Forget the Dashboard? Ask the Shop Floor: Wearable Multi-Turn Voice Assistance for Smart Factories

Mathias Pelka

Technische Hochschule Lübeck, Lübeck, Germany
mathias.pelka@th-luebeck.de

Abstract. Should operators forget the dashboard and simply ask the shop floor? Dashboards and stationary terminals separate digital information from the physical task context, while spoken queries such as "What is wrong with this machine?" require both live data access and spatial grounding about nearby equipment.
This paper presents an open platform for wearable voice assistance using Android-based smart glasses. Spoken requests are processed through modular HTTP endpoints for speech recognition, language processing, and speech synthesis, grounded in live Message Queuing Telemetry Transport (MQTT) data and enriched with spatial context from wearable sensing, including position and view-aligned device selection.
The system is evaluated with 80 annotated spoken queries across eight scenario classes. The Instruct variant reaches an LLM-judge pass rate of 87.6 % at a mean end-to-end latency of 2.55 s; a Thinking variant shows no accuracy gain but substantially higher latency. Source code, Android application, and evaluation data are released as open-source artifacts.

Keywords: Smart factory · Artificial Intelligence · wearable computing · human-machine interaction · context-aware assistance · spatial context · MQTT

1 Introduction

Smart-factory operators often need machine information while standing at the workstation, yet access is still commonly mediated by dashboards, stationary terminals, or application-specific interfaces. This creates a mismatch between the physical task context and the digital information required for quick decisions on the shop floor. Voice user interfaces can reduce this friction, but prior work reports persistent challenges in robustness, workflow integration, and grounding responses in current production data [11,19].

Wearable displays move assistance closer to the point of action by enabling hands-free interaction in production and maintenance settings [6]. Recent work also shows growing interest in large language model (LLM)-based assistants for factories, including knowledge support, assembly assistance, and voice dialogue with industrial machines [5,12]. However, these approaches are rarely combined

in one open platform that supports wearable multi-turn voice interaction, live shop-floor telemetry, and spatial operator context.

Spatial context is essential when operators ask ambiguous questions such as "What is wrong with this machine?" The intended machine can be inferred from position or view-aligned context rather than being named explicitly. This paper therefore presents an open experimental platform for Android-based smart glasses that combines spoken interaction, live Message Queuing Telemetry Transport (MQTT) data, and spatial context. The modular client-server pipeline performs speech recognition, language-based response generation, and speech synthesis, while the response generation can be grounded in both machine telemetry and the operator's current spatial context.

The paper makes four contributions:

- an open-source end-to-end wearable voice-assistance platform for Android-based smart glasses in smart-factory environments,
- an MQTT-grounded interaction pipeline that links spoken operator queries to live machine data,
- the integration of spatial context into spoken factory queries, using position and view-aligned device selection for target grounding,
- a compact benchmark with 80 annotated spoken queries covering status, value, interpretation, comparison, follow-up, spatial context, clarification, and unanswerable cases.

The remainder of the paper is structured as follows. Section 2 reviews related work. Section 3 outlines the system architecture. Section 4 describes the smart-glasses platform and the Android client app. Section 5 presents the evaluation. Section 6 concludes the paper and outlines future work.

2 Related Work

Ludwig et al. review voice user interfaces in manufacturing logistics and highlight their practical promise and integration challenges [11]. Wellsandt et al. introduced voice-enabled assistance for predictive maintenance and extended this toward hybrid-augmented intelligence [20,19]. More recently, Mukherjee et al. presented an LLM-based voice interface for dialogues with industrial machines, and Bousdekis et al. combined voice assistance with data-driven analytics in a real manufacturing setting [12,3]. These studies establish the relevance of shop-floor voice interaction but do not address wearable multi-turn assistance grounded in live telemetry and spatial context.

Kernan Freire et al. explored cognitive assistants for factories and LLM-powered manufacturing knowledge sharing [9,10]. Colabianchi et al. evaluated an LLM-based digital assistant in assembly manufacturing and reported benefits for workload and process quality [5]. These approaches demonstrate that LLMs can support industrial assistance, but focus on knowledge access rather than wearable, context-grounded voice interaction in front of live equipment.

