LaRa Development Session Report

Feb 24, 2026 — Stabilization, Speed Optimization & Wake-Word Interrupt

LaRa Development Session — Feb 24, 2026

Complete documentation of all engineering work performed on the LaRa system (Low-Cost Adaptive Robotic-AI Therapy System for Down Syndrome and Autistic Children).

System Overview

The LaRa audio pipeline consists of:

1. **Microphone Input** → 16kHz mono audio, 30ms frames
2. **WebRTC VAD** → Voice Activity Detection (Mode 3) + RMS energy gate
3. **Whisper small.en** → Offline speech-to-text transcription
4. **Ollama LLM** → AgentricAI_TLM, low temperature (0.15), max 120 tokens
5. **Piper TTS** → Offline text-to-speech with length_scale=1.2
6. **Audio Playback** → Cross-platform (afplay macOS / aplay Linux)

System State Machine

The system operates in three defined modes:

- **RESTING** — Waiting for wake word "friday". Whisper transcribes, but only checks for activation keyword.
- **LISTENING** — Full Whisper transcription active. User speech is captured, transcribed, and sent to the LLM.
- **SPEAKING** — TTS is active. KWS monitors microphone for wake-word "LaRa" to allow safe interruption.

State transitions:

- RESTING → LISTENING: User says "friday"
- LISTENING → SPEAKING: LLM response ready, TTS begins
- SPEAKING → LISTENING: Speech completes naturally OR user says "LaRa"

- LISTENING → RESTING: User says "shutdown"

Work Completed

1. Streaming Removed — Buffered Generation

Problem: LLM chunks were streamed to console character-by-character before validation. This created a potential mismatch between what was displayed on screen and what was spoken aloud.

Fix: The streaming API endpoint is still used internally (proven reliable), but text is now accumulated silently into a buffer. After the full response is generated, it passes through validation, then is displayed and spoken atomically.

Result: What the child reads = what the child hears. Predictability is preserved.

File modified: `src/whispercpp_STT.py`

---

2. Response Safety Validation

Problem: Even with token limits (120 tokens), the LLM may generate cognitively dense output with compound instructions or complex sentence structures.

Solution: Added `validate_response()` function that enforces:

- **Maximum 3 sentences** — Excess sentences are trimmed
- **No compound connectors** — Strips ", and then" and ", then"
- **Maximum 2 commas per sentence** — Truncates at 3rd comma
- **All trimming events logged** — For clinical transparency

File modified: `src/whispercpp_STT.py`

---

3. Barge-In Detection Hardened

Problem: A single VAD frame could interrupt LaRa's response generation. Neurodiverse children may vocalize (cough, hum, click tongue) without intending to interrupt.

Fix: Require **10 consecutive speech frames** (300ms at 30ms/frame) above the noise gate threshold before triggering an interruption. The counter resets on any non-speech frame, ensuring only intentional, sustained speech triggers a barge-in.

Configuration: `BARGE_IN_FRAME_THRESHOLD = 10`

File modified: `src/whispercpp_STT.py`

---

4. Audio Normalization Fixed

Problem: The previous normalization code `full_audio / np.max(np.abs(full_audio))` would amplify near-silence (background noise) to full volume, feeding static noise into Whisper and causing phantom transcriptions.

Fix: Only normalize weak signals where peak amplitude is below 0.5:

```
peak = np.max(np.abs(full_audio))
if 0 < peak < 0.5:
    full_audio = full_audio / peak
```

Strong signals and silence are left untouched. This prevents noise amplification while still boosting genuinely quiet speech.

File modified: `src/whispercpp_STT.py`

---

5. TTS Playback Duration Logging

Added timing instrumentation around TTS playback:

- Records `time.time()` before and after playback
- Logs the duration with every utterance
- Emits a clinical **warning** if speech exceeds **12 seconds** (cognitive comfort threshold for neurodiverse children)

Example log output:

```
[TTS Out] Spoke: Hello! I am here to play and learn with you. | Duration: 4.4s
[TTS Duration Warning] Speech lasted 13.8s — may exceed cognitive comfort
```

