DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 14 No. Especial (2026) 149-156

Implementación de un sistema portátil de traducción del alfabeto dactilológico mexicano en Raspberry Pi
Development of a Portable System for Translating the Mexican Sign Language Alphabet on a Raspberry Pi

V. Becerra-Tapia [a], G. R. Peñaloza-Mendoza [b*], M. S. Castro-Zenil [b], N. V. Vázquez-Belmonte [a]

[a] *CIIDT de Ingeniería Biomédica, TecNM - Instituto Tecnológico Superior de Pátzcuaro, 61615, Pátzcuaro, Michoacán, México.*
[b] *CA-IIDT Departamento de Ingeniería Biomédica, TecNM - Instituto Tecnológico Superior de Pátzcuaro, 61615, Pátzcuaro, Michoacán, México.*

Resumen

De acuerdo a datos del Instituto Nacional de Estadística y Geografía (INEGI) en México, más de 2.8 millones de personas presentan algún grado de discapacidad auditiva, y aproximadamente el 60% de ellas enfrenta barreras para comunicarse con quienes no dominan o desconocen la Lengua de Señas Mexicana (LSM). En respuesta a esto se presenta un prototipo portátil basado en una Raspberry Pi y tecnologías de visión artificial, capaz de detectar y traducir en de forma inmediata gesticulaciones correspondientes al alfabeto dactilológico de la LSM. El sistema despliega en pantalla una letra de forma aleatoria, espera que el usuario realice la seña correspondiente frente a la cámara y a partir de las imágenes capturadas, se aplican algoritmos de detección de la posición y movimiento de los dedos para verificar la correspondencia entre la gesticulación y la letra mostrada. Esta propuesta, de bajo costo y escalable, busca ofrecer una herramienta tecnológica con potencial educativo que facilite el aprendizaje del alfabeto en lengua de señas. Aunque aún no se ha implementado en entornos escolares, los resultados preliminares muestran su viabilidad como recurso de apoyo para procesos de enseñanza.

Palabras Clave: Alfabeto dactilológico, Inclusión, Lenguaje de señas, Raspberry Pi.

Abstract

According to data from the National Institute of Statistics and Geography (INEGI) in Mexico, more than 2.8 million people have some degree of hearing impairment, and approximately 60% of them encounter communication barriers when interacting with individuals who are not proficient in or unfamiliar with Mexican Sign Language (MSL). In response, this work presents a portable prototype based on Raspberry Pi and computer vision technologies, capable of real-time detection and translation of hand gestures corresponding to the MSL fingerspelling alphabet. The system randomly displays a letter on the screen and prompts the user to perform the corresponding sign in front of the camera. Using the captured images, algorithms are applied to analyze finger position and movement in order to verify the correspondence between the gesture and the displayed letter. This low-cost and scalable proposal aims to provide a technological tool with educational potential to facilitate the learning of the sign language alphabet. Although not yet implemented in educational environments, preliminary results demonstrate its feasibility as a supportive resource for teaching and learning processes.

Keywords: Fingerprinting, Inclusion, Sign Language, Raspberry Pi.

1. Introducción

En el marco de una sociedad incluyente, la comunicación constituye un derecho humano fundamental sin distinción de capacidades físicas o sensoriales, reconocido por organismos internacionales como la Organización de las Naciones Unidas (ONU, 2006). En México, de acuerdo con la Encuesta Nacional de la Dinámica Demográfica 2023 (ENADID), existen 8.8 millones de personas mayores de cinco años que declararon tener alguna discapacidad, lo que representa el 7.2% de la población total (INEGI, 2024). De ellas, aproximadamente el 2.4% presenta algún grado de discapacidad auditiva (INEGI, 2020), y se estima que alrededor del 60% enfrenta dificultades para comunicarse con

*Autor para la correspondencia: grey@itspa.edu.mx
Correo electrónico: vikshelby.500@gmail.com (Víctor Becerra-Tapia), grey@itspa.edu.mx (Guillermo Rey Peñaloza-Mendoza), mcastro@itspa.edu.mx (Mario Salvador Castro-Zenil), ciidtbiomedica@itspa.edu.mx (Netzha Valentino Vázquez-Belmonte).

Historial del manuscrito: recibido el 29/06/2026, última versión-revisada recibida el 16/08/2026, aceptado el 08/02/2026, publicado el 10/03/2026. **DOI:** https://doi.org/10.29057/icbi.v14iEspecial.15390

individuos que desconocen la Lengua de Señas Mexicana (González, 2021). Esta situación refleja una problemática crítica, la comunicación es un pilar fundamental para la integración de las personas en la sociedad; sin embargo, las personas con discapacidad auditiva enfrentan barreras significativas en la interacción con su entorno, especialmente en contextos donde no se cuenta con intérpretes o personas familiarizadas con la Lengua de Señas Mexicana. Esta limitación no solo afecta su vida social, sino que también restringe sus oportunidades educativas y profesionales.

