LOCAL LLM INTERNSHIP REPORT

Part 2: Ollama Benchmarking · Qwen3 Malayalam · PyLM Studio App
Encoding Investigation · Model Comparison · Custom Desktop Application

Edge AI for Robotic Bell | June 2026 | CPU-Only Windows Deployment

This report documents the second phase of the local AI deployment internship — covering the Ollama model benchmark across three machines, the Malayalam encoding investigation and its resolution, the discovery and evaluation of Qwen3 4B as a Malayalam-capable model, the development of PyLM Studio (a custom desktop AI application with RAG, voice, vision, and face recognition capabilities), all running entirely offline on consumer hardware.

1 HARDWARE OVERVIEW — THREE-MACHINE TEST ENVIRONMENT

Benchmarks were performed across three machines representing different capability tiers:

Component	ANASWAR_CS (Dev Laptop)	v-robotics2 (Bell PC)	DESKTOP-ASCP5AV (Lab PC)
CPU	AMD Ryzen 7 7445HS @ 3.20 GHz	Intel i3-2100 @ 3.1 GHz (2C/4T)	Intel i3-1115G4 @ 3.00 GHz (11th Gen)
RAM	32 GB (31.3 usable)	12.0 GB	8.00 GB (7.80 usable)
GPU	RTX 3050 (4 GB) + Radeon 740M iGPU	None	None (integrated UHD only)
OS	Windows 11 + WSL2	Ubuntu 26.04 LTS	Windows 11
Storage	762 GB free	146 GB free	278 GB / 477 GB used
Role	Development & GPU tests	Production Bell PC	School lab / CPU-only

2 OLLAMA MODEL BENCHMARK — EDGE AI FOR ROBOTIC BELL

The project objective was to integrate a local LLM with an existing robotic bell system to generate science facts, motivational quotes, and word-of-the-day content — fully offline, on edge hardware with no dedicated GPU. Seven models were evaluated across the three machines.

2.1 Models Tested

Model	Engine	Size	Loaded?	Resp. Time	Output Quality
Qwen2.5:0.5b	Ollama	~400 MB	Yes	5–10 s	Good, coherent, on-topic
Qwen2.5:1.5b	Ollama	~1 GB	Yes	10–20 s	Good, more detailed
Qwen2.5:3b	Ollama	~2 GB	Yes	20–40 s	Best of the Qwen set
Boomer-1b	Transformers	2.14 GB	Yes	1–2 min	FAIL – repetitive loops

Model	Engine	Size	Loaded?	Resp. Time	Output Quality
Boomer-634m	Transformers	~700 MB	Yes	1–3 min	FAIL – nonsense output
Hex-1 (4B)	Transformers	~8 GB	No (0% stall)	—	CRASH – OOM
Qwen2.5:7b / Aya:8b	Ollama	~5–8 GB	Not tested	Not tested	Identified as alternative

2.2 Detailed Notes Per Model

Qwen2.5 Family (Ollama)

- Installed with a single command: `ollama pull qwen2.5:0.5b` (and 1.5b / 3b variants).
- Ran cleanly on all three machines — including v-robotics2 (i3-2100, no GPU).
- Qwen2.5:0.5b was fastest with usable accuracy; Qwen2.5:3b delivered the best quality.
- Generated accurate science facts, quotes, and word-of-day content on first attempt.
- Qwen2.5:3b confirmed with GPU acceleration on ANASWAR_CS (RTX 3050).

BudEcosystem Boomer-1b & Boomer-634m (Transformers)

- Required manual fixes to `generate.py` — hardcoded to CUDA `device_map` (unavailable on test hardware).
- Required additional packages: `sentencepiece`, `tiktoken`, `accelerate` (not in requirements.txt).
- Root cause: both models are pretrained-only — text completion, not instruction following. Output degenerated into loops like 'the next day, the next day…' or mathematical notation.
- Benchmark scores confirm: Boomer-1b scored ARC 22.35 / MMLU 25.92 — near random-chance for a 4-choice task.

