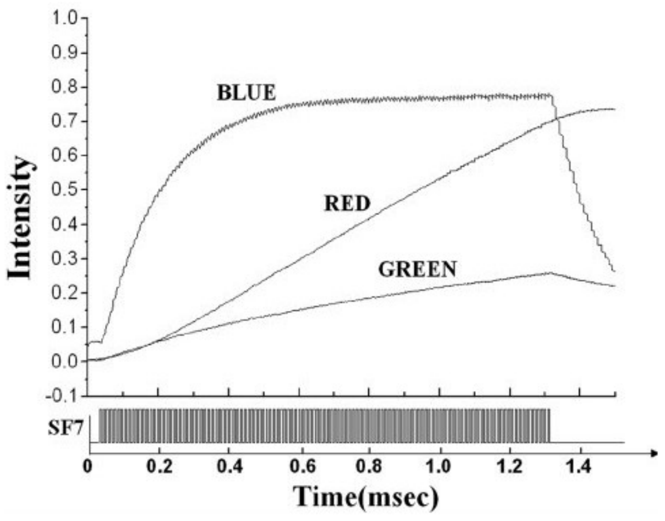
Architetture dei Sistemi di Elaborazione	Delivery date: 10th December 2024
Laboratory 10	Expected delivery of lab_10.zip must include: - zipped project folder of exercise 1 and exercise 2. - this lab track completed and converted to pdf format.

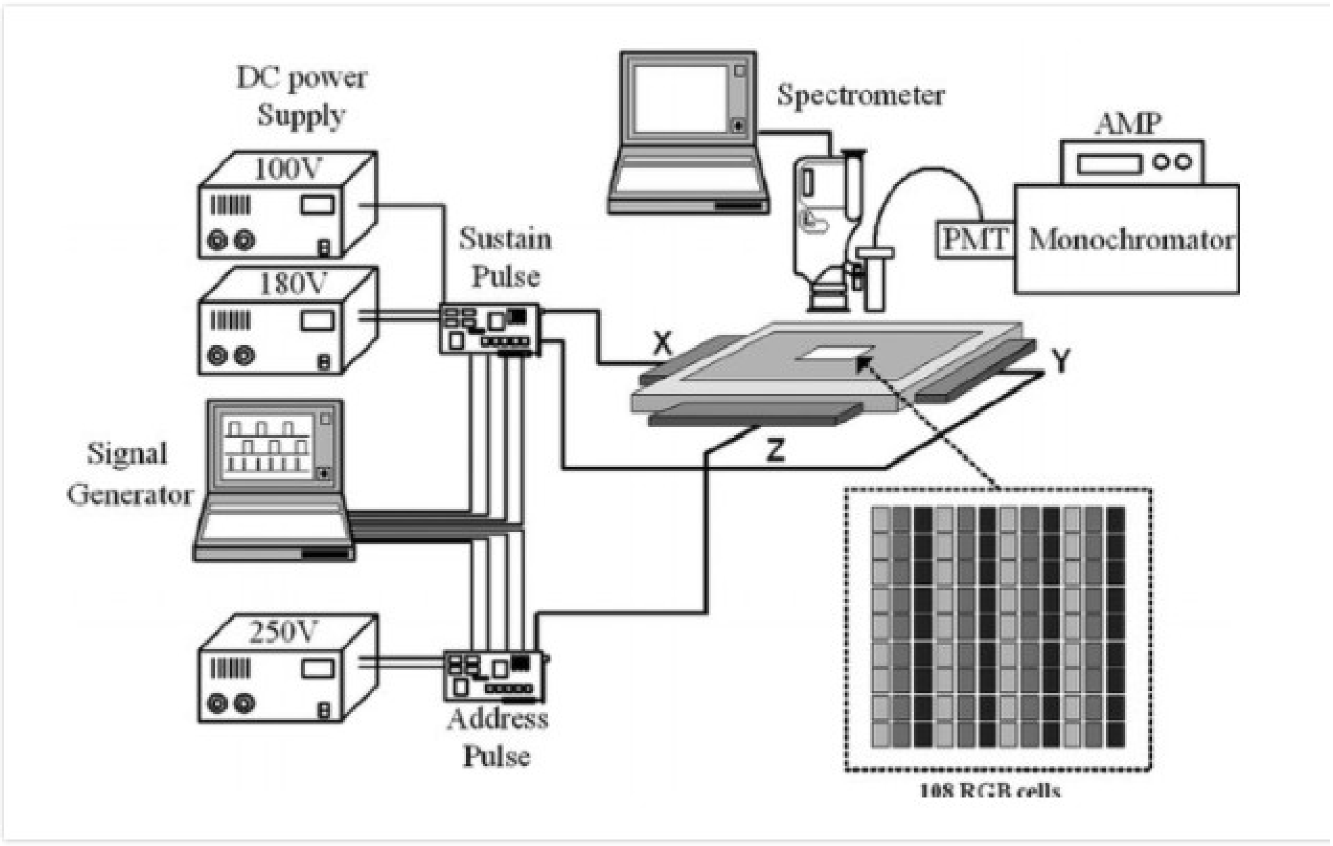
Exercise 1) Implement a system on the LANDTIGER board that can tune the brightness of an LED by making use of TIMERS.



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.

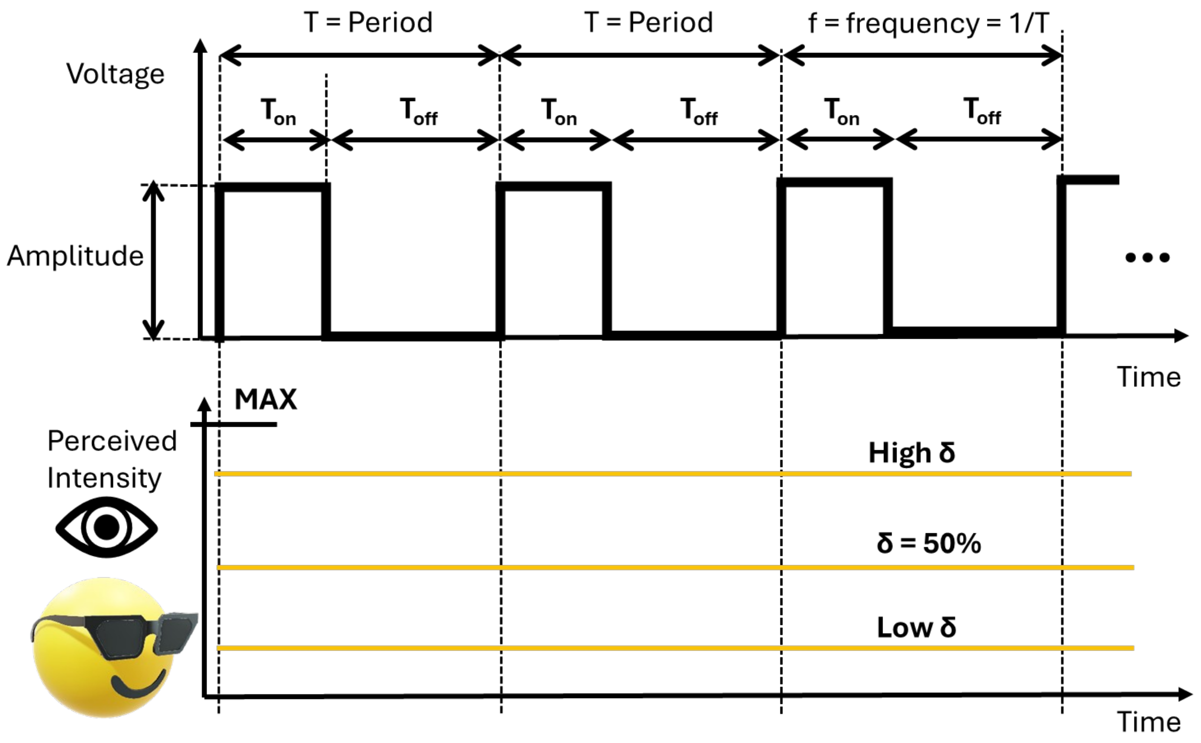


In this lab, you are required to experiment with LEDs dimming using Pulse Width Modulation (PWM) via software, i.e., achieved by using GPIO outputs (on LEDS) and timers.
There exist specific peripherals for generating PWM signals (even in the LPC1768 SoC, look for them!), but they are out of the scope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



<u>Remember that:</u>

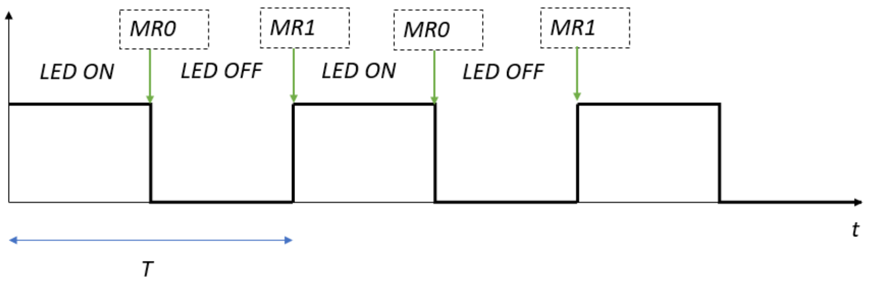
- Period (T) or frequency (1/T) is the amount of time for which the digital signal repeat itself.
- Time On (T_{on}) is the amount of time for which the digital signal stays at the logic value **ONE** (from an electrical perspective, it is the maximum voltage amplitude your GPIOs pins can reach).
- Time 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:
  $$\delta = \frac{T_{on}}{T}[\%] \, where \, T = T_{on} + T_{off}$$
  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?
The first LED (LED number 0) emits a stable, everlasting light, whereas the second LED (LED number 1) flickers very rapidly.
This is because in the first case I set the timer 2 interval having in mind a Time Period (0.01 s in my code) that is the opposite of a frequency residing residing in the range [60, 90] Hz or much higher than that, meanwhile I set timer 3 using a much higher Time Period (0.07 in my code) that is the opposite of a much lower frequency (about 15 Hz in my code).

This is related to how the human eye works and actually perceives light. For frequencies >> 60 Hz (ex 100 Hz) the flickering is almost erased and the human eye doesn't perceive a significative change in light brightness. For much lower frequencies, on the other way, the flickering is visible.

Fill in the table the required brightness for Timer 2.

Brightness [%]	MR0	MR1	Time ON	Time OFF	Period	Frequency
25	**0xF424**	**0x3D090**				**100 Hz**
50	**0x1E848**	**0x3D090**				**100 Hz**
75	**0x2DC6C**	**0x3D090**				**100 Hz**
100	**0x0**	**0x3D090**				**100 Hz**

Fill in the table the required brightness for Timer 3.

Brightness [%]	MR0	MR1	Time ON	Time OFF	Period	Frequency
25	**0x6ACFC**	**0x1AB3F**				**15 Hz**

		0				
50	**0xD59F8**	**0x1AB3F 0**				**15 Hz**
75	**0x1406F4**	**0x1AB3F 0**				**15 Hz**
100	**0x0**	**0x1AB3F 0**				**15 Hz**

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?
The LEDs used in Ex1 do not behave well, particularly they seem to be both switched off or in some cases just one of them emits light and the other is completely dark. This is because the system has to handle distinct Interrupts (KEY1 button, RIT for debouncing, timer 0, 1, 2 and 3) that can potentially be raised at the same time.

How can you solve the issue?
The issue can be solved by manually modifying the Priorities of the interrupts in the NVIC.
We have these interrupts in the program:

- Interrupt related to the timers 0, 1, 2, 3
- Interrupt related to button KEY1
- RIT Interrupt

I have to manage these interrupts properly to priviledge the good behavior of the LEDs, so I set the priority of timers 2, 3 to the **maximum possible priority level (so, the minimum integer: 0)**, then I set the priority of KEY1 Interrupt to a higher integer value, signifying **lower priority** (ex value 5).
Also, in order to correctly prevent deboucing I have to set the priority of RIT to a higher priority than KEY1 Interrupt (So I'd give RIT priority a lower integer than KEY1 priority),