Designing the Finch: Creating a Robot Aligned to Computer Science Concepts

Tom Lauwers and **Illah Nourbakhsh**
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

Abstract

We present a new robot platform, the Finch, that was designed to align with the learning goals and concepts taught in introductory computer science courses. The Finch was developed in the context of the CSbots program, the goal of which is to improve retention and learning in computer science courses through the use of robots and other physically embodied hardware. This paper concentrates on design constraints that were determined in earlier CSbots studies and how those constraints were instantiated by the Finch. We also present some preliminary results from pilot studies in which Finch robots were used in CS1 and CS2 classes.

Motivation and Program Goal

In the context of steeply declining enrollments in Computer Science (Vegso 2005), the CSbots program focused on developing curricular modules for introduction to Computer Science (CS1) classes in which robots are used as educational tools to motivate students about applications of Computing. Sharp enrollment declines in Computer Science during the first half of the 2000s led to a wide number of approaches to motivate students to study computer science (Cooper, Dann, and Pausch 2003), (Maloney et al. 2008), (Bell and Fellows 2010).

Robots, too, have been used by a number of Computer Science educators, both in introductory Computer Science courses (CS1) (Fagin and Merkle 2003), (Blank et al. 2007), and higher level courses (McNally 2006). The results to date have been mixed, with a very large study using Lego Mindstorms robots in CS1 (Fagin and Merkle 2003) finding two critical weaknesses of using robots: Firstly, robots are typically too expensive for student ownership, and so students must work on robot programming assignments in labs with limited hours. Secondly, feedback is delayed due to the need to observe a program running in the physical environment, and so students must devote more time to tedious debugging, and less to developing solutions. These weaknesses can be seen as failures of alignment of the tool to the ecosystem of the introductory Computer Science course. A more recent study in which students had personal ownership of a robot and could take them home showed significant improvements in retention and grades compared to other approaches (Summet et al. 2009).

The aim of CSbots, then, is to create a robot design and accompanying curriculum that has the advantages of robots seen in other educational contexts, specifically improved motivation and interest in studying STEM (Melchior et al. 2005), while eliminating the weaknesses found in the Fagin study. This paper discusses the process of designing the Finch, a robot specifically designed for introductory computer science education, in the context of the wider approach taken by the CSbots program.

Project Background and Approach

Prior to the work presented in this paper, we conducted an initial evaluation of the introductory computer science field involving a textbook survey and phone interviews with 37 CS educators (Lauwers and Nourbakhsh 2007). We used the results of this evaluation to create an initial robot platform, software API, and curriculum, and tested this platform with roughly 70 students at a local community college (Lauwers, Nourbakhsh, and Hamner 2009). The initial evaluation and first design cycle helped us identify crucial design principles and constraints for the work presented in this paper.

Throughout the CSbots program, we have relied heavily on three methods; feedback and evaluation driven iteration of design, deep partnerships with educators working in the Computer Science field, and alignment of the robot and curriculum with the learning goals and audience of the CS1 class.

Iterative Design

We engage in an iterative design process with deep participation from computer science educators. The process is composed of the following steps:

Design. The design step involves the creation of a robotic platform and associated software, and the development of assignments and course outline.

Pilot. The pilot phase involves the pilot of the designed curriculum and technology in a CS1 course.

Evaluation. Learning, motivation, and retention are tracked with standard exams, weekly student surveys, and comparisons of drop-out rates. These data are used to inform the next design step.

Partnerships

The members of the original design team had no experience teaching introductory Computer Science, and quickly recognized the vital importance of partnering with CS1 educators. Our intention was to develop a partnership that was more than simply advisory, with a two-way sharing of domain knowledge and skill. Some of our partners have been involved from the start of the first design cycle, allowing them to test and comment on early versions of the robots and software, while providing us access to existing curricula and their understanding of what works with students in the CS1 class. While the details of the robot design have mostly been left up to us, our partners are full members of the design team when it comes to the software and curricular activities accompanying the robot.

