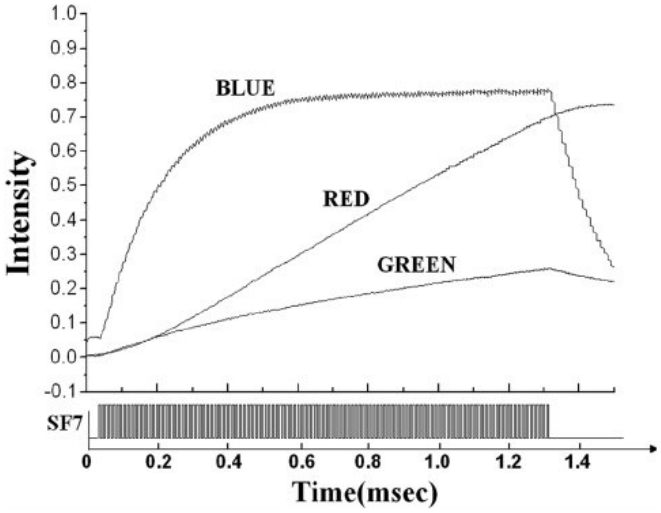
Architetture dei Sistemi di Elaborazione	Delivery date: 10th December 2024
Laboratory 10	Expected delivery of lab_10.zip must include: - zipped project folder of exercise 1 and exercise 2. - this lab track completed and converted to pdf format.

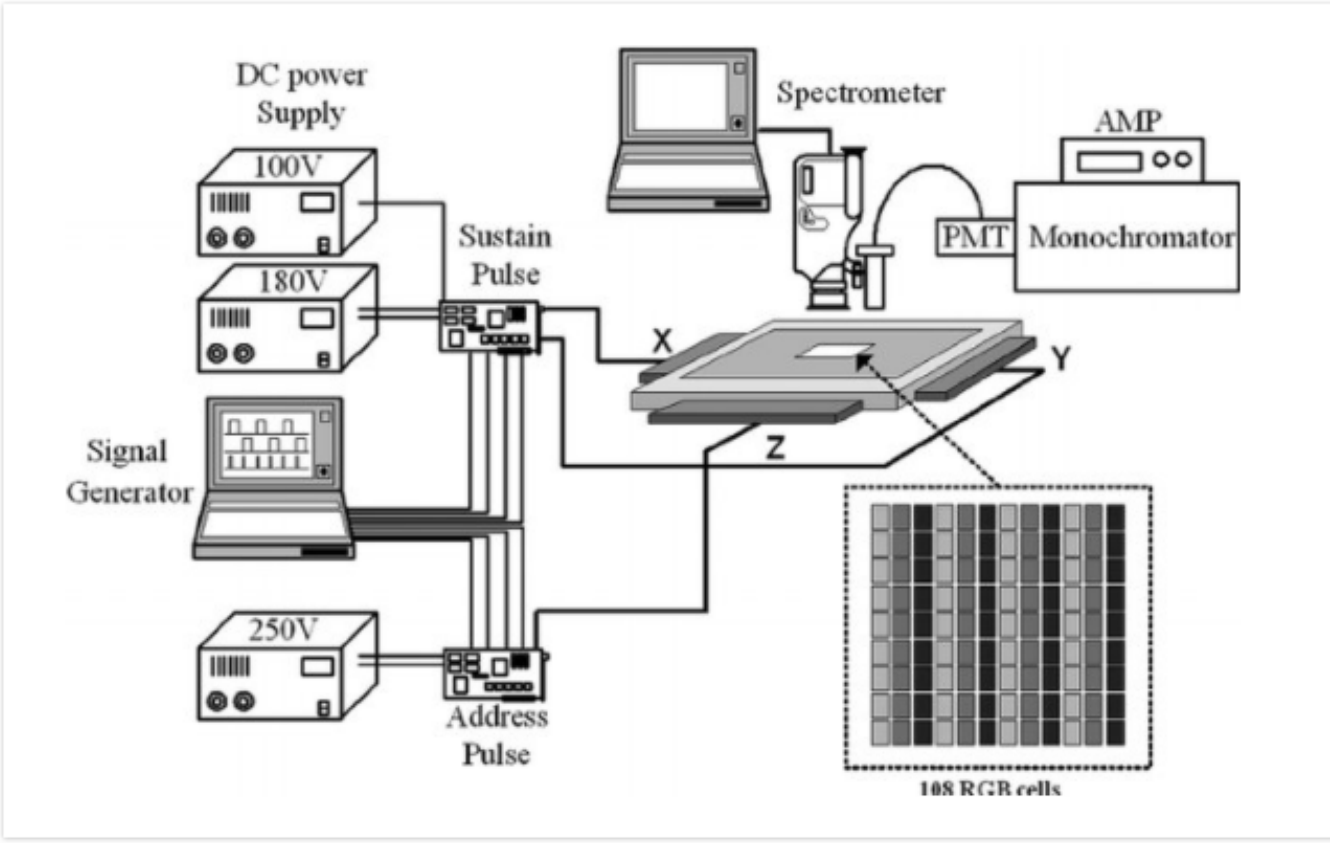
Exercise 1) Implement a system on the LANDTIGER board that can tune the brightness of an LED by making use of TIMERS.



