Universidad Veracruzana

REPORTE DE PROYECTO

EE: Inteligencia Artificial
PE: Tecnologías De La Información en las Organizaciones
Alumna: Tania Arenas Sánchez
Matricula: S22004522
Semestre: Sexto
Docente: Jesus Leonardo López Hernández

13 junio 2025

TABLA DE CONTENIDO

Problemática.

Hoy en día, escribir un mensaje, llenar un formulario o simplemente interactuar con una computadora es algo que muchas personas hacen con facilidad. Para la mayoría, usar un teclado físico o una pantalla táctil es una acción común y cotidiana. Sin embargo, para quienes tienen alguna discapacidad motriz o dificultad física, estas tareas tan simples pueden convertirse en grandes obstáculos.

Imagina no poder usar tus manos para escribir en un teclado convencional. Para personas con parálisis o enfermedades la comunicación digital se convierte en una barrera. Aunque existen soluciones tecnológicas, muchas son costosas, requieren equipos especializados o no están al alcance de todos.

Pensando en esto, se elaboró la idea de desarrollar un teclado digital que puede ser controlado con gestos de la mano, captados por una cámara. En lugar de presionar físicamente una tecla, el usuario simplemente mueve su dedo frente a la cámara para seleccionar letras o palabras en una interfaz virtual.

Este proyecto tiene como objetivo facilitar el acceso a la escritura digital, sin necesidad de contacto físico, de forma accesible y económica. Utilizando visión por computadora con MediaPipe y una interfaz gráfica construida en Python, se crea una alternativa que puede ser útil no solo para personas con discapacidad, sino también en contextos donde se requiere evitar el contacto directo con superficies.

Más que un teclado, este sistema representa una nueva forma de comunicación una herramienta pensada para incluir, conectar y dar voz a quienes enfrentan barreras físicas en su vida diaria.

Justificación.

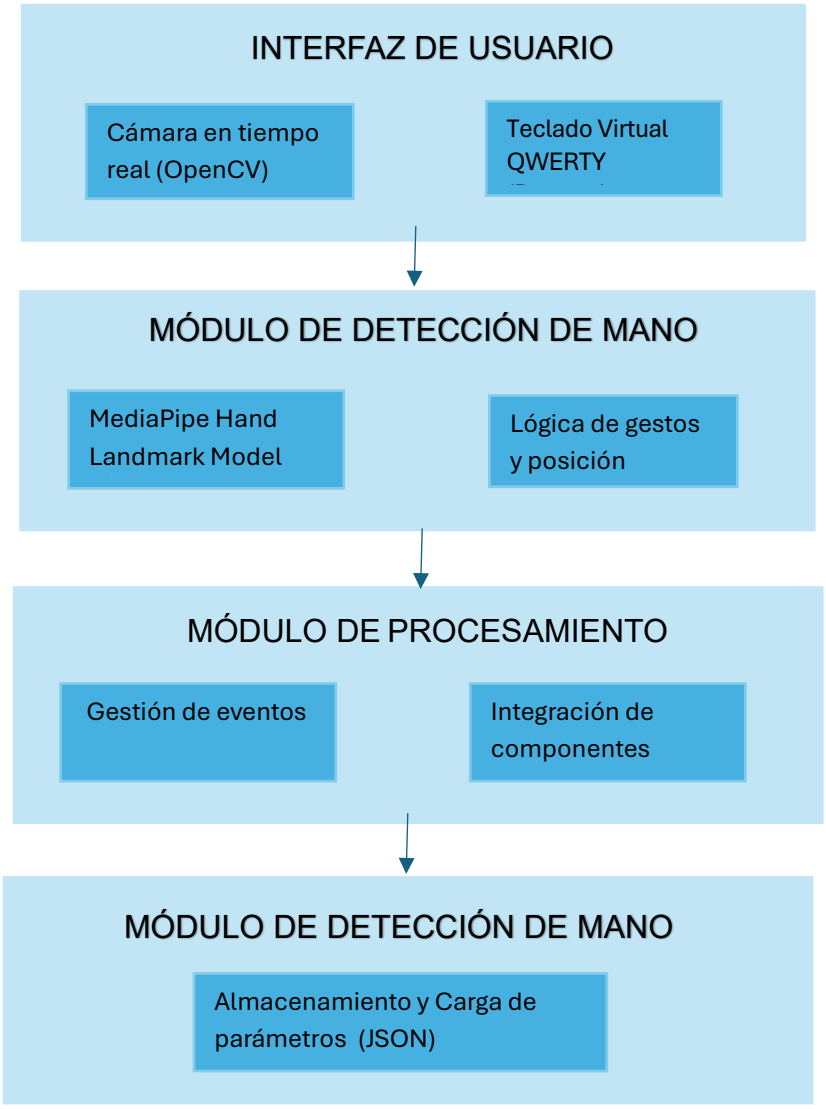
En la actualidad, la tecnología se ha convertido en una herramienta esencial en la vida cotidiana. Sin embargo, su accesibilidad sigue siendo un desafío para muchas personas con discapacidades motrices, quienes enfrentan dificultades para interactuar con dispositivos tradicionales que no están diseñados pensando en sus necesidades. Aunque existen soluciones adaptadas, la mayoría son costosas, difíciles de conseguir o requieren conocimientos avanzados para su configuración.