Alignment

Alignment is a well-known curriculum design principle; essentially it advises the course designer to align the learning goals, instruction, and assessment, so that each supports the other. Although in some cases classes may be intuitively aligned by their designers, aligning a course is a process that can be performed in a premeditated, explicit way (Wiggins and McTighe 2005). This is often done through a process known as *backwards design*, which begins by detailing the learning goals for the students, as these goals will drive the required types of assessment and instruction. Alignment through backwards design is a method that can be extended to the design of educational tools, so long as those tools are targeted at a specifical curriculum or educational activity. Considering goals, instruction, and assessment as parts of the design leads to a design process that is qualitatively different than if we were designing a robot outside of an educational context.

Designing the Finch

The evidence at the end of the first design cycle provided support for the notion that robots could be effective educational tools in Computer Science education. Our initial design was lacking in important ways; we intentionally combined two commercial products (an iRobot Create[1] and a Qwerk controller[2]) to create a robot platform that was highly feature rich. The trade-off of this choice was that the robot was both too expensive and too bulky to be used as a personal robot. We knew that this was one of the major problems identified by earlier studies, but also felt that for an initial design step, an extremely feature rich robot would help us evaluate which features were actually important to student interests and to the design of good assignments.

After evaluating our initial pilot we engaged in a redesign of the robot, software, and curriculum to align with CS1 learning goals and logistics. We discuss the principles behind the new design, the details of the robot, and preliminary results from pilots next.

[1] www.irobot.com/create/

[2] http://www.charmedlabs.com

Design Constraints

Many sources of information informed our redesign: Our initial evaluation and literature review, our experiences with high school teachers testing different robot features, our earlier pilot study evaluation results, and much feedback on the initial robot from students and teachers. From these varying pieces of data, we formulated a set of evidence based design constraints:

- The robot should be sufficiently low cost for individual use. This was borne out both from past studies, from our own experience running a separate closed robot programming lab in an earlier study, and from feedback from high school teachers who were constrained to working with a single robot.
- It should be possible to complete assignments with the robot at home. This suggests three design constraints: Firstly, the robot hardware must be sufficiently robust to survive students' home environments and transport between home and school. Secondly, the robot must be small enough to be carried back and forth easily. Lastly, the robot software must work on the diverse computer and OS platforms students have available at home.
- The robot should be aesthetically appealing. In our earlier study, researchers often observed students personalizing robots and giving them anthropomorphic agency when their programs were running. We thought this sense of agency important to motivating the students, and felt an aesthetically interesting design would support it.
- The robot should be capable of interesting interaction with students and with the environment. *There was no one sensor or actuator that we had to include based on our earlier experiences with the initial design*. Instead, assignments that were successful in earlier studies leveraged multiple sensors and outputs to create richly interactive programs.

Robot

Much of the CSbots project time in 2008 was spent formulating the above design constraints and deriving from them an appropriate robot hardware design. By the end of the year we had developed a small, inexpensive, highly interactive robot, which we dubbed the Finch. The Finch design is unusual for a mobile robot - it must be tethered to a computer at all times. Deciding to tether the Finch provided a number of marked advantages to the design:

- The Finch derives power from the tether, thus there are no batteries to charge and robot behavior can not be affected by on-board power levels.
- The Finch has very little need for on-board processing; instead, a low cost microcontroller sends and receives commands over USB to the computer. Programs created by the student run entirely on the computer. In this way, the Finch is more of a computer peripheral than an autonomous agent.
- As a USB device, it is relatively easy to support the Finch in different programming languages.

- Setting up the Finch is simple - one need only install a USB driver (available for all major operating systems) and use our cross-platform software package.
- Tethering reduces the need for complexity in the Finch, which in turn significantly reduces the cost: We estimate that the Finch can be sold for under $100 commercially.

Due to its tether, the Finch is not a very capable mobile robot. This is not a reflection of a flaw in the design but of the design constraints themselves - capability as a mobile robot was ultimately less important than simplicity and robustness of design. Tethering provided a simple solution to meeting several of the design constraints simultaneously. We discuss how we met the constraints of aesthetics and interactivity next.

