Talk with the Things: Integrating LLMs into IoT Networks

Alakesh Kalita
Department of Mathematics and Computing
Indian Institute of Technology (ISM) Dhanbad, India
alakesh@iitism.ac.in

***Abstract*—The convergence of Large Language Models (LLMs) and Internet of Things (IoT) networks opens new opportunities for building intelligent, responsive, and user-friendly systems. This work presents an edge-centric framework that integrates LLMs into IoT architectures to support natural-language-based control, context-aware prompt construction, and automated device actuation. In the proposed framework, LLMs are deployed on an edge computing device connected to an IoT gateway, enabling local processing of user commands and sensor information while improving privacy and reducing dependence on cloud services. We demonstrate the feasibility of the framework through a smart-home prototype using Llama 3 and Gemma 2B models to control basic appliances through MQTT. The preliminary results highlight the trade-off between model accuracy and inference time with respect to model size. Finally, we discuss potential applications of LLM-based IoT systems and identify key challenges related to latency, reliability, resource constraints, and scalable evaluation.**

***Index Terms*—Large Language Models (LLMs), Internet of Things (IoT), MQTT, Edge Computing**

I. Introduction

With the development of digitization, IoT (*Internet of Things*) technology has become a prominent topic in the field of networking and communication. IoT aims to connect billions of physical devices, which can be both living and non-living entities found on the earth's surface, with sensing/actuation and communication capacities. By enabling seamless connectivity and data exchange, IoT is transforming how we interact with the world around us [1]. On the other hand, new deep generative-based *Large Language Models* (LLMs) are designed to understand and generate human language. Recent advances in generative AI and transformer-based LLMs have demonstrated remarkable capabilities in understanding natural language and reasoning. LLMs, such as GPT-4, GPT-4o, and Llama 3.1, are trained on vast amounts of text data, enabling them to perform a wide range of language tasks, including translation, summarization, and conversation.

Notably, traditional IoT networks only work commands and actions basis in the environment like smart home, industry etc. However, the integration of LLMs in IoT networks can make a huge transformation towards more intelligent, efficient, and responsive IoT systems [2], [3], [4]. This convergence leverages the natural language understanding capabilities of LLMs in IoT, allowing devices to communicate in a more meaningful and context-aware manner [5]. For example, if a user says "`Set up for movie night`", LLMs can help to *set the Dim lights in the living room*, *close the blinds, adjust the thermostat to a comfortable temperature, turn on the TV and navigate to the user's preferred streaming service*. All can be done with a single command, which requires multiple individual commands for each actuation in the current IoT systems. In brief, IoT can serve as the "*sensing limbs*" of an environment, feeding real-time sensor inputs (e.g., temperature, motion, vibrations) to LLMs, while LLMs act as the "*brain*" that analyzes data, draws conclusions, and communicates or acts upon them. Thus, LLM-based communication in IoT can significantly improve efficiency, bandwidth utilization, and user interactions.

However, the devices used in IoT networks are resource-constrained in terms of processing capacity, memory, and power. Therefore, using LLM in this kind of device is not practically trivial. Similarly, technologies like cloud computing have their disadvantages, like higher latency and cost. Therefore, edge computing can be a suitable approach to use LLM in IoT applications such as smart homes, smart industries, etc. Edge computing processes data closer to the source of the data generators rather than relying on centralized data centres at higher propagation delay [6]. Edge computing reduces latency, conserves bandwidth, and enhances real-time data processing capabilities.

Therefore, in this article, we briefly discuss how LLMs can be used to establish efficient communication in edge computing-based IoT systems. In the next section, we present our proposed framework for integrating LLMs into edge-based IoT systems. We then describe our experimental setup in a smart home environment and discuss the results obtained. Finally, before concluding, we highlight the advantages of using LLMs in IoT systems, explore additional use cases beyond smart homes, and outline key challenges that must be addressed to make such systems more efficient and robust.

