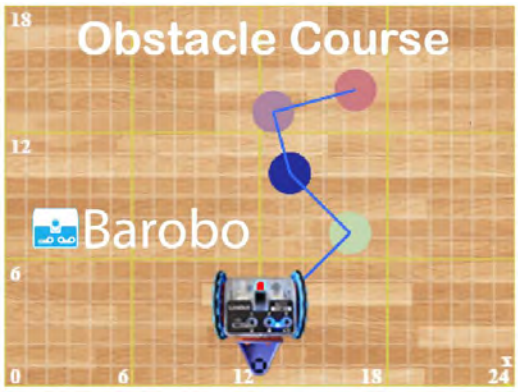
Barobo

Linkbot Obstacle Course: An Introduction to Coding and Robotics

Lesson Plan and Teacher's Notes for an Hour of Code™ Activity



https://www.roboblockly.com/curriculum/hourofcode/
info@barobo.com
http://www.barobo.com
Barobo, Inc.

Outline

A. **Introduction**
B. **Overview of the Activities**
C. **How to Use Barobo's Hour of Code™ Activities in your Classroom**
D. **Before Hour of Code™**
E. **Getting Started in Class**
F. **Wrapping Up in Class**
G. **Linkbot Obstacle Course Activities**
H. **Connecting Hardware Linkbots to the Computer and Initializing Them in RoboBlockly**

A. Introduction

Guide Linkbot through an obstacle course! In this series of activities students will learn how to control a Barobo Linkbot robot and help it navigate successfully on a journey around various paths. They will also learn some basics of computer programming, including the very useful and powerful programming concept of loops. The activities are suitable for beginning students in grades 4 and above.

Students will use the drag-and-drop RoboBlockly interface at www.roboblockly.com (no registration required) to program the Linkbot to maneuver through the obstacle courses. The

Linkbot may be either an online virtual robot or an actual hardware robot, which provides more hands-on and engaging practice for students. After the Hour of Code™ students may continue to explore the many coding and math-related activities available at www.roboblockly.com.

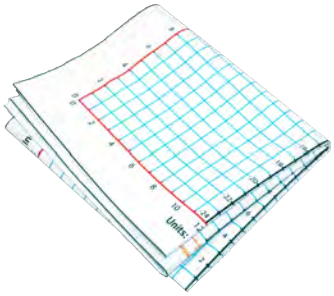
Learning Objectives

1. Learn the basic terminology and concepts of computing and controlling robots, including the concepts of code blocks, commands, arguments (input values to commands), run/execute, bugs, debugging, and repeat loops.
2. Gain practice with the mathematical concepts of the number line, two-dimensional x-y grids, right angles, and squares.

Equipment and Software Needed for Each Student or Pair of Students

1. Computer (PC, Mac, or Chromebook) with internet connection.
2. Web-based RoboBlockly interface at www.roboblockly.com to control the virtual (or hardware) Linkbot robot and run the activities (no registration needed).
3. Optional: Hardware Linkbot robot and either USB cable or wireless dongle, plus Linkbot Labs software to connect hardware Linkbots to the computer--free download from https://www.barobo.com/downloads. Hardware robots available at https://www.barobo.com/shop. Linkbot Super Kit recommended:

Activity mats:



B. Overview of the Activities (details in Section G below)

1. Welcome to Linkbot: Driving Forward
2. Moving Forwards and Backwards
3. Obstacle Course: Turning and Debugging
4. Traversing a Square, Part 1
5. Traversing a Square, Part 2 (Loops!)
6. Obstacle Course Challenge!

C. How to Use Barobo's Hour of Code™ Activities in your Classroom

It will take students about an hour to get through each set of activities. However, we believe that students should be able to learn at their own pace, and so we encourage you to give students additional time if needed to complete the activities or make it clear that they don't need to finish the entire set of activities. The final activity is open-ended if there are students who finish sooner. When using hardware Linkbots time should be budgeted for setting up and connecting the equipment (usually 3-5 minutes for an individual student or 5-10 minutes for a group).

