Driver para interface de leds HUB-75

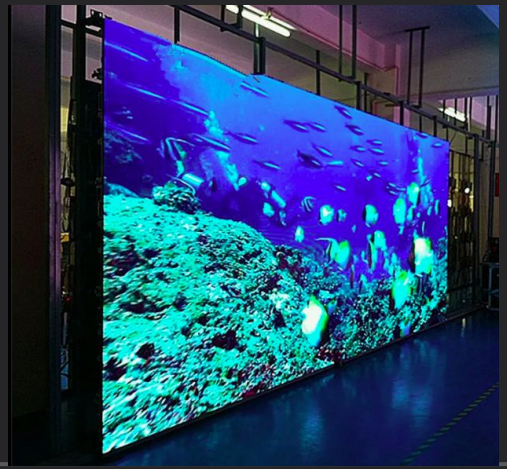
Laboratorio 3 - Instituto Balseiro

Lorenzo Cabrera Blanch - 22-11-2024

Motivación

- Display para usuario final de baja resolución

- Pantallas enormes



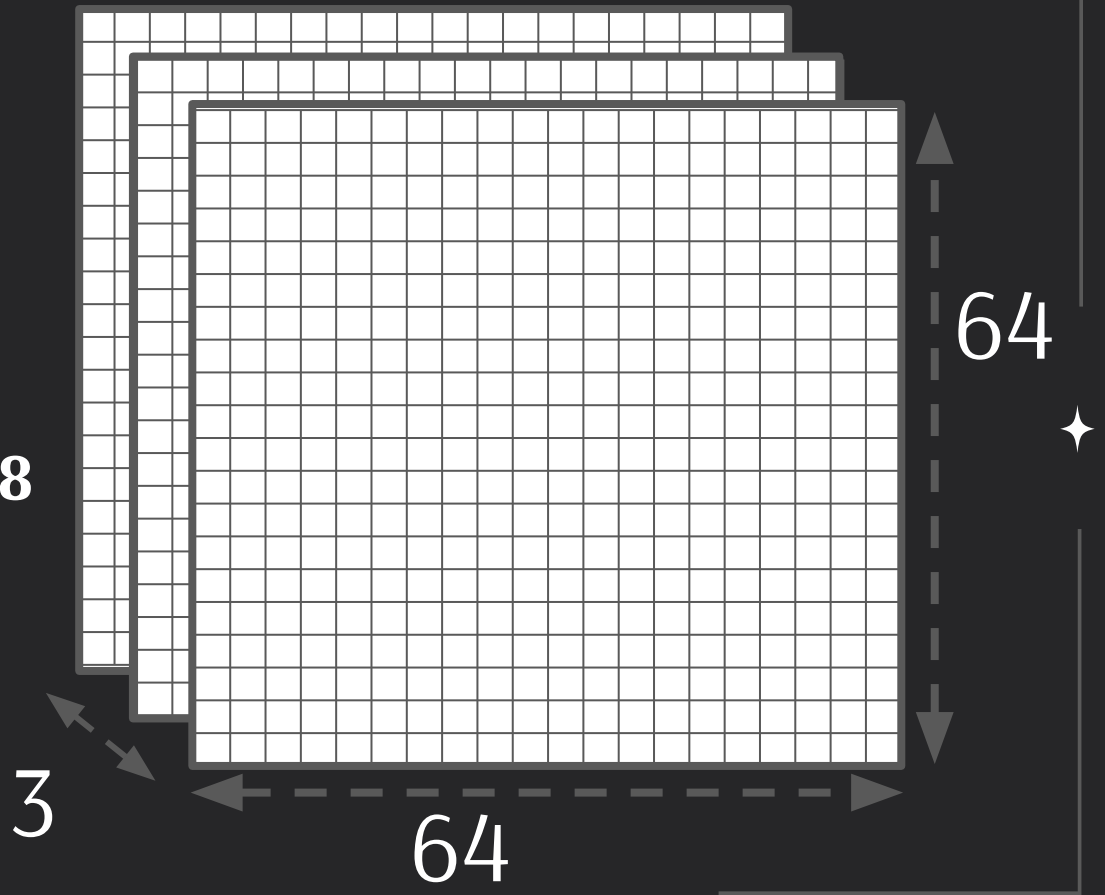
- **LUCES!**

Desafíos

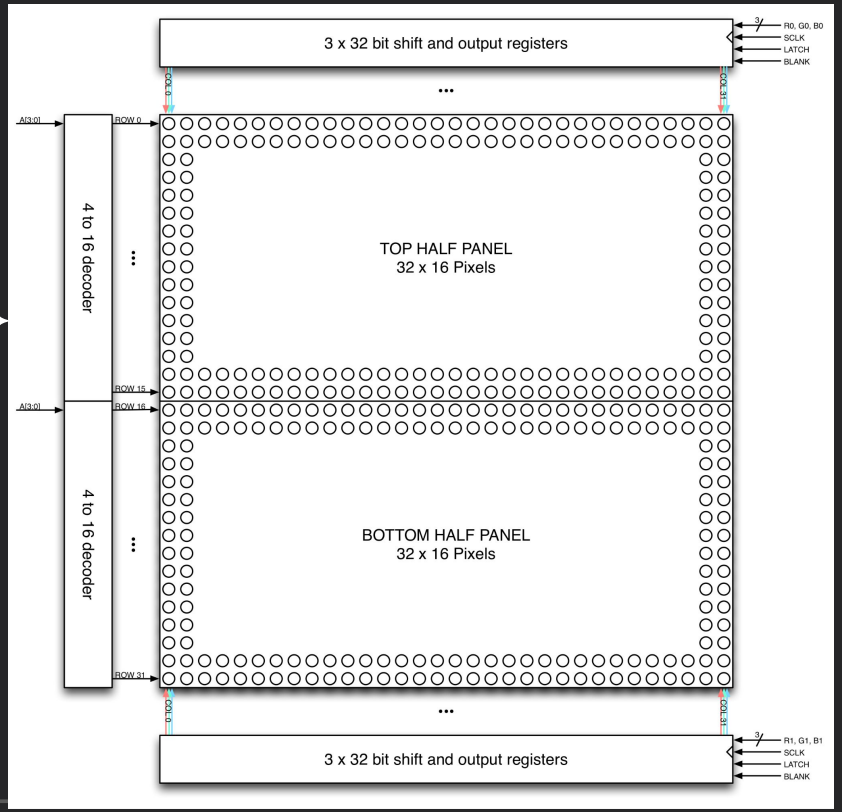
Para una profundidad de colores de 1 byte cada imagen son

98304 Bytes = 64x64x3x8

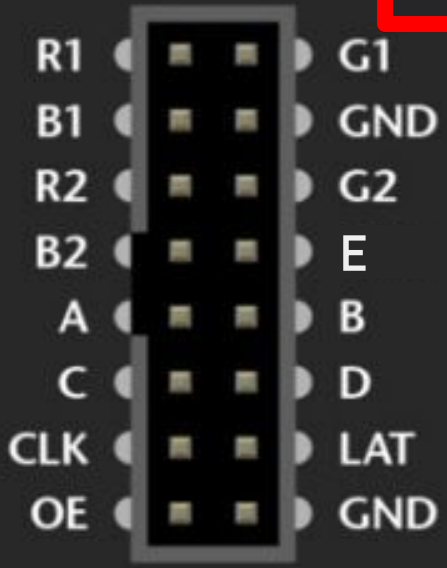
Para prevenir el parpadeo este proceso hay que **repetirlo entre 30 y 100 veces por segundo**.



