

Para finalizar, se verificó el método de conexión entre el sistema y la base de datos, el cual permitió establecer la comunicación para guardar la información del paciente y la imagen en una base de datos previamente creada. Todo esto se muestra en la Figura 51.

Figura 51

Código guardar base de datos

```
#Image need to be conver into binary before insert into database
def convertToBinaryData(filename):
    # Convert digital data to binary format
    with open(filename, 'rb') as file:
        blobData = file.read()
    return blobData

def insertBLOB():
    directory = r'/home/xheo/Desktop/Tesis/Imagenes'
    os.chdir(directory)
    try:
        empId = idcam.get()
        name = namecam.get()
        apellido = apellidocam.get()
        foto = namecam.get()+apellidocam.get()+'.jpg'
        sqliteConnection = sqlite3.connect('SQLite_Fotografias.db')
        cursor = sqliteConnection.cursor()
        print("Connected to SQLite")
        sqlite_insert_blob_query = """ INSERT INTO SqliteDb_pacientes
                                  (id, nombre, apellido, foto) VALUES (?, ?, ?, ?)"""

        empPhoto = convertToBinaryData(foto)
        # Convert data into tuple format
        data_tuple = (empId, name, apellido, empPhoto)
        cursor.execute(sqlite_insert_blob_query, data_tuple)
        sqliteConnection.commit()
        print("Image and file inserted successfully as a BLOB into a table")
        cursor.close()

    except sqlite3.Error as error:
        print("Failed to insert blob data into sqlite table", error)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("the sqlite connection is closed")
```

Fuente: Autoría

4.1.2.2 Verificación de Registro

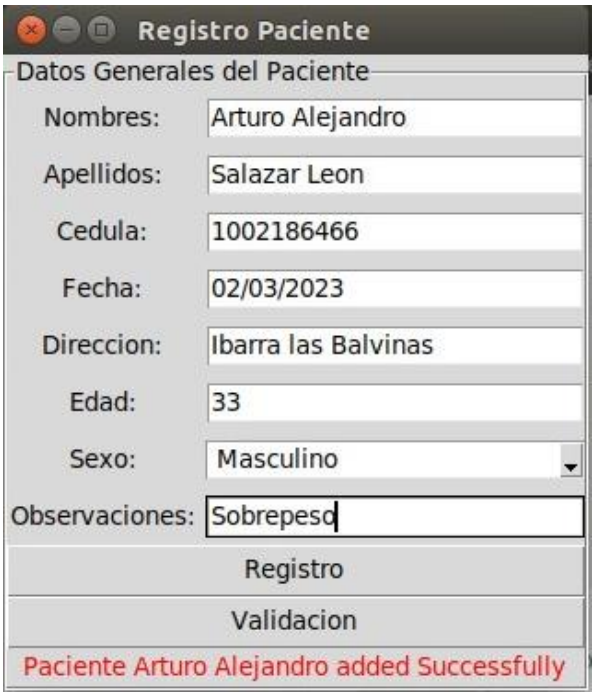
En este apartado se simuló el registro de una ficha médica de manera digital. Para ello, se ingresó información del paciente, como nombres, apellidos, cédula, edad, sexo y otros campos que pasaron por un método de validación previamente programado, creando un filtro que evitaba campos vacíos o con información errónea.

Si los campos cumplían con este filtro, se guardaban en la base de datos, la cual se utilizaba como registro digital de las visitas del paciente. Además, se podía agregar información necesaria, como el progreso por el campo de descripción.

Todo este proceso se llevaba a cabo a través de un script, que llamaba una ventana con todos los campos mencionados. La Figura 52 ilustra esta ventana.

Figura 52

Interfaz de registro



Fuente: Autoría

En este apartado se explicaron ciertas partes del código con su respectiva prueba para entenderlo mejor. En primer lugar, se ejecutaron las librerías necesarias, al igual que en el caso anterior, para hacer llamado a las funciones que se necesitan.

Por otro lado, como se puede observar en la Figura 53, se crearon las etiquetas y los cuadros de texto donde se llenará la información del paciente con los campos mencionados anteriormente.

Figura 53

Código validaciones

```
        # Creating a Frame Container
        self.frame = LabelFrame(self.wind, text = 'Datos Generales del Paciente')
        self.frame.grid(row = 0, column = 1, columnspan = 3, pady = 0)

        # Nombres Input
        Label(self.frame, text = 'Nombres: ').grid(row = 1, column = 0, pady = 5)
        self.name = Entry(self.frame, width = 25)
        self.name.focus()
        self.name.grid(row = 1, column = 1)
        # Aplellidos Input
        Label(self.frame, text = 'Apellidos: ').grid(row = 2, column = 0, pady = 5)
        self.apellido = Entry(self.frame, width = 25)
        self.apellido.grid(row = 2, column = 1)
        # Cedula Input
        Label(self.frame, text = 'Cedula: ').grid(row = 3, column = 0, pady = 5)
        self.cedula = Entry(self.frame, width = 25)
        self.cedula.grid(row = 3, column = 1)

        # Fecha Input
        Label(self.frame, text = 'Fecha: ').grid(row = 4, column = 0, pady = 5)
        self.fecha = Entry(self.frame, width = 25)
        self.fecha.grid(row = 4, column = 1)

        # Direccion Input
```

Fuente: Autoría

A continuación, se muestra el método de validación de caracteres. Este método, mediante ciertas líneas de código, discrimina entre letras mayúsculas, minúsculas, caracteres y números, permitiendo tener de manera más ordenada y precisa la información del paciente. Además, si un dato está mal introducido, se muestra un mensaje en la ventana informando que se debe corregir. Esto se realiza mediante el código que se muestra en las Figura 54 y 55.

Figura 54

Código patrones de validación

```
def definir_patrones_validaciones(self):
    #Nombres
    patron_nombre =  r'^[a-zA-Z]+( [a-zA-Z]+)+$'
    self.regex_nombre = re.compile(patron_nombre)
    #Apellidos
    patron_apellidos = r'^[a-zA-Z]+( [a-zA-Z]+)+$'
    self.regex_apellidos = re.compile(patron_apellidos)
    #Cedula
    patron_cedulav = r'^[0-9]{10,13}$'
    self.regex_cedulav = re.compile(patron_cedulav)
    #Fecha
    patron_fechav = r'^(?:3[01]|[12][0-9]|0?[1-9])([\-/.])(0?[1-9]|1[1-2])\1\d{4}$'
    self.regex_fechav = re.compile(patron_fechav)
    #Direccion
    patron_direccionv = r'^[^\s]+( [^\s]+)+$'
    self.regex_direccionv = re.compile(patron_direccionv)
    #Edad
    patron_edadv = r'^[0-9]{1,2}$'
    self.regex_edadv = re.compile(patron_edadv)
    #Observaciones
    patron_obs = r'^[^\s]+( [^\s]+)+$'
    self.regex_obs= re.compile(patron_obs)
```

Fuente: Autoría

Figura 55

Código de guardar cambios

```
def guardar(self):
    pass
    #Nombre
    nombre = self.name.get().strip()
    if re.match(self.regex_nombre, nombre) is None:
        #messagebox.showwarning('Mensaje', 'El campo Nombre debe incluir los Nombre del Paciente')
        self.message['text'] = 'Campo: Nombres is Required'
        return
    #Apellido
    apellidos = self.apellido.get().strip()
    if re.match(self.regex_apellidos, apellidos) is None:
        #messagebox.showwarning('Mensaje', 'El campo Apellido debe incluir los Apellidos del Paciente')
        self.message['text'] = 'Campo: Apellidos is Required'
        return
    #Cedula
    cedulav = self.cedula.get().strip()
    if re.match(self.regex_cedulav, cedulav) is None:
        #messagebox.showwarning('Mensaje', 'El campo Cedula debe tener de 10 a 13 caracteres numericos')
        self.message['text'] = 'Campo: Cedula is Required'
        return
    #Fecha
    fechav = self.fecha.get().strip()
    if re.match(self.regex_fechav, fechav) is None:
        #messagebox.showwarning('Mensaje', 'El campo Fecha debe tener el formato dd/mm/aaaa')
        self.message['text'] = 'Campo: Fecha is Required formato dd/mm/aaaa'
        return
    #Direccion
    direccionv = self.direccion.get().strip()
    if re.match(self.regex_direccionv, direccionv) is None:
        #messagebox.showwarning('Mensaje', 'El campo Direccion no es correcto ')
        self.message['text'] = 'Campo: Direccion is Required'
        return
```

Fuente: Autoría

Para finalizar, en la Figura 56, se muestra el código utilizado para guardar la información del registro del paciente en la base de datos. En este código, se captura la información de los cuadros de texto y se verifica si están llenos. Luego, se procede a ubicarlos en la estructura de la base de datos para guardar la información correctamente.

Figura 56

Código guardar en base de datos

```
# Function add pacient in Database
def add_pacient(self):
    if self.validation():
        query = 'INSERT INTO registro VALUES(NULL, ?, ?, ?, ?, ?, ?, ?, ?)'
        parameters =  (self.name.get(), self.apellido.get(), self.cedula.get(), self.fecha.get(), self.direccion
        self.run_query(query, parameters)
        #messagebox.showwarning('Mensaje', 'El Paciente a sido registrado correctamente')
        self.message['text'] = 'Paciente {} added Successfully'.format(self.name.get())
        self.name.delete(0, END)
        self.apellido.delete(0, END)
        self.cedula.delete(0, END)
        self.fecha.delete(0, END)
        self.direccion.delete(0, END)
        self.edad.delete(0, END)
        self.sexo.delete(0, END)
        self.observacion.delete(0, END)
    else:
        self.message['text'] = 'Datos is Required'
        #messagebox.showwarning('Mensaje', 'El Registro debe contener todos los datos del paciente')
```

Fuente: Autoría

4.1.2.3 Verificación de Evaluación

En este apartado de evaluación se ejecutaron los algoritmos, los cuales ayudaron a detectar los ángulos e índices necesarios para determinar la hiper/hipo cifosis o lordosis. La

verificación de estos se realizó por medio de una interfaz visual en la que se debía marcar el ángulo y las distancias obtenidas a través de imágenes o videos.

Como punto de partida para la comprobación de este método de evaluación, se describieron partes clave del código necesarias para llegar al resultado esperado.

En el caso de los ángulos, como se estableció en la sección de diseño 3.9.1.1, se procedió a la determinación de 3 puntos, de los cuales dos fueron proporcionados por los marcadores y el tercer punto fue ubicado por el fisioterapeuta en el sistema para el cálculo del ángulo. Este proceso se encuentra en las siguientes líneas de código de la Figura *57* Código de detección de ángulos.

Figura 57

Código de detección de ángulos

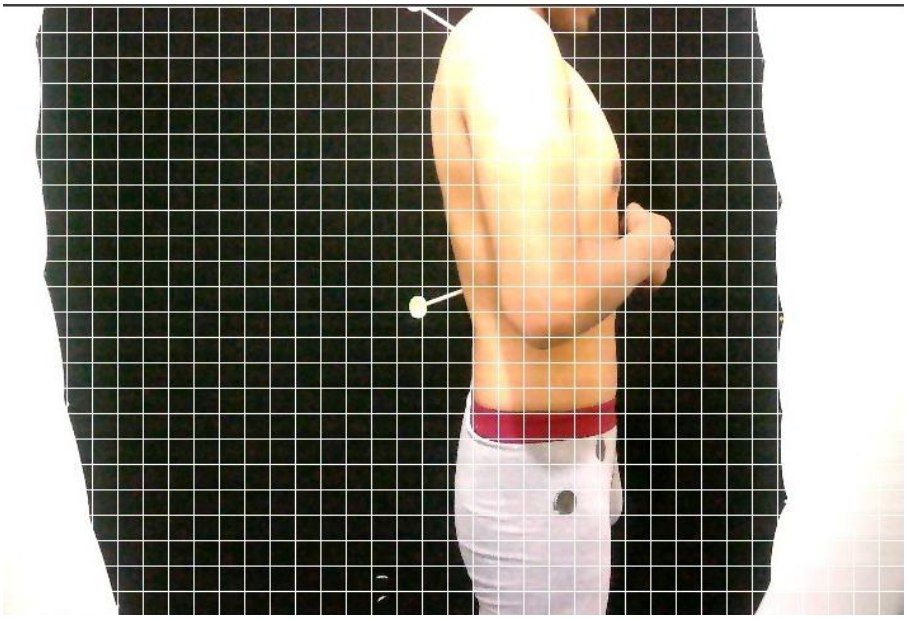
```
# Función para capturar los puntos del mouse
def mousePoints(event, x, y, flags, params):
    if event == cv2.EVENT_LBUTTONDOWN:
        size = len(pointsList)
        if size != 0 and size % 3 != 0:
            cv2.line(img, tuple(pointsList[round((size-1)/3)*3]), (x,y), (0,0,255), 2)
        cv2.circle(img, (x,y), 5, (0,0,255), cv2.FILLED)
        pointsList.append([x,y])

# Función para calcular el ángulo entre dos líneas
def gradient(pt1, pt2):
    return (pt2[1] - pt1[1]) / (pt2[0] - pt1[0])

def getAngle(pointsList):
    pt1, pt2, pt3 = pointsList[-3:]
    m1 = gradient(pt1, pt2)
    m2 = gradient(pt1, pt3)
    angR = math.atan((m2 - m1) / (1 + (m2 * m1)))
    angD = round(math.degrees(angR))
    angD = abs(angD)
    cv2.putText(img, str(angD), (pt1[0] - 40, pt1[1] - 20), cv2.FONT_HERSHEY_COMPLEX, 1.5, (0,0,255), 2)
    return (angD)
```

Fuente: Autoría

Una vez establecidos los puntos y especificado el algoritmo, se procedió, a través de una imagen previamente capturada, a analizarla y ubicar los puntos necesarios para la determinación de cada ángulo, como se muestra en la Figura 58.