II. System Architecture: How LLMs Work in an IoT Network

IoT devices are often resource-constrained regarding processing capacity, memory, and power. Consequently, running LLMs directly on these devices is not technically feasible. To address this challenge, we propose a framework where an

ISBN 978-3-903176-82-9 © 2026 IFIP

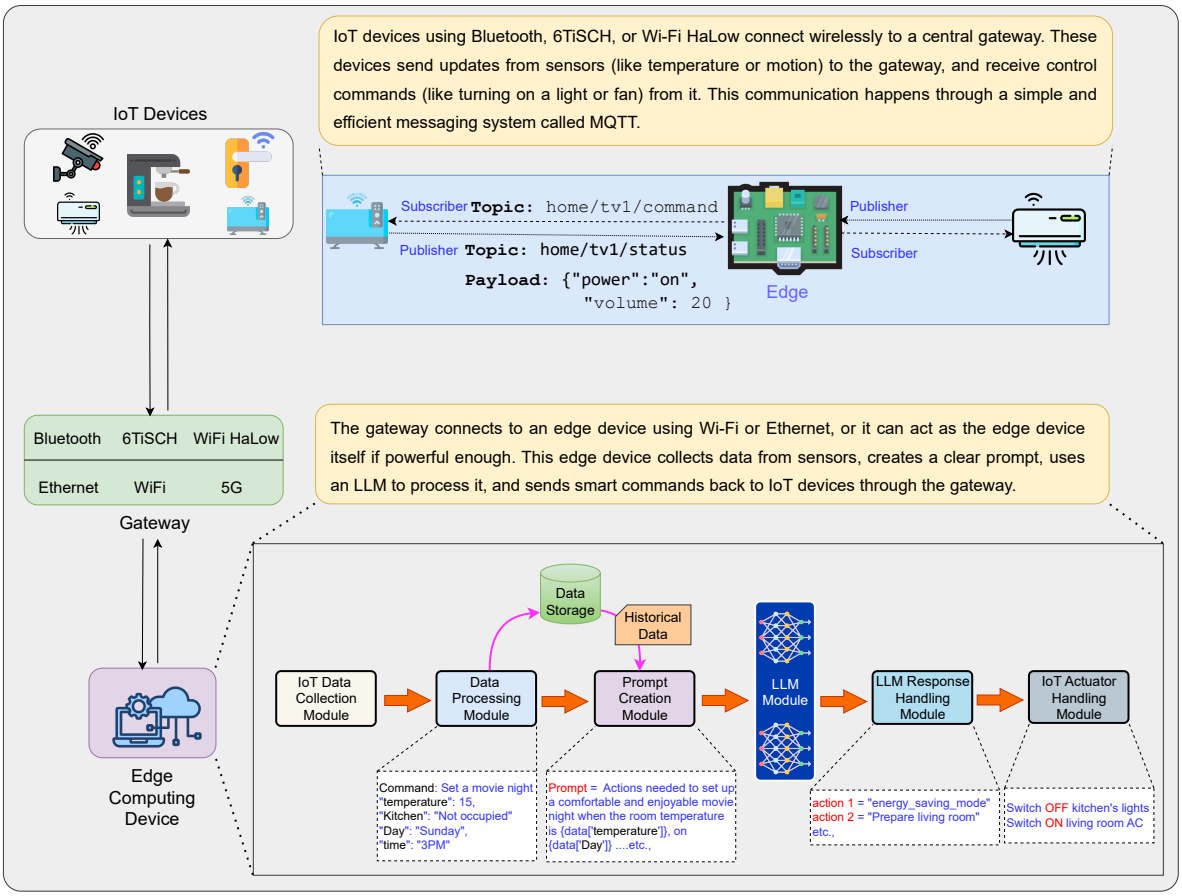

Fig. 1. Proposed framework is divided into multiple structured modules that handle data collection, processing, prompt creation, response handling, and actuator management. Note that we mention some of the possible communication interfaces, but not evaluated in this prototype.

IoT network's border router (BR) or gateway is connected to an Edge computing device equipped with sufficient resources to infer user requests using LLMs. Therefore, in our proposed framework, LLMs are deployed at the edge of the network, and the edge device is connected to the gateway or BR of the IoT network. IoT devices attached to sensors collect information from the environment and transmit their data to the gateway using multi-hop (mesh topology) or single-hop (star topology) communication networks. The gateway then forwards the data to the edge computing device for processing by the LLM deployed at the edge. Note that some edge devices can also serve as IoT gateways. Apart from the input provided by the IoT devices, users can also provide their input directly to the LLMs in the form of text, video, or voice commands. For example, in smart home environments, users can send voice commands to perform different activities. The transmission of data from the IoT devices to the gateway or edge follows traditional source and channel encoding.

