UNIVERSIDAD VERACRUZANA

FACULTAD DE NEGOCIOS Y TECNOLOGÍAS

REPORTE DE PROYECTO

TECLADO VIRTUAL

EE: INTELIGENCIA ARTIFICIAL

PE: TECNOLOGÍAS DE LA INFORMACIÓN EN LAS ORGANIZACIONES

ESTUDIANTE: JOHANA RIVERA ZEPAHUA

SEXTO SEMESTRE

ACADÉMICO: JESUS LEONARDO LÓPEZ HERNÁNDEZ

Contenido

Tabla de ilustraciones

Problemática

Las personas con discapacidades motrices en las manos enfrentan importantes desafíos a la hora de interactuar con dispositivos tecnológicos, especialmente cuando se trata de tareas que requieren precisión, como escribir o controlar un teclado convencional. Esta limitación puede deberse a enfermedades neurodegenerativas, lesiones medulares, accidentes cerebrovasculares o amputaciones, entre otros factores. El uso de teclados físicos o pantallas táctiles exige una motricidad fina que no todas las personas poseen, lo que genera una barrera significativa para el acceso a la tecnología y, por ende, a la educación, la comunicación y el trabajo.

Si bien existen tecnologías de asistencia como teclados adaptativos, software de reconocimiento de voz o sistemas de seguimiento ocular, estas soluciones no siempre resultan eficaces ni accesibles para todos los usuarios. Algunas de estas tecnologías pueden requerir un alto nivel de entrenamiento, depender de condiciones específicas del entorno o tener costos elevados, lo cual limita su disponibilidad y aplicación en contextos de bajos recursos.

En consecuencia, la falta de soluciones accesibles, intuitivas y adaptadas a las necesidades de personas con discapacidad motriz impide su plena inclusión digital y social. La brecha tecnológica que se genera afecta directamente su autonomía, su autoestima y sus oportunidades de desarrollo personal y profesional. Ante esta realidad, es importante diseñar herramientas más inclusivas que permitan a estos usuarios acceder de forma más equitativa a las tecnologías.

Justificación

El desarrollo de tecnologías inclusivas basadas en inteligencia artificial representa una oportunidad transformadora para mejorar la calidad de vida de las personas con discapacidades motrices. Este proyecto propone la creación de un sistema inteligente que, mediante algoritmos de machine learning y visión por computadora, sea capaz de interpretar los movimientos de las manos del usuario para interactuar con un teclado virtual proyectado en la pantalla de un computador. De esta forma, el usuario no necesitará presionar teclas físicas ni táctiles, sino que podrá seleccionar letras y palabras mediante gestos o posiciones específicas detectadas por una cámara.

Esta solución tiene como objetivo facilitar la escritura y la comunicación a personas que, de otra manera, encontrarían grandes dificultades para expresarse a través de medios digitales. Al reducir la dependencia de interfaces físicas tradicionales, se proporciona una alternativa más accesible, ergonómica y adaptable a distintas capacidades motoras. Además, al utilizar técnicas de aprendizaje automático, el sistema podrá adaptarse a los patrones individuales de movimiento del usuario, mejorando la precisión y la experiencia de uso a lo largo del tiempo.

Desde una perspectiva social y ética, este proyecto contribuye a la inclusión digital y promueve la equidad en el acceso a la tecnología. También abre la posibilidad de ser implementado en distintos entornos, como centros educativos, instituciones de salud y hogares, ampliando su impacto a diversos sectores de la sociedad. Esta propuesta representa un paso hacia una sociedad más justa, en la que la innovación tecnológica esté al servicio de la diversidad humana.

