UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DESARROLLO E IMPLEMENTACIÓN DE UN PROTOTIPO CON TECNOLOGÍA ASISTIVA PARA AUDIOLECTURA DIRIGIDO A PERSONAS CON DISCAPACIDAD VISUAL DE LA INSTITUCIÓN UNIDAD ESPECIALIZADA MARIANA DE JESÚS

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Electrónica y Automatización

AUTOR: Henry Gabriel Criollo Oña

Anthony Ariel Urresta Gaona

TUTOR: Andrés Sebastián Calero Calero

Quito-Ecuador

2026

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Henry Gabriel Criollo Oña con documento de identificación N° 1725599136 y Anthony Ariel Urresta Gaona con documento de identificación N° 2250101827; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 6 de abril del año 2026

Atentamente,

Henry Gabriel Criollo Oña

1725599136

Anthony Ariel Urresta Gaona

2250101827

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Henry Gabriel Criollo Oña con documento de identificación N° 1725599136 y Anthony Ariel Urresta Gaona con documento de identificación N° 2250101827 , expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Desarrollo e implementación de un prototipo con tecnología asistiva para audiolectura dirigido a personas con discapacidad visual de la Institución Unidad Especializada Mariana de Jesús", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica y Automatización, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 6 de abril del año 2026

Atentamente,

Henry Gabriel Criollo Oña

1725599136

Anthony Ariel Urresta Gaona

2250101827

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Andrés Sebastián Calero Calero con documento de identificación N° 1719252346, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO E IMPLEMENTACIÓN DE UN PROTOTIPO CON TECNOLOGÍA ASISTIVA PARA AUDIOLECTURA DIRIGIDO A PERSONAS CON DISCAPACIDAD VISUAL DE LA INSTITUCIÓN UNIDAD ESPECIALIZADA MARIANA DE JESÚS, realizado por Henry Gabriel Criollo Oña con documento de identificación N° 1725599136 y Anthony Ariel Urresta Gaona con documento de identificación N° 2250101827, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 6 de abril del año 2026

Atentamente,

Ing. Andrés Sebastián Calero Calero, MSc

1719252346

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

Desarrollo de un prototipo de tecnología asistiva para audiolectura dirigido a personas con discapacidad visual de la Unidad Especializada Mariana de Jesús en Quito, Ecuador el cual consiste en un sistema de gafas inteligentes con una cámara ArduCam de 64MP conectada a una Raspberry Pi 4 con el fin de capturar texto de un docmento y convertirlo en audio mediante tecnologías de reconocimiento de voz, OCR y síntesis de voz. Las pruebas realizadas demuestran que el prototipo puede reconocer comandos de voz y transformar texto impreso en información audible.

ABSTRACT

This thesis presents the development of an assistive technology prototype for audio reading designed for visually impaired users from the Specialized Educational Unit Mariana de Jesús in Quito, Ecuador. The system consists of smart glasses equipped with an ArduCam camera connected to a Raspberry Pi 4, capable of capturing printed text and converting it into audio using voice recognition, OCR, and text-to-speech technologies. Tests show that the prototype can recognize voice commands and transform printed text into audible information.

INTRODUCCIÓN

Desde que se nace, la visión es el sentido más dominante de los 5 sentidos, ya que juega un papel importante dentro del desarrollo, la comunicación, el aprendizaje y la participación social. En este sentido la Organización Mundial de la Salud [OMS] (2023) señala que más de 2.2 mil millones de personas padecen algún grado de discapacidad visual o ceguera, de las cuales alrededor de 1 mil millones pudo haber prevenido o aún no ha tratado su condición. Además, que la discapacidad visual afecta de forma negativa, la capacidad de leer, estudiar y de acceder a información, donde estas barreras tienen efectos directos sobre la inclusión educativa y laboral.

En este contexto, en el Ecuador existen alrededor de 57.384 personas aproximadamente con discapacidad visual registrada hasta junio del 2025 (Consejo Nacional para la Igualdad de Discapacidades, 2026). Como búsqueda de una solución a esta problemática, el presente proyecto propone la creación de un asistente visual para audio lectura, diseñado para personas con discapacidad visual, que pueda capturar texto impreso del entorno y convertirlo en una narración de audio de manera eficiente por medio de la combinación de hardware y software especializados.

El alcance del proyecto se centra en la construcción de un prototipo que consiste en unas gafas inteligentes equipadas con una cámara de alta resolución (ArduCam 64 MP), la cual se encuentra conectada a una microcomputadora Raspberry Pi 4 de 8 GB. En este sentido, el sistema se encargará de capturar imágenes de texto del entorno del usuario.

El procesamiento principal se realizará en la Raspberry Pi, con el apoyo de servicios en la nube. A partir de un Modelo de Lenguaje Visual (VLM) avanzado fue realizado la extracción del contenido textual de las imágenes (un proceso de OCR) y un sistema de Texto-a-Voz (TTS) para generar el audio. La interacción del usuario con el dispositivo se realizará mediante comandos de voz, captados por un micrófono integrado en unos auriculares de conducción ósea, lo que permitirá a la persona operar el asistente con instrucciones verbales simples como “leer documento” o “salir”.

Finalmente, el audio generado se reproducirá luego de que el texto obtenido sea procesado por un sistema Texto a voz que se ejecuta en una API (TTS) a través de los mismos

auriculares, permitiendo al usuario escuchar la lectura del texto, el alcance de esta fase se limita a la lectura de texto impreso en español.

CAPÍTULO 1

PROBLEMA

En la actualidad, el acceso a la información escrita representa un desafío para las personas con discapacidad visual, con repercusiones directas en su desarrollo educativo, profesional y personal. Saiz de la Hoya (2024) menciona que se estima que, más de 285 millones de personas viven con alguna discapacidad visual, de las cuales, entre 39 y 43 millones, experimentan ceguera total.

Asimismo, los autores resaltan que, dentro del ámbito de la lectoescritura, el sistema Braille ha sido la principal herramienta utilizada por la población, sin embargo, no es universal. En este sentido, algunas de las razones que limitan el uso del Braille dentro de la población ciega, incluyen: dificultades cognitivas o motoras que obstaculizan su aprendizaje, falta de acceso a una formación especializada a temprana edad o incluso desinterés o desmotivación para aprenderlo en la adultez

Como resultado de esta situación, existe un alto índice de analfabetismo funcional dentro de la comunidad ciega, generado tanto por la falta de dominio del Braille como por la falta de alternativas de lectura (Unión Latinoamericana de Ciegos, 2023). Esta situación genera la creación de una importante brecha dentro del acceso al conocimiento, limitado tanto la autonomía y la inclusión social plena de esta población.

En este contexto, incluso para aquellos que dominan el Braille, persisten diferentes barreras estructurales. Saira & Escobedo (2025) mencionan que una de ellas es la producción de materiales, ya que este tipo de sistemas se asocian con un proceso costoso y de alcance limitado por la necesidad de aplicar una técnica especializada y la necesidad de un mayor volumen de papel que incrementa de forma marcada el precio final del libro al compararse con los libros convencionales de tinta. Como resultado, dentro de la región se ha identificado la escasez de producción Braille y la casi nula producción de material en relieve en numerosos países de la región, reduciendo las oportunidades educativas, culturales y de acceso a información vigente para las personas ciegas, haciendo referencia que menos del 25% de las bibliotecas públicas ofrecen libros en Braille (Unión Latinoamericana de Ciegos, 2023).

En este contexto, Calle et al. (2021) señalan que la falta de materiales y herramientas inclusivas agudiza la exclusión social de las personas con discapacidad visual, ya que se reduce el acceso a la información en formato físico, además que no se cuenta con recursos adaptados, reduciendo la posibilidad de las personas con discapacidad visual de adquirir una formación académica de calidad, de competir en el mercado laboral y de vivir de forma autónoma.

Asimismo, los autores mencionan que este problema se agrava por la falta de tecnologías de asistencias que sean accesibles económicamente. Si bien existen en el mercado, dispositivos y aplicaciones diseñadas para apoyar a las personas ciegas, estos no se ajustan a la realidad económica de los usuarios en países en desarrollo.

Los autores continúan señalando que problema se ve agravado por la ausencia de tecnologías de asistencia que sean económicamente asequibles. Si bien existen en el mercado dispositivos y aplicaciones diseñadas para apoyar a las personas ciegas como lectores de pantalla, lupas electrónicas y dispositivos portátiles de lectura, su elevado costo los hace inalcanzables para la mayoría. Ejemplos de estos casos son: las gafas inteligentes con reconocimiento de textos, dispositivos avanzados como OrCam MyEye o Envision Glasses, asociados con precios muy elevados, además de que la mayoría de estas soluciones provienen de mercado extranjeros, por lo que no siempre están adaptados a contextos locales lo que puede dificultar el idioma y el soporte técnico.

Por lo tanto, el problema central de esta investigación es el acceso limitado a la lectura dentro de la población con discapacidad visual. Esta situación es el resultado de una combinación de factoras como: la baja tasa de alfabetización en Braille, la falta en la accesibilidad de materiales impresos y dispositivos de audiolectura.

Este panorama resalta la urgencia de desarrollar soluciones innovadoras que utilicen tecnologías de visión artificial y síntesis de voz para superar estas barreras. Es por esto que la propuesta se centra en otorgar autonomía a los usuarios, por medio del acceso por sí mismos de textos impresos de uso cotidiano, con la finalidad de cerrar la brecha informativa, fomentando su inclusión dentro del ámbito educativo, laboral y social.

ANTECEDENTES

Internacionales

En el ámbito comercial, existen gafas inteligentes desarrolladas por empresas especializadas en este tipo de tecnologías. Alguno, ejemplos de estos son los dispositivos OrCam MyEyes y Envision Glasses que integran cámaras miniaturizadas con algoritmos de visión por computador e inteligencia artificial, ofreciendo lectura de textos en voz alta, el reconocimiento de rostros y la descripción de entornos (Calle et al., 2021). En este sentido, esos productos han validado la utilidad de la visión artificial como una herramienta de asistencia cotidiana para personas ciegas.

A continuación, se presentan detalles de los dispositivos mencionados:

- OrCam MyEye 2.0 se acopla magnéticamente a cualquier montura de lentes, siendo capaz de leer textos impresos y digitales en tiempo real, incluso permite identificar códigos de barras y rostros de personas registradas previamente. A su vez, operan de manera autónoma sin conexión a internet, ofreciendo ventajas en cuanto a privacidad y velocidad de respuesta (Balbi, 2017)
- Las gafas Envision utilizan una cámara con altavoz integrado para proporcionar descripciones de escenas, detección de objetos y lectura de textos en varios idiomas (Naayini et al., 2025).

Asimismo, Balbi (2017) menciona que el principal obstáculo de este tipo de iniciativas es su costo elevado, lo que limita su posicionamiento en mercado ya que solamente son accesibles por usuarios con altos ingresos o a través de programas de financiamiento institucional.

Dentro del área académica y de investigación ha surgido numerosos proyectos asociados al desarrollo de sistemas alternativos y de bajo costo. Uno ejemplo de estas iniciativas es el desarrollado por (Agastin et al. 2025) quienes desarrollaron unas gafas inteligentes utilizando una Raspberry Pi Zero 2W para la detección de objetos utilizando el algoritmo YOLO, el reconocimiento facial mediante Haar Cascade/LBPH y la lectura de texto a través de OCR con salida de audio por auriculares integrada. Este prototipo afirma que la creación de un asistente portátil multifuncional a través de componentes de bajo costo es

viable, sin embargo, se detectaron desafíos como la duración de la batería y la necesidad de optimizar los algoritmos para su ejecución dentro de hardware de baja capacidad.

Otra iniciativa es la desarrollada por Hoang et al. (2026), los cuales presentaron unas gafas asistenciales integrales que combinan la detección de objetos, el reconocimiento de texto y la posibilidad de traducción en tiempo real. Estas acciones estaban controladas por un microprocesador y activadas por un botón físico, a su vez, el sistema utiliza una cámara de Raspberry Pi y modelos de aprendizaje profundo para la interpretación del entorno y lectura de documentos, comunicando la información al usuario a través de audio.

Nacionales

En el contexto ecuatoriano, existen evidencias de iniciativas de prototipos que buscan asistir a personas con discapacidad visual en la lectura. Una de ellas es el trabajo de Basantes & Chalaco (2019), quienes diseñaron un sistema de gafas electrónicas para personas ciegas, integrando una microcomputadora Raspberry Pi 3B+, una cámara estándar Pi, audífonos alámbricos y la biblioteca OpenCV 4.0.

En este sentido, aunque este dispositivo logró demostrar la viabilidad técnica de convertir texto impreso a voz en un formato portátil, también presentó limitaciones significativas. Esto, debido a que todo el procesamiento realizado en la Raspberry Pi exigía un preprocesamiento intensivo para optimizar las imágenes antes del reconocimiento óptico de caracteres, lo que resultada demandante para la CPU y un considerable consumo energía. A su vez, esto reducía la autonomía de la batería y del rendimiento general. Adicionalmente, el uso de audífonos convencionales que permiten el aislamiento del usuario de los sonidos ambientales suponía un riesgo para su seguridad durante el desplazamiento.

Otra iniciativa es la desarrollada por Chinchero & Parra (2019). Este prototipo también se basó en una Raspberry Pi como unidad de procesamiento central, complementada con una cámara digital de alta definición conectada por USB y audífonos inalámbricos Bluetooth. Sin embargo, aunque lograba la captura del texto para reproducirlo como audio, su diseño carecía de ergonomía, ya que no estaba fijo a un soporte cómodo o gafas,

además de que los audífonos utilizados también bloqueaban la percepción auditiva del entorno.

En síntesis, aunque ambas iniciativas fueron esfuerzos pioneros en el Ecuador y lograron su objetivo principal, no supieron superar desafíos claves como la portabilidad y la aplicabilidad en contextos reales. Esto, debido a que los prototipos resultaron se poco prácticos, evidenciándose la necesidad de introducir mejoras sustanciales que logren generar una solución con verdadera funcionalidad.

Limitaciones observadas

Con base a los antecedentes descritos pueden decirse que, si bien existen diferentes iniciativas orientadas a la resolución del acceso a la lectura por parte de las personas con discapacidad visual, aún existen desafíos importantes.

Las soluciones comerciales son altamente funcionales, pero carecen de accesibilidad por ser de alto costo. Asimismo, los prototipos de bajo costo desarrollados en entornos académicos demuestran su factibilidad técnica, aunque presentan deficiencias en términos de eficiencia, portabilidad o experiencia de usuario.

Por lo tanto, la presente investigación busca recoger todos estos aspectos para diseñar un asistente visual innovador que logre la efectividad de la inteligencia artificial en la nube junto a la accesibilidad y adaptación al contexto local, proponiendo un prototipo portátil, energético y viable económicamente que responda a las necesidades del usuario con discapacidad visual.

JUSTIFICACION

Dentro del contexto social, el proyecto propuesto busca fomentar la inclusión de la población con discapacidad visual. Esto, debido a que el acceso a la información garantiza la mejora de oportunidades educativas, laborales y culturales. Por lo tanto, al desarrollar un prototipo de asistencia visual que permita a las personas con discapacidad visual “audio-leer” textos impresos con autonomía, se contribuiría a mejorar la accesibilidad a la educación, al conocimiento y a la eliminación de barreras para las personas con discapacidad.

La Unión Mundial de Ciegos (2017) enfatiza que la accesibilidad a la lectura es esencial para la alfabetización, la libertad de expresión y la inclusión social de las personas con discapacidad visual. En este sentido, en el Ecuador la población ciega depende de terceros o de materiales especiales escasos para acceder a información impresa.

Asimismo, dentro de la Institución Unidad Especializada "Mariana de Jesús" ubicada en Quito, Ecuador y dedicada al trabajo con niños ciegos y sordos se logró identificar que los alumnos ciegos no cuentan con suficientes libros ni ayudas tecnológicas modernas. El prototipo propuesto, beneficiaría a esta comunidad educativa, facilitando su aprendizaje e interacción con material educativo convencional.

Desde un enfoque técnico, el proyecto propuesto es pertinente con la problemática actual ya que aprovecha y fusiona avances relacionados con inteligencia artificial. Actualmente se dispone de modelos de reconocimiento de imágenes y de voz de alta precisión y de servicios en la nube que posibilita ejecutar estos modelos con hardware de bajo costo. Por lo tanto, la propuesta incluye la integración de un Modelo de Lenguaje Visual (VLM) para OCR y un servicio de síntesis de voz (TTS) avanzado, garantizando una calidad superior de reconocimiento frente a enfoques tradicionales.

El uso de la Raspberry Pi como microcomputador, permitirá flexibilizar procesos de mejora futuras. Asimismo, la propuesta involucra el diseño con software CAD para la fijación de la cámara en gafas, añadiéndole valor a la propuesta al crear una solución ergonómica que el usuario pueda usar dentro de su rutina. Además, el éxito de la propuesta demostraría las nuevas posibilidades al integrar tecnologías IoT y de IA aplicada (OT+IT) en el campo de la tiflotecnología. Afirmación respaldada por Hoang et al. (2026) quienes señalan que la integración de sistemas de voz y texto, representan un avance crucial dentro de la independencia de las personas con discapacidad visual.

En al ámbito económico, el objetivo del prototipo es sentar las bases para futuras innovaciones a nivel comercial. Al utilizar un hardware de bajo costo, componentes comerciales estándar y en su mayor parte software de código abierto, se favorece la replicabilidad y escalabilidad de la propuesta.

Asimismo, una vez refinado el dispositivo, este podría ser producido a nivel local y nacional, facilitando su adopción en el contexto nacional en diferentes niveles. En este

sentido la propuesta tiene un importante retorno social dentro de la tecnología asistiva, ya que se busca dotar a una persona con discapacidad visual de herramientas que le permitan se más independiente, además de que se reducen los gastos de adquisición de una asistencia individual, mejorando su calidad de vida. Esto se respalda por lo mencionado por la Unión Latinoamericana de Ciegos (2023) en donde mencionan que, dentro de este contexto, la tecnología es un recurso indispensable para la población ciega, ya que les proporciona autonomía y mayor amplio de rango en el desempeño de actividades.

OBJETIVOS DEL PROYECTO

Objetivo general

Desarrollar un prototipo de asistencia visual para audiolectura dirigido a personas con discapacidad visual, mediante hardware y softwares especializados.

Objetivos específicos

- Indagar literatura especializada referente a modelos de lenguaje visual (VLM) para su implementación en un asistente virtual, mediante bases de datos de artículos académicos indexados.
- Diseñar un módulo de asistencia visual para su acondicionamiento hacia al usuario mediante software CAD.
- Desarrollar un asistente virtual comandado por voz para la interacción humano-máquina mediante tecnología OT e IT.
- Implementar una arquitectura de comunicación entre el asistente visual y la nube para la transmisión de información, mediante IoT.
- Verificar el funcionamiento del asistente visual para su validación mediante pruebas de campo en espacios controlados.

