A Maker Approach to Computer Science Education: Lessons Learned from a First-Year University Course

Dag Svanæs

Department of Computer and Information Science,
NTNU
NO-7491 Trondheim, Norway
dags@idi.ntnu.no

Abstract. We report from a one-semester introductory course for first-year computer science students where Arduino, robot programming and app development with Processing was used to foster engagement and creativity. The main learning objective for the students was to learn basic hardware and software skills, while at the same time motivating for further computer science courses. The course had a total of 250 university students with two teachers and ten teaching assistants. The major challenges were related to creating exercises, educational material and a physical work environment for the students that allowed for creativity in the spirit of the maker culture. We learned that much effort must be placed on making a high number of well-documented small and complete examples that the students can use as a starting point in their projects. We also learned that the teaching assistants should themselves have spent much time experimenting with the actual technology.

Keywords: Maker culture, Arduino, computer science education, robots.

1 Introduction

The computer science education at the Norwegian University of Science and Technology (NTNU) has traditionally been fairly theoretical in the first two years, with a focus on basic courses in mathematics and programming and little time for the students to explore technology or apply the theory on real-life problems before later in the study. This has its roots in the German engineering education of the early 1900s, with its emphasis on a natural science approach to engineering. The earliest engineering curriculums at NTNU were written in the 1910s and 20s by professors who had their training in this tradition, and the wisdom of the "theory first" approach to engineering education has not been challenged until recently.

In [1] Felder contrast this traditional *deductive* approach to engineering education with the *integrated* view:

- "**Deductive (Fundamentals --> Applications)**: Begin the first year with basic mathematics and science, teach "engineering science" in Years 2 and 3, and get to realistic engineering problems and engineering practice in the capstone course.
- **Integrated**: Introduce engineering problems and projects starting in Year 1, and bring in the math and science (and communication and economics and ethics) in the context of the problems and projects." (Ibid, p. 3)

Over the last three years the computer science curriculum at NTNU has been through a major revision, aiming for a more integrated model. One of the major motivations for making changes has been an increasing dropout rate in the first two years of the study, and a general dissatisfaction among the students with the strong emphasis on theory early in the study.

One of the changes implemented in the new curriculum is a new project-based "programming laboratory" course in the second semester of the first year. The main learning objective of this course is for the students to learn basic computer hardware and software skills, while at the same time motivating for further computer science courses. The course was run for the first time this year, and we made use of Arduino [2], robot programming [4], and app development with Processing [3] to foster engagement and creativity. The course was to a large extent inspired by the maker culture.

We will here present the structure of the course and some preliminary findings.

2 Course Structure

The course was named "Programming laboratory 1" to signal that it is a hands-on course. It is followed by a course "Programming laboratory 2" in the following semester that builds on this course. The course is mandatory for all 250 first-year computer science students at NTNU, and runs for the full 14 weeks of their second semester. The students participating in the course have learned basic programming principles in a first-semester course, but their experience with programming is limited.

Two teachers and ten teaching assistants were allocated to the course this year. The course was organized as teamwork in the lab, combined with one two-hour lecture per week. The course is 7.5 ECTS, corresponding to 25% of their teaching load. We consequently expected them to use at least one full day on the course per week, although each team only had access to teaching assistants four hours per week.

The course has no exam, and the grading was passed/not passed (P/NP) to put less pressure on the students. The course had four mandatory exercises throughout the semester that all had to be successfully completed to pass.

3 The Exercises

The two first exercises were individual, and two last were done in teams of five students. Starting out with two individual exercises was done to ensure that all students got the basic skills in electronics and Arduino programming to enable them to take active part in the following two team exercises. Our motivation for doing this was previous experience from programming courses with team exercises where we often found that the teamwork was taken over by one or two students with experience in programming, alienating the students with less programming experience.

The four exercises were:

1. Street signal (individual, 2 weeks).
2. Musical instrument (individual, 3 weeks).
3. Robot competition (team, 4 weeks)
4. Xbot - A robot and its world (team, 5 weeks)

3.1 Hardware, Software and Building Material

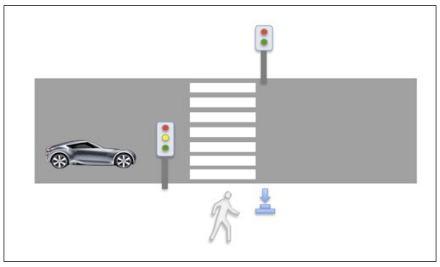
The learning material for the course consisted of an Arduino starter kit for each student and an electronics and robot kit for each team. In addition each team got simple cardboard building material for exercise 4.