Interface HUB-75

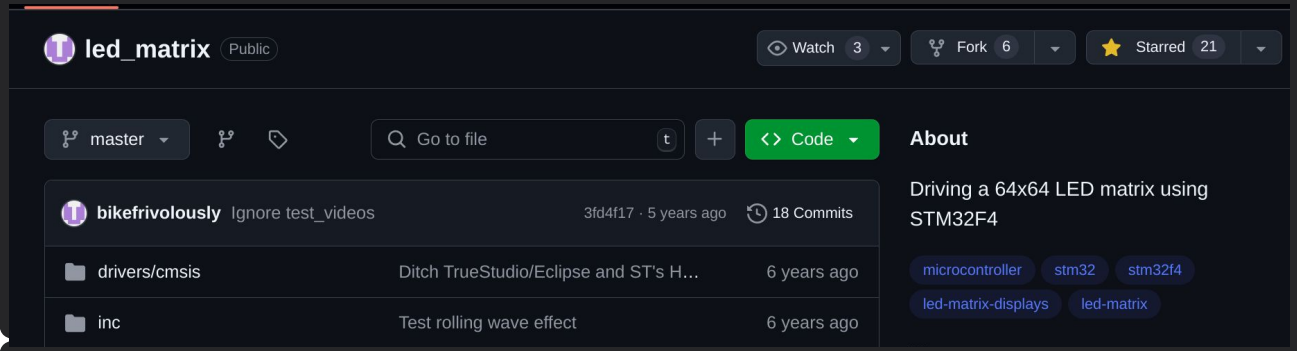


Escribir de a filas



CLK: Clock de entrada de datos
LAT: Señal de que terminó una fila
OE: Output enable
ABCDE: Selector de fila

Driver de Leds



Configura los clocks y **dos timers**, GPIO y DMA:

Generación del CLK (TIM8):

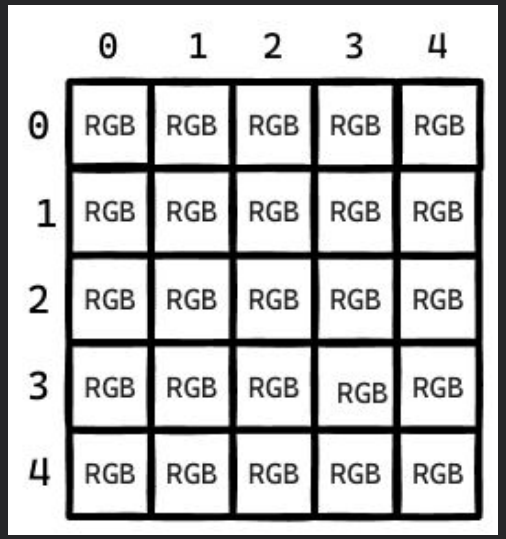
- **TIM8** en modo PWM genera un reloj constante para la matriz.
- Cada flanco descendente del reloj dispara una transferencia de **DMA**.