Our group decided to create a molded plastic shell for the Finch to both protect the robot electronics and to appeal aesthetically to students. We drew up three shell concepts (see figure 1) and distributed these to the high school teachers involved in CSbots. They and their students voted on which they preferred and provided comments, leading to a final concept that borrowed elements from each of the sketches.

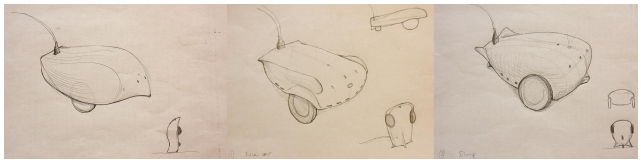

Figure 1: Sketches of Finch shell concepts

The final shell pictured in figure 2 intentionally expresses zoomorphic traits, with a beak, two eyes, a nose with two openings, and ear holes. Naming the robot Finch is a further effort to carry on this zoomorphic theme. The shell is designed to be easily grasped by students, with the curvature of the shell suggesting that it can be held with both hands with thumbs resting near the ear holes. Additionally, the rear part of the shell offers a tripod support to allow the Finch to be placed vertically on a flat surface - this is very useful when debugging a program in which the Finch's wheels are moving.

Figure 2: Finch shells

As the shell supports the notion of physically interacting with the Finch, the robot's hardware is similarly oriented around interactivity. From our initial pilots we realized that it was not so much a single sensor or actuator that was critical to successful interactions between students and the robot, but the combination of a number of features engaging students through multiple sensory paths: The initial robot could talk, flash lights, sense bumps, and move about; successful assignments were those in which students needed to utilize most of these features in a way that also involved their interacting with the robot in some way (for example by bumping the bump sensor, listening to the robot speak and responding, or moving the robot through a GUI designed by the student). As such, we sought to turn the Finch into a highly multi-modal device, one that can interact with students by engaging a combination of students' visual, auditory, and kinesthetic senses. The Finch can express motion through a differential drive system, light through a color-programmable LED, and sound through a beeper and using computer speakers. Similarly, it can sense light levels through two photoresistors, temperature through a thermistor, distance traveled through two wheel encoders, obstacles placed in front of it, and its orientation in three dimensional space through an accelerometer (see figure 3 for placement of the Finch's actuators and sensors). In addition to these hardware-based capabilities, the accompanying software allows students to easily have the Finch speak or play songs over computer speakers, read real-time data from internet RSS feeds, and react to video from computer webcams.

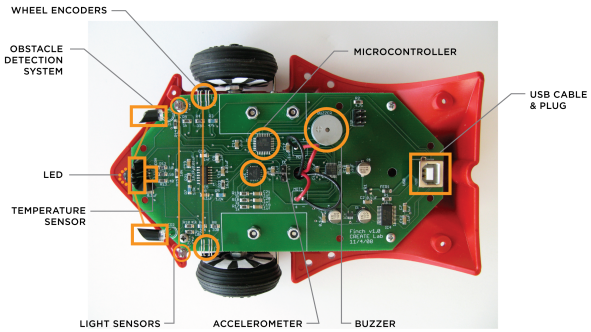

Figure 3: The Finch robot's sensors and actuators

Roboticists tend to think of mobile robots as creatures of their environment - they move within a space and react to stimuli in that space. Though the Finch is capable of being programmed to behave in this manner, the Finch's capabilities provide two other frequently used modes of operation. Firstly, there are programs that use the Finch as an input device to the computer; for example, the accelerometer data can be used to move a cursor on the screen. Secondly, the Finch can be used as a way to convey information from the computer and internet to the user in the physical world; for example, by shivering if it is cold outside or setting an alarm when an earthquake has struck somewhere in the world. Using these three loosely defined and overlapping modes, the Finch expands on the ability of a computer to sense and act in the world. This enhancement provides students with a

wider range of programs to write, ultimately leading, as we have tentatively begun to see, to more interesting and relevant assignments and more engaged students.