Figura 58

Interfaz de obtención de ángulos

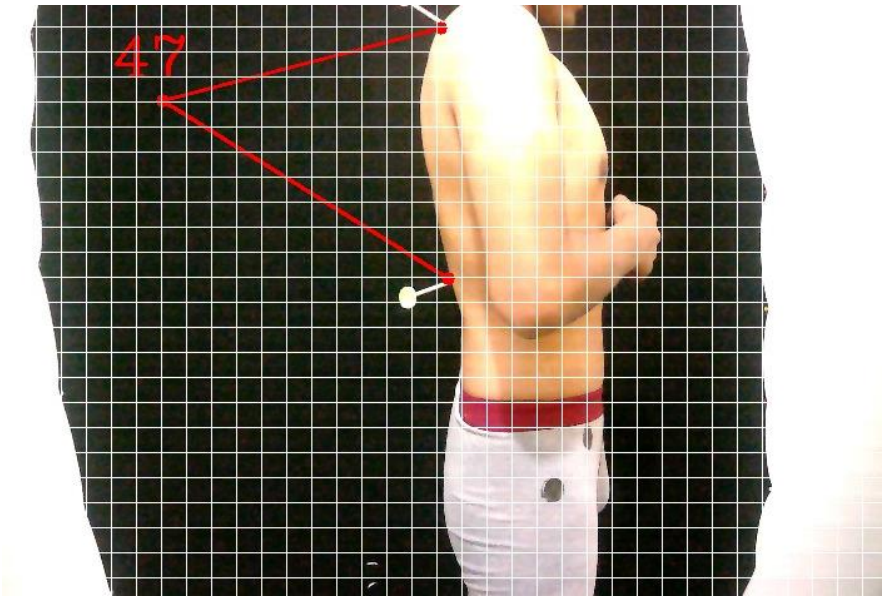


Fuente: Autoría

Para obtener el ángulo correspondiente a la región torácica, se tomaron los puntos de los dos marcadores y, con la ayuda del fisioterapeuta, se ubicó el tercer punto necesario para obtener el ángulo deseado.

Figura 59

Angulo Cifotico

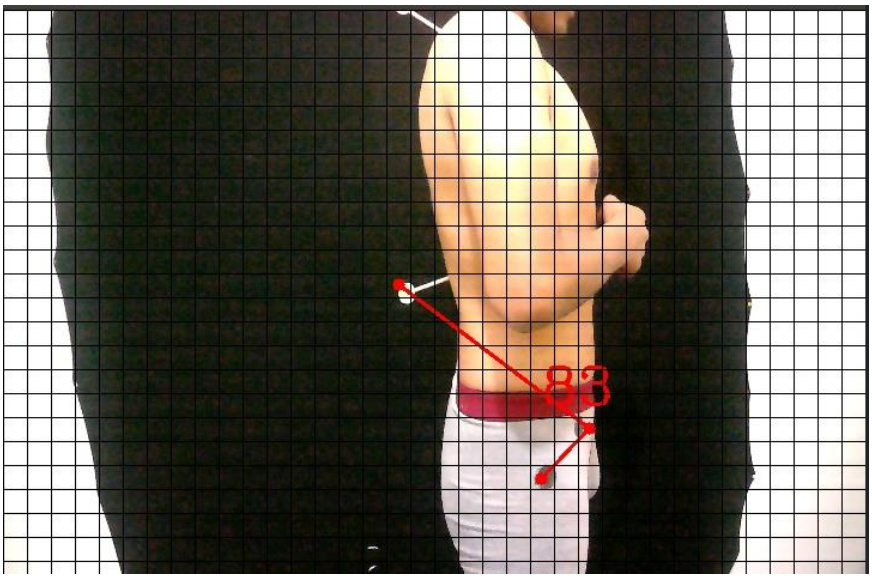


Fuente: Autoría

Para obtener el ángulo correspondiente a la región lumbar, se consideraron dos puntos adicionales junto con el punto ubicado en T12. Con estos tres puntos, se estableció el ángulo deseado, como se muestra en la siguiente imagen.

Figura 60

Angulo Lumbar



Fuente: Autoría

En el cálculo de las distancias, como se explicó en la sección de diseño 3.9.1.1, utilizamos un algoritmo que nos permitió seleccionar la región de interés a analizar mediante la función ROID. Esta función nos ayudó a establecer una dimensión específica para nuestra imagen, lo que permitió establecer la relación entre distancia y píxeles necesaria para el cálculo de las distancias requeridas. Este proceso se muestra en la Figura 61..

Figura 61

Código de perspectiva

```python
def clics(event,x,y,flags,param):
    global pun
    if event == cv2.EVENT_LBUTTONDOWN:
        pun.append([x,y])

def dibujando_puntos(pun):
    for x, y in pun:
        cv2.circle(frame,(x,y),5,(255,255,255),2)

def uniendo4puntos(pun):
    cv2.line(frame,tuple(pun[0]),tuple(pun[1]),(255,0,0),1)
    cv2.line(frame,tuple(pun[0]),tuple(pun[2]),(255,0,0),1)
    cv2.line(frame,tuple(pun[2]),tuple(pun[3]),(255,0,0),1)
    cv2.line(frame,tuple(pun[1]),tuple(pun[3]),(255,0,0),1)

def roi(image, ancho, alto):
    imagen_alineada = None
    global pun
    cv2.namedWindow('frame')
    cv2.setMouseCallback('frame',clics)
    dibujando_puntos(pun)
    if len(pun) == 4:
        pts1 = np.float32([pun])
        pts2 = np.float32([[0,0], [ancho,0], [0,alto], [ancho,alto]])
        M = cv2.getPerspectiveTransform(pts1, pts2)
        imagen_alineada = cv2.warpPerspective(image, M, (ancho,alto))

    return imagen_alineada
```

Fuente: Autoría

Una vez obtenida la imagen previamente capturada, se la utilizó como base para detener nuestros marcadores, como se puede apreciar en la Figura 62.. En el código de detección de marcadores, se identificaron los círculos de colores correspondientes a cada punto de la flecha sagital, y se calculó la distancia hasta el eje más cercano (en este caso, X=0). A partir de estas distancias, se realizó un proceso matemático para el cálculo de los índices.

Figura 62

Código de detección de marcadores

```python
if imagen_A4 is not None:
    puntos = []
    puntosv = []
    imagenHSV = cv2.cvtColor(imagen_A4, cv2.COLOR_BGR2HSV)
    rojoBajo1 = np.array([0, 100, 20], np.uint8)
    rojoAlto1 = np.array([10, 255, 255], np.uint8)
    rojoBajo2 = np.array([175, 100, 20], np.uint8)
    rojoAlto2 = np.array([180, 255, 255], np.uint8)
    maskRojo1 = cv2.inRange(imagenHSV , rojoBajo1, rojoAlto1)
    maskRojo2 = cv2.inRange(imagenHSV , rojoBajo2, rojoAlto2)
    maskrojo =  cv2.add(maskRojo1, maskRojo2)
    cnts = cv2.findContours(maskrojo, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:2]
    for c in cnts:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(imagen_A4, (x, y), (x+w, y+h), (255, 0, 0), 2)
        puntos.append([x, y, w, h])
    if len(puntos) == 2:
        x1, y1, w1, h1 = puntos[0]
        x2, y2, w2, h2 = puntos[1]
        if x1 < x2:
            distancia_pixeles = abs(x1)
            distancia_cm1 = ((distancia_pixeles*100)/720)
            cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cm1), (distancia_pixeles//2, y1), 2, 0.8, (0,0,25
            cv2.line(imagen_A4,(0,y1),(x1, y1),(0, 0, 255),2)
        if x1 < x2:
            distancia_pixeles = abs(x2)
            distancia_cm2 = (distancia_pixeles*100)/720
            cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cm2), (distancia_pixeles//2, y2), 2, 0.8, (0,0,25
            cv2.line(imagen_A4,(0,y2),(x2, y2),(0, 0, 255),2)
```

Fuente: Autoría

Para demostrar esto, realizamos la ejecución del script que permitió, por medio de un video en tiempo real, seleccionar la región de interés correspondiente al fondo blanco en la parte posterior, como se muestra en la siguiente imagen.

Figura 63

Interfaz de flechas sagital

Fuente: Autoría

Seleccionada la región de interés este procede a detectar los marcadores y darnos las distancias como se muestra en la Figura 64.

Figura 64

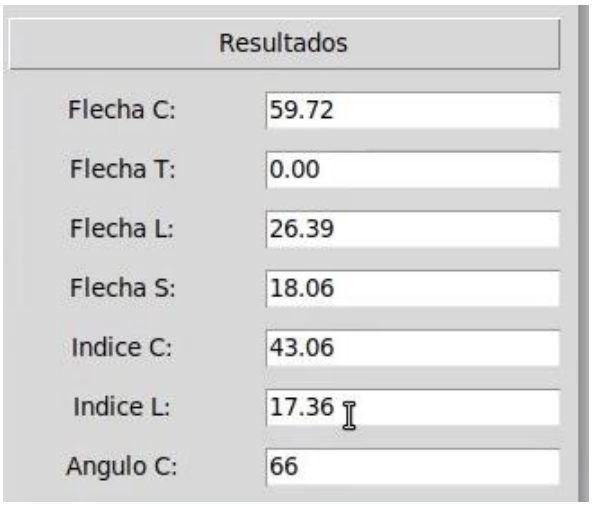
Detección de flechas sagitales

Fuente: Autoría

Con estas distancias, se procedió a la interfaz final de evaluación y se obtuvieron los resultados tanto para los índices que estaban dados en milímetros como para los ángulos dados en grados.

Figura 65

Interfaz de evaluación



Fuente: Autoría

4.1.2.4 Verificación de Base de Datos

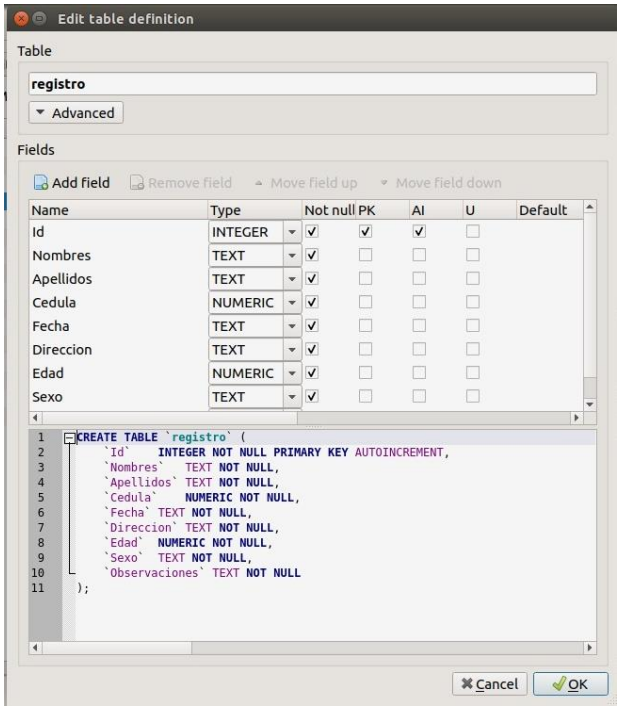
En el sistema, la base de datos se utilizó para almacenar toda la información del paciente, incluyendo su ficha y los resultados de la evaluación. Para lograr esto, el subsistema se basó en sqlite3, permitiendo una base de datos local.

En lo que respecta al registro de pacientes, el sistema digitalizó una ficha médica en la que se solicitaba información del paciente, como su nombre, apellido, cédula, entre otros datos. Al presionar el botón correspondiente, la información ingresada se conectaba con la base de datos y se almacenaba en una tabla previamente creada. Para verificar que este proceso se realizó correctamente, se verificó el registro digital y se comparó con la información almacenada en la base de datos.

A continuación, se procedió a crear la tabla correspondiente a través de la interfaz gráfica de Sqlite3, tal y como se muestra en la Figura 66, Ya sea mediante código o interfaz, se creó una tabla que cumplía con las especificaciones necesarias para almacenar la información de la ficha digital.

Figura 66

Creación tabla de registro



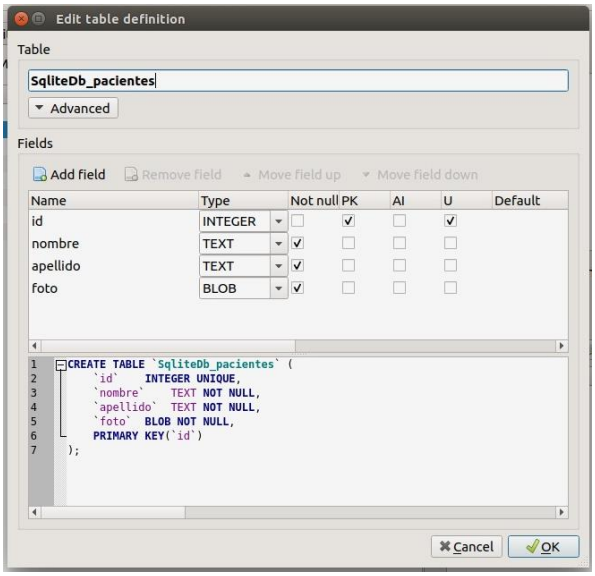
Fuente: Autoría

En el sistema, era posible realizar el mismo proceso para cada una de las tablas, tanto para la de imágenes del paciente como para la de resultados. Para ello, se especificaba el campo y el tipo de dato al que pertenecía cada entrada en la tabla correspondiente. Además, se podían

guardar imágenes como datos binarios utilizando el tipo de dato "BLOB" (Binary Large Object), tal y como se muestra en la Figura 67.

