SmartWater – Informe IA

Camean Pablo Ezequiel, López Fernando Antonio, Palmieri Marco Tiziano, Quiroga Piegari Lucila, Romero Lucas Nicolas
34520666, 42537132, 45542385, 4156674, 43780360
Martes, Grupo M2

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

Objetivo

El objetivo de este informe es comparar el rendimiento en el proceso de desarrollo y funcionamiento del software utilizando dos procesos, uno utilizando Vibe Coding Parcial y otro de forma total.

Herramientas utilizadas

Modelo usado

Se utiliza el modelo SWE-1 de Windsurf

Prompts utilizados

Quiero que escribas un proyecto con los siguientes datos:
Funcionalidad a desarrollar:

- Se quiere hacer un sistema de control para controlar el consumo de agua de una casa y que este permita activar o desactivar la entrada de agua en base a un consumo determinado.
- El sistema debe accionar una alarma sonora cuando se superan ciertos umbrales de consumo de agua. Se tendrán 3 umbrales de consumo previos al umbral límite, serán al 25%, 50%, y 75% del consumo límite configurado.
- El sistema debe desactivar la entrada de agua cuando se supera el umbral límite de consumo.
- El sistema tendrá un pulsador que permitirá activar o desactivar la entrada de agua independientemente del consumo actual.
- El pulsador reiniciará el consumo a 0 cuando se alcance el consumo límite.

Especificaciones técnicas:

- Tiene que estar desarrollado en lenguaje Wiring para el IDE de Arduino. No tiene que utilizar PlatformIO.
- El código tiene que estar modularizado en distintos archivos para que sea legible y mantenible.
- Debe tener un archivo de configuraciones para centralizar los valores de constantes.
- El código debe estar escrito y comentado en Inglés.
- Tiene que funcionar en una placa de desarrollo ESP32 DevKit V1.
- Tiene que utilizar el SO FreeRTOS.

- Para todas las funciones del sistema deben utilizarse tareas y queues para sincronización de FreeRTOS para no bloquear la ejecución.
- Tiene que implementar el concepto de máquina de estados y para esto debe seguir el patrón de switch anidados.
- **El sistema contará con los siguientes sensores y actuadores:**
    - Caudalímetro (Modelo: YF-S201)(Pin: 5): para calcular el consumo actual de agua.
    - Relay (Modelo: KY-019)(Pin: 16): para activar o desactivar una electroválvula que controla la entrada de agua a la casa.
    - Botón pulsador (Modelo: Genérico)(Pin: 19): utilizado para activar o desactivar el relay e interactuar con el sistema.
    - Buzzer (Modelo: Pasivo Genérico)(Pin: 4): usado para emitir una alarma sonora cuando se alcancen ciertos umbrales de consumo de agua.
    - Pantalla LCD (Modelo: 16x2 con I2C integrado): usada para mostrar el consumo acumulado de agua.
- No debe usarse la función delay()
- El sistema debe utilizar el protocolo MQTT a través de WiFI para:
    - Enviar un log a un tópico cada cierto tiempo que contenga:
    - El consumo actual en Litros.
    - El estado de la electroválvula (activa/inactiva).
    - Recibir comandos a través de otro tópico que tengan el mismo efecto que el accionado del pulsador físico.
- Deben utilizarse Logs en el monitor serial para mostrar información útil que permita ver el estado del sistema y hacer debug.
- Los eventos solo se deben obtener dentro de la función get_input() y solo se pueden cambiar de estados dentro del switch de la máquina de estados.
- Los sensores solo se deben leer dentro de la función get_input() y no debe haber sentencias if dentro de los switches de la máquina de estados.

Luego del prompt inicial:

- Se envía log de error de compilación y se pide fix.
- Se repite el punto anterior.
- Se repite el punto anterior.
- Se pide fix porque el sistema se reinicia por corrupción de memoria.
- Se pide fix porque al presionar el botón no cambia el estado del relay.
- Se repite punto anterior
- Se pide que el buzzer suene solo durante un tiempo
- Se envía log de error de compilación y se pide fix.
- Se pide fix porque transiciona mal entre estados
- Se repite el punto anterior
- Se repite el punto anterior

