Universidad Nacional de Tucumán

Interfaz de usuario en STM32

Electrónica IV

Trabajo Integrador

Fecha de Exposición

Integrantes:

Juan Jose Armando, Gerez Jimenez DNI: 41649897

Índice

1- Introducción

El presente informe describe el desarrollo de una interfaz de usuario embebida utilizando una placa de desarrollo STM32 Nucleo-L031K6. El sistema consta de una pantalla LCD alfanumérica de 16x2 conectada en modo paralelo de 4 bits, un LED de diagnóstico y tres pulsadores configurados como entradas con resistencias pull-up internas. Esta combinación de elementos se emplea para implementar un menú interactivo de tres estados (Inicio, Información, Configuración) navegable mediante distintos patrones de pulsación: simple, doble y larga duración. La motivación de este trabajo radica en diseñar un interfaz humano-máquina básica sobre un microcontrolador de recursos limitados, aplicando conceptos de electrónica digital y sistemas embebidos aprendidos en el ámbito académico. Se enmarca en el contexto de la materia Electrónica IV de la FACET (UNT), donde se integran conocimientos de programación de microcontroladores, electrónica digital y técnicas de control de sistemas.

En el contexto tecnológico actual, las interfaces embebidas de bajo nivel siguen desempeñando un papel importante en dispositivos de uso cotidiano, a pesar de la disponibilidad de interfaces gráficas avanzadas. Muchas soluciones modernas utilizan pantallas gráficas táctiles o protocolos seriales de alta abstracción; sin embargo, en sistemas de recursos limitados o en entornos educativos se recurre frecuentemente a displays de caracteres (como LCD 16x2) y pulsadores mecánicos por su simplicidad y bajo costo. Estas interfaces clásicas permiten ilustrar directamente el manejo de hardware a bajo nivel (pines GPIO, temporizaciones, etc.), sentando las bases para comprender sistemas más complejos. No obstante, presentan desafíos técnicos como el rebote mecánico (bounce) de los pulsadores y la necesidad de control manual de periféricos (p. ej., gestión de la secuencia de inicialización del LCD). El fenómeno de rebote implica que al pulsar o soltar un botón, los contactos metálicos vibran generando múltiples transiciones espurias de la señal digital[1]. Esto puede ocasionar lecturas erróneas (por ejemplo, el microcontrolador podría interpretar múltiples pulsaciones cuando el usuario solo presionó una vez). Para abordar este problema se emplea la técnica de antirrebote, ya sea por software (ignorando cambios transitorios durante un intervalo corto) o por hardware (filtros RC, circuitos Schmitt-Trigger, etc.), con

el fin de garantizar una lectura estable de cada pulsación[1]. Por otro lado, la interfaz paralela de la pantalla LCD requiere manejar varias líneas digitales de manera coordinada para enviar datos y comandos. En este proyecto se optó por el modo de 4 bits, que utiliza únicamente los 4 pines de datos más significativos (D4–D7) del LCD, además de las líneas de control (RS, E) y alimentación. Este modo reduce la cantidad de pines GPIO necesarios (7 conexiones en total) en comparación con el modo de 8 bits que requiere hasta 11 líneas, a cambio de enviar los datos en dos nibbles en lugar de uno solo[2]. Dicha configuración es común en sistemas embebidos con microcontroladores de bajo costo, donde optimizar el uso de pines es prioritario. Finalmente, en cuanto al estado del arte de las interfaces embebidas de bajo nivel, cabe mencionar que persiste la utilización de máquinas de estado finitas implementadas en firmware para la navegación de menús simples, combinada con técnicas tradicionales de depuración (LEDs indicadores) y simulación por software. Este proyecto aplica esos conceptos, alineándose con prácticas estándar en la industria y en laboratorios de enseñanza de sistemas embebidos.

2- Objetivos

Los objetivos principales del sistema implementado son:

- Implementar una interfaz de usuario básica en un microcontrolador STM32, que permita navegar entre diferentes pantallas de menú (Inicio, Información, Configuración) de forma intuitiva mediante un solo botón multifunción.
- Detectar y diferenciar patrones de pulsación (clic simple, doble clic y pulsación larga) sobre el botón de menú, garantizando la confiabilidad de la lectura mediante técnicas de debounce de software.
- Permitir el ajuste de un parámetro interno del sistema en tiempo real, utilizando dos pulsadores adicionales (incremento y decremento) activos únicamente en el modo Configuración, mostrando el resultado en la pantalla LCD.
- Aplicar buenas prácticas de diseño embebido, incluyendo la correcta inicialización y manejo de una pantalla LCD en modo 4 bits, el uso de un LED de diagnóstico para señalización del estado del sistema, y la estructuración del programa en una lógica de máquina de estados.
- Verificar el correcto funcionamiento del sistema en simulación (usando Wokwi) analizando señales con herramientas como analizadores lógicos (PulseView) para confirmar el filtrado de rebotes y la temporización de eventos. Se busca corroborar que se cumplen los tiempos de pulsación definidos y que el LCD refleja adecuadamente cada cambio de estado.

3- Desarrollo

Hardware utilizado: La plataforma central es la placa de desarrollo **STM32 Nucleo-L031K6**, la cual integra un microcontrolador ARM Cortex-M0+ de 32 bits a 32 MHz, 32 KB de Flash y 8 KB de SRAM[3]. Esta placa Nucleo-32 proporciona conectores que permiten acceso a múltiples pines de GPIO configurables. En particular, cuenta con un LED de usuario (LD3) incorporado en el microcontrolador (identificado como D13 en la numeración Arduino)[3], el cual se ilumina cuando dicho pin se lleva a nivel alto. Este LED se utilizó como indicador de diagnóstico en el proyecto. Además, se dispusieron **tres pulsadores** externos conectados a pines de la placa, configurados como entradas con resistencia **pull-up** interna habilitada. En la configuración pull-up, el pin se mantiene normalmente en nivel lógico alto (1) y al presionar el botón se conecta a tierra, proporcionando un nivel bajo (0) que el microcontrolador puede detectar[1]. Esta disposición es ventajosa ya que elimina la necesidad de resistencias físicas adicionales y asegura un estado definido (HIGH) cuando el pulsador está inactivo. Cada pulsador está cableado entre el pin de entrada correspondiente y tierra; al activarse, genera una transición de 1 a 0 lógica en el pin.

