🦐 ShrimpHub: The Encrypted Reef Challenge

Embedathon '26 - Hardware & Embedded Systems Hackathon

Team Members:

- Shouryadip Chakrobarty
- Abhishek Agrawal
- Joseph Verghese

Communication Backbone – MQTT

The Nervous System of the Reef

Protocol: Message Queuing Telemetry Transport (MQTT)

Why MQTT

- Low-bandwidth, pub/sub architecture allows asynchronous communication between the "Reef Broker" and our ESP32

Function

- Handled all real-time triggers, image payloads, and task-switching commands

The Tech Stack – ESP-IDF & RTOS

Why ESP-IDF over Arduino?

Precision

Granular control over task priorities and hardware interrupts

FreeRTOS

Enables deterministic behavior for time-critical tasks (Timing, Synchronization)

Optimization

Direct memory management for handling image buffers and JSON parsing (cJSON)

Subtask – The Plankton Whisper

The Challenge

Build a mood-responsive system using Serial JSON

Input

Serial string (Text + Heartbeat speed)

Action

Parsed JSON to display text on screen and dynamically update LED blinking frequency

Outcome

Established baseline communication between physical hardware and external data streams

Task 1 – The Timing Keeper

CHALLENGE

Replicate complex RGB pulse patterns from MQTT JSON arrays with zero drift

PROBLEM

Coordinating three independent color channels without blocking the CPU

SOLUTION

Concurrency: Three independent FreeRTOS tasks for R, G, and B

Synchronization: Semaphore mutexes ensured thread-safe pattern updates

RESULT

Millisecond-accurate bioluminescent pulses

Task 2 – The Priority Guardian

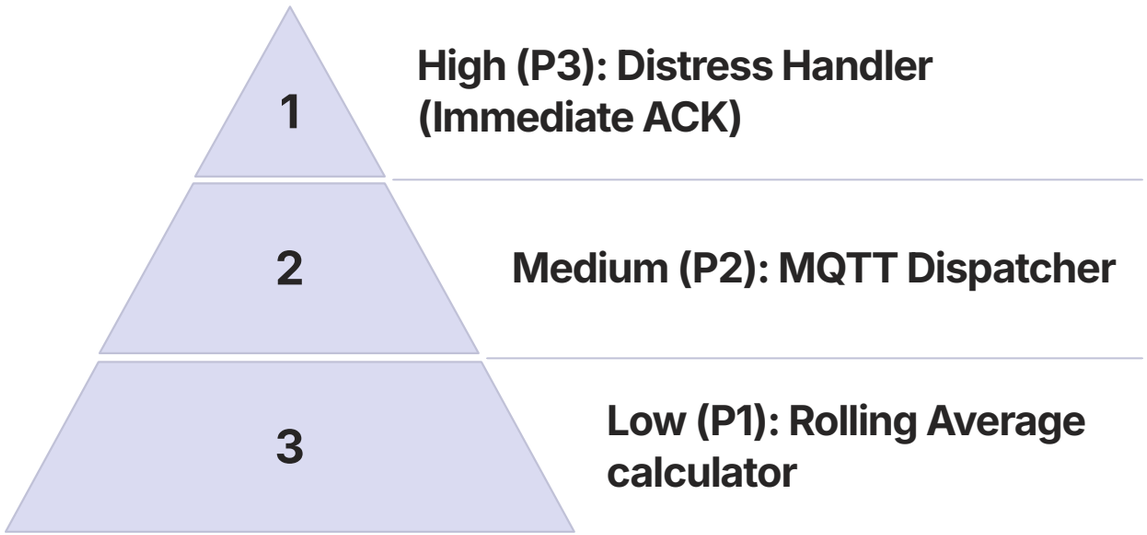
The Challenge

Manage background data streams while responding to distress calls within 250ms

The Problem

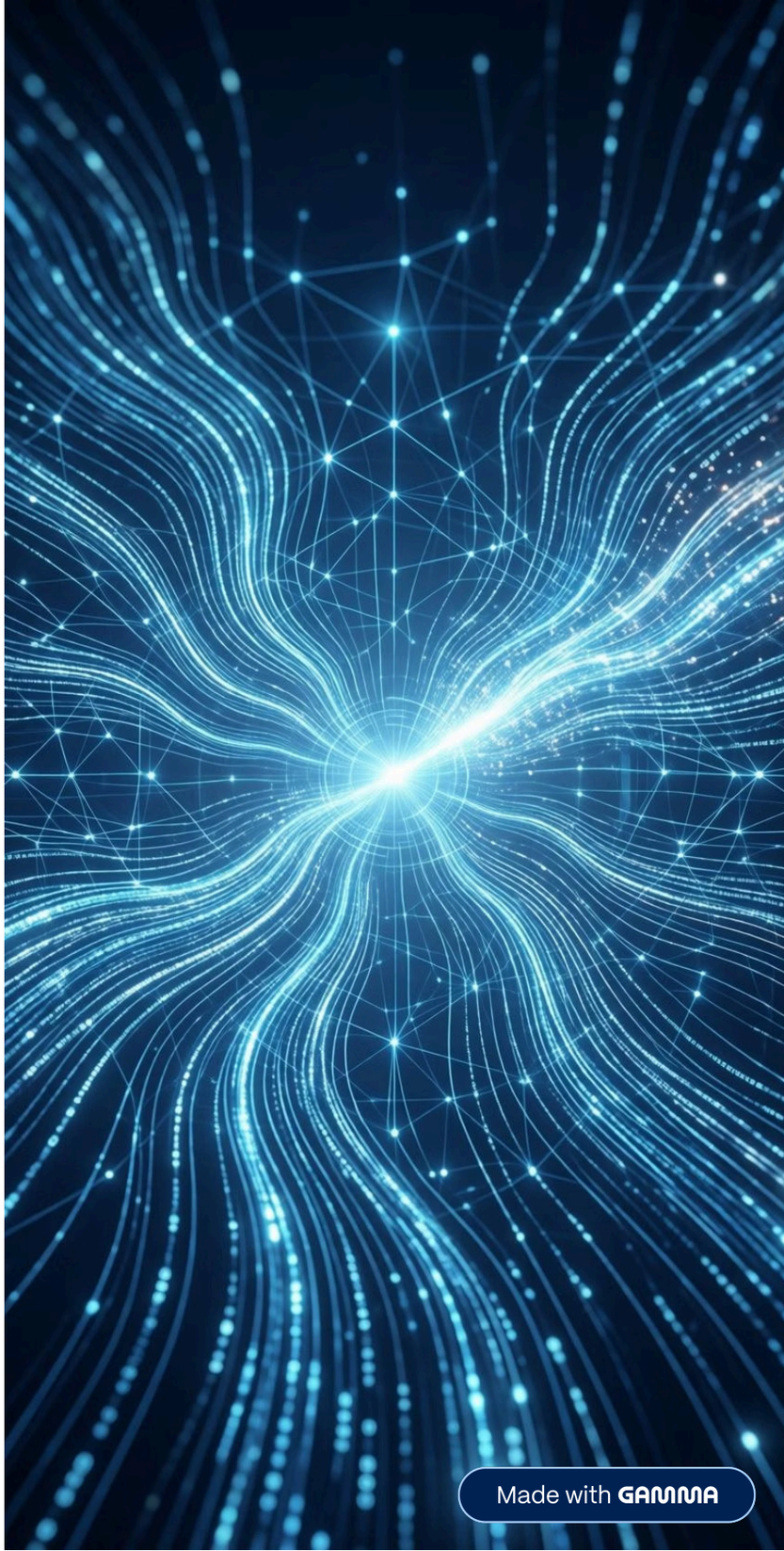
Computational "heavy lifting" (rolling averages) must not delay emergency responses

