SmartWater

Camean Pablo Ezequiel, López Fernando Antonio, Palmieri Marco Tiziano,
Quiroga Piegari Lucila, Romero Lucas Nicolas
34520666, 42537132, 45542385, 41566741, 43780360
Martes, Grupo M2

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

Resumen.

Sistema de Detección de Consumo de Agua basado en ESP32 con FreeRTOS.

Este trabajo presenta el diseño y desarrollo de SmartWater, un sistema embebido destinado a automatizar la detección y control del consumo de agua en hogares ubicados en áreas de baja disponibilidad hídrica, mediante el uso de un microcontrolador ESP32 que, por medio de una máquina de estados, gestionará de manera eficiente el flujo de agua.
Se implementa una arquitectura basada en eventos y tareas concurrentes gestionadas con FreeRTOs. El diseño del sistema contempla diversas características, tales como un modo de depuración, la posibilidad de configurar los parámetros según las necesidades del usuario y una clara separación de responsabilidades entre los distintos componentes. Este enfoque no solo busca optimizar el uso del agua, sino también proporcionar una herramienta flexible y eficiente para la gestión responsable de este recurso en contextos de escasez.

Palabras claves: ESP32, FreeRTOs, electroválvula, caudalímetro, buzzer.

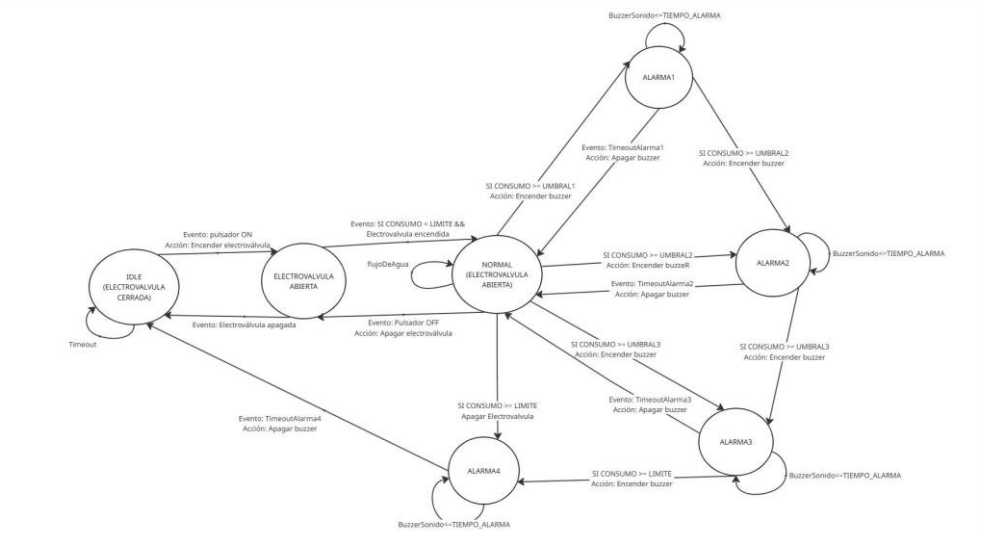
1 Introducción

En un planeta donde el agua dulce se ha convertido en un recurso cada vez más escaso, contar con tecnologías capaces de controlar su consumo es fundamental para garantizar un uso eficiente.

