LAB 1 - Introducción al ESP32-C6

Sistemes Encastats i Ubics

Estudiantes: Zixin Zhang & Tomeu Uris Tortella
Facultad: FIB - UPC
Curso: 2025-2026

Comentario previo

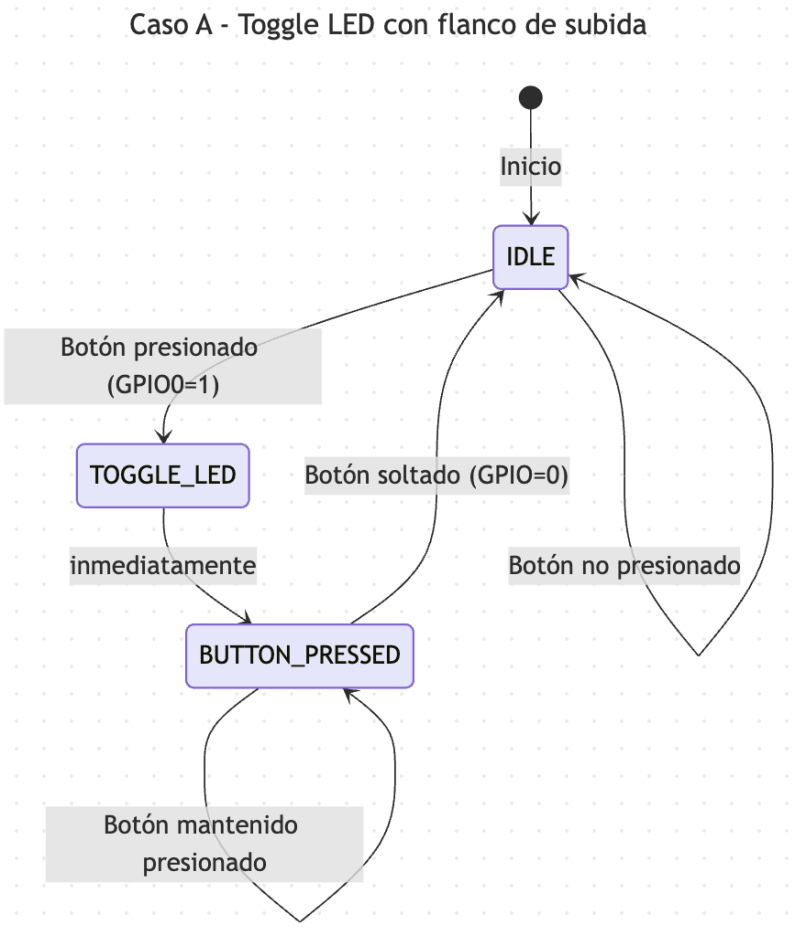
Los códigos para los diferentes apartados están en la carpeta *main*, cada uno con la letra del apartado al que corresponde. Para la ejecución del ejercicio específico, es necesario cambiar en el *CMakeLists.txt*, que se encuentra también en la carpeta *main*, a que archivo hace referencia.

Ejercicios

A) Hemos usado una estrategia de “pooling” para detectar cambios en el input (el botón) por parte del usuario. En cada iteración comprobamos el estado del pin de entrada si ha ocurrido un borde ascendente. Para ello guardamos el estado de la iteración y al comparar, si el anterior era 0 (no pulsado) y el actual es 1 (pulsado) cambiamos el estado del LED.

