Open-Ended Robotics Exploration Projects for Budding Researchers

David R. Musicant, Abha Laddha, Tom Choi
Carleton College
1 North College Street
Northfield, MN 55057

Abstract

There are many benefits to introducing students to the idea of doing projects where the outcome is unknown or unsure. Some have proposed that engaging students in research can help with retention of underrepresented groups. In this paper, we report on a particular approach we have used to introduce high school students to open-ended robotics projects in a three-week summer program. We describe the structure of our summer program, how we ramp the students up to speed, and we summarize the five open-ended "research" projects that the students work on. These projects can be adopted for open-ended work elsewhere by high school students or undergraduates.

Introduction

There are many benefits to introducing students to the idea of doing projects where the outcome is unknown or unsure. Some have proposed that engaging students in research can help with retention of underrepresented groups (Peckham et al. 2007; Cuny and Aspray 2000). In this paper, we report on a particular approach we have used to introduce high school students to open-ended robotics projects in a three-week summer program. Our goals are to introduce students to the idea of having control over their own projects, but in a structured way. Is the work that they do "research"? At the level we're working with the students, we don't expect them to complete innovative work that is new to the field, but we do hope to see students show creativity and begin engaging some key academic principles of robotics.

It is our belief that the projects we use, and the structure accompanying them, can work well for older high-school or lower-level undergraduate students at a variety of experience levels. At least one of our projects is designed for students with essentially no computing experience at all; others are designed for students with previous programming experience. In this paper, we describe the structure of our summer program, how we ramp the students up to speed, and most importantly, we summarize the various open-ended "research" projects that the students work on.

When possible, we point students towards papers that have been written that target the work that they are doing. Being able to read research papers is an important skill that students can begin working on. We have found that for this audience, unpublished manuscripts, masters theses, and other like works can often be a rich source of material.

To summarize, here are the key contributions that we provide via this paper and the associated online materials (Musicant 2016):

- We present four structured labs for bringing students up to speed on a variety of robotics concepts, including PID control, finite state machines, and Q-learning.
- We present five open-ended research-like projects. In addition to presenting the project ideas and frameworks (some of which were conceived of by others), we discuss successful techniques as well as pitfalls that we have run into.

Initial Labs

Our program runs for three weeks. During the first two days, students begin with four half-days of structured classes, designed to bring them up to speed. In our program, the experience students have varies from nearly none at all, to many years of programming and/or high school robotics contests; therefore, we believe the activities we present here could work for a variety of experience levels. It is our goal to introduce programming skills as well as academic robotics ideas at a level appropriate for a range of experiences. While these initial labs are not the main emphasis of this paper, we are happy to share them. They can be adapted in a variety of ways for other programs. The full text from the most recent versions of our labs is available online (Musicant 2016). We emphasize vigorously that in constructing these labs, we stand on the shoulders of giants. Many aspects of these projects have been done before, in a variety of ways.

Our first half-day lab is intended simply as a warmup to programming with the robots we have. We currently use LEGO Mindstorms NXT robots. In the past, we used the NXC programming language (Hansen 2011); more recently, we have switched over to leJOS (leJOS 2015). On the second half day, we introduce them to PID control, and task them in particular with having their robot follow a line. We have found that of the students who enroll for our program, a very few of them have encountered this beforehand. Nonetheless, the exercise seems to challenge them. On the third half day, we introduce finite state machines and subsumption as architectures for organizing robot programs. Realistically, we spend most of the time we have working on FSMs; subsumption is something we end up presenting as an additional tool if students have extra time. A few students come to our program knowing about FSMs, but it seems that essentially none of them have programmed an FSM-like structure into code. Finally, on the fourth half-day, students learn about reinforcement learning. They implement a version of Q-learning in order for a robot to learn how to follow a wall, based on a particular reinforcement structure. This lab is challenging to complete in the time allotted, but is a particularly worthwhile experience for more advanced students.

Research Structure

Once students have completed the four initial labs, they then move on to planning and completing a research project. Students work on these research projects in teams of two that we create, where we do our best to assign students so that they have partners of approximately equal computing experience. We then present the students with five research projects to choose from. After we present the projects to the students as a group, we then invite the pairs to look through the projects, brainstorm some ideas among themselves, and rank their top three choices. We then meet with each pair individually, talk through their options and any details they may need to think about, and help them make a final choice. Generally, we try to make sure that we don't have too many pairs working on the same project, as it is more fun to have variety. We also try to ensure that each pair is working on a project that fits their level of expertise.