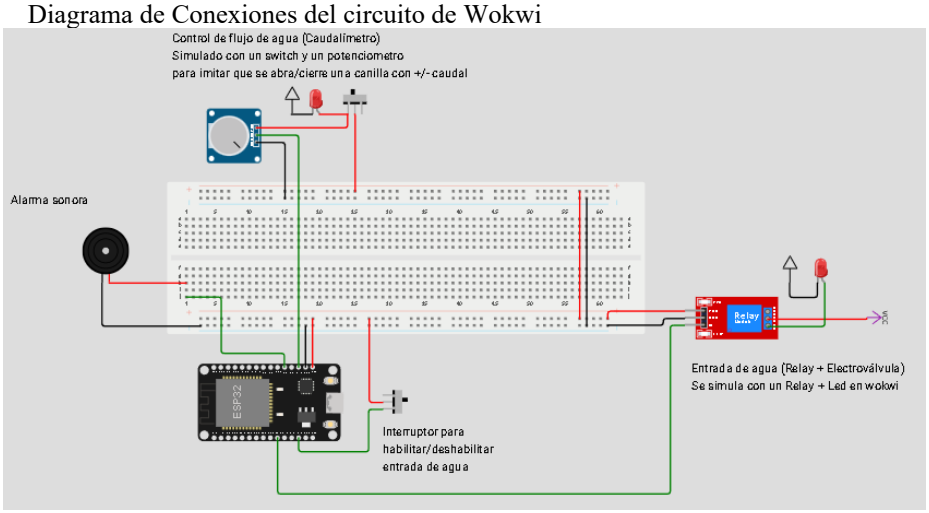
El proyecto SmartWater es una aplicación destinada al control responsable del agua en los hogares. Basado en un microcontrolador ESP32 y ejecutando FreeRTOS, este sistema embebido opera como una máquina de estados que supervisa en tiempo real el flujo de agua. A través de un caudalímetro, detecta el consumo, y al superar ciertos umbrales de consumo previamente definidos, se activa una alarma acústica (buzzer) que notifica al usuario sobre el nivel alcanzado. Este proceso se repite en cuatro etapas, cada una con un umbral de consumo distinto. A medida que se superan los umbrales, la intensidad de la alarma aumenta progresivamente, alertando de forma enfática al usuario. Una vez alcanzado el cuarto umbral de consumo, el sistema activa una electroválvula que corta el paso de agua a través del caudalímetro, limitando así el consumo diario a una cantidad predeterminada. Cuando el gasto de agua sobrepasa niveles críticos, el sistema emite alertas sonoras progresivas y, finalmente, acciona una electroválvula que interrumpe el suministro, asegurando un uso responsable y perfectamente controlado.
A su vez, el proyecto cuenta con una aplicación móvil donde se podrá visualizar el consumo, así como recibir notificaciones tras alcanzar los umbrales definidos.

2 Desarrollo

- Prototipo en Wokwi: Wokwi
- Repositorio de github: Repositorio
- Diagrama de estados:



- Diagrama de Conexiones del circuito de Wokwi



Manual de usuario del embebido simulado

Este sistema implementa un control automático de consumo de agua basado en una máquina de estados. Está pensado para detectar el caudal instantáneo, generar alertas sonoras en cuatro niveles (ALARMA1, ALARMA2, ALARMA3, ALARMA4) y, al llegar al límite diario definido, cerrar la electroválvula para cortar el suministro. El control se ejecuta sobre un ESP32 DevKit1 con multitarea gestionada por FreeRTOS, lo que permite separar lecturas de sensor, lógica de estado, generación de sonido y actuación de la válvula en tareas independientes.

Componentes del sistema:

Componente	**Descripción**
ESP32	Microcontrolador que gestiona el sistema
Caudalímetro	Sensor que mide el caudal de agua que transita por el mismo.
Relé	Módulo para el control de la válvula
Electroválvula	Actuador que permite/restringe el transito de agua

Buzzer	Actuador que notificará mediante distintos niveles de sonido al usuario sobre su consumo de agua.
Pulsador	Interruptor de emergencia que permite cerrar/abrir el transito de agua manualmente.

Funcionamiento de los sensores/actuadores

- **Caudalímetro:** El modelo que usaremos usa el efecto Hall para medir pulsos eléctricos que son generados por una pequeña turbina mecánica, la cantidad de pulsos es directamente proporcional a la velocidad de la turbina que a la vez es directamente proporcional al flujo de agua.
- **Relé:** Este actuador permite a través de un electroimán y una bobina mover una serie de contactos que funcionan como interruptor para abrir/cerrar una conexión.
- **Electroválvula:** Usa el mismo principio que el Relé, al energizar una bobina esta atrae un émbolo que permite el paso del fluido por la válvula.
- **Buzzer:** Utiliza pulsos eléctricos para mover un piezoeléctrico que hace mover una membrana de metal que genera el sonido.