BudEcosystem Hex-1 4B (Transformers)

- Attempted on DESKTOP-ASCP5AV (8 GB) — insufficient RAM, load not attempted.
- On v-robotics2 (12 GB) — download completed but weight-loading stalled at 0%.
- Root cause: ~8 GB for weights alone leaves no headroom for OS + PyTorch overhead.
- `vllm` (alternative engine) tried but failed — vllm requires a GPU.
- No public GGUF/quantized version of Hex-1 existed at time of testing.

Root Cause Summary: Boomer variants are base models (text completion, not instruction-tuned) and produce unusable output for Q&A tasks. Hex-1 could not be loaded within available RAM and lacked a practical quantized distribution at time of testing.

2.3 Benchmark Conclusion & Recommendation

Recommendation: Qwen2.5 via Ollama is selected as the practical edge AI engine for the robotic bell project. Qwen2.5:3b offers the best accuracy within safe RAM limits on v-robotics2 (12 GB); Qwen2.5:0.5b and 1.5b are faster fallbacks. Installation is a single command, inference is reliable, and no GPU is required.

3 MALAYALAM ENCODING INVESTIGATION

After the Hex-1 GGUF deployment (Part 1), a dedicated investigation was undertaken to understand why Malayalam text was being corrupted before reaching the model. Three separate layers of encoding failure were identified and resolved.

3.1 Problem Symptoms

- Malayalam prompt appeared as **??????** (literal question marks) — complete data loss before the model received the text.

- After switching to file-based prompt (`-f prompt.txt`), prompt echoed as garbled symbols instead of real Malayalam characters.
- Model output was nonsense timestamps and numbers, not Malayalam text.

3.2 Root Causes Identified

Layer	Root Cause	Effect
Console codepage	Windows PowerShell legacy codepage 1252	Keyboard input converted to '?' before reaching llama.cpp
File save encoding	Notepad defaulted to ANSI (not UTF-8)	File appeared correct visually but was mis-encoded on disk
System locale	'Language for non-Unicode programs' set to CP-1252	C runtime corrupted UTF-8 bytes inside llama.cpp even with a correctly saved file

3.3 Fixes Applied

- Saved `prompt.txt` via Notepad with explicit UTF-8 encoding and full path in the Save As dialog.
- Verified file integrity using PowerShell's .NET UTF-8 decoder — confirmed Malayalam was stored correctly on disk.
- Enabled **Settings** → **Time & Language** → **Beta: Use Unicode UTF-8 for worldwide language support** and restarted Windows. This fixed the system locale issue at the C runtime level.
- Used `--verbose-prompt` flag to inspect tokenizer input — confirmed Malayalam text reached the model correctly after the fix.

Key Insight: Windows console encoding is a significant, multi-layered obstacle for non-ASCII LLM prompts. The UTF-8 system locale Beta setting (Settings → Time & Language) is the most reliable end-to-end fix. Encoding problems are infrastructure issues — they are fully resolvable and do not reflect model capability.

4 MODEL TESTING — HEX-1 Q4_K_M MALAYALAM RESULTS

With encoding fully resolved, Hex-1 Q4_K_M was tested with five structured prompts covering English, Malayalam understanding, translation, and generation.

Test	Result	Status
English prompt	Correct answer in English	PASS
Malayalam prompt (raw)	Encoding fixed — prompt received correctly	PASS
Malayalam → English	Correct translation	PASS
Malayalam text generation	Garbage output — timestamps / numbers	FAIL
Malayalam fluency test	Mixed Hindi, Tamil, Kannada output	FAIL

Conclusion: Hex-1 demonstrates good Malayalam comprehension and can translate accurately. However it cannot generate fluent Malayalam. This is a training data imbalance — the model encountered enough Malayalam to learn its meaning but not enough to produce it. This is a model capability limitation, not an encoding problem.

5 QWEN3 4B Q4_K_M — MALAYALAM-CAPABLE REPLACEMENT MODEL

Qwen3 4B was downloaded from Hugging Face (~2.5 GB GGUF) and run through the identical five-prompt test suite. It was selected for its strong multilingual training coverage and suitability for 8 GB RAM systems at Q4_K_M quantization.

