

Tradurre meeting live con AI locale — guida v5.1 (aggiornata)

Tag: AI · GPU · Guide · Livetest · multi-lingua · Traduzioni · Whisper

Ultimo aggiornamento: 2026-09-27

Tradurre meeting live con AI locale — guida v5.1 (aggiornata)

Evoluzione della guida v2. Descrive lo stato attuale del sistema `live_translate` su `pcsebus` (traduzione live con AI locale, tutto in LAN, nessun cloud). Le sezioni v3/v4/v5 si sostituiscono alla v2. Ultimo aggiornamento: 2026-09-27.

1. Obiettivo

Tradurre in tempo (quasi) reale un meeting, usando solo hardware locale. Dal telefono/browser si manda il microfono a un server su `pcsebus`; il server restituisce trascrizione, traduzione e audio TTS frase per frase. Risultato attuale: audio tradotto ~0,4-0,6 s dopo la fine della frase.

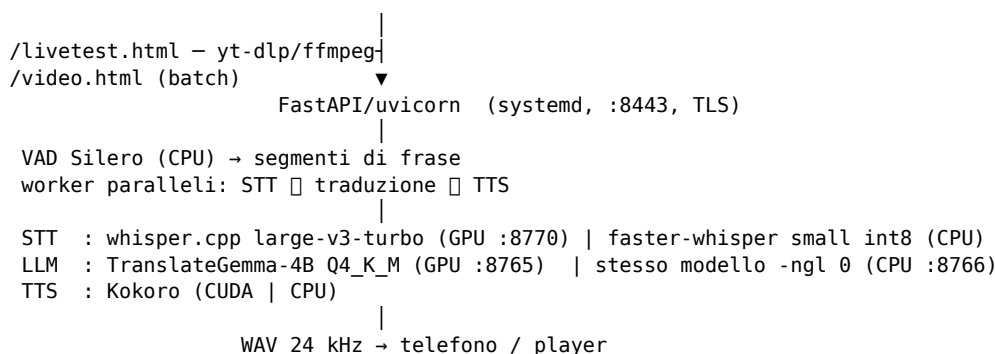
Oltre al live: traduzione video (batch: YouTube/file → audio doppiato + SRT o MP4), multi-lingua (EN/IT/FR/ES/ZH/PT/JA) e un harness di test che misura la pipeline su input ripetibile.

2. Cosa è cambiato dalla v2

Versione	Novità principali
v3	Guardia VRAM + fallback CPU; modelli CPU alternativi (NLLB/M2M); Tecnico IT con glossario; traduzione video.
v4	<code>systemd</code> ; file <code>.mode</code> ; API autenticate (<code>/api/mode</code> , <code>/api/restart</code>); pannello modalità nella PWA; sidecar selettivi; token WS.
v5	Harness livetest; voce donna/uomo/auto (stima F0); lessico di pronuncia TTS ("file"→"fail"); guard anti-allucinazione + retry contestuale; preferenze persistenti.
v5.1	Posizionamento GPU/CPU per-componente con priorità + pagina <code>/status.html</code> ; multi-lingua (fr/es/zh/pt/ja); PWA a tab (Live/Latenza/Trascrizione); auto-connetti; tasto Rinizia; sottotitoli sincronizzati con l'audio; fix accumulo audio e direzione.

3. Architettura

TELEFONO (PWA) —WSS :8443 —



I tre stadi girano su worker asincroni paralleli (STT del segmento N+1 mentre si traduce/sintetizza N).

4. Posizionamento GPU/CPU per componente (priorità)

Non più "tutto GPU o tutto CPU": un pianificatore (`server/placement.py`) decide per ciascun componente dove farlo girare.