TIM5 está sincronizado con **TIM8** pero opera a una frecuencia **64** veces menor.

- **CH1 (LAT)**: Activar el latch tras completar el envío de 64 bits de datos.
- **CH2 (OE)**: Controla el tiempo que las filas seleccionadas están activas, modulando el brillo.

Implementación

```
struct RGB {
    uint8_t R;
    uint8_t G;
    uint8_t B;
};
```




```
RGB frame[64*64];
```

Frame

Se adapta a

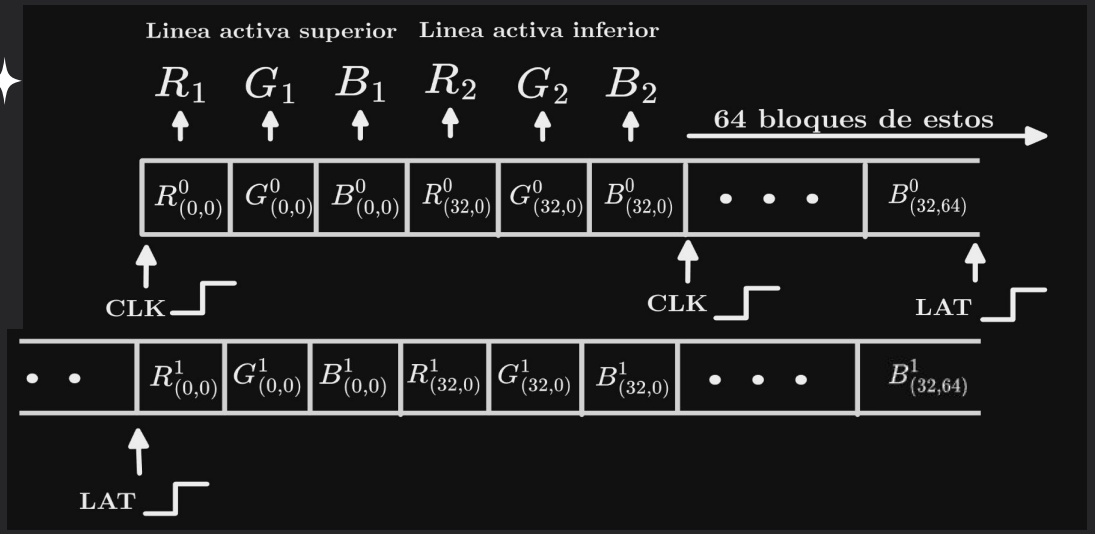
Buffer 1

Buffer 2

Mientras leo uno escribo en el otro

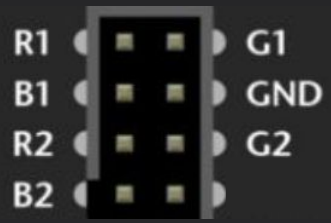
Direct Memory Access - DMA

Buffer Actual



Falling Edge
Tim 8

DMA



(Valores RGB)

Después
de 64

TIM5 → **LAT**

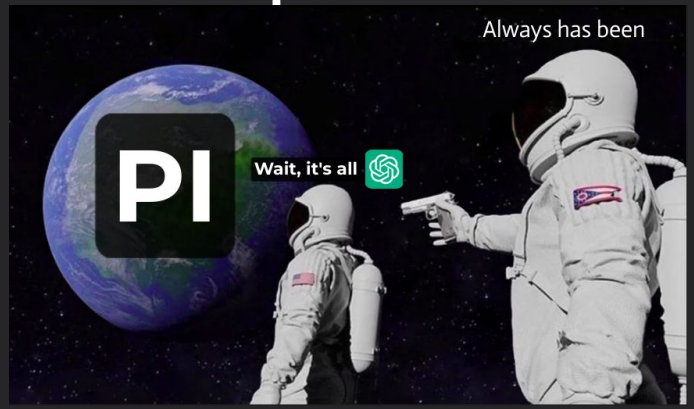
Cambia PWM de OE
Cambia fila

Problemas Encontrados

- **Conflicto entre interrupciones y lecturas del ADC:**

TIM6 global interrupt and DAC1, DAC2 underrun error interrupts	☐	0

- **Setup original:**
  - **Las configuraciones se hacían directamente con operaciones de bits.**




```
DMA2_Stream2->PAR = (uint32_t)&(GPIOC->ODR);
DMA2_Stream2->M0AR = (uint32_t)buffer1;
DMA2_Stream2->M1AR = (uint32_t)buffer2;
DMA2_Stream2->FCR |= DMA_SxFCR_DMDIS;
DMA2_Stream2->CR |= DMA_SxCR_TCIE;
DMA2->LIFCR |= DMA_LIFCR_CTCIF2;
```