

Manual de programador



Handspeak-V1.0

23/10/2024

Desarrollo De Software

Índice

Portada-----	1
Índice-----	2
Introducción-----	3
Entorno de desarrollo-----	4
Código fuente-----	5
Realizar cambios al código-----	5
Explicación del código fuente-----	7
Mas información-----	11

Introducción

Handspeak está desarrollado utilizando el lenguaje de programación Python en su 100%, hemos decidido utilizar este lenguaje gracias a su gran cantidad de bibliotecas que nos permiten tener las herramientas necesarias para poder crear, diseñar, desarrollar y entrenar modelos de aprendizaje de forma efectiva, hemos creado un website de fácil acceso al público para que puedan tener la información completa de las funcionalidades de nuestro programa y puedan utilizarlo desde cualquier tipo de computadoras, hemos diseñado una interfaz que sea capaz de ser manejada de forma sencilla y práctica.

Para lograr esto hemos mantenido la interfaz lo más practica posible donde las personas pueden ejecutar el programa y de forma inmediata este empieza a funcionar y cada usuario puede cerrar el programa de forma sencilla.

Útil para ambientes educacionales y con fines de traducción, en este manual describiremos la documentación técnica de Handspeak, una herramienta diseñada para facilitar la interpretación del lenguaje de señas. Aquí encontrarás información detallada sobre la instalación, el acceso al código fuente, y las herramientas utilizadas para su desarrollo. Asimismo, se proporcionan instrucciones para contribuir con cambios al código.

Entorno de desarrollo

1-**Editor de texto**: Se recomienda Visual Studio Code para facilitar el desarrollo.

2-**Python**: Versión recomendada: 3.x o superior.

3-**TensorFlow**: Librería para machine learning

4-**Mediapipe**: Librería para el reconocimiento de gestos y manos

5-**Google Text to Speech (gTTS)**: Para la conversión de texto a voz

6-**Pygame**: Librería para el desarrollo de interfaces gráficas y multimedia

7-**OpenCV**: Librería para procesamiento de imágenes

8-**Git**: Para el control de versiones

A continuación mostraremos como instalar cada una de estas librerías en Visual Studio Code.

1-pip install tensorflow

2-pip install mediapipe

3-pip install gtts

4-pip install pygame

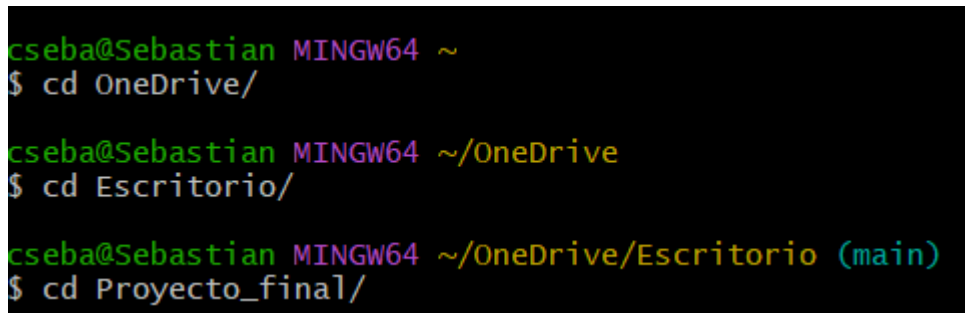
5-pip install opencv

6-<https://kinsta.com/es/base-de-conocimiento/instalar-git/>

Como obtener el código fuente

Para el desarrollo de la aplicación se ha utilizado un repositorio Git hospedado en GitHub. Para obtener una copia de este hay que proceder de la siguiente manera:

1. Abrir la terminal Git Bash.
2. Desplazarse al directorio donde se desee copiar el repositorio (utilizando el comando `cd`).



```
cseba@Sebastian MINGW64 ~  
$ cd OneDrive/  
  
cseba@Sebastian MINGW64 ~/OneDrive  
$ cd Escritorio/  
  
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio (main)  
$ cd Proyecto_final/
```

3. Introducir el siguiente comando: `git clone https://github.com/sebaas256/Proyecto_final.git` .
4. Se iniciará la descarga del repositorio, cuando finalice se dispondrá de una copia completa de este.

