

# 1) Arquitectura Recomendada (Concentrador)

## Topología

### 1 Gateway central (ESP32 cerca del router) = Concentrador

- Conectado a **WiFi cuando haya** (internet puede caer horas).
- Funciones (todo centralizado):
- **Receptor LoRa**: recibe telemetría de todas las cajas.
- **Servidor local**:
  - **Web local** (iPhone/PC desde el WiFi)
  - **API local** (Android y/o la misma web)
- **Histórico local + cola de sincronización** (*store & forward*).
- **Reglas completas de alarmas** (anti-spam, histeresis, roles, logs).
- Cuando hay internet: **sincroniza a Supabase por lotes**.

### Cajas remotas (ESP32 + LoRa) = Sensado + acción local crítica

- Cada caja puede manejar **varios reefers** (mínimo 2), con “canales” por reefer.
- Por cada reefer (canal dentro de la caja):
- **2 sensores de temperatura**.
- **Entrada DEFROST** (descongelando).
- (Opcional) **puerta**.
- (Opcional pero recomendado) **salida a relé (sirena/luz)** para alarmas críticas.
- Envían telemetría por **LoRa** al Gateway.

### Reefer cerca del router

- Puede ser:
- parte de una caja LoRa (unificado), o
- reportar por WiFi (opcional).
- Recomendación: **unificar LoRa** para simplificar.

---

## 2) Comunicación

### LoRa (estrella)

- Todas las cajas remotas → Gateway.
- Distancias hasta ~200 m: viable con LoRa.
- Mensajes pequeños por **reefer\_id** dentro de cada caja:
- `temp1`, `temp2`
- `defrost_state`
- `door_state` (si existe)
- `local_alarm_state` (si la caja dispara alarmas críticas)
- `RSSI/SNR` (opcional)

## Latencia / frecuencia

- Envío típico: **cada 30–60 s por reefer.**
  - Heartbeat por caja: **cada 60–120 s** (si no hay datos).
- 

## 3) Sensores y Hardware (hostil, frío, exterior)

### Temperatura (2 sensores por reefer)

#### Opción A (simple y probada): DS18B20 impermeable

- Rango **-55 a +125 °C.**
- Importante:
  - sensor de calidad
  - sellado real
  - cable correcto
  - pull-up bien dimensionado
- Se usa doble sensor para:
  - redundancia
  - detectar sensor fallando (diferencia > X °C, saltos raros, desconectado)

#### Opción B (industrial): PT100/PT1000 + MAX31865

- Más robusto para tiradas largas y entornos con ruido, más caro.

### Señal “DESCONGELANDO” (defrost)

- Entrada digital aislada (opto) si viene de 220V o circuito ruidoso.
- Si es contacto seco: pull-up/pull-down + protección.
- Si es 220V: **NO directo** → opto + driver.

### Puerta (opcional)

- Reed switch o micro switch.
- Métricas:
  - alerta por tiempo abierta
  - historial (aperturas, duración)

### Salida Sirena / Luz (recomendado)

- Relé/SSR para activar luz, sirena o ambas.
- Debe funcionar aunque no haya internet y aunque el gateway esté caído.

### Caja exterior

- IP65/IP67 + prensaestopas + desecante.
- Pasacables sellado hacia el interior del reefer.
- Protección eléctrica:

- fusible
  - MOV/TVS
  - fuente 220→5V confiable (industrial)
- 

## 4) Roles del sistema (qué hace cada parte)

### Gateway (Concentrador)

- Recibe y registra todo.
- Sirve panel local y API.
- Gestiona configuración y usuarios:
  - encargado / gerente / jefe cocina
- Ejecuta reglas completas:
  - umbrales por reefer
  - defrost y demora post-defrost
- puerta
- anti-spam e histéresis
- "sin señal" (nodo caído)
- Sincroniza a Supabase cuando hay internet.

### Cajas remotas (Fail-safe)

- Miden y transmiten.
  - Ejecutan solo reglas críticas mínimas para no depender del gateway:
    - temperatura crítica sostenida → sirena/luz
    - sensor inválido → sirena/luz (opcional)
    - pérdida de comunicación prolongada → aviso local (opcional)
  - Respetan DEFROST / RECOVERY\_DELAY para NO alertar cuando corresponde (aunque sea crítica).
- 

## 5) Lógica de "Descongelamiento" (por reefer / canal)

Estados por reefer (independientes, aunque compartan caja):

### MONITORING

- Evalúa temperaturas y puerta.

