Rethinking IoT Programmability - From Things to Generative Smart Spaces

Abdelsalam (Sumi) Helal
Department of Computer Science and Engineering (DISI), University of Bologna
Bologna, Italy
sumi.helal@unibo.it

***Abstract*—*The Internet of Things has inspired compelling visions of intelligent homes, hospitals, cities, and environments, yet a fundamental systems question remains unresolved: how do we program an IoT environment whose components are heterogeneous, transient, context dependent, and only partially known at design time? This keynote position paper argues that the next step in IoT programmability is a shift from programming individual Things and pre-wired applications toward generative smart spaces that can construct and interpret their own interaction structures. We develop the position through two complementary directions. Generative MQTT (gMQTT) introduces a generative broker-control plane that synthesizes and evolves MQTT-compatible topic spaces from semantic device descriptions, runtime context, policies, and application intent. Sensor Language Modeling (SLM) treats the collective behavior of sensors as an intrinsic language whose vocabulary is grounded in static device semantics and dynamic runtime semantics and whose grammar is learned from experience. We propose that these directions form a closed semantic loop: gMQTT generates how a smart space communicates, while SLM learns what the evolving space means; learned context can in turn guide subsequent communication structures. The resulting architecture keeps Things explicit, generation inspectable, physical execution safety-bounded, and higher-level programmability centered on intent.*

Keywords*—*Internet of Things; generative smart spaces; Generative MQTT; Sensor Language Modeling; MQTT; semantic middleware; foundation models; smart spaces

I. INTRODUCTION

The Internet of Things (IoT) is often presented as a collection of connected sensing and actuation devices supported by edge, cloud, and application software. That view hides a deeper programmability problem. A smart space is a computational environment whose membership, capabilities, and context evolve while the system is running. Devices appear and disappear, expose heterogeneous services, and participate in interactions that may not have been anticipated by either the device developer or the application developer. Traditional middleware has made substantial progress in protocol translation and device discovery, but semantic rigidity and manual configuration remain persistent barriers in heterogeneous IoT systems [1].

Our earlier work approached this challenge from the bottom up. The Gator Tech Smart House demonstrated the idea of a programmable pervasive space [2]. Atlas then provided a service-oriented sensor platform in which physical Things could be represented and composed as computational entities [3]. IoT-DDL extended this architectural direction by giving Things machine-readable descriptions of identity, services, units, and semantic properties [4]. The lesson is architectural: there are no shortcuts to IoT programmability. The Thing must be explicit enough for software to discover what it is, what it can do, and how it can safely participate.

Yet explicit Things solve only half of the problem. If future IoT environments are large, transient, and opportunistic, developers cannot manually anticipate every useful interaction or permanently encode every communication path. Recent LLM-IoT systems show that language models can reason about multi-step tasks and orchestrate device capabilities [5], [6], but a model that merely calls predefined tools still inherits the interaction structure designed in advance. The stronger opportunity is to let middleware construct that structure from the semantics of the environment while keeping generative reasoning outside the final physical authority.

This paper advances a position around two emerging directions. The first direction is Generative MQTT (gMQTT) which makes the interaction space itself generative: topics, subscriptions, routing structures, and capability bindings can be synthesized and revised over time from device descriptions, runtime context, policy, and intent. The second direction is Sensor Language Modeling (SLM) which asks whether the observations of an intelligent environment form an intrinsic language whose vocabulary, grammar, and narratives can be learned. We argue that these are complementary halves of a generative smart space: gMQTT generates how the environment communicates and interacts; SLM learns what the environment is saying about itself and its occupants.

II. ARCHITECT THE THINGS, THEN GENERATE ABOVE THEM

The proposed shift does not replace device architecture with AI. It depends on stronger device architecture. Atlas and IoT-DDL provide a structured basis for discovery, interoperability, and service composition [3], [4]. SLM similarly begins from explicit device knowledge - identity, location, sensing modality, capability, relationships, units, and intended function - rather than anonymous numerical streams. The gMQTT direction uses the same kind of descriptions as evidence from which communication structures can be synthesized.

