Aqua Controller USB - Documento Maestro de Diseño

Versión: 4.7 (Edición Maestra con Diseño Detallado Integrado)
Fecha: 20 de junio de 2025

Preámbulo: Propósito y Legado de este Documento

Este documento representa la **fuente única de verdad** para el proyecto "Aqua Controller USB". Su propósito es servir como una "bóveda" de conocimiento que contiene la historia, las decisiones estratégicas, la arquitectura de hardware y software, los protocolos de comunicación y los procedimientos operativos.

Instrucciones de Uso:

1. **Contexto Integral:** Se debe cargar o consultar este documento completo para que cualquier colaborador (humano o IA) tenga un contexto integral del proyecto.
2. **Referencia Central:** Utilizar los ejemplos, reglas y scripts aquí documentados como la base para expandir, auditar o depurar el sistema.
3. **Guía de Decisiones:** Consultar siempre las secciones de "Anti-Patrones" y "Decisiones Estratégicas" antes de proponer o implementar cambios en el núcleo del sistema.
4. **Documento Vivo:** Este documento debe ser actualizado ante cualquier migración, descubrimiento de un bug importante, expansión de funcionalidades o cambio de paradigma en la arquitectura.
5. **Manifiesto Fundacional:** Los principios de **seguridad, modularidad y operación "Local-First"** son inmutables y no deben ser comprometidos.

Capítulo 1: Fundamentos Estratégicos del Proyecto

1.1. Génesis y Motivación

El sistema surge como respuesta a la frustración con controladores comerciales (Apex, GHL) y soluciones DIY de primera generación, debido a deficiencias clave que comprometen la seguridad de un ecosistema marino:

- **Falta de Tolerancia a Fallos Reales:** Muchos sistemas dependen de un punto único de fallo.
- **Dependencia de la Nube y Apps Externas:** La pérdida de conexión a internet no debe suponer la pérdida de control vital.
- **Limitación para Integrar Sensores y Reglas Personalizadas:** Las arquitecturas cerradas impiden la personalización.
- **Dificultad para Escalar o Migrar:** Los sistemas propietarios dificultan la

expansión.

1.2. Iteraciones Iniciales y Lecciones Aprendidas

- **Prototipo v1 (Arduino + WiFi):** Descartado por latencias inaceptables y poca robustez.
- **Prototipo v2 (ESP32 + MQTT sobre WiFi):** Descartado; la red doméstica nunca es tan fiable como un bus cableado para funciones de soporte vital.

1.3. Decisión de Arquitectura Definitiva: El Enfoque Híbrido USB-First

Se optó por una arquitectura que prioriza la fiabilidad sobre la conveniencia inalámbrica para las funciones críticas, utilizando el **bus USB CDC** como canal de comunicación principal.

1.4. Anti-Patrones Documentados y Prohibidos

- Nunca lógica vital en la nube.
- Nunca WiFi para funciones críticas de control.
- Nunca dependencias de hardware/software cerrado.
- Nunca cambios de rol por recompilación/hardcode (siempre por configuración en NVS).

1.5. Visión a Largo Plazo y Evolución Esperada

La plataforma está concebida para evolucionar hacia un sistema de mantenimiento predictivo e inteligente, capaz de autoajustarse mediante IA, detectar patrones anómalos antes de que sean críticos, y reducir la intervención humana a tareas realmente necesarias. La arquitectura modular y la documentación viva garantizan que el sistema pueda expandirse, migrar o adaptarse a tecnologías futuras sin comprometer sus principios fundacionales.

Capítulo 2: Arquitectura General del Sistema

2.1. Topología General y Componentes

El sistema Aqua Controller USB v3.0 se compone de un MASTER (Raspberry Pi 5) conectado a uno o más módulos esclavos (“paneles”) mediante un bus USB CDC. La filosofía es que el cable USB (y un hub alimentado) sean la arteria central de control.

- **MASTER (RPi 5):** Cerebro lógico, motor de reglas, HMI local.
- **PANELES (ESP32-S3):** Nodos especializados (AC, DC, IO).
- **ATLAS_PI (RPi Zero 2W):** Hub de sensores científicos.

Se garantiza que la HMI local nunca depende de la nube ni de conectividad externa

para el control de funciones críticas. Todos los paneles y el MASTER priorizan el funcionamiento en modo local-first, asegurando la autonomía y resiliencia del ecosistema.

Expansión Futura: La arquitectura contempla la posible adición de módulos AUX (paneles adicionales, simuladores, expansores de entradas/salidas), asegurando compatibilidad plug & play y autoidentificación por USB.

2.2. Diagrama Físico/Lógico

(Ver Anexo 9.1 para diagrama detallado).

2.3. Estructura de Alimentación

- Fuente principal Meanwell 12V 10A.
- Conversores Buck 12V⟶5V.
- Fusibles y TVS en cada panel.
- Aislamiento galvánico para señales críticas.
- **Nota sobre Redundancia:** El diseño prevé la redundancia física y lógica: cada línea de alimentación puede ser reforzada o duplicada, y los módulos están diseñados para minimizar el impacto de un fallo aislado.

