The American Journal of Applied Sciences
ISSN 2689-0992 | Open Access

TYPE Original Research
PAGE NO. 78-88
DOI 10.37547/tajas/Volume07Issue10-09

Godzilla Vs. Kong Escape Room: Pipelines, Firmware, And Live Systems

Dmytro Novoselskyi
Senior Software Developer, 60out Escape Rooms, Los Angeles, USA

OPEN ACCESS

SUBMITED 19 August 2025
ACCEPTED 15 September 2025
PUBLISHED 17 October 2025
VOLUME Vol.07 Issue 10 2025

CITATION
Dmytro Novoselskyi. (2025). Godzilla Vs. Kong Escape Room: Pipelines, Firmware, And Live Systems. The American Journal of Applied Sciences, 7(10), 78–88. https://doi.org/10.37547/tajas/Volume07Issue10-09

Abstract: This article examines the architecture of pipelines, embedded systems, also real-time systems when they are implemented via an event-driven approach, using the Godzilla vs. Kong escape room works as an empirical case study. The study is relevant as the location-based entertainment industry rapidly advances and technological installations escalate in complexity, where numerous heterogeneous hardware and software components reliably integrate. Customary monolithic architectures, also cloud-centered solutions, are often insufficient under stringent requirements for modularity, fault tolerance, and low interaction latency. For identifying universal principles for designing distributed real-time systems, the study will deconstruct and analyze the system architecture through a real commercial project. This work formalizes practical solutions derived from applied engineering within the context of event-driven architecture (EDA) and distributed-systems theory, contributing to scientific novelty. It becomes possible when elevating of the empirical COGS bus links nodes to a canonical architectural pattern. The principal results show the proposed model achieves extremely low response latency of about 30 ms as well as scalability plus loose coupling of modules. This model enables efficient computational load redistribution between peripheral devices and a central system. The article will help out researchers and also engineers that work for embedded systems and real-time systems as well as develop engaging interactive environments.

Keywords: Event-driven architecture, embedded systems, real-time systems, distributed systems, immersive environments.

Introduction

The location-based entertainment (LBE) industry has transformed markedly over the course of the past decade. Escape rooms now differ; they contain interactive elements, special effects, detailed plotlines within engaging high-technology environments rather than simple puzzles or mechanical locks (Makri et al., 2021). Since digital technologies with a broad spectrum integrate microcontrollers, single-board computers, custom user interfaces, sensors, and actuators, this significant advance has continued to progress.

Integrating diverse hardware as well as software components into just a single fault-tolerant system becomes a truly fundamental engineering problem on account of this complexity (Pfeifer et al., 2021). For bespoke installations of this caliber here, truly customary monolithic control systems often do prove not sufficiently flexible, or scalable, and even reliable because only a single central controller handles absolutely all logic. Failures at the central node can precipitate system-wide outages. Complex as well as risk-laden modifications to existing code are demanded when adding new interactive elements.

As an exemplary case study to address this problem, the present research examines the Godzilla franchise. That research features Godzilla versus. Kong escape room, namely GvK, was implemented by 60Out Escape Rooms. The design of this work provides a concrete, analytical foundation. It comprises several distributed and autonomous subsystems.

For distilling its key design principles, deconstructing and analyzing the GvK system architecture is the primary objective. Set forth at this point are the tasks as follows:

1. Characterize the COGS orchestration system within formal architectural patterns.

2. Analyze exact design decisions aimed at real-time performance and network-load management in distributed subsystems.

3. Evaluate system performance in the context of human–computer interaction (HCI) and compare it to established academic criteria.

The scientific novelty lies in presenting a detailed technical analysis of a complex, commercially successful immersive system and formalizing its practical engineering solutions through established concepts from computer science, such as event-driven architecture (EDA) and distributed systems theory. During the development of GvK, the engineers, confronted with practical challenges, intuitively arrived at a system they described as COGS event bus (a local publish–subscribe orchestration bus). This approach, born of engineering necessity, is in essence a classical realization of the publish–subscribe pattern at the heart of EDA (TIBCO, n.d.). Thus, the present study elevates an ad-hoc practical solution to a validated architectural archetype, demonstrating the organic applicability of EDA to the specific demands of the LBE industry: managing heterogeneity, ensuring modularity, and reacting in real time.

Methodology

The work is grounded in a qualitative single-case study. This approach is optimal for deep, holistic examination of a contemporary phenomenon in its real-world context—especially salient when analyzing complex engineering systems in which design decisions are tightly interwoven with practical constraints and objectives.

