**Universidad Autónoma de Guadalajara**

Ingeniería Electrónica Biomédica

System designing with Microprocessors

*“*Practice 4: Fan Speed Control”

Jesús Arnoldo Zerecero Núñez

2885993

Andrea Alejandra Mondragón Olivos

2915351

Practice 4. Fan Speed Control

**Introduction**

A timer is a specialized type of clock for measuring time intervals. A hardware timer utilizes a quartz clock for high precision timing measurements, while a software timer relies on a digital clock generator. Timers are a very valuable resource for many applications, as using them can lead to easier information gathering.

Hardware timer modules are very flexible because of their many uses in software and hardware description languages. Most of the timer modules support these 3 types of operations:

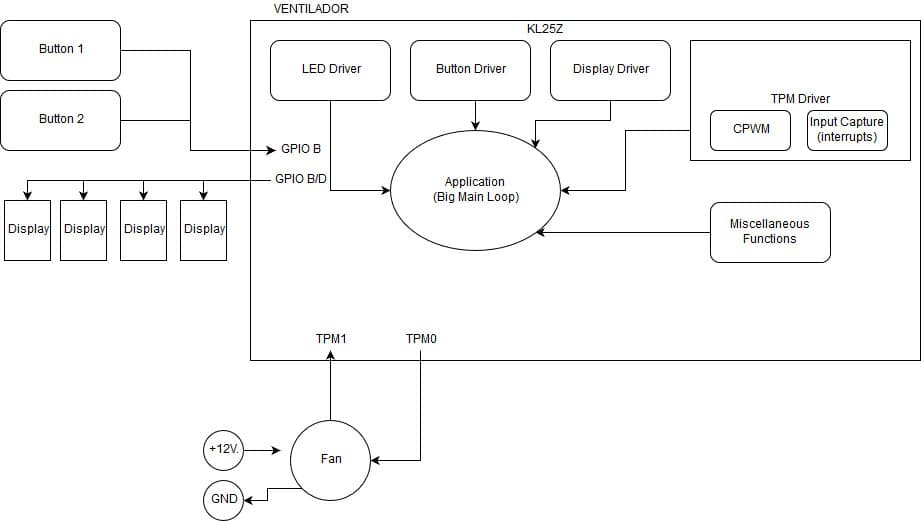
* **Input Capture Mode:** The counter counts up or down indefinitely and circles back to its reset state whenever it reaches its limit (Timer Overflow). Whenever an external event occurs, a timestamp is made of the current counter value, which can be saved and read by software.
* **Output Compare Mode:** Similar counting procedure as the Input Capture, except that a value is preloaded to the timer. Whenever the counter value is equal to this preloaded value, an event is generated.
* **PWM Mode:** The counter counts up until reaching its maximum limit then down until it reaches minimum limit indefinitely. The period of time it takes to count from one limit to another can be adjusted, thus generating an adjustable duty cycle for a PWM signal.

In this practice, a fan speed control application was built using the TPM module of the KL25z. External buttons can increment or decrement the speed of the fan ranging from 7200 RPM up to 12000 RPM with a 12v voltage source. The fan used for this practice was the PMD1204PQB1-A, which can reach up to 15000 RPM with a 13.8v voltage source.

The RPM were calculated by reading the fan’s tachometer, and were shown using 4 external seven segment displays.

**Development.**

**Block diagram/modules implemented:**

****

**Conclusion:**

The only difficulty we had was coming up with a formula to translate a 2 pulse tach while having the TPM’s input capture clk divided into RPM. With this practice I learned the importance of using timers for any application. Other smaller problems included planning the code’s structure and drivers’ format in order to make the code very time efficient. It’s important to note that the PMD1204PQB1-A tachometer gives two pulses per revolution, so this must be taken into account when generating a formula that turns the time difference between 2 input capture values into RPMs.

**Link to demonstration video:** [**https://photos.app.goo.gl/3ffdFbdUqYCATYiy1**](https://photos.app.goo.gl/3ffdFbdUqYCATYiy1)