Para la interfaz de salida visual se emplea un **display LCD 16x2 de caracteres**. El LCD se conectó al STM32 en **modo de 4 bits**, aprovechando 4 líneas de datos (D4 a D7) hacia pines GPIO configurados como salidas digitales, más dos líneas de control: **RS** (Register Select) para seleccionar comandos/datos y **E** (Enable) para temporizar la transferencia. La línea R/W del LCD se puso a masa (configuración solo-escritura) para simplificar el control, dado que solo se requieren escribir datos o comandos al display. También se conectó el pin **V0 (contraste)** del LCD a un divisor de tensión con potenciómetro para ajustar la visibilidad de los caracteres, y se alimentó la retroiluminación con 5 V a través de sus pines A/K. En la Tabla (ver Anexo I: Esquemático) se resumen las conexiones principales del hardware:

- LCD D4–D7 conectados a pines GPIO [D3-D6] del STM32 (salidas digitales).
- LCD RS conectado a pin GPIO [D11] (salida digital).

- LCD E conectado a pin GPIO [D12] (salida digital).
- LCD R/W conectado a GND (solo modo escritura).
- LCD Vcc a 5 V, GND a 0 V, V0 a potenciómetro (control de contraste).
- LED de usuario (D13) utilizado como indicador (salida digital).
- Pulsador MENU conectado a pin GPIO [D2-entrada con pull-up].
- Pulsador UP (incremento) conectado a pin GPIO [A0-entrada con pull-up].
- Pulsador DOWN (decremento) conectado a pin GPIO [A1-entrada con pull-up].

La selección de operar el LCD en 4 bits, en lugar de 8, obedece a la **eficiencia en el uso de pines**: utilizando solo 7 líneas en total (4 datos + 3 control incluyendo Enable, RS y, opcionalmente, una línea de backlight) se libera el resto de los GPIO para otras funciones[2]. Si bien el modo de 8 bits ofrece una transmisión ligeramente más rápida de datos (envía el byte completo en un solo ciclo en vez de en dos mitades), en este proyecto la velocidad de actualización del LCD no es crítica, por lo que se privilegió la economía de pines.

Descripción del firmware y lógica de estados: El software se desarrolló en el entorno para STM32 (“STM32duino”) usando la librería LiquidCrystal. Se implementó una máquina de estados finita con tres estados: **MENU_INICIO**, **MENU_INFO** y **MENU_CONF**. Una variable global **menuActual** indica el estado activo. El programa sigue un superloop que: (i) hace parpadear el LED de usuario (~1 Hz) como señal de vida, (ii) atiende el pulsador principal para navegar el menú y (iii) en MENU_CONF procesa UP/DOWN para ajustar un valor configurable.

Las transiciones implementadas son:

- Clic simple (pulsación corta) → **MENU_INICIO**.
- Doble clic (dos pulsaciones cortas dentro de una ventana) → **MENU_INFO**.
- Pulsación larga → **MENU_CONF**.

La función **mostrarMenu()** centraliza la actualización del LCD en cada cambio de estado, mostrando la pantalla correspondiente y, en **MENU_CONF**, el valor actual de la variable de configuración.

Manejo de temporización y anti-rebote: Dado el rebote mecánico de los pulsadores (Ver figura 3-1, 3-2, 3-3), se implementó un antirrebote por software basado en dos etapas complementarias:

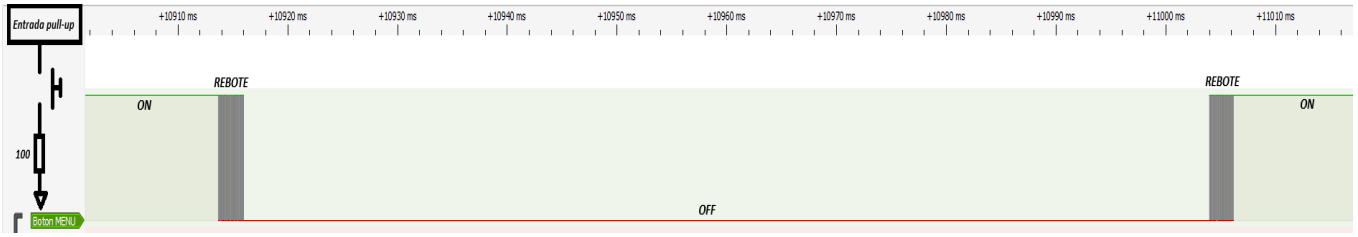

Figura 3-1. Pulsado del botón con sus rebotes al oprimir y soltar.

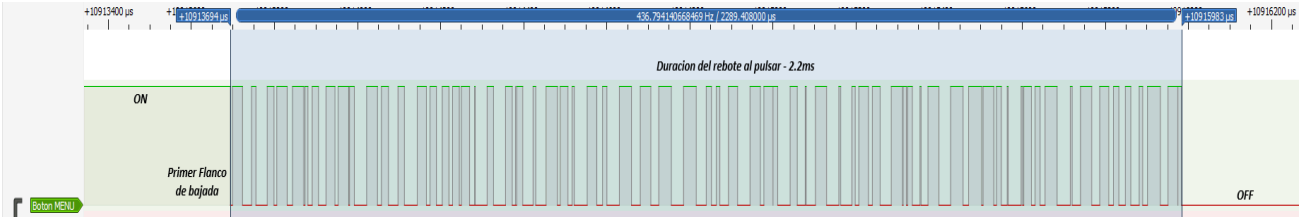

Figura 3-2. Zoom al rebote al oprimir (duración 2.2ms).