### DEFROST\_ACTIVE

- `defrost = ON`
- Pausa alertas de temperatura
- Sigue registrando (historial)

## RECOVERY\_DELAY

- defrost pasa a OFF
- Arranca timer configurable (ej: 30 min)
- Durante ese tiempo NO alertar temperatura

## BACK\_TO\_MONITORING

- Termina delay → vuelve a monitorear normal

### Extras

- Si defrost vuelve ON durante RECOVERY\_DELAY → vuelve a DEFROST\_ACTIVE.
  - Log:
    - inicio/fin defrost
    - duración
    - frecuencia diaria
- 

## 6) Alarmas (local + LAN + nube) con anti-spam

### Umbral configurable por reefer

- Para reefers de **-20 °C**:
  - setpoint y umbrales (alta / crítica)
- Para reefers de **0 °C**:
  - setpoint y umbrales (alta / crítica)
- Opcional:
  - alarma por “demasiado frío” si aplica

### Reglas anti falsas alarmas

- Disparo por:
  - N lecturas consecutivas, o
  - X minutos fuera de rango
  - Histeresis para evitar “entra y sale” permanente.
- Sensor inválido:
  - desconectado
  - lecturas imposibles
  - “saltos” no físicos → alerta técnica

### Entrega de alertas (3 niveles)

- 1) **Local (siempre / fail-safe):** - sirena/luz por relé en la caja remota (críticas)
  - 2) **LAN (sin internet):** - panel web rojo + notificación dentro del sistema local
  - 3) **Nube (cuando hay internet):** - push / WhatsApp / email (según definan)
-

## 7) “Modo Local” real (sin internet) — sin Raspberry

### Gateway ESP32 como servidor local

- Levanta:
    - web local
    - API local
  - Guarda histórico local:
    - flash (LittleFS) o
    - microSD (recomendado si querés mucho historial)
  - Maneja usuarios y configuración.
  - Funciona aunque internet esté caído.
- 

## 8) Sincronización a Supabase (cuando hay internet)

### Modelo store & forward centralizado

- El gateway guarda todo local con `uploaded=false`.
- Detecta conectividad.
- Cuando hay internet:
  - sube por lotes
  - marca subidos
  - si se corta, retoma

### Datos mínimos por registro

- `timestamp`
  - `node_id` (caja)
  - `reefer_id` (canal)
  - `temp1`, `temp2`
  - `defrost_state`
  - `door_state` (si está)
  - `alarm_state` (derivado)
  - `rssi/snr` (opcional)
- 

## 9) Apps: Android + Web (iPhone/PC)

### Web (LAN + nube)

Panel por reefer: - temps actuales (2 sensores) - estado: monitoring / defrost / recovery - puerta + tiempo abierta - gráfico 24h - log de eventos - último “heartbeat” (si está vivo)

Config: - umbrales por reefer - demora post-defrost - tiempo máximo puerta abierta - activar/desactivar relé de sirena/luz por evento - listado de contactos/roles

## Android

Puede ser: - misma web (PWA), o - app nativa consumiendo API local del gateway

---

## 10) Fiabilidad y seguridad

- Watchdog en cada ESP32.
- Heartbeat por caja y por reefer.
- Si una caja desaparece → alerta "sin señal".
- Firmware por SD/serial (OTA solo si red estable).

Seguridad: - login + roles - tokens simples en LAN

---

## 11) Roadmap de implementación

### Fase 1: PoC (1 caja remota manejando 2 reefers + gateway)

- 2 reefers:
- 2 temps por reefer
- entrada defrost por reefer
- LoRa al gateway
- Panel local básico
- Relé sirena/luz (crítico)
- Log local + cola para Supabase

### Fase 2: Escalar a 6 reefers

- Definir cajas multi-reefer:
- ej 3 cajas (2+2+2) o 2 cajas (3+3) según distancias y cableado
- Ajustar LoRa (SF/intervalos)
- Robustecer cajas y entradas

### Fase 3: Supabase + reportes + usuarios

- histórico nube
- reportes de defrost
- métricas puerta / fuera de rango

### Fase 4 (opcional): Door monitoring completo

- historial avanzado y ranking
-

## 12) Decisiones clave final (concentrador)

- Un solo equipo (Gateway ESP32) hace:
  - receptor LoRa
  - servidor local (web + API)
  - histórico local
  - sync Supabase
  - Cajas remotas multi-reefer:
  - miden y transmiten
  - ejecutan alarmas críticas locales (fail-safe)
  - Defrost:
  - pausa alertas + delay configurable post-defrost
  - Internet intermitente:
  - no bloquea operación local
  - supabase se actualiza cuando vuelve
- 

## 13) Mega Máquina de Estados (Arquitectura Operativa)

Esta sección baja el sistema a algo “implementable”: estados, variables, eventos y reglas.