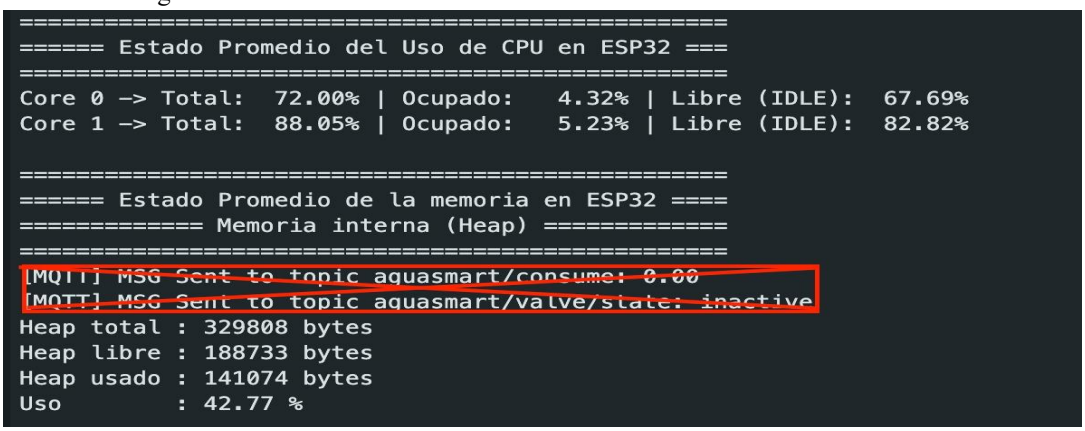
Tiempo utilizado

Desde el primer prompt hasta alcanzar el funcionamiento deseado transcurrieron aproximadamente **3 horas**. Esto se contrasta con las **12 horas** necesarias para el desarrollo utilizando vibe coding parcial.

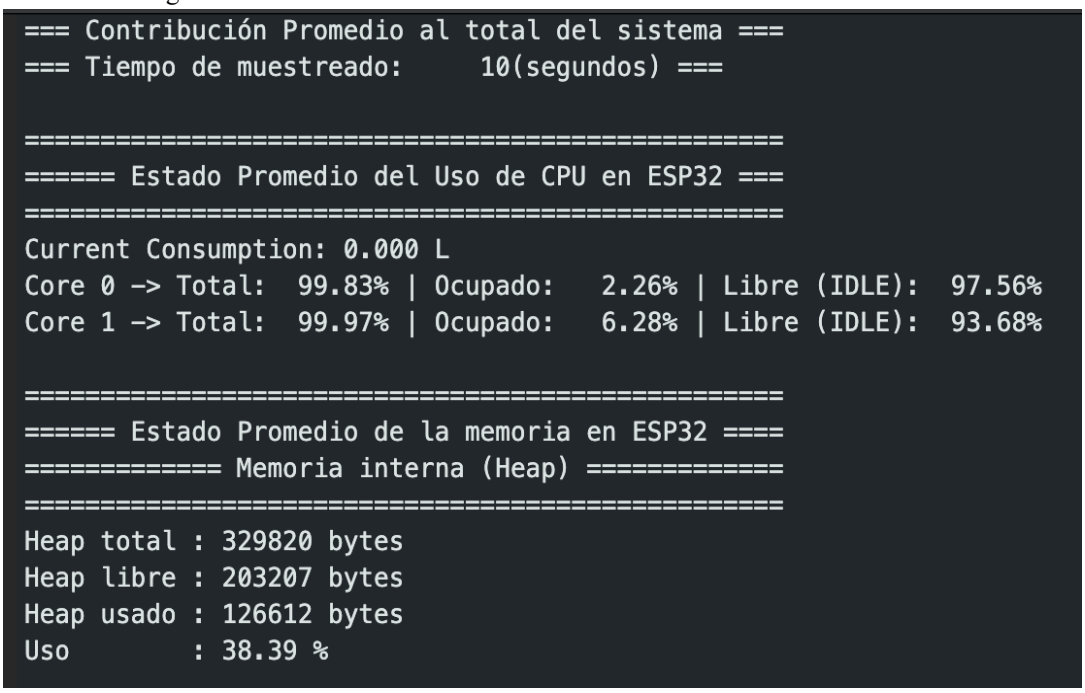
Métricas de uso

Sistema en reposo

- Sin vibe coding



- Con vibe coding



Sistema midiendo consumo de agua

- Sin vibe coding

```
=== Contribución Promedio al total del sistema ===
=== Tiempo de muestreado:     10(segundos) ===

====================================================
====== Estado Promedio del Uso de CPU en ESP32 ===
====================================================
Core 0 -> Total:  77.24% | Ocupado:   4.24% | Libre (IDLE):  73.00%
Core 1 -> Total:  89.93% | Ocupado:   5.21% | Libre (IDLE):  84.72%
[MQTT] MSG Sent to topic aguasmart/consume: 4.43

====================================================
[MQTT] MSG Sent to topic aguasmart/valve/state: inactive
====== Estado Promedio de la memoria en ESP32 ====
============= Memoria interna (Heap) =============
====================================================
Heap total : 329808 bytes
Heap libre : 187852 bytes
Heap usado : 141956 bytes
Uso        : 43.04 %
```

