

Experiencia educativa:

INTELIGENCIA ARTIFICIAL

**Proyecto Final: Inteligencia Artificial**

Integrante:

Sergio Cariño Hernández

**Especificación Técnica: Prototipo de Teclado por Gestos "Mano-Letra"**

**1. Introducción**

* **Problemática**:

Imaginemos el día a día de una persona con movilidad limitada en las manos. Actos tan simples como enviar un mensaje, buscar información en la web o participar en redes sociales se convierten en tareas monumentales. Las herramientas estándar no fueron diseñadas para ellos, creando una "barrera digital" que los aísla. Nuestro proyecto nace de una pregunta fundamental: ¿Cómo podemos demoler esa barrera usando tecnología que ya está al alcance de todos, como una simple cámara web?

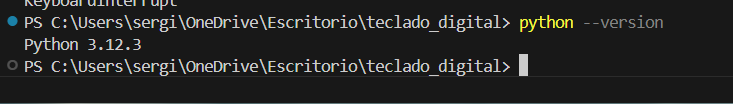
* **Justificación:**

Este proyecto no busca reinventar la rueda, sino hacerla accesible para todos. Proponemos una solución de software que transforma cualquier cámara web estándar en una interfaz de texto avanzada. Nuestra propuesta de valor se basa en tres pilares:

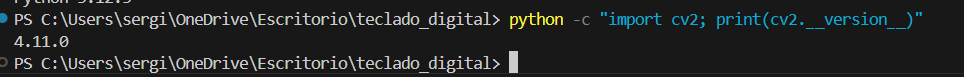
1. **Accesibilidad Radical:** Sin necesidad de hardware costoso o especializado.
2. **Interacción Intuitiva:** Se utiliza el gesto universal de señalar para una curva de aprendizaje mínima.
3. **Eficiencia Inteligente:** Integramos un motor de predicción de texto basado en Machine Learning para reducir el esfuerzo y aumentar la velocidad de escritura, haciendo la comunicación más fluida y menos fatigante.

**2. Arquitectura**

* **Pila Tecnológica (Tech Stack):**
  + **Lenguaje de Programación:** Python 3.9+



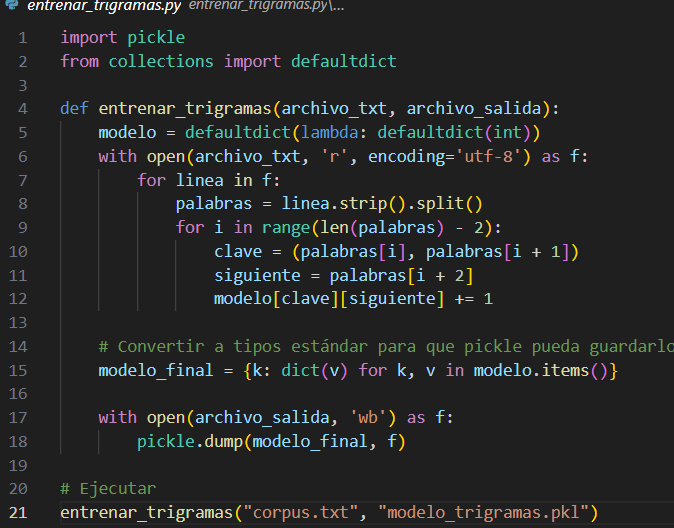
* + **Visión por Computadora:**
    - **OpenCV:** Para la captura, manipulación y renderizado de video en tiempo real.



* + - **MediaPipe (de Google):** Para la detección robusta y de alto rendimiento de los 21 puntos clave de la mano.



