GoGo Board Self-Driving Car Challenge: The Influence of Exploration vs. Starter Material on Student Outcomes

RJ Pimentel and Bijal Patel

Education 130

May 8 2018

Table of Contents

Abstract

We conducted two similar lessons in two different computer science classrooms. Each class was given a set of GoGo board robotics kits and basic instructions to build a basic self-driving car that could autonomously navigate a short obstacle course. One class was given time to explore the entire set of GoGo board tools and was asked to come up with a self-driving car design on their own. The other class was given the basic structure already built and some introductory starter code to turn on the motors. We explored the discussions that occurred in each respective classroom and compared the benefits and drawbacks of both lesson types. At the end of our research we found that there were distinct benefits and drawbacks of either approach. Allowing the students to explore the tools on their own allowed them to build their understanding and interest in the broader design aspect of the lesson, while the class with more starter code was able to progress further in the lesson and came closer to actually implementing a basic autonomous vehicle.

Introduction

This semester, we both had the opportunity to participate in a field placement in a Computer Science classroom at Berkeley High School. As we are both Computer Science majors, this was a unique opportunity and we learned a lot about teaching in the real world from our mentor teacher, Ira Holston. Though our individual experiences were a little different as only one of us was able to volunteer in the AP class, we were able to create a joint lesson plan that allowed us to compare the same lesson taught in two different classes.

Unlike most high school classrooms, Mr. Holston's classroom was an open environment that encouraged a lot of student discussion and self-paced student learning. Mr. Holston didn't lead class with a lecture everyday. Instead, students used Khan Academy tutorials to learn the new languages at their own pace. He also assigned projects that had to be completed with a week or two. This allowed students to work with their peers and refer back to the tutorials as needed to finish the project. Our job in the classroom was to walk around and assist students as needed. When the students didn't raise their hands, or were too shy to ask questions, we were asked to talk to the students about their code and make sure they fully understood it. Pamela Fox, the creator of the computing curriculum on Khan Academy, would come into the classroom sometimes to work with the students as well. Her experience was really beneficial to the classroom because both of us are still students and Mr. Holston is self-taught. This open, collaborative environment allowed us to easily engage with the students on an individual basis and gave students the confidence to ask for help when needed.

Both of us had a general idea of what our research question would be going into the lesson. As both of us were in at least one period of regular computer science, we wanted to compare the student discussions and project results from both periods. This allowed us to incorporate many of the themes that we discussed in our class Education 130: Knowing and Learning in Math and Science: discovery learning, the positive effect of group work on student learning, and creating a collaborative classroom culture. We also really inspired by Professor Paulo Blikstein's lecture on technology in classrooms. As we were working with computer science students, the GoGo kits were the obvious choice for our problem because it would build off of the students' prior knowledge in a unique and practical way. While we chose the Robotics problem with the computer science students in mind, we decided to create a more versatile lesson plan that could be implemented in any classroom. In this way we could work on developing an easily accessible computer science activity which could have a significant impact even outside of our research.

While picking a problem to teach, we were both inspired by Professor Paulo Blikstein's paper "Travels in Troy with Freire". In his travels, Professor Blikstein uses technology as a means to allow students to participate in the real world by physically build projects[1]. Similarly, we hoped to provide the students with a real world example of how they can use the skills they were building in class to make a motor move and build physical machines. Our research focuses on the difference between an open ended lesson versus a structured lesson and uses Fruedenthal's ideas to compare them. Fruedenthal explains the importance of discovery learning in *Geometry between the Devil and the Deep Sea*[2]. This concept discusses the importance of students being left to solve problems without revealing too much of the solution beforehand,

[1] Blikstein, P. (2008). Travels in Troy with Freire: Technology as an agent for emancipation. In P. Noguera & C. A. Torres (Eds.), Social justice education for teachers: Paulo Freire and the possible dream (pp. 205-244). Rotterdam, Netherlands: Sense.
[2] Freudenthal, H. (1971). Geometry between the devil and the deep sea. Educational Studies in Mathematics, 3(3/4), 413-435.

which we related to the impact of varying amounts of starter code or starter electronics kits during many of our computer science projects. Another key aspect of these lessons was the group work involved. Although we did not have enough materials to provide each student with their own kits, allowing them to work in groups provided many benefits to the students. Students were able to bring their unique skills to the table and work together to build the project. In "Learning Mathematics in a Classroom Community of Inquiry", Goos describes a "community of practice" as a classroom that promotes student discussion[3]. As the paper explores, creating a collaborative environment provides many benefits. We explore these benefits in our research, specifically the effects of discovery learning on the types of collaborations between students in a computer science classroom. Though this lesson was done in a computer science classroom, there are countless real world applications and thus this problem can be implemented into a variety of different classrooms.

[3] Goos, M. (2004). Learning Mathematics in a Classroom Community of Inquiry. *Journal for Research in Mathematics Education*, *35*(4), 258-291.

Design

Motivation

The GoGo boards presented a perfect opportunity for our students to practically apply the skills that they were already developing in their Computer Science curriculum. Learning how software can be applied to a completely customizable physical system is both powerful and exciting, and we wanted to give our students access to technology they might not normally have the opportunity to engage with. Also, the relationship between sensors, software and motors is a fundamental concept of robotics and one that the GoGo board kits capture extremely well.

The Problem

"How can you use motors and sensors combined with robotics and programming to create an autonomous car that can navigate an obstacle course?"

This problem is adapted from the Professor Paulo Blikstein's problem presented in the Spring 2018 Education 130 class, originally inspired from his paper "Travels in Troy with Freire." The problem has a very low barrier for entry due to the simplicity of the GoGo system and interface, however it also offers a high ceiling for those who want to take their vehicles further. We decided to focus on this topic because it both interested us as Computer Science majors and seemed relevant and engaging for our high school placement students. Also because our field placement was a Computer Science class itself, this lesson didn't dramatically deviate from their curriculum. We chose to focus specifically on self-driving cars because there is a large amount of investment and interest in autonomous technology industry currently as well as robotic technologies in general, making the lesson both relevant and accessible.

The target learner group for this project is high school students, preferably in a computer science class although no prior programming experience is necessary. The lesson will first teach students about the basic structure of robots - a computer using sensors for inputs and motors or other mechanisms for outputs. It will also teach some basic programming, especially the use of loops and conditional statements. However, primarily the GoGo board problem is designed to get students to engineer a solution to a real-world problem and use various resources and 'techno-junk' to make their ideas come to life. It will also give students experience engineering in a team setting, mimicking a real-world engineering environment. Solving this problem can allow students to have a practical outlet for coding skills or offer an engaging way to be introduced to programming. General engineering and design skills are also fostered through this process, as students have to go through the entire design and development cycle for their device.

Our Experience

When this problem was first introduced in our classroom, we were given almost two hours in class and many used several days of extra time outside of the class to create a project. We were given the GoGo board kits and were asked to explore the tools given to us. Once the different parts of the board and the wires were explained to us, we went into our first assignment: using a sensor of our choice to make a motor move. Over the week, groups of students worked on using the sensors and household materials to create a machine that could solve a simple problem.