File modified: `src/piper_TTS.py`

---

6. Speed Optimizations

Change	File	Effect
`SILENCE_DURATION_MS` : 2500 → 1800	whispercpp_STT.py	700ms faster response per turn
Removed `fp16=False` from Whisper init	whispercpp_STT.py	Fixed crash (unsupported param)
Removed `num_threads` / `beam_size` from `transcribe()`	whispercpp_STT.py	Fixed crash (params set at init)
Fixed `SynthesisConfig` for Piper	piper_TTS.py	`length_scale` via config object
Removed `\n` from LLM stop tokens	AgentricTLM.py	Prevents premature response cutoff

7. Wake Word & Shutdown Commands Updated

Old Command	New Command	Action
"model wake up"	"friday"	Activates LaRa
"model shutdown"	"shutdown"	Shuts down LaRa
(none)	"lara"	Interrupts speech

Bug fix: The Python condition `if "shutdown" or "SHUTDOWN" in text.lower()` always evaluates to `True` because `"shutdown"` is a truthy string. Fixed to `if "shutdown" in text.lower()`.

8. Wake-Word Interrupt During Speech (Headline Feature)

This is the most significant feature added today. It transforms LaRa from an "assistant that talks at you" into an "attentive companion that listens to you."

Why This Matters

When a neurodiverse child interrupts, they are often:

- Seeking control over their environment

- Seeking reassurance that someone is listening
- Testing the system's responsiveness

If LaRa responds instantly and calmly, that **builds trust**. If LaRa ignores them until finishing speech, that **builds frustration**. This feature is relational architecture.

Architecture

TTS Module (`piper_TTS.py`):

- `subprocess.run()` (blocking) replaced with `subprocess.Popen()` (non-blocking) + 50ms polling loop
- New `interrupt_speech()` method: kills the playback process, sets `is_speaking = False` , logs the event
- 1-second cooldown prevents rapid re-triggers from accidental noise
- `speak()` now returns `True` (completed) or `False` (interrupted)

Main Loop (`whispercpp_STT.py`):

- New `speak_and_monitor()` function runs TTS in a daemon thread while monitoring audio in the main thread
- During SPEAKING mode, each audio frame is checked with VAD + RMS
- After 300ms of sustained speech, runs Whisper `small.en` on the short clip
- If transcription contains "lara" → calls `interrupt_speech()` , plays "Okay. I am listening.", transitions to LISTENING

Interrupt Flow

1. LaRa is speaking (SPEAKING mode)
2. Child says "LaRa"
3. Microphone detects sustained speech (≥ 300ms via VAD + RMS)
4. Whisper transcribes the short clip → detects "lara"
5. `interrupt_speech()` kills the `afplay` / `aplay` process
6. LaRa says calmly: "Okay. I am listening."
7. System transitions to LISTENING mode
8. Child speaks their next utterance normally

Safety Protections

Protection	Implementation
No abrupt silence	"Okay. I am listening." always follows interrupt
No false positives	300ms sustained VAD + RMS > threshold required
No rapid re-trigger	1-second cooldown after each interrupt

Protection	Implementation
No audio feedback	Queue flushed after every TTS event
No race conditions	Daemon thread + flag-based signaling

Files Modified Summary

File	Key Changes
`src/piper_TTS.py`	Non-blocking Popen, interrupt_speech(), duration logging, cooldown
`src/whispercpp_STT.py`	SystemMode enum, validate_response(), speak_and_monitor(), KWS, barge-in hardening, normalization fix, bug fixes
`src/AgentricAi/AgentricTLM.py`	Removed newline stop token

Compliance Audit — agents_instruction.txt

#	Constraint	Status
1	Predictability over randomness	PASS — 300ms threshold, cooldowns, no snap reactions
2	No rapid state switching	PASS — Enum-based state machine with logged transitions
3	Confidence-validated detection	PASS — VAD + RMS + duration gates on all triggers
4	Calm, encouraging, non-judgmental	PASS — All interrupt responses are gentle
5	Clear audio, moderate pacing	PASS — length_scale=1.2, no abrupt audio cuts
6	No invented behavioral states	PASS — Exactly 3 defined SystemMode states
7	Lightweight deployment	PASS — Single Whisper model reused for both tasks
8	Log state transitions	PASS — All interrupts, detections, durations logged
9	Short sentences, simple vocabulary	PASS — validate_response() enforces structurally
10	When uncertain: slow down	PASS — Fail-safe responses, cooldowns, never escalate