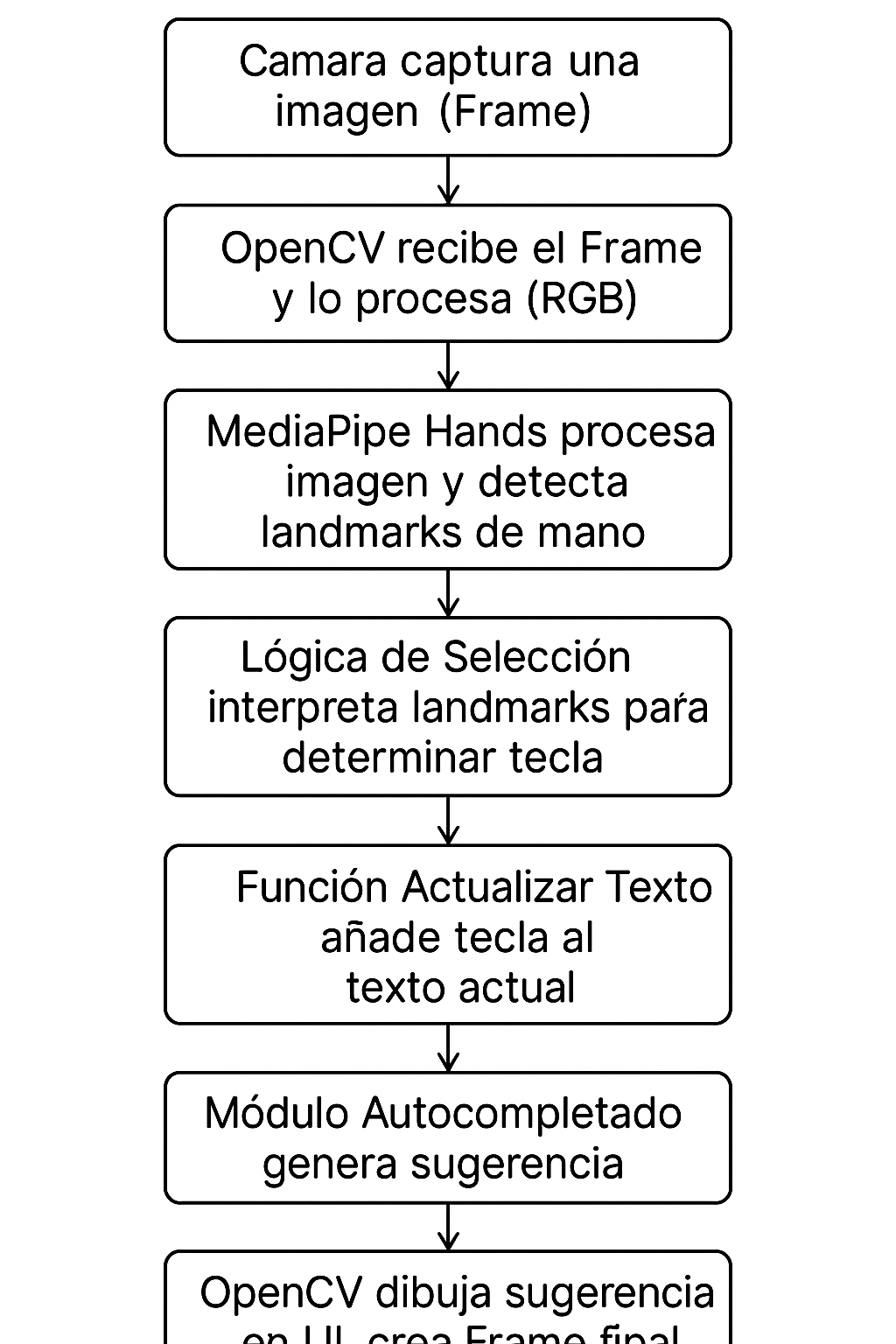
* + **Modelo de Lenguaje:** Modelo N-Gram (Trigramas) entrenado a medida y serializado con pickle para una carga rápida.



**Librerías Auxiliares:** Pygame para la retroalimentación auditiva, NumPy para cálculos vectoriales (no incluido en tu código, pero sería una mejora).

* Diseño del Sistema (Diagrama de Componentes):

El sistema se concibe con una arquitectura modular para facilitar su mantenimiento y escalabilidad.



**3. Desarrollo y Prototipo**

* Flujo de Proceso (De la Idea a la Ejecución):

El desarrollo se abordó en fases iterativas:

* 1. **Fase de Datos:** Se creó un script (entrenar\_trigramas.py) para procesar un corpus de texto y generar nuestro modelo predictivo.
  2. **Fase de Visión:** Se implementó el núcleo de la aplicación, logrando un seguimiento estable de la mano y el puntero del dedo índice.
  3. **Fase de Interfaz:** Se desarrolló la lógica para dibujar el teclado y la mecánica de selección por tiempo de permanencia ("dwell time").
  4. **Fase de Integración:** Se conectó el modelo de predicción a la aplicación principal, permitiendo que las sugerencias se mostraran en tiempo real.
  5. **Fase de Gestos Adicionales:** Se añadió la funcionalidad de "palma abierta" para el borrado rápido como una mejora a la experiencia de usuario.
* Descripción Funcional del Prototipo:

El prototipo actual es una aplicación de escritorio que, al ejecutarse, permite al usuario:

* 1. Controlar un puntero en pantalla moviendo su dedo índice.
  2. Escribir caracteres al mantener el puntero sobre una tecla por un segundo.
  3. Recibir hasta tres sugerencias de palabras para autocompletar, seleccionables de la misma manera.
  4. Borrar todo el texto de forma instantánea mostrando la palma de la mano a la cámara.

**4. Implementación**

* Módulos de Código Clave:

El proyecto se estructura en torno a lógicas modulares, incluso si están en el mismo archivo.

* **Módulo de Entrenamiento (entrenar\_trigramas.py):**
* **Función:** entrenar\_trigramas()
* **Explicación:** Este script es el responsable de la "inteligencia" predictiva. Lee un texto plano, lo descompone en secuencias de tres palabras (trigramas) y cuenta sus apariciones. El resultado es un modelo probabilístico que sabe qué palabra es más probable que aparezca después de un par de palabras dadas. Se guarda con pickle para no tener que re-entrenarlo cada vez.

**Módulo de Interacción (Lógica en el bucle while):**

* **Función:** detectar\_gesto\_palma\_abierta() y la lógica de dwell time.
* **Explicación:** Este es el cerebro en tiempo real. Interpreta los datos de MediaPipe. La función de palma abierta es un claro ejemplo de reconocimiento de gestos basado en reglas (posiciones relativas de los dedos). La lógica de "dwell time" (if time.time() - tiempo\_inicio > 1:) actúa como un temporizador que dispara una acción, simulando un clic sin contacto físico.

**Módulo de Predicción (Lógica de consulta):**

* **Función:** predecir\_palabra()
* **Explicación:** Este módulo actúa como un puente entre la interacción del usuario y el modelo entrenado. Toma la palabra que se está escribiendo, la usa como un "prefijo" y busca en el modelo todas las posibles terminaciones, presentando las más relevantes.

**5. Pruebas**

Plan de Verificación y Validación:

Para asegurar la calidad del prototipo, se propone un plan de pruebas en tres niveles:

**Pruebas Unitarias (Sugeridas para el futuro):**

* **Objetivo:** Probar cada función de forma aislada.
* **Ejemplo:** Crear un test que llame a predecir\_palabra("PROG", modelo\_mock) y verifique que devuelve ["PROGRAMA"]. Probar detectar\_gesto\_palma\_abierta con coordenadas de landmarks predefinidas para una palma abierta y una cerrada.

