Pau Pont Camp

INFORME FINAL DE PRÀCTIQUES

Pràctica 1: Blink amb ESP32-S3

1. Objectiu

L'objectiu d'aquesta pràctica és produir el parpelleig periòdic d'un LED utilitzant el microcontrolador ESP32-S3. També s'inclou l'enviament d'informació a través del port sèrie i l'optimització del codi per accedir directament als registres del microcontrolador.

2. Configuració i funcionament bàsic

Inicialment, es configura el pin de sortida per al LED i es programa un bucle infinit per alternar el seu estat entre **encès** i **apagat** amb un retard de 500 ms. A més, s'imprimeixen missatges pel monitor sèrie per verificar el funcionament correcte.

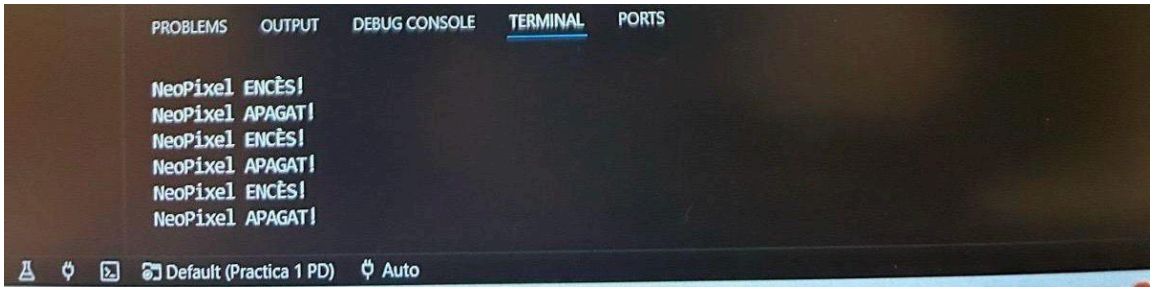
Codi utilitzat (fragment rellevant):

```C/C++
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
    Serial.begin(115200);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("ON");
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("OFF");
    delay(500);
}
```

3. Modificació per a l'enviament de dades pel port sèrie

S'ha afegit la funcionalitat d'enviar missatges pel port sèrie cada vegada que el LED canvia d'estat. Això permet verificar el seu funcionament en temps real.



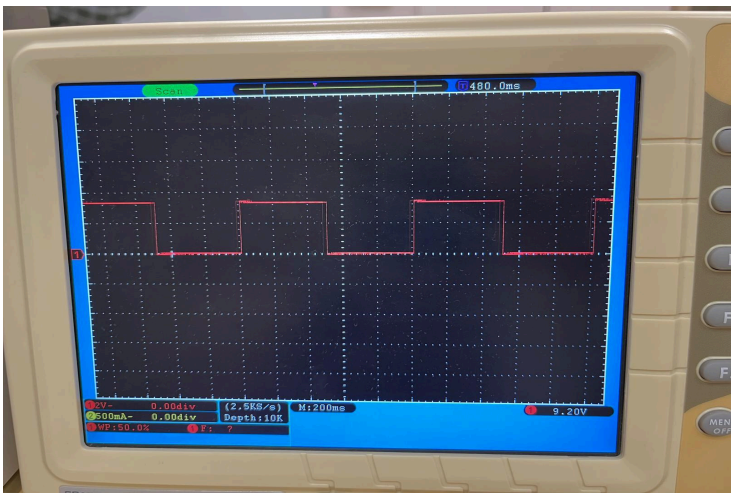
4. Accés directe als registres del microcontrolador

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.



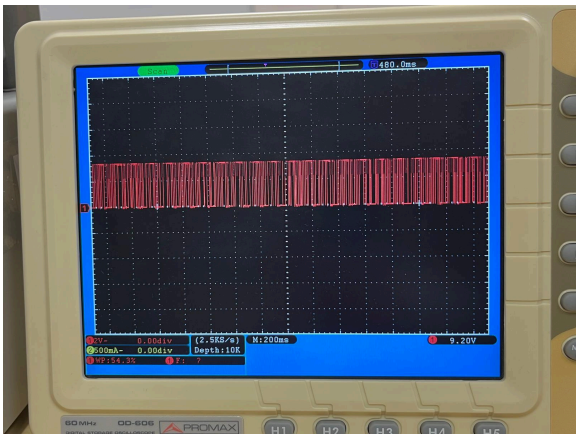
5. Eliminació de delays i mesura de la freqüència màxima