Smart glasses and head-mounted displays have been studied as hands-free interfaces for assembly, maintenance, and remote assistance [6,7], though reviews confirm that most systems emphasize visualization over conversational machine access [17]. Context-aware industrial information systems demonstrate the value of linking operator support to real-time production context [1,21]. Choi et al. showed that multimodal smart-glasses queries can support context-sensitive information delivery in mixed reality [4], and Janaka et al. introduced TOM as a general platform for wearable intelligent assistants [8].

Spatial context is particularly relevant when assistance is used directly at machines. Pelka and Willemsen presented a smart-glasses system that streams 3D-printer status via MQTT with location-aware AR feedback [14] and extended this toward view-aligned asset identification using UWB positioning and gyroscope orientation [15]. These approaches ground wearable assistance in the operator's spatial relation to equipment, but use spatial context for device selection and visualization rather than as part of a conversational LLM pipeline.

This paper combines these elements in an open platform for smart-factory assistance based on Android smart glasses, multi-turn voice interaction, live MQTT-based shop-floor data, and spatial context from wearable sensing. The resulting system allows an operator to ask questions about the current production situation while standing in front of equipment; responses can be grounded in both machine telemetry and inferred operator context, such as position or view-aligned asset selection.

3 System Overview

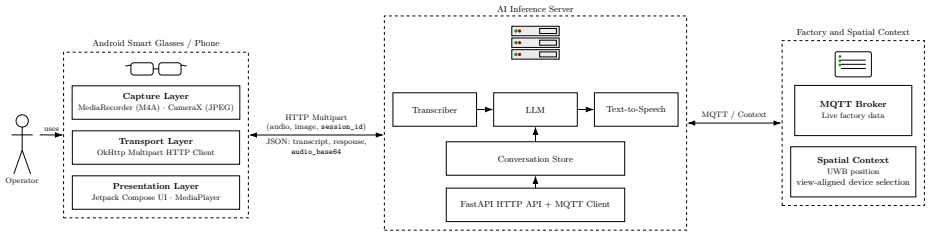

Fig. 1: System overview. User queries from the Android smart-glasses app are processed by an AI inference server grounded in live MQTT-based factory data and optional spatial context from wearable sensing.

Figure 1 shows the overall architecture of the proposed platform. The interaction starts with an operator using Android-based smart glasses or, alternatively, the same client on an Android phone. The user submits a spoken query about the current production situation. The client records audio and, if required, an additional image. The request is sent to an AI inference server, which combines

multimodal user input with live factory data received via MQTT (Message Queuing Telemetry Transport) and spatial context from wearable sensing. The answer is returned as text and optional synthesized speech, enabling hands-free, multi-turn interaction close to the point of action.

The mobile client handles query capture, server communication, and result presentation. It records speech, sends the request to the backend, and presents the returned transcript and answer as text and optional audio. This keeps the client lightweight and allows the same interaction concept to run on different Android devices.

On the server side, the platform is implemented as a modular inference pipeline behind a FastAPI-based HTTP interface. Incoming audio is transcribed and passed to the language model together with the current MQTT-based factory context.

The factory state is provided through direct context injection rather than through a tool-calling or protocol-based retrieval mechanism such as the Model Context Protocol (MCP). This design is motivated by two properties of the target scenario. First, the complete MQTT snapshot covering all monitored devices amounts to only a few hundred tokens, well within the context capacity of the deployed model, so selective retrieval offers no practical benefit. Second, direct injection avoids the additional inference round-trip that tool-call-based approaches require: a protocol-mediated fetch would force the model to first decide which topics to query, execute the retrieval, and only then generate a response, substantially increasing end-to-end latency in an interactive voice setting. Snapshot injection also guarantees temporal consistency, because all machine values reflect the same instant regardless of the order in which they are accessed.

Spatial context provides additional information about the operator's relation to nearby equipment, for example through position estimates and view-aligned device selection. Position information is obtained from a UWB system based on two-way ranging and multilateration. The position estimate is computed with a two-stage estimator using least squares and Newton iteration [13,14]. The resulting position is combined with the viewing direction estimated from the smart glasses' internal gyroscope to identify the device that is most likely relevant to the current query [15].