To simplify our proposed LLM in IoT network framework to facilitate efficient and intelligent interactions within IoT ecosystems, we divide it into multiple structured modules that handle data collection, processing, prompt creation, response handling, and actuator management. These modules are shown in Figure 1 and discussed as follows,

IoT Data Collection Module: This module is responsible for gathering data from various IoT devices, including the commands received from the users. This module ensures real-time data collection from IoT devices and users. For this, light-weight application layer protocol MQTT (Message Queuing Telemetry Transport) can be used. For example, in a smart home environment, both the Smart TV and the edge device act as publishers and subscribers. The Smart TV publishes its status updates (e.g., power state, volume) to a topic like `home/tv1/status`, which the edge device subscribes to. Conversely, the edge device sends control commands (e.g., `turn on/off`) by publishing to a topic such as `home/tv1/command`, which the Smart TV subscribes to. This bidirectional communication allows seamless monitoring and control using the lightweight MQTT protocol. The broker for the MQTT protocol can be deployed in the edge or gateway of the IoT network.

Data Processing Module: Once the data is collected, it is processed to extract relevant information and convert it into a suitable format for further analysis. This module applies

filtering, aggregation, and transformation techniques to ensure the data is ready for the next stages.

Storage Module: This module is responsible for maintaining historical data, which can be fed to the LLM for referencing past information, providing context, and improving the experience of the end users. In brief, in addition to sensory input and user input, the LLM model is fed with historical data for better inference. For instance, when a user issues a voice command to set up a movie night, our proposed framework can consider previous historical data from the user to adjust the temperature.

Context-Aware Prompt Construction Module: This module prepares structured prompts that combine the user's natural-language command with the current state of the IoT environment. In the present prototype, contextual information is inserted into the prompt using real-time sensor readings, device status information, and predefined domain rules available at the edge device. This allows the LLM to interpret user commands with respect to the current environment instead of treating them as isolated text inputs. For example, a command such as "*prepare the room for sleep*" can be mapped to different device actions depending on the current light, fan, and television states. The generated prompt constrains the LLM to consider only the available devices and to return the required actions in a structured format. This module therefore provides context-aware prompt construction rather than a full retrieval-augmented generation pipeline. A complete RAG implementation with indexed historical data, retrieval ranking, and retrieval-latency evaluation is left as future work.

```
structured_prompt = (
    f"{session['user_command']}.\n"
    f"Only consider these devices: "
    f"{', '.join(session['devices'])}.\n"
    f"Current device states: "
    f"{json.dumps(session['device_states'])}\n"
    f"Current sensor readings: "
    f"{json.dumps(session['sensor_values'])}\n"
    "Return only valid JSON in the following format.
     "
    "Do not include any text before or after the
    JSON:\n"
    "{\n"
    '  "description": "short textual description",\n
    '
    '  "commands": [\n'
    '    {\n'
    '      "device": "light",\n'
    '      "action": "on|off|dim",\n'
    '      "mode": "optional mode string"\n'
    '    }\n'
    '  ]\n'
    "}"
)
```

Listing 1. Structured Prompt Generation

LLM Module: The core component of the framework, the LLM Module, uses open-source language models to interpret structured prompts and generate context-aware responses. The LLM processes the user's command together with the available IoT context and produces device-level actions in the required output format. Some representative open-source LLMs that can be considered for edge deployment are listed in Table I.

```
def get_llama_response(prompt):
    url = "http://localhost:11434/api/generate"
    payload = {
        "model": "llama3:8b",
        "prompt": prompt,
        "stream": False,
        "format": "json"
    }
```

