Aaron Schulman - Teaching Statement

Educating for the imminent confluence of Computer Science and Electrical Engineering.

During the next ten years we are facing a new challenge in educating students about computing. Computer Scientists are finding that the abstractions of hardware that have served us so well for many years are now reaching their limits. This is driven by the increased demand to improve the battery life of mobile devices, and the increased performance requirements of computing systems. What's more, the computing industry is changing: the growth of desktop and laptop computing is slowing and the growth of embedded systems—tiny computing devices that provide digital computation on the analog signals that are at the core of everyday objects—is greatly increasing.

Electrical Engineers are facing a similarly changing landscape. The main application domain of new innovations in electronics is in computing systems. For example, the incredible minification of sensors with MEMS technology has driven innovation in computing devices such as MEMS accelerometers in smartphones, MEMS gyroscopes in video games controllers, and MEMS barometers in fitness trackers. Hardware developers only reach wide adoption of their innovations if they provide an intuitive interface for the software developers of the systems that these devices will be embedded in.

My goal as an educator is to make it so the next generation of students in these fields can communicate effectively. CS students should understand the low-level details of hardware and signals that are hidden behind software abstractions. Electrical engineering students should understand the low-level details of computer architecture and compilers. I aim to give both sets of students a more complete picture of what is going on under the hood of a computing system so they can innovate together.

I have found that an excellent way to teach across the divide is through embedded systems. Embedded systems are well suited because they combine computation with signals and circuits. I have found that with CS students, I can focus on teaching signals and circuits while they apply their existing knowledge of computation, and vice versa with EE students.

During the past three years, I designed and taught the first embedded systems courses for CS undergraduates at Maryland and EE undergraduates at Stanford. In both cases embedded systems was an effective tool for bridging the gap between hardware design and software development (discussed in detail below).

My teaching philosophy can be summed up in four statements: (1) **teach material in students' comfort zone**, (2) have them **build something that is useful**, (3) **give them room to break something** and learn from their mistakes, and (4) require them to use what they developed to **perform scientific experiments**. The rest of this statement details how I apply this philosophy to my in-class teaching as well as my course projects.

Bridging CS students with EE: My approach to teaching EE concepts to CS students is to first encourage them to ask questions about the hardware that provides the functionality that they use in software. For example, I enjoy asking my CS students a simple question: what hardware does computer use to keep track of time? I then explain the electrical phenomenon by relating it to something from their everyday lives. For oscillators, I describe a tuning fork and how its oscillation settles at a particular frequency after it is banged against an object. Next, I connect the phenomenon they are familiar with with the electrical one. I describe how the oscillator in their computer is actually shaped like a tuning fork, and is essentially banged by electronics to start it up. Finally, I demonstrate the electronic behavior that I described in class using common electronics testbench tools such as oscilloscopes, logic analyzers, and volt meters. These tools enable the students to experience the behavior of low-level electronics first hand.

I create projects that are split up per hardware unit in an embedded system. Examples of this include GPIO, DMA, digital logic busses, clocks, amplifiers, and filters. That way, as the students learn about how a piece of hardware works in class, they can build on that knowledge by experimenting with that same hardware in the lab. I endeavor to design the projects reliable hardware. The is because CS students are often discouraged by

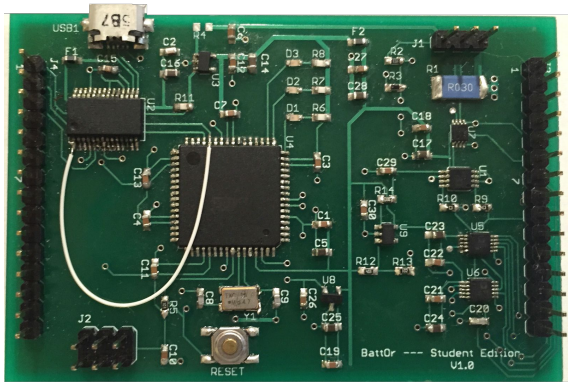
thought was a software bug turns out to be a hardware bug (e.g., poor soldering) because they do not understand yet that hardware can be buggy.