Funcionamiento general

El sistema se comporta como una máquina de estados con estos objetivos principales: leer el caudal (pulsos por litro) y acumular consumo; comparar consumo acumulado con una serie de umbrales: UMBRAL1, UMBRAL2, UMBRAL3, LIMITE; al superar cada umbral se activa una alarma sonora con patrón/ intensidad creciente; si se alcanza LIMITE, se dispara la acción de corte: activar el relé para cerrar la electroválvula y detener el flujo.

El sistema puede operar de dos maneras

- Modo automático: la electrovalvula se cierra automáticamente cuando se supera el limite de consumo diario.
- Modo manual: mediante el potenciometro puede cerrarse/abrirse la electrovalvula manualmente a demanda.

Estados del sistema (resumen y comportamiento)

IDLE (Electroválvula cerrada): Estado de reposo; espera que el sistema sea habilitado (pulsador ON) o que se solicite apertura manual.

ELECTROVALVULA ABIERTA: Evento: pulsador ON / orden de apertura. Acción: energizar relé para abrir válvula; transicionar a NORMAL.

NORMAL (Electroválvula abierta): Estado operativo con válvula abierta; se acumula consumo. Transiciones principales: si CONSUMO >= UMBRAL1 →

ALARMA1; si >= UMBRAL2 → ALARMA2; si >= UMBRAL3 → ALARMA3; si >= LIMITE → ALARMA4 (o acción de límite). También puede volver a IDLE si se presiona manualmente el pulsador.

ALARMA1 / ALARMA2 / ALARMA3 / ALARMA4: Cada una representa un nivel de aviso progresivo. Acciones: encender buzzer con patrón específico. Eventos de timeout (TimeoutAlarmaX): apagan el buzzer temporalmente; si el consumo sigue por encima del umbral correspondiente, la alarma se reenciende o permanece según la lógica. ALARMA4 se asocia al LIMITE y, además de buzzer, debe provocar la acción de corte: ordenar el cierre de la electroválvula y transicionar a IDLE o a un estado de seguridad.

Condiciones de alarma y corte (configurables)

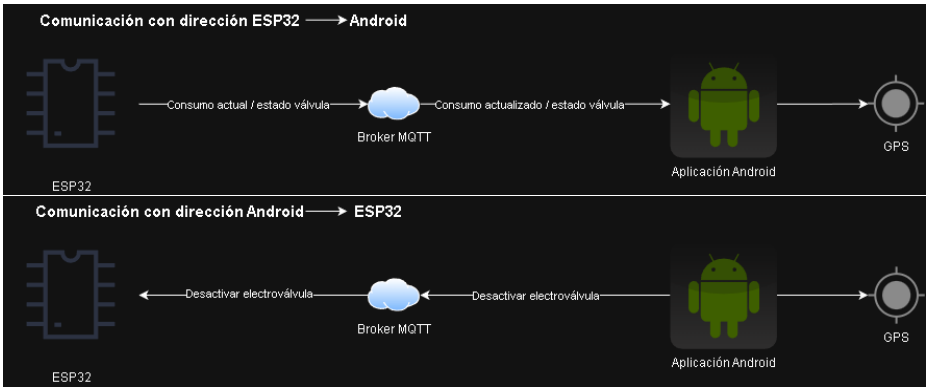
UMBRAL1 — primer aviso audible (p. ej. 25% del límite).
UMBRAL2 — segundo aviso con tono más insistente (p. ej. 50%).
UMBRAL3 — tercer aviso; tono/patrón más largo (p. ej. 75%).
LIMITE — alcanza el tope diario; accionamiento de corte y ALARMA4 (p. ej. 100%).

Los valores de UMBRALx y LIMITE deben definirse en constantes configurables en código (litros o pulsos).