Although this was an excellent lesson for the GoGo boards, it simply wasn't possible to complete in one or even two 50 minute periods. We still wanted to capture the creative spirit of our experience in Education 130, so we compromised by defining the problem the students were solving beforehand - to create a self-driving car.

<u>Part 1: Learning how to use the GoGo tools and technology</u>

For this activity, we are using the GoGo board kit. This includes a circuit board, a variety of sensors (light, thermal, humidity etc.), motors, and wires. The board has a micro USB port in the top right corner that connects to the computer. It also has white 2 pin ports on the top left which connect to motors and white 4 pin ports on the bottom which connect to sensors (the lettered ones are motor ports and the numbered ones are sensor ports). With only some basic instruction on what wires to use for motors and sensors, they will be allowed to explore the available sensors and their functionalities. The goal is to use any sensor to make a motor spin. Many students, especially those without coding experience will struggle to find the right tools to use. There are many skills that are involved in this simple task. Firstly, they need to find a way to manipulate the sensor

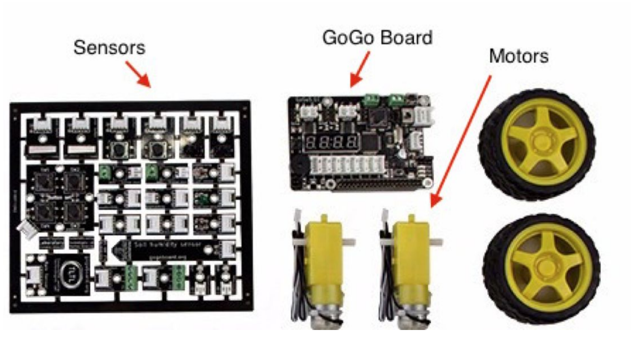


Figure 1 the kit includes a variety of sensors, 2 motors, wheels, a micro usb cable (not shown), wires (not shown) and a GoGo board that will connect to a computer

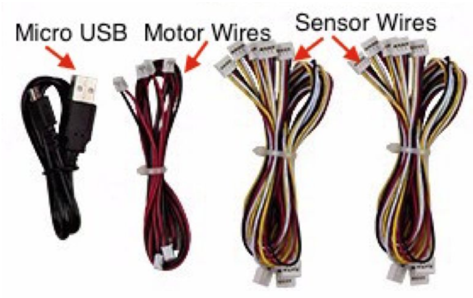


Figure 2 there are 3 different types of wires in the kit

readings. Secondly, they will need to interpret the readings from the sensors and find a way to use that information. Then, the students will need to find a way to talk to the motor in order to make it move.

The GoGo Learning interface:

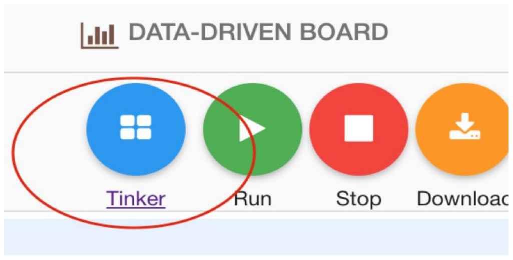
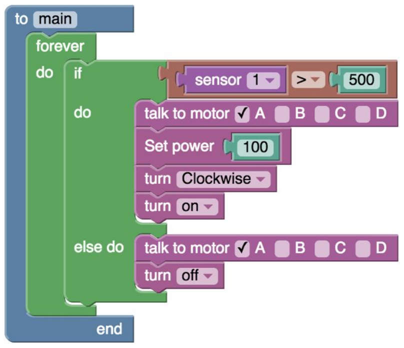


Figure 3 the tinker button will take you to the website that will allow you to code your GoGo board

The GoGo Learning app allows students to use preformatted puzzle pieces to create their program so they don't have to learn a whole new language. Everything the students code must go inside the "to main" block. Students will only need to use the blocks in the Program Control, Sensing, and the DC Motor categories but are encouraged to look through the other categories if they wish to do so. Program Control contains blocks that allow students to make comparisons and perform loops. Sensing contains blocks that talk to the sensors. DC Motor contains blocks that talk to the motor. In figure 4, the blue section is the main block, the green section is a program control block, the brown section is a sensing block and the purple sections are DC motor blocks.



```
to main
  forever
  do  if    sensor 1 > 500
      do    talk to motor ✓A  B  C  D
            Set power 100
            turn Clockwise
            turn on
      else do talk to motor ✓A  B  C  D
            turn off
end
```

Figure 4 shows sample code for using a sensor to move a motor

Part 2: Building a Self-Driving Car

In this part, students will be asked to use the motors and sensors given to them to build a rudimentary version of a self-driving car. They will also be able to test their car on a simple

obstacle course set up by the teacher. This will involve three parts. First students will need to program the car to follow the simple obstacle course. Then, they will be provided with materials such as cardboard and tape to build a physical car around their GoGo board. Finally, they will bring their car to the obstacle course, test it, and return to their desks to make edits.

One of the most prevalent ideas would utilize the proximity sensors in order to detect and avoid obstacles. Although this is the most likely approach for students, it's nonetheless technically complex to implement. In order to make it functional, students will likely have to utilize the proximity sensor suspended a certain distance from the outside of the car. It would then have to respond accordingly by powering up one motors at a time in order to turn the car. The biggest downside to this approach, besides the extra work students will have to put in to make it functional, would be the amount of materials available. If the classroom runs out of proximity sensors, students may have to make the car turn on hardcoded timed intervals. Although this approach wouldn't necessarily directly respond to obstacles, it could potentially be customized for the particular obstacle course.

Figure 6 wheels can be attached to the motors to create cars (Warning: make sure they both turn the same way)

<u>Background Knowledge</u>

This problem is once again particularly useful because of it's low barrier to entry and high ceiling for those who want to take it above and beyond. Students with less confidence in their coding abilities will still be able to heavily contribute to the idea and design of their device without losing out on the core value of the engineering process. However, if the lesson is taught in a classroom of computer science students there can be far deeper coding concepts that are applied to make more complex commands given different inputs.

In terms of the robotics, no experience with electrical engineering is needed beyond a simple familiarity with wires and inputs. Some basic construction skills will also be useful for creating the frame and structure of the vehicle. If there happens to be a 3D printer accessible, such as Mr. Holston's class, knowledge of how to model and print could be an asset in expanding the capabilities and durability of the vehicles. Beyond these skills, every student will be able to draw from their own unique perspectives and ideas in almost every stage of the design process.

Challenges

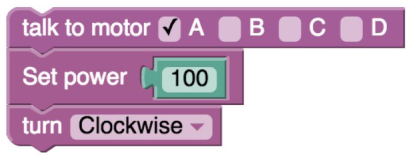


Figure 9 shows how the blocks that allow students to change the power and orientation of the wheels

One of the major challenges students will face is learning how to make the car start and stop at the appropriate times. They will also need to find a way to make the car turn, which will require the use of at least two individual motors. There are many ways students may approach this problem. One way is the make the inside wheel turn backwards while the outside wheel turns forward. Another way is to stop the wheel on the side and keep at least one motor on at all times. Both these methods will require motor power changes or orientation changes. Figure 9 shows the blocks needed to make these changes.