Figura 67

Creación tabla de imágenes

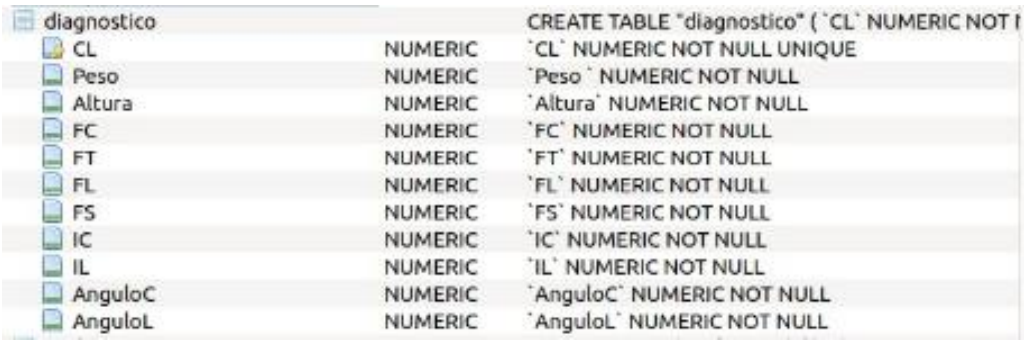


Fuente: Autoría

En el sistema, se podía utilizar la misma función que se utilizó para el registro para crear la tabla de resultados. En dicha tabla, se almacenaban los ángulos, las distancias y los índices correspondientes a cada paciente como se muestra en la Figura 68. Dado que esta tabla era la pieza central del sistema, se almacenaban los resultados junto con la cédula del paciente, ya que esta era un identificador único que no podía repetirse.

Figura 68

Tabla de resultados

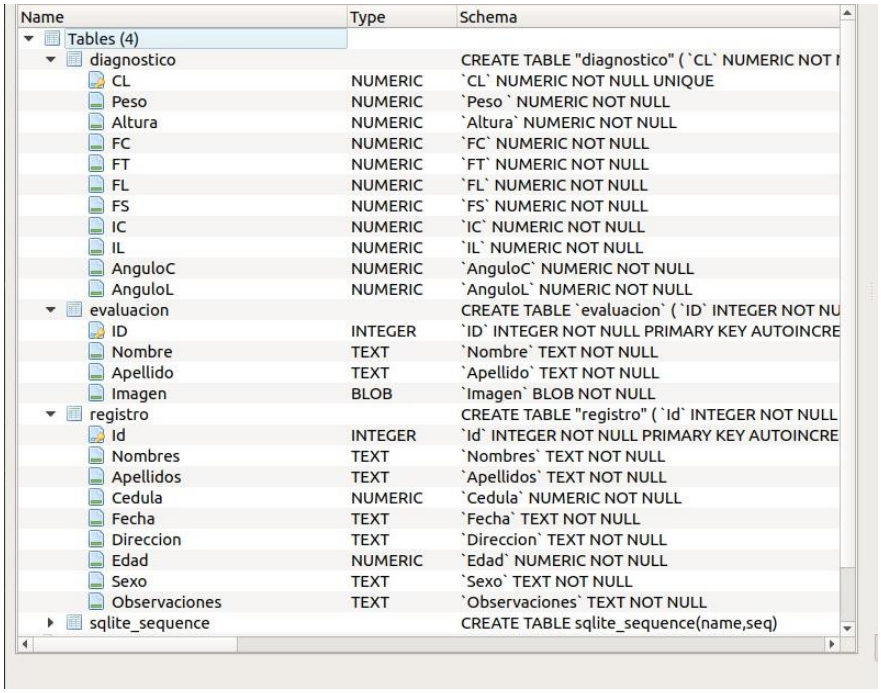


Fuente: Autoría

Una vez que fueron creadas las tablas se procedió a la comprobación en la ventana principal de Sqlite3, donde se visualizó un resumen general de las tablas y secciones creadas.

Figura 69

Verificación creación de tablas



Name	Type	Schema
Tables (4)		
diagnostico		CREATE TABLE "diagnostico" (`CL` NUMERIC NOT
CL	NUMERIC	`CL` NUMERIC NOT NULL UNIQUE
Peso	NUMERIC	`Peso ` NUMERIC NOT NULL
Altura	NUMERIC	`Altura` NUMERIC NOT NULL
FC	NUMERIC	`FC` NUMERIC NOT NULL
FT	NUMERIC	`FT` NUMERIC NOT NULL
FL	NUMERIC	`FL` NUMERIC NOT NULL
FS	NUMERIC	`FS` NUMERIC NOT NULL
IC	NUMERIC	`IC` NUMERIC NOT NULL
IL	NUMERIC	`IL` NUMERIC NOT NULL
AnguloC	NUMERIC	`AnguloC` NUMERIC NOT NULL
AnguloL	NUMERIC	`AnguloL` NUMERIC NOT NULL
evaluacion		CREATE TABLE `evaluacion` (`ID` INTEGER NOT NU
ID	INTEGER	`ID` INTEGER NOT NULL PRIMARY KEY AUTOINCRE
Nombre	TEXT	`Nombre` TEXT NOT NULL
Apellido	TEXT	`Apellido` TEXT NOT NULL
Imagen	BLOB	`Imagen` BLOB NOT NULL
registro		CREATE TABLE "registro" (`Id` INTEGER NOT NULL
Id	INTEGER	`Id` INTEGER NOT NULL PRIMARY KEY AUTOINCRE
Nombres	TEXT	`Nombres` TEXT NOT NULL
Apellidos	TEXT	`Apellidos` TEXT NOT NULL
Cedula	NUMERIC	`Cedula` NUMERIC NOT NULL
Fecha	TEXT	`Fecha` TEXT NOT NULL
Direccion	TEXT	`Direccion` TEXT NOT NULL
Edad	NUMERIC	`Edad` NUMERIC NOT NULL
Sexo	TEXT	`Sexo` TEXT NOT NULL
Observaciones	TEXT	`Observaciones` TEXT NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

Fuente: Autoría

La comprobación del registro se llevó a cabo llenando la ficha digital correspondiente y, al presionar el botón correspondiente, se ejecutó una función para establecer la conexión con la base de datos y comprobar si ésta ya estaba creada. El proceso fue similar al utilizado en la comprobación previa del registro, como se muestra en la Figura 70.

Figura 70

Código de conexión a la base de datos

```
class Database:
    def __init__(self, database_name):
        self.connection = sqlite3.connect(database_name)
        self.cursor = self.connection.cursor()

    def execute_query(self, query, params):
        self.cursor.execute(query, params)
        return self.cursor.fetchall()

    def __del__(self):
        self.connection.close()
```

Fuente: Autoría

Con la función siguiente, se recogieron los datos que se encontraban en cada etiqueta de entrada. Se compararon los datos para asegurarse de que existieran en la tabla y se rellenaron según la posición indicada. Es importante destacar que tanto la tabla como el método debían tener el mismo formato de celdas para evitar problemas de funcionamiento.

Figura 71

Código para guardar información en la base de datos

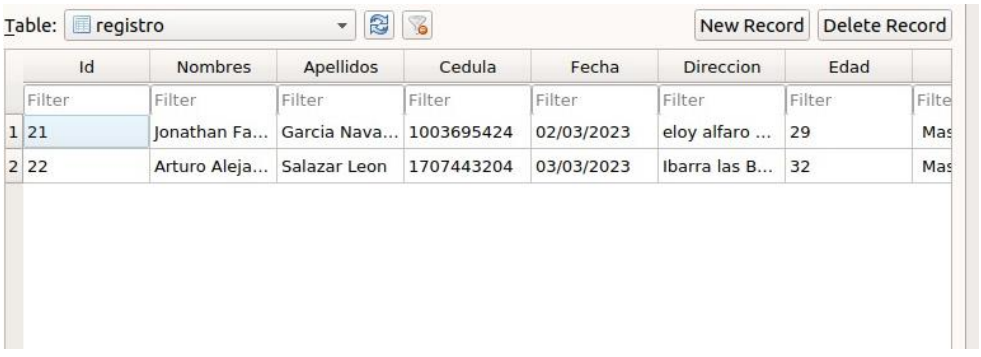
```
# Function add pacient in Database
def add_pacient(self):
    if self.validation():
        query = 'INSERT INTO registro VALUES(NULL, ?, ?, ?, ?, ?, ?, ?, ?)'
        parameters =  (self.name.get(), self.apellido.get(), self.cedula.get(), self.fecha.get(), self.direccion
        self.run_query(query, parameters)
        #messagebox.showwarning('Mensaje', 'El Paciente a sido registrado correctamente')
        self.message['text'] = 'Paciente {} added Successfully'.format(self.name.get())
        self.name.delete(0, END)
        self.apellido.delete(0, END)
        self.cedula.delete(0, END)
        self.fecha.delete(0, END)
        self.direccion.delete(0, END)
        self.edad.delete(0, END)
        self.sexo.delete(0, END)
        self.observacion.delete(0, END)
    else:
        self.message['text'] = 'Datos is Required'
        #messagebox.showwarning('Mensaje', 'El Registro debe contener todos los datos del paciente')
```

Fuente: Autoría

Para verificar este subsistema, se utilizó la interfaz gráfica de Sqlite3 para comprobar si las tablas habían guardado correctamente la información correspondiente de la interfaz del registro, evaluación o toma de imágenes en cada una de las tablas correspondientes. Esto se puede observar en la primera imagen, que muestra la tabla de registro.

Figura 72

Verificación de la información en la base de datos



Fuente: Autoría

Esto se lo realizo con cada una de las tablas creadas y su respectiva información como se visualiza en la siguiente imagen.

Figura 73

Verificación de la base de datos de imágenes

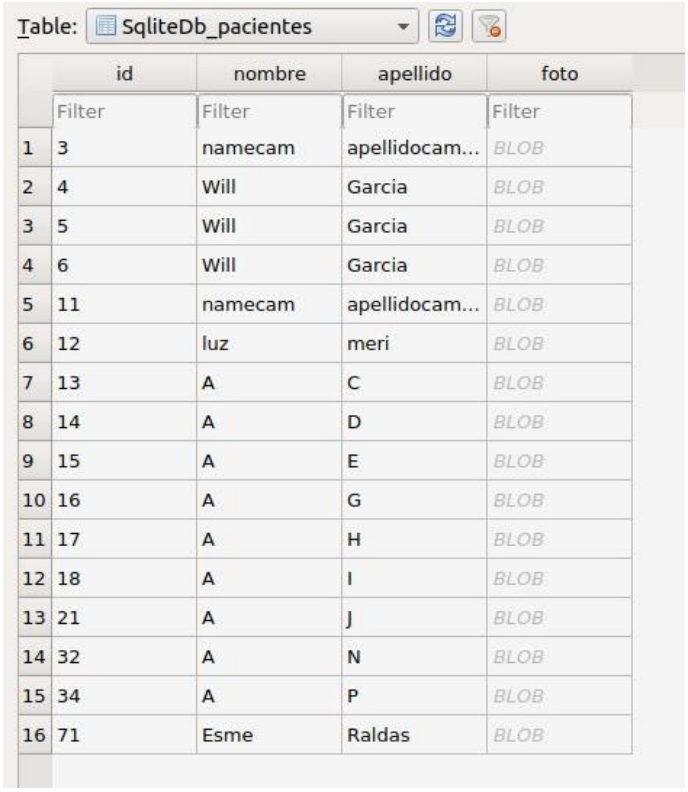


Table: SqliteDb_pacientes

	id	nombre	apellido	foto
	Filter	Filter	Filter	Filter
1	3	namecam	apellidocam...	BLOB
2	4	Will	Garcia	BLOB
3	5	Will	Garcia	BLOB
4	6	Will	Garcia	BLOB
5	11	namecam	apellidocam...	BLOB
6	12	luz	meri	BLOB
7	13	A	C	BLOB
8	14	A	D	BLOB
9	15	A	E	BLOB
10	16	A	G	BLOB
11	17	A	H	BLOB
12	18	A	I	BLOB
13	21	A	J	BLOB
14	32	A	N	BLOB
15	34	A	P	BLOB
16	71	Esme	Raldas	BLOB

Fuente: Autoría

Después de haber terminado la etapa de comprobación de cada subsistema, se realizaron pruebas para validar todo lo propuesto en el proyecto. El objetivo de estas pruebas fue verificar que se tuviera datos con coherencia objetiva para poder comparar la medición del sistema versus la medición manual.

4.2. Fase 4: Validación del Sistema

Una vez que se validaron por separado cada uno de los subsistemas, se procedió a su implementación conjunta en el dispositivo para llevar a cabo la última fase, que consistió en la validación del sistema a través de pruebas realizadas en pacientes del centro de Fisioterapia. La realización de estas pruebas contó con el criterio de una experta, la Lic. Estefanía Hidalgo,

responsable a cargo, lo que permitió comparar la evaluación del sistema con la medición manual.

4.2.1. Pruebas 3: Adquisición de Datos

El profesional realizó una evaluación manual en 10 pacientes, quienes fueron evaluados tanto para la obtención de los ángulos como para las flechas sagitales. Además, se tomaron muestras tanto manuales como del sistema para poder hacer una comparación. Para evaluar los resultados, se tuvo en cuenta el criterio de un experto que evaluó los datos en base a dos criterios: "Aceptable" (cuando el sistema arrojaba coherencia objetiva con un mínimo de error) y "No Aceptable" (cuando los valores no eran coherentes y tenían un alto grado de error).

Los resultados obtenidos por la implementación del sistema tanto en los ángulos como en los índices se basaron en ciertos rangos establecidos previamente, los cuales se muestran en la Tabla 15. Es importante destacar que estos rangos correspondían a los valores normales de curvatura en la región lumbar y cifótica. En otras palabras, si las medidas capturadas no estaban dentro de estos rangos establecidos (ya sea mayores o menores), se consideraron como hipercifosis/lordosis para los ángulos mayores a los normales, e hipercifosis/lordosis para los ángulos menores a los normales.