Este proyecto ofrece una alternativa accesible e innovadora que permite a las personas escribir sin necesidad de utilizar un teclado físico, empleando únicamente el movimiento de sus manos. De este modo, quienes tienen dificultades de movilidad pueden acceder y utilizar la tecnología con mayor facilidad

Lo que hace valioso a este proyecto es que:

- Utiliza herramientas gratuitas y accesibles, como Python y una cámara.
- Se adapta a distintas necesidades.
- Puede aplicarse en entornos donde el contacto físico con objetos está restringido, como hospitales o laboratorios.
- Es fácil de usar y flexible, permitiendo modificaciones según las necesidades individuales.

Arquitectura del sistema



Tecnologías Implementadas.

1. Capa de Presentación (Interfaz de Usuario)

- **Pygame** (v2.6.1+):
    - Renderizado del teclado virtual QWERTY
    - Gestión de eventos de interfaz
    - Visualización combinada (cámara + teclado)
- **OpenCV** (v4.5+):
    - Captura de video en tiempo real
    - Procesamiento inicial de frames
    - Conversión de espacios de color

2. Capa de Detección y Seguimiento

- **MediaPipe Hands** (v0.10+):
    - Modelo preentrenado para detección de landmarks de mano
    - Seguimiento de 21 puntos clave por mano
    - Estimación de posición 3D relativa
- **Lógica de Gestos Personalizada**:
    - Detección de posición del dedo índice (x,y)
    - Algoritmo de detección de "clic" (umbral de movimiento vertical)
    - Filtrado básico de movimientos

3. Capa de Procesamiento Central

- **Integración de Componentes**:
    - Coordinación entre detección y teclado

    - Mapeo posición física → posición en interfaz
    - Gestión de estados (hover, selección)
- **Lógica de Texto**:
    - Construcción de cadena de texto
    - Manejo de teclas especiales (BACKSPACE, ENTER, SPACE)

4. Capa de Persistencia

- **Configuración JSON**:
    - Almacenamiento de parámetros ajustables
    - Persistencia entre sesiones
    - Sistema de carga/guardado automático

Flujo de datos:

Cámara → OpenCV → MediaPipe → Procesamiento Gestos → Pygame → Interfaz Usuario

1. **Cámara:** Captura imágenes en tiempo real del movimiento de la mano.
2. **OpenCV:** Optimiza las imágenes para mejorar la detección, ajustando brillo y contraste.
3. **MediaPipe:** Identifica la mano y la posición de los dedos en cada imagen.
4. **Procesamiento de Gestos:** Traduce los movimientos en acciones, como seleccionar o hacer clic.
5. **Pygame:** Representa gráficamente el teclado y las teclas activas según el gesto realizado.