The starter kit consisted of an Arduino, a breadboard, and basic sensors and actuators. The team kit contained additional sensors, actuators, and Bluetooth communication modules, in addition to a Zumo robot [4] with an Arduino on top.

To enable remote control of the robots from the students´ mobile phones, we made an Android app that executes Processing code on the mobile and communicates with the robot through Bluetooth.

3.2 Exercise 1: Arduino Street Signal (individual)

In exercise 1 the students were asked to use their Arduino starter kit to program a pedestrian crossing with a street signal.



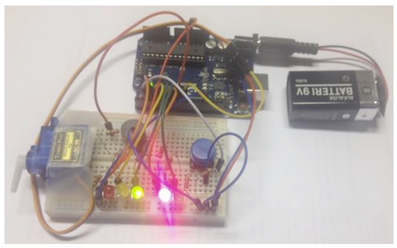


Figure 1. Street signal illustration (left) and Arduino implementation (right).

Figure 1 shows to the left the illustration used in the exercise text, and to the right an implementation of the street signal. In this implementation a servo is added to stop the pedestrians and a buzzer is added for sound. The feedback from the students was that they liked the exercise very much because it was easy to understand and because it was a real-world problem.

3.3 Exercise 2: Arduino Musical Instrument (individual)

In exercise 2 the students used the Arduino starter kit and sensors from the team kit to program an Arduino musical instrument of their own choice.

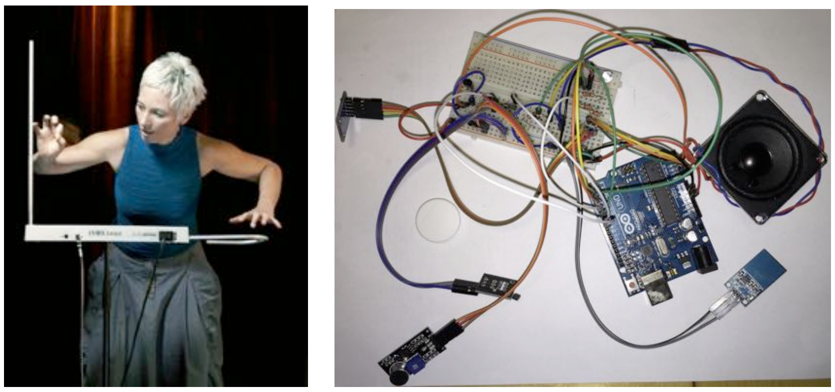


Figure 2. Theremin (left) and Arduino musical instrument (right).

Figure 2 shows to the left a picture of a Theremin musical instrument that was described as an inspiration. To the right is an implementation of an Arduino musical instrument using a combination of sensors (touch, tilt, magnetic and sound).

The students liked the exercise, in particular that the problem was fairly open and allowed them to be creative in choice of sensors and behavior. They further appreciated that we had provided a number of working examples on the course wiki for the sensors, together with examples of how to program tones and sounds through a loudspeaker.

3.4 Exercise 3: Zumo Robot Competition (team)

In exercise 3 the students worked in teams and were given the task of programming their Arduino Zumo robot [4] to compete against another robot in the zumo ring. The zumo robot is actually an Arduino shield on its head. We stages competitions between the 55 teams, leading to final rounds in one of the lectures. The two winning teams won cinema tickets for the team members. We used a slightly modified version of the international sumo robot competition rules [5]. The aim of the robot competition was simpoly to push the other robot out of the ring. Each match lasted for a maximum of two minutes, and the maximum weight was 500 grams.

The robot competition ran in two classes: (1) autonomous and (2) remote control. In the autonomous class the robots were programmed to compete with the other robot only navigating with sensors. For the remote control class we provided Bluetooth modules that enabled them to control the robot from an app that they programmed in Processing for their mobile phones. We had made a reference implementation of the app that they could use as a starting point for their own robot control app.