Realizar cambios a el código

Para poder realizar cambios el código directamente desde el repositorio tendrás que crear otra rama utilizando la terminal de Git o con la GUI de Git. Luego de crear la rama pushear tus cambios al repositorio.

Pasos para crear una rama paralela en nuestro repositorio.

- 1-Primero ubícate en el archivo donde este el código con el comando `cd Proyecto_final`.
- 2-Luego crear la rama con el comando `git checkout -b rama-name`.
- 3-Con el comando `git branch` podemos ver todas nuestras ramas.

4-Con el comando git checkout -b nombre-rama podemos cambiar de una rama a otra.

```
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git checkout -b example
Switched to a new branch 'example'

cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (example)
$ git branch
* example
  main
  test
  website

cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (example)
$ git checkout website
Switched to branch 'website'
Your branch is up to date with 'origin/website'.
```

5-Con el comando git branch -D podemos borrar una rama remota de git.

```
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git branch -D example
Deleted branch example (was 6d10439d).
```

6-Para poder agregar todos tus cambios realizados utiliza el comando git add si la respuesta de git add no dio nada indica que se agregó correctamente.

```
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git status
On branch website
Your branch is up to date with 'origin/website'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   inicio.html
        modified:   secciones.html

no changes added to commit (use "git add" and/or "git commit -a")

cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git add .
```

7-Luego realiza un commit para poder guardar tus cambios con el comando git commit -m "mensaje del commit-example".

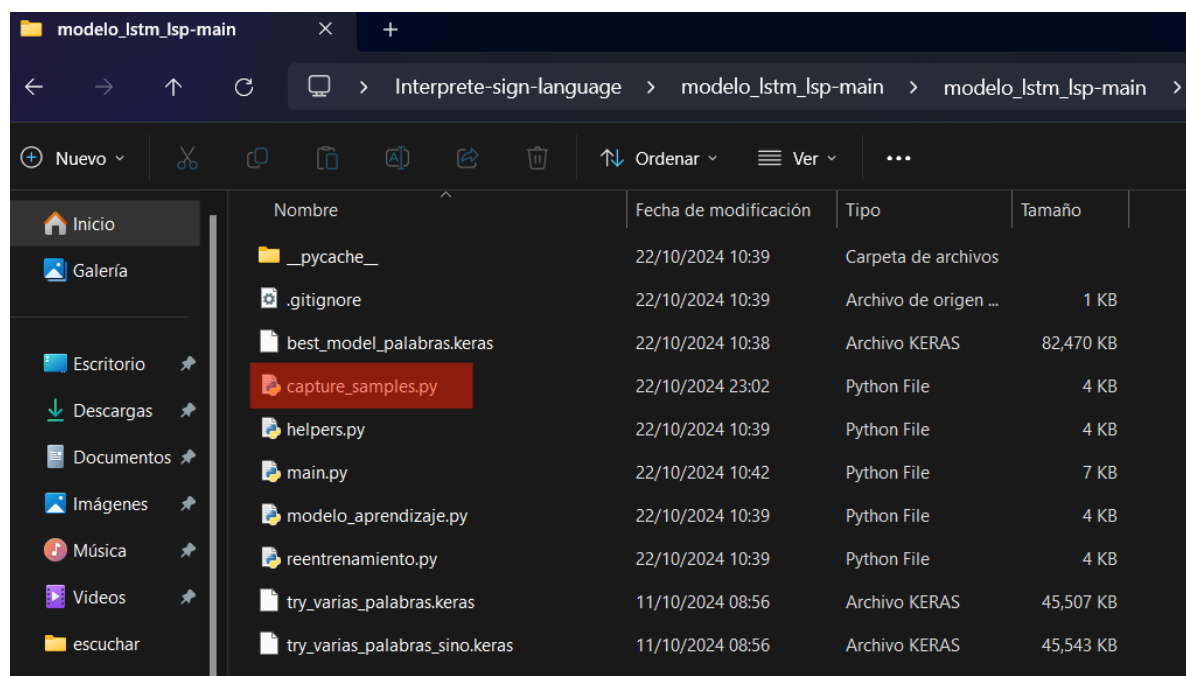
```
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git commit -m "fix mistakes"
[website 6d10439d] fix mistakes
 2 files changed, 1 insertion(+), 2 deletions(-)
```