### 13.1 Entidades

#### A) Node (Caja remota)

- Identidad: `node_id`
- Tiene N reefers: `reefer_id` (A, B, C...)

#### B) ReeferChannel (un reefer dentro de un node)

- Identidad: `node_id + reefer_id`
- Sensores: `temp1`, `temp2`
- Entradas: `defrost_in`, `door_in` (opcional)
- Salidas: `alarm_relay` (opcional)

#### C) Gateway

- Registro + UI + Sync + Config + Reglas completas
- 

### 13.2 Variables por Reefer (mínimo)

**Medición / calidad** - `t1`, `t2` - `t_valid_1`, `t_valid_2` - `t_used` (valor final usado para decisiones: promedio / min / estrategia definida) - `sensor_delta = |t1 - t2|`

**Estados** - `state` ∈ { `MONITORING`, `DEFROST_ACTIVE`, `RECOVERY_DELAY` } - `defrost` (bool) - `door_open` (bool)

**Timers** - `ts_last_sample` - `ts_defrost_start` - `ts_defrost_end` - `ts_recovery_end` - `ts_door_open_start`

**Alarmas** - `alarm_level` ∈ { `OK`, `WARN`, `CRIT`, `SENSOR_FAIL`, `NO_SIGNAL` } - `alarm_active` (bool) - `alarm_last_sent_ts`

**Config (por reefer)** - `setpoint` - `warn_high`, `crit_high` - `warn_low`, `crit_low` (opcional) - `debounce_minutes` (ej: 2-5) - `recovery_delay_minutes` (ej: 30) - `door_max_open_minutes` (opcional) - `sensor_max_delta` (ej: 2.0 °C)

---

## 13.3 Eventos

- `EV_SAMPLE(node_id, reefer_id, t1, t2, defrost, door)`
  - `EV_HEARTBEAT(node_id)`
  - `EV_DEFROST_ON`
  - `EV_DEFROST_OFF`
  - `EV_TIMER_RECOVERY_DONE`
  - `EV_DOOR_OPEN`
  - `EV_DOOR_CLOSE`
  - `EV_NO_SIGNAL_TIMEOUT`
- 

## 13.4 Transiciones de Estado (por Reefer)

### MONITORING

- Si `defrost = ON` → **DEFROST\_ACTIVE** (guardar `ts_defrost_start`)
- Si `defrost = OFF` → evaluar alarmas de temperatura y puerta

### DEFROST\_ACTIVE

- Si `defrost = OFF` → pasar a **RECOVERY\_DELAY**
- setear `ts_defrost_end`
- setear `ts_recovery_end = now + recovery_delay_minutes`
- Mientras `defrost = ON` → NO alertar temperatura (solo log)

### RECOVERY\_DELAY

- Si `defrost = ON` → volver a **DEFROST\_ACTIVE** (nuevo ciclo)
  - Si `now >= ts_recovery_end` → volver a **MONITORING**
  - Mientras esté en delay → NO alertar temperatura (solo log)
-

## 13.5 Cálculo de temperatura usada (t\_used)

Estrategia recomendada (simple y robusta): 1) Si ambos sensores válidos y  $\text{sensor\_delta} \leq \text{sensor\_max\_delta} \rightarrow \text{t\_used} = (\text{t1} + \text{t2})/2$  2) Si uno inválido  $\rightarrow \text{t\_used} = \text{el válido}$  y  $\text{alarm\_level} = \text{SENSOR\_FAIL}$  (técnica) 3) Si ambos válidos pero delta grande  $\rightarrow \text{t\_used} = \min(\text{t1}, \text{t2})$  para *conservador* +  $\text{alarm\_level} = \text{SENSOR\_FAIL}$  (requiere revisión)

---

## 13.6 Reglas de alarma (Gateway = completas)

- Temperatura alta:
  - si  $\text{t\_used} \geq \text{warn\_high}$  durante  $\text{debounce\_minutes}$   $\rightarrow$  WARN
  - si  $\text{t\_used} \geq \text{crit\_high}$  durante  $\text{debounce\_minutes}$   $\rightarrow$  CRIT
- Temperatura baja (opcional):
  - $\text{t\_used} \leq \text{warn\_low}$  /  $\text{crit\_low}$  con debounce
- Puerta (opcional):
  - si  $\text{door\_open}$  y  $\text{elapsed} \geq \text{door\_max\_open\_minutes}$   $\rightarrow$  WARN/CRIT según config
- Sin señal:
  - si no hay  $\text{EV\_SAMPLE}$  de ese reefer en  $X$  minutos  $\rightarrow$   $\text{NO\_SIGNAL}$
- Anti-spam:
  - no repetir notificación si no cambió el nivel, o si no pasó un cooldown.

