CMPE1250 – GPIO and LEDs

By now you will have seen how to initialize the port that the switches and LEDs are connected to. You should also be able to turn on or turn off an individual LED by manipulating the bit necessary in `PT1AD1`. Remember: we only ever manipulate the minimum number of bits necessary to achieve our goal, as altering other bits could have undesired side effects.

In this activity, you will experiment with turning the LEDs on and off and will explore some of the timing characteristics of your micro. It will become common practice for you to measure time intervals with an oscilloscope or your AD2 and GPIO pins. This is a useful diagnostic practice that can help characterize code and validate code behavior.

Use comments to clearly label the sections of your code that are related to the parts of this assignment.

Part 1:

Create a program that will initialize the switch/LED port correctly and will ONLY toggle the red LED each time the infinite for-loop body repeats. When you run your code, what is the appearance of the red LED, and how can you explain what you are observing?

Use an oscilloscope or your AD2 to measure the frequency you find on pin `82` (this is `PAD15`, the actual pin on the chip the red LED is connected to). What frequency do you measure when the code toggles the LED as fast as it can? Include a screenshot of this measurement in your submission (label it `Part1`).

Part 2:

Using a variable, create a looping expression that will loop `500,000` times, doing nothing. Inject this code just after you toggle the red LED.

A dead loop like this creates a *blocking delay*. The CPU is busy executing your loop, and nothing else, so code is blocked for a period of time.

Run your code. What is the LED doing now? What type of variable did you use? Would `unsigned int` work or not? Explain why.

Part 3:

Decrease the loop count to iterate only `1000` times. Run your code. Use the oscilloscope or your AD2 to measure the frequency you find on pin `82`. What period and frequency do you find on this pin? Experiment changing the type of the variable used in the looping statement making sure you try both `unsigned int` and `unsigned long`. Measure the period and frequency for both cases. Do the period and frequency change? If so, how could you explain the change?

Part 4:

Now that you know (by measurement) the delay generated and the loop count, do some math and adjust the loop count so that the delay time is as close to `1[ms]` as you can get. Note: this should not be a trial-and-error type of activity. Remember that the delay code generates only half the period seen on the signal (one delay to turn the pin high, one delay to turn the pin low), so if you have done this correctly, you should see a signal close to 500[Hz] on pin `82`.

Do math here, **sketch the waveform and label it Part4** with period and frequency information: