

Informe Pràctica 1: Blink

Introducció

L'objectiu principal de la pràctica ha estat produir el parpelleig periòdic d'un LED utilitzant un ESP32-S3 i analitzar el seu comportament en diferents escenaris. També s'ha explorat la gestió del port sèrie i l'accés directe als registres del microcontrolador per optimitzar la freqüència d'encesa i apagada.

Descripció de la placa a utilitzar

L'ESP32-S3 és un microcontrolador de la família ESP32 amb capacitats de connectivitat Wi-Fi i Bluetooth. Aquesta versió millora la potència de càlcul i ofereix una major eficiència energètica. En la pràctica, s'ha utilitzat un dels seus GPIOs per controlar un LED i mesurar el senyal resultant amb un oscil·oscopi.



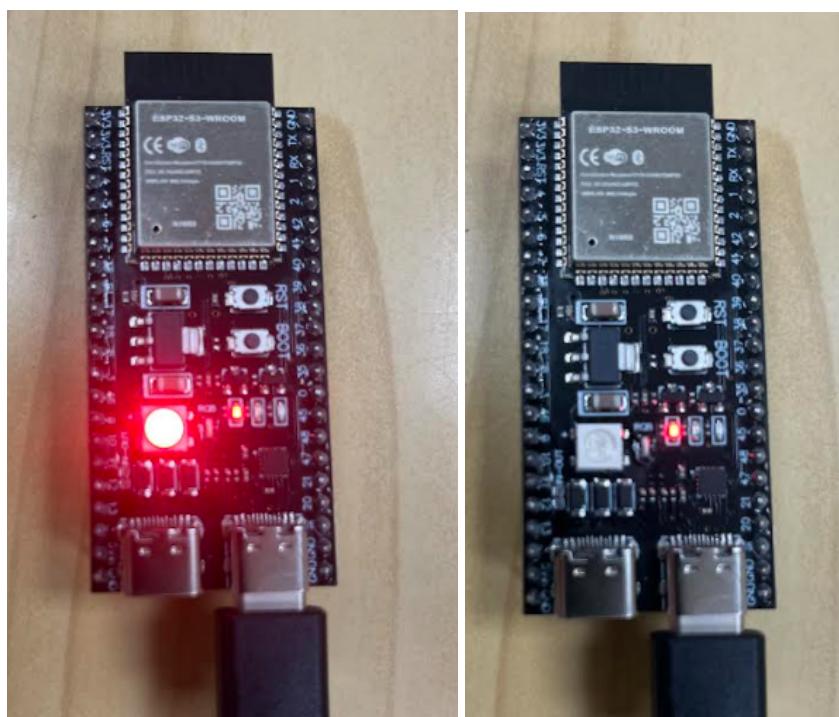
Desenvolupament de la pràctica

Punt 1: Parpelleig bàsic del LED

S'ha iniciat un LED com a sortida i s'ha creat un bucle infinit que alterna el seu estat cada 500 ms. Aquest és el comportament bàsic d'un "blink".

Resultat obtingut:

El LED s'encén i s'apaga de manera periòdica, complint amb els 500 ms de retard establerts.

