TEACHING STATEMENT

Pengyu Zhang
Stanford University
pyzhang@cs.stanford.edu
web.stanford.edu/~pyzhang

I have an interdisciplinary background in EE and CS, having my bachelor in EE, MS/Ph.D. in CS, and postdoc in EE. During my education, I have had the opportunity to teach CS courses like Operating Systems and EE courses like Networked Embedded Systems. My background allows me to teach CS students to understand EE concepts and vice versa. This is how I designed and taught Stanford Networked Embedded System (EE107) as a co-instructor in Spring/Fall 2016 and Fall 2017. In this course, students learn how to build a system that is similar to Google Glass and Amazon Echo, and does continuous vision and audio sensing. Students start from building hardware, then write drivers for each hardware module, and eventually build a vision/audio application. This class is an ideal place to teach CS students EE concepts because students do lots of practice on using software to control hardware peripherals and understand how software and hardware are glued.

I followed three principles when teaching this class. 1) I teach students **hands-on experiences** in programming and debugging. The only way to learn how to build a computing system is building it by yourself. 2) I design the course project to make sure that the system built by students is **useful and solves a real-world problem**. Students spent more time than I expect because they are excited about building useful things. 3) I allow students to **fail**. Students learn more when they did something wrong because they will spend a lot of time figuring out what is wrong. I detail how I follow the three principles when teaching.

Hands-on experiences in programming and debugging

Students learn hands-on experiences by writing firmware for each hardware module, understanding the assembly code produced by the compiler, and using oscilloscope and logic analyzer to do joint software and hardware debugging. Students write firmware driver running on an MSP430 MCU and control several peripherals, including UART, SD Card, DAC, Microphone, and camera. I teach students how to interpret PCB board schematic, PCB board layout, and MSP430 MCU datasheet, and let them know how different hardware modules are connected and what MCU registers should they control. Students are excited when they can use C code to control the low-level registers of these peripherals and see how the software configuration impacts the underlying hardware behavior.

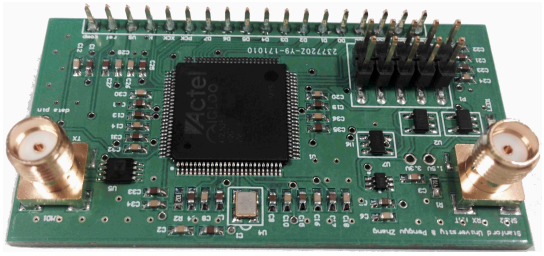
Sometimes the compiler does not translate the code as students expect. One example of this is the volatile variable. The compiler could knock off a section of code because the variable makes it think that this part of code will never be executed. I saw students struggling with this problem because they have no idea why part of the code is never executed. Then, I teach them how to generate the assembly code from the C code and how to understand them. By checking the assembly code, students realize that one section of code is knocked off without notifying developers. I also teach other examples where students find assembly code is very useful for debugging.

CS students are usually not familiar with hardware. So I also teach them how to do joint hardware and software debugging. I teach them how to use oscilloscope and logic analyzer to capture the signal produced by the hardware, and how to associate the hardware signal with the software they just wrote. The back forth debugging between software and hardware helps students obtain a deeper understanding of the system.

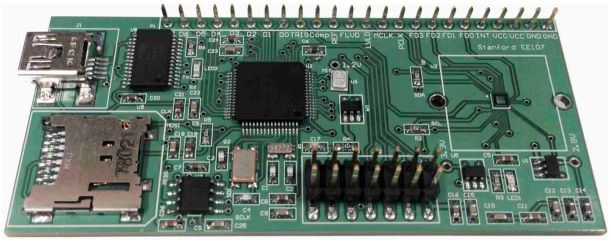
Build a system that solves a real-world problem

One problem of many system courses is that the system built by students is a toy system and cannot be used to solve a real-world problem. As a result, students' passion fades quickly when they are struggling with low-level bugs. I tried hard to avoid this problem when designing the EE107 course project. I want to make sure that the system built by students is useful and can be used to solve a real-world problem. Several students told me that they chose to take and continue this course just because this course project looks cool and useful.

The EE107 course project is motivated by the observation that vision and audio systems consume lots of power. Therefore, they are either connected to the wall for power supply or have a short battery lifetime. One example of this is Google Glass, which



(a) The backscatter radio board built by students. Students programed the FPGA to enable backscatter communication with 802.11b WiFi devices. One group of students re-designed the codeword translator running on the FPGA and obtained 1Mbps data communication rate, 3× higher than the state-of-the-art.



(b) The camera MCU board built by students. Students wrote the MCU firmware to fetch and process the image data from the HM01B0 low-power camera. One group of students was able to leverage cloud to do accurate face recognition while the camera MCU board only consumed 5mW of power.

Figure 1: Backscatter radio board and camera MCU board built by students in Stanford EE107 class.

only lasts for 38mins when doing continuous face detection. In the course project, we try to solve this problem by building a vision and audio sensing system that is ultra-low power and can last for one month when powered by an AA battery.

Figure 1(a) and Figure 1(b) show the two hardware boards built by students. One is the microcontroller board and the other is the backscatter radio board. Students write microcontroller software to sample the microphone and obtain the image data from the camera. Then, the vision and audio data is transferred to the cloud via the backscatter radio. To reduce the device power consumption, the audio and vision applications are hosted in the cloud rather than on the device. Taking advantage of the backscatter radio we designed in our research, one group of students can continuously run the system with a power consumption less than 4mA (25 days), much better than the state-of-the-art. Most of the students are motivated by this ambitious goal and work hard to build and optimize their systems.

Learn from failure

Another important principle I followed in teaching EE107 is we should allow students to fail. Students actually learn more because they will spend lots of time figuring out what is wrong. When students started working on the project, I gave them bare-bone PCB boards. That means students need to solder surface mount components on the PCB boards by themselves. Many students do not know how to do surface mount soldering, and they made lots of mistakes. Many of them actually broke at least one PCB board for various reasons. For example, one student detached one pad from the PCB board because of overheating. Another student bent one pin of the MCU because of incorrect wick usage. I help each student and correct his or her mistakes. Most of the students eventually master the skills of doing surface mount components soldering and have a working PCB board.

Courses I can teach

I have a background in both EE and CS. For the undergraduate level, I am comfortable with teaching the following CS courses: operating systems, distributed systems, computer networks, data structures, and programming languages. I would like to teach the following EE courses as well: embedded systems, wireless networks, signals and systems, digital signal processing, and wireless communication systems. For the graduate level, I would like to teach seminars discussing research papers in the area of Internet-of-Things and Cyber-Physical Systems. We will cover various topics, including but not limited to networked systems, energy-efficient computing, machine learning for IoT and CPS, and security.

Mentoring undergraduate and graduate students

I have worked with many undergraduate and graduate students when I stay at Stanford, UMass Amherst, and Tsinghua. I am working closely with Colleen Josephson and Manikanta Kotaru from Stanford. Both of them published papers on CoNext 2017. I also worked with Pan Hu and Mohammad Rostami from UMass Amherst and we published three papers in Sigcomm. I also worked with several undergraduate students from Stanford, UMass Amherst, Tsinghua and Peking Universities. I spend lots of time discussing research ideas with students. I am also willing to help students debug one line of code or fix hardware bugs. The reward of working with them closely is our productive research.

To summarize, I love teaching EE and CS system courses and mentoring students. **The major reason for me to choose a faculty career is working with these brilliant young people and innovating the state-of-the-art. They are the sources of innovation, and they can push the frontier of research.**