En los últimos años, el desarrollo tecnológico orientado al reconocimiento de la Lengua de Señas ha avanzado de forma notable, explorando distintas metodologías que van desde soluciones basadas en sensores hasta sistemas de visión por computadora. Por ejemplo, BrightSign es un guante que incorpora múltiples sensores para reconocer y registrar los movimientos individuales de la mano, expresando el significado de los gestos en forma de texto o voz (BrightSign, 2022). De manera similar, CyberGlove es un sistema de captura de movimiento de manos que, aunque no fue concebido originalmente como traductor de Lengua de Señas, utiliza sensores de flexión en cada articulación de los dedos, permitiendo su aplicación en la interpretación de gestos manuales (CyberGlove, 2015). Estas propuestas demuestran la eficacia de los sistemas basados en sensores, aunque presentan desventajas como el costo elevado, la necesidad de calibración y el carácter intrusivo del uso de guantes.

Paralelamente, los avances en visión artificial han impulsado el desarrollo de soluciones más versátiles y no intrusivas. En este ámbito, Wang, Chen y Li (2020) demostraron la viabilidad del uso de modelos de aprendizaje profundo para el reconocimiento de la Lengua de Señas en tiempo real. Asimismo, bibliotecas como MediaPipe Hands han facilitado la identificación de los 21 puntos clave de la mano con bajo consumo computacional, favoreciendo el desarrollo de aplicaciones portátiles y de respuesta inmediata (Zhang et al., 2020).

Aunado a ello, investigaciones recientes han evidenciado que los modelos basados en redes neuronales convolucionales tridimensionales (CNN 3D) y Transformers alcanzan altos niveles de precisión en vocabularios extensos, permitiendo la transición de sistemas experimentales hacia aplicaciones prácticas orientadas a la inclusión (Camgoz et al., 2020; Li et al., 2023). Dichos avances han sido posibles gracias a la disponibilidad de conjuntos de datos de gran escala, como el WLASL (Word-Level American Sign Language), que cuenta con más de dos mil signos y múltiples participantes, constituyéndose en un referente para comparar arquitecturas y evaluar la capacidad de generalización de los modelos (Li, 2020).

En la literatura especializada, las soluciones para el reconocimiento de señas se agrupan principalmente en dos enfoques. El primero, basado en sensores físicos, como los presentes en BrightSign y CyberGlove, logra alta precisión en la detección de movimientos, pero a costa de ser intrusivo y costoso (BrightSign, 2022; CyberGlove, 2015). El segundo enfoque, centrado en la visión por computadora, abarca desde modelos ligeros hasta arquitecturas profundas (CNN 3D y Transformers) con gran exactitud, aunque con un consumo computacional considerable que dificulta su implementación en dispositivos portátiles (Wang et al., 2020; Camgoz et al., 2020; Li et al., 2023).

Considerando estas limitaciones, resulta necesario desarrollar un enfoque no intrusivo, eficiente y de bajo costo, capaz de operar en hardware embebido. En este contexto, la Raspberry Pi se presenta como una alternativa idónea por su accesibilidad, bajo consumo energético y ecosistema maduro de programación en Python (Upton & Halfacree, 2014; Muhammad et al., 2022). Además, bibliotecas como OpenCV (Bradski, 2008) y MediaPipe (Lugaresi et al., 2019; Zhang et al., 2020) ofrecen herramientas que permiten implementar detección de manos, segmentación de dedos y análisis de movimiento en tiempo real, logrando un equilibrio entre precisión y eficiencia computacional.

En conjunto, estos desarrollos sustentan la pertinencia de diseñar dispositivos portátiles de traducción del alfabeto dactilológico, que integren hardware accesible con algoritmos de visión artificial optimizados, orientados al fortalecimiento de la inclusión y la educación en comunidades con discapacidad auditiva.

A pesar de estos avances, aún no existen dispositivos portátiles, accesibles y confiables que permitan traducir el alfabeto dactilológico mexicano de manera inmediata. Las soluciones actuales suelen depender de modelos de inteligencia artificial que requieren altos recursos computacionales, así como de bases de datos limitadas o hardware especializado, lo cual restringe su aplicación práctica en contextos educativos y sociales donde se busca fomentar la inclusión de personas con discapacidad auditiva.

Ante esta problemática, el presente trabajo propone el diseño e implementación de un prototipo autónomo basado en Raspberry Pi 3, que integra algoritmos geométricos para el reconocimiento de gestos manuales sin depender de modelos de aprendizaje profundo. Esta solución busca ofrecer una alternativa tecnológica asequible, eficiente y de fácil implementación, que contribuya al aprendizaje y la práctica del alfabeto dactilológico mexicano, promoviendo la inclusión y la accesibilidad comunicativa en entornos educativos. Asimismo, se valida experimentalmente con usuarios no expertos y sin discapacidad auditiva, alcanzando niveles de precisión entre 85% y 93%. Estos resultados confirman la viabilidad del sistema como una herramienta educativa y de inclusión social.

2. Metodología

2.1. Selección de hardware y configuración del entorno

La selección de la tarjeta de desarrollo se realizó con base en los requerimientos específicos del sistema, priorizando la compatibilidad con bibliotecas de visión artificial como OpenCV y MediaPipe, el soporte para Python y la capacidad de procesamiento necesaria para el análisis de imágenes en tiempo real. Se optó por la plataforma Raspberry Pi, evaluando distintos modelos considerando aspectos como la memoria

RAM, el rendimiento del procesador y el consumo energético, dado que el dispositivo debe operar de forma autónoma con alimentación por batería, esto se puede observar en la Tabla 1, donde se comparan 4 diferentes plataformas para su implementación. La elección de Raspberry Pi 3 prioriza bajo costo, consumo y ecosistema Python/OpenCV; el algoritmo propuesto no depende de inferencia DL pesada, por lo que plataformas con GPU/TPU no son necesarias. Si se escalara a palabras o modelos temporales profundos, Jetson/Coral serían candidatos viables de plataformas que se emplearían.

