ECE 315 - Computer Interfacing

Assignment #1

Due: In the ECE 315 assignment box at 15:45 on Wednesday, Feb. 5, 2020

---

1. Briefly explain the similarities and differences between a hard real-time embedded system and a soft real-time embedded system. Why is is useful, when selecting an implementation strategy, to distinguish between hard and soft real-time embedded systems?
2. Using the latest, fashionable technical jargon is an inexpensive way of appearing technically "up-to-date" to your colleagues, your managers, and your customers; however, such terms are sometimes actually quite poorly defined. Consult reputable Internet resources and provide working definitions for the following four computing architectures: cloud computing, edge computing, fog computing, and mist computing. For each term provide your assessment of how well-defined each of the terms is. Also for each term give the advantages that the computing architecture is supposed to provide over existing practice.
3. There are now many situations where embedded computers provide assistance to human operators. Most typically, however, the actions of the human operator must be given priority over the recommended actions of the embedded systems. However, in some situations it may be warranted to allow the embedded system to lock out and override the control of the human operator. Give three distinct examples of such situations, and for each explain why it would be reasonable for the embedded computer to have priority over the human operator.
4. Briefly describe the life cycle of a properly handled pointer value in C. Begin by explaining the different ways in which a pointer value is correctly created. Then explain the various ways in which a pointer value can be correctly modified (not just overwritten). Finally, explain when it is safe to discard the last copy of a pointer value.
5. In the context of microcomputers, what is meant by a "memory leak"? How can using software objects be used to help ensure that memory leaks and initialization errors are avoided in embedded systems?
6. What is meant when it is said that an embedded system is vulnerable to buffer overflow? What can cause buffer overflow and why is buffer overflow a serious problem? What would be the main strategies that a designer could use to avoid the possibility of buffer overflow?
7. Briefly describe the various ways in which the MicroC/OS real-time operating system (RTOS) has been designed and implemented to maximize its portability across different microcomputer platforms.
8. Sufficient idle time must be present in real-time systems. What would happen to the handling behaviour of externally and internaly initiated events by an embedded system if the amount of idle time were reduced below a safe percentage of the CPU's execution time? What sort of problems could one expect to see when the idle time started to become insufficient?
9. The single-threaded, non-preemptive loop architecture is recommended when designing hard real time embedded systems that are implemented on "bare metal", without an operating system. In such an architecture the execution times of the single thread are controlled by a hardware timer. First, briefly describe the major elements of this architecture. Then briefly explain how adopting this architecture simplifies the problem of meeting each of the following kinds of real-time specifications: (a) the maximum response times, (b) the maximum variability in the response times, (c) the accuracy of repetition frequencies, (d) the maximum variability in the repetition frequencies, and (e) the control of degradation behaviour.