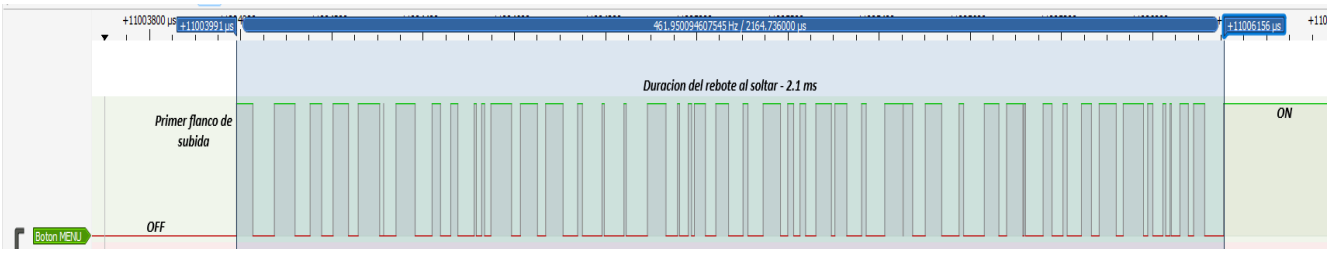

Figura 3-3. Zoom al rebote al soltar (duracion 2.1ms)

(a) Filtrado de flanco con verificación retardada (debounce básico).

Cada vez que la señal del botón cambia de estado, el firmware realiza:

1. Detección de flanco (bajada para inicio de pulsación o subida para liberación) comparando la lectura actual con el estado anterior.
2. Espera fija de verificación ANTI_REBOTE_MS ≈ 30 ms mediante delay(...).

3. Relectura del pin: si el estado persiste (estable), entonces se confirma el evento; si cambió, se descarta por tratarse de rebote.

Con este filtro, al presionar el botón se garantiza que los "peines" de rebote (oscilaciones de pocos milisegundos) no se contabilicen como múltiples pulsaciones.

(b) Ventanas temporales que "encapsulan" el rebote y clasifican la pulsación.

Además del filtro de flanco, se usan marcas temporales con millis():

- tBajada: instante confirmado del flanco de bajada.
- tSubida: instante confirmado del flanco de subida.
- dur = tSubida − tBajada: duración de la pulsación.
- tPrimeraSubida: marca para abrir la ventana de doble clic.

La clasificación del botón MENU es:

- Pulsación larga si dur ≥ PULSO_LARGO_MS (≈800 ms). La acción se ejecuta inmediatamente tras la liberación (flanco de subida confirmado).
- Doble clic si, tras un primer clic corto, se detecta un segundo clic cuyo intervalo cumple tBajada(segundo) − tPrimeraSubida(primero) < DOBLE_CLIC_MS (≈350 ms). En ese caso se dispara el evento de doble clic y se cancela el clic pendiente.
- Clic simple si la ventana de doble clic expira sin segundo clic: millis() − tPrimeraSubida > DOBLE_CLIC_MS.

Este segundo nivel de temporización encapsula cualquier rebote remanente aun si el primer filtro fuese insuficiente, y hace mutuamente excluyentes los tres patrones (corto, doble, largo). En particular:

- MENU toma la decisión en la subida, cuando se conoce la duración real.
- UP/DOWN actúan en la bajada (más inmediata) y solo en el estado CONFIG, con su propio antirrebote de 30 ms, evitando repeticiones espurias.

Justificación de parámetros.

- ANTI_REBOTE_MS ≈ 30 ms cubre rebotes típicos de 1–5 ms con margen.

- DOBLE_CLIC_MS ≈ 350 ms es un intervalo natural para el gesto de doble pulsación.
- PULSO_LARGO_MS ≈ 800 ms evita activaciones accidentales y es cómodo para el usuario.

[1].

Funcionamiento de UP/DOWN en configuración: Los pulsadores UP y DOWN solo tienen efecto cuando menuActual == MENU_CONF. Cada pulsación válida incrementa o decrementa la variable **valorConfig** en una unidad, respetando límites (VAL_MIN = 0, VAL_MAX = 99). El LCD refleja en tiempo real el valor ajustado, brindando retroalimentación inmediata al usuario. En los demás estados las entradas UP/DOWN se ignoran para prevenir cambios no deseados.

Estructura del código: Se utilizaron constantes simbólicas para pines y tiempos (PIN_BTN, PIN_UP, PIN_DOWN, PIN_LED, ANTI_REBOTE_MS, DOBLE_CLIC_MS, PULSO_LARGO_MS) y funciones específicas para tareas repetitivas (p. ej., mostrarMenu()), buscando claridad y mantenibilidad.

4- Resultados

En el entorno de Wokwi se recreó el montaje con la placa STM32 Nucleo-L031K6, un LCD 16×2 operado en modo 4 bits, tres pulsadores con pull-up interno y el LED de usuario (D13). El sketch (con la librería LiquidCrystal) ejecutó el parpadeo del LED (~1 Hz) como indicador de actividad y gestionó la navegación del menú. La temporización se basó en millis() y retardos con delay(), sin configurar directamente el temporizador SysTick [3].

El LCD mostró los mensajes esperados en cada estado:

- MENU_INICIO: “Menu principal”.
- MENU_INFO: “Info del sistema”.
- MENU_CONF: “Config: <valor>”, actualizando en tiempo real valorConfig con UP/DOWN.

Se trabajaron los tres patrones del pulsador MENU con los umbrales del sketch (ANTI_REBOTE_MS = 30 ms, DOBLE_CLIC_MS = 350 ms, PULSO_LARGO_MS = 800 ms):

- Pulsación corta → confirmada como clic simple al expirar la ventana.
- Doble clic → reconocido cuando el segundo evento ocurrió dentro de la ventana definida.
- Pulsación larga → transición inmediata a Configuración al superar el umbral.

Validación de la clasificación temporal (MENU):

En la simulación se confirmaron los umbrales de clasificación definidos en el firmware. La Fig. 4-1 muestra una pulsación larga del botón MENU: la línea del botón permanece en nivel bajo ≈1,2 s; al liberar (subida confirmada) la lógica identifica dur ≥ 800 ms y transita a CONFIG sin esperar ventana de doble clic.