Listing 2. An example of using an open-source LLM

TABLE I
COMPARISON OF OPEN-SOURCE LLMS

Model	Params	Q4 Size	Min RAM
TinyLlama-1.1B	1.1B	0.55–0.7 GB	1.5 GB
StableLM-Zephyr-3B	3B	1.8–2.2 GB	4 GB
Phi-2	2.7B	1.5–1.8 GB	3 GB
Gemma-2B	2B	1.3–1.6 GB	3 GB
Llama-3-8B	8B	3.8–4.5 GB	6 GB
Mistral-7B	7B	4.0–4.5 GB	6 GB
GPT-J-6B	6B	3.0–3.5 GB	5 GB
DistilGPT-2	82M	0.3–0.5 GB	1 GB

LLM Response Handling Module: After the structured prompt is generated, it is sent to the deployed LLM, such as Llama 3, for inference. The LLM processes the prompt and returns a response containing the intended actions to be performed on the IoT devices. This response is then parsed to extract actionable commands in a predefined `JSON` format. The parsing step is essential to verify that the output is valid, structured, and interpretable by the downstream actuator-handling module. Invalid or incomplete JSON responses should be rejected instead of being directly forwarded to the IoT devices.

IoT Actuator Handling Module: The final module in the framework, the IoT Actuator Handling Module, translates the validated LLM response into actionable commands for IoT devices. It ensures that the generated actions are communicated to the appropriate devices through the IoT network, enabling automated control.

```
def control_device(device, action):
    if device == "fan":
        publish_fan(action)
    elif device == "light":
        publish_light(action)
    elif device == "tv":
        publish_tv(action)
```

Listing 3. Transmitting commands to the IoT devices using MQTT

Note that these modules can be integrated with existing IoT infrastructures without redesigning the entire system from scratch. For example, an LLM-based edge module can interface with existing smart-home assistants and IoT devices through APIs and standard communication protocols such as MQTT. This allows the system to interpret user commands, incorporate available contextual information, and execute device-level actions while remaining compatible with current IoT deployments.

III. EXPERIMENTAL SETUP AND EVALUATION

To validate the feasibility of deploying LLMs at the edge of IoT networks for natural-language-based control, we devel-

TABLE II
PRELIMINARY COMPARISON OF LLAMA 3 AND GEMMA 2B IN A SMART-HOME ENVIRONMENT

Model	Command	Expected Output	LLM Response	Inference Time (s)
Llama 3	1. `Set the room for Study`	Light: `on`, Fan: `on`, TV: `off`	Light: `on`, Fan: `on`, TV: `off`	208
	2. `Set the room for movie night`	Light: `on`, Fan: `on`, TV: `on`	Light: `on`, Fan: `on`, TV: `on`	204
	3. `I want to sleep now`	Light: `off`, Fan: `off`, TV: `off`	Light: `on`, Fan: `off`, TV: `off`	126
Gemma 2B	1. `Set the room for Study`	Light: `on`, Fan: `on`, TV: `off`	Light: `on`, Fan: `on`, TV: `on`	28
	2. `Set the room for movie night`	Light: `on`, Fan: `on`, TV: `on`	Light: `on`, Fan: `on`, TV: `off`	29
	3. `I want to sleep now`	Light: `off`, Fan: `off`, TV: `off`	Light: `off`, Fan: `off`, TV: `off`	30

oped a smart home prototype using a Raspberry Pi 5 with 8 GB RAM as the edge computing device. Three appliances, *i.e.,* a light, a TV, and a fan, were connected to the edge system through `ESP8266 NodeMCU` microcontrollers using 5V mechanical `relay` modules. Communication between the Raspberry Pi and the IoT nodes was established over Wi-Fi using the MQTT protocol.

Each IoT device subscribed to a unique MQTT topic for receiving control commands and published its current status to a dedicated topic. The user issued natural-language commands through a local text-based interface, which were converted into structured prompts and processed by the LLMs. The resulting response was parsed into `JSON` format and translated into MQTT commands, which were then sent to the appropriate devices. We tested two different LLMs, **Llama 3 8B** and **Gemma 2B**. Each model was evaluated with three representative commands related to smart-room setup. For each command, we recorded the LLM response and the inference time from prompt submission to response generation. The expected device states were predefined for each use case. Table II summarizes the results. Since this evaluation uses a small number of commands without repeated trials, the results should be interpreted as a preliminary feasibility demonstration rather than a statistically complete performance evaluation.

