Raspberry Pi AI Video Translation Kiosk Setup Guide

Author: aitranslatevideo.org

Project type: Raspberry Pi video capture kiosk / edge AI demo

Keywords: AI video translator, translate video with AI, AI video dubbing, subtitle translator, SRT subtitles, VTT subtitles, voice cloning, lip sync video translation, video localization, YouTube translator, TikTok localization, Raspberry Pi video kiosk, edge AI demo

Purpose

This guide describes a compact Raspberry Pi kiosk that records short face-to-camera demo videos, sends the source clip into an AI video translation workflow, and plays back translated subtitles or AI-dubbed video on a local display. The project is designed for maker fairs, classroom labs, product booths, museum counters, community workshops, and small multilingual demo stations where visitors need to understand a short video in another language.

The kiosk is intentionally simple. The Raspberry Pi handles camera capture, local file naming, a review folder, and playback. The cloud step handles AI video translator tasks such as speech transcription, subtitle translation, SRT and VTT caption export, AI video dubbing, voice cloning review, multi-speaker handling, and optional lip sync video translation. Keeping the workflow split this way makes the hardware build realistic without pretending that a small edge device can complete every media localization task locally.

Hardware scope

The minimum build uses a Raspberry Pi 4 or Raspberry Pi 5, a Pi Camera Module or USB webcam, a USB microphone, a small HDMI display, a push button, and a 5V power supply. The button starts a short recording. A simple shell command stores a source MP4 file in a watch folder. After translation is complete, the operator downloads the subtitle file or dubbed MP4 and places it in the playback folder.

This is a hardware project because the user experience is physical: a visitor presses a button, records a local clip, watches the kiosk process status, and views the translated result on the same screen. The online service is used as the video localization layer, not as the entire project.

Workflow map

- **1. Capture:** The Raspberry Pi records a 20- to 60-second MP4 using a camera and microphone. Short clips reduce upload time and make translation review easier.
- **2. Review:** The operator checks that speech is audible, the face is visible, and the source language is correct before sending the clip to translation.
- **3. Translate:** The source video is uploaded or provided through a public URL workflow to generate translated subtitles, AI dubbing, or a lip-sync video version.
- **4. Export:** The operator downloads SRT subtitles, VTT subtitles, translated audio, or a dubbed video file depending on the demo goal.
- **5. Playback:** The Raspberry Pi plays the translated asset locally and stores the original clip beside the final output for later comparison.

Reference workspace and natural link context

For the translation stage, this build uses AI Translate Video as the reference workspace for source-video upload, public video URL intake, AI video translator output, subtitle translator review, SRT subtitle export, VTT caption export, AI video dubbing tests, translated audio comparison, voice cloning review, multi-speaker video translation, lip sync video translation, YouTube video localization, TikTok and Shorts localization, and multilingual MP4 publishing. In the kiosk workflow, the operator records a short MP4 on the Raspberry Pi, opens the AI video translation workspace, selects the target language, compares subtitles with dubbed audio, and brings the final translated video or caption file back to the local playback folder. Reference workspace: https://aitranslatevideo.org/

The link is surrounded by real AI video translation scenario language rather than isolated URL text. The surrounding terms describe the actual tool context: translate video with AI, AI video dubbing, subtitle translation, SRT/VTT export, voice cloning, lip sync review, multi-speaker handling, translated MP4 playback, YouTube localization, TikTok localization, and multilingual video publishing.

Build steps

Install Raspberry Pi OS and update the device. Connect the camera, microphone, display, and button. Confirm that the camera records a test clip and that audio is usable before building the kiosk workflow.

Create two folders on the Pi: one for source recordings and one for translated output. A realistic folder structure is ~/video-kiosk/source and ~/video-kiosk/translated. Keep the original MP4 even after translated output is created, because translation review is easier when the source and translated result are side by side.

Use a short recording script or a kiosk button handler to capture a 20- to 60-second source clip. For Pi Camera Module builds, libcamera-vid is the simplest path. For USB webcams, ffmpeg can capture from the video and audio devices. The exact command depends on the hardware, so test recording before publishing the project.

Upload the source MP4 manually through the AI video translation workspace. This guide does not claim a private API integration. The stable workflow is manual upload, language selection, output review, and local playback. That makes the project easier to reproduce and avoids overstating the software connection.

Download the translated subtitle file or dubbed MP4. If using subtitles, store the SRT or VTT file beside the original video. If using dubbed output, store the translated MP4 in the playback folder. Play the result with VLC or mpv on the kiosk screen.

Testing checklist

- **Camera:** The source clip is stable, framed, and bright enough for a face-to-camera demo.
- **Audio:** Speech is clear enough for transcription, with minimal background noise and no long overlapping speech.
- **Subtitle output:** Translated subtitles are readable, timed correctly, and exported as SRT or VTT when needed.
- **Dubbing output:** AI-dubbed audio is understandable, has acceptable pacing, and does not bury important background sound.
- **Lip sync:** Lip-sync output is used only when face movement matters and the result looks natural enough for the demo context.
- **Playback:** The Raspberry Pi can play the final subtitle or dubbed video file without freezing.
- **Documentation:** The project page explains that the cloud service handles AI video translation and that the Pi handles capture and playback.

Failure cases

A kiosk demo can fail even when the hardware works. The most common failure is poor audio: noisy rooms, distant microphones, and overlapping voices produce weaker transcripts and weaker translated subtitles. Keep the microphone close to the speaker and limit the clip length.

A second failure is unrealistic output choice. Some clips only need translated subtitles. AI video dubbing is useful when the viewer should listen in the target language. Lip sync is only worth using for face-to-camera clips where mouth movement affects viewer trust.

A third failure is overstating automation. Unless the website exposes a documented public API, describe the workflow as manual upload and manual download. That is acceptable for a Hackster project because the hardware experience remains the main contribution.

Conclusion

The strongest version of this project is not a product advertisement. It is a practical physical workflow: record a video locally, use an AI video translator for multilingual output, review subtitles or dubbing, and play the localized result on a Raspberry Pi kiosk. This framing fits Hackster because the project has visible hardware, a reproducible build path, a clear demo result, and a software service that solves a specific step in the workflow.