ALCANCE

El alcance de este proyecto abarca desde la fase de conceptualización hasta la validación del prototipo de asistencia visual en un entorno controlado.

A continuación, se describirá cada alcance identificado:

- Diseño e implementación de un dispositivo compuesto por unas gafas con cámara integrada y una unidad de procesamiento (Raspberry Pi 4), junto con auriculares de conducción ósea con micrófono, formando un sistema asistivo completo. Se desarrollará un modelo CAD para el soporte de la cámara en las gafas y se fabricará (mediante impresión 3D u otro medio) un armazón o carcasa para montar los componentes de forma compacta y cómoda.
- Desarrollo del software del asistente en lenguaje Python, incorporando los siguientes módulos:
- Reconocimiento de voz local para activar comandos permitiendo al usuario operar el dispositivo con órdenes verbales en español.
- Captura de imágenes a través de la cámara, asegurando la obtención de fotografías nítidas del texto objetivo.
- Procesamiento en la nube con integración de la API de OpenAI para enviar la imagen capturada y recibir el texto reconocido y posteriormente convertir ese texto en habla.
- Reproducción de audio a través de un módulo que reciba el archivo de audio resultante y lo reproduzca en tiempo real por los auriculares de conducción ósea.
- Manejo de excepciones y retroalimentación al usuario mediante mensajes de voz.
- Arquitectura IoT y de comunicaciones donde se establecerá la conexión de la Raspberry Pi a internet y se programarán las solicitudes a las APIs remotas con autenticación segura.
- Pruebas funcionales en el laboratorio de la Universidad y en campo donde se realizarán pruebas iniciales con texto impreso estándar en ambientes controlados para medir la precisión del OCR y la claridad del TTS. Luego, se llevarán a cabo pruebas con usuarios con discapacidad visual de la Institución Mariana de Jesús en escenarios simulados, monitoreando el desempeño del prototipo

DELIMITAION

A continuación, se presentan los limites contextuales, temporales y técnicos delimitados para el desarrollo de la investigación:

Ámbito geográfico y poblacional: El prototipo está pensado para beneficiar inicialmente a los estudiantes con discapacidad visual de la Institución Unidad Especializada Mariana

de Jesús (ubicada en Quito, Ecuador), por lo que las pruebas y ajustes se realizarán bajo el contexto de los de esta institución.

Recursos tecnológicos: Para el reconocimiento de voz se utilizará el modelo Vosk con un vocabulario en español predefinido en un archivo JSON.

Para OCR y TTS, se implementarán los modelos provistos por la API de OpenAI. Se asume que habrá conectividad a internet estable durante las pruebas; el proyecto no abarca el desarrollo de modelos OCR/TTS locales debido a limitaciones de procesamiento en la Raspberry Pi.

Delimitaciones técnicas: El prototipo estará optimizado para reconocer y leer texto impreso en idioma español en tipografías comunes, restringiendo en tamaño de letra a 8, 10 y para la realización de las pruebas.

Asimismo, la cámara a utilizar (ArduCam 64MP) proporciona una alta resolución, pero el sistema puede requerir que el usuario encuadre correctamente el texto. El audio de salida se generará en español neutro.

Consideraciones éticas y de seguridad: El dispositivo será utilizado bajo supervisión durante las pruebas, en entornos seguros. En términos de privacidad, las imágenes capturadas se transmitirán a la nube de forma segura y únicamente con fines de OCR inmediato, sin almacenarse en bases de datos ni incluir el reconocimiento facial de personas

Alcance de la validación: Enfocada en comprobar que el dispositivo lee correctamente en voz alta documentos impresos sencillos, por medio de la identificación de la tasa de palabras correctamente reconocidas y entendidas por el usuario.

MARCO TEORICO

1.1 La discapacidad visual

La OMS (2023) define a la discapacidad visual como una condición que afecta la percepción del entorno de una persona a través de la vista, pudiendo ser total o parcial de acuerdo con el grado de afectación, con posibilidad de corregirse con ayudas óptimas como gafas o lentes de contacto.

En este sentido, Calle et al. (2021) mencionan que la discapacidad visual no solo hace referencia a la falta de una claridad del área visual, sino que incorpora factores funcionales, psicológicos y contextuales con la capacidad de impactar en la calidad de vida y social de la persona. En este sentido, la discapacidad visual puede ser de tipo hereditario, adquirido, traumático o clínico, cuyo impacto varía en función del grado de la pérdida visual, la edad en la que apareció la deficiencia y los medios de ayuda disponibles

Actualmente la interpretación que se tiene no se limita a una perspectiva médica, sino que incorpora factores prácticos, sociales y ambientales que determinan la independencia de la persona afectada (Saiz de la Hoya, 2024).

1.1.1 Clasificación de la discapacidad visual

La OMS (2023) menciona que la deficiencia visual puede manifestarse de diferentes maneras, tomando en cuenta la gravedad, origen e impacto funcional. Por lo tanto, al momento de realizar una clasificación de la condición, el enfoque no debe limitarse únicamente a la parte clínica, sino que debe incluirse la parte funcional, teniendo en consideración el nivel de afectación de la autonomía de la persona.

Asimismo, la institución ofrece una clasificación internacional para la discapacidad visual basada en la agudeza visual, tomando en cuenta la relación con lo que una persona con visión normal debería poder ver a cierta distancia. Esta clasificación incluye los siguientes niveles:

- **Visual leve**: agudeza visual inferior a 6/12 pero igual o superior a 6/18.
- **Visual moderada**: agudeza visual inferior a 6/18 pero igual o superior a 6/60.

- **Visual grave**: agudeza visual inferior a 6/60 pero igual o superior a 3/60.
- **Ceguera**: agudeza visual inferior a 3/60, lo que implica que la persona solo percibe luz o, en casos más severos, ni siquiera eso.

En este contexto, Arranz (2023) menciona que, en caso de evaluar la deficiencia de la visión de cerca de una persona, se establece que, si a una distancia de 40 centímetros y con algún tipo de corrección es incapaz de ver claramente se considera que tiene una agudeza visual inferior a N6 o N8. Asimismo, para comprender como clasificar el grado de visión de una persona, puede decirse que para que una persona con discapacidad leve pueda ver un objeto a 3.65 metros, necesitaría estar a una distancia de 1.8 metros para distinguirlo.

1.1.2. Tipos de discapacidad visual

Calle et al. (2021) señalan que, los tipos de discapacidad visual más frecuentes en niños son:

- **Ceguera total:** Donde existe una ausencia absoluta de la percepción visual, indicando que el niño no distingue luz ni formas, por lo que requiere el uso de herramientas táctiles y auditiva para lograr la interacción con el entorno.
- **Ceguera legal:** Asociada con una agudeza visual o menor a 20/200 en el mejor ojo con corrección, o como campo visual restringido a menos de 20 grados. En algunos casos existe una mínima percepción de luz, aunque esta suele ser insuficiente para desempeñar rutinas de forma autónoma. Por lo tanto, no se considera una ausencia total de la visión, si no de una pérdida funcional significativa de la forma en que la persona se relaciona con su entorno.
- **Baja visión moderada o severa:** El individuo tiene capacidad visual residual que le permite distinguir formas, colores, movimiento o luces.
- **Deficiencia visual cortical:** Es una condición de origen neurológico donde el cerebro es incapaz de procesar de forma correcta la información visual

Por lo tanto, cada una de estas condiciones implica el implemento de estrategias particulares de apoyo educativo, accesibilidad y uso de tecnologías asistivas. Por lo que entender cada una estas clasificaciones con sus diferencias, no es un ejercicio técnico, es

un paso necesario para el diseño de políticas, entornos y herramientas verdaderamente inclusivas.

1.1.3 Dificultades de aprendizaje asociadas a la discapacidad visual

El impacto de la discapacidad visual sobre los sistemas educativos afecta de manera significativa el acceso a la información y al aprendizaje. En este contexto, Basantes & Chamba (2019) señalan que los niños que presentan ceguera o baja visión muestran deficiencias y desafíos en el desarrollo de habilidades de lectoescritura, conceptos espaciales y matemáticos.

1.2 Tecnología asistiva y accesibilidad

1.2.1 Definición y principios

Moreno et al. (2023) señalan que la tecnología asistiva abarca a cualquier tipo de dispositivo, equipo, instrumento o programa informático, orientando a mantener o mejorar la autonomía funcional de sus usuarios.

Dispositivo con tecnología asistiva que describe la pantalla de la computadora (Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad, 2016)

En este contexto, la tecnología asistiva cumple con varios principios, los cuales se presentan a continuación:

- **Accesibilidad universal:** Busca eliminar barreras sensoriales, físicas y cognitivas (Benavides et al., 2023).
- **Diseño centrado en el usuario:** El papel del individuo con discapacidad no se limita solo al uso de la tecnología, sino que también es cocreador de la misma (Benavides et al., 2023).
- **Autonomía:** Debe ayudar al individuo a vivir sin depender de otros, facilitando la cotidianidad (Cotán et al., 2024).
- **Adaptabilidad y flexibilidad:** La tecnología debe ajustarse a contextos y necesidades personales (Cotán et al., 2024).
- **Asequibilidad y sostenibilidad:** Debe tener un acceso real y mantenimiento a largo plazo (Cotán et al., 2024).

1.2.2 Aplicación en asistencia visual para personas con discapacidad

Cardona & Vasquez (2019) señalan que la tecnología asistiva (TA) es el producto de un conjunto de herramientas agrupadas. A continuación, se describirá a cada una de ellas:

- **Detección y movilidad:** Bastones blancos inteligentes que identifican obstáculos y guían al usuario como un perro guía cibernético. Asimismo, prototipos como Shape que tantean con la navegación táctil avanzada (Cardona & Vasquez, 2021).
- **Navegación:** Sistemas GPS, sensores LiDAR dibujan rutas seguras, además de voces robóticas para la lectura de carteles (Cardona & Vasquez, 2021).
- **Acceso a la información:** Lectores como NVDA conviertes el texto digital en voz o Braille, abriendo a la educación y al empleo (Basantes et al., 2018).
- **Sistemas multimodales:** Aplicaciones que combinan visión artificial OCR y TTS, los cuales permiten describir escenas y leer documentos en tiempo real (Basantes et al., 2018).

Sin embargo, este tipo de tecnologías no se encuentran al alcance de todos. En este sentido, la disponibilidad tecnológica no garantiza la inclusión, ya que deben superar barreras como: economía, falta de capacitación y desigualdades. Por lo tanto, la creación de tecnologías debe nutrirse de la articulación de políticas públicas reales, accesibles y sostenidas (Cotán et al., 2024; Benavides et al., 2023).

1.3 Modelos de lenguaje visual (VLM)

1.3.1. Arquitectura y fundamentos

Alayrac et al. (2022) señalan que los modelos de lenguaje visual (VLM) son sistemas diseñados para articular una frase en el momento en que se observa una imagen producto de un proceso de aprendizaje profundo. Asimismo, los autores mencionan que dentro de este tipo de modelos se encuentran dos elementos esenciales:

- **El ojo computacional:** Actúa como codificador visual dentro de una red neural convolucional (CNN) o un Vision Transformer, ya que traduce una imagen en vectores numéricos.

- **La voz:** Decodificador lingüístico, donde por medio de un modelo de lenguaje autorregresivo, recibe vectores visuales y los transforma en frases.

1.3.2. Avances recientes y aplicaciones

Alayrac et al. (2022) mencionan que modelos como CLIP y Flamingo han procesado millones de pares imagen-texto hasta alcanzar una capacidad de adaptación muy alta. Sin embargo, se ha visto limitada en el mundo hispanohablante, ya que el idioma no siempre coincide con la interfaz, por lo que los VLM comienzan a destacar dentro de la accesibilidad, cultura y pedagogía.

Asimismo, los autores señalan que, en ciertos proyectos iberoamericanos, los VLM generan descripciones en castellano que rompen barreras para personas con discapacidad visual, superando ampliamente las capacidades de tecnologías más tradicionales como el OCR.

1.4. Interfaz de Programación de Aplicaciones (API)

Martínez (2021) la define como un conjunto de reglas y protocolos estructurales. En este sentido, la API REST actúa como la directora de orquesta de los microservicios, exponiendo recursos a través de protocolos HTTP y hablando el lenguaje universal del JSON.

Asimismo, los autores señalan que esta interfaz coordina un proceso de tres pasos:

- Envío de la imagen capturada a un modelo de lenguaje visual en la nube.
- Recepción de la descripción textual correspondiente y almacenamiento en un microcomputador.
- Entrega del texto a un sistema TTS que lo transforma en audio por unos audífonos.

1.5. Speach to text

Acosta (2022) lo define como una tecnología que cumple la función del oído atento, cuyo trabajo no solo se limita a oír, sino que interpreta, filtra y procesa. En este sentido, el proceso sigue cinco fases principales:

- Captura de audio.
- **Preprocesamiento:** filtrado de ruido, normalización y segmentación.
- Cálculo de probabilidades y encaje de fonemas por medio de un modelo neuronal.
- Decodificación en un texto legible o comandos funcionales.

Asimismo, el autor menciona que este tipo de procesamiento pueden realizarse en hardwares limitados como Rapsberry Pi, donde el procesamiento utiliza bibliotecas como Vosk que pueden ejecutar modelos de apenas 50 MB y tasas de error aceptables en español conversacional.

1.6. Síntesis de voz (TTS)

Tan y Qin (2022) describen la síntesis texto-a-voz (TTS) como un proceso organizado en tres módulos principales: análisis de texto, modelo acústico y vocoder. En el análisis de texto, la entrada escrita se transforma en características lingüísticas (por ejemplo, normalización y segmentación); luego, el modelo acústico convierte esas características en características acústicas; y, finalmente, el vocoder sintetiza la señal de voz a partir de dichas características.

Desde el enfoque tecnológico, Tan y Qin (2022) resumen la evolución del TTS en tres líneas de trabajo ampliamente usadas en la literatura y en aplicaciones prácticas:

- Síntesis concatenativa que genera voz mediante la selección y unión de unidades grabadas, buscando alta inteligibilidad a costa de depender de bases de datos grandes.
- Síntesis paramétrica (estadística) que modela la voz mediante parámetros y técnicas como HMM para predecir características acústicas, con mayor flexibilidad, pero con riesgo de una calidad percibida más “robótica”.
- Síntesis neuronal que emplea redes neuronales para mejorar la naturalidad y reducir parte del preprocesamiento manual, manteniendo con frecuencia la estructura modular (modelo acústico + vocoder) o avanzando hacia esquemas más integrados.

1.7. Diseño 3D en software CAD

Morales et al. (2025) mencionas que el diseño asistido por computadora (CAD) permite modelar objetos futuros, simular su comportamiento sin fabricarlos y anticipar errores antes de cometerlo, sin recurrir al resultado tangible

Asimismo, los autores mencionan que las soluciones CAD actuales como Autodesk Fusion, se ha consolidado como una especie taller donde el modelado 3D, la simulación estructural, la manufactura aditiva y el diseño electrónico conviven una sola interfaz. Además, ofrecen modelados paramétricos, libres o sculpt y directos, junto a simulaciones por elementos finitos y herramientas CAM que permiten generar rutas CNC directamente desde el mismo archivo.

1.8. Tecnología OT

Pancho & Galarza (2014), lo define como el conjunto de hardware y software que supervisa o controla procesos físicos en tiempo real. En el contexto del prototipo propuesto, Raspberry Pi y la cámara conforman un pequeño ecosistema IT de borde, ya que capturan la escena impresa, ajustan el encuadre e iluminación y activan la salida de audio por conducción ósea.

1.9. Tecnología IT

Goyes (2020) define a la tecnología IT como el uso de computadoras, redes y servicios en la nube para almacenar, procesar y transmitir datos. En el contexto del prototipo propuesto, la capa IT se encuentra en la nube, donde un modelo de lenguaje visual GPT-4o-mini-vision recibe la imagen enviada por Raspberry Pi, la interpreta y genera un texto. Ese texto, a su vez, viaja a un servicio neuronal de TTS que lo convierte en voz con una naturalidad casi inquietante.

1.10. Internet de las Cosas (IoT)

La Unión Internacional de Telecomunicaciones (2025) define al Internet de las cosas (IoT) como una infraestructura mundial que habilita servicios avanzados mediante la interconexión de objetos físicos y virtuales, apoyada en capacidades de identificación,

adquisición y procesamiento de datos, y comunicación, considerando además requisitos de seguridad y privacidad.

Para comprender su funcionamiento, la UIT (2025) propone un marco de capacidades organizado en capas (aplicación, soporte de servicios y aplicaciones, red y dispositivos) y capacidades transversales, destacando la seguridad como componente clave. En el contexto del prototipo, la Raspberry Pi se interpreta como un nodo IoT orientado a visión: captura información visual, realiza procesamiento local básico y/o transmite los datos hacia componentes de borde y nube para obtener una salida útil para el usuario.

Bajo esta perspectiva, el IoT del prototipo puede describirse mediante las siguientes dimensiones (alineadas con el marco por capas de la UIT):

- **Capa de dispositivo (percepción):** comprende la captura de imágenes, su digitalización y la preparación de datos desde el dispositivo de visión y/o pasarela (p. ej., Raspberry Pi).
- **Capa de borde (soporte de servicios y aplicaciones):** actúa como primer nivel de procesamiento cercano a la fuente (preprocesamiento, compresión, filtrado de redundancia y decisiones iniciales) para optimizar recursos de red y cómputo.
- **Capa de transporte (red):** se encarga del envío de información entre el dispositivo/borde y los servicios remotos, priorizando comunicaciones confiables y protegidas según el diseño del sistema.
- **Capa de plataforma o cloud (soporte de servicios y aplicaciones):** recibe los datos, ejecuta análisis (por ejemplo, modelos multimodales para generar descripciones) y entrega resultados a los servicios de salida (como TTS).
- **Capa de aplicación:** integra el servicio final para el usuario, cerrando el ciclo de "captura–procesamiento–entrega", por ejemplo, reproduciendo el audio generado mediante audífonos de conducción ósea.
- **Seguridad (capacidad transversal):** contempla mecanismos de protección de datos y del sistema (cifrado, verificación de información sensible, protección de acceso y medidas de seguridad a nivel de red y plataforma), lo cual es consistente con un enfoque IoT moderno.

CAPITULO II

MARCO METODOLÓGICO

2.1 Enfoque metodológico

El enfoque para el desarrollo de este proyecto es bajo la metodología de investigación aplicada pues lo que busca es no solo el estudio sino también el diseño y validación de un prototipo funcional que resuelva un problema real el cual es la dificultad que las personas con discapacidad visual tienen para acceder a información escrita.