D. Before Hour of Code™

1. Prepare your classroom

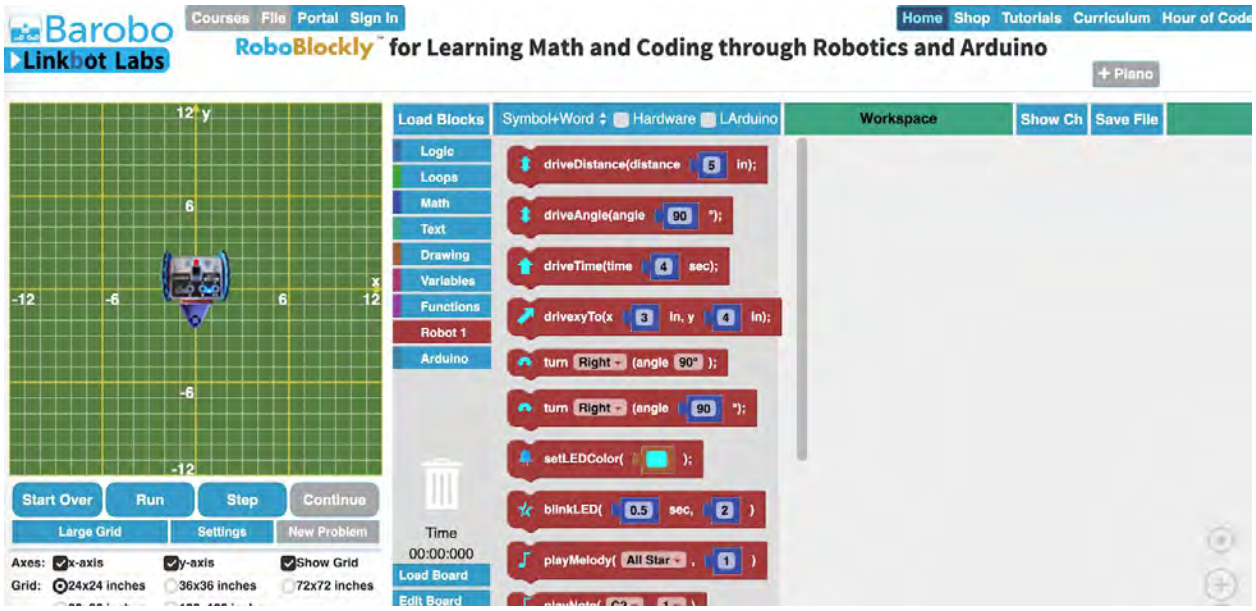
- Make sure you have a computer (PC, Mac, or Chromebook) for each student or pair of students. (Tablets are suitable for doing the activities using virtual robots, but not for hardware robots.)
- Make sure you have good Internet access in the classroom, as you will need to access the RoboBlockly activities at www.roboblockly.com to control the virtual or hardware Linkbot and run the activities (no registration required).
- Make sure you have a modern browser installed on the computers or devices. Test RoboBlockly on students' computers.
- If you are using hardware Linkbots (available individually or in classroom bundles at https://www.barobo.com/linkbot-and-accessories), make sure to have enough for each student or pair of students, and connect them to the computer and test them beforehand using either USB cables or wireless dongles. (Instructions are listed in Section H below and also available here: https://www.barobo.com/faq-linkbot-labs-and-linkbots.)
- Hardware Linkbots require the use of Linkbot Labs software to connect them to the computer--free download is available for installation from https://www.barobo.com/downloads.
- If using hardware Linkbots, test each of them beforehand to ensure there are no connection issues (e.g., firmware update needed). Instructions here: https://www.barobo.com/faq-linkbot-labs-and-linkbots. For firmware issues: https://www.barobo.com/faq-troubleshooting.
- The activities do not use sound, but audio commands that enable a robot to beep or play music are available in RoboBlockly. So you may want to provide headphones or ask students to mute their speakers.

2. Prepare yourself

- Go through the Hour of Code activities yourself so that you are familiar with what your students will be experiencing.
- Each activity has an introduction that describes the activity and concepts involved, and a problem statement that instructs students on what they should do. (Full details in Section G below.) These are designed to be self-contained and self-explanatory, but some students may need clarifications or other assistance.

3. Prepare your students

- When using RoboBlockly in class, first demonstrate to students how to navigate and use the RoboBlockly website, as shown below. (Your view of the website may be slightly different, depending on the activity loaded.) The basic idea is that users create programs to control the robot by dragging code blocks (commands) from the middle Blocks section to the Workspace on the right. Code blocks can be clicked together to form a sequence of commands. Clicking the Run button below the grid on the left will then run ("execute") the commands in sequence, with the virtual robot on the grid showing the results (as well as moving a hardware robot, if connected).



- Introduce dual programming for collaborative learning. This learning model is especially helpful for students who may need some extra assistance.
- Help students get excited about Hour of Code by inspiring students and discussing how computer science impacts every part of our lives. As a class, list things that use code in everyday life, or discuss different ways technology impacts our lives, etc.

E. Getting Started (10 minutes for intro, 40-45 minutes for activities)

1. If using hardware robots, guide each student (or pair of students) in connecting the Linkbot to the computer via a direct USB connection or wireless dongle, and using the Linkbot Labs sidebar at www.roboblockly.com to enter the individual Linkbot's ID number and establish the connection (5-10 minutes).
2. Whether using hardware or virtual robots, going through the first activity as a group is a good way to start. The RoboBlockly website contains far more options than are used in the Hour of Code activities, so the most important features to cover are:
    a. Terminology: program, code, code blocks, commands, arguments (input values to commands), run/execute, debugging
    b. The basic layout: the grid, the middle code block section, the Workspace
    c. Important buttons: Run, Reset (label that appears on the Run button after Run is clicked), Step, Start Over, the Show Robot checkbox (in the "Robot 1" section at the lower left).
    d. Basic actions: running pre-placed code blocks, changing the argument values (input values) for commands, dragging and dropping code blocks from the center block section to the Workspace, unhooking code blocks from other blocks in the Workspace and dragging them back to the center block section to delete.
3. Point the students to the rest of the activities at https://roboblockly.com/curriculum/hourofcode/obstaclecourse/ (40-45 minutes).
4. Circulate around to offer encouragement and give assistance as needed.

