Hands-on Introduction to Computer Science at the Freshman Level

Raghuraman Balasubramanian Zachary York Matthew Dorran
Aritra Biswas Timur Girgin Karthikeyan Sankaralingam
University of Wisconsin-Madison
karu@cs.wisc.edu

ABSTRACT

This paper details the creation of a hands-on introduction course that reflects the dramatic growth and diversity in computer science. Our aim was to enable students to get an end-to-end perspective on computer system design by building one. We report on a two-year exercise in using the Arduino platform to build a series of hands-on projects. We have used these projects in two course instances, and have obtained detailed student feedback, which we analyze and present in this paper. The instructions, code and videos developed are available open-source.

1. INTRODUCTION

Computer science as a field has changed dramatically over the past few decades. Today's embedded computing devices, like the Fitbit [5], provide more computing capability than a super-computer from 30 years ago. Such systems embody several computer science principles such as programming, compilation, operating systems, concurrent programming and hardware organization. Computer science education, especially freshman introduction to computer science, typically includes lecture based instructions and simplistic programming assignments and has not evolved much since the 1980s. CS252 from UW-Madison and CS312 [1] from the UT-Austin are representative example of many introductory CS courses. There are a few examples like Stanford's CS1C that make a radical departure. Such a departure may not be amenable to all institutions or undergraduate programs.

Our goal and motivation is to develop a freshman introduction to computer science course which, without major disruption to the existing curriculum, incorporates a hands-on building component. The aim is to develop material in which a team of students design and build a computer or computing system and perform an evaluation of the system. Many studies on teaching computer science have shown that construction and team exercise accelerate learning [7]. Others have shown that the material being "fun" is important [6, 8, 9] and the need for course projects to be challenging but not daunting. Our other goal is to open source instructional materials to increase availability. We intentionally avoid choosing projects that are specific to a single paradigm, like robotics, by providing hardware and software that can be assembled in a variety of ways to embody a computing platform. This allows students to choose from a diverse array of projects aligning with their natural inclination.

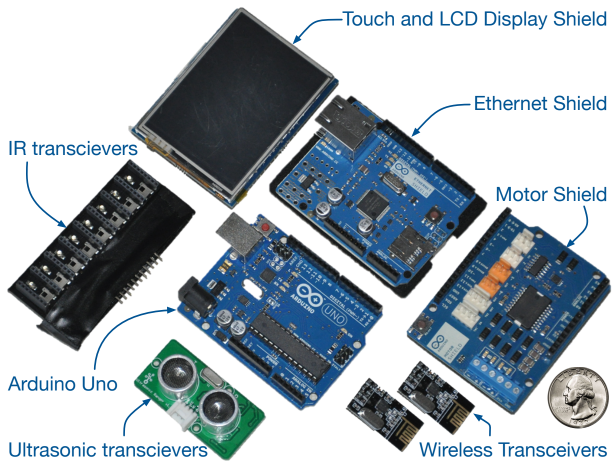


Figure 1: Arduino Uno board and select peripherals.

Due to the benefits of Moore's law and constantly reducing costs in the electronics industry, there are many viable platforms to choose from. Specifically, the Arduino [3] platform shown in Figure 1 provides tremendous hardware capability and a simple programming interface based on C language. The Arduino supports a vast array of "shields" or peripherals like ultra-sonic sensors, accelerometers, touch displays and wireless communication hardware that can be connected to the basic platform to build interesting systems. There is a widespread community of hobbyist computing resembling the 80s era of hobbyist PC building that is based on Arduino, which costs only $20. The Arduino platform has wide-spread open source community support, numerous do-it-yourself projects, and is easy to use, making it an excellent gateway to get students excited about computer engineering. The challenge from a curriculum or teaching perspective is to bring all this hardware and software into a cohesive platform for computer science pedagogy. This is particularly complex in a freshman course that is designed to introduce the core ideas of computer science spanning *bits, gates and binary logic* to *programming and computer organization*, The fundamental question is *how to design hands-on* *projects to enhance the experience of learning the fundamentals of computer science?*

In this paper, we share our experience gained from developing such a course at a public university with a large computer science yearly enrollment of over 2500 students. This paper is a report on the development of this curriculum spanning a pilot version we implemented in Spring 2012, the development of instructional material, assessing the material, and a second version of the course in Spring 2013 based on this new material. In each instance, we also report on detailed survey data and feedback collected from the students.