A partir de este punto de vista metodológico y el desarrollo tipo tecnológico experimental que se debe a que la construcción de un sistema que se integra tanto hardware como software que fue sometido a pruebas en un entorno controlado para medir su rendimiento y viabilidad.

Esta metodología utilizada permite combinar los conocimientos de electrónica, la programación y la inteligencia artificial para el desarrollo de un sistema de tecnología asistiva y además garantizando que el sistema cumpla con los objetivos propuestos y satisfaga las necesidades de los usuarios finales.

2.1.2 Diseño del sistema

Para el diseño del sistema definimos una estructura lógica y física así como la comunicación de los módulos de captura, procesamiento y salida de datos además diseñamos la estructura física que son la gafa inteligente haciendo uso de un software CAD, Fusion 360 fue el software utilizado para este caso, en dichas gafas irán integradas el módulo de cámara, y los audífonos de conducción ósea además de un micrófono externo de mesa que mejora la detección de los comandos del dispositivo y todo esto asegurando la ergonomía y funcionalidad del dispositivo.

2.1.3 Implementación

En esta fase utilizamos una Raspberry Pi 4, una cámara Arducam de 64 MP, audífonos de conducción ósea y un micrófono externo de mesa. Para la programación se hizo uso del software en Python integrando APIs para el reconocimiento de comandos por voz, la

conversión de imágenes a texto y finalmente a audio probando la funcionalidad de las piezas antes unirlas en el prototipo final.

2.1.4 Pruebas y validación

Para concluir con el desarrollo del prototipo se realizaron pruebas en espacios controlados con el fin de evaluar la eficacia al reconocer los comandos por voz, la nitidez en la captura de imágenes, la precisión de la conversión de texto a audio y la reproducción a través de los audífonos de conducción ósea. Una vez obtenidos los resultados, nos permitió hacer los ajustes finales para obtener un desempeño eficiente del prototipo de asistencia visual.

2.2.1 Diagrama del sistema

Para este prototipo la arquitectura se distribuye de manera lógica combinado tanto hardware como software para asistir a las personas con discapacidad visual, el diseño esta divido por módulos, es decir piezas independientes lo que hace más fácil detectar posibles fallos y ajustarlos de manera más fácil, además que permite agregar más funciona en un futuro sin modificar todo el dispositivo.

Su funcionamiento está basado en una estructura que se divide en 3 etapas que son adquisición, procesamiento y salida, haciendo referencia a los procesos de la captura de imagen, la conversión del texto con inteligencia artificial y convertirlo en voz para el usuario.

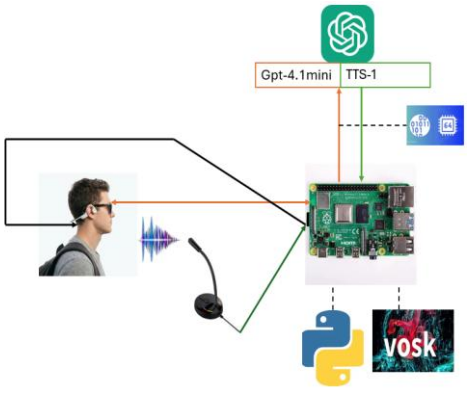


Figura. 2.1 Diagrama del sistema
Diagrama de la arquitectura del prototipo de asistencia para ciegos. Elaborado por Henry Criollo y Anthony Urresta

En el diagrama de la figura 2.1 podemos observar el cómo el proceso interactúa con el usuario al inicio y al final de la siguiente manera:

El proceso inicia cuando el usuario manda un orden “Leer documento” a través del micrófono de mesa y esta señal llega a la Raspberry Pi 4 misma que actúa con cerebro de todo el sistema.

Con la ayuda de la Vosk permite que el sistema entienda los comandos que viene del micrófono y convierte esa instrucción a texto y así activar la siguiente etapa del sistema.

Una vez enviada la orden se activa la cámara Arducam que toma una fotografía en alta resolución del texto y la envía para su análisis.

Para poder sacar el texto con la más alta precisión posible se envía la imagen a un servidor externo mediante una API en este caso OPENAI la cual hace un análisis de reconocimiento (OCR) para transformar la imagen a texto y con un motor de síntesis de voz lo convierte a audio.

Por último, el audio regresa al dispositivo y se reproduce en los audífonos de conducción ósea cuya eficiencia radical en poder transmitir la lectura del texto al usuario sin tapar el oído y así no interferir con los sonidos del entorno.

2.3.1 Desarrollo del asistente virtual comandado por voz

Para hacer posible la interacción comando-acción-respuesta por audio. El asistente escucha continuamente al usuario, hasta que esté de un comando verbal valido, luego ejecuta la captura de imagen y posteriormente devuelve una respuesta audible. El proceso i2mplementa tecnología OT (Operational Technology) al realizar la operación de los dispositivos físicos como la cámara el micrófono y la reproducción de audio, con tecnología IT (Information Technology) al hacer uso de procesamiento lógico, manejo de datos y servicios en la nube.

2.3.2 Definición de comandos y semántica:

Para definir los comandos de accionamiento del asistente se elaboró el archivo `commands.json`, como se puede observar en la figura 2.2, donde a las funciones `read_document` y `exit` que nos permiten capturar imagen y salir del asistente, se las asocia a un alias lingüístico para su ejecución. Además, al hacer uso del archivo JSON

se puede ampliar la frase de accionamiento agregando nuevos alias sin modificar la lógica principal.

```
"read_document": [
    "leer documento",
    "quiero que leas",
    "puedes leer esto",
    "lee esto",
    "lee el documento",
    "leer texto"
],
"exit": [
    "salir",
    "terminar",
    "adiós",
    "bye",
    "cerrar"
```

Figura. 2.2 Comandos de ejecución JSON
Comandos que el prototipo recibe a través del micrófono. Elaborado por Henry Criollo y Anthony Urresta

En el programa los comandos son cargados por la función `load_commands_from_file,` como se puede observar en la figura 2.3, ubicada en nuestro `main`, esta función valida la estructura del archivo JSON, si el archivo no existe o está mal formado el sistema utilizará comandos por defecto.

```
def load_commands_from_file(file_path="commands.json"):
    """Carga comandos desde archivo JSON o retorna comandos por defecto"""
    try:
        if os.path.exists(file_path):
            with open(file_path, "r", encoding="utf-8") as f:
                commands = json.load(f)
                if not isinstance(commands, dict):
                    raise ValueError("El archivo de comandos debe contener un diccionario")
```

Figura. 2.3 Validación de comandos JSON
Elaborado por Henry Criollo y Anthony Urresta

2.3.3 Captura y acondicionamiento de audio:

En el módulo `audio/recognizer.py` se implementó la librería `VoskRecognizer`. Como elemento principal del reconocimiento de voz, esta clase nos permite vincular el flujo de entrada de audio mediante `sounddevice.InputStream` el cual entregara el audio al sistema mediante un callback como se ve en la Figura 2.4, asociado al stream de escucha, para desacoplar la captura del audio de su procesamiento se almacena en una estructura `queue.Queue` la cual permite un intercambio seguro de datos entre hilos. Finalmente, el audio es procesado por `KaldiRecognizer` del toolkit de Vosk para convertir la señal capturada a texto de validación de comando. Además, se hace uso de

`threading.Thread` para mantener la escucha y reconocimiento de comandos en un hilo independiente, dejando el hilo principal para la gestión de acciones y estados.

```
def _callback(self, indata, frames, time, status):
    if status:
        print("Status:", status)
    # Solo procesar audio si no está pausado
    if not self.paused:
        self.q.put(bytes(indata))
```

Figura. 2.4 Callback de audio
Elaborado por Henry Criollo y Anthony Urresta

Una vez realizado el callback la función `star_listening` como se observa en la figura 2.5, iniciara el stream de audio lo cual permite la escucha de comandos por el sistema.

```
def start_listening(self):
    """Inicia el stream de audio"""
    if not self.initialized:
        raise RuntimeError("Vosk no está inicializado. Llama a initialize() primero.")

    try:
        self.stream = sd.RawInputStream(
            samplerate=16000,
            blocksize=8000,
            dtype='int16',
            channels=1,
            callback=self._callback
```

Figura. 2.5. Comandos de ejecución
Elaborado por Henry Criollo y Anthony Urresta

2.3.4 Reconocimiento y emparejamiento de comandos (IT):

Una vez que el sistema obtenga texto, La cadena reconocida se compara con los alias del diccionario mediante la función `get_best_command_match,` como se observa en la Figura 2.6, ubicada en `main.py`. Para lograr la validación del comando se aplica un emparejamiento difuso que acepta variaciones del habla al momento de pronunciar los comandos para esto, se define un umbral `cutoff=0.6` para aceptar únicamente coincidencias suficientemente cercanas.

```python
def get_best_command_match(text, command_aliases, cutoff=0.6):
    """Encuentra la mejor coincidencia de comando usando búsqueda difusa"""
    if not text or not command_aliases:
        return None

    # Normalizar el texto de entrada
    text = text.lower().strip()

    all_phrases = {
        alias.lower(): action
        for action, aliases in command_aliases.items()
        for alias in aliases
    }

    match = difflib.get_close_matches(text, all_phrases.keys(), n=1, cutoff=cutoff)
    return all_phrases[match[0]] if match else None
```

Figura 2.6 Emparejamiento difuso de comandos
Elaborado por Henry Criollo y Anthony Urresta

2.3.5 Ejecución de acciones:

Una vez se haya determinado la acción (read_document, exit) la función `handle_command(action)`como se observa en la figura 2.7, ubicada en `main.py` actúa como despachador. Si se ejecuta `read_document` se invoca a `vision/camera.py`; si requiere servicios en la nube (OCR/TTS), se llama a los módulos correspondientes. Aquí la Raspberry Pi funciona como Gateway OT/IT, coordinando subsistemas físicos y lógicos.

```python
def handle_command(action):
    """Maneja la ejecución de comandos con bloqueo para evitar concurrencia"""
    filename = None
    try:
        with command_lock:
            # Pausar la escucha mientras se procesa el comando
            pause_listening()

            if action == "read_document":
                speak("Tomando foto del documento...")
                filename = take_picture()
```

Figura. 2.7 Dispatcher de acciones
Elaborado por Henry Criollo y Anthony Urresta

2.3.6 Captura de imagen

Cuando el usuario acciona el comando de audiolectura el programa pasa al módulo `vision/camera.py`, donde ejecuta la función `take_picture` que a su vez ejecuta la funcion `_take_picture_system_command` como se observa en la figura 2.8, que un comando del sistema (rpicam-jpeg) desde Python a través de `subprocess.run`, función la cual nos permite ejecutar comandos del sistema operativo en la terminal. De esta forma se logra controlar a la cámara y guardar la imagen en el dispositivo.

```python
def _take_picture_system_command(filename):
    """Usar comandos del sistema para tomar foto (más confiable en RPi)"""
    filename = Path(filename)
    filename.parent.mkdir(parents=True, exist_ok=True)

    commands_to_try = [
        # libcamera (Raspberry Pi OS Bullseye+)
        ["libcamera-still", "-o", str(filename), "-t", "1000", "--width", "1920", "--height", "1080"],
```

Figura. 2.8 Toma de fotografía
Elaborado por Henry Criollo y Anthony Urresta

2.3.7 Procesamiento cognitivo:

La función `ocr_image`, como se observa en la figura 2.9, ubicada del módulo `vision/ocr.py` transforma la imagen capturada por la cámara al formato Base64 que permite enviar archivos a través de internet dentro de una solicitud http, esta imagen codificada se envía a la API del modelo visual (VLM).

Una vez que la imagen llega al servicio en la nube, el modelo la analiza, siguiendo las instrucciones dadas por el prompt, devolviendo texto como respuesta. Debido al procesamiento realizado en la nube la raspberry Pi no necesita realizar procesamiento complejo de manera local

```python
def ocr_image(image_path):
    """Extrae texto de una imagen usando la API de OpenAI GPT-4 Vision"""
    if not OPENAI_API_KEY:
        return "Error: No se encontró la clave API de OpenAI."

    try:
        with open(image_path, "rb") as img:
            encoded = base64.b64encode(img.read()).decode("utf-8")

        url = "https://api.openai.com/v1/chat/completions"
        headers = {
            "Content-Type": "application/json",
            "Authorization": f"Bearer {OPENAI_API_KEY}"
        }
```

Figura. 2.9 Llamada al VLM para OCR
Elaborado por Henry Criollo y Anthony Urresta

2.3.8 Retroalimentación por voz.

Con el texto obtenido, el programa usa la función `speak` ,como se observa en la figura 2.10, ubicada en el módulo `audio/speaker.py`, que nos permite enviar la solicitud a la API de texto a voz (TTS). Además, nos permite configurar el modelo de voz a utilizar en este caso “tts-1” de OPEN AI y la configuración de voz del asistente.

Una vez que la API responde entrega un archivo de audio en formato wav, el sistema guarda este archivo de manera temporal y lo reproduce localmente haciendo uso del comando del sistema aplay desde Python haciendo uso de la función subprocess.run

```python
def speak(text):
    """Convierte texto a voz usando OpenAI TTS"""
    if not text or not text.strip():
        return

    text = text.strip()

    if not OPENAI_API_KEY:
        print(f"[Error] No hay clave API de OpenAI configurada. Texto: {text}")
        return

    _speak_with_openai(text)

def _speak_with_openai(text):
    """Usar OpenAI TTS para generar voz"""
    try:
        url = "https://api.openai.com/v1/audio/speech"
        headers = {
            "Authorization": f"Bearer {OPENAI_API_KEY}",
```

Figura. 2.10 Síntesis y reproducción de voz
Elaborado por Henry Criollo y Anthony Urresta

2.3.9 Gestión de estados y concurrencia

Mientras se ejecuta un comando el programa hace uso de las funciones `pause_listening` para pausar la escucha y `resume_listening` como se observa en la figura 2.11 para reanudarla, estas funciones nos permiten que cuando se detecte un comando el programa detenga temporalmente el reconocimiento de comandos, y una vez que el audio se reproduzca por completo, permite reanudar la escucha para recibir el siguiente comando del usuario.

```python
def pause_listening(self):
    """Pausa el procesamiento de audio sin detener el stream"""
    if self.listening:
        self.paused = True
        # Limpiar la cola de audio acumulado
        while not self.q.empty():
            try:
                self.q.get_nowait()
            except:
                break
        print("🔇 Escucha pausada durante procesamiento...")

def resume_listening(self):
    """Reanuda el procesamiento de audio"""
    if self.listening:
        self.paused = False
```

Figura. 2.11Control de escucha
Elaborado por Henry Criollo y Anthony Urresta

2.3.10 Limpieza y seguridad operativa

El sistema durante su funcionamiento genera archivos temporales de fotografía(jpeg) y audio(.wav) resultantes de la síntesis de voz, si estos archivos no se borran del sistema se acumularían pudiendo llenar el almacenamiento con cada uso, por esta razón se implementó la función `cleanup_temp_files`, como se observa en la figura 2.12 que

permitirá borrar los archivos creados en los ciclos de audio lectura. Adicionalmente, el programa hace uso de bloques `try/except` para la gestión de errores de manera controlada, logrando que en vez de que el programa se cierre abruptamente al reconocer errores de funcionamiento pueda registrar el error, continuar operando o finalizar de manera segura.

```python
def cleanup_temp_file(filename):
    """Limpia archivos temporales de manera segura"""
    if filename and os.path.exists(filename):
        try:
            os.remove(filename)
            print(f"Archivo temporal eliminado: {filename}")
        except OSError as e:
            print(f"No se pudo eliminar archivo temporal {filename}: {e}")
```

Figura. 2.12 Eliminación de temporales
Elaborado por Henry Criollo y Anthony Urresta

2.4 Arquitectura de comunicación IoT (Edge–Cloud)

El propósito de esta fase es establecer una comunicación segura y confiable entre el dispositivo Raspberry Pi (capa edge) y los servicios en la nube (capa cloud) encargados de dos tareas críticas del prototipo: extracción de texto desde imágenes (OCR mediante VLM) y síntesis de voz (TTS). En esta arquitectura, la Raspberry Pi actúa como un nodo IoT que captura datos del entorno (imagen) y consume servicios remotos para transformarlos en información útil (texto y audio), garantizando continuidad operativa incluso ante condiciones reales de red.

2.4.1 Etapas principales

2.4.1.1 Verificación de conectividad

Antes realizar cualquier instrucción de envió a la nube lo primero que debe realizar el sistema es asegurarse que exista conexión a internet de esta manera evita perder tiempo sin no existe conexión al igual de boques del flujo principal y reducir errores.

2.4.1.2 Envío de datos por HTTP/REST

Una vez tomada la fotografía por medio de la Arducam, esta se prepara para la transmisión de datos por medio de Base64 y de esta manera poder enviar el archivo como payload de una solicitud de HTTP en formato JSON para luego enviar el archivo a un

servicio de procesamiento visual (OCR) a través del enfoque REST aquí el Edge actúa como cliente y la nube como servidor

2.4.1.3 Texto que recibe del OCR y reenvío al servicio TTS

Aquí recibimos el texto leíble que se extrajo de la fotografía usando el OCR mismo que será usado como solicitud dirigida al servicio TTS, esto corresponde al flujo cloud divida en una etapa imagen a texto y una segunda etapa que va de texto a audio

2.4.1.4 Almacenamiento temporal y limpieza

El prototipo está diseñado para almacenar archivos solo el tiempo necesario para la conversión de imagen a texto y de texto a audio, una vez termina la iteración estos archivos se eliminan automáticamente, esto se hace para no saturar la memoria del dispositivo y reduce la probabilidad de que se acumule información sensible.

2.4.2 Como está Programado

2.4.2.1 Verificación de internet con sockets

Aquí validamos que exista conectividad con internet y para ellos se usa la función check internet () la cual primero intenta conectarse a nivel socket antes realizar alguna solicitud HTTP.

2.4.2. OCR en la nube usando VLM

Lo primero que se hace es tomar la fotografía a través de BASE64 convertirla a un formato de texto largo para que pueda subir a internet, una vez ahí y con una instrucción (promt) y en .json se envía una solicitud con request.post para enviarla a un servidor potente (VLM) y finalmente recibir el texto extraído de la imagen.

2.4.3 Conversión de voz en la nube

En este punto ya se tiene el texto extraído de la imagen asique se usa a función speak() que envia el texto por internet a un servidor TTS para convertirlo en audio. El archivo recibido será un audio en formato .wav a una carpeta temporal (TEMP_DIR) que

finalmente será reproducida por los audífonos de conducción ósea concluyendo el proceso.

2.4.4 Manejo de errores y continuidad de servicio

Para evitar errores en el sistema el código tiene bloques de control de errores try/except que intercepta anomalías como fallo de conexión y cuando esto ocurre en lugar de que el proceso se interrumpa el sistema avisa al usuario mediante audio que ocurrió algún imprevisto.

