UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

PROGRAMA DE TITULACIÓN POR TESIS

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

PROTOTIPO DE TRADUCTOR DE VOZ A LENGUAJE DE SEÑAS PERUANA MEDIANTE ANIMACIÓN TRIDIMENSIONAL PARA INDICACIONES DE POSICIONAMIENTO DEL PACIENTE DURANTE LA TOMA DE RADIOGRAFÍA DE TÓRAX

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADA POR

Bach. MEDRANO LÓPEZ, DAVID JOSEPH

Bach. CONDORI OROSCO, GERSON JACOB

Asesor: Dr. Ing. HUAMANÍ NAVARRETE, PEDRO FREDDY

LIMA-PERÚ

2021

DEDICATORIA

A mis padres David e Ysabel y hermana Najhu, quienes con su amor, paciencia y esfuerzo me han permitido llegar a cumplir hoy un sueño más. Gracias por inculcar en mí el ejemplo de esfuerzo y valentía, de no temer las adversidades porque Dios está conmigo siempre.

David Medrano.

A mi familia, a mis queridos tíos: Juana y Modesto, por forjar en mí la sencillez, el amor y la solidaridad por la familia en una etapa de mi vida y a mi madre María, por su apoyo incondicional, por confiar en mí y llevar su espíritu luchador y ser mi ejemplo a seguir. Y a mi hijo Adriano que es mi motivación de continuar y mejorar como padre día a día.

Gerson Condori.

AGRADECIMIENTO

Agradecemos a nuestros padres, por el apoyo incondicional que siempre nos brindaron, a nuestra Universidad Ricardo Palma, por formar las bases de nuestros conocimientos.

David Medrano, Gerson Condori

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El prototipo de traductor de voz a lengua de señas peruana mediante animación tridimensional, propuesto en este trabajo de tesis, surge a raíz de la pandemia generada por el virus SARS-COV-2, ya que, la necesidad de tomar radiografías de tórax se incrementó considerablemente, dejando al descubierto un déficit en inclusión para las personas sordas en los hospitales, debido a la ausencia de intérpretes calificados que puedan traducir las indicaciones del tecnólogo. Así mismo, el desarrollo del prototipo está conformado por cinco etapas, la primera etapa enfocada en la adquisición de la voz y el diseño de un filtro digital en Matlab que atenúe los ruidos en las salas de radiografías, la segunda etapa se centró en la conversión de la voz a texto usando el servicio de IBM, Watson Speech to Text, en la tercera etapa se implementó un algoritmo en lenguaje Python para la correcta verificación de las conversiones previamente realizadas, una cuarta etapa desarrollada en el software DAZ Studio, donde se realizó la caracterización de un avatar y animación tridimensional de la Lengua de Señas Peruana (LSP), con frases enfocadas en la toma de radiografías de tórax y, como última fase, la integración de todas las etapas mencionadas en lenguaje Python. Finalmente, se realizaron las pruebas del funcionamiento del proyecto dentro de la sala de radiografía del Hospital Nacional Hipólito Unanue con apoyo de 04 tecnólogos, dos varones y dos mujeres, de los cuales se obtuvo un porcentaje máximo de efectividad del 100% en la conversión de voz a texto, para frases como “levante los brazos” y “se puede retirar”, y un porcentaje mínimo de efectividad de 87.17% y 87.59% para frases como “quítese las prendas de la parte superior” y “buenas noches” respectivamente.

Palabras claves: Filtro digital en Matlab, Watson Speech to text, algoritmo en Python, traductor de voz a texto, DAZ Studio, radiografías de tórax.

ABSTRACT

The prototype of a Peruvian voice-to-sign language translator using three-dimensional animation, proposed in this thesis work, arose as a result of the SARS-COV-2 pandemic, since the need to take chest X-rays increased considerably, revealing a lack of inclusion for deaf people in hospitals, due to the absence of qualified interpreters capable of translating the technologist's instructions. Likewise, the development of the prototype is made up of five phases, the first phase focused on the acquisition of voice and the design of a digital filter in Matlab to reduce noise in the X-ray rooms, the second phase focused on the conversion of voice to text using IBM's Watson Speech to Text service, and the third phase implemented an algorithm. Python language rhythms for the correct verification of previous conversions, a fourth stage developed in the DAZ Studio software, where the characterization of an avatar and three-dimensional animation of the Peruvian Sign Language (LSP) was performed, with phrases focused on the taking of chest X-rays and, as a final stage, the integration of all the steps mentioned in Python language. Finally, tests of the operation of the project were carried out in the X-ray room of the Hipólito Unanue National Hospital with the support of 04 technologists, two men and two women, from whom a maximum percentage of 100% effectiveness was obtained in the conversion from voice to text, for phrases such as "raise your arms" and "can be removed", and a minimum percentage of effectiveness was obtained. 87. 17% and 87. 59% for phrases such as "remove your upper garments" and "good night" respectively.

Keywords: Digital filter in Matlab, Watson Speech to text, Python algorithm, speech to text translator, DAZ Studio, chest X-rays.

INTRODUCCIÓN

En la actualidad, gran parte de la población peruana, ignora la mayoría de problemas a los que se enfrentan las personas sordas que tienen como lengua materna la Lengua de Señas Peruana (LSP). Uno de estos problemas, es la exclusión del derecho a la salud, debido a la nula capacidad de los centros de salud al momento de comunicarse con estas personas, ya que, no cuentan con personal capacitado en Lengua de Señas Peruana. Así mismo, con la pandemia generada por el SARS-COV-2, se generó un aumento considerable en la toma de Rayos X de tórax, para poder diagnosticar posibles neumonías. Razón por la cual, los tecnólogos encargados, necesitan dar indicaciones de posicionamiento a los pacientes, para poder obtener una radiografía de tórax adecuada. Debido a esto, las personas sordas, necesitan ir acompañadas por una persona que pueda traducir las indicaciones sonoras dadas por el tecnólogo a la Lengua de Señas Peruana, condicionando la atención médica de dichos pacientes.

Es por esto que, se planteó el desarrollo de un prototipo capaz de traducir las indicaciones habladas por el tecnólogo a la Lengua de Señas Peruana, a través de animaciones tridimensionales. El prototipo cuenta con una etapa de filtrado de la voz desarrollado en Matlab e implementado en Python, seguido del servicio Watson Speech to Text para la conversión de voz a texto también implementado en Python, el cual pasa por un algoritmo para confirmar la correcta conversión del mismo y, finalmente, la etapa de animación tridimensional del avatar y la Lengua de Señas Peruana.

Así mismo, en el capítulo I se muestra el desarrollo del planteamiento y delimitación del problema, los objetivos, la importancia y justificación de la tesis. El capítulo II consta del marco teórico, dividido en investigaciones relacionadas con el tema, bases teóricas, definición de términos básicos y diseño de la investigación. El capítulo III muestra el desarrollo detallado del proyecto desde la etapa inicial conformado por la adquisición de la voz, hasta la etapa final que es la integración de todas las etapas en lenguaje Python. Finalmente, en el capítulo IV, se muestran las pruebas y resultados obtenidos con las voces reales de los tecnólogos dentro de la sala de rayos X del Hospital Nacional Hipólito Unanue.

CAPÍTULO I: PLANTEAMIENTO Y DELIMITACIÓN DEL PROBLEMA

1.1. Formulación del problema

Mayanga, Guerra, Alcides y Pastor (2020) señalan que, por la pandemia ocasionada por el SARS-COV-2, se ha generado un incremento de los casos y las necesidades de requerir los servicios de salud, los cuales no se abastecen debido a la sobre demanda de pacientes clínicamente graves y otros, con enfermedades no graves con presencia o no de factores de riesgo, que requieren atención médica y de un diagnóstico por imágenes ante la sospecha de neumonía. Así mismo, según el Censo Nacional de Población (INEI 2017), en el Perú hay más de 230 000 personas con dificultad para oír, de las cuales cerca de 9000 personas registraron la Lengua de Señas Peruana como lengua materna; así mismo, la Defensoría del Pueblo (2020) afirma recibir constantemente quejas y petitorios referidos a las barreras que enfrentan las personas con esta discapacidad en los servicios educativos, de salud, entre otros, debido a la falta de docentes e intérpretes calificados en Lenguaje de Señas Peruana. Razón por la cual, el personal de salud encargado de la toma de radiografías se encuentra con una barrera lingüística que ralentiza su función, debido a que necesita esforzarse para darle indicaciones, muchas veces sin éxito, a los pacientes que tienen el lenguaje de señas peruana como lengua materna. Tales indicaciones corresponden al correcto posicionamiento que éstos deben adoptar para poder realizar la toma de radiografía requerida; por ello, estos pacientes necesitan ir acompañados por un intérprete para poder ser atendidos. Sin embargo, esto condiciona la toma de radiografías torácicas de estas personas, ya que, los centros de salud no cuentan con personal capacitado en lenguaje de señas peruana, fomentando la exclusión social de personas con esta discapacidad auditiva, además de privarlos de un servicio básico como la asistencia médica.

1.1.1. Problema General

¿Cómo realizar un prototipo de traductor de voz a Lenguaje de Señas Peruana mediante animación tridimensional para dar indicaciones de posicionamiento del paciente durante la toma de radiografías de tórax?

1.1.2. Problemas Específicos

a) ¿Es posible diseñar un filtro digital en Matlab, e implementarlo en lenguaje Python, para atenuar el ruido generado en la sala de radiografías del Hospital Nacional Hipólito Unanue durante la adquisición de la voz?

b) ¿Cómo convertir la voz, previamente filtrada, en texto a través del servicio IBM Watson Speech to Text, e implementar un algoritmo para verificar la correcta conversión de voz a texto?

c) ¿De qué manera diseñar las animaciones tridimensionales de la Lengua de Señas Peruana, de acuerdo a la previa conversión de voz a texto utilizando el software DAZ Studio, para mostrar indicaciones visuales de posicionamiento al paciente durante la toma de la radiografía de tórax?

1.2. Objetivos

1.2.1. Objetivo General

Implementar un prototipo de traductor de voz a Lenguaje de Señas Peruana, mediante animación tridimensional para dar indicaciones de posicionamiento del paciente durante la toma de radiografías de tórax.

1.2.2. Objetivos Específicos

a) Diseñar un filtro digital en Matlab e implementarlo en lenguaje Python para atenuar el ruido generado en la sala de radiografías del hospital Hipólito Unánue, durante la adquisición de la voz.

b) Convertir la voz, previamente filtrada, en texto a través del servicio IBM Watson Speech to Text, e implementar un algoritmo para verificar la correcta conversión de voz a texto.

c) Diseñar las animaciones tridimensionales de la Lengua de Señas Peruana, de acuerdo a la previa conversión de voz a texto, utilizando el software DAZ Studio, para mostrar indicaciones visuales de posicionamiento al paciente durante la toma de la radiografía de tórax.

1.3. Importancia y justificación

El SARS-COV-2, ha generado un incremento notable del requerimiento de atención médica y, la necesidad de un diagnóstico por imágenes ante la sospecha de neumonía, dando a evidenciar la falta de intérpretes y personal capacitado que pueda atender a personas que tienen a la Lengua de Señas Peruana como lengua materna, generando que sean excluidos de los servicios de salud, en especial de la toma de radiografías, las cuales son importantes durante esta pandemia. Por tal motivo, es necesario el desarrollo de un prototipo traductor de voz a Lengua de Señas Peruana mediante animación tridimensional, para dar indicaciones de posicionamiento al paciente durante la toma de radiografías de tórax. Este prototipo tiene una etapa de filtrado de la señal de voz, una etapa para el reconocimiento y conversión de voz a texto a través de IBM Watson Speech to Text, otra etapa para la verificación de la conversión de voz a texto generado y finalmente una etapa de ejecución de la Lengua de Señas Peruana a través de una animación en 3D, realizado en el software DAZ Studio. Así mismo, debido a la experiencia en diseño de filtros digitales a través de Matlab, los conocimientos de programación en lenguaje Python, el uso gratuito e intuitivo del software DAZ Studio, y el ingreso autorizado que se tiene a la sala de Rayos X del Hospital Nacional Hipólito Unanue, este proyecto se justifica y es perfectamente realizable.

1.4. Limitaciones

El trabajo de tesis se centra solamente en indicaciones de comunicación de tecnólogo a paciente, utilizando un número de palabras suficientes que conforma la base de datos. Así mismo, es aplicado para el laboratorio de Rayos X del Hospital Nacional Hipólito Unanue. Además, se centra solamente en pacientes con sordera, que tengan a la Lengua de Señas Peruana como lengua materna, que no tengan algún impedimento físico para caminar, mantenerse en pie, o algún otro que requiera la ayuda de otra persona para poder realizarlo, así mismo, se centra en personas no gestantes y personas mayores de edad que puedan asistir solas al centro médico. Adicionalmente, las indicaciones del tecnólogo están enfocadas únicamente a la toma de radiografías torácicas, por un lapso de tiempo no mayor a 5 minutos para la adquisición y procesamiento de la voz, conversión de voz a texto, verificación de conversión y presentación de la animación tridimensional.

CAPÍTULO II: MARCO TEÓRICO

2.1. Marco histórico

Según la Ley General de Personas con Discapacidad N° 29973 (2020), en el Perú mediante sus artículos establecen que toda persona con discapacidad tiene derecho a utilizar la lengua de señas, el sistema braille y otros formatos o medios aumentativos o alternativos de comunicación en los procesos judiciales y en los procedimientos administrativos que siga ante la administración pública y los proveedores de servicios públicos. Para tal fin, dichas entidades proveen a la persona con discapacidad, de manera gratuita y en forma progresiva, el servicio de intérprete cuando ésta lo requiera. Asimismo, el Ministerio de Salud y los gobiernos regionales garantizan la disponibilidad y el acceso de la persona con discapacidad a medicamentos de calidad, tecnologías de apoyo, dispositivos y la ayuda compensatoria necesaria para su atención y rehabilitación, tomando en cuenta su condición socioeconómica, geográfica y cultural. Sin embargo, en la realidad que vivimos estos servicios de intérprete y tecnologías de apoyo, no se implementaron debido a la falta de iniciativa por parte del Ministerio de salud. Por ello se pretende realizar la implementación y optimización de este prototipo en los establecimientos de salud como hospitales y centros de salud para personas con este tipo de discapacidad. Además, BBC News Mundo (2020). El traductor de Google, señala que ha dado pasos gigantes desde que saliera al mercado en 2006. Tras pasar por una etapa en la que se dedicaba a buscar el equivalente en otra lengua de una frase determinada, en 2016 el gigante tecnológico introdujo una nueva forma de operar: la traducción automática neuronal. El resultado son textos mucho más fluidos que tienen en cuenta el significado global del texto en vez de traducir palabra por palabra. A esto se suman mejoras que en los últimos años han permitido que se pueda traducir también sonidos y documentos utilizando el micrófono y la cámara del celular.

2.2. Investigaciones relacionadas con el Tema

Según Carguacundo, M., Constante, P. (2018), en la tesis titulada: Traductor de texto y voz a lengua de señas ecuatoriana a través de un avatar implementado para dispositivos Android. El objetivo de este sistema traductor es visualizar el avatar con las respectivas señales con cada uno de sus movimientos de acuerdo con la

palabra ingresada ya sea por el teclado o por voz mediante reconocimiento por Google Now en un dispositivo móvil, para esto se establece las especificaciones técnicas requeridas por las necesidades del usuario, los cuales son: La función del sistema es visualizar el avatar con la señal de las manos de acuerdo con cada palabra. Las dimensiones del avatar deben ser proporcionales al cuerpo humano. El avatar debe de mover la parte superior del cuerpo cabeza, cuello, brazos y manos. El sistema es compatible con dispositivos Android, ya que fue desarrollado en Android Studio. En conclusión, se comprobó que cada palabra obtenida del motor de reconocimiento de voz se compare con una letra de la base de datos y se obtenga la señal respectiva. Para esta prueba se obtuvo un total de 170 palabras. Los resultados de la prueba de voz entregaron un reconocimiento de 162 palabras mientras que 8 palabras no fueron reconocidas, con estos datos se calcula el error que es el 5% y un 95% de eficiencia.