The server maintains a session-based conversation store for follow-up questions. Each new request is assembled as a message sequence consisting of the prior user turns and model responses followed by the current user utterance with the MQTT snapshot and spatial context appended. The factory state is therefore injected only once per request, at the position of the current turn, so that the model always reasons against the most recent machine data while retaining the full conversational context. Figure 2 illustrates this assembly for two consecutive turns. After response generation, the text is converted into speech by a text-to-speech engine and returned to the client as part of the JSON response.

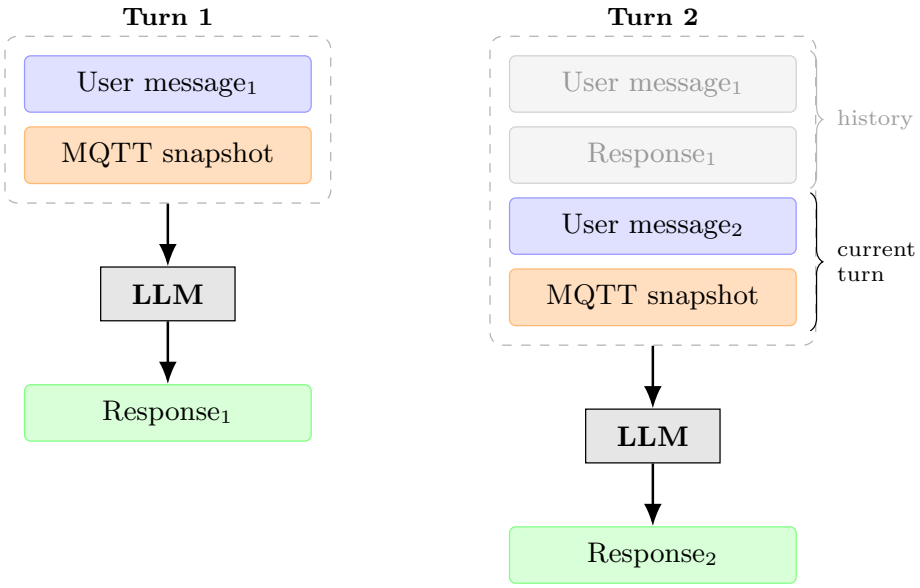

Fig. 2: Context assembly for multi-turn interaction. For each turn, prior exchanges are carried as history (grayed out) while the MQTT snapshot is appended only to the current user message, ensuring the model always reasons against the latest factory state.

4 Smart Glasses

In the proposed system, smart glasses serve as the operator-side interface for hands-free voice interaction close to the point of action. The current implementation focuses on the Vuzix Blade 2, shown in Fig. 3, because it was available in the laboratory setup and provides a suitable Android-based near-eye display for interactive assistance.

The software architecture is not tied to a single device. In this work, the Vuzix Blade 2 was selected as the primary target platform because it provides a comfortable wearable form factor and a display that remains clearly readable during use. Since the client is implemented as an Android application, the same interaction concept can also be deployed on other Android-based smart-glasses platforms such as the Google Glass Enterprise Edition 2 and the Vuzix M4000. The hardware choice therefore affects the form factor and user experience, but not the overall system design.

Fig. 3: Vuzix Blade 2 smart glasses used in the laboratory setup as the primary target platform of the proposed system.

4.1 Smart Glass App

The Android client app provides the front end for recording spoken queries, communicating with the inference server, and presenting the returned results on the smart glasses. Although it is optimized for the Blade 2, the same application can also run on other Android-based smart glasses and Android phones.

The app is organized into three tabs, as shown in Fig. 4. The **Main tab** is used for the actual interaction. It allows the user to start and stop audio recording, displays the generated response, shows the automatic transcript, and provides a reset function for clearing the current dialogue session.

The **Server tab** is used to manage backend endpoints. It supports storing, selecting, adding, and removing server URLs so that different inference backends can be switched directly on the device.

