



Universidad Veracruzana
FACULTAD DE NEGOCIOS Y TECNOLOGIAS

PROYECTO FINAL INTELIGENCIA ARTIFICIAL

Desarrollo de un prototipo que permita la entrada de texto de manera eficiente mediante una interfaz visual que se gestione por medio del seguimiento ocular.

Estudiante: Julián Francisco Ruiz
Ramírez
LTIO 601


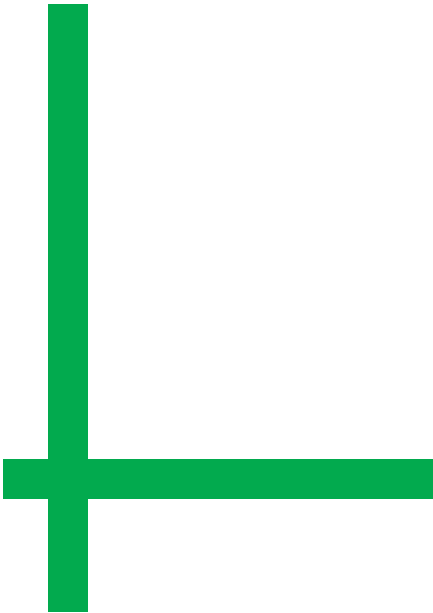
Profesor: Jesús Leonardo López
Hernandez



Indice



Introducción	3
Arquitectura	4
Desarrollo y prototipo	5
Implementación y código	6
Pruebas del sistema	7
Conclusiones	8



Introducción

El acceso a la tecnología es un derecho fundamental, sin embargo, millones de personas con discapacidad motriz enfrentan limitaciones diarias al intentar interactuar con dispositivos digitales. Entre las herramientas más básicas de comunicación se encuentran los teclados físicos, los cuales exigen una motricidad fina que no todas las personas poseen.

Diversas condiciones como la parálisis cerebral, esclerosis lateral amiotrófica (ELA), lesiones medulares o enfermedades neurodegenerativas pueden impedir el uso de teclados convencionales. Para estas personas, un sistema de entrada alternativo que permita escribir o seleccionar texto sin utilizar las manos representa una mejora significativa en su autonomía, comunicación y calidad de vida.

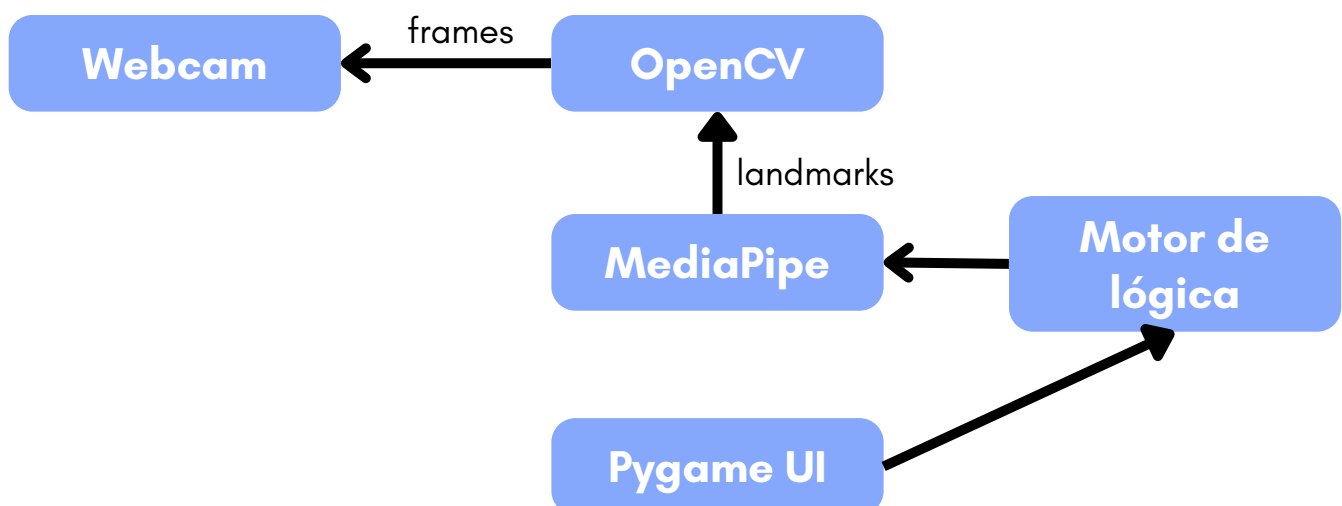
Este proyecto tiene como objetivo desarrollar un prototipo de teclado virtual que pueda ser controlado únicamente mediante el movimiento de los ojos y el parpadeo. Para lograrlo, se emplean tecnologías de visión por computadora y gráficos interactivos que permiten que una persona con movilidad reducida escriba texto observando una pantalla y fijando la mirada sobre letras, sin necesidad de hardware adicional especializado.