8-Luego realiza push a tu commit apuntando a tu Branch con el siguiente comando git push -u origin rama-name.

```
cseba@Sebastian MINGW64 ~/OneDrive/Escritorio/Proyecto_final (website)
$ git push -u origin website
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 362 bytes | 362.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/sebaas256/Proyecto_final.git
   88610685..6d10439d website -> website
branch 'website' set up to track 'origin/website'.
```

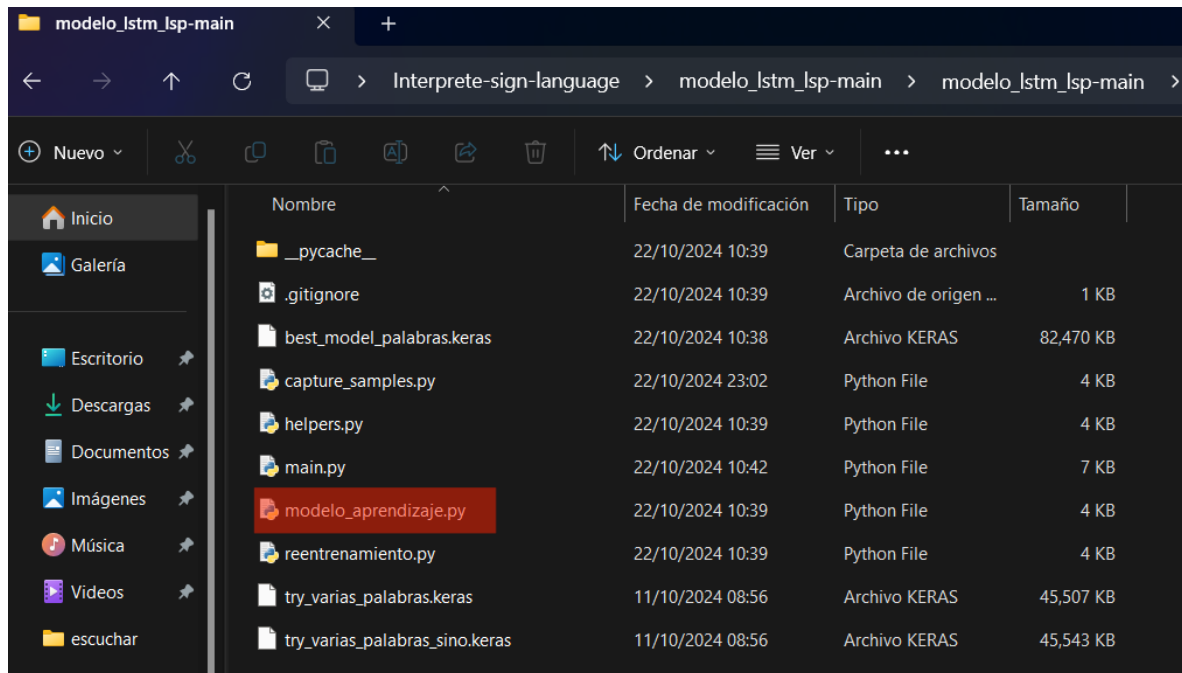
Explicación del código fuente

Capture_samples.py