F. Wrapping Up (5 minutes)

- If you and/or your students use social media, we encourage you to share your Hour of Code experience (as appropriate). For example, "I've done an Hour of Code obstacle course challenge with @linkbots robots! Have you? #HourOfCode #robotics4mathlearning." Use the hashtag #HourOfCode (with capital letters H, O, C).
- After Hour of Code, encourage your students to continue learning to program on RoboBlockly by exploring the Playground, Robotics, Computing, or grade-level Math activities.
- ***All student activities on RoboBlockly are free - students do not have to create accounts!***

G. Linkbot Obstacle Course Activities

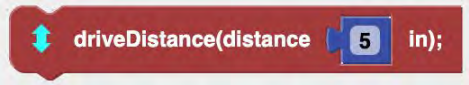
This set of activities introduces students to the basics of computing and controlling robots, including the concepts of code blocks, commands, arguments (input values to commands), run/execute, bugs, debugging, and repeat loops. It also gives them practice with the mathematical concepts of the number line, two-dimensional x-y grids, right angles, and squares.

The following sections reproduce the online activity descriptions from www.roboblockly.com.

Activity 1. Welcome to Linkbot: Driving Forward

Initial prompt: In this introductory lesson, you will learn how to move the Linkbot robot a specified distance by using the driveDistance code block. You can use positive and negative numbers to specify the distance to move.

Lesson description: This Hour of Code lesson introduces you to the Linkbot robot. You can control its movements on the grid on the left side of the window by dragging and dropping various "code blocks," or "commands," from the middle "Block" section to the "Workspace" on the right and then clicking the "Run" button (located just below the grid). These commands are simply instructions that tell the robot what to do. Once the commands are in place, clicking the Run button will cause the computer to run or "execute" the commands in sequence, one after the other, thus activating and controlling the robot. We will get the hang of it by starting with just a single command that has already been placed in the Workspace for you--the driveDistance command:



To tell the robot how far to move, you enter a number into the blue box that is in the driveDistance command, then click **Run**. This number is called the "**argument**" for the command. To move the robot again, click "**Reset**" (which appears in place of the **Run** button after something is run), enter a different value for the argument, and then click **Run** again. (If you have a hardware Linkbot robot connected, the hardware robot will move in the same way as the virtual robot you see on the grid.)

Blocks used: driveDistance()

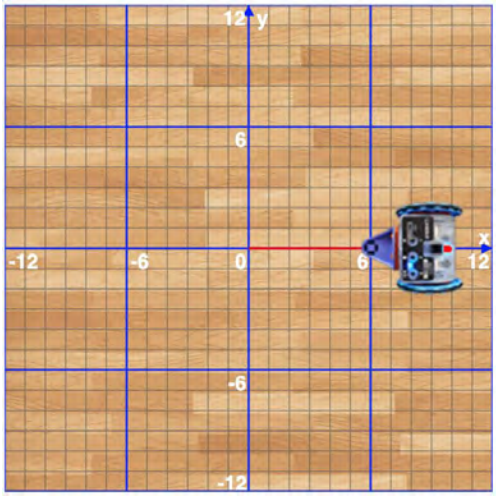
Pre-placed blocks: driveDistance()

Problem statement: Hello, I am a Linkbot robot. Please help me move forward 8 units (to the right, the direction I'm facing).

Hint: The preplaced code block drives the Linkbot 5 inches forward because the "argument" (the specified input value for the driveDistance command) is initially set at 5 inches. But the specified task is to move the Linkbot 8 inches, so change the value of the argument to 8.

Possible solution:

Solution image:


12
y
6
-12
-6
0
6
12
x
-6
-12

Activity 2. Moving Forwards and Backwards

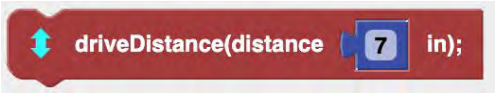
Initial prompt: In this lesson, you will learn how to use multiple commands to move the Linkbot forwards and backwards.

Lesson description: In our first lesson you learned how to move a Linkbot robot using a single driveDistance command. In this lesson you will learn how to use multiple driveDistance commands to move the robot along the number line. It will also give you a little practice with adding and subtracting numbers. To add a new driveDistance command to the Workspace, use the computer mouse (or trackpad) to drag and drop a driveDistance code block from the Blocks section in the middle to the Workspace on the right, and position it just under the pre-placed code block so that they click together. Repeat this process for as many code blocks as you need, and then change the arguments (input values) for each code block as needed. Then click the **Run** button, and the code blocks will be executed in sequence, one by one. You can also step through them slowly one at a time by clicking the "**Step**" button instead of the **Run** button.