2.5 Diseño del CAD del prototipo de asistencia visual

Para el diseño de módulo de asistencia visual de desarrollo mediante el software CAD Fusion 360 el cual permite modelar, diseñar y optimizar todo el desarrollo físico del prototipo antes de su fabricación.

2.5.1 Primer diseño del módulo de asistencia visual

Para este primer diseño se inició con la idea de hacer una estructura robusta con el fin de proteger los módulos de la cámara y el conector HDMI, debido a esto el armazón de las gafas muestra un volumen considerable en sus dimensiones pues la cámara se ubicó en la parte superior de las lunas para garantizar la calidad de las fotos, también cabe mencionar que para la impresión se usó el filamento PLA (Poliácido Láctico) por ser un material más común y rápido para realizar impresiones 3D y con un relleno del 30% la impresión cumplía las necesidades que se buscaban de manera óptima y eficiente.

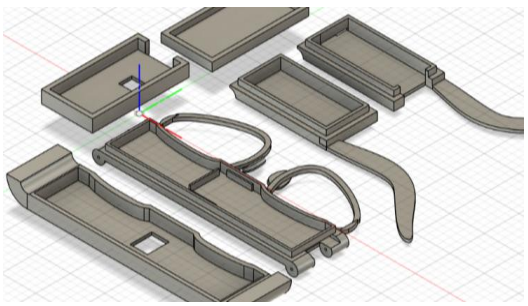


Figura. 2.13 Primer prototipo CAD
Gafas robustas con peso considerable funcional para pruebas de desarrollo. Elaborado por Henry Criollo y Anthony Urresta

Esta primera versión tenia a los módulos de la cámara y HDMI en sitios diferentes, la cámara arriba de las lunas y el módulo HDMI en el soporte de la oreja derecha, y para

proteger los módulos también se tuvo que diseñar unas tapas para cubrir los componentes electrónicos y evitar daños por golpes, sin embargo, en las pruebas realizadas se tomó la decisión de cambiar el diseño, esto debido al peso del prototipo y de o estar integrado a los auriculares de conducción ósea.

2.5.2 Segundo diseño del asistente visual

En esta segunda etapa se tuvo énfasis en un diseño más ligero y estético por lo cual se rediseño por completo el armazón reemplazando los soportes de las orejas por unas correas ajustables y se cambió el lugar de la cámara, estos cambios hiso recudir en gran medida el peso de las gafas para hacerlo más cómodo y personalizado. El filamento utilizado fue el mismo que en el primer diseño, esta vez con un relleno del 40% pues el armazón era mucho más delgado por lo que se subió el relleno para fortalecer su resistencia, para las correas se usó filamento TPU (Poliuretano Termoplástico), que gracias a su flexibilidad y robustez lo hizo ideal para imprimir las correas ya que permitían ajustar las correas a comodidad del usuario.

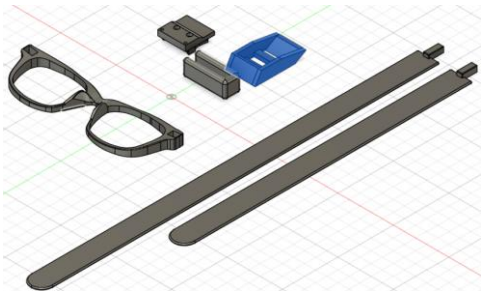


Figura. 2.14 Segundo prototipo CAD
Diseño más ligero con ajuste de acuerdo con el usuario con ligeros inconvenientes al colocarse. Elaborado por Henry Criollo y Anthony Urresta

Otro cambio importante que se hizo fue integra la cámara y el HDMI en un solo modulo para ir ubicado al frente de las gafas justo en la mitad de las gafas. Estos cambios ayudaron en gran medida la estabilidad de las gafas corrigiendo por completo la inclinación que ocurrían debido al peso y mejorando el campo de visión de la cámara al estar ubicada en el punto de vista natural de los ojos.

También se logró a integración de las gafas con los audífonos a través de unos soportes que unen los audífonos a las correas del dispositivo logrando unir todo el sistema en un solo modulo cómodo y estable.

2.5.3 Tercer diseño y final del asistente visual

Este es el diseño más reciente y definitivo basándose en las criterios y aspectos no tomados en cuenta del diseño anterior para hacer al diseño lo más funcional posible.

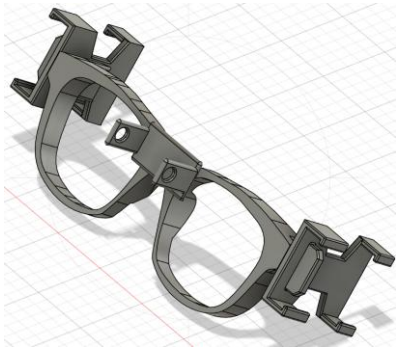


Figura. 2.15 Prototipo CAD final
Diseño definitivo de las gafas cómodo y ligero. Elaborado por Henry Criollo y Anthony Urresta

Lo primero que se implemento fue un mecanismo de soportes de liberación lateral que permite a una persona colocarse el dispositivo con mayor facilidad y rapidez pues ofrece una respuesta táctil y sonora al colocarse y/o retirarse las gafas.

Para la integración con los auriculares de conducción ósea se optó por envolver a los audífonos con una funda protectora manteniendo expuesto los botones y el pin de carga y así unir las gafas a los auriculares por medio de las hebillas, por último, la integración con el módulo de la cámara se agregaron unos soportes en el puente de las gafas donde estar ubicada el módulo de la cámara dando como resultado un prototipo compacto cumpliendo con la eficiencia y eficacia de una herramienta de asistencia moderna.

RESULTADOS

Ahora que el prototipo de asistencia visual se encuentra terminado se detallará cada una de sus etapas, las pruebas de funcionamiento y encuestas realiza en la Institución Unidad Especializada Marina de Jesús.

Figura. 2.16 Realización de pruebas
Momento en el que se realizaron las pruebas de funcionamiento en la Institución Unidad Especializada Mariana de Jesús. Elaborado por Henry Criollo y Anthony Urresta

2.6 Evaluación y reconocimiento de captación de comandos

Se evaluaron dos elementos de entrada de audio el micrófono integrado en los audífonos de conducción ósea y un micrófono de mesa USB, el comando a evaluar fue "documento" el cual activa el proceso de audiolectura, se tomó 10 muestras en ambos dispositivos la activación exitosa se marcó con "✓" y la errónea con "✗". En la tabla 3.1 se muestran los resultados.

Tabla 3.1 Comparación de captación de comando por micrófono de los audífonos y micrófono de mesa

Captación de comandos		
№ Prueba	audífonos	micrófono
1	✓	✓
2	✗	✓
3	✗	✓
4	✓	✓
5	✗	✓
6	✓	✓
7	✗	✓
8	✓	✗
9	✗	✓
10	✗	✓
11	✓	✓
12	✗	✓
13	✗	✓
14	✗	✓
15	✗	✓

Cuando el reconocimiento del comando el correcto se marca con un "✓" y cuando no reconoce se denota con "✗". Elaborado por Henry Criollo y Anthony Urresta

Porcentaje de éxito = (Número de éxitos / Total de muestras) × 100

Porcentaje de error = (Número de errores / Total de muestras) × 100

Tabla 3.2 Porcentaje de éxitos y errores de captación de comandos de voz

Micrófono	Muestras	Aciertos	Errores	Éxito
Audífonos	15	4	11	26,67 %
Mesa	20	19	1	95 %

Análisis de las muestras tomadas en base a si recibió correctamente el comando o no. Elaborado por Henry Criollo y Anthony Urresta

Una vez implementado el micrófono de mesa se evidencio una enorme mejoría en la captación de los comandos de voz con un 95% de exactitud frente a u 26,67 porciento del micrófono de los audífonos, esto nos indica que la captación y reconocimiento de los comandos de voz es eficiente para que el prototipo funcione correctamente.

Figura 2.17 Detección de comando

```
[0:02] Decoding of output.mp3 finished.
Detectado: documento
Comando detectado: documento
Escucha pausada durante procesamiento...
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3
version 1.31.2; written and copyright by Michael Hipp and others free software (LGPL)
without any warranty but with best wishes
```

Al momento de recibir el comando y lo detecta correctamente como se muestra en la figura 2.17 procede a iniciar con la extracción de texto. Elaborado por Henry Criollo y Anthony Urresta

2.7 Captura de imagen

Para la captura de imagen se optó por tomas muestras con tres diferentes tipos de resolución siendo estas 8 MP, 16 MP y 32 MP los resultados obtenidos de dichas pruebas se muestra en la Tabla 3.3

Tabla 3.3 Comparación de la nitidez a 8MP, 16 MP y 32MP

N- Muestras	Nitidez de la imagen a 8MP	Nitidez de la imagen a 16MP	Nitidez de la imagen a 32MP
1	✗	✗	✓
2	✗	✓	✓
3	✗	✗	✓
4	✗	✗	✓
5	✓	✗	✓
6	✗	✓	✗
7	✓	✓	✓
8	✗	✗	✓
9	✗	✓	✓
10	✓	✗	✗
11	✗	✗	✓
12	✓	✓	✓
13	✗	✓	✓
14	✗	✗	✓
15	✗	✓	✓

Elaborado por Henry Criollo y Anthony Urresta

Una vez terminadas las muestras se obtuvieron los porcentajes de éxito y error, los cuales dejaron en claro que la mejor resolución para el prototipo es la resolución de 32 MP la cual nos dio un 86.67% en la nitidez de las imágenes capturadas frente a un 53.33% de nitidez de a 16 MP y 26.67% de nitidez a 8 MP cuya resolución de imágenes no era óptimo para enviar al OCR.

Porcentaje de éxito = (Número de éxitos / Total de muestras) × 100

Porcentaje de error = (Número de errores / Total de muestras) × 100

Tabla 3.4 Porcentaje de éxitos y errores de la nitidez en la captura de imagen

Resolución	Aciertos	Errores	Éxito (%)	Error (%)
8 MP	4	11	26,67 %	73,33 %
16 MP	8	7	53,33 %	46,67 %
32 MP	13	2	86,67 %	13,33 %

Compara los porcentajes de éxitos y errores para selección la mejor opción. Elaborado por Henry Criollo y Anthony Urresta

Con la elección definitiva de la resolución de la cámara siendo esta 32 MP la nitidez de las imagines capturadas deja de ser un problema pues entra dentro del margen de errores

aceptable y no perjudica el resto del funcionamiento del sistema pues si una imagen no está bien enfocada simplemente se elimina y se procede a tomar otra para seguir con el proceso.

Figura 2.18 Texto de prueba

Uga la tortuga - Cuento infantil sobre la perseverancia

- ¡Caramba, todo me sale mal!, se lamenta constantemente Uga, la tortuga.

Y es que no es para menos: siempre llega tarde, es la última en acabar sus tareas, casi nunca consigue premios a la rapidez y, para colmo es una dormilona.

- ¡Esto tiene que cambiar!,- se propuso un buen día, harta de que sus compañeros del bosque le recriminaran por su poco esfuerzo al realizar sus tareas.

Y es que había optado por no intentar siquiera realizar actividades tan sencillas como amontonar hojitas secas caídas de los árboles en otoño, o quitar piedrecitas de camino hacia la charca donde chapoteaban los calurosos días de verano.

- ¿Para qué preocuparme en hacer un trabajo que luego acaban haciendo mis compañeros? Mejor es dedicarme a jugar y a descansar.

- No es una gran idea - dijo una hormiguita - Lo que verdaderamente cuenta no es hacer el trabajo en un tiempo récord; lo importante es acabarlo realizándolo lo mejor que sabes, pues siempre te quedará la recompensa de haberlo conseguido.

No todos los trabajos necesitan de obreros rápidos. Hay labores que requieren tiempo y esfuerzo. Si no lo intentas nunca sabrás lo que eres capaz de hacer, y siempre te quedarás con la duda de si lo hubieras logrado alguna vez.

Por ello, es mejor intentarlo y no conseguirlo que no probar y vivir con la duda. La constancia y la perseverancia son buenas aliadas para conseguir lo que nos proponemos; por ello yo te aconsejo que lo intentes. Hasta te puede sorprender de lo que eres capaz.

- ¡Caramba, hormiguita, me has tocado las fibras! Esto es lo que yo necesitaba: alguien que me ayudara a comprender el valor del esfuerzo; te prometo que lo intentaré.

Pasaron unos días y Uga la tortuga se esforzaba en sus quehaceres.

Se sentía feliz consigo misma pues cada día conseguía lo poquito que se proponía porque era consciente de que había hecho todo lo posible por lograrlo.

- He encontrado mi felicidad: lo que importa no es marcarse grandes e imposibles metas, sino acabar todas las pequeñas tareas que contribuyen a lograr grandes fines.

FIN

Texto. Usado para hacer las pruebas de reconocimiento del OCR. Elaborado por Henry Criollo y Anthony Urresta

Texto extraído de la página guiainfantil.com que fue seleccionado a para realizar la captura de imagen por la cámara Arducam y así continuar con el proceso de extracción de texto de la imagen. Elaborado por Henry Criollo y Anthony Urresta

2.8 Verificación del texto extraído con el texto original

Para la evaluación y verificación de exactitud al extraer y posteriormente leer texto se evaluará la capacidad OCR del dispositivo, debido a que la exactitud de la lectura dependerá directamente de la capacidad OCR de la API, para cuantificar esta prueba se hará uso de un algoritmo de python que usará el modelo de la Distancia de Levenshtein (SequenceMatcher), para la comparación exacta incluyendo comas, espacios y mayúsculas, esto nos permite cuantificar la exactitud de lectura y capacidad OCR del dispositivos en varios tipos de texto como se puede observar en la tabla 3.5.

Tabla 3.5 Pruebas de exactitud de diferentes textos.

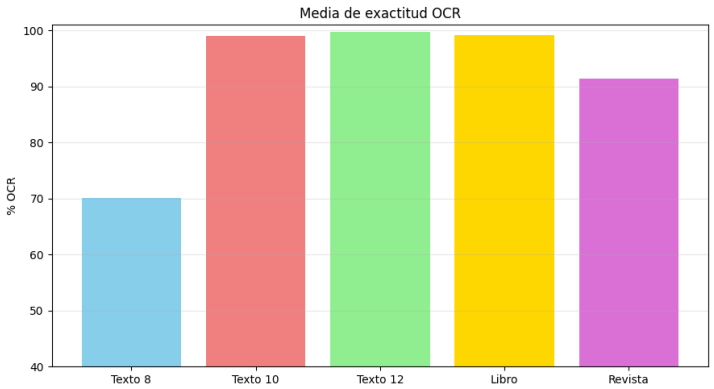
Fuentes evaluadas	Texto 8	Texto 10	Texto 12	Libro	Revista
№ Prueba	% OCR				
1	63,94%	99,90%	99,95%	99.78%	79.88%
2	73,65%	99,98%	99,93%	99,63%	99,31%
3	72,86%	98,83	99,36%	99,54%	98,59%
4	81,6%	98,97%	99,81%	99,62%	96,45%
5	85,3%	99,62%	99,94%	98,64%	87,36%
6	74,23%	98,4%	99,93%	97,84%	86,5%
7	69,32%	98,97%	99,39%	98,79%	92,63%
8	59,61%	99,23%	99,90%	99,63%	97,34%
9	46,65%	98,65%	99,41%	99,57%	84,63%
10	74,26%	98,35%	99,67%	99,23%	91,57%
$\bar{x}$	70.14%	99,09%	99,73%	99,23%	91,43%

Muestras tomadas con diferentes tipos de textos y diferentes tipos de letras para observar la funcionalidad del prototipo. Elaborado por Henry Criollo y Anthony Urresta

Para la síntesis de resultados obtenidos se hizo uso de la media aritmética con lo que se obtuvo los siguientes resultados como se muestran en la figura 2.19 donde con el texto tamaño 8 se logró un 70,14% de efectividad, con el texto tamaño 10 se logró un 99,09%, con el texto tamaño 12 se logró un 99,73%, con los resultados del libro se logró un 99.23% y finalmente con la revista se logró un 91,43%.

Siendo los mejores resultados los textos a tamaño 10,12 y el libro ya que se obtuvieron resultados mayores al 99% de exactitud.

Figura 2.19 grafica estadística de comparación de exactitud OCR.



Elaborado por Henry Criollo y Anthony Urresta

PRUEBAS DE LATENCIA EN FUNCIONAMIENTO

Para las pruebas de latencia se realizó evaluaciones con internet doméstico, internet de la Universidad Politécnica Salesiana y datos móviles con el fin de comparar que red garantiza una mejor latencia. Los resultados de las pruebas se muestran en la tabla 3.6.

Tabla 3.6 Latencia del dispositivo con diferentes tipos de red

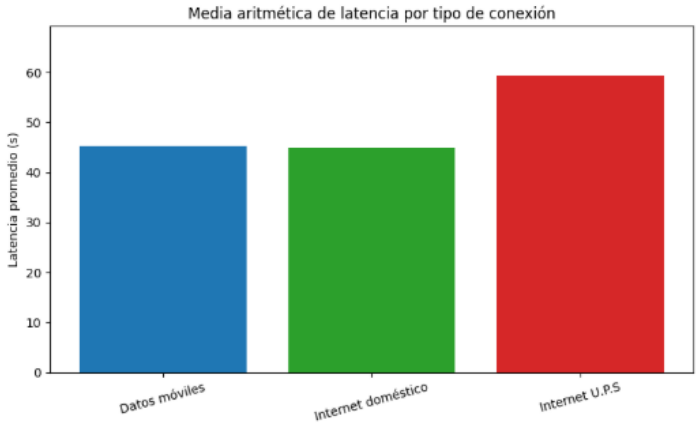
Pruebas de latencia			
№ Prueba	Datos móviles	Internet domestico	Internet U.P.S.
1	49s	43s	53
2	45s	44s	55
3	47s	43s	59
4	46s	45s	63s
5	43s	46s	65s
6	41s	44s	69s
7	43s	46s	64s
8	44s	45s	56s
9	46s	48s	54s
10	49s	46s	55s
x	45,3s	45s	59,3

Tabla en la que se observa las muestras recolectadas de la latencia con distintos tipos de red. Elaborado por Henry Criollo y Anthony Urresta.

Para la síntesis de resultados obtenidos se usó la media aritmética para obtener tiempos promedio de funcionamiento, con lo cual se verifico que haciendo uso de datos móviles se obtuvo 45,3 segundos de latencia en promedio, con internet domestico 45 segundos y

finalmente con el internet de la universidad se obtiene un promedio de 59,3 segundos las comparaciones de tiempos se muestran en la figura 2.x.

Figura 2.20 Gráfica de comparación latencia con distintos tipos de red.