Tabla 15

Rangos de evaluación

Rangos de Evaluación	
Ángulos	**Rango normal de curvatura**
Cifotico	20º – 45º
Lumbar	40º - 60º
Índices	**Rango normal de curvatura**
Cifotico	20 - 40
Lumbar	30 – 50

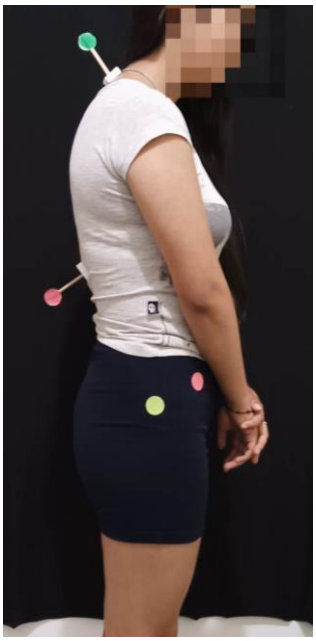
Fuente: Autoría

4.2.1.1 Pruebas ángulo Cifolumbar

En relación con el proceso de evaluación, se llevó a cabo la toma de muestras en un grupo de 10 pacientes, de los que se tomaron muestras para la detección del ángulo lumbar y cifótico mediante el método manual y con el método desarrollado para el sistema, como parte del proceso de evaluación. En el método del sistema, era importante que el paciente estuviera en la perspectiva sagital con los marcadores visibles, para que el sistema pudiera detectarlos, tal como se muestra en la Figura 74. Además, los puntos debían ser colocados en partes específicas de la espalda. Para el ángulo cifótico, el primer punto debía estar en C7 y el segundo en T12. Por otra parte, para el ángulo lumbar, se debían colocar los marcadores en las siguientes partes: el primero en T12, el segundo en la parte del trocante mayor y el último en la parte de la espina iliaca. Una vez colocados los puntos, se podía adquirir una imagen completa del paciente, con la cual, mediante nuestro sistema, se seleccionaron los 3 puntos que conformaban cada ángulo para su obtención. Se destaca que las muestras fueron obtenidas mediante el método manual y mediante el sistema para realizar la evaluación.

Figura 74

Método de medición Sistema

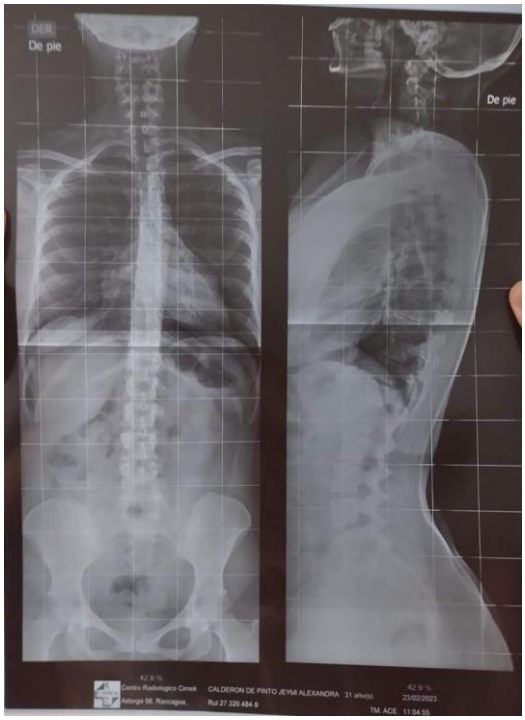


Fuente: Autoría

Por otra parte, para el método manual, se realizaron dos técnicas: radiografías y flexo-extensor. El último método se llevó a cabo en aquellos pacientes que no contaban con radiografías de su columna vertebral. En las radiografías se utilizó el método de Cobb, el cual mide los ángulos de lordosis y cifosis en la columna vertebral a través de las radiografías. El fisioterapeuta trazó líneas paralelas a las superficies superiores e inferiores de las vértebras que limitan la curvatura y se midió el ángulo de intersección para determinar la magnitud de la curvatura y progresión de la enfermedad. Por último, el método del flexo-extensor se emplea para evaluar escoliosis, cifosis, lordosis y otros trastornos de la columna vertebral, así como para medir la progresión de la enfermedad y la eficacia del tratamiento. Es una técnica no invasiva que no requiere equipo especializado, por lo que se utiliza ampliamente en la práctica clínica.

Figura 75

Método de medición Manual



Fuente: Autoría

Una vez explicados los métodos de obtención de los ángulos mediante el sistema y de forma manual, se procedió a crear la Tabla 16 para mostrar los datos recopilados de 10

pacientes tanto en el plano sagital como con radiografías. En dicha tabla se presentan los resultados obtenidos para el ángulo cifótico junto con su porcentaje de error y la evaluación del experto.

Tabla 16

Tabla de evaluación Cifosis

Paciente	Angulo Cifotico Sistema	Angulo Cifotico Manual	Error	Criterio del experto
1	21º	25.5º	4.5º	Acepta
2	42º	39.7º	1.3º	Acepta
3	45º	41.1º	3.9º	Acepta
4	34º	34º	0º	Acepta
5	43º	41.8º	1.2º	Acepta
6	38º	38.5º	0.5º	Acepta
7	45º	45º	0º	Acepta
8	37º	39º	2º	Acepta
9	47º	43.5º	3.5º	Acepta
10	29º	35º	5º	No Acepta

Fuente: Autoría

En la Tabla 17, se presentó la comparación entre los ángulos lumbares obtenidos mediante el método manual y el desarrollado por el sistema. Se realizó la comparación siguiendo el mismo criterio utilizado anteriormente, calculando la diferencia entre ambos ángulos y aplicando el criterio del experto para su evaluación.

Tabla 17

Tabla de evaluación Lumbar

Paciente	Angulo Lumbar Sistema	Angulo Lumbar Manual	Error	Criterio del experto
1	25º	27º	2º	Acepta
2	32º	38.1º	6º	No Acepta
3	57º	60º	3º	Acepta
4	65º	62º	2º	Acepta
5	48º	45.7º	2.3º	Acepta

6	60º	62.5º	2.5º	Acepta
7	30º	32º	2º	Acepta
8	48º	46.2º	1.8º	Acepta
9	50º	50.7º	0.7º	Acepta
10	60º	55º	5º	No Acepta

Fuente: Autoría

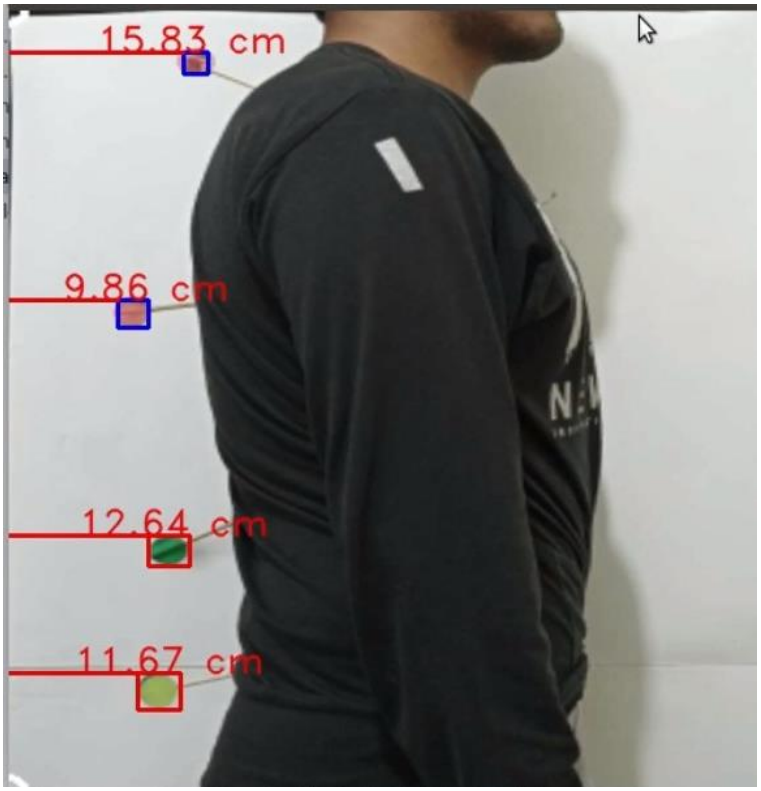
A continuación, se realizaron las pruebas para las flechas sagitales con el mismo criterio anterior tanto manuales como por el dispositivo.

4.2.2.1 Pruebas test Flecha Sagital

Para finalizar la recopilación de datos, se llevó a cabo la evaluación de las flechas sagitales en los pacientes, lo que permitió determinar los índices cifótico y lordótico. El proceso consistió en simular el método de la plomada, utilizando marcadores en los puntos relevantes de la columna vertebral, como C7, el punto más alto y bajo de la columna, y S2. Mediante la detección de estos marcadores por nuestro sistema, se calculó la distancia de cada punto respecto del eje Y. Se seleccionó la distancia más cercana a este eje para luego realizar un análisis matemático y calcular los índices cifótico y lumbar, tal como se muestra en la siguiente figura.

Figura 76

Método de Flechas Sagitales



Fuente: Autoría

En el método manual, se llevaba a cabo la evaluación mediante el uso de una plomada. Para ello, el fisioterapeuta colocaba al paciente en posición sagital y adhería la plomada a su cuerpo, determinando el punto de contacto que en algunos casos podían ser FT o FS. A continuación, se hacía coincidir toda la cuerda a ese punto, tal como se muestra en la figura. El fisioterapeuta medía la distancia a cada flecha con una regla y, con estos datos, se resolvía una ecuación para determinar el índice cifótico y lumbar.

Figura 77

Método de Flechas Sagitales manual

Fuente: Autoría

Explicado lo anteriormente se procedió a llenar la siguiente tabla donde compararemos el método manual con el sistema. En la obtención de las flechas sagitales.

Tabla 18

Tabla Flechas Sagitales

Paciente	Evaluación Sistema				Evaluación Manual				Criterio del experto
	FC	FT	FL	FS	FC	FT	FL	FS	
1	59.7	0	27.8	17.5	57.2	0	26.9	18.1	Acepta
2	70,9	0	53,2	4,2	68.5	0	51.6	3.7	Acepta
3	109,2	45,2	79,8	0	105.8	44.1	79.2	0	Acepta
4	98	11,5	42,4	0	97,3	11,9	41,7	0	Acepta
5	73,5	0	56	22,5	71,4	0	58	21,9	Acepta
6	102,3	13,9	57,9	0	98,8	12,1	56,4	0	Acepta
7	74,6	12	56,8	0	75,1	11,7	54,8	0	Acepta
8	76,8	0	33,8	2,6	84,9	0	30,9	1,8	Acepta
9	53,8	0	31,1	11,7	50,2	0	32,4	12,1	Acepta
10	92,4	0	52,7	27,2	93,1	0	51,4	26,8	Acepta

Fuente: Autoría

Con lo referente a la obtención de los datos de las flechas se aplica la formula establecida en la sección 2.1.4.3 para el cálculo de los índices los cuales se muestran en las Tablas 19 y 20.

Tabla 19

Tabla de Indicé de Cifosis

Paciente	Indicé Cifotico Sistema	Índice Cifotico Manual	Error	Criterio del experto
1	43.75	42.05	1,7	Acepta
2	62.05	60.05	2	Acepta
3	49.3	48.4	0,9	Acepta
4	58.7	57.6	1,1	Acepta
5	64.75	64.7	0,05	Acepta
6	62.2	65.5	3,3	Acepta
7	53.7	53.25	0,45	Acepta
8	55.3	57.9	2.6	Acepta
9	42.45	41.3	1,15	Acepta
10	72.55	72.25	0,3	Acepta

Fuente: Autoría

Tabla 20

Tabla de índice lumbar

Paciente	Indicé Lumbar Sistema	Índice Lumbar Manual	Error	Criterio del experto
1	19.05	17.85	1,2	Acepta
2	51.1	49.75	1,35	Acepta
3	57.1	57.15	0,05	Acepta
4	36.65	35.75	0,9	Acepta
5	49.75	47.05	2,7	Acepta
6	50.95	50.35	0,6	Acepta
7	50.8	48.95	1,85	Acepta
8	32.5	30	2,5	Acepta
9	25.25	26.35	1,1	Acepta

10	39.1	38	1,1	Acepta

Fuente: Autoría

4.2.2. Resultados Obtenidos

Al comparar el sistema con el método tradicional, se establece que el sistema embebido fue una opción viable como método alternativo de evaluación debido a sus factores positivos en el pasado.

Entre los factores positivos del sistema, se destaca que ayuda a realizar evaluaciones de manera más rápida, siendo un método de evaluación ágil y novedoso en el pasado. Además, el sistema permitió una disminución en el uso de radiografías en la mayoría de los casos, ya que las pruebas realizadas evidenciaron un margen de error mínimo en la visualización de ángulos y distancias en el pasado. Se puede añadir que el dispositivo podía seguir mejorando para brindar un mejor rendimiento.

En lo que respecta a los métodos de evaluación del sistema, si este contaba con un entorno adecuado y la calibración correcta, permitía al fisioterapeuta y paciente reducir significativamente el tiempo de evaluación en el pasado, lo que permitía tomar datos dentro de los límites propuestos en el pasado.

Para finalizar, el fisioterapeuta a cargo dejó constancia de que, mediante los resultados obtenidos en la implementación del sistema, como también en las pruebas realizadas en el pasado, se pudo determinar que fue un dispositivo viable para permitir evaluar los ángulos y distancias en tiempo real en el pasado.

Como recomendación, se estableció que el sistema, al depender del entorno en el cual se realizaban las pruebas, debía ser lo más controlado posible en el pasado debido a que había muchos factores que alteraban las imágenes al momento de adquirirlas. Además, se destacó la importancia del uso de marcadores más adecuados en el pasado, ya que, si estos no sobresalían del cuerpo del paciente, el sistema no podía reconocerlos y, como resultado, arrojaba datos no coherentes e independientes en el pasado.

4.3. Fase 5: Mantenimiento del Sistema

Con respecto al mantenimiento del sistema, este fue un proceso de revisión, mejora y actualización después de su implementación. El objetivo del mantenimiento era garantizar que el dispositivo continuara funcionando de manera óptima y eficiente a lo largo del tiempo.

El mantenimiento incluía la corrección de errores, la actualización de características, la mejora del rendimiento, la optimización del uso de recursos, la eliminación de bugs y la capacitación para el uso del dispositivo.