With the wide variety of sensors that the students are provided with, there are many errors they may run into. Here are a few common issues and their solutions.

1. Interface Errors

A common issue is reading or talking to the wrong sensor or motor. Make sure that the sensor number and motor letter match to the corresponding number on letter on the GoGo board. Figure 11 shows the GoGo widget and example of different motor settings and sensor readings. If the code seems correct, but the motor doesn't work correctly, first make sure the run light on the board is on. Then, make sure to click the red "Write to GoGo Board" button at the top of the tinker page. This ensures that the board received your code. Finally, to run the code, either click on the "Run" button on the tinker page or press the Run button on the GoGo board (located near the mini USB port).

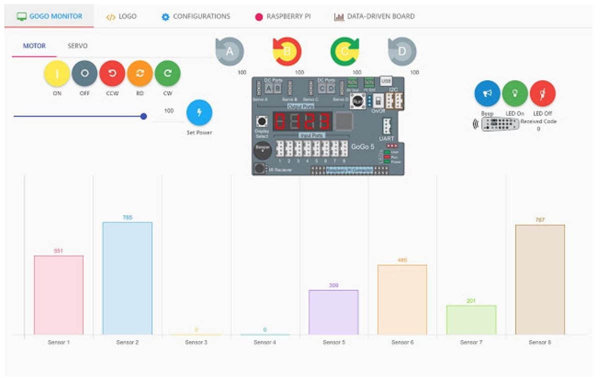


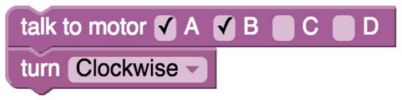
Figure 11 the 4 circles in top center represent the motors – arrows indicate the direction the motors are turning and red motors will turn in the opposite direction of green arrows. The colorful bars at the bottom show sensor readings.

2. Programming Errors

There are many common programming pain-points that could occur during the duration of this lesson. The primary issue that would be expected would be due to varying experience levels with programming. Luckily the Snap!-based language guarantees that no syntax errors can occur, but misconceptions can still occur with the loop and if-statement logic especially if students haven't been thoroughly exposed to these concepts. Doing a short demo and having a working code sample (with limited functionality) projected on the board would go a long way to assisting students with these more fundamental issues, as the structure of Snap! lends itself to very natural reading and understanding of the codebase. Other programming issues will likely be specific to the GoGo board hardware, such as forgetting to set the motor power before telling it to rotate. These issues can also benefit from a working code sample, as well as case-by-case teacher assistance.

3. Hardware Errors

Other common errors that can occur involve the actual connection and communication of hardware devices. Recognizing which ports different motor or sensor wires go into and their corresponding output levels may be initially unintuitive. Once again, a introductory demo would go a long way to smoothing out these misconceptions. Having students follow along and all get to a certain point, such as getting the motor working and responding to a sensor input, would be a great way to ensure these misconceptions are addressed early on. Having spare parts for sensors or motors that may be dysfunctional is also essential to avoid critical errors in the lesson. The most common problem when building a car is forgetting to make the motors turn in the same direction. This has a simple solution that students will often figure out on their own. When connecting two motors, make sure they are facing the same direction and in the set the both motors to either turn clockwise or counterclockwise as can be seen in figure 10.



```
talk to motor ✓ A ✓ B ☐ C ☐ D
turn Clockwise
```

Figure 10 shows how to ensure both motors are turning in the same direction

5E Lesson Plan

Cal Teach Student Names: Bijal Patel and Rodney Pimentel

Mentor Teacher Name: Ira Holston **School/Room#:** Berkeley High School Rm. H205

Grade Level and Subject: High School Computer Science

Date/Time to be Taught: April 10, 2018 2:30-3:30, April 16, 2018 1:30-2:30

Lesson Source(s):

What curriculum sources inspired and informed the lesson design?

This lesson was inspired by Professor Paulo Blikstein's paper "Travels in Troy with Freire" and the lecture he gave in our class.

Focus/Essential Question:

What is the focus or essential question that the lesson will address?

How can we use basic programming, motors and sensors to create an autonomous self-driving car?

Student Learning Objectives:

What 3 to 5 learning outcomes could be demonstrated by students as a result of the lesson's Exploration, Explain, and Elaborate sections?

1. Explore the GoGo board and accessories to learn how to use sensors and motors.
2. Combine code and engineering to build a self-driving car.
3. Collaborate with other students to guide their car through an obstacle course.

CCSSM or NGSS Standards:

List 1 or 2 standards that will be addressed by these objectives and experiences.

HS-PS2-3. **Apply science and engineering ideas to design, evaluate, and refine a device that minimizes the force on a macroscopic object during a collision.*** [Clarification Statement: Examples of evaluation and refinement could include determining the success of the device at protecting an object from damage and modifying the design to improve it. Examples of a device could include a football helmet or a parachute.] [*Assessment Boundary: Assessment is limited to qualitative evaluations and/or algebraic manipulations.*]

Student Prior Knowledge:

What prior knowledge or skills related to this lesson do you expect students to have?

Students will need to have a basic understanding of orientation (clockwise/counterclockwise) and be able to read and interpret bar graphs. They will also need to understand basic logic operations such as if/else do commands and repeat loops. Minor programming experience can be beneficial.

What possible gaps or misunderstandings might exist within this topic?

Some misunderstandings may occur during the implementation part of the lesson. Since most students will never have used a GoGo board before, they may struggle with understanding where the wires are supposed to go and how the motors work. There may be some confusion about how the coding part works as well because the students are used to coding in a programming language while the GoGo Widget uses Snap!-Like structure for programming. People may misunderstand the interface or navigation within the Chrome application.

Lesson Agenda for Your Students:

Create a short list of bullet points that you will post on the board to guide students.

1. Explore the different commands for motors and sensors
2. What is a forever loop? Why may it be important?
3. What sensors are best for this project?
4. Can you make the car move forward and stop?
5. Can you make the car turn?
6. Test your car on the obstacle course. What parts did your car do well? Where can you improve the code?
7. Fix some of the errors that you may have noticed and retest it!

Lesson Rationale:

How will these specific activities, materials, or technology experiences support progress towards the learning objectives?

Students will be able to learn how to use their code in the real world. Through these activities, they will explore the GoGo board and the materials provided in the kits, which will allow them to use the GoGo board to interact will tangible machines, such as the motors. Instead of telling them how to write their code, they will use trial and error to see the effects of their code in real time by testing their cars on the obstacle course multiple times. This will allow students to experience an iterative design process which closely models the research and development process common in the real world.

Materials and Technology List:

What materials will the students need during the lesson? (Describe total amounts and plan for packaging for easy distribution to students or groups.)

