Informe de la Pràctica 1: Blink amb ESP32-S3

Objectiu

L'objectiu d'aquesta pràctica és generar el parpelleig periòdic d'un LED utilitzant diferents mètodes de programació en un ESP32-S3. A més, s'inclourà l'enviament d'informació pel port sèrie i l'accés directe als registres del microcontrolador per avaluar el seu rendiment.

Material Utilitzat

- Microcontrolador **ESP32-S3**
- LED NeoPixel
- Software **PlatformIO**
- Biblioteca **Adafruit NeoPixel**
- Oscil·loscopi per mesurar la freqüència màxima de commutació

Codis Implementats

1. Parpelleig amb NeoPixel i Adafruit Library

Fitxer: ***`codi_1.cpp`***

- Configura el LED com a sortida i el fa parpellejar amb un retard de 500 ms.
- Utilitza la biblioteca Adafruit NeoPixel per gestionar el LED RGB.
- Mostra missatges pel port sèrie indicant l'estat del LED.

Exemple de codi comentat:

```
C/C++

void loop() {

      // Encén el LED en blanc (RGB: 255, 255, 255)

      strip.setPixelColor(0, strip.Color(255, 255, 255));

      strip.show();

      Serial.println("NeoPixel ENCÈS!"); // Missatge al port sèrie

      delay(DELAY_TIME);

      // Apaga el LED

      strip.setPixelColor(0, strip.Color(0, 0, 0));

      strip.show();

      Serial.println("NeoPixel APAGAT!"); // Missatge al port
      sèrie

      delay(DELAY_TIME);

}
```

Aquest codi estableix el color del LED a blanc, el mostra i envia un missatge al port sèrie. Després espera un temps definit per apagar-lo i repetir el procés.

2. Variant amb ajust de brillantor

Fitxer: ***codi_2.cpp***

- Configura el LED i ajusta la brillantor al màxim.
- Modifica el retard a 1000 ms.

3. Control directe dels registres GPIO

Fitxer: *codi_3.cpp*

- Fa servir accessos directes a registres del microcontrolador per encendre i apagar el LED.
- Evita l'ús de biblioteques externes.

4. Implementacions amb `digitalWrite` i registres GPIO

Fitxers: *codi_4_1.cpp*, *codi_4_2.cpp*, *codi_4_3.cpp*, *codi_4_4.cpp*

- Diferents aproximacions per fer parpellejar el LED, utilitzant `digitalWrite` i accessos directes a registres.

DEMOSTRACIÓ DEL QUE SURT PER PANTALLA:

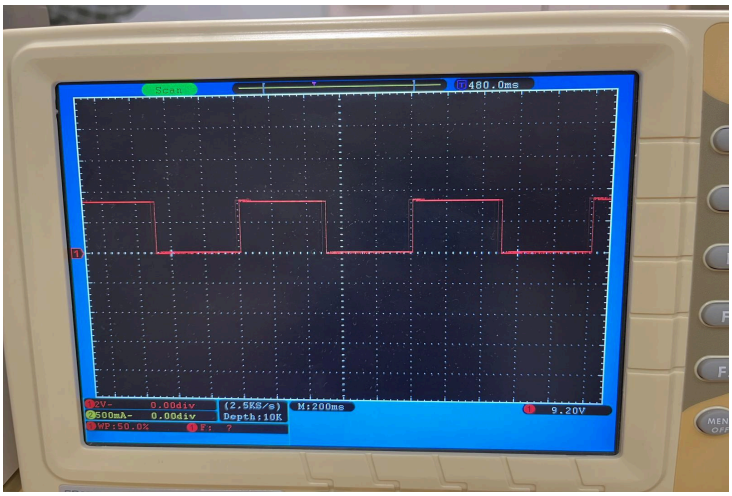
Accés directe als registres del microcontrolador

Per millorar l'eficiència del programa, s'ha modificat el codi perquè actuï directament sobre els registres GPIO del microcontrolador, evitant l'ús de funcions d'alt nivell com **`digitalWrite()`**.

Codi optimitzat (fragment rellevant):

```C/C++
volatile uint32_t *gpio_out = (volatile uint32_t *)GPIO_OUT_REG;
*gpio_out |= (1 << LED_PIN);  // Encendre el LED
*gpio_out &= ~(1 << LED_PIN); // Apagar el LED
```

Aquest mètode redueix la latència i permet obtenir una freqüència màxima més elevada.