Es importante destacar que, en lo referente a la corrección de errores, no solo se trata de una tarea técnica, sino que también implica tareas como la adecuación del área de trabajo, la cual resultó ser un factor importante para el sistema. Un buen mantenimiento implicaba la capacidad de responder rápidamente si surgía un error, brindando soporte técnico, capacitación y soluciones a los problemas que pudieran presentarse después de la implementación.

Para ello, durante el tiempo de duración del proyecto se capacitó al fisioterapeuta encargado para que pudiera solucionar errores básicos que pudieran presentarse en el sistema en el futuro. Cabe destacar que este proyecto aún puede mejorarse, ya que el campo de visión artificial avanza rápidamente y se desarrollan nuevos métodos con el tiempo. Se podrían corregir errores como la adquisición de datos y mejorar el sistema de evaluación.

Con el desarrollo y el tiempo de ejecución del proyecto, se cumplieron los parámetros necesarios requeridos por las personas involucradas en él, hasta su implementación final y pruebas de funcionamiento para evaluar si el sistema era viable como alternativa de evaluación. Cabe destacar que, si bien los métodos tradicionales son más precisos y dan resultados más exactos, requieren mucho tiempo para su ejecución. En cambio, el sistema propuesto se ejecutaba rápidamente y proporcionaba resultados casi tan precisos como los tradicionales.

Conclusiones

En el presente proyecto se propuso un sistema embebido que emplee el uso de visión artificial con el objetivo de medir el ángulo cifolumbar en la región de la columna vertebral el mismo que demostró un nivel de confiabilidad del 91% después de la realización de las pruebas, de esta manera cumpliendo con los parámetros impuestos por el centro de fisioterapia.

El sistema de medición de ángulos y distancias enfocado en la región de la espalda es un instrumento funcional que puede ser utilizado como un método de evaluación opcional en el área de fisioterapia, debido a que, en diferentes pruebas realizadas, se observa una respuesta optima que no supera un margen de error de 3º siendo un método casi optimo en comparación con la medición manual a través del método de la lomada y las radiografías.

El sistema que fue desarrollado a través de la visiona artificial es un método de evaluación no invasivo para el paciente por ende permitiendo de una manera más rápida y ágil el registro y la evaluación del paciente por parte del profesional encargado, además de convertirse en una herramienta alternativa para los procesos de evaluación en este caso.

La ayuda de un profesional de fisioterapia al momento de realizar el diseño de los algoritmos es fundamental para tener una mejor claridad de como este tendría que funcionar, desarrollando un sistema el cual permita la toma de datos y así también como sintetizar el registro de pacientes cumpliendo así con los objetivos planteados.

El sistema cumple satisfactoriamente con los objetivos que nos planteamos al iniciar este proyecto, utilizando tecnología innovadora y adaptándonos a los diferentes escenarios que se presentaron en el desarrollo de este.

Recomendaciones

Es recomendable para realizar las mediciones de los ángulos que el dispositivo se encuentre aproximadamente a una distancia de 1.5 a 2 metros en la cual se pueda tener una visibilidad completa del plano sagital del paciente esto puede variar independientemente de las características del paciente.

Para mejorar la toma de datos es recomendable una cámara de resolución promedio de 720p o 1080p para sistemas de visión artificial en entornos controlados por su facilidad de adquisición y su costo son una buena, por el contrario, una cámara de alta resolución de 2k o inclusive 4K es una opción viable si el entorno no está controlado.

Es recomendable si se usa una cámara web realizar antes la calibración de esta debido a que la mayoría de las cámaras cuentan con el efecto de ojo de pez el cual da a la imagen una perspectiva angular innecesaria y por medio de la calibración esta se puede corregir de una manera más optima o es recomendable el uso de una cámara de celular la cual nos da una imagen más plana debidos a sus sensores y resolución.

Para aumentar el margen de detección se debe tratar en entornos con un fondo de preferencia negro y sin objetos ni personas que intervengan en la adquisición de datos del sistema.

Para mejorar la medición de las distancias se debe tener un control del entorno tanto en la resolución de imágenes como en el factor de relación de aspecto para que en la región de interés que se seleccione se aplica la relación de pixel/distancia para la toma de datos más correctos.

Es necesario que la persona al momento de ser evaluado por el sistema use ropa ajustada de preferencia de colores obscuros y no use ropa holgada o chompa, debido a que estas obstruyen la correcta colocación de los marcadores generando errores de detección esquelética.

Para evitar los errores por movimiento del paciente en el plano a evaluarse es recomendable que tanto el entorno se encuentre en condiciones óptimas, al mismo tiempo que se encuentren marcados los puntos de ubicación correcta del paciente para la adquisición de la imagen en el plano deseado.

Se propone que para mejorar la experiencia del usuario en un futuro se implemente un entorno de adquisición más optimo el cual permita de una manera más rápida la adquisición de imágenes sin mucho índice de error.

Es recomendable el desarrollo de un modelo más optimizado que pueda ser portable usando menos recursos de hardware y permita la evaluación del paciente independientemente del entorno con un tiempo de respuesta óptimo para la detección y procesamiento de datos.

Referencias

Aguilera, G. (2010). PROCESAMIENTO DE IMÁGENES. *300910.* UNIVERSIDAD DE SALAMANCA, España.

Aguirre, D. (2013). IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE SEÑALES DE TRÁFICO MEDIANTE VISIÓN ARTIFICIAL BASADO EN FPGA. *(Tesis de Ingeniería Superior de Telecomunicación).* Universidad de Sevilla, Sevilla.

Aittor, L. (2000). Pruebas diagnósticas en la Escoliosise indicaciones ortopédicas. *Monografico RPG* , 4.

Alejandra, G. (2014). *LA ESPINA BÍFIDA COMO PUNTO DE APRTIDA PARA EL DESARROLLO DE UN PROGRAMA DE INFORMACION Y ORIENTACION ENFERMERA DIRIGIDA HACIA LSO PADRES DE NIÑOS AFECTADOS[Tesis de Enfermeria], Universidad de Zaragoza.* Repositorio Institucional. Obtenido de https://zaguan.unizar.es/record/13558/files/TAZ-TFG-2014-049.pdf

Ambrosio, G., González, J., & Arévalo, M. (2004). La Libreria de Vision artficial OPENCV aplicaicon a la docencia e investigacion. *Dpto. De Ingeniería de Sistemas y Automática*, 6.

Ana, C. (2004). Programa de segmentación de regiones en imágenes médicas en MATLAB. *(Tesis en Licenciatura en Ingeniería en Electrónica y Comunicaciones).* Universidad de las Américas Puebla, Mexico.

Anonimo. (28 de 04 de 2022). *https://www.um.es*. Obtenido de https://www.um.es: https://www.um.es/geograf/sigmur/teledet/tema06.pdf

AprendeIA. (10 de 02 de 2023). *Aprendizaje Supervisado*. Obtenido de https://aprendeia.com/todo-sobre-aprendizaje-supervisado-en-machine-learning/

Arduino. (29 de 01 de 2023). *Arduino*. Obtenido de Arduino: https://store-usa.arduino.cc/products/nicla-vision

Arévalo, M., González, J., & Ambrosio, G. (14 de 01 de 2023). *http://mapir.isa.uma.es/:* . Obtenido de http://mapir.isa.uma.es/: : http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf

ASUSIOT. (29 de 01 de 2023). *ASUS IOT*. Obtenido de ASUS IOT: https://iot.asus.com/products/AI-accelerator/Coral-Dev-Board/

Blumenscheid, B. (15 de Mayo de 2022). *10 ejemplos de sistemas empotrados*. Obtenido de DIGI: https://es.digi.com/blog/post/examples-of-embedded-systems

Busquets, E. (17 de 03 de 2022). *VIaEmpresa*. Obtenido de ReHub: visión artificial aplicada a la rehabilitación: https://www.viaempresa.cat/es/empresa/rehub-vision-artificial-aplicada-rehabilitacion_2164940_102.html

Cervantes, O., & Gómez, F. (2012). *Taxonomía de los modelos y metodologías de desarrollo de software más utilizados*. Obtenido de Universidades: https://www.redalyc.org/articulo.oa?id=37326902005

Chambi, V. (2010). *DETECCION DE INCIDENCIA DE ALTERACIONES DE LA COLUMNA.* La Paz, Bolivia: Centrod e Investigacion.

Charaxes, J. (01 de 02 de 2023). *Fisify*. Obtenido de EcosistemaStarup: https://elreferente.es/startup/fisify/

COGNES Corporation. (2016). Introduccion a la Vision Artificial. *COGNEX, I*(1), 24.

Consejo Nacional de Planificacion. (2017). Plan Nacional Para el Buen Vivir 2017-2021. *Plan Nacional Para el Buen Vivir 2017-2021*, 47.

David, A., & Perez, A. (2009). Sistemas Embebidos y SIstemas Operativos Embebidos . *Lectura en Ciencias de la Computación* , 16.

Desarrolladores Web . (15 de 11 de 2022). *Digital Guide IONOS*. Obtenido de Digital Guide IONOS: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

Gallego, T. (2007). *Bases teóricas y fundamentos de la fisioterapia.* Buenos Aires: Médica Panamericana.

Gallo, C. (11 de 08 de 2021). Las otras dolencias que dejan los confinamientos y el teletrabajo. *Informacion COrona Virus*, pág. 1. Obtenido de France 24: https://www.france24.com/es/salud/20210210-pandemia-otras-enfermedades-confinamiento-teletrabajo

Garcia, I. (2008). *Vision Artificail y procesamiento Digital de Imagenes usando Matlab.* Ibarra: PUCE.

Gonzalez, R., & Woods, R. (2007). *Digital Image Processing* . Tennessee: Pearson.

Guevara, R. (2008). *Sentencias básicas usadas en la programación de computadoras.* Medellín: Textos Academicos.

Heath, S. (2003). *Embedded Systems Desing.* USA: Newnes.

Honesto, G. (s.f.). Sistema embebido para la segmetnacion de imagenes en tiempo real . *(Trabajo Terminal).* Escueal Superior de Computo , Mexico.

Hurwitz, J., & Kirsch, D. (2018). *Machine Learning For Dummies.* United States of America:: IBM Limited Edition.

José Francisco, V. S., Ana Belén, M. D., & Ángel Sánchez, C. (2016). *Visión por computador*. Madrid: Dykinson.

Katzman, W., & Wanek, L. (20 de 06 de 2010). *Age-Related Hyperkyphosis: Its Causes,*. Obtenido de Journal Orthopaedics Sports Physical: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2907357/

Leonidas, H., Navarro, O., Jorge Díaz, G., & Lizana, P. (2018). *Evaluación Postural y Prevalencia de Hipercifosis e Hiperlordosis en Estudiantes de Enseñanza Básica.* Arica-Chile: Int. J. Morphol.

Li, D., & Du, Y. (2016). *Artificial Intelligence with Uncertainty.* CRC Press.

Lyyn, t., & Staheli. (2003). *Ortopedia Pediatrica.* Marban.

Mehl, H., & Peinado, O. (06 de 12 de 2021). *Fundamentos del procesamientodigital de imágenes.* Obtenido de DocPlayer: https://docplayer.es/7230746-Fundamentos-del-procesamiento-digital-de-imagenes.html

Moe, J., & Winter, B. (2003). *DEFORMACIONES DE LA COLUMNA VERTEBRAL.* Madrid: Editorial Salvat.

Müller, A., & Guido, S. (2016). *Introduction to Machine Learning with Python.* O'Reilly Media.

Navarrete, H. (2010). Procesamiento Digital de Imagenes de Ultrasonido. *(Tesis de Ingenieria en comunicaciones y electronica).* Instituto Politecnico Nacional, Mexico.

NumPy. (30 de 01 de 2023). *numpy.org*. Obtenido de numpy.org: https://numpy.org/news/

OMS. (1 de 12 de 2020). *Organizacion Mundial de la salud* . Obtenido de Discapacidad y salud: https://www.who.int/es/news-room/fact-sheets/detail/disability-and-health

Pajankar, A. (2017). *Raspberry Pi Image Processing Programming.* Apress.

Pajares, G., & Garcia, J. (2007). *Ejercicios resueltos de Visi´on por Computador.* RA-MA .

Pastor, A., & Santoja, F. (10 de 08 de 2021). *Procedimientos Ortopedicos y de Traumatologia.* Obtenido de academiaedu: https://www.academia.edu/26679574/Cifosis_y_lordosis

Pedersen, S. (2007). Circular Hough Transform. *(Articulo Vision, Graphics, and Interactive Systems).* Aalborg University, Aalborg.

Petrou, M. :. (2010). *Image Processing: The Fundamentals.* London: John Wiley and Sons.

Pezonaga, S. (2015). Tratamiento de imagenes con ruido impulsivo medainte reglas difusas y algoritmos geneticos. *(Trabajo Fin de Grado en Ingenieria Informatica).* E.T:S de Ingenieria Indutrias, Informatica y de Telecomunicaciones, Pamplona.

Platero Dueñas, C. (2009). Apuntes de visión artificial. *Departamento de Electrónica, Automática*.

Priyam, D., & Dipanjan, S. :. (2016). Edge Detection by Using Canny and Prewitt. *International Journal of Scientific & Engineering Research*, 4.

Rafael, G., & Woods, R. (2002). *Digiral Image Processing.* United States of America: Prentice Hall.