Materials per Group:

- GoGo Board
- 1 proximity sensor
- 2 motors
- 2 wheels
- 1 sensor wire
- 4 motor wires
- A computer with the GoGo Widget downloaded on it and internet access

General Materials for the Class:

- 1 roll of tape
- Cardboard (approx. one 18" x 12" piece per group)
- 10-12 pairs of scissors
- 8-10 GoGo Kits

Preparation Tasks:

We will split the class into 8 groups, making sure that there is at least one computer with the GoGo Widget downloaded on it per group. We will also check that the boards are running by plugging them into a computer and turning them on. The widget also has a "beep" button that can be used to cause the board to make a sound and check that it is connected.

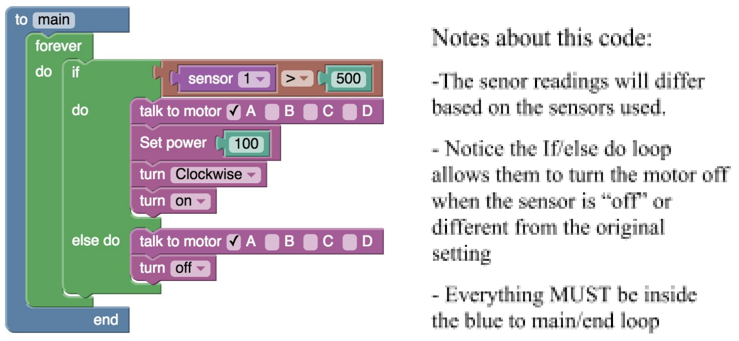
Safety Concerns:

The car may run into things if it doesn't stop. Make sure there are no delicate/breakable items in reach of the cars. Also make sure the cars are only driven on the ground to avoid having them fall off tables.

Lesson Title: Self-Driving Cars

Engage: *Activities that engage students' interest and build connections to their lives and prior knowledge.*	*Previous Experience and Baseline Learning*
- If you were given the tools/materials to fix any world problem what problem would you try to fix and why? - Some of the world's leading technologies fix a variety of random world problems. Self-Driving cars are one of the newer technologies that have polarized the public. Some claim that they are a great way to reduce human-caused accidents while others claim that it's dangerous to give a computer that much control over a task that could cost a lot of lives on the road. Despite this dispute, we will be trying to create a very basic self-driving car today.	*<u>Watch for:</u>* Using prior knowledge of Snap! Style programming language. Very basic understanding of computer programming to very skilled programmers. *<u>Response:</u>*

- What are some tools you would need to create a self-driving car? Why? Keep it as simple and minimalist as possible. - Today we will be using a basic circuit board, some sensors and simple motors to create a self-driving car that will be tested by sending it through an obstacle course. * This project doesn't require a lot of English background. However, if it becomes a hindrance, since this is group work, English language learners can bring their coding/engineering skills to help their group build or design the car. **Time**: <u>5 minutes</u>	Encouraging students who have experience with Snap! to help their teammates.
Explore: *Hands-on tasks designed to explore ideas and to develop skills together.*	*Focus, Involvement, Collaboration, Results, and Recording*
Each group will be given a GoGo board kit and a computer. Once the widget is open, students will plug their GoGo boards into the computer, turn it on and click on the beep button on the widget to ensure that the board is connected and running. Introduce the components of the GoGo board. The board has 2-pin white slots on the top that are labeled with letters. These slots are for connecting motors. The bottom left part of the board has 4-pin white slots for connecting sensors. The board also has a run button on the right side that can be used to run the program the code on the widget. This can also be run by clicking the run/stop button on the widget. Students will start with a basic task of using any sensor of their choosing to make a motor spin. (2 min) Once the students are given sufficient time to explore the sensors and have started to work on making the motor spin, display the correct code needed.	*<u>Watch for:</u>* Students trying sensors and playing with them to understand the data outputted in the widget. *<u>Response:</u>* Have you tried using more than one sensor? How can you optimize your code to use as few blocks as possible? How would you describe the function of the forever loop? What happens if you change the motor's orientation after you turn it on? What happens if your code is outside the main loop?



```
to main
  forever
  do  if  sensor 1 > 500
      do  talk to motor ✓A  B  C  D
          Set power 100
          turn Clockwise
          turn on
      else do  talk to motor ✓A  B  C  D
               turn off
end
```

Notes about this code:

-The senor readings will differ based on the sensors used.

- Notice the If/else do loop allows them to turn the motor off when the sensor is “off” or different from the original setting

- Everything MUST be inside the blue to main/end loop

Discuss the importance of a
forever loop: The GoGo board is continuously getting information from the sensor. Without the forever loop, it will only read the sensor once and perform the action in that small time interval. In order to make the board perform the actions continuously, the code requires a forever loop.

Now that the students learned how to make one motor run, their next challenge is to create a self-driving car with 2-4 motors that work simultaneously.
Students will now be given time to work on their cars.

Some tips for them to get started:

- The proximity or magnetic sensors will be the most useful

- Think about when the car should move and when it should stop

- How will the car turn? Are there different methods to do this?

- Will the car the car be a 2-wheel drive (i.e. just 2 motors in the front) or a 4-wheel drive (i.e. all 4 wheels have a motor)

Challenges students may face:

<u>Problem:</u> The car can’t move forward because the wheels are turning the same way.

<u>Fix:</u> Make sure the orientation of the wheels is set to go in the same direction and ensure that the motors are facing the same direction.

<u>Problem:</u> How to make the car turn?

<u>Fix:</u> There are several methods to achieve this. Students can turn the inside wheel backwards and

turn the outside wheel forward. Students can also lower the power of the inner wheel so the inner wheel turns slower than the outer wheel. These methods can be adopted for a four-wheel drive by applying the condition to both inside wheels. Problem: The students have written code but it doesn't run. Fix: Make sure the students click on "write to GoGo board" on the widget. If it still doesn't work, make sure the board is turned on. If it still doesn't work, there may be an issue in the code. It's important for students to have a forever loop, an if loop and ensure that they are talking to the motor and turning it on. Once all the cars are completed or once the 40 min are up, the groups will take turns testing their cars in the obstacle course. **Time:** 40 min	
Explain: *Students explain the phenomena they explored and discuss their different ideas and perspectives.*	*Participation, Reporting, Debating, and Evidence-Based Reasoning*
As the groups test their cars, they will explain the process they used to create their car Talking Points: - What and how many sensors and motors did they use and why? - Were there any unique aspects to their code? - Was it difficult to code? If so, why? And how did they solve it? - What kind of struggles did they face? How did they overcome those problems? - If there are any issues during the obstacle course, why do they think that happened? What are some potential solutions. Class discussion: Overall, what were key misunderstandings that hindered students? What were aspects that went well and were easy? If they had more	*Watch for:* Common misconceptions: the motors may all turn in the same direction by default, but they still need to set them to turn clockwise/ counterclockwise to make their project easy to replicate. *Response:* Why is it important to set the power and the orientation of the motors? What added functionalities would make this car better? (color detector – to check for signals, sound detector – to check for honks at the car)