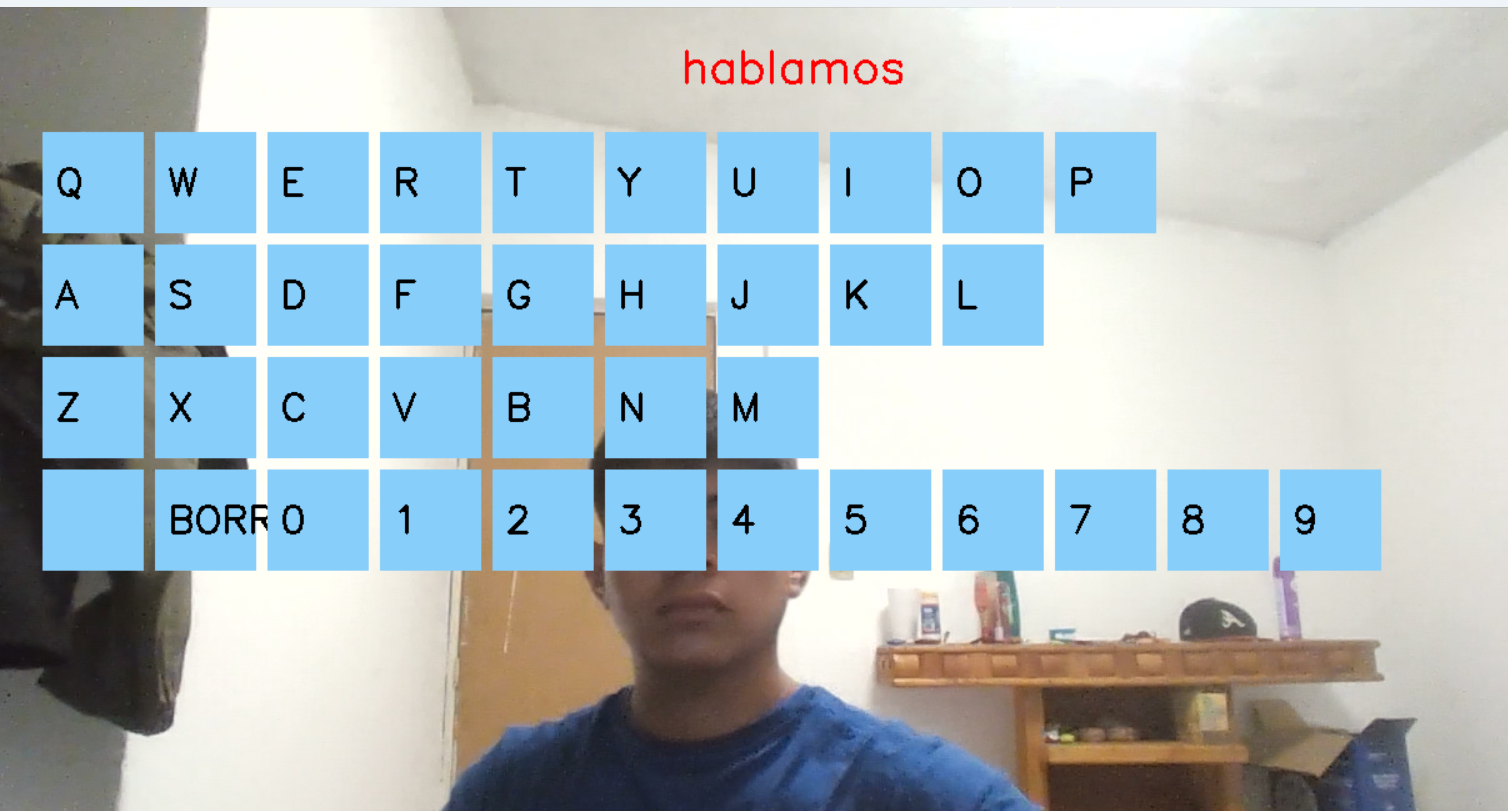
**Pruebas de Integración (Realizadas):**

* **Objetivo:** Asegurar que los módulos funcionan juntos correctamente.
* **Caso de Prueba:** Escribir "HOLA". Verificar que el texto se actualiza en la GUI, el sonido se reproduce y la función de predicción se activa para sugerir la siguiente palabra.

**Pruebas de Usabilidad (Cualitativas):**

**Objetivo:** Evaluar la experiencia del usuario.

* **Caso de Prueba:** Un usuario intenta escribir la frase "HOLA COMO ESTAS". Se mide el tiempo y se anota cualquier punto de frustración (ej., "la selección por tiempo es muy lenta", "el puntero tiembla mucho").



**6. Conclusiones**

* Análisis de Resultados y Lecciones Aprendidas:

El prototipo demuestra exitosamente que es posible construir una herramienta de comunicación asistida efectiva y de bajo costo utilizando hardware común y librerías de código abierto. La lección más importante es el balance entre la complejidad técnica y la usabilidad: la predicción de texto, aunque simple, añade un valor inmenso a la experiencia, mientras que la mecánica de selección por "dwell time", aunque fácil de implementar, es un cuello de botella para la fluidez.

* Próximos Pasos y Hoja de Ruta (Roadmap):

Este prototipo es una base sólida. El plan de desarrollo futuro incluye:

**Corto Plazo (Próximo Sprint):**

* **Reemplazar "Dwell Time":** Implementar un gesto de "clic por pellizco" (unir pulgar e índice) para una selección instantánea.
* **Refactorizar:** Separar la lógica en clases (ej. Keyboard, GestureDetector, Predictor).

**Medio Plazo (Próximo Trimestre):**

* **Mejorar el Modelo de Lenguaje:** Experimentar con modelos más avanzados (LSTMs) para predicciones contextuales.
* **Personalización:** Permitir al usuario ajustar el tamaño del teclado y los tiempos de respuesta.

**Largo Plazo:**

* **Pruebas de Campo:** Realizar pruebas con usuarios de la comunidad de discapacidad motriz para obtener retroalimentación real.
* **Añadir Funciones:** Incorporar control por voz para cambiar de modo o dictar palabras complejas.