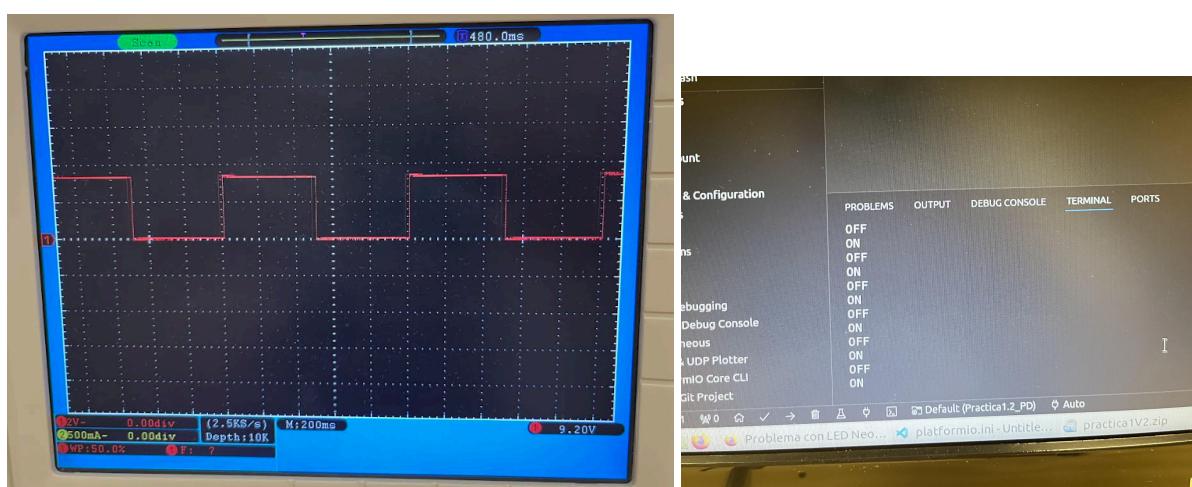


Punt 2: Afegir comunicació pel port sèrie

S'ha modificat el programa per incloure la inicialització del port sèrie i enviar els missatges "ON" i "OFF" cada cop que el LED canvia d'estat.

Resultat obtingut:

El terminal sèrie mostra de manera sincronitzada els missatges corresponents ("ON" quan el LED s'encén i "OFF" quan s'apaga). Això confirma que la comunicació pel port sèrie funciona correctament.



Punt 3: Accés directe als registres del microcontrolador

S'ha modificat el codi perquè el control del LED es realitzi directament sobre els registres de GPIO, en lloc d'utilitzar les funcions de l'API d'Arduino.

Resultat obtingut:

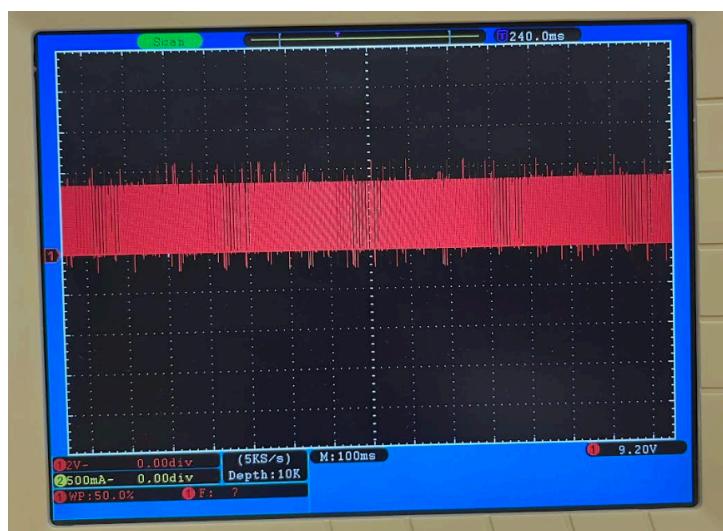
S'ha aconseguit un control més ràpid del LED. En comptes d'utilitzar `digitalWrite()`, la manipulació directa dels registres permet una execució més eficient del codi.

Punt 4: Eliminació dels retards i mesura amb oscil·oscopi

S'han realitzat quatre proves modificant l'accés als registres i la comunicació sèrie per mesurar la freqüència màxima d'encesa i apagada del LED.

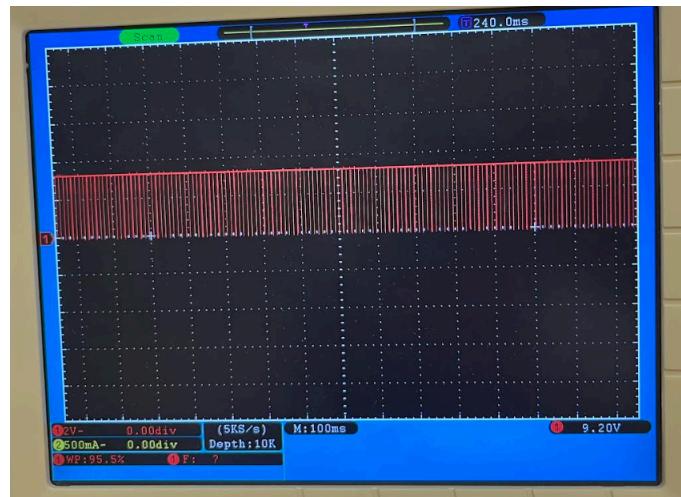
Resultats obtinguts:

- **Amb enviament pel port sèrie i funcions d'Arduino:** La freqüència de parpelleig es veu reduïda per la latència de `Serial.println()`.