time and resources what functionalities would they add and why? Though we only looked at one project that can be built using the GoGo board, there are MANY different projects that can use it. **Time:** <u>10 min</u>	Make sure everyone understands the for loops and sensor/motor code blocks and their purpose How did the students each use their unique skills together to make the car?
Elaborate: *Teacher-stimulated application and clarification of concepts, skills, attitudes, processes or terminology.*	*Demonstrated Understanding, Use of Skills, and Other Applications*
Show examples of past projects that have used the GoGo board. (Pictures attached) GoGo boards can be useful in ANY field. How might fields such as biology or history use a GoGo board? Key Take-Home Messages: - Technology is becoming an integral part of society and in everything we do. It's important to learn how to use it so that we can utilize it in the best way possible for any project we take up in the future. - Sometimes computer programming is daunting because it's a complicated language that isn't always intuitive to humans. That's why companies have created tools like the GoGo board to allow students like you to bring your innovative ideas to life. *This project doesn't require a lot of English background. However, if it becomes a hindrance, since this is group work, English language learners can bring their coding/engineering skills to help their group build or design the car. **Time:** <u>10 min</u>	<u>*Watch for:*</u> Students uniquely applying their experience to the real world. What are some real-world problems they could solve using the GoGo board? <u>*Response:*</u> How might technology be beneficial in a biology classroom? What are some sensors that lend themselves uniquely to such projects? (the soil humidity sensor, water sensor) What are some everyday problems that can be solved with the materials provided? (Put your phone on a car and make it roll away in the morning. Add lights and a light sensor to turn on lights when it's dark)

Samples pictures of past GoGo projects (courtesy of Paulo Blikstein):

Coffee Maker

Boxing Glove Controller

Water Purification System

Keyboard

Methods

We used cameras and phones as the main medium to gather data for our research. The bigger camera were set up at the front of the room* while we used our phones to record specific group work around the classroom. This allowed us to see the overall classroom dynamic as well as the collaboration that was happening between students. We also took pictures of the cars the students made as well as the code they worked on as a means to quantify the work they did. Lastly, we also used the data from the students' trial runs on the obstacle course to gauge how well they were able to complete the task. Throughout the lessons, we split our time between the As this lesson was taught in two different classrooms, each classroom had slightly different methods as outlined below. The benefits and drawbacks of each classroom are discussed at the end of the paper.

*The students who did not want to be filmed were on a different side of the classroom

One lesson was taught in an open environment where the students were given time to explore the tools given to them. The classroom was setup with one camera in the front of the class and we had an assistant using a phone camera to record student group work. The lesson started with a simple assignment that allowed the students to learn how to use the sensors and motors. First they were given instructions on where to connect the sensors and where to connect the motors. They were then asked to pick any sensor and use it to make a motor turn on. This required them to plug the physical components into the board and write the appropriate code to make the motor run. Once they were able to complete this task, we moved onto the actual lesson where we asked them to build a rudimentary version of a self-driving car. In this lesson, we did not have enough time to allow them to actually build or test the car, but they were able to understand *how* the car would be built. Additionally, to allow the students in this class to have a chance to do the second part of the lesson, our mentor teacher allowed us to continue the lesson on another day. We did not film this lesson, but we were able to take pictures of their cars and their code to analyze alongside the picture we took of the other class's work.

The second lesson was taught with a little more structure. The classroom was filmed in a similar way to the first lesson with a camera in the front and an assistant filming close ups of the group work. Given our experience from the first lesson, we realized that the audio was hard to hear. Therefore for this lesson, we also used a phone record the audio of groups talking through the problem. Unlike the first class, this class was given a GoGo board with the correct proximity sensor and motor attached to it. They were also provided with sample code that would allow them to use the proximity sensor to make the motors move. As the students did not have to spend as much time setting up the board as the students in the other class did, they were able to start coding their car right away. We also provided them with cardboard, tape, and scissors to build a physical structure around their board. Once they had a sturdy structure built, they were able to test their cars on a simple obstacle course we set up at the front of the classroom.

Looking back at the video, the camera that was at the front of the classroom didn't prove very helpful because it was too far away to hear an audio. Despite this, we were able to take note of how different groups were working across the classroom. The phone camera was a lot more useful because it was close enough to catch audio. It also had the added benefit of capturing our interactions with the students. Both these methods were helpful to us as researchers, however they have some drawbacks. The students were aware that they were being filmed and thus acted a little differently than they would have in a normal classroom setting. There didn’t seem to be any way around this because we wanted to make sure we have their content to film them and use their work for research purposes. However, it was an important fact to note while we were analyzing our data. Another small drawback was that students sometimes came up to the camera and made silly faces at it while we were working with groups in the back. Though this didn’t affect the recording, it was clearly distracting for the students to have a camera present while we were teaching the lesson.

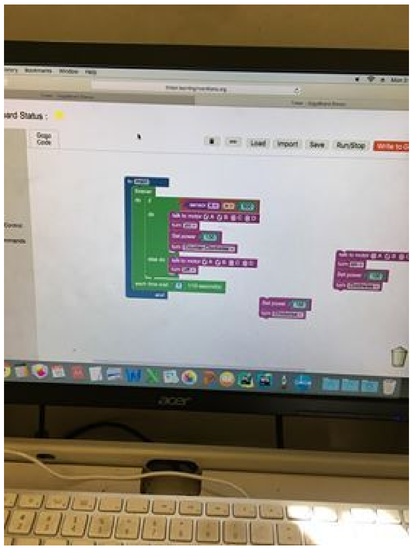
Results

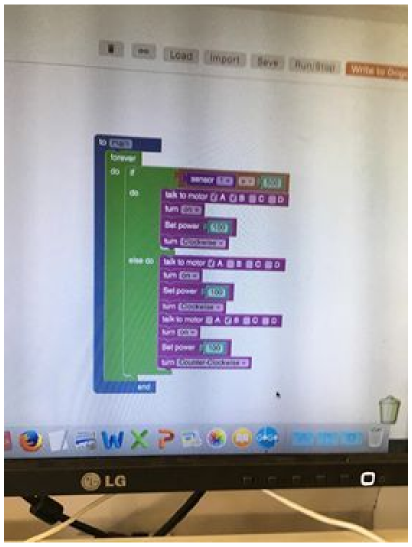
Through our research, we explored the discussions that happened and the work the students did on their project*. Going into the lessons, we hoped to use footage of the group work and the discussions between students as our main form of data. However, we found that a lot of the audio was hard to hear. Also, though we did our research over the course of multiple days, we were not able to record all the session. Thus, we relied on images of the cars they designed and screenshots of the code they implemented to compare the two lessons. With the limited audio we had, we also tried to use a few transcripts of the students' conversions to gain a better understanding of the discussions that took place in the classroom. By the end of our analysis, we weren't able to come to a clear conclusion. Nevertheless, we found both lessons to be engaging and got wonderful feedback from the students. One student from the first class (who did not want to be filmed) enthusiastically told me that she loved robotics and that's what she wanted to go to college for. Many students also expressed their disappointment at having to end the lesson.