Modos de operación
Modo Debug: imprime en Serial todas las transiciones, lecturas de caudal y eventos importantes.
Modo Productivo: desactiva impresiones para ahorrar CPU y ancho de banda serial.

Control por constante:
#define MODO_DEBUG true // true = logs activados; false = silenciar

Arquitectura del sistema



Comunicación con Hivemq y MQTT

La arquitectura del sistema implementada incluye comunicación inalámbrica basada en el protocolo MQTT, facilitando de esta manera el control remoto de la electroválvula desde la App Android, así como también la visualización del consumo desde la misma.
A su vez, utilizaremos GPS para detectar la posición actual del usuario con la intención de notificar al usuario si se fue de su casa y dejó alguna canilla abierta.

Los tópicos empleados se describen a continuación:

- **aguasmart/consume:** el ESP32 publica en este tópico el consumo actual detectado por el caudalímetro, la aplicación Android leerá el consumo, que será un valor float, para mostrarlo en la pantalla.
- **aguasmart/valve/state:** a través de este tópico el ESP32 comunicará a la aplicación de Android acerca del estado de la válvula: “active” si está encendida, “inactive” si está apagada.
- **aguasmart/valve/cmd:** a través de este tópico se enviará al ESP32, con el mensaje “button_push”, la orden de activar/desactivar la válvula desde la aplicación de Android cuando el usuario presione el botón como se describirá más abajo en el presente informe.

Android

Como complemento del sistema embebido SmartWater, se desarrolló una aplicación Android destinada a brindar al usuario una experiencia completa de monitoreo y control remoto del consumo de agua. La App permite visualizar en tiempo real el caudal registrado por el caudalímetro, recibir alertas correspondientes a los distintos umbrales configurados y realizar la activación o desactivación de la electroválvula. Además, facilita la configuración de parámetros diarios y la administración de límites de consumo.
La comunicación entre la aplicación y el ESP32 se realiza mediante el protocolo MQTT utilizando un bróker hivemq o emqx. La app emplea la librería Paho MQTT para suscribirse a los tópicos del sistema y publicar comandos de control. Esto permite que la información viaje en tiempo real entre el dispositivo embebido, el broker y la aplicación móvil, garantizando una experiencia fluida y fiable para el usuario.
Además, la aplicación cuenta con un sensor GPS que detecta si el usuario se aleja del sistema embebido y desactiva la electroválvula para asegurar el ahorro de agua.

Navegación de las Activities

Main Activity

Es la pantalla principal de la aplicación. Brinda acceso rápido a las funcionalidades “Ver consumo de agua”, “Fijar ubicación de GPS” y “Desactivar válvula”.

Características:

- El botón “Desactivar válvula” se comunica con el ESP32 para activar/desactivar la válvula remotamente en caso de una emergencia o que simplemente el usuario desee activarla/desactivarla remotamente.
  El usuario recibirá una notificación que le informará si la válvula fue activada/desactivada exitosamente.
- El botón “Fijar ubicación ESP32” nos permite fijar las coordenadas del sistema embebido para que la aplicación detecte si el usuario, junto con su dispositivo móvil, sale del rango de alcance y desactive la válvula. Esta función utiliza sensor GPS de Android.
  La aplicación notificará al usuario cuando salga del rango establecido y le informará que la válvula se desactivará.
- El botón “Ver consumo de agua” nos lleva a la activity **VerConsumoDeAgua**