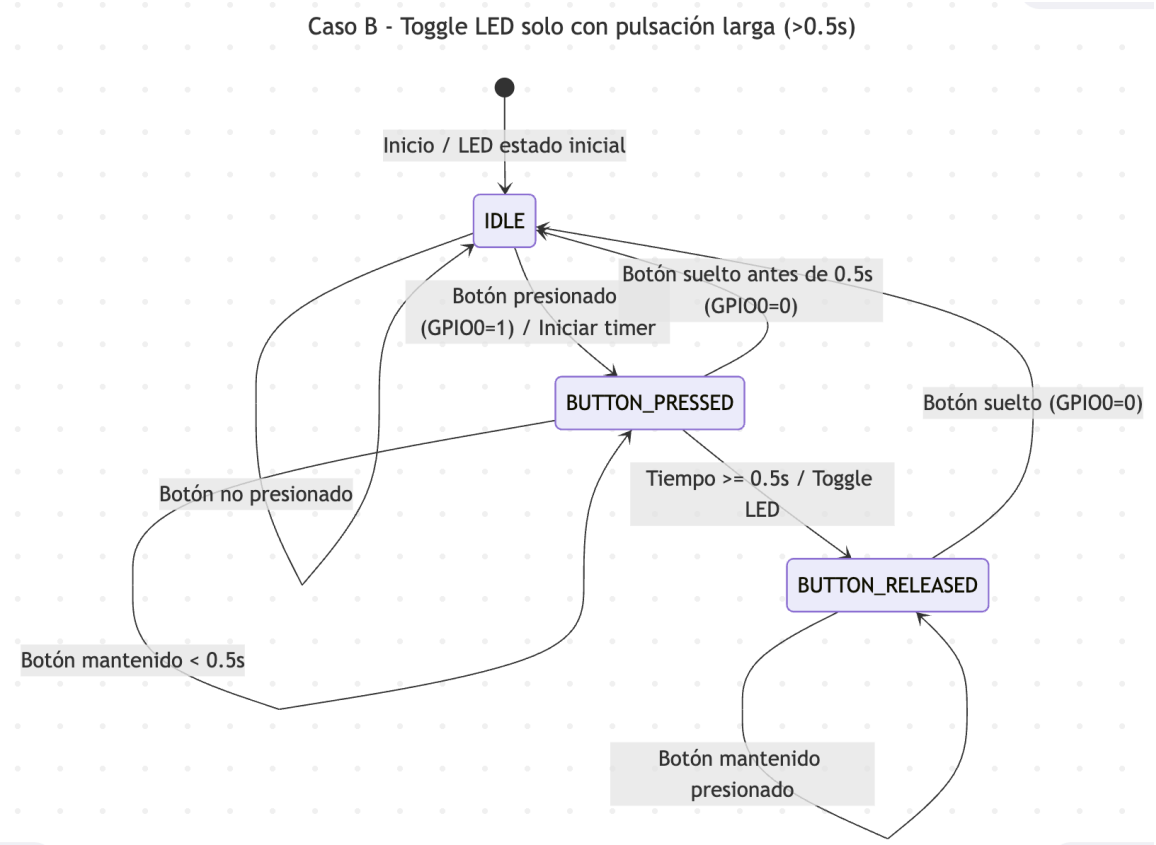
El código no está plenamente hecho como una máquina de estados ya que el problema es muy sencillo. De todas maneras, el diagrama siguiente se comporta de la misma manera que el código.

- El sistema empieza en estado **IDLE** que significa que el botón no ha sido pulsado aún.
- El estado **TOGGLE_LED** solo cambia el estado del LED y cambia inmediatamente al estado **BUTTON_PRESSED**.
- El estado **BUTTON_PRESSED** comprueba que se suelte el botón (para la detección de bordes ascendentes).



B) Para este ejercicio hemos seguido en la línea del anterior. Seguimos con la estrategia de “pooling” comprobando por bordes ascendentes y hemos añadido un sistema de estados para detección de pulsada larga (que sustituye el guardado del estado anterior).

- El sistema empieza en estado **IDLE** que significa que el botón no ha sido pulsado aún.
  - i) Si se detecta un borde ascendente cambiamos al estado **BUTTON_PRESSED** y se guarda el tick count para ese momento el cual usaremos para contabilizar el tiempo transcurrido de la pulsación.
  - ii) Si no se detecta un borde ascendente se continua en el estado **IDLE**.
- Si el sistema está en el estado **BUTTON_PRESSED** significa que el botón ha pasado a estar pulsado.
  - i) Si se detecta que se ha soltado el botón, se vuelve al estado **IDLE**.
  - ii) Si se detecta que ha transcurrido el tiempo suficiente para que se considere una pulsación larga (se habrá definido el tiempo en el programa en milisegundos), se cambia el estado del led y se cambia el sistema a estado **BUTTON_RELEASED**.
- Si el sistema está en el estado **BUTTON_RELEASED** significa que si ha transcurrido el tiempo suficiente para que se considere pulsación larga, pero el usuario aún no ha soltado el botón. Esto consigue evitar que el LED cambie de estado continuamente si el usuario mantiene el botón pulsado.
  - i) Si se detecta que se ha soltado el botón se cambia al estado **IDLE** para poder empezar a detectar una nueva pulsación.
  - ii) Si se detecta que aún no se ha soltado, se mantiene en este estado.



