**Practice 2: PWMs**

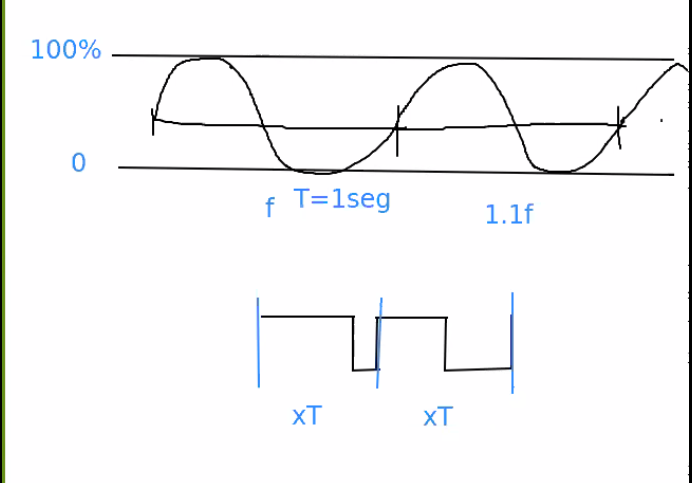
1. **Objective**

Vary the **intensity of brightness** using digital output(PWM) and the **frequency of cycle**(On-Off) in a LED. The LED must vary its intensity according to a sinusoidal signal. Once an on-off cycle due to sine wave has concluded, the frequency of the next cycle must increment in 0.1f until reaching 10f. After reaching 10f, decrement in steps of -0.1f down to 1 and repeat the sequence

Requirements:

* Use Interruptions
* At least 2 timers are requires: 1 to generate the period and another to calculate the value of the sine wave.
* Use library Libopencm3.
* Initial frequency is 1Hz.

.



F1. Variación de intensidad y periodo del LED

1. **Development**

**2.1 Reproducing a sine wave**

Angular FrequencyThe formula for the Sine wave is, Sine Wave Equation.

A = Amplitude of the Wave

ω = the angular frequency, specifies how many oscillations occur in a unit time interval.

Remembering: the PWM output on a digital pin is the average voltage.

AFIO(Alternate Function IO): Además de trabajar como GPIO, muchos pins tienen otra funcionalidad(SPI, CAN, I2C…), Alternate Function permite seleccionar en que modo trabaja el pin indicado y además provee formas de remapearlos a los pines indicados por el fabricante.

Condición para incrementar/disminuir el periodo:

Si el periodo(T) es igual a el tiempo(t) que ha pasado en ese ciclo, T==t, significa que se ha completado un ciclo(de prendido apagado) por lo que es necesario cambiar a la siguiente frecuencia.

**Output Control**

Control de señal de salida (sección **15.3.8 Output compare mode)** debes configurar Output Control para indicar cuando el periodo ha concluido.

**PWM Generation**

La sección **15.3.9 PWM mode**, describe los modos de generación PWM:

Pulse width modulation mode allows generating a signal with:

- A frequency determined by the value of TIMx\_ARR.

- A duty cycle determined by the value of the TIMx\_CCRx

Frequency generation

15.3.10 One-pulse mode: makes the counter stop automatically at the next update event UEV(for example: Counter overflow/underflow, manual setting UG bit…).

Bit 0 UIF: Update interrupt flag

– This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow and if the UDIS=0 in the TIMx\_CR1 register.

Questions

1. How can we entrust a project with unstable components?
2. How can we deal with a developing Library and its unstable API?

Status and API

**--------------**

The libopencm3 project is currently work in progress. Not all subsystems

of the microcontrollers are supported, yet.

**\*\*IMPORTANT\*\***: The API of the library is *\_NOT\_* yet considered stable! Please do

           not rely on it, yet! Changes to function names, macro names, etc.

           can happen at any time without prior notice!

*\_TIP\_*: Include this repository as a Git submodule in your project to make sure

     your users get the right version of the library to compile your project.

     For how that can be done refer to the libopencm3-examples repository.