Para la captura de imágenes, se seleccionó una Cámara Raspberry Pi (RaspiCam) considerando su resolución de 5 MP y su capacidad de operar a diferentes tasas de fotogramas, la cual se conecta por CSI (Camera Serial Interface) tal como se observa en la Figura 1. Estos parámetros son críticos para asegurar una detección fluida y precisa de gestos manuales, ya que una alta resolución o una tasa de cuadros inadecuada podría afectar negativamente el rendimiento en tiempo real del sistema.

Un criterio clave en la selección del hardware fue el consumo de energía debido a la naturaleza portátil del prototipo. Se estimó el consumo combinado de una Raspberry Pi 3, la cámara RaspiCam y una pantalla HDMI de 7 pulgadas entre 8.5 a 10.5 watts, además del uso de periféricos como un teclado inalámbrico. Esto permitió seleccionar una batería adecuada que garantizara autonomía operativa durante las sesiones de uso.

Tabla 1: Comparación de plataformas tecnológicas

Plataforma	**CPU/ GPU**	**Consumo / Costo Costo**		**Adecuación al enfoque**
Raspberry Pi 3	CPU ARM	Bajo	Bajo	**Óptima** para reglas geométricas en tiempo real
Raspberry Pi 4	CPU más rápida	Medio	Medio	Si se requiere más FPS
NVIDIA Jetson Nano	CPU + GPU	Medio	Medio–Alto	Para DL en el borde
Google Coral Dev Board	CPU + Edge TPU	Bajo–Medio	Medio–Alto	DL eficiente

En cuanto al entorno de software, se llevó a cabo la instalación y configuración de OpenCV y MediaPipe en la Raspberry Pi. Esta etapa incluyó la preparación del entorno Python y la verificación de la compatibilidad de las versiones de las bibliotecas con el sistema operativo. La configuración se orientó a garantizar el procesamiento eficiente de imágenes y la detección de manos en tiempo real, utilizando los puntos clave generados por MediaPipe como base para el análisis de posiciones y gestos de los dedos.

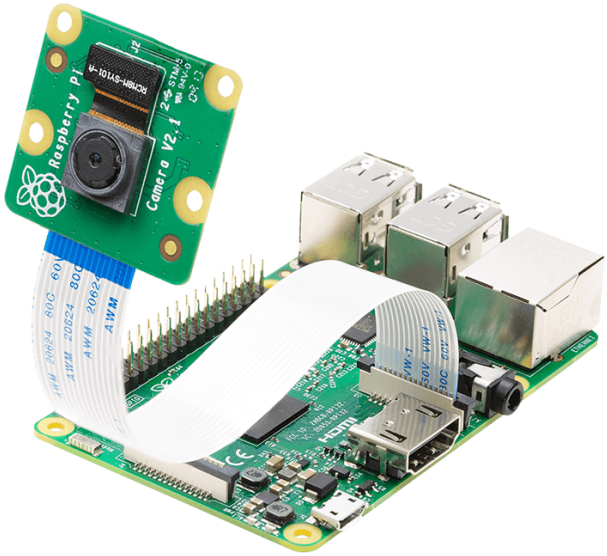

Figura 1: Puerto de conexión para RaspiCam.

Previo a la instalación de las bibliotecas requeridas, se actualiza el sistema operativo de la Raspberry Pi 3 para garantizar compatibilidad y estabilidad. Este proceso se realiza mediante los siguientes comandos en la terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

Posteriormente, se instalan las dependencias necesarias para compilar y ejecutar bibliotecas de visión artificial. Estas incluyen herramientas de desarrollo, bibliotecas de OpenCV y otros componentes esenciales:

```
sudo apt-get install -y build-essential cmake
sudo apt-get install -y libopencv-dev libopencv-highgui-dev libopencv-core-dev libopencv-imgproc-dev
sudo apt-get install -y libatlas-base-dev gfortran
sudo apt-get install -y python3-dev python3-pip
```

La biblioteca *NumPy*, fundamental para operaciones matemáticas en Python, se instala mediante:

```
pip3 install numpy
```

Para asegurar una instalación actualizada y funcional de OpenCV, se emplea *pip3*:

```
pip3 install opencv-python
```

Asimismo, se instala MediaPipe, biblioteca desarrollada por Google que proporciona herramientas para la detección y seguimiento de poses, manos y rostros:

```
pip3 install mediapipe
```

Finalmente, se realizan pruebas básicas para verificar la correcta instalación de las bibliotecas. Esto se hace ejecutando los siguientes comandos en la terminal:

```
# Verificación de OpenCV
python3 -c "import cv2; print(cv2.__version__)"
```

```
# Verificación de MediaPipe
python3 -c "import mediapipe as mp; print(mp.__version__)"
```

La correcta ejecución de estos comandos sin errores confirma que el entorno de desarrollo está debidamente configurado y listo para el procesamiento de imágenes en tiempo real en la Raspberry Pi 3.