The principal data source is a detailed technical description of the GvK project provided by its lead developer. This document views architectural decisions with details their implementation including selection of hardware and fragments of firmware also optimizes performance. This source gives one kind of accuracy. Granularity is unattainable from external observation.

It was systematically reviewed peer-reviewed literature using Scopus, IEEE Xplore, and ACM Digital Library to establish the theoretical foundation and contextualize practical decisions. Keywords which were grouped into three core domains became a focus of the search.

1. System architecture: Event-Driven Architecture, also distributed interactive systems, along with a message bus, and state synchronization. These sources were the theoretical basis to analyze the COGS system. The COGS system is described within TIBCO (n.d.).

2. For Raspberry Pi, for Arduino, for IoT orchestration, for escape room technology: systems are embedded and IoT. This body of work let us analyze the hardware's implementation along with embedded devices' interacting principles (Pfeifer et al., 2021).

3. Human–computer interaction (HCI): low-latency UI, perceptual thresholds, system response time. Literature in this area was used to quantitatively assess user-interface quality and its compliance with human

psychophysiological perception thresholds (Attig et al., 2017).

The analytical framework is built on aligning practical implementations with theoretical concepts extracted from secondary sources. In particular, COGS is analyzed as an EDA realization, the game cluster and lava simulation are examined as distributed real-time systems, and the Starship user interface is evaluated according to design principles for low-latency HCI systems.

Results And Discussion

The COGS system is the central nervous system of the GvK escape room. It functions as an orchestration bus responsible for synchronizing the state of independent modules by transmitting small state updates. Examples include player-readiness information, score changes, or narrative signals (e.g., all units READY → PREPARE → START).

Under formal analysis, the COGS architecture fully conforms to the principles of event-driven architecture (EDA) (Geeks for Geeks, 2025). In this model, the first component is the event producers—i.e., discrete nodes such as rhythm-game consoles (registering button presses), game-logic controllers (starting a new scene), or the operator panel (issuing a SKIP command). Second are the events themselves: the minor state updates are the events—atomic packets encoding a system-state change (e.g., score_change, gate_state_updated). Third is the event broker: COGS plays the role of the central event broker or message bus, routing events from producers to consumers who need no direct knowledge of one another. Finally, there come the event consumers: nodes such as a Raspberry Pi, responsible for the leaderboard (subscribed to score_change), or Arduino Due controllers (subscribed to gate_matrix updates), that react upon receipt of relevant information.

The key advantage lies in modularity and loose coupling. EDA enables the development, testing, and modification of firmware for a single module, such as the lava simulation, independently of other modules, like the starship interface. Modularity is crucial for managing complex systems with numerous interactive elements, as well as for parallel teams to perform workflows (Geeks for Geeks, 2025).

Industrial platforms fall short of a customized COGS solution. The reason that this is occurring is because of a deliberate engineering trade-off. Either DDS or CORBA can be examples of heavyweight middleware, which is often relied upon by high-level distributed systems. Although developers face expenses, a steep learning curve, and significant performance overhead (Schmidt, 2002), these systems offer guaranteed delivery, similar to QoS. Modern IoT platforms provide easy integration; however, reliance on cloud infrastructure creates potential latency and failure points that are unsuitable for a local, real-time application (Sharma, 2023).

COGS occupies the optimal middle ground. It is more reliable and scalable than direct peer-to-peer links among nodes, yet it avoids the excess complexity and overhead associated with enterprise software or cloud services. This lightweight, local EDA is an ideal pattern for LBE's specific domain, balancing power against pragmatism. A comparative analysis of orchestration approaches is presented in Table 1.

Table 1. Comparative analysis of architectural approaches to orchestration

Architectural pattern	Latency	Scalability	Modularity / Loose coupling	Development complexity	Hardware costs	Fault tolerance
Monolithic control	Low (in-process)	Low	Very low	High (as scale increases)	Low	Low (single point of failure)
Peer-to-Peer	Low	Medium	Medium	Medium (N^2 connections)	Low	High (local failures)
Lightweight EDA (COGS)	Very low (local network)	High	Very high	Medium	Low	Medium (broker = single point of failure)

Enterprise middleware (DDS / CORBA)	Low–Medium	Very high	Very high	Very high	Medium	Very high
Cloud IoT platform	High (internet latency)	Very high	Very high	Low	Low	High (depends on provider)

The eight-player rhythm game is implemented as a tiny distributed system. The architecture comprises eleven Raspberry Pi nodes eight for game consoles, one host node for handling game logic, one regarding the leaderboard, along with one regarding the cinematic display.