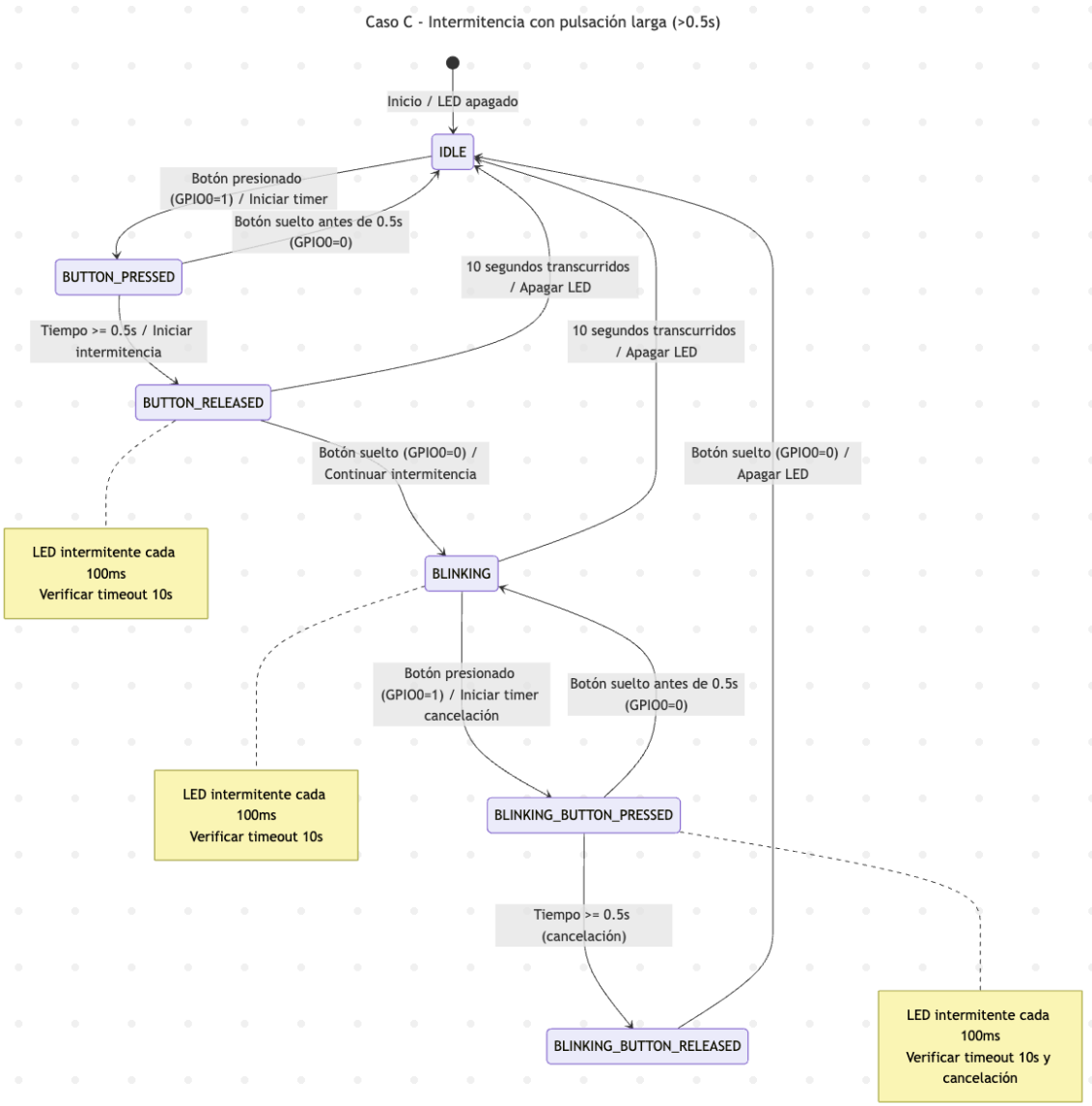
C) Para este ejercicio hemos evolucionado la máquina de estados del caso anterior hacia un **sistema de estados con gestión temporal múltiple**. Mantenemos la estrategia de "polling" y detección de pulsaciones largas, pero añadimos **estados de proceso activo** para ejecutar la intermitencia del LED mientras simultáneamente monitorizan nuevos eventos de entrada.