6. **Interfaz de Usuario:** Muestra el texto escrito y permite la interacción con el sistema.
7. **Configuración:** Permite ajustes como sensibilidad del gesto, tamaño del teclado y personalización del comportamiento.

Interacción:

- La configuración influye en la detección y respuesta del sistema.
- La interfaz de usuario recibe los datos procesados y muestra el resultado final.

Proceso de desarrollo

Objetivo:

Desarrollar un teclado virtual controlado por gestos manuales para personas con discapacidad motriz, utilizando visión por computadora y machine learning.

Fase 1: Diseño y Planificación

En esta etapa se establecieron las bases del sistema. Se comenzó analizando a fondo las necesidades del proyecto y las tecnologías disponibles. Se definió una arquitectura modular, dividiendo el sistema en partes principales: captura de imágenes, detección de gestos, interfaz de usuario y predicción de texto. Se eligieron herramientas específicas para cada tarea: OpenCV para procesar imágenes en tiempo real, MediaPipe para seguir con precisión los movimientos de la mano, Pygame para construir la interfaz gráfica, y TensorFlow para entrenar el modelo predictivo con LSTM.

Fase 2: Implementación Técnica

Aquí se construyó cada módulo del sistema. La parte de captura visual fue optimizada para trabajar a 30 cuadros por segundo, con mejoras en la iluminación y el color de las imágenes.

El sistema de gestos fue diseñado para reconocer movimientos intencionados, como un “clic” con el dedo índice, y evitar errores usando un historial de movimientos.
La interfaz gráfica se desarrolló con un teclado virtual dividido en tres capas (letras, números, mayúscula y opción de borrado) un área de texto y un panel con sugerencias.
El modelo LSTM se entrenó con un corpus en español de palabras, combinando inteligencia artificial y análisis por frecuencia para dar mejores sugerencias de palabras.

Fase 3: Integración y Optimización

Una vez que todos los módulos funcionaban por separado, se integraron en un solo sistema. Durante esta fase surgieron problemas como demoras en las predicciones o gestos mal detectados.

Para solucionarlos, se usó una caché (con Joblib) que guarda sugerencias frecuentes y se calibraron los gestos para mejorar la precisión. También se mejoró la interfaz agregando animaciones al presionar teclas, opciones de accesibilidad como cambio de tamaño de letra o temas de alto contraste.

Elementos del prototipo desarrollado.

1. Módulo de Captura Visual

- Implementado con OpenCV para adquisición de imágenes en tiempo real (720p a 30 FPS)

Sistema de preprocesamiento con:

- Corrección automática de brillo/contraste
- ROI (Region of Interest) dinámica para reducir carga computacional
- Conversión BGR→RGB para compatibilidad con MediaPipe

2. Motor de Seguimiento Manual

Pipeline de detección con MediaPipe Hands:

- Identificación de 21 landmarks anatómicos
- Algoritmo de estabilización basado en filtro de Kalman

Detección de gestos configurables:

- Clic: Movimiento vertical rápido (umbral ajustable en config.json)
- Hover: Posición estática sobre tecla (>0.5s)
- Scroll: Movimiento horizontal continuo

3. Teclado Virtual QWERTY

Implementado en Pygame con:

- capas intercambiables (letras/números/símbolos)

Teclas de función personalizables:

- Borrado inteligente (por carácter/palabra completa)
- Cambio de idioma (ES/EN)
- Modo manos libres (activación por gesto)

Diseño adaptable:

- Tamaño de teclas ajustable (20px-60px)
- 4 esquemas de color predefinidos (claro/oscuro/alto contraste)

4. Sistema Predictivo

Modelo híbrido LSTM + N-gramas:

- Red neuronal con 2 capas LSTM (128 unidades)
- Vocabulario base de 50k palabras en español
- Mecanismo de fallback a diccionario frecuencial

Interfaz de sugerencias:

- Panel desplegable con 3 opciones
- Autocompletado con tecla rápida (gesto de "empujar")

5. Módulo de Configuración

Archivo JSON con parámetros editables

6. Sistema de Feedback

Retroalimentación multimodal:

- Visual: Animación de teclas presionadas + tooltips
- Auditivo: Sonidos opcionales por acción (activado en config)
- Háptico: Vibración simulada mediante parpadeo de tecla.

Implementación

Explicación del código

1. **ajustes_sistema.py**

Gestiona toda la configuración personalizable del sistema. Almacena y recupera ajustes como la sensibilidad de los gestos, el tema visual (claro/oscuro), el tamaño del teclado y otros parámetros de accesibilidad en un archivo JSON. Permite modificar estos valores sin necesidad de alterar el código principal.

2. **deteccion_manos.py**

Es el módulo de visión por computadora que utiliza MediaPipe para detectar y seguir los movimientos de la mano en tiempo real. Identifica la posición exacta del dedo índice entre 21 puntos anatómicos clave y convierte estos movimientos en acciones como clics o desplazamientos. Implementa algoritmos para filtrar falsos positivos y estabilizar el seguimiento.

3. **ejecutar.py**

Actúa como el núcleo central que coordina todos los componentes. Inicializa la cámara, carga los módulos de detección y teclado, y ejecuta el bucle principal que mantiene actualizada la interfaz. Gestiona el flujo de datos entre la captura de imágenes, el procesamiento de gestos y la visualización.

4. **teclado.py**

Contiene la lógica completa del teclado virtual QWERTY. Define la disposición física de las teclas, maneja las capas intercambiables (letras, números, símbolos) y traduce las coordenadas del dedo en selecciones de caracteres. También gestiona funciones especiales como borrado, espacio y cambio de mayúsculas.

5. visualizacion_teclado.py

Se encarga de renderizar todos los elementos gráficos usando Pygame. Dibuja dinámicamente el teclado con efectos visuales al interactuar, muestra el texto que se va escribiendo en un área dedicada y presenta las sugerencias predictivas en un panel inferior. Aplica los temas de color configurados.

6. modelo_palabras/

Directorio que contiene el sistema de predicción de texto. Incluye el modelo de lenguaje entrenado (normalmente una red LSTM o Transformer), los archivos de vocabulario y los scripts para generar sugerencias inteligentes basadas en el contexto de escritura.

7. sounds/

Almacena el archivo de audio que proporcionan retroalimentación sonora al usuario. Contiene sonido característico para la selección de teclas, acciones de borrado o errores en el reconocimiento de gestos.

8. *pycache*

Directorio generado automáticamente por Python para almacenar versiones compiladas de los módulos y mejorar el rendimiento durante la ejecución. No requiere intervención manual.

Flujo de funcionamiento: El sistema inicia cargando la configuración desde ajustes_sistema.py. La cámara captura imágenes que son procesadas por deteccion_manos.py para identificar gestos. Estas interacciones se traducen en acciones sobre el teclado virtual, cuyo estado se actualiza en teclado.py. La interfaz visible se renderiza mediante visualizacion_teclado.py, mientras que el modelo de