**Importante:** si  $\text{state} \in \{\text{DEFROST\_ACTIVE}, \text{RECOVERY\_DELAY}\} \rightarrow$  no evaluar temperatura (sí puerta y sí sin señal).

---

## 13.7 Reglas críticas mínimas (Caja remota = fail-safe)

Objetivo: que suene la sirena aunque el gateway o la red fallen.

- En  $\text{MONITORING}$  :
    - si  $\text{t\_used} \geq \text{crit\_high}$  sostenido  $\text{debounce\_minutes}$   $\rightarrow$  activar  $\text{alarm\_relay}$
    - si sensor inválido (ambos inválidos)  $\rightarrow$  activar  $\text{alarm\_relay}$  (opcional)
  - En  $\text{DEFROST\_ACTIVE}$  y  $\text{RECOVERY\_DELAY}$  :
    - NO activar sirena por temperatura
    - Opcional: si no puede contactar al gateway durante  $Y$  minutos  $\rightarrow$  alarma local técnica.
-

## 13.8 Librerías recomendadas (ESP32 / Arduino framework)

**LoRa** - RadioLib (muy completa) o LoRa (Sandeep Mistry) (simple).

**Web / API (Gateway)** - ESPAsyncWebServer + AsyncTCP (rápido para UI/API)

**JSON / Config** - ArduinoJson

**Almacenamiento local** - LittleFS (flash) - SD (si necesitas histórico largo)

**WiFi / Reconexión** - WiFi + eventos (WiFi.onEvent)

**NTP (cuando hay internet)** - configTime / SNTP (guardar timestamps consistentes)

**Sensores** - OneWire + DallasTemperature (DS18B20) - (si PT100) librería del MAX31865

---

## 13.9 Checklist de revisión (para ver si hay que cambiar algo)

- ¿Cuántos reefers por caja realmente conviene por distancias de cable?
  - ¿DS18B20 alcanza o conviene PT100 en los más críticos?
  - ¿Alarmas críticas locales sí o sí? (recomendado)
  - ¿Histórico local: flash alcanza o SD obligatoria?
  - ¿Cómo identificas defrost? ¿contacto seco o 220V?
  - ¿Quieres que el gateway mande "ACK" a los nodos para confirmar recepción? (mejora robustez)
- 

