LIFTR: El Entorno de Ejecución de Funciones IoT Tiny

LIFTR (**L**ightweight **I**oT **F**unction **T**iny **R**untime) es un sistema de **Funciones como Servicio (FaaS)** ultraligero y de código abierto. Está diseñado y optimizado para el despliegue de lógica de negocio en dispositivos **IoT de bajos recursos** y entornos de **Edge/Fog Computing**.

Desarrollado en Python, LIFTR ofrece un motor de ejecución con una **mínima huella de memoria y CPU**, eliminando la latencia y la sobrecarga que enfrentan las soluciones *Cloud-first* en el borde de la red.

---

1. Identidad Comercial y Posicionamiento

Atributo	Definición
Nombre Comercial	**LIFTR**
Sigla Completa	**L**ightweight **I**oT **F**unction **T**iny **R**untime
Énfasis de Marca	**TINY** (Ultra-Ligero)
Eslogan	**LIFTR: Python en el Borde. Latencia Cero.**

Valor Clave	El motor FaaS **más ligero y eficiente** para ejecutar código **Python** en dispositivos IoT/Edge, garantizando una ejecución **Tiny** incluso en hardware limitado.

---

2. Arquitectura y Conectividad

LIFTR se estructura en dos interfaces principales (Administración y Función) y se integra con las tecnologías más comunes en el ámbito del IoT (Python, MicroPython, C++).

A. Interfaz de Administración (Admin Dashboard)

El dashboard web (/admin/gui) permite la gestión del ciclo de vida de la función con **monitorización activa**:

- **Despliegue Sencillo:** Carga de funciones (.py) y dependencias opcionales (requirements.txt).
- **Monitorización de Recursos:** El estado del servidor (CPU, RAM, Uptime, Funciones cargadas) se actualiza de forma **automática cada 5 minutos**.
- **Consola de Operaciones:** Un registro en vivo que etiqueta claramente todas las acciones (ej. <INVOKE>, <DELETE>) y muestra los resultados de la API en formato JSON completo para facilitar la auditoría.

B. Interfaz de Ejecución de Funciones (Runtime)

Las funciones se exponen a través de endpoints HTTP dedicados.

- **Modelo de Activación:** Las funciones se activan mediante peticiones **HTTP POST** con los argumentos en formato JSON.
- **Endpoint:** Accesible vía http://[IP]:8080/function/[nombre_funcion].

C. Ecosistema y Librerías de Conectividad

LIFTR está diseñado para ser invocado por los dispositivos finales, garantizando la integración con las siguientes plataformas:

Entorno/Dispositivo	Estrategia de Conexión	Beneficio

Python Estándar	Uso de la librería requests para invocar funciones FaaS y gestionar el *payload* JSON.	Ideal para **Gateways IoT** y dispositivos Linux (Raspberry Pi).
MicroPython (ESP32/ESP8266)	Uso de módulos HTTP ligeros (ej. urequests) para construir llamadas POST con **mínimo consumo de recursos**.	Permite que los propios sensores actúen como *triggers* y reciban respuestas con eficiencia energética.
C++ (Firmware ESP32/ESP8266)	Integración vía librerías HTTP Cliente (ej. Arduino), lo que garantiza la **máxima velocidad y fiabilidad** en el punto más cercano al *hardware*.	Ideal para la lógica de respuesta crítica que requiere ejecución inmediata.

---

3. Ventajas Competitivas del Tiny Runtime

LIFTR se distingue en el mercado por su enfoque implacable en la eficiencia, clave para el éxito del IoT de bajos recursos.

- **Motor Tiny:** Ofrece la **mínima huella de memoria y CPU** para coexistir con otras tareas críticas en hardware limitado.
- **Cero Latencia de Origen:** El *runtime* minimiza los tiempos de arranque en frío (*cold start*), asegurando que el procesamiento de eventos sea **cercano al tiempo real**.
- **Logs Fieles a la API:** El diagnóstico es simple y transparente gracias a la presentación de la **respuesta cruda de la API** en la Consola y en las alertas emergentes.
- **Potencia de Python en el Borde:** Permite el desarrollo rápido de lógica avanzada (ML, pre-procesamiento) utilizando el ecosistema de Python en el entorno Fog.