From Table II, Llama 3 produced exact device-state matches for two out of three commands, while Gemma 2B produced an exact match for one out of three commands. At the individual-device level, Llama 3 correctly predicted eight out of nine device states, whereas Gemma 2B correctly predicted seven out of nine device states. These preliminary results indicate that the larger model produced more accurate control decisions in this small command set. However, the evaluation is not sufficient to claim general semantic accuracy because it does not include a large command set, repeated runs, confidence intervals, variance measurements, or comparison with rule-based and cloud-hosted baselines.

The inference time of Llama 3 was significantly higher, ranging from *126* to *208 seconds*. In contrast, Gemma 2B demonstrated lower inference time, ranging from *28* to *30 seconds* across the tested commands. This shows a trade-off between model size, response quality, and inference time on the Raspberry Pi 5 edge platform. However, the reported values correspond only to LLM inference time. They do not include the complete end-to-end latency of the system, such as prompt construction delay, MQTT transmission delay, command propagation delay, relay actuation delay, or queuing delay. Therefore, a complete latency-budget analysis is required before making strong claims about real-time deployment.

Overall, the results indicate that LLMs can be integrated with MQTT-based IoT control systems at the edge. Llama 3 provided more accurate responses in this small test, but required higher computation time, whereas Gemma 2B provided faster local inference with more frequent semantic deviations. This trade-off between inference quality and latency is important when selecting models for edge-based IoT applications. In a more complete evaluation, these results should be extended using more commands, repeated trials, token-count normalization, structured-output validation, full pipeline latency measurement, and baseline comparison against deterministic rule-based controllers and cloud-hosted LLMs.

To further illustrate how context-aware prompt construction can support different actions for the same or similar user intent, Table III presents representative examples. These examples show how command interpretation may depend on the current device state, sensor readings, and predefined rules available at the edge device.

These examples distinguish simple command translation from context-aware control. For example, the command "`Make the room comfortable`" does not explicitly specify any device action, and its interpretation depends on the available sensor value, such as room temperature. Similarly, the command "`Save energy`" depends on occupancy and current device states. Such cases should be included in a larger evaluation to verify whether the LLM can use contextual information correctly rather than merely mapping fixed commands to predefined actions.

The output of one user command, "`I want to sleep now`", is shown below. In this example, the expected sleep-mode action is to turn off the light, TV, and fan. Therefore, an output that keeps the light on or only dims the light should be treated as a mismatch with the predefined expected state.

```
# User Input
Enter your smart home command: I want to sleep now

# Sending to Llama 3...
Raw Output from Llama 3:
{
  "description": "Prepare the room for sleep by
    turning off active devices.",
  "commands": [
    {"device": "light", "action": "off"},
    {"device": "tv", "action": "off"},
    {"device": "fan", "action": "off"}
  ]
```

TABLE III
ILLUSTRATIVE EXAMPLES OF CONTEXT-AWARE SMART-HOME COMMAND INTERPRETATION

User Command	Available Context	Expected Device Action
1. `I want to sleep now`	Light: `on`, Fan: `off`, TV: `on`	Light: `off`, Fan: `off`, TV: `off`
2. `Set the room for Study`	Light: `off`, Fan: `off`, TV: `on`	Light: `on`, Fan: `on`, TV: `off`
3. `Make the room comfortable`	Temperature: `32°C`, Fan: `off`	Fan: `on`
4. `Save energy`	Occupancy: `not detected`, Light: `on`, TV: `on`, Fan: `on`	Light: `off`, TV: `off`, Fan: `off`

```
}

# Post processing of LLM output
Description: Prepare the room for sleep by turning
    off active devices.

Commands:

Device: light | Desired: off | Current: on
Turning OFF light

Device: tv | Desired: off | Current: on
Turning OFF tv

Device: fan | Desired: off | Current: off
No action needed for fan

# Final Action
Light = Turn Off
TV = Turn Off
Fan = No Action
```

Listing 4. Example structured response for the sleep command

IV. ADVANTAGES OF USING LLMS IN IOT NETWORKS

Improved Decision Making: LLMs can improve IoT decision-making by interpreting user intent together with environmental context. Instead of relying only on fixed command-action rules, an LLM can reason over the user's request, current sensor readings, and device states. For example, a command such as "`prepare the room for sleep`" can be translated into coordinated actions such as turning off the TV, switching off or dimming the light, and adjusting the fan without requiring the user to specify each action separately.