Bernall, S., Páez, J., (2012) en la tesis titulada: Sistema inteligente de reconocimiento de voz para la traducción del lenguaje verbal a la lengua de señas colombiana. El Sistema Inteligente de reconocimiento de voz para la traducción del lenguaje verbal a la lengua de señas colombiana (VLSC), surge a partir de la necesidad de inclusión de estudiantes sordos a la formación en la educación superior, como es el caso de la Universidad Pedagógica Nacional, donde se brinda el apoyo de un intérprete para facilitar la comunicación con los docentes y compañeros de clase y así facilitar el proceso de enseñanza aprendizaje. Se concluye que el objetivo es diseñar un sistema inteligente que reconozca los patrones de voz del locutor y los traduce la lengua de señas colombiana por medio de una interfaz gráfica compuesta de la representación de la seña, apoyada de la imagen de deletreo con su respectivo texto, dicho sistema se denomina SISTEMA VLSC, ya que se integra el reconocimiento de voz y la lengua de señas colombiana. El presente diseño del sistema utiliza la implementación de redes neuronales aplicadas al reconocimiento de patrones de voz desarrollado en Matlab; complementado con el diseño de un modelo tridimensional de una persona que representa el lenguaje de señas colombiano para generar el proceso de comunicación básico con el estudiante sordo. De esta investigación el aporte que podemos rescatar es el diseño del filtro pasa alto para eliminar frecuencias cercanas a 60Hz y también la selección del filtro elíptico ya que son eficientes para aproximación en magnitud; ambos filtros digitales fueron desarrollados en Matlab.

Tapia, V., Reinoso, R., Carrillo, E. (2017) en la tesis titulada: Sistema de traducción de voz y texto a lenguaje de señas. El objetivo ha sido el desarrollo de software que se utilizó en este trabajo y ha permitido la implementación a través de la creación de un prototipo que ha tenido la interacción constante de los usuarios, lo que también ayudó a validar el despliegue del producto final. Se cuenta ahora con la aplicación móvil denominada "Trad Señas" que se constituye en la primera versión del sistema compatible con dispositivos móviles Android y disponible en la línea a través de Google play. La aplicación fue desarrollada para mejorar el proceso de comunicación con personas que padecen discapacidad auditiva; para validar el funcionamiento de la misma se aplicaron pruebas a 20 usuarios evaluando su experiencia de uso, los resultados obtenidos son positivos y permiten determinar que la aplicación es eficiente, efectiva y satisfactoria. Se concluye que la aplicación desarrollada cumple con el objetivo, ya que permite la traducción de texto a señas, de voz a señas y presenta un catálogo completo de las señas utilizadas en esta localidad; posibilita por lo tanto la comunicación con personas que padecen sordera a través del uso de un dispositivo móvil. En la presente investigación el aporte que nos da es la utilidad de otras herramientas de software adicionales como "Eclipse Luna", "Plugin de voz de google", "Visual paradigm", "Balsamiq Mockups" y "Sdk y jdk java".

Según Sajjad, L., Hamdan, W. (2020) en el artículo titulado: Diseño e implementación de un sistema inteligente traducir texto árabe al lenguaje de señas árabe". En este artículo, se diseñó un sistema inteligente que puede analizar el texto árabe, teniendo en cuenta el contexto del texto, y luego, traduzca el texto al lenguaje de señas árabe. No existe una versión común o universal del lenguaje de señas. Esto es especialmente cierto para el lenguaje de señas árabe (ArSL) que tiene varias versiones dependiendo de alguna parte del país árabe donde se esté usando. Este sistema se implementó usando lenguaje de programación Java, incluido con la aplicación Screen Builder para crear interfaces de usuario, botones de pantalla y campos de texto. El sistema que utilizaron para la animación de personajes fue proporcionado por MindRockets, Inc. para mostrar el Lenguaje de Señas Arabe (ArSL) de la palabra traducida. El resultado preliminar de las pruebas del sistema ha demostrado una precisión general del 85% en traducir correctamente palabras árabes al lenguaje de señas árabe. Este artículo nos permite conocer que existen otros softwares como el lenguaje de programación Java, incluido con la aplicación

Screen Builder. Y para la animación tridimensional utiliza Mind Rockets Inc. software de desarrollo privado, que desarrolla servicios que sean accesibles para las personas sordas utilizando avatares inteligentes personalizables.

Anwar, J., Ahmed, A. (2017) en el artículo titulado: Sobre el diseño e implementación de una señal de voz / sistema texto. En este artículo, se han utilizado dos técnicas diferentes para Implementar un sistema de movimientos de una mano / sistema texto. El primer enfoque se basa en la visión y es fácil de implementar en computadoras de laboratorio o dispositivos inteligentes (móviles, tabletas o Ipad). Sin embargo, la principal limitación de este método es que depende del entorno ambiental y la alineación de la cámara. En la segunda parte es un sistema se basa en un guante con interfaz inalámbrica, con micro controlador y presentación de dispositivos para traducir el lenguaje de señas a voz al idioma árabe / inglés. Estos sistemas pueden traducir la señal de texto a voz para ayudar a sordos y mudos, en un guante que está diseñado e implementado. En este artículo se ha cumplido con el objetivo analizando dos enfoques de investigación para diseñar e implementar un traductor de movimiento de señas a texto.

Según Pujos, L. (2020) en la tesis titulada: Aplicación Móvil y su relación con el aprendizaje de personas con capacidades especiales Auditivas.La inclusión de la Realidad Aumentada (RA) y Realidad Virtual (RV) en aplicaciones móviles ha sido un hecho muy evidente en los últimos años, el ámbito educativo de las personas con capacidades especiales auditivas requiere mucho más de este recurso para aprovechar su facultad visual. Por lo que es importante recabar información para conocer la relación existente dentro del proceso de enseñanza-aprendizaje. La investigación tuvo como objetivo el desarrollo de una aplicación móvil con diseño a través de avatares predeterminados, con el software Daz Studio que es una herramienta para el diseño de animaciones. Que permita fortalecer el aprendizaje de los estudiantes con capacidades especiales auditivas de la Unidad Educativa Especializada Dr. Camilo Gallegos Domínguez. El proyecto tiene un enfoque cuantitativo, el cual nos ayudó al análisis de datos y cualitativo para conocer el grado de asimilación que muestran los estudiantes sobre la aplicación móvil. Utilizamos el Modelo de Aceptación Tecnológico (TAM) y Diseño para conocer la percepción de los alumnos por medio de una encuesta como instrumento. Los resultados mostraron un alto grado de satisfacción hacia el uso de la RA y RV en la aplicación móvil como método de aprendizaje.

Según Wulan M., Filda A. (2017) en el artículo titulado: Filtrado de voz humana con filtro Elimina Banda diseñado en MATLAB. Este artículo analiza el filtrado de la voz humana. El filtrado se realizó atenuando las frecuencias de la voz humana, dentro de las frecuencias de una canción usando un diseño de filtro elimina banda. El algoritmo se hizo para filtrar la voz humana de un audio. Luego, el resultado se optimiza para encontrar el mejor resultado para el filtrado. Se utilizan los métodos para la optimización, para simular el diseño del filtro elimina banda se realiza con diferentes rechazos de banda, el resultado para mostrar su espectro de frecuencia (FTT), y se repite el proceso hasta que la voz escuchada tenga el resultado óptimo. De la investigación realizada, se concluye que un diseño de filtro simulado en MATLAB es adecuado para el filtrado de la voz humana. Se muestra el diseño del filtro para ayudar al proceso, tanto la optimización como la comparación final. El diseño del filtro funciona tanto para hombres y voz femenina.

2.3. Bases Teóricas relacionadas con el tema

2.3.1. Prototipo traductor de voz a la Lengua de Señas Peruana:

IBM Cloud: Según IBM (2020), es una plataforma en la nube de IBM, combina una plataforma como servicio (PaaS) con la infraestructura como servicio (IaaS) para proporcionar una experiencia integrada. En esta plataforma se puede desarrollar proyectos multidisciplinarios, los cuales utilizan diversos servicios de la nube para brindar la mejor solución posible.

Software MatLab: Según López, C. P. (2002), es un software de computación científica para optimizar la resolución de problemas de ingeniería, así como también del tipo científico. El presente software es utilizado en diversos centros de investigación, universidades e institutos. Posee diversas aplicaciones desde ser una herramienta matemática hasta el desarrollo complejo de diversos proyectos tecnológicos. Adicionalmente, Matlab posee diversas librerías y aplicaciones como son el caso de las interfaces gráficas, entre las cuales tenemos: Simulink, GUIDE y App Designer, cada una de estas cuentan con una infinidad de aplicaciones.

Daz Studio: Según Rasmussen. A. (2015), es un software 3D que permite al artista posar, animar y renderizar figuras y modelos 3D. No es un software

de modelado 3D, aunque puede hacer algunos modelados básicos utilizando primitiva las escenas se construyen en un espacio tridimensional y se renderizan en imágenes o animaciones bidimensionales; aunque muy recientemente la empresa DAZ ha comenzado a ofrecer impresión 3D.

Técnicas de diseños de filtros: Según Oppenheim A., Schafer R. (2011), Los filtros son una clase particularmente importante de sistemas lineales e invariantes con el tiempo. Estrictamente hablando, el término filtro selectivo en frecuencia nos sugiere un sistema que deja pasar ciertas componentes de frecuencia y rechaza completamente otras, pero en un contexto más amplio cualquier sistema que modifique ciertas frecuencias con respecto a otras se denomina también filtro.

Speech To Text: Según IBM (2017), El servicio Speech to Text convierte la voz en texto legible de acuerdo con el idioma que el usuario especifica. El servicio es capaz de transcribir voz de varios idiomas y formatos de audio a texto con baja latencia. Para la mayoría de los idiomas, el servicio admite dos anchos de banda: banda ancha y banda estrechas.

2.3.2. Indicaciones de posicionamiento del paciente:

Indicaciones de posicionamiento: Según Chen M. (2006), Indicaciones metodológicas de imagen más frecuente para estudiar el corazón y los grandes vasos es la radiografía de tórax, con proyecciones posteroanterior (PA) y lateral izquierda (LAT) en bipedestación. Los términos PA y lateral izquierda se refieren a la dirección en que el haz de rayos X atraviesa el cuerpo antes de alcanzar la placa radiográfica. Lo ideal es realizar las radiografías en inspiración máxima.

2.4. Definición de términos básicos

Python: Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”.

Lengua de Señas Peruana: Es un lenguaje que se percibe a través de la vista y requiere el uso de las manos como articuladores activos, el uso del espacio como lugar de articulación (estructura fonológica) y como referencia temporal.

Radiografía: Una radiografía es una prueba rápida e indolora que genera imágenes de las estructuras internas del cuerpo, en especial de los huesos. Los haces de rayos X pasan a través del cuerpo y se absorben en diferentes cantidades según la densidad del material a través del cual pasan.

Rayos X: Un rayo X es un paquete discreto de energía electromagnética llamada fotón. Desde este punto de vista, es similar a otras formas de energías electromagnéticas como la luz visible, infrarrojas, ultravioleta, ondas de radio o rayos gamma.

Animación 3D: La animación consiste en dar vida a aquellos objetos que no la tienen, en la animación 3D, además de esto, dichos objetos se pueden girar y mover en un espacio tridimensional.

2.5. Diseño de la Investigación

2.5.1. Variables de investigación

La variable independiente es: Prototipo de traductor de voz a la Lengua de Señas Peruana.

La variable dependiente es: Indicaciones de posicionamiento del paciente.

2.5.2. Tipo y Método de investigación

El tipo de investigación es aplicada, porque está enfocada en resolver un problema real de la sociedad y tecnológica, porque utiliza diferentes softwares especializados en procesamiento de voz, conversión de voz a texto, programación en lenguaje Python y animación tridimensional. De igual forma, el método de investigación es empírico y experimental porque se va adquirir señales de audio a través de un micrófono externo conectado a una laptop, esta señal posteriormente pasará por una etapa de filtrado digital para reducir los ruidos generados por el ambiente de la sala de Rayos X del Hospital Nacional Hipólito Unanue. Así mismo, luego ingresará la señal de voz filtrada a la plataforma de servicio de IBM Watson Speech to Text para la conversión de voz a texto. Continuamente, se diseñará un algoritmo en Python que pueda analizar la señal final, para corroborar que

el texto traducido corresponde con el audio recibido. Finalmente, luego de verificar la traducción correcta de voz a texto, pasará a la etapa de ejecución de la animación tridimensional de las palabras traducidas por el software DAZ Studio, mostrando visualmente en la pantalla del ordenador, la traducción final de la frase en Lengua de Señas Peruana.

2.5.3. Técnicas e Instrumentos de recolección de datos

El instrumento utilizado para recolectar los datos, los cuales corresponden a la voz de 04 tecnólogos, es un micrófono externo conectado a través de plug 3.5, a una laptop con un procesador Core i3 de tercera generación y una memoria RAM de 8Gb. Así mismo, la técnica de recolección de datos será a través de una grabación directa de voz, mediante el micrófono externo con las siguientes especificaciones técnicas: Tipo de patrón polar: omnidireccional, frecuencia de respuesta: 50Hz – 16kHz, sensibilidad: -30dB± 3dB, impedancia: ≤2.2kΩ, y puerto de comunicación: miniplug stereo 3.5mm.

2.5.4. Procedimiento para la recolección de datos

Para el procedimiento de recolección de datos, se siguieron los siguientes pasos:

- Primero se tuvo que dar la situación del tecnólogo y el paciente que tenga la Lengua de Señas Peruana como lengua materna, luego se procedió a la adquisición de la voz.
- Después, se comenzó a grabar la voz del tecnólogo, mediante la app nativa del celular y el programa grabador de voz de Windows 10, con las indicaciones de posicionamiento correspondientes de cada tecnólogo, a través del micrófono externo conectado a la laptop y el micrófono interno del celular, que posteriormente fueron almacenados en la laptop.
- Finalmente, luego de almacenadas las voces con las indicaciones de posicionamiento brindadas por cada tecnólogo en la laptop con una duración promedio de 4 minutos aproximadamente por archivo, se convirtieron al formato WAV para pasar a la etapa de filtrado de ruido.

CAPÍTULO III: DESARROLLO DEL PROYECTO

En este capítulo se muestra el desarrollo del proyecto, detallando el procedimiento realizado en cada etapa.

3.1. Diagrama de bloques

Seguidamente, se muestra el diagrama de bloques del proyecto de tesis planteado (Ver figura N°1). Donde, se aprecian las principales etapas del proyecto, comenzando con la adquisición y filtrado de la voz, seguido de la conversión de voz a texto previamente filtrada, posteriormente la verificación de la conversión de voz a texto, de ser correcta se ejecuta la animación 3D correspondiente, caso contrario, regresa a la primera etapa.

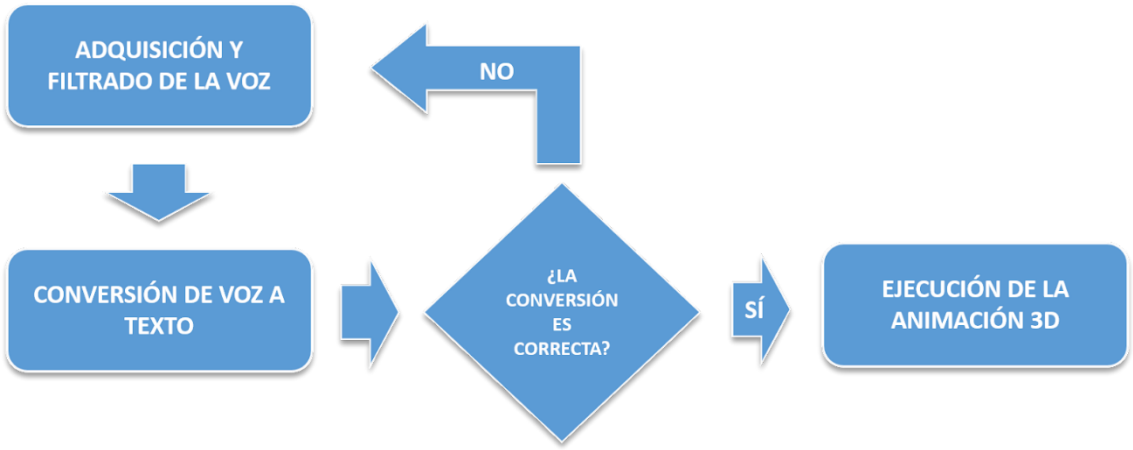

Figura Nº 1: Diagrama de bloques.
Fuente: Elaboración propia.