Summary of implementation The projects were added to an existing freshman introduction to computing course (CS252) which is based on the Patt and Patel [10] book and did not originally include any hands-on building component. In the Spring 2012 offering of the course, the strategy was to get something off the ground. We envisioned a few Arduino-based projects, wrote up a description in a couple of paragraphs, and provided students with hardware. We provided little supplementary instructional materials. One graduate student experienced in using the Arduino platform was available for one hour each week to help students. A 5% bonus credit was offered. We were pleasantly surprised by the outcome. Figure 2 shows the completed projects, which were all functional. All but one team completed the project they had picked. In total, 73 of 191 enrolled students did the projects. Based on their feedback, we then created detailed instruction material with extensive videos, code tutorials, and a week-by-week plan over the Summer and Fall 2012 semesters. The material itself was largely developed by 4 under-graduate students who took the course in Spring 2012. The course was then offered in Spring 2013. We provided the instructional material we developed, extensive daily office hours and support through a Piazza forum. This paper reports on this effort with the goal of providing lessons learned and insights on how others can reuse and enhance our material. We are making all resources publicly available at this URL: `http://www.cs.wisc.edu/vertical/arduino-252`.

Overall our findings are the following :

- Hands-on projects based on Arduino platforms can be effectively built as a 3 or 4 week project.
- The projects help teach and reinforce basic computer science concepts.
- Students feel the projects serve the pedagogical purpose and the experience gives them an enhanced understanding of the subject.
- The projects themselves can each be constructed with hardware costing between \$97 and \$254 as a one-time expense.

This paper is organized as follows. In Section 2 we describe how the pilot Spring 2012 version was developed, including a detailed explanation of the actual hands-on projects. In Section 3 we describe how we used student feedback to develop instruction material, and what instructional material was developed. In Section 4 we describe how the Spring 2013 version of the course was run based on this instruction material and student feedback. Section 5 addresses potential issues which may arise during the adoption of this material in other institutions. Finally, Section 6 concludes with thoughts on lessons learned.

2. SPRING 2012

We now describe the first instance of the projects offered in a *give it to them and have them build it* mode, covering logistics of implementation, a detailed description of the projects and the outcome and student feedback.

2.1 Mechanics and Implementation

We provided the students choice of five projects and seven weeks to complete them. We provided them some weekly milestones to aim for and a brief progress report was due each Friday. A final demo date was set at the end of seven weeks. Students were informed that "completing" a working demo would be sufficient and no other detailed grading was involved. We provided bonus of 5% course credit awarded for a completed project. We allowed project groups of two to six students, and facilitated some matching of students to projects based on programming experience. A total of seventy three students opted to attempt the extra credit course material. As the projects were optional, no lecture hours were devoted to teaching particular skills to complete the projects.

We provided a simple five minute demo of Arduino platform at the start of the projects. For all projects we provided a description and relevant pointers to Internet resources for the specific "shields" (sensors and actuators - see also Figure 1). We also provided detailed usage notes to get started on the following four topics since they spanned many projects: i) Installing the software and getting a simple blink LED demo to work. ii) Getting two Arduinos to talk to each other through a wireless transceiver. iii) Using Analog ports as Digital Ports. iv) Wiki links on how to use "motor shields" to control the speed and direction of the wheels for the projects with robots.

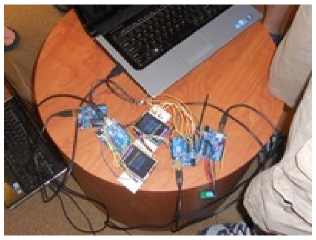
2.2 Description of Projects

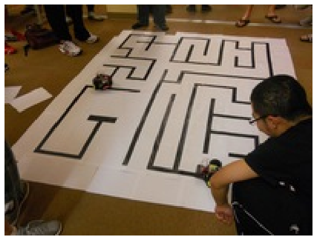
We describe the projects below and in the Appendix list the hardware required and cost per project. All software tools used are free and open source.