Software

At the moment, students program for the Finch with Java. This was a choice we made based on our initial evaluation of the CS1 landscape; we found that over half of CS1 educators were using Java and very few were planning to change languages in the next two years. Additionally, changing the programming language used was seen as a major undertaking requiring significant buy-in from within and outside the CS department. As programs written for the Finch execute on the computer, we intend to to develop support for additional languages in the near future.

One of the major themes in our conversations with partner educators as we were developing the Java API was to simplify as much as possible. As such, all of the methods related to the Finch are encapsulated in a single Finch class. The students instantiate an object of this class, and then call get and set methods to operate sensors and actuators. At the low level, instantiating an object of the class automatically causes the computer to open a connection to the Finch, and additional method calls result in data being sent over USB to and from the Finch. Students are provided with a full javadoc-style documentation of all of the Finch class methods. As this listing can be overwhelming to the novice, we also provide a short listing of the most important method calls. The software and documentation are freely downloadable[3].

The software package also combines three open-source packages to support native computer capabilities; a package to generate speech from text, one to read data from RSS feeds, and one to use video data from attached webcams. Standalone packages were created to allow teachers not using the Finch to benefit from these capabilities[4]. The entire package is encapsulated in a .jar file that students must include in java projects using the Finch; this can be done in any IDE or through the command line, but in our experience it does present a common stumbling block for novice users used to compiling a single .java file.

Curriculum

A strong finding of our initial evaluation was that major changes to the curriculum are difficult to execute independently for most computer science faculty. As such, the focus of our design when it comes to curriculum is to avoid changing the structure and learning goals of the introductory Computer Science course and to use the robot primarily as an additional tool for use in assignments. For example, when the Finch was used in a local community college pilot, the lecture slides and tests were not altered from prior years any more than is typical. The Finch was introduced in the fourth week of instruction; one lecture was devoted to explaining the Finch's features and demonstrating to students how to create Java projects and compile files within those projects. The delay introducing the Finch was deemed necessary by our partner to allow students time to learn basic Java syntax as well as the concept of external classes before beginning with the Finch. Table 1 shows the learning goals and description of each assignment in this pilot.

Learning Goal(s)	Description
Compiling a program; printing.	Write a program that prints to screen.
Variable types; operators and expressions.	Calculate miles per gallon.
User input.	Read in age and print life expectancy.
Selection structures.	Read the Finch's orientation (flat, beak up, or beak down) and say what it is.
Looping	Move around the room and avoid obstacles. Apologize if there was a near collision.
Student-written methods.	Students create a note player method that plays notes read in from user input.
Student-written classes.	Write a class that tracks the internal 'emotional' state of the Finch and expresses it when the playEmotion method is called.
Arrays.	Move the Finch through five points and collect sensor data at each point; store sensor data in an array and get the average, max, and min.
Begin graphics and events.	Create a tele-op interface.
Continue graphics.	Create a slider based tele-op interface to control LED.

Table 1: Assignment schedule of a Finch pilot

Finch Pilots and Evaluation

In January 2009 we built 100 Finch robots and began a lending program with interested educators. To date, Finches have been loaned to 16 high schools, 7 universities, 2 after school programs, and over 150 students at a local community college. Reviews have generally been favorable. In two cases, we have collected formal evaluation data from these pilots; one was a full test at a local community college in which every student was given a robot, the other was a loaning program with nine high school teachers. Results from the high school loaning program have been previously published (Lauwers, Nourbakhsh, and Hamner 2010). We present preliminary findings of the community college pilot here; a full analysis will be published later.

Community College Pilot

Finches were used in the spring and fall 2009 semesters in the CIT-111 and CIT-130 classes (CS1 and CS2) taught at

[3]http://csbots.wetpaint.com/page/Downloads

[4]http://csbots.wetpaint.com/page/No+Robot+Required

the Community College of Allegheny County by one of our partners. In the spring, three sections of CIT-111 and one section of CIT-130 used the Finch, and in the fall two sections of CIT-111 and one section of CIT-130 used the Finch. Every student in each section was loaned a Finch and could take the robot home with them.