3.2. Adquisición y procesamiento de la voz

3.2.1. Adquisición de la voz

Está conformado por un micrófono de tipo condensador con captación de sonido omnidireccional, voltaje de operación de 1.5V, respuesta de frecuencia de 50Hz-16Khz y puerto de comunicación por miniplug alámbrico.

A continuación, se muestra la siguiente figura N°2 que corresponde a la imagen del micrófono utilizado.

Figura N° 2 Micrófono alámbrico conexión miniplug

Fuente: Linio

3.2.2. Diseño del filtro digital pasabanda

En la siguiente etapa se diseñó un filtro digital pasa banda Chebyshev a través del software Matlab, el cual se utilizó para filtrar las frecuencias de la voz, en el rango de trabajo de 250Hz - 3KHz, denominando estas frecuencias como las más cercanas a la frecuencia de voz humana. Con el uso de este filtro se disminuyó el ruido del ambiente generado en la sala de Rayos X, permitiendo que el tecnólogo brinde las indicaciones de posicionamiento para realizar la toma de radiografía tórax. Así mismo, se realizó el diseño y las pruebas en el software Matlab para ver la respuesta en el espectro de frecuencia del filtro digital y, verificar el correcto filtrado del audio en formato WAV. Posteriormente se procedió a obtener la función de transferencia del filtro utilizado, para su implementación en otra plataforma como Python.

A continuación, en la figura N° 3 se muestra el código de programación en Matlab.

```
21      %%
22      %filtro Chebyshev
23 -    FC=[250 1500];
24 -    n=4; %orden del filtro
25 -    wp=5 %rizado de banda de paso de pico a pico
26 -    [b,a]=cheby1(n,wp,(FC/(FS/2)),'bandpass');
27 -    FT=tf(b,a,1)
28
29 -    [h,w]=freqz(b,a,FS,FS);
30      %plot(w/1000,(abs(h)))
31      %zplane(b,a)
32
```

Figura N° 3: Algoritmo de diseño del filtro digital pasabanda en MatLab

Fuente: Captura de pantalla del software Matlab

Así mismo, en la figura N°4 se muestra la función de transferencia del filtro digital Chebyshev pasa banda de orden 8.

```
FT =

          3.689e-06 z^8 - 1.475e-05 z^6 + 2.213e-05 z^4 - 1.475e-05 z^2 + 3.689e-06
  ------------------------------------------------------------------------------------------
  z^8 - 7.88 z^7 + 27.22 z^6 - 53.82 z^5 + 66.64 z^4 - 52.9 z^3 + 26.3 z^2 - 7.486 z + 0.934
```

Figura N° 4: Función de transferencia en MatLab

Fuente: Captura de pantalla del software Matlab

Posteriormente, en la figura N°5 se muestra el gráfico de la respuesta en frecuencia del módulo del filtro pasabanda diseñado.

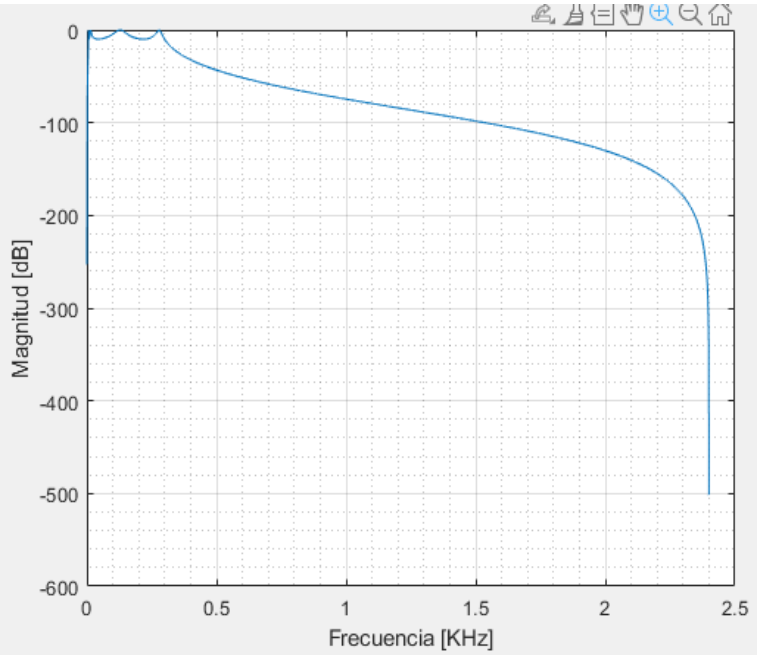

Figura N° 5 Respuesta en frecuencia del Módulo del Filtro pasabanda diseñado

Fuente: Captura de pantalla del software Matlab

3.3. Conversión de la voz procesada en texto

3.3.1. Configuración de parámetros en IBM Watson Speech to Text

Según la página web IBM Cloud (2021), es una plataforma multicloud híbrida de nueva generación, con funcionalidades avanzadas de datos e Inteligencia Artificial y una amplia experiencia empresarial en 20 sectores. Asimismo, IBM Cloud cuenta con más de 170 productos y servicios que se extienden a datos, contenedores, inteligencia artificial (IA), internet de las cosas (IOT) y Blockchain. Por tal razón el desarrollo de esta Tesis planteada está enfocada en uno de los servicios de inteligencia artificial, en especial del IBM Watson. Adicionalmente, IBM Watson ofrece servicios de procesamiento de voz natural, reconocimiento visual y machine Learning. Para el desarrollo del presente proyecto se utilizó el servicio Speech to Text, el cual proporciona prestaciones de transcripción de voz humana a texto con precisión para las diferentes aplicaciones que se puedan desarrollar.

Para el uso de todos los servicios que brinda IBM Cloud, se debió crear una cuenta gratuita, ya que, algunos son con opción de pago o también libres, la única limitación con los servicios libres o gratuitos es el tiempo o almacenamiento de uso. A continuación, en la Figura N° 6 se muestra la imagen del Catálogo del IBM Cloud del Servicio Speech to Text.

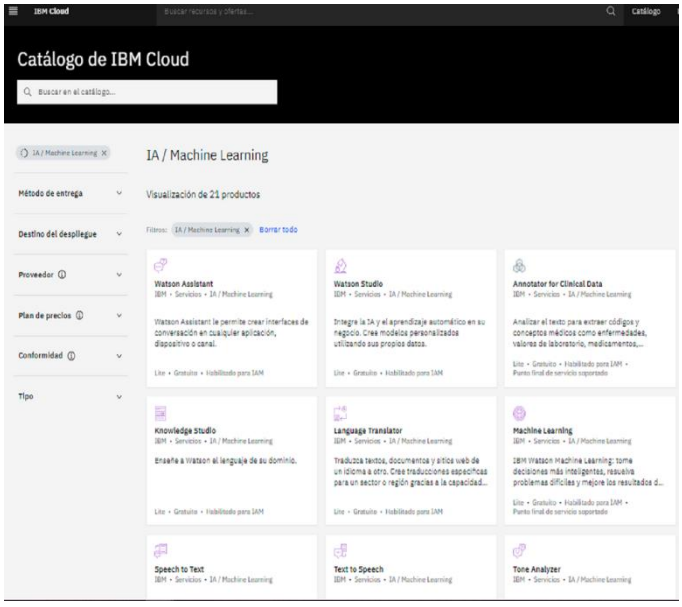

Figura Nº 6: Catálogo de IBM Cloud

Fuente: https://cloud.ibm.com

El servicio de Speech to Text tiene planes con diferentes costos y también un plan Lite gratuito, el cual brinda 500 minutos de uso al mes, en la presente investigación se utilizó este último por permitir un tiempo suficiente y de manera óptima. A continuación, luego de obtener el servicio de Speech to Text creado, se procedió a generar las credenciales de conectividad, como CLAVE API y el URL; ya que, una vez generado estos accesos se puedo utilizar en múltiples entornos de programación y accesos como el lenguaje de programación Python.

En la Figura N° 7 se muestra el servicio de Speech to Text con sus respectivas credenciales CLAVE API y URL.

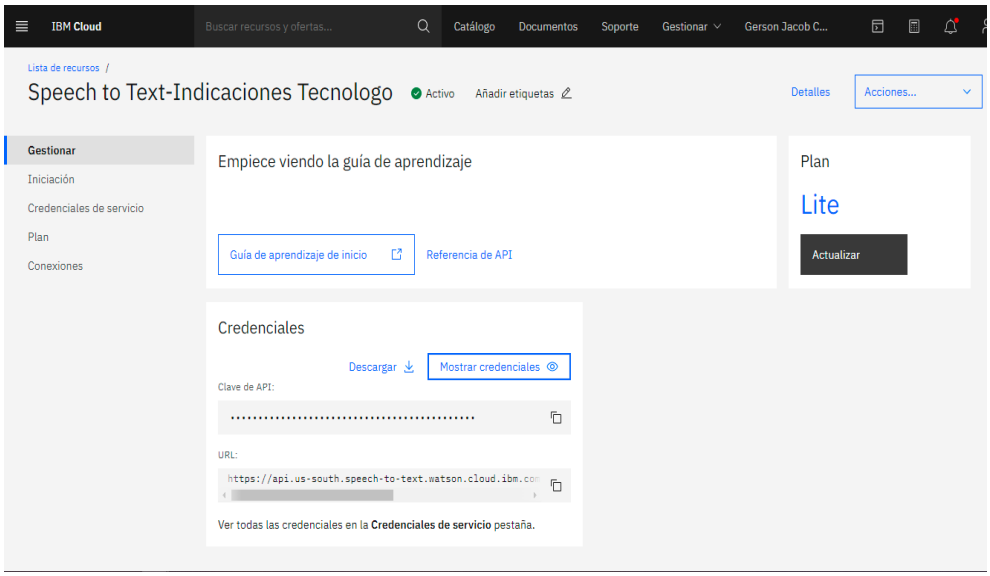

Figura Nº 7: Servicio de Speech to Text de IBM Cloud

Fuente: https://cloud.ibm.com

3.4. Algoritmo para verificación de la conversión de voz a texto

La implementación del algoritmo para verificar la conversión de voz a texto realizado por el servicio Watson Speech to Text es una etapa importante, ya que, así se asegura una comunicación acertada de tecnólogo a paciente, evitando dar indicaciones erróneas.

3.4.1. Diagrama de flujo del algoritmo

El funcionamiento del algoritmo está enfocado a partir de la obtención del texto, la cual almacena en una variable y muestra un mensaje en la pantalla preguntando “Quisiste decir: texto” con opción a responder sí o no, si la

respuesta es sí, entonces ejecuta la animación correspondiente a la frase traducida, limpia la variable y regresa al menú de grabación de voz, si la respuesta es no, muestra un mensaje de advertencia con que dice “vuelva a grabar”, luego limpia la variable y regresa al menú de grabación de voz.

A continuación, en la figura N°8, se puede apreciar el diagrama de flujo del algoritmo de verificación.

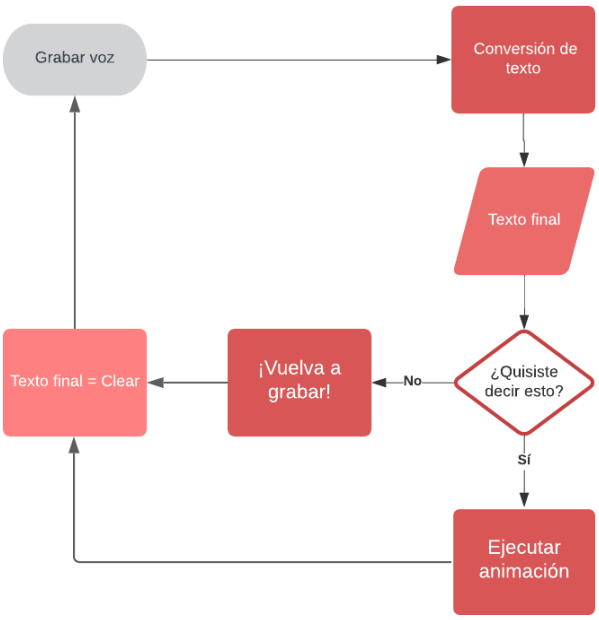

Figura Nº8: Diagrama de flujo del algoritmo de verificación.

Fuente: Elaboración propia.

3.5. Caracterización del avatar y animación 3D de la lengua de señas peruana utilizando DAZ Studio

3.5.1. Caracterización del avatar en DAZ Studio

La personificación del avatar se realizó por etapas, las cuales constan de la selección de un molde base para el avatar dependiendo el género, la caracterización de su rostro, cabello, prendas de vestir y un ambiente como fondo.

- Selección de molde base

Este proyecto se implementó con el paquete llamado “Genesis 8 Starter Essentials” el cual viene previamente instalado dentro del software DAZ Studio. Seguidamente en la figura N°9, se muestra la imagen referencial del paquete “Genesis 8 Starter Essential”.

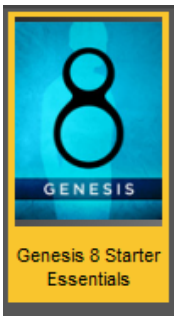

Figura Nº9: Genesis 8 Starter Essentials.

Fuente: Software DAZ Studio.

Dentro de este paquete se seleccionó un personaje de género masculino, llamado “Genesis 8 Basic Male” el cual se muestra a continuación en la figura N°10.

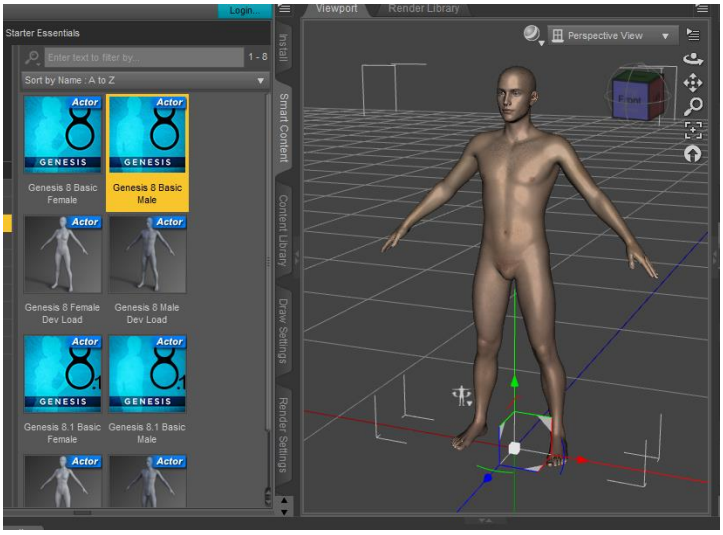

Figura Nº10: Genesis 8 Basic Male.

Fuente: Software DAZ Studio.

- Caracterización del rostro y cabello

  Para la caracterización del rostro se agregó pestañas y una expresión sonriente, estas características vienen dentro del paquete “Genesis 8 Starter Essentials”.

  A continuación, en la figura N°11 se muestra un antes y después. La imagen de la izquierda sin pestañas y la imagen de la derecha con pestañas.