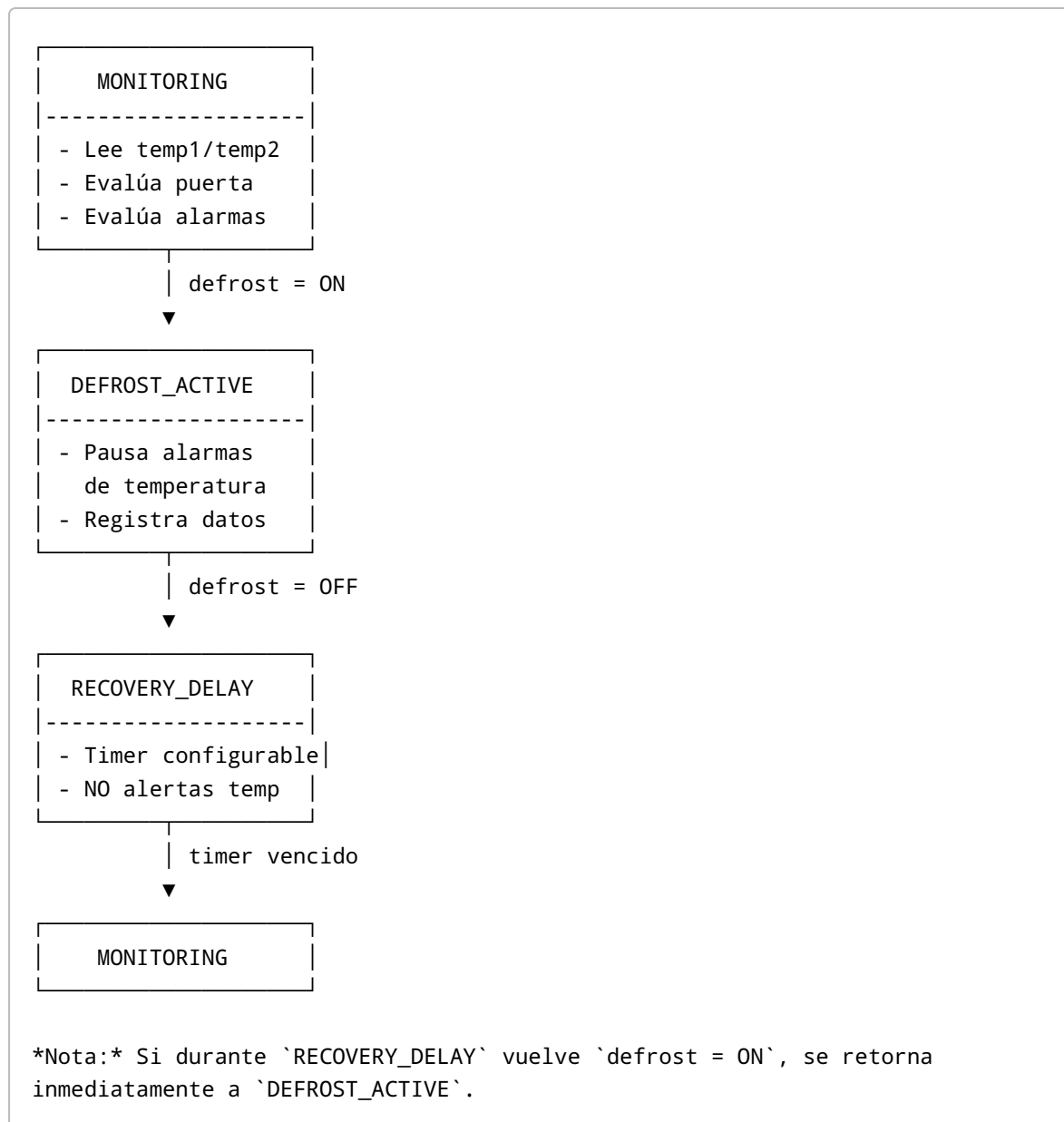
## 13.10 Próximo paso (para implementarlo)

- Definir el **formato de IDs** (`node_id`, `reefer_id`) y mapeo físico.
  - Diseñar el **payload LoRa** compacto.
  - Definir timings exactos:
    - sample interval
    - heartbeat
    - timeouts de "sin señal"
  - Definir el **JSON de configuración por reefer** editable desde la web/app.
- 

## 14) Diagramas de Máquina de Estados (Visual / Cuadraditos)

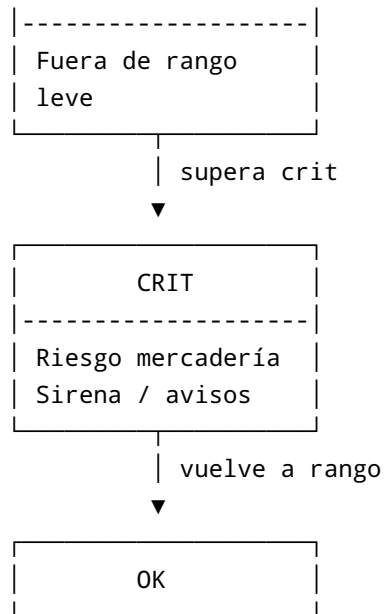
Esta sección representa la lógica en **bloques tipo diagrama**, pensados para documentación, revisión de arquitectura y validación antes de código.

## 14.1 Máquina de Estados por Reefer (Temperatura + Defrost)



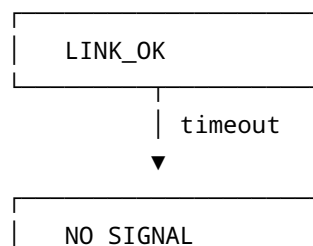
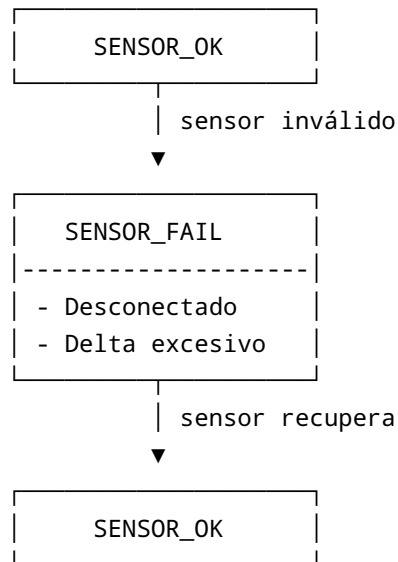
## 14.2 Máquina de Estados de Alarmas (Gateway – Completa)