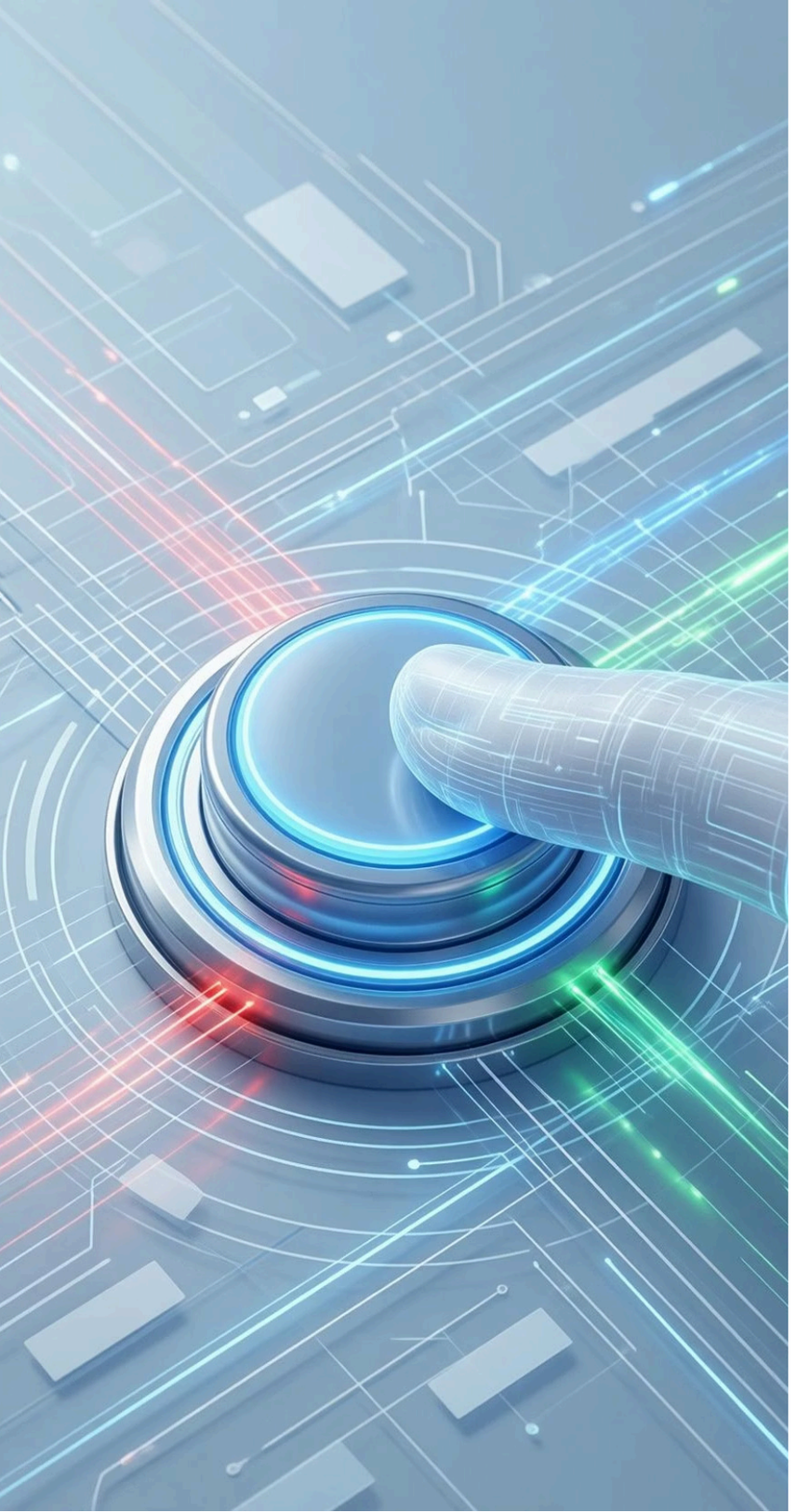
The Solution: 3-Tier Priority Architecture



Outcome

Guaranteed preemption; distress signals were acknowledged in <150ms





Task 3 – The Window Synchronizer

The Challenge

Match a physical button press to a digital MQTT window (500–1000ms)

The Problem

Human reaction time vs. high-speed digital windows (±50ms tolerance)

The Solution

1

Interrupts

Low-latency GPIO ISR for the button

2

Queues

Passed timestamps to a validation task to calculate delta

3

Feedback

Visual LED status (Red=Wait, Blue=Window, Green=Sync Success)

Task 4 – The Silent Image

The Challenge

Extract hidden coordinates from fragmented, non-human-readable payloads

The Problem

Information was hidden in the relationships between RGB channels, not the pixels themselves

The Solution

01

Reconstructed the PNG payload

02

Decoded steganographic patterns via channel-relationship analysis

Result

Extracted the "ACE!!" key and the URL for the final spatial blueprint

Task 5 – The Pixel Sculptor

The Challenge

Rearrange a source image to match a target blueprint while keeping original colors

The Problem

Global "Optimal Transport" is too heavy for embedded systems

The Solution: Approximate Optimal Transport (OT)

01

Sorted source and target pixels by perceptual luminance (0.299R + 0.587G + 0.114B)

02

Mapped source colors to target positions based on brightness rank

Outcome

Achieved SSIM scores ≥ 0.70 with full color fidelity

Conclusion

Key Takeaways

- **Embedded Intelligence:** Moving from simple loops to multi-threaded RTOS architectures
- **Real-time Response:** Balancing data processing with strict interrupt latency
- **Security & Steganography:** Recovering data from noise through relational analysis
- **Efficiency:** Approximating complex mathematical models (OT) for performance