Lab 2 report

Erik Alm & Mario Rios

2016-04-07

Abstract

In this session we are getting familiar with PWM (Pulse Width Modulation), and handling of input and output via GPIO (by controlling LEDs, and capturing button events). Moreover, aside session requirements, we are using POSIX signal handling, to capture SIGINT events, allowing us to properly shut down (uninitialise) GPIO pins, when program is forced to quit.

1 Breathing LED

This was our first time with Power Width Modulation (PWM), so it took some getting used to it. Once we got familiar with the concept, we realised that the shorter the period (on and off time), the smoother the effect was, to a point where it can't actually be noticed that the light is turning on an off. Depending on the balance between on an off time, we could control the led intensity.

In relation to our implementation, we decided to run the pulse at 60Hz (is the most common refresh time for monitors). In order to implement the breathing effect, we use a counter, whose range goes from 0 to 100 (representing the duty percent). The counter is increased or decreased, depending on the direction of the breath (so once the counter reaches one of the bounds, the direction is inverted).

2 Using Buttons (On/Off)

This was the first time we had to read from GPIO (instead of just writting to it). The biggest challenge in this section, was capturing the actual trigger event. Our first approach, was just checking the value of the button pin, switching the status of the led depending on that button value. Of course this wouldn't work, since human reaction time is much slower than the one of the process, so chances were, that by keeping the button pressed, the led would be constantly changing it's status.

Instead of this, our final working approach checks wether the button is changing from low to high (pressed to released) and viceversa. To implement this, we use a variable that stores the last known value of the button (called btnstatus).

The initial value of that variable is 1 (button up). When the old value is 1 and the read input value is 0, that means the button has been pressed, thus we update the new value of the button to 0. The other way around, if the old value is 0, and the new value is 1, we update the old value to 1, and force the led status to change. (So only when the button is depressed, the led value is switched).

3 Using Buttons (PWM)

The button implementation is the same one explained in the previous section. What changes, is the action performed when the button is depressed. Instead of just switching the value, we have 5 different states, stored in an array (each state is the light intensity). Every time the button is pressed, a counter is increased (within the bounds of the array, cycling back to 0 when top level is reached). That counter is used, to send the proper intensity to the led.