This leads to a layered view of generative smart spaces. At the bottom are physical Things and their deterministic interfaces. Above them is a semantic description layer that makes capabilities and constraints explicit. Above that sits a generative control plane that may interpret intent, propose

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

mappings, synthesize topic structures, and identify candidate orchestrations. Before any proposal reaches the physical data plane, deterministic middleware checks whether it is type-correct, authorized, policy-compliant, and safe. Reliable IoT domain-specific languages such as IRON illustrate the role that a formal or rule-based layer can play in intercepting unsafe generated control logic [7].

The distinction is important. A language model is valuable precisely where conventional middleware is weak: semantic mediation, ambiguity resolution, candidate planning, and generalization across heterogeneous descriptions. Conventional middleware is valuable precisely where probabilistic generation is weak: enforcement, lifecycle control, access control, state consistency, and repeatable execution. Generative smart spaces should therefore combine rather than conflate these roles.

III. GENERATIVE MQTT: A GENERATIVE INTERACTION SPACE

A. Why Static Topic Trees Become a Programming Constraint

MQTT is attractive because publish-subscribe communication decouples producers and consumers, but deployments normally assume that the topic hierarchy is engineered in advance. MQTT design guidance treats topic structure as an explicit system design decision [8]. This assumption is reasonable when devices and applications are stable. It becomes a programming constraint when the environment is transient, when previously unseen devices must collaborate, or when a new goal requires a communication pattern that no developer encoded.

Adaptive broker research already demonstrates that message exchange can be modified through broker-level adaptation and topic rewriting [9]. The gMQTT position extends this idea from adaptation of a known communication structure to synthesis of the communication structure itself. A newly arrived device is not merely attached to an existing branch. Its semantic description is interpreted in the context of the active environment, candidate roles are identified, and the broker-control plane can create or revise the topics, subscriptions, filters, and routing relationships required by current goals.

B. The Generative Topic Space

We use Generative Topic Space to denote the runtime artifact produced by this process. It is intentionally broader than a set of topic strings. A useful generated space may contain topic hierarchy and aliases, publisher and subscriber bindings, content or semantic filters, routing rules, quality-of-service choices, retention and lifecycle properties, access-control constraints, dependencies, and provenance. The data plane can remain MQTT-compatible while the control plane becomes substantially more semantic and adaptive.

The generative process can be expressed conceptually as G = f(D, C, I, P), where D is the set of current device descriptions, C is runtime context, I is user or application intent, and P is policy and safety constraints. G is not immediately executable. It is a candidate broker configuration that must pass deterministic validation. This distinction allows a probabilistic model to contribute semantic flexibility without turning a generated token sequence into a physical command.

C. Transient Devices and Zero-Configuration Collaboration

Transient environments provide a natural stress case. LLM-driven auto-configuration has been proposed for transient IoT collaboration in which previously unplanned devices must discover how to work together [10]. gMQTT treats the arrival or departure of such a device as a topology event. The system re-evaluates which semantic roles are now possible, which existing interactions are affected, and whether a temporary communication structure should be created, revised, merged, or retired.

The objective is not simply configuration convenience. The deeper objective is to make topology a programmable runtime object. If a wearable, robot, or guest sensor enters a space, the middleware should be able to reason from its description to a bounded role in the local interaction graph without requiring a human to redesign the broker namespace. Likewise, when the device leaves, the interaction structure should contract without leaving stale routes or permissions.

D. From Tool Use to Procedural Capabilities

Recent LLM-IoT systems show the value of autonomous agents that select and sequence device functions for complex tasks [5], [6]. gMQTT provides a middleware substrate for this procedural view. Rather than exposing every device function as an isolated tool, semantic descriptions can support discoverable capability bundles or skills. An intent such as preparing a room for a private teleconsultation may require sensing, lighting, display, access control, and notification functions. The planner proposes a capability composition; gMQTT materializes the communication structure that connects those capabilities; deterministic middleware enforces the allowed execution envelope.