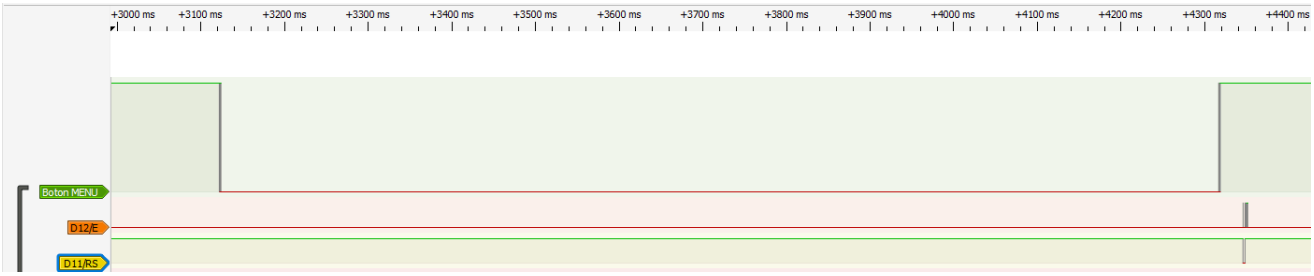

Figura 4-1. Se observa el tramo sostenido en LOW y, tras la subida, la actualización del LCD (actividad en E/RS 30 ms después) correspondiente al ingreso a CONFIG.

En la Fig. 4-2 y Fig. 4-3 se ilustran los rebotes típicos en el flanco de bajada y flanco de subida, del orden de ≈2 ms. Nótese que el firmware no reacciona hasta completar el ANTI_REBOTE_MS ≈ 30 ms, por lo que solo se registra un evento.

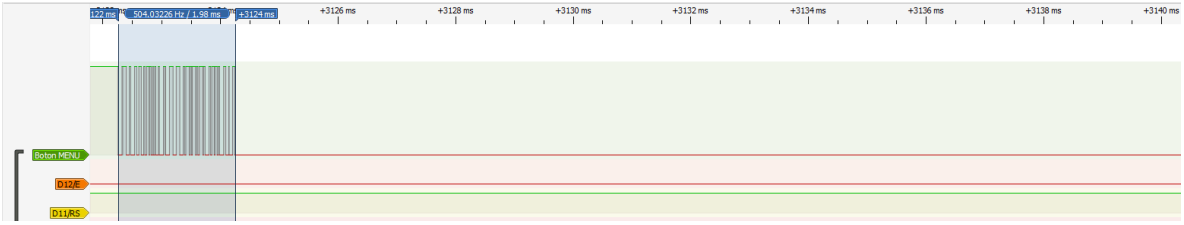

Fig. 4-2. Rebote al pulsar (flanco descendente, ≈1,98 ms). Peine de oscilaciones 1↔0 antes de estabilizar en LOW; sin reacción del sistema hasta validar la estabilidad.

Fig. 4-3. Rebote al soltar (flanco ascendente, ≈2 ms). Oscilaciones transitorias 0↔1 antes de estabilizar en HIGH; el evento se confirma tras el retardo de verificación.

La Fig. 4-4 cuantifica la latencia de antirrebote: entre el flanco y la primera actividad del micro (p. ej., escritura al LCD) se mide ≈29 ms, coherente con el parámetro programado.

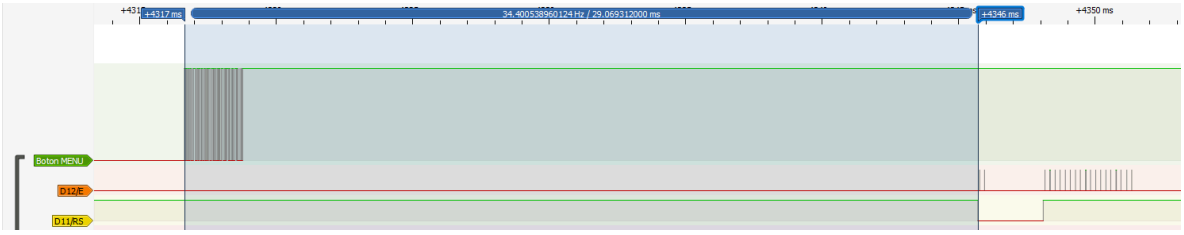

Fig. 4-4. Ventana de antirrebote hasta la respuesta del micro (≈29 ms). Confirma que el sistema ignora transitorios dentro de esa ventana.

Clic simple y doble clic:

En la Fig. 4-5 se muestra un clic simple: tras la liberación, el firmware espera a que expire la ventana de doble clic (≈350 ms) antes de confirmar el evento y actualizar la pantalla a INICIO.

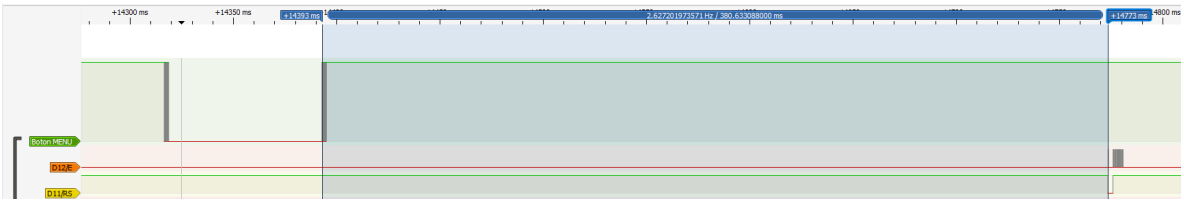

Fig. 4-5. Pulsación corta de MENU: ventana de doble clic + antirrebote (≈380 ms). El retardo visible corresponde a la suma de la ventana de decisión y el debounce.

La Fig. 4-6 evidencia un doble clic: dos pulsaciones dentro de ≈252 ms entre ellas. Al segundo evento, el sistema no espera más y transita de inmediato a INFO.

Fig. 4-6. Doble clic de MENU (dos pulsos en ≈252 ms). El intervalo cae por debajo de DOBLE_CLIC_MS, validando la transición a INFO.

La **Fig. 4-7** destaca que, incluso durante el doble clic, el rebote queda **contenido** por el antirrebote (≈29,1 ms), evitando conteos múltiples dentro de una misma pulsación.

Fig. 4-7. Ventana de antirrebote durante el doble clic (≈29,1 ms). El peinado de rebote no afecta la contabilización de eventos.

UP/DOWN en modo Configuración:

Finalmente, la Fig. 4-8 muestra el comportamiento de UP (o DOWN) en CONFIG: la acción se dispara en el flanco de bajada tras ≈29 ms de antirrebote, con la consiguiente actualización del LCD ("Config: valor").