Romero, V. (2019). *RECONOCIMIENTO DE IMÁGENES PARA DETECCIÓN TEMPRANA DE ALTERACIONES POSTULARES [Tesis Maestria en Automatizacion]*. Repositorio Institucional. Obtenido de https://repositorio.uta.edu.ec/bitstream/123456789/29664/1/Tesis_t1567masc.pdf

Santoja, f. (28 de 04 de 2022). *Las desviaciones sagitales del raquis y su relacion con la practica deportiva*. Obtenido de docplayers: https://docplayer.es/8504394-Capitulo-23-las-desviaciones-sagitales-del-raquis-y-su-relacion-con-la-practica-deportiva.html

Sight, M. (30 de 01 de 2023). *SimpleCV*. Obtenido de SimpleCV: http://simplecv.org

Torres, S. (2009). *CURSO DE INTRODUCCIÓN ALA INGENIERÍA DEL SOFTWARE.* Obtenido de Academia.edu: https://www.academia.edu/12390556/CURSO_DE_INTRODUCCIÓN_A_LA_INGENIERÍA_DEL_SOFTWARE_Laboratorio_Nacional_de_Calidad_del_Software

Trak. (30 de 01 de 2023). *TRAK*. Obtenido de TRAK: https://www.trakphysio.com/es/por-que-trak-la-aplicacion-de-fisioterapia/

Unidas, N. (2006). Convención sobre los derechos de las personas con discapacidad . *Convención sobre los derechos de las personas con discapacidad* (pág. 64). Nueva York: Naciones Unidas.

Valverde, j. (04 de 05 de 2007). https://www.researchgate.net. *(Articulo).* Universidad Nacional de Trujillo. Obtenido de Detección de bordes mediante el algoritmo de

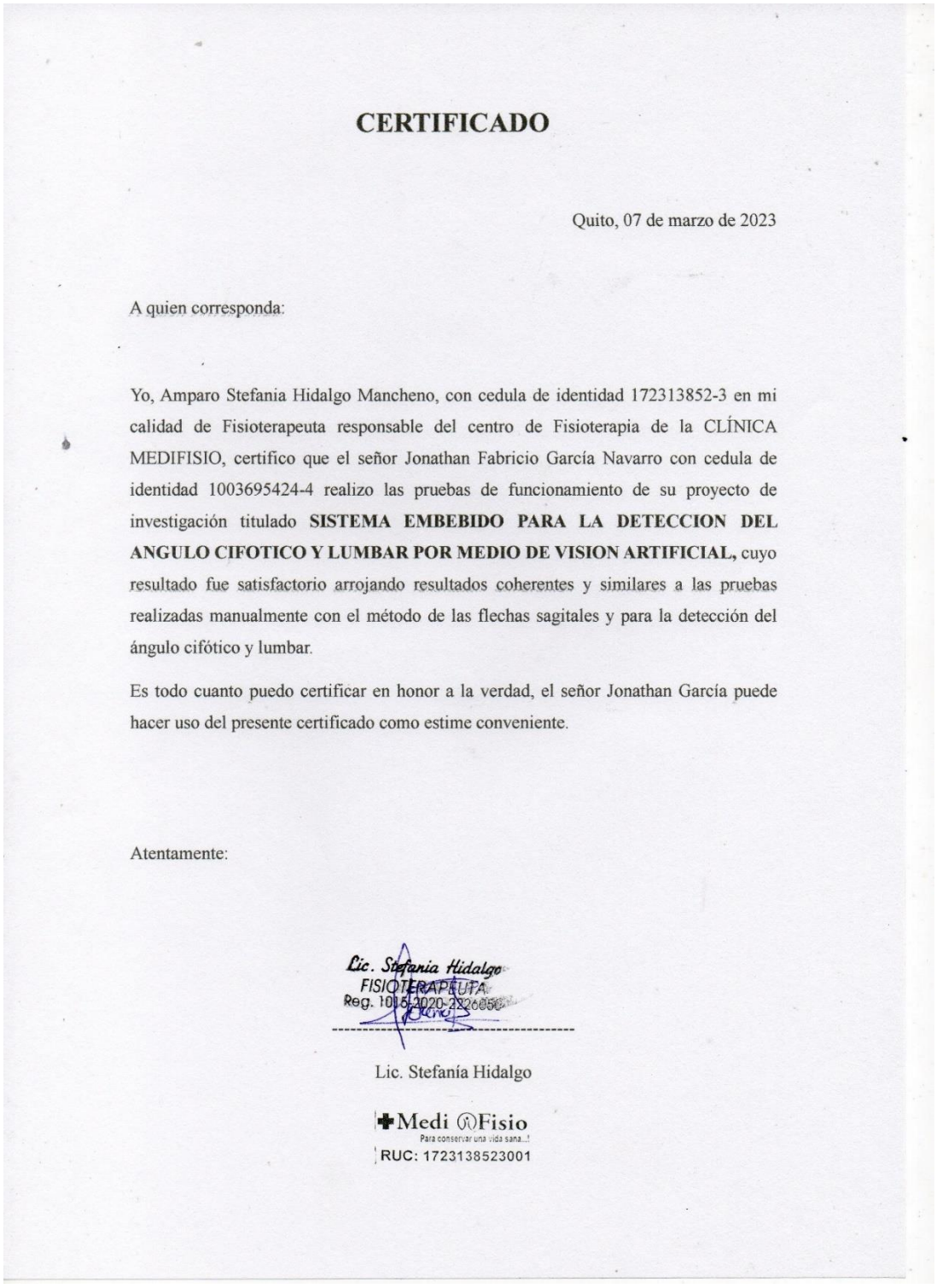
Canny: https://www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny

Vida.ip. (03 de 02 de 2021). *Vision Artificial en el Sector Medico*. Obtenido de Vida.ip: https://www.vidaip.es/vision-artificial-en-el-sector-medico/

Xu, D. (2010). *EMBEDDED VISUAL SYSTEM AND ITS APPLICATIONS ON ROBOTS.* P.R. China: Bentham Science Publishers.

Anexos

ANEXO A. *Certificación de pruebas de funcionamiento*



CERTIFICADO

Quito, 07 de marzo de 2023

A quien corresponda:

Yo, Amparo Stefania Hidalgo Mancheno, con cedula de identidad 172313852-3 en mi calidad de Fisioterapeuta responsable del centro de Fisioterapia de la CLÍNICA MEDIFISIO, certifico que el señor Jonathan Fabricio García Navarro con cedula de identidad 1003695424-4 realizo las pruebas de funcionamiento de su proyecto de investigación titulado **SISTEMA EMBEBIDO PARA LA DETECCION DEL ANGULO CIFOTICO Y LUMBAR POR MEDIO DE VISION ARTIFICIAL,** cuyo resultado fue satisfactorio arrojando resultados coherentes y similares a las pruebas realizadas manualmente con el método de las flechas sagitales y para la detección del ángulo cifótico y lumbar.

Es todo cuanto puedo certificar en honor a la verdad, el señor Jonathan García puede hacer uso del presente certificado como estime conveniente.

Atentamente:

Lic. Stefania Hidalgo
FISIOTERAPEUTA
Reg. 1016-2020-2226056

Lic. Stefanía Hidalgo

Medi Fisio
Para conservar una vida sana...!
RUC: 1723138523001

ANEXO B. *Pacientes del Centro de Fisioterapia*

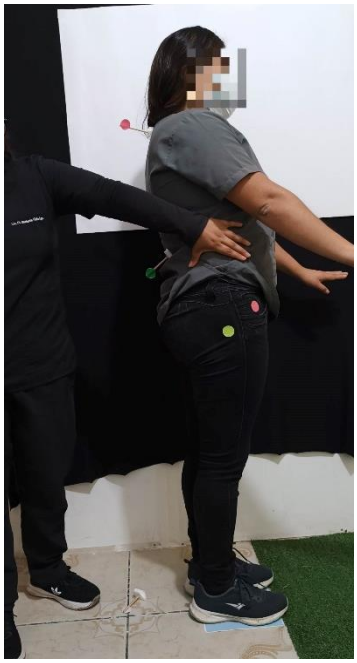


Ilustración 1. Medición ángulo Lumbar

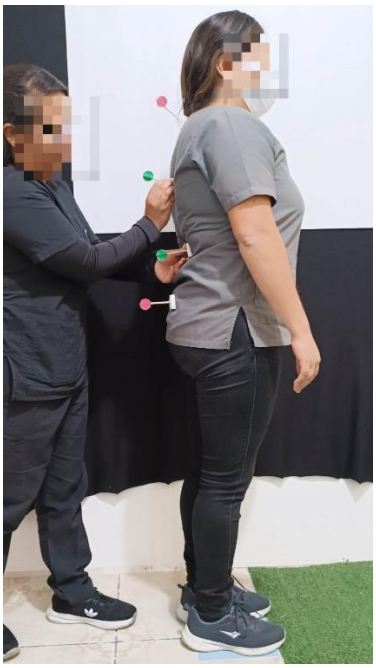


Ilustración 2. Medición flechas sagitales

Ilustración 3, Medición ángulo lumbar

Ilustración 4. Medición Flechas sagitales

ANEXO C. *Script Interfaz Principal*

```
import tkinter as tk
import cv2
import sqlite3
import math
import numpy as np
import imutils
import requests
import subprocess
import os
from tkinter import ttk
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from tkcalendar import DateEntry
from PIL import Image, ImageTk
from PIL import Image
from PIL import ImageTk
########################################################################################
class Database:
    def __init__(self, database_name):
        self.connection = sqlite3.connect(database_name)
        self.cursor = self.connection.cursor()

    def execute_query(self, query, params):
        self.cursor.execute(query, params)
        return self.cursor.fetchall()
    def __del__(self):
        self.connection.close()
########################################################################################
class VentanaPrincipal:
    db_name = 'database.db'
    def __init__(self, root):
        self.database = Database(self.db_name)
        self.root = root
        self.root.title("Cámara Web")
```

```
##########################################################################################
# Crear etiquetas para el nombre y apellido del usuario
label_nombre = tk.Label(root, text="Cedula:")
label_nombre.grid(row=2, column=0, padx=5, pady=5)
label_apellido = tk.Label(root, text="Apellido:")
label_apellido.grid(row=3, column=0, padx=5, pady=5)
label_cedula = tk.Label(root, text="Nombre:")
label_cedula.grid(row=4, column=0, padx=5, pady=5)
label_Methodos = tk.Label(root, text="Evaluacion")
label_Methodos.grid(row=7, column=0, columnspan=2, padx=5, pady=5, sticky = W + E)
label_peso = tk.Label(root, text="Peso (kg): ")
label_peso .grid(row=5, column=0, padx=5, pady=5, sticky = W + E)
label_altura = tk.Label(root, text="Altura (m): ")
label_altura.grid(row=6, column=0, padx=5, pady=5, sticky = W + E)
# Crear cuadros de entrada para el Peso y Altura
self.entry_peso = tk.Entry(root)
self.entry_peso.grid(row=5, column=1, padx=5, pady=5)
self.entry_altura = tk.Entry(root)
self.entry_altura.grid(row=6, column=1, padx=5, pady=5)
# Crear cuadros de entrada para cedula
self.entry_nombre = tk.Entry(root)
self.entry_nombre.focus()
self.entry_nombre.grid(row=2, column=1, padx=5, pady=5)
# Crear cuadros de Salida para el nombre y apellido
self.result_label_apellido = tk.Label(root)
self.result_label_apellido.grid(row=3, column=1, padx=5, pady=5)
self.result_label_cedula = tk.Label(root)
self.result_label_cedula.grid(row=4, column=1, padx=5, pady=5)
##########################################################################################################
# Crear botón para mostrar la cámara web
boton_camara = tk.Button(root, text="Angulo Toraxico", command=self.abrir_ventana_emergente)
boton_camara.grid(row=9, column=0, columnspan=1, padx=5, pady=5, sticky = W + E)
# Crear botón para mostrar la cámara web
boton_camara = tk.Button(root, text="Angulo Sacro", command=self.AnguloC)
boton_camara.grid(row=9, column=1, columnspan=1, padx=5, pady=5, sticky = W + E)
##########################################################################################
```

```
# Crear botón para mostrar consulta
boton_camara = tk.Button(root, text="Registro", command=self.Registro)
boton_camara.grid(row=0, column=0, columnspan=1, padx=5, pady=5, sticky = W + E)
# Agregar el botón para ejecutar la consulta
search_button = tk.Button(root, text="Buscar", command=self.get_results)
search_button.grid(row=1, column=1, columnspan=1, padx=5, pady=5, sticky = W + E)
# Crear botón para mostrar la consulta
boton_informacion = tk.Button(root, text="Informacion", command=self.Informacion)
boton_informacion.grid(row=1, column=0, columnspan=1, padx=5, pady=5, sticky = W + E)
# Agregar el botón para ejecutar la consulta
search_pacientes = tk.Button(root, text="Pacientes", command=self.Pacientes)
search_pacientes.grid(row=0, column=1, columnspan=1, padx=5, pady=5, sticky = W + E)
##############################################################################################
button_foto = tk.Button(root, text="Tomar foto", command=self.tomar_foto)
button_foto.grid(row=8, column=0, columnspan=1, padx=5, pady=5, sticky = W + E)
self.button_ver_foto = tk.Button(root, text="Ver foto", command=self.ver_foto, state="disabled")
self.button_ver_foto.grid(row=8, column=1, columnspan=1, padx=5, pady=5, sticky = W + E)
self.image_label = tk.Label(root)
self.image_label.grid(row=0, column=3, rowspan=11)
##############################################################################################
search_distancias = tk.Button(root, text="Distancias")
search_distancias.grid(row=10, column=0, columnspan=2, padx=5, pady=5, sticky = W + E)
##############################################################################################
# Agregar el botón para ejecutar el txt
search_pacientes = tk.Button(root, text="Resultados", command=self.mostrar_resultados)
search_pacientes.grid(row=11, column=0, columnspan=2, padx=5, pady=5, sticky = W + E)
#Labels de Resultado de Distancias
label_FC = tk.Label(root, text="Flecha C: ")
label_FC .grid(row=12, column=0, padx=5, pady=5, sticky = W + E)
label_FT = tk.Label(root, text="Flecha T: ")
label_FT .grid(row=13, column=0, padx=5, pady=5, sticky = W + E)
label_FL = tk.Label(root, text="Flecha L: ")
label_FL .grid(row=14, column=0, padx=5, pady=5, sticky = W + E)
label_FS = tk.Label(root, text="Flecha S: ")
label_FS .grid(row=15, column=0, padx=5, pady=5, sticky = W + E)
label_IC = tk.Label(root, text="Indice C: ")
```

```
        label_IC .grid(row=16, column=0, padx=5, pady=5, sticky = W + E)
        label_IL = tk.Label(root, text="Indice L: ")
        label_IL .grid(row=17, column=0, padx=5, pady=5, sticky = W + E)
        # Donde se visualizaran los resultados
        self.entry_FC = tk.Entry(root)
        self.entry_FC.grid(row=12, column=1, padx=5, pady=5)
        self.entry_FT = tk.Entry(root)
        self.entry_FT.grid(row=13, column=1, padx=5, pady=5)
        self.entry_FL = tk.Entry(root)
        self.entry_FL.grid(row=14, column=1, padx=5, pady=5)
        self.entry_FS = tk.Entry(root)
        self.entry_FS.grid(row=15, column=1, padx=5, pady=5)
        self.entry_IC = tk.Entry(root)
        self.entry_IC.grid(row=16, column=1, padx=5, pady=5)
        self.entry_IL = tk.Entry(root)
        self.entry_IL.grid(row=17, column=1, padx=5, pady=5)
        #Labels Angulos
        label_AC = tk.Label(root, text="Angulo C: ")
        label_AC .grid(row=18, column=0, padx=5, pady=5, sticky = W + E)
        label_AL = tk.Label(root, text="Angulo L: ")
        label_AL .grid(row=19, column=0, padx=5, pady=5, sticky = W + E)
        # Visualizar angulos
        self.entry_AC = tk.Entry(root)
        self.entry_AC.grid(row=18, column=1, padx=5, pady=5)
        self.entry_AL = tk.Entry(root)
        self.entry_AL.grid(row=19, column=1, padx=5, pady=5)
##########################################################################################################
    def Registro(self):
        exec(open("Registro.py").read())
    def Informacion(self):
        exec(open("Informacion.py").read())
    def Pacientes(self):
        exec(open("Pacientes.py").read())
    def AnguloC(self):
        exec(open("dist_nuevo.py").read())
##########################################################################################################
```

```
    def update_label(self, angulo):
        label_Angulo =tk.Label(root, text=f"Angulo Toraxico: ({angulo})").grid(row=7, column=0, padx=10, pady=10)
    def abrir_ventana_emergente(self):
        # Crear ventana emergente para mostrar la captura de la cámara web
        ventana = tk.Toplevel(self.root)
        ventana.title("Cámara Web")
        # Inicializar cámara web
        self.cap = cv2.VideoCapture(0)
        # Crear etiqueta para mostrar la captura de la cámara web
        self.label_camara = tk.Label(ventana)
        self.label_camara.pack()
        # Actualizar la captura de la cámara web en la etiqueta
        self.actualizar_imagen()
        # Cerrar la cámara web al cerrar la ventana emergente
        ventana.protocol("WM_DELETE_WINDOW", self.cerrar_camara)
    def actualizar_imagen(self):
        # Capturar imagen de la cámara web
        ret, frame = self.cap.read()
        #frame = imutils.resize(frame, width=720)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Aplicar filtro Gaussiano para reducir ruido
        gray = cv2.GaussianBlur(gray, (5, 5), 0)
        # Detectar círculos en la imagen
        circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 50, param1=50, param2=20, minRadius=10,
maxRadius=50)
        # Dibujar círculos encontrados y líneas entre centros
        if circles is not None:
            circles = np.round(circles[0, :]).astype("int")
            for (x, y, r) in circles:
                cv2.circle(frame, (x, y), r, (0, 255, 0), 2)
            if len(circles) == 3:
                circle_centers = circles[:, :2]
                for i in range(3):
                    cv2.line(frame, tuple(circle_centers[i]), tuple(circle_centers[(i+1)%3]), (255, 0, 0), 2)
                # Calcular ángulo formado entre el círculo más cercano al eje X y los círculos extremos
                x_dist = circle_centers[:, 0]
                idx_x = np.argmin(x_dist)
```

```
            idx_ext = [i for i in range(len(circle_centers)) if i != idx_x]
            p1 = circle_centers[idx_x]
            p2 = circle_centers[idx_ext[0]]
            p3 = circle_centers[idx_ext[1]]
            vec1 = p2 - p1
            vec2 = p3 - p1
            cos_angle = np.dot(vec1, vec2) / (np.linalg.norm(vec1) * np.linalg.norm(vec2))
            angle = int(math.degrees(math.acos(cos_angle)))

            # Dibujar resultado en el círculo del medio
            x_center = np.mean(circle_centers[:, 0]).astype("int")
            y_center = np.mean(circle_centers[:, 1]).astype("int")
            cv2.putText(frame, str(angle) + "°", (x_center-40, y_center+10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255),
2)
            self.update_label(angle)
        # Convertir imagen a formato compatible con Tkinter
        imagen = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        imagen = Image.fromarray(imagen)
        imagen_tk = ImageTk.PhotoImage(imagen)
        # Actualizar la etiqueta con la nueva imagen
        self.label_camara.config(image=imagen_tk)
        self.label_camara.image = imagen_tk
        # Programar la próxima actualización de la imagen
        self.root.after(10, self.actualizar_imagen)
    def cerrar_camara(self):
        self.cap.release()
        self.ventana.mainloop()
    def get_results(self):
        # Obtener el nombre introducido por el usuario
        nombre = self.entry_nombre.get()
        # Hacer la consulta por nombre y mostrar el apellido y la cédula
        result = self.database.execute_query("SELECT Apellidos, Nombres FROM registro WHERE Cedula = ?", (nombre,))
        # Borrar los resultados anteriores
        self.result_label_apellido.config(text="")
        self.result_label_cedula.config(text="")

        for i, row in enumerate(result):
```

```
            # Mostrar el apellido en la columna 0
            self.result_label_apellido.config(text=self.result_label_apellido.cget("text") + f"{row[0]}")
            # Mostrar la cédula en la columna 1
            self.result_label_cedula.config(text=self.result_label_cedula.cget("text") + f"{row[1]}")
    def tomar_foto(self):
        directory = r'/home/xheo/Desktop/Tesis/Imagenes'
        os.chdir(directory)
        nombre = self.entry_nombre.get()
        IMAGE_WIDTH = 1920
        IMAGE_HEIGHT = 1080
        # Capturar imagen de la cámara
        url = "http://192.168.1.3:8080/shot.jpg"
        #cap = cv2.VideoCapture(0)
        #ret, frame = cap.read()
        img_resp = requests.get(url)
        img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
        img = cv2.imdecode(img_arr, -1)
        # Guardar imagen
        filename = f"{nombre}.jpg"
        cv2.imwrite(filename, img)
        # Habilitar botón para ver la foto
        self.button_ver_foto.config(state="normal")
    def ver_foto(self):
        directory = r'/home/xheo/Desktop/Tesis/Imagenes'
        os.chdir(directory)
        # Mostrar imagen en el label correspondiente
        filename = f"{nombre}.jpg"
        image = Image.open(filename)
        image = image.resize((400, 400))
        photo = ImageTk.PhotoImage(image)
        self.image_label.configure(image=photo)
    def mostrar_resultados(self):
        with open('distancias.txt', 'r') as archivo:
            lineas = archivo.readlines()
            FC = float(lineas[0].split(": ")[1])
            FC = "{:.2f}".format(FC)
```

```
            self.entry_FC.delete(0, tk.END)
            self.entry_FC.insert(0, FC)
            FT = float(lineas[1].split(": ")[1])
            FT = "{:.2f}".format(FT)
            self.entry_FT.delete(0, tk.END)
            self.entry_FT.insert(0, FT)
            FL = float(lineas[2].split(": ")[1])
            FL = "{:.2f}".format(FL)
            self.entry_FL.delete(0, tk.END)
            self.entry_FL.insert(0, FL)
            FS = float(lineas[3].split(": ")[1])
            FS = "{:.2f}".format(FS)
            self.entry_FS.delete(0, tk.END)
            self.entry_FS.insert(0, FS)
            IC = float(lineas[4].split(": ")[1])
            self.entry_IC.delete(0, tk.END)
            self.entry_IC.insert(0, IC)
            IL = float(lineas[5].split(": ")[1])
            self.entry_IL.delete(0, tk.END)
            self.entry_IL.insert(0, IL)
        with open('AnguloC.txt', 'r') as archivo:
            lineas2 = archivo.readlines()
            AC = int(lineas2[0].split(": ")[1])
            self.entry_AC.delete(0, tk.END)
            self.entry_AC.insert(0, AC)
        with open('AnguloL.txt', 'r') as archivo:
            lineas3 = archivo.readlines()
            AL = int(lineas3[0].split(": ")[1])
            self.entry_AL.delete(0, tk.END)
if __name__ == "__main__":
    root = tk.Tk()
    VentanaPrincipal(root)
    root.mainloop()
```

ANEXO D. *Script Medición de Ángulos*

```
import cv2
import math
import os
from tkinter import filedialog
directory = r'/home/xheo/Desktop/Tesis/Imagenes'
os.chdir(directory)
file_path = filedialog.askopenfilename()
if file_path:
    # Cargar la imagen seleccionada utilizando OpenCV
    img = cv2.imread(file_path)

    # Redimensionamos la imagen a una resolución de 720p
    img = cv2.resize(img, (720, 480))

    # Tamaño de la cuadrícula
    gridSize = 20

    # Lista de puntos
    pointsList = []

    # Función para capturar los puntos del mouse
    def mousePoints(event, x, y, flags, params):
        if event == cv2.EVENT_LBUTTONDOWN:
            size = len(pointsList)
            if size != 0 and size % 3 != 0:
                cv2.line(img, tuple(pointsList[round((size-1)/3)*3]), (x,y),
(0,0,255), 2)
            cv2.circle(img, (x,y), 5, (0,0,255), cv2.FILLED)
            pointsList.append([x,y])

    # Función para calcular el ángulo entre dos líneas
    def gradient(pt1, pt2):
        return (pt2[1] - pt1[1]) / (pt2[0] - pt1[0])

    def getAngle(pointsList):
        pt1, pt2, pt3 = pointsList[-3:]
        m1 = gradient(pt1, pt2)
        m2 = gradient(pt1, pt3)
        angR = math.atan((m2 - m1) / (1 + (m2 * m1)))
        angD = round(math.degrees(angR))
        angD = abs(angD)
        cv2.putText(img, str(angD), (pt1[0] - 40, pt1[1] - 20),
cv2.FONT_HERSHEY_COMPLEX, 1.5, (0,0,255), 2)
        return (angD)

    # Bucle principal
    while True:
        # Si hay tres puntos, se calcula el ángulo
        if len(pointsList) % 3 == 0 and len(pointsList) != 0:
            ad = getAngle(pointsList)

        # Dibujar la cuadrícula
        for i in range(0, img.shape[1], gridSize):
            cv2.line(img, (i, 0), (i, img.shape[0]), (255, 255, 255), 1)
        for i in range(0, img.shape[0], gridSize):
            cv2.line(img, (0, i), (img.shape[1], i), (255, 255, 255), 1)

        # Mostramos la imagen
        cv2.imshow('Image', img)
        # Llamamos a la función del mouse
```

```
        cv2.setMouseCallback('Image', mousePoints)
        # Si se presiona la tecla 'q', se cierra la ventana
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
##########################################################################################
    direc = r'/home/xheo/Desktop/Tesis/Archivos de uso/circulos'
    os.chdir(direc)
    print('El Angulo es')
    numero_absoluto = abs(ad)
    print(numero_absoluto)
    # Guardar los resultados en un archivo de texto
    with open("AnguloC.txt", "w") as archivo:
        archivo.write("Angulo Cifotico: {}\n".format(numero_absoluto))
##########################################################################################
# Cerramos todas las ventanas
cv2.destroyAllWindows()
```