2.2. Algoritmo para la detección de señas

Posteriormente, se debe implementar el algoritmo para la detección de manos y reconocimiento de letras del alfabeto dactilológico en tiempo real utilizando MediaPipe y OpenCV sobre la Raspberry Pi 3, este se muestra en la Figura 2.

a) *Importación de bibliotecas:* Se importan las bibliotecas necesarias para el funcionamiento del sistema: *cv2* (OpenCV) para la captura y visualización de imágenes. mediapipe para la detección y seguimiento de manos, además de funciones auxiliares.

b) *Configuración de la cámara:* Se inicializa la cámara utilizando *cv2.VideoCapture()*, definiendo parámetros como el ancho (wCam) y alto (hCam) del fotograma para optimizar el rendimiento en tiempo real.

c) *Inicialización del modelo de MediaPipe Hands:* Se instancia el módulo *mp_hands.Hands* con los siguientes parámetros:

static_image_mode: False (detección en secuencia continua).
max_num_hands: número máximo de manos a detectar (usualmente 1).
min_detection_confidence: umbral mínimo de confianza para la detección (ejemplo 0.7).

d) *Captura y procesamiento de fotogramas:* Se inicia un bucle principal en el que se capturan los fotogramas en tiempo real desde la cámara. Cada imagen es convertida de BGR a RGB y procesada mediante *hands.process()* para detectar la(s) mano(s).

Las imágenes capturadas por la cámara se representan inicialmente en formato BGR (Blue–Green–Red), el cual es estándar en la biblioteca OpenCV. Para su procesamiento, se convierten al formato RGB (Red–Green–Blue), ya que este es requerido por los modelos de MediaPipe para la detección de manos.

e) *Extracción de puntos de referencia:* Una vez detectada la mano, MediaPipe devuelve un conjunto de 21 puntos de referencia (landmarks), cada uno definido por coordenadas normalizadas respecto al marco de referencia de la imagen (ejes x, y en el plano de la imagen y un eje z relativo a la profundidad de la palma). Estas coordenadas no tienen unidades físicas absolutas, sino que son proporcionales al tamaño de la imagen capturada, lo que permite generalizar la detección en diferentes resoluciones.

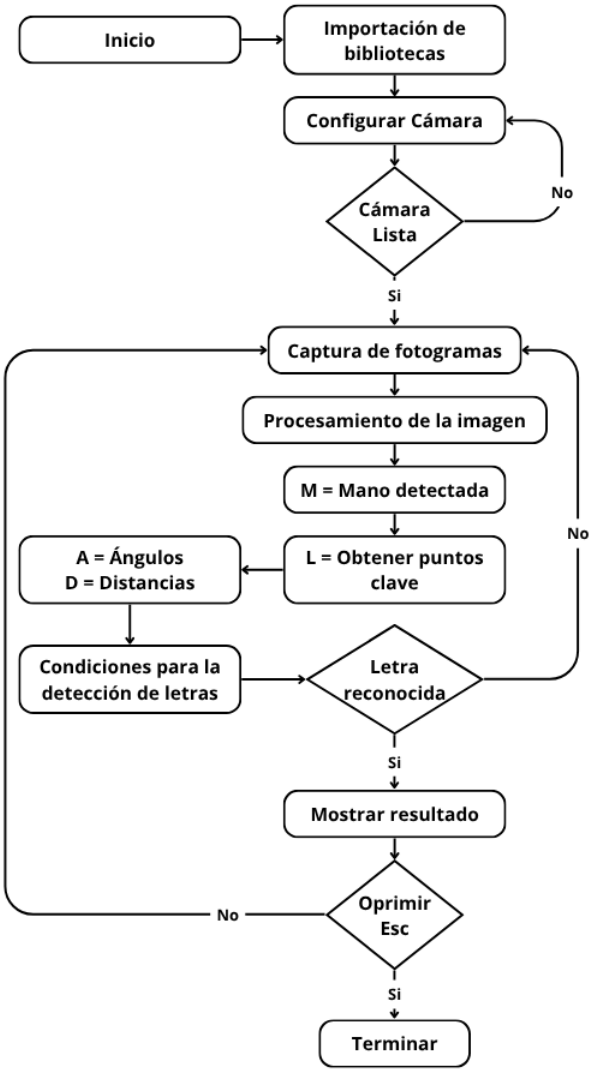

Figura 2: Algoritmo de detección.

f) *Extracción de puntos de referencia:* Una vez detectada la mano, MediaPipe devuelve un conjunto de 21 puntos de referencia (landmarks), cada uno definido por coordenadas normalizadas respecto al marco de referencia de la imagen (ejes x, y en el plano de la imagen y un eje z relativo a la profundidad de la palma). Estas coordenadas no tienen unidades físicas absolutas, sino que son proporcionales al tamaño de la imagen capturada, lo que permite generalizar la detección en diferentes resoluciones.

g) *Cálculo de ángulos y estado de los dedos:* Para calcular el estado de los dedos, se emplean relaciones geométricas basadas en distancias y ángulos. Para encontrar la distancia entre dos puntos (landmarks) mostrados en la Figura 3, $P_i(x_i, y_i, z_i)$ que representa el punto de referencia de una articulación y $P_j(x_j, y_j, z_j)$ que representa el punto de análisis en otra articulación, se emplea la ecuación (1). Esto permite verificar, por ejemplo, si la punta de un dedo está más cerca de la palma que de otra articulación.