Tip: To see the distance traveled more clearly, you can uncheck the "Show Robot" box in the "Robot 1" section (lower left) to hide the image of the robot.

Blocks used: driveDistance()

Pre-placed blocks:

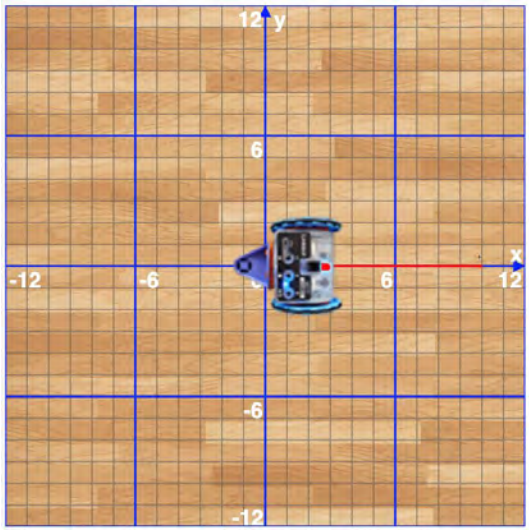


Problem statement: Starting at the origin (0), the Linkbot wants to travel along the number line in three segments. The first pre-placed driveDistance command moves it to the point x = 7. Add another command to move it to x = 10, and then one more to have the robot end its trip at x = 1.

Hint: This problem is equivalent to 7 + __ = 10 and 10 - __ = 1. To move backwards on the number line use a negative number.

Possible solution:

Solution image:


12
y
6
x
-12
-6
6
12
-6
-12

Activity 3. Obstacle Course: Turning and Debugging

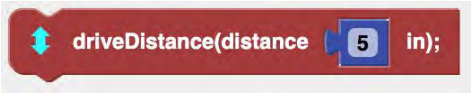
Initial prompt: In this lesson you'll learn how to use the turn command and guide the Linkbot through an obstacle course!

Lesson description: When we put together one or more commands to control the Linkbot robot, we are creating a "**program**." In other words, a computer program is like a recipe, a series of instructions to get something done. Often when we create a program we make mistakes, such as using the wrong command, or the right command at the wrong time, or the wrong numbers in the right command, or so on and so forth. These errors are called "**bugs**," and so we often need to "**debug**" the program in order to fix it and get it working properly. The **Step** button (just to the right of the **Run** button) is helpful in debugging, because it will step through a program's commands one at a time. At each step the command being executed is displayed in orange in the Workspace. This lesson gives you some practice in debugging, and also introduces the **turn** code block, which instructs the robot to turn left or right at a certain angle (90 degrees for a right angle turn):

Tip: If you want to speed up the robot's journey, add a **setSpeed** block at the beginning. (You may have to scroll down in the list of code blocks to find it.)

Blocks used: turn(), driveDistance()

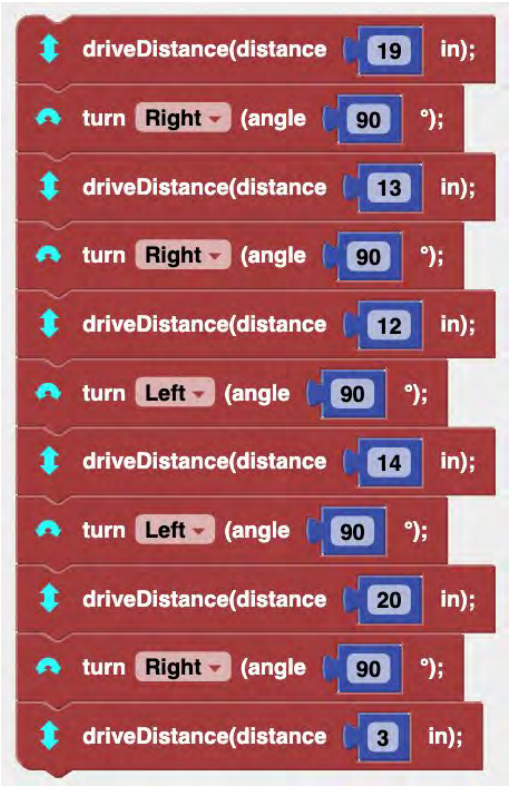
Pre-placed blocks:



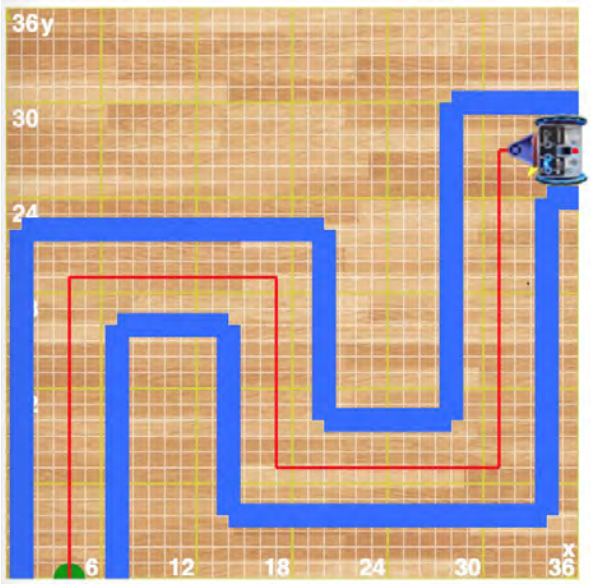
Problem statement: Modify the pre-placed block and add more **driveDistance** and **turn** blocks to drive the robot through the obstacle course to the star. Make sure you do not cross the tape!

Hint: Use the **Step** button to step through the commands one at a time. It will take a little trial and error (debugging) to get a good path.

Possible solution:



Solution image:



Activity 4. Traversing a Square, Part 1

Initial prompt: This lesson introduces the **traceColor** command and gives you practice in controlling the robot to trace out square shapes.

Lesson description: In the previous lesson we learned to use the **turn** command, so now it's possible for us to create programs that have the Linkbot travel along geometrical shapes. We'll try an obstacle course in the shape of a square, and to make things slightly more colorful, we'll use the **traceColor** command to change the color and width of the robot's trace as it travels along:

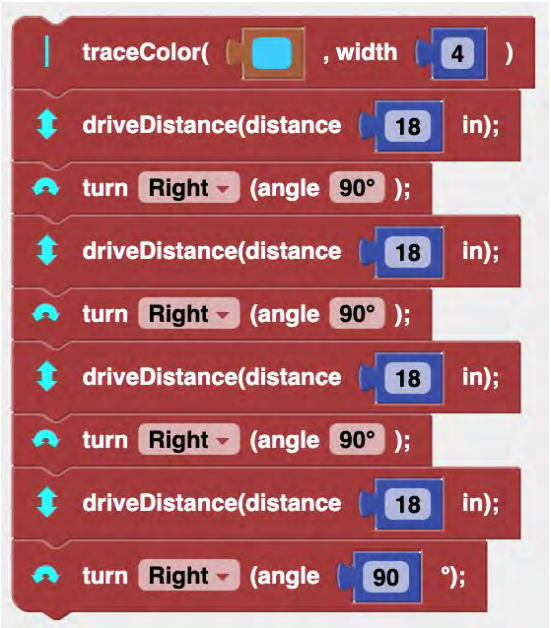
Blocks used: traceColor(), driveDistance(), turn()

Pre-placed blocks:

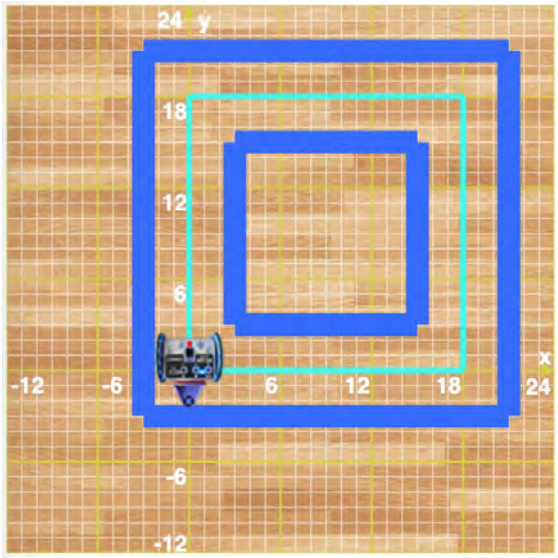
Problem statement: Add the necessary **driveDistance** and 90-degree **turn** blocks to have the Linkbot journey clockwise around the square and back to its starting orientation.

Hint: You should have four **driveDistance** commands and four **turn** commands.

Possible solution:



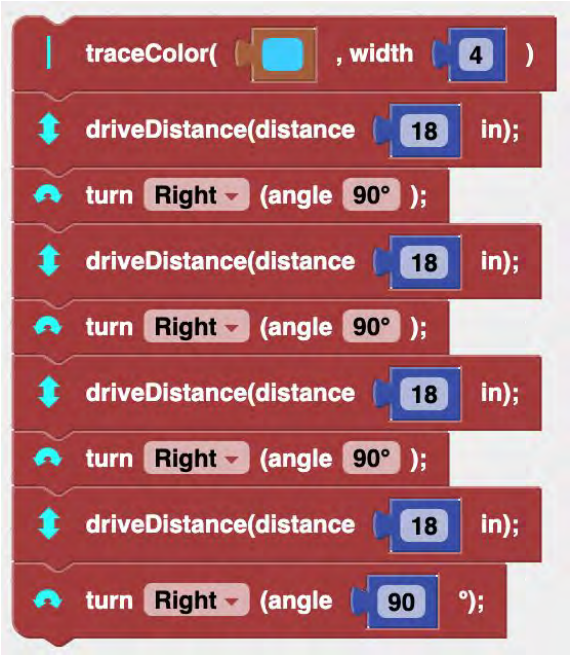
Solution image:



Activity 5. Traversing a Square, Part 2 (Loops!)