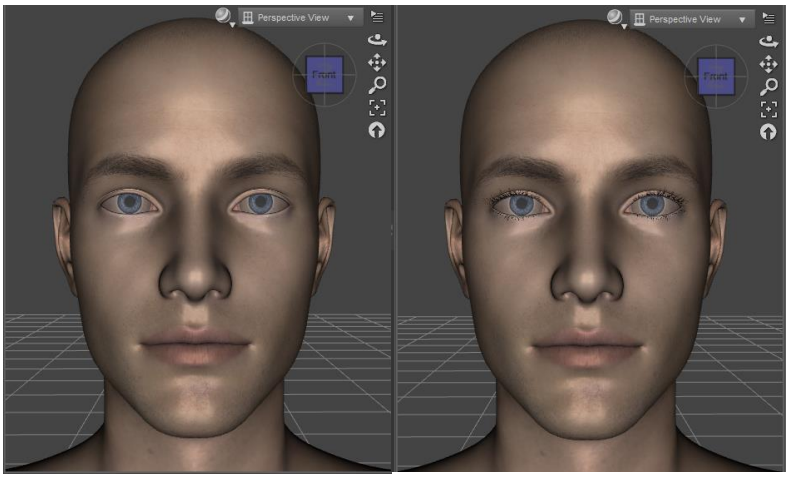

Figura Nº 11: Agregado de pestañas

Fuente: Software DAZ Studio.

Seguidamente, en la figura N°12 se muestra el panel para editar expresiones, desde donde se modificó el parámetro para obtener un rostro sonriente.

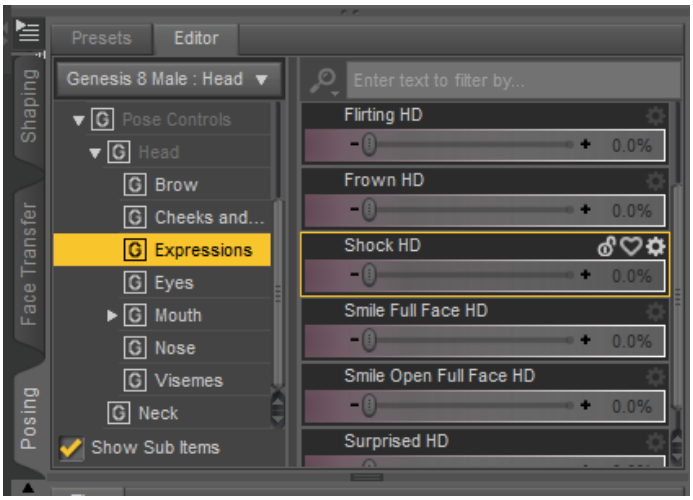

Figura Nº12: Editor de expresiones

Fuente: Software DAZ Studio.

Luego, en la figura N°13, se muestra un antes y después del rostro, en el que se puede apreciar un cambio de expresión seria a una expresión sonriente.

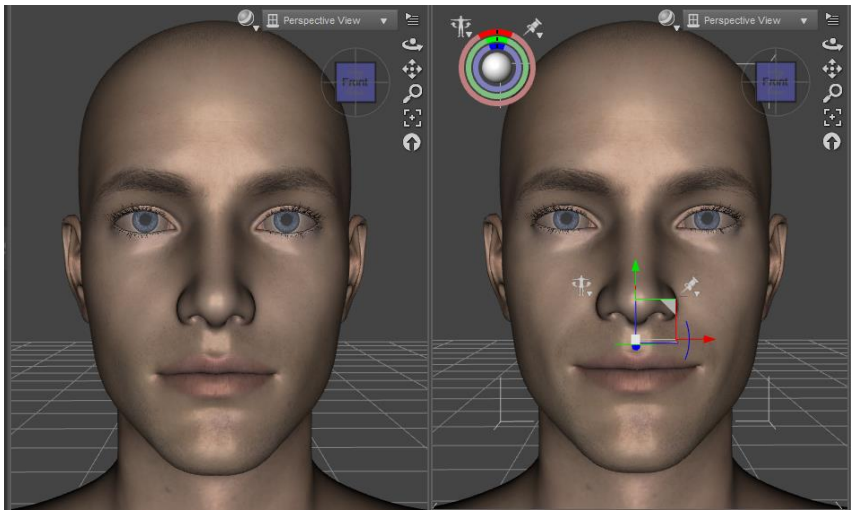

Figura Nº 13: Expresión sonriente.

Fuente: Software DAZ Studio.

Después, se agregó el cabello “Armani Style”, ya que es el único estilo de cabello para varón que se puede encontrar dentro del paquete gratuito “Genesis 8 Starter Essentials”. El estilo de cabello se puede apreciar el figura N°14.

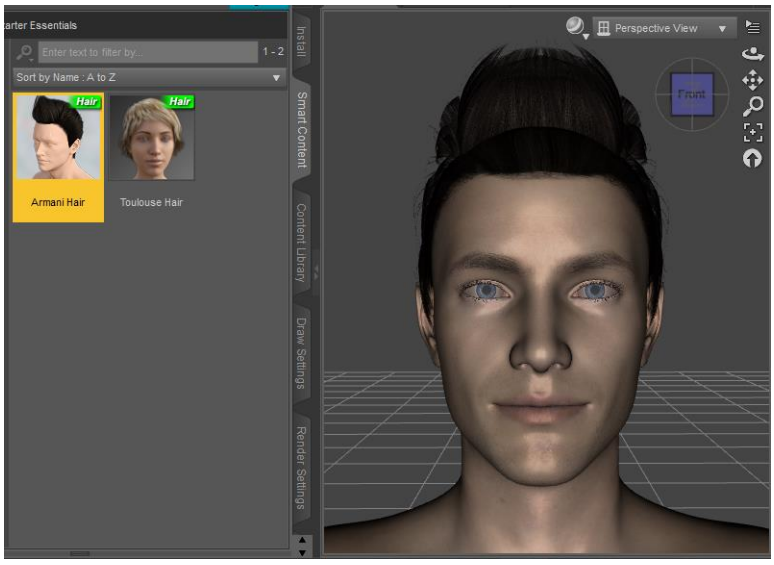

Figura Nº14: Selección de cabello.

Fuente: Software DAZ Studio.

Así mismo, se cambió el color del cabello a “brown dark”, este cambio se puede apreciar en la figura N°15.

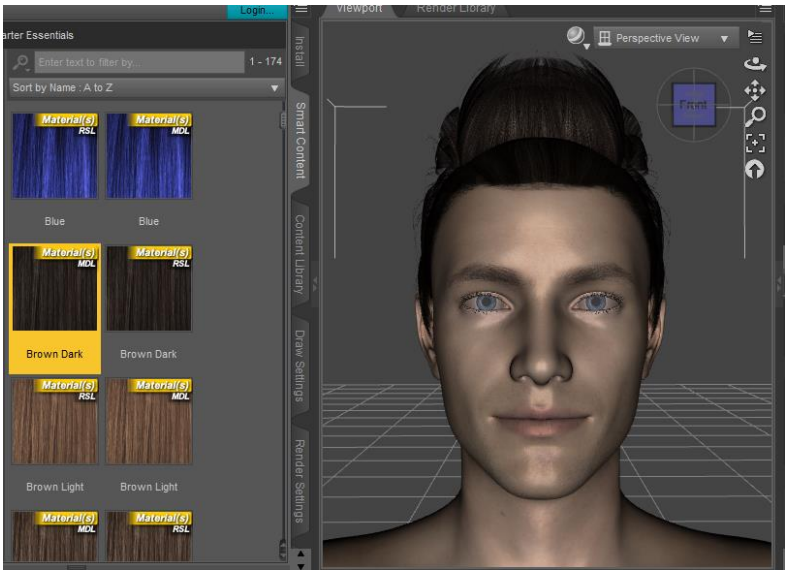

Figura Nº 15: Selección del color de cabello.

Fuente: Software DAZ Studio.

- Vestimenta del personaje

  Se eligió la vestimenta básica “Basic Wear Outfit 1” que viene dentro del paquete “Genesis 8 Starter Essentials”, ya que, es el más indicado para nuestra aplicación dentro del paquete gratuito disponible.

  A continuación, en la figura N°16 se muestra al personaje vestido con el “Basic Wear Outfit 1”.

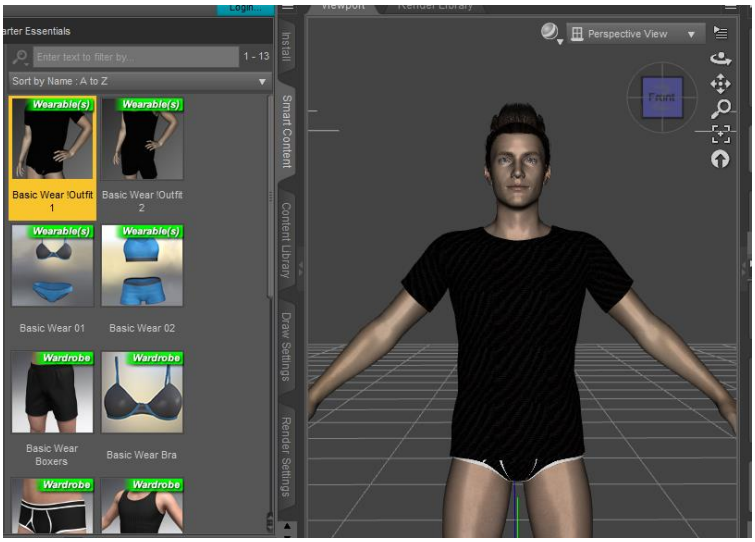

Figura Nº 16: Selección de vestimenta.

Fuente: Software DAZ Studio.

Seguidamente se seleccionó el “Basic Wear Boxer” para cubrir la parte inferior del cuerpo, como se muestra en la figura N°17.

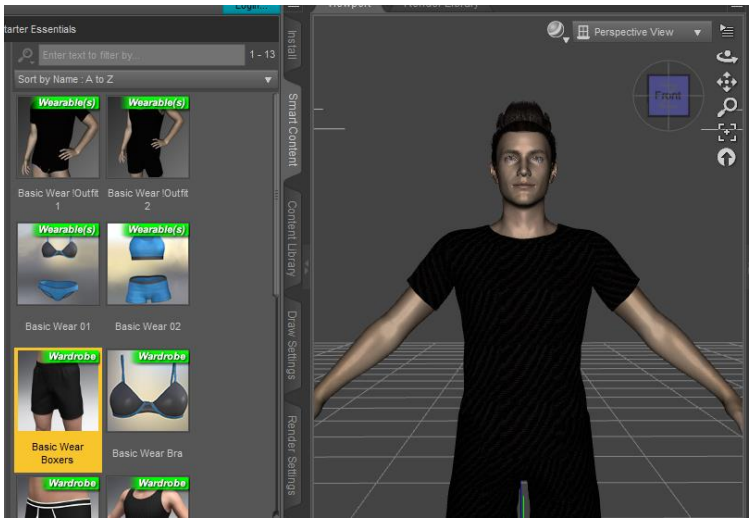

Figura Nº 17: Selección de la parte inferior “Basic wear boxers”.

Fuente: Software DAZ Studio.

Finalmente, se cambió el color de la vestimenta completa a blanco, esto con la finalidad de dar un aspecto más apropiado para una sala de Rayos X.

A continuación, en la figura N°18 se muestra el cambio de color de la vestimenta completa del personaje.

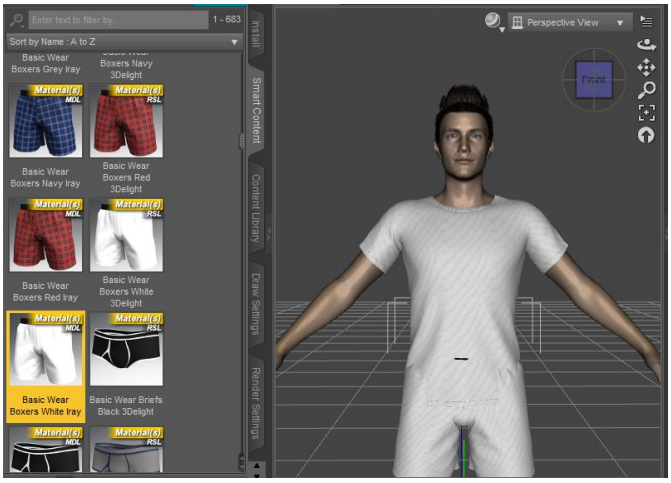

Figura Nº 18: Cambio de color de a blanco.

Fuente: Software DAZ Studio.

- Selección e inserción de fondo

  Para agregar un ambiente (fondo), primero se seleccionó una imagen desde Google Chrome relacionada a una sala de Rayos X. Esta imagen se muestra en la figura N°19.

Figura Nº 19: Sistema de radiografía Multix Fusion Max

Fuente: Siemens Healthineers.

Posteriormente, se seleccionó esta imagen como “Environment” dentro del software DAZ Studio, en la pestaña “Windows” → “Panes (tab)”.

A continuación, en la figura N°20 se puede apreciar la ubicación para seleccionar la opción “Environment”.

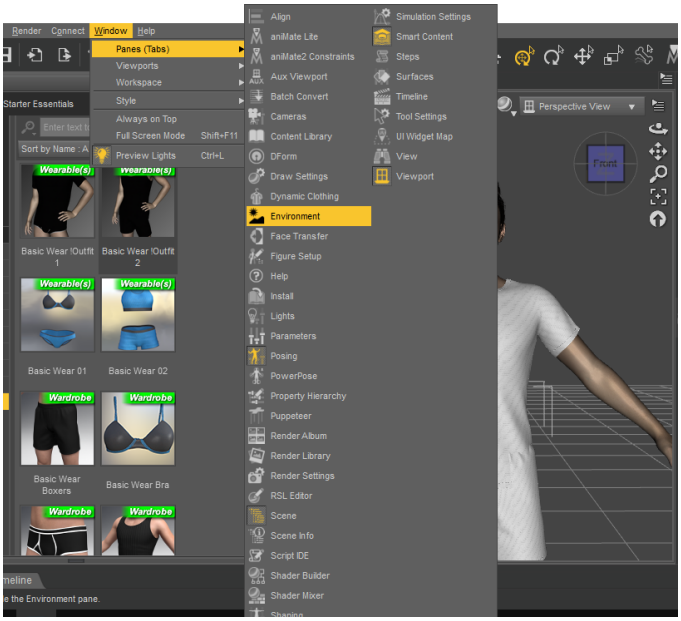

Figura Nº 20: Ubicación de “Environment”.

Fuente: Software DAZ Studio.

Seguidamente, en la opción “Type” → Backdrop se seleccionó la imagen previamente descargada. La ubicación se muestra en la figura N°21.

Figura Nº 21: Avatar ambientado.

Fuente: Software DAZ Studio.

3.5.2. Proceso para la animación de la Lengua de Señas Peruana

Para la animación de las frases en Lengua de Señas Peruana, se debió seguir diferentes pasos, los cuales varían en los parámetros dependiendo de las palabras empleadas; sin embargo, los pasos son repetitivos e intuitivos, razón por la cual solo se muestra el proceso de animación de la frase "aguante la respiración". Este proceso se dividió en diferentes etapas, comenzando por la preparación del entorno, en donde fue necesario fijar una postura inicial para las animaciones, luego se debió traducir las palabras a la Lengua de Señas Peruana con ayuda de un manual y, finalmente, luego de tener las etapas anteriores listas, se procedió a diseñar las animaciones.

- Preparación del entorno

  Para preparar el entorno, primero se definió una duración inicial de animación de tres segundos, el tiempo de duración varía por cada frase y se determinó una cantidad de treinta fotogramas por segundo y una postura inicial del avatar con los brazos relajados.

  En la figura N°22 se muestra el fijado del tiempo en tres segundos.

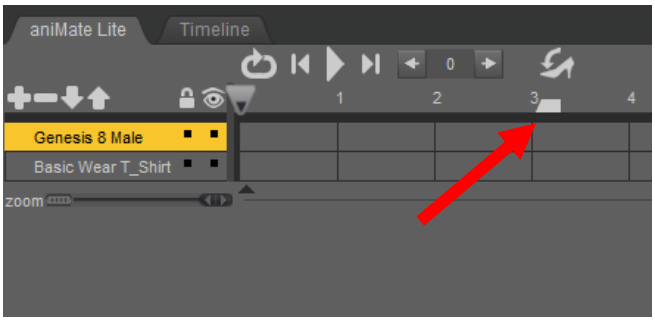

Figura Nº 22: Definición del tiempo de animación

Fuente: Software DAZ Studio

Seguidamente, en la figura N°23 se muestra el fijado de los FPS en 30.