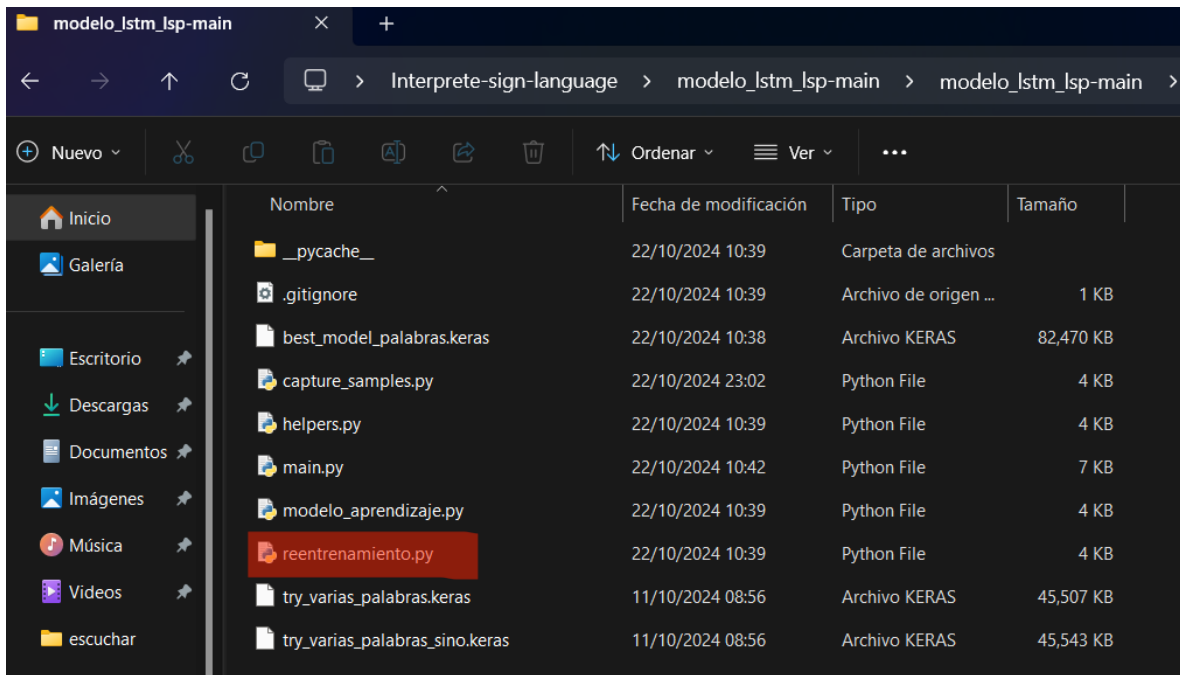
Script para capturar imágenes de gestos en tiempo real utilizando MediaPipe. Las imágenes se almacenan en directorios específicos para cada gesto.

Modelo_aprendizaje.py



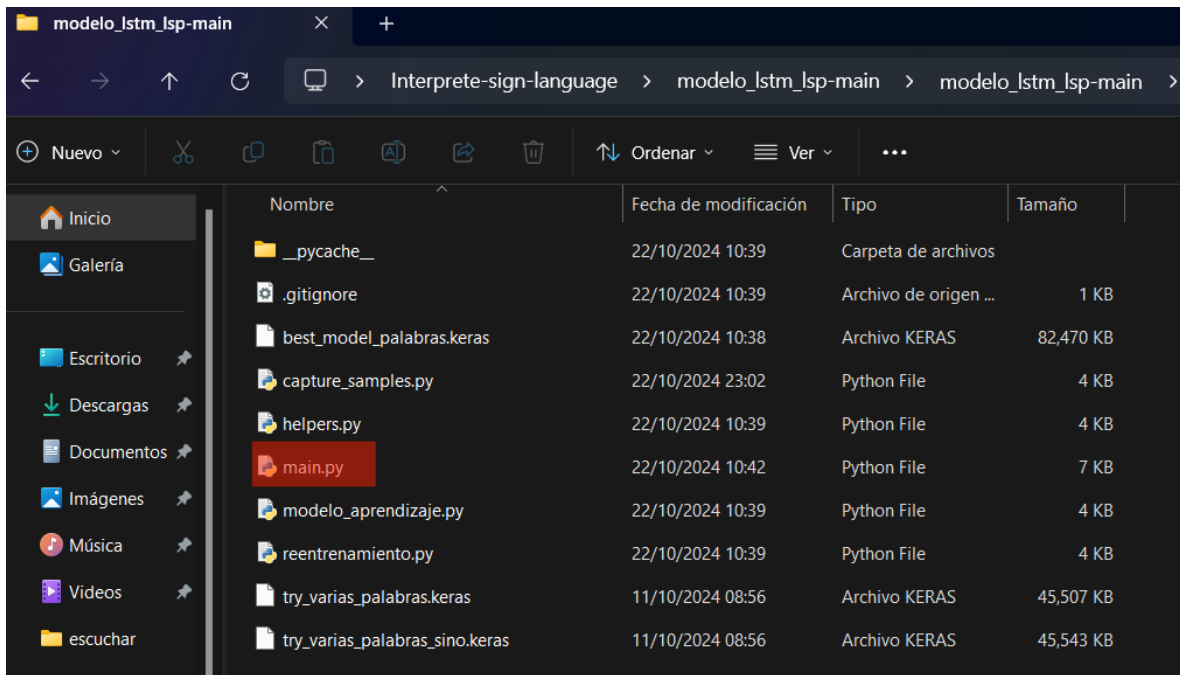
Este script esta programado para poder crear nuestra red neuronal y poderla entrenar con cualquier gesto que quieras, ten en cuenta que para el entrenamiento nosotros utilizamos un modelo preentrenado (MobileNetV2) que funciona para el reconocimiento de imágenes, este puedes cambiarlo por cualquiera que quieras