Initial prompt: In this lesson we'll learn about a better way of drawing a square, using the very important programming concept of loops.

Lesson description: In the previous lesson we learned how to use the Linkbot to trace a square. To do so, we used the code blocks shown below.



Note that we used the same driveDistance and turn commands four times in a row. That is, we instructed the Linkbot to drive 18 units, then turn right 90 degrees, then drive 18 units, then turn right 90 degrees, and so on for the third and fourth sides. In programming we often encounter situations like this, where we need to repeat the same code several times in a row (or even hundreds or thousands of times in a row). Instead of writing (or copying and pasting) the same commands over and over, we can use what's known as a "**loop**." For a basic loop, we put the commands we want to have repeated inside a special loop code block, and then specify how many times we want the commands to be repeated. This is known as a "**repeat loop**." (As you continue to learn about programming, you will come across other types of loops as well.) In this lesson you will get to experiment with a repeat loop to have the Linkbot traverse the square.

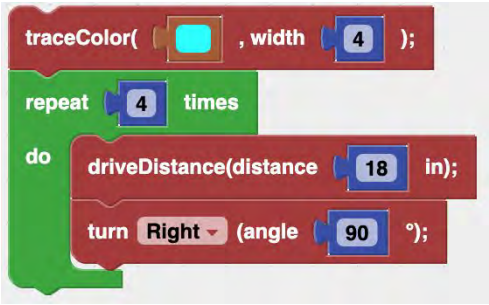
Blocks used: traceColor(), repeat(), driveDistance(), turn()

Pre-placed blocks:

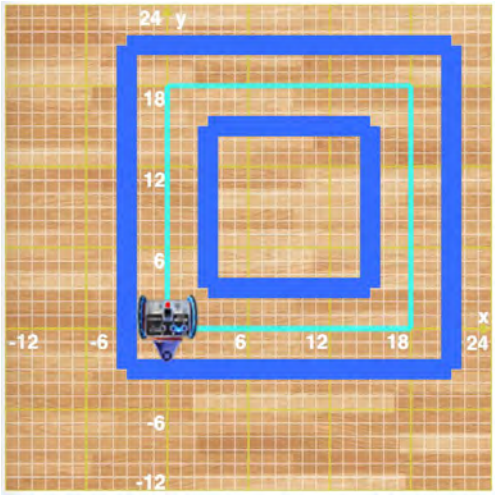
Problem statement: The pre-placed code blocks use a loop to draw two sides of a square. Use the **Run** button to run the code and see the result, and then **Reset** and use the **Step** button to step through the code piece by piece. Then change the value of the argument in the loop so that the code has the robot draw a complete square with four sides. What happens if you have the loop repeat more than four times?

Hint: To draw the four sides of the square, we need the loop to repeat four times.

Possible solution:



Solution image:



Activity 6. Obstacle Course Challenge!

Initial prompt: A new challenge: an obstacle course made of a random path.

Lesson description: Now that we've learned how to control robots to traverse square paths, we will introduce a new type of code block, drivexyTo, which allows us to instruct the RAV robot to travel in a straight line to the specified point (x,y) on the grid:

We'll use this in a new challenge: following randomly generated obstacle course paths! (We have hidden the image of the robot to make it easier to trace the path. The position of the robot is indicated by the small red bar. If you want to see the robot image, click "Show Robot" in the "Robot 1" section at the lower left.)

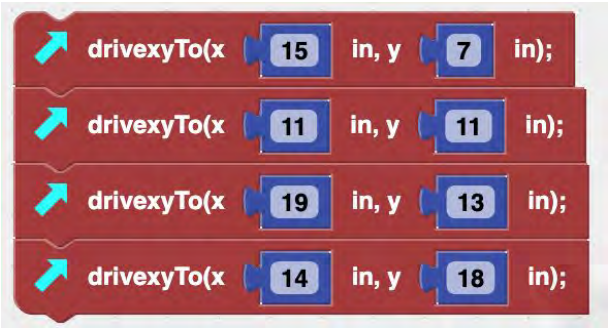
Blocks used: drivexyTo()

Pre-placed blocks:

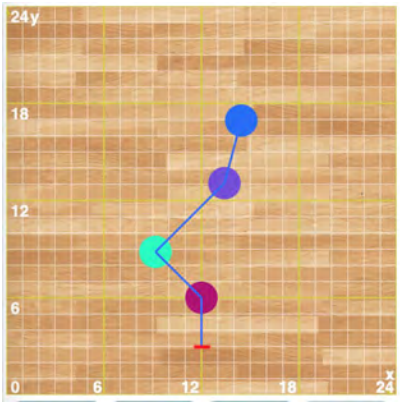
Problem statement: Use four drivexyTo blocks to traverse the path of the obstacle course! (Note that the robot starts at the point x = 12, y = 3.)