5.1 Test Results

Test	Result	Status
English prompt	Correct answer	PASS
Malayalam understanding	Correctly understood Malayalam	PASS
Malayalam → English translation	Accurate translation	PASS
Malayalam text generation	Clean Malayalam response	PASS
Malayalam fluency (2–3 sentences)	Fluent, coherent Malayalam output	PASS

5.2 Performance Metrics

Metric	Value
Prompt processing speed	~8.6 tokens / second
Generation speed	~9.5 tokens / second
RAM usage	Within 8 GB limit — no memory paging observed
Thinking mode	[Start thinking]…[End thinking] — reasoning in English, output in Malayalam
Quantization	Q4_K_M GGUF, ~2.5 GB file

Qwen3 4B replaces Hex-1 as the recommended model for Malayalam language tasks. It passes all five tests including fluent Malayalam generation, runs within the 8 GB RAM constraint, and achieves ~9.5 t/s generation speed.

6 FULL MODEL COMPARISON SUMMARY

Capability	Hex-1 Q4_K_M	Qwen3 4B Q4_K_M	Qwen2.5:3b (Ollama)
English understanding	Good	Good	Good
Malayalam understanding	Good	Good	Not tested
Malayalam → English	Good	Good	Not tested
Malayalam generation	Poor	Good	Not tested
CPU inference (8 GB RAM)	Yes	Yes	Yes (12 GB target)
Multilingual support	Poor	Good	Good
Installation simplicity	Manual (GGUF)	Manual (GGUF)	One command (Ollama)
Robotic bell use case	Not suitable	Suitable	Recommended

7 INFRASTRUCTURE SETUP

7.1 llama.cpp (Terminal)

- Downloaded pre-built Windows CPU-only binary from GitHub (build b9651).
- Resolved encoding as documented in Section 3.
- Used `-f prompt.txt` for all non-ASCII (Malayalam) prompts.

Key inference flags:

```
.\llama-cli.exe -m models\qwen3-4b-q4_k_m.gguf \
  --jinja          # enable chat template
  --threads 4      # match i3-1115G4 thread count
  -n 200          # max tokens to generate
  -f prompt.txt   # UTF-8 file for Malayalam input
```

7.2 LM Studio (Desktop GUI)

- Installed LM Studio for a no-terminal graphical chat interface.
- Imported Qwen3 4B GGUF by placing it under `C:\Users\Suresh\.lmstudio\models\Qwen\Qwen3-4B\`.
- Tuned settings to fit within 8 GB RAM: context length reduced 8192 → 2048, model loading guardrails disabled.

8 PYLM STUDIO — CUSTOM DESKTOP AI APPLICATION

The final phase of the internship involved building PyLM Studio — a fully custom Python desktop application that extends local LLM inference with document grounding, voice I/O, vision understanding, and face recognition — entirely offline.

8.1 Technology Stack

Component	Technology / Library
UI framework	PyQt6 — desktop GUI with dark modern theme
LLM backend	LM Studio local API (Qwen3 4B GGUF via llama.cpp, port 8080)
Vector store	ChromaDB — embedding-based document retrieval
Embedding model	sentence-transformers: all-MiniLM-L6-v2
RAG document support	PDF, DOCX, TXT, MD, CSV, and web URLs
Speech-to-Text	Vosk (fallback — faster-whisper torch DLL broken in conda env)
Text-to-Speech	pyttsx3
Vision model	Second llama-server instance (port 8081) — vision-capable GGUF
Face recognition	face_recognition + OpenCV — register & recognize via webcam
Agent web search	DuckDuckGo search integration
Agent file tools	File read/write, Python code execution, directory listing
Response style	Streaming responses with real-time token display

8.2 Key Features

Phase 1 — Core Chat

- PyQt6 UI with sidebar + chat panel, dark modern theme.