Maze Navigating Robots In this project, students built a robot that can auto-navigate in a maze to reach an exit. The robot followed an arbitrarily drawn black path in a bright background using a set of infrared sensors that were positioned roughly orthogonal to the line on the ground. The data generated by the sensors are analyzed by the Arduino processor which in turn controls DC motors to navigate the robot. The students were expected to develop a fully-functional line-tracking robot that could handle any path complexity. We also provided a pointer to a PDF manual of the infrared line follower kit which explained the wiring of the infrared sensors and their communication protocol.

Obstacle Avoiding Robot In this project, the students equipped robots with active range sensors to detect proximity to objects and programmed them to avoid obstacles. Students employed a commonly used method of depth extraction to calculate the distance of objects around the robot by using an ultrasonic sensor. A set of sensors that were positioned around the robot to act as its "eyes". Students wrote code that polled on the value returned by the ultrasonic sensors periodically. They performed calibration to obtain the thresholds to detect proximity. The obstacles were not necessarily stationary — the robot was required to periodically check its surroundings and react dynamically. We provided

Twitter


Wireless Tic-Tac-Toe


Maze Navigation Robot

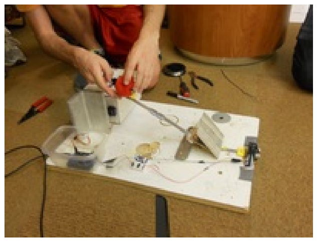

Angry Birds

Figure 2: Examples of the four projects students

the following pointers: (i) A Wikipedia link to how an ultrasonic sensor works and (ii) a page describing the ultrasonic range measurement module we used and sample code.

Tic Tac Toe In this project, students created a handheld game console. They were expected to design the game, the input and output interfaces (using a touch screen LCD). They also had to add a wireless module over two devices for a multi-player mode. The students formed three groups (i) the Game Designers - who implemented core game logic. (ii) the user interface team - who coded what was displayed on the screen. (iii) the wireless team - who setup communication between two Arduino boards. We provided links to the following online resources: (i) the hobbyist website [4] which lists instructions on how to use the nRF24L01 wireless transceiver and (ii) the touch screen shield manual [2].

Twitter In this project, the students created a door display that displayed messages of a twitter feed. They used a simple two line LCD (optionally a graphical color LCD) powered by an Arduino combined with an Ethernet shield that allows the Arduino to connect to the Internet. Once connected, students wrote code that used Twitter's APIs to read and write tweets. We provided links to an LCD shield tutorial and a link to the Arduino Ethernet library.

Angry Birds This was the most ambitious project. In this project students used a sensor to determine how far a target was, and hurled an angry bird plush toy at it. We suggested the use of an ultrasound sensor to detect the distance from the target. Students wrote code that used this to calculate the velocity at which the angry bird needed to be launched. We also suggested the use of a servo motor and some kind of spring mechanism and shaft assembly to control the distance to hurl the angry bird. In addition to links on the working of ultrasonic sensors and sample code, we also provided online coding manuals for the servo motor.

2.3 Outcome and Feedback

Outcome Of the 16 teams, all but one team completed their projects before the deadline and successfully demonstrated it on the final demo day. A total of seventy three students finished the projects. The Arduino platform proved to be powerful yet simple. The longest code contained less than 1000 lines. The quality of the projects was quite sophisticated as shown by the photos in Figure 2. In particular, the completion of the Angry Birds projects was impressive, as the students independently improvised the use of rubber-bands to act as springs to control the hurling distance. The grading of the projects was quite straight-forward – we felt all projects were complete and deserved full credit. We elaborate more on enthusiasm and other qualitative comments in the conclusion. Video demos are available on our website: `http://www.cs.wisc.edu/vertical/arduino-252`.

Feedback At the end of the seven week period, students were asked to fill out a survey (anonymously if they chose to) and the feedback was recorded. The top half of Figure 3 shows the response to five basic questions. Overall most students liked the project, but a significant number felt more help was needed and that the project could not be completed as a two person team. We used this feedback to develop instruction materials over the Summer of 2012. The most common responses on what changes the students wanted to see were: i) Better explanation of the Arduino libraries. ii) Detailed and week-by-week instructional materials and milestones. iii) Projects to be given out at the beginning of the semester rather than towards the end of the semester.