The **Debug tab** provides a timestamped event log for development and troubleshooting. It displays technical events such as requests, responses, and audio playback activity, and can be cleared locally inside the app.

(a) Main tab: voice input and AI response.

(b) Server tab: backend endpoint management.

(c) Debug tab: timestamped event log.

Fig. 4: Android client app running on the smart glasses. The interface covers query interaction, backend configuration, and technical diagnostics.

5 Evaluation

The proposed system was evaluated with a scenario-driven benchmark under controlled laboratory conditions. The goal was to assess whether the platform can process spoken machine-related queries end to end, combine them with MQTT-based shop-floor data, and generate context-appropriate answers in single-turn and short multi-turn interactions.

The benchmark comprised 80 annotated evaluation queries. These queries were organized into eight scenario classes with 10 queries per class. Each query consisted of a recorded spoken utterance, a reference transcript, a corresponding MQTT snapshot, and a structured gold annotation describing the expected semantic content of the answer. For follow-up queries, the required dialogue history was predefined and provided through the session context. The evaluation therefore focused on semantic correctness rather than exact sentence matching.

For each run, the system processed the spoken request through the complete inference pipeline. The generated answer, automatic transcript, associated MQTT snapshot, and runtime logs were stored automatically. The outputs were then compared against the structured gold annotations.

5.1 Scenario Classes

The benchmark comprises eight scenario classes that cover complementary types of machine-related interaction. Table 1 lists the eight classes and one representative example query per class.

Table 1: Overview of the eight scenario classes and example queries. For each class, 10 queries were evaluated.

Scenario class	Example query
Status	What is the current status of printer c?
Value	What is the current bed temperature of printer b?
Interpretation	Is printer c currently ready to print?
Comparison	Which printer currently has the highest bed temperature?
Unanswerable	What is the current energy consumption of printer a?
Spatial context	What is the printer near me doing?
Clarification	What is its current nozzle temperature?
Follow-up	After asking "What is the current status of printer c?", "And is it connected?"

The classes were chosen to cover direct retrieval, context-sensitive reasoning, situated interaction, and ambiguity handling. *Status* queries ask for the current state of a specific machine. *Value* queries request one concrete live value from the MQTT context. *Interpretation* queries require a short semantic assessment derived from current machine data rather than verbatim value reporting. *Comparison* queries require the system to compare multiple machines or signals within the same snapshot. *Follow-up* queries test whether the assistant can resolve references to the preceding turn. *Unanswerable* queries test whether the system avoids unsupported claims when the requested information is not available in the current MQTT context. *Spatial context* queries test whether the system can combine spoken input, structured spatial context, and live machine data to answer situated questions without requiring an explicit machine identifier. *Clarification* queries test whether the assistant asks for the missing machine reference instead of guessing when a spoken query contains an unresolved expression such as "it" or "this one".

5.2 Ground-Truth Scenario Format

Each evaluation item is stored as a structured JSON object containing the user query, the factory context at query time, and the expected answer constraints. This supports reproducible offline evaluation because each test case contains both the spoken input and the exact MQTT snapshot on which the response must be grounded.

A scenario entry contains a unique identifier (`id`), a scenario class (`scenario_class`), a short description, the recorded audio file, and the normalized user utterance in `gold_transcript`. For multi-turn scenarios, `parent_id` links a follow-up query to the preceding interaction; otherwise, it is set to `null`.

The central field is `mqtt_snapshot`, which stores the relevant shop-floor state at query time. The expected output is specified in `gold_answer` through semantic constraints such as `semantic_description`, `must_mention`, and `must_not_mention`.

Responses were evaluated with an LLM as a judge [22]. Instead of comparing an answer to a single reference sentence, the judge checked whether it satisfied the structured scenario definition. This made it possible to accept correct paraphrases while keeping the predefined scenario annotations as the ground truth.

Listing 1.1 shows a shortened, human-readable scenario excerpt. Here, printer b is printing and `print_time_left=675` corresponds to about 11 minutes remaining. An answer stating "675 minutes" or describing the printer as idle would therefore be incorrect.

