

Informe de la Pràctica 1: Blink

1. Introducció

L'objectiu principal d'aquesta pràctica és comprendre el funcionament bàsic de la programació d'un microcontrolador ESP32 mitjançant la creació d'un programa senzill que faci parpellejar un LED. A més, s'introduceix l'ús del port sèrie per depurar el programa i verificar-ne el correcte funcionament.

Aquesta pràctica representa el primer pas en la programació de sistemes integrats, permetent familiaritzar-se amb la configuració de pins com a sortides digitals i la utilització de funcions bàsiques de retard en el temps.

2. Fonaments Teòrics

Els microcontroladors com l'ESP32 tenen pins digitals que es poden configurar com a entrades o sortides. En aquest cas, configarem un pin com a sortida per controlar un LED.

Els conceptes clau que s'utilitzen en aquesta pràctica són:

- **Pins GPIO (General Purpose Input/Output)**: són les entrades i sortides digitals del microcontrolador.
- **Funció `pinMode(pin, mode)`**: configura un pin determinat com a entrada (`INPUT`) o sortida (`OUTPUT`).
- **Funció `digitalWrite(pin, value)`**: estableix l'estat d'un pin digital (`HIGH` per encendre, `LOW` per apagar).
- **Funció `delay(ms)`**: introduceix una pausa en l'execució del programa durant el temps especificat en mil·lisegons.
- **Port sèrie**: permet enviar informació al monitor sèrie per comprovar l'estat del programa mitjançant `Serial.begin(baudrate)` i `Serial.println(missatge)`.

El codi implementat segueix un esquema cíclic, on el LED s'encén, es manté encès durant un període determinat, s'apaga i es torna a repetir el procés.

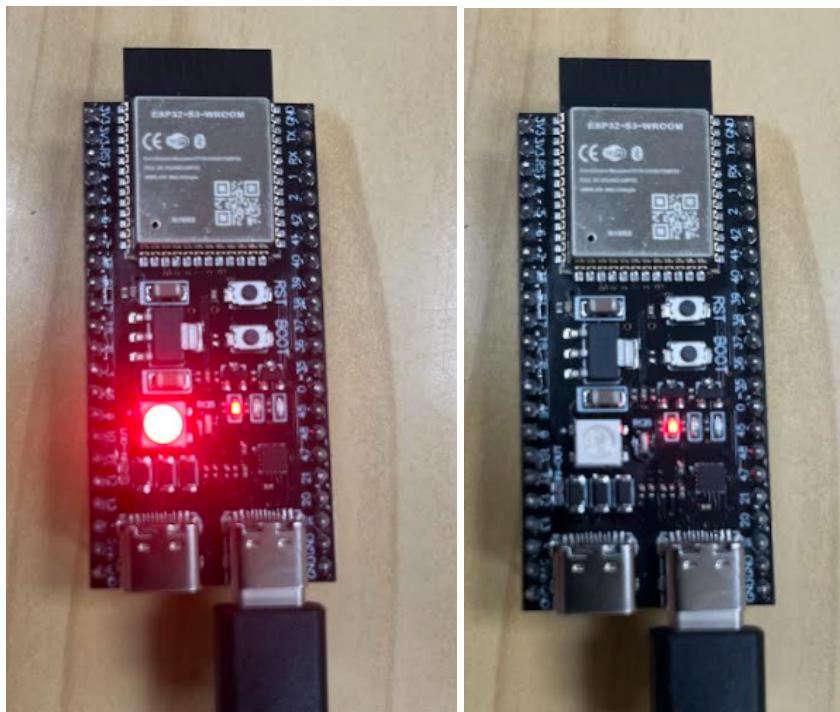
3. Treball i preguntes

3.1 Parpelleig del LED

Quan el programa s'executa a l'ESP32, el LED integrat en el pin predeterminat comença a parpellejar amb un interval de temps definit. Aquest efecte es produeix mitjançant les funcions digitalWrite() per encendre i apagar el LED i delay() per mantenir-lo en cada estat durant un temps específic.

En el codi, el LED s'encén i es manté encès durant 500 mil·lisegons. Després, el LED s'apaga i es manté apagat durant el mateix període. Aquest cicle es repeteix indefinidament dins de la funció loop().

