Lab 2: Breathing LED, Buttons

Get familiar with and implement PWM, practice with buttons

Farhang Nemati, March 2016

In this lab you will learn and practice the concept of Pulse Width Modulation (PWM). Then you will use it to implement a breathing LED. You will later on need PWM in Lab 4 as well. You will also learn how to work with buttons.

A. Breathing LED

A.1.

Reuse the blinking LED program you implemented in Lab 1. Instead of sleep() use clock_nanosleep() function to have more precise delays. Change the program so that the LED is turned on for 20ms (milliseconds) and it is off for another 20ms. Let the program do the cycle for enough number of times (e.g., 2000 times). Now change the program so that the LED is turned on for 10ms and is turned off for 20ms. Finally change the program so that the LED is on for 5ms and is off for 20ms. What is the difference among the alternatives with regard to the shining strength (brightness) of the LED?

Pulse Width Modulation

The main use of PWM is to control the power supplied to electrical devices, especially to inertial loads such as motors etc. Assume a motor is going to be controlled by an output that supplies power to the motor. When the output is high the motor will run with a constant speed. What would you do so that the motor runs with half of this speed? If the output is high for a duration of time, say t1 and is low for t1 time units the motor will run for t1 time units and it will stop for t1 time units. If this process is continued the motor will be running for 50% of time, thus it will have 50% of full speed.

For example if the power is on for 1 second and is off for 1 second the motor will run every other second and its average speed would be 50% of its full speed. However, the duration of 1 second is too long and one can clearly see that running and stopping of the motor is too discrete. If the frequency of running and stopping the motor (turning on and off the power output) is decreased enough one will not notice that running and stopping is discrete but it will seem more continues. For example if the power output is on for 10ms and is off for 10ms the motor will seem to run continuously. It will still run with 50% of its full speed because it runs for only 50% of time. It seems continues because we increased the frequency of turning on and off. What we do is to have pulses of 20ms and during it the motor is on for 10ms and off for 10ms, i.e., for 50% of the pulse the motor is on and it is off for another 50% of each pulse. What would you do to run the motor with 25% of its full speed? This is how PWM is used.

A PWM has a frequency which indicates the length of the pulse. For example if the frequency of PWM is 100HZ (100 pulses per second) it means the length of each pulse is 1/100 second = 10ms. The duration of time during which the output is high (for example 25%) is called *duty percent*. For the motor example mentioned above, to control the speed of a motor we can have a PWM that has a pulse length of 20ms (the frequency is 50HZ). If we want the motor runs with 25% speed we have to set the duty percent to 25%. This means during of each pulse (20ms) the output is on for 25% of the pulse (for 5ms) and is off for 75% of the pulse (15ms).

A.2.

Implement PWM. Download the code (pwm.h and pwm.c) from Blackboard. Implement functions `pwmCreate()` and `pwmPulse()` in pwm.c. `pwmCreate()` is used to create a PWM on a GPIO pin with a frequency and `pwmPulse()` is used to issue a duty percent on the pin. When you create a PWM on a GPIO with frequency, say f,

the pin will be set to an output pin and when `pwmPulse()` is called for a given duty percent, say d, the GPIO pin should be high for d% of the length of a pulse that has frequency f and it should be low for (100-d)% of the length of the pulse. The length of a pulse with frequency of f is 1/f second. Call `pwmDestroy()` on any PWM `you create` after you are finished with it

A.3.

Use the PWM for implementing a breathing LED. The brightness of a LED has to gradually increase from 0% (off) to 100% (fully shining) and then the brightness has to gradually decrease from 100% back to 0%. Repeat this in a loop. Choose a good enough frequency for PWM so that it's continuous enough that it looks like the LED is breathing.

B. Using Buttons

There are 2 buttons already installed on your breadboard. Read Section "Using a GPIO pin for buttons and switches" in the document for General Instructions (it is on Blackboard). Use pull-up buttons here. Remember that the value of an input pin that a pull-up button is connected to by default is high. When the button is pushed the value of the input pin becomes low. One pin of each button is already connected to the GND. The other pin has to be connected to a GPIO pin which has to be created as a pull-up input pin.

B.1.

Implement a program where each of the LEDs are controlled by one of the buttons. When a button is pushed its corresponding LED has to be turned on. When the button is pushed again the LED should turn off.

B.2.

In this Section you will reuse the PWM you implemented in Section A. Implement a program where the brightness of a LEDs is controlled by a button. Each LED has to have 5 states; state 1 where the LED is turned off, state2 where the LED shines with 25% of its full brightness, state 3 where the LED shines 50%, state 4 where the LED shines 75%, and state 5 where the LED shines fully (100%). By default the LED has to be off. When its corresponding button is pushed, the LED should go to state 1, and when the button is pushed again it will go to state 2, and to state 3 when the button is pushed for the third time, to state 4 when the button is pushed for 4th time, to state 5 when it is pushed for 5th time, and finally it should go back to state 1 (turned off) when the button is pushed for the 6th time.

Examination

To pass the lab hand in a short report and your code (only in Blackboard!). You also need to demonstrate the programs for the lab assistant in one of the lab sessions. The lab assistant may ask you questions related to lab task.