The Art of Debugging

Description: There can be no design implementation without debugging. When designing a complicated system, just as much time will be spent figuring out why the design does not work as intended, as the designer put into coming up with the design in the first place. This is particularly true for novel or cutting edge work, which is often the case in research and science. But, most coursework solely focuses on the design aspect, leaving the art of debugging up to the student to discover on their own. This course seeks to fill this knowledge gap with theoretical and heuristical approaches, and give hands-on experience with projects of the students' choosing. The main focus will be on analog and digital electronics and microcontroller programming, but the concepts will be applicable to a wide range of problems.

Prerequisites: Familiarity with basic electronics.

Units: 6 (2-2-2)

Schedule: One in-class session a week of 2h, with optional lab hours.

Grading: Pass/Fail, although entirely open to it just being a lecture series with no credit given as well. If for-credit, then a passing grade is given for those who attended the lectures and participated in lectures and labs. There will be a few homeworks and readings, but the work will mostly be debugging circuits as a group.

Syllabus: Topics will include:

1. Choosing a methodology: divide and conquer, decision trees, first principles, and shotgun methods. Breadth versus depth in your approach, and the tradeoffs with respect to time and materials.

2. Tools of the trade: what is the correct tool for the job? Multimeters, oscilloscopes, LCR meters, function generators, spectrum analyzers, JTAG, and logic analyzers.

3. Rework: what is the best way to remove and reinstall broken components? Which technique should you use given time and cost; prototype versus antique equiptment?

4. Components: what are the common failure modes of capacitors, resistors, transistors, etc., and how to detect whether they have occured. What parts are routinely underspecified in commercial designs, and therefore more likely to fail? How does heat, humidity, and time effect parts?

5. Schematics: following signal flow through a circuit, locating parts on a board.

6. Reverse engineerng: what to do when you do not have documentation for your board.

7. Design for debugging: how to save your future self a lot of trouble. Schematic and board annotation and board layout and extra components (jumpers, current sense resistors, etc.). Documentation of design decisions.

8. Online resources: where to find information on commercial boards, and where to ask for help. What is the best way to ask for help on a forum to get good responses?

9. Esoteric issues: interference, sporadic issues, mechanical stress, high speed signals, mismatched impedances, etc.

10. Microcontrollers: how to get the microcontroller to tell you what it's actually doing without modifying what it's doing. Decoding digital signals, using pin toggles, and understanding the pitfalls of interrupts and watchdog timers.