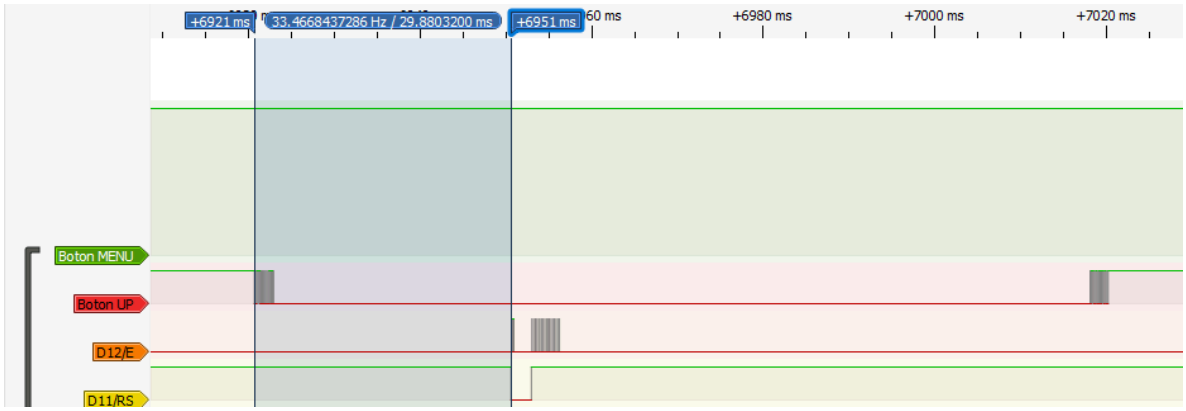

Fig. 4-8. Antirrebote en el botón de incremento (≈29 ms) y actualización del LCD.

Estas mediciones en simulación confirman que el antirrebote por software (≈30 ms) elimina eficazmente los transitorios de 1–3 ms en ambos flancos y que la lógica temporal (PULSO_LARGO_MS, DOBLE_CLIC_MS) clasifica correctamente los patrones de pulsación, con actividad observable en el bus del LCD (RS/E) inmediatamente después de la decisión. Así, la máquina de estados no genera transiciones espurias y presenta tiempos de respuesta acordes a la interacción humana.

5- Conclusiones

Se implementó con éxito una interfaz de tres estados sobre la Nucleo-L031K6, operada con un único pulsador multifunción (corto, doble, largo) y dos pulsadores auxiliares (UP/DOWN) para ajuste en configuración. La detección fue fiable gracias al antirrebote por software y a la clasificación temporal basada en millis() y delay () (sin necesidad de configurar SysTick/HAL explícitamente). El LCD 16×2 en modo 4 bits demostró ser adecuado y eficiente en uso de pines; las entradas con pull-up internas simplificaron el hardware sin comprometer robustez.

El sistema se mostró estable: no se apreciaron transiciones espurias ni pantallas inconsistentes durante sesiones prolongadas. Los tiempos de respuesta (debounce ~20–30 ms y ventana de doble clic ~350 ms) resultaron naturales para el usuario. La arquitectura empleada (superloop + máquina de estados + debounce) deja una base clara para ampliar el menú, añadir retroalimentación acústica (buzzer), incorporar temporizadores de inactividad o migrar a interrupciones EXTI y modos de bajo consumo. A futuro, puede añadirse registro no volátil (EEPROM de datos interna de 1 KB disponible en la familia L0) y/o telemetría por USART/USB-CDC para persistir el parámetro y auditar eventos [3].

6- Referencias

- [1] Pulsadores y antirrebote con Arduino

  http://arduparatodos.blogspot.com/2017/01/pulsadores-y-antirrebote-con-arduino.html

- [2] Todo sobre LCD 16x2

  https://www.allelcoelec.es/blog/Everything-About-LCD-16X2.html

- [3] Wokwi. (2025). Documentación del simulador Wokwi para placa STM32 Nucleo-L031K6. Recuperado de https://docs.wokwi.com/parts/board-st-nucleo-l031k6.
- [4] Table 13. Arduino Nano Connectors On Nucleo-L031K6 - ST NUCLEO-F031K6 User Manual.

  https://www.manualslib.com/manual/1934873/St-Nucleo-F031k6.html?page=26

- Volentini, E. - Teodovich, L. (2025). Apuntes de Electrónica IV – Teoría. Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán. Material de cátedra sobre microcontroladores ARM Cortex y sistemas embebidos.
- Saravia, T. G. (2013). Diseño e implementación de sistemas embebidos con PIC. Buenos Aires: MC Electronics.
- Ogata, K. (2010). Ingeniería de Control Moderna. México DF: Pearson Educación.
- Código fuente del proyecto Interfaz de Usuario en STM32 (2025). Archivo ino y archivos de cabecera, provistos por el autor (ver Apéndice A).

7- Apéndices

Apéndice A. Código Fuente Principal :

El código fuente completo se adjunta en archivos separados en el repositorio del proyecto.

8- Anexos

Anexo I: Esquemático del Circuito.

Esquema eléctrico del sistema implementado, mostrando la conexión entre la placa Nucleo-L031K6, los pulsadores (MENU, UP, DOWN), la pantalla LCD 16x2 y los componentes auxiliares (potenciómetro de contraste, etc.).