Listing 1.1: Example ground-truth scenario stored as JSON.

```
{
  "id": "value_06",
  "scenario_class": "value",
  "description": "Remaining print time on printer b while a job is
  running.",
  "audio_file": "value_06.wav",
  "gold_transcript": "How much time is left on printer b's current
  print?",
  "parent_id": null,
  "mqtt_snapshot": {
    "sf/printer/a": "printer_status,... state="Operational",
  print_time_left=0,...",
    "sf/printer/b": "printer_status,... state="Printing",print_time
  =1325,print_time_left=675,...",
    "sf/printer/c": "printer_status,... state="Operational",
  print_time_left=0,...",
    "sf/environment/labor": "environment,... temperature=23.1,humidity
  =26.4,pressure=1021.6"
  },
  "gold_answer": {
    "semantic_description": "The assistant reports that printer b has
  roughly 11 minutes left.",
    "must_mention": ["printer b", "11"],
    "must_not_mention": ["idle", "ready to print", "error"]
  }
}
```

5.3 Model and Runtime Configuration

Speech recognition uses Whisper Large V2 [16] via the `faster-whisper` runtime [18], chosen for offline execution and robust multilingual transcription. Response generation compares two local configurations of `Qwen/Qwen3-VL-8B`: an Instruct variant with extended thinking disabled and a Thinking variant with extended reasoning enabled [2]. Both variants receive the same transcribed query, MQTT snapshot, spatial context, and dialogue history, but use variant-specific prompts for the Instruct and Thinking configurations. Both prompts encode the same task requirements: MQTT field semantics, answers limited to two to three

sentences, grounding in the provided factory and spatial context, and clarification instead of guessing when ambiguous device references cannot be resolved from UWB position, view-aligned device selection, or dialogue history.

Automated evaluation follows the LLM-as-judge paradigm [22]. The judge uses a separate Instruct instance of `Qwen/Qwen3-VL-8B-Instruct` for both model variants. This keeps the judging process identical for both variants. The judge compares each answer with the structured gold constraints and returns a PASS/FAIL verdict.

5.4 Reproducibility and Deterministic Inference

Because the system incorporates timestamped context information from the factory environment, repeated executions of the same query are not strictly deterministic. Although the MQTT snapshot is fixed for each scenario, timestamped request metadata and generated dialogue context can vary across repeated executions. This may lead to different responses or gradual drift in multi-turn interactions.

To account for this effect, each experiment was repeated 20 times. The reported results therefore reflect not only single-run behavior but also the variability introduced by time-dependent context and the generative nature of the pipeline. This procedure quantifies run-to-run variability instead of relying on a single execution.

All experiments were executed on a workstation equipped with an NVIDIA RTX 4000 Ada Generation GPU, an Intel Core i9-14900K CPU, and 128 GB of system memory.

5.5 Results and Model Comparison

Table 2 compares the Instruct and Thinking model variants on the same benchmark of 80 evaluation queries. Both variants were evaluated over 20 repeated runs, resulting in 1600 individual evaluations per variant and 3200 evaluations in total. This repeated evaluation accounts for minor variation in the generated answers and provides a more stable estimate of task success and latency.

The Instruct variant reaches an overall LLM-judge pass rate of 87.6 %, slightly above the Thinking variant at 83.9 %. Across 20 repeated runs each, the Instruct variant answers a mean of 70.10 of 80 queries correctly, the Thinking variant 67.15. Extended internal reasoning therefore offers no observable accuracy advantage in the tested factory-query scenarios. The small gap even favors the Instruct variant. The main differences lie in failure modes and latency.

The Instruct variant performs best on comparison and unanswerable queries, with pass rates of 99.0 % and 94.5 %. It handles explicit MQTT value comparisons reliably and usually rejects unavailable information instead of guessing. Its weakest class is spatial context, with 70.0 %. Most errors result from inconsistent spatial grounding: the model sometimes asks for clarification although UWB position or smart-glasses orientation is available, while in other cases it resolves a target without reliable spatial evidence.

Table 2: Comparison of the Instruct and Thinking model variants aggregated over 20 runs per scenario class. Pass rates are reported as mean values with standard deviations in percentage points.