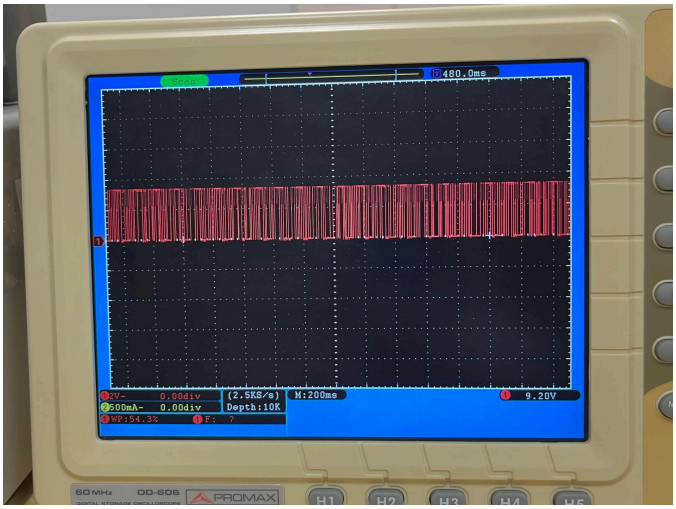
Mesures de Freqüència

S'han realitzat mesuraments per avaluar la freqüència màxima d'encesa i apagada del LED en diferents escenaris:

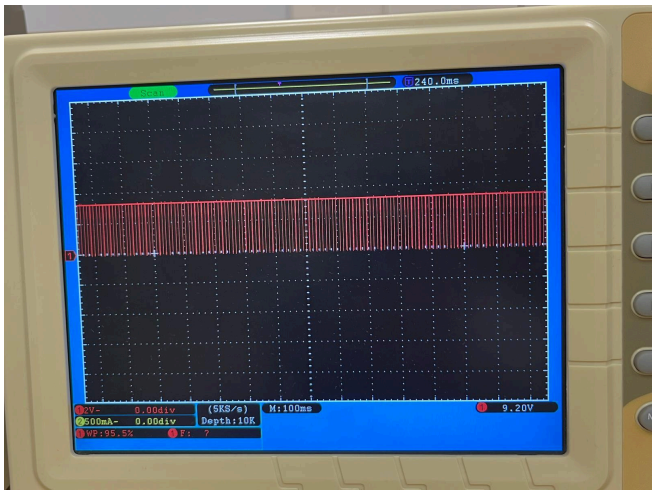
1. **Amb enviament de dades al port sèrie i funcions d'Arduino:** Freqüència més baixa degut al temps d'execució de `Serial.println()`.
2. **Amb enviament de dades i accés directe a registres:** Millora la velocitat respecte al cas anterior.
3. **Sense enviament de dades i amb funcions d'Arduino:** La freqüència augmenta ja que s'eliminen les latències de comunicació sèrie.
4. **Sense enviament de dades i accés directe a registres:** S'obté la freqüència més alta possible.

Les mesures amb l'oscil·loscopi han donat com a resultat:

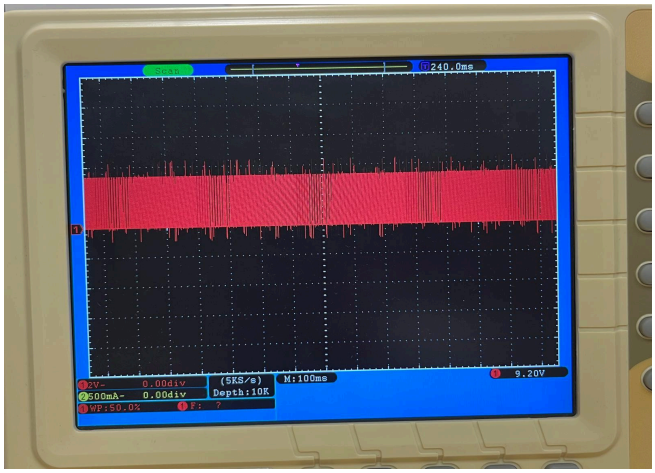
- **Cas1:**



- **Cas 2:**



- **Cas 3:**



- **Cas 4:**

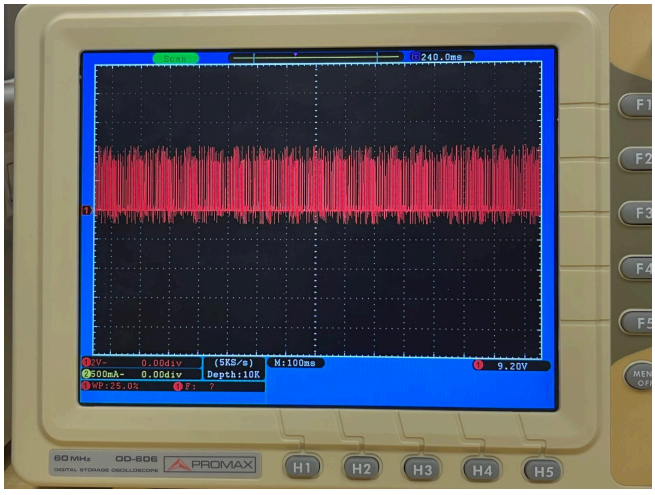


Diagrama de flux del programa

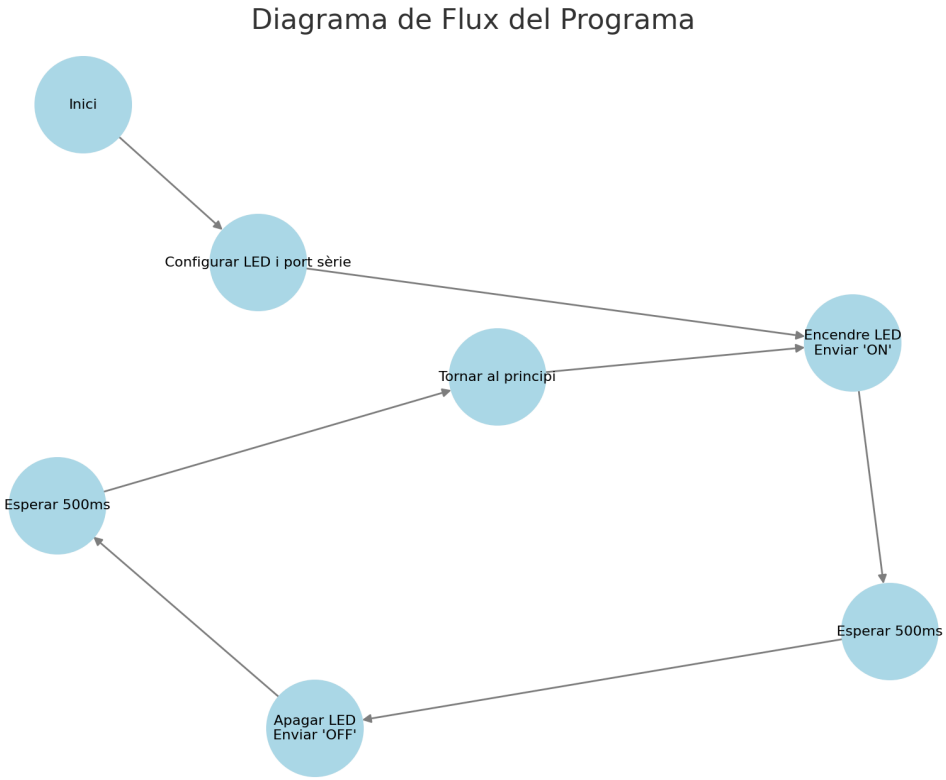
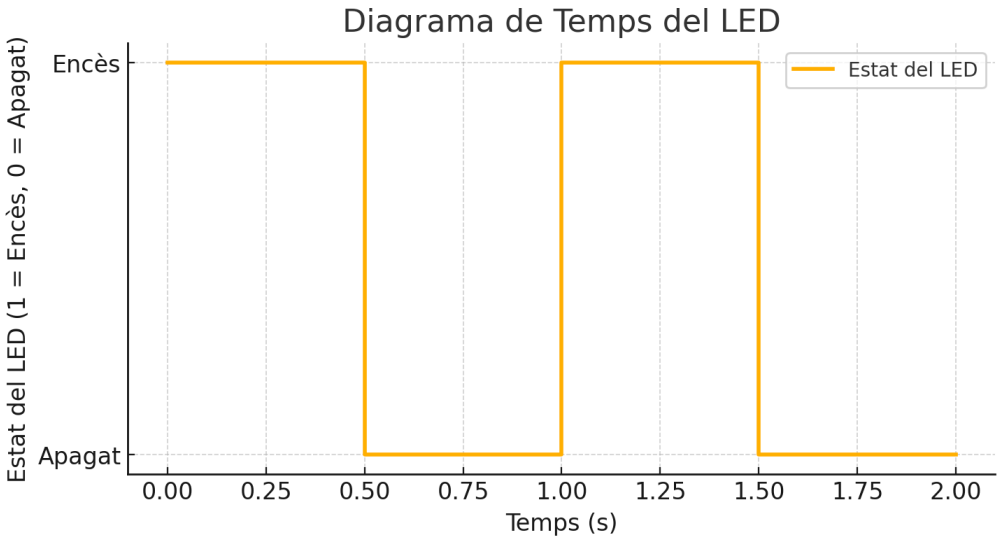


Diagrama de temps



Pregunta: Quin és el temps lliure del processador?

El temps lliure del processador depèn de la implementació utilitzada. En les versions amb `delay()`, el processador es manté inactiu durant aquest temps. En canvi, sense `delay()`, el processador pot executar altres tasques.

Si es vol aprofitar millor el temps lliure del processador, es pot implementar un sistema basat en interrupcions, permetent que l'ESP32-S3 faci altres tasques mentre espera canvis d'estat.

Conclusions

- L'ús de la biblioteca **Adafruit NeoPixel** facilita el control dels LED RGB, però pot introduir latències.
- L'**accés directe als registres** permet assolir una freqüència màxima d'encesa i apagada del LED.
- L'**ús del port sèrie** redueix la velocitat d'execució.