Per mesurar la freqüència màxima de commutació del LED, s'ha eliminat la funció **`delay()`**, i s'han realitzat proves en quatre casos diferents:

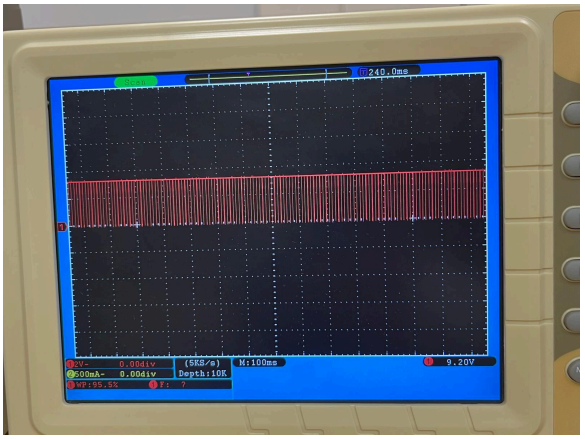
1. Amb enviament pel port sèrie i ús de funcions d'Arduino.
2. Amb enviament pel port sèrie i accés directe als registres.
3. Sense enviament pel port sèrie i ús de funcions d'Arduino.
4. Sense enviament pel port sèrie i accés directe als registres.

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

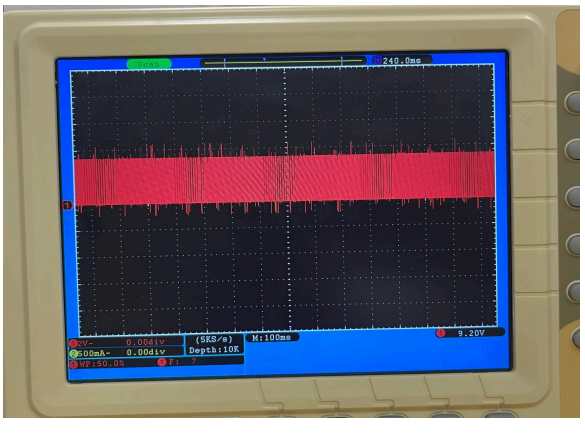
- **Cas1:**



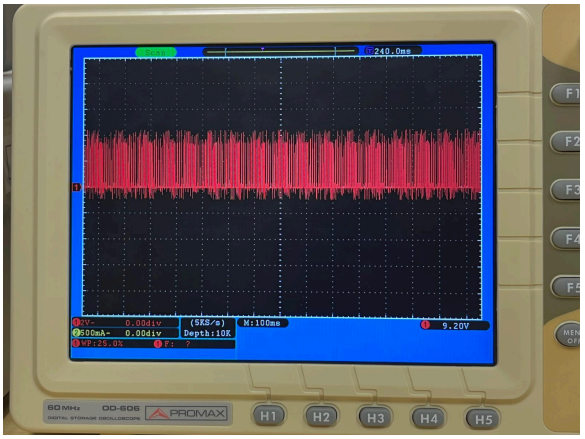
- **Cas 2:**



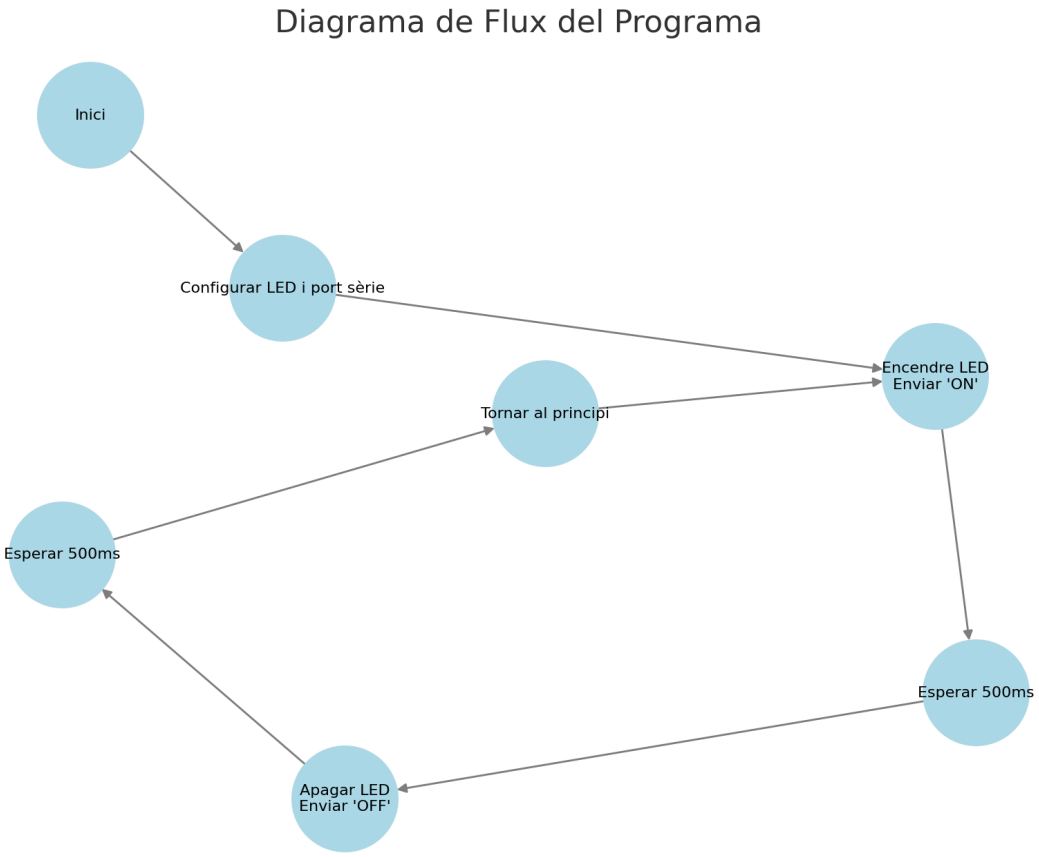
- **Cas 3:**



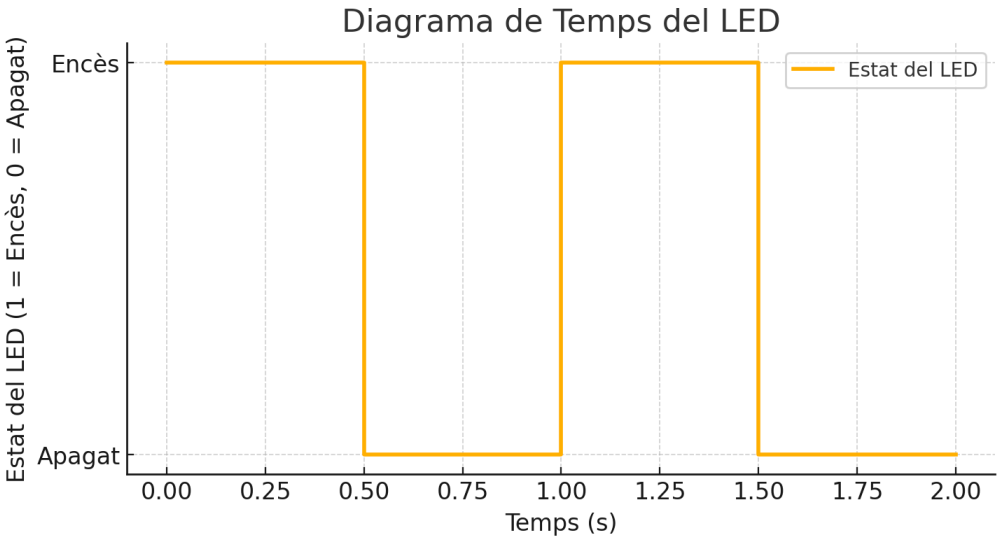
- **Cas 4:**



6. Diagrama de flux del programa



7. Diagrama de temps



8. Temps lliure del processador

La quantificació del temps lliure del processador depèn del temps que el microcontrolador passa executant instruccions versus el temps que està esperant.

En aquest cas, si utilitzem **`delay()`**, el processador no està fent res útil durant aquest temps. No obstant això, en eliminar els **`delay()`**, el processador queda gairebé totalment ocupat alternant l'estat del LED.

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.

Conclusió: Aquesta pràctica ha permès comprendre el funcionament del parpelleig d'un LED en un ESP32-S3, optimitzar el codi per a un millor rendiment i mesurar la freqüència màxima que es pot obtenir en diferents escenaris. També s'han analitzat els avantatges de l'accés directe als registres del microcontrolador en comparació amb l'ús de funcions d'alt nivell.