This separation also permits distributed execution. Higher-level semantic planning may occur where larger models are available, while the validated topic topology and action bindings can be instantiated at the broker and edge. The proposal document motivating gMQTT explicitly frames this as a cloud planning and edge execution split, with safety-bounded execution at the edge. The architectural principle is more important than the placement choice: semantic generation and physical execution should remain separable.

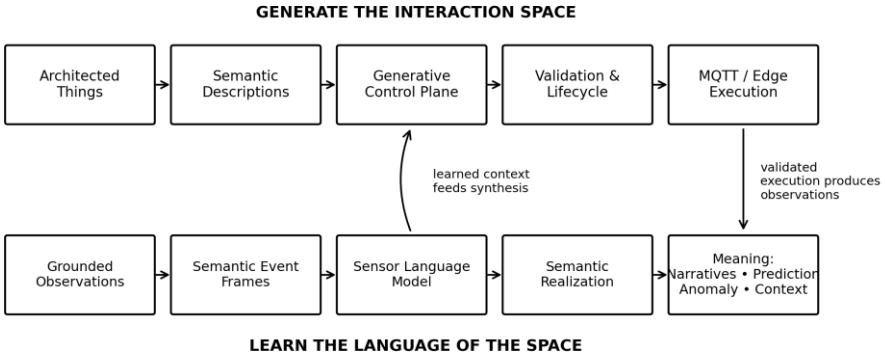


Fig. 1. Unified generative smart-space architecture. The upper path generates and validates interaction structure; the lower path constructs and learns sensor language. Both remain grounded in explicit Things and evidence.

IV. SENSOR LANGUAGE MODELING: LEARNING THE LANGUAGE OF THE SPACE

A. The Sensor-Language Hypothesis

Smart-space sensing has traditionally treated observations as numerical measurements from which predefined human activities are inferred. Activity-recognition research has used probabilistic reasoning, statistical learning, and deep neural models for this mapping [11], [12]. SLM asks a more fundamental question: could the sensors of an intelligent environment collectively be speaking their own language?

The SLM hypothesis is that sensor observations possess linguistic structure. Individual grounded events may act like words; temporal combinations like phrases; sequences of phrases like sentences; and longer histories like narratives describing the evolving state of the environment. The goal is not to force sensor data into natural language. It is to discover the latent vocabulary, syntax, semantics, and grammar already present in the collective behavior of sensing devices. Foundation-model research motivates learning latent structure from large sequential corpora [13], and transformer architectures provide mechanisms for long-range contextual dependencies [14].

B. Static and Dynamic Semantic Lexicons

Unlike natural language, sensor language is not directly observable. A raw sensor value does not say what device produced it, what entity it represents, where it is located, or what a change means. SLM therefore constructs vocabulary from two complementary semantic layers. The static lexicon comes from semantic middleware and device descriptions such as Atlas and IoT-DDL [3], [4]. It captures relatively stable properties including identity, location, modality, capabilities, relationships, units, and intended function.

The dynamic lexicon emerges from runtime behavior. Concepts such as persistent, frequently activated, unusually quiet, intense, intermittent, regular, or anomalous depend on temporal history and context. They cannot be permanently encoded in a device descriptor. SLM combines the raw observation, static semantics, and dynamic semantics into a Semantic Event Frame. This frame acts as the grounded linguistic unit while preserving explicit links to the originating sensor evidence.

C. Semantic Construction Before Semantic Learning

A central architectural choice in SLM is to separate semantic construction from semantic learning. Raw observations are first enriched into Semantic Event Frames. Frames can then produce linguistic candidates and be aggregated into episode context packs that provide longer temporal context. Only after this grounding step does the persistent Sensor Language Model learn vocabulary, phrases, temporal grammar, semantic relationships, dynamic modifiers, recurring episodes, and narrative patterns.

This architecture places foundation-model reasoning after grounding rather than directly on raw streams. It therefore differs from approaches that convert sensor events into natural-language descriptions for activity recognition, such as TDOST [15], and from sensor-language foundation models that learn aligned representations from paired sensor and text data, such as SensorLM [16]. In SLM, human language is an interpretive bridge; the primary object being learned is the intrinsic language of the sensing environment.