Arquitectura

La arquitectura del sistema se basa en cuatro componentes principales que interactúan para brindar una experiencia accesible y responsiva:

1. **Captura de Video (OpenCV):** Utiliza una cámara convencional para obtener video en tiempo real.
2. **Seguimiento Facial y Ocular (MediaPipe):** Detecta el rostro del usuario y localiza puntos clave del iris (landmark 468) para determinar la dirección de la mirada.
3. **Interfaz Gráfica (Pygame):** Dibuja el teclado virtual, detecta las zonas activas (teclas y sugerencias), y visualiza el cursor que representa la mirada.
4. **Motor de Lógica:** Realiza la calibración inicial, convierte coordenadas de cámara a pantalla, maneja el dwell-time, autocompletado y limpieza automática del texto.

A continuación, se muestra el diagrama de flujo del sistema:



Desarrollo y prototipo

El prototipo fue desarrollado en Python utilizando librerías abiertas que garantizan compatibilidad multiplataforma. Se implementó una calibración inicial donde el usuario debe mirar al centro de la pantalla para establecer una referencia de su mirada en reposo.

El teclado virtual se diseñó con teclas grandes (80x80 píxeles) y colores pastel para facilitar la visibilidad y la selección mediante mirada. La fuente 'Comic Sans MS' se eligió por ser amigable y de alta legibilidad.

La lógica de selección se basa en el concepto de '**dwell time**', donde el usuario debe mantener su mirada fija sobre una tecla durante al menos 1.5 segundos para activarla. Esto evita selecciones accidentales. Las palabras escritas se muestran en pantalla y, después de 5 segundos sin actividad, el sistema limpia el texto automáticamente.

Adicionalmente, se implementó un sistema de autocompletado que sugiere hasta tres palabras comunes basadas en el prefijo ingresado. Estas sugerencias se presentan como botones que también pueden ser seleccionados con la mirada.

Implementación y código

El código se divide en varias funciones clave que organizan la lógica del sistema:

- **calibrate(cap):** Captura múltiples frames mientras el usuario mira al centro, promediando las coordenadas del iris para establecer un punto de referencia.
- **eye_to_screen(gx, gy, cx, cy):** Convierte la posición relativa de la mirada en coordenadas adaptadas a la interfaz gráfica.
- **predict(prefix, words):** Utiliza la librería difflib para encontrar coincidencias aproximadas a partir del texto ingresado y ofrecer sugerencias.
- **draw_keyboard():** Dibuja todas las teclas con estilos visuales atractivos y prepara sus zonas activas.
- **Lógica dwell-time:** Cuando una tecla o sugerencia es 'mirada' por más de 1.5 segundos, se confirma su selección.
- **Autoborrado:** Un temporizador se reinicia cada vez que el usuario escribe. Si pasan 5 segundos sin interacción, el texto se borra automáticamente.

Pruebas del sistema

Se realizaron múltiples pruebas funcionales para verificar la precisión y usabilidad del sistema:

Caso 1 – Selección de carácter:

- **Acción:** El usuario fija la mirada sobre la tecla 'T'.
- **Resultado esperado:** Después de 1.5 segundos, la letra 'T' aparece en el área de texto.

Caso 2 – Escritura de una palabra:

- **Acción:** El usuario selecciona consecutivamente las letras 'H', 'O', 'L', 'A'.
- **Resultado:** Se forma la palabra 'HOLA' en pantalla.

Caso 3 – Sugerencia de palabras:

- **Acción:** El usuario escribe 'TI'. El sistema sugiere 'TIEMPO'.
- **Resultado esperado:** Al mirar la sugerencia durante 1.5 s, la palabra completa se inserta.

Caso 4 – Autolimpieza:

- **Acción:** El usuario escribe una palabra y luego deja de interactuar.
- **Resultado:** A los 5 segundos, el campo de texto se borra automáticamente.

Conclusiones

El prototipo demuestra que es posible implementar una interfaz accesible, responsiva y económica utilizando únicamente software libre y una cámara web común. Este teclado virtual permite que personas con movilidad reducida puedan escribir mediante el movimiento de sus ojos, sin necesidad de dispositivos costosos o invasivos.

Limitaciones actuales:

- Precisión variable dependiendo de la calidad de la cámara y la iluminación.
- No cuenta con calibración multipunto (actualmente solo un punto central).
- El predictor de palabras es simple y no se adapta al contexto o historial del usuario.

Mejoras futuras:

- Implementar calibración por esquinas para mejorar la correspondencia mirada-pantalla.
- Reemplazar el predictor con un modelo de lenguaje entrenado para cada idioma.
- Añadir retroalimentación por voz para usuarios con discapacidad visual.
- Habilitar compatibilidad con controladores de voz, interfaces cerebro-computadora, o asistentes virtuales.