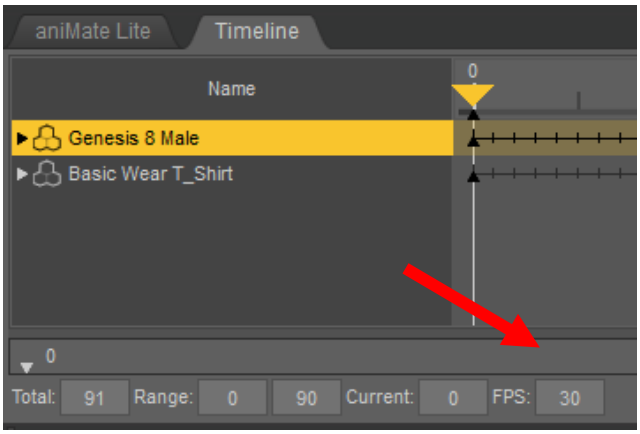

Figura Nº23: Fijado de fotogramas por segundo.

Fuente: Software DAZ Studio.

Para el posicionamiento inicial del avatar, se seleccionó el fotograma cero y se hizo una flexión del hombro derecho con rotación en el eje Z igual a +27 y una flexión del hombro izquierdo con rotación en el eje Z igual a -27.

A continuación, en la figura N°24 se muestra el fijado de rotación del hombro derecho.

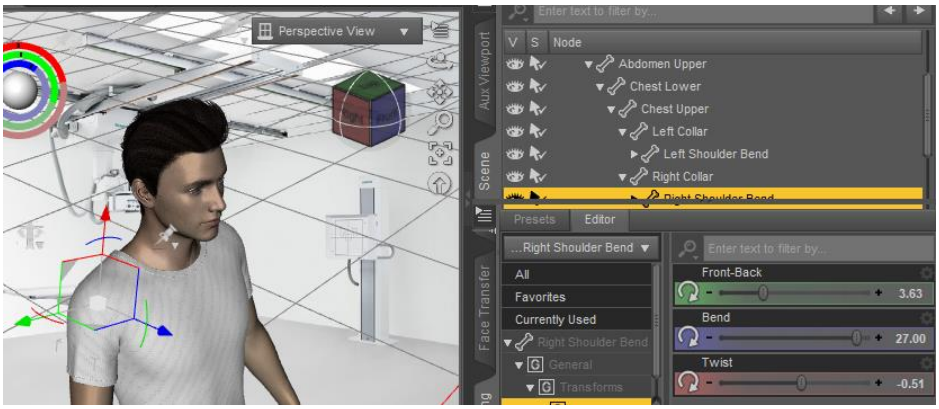

Figura Nº24: Rotación del eje Z del hombro derecho a +27.

Fuente: Software DAZ Studio.

Luego, en la figura N°25 se muestra el fijado de rotación del hombro izquierdo.

Figura Nº25: Rotación del eje Z del hombro izquierdo a -27

Fuente: Software DAZ Studio.

Finalmente, en la figura 26, se muestra la posición inicial del avatar para todas las animaciones.

Figura Nº 26: Posición inicial del avatar.

Fuente: Software DAZ Studio.

- Traducción de la frase a Lenguaje de Señas Peruana

Para poder traducir las frases a la Lengua de Señas Peruana (LSP), se hizo uso del libro Ministerio de Educación (2015). Lengua de Señas Peruana, el cual sirve como guía para el aprendizaje de esta lengua.

A continuación, en la figura N°27 se muestra los equivalentes para algunas frases usadas, por ejemplo: “buenos días” y “buenas noches”.

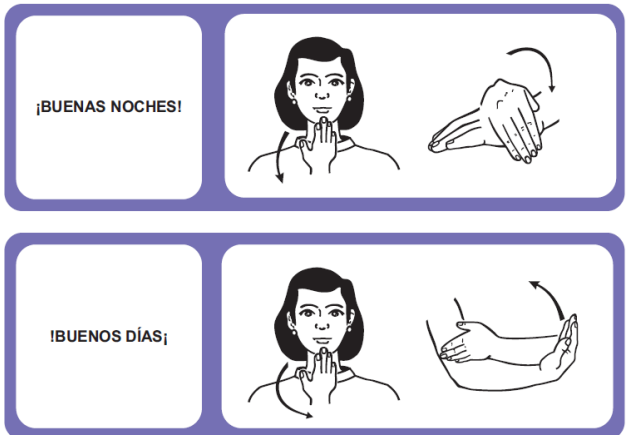

Figura Nº 27: Buenos días y buenas noches en LSP.

Fuente: "Lengua de Señas Peruana"

Seguidamente, en la figura N°28 se muestra la traducción en LSP para las frases "aguante la respiración" y en la figura N°29 se nuestra la traducción en LSP para la frase "terminamos, gracias".

Figura Nº 28: Aguante la respiración en LSP.

Fuente: "Lengua de Señas Peruana".

Figura Nº 29: "Terminamos, gracias" en LSP.

Fuente: "Lengua de Señas Peruana".

- Animación de la Lengua de Señas Peruana en DAZ Studio

Seguidamente, se procedió a recrear las señas en el software DAZ Studio para cada frase, las cuales fueron un total de diez frases enfocadas en las indicaciones empleadas por el tecnólogo, al momento de realizar una toma de Rayos X de tórax. A continuación, se muestra el proceso para la animación de la frase “aguante la respiración”.

En primer lugar, se configuró los FPS a treinta y una duración para la animación de aproximadamente cuatro segundos, como se aprecia en la figura N°30.

Figura Nº 30: Edición de parámetros.

Fuente: Software DAZ Studio.

Luego, se realizó la posición para el primer movimiento, ubicado en el frame número veinticinco, en dónde el avatar realiza la seña de “Tú” en LSP, teniendo como posicionamiento del hombro derecho en el eje X de +23.21, en el eje Y de +30.66, en el eje Z de + 22.44 y como posicionamiento del antebrazo derecho en el eje X de -24.86, en el eje Y de +62. Para los dedos de la mano derecha, se tuvo todos los dedos totalmente flexionados, con excepción del dedo índice, teniendo como resultado final la seña de “tú”.

En la figura N°31 se muestra el resultado de la seña para la palabra “tú”.

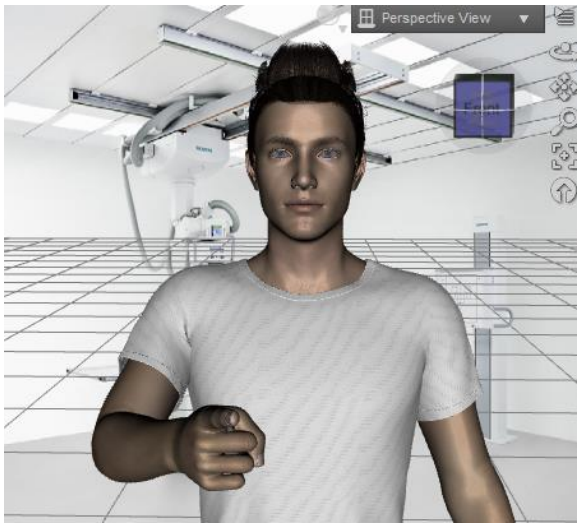

Figura Nº 31: Animación de “tú” en LSP.

Fuente: Software DAZ Studio.

Una vez finalizada la seña, se creó una pausa de siete frames, desde el frame veinticinco, al frame treinta y dos, con la finalidad de marcar más la seña realizada, para posteriormente pasar a una nueva posición. Para el segundo movimiento, ubicado en el frame número cuarenta y nueve, en dónde el avatar realiza la seña inicial de “aguantar” en LSP, se tuvo como posicionamiento del hombro derecho en el eje X de +37.12, en el eje Y de +52, en el eje Z de + 24.87 y como posicionamiento del antebrazo derecho en el eje X de -65.58, en el eje Y de +102. Para los dedos de la mano derecha, se tuvo todos los dedos totalmente flexionados, con excepción del dedo pulgar, se creó una pausa de cinco frames, desde el frame cuarenta y nueve, hasta el frame cincuenta y cuatro. Seguidamente, en el frame número sesenta y cinco, se creó la posición final de la seña “aguantar”, la cual tuvo como posicionamiento del hombro derecho en el eje X de +39.65, en el eje Y de +46.90, en el eje Z de + 21.26 y como posicionamiento del antebrazo derecho en el eje X de -66.60, en el eje Y de +101. La posición de los dedos de la mano se mantuvo igual. Se creó una pausa de cinco frames, desde el frame sesenta y cinco, hasta el frame setenta, teniendo como resultado final la seña de “aguantar” en LSP.

A continuación, en la figura N°32 se muestra el movimiento del brazo derecho desde los labios hasta la barbilla.

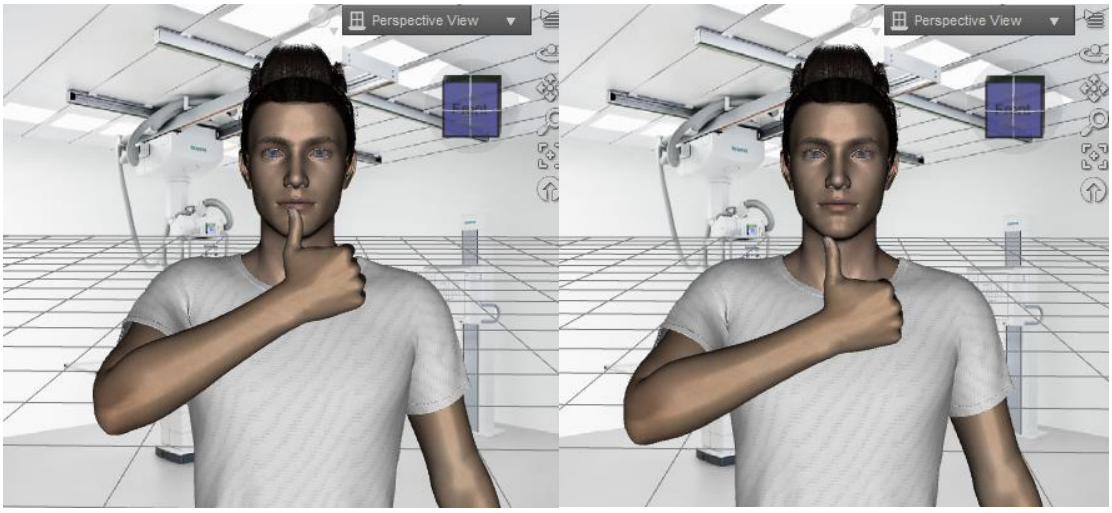

Figura Nº 32: Animación de “aguantar” en LSP.

Fuente: Software DAZ Studio.

Para el tercer y último movimiento, ubicado en el frame número ochenta y cinco, en dónde el avatar realiza la seña inicial de “respirar” en LSP, se tuvo un posicionamiento del hombro derecho en el eje X de -7.02, en el eje Y de +33.58, en el eje Z de -2.92 y como posicionamiento del antebrazo derecho en el eje X de -88.52, en el eje Y de +121. Para los dedos de la mano derecha, se tuvo todos los dedos totalmente flexionados, con excepción del dedo medio e índice, se creó una pausa de ocho frames, desde el frame ochenta y cinco, hasta el frame noventa y tres. Seguidamente, en el frame número ciento cuatro, se creó la posición final de la seña “respirar”, la cual tuvo como posicionamiento del hombro derecho en el eje X de -16.30, en el eje Y de +44.20, en el eje Z de -4.86 y como posicionamiento del antebrazo derecho en el eje X de -88.25, en el eje Y de +117. La posición de los dedos de la mano se mantuvo igual. Se creó una pausa de ocho frames, desde el frame ciento cuatro, hasta el frame ciento doce, teniendo como resultado final la seña de “respirar” en LSP.

Seguidamente, en la figura N°33 se muestra el movimiento a realizar para la seña “respirar”, el cual inicia con el dedo índice y medio a la altura de la nariz y finaliza con un movimiento vertical de los mismo hasta la altura de la ceja.

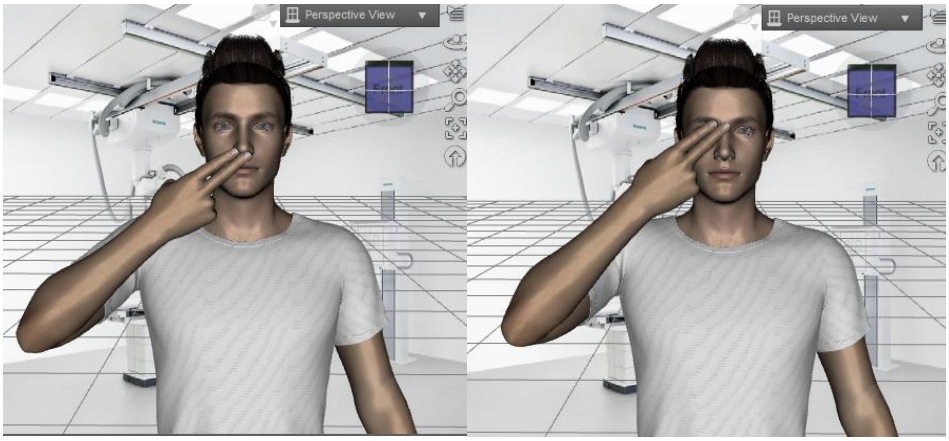

Figura Nº 33: Animación de “respirar” en LSP.

Fuente: Software DAZ Studio.

Finalmente, se procedió a renderizar la animación en el software DAZ Studio, para esto se configuró los parámetros de la siguiente manera, Dimensión prestablecida: 16:9, Tipo de renderizado: película en formato

AVI, Rango de renderizado: Inicio 0 – Final 120, como se muestra en la figura N°34.

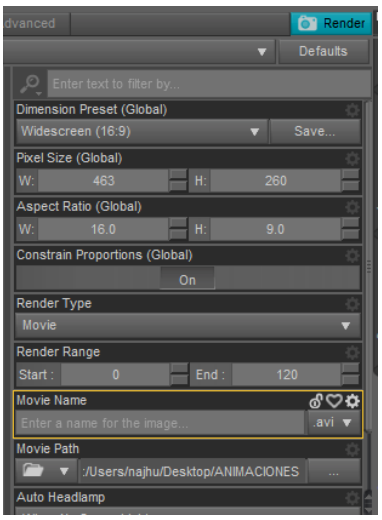

Figura Nº 34: Parámetros para renderizado.

Fuente: Software DAZ Studio.

Así se finalizó el proceso de animación y renderizado de las señas en LSP, este procedimiento es repetitivo para todas las frases, lo único que varía es la posición de las extremidades superiores y la duración de cada frase dependiendo de la palabra a animar.

3.6. Integración de las etapas del proyecto en lenguaje Python

3.6.1. Transcripción del filtro desarrollado en Matlab

Una vez realizada la simulación del diseño de filtro digital pasabanda en el Matlab, y luego de obtener la función de transferencia del mismo, se procedió a realizar la implementación del filtro digital pasabanda Chebyshev en lenguaje Python. Para ello se realizó la instalación de las librerías necesarias como el SCIPY y MATPLOTLIB; que son utilizadas a través de NUMPY, el cual ayuda en la ejecución del lenguaje Python.

- Se realizó la adquisición, configuración e instalación de las librerías en el entorno creado en el Spyder y se desarrolló el código en Python, posteriormente se realizó la inyección de un audio grabado en formato WAV con las indicaciones del tecnólogo hacia el paciente.

A continuación, en la figura N°35 se muestra el llamado de las librerías en Python.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import wave
from scipy.fftpack import fft
```

Figura N° 35: Algoritmo para uso de las librerías en Python

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Seguidamente, se realizó el algoritmo para convertir el audio a monoaural, esto con la finalidad de tener todo el audio en un solo canal, a diferencia de un audio estéreo que usa dos canales (L y R), lo cual ayudó al procesamiento de filtrado del mismo. A continuación, en la figura N°36, se muestra el código de programación de la conversión a monoaural.