2.4. Escenarios de Falla y Resiliencia de Módulos

Módulo	Escenario de Falla	Modo de Recuperación/Resiliencia
MASTER	Fallo de comunicación USB	Paneles entran en SAFE MODE autónomo.
PANEL_AC	Fallo de sensor de temp	Detiene calentadores, enciende alarma local, notifica al MASTER.
PANEL_DC	Sobreconsumo o cortocircuito	Apaga todas las salidas DC, reporta error crítico.
PANEL_IO	Falla en detección de nivel/fuga	Notifica, activa alarma local y bloquea acciones relacionadas (ej. ATO).
ATLAS_PI	Fallo de lectura de sensor	Marca valores como inválidos, alerta al MASTER.

Capítulo 3: Especificaciones de Hardware y Módulos

3.1. Tabla de Módulos y Funciones

MÓDULO	MCU/SOC	I/O PRINCIPAL	FUNCIONES CLAVE
MASTER	RPi 5 (4GB/8GB)	USB3.0, LAN, HDMI	Motor de reglas, HMI, logging, polling, OTA.
PANEL_AC	ESP32-S3-WROOM	8 relés, 2 DS18B20	Control AC, temperatura, SAFE MODE térmico.
PANEL_DC	ESP32-S3-WROOM	8 MOSFET, 4 PWM	Control DC, bombas, válvulas, SAFE MODE.
PANEL_IO	ESP32-S3-WROOM	16 entradas optoacopladas	Sensores nivel/fuga, pulsadores, alarmas.
ATLAS_PI	RPi Zero 2W	I2C, USB	Sensores Atlas con aislamiento galvánico.

3.2. Criterios de Selección y Detalles de Componentes

- **Topología Eléctrica Simplificada:** Cada módulo recibe energía a través de líneas separadas protegidas por fusible y TVS. El GND de señal está aislado del GND de potencia donde aplica (especialmente para sensores Atlas y entradas optoacopladas).
- **Relés:** Seleccionados por capacidad (10A+), reputación industrial (ej. Omron, Songle) y soporte de ciclos de vida >50k. Se usan SSR para cargas resistivas y EMR con snubber para inductivas.
- **MOSFET:** Elegidos por baja Rds(on) y capacidad de disipación térmica (ej. IRLZ44N), permitiendo operación continua sin sobrecalentamiento.
- **Fuentes:** Fuentes Meanwell seleccionadas por su robustez y MTBF (Mean Time Between Failures) comprobado.
- **Sensores:** DS18B20 y Atlas Scientific por su fiabilidad comprobada en ambientes marinos.
- **Notas sobre Expansión:** El PANEL_IO está diseñado con headers I2C para expansión de E/S con MCP23017. El PANEL_DC puede ampliarse para controlar drivers LED multicanal.

Capítulo 4: Arquitectura de Software y Firmware

4.1. Estrategia de Firmware Universal (ESP-IDF)

Se utiliza un único binario de firmware para todos los paneles ESP32, basado en ESP-IDF nativo. El rol específico (AC, DC, IO) se asigna en el primer arranque vía consola serie y se almacena en la memoria NVS.

4.2. Flujo de Inicialización y Ciclo Principal del Firmware

- **Flujo de Inicialización de Rol:** Al primer arranque, si el NVS está vacío, el panel entra en modo de configuración y espera un comando de asignación de rol desde el MASTER. Una vez asignado, lo almacena en NVS y reinicia. Si el NVS se corrompe, el panel vuelve a este modo para reasignación segura.
- **Estructura Típica de Ciclo Principal (main loop):**
  1. Inicialización de hardware y lectura de rol desde NVS.
  2. Conexión al bus USB CDC y sincronización con el MASTER.
  3. Bucle infinito para recepción y parseo de comandos JSON.
  4. Ejecución de lógica específica por rol (AC/DC/IO), incluyendo la lógica SAFE MODE si corresponde.
  5. Reporte periódico de status y manejo de eventos críticos asíncronos.
  6. Supervisión constante del watchdog de software y verificación de la salud de los periféricos.

4.3. Estrategia de Actualización OTA

- El MASTER puede actualizar cualquier panel mediante el bus USB o el canal secundario WiFi, utilizando la partición OTA segura.
- El proceso incluye verificación de integridad del firmware (checksum) y rollback automático a la partición anterior en caso de error en el arranque.
- Antes de actualizar, el sistema puede realizar un backup de la configuración crítica y los logs del panel.

Capítulo 5: Protocolo de Comunicación y Contrato de API

5.1. Modelo de Comunicación (Polling Híbrido)

Se utiliza un modelo que combina un polling de alta frecuencia (<100ms) para supervisión con el envío asíncrono de eventos críticos.