L'efecte es pot observar físicament en la placa ESP32, com es mostra a les imatges adjuntes, on es veu el LED en estat encès i apagat.

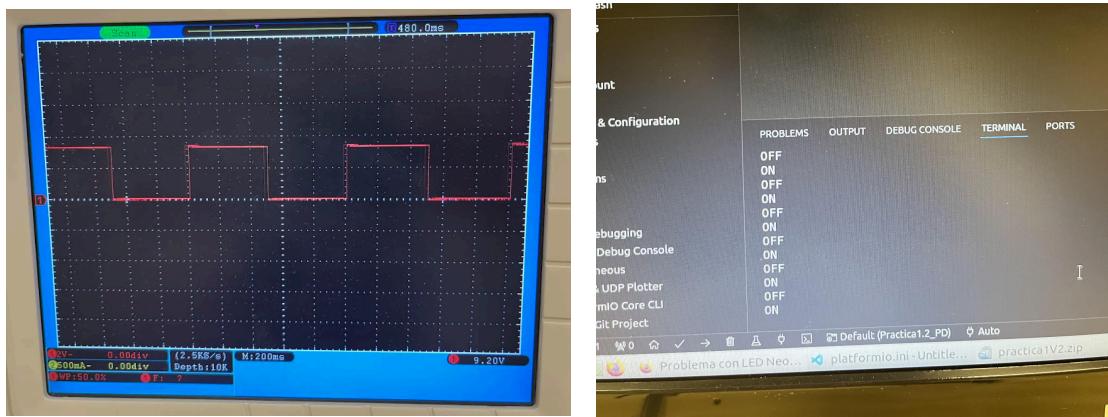


3.2 Enviament de l'estat del LED pel port sèrie

Per obtenir el resultat d'aquest punt de la pràctica, s'ha modificat el programa original perquè, a més de fer parpellejar el LED, envii un missatge pel port sèrie cada vegada que el seu estat canvia.

Primer, s'ha inicialitzat el port sèrie amb `Serial.begin(115200)`, permetent la comunicació entre l'ESP32 i el monitor sèrie de l'ordinador. A continuació, dins del bucle infinit, cada vegada que el LED s'encén, s'envia el missatge "ON" pel port sèrie, i quan s'apaga, s'envia "OFF".

El temps d'encès i apagat del LED s'ha ajustat a 1000 mil·lisegons, per poder visualitzar clarament tant el canvi de llum com el registre dels missatges en el monitor sèrie. Aquest procés permet verificar que el programa funciona correctament i que el LED realitza el parpelleig esperat.



3.3 Control del LED mitjançant els registres de ports d'entrada i sortida

En aquesta part de la pràctica, s'ha modificat el programa perquè el control del LED es faci directament manipulant els registres de l'ESP32 en lloc d'utilitzar les funcions d'Arduino com `digitalWrite()`.

Aquesta tècnica permet una execució més ràpida, ja que escriure directament als registres de maquinari elimina la sobrecàrrega de les funcions d'alt nivell. En aquest cas, s'ha definit un punter al registre corresponent i s'han utilitzat operacions binàries per establir o canviar l'estat del pin del LED.

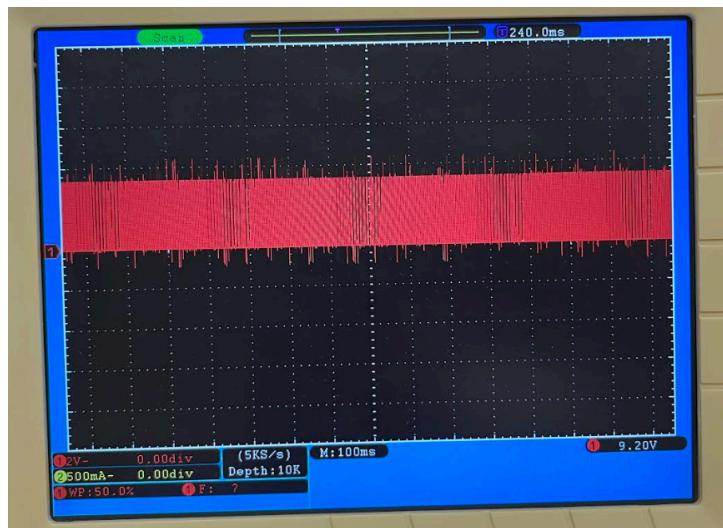
Aquesta modificació optimitza el rendiment del sistema, fent que el canvi d'estat del LED sigui més ràpid i eficient.