D. Beyond Activity Recognition

Once the sensing infrastructure has a learned language, activity recognition becomes one interpretation rather than the organizing objective. A language model of the space can support narrative generation, semantic querying, anomaly detection, prediction, question answering, and discovery of previously unspecified behavioral regularities. The SLM prototype described in the source paper demonstrates the feasibility of constructing evidence-backed semantic representations from realistic smart-home traces while maintaining traceability to raw observations. The full persistent learning core remains future work.

V. CLOSING THE LOOP BETWEEN GMQTT AND SLM

The strongest version of the generative smart-space position emerges when gMQTT and SLM are treated as a closed semantic loop rather than independent projects. As illustrated in Fig. 1, gMQTT is constructive: it creates a communication topology from the Things that exist, the context that is known, and the intent that must be served. SLM is interpretive: it learns structure from the grounded observations produced by

those Things. One generates the interaction space; the other learns the meaning space.

The feedback from SLM to gMQTT is especially important. Static descriptions may say that a sensor measures motion in a corridor, but SLM may learn that the sensor is currently unusually quiet, that it repeatedly co-occurs with another device in a recurring episode, or that a previously unseen pattern is emerging. These learned semantic constructs can become contextual inputs to the next gMQTT synthesis cycle. The broker may create a temporary topic, add a subscription, narrow a filter, or expose a learned episode to an application because the meaning of the environment has changed.

The reverse direction is equally useful. A generated topic space contains explicit information about intended coordination. It can therefore help SLM distinguish purposeful interaction from mere temporal co-occurrence. The creation and retirement of a generated topology can also provide episode boundaries. Provenance from gMQTT - why a route was created, which intent justified it, and which devices were selected - becomes additional semantic evidence for interpreting subsequent observations.

This loop creates a new programming abstraction. Instead of programming a fixed graph of devices and topics, a developer states intent and constraints over a semantically grounded environment. The system decides which Things are relevant, generates an interaction structure, observes the resulting behavior, learns its sensor-language meaning, and revises the interaction structure when the learned context warrants change. The abstraction is high-level, but its correctness still depends on explicit lower-level representations.

VI. FIVE RESEARCH PROPOSITIONS

The following propositions sharpen the position into testable systems claims. They are not presented as results already established by the two source projects; rather, they articulate the research program implied by combining them.

P1. Topic Topology Can Become a Runtime Synthesis Object

A topic hierarchy need not remain a design-time artifact. Given sufficiently rich device descriptions, runtime context, intent, and policy, an MQTT-compatible interaction topology can be synthesized at runtime and validated before activation. The key comparison is not merely whether generation is possible, but whether generated configurations reduce manual integration while preserving functional correctness, bounded latency, and predictable lifecycle behavior relative to static or adaptive broker baselines [9], [10].

P2. Semantic Grounding Can Bound Generative Freedom

The useful role of a generative model is constrained semantic mediation, not unrestricted physical control. If every proposed topic, capability binding, and orchestration step is grounded in explicit device descriptions and checked by deterministic policy and safety mechanisms, the system can exploit model flexibility without making model output the final execution authority. This proposition connects the semantic role of IoT-DDL [4] with the safety role illustrated by IRON [7].

P3. Sensor Grammar Can Be Learned Above Grounded Events

The SLM hypothesis predicts that recurrent temporal and semantic structures will emerge more reliably when learning operates over Semantic Event Frames than when it operates directly over anonymous raw sensor activations. The relevant evidence is not natural-language fluency. It is whether learned phrases, grammar, episode structures, and anomaly baselines are stable, predictive, transferable, and traceable to observations.

P4. Learned Sensor Meaning Can Improve Runtime Orchestration

If SLM discovers dynamic semantic concepts that are absent from static device descriptions, those concepts should improve gMQTT synthesis. A system that knows only device type and location may construct one topology; a system that additionally knows that a sensor is behaving abnormally or that a recurring episode is underway may construct a more appropriate one. This proposition makes the SLM-to-gMQTT feedback path experimentally testable.