Dimming a LED is done by appropriately turning it ON and OFF. Usually, the maximum brightness (100% brightness) is achieved when it is ON for at least a period T period. Such T time value can be often found in the LED datasheet (when available). Please see an example, where also blue and green LEDs are considered, which shows a different behavior (non-linear).

For LANDTIGER red LEDs, the time to saturate should be in the order of few milliseconds; during this time the brightness increases almost linearly from no light to the maximum luminescence.

To determine the brightness of a LED is quite difficult and require expensive machineries (as shown below). We suggest you perform just a visual comparison with a setup where a LED modulated while another is always on.

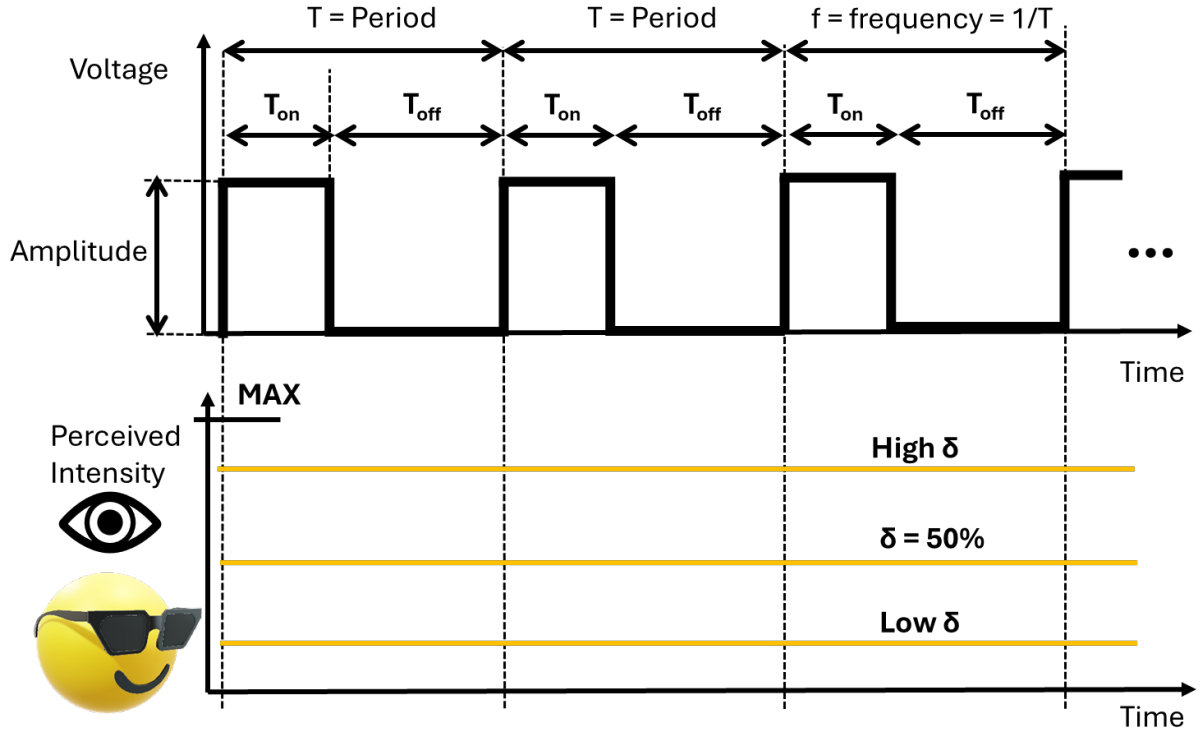


In this lab, you are required to experiment with LEDs dimming using Pulse Width Modulation (PWM) via software, i.e., achieved by using GPIO outputs (on LEDS) and timers.
There exist specific peripherals for generating PWM signals (even in the LPC1768 SoC, look for them!), but they are out of the scope for this lab.

Unfortunately, a spectrometer for measuring LEDs intensity is not available in the lab.
We are going to use a rudimental non-linear photometric sensing device, **the human eyes** 😲.

The maximum frequency at which a human eye can perceive a light source (like an LED) flickering is related to the concept of the flicker fusion threshold. This is the frequency above which the human eye perceives a flickering light as a steady, continuous glow. For most people, the flicker fusion threshold is around 60 Hz to 90 Hz under typical lighting conditions. Thus, even if human eyes are the souls' mirror, they do not work good as intensity sensing device for digital signals!

The following figure represents how a PWM signal (based on a square wave) is build and how it is perceived by human eyes.