Students were asked to fill out a pre-survey, a post-survey, and short surveys after each assignment. Pre/post surveys sought to determine characteristics of students who successfully passed the course, as well as interest in using the Finch. Assignment surveys were designed to track student interest and frustration on an assignment by assignment basis.

In addition to these surveys, we also compared the spring and fall 2009 CIT 111 class to previous courses taught by our partner; we compared the retention rates in the class and the average grades of students taking the class. We were concerned with the grades data primarily as a check that students were not performing more poorly due to the Finch. Though we do not report grades in depth here, the average grades were not significantly worse than prior years. We did not do a similar comparison for the CIT-130 classes because the low number of participants and intermittent teaching of the course did not provide enough participants to make such comparisons statistically valid.

Retention

We investigated the retention rate among students who stayed in the course through the first exam. We did this for two reasons: Firstly, many students who begin the CCAC course drop out in the first few weeks because they had a poor understanding of the course subject matter. Secondly, the Finch was not introduced until assignment 4, roughly at the same time as the first exam, and so it likely had no impact on retention before the first exam.

Comparing the pilot year in this way we see that the spring pilot slightly underperformed compared to the prior four years; the retention rate of 52% was lower than the rate seen in three of the four prior years and was slightly below the four year average of 56%; even so, none of these differences was statistically significant and it should be noted that the retention rate improved compared to 2008.

The fall course outperformed all other years and the four year average. The 80% retention rate was significantly better than 2006 and the earlier pilot year of 2007 ($p < 0.01$ for both). The rate was markedly above the four year average of 57% and above 2005 and 2008 as well, though these differences were not significant. It should be noted that we reworked the assignments in fall 2009 based on our experiences with the spring pilot, and believe that this coupled with a greater deal of experience using the Finch contributed to the improved fall retention rate.

When comparing between the study year and the prior four years, we see no significant differences in the retention rate, although the rate for 2009 is higher than for all other years (65% compared to an average of 56%).

Interest in the Finch

We sought to determine interest students had in the Finch through three questions. We asked incoming students if they would prefer programming a computer or robot, and asked outgoing students if they had shown the Finch to anyone, and if they had worked on programs for the Finch that were not assigned.

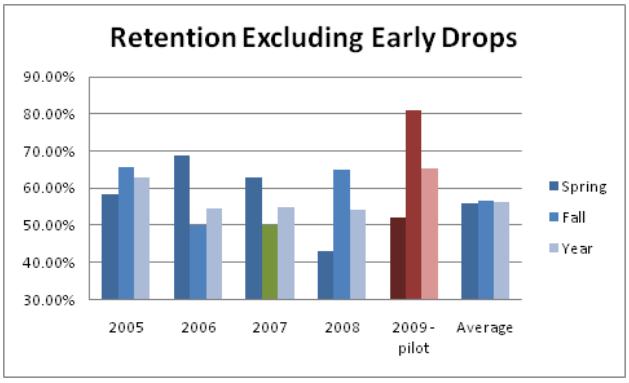

Figure 4: Retention rates excluding students who dropped before exam 1

We coded the answers to the first question into three broad categories - robot better, computer better, and not sure. Table 2 summarizes the answers to these questions split by class. Generally, roughly 2/3 of incoming students believed they would prefer programming a robot to programming a computer, with the third split between being undecided and believing they would prefer programming a computer.

Class	Robot Better	Not sure	Computer Better
CIT 111	68%	18%	14%
CIT 130	66%	10%	24%

Table 2: Percent of students who prefer to program computers or robots

Regarding the questions to outgoing students, every student who answered the post survey indicated they had shown the Finch to someone - typically friends and family. Most went on to describe the reactions of the people who saw the Finch; of these reactions, all but one was positive.