Bridging EE students with CS: My approach to teaching CS concepts to EE students is to extend their understanding of the hardware and abstract it back to how this hardware would be accessible from software. For example, with Memory Mapped IO, I start by talking about how peripherals have hardware registers that need to be configured by the software. Then I ask the students, how can you access registers from software? I then go back to the the hardware level and ask them, for each of those options how the electrical signal would be routed between the CPU and the peripheral?. Finally we settle on the memory abstraction as providing all the features that are needed to achieve setting the peripheral, and we talk about how it does not add significant complexity to the CPU because it already has a memory controller.

For EE students, I have a two-part approach to course projects (1) I focus on achieving software goals using the concepts that they have learned. (2) I have them build the full device from scratch so they experience the connection is between their hardware and their software through diagnosing hardware failures. For example, I one of the projects I assign is to implement the file stream for printf(). They have to implement getchar() and putchar() using a UART bus, then they assign that stream to be stdout and stdin, so the stream is used by printf(). The feedback I have received from the students is that this project is the first that truly made them aware of the connection between software and hardware.

Overall approach to course projects: I enjoy making course projects that are instructional and are also useful. As I describe in my research statement, I strongly believe that future electrical engineers and computer scientists will have to be very aware of energy efficiency. As such, I try to connect power measurement and energy management into my course projects. For example, when I was teaching the embedded systems course for EE students at Stanford, I realized that a simpler version of the BattOr power monitor would be an excellent first embedded system for students to build, and program. I simplified the PCB layout, so it would be easier for them to assemble, although I still required them to do surface mount soldering (a skill that will be invaluable to them for future prototyping and debugging).

The students built their own BattOr, which starts as a blank circuit board. They solder the chips on to the board, develop the firmware, debug the hardware and firmware, and use to run experiments. This is a board that one of the ten student teams created.



This is the BattOr connected to a smartphone's battery. The students used the BattOr that they built during the course for the final project where they compared the energy consumption of popular apps.

For the final project in the class, the students used the BattOr that they assembled and programmed to measure the power consumption of popular Android applications that they selected. This project incorporated everything that they learned in the class, as they were using their embedded system to find energy bugs in another embedded system, a smartphone. Applying what they learned from the power management lectures, they were able to create hypotheses for why these different energy bugs might be there. Several teams had particularly interesting results, and we will be polishing them and posting them on a public blog for others to evaluate and reproduce.

Mentorship experience: I have been fortunate enough to mentor for several excellent undergraduate and graduate students over my time in graduate school and my postdoc. Bartendr, my most cited paper, was a collaboration with an undergraduate student. I also have published three papers with graduate students that I have mentored at University of Maryland and Stanford.

I like to treat my undergraduate collaborators as full-fledged co-authors. I believe that they should be brought in intimately into the research process as early as possible. For example, soon after starting to work with me, I brought one undergraduate student with me to run experiments; driving and measuring the power consumption of different devices in low signal strength areas. In my experience I have found that engaging undergraduates can define the nature of a research group. I enjoy nurturing their excitement and interests.

My mentorship policy for graduate students is to find what makes them passionate about research, then I help them execute a project that builds on that interest. I am always excited to explore a new area that one of my students is passionate about. For example, for the past four years I have been mentoring a student who is interested in challenging the fundamental assumptions that researchers and practitioners have about the Internet. Initially I taught him what I knew about the subject, then he took the research a completely different direction that I would have never considered. The work we did together was published in the 2015 Internet Measurement Conference.

My primary reasons for applying for a faculty position are as follows: I want the opportunity to teach and mentor students, I want to be inspired by their seemingly limitless energy, and I want to help them grow into independent researchers.