Scenario class	Instruct		Thinking	
	Pass rate	Latency	Pass rate	Latency
Status	$90.0 \pm 0.0\,\%$	$3.38 \pm 1.49\,\mathrm{s}$	$88.0 \pm 7.0\,\%$	$74.83 \pm 57.32\,\mathrm{s}$
Value	$93.0 \pm 4.7\,\%$	$2.11 \pm 0.33\,\mathrm{s}$	$81.5 \pm 7.5\,\%$	$33.34 \pm 38.35\,\mathrm{s}$
Interpretation	$85.5 \pm 5.1\,\%$	$3.20 \pm 0.84\,\mathrm{s}$	$95.0 \pm 6.1\,\%$	$71.81 \pm 46.66\,\mathrm{s}$
Comparison	$99.0 \pm 3.1\,\%$	$2.65 \pm 0.58\,\mathrm{s}$	$63.0 \pm 13.8\,\%$	$81.08 \pm 47.00\,\mathrm{s}$
Follow-up	$90.0 \pm 0.0\,\%$	$2.38 \pm 0.61\,\mathrm{s}$	$80.5 \pm 7.6\,\%$	$95.05 \pm 62.04\,\mathrm{s}$
Spatial Context	$70.0 \pm 6.5\,\%$	$2.77 \pm 0.85\,\mathrm{s}$	$76.0 \pm 7.5\,\%$	$72.38 \pm 49.87\,\mathrm{s}$
Clarification	$79.0 \pm 3.1\,\%$	$1.89 \pm 0.16\,\mathrm{s}$	$98.5 \pm 3.7\,\%$	$71.06 \pm 52.70\,\mathrm{s}$
Unanswerable	$94.5 \pm 5.1\,\%$	$2.04 \pm 0.38\,\mathrm{s}$	$89.0 \pm 8.5\,\%$	$55.72 \pm 38.57\,\mathrm{s}$
Total	$87.6 \pm 1.8\,\%$	$2.55 \pm 0.91\,\mathrm{s}$	$83.9 \pm 3.0\,\%$	$69.09 \pm 51.55\,\mathrm{s}$

The Thinking variant shows a different error profile. It reaches 98.5 % for clarification and 95.0 % for interpretation queries, but drops to 63.0 % for comparison queries. These failures mainly reflect inconsistent rule application rather than missing reasoning capability. The model sometimes treats cross-printer comparisons as unresolved single-printer queries and asks for clarification instead of comparing all available printers. Further errors include unresolved pronoun references, hallucinated spatial targets, ASR-sensitive printer names, and one benchmark artifact.

Latency is the decisive difference. The Instruct variant answers in 2.55 s on average and remains suitable for interactive voice use. The Thinking variant reaches 69.09 s on average, with high variance (± 51.55 s) and individual queries approaching the 180 s client timeout. It is therefore about 27 times slower without a measurable accuracy gain.

Three main findings follow from the comparison.

- **The system supports real-time factory queries, but spatial grounding remains the main weakness.** The Instruct variant reaches an overall pass rate of 87.6 % and answers within 2.55 s on average. This supports interactive voice-based access to live factory data. However, the low pass rate for spatial context queries (70.0 %) shows that target resolution based on spatial context is not yet reliable enough. The system works best when the queried machine or value is explicitly named.
- **Extended thinking does not improve accuracy and is too slow for interactive use.** The Thinking variant reaches a slightly lower overall pass rate than the Instruct variant (83.9 % vs. 87.6 %), while increasing mean latency from 2.55 s to 69.09 s. This corresponds to an increase by a factor of about 27 without an observed accuracy gain. The error pattern indicates that the main limitation is not missing reasoning capability, but inconsistent

rule application, most visibly in the comparison class, where the pass rate drops to 63.0 %.
- **A subset of failures stems from deterministic subtasks rather than language understanding.** Several recurring errors concern target resolution, arithmetic, unit conversion, time formatting, and explicit cross-printer comparisons. These operations have fully determined answers based on the MQTT snapshot. The failures therefore do not primarily indicate insufficient language capability, but inconsistent execution of well-defined decision rules.