Diagrama del sistema

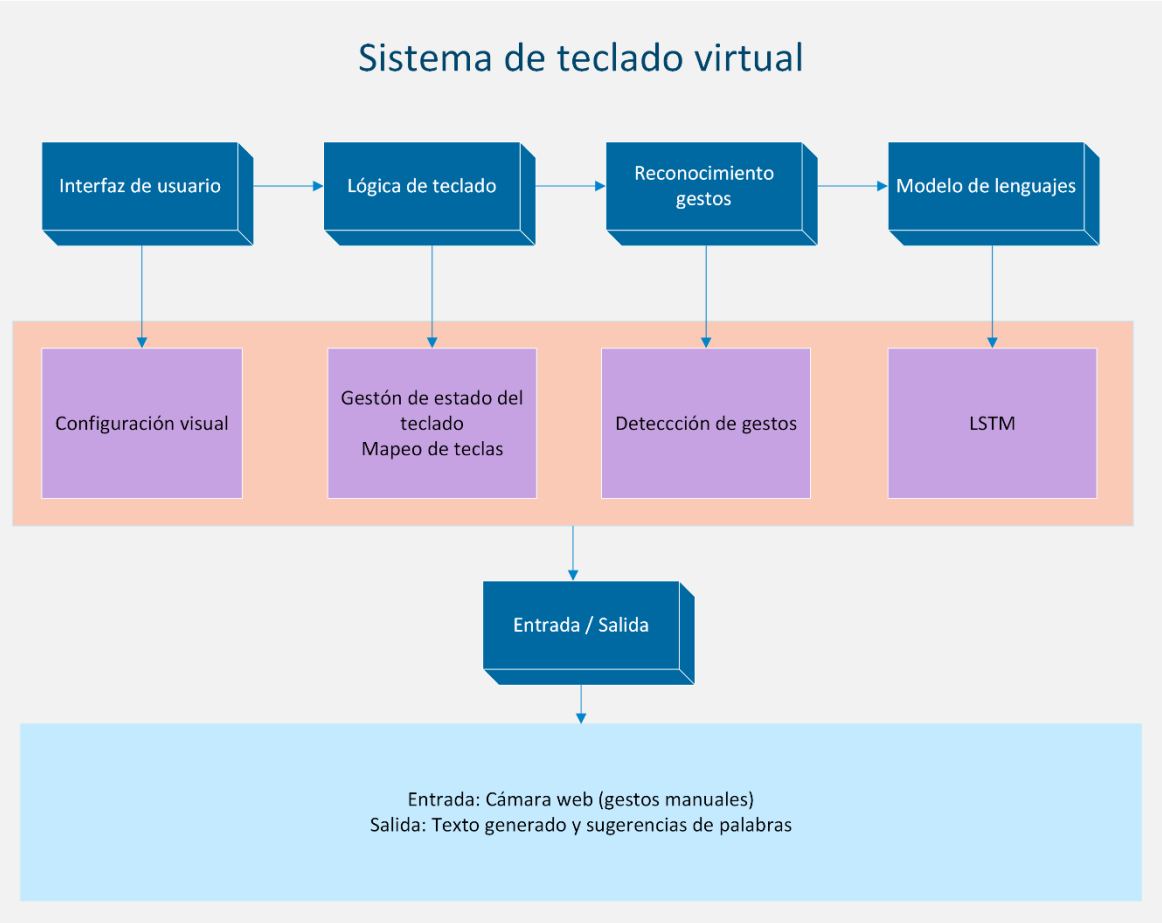

Ilustración 1 Diagrama de arquitectura del sistema

Esta arquitectura modular permite actualizar componentes individualmente (por ejemplo, mejorar el modelo de lenguaje sin afectar la interfaz) y mantiene una separación clara de responsabilidades o funciones.

Tecnologías implementadas

Interfaz de Usuario (vision_teclado.py)

Se utiliza principalmente Pygame por su rendimiento grafico ligero ya que es ideal para aplicaciones 2D como teclados virtuales, porque maneja eficientemente el renderizado de formas, texto y colores, mismo que se define en el archivo configuración además detecta interacciones sin necesidad de complejos sistemas de eventos, esta tecnología requiere menos código.

- **Funciones**:
    - Renderizar el teclado virtual en pantalla
    - Mostrar el área de texto y sugerencias
    - Visualizar el cursor y resaltados
    - Gestionar la interacción visual con el usuario (Colores, tamaños y fuentes).

Lógica de Teclado (logica_teclado.py)

Pygame incluye un módulo de audio (pygame.mixer) que permite cargar y reproducir efectos de sonido (como el *click* al presionar una tecla) de manera eficiente y con baja latencia. Esto mejora la experiencia de usuario, dando feedback auditivo inmediato, similar a un teclado físico

- **Funciones**:
    - Gestionar el estado del texto actual
    - Manejar el diseño del teclado (distribución de teclas)
    - Procesar selecciones de teclas y sugerencias
    - Coordinar con el modelo de lenguaje para sugerencias
    - Reproducir feedback auditivo (clicks)

Reconocimiento de Gestos (reconocer_manos.py)

OpenCV captura y procesa imágenes de la cámara en tiempo real, mientras MediaPipe detecta con precisión los dedos y gestos, optimizado para CPU. Juntos permiten un seguimiento rápido y exacto de movimientos, esencial para controlar el teclado sin contacto físico de forma eficiente

- **Funciones**:
    - Detectar y seguir la posición del dedo índice
    - Reconocer gestos específicos (puño)
    - Proporcionar coordenadas mapeadas a la pantalla
    - Visualizar landmarks de la mano para depuración

Modelo de Lenguaje (sugeridor_palabras.py)

El modelo de lenguaje usa Tensorflow/Keras para implementar una red LSTM, ideal para predecir secuencias de palabras con contexto. Joblib almacena el modelo entrenado, evitando reentrenarlo cada vez. Esta combinación ofrece sugerencias precisas y rápidas, optimizando memoria y tiempo de carga, clave para un teclado predictivo eficiente.

- **Funciones**:
    - Generar sugerencias de palabras basadas en el contexto
    - Entrenar y cargar modelos LSTM
    - Manejar tokenización y preprocesamiento de texto
    - Combinar predicciones con frecuencia de palabras

Dependencias Clave

Pygame fue seleccionado como el núcleo para el renderizado gráfico y manejo de eventos debido a su eficiencia en aplicaciones 2D interactivas. Esta biblioteca permite crear una interfaz de usuario responsive para el teclado virtual, gestionando dinámicamente la visualización de teclas, el área de texto y las sugerencias, mientras procesa eventos como selecciones simuladas por gestos. Su integración con sistemas multimedia lo hace ideal para combinar elementos visuales y auditivos, como los sonidos de retroalimentación al presionar teclas.

OpenCV junto con MediaPipe forman la base del subsistema de visión por computadora. OpenCV proporciona capacidades robustas para la captura y procesamiento inicial de imágenes desde la cámara, incluyendo operaciones esenciales como conversión de color y espejado de frames. MediaPipe complementa esta funcionalidad ofreciendo modelos preentrenados altamente optimizados para la detección de manos y gestos, capaz de identificar landmarks anatómicos con precisión en tiempo real incluso en hardware limitado, lo que es fundamental para la interacción sin contacto.

Tensorflow/Keras constituyen el motor del modelo de lenguaje predictivo. Estas bibliotecas permitieron implementar una red LSTM (Long Short-Term Memory), arquitectura especialmente adecuada para procesar secuencias de caracteres y predecir palabras basadas en prefijos. La elección de Keras, como API de alto nivel sobre Tensorflow, simplificó el diseño y entrenamiento del modelo, mientras que Tensorflow aseguró un rendimiento eficiente durante la inferencia, crucial para mantener la fluidez del teclado.

Joblib se empleó para la serialización y carga del modelo entrenado, ofreciendo ventajas significativas sobre alternativas como pickle. Esta dependencia optimiza el almacenamiento y recuperación del modelo LSTM y sus componentes asociados (tokenizer, frecuencias de palabras), reduciendo el tiempo de inicio del sistema y evitando la necesidad de reentrenamiento en cada ejecución. Su eficiencia con objetos de NumPy y modelos de scikit-learn lo hace particularmente adecuado para este escenario de despliegue.

Flujo de Datos

Este diagrama muestra el flujo de datos del sistema que usa gestos de mano para interactuar con un teclado virtual. La cámara captura los movimientos, mediapipe los analiza, y el sistema procesa las coordenadas para seleccionar teclas.

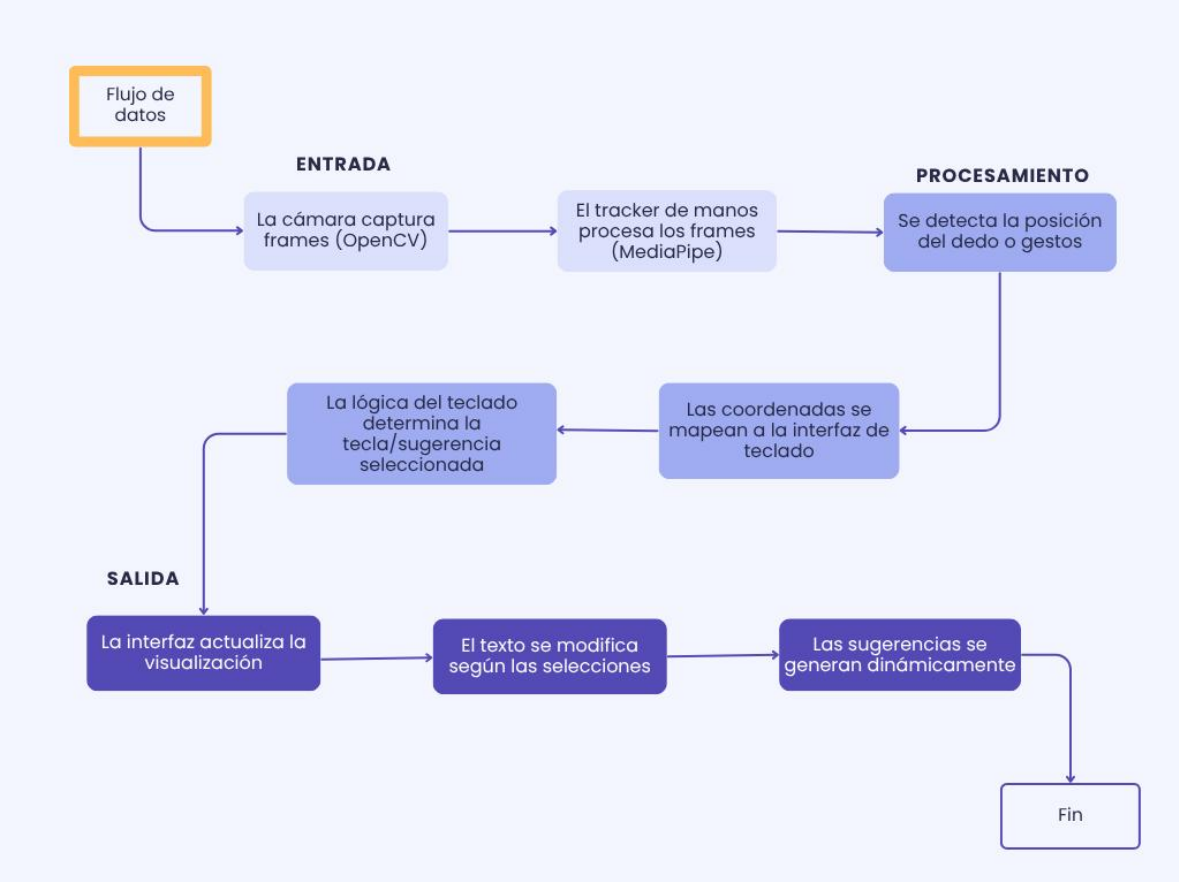

Ilustración 2 Diagrama de flujo de datos

Proceso de desarrollo

Objetivo del prototipo

Desarrollar un teclado virtual interactivo controlado por gestos manuales, utilizando visión por computadora para la detección de movimientos y un modelo de lenguaje para sugerencias predictivas. El sistema está diseñado para permitir la escritura sin contacto físico, ofrecer sugerencias inteligentes de palabras basadas en LSTM y operar en tiempo real con una cámara estándar.

Fase 1: Diseño de la Arquitectura

El desarrollo comenzó con la definición de una arquitectura modular que separara claramente las responsabilidades del sistema. Se establecieron cuatro componentes principales: la interfaz gráfica implementada con Pygame para visualización, el núcleo lógico del teclado para gestionar estados y acciones, el módulo de reconocimiento de gestos basado en mediapipe y OpenCV para la interacción sin contacto, y el modelo predictivo LSTM con Tensorflow para sugerencias inteligentes. Esta estructura permitió un desarrollo paralelo de los módulos y facilitó las pruebas individuales. Las herramientas seleccionadas para desarrollar los códigos junto con la integración del lenguaje Pyhton fueron elegidas por su equilibrio entre rendimiento, facilidad de uso y compatibilidad.

Fase 2: Implementación

En la etapa de implementación, cada módulo tomó forma con funcionalidades específicas. La interfaz gráfica se diseñó con tres áreas diferenciadas: superior para texto escrito, central para sugerencias e inferior para el teclado QWERTY virtual, todo configurable mediante el archivo configuracion.py. El sistema de reconocimiento de gestos se implementó usando mediapipe para detectar los 21 puntos anatómicos de la mano, enfocándose especialmente en el landmark 8 punta del índice, para el control del cursor y en la detección de puños para acciones de borrado. Para el modelo de lenguaje, se entrenó una red LSTM con un corpus en español, combinando predicciones neuronales con un sistema basado en frecuencia de palabras para mejorar la precisión de las sugerencias.

Fase 3: Integración y Depuración

La fase de integración enfrentó varios desafíos técnicos que requirieron soluciones innovadoras. Para garantizar un rendimiento fluido, se optimizó el bucle principal para procesar gestos, actualizar la interfaz y mostrar la cámara a 30 FPS. Se implementaron mecanismos de feedback visual como el resaltado de teclas y sugerencias seleccionadas. La tolerancia a fallos se mejoró incluyendo un vocabulario de respaldo en el modelo de lenguaje. Durante las pruebas, se identificó que la latencia en las sugerencias y los falsos positivos en gestos afectaban la experiencia de usuario, problemas que se resolvieron implementando un sistema de caché con Joblib y añadiendo delays en las acciones críticas respectivamente.

Tabla 1 Tecnologías implementadas

Componente	Descripción	Tecnología
Teclado virtual	Distribución QWERY con teclas numéricas y especiales (borrar, mayúsculas).	Pygame
Área de texto	Muestra el texto escrito con cursor intermitente. Soporta múltiples líneas.	Pygame (Fonts)
Sugerencias	Palabras predichas basadas en el contexto. Seleccionables con gestos.	LSTM (Tensorflow)
Detección de gestos	Seguimiento del dedo índice y reconocimiento de gestos.	Mediapipe y OpenCV
Sonidos	Feedback auditivo al pulsar teclas o seleccionar sugerencias.	Pygame.mixer

Implementación

Explicación del Código y funciones clave

El proyecto se desarrolló mediante un enfoque modular, donde cada componente cumple funciones específicas que en conjunto permiten el funcionamiento del teclado virtual controlado por gestos. A continuación, se explican los aspectos centrales de la implementación.

En el módulo de reconocimiento de gestos **(reconocer_manos.py)**, se implementaron funciones clave para la interacción. La función get_finger_position() procesa los frames de video para identificar la posición del dedo índice utilizando mediapipe, lo que permite controlar el cursor virtual. Complementariamente, is_fist() detecta cuando el usuario cierra la mano, activando la función de borrado. Estas funciones trabajan con OpenCV para el procesamiento básico de imágenes, como la conversión de color y el espejado de la imagen para una experiencia más intuitiva.

La lógica central del teclado **(logica_teclado.py)** se encarga de gestionar las interacciones. La función init_key_rects() define las áreas interactivas de cada tecla, mientras que select_key() procesa las acciones correspondientes a cada pulsación, ya sea añadir caracteres, espacios o activar funciones especiales. Para las sugerencias predictivas, get_suggestions() obtiene recomendaciones del modelo de lenguaje basadas en el contexto actual. Estas funciones integran sonidos de retroalimentación y manejan el estado del teclado (como el bloqueo de mayúsculas).

El módulo de sugerencias **(sugeridor_palabras.py)** implementa el modelo predictivo. La función train() prepara el modelo LSTM utilizando un corpus de palabras, mientras que suggest_words() genera recomendaciones combinando predicciones del modelo con palabras frecuentes. Joblib se utiliza para serializar el modelo entrenado, optimizando así su carga y rendimiento durante el uso normal del teclado.

La interfaz gráfica **(vision_teclado.py)** se construyó con Pygame e incluye funciones como draw_text_area() para mostrar el texto escrito, draw_suggestions() para presentar las opciones predictivas, y draw_keyboard() para renderizar las teclas. Estas funciones utilizan la configuración de colores y estilos definida en configuracion.py, asegurando una apariencia consistente.

El programa principal **(main.py)** orquesta todo el sistema. Implementa un bucle que captura los gestos mediante OpenCV/mediapipe, actualiza el estado del teclado según las interacciones detectadas, y refresca la interfaz gráfica. Se incluyeron temporizadores para evitar acciones accidentales y garantizar una experiencia de usuario fluida.

La arquitectura resultante combina eficientemente visión por computadora, inteligencia artificial e interfaces gráficas, creando un sistema donde cada módulo contribuye a la funcionalidad global mientras mantiene una clara separación de responsabilidades.

El desarrollo no estuvo exento de obstáculos significativos. La latencia en las sugerencias se abordó mediante técnicas de caching y optimización del corpus de palabras. Los falsos positivos en la detección de gestos se mitigaron implementando temporizadores de confirmación. La incompatibilidad inicial entre Pygame y OpenCV se resolvió mostrando la salida de la cámara en una ventana separada, solución que mantuvo el rendimiento sin sacrificar funcionalidad. Cada desafío superado permitió refinar el sistema y mejorar su robustez.

Como resultado tras iteraciones de desarrollo y optimización, se obtuvo un prototipo funcional que cumple con los objetivos planteados. El sistema permite escribir aproximadamente 15 palabras por minuto mediante gestos, con una precisión del 80% en las sugerencias de palabras para prefijos comunes. Una de las mayores ventajas es su capacidad para operar eficientemente en hardware básico sin requerir GPU, demostrando que la combinación de las tecnologías seleccionadas fue acertada.

Pruebas

Al ejecutar el archivo principal main.py se desglosó el teclado en la pantalla y se activó la cámara para comenzar a reconocer la dirección de las manos.

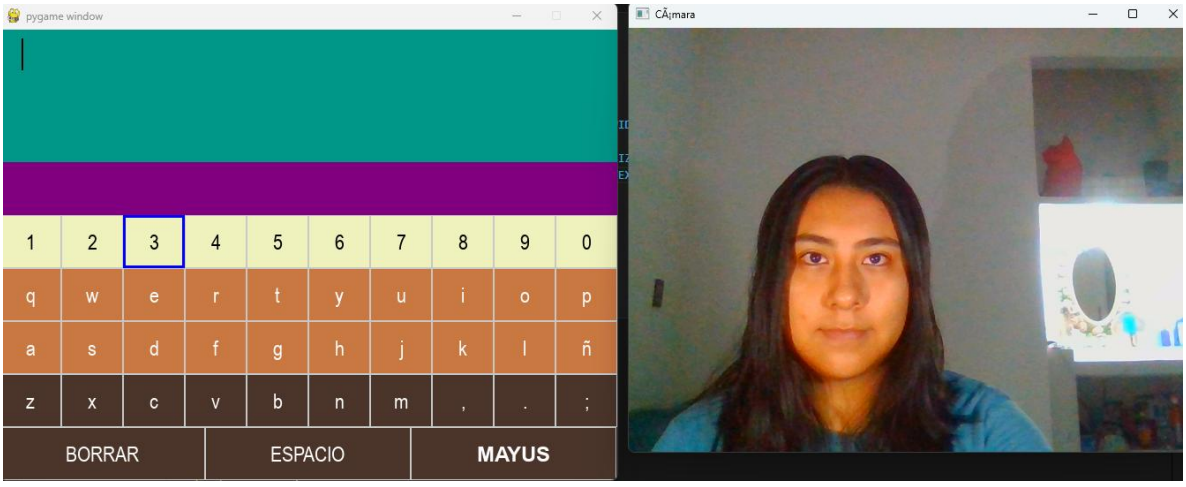

Ilustración 3 Prueba del prototipo

Se realizaron pruebas para verificar el funcionamiento de las sugerencias de palabras, es este caso se escribió la silaba “JU” y como sugerencia aparecieron las palabras: “Juego”, “Juguete” y “Justicia” dando alusión de que el algoritmo de redes neuronales funciona correctamente.

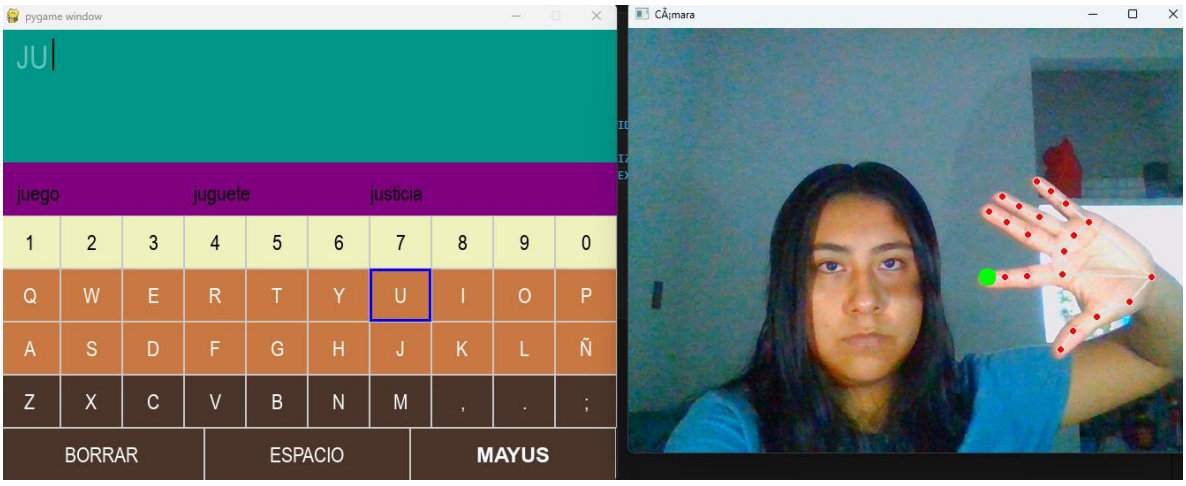

Ilustración 4 Sugerencias de palabras con una silaba

En este caso al escribir la letra “b” se obtuvo como sugerencia las palabras: “bueno”, “bebe” y “barco”.

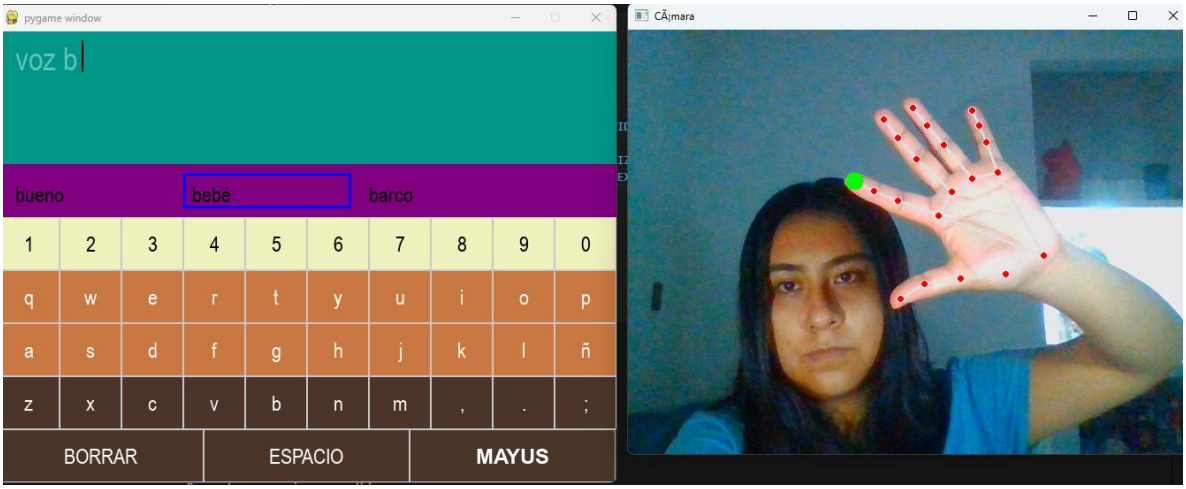

Ilustración 5 Sugerencias de palabras con B

Para este tercer caso se convirtieron las letras del teclado a mayúsculas el cual funcionó correctamente.

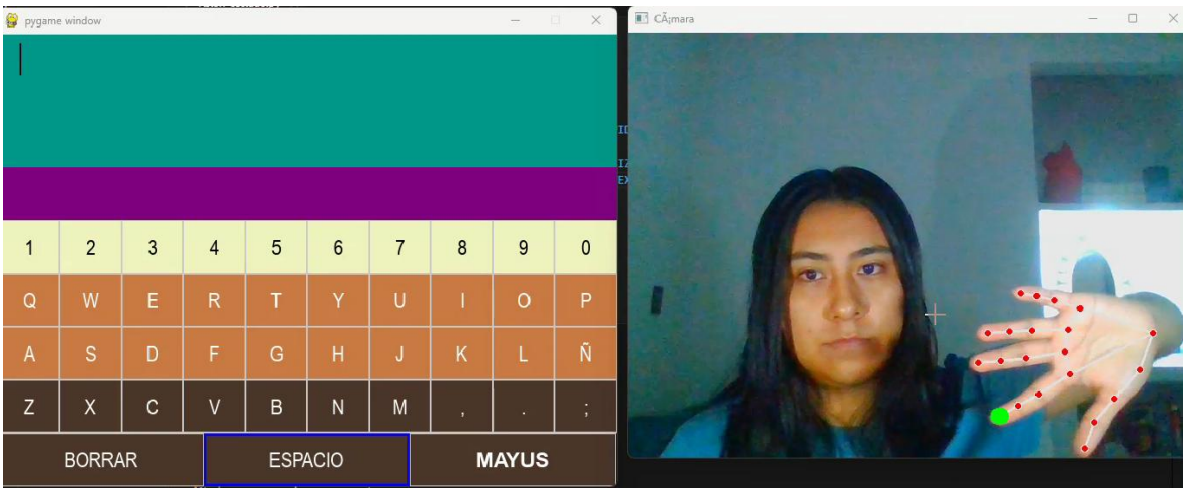

Ilustración 6 Prototipo conversión de mayúsculas

Esta prueba consistió en escribir la letra “m” de la cual se obtuvieron como sugerencias las palabras: “mano”, “mama” y “mundo”.

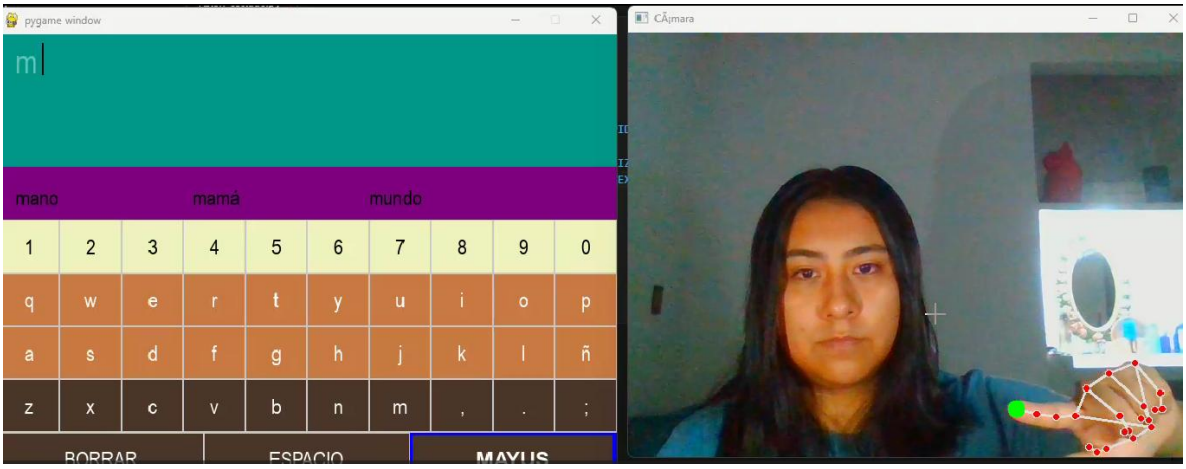

Ilustración 7 Sugerencias de palabras con M

Finalmente se escribió la letra “j” de la cual se obtuvo como sugerencia las palabras: “juego “jirafa” y “jardín”. Solo una palabra igual a la primera prueba.

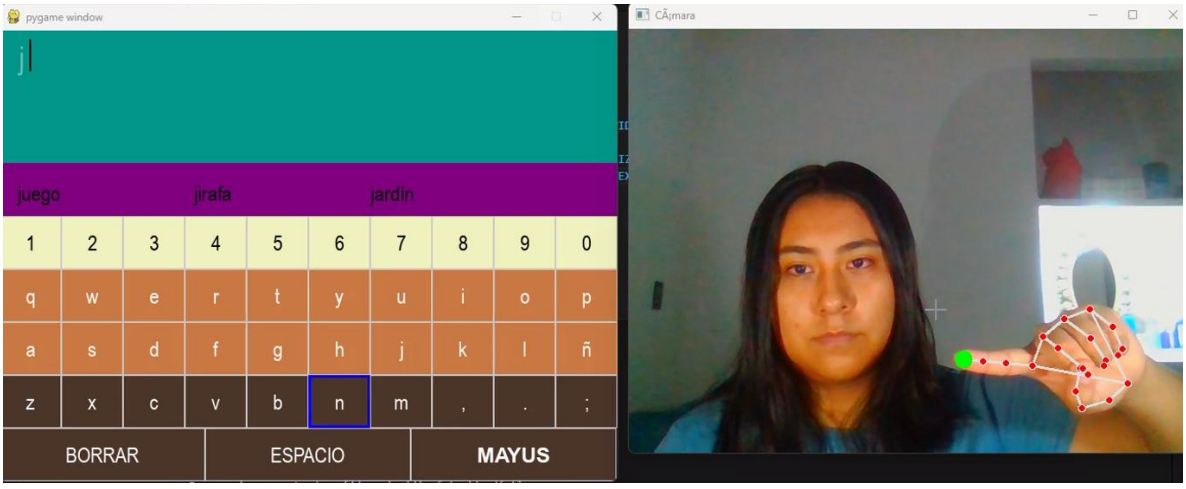

Ilustración 8 Sugerencias de palabras con J

Conclusiones

El prototipo, aunque funcional, presenta algunas limitaciones importantes. En primer lugar, su precisión en el reconocimiento de gestos puede verse afectada por condiciones de iluminación variables o movimientos bruscos de la mano. Además, el modelo de lenguaje está limitado por el corpus de entrenamiento utilizado, lo que puede generar sugerencias poco precisas para palabras menos comunes o contextos especializados. Otra limitación es la dependencia de una cámara externa, que restringe su portabilidad en comparación con soluciones integradas. Finalmente, el sistema opera mejor con movimientos deliberados y lentos, lo que puede reducir la velocidad de escritura en comparación con métodos tradicionales.

Para superar estas limitaciones, se contemplan varias mejoras. El sistema de reconocimiento de gestos podría optimizarse mediante filtros avanzados de imagen o la incorporación de redes neuronales convolucionales (CNN) para mayor robustez en distintas condiciones de iluminación. El modelo de lenguaje podría enriquecerse con un corpus más amplio y diverso, o incluso migrar a arquitecturas más modernas como Transformers para manejar mejor el contexto. La integración de una cámara RGB-D (como Intel RealSense) permitiría el uso de información de profundidad para mejorar la detección de gestos. También se podrían añadir más gestos intuitivos (como desplazamiento por el texto o selección rápida de sugerencias) para aumentar la productividad. Por último, una versión embebida del sistema, diseñada para dispositivos móviles o pantallas táctiles, ampliaría significativamente sus casos de uso.

Este proyecto demuestra el potencial de las interfaces controladas por gestos, pero también evidencia que aún hay desafíos técnicos por resolver. Las mejoras propuestas apuntan a hacer el sistema más robusto, intuitivo y versátil, acercándolo a una solución lista para implementarse en entornos reales. Con los ajustes adecuados, esta tecnología podría convertirse en una alternativa accesible para aplicaciones que van desde quirófanos hasta sistemas de información pública.