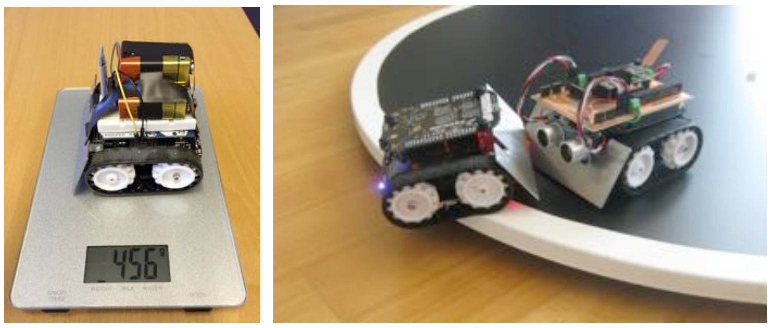


Figure 3. An Arduino zumo robot on a scale (left) and two robots competing (right).

The students found this exercise very inspiring and engaging, and the teams spent much time on perfecting their robots.

3.5 Exercise 4: Xbot - a Robot and its World (team)

Exercise 4 was more open than the previous exercise, challenging the students to be creative. The challenge was to make the robot into an "Xbot", a robot with some specific properties of the team´s choice. The teams were given an A0 cardboard (120x85 cm) that should be the Xbot´s world. In addition to the robot, the teams were required to use the Arduinos of the team members as stationary "helper bots".

Each team documented its Xbot in a 30 seconds video. The videos were shown in sequence on the final lecture of the course, and the students scored them using a modified version of the Kahoot system. The members of the two winning teams got a gift from a local semiconductor company.

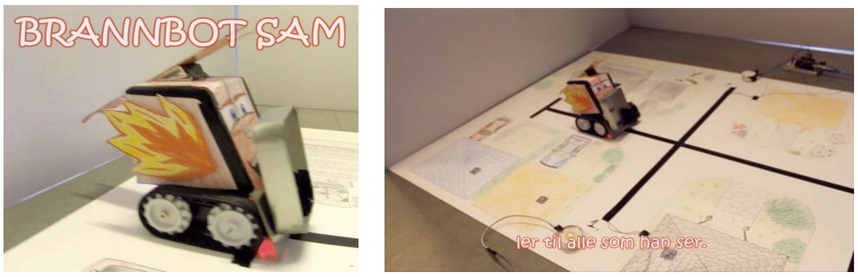


Figure 4. "Firebot Sam" (left) and his world (right).

In Figure 4 we see "Firebot Sam", the Xbot of the winning team. The robot was programmed to put out fires in candles placed on the side of its "world". It communicated through Bluetooth with helper bots that detected heat with temperature sensors. It then navigated by line-following to the right candle. A servo on the robot was user to tilt a metal top to put out the candle.

The students enjoyed working with the exercise, and a number of innovative Xbots were created. Some teams had problems coordinating the work among the team members though, leading to some frustration.

3.6 General Observations

One of our major challenges was to create exercises, educational material and a physical work environment for the students that allowed for creativity in the spirit of the maker culture. We learned that much effort must be placed on making a high number of well-documented small and complete examples that the students can use as a starting point in their projects. The planning and preparation of the exercises was done in the previous semester and involved developing code examples, setting up a wiki and buying electronics. Two teaching assistants were hired for this purpose in addition to approx. 20% of the work time of the two teachers and an engineer.

Although the teaching assistants had a background in programming, we gave them a crash course in Arduino in the previous semester and encouraged them to experiment on their own. As a consequence they became an invaluable recourse.

4 Conclusions

The presented course was inspired by the maker culture, and we have made extensive use of Arduino and the open source maker culture around this product. A combination of structured and open exercises worked well to teach basic skills, and at the same time opened up for creativity. The course was well received, and we plan to follow the same overall structure next year with some minor modifications.

Acknowledgments. Thanks to the other course team members Asbjørn Thomassen, Terje Røsand, Pia Lindkjølen and Inge E. Halsaunet.

References

1. Felder, Richard M. "Engineering education: A tale of two paradigms." *Shaking the Foundations of Geo-engineering Education* (2012): 9-14.
2. Evans, Brian. *Beginning Arduino Programming*. Apress, 2011.
3. Reas, Casey, and Ben Fry. "Processing: programming for the media arts." *AI & SOCIETY* 20, no. 4 (2006): 526-538.
4. *Zumo Robot for Arduino*. https://www.pololu.com/product/2506 (retrieved 9. July 2015).
5. *Unified Sumo Robot Rules*. http://robogames.net/rules/all-sumo.php (retrieved 9. July 2015).