3.4 Mesura de la freqüència màxima de commutació del microcontrolador

En aquesta part de la pràctica, s'ha eliminat la funció `delay()` per permetre que el LED canviï d'estat el més ràpidament possible. A més, s'ha modificat el programa perquè el pin de sortida es pugui seleccionar lliurement entre els disponibles a l'ESP32.

Per analitzar el rendiment del microcontrolador, s'ha utilitzat un oscil·oscopi per mesurar la freqüència màxima d'encesa i apagat del LED en quatre escenaris diferents:

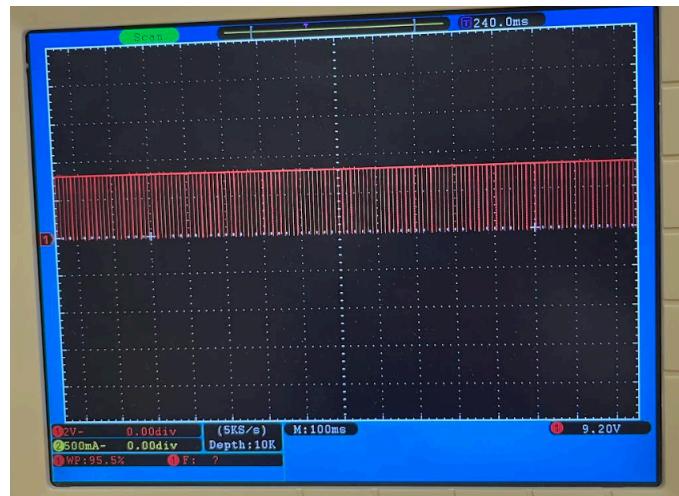
3.4.1. Amb enviament de missatges pel port sèrie i utilitzant les funcions d'Arduino.



-L'oscil·oscopi mostra una commutació molt lenta a causa del temps addicional necessari per enviar els missatges pel port sèrie.

-L'alta densitat de dades en el senyal és conseqüència de l'impacte del retard introduït per la comunicació sèrie.

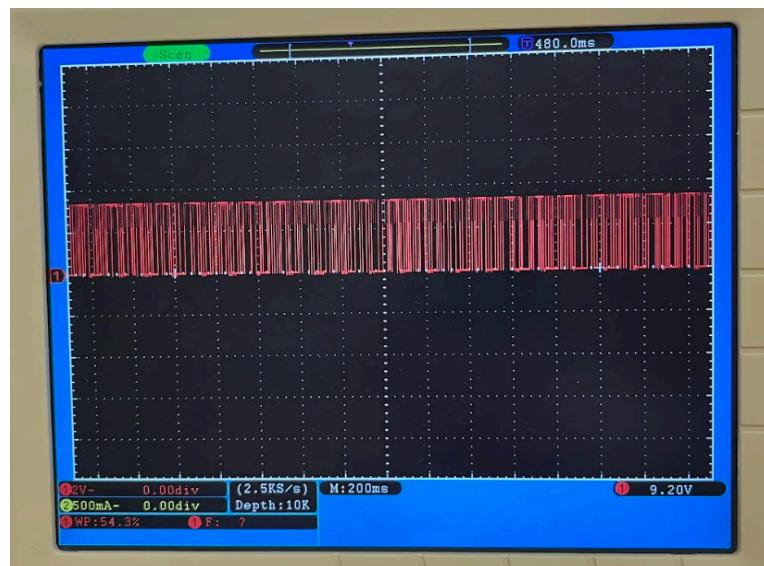
3.4.2. Amb enviament de missatges pel port sèrie i accedint directament als registres.



-La freqüència de commutació millora notablement respecte a la primera imatge, perquè l'accés directe als registres és més eficient que digitalWrite().

-Tot i així, l'enviament de dades pel port sèrie encara introduceix una limitació.