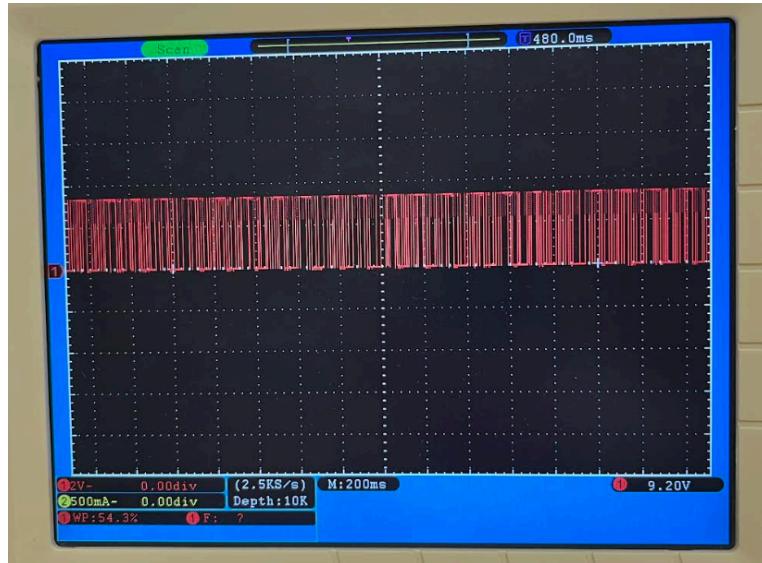
Aquesta imatge confirma que l'ús de `Serial.println()` redueix significativament la freqüència de parpelleig del LED, fent que la seva commutació no sigui òptima per aplicacions que requereixen rapidesa i estabilitat en el control d'un dispositiu.

- **Amb accessos directes als registres i enviament pel port sèrie:** La resposta millora, però encara està limitada pel port sèrie.



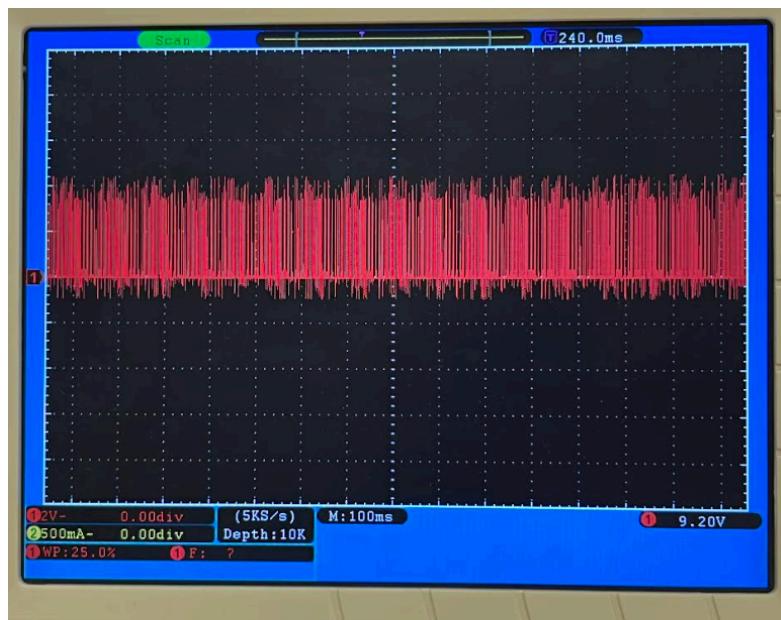
Es pot observar que la freqüència de parpelleig ha augmentat respecte a l'ús de `digitalWrite()`, però encara està limitada per la latència introduïda per `Serial.println()`. Tot i la millora en la resposta, el port sèrie continua afectant la regularitat del senyal, impedint que s'assoleixi la màxima freqüència possible.