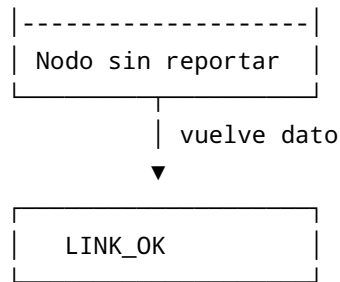




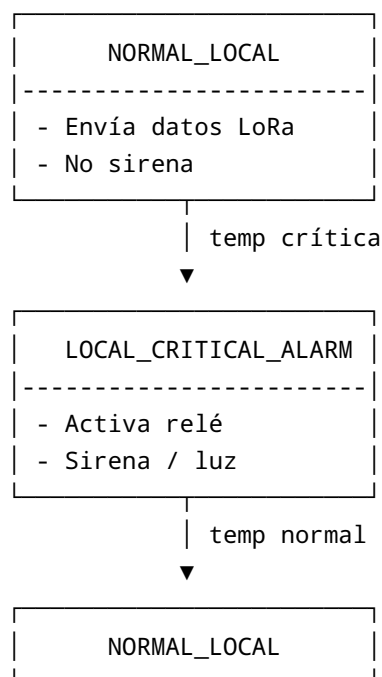
\*Reglas aplicadas:\* debounce, histeresis, anti-spam.

### 14.3 Máquina de Estados Técnica (Sensor / Comunicación)



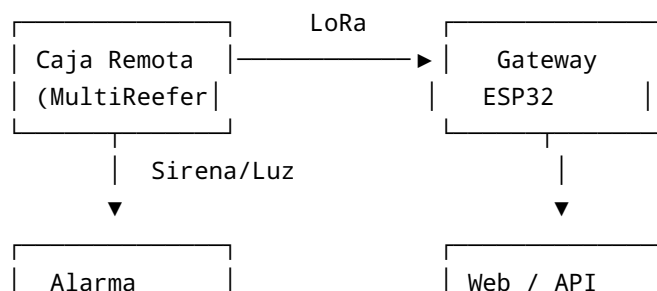


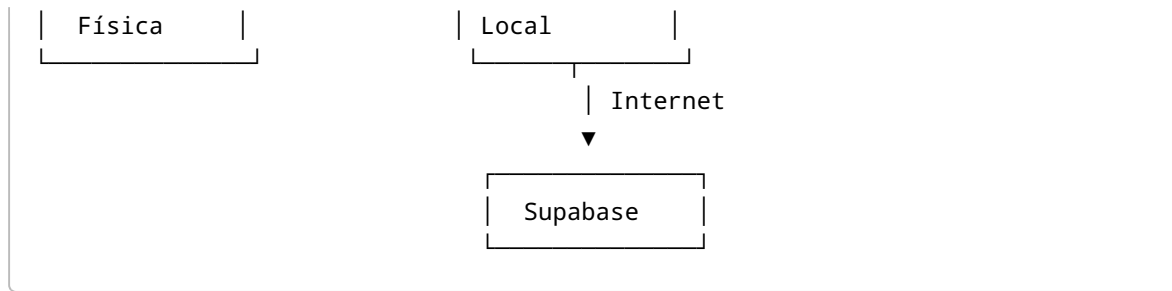
## 14.4 Máquina de Estados de Caja Remota (Fail-Safe Local)



\*Condición:\* solo válido si **\*\*NO\*\*** está en `DEFROST\_ACTIVE` ni `RECOVERY\_DELAY`.

## 14.5 Flujo General del Sistema (Vista Macro)





## 14.6 Validación de Arquitectura (Checklist Final)

- ¿Cada reefer tiene **estado independiente** aunque comparta caja?
- ¿Defrost realmente **silencia** alarmas y respeta el delay?
- ¿La caja puede **alarmar sola** si el gateway falla?
- ¿El gateway es el **único** que habla con Supabase?
- ¿Se puede operar todo **sin internet**?
- ¿El sistema degrada de forma segura ante fallos?

Si todas son **sí**, la arquitectura es sólida y lista para implementación.