$$d_{ji} = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2 + \left(z_j - z_i\right)^2} \qquad (1)$$

Con estas distancias se determinan vectores entre articulaciones. Para determinar la extensión o dobles del dedo, se calcula el ángulo entre los puntos de dos marcas, usando un punto de referencia $P_k(x_k, y_k, z_k)$ y el producto punto. Si se define $\vec{u} = P_j - P_i$ y $\vec{v} = P_k - P_i$, el ángulo θ puede calcularse por medio de la ecuación (2) usando el producto punto.

$$cos(\theta) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|} \quad (2)$$

Se aplican funciones personalizadas que analizan las posiciones relativas de los puntos clave para calcular ángulos o distancias entre articulaciones. Con base en estos cálculos, se determina el estado (extendido o doblado) de cada dedo. Si el ángulo entre la falange proximal y la falange intermedia de un dedo es cercano a 180°, el dedo está extendido. Si el ángulo es cercano a 0°–45°, el dedo está doblado. De esta forma cada dedo se representa como un valor binario *estado_dedo* (doblado = 0, extendido = 1).

Posteriormente, la función *condicionalesLetras* asocia patrones de dedos extendidos/doblados con cada letra. Este mapeo no es una correlación estadística, sino un algoritmo determinista de reglas geométricas programadas manualmente.

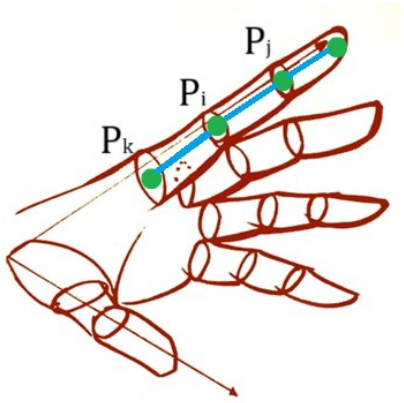

Figura 3: Representación de marcas (landmarks) en la mano.

h) *Reconocimiento de letras:* Se implementa lógica condicional basada en combinaciones específicas del estado de los dedos, para identificar la letra correspondiente del alfabeto dactilológico. Esta lógica está programada manualmente según patrones definidos. Dentro de la función condicionalesLetras, se define una serie de estructuras condicionales (if) que comparan patrones específicos de la lista dedos, asociados a configuraciones particulares de los dedos. Cada patrón condicional representa una letra distinta del alfabeto. Esta lógica manual permite mapear combinaciones de posturas digitales a letras concretas, habilitando una forma básica de traducción de lenguaje de señas a texto visualizado en tiempo real.

i) *Visualización de resultados:* Los resultados del reconocimiento se superponen sobre el fotograma mediante funciones de OpenCV, y se visualizan en una ventana utilizando *cv2.imshow()*.
Al coincidir la seña con una letra, se ejecuta una acción correspondiente: se superpone la letra detectada en el fotograma mediante OpenCV y, adicionalmente, se imprime su identificación en la consola del sistema.

j) Finalización del programa: El bucle de procesamiento se interrumpe al presionar una tecla específica (por ejemplo, Esc). Posteriormente, se liberan los recursos de la cámara y se cierran todas las ventanas activas con *cv2.destroyAllWindows()*.

2.3. Sistema físico

La arquitectura física del prototipo puede representarse como un diagrama de bloques (Figura 4), donde se distinguen sus principales componentes:

- Módulo de adquisición (Cámara RaspiCam): captura en tiempo real imágenes de la mano del usuario.
- Módulo de procesamiento (Raspberry Pi 3): ejecuta los algoritmos de visión artificial con OpenCV y MediaPipe.
- Módulo de visualización (Pantalla HDMI 7”): despliega el video en tiempo real con la letra reconocida superpuesta.
- Módulo de interacción (teclado inalámbrico): permite controlar el sistema.
- Módulo de alimentación (batería 5V/3000 mA): asegura portabilidad y autonomía por casi 5 horas.

Cada módulo cumple un papel específico, y su integración garantiza el funcionamiento autónomo del sistema sin necesidad de conexión a una computadora externa.

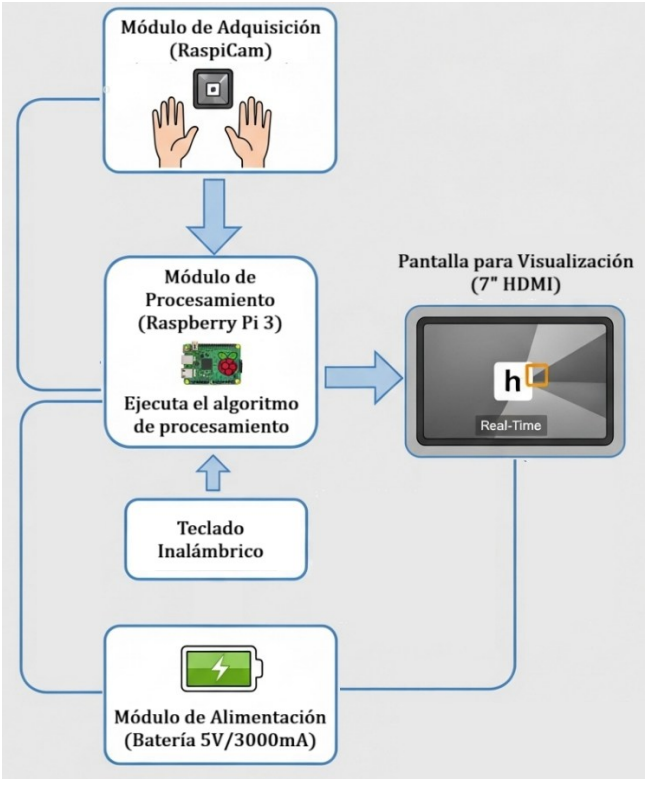

Figura 4: Diagrama de bloques de la conexión del sistema físico.

