Milestone #1 (11:00am section)

Due: To be shown in class on 9/6 & 9/8.
Your team will demonstrate the following implementation:

1. Create a Harmony configuration that uses FreeRTOS. Do not use a task delay for your tasks (aka "apps" in Harmony). You should create a single task. Your implementation will continuously output a character to a PIC USART of the PIC, where the value changes every 50 milliseconds. The values will rotate through the ascii values for the characters of your first and last names (with a space between them). You will need to use the Harmony API to configure the USART and to write data out to the USART. The USART does not need to be interrupt-driven. (For right now, eventually all device interfaces will have to be.)
2. You will use a FreeRTOS timer to "drive" the 50 millisecond timing. The timer callback subroutine should send a message to a message queue, where that message queue is read by your single task. That task should block on that message queue until a message is received (portMAX_DELAY).
3. Define a debugging scheme that enumerates “events” in your code. For example, the sending and receiving of messages from (2) should be events. The events should be defined in a “debug.h” file. You should have a standard scheme that your team uses to output these events as eight-bit values that can be viewed on the logic analyzer. You should be able to show that you can configure the logic analyzer to trigger on a particular event. You may either define preprocessor macros to handle events, or define functions in a “debug.c” file to handle the events. Obviously, there should be two parts to this code; something initializes the GPIO lines, and then the “event handler” routine. You should be able to easily redirect the output, for example to the logic analyzer or as a UART stream that might go over the WiFly.
4. The timer callback function should be in its own file. Do not put it in the same file with the code for the task (e.g., app1.c & app1.h if you used the name "app1" -- I'll assume you used app1, but it is fine to use something different).
5. The timer callback function should *not* include app1.h. It should only include app1_public.h.
6. app1_public.h should contain the declaration of the routine that puts a value into the message queue that is created in app1.c. For example, you might call the routine int app1SendTimerValToMsgQ(unsigned int millisecondsElapsed).
7. Check the return code of any subroutine call that returns a value. If you get an error code, call a routine (that you create) that brings everything to a halt in an obvious way.