Elaborado por Henry Criollo y Anthony Urresta

2.9 Pruebas de síntesis y reproducción de voz

En esta fase lo que se busco fue verificar que el sistema capaz no solo de leer sino de comunicar de forma clara y entendible al usuario el texto, dicha prueba evaluó la transformación de los datos digitales que vienen del OCR a n audio claro y entendible.

Se comienza enviando el texto extraído previamente por un OCR (OPENAI), para luego enviaros a una síntesis de voz (TTS) el cual genera un archivo .WAV. Ya con el archivo creado la Raspberry lo envía para reproducirlo mediante los audífonos de conducción ósea.

Una vez que se reprodujo el audio de manera concisa, clara y comprensible se determinó que el prototipo cumple con éxito todo el proceso del sistema haciendo de este un prototipo funcional, auditivamente seguro y comprensible.

2.10 Análisis de encuesta

Encuesta 1: Niños usuarios del dispositivo

Preguntas

- ¿Qué tan fácil fue colocarte el equipo?

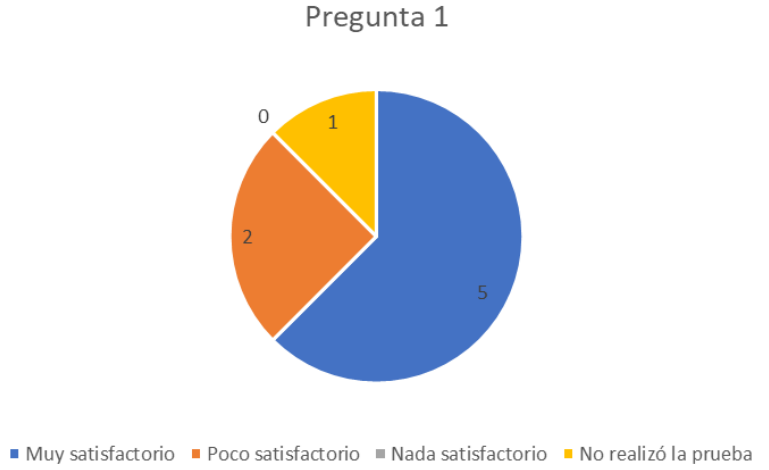
Tabla 3.7 Número de encuestados de la pregunta 1

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	5	62,5 %
Poco satisfactorio	2	25,0 %
Nada satisfactorio	0	0 %
No realizó la prueba	1	12,5 %
Total	8	100 %

Encuestados en la pregunta 1. Elaborado por Henry Criollo y Anthony Urresta

Cinco niños califico de muy satisfactoria el diseño funcional y la comodidad del dispositivo, por otro lado, dos niños presentaron inseguridad por ser un objeto desconocido sin embargo pudieron colorarse el dispositivo de manera cómoda.

Figura. 2.21 Encuesta realizada a los niños de discapacidad visual pregunta 1



Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta.

- ¿Qué tan fácil fue decir los comandos de voz al dispositivo?

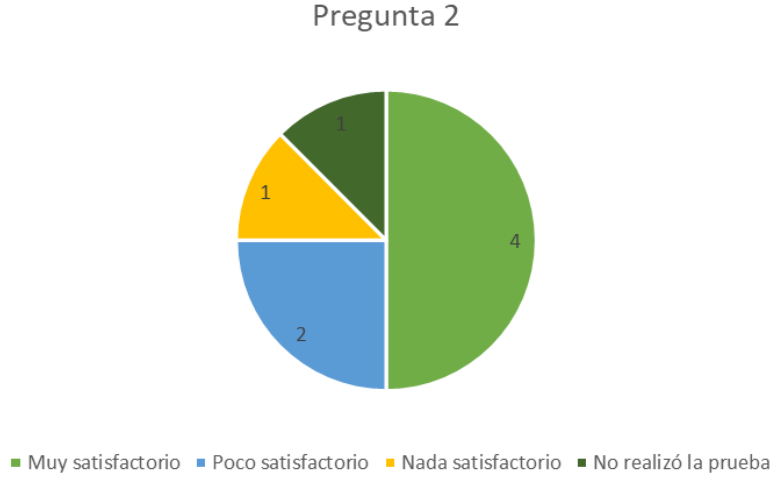
Tabla 3.8 Numero de encuestados de la pregunta 2

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	4	50,0 %
Poco satisfactorio	2	25,0 %
Nada satisfactorio	1	12,5 %
No realizó la prueba	1	12,5 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 2. Elaborado por Henry Criollo y Anthony Urresta

En esta ocasión cuatro de los ocho niños fueron no tuvieron problemas al decir los comandos al dispositivo, aunque dos niños si tuvieron problemas al decir los comandos pues al inicio no pronunciaban el comando con suficiente fuerza.

Figura. 2.22 Encuesta realizada a los niños de discapacidad visual pregunta 2



Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta

- ¿Qué tan satisfecho estás con la forma en que el dispositivo leyó los textos?

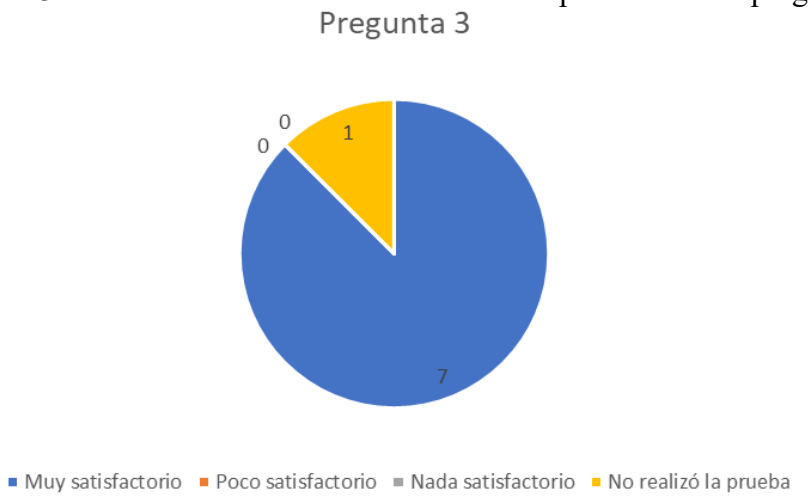
Tabla 3.9 Numero de encuestados de la pregunta 3

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	7	87,5 %
Poco satisfactorio	0	0 %
Nada satisfactorio	0	0 %
No realizó la prueba	1	12,5 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 3. Elaborado por Henry Criollo y Anthony Urresta

La satisfacción en este apartado es evidente pues todos los niños que hicieron la prueba estuvieron muy conformes con la reproducción del audio además que la reproducción por audífonos de conducción ósea les pareció interesante y estuvieron muy concentrados en la lectura del texto que claro y entendible, aunque un niño no realizo la prueba.

Figura. 2.23 Encuesta realizada a los niños de discapacidad visual pregunta 3



Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta

- ¿Qué tan cansado te sentiste al usar el equipo durante las pruebas de reconocimiento de imágenes y lectura?

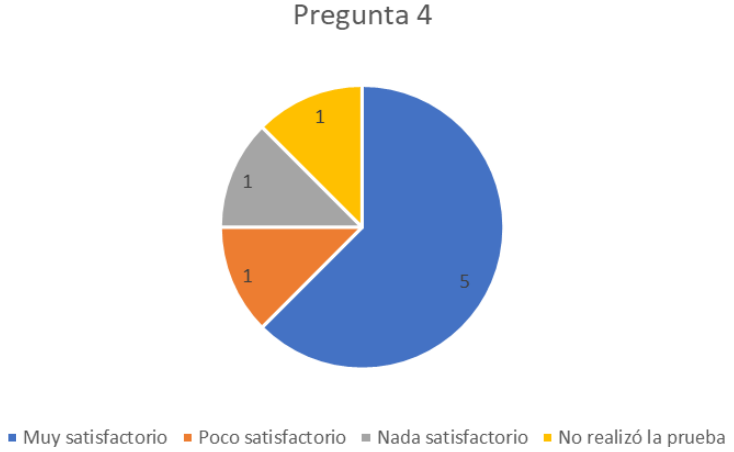
Tabla 3.10 Numero de encuestados de la pregunta 4

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	5	62,5 %
Poco satisfactorio	1	12,5 %
Nada satisfactorio	1	12,5 %
No realizó la prueba	1	12,5 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 4. Elaborado por Henry Criollo y Anthony Urresta

Al momento de preguntarles si sintieron cansancio de usar las gafas de audio lectura 5 niños no presentaron cansancio, aunque un niño si presento cansancio importante al terminar las pruebas y otro niño si paso por mucho cansancio mientras realizaba las pruebas.

Figura. 2.24 Encuesta realizada a los niños de discapacidad visual pregunta 3



Grafica que muestra a las personas que participaron en las pruebas del prototipo.
Elaborado por Henry Criollo y Anthony Urrest

Análisis general

La presente encuesta muestra que el prototipo de asistencia visual presenta un alto nivel de aceptación por parte de los niños de discapacidad visual encuestados además de ser muy eficiente pues la mayoría de los encuestados interactuó con el dispositivo sin inconvenientes tanto en la colocación del dispositivo, como los comandos de voz y la comprensión del texto leído.

Durante el transcurso de las pruebas cada niño presento diferencias en el proceso de adaptación propias del contexto de educación inicial pues algunos niños son defensivo táctil, es decir les asusta cualquier objeto extraño, o defensivo auditivo lo que significa que no les gusta los ruidos fuertes y los asusta como es el caso del niño que no fue capaz de realixzar la prueba pues era tanto defensivo táctil como defensivo auditivo.

Gracias a tener siempre presente a un docente durante las pruebas por comodidad del niño y que los niños deben estar siempre bajo su supervisión, así se logró una mejor adaptación pues no requirieron mayor esfuerzo para hacer uso de los comandos ni durante la etapa de lectura del texto por los audífonos de conducción ósea. Este aprendizaje de cómo es el entorno de los niños con discapacidades confirman la necesidad de dispositivos asistivos flexibles y adaptables para una inclusión adecuada en entornos educativos.

Encuesta 2: Padres de familia y docentes

Preguntas

- ¿Permitiría que su hijo o estudiante use este dispositivo de asistencia visual?

Tabla 3.11 Numero de encuestados de la pregunta 1

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	8	100 %
Poco satisfactorio	0	0 %
Nada satisfactorio	0	0 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 1. Elaborado por Henry Criollo y Anthony Urresta

Todos los profesores están de acuerdo admitir el uso del dispositivo de asistencia visual para sus estudiantes, lo que evidencia una gran confianza y seguridad hacia el prototipo como herramienta de apoyo para el aprendizaje de los niños con discapacidad visual.

Figura. 2.25 Encuesta realizada a los profesores de la Institución Unidad especializada Mariana de Jesús pregunta 1

Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta

- ¿Considera conveniente el uso de este dispositivo para las actividades de lectura del niño?

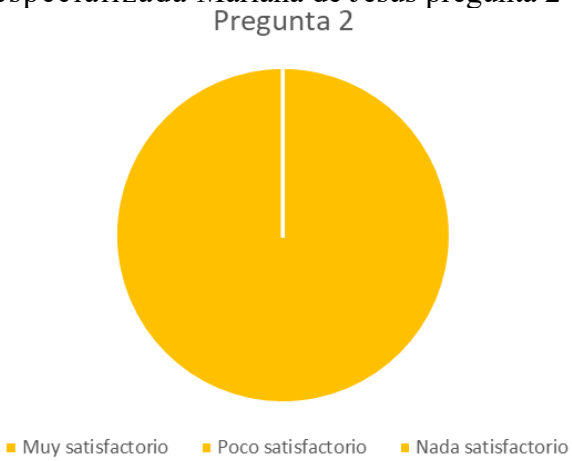
Tabla 3.12 Numero de encuestados de la pregunta 2

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	8	100 %
Poco satisfactorio	0	0 %
Nada satisfactorio	0	0 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 2. Elaborado por Henry Criollo y Anthony Urresta

Todos los maestros encuestados tienen una opinión muy satisfactoria en hacer so del prototipo en las actividades académicas de los niños para facilitarle el acceso a a información a la vez que también ayuda a la comprensión de textos en especial con textos impresos pues representa una barrera para ellos.

Figura. 2.26 Encuesta realizada a los profesores de la Institución Unidad especializada Mariana de Jesús pregunta 2



Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta

- ¿Incorporaría este dispositivo en el hogar o en el colegio como un sistema de apoyo para que los niños puedan escuchar el entorno mediante auriculares de conducción ósea?

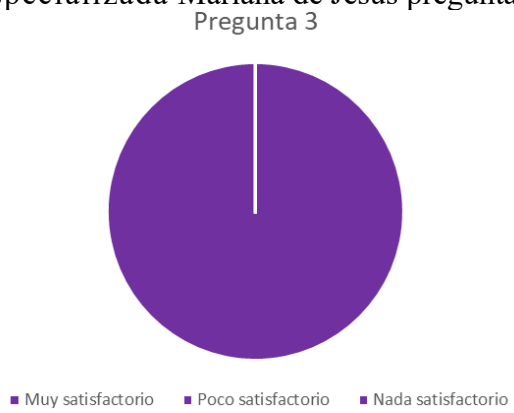
Tabla 3.13 Numero de encuestados de la pregunta 3

Respuesta	Cantidad	Porcentaje
Muy satisfactorio	8	100 %
Poco satisfactorio	0	0 %
Nada satisfactorio	0	0 %
Total	8	100 %

Muestra el total de encuestados en la pregunta 3. Elaborado por Henry Criollo y Anthony Urresta

Todos los maestros encuetados calificaron con total satisfacción el uso de este dispositivo en el hogar lo que resalta la capacidad del prototipo de adaptarse a la vida diaria del usuario, siendo un punto fuerte los audífonos de conducción ósea que permiten a os niños recibir información del entorno favoreciendo la seguridad del usuario en el aula y en la casa.

Figura. 2.27 Encuesta realizada a los profesores de la Institución Unidad especializada Mariana de Jesús pregunta 3



Grafica que muestra a las personas que participaron en las pruebas del prototipo. Elaborado por Henry Criollo y Anthony Urresta

CONCLUSIONES

Con la finalización satisfactoria del desarrollo de este prototipo de asistencia visual, demostrando así la integración entre hardware y software avanzado los cuáles al combinar una Raspberry Pi 4 con servicios de inteligencia artificia como OpenCV y los audífonos de conducción ósea se logró un sistema capaz de transformar el texto de las imágenes en información auditiva precisa, brindando una solución tecnológica al usuario, en este caso los estudiantes con discapacidad visual de la Institución Unidad Especializada Mariana de Jesús.

El diseño del armazón de las gafas de asistencia visual mediante software CAD es cómodo, ergonómico y de fácil uso por parte del usuario como lo demuestra la tabla 3.7 con un 62.5% de los niños menciono que es fácil y cómodo colocarse las gafas, mientras que un 25% menciono que es poco satisfactorio dejando claro que la mayoría de los niños se adaptó rápidamente al diseño propuesto sin dificultades. Asimismo, en relación con la comodidad durante el uso del dispositivo y tomando de referencia los resultados obtenidos en la tabla 3.9, el 87.5% de los niños califico la experiencia de muy satisfactoria frente a un 12.5% que fue del niño que no participó en la encuesta por ser muy sensitivo táctil y sensitivo audible. Estos resultados confirman que le diseño CAD permitió la integración adecuada de componentes y acondicionamiento óptimo para el usuario.

Para la interacción humano maquina mediante la integración de tecnologías OT e IT se cumplió con el objetivo planteado evidenciado en los resultados de la tabla 3.8 que un 50% de los niños encuestados no presentaron ninguna dificultad al emitir los comandos de voz, al 25% presentaron dificultades iniciales relacionadas con la pronunciación y el volumen d la voz que fueron superadas con la ayuda del docente a cargo durante las pruebas y 12.5% que a pesar de realizar las pruebas no sintió muy cómodo con el uso de comandos de voz. A través de la tabla 3.10 muestra que el 62.5% no se sintió cansado mientras usaba el dispositivo, un 12.5% si presento leves signos de cansancio al igual que un 12.5% no estuvo nada satisfecho debido a que si se cansó usando el prototipo. Estos resultados evidencian que el asistente de voz basado el reconocimiento de voz en español constituye una interfaz adecuada para persona con discapacidad visual eliminado la necesidad de controles físicos favoreciendo una interacción más intuitiva.

La implementación de la arquitectura de comunicación entre el asistente visual y la nube mediante tecnologías IOT fue exitosa y permitió cumplir con el objetivo propuesto De acuerdo con los resultados de la pregunta 4 de la encuesta aplicada a los niños, el 62.7% de los usuarios calificó el funcionamiento del dispositivo durante la lectura como muy satisfactorio, mientras que el 12.5% lo consideró poco satisfactorio, registrando valoraciones de insatisfacción en 12.5%. Estos resultados evidencian que el sistema logró establecer una comunicación efectiva con la nube para el procesamiento de la información y la generación de respuestas en audio, aunque una parte de los usuarios percibió ligeras demoras o dificultades propias del uso de servicios en la nube. No obstante, las valoraciones negativas fueron mínimas lo que confirma que la arquitectura IOT implementada es funcional y viable, constituyéndose como un componente clave del sistema de asistencia visual y sentando las bases para futuras mejoras orientadas a optimizar la experiencia del usuario.

El funcionamiento del asistente visual fue evaluado y dado como exitoso, mediante la ejecución de pruebas de campo en espacios controlados, lo que permitió validar el desempeño general del sistema en condiciones reales de uso, durante estas pruebas, el dispositivo demostró estabilidad en la captura de imágenes siendo de 32MP la mejor resolución para tomar fotografías con un 86.67% de éxito en las pruebas realizadas, procesamiento de la información siendo en este apartado el tamaño de letra 12 la mejor opción con un 99.73% de prueba realizadas con éxito y reproducción de audio, permitiendo a los usuarios interactuar con el sistema de forma continua y comprensible. Los resultados obtenidos evidencian que el asistente visual respondió adecuadamente ante diferentes textos y condiciones de prueba, manteniendo un nivel aceptable de precisión y tiempos de respuesta acordes a los requerimientos del proyecto. Asimismo, los maestros confirman que el sistema puede ser utilizado de manera funcional dentro de un entorno educativo controlado, pues estuvieron 100% de acuerdo en este apartado así también con que se use este dispositivo para actividades de lectura validando así su aplicabilidad como herramienta de apoyo para la lectura asistida en personas con discapacidad visual.

RECOMENDACIONES

En primer lugar, se recomienda optimizar el diseño del armazón de las gafas de asistencia visual, considerando materiales aún más ligeros y flexibles, con el objetivo de mejorar la adaptación del dispositivo en niños que presentan hipersensibilidad táctil o auditiva. Esto permitiría reducir los porcentajes de insatisfacción observados en las encuestas y aumentar la aceptación del dispositivo por parte de todos los usuarios.