ANEXO E. *Medición Flechas Sagital*

```
import cv2
import numpy as np
import imutils
import requests

def clics(event,x,y,flags,param):
    global pun
    if event == cv2.EVENT_LBUTTONDOWN:
        pun.append([x,y])

def dibujando_puntos(pun):
    for x, y in pun:
        cv2.circle(frame,(x,y),5,(255,255,255),2)

def uniendo4puntos(pun):
    cv2.line(frame,tuple(pun[0]),tuple(pun[1]),(255,0,0),1)
    cv2.line(frame,tuple(pun[0]),tuple(pun[2]),(255,0,0),1)
    cv2.line(frame,tuple(pun[2]),tuple(pun[3]),(255,0,0),1)
    cv2.line(frame,tuple(pun[1]),tuple(pun[3]),(255,0,0),1)

def roi(image, ancho, alto):
    imagen_alineada = None
    global pun
    cv2.namedWindow('frame')
    cv2.setMouseCallback('frame',clics)
    dibujando_puntos(pun)
    if len(pun) == 4:
        pts1 = np.float32([pun])
        pts2 = np.float32([[0,0], [ancho,0], [0,alto], [ancho,alto]])
        M = cv2.getPerspectiveTransform(pts1, pts2)
        imagen_alineada = cv2.warpPerspective(image, M, (ancho,alto))

    return imagen_alineada

pun = []
#cap = cv2.VideoCapture(0)
url = "http://192.168.0.100:8080/shot.jpg"

while True:

    #ret, frame = cap.read()
    #if ret == False: break
    img_resp = requests.get(url)
    img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
    frame = cv2.imdecode(img_arr, -1)
    frame = imutils.resize(frame, width=720)
    imagen_A4 = roi(frame, ancho=720, alto=509)

    if imagen_A4 is not None:
        puntos = []
        puntosv = []
        imagenHSV = cv2.cvtColor(imagen_A4, cv2.COLOR_BGR2HSV)
        rojoBajo1 = np.array([0, 100, 20], np.uint8)
        rojoAlto1 = np.array([10, 255, 255], np.uint8)
        rojoBajo2 = np.array([175, 100, 20], np.uint8)
        rojoAlto2 = np.array([180, 255, 255], np.uint8)
        maskRojo1 = cv2.inRange(imagenHSV , rojoBajo1, rojoAlto1)
        maskRojo2 = cv2.inRange(imagenHSV , rojoBajo2, rojoAlto2)
        maskrojo =  cv2.add(maskRojo1, maskRojo2)
```

```
        cnts = cv2.findContours(maskrojo, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:2]
        for c in cnts:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(imagen_A4, (x, y), (x+w, y+h), (255, 0, 0), 2)
            puntos.append([x, y, w, h])
        if len(puntos) == 2:
            x1, y1, w1, h1 = puntos[0]
            x2, y2, w2, h2 = puntos[1]
            if x1 < x2:
                distancia_pixeles = abs(x1)
                distancia_cm1 = ((distancia_pixeles*100)/720)
                cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cm1),
(distancia_pixeles//2, y1), 2, 0.8, (0,0,255), 1, cv2.LINE_AA)
                cv2.line(imagen_A4,(0,y1),(x1, y1),(0, 0, 255),2)
            if x1 < x2:
                distancia_pixeles = abs(x2)
                distancia_cm2 = (distancia_pixeles*100)/720
                cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cm2),
(distancia_pixeles//2, y2), 2, 0.8, (0,0,255), 1, cv2.LINE_AA)
                cv2.line(imagen_A4,(0,y2),(x2, y2),(0, 0, 255),2)
##########################################################################################
        verdeBajo = np.array([35, 43, 35], np.uint8)
        verdeAlto = np.array([90, 255, 255], np.uint8)
        maskverde = cv2.inRange(imagenHSV, verdeBajo, verdeAlto)
        cntsv = cv2.findContours(maskverde, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
        cntsv = sorted(cntsv, key=cv2.contourArea, reverse=True)[:2]
        for cv in cntsv:
            xv, yv, wv, hv = cv2.boundingRect(cv)
            cv2.rectangle(imagen_A4, (xv, yv), (xv+wv, yv+hv), (0, 0, 255), 2)
            puntosv.append([xv, yv, wv, hv])
        if len(puntosv) == 2:
            x3, y3, w3, h3 = puntosv[0]
            x4, y4, w4, h4 = puntosv[1]
            if x3 < x4:
                distancia_pixelesv = abs(x3)
                distancia_cmv1 = (distancia_pixelesv*100)/720
                cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cmv1),
(distancia_pixelesv//2, y3), 2, 0.8, (0,0,255), 1, cv2.LINE_AA)
                cv2.line(imagen_A4,(0,y3),(x3, y3),(0, 0, 255),2)
            if x3 < x4:
                distancia_pixelesv = abs(x4)
                distancia_cmv2 = (distancia_pixelesv*100)/720
                cv2.putText(imagen_A4, "{:.2f} cm".format(distancia_cmv2),
(distancia_pixelesv//2, y4), 2, 0.8, (0,0,255), 1, cv2.LINE_AA)
                cv2.line(imagen_A4,(0,y4),(x4, y4),(0, 0, 255),2)

##########################################################################################
        cv2.imshow('imagen_A4',imagen_A4)

    cv2.imshow('frame',frame)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
print('Distancias')
print('dist Toraxica')
print(distancia_cm1)
print('dist Cervical')
print(distancia_cm2)
print('dist Sacra')
```

```
print(distancia_cmv1)
print('dist Lumbar')
print(distancia_cmv2)

if distancia_cm1 < distancia_cmv1:
    print('Distancias Si FT = 0')
    FlechaT = 0
    FlechaC = (distancia_cm2 - distancia_cm1) * 10
    FlechaL = (distancia_cmv2 - distancia_cm1) * 10
    FlechaS = (distancia_cmv1 - distancia_cm1) * 10
    print('Flecha Cervical')
    print(FlechaC)
    print('Flecha Toraxica')
    print(FlechaT)
    print('Flecha Lumbar')
    print(FlechaL)
    print('Flecha Sacra')
    print(FlechaS)
    print('Indice Cifotico')
    IndC = ((FlechaC+FlechaL)/2)
    IndC = "{:.2f}".format(IndC)
    print(IndC)
    print('Indice lordotico')
    IndL = (FlechaL-((1/2)*FlechaS))
    IndL = "{:.2f}".format(IndL)
    print(IndL)
    # Guardar los resultados en un archivo de texto
    with open("distancias.txt", "w") as archivo:
        archivo.write("Flecha Cervical: {}\n".format(FlechaC))
        archivo.write("Flecha Toraxica: {}\n".format(FlechaT))
        archivo.write("Flecha Lumbar: {}\n".format(FlechaL))
        archivo.write("Flecha Sacra: {}\n".format(FlechaS))
        archivo.write("Indice Cifotico: {}\n".format(IndC))
        archivo.write("Indice lordotico: {}\n".format(IndL))
else:
    print('Distancias Si FS = 0')
    FlechaS = 0
    FlechaC = (distancia_cm2 - distancia_cmv1) * 10
    FlechaT = (distancia_cm1 - distancia_cmv1) * 10
    FlechaL = (distancia_cmv2 - distancia_cmv1) * 10
    print('Flecha Cervical')
    print(FlechaC)
    print('Flecha Toraxica')
    print(FlechaT)
    print('Flecha Lumbar')
    print(FlechaL)
    print('Flecha Sacra')
    print(FlechaS)
    print('Indice Cifotico')
    IndC = (((FlechaC+FlechaL+FlechaS)/2)-FlechaT)
    IndC = "{:.2f}".format(IndC)
    print(IndC)
    print('Indice lordotico')
    IndL = (FlechaL-((1/2)*FlechaT))
    IndL = "{:.2f}".format(IndL)
    print(IndL)
    # Guardar los resultados en un archivo de texto
    with open("distancias.txt", "w") as archivo:
        archivo.write("Flecha Cervical: {}\n".format(FlechaC))
        archivo.write("Flecha Toraxica: {}\n".format(FlechaT))
        archivo.write("Flecha Lumbar: {}\n".format(FlechaL))
        archivo.write("Flecha Sacra: {}\n".format(FlechaS))
```

```
        archivo.write("Indice Cifotico: {}\n".format(IndC))
        archivo.write("Indice lordotico: {}\n".format(IndL))


cap.release()
cv2.destroyAllWindows()
```

***ANEXO F**. Script Registro Pacientes*

```
from tkinter import ttk
from tkinter import *
from tkinter import filedialog
from PIL import Image
from PIL import ImageTk
from tkinter import messagebox
from tkcalendar import DateEntry
import cv2
import imutils
import numpy as np
import sqlite3
import math

class Registro:
    # connection dir property
    db_name = 'database.db'

    def __init__(self, window):
        # Initializations
        self.wind = window
        self.wind.title('Registro Paciente')
        #Validacion de datos
        self.definir_patrones_validaciones()
        # Creating a Frame Container
        self.frame = LabelFrame(self.wind, text = 'Datos Generales del Paciente')
        self.frame.grid(row = 0, column = 1, columnspan = 3, pady = 0)

        # Nombres Input
        Label(self.frame, text = 'Nombres: ').grid(row = 1, column = 0, pady = 5)
        self.name = Entry(self.frame, width = 25)
        self.name.focus()
        self.name.grid(row = 1, column = 1)
        # Aplellidos Input
        Label(self.frame, text = 'Apellidos: ').grid(row = 2, column = 0, pady = 5)
        self.apellido = Entry(self.frame, width = 25)
        self.apellido.grid(row = 2, column = 1)
        # Cedula Input
        Label(self.frame, text = 'Cedula: ').grid(row = 3, column = 0, pady = 5)
        self.cedula = Entry(self.frame, width = 25)
        self.cedula.grid(row = 3, column = 1)

        # Fecha Input
        Label(self.frame, text = 'Fecha: ').grid(row = 4, column = 0, pady = 5)
        self.fecha = Entry(self.frame, width = 25)
        self.fecha.grid(row = 4, column = 1)

        # Direccion Input
        Label(self.frame, text = 'Direccion: ').grid(row = 5, column = 0, pady = 5)
        self.direccion = Entry(self.frame, width = 25)
        self.direccion.grid(row = 5, column = 1)
        # Edad Input
        Label(self.frame, text = 'Edad: ').grid(row = 6, column = 0, pady = 5)
        self.edad = Entry(self.frame, width = 25)
        self.edad.grid(row = 6, column = 1)
        # Sexo Input
        Label(self.frame, text = 'Sexo: ').grid(row = 7, column = 0, pady = 5)
        self.sexo = ttk.Combobox(self.frame, width = 22)
        genero = (' Masculino   ' , ' Femenino   ')
        self.sexo['values'] = genero
        self.sexo.current(0)
```

```
        self.sexo.grid(row = 7, column = 1, sticky = W + E)
        # Observacion Input
        Label(self.frame, text = 'Observaciones: ',).grid(row = 8, column = 0, pady =
5)
        self.observacion = Entry(self.frame, width = 25)
        self.observacion.grid(row = 8, column = 1)
        # Button Add Reg
        ttk.Button(self.frame, text = 'Registro', command = self.add_pacient).grid(row
= 9, columnspan = 2, sticky = W + E)
        ttk.Button(self.frame, text='Validacion', command = self.guardar).grid(row =
10, columnspan = 2, sticky = W + E)
        #Output Messages
        self.message = Label(self.frame, text = '', fg = 'red')
        self.message.grid(row = 11, column = 0, columnspan = 2, sticky = W + E)
    # Function to Execute Database Querys
    def run_query(self, query, parameters = ()):
        with sqlite3.connect(self.db_name) as conn:
            cursor = conn.cursor()
            result = cursor.execute(query, parameters)
            conn.commit()
        return result
    # User Input Validation
    def validation(self):
        return len(self.name.get()) != 0 and len(self.apellido.get()) != 0 and
len(self.cedula.get()) != 0 and len(self.fecha.get()) != 0 and
len(self.direccion.get()) != 0 and len(self.edad.get()) != 0 and len(self.sexo.get())
!= 0 and len(self.observacion.get()) != 0

    def definir_patrones_validaciones(self):
        #Nombres
        patron_nombre =  r'^[a-zA-Z]+( [a-zA-Z]+)+$'
        self.regex_nombre = re.compile(patron_nombre)
        #Apellidos
        patron_apellidos = r'^[a-zA-Z]+( [a-zA-Z]+)+$'
        self.regex_apellidos = re.compile(patron_apellidos)
        #Cedula
        patron_cedulav = r'^[0-9]{10,13}$'
        self.regex_cedulav = re.compile(patron_cedulav)
        #Fecha
        patron_fechav = r'^(?:3[01]|[12][0-9]|0?[1-9])([\-/.])(0?[1-9]|1[1-
2])\1\d{4}$'
        self.regex_fechav = re.compile(patron_fechav)
        #Direccion
        patron_direccionv = r'^[^\s]+( [^\s]+)+$'
        self.regex_direccionv = re.compile(patron_direccionv)
        #Edad
        patron_edadv = r'^[0-9]{1,2}$'
        self.regex_edadv = re.compile(patron_edadv)
        #Observaciones
        patron_obs = r'^[^\s]+( [^\s]+)+$'
        self.regex_obs= re.compile(patron_obs)
    def guardar(self):
        pass
        #Nombre
        nombre = self.name.get().strip()
        if re.match(self.regex_nombre, nombre) is None:
            #messagebox.showwarning('Mensaje', 'El campo Nombre debe incluir los
Nombre del Paciente')
            self.message['text'] = 'Campo: Nombres is Required'
            return
        #Apellido
        apellidos = self.apellido.get().strip()
```

```
        if re.match(self.regex_apellidos, apellidos) is None:
            #messagebox.showwarning('Mensaje', 'El campo Apellido debe incluir los
Apellidos del Paciente')
            self.message['text'] = 'Campo: Apellidos is Required'
            return
        #Cedula
        cedulav = self.cedula.get().strip()
        if re.match(self.regex_cedulav, cedulav) is None:
            #messagebox.showwarning('Mensaje', 'El campo Cedula debe tener de 10 a 13
caracteres numericos')
            self.message['text'] = 'Campo: Cedula is Required'
            return
        #Fecha
        fechav = self.fecha.get().strip()
        if re.match(self.regex_fechav, fechav) is None:
            #messagebox.showwarning('Mensaje', 'El campo Fecha debe tener el formato
dd/mm/aaaa')
            self.message['text'] = 'Campo: Fecha is Required formato dd/mm/aaaa'
            return
        #Direccion
        direccionv = self.direccion.get().strip()
        if re.match(self.regex_direccionv, direccionv) is None:
            #messagebox.showwarning('Mensaje', 'El campo Direccion no es correcto ')
            self.message['text'] = 'Campo: Direccion is Required'
            return
        #Edad
        edadv = self.edad.get().strip()
        if re.match(self.regex_edadv, edadv) is None:
            #messagebox.showwarning('Mensaje', 'El campo Edad no es correcto')
            self.message['text'] = 'Campo: Edad is Required'
            return
        #Observaciones
        observacionv = self.observacion.get().strip()
        if re.match(self.regex_obs, observacionv) is None:
            #messagebox.showwarning('Mensaje', 'El campo Observacion no es correcto ')
            self.message['text'] = 'Campo: Observacion is Required'
            return
    # Function add pacient in Database
    def add_pacient(self):
        if self.validation():
            query = 'INSERT INTO registro VALUES(NULL, ?, ?, ?, ?, ?, ?, ?, ?)'
            parameters =  (self.name.get(), self.apellido.get(), self.cedula.get(),
self.fecha.get(), self.direccion.get(), self.edad.get(), self.sexo.get(),
self.observacion.get())
            self.run_query(query, parameters)
            self.message['text'] = 'Paciente {} added
Successfully'.format(self.name.get())
            self.name.delete(0, END)
            self.apellido.delete(0, END)
            self.cedula.delete(0, END)
            self.fecha.delete(0, END)
            self.direccion.delete(0, END)
            self.edad.delete(0, END)
            self.sexo.delete(0, END)
            self.observacion.delete(0, END)
        else:
            self.message['text'] = 'Datos is Required'

if __name__ == '__main__':
    window = Tk()
    application = Registro(window)
    window.mainloop()
```