Reentrenamiento.py



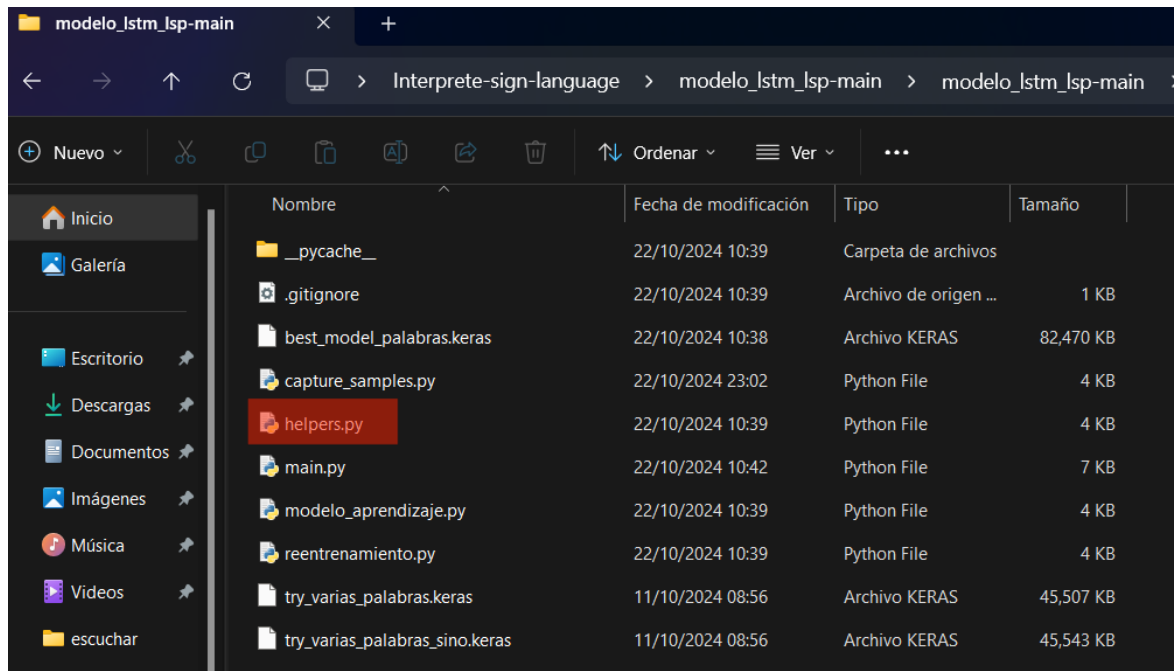
Este script funciona para poder reentrenar redes neuronales con imágenes nuevas ya sea de cualquier mismo gesto o algún nuevo gesto que quieras agregar al igual que en el entrenamiento utilizamos MobileNetV2.

Main.py



Este archivo es el encargado de procesar el video de la cámara, reconocer los gestos y mostrar la salida en tiempo real. Utiliza modelos preentrenados para realizar la predicción de gestos, convertirlos en texto y voz.

Helpers.py



Proporciona funciones para visualizar los puntos clave de las manos, dibujar barras de confianza, y mostrar texto en la pantalla durante la detección de gestos.

Mas informacion

Puedes obtener más información acerca del programa en nuestra pagina web entrando al siguiente link: handspeak.site