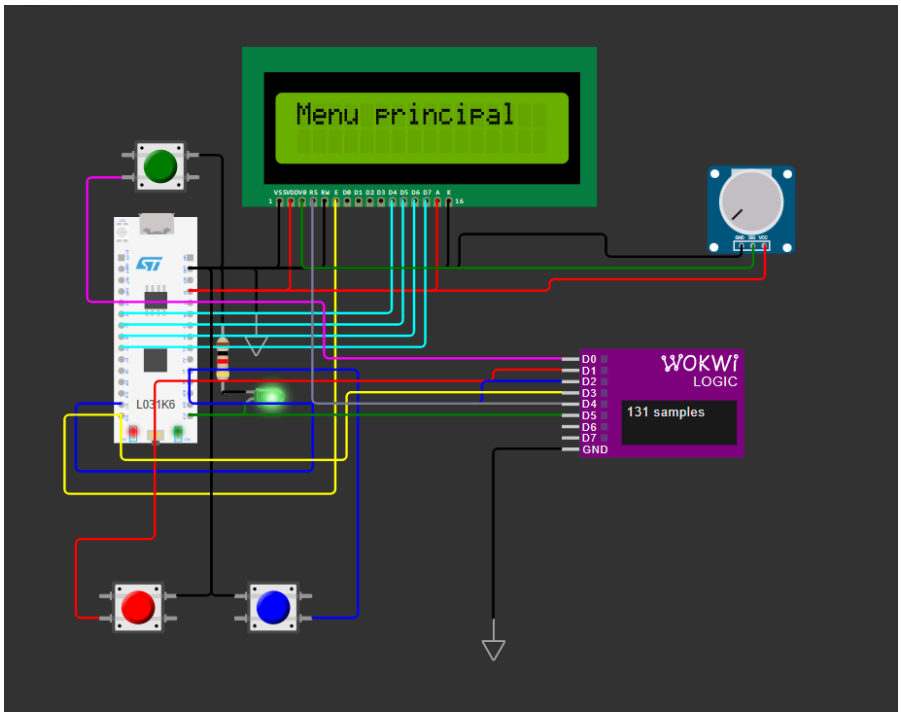

Figura 8-1: Diagrama esquemático en wokwi con identificadores de pines y conexiones.

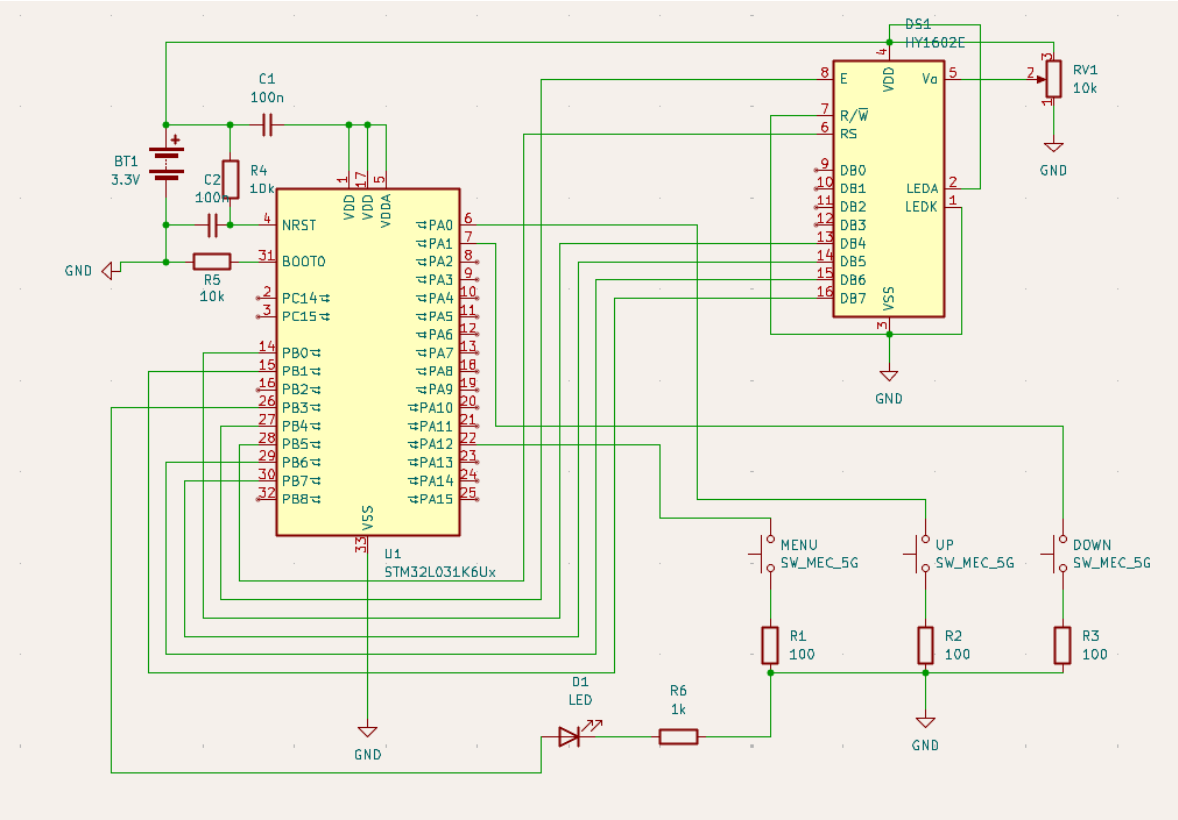

Figura 8-2: Diagrama esquemático en Kicad con identificadores de pines y conexiones.

Nota: En Kicad no se encontró un STM32 que use la misma nomenclatura de Wokwi para sus pines. Para poder hacer el conexionado se hizo uso del datasheet del micro[4]:

LCD 16×2 (HD44780, modo 4 bits)

- *RS → PB5 (D11)*
- *E → PB4 (D12)*
- *D4 → PB0 (D3)*
- *D5 → PB7 (D4)*
- *D6 → PB6 (D5)*
- *D7 → PB1 (D6)*
- *R/W → GND*
- *VSS → GND, VDD → +3.3 V (o +5 V)*
- *V0 → cursor de potenciómetro 10 kΩ (extremos a VDD y GND)*

- *Backlight: A(LED+) → VDD (con resistencia si el módulo no la integra), K(LED−) → GND*

Pulsadores (con INPUT_PULLUP en firmware, entre pin y GND+R=100Ω para proteccion)

- *MENU → PA12 (D2) ↔ GND (100Ω)*
- *UP → PA0 (A0) ↔ GND (100Ω)*
- *DOWN → PA1 (A1) ↔ GND (100Ω)*

LED de usuario

- *LD3 integrado en PB3 (D13).*
- *Si usa LED externo: PB3 (D13) → LED → GND (R 330–1 kΩ).*

Alimentación y referencias

- *VDD, VDDA → +3.3 V (con desacople de 100 nF)*
- *VSS → GND*
- *BOOT0 → GND (10 kΩ)*
- *NRST con pull-up 10 kΩ a 3.3 V (opcional)*

Anexo II: Manual de Usuario.