```python
        #Convierte a monoaural
        monoaural = None
        for canal in canales:
            if monoaural is None:
                monoaural = canal
            else:
                monoaural = monoaural + canal
        monoaural = monoaural/audioInfo['nchannels']
        filtrado = self.filtrarOnda(monoaural)
        return t, monoaural, filtrado, audioInfo
```

Figura N° 36: Algoritmo de programación para la conversión a monoaural

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Posteriormente, durante el procesamiento, se comenzó con el algoritmo de extracción de la frecuencia de muestreo del archivo WAV, desde donde se puede editar la frecuencia de muestreo de ser necesario o cambiar parámetro del filtro pasabanda, tales como la frecuencia de corte inferior o superior, el rizado, y el orden del filtro Chebyshev. En la Figura N° 37 se muestra el algoritmo de parametrización.

```python
if __name__ == "__main__":
    #archivo = 'output.wav'
    archivo = 'AudiotecnologoVaron3.wav'
    audio = wave.open(archivo, 'rb')
    fs = audio.getframerate()
    audio.close()

    #fs = 44100 #Otra

    param = {'fs': fs,
             'lowcut':60,
             'highcut':3000,
             'ripple':8,
             'order':4,
             'type': 'Cheby'}
```

Figura N° 37: Algoritmo de programación de la parametrización

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Una vez decodificado, se procedió con la lectura del archivo de audio y la extracción de parámetros importantes a través del audioInfo como, la frecuencia de muestreo, el número de muestras; si es de 8 o 16 bits y el número de canales. En la Figura N° 38 se muestra el algoritmo para decodificar el archivo de audio.

```
    def decodificarAudio(self, archivo_entrada):
        #Abre audio y obtiene parámetros importantes
        audio = wave.open(archivo_entrada, 'rb')
        fs = audio.getframerate()
        nsamples = audio.getnframes()
        samplewidth = audio.getsampwidth()
        nchannels = audio.getnchannels()
        #Comprime la información en un diccionario
        audioInfo = {'fs':fs,
                     'nsamples':nsamples,
                     'samplewidth':samplewidth,
                     'nchannels':nchannels
            }
        #Abre vectores de tiempo y magnitud
        t = np.arange(0,nsamples)/fs
        signal = audio.readframes(-1)
        #Decodifica la magnitud según tamaño de muestra
        if samplewidth == 1:
            signal = np.array(np.frombuffer(
                signal, dtype='UInt8')-128, dtype='int8')
        elif samplewidth == 2:
            signal = np.frombuffer(signal, dtype='int16')
        #Separa los canales
        canales = [signal[idx::nchannels]
                           for idx in range(nchannels)]
        audio.close()
        return (t, canales, audioInfo)
```

Figura N° 38: Algoritmo para decodificar el archivo de audio.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Seguidamente, en el algoritmo de diseño del filtro pasabanda, se empieza a ejecutar el filtrado en formato WAV a través de los parámetros ya establecidos. En la Figura N° 39 se muestra el algoritmo de diseño del filtro en Python.

```
#Clase para diseñar y usar el filtro
class crearFiltro():
    #Atributos de clase
    modelo_sos = None
    fs = None

    #Diseña el filtro según tipo
    def __init__(self, param):
        self.fs = param['fs']
        if param['type'] == 'Cheby':
            self.modelo_sos = self.crearCheby(param)
        elif param['type'] == 'Butter':
            self.modelo_sos = self.crearButter(param)

    #Diseña el filtro Chebyshev tipo 1
    def crearCheby(self, param):
        nyquist = param['fs']*0.5
        corte = np.array([param['lowcut'],param['highcut']])/nyquist
        sos = signal.cheby1(param['order'],
                            param['ripple'],
                            corte,
                            analog=False,
                            btype='band',
                            output='sos')
        return sos
```

Figura N° 39: Algoritmo de diseño del filtro pasabanda

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- A continuación, se muestra el algoritmo en Python con el que se procedió a graficar en el tiempo las señales de audio original y filtrado, así mismo, también se realizaron los gráficos de los espectros en frecuencia para los dos casos. En la Figura N° 40 se muestra el algoritmo que realiza los gráficos en Python.

```
    def graficarOndas(self, t, datos, filtrados):

        plt.figure()
        plt.clf()
        plt.plot(t, datos, label='Señal original')
        plt.plot(t, filtrados, label='Señal filtrada')
        plt.xlabel('Tiempo (s)')
        plt.grid(True)
        plt.legend(loc='best')
        plt.show()

        plt.figure()
        plt.clf()
        N = len(filtrados)
        xf = np.linspace(0.0, self.fs/2, int(N/2))
        yf1 = fft(datos)
        plt.semilogx(xf, 2.0/N * np.abs(yf1[:N//2]), label='Señal original')
        plt.xlabel('Frecuencia (Hz)')
        plt.grid(True)
        plt.legend(loc='best')
        plt.show()

        plt.figure()
        plt.clf()
        yf2 = fft(filtrados)
        plt.semilogx(xf, 2.0/N * np.abs(yf2[:N//2]), label='Señal filtrada')
        plt.xlabel('Frecuencia (Hz)')
        plt.grid(True)
        plt.legend(loc='best')
        plt.show()

        plt.figure()
        plt.clf()
        plt.semilogx(xf, 2.0/N * np.abs(yf1[:N//2]), label='Señal original')
        plt.semilogx(xf, 2.0/N * np.abs(yf2[:N//2]), label='Señal filtrada')
        plt.xlabel('Frecuencia (Hz)')
        plt.grid(True)
        plt.legend(loc='best')
        plt.show()
```

Figura N° 40: Algoritmo de gráficos en lenguaje Python.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Luego se almacenó el archivo a través de la interfaz “cargar archivo WAV y realizar procesamiento” llamando al vector: “directorio_inicial”, el cual permite que seleccionemos un archivo WAV de un directorio y quede listo para filtrar. A continuación, confirmamos la opción de guardar el archivo WAV filtrado, creada con la librería tkinter.

  A continuación, en la figura N°41 se muestra la interfaz gráfica “Filtrar archivo WAV” que se usó para las pruebas iniciales.

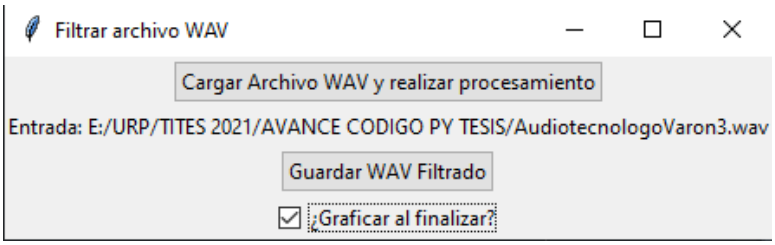

Figura N°41: Interface gráfica para cargar el audio de las indicaciones

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Finalmente, una vez confirmado en la interfaz gráfica “Guardar WAV filtrado” se visualiza la señal de audio original y filtrado en el tiempo, seguido del espectro en frecuencia de ambas señales. En la Figura N° 42 y en la Figura N° 43, se muestran las señales de audio en el tiempo y de los espectros en frecuencia respectivamente.

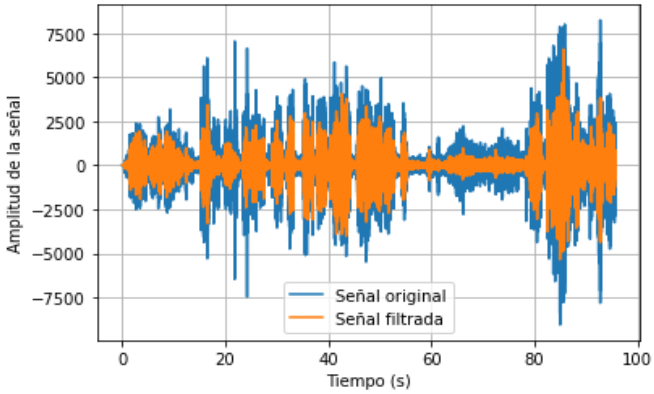

Figura N° 42: Gráfico de las señales de audio en el tiempo

Fuente: Captura de pantalla de software Anaconda entorno Spyder

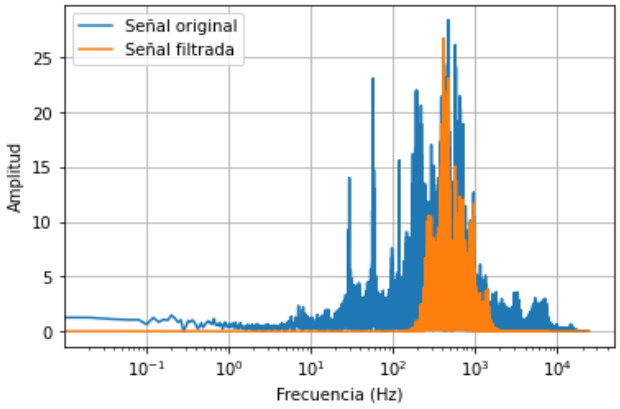

Figura N° 43: Gráfico de los espectros en frecuencia de las señales de audio

Fuente: Captura de pantalla de software Anaconda entorno Spyder

3.6.2. Aplicación del servicio Watson Speech to Text

- Se procedió a descargar las librerías de IBM Watson, Websocket-client y Pyaudio mediante el entorno de Anaconda Prompt. Una vez instaladas en el entorno Tesisurp2-SPYDER, se ejecutaron desde el archivo Python Proyecto_01.

  En la Figura N°44 se muestra el entorno de Anaconda Prompt al momento de descargar la librería websocket-client.

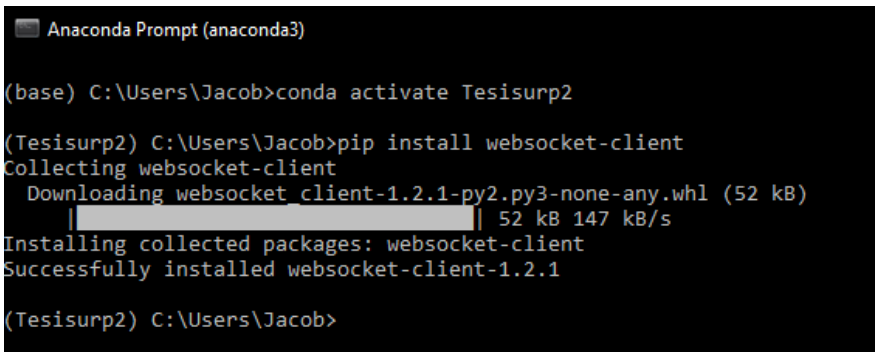

Figura N° 44: Gráfico descarga de la librería websocket-Client.

Fuente: Captura de pantalla de software Anaconda entorno Anaconda Prompt

- A continuación, se obtuvieron las identificaciones con el IBM Watson, que vienen a ser las cabeceras (headers) y el URL. Se creó un Websocket con tres parámetros: on_message, on_error, on_close para la comunicación. Luego se agregó la apertura de comunicación con el ws.on_open y el ws.run_forever, este último da la orden de envío datos (audio) a través del Websocket, y en simultáneo recibir datos (texto) del servidor del IBM Watson por transcripción. En la figura N°45 se muestra el algoritmo que realiza la comunicación del usuario y el servidor de IBM Watson.

```
164  ##FUNCION PRINCIPAL
165  def main():
166      #Comunicaciones
167      headers = libW.get_headers()
168      url = libW.get_url()
169
170      ws = websocket.WebSocketApp(url,
171                                  header=headers,
172                                  on_message=on_message,
173                                  on_error=on_error,
174                                  on_close=on_close)
175
176      ws.on_open = on_open
177      ws.run_forever()
178
179      print("**RESULTADO FINAL**")
180      transcript = ".".join([x['results'][0]['alternatives'][0]['transcript']
181                             for x in FINALS])
182      print(transcript)
183
184  if __name__ == "__main__":
185      main()
```

Figura N° 45: Gráfico de comunicación a través del Websocket

Fuente: Captura de pantalla de software Anaconda entorno Spyder.

- Luego se realizó la captura del audio a través de Pyaudio, se abrió el archivo y almacenó en la variable “stream”. Se creó la variable npData que recibe la información en número entero de 16 bits, el cuál pasa por el filtro y entrega la información filtrada en la variable data2, para luego ser almacenado en la variable arrOrig.append, quedando listo para ser enviado al servidor Speech to Text a través del Websocket. En la figura N°46 se muestra el algoritmo que realiza la captura del audio del usuario y el almacenamiento antes de enviarlo al IBM Watson.

```
## Función de grabación
def read_audio(ws):
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    fs = RATE
    filtro = crearFiltro(fs)
    print("*GRABANDO*")
    rec = RECORD_SECONDS

    for i in range(0, int(RATE / CHUNK * rec)):
        data = stream.read(CHUNK)
        #Lectura del buffer y filtrado
        npData = np.frombuffer(data, dtype=np.int16)
        npData = filtro.filtrarOnda(npData)
        npData = np.clip(npData, -CLIP_VAL, CLIP_VAL)
        #Reconversión al buffer
        data2 = npData.astype(np.int16).tobytes()
        #Llenado de arreglos de audio, original y filtrado
        arrOrig.append(np.frombuffer(data, dtype=np.int16))
        arrData.append(np.frombuffer(data2, dtype=np.int16))
        #Envío de datos para STT
        ws.send(data2, ABNF.OPCODE_BINARY)

    # Disconnect the audio stream
    stream.stop_stream()
    stream.close()
    print("***GRABACIÓN FINALIZADA***")
```

Figura N° 46: Gráfico del algoritmo de captura de audio y almacenamiento.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Seguidamente, se realizó el control de recepción del mensaje, apertura, cierre y error a través de los parámetros ya establecidos, del envío como respuesta desde el servidor de IBM Watson, hasta el lugar donde se ha realizado el envío de datos. En la figura N°47 se muestra el algoritmo que realiza la comunicación del usuario y el servidor de IBM Watson.

```
##Funciones relacionadas a comunicaciones
def on_message(self, msg):
    global LAST
    data = json.loads(msg)
    if "results" in data:
        if data["results"][0]["final"]:
            FINALS.append(data)
            LAST = None
        else:
            LAST = data
        #Imprime el fragmento actual
        print(data['results'][0]['alternatives'][0]['transcript'])

def on_error(self, error):
    print(error)

def on_close(ws):
    global LAST
    if LAST:
        FINALS.append(LAST)
    print("Conexión cerrada")


def on_open(ws):
    data = {
        "action": "start",
        "content-type": "audio/l16;rate=%d" % RATE,
        "continuous": True,
        "interim_results": True,
        "word_confidence": True,
        "timestamps": True,
        "max_alternatives": 3
    }
```

Figura N° 47: Algoritmo que realiza el control de recepción del servidor de IBM Watson.
Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Y por último, se esperó un tiempo de respuesta para la desconexión de la transmisión de la variable "data", y así enviar la información mediante el comando: ws.send(json.dumps(data).encode('utf8')), para el cierre de las comunicaciones. A continuación, en la figura N°48 se puede apreciar el código de programación previamente mencionado.

```
    # Disconnect the audio stream
    stream.stop_stream()
    stream.close()
    print("***GRABACIÓN FINALIZADA***")

    #Finaliza las comunicaciones
    data = {"action": "stop"}
    ws.send(json.dumps(data).encode('utf8'))
    #Espera un tiempo por la respuesta y cierra las comunicaciones
    time.sleep(3)
    ws.close()
    p.terminate()

    #Guarda los arreglos de ambos audios como wav
    #Y grafica sus espectros
    numOrig = aplanarLista(arrOrig)
    numData = aplanarLista(arrData)

    write('audioOrig.wav', fs, numOrig)
    write('audioFilt.wav', fs, numData)
```

Figura N° 48: Algoritmo que realiza la desconexión con el servidor de IBM Watson.
Fuente: Captura de pantalla de software Anaconda entorno Spyder

3.6.3. Implementación del algoritmo para la verificación de conversión de voz a texto

Definido el funcionamiento del algoritmo, se procedió a transcribirlo en lenguaje Python.

- En primer lugar, se creó una ventana principal con la librería tkinter, la cual tiene la función de interfaz y cuenta con un botón para grabar la voz. Para lograr esto, se creó una función llamada window1, luego se agregó el título "Traductor de voz a la lengua de señas peruana", se estableció el tamaño de la ventana y la posición dónde aparecerá, después se agregó un color de fondo "sky blue", se creó una etiqueta que dice "Bienvenido" y un botón que tiene como función grabar la voz.

  A continuación, en la figura N°49 se muestra el código de programación para la creación de la ventana principal y en la figura N°50 se muestra el resultado de la misma.