Estados implementados:

- El sistema empieza en estado **IDLE**: Estado de reposo, LED apagado, esperando detección de pulsación inicial.
    - Si se detecta una pulsación (borde ascendente) se “inicia el temporizador” para la detección de pulsación larga y se cambia al estado **BUTTON_PRESSED**.
    - En caso contrario, se mantiene en este estado.
- Si el sistema está en el estado **BUTTON_PRESSED**.
    - Si se ha soltado el botón se vuelve al estado **IDLE**.
    - En caso contrario, se valida el tiempo transcurrido de pulsación comprobando si se ha llegado al límite definido en el código en milisegundos. Si es así se “inician los temporizadores” para el LED (uno para controlar la intermitencia y el otro para su duración), y se cambia el estado a **BUTTON_RELEASED**.
- Si el sistema está en el estado **BUTTON_RELEASED**, significa que se ha iniciado/parado la intermitencia del LED en esta pulsación pero el usuario aún no ha soltado.
    - En cualquier caso, si la duración de la intermitencia supera el tiempo definido (con el temporizador de duración). Se para y se cambia el estado a **IDLE**. Esto genera un problema, si el usuario mantiene pulsado el botón más de 10 segundos, automáticamente se inicia un nuevo ciclo al instante de acabar el anterior.
    - Si no, si se ha soltado el botón, se cambia al estado **STATE_BLINKING**.
    - En caso contrario, se continúa con la intermitencia. Para ello, se comprueba el si hay que cambiar el estado del led mediante la duración transcurrida desde el último cambio (con el temporizador de intermitencia). Y no se cambia de estado.
- Si el sistema está en estado **BLINKING** significa que el proceso de intermitencia está activo con botón liberado. Continúa el parpadeo, monitoriza nuevas pulsaciones y controla timeout.
    - En cualquier caso, si la duración de la intermitencia supera el tiempo definido (con el temporizador de duración), se cambia el estado a **IDLE** (efectivamente parando la intermitencia).
    - Si se detecta una pulsación, se inicia el temporizador de pulsación y se cambia al estado **BLINKING_BUTTON_PRESSED**.
- Si el sistema está en estado **BLINKING_BUTTON_PRESSED** significa que se está comprobando la detección de posible cancelación temprana durante la intermitencia.
    - Si se suelta el botón se vuelve al estado **BLINKING** ya que significa que se ha soltado antes de haber cumplido con el tiempo necesario para la cancelación temprana.
    - Si se detecta que la intermitencia supera el tiempo definido se cambia al estado **IDLE**.

- Si se detecta que la pulsación ha durado más del límite de tiempo, provocando una cancelación temprana, se pasa al estado **BLINKING_BUTTON_RELEASED** (efectivamente parando la intermitencia).
- En última instancia, se continúa con la intermitencia.

- Si el sistema está en estado **BLINKING_BUTTON_RELEASED** significa que el usuario ha provocado exitosamente una parada temprana de la intermitencia, pero aún no ha soltado el botón. Este estado evita que se lanze un ciclo de intermitencia inmediatamente después de cancelar la anterior.
  - Si el usuario suelta el botón, se cambia al estado **IDLE** permitiendo así la comprobación para un nuevo ciclo.
  - Mientras el usuario lo mantenga presionado se seguirá en este estado.



MEJORA C:
Se ha corregido el error que provocaba que seguir pulsando el botón encadenara ciclos indefinidamente. Para ello se ha renombrado el estado **BLINKING_BUTTON_RELEASED** a **BLINKING_ENDED_BUTTON_RELEASED** y el estado **BUTTON_RELEASED** cambia a este en lugar de a **IDLE** cuando ha transcurrido el tiempo máximo de intermitencia. Esto fuerza al usuario a soltar el botón y volver a pulsarlo de nuevo para iniciar un nuevo ciclo. También se ha limpiado un poco el código.

Hemos decidido mantenerlo como un C mejorado (C_improved.c) ya que no hemos podido probarlo con el hardware. De todas maneras se ha probado con el simulador Wokwi integrado en VS Code.