Durante las pruebas experimentales, se establecieron las conexiones necesarias entre la Raspberry Pi 3 y sus periféricos principales: la cámara, la pantalla y los dispositivos de entrada. Esta etapa es crucial para asegurar el correcto funcionamiento del prototipo de reconocimiento de señas.

La cámara RaspiCam se conectó al puerto CSI (Camera Serial Interface) de la Raspberry Pi, lo cual permitió una comunicación de alta velocidad y baja latencia con el procesador gráfico del dispositivo. Esto garantiza una captura eficiente de imágenes en tiempo real, elemento clave para el procesamiento mediante visión artificial. La cámara obtiene su alimentación directamente a través del mismo conector CSI, lo que simplifica el diseño del hardware y reduce el cableado adicional.

La pantalla se conecta a la Raspberry Pi mediante el puerto HDMI. Para la funcionalidad táctil se añadió una conexión USB desde la pantalla a uno de los puertos USB disponibles en la placa, habilitando la interacción directa con la interfaz del sistema. Esta pantalla cumple una doble función: permite visualizar en tiempo real el video con las detecciones sobrepuestas y facilita la interacción del usuario con la interfaz gráfica, en caso de ser necesaria.

Para la alimentación del sistema, se utilizó una batería externa recargable de 5V y 3000mA, capaz de suministrar energía tanto a la Raspberry Pi como a los periféricos conectados. Esta configuración asegura portabilidad, independencia de fuentes fijas de energía y estabilidad durante la operación continua.

Finalmente, se incorporó un teclado inalámbrico con touchpad para facilitar el control del sistema sin necesidad de dispositivos adicionales

El ensamblaje completo fue diseñado buscando una disposición compacta y funcional, permitiendo que todos los componentes operen de manera coordinada. El sistema resultante es completamente autónomo, capaz de capturar imágenes, procesarlas en tiempo real, reconocer letras del alfabeto dactilológico y desplegar resultados sin necesidad de intervención externa o conexión a una computadora adicional.

3. Resultados

Durante el desarrollo del dispositivo portátil para la traducción del alfabeto dactilológico, se logró integrar de manera efectiva los módulos de hardware y software sobre la plataforma Raspberry Pi 3 alimentada por batería, confirmando su funcionamiento autónomo y portátil. La conexión física del ensamblaje se muestra en la Figura 5, mientras que en la Figura 6 se muestra el sistema que incluye la cámara RaspiCam, una pantalla de 7” y un teclado con touchpad inalámbrico. La alimentación con una batería de 5V y 3000 mA demostró proporcionar autonomía suficiente para pruebas continuas durante 4 horas y 56 minutos en promedio, sin interrupciones ni inestabilidad.

Figura 5: Conexión física de los elementos del sistema.

Figura 6: Sistema de reconocimiento.

La integración de OpenCV y MediaPipe permitió una detección precisa de los gestos de la mano. El algoritmo, desarrollado en Python, interpreta la posición de los dedos mediante análisis geométrico (ángulos y distancias) y traduce el gesto a una letra del alfabeto con retroalimentación inmediata en pantalla, se realizaron pruebas con todas las letras del alfabeto, en la Figura 7 se puede observar el reconocimiento de la letra A a partir de su seña, en la Figura 8 la letra B y en la Figura 9 la letra I.

Figura 7: Reconocimiento de la letra A.

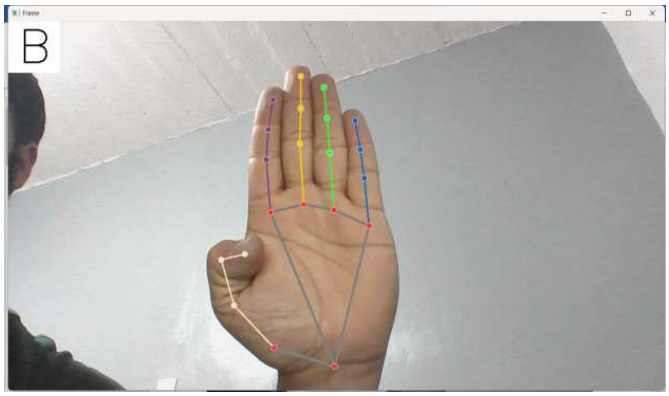

Figura 8: Reconocimiento de la letra B.

