

# RSmartbox - Informe tecnico

Fecha: 2025-12-21 23:05

## Resumen ejecutivo

Sistema protector inteligente para bombas con medicion de corriente y voltaje, control de relay/contactor, interfaz local LCD+boton, telemetria por SIM7000G/MQTT, panel web con tiempo real, historico en SQLite y programacion de horarios.

## Arquitectura de software

### Estructura de carpetas principal

Ruta	Descripcion
platformio.ini	Configuracion PlatformIO, librerias y puerto de carga
src/main.cpp	Firmware ESP32 (logica de proteccion, MQTT, LCD, boton)
collector_api.py	Servicio PC: MQTT -> SQLite + API REST /lecturas
serve.py	Servidor Flask que sirve el panel web y arranca MQTT
panel_streamlit/mqtt_dashboard.html	Panel web HTML (tiempo real, historico, mapa, programacion)
panel_streamlit/assets/	Imagenes de fondo editadas del panel
panel_streamlit/app.py	Panel Streamlit legado (opcional)
start_services.bat	Arranque automatico de serve.py y cloudflared

## Flujos y servicios

- 1) Firmware publica estado y GPS por MQTT (SIM7000G).
- 2) collector\_api.py consume MQTT, guarda en SQLite y expone REST.
- 3) serve.py entrega el panel HTML y reutiliza la API.
- 4) El panel HTML consume MQTT (tiempo real) y REST (historico).

## Firmware (ESP32)

### Pines y perifericos

Senal	Pin	Uso
LED azul	GPIO13	Indicador de vida
PWRKEY	GPIO4	Encendido del modem
UART modem TX/RX	GPIO27/26	SIM7000G
SCT013 (corriente)	GPIO36	ADC1 con bias
ZMPT101B (voltaje)	GPIO39	ADC1 con divisor
LCD I2C	GPIO21/22	SDA/SCL
Relay SSR	GPIO25	Salida control bomba
Pulsador	GPIO32	Entrada activa HIGH

## Logica de proteccion

- Estado RUNNING: relay ON, muestra Bombeando, I y V.
- Sobrecorriente:  $I > \text{setpointA} * (1 + \text{rangoSobre})$  -> detenido\_sobrecorriente (latched).
- Baja corriente:  $I < \text{setpointA} * (1 - \text{rangoBaja})$  -> recuperando (timer).

- Voltaje fuera de rango: V fuera de setpointV +/- rangoVolt -> detenido\_subvoltaje o detenido\_sobrevoltaje, espera 2 min y reintenta.
- Gracia inicial: 10 s al entrar en RUNNING antes de evaluar protecciones.

## MQTT y payloads

Broker: test.mosquitto.org:1883

Topics: sim7000g/estado, sim7000g/gps, sim7000g/cmd

Estado JSON incluye: corriente, voltaje, estado, t\_trabajo\_min, t\_recuperando\_min, t\_rec\_total\_min, corriente\_falla, voltaje\_falla, relay\_on.

## Panel web

Panel HTML con imagen de fondo y overlays, control remoto de relay, graficas en tiempo real, historico en tabla + graficas, mapa GNSS y programacion de horarios. La programacion actualmente envia payload schedule por MQTT y requiere implementacion en firmware.

## Backend (PC)

collector\_api.py guarda lecturas en SQLite (lecturas.db) y expone /lecturas?start=&end;=. serve.py sirve el panel y arranca el hilo MQTT. start\_services.bat automatiza el arranque junto con cloudflared.

# Electronica (segun esquema y confirmaciones)

## Alimentacion y control

- LilyGO T-SIM7000G alimentado por VIN 5 V.
- SSR/relay controlado por GPIO25 (activo HIGH) y GND comun.
- Contactor con bobina 230 VAC. Snubber en bobina: valor recomendado por fabricante del contactor (no numerico).

## Medicion de corriente (SCT013 100A/1V)

- SCT013 conectado a GPIO36 (ADC1).
- Bias a mitad de Vcc con R1=10k (3.3V->Vbias) y R2=10k (Vbias->GND).
- C1=0.1 uF entre Vbias y GND.
- GND comun con ESP32.

## Medicion de voltaje (ZMPT101B)

- ZMPT101B alimentado a 3.3 V.
- OUT a divisor: R4=5.6k (OUT->nodo ADC), R3=4.7k (nodo ADC->GND).
- Nodo ADC a GPIO39.
- Capacitor 100 uF en alimentacion del modulo.

## LCD I2C

- LCD 16x2 con I2C, direccion 0x27.
- VCC = 5 V, GND comun.
- SDA GPIO21, SCL GPIO22.
- Capacitor 100 uF en alimentacion del LCD.

## Pulsador

- Pulsador entre 3.3 V y GPIO32 (activo HIGH).
- Firmware usa INPUT\_PULLDOWN y debounce por software.

## Datos pendientes

- Valor exacto del snubber (RC/MOV) en la bobina del contactor: segun fabricante.
- Modelo exacto del SSR y potencia nominal: desconocido.
- Fuente de alimentacion del contactor y protecciones adicionales: desconocido.