Learning In terms of pedagogy and learning, it was quite clear that students learned quite a bit about programming from these projects and felt this was a good introduction to programming and computer science — over 85% answered positively (52.7% say strongly agree) to the question: "Did you learn a lot about computer science from these projects?". Although, we could not keep track of improvements in exams or other homeworks because of the projects, many commented in the survey that they learned a lot about programming. The following are two representative student comments that capture the learning effectiveness: i) *"Arduino opens a wide range of fun projects that you can do, which allow you to/force you to understand how the chip, memory, buses, input.output types, computing, and general computing magic happen. It gets you hooked on learning."* ii) *"Creating our Arduino robot helped me understand more how computers work and function than assembly language did."*

3. INSTRUCTIONAL MATERIAL

The Spring 2012 feedback gave us insights into a few issues in project implementation. We understood that more instructional material was key for the projects to become part of core curriculum. We created a web based directory to house the project instructional materials. Within this web based directory, each project had its own respective page comprising instructional videos, documents on hardware assembly and wiring, pseudo code and basic code structure for each project and a chronologically ordered list of weekly milestones for an eight week plan. This structure provides the students with a user friendly environment to access the materials, which include step-by-step instruction manuals while maintaining the spirit of discovery that is central to these projects.

To expand the options available to the students, we added two more projects. One is a rendition of the famous "Pong" game using the Arduino micro-controller and a "GameBoy"

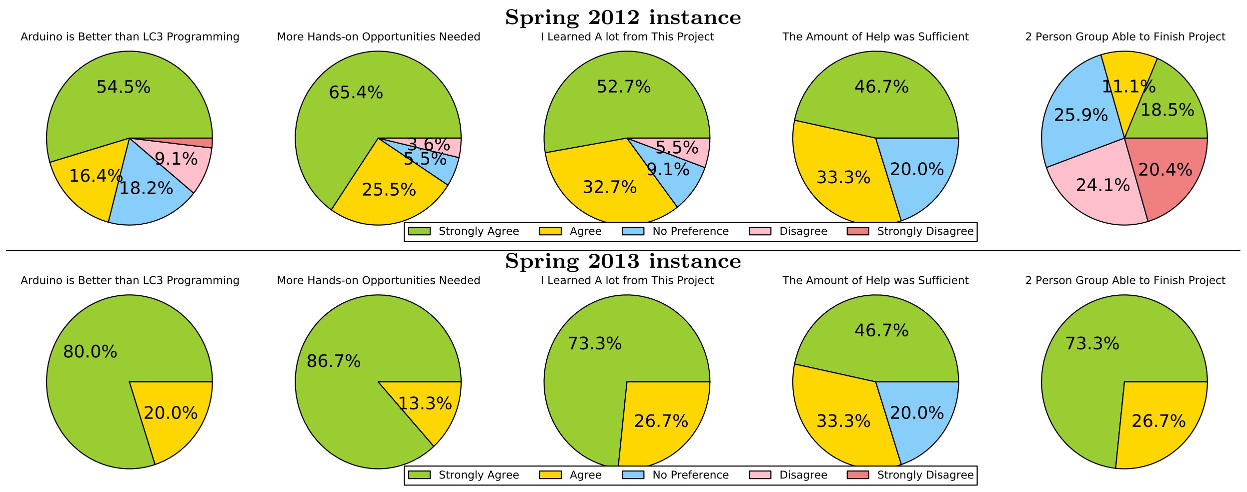


Figure 3: Summary of feedback. LC3 programming refers to the assembly programming originally in CS252.

like accessory that includes a joystick and two selector buttons. The second, was a "Word Scramble" that would give the user a scrambled word on an LCD screen. The user then uses a PS2-based keyboard to enter the unscrambled word. We also provided much more of the mechanical hardware required for the Angry Birds project as part of the instructional development.

Testing We tested the material using two undergraduate students from the Spring 2012 offering. They were requested to follow the step-by-step procedure located in the web directory for each project, just as a real student of the class would do. We monitored and recorded their progress and obtained their feedback at each step. Based on this feedback, we made many presentation changes and improvements.

4. SPRING 2013

We now describe the Spring 2013 offering which was based on the instruction materials created in Summer and Fall 12.

4.1 Mechanics and Implementation

In the Spring of 2013, we offered the students of the CS252 class an opportunity to work with the Arduino microcontroller. Out of 160 students enrolled in the class, more than 60 were interested. But we wanted to test in a more controlled setting with a known TA/student ratio - we randomly selected sixteen for doing the projects. Students were then divided into teams of two to three. Each team had the freedom to choose which of the seven projects they wanted to work on, with some guidance from us based on their skill level. The instructional material housed in the web-page was also made available to the students.