From this point forward, students essentially work on their research project full-time for the remainder of the first week, and then in the afternoons for the following two weeks. (The mornings for the second and third weeks are used for another purpose in our summer program unrelated to this paper.) During that designated research time, the students work to get as far as they can in completing their projects. Support is provided by a faculty member, as well as by an undergraduate lab assistant. (The supervising faculty member, and the undergraduate lab assistants from 2015 and 2016, are authors on this paper.) We wander the room, helping those that need it, and directing them as necessary. Early on in the third week students begin work on a poster summarizing their project. On Friday morning of the last week, students present their work at a poster and demo session where they can talk about their work and show their robots perform.

The Projects

In each of the following sections, we describe the projects that the students work on. More detail can be found in the actual prompts that we give the students (Musicant 2016). It is important to again note that we do not claim originality on the concepts for these projects; we learned much from looking at work that others have done. Some of these are well-known favorites, and others are based on singular implementations that we found. Our contribution here is in describing how these projects can be structured in a research-like environment with a fixed time frame, and lessons that we and our students have learned in implementing them.

Robot Pet

The main objective of this project is to build a robot pet that is as lifelike as possible. This project is intended specifically for beginning programmers. It works well for this group because of the easy access to well documented robot pet designs, and no rigid boundaries that restrict the movement of their pets. Given the open-ended nature of the project and the lack of experience by students who work on this project, any solution is likely a good one. That said, as our goal is to nonetheless help them experience some of the flavor of doing research, we do structure the project accordingly.

The first step for students is to physically design their pet, using LEGOs. The difficulty of this step shouldn't be underestimated, and was something of a surprise to us. It seems to be the case that significant inexperience in programming correlates highly with inexperience in physical engineering. We therefore recommend to students that they can choose how to spend the time that they have. They are welcome to spend time designing their own physical robot, and accordingly put less time into the programming. Alternatively, they can research, build, and cite someone else's physical design. Every time that we have offered this project, students choose to use a LEGO design that they find online. Remarkably, in the four years that we have been doing this project, every year a group chooses to build the NXT Puppy (Parker 2011). (It is easy to find via popular search engines.) We also had one group assemble a llama (Parker and Rhodes 2016). In all cases, we did not have precisely the same LEGO parts as these designs show, or students chose to add additional sensors, so they still needed to add their own design ideas.

Once students have a physical design, they move on to add functionality. In the early phase of the research, for example, the students start out making their robots freely roam around the room. Then, they think of other interesting actions that they can add such as chasing an object. Groups vary, but here is a list of features that we have seen them implement:

- Chase an infrared ball; we have HiTechnic IR balls and sensors on hand
- Bark when some sensor value triggers, such as hear a loud noise, or detect an object nearby; students download barking sound files, or record their own
- Respond when the person touches it, i.e., when buttons are pressed
- Detect colors of dog bones on the floor while rolling over them, and dance accordingly
- Maintain a "happiness" state value, and increment it or decrement it according to sensor readings; behave accordingly (perhaps make a whimper noise when sad)

To help the students keep their code organized, and to encourage them to integrate principled ideas, we heavily encourage the students to implement their program as a finite state machine (FSM). In practice, with beginning programmers working on this task, we often find it challenging to get them to actually do so. Their level of programming skill is often not high enough to see the merit of the FSM implementation, or to be able to understand the nuances of it. When their programs get too complex to easily debug, as they often do, encouraging the students to rethink their behavior as an FSM at this point has sometimes been more successful.

Soccer Player

The main objective of the soccer player robot is to program a robot to play soccer which includes tracking and following a ball and shooting it towards the goal. We point students towards two papers that describe robotic soccer with LEGO Mindstorms (Junghans 2001; Lund and Pagliarini 2000), that use a variety of different approaches. To date, we have seen this project only done by a a single team that focused on the accuracy of shooting at the goal, but we also think this project would be wonderful if two teams were willing to work together to play against each other. The project expects some previous experience in programming.

