Hand Gestures Translator

- Application:

  . How does the app work?

  The app was created using MIT App inventor.

- **Step 1:** We enter the IP address that the app will connect to with the ESP.
- **Step 2:** The app will automatically connect to the ESP if the link is working, then another screen will appear.
- **Step 3:**
  - At the bottom of the screen, there is an "End Connection" button to disconnect.
  - Above it, there is an input box where we can type. When the "Translate" button is pressed, the text will be translated into images. The purpose of this step is to assist anyone who doesn't know sign language in communicating with people with disabilities.

    . If the input box is empty and the button is pressed, it will copy the text from the box above.
- **Step 4:** The function of the box above is to receive the characters processed by the ESP. This box cannot be modified except by the ESP.
- **Step 5:** Next to it, there is a button called "Speech" that converts the text in this box into speech.

- **ESP:**

  1. We used WebSocket protocol to get the character translated from the AI model and the ESP was a WebSocket server and python acted as a WebSocket client and the character was received as bytes so we had to do type casting to convert it to a char variable again.
  2. The character is also sent to the MIT application by WebSocket protocol but in this time the ESP acted as a WebSocket client and the application as a WebSocket server as WebSocket protocol is bidirectional, so it is easy to get data and send it.
  3. We used UART protocol to link the ESP with the Arduino and sent it the character by the serial.

Sign Language Recognition System

Overview

This project is a real-time sign language recognition system using computer vision and machine learning. The system captures hand gestures via webcam, processes the input using the MediaPipe library, and predicts the corresponding sign (Arabic, English, Numbers) using a trained classification model (RandomForest Classifier). It provides a user-friendly GUI and includes voice output and device control features.

Project Structure

- GUI Framework: Tkinter for (Arabic, English , Numbers).

- Hand Tracking: MediaPipe

- Model Inference: Trained models using pickle

- Speech Output: pyttsx3 , gtts ,pygame

- External Control: Communication via WebSocket (WSClient)

- Video Input: cv2 (OpenCV)

Key Functionalities

Real-Time Gesture Detection

Uses webcam to detect a single hand in real time.
- Extracts 21 landmarks per hand with MediaPipe Hands.
- Normalizes the coordinates to form a feature vector of 42 values.
- Passes the vector to a pre-trained model for prediction.

Prediction Handling

- Recognizes individual letters, 'SPACE', 'DEL', and 'nothing'.
- Accumulates characters into a sentence with auto-spacing and deletion support.
- Displays prediction and sentence dynamically.

Graphical Interface

- Displays the live video stream with hand landmark overlays.
- Neon-bordered canvas for visual appeal.
- Buttons to Reset the sentence or Speak it aloud.

Text-to-Speech (TTS)

Uses pyttsx3 and gtts to convert the final sentence into speech.

Device Communication

Sends recognized characters to an external device (likely ESP32/ESP8266) using WebSockets (WSClient).
Special commands:
- "*": Reset command
- " ": Space
- "#": Delete

Keyboard Shortcuts

- Spacebar: Add space
- Backspace: Delete character
- Enter: Speak sentence
- Escape: Exit application