Hint: Don't forget that the robot starts at the point x = 12, y = 3.

Possible solution: Varies depending on the random obstacle course generated. Sample:



Solution image: Varies depending on the random obstacle course generated. (Sample course shown below, with red bar shown in place of robot image.)


24 y
18
12
6
0
6
12
18
x
24

H. Connecting Hardware Linkbots to the Computer and Initializing Them in RoboBlockly

To control hardware Linkbots you need to download and install Linkbot Labs software.

H.1. Instructions for PCs and Macs (instructions for Chromebooks follow; tablets are not supported)

(1) Download and install the latest version of Linkbot Labs from the Barobo Downloads page at https://www.barobo.com/downloads.

(2) Launch RoboBlockly at www.roboblockly.com using a regular browser (Chrome is recommended).

(3) Connect the hardware Linkbot to the computer using either a micro USB cable or a wireless dongle, and then add the Linkbot's ID (found on the Linkbot's label, e.g., "ZG81") using the Linkbot Labs sidebar on the left side at RoboBlockly, as shown in Figure 2 below. (Click on the "Linkbot Labs" arrow just below the RoboBlockly heading at the top left to open or close the sidebar, as circled in red in Figure 1.) Note that when first opening RoboBlockly it may take a few seconds for the Linkbot Labs arrow to appear.

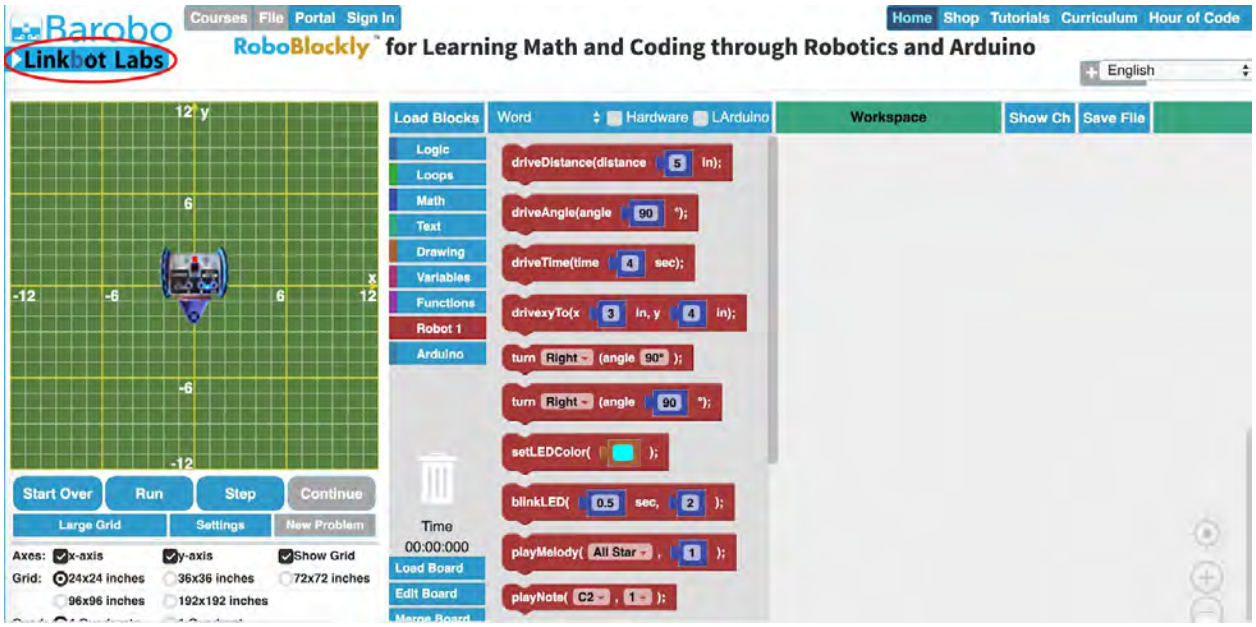


Figure 1: RoboBlockly user interface with the Linkbot Labs sidebar closed. Click on the "Linkbot Labs" arrow, as circled in red in the image, to open the sidebar (or close the sidebar, if open).

