mbed Lab 5: Excuse me, but haven't we been here before?

CS 220

Introduction

For this lab, you'll use a pushbutton switch and the KL25Z's touch slider and tri-color LED (the "haven't we been here before" part of the lab title) along with some of the mbed system's interrupt facilities (the "excuse me" part of the lab title) in order to "simultaneously" control three "threads" running on the KL25Z. This will shed a bit of light (Pun intended!) on how an operating system manages its trick of running hundreds of threads "simultaneously" on a computer that has only a handful of CPU cores. Each of the three threads controls one of the tri-color LED's individual LEDs. One of the threads simulates a CPU intensive process, consuming five seconds of CPU time before relinquishing the CPU to the other threads. This thread controls the red LED. The remaining two threads are user-oriented and would be considered to be I/O intensive threads by an operating system. Operating systems are usually designed to respond quickly to I/O intensive threads, giving them priority over CPU intensive threads. One of these I/O intensive threads uses the switch to control the blue LED. The other thread uses the KL25Z's touch slider (the gray-ish rectangle at the end of the board opposite from the USB connector end of the board, — slide a finger along the slider) to control the brightness of the green LED.

Lab Objectives

1. Learn the fundamentals of using a CPU's interrupt facilities to allow an operating system to re-gain control from a running program.
2. Use the `InterruptIn` interface to get the KL25Z to immediately respond to external events.
3. Use the `Ticker` interface to sample an external device's state at fixed time intervals and take an action.

Lab Parts List

1. KL25Z board and Quick Reference Card
2. USB cable
3. Breadboard, with the normally-open momentary pushbutton switch already mounted on it.
4. Two wires

Lab Procedure

Read each of the following steps **completely** before acting on them.

1. You won't need to apply power to the KL25Z until later, so set the USB cable aside for now.

2. As you did in Lab 1, wire one side of the pushbutton switch to one of the KL25Z's ground pins and the other side of the switch to KL25Z pin `PTA1`.

3. Go here and import the lab's starter code into the compiler. Remember to tick the Update box in the import dialog.

4. Open `main.cpp` and study it. Pay particular attention to the `while` loop in `main()`:

   ```
   while (1) {
      cpuIntensive();

      // Use the touch sensor to scale the green LED's brightness
      // to be between LED_ON and LED_OFF.
      greenLed = LED_ON + (LED_OFF - LED_ON) * (1.0 - tsi.readPercentage());

      if (button)
         blueLed = LED_OFF;
      else
         blueLed = LED_ON;
   }
   ```

   This loop simulates three threads meant to execute "simultaneously" — a CPU intensive thread that runs in five second CPU bursts and two I/O intensive threads, one associated with the KL25Z's touch slider and the other associated with the button.

5. What do you think will happen if you interact with the button or the touch slider while the CPU intensive thread is running?

   Try it out — connect your KL25Z to the host system, compile and download the `Interrupts` program, and run the program.

6. Notice how the CPU intensive thread "hogs" the CPU. Typically, an operating system gives such threads a lower priority than I/O intensive threads. You'll now use some interrupt techniques to give the I/O intensive threads priority over the CPU intensive thread.

7. Go here and look at the documentation for the `InterruptIn` interface.

   In your program, change `button`'s type to `InterruptIn` and comment-out the code acting on `button` in the `while` loop.

   Looking at the `InterruptIn` interface documentation, you're going to use the `rise()` and `fall()` functions to attach functions that are run when the button is pressed (resulting in a fall event) and when the button is released (resulting in a rise event). These two functions, which you'll need to write, should control the blue LED in such a way that it lights when the button is pushed and is dark otherwise.

8. Now, take a look at the documentation for the `Ticker` interface, available here.

   In your program, comment-out the code acting on the touch slider in the `while()` loop.

   Create a variable of type `Ticker` and attach a function (you'll need to write this function, too) that samples the touch slider and updates the green LED's value. Choose a sampling interval that is short enough to be completely interactive (in human terms), but not so short that the sampling function ends up hogging the CPU.

9. At this point, the only statement in your `while()` loop should be the call to `cpuIntensive()`.

   Compile, download, and run your program. How is its responsiveness to the button and touch slider now?

10. When you're finished, carefully remove the two wires from the breadboard, placing them back into the clear plastic bag.