- Priorità (LT_GPU_PRIORITY, default llm, stt, tts): dal componente più importante per la velocità al meno.
- Bisogno VRAM per componente (LT_VRAM_NEED_{LLM,STT,TTS}_MB, default 3400/1500/1700) e riserva (LT_VRAM_RESERVE_MB, default 400).
- In ordine di priorità: se il componente entra nella VRAM libera → GPU, altrimenti → CPU/RAM. Il piano è scritto in `.placement.json` (device + motivo per componente).
- La modalità si imposta con il file `/localIA/live_translate/.mode`: auto | gpu (posizionamento a priorità con fallback) | cpu (tutto CPU).

Componente	GPU	CPU
STT	whisper.cpp (:8770)	faster-whisper small int8 (in-app)
Traduzione	llama-server TG-4B (:8765)	llama-server TG-4B -ngl 0 (:8766)
TTS	Kokoro CUDA	Kokoro CPU

Pagina `/status.html` (link "□ Stato"): tabella con, per ogni componente, Dove (□ GPU / □ CPU), backend attivo, VRAM richiesta e motivo; si aggiorna ogni 5 s. Il piano si ricalcola a ogni avvio/riavvio (`lt.sh sidecars_guard / pre-systemd`, che usa le primitive `sidecars.sh start/stop-whisper`, `start/stop-llm-gpu`, `start-cpu`).

4.1 Build CUDA e GPU

I binari `whisper-server` e `llama-server` devono essere compilati per l'architettura della GPU. Su pcsebus (RTX 3060 12 GB, sm_86) la build precedente era solo sm_120 → no kernel image is available. Ricompilare con:

```

export PATH=/usr/local/cuda-13.2/bin:$PATH
cd /localIA/llama/whisper.cpp
cmake -B build -DCMAKE_CUDA_ARCHITECTURES=86 -DGGML_CUDA=ON -DCMAKE_BUILD_TYPE=Release && cmake --build build -j 12
cd /localIA/llama/llama.cpp
cmake -B build -DCMAKE_CUDA_ARCHITECTURES=86 -DGGML_CUDA=ON -DCMAKE_BUILD_TYPE=Release && cmake --build build --target llama-server -j 12
  
```

Verifica: `cuobjdump --list-elf build/bin/libggml-cuda.so.0 | grep -oE 'sm_[0-9]+' | sort -u`.

5. Servizio systemd

`live-translate.service` (User `pi-agent`, WorkingDirectory `/localIA/live_translate`):
`ExecStartPre=lt.sh pre-systemd` (smonta istanze manuali, calcola il posizionamento e avvia i sidecar), poi `uvicorn server.app:app` in `HTTPS :8443`. `Restart=on-failure`. Unit non abilitata a boot (avvio manuale). Sudoers `/etc/sudoers.d/piagent-lt` per start/stop/restart senza password.

```
sudo systemctl start|stop|restart live-translate
./lt.sh status
```

6. Lingue supportate

Elenco configurabile con `LT_LANGUAGES` (default `en,it,fr,es,zh,pt,ja`). Si scelgono nell'app con due menu Da / A + pulsante $\rightleftharpoons$ (inverti), presenti in PWA, Video e Livetest. La direzione inviata è `src-tgt` (es. `en-fr`).

Voci Kokoro per lingua e sesso:

Lingua	Femminile	Maschile
it	if_sara	im_nicola
en	af_heart	am_michael
fr	ff_siwis	ff_siwis
es	ef_dora	em_alex
zh	zf_xiaobei	zm_yunjian
pt	pf_dora	pm_alex
ja	jf_alpha	jm_kumo

Le voci si scaricano al primo uso (~pochi MB; la prima sintesi è più lenta). STT usa la lingua sorgente, TTS la voce della lingua target. Guardia lingua: l'euristica `detect_lang` è affidabile solo per `en/it`; per le altre lingue la verifica viene saltata (altrimenti `es/fr` verrebbero scambiate per `it` e scartate).

Persistenza: ogni pagina ricorda le sue lingue (`localStorage`); il server salva l'ultima direzione in `.prefs.json` (`last_direction`, vedi `server/prefs.py`) e la espone in `/health.prefs` come default condiviso.

7. La PWA (telefono)

`https://192.168.1.2:8443` (cert self-signed; il microfono richiede HTTPS). Organizzata in tre tab:

- Live: Connetti / Avvia microfono (si connette da solo), Da/A, sesso voce (Auto/Donna/Uomo), Tecnico IT, □ Uso le cuffie, token. Sottotitoli: mostra solo la frase il cui audio sta suonando, con originale e traduzione stessa dimensione ma colori diversi. □ Rinizia: scarta l'arretrato (audio in coda + pipeline server) e riprende da adesso; fermare il microfono blocca subito tutto il flusso. □ Uso le cuffie: con le cuffie il telefono continua ad ascoltare mentre riproduce (full-duplex) → il ritardo non cresce.

- □ Latenza: modalità server (Auto/GPU/CPU) + slider in diretta (taglio massimo frase, pausa che chiude la frase, fusione frammenti) con "Ripristina default". Si applicano subito alla sessione attiva.
- □ Trascrizione: istantanea statica (non live) di tutte le frasi, con ↓ Tutto / Solo originale / Solo traduzione, Copia e Pulisci.

8. Traduzione video (batch)

/video.html: URL YouTube o file locale → Da/A, Tecnico sì/no, output audio (WAV + SRT) o video (MP4 doppiato + sottotitoli). Job seriali, persistenti, auto-cancellati dopo 7 giorni, file con HTTP Range.

9. Livetest — harness di test

/livetest.html: alimenta la stessa pipeline con l'audio di un video (YouTube/file, 1x o turbo) e produce trascrizione + traduzione per segmento, metriche (stt_ms, mt_ms, tts_ms, latenza dall'inizio del parlato, RMS, confidenza STT, allucinazioni/scarti), WAV per segmento e metrics.jsonl. Serve a misurare ogni modifica su input ripetibile.

10. Qualità: guardie e lessico

- 1 Guard anti-allucinazione STT (LT_STT_GUARD=off|log|drop): Whisper su segmenti brevi/ambigui inventa frasi ("and that's cool" → "I promise."). Si usano le probabilità per parola (verbose_json): validi avg_prob ~0,82-0,97; allucinazioni ~0,38-0,74 con min_prob ~0,24. Retry contestuale: se sospetto, ri-trascrive col contesto precedente (testo e, se serve, l'audio del segmento precedente) e tiene solo la parte nuova → la frase è recuperata; altrimenti, con drop, è scartata.
- 2 Guardia lingua (solo en/it, vedi §6) e scarto degli "elenchi di alternative" del modello.
- 3 Glossario Tecnico IT: i termini standard restano in inglese.
- 4 Lessico di pronuncia TTS (LT_TTS_LEXICON_IT, es. file=fail;files=fails): applicato solo all'audio, il testo a schermo resta "file".

Tutto opzionale/disabilitabile da .env e dall'interfaccia.

11. Prestazioni misurate

GPU (warm): STT ~0,16 s · traduzione ~0,15 s · TTS ~0,06 s → latenza percepita ~0,4-0,6 s (max ~0,7).

CPU vs GPU (stesso video, 1x): STT 1,6 vs 0,16 s · traduzione 2,1 vs 0,15 s · TTS 1,0 vs 0,06 s · perceived media 4,8 vs 0,4 s.

Qualità: WER 0,0 su testo EN noto (voci F e M); guard verificato (nessuna frase valida persa). Modelli più piccoli testati: NLLB-200-600M (~1,3 GB VRAM), M2M100-418M (~0,9 GB), opus-mt-en-it (~0,16 GB) — qualità inferiore (più letterali). VRAM attuale dello stack: ~5,8 GB.