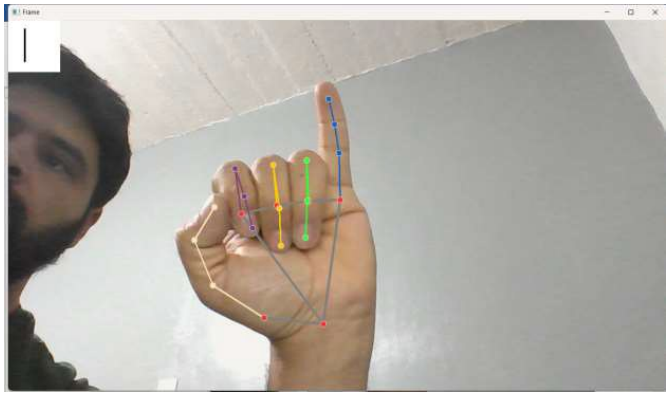

Figura 9: Reconocimiento de la letra I.

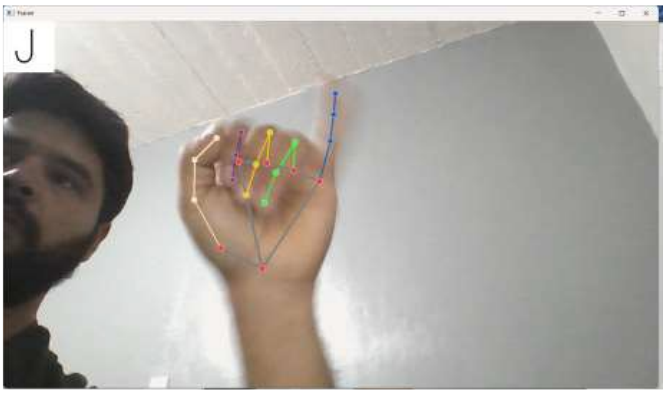

Figura 10: Detección de la letra J.

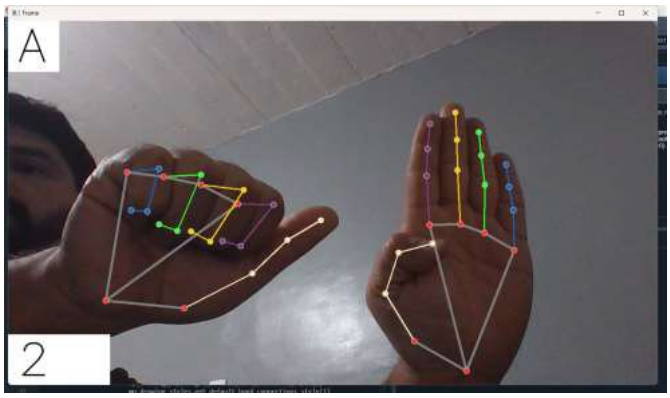

Figura 11: Detección de las dos manos.

Uno de los logros más destacados fue la correcta detección de letras que implican movimiento, tal como la letra "J", cuya prueba se muestra en la Figura 10, esta función es vital en la traducción de la Lengua de Señas debido a que algunas letras del alfabeto dactilológico requieren movimiento, sin embargo, la gran mayoría de las señas que se refieren a palabras requieren la detección del movimiento.

Adicionalmente, se añadió la funcionalidad para detectar la presencia de dos manos, esto no difiere del algoritmo principal, sin embargo, se agrega una variable adicional. Esta función permite almacenar la traducción de una letra cuando aparece una segunda mano, esto es, únicamente traduce la primera mano detectada y al aparecer la segunda mano, la seña detectada será almacenada, facilitando así la escritura de palabras o frases completas. Muestra del reconocimiento de la segunda mano se presenta en la Figura 11, en esta figura se pueden observar dos cuadros donde se visualiza la letra A y un número 2, esto indica que la primera mano reconocida fue la derecha que hace la letra A, posteriormente se agrega la segunda mano y el sistema muestra un número 2, esto hace que el sistema guarde la letra A detectada y vaya almacenando los símbolos para formar una palabra.

Para evaluar la efectividad del sistema, se realizaron pruebas controladas con tres participantes, cada uno mostrando las 27 letras del alfabeto con variaciones mínimas de posición y orientación. En la Tabla 2 se muestran los resultados obtenidos de esas pruebas, esta refleja una precisión media entre 85% y 93%, lo cual respalda el funcionamiento confiable del prototipo.

Tabla 2: Pruebas de validación

	Persona 1	Persona 2	Persona 3
Total de letras evaluadas	27	27	27
Letras detectadas correctamente	24	25	23
Letras no detectadas o mal detectadas	3	2	4
Letras con movimiento correctas	2	2	1
Precisión (%)	88.9	92.6	85.2

Dividir las letras del alfabeto en grupos según su complejidad permite mostrar la robustez del sistema frente a distintos retos del lenguaje de señas, en la Tabla 3 se muestran los resultados de esta evaluación donde el sistema tiene mejor rendimiento en letras estáticas, pero logra un buen desempeño en letras con movimiento, lo que muestra su versatilidad.