P5. Intent-Level Programming Requires Explicit Lower Layers

Higher-level intent does not eliminate architecture; it increases the need for it. Intent-level programmability will be dependable only if generated structures can be traced downward to explicit Things, capabilities, policies, observations, and execution constraints. The proposition is deliberately counter to an 'LLM as middleware' interpretation. The model is a component of the control plane, while the architecture remains the source of enforceable truth.

VII. EVALUATION AGENDA

A. gMQTT: Correctness, Adaptation, and Convergence

A gMQTT evaluation should begin with controlled transient-device scenarios in which devices join, leave, or change role while applications maintain active goals. Functional correctness measures whether the generated topology enables the requested interaction without syntax, type, or binding failures. Adaptation latency measures the time from a topology-changing event to a validated usable configuration. Resource overhead measures the broker, edge, and model cost introduced by generation.

A less obvious metric is convergence. A generative broker-control plane must know when to stop changing itself. Repeated topic creation, merging, rewriting, and retirement can otherwise produce oscillation or uncontrolled growth. After a change in membership or intent, the system should reach a stable configuration unless new evidence justifies another transition. Equivalent intent should also lead to behaviorally equivalent outcomes even when the planner generates different surface representations.

B. SLM: Language Quality Rather Than Label Accuracy Alone

SLM requires evaluation beyond conventional activity-recognition accuracy. The learned vocabulary should exhibit semantic consistency across time; learned phrases and

grammar should capture recurring temporal structures; narratives should preserve evidence links; anomaly judgments should correspond to deviations from learned sensor-language regularities; and learned structures should support prediction or retrieval. Activity recognition remains a useful downstream task, but it should not define the language.

Cross-space transfer is another central test. TDOST demonstrates that semantic textualization can improve layout-agnostic activity recognition [15], while SensorLM demonstrates transferable sensor-language representations [16]. SLM asks which elements of an intrinsic sensor language transfer across homes or spaces when devices, layouts, and local routines differ. A useful model may combine a reusable grammatical core with environment-specific vocabulary and narratives.

C. Joint Evaluation of the Closed Loop

The combined system should be evaluated with perturbations that affect both communication and meaning: a guest device appears, a sensor becomes unreliable, a recurring episode changes, or a new intent is introduced. The experiment should compare a static MQTT baseline, a gMQTT-only system, an SLM-only interpretive layer over static messaging, and the closed-loop gMQTT+SLM architecture. The central question is whether learned meaning improves topology synthesis and whether generated topology improves semantic interpretation without compromising safety or stability.

VIII. DISCUSSION: WHAT CHANGES IN IOT PROGRAMMABILITY?

The proposed architecture changes the unit of programming. Early programmable smart spaces focused on exposing Things and composing their services [2], [3]. gMQTT raises the unit from a service call to an interaction topology. SLM raises the unit of interpretation from a sensor value or activity label to a learned environmental language. The programmer increasingly works with goals, semantic constraints, and expected behavior rather than hard-coded device addresses and topic strings.

This does not imply that applications disappear. Some interactions are stable enough to remain explicitly engineered, and safety-critical functions may appropriately avoid generation. Generative mechanisms are most valuable where heterogeneity, transience, or contextual novelty makes preconfiguration expensive or incomplete. A practical system should therefore allow static, adaptive, and generative interaction structures to coexist.

Nor does the position imply that natural language becomes the internal representation of the IoT. gMQTT may accept natural-language intent, but its output is a structured broker configuration. SLM may expose human-readable narratives, but its core language is grounded in sensor semantics and temporal behavior. Natural language is an interface and interpretive bridge, not a substitute for the internal computational structures of the smart space.

The larger implication is that IoT middleware can evolve from a passive integration substrate into an active architectural participant. It can construct interaction structures, observe the consequences, learn recurring semantic organization, and revise those structures while preserving explicit constraints. That is a qualitatively different role from protocol translation or static service discovery.

IX. CONCLUSION

IoT programmability should no longer be framed only as the composition of predefined device services. The physical Things still require explicit architectures, semantic descriptions, and deterministic interfaces, but the interaction structures above them should be able to emerge and evolve with the environment. gMQTT represents this constructive half of the vision by turning the MQTT topic space into a generated, inspectable, policy-bounded middleware artifact.

At the same time, intelligent environments should not be limited to mapping numerical streams into predefined activity labels. SLM represents the interpretive half by constructing grounded linguistic units from static and dynamic semantics and learning the vocabulary, grammar, and narratives of the sensing environment. The resulting language can support activity understanding, prediction, anomaly detection, semantic querying, and narrative reasoning.

Together, the two directions suggest a concise agenda: architect the Things, generate the interaction space, learn the language of the environment, and program at the level of intent. This is not a call to place an unconstrained LLM in the physical control loop. It is a call for middleware in which generative intelligence is useful because it is surrounded by explicit semantics, provenance, inspectable structures, deterministic validation, and safety boundaries. The long-term goal is an IoT that can organize how it communicates, learn what its environment means, and adapt both under explicit architectural control.

REFERENCES

[1] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K.-I. Kim, "A comprehensive review of Internet of Things: Technology stack, middlewares, and fog/edge computing interface," Sensors, vol. 22, no. 3, p. 995, 2022.

[2] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator Tech Smart House: A programmable pervasive space," Computer, vol. 38, no. 3, pp. 50–60, 2005.

[3] J. King, R. Bose, H. Yang, S. Pickles, and A. Helal, "Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces," in Proc. 31st IEEE Conf. Local Computer Networks, 2006, pp. 630–638.

[4] A. E. Khaled, A. Helal, W. Lindquist, and C. Lee, "IoT-DDL: Device Description Language for the 'T' in IoT," IEEE Access, vol. 6, pp. 24048–24063, 2018.

[5] H. Cui, Y. Du, Q. Yang, Y. Shao, and S. C. Liew, "LLMind: Orchestrating AI and IoT with LLM for complex task execution," IEEE Communications Magazine, vol. 63, no. 4, pp. 214–220, 2025.

[6] D. Rivkin, F. Hogan, A. Feriani, A. Konar, A. Sigal, X. Liu, and G. Dudek, "AIoT smart home via autonomous LLM agents," IEEE Internet of Things Journal, vol. 12, no. 3, pp. 2458–2472, 2025.

[7] D. Cacciagrano and R. Culmone, "IRON: Reliable domain specific language for programming IoT devices," Internet of Things, vol. 9, p. 100020, 2020.

[8] Amazon Web Services, "Designing MQTT topics for AWS IoT Core," AWS Whitepaper, 2020.

[9] H. H. Hassan, G. Bouloukakis, L. Scalzotto, N. Khaled, D. Conan, A. Kattepur, and D. Belaïd, "A message broker architecture for adaptive

data exchange in the IoT," in Proc. 2024 IEEE 21st Int. Conf. Software Architecture Companion (ICSA-C), 2024, pp. 151–158.

[10] H. Shastri, W. A. Hanafy, L. Wu, D. Irwin, M. Srivastava, and P. Shenoy, "LLM-driven auto configuration for transient IoT device collaboration," 2025.

[11] E. Kim, S. Helal, and D. J. Cook, "Human activity recognition and pattern discovery," IEEE Pervasive Computing, vol. 9, no. 1, pp. 48–53, 2010.

[12] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," Sensors, vol. 16, no. 1, p. 115, 2016.

[13] R. Bommasani et al., "On the opportunities and risks of foundation models," arXiv:2108.07258, 2022.

[14] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[15] M. Thukral, S. G. Dhekane, S. K. Hiremath, H. Haresamudram, and T. Ploetz, "Layout-agnostic human activity recognition in smart homes through textual descriptions of sensor triggers (TDOST)," Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 9, no. 1, Art. no. 19, 2025.

[16] Y. Zhang et al., "SensorLM: Learning the language of wearable sensors," in Proc. 39th Annual Conf. Neural Information Processing Systems, 2026.