*As the first class didn't have enough time to build the car on the first day, we extended the activity by one day. The data for this class is from the second day.

Samples of Code from the First Lesson:

Since the two lessons were taught differently, we wanted to compare the quality of the work that the students were able to complete. The first thing we compared was the code that the groups in each class wrote. Some of these examples are included below. Through our analysis of the code, we found that the first class, the one that was given more time to explore the different parts of the GoGo board, had simpler code than the second class. One of the biggest differences was that the code form this class had fewer if/else statements. As the obstacle course required multiple if/else statements, this was an indication that the students either did not completely understand the task or were unable to complete the assignment. The latter is a likely option because this class spent more time playing with the board, however, given that they had a second



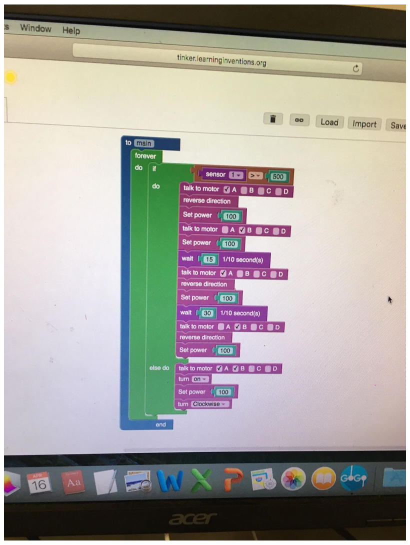


```
to main
  forever
  do  if  sensor 1 > 500
      do       talk to motor ☑A ☐B ☐C ☐D
               turn on
               Set power 100
               turn Clockwise
      else do  talk to motor ☑A ☐B ☐C ☐D
               turn on
               Set power 100
               turn Clockwise
               talk to motor ☐A ☑B ☐C ☐D
               turn on
               Set power 100
               turn Counter-Clockwise
end
```

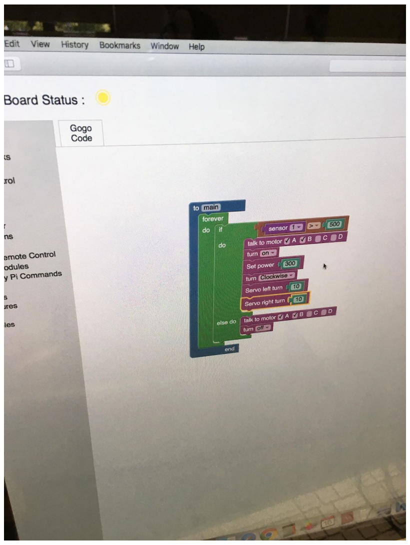
day to work on the cars, the former is also a likely possibility.

They only have one if/else loop and don't use the proximity sensor correctly

They used the proximity sensor correctly, but also uses only one if/else loop



```
to main
  forever
  do  if  sensor 1 > 500
      do       talk to motor ☑A ☐B ☐C ☐D
               reverse direction
               Set power 100
               talk to motor ☐A ☑B ☐C ☐D
               Set power 100
               wait 15 1/10 second(s)
               talk to motor ☑A ☐B ☐C ☐D
               reverse direction
               Set power 100
               wait 30 1/10 second(s)
               talk to motor ☐A ☑B ☐C ☐D
               reverse direction
               Set power 100
      else do  talk to motor ☑A ☑B ☐C ☐D
               turn on
               Set power 100
               turn Clockwise
end
```



```
to main
  forever
  do  if  sensor 1 > 500
      do       talk to motor ☑A ☑B ☐C ☐D
               turn on
               Set power 300
               turn Clockwise
               Servo left turn 10
               Servo right turn 10
      else do  talk to motor ☑A ☑B ☐C ☐D
               turn off
end
```

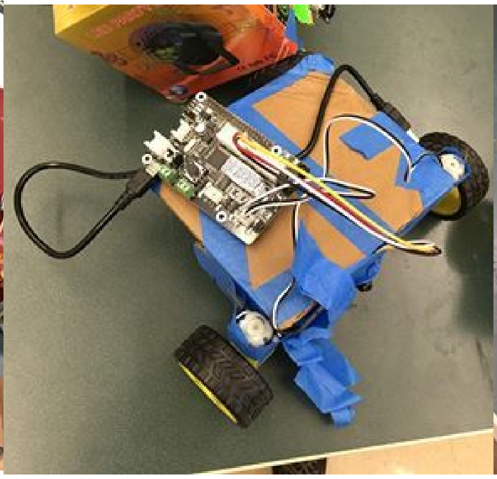
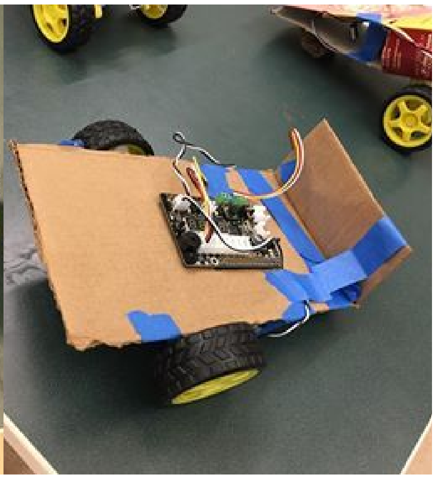
This seems complicated, but the code is

This code is well written, but doesn't reflect

repetitive and only includes one for loop	the complexity of the obstacle course

Samples of Cars from the First Lesson:

Another factor we took into account while comparing the two classes was the students' car designs and the length travelled on the obstacle course. Though the code from the first class did not have the if/else statements we expected, we were surprised to see that many of the cars actually made it pretty far through the obstacle course. On second glance though, we realized that many of the cars didn't go through the course like we thought they would. (i.e. We expected the cars to hit an obstacle and turn to avoid the obstacle, but some cars started a slant and were able to clear multiple obstacles without turning.) We never stated that the cars couldn't do this however, so we saw this more as an innovative spin to the project rather than a misunderstanding. While there was a lot to notice in the students' code and the distance travelled on the obstacle course, the cars were surprisingly similar. Both classes had simple car designs with a support for the GoGo board and a place for the proximity sensor. However, we did notice that some groups in both classes placed the proximity sensor in the back instead of the front, causing the car to move backwards.

There were a variety of designs, but over all they were all very simple.

Sample Conversations from the First Lesson:

Here are several conversations within groups from the first class, transcribed from our audio data. Names have been reduced to their first initial for confidentiality.

[Group 1]

A: Can you pass me the buttons? What if we used that as our sensor and press them to turn?
S: That could work, I think this [proximity] sensor could also work though
K: Yeah the proximity sensor would be more fun too
A: Yeah that'd be cool if it could drive itself, let's do that.
S: Which slot should it go into?
A: You can put it in slot two, we can test the other sensors at the same time.
S: Sounds good!