-Sense enviament pel port sèrie i usant funcions d'Arduino: La freqüència augmenta, però `digitalWrite()` introduceix cert retard



Aquesta imatge de l'oscil·loscopi mostra la prova "Sense enviament pel port sèrie i usant funcions d'Arduino". La freqüència de parpelleig del LED ha augmentat significativament en comparació amb els casos anteriors, ja que s'ha eliminat la latència del port sèrie. Tot i això, `digitalWrite()` encara introduceix un cert retard, fent que el senyal no sigui tan estable ni tan ràpid com quan s'accedeix directament als registres GPIO.

-Sense enviament pel port sèrie i accedint directament als registres: Es regista la màxima freqüència de parpelleig possible, sense interferències externes.



Es pot observar una alta densitat de commutacions amb una freqüència màxima, ja que no hi ha latències introduïdes per `Serial.println()` ni per `digitalWrite()`. L'accés directe als registres GPIO permet obtenir un senyal net i constant, aconseguint el màxim rendiment del microcontrolador.

Conclusió

Aquesta pràctica ha permès estudiar com la manera de gestionar les entrades i sortides d'un microcontrolador afecta el seu rendiment. L'ús directe dels registres ha demostrat ser més eficient que les funcions d'Arduino, alliberant el processador per altres tasques i permetent assolir una freqüència d'encesa i apagada més alta.

Resposta a la pregunta: Quin és el temps lliure del processador?

El temps lliure del processador depèn del mètode utilitzat per commutar el LED:

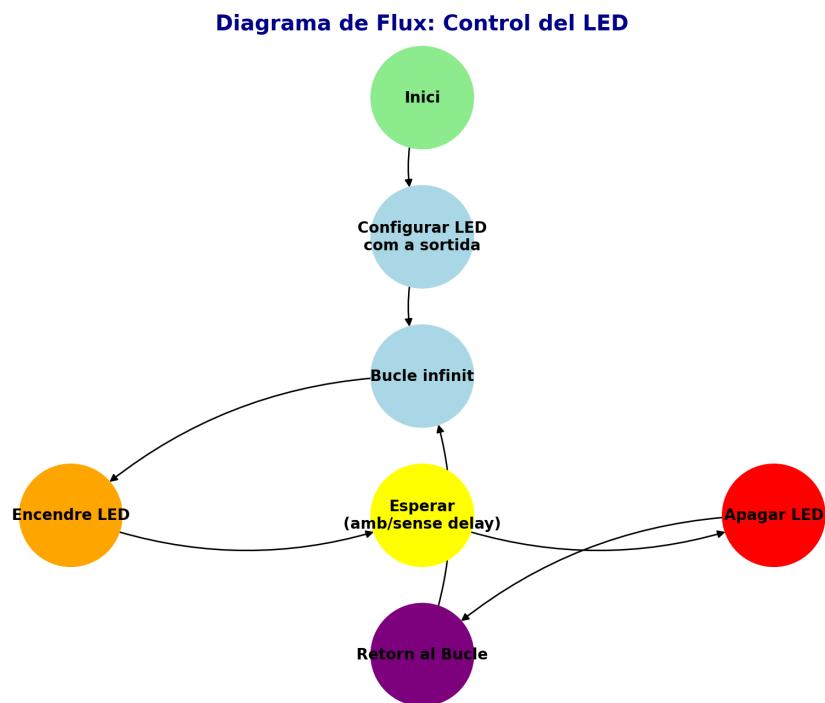
- Amb `Serial.println()` i funcions d'Arduino: La CPU està ocupada la major part del temps gestionant la comunicació sèrie.
- Amb accés directe als registres i sense port sèrie: El processador queda lliure gairebé al 100%, excepte pels cicles necessaris per alternar el bit GPIO.

En el millor dels casos (accés als registres i sense comunicació sèrie), el temps lliure del processador és pràcticament total, ja que l'operació de canvi d'estat és gairebé instantània.

Diagrama de flux i diagrama de temps

A continuació es presenten els diagrames generats:

- **Diagrama de flux:** Representa el funcionament general del control del LED, des de la configuració inicial fins al bucle d'encesa i apagat.



- **Diagrama de temps:** Mostra la variació de l'estat del LED en funció del temps, representant una ona quadrada segons la implementació utilitzada.