```
def window1():

    window.title("TRADUCTOR DE VOZ A LA LENGUA DE SEÑAS PERUANA")
    window.geometry('480x300+430+180')
    window.configure(background="sky blue")

    lbl = tk.Label(window, text="BIENVENIDO",bg="green", font = ("Arial Bold",16))
    lbl.pack(padx=20,pady=50)
    btn = tk.Button(window,text='Grabar voz',bg="green",font = ("Arial Bold",16), command=main)
    btn.pack()
```

Figura Nº 49: Lenguaje de programación para la ventana principal.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

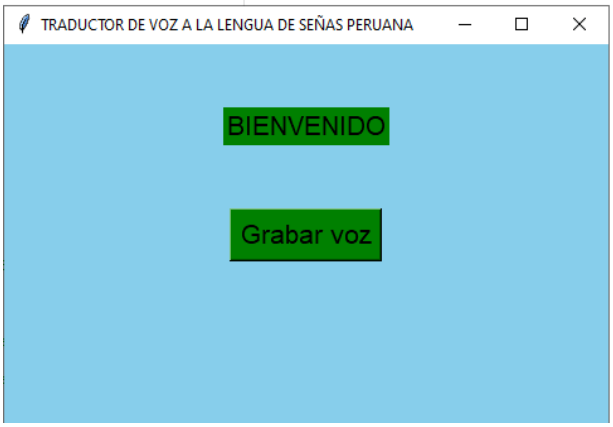

Figura Nº 50: Ventana principal.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Seguidamente, se almacenó el resultado de la conversión de voz a texto, en la variable “a”, luego creamos una segunda variable “b” que contenga la frase “Quisiste decir:” más el texto almacenado en la variable “a”. Todo esto se mostrará en una ventana de pregunta con respuesta “sí” y “no”, creada con la librería tkinter.

  A continuación, en la figura N°51 se muestra el código fuente para almacenamiento de la variable y el uso de la misma. Así mismo en la figura N°52 se muestra el resultado de la ventana de para verificación.

```
a = TextFinal
b = "Quisiste decir: " + a
respuesta=messagebox.askquestion('Verificación de traducción', b )
```

Figura Nº 51: Almacenamiento del texto convertido en variable.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

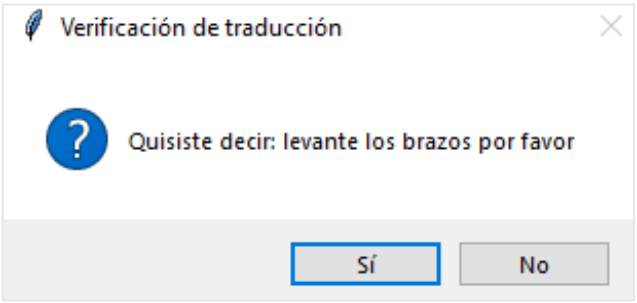

Figura Nº 52: Ventana de verificación.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

- Luego, se usó los condicionales “if” y “elif”, siendo el “if” si la respuesta es “Sí” y “elif” si la respuesta es “No”. Si la respuesta es “Sí”, ejecuta la animación correspondiente al texto convertido, siempre y cuando la animación esté dentro de la base de datos, caso contrario muestra un mensaje diciendo “Frase sin animación 3D” y finalmente limpia la lista FINALS, si la respuesta es “No”, ejecuta una ventana de advertencia con un texto diciendo “VUELVA A GRABAR” y limpia la lista FINALS.

  Seguidamente, en la figura N°53 se muestra el código de programación para una respuesta afirmativa, en la figura N°54 se muestra el código de programación si la respuesta es negativa y en la figura N°55 se muestra la ventana de advertencia consecuente de la misma.

```
if respuesta == "yes":

    if a == "buenos días " or "buen día " or "un buen día " or "muy buenos días " or "buen días ":
        anim_01()
        FINALS.clear()
    elif a == "buenas tardes " or " buen tarde " or "muy buenas tardes " or "muy buen tarde ":
        anim_02()
        FINALS.clear()
    elif a == "buenas noches " or "muy buenas noches " or "buen noche " or "muy buen noche ":
        anim_03()
        FINALS.clear()
    elif a == "aguante la respiración " or "aguante respiración " or " en aguante respiración ":
        anim_04()
        FINALS.clear()
    elif a == "levante los brazos " or "levante brazo " or "eleve los brazos " or "suba los brazos ":
        anim_05()
        FINALS.clear()
    elif a == "ya puede respirar " or "respire " or "tome aire " or "puede respirar ":
        anim_06()
        FINALS.clear()
    elif a == "quítese las prendas de la parte superior " or "quítese el polo " or "quítese la blusa " :
        anim_07()
        FINALS.clear()
    elif a == "ya terminamos gracias " or "terminamos " or "eso es todo " or "eso es todo gracias ":
        anim_08()
        FINALS.clear()
    elif a == "colóquese ahí " or "en colóquese ahí " or "colóquese aquí " or "colóquese aca " or "póngase aca " or "
        anim_09()
        FINALS.clear()
    elif a == "ya puede retirarse " or "puede retirarse " or "ya puede irse " or "puede irse ":
        anim_10()
        FINALS.clear()
    else:
        messagebox.showwarning('ADVERTENCIA', 'ESA FRASE NO ESTÁ EN LA BASE DE DATOS')
        FINALS.clear()
elif respuesta == "no":
    messagebox.showwarning('ADVERTENCIA', 'VUELVA A GRABAR')
    FINALS.clear()
```

Figura Nº 53: Algoritmo si la respuesta es “Sí”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

```
elif respuesta == "no":
    messagebox.showwarning('ADVERTENCIA', 'VUELVA A GRABAR')
    FINALS.clear()
```

Figura Nº 54: Algoritmo si la respuesta es “No”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

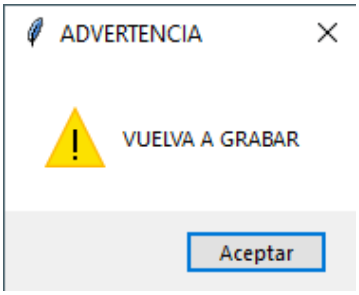

Figura Nº 55: Ventana de advertencia.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

3.6.4. Algoritmo de ejecución de las animaciones tridimensionales

Luego de renderizar las animaciones en formato .avi, estas se almacenaron en una carpeta, desde dónde son ejecutadas con la librería “opencv” desde Python.

Seguidamente, se crearon 10 funciones llamada anim_01, anim_02, anim_10, y se asignó el video “nombre del video” a la variable cap. Posteriormente, se creó un condicional para asegurar que existe el video dentro de la carpeta, luego comienza a capturar el video frame por frame, y lo reproduce hasta que se presione la tecla “q” o termine la reproducción del video o no encuentre el video dentro de la carpeta, finalmente libera la variable y cierra la ventana de la animación. Este proceso se repitió para todas las animaciones, lo único que varió fue el nombre del video a ejecutar. A continuación, en la figura N°56 se muestra el código de programación para la ejecución de las animaciones, este código es repetitivo a diferencia del nombre del video a ejecutar.

```
def anim_01 ():
    cap = cv2.VideoCapture('buendia.avi')

    if (cap.isOpened()== False):
        print("Error opening video stream or file")

    while (cap.isOpened()):
        ret, frame = cap.read()

        if ret == True:
            cv2.imshow('Frame', frame)

            if cv2.waitKey(40) & 0xFF == ord('q'):
                break

        else:
            break

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

Figura Nº 56: Algoritmo para reproducir animaciones.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

CAPÍTULO IV: PRUEBAS Y RESULTADOS

En este capítulo, se realizó las cuatro pruebas con los tecnólogos y sus respectivos resultados; con porcentajes de reconocimiento en función al número de letras, que detecta el IBM Watson al realizar la transcripción. Así mismo, en las pruebas realizadas se utilizó una de las salas de Rayos X del Hospital Nacional Hipólito Unánue con la consola o mando de control (donde se ubica el tecnólogo), para poder verificar la eficiencia del filtro pasabanda Chebyshev en el presente trabajo de investigación. Para todos los resultados de la transcripción, se utilizó la fórmula de porcentaje (%) de reconocimiento (ver la figura N°57); la cual se representa como el resultado de substraer al número 100, la diferencia entre el número de letras de la frase correctamente expresada por el tecnólogo y el número de letras de la frase reconocida por la librería Speech To Text (STT), ambas divididas entre el número de letras de la frase correctamente expresada por el tecnólogo y, multiplicado por 100. Así mismo, los resultados de las pruebas son mostrados más adelante en las tablas siguientes.

$$(\%)Reconocimiento = \left[100 - \left(\frac{N^{o}\ de\ letras\ de\ frase\ correcta - N^{o}\ de\ letras\ de\ frase\ correcta\ obtenida\ STT}{N^{o}\ de\ letras\ de\ frase\ correcta}\right) \times 100\right]$$

Figura Nº 57: Fórmula para calcular porcentaje de Reconocimiento de frase del STT.
Fuente: Elaboración Propia

4.1. Detección de las indicaciones del primer tecnólogo varón

Primero, se procedió a realizar pruebas del prototipo con el primer tecnólogo varón, con el propósito de ver la eficiencia del Speech to Text de IBM Watson, para lograr el reconocimiento de frases correctas al momento de realizar la transcripción.

A continuación, en la tabla N°1, se muestra en la primera columna las frases en texto del primer tecnólogo varón, en la segunda columna la detección de las frases de reconocimiento del IBM Watson y en la tercera columna el porcentaje de reconocimiento en función al número de letras del Speech to Text.

Tabla Nº1: Frases del primer tecnólogo varón en texto, frases de reconocimiento y porcentaje de reconocimiento del Speech to Text.

Indicaciones en texto del primer tecnólogo varón	**Frase de reconocimiento del IBM Watson**	**Porcentaje de reconocimiento en función al número de letras del STT**
Buenos días	Bueno bien	50%
Buenas tardes	Buenas tardes	100%
Buenas noches	Buenas noches	100%
Póngase ahí	Póngase en	70%
Contenga la respiración	Contenga la respiración	100%
Ya puede respirar	Ya puede respirar	100%
Quítese las prendas de la parte superior	quítense la prenda de la parte superior	82.86%
Levante los brazos	Levante los brazos	100%
Ya terminamos gracias	Ya terminamos gracias	100%
Se puede retirar	Se puede retirar	100%

Fuente: Elaboración propia.

Seguidamente, se muestra una captura de pantalla de la consola de Python cuando se realizó el reconocimiento del STT de IBM Watson al 50%.

```
In [1]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
*GRABANDO*
bueno bien
bueno bien
***GRABACIÓN FINALIZADA***
Conexión cerrada
**RESULTADO FINAL**
bueno bien
```

Figura Nº 5*8*: Captura de pantalla del código de la consola de Python de la indicación “buenos días”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

Así mismo, se muestran las figuras N° 59 y 60 los dos casos adicionales cuando se realizó el reconocimiento del STT de IBM Watson al 70 % y al 82.86%.

```
In [2]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
Reloaded modules: libWatson, claseFiltro
*GRABANDO*
pónganse en
pónganse en
***GRABACIÓN FINALIZADA***
Conexión cerrada
**RESULTADO FINAL**
pónganse en
```

Figura Nº 59: Ventana de Python de la indicación *“p*óngase ahí*”*.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

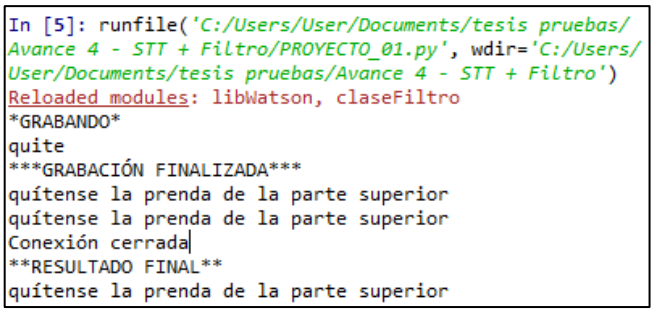

Figura Nº 60: Ventana de Python de la indicación "Quítese la prenda de la parte superior".

Fuente: Captura de pantalla de software Anaconda entorno Spyder

4.2. Detección de las indicaciones del segundo tecnólogo varón

Para el segundo tecnólogo Varón, también se procedió a realizar las pruebas de su voz con el prototipo, con el propósito de ver la eficiencia del Speech to Text de IBM Watson.

A continuación, en la Tabla N° 2, se muestra en la primera columna las frases en texto del segundo tecnólogo varón, en la segunda columna la detección de las frases de reconocimiento del IBM Watson y en la tercera columna el porcentaje de reconocimiento en función al número de letras del Speech to Text (STT).

Tabla Nº2: Frases del segundo tecnólogo varón en texto, frases de reconocimiento y porcentaje de reconocimiento del Speech to Text.

Indicaciones en texto del segundo tecnólogo varón	**Frase de reconocimiento del IBM Watson**	**Porcentaje de reconocimiento en función al número de letras del STT**
Buenos días	Buenos días	100%
Buenas tardes	Buen tarde	91.67%
Buenas noches	Buenas noches	100%
Póngase ahí	Póngase ahí	100%
Contenga la respiración	Condenó la respiración	76.20%
Ya puede respirar	Ya puede respirar	100%
Quítese las prendas de la parte superior	Quítense la pena en la parte superior	68.58%
Levante los brazos	Levante los brazos	100%
Ya terminamos, gracias	Ya terminamos gracias	100%
Se puede retirar	Se puede retirar	100%

Fuente: Elaboración propia.

A continuación, en la figura N°61 se muestra una captura de pantalla de la consola de Python cuando se realizó el reconocimiento del STT de IBM Watson al 91.67%.

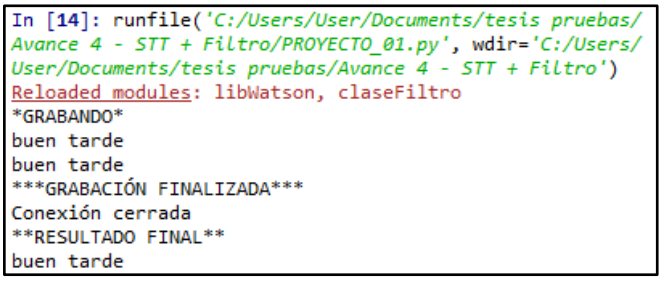

Figura Nº61: Ventana de Python de la indicación “buenas tardes”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

Así mismo, se muestran las figuras N° 62 y 63 de los dos casos adicionales cuando se realizó el reconocimiento del STT de IBM Watson al 76.20% y al 68.58%.

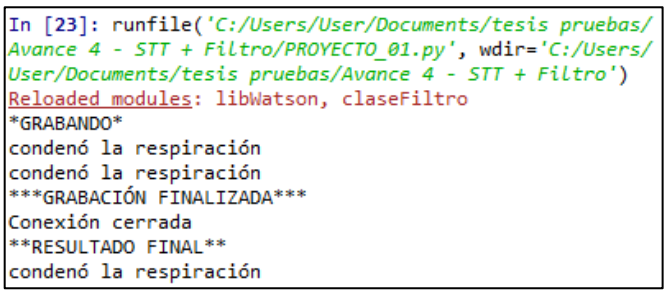

Figura Nº 62: Ventana de Python de la indicación “contenga la respiración”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

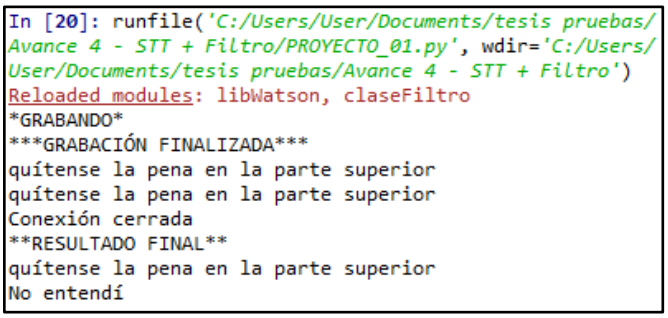

Figura Nº 63: Ventana de Python de la indicación “quítese las prendas de la parte superior”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

4.3. Detección de las indicaciones de la primera tecnóloga mujer

Seguidamente, con la primera tecnóloga mujer, se procedió a realizar las pruebas de su voz con el prototipo, con el propósito de ver la eficiencia del Speech to Text de IBM Watson.