En cuanto a la interacción humano–máquina mediante comandos de voz, se recomienda incorporar un proceso de calibración inicial de voz y volumen para cada usuario, así como implementar retroalimentación auditiva guiada, con el fin de facilitar el uso del asistente virtual en niños que presentan dificultades iniciales de pronunciación. De esta manera, se podría mejorar la experiencia de uso y aumentar el porcentaje de interacción satisfactoria con el sistema.

Respecto a la arquitectura de comunicación entre el asistente visual y la nube mediante tecnologías IoT, se recomienda optimizar los tiempos de transmisión y respuesta, ya sea mediante el uso de conexiones a internet más estables o la incorporación de procesos locales de respaldo (edge computing), con el fin de minimizar las demoras percibidas por algunos usuarios durante el proceso de lectura.

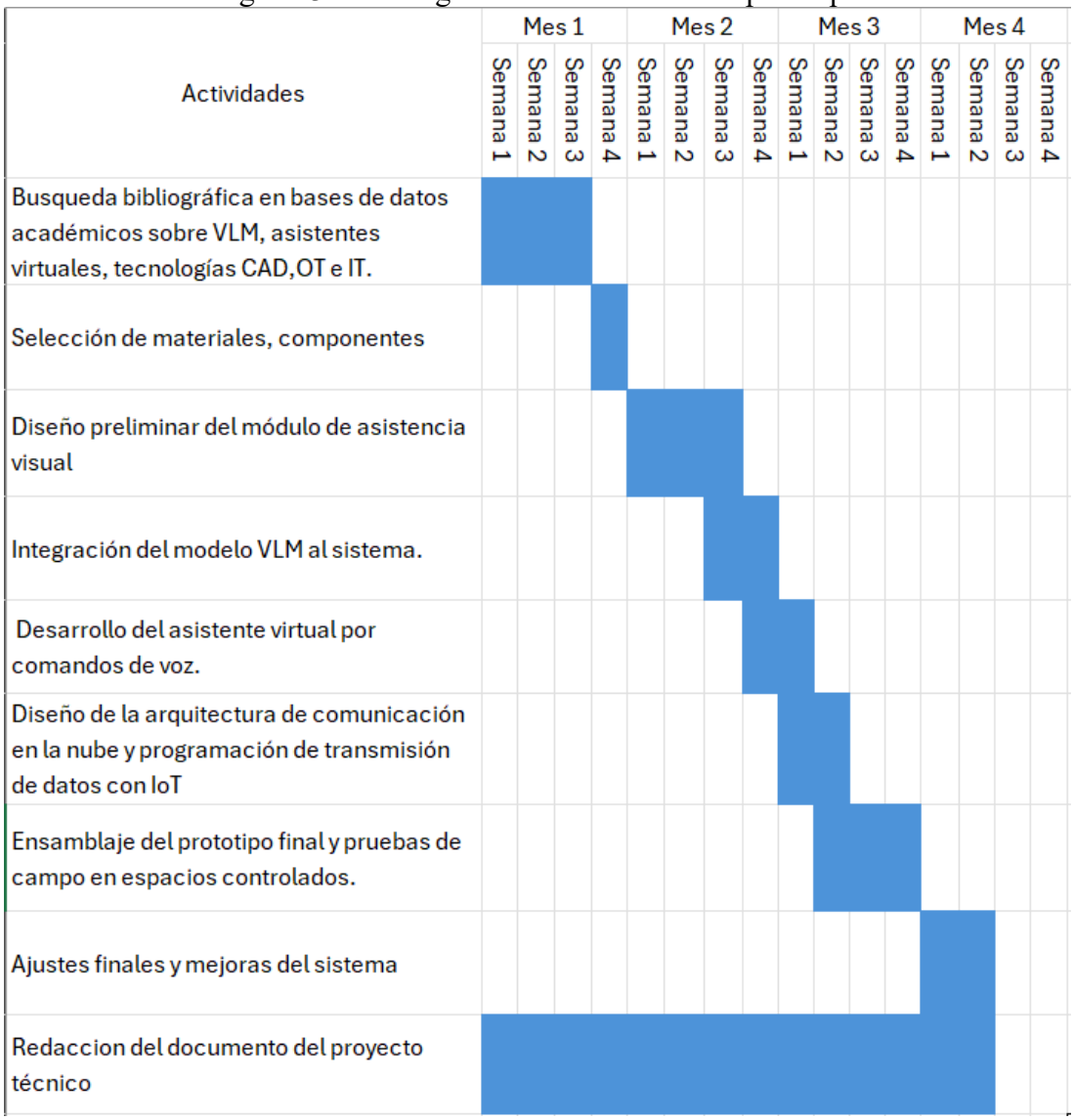
Asimismo, se recomienda mantener la configuración óptima identificada durante las pruebas de campo, utilizando una resolución de imagen de 32 MP y un tamaño de letra 12, ya que estos parámetros demostraron altos niveles de éxito en el reconocimiento y procesamiento del texto. No obstante, se sugiere implementar un ajuste automático de estos parámetros para adaptarse a diferentes tipos de texto y condiciones de iluminación.

Finalmente, se recomienda ampliar las pruebas de campo a entornos menos controlados y con un mayor número de usuarios, con el fin de evaluar el desempeño del sistema en escenarios reales más exigentes. Esto permitirá validar la escalabilidad del prototipo y su potencial implementación como una herramienta tecnológica de apoyo permanente para estudiantes con discapacidad visual en instituciones educativas.

CAPITULO III

CRONOGRAMA

Figura. 3.1 Cronograma del desarrollo del prototipo



Elaborado por Henry Criollo y Anthony Urresta

PRESUPUESTO

Figura. 3.2 Presupuesto usado para el prototipo

Materiales		Cantidad	Costo Unitari	Costo Total
Raspberry Pi 4	UNIDADES	1,00	$200,00	$200,00
Módulo cámara 64 MP	UNIDADES	1,00	$115,00	$115,00
Impresión 3D	UNIDADES	1,00	$90,00	$90,00
Auricular conducción ósea	UNIDADES	1,00	$30,00	$30,00
Cables de conexión	UNIDADES	1,00	$0,70	$0,70
Micrófono externo	UNIDADES	1,00	$18,00	$18,00
SUBTOTAL MATERIALES / INSUMOS				**$453,70**

Elaborado por Henry Criollo y Anthony Urresta

Cabe mencionar que durante la realización de las pruebas del prototipo el micrófono, propio de los auriculares de conducción ósea no era eficiente al momento de captar los comandos de voz por ende se optó por usar un micrófono de mesa el cual se agregó después al presupuesto inicial con el fin de manejar la captación de comandos y asegurar el funcionamiento eficiente.

REFERENCIAS

Acosta, J. (2022). *Síntesis del habla por medio de generación sintética de Embeddings de Tacotron 2 Multispeaker, basándose en el género de voz.* Ciudad de México: Universidad Autónoma de México. Obtenido de https://calebrascon.info/pubre/ThesisMas.2022.VoiceSynth.pdf

Agastin, J., Hareesh, P., Hariharan, K., & Kailash, K. (2025). Real-Time Object Detection Using Raspberry Pi Zero 2W: An Optimized Approach. *International Research Journal on Advanced Engineering and Management, 3*(12), 3401-3405. doi:https://doi.org/10.47392/IRJAEM.2025.0537

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., . . . Samangooei, S. (2022). Flamingo: a Visual Language Model for Few-Shot Learning. *Arxib*, 1-54. doi:https://doi.org/10.48550/arXiv.2204.14198

Arranz, E. (22 de Mayo de 2023). *Cuáles son los tipos de discapacidad visual*. Obtenido de Fundación Adecco: https://fundacionadecco.org/blog/cuales-son-los-tipos-de-discapacidad-visual/

Balbi, M. (24 de Enero de 2017). *La solución para no videntes que convierte texto en voz*. Obtenido de Infobae: https://www.infobae.com/play-tv/2017/01/24/la-solucion-para-no-videntes-que-convierte-texto-en-voz/

Basantes, A., Guerra, F., Naranjo, M., & Ibadango, D. (2018). Los Lectores de Pantalla: Herramientas Tecnológicas para la nclusión Educativa de Personas no Videntes. *Información Tecnológica, 29*(5), 81-90. Obtenido de https://www.scielo.cl/pdf/infotec/v29n5/0718-0764-infotec-29-05-00081.pdf

Basantes, D., & Chalaco, E. (2019). *Desarrollo de un prototipo de gafas para lectura de texto con visión artificial que asista a personas con discapacidad visual.* Quito: Universidad Politécnica Salesiana. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/17841/1/UPS%20-%20ST004367.pdf

Benavides, T., Bedoya, N., & Cruz, E. (2023). El acceso a tecnologías de asistencia por parte de personas con discapacidad. *Revista crítica de ciencias sociais*(132), 99-120. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=9370450

Calle, A., López, A., & Campillay, A. (2021). Inclusión social de las personas con discapacidad visual: Una revisión sistemática cualitativa. *New Trends in*

Qualitative Research, 8, 617-629. doi:https://doi.org/10.36367/ntqr.8.2021.617-629

Cardona, A., & Vasquez, R. (2021). Dispositivos de asistencia para la movilidad en personas con discapacidad visual: Una revisión bibliográfica. *Revista Politécnica, 15*(28), 107-116. doi:https://doi.org/10.33571/rpolitec.v15n28a10

Consejo Nacional para la Igualdad de Discapacidades. (20 de Enero de 2026). *Estadísticas de discapacidad*. Obtenido de Programas y servicios: https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Cotán, A., Álvarez, K., Márquez, J., & Gallardo-López, J. (2024). Recursos tecnológicos y educación inclusiva: propuestas y recomendaciones de estudiantes universitarios con discapacidad. *Edutec*, 111-127. doi:https://doi.org/10.21556/edutec.2024.90.3521

Goyes, J. (2020). *Estudio de impacto del modelo cloud computing en la gestión de servicios de información gerencial en la banca privada.* Quito: Universidad Andina Simón Bolívar. Obtenido de https://repositorio.uasb.edu.ec/bitstream/10644/7468/1/T3265-MAE-Goyes-Estudio.pdf

Hoang, H., Linh, P., & Sieu, H. (2026). Building an AI-Integrated Braille Recognition Model for Blind Education: From Mechanical Devices to Personalized Learning Systems. *World Journal of Engineering and Technology, 14*(1), 1-13. doi:https://doi.org/10.4236/wjet.2026.141001

Martínez, J. (2021). *Desarrollo de una arquitectura eficiente orientada a microservicios con API rest utilizando la calidad externa de las normas ISO/IEC 25023.* Ibarra: Universidad Técnica del Norte. Obtenido de https://repositorio.utn.edu.ec/handle/123456789/15165?mode=full

Morales, E., Martínez, J., Fuentes, D., Morgado, I., & Aristeo, C. (2025). El software CAD como herramienta para la enseñanza en la ingeniería. *Ciencia Latino tecnología, 9*(2), 7593-7606. Obtenido de https://cienciala tina.org/index.php/cienciala/article/view/17484

Moreno, A., Cruz, L., Capistrán, L., & Segura, L. (2023). La intervención con perspectiva transdisciplinar de la diversidad auditiva. *Cruz, Laura, 8*(2), 105-117. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=9986031

Naayin, P., Kumar, P., Bura, C., Kumar, A., & Kamatala, S. (2025). AI-Powered Assitive Technologies for Visual Impairment. *Arxiv*, 1-12. doi:https://doi.org/10.48550/arXiv.2503.15494

Organización Mundial de la Salud. (10 de Agosto de 2023). *Blindness and vision impairment*. Obtenido de Detail: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

Pancho, G., & Galarza, F. (2014). Formulación de un Marco de Referencia de Convergencia IT/OT. *Revista Técnica Energía, 10*(1), 208-214. Obtenido de https://revistaenergia.cenace.gob.ec/index.php/cenace/article/view/117

Parra, F., & Chinchero, J. (2019). *Desarrollo de un dispositivo que mediante visión artificial permita adquirir imágenes con palabras para la conversión a audio, orientado a la ayuda de personas invidentes.* Quito: Universidad Israel. Obtenido de https://repositorio.uisrael.edu.ec/handle/47000/2117

Saira, M., & Escobedo, F. (2025). *Braille y Tecnología: Innovaciones y Desafíos en la Era Digital.* Ucayali: Universidad Nacional de Ucayali. Obtenido de http://revistas.unu.edu.pe/index.php/iu/article/view/170

Saiz de la Hoya, M. (2024). *Igualdad de oportunidades educativas: Derechos y desafíos de las personas con discapacidad visual en España.* Madrid: Universidad Pontificia Comillas. Obtenido de https://repositorio.comillas.edu/jspui/handle/11531/81497

Tan, X. (24 de Enero de 2021). *TTS tutorial @ ISCSLP 2021.* Obtenido de Microsoft Research Asia: https://www.microsoft.com/en-us/research/wp-content/uploads/2021/07/ISCSLP2021-TTS-Tutorial-Xu-Tan.pdf

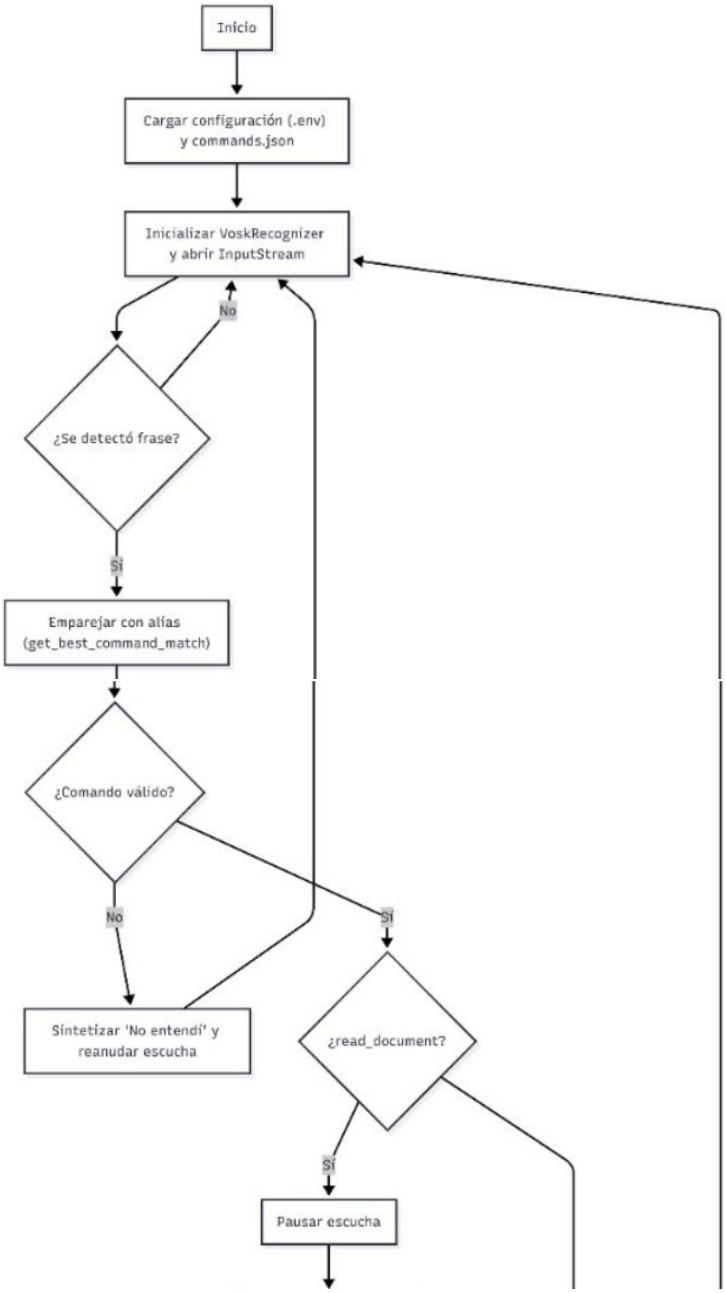
Unión Internacional de Telecomunicaciones. (Enero de 2025). *Recomendación UIT-T Y.4231 (01/2025): Requisitos y marco de capacidades de la Internet de las cosas para la visión.* Obtenido de Unión Internacional de Telecomunicaciones: https://www.itu.int/rec/dologin_pub.asp?id=T-REC-Y.4231-202501-I%21%21PDF-S&lang=f&type=items

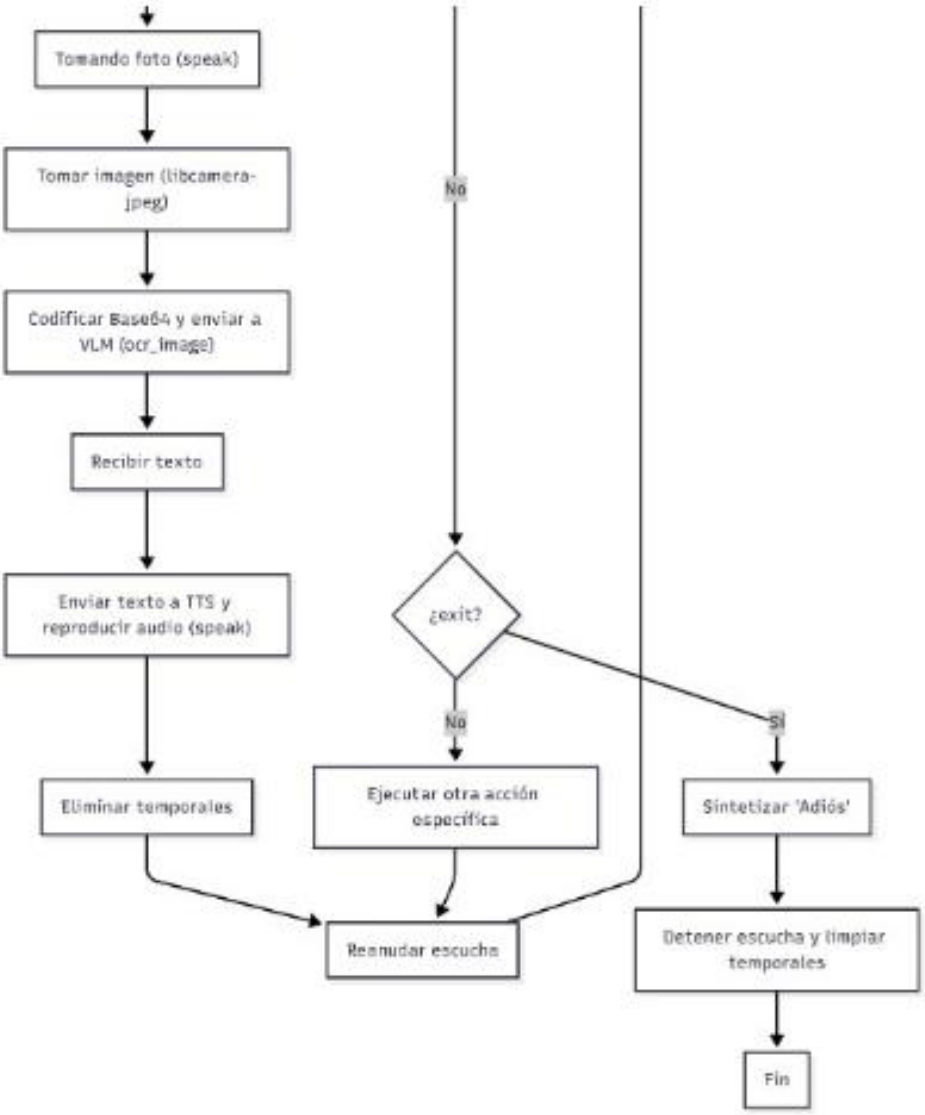
Unión Latinoamericana de Ciegos. (2023). *Manual de rehabilitación para personas con discapacidad visual de América Latina.* Montevideo: ICEVI Latinoamérica. Obtenido de https://www.ulacdigital.org/media/attachments/2023/11/16/manual-de-rehabilitacion-america-latina.pdf

Unión Mundial de Ciegos. (5 de Enero de 2017). *Comunicado de la Unión Mundial de Ciegos, sobre la Importancia de la alfabetización Braille*. Obtenido de Noticias: https://www.riadis.org/comunicado-union-mundial-de-ciegos-alfabetizacion-braille/

ANEXOS

Anexo. 1.1 Flujograma del prototipo




Tomando foto (speak)
Tomar imagen (libcamera-jpeg)
Codificar Base64 y enviar a VLM (ocr_image)
Recibir texto
Enviar texto a TTS y reproducir audio (speak)
Eliminar temporales
No
¿exit?
No
Ejecutar otra acción específica
Sí
Sintetizar 'Adiós'
Reanudar escucha
Detener escucha y limpiar temporales
Fin

Anexo. 1.2 Encuesta de satisfacción para niños con discapacidad visual

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

ENCUESTAS DE EVALUACIÓN DEL DISPOSITIVO DE ASISTENCIA VISUAL PARA LECTURA

Instrucciones generales: Marque con una X la opción que mejor represente su experiencia u opinión.