Students were less likely to have written programs for the Finch for fun; of twenty eight post survey responses, eight indicated that they had done so. Most of these students wrote small programs for their friends and family; four students gave an estimate of the amount of time they spent out of class on the Finch. Three students spent 3-5 hours on the Finch, and one indicated that she spent 30-40 hours out of class using the Finch. Though we don't have comparable data of students in regular programming courses at CCAC, we believe that it is rare in such courses for students to write programs purely for enjoyment.

Finch Robustness

We tracked failures of the Finch hardware over the course of the year. There were very few failures of any type:

- One Finch beak LED did not light up correctly due to poor soldering during assembly.
- On three Finches, screws came loose in the drive mechanism, preventing a wheel from turning.
- One Finch's USB cord was destroyed by a puppy.

Of these errors, only the LED failure is irreparable by students, and this was a manufacturing defect and not a result of use.

Summary

The 2009 CCAC pilots suggest that our revised design met some of our objectives: Most importantly, retention improved significantly in the fall course, students continued to struggle primarily with computer science concepts as opposed to robot hardware/software, the Finch robots had very few hardware failures despite extensive use by students at home, and students passing the course demonstrated excitement in the class by showing the Finch to friends and family and in some cases by working on programs recreationally. At the same time, the relatively poorer retention rates of the spring course demonstrates the importance both of properly aligning assignments and activities and of past experience teaching with a new tool like the Finch. It should be noted that while the spring course's retention rates were a few percent below average, the overall year's retention was better than average, and performance may continue to improve with increasing instructor experience using the Finch.

Acknowledgments

Don Smith at the Community College of Allegheny County has been involved with this project for several years and has been critically important to both the curriculum design and to piloting the Finch and our earlier robot platforms. Thanks also to Chris Bartley for his significant work on the software environment.

This work was funded by the National Science Foundation's Cource, Curriculum, and Laboratory Improvement program.

References

Bell, T., and Fellows, M. 2010. Cs unplugged. Http://csunplugged.org.

Blank, D.; Kumar, D.; Marshall, J.; and Meeden, L. 2007. Advanced robotics projects for undergraduate students. In *Symposium on Robots and Robot Venues: Resources for AI Education*.

Cooper, S.; Dann, W.; and Pausch, R. 2003. Teaching objects first in introductory computer science. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003*.

Fagin, B., and Merkle, L. 2003. Measuring the effectiveness of robots in teaching computer science. In *SIGCSE*, 307–311.

Lauwers, T., and Nourbakhsh, I. 2007. Informing curricular design by surveying cs1 educators. In *Proceedings of the 4th International Symposium on Autonomous Minirobots for Research and Edutainment*.

Lauwers, T.; Nourbakhsh, I.; and Hamner, E. 2009. Csbots: design and deployment of a robot designed for the cs1 classroom. *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education* 428–432.

Lauwers, T.; Nourbakhsh, I.; and Hamner, E. 2010. A strategy for collaborative outreach: Lessons from the csbots project. *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education*.

Maloney, J. H.; Peppler, K.; Kafai, Y.; Resnick, M.; and Rusk, N. 2008. Programming by choice: urban youth learning programming with scratch. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2008*.

McNally, M. 2006. Walking the grid: Robotics in cs2. In *Proceedings of the 8th Austalian conference on Computing education*, 151–155.

Melchior, A.; Cohen, F.; Cutter, T.; and Leavitt, T. 2005. More than robots: An evaluation of the first robotics competition participant and institutional impacts. *Heller School for Social Policy and Management, Brandeis University*.

Summet, J.; Kumar, D.; O'Hara, K.; Walker, D.; Ni, L.; Blank, D.; and Balch, T. 2009. Personalizing cs1 with robots. *SIGCSE Bull.* 41(1):433–437.

Vegso, J. 2005. Interest in cs as a major drops among incoming freshmen. *Computing Research News*.

Wiggins, G., and McTighe, J. 2005. *Understanding by Design, Expanded 2nd Edition*. Prentice Hall.