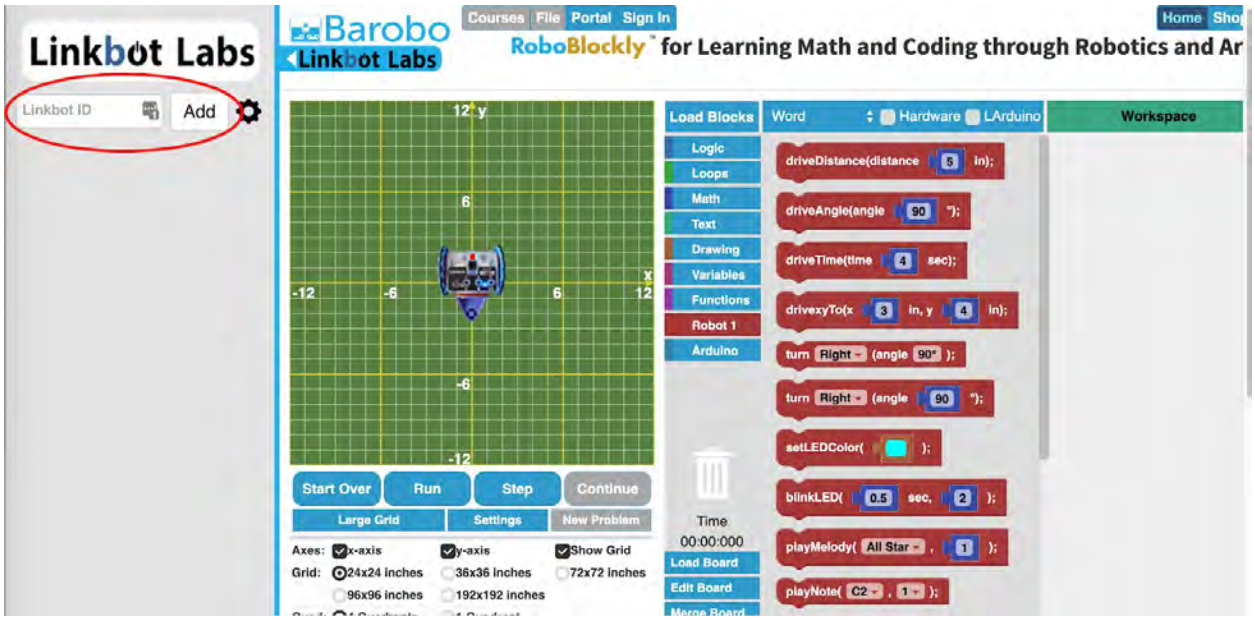


Figure 2: RoboBlockly user interface with the Linkbot Labs sidebar open. Enter the ID for a Linkbot in the box (circled in red) and click the “Add” button.

Instructions, continued:

(4) Drag a Linkbot instruction block such as driveDistance() to the Workspace area of RoboBlockly.

(5) Click “Run” or “Step” to move the virtual Linkbot on the grid and the hardware Linkbot at the same time.

(6) If the hardware Linkbot does not move, you may need to restart the Barobo Linkbot Service program by right-clicking on the “L” in the Windows task list at the bottom right of the screen (Figures 3 and 4 below) or clicking on the “L” in the Mac menu bar at the top of the screen (in the list of icons on the right side of the menu bar, Figure 5 below). Choose “Restart Linkbot Service” from the popup menu.

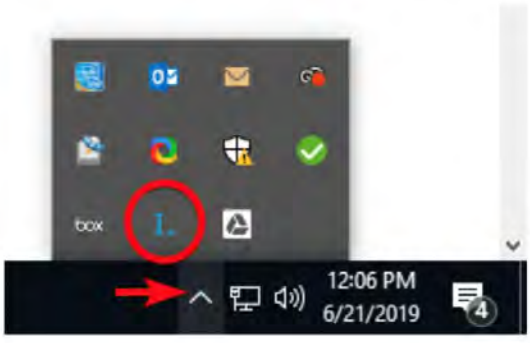


Figure 3: Accessing the Linkbot Service “L” in the Windows task list.

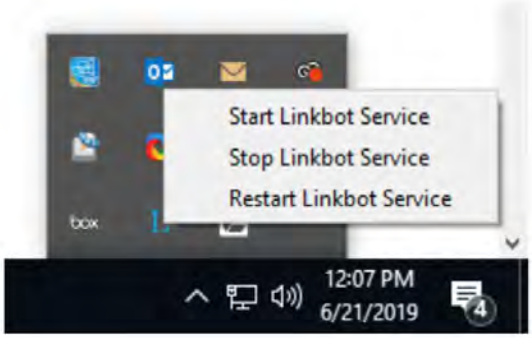


Figure 4: The popup menu with Linkbot Service options (Windows interface).

Figure 5: The Linkbot Service “L” in the Mac menu bar (your list of icons will be different). A similar popup window as for Windows will appear when you click the L.

Instructions, continued:

(7) If you receive messages about firmware needing updating, see https://www.barobo.com/faq-troubleshooting for guidance.

H.2. Instructions for Chromebooks

Download and install the Linkbot IDE (Integrated Development Environment) and Firmware Updater for Chromebooks, available via the Barobo Downloads page at https://www.barobo.com/downloads. Then open roboblockly.com in the browser and follow steps 3-5 in the instructions for Windows and Mac machines. If you receive messages about firmware needing updating, see https://www.barobo.com/faq-troubleshooting for guidance.