Local Processing and Reduced Cloud Dependence: When deployed at the edge, LLMs can process user commands locally and reduce dependence on cloud services. This can improve privacy, reduce external connectivity requirements, and allow the system to continue operating even when cloud access is limited. However, the actual response time depends on several factors, including model size, quantization level, prompt length, hardware capability, and communication overhead within the IoT network.

Enhanced User Experience: LLM-based natural-language interfaces allow users to interact with IoT systems more intuitively through simple text or voice commands. Users do not need to remember device-specific commands or manually configure multiple appliances. By incorporating contextual and historical information, such systems can also support personalized automation, such as adapting room settings according to user preferences, time of day, or previous usage patterns.

Scalability and Adaptability: LLMs provide a flexible interface for controlling multiple devices and services without manually defining a large number of fixed rules. As new devices are added to an IoT environment, the LLM can help interpret new device names, functions, and control patterns through structured prompts. This makes LLM-based control useful for evolving smart homes, industrial deployments, and other IoT systems where device configurations may change over time.

V. USE CASES

This section discusses representative use cases where LLMs can be integrated into edge-based IoT systems.

Industrial IoT: In Industrial IoT (IIoT), LLMs can support real-time monitoring, anomaly detection, predictive maintenance, and operator assistance. Edge-level LLM agents may analyze local sensor readings, machine states, alarms, and event logs to generate context-aware summaries or recommendations. Cloud-assisted models can further combine historical logs, maintenance records, and operational knowledge for higher-level decision support. Such systems can help detect early signs of equipment failure, summarize machine status, and translate natural-language operator instructions into machine-executable actions. Resource-efficient and domain-specialized LLMs can therefore improve usability, transparency, and operational intelligence in industrial environments.

Smart Healthcare: In Internet of Medical Things (IoMT) systems, LLMs can assist with patient monitoring, alert generation, health-data summarization, and personalized support. Edge-deployed models may process physiological signals such as heart rate, oxygen saturation, glucose level, or activity patterns to generate context-aware alerts while preserving user privacy. Conversational LLM interfaces can also support symptom reporting, medication reminders, and patient education in a more natural manner. However, healthcare applications require strong reliability, privacy protection, clinical validation, and human oversight before deployment in real-world settings.

Semantic Communication: LLMs can also support semantic communication in IoT networks, where the goal is to transmit meaningful information rather than raw data. For example, instead of sending a full camera image, an edge device may transmit compact semantic features such as object descriptions, event summaries, or anomaly reports. This can reduce bandwidth usage and network congestion while preserving task-relevant information [7], [8]. Beyond compression, LLMs can help prioritize important events, summarize sensor streams, and support context-aware data exchange. In future 5G and

6G IoT systems, LLM-assisted semantic communication may improve scalability, responsiveness, and bandwidth efficiency.

VI. Open Challenges and Issues

Although LLMs can enhance edge-based IoT systems, several challenges must be addressed before practical deployment. This section highlights two major issues.

A. Privacy and Security

LLM-enabled IoT systems may process sensitive data such as camera feeds, health measurements, location information, and daily activity patterns. Therefore, privacy depends strongly on where the data is processed and who can access it. Running LLMs locally on edge devices can keep data within the user's trusted environment and reduce exposure to third-party services [9]. In contrast, cloud-based LLMs may provide higher performance and scalability, but they introduce risks related to data transmission, external storage, and unauthorized access.

A practical solution may be a hybrid architecture, where sensitive preprocessing is performed locally and only encrypted summaries or non-sensitive features are sent to the cloud when needed. Security is also critical because LLMs can be vulnerable to prompt injection, malicious inputs, and unsafe command generation. For example, a manipulated command may attempt to bypass safety rules or force the system to reveal private information. Therefore, LLM-based IoT systems should include input filtering, access control, sandboxing, structured-output validation, and rule-based safety checks before executing physical actions.