Tabla 3: Resultados por tipo de letra realizada

Grupo de letras	Letras incluidas	Precisión promedio (%)
Letras simples estáticas	A, B, C, D, E, F, G, H, entre otras	95.5%
Letras con formas similares	M, N, U, V	86.0%
Letras con movimiento	J, Z	83.3%

Los resultados obtenidos muestran que el sistema presenta un rendimiento alto en letras estáticas (95.5%) frente a letras con movimiento (83.3%). Esta diferencia se debe a la complejidad inherente de las señas dinámicas, donde la

trayectoria y la velocidad del movimiento influyen en la detección. Sin embargo, el desempeño logrado en letras con movimiento resulta competitivo frente a sistemas basados en modelos de aprendizaje profundo, que si bien alcanzan precisiones cercanas al 98% (Wang et al., 2020), requieren equipos de cómputo especializados. Se identificaron algunas limitaciones, como la sensibilidad a condiciones de iluminación y las variaciones personales en la ejecución de señas, lo cual señala áreas de mejora para futuras versiones. No obstante, los resultados respaldan la factibilidad del sistema como herramienta educativa y de apoyo en contextos de inclusión.

.

4. Conclusiones

El desarrollo del prototipo portátil para la traducción del alfabeto dactilológico mexicano, basado en Raspberry Pi 3, demostró ser una solución técnicamente viable, accesible y eficiente para apoyar la comunicación de personas con discapacidad auditiva. La integración de las bibliotecas OpenCV y MediaPipe permitió una detección precisa de gestos en tiempo real, facilitando la interpretación de señales manuales mediante un enfoque geométrico que evita el uso de redes neuronales de alto costo computacional. Este diseño alcanzó resultados satisfactorios en la identificación de letras, incluyendo aquellas que implican movimiento, con tiempos de respuesta reducidos.

En síntesis, la principal contribución de este trabajo radica en demostrar que es posible implementar un sistema portátil, autónomo y de bajo costo, que logra precisiones superiores al 85% en condiciones reales de uso y con un rendimiento competitivo frente a sistemas más complejos. La incorporación de una interfaz con pantalla y teclado con touchpad mejoró la experiencia del usuario, brindando control intuitivo y retroalimentación inmediata.

Un aspecto relevante del sistema es su capacidad para reconocer letras que implican movimiento, como la “J” y la “Z”. Si bien la mayoría de los trabajos reportados en la literatura se enfocan en gestos estáticos (BrightSign, 2022; Wang et al., 2020; Camgoz et al., 2020), la detección confiable de movimientos constituye un avance significativo. Esto abre la posibilidad de afrontar reconocimiento de palabras y gestos fluidos en conversación como trabajo futuro.

Si bien se identificaron áreas de mejora, como optimizar la detección en condiciones de baja iluminación y ampliar las pruebas con un mayor número de usuarios, los resultados evidencian que la propuesta responde de manera efectiva al problema inicial y sienta las bases para la evolución hacia sistemas capaces de traducir palabras o frases completas.

Referencias

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O’Reilly Media. ISBN: 978-0-596-51613-0

BrightSign Glove. (2022). Translate any sign into any language [Sitio web]. https://www.brightsignglove.com/

Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 10023–10033). IEEE. https://doi.org/10.1109/CVPR42600.2020.01004

CyberGlove Systems. (2015). CyberGlove II [Sitio web]. https://www.cyberglovesystems.com/cyberglove-ii/

González, C., Martínez, A., & López, J. (2021). Accesibilidad y comunicación en personas con discapacidad auditiva en México. Revista Latinoamericana de Inclusión Educativa, 15(1), 45–59.

Instituto Nacional de Estadística y Geografía (INEGI). (2020). Censo de Población y Vivienda 2020. INEGI.

Instituto Nacional de Estadística y Geografía (INEGI). (2024, 28 de noviembre). Estadísticas a propósito del Día Internacional de las Personas con Discapacidad (3 de diciembre) (Comunicado de prensa núm. 684/24) [PDF]. https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2024/EAP_PCD24.pdf

Li, D., Rodriguez, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (pp. 1459–1469). IEEE. https://openaccess.thecvf.com/content_WACV_2020/papers/Li_Word-level_Deep_Sign_Language_Recognition_from_Video_A_New_Large-scale_WACV_2020_paper.pdf

Li, D., Yu, X., Xu, C., & Li, H. (2023). Sign language recognition with transformer-based spatio-temporal features. Pattern Recognition, 139, 109488. https://doi.org/10.1016/j.patcog.2023.109488

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... Grundmann, M. (2019, junio 14). MediaPipe: A framework for building perception pipelines [Preprint]. arXiv. https://arxiv.org/abs/1906.08172

Muhammad, Y., Jan, M. A., Ahmad, S., Mastorakis, S., & Zada, B. (2022). A deep learning-based smart assistive framework for visually impaired people. In Proceedings of the 2022 IEEE International Conference on Omni-Layer Intelligent Systems (COINS). IEEE. https://doi.org/10.1109/COINS54846.2022.9854984

Organización de las Naciones Unidas (ONU). (2006). Convención sobre los Derechos de las Personas con Discapacidad.

Szeliski, R. (2022). Computer vision: Algorithms and applications (2.ª ed.). Springer. https://doi.org/10.1007/978-3-030-34372-9

Upton, E., & Halfacree, G. (2014). Raspberry Pi user guide (3.ª ed.). John Wiley & Sons.

Wang, Y., Chen, J., & Li, F. (2020). Real-time sign language recognition using deep learning models. International Journal of Computer Vision and Signal Processing, 8(4), 112–125.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C., & Grundmann, M. (2020). MediaPipe Hands: On-device real-time hand tracking. arXiv. https://arxiv.org/abs/2006.10214