Interactive Lectures for Embedded Systems Programming

UNIVERSITY OF OULU

D.Sc.(Tech.) Teemu Leppänen, Center for Ubiquitous Computing, Faculty of ITEE

1. Course: Introduction to Computer Systems (2017)

5 ECTS, 150+ students, from 2nd year B.Sc. to 5th year M.Sc.

Students from several faculties (2/3 computer science and electrical engineering, 1/3 other engineering fields, mathematics, physics, wellness technology , etc).

Course topics: basics of computer architecture and CPU operation, data representations, basic I/O, interrupts, basics of the C programming language and embedded systems programming.

Course implementation:

1. Lectures and ~25 programming exercises (one month).

2.Course project (in pairs) for a real-world embedded Internet of Things device (one to two months).

2. Why interactive lectures?

The full course material (lectures, exercises, course project, etc.) is available in online, but classroom teaching is needed due to varying knowledge of students and generally challenging topics. The aim is to **move forward from the one-directional teaching during traditional mass lectures:**

- To **motivate** the students with different backgrounds. Address the varying backgrounds and levels of previous knowledge.
- To **activate** students with videos and lecture exercises. Initiate discussions.
- Get **immediate feedback** on how the students understood the presented topics.

3. Implementation

The sets of exercises were created using **Socrative** Web application:

1. For each lecture, teacher created a set of 1-7 questions.
2. Questionnaires were presented at the beginning of a lecture.
3. Students, during the lecture, submitted anonymous answers.
4. At the end, correct answers were presented.

4. Questionnaires

First set aims to find out what is the **starting knowledge of the students** regarding programming in general and embedded systems programming.

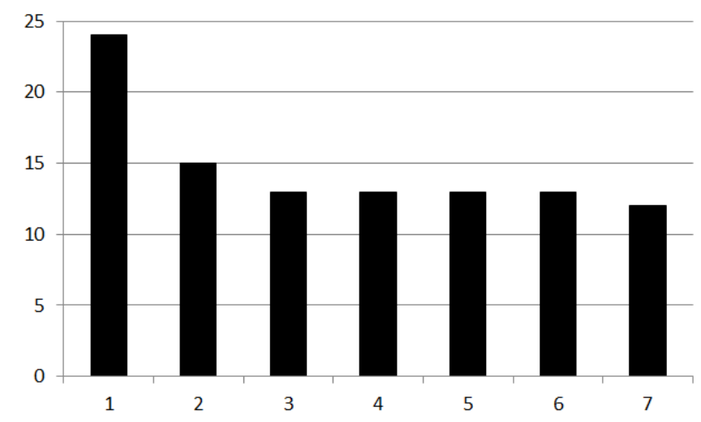
- *What is your degree program?*
- *When have you last programmed a computer? A: yesterday, B: one month ago, C: one year ago, D: never.*
- *Which programming languages are you familiar with? A: C / C++, B: Python, C: Java, ..*
- *Have you used any of the following programming tools? A: gcc, B: Python interpreter, C: Arduino IDE, D: github, ..*
- *What is your relation to programming? A: I love it, B: I manage the programming courses well, C: I am interested, D: I only take this course because I have to.*

Lecture questions focused on topics that were **identified difficult by the previous classes**: number systems and data representations, bitwise operations, C language syntax and pointers and arrays. Also, some general questions were presented to initiate discussion.

- *Convert binary numbers to signed / unsigned decimal and hexadecimal representations?*
- *In two's complement form, add..*
- *What are the results of the following bitwise operations?*
- *What are the smallest C variable types that can represent these values..*
- *What the data values are retrieved by using the following pointers..*
- *Fix the following C language sentences..*
- *Which architecture you like better? A: Harvard, B: von Neumann*

5. Results

Example: Out of 38 students visiting the exercise page, one third tried to answer all seven questions.



Student feedback:

" Lecture exercises are useful"
"Yes, but too many exercises per lecture"
"Yes, but disrupted me from following the teaching as I try to solve them"

6. Conclusion

- Videos and lecture exercises were found useful by the students.
- Exercises should not be laborous and not too many per topic.
- Exercises are better to present immediately after each topic, not the whole set in the beginning.
- Socrative was easy to use by both the teacher and students.
- Automatic answer checking in Socrative is difficult for programming exercises.

Poster presentation at Opintori 2018, University of Oulu, Finland, May 17th 2018.