The preliminary assessment revealed an important impediment to performance. Every actuation sent by eight players across the network resulted in network congestion, inducing large load spikes and intolerable jitter. Speed of reaction is foremost inside distributed interactive systems given this standard quandary.

The solution entailed a radical redesign of the data pipeline. Instead of transmitting raw input events, each console's Raspberry Pi processes button presses locally. The input arrives directly via a USB HID encoder, allowing timing judgments to be made on the device itself. Only after local processing is a compact score update sent to COGS.

This is an instance of reducing network load through edge processing. It shifts computational burden from the central network to endpoint nodes. The design pattern minimizes latency, ensuring near-zero console delay, and guarantees a quiet, predictable network that carries only essential orchestration commands. This approach aligns with load-management techniques in real-time systems described in the literature (Tanseer et al., 2020). The system architecture is shown in Fig. 1.

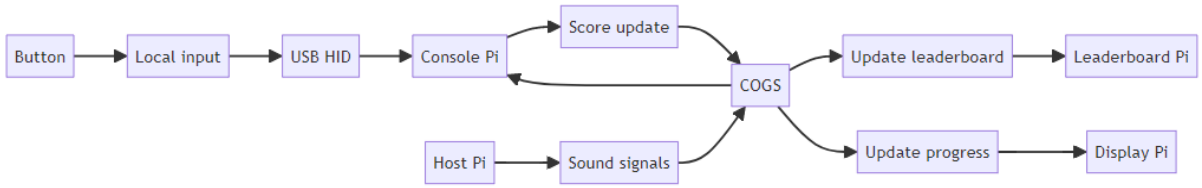

Fig. 1. Architectural diagram of the rhythm game cluster

The volcano lava effect is realized not as a pre-rendered video, but as a distributed simulation running in real-time on two peer Arduino Due microcontrollers. There is no leader–follower hierarchy; each Arduino Due independently computes and displays the heat field on the LED-strip segments assigned to it.

The system uses a hybrid control model: COGS acts as a high-level show orchestrator, sending small control packets that determine global state (e.g., which stone gates are open, the scene phase). Low-level computation—the heat-diffusion algorithm (heat[i]+=T; … cool(heat[k]);)—is executed independently and in parallel on each Arduino, as shown in Fig. 2.

```
// inject + diffuse + cool (per node)
heat[i-1]+=T/3; heat[i]+=T; heat[i+1]+=T/2;
for (int k = heatSize; k > 1; k--) heat[k] = heat[k-1];
for (int k = 0; k < heatSize; k++) heat[k] = max(4, cool(heat[k]));

// map segment with fade
leds[j] = HeatColor(qsub8(min(heat[j+off], maxT), fade));
```

Fig. 2. Heat-Field Solver with Gate-Driven Scene Control for LED Arrays

This model exploits the strengths of each component: COGS for narrative control, and Arduino for deterministic real-time hardware control (Kurkovsky & Williams, 2017).

A main issue for distributed rendering is assuring segments computed independently nearby are even. The GvK solution becomes a small mix zone where Arduinos drive matching LED strip setups. Upon state changes, both of the controllers apply complementary of the masks as the masks fade out their respective of the halves of the blend zone so that they produce the illusion of a single of a continuous lava flow.

This architecture for peer-to-peer simulation is quite efficient. It tolerates faults also very well. There is no single master controller that is responsible for rendering. Therefore, that failure of just one Arduino affects simply its portion of that display. Using small control packets from COGS, network traffic is kept minimal, while high precision without resource contention is guaranteed for the simulation via local processing. In Distributed Interactive Simulation (DIS), entities exchange state updates yet remain responsible for their own simulation loops.

Starship UI fully interacts as an application that is running on a Raspberry Pi, in place of a passive video file. User input creates on-screen feedback after approximately 30 ms using 60 fps updates.

Latency influences user performance also the subjective feeling of immersion within HCI (Attig et al., 2017). Response time for the system must fall to a point below human perceptual thresholds. Then, an action seems instantaneous in time.

Classical guidelines cite a threshold of about 100 ms for interactions to feel immediate upon comparison with HCI principles. Newer modality-specific studies set tighter limits. Ideally, visual feedback ≤ 85 ms is best and haptic feedback must be < 50 ms with 25 ms needed in some contexts (Attig et al., 2017).