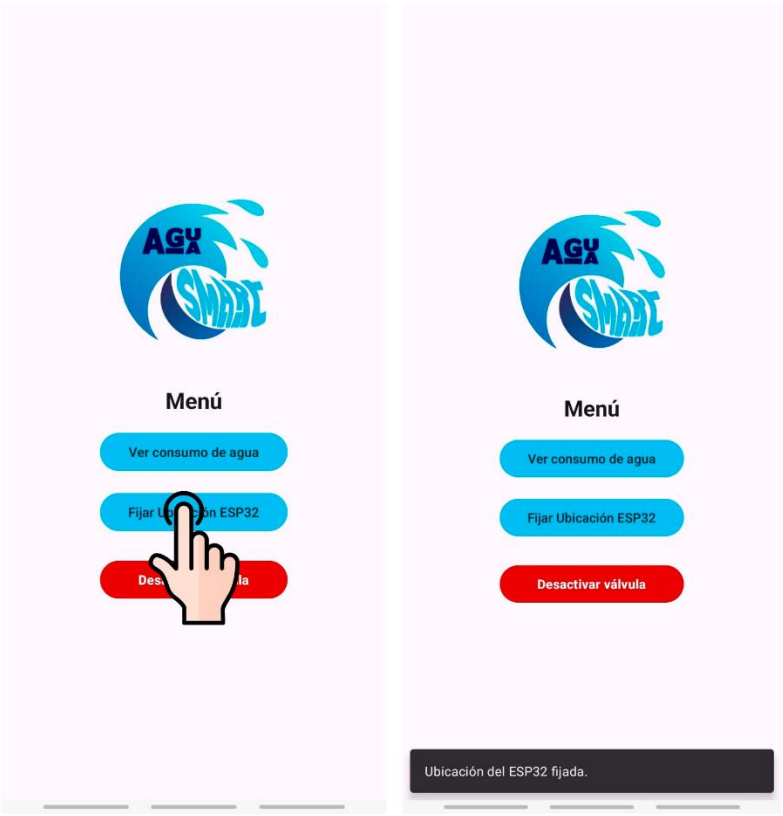


Ilustración 1- Se muestra un snackbar al fijar la ubicación del ESP32

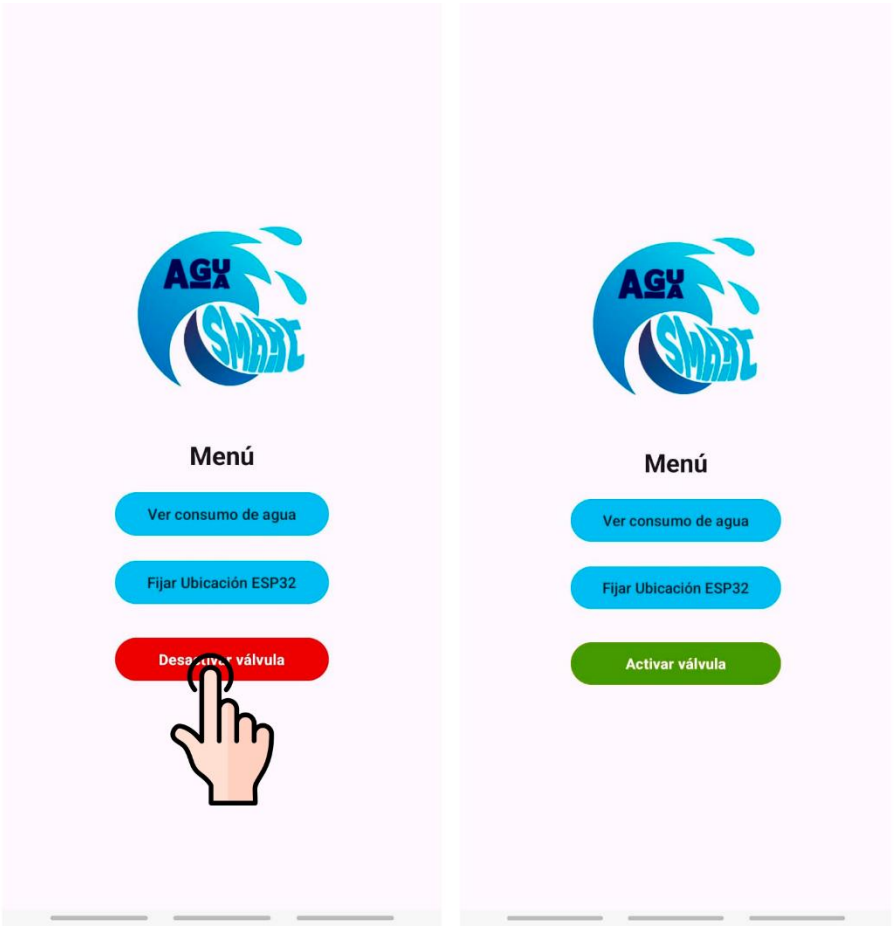


Ilustración 2 - Actualización de la UI por desactivar la válvula

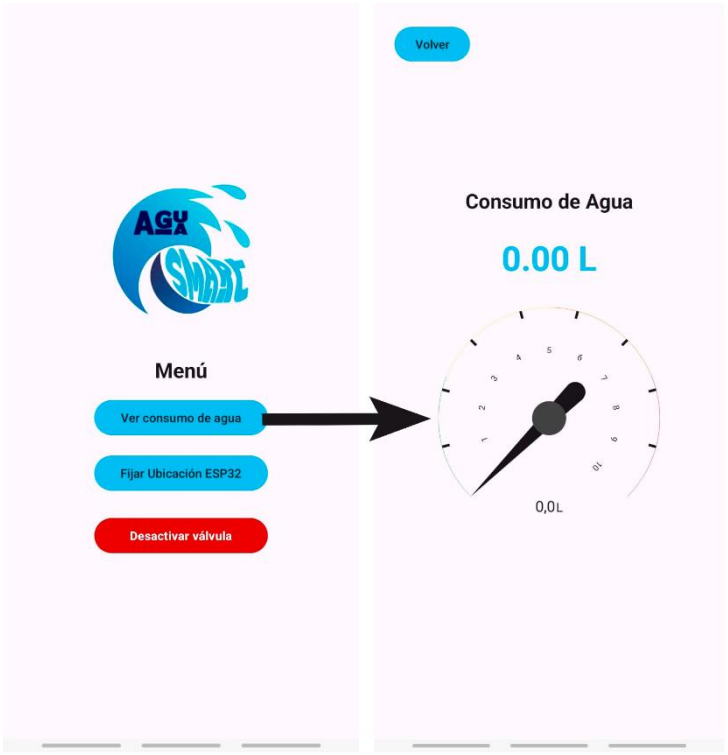


Ilustración 3- Transición de MainActivity a ConsumoActivity

"Ver consumo de agua" Activity

Función:
Esta pantalla nos permite visualizar cual es el consumo de agua actual medido por nuestro sistema.
El medidor de agua cambiará de posición a medida que se vayan superando los distintos umbrales de consumo de agua especificados.
La aplicación alertará al usuario mediante una notificación si se supera el umbral 3 (75% de consumo) y cuando se alcance el umbral 4 (100% de capacidad de consumo).

3 Conclusiones

Con respecto a los problemas que tuvimos que resolver, uno destacable fue la incompatibilidad de la librería que estábamos usando para integrar MQTT a nuestra aplicación Android. Investigando en internet encontramos la versión

actualizada de esa librería que es compatible con la versión de Android que utilizamos a la hora de hacer el desarrollo.

Además, otra cuestión a destacar como aprendizaje realizado por parte del equipo es que logramos aprender como comunicar a través de internet dos dispositivos diferentes, en este caso un teléfono Android con un chip ESP32. Este aprendizaje consideramos que nos permitirá ampliar un poco más la cantidad de herramientas disponibles a nuestro alcance a la hora de tener que resolver algún problema similar a lo largo de nuestra carrera profesional.

4 Referencias

1. Sistemas Embebidos e Internet de las Cosas. Sistemas Operativos Avanzados, Universidad Nacional de La Matanza (2025). Sistemas embebidos e Internet de las Cosas - SOA - Wiki - Unlam

2. Tiempo Real. Sistemas Operativos Avanzados, Universidad Nacional de La Matanza (2025). Tiempo Real - SOA - Wiki - Unlam