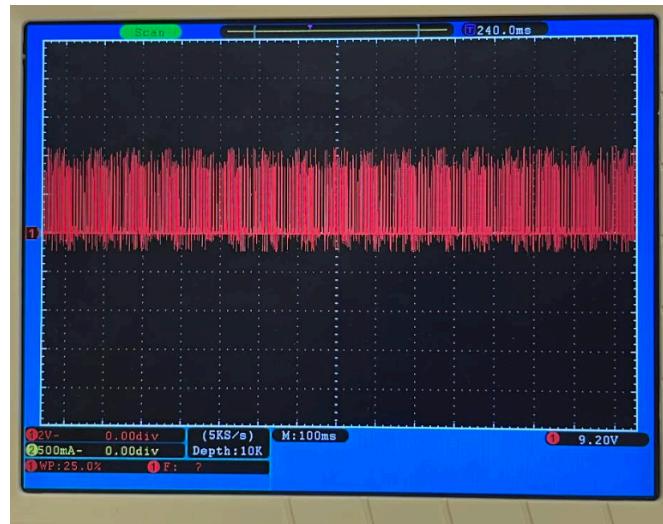
3.4.3. Sense enviament de missatges pel port sèrie i utilitzant les funcions d'Arduino.



La freqüència de commutació augmenta considerablement respecte als casos anterioris, ja que el sistema ja no ha d'esperar la transmissió sèrie.

No obstant això, digitalWrite() continua introduint un retard considerable respecte a l'ús directe dels registres.

3.4.4. Sense enviament de missatges pel port sèrie i accedint directament als registres.



-Es mesura la freqüència màxima de parpelleig sense factors externs que la limitin, obtenint una commutació molt ràpida.

-L'accés directe als registres GPIO evita els retards causats per Serial.println() i digitalWrite(), permetent un senyal més estable i eficient.

Conclusió 3.4

Aquest experiment permet determinar com afecta la comunicació sèrie i el mètode de control del pin a la velocitat màxima d'execució del microcontrolador. Com era d'esperar, l'accés directe als registres proporciona la major velocitat de commutació, mentre que l'enviament de dades pel port sèrie introduceix un retard significatiu en la freqüència d'encesa i apagat del LED.

4. Diagrama de flux i diagrama de temps

Diagrama de flux:

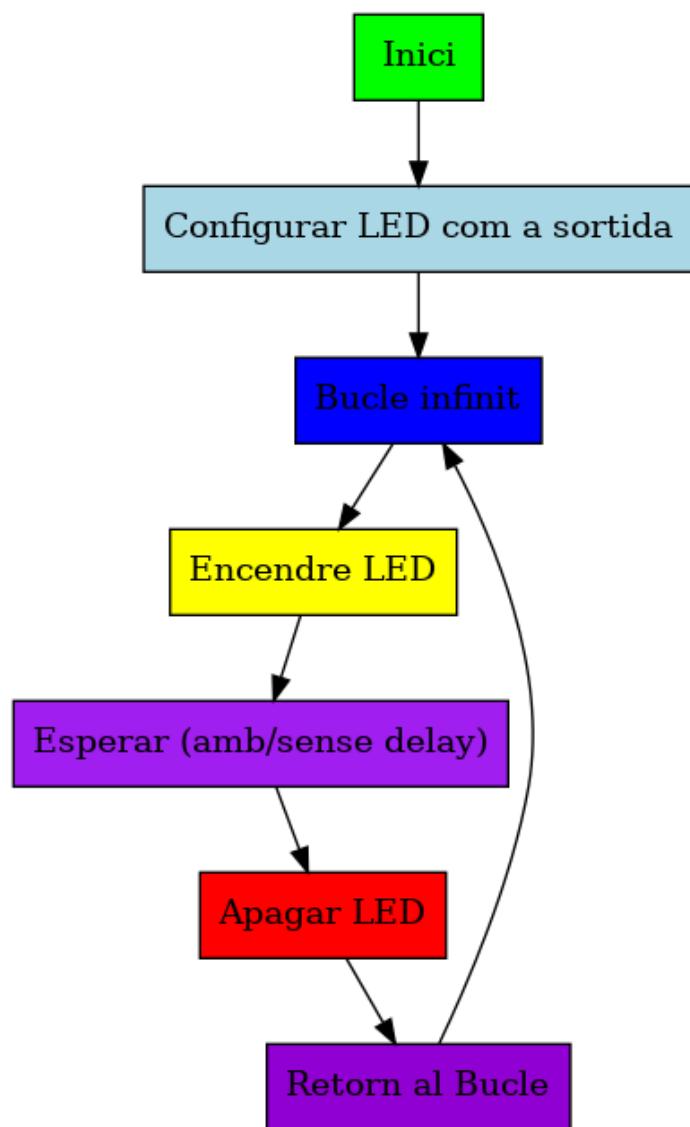
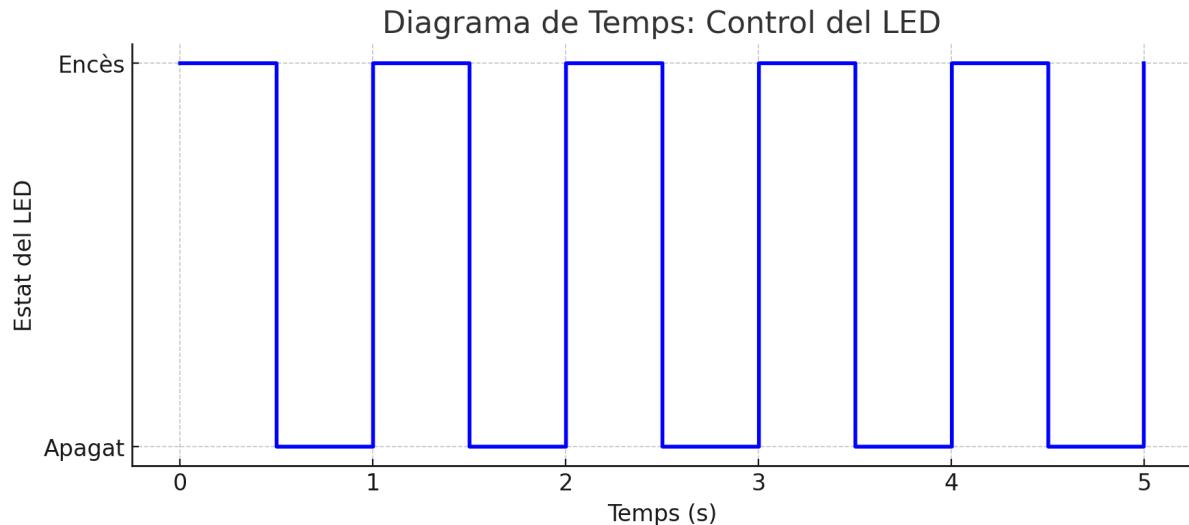


Diagrama de temps:



5. ¿Cuál es el tiempo libre que tiene el procesador ?

El processador ESP32 queda inactiu durant els retards (`delay(1000)`), cosa que significa que no està realitzant cap altra tasca en aquests moments. Com que el microcontrolador no

més està esperant que passi el temps, podem dir que el seu temps lliure és molt alt, pràcticament del 100% en aquest codi bàsic.

Si es volgués aprofitar aquest temps per fer altres tasques, una alternativa seria utilitzar interrupcions o temporitzadors en lloc de `delay()`. Això permetria que el processador executés altres instruccions mentre espera el canvi d'estat del LED, optimitzant millor els recursos disponibles.

6. Conclusions

La pràctica del Blink ha permès comprendre el funcionament bàsic de l'ESP32 mitjançant la programació d'un LED intermitent. S'han aplicat conceptes fonamentals com la configuració de pins GPIO, l'ús de funcions bàsiques com `'digitalWrite()'` i `'delay()'`, així com l'enviament de dades pel port sèrie per monitoritzar l'execució del programa. A més, s'ha explorat el control directe del LED mitjançant registres per millorar l'eficiència. L'anàlisi de la freqüència de commutació ha demostrat l'impacte del mètode de control i la comunicació sèrie en el rendiment del microcontrolador. Finalment, s'ha evidenciat que el temps lliure del processador és elevat en aquesta implementació, suggerint alternatives com l'ús d'interrupcions per optimitzar-ne l'ús.