We supply our students with a HiTechnic IR ball and corresponding sensor, so students begin by simply experimenting with the ball and sensor and learning how it works. The team that worked on this project next moved on to creating a suitable design for their soccer robot in order to be able to both track the ball, and also be able to navigate within their homemade cardboard soccer field. The sensors they used in the final prototype included:

- Infrared Sensor: used to pinpoint the ball's location and move towards it.
- Ultrasonic Sensor 1: used to determine where the robot is with respect to the goal. The goal had a piece of cardboard protruding higher than the rest of the soccer field, so this ultrasonic sensor was placed at a similar height.
- Ultrasonic Sensor 2: used to determine whether or not the robot has the ball within its grasp. This sensor was placed low on the robot, near where the robot would catch the ball.
- Light Sensor: helped the robot determine the boundaries of the field. Shading and lines were used on the soccer field to help determine where on the field the robot was.

Behaviorally, the team used an FSM to organize the different functions of the robot as different states. The states included:

- Find Ball: the robot follows the ball until its ultrasonic sensor registers that it has the ball, in which case it transitions to Find Goal
- Find Goal: the robot turns until the goal sensor registers the goal in front of the robot, in which case it transitions to Go To Goal
- Go to Goal: the robot moves towards the goal. If it ends up close enough, transition to Made Goal; if it loses the ball, transition to Find Ball
- Made Goal (terminal state): the robot stops when it reaches the goal with the ball in its grasp

Using these techniques they were able to achieve a smooth and logically performing soccer robot that could push a ball towards a goal. A major challenge they faced was detecting whether the ball was in the robot's grasp. Originally, they tried to to do this with the infrared sensor, by measuring the intensity of the infrared signal. This proved not to be precise enough, however. They also tried to use a light sensor to look for a shadow made by the ball, or a touch sensor triggered by the ball touching it. They struggled in both cases to make this work reliably. For this particular group, they ended up using the ultrasonic sensor approach. This did require physical redesign of their robot to keep the two ultrasonic sensors from interfering with each other. Another challenge they faced was that the ultrasonic sensor could not detect the goal accurately from too far away. A potential improvement would be a multicolored field and using color sensors which might allow for goal detection from a further distance.

Optical Code Reader

This project invites students to build an optical code recognition system that is described in a published paper (Comite and Moro 2009). It offers the students an opportunity to review a research paper written at a level accessible to them. It also exposes them to Hopfield networks, which are a particular form of recurrent artificial neural networks. This project is intended for students who are more proficient in coding.

The objective is to build a robot that can be programmed by paper strips. Specifically, students create paper strips with light and dark blocks in varying patterns, and they build a reader for those strips that utilizes a light sensor for scanning it. The system they build needs to recognize which bar code has been read, and then execute external commands as specified. While the code reading portion of the project is well specified, deciding what to do with the codes is up to the students.

Students start off the project building the physical code reader device. The research paper shows a photo of a prototype, so that usually helps point the students in the right direction. Nonetheless, the photo is not entirely clear, and the paper does not include step-by-step instructions on how to build it, so the students spend time thinking through how to design it themselves. The standard LEGO light sensor works well enough for determining light vs. dark regions on a strip of paper. More challenging for the students is building an apparatus that uses LEGO motors and gears to linearly feed a strip of paper through. A LEGO rack and pinion set, which is obtainable, is typically a huge help. Students commonly choose to use a second light sensor, but for different purposes. We have seen one group using a light sensor as part of the feeding mechanism, in order to determine whether a paper strip is on position. We saw another group choose to use a second sensor simply to read two columns of coding data at a time. This latter idea was incorporated by a pair of students that specifically wanted to improve the efficiency beyond what they read in the original paper.

Once the apparatus is built, students work to be able to read strips of paper and read dark and light spots on them. This is likely the most straightforward part of the project.

Finally, students need to match a strip that they read against a library of known strips. If the reader is too robust, and the strips are too carefully made, there isn't much of a challenge. This project is interesting when a variety of strips are made that represent the same code, but are imprecise enough so as to cause errors in reading or matching. If the reader cannot read a strip with 100% accuracy, or of two strips are supposed to be identical but read differently, this is where the algorithmic aspect of the project becomes more involved. Students then engage in a variety of approaches for determining which strip is really intended following a noisy read. The paper that they are provided with recommends the use of Hopfield networks. Though this may not be the first approach of choice for a modern practitioner, they're a lot of fun to try. Most notably, they introduce the student to neural-network-style computation in a compelling example that is quite implementable. It does require some mathematical sophistication to be able to understand, which in turn reinforces the research-like nature of the project. Typically, we encourage the students to start off with a quick ad-hoc method for implementing matching, and then suggest that they implement Hopfield networks afterwards. They can then compare and contrast the approaches, both intuitively as well as experimentally.

We have seen three different teams work on this project, all of whom found it quite doable. They varied notably in where they chose to put their efforts. One of the three teams did choose to implement a Hopfield network approach after having prototyped an ad-hoc solution. A major challenge they faced in implementation, as would be expected, was in debugging their mathematical formulas. Another team first prototyped code matching via a k-nearest neighbor algorithm that we we suggested. Ironically, it worked so well that the students could not be coerced into attempting the more challenging Hopfield network approach, which carries more of an AI flavor to it. This was a learning lesson for us: on this project, keeping k-nearest neighbor in reserve as a backup algorithm is a great idea, but it is so easy that introducing it to students first may dissuade them from studying other approaches. The third team that attempted this project found the user interface and object-oriented aspects of the project more interesting than the code matching, and chose to stick with their ad-hoc matching technique. Instead, they decided to spend their time on an object-oriented design model that allowed them to use their code reader to interact with their computer in different ways (act as a calculator, act as mouse/keyboard input to the computer, or play music).

Emergency Rescue Robot

We describe the goal of this project as building a robot that can navigate and locate a target trapped in a place like a mine. In practicality, this is an easily recognizable classic robotics project; the robot needs to learn how to navigate a maze. We give this a research-like flavor by encouraging the students to develop their own robot designs and their own navigation algorithms before we point them to some classic approaches. They also have significant control over the design of the physical maze itself. Specifically, we use two different approaches to the physical maze. One of them is a wooden maze with vertical walls that we constructed ourselves; the other approach is to use vinyl square floor tiles[1], which can be laid out as desired. Each group picks one of the two approaches and makes a robot that explores and maps the maze on its own, and ideally pursues optimal path finding as well.

The project is very flexible in terms of the experience levels of the students. Those without much programming experience can focus on getting their robots to simply explore the maze following along the wall. Further directions where the project can go include:

- Keeping track of the robot's own position in the maze
- Recording a map of portions visited
- Displaying the map on the robot's LCD screen
- Transmitting the map to a desktop via Bluetooth communication, and having a program on the desktop draw the map in progress
- After having explored the maze, determine shortest path to a goal via breadth-first search, A* search, or D* search.

This variety of aspects of the project enables them to freely readjust their goals in accordance with their progress and the levels of their expertise.

As mentioned earlier, we let each research group pick either the wooden maze or the floor tiles. The wooden maze has physical walls that are few inches taller than the rescue robots. The other maze is a set of 1'x1' white floor tiles, which allows the students to configure their own layouts.

Knowing how far a robot has traveled is essential for updating its position and mapping the maze. It is in principle possible to keep track of how far it has traveled by counting the number of motor rotations for each direction. We have had better success by encouraging students to apply tape to the borders between regions of the maze. Students then use light sensors to detect when the robot has transitioned from region to the next.

For both forms of this project, simply navigating throughout the maze is a more challenging task for the students than perhaps it would seem on first look. In the case of the wooden maze, students need to be able to determine whether or not there is a wall on one side to determine whether or not the robot needs to explore a new route. A touch sensor can do this, but can be difficult to physically engineer so that it can trigger reliably. In our experience, an ultrasonic sensor works better. Keeping the robot oriented correctly in a given direction is challenging as well. Our students have generally solved this by using strips of black tape across the maze that run across the path that the robot takes. Their robot uses two downward-facing light sensors, one on the left side of the robot and one on the right. If one sensor hits a strip of tape before the other, the robot straightens itself out accordingly.

In the case of the floor tile maze, our students use tape to mark the "walls," and accordingly use a light sensor to detect when the robot is hitting one. Determining when the robot has transitioned from one tile to another for mapping

[1]Thanks to J. Ben Schafer at University of Northern Iowa for this idea.

purposes, is often done by using strips of tape at each tile transition point near the center of the robot, and using a light sensor there as well. One place where we have seen significant variation in projects is in the navigation algorithm used. One team took a "discrete tile" approach, where the robot simply rolled forward from one tile to the next, where each transition was treated as a single action. To determine if the robot should turn, it would simply try it; it would turn, and see if the next tape detected indicated a wall or a tile transition. If it encountered a wall it would reverse its action, and try a different direction. Another team instead used a "continuous" approach, where they followed along a wall as a line-following operation, not paying attention to individual tile boundaries. A turn, then, took no extra programming work at all; it happened automatically as part of the line following operation. The tricky part for this group was detecting for mapping purposed when a turn happened, which they did via timing; they determined a turn happened if the robot lost touch with a wall for a certain calibrated period of time. This approach admittedly is not one that we would choose to implement ourselves, but the students involved were quite proud of it and took considerable ownership for their technique. Furthermore, it enabled the students to leverage the PID control code that they had written during the first week.

Exploration Robot

The exploration robot project involves having a robot explore a mostly open room and map it. It is motivated by the fact that there are places that humans can't go, and robots would prove extremely useful in these circumstances. This seems similar perhaps to the rescue robot project described above; the key difference here is the lack of structure. Instead of a detailed maze, the robot is in a relatively free-form space. The project is designed for those with at least moderate previous programming experience. It is also designed at three different levels depending on the skills and interests of the groups:

- Explore a space with walls and obstacles (such as boxes) and use the ultrasonic sensor to measure distances to objects, and to build a map as you go. Render the map in some rough form on the robot LCD screen.
- Improve the representation of the map beyond the LCD screen by drawing it on a nearby desktop, using Bluetooth communication between the two.
- Solve the so-called "kidnapped robot problem": have the robot locate where it is in the space. if it were suddenly dropped in an unknown location, after having previously mapped the space. We point them to a paper that has described how to do precisely this with a LEGO NXT robot (Singh Jasmeet 2013). To date, students that have worked on this project haven't gotten to this stage, but we think that having this out there as an advanced goal is worthwhile to motivate and set up what they do.

We have had two teams take on this project so far. In both cases, we learned that this is a fairly complex project, and each team chose a different subgoal to work on. The first team that worked on this project choose to carefully map a single large object of (relatively) arbitrary shape; they started off mapping rectangular boxes, then moved on to L-shaped arrangements. To do this, they designed a robot that used an ultrasonic sensor for following along the edge of the object. The actual distance traversed by the robot was recorded by the tachometers in the drive wheels. Two touch sensors on the front allowed the robot to detect surfaces directly in front of it.

Algorithmically, they accomplished their task by having the robot maintain a constant distance of 20 cm from the wall via a PID controller. When the robot was no longer able to sense a wall next to it, or it hit an obstacle, it would record the end of the wall, and then turn. Each turn was a simple routine designed to put the robot in position to follow the next wall. A complication here that they dealt with was that the box they used had slightly rounded corners, which means that detecting the end of the edge was tricky. They resolved the issue by recording a corner if one wheel had traveled significantly farther than the other one. They were able to ultimately generate a map of the entire object, including its orientation, which they transmitted to a desktop and displayed on the screen.

The second team that took on this project decided to start with the subgoal of keeping the robot in a fixed location, and mapping the visible horizon around the robot via the ultrasonic sensor. In other words, the robot started in a fixed location, and had the ultrasonic sensor on a motor; it swept the sensor in a 360 degree circle around the robot, measuring the distance in all directions. This was to be a first step in a particle-filtering approach, where the robot would then move to multiple locations, take similar measurements, and then combine. However, this team ran into a challenging problem. They ran an initial test by placing their robot inside a cardboard box, measuring the distance registered in all directions around the robot, and drawing what they found. They expected to see a rectangle, but instead, they found that the corners didn't match; the left and right walls of the box didn't intersect with the front and back walls. Furthermore, the walls came out appearing to be curved. This stumped the students for quite a while, and they spent considerable time trying to debug their code and their trigonometry to see what was going on. In the end, there were two different sources of error contributing to what they saw:

- They discovered a non-linearity in the distances reported by the LEGO ultrasonic sensor; at very close distances, their distance sensor increasingly and predictably misestimated the distance. They solved this by experimentally measuring actual distance with a ruler vs. distance reported by the sensor at a variety of values, and used a curve fitting approach to correct for the error.
- They had inadvertently made a very subtle physical error in converting from polar coordinates to rectangular coordinates in order to plot the space on the screen. The issue was not in their trigonometry, which they spent a lot of time re-checking. In the end, the problem was that the LEGO ultrasonic sensor reports distance from the edge of the sensor device, not from the junction point where it would be attached to a motor. When determining the dis-

tance from the pivot to a wall, the students had neglected to add on the width of the ultrasonic sensor device itself.