For the present real-time factory voice assistant, the Instruct variant is the more suitable choice. The results also indicate where further development has the highest expected benefit. Spatial references need a more reliable target-resolution layer, instead of leaving this step entirely to the language model. Deterministic subtasks such as arithmetic, unit conversion, time formatting, and cross-printer comparisons can be moved into preprocessing modules or tool calls. This would reduce avoidable failures in the value, comparison, and spatial context classes. Since extended reasoning increases latency without improving overall accuracy, further latency improvements are more likely to come from smaller Instruct checkpoints, quantization, and deterministic preprocessing than from reasoning-heavy model variants.

6 Conclusion and Outlook

Forget the dashboard? No. Dashboards remain useful for monitoring and analysis, but they are poorly suited to hands-free, situation-specific interaction on the shop floor. The proposed system complements them with a wearable, voice-driven interface for context-aware operator support.

The presented platform combines an Android smart-glasses client, a server-side AI pipeline, and MQTT-grounded factory context for spoken operator queries. The Instruct variant reached an 87.6 % pass rate at 2.55 s mean latency, whereas the Thinking variant performed slightly worse (83.9 %) while requiring about 27 times the latency. Extended reasoning therefore offered no measurable benefit in this benchmark. Direct state questions, comparisons, and short multi-turn interactions work reliably; the main limitation lies in spatial context queries (70.0 %), where grounding from UWB position and smart-glasses orientation is not yet consistent.

Future work targets a deterministic spatial-grounding layer, moving further subtasks such as unit conversion and explicit comparisons out of the language model, and exploring smaller Instruct checkpoints for additional latency headroom. The platform provides a compact basis for further research on wearable AI assistance in smart-factory environments.

Open-Source Artifact

The Android smart-glasses app, the server-side AI pipeline including configuration, and the evaluation data are released as open-source artifacts to support

reproducibility and follow-up research on wearable industrial voice assistance. The repository is available at github.com/thl-smart-factory-lab/forget-the-dashboard.

References

1. Alexopoulos, K., Sipsas, K., Xanthakis, E., Makris, S., Mourtzis, D.: An industrial internet of things based platform for context-aware information services in manufacturing. International Journal of Computer Integrated Manufacturing **31**(11), 1111–1123 (2018). https://doi.org/10.1080/0951192X.2018.1500716
2. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
3. Bousdekis, A., Foosherian, M., Fikardos, M., Wellsandt, S., Lepenioti, K., Bosani, E., Mentzas, G., Thoben, K.D.: Augmented intelligence with voice assistance and automated machine learning in industry 5.0. Frontiers in Artificial Intelligence (2025). https://doi.org/10.3389/frai.2025.1538840
4. Choi, S.H., Kim, M., Lee, J.Y.: Smart and user-centric manufacturing information recommendation using multimodal learning to support human-robot collaboration in mixed reality environments. Robotics and Computer-Integrated Manufacturing **91**, 102836 (2025). https://doi.org/10.1016/j.rcim.2024.102836, https://doi.org/10.1016/j.rcim.2024.102836
5. Colabianchi, S., Costantino, F., Sabetta, N.: Assessment of a large language model based digital intelligent assistant in assembly manufacturing. Computers in Industry **162**, 104129 (2024). https://doi.org/10.1016/j.compind.2024.104129
6. Danielsson, O., Holm, M., Syberfeldt, A.: Augmented reality smart glasses in industrial assembly: Current status and future challenges. Journal of Industrial Information Integration **20**, 100175 (2020). https://doi.org/10.1016/j.jii.2020.100175
7. Fang, W., Chen, L., Zhang, T., Chen, C., Teng, Z., Wang, L.: Head-mounted display augmented reality in manufacturing: A systematic review. Robotics and Computer-Integrated Manufacturing **83**, 102567 (2023). https://doi.org/10.1016/j.rcim.2023.102567, https://doi.org/10.1016/j.rcim.2023.102567
8. Janaka, N., Zhao, S., Hsu, D., Tan Jing Wen, S., Koh, C.K.: Tom: A development platform for wearable intelligent assistants. arXiv preprint arXiv:2407.15523 (2024)
9. Kernan Freire, S., Foosherian, M., Wang, C., Niforatos, E.: Harnessing large language models for cognitive assistants in factories. In: Proceedings of the 5th International Conference on Conversational User Interfaces. pp. 1–6. CUI '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3571884.3604313, https://doi.org/10.1145/3571884.3604313
10. Kernan Freire, S., Wang, C., Foosherian, M., Wellsandt, S., Ruiz-Arenas, S., Niforatos, E.: Knowledge sharing in manufacturing using LLM-powered tools: user study and model benchmarking. Frontiers in Artificial Intelligence **7**, 1293084 (2024). https://doi.org/10.3389/frai.2024.1293084, https://doi.org/10.3389/frai.2024.1293084
11. Ludwig, H., Schmidt, T., Kühn, M.: Voice user interfaces in manufacturing logistics: a literature review. International Journal of Speech Technology **26**, 627–639 (2023). https://doi.org/10.1007/s10772-023-10036-x
12. Mukherjee, A., Karande, A., Häfner, P., Poonia, M.D., Kimmig, A., Kreuzwieser, S., Vlas, R., Klar, M., Sykora, T., Grethler, M.: A llm-based voice user interface for