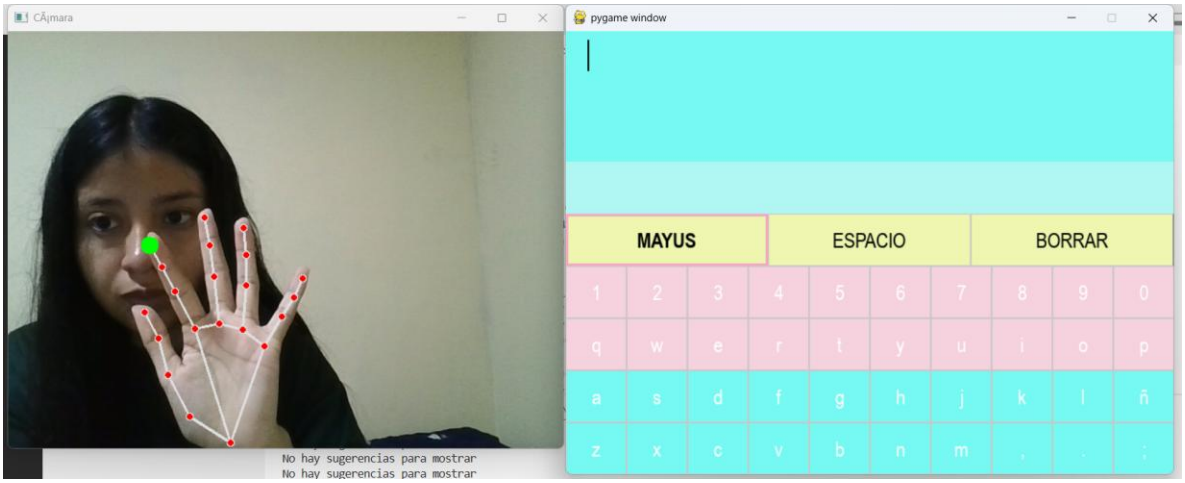
lenguaje en modelo_palabras/ sugiere completados inteligentes. Todo este proceso es coordinado y mantenido en ejecución por ejecutar.py.

Pruebas.

Prueba 1: Funcionamiento de teclas MAYÚSCULAS/minúsculas

Se verificó el correcto cambio entre modos de escritura.

Al seleccionar la tecla MAYUS, el teclado cambió a modo mayúsculas, mostrando todas las letras en versión (Q, W, E, etc.). Y al deseleccionar MAYUS, el teclado regresó a minúsculas (q, w, e, etc.), confirmando que la función de conmutación opera según lo diseñado.

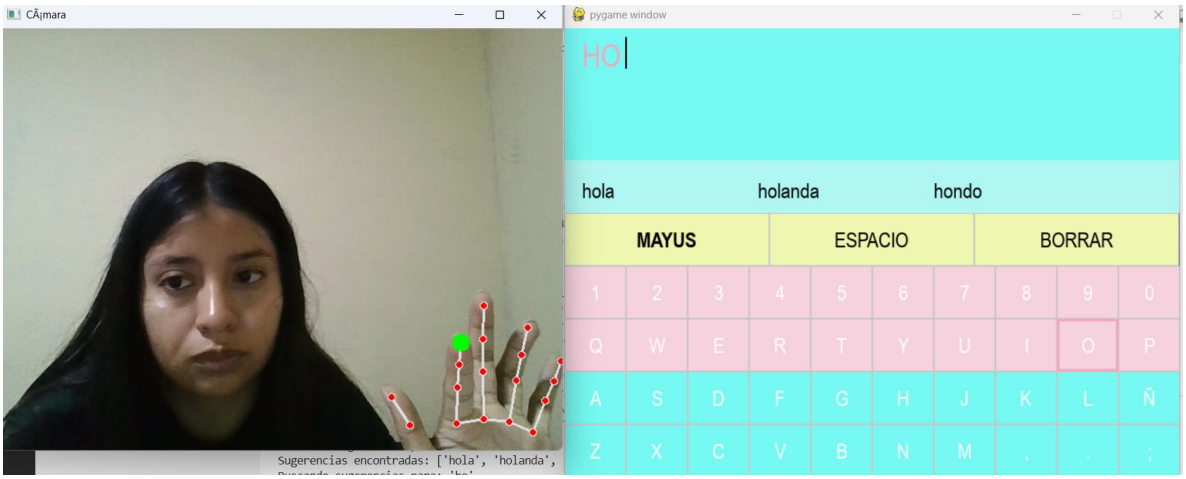


Prueba 2: Panel de sugerencias predictivas

Se validó la aparición contextual de sugerencias.

Al escribir **HO**, el sistema mostró opciones relevantes (hola, holanda, hondo).

Las sugerencias se desplegaron únicamente en modo minúsculas, lo que indica un comportamiento adaptativo al contexto de escritura.