For GvK, the measured ~30 ms interface latency is within the optimal range for high-quality visual, and even haptic-like, interactions, well below the general 100 ms threshold. The high level of engineering refinement for the user experience corroborates this result. This evidence supports the developer's claim that every press feels instantaneous.

The Raspberry Pi is by far an inexpensive platform. Engineering with meaning enabled performance of such a nature on it. It is enabled through the running of an optimized application that changes state atomically by using state diffs in place of redrawing the entire screen, and communicating with COGS's low-overhead protocol. Careful software design does show that stringent real-time HCI requirements can be met even without costly specialized hardware.

A hybrid federated architecture is what the GvK system must be regarded as being. It unites general-purpose single-board computers' (Raspberry Pi) strengths for user interfaces plus complex logic. It uses deterministic real-time capabilities of microcontrollers (Arduino) for direct hardware control also (Kurkovsky & Williams, 2017). A lightweight local event bus, COGS, synchronizes this entire federation, ensuring that components remain loosely coupled.

Conclusion

Theoretical foundations central to the article are event-driven architecture (EDA), distributed-systems principles, human computer interaction (HCI) latency requirements, and also edge processing. EDA provides modularity and loose coupling to simplify independent development and runtime composition distributed-systems theory frames trade-offs between local determinism and global orchestration to achieve scalability and fault tolerance edge processing moves temporal decision logic to endpoints to reduce network load and jitter and HCI thresholds (commonly cited ≤100 ms, with modality-specific targets substantially lower and with the study reporting an operational target near 30 ms) define the latency budget necessary for perceived immediacy.

In the Godzilla vs. In Kong case these theories are instantiated concretely. In just a federated topology which links Raspberry Pi and Arduino nodes, just a lightweight local event bus (COGS) functions as the orchestration layer: Arduino controllers perform deterministic, local simulations for LED effects, while Raspberry Pi units execute interface logic and then transmit compact state diffs to the bus. This combination using edge processing and brokered events reduced network spikes, constrained jitter, produced an observed interface latency of ≈30 ms, and it yielded graded fault tolerance through distributed responsibility for subsystems.

Therefore, the examined theories not only explain the GvK engineering choices but form a transferable architectural template as well. These theories may apply

to many of the similar location-based entertainment systems. Lightweight local EDA along with endpoint processing and with latency-aware HCI design make up that minimal set of reproducible principles. These principles address themselves to the principal metrics of such projects: modularity, low latency, as well as controlled network load.

References

1. Attig, C., Rauh, N., Franke, T., & Krems, J. F. (2017). System Latency Guidelines Then and Now – Is Zero Latency Really Considered Necessary? Engineering Psychology and Cognitive Ergonomics: Cognition and Design, 3–14. https://doi.org/10.1007/978-3-319-58475-1_1
2. Geeks for Geeks. (2025, August 18). Event-Driven Architecture System Design. Geeks for Geeks. https://www.geeksforgeeks.org/system-design/event-driven-architecture-system-design/
3. Kurkovsky, S., & Williams, C. (2017). Raspberry Pi as a Platform for the Internet of Things Projects. Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. https://doi.org/10.1145/3059009.3059028
4. Makri, A., Vlachopoulos, D., & Martina, R. A. (2021). Digital Escape Rooms as Innovative Pedagogical Tools in Education: A Systematic Literature Review. Sustainability, 13(8), 4587. https://doi.org/10.3390/su13084587
5. Pfeifer, M., Völker, B., Böttcher, S., Köhler, S., & Scholl, P. M. (2021). Teaching Embedded Systems by Constructing an Escape Room. Proceedings of SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education. https://doi.org/10.1145/3408877.3432485
6. Schmidt, D. C. (2002). Middleware for real-time and embedded systems. Communications of the ACM, 45(6), 43–48. https://doi.org/10.1145/508448.508472
7. Sharma, P. (2023). Internet of Things (IoT): Study of Arduino and Raspberry Pi and their applications in various domains. International Journal of Research Publication and Reviews, 4(9), 2468–2477. https://doi.org/10.55248/gengpi.4.923.92507
8. Tanseer, I., Kanwal, N., Asghar, M. N., Iqbal, A., Tanseer, F., & Fleury, M. (2020). Real-Time, Content-Based Communication Load Reduction in the Internet of Multimedia Things. Applied Sciences, 10(3). https://doi.org/10.3390/app10031152
9. TIBCO. (n.d.). What is Event-driven Architecture? TIBCO. Retrieved August 2, 2025, from https://www.tibco.com/glossary/what-is-event-driven-architecture