Este anexo describe cómo operar el sistema en la simulación Wokwi con placa STM32 Nucleo-L031K6, pantalla LCD 16×2 y tres pulsadores: MENU, UP y DOWN. Incluye el significado de cada pantalla, la función de los botones y recomendaciones de uso.

1) Elementos de la interfaz

LCD 16×2 (HD44780, modo 4 bits).

Mensajes posibles:

- Menu principal
- Info del sistema
- Config: <valor> (muestra el valor configurable)

Pulsadores (con resistencia pull-up interna, activos en nivel bajo):

- MENU (navegación principal)
- UP (incrementa)
- DOWN (decrementa)

LED de usuario (D13).

Parpadea de manera periódica para indicar que el programa está en ejecución.

Potenciómetro de contraste (simulado).

Ajusta la visibilidad de los caracteres del LCD.

2) Puesta en marcha (simulación Wokwi)

1. Abrir el proyecto en Wokwi y pulsar “Play”.

2. Si el LCD no se lee con claridad, girar el potenciómetro hasta obtener el contraste adecuado.

3. Verá inicialmente la pantalla `Menu principal`.

3) Funcionamiento del botón MENU

El botón MENU permite navegar entre pantallas mediante tres patrones de pulsación:

- Pulsación corta (duración < 800 ms):

Pasa a `Menu principal`.

Nota: al soltar el botón, el sistema espera hasta 350 ms (ventana de doble clic) antes de confirmar el clic simple. Esta espera es deliberada.

- Doble pulsación (dos pulsaciones cortas separadas por < 350 ms):

Pasa a `Info del sistema`.

- Pulsación larga (duración ≥ 800 ms):

Entra a `Config: <valor>` (modo Configuración).

Observación: El sistema aplica antirrebote por software (~30 ms). Por ello, la acción de un clic corto puede percibirse tras la pequeña espera indicada.

4) Modo Configuración: ajuste con UP/DOWN

El ajuste del parámetro sólo está habilitado cuando el LCD muestra `Config: <valor>`.

- UP: incrementa el valor en 1 unidad.
- DOWN: decrementa el valor en 1 unidad.

Límites del valor configurable:

- Mínimo: 0
- Máximo: 99

Si se alcanza un extremo, el valor no sobrepasa ese límite.

Cada pulsación UP/DOWN se reconoce en el flanco de bajada del botón y, tras el antirrebote (~30 ms), se actualiza el LCD con el nuevo valor.

5) Resumen de navegación

Desde cualquier pantalla:

- MENU (larga ≥ 800 ms) → `Config: <valor>`

En `Config: <valor>`:

- UP → incrementa el valor
- DOWN → decrementa el valor

Con MENU (doble) → `Info del sistema`

Con MENU (corta) → `Menu principal` (tras la ventana de 350 ms)

6) Recomendaciones de uso

Ritmo de pulsación: para una doble pulsación, realice dos clics cortos con un intervalo inferior a 350 ms. Si el intervalo es mayor, el sistema interpretará dos clics simples separados.

Pulsación larga: mantenga presionado el botón al menos 800 ms y suelte para ingresar a Configuración.

Antirrebote: tras cada pulsación puede existir una pequeña latencia (≈30 ms) antes de que el LCD se actualice; es parte del filtrado por software.

Contraste del LCD: si no se distinguen caracteres, ajuste el potenciómetro del contraste.

7) Resolución de problemas comunes

No se ve texto en el LCD: ajustar el contraste; comprobar alimentación del LCD en la simulación.

Doble pulsación no reconocida: reducir el intervalo entre los dos clics (debe ser < 350 ms).

UP/DOWN no modifican el valor: verifique estar en la pantalla `Config: <valor>`; fuera de ese modo los botones no tienen efecto.

Cambios “dobles” con una sola pulsación: aumentar ligeramente el tiempo de antirrebote (por ejemplo, de 30 ms a 40 ms) o cuidar que la pulsación sea nítida en el simulador.

8) Notas técnicas (referencia rápida)

Temporizaciones principales:

- Antirrebote: ~30 ms
- Ventana doble clic: 350 ms
- Umbral pulsación larga: 800 ms

Operación del LCD (4 bits):

- RS (D11) indica comando (0) o dato (1).
- E (D12) para la captura de nibbles; cada carácter genera dos pulsos en E.

Con estas indicaciones el usuario puede navegar por las pantallas y ajustar el valor configurado de manera simple y consistente con el comportamiento implementado en el firmware de la simulación.

Anexo III: Diagrama de Flujo del Programa.