Escala de respuesta: - **Mucho satisfactorio** – **Poco satisfactorio** – **Nada satisfactorio**

ENCUESTA 1: NIÑOS USUARIOS DEL DISPOSITIVO

Preguntas

1. ¿Qué tan fácil fue colocarte el equipo?
   - [x] Muy satisfactorio
   - [ ] Poco satisfactorio
   - [ ] Nada satisfactorio
2. ¿Qué tan fácil fue decir los comandos de voz al dispositivo?
   - [x] Muy satisfactorio
   - [ ] Poco satisfactorio
   - [ ] Nada satisfactorio
3. ¿Qué tan satisfecho estás con la forma en que el dispositivo leyó los textos?
   - [x] Muy satisfactorio
   - [ ] Poco satisfactorio
   - [ ] Nada satisfactorio
4. ¿Qué tan cansado te sentiste al usar el equipo durante las pruebas de reconocimiento de imágenes y lectura?
   - [ ] Muy satisfactorio
   - [x] Poco satisfactorio
   - [ ] Nada satisfactorio

Anexo. 1.3 Encuesta de satisfacción para niños con discapacidad visual

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

ENCUESTAS DE EVALUACIÓN DEL DISPOSITIVO DE ASISTENCIA VISUAL PARA LECTURA

Instrucciones generales: Marque con una X la opción que mejor represente su experiencia u opinión.

Escala de respuesta: - **Mucho satisfactorio – Poco satisfactorio – Nada satisfactorio**

ENCUESTA 1: NIÑOS USUARIOS DEL DISPOSITIVO

Preguntas

1. ¿Qué tan fácil fue colocarte el equipo?
- [x] Muy satisfactorio
- [ ] Poco satisfactorio
- [ ] Nada satisfactorio

2. ¿Qué tan fácil fue decir los comandos de voz al dispositivo?
- [x] Muy satisfactorio
- [ ] Poco satisfactorio
- [x] Nada satisfactorio

3. ¿Qué tan satisfecho estás con la forma en que el dispositivo leyó los textos?
- [x] Muy satisfactorio
- [ ] Poco satisfactorio
- [ ] Nada satisfactorio

4. ¿Qué tan cansado te sentiste al usar el equipo durante las pruebas de reconocimiento de imágenes y lectura?
- [x] Muy satisfactorio
- [ ] Poco satisfactorio
- [ ] Nada satisfactorio

Anexo. 1.4 Encuesta de satisfacción para padres

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

ENCUESTAS DE EVALUACIÓN DEL DISPOSITIVO DE ASISTENCIA VISUAL PARA LECTURA

Instrucciones generales: Marque con una X la opción que mejor represente su experiencia u opinión.

Escala de respuesta: - **Muy satisfactorio – Poco satisfactorio – Nada satisfactorio**

ENCUESTA 2: PADRES DE FAMILIA Y DOCENTES

Preguntas

1. ¿Permitiría que su hijo o estudiante use este dispositivo de asistencia visual?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

2. ¿Considera conveniente el uso de este dispositivo para las actividades de lectura del niño?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

3. ¿Incorporaría este dispositivo en el hogar o en el colegio como un sistema de apoyo para que los niños puedan escuchar el entorno mediante auriculares de conducción ósea?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

Anexo. 1.5 Encuesta de satisfacción para padres

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

ENCUESTAS DE EVALUACIÓN DEL DISPOSITIVO DE ASISTENCIA VISUAL PARA LECTURA

Instrucciones generales: Marque con una ✕ la opción que mejor represente su experiencia u opinión.

Escala de respuesta: - **Muy satisfactorio – Poco satisfactorio – Nada satisfactorio**

ENCUESTA 2: PADRES DE FAMILIA Y DOCENTES

Preguntas

1. ¿Permitiría que su hijo o estudiante use este dispositivo de asistencia visual?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

2. ¿Considera conveniente el uso de este dispositivo para las actividades de lectura del niño?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

3. ¿Incorporaría este dispositivo en el hogar o en el colegio como un sistema de apoyo para que los niños puedan escuchar el entorno mediante auriculares de conducción ósea?

☑ Muy satisfactorio
☐ Poco satisfactorio
☐ Nada satisfactorio

Anexo. 1.6 Prueba en Institución

Anexo. 1.7 Prueba en Institución

Anexo 1.8 Código fuente - main.py

```
import json
import os
import threading
import difflib
import sys
import time
from pathlib import Path


from audio.recognizer import initialize_recognizer, start_listening, stop_listening,
listen_command, pause_listening, resume_listening
from audio.speaker import speak
from vision.camera import take_picture
from vision.ocr import ocr_image
from utils.internet import check_internet
from config import TEMP_DIR


command_lock = threading.Lock()


def get_best_command_match(text, command_aliases, cutoff=0.6):
    """Encuentra la mejor coincidencia de comando usando búsqueda difusa"""
    if not text or not command_aliases:
        return None

    # Normalizar el texto de entrada
    text = text.lower().strip()

    all_phrases = {
        alias.lower(): action
        for action, aliases in command_aliases.items()
        for alias in aliases
    }

    match = difflib.get_close_matches(text, all_phrases.keys(), n=1, cutoff=cutoff)
    return all_phrases[match[0]] if match else None


def load_commands_from_file(file_path="commands.json"):
    """Carga comandos desde archivo JSON o retorna comandos por defecto"""
    try:
        if os.path.exists(file_path):
            with open(file_path, "r", encoding="utf-8") as f:
                commands = json.load(f)
                if not isinstance(commands, dict):
                    raise ValueError("El archivo de comandos debe contener un
diccionario")
                # Validar que cada comando tenga una lista de aliases
                for action, aliases in commands.items():
                    if not isinstance(aliases, list) or not aliases:
                        raise ValueError(f"El comando '{action}' debe tener una lista no
vacía de aliases")
                return commands
    except (json.JSONDecodeError, ValueError, FileNotFoundError) as e:
        print(f"Error cargando comandos desde {file_path}: {e}")
        print("Usando comandos por defecto...")

    # Comandos por defecto simplificados para Raspberry Pi
    return {
        "read_document": [
            "leer documento", "quiero que leas", "puedes leer esto",
            "lee esto", "lee el documento", "leer texto"
        ],
        "exit": ["salir", "terminar", "adiós", "bye", "cerrar"]
    }


def cleanup_temp_file(filename):
    """Limpia archivos temporales de manera segura"""
    if filename and os.path.exists(filename):
        try:
            os.remove(filename)
            print(f"Archivo temporal eliminado: {filename}")
        except OSError as e:
            print(f"No se pudo eliminar archivo temporal {filename}: {e}")
```

```
def handle_command(action):
    """Maneja la ejecución de comandos con bloqueo para evitar concurrencia"""
    filename = None
    try:
        with command_lock:
            # Pausar la escucha mientras se procesa el comando
            pause_listening()

            if action == "read_document":
                speak("Tomando foto del documento...")
                filename = take_picture()

                if not filename:
                    speak("No pude tomar la foto del documento.")
                    return

                # Verificar conexión
                if not check_internet():
                    speak("Sin conexión a internet.")
                else:
                    speak("Procesando documento con inteligencia artificial.")

                text = ocr_image(filename)
                if text and text.strip():
                    speak(f"El documento dice: {text}")
                else:
                    speak("No pude leer texto en el documento.")

                speak("Comando completado. Puedes dar otro comando o decir 'salir' para
terminar.")

            elif action == "exit":
                speak("Hasta luego.")
                stop_listening()
                cleanup_temp_files()
                sys.exit(0)

    except Exception as e:
        print(f"Error ejecutando comando {action}: {e}")
        speak("Ocurrió un error ejecutando el comando.")
    finally:
        # Siempre limpiar el archivo temporal
        if filename:
            cleanup_temp_file(filename)
        # Reanudar la escucha al finalizar el comando
        resume_listening()


def cleanup_temp_files():
    """Limpia todos los archivos temporales al salir"""
    try:
        for file_path in TEMP_DIR.glob("*"):
            if file_path.is_file():
                file_path.unlink()
        print("Archivos temporales limpiados.")
    except Exception as e:
        print(f"Error limpiando archivos temporales: {e}")


def main():
    """Función principal del asistente"""
    try:
        print("Iniciando asistente de voz...")

        # Mostrar configuración actual
        print(f"Configuración - OCR: OpenAI")
        print(f"Configuración - TTS: OpenAI")

        # Cargar comandos
        known_commands = load_commands_from_file()
        if not known_commands:
            print("Error: No se pudieron cargar los comandos")
            return
```

```
        # Inicializar reconocedor de voz
        print("Inicializando reconocedor de voz...")
        if not initialize_recognizer():
            print("Error: No se pudo inicializar el reconocedor de voz")
            return

        # Iniciar escucha
        print("Configurando escucha de audio...")
        if not start_listening():
            print("Error: No se pudo iniciar la escucha de audio")
            return


        def listen_loop():
            """Bucle principal de escucha de comandos"""
            # Ahora sí está realmente listo
            speak("Asistente listo. Di un comando para comenzar.")

            while True:
                try:
                    command_text = listen_command()
                    if command_text:
                        print(f"Comando detectado: {command_text}")
                        action = get_best_command_match(command_text, known_commands)

                        if action:
                            if command_lock.locked():
                                speak("Espera un momento, estoy procesando otro comando.")
                            else:
                                # Ejecutar comando en hilo separado
                                threading.Thread(
                                    target=handle_command,
                                    args=(action,),
                                    daemon=True
                                ).start()
                        else:
                            speak("No entendí el comando. Intenta de nuevo.")
                    else:
                        # Pequeña pausa para no saturar el CPU
                        time.sleep(0.1)

                except KeyboardInterrupt:
                    break
                except Exception as e:
                    print(f"Error en listen_loop: {e}")
                    speak("Error detectando comando.")
                    time.sleep(1)  # Pausa antes de continuar


        # Iniciar hilo de escucha
        listen_thread = threading.Thread(target=listen_loop, daemon=True)
        listen_thread.start()


        try:
            # Mantener el programa ejecutándose
            while listen_thread.is_alive():
                listen_thread.join(timeout=1)
        except KeyboardInterrupt:
            print("\nInterrumpido por el usuario")
            speak("Cerrando asistente.")
            stop_listening()
            cleanup_temp_files()
            sys.exit(0)

    except Exception as e:
        print(f"Error en main: {e}")
        stop_listening()
        cleanup_temp_files()
        sys.exit(1)

if __name__ == "__main__":
    main()
```

Anexo 1.9 Código fuente - config.py

```
import os
from pathlib import Path
from dotenv import load_dotenv

# Cargar variables de entorno desde archivo .env si existe
load_dotenv()

# Configuración de API
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

# Configuración de TTS (solo OpenAI)
OPENAI_TTS_VOICE = os.getenv("OPENAI_TTS_VOICE", "alloy")  # Opciones: alloy, echo, fable,
onyx, nova, shimmer

# Configuración de modelos
VOSK_MODEL_PATH = os.path.expanduser("./models/vosk-model-small-es-0.42")

# Configuración de archivos temporales
TEMP_DIR = Path.cwd() / "temp"
TEMP_DIR.mkdir(exist_ok=True)

# Configuración de TTS
TTS_OUTPUT_FILE = TEMP_DIR / "output.wav"

# Configuración de cámara
DEFAULT_IMAGE_FILENAME = TEMP_DIR / "captured_image.jpg"
# Rotación de la imagen en grados (0, 90, 180, 270)
# 0 = sin rotación, 90 = 90° horario, 180 = boca abajo, 270 = 90° antihorario
CAMERA_ROTATION = int(os.getenv("CAMERA_ROTATION", "0"))

# Configuración de reconocimiento de voz
SAMPLE_RATE = 16000
BLOCK_SIZE = 8000

# Tiempos de espera
INTERNET_CHECK_TIMEOUT = 3
API_REQUEST_TIMEOUT = 30
```

Anexo 1.10 Código fuente - audio/recognizer.py

```
import queue
import json
import sounddevice as sd
from vosk import Model, KaldiRecognizer
from config import VOSK_MODEL_PATH


class VoskRecognizer:
    def __init__(self):
        self.q = queue.Queue()
        self.model = None
        self.recognizer = None
        self.stream = None
        self.initialized = False
        self.listening = False
        self.paused = False

    def _callback(self, indata, frames, time, status):
        if status:
            print("Status:", status)
        # Solo procesar audio si no está pausado
        if not self.paused:
            self.q.put(bytes(indata))

    def initialize(self):
        """Inicializa el modelo Vosk y el reconocedor"""
        try:
            print("Cargando modelo Vosk...")
            self.model = Model(VOSK_MODEL_PATH)
            self.recognizer = KaldiRecognizer(self.model, 16000)
            print("Modelo Vosk cargado exitosamente.")
            self.initialized = True
            return True
        except Exception as e:
            print(f"Error inicializando Vosk: {e}")
            return False

    def start_listening(self):
        """Inicia el stream de audio"""
        if not self.initialized:
            raise RuntimeError("Vosk no está inicializado. Llama a initialize() primero.")

        try:
            self.stream = sd.RawInputStream(
                samplerate=16000,
                blocksize=8000,
                dtype='int16',
                channels=1,
                callback=self._callback
            )
            self.stream.start()
            self.listening = True
            self.paused = False
            print("Escuchando continuamente con Vosk...")
            return True
        except Exception as e:
            print(f"Error iniciando stream de audio: {e}")
            return False

    def stop_listening(self):
        """Detiene el stream de audio"""
        if self.stream:
            self.stream.stop()
            self.stream.close()
        self.listening = False
        self.paused = False

    def pause_listening(self):
        """Pausa el procesamiento de audio sin detener el stream"""
        if self.listening:
            self.paused = True
            # Limpiar la cola de audio acumulado
            while not self.q.empty():
                try:
```

```
                    self.q.get_nowait()
                except:
                    break
            print("🔇 Escucha pausada durante procesamiento...")

    def resume_listening(self):
        """Reanuda el procesamiento de audio"""
        if self.listening:
            self.paused = False
            # Limpiar la cola antes de reanudar
            while not self.q.empty():
                try:
                    self.q.get_nowait()
                except:
                    break
            print("🎤 Escucha reanudada...")

    def listen_command(self):
        """Escucha un comando de voz"""
        if not self.initialized or not self.stream or self.paused:
            return None

        try:
            data = self.q.get_nowait()
            if self.recognizer.AcceptWaveform(data):
                result = json.loads(self.recognizer.Result())
                text = result.get("text", "").strip().lower()
                if text:
                    print(f"Detectado: {text}")
                    return text
        except queue.Empty:
            pass
        except Exception as e:
            print(f"Error en reconocimiento: {e}")

        return None

# Instancia global del reconocedor
recognizer_instance = VoskRecognizer()

def initialize_recognizer():
    """Inicializa el reconocedor de voz"""
    return recognizer_instance.initialize()

def start_listening():
    """Inicia la escucha de comandos"""
    return recognizer_instance.start_listening()

def stop_listening():
    """Detiene la escucha de comandos"""
    recognizer_instance.stop_listening()

def pause_listening():
    """Pausa la escucha de comandos"""
    recognizer_instance.pause_listening()

def resume_listening():
    """Reanuda la escucha de comandos"""
    recognizer_instance.resume_listening()

def listen_command():
    """Función de compatibilidad para escuchar comandos"""
    return recognizer_instance.listen_command()
```

Anexo 1.11 Código fuente - audio/speaker.py

```
import subprocess
import os
import requests
from pathlib import Path
from config import TTS_OUTPUT_FILE, OPENAI_API_KEY, OPENAI_TTS_VOICE


def speak(text):
    """Convierte texto a voz usando OpenAI TTS"""
    if not text or not text.strip():
        return

    text = text.strip()

    if not OPENAI_API_KEY:
        print(f"[Error] No hay clave API de OpenAI configurada. Texto: {text}")
        return

    _speak_with_openai(text)


def _speak_with_openai(text):
    """Usar OpenAI TTS para generar voz"""
    try:
        url = "https://api.openai.com/v1/audio/speech"
        headers = {
            "Authorization": f"Bearer {OPENAI_API_KEY}",
            "Content-Type": "application/json",
        }

        data = {
            "model": "tts-1",
            "input": text,
            "voice": OPENAI_TTS_VOICE,
            "response_format": "mp3"
        }

        response = requests.post(url, headers=headers, json=data, timeout=30)

        if response.status_code == 200:
            output_file = str(TTS_OUTPUT_FILE).replace('.wav', '.mp3')

            with open(output_file, 'wb') as f:
                f.write(response.content)

            # Reproducir el archivo de audio en Raspberry Pi
            try:
                subprocess.run(["mpg123", output_file], check=True)
            except FileNotFoundError:
                # Fallback a ffplay si mpg123 no está disponible
                try:
                    subprocess.run(["ffplay",   "-nodisp",   "-autoexit",   output_file],
check=True)
                except FileNotFoundError:
                    # Último fallback: convertir a wav y usar aplay
                    wav_file = output_file.replace('.mp3', '.wav')
                    subprocess.run(["ffmpeg", "-i", output_file, wav_file], check=True)
                    subprocess.run(["aplay", wav_file], check=True)
                    os.remove(wav_file)

            # Limpiar archivo temporal
            try:
                os.remove(output_file)
            except OSError:
                pass
        else:
            print(f"[OpenAI-TTS Error] Error {response.status_code}: {response.text}")

    except requests.exceptions.RequestException as e:
        print(f"[OpenAI-TTS Error] Error de conexión: {e}")
    except Exception as e:
        print(f"[OpenAI-TTS Error] Error inesperado: {e}")
```