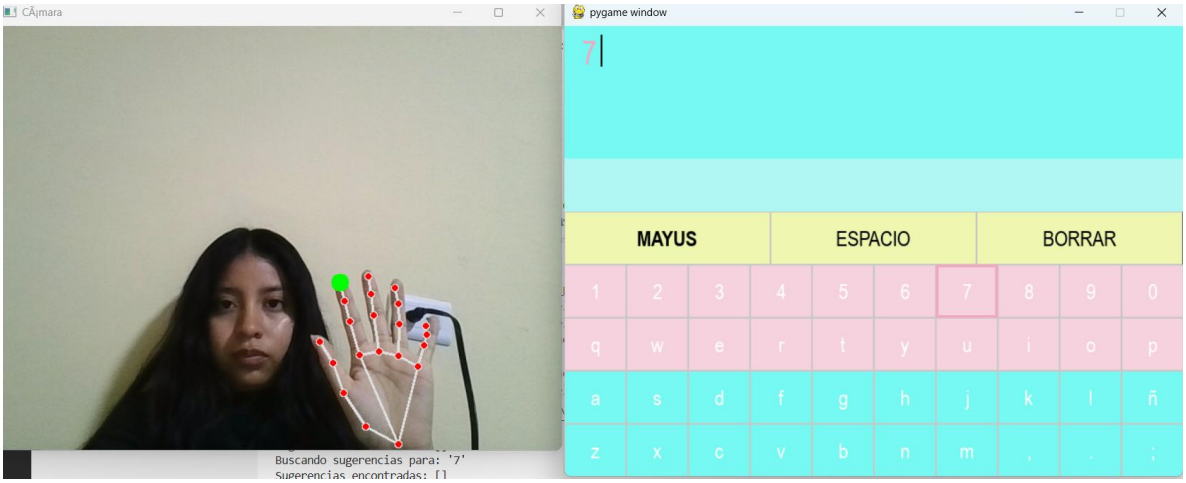


Prueba 3: Funcionamiento del teclado numérico.

Se verifico el funcionamiento de los números.

Se muestra el teclado en modo numérico (números 1-0) seguido de letras en minúsculas (q-p, a-ñ).

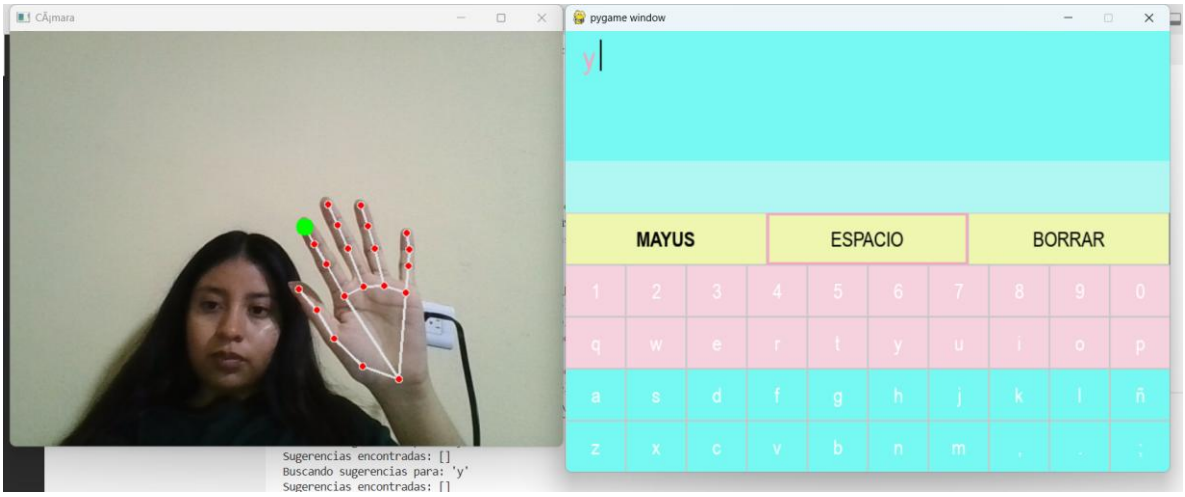
.