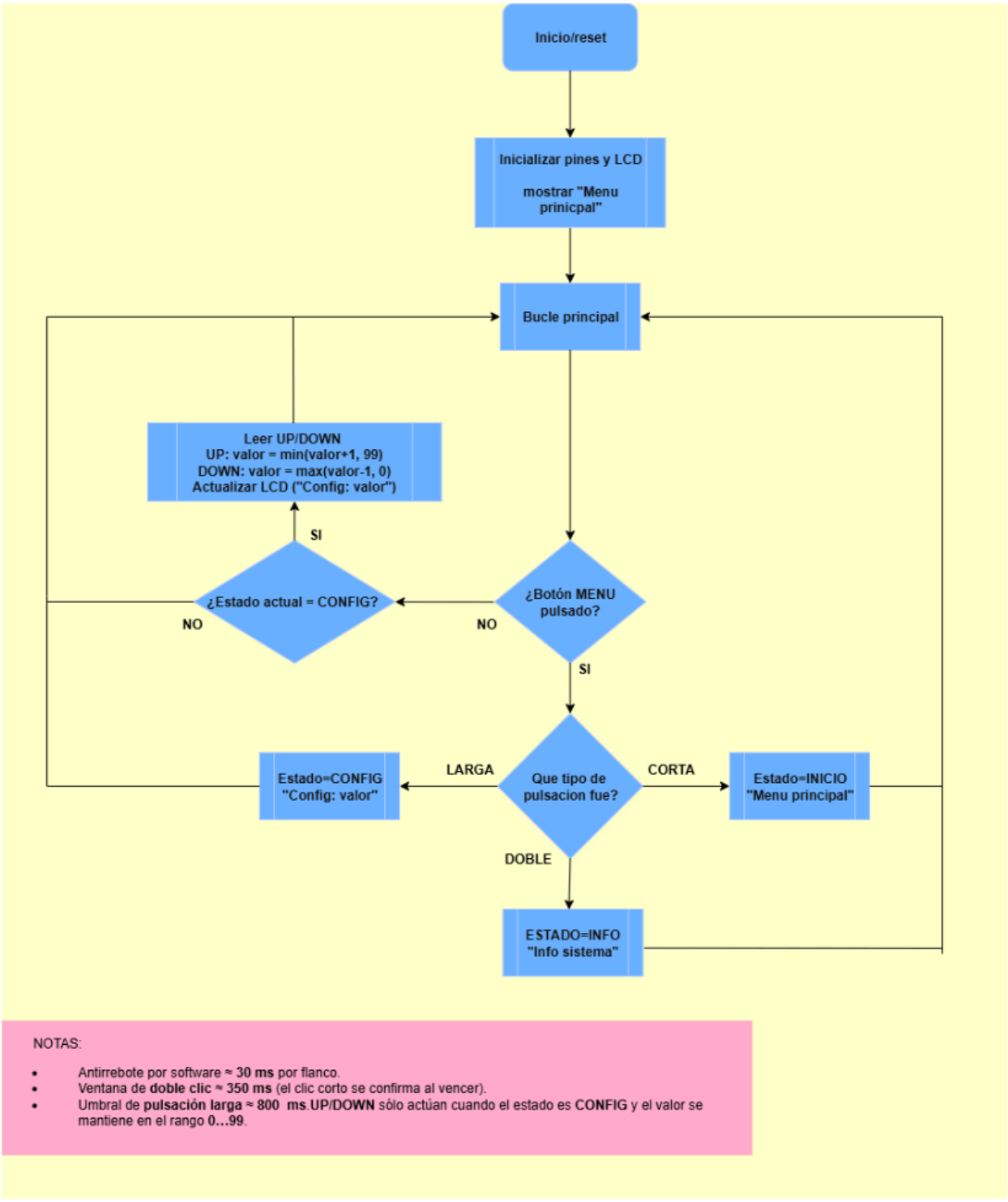

Figura 8-3: Diagrama de flujo del sistema.

Anexo IV: Capturas de Pantalla y Mediciones.

Colección de imágenes demostrativas del funcionamiento:

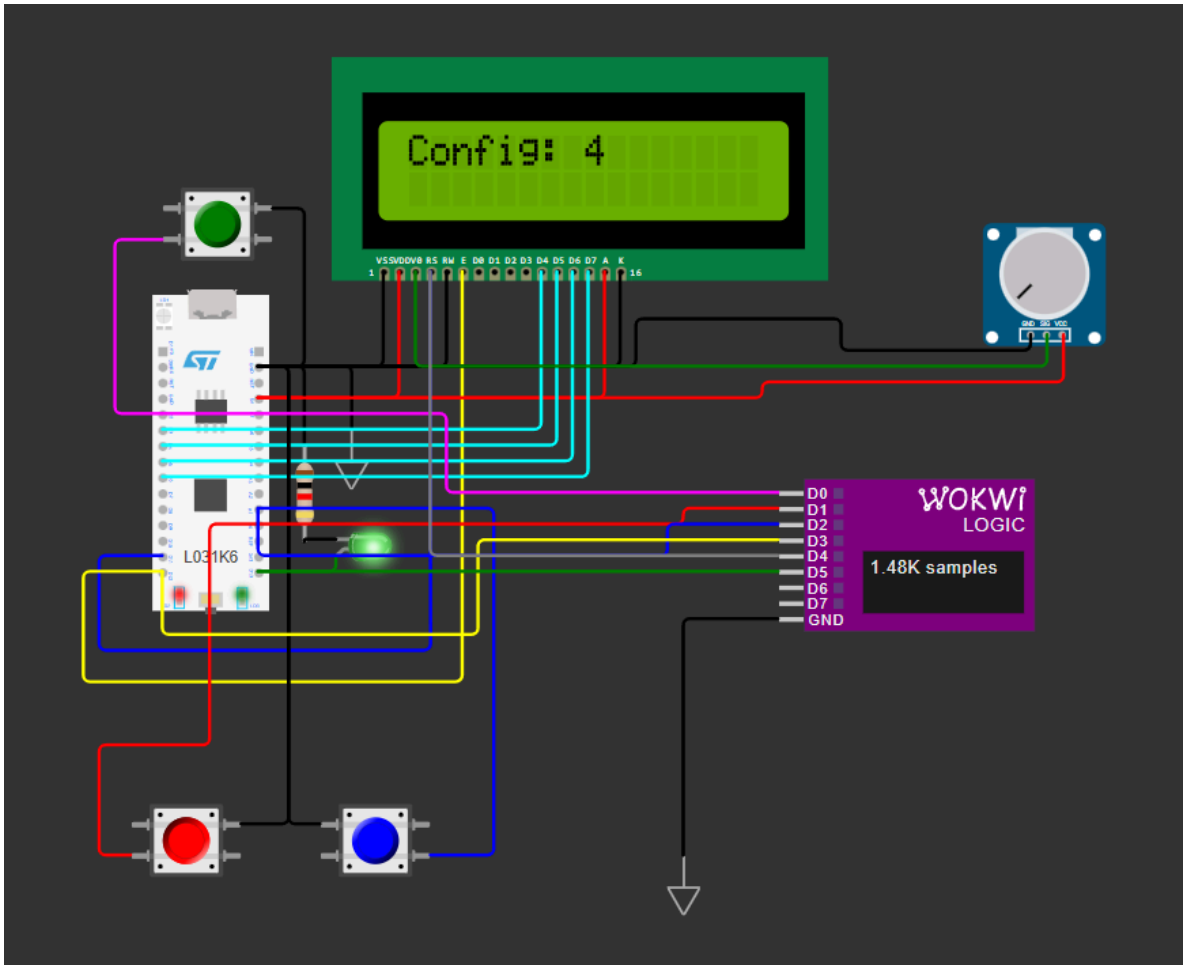

Figura 8-4: Captura de la simulación Wokwi con el LCD mostrando el estado "Configuración" y valor numérico, mientras se indica un botón siendo presionado (ROJO).