voice dialogues between user and industrial machines. Procedia CIRP **134**, 378–383 (2025). https://doi.org/10.1016/j.procir.2025.02.138
13. Pelka, M., Goronzy, G., Hellbrück, H.: Impact of altitude difference for local positioning systems and compensation with two-stage estimators. In: 2016 International Conference on Localization and GNSS (ICL-GNSS). pp. 1–7 (2016). https://doi.org/10.1109/ICL-GNSS.2016.7533838
14. Pelka, M., Willemsen, T.: Step into context: Location-aware ar feedback for smart manufacturing using low-cost hardware. In: Proceedings of the 2025 International Conference on Indoor Positioning and Indoor Navigation (IPIN). pp. 1–6. Tampere, Finland (2025). https://doi.org/10.1109/IPIN66788.2025.11213170
15. Pelka, M., Willemsen, T.: Look where you work: View-aligned asset identification with uwb and smart glasses. In: Workshop for Computing and Advanced Localization. Rome, Italy (2026), under review
16. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. In: Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 28492–28518. PMLR (2023)
17. Solomashenko, A., Afanaseva, O., Shishova, M., Gulianskii, I., Sobolnikov, S., Petrov, N.: A systematic review of assisted reality applications in manufacturing. Virtual Reality **30**, 32 (2026). https://doi.org/10.1007/s10055-025-01281-3
18. SYSTRAN: faster-whisper: Faster Whisper transcription with CTranslate2. https://github.com/SYSTRAN/faster-whisper (2023), accessed: 2025
19. Wellsandt, S., Klein, K., Hribernik, K.A., Lewandowski, M., Bousdekis, A., Mentzas, G., Thoben, K.D.: Hybrid-augmented intelligence in predictive maintenance with digital intelligent assistants. Annual Reviews in Control **53**, 382–390 (2022). https://doi.org/10.1016/j.arcontrol.2022.04.001
20. Wellsandt, S., Rusák, Z., Ruiz Arenas, S., Aschenbrenner, D., Hribernik, K.A., Thoben, K.D.: Concept of a voice-enabled digital assistant for predictive maintenance in manufacturing. In: Proceedings of the 9th International Conference on Through-life Engineering Services (TESConf 2020). pp. 1–8 (2020). https://doi.org/10.2139/ssrn.3718008
21. Yang, C., Yu, H., Zheng, Y., Feng, L., Ala-Laurinaho, R., Tammi, K.: A digital twin-driven industrial context-aware system: A case study of overhead crane operation. Journal of Manufacturing Systems **78**, 394–409 (2025). https://doi.org/10.1016/j.jmsy.2024.12.006
22. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., et al.: Judging LLM-as-a-judge with MT-Bench and chatbot arena. In: Advances in Neural Information Processing Systems. vol. 36 (2023)