Once the team had chosen the project they wanted to work on, they were given a box containing the necessary material/equipment to accomplish the corresponding project. The goal set for all teams was to finish their assigned project in six weeks or less. In comparison to the Spring 2012 version in which we provided little or no support, we increased the support provided to the Spring 2013 batch. For instance, office hours with multiple (up to 4) TAs were made available every day for 3 hours. A "Frequently Asked Questions" page was also created for each project with answers to questions that students might have while building their project. The Q&A platform "Piazza.com" was also used for students to ask questions about their projects online. A total of 21 questions were asked by the students, and 100% of the questions were answered by the TAs with an average time of 15 minutes. At the end of the given period, the students were required to demonstrate their finished projects.

4.2 Outcome and Feedback

At the end of the given time-frame, all seven projects were correctly completed. Compared to the previous Spring, the final quality of the projects was significantly better, as many groups did more than what was initially expected. For example, the Pong game was implemented with an accelerometer enabling the game to be played by tilting the screen instead of using a joystick. Furthermore, the speed at which the projects were finished was greatly improved, as most teams had completed their project by week 4, and all teams had finished by week 5. It was also observed that teams attended less office hours than the previous year, getting most of the help needed using the online instructional material that was provided to them.

One major concern was to see whether the help provided during Spring 2013 gave better results compared to the implementation of the project in Spring 2012, where guidance was minimal. During the six weeks period of Spring 2013, students' feedback regarding the entire process was obtained on a weekly basis - they were asked to write up a short report each week on the instruction material and critique it. We analyzed this data and used this information to enhance the instructional material.

Overall, the weekly feedbacks we received were positive. However during the first two weeks, we noticed that the main complaints students reported was that the instructional material given to them did not meet their expectations. Students started their projects thinking that all information necessary to finish their project will be given to them. However, we intentionally limited the amount of information we provided – making sure that some browsing and thinking was necessary to understand the given material. Some information was purposely left out to encourage the teams to think by themselves and use their imagination to accomplish their project. By the time they completed their projects, students realized that the instructional material given to them was actually of great help, some teams went further by saying that it was too much help.

We did the same survey as in Spring 2012, and the results are shown in Figure 3. Overall the response was overwhelmingly positive. Except for one question, students answered with a definitely yes or yes. For question-4 about whether there was sufficient help, 20% were neutral, 80% were definitely yes or yes. To one of the most important questions of "Did you feel you learned about programming and computer science from the Arduino projects", 73.3% replied "Definitely yes", and 26.7% replied "Yes".

When asked what they thought of their project and how hard it was for them to build it, one student replied *"Not a lot–for the scope it was intended, I think this was a good project. I think most people with minimal to no prerequisite programming knowledge should be able to accomplish something decent given the code provided, yet there is still room for refinement for more advanced students."* Another student had a valuable comment saying *"I would have liked our project to be a little more complex. This could have been done by providing no skeleton code, allowing students to dig deep into the Arduino reference page."*

Between the comments stating that the projects were too easy, there was too much instructional material, and that all groups but one finished their project early, we can conclude that the provided instructional material is definitely sufficient in helping students complete these hands on projects in the required time frame. It is clear from the surveys and overall time spent completing each project that the result of instructional material is beneficial to the students. From student surveys we can also see that the overall feeling about having these projects as part of the course curriculum is positive. They believe that having a hands on experience similar to one they might have in the real world is an important factor in determining whether or not to pursue Computer Science/Computer Engineering as a major.

5. ADOPTION AT OTHER INSTITUTIONS

Our goal in writing about our experience is to facilitate the adoption of the material we developed in similar courses in other institutions. To that end we now comment on some related issues.

Instructor background and training When we first envisioned the course in Spring 2012, the instructor (the last author on this work) had no experience programming Arduino - neither hardware nor software but was able to ramp up quickly. Coding on the Arduino platform is straightforward to anyone proficient in C/C++ programming. Going through our online tutorials and other Arduino tutorials, one can become proficient with 10 to 20 hours of work. The potentially biggest impediment to implementing certain projects is minor hardware involvement. For example, a few Arduino shields need some soldering - intimidating and unfamiliar to some CS students and CS instructors. This is rare and in our case applied only for the Twitter project - simply avoiding it can eliminate this issue.