- Con vibe coding

```
=== Contribución Promedio al total del sistema ===
=== Tiempo de muestreado:     10(segundos) ===

====================================================
====== Estado Promedio del Uso de CPU en ESP32 ===
====================================================
Current Consumption: 2.449 L
Core 0 -> Total:  99.84% | Ocupado:   2.17% | Libre (IDLE):  97.67%
Core 1 -> Total:  99.94% | Ocupado:   6.20% | Libre (IDLE):  93.74%
Warning 3 - Buzzer activated for 3 seconds

====================================================
====== Estado Promedio de la memoria en ESP32 ====
============= Memoria interna (Heap) =============
====================================================
Heap total : 329796 bytes
Heap libre : 200117 bytes
Heap usado : 129678 bytes
Uso        : 39.32 %
```

Rúbricas

Sin Vibe Coding

Legibilidad y Mantenibilidad del Embebido

CRITERIOS	SUFICIENTEMENTE LOGRADO (A)	MEDIANAMENTE LOGRADO (B)	INSUFICIENTEMENTE LOGRADO (C)	Puntaje	Eval
PATRÓN DE MÁQUINA DE ESTADOS	-respeta un 90% el patrón de máquina de estados	-respeta un 70% el patrón de máquina de estados	Respeta el patrón de máquina de estados menos de un 70%	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
TAMAÑO DE FUNCIONES	Funciones < a 30 líneas	30 líneas< Funciones < 50 líneas	Funciones >50 líneas	A: Hasta 20 pts. B: Hasta 10 pts C: 0	A
NOMBRES DESCRIPTIVOS VARIABLES Y MÉTODOS	Utiliza nombres descriptivos usando convención	Utiliza nombres descriptivos, pero sin usar convención	Los nombres no son descriptivos	A: Hasta 15 pts. B: Hasta 10 pts. C: 0	A
COMENTARIOS EN CODIGO	-posee el 60% del código con comentarios explicando que hace cada parte el mismo	-posee el 20% del código con comentarios explicando cada parte del código	-posee menos del 20% del código con comentarios explicando cada parte del código	A: Hasta 10 pts. B: Hasta 5 pts. C: 0	A
CODIGO MODULAR	- el código está separado en módulos y no se encuentra acoplado	El código está separado en módulos, pero se encuentra medianamente acoplado	El código no es modular y está fuertemente acoplado	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
CANTIDAD DE WARNING ANDROID	Tiene menos de 10 warnings	10<Tiene entre<20	Tiene más de 20 warnings	A: Hasta 15 pts. B: Hasta 10 pts. C: 0	A
TOTAL				**100**	100

Funcionalidad Embebido

CRITERIOS	SUFICIENTEMENTE LOGRADO (A)	MEDIANAMENTE LOGRADO (B)	INSUFICIENTEMENTE LOGRADO (C)	Puntaje	Eval
FUNCIONA SIN ERRORES	-funciona el 100% de las veces sin errores	-funciona el 70% de las veces sin errores	Funciona el 50% de las veces sin errores	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
EMPLEA TAREAS DE FREERTOS	-utiliza varias tareas de freeRTOS	-Utiliza una sola tarea con freeRTOS	-no utiliza freeRTOS	A: Hasta 20 pts. B: Hasta 10 pts C: 0	A

NO USA ESPERAS BLOQUEANTES	Reemplaza las esperas bloqueantes por tareas con freeRTOS Utiliza temporizadores con freeRTOS	-utiliza temporizadores sin freeRTOS	-utiliza espera bloqueante	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
UTILIZA LOS SENSORES Y ACTUADORES SOLICITADOS	-utiliza la cantidad de sensores y actuadores solicitados de manera eficiente	-utiliza la cantidad de sensores y actuadores de manera poco efectiva	-no utiliza la cantidad de sensores y actuadores solicitados o no funcionan de manera correcta.	A: Hasta 20 pts. B: Hasta 12 pts. C: 0	A
SE COMUNICA CON BROKER CON MQTT	-funciona correctamente el sistema usando la comunicación por MQTT sin problemas	-funciona el sistema correctamente, pero posee algunos problemas de funcionamiento el sistema.	No funciona la comunicación MQTT Funciona la comunicación, pero falla en el funcionamiento del sistema	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
TOTAL				**100**	100