A continuación, en la Tabla N° 3, se muestra en la primera columna las frases en texto del tecnólogo N°03, en la segunda columna la detección de las frases de

reconocimiento del IBM Watson y en la tercera columna el porcentaje de reconocimiento en función al número de letras del Speech to Text.

Tabla Nº3: Frases de la primera tecnóloga mujer en texto, frases de reconocimiento y porcentaje de reconocimiento del Speech to Text.

Indicaciones en texto de la primera tecnóloga mujer	**Frase de reconocimiento del IBM Watson**	**Porcentaje de reconocimiento en función al número de letras del STT**
Buenos días	Buenos días	100%
Buenas tardes	Buenas tardes	100%
Buenas noches	En noches	50%
Póngase ahí	Póngase ahí	100%
Contenga la respiración	Contenga la respiración	100%
Ya puede respirar	Ya puede respirar	100%
Quítese las prendas de la parte superior	Quítense las prendas de la parte superior	97.15%
Levante los brazos	Levante los brazos	100%
Ya terminamos, gracias	Ya terminamos gracias	100%
Se puede retirar	Se puede retirar	100%

Fuente: Elaboración propia.

A continuación, se muestra una captura de pantalla de la consola de Python cuando se realizó el reconocimiento del STT de IBM Watson al 50%.

```
In [27]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
Reloaded modules: libWatson, claseFiltro
*GRABANDO*
en noches
en noches
***GRABACIÓN FINALIZADA***
Conexión cerrada
**RESULTADO FINAL**
en noches
```

Figura Nº 64: Ventana de Python de la indicación “buenas noches”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

Así mismo, se muestra 01 caso adicional cuando se realizó el reconocimiento del STT de IBM Watson al 97.15%.

```
In [30]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
Reloaded modules: libWatson, claseFiltro
*GRABANDO*
***GRABACIÓN FINALIZADA***
quítense las prendas de la parte superior
quítense las prendas de la parte superior
Conexión cerrada
**RESULTADO FINAL**
quítense las prendas de la parte superior
```

Figura Nº65: Ventana de Python de la indicación. “quítese las prendas de la parte superior”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

4.4. Detección de las indicaciones de la segunda tecnóloga mujer

Y finalmente, con la segunda tecnóloga mujer, se procedió a realizar las pruebas de su voz con el prototipo, con el propósito de ver la eficiencia del Speech to Text de IBM Watson.

A continuación, en la Tabla N°4, se muestra en la primera columna las frases en texto del tecnólogo N° 04, en la segunda columna la detección de las frases de reconocimiento del IBM Watson y en la tercera columna el porcentaje de reconocimiento en función al número de letras del Speech to Text.

Tabla Nº4: Frases de la segunda tecnóloga mujer en texto, frases de reconocimiento y porcentaje de reconocimiento del Speech to Text.

Indicaciones en texto de la segunda tecnóloga mujer	**Frase de reconocimiento del IBM Watson**	**Porcentaje de reconocimiento en función al número de letras del STT**
Buenos días	Buenos días	100%
Buenas tardes	Buenas tardes	100%
Buenas noches	Buenas noches	100%
Póngase ahí	Póngase ahí	100%
Contenga la respiración	Contenga la respiración	100%
Ya puede respirar	Yo puede respirar	93.34%
Quítese las prendas de la parte superior	Quítese las prendas de la parte superior	100%
Levante los brazos	Levante los brazos	100%
Ya terminamos, gracias	Ya terminamos gracias	100%
Se puede retirar	Se puede retirar	100%

Fuente: Elaboración propia.

A continuación, se muestra una captura de pantalla de la consola de Python cuando se realizó el reconocimiento del STT de IBM Watson al 100%.

```
In [35]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
Reloaded modules: libWatson, claseFiltro
*GRABANDO*
póngase ahí
póngase ahí
***GRABACIÓN FINALIZADA***
Conexión cerrada
**RESULTADO FINAL**
póngase ahí
```

Figura Nº 66: Ventana de Python de la indicación “póngase ahí”.

Fuente: Captura de pantalla de software Anaconda entorno Spyder

Así mismo, se muestra 01 caso adicional cuando se realizó el reconocimiento del STT de IBM Watson al 93.34%.

```
In [36]: runfile('C:/Users/User/Documents/tesis pruebas/
Avance 4 - STT + Filtro/PROYECTO_01.py', wdir='C:/Users/
User/Documents/tesis pruebas/Avance 4 - STT + Filtro')
Reloaded modules: libWatson, claseFiltro
*GRABANDO*
yo puede respirar
yo puede respirar
***GRABACIÓN FINALIZADA***
Conexión cerrada
**RESULTADO FINAL**
yo puede respirar
```

Figura Nº 67: Ventana de Python de la indicación "ya puede respirar".

Fuente: Captura de pantalla de software Anaconda entorno Spyder

4.5. Comparación de las indicaciones de los cuatro tecnólogos

En estas pruebas realizadas, se repitieron las frases en texto de los 04 tecnólogos. Cabe resaltar que estas pruebas se realizaron con ruido ambiental de la sala de rayos X y de acuerdo a la disponibilidad de tiempo de cada tecnólogo. El porcentaje de efectividad del Speech to Text.

Tabla Nº5: Cantidad total de las frases de los tecnólogos, resultados obtenidos y porcentaje de efectividad del Speech to Text.

Frases de reconocimiento del primer tecnólogo varón	**R. (%)**	**Frases de reconocimiento del segundo tecnólogo varón**	**R. (%)**	**Frases de reconocimiento del primer tecnólogo mujer**	**R. (%)**	**Frases de reconocimiento del segundo tecnólogo mujer**	**R. (%)**	**Porcentaje de efectividad del STT**
Buenos días	50	Buenos días	100	Buenos días	100	Buenos días	100	**87.5%**
Buenas tardes	100	Buenas tardes	91.67	Buenas tardes	100	Buenas tardes	100	**97.91%**
Buenas noches	100	Buenas noches	100	Buenas noches	50	Buenas noches	100	**87.5%**
Póngase ahí	70	Póngase ahí	100	Póngase ahí	100	Póngase ahí	100	**92.5%**
Contenga la respiración	100	Contenga la respiración	76,20	Contenga la respiración	100	Contenga la respiración	100	**94.05%**
Ya puede respirar	100	Ya puede respirar	100	Ya puede respirar	100	Ya puede respirar	93.34	**98.34%**

Quítese las prendas de la parte superior	82.86	Quítese las prendas de la parte superior	68.6	Quítese las prendas de la parte superior	97.20	Quítese las prendas de la parte superior	100	**87.17%**
Levante los brazos	100	Levante los brazos	100	Levante los brazos	100	Levante los brazos	100	**100%**
Ya terminamos, gracias	100	Ya terminamos, gracias	100	Ya terminamos, gracias	100	Ya terminamos, gracias	100	**100%**
Se puede retirar	100	Se puede retirar	100	Se puede retirar	100	Se puede retirar	100	**100%**

Fuente: Elaboración propia.

De estas pruebas realizadas con el prototipo, se concluye que el porcentaje mínimo de efectividad del Speech to Text obtenido fue del 87.17%, esto considerando un ambiente con ruido real en una sala de rayos X, además se obtuvo también un porcentaje máximo de efectividad del Speech to Text que obtuvimos es del 100% debido a la claridad en la pronunciación que se dieron en el momento de la grabación.

Por otro lado, en los diferentes casos de detección, mientras se realizaban las pruebas con los tecnólogos se emitían ruido por personas que transitaban en el pasadizo del Hospital, y ruido de otros equipos de rayos X en funcionamiento cercano a la sala de rayos X , siendo estos irrelevantes gracias a la eficacia del filtro al rechazar estas frecuencias puesto que el prototipo llegó a realizar la traducción de voz a texto de forma correcta y se pudo ejecutar la animación tridimensional en Lengua de Señas Peruana del avatar de forma exitosa. Así mismo, al realizar las pruebas con el prototipo se tuvo una limitación en la toma de datos y en la cantidad de los tecnólogos para poder realizar las pruebas, a esto se suma el poco tiempo libre con el que cuentan, razón por la cual solo se pudo realizar una repetición de cada palabra, es por eso que se logró acceder a la toma de datos de cuatro tecnólogos con las frases mencionadas.

4.6. Imágenes de los tecnólogos dando indicaciones en la sala de Rayos X

En esta prueba se realizó teniendo a los tecnólogos a una cercanía de 10 centímetros aproximadamente del micrófono al momento realizar la toma de datos, los valores obtenidos se encuentran en las tablas N°1 y N°2. Y como se aprecia en la figura N°68 la visibilidad del prototipo y la funcionalidad cuando se muestra cada

animación, en la figura izquierda está el primer tecnólogo varón y a la derecha el segundo tecnólogo varón.

Figura Nº 68: Toma de fotografías durante las pruebas

Fuente: Elaboración propia

Así mismo, los valores adquiridos se encuentran en las tablas N° 3 y N° 4. Y como se aprecia en la figura N°69 la visibilidad del prototipo y la funcionalidad cuando se muestra cada animación, en la figura izquierda está la primera tecnóloga mujer y a la derecha la segunda tecnóloga mujer.

Figura Nº 69: Toma de fotografías durante las pruebas

Fuente: Elaboración propia

4.7. Presupuesto

Para el desarrollo de esta tesis se necesitó de un micrófono externo con la finalidad de mejorar la etapa de adquisición de la voz; el costo aproximado de acuerdo al mercado actual se muestra en la siguiente Tabla N°6.

Tabla Nº6: Cantidad total de material utilizado.

Dispositivos	**Precio (S/.)**
Micrófono de condensador con conector 3.5mm	45.00
TOTAL	**45.00**

Fuente: Elaboración propia.

CONCLUSIONES

1. Se realizó el diseño de un filtro digital pasabanda Chebyshev de orden 8 con frecuencias de corte de 250 a 3000 Hz en el software de Matlab como se muestra en la figura N°3, el cual cumple con la función de atenuar los ruidos generados en la sala de radiografías, por esta razón se realizaron pruebas en el hospital Hipólito Unanue, posteriormente el filtro diseñado fue implementado utilizando el lenguaje de programación Python, teniendo como su respuesta gráfica en función de la frecuencia y el tiempo, como se muestra en las figuras N°42 y 43.

2. Se convirtió la voz en texto usando el servicio de IBM, Watson Speech To Text, el mismo fue implementado y ejecutado desde Python como se muestra desde la figura N°44 hasta la figura N°48. Así mismo, se implementó el algoritmo en lenguaje Python para la verificación de conversión de voz a texto, el cual contó con un menú interactivo desarrollado con la librería tkinter para un uso más intuitivo y amigable con el tecnólogo, el cual muestra un mensaje de confirmación para la correcta traducción, mensaje de advertencia para volver a hablar y un menú principal, como se mostró desde la figura N°49 hasta la figura N°54.

3. Se realizó la caracterización del avatar como se muestra en la figura N°21 y las animaciones tridimensionales de la Lengua de Señas Peruana como se muestra en las figuras N°31, N°32 y N°33, las cuales fueron conformadas de un total de 10 frases enfocadas para una toma de Rayos X, estas frases fueron traducidas con la ayuda del libro Lengua de Señas Peruana, la cual es una guía para el aprendizaje del mismo. Todas las animaciones se realizaron a través del software DAZ Studio y se almacenaron en formato AVI, para posteriormente ser ejecutados desde Python.

RECOMENDACIONES

1. Se recomienda una pronunciación clara y pausada de las indicaciones, y el uso de un micrófono externo para la adquisición de la voz que cuente con supresión de ruido, con la finalidad de reducir el error de conversión de voz a texto.
2. Se recomienda el uso del algoritmo de verificación para evitar dar mensajes erróneos al paciente.
3. Se recomienda contar con el apoyo de un traductor especializado en la Lengua de Señas Peruana, para poder realizar la animación de cualquier indicación realizada por los tecnólogos.
4. Se recomienda contar con una computadora de alta gama enfocada en el diseño gráfico para el proceso de animación tridimensional y renderizado de la Lengua de Señas Peruana.

REFERENCIAS BIBLIOGRÁFICAS

B. A. Shenoi (2006) *Introducción al procesamiento digital de señales y al diseño de filtros*. Editorial Wiley-Interciencia. (EEUU).

Bhaskar C.(2015) *Planos de desarrollo de aplicaciones de GUI de Tkinter*. Editorial Packt Open source. (Reino Unido).

Carguacundo M., Constante P. (2018) *Traductor de texto y voz a lengua de señas ecuatoriana a través de un avatar implementado para dispositivos Android* Proyecto de Tesis. Universidad de las Fuerzas Armadas ESPE Extensión Latacunga. Ecuador (pp. 1-6)IEEE.

Chamorro E., Tascón A.(2020) "RADIOLOGIA" *Diagnóstico Radiológico del paciente con COVID-19*. Uruguay.

Ciccone P. (2013). *La guía completa de DAZ Studio 4: Da vida a tus personajes 3D con DAZ Studio*. Editorial publicación de Packet. (Reino Unido).

Dirección General Básica de Educación Especial (2015) "*Guía para el Aprendizaje de Lengua de Señas Peruana"*.Ministerio de Educación. Perú.

Gary S.(2020) *Programación práctica de Python para IOT: Proyectos avanzados de IOT utilizando Raspberry Pi 4, MQTT, API RESTFUL, WebSockets y Python 3.* Editorial Publicación de Packt. (Reino Unido).

Gonzales R. (2018) *Python para Todos*. Mi primer programa (ESPAÑA).

Grayson J.(2000) *Programación en Python y Tkinter*. Editorial Publicaciones de Manning. (MEXICO).

IBM (2020).*Building Cognitive Aplication with IBM Watson Services: Volume 6 Speech to text and text to Speech*. IBM Redbooks (EEUU).

Les Pardew (2009) *Figures, Characters, And Avatars: THE OFFICIAL GUIDE TO USING DAZ STUDIO TO CREATE BEAUTIFUL ART*. Course Technology PTR Cengage Learning.(EEUU).

Lombardi A.(2015) *WebSocket: Comunicaciones cliente-servidor ligero*. Editorial Oreilly.(EEUU).

Navin S., Sudipta B., Neha (2020) *Desarrollo de Bots cognitivos utilizando IBM Watson Engine*. Editorial APRESS.(EEUU).

Pérez C. (2002) *MATLAB y sus aplicaciones en las ciencias y en la Ingeniería*.Pearson Educación S.A(MADRID – ESPAÑA).

Sarkar S., Mehedi J.(2018) *Estudio integral para la selección del filtro IIR adecuado: especificaciones de Enfoque dependiente*. 2018 3rd International Conference for Convergence in Technology (I2CT)(pp. 1-7)IEEE.

Schuller G. (2020) *Bancos de filtros y codificación de audio: comprimir señales de audio usando Python.* Editorial Springer International Publishing (SUIZA).

Steven I., Brian G. (2017). *Introducción al modelado y la simulación con MATLAB y Python*. Editorial CRC Press. (EEUU).

Tapia V, Reinoso R., Carrillo E. (2017) *Sistema de traducción de voz y texto a lenguaje de señas*. Proyecto de Tesis. Universidad Técnica de Cotopaxi Facultad de ciencias de la Ingeniería y Aplicadas. Ecuador.)(pp. 1-5)IEEE.

Taylor F.(2012). *Filtros Digitales: principios y aplicaciones con MATLAB.* Editorial Prensa Wiley-IEEE (Canadá).