Scaling to larger classes We believe these projects can be considered for adoption in large classes in excess of 100 or 200 students. Much of the instructional material is web-based and hands-on input from instructors and TAs are limited. The main issues to consider are cost, physical space, and managing hardware failures - which we comment on briefly. Cost per project ranged from about \$100 to \$250 in 2012 (see Appendix for a breakdown). As the Arduino IDE works across platforms and is free, we left it to the students to use any machine they preferred. For physical space, we feel class room space is sufficient. We over-provisioned our inventory anticipating hardware failures. However failures were extremely rare – two wireless modules failed and a robot kit missed a gear component – and we were able to instantly replace them.

Role of the TAs and TA training In our pilot run, the TA's spent significant time defining the projects. One particular graduate student (the lead author on this paper) was available for consulting on hardware and software issues.

In the second run, the only role of the TAs was to be available for 3 hours per weekday and answer questions on Piazza. These were all under-graduate students who had earlier taken the course. Their training was simple — they were tasked with doing all the projects once and from this learned pretty much all that was involved. The hardware and software components of the Arduino platform are easy to use and learn. In our experience, teaching assistants with basic computer engineering knowledge were able to come up-to-speed with the environment and test run the projects with ease. In this second run, after the first two weeks, there was little to no demand on the TA time - only a total of 21 questions were asked/answered on Piazza.

In summary, if built upon the projects we have already developed, this could be implemented without TA support if need be. It will likely require office hours contact with the instructor, and his/her familiarity with the tools.

Alternatives to Arduino An important question to also consider is alternatives to Arduino, of which quite a few exist. They differ primarily in their software stack, hardware capability, diversity of "shield" and sensors, and cost. Platforms with Arduino-like software environments include mbed, OOPC, RobotC, and TinkerForge. Although they have similar hardware capability and cost, we feel community support and availability of natively supported shields is much less than for Arduino. Platforms with more sophisticated software environments like .NET, Java and Python include ioBridge, Make Controller Kit, Freedom Dev Board, Gadgeteer and Netduino. Finally platforms with full-fledged Linux/GCC include Intel Galileo, UDOO Raspberry Pi, and Beaglebone. The latter two categories introduce more barriers and complexity in interfacing with peripherals (and hence sensors), making system-building harder. In our view, Arduino's simplicity, cost, widespread availability of shields (hardware and software) and vast community support at this time is unmatched and alternatives do not appear compelling to us.

6. CONCLUSION

We have learned many things from the exercise of building these projects, instructional material and getting student feedback. Most importantly, we found that students liked the hands-on learning experience and appreciated this as a great way to get introduced to computer science. Second, we feel that with the materials in-place, and a pipeline of trained TAs providing support, it is quite easy to establish this as a continuous offering without significant recurring capital or human resource investment. As all necessary instructional materials are located in a web based directory, the bandwidth for providing these projects to 40 or more students has already been established. We conclude making a few other observations.

Diversity in technical background A challenge in any introductory course is the diversity in the technical background of the students. Exceptionally talented students sometimes had no computer science background. A few, even at the freshman level, had good understanding of programming. and had modest experience with high level programming languages like Java. Thanks to the the nature of the projects and the capability of the Arduino platform, we were able to create a from-scratch learning environment for students with little exposure and an open-ended structure to challenge students with prior knowledge.

Time spent and enthusiasm We observed that students were extremely enthusiastic about the projects, often spending significantly more time on these than the course homeworks or exam preparation. Some specific anecdotes: i) Two teams wanted to keep the hardware as they wanted to continue modifying and fiddling with them. ii) One team went above and beyond what was required by the Pong game, by adding accelerometers and making the game controller-free and instead simply tilt to play. iii) One team reverse-engineered the PS2 protocol and found a way to add a keyboard to the Twitter project.

Less is more? Compared to the Spring 2012 offering, The Spring 2013 version included a more detailed instruction material. However, we feel that the students learned much more in the Spring 2012 offering. Although it was more time consuming for the students, we think the students learned to learn — with a number of "Aha!" moments while discovering things on their own.