Con Vibe Coding

Legibilidad y Mantenibilidad del Embebido

CRITERIOS	SUFICIENTEMENTE LOGRADO (A)	MEDIANAMENTE LOGRADO (B)	INSUFICIENTEMENTE LOGRADO (C)	Puntaje	Eval
PATRÓN DE MÁQUINA DE ESTADOS	-respeta un 90% el patrón de máquina de estados	-respeta un 70% el patrón de máquina de estados	Respeta el patrón de máquina de estados menos de un 70%	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
TAMAÑO DE FUNCIONES	Funciones < a 30 líneas	30 líneas< Funciones < 50 líneas	Funciones >50 líneas	A: Hasta 20 pts. B: Hasta 10 pts C: 0	C
NOMBRES DESCRIPTIVOS VARIABLES Y MÉTODOS	Utiliza nombres descriptivos usando convención	Utiliza nombres descriptivos, pero sin usar convención	Los nombres no son descriptivos	A: Hasta 15 pts. B: Hasta 10 pts. C: 0	A
COMENTARIOS EN CODIGO	-posee el 60% del código con comentarios explicando que hace cada parte el mismo	-posee el 20% del código con comentarios explicando cada parte del código	-posee menos del 20% del código con comentarios explicando cada parte del código	A: Hasta 10 pts. B: Hasta 5 pts. C: 0	A
CODIGO MODULAR	- el código está separado en módulos y no se encuentra acoplado	El código está separado en módulos, pero se encuentra medianamente acoplado	El código no es modular y está fuertemente acoplado	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	C
CANTIDAD DE WARNING ANDROID	Tiene menos de 10 warnings	10 < Tiene entre < 20	Tiene más de 20 warnings	A: Hasta 15 pts. B: Hasta 10 pts. C: 0	A
TOTAL				**100**	**60**

Funcionalidad Embebido

CRITERIOS	SUFICIENTEMENTE LOGRADO (A)	MEDIANAMENTE LOGRADO (B)	INSUFICIENTEMENTE LOGRADO (C)	Puntaje	Eval
FUNCIONA SIN ERRORES	-funciona el 100% de las veces sin errores	-funciona el 70% de las veces sin errores	Funciona el 50% de las veces sin errores	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
EMPLEA TAREAS DE FREERTOS	-utiliza varias tareas de freeRTOS	-Utiliza una sola tarea con freeRTOS	-no utiliza freeRTOS	A: Hasta 20 pts. B: Hasta 10 pts C: 0	A
NO USA ESPERAS BLOQUEANTES	Reemplaza las esperas bloqueantes por tareas con freeRTOS Utiliza temporizadores con freeRTOS	-utiliza temporizadores sin freeRTOS	-utiliza espera bloqueante	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	C
UTILIZA LOS SENSORES Y ACTUADORES SOLICITADOS	-utiliza la cantidad de sensores y actuadores solicitados de manera eficiente	-utiliza la cantidad de sensores y actuadores de manera poco efectiva	-no utiliza la cantidad de sensores y actuadores solicitados o no funcionan de manera correcta.	A: Hasta 20 pts. B: Hasta 12 pts. C: 0	A
SE COMUNICA CON BROKER CON MQTT	-funciona correctamente el sistema usando la comunicación por MQTT sin problemas	-funciona el sistema correctamente, pero posee algunos problemas de funcionamiento el sistema.	No funciona la comunicación MQTT Funciona la comunicación, pero falla en el funcionamiento del sistema	A: Hasta 20 pts. B: Hasta 15 pts. C: 0	A
TOTAL				**100**	80

Conclusiones

Luego de realizar el desarrollo del proyecto utilizando vibe coding total y parcial, y analizando las métricas obtenidas junto a la evaluación de las rúbricas dispuestas podemos ver que:

- Es determinante el tiempo necesario para obtener el software funcional de 4hs con vibe coding contra 12hs sin este (x3 veces más rápido el desarrollo).
- La utilización de memoria es ligeramente inferior con vibe coding.
- La utilización de CPU es similar en ambos.
- La legibilidad y mantenibilidad es superior sin vibe coding ya que es escrito por y para personas.

Al analizar estos puntos podemos concluir que el desarrollo con vibe coding total impulsa la velocidad para obtener un producto funcional que puede servir como POC (prueba de concepto) para evaluar viabilidad del proyecto y otras cuestiones. En cambio, el desarrollo con vibe coding parcial se destaca por la resiliencia y mantenibilidad del sistema más orientado a un producto software final con posibilidad de ampliación y mejor mantenibilidad a futuro.