- Streaming responses from local Qwen3 4B via llama.cpp server (port 8080).
- RAG pipeline: documents ingested, chunked, embedded into ChromaDB. At inference time, user query is embedded and the most relevant chunks are injected into the model context window — grounded answers from local files with no cloud dependency.
- Supported formats: PDF, DOCX, TXT, Markdown, CSV, website URLs.
- BM25 over PDFs/URLs as initial retrieval layer.

Phase 2 — Voice

- Microphone input with STT via Vosk (used as fallback since faster-whisper's torch DLL is broken in the conda environment).
- Text-to-Speech output via pyttsx3.
- Continuous conversation mode for hands-free interaction.
- Animated voice state indicators in the UI (listening / processing / speaking).

Phase 3 — Vision + Face Recognition

- A second llama-server instance (port 8081) handles vision model queries — image input is base64-encoded and sent to the vision-capable GGUF model for scene description or visual Q&A.;
- Face recognition system using face_recognition + OpenCV — register known faces via webcam, then recognize them live with bounding-box overlays.
- Personalized time-of-day greetings triggered via TTS when a registered face is detected, with a 60-second cooldown to prevent repeated announcements.
- All inference runs locally — no cloud APIs involved at any stage.

Significance: PyLM Studio demonstrates that a complete, production-grade AI assistant — with document Q&A, voice I/O, vision understanding, face recognition, agent tool use, and streaming responses — can be built and run entirely on consumer hardware (8 GB RAM, no GPU) using open-weight models and open-source libraries. No cloud API, no data sharing, no recurring cost.

9 KEY FINDINGS

#	Finding	Detail
1	Windows encoding is a multi-layered problem	Console codepage, file save encoding, and system locale each independently corrupt Malayalam text. The UTF-8 system locale Beta setting resolves all three at the root.
2	Model capability — not infrastructure — was the Malayalam bottleneck	Once encoding was fixed, llama.cpp worked correctly. Hex-1's failure to generate Malayalam reflects training data imbalance, not a deployment or hardware issue.
3	Qwen3 4B Q4_K_M is the practical Malayalam model for 8 GB CPU systems	Passes all five Malayalam tests including fluent generation, runs within 8 GB RAM, achieves ~9.5 t/s on the i3-1115G4.
4	BudEcosystem Boomer models are unsuitable for instruction-following	Pretrained base models (text completion only), confirmed by near-random benchmark scores. Training methodology limitation, not a hardware issue.
5	Qwen2.5 via Ollama is the practical choice for the robotic bell	Single-command install, reliable output, confirmed on production bell PC (i3-2100, 12 GB RAM, no GPU). Qwen2.5:3b offers best quality within limits.
6	Full offline AI is viable on consumer hardware	PyLM Studio proves RAG + voice + vision + face recognition + agent capabilities can run entirely locally on an 8 GB CPU laptop with no cloud dependency.

10 CONCLUSION

This phase of the internship progressed from a single-model deployment (Hex-1) to a comprehensive evaluation of the local LLM ecosystem for two use cases: multilingual inference (Malayalam) and edge AI content generation (robotic bell).

The encoding investigation was a non-trivial systems challenge — three independent layers of Windows encoding corruption were identified, systematically isolated, and fully resolved. This demonstrates that deploying LLMs on Windows for non-English scripts requires deliberate UTF-8 configuration at the OS level, not just the application level.

Model selection proved equally critical. Hex-1 demonstrates that an Indian-language focused model does not automatically provide strong generation capability in all supported scripts. Qwen3 4B, with its broad multilingual training, outperformed Hex-1 on every Malayalam generation test while running within identical hardware constraints.

PyLM Studio represents the practical culmination of the internship — demonstrating that the infrastructure, models, and techniques explored can be assembled into a functional, user-facing product running entirely offline. Phase 3 extended this further with vision understanding and face recognition, all without any cloud dependency.

Final Status

✓	Local AI deployment infrastructure is fully functional.
✓	Malayalam-capable offline inference achieved with Qwen3 4B Q4_K_M.
✓	Robotic bell content pipeline operational with Qwen2.5 via Ollama.
✓	PyLM Studio delivers RAG + Voice + Vision + Face Recognition fully offline on 8 GB CPU hardware.