5.2. Catálogo de Comandos y Respuestas JSON

- **Comandos Principales (MASTER → Panel):**
  - **get_status:** Solicita el estado completo del panel. Respuesta:

status_response.
    - **set_output:** Activa/desactiva un canal específico. Respuesta: ack.
    - **update_config:** Cambia la configuración del panel.
    - **get_telemetry:** Solicita telemetría extendida (logs, uso).
- **Eventos Críticos (Panel → MASTER):**
    - **event_critical:** Reporta condición de riesgo o falla (ej. fuga, sobrecalentamiento).
- **Error Codes y Versionado:** Todos los mensajes pueden incluir "error_code": <num>, "error_msg": <texto>, y "api_version": "1.0".
- **Convenciones:** Nombres de campos en snake_case. Cada panel responde con su id, type, y fw_version.

Capítulo 6: Lógica de Automatización (Motor de Reglas)

6.1. Arquitectura del Motor (Doble Nivel)

Se implementa un motor de dos niveles en el MASTER.

- **Nivel 1 (Motor Declarativo):** El núcleo procesa reglas definidas en un formato estructurado JSON/YAML.
- **Nivel 2 (DSL Natural):** Una capa de abstracción permite a usuarios avanzados escribir reglas en una sintaxis legible similar a Python.

6.2. Catálogo de Triggers, Conditions y Actions

- **Triggers:** sensor_threshold (ej. temp_water < 24.5), on_change (cuando un valor cambia), periodic, at_time.
- **Conditions:** Comparaciones (==, !=, <, >) y operadores lógicos (AND, OR, NOT).
- **Actions:** set_output, send_notification, log, set_timer (para acciones diferidas).

6.3. Ciclo de Vida y Gestión de Reglas

Las reglas pasan por un ciclo de edición (HMI o editor), validación de sintaxis, carga en el motor (con hot reload), ejecución y logging. Si varias reglas entran en conflicto, la prioridad se determina por orden o severidad.

Capítulo 7: Interfaz y Experiencia de Usuario (HMI)

7.1. Wireframes y Pantallas Clave

- **Dashboard General:** Vista de estado en tiempo real de todos los módulos (OK/advertencia/crítico), gráficos históricos de parámetros clave, y lista de alarmas activas.
- **Detalle de Módulo:** Pantalla para cada panel con estado y control manual de sus

E/S, e historial de eventos.
- **Edición de Reglas:** Visualizador/editor de reglas con validación de sintaxis y opción de prueba en caliente.
- **Configuración y Backup:** Interfaz para exportar/importar reglas y configuración, y para iniciar backups manuales.

7.2. Estrategia de Alertas y Notificaciones

- **Niveles:** INFO, WARNING, CRITICAL.
- **Canales:** HMI local (siempre), y notificación push/email/WhatsApp configurable por nivel.
- **Lógica de Supresión:** Alertas repetidas en un corto período se agrupan para evitar la "fatiga de alarmas".

Capítulo 8: Persistencia de Datos y Base de Datos

8.1. Estrategia de Persistencia

Los datos históricos y logs se almacenan en una base de datos SQLite3 en el MASTER, con rotación automática de logs.

8.2. Esquema de Base de Datos (SQL Simplificado)

```
CREATE TABLE telemetry_log (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
  module_id TEXT NOT NULL,
  sensor_name TEXT NOT NULL,
  value REAL,
  -- Índice para búsquedas rápidas por tiempo y sensor
  INDEX idx_telemetry_time_sensor (timestamp, sensor_name)
);

CREATE TABLE event_log (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
  module_id TEXT,
  event_type TEXT,
  description TEXT,
  severity TEXT NOT NULL,
  -- Índice para filtrar por severidad y tiempo
```

```
  INDEX idx_event_severity_time (severity, timestamp)
);

CREATE TABLE module_config (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  module_id TEXT UNIQUE NOT NULL,
  config_json TEXT,
  last_update DATETIME
);
```

Capítulo 9: Anexos Técnicos

9.1. Tabla de Pinout (Ejemplo para PANEL_AC)

(Se requiere un documento detallado por cada panel)
Panel	Función	Pin GPIO	Comentarios
PANEL_AC	Relé Calentador	GPIO5	Canal 1
PANEL_AC	Relé Luces	GPIO6	Canal 2
PANEL_AC	Sensor Temp	GPIO21	Bus 1-Wire DS18B20
PANEL_AC	Botón Emergencia	GPIO16	Input, con pull-up interno

9.2. Matriz de Alarmas

Sensor	Evento	Acción Inmediata del Sistema
Nivel Sump (Bajo)	Agua insuficiente para retorno	Apagar bomba de retorno, activar alarma WARNING.
Fuga	Humedad detectada	CERRAR TODAS LAS VÁLVULAS, APAGAR BOMBAS PRINCIPALES, ALARMA CRITICAL.
Temperatura (Alta)	Sobrecalentamiento	Apagar calentadores y luces, activar enfriador (si existe), alarma WARNING.

9.3. Procedimientos de Validación

- Chequeo de voltajes y fusibles antes de energizar paneles.
- Prueba de comunicación USB con WHO_ARE_YOU.
- Test de SAFE MODE: desconexión del MASTER y verificación de operación autónoma.
- Validación de reglas con eventos simulados.