Computer science pedagogy In terms of computer science pedagogy, the projects convey many concepts in a hands-on and principled manner and are more effective than lectures with small homeworks. The students wrote programs and debugged them — getting introduced to a key skill very early on. Many technical concepts like the right hand rule, the fact that the maze can be represented as a graph, the concept of backtracking (and hence stack) are all naturally discovered by students as they implemented the maze project. In handling issues in the ultra-sonic sensor and infra-red sensor the students realize the concept of noisy data, they independently discovered the concept of sampling, averaging and noise correction. Most importantly, when doing the projects, they realize an important principle of system design – not to assume that things will work the first time. Students naturally discover these principles and hence this yields to be more effective than being taught with a contrived or abstract example.

Overall, we feel this exercise in CS252 has provided a project that can be done in a 4 to 6 week time-frame and added to an existing 2 or 3 credit course on introductory computer science. In our experience, it is an effective way to introduce computer science in a hands-on fashion to covey to freshman the excitement and diversity in the field.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for comments. Many thanks to Mark Hill and Guri Sohi for facilitating the projects in CS252 in Spring 2013. Thanks to Rebecca Lam and Newsha Ardalani for much support in Spring 2012 in managing the projects. Support for this research was provided by NSF under the following grants: CCF-0845751, CNS-0917213, and CCF-0917238.

8. REFERENCES

[1] "312 introduction to programming, ut-austin, cs, http://www.cs.utexas.edu/undergraduate-program/courses/312-introduction-programming."

[2] "Adafruit, www.adafruit.com."

[3] "Arduino, http://www.arduino.cc/."

[4] "Code for nrf24l01 modules, https://github.com/maniacbug/RF24."

[5] "Fitbit flex teardown, http://www.ifixit.com/Teardown/Fitbit+Flex+Teardown/16050/1."

[6] J. Andrus and J. Nieh, "Teaching operating systems using android," in *SIGCSE '12*, pp. 613–618.

[7] J. Bayzick, B. Askins, S. Kalafut, and M. Spear, "Read, write, play: An educational mobile gaming platform," in *SIGCSE 2013*.

[8] S. Loveland, "Human computer interaction that reaches beyond desktop applications," in *SIGCSE '11*.

[9] R. L. McFall and M. DeJongh, "Increasing engagement and enrollment in breadth-first introductory courses using authentic computing tasks," in *SIGCSE '11*.

[10] S. Patel and Y. Patt, *Introduction to Computing Systems: From Bits & Gates to C & Beyond*. McGraw-Hill Professional.

APPENDIX

Projects budget

1. Obstacle Avoidance - $254.93
   (a) Arduino Uno Starter Pack ($65.00) x1
   (b) Motor Shield ($16.00) x1
   (c) Turtle 2WD Mobile Platform ($41.00) x1
   (d) Ping Ultra Sonic Sensor ($29.99) x3
   (e) Servo Extension Cables ($2.49) x3
   (f) AA Batteries ($1.36) x4
   (g) 9 volt Battery ($2.05) x1
2. Maze Navigating - $139.44
   (a) Arduino Uno Starter Pack ($65.00) x1
   (b) Motor Shield ($16.00) x1
   (c) Turtle 2WD Mobile Platform ($41.00) x1
   (d) Infrared Sensor ($9.95) x1
   (e) AA Batteries ($1.36) x4
   (f) 9 volt Battery ($2.05) x1
3. Tic Tac Toe - $381.40
   (a) Arduino Uno Starter Pack ($65.00) x4
   (b) TFT Touch Shield ($59.00) x2
   (c) NRF24L01 Radio Transceiver ($3.70) x2
4. Pong - $163.36
   (a) Arduino Uno Starter Pack ($65.00) x1
   (b) Liquidware InputShield ($39.36) x1
   (c) TFT Touch Shield ($59.00) x1
5. Twitter - $130.70
   (a) Arduino Uno Starter Pack ($65.00) x1
   (b) Graphic ST7565 Positive LCD (128x64) with RGB backlight + extras ($17.95) x1
   (c) Ethernet Shield ($45.00) x1
   (d) Ethernet Cord ($2.75) x1
6. Word Scramble - $96.65
   (a) Arduino Uno Starter Pack ($65.00) x1
   (b) Graphic ST7565 Positive LCD (128x64) with RGB backlight + extras ($17.95) x1
   (c) PS2 Keyboard ($13.70) x1