Diagnosing these two problems was quite challenging to the students, but they got a major reward when fixing them resulted in a nearly perfect looking rectangle being drawn on their robot's screen. In the end, both teams working on this project ended up going in a somewhat different and lesser direction than the project description entails, they both ended up facing real challenges that they had to learn how to solve.

Challenges and Lessons Learned

Working with high school students in a research-like context comes with its share of challenges, and we have learned a lot from the four times (once each summer for the last four years) that we have run this program. One key thing we learned is that it is important that we provide a list of projects to choose from, as opposed to them designing their own from scratch. The first year that we offered our program, we invited students to design and choose their own projects. We found that they did not know enough of the robotics landscape to know what to pick. For the most part, the projects they picked were either impossibly hard to do (which we steered them away from), or fairly dull. The quality of their work, as well as the amount they learned, increased dramatically once we began providing a choice of projects.

Every year that we have done this, one team has chosen to implement some sort of multithreaded code. LEGO NXT robots provide the capability for doing this. It has been our experience that multithreaded code for this audience is always a bad idea in the end. Debugging race conditions is challenging under the best of circumstances, let alone being done so by relatively inexperienced students on robots that behave unpredictably due to real-world interactions. It is our recommendation to heavily discourage students from trying to implement multithreading.

Our experience has not been free from the challenges that inevitably arise when students work in pairs. In our experience, it is absolutely critical that students must have similar skill levels in order to work on the same team; otherwise, one team member thoroughly dominates the other. This seems to perhaps be an even worse problem in an open-ended project such as this one, than it would be perhaps in a more directed class assignment.

Finally, a challenge that readers of this paper might face is implementing our projects under different timing constraints. Rather than offering this as summer program, some might think about doing this as lower-level undergraduate research seminar, for example. We believe that the structure we present would not need to change dramatically. In our morning classes, in which we do the structured labs, we have a 3 hour time frame. The first window of that is spent discussing the underlying concepts and the labs, and then the students have approximately two hours to work on that lab itself. Each lab could instead be done as a week of three one-hour classes; the first would be a lecture doing the setup, and then the remaining two classes would have the students working on the lab itself.

Conclusion

We have presented a structure and a number of projects to engage younger students (high school, or perhaps at the early college level) in research-like experiences using robotics. The prompts that we provide the students with are available online (Musicant 2016). Many of the students that we work with are in the process of deciding whether they want to further study computer science as a major. We believe that these projects provide tangible ways in which the students see their code come to life, which we hope will help in creating and sustaining a long term interest in the field.

Acknowledgments

The three week structure for our program, described early in this paper, was collaboratively developed by a number of faculty and staff members at Carleton College. In particular, we thank Jeremy Updike, Amanda Williams, and Katie Lauer at the Carleton Summer Academic Programs office for their insights in assembling and running this program. Daniel Barter and Eric Ewing were undergraduate lab assistants during the program's first two years, and their efforts were critical in the development of these projects.

References

Comite, M., and Moro, M. 2009. How introducing artificial intelligent behaviours in educational robotics. In *INTED2009 Proceedings*, 3rd International Technology, Education and Development Conference, 2476–2483. IATED.

Cuny, J., and Aspray, W. 2000. Recruitment and retention of women graduate students in computer science and engineering. Technical report, Computing Research Association's Committee on the Status of Women in Computing Research.

Hansen, J. 2011. Not eXactly C. http://bricxcc.sourceforge.net/nbc.

Junghans, A. 2001. Collaborative robotics with Lego Mindstorms. Master's thesis, Karlsruhe University of Applied Sciences, Germany.

leJOS. 2015. leJOS. http://www.lejos.org/.

Lund, H. H., and Pagliarini, L. 2000. RoboCup Jr. with LEGO Mindstorms. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*.

Musicant, D. 2016. Carleton SCSI labs and research projects. http://www.cs.carleton.edu/faculty/dmusicant/scsiweb/eaai.

Parker, D., and Rhodes, R. 2016. Lego NXT Llama. http://lego.build/2cNgBTF.

Parker, D. 2011. Puppy. http://nxtprograms.com/puppy.

Peckham, J.; Stephenson, P.; Hervé, J.-Y.; Hutt, R.; and Encarnação, M. 2007. Increasing student retention in computer science through research programs for undergraduates. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, 124–128. New York, NY, USA: ACM.

Singh Jasmeet, B. P. 2013. Map construction and localization using Lego Mindstorms NXT. *Journal of Automation, Mobile Robotics and Intelligent Systems* 7(3):22–30.