[Group 2]

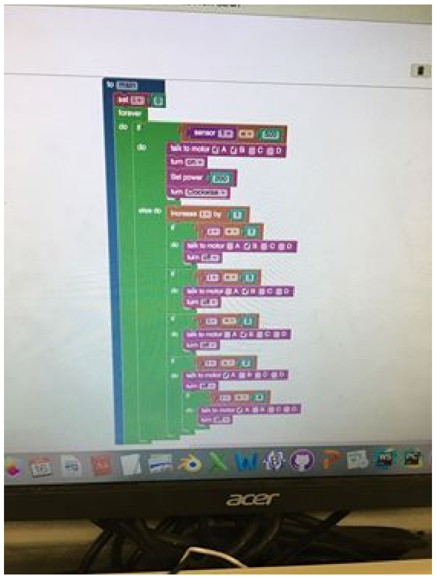
B: What does this block do?
C: Oh I know, it sets a variable so we can use it later on
B: Oh cool, we could probably use that right?
C: Yeah we can use it to keep track of what obstacle we're on, I think we're going to need to turn different ways around each one.
B: Yeah that sounds good. What about this one?
S: It says while else, I think it loops inside the while part for a certain number of times
C: Yeah it's until that thing is false [points to boolean condition]
B: Ohh, got it.

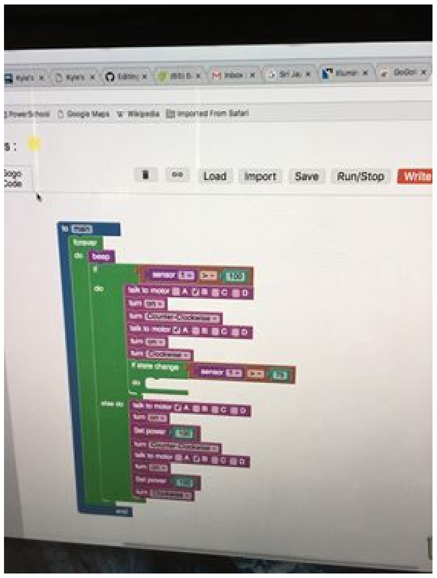
Both of these conversation show a lot of understanding about the process. The first group was discussing which sensor to use to build their car. Through their conversation, we can see that they were able to play around with the sensors. As the first part of the lesson involved setting up a simple system to make a motor move, many groups chose to use sensors similar to a button because that was the easiest to use. It only has two states: pressed and release, so the sensor readings were easier to work with and harder to get wrong. This group must have used a button in the first part, so student A suggested using it again to build the self-driving car. However, as student S points out, the proximity sensor is a more useful sensor for the second part. A subtle part of the conversion we noticed was that the students decided to put the sensor into the second slot, implying that there must have been a sensor in the first slot. This was clever because this group chose to use two sensors instead of one. Incase the proximity sensor didn’t work, they had a button as a backup to make the car stop if needed.

The second group was recorded while they were discussing the coding part of their car. Here the students can be seen exploring the different command on the widget. Before the lesson, we identified specific categories of command that would be useful to them, but encouraged them to explore the other categories. Surprisingly, the commands these students were talking about were not listed under the categories we suggested. This implied that this group went out of their way to learn some of the other commands and explore all of the options available to them.

Samples of Code from the Second Lesson:

The second lesson was different from the first in that the students were provided a board with the motors and correct sensor attached from the beginning. They were also provided with sample code to get them started. Though the rest of the lesson was exactly the same, we found there to be a significant difference between the two classes. As stated above, the first thing we compared were samples of the code from both classes. These images, shown below showed a lot more complexity than the samples of code from the first class. There are a lot more if/else

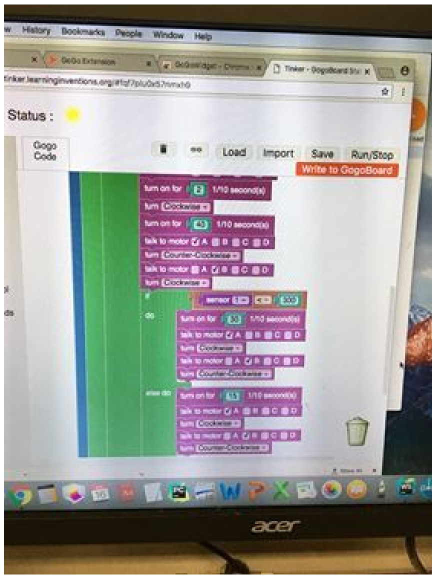


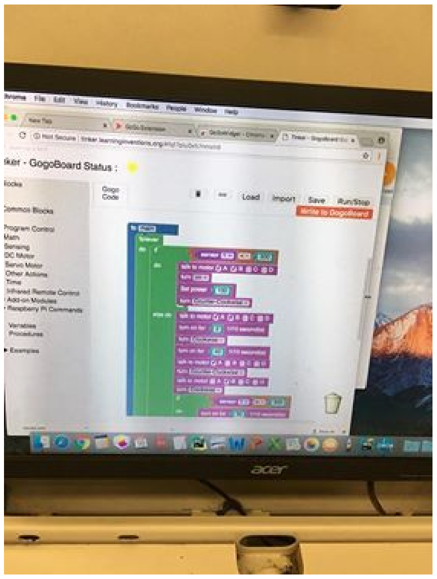


statements, which showed a better understanding of the obstacle course.

Good use of multiple if/else statements and used the turn command given

Only one extra if/else statement, but it seems incomplete rather than incorrect





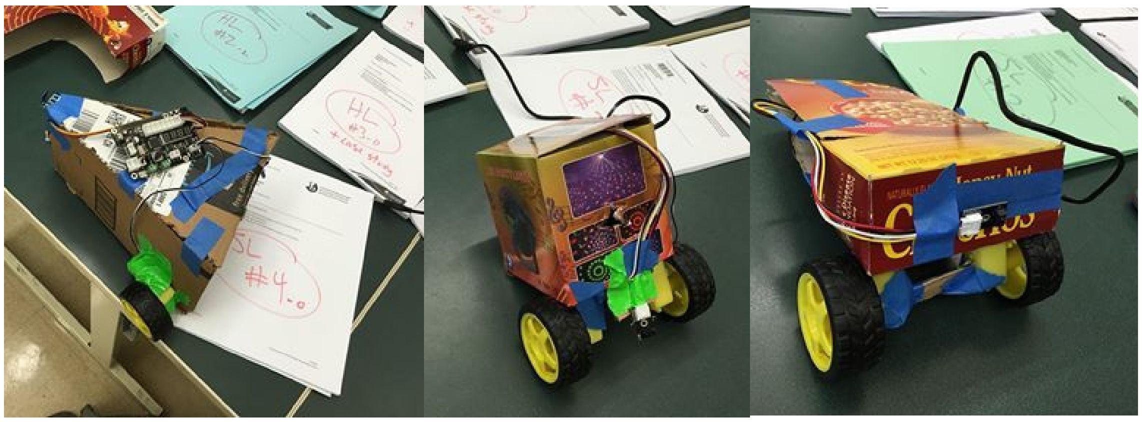
[Snippet of the full code] Shows a lot of