Remember that:

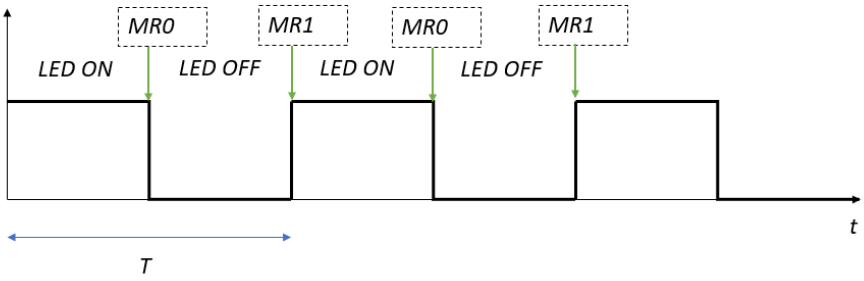
- Period (T) or frequency (1/T) is the amount of time for which the digital signal repeat itself.
- Time On (T_{on}) is the amount of time for which the digital signal stays at the logic value **ONE** (from an electrical perspective, it is the maximum voltage amplitude your GPIOs pins can reach).
- Time Off (T_{off}) is the amount of time for which the digital signal stays at the logic value **ZERO**.
- The duty cycle is defined as the amount of time the digital signal stays at the logic value ONE divided by the Total amount of time the digital signal repeats itself.
  In other words:
  $$\delta = \frac{T_{on}}{T}\ [\%] \quad where\ T = \ T_{on} + T_{off}$$
  A higher duty cycle results into a higher perceived intensity of the LED. A lower duty cycle results into a lower perceived intensity.

Therefore, to achieve a 50% brightness (perceived intensity), for instance, the LED would have to be ON half of the time in each T period (δ = 50 %).

Note: When a LED is blinking with period T without mention the duty cycle, it implies that the duty cycle is 50 %.

To solve the exercise:

1. Select one of the available LEDs, and use two Match Registers (MR0 and MR1) of TIMER 2 to synchronize the two phases. Specifically, it is necessary to set the two Match Registers so that, when TIMER 2 reaches MR0 the LED turns OFF; when the timer reaches the value held in MR1, the LED must light up.
    a. Selects a frequency that avoid LED flickering.



2. Select another of the available LEDs, and use two Match Registers (MR0 and MR1) of TIMER 3 to synchronize the two phases (as before).
    a. Selects a frequency that does not avoid LED flickering.
3. Compare the TWO LEDS with the same duty cycle (50%) and different frequency.

What is happening?
One of the two leds, the one with the highest frequency, appears always active even though faded. The other one is visibly blinking, but when it's ON it can be seen that it has the usual amount of light.

Fill in the table the required brightness for Timer 2.

Brightness [%]	MR0	MR1	Time ON	Time OFF	Period	Frequency
25	*0x00007A12*	*0x0001E848*	*0.00125ms*	*0.00375ms*	*0.005ms*	*200Hz*
50	*0x0000F424*	*0x0001E848*	*0.0025ms*	*0.0025ms*	*0.005ms*	*200Hz*
75	*0x00016E36*	*0x0001E848*	*0.00375ms*	*0.00125ms*	*0.005ms*	*200Hz*
100	*0x0001E848*	*0x0001E848*	*0.005ms*	*0ms*	*0.005ms*	*200Hz*

Fill in the table the required brightness for Timer 3.

Brightness [%]	MR0	MR1	Time ON	Time OFF	Period	Frequency
25	*0x001DE848*	*0x0077A120*	*0.005ms*	*0.015ms*	*0.02ms*	*50Hz*
50	*0x003BD090*	*0x0077A120*	*0.01ms*	*0.01ms*	*0.02ms*	*50Hz*
75	*0x0059B8D8*	*0x0077A120*	*0.015ms*	*0.005ms*	*0.02ms*	*50Hz*

100	*0x0077A120*	*0x0077A120*	*0.02ms*	*0ms*	*0.02ms*	*50Hz*

NOTE:

1. To use TIMER2 and TIMER3 functionalities, you must extend the available libraries and create the appropriate functions.

Exercise 2) Starting from the project of Exercise 1, enhance the firmware with the following features:

- A free running timer (TIMER0), i.e., a timer that does not reset neither generate interrupts.
- A sampling timer (TIMER1) that retrieves the current counter of TIMER0 with a frequency 100 times bigger than the TIMER2 frequency. The current counter of TIMER0 is saved in a circular buffer of 7000 elements.
- When KEY1 is pressed (the buttons are debounced!!) in the related interrupt handler it starts the computation of the circular buffer (you need to process the 7000 elements independently from the circular buffer index).
    - First, you need to sort the vector with a $O(n^2)$ in time complexity (e.g., the bubble sort algorithm).
    - Second, you need to compute the average value in the circular buffer.
    - Third, you need to clean the vector with the pattern 0xCAFECAFE.

What is happening to LED dimming and frequencies if you repeatedly press KEY1?

LEds stop blinking because the interrupt related to the RIT is CPU intensive and so other interrupts are disabled. Also, the IR bits of the timers are set but are never reset. So, the LED blinking never resumes because both interrupts of MR0 and MR1 are set and no condition checks this case.

How can you solve the issue?

The solution is to add a condition in the timer handler that, when both interrupts of MR0 and MR1 are activated, they will be reset.