B. Accuracy and Reliability in LLM-Enabled IoT Systems

Reliability is essential in IoT applications because incorrect LLM outputs may lead to physical consequences. For example, a wrong interpretation of smoke, medical, or industrial sensor data may trigger unsafe actions, missed alarms, or unnecessary interventions. In smart-home settings, an incorrect response may only cause inconvenience, but in healthcare or industrial environments, the consequences can be more serious. Therefore, LLM-generated commands should not be executed blindly in safety-critical environments.

Reliable deployment requires domain adaptation, structured outputs, deterministic safety verification, and extensive testing. Fine-tuning or adapting LLMs using domain-specific sensor logs can improve task accuracy [10], [11]. In addition, rule-based verifiers should check whether LLM-generated actions satisfy physical constraints and safety policies before they are sent to actuators. Finally, evaluation benchmarks should measure not only language quality but also task accuracy, latency, robustness, resource usage, structured-output validity, and behavior under unusual or degraded network conditions.

VII. Conclusion

In this article, we discussed how LLMs can enhance the efficiency and intelligence of edge-based IoT systems. We proposed a modular framework that integrates LLMs into IoT architectures and outlined the role of each component within the system, from data collection and prompt generation to response handling and device control. We also explored the advantages of using LLMs for natural language-based interaction, personalized automation, and context-aware decision-making in IoT environments. Finally, we highlighted a few use cases of LLM-based IoT systems, key challenges such as privacy, security, reliability, and model optimization that must be addressed to enable safe and effective deployment of LLM-enabled IoT systems in real-world scenarios.

Acknowledgment

This work was supported in part by the Anusandhan National Research Foundation (ANRF), Government of India, under Grant ANRF/ECRG/2024/001053/ENS, and in part by the Faculty Research Scheme, IIT (ISM) Dhanbad, under Grant MISC 0111.

References

[1] A. Kalita and M. Khatua, "6TiSCH – IPv6 Enabled Open Stack IoT Network Formation: A Review," *ACM Trans. Internet Things*, vol. 3, no. 3, jul 2022.

[2] I. Kok, O. Demirci, and S. Ozdemir, "When IoT Meet LLMs: Applications and Challenges," 2024. [Online]. Available: https://arxiv.org/abs/2411.17722

[3] Y. Gao, Z. Ye, M. Xiao, Y. Xiao, and D. I. Kim, "Guiding IoT-Based Healthcare Alert Systems with Large Language Models," 2024. [Online]. Available: https://arxiv.org/abs/2408.13071

[4] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, and K. Huang, "Pushing Large Language Models to the 6G Edge: Vision, Challenges, and Opportunities," 2024. [Online]. Available: https://arxiv.org/abs/2309.16739

[5] T. An, Y. Zhou, H. Zou, and J. Yang, "IoT-LLM: Enhancing Real-World IoT Task Reasoning with Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2410.02429

[6] A. Hazra, A. Kalita, and M. Gurusamy, "Meeting the Requirements of Internet of Things: The Promise of Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 7474–7498, 2024.

[7] S. Barbarossa, D. Comminiello, E. Grassucci, F. Pezone, S. Sardellitti, and P. Di Lorenzo, "Semantic Communications Based on Adaptive Generative Models and Information Bottleneck," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 36–41, 2023.

[8] A. Kalita, "Large Language Models (LLMs) for Semantic Communication in Edge-based IoT Networks," 2024. [Online]. Available: https://arxiv.org/abs/2407.20970

[9] K. Khatiwada, J. Hopper, J. Cheatham, A. Joshi, and S. Baidya, "Large Language Models in the IoT Ecosystem – A Survey on Security Challenges and Applications," 2025. [Online]. Available: https://arxiv.org/abs/2505.17586

[10] R. Birkmose, N. M. Reece, E. H. Norvin, J. Bjerva, and M. Zhang, "On-Device LLMs for Home Assistant: Dual Role in Intent Detection and Response Generation," 2025. [Online]. Available: https://arxiv.org/abs/2502.12923

[11] Z. Yu, Z. Wang, Y. Li, H. You, R. Gao, X. Zhou, S. R. Bommu, Y. K. Zhao, and Y. C. Lin, "EDGE-LLM: Enabling Efficient Large Language Model Adaptation on Edge Devices via Layerwise Unified Compression and Adaptive Layer Tuning and Voting," 2024. [Online]. Available: https://arxiv.org/abs/2406.15758