Anexo 1.12 Código fuente - vision/camera.py

```
import time
import sys
import os
import subprocess
from pathlib import Path

# Agregar el directorio padre al path para poder importar config
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

try:
    from config import DEFAULT_IMAGE_FILENAME, CAMERA_ROTATION
except ImportError:
    # Fallback si no se puede importar config
    DEFAULT_IMAGE_FILENAME = Path.cwd().parent / "temp" / "captured_image.jpg"
    DEFAULT_IMAGE_FILENAME.parent.mkdir(exist_ok=True)
    CAMERA_ROTATION = 0

# Importar OpenCV solo para rotación de imágenes
try:
    import cv2
    CV2_AVAILABLE = True
except ImportError:
    CV2_AVAILABLE = False
    print("OpenCV no disponible - rotación de imágenes deshabilitada")

def take_picture(filename=None, rotation=None):
    """Toma una foto usando comandos del sistema de Raspberry Pi"""
    if filename is None:
        filename = DEFAULT_IMAGE_FILENAME

    if rotation is None:
        rotation = CAMERA_ROTATION

    filename = Path(filename)

    try:
        # Solo usar comandos del sistema para Raspberry Pi
        if _take_picture_system_command(filename):
            result_path = str(filename)

            # Aplicar rotación si es necesario
            if rotation != 0:
                print(f"Aplicando rotación de {rotation}°...")
                _rotate_image(result_path, rotation)

            return result_path
        else:
            print("Error: No se pudo tomar la foto con ningún comando disponible")
            return None

    except Exception as e:
        print(f"Error tomando foto: {e}")
        return None

def _take_picture_system_command(filename):
    """Usar comandos del sistema para tomar foto (más confiable en RPi)"""
    filename = Path(filename)
    filename.parent.mkdir(parents=True, exist_ok=True)

    commands_to_try = [
        # libcamera (Raspberry Pi OS Bullseye+)
        ["libcamera-still", "-o", str(filename), "-t", "1000", "--width", "1920", "--
height", "1080"],
        # raspistill (Raspberry Pi OS legacy)
        ["raspistill", "-o", str(filename), "-t", "1000", "-w", "1920", "-h", "1080"],
        # fswebcam (USB cameras)
        ["fswebcam", "-r", "1280x720", "--no-banner", str(filename)],
        # uvccapture (USB cameras alternative)
        ["uvccapture", "-o", str(filename), "-x", "1280", "-y", "720"]
    ]
```

```
    for cmd in commands_to_try:
        try:
            print(f"Intentando comando: {' '.join(cmd)}")
            result = subprocess.run(cmd, capture_output=True, text=True, timeout=10)

            if result.returncode == 0 and filename.exists() and filename.stat().st_size >
0:
                print(f"✅ Foto capturada con {cmd[0]}: {filename}")
                return True
            else:
                print(f"❌ {cmd[0]} falló: {result.stderr}")

        except subprocess.TimeoutExpired:
            print(f"❌ {cmd[0]} timeout")
        except FileNotFoundError:
            print(f"❌ {cmd[0]} no disponible")
        except Exception as e:
            print(f"❌ Error con {cmd[0]}: {e}")

    return False


def _rotate_image(image_path, rotation_degrees):
    """Rota una imagen según los grados especificados"""
    if not CV2_AVAILABLE or rotation_degrees == 0:
        return

    try:
        # Leer la imagen
        image = cv2.imread(str(image_path))
        if image is None:
            print(f"❌ No se pudo leer la imagen para rotación: {image_path}")
            return

        # Normalizar el ángulo de rotación
        rotation_degrees = rotation_degrees % 360

        if rotation_degrees == 0:
            return  # No hay necesidad de rotar

        # Obtener dimensiones de la imagen
        height, width = image.shape[:2]

        # Calcular el centro de rotación
        center = (width // 2, height // 2)

        # Crear la matriz de rotación
        rotation_matrix = cv2.getRotationMatrix2D(center, rotation_degrees, 1.0)

        # Calcular las nuevas dimensiones después de la rotación
        cos = abs(rotation_matrix[0, 0])
        sin = abs(rotation_matrix[0, 1])
        new_width = int((height * sin) + (width * cos))
        new_height = int((height * cos) + (width * sin))

        # Ajustar la matriz de rotación para centrar la imagen
        rotation_matrix[0, 2] += (new_width / 2) - center[0]
        rotation_matrix[1, 2] += (new_height / 2) - center[1]

        # Aplicar la rotación
        rotated_image = cv2.warpAffine(image, rotation_matrix, (new_width, new_height))

        # Guardar la imagen rotada
        success = cv2.imwrite(str(image_path), rotated_image)
        if success:
            print(f"✅ Imagen rotada {rotation_degrees}° correctamente")
        else:
            print(f"❌ Error guardando imagen rotada")

    except Exception as e:
        print(f"❌ Error rotando imagen: {e}")

def test_camera_fast():
```

```
    """Prueba rápida de cámara para Raspberry Pi"""
    print("=== PRUEBA DE CÁMARAS RASPBERRY PI ===")
    print(f"OpenCV disponible: {CV2_AVAILABLE}")
    print(f"Rotación configurada: {CAMERA_ROTATION}°")

    # Probar comandos del sistema
    print("\nProbando comandos del sistema...")
    temp_file = Path("/tmp/test_camera.jpg")
    if _take_picture_system_command(temp_file):
        print("✅ Comandos del sistema funcionan")
        if temp_file.exists():
            temp_file.unlink()
        return True
    else:
        print("❌ No se encontraron comandos de cámara funcionales")
        return False


if __name__ == "__main__":
    print("📹 PRUEBA DE CÁMARA RASPBERRY PI 📹\n")

    start_time = time.time()
    if test_camera_fast():
        print("\n=== TOMANDO FOTO DE PRUEBA ===")
        result = take_picture("test_photo.jpg")

        if result:
            print(f"✅ Foto guardada: {result}")
            if os.path.exists(result):
                size = os.path.getsize(result)
                print(f"Tamaño: {size} bytes")

                # La rotación se aplica automáticamente en take_picture()
                if CAMERA_ROTATION != 0:
                    print(f"✅ Rotación de {CAMERA_ROTATION}° aplicada")
        else:
            print("❌ Error tomando foto")
    else:
        print("❌ No se encontraron cámaras funcionales")

    total_time = time.time() - start_time
    print(f"\n⏱ Tiempo total: {total_time:.2f} segundos")
```

Anexo 1.13 Código fuente - vision/ocr.py

```
import base64
import requests
from config import OPENAI_API_KEY


def ocr_image(image_path):
    """Extrae texto de una imagen usando la API de OpenAI GPT-4 Vision"""
    if not OPENAI_API_KEY:
        return "Error: No se encontró la clave API de OpenAI."

    try:
        with open(image_path, "rb") as img:
            encoded = base64.b64encode(img.read()).decode("utf-8")

        url = "https://api.openai.com/v1/chat/completions"
        headers = {
            "Content-Type": "application/json",
            "Authorization": f"Bearer {OPENAI_API_KEY}"
        }

        data = {
            "model": "gpt-4.1-mini",
            "messages": [
                {
                    "role": "user",
                    "content": [
                        {
                            "type": "text",
                            "text": "Extrae todo el texto visible en esta imagen. Devuelve
únicamente el texto sin comentarios adicionales, manteniendo el formato y estructura
original cuando sea posible."
                        },
                        {
                            "type": "image_url",
                            "image_url": {
                                "url": f"{encoded}"
                            }
                        }
                    ]
                }
            ],
            "max_tokens": 500
        }


        response = requests.post(url, headers=headers, json=data, timeout=30)

        if response.status_code == 200:
            result = response.json()
            text = result["choices"][0]["message"]["content"]
            print(f"Texto extraído con OpenAI: {text}")
            return text.strip()
        else:
            print(f"Error en API OpenAI para OCR: {response.status_code}")
            print(f"Respuesta: {response.text}")
            return "Error al extraer texto con inteligencia artificial."

    except FileNotFoundError:
        return "Error: No se pudo encontrar el archivo de imagen."
    except requests.exceptions.RequestException as e:
        print(f"Error de conexión en OCR OpenAI: {e}")
        return "Error de conexión al servicio de extracción de texto."
    except Exception as e:
        print(f"Error inesperado en ocr_image_openai: {e}")
        return "Error inesperado al extraer texto de la imagen."
```

Anexo 1.14 Código fuente - utils/internet.py

```
import socket

def check_internet(host="8.8.8.8", port=53, timeout=3):
    try:
        socket.setdefaulttimeout(timeout)
        socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect((host, port))
        return True
    except Exception:
        return False
```

Anexo 1.15 Código fuente - validate_setup.py

```
#!/usr/bin/env python3
"""
Script de validación simple para la aplicación de IA Asistiva
Prueba la funcionalidad principal sin dependencias pesadas
"""
import sys
import os

def test_config():
    """Probar carga de configuración"""
    try:
        import config
        print("✅ Configuración cargada exitosamente")
        print(f"   - OCR OpenAI: {config.USE_OPENAI_OCR}")
        print(f"   - TTS OpenAI: {config.USE_OPENAI_TTS}")
        print(f"   - Voz TTS: {config.OPENAI_TTS_VOICE}")
        print(f"   - Clave API configurada: {'Sí' if config.OPENAI_API_KEY else 'No
(configura variable OPENAI_API_KEY)'}")
        return True
    except Exception as e:
        print(f"❌ Prueba de configuración falló: {e}")
        return False


def test_commands():
    """Probar carga de comandos"""
    try:
        import json
        if os.path.exists('commands.json'):
            with open('commands.json', 'r', encoding='utf-8') as f:
                commands = json.load(f)
            print("✅ Comandos cargados exitosamente")
            print(f"   - Comandos disponibles: {list(commands.keys())}")
            return True
        else:
            print("❌ commands.json no encontrado")
            return False
    except Exception as e:
        print(f"❌ Prueba de comandos falló: {e}")
        return False


def test_imports():
    """Probar importaciones críticas"""
    modules = [
        ('requests', 'Peticiones HTTP'),
        ('json', 'Manejo de JSON'),
        ('threading', 'Soporte de hilos'),
        ('pathlib', 'Manejo de rutas'),
        ('difflib', 'Coincidencias difusas'),
    ]

    all_good = True
    for module, description in modules:
        try:
            __import__(module)
            print(f"✅ {module} - {description}")
        except ImportError as e:
            print(f"❌ {module} - {description}: {e}")
            all_good = False

    return all_good


def test_openai_integration():
    """Probar integración con OpenAI (sin hacer llamadas a la API)"""
    try:
        import openai
        print("✅ SDK de OpenAI disponible")

        import config
        if config.OPENAI_API_KEY and config.OPENAI_API_KEY != "your_openai_api_key_here":
```

```
            print("✅ Clave API de OpenAI está configurada")
        else:
            print("⚠️  Clave API de OpenAI no configurada - funciones de OpenAI estarán
deshabilitadas")

        return True
    except ImportError:
        print("❌ SDK de OpenAI no instalado")
        return False


def test_directories():
    """Probar directorios requeridos"""
    directories = ['temp', 'audio', 'vision', 'utils']
    all_good = True

    for directory in directories:
        if os.path.exists(directory):
            print(f"✅ directorio {directory}/ existe")
        else:
            print(f"❌ directorio {directory}/ faltante")
            all_good = False

    return all_good


def main():
    """Ejecutar todas las pruebas"""
    print("🔍 Validando Configuración de la Aplicación de Voz Asistiva con IA")
    print("=" * 50)

    tests = [
        ("Configuración", test_config),
        ("Comandos", test_commands),
        ("Importaciones Principales", test_imports),
        ("Integración OpenAI", test_openai_integration),
        ("Estructura de Directorios", test_directories),
    ]

    passed = 0
    total = len(tests)

    for test_name, test_func in tests:
        print(f"\n📋 Probando {test_name}:")
        if test_func():
            passed += 1
        else:
            print(f"    ⚠️  Prueba de {test_name} tuvo problemas")

    print("\n" + "=" * 50)
    print(f"🎯 Resultados de Pruebas: {passed}/{total} pasaron")

    if passed == total:
        print("🎉  ¡Todas las pruebas pasaron! La aplicación debería funcionar
correctamente.")
        print("\n📌 Próximos pasos:")
        print("   1. Configura la variable de entorno OPENAI_API_KEY o crea archivo .env")
        print("   2. Ejecuta: python main.py")
    else:
        print("⚠️  Algunas pruebas fallaron. Por favor revisa los problemas anteriores.")

    return passed == total

if __name__ == "__main__":
    success = main()
    sys.exit(0 if success else 1)
```

Anexo 1.16 Dependencias - requirements.txt

```
vosk
sounddevice
opencv-python
pytesseract
requests
python-dotenv
TTS
difflib
```

Anexo 1.17 Script de configuración - setup.sh

```
#!/bin/bash

# Script de configuración para la Aplicación de Voz Asistiva con IA
echo "🤖 Configurando Aplicación de Voz Asistiva con IA..."

# Verificar si Python 3 está instalado
if ! command -v python3 &> /dev/null; then
    echo "❌ Python 3 es requerido pero no está instalado. Por favor instala Python 3.8+
primero."
    exit 1
fi

echo "✅ Python 3 encontrado: $(python3 --version)"

# Crear entorno virtual si no existe
if [ ! -d "venv" ]; then
    echo "🔧 Creando entorno virtual..."
    python3 -m venv venv
fi

# Activar entorno virtual
echo "🔧 Activando entorno virtual..."
source venv/bin/activate

# Actualizar pip
echo "🔧 Actualizando pip..."
pip install --upgrade pip

# Instalar dependencias
echo "📦 Instalando dependencias de Python..."
pip install -r requirements.txt

# Crear directorio temporal
echo "📁 Creando directorio temporal..."
mkdir -p temp

# Crear commands.json si no existe
if [ ! -f "commands.json" ]; then
    echo "📝 Creando commands.json predeterminado..."
    cat > commands.json << 'EOF'
{
    "read_document": [
        "leer documento",
        "quiero que leas",
        "puedes leer esto",
        "lee esto",
        "lee el documento",
        "leer texto"
    ],
    "describe_scene": [
        "describir escena",
        "qué ves",
        "descríbeme el lugar",
        "describe la imagen",
        "qué hay aquí",
        "dime qué ves"
    ],
    "exit": [
        "salir",
        "terminar",
        "adiós",
        "bye",
        "cerrar"
    ]
}
EOF
fi
```

```
# Verificar dependencias del sistema
echo "🔍 Verificando dependencias del sistema..."

# Verificar Tesseract OCR
if ! command -v tesseract &> /dev/null; then
    echo "⚠️  Tesseract OCR no encontrado. Instalando..."
    if [[ "$OSTYPE" == "darwin"* ]]; then
        # macOS
        if command -v brew &> /dev/null; then
            brew install tesseract tesseract-lang-spa
        else
            echo "❌ Homebrew no encontrado. Por favor instala Tesseract manualmente o
instala Homebrew primero."
        fi
    elif [[ "$OSTYPE" == "linux-gnu"* ]]; then
        # Linux
        sudo apt-get update
        sudo apt-get install -y tesseract-ocr tesseract-ocr-spa
    fi
else
    echo "✅ Tesseract OCR encontrado"
fi

# Verificar herramientas de audio
if [[ "$OSTYPE" == "linux-gnu"* ]]; then
    # Verificar espeak (TTS de Linux)
    if ! command -v espeak &> /dev/null; then
        echo "⚠️  espeak no encontrado. Instalando..."
        sudo apt-get install -y espeak
    else
        echo "✅ espeak encontrado"
    fi

    # Verificar aplay (reproductor de audio de Linux)
    if ! command -v aplay &> /dev/null; then
        echo "⚠️  aplay no encontrado. Instalando utilidades ALSA..."
        sudo apt-get install -y alsa-utils
    else
        echo "✅ aplay encontrado"
    fi

    # Opcional: mpg123 para reproducción MP3 (OpenAI TTS)
    if ! command -v mpg123 &> /dev/null; then
        echo "⚠️  mpg123 no encontrado. Instalando para soporte de OpenAI TTS..."
        sudo apt-get install -y mpg123
    else
        echo "✅ mpg123 encontrado"
    fi
fi

# Crear archivo .env.example
if [ ! -f ".env.example" ]; then
    echo "📝 Creando archivo .env.example..."
    cat > .env.example << 'EOF'
# Clave API de OpenAI para funciones mejoradas
OPENAI_API_KEY=your_openai_api_key_here

# Configurar qué herramientas usar
USE_OPENAI_OCR=true
USE_OPENAI_TTS=true

# Voz de OpenAI TTS (alloy, echo, fable, onyx, nova, shimmer)
OPENAI_TTS_VOICE=alloy
EOF
fi

echo ""
echo "🎉 ¡Configuración completa!"
```

```
echo ""
echo "📋 Próximos pasos:"
echo "1. Copia .env.example a .env y agrega tu clave API de OpenAI (opcional pero recomendado)"
echo "2. Ejecuta la aplicación con: python3 main.py"
echo ""
echo "ℹ️  Configuración:"
echo "   - Edita config.py para cambiar preferencias de OCR/TTS"
echo "   - Edita commands.json para agregar comandos de voz personalizados"
echo "   - Revisa README.md para instrucciones detalladas de uso"
echo ""
echo "🚀 Listo para ejecutar: python3 main.py"
```

Anexo 1.18 Configuración de comandos - commands.json

```
{
    "read_document": [
        "leer documento",
        "quiero que leas",
        "puedes leer esto",
        "lee esto",
        "lee el documento",
        "leer texto"
    ],
    "exit": [
        "salir",
        "terminar",
        "adiós",
        "bye",
        "cerrar"
    ]
}
```