[Snippet of the full code] Shows multiple if/else

complexity, but was a bit too much	statements and an understanding of the course

Samples of Cars from the Second Lesson:

The second factor that we considered was their car design and the distance they travelled on the obstacle course. Unlike the first class, this class was able to work on making the turns necessary to go through the course as intended. This showed a lot more understanding of what the second half of the project. However, when the students went to implement the turns, a lot of times, the cars turned too much or missed the obstacle. We also saw more cases of the cars going backwards than we saw in the first class. Often times this was caused by the car spotting a obstacle, turning too far and then trying to move in the wrong direction. While this was more of a minor error on the part of the code, many students tried to just pick up their car and physically turn it so that it was facing the right direction. This small fix was actually a big mistake and showed a flaw in their understanding because when they turned the car around, the proximity sensor would be in the back and thus, it wouldn't be able to detect the obstacles that were in front of the car. Despite the difference between the two class's code, their cars looked very similar. The cars seem a little more structured in the picture, but since this class was able to build first, they were able to use the box like pieces of cardboard that the other class didn't have. In terms of design, both classes had relatively simple designs, as the project didn't require much more.



There were a variety of designs, but over all they were all very simple.

<u>Sample Conversations from the Second Lesson</u>

<u>[Group 1]</u>

V: How do you guys think we can turn around the first obstacle?

S: We could turn one wheel backwards, that'd make the car spin right?

V: I think that might turn us backwards though.

S: You're right. Let's make one wheel go faster than the other, we can set the power lower for the left motor or something

V: Sounds good, let's try it out. If we can get past the first obstacle then the rest shouldn't be bad

[Group 2]

T: I think the wheels are crooked, is there more tape?

C: I'm not sure tape can fix it, what if we cut holes in the cardboard so they can't move?

D: Yeah I think that should work, where're the scissors?

T: [Finds scissors on table] I'll work on this, keep working on the code

C: Sounds good, as long as our tires are straight everything should work to get at least halfway down the course.

Both of these conversations focus more on the obstacle course than the conversation from the first class. These conversations took place a similar time in this class as the conversations from the last class. However, the students in the first class weren't at the point where they could test their cars on the obstacles course at this time. We chose to compare these conversation despite this because it's a good reflection of how far each class got.

The first group was discussing how to make the turns once the cars came close to an obstacle. In this conversation, student S makes the observation that they could move one wheel differently than the other. This conversation was unique to this group because most other groups found a turn command on the widget. As this is what we intended for the students to do, this was a really important discussion. (We didn't even realize that there was a built in turn command, and thus were very surprised to see so many students use it). Unlike the first class, this group showed a better understanding of the obstacle course and the work they needed to do in order to traverse it. This observation is important because it showed that the students learnt a lot more in the second class. However, this is not necessarily better or worse than the first lesson (as discussed in the Conclusion).

The conversation from the second group takes place while they were building their car. Though there wasn't much variety in the cars, small details such as the alignment of the wheel could affect the car's performance as this group had realized. Many groups did not give much thought into the design of the car, however since the races through the course were all really close, sometimes these small details made a big difference. Not only was this a way for the group to get ahead on the course, but it also showed a deeper understanding of the mechanics of the car. Students in both classes focused heavily on the code, which is not surprising because they were a computer science classes. However, the mechanics were also an important part of the lesson, and it was interesting to see some students take interest in that.

Conclusion

Through our research we found that the open ended structure of the first lesson promoted more enthusiasm for the topic. This excitement encouraged classroom discussion about topic beyond the specific assignment. We also found that these students had a better understanding of the board, how to connect the sensors and how to read the different outputs. However, allowing them to explore took away precious time from the given assignment as the students didn't have enough time to actually build the car or test it on the course. When the students were provided a basic structure at the beginning, they were able to focus on the task far better. This allowed them to take ownership of the project and thus fostered enthusiasm for the activity and there was more motivation to succeed on the obstacle course. The code for these projects were a lot more developed and showed a better understanding of the activity. Though both lessons provided different opportunities for the students, the student discussions both played a key role in their understanding. However, the exploratory class's discussions reflected a deeper understanding of the robotics as they were able to connect the sensors and motors themselves [see First Class, Group 1]. The exploratory class also set aside time for students to share knowledge about topics like Snap! programming tips [First class, Group 2]. RJ's class didn't have those same constructive conversations, however they were able to focus far more on actually constructing the car and completing the obstacle course. Discussions focused primarily on specific implementation details such as turning the car or deciding how to respond to sensor input [Second class, Group 1]. A lot of time in this class was also spent on construction details such as fixing the wheels in place [Second, Group 2]. Although the starter material may not have allowed the same level of breadth and curiosity as the exploratory approach, these conversations reflected the fact that it helped keep students focused on actually completing the lesson.

In the end, there are unavoidable tradeoffs to either of our lesson styles. However, our lesson comparisons showed that constraining students' choices and giving them less time for exploration can help dramatically with classroom efficiency while reducing the amount of understanding of underlying concepts. In our example, giving students time to explore seemed to generate more understanding and enthusiasm for the technology itself, but didn't give them enough time to tackle problems related to the programming and construction of the actual vehicle like RJ's class. Understanding these pros and cons to each approach can help guide teachers in maintaining the delicate balance between exploration and efficiency.

Reflection (Bijal)

As I found through the research, the exploratory lesson was a little long and thus the students were unable to actually build a car. If I were to do this lesson again, I think I would design it to take 2 days. Both the exploratory stage of the process and the building stage were important and it’s not feasible to try to do both in one 50 minute period. I also found that the videos we took did not cover as much student conversation as I would have like. Next time, I would focus on videotaping more student interactions and use a voice recording device to get clearer conversations.

Reflection (RJ)

Although I would have loved to give the students the same opportunity to explore with the GoGo board as I had, I believe my lesson went very well considering the time constraints. There were many impressive designs, and although the students didn’t have as much understanding of how the sensors interfaced with the board, the starter code and GoGo board with motors/proximity sensor already attached allowed the students to invest more time into the actual coding and construction of the cars. Next time, I would try to incorporate the range of other GoGo sensors and potentially broaden the prompt to include non-vehicles.

References

Blikstein, P. (2008). Travels in Troy with Freire: Technology as an agent for emancipation. In P. Noguera & C. A. Torres (Eds.), Social justice education for teachers: Paulo Freire and the possible dream (pp. 205-244). Rotterdam, Netherlands: Sense.

Goos, M. (2004). Learning Mathematics in a Classroom Community of Inquiry. *Journal for Research in Mathematics Education*, *35*(4), 258-291.

Freudenthal, H. (1971). Geometry between the devil and the deep sea. Educational Studies in Mathematics, 3(3/4), 413-435.