12. Uso quotidiano

```
cd /localIA/live_translate
./lt.sh start|stop|restart|status|logs
./lt.sh stop all # ferma anche i sidecar
./lt.sh mode [auto|gpu|cpu]
```

```
./sidecars.sh start|start-cpu|start-whisper|start-llm-gpu|stop|stop-gpu|stop-cpu|stop-whisper|stop-llm-gpu|status
```

/health riporta mode, backends, placement, languages, prefs.

13. Configurazione (.env, principali)

Variabile	Default	Significato
LT_WS_TOKEN	(impostato)	token WS + API di controllo
LT_LANGUAGES	en,it,fr,es,zh,pt,ja	lingue offerte nell'app
LT_GPU_PRIORITY	llm,stt,tts	ordine di priorità per il posizionamento
LT_VRAM_NEED_{LLM,STT,TTS}_MB	3400/1500/1700	VRAM stimata per componente
LT_VRAM_RESERVE_MB	400	riserva VRAM
LT_VOICE_GENDER	auto	voce: auto/f/m
LT_GENDER_F0_THRESHOLD	160	soglia F0 (Hz) per la stima del sesso
LT_KOKORO_VOICE_{LANG}_{F,M}	(vedi §6)	catalogo voci per lingua/sesso
LT_TTS_LEXICON_IT	file=fail;files=fails	lessico pronuncia TTS (vuoto = off)
LT_STT_GUARD	log (drop su pcsebus)	anti-allucinazione: off/log/drop
LT_STT_CONTEXT_RETRY	1	retry col contesto
LT_MIN_SEGMENT_S	0,8	fusione frammenti brevi
LT_CARRY_MAX_WAIT_MS	700	attesa massima del frammento breve
LT_CPU_LLM_MODEL	translategemma	LLM CPU (nllb/m2m/qwen2.5:3b)
LT_VAD_MIN_SILENCE_MS	800	pausa che chiude la frase
LT_MAX_SEGMENT_S	6,0	taglio forzato dei segmenti lunghi

14. Problemi incontrati → soluzioni

Problema	Causa	Soluzione
App muta dopo riconnessione	Audio.play() rifiutato	rilascio on-reject + watchdog + reset coda
Ritardo crescente fino a ~20 s	l'audio tradotto era il 51% più lungo (TG-4B allungava il testo)	cap dinamico output + speed=1,3 → rapporto ~1,0

Problema	Causa	Soluzione
Direzione disallineata (UI EN→IT, pipeline IT→EN)	session_ready sovrascriveva la scelta, il server non confermava	session_start conferma con direction_changed
Traduzioni "senza senso"	allucinazione Whisper su segmenti brevi	guard + retry contestuale (§10)
Altre lingue "non funzionano"	guardia lingua scambiava es/fr per it e scartava	guardia applicata solo per en/it
Frammento finale perso	carry non forzato a fine flusso	flush accoda comunque
GPU: "no kernel image"	build CUDA per arch sbagliata	ricompilare per sm_86 (§4.1)
TG CPU crash CUDA con -ngl 0	build esegue matmul Q4 su CUDA	CUDA_VISIBLE_DEVICES= in modalità CPU
Repo M2M inesistente	m2m_100_418M	m2m100_418M

15. File di riferimento

```

/localIA/live_translate/
  lt.sh / sidecars.sh / run_server.sh
  server/{app,pipeline,config,vad,tts_kokoro,video_jobs,session,avfeed,livetest,gender,placement,prefs,
runtime}.py
  web/{index,app.js,video.html,video.js,livetest.html,livetest.js,status.html,status.js}
  .mode .env .placement.json .vad.json .prefs.json logs/server.log
/localIA/llama/llama.cpp/build/bin/llama-server      (TG, GPU :8765 / CPU :8766)
/localIA/llama/whisper.cpp/build/bin/whisper-server (STT, GPU :8770)
/etc/systemd/system/live-translate.service
/etc/sudoers.d/piagent-lt

```

Guida aggiornata allo stato v5.1 (2026-09-27). La v2 resta come riferimento storico.