Prueba 4. Funcionamiento de la tecla espacio

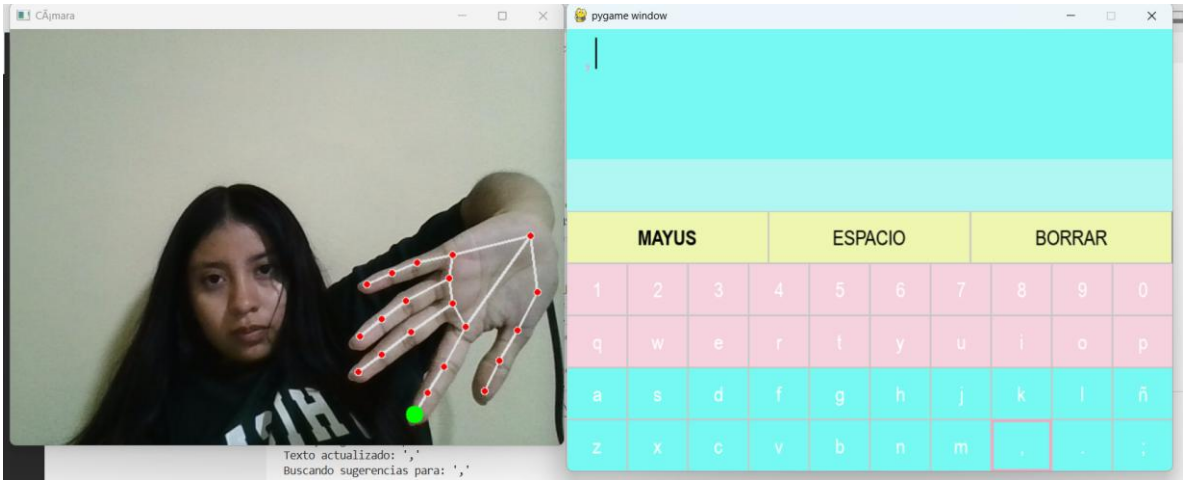
Se seleccionó la tecla ESPACIO mediante gesto manual (movimiento rápido hacia abajo del dedo índice)

El sistema insertó correctamente el carácter de espacio en blanco en todas las situaciones probadas



Prueba 5: Funcionamiento de puntos y comas.

En esta prueba se verificó cómo el sistema maneja los caracteres de puntuación, específicamente los puntos y comas. Se observa que el sistema lo hizo correctamente.



Conclusiones.

Este teclado basado en gestos representa un avance significativo para personas con movilidad limitada, pero aún enfrenta desafíos que afectan su precisión y usabilidad. Actualmente, la detección de movimientos no es completamente precisa, lo que puede generar errores o confusiones, especialmente en condiciones de poca luz o cuando el usuario intenta escribir rápidamente.

Para mejorar su rendimiento, se pueden explorar tecnologías como sensores infrarrojos para una mejor detección en la oscuridad o la integración de comandos de voz para ofrecer una alternativa combinada de gestos y palabras. También se podría implementar un sistema de vibración táctil en accesorios como brazaletes o guantes, brindando una confirmación física cuando se selecciona una tecla.

Además, es fundamental considerar opciones para quienes tienen movilidad muy reducida, como la posibilidad de controlar el teclado mediante pequeños movimientos de cabeza o parpadeos, ampliando su accesibilidad y adaptabilidad. Un sistema de predicción inteligente que aprenda de cada usuario reconozca errores comunes y optimice la escritura también sería una mejora valiosa. también, los símbolos y signos de puntuación podrían incorporarse en un menú más accesible.

En conclusión, este proyecto tiene un gran propósito, pero debe evolucionar hacia una versión más precisa, adaptable y confiable. Cada ajuste y mejora puede marcar la diferencia para quienes dependen